

An SRAM-based fully-integrated analog closed-loop in-memory computing accelerator

Piergiulio Mannocci¹, Carlo Zucchelli¹, Irene Andreoli¹, Andrea Pezzoli¹,
Enrico Melacarne¹, Giacomo Pedretti², Flavio Sancandi¹, Corrado Villa¹,
Zhong Sun³, Umberto Spagnolini¹, Daniele Ielmini¹

¹Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza
Leonardo da Vinci 32, Milano, 20133, Italy.

²Artificial Intelligence Research Lab (AIRL), Hewlett Packard Labs, Milpitas, CA, USA.

³Institute for Artificial Intelligence, School of Integrated Circuits, Peking University, Beijing
100871, China.

Contributing authors: piergiulio.mannocci@polimi.it; daniele.ielmini@polimi.it;

Abstract

Conventional processors are unsuited for the increasingly data-intensive workloads brought upon by the resurgence of artificial intelligence (AI) and machine learning (ML) in the latest years. The widespread diffusion of ML techniques in many different technological fields has prompted both academia and industry to rethink the processor structure, abandoning the outdated physical separation between memory and computing units of von Neumann's architecture. Novel computing paradigms have emerged, such as in-memory computing (IMC), in which memory and computing are blended together in a single processing unit. IMC allows the complete elimination of the energy and latency overheads associated with the back-and-forth data transfer between memory and computing units. When implemented using crossbar arrays (CBAs) of memory devices, IMC represents a competitive candidate for next-generation energy-efficient accelerators for ML and AI on the edge. IMC was shown to be particularly effective in accelerating low-level, data-intensive algebraic operations such as matrix-vector (MVM) and inverse-matrix-vector multiplication (IMVM). However, while several IMC-based MVM demonstrators have been reported in recent years, IMVM demonstrators have faced additional challenges owing to the increased complexity of the circuit implementation, entailing analog feedback operation and suffering from increased sensitivity. Nonetheless, closed-loop IMC (CL-IMC) IMVM may capitalize on the IMC advantage even more than MVM, owing to the higher $\mathcal{O}(n^3)$ computational complexity, where n is the matrix size, which is instead theoretically reduced to $\mathcal{O}(1)$ in IMC-based systems. Here, we present a fully integrated IMC chip for IMVM designed and fabricated in 90 nm complementary metal-oxide-semiconductor (CMOS) technology. The chip features two 64×64 memory arrays, enclosed in an analog feedback loop by on-chip operational amplifiers (opamps), digital/analog (DAC), and analog/digital converters (ADC), providing the first complete primitive for acceleration of inverse operations under the IMC framework. We validate the integrated circuit (IC) by performing experiments on three real-life toy problems, namely, solutions of systems of differential equations by recursive block inversion (RBI), sensor fusion by Kalman filter for simultaneous 3D trajectory and velocity estimation in sounding rockets, and acceleration of inverse kinematics in robotic arms for industrial automation and autonomous robots. Experimental results closely match the accuracy of fully digital systems working at the equivalent IC precision while simultaneously providing consistent advantages in terms of latency, energy, and area consumption. The obtained results represent the first large-scale experimental demonstration of the CL-IMC concept and consolidate its position as a promising candidate for next-generation energy-efficient accelerators on-the-edge.

Keywords: in-memory computing, closed-loop computing, block inversion, Kalman filter, inverse kinematics

1 Introduction

Conventional computing processors are based on the von Neumann architecture shown in Fig. 1a, predicated on the physical separation of memory and processing unit [1]. The two units are interconnected by a data bus, over which data is transmitted from the memory unit to the processing unit and back to store the processed data. This flexible framework allows implementation on computing platforms with different characteristics and requirements but is intrinsically unoptimized for modern-day data-intensive workloads. While computational latency can be minimized by leveraging high-bandwidth external memories [2, 3], or large on-chip static random access memory (SRAM) [4, 5], both result in unsustainably high energy consumption for continuous data access and transfer [6, 7].

