Advanced Intelligent Systems

ADVANCED INTELLIGENT SYSTEMS Open Access

www.advintellsyst.com

RESEARCH ARTICLE OPEN ACCESS

A Flexible and Energy-Efficient Compute-in-Memory Accelerator for Kolmogorov–Arnold Networks

Chirag Sudarshan[1] | Oliver Artner[2,3] | Paul-Philipp Manea[1,2] | Sebastian Siegel[1] | Susanne Hoffmann-Eifert[3] | Regina Dittmann[2,3] | John Paul Strachan[1,2]

[1]Forschungszentrum Jülich GmbH, Peter Grünberg Institute 14, Jülich, Germany | [2]Faculty of Electrical Engineering and Information Technology, RWTH Aachen University, Aachen, Germany | [3]Forschungszentrum Jülich GmbH, Peter Grünberg Institute 7, Jülich, Germany

Correspondence: Chirag Sudarshan (c.sudarshan@fz-juelich.de) | John Paul Strachan (j.strachan@fz-juelich.de)

Received: 30 September 2025 | **Revised:** 23 November 2025 | **Accepted:** 16 December 2025

Keywords: compute-in-memory | Kolmogorov–Arnold networks | memristors | nonlinear computing

ABSTRACT

Emerging Kolmogorov–Arnold networks (KANs) replace the linear weights of neural networks with trainable nonlinear functions. This modification is particularly attractive for scientific computing, where KANs can match the accuracy of conventional multilayer perceptrons (MLPs) while reducing model size by up to 100×. However, this efficiency comes at the cost of computationally expensive nonlinear evaluations, unlike conventional MLPs dominated by linear matrix multiplications. We present a flexible and energy-efficient compute-in-memory accelerator tailored for KANs, developed through cross-layer optimization across algorithm, architecture, circuit, and device levels. The accelerator computes arbitrary nonlinear functions using a single-read scheme and read-optimized memory arrays with nonvolatile memristive devices. Our system achieves a lowest energy of 8.69 pJ per KAN function. In terms of energy-delay product, it provides 1996× improvement over CPUs, 208× over standard MLP-oriented compute-in-memory accelerators, and up to 71× over prior KAN accelerators. These results establish energy-efficient hardware primitives for implementing advanced nonlinear networks in scientific computing.

1 | Introduction

Deep learning models have achieved state-of-the-art accuracy on multiple fronts, including image processing, natural language processing, translation, and beyond. These advancements have driven the integration of Artificial Intelligence (AI) across a wide range of systems, spanning from high-performance to edge computing. However, as AI models scale up, their escalating costs and energy demands are a critical challenge for sustaining this progress [1, 2]. To address the energy issue, a shift towards Application Specific Integrated Circuit (ASIC) accelerators can offer much higher energy efficiency than traditional General-Purpose (GP) chips like GPUs and CPUs [3, 4]. In this context, Compute-in-Memory (CIM) is a prominent architectural approach that offers orders of magnitude higher energy efficiency (i.e. 100−1000×) compared to GP chips [5–11]. Here, memory and computation are tightly integrated to reduce the dominating costs of frequent data transfers encountered in traditional GP architectures. Furthermore, CIM leverages the inherent physical laws of crossbar array circuits (Ohm's and Kirchhoff's laws) to perform highly parallel and energy-efficient Multiply-Accumulate (MAC) or vector-matrix multiplication (VMM) operations that prevail in Multilayer Perceptrons (MLPs) and other Deep Neural Networks (DNNs) [12, 13]. Here, we refer to this type of existing VMM optimized CIM, best suited for MLP or DNN, as VMM-CIM.

Currently, VMM-CIM architectures offer less benefits to many forms of scientific computing or emerging AI models that require a large number of nonlinear computations. One such recent model is the Kolmogorov–Arnold Network (KAN) [14, 15], which explicitly targets AI + Science applications. KAN is based on the Kolmogorov–Arnold (KA) theorem [16], which states that any multivariable continuous function can be represented as a superposition of single-variable functions. KAN builds on this theorem

with two key principles: 1) unlike the original KA theorem with only two layers ($L = 2$), KAN proposes to have $L > 2$ layers, and 2) unlike MLP, which trains the weights between nodes, KAN trains nonlinear activation functions. In comparison to MLP, KAN can reduce the parameter count by 100× without significant accuracy loss [14]. Research on KAN has rapidly grown with numerous studies exploring extensions and adaptations [17–25]. KAN's ability to represent complex multivariable equations or datasets as L-layer superpositions of single-variable functions is advantageous in domains like scientific computing, physical modeling, circuit simulations, and materials simulations where MLP or DNNs are yet to reach state-of-the-art accuracy [26]. However, the nonlinear functions central to KAN are energy-inefficient and latency-intensive on standard computing platforms [27, 28]. On a GPU, KAN layers exhibit $5 - 100\times$ higher latency than even larger MLP layers, with latency scaling more rapidly as layer size increases (see latency graph in Figure 1). Similarly, existing VMM-CIM architectures are not well-suited for KAN. Developing specialized hardware to address this is essential for KAN models to be able to scale-up, analogous to the extreme scaling of MLPs and DNNs.

This work presents KA-CIM, a custom hardware accelerator designed to efficiently compute KANs using an in-memory approach. KA-CIM targets several use cases, including energy-efficient and low-latency evaluation of scientific equations converted into KAN form, KAN inference for complex physical models, and fast derivative computations. The accelerator reaches high energy efficiency and low latency through systematic co-optimization spanning algorithm, architecture, circuit, and device levels. Piecewise-linear (PWL) approximation transforms nonlinear computations into single multiply–accumulate (MAC) operations, yielding fixed latency and energy independent of function complexity. At the same time, the PWL approximation is shown to be low error. At an architectural level, KA-CIM adopts a read-centric design in which all computations are executed through single array reads and compact array dimensions. These choices enable aggressive circuit- and device-level optimization, including a memristive array with a proposed RC-sensing scheme that achieves extremely low read latency and energy. Through this end-to-end co-optimization, KA-CIM attains less than 60 ns latency and 60 pJ energy for a wide range of multivariable functions, demonstrating a promising path toward practical and efficient scientific computing at the edge.

Critical memristive device characteristics that complement the RC-discharge-based sensing scheme form the device-level foundation of KA-CIM's performance. A low-ohmic low resistance state (LRS) shortens the RC time constant (and hence latency), a large separation of LRS and high resistance state (HRS) improves sense margin, and a small parasitic capacitor precharge value defines the read energy. We fabricated and characterized $Pt/HfO_2/TiO_x/Ti/Pt$ memristive devices (100×100 nm) that exhibit the target LRS of 10 kΩ, HRS of ∼3 MΩ, ON/OFF ratio exceeding 100×, and a read voltage of 0.4 V. This device–circuit co-optimization, together with small array dimensions, achieves a read latency of 1.5 ns and a read energy of 4.1 fJ/bit, providing the device-level basis for the superior system-level efficiencies achieved by KA-CIM.

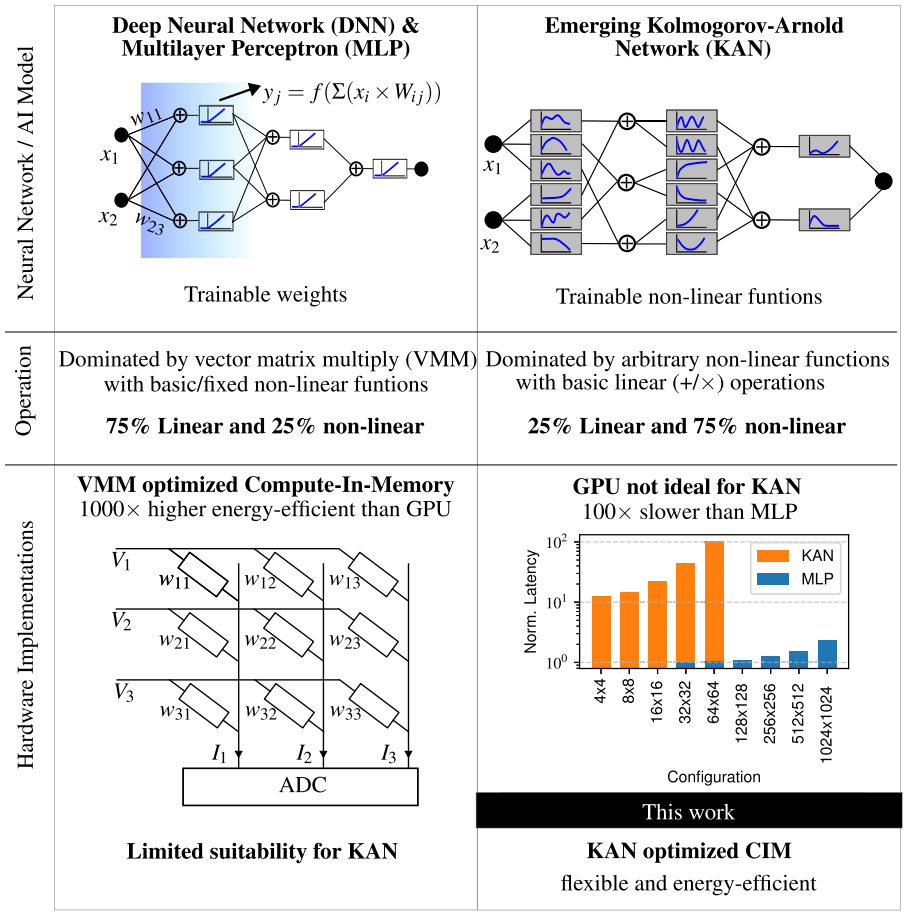


FIGURE 1 | Overview and motivation for KAN-specific hardware accelerator.

At the system level, we designed a 16-core KA-CIM accelerator capable of parallel computing of 384 nonlinear KAN functions. To assess flexibility, we evaluated 7 single-variable and 8 multivariable tasks, including the Hodgkin–Huxley neuron model. Across all benchmarks, KA-CIM maintained minimal error (median 10^{-4}–10^{-3}) relative to 32-bit floating-point baselines. Overall, KA-CIM achieves a 1996× improved energy–delay product over a conventional CPU and a 35× over a dedicated ASIC [28] for nonlinear computations. It also surpasses a 100-TOPS/W VMM-CIM executing the same task through an MLP, offering up to 27× lower energy and 7.4× lower latency. Compared with other CIM-based KAN accelerators [29, 30], KA-CIM provides 22×–27× lower latency and 42×–71× better energy–delay product than [29], and 2.31× lower power than [30]. A further distinguishing feature is KA-CIM's ability to compute both function outputs and their (partial) derivatives simultaneously, a capability absent in prior KAN accelerators [29–31].

2 | Results

As shown in Figure 1, KAN is fundamentally different from standard DNNs or MLPs. Over the years, CIM hardware accelerators [8, 9, 32] have been extensively optimized for DNNs, supporting highly parallel and energy-efficient VMM. CIM designs explore a particular trade-off in array dimensions, multilevel cells, and ADCs but are not ultimately suited for arbitrary nonlinear computations. Recent efforts [29] applied VMM-CIM to KANs, but the computation of the predominant nonlinear basis are handled by standard CMOS logic. To match the low precision available in analog VMM-CIM (e.g., 8-bit), these designs also require hardware-in-the-loop retraining and sparsity-aware weight mapping. Rather than adapting VMM-CIM to KAN through complex methodologies, we introduce a KAN-specific fully memory-centric accelerator developed via cross-layer co-optimization of algorithm, architecture, circuit, and device. Furthermore, it also does not require any hardware-specific retraining. We present results for our fully memory-centric implementation natively tailored to KAN in this section.

2.1 | Algorithmic Approximation of KAN Nonlinear Functions

KA-CIM begins by converting a target dataset or multivariable equation into its KAN representation (Equation (1)), where L denotes the number of layers, K the number of inputs, K_L the number of activation functions in layer L, $\phi(x)$ the single-variable nonlinear functions, and i_n represent the summation index. A central challenge for hardware acceleration of KANs is the efficient and highly parallel computation of nonlinear functions. To address this, our approach applies a first-order PWL approximation, reducing complex nonlinear functions to a single multiply–accumulate operation. This guarantees fixed and predictable latency and energy consumption, independent of function complexity. Each trained function $\phi(x)$ is approximated by partitioning it into N linear segments (Figure 2a). The segmentation granularity is determined by the function's smoothness, with larger N yielding greater accuracy. For a given input x, the appropriate segment is selected, and the function is approximated as $\phi(x) \approx M_S \cdot x + Y_S$ where M_S and Y_S are the slope and intercept of the chosen segment. This simplification requires storing the segment parameters (breakpoints, slopes, and intercepts) in memory, a feature that motivates our memory-centric design.

$$F(x_1, x_2, \ldots, x_K) = \sum_{i_L}^{K_L} \phi_{L,i_{L+1},i_L} \left(\cdots \left(\sum_{i_1}^{K_1} \phi_{1,i_2,i_1} \left(\sum_{i_0}^{K} \phi_{0,i_1,i_0}(x_{i_0}) \right) \right) \right) \quad (1)$$

2.2 | Fully Memory-Centric Hardware Architecture

Building on the algorithmic foundation, our design introduces two key architectural innovations for the efficient realization of PWL-based KAN computation. First, all memory arrays used for PWL segment computation and parameter storage operate in a single-row read mode, similar to conventional memories but in sharp contrast to VMM-CIM designs that rely on multirow activation for analog VMM. This read-centric approach is motivated primarily by the need to suppress variation-induced errors common in analog VMM-CIM accelerators. An additional major benefit is the elimination of area-, latency-, and power-intensive ADCs, replaced by lightweight, low-energy, low-latency sense amplifiers. Crucially, this simplification does not introduce extra latency through multiple reads; the architecture performs single-variable function computation with one read per array and a latency of less than 10 ns, which is significantly faster than typical VMM-CIM implementations. Second, because KA-CIM is inherently read-centric, reductions in array read energy and latency directly translate into overall performance and energy gains. To exploit this, the system uses compact 16×64 and 32×16 arrays, whose dimensions remain fixed irrespective of the PWL segmentation factor N, enabling aggressive circuit- and device-level optimizations to further reduce read energy and latency. The remainder of this section first details our novel single-read CIM architecture, followed by the strategy and its evaluation results used to keep array dimensions compact regardless of N.

2.2.1 | Single-Read CIM Hardware for PWL Approximation

The fundamental building block of the proposed design is the KA-CIM tile (Figure 2b), designed for energy-efficient computation of an arbitrary PWL-approximated single-variable KAN function $\phi(x)$. KA-CIM tiles operate in parallel to evaluate all KAN functions $\phi(x)$, whose outputs are then aggregated to produce the final result. To compute the PWL approximation of $\phi(x)$, each tile performs three steps, implemented by dedicated hardware units. First, a CIM-based segment selection unit (CIM-SSU) identifies the appropriate PWL segment by comparing the BFloat16 input x with N stored breakpoints, producing an N-bit segment index $[S_N, \ldots, S_1]$ with exactly one active bit indicating the chosen segment. Second, this selection signal is used to retrieve the corresponding slope M_S and intercept Y_S values from dedicated memory arrays, each sized $N \times 16$, where every row stores one BFloat16 value. Finally, a digital multiply–accumulate (MAC) unit computes the output as $\phi(x) \approx M_S \cdot x + Y_S$, operating near the memory periphery to minimize data movement.

