2024 34th International Conference on Field-Programmable Logic and Applications (FPL) | 979-8-3315-3007-5/24/$31.00 ©2024 IEEE | DOI: 10.1109/FPL64840.2024.00038

IMAGine: An *I*n-*M*emory *A*ccelerated *G*EMV Eng*ine* Overlay

MD Arafat Kabir*, Tendayi Kamucheka*, Nathaniel Fredricks*,
Joel Mandebi†, Jason Bakos‡, Miaoqing Huang*, and David Andrews*
*Department of Electrical Engineering and Computer Science, University of Arkansas,
‡Department of Computer Science and Engineering, University of South Carolina,
†Advanced Micro Devices, Inc. (AMD)
{makabir, tfkamuch, njfredri, mqhuang, dandrews}@uark.edu, jmandebi@amd.com, jbakos@cse.sc.edu,

***Abstract*—Processor-in-Memory (PIM) overlays and alternative reconfigurable tile fabrics have been proposed to eliminate the von Neumann bottleneck and enable processing performance to scale with BRAM capacity. The performance of these FPGA-based PIM architectures has been limited due to a reduction of the BRAMs maximum clock frequencies and less than ideal scaling of processing elements with increased BRAM capacity. This paper presents IMAGine, an In-Memory Accelerated GEMV engine, a PIM-array accelerator that clocks at the maximum frequency of the BRAM and scales to 100% of the available BRAMs. Comparative analyses are presented showing execution speeds over existing PIM-based GEMV engines on FPGAs and achieving a 2.65× – 3.2× faster clock. An AMD Alveo U55 implementation is presented that achieves a system clock speed of 737 MHz, providing 64K bit-serial multiply-accumulate (MAC) units for GEMV operation. This establishes IMAGine as the fastest PIM-based GEMV overlay, outperforming even the custom PIM-based FPGA accelerators reported to date. Additionally, it surpasses TPU v1-v2 and Alibaba Hanguang 800 in clock speed while offering an equal or greater number of multiply-accumulate (MAC) units.**

***Index Terms*—Processing-in-Memory, System Design, Block RAM, GEMV engine, Processor Array.**

I. Introduction

The exponential growth of Internet-of-Things (IoT) devices and social media applications has significantly changed the landscape of computing workloads. Modern workloads, such as scientific computation, graph processing, and machine learning, generate and process datasets that are expanding at a rate that outpaces Moore's Law [1]. However, today's processors remain constrained by the "Memory Wall" of the von Neumann architecture, which limits the ability to exploit the parallelism within these memory-intensive tasks. Processing-in-memory (PIM) architectures are being pursued [2]–[15] to mitigate the memory wall and enable processing performance to scale with memory capacity.

Modern Field Programmable Gate Arrays (FPGAs) with 100s of Mbits of SRAM distributed throughout the device in the form of disaggregated memory resources can provide several TB/s of internal bandwidth. This is an ideal programmable substrate for creating customized Processor In/Near Memory accelerators. Several PIM array-based accelerator designs [6]–[13] have been proposed to harness this massive internal bandwidth. However, results reported to date show achievable clock frequencies and compute densities are not sufficient to compete with their custom Application Specific Integrated Circuit (ASIC) counterparts.

Such shortcomings have motivated redesigns of the separate Block-RAM (BRAM) and LUT resources into tightly integrated PIM tiles. While these redesigns have increased chip compute densities, the maximum achievable clock frequency remains no better than their overlay counterparts. Additionally, the adoption of a bigger FPGA with an increased resource capacity does not translate into a linear increase in compute performance.

This paper presents IMAGine, a PIM array-based GEMV accelerator that clocks at the maximum frequency of the BRAM. The PIM tile array architecture of IMAGine has been designed to achieve linear scalability of the number of compute units with increased BRAM densities. Comparative studies are presented that show it is the fastest and most scalable PIM array-based GEMV accelerator reported to date. Run time results also show that IMAGine shatters some of the myths concerning performance limitations of PIM-array accelerators and FPGA overlays in general. Our contributions can be summarized as follows,

- A set of aspirational but practical design goals for PIM array-based accelerators. We argue these goals need to be met to claim a "Scalable High-Performance PIM design" on FPGAs.
- We present the design and implementation of IMAGine, an In-Memory Accelerated GEMV engine overlay, that breaks some existing myths around FPGA design, clocking faster than Google's TPU v1-v2 with equal or more processing elements (PEs) using an off-the-shelf datacenter-grade FPGA.
- We present a comparative study of IMAGine with existing PIM-array accelerators, establishing it as the fastest and most scalable FPGA PIM-based GEMV accelerator.

IMAGine has been published at [16] as open-source implementation and is freely available for study, use, modification, and distribution without restriction.

This material is based upon work supported by the National Science Foundation under Grant No. 1955820.

