

Increasing the Efficiency of Associative Processor Architectures via CMOS-Compatible Hybridization

Socrates S. Wong,^{*†} Cecilio C. Tamarit,^{*†} Mohammad M. Sharifi,[‡] Zephan Enciso,[‡]

Dayane Reis,[§] Michael Niemier,[‡] X. Sharon Hu,[‡] and José F. Martínez[†]

[†]Cornell University, Ithaca, NY 14850 USA {ssw96,ct652,martinez}@cornell.edu

[‡]University of Notre Dame, Notre Dame, IN 46556 USA {msharif1,zenciso,mnkiemier,shu}@nd.edu

[§]University of South Florida, Tampa, FL 33620 USA dayane3@usf.edu

Abstract—We present a hybrid, general-purpose, associative processing-in-memory architecture that combines the energy and area advantages of a primary FeFET-based CAM array with the write performance and endurance of a much smaller CMOS-based sidekick. The hybrid nature of the architecture is transparent to the programmer, who uses a RISC-V ISA with standard RVV vector extensions. Detailed SPICE- and system-level simulations show our hybrid design dramatically curbs the endurance disadvantages of a pure FeFET design and delivers, on average, 30% and 11% area and energy savings over a purely CMOS implementation, respectively, at a performance loss of barely 1% over pure CMOS.

I. INTRODUCTION

Processing-using-memory (PUM) architectures perform computation directly in a memory array. Although associative memories have long been used in this way to accelerate specific kernels (e.g., pattern matching, machine learning) [12], [13], [16], [20], the associative processor paradigm (AP) [10] has recently regained attention as a general-purpose approach to PUM that can deliver order-of-magnitude speedups [3], [4], [30]–[32], [35]. For example, CAPE [4] reports up to 254× over an area-equivalent out-of-order CPU core. These APs can perform arithmetic and logic operations simultaneously on many data items via sequences of bulk CAM search and update operations, without any ALU circuitry.

At the same time, research on emerging memory technologies (EMTs), such as the ferroelectric field-effect transistor (FeFET), has shown that these devices can offer improvements in storage density and energy efficiency compared to CMOS, especially for PUM designs [11], [31], [35]. However, EMTs can also present significant challenges, including high write latency, high write energy, and reduced endurance. High write latency and wearout, in particular, can be lethal to AP performance and endurance, as most AP arithmetic operations involve numerous bulk updates.

We present a novel hybrid CMOS+FeFET AP solution that leverages the area and energy advantages of FeFET-based storage while retaining the performance and endurance benefits of a CMOS implementation. In this architecture, only a small

fraction of the total AP storage is CMOS-based, which the microarchitecture engages selectively to take advantage of CMOS’ faster and more resilient writes; the remaining storage is FeFET-based, yielding significant improvements over both full-CMOS and full-EMT implementations. Critically, the programmer remains oblivious to the hybrid nature of the microarchitecture, as all required microarchitecture and microcode changes are transparent to the RISC-V vector ISA.

We choose FeFET-based CAM cells because they offer comparable read times to CMOS-based ones, low read energy, and high density (Table I). They also allow for compatible read voltages between both FeFET and CMOS cells (details in Section III). Their physical compatibility reduces peripheral circuit requirements, resulting in lower overall area and power consumption than other EMTs. Nevertheless, the principles behind our approach could be applied to other EMTs to harness their own unique characteristics.

We evaluate our proposed mechanism using detailed SPICE- and architecture-level simulations. The results show that our hybrid AP design dramatically mitigates the FeFET design’s endurance disadvantages, reducing area and energy by 30% and 11% over a purely CMOS implementation, respectively, at a performance loss of only 1% over pure CMOS.

II. BACKGROUND

An associative processor (AP) [10] can be viewed as a vector processor in which each vector comprises tens or hundreds of thousands of elements. The underlying storage is a content-addressable memory. APs implement vector arithmetic and logic operations via sequences of bulk searches and updates, directly on the stored values, without ALU circuitry. These

TABLE I
CHARACTERISTICS OF CONTENT-ADDRESSABLE
MEMORY (CAM) CELLS* USED IN THIS WORK

Figure-of-Merit	CMOS [5], [34]	FeFET [7], [15]
Read Time	<1 ns	<1 ns
Write Time	<1 ns	<20 ns
Read Energy	Low	Low
Write Energy	Low	Medium
Leakage Power	High	Low
Write Endurance	$> 10^{16}$	$10^6 - 10^{12}$
CAM Cell Size	$> 100F^2$	$\sim 20F^2$

*Energy and access time differences between CMOS and FeFET technologies are larger at the array level, as the parasitic component of CAM arrays tends to be smaller in dense arrays.

*Socrates Wong and Cecilio Tamarit are co-lead authors.

This work was supported in part by the Semiconductor Research Corporation (SRC) through the ACE and SUPREME research centers, both part of the JUMP 2.0 program co-sponsored by DARPA, by the CRISP and ASCENT research centers, both part of the JUMP 1.0 program also co-sponsored by DARPA, and by NSF award DBI-2019674.