Figure 2c details the in-memory segment selection using a single read per array. The 16-bit breakpoint value X_S is divided into four

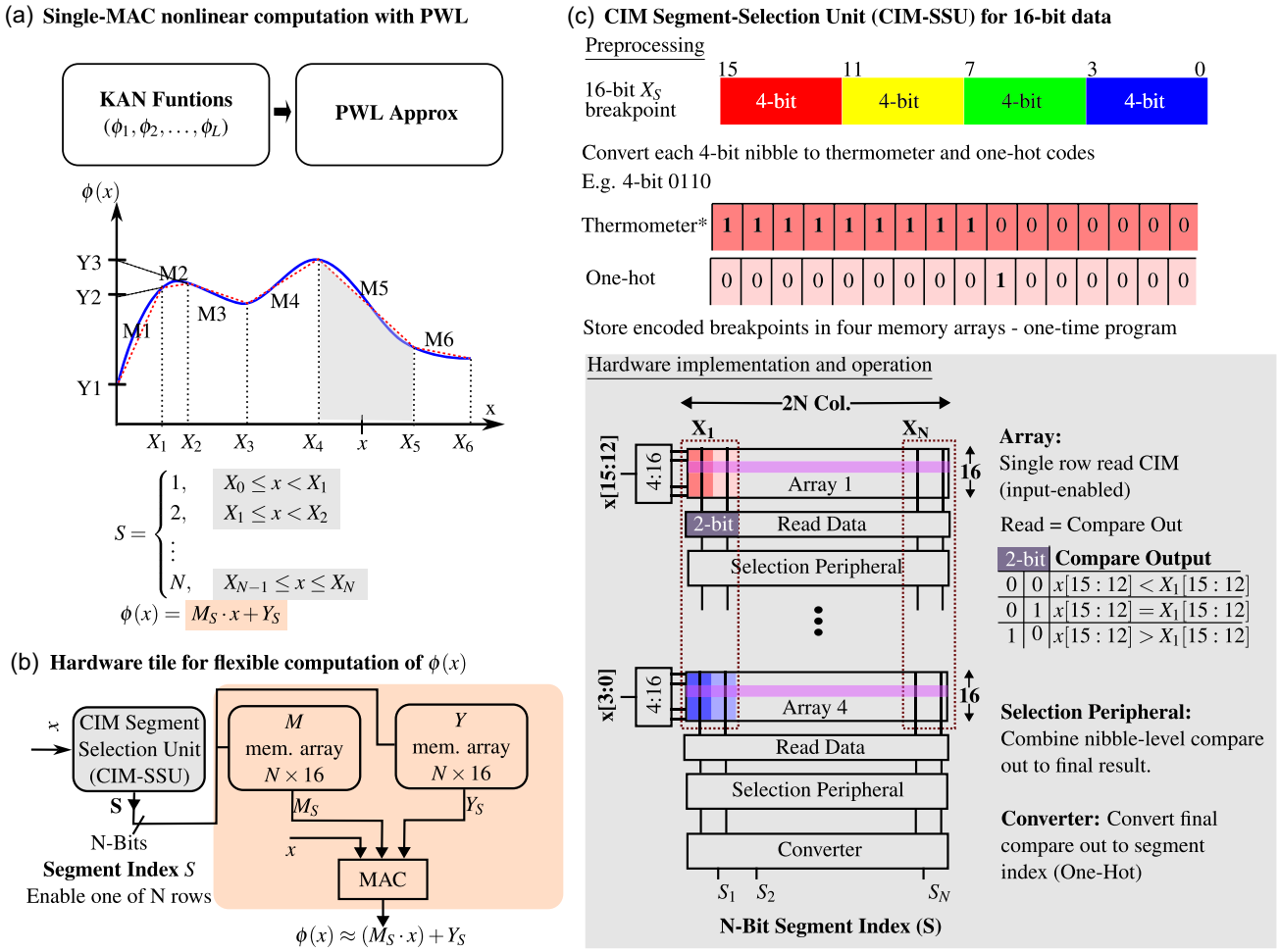


FIGURE 2 | Single-MAC CIM hardware implementation for evaluating nonlinear KAN functions $\phi(x)$ using PWL approximation. (a) Example PWL representation with breakpoints X_S, slopes M_S, and intercepts Y_S. (b) Hardware schematic of a single KA-CIM tile comprising: Block 1, a compute-in-memory (CIM) segment-selection unit (CIM-SSU) that identifies the segment index S and retrieves the corresponding slope M_S and intercept Y_S; Block 2, memory arrays storing all M_S and Y_S; and Block 3, a multiply–accumulate (MAC) stage evaluating $\phi(x) \approx M_S \cdot x + Y_S$. (c) CIM-SSU implementation for 16-bit inputs, illustrating breakpoint encoding and comparison operation with a single row read.

4-bit groups. Each group is encoded into a 16-bit thermometer and one-hot code: the thermometer code performs greater-than or less-than comparisons, while the one-hot code detects equality. The encoded groups are stored in four compact $16 \times 2N$ arrays, with two columns reserved per breakpoint. The N breakpoints are organized in ascending order across the $2N$ columns. During operation, the 16-bit input is partitioned into four corresponding 4-bit groups via hardwired connections and applied to the respective arrays. Within each array, the 4-bit input group activates a single row, and the sensed data directly represents the comparison results. The $2N$-bit readout–arranged as two bits per breakpoint–encodes the comparison outcome (see the truth table in Figure 2c). For example, a code of '10' indicates that the input group exceeds the corresponding breakpoint group. A lightweight selection peripheral aggregates the comparison outputs from all groups, while a converter block determines the transition point where the result changes from 'less than' to 'greater than or equal to' and activates the corresponding segment index S. Both the selection peripheral and converter require only a small number of logic gates per breakpoint, minimizing area overhead. All four CIM-SSU arrays are sensed in parallel, and the total delay is limited to a single array read plus a few gate delays in the selection peripheral and converter. Overall, the computation of an arbitrary single-variable function, irrespective of the fuction type, follows the same sequence: four parallel read operations plus a few logic operation in the CIM-SSU for segment selection, two parallel reads for retrieving M_S and Y_S, and a final single MAC operation to compute $F(X)$ output. Consequently, each tile exhibits fixed energy and latency per function for a given N. The total tile energy primarily depends on the array read energy, which scales with N, ensuring that the computation cost is determined by N rather than the function type. The design supports signed, fixed-point, and floating-point representations. Full implementation details, including 16-bit examples, are provided in Figure S1, S2, S3, and Table S1 of Supporting Information. Overall, whether in the CIM-SSU or in the M and Y arrays, all components are based on a single-read architecture.

2.2.2 | N-Independent KA-CIM Tiles with Compact Arrays

The accuracy and energy consumption of a KA-CIM tile (Figure 2) depend on the number of PWL segments, N. Figure 3a illustrates this trade-off for an exponential function, where increasing N reduces approximation error (logarithmic scale) while the

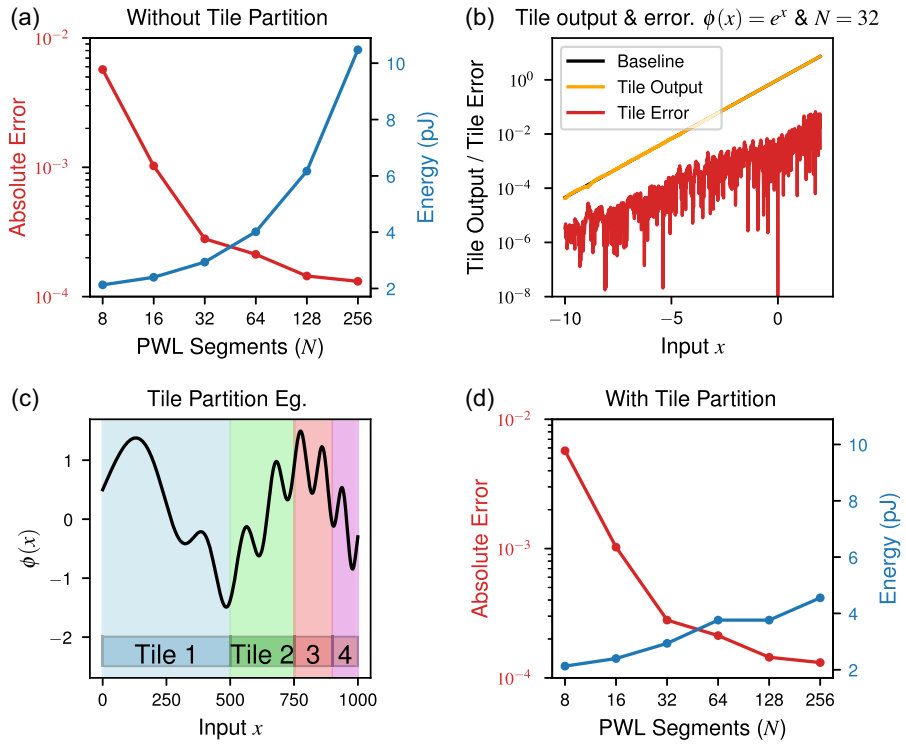


FIGURE 3 | Tile partitioning approach for emulating large-N PWL functions with $N = 32$ tile design. (a) Tile error and energy (excluding communication and chip-level peripheral overhead) as a function of the number of PWL segments (N), where larger N typically requires proportionally larger arrays. Similar evaluations were conducted for other elementary functions (see Figure S4 of Supporting Information). The crossover point between energy and error occurs around $N = 32$ (either between 16 and 32 or 32 and 64) in all the considered functions. This motivates our choice of $N = 32$ as the standard tile configuration. (b) Tile output error for $N = 32$ when computing an exponential function; error scales with output magnitude. Baseline FP32 results (black) largely overlap with tile outputs (orange). (c) Tile partitioning example: functions requiring $N > 32$ are split into multiple groups, each mapped to a dedicated $N = 32$ tile. (d) Illustration of the tile partitioning approach to emulate higher N values with $N = 32$ tiles. The energy increases at a much lower rate for all $N > 32$ while continuing to reduce approximation error.

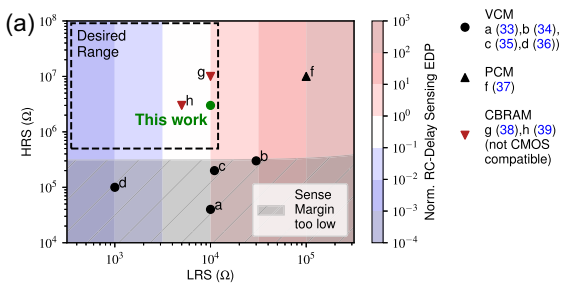
corresponding energy cost increases comparatively slowly. In Figure S4 of Supporting Information, we provide corresponding results for several other standard single-variable elementary functions. Across all evaluated functions, the crossover point between error and energy consistently occurs near $N = 32$ (i.e., either between 16 and 32 or between 32 and 64). Based on this trend, we adopt $N = 32$ as the default configuration for the array dimensions within each tile. Accordingly, the four CIM-SSU arrays are sized at 16×64 (i.e. $16 \times 2N$), while the M_S and Y_S arrays use 32×16 dimention (i.e. $N \times 16$). With these dimensions, KA-CIM achieves very low output error relative to baseline FP32 computations, despite the example functions having large dynamic output ranges (e.g., 4.54×10^{-5} to 7.389). Across the entire output range, the approximation error remains consistently small relative to the output value (Figure 3b). In Figure S4 or Supporting Information, we also observe that the energy values remain identical across all evaluated functions, showcasing the fixed tile energy.

At the system level, $N = 32$ suffices for many applications, presented in later sections. However, to support generic adaptation of KA-CIM to functions with wider input domains or more complex behavior that require higher accuracy, the architecture must be capable of handling larger N values. To support higher N without increasing array dimensions, we introduce a tile partitioning approach, which reuses the $N = 32$ tile design to emulate larger N. As illustrated in Figure 3c, the domain of $\phi(x)$ is divided into groups, each assigned to a dedicated tile configured for $N = 32$. For example, if $\phi(x)$ spans 0–2000, it can be split into four groups (0–500, 500–750, etc.), each mapped to a separate tile. When an input x falls within a particular group, only the corresponding tile is activated, while others remain idle. A lightweight digital comparator in each tile ensures that the tile is only enabled if the input lies within its configured range, before fine-grained CIM-based segment selection and $\phi(x)$ computation. Consequently, for any given input, only a single tile is active with a small additional energy for digital comparators, thereby minimizing energy consumption even for $N > 32$. In this way, multiple tiles collectively approximate a function with $N > 32$ while maintaining the energy efficiency of a single $N = 32$ tile. Figure 3d illustrates tile partitioning for computing functions with $N = 64$ (two tiles), $N = 128$ (four tiles), and $N = 256$ (eight tiles) using $N = 32$ tiles. Energy consumption increases at a much lower rate for all $N > 32$, whereas the approximation error continues to decrease with increasing N. The energy overhead is limited to approximately 0.79 pJ per additional 128 increase in N, corresponding to the lightweight digital comparator whose energy increments for every additional four tiles per function. For instance, for $N = 256$, the total energy is 4.38 pJ with tiling, which is substantially lower than the energy required for a lower $N = 128$ implementation without tiling. This tiling approach allows the KA-CIM tile to use compact $N = 32$ based array dimensions without compromising accuracy. It also provides architectural support for computing any complex

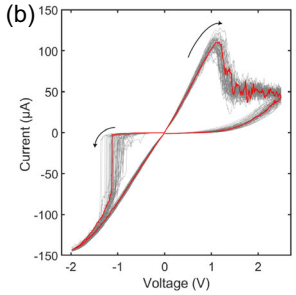
function. The low array dimension further enables aggressive circuit and device optimizations, achieving low read latency and energy, as detailed in the next section.