979-8-3315-3007-5/24/$31.00 ©2024 IEEE
DOI 10.1109/FPL64840.2024.00038

TABLE I
MAXIMUM FREQUENCY (MHZ) OF EXISTING FPGA-PIM DESIGNS

PIM Design	Type	Device	f_{BRAM}	f_{PIM}	Rel.	f_{Sys}	Rel.
CCB	Custom	Stratix 10	1000	624	62%	455	46%
CoMeFa-A	Custom	Arria 10	730	294	40%	288	39%
CoMeFa-D	Custom	Arria 10	730	588	81%	292	40%
BRAMAC-2SA	Custom	Arria 10	730	586	80%	-	-
BRAMAC-1DA	Custom	Arria 10	730	500	68%	-	-
M4BRAM	Custom	Arria 10	730	553	76%	-	-
SPAR-2	Overlay	UltraScale+	737	445	60%	200	27%
PiCaSO	Overlay	UltraScale+	737	737	100%	-	-

II. RELATED WORK

A. Custom-BRAM PIMs

Wang et al [6] proposed the Compute-Capable BRAM (CCB) based on Neural Cache [17]. CCB exposes compute parallelism within a BRAM by converting each BRAM bitline into a bit-serial Processing Element (PE). CCB was used to build RIMA [6] to accelerate recurrent neural networks (RNNs). RIMA achieved 1.25× and 3× higher performance compared to the Brainwave DL soft processor [18] for 8-bit integer and block floating-point precisions, respectively.

Arora et al [10], [11] proposed CoMeFa that uses bit-serial PEs per SRAM bitline like CCB, but exploits the dual-port nature of BRAMs to simultaneously read two operands. To evaluate the performance and energy benefits of CoMeFa RAMs, various microbenchmarks, including General Matrix-Vector Multiplication (GEMV) and General Matrix-Matrix Multiplication (GEMM) were studied in [11]. Augmenting an Intel Arria 10-like FPGA with CoMeFa RAMs delivered a geomean speedup of 2.55× across diverse applications.

Chen et al proposed BRAMAC [12] and M4BRAM [13], which bypass MAC computation on the slow and power-hungry primary BRAM array by copying operands to a smaller "dummy array". BRAMAC requires 2-/4-/8-bit predefined weights and activations, limiting its use to quantized uniform-precision deep neural nets. M4BRAM overcomes some of these limitations by enabling variable activation precision between 2 and 8 bits with linearly scaled MAC latency. Combining BRAMAC-2SA/BRAMAC-1DA with Intel's DLA [19] resulted in an average speedup of 2.05×/1.7× for AlexNet and 1.33×/1.52× for ResNet-34. M4BRAM surpassed BRAMAC by an average of 1.43× across diverse benchmarks.

B. BRAM-Overlay PIMs

To leverage the benefits of PIM architectures in contemporary FPGAs, PIM overlay architectures have been proposed. Panahi et al [7]–[9] proposed SPAR-2, a SIMD PIM-array overlay accelerator, connecting bit-serial PEs from the programmable fabric with BRAMs. SPAR-2 was implemented on Virtex-7 and Virtex UltraScale FPGAs with 10K PEs to accelerate several deep learning applications. It achieved up to 34.2× and 3.5× speedups compared to other custom HLS-based and RTL-based accelerators, respectively.

Building upon the PIM overlay of SPAR-2, Kabir et al proposed PiCaSO [15] with configurable pipeline stages along the datapath. PiCaSO introduced an intermediate muxing module to enable zero-copy in-block reduction and a "binary-hopping" pipelined NEWS network for array-level reduction. PiCaSO provided competitive performance and memory utilization efficiency compared to both CCB and CoMeFa custom-BRAM architectures.

TABLE II
DELAY (NS) BREAKDOWN OF 1-LEVEL LOGIC PATH IN AMD DEVICES

	FF-C2Q[1]	LUT	FF-Setup	Total[2]	BRAM[3]	Net Budget	SB-Min[4]
V7	0.290	0.34	0.255	0.885	1.839	0.954	0.272
US+	0.087	0.15	0.098	0.335	1.356	1.021	0.102

[1] Clock-to-Q delay of flip-flops
[2] Total cell delay
[3] BRAM pulse-width requirement, clock period for Fmax
[4] Minimum net delay through a switchbox

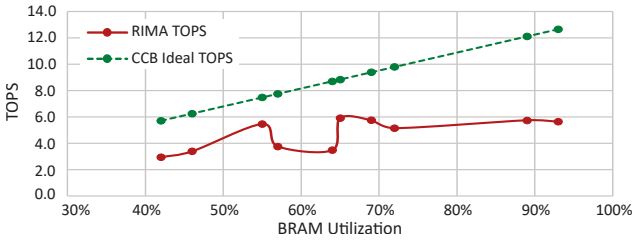

Fig. 1. Ideal scaling vs. actual TOPS of RIMA on Stratix 10 GX2800

III. MOTIVATION AND DESIGN GOALS

Table I summarizes the maximum frequencies of the PIM designs discussed in section II. The relative frequency columns (Rel.) show that the clock frequency f_{PIM} of all the PIM tiles are significantly slower compared to the maximum frequency for the device BRAMs (f_{BRAM}), except for PiCaSO. Their fastest system frequencies (f_{Sys}) are 2.1× – 3.7× slower than the BRAM maximum frequencies (f_{BRAM}). This slower frequency was attributed to the limitations of the soft logic and the routing resources of the FPGAs. It was also reported as unlikely that an FPGA accelerator at the system level would operate at a frequency surpassing the degraded frequency (f_{PIM}) of these PIM designs, even in a more advanced node than the evaluation platforms [10]–[13].

Further observation yielded that most of these systems could not utilize all available BRAMs as PIMs. This lower utilization combined with a lower clock frequency results in less efficient use of the available internal BRAM bandwidth of the devices and a lower system-level compute density. A final observation shows a troubling common pattern: as the utilization of BRAMs increases the achievable system-level clock frequency decreases [6], [11].

These observations motivated our interest in understanding if these results were a new reality of BRAM PIM arrays or symptomatic of specific design and implementation choices.

