Smoothing Disruption Across the Stack: Tales of Memory, Heterogeneity, & Compilers

M. Niemier, Z. Enciso, M. Sharifi, X.S. Hu
University of Notre Dame
Notre Dame, IN USA
{mniemier, msharif1, zenciso, shu }@nd.edu

Ian O'Connor
Ecole Centrale de Lyon
Lyon, France
Ian.Oconnor@ec-lyon.fr

A. Graening, R. Sharma, P. Gupta
University of California at Los Angeles
Los Angeles, CA USA
{agraening, sravit, puneetg}@ucla.edu

J. Castrillon, J.P.C. Lima,
A.A. Khan, H. Farzaneh
TU Dresden
Dresden, Germany
{jeronimo.castrillon, joao.lima,
asif_ali.khan, hamid.farzaneh}@tu-dresden.de

N. Afroze, A. Khan
Georgia Institute of Technology
Atlanta, GA USA
{nafroze, akhan40}@gatech.edu

Julien Ryckaert
IMEC
Leuven, Belgium
Julien.Ryckaert@imec.be

***Abstract*—Multiple research vectors represent possible paths to improved energy and performance metrics at the application-level. There are active efforts with respect to emerging logic devices, new memory technologies, novel interconnects, and heterogeneous integration architectures. Of great interest is quantifying the potential impact of a given solution to prioritize research vectors accordingly. In this paper, we discuss two efforts – one focused on emerging memory technology, and another focused on heterogeneous integration technology – that speak to best practices for, and needed contributions from the design automation (DA) community to explore this vast design space. Furthermore, we highlight new research efforts that aim to develop the novel compiler abstractions and frameworks that are ultimately needed to derive maximum value from new memory and/or heterogeneous and monolithic integration architecture, and that can also play an important role with respect to design space exploration efforts.**

***Index Terms*—Compute-in-memory; associative memory; crossbar architectures; ferroelectric field effect transistors (FeFETs); distance functions; heterogeneous integration; chiplet; pathfinding; compilers**

I. Introduction

There is growing interest in **memory technology** that can **(1)** *expand the memory hierarchy* – e.g., serve as storage-class memory between DRAM and SSDs [1], **(2)** *increase storage density* given growing dataset sizes in artificial intelligence (AI), privacy preserving computation (PPC) [2], etc., and **(3)** *enable in situ computation* to reduce overheads associated with data transfer [3] – e.g., in-memory computing (IMC) fabrics including **(a)** *computing at the periphery (CAP)* that exploits internal memory bandwidth to achieve parallelism/perform Boolean operations [4], [5], **(b)** *analog crossbar arrays* for highly parallel matrix-vector multiplies [6]–[8], and **(c)** *content addressable memories (CAMs)* that perform parallel search operations in the memory per an input query and a desired association/matching function [9]. Indeed, recent work suggests there is great promise from denser memories and/or individual IMC solutions for transformer networks [10]–[12], recommendation systems [13], PPC [14], etc.

Simultaneously, there has been a rapid expansion with respect to the capabilities/promise of **heterogeneous and monolithic integration architectures** that aim to deliver solutions with 1 billion transistors/mm^2 at high energy efficiency. Said systems will transcend horizontal/vertical scales, and offer the potential for the integration of many diverse technologies owing to monolithic 3D (M3D) tiers and chiplets. Increased connectivity should positively impact computing, storage, sensing, and communication alike. Heterogeneous integration (HI) represents another path toward improvements for compute density and energy efficiency for application-level workloads such as transformer networks [15], etc.

Ideally, application-level improvements would be achieved *with minimal "disruption" to existing architectures and fabrication process flows.* Therefore, it is of great interest to benchmark "deltas of improvement" that advances in either memory and/or integration may promise with respect to latency, energy, as well as the "fidelity" of the result produced (e.g., accuracy in AI workloads). As a representative example, Fig. 1 illustrates different architectural/memory-centric design points that could be employed to accelerate a memory augmented neural network (MANN) [16] model. Peak accuracies for selected design points are noted. Regions along each axis qualitatively consider "disruption" from the state-of-the-art. Drawing from a Sec. III case study, Fig. 1 illustrates potential savings in search energy for the MANN workload when transitioning between design points.

Per Fig. 1, when ferroelectric devices [17] are employed as a binary or multi-bit random access memory (RAM), search *energy* savings are negligible (Fig. 1-A). However, when (larger) ferroelectric devices are used to implement a multi-bit CAM (MCAM) that performs in-situ Euclidean distance [18], energy savings are more substantial owing to reduced data transfer (Fig. 1-B). However, devices are more *scalable*

This work was supported in part by **(1)** ASCENT, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA; **(2)** CHIMES and SUPREME, two of seven centers in JUMP 2.0, a Semiconductor Research Corporation (SRC) program sponsored by DARPA; **(3)** the German Research Council (DFG) through the HetCIM project (502388442) under the Priority Program on 'Disruptive Memory Technologies' (SPP 2377), and **(4)** the German Federal Ministry of Education and Research (BMBF) within the project ScaDS.AI (BMBF 01IS18026A-D).

979-8-3503-4859-0/DATE24/© 2024 EDAA
Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

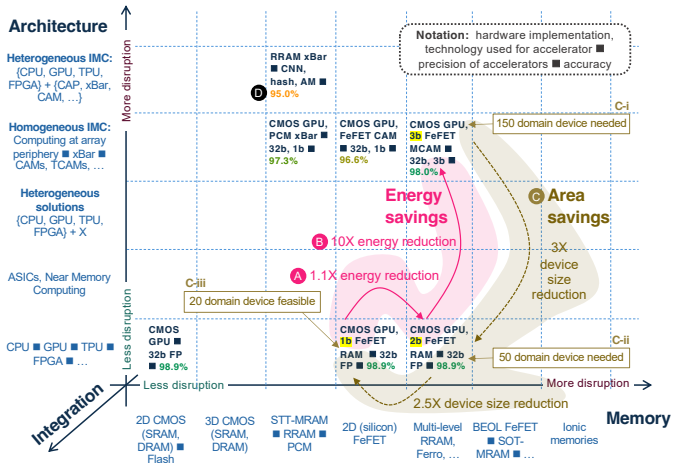

Fig. 1. Architectural/technological solutions for a given workload; representative FOM (accuracy, search energy, device scaling) are illustrated. Interestingly, for ferroelectric solutions, substantial energy savings are possible for search operations even with larger device size requirements.

when used as conventional memory, as larger devices are needed to achieve iso-accuracy when employed to implement an MCAM (Fig. 1-C i-iii). Projected drops in *accuracy* for RRAM-based solutions (Fig. 1-D) [19] might demand more substantial improvements with respect to energy and latency (or improvements in accuracy) to justify migration.

Furthermore, in Fig. 1, area projections for ferroelectric domains are captured as a function of the number of ferroelectric domains that might comprise a device (see Sec. II-A). Any IMC-like solution will also be comprised of peripheral circuitry that must be included to realize a desired compute functionality, and could have an outsized impact on different FOM (i.e., peripherals may dominate area overhead, thereby minimizing the potential impact of device scaling). One can envision similar tradeoffs in an integration design space axis.

Finally, despite exciting breakthroughs with respect to memory/integration technologies, programming models are typically low-level and specific to particular system implementations. Developing novel **compiler abstractions and frameworks** is essential to derive maximal value from this space, as widespread adoption will be influenced by the software ecosystem. As intelligent memory and integration technology are highly heterogeneous, and may speak to targeted or more general-purpose workloads, compiler abstractions/frameworks that both enable device agnostic and device specific optimizations are needed. We must design hierarchical models that allow for reasoning about computational primitives at the right level of abstraction. Progressively refining abstractions in a compiler framework will enable the identification of the most suitable target for each primitive in a given application. Transformations at different abstractions to optimize for individual memory and integration technologies are needed. Device and architecture models will both enable automatic compiler re-targetability, and also allow for system-level design space exploration.