2.3 | Device-Circuit Codesign for Low-Latency and Low-Energy Memristor Array

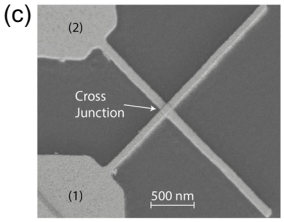
The aforementioned architectural advantages are leveraged at the circuit and device levels to design read-optimized compact arrays, which directly drive KA-CIM's overall latency and energy efficiency. Through a circuit–device codesign strategy, we achieve aggressive array-level targets of <2 ns read latency and 3–4 fJ/bit read energy. To realize these aggressive targets, we use an RC-discharge-based sensing approach, where read energy is governed by the capacitor's precharge voltage and latency by the RC time constant. This approach imposes specific constraints on the VCM device: the LRS must discharge the capacitor quickly, while the HRS must maintain charge long enough to provide a robust sensing margin. Figure 4a shows the HRS and LRS of various reported memristor devices. Our first evaluation



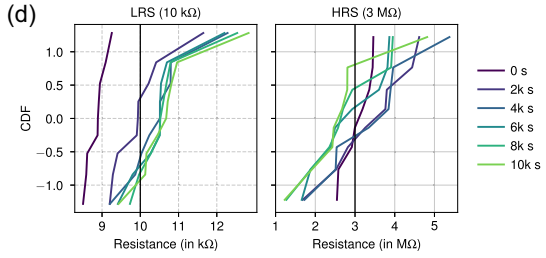
Requirements on LRS and HRS for low Energy-Delay Product (EDP)



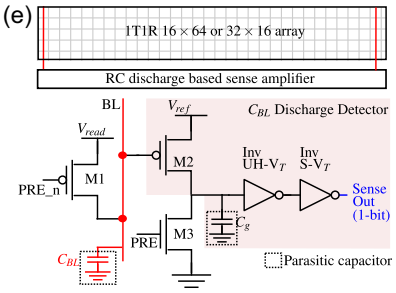
$I-V$ switching characteristic of the 1T1R cell



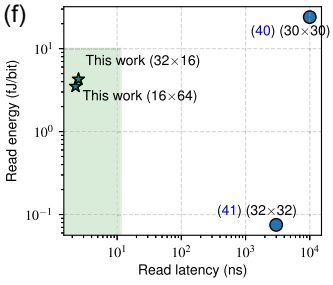
The CMOS integrated VCM nano crossbar device



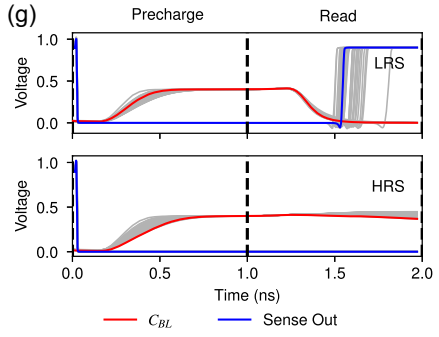
Read Drift of LRS and HRS



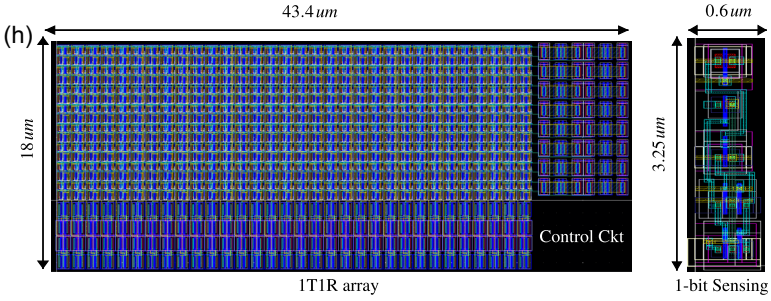
RC discharge time based sense amp.



Array read latency and energy



CMOS and RRAM variation on sensing



Scaled area of 16 × 64 1T1R array and sensing circuit in 28 *nm*

FIGURE 4 | Overview of the sensing performance of different resistive switching devices and characteristics of the metal oxide-based VCM device used in this work. (a) Impact of the LRS and HRS on RC discharge sense margin and energy delay product for various device types [33–39]. The sense margin is considered to be low if the difference of LRS and HRS corresponding RC delay for a bitline parasitic capacitance of 1.7 pF is less than 5 ns. Beside VCMs, area-type devices made from (Pr, Ca) MnO_3 (PCMO), phase change devices (PCM), and conductive bridge cells (CBRAM) are considered. The non-CMOS compatibility indicated here is specific to the cited work and not for the technology itself. (b) Current–voltage (I–V) characteristics of the VCM device measured in a one-transistor-one memristor configuration using a transistor made with 180 nm technology. (c) Scanning electron microscopic image of the 100 × 100 nm nano crosspoint Pt/3 nm HfO_2/3 nm TiO_x/10 nm Ti/Pt device. (d) Read drift behavior of the VCM device measured with a read voltage of 0.4 V for 1 s every 1000 s. (e) Circuit diagram of the proposed RC-discharge based low-latency, low-energy sensing, optimized for compact array dimensions. (f) Comparison of array read energy and latency (including the precharge overhead) with prior works [40, 41]. The read energy and latency evaluation in our work includes CMOS variations using Monte Carlo simulations and considers worst-case resistance values, i.e., minimum HRS and maximum LRS (see Figure S6). (g) Impact of RRAM and CMOS variations on sensing latency, evaluated with 50 Monte Carlo simulations across 64 sense amplifiers per array. (h) Scaled area of the compact array and proposed sensing circuit in 28 nm technology.

examines the impact of different device resistance values on RC delay times and, consequently, on sensing reliability. For this analysis, we use a wire capacitor value of 1.7 fF/12.5 µm, extracted from post-layout bitline parasitics. When the RC delay difference between the LRS and HRS falls below 5 ns, the sensing margin becomes insufficient to ensure robustness under worst-case device variations. The RC delay also affects the overall energy consumption. Although the energy stored in the capacitor is independent of device resistance ($E = \frac{1}{2}CV^2$), the longer RC delay requires the sensing circuit to remain active for a longer duration. Consequently, the total energy consumed during sensing increases with device resistance. Figure 4a shows the calculated energy delay product (EDP, in color) as a function of (HRS, LRS) combinations. Limitations are defined by a low sense margin, a very high latency, or a high EDP. To maintain a sufficient sense margin and low EDP, the LRS should be $\leq 15\,k\Omega$ and the HRS should be $\geq 500\,k\Omega$. By carefully selecting the memristive device parameters to align with the sensing circuit design constraints, we can directly optimize the array read latency and energy. The benefits of the RC-sensing scheme is particularly pronounced for compact arrays, since larger arrays introduce additional leakage paths that can compromise sensing reliability and increase both latency and energy consumption. We first describe the details of the VCM device that meets these criteria, followed by the sensing circuit details.

The memristive devices are realized in 100×100 nm nano crossbar structure with a Pt/3 nm HfO_2/3 nm TiO_x/10 nm Ti/Pt stack, manufactured in a CMOS-compatible back-end-of-line process. Figure 4c shows a microscopic image of the Nanometer-scale VCM device located at the intersection of the two electrode fingers. The fabrication process is described in the "Materials and Methods" section. For monolithic integration, Pt can be replaced by TiN without changing the switching behavior. Our device stack results from a modification of the commercially available HfO_2/Ti stack [42], by addition of a thin TiO_x layer that enables an analog-type switching in a limited resistance range (see VCM d [37] in Figure 4a). Rather than introducing new materials or specialized processing, our approach investigates the established filamentary-type VCM device under a broader voltage regime to achieve the target resistance window. The device is integrated with a 5 V, 180 nm NMOS access transistor in a one-transistor–one-memristor (1T1R) configuration [43]. A representative bipolar resistive switching characteristics is shown in Figure 4b. Following the established programming schemes of 1T1R elements, a positive and negative sign indicates that the voltage is applied via the bit line (BL) and the select line (SL), respectively (see [43] for details). For SET, a current compliance of 150 µA defined by the voltage on the wordline (WL) controls the LRS.

In our KA-CIM design, the programming of PWL parameters (i.e., write operation) is infrequent, and inference computations of nonlinear functions for a long stream of data is the main operation. Since the computation is primarily dependent on array read, this requires long-term stability of the resistance states. However, drift over time and read-disturb are commonly reported for similar memristive devices [44, 45]. To account for this effect, we have chosen LRS and HRS values of 10 and 3 MΩ, respectively, that correspond to the limits given by the sensing circuit, in detail, LRS $\leq 10\,k\Omega$ and HRS $\geq 500\,k\Omega$. The stability of the states was tested by drift measurements, for which ten programmed binary states were continuously read at a READ voltage of 0.4 V for 1 s every 1000 s for a total time of 10000 s. The resulting resistance values are shown in Figure 4d as CDF curves for both the LRS and HRS, where the colors encode the different time stamps. The LRS exhibits only a minor drift, with a maximum value of 13.5 kΩ, whereas the HRS shows a broader spread. The limited variation in LRS helps maintain low RC delay fluctuations and consequently low sensing latency, while an HRS above 1 MΩ ensures a sufficient sensing margin. In Figure S5 of Supporting Information, we present the extrapolation results of read drift analysis beyond 10 ks. The physical explanation behind the tilting of the HRS CDF curve is the diffusive migration of oxygen vacancies from the vacancy-rich filament into the vacancy-poor area in front of the active electrode and vice versa, thereby decreasing/increasing the resistance of the memristive devices [46]. In addition to the read drift, read disturb measurements were successfully performed with 32 consecutive read pulses of 16 s width; see Figure S6. From a system point of view, this duration corresponds to 320 billion read operations per cell, which is a substantially large number. In the worst case, the HRS reaches a value down to 550 kΩ, while the LRS rises to up to 15 kΩ. We will show later that the proposed sense amplifier can handle this variation without any sensing error. More details on read disturbance CDF, cycling endurance measurements, and device-to-device variability of the VCMs are provided in Figure S6, S7, and S8 of Supporting Information. The 1T1R structures shown here used a large 5 V access transistor in 180 nm technology node. In contrast, KA-CIM is envisioned for implementation in advanced 28 nm technology nodes, using compact 1.8 V NMOS access transistors to minimize cell area to 565×800 nm while still meeting the memristors' voltage requirements. This cell area is consistent with the area ($0.4 - 0.5\,\mu m^2$) of various emerging memories, including memristive devices [47]. The scaled 1T1R cell primarily serves to estimate the array area, while the array read energy and latency (detailed in the next paragraph) are largely determined by the VCM resistance values. Figure S11 and Table S2 of Supporting Information demonstrates that a 28 nm 1T1R cell with a 1.8 V access transistor can meet the required voltage and current compliance for the investigated VCM device. Furthermore, motivated by the increasing interest from technology providers in supporting emerging memory IPs, we reasonably assume that advanced CMOS nodes will provide suitable, low-area, memory device-specific transistors in the future.

Figure 4e shows the circuit of the proposed RC-discharge-based sensing scheme. C_{BL} is the parasitic capacitance, whose value is determined by both the bitline wire and the parasitic capacitances of all transistors connected to it. During read operation, C_{BL} is precharged to a configurable voltage of 400 mV. The VCM LRS discharges the capacitor rapidly, whereas the HRS leaks very slowly, keeping the voltage above the trigger threshold throughout the sensing period (Figure 4g). Unlike conventional analog comparators that continuously draw current, the proposed detector relies on inverters that only consume energy during switching. When C_{BL} discharges close to zero, the ultrahigh V_T PMOS $M2$ turns ON, charging the gate parasitic capacitor C_g to the configurable reference voltage (V_{ref}), which triggers the inverter to sense the LRS. For HRS, C_{BL} remains well above the trigger point, keeping $M2$ near OFF or weakly ON, so C_g cannot reach V_{ref} within the sensing interval. As shown in Figure 4f, it achieves a simulated worst-case read latency of 1.25 – 1.5 ns (excluding precharge) and a read energy of 3.5 – 4.2 fJ/bit, forming the foundation for KA-CIM's overall performance gains. The ultralow

energy and latency result from the combination of small precharge voltage, inverter-based trigger, and low VCM LRS with high ON/OFF ratio. These results incorporate transistor, memristor, and parasitic capacitor variation evaluated through 50 Monte-Carlo simulations across 64 sensing circuits per array (equivalent to 3200 evaluations) and 288 sensing circuits per tile (equivalent to 14 400 evaluations). The reported read latency of 1.5 ns accounts for worst-case resistive and capacitive variations, including transistor parasitics. This represents a conservative upper bound corresponding to large parasitic capacitance at an LRS of 10 kΩ; under typical conditions, the read latency is below 1 ns (see Figure 4g). Our variation analysis also shows that the sensing margin remains sufficient to reliably distinguish LRS and HRS, ensuring robust read operation even under long-term drift and read-disturb conditions (Figure 4g). Detailed timing and voltage waveforms of the RC-discharge sensing circuit are provided in Figure S12 of Supporting Information. Figure 4h shows the layout and area of the 1T1R array and the sensing circuit in 28 nm technology.

2.4 | KA-CIM Accelerator, Floorplan, Area, and KAN Mapping

The KA-CIM accelerator features a hierarchical architecture composed of cores, macros, and tiles, designed to efficiently compute KANs. At the macro level, four tiles share a digital MAC unit and tile enable comparators (i.e. 4× - one per tile). The tile enable comparators are used during tile partitioning for selective activation of tiles. The accelerator can concurrently compute up to 384 single-variable nonlinear functions. The single variable outputs from multiple tiles are summed using the Σ blocks to produce each KAN layer output (e.g., $\sum_{i_0}^{n_0} \phi_{0,i_1,i_0}(x_{i_0})$). These results propagate through successive layers in a pipelined, tile-parallel fashion, ultimately computing the final output. Figure 5a illustrates the floorplan and area breakdown of the 1.875 Mb accelerator, which is designed using the proposed 1T1R array and the BFloat16 data type. It is implemented in 28 nm technology and occupies a total area of 3.14 mm^2. To further enhance the area efficiency, the KA-CIM architecture can leverage the 3D stacking capability of emerging memory technologies [48–50] for improved area efficiency and memory capacity in the future. As a proof-of-concept, all hardware evaluations presented in this work are for a 2D design while the architecture is scalable for 3D.

Figure 5b–e presents the example 12-variable KAN, its mapping, and the energy breakdown. For this evaluation, we consider a nominal mapping that uses two out of 16 cores. Our evaluations show that the example 12-variable function on KA-CIM is computed in 45 ns with an energy consumption of 0.16 nJ per output sample. From Figure 5e, ≈35% of energy is spent on data communication (C), access (FI), and storage (ST), while the remaining 65% is consumed for computation (Tile + Σ). Future optimizations, such as 3D integration, could further reduce communication energy and increase the overall energy efficiency. The higher throughput is achieved by processing the subsequent layers in a pipelined manner. For instance, while layer 2 functions Φ_1 and Φ_2 are computed for the current data sample, layer 1 functions $\Psi_{p,q}$ are calculated for the next sample. For this example, KA-CIM achieves a throughput of 38.46×10^6 output samples per second. The chip-level instruction set for external control of KA-CIM is provided in Supporting Information Text 6, while the energy and latency of each individual block are reported in Table S3 of Supporting Information. Figure S14 of Supporting Information presents a detailed pipeline diagram of a similarly sized KAN from a real application used in our comparison section, illustrating a flow analogous to the example in Figure 5.