A. System Clock Speed Goal

In FPGAs, BRAMs are the single component with the longest latency [20]–[22]. Thus, we propose using the maximum frequency (Fmax) of the BRAM as the target frequency for the PIM-array accelerators. To assess the practicality of this

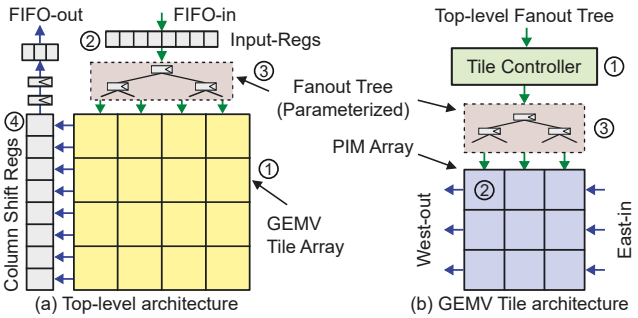

Fig. 2. System architecture of IMAGine illustrating the data and instruction flow (a) through the GEMV engine and (b) within GEMV tiles.

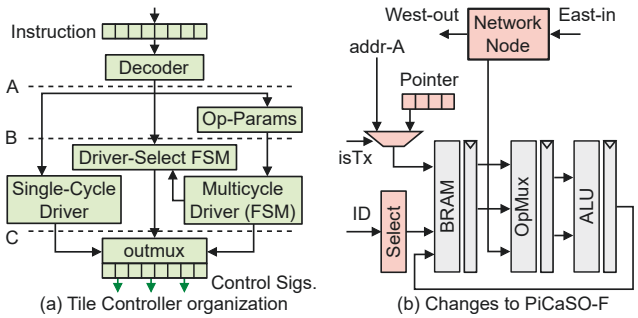

Fig. 3. Architectures of (a) GEMV controller and (b) PiCaSO-IM, the adapted version of PiCaSO-F [15].

design goal, we closely examined two AMD FPGA families: Virtex-7 and UltraScale+. We created a test design where all timing paths are one logic level deep and averaged all paths to obtain Table II. The Total column sums the cell delays in the columns to its left. The BRAM column lists the clock period for BRAM Fmax. The SB-Min column displays the minimum delay of a net passing through a switchbox. Net Budget is derived by subtracting the Total column from the BRAM column. Comparing the net budget with the minimum net delay shows that, it is feasible to design at least two LUTs deep logic paths clocking at the BRAM Fmax.

B. Performance Scaling Goal

We posited that the peak-performance of a PIM design needs to scale linearly with the on-chip BRAM resource. The compute capacity in custom-BRAM-based PIM designs [6], [10]–[13] scales linearly with BRAM count if all BRAM tiles are used in PIM mode. However, a significant sacrifice is imposed in the clock frequency that ends up limiting the achievable peak-performance on the device. Table I f_{PIM} column indicates that the custom-BRAM PIM designs are up to 2.5× slower than the BRAM Fmax. Fig. 1 plots RIMA's peak-performance from Table-II of [6], computed using reported BRAM utilization and M-DPE clock frequency. The irregular trend is attributed to RIMA's system-level architecture. If RIMA adhered to the proposed performance scaling goal, even at the degraded CCB frequency of 624 MHz, its peak-performance would align with the CCB Ideal TOPS line. The gap between these plots represents wasted compute capacity and memory bandwidth provided by CCB BRAMs.

IV. IMAGine Architecture

A. System-Level Architecture

The top-level system is illustrated in Fig. 2(a). It consists of (1) a 2D array of GEMV tiles, (2) a set of input registers, (3) a fanout tree connecting the input registers to the tile array, and (4) a column of shift-registers to read out the final result. The front-end processor sends instructions to the GEMV tiles through the input registers. The fanout tree is parameterized to be adjusted during implementation. The 2D tile array is implemented as a parameterized module that instantiates and connects the tiles. At the end of the GEMV operation, the output vector is stored in the column shift registers, which is shifted up and read through the FIFO-out port, one element per cycle.

B. IMAGine GEMV Tile Architecture

Illustrated in Fig. 2(b), the GEMV tile is the heart of IMAGine. It consists of (1) an FSM-based controller, (2) a 2D array of PIM blocks, and (3) a fanout tree between them. The controller receives the instruction written to the input registers at the top level, decodes it, and generates the sequence of control signals needed to execute the instruction. The fanout tree connects the control signals to all PEs in the PIM array and is parameterized for adjustment during implementation. The PIM array interfaces allow cascading with arrays in neighboring tiles on each side. During accumulation, partial results move from east to west through PIM arrays, ultimately accumulating in the left-most PE column of the left-most GEMV tile.

C. Tile Controller

Fig. 3(a) shows the architecture of the tile controller. It takes a 30-bit instruction, which is executed by either the single-cycle or the multicycle driver, selected by the 2-state driver-selection FSM. The single-cycle driver can execute one instruction every cycle, while the multicycle driver takes several cycles to execute instructions like ADD, SUB, MULT, etc. including an additional cycle to load its parameters from the Op-Params module. All inputs and outputs are registered to localize timing paths within the controller. The combinatorial logic in the controller is grouped into meaningful steps and optional pipeline stages are added as illustrated by the dashed lines A, B, and C in Fig. 3(a). Running synthesis, we ensured that each step could be implemented in one or two logic levels.