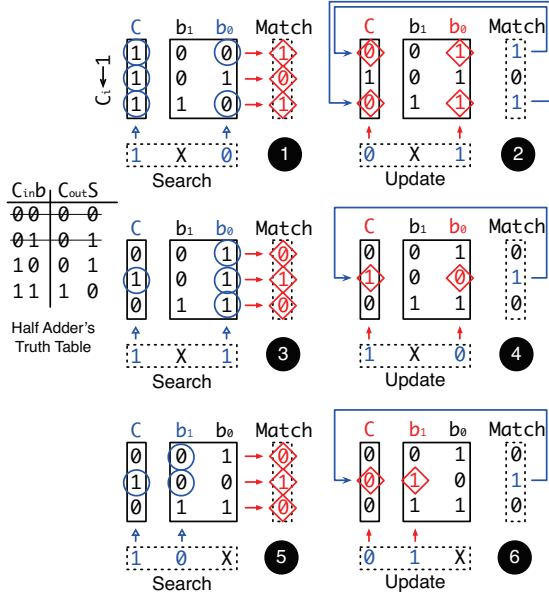


Fig. 1. Associative increment instruction (+1) as a bit-serial sequence of search-update operations to a vector of three two-bit elements b_1b_0 . Carry-bit column C is initialized to 1. Blue circles and red diamonds indicate bit matches and updates, respectively. The fourth search-update pair, not shown, would result in no updates, since there would be no $Cb_1 = 11$ matches.

sequences amount essentially to microcode, hidden from the programmer, who can interact with the AP via a vector ISA.

Figure 1 illustrates an AP increment operation over multiple vector elements in parallel. The microcode walks the truth table for a half adder, one bit of the operand (i.e., bit-serially) and one row of the table at a time. For each bit position i , the AP: ❶ Searches for vector elements where the i^{th} bit is 0 and the running carry (an additional bit of storage per element held by the AP) for that element is 1 ($CB_i = 10$) and “tags” them (another additional bit of AP storage stores the matches); ❷ Updates the i^{th} bit of the tagged elements to 1 and their running carry to 0; ❸ Searches for vector elements whose i^{th} bit is 1 and the running carry for that element is also 1; ❹ Updates the i^{th} bit of the matching elements to 0 and the running carry to 1. ❺ and ❻ show the next iteration, for bit $i + 1$. (In the example, the hardware skips the two cases in the truth table where inputs and outputs are the same, since they would require no updates.) For a relatively simple increment instruction on a 32-bit value, this would take over one hundred search-update operations [4]. The key to AP performance, however, is that operations work simultaneously on an extremely large number of vector elements.

A. Content-Addressable Processing Engine (CAPE)

A Content-Addressable Processing Engine (CAPE) [3], [4], [30] is an AP architecture composed of four main blocks. The Control Processor (CP) is a tiny in-order core that can execute RISC-V code with RVV standard vector extensions [29]. It offloads vector instructions to the Compute-Storage Block (CSB), which is CAPE’s associative processing engine. It is essentially a binary content-addressable memory (CAM) organized into very large vectors (tens of thousands of elements), made up of CMOS 6T push-rule CAM bitcells [14] that support the

four microoperations used in CAPE’s computational model: single-element reads and writes, as well as highly-efficient multi-element (vector) searches and updates. The RISC-V vector register names in each instruction are used to index the appropriate vector operands within the CSB; the program *never* uses addresses to access the CSB directly.

Load/store vector instructions go through a Vector Memory Unit (VMU), which interfaces directly with DRAM; other vector instructions go through a Vector Control Unit (VCU). The VMU and VCU generate the appropriate control/data signals to the CSB—e.g., the sequence of search and update pairs in the vector increment example above. Once a vector instruction is dispatched, its ROB entry remains in the CP until the vector operation completes. Meanwhile, subsequent scalar instructions may issue and execute (if not data-dependent with the vector instruction) but not commit.

Internally, the original CAPE architecture organizes the CSB into subarrays of 32×32 bitcells (35×32 when accounting for additional columns that hold metadata for computation), along with some peripheral logic. This geometry allows CAPE to be clocked fast. Vector elements are bit-sliced across CSB subarrays of the same column: subarray i stores the i^{th} bit of the 32 vector elements corresponding to that column, across all 32 RISC-V vector names, which facilitates bit-parallel operations. Subarrays in the same column are chained to support propagation (e.g. carry) in bit-serial operations. The original CAPE paper describes this in more detail [4], reporting an average speedup of $14 \times$ (up to $256 \times$) over an area-equivalent O3CPU core.

B. Emerging Memory Technologies (EMTs)

EMTs enable the development of dense, power-efficient solutions. Among them, FeFET devices are attractive due to their low operating voltage, fast switching speed, large memory window, and CMOS compatibility [1], [6], [22], [22], [25]. These devices consist of a ferroelectric layer integrated atop the gate stack of a MOSFET, where the polarization of the Fe layer determines the device’s threshold voltage (V_{th}). The V_{th} of FeFETs can be precisely controlled by applying voltage pulses to the gate. However, changing the polarization state requires comparatively high voltages, so additional thick-oxide switches are needed to decouple and protect downstream CMOS devices from oxide breakdown. Traditionally, FeFETs are fabricated on the front-end-of-line (FEOL) using Zirconium-doped HfO_2 (HZO) as the ferroelectric layer. While these FeFETs have demonstrated large memory windows and non-volatility, silicon-based FeFETs face persistent challenges, such as charge trapping, which leads to delays after write operations and hinders scaling down the channel length [22]. These challenges are mitigated when back-end-of-line (BEOL) materials, such as Indium Tungsten Oxide (IWO), are used to construct the channel of the FeFET devices. BEOL FeFETs also exhibit lower write voltages compared to their FEOL counterparts, which reduces the amount of energy required to write the FeFET device.