2.5 | KA-CIM System-Level Error, Energy, and Latency Analysis for Various Applications

This section presents the error, latency, and energy consumption of KA-CIM for computing various single-variable and multivariable functions converted to KAN form with PWL $N = 32$. The experiments include functions with large input ranges (0 to 400), large output dynamic ranges (e.g., $\exp(x) = 4.540 \cdot 10^{-5}$ to 7.389), dynamical systems (dn/dt), and complex four-variable trigonometric equations. The detailed equations, input/output ranges, and the required KAN conversions for all multivariable functions are provided in Supporting Information Text 10 and 11. As an example of a challenging dynamical system, we consider the Hodgkin-Huxley (HH) neuron model, whose gating variables (Equation (2)) evolve according to $n_t = dn + n_{t-1}$. In such cases, numerical errors due to approximation can accumulate over time, potentially causing output divergence. Figure 6 demonstrates that KA-CIM is resilient to such divergence, maintaining low errors centered around zero. The KA-CIM output overlaps the FP32 baseline output (without PWL) in all parts of the curve, and similar trends are observed for other tasks as well.

$$\frac{dn}{dt} = \alpha_n \cdot (1-n) - \beta_n \cdot n \qquad \alpha_n = n\left(\frac{0.01(10-V)}{\exp\left(\frac{10-V}{10}\right)-1}\right) \qquad \beta_n = 0.125\exp\left(\frac{-V}{80}\right) \tag{2}$$

Figure 7a presents KA-CIM error results across various benchmark examples. Across these diverse and complex tasks, the proposed accelerator consistently achieves low error relative to the FP32 baseline, with median errors ranging from 10^{-4} to 10^{-2} and slightly higher values for functions with a high output range. Notably, these error levels closely match those obtained using standard BFloat16 computation without PWL approximation, demonstrating that the proposed approach maintains high numerical fidelity while leveraging the PWL-based accelerator (see Table S5). The consistently low error across multiple equations highlights KA-CIM's flexibility in computing a wide range of KAN-converted multivariable functions. This demonstrates its effectiveness not only for scientific equation computation but also for KAN inference. For example, training of the relativistic addition function (Task 11: $\frac{u+v}{1+uv}$) produced a [2,1,1] KAN representation ($\tanh(\operatorname{atanh}(u) + \operatorname{atanh}(v))$), whose inference on KA-CIM achieved a low median error of 1.82×10^{-3}.

The error values can be further reduced by increasing the number of PWL segments. KA-CIM's tile partitioning mechanism ensures that this refinement incurs minimal energy overhead, even for higher values of N (Figure 3). Figure 7c illustrates the spatial localization of high-error values (above the 75th percentile) within the input space. The errors are clustered in specific regions, suggesting that increasing PWL segments in these areas can further decrease the error. Most of the high-error points are at least 3× lower than the maximum error, indicating that

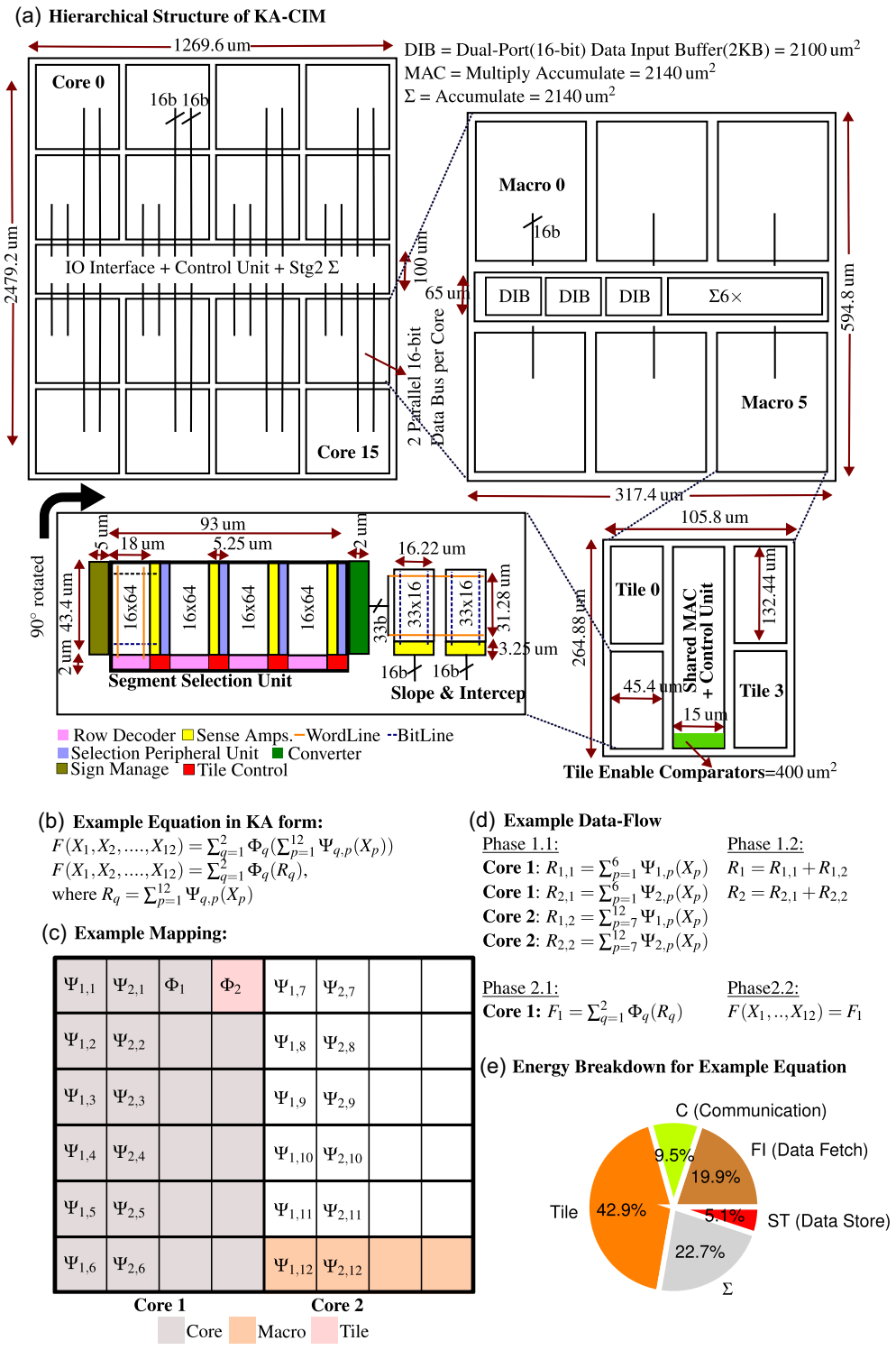


FIGURE 5 | KA-CIM accelerator area, mapping, and performance. (a) Area and floorplan of 1.875 Mb, 3.14 mm^2 KA-CIM designed using the proposed 1T1R array, RC-discharge based sensing, and BFloat16 data type. (b) An example of a multivariable function in KAN form. (c,d) Data mapping and data flow for the given example function. (e) Energy breakdown of KA-CIM, where Tile = Energy for computing a single variable nonlinear function (E.g., ϕ_q or $\Psi_{1,p}$) in a single tile, Σ = Energy for stage 1 and stage 2 summation unit, FI = Energy for fetching input x from one of the SRAM based dual-port input buffer (DIB), ST = Energy for storing the output in one of the DIB for subsequent KAN layer computation (E.g., Phase 2) or in data out buffer if this is the last layer, C = Communication energy. The total energy per data sample = 0.16 nJ, total latency per data sample = 45 ns, and throughput = 50×10^6 Output-Samples/s for the given equation $F(X_1,X_2, \ldots,X_{12}) = \sum_{q=1}^{2} \Phi_q(\sum_{p=1}^{12} \Psi_{q,p}(X_p))$.

extreme errors are rare and should not be the representative of overall performance. These observations highlight that the much lower median or 75th percentile error provides a more accurate reflection of KA-CIM's accuracy, and that targeted refinement in high-error regions can further enhance accuracy. It is important to note that the allocation of additional PWL segments is not

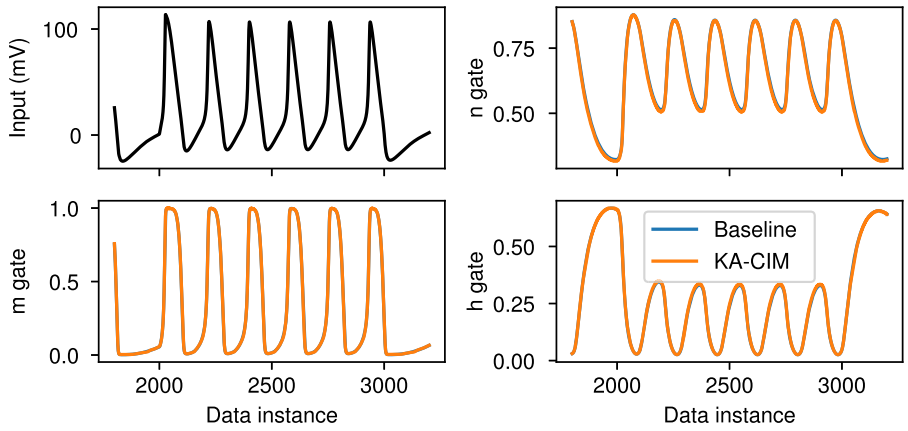


FIGURE 6 | Resilience of KA-CIM to output divergence in dynamic systems: three "gating" variables of Hodgkin–Huxley neuron model simulated on KA-CIM (32 PWL segments) vs baseline full precision (FP32 and no PWL approximation). Note: The blue Baseline curve overlaps with the orange curve in most parts of the graph, indicating the low error of KA-CIM.

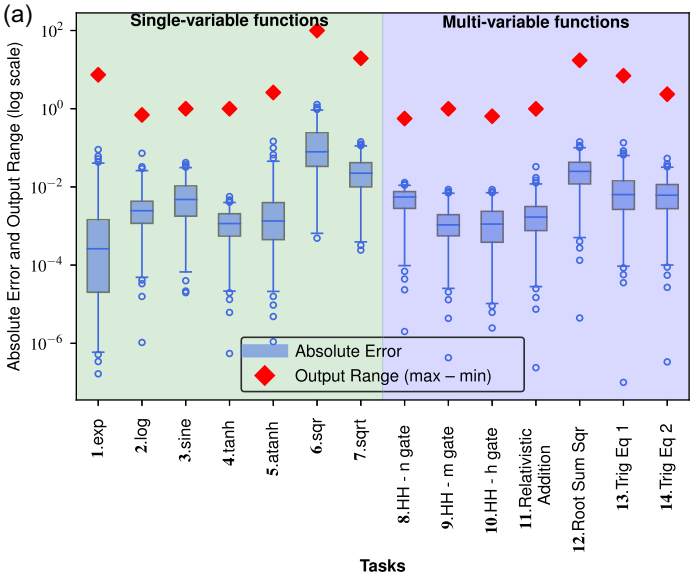
dynamic; rather, it is predefined at the software level during the PWL conversion of KAN functions.

Figure 7b shows the energy, latency, and energy–delay product (EDP) of KA-CIM across various tasks. A key feature of the proposed accelerator is that its latency and energy consumption depend solely on the number of KAN layers, rather than on the complexity or data range of the equation. For example, all single-variable functions exhibit a fixed latency of 15 ns and an energy consumption of 8.69 pJ. Similarly, Tasks 11 and 13, both implemented as two-layer KANs ([2,1,1]; see Supporting Information Text 11), consume the same energy and latency. KA-CIM maintains energy consumption below 60 pJ and latency below 60 ns even for the complex gating-variable equations of the Hodgkin–Huxley (HH) model. Compared with the custom ASIC Non-Linear Processing Unit (ASIC-NPU) of the SpiNNaker 2 system [28] and with a conventional CPU, KA-CIM demonstrates substantial advantages: it consumes up to 45× less energy than the ASIC-NPU and 315× less than the CPU. While KA-CIM operates 6.33× faster than the CPU, it is approximately 1.25×–3× slower than the ASIC-NPU. However, these latency differences are offset by KA-CIM's far lower energy consumption and its ability to support a wide range of nonlinear functions, whereas the ASIC-NPU is limited to exp and log. Both the CPU and ASIC-NPU in [28] were benchmarked for exp, making KA-CIM's single-variable results the fair point of comparison. Considering the architectural differences, we evaluate overall efficiency using the EDP and the energy–delay–area product (EDAP). KA-CIM achieves improvements of 1996× over the CPU and 6.13×–35.7× over the ASIC-NPU in terms of EDP and delivers a 4.4×–25.9× improvement over the ASIC-NPU in EDAP. These results highlight KA-CIM's architectural flexibility and its superior trade-offs compared with the fixed-function, ASIC-based ASIC-NPU. Detailed comparison tables are provided in Table S6 of Supporting Information. As a representative edge-scientific computing use case, we evaluated KA-CIM on the Kinematic Bicycle Model (Figure 7d) that is used for autonomous vehicle motion planning [51]. The energy and latency of KA-CIM remain exceptionally low, comparable to those observed in other tasks, and are significantly more efficient than conventional CPU and ASIC implementations. The current median error is approximately $\pm 5 \times 10^{-2}$ across the input variable range of a nominal vehicle, and it decreases to below $\pm 1 \times 10^{-2}$ at lower speeds. We anticipate that, with realistic datasets and deeper application-specific optimization, the KA-CIM parameters can be fine-tuned to achieve even lower error levels. It is important to note that the primary focus of this work is the development of a novel hardware accelerator for KAN inference with representative evaluations, rather than an exhaustive exploration of specific applications. We believe that such an accelerator can serve as a foundation for future research into energy-efficient scientific and edge computing application exploration.

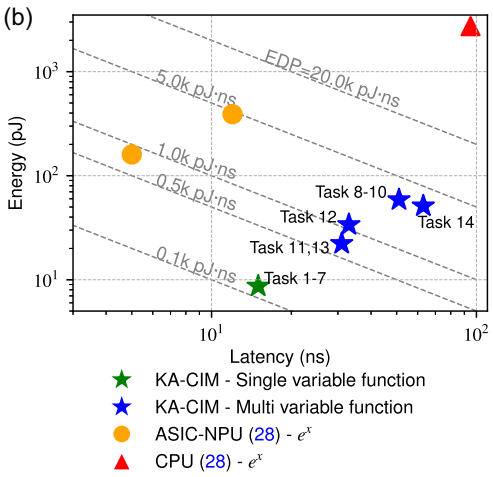
2.6 | Derivative Computation

Figure 8 summarizes the error, energy, and latency for computing the output of a given equation and its derivative value using KA-CIM. The operation flow begins by identifying the elementary functions $\phi_{i,j}$ (e.g. sqr, sin, exp) and converting $F(X)$ into KAN form, similar to the KAN conversions shown in Supporting Information Text 11 and 12. In the resulting KAN formulation, the derivative of each elementary function ($\phi'_{i,j}$) is placed in parallel to its respective $\phi_{i,j}$. Only the elementary $\phi'_{i,j}$ is preknown; the final $F'(X)$ solution remains unknown. This forms a KAN-like structure (Figure 8) whose data flow, akin to Automatic Differentiation [52, 53], is as follows:

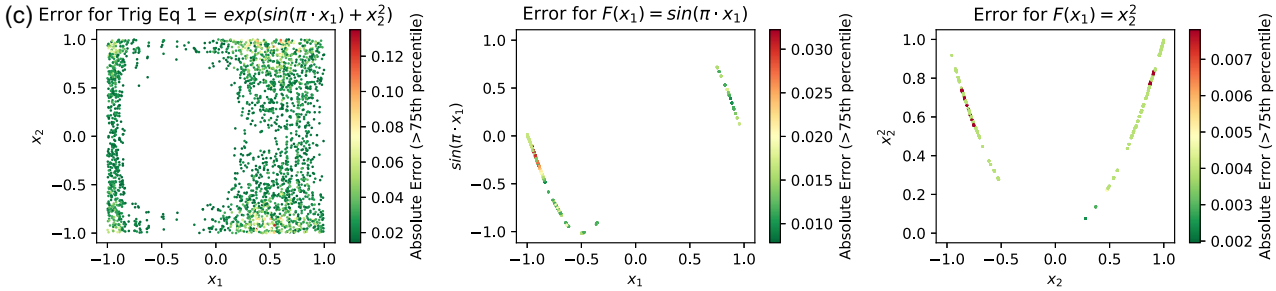
- The input to each stage is a dual number consisting of the *actual-value* (a) and the *derivative-value* (a').
- The $\phi_{i,j}$ and $\phi'_{i,j}$ are PWL converted and computed on the KA-CIM tile using the *actual-value* (a) as input.
- The derivative-output is multiplied by the previous stage's *derivative-value* to generate a new dual number. The Σ units available at various hierarchical stages are in fact MAC units that can be used for this operation.
- For the addition/subtraction operation in the equation, the *actual-values* (e.g. a and b) and *derivative-values* (e.g. a' and b') are added/subtracted as per chain-rule to form a new dual number: $((a + b),(a' + b'))$.
- For multiplication operations the output dual number = $((a \cdot b),(a' \cdot b + a \cdot b'))$.