In this paper, we highlight the promise of, challenges for, and pressing needs and interests for the memory technology and heterogeneous integration themes introduced above. We will discuss strategies, best practices, needs, and new engagements needed from the design automation (DA) community to complete roadmaps like the framework in Fig. 1. Of particular interest is engagement with researchers at *lower levels of the stack* – e.g., per the memory technology case study, ferroelectric domain size will not only impact the "real estate" of a given functional unit, but also application-level accuracy. We must also *move up the stack* and feed information about expected device behavior to compiler developers such that software-to-hardware mappings can be considered in terms of **(a)** meeting functional computational needs, and **(b)** delivering the necessary fidelity required for a given task. Said activities will identify optimal design investments for industry with respect to deriving maximal near-term, mid-term, and long-term impacts for application-level workloads of interest.

Our discussions are organized as follows. In Sec. II, we briefly review relevant background related to the IMC- and heterogeneous integration (HI)-centric case studies that motivate this work. In Sec. III, we consider how associative memory (AM) technologies may be applied to efficiently address AI-type workloads. CMOS solutions, as well as approaches based on emerging technologies are considered. Of particular interest are the prospects for reliably realizing said functionality with respect to advances in materials/device research, as well as projections for FOM such as area/cost, energy efficiency, compute fidelity, etc. In Sec. IV, HI solutions are investigated with respect to both cost and performance in the context of large language models (LLMs). Then, in Sec. V, we discuss programmability challenges associated with novel architectures, emphasizing the need for novel, high-level programming frameworks that make systems accessible to non-experts, and automatically enable device-aware, and device-agnostic optimizations.

II. Background

Given the IMC-focus of Secs. III and V, we highlight emerging memory technologies in Sec. II-A, and IMC architectural solutions in Sec. II-B. Integration technology is briefly reviewed in Sec. II-C.

A. Logic and Memory Technology

An IMC case study in Sec. III considers ferroelectric field effect transistors (or FeFETs). The structure of an FeFET resembles a MOSFET, except a layer of ferroelectric (FE) oxide is deposited in the gate stack. Due to the coupling between the FE and CMOS capacitances, the threshold (turn-on) voltage of a device can be shifted. This effect can be used to (non-volatilely) store information in the FeFET. FeFETs can store multiple V_{TH} levels through partial polarization switching of the FE layer [8], [20], [21]. Si FeFETs require relatively high write voltage pulses, may have limited write endurance, and large read-after-write latencies. Achieving a suitable/reliable memory window is challenging – e.g., per Fig. 2a, as devices scale, the memory window may "collapse" [22]. Per Fig. 2b, 2-bit or 3-bit cells may be achievable (typically with write-and-verify programming schemes), although some cell state overlap may still exist (which may or may not be tolerable at the application-level). Additionally, low voltage, high speed memory operations

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

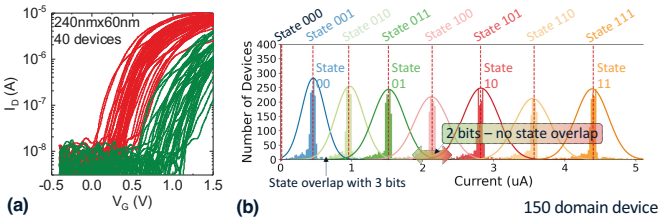

Fig. 2. **(a)** Memory window with scaled devices; **(b)** cell state overlap.

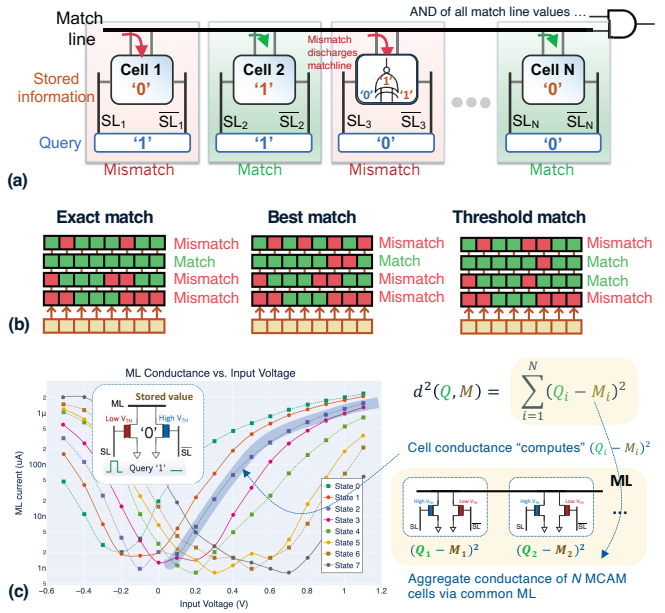

Fig. 3. **(a)** CAM functionality; **(b)** exact, best, threshold match; **(c)** approximate quadratic conductance [18] associated with query/stored value comparison can be used to approximate Euclidean distance.

with high write endurance have been demonstrated using BEOL compatible FeFET devices by eliminating the defective interlayer between the FE and channel [23]. Reported FOM include **(1)** 1.2V memory windows, **(2)** write latencies of 20 ns with $\pm$2V write pulses, **(3)** read-after-write latencies of $<$200ns, **(4)** write endurance cycles exceeding 5×10^{10}, and **(5)** multi-bit/cell programming capability. Analysis suggests that BEOL FeFETs are promising for logic-compatible, high-performance on-chip memories and multi-bit cells for IMC accelerators [24].

B. IMC Architectures

1) Associative Memories: In conventional memory, data is retrieved from a given address. In a CAM, data is supplied to memory and all entries that best match a provided query can be returned. Per Fig. 3a, in a CAM, each bit of a query is XNORed with the corresponding bit of every stored entry. With a perfect match, the matchline (ML) does not discharge as cell matches function as "open switches." Mismatches function as "closed switches" causing the ML to discharge. In a ternary content addressable memory (TCAM), "don't care" states can be stored/searched, and cell-level comparisons are treated as a match regardless of values. Search operations are performed directly within the memory itself in O(1) time, eliminating expensive data transfer to a compute unit.

We can classify AMs according to **(1)** the data representation in a cell (binary, ternary, multi-bit, analog), and **(2)** the type of match performed. Fig. 3c-inset shows the most compact FeFET AM cell design that was originally proposed to accomplish exact match, best match, and threshold match TCAM functions (Fig. 3b) based on Hamming distance. (Different match types employ different sensing circuits [25].) The same cell design can also function as a multi-bit CAM (MCAM) that performs all of the aforementioned matching functions[1]. This improves storage density and enables unique, in-memory distance functions – e.g., approximations of Euclidean distance, which has utility for machine learning, AI, bioinformatics, etc. even given cell state overlap as in Fig. 2b. Per Fig. 3c, if a multi-bit state is encoded via FeFET V_{TH} values [18], as a query deviates from the stored value (in either direction), cell conductance increases quadratically, which mimics $(Q_i - M_i)^2$ of a Euclidean squared distance function. (Euclidean squared distance is a proxy for Euclidean distance [26] that is commonly employed in machine learning algorithms.) By connecting multiple MCAM cells to a common ML, we can also capture elements of summation ($d^2(Q_i, M_i) = \sum_{i=1}^{N}(Q_i - M_i)^2$).

2) Crossbar Architectures: In AI workloads, a ubiquitous computation is the multiply and accumulate (MAC) operation (e.g., $y = \sum_{i=1}^{n} w_i x_i$) that multiplies weights by inputs. Projections suggest that the energy to retrieve a weight for a MAC operation from off-chip memory may be two orders of magnitude higher than the integer MAC operation itself [27], and co-locating weight data with logic for MACs is appealing. With inputs represented as voltages on horizontal rows, emerging devices (e.g., RRAM, FeFETs, PCM, etc.) can serve as tunable resistors with multiple analog states, and the results of MAC operations are captured by the summation of currents from crosspoint connections in a given column.