To overcome the *von Neumann bottleneck*, in-memory computing (IMC) was proposed [8]. In IMC, computation is performed *in situ* within the memory elements, thus completely eliminating the need for data transfer. Several different flavors of IMC have been proposed in the latest years, relying on various types of memory technologies, such as volatile static random access memories (SRAM) [9] or dynamic random access memories (DRAM) [10], as well as nonvolatile resistive memories such as resistive random access memory (RRAM), phase change memory (PCM), or spin-transfer torque magnetic random access memory (STT-MRAM) [11, 12]. Both volatile and nonvolatile memories have been shown to be compatible with different IMC implementations, namely *digital IMC*, where logic gates [13] are used to perform multiply-accumulate operations, and *analog IMC*, where physical equivalents of mathematical operations, such as Ohm’s law and Kirchhoff’s current law for multiplication and accumulation, respectively, are typically exploited to enable high-throughput, low-power computing [10]. Fig. 1b shows a conceptual scheme of an analog IMC core composed of an array of nonvolatile memories and dedicated peripheral circuitry tasked with programming and operating on the array. IMC cores are typically envisioned as elementary functional units interconnected by dedicated networks-on-chip (NoCs) in more complex IMC processors, as shown in Fig. 1c [14].

IMC is particularly suited to accelerate data-intensive matrix-based operations by taking advantage of the high-density, highly-parallelized crossbar array (CBA) structure, whereby each memory device is located at the intersections of row and column metal lines [12]. Fig. 1d shows a prototypical CBA-based IMC accelerator for matrix-vector multiplication (MVM), a computationally intensive operation in conventional computers that lies at the heart of many applications of both academic and industrial interest. Among these, neural networks [15, 16], image processing [17, 18], and combinatorial optimization [19, 20] have been demonstrated in the recent years. In such an open-loop IMC (OL-IMC) MVM circuit, elements A_{ij} of a target matrix $\mathbf{A}$ are first stored in the memory devices of a CBA as conductances G_{ij} . A voltage vector $\mathbf{v}$, whose elements V_j represent the components x_j of the input vector $\mathbf{x}$, is then applied to the columns of the CBA with grounded rows. According to Ohm’s law, each element yields a current $I_{ij} = G_{ij}V_j$, which is accumulated on the CBA rows following Kirchhoff’s law, namely:

$$I_i = \sum_{j=1}^N G_{ij}V_j, \quad (1)$$

which is equivalent to the MVM operation $\mathbf{i} = \mathbf{G}\mathbf{v}$, describing the mathematical expression $\mathbf{b} = \mathbf{A}\mathbf{x}$. Owing to the intrinsic spatial parallelism of the CBA structure, the MVM operation is performed in just one computational step [21] or $\mathcal{O}(1)$, thus providing a remarkable advantage with respect to the $\mathcal{O}(N^2)$ complexity, where N is the matrix size, of MVM in conventional digital computers [22].

The computational capability of IMC can be further extended by enclosing the CBA in a feedback loop using analog operational amplifiers (opamps). In this closed-loop IMC (CL-IMC) concept, shown in Fig. 1e [23], the CBA rows become virtual ground nodes of the circuit thanks to the feedback effect. By injecting an external current vector $\mathbf{i}$ at the CBA rows, the output voltage vector $\mathbf{v}$ probed on the opamp outputs then satisfies:

$$\mathbf{v} = -\mathbf{G}^{-1}\mathbf{i}, \quad (2)$$

which thus provides a physical accelerator for the solution of a system of linear equations $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, where $\mathbf{A}$, $\mathbf{x}$, $\mathbf{b}$ are the linear system matrix, unknown, and known term vectors respectively. Note that such an inverse-matrix-vector multiplication (IMVM) is an even harder task for conventional computers, requiring a