In the context of PUM, and in particular APs, EMTs could

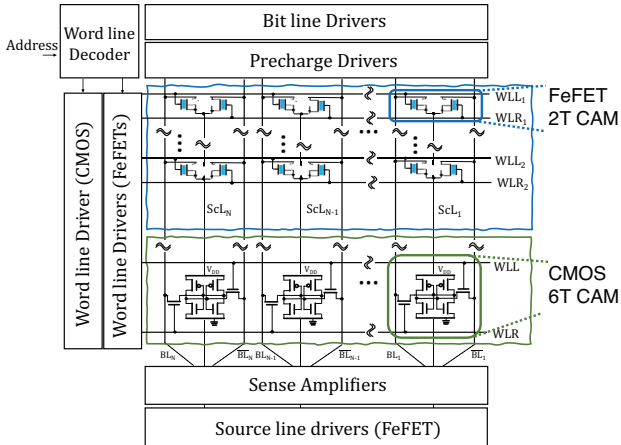


Fig. 2. Diagram representing one of the CSB’s thousands of CMOS+FeFET hybrid subarrays, not drawn to scale. Over 90% of a subarray’s area is 2T FeFET-based [33] (blue); the CMOS side [14] is comparably small (green).

potentially provide significant area and energy savings. However, as we show in our evaluation (Section IV), extended write times and lower endurance make EMT-only designs impractical. Thus, the limitations of EMT-only implementations of AP architectures are likely to outweigh their benefits in area and energy efficiency, motivating a hybrid solution. Encouragingly, recent device-level work has shown both the feasibility of and opportunity for co-integrating EMTs and CMOS. A variety of studies emphasize that FeFET devices are readily compatible with CMOS [2], [6], [25], and recent works have explored the potential of compute-in-memory solutions based on hybrid technologies, but for specific applications [8], [9].

III. A HYBRID ASSOCIATIVE PROCESSOR WITH TECHNOLOGY-AWARE MICROCODE

The key principle behind our proposed mechanisms is to leverage the faster CMOS latency and higher endurance in a small set of CMOS-based intermediate vector registers by judiciously mapping operands to them and, at the same time, allowing the architecture to take advantage of the benefits of the EMT, which is what most of the AP storage array is constructed with. Importantly, this mechanism is hidden from the programmer at all times.

A. Design and Operation of the Hybrid AP Subarray

Figure 2 depicts the subarray-level view of our hybrid design. We divide all subarrays into two sections: one uses 2T FeFET CAM cells [33], while the other is a CMOS-based 6T push-rule cell [14]. The vast majority of the area is devoted to smaller, power-thrifty FeFET cells (blue in the figure), with only a handful of rows implemented in CMOS (green). The two blocks lie adjacent and share column bitlines, precharge drivers, and sense amplifiers, yet keep separate wordline drivers: the CMOS driver supplies standard V_{DD} pulses, while the FeFET driver delivers the higher write voltage V_W . A common decoder triggers whichever driver is needed, so only targeted rows see the required voltage level.

Table II summarizes all operating modes. While CMOS rows perform ordinary complementary-bitline writes, FeFET

TABLE II
SUMMARY OF SEARCH AND UPDATE OPERATIONS IN THE HYBRID-TECHNOLOGY SUBARRAY OF FIGURE 2(B)

Technology	Operation	WLL	WLR	BL	\overline{BL}	SCL	DNW
CMOS	Write ‘0’	V_{DD}	V_{DD}	V_{SS}	V_{DD}	NA	NA
	Write ‘1’	V_{DD}	V_{DD}	V_{DD}	V_{SS}	NA	NA
	Search	D	D	V_{PRE}	V_{PRE}	NA	NA
FeFET	Macro-erase	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_W
	Write ‘1’	V_W	V_{SS}	V_{SS}	V_{SS}	V_{SS}	V_{SS}
	Inhibit write	$V_{L,WL}$	$V_{L,WL}$	$V_{L,BL}$	$V_{L,BL}$	V_{SS}	V_{SS}
	Search	D	\overline{D}	V_{PRE}	V_{PRE}	V_{SS}	V_{SS}

rows remain untouched at 0V. On the other hand, FeFET programming starts with a macro-erase, then applies a V_W pulse on WLL with grounded BL and \overline{BL} as neighboring cells receive inhibit voltages. The CMOS driver holds its wordlines low throughout FeFET writes, isolating the CMOS portion for read operations, because both technologies read at the same precharge level V_{PRE} , the entire hybrid array can be searched simultaneously via the shared bitline/sense path.

B. Increasing Performance with CMOS-side Computation

Operating a hybrid subarray like the one we just described introduces a unique set of challenges. Namely, technology-aware microcode is required to determine when, how often, and for what purpose we should use either the EMT or the CMOS side. In this subsection, we describe the *CMOS-side Computation* mechanism as a series of design points (and their corresponding utilization strategies) that make these decisions transparently. Each design point builds on the previous one, gradually merging into what we consider our capstone solution: Adaptive CMOS-side Computation (ACC).

1) *SCC – Single-Instruction CMOS-side Computation*: Our first design point, Single-instruction CMOS-side Computation (SCC), exposes 32 architectural vector registers to the programmer, but achieves higher performance by temporarily mapping all *intermediate* updates and subsequent searches to a small subset of CMOS physical vector registers and limiting the number of updates performed on the EMT physical vector register to one per instruction: the final writeback to the destination register. By doing so, we aim to approximate CMOS speed and endurance at minimal cost while still benefiting from EMT-like area and energy savings.

SCC incurs little area or control overhead. Recall that the vector registers of baseline CAPE are implemented as chains of 35x32 bitcell subarrays. This implies that 32 physical vector registers are exposed and map one-to-one to the architectural ones, while the three remaining ones are only used internally to store intermediate values: metadata that is a byproduct of the computation. SCC transparently replaces these columns that were already being used to hold internal state of the AP with a CMOS counterpart. Altogether, for every subarray, 32x32 bitcells would be FeFET-based and map one-to-one to the architectural vector names, while 3x32 would be CMOS-based. A single additional bit in the wordline decoder is sufficient to be able to specify the source or destination register in any case, whether it be an EMT architectural register or one of the CMOS registers holding an intermediate result.