KA-CIM error analysis for various tasks using 32 PWL segments (i.e., N = 32) without tile partitioning. The lower and upper whiskers represent the 0.01 and 0.99 quartiles, respectively.



Analysis of KA-CIM energy, latency, and energy-delay product (EDP) across various tasks in comparison to ASIC and CPU performance for executing e^x function. Contrary to KA-CIM's flexibility, the ASIC (28) variant is capable of computing only exp and log functions.



Localization of high-error values (i.e. all values above 75th percentile). Thus, highlighting areas where increasing PWL segments can significantly reduce the already low error. For *Trig Eq 1*, the 75th percentile, 99th percentile, and maximum errors are 1.44×10^{-2}, 6.36×10^{-2}, and 1.56×10^{-1}, respectively.

(d)

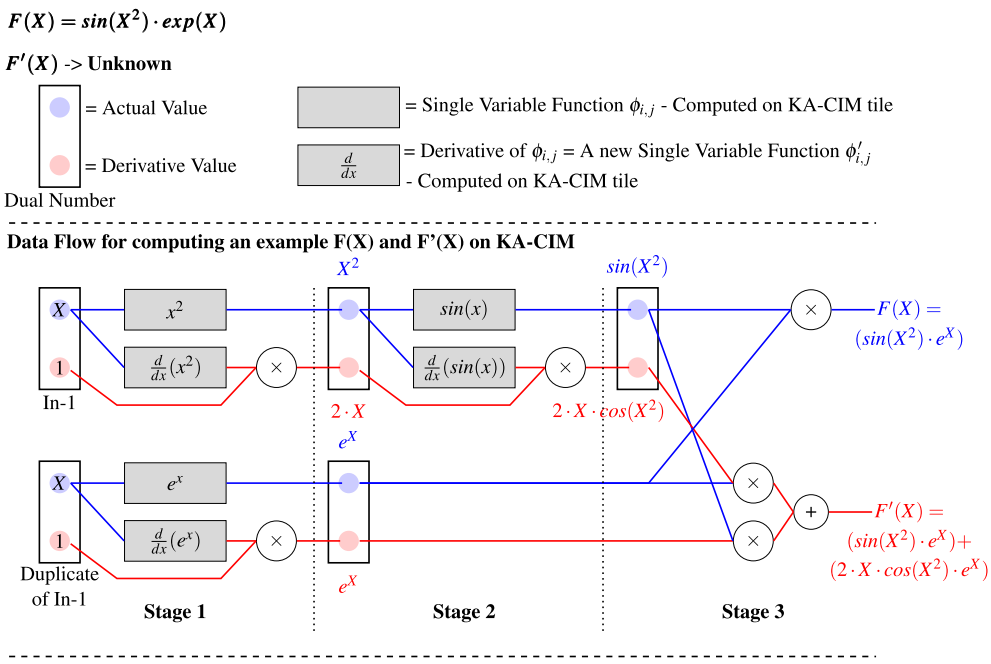
Example Edge Scientific Computation Use-Case: Kinematic Bicycle Model (51)					
Function - $F(V, \psi, u)$	Median Error		Lat. (in ns)	Eng. (in pJ)	Notes
	Std. Comp. w/ BFloat16	KA-CIM			Input Range
$\dot{X} = V \cdot cos(\psi + atan(0.5 \cdot tan(u)))$	1.47×10^{-2}	5.53×10^{-2}	49	39.53	V = 0 – 40m/s(144Km/h) ψ = Heading Angle = $\pm\pi$ rad
$\dot{Y} = V \cdot sin(\psi + atan(0.5 \cdot tan(u)))$	1.12×10^{-2}	4.07×10^{-2}			u = Wheel Angle = $\pm$0.8 rad

Evaluation results of KA-CIM on the Kinematic Bicycle Model, an example edge use-case for autonomous vehicle motion planning. This evaluation is based on randomly generated data to demonstrate KA-CIM's superior energy efficiency while maintaining low error. With realistic input data and application specific approximation, the error can be further reduced. The energy and latency results shown here correspond to the total energy consumption and latency of KA-CIM when simultaneously computing both $\dot{X}$ and $\dot{Y}$.

FIGURE 7 | KA-CIM error, energy, and latency evaluation. KA-CIM consistently exhibits low error while consuming ultralow energy (tens of pJ) and achieving low latency (tens of ns) for all tasks.

- For division operations, we convert $\frac{a}{b}$ to $e^{\ln(a) - \ln(b)}$. This eliminates the integration of divider units in KA-CIM.

The KA-CIM achieves low latency and energy consumption, computing both $F(X)$ and its derivative output samples in 43 ns (1 clk = 1 ns) and 53.04 pJ, respectively. This facilitates the efficient computation of algorithms that incorporate derivatives on data streams in real-time, energy-constrained environments such as edge computing. The reprogrammability and computational flexibility can further support diverse applications.



KA-CIM results for simultaneous computation of both $F(X)$ and $F'(X)$

Function	Median Error		Latency (in ns)	Energy (in pJ)	Notes
	Std. Comp. w/ BFloat16	KA-CIM			Input Range
$F(X) = sin(X^2) \cdot exp(X)$	7.03×10^{-4}	1.95×10^{-3}	43	53.04	-2 to 2
$F'(X) = \frac{d}{dX}(F(X))$	1.95×10^{-3}	5.46×10^{-3}			

FIGURE 8 | Derivative computation with KA-CIM Architecture. The blocks $\frac{d}{dx}(x^2)$, $\frac{d}{dx}(e^x)$, and $\frac{d}{dx}(\sin(x))$ are single variable funtions which equates to $2x$, e^x, and $\cos(x)$, respectively. The energy and latency results shown here correspond to the total energy and latency of KA-CIM for simultaneously computing both $F(X)$ and $F'(X)$.

2.7 | Comparison

This section compares KA-CIM with standard VMM-CIM and prior KAN accelerators. Since GPUs are primarily optimized for matrix multiplication, and VMM-CIM has been proven to deliver order-of-magnitude better efficiency than GPUs for MLP inference [5–9, 32], our comparison with VMM-CIM effectively subsumes the GPU baseline. Table 1 compares KA-CIM with a prior KAN accelerator [29] and with VMM-CIM. The VMM-CIM candidate represents a projected best-case analog design, operating at 100 TOPS/W with 8-bit precision and maximum parallelism. Here, we focus on two applications (i.e., Application 1 and Application 2). Application 1 involves predicting the signature of a knot that has 17 inputs and 14 outputs, where a [17, 1, 14] KAN-Inference model achieves a similar or better accuracy than an [17, 300, 300, 300, 14] MLP. Application 2 involves the computation of the equation $(F(x_1, x_2) = exp(-x_1) \cdot sinh(x_2))$, using a small [2,1] KAN, where the KA-CIM error for this task is $3.77 \times 10 - 2$. This application is not best-suited for MLP or DNN, especially on VMM-CIM with a fixed-point 8-bit format. However, to give the benefit of the doubt, we consider an 8-bit [2, 64, 64, 64, 1] MLP as a VMM-CIM candidate. Here, we include an MLP/DNN baseline to show that high energy efficiency is not always achieved through on VMM-CIM architectures. Novel architectures like KA-CIM, optimized for KAN, can achieve superior energy efficiency in scientific and equation-driven tasks, where MLPs and DNNs have not yet established themselves as the standard benchmark. We first discuss KA-CIM vs VMM-CIM, followed by prior KAN accelerators.

KA-CIM computing KAN exhibits clear advantages in latency, energy consumption, and throughput across both applications in comparison to MLP computed on VMM-CIM. The energy-delay product shows that the KA-CIM is two orders better than VMM-CIM for both applications. These improvements stem from KAN's ability to compress large MLPs into smaller networks–for example, reducing an MLP requiring 189, 300 MAC operations into a KAN with only 31 nonlinear functions. KA-CIM further enhances efficiency by approximating these 31 functions with PWL models, thereby converting them into just 31 MAC operations. The KA-CIM's CIM-SSU block contributes additional energy savings by enabling PWL comparisons through a single memory read, requiring only a simple 1-bit sensing circuit instead of a power-hungry ADC. Finally, KA-CIM's optimized compact array design for reduced read energy and latency leads to substantial overall system-level improvements. While KA-CIM has demonstrated advantages in such applications, its performance on conventional datasets (e.g., image, voice, and text) is beyond the scope of this work. For these datasets, VMM-CIM accelerators have already demonstrated orders of magnitude higher energy efficiency compared to CPU/GPUs [32].

A prior KAN accelerator, ASP-KAN-HAQ [29], reported results for a hardware that was specifically designed for a KAN model of size

TABLE 1 | Comparison of KAN-CIM vs state-of-the-art KAN accelerator and conventional-CIM for scientific applications. For fair comparison with ASP-KAN-HAQ [29], which is explicitly designed for KAN [17,1,14], KA-CIM and VMM-CIM are also explicitly designed for the given network of application 1.

Application 1: Predicting the signature of a knot [14] - Inference					
	KA-CIM (This work)		**ASP-KAN-HAQ [29]**		**VMM-CIM (100 TOPS/W[a])**
Network	KAN = [17,1,14]		KAN = [17,1,14]		[17,300,300,300,14]
Total num operations	31 Functions + Sum		31 Functions + Sum		189 300 MACs
Data type	BFloat (16-bit)		Fixed-Point (8-bit)		Fixed-Point (8-bit)
Memory cell type	1T1R		1T1R		1T1R
Variant	Low Area	Medium Area	Low (KAN1)	Medium (KAN2)	—
Training parm (G,K)	(10,3)	(60,3)	(7–10,–)	(30–60,–)	NA
Num. PWL segment	16	32	NA	NA	NA
HW accuracy (%)	81.6	83.93	81.03	86.74	—
(GPU/CPU accuracy)	(81.9)	(85.9)	—	—	(78 [29])
Retraining	No	No	Yes	Yes	No
Energy per 14 output samples (pJ)	**135.79**	**152.4**	257.13	392.76	3786
Latency (ns)	**30**	**30**	664	832	224
Area (mm^2)	0.176	0.25	0.0143	0.063	—
EDP (pJ · ns)	4.07×10^3	4.57×10^3	1.71×10^5	3.27×10^5	8.48×10^5
EDAP (pJ · ns · mm^2)	$\mathbf{0.71 \times 10^3}$	$\mathbf{1.14 \times 10^3}$	2.44×10^3	2.06×10^4	—
Total memory elements	4.238 K	9.6875 K	0.27 K	2.17 K	184.86 K

Application 2: Trigonometry Equation: $F(x_1, x2) = \exp(-x_1) \cdot \sinh(x_2)$		
	KA-CIM (This work)	**MLP-CIM - 100 TOPS/W**
Network	KAN = [2,1]	MLP = [2,64,64,64,1]
Data type	BFloat (16-bit)	Fixed-Point (8-bit)
Total num operations	2 Functions + Mul	8384 MACs
Energy per 2 output samp (pJ)	14.04	167.68
Latency (ns)	15	224
Output samples/s	66.66 M	17.85 M
Energy delay product (pJ · ns)	$\mathbf{2.1 \times 10^2}$	3.76×10^4
Total memory size	1.25 KB	8.1875 KB

[a]For VMM-CIM, we consider the following assumptions:. 1) Eight-bit data width. 2) All the MACs of a single MLP layer are computed in parallel and require eight iterations to output the final MAC result. 3) Latency for a single MLP layer = 8 × 1_Iteration_Lat = 8 × 7 ns = 56 ns. 4) 1_*Iteration_Lat* = *CrossbarRead* + *ADC* + *Accumulation* = 5 ns + 1 ns + 1 ns = 7 ns. The read latency in the VMM-CIM case is 5 ns due to larger array size requirements, which is not the case for KA-CIM.

[17,1,14], consisting of 31 nonlinear functions. It introduced two hardware variants, referred to as KAN1 (low area or high performance) and KAN2 (medium area or high accuracy). In contrast, our proposed KA-CIM accelerator (shown in Figure 5) is a multi-core architecture capable of flexibly supporting KAN models with arbitrary dimensions and up to 384 nonlinear functions. To enable a direct comparison with ASP-KAN-HAQ [29], we re-evaluate KA-CIM, explicitly designed for the [17,1,14] KAN with 31 functions. As with ASP-KAN-HAQ, we also report results for two design variants that are directly comparable to KAN1 and KAN2 of [29], i.e., low-area (high-performance) and a medium-area (high-accuracy) implementation (refer to Figure S15 and S16 of Supporting Information). These re-evaluated results are applicable only for Application 1. As shown in Table 1, KA-CIM achieves superior energy efficiency (1.89 × –2.57 ×) and significantly lower latency (22 × –27 ×) compared to ASP-KAN-HAQ [29]. These gains are attributed to KA-CIM's ability to compute single-variable functions using a single MAC operation, in contrast to the multi-MAC approach used in [29]. Furthermore, our architecture inherently enforces small array dimensions, which in turn enable optimized array design and effectively reduce overall energy and latency. To enable a comprehensive architectural efficiency comparison, we evaluate both designs using the energy-delay product (EDP) and the energy-delay-area product (EDAP). KA-CIM outperforms ASP-KAN-HAQ with 42 × –71× improvement in EDP and 3.4 × –18× improvement in EDAP. Our low-area KA-CIM variant achieves the same inference accuracy as prior work, while the medium-area variant shows a modest 2% reduction compared to GPU/CPU. However, it is important to note that our inference accuracy is obtained without any hardware-specific retraining,

unlike ASP-KAN-HAQ, which relies on retraining for accuracy recovery. With retraining, KA-CIM's accuracy can be further improved, effectively compensating for this loss. It is important to note that as the G value increases, the memory capacity gap decreases, narrowing to approximately 5×, while the EDAP advantage increases, reaching up to 18×. This trend clearly demonstrates that for realistic, high-complexity scenarios requiring larger G values, KA-CIM significantly outperforms [29], both in efficiency and scalability. The choice of $G = 60$ for our medium-area variant is intended solely to demonstrate that a 32-segment PWL approximation can effectively represent complex functions trained using higher G values and does not imply that our work requires a higher G value. The same energy, area, and latency results hold true for other G values as well, provided the number of PWL segments is the same. Furthermore, with advances in 3D integration and continued scaling of memory technologies, our fully memory-centric KA-CIM architecture is well positioned to further improve in area efficiency and scaling to larger networks.