C. Integration Technology

Chiplet-based systems including 2.5D (single layer of chiplets on a substrate) [28] and 3D (dies stacked one on top of the other) have an ever-increasing number of design options. The integration substrate can be silicon or glass (achieving high interconnect density at the expense of cost), a variety of organic substrates (often cheaper but with lower interconnect density and higher bonding pitch), or EMIB [29] which uses silicon bridge dies embedded in organic substrates to possibly obtain "the best of both worlds." These different options can have substantial impacts on cost and performance.

III. FeFET-based Associative Memories

We first examine the IMC case study introduced in Sec. I in more detail. Of general interest is understanding what value may be derived from technology-enabled IMC solutions – e.g., multi-bit FeFET random access memories (RAM) and/or multi-bit FeFET memories with unique compute capabilities. Of

[1]The ability to use multi-state FeFETs for analog current domain computing lies in being able to set the state of a finite number of polarization domains in the ferroelectric material in a transistor's gate stack. The number of domains in a given state directly influences a device's threshold voltage. As the number of target states increases, more domains/larger device sizes are required to mitigate stochasticity, device non-idealities, etc. in order to obtain sufficient *separation* between states to achieve a desired application-level accuracy.

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

particular interest is identifying what benefits may be derived from FeFET-centric solutions as compared to scaled CMOS solutions – especially if larger FeFET devices are needed to realize a given compute kernel with a technology-enabled IMC approach. Said studies will *quantify* potential application-level benefits (thereby "justifying" the potential value of a new logic/memory device), and also help prioritize research at the materials science/device-levels, such that technology evolves in a way that affords maximum application-level impact.

A. MANN Background

For this study, we consider how emerging devices may be used to support computations associated with MANNs [16]. Using an image classification task as an example, a MANN can employ a traditional convolutional neural network (CNN) as a *controller* that can identify N different classes of images. However, instead of feeding output neuron values to a SoftMax function to generate a probability distribution/classification, the real-valued neuron outputs are instead stored in the network's memory. When a new inference operation is performed, a real-valued output feature vector is generated, and is subsequently compared to all learned feature vectors via a distance function (e.g., cosine or Euclidean distance). The query/class combination with minimal distance can then represent the inference/prediction made by the network. An advantage of this approach is that – assuming a sufficiently large memory – the existing network controller can be shown/learn new classes post-training. An output feature vector (or vectors assuming multiple instances of the same new class) is created, and searched in subsequent inference operations.

One might employ GPUs, crossbar architectures, etc. to support matrix-vector multiplication operations that are typically associated with a CNN. In initial MANN models [16], cosine distance was typically performed to identify a best match – which necessitates the transfer of all stored/learned classes from memory to a GPU. (Data transfer overhead typically dominates figures of merit such as energy and latency [25].) The ability to perform matching functions in the memory itself is appealing as data transfer overhead could be eliminated. However, CAM matching functions were typically limited to Hamming distance (real-valued CNN outputs were hashed to binary signatures [25]), and swapping cosine distance for Hamming distance often results in degraded accuracy [18]. However, network outputs can also be quantized to N-bit numbers. Furthermore this state can be stored in a multi-bit FeFET structure, which in-turn can implement an MCAM structure capable of approximating Euclidean distance per Sec II-B. Existing work suggests that 3-bit representations are sufficient for iso-accuracy in hyperdimensional computing problems [18]; work discussed below also suggests said structures can achieve iso-accuracy for MANN workloads with 3-bit precision/some cell state overlap.

B. Initial Benchmarking

Fig. 4 captures how the number of ferroelectric domains, the read/write scheme employed, and the memory modality employed impact search energy for MANN workloads [30]. (We use SPICE and an FeFET Preisach model [31] to estimate

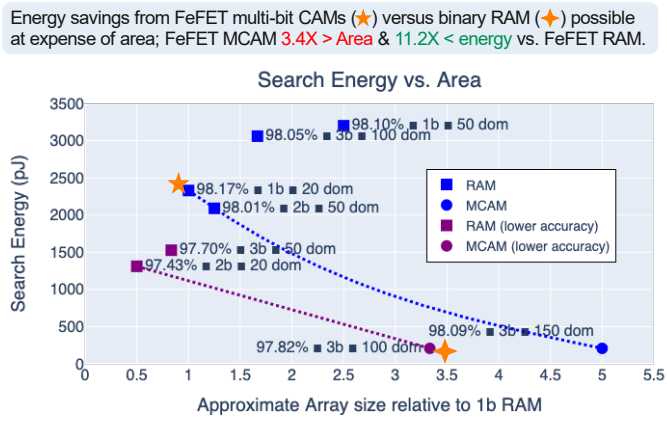

Fig. 4. Initial Pareto frontier projections; area projections are a function of the number of ferroelectric domains assumed for a device.

search energy for FeFET-based CAMs. Energy estimates consider interconnect parasitics and peripheral energies extracted from the layout.) Design space explorations (DSEs) suggest larger (150 domain) devices are needed to estimate Euclidean distance such that the application accuracy delivered by a GPU can be matched (98.0%-98.3%). As devices scale, overlap between cell states (Fig. 2b) is more likely, increasing the likelihood of mis-programmed values/accuracy degradation. Of course, FeFETs could also implement a compact RAM to store MANN data, which affords the ability to realize scaled devices. Per Fig. 4, an iso-accuracy Pareto frontier is formed by **(1)** 1b ferroelectric RAM (highly scaled, 20 domains), **(2)** a multi-bit ferroelecric RAM (2b/cell, 50 domains, but 50% reduction in the number of columns), and **(3)** the aforementioned 3b MCAM with in situ Euclidean distance (150 domains, but reduced data transfer)[2]. In all instances, write-and-verify operations were used [32], which would also increase write energy/latency. **(4)** If small accuracy degradations are acceptable, more substantial area and energy savings are possible (purple data points). *An ultimate goal is iso-accuracy solutions, with the energy/latency savings of IMC solutions, with scaled devices.*

C. Layout-based Analysis

To construct a more accurate view of the MANN landscape, we conducted a layout-based analysis that accounts for the additional area overheads associated with writing data to FeFET bitcells and performing search operations. We estimated the MCAM macro area, assuming a 100-domain device (see Footnote 3), by designing the entire macro in a commercial 65 nm process, and then scaling down the peripheral circuitry to 5 nm. To perform scaling, we compare the SRAM bitcell area for the commercial 65 nm process to the SRAM bitcell area for the same foundry's 5 nm process and divide our area accordingly. We scaled the memory array itself to match 100-domain bitcells assuming 10 nm^2 per domain [33], [34] (as this is essentially iso-accuracy with a 150 domain device—98.09% versus 97.82%—and we aim to understand the impact

[2]For cases **(1)** and **(2)** the FeFET is simply used to store data that is sent to a GPU for a distance function calculation; ignoring the potential impact of endurance, etc. In the best case, this could reduce memory footprint – e.g., requiring one device in lieu of a 6T SRAM.

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

of peripherals/physical layouts on ultimately scaled solutions). The MCAM macro has four main components: **(1)** the memory array, **(2)** match-line (ML) drivers, **(3)** data-line (DL) drivers, and **(4)** capture circuitry. Fig. 4 (inset) depicts the mirrored top half of the MCAM macro, with relevant components labeled.