Algorithm 1 shows an example of AP microcode, demonstrating hardcoded SCC usage when performing an XOR vector

Algorithm 1 Associative Vector XOR with Single-Instruction CMOS-side Computation (SCC).

$v_{xor.vv} v4, v1, v2$	▷ Inputs: $v1, v2$; Outputs: $v4$
1: Update $CMOS_i \leftarrow 1$	▷ Initialize CMOS-side register i
2: Search $0, 0$	▷ Sets $tags_i$ if $v1_i, v2_i == 0, 0$
3: Search-Accum. $1, 1$	▷ Sets $tags_i$ if $v1_i, v2_i == 1, 1$
4: Cond. Update $CMOS_i \leftarrow 0$	▷ Writes 0 if $tags_i$ is set
5: Copy $v4_i \leftarrow CMOS_i$	▷ Write back to EMT-side $v4$

instruction. The operations that happen in CMOS are highlighted green. In line 1, a CMOS register becomes the temporary placeholder for destination register $v4$, with all entries initialized to 1. (Note that XOR and other logic operations are bit-parallel in CAPE. In bit-serial arithmetic operations, EMT-to-CMOS handovers happen in parallel across all bits of the destination vector register, but arithmetic still takes place bit-serially.) In line 4, entries whose $(v1, v2)$ inputs are $(0, 0)$ or $(1, 1)$ are selectively set to 0, thus completing the XOR operation. Note that source operands $v1$ and $v2$ remain on the EMT side and continue to participate in searches, since FeFET reads are fast and do not contribute to wearout. At the end of the instruction (line 5), the result for $v4$ is written back to the EMT side, and the CMOS register becomes available. Indeed, under SCC, each instruction completely reverts to the EMT array upon completion of execution.

2) *MCC – Multi-Instruction CMOS-side Computation*: Our second design point extends SCC by also optimizing *across* instructions. The key insight behind MCC is that, by eliding EMT writebacks and leveraging CMOS vector registers to reuse operands across instructions, it is possible to approximate CMOS performance even further in the presence of data-dependent instruction sequences. In addition to the three physical vector registers that enable SCC functionality, the system employs n additional MCC-type CMOS registers that are used to support data dependencies across instructions. Because the use of CMOS registers extends beyond the per-instruction microcode, MCC also requires a small mapping table that tracks which architectural vector register each MCC register holds. When referencing a vector name, the microarchitecture checks the mapping table. If there is a match, it uses the corresponding MCC entry; if not, it defaults to the EMT one.

Looking back at Algorithm 1, in MCC there would be no writeback to EMT storage at the end of the instruction (line 5). Instead, the CMOS-side register would be labeled “ $v4$ ” in the register mapping, and a subsequent instruction sourcing $v4$ would read off this register instead of accessing its EMT counterpart. If an instruction needs an MCC register but they are all taken up, one is selected as a victim and written back to the EMT side. We evaluated various MCC replacement policies and found that their effects were negligible, so our design uses FIFO for simplicity. In summary, for MCC, the microcode actively moves vectors from the EMT to the CMOS side as needed, and these eventually trickle back to the EMT side per the replacement policy.

C. Mitigating the Drawbacks of CMOS

SCC and MCC improve performance by relying heavily on the CMOS side. As we discuss in our evaluation, this

TABLE III
EXPERIMENTAL SETUP FOR CELL-LEVEL EVALUATION

Parameter	CMOS-based Model	FeFET-based Model
Technology Node	20 nm @ 1 GHz (scaled from 7 nm [27])	22 nm @ 1 GHz (circuit-level simulations)
Simulation Libraries	ASAP 7 nm PDK [28]	Preisach, multi-domain [21] + CMOS PTM 22 nm [17]
Peripherals	Mem (SA, decoders, etc) + command distribution	Mem (SA, decoders, etc) + command distribution
Write Voltage/Time	0.8 V @ <1 ns [4]	± 2.8 V @ 30 ns [7], [15]
Read Voltage/Time	0.8V @ <1 ns [4]	0.8 V @ <1 ns [24], [26]

TABLE IV
EXPERIMENTAL SETUP FOR SYSTEM-LEVEL EVALUATION

	Baseline O3CPU (Skylake)	AP Control CPU
System configuration	out-of-order core, 3.6 GHz 32 kB/32 kB/1 MB L1D/L1/L2 5.5 MB L3 (shared), 512 B LL cache line	in-order core, 2.7 GHz 32 kB/32 kB/1 MB L1D/L1/L2 512 B L2 cache line
Core configuration	8-issue, 224 ROB, 72 LQ, 56 SQ 4/4/4/3/1 IntAdd/IntMul/FP/Mem/Br units Tournament BP, 4096 BTB, 16 RAS	2-issue in-order, 5+5-entry LSQ 4/1/1/1 Int/FP/Mem/Br units Tournament BP, 4096 BTB, 16 RAS
L1 D/I cache	8-way, LRU, MESI, 2 tag/data latency	8-way, LRU, 2 tag/data latency
L2 cache	16-way, LRU, MESI, 14 tag/data latency	16-way, LRU, 14 tag/data latency
L3 cache	11-way, LRU, 50 tag/data latency, shared	N/A
Main memory	4H HBM, 8 channels, 16 GB/s, 512 MB per channel	

undermines some of the potential benefits of EMTs. The following optimization focuses on striking a balance.

1) *ACC – Adaptive CMOS-side Computation*: Our final design point leverages two key observations: (1) There is fundamentally no architectural or technological difference between SCC and MCC entries—only a functional one. (2) The responsibility for storing the architectural state can be shared between both CMOS and FeFET registers, enabling us to reduce the total number of physical registers, thus saving on silicon area.