Compared to a Stochastic Computing-based KAN accelerator [30], KA-CIM achieves 2.15× lower power consumption (0.73 mW vs. 1.57 mW) and 2× better accuracy (mean error of 1.09×10^{-2} vs 2.1×10^{-2}) for computing $\exp(\sin(\pi \cdot x_1) + x_2^2)$. SMURF [54] is another stochastic-based hardware architecture suitable for KAN acceleration due to its ability to flexibly compute arbitrary nonlinear functions. Compared to SMURF, KA-CIM achieves significantly lower latency. Computing the single variable tan h function in KAN-CIM requires 15 cycles (or 15 ns) and has a mean error of 1.09×10^{-2}, whereas SMURF requires 256 cycles to attain a similar error. At a reduced latency of 64 cycles, SMURF's mean error increases to 3.7×10^{-2}. This highlights a fundamental limitation of stochastic methods, i.e., accuracy is directly dependent on bit-stream lengths that increase the latency. Recently, photonic device–based KAN accelerators have also been proposed [31]. Such designs report a high power consumption of 2–3 W for a 500-parameter KAN model, whereas KA-CIM achieves a significantly lower power consumption of 5.08 mW for a larger 2.17 K parameter KAN model that generate 17 output sample in parallel (i.e. Application 1). Nevertheless, a direct comparison between these two fundamentally different device technologies would not be entirely fair.

NN_LUT [55] is a LUT-based universal nonlinear function approximator using PWL, similar to KA-CIM, which can also be adapted for KAN acceleration. Unlike the memory-centric, thermometer and one-hot encoded KA-CIM tile, NN_LUT relies on standard digital logic (Figure S19 of Supporting Information) and uses binary encoded breakpoints. To an extent, NN_LUT serves as KA-CIM's equivalent digital-logic counterpart. KA-CIM's design results in superior energy (1.29 pJ vs. 4.118 pJ) and a smaller area footprint (6,012.77 μm^2 vs. 9000 μm^2) by replacing 32× 16-bit digital comparators and PWL breakpoint registers with a specialized CIM-SSU. Note that this comparison does not account for the area and energy consumption associated with MAC, as it is identical for both architectures in the context of PWL approximation. Furthermore, KAN accelerators typically require parallel computation of multiple single-variable functions that increase the memory demands for storing PWL parameters (breakpoints, slopes, and intercepts). In this regard, KA-CIM's memory-centric architecture offers a more efficient solution than traditional CMOS logic-based implementations like NN_LUT, which require frequent transfer of PWL parameters to be moved between the on-chip memory and comparators.

The prior work [56–59] proposed Logic-in-Memory architectures for flexible computation but required breaking computations down to logic gate level, making them inefficient for complex nonlinear tasks. As per [59], an 8-bit arithmetic and logic operation consumes energy of 38.22 pJ and a latency of 1397 cycles. Compared to these results, KA-CIM consumes an energy of 33.85 pJ and a latency of 33 clock cycles (or 33 ns) for a 16-bit Root-Sum-Square computation $\left(\sqrt{x_1^2 + x_2^2 + x_3^2 + x_4^2}\right)$. A few state-of-the-art methods, such as CompressedLUT [60], Flopoco [61], and Unary [62], proposed the design approaches that generate highly optimized hardware for a given nonlinear function. These approaches are well-suited for fixed-function implementations but lack the flexibility required for KAN hardware accelerators. Since KA-CIM is designed for the flexible computation of any nonlinear function, comparing it with function-specific optimized implementation obtained from these methods would be unfair.

3 | Discussion

This work introduces KA-CIM, a memory-centric accelerator that demonstrates how cross-layer codesign—spanning algorithm, architecture, circuits, and devices—can enable efficient computation of Kolmogorov–Arnold Networks (KANs). By exploiting PWL approximations, we reduce arbitrary nonlinear functions to single multiply–accumulate (MAC) operations, achieving predictable latency and energy independent of function complexity. Compact 1T1R arrays (16×64 and 32×16), fabricated with HfO_2-based memristive devices, are co-optimized with an RC-discharge sensing scheme, yielding fast reads (1.5 ns) at low energy (4.2 fJ/bit). We demonstrated KA-CIM's use cases beyond KAN inference for multivariable equation solving and derivative computations, illustrated across several example tasks. Across these, KA-CIM consistently operated within tens of picojoules of energy and tens of nanoseconds of latency. It further outperformed existing CPUs, ASICs, VMM-CIMs, and prior KAN accelerators in terms of both energy–delay product and energy–delay–area product. These results can enable broader KAN deployment for energy-efficient scientific computing and AI + Science applications.

We note the potential to develop hybrid chips integrating two complementary energy-efficient blocks: VMM-CIM and KA-CIM. While KA-CIM is tailored for nonlinear workloads, VMM-CIM excels at matrix operations required by MLPs and DNNs. A hybrid architecture combining these two paradigms could substantially broaden the scope of energy-efficient computing beyond what either can achieve in isolation. Several application domains stand to benefit from such a design. In chip design automation, KA-CIM could accelerate equation-defined circuit evaluations, while VMM-CIM supports DNN-driven layout optimization and mapping. In scientific computing, KA-CIM offers efficient computation of physics- and chemistry-based equations, while VMM-CIM extracts insights from the resulting data streams through conventional AI models. Similarly, in climate modeling, financial analysis, edge computing, and other domains, such a hybrid architecture could simultaneously manage large-scale data-driven workloads and complex equation-driven computations. Beyond such coarse-grained applications, hybrid designs may inspire AI models that

interleave conventional MLP layers with mathematical equation-based KAN layers. Such models are speculated to reduce network depth and width, improve task adaptability, and enable compact multitask networks with minimal energy overhead. By efficiently mapping these hybrid models, VMM-CIM and KA-CIM together could define a new class of accelerators that unify data-driven and equation-driven computing. Lastly, the architecture can be extended to KAN training, where a KA-CIM tile, in combination with a digital-logic- or VMM-CIM-based gradient descent computation unit, directly updates the PWL slope and intercept. For such a configuration, we propose using a volatile memory instead of the RRAM array used here, but this will be quantified and explored in future work.

4 | Materials and Methods

4.1 | Memristor Fabrication

The investigated memristive devices were fabricated on a 180 nm technology chip provided by X-FAB semiconductor foundries GmbH with a CMOS compatible back end of line process. This 180 nm technology is only used for memristor device fabrication and characterization. All other evaluations of the KA-CIM accelerator are using 28 nm technology. The fabrication of the 180 nm CMOS chip was stopped at metal layer 4 with tungsten vias connected to the transistors. The process flow starts with contact pads of 5 nm Ta and 50 nm Pt obtained by DC sputtering from the respective metal targets at room temperature, patterning by UV lithography and structuring by reactive ion beam etching (RIBE) with argon at room temperature. The 100×100 nm sized filamentary VCM-type $Pt/HfO_2/TiO_x/Ti/Pt$ devices were manufactured between the contact pads using stacks built from metal layers and metal oxide layers patterned by electron beam lithography and structured with a room temperature RIBE process using Argon. The metal layers, consisting of 5 nm Ta and 25 nm Pt for the bottom electrode and 10 nm Ti and 20 nm Pt for the top electrode, were deposited by DC sputtering at room temperature. The 3 nm HfO_2 and 3 nm TiO_x layers were deposited by atomic layer deposition (ALD) in a FlexAl system of Oxford Plasma Technologies at a table temperature of 300°C [37, 63]. For HfO_2, tetrakis(ethylmethylamino)hafnium (TEMAH) and oxygen plasma were used, and the TiO_x layer was deposited from tetrakis(dimethylamino)titanium (TDMAT) and water vapor. The semiconductor-grade metal precursors were purchased from Dockweiler Chemicals GmbH. The device fabrication was finalized by metal lines (50 nm Pt), sputtered and structured by lift-off, which connect the BE and TE with the respective contact pad of the underlying CMOS. The resulting 1T1R structure has the active electrode (Pt) of the VCM cell connected to the source of the NMOS transistor of the size $W/L = 1$.

4.2 | Memristive Device Measurements

The electrical characterizations of the 1T1R structures were performed using a Keithley 4200A SCS equipped with two channel pulse measure units (4225-PMUs) and four remote amplifiers (4225-RPMs). To enable a good control of the voltage applied to the VCM device for electroforming and set programming, the active electrode (Pt BE) is connected with the transistor's source side (see ref. [64]). For the set process, a positive voltage V_{SL} is applied via the SL while the BL is grounded ($V_{BL} = 0\,V$). The maximum current is defined by the transistor's gate current controlled by the voltage on the WL. For the reset process, a positive voltage is applied to the BL while keeping the SL grounded ($V_{SL} = 0V$). During reset, the transistor is fully opened. The electroforming process was carried out with a triangular voltage signal with a sweep rate of 1 kV/s and a current compliance of 150 μA defined by the transistor's gate voltage. After forming, about a hundred switching cycles were performed to ensure proper device functionality. For further discussion, the voltage polarity is defined by its sign. Hence, the operation parameters to achieve the desired LRS and HRS values of 10 kΩ and 3 MΩ, respectively, were as follows: a forming voltage of $V_{FORM,\,max} = -3.3\,V$, a SET voltage of $V_{SET,\,max} = -2.0\,V$ at $V_{WL} = 2.8\,V$, a RESET voltage of $V_{RESET,\,max} = 2.4\,V$ at $V_{WL} = 5\,V$, and a READ bias of V_{READ} between 0.3 V and 0.4 V. We first verified that the deeper HRS can be reached across all devices. For this, we performed one-shot programming on 18 devices, all of which successfully reached the target 3MΩ HRS, suggesting that using a read–verify scheme would reliably achieve the desired state. We then selected one of the worst-performing devices for long-term resistance stability analysis. The rest of the device measuments were based on a read-and-verify programming scheme consisting of set, reset, and read pulses that gradually pulses the cell into the desired resistance state. After each set and reset pulse, a read is done to check the resistance value. After the intended measurement, three switching sweeps are performed in order to prepare the cell for the following read and verify algorithm and check if the device is working. The read drift measurements were performed by applying a 0.4 V read pulse for 1 s to the device every 1000s for a total number of 10 repetitions. To further stress the resistance state, a read disturb experiment was performed. A 500 s long read pulse is applied on the 1T1R structure while recording the current values every one millisecond. In both cases, a small tilt of the LRS distribution was observed, while the tilt of the HRS was much stronger. This tilt of HRS value to 550 kΩ and LRS to 15 kΩ is within the limits of the reliable sensing of the data by the RC discharge-based sensing circuit. Please refer to Supporting Information Text 3 for more details. For all circuit- and system-level simulations, we used the worst-case upper-limit LRS and lower-limit HRS from the worst performing device, thereby accounting for both device-to-device variation and long-term variation. This ensures that the resulting errors remain within the reported range to support the high-precision claims of the manuscript.

4.3 | T1R Array and Sensing Circuit

4.3.1 | Area

All evaluations of KA-CIM were carried out using TSMC 28 nm CMOS technology. To estimate the area of a 1T1R cell in this technology, we implemented a 2×2 cells using 1.8 V NMOS devices in Cadence design tools. The area estimation is based on the footprint of the access transistor, over which a 100×100 nm VCM device can be integrated. We assume that the VCM device is integrated between Metal 5 and Metal 6. This cell design was then extended to the full array size (16×64 and 32×16) to evaluate all the array area within a tile. Post-layout parasitic extraction was performed to obtain the bitline and wordline capacitances, which are essential for latency and energy estimation. The RC discharge–based sensing circuit was also implemented at the layout level in Cadence for

area estimation. The sense amplifier layout was matched to the width of the 1T1R cell to ensure integration feasibility.

4.3.2 | Energy and Latency

This work focuses on read energy and latency, as KA-CIM is read-dominant and writes are performed only once during initialization. Consequently, write energy and latency have negligible amortized impact, whereas read operations dominate both system performance and energy consumption. This simplification allows for a more accurate estimation of array-level read performance without introducing the complexity of write modeling. This approach is consistent with the methodology in other KAN accelerators [29, 30] and many VMM-CIM architecture studies focusing on DNN inference. A detailed netlist of both the array and the sensing circuit was implemented in Cadence Virtuoso. The memory cell was constructed using 1.8 V NMOS access transistors from a 28 nm PDK and passive resistors configured to represent suitable LRS and HRS values (including their variation). The LRS and HRS values and its variations are based on post-silicon VCM device characterization. Passive resistors were used in place of a compact memristor model, as both exhibit equivalent behavior during read operations for a given resistance. The array circuit simulations included post-layout extracted wire parasitics and internally considered transistor parasitics to ensure accurate latency and energy estimation. Device-level variations were also considered: CMOS variations captured through Monte Carlo simulations and cell resistance variation derived from measured read distributions. The wire capacitance was estimated to be 1.7 fF/12.5 µm. In our evaluations, the capacitance of the RC-discharge sensing path includes not only wire parasitics but also all cell-transistor parasitics. Our estimations show a worst-case read latency of 2.5 ns (excluding precharge, 1.5 ns) and energy of 4.2 fJ/bit (including precharge).

Although write energy/latency was not estimated in this work, we evaluated that the proposed 1T1R cell design with a 1.8 V, 28 nm NMOS access transistor can meet the programming voltage and current compliance requirements of the memristor device. Details are provided in Figure S11 and Table S2 of Supporting Information. The objective was not to reproduce the full write operation, but rather to verify that the voltage drop across the memristor can reach the required high levels without exceeding NMOS device limits. Furthermore, we also verified that the current compliance is consistent with post-silicon measurements from the 180 nm test chip during critical phases such as HRS-to-LRS and LRS-to-HRS transitions. To this end, passive resistors were used in place of a memristor model, as our evaluation was limited to ensuring set/reset voltage and current feasibility with compact transistors rather than estimating write latency or energy. We also consider that in the future, we can fabricate forming-free VCM devices. Hence, eliminating the need for 3.3 V operation. Furthermore, motivated by the increasing interest from technology providers in supporting emerging memory IPs, we reasonably assume that advanced CMOS nodes will provide suitable, low-area, memory device-specific transistors in the future.

4.4 | KA-CIM Tile

The array and sensing circuit IP blocks were integrated to construct the KA-CIM tile, including the CIM-SSU, slope array, and intercept array. The remaining digital peripheral circuits, such as the row decoder, row driver, selection logic, and data converters, were implemented using standard logic gates from the technology library. The post-layout area of both the array and the digital logic was used to estimate the tile area. The complete tile design, including control circuitry, was implemented in Cadence Virtuoso for energy and latency simulations. To assess the impact of process variations on circuit behavior, Monte Carlo simulations were performed. The read timing skew between arrays and between the array bitlines is avoided by enabling the subsequent selection peripheral logic after the worst-case read latency using flip-flop based half clock and quarter clock control circuitry.