The ML and BL drivers decode a digital input that represents which voltage (including a high-impedance state) to drive the line to and subsequently charge or discharge the line. To enable the write operation of the FeFET bitcells, both the ML and DL drivers must support driving the line to V_{SS}, V_{DD}, and an inhibit voltage V_I. The inhibit voltage is essential to avoid the unintentional writing of adjacent cells. Additionally, the DL drivers must support a write voltage V_W. The write voltage is often significantly higher than the maximum tolerable voltage for core FETs, so the switches that connect the DL to V_W must be made from thick-oxide FETs to prevent oxide breakdown. In addition, the signal controlling the V_W switch must be boosted from the core voltage to the write voltage, lest the switch is not fully turned on. This thick-oxide circuitry requires larger FETs and PDK-mandated spacing, which increases the area footprint of the DL drivers. If the inhibit voltages are also greater than the maximum core voltage, both the ML and DL drivers will incur similar/additional area penalties. The drive strength of the line charging switches also impacts the area of the drivers. In this study, we choose switches that can charge and discharge the MLs and DLs sufficiently fast to support 400 MHz operation.

FeFET-based architectures typically assume a negative voltage erase scheme in which drivers apply negative voltage pulses to the gate of a device to reset the polarization state [21]. However, negative voltages require reverse substrate biasing, which may cause unwanted current flow through the p-n junction of the substrate. Also, each switch that drives a negative voltage must also have a level shifter to convert the control signal, which increases driver area and power. Finally, negative voltage switches and their level shifters must be placed within an isolated deep n-well, which imposes spacing requirements between subsequent drivers and increases their effective area. For this study, we chose to eschew negative voltages entirely in favor of an array-wide macro erase, which biases the body of all FeFET bitcells to a high voltage to erase all cells simultaneously. Since this approach requires well contacts to bias the deep n-well and extra spacing, it does impose a small area overhead to the bitcell array and also limits write-verify operation[3]. It is feasible to support column-wise or row-wise erase—and thus more robust write-verify—but doing so requires separate wells, which would decrease array density. Preliminary estimates suggest an area increase of ∼2.4x for the bitcell array itself, a ∼12% area increase for the ML drivers, and an overall area increase of ∼21% compared to the macro erase approach we employ.

[3]As MANNs do not require frequent writes, this is acceptable as we could begin with an erased memory and add data to it. That said, write-verify approaches would need additional investigation, as with 100-150 domain devices, between 3 and 5 "reset" operations may be needed to set a desired polarization state to represent a given 3b value [32], which may be prohibitive given the erase scheme assumed here. In this work we intentionally assume a "best case" scenario in terms of write energy, and area efficiency for the CAM array itself to establish a "floor" for the impact of drive circuitry and layouts. We discuss impact on future materials/device research in Sec. III-D, and will revisit the notion of write-verify operations in the context of accuracy, etc.

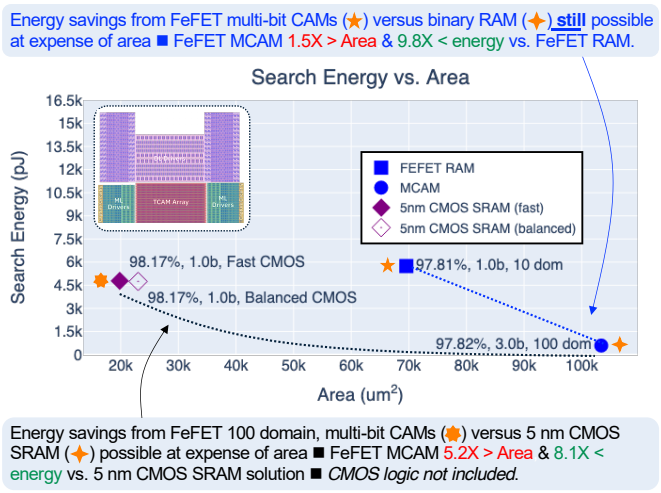

Fig. 5. Updated Pareto frontier projections.

To estimate MCAM power, we extracted parasitic capacitance and resistance from our 65 nm design, measured the power, and then scaled it to 5 nm. For FeFET RAM estimates, we use the same driver circuitry, but assume 10-domain devices (one-tenth of the memory array area, and line capacitance). We estimated the CMOS SRAM area and power by using the memory compiler for the same commercial 65 nm process, and again scaled the results. We compiled for both fast and balanced memory with the same capacity at different dimensions and chose optimally-sized memory to build the Pareto frontier.

These estimations reinforce the conclusion that FeFET devices can be Pareto-optimal compared to scaled CMOS when used in novel modes. Comparing a 3b, 100-domain FeFET MCAM to a 5 nm CMOS SRAM solution, the area of the FeFET solution is ∼5X higher, while the energy required for the FeFET solution is ∼8X lower[4]. However, scaled CMOS can outperform FeFETs when the FeFETs are used as a drop-in SRAM replacement, as the peripheral power and area dominates. As technology scaling continues, the peripheral circuitry can be made smaller, so it consumes less of the total macro area. However, the size of the FeFET MCAM bitcell may ultimately be limited by the number of domains needed to support its operation, so the area of the FeFET MCAM bitcell itself might increase relative to a scaled SRAM bitcell. Future work will identify if there is an inflection point beyond which further technology scaling favors a purely-SRAM solution. Finally, per Sec. V, advances in *compilers* may influence how frequently IMC solutions are employed in end-to-end workloads.

D. Future Research Directions

To close, we briefly discuss a subset of possible future research directions/needs given the above context. As examples, work presented in Sec. III-C considers layouts that facilitate limited write-verify operations in an effort to mitigate adverse

[4]Again, in this example, the FeFET numbers are likely optimistic owing to need for write-verify operations. However, CMOS projections are also likely optimistic as no area for logic – i.e., to calculate cosine distance – is included.

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

area overheads. That said, as noted in Footnote 3, between 3-5 reset operations [32] may be required to properly set FeFET polarization state given the device sizes considered here. We subsequently revisited software-based simulations to estimate the impact on accuracy assuming a write scheme that employed a train of pulses (where cell state could be checked between pulses) without any reset operations, and observed an accuracy of 96.29% for the 20-way, 5-shot MANN problem studied above (a reduction of ∼1.5%).

This suggests multiple research directions for future DSE and benchmarking activities that must engage researchers from across the design stack. As representative examples, assuming no changes to technology are possible (e.g., device stochasticity is not improved) we may re-target work at the *algorithmic-level* in an effort to improve accuracy with lower precision solutions (i.e., as in [35] from IBM). Alternatively, we can revisit efforts with respect to *layout and chip-design*. Preliminary analysis suggests that support for individual body biasing—each row has a separate n-well to enable write/erase operations at the cell-level—could be possible, although area will increase by ∼21%. The impact on energy, inhibit voltage schemes, etc. would also need to be revisited to determine if the design is still Pareto-optimal. Engagements with *material scientists* and *device engineers* are also possible/necessary. In this regard, the above case study suggests that some application-level workloads do not demand frequent writes. Therefore, while write voltage is an important design parameter, additional application-level benefits may be derived from device size scaling (i.e., to increase the number of domains while still promoting scaling) and/or reduced switching stochasticity. As such, we are developing a multi-pronged approach that will leverage structural metrology and microscopy techniques such as transmission electron microscopy (TEM), scanning transmission electron microscopy (STEM) coupled with differential phase contrast (DPC), high throughput 4D STEM techniques with materials modelling in tandem with device, circuit, and systems level studies.

IV. Heterogeneous Integration (HI) for Compute

We now consider a co-assessment of systems and technology in terms of cost and performance. As machine learning models such as Large Language Models (LLMs) are a primary application driver for innovation in advanced compute, we examine LLM applications on a multi-GPU setup. Below, we introduce two modeling frameworks – for performance and cost modeling – that may be employed. The chiplet cost model is used to estimate the total fabrication+assembly cost of a 2.5D/3D heterogeneously integrated system, while the *DeepFlow* system-technology co-optimization framework is used to estimate the application performance (LLM training) on a heterogeneously-integrated multi-GPU-HBM 2.5D package. The cost and performance modeling frameworks can be used to analyze the effect of technological and algorithmic changes on hardware cost and training time. In particular, we vary the types/degrees of parallelism, the inter-node bandwidth, and the IO/substrate configuration.