computational complexity $\mathcal{O}(N^\alpha)$, where α is between 2 and 3 [24]. In IMC-based IMVM, the time complexity is instead independent of the matrix size and defined by the smallest matrix eigenvalue [25, 26], providing a promising avenue for energy-efficient acceleration of inverse operations in several fields. The introduction of the CL-IMC concept [23] has spun out investigation of inverse-matrix acceleration in telecommunications [27, 28], sensor fusion [29–31], ranking systems [32–34], machine learning [35–38], combinatorial optimization [39, 40], and general linear algebra [41–45]. CL-IMC also bears similarities with well-known Boltzmann machines and Hopfield networks [20]. To fully unleash the CL-IMC potential, several challenges must be overcome. At the memory level, large and reliable CBAs are needed to approach the typical large-scale problems of industrial interest. At the circuit level, high-gain and high-bandwidth opamps are required to provide low computation latency and high accuracy, together with careful design of the analog-digital conversion blocks. Finally, a comprehensive portfolio of mathematical operators and experimental demonstrations of relevant use cases should be assembled to encourage the adoption of the CL-IMC framework in next-generation processors.

In this work, we present an integrated circuit (IC) for CL-IMC fabricated in 90 nm CMOS process, comprising two 64×64 CBAs and on-chip opamps, digital-to-analog (DAC) and analog-to-digital (ADC) converters. We provide extensive characterization of the memory cell and circuit peripherals and introduce novel error correction algorithms, with significantly reduced overhead with respect to our previous work [31]. We further demonstrate three novel applications, namely solution of systems of differential equations (DEs), simultaneous three-dimensional (3D) trajectory and velocity tracking in sounding rockets by Kalman filter (KF), and in-memory acceleration of inverse kinematics (IK) in robotics. Measurement results provide near-software equivalent accuracy, validating the CL-IMC concept for modern-day data-intensive applications.

2 Closed-loop in-memory computing testchip

We designed an IC to execute the IMVM within a memory CBA with state-of-the-art complementary metal-oxide-semiconductor (CMOS) technology at the 90 nm node. Fig. 2a shows the packaged IC while Fig. 2b illustrates the overall chip architecture. The chip features a primary 64×64 memory CBA enclosed in a feedback loop by on-chip opamps. A secondary 64×64 memory CBA, whose columns are connected to the columns of the primary array through inverting analog buffers, allows representation of positive and negative values under a differential scheme, *i.e.* $\mathbf{G} = \mathbf{G}_+ - \mathbf{G}_-$ (see Extended Data Fig. 2). Input signals are applied to the circuit through on-chip 7-bit voltage DACs coupled to the CBAs by 3-bit programmable resistors, whereas output signals are converted to the digital domain by on-chip 8-bit voltage ADCs. The IC can be reconfigured by dedicated configuration registers, which are accessed through an 8-bit parallel input/output (I/O) bus by an external microcontroller unit (MCU) hosted on the dedicated printed circuit board (PCB) shown in Extended Data Fig. 1a.

A key requirement for MVM and IMVM concepts in Fig. 1 is the linearity of Ohm’s law, namely $\mathbf{i} = \mathbf{G}\mathbf{v}$. Emerging non-volatile memory technologies generally do not provide sufficiently linear I-V curves [46–49]. Therefore, to ensure high linearity of the memory cell and decouple the experimental validation of the closed-loop concept from nonidealities of emerging memory devices, we opted for the 40T5R (40T5R) cell structure schematically shown in Fig. 2c. The 40T5R memory cell consists of five integrated n-well resistors with scaled conductances $G_0 = 6.25 \mu\text{S}$, $2G_0$, $4G_0$, $8G_0$ and $16G_0$, each connected to a dedicated two-transistor transmission gate (TG) enabled by a corresponding six-transistor SRAM. Each SRAM state D_n , where $n = 0, 1, \dots, 4$, can be reconfigured to achieve a desired conductance:

$$G = \sum_{n=0}^4 D_n G_0 2^n \quad (3)$$

with 5-bit precision. Fig. 2d shows the layout of the 40T5R memory cell, where integrated resistors occupy approximately one-half of the cell area for an overall $442 \mu\text{m}^2$ cell size. Memory cells were individually characterized using direct memory access (DMA) and an on-board characterization circuit (Extended Data Fig. 1b). Fig. 2f shows the measured conductance G for 4096 cells programmed with increasing values from $G = 0$ to $G = 31G_0$, corresponding to a 5-bit unsigned integer (UINT5) precision. The characterization results highlight

the high linearity of the memory cell, with a calculated $R^2 = 0.9996$ (see Extended Data Fig. 1e. Notably, higher conductive states show an increased standard deviation σ_G compared to lower conductive states owing to the cumulative variability of the selected devices (Extended Data Fig. 1c-d). The standard deviation covers the range $0.1 \mu\text{S} < \sigma_G < 1 \mu\text{S}$, corresponding to less than 0.5% of the full-scale conductance of $194 \mu\text{S}$. While the proposed cell allows emulation of a potentially mature resistive memory, it might not provide ideal area efficiency and scalability for use in highly optimized IMC macros. To improve cell scalability, integrated resistors could be replaced by BEOL-fabricated resistive memories such as RRAM [10]. The higher programming cost of RRAM can be amortized by performing one-time programming of binary-scaled conductance levels with high-precision program-and-verify techniques [50], allowing mitigation of fabrication variability, conversely retaining fast reprogrammability at runtime thanks to SRAM-TG pairs. Eventually, the latter could be replaced by non-volatile selectors realized by memtransistors [51] or flash memory-based cells [10], enabling nonvolatile data storage.

3 In-memory matrix inversion

To characterize the solution accuracy of the IC, we first performed linear system solution experiments on random positive-definite matrices with increasing condition number κ , corresponding to an increased solution sensitivity to small perturbations [26]. To this aim, the IC was configured to operate as single-matrix circuit (SMC) (Extended Data Fig. 2a) or double-matrix circuit (DMC) (Extended Data Fig. 2b) depending on the sign of the matrix elements, providing equivalent UINT5 or 6-bit integer (INT6) precision respectively. Dedicated calibration algorithms (see Methods, Sec. 7.4, and Extended Data Fig. 3) were implemented in the onboard MCU to mitigate errors in the analog domain, including analog noise, opamp offset and saturation, and ADC nonlinearity.

Fig. 3a shows the measured output voltages as a function of the expected output voltages for 1000 experiments of 38×38 linear system solution with increasing condition number κ , highlighting the chip accuracy. Absolute voltage error histograms reported in Fig. 3b particularly highlight a zero-mean voltage error. Notably, the standard deviation exhibits a non-monotonic behavior with κ owing to the concurrency of different non-idealities, including DAC variability, memory variability, parasitic I-R drop, finite opamp gain, and opamp output resistance. Particularly, DAC variability dominates at lower κ , with interplay effects resulting in partial improvement for intermediate κ . To quantify the absolute precision of our IC, we considered the maximum standard deviation $\sigma_v \leq \pm 11 \text{ mV}$ in the κ interval of interest and compared it with the 1.8 V output full-scale range, corresponding to an equivalent output precision of 7.1 bits.

Notably, the SMC and DMC are limited to the inversion of positive-definite matrices [23] for the feedback stability constraints. To allow the inversion of indefinite matrices, always-stable topologies have been proposed [41] based on the pseudoinverse concept. Here, we used a modified version of the IMC linear regression circuit (LRC) exploiting inverting amplifiers only, which was block-mapped [26] to the CBA. Negative coefficients in the matrix were implemented employing a reference-topology (RT) approach [31], trading off circuit accuracy for stability. The complete circuit and block mapping are shown in Extended Data Fig. 2c and allow the solution of indefinite linear systems in signed 5-bit integer (INT5) precision, compatible with a wide range of error-tolerant applications [17, 27, 37, 38, 52, 53].

