Corsair: An In-Memory Computing Chiplet Architecture for Inference-Time Compute Acceleration

Satyam Srivastava, Akhil Arunkumar, Nithesh Kurella, Amrit Panda, Gaurav Jain, Purushotham Kamath, Mark Wutzke, Arun Tiruvur, Mayank (Mike) Gupta, Ilya Soloveychik, Vamsi Darsi, Malav Dalal, Vinayak Patankar, Sasidhar Dudyala, Senthil Duraisamy, Santhosh Ramchandran, Raghav Venkatasubramanian, Yuwei Qin, Xin Wang, Jayaprakash Balachandran, Ali Murat Gok, Piotr Wojciechowski, Saliya Ekanayake, Chris Ng, Ranju Sarma, Shubhankit Rathore, Tristan Trouwen, Siwei Zhuang, Chris Nicol, and Sudeep Bhoja, *d-Matrix Corporation, Santa Clara, CA, 95054, USA*

Advances in generative AI (GenAI) have reinvigorated research into novel computing architectures such as Transformer. Transformer, characterized by low arithmetic intensity during most of the inference time, has become the cornerstone of GenAI underlying large language models (LLMs) and reasoning models (RMs). Numerous solutions to the intense memory bandwidth problem have been proposed. Corsair is an architecture that targets this need using chiplet design, digital in-memory computing-based matrix engine, efficient die-to-die interconnects, block floating point numerics, and large high-bandwidth on-chip memories. We describe the Corsair chiplet, scaling approaches to compose larger systems, and outline the software stack. We formulate the inference-time requirements of LLM and RM computation, memory bandwidth, memory capacity, and interconnect efficiency for scaling. We also show how the Corsair design fits perfectly these workloads. We present benchmark results from Corsair silicon that correlate strongly with the design and preview an estimate of workload-level improvements expected with Corsair.

Large language models (LLMs) have become the cornerstone of modern artificial intelligence (AI). These models have traditionally charted the path indicated by the LLM scaling laws with state-of-the-art models sporting hundreds of billions or even trillions of parameters.[1,2] However, after reaching the trillion parameter scale in 2023, the scaling laws of LLM pre-training seem to have plateaued. This is because 1) the computational needs of training ever larger models are proving to be impractical, and 2) the available data to train the models is finite. Today, we are at the onset of a new paradigm; that of inference-time compute where relatively smaller models achieve higher accuracy by allowing for iteration and reasoning during inference.

While reasoning models[3,4] achieve superior results on various measures of model quality, they come with significant performance overheads. For example, a one billion (1B) parameter reasoning LLM that uses up to 128 generations, achieves a math500 score similar to an 8 billion parameter model under zero-shot inference. However, the increased math500 score of the 1B reasoning LLM comes at a considerable performance cost compared to a one billion parameter model under zero-shot inference. Thus, reducing execution latency is crucial for enhancing user experience and lowering deployment costs by enabling higher system throughput.

Four key factors influence LLM inference execution latency. First, the prefill stage is primarily

Digital Object Identifier 10.1109/MM.2025.3593444
Date of publication 4 August 2025; date of current version 31 October 2025.

compute-throughput bound, as parameters fetched from memory are reused proportionally to the input sequence length. Second, LLMs store and repeatedly access intermediate activations within a per-sequence key-value (KV) cache during token generation. Because each sequence has a unique KV cache, memory demands increase proportionally with batch size, requiring high-capacity, high-bandwidth memory. Third, these memory and compute constraints highlight the importance of supporting low-precision numerics, which enhance compute throughput and effectively increase memory capacity and bandwidth utilization. Finally, as LLMs grow in size, their execution typically involves partitioning individual layers across multiple devices and distributing different layers across device clusters. This requires efficient system scaling through low-latency, high-bandwidth interconnects.

THERE IS A CLEAR NEED FOR ARCHITECTURES OPTIMIZED FOR LATENCY-BOUNDED THROUGHPUT, BALANCING USER INTERACTIVITY (LOW LATENCY) WITH COST EFFICIENCY (HIGH THROUGHPUT).

While massively parallel GPUs have driven significant advancements in AI, they often fall short on the four key factors required for LLM inference (as described previously). For instance, although Moore's law and low-precision arithmetic have boosted GPUs' compute capabilities, their memory and interconnect bandwidth have not scaled proportionally.[5] While some alternative accelerators[6,8] are used for LLM inference, they are usually not specifically designed for this purpose. They tend to provide higher memory bandwidth but often fall short in memory capacity and low-precision numeric support. These architectural mismatches lead to severe under-utilization (as with GPUs) or excessively large system footprints (as with other accelerators). Consequently, such solutions suffer from high CAPEX and OPEX costs, negatively impacting their economic viability for widespread deployment. Thus, there is a clear need for architectures optimized for latency-bounded throughput, balancing user interactivity (low latency) with cost efficiency (high throughput).

We present Corsair, an accelerator designed specifically to address the challenges of LLM inference. Corsair is built on four key pillars:

- *Digital in-memory compute (DIMC)*: We develop an energy-efficient DIMC-based tensor compute engine that achieves up to 47 TOPS/W and provides 2400–9600 TOPS (8-b/4-b precision) per card.
- *Block-floating point numerics*: To provide sufficient numerical accuracy at low-precisions with energy-efficient DIMC-based compute, we adopt MXINT-like numerics.
- *Large high-bandwidth on-chip memories*: Corsair consists of 2 GB of on-chip static random-access memory (SRAM) per card that provide a net bandwidth of 150 TB/s.
- *Efficient interconnects and multi-chiplet design*: Motivated by rising silicon costs, Corsair is designed from the ground up to be a multi-chiplet design. To enable this, we build a custom die-to-die (D2D) on-package interconnect and leverage standard PCIe interconnect beyond a package.