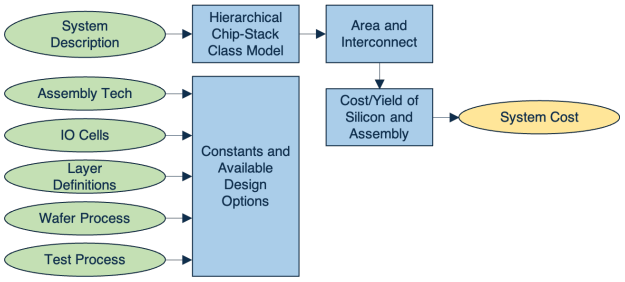

Fig. 6. Diagram of Cost Model Structure.

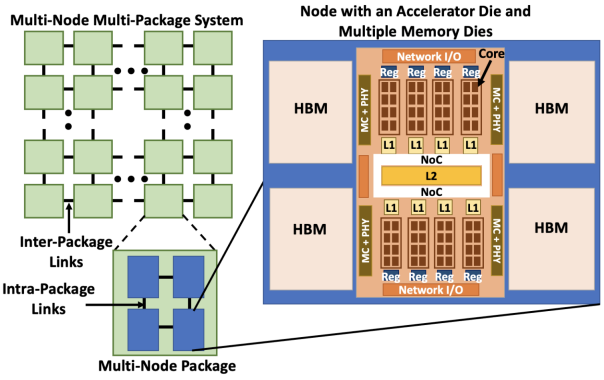

Fig. 7. Multi-package, multi-node hardware system assumed by DeepFlow.

A. Evaluating Cost of HI Systems

Our cost model is a custom Python tool based on tooling work from [36] (available at https://github.com/nanocad-lab/cost_model_chiplets.git). This tool allows us to describe different technologies and systems for overarching evaluations, and allows for the evaluation of the cost of arbitrary chiplet-based systems that utilize 2.5D integration and 3D stacking. The general format of the Python tool is shown in Fig. 6. We pass a physical system description file along with a netlist to the system, and select from a collection of defined parameters in library files for assembly technologies, IO cells, layers/technology nodes, wafer processes, and test processes. These parameters, as well as the system definition, are used to construct a model of the system and compute parameters such as area and power. These values can in-turn be used to compute system cost.

Area is calculated based on the core area of the component chips in addition to the area of any IO cells required to drive the interconnects. Additionally, some additional area may be required for fan-out if the area required for bumps is greater than the area required by the core and IO cells. Substrate area is determined by the area of stacked dies plus die spacing.

Individual die yield is calculated using a negative binomial yield model. A 2.5D/3D stack yield assumes a known good die coupled with an assembly yield model that takes the probability of individual bonds failing, as well as the probability of die misalignment.

The model allows for the definition of multiple technologies with different yield and cost parameters. The connections also include IO type definitions consisting of IO cell areas, IO reach, number of wires per IO cell, etc. to fully define the system.

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

B. Evaluating Performance of HI Systems

DeepFlow [37] is a cross-stack analysis and pathfinding framework for machine learning workloads. It addresses the low hardware utilization of AI systems due to the inefficiencies across layers of the compute stack [38]. DeepFlow can be used as **(1)** a performance-modeling framework to predict training time and core/memory utilization, and **(2)** a pathfinding framework to optimize area, power, and perimeter.

The performance-modeling framework in DeepFlow estimates training time through a two-step process. In the first step, a parallelized machine learning compute graph is mapped onto a hardware model constructed using technology, architecture, and hardware resource allocation inputs. This hardware model, which consists of a multi-node, multi-package network connected to memory, is further detailed in Fig. 7. It then uses event-driven simulation to estimate the total time taken to perform inference.

DeepFlow can also be used as a pathfinding framework, to determine the optimal set of area, power, and perimeter parameters that minimize the total training time of an LLM on a hardware system through a gradient descent approach by treating it as a constrained black-box continuous optimization problem. Prior work [37] has demonstrated the applicability of DeepFlow in predicting the effect of technology scaling, high-bandwidth memories, and parallelism strategies on execution time, as well as the performance improvement provided by increasing the number of nodes per package.

DeepFlow enables a comparison of the effect size of the different variables across different layers of the stack that collectively influence total training time of AI workloads. This enables cross-stack co-optimization, as it allows for the identification of those factors that could improve training time, and predicts/quantifies how changes impact improvement. Additionally, when paired with the cost model discussed in Sec. IV-A, we can quantify cost-benefit tradeoffs concerning changes in parallelization and hardware.

C. A Large Language Model Accelerator Case Study

We have studied a system model that is representative of 4 NVIDIA V100 GPUs in a single package [39]. The GPUs have 4 connected high bandwidth memory (HBM) chips, each with 4GB DRAM. Within a package, GPUs are physically connected in a grid, but the network employs a ring topology. As a training model, we assumed a large LSTM-based language model with a batch size of 16, a vocabulary size of 40000, 750 layers, a layer size of 1024, and a sequence length of 1. Each cell contains 4 gates, 5 non-linear operations, and 8 addition operations.

To examine the impact of different substrates and IO types, we evaluated the performance of said system assuming different inter-GPU bandwidths within the 4-GPU package, where the inter-package bandwidth is set at 50GB/s, and the inter-GPU bandwidth within the package is varied. In general, increasing bandwidth increases performance while increasing cost. In this case, we used an AIB-style IO cell [40] based on a custom reduced-size IO cell [41] to represent a parallel IO scheme, and compared it to a UCIe standard style serial interface [42]. We did this for both a low-pitched silicon substrate [28] [43] and a larger-pitch organic substrate. Two different bonding pitches for each substrate were studied.

One design option for silicon employs a parallel IO type at a 10μm bump pitch [28]. Due to the wider pitch in an organic substrate, this integration style/design option was not feasible and could not be used between the GPUs at the higher pitch. We also compared this design option to a less expensive, 45μm bump pitch silicon integration approach using a UCIe advanced style interconnect. For the results on an organic substrate, the first bar in Fig. 9 is the baseline at a 110μm bonding pitch. This design is expensive largely because of the large number of connections required by the HBM2 in the GPU. In order to investigate possible improvements, we considered a smaller bonding pitch (55μm), as well as some modifications to the HBM connection scheme. We considered **(1)** switching the HBM interface to a serial interface chosen according to the bandwidth requirements rather than the HBM spec parallel interface, **(2)** a reduced width parallel interface (also based on the application bandwidth requirements), and **(3)** a version similar to the baseline case but that used IO cells with larger reach to increase the valid region for placing bumps. This study shows that even for high bandwidths, organic substrates can be advantageous with well-chosen interconnect schemes.

Figure 10 shows how the normalized runtime of the large language model on the GPU changes as we vary parallelism strategy and intra-node bandwidth. The total number of GPUs in all cases is 4, as the product of data parallelism and kernel parallelism degree (dp and kp) is 4. Additionally, there is only one package, so all parallelism-related communication that takes place is intra-node. Higher data parallelism (and lower kernel parallelism) and higher intra-node bandwidth both lead to better runtime. However, the effect of changing the parallelism strategy is much stronger than the effect of increasing intra-node bandwidth. As a result, rather than increasing bandwidth (which would increase hardware costs), a simpler and more effective solution would be to increase data parallelism (i.e., via software). The marginal benefit of extra in-package bandwidth (and consequently any integration technology advancements) depends strongly on the nature of the application and its parallelization strategy. This simple example emphasizes the need for system-technology co-optimization spanning the entire technology-hardware-software stack, as well as for frameworks such as DeepFlow which enable such co-optimizations with consistent modeling.