In ACC, we no longer distinguish between SCC and MCC registers. If an instruction needs a CMOS register to store an intermediate result and none is available, it can force the eviction of one of the CMOS-mapped architectural registers to the EMT side. The total number of registers is still strictly greater than 32 to allow movement between the CMOS and EMT sides; however, the number of EMT registers may be fewer than 32, since some CMOS registers may be used to store architectural state. Therefore, the register mapping table must be extended to cover all physical registers regardless of their type. In general, the composition and microprogramming should be such that at least one CMOS register becomes free at the end of each instruction so that the microcode may carry out any swaps needed to process the next instruction.

IV. EVALUATION

A. Experimental Setup

We perform the layout of our CAM-based subarray, including peripherals, and extract parasitic capacitance for SPICE-level simulations of power and latency. Table III summarizes the evaluation parameters used in CMOS- and FeFET-based designs. They both employ 32×128 subarrays. Peripheral circuits, such as the command distribution network and row and column decoders, are implemented in CMOS for both designs. The CMOS-based implementation follows the previously published CAPE design [4], which was implemented with the ASAP

7 nm PDK library. The FeFET-based design employs 2-FeFET TCAM cells [33]. No FeFET model is available at the same 7 nm technology node, so we evaluate the FeFET-based design using the Preisach multi-domain model for FeFETs with the underlying CMOS 22 nm PTM [21]. To ensure a fair comparison between both, we scale CMOS to 20 nm using the equation-based scaling method proposed in [27]. Furthermore, we leverage the leakage power evaluation reported in [19] for CMOS and FeFET CAMs.

For system-level modeling (Table IV), we heavily modify the gem5 simulator [18] to reproduce the results of the original (CMOS) CAPE design [4]. The final latency and dynamic power statistics are derived from the CAM read/write latency and voltage data for the individual 6T CMOS cells. For our FeFET-only design and the hybrid designs, we adjust timing and energy parameters based on our FeFET and hybrid cell modeling results, as shown in Table III. The set of evaluated workloads comprises the entire Phoenix suite — a diverse set of shared-memory, data-intensive applications originally intended for multicore/ccNUMA machines [23] that, as with the baseline, we carefully retarget via hand-vectorization [4].

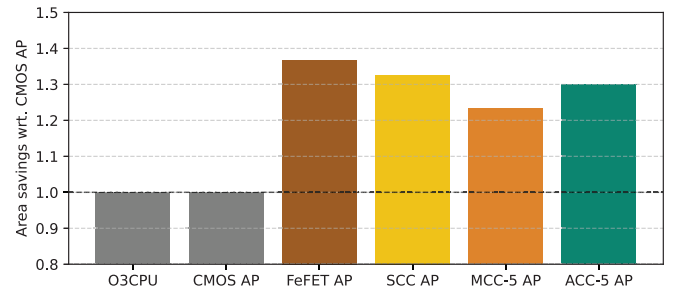
We simulate six designs: (1) A CMOS-based CAPE-like AP design; (2) **O3CPU**, an aggressive out-of-order scalar core roughly equivalent in area to CAPE; (3) a CAPE-like AP design with a **FeFET**-based CSB; and our three proposed hybrid-technology AP designs, (4) **SCC**, (5) **MCC**, and (6) **ACC**. All AP designs comprise 32k-element vectors.

B. Results

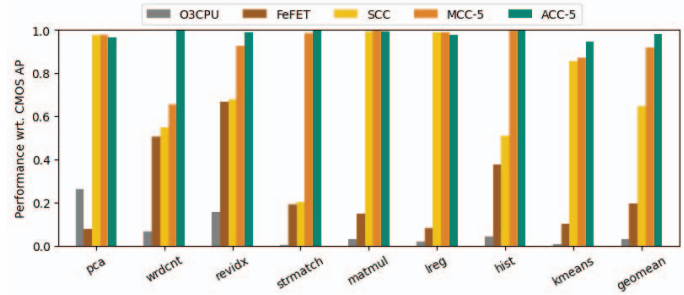
Using the CAPE baseline configuration, we conducted a basic AP execution profile of the Phoenix applications to quantify the number of search and bulk update operations. In general, all applications exhibited a roughly even search:update ratio. Recall that FeFET CAM cells perform similarly to CMOS in searches; however, the abundance of update operations across the board suggests that the FeFET-only AP design is likely to suffer from performance and endurance degradation. As we will see shortly, our results bear out this prediction.

Figure 3 shows area, performance, and energy results (higher is better) for the configurations under study, all normalized to CMOS-based CAPE. The O3CPU performance results largely track those of the original CAPE paper [4]; we include them here for reference purposes to confirm that AP designs are significantly advantageous in terms of performance on an area-equivalent basis.

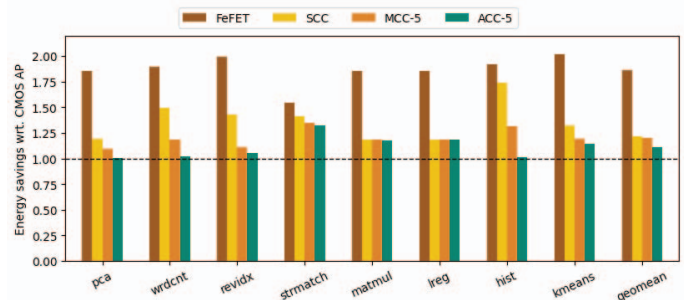
Performance-wise, all applications benefit from our simplest hybrid approach, SCC. Notably, it would be difficult to justify running *pca* or *lreg* on an AP implemented solely with FeFETs, even though it is almost $1.4\times$ smaller. In those cases, SCC suffices to completely bridge the performance gap between the FeFET and CMOS-based designs while retaining an area footprint similar to that of the FeFET design. However, some applications like *strmatch* or *revidx* still underperform, remaining at (or near) FeFET levels, suggesting that they fail to exploit intra-instruction optimizations. Indeed, a closer inspection reveals that the dominant instructions in these applications exhibit few intra-instruction updates, for example,



(a) Area savings (higher is better) normalized to CMOS AP.