In many application fields of practical interest, the linear system size typically exceeds the hundreds of equations [54–56], which might be larger than the physical array size. To allow the solution of such large problems with the available CBA size in the chip, recursive block-inversion (RBI) algorithms have been proposed [57, 58], whereby a large matrix $\mathbf{A}$ is partitioned in smaller sub-blocks that are processed by multiple MVMs and IMVMs (Extended Data Fig. 4). For matrices whose size is compatible with the IC array size, direct inversion (DI) is used with either SMC or DMC at INT6 precision or LRC and RT at INT5 precision. Conversely, when the matrix size exceeds the available hardware, the RBI technique is used, with each sub-operation performed via on-chip MVM and IMVM. Sub-operations are further amenable to parallelization and pipelining for increased throughput when multiple IMC cores are available to distribute the RBI computation. We validated the IC accuracy in solving linear systems of increasing size N from $N = 2$ to $N = 512$ and

condition number κ from $\kappa = 10$ to $\kappa = 100$. Fig. 3c reports the obtained ℓ_2 -norm relative error, namely [24]:

$$\varepsilon = \frac{\|\mathbf{v}_{out} - \mathbf{v}_{out,id}\|_2}{\|\mathbf{v}_{out,id}\|_2}, \quad (4)$$

where $\mathbf{v}_{out}$, $\mathbf{v}_{out,id}$ are the measured and ideal output voltage vectors, for linear systems described by matrices of increasing size. Particularly, linear systems up to $N = 32$ were solved by DI, whereas larger matrices were processed by RBI. Notably, the error increases with κ and is mostly independent of N for DI, in line with theoretical results [26]. For RBI, the solution error is determined by two concurring processes, namely (i) the intrinsic error experienced by sub-operations and (ii) error propagation and accumulation across the RBI steps. At high κ , (i) dominates over (ii), with larger matrices showing improved accuracy owing to the reduced condition number of sub-blocks [59]. Conversely, for well-conditioned problems, (ii) dominates over (i), with smaller matrices incurring reduced accumulation and improved accuracy. Careful optimization of the array size should target reduced error accumulation while mitigating the impact of space-dependent analog non-idealities such as I-R drop [60]. Another significant source of error is linked to the inaccurate computation of the Schur complements, where error correction strategies such as iterative refinement techniques [61] or mixed-precision systems [42, 62] could be used to increase the computation accuracy. Since the Schur complement is independent of the known term and only depends on submatrices of the linear system matrix, the higher computational cost can be amortized by reusing the high-precision Schur complement for multiple linear system solutions.

Large-scale matrix inversion is a core operation for the solution of DEs [63] in many scientific fields, including quantum computing [64, 65], computational biology [66], and mechanical analysis [67]. Among state-of-the-art techniques for the solution of DEs is the finite differences method [68], where continuous domains are discretized in space and time to allow fast numerical integration by energy-hungry graphics processing units (GPUs) [69]. Here, we demonstrate in-memory solution by CL-IMC of a system of DEs describing the time dynamics of a 32×32 CL-IMC circuit [26] over 100 time-steps, corresponding to the solution of a 3200×3200 linear system (see Supplementary Note 8.1). Experimental RBI results in Fig. 3d show comparable accuracy with respect to a digital solver at 5-bit precision, where iterative techniques [61, 70] or mixed-precision systems [42, 62] can be used to further refine the solution beyond the analog limit for scientific computing applications.

To quantitatively assess the performance of our circuit, we computed the main performance metrics for our 64×64 IMVM core in 90 nm technology, as well as projections for an optimized 256×256 core in 14 nm technology, comprising improved circuit peripherals (see Methods, Sec. 7.6 and Extended Data Fig. 7a-b), for the solution of a 1024×1024 linear system by RBI processing. While the latency of our current testchip is mostly limited by the ADC conversion latency, effectively providing $\mathcal{O}(1)$ computation, after careful optimization of the analog peripherals, we expect the performance to be limited by the digital post-processing, with a worst-case $\mathcal{O}(N)$ complexity, retaining significant advantage over the typical $\mathcal{O}(N^\gamma)$ complexity, where γ is between 2 and 3, of linear system solution in digital processors, with potential for comparable throughput, $1330 \times$ energy efficiency, and $985 \times$ area efficiency compared to commercial GPUs [71] (Extended Data Fig. 7c).