V. Compilers for heterogeneous IMC systems

IMC systems excel in energy and performance efficiency across various application domains. However, owing to limited programming support, these benefits remain accessible only to a few experts after exerting significant effort. As an example, many memory technologies allow multi-state cell operation. Effectively harnessing this capability requires reasoning about device-specific compute primitives (e.g., matrix multiplication, bulk bitwise, and search operations), device-favorable access patterns, custom data types, and type conversions when mapping a kernel to an IMC system. Memory arrays may also

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

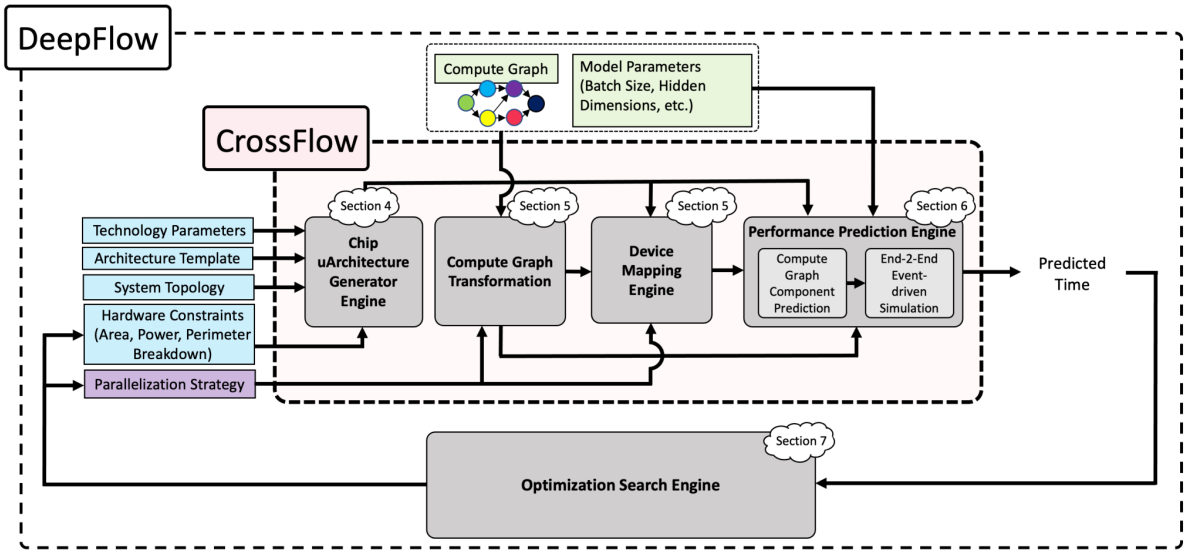

Fig. 8. Overview of DeepFlow; Relationship between microarchitectural inputs, performance prediction engine (CrossFlow), and optimization search engine.

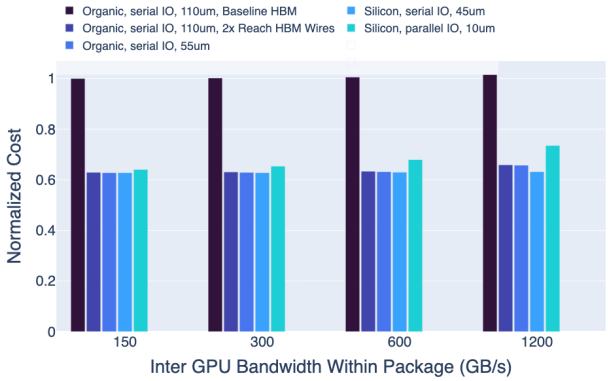

Fig. 9. Cost of the Example System for Different Bonding Pitches and IO Types. Costs normalized to 150GB/s inter GPU within package bandwidth, organic substrate, serial IO protocol, and 110μm bump pitch with baseline HBM configuration.

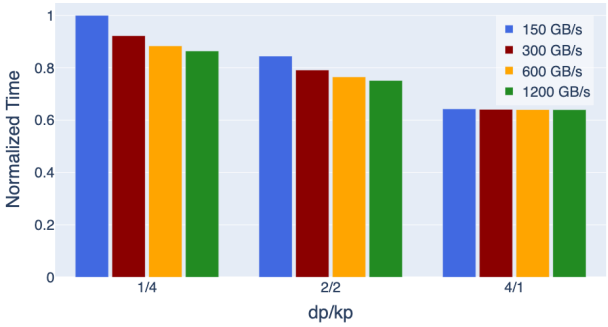

Fig. 10. Effect of Bandwidth and Parallelization Strategy on Training Time. 4 intra-node bandwidths studied: 150GB/s, 300GB/s, 600GB/s, and 1200GB/s; three parallelism strategies for data parallelism (dp) and kernel parallelism (kp) studied: 1/4, 2/2, and 4/1; time values normalized according to the slowest case, which is dp/kp=1/4 and intra bandwidth=150 GB/s.

require row/column masking for fine-grain operations, which opens up avenues for optimizing power consumption, memory utilization, throughput and sharing peripheral resources.

Presently, most IMC systems are programmed using low-level device libraries. The responsibility for deciding which kernels to map, where to insert synchronization barriers, and which domain/device-specific optimizations to apply ultimately lies with the programmer. This hinders the widespread adoption of these innovative architectures, even for domain/device experts, as manually rewriting large applications and making intelligent mapping/optimization decisions are non-trivial and error-prone. The diversity of IMC devices and architectures poses challenges for application portability and programming in heterogeneous environments. Overcoming these obstacles requires developing novel programming frameworks that take high-level (device-agnostic) program descriptions and architectural specifications as input, and automatically generate efficient code.

In isolated efforts, compiler stacks have been proposed to automate the mapping of appropriate compute primitives to memory devices, load balancing, and implementing technology-specific optimizations. However, these efforts predominantly focus on homogeneous architectures and are domain-specific – e.g., graph processing [44], programmable logic [45] and matrix multiplication [46] on memristive crossbars and tree-based models on CAMs [47], [48]. Recently, we have worked to develop CINM – a unified compilation framework that targets reusability and extensibility across the compiler stack for heterogeneous (non-)IMC architectures [49]. CINM is based on the multi-level intermediate representation (MLIR) framework that enables representing and transforming intermediate representations (IR) at various abstraction levels (referred to as dialects), catering to diverse application domains and heterogeneous hardware targets. Through progressive lowering of abstractions, one can reason about low-level compute primitives, their memory access patterns and optimizations for various metrics at various abstraction levels.

As input, CINM [49] takes applications represented in high-level (domain-specific) languages such as (a subset of) `C`, `PyTorch`, `linalg` and `Tensor Comprehensions`. In addition to IMC architectures, CINM also introduces abstractions for compute-near-memory (CNM) architectures. More