D. PIM Module

We adopted PiCaSO [15] as IMAGine's PIM module for the following three reasons: (1) it is publicly available and open-source [23], (2) it is a modifiable overlay that can be ported and studied on existing AMD devices, and (3) PiCaSO-F, a pipelined configuration of PiCaSO, can be clocked at the BRAM Fmax. The modifications highlighted in red in Fig. 3(b) were applied to PiCaSO-F to build PiCaSO-IM for IMAGine. The original NEWS network was replaced with a simpler east-to-west data movement network. Block-ID-based selection

TABLE III
UTILIZATION AND FREQUENCY OF 12×2 GEMV TILE COMPONENTS

	Controller	Rel.	Fanout	Rel.	PIM Array	Rel.	Tile
LUT	167	5.8%	0	0.0%	2736	94.2%	2903
FF	155	4.0%	615	15.9%	3096	80.1%	3866
DSP	0	-	0	-	0	-	0
BRAM	0	0.0%	0	0.0%	12.0	100.0%	12
Freq. (MHz)	890	1.2×	890	1.2×	737	1×	737

logic was included in PiCaSO-IM. IMAGine's accumulation algorithm requires 3 addresses to maximize the overlap of data movement and computation. As PiCaSO-F supports only 2 simultaneous addresses, we added a pointer register for the third address. If PiCaSO is realized as a custom-BRAM tile as proposed in [15], these changes can be implemented in programmable logic fabric, keeping registerfile, OpMux, and ALU modules within the BRAM tile. We name such a custom-BRAM implementation of PiCaSO-IM as PiCaSO-CB.

V. IMPLEMENTATION AND ANALYSIS

In this section, we discuss the bottom-up implementation and analysis of IMAGine, targeting the design goals discussed in Section III. In [15], PiCaSO was studied on AMD Alveo U55C (xcu55c, -2 speed grade). We use the same device as our implementation platform to keep the results predictable. The BRAM Fmax on this device is 737 MHz [21], which sets the target clock period to be 1.356 ns. All of the following studies were carried out using Vivado 2022.2.

A. GEMV Tile

The components of the GEMV tile were studied individually to verify if they met the design requirements. Each tile contains a 12×2 PIM array and 2 stages of pipeline in the fanout tree, which best fits the physical layout of the Alveo U55 FPGA as discussed later in this section. Table III shows the utilization and performance of these components and their relative values compared to the entire GEMV tile.

The controller together with the fanout network passed the timing constraints at a clock rate of 890 MHz. Because the PIM array contains the BRAM, it cannot run faster than the BRAM Fmax. It passed the timing at 737 MHz, the BRAM Fmax. As observed in Table III, the logic utilization of the controller is around 5% of the entire tile and requires no DSPs, while around 90% of the logic resources are consumed by the PIM array. Thus, the controller and the fanout tree are not expected to bottleneck system frequency or utilization. The GEMV tile's speed and scalability are fundamentally dependent on the PIM array, which is the desired outcome.

B. Scalability Study

To evaluate the scalability of our architecture on different device families, we followed the approach in [15]. Along with Alveo U55, four representatives were selected from AMD's Virtex-7 and UltraScale+ devices based on two criteria: BRAM capacity and LUT-to-BRAM ratio. Table IV lists these devices with their BRAM capacity, LUT-to-BRAM ratio, and a short ID used in Fig. 4. The target clock frequency of the system

TABLE IV
REPRESENTATIVES OF VIRTEX-7 AND ULTRASCALE+ FAMILIES [15]

Device	Tech	BRAM#	Ratio[1]	Max PE#[2]	ID
xcu55c-fsvh-2	US+	2016	646	64K	U55
xc7vx330tffg-2	V7	750	272	24K	V7-a
xc7vx485tffg-2	V7	1030	295	32K	V7-b
xc7v2000tfhg-2	V7	1292	946	41K	V7-c
xc7vx1140tflg-2	V7	1880	379	60K	V7-d
xcvu3p-ffvc-3	US+	720	547	23K	US-a
xcvu23p-vsva-3	US+	2112	488	67K	US-b
xcvu19p-fsvb-2	US+	2160	1892	69K	US-c
xcvu29p-figd-3	US+	2688	643	86K	US-d

[1] LUT-to-BRAM ratio
[2] Number of PEs utilizing all BRAMs as PIMs

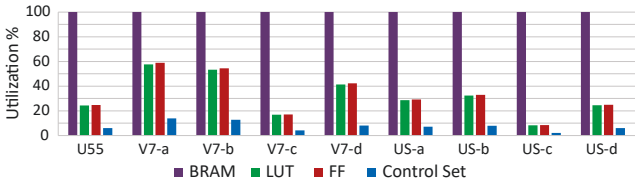

Fig. 4. Resource usage of IMAGine on representatives of Virtex-7 and Ultrascale+ families utilizing 100% BRAMs as PIM overlays.

was set to 100 MHz on all devices to avoid timing issues and only focus on the logic utilization of the system at this point.

Fig. 4 shows a bar graph of post-implementation utilization numbers of IMAGine on the representative devices. As observed, IMAGine can utilize 100% of the available BRAMs as PIM overlays providing 64K PEs in U55, with only 25% logic and 6% control set utilization. This leaves sufficient logic resources to implement the fanout trees and pipeline stages if they are needed to achieve the target clock speed. In fact, IMAGine scaled up to 100% of available BRAM in all the representative devices for Virtex-7 and UltraScale+ families.