4 In-memory Kalman filter

Fig. 4a shows the conceptual scheme of a Kalman filter (KF), a linear quadratic estimation algorithm for a set of unknown variables or *state* $\boldsymbol{\theta}$ [72]. KF operates in a prediction-correction fashion by estimating the joint probability distribution between a series of measurements and the estimates of the unknown variables, as illustrated in Fig. 4b. At each iteration, *a-posteriori* estimates $\boldsymbol{\theta}_{n-1|n-1}$ at the preceding timestep are first used to compute an *a-priori* estimate $\boldsymbol{\theta}_{n|n-1}$ at the current timestep using a model $\mathbf{A}$ of the observed physical system. Then, measurements $\mathbf{x}_n$ of either the unknown or related physical variables are taken into account, where the relation between measurements and unknowns is described by the measurement model $\mathbf{B}$ and its covariance $\mathbf{C}_{\beta\beta}$. Measurements are used to compute a correction for the *a-priori* prediction, such that the *a-posteriori* estimate of the state at the current timestep is given by:

$$\boldsymbol{\theta}_{n|n} = \mathbf{A}\boldsymbol{\theta}_{n-1|n-1} + \mathbf{K}(\mathbf{x}_n - \mathbf{B}\mathbf{A}\boldsymbol{\theta}_{n-1|n-1}), \quad (5)$$

where the Kalman gain matrix $\mathbf{K}$ defines the weight of the correction and is given by:

$$\mathbf{K} = \mathbf{C}_\theta \mathbf{B}^T (\mathbf{B} \mathbf{C}_\theta \mathbf{B}^T + \mathbf{C}_{\beta\beta})^{-1}, \quad (6)$$

where $\mathbf{C}_{\theta(n|n)}$ reads:

$$\mathbf{C}_{\theta(n|n)} = (\mathbf{I} - \mathbf{K}\mathbf{B})\mathbf{C}_{\theta(n|n-1)}. \quad (7)$$

Thanks to the prediction-update mechanism, KF provides improved precision for measurements alone and prevents the uncertainty from increasing indefinitely with time [72]. Typical applications include guidance and navigation control of vehicles [73], robotic motion control [74], multi-sensor fusion [75], as well as time series forecasting in economics [76].

Fig. 4c shows the equivalent circuit matrix for KF, where each KF block is mapped to a corresponding memory array location [31] to represent Eqs. (5)-(6) (see Supplementary Note 8.2). Owing to the presence of both IMVMs and MVMs, the overall computational complexity is given by $\mathcal{O}(m^3 + m^2 + 4ms^2 + 3ms + 2s^2)$, where m and s are the number of measurements and state variables, respectively. The last *a-posteriori* estimate $\theta_{(n-1|n-1)}$ and current timestep measurements $\mathbf{x}_n$ are provided to the circuit as analog input currents, while the prediction and correction vectors are obtained as output analog voltages. Previous experimental demonstrations were limited to two-dimensional tracking of spatial coordinates in the presence of measurements of the same quantities [31]. In this work, we consider the more complex problem of simultaneously estimating *xyz* position and *xyz* velocity for a three-dimensional sounding rocket [77], assuming the latter to be equipped with position sensors for the *xyz* coordinates only. Notably, kinematic Kalman filters inferring unmeasured quantities are more susceptible to numerical inaccuracies with respect to static Kalman filters, which only refine measured data [78, 79]. As such, we take full advantage of the adaptive precision IMVM algorithm introduced in this work (see Methods, Sec