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

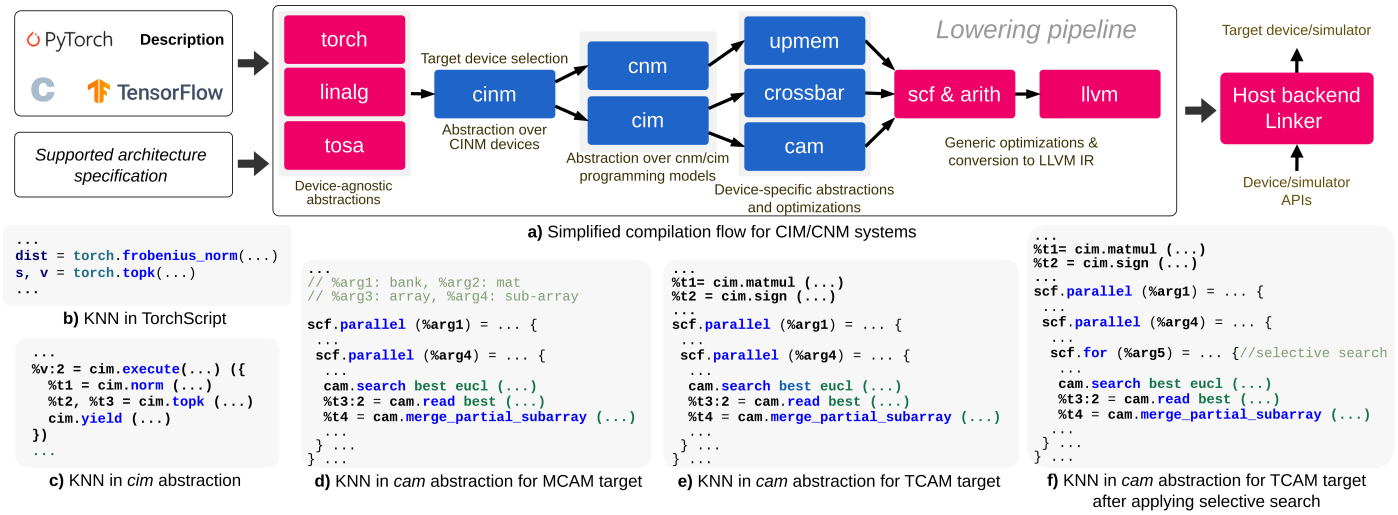

Fig. 11. A high-level overview of a compilation flow for CINM systems.

specifically, it introduces `cnm` and `cim` dialects to abstract over functions and types that are common to CNM and IMC devices, which are subsequently lowered to hardware targets in their respective device dialects, as shown in the lowering pipeline of Fig. 11a. As an example, C4CAM [50] introduces a CAM-specific dialect (`cam` in the figure) which consumes the `cim` abstraction and reuses most of the CINM pipeline. C4CAM automatically maps applications like HDC, k-NN and DNA read mapping by identifying code patterns of similarity measures (e.g., dot-product, cosine, Euclidean and Hamming distances) and lowers these operations to CAM searches.

The high-level compilation flows above facilitate programming IMC systems and also optimize them. In addition to standard parallelization and locality optimizations, CINM allows rewriting non-IMC-friendly compute primitives into IMC-amenable primitives (when possible), minimizing the number of writes to NVM cells (improving lifetime), reducing power consumption, and improving resource utilization of IMC arrays. To illustrate, consider the mapping of a k-NN kernel using a Euclidean norm into a CAM-based accelerator (Fig. 11b-f). The compilation stack takes the TorchScript code (Fig. 11b) and lowers it to a high-level domain-specific abstraction (Fig. 11c). The analysis passes at this abstraction determines that `cim.norm` and `cim.topk` represent a similarity measure and can be rewritten as CAM primitives as shown in Fig. 11d-e. Depending on the target CAM type (i.e., multi-bit or ternary), queries and stored data must undergo a bitwidth and/or dimensionality reduction to align with the array size and cell precision. The automated flow recognizes the need to incorporate additional operations, such as locality-sensitive hashing (LSH) or quantization, to ensure compatibility with TCAMs or MCAMs, respectively (Fig. 11d-e). If the target CAM device supports selective search [51], the compiler can generate code that selectively searches the restricted sets of rows/columns, thus optimizing for performance, energy consumption, or device utilization (Fig. 11f). Synchronization primitives are used to restrict the maximum number of sub-arrays activated concurrently, hence limiting the power consumption at the cost of increased latency. In addition to the abovementioned optimizations, CINM employs loop interchange to minimize write operations, leading to improved device lifetimes for memristive crossbars.

We believe that frameworks like CINM are badly needed to tackle the programmability challenges of IMC architectures. Today, CINM still relies on the programmer to decide whether to offload program regions to a given IMC accelerator. Automating these decisions is non-trivial as it requires abstract cost models and, moreover, optimizing for individual application regions might not necessarily lead to an optimal implementation of the entire application. CINM is being extended with sound support for custom number representations, using the `base2` dialect [52]. This will make it possible to safely target different accelerators while exploring quantized, non-standard number encodings and automatic bit-slicing (e.g. for emerging associative processors [53]). As for cost modeling, several simulation tools are available for estimating parameters such as memory access time, area, leakage, and dynamic power [54]. However, integrating these tools with CINM is currently challenging. The integration not only extends evaluation time due to expensive simulation processes but also faces limitations in capturing device, architecture, and communication details, such as stochasticity and non-idealities of memory devices, as well as system-level interactions and communication.

In addition to automatically generating optimized code for performance, power, and device utilization, existing compiler frameworks like CINM can be extended to support DSEs, allowing variation in architectural parameters without changing the code. However, identifying the optimal mapping strategy for heterogeneous systems, while considering diverse optimization targets, remains a subject for future research. Since heterogeneity in future architectures is expected to increase, there is a pressing need to develop cost models that account for device-agnostic and device-specific optimizations and drive the mapping of more complex applications without relying on detailed and slower simulations.

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.

REFERENCES

[1] K. Ishimaru, "Future of non-volatile memory -from storage to computing-," in *IEEE Int. Electron Devices Meeting (IEDM)*, 2019, pp. 1.3.1–1.3.6.

[2] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[3] S. Han *et al.*, "EIE: Efficient Inference Engine on Compressed Deep Neural Network," in *ISCA*, ser. ISCA '16, Piscataway, NJ, USA, 2016, pp. 243–254. [Online]. Available: https://doi.org/10.1109/ISCA.2016.30

[4] A. Sebastian *et al.*, "Temporal correlation detection using computational phase-change memory," *Nature Comms.*, vol. 8, no. 1, p. 1115, 2017.

[5] Jain, S and Ranjan, A and Roy, K and Raghunathan, A, "Computing in Memory With Spin-Transfer Torque Magnetic RAM," *IEEE Transactions on VLSI*, vol. 26, no. 3, pp. 470–483, Mar. 2018.

[6] H.-S. P. Wong *et al.*, "Metal–oxide RRAM," *Proceedings of the IEEE*, vol. 100, no. 6, pp. 1951–1970, 2012.

[7] G. W. Burr *et al.*, "Neuromorphic computing using non-volatile memory," *Advances in Physics: X*, vol. 2, no. 1, pp. 89–124, 2017.

[8] M. Jerry *et al.*, "A ferroelectric field effect transistor based synaptic weight cell," *J. of Phys. D: App. Phys.*, vol. 51, no. 43, p. 434001, 2018.

[9] X. S. Hu *et al.*, "In-memory computing with associative memories: A cross-layer perspective," in *2021 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2021, pp. 25–2.

[10] A. F. Laguna *et al.*, "Hardware-software co-design of an in-memory transformer network accelerator," *Frontiers in Electronics*, vol. 3, p. 847069, 2022.

[11] S. Liu *et al.*, "16.2 A 28nm 53.8TOPS/W 8b Sparse Transformer Accelerator with In-Memory Butterfly Zero Skipper for Unstructured-Pruned NN and CIM-Based Local-Attention-Reusable Engine," in *2023 IEEE International Solid- State Circuits Conference*, 2023, pp. 250–252.

[12] S. Sridharan *et al.*, "X-former: In-memory acceleration of transformers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 08, pp. 1223–1233, aug 2023.

[13] M. Li *et al.*, "Imars: An in-memory-computing architecture for recommendation systems," in *DAC*, 2022, pp. 463–468.

[14] H. Geng *et al.*, "Privacy preserving in-memory computing engine," arXiv:2308.02648, 2023.

[15] W. Li *et al.*, "H3datten: Heterogeneous 3-d integrated hybrid analog and digital compute-in-memory accelerator for vision transformer self-attention," *IEEE T. on Very Large Scale Integration (VLSI) Systems*, 2023.

[16] A. Santoro *et al.*, "Meta-learning with memory-augmented neural networks," in *ICML*, 2016, pp. 1842–1850.