(b) Performance (higher is better) normalized to CMOS AP.



(c) Energy savings (higher is better) normalized to CMOS AP.

Fig. 3. Area, performance, and energy results (higher is better) for the configurations under study, normalized to CMOS-based CAPE in all cases.

with many bit-parallel operations but relatively few arithmetic ones. The energy results highlight this more clearly: since SCC doesn't engage the more power-hungry CMOS side as often for these applications, energy always lies in between CMOS and FeFET, trending towards one or the other to different degrees.

The MCC design, where CMOS-side accesses are leveraged within and across instructions, dramatically improves this outlook, bringing the geometric mean of performance within 10% of the CMOS-based baseline, CAPE. We conducted a sensitivity study to examine the impact of increasing the number of MCC-specific CMOS registers (beyond the three SCC-dedicated ones); Figure 4 shows our results. The Y axis is normalized to the performance of the CMOS-only associative processor baseline. The general trend is that as the number of MCC vector registers increases, performance approaches that of ideal CMOS. For MCC-5 (five MCC vector registers), on average 95% of updates occur on the CMOS side. Any further increase in MCC size yields diminishing returns as silicon area also increases. Thus, we use MCC-5 as the representative

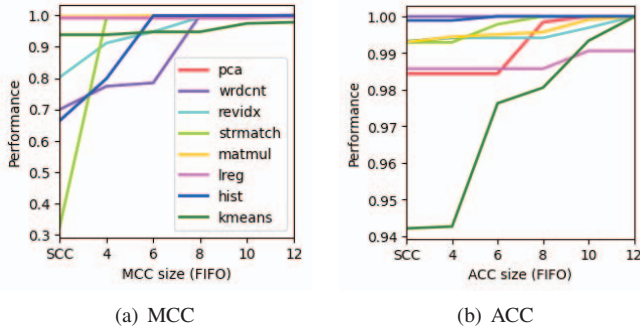


Fig. 4. Performance of increasing MCC and ACC sizes, normalized to CMOS. Note the different scales on the Y axes.

TABLE V
ACC PER-APPLICATION ENDURANCE RESULTS

	pca	wrdcnt	revidx	strmatch	matmul	lreg	hist	kmeans
% of FeFET writes	0.21	0	4.8	0.09	0.1	0.06	0.16	1.13
Point of failure	FeFET	CMOS	FeFET	CMOS	CMOS	CMOS	CMOS	FeFET

configuration when considering area and energy in Figure 3. Here we observe that increasing the size of the CMOS slice up to this point incurs a significant yet manageable area and energy penalty relative to FeFET, given the great performance gains across all applications.

Lastly, ACC enhances SCC and MCC by enabling MCC to use idle SCC registers and store part of the architectural state in CMOS registers instead. Our sensitivity study for ACC (Figure 4) shows an upward shift in performance and allows us to compare the CMOS-side “hit-rate” curves for an ACC configuration vs. the corresponding MCC design (note the different Y-axis scales). The benefits netted from ACC are largely dependent on the vector instructions involved. In general, if the number of intermediate updates for an instruction is below 3, we were able to use the spare ones as additional MCC entries. As a result, ACC performs updates on the CMOS side in over 99% of cases. While MCC would have required seven vector registers to achieve this, ACC achieves the same performance with only five, borrowing the remaining registers dynamically from the SCC entries. This, in effect, allows ACC-5 (5 ACC vector registers) to reach CMOS-like performance at 25% less area overhead than MCC-7 would. When compared to MCC-5, as in Figure 3, ACC-5 offers additional performance advantages over MCC-5 as a result of this adaptivity: ACC-5 reaches 98.3% of CMOS performance while MCC-5 follows at 91.8%.

C. Endurance and Reliability Considerations

An important effect of shifting updates from the EMT to the CMOS side is the potential to extend the design lifetime if the EMT exhibits significantly lower endurance. To characterize the endurance of the memory cells in isolation, we perform a Monte Carlo simulation of the ACC design point. We start by randomly assigning initial mappings to the CMOS and FeFET registers to identify the failure point in the memory array. The results in Table V show that, for half of the applications, the hybrid design is actually more likely to fail on the CMOS side first. For the applications where the FeFET side is the likely

point of failure, we perform further analysis and estimate the remaining life of the CMOS side to be between 1.7% to 47% (9.06% on average).

To assess reliability, we estimated the resilience of our hybrid design using results from our Monte Carlo simulations. Across the applications, we found our memory array to have an average time to failure of 268k hrs (39k - 2,123k hrs), resulting in an average lifespan of over 30 years with applications running continuously. Therefore, our design significantly alleviates the write endurance concerns associated with FeFET usage and yields levels of resilience likely to exceed the system lifetime in many environments.

D. Putting It All Together

The following are our key observations on the efficacy of our hybrid design and the mechanisms that enable it:

- **Near-CMOS Performance and Resilience:** Our hybrid architecture is barely 1% slower than a full CMOS implementation, largely due to ACC’s ability to dynamically utilize the CMOS slice for both intermediate and architectural state. This ensures 99% of writes occur on the more resilient CMOS side.
- **Enhanced Energy Efficiency:** While most writes happen on the CMOS side, a majority of reads still take place on the much larger EMT portion, which yields average energy savings of 11% with respect to the CMOS baseline.
- **Significant Area Savings:** Since the vast majority of the storage is composed of smaller, denser FeFET cells, our design nets 30% area savings with respect to pure CMOS.
- **High-Endurance Design:** Monte Carlo simulations reveal that it would take over 30 years of continuous runtime for the design to fail. Even considering the lower endurance of FeFETs, the CMOS side is expected to fail first for half of the evaluated applications.
- **Programmer Transparency:** The microarchitectural mechanisms and the hybrid nature of the implementation remain concealed from the programmer, since changes only involve the search-update microcode that implements the individual RISC-V vector instructions.