4.5 | Tile Enable Comparator, MAC Unit, Σ Units, Data Buffers, and Data Communication

The area, energy, and latency of the tile-enable comparators were obtained from SystemVerilog implementations synthesized to layout using Synopsys tools, including Design Compiler, IC Compiler, and PrimePower. The BFloat16 digital MAC unit metrics (area, energy, and latency) were taken from a prior work in 28 nm CMOS technology [65]. The Σ6× and Stage-2 Σ units were assumed to have identical characteristics to the MAC unit, reflecting architectural scalability to KAN 2.0. The dual-port data input buffer (DIB) and output data buffer, both implemented using SRAM, were characterized based on values reported in prior VMM-CIM literature [8]. Communication energy was estimated by scaling the reported 6 pJ for 64-bit transmission over a 1 mm bus [66, 67] to match our bus length and width. This work draws on post-silicon measurement data previously reported in [66, 67].

4.6 | Overall

The area of all units is summarized in Figure 5, and the corresponding energy and latency metrics are presented in Table S3 of Supporting Information.

4.6.1 | Performance Analysis

A Python model of KA-CIM was developed to perform hardware evaluations for various tasks. This model is configured with individual circuit worstcase energy and latency listed in Table S3 of Supporting Information, programmed with PWL-converted single variable functions, and is evaluated with a large stream of input data samples to obtain the final KA-CIM error, energy, and latency for any given KAN. All latency results presented in this work assume a 1 ns clock cycle, but the actual circuit latency can be longer than 1 ns (i.e., multiple clock cycles). For example, the tile latency presented in Table S3 is 4 ns, which corresponds to 4 cycles that include all tile array read and precharge latency. The conversion of tasks/equations corresponding to Figure 7a and Table S5 to the KAN form is exact and is done manually (Supporting Information Text 11). By doing so, we can focus solely on the errors induced by PWL and KA-CIM architecture, while eliminating training-related loss. The KAN for demonstrating KA-CIM results in Table 1 is extracted from training the dataset provided in [14]. The conversion of single-variable functions of KAN to the PWL form is accomplished using the PWLF Python library [68]. Please note that the functions to be PWL approximated need not be an interpretable or known functions but can be any arbitrary

function represented as a two-column dataset (i.e. list of input and output values). All the results in this paper are with $N = 32$, demonstrating that 32 PWL segments suffice for most cases. The PWL breakpoints, slope, and intercept values obtained from PWLF library are initially represented in FP32 and quantized to BFloat16 before being programmed into KA-CIM. Since BFloat16 retains the same sign and exponent width as FP32 but has a reduced mantissa (7-bit vs. 23-bit), this quantization step simply discards the least significant 16 bits. The conversion is performed using the Python PyTorch library.

The GPU latency evaluations of KAN and MLP layers in Figure 1 were conducted on NVIDIA A100. The MLP layer was implemented using the PyTorch Python library [69], while the B-spline methodology was used to implement the KAN layers (as suggested in [14]). Latency measurements were recorded using PyTorch's profiling tool, which provides detailed duration data for all CPU and GPU operations. A batch size of 1000 was chosen for both the MLP and KAN layers to ensure efficient GPU utilization.

Acknowledgments

This work was partially funded by the Federal Ministry of Education and Research (BMBF, Germany) in the project NEUROTEC-II (project number: 16ME0398K) and by Deutsche Forschungsgemeinschaft within TRR 404 Active-3D (project number: 528378584). O.A. and S.H.E. thank D. Storelli, G. Potemkin, S. Shoja, M. Gerst, M. Deckers, S. Wiefels (all PGI-7), B. Bennemann (PGI-9), F. Cüppers (PGI-10), S. Trellenkamp, F. Lentz, T. Wessels (HNF), and D. Nielinger (PGI-4) for technical support and discussions.

Open Access funding enabled and organized by Projekt DEAL.

Funding

This study was supported by Deutsche Forschungsgemeinschaft (528378584) and Bundesministerium für Forschung und Technologie (16ME0398K).

Ethics Statement

The authors have nothing to report.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

The error, energy, and latency data generated in this study and used in the figures and plots within this manuscript have been deposited in the public database repository under https://iffgit.fz-juelich.de/sudarshan/ka-cim. The raw memristive measurements will be made available upon reasonable request.

Code Availability

The codes that support the findings of this study have been deposited in the public database repository under https://iffgit.fz-juelich.de/sudarshan/ka-cim.

References

1. K. Bourzac, "Fixing AI's Energy Crisis," accessed November 19, 2024, https://www.nature.com/articles/d41586-024-03408-z.

2. Decadal Plan for Semiconductors, *Technical Reports* (Semiconductor Research Corporation, 2021).

3. K. Guo, W. Li, K. Zhong, et al., "Neural Network Accelerator Comparison," accessed September 30, 2024, https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator.html.

4. N. P. Jouppi, D. Hyun Yoon, M. Ashcraft, et al., "Ten Lessons From Three Generations Shaped Google's TPUv4i: Industrial Product," In *2021 ACM/IEEE. 48th Annual International Symposium on Computer Architecture (ISCA* (IEEE, 2021), 1–14, https://doi.org/10.1109/ISCA52012.2021.00010.

5. F. Aguirre, A. Sebastian, M. Le Galo, et al., "Hardware Implementation of Memristor-Based Artificial Neural Networks," *Nature Communications*, 15 (2024): 1974, https://doi.org/10.1038/s41467-024-45670-9.

6. A. Sebastian, M. Le Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, "Memory Devices and Applications for in-Memory Computing," *Nature Nanotechnology* 15 (2020): 529–544, https://doi.org/10.1038/s41565-020-0655-z.

7. D. Ielmini and H.-S. P. Wong, "In-Memory Computing With Resistive Switching Devices," *Nature Electronics* 1 (2018): 333–343, https://doi.org/10.1038/s41928-018-0092-2.

8. A. Shafiee, A. Nag, N. Muralimanohar, et al., "ISAAC: A Convolutional Neural Network Accelerator With In-Situ Analog Arithmetic in Crossbars," *2016 ACM/IEEE. 43rd Annual International Symposium on Computer Architecture (ISCA)*, (IEEE, 2016), 14–26, https://doi.org/10.1109/ISCA.2016.12.

9. A. Ankit, I. El Hajj, S. R. Chalamalasetti, et al., "PUMA: A Programmable Ultra-Efficient Memristor-Based Accelerator for Machine Learning Inference," In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems, ASPLOS '19, (Association for Computing Machinery, 2019), 715–731, https://doi.org/10.1145/3297858.3304049.

10. M.-K. Kim, I.-J. Kim, and J.-S. Lee, "CMOS-Compatible Compute-in-Memory Accelerators Based on Integrated Ferroelectric Synaptic Arrays for Convolution Neural Networks," *Science Advances* 8 (2022): eabm8537, https://doi.org/10.1126/sciadv.abm8537.

11. Z. Xiao, V. B. Naik, J. H. Lim, Y. Hou, Z. Wang, and Q. Shao, "Adapting Magnetoresistive Memory Devices for Accurate and on-Chip-Training-Free in-Memory Computing," *Science Advances* 10 (2024): eadp3710, https://doi.org/10.1126/sciadv.adp3710.

12. K. Simonyan and A. Zisserman, Very Deep Convolutional Networks for Large-Scale Image Recognition. (2015), https://doi.org/10.48550/arXiv.1409.1556.

13. K. He, X. Zhang, S. Ren, and J. Sun, 2015, "Deep Residual Learning for Image Recognition, 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)," https://doi.org/10.1109/CVPR.2016.90.

14. Z. Liu, Y. Wang, S. Vaidya, et al.,2025 "KAN: Kolmogorov–Arnold Networks," In The Thirteenth International Conference on Learning Representations, https://openreview.net/forum?id=Ozo7qJ5vZi.

15. Z. Liu, P. Ma, Y. Wang, W. Matusik, and M. Tegmark, "Kolmogorov-Arnold Networks Meet Science," *Physical Review*. X 15 (2025): 041051, https://doi.org/10.1103/4t7t-v19l.

16. A. K. Kolmogorov, "On the Representation of Continuous Functions of Several Variables by Superposition of Continuous Functions of One Variable and Addition," *Doklady Akademii Nauk SSSR* 114 (1957): 369–373.

17. C. Li, X. Liu, W. Li, et al., 2024,. "U-KAN makes strong backbone for medical image segmentation and generation," In Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty- Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence," AAAI'25/IAAI'25/EAAI'25, (AAAI Press, 2025), https://doi.org/10.1609/aaai.v39i5.32491.

18. X. Han, X. Zhang, Y. Wu, Z. Zhang, and Z. Wu, *KAN4TSF: Are KAN and KAN-Based Models Effective for Time Series Forecasting?*, 2024, https://doi.org/10.48550/arXiv.2408.11306.

19. N. Ranasinghe, Y. Xia, S. Seneviratne, and S. Halgamuge, *GINN-KAN: Interpretability Pipelining With Applications in Physics Informed Neural Networks*, arXiv preprint arXiv:2408.14780, 2024, https://doi.org/10.48550/arXiv.2408.14780.

20. E. Zeydan, C. J. V. Rubio, L. Blanco, et al., R. Pereira, M. Caus, and A. Aydeger, "*F-KANs: Federated Kolmogorov-Arnold Networks, 2025 IEEE 22nd Consumer Communications & Networking Conference (CCNC)*," https://doi.org/10.1109/CCNC54725.2025.10976205.

21. L. Li, Y. Zhang, G. Wang, and K. Xia, "Kolmogorov–Arnold Graph Neural Networks for Molecular Property Prediction," *Nature Machine Intelligence* 7 (2025): 1346–1354, https://doi.org/10.1038/s42256-025-01087-7.

22. A. A. Howard, B. Jacob, and P. Stinis "Multifidelity Kolmogorov–Arnold networks," *Machine Learning: Science and Technology* 6 (2025): 035038, https://doi.org/10.1088/2632-2153/adf702.

23. A. Ning, M. Xue, J. He, and C. Song, *"KAN See in the Dark," In IEEE Signal Processing Letters*, vol. 32 (2025), pp. 891-895, https://doi.org/10.1109/LSP.2025.3540700.

24. Y. Chen, Z. Zhu, S. Zhu, et al., "SCKansformer: Fine-Grained Classification of Bone Marrow Cells via Kansformer Backbone and Hierarchical Attention Mechanisms," *IEEE Journal of Biomedical and Health Informatics* 29 (2025): 558-571, https://doi.org/10.1109/JBHI.2024.3471928.

25. D. W. Abueidda, P. Pantidis, and M. E. Mobasher, "DeepOKAN: Deep Operator Network Based on Kolmogorov Arnold Networks for Mechanics Problems," *Computer Methods in Applied Mechanics and Engineering* 436 (2025): 117699, https://doi.org/10.1016/j.cma.2024.117699.

26. S. A. Faroughi, N. Pawar, C. Fernandes, et al., "Physics-Guided, Physics-Informed, and Physics-Encoded Neural Networks and Operators in Scientific Computing: Fluid and Solid Mechanics," *ASME Journal of Computing and Information Science in Engineering* April 2024; 24, no. 4: 040802, https://doi.org/10.1115/1.4064449.

27. J. Li, Z. Yuan, Z. Li, et al., "Hardware-Driven Nonlinear Activation for Stochastic Computing Based Deep Convolutional Neural Networks," in *2017 International Joint Conference on Neural Networks (IJCNN)*, (IEEE, 2017), 1230–1236, https://doi.org/10.1109/IJCNN.2017.7965993.

28. M. Mikaitis, D. R. Lester, D. Shang, et al., "*Approximate Fixed-Point Elementary Function Accelerator for the SpiNNaker-2 Neuromorphic Chip*," In 2018 IEEE. 25th Symposium on Computer Arithmetic (ARITH), (IEEE, 2018), 37–44, https://doi.org/10.1109/ARITH.2018.8464785.

29. W.-H. Huang, J. Jia, Y. Kong, et al., "*Hardware Acceleration of Kolmogorov-Arnold Network (KAN) for Lightweight Edge Inference," In Proceedings of the 30th Asia and South Pacific Design Automation Conference (ASPDAC '25)*, 2024, 693–699,https://doi.org/10.1145/3658617.3697677.

30. K. Hu, J. Xie, K. Han, and F. Yang, 2024, SCKAN: A Stochastic Computing-Based Accelerator for Efficient Implementation of Kolmogorov-Arnold Networks. TechRxiv, https://doi.org/10.36227/techrxiv.172651873.32203440/v1.

31. Y. Peng, S. Hooten, T. V. Vaerenbergh, X. Xiao, M. Fiorentino & R. G. Beausoleil, "*Photonic KAN: a Kolmogorov-Arnold Network Inspired Efficient Photonic Neuromorphic Architecture," In NeurIPS 2024 Workshop Machine Learning with new Compute Paradigms* (2024), https://openreview.net/forum?id=xGymSunYzF.

32. T. Soliman, F. Muller, T. Kirchner, et al. "*Ultra-Low Power Flexible Precision FeFET Based Analog In-Memory Computing," In 2020 IEEE International Electron Devices Meeting (IEDM)*, (IEEE, 2020), 29.2.1–29.2.4, https://doi.org/10.1109/IEDM13553.2020.9372124.

33. A. Kawahara, R. Azuma, Y. Ikeda, K. Kawai, et al. "An 8 Mb Multi-Layered Cross-Point ReRAM Macro With 443 MB/s Write Throughput," *IEEE Journal of Solid-State Circuits* 48 (2013): 178–185, https://doi.org/10.1109/JSSC.2012.2215121.

34. W. Otsuka, K. Miyata, M. Kitagawa, et al. "*A 4Mb Conductive-Bridge Resistive Memory With 2.3GB/s Read-Throughput and 216MB/s Program-Throughput," In 2011 IEEE International Solid-State Circuits Conference*, (IEEE, 2011), 210–211, https://doi.org/10.1109/ISSCC.2011.5746286.

35. A. Kawahara, K. Kawai, Y. Ikeda, et al. "Filament Scaling Forming Technique and Level-Verify-Write Scheme With Endurance Over 107 Cycles in ReRAM," In 2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers, (IEEE, 2013), 220–221, https://doi.org/10.1109/ISSCC.2013.6487708.

36. A. Hardtdegen, C. La Torre, F. Cüppers, S. Menzel, R. Waser, and S. Hoffmann-Eifert, "Improved Switching Stability and the Effect of an Internal Series Resistor in HfO2/TiOx Bilayer ReRAM Cells," *IEEE Transactions on Electron Devices* 65 (2018): 3229–3236, https://doi.org/10.1109/TED.2018.2849872.

37. M. Le Gallo, R. Khaddam-Aljameh, M. Stanisavljevic, et al., "A 64-Core Mixed-Signal in-Memory Compute Chip Based on Phase-Change Memory for Deep Neural Network Inference," *Nature Electronics* 6 (2023): 680–693, https://doi.org/10.1038/s41928-023-01010-1.