In the Virtex-7 family, the device V7-a has the smallest number of BRAMs and the smallest LUT-to-BRAM ratio. IMAGine used around 60% logic resources to provide 24K PEs in V7-a. In the UltraScale+ family, US-a and US-b have the smallest number of BRAMs and the smallest LUT-to-BRAM ratio, respectively. In these devices IMAGine provide 23K and 67K PEs, respectively, using roughly 30% logic resources. For devices with more BRAMs and a higher LUT-to-BRAM ratio the logic utilization is very small: the logic utilization in US-c is less than 10% providing 69K PEs. Thus, IMAGine is scalable up to 100% BRAM capacity irrespective of the available logic resources in existing devices.

C. System-Level Timing Optimizations

For the final implementation, the target clock was set to 1.356 ns to match the BRAM Fmax of Alveo U55. The goal of the study was to find out how close we can get to the target clock rate, and what are the practical challenges that limit us from achieving it. We ran the first iteration using the default settings of Vivado and achieved a setup slack of -0.52 ns. The critical paths were within the controller with a logic depth of 4, going through the pipeline stage A of the controller as shown in Fig. 3(a). So, we enabled the pipeline stage A in the controller for the next iteration.

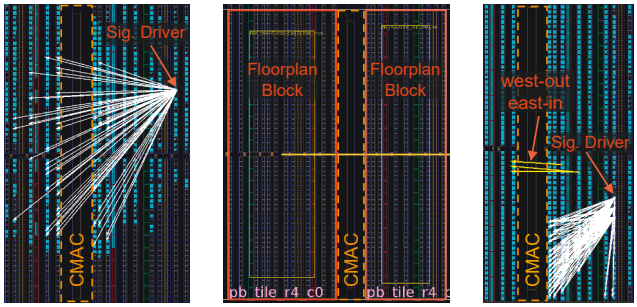

Fig. 5. Avoiding unnecessary hard-block (CMAC) crossing by floorplanning (a) placement and net connections before floorplanning, (b) floorplan localizing logic and routing, (b) placement and net connections in the final design.

At the end of the second iteration of implementation, we achieved a setup slack of -0.38 ns. The control signals between the controller and the PIM array were failing the timing due to their high fanout and long routes. Thus, we synthesize a fanout tree between the controller and the PIM array empirically choosing 2 levels and a fanout of 4 for the next iteration.

The design achieved a setup slack of -0.27 ns in the third iteration. The long routes crossing hard blocks, like an Ethernet port (CMAC) [24], were failing the timing. The white lines in Fig. 5(a) highlight some of those critical nets. To avoid placement results generating such paths, we created floorplanning blocks (Pblocks) [25] as shown in Fig. 5(b), to localize the logic placement and routing of a tile. This required defining a tile with 12×2 PIM array on Alveo U55. Fig. 5(c) shows the placement and net connections in the final iteration. The logic and routing of each tile are localized on either side of the hard block. Only the inter-tile connections for east-to-west accumulation, highlighted in yellow lines, cross the CMAC block requiring minimal routing resources.

The final design met the timing at 737 MHz clock, demonstrating the practical achievability of the proposed clocking goal. Utilizing 100% available BRAMs as PIMs, this design also achieved linear scaling of peak-performance. Surprisingly, this clock rate is faster than custom GEMM accelerator ASICs TPU v1-v2 [1], [26] and Alibaba Hanguang 800 [27], that run at 700 MHz. Both Alveo U55 and TPU v2 are manufactured at 16 nm and Hanguang 800 at 12nm technology nodes. So, this clock improvement is not due to a technology node difference. On Alveo U55, IMAGine has an equal number of PEs compared to TPU v1 (64K), and 4× of TPU v2 (16K). However, IMAGine can only deliver up to 0.33 TOPS at 8-bit precision, which is significantly smaller compared to TPU v1 (92 TOPS) and v2 (46 TOPS), due its bit-serial architecture. These results dispel the myth that FPGA designs are always slower and have less compute density compared to ASICs.

D. Comparison With Other PIM-Array Accelerators

Table V shows the utilization and system frequencies of existing GEMV engines and equivalent PIM-array accelerators. System-level utilizations and frequencies for BRAMAC and M4BRAM-based systems were not reported in [12], [13].

TABLE V
UTILIZATION AND FREQUENCY OF PIM-BASED GEMV/GEMM ENGINES

	LUT	FF	DSP	BRAM	f_{Sys}[1]	Rel. Freq
RIMA-Fast	60%		50%	55%	455	45.5%
RIMA-Large	89%		50%	93%	278	27.8%
CCB GEMV	27.9%		90.1%	91.8%	231	31.6%
CoMeFa-A GEMV	27.9%		90.1%	91.8%	242	33.2%
CoMeFa-D GEMM	25.5%		92.4%	86.7%	267	36.6%
SPAR-2 (US+)	11.3%	2.4%	0.0%	14.5%	200	27.1%
SPAR-2 (V7)	28.5%	7.0%	0.0%	30.4%	130	23.9%
IMAGine	35.6%	24.8%	0.0%	100.0%	737	100.0%
IMAGine-CB[2]	10.1%	7.2%	0.0%	100.0%	737	100.0%

[1] System frequency in MHz
[2] IMAGine with custom-BRAM PIM tile (PiCaSO-CB)

RIMA [6] was evaluated on a Stratix 10 GX2800 FPGA with a BRAM Fmax of 1 GHz [22]. Its fastest reported configuration (RIMA-Fast) runs at 455 MHz, which is 2.2× slower than its BRAM Fmax. The largest reported configuration (RIMA-Large) utilizes 93% of BRAMs and runs at 278 MHz, 4× slower than the BRAM Fmax. The GEMV/GEMM systems based on CCB and CoMeFa were evaluated on an Arria 10 GX900 with a BRAM Fmax of 730 MHz [11]. Though CoMeFa-based designs run slightly faster than the CCB-GEMV engine, they are still roughly 3× slower than the BRAM Fmax. Thus, CCB and CoMeFa-based GEMV/GEMM engine performance did not scale well at the system level.