V. CONCLUSION

In this paper, by proposing a hybrid CMOS-compatible approach to APs, we have shed light on how to overcome obstacles that prevent the adoption of EMTs in the broader context of general-purpose PUM designs. Our proposed mechanisms are all relatively simple, yet effective. Our approach achieves near-CMOS performance while leveraging the energy efficiency and area advantages of EMTs, yielding a design that draws on the best of both worlds. Additionally, by dramatically reducing the number of writes to FeFET vector registers, the architecture’s endurance is significantly increased.

REFERENCES

- [1] K. A. Aabrar, S. G. Kirtania, F.-X. Liang, J. Gomez, M. San Jose, Y. Luo, H. Ye, S. Dutta, P. G. Ravikumar, P. V. Ravindran *et al.*, “Beol-compatible superlattice fefet analog synapse with improved linearity and symmetry of weight update,” *IEEE Transactions on Electron Devices*, vol. 69, no. 4, pp. 2094–2100, 2022.

- [2] E. T. Breyer, H. Mulaosmanovic, J. Trommer, T. Melde, S. Dunkel, M. Trentzsch, S. Beyer, T. Mikolajick, and S. Slesazcek, "Ultra-dense co-integration of FeFETs and CMOS logic enabling very-fine grained Logic-in-Memory," in *ESSDERC 2019 - 49th European Solid-State Device Research Conference (ESSDERC)*. Cracow, Poland: IEEE, Sep. 2019, pp. 118–121. [Online]. Available: <https://ieeexplore.ieee.org/document/8901735/>
- [3] H. Caminal, Y. Chronis, T. Wu, J. M. Patel, and J. F. Martínez, "Accelerating Database Analytic Query Workloads Using an Associative Processor," *New York*, p. 15, 2022.
- [4] H. Caminal, K. Yang, S. Srinivasa, A. K. Ramanathan, K. Al-Hawaj, T. Wu, V. Narayanan, C. Batten, and J. F. Martínez, "CAPE: A Content-Addressable Processing Engine," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, Feb. 2021, pp. 557–569, ISSN: 2378-203X.
- [5] M.-T. Chang, P. Rosenfeld, S.-L. Lu, and B. Jacob, "Technology comparison for large last-level caches (l3cs): Low-leakage sram, low write-energy stt-ram, and refresh-optimized edram," in *2013 IEEE 19th international symposium on high performance computer architecture (HPCA)*. IEEE, 2013, pp. 143–154.
- [6] D. Das and A. Khan, "Ferroelectricity in CMOS-Compatible Hafnium Oxides: Reviving the ferroelectric field-effect transistor technology," *IEEE Nanotechnology Magazine*, vol. 15, pp. 20–32, Oct. 2021.
- [7] S. Dutta, A. Khanna, H. Ye, M. Sharifi, A. Kazemi, M. Jose, K. Aabrar, J. Mir, M. Niemer, X. Hu, and S. Datta, "Lifelong learning with monolithic 3d ferroelectric ternary content-addressable memory," in *2021 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2021, pp. 1–4.
- [8] S. Dutta, H. Ye, W. Chakraborty, Y.-C. Luo, M. S. Jose, B. Grisafe, A. Khanna, I. Lightcap, S. Shinde, S. Yu, and S. Datta, "Monolithic 3D Integration of High Endurance Multi-Bit Ferroelectric FET for Accelerating Compute-In-Memory," in *2020 IEEE International Electron Devices Meeting (IEDM)*. San Francisco, CA, USA: IEEE, Dec. 2020, pp. 36.4.1–36.4.4. [Online]. Available: <https://ieeexplore.ieee.org/document/9371974/>
- [9] D. Florini, D. Gandolfi, J. Mapelli, L. Benatti, P. Pavan, and F. M. Puglisi, "A Hybrid CMOS-Memristor Spiking Neural Network Supporting Multiple Learning Rules," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2022. [Online]. Available: <https://ieeexplore.ieee.org/document/9889230/>
- [10] C. C. Foster, *Content-Addressable Parallel Processors*. John Wiley & Sons, Inc., 1976.
- [11] M. Imani, S. Gupta, Y. Kim, and T. Rosing, "Floatpim: In-memory acceleration of deep neural network training with high precision," in *2019 ACM/IEEE 46th Annual International Symposium on Computer Architecture (ISCA)*, 2019, pp. 802–815.
- [12] M. Imani, D. Peroni, Y. Kim, A. Rahimi, and T. Rosing, "Efficient neural network acceleration on GPGPU using content addressable memory," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2017, Mar. 2017, pp. 1026–1031.
- [13] M. Imani, X. Yin, J. Messerly, S. Gupta, M. Niemier, X. S. Hu, and T. Rosing, "SearchHD: A Memory-Centric Hyperdimensional Computing With Stochastic Training," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2422–2433, Oct. 2020.
- [14] S. Jeloka, N. B. Akesh, D. Sylvester, and D. Blaauw, "A 28 nm configurable memory (tcam/bcam/sram) using push-rule 6t bit cell enabling logic-in-memory," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 4, pp. 1009–1021, 2016.
- [15] S. G. Kirtania, O. Phadke, E. Sarker, K. A. Aabrar, D. Chakraborty, F. Waqar, S. Jaewon, T. Pantha, S. Dutta, A. Khan *et al.*, "Amorphous indium oxide channel fefet with write voltage of 0.9 v and endurance; 10 12 for refresh-free 1t-1fefet embedded memory," in *2024 IEEE International Electron Devices Meeting (IEDM)*. IEEE, 2024, pp. 1–4.