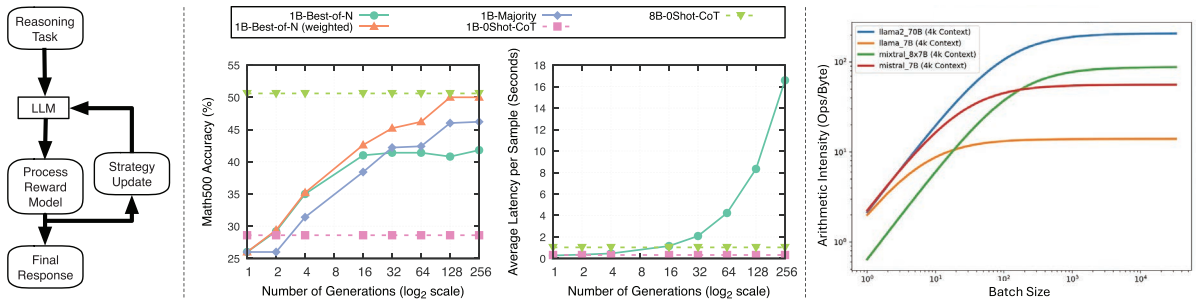
INFERENCE TIME COMPUTING

Adaptive compute scaling during inference substantially improves LLM reasoning capabilities. A typical reasoning architecture [Figure 1(a)] iteratively generates candidate responses using an LLM, guided by a process reward model (PRM), usually another LLM. After multiple iterations, the best-scoring response is selected. Figure 1(b) compares math500 accuracy of LLaMA-1B using Hugging Face's reasoning framework, evaluating best-of-N methods against zero-shot chain-of-thought inference for LLaMA-1B and 8B on an A100 GPU. With 128 reasoning steps, LLaMA-1B matches the accuracy of zero-shot LLaMA-8B but incurs ~9× higher latency, showing a tradeoff between reduced training costs and increased inference expense.

This latency presents two key challenges at scale: poor user experience (~9 s/sample on math500) and low throughput from batch-size-one inference. Larger batches improve throughput but worsen latency. Profiling pinpoints token generation and scoring as the main bottlenecks, underscoring the need for faster inference.

LLM Inference Performance Characteristics

LLMs are autoregressive generative models built upon decoder-only transformer architectures. Inference involves two distinct phases: *prefill* (processing the input prompt) and *decode* (generating tokens). Achieving high throughput (tokens/s) demands high resource utilization in both phases. In contrast, interactivity is



(a) (b) (c)

FIGURE 1. (a) A typical reasoning model architecture from Beeching[3]. (b) The math500 accuracy achieved by a LLaMA-1B-based reasoning model and associated inference latency. (c) Arithmetic intensity of LLMs during token generation.

governed by low latency metrics, specifically, time to first token (TTFT) and time per output token (TPOT).

- *Prefill*: In this compute-bound phase, the model processes input context, repeatedly accessing parameters from memory. Parameter reuse is proportional to the prompt length, making compute throughput a key determinant of TTFT.
- *Decode*: The model generates tokens autoregressively, conditioned on the full history. To avoid redundant computation, past intermediate activations, "keys" (K) and "values" (V), are stored in a KV cache. Each generated token needs accessing model parameters and KV cache from memory.

CORSAIR ADDRESSES THIS WITH SUBSTANTIALLY LARGER SRAM, PROVIDING BOTH THE BANDWIDTH AND CAPACITY NEEDED FOR EFFICIENT DECODE PERFORMANCE.

KV cache size scales linearly with context length, model size, attention heads, and batch size, making the decode phase memory-bound. Techniques like group query attention reduce KV size, but memory bandwidth remains the primary bottleneck, captured by arithmetic intensity (Ops/Byte).

Figure 1(c) shows that increasing batch size initially raises arithmetic intensity but quickly saturates—an inherent property of LLM inference. To maximize utilization during token generation, the system's compute-to-bandwidth ratio must align with the workload's arithmetic intensity, which saturates between 10–200 Ops/Byte. In contrast, modern accelerators (e.g., Nvidia H100/H200/B200, AMD MI325X) offer far higher ratios (~430–600 Ops/Byte), highlighting a mismatch. Improving tokens/s and user TPOT, therefore, demands much higher memory bandwidth, yet scaling high bandwidth memory is constrained by energy and die area. Corsair overcomes this by delivering an order-of-magnitude bandwidth increase over state-of-the-art GPUs.

While LLMs support long contexts (128 K+ tokens), real prompts are typically shorter. However, step-by-step reasoning can generate outputs ~10× longer than the prompt, significantly increasing KV cache demand. Corsair addresses this with substantially larger SRAM, providing both the bandwidth and capacity needed for efficient decode performance.

Scale-Out for LLM Inference

At-scale LLM serving extends beyond a single device, aggregating compute, memory, and interconnects across data center-scale deployments. Training clusters prioritize throughput, while interactive inference emphasizes latency, making distributed LLM execution a hardware–algorithm co-design challenge. It requires the following:

- memory capacity/bandwidth gains to outweigh the increased footprint of execution
- aggregate compute gain to outweigh the overheads from collective communication in large layers.

Scaling is typically achieved via sharding layers (tensor parallel or TP) and distributing layers (pipeline parallel or PP). TP collectives, such as AllReduce, are quite sensitive to latency and interconnect topology, and are commonly implemented using ring or one-hop tree AllReduce. Based on a latency-bandwidth model,

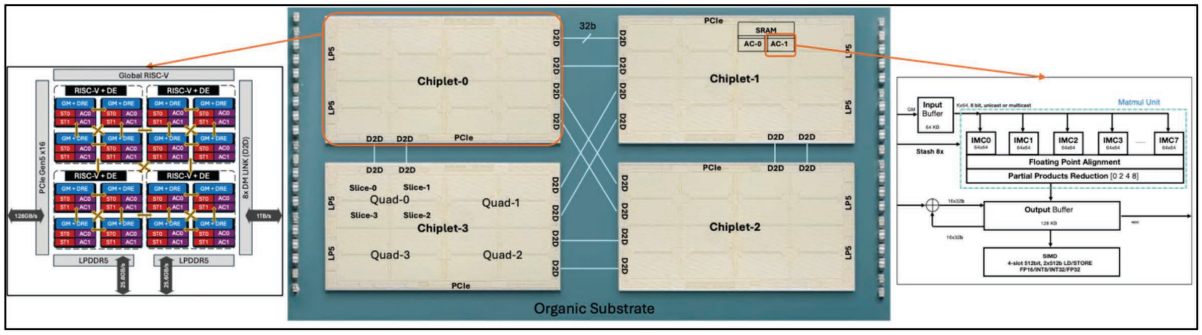


FIGURE 2. A labeled picture of the Corsair MCM showing the organization into interconnected chiplets, quads, and slices. Block diagrams of the SoC and the Apollo Core are also illustrated.

the communication cost for N-device ring is represented as

$$2 * (N-1) * \left[L + \frac{T}{N * B}\right].$$

Here, N is the number of devices, L is link-latency, B is total inter-device bandwidth per device, and T is tensor size. Here, bandwidth cost is nearly independent of device count, but latency grows linearly with the number of devices making it suboptimal for small-message size communications.