SPAR-2 [8] utilized only 30% of the BRAMs while running 4× slower than BRAM Fmax on both platforms. Thus, its performance and scalability are even worse than CCB and CoMeFa-based systems. On the other hand, IMAGine has a system clock running at the BRAM Fmax while utilizing 100% device BRAM as PIMs. Outperforming all existing designs, IMAGine is the fastest PIM array-based GEMV engine implemented on any FPGA, running at a clock rate 2.65× – 3.2× faster than any existing design. This is an important proof of concept design that dispels earlier beliefs that PIM arrays and overlay accelerators cannot achieve BRAM Fmax clock frequencies at the system level [10]–[13].

As observed in Table V, RIMA and CCB/CoMeFa-based designs exhaust either the logic resources or the DSPs of the device even though their PIM blocks are implemented by customizing the BRAM tile itself. Even after being an overlay, IMAGine is achieving faster clock and better scalability using 0 DSPs and only one-third of the device logic resources due to its near-optimal architectural choices. Like SPAR-2, IMAGine does not use DSPs to implement the bit-serial PEs. With a custom-BRAM implementation of the PIM module, like PiCaSO-CB discussed in Section IV-D, IMAGine would consume about 10% of device resources while being fully scalable and implementable even in resource-limited FPGAs.

E. GEMV Execution Latency

Fig. 6 plots the GEMV latency of PIM-array accelerators, with square-matrix dimensions on the x-axis and latency in log scale on the y-axis. The execution times in Fig. 6(b) are computed by multiplying cycle latencies with the corresponding clock periods from Table V system frequencies.

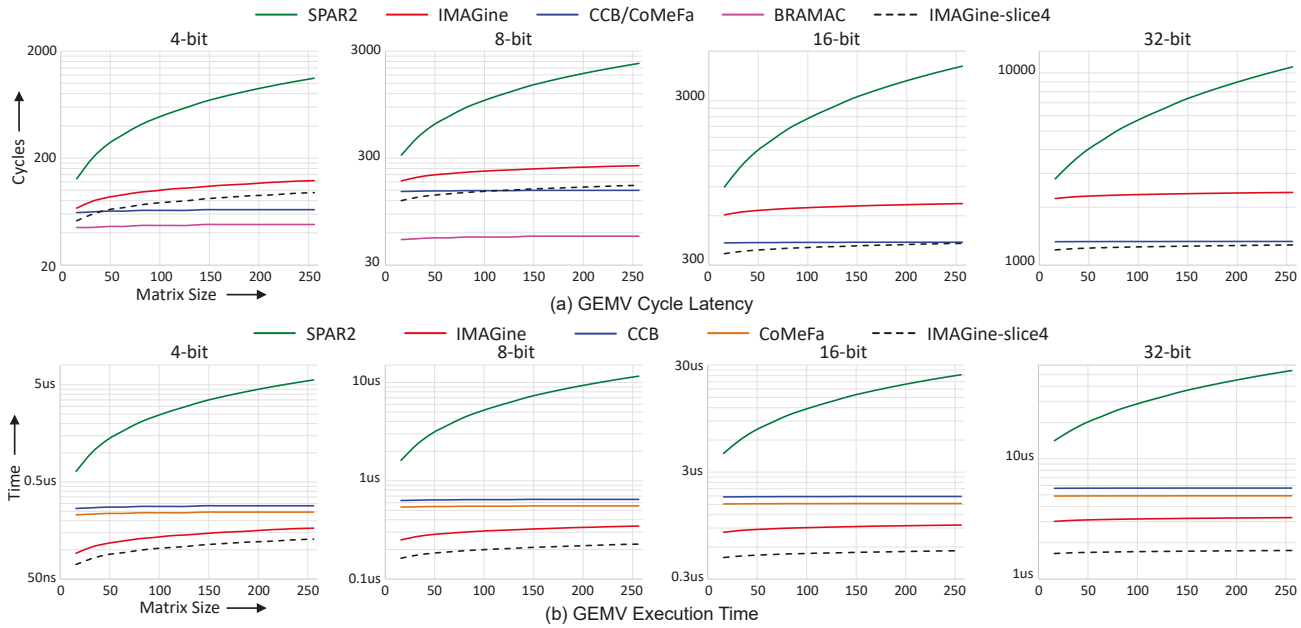

Fig. 6. Cycle latency and execution time of GEMV operation on different PIM array-based FPGA accelerators

We adopted the approach in [12] to model the block-level cycle latencies of CCB, CoMeFa, BRAMAC, and SPAR-2 using their analytical models. IMAGine's latency model was developed and validated by running a prototype on hardware.

As observed in Fig. 6(a), BRAMAC has the shortest cycle latency, due to their hybrid bit-serial & bit-parallel MAC2 algorithm. BRAMAC's MAC latency grows linearly with operand bit-width, while it grows quadratically in the other bit-serial architectures. BRAMAC is designed specifically for low-precision (2, 4, and 8-bit) quantized neural networks, rendering it unsuitable for general computing tasks like GEMV. BRAMAC did not report the system-level frequency which is why we could not plot its execution time.