[17] S. Dünkel *et al.*, "A FeFET based super-low-power ultra-fast embedded NVM technology for 22nm FDSOI and beyond," in *2017 IEEE International Electron Devices Meeting (IEDM)*, Dec. 2017, pp. 19.7.1–19.7.4.

[18] A. Kazemi *et al.*, "Achieving software-equivalent accuracy for hyperdimensional computing with ferroelectric-based in-memory computing," *Scientific Reports*, vol. 12, no. 1, p. 19201, 2022.

[19] R. Mao *et al.*, "Experimentally validated memristive memory augmented neural network with efficient hashing and similarity search," *Nature communications*, vol. 13, no. 1, p. 6284, 2022.

[20] A. Kazemi *et al.*, "A Hybrid FeMFET-CMOS Analog Synapse Circuit for Neural Network Training and Inference," in *ISCAS*, 2020, pp. 1–5.

[21] X. Yin *et al.*, "FeCAM: A Universal Compact Digital and Analog Content Addressable Memory Using Ferroelectric," *IEEE TED*, vol. 67, no. 7, pp. 2785–2792, 2020.

[22] A. I. Khan *et al.*, "The future of ferroelectric field-effect transistor technology," *Nature Electronics*, vol. 3, no. 10, pp. 588–597, 2020.

[23] S. Dutta *et al.*, "Logic compatible high-performance ferroelectric transistor memory," *IEEE EDL*, vol. 43, no. 3, pp. 382–385, 2022.

[24] S. Dutta *et al.*, "Lifelong Learning with Monolithic 3D Ferroelectric Ternary Content-Addressable Memory," in *IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2021, pp. 17–1.

[25] K. Ni *et al.*, "Ferroelectric ternary content-addressable memory for one-shot learning," *Nature Electronics*, vol. 2, no. 11, pp. 521–529, 2019.

[26] Q. Kuang and L. Zhao, "A practical gpu based knn algorithm," in *Proceedings. The 2009 International Symposium on Computer Science and Computational Technology (ISCSCI 2009)*. Citeseer, 2009, p. 151.

[27] S. Han *et al.*, "EIE: efficient inference engine on compressed deep neural network," in *ISCA*, 2016, pp. 243–254.

[28] S. Pal *et al.*, "Designing a 2048-Chiplet, 14336-Core Waferscale Processor," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, Dec. 2021, pp. 1183–1188, iSSN: 0738-100X.

[29] R. Mahajan *et al.*, "Embedded Multi-die Interconnect Bridge (EMIB) – A High Density, High Bandwidth Packaging Interconnect," in *2016 IEEE 66th Electronic Components and Technology Conference (ECTC)*, May 2016, pp. 557–565.

[30] A. F. Laguna *et al.*, "Invited paper: Algorithm co-design for few-shot learning at the edge," in *International Conference on Computer Aided Design (ICCAD) (to appear)*, 2023.

[31] K. Ni *et al.*, "A circuit compatible accurate compact model for ferroelectric-FETs," in *2018 IEEE Symposium on VLSI Technology*. IEEE, 2018, pp. 131–132.

[32] M. M. Sharifi *et al.*, "Application-driven design exploration for dense ferroelectric embedded non-volatile memories," in *International Symposium on Low Power Electronics and Design (ISLPED)*, 2021, pp. 1–6.

[33] H. Mulaosmanovic *et al.*, "Ferroelectric transistors with asymmetric double gate for memory window exceeding 12 V and disturb-free read," *Nanoscale*, vol. 13, no. 38, pp. 16 258–16 266, 2021.

[34] E. D. Grimley *et al.*, "Atomic structure of domain and interphase boundaries in ferroelectric HfO2," *Advanced Materials Interfaces*, vol. 5, no. 5, p. 1701258, 2018.

[35] G. Karunaratne *et al.*, "Robust high-dimensional memory-augmented neural networks," *Nature communications*, vol. 12, no. 1, p. 2468, 2021.

[36] A. Graening *et al.*, "Chiplets: How small is too small?" in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[37] N. Ardalani *et al.*, "Deepflow: A cross-stack pathfinding framework for distributed ai systems," *ACM Trans. Des. Autom. Electron. Syst.*, dec 2023. [Online]. Available: https://doi.org/10.1145/3635867

[38] Z. Jia *et al.*, "Beyond data and model parallelism for deep neural networks." *Proceedings of Machine Learning and Systems*, vol. 1, pp. 1–13, 2019.

[39] NVIDIA, "Nvidia Tesla V100 GPU Architecture, The World's Most Advanced Data Center GPU," 2017.

[40] D. Kehlet, "Accelerating Innovation Through a Standard Chiplet Interface," *Intel White Paper*.

[41] S. Pal *et al.*, "I/O Architecture, Substrate Design, and Bonding Process for a Heterogeneous Dielet-Assembly based Waferscale Processor," in *Electronic Components and Technology Conference*, 2021, pp. 298–303.

[42] D. Das Sharma *et al.*, "Universal Chiplet Interconnect Express (UCIe): An Open Industry Standard for Innovations With Chiplets at Package Level," *IEEE Transactions on Components, Packaging and Manufacturing Technology*, vol. 12, no. 9, pp. 1423–1431, Sep. 2022.

[43] S. Jangam *et al.*, "Latency, Bandwidth and Power Benefits of the SuperCHIPS Integration Scheme," in *2017 IEEE 67th Electronic Components and Technology Conference (ECTC)*. IEEE, May 2017, pp. 86–94.

[44] Y. Huang *et al.*, "A heterogeneous PIM hardware-software co-design for energy-efficient graph processing," in *2020 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2020, pp. 684–695.

[45] S. Frerix *et al.*, "Comprime: A compiler for parallel and scalable reram-based in-memory computing," in *2019 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, 2019, pp. 1–6.

[46] A. Siemieniuk *et al.*, "OCC: An automated end-to-end machine learning optimizing compiler for computing-in-memory," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems (TCAD)*, vol. 41, no. 6, pp. 1674–1686, Aug. 2021.

[47] G. Pedretti *et al.*, "X-TIME: An in-memory engine for accelerating machine learning on tabular data with CAMs," *arXiv:2304.01285*, 2023.

[48] M. Rakka *et al.*, "Dt2cam: A decision tree to content addressable memory framework," *IEEE Transactions on Emerging Topics in Computing*, 2023.

[49] A. A. Khan *et al.*, "CINM (Cinnamon): A Compilation Infrastructure for Heterogeneous Compute In-Memory and Compute Near-Memory Paradigms," 2023. [Online]. Available: https://arxiv.org/abs/2301.07486

[50] H. Farzaneh *et al.*, "C4cam: A compiler for cam-based in-memory accelerators," 2023. [Online]. Available: https://arxiv.org/abs/2309.06418

[51] C. A. Zukowski and S.-Y. Wang, "Use of selective precharge for low-power content-addressable memories," in *1997 IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, vol. 3. IEEE, 1997, pp. 1788–1791.

[52] K. F. A. Friebel *et al.*, "BASE2: An IR for binary numeral types," in *13th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies (HEART 2023)*, ser. HEART2023, 2023, pp. 19–26. [Online]. Available: https://doi.org/10.1145/3597031.3597048

[53] J. P. C. de Lima *et al.*, "Full-stack optimization for cam-only dnn inference," in *Proceedings of the 2024 Design, Automation and Test in Europe Conference (DATE)*, ser. DATE'24. IEEE, Mar. 2024, pp. 1–6.

[54] M. Niemier *et al.*, "Cross layer design for the predictive assessment of technology-enabled architectures," in *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2023, pp. 1–10.

Authorized licensed use limited to: UCLA Library. Downloaded on September 14,2024 at 00:06:32 UTC from IEEE Xplore. Restrictions apply.