On the other hand, the communication cost for a one-hop tree-based *AllReduce* can be expressed as

$$2 * \left[L + \frac{T}{B}\right].$$

Here, the latency cost is also constant while the bandwidth cost is unchanged, making it latency-optimal.

Device connectivity strongly influences algorithm performance. One-hop tree requires a fully connected topology, whereas ring *AllReduce* can run on a variety of topologies. Corsair's scale-out architecture addresses both link-latency and effective bandwidth to support multiple parallelism strategies.

CORSAIR ARCHITECTURE

Corsair is a data center AI inference accelerator that caters to the high compute throughput and memory bandwidth demands of the prompt and token generation phases of LLMs, respectively. Figure 2 shows a picture of an un-lidded Corsair multichip module (MCM) where the parts and links have been annotated. A DIMC array coupled with a vector single instruction, multiple data (vSIMD) unit and dedicated scratchpad memory forms the compute building block of Corsair referred to as an Apollo Core (AC). Two ACs form a slice that share a larger data buffer–global memory (GM), a data reshaper, and utilize low-latency interconnects to efficiently tackle tensors of higher dimensions. Four slices are further clustered to form a quad which is also equipped with a RISC-V-based control core and an instruction dispatcher–dispatch engine (DE) and can autonomously execute subgraphs. The Corsair system-on-chip (SoC or chiplet) comprises of four such quads, a global control core along with dynamic data refresh (DDR) and peripheral component interconntect express die-to-die (PCIe) subsystems. Four Corsair chiplets, fully connected to each other using D2D links shown in cyan, constitute the MCM. And finally, two interconnected MCMs form a Corsair PCIe card.

Digital In-Memory Compute

A key innovation in the corsair architecture is the DIMC arrays. These arrays are designed to perform matrix–matrix multiply and accumulate operations in a highly energy-efficient manner. The in-memory nature of the computation allows for data reuse of the weight tensors to be exploited efficiently by multiplying them with multiple rows of the activation tensor. Furthermore, the fully digital design allows these cores to be as accurate and precise as other well-known matrix multiplication engines without the challenges associated with analog and resistive in-memory compute technologies.

The block diagram of a DIMC array is shown in Figure 3(a). Activations are streamed in dual-bit-serial manner to each of the 64 columns of the weight buffer. Each column comprises a pair of multipliers and adder trees that allow the unit to process all combinations of 4- and 8-b operands. In addition, the DIMC cores are paired with a flexible partial product reduction (PPR) engine that allows for PPR to be performed at a granularity of 1×, 2×, 4×, and 8× DIMCs. Furthermore, dedicated shift/align logic handles shared exponents which forms the basis of accurate

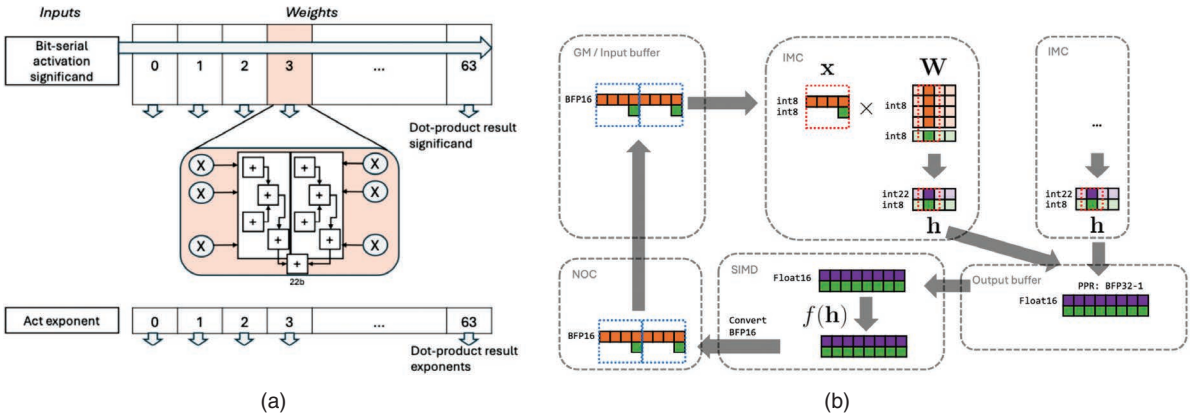


FIGURE 3. (a) A block diagram of the Corsair DIMC array. (b) An illustration of data types and conversions during a sequence of operations.

and efficient block floating point (BFP) tensor operations in Corsair.

BFP formats[7,9] combine wide dynamic range, numerical accuracy, and efficient hardware implementation of inner products using simple integer arithmetic. BFP formats are represented by a block of integer elements sharing one exponential scaling factor. Corsair carries numerous hardware converters that allow seamless conversion of tensors between original formats (FP32, FP16) and BFP formats, as illustrated by the example in Figure 3(b).

vSIMD Compute Cores

The vSIMD compute cores in Corsair are custom built for accelerating linear and nonlinear activation functions that include memory-intensive element-wise operations as well as compute-intensive trigonometric, transcendental, and reduction operations. Corsair exploits the data-level parallelism offered by the activation functions via a vector processing unit that is coupled with a scalar core to cater to the programmability demands of these functions. The vSIMD cores in conjunction with Corsair network on chip (NoC) are designed with scalability in mind and are conducive to distributed algorithms that tackle the increasing model dimension and context length.

A vSIMD core is a four-wide very large instruction width machine with fully pipelined functional units that support integer and floating-point compute. The activation functions are captured as vSIMD kernels that reside in a 32-kB private instruction scratchpad within a core. The primary data buffer for streaming tensors in and out of vSIMD cores is a multibanked scratchpad memory known as the output buffer (OB) that is shared between a vSIMD core and a DIMC-array. The OB offers low latency and high bandwidth memory accesses that can sustain up to three memory operations every cycle while the dynamic instruction slot mapping effectively handles both the compute-intensive and memory-intensive operations.

Memory Hierarchy

Corsair's memory hierarchy is designed to address both the capacity and the bandwidth needs posed by LLM acceleration.

Stash Memory

The stash memory is the high-bandwidth SRAM used to store model weights and KV cache tensors that feed directly into the DIMC cores. Recall that LLM inference is dominated by the performance of vector-matrix multiplications. Therefore, the bandwidth delivered to DIMC cores governs the performance of the workload. The stash memory is designed as a highly banked (1× bank per DIMC core) SRAM memory and feeds each DIMC core through a 64 B/clock interface. Furthermore, this is implemented with high-density SRAM cells to maximize available capacity. Multiple stash banks can be triggered by software to load their shard of weight tile concurrently.

Global Memory

The global memory is a shared SRAM resource that is available for each slice. It is used as a staging area for 1) input activations, 2) output/intermediate activation collection, and 3) collective operations. To service the aforementioned use cases in parallel and support

compute-data transfer overlap, the global memory is designed as a multibanked resource.

OB

The OB is a shared scratchpad SRAM that forms the primary data buffer between a DIMC-array and a vSIMD unit within an AC. The OB is organized as 16 banks of 8 kB each and supports simultaneous accesses by multiple streams—eight DIMC streams (one per core), three vSIMD streams, and two NoC streams. A key aspect of OB is an in-place reduction primitive that can be exercised during data transfers across OBs. This reduction primitive accelerates accumulating partial sums across ACs.

Main Memory

Corsair SoC consists of a dual low power dynamic data refresh (LPDDR) interface that supports the main memory. Each SoC is connected to up to 32 GB of LPDDR5 memory, providing 50 GB/s bandwidth. At a card level, this results in up to 256 GB of memory capacity and 400 GB/s bandwidth. Corsair main memory serves multiple purposes. First, it is the main interface for host–device communication. Second, with the large capacity afforded by DRAM technology, it is used to support a variety of usage scenarios such as rapid model swapping, prompt KV caching, inference on small device footprint, and offline execution of large models, contexts, and batches.

NoC

The NoC forms an integral part of the low-latency inference execution puzzle. We designed a hierarchical NoC providing low-hop connectivity to frequently accessed end points. At the highest level, each quad contains a quad-crossbar switch, and the four quads are interconnected by a physical all-to-all topology. Each quad-crossbar also connects to the four slices within a quad, the RISC-V CPU, and certain input–output (I/O). Depending on the quad, the I/O ports are used to connect one or more PCIe, D2D, and LPDDR interfaces. The end-points within a slice such as the DIMC cores, GM, Stash, DRE, and so on, are interconnected by a slice-crossbar. The slice-crossbar also provides connectivity to the quad-crossbar and to the rest of the SoC.

I/O, Control, and Fixed Functions

Each Corsair SoC consists of a PCIe Gen5 x16 interface that is either used for host connectivity or inter-SoC connectivity at 128 GB/s bidirectional bandwidth. Additionally, each SoC consists of multiple D2D interfaces, a low-latency, energy-efficient on-package interconnect technology used to connect multiple SoCs in a package. Each quad in the SoC forms a basic unit of graph execution, and the control plane comprises of a hardware-based instruction "dispatch engine" and a RISC-V CPU that runs the chip firmware. Finally, each slice in the SoC consists of a data reshape engine (DRE). The DRE provides acceleration for the common reshape operation present in neural networks such as transpose, tensor insertion, and tensor extraction.

Data Flow

Corsair implements a data flow architecture based on a spatial array of processing units. The key attributes of the execution model are as follows:

- Every workload is expressed as a Corsair instruction set architecture (ISA) graph and is spatiotemporally mapped. Compute nodes naturally map to various compute units like DIMC and SIMD while data dependency edges translate to DMA operations.

THE DRE PROVIDES ACCELERATION FOR THE COMMON RESHAPE OPERATION PRESENT IN NEURAL NETWORKS SUCH AS TRANSPOSE, TENSOR INSERTION, AND TENSOR EXTRACTION.

- Once mapped, Corsair exploits the streaming nature of LLMs by unrolling the ISA graph temporally in hardware.
- All data buffers and residency lifetimes are explicitly managed by software.
- Quads autonomously execute subgraphs without requiring lockstep synchronization with other quads.
- There is no timing assumption during execution. Every causal relationship is explicitly expressed in the ISA graph.

During load time, the model parameters (and the execution graph) are delivered into the DDR of the chip by the host. Subsequently, input activations are streamed in and results streamed out via PCIe (using GM or DDR). As part of model initialization, static parameters are read into the SRAMs (GM and/or stash) where they ideally remain resident until the end of the inference session.

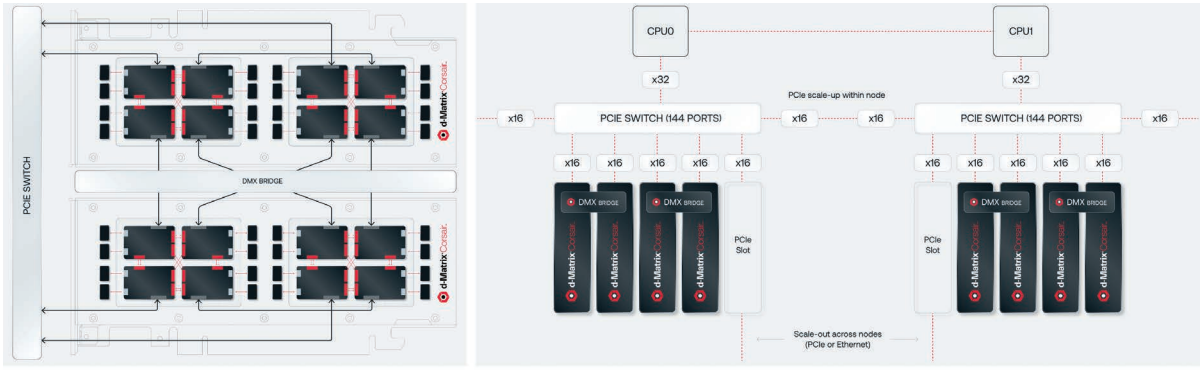


FIGURE 4. (a) A PCIe-based TP unit composed of two Corsair cards connected by the DMX bridge. (b) A schematic view of a server with eight Corsair accelerators connected over a PCIe fabric.

MULTIDEVICE SCALING

Recall that tensor parallelism and pipeline parallelism require low-latency, high-bandwidth communication. Both one-hop tree and ring algorithms render unique characteristics to the collective operation. During token generation, collective communication involves tiny tensors and is thus heavily latency bound. Achieving ultralow latency collective execution requires a combination of latency-optimal one-hop tree algorithms and a fully connected TP unit. The Corsair system includes an abstraction that allows up to 16 chiplets to form a hierarchically fully connected logical mesh, thereby becoming a natural TP unit. This is shown in Figure 4(a). Simultaneously, it allows for multiple such TP units to form a logical pipeline as illustrated in Figure 4(b). The physical connectivity that allows low-latency, high-bandwidth communication is enabled by 1) die-to-die connections for short distances within a package and 2) PCIe links over longer distances.

Die-to-Die Interconnect

The MCM package (Figure 2) consists of four chiplets interconnected in an all-to-all mesh topology using high-bandwidth, low-latency, and low-power D2D links. These D2D links are based on ODSA BoW 1.0 spec.[12] The D2D links are implemented using conventional organic Ajinomoto build-up film (ABF) substrate technology with extended reach up to 25-mm interconnect length. This extended reach enables diagonal connections in the mesh, reducing communication latency between chiplets.

D2D links maintain signal integrity using low-loss ABF dielectric material and stacked multilayer vias. The D2D physical layer (PHY) transceiver further improves the signal integrity through flexible CLK-data skew compensation and sampler voltage detection threshold tuning that maximizes eye opening on a per-lane basis. The PHY aligns the receiver's sampling point to the center of the aggregated eye across the lanes for reliable data capture.

Each D2D PHY integrates 16 transmitter and 16 receiver lanes, with each lane operating at a data rate of 16 Gb/s. This configuration delivers a total bandwidth of 256 Gb/s per PHY, corresponding to a bandwidth density of 0.19 Tb/s/mm per direction. Notably, the achieved bandwidth density and energy efficiency demonstrate a 10× improvement over conventional PCB-level signaling technologies. By aggregating four D2D PHYs per link, an inter-die bandwidth of 1 Tb/s is achieved (128 GB/s) in each direction between any two pair of dies at 0.35 pJ/bit energy efficiency.

By leveraging conventional packaging technology, our D2D solution is simpler to implement at an affordable cost, unlike the designs that depend on multireticle stitched silicon interposers and advanced packaging. This innovative chiplet arrangement seamlessly aggregates memory bandwidth and capacity across every chiplet in the package, empowering TP configurations for advanced distributed compute in Corsair's architecture.

PCIe Fabric

When connecting chiplets that span more than one MCM, we utilize standard PCIe interfaces. Each Corsair chiplet carries a Gen5 PCIe port resulting in a total of eight ports on a Corsair card. The usage of these

ports is shown in Figure 8 where two Corsair cards are connected to a host as well as to each other using the DMX bridge connector. There are two methods in which Corsair chiplets establish peer-to-peer data transfers over PCIe:

- *PCIe switch*: This allows connecting multiple cards in an any-to-any connectivity matrix. Transfers can happen between any two devices but incur the latency of transiting through the switch. Typically, chiplets linked to the main card connector (gold fingers) use switch-based peering.
- *PCIe back-to-back*: This allows direct connection of two chiplets using either a board trace or the DMX bridge. These fixed connections do not incur the switch latency but are limited to at most two cards.

The back-to-back PCIe links (along with D2D links within the MCMs) result in the 16-chiplet logical TP unit described earlier. At the same time, switch-based connection provides the scalability to form more complex multicard topologies particularly using multiple levels and hierarchical schemes.

SOFTWARE ABSTRACTION

Corsair architecture is programmed using the Aviator software development kit. While the details of the Aviator stack are beyond the scope of this article, it is instructive to highlight features that involve significant hardware–software co-design.

ISA Representation

Corsair exposes a compact instruction set (ISA) for interacting with the various hardware units. These instructions are categorized into 1) data movement, 2) computation, 3) reshaping, and 4) control operations. Any workload can be composed by adequate combination of these instructions, normally by the Aviator compiler. An ISA graph representing a workload can be expressed as a directed acyclic graph, which captures the node-level dependencies as graph edges or a unique hierarchical queue data structure, which spatially spreads concurrent operations and registers dependencies using serial time steps and explicit node edges.

Data Movement and In-Graph Collectives

Corsair is a data flow architecture where the producer–consumer relationships are readily captured in the execution graph with the rare need to store intermediates into memory. Consequently, a majority of data movements are push based where the producer writes its output directly into the consumer's buffers. The readiness of the consumer and availability of allocated buffer space is ensured by software. In fact, such transfers do not involve any handshake between the sender and receiver irrespective of their distance. The key insight is that the only entity requiring knowledge of the transfer completion is the downstream operation at the receiver. Hence, a local barrier associated with the data transfer identity is used. This local barrier is removed as soon as the data are received.

Unlike traditional collective communication library[10] operations which are invoked as a host-based function, our architecture allows collective operations to be baked directly into the execution graph. Such operations leverage an autonomous transfer protocol and occur directly between peer devices in a Corsair-based system. This eliminates host CPU interaction and expensive synchronizations, leading to ultralow latency collectives and greater opportunity for overlap of computation and communication phases of execution.

Control Flow and Autoregression Engine

Corsair follows a fine-grained control flow where every operation is explicitly scheduled. Dependent operations are triggered only after upstream operations are completed. Completion signals typically travel back to the control processor to allow subsequent work dispatch. However, this adds considerable turn-around latency between consecutive tasks. Instead, Corsair defines a dedicated unit for work dispatch and completion processing–"dispatch engine" to assist the control processor. Furthermore, operations are arranged in a unique hardware data structure that maximizes concurrent execution without breaking graph causality. This design allows a single CPU to control over 50 TOPS of compute without incurring a large area cost of control logic.

Autoregression and caching of key-value elements are hallmark features of large language models. Corsair provides a novel solution for both KV caching and constantly changing context length without requiring a recompilation of the graph. This is made possible by our representation of workloads using Corsair ISA and a multistage compilation flow where the final resolution of variables occurs on-device prior to execution. Hence, a single compiled graph can automatically grow in instructions, memory usage, and compute units based on the parameters of a specific iteration.

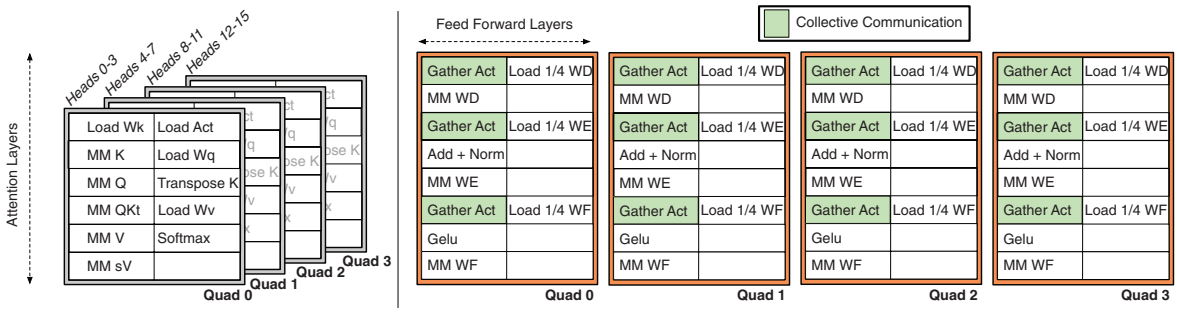


FIGURE 5. Illustration of spatiotemporal mapping of an LLM decoder onto four quads of a Corsair chiplet.

Input Length_Output Length	O/p token throughput advantage		Latency Advantage	
	Per card	*Per card per watt*	*TPOT*	*TTFT*
256_256	1.8x	2.1x	6.4x	4.1x
512_128	1.6x	1.9x	7.3x	6.2x
128_1024	0.8x	1.0x	13.3x	21.3x
4096_256	2.7x	3.2x	10.8x	5.7x
2048_2048	1.5x	1.7x	11.7x	9.4x
4096_4096	2.2x	2.6x	8.8x	4.4x
1024_3072	1.2x	1.4x	9.4x	6.8x
3072_1024	2.3x	2.7x	9.2x	3.4x
16384_16384	2.5x	2.9x	10x	1x
Average	**1.8x**	**2.2x**	**9.7x**	**6.9x**

(a)

M	K	N	Cycles	Utilization
64	1024	1024	3444	60%
64	2048	2048	10708	77%
64	4096	4096	39344	84%
128	1024	1024	5932	70%
128	2048	2048	20252	82%
128	4096	4096	77524	85%
1024	1024	1024	39324	84%

(b)

M	K	N	Utilization	Total Power 1167 MHz (W)	DIMC TOPS/W	
					1167MHz	*800MHz*
1	4096	4096	0.05	24	11.7	17.4
4	4096	4096	0.22	26	31.1	46.8
8	4096	4096	0.39	35	27.3	41
16	4096	4096	0.57	35	34.3	50.7
32	4096	4096	0.72	46	23.7	35.4
64	4096	4096	0.83	48	23.8	35.3

(c)

FIGURE 6. (a) Per-card output token throughput, latency, and efficiency advantage of Corsair versus comparable GPUs. (b) Cycle counts and TOPS utilization for GEMMs of various dimensions (M*K*N) in one quad. (c) Impact of scaling activation rows M on DIMC utilization, power, and efficiency.

RESULTS AND ANALYSIS

The key evaluation metrics of an accelerator for AI workloads are computing TOPS, memory bandwidth, utilization, scalability, and energy efficiency. We present results of targeted benchmark and ML model execution measured on Corsair *A0* silicon. We then extend these benchmarks to generative and reasoning AI models.

Evaluation Methodology

Measurements on silicon are obtained by running workloads represented as ISA graphs and recording adequate metrics. Specifically:

1) Microbenchmarks are defined to exercise specific attributes of the SoC including both the functional operations and their spatiotemporal mapping over desired number of chiplets.
2) A corresponding ISA graph is generated by compiling operations with dependencies and resource bindings.
3) The Corsair performance monitoring unit provides built-in counters to measure elapsed cycles for every ISA operation including latencies of launch and completion signaling.
4) Workload time is the cycles elapsed from the launch of the first operation until the completion of the last operation. Effective TOPs is the number of math operations in the computation $[(2K - 1)]$ over the time taken (at a clock rate of 1.167 GHz). Utilization is defined as the ratio of the effective TOPs to the roofline TOPs of a compute unit (DIMC) with an ideal 512 cycles to complete a $64 \times 512 \times 64$ matrix multiplication.

Microbenchmark Results

Figure 6(b) shows the cycle counts for matrix multiplication operations of different dimensions executed on a single quad of Corsair. The cycle counts include the time overhead required to dispatch instructions from the control CPU, transfer time of the activations and weights from SRAM into the compute cores, matrix multiplication, and transfer time of the results back to SRAM. The matrix multiplications are distributed spatially and/or temporally to run on an appropriate

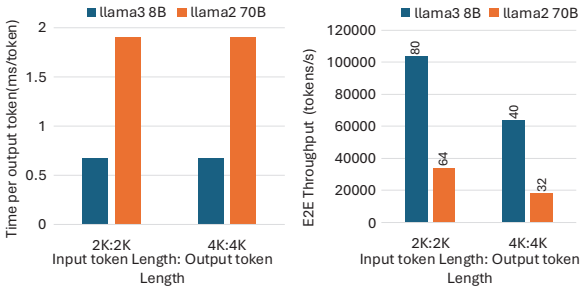


FIGURE 7. Latency and throughput with Corsair for 8B and 70B LLaMA models. The labels on the bars on the throughput chart indicate the batch sizes used.

number of cores within a Quad. Furthermore, the transfer and computation cycles are pipelined for high compute utilization, resulting in linear scaling of latency with computation size. As the size of the matrix multiplication increases, the time overheads are amortized over larger compute times.

Figure 6(b) shows compute utilization, total power, and DIMC TOPS/W for matrix multiplications whose weights are fixed at (4096,4096) and the rows of the activation matrix sweep from 1 to 64, representing typical matrix sizes in token-generation phase of LLMs where the rows correspond to batch size. We observe that Corsair achieves compute utilization of nearly 71% at batch size 32. Achieving such a high utilization with such low values of M is possible primarily due to the high memory bandwidth available on Corsair. Figure 6(c) also shows the power efficiency of Corsair at 800 MHz to be 35.3TOPS/W which is on par with SOTA[11] efficiency for an 8-b implementation while being part of a high-volume product.

LLM Performance Analysis

Consider a small LLM decoder comprising attention heads and feed-forward layers. Figure 5 shows an example of how such a decoder would be mapped onto a Corsair chiplet such that the four quads form a TP configuration. The mapping exploits spatial resources for concurrent attention head processing and sharded linear layers processing. It further illustrates the pipelining opportunities during the different operations. Note that overlapping is performed in a fine-grained manner such that downstream operations on a subtensor begin without waiting for the entire operation to finish. As an example, a full bidirectional encouder representation of transformers large encoder running on a single Corsair quad with context size of 384, finishes in about 296k cycles. This translates to an inference throughput of 5400

DIMC Cores	2048
Perf Memory	2GB, 150TB/s
Capacity Memory	Up to 256GB LPDDR5, 400GB/s
Data Types	Native formats: BFP16, BFP12, BFP24 Supported industry formats: MXINT16/8/4
Compute TFLOPS	2400 (8-bit dense) 9600 (4-bit dense)
Form Factor	PCIe FHFL
Connector	PCIe Gen5: 128GB/s bidirectional DMX Bridge: 512GB/s bidirectional
Security	Secure boot
Management	Redfish, PLDM, SPDM (in dev)
TDP	600 W
Thermals	Air cooling (dual slot)

FIGURE 8. A Corsair PCIe card showing the two MCMs and the brief specifications of the card.

inferences/s on a single corsair card at 1.167 GHz and a batch of 32.