38. M.-M. Jin, L. Cheng, Y. Li, et al, "Reconfigurable Logic in Nanosecond Cu/GeTe/TiN Filamentary Memristors for Energy-Efficient in-Memory Computing," *Nanotechnology* 29, 38 (2018): 10.1088/1361-6528/aacf84.

39. A. Mazumder, T. Ahmed, E. Mayes, et al., "Nonvolatile Resistive Switching in Layered InSe via Electrochemical Cation Diffusion," *Advanced Electronic Materials* 8, 4 (2022): 2100999, https://advanced.onlinelibrary.wiley.com/doi/pdf/10.1002/aelm.202100999,

40. K. Jeon, J. Kim, J. J. Ryu, et al., "Self-Rectifying Resistive Memory in Passive Crossbar Arrays," *Nature Communications* 12 (2021): 2968, https://doi.org/10.1038/s41467-021-23180-2.

41. J. Li, S.-G. Ren, Y. Li, et al., "Sparse Matrix Multiplication in a Record-Low Power Self-Rectifying Memristor Array for Scientific Computing," *Science Advances* 9 (2023): eadf7474, https://www.science.org/doi/pdf/10.1126/sciadv.adf7474,

42. V. Milo, A. Glukhov, E. Perez, et al. "Accurate Program/Verify Schemes of Resistive Switching Memory (RRAM) for In-Memory Neural Network Circuits," *IEEE Transactions on Electron Devices* 68 (2021): 3832–3837, https://doi.org/10.1109/TED.2021.3089995.

43. X. Liu, C. Bengel, F. Cüppers, et al., "Effect of Transistor Transfer Characteristics on the Programming Process in 1T1R Configuration," *IEEE Transactions on Electron Devices* 71 (2024): 2423–2430, https://doi.org/10.1109/TED.2024.3370536.

44. D. Maldonado, A. Baroni, S. Aldana, et al., "Kinetic Monte Carlo Simulation Analysis of the Conductance Drift in Multilevel HfO2-Based RRAM Devices," *Nanoscale* 16 (2024): 19021–19033, https://doi.org/10.1039/D4NR02975E.

45. S. Wiefels, N. Kopperberg, K. Hofmann, et al., "Reliability Aspects of 28 nm BEOL-Integrated Resistive Switching Random Access Memory," *Physica Status Solidi A*, 221,22 (2024): 2300401, https://doi.org/10.1002/pssa.202300401.

46. S. Wiefels, C. Bengel, N. Kopperberg, K. Zhang, R. Waser, and S. Menzel, "HRS Instability in Oxide-Based Bipolar Resistive Switching Cells," *IEEE Transactions on Electron Devices* 67 (2020): 4208–4215, https://doi.org/10.1109/TED.2020.3018096.

47. G. Molas and E. Nowak, "Advances in Emerging Memory Technologies: From Data Storage to Artificial Intelligence," *Applied Sciences* 11 (2021), https://doi.org/10.3390/app112311254.

48. Y. Ding, J. Yang, Y. Liu, et al., "16-layer 3D Vertical RRAM with Low Read Latency (18ns), High Nonlinearity (>5000) and Ultra-low Leakage Current (pA) Self-Selective Cells," In *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, (IEEE, 2023), 1–2, https://doi.org/10.23919/VLSITechnologyandCir57934.2023.10185341.

49. M. Xie, Y. Jia, C. Nie, et al., "Monolithic 3D Integration of 2D Transistors and Vertical RRAMs in 1T–4R Structure for High-Density

Memory," *Nature Communications* 14 (2023): 5952, https://doi.org/10.1038/s41467-023-41736-2.

50. J. Wu, F. Mo, T. Saraya, T. Hiramoto, and M. Kobayashi, "A Monolithic 3D Integration of RRAM Array With Oxide Semiconductor FET for in-Memory Computing in Quantized Neural Network AI Applications," In 2020 IEEE Symposium on VLSI Technology, (IEEE, 2020), 1–2, https://doi.org/10.1109/VLSITechnology18217.2020.9265062.

51. P. Polack, F. Altché, B. D. Andrea Novel, and A. de La Fortelle. *The Kinematic Bicycle Model: A Consistent Model for Planning Feasible Trajectories for Autonomous Vehicles?*," in 2017 IEEE Intelligent Vehicles Symposium (IV), (IEEE, 2017), 812–818, https://doi.org/10.1109/IVS.2017.7995816.

52. A. G. Baydin, B. A. Pearlmutter, A. A. Radul, and J. M. Siskind, "Automatic Differentiation in Machine Learning: A Survey," *Journal of Machine Learning Research* 18 (2018): 1–43.

53. S. Linnainmaa, "Taylor Expansion of the Accumulated Rounding Error," *BIT Numerical Mathematics* 16 (1976): 146–160.

54. X. Feng, G. Shen, J. Hu, M. Li, and N. Wong, "Stochastic Multivariate Universal-Radix Finite-State Machine: A Theoretically and Practically Elegant Nonlinear Function Approximator," In *ASPDAC '25: Proceedings of the 30th Asia and South Pacific Design Automation Conference*, (Association for Computing Machinery, 2025), 211–217, https://doi.org/10.1145/3658617.3697714.

55. J. Yu, J. Park, S. Park, et al., "NN-LUT: Neural Approximation of Non-Linear Operations for Efficient Transformer Inference," In *Proceedings of the 59th ACM/IEEE Design Automation Conference, DAC'22*, (Association for Computing Machinery, 2022), 577–582, 22, https://doi.org/10.1145/3489517.3530505.

56. Y. Pan, X. Jia, Z. Cheng, et al., "An STT-MRAM Based Reconfigurable Computing-in-Memory Architecture for General Purpose Computing," *CCF Transactions on High Performance Computing* 2 (2020): 272–281, https://doi.org/10.1007/s42514-020-00038-5.

57. S. Gupta, M. Imani, and T. Rosing, "FELIX: Fast and Energy-Efficient Logic in Memory," In *Proceedings of the 2018 IEEE International Conference on Computer-Aided Design (ICCAD)*, (IEEE, 2018), 1–7, https://doi.org/10.1145/3240765.3240811.

58. A. Eliahu, R. Ben Hur, A. Haj Ali, and S. Kvatinsky, "mMPU: Building a Memristor-based General-purpose In-memory Computation Architecture", In Multi-Processor System-on-Chip 1, chap. 6, (John Wiley Sons, Ltd., 2021), 119–131. https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119818298.ch6.

59. S. Singh, C. K. Jha, A. Bende, et al., "*MemSPICE: Automated Simulation and Energy Estimation Framework for MAGIC-Based Logic-in-Memory*," In 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC), (IEEE, 2024), 282-287, https://doi.org/10.1109/ASP-DAC58780.2024.10473924.

60. A. Khataei and K. Bazargan, "*Compressed LUT An Open Source Tool for Lossless Compression of Lookup Tables for Function Evaluation and Beyond*," in Proceedings of the 2024 ACM/SIGDA International Symposium on Field Programmable Gate Arrays, FPGA, (Association for Computing Machinery, 2024), 2–11, 24, https://doi.org/10.1145/3626202.3637575.

61. F. de Dinechin and B. Pasca, "Designing Custom Arithmetic Data Paths With FloPoCo," *IEEE Design & Test of Computers* 28 (2011): 18–27, https://doi.org/10.1109/MDT.2011.44.

62. S. R. Faraji and K. Bazargan, "Hybrid Binary-Unary Hardware Accelerator," *IEEE Transactions on Computers* 69 (2020): 1308–1319, https://doi.org/10.1109/TC.2020.2971596.

63. F. Cüppers, S. Menzel, C. Bengel, et al., "Exploiting the Switching Dynamics of HfO2-Based ReRAM Devices for Reliable Analog Memristive Behavior," *APL Materials* 7 (2019): 091105, https://doi.org/10.1063/1.5108654.

64. C. Bengel, J. Mohr, S. Wiefels, et al., "Reliability Aspects of Binary Vector-Matrix-Multiplications Using ReRAM Devices," *Neuromorphic Computing and Engineering* 2, (2022): 034001, https://doi.org/10.1088/2634-4386/ac6d04.

65. H. Zhang, D. Chen, and S.-B. Ko, "New Flexible Multiple-Precision Multiply-Accumulate Unit for Deep Neural Network Training and Inference," *IEEE Transactions on Computers* 69 (2020): 26–38, https://doi.org/10.1109/TC.2019.2936192.

66. P. Kogge and J. Shalf, "Exascale Computing Trends: Adjusting to the "New Normal" for Computer Architecture," *Computing in Science & Engineering* 15 (2013): 16–26, https://doi.org/10.1109/MCSE.2013.95.

67. W. J. Dally, "Challenges of Future Computing," in *HiPEAC Keynote* (USENIX Association, 2015).

68. C. Jekel, pwlf, https://pypi.org/project/pwlf/.

69. A. Paszke, S. Gross, F. Massa, et al., 2019, PyTorch: An Imperative Style, High-Performance Deep Learning Library, https://doi.org/10.48550/arXiv.1912.01703.

Supporting Information

Additional supporting information can be found online in the Supporting Information section. **Supporting Fig. S1:** CIM-based Segment Selection Unit's (CIM-SSU's) operation for positive and negative numbers. **Supporting Fig. S2:** Example operation of the CIM-SSU for a 16-bit fixed-point format with four breakpoints. Input $x = 10.5$ and breakpoints $X_1 = -22.984375$, $X_2 = -6.953125$, $X_3 = 1.76171875$, $X_4 = 36.53515625$. Please note that the sense amplifier output (i.e. RG) is 0 for disabled columns. **Supporting Fig. S3:** Example operation of the CIM-SSU for a 16-bit BFloat format with four breakpoints. Input $x = 10.5$ and breakpoints $X_1 = -22.984375$, $X_2 = -6.953125$, $X_3 = 1.76171875$, $X_4 = 36.53515625$. Please note that the sense amplifier output (i.e. RG) is 0 for disabled columns. **Supporting Fig. S4:** Extended results of Figure 3a. Tile error and energy (excluding communication and chip-level peripheral overhead) as a function of the number of PWL segments (N) for various elementary functions. The crossover point between energy and error occurs around $N = 32$ (i.e., either between 16-32 or 32-64). This motivates choosing $N = 32$ as the standard tile configuration. **Supporting Fig. S5:** Extrapolation of read drift measurements of Figure 4d for long term reliability verification. **Supporting Fig. S6:** a) Schematic of a 1T1R device showing V_{SL}, V_{WL} and V_{BL}. b) Programming scheme of the READ disturb. c) Read disturb behavior of our VCM device applied with 31 consecutive read pulses of 16 s each with a 0.4 V read voltage. This duration corresponds to 320 billion read operations per cell, considering the array read time of 1.5 ns. As presented in main text, the sense amplifier can handle this worst-case variation without any errors even with worst-case transistor variation. **Supporting Fig. S7:** a) Programming scheme of the 1T1R device for sweep measurements. b) Device-to-device variation: CDF values of LRS and HRS state obtained by one-shot switching of 18 1T1R devices for at least 100 cycles without a read and verify. The goal here is to show that all devices reach the desired resistance states proposed in the application, which are LRS $\leq 10k\Omega$ and HRS $\geq 3M\Omega$ (dashed line). **Supporting Fig. S8:** a) Programming scheme of the 1T1R device for endurance measurements. READ pulses are applied after every 10^n-3 {n = 1,2,3,..} SET/RESET operations. b) Stable switching of the 1T1R devices was typically obtained up to $5 \cdot 10^5$ cycles. This value, which allows the use of the devices in the proposed application, represents a lower limit of the device performance. Note: the variation of HRS and LRS is due to not using a read and verify scheme. **Supporting Fig. S9:** The SET probability as a function of VSL and VWL with different pulse widths of 1µs (a) and 10µs (b) on the 180nm same 1T1R structure. **Supporting Fig. S10:** Extracted LRS distribution after the different SET pulses plotted for one exemplary HRS distribution. **Supporting Fig. S11:** Simulated operating voltages of the proposed 28nm 1T1R cell design using a 1.8 V NMOS access transistor, confirming compliance with the required voltage ranges for forming, SET, RESET, and READ modes. **Supporting Fig. S12:** Waveform of the KA-CIM tile depicting the CIM-SSU array, Slope array, and Intercept array during read operations. **Supporting Fig. S13:** An example two-layer KAN with 5 inputs and

1 output. Here, F_{15} is assumed to be distributed across two tiles with Tile Partition enabled. **Supporting Fig. S14:** Pipelined timing diagram of KA-CIM for [17, 1, 14]. Latency: 30 cycles; throughput:1/max(Layer1lat, Layer2lat) = 1/19ns ≈ 52.63M samples/s. **Supporting Fig. S15:** Medium Area Variant: Single core design of KA-CIM for KAN [17,1,14] with 32 PWL segments per function. Suitable for KAN trained with up to G = 60, K = 3. Comparable to the KAN2 variant in (29). **Supporting Fig. S16:** Low Area Variant: Single core design of KA-CIM for KAN [17,1,14] with 16 PWL segments per function. Suitable for KAN trained with up to G = 10,K = 3. Comparable to the KAN1 variant in (29). **Supporting Fig. S17:** Energy breakdown for the two KA-CIM variants designed for [17,1,14] KAN. Tile = SB + AS + M, C = SI + SO + SC, Σ = stage1 + stage2. Refer Supplementory Material 8 for the definitions of FI, SI, SB, AS, M, SO, SC, and ST. **Supporting Fig. S18:** Error Behavior of KA-CIM in Root-Sum-Square Computations with Quadratic Scaling of Intermediate Outputs:Four-Variable Equation $\sqrt{(x_1^2 + x_2^2 + x_3^2 + x_4^2)}$ Simulated on KA-CIM (32 PWL Segments) vs Baseline Full Precision (FP32). Note: The blue Baseline curve overlaps with the orange curve in most parts of the graph, indicating the low error of KA-CIM (i.e. median error = 90th percentile error = 6.32×10^{-2}). **Supporting Fig. S19:** a) Standard digital-logic-based PWL approximation unit, similar to NN_LUT (55). Layout of digital-logic basedPWL approximation unit in 28 nm technology (excluding the MAC). **Supporting Table S1.** Example encoding for positive and negative breakpoints before storing them in a desired memory array in case of BFloat16. **Supporting Table S2.** Comparison of memristor operating currents: measured vs simulated. **Supporting Table S3.** Latency and Energy of various KA-CIM circuits. **Supporting Table S4.** Mapping of KAN [17,1,14] to a single core KAN-CIM design. **Supporting Table S5.** Analysis of KA-CIM error, latency, and energy for various applications is conducted with N = 32 and no tilepartitioning. Errors for both KA-CIM and standard compute using the BFloat16 data type are benchmarked against the 32-bitfloating-point baseline. Median error is reported instead of the average error, as it effectively handles outliers. We assume thatthe outlier errors can be minimized by assigning individual PWL segments or increasing segments using multiple tiles in thefuture. Details on converting these example multi-variable equation to KAN are provided in Supplementary Material 11 and 12. **Supporting Table S6.** Comparison of KAN-CIM with ASIC accelerator and CPU for non-linear function computation. Exp function isused as a target task for all three architecture types. EDAP = Energy-Delay-Product, EDAP = Energy-Delay-Area-Product.