SPAR-2 has the longest latency across all precisions, due to its slow NEWS accumulation network, with latency increasing almost linearly with matrix dimension. CCB and CoMeFa-based GEMV engines have the shortest cycle latency among bit-serial architectures across all precisions. This is due to their fast reduction algorithm based on a popcount-based adder and pipelined adder tree. The cycle latency of IMAGine is significantly shorter compared to SPAR-2 but longer than CCB/CoMeFa-based implementations. However, IMAGine clocks at least 2× faster than any of the other GEMV engines. As a result, IMAGine outperforms all other GEMV engines in terms of overall execution time. This highlights the importance of the system clock speed over the cycle latency; despite the CCB/CoMeFa GEMV engines' shorter cycle latency, their slower clock significantly degrades the execution time.

Because IMAGine is utilizing only 30% of the logic resources in U55, the remaining resources can be used to further improve its performance. The IMAGine-slice4 curves in Fig. 6 shows the latency of a variant of IMAGine with a 4-bit sliced accumulation network and a PE implementing Booth's radix-4 multiplication (default is radix-2). This latency is estimated by adjusting the analytical model of IMAGine assuming no effect on the clock rate. In terms of cycle latency, it can run almost as fast as CCB/CoMeFa-based GEMV implementations, while significantly outperforming them in execution time.

VI. Conclusions and Future Work

Processor In/Close to Memory (PIM) architectures have become popular frameworks replacing classic von Neumann architectures within domain-specific machine learning accelerators. This paper presented a study proposing the performance and scalability goals for PIM array-based accelerators on FPGAs. The design, implementation, and analysis of IMAGine was presented demonstrating how a PIM-array accelerator could achieve the BRAM Fmax as the system frequency. A scalability study was presented showing processing capacity scaling linearly with increasing BRAM density, even for devices with low LUT-to-BRAM ratios. An implementation with 64K PEs was run on Alveo U55, clocking faster than the Tensor Processing Unit (TPU v1-v2) and Alibaba Hanguang 800. This breaks the myth that FPGA overlays and fabrics must clock slower than ASIC designs.

A comparative study with state-of-the-art PIM-array accelerators was presented showing IMAGine has 2.65× – 3.2× faster system frequency, and significantly outperforms them in execution time, establishing IMAGine as the fastest and most scalable PIM array-based GEMV engine reported to date.

Our future work includes the completion of an MLIR-based compiler framework for hardware/software codesign and application-specific customization of IMAGine-like PIM array-based accelerators.

REFERENCES

[1] N. P. Jouppi, D. Hyun Yoon, M. Ashcraft, M. Gottscho, T. B. Jablin, G. Kurian, J. Laudon, S. Li, P. Ma, X. Ma, T. Norrie, N. Patil, S. Prasad, C. Young, Z. Zhou, and D. Patterson, "Ten lessons from three generations shaped google's TPUv4i : Industrial product," in *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, Jun. 2021, pp. 1–14.

[2] J.-H. Kim, J. Lee, J. Lee, H.-J. Yoo, and J.-Y. Kim, "Z-PIM: An Energy-Efficient Sparsity Aware Processing-In-Memory Architecture with Fully-Variable Weight Precision," in *2020 IEEE Symposium on VLSI Circuits*. IEEE, Jun. 2020, pp. 1–2.

[3] B. Zhang, S. Yin, M. Kim, J. Saikia, S. Kwon, S. Myung, H. Kim, S. J. Kim, J.-S. Seo, and M. Seok, "PIMCA: A Programmable In-Memory Computing Accelerator for Energy-Efficient DNN Inference," *IEEE Journal of Solid-State Circuits*, vol. 58, no. 5, pp. 1436–1449, May 2023.

[4] C.-F. Lee, C.-H. Lu, C.-E. Lee, H. Mori, H. Fujiwara, Y.-C. Shih, T.-L. Chou, Y.-D. Chih, and T.-Y. J. Chang, "A 12nm 121-TOPS/W 41.6-TOPS/mm2 All Digital Full Precision SRAM-based Compute-in-Memory with Configurable Bit-width For AI Edge Applications," in *2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*. IEEE, Jun. 2022, pp. 24–25.

[5] Y. Kwon, G. Kim, N. Kim, W. Shin, J. Won, H. Joo, H. Choi, B. An, G. Shin, D. Yun, J. Kim, C. Kim, I. Kim, J. Park, C. Park, Y. Song, B. Yang, H. Lee, S. Park, W. Lee, S. Lee, K. Kim, D. Kwon, C. Jeong, J. Kim, E. Lim, and J. Chun, "Memory-Centric Computing with SK Hynix's Domain-Specific Memory," in *2023 IEEE Hot Chips 35 Symposium (HCS)*, 2023, pp. 1–26.

[6] X. Wang, V. Goyal, J. Yu, V. Bertacco, A. Boutros, E. Nurvitadhi, C. Augustine, R. R. Iyer, and R. Das, "Compute-Capable Block RAMs for Efficient Deep Learning Acceleration on FPGAs," *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pp. 88–96, 2021.

[7] S. Basalama, A. Panahi, A.-T. Ishimwe, and D. Andrews, "SPAR-2: A SIMD Processor Array for Machine Learning in IoT Devices," in *2020 3rd International Conference on Data Intelligence and Security (ICDIS)*. IEEE, 2020, pp. 141–147.

[8] A. Panahi, S. Balsalama, A.-T. Ishimwe, J. M. Mbongue, and D. Andrews, "A Customizable Domain-Specific Memory-Centric FPGA Overlay for Machine Learning Applications," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, Aug. 2021, pp. 24–27.

[9] A. Panahi, "A memory-centric customizable domain-specific FPGA overlay for accelerating machine learning applications," Ph.D dissertation, University of Arkansas, 2022.

[10] A. Arora, T. Anand, A. Borda, R. Sehgal, B. Hanindhito, J. Kulkarni, and L. K. John, "CoMeFa: Compute-in-Memory Blocks for FPGAs," in *2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, May 2022, pp. 1–9.

[11] A. Arora, A. Bhamburkar, A. Borda, T. Anand, R. Sehgal, B. Hanindhito, P.-E. Gaillardon, J. Kulkarni, and L. K. John, "CoMeFa: Deploying Compute-in-Memory on FPGAs for Deep Learning Acceleration," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 16, no. 3, pp. 1–34, Sep. 2023.

[12] Y. Chen and M. S. Abdelfattah, "BRAMAC: Compute-in-BRAM Architectures for Multiply-Accumulate on FPGAs," in *2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. Marina Del Rey, CA, USA: IEEE, May 2023, pp. 52–62.

[13] Y. Chen, J. Dotzel, and M. S. Abdelfattah, "M4BRAM: Mixed-Precision Matrix-Matrix Multiplication in FPGA Block RAMs," Nov. 2023, arXiv:2311.02758 [cs]. [Online]. Available: http://arxiv.org/abs/2311.02758

[14] M. A. Kabir, J. Hollis, A. Panahi, J. Bakos, M. Huang, and D. Andrews, "Making BRAMs Compute: Creating Scalable Computational Memory Fabric Overlays," in *2023 IEEE 31st Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, May 2023, pp. 224–224.

[15] M. A. Kabir, E. Kabir, J. Hollis, E. Levy-Mackay, A. Panahi, J. Bakos, M. Huang, and D. Andrews, "FPGA Processor In Memory Architectures (PIMs): Overlay or Overhaul ?" in *2023 33rd International Conference on Field-Programmable Logic and Applications (FPL)*. Gothenburg, Sweden: IEEE, Sep. 2023, pp. 109–115.

[16] M. A. Kabir, T. Kamucheka, N. Fredricks, J. Mandebi, J. Bakos, M. Huang, and D. Andrews, "IMAGine: An In-Memory Accelerated GEMV Engine Overlay." [Online]. Available: https://github.com/Arafat-Kabir/IMAGine

[17] C. Eckert, X. Wang, J. Wang, A. Subramaniyan, R. Iyer, D. Sylvester, D. Blaauw, and R. Das, "Neural Cache: Bit-Serial in-Cache Acceleration of Deep Neural Networks," in *2018 ACM/IEEE 45Th annual international symposium on computer architecture (ISCA)*, 2018, pp. 383–396.

[18] J. Fowers, K. Ovtcharov, M. Papamichael, T. Massengill, M. Liu, D. Lo, S. Alkalay, M. Haselman, L. Adams, M. Ghandi, S. Heil, P. Patel, A. Sapek, G. Weisz, L. Woods, S. Lanka, S. K. Reinhardt, A. M. Caulfield, E. S. Chung, and D. Burger, "A Configurable Cloud-Scale DNN Processor for Real-Time AI," in *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 2018, pp. 1–14.

[19] U. Aydonat, S. O'Connell, D. Capalija, A. C. Ling, and G. R. Chiu, "An OpenCL™ Deep Learning Accelerator on Arria 10," in *Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. Association for Computing Machinery, 2017, p. 55–64.

[20] *Virtex-7 T and XT FPGAs Data Sheet: DC and AC Switching Characteristics*, AMD, 2021. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ds183_Virtex_7_Data_Sheet

[21] *Virtex UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics*, AMD, 2021. [Online]. Available: https://docs.xilinx.com/v/u/en-US/ds923-virtex-ultrascale-plus

[22] *Intel® Stratix® 10 Device Datasheet*, Intel. [Online]. Available: https://www.intel.com/content/www/us/en/docs/programmable/683181/current/memory-block-specifications.html

[23] M. A. Kabir, E. Kabir, J. Hollis, E. Levy-Mackay, A. Panahi, J. Bakos, M. Huang, and D. Andrews, "PiCaSO: A Scalable and Fast PIM Overlay." [Online]. Available: https://github.com/Arafat-Kabir/PiCaSO

[24] *Alveo U55C Data Center Accelerator Card User Guide*, AMD. [Online]. Available: https://docs.xilinx.com/r/en-US/ug1469-alveo-u55c

[25] *Vivado Design Suite User Guide: Design Analysis and Closure Techniques*, AMD. [Online]. Available: https://docs.amd.com/r/en-US/ug906-vivado-design-analysis/Using-Pblock-Based-Floorplanning

[26] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th annual international symposium on computer architecture*, 2017, pp. 1–12.

[27] Y. Jiao, L. Han, R. Jin, Y.-J. Su, C. Ho, L. Yin, Y. Li, L. Chen, Z. Chen, L. Liu, Z. He, Y. Yan, J. He, J. Mao, X. Zai, X. Wu, Y. Zhou, M. Gu, G. Zhu, R. Zhong, W. Lee, P. Chen, Y. Chen, W. Li, D. Xiao, Q. Yan, M. Zhuang, J. Chen, Y. Tian, Y. Lin, W. Wu, H. Li, and Z. Dou, "7.2 a 12nm programmable convolution-efficient neural-processing-unit chip achieving 825tops," in *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. IEEE, Feb. 2020, pp. 136–140.