The TP execution of the previous decoder extends to a Corsair TP unit and multiple such TP units are connected in a pipeline parallel manner to orchestrate full model execution. The cumulative effect of efficient compute cores, very large bandwidth, fast interconnections, and software techniques is a step-function improvement in both throughput and latency. We estimate the performance of larger LLMs with an in-house performance model that is correlated to silicon. Figure 7 shows the latency and throughput of 8 billion and 70 billion parameter models.

Figure 6(a) provides a comparison of llama 70B performance metrics between 64xCorsair (TP2-PP32) and 8xH100 (TP-8-PP1). The throughput and throughput/watt metrics are normalized per card. The latency on Corsair on average is ~9.7× better than a typical GPU while generating ~1.8× better system throughput per card. Normalizing for TDP (700 W for H100 versus 600 W for Corsair) gives ~2.2× energy-efficiency advantage, impacting operating costs. Under best throughput configurations, Corsair shows an average of 6.9× advantage over H100. The prefill phase is compute-bound; Corsair's 2400 MXINT8 TOPS compare favorably to H100's 1979 FP8 FLOPs. Furthermore, we aggregate the compute of 64 cards versus H100's eight cards. While it is possible to get a similar TTFT performance with a larger (~64) H100s, it is not a typical deployment scenario due to other reasons such

as host complexity and costs. Additionally, H100 can also reduce TTFT by lowering the batch size, but this degrades its throughput metric, increasing Corsair's advantage on that axis.

Discussion and Future Work

A Corsair PCIe card and its specification is shown in Figure 8. Having established the key attributes of an LLM inference accelerator, Corsair's performance benefits are per expectations. For inference, batching is often a tradeoff between latency and throughput. With corsair's low arithmetic intensity and high bandwidth, we can reach high utilizations (~70%) with a relatively small batch of 32. This provides an advantage compared to GPUs which needs a much larger number of concurrent users to get similar utilizations. Coupled with industry-leading TOPS/W, Corsair can deliver a lower total cost of ownership with a much lower concurrent user count compared to GPUs, making GenAI deployments economically viable at scale.

While beyond the scope of this article, Corsair-based systems scale seamlessly to multiple racks via a combination of network interface card top of rack and TOR switches to enable disaggregated deployments for even larger models. For future generations of hardware, increasing memory capacity while keeping the effective bandwidth similar would reduce the number of cards needed per deployment and increase utilization and efficiency. One such potential technological directions to reach the aforementioned capacity and bandwidth targets would be 3-D stacking memory on top of logic dies.

CONCLUSION

We present Corsair—a novel computing architecture and system designed to accelerate generative AI efficiently and at scale. We characterize the inference-time computational properties of language and reasoning models, highlighting their compute- and memory-bounded phases of execution. We discuss how Corsair addresses these workload needs through energy-efficient tensor computations, fast interconnects, enormous memory bandwidths, and block floating-point numerics. We present benchmarking results that show sustained high compute throughput, energy efficiency, and memory bandwidth in our A0 silicon. A purpose-built system like Corsair that optimizes for latency-bounded throughput unlocks unprecedented levels of performance and greatly improve the economic viability of language, reasoning, and agentic workflows.

ACKNOWLEDGMENTS

The authors sincerely appreciate the contributions, feedback, and discussions from colleagues and collaborators. In particular, we thank Aayush Ankit, Marco Heddes, Irene Quek, Matthew Chapman, Cliff Young, Prashant Nair, Sai Rahul Chalamalasetti, Timothy Heil, Cheng Ru Ho, and Ramyad Hadidi.

REFERENCES

1. DeepSeek-AI et al., "DeepSeek-V3 technical report," 2024, *arXiv:2412.19437*.
2. J. Kaplan et al., "Scaling laws for neural language models," 2020, *arXiv:2001.08361*.
3. E. Beeching, L. Tunstall, and S. Rush, "Scaling test-time compute with open models," *Hugging Face*, Dec. 16, 2024. [Online]. Available: https://huggingface.co/spaces/HuggingFaceH4/blogpost-scaling-test-time-compute
4. H. Lightman et al., "Let's verify step by step," 2023, *arXiv:2305.20050*.
5. A. Gholami, Z. Yao, S. Kim, C. Hooper, M. W. Mahoney, and K. Keutzer, "AI and memory wall," *IEEE Micro*, vol. 44, no. 3, pp. 33–39, May/Jun. 2024, doi: 10.1109/MM.2024.3373763.
6. D. Abts et al., "Think fast: A tensor streaming processor (TSP) for accelerating deep learning workloads," in *Proc. 47th ACM/IEEE Annu. Int. Symp. Comput. Archit. (ISCA)*, 2020, pp. 145–158, doi: 10.1109/ISCA45697.2020.00023.
7. B. Darvish Rouhani et al., "Microscaling data formats for deep learning," 2023, *arXiv:2310.10537*.
8. C. Wolters, X. Yang, U. Schlichtmann, and T. Suzumura, "Memory is all you need: An overview of compute-in-memory architectures for accelerating large language model inference," 2024, *arXiv:2406.08413*.
9. I. Soloveychik, I. Lyubomirsky, X. Wang, and S. Bhoja, "Block format error bounds and optimal block size selection," 2022, *arXiv:2210.05470*.
10. M. Hidayetoglu et al., "HiCCL: A hierarchical collective communication library," 2024, *arXiv:2408.05962*.
11. B. Keller et al., "VLSI: A 17–95.6 TOPS/W deep learning inference accelerator with per-vector scaled 4-bit quantization for transformers in 5nm," in *Proc. IEEE Symp. VLSI Technol. Circuits (VLSI Technol. Circuits)*, 2022, pp. 16–17, doi: 10.1109/VLSITechnologyandCir46769.2022.9830277.
12. "Bunch of wires (BoW) PHY specification," Open Compute Project, Austin, TX, USA, Mar. 2023. [Online]. Available: https://www.opencompute.org/documents/bow-specification-v2-0d-1-pdf

SATYAM SRIVASTAVA is a distinguished architect with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research

interests include hardware-software architecture, machine learning, signal processing, distributed systems, and sustainable computing. Srivastava received his Ph.D. degree in electrical and computer engineering from Purdue University. Contact him at y2satyam@gmail.com and ssrivastava@d-matrix.ai.

AKHIL ARUNKUMAR is a senior principal engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include machine learning systems, inference acceleration, distributed frameworks for ML execution, hardware-software co-design and computer architecture. Arunkumar received his Ph.D. degree in computer science from Arizona Statue University. Contact him at aarunkumar@d-matrix.ai.

NITHESH KURELLA is a senior principal engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include acceleration of ML inference, system design for data center, hardware-software co-design, and computer architecture. Kurella received his Master's degree in computer engineering from the University of Massachusetts. Contact him at nitheshk@d-matrix.ai.

AMRIT PANDA is a senior principal engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Panda received his Ph.D. degree in computer architecture from Arizona State University. Contact him at apanda@d-matrix.ai.

GAURAV JAIN is a technical director of software with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Jain received his Master's degree in computer architecture from the University of Wisconsin. Contact him at gauravj@d-matrix.ai.

PURUSHOTHAM KAMATH is a vice president of software with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Kamath received his Ph.D. degree in computer engineering from the University of Southern California. Contact him at ukamath@d-matrix.ai.

MARK WUTZKE is a distinguished engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include network and security protocols, with a focus on embedded solutions. Wutzke received his Bachelor's degree in computer science from the University of Technology, Sydney, Australia. Contact him at mwutzke@gmail.com and mwutzke@d-matrix.ai.

ARUN TIRUVUR is a vice president of design with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Tiruvur received his Master's degree in electrical engineering from Kansas State University. Contact him at atiruvur@d-matrix.ai.

MYANK (MIKE) GUPTA is a distinguished engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include tinkering and building on a variety of engineering projects. Gupta received his M.S. degree in electrical engineering from the University of Michigan. Contact him at mgupta@d-matrix.ai.

ILYA SOLOVEYCHIK is a principal engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include algorithmic optimizations allowing efficient AI inference on high-throughput hardware. Soloveychik completed his postdoctoral research at Harvard School of Engineering. Contact him at ilyas@d-matrix.ai.

VAMSI DARSI is a principal hardware design engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include computer architecture for AI accelerators, control path designs, and memory subsystem designs. Darsi received his Master's degree in VLSI technology from JNTU Hyderabad, India. Contact him at vdarsi@d-matrix.ai.

MALAV DALAL is a principal design verification engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include the engineering of distributed systems, system-level communications, protocols like PCIe, and modern silicon and system architectures. Dalal received his M.Tech. degree in microelectronics from BITS, Pilani, India. Contact him at mdalal@d-matrix.ai.

VINAYAK PATANKAR is a vice president of design validation with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Patankar received his Bachelor's degree from Shivaji University. Contact him at vpatankar@d-matrix.ai.

SASIDHAR DUDYALA is a senior principal verification engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include designing and optimizing hardware infrastructure for scalable, high-performance AI systems, with a focus on AI accelerators, memory architectures, interconnects, and system-level co-design. Dudyala received his Master's degree in computer system engineering from University of East London. Contact him at sdudyala@d-matrix.ai.

SENTHIL DURAISAMY is a senior principal design verification engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include building optimum architecture for front end verification and validation of SoC systems (accelerators and switches). Duraisamy received his B.E. (Honors) degree in electrical and electronics from BITS, Pilani, India. Contact him at senthild@d-matrix.ai.

SANTHOSH RAMCHANDRAN is a principal engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include reference modeling, hardware-software cosimulation, and computer architecture for AI inference. Ramachandran

received his M.S. degree in computer engineering from North Carolina State University. Contact him at santoshr@d-matrix.ai.

RAGHAV VENKATASUBRAMANIAN is a senior staff software engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include computer architecture, efficient kernel design for machine learning workloads, and hardware-software co-design. Venkatasubramanian received his M.S. degree in computer engineering from Purdue University. Contact him at raghavv@d-matrix.ai.

YUWEI QIN is a senior staff engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include accelerating AI inference across edge and data-center applications through system-level co-design. Qin received a Ph.D. degree in electrical and computer engineering from Carnegie Melon University. He is a Member of IEEE. Contact him at yuweiq@d-matrix.ai.

XIN WANG is a director of machine learning with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include machine learning, data science, systems, and computational neuroscience. Wang received his Ph.D. degree in neuroscience from the University of Southern California. Contact him at xwang@d-matrix.ai.

JAYAPRAKASH BALACHANDRAN is a technical director of advanced packaging with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Balachandran received his Ph.D. degree from KU Leuven. Contact him at jayap@d-matrix.ai.

ALI MURAT GOK is a senior staff kernel engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include kernel design and optimization for ML/AI workloads, hardware-software co-design and computer architecture. Gok received his Ph.D. degree in computer architectures from Northwestern University. Contact him at agok@d-matrix.ai.

PIOTR WOJCIECHOWSKI is a principal software engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include inference optimization techniques for deep neural networks. Wojciechowski received his B.Sc. degree in computer science from the University of Warsaw. Contact him at piotrw@d-matrix.ai.

SALIYA EKANAYAKE is a senior principal software engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include AI systems, distributed computing, and high-performance software architecture. Ekanayake received his Ph.D. degree in computer science from Indiana University. Contact him at esaliya@d-matrix.ai.

CHRIS NG is a senior staff software engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Ng received his Master's degree in computer engineering from the University of British Columbia. Contact him at cng@d-matrix.ai.

RANJU SARMA is a technical director of design with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Sarma received her Master's degree in electrical engineering from Missouri University of Science and Technology, and her Master's degree in physics from Guwahati University. Contact her at rsarma@d-matrix.ai.

SHUBHANKIT RATHORE is a staff engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include ML systems, high performance computer architecture, and performance modeling and optimization for ML/AI workloads. Rathore received his Master's degree in computer science from the University of Wisconsin-Madison, and his Master's degree in electrical engineering from IIT-Bombay. Contact him at shubhankitr@d-matrix.ai.

TRISTAN TROUWEN is a staff software architect with d-Matrix Corporation, Santa Clara, CA, 95054, USA. Trouwen received his Master's degree from Eindhoven University of Technology. Contact him at ttrouwen@d-matrix.ai.

SIWEI ZHUANG is a principal software engineer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include operating system, microkernel, real-time system, and CPU architecture. Zhuang received his Master's degree in information technology from the University of New South Wales. Contact him at szhuang@d-matrix.ai.

CHRIS NICOL is a vice president with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include computer and silicon systems. Nicol received his Ph.D. degree from the University of New South Wales. Contact him at cnicol@d-matrix.ai.

SUDEEP BHOJA is the chief technology officer with d-Matrix Corporation, Santa Clara, CA, 95054, USA. His research interests include creating solutions for generative AI inference. Bhoja received his Master's degree in electrical engineering from Purdue University. Contact him at sbhoja@d-matrix.ai.