- [16] A. F. Laguna, M. Niemier, and X. S. Hu, "Design of Hardware-Friendly Memory Enhanced Neural Networks," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Mar. 2019, pp. 1583–1586.
- [17] X. Li, W. Zhao, Y. Cao, Z. Zhu, J. Song, D. Bang, C.-C. Wang, S. H. Kang, J. Wang, M. Nowak *et al.*, "Pathfinding for 22nm cmos designs using predictive technology models," in *2009 IEEE Custom Integrated Circuits Conference*. IEEE, 2009, pp. 227–230.
- [18] J. Lowe-Power *et al.*, "The gem5 Simulator: Version 20.0+," Sep. 2020, arXiv:2007.03152 [cs]. [Online]. Available: <http://arxiv.org/abs/2007.03152>
- [19] S. Narla, P. Kumar, A. F. Laguna, D. Reis, X. S. Hu, M. Niemier, and A. Naeemi, "Modeling and design for magnetoelectric ternary content addressable memory (tcam)," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 8, no. 1, pp. 44–52, 2022.
- [20] C. Ni, S. Chen, C.-K. Liu, L. Liu, M. Imani, T. Kämpfe, K. Ni, M. Niemier, X. S. Hu, C. Zhuo, and X. Yin, "TAP-CAM: A Tunable Approximate Matching Engine based on Ferroelectric Content Addressable Memory," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. Newark Liberty International Airport Marriott New York NY USA: ACM, Oct. 2024, pp. 1–9.
- [21] K. Ni, M. Jerry, J. A. Smith, and S. Datta, "A circuit compatible accurate compact model for ferroelectric-fets," in *2018 IEEE Symposium on VLSI Technology*, 2018, pp. 131–132.
- [22] M. Passlack, N. Tasneem, Z. Wang, K. A. Aabrar, J. Hur, H. Chen, V. D.-H. Hou, C.-S. Chang, M.-F. Chang, S. Yu *et al.*, "Direct quantitative extraction of internal variables from measured pund characteristics providing new key insights into physics and performance of silicon and oxide channel ferroelectric fets," in *2022 International Electron Devices Meeting (IEDM)*. IEEE, 2022, pp. 32–4.
- [23] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrak, "Evaluating mapreduce for multi-core and multiprocessor systems," in *2007 IEEE 13th International Symposium on High Performance Computer Architecture*, 2007, pp. 13–24.
- [24] D. Reis, K. Ni, W. Chakraborty, X. Yin, M. Trentzsch, S. D. Dinkel, T. Melde, J. Müller, S. Beyer, S. Datta, M. T. Niemier, and X. S. Hu, "Design and analysis of an ultra-dense, low-leakage, and fast fefet-based random access memory array," *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits*, vol. 5, no. 2, pp. 103–112, 2019.
- [25] S. Beyer *et al.*, "FeFET: A versatile CMOS compatible device with game-changing potential," *2020 IEEE International Memory Workshop (IMW)*, pp. 1–4, May 2020.
- [26] S. Slesazcek, U. Schroeder, and T. Mikolajick, "Embedding hafnium oxide based fefet in the memory landscape," in *2018 International Conference on IC Design & Technology (ICICDT)*, 2018, pp. 121–124.
- [27] A. Stillmaker and B. Baas, "Scaling equations for the accurate prediction of cmos device performance from 180nm to 7nm," *Integration*, vol. 58, pp. 74–81, 2017. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167926017300755>
- [28] V. Vashishtha, M. Vangala, and L. T. Clark, "Asap7 predictive design kit development and cell design technology co-optimization," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 2017, pp. 992–998.
- [29] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovic, "The RISC-V Instruction Set Manual," vol. I, p. 1356, 2011, ISBN: 9781467303422.
- [30] S. S. Wong, C. C. Tamarit, and J. F. Martínez, "PUMICE: Processing-using-Memory Integration with a Scalar Pipeline for Symbiotic Execution," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, Jul. 2023, pp. 1–6. [Online]. Available: <https://ieeexplore.ieee.org/document/10247818>
- [31] L. Yavits, S. Kvatinsky, A. Morad, and R. Ginosar, "Resistive Associative Processor," *IEEE Computer Architecture Letters*, vol. 14, no. 2, pp. 148–151, Jul. 2015. [Online]. Available: <http://ieeexplore.ieee.org/document/6966736/>
- [32] L. Yavits, A. Morad, and R. Ginosar, "Computer Architecture with Associative Processor Replacing Last-Level Cache and SIMD Accelerator," *IEEE Transactions on Computers*, vol. 64, no. 2, pp. 368–381, Feb. 2015, conference Name: IEEE Transactions on Computers.
- [33] X. Yin, K. Ni, D. Reis, S. Datta, M. Niemier, and X. S. Hu, "An ultra-dense 2fefet tcam design based on a multi-domain fefet model," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 66, no. 9, pp. 1577–1581, 2018.
- [34] S. Yu and P.-Y. Chen, "Emerging memory technologies: Recent trends and prospects," *IEEE Solid-State Circuits Magazine*, vol. 8, no. 2, pp. 43–56, 2016.
- [35] Y. Zha and J. Li, "Hyper-Ap: Enhancing Associative Processing through A Full-Stack Optimization," *Proceedings - International Symposium on Computer Architecture*, vol. 2020-May, pp. 846–859, 2020, ISBN: 9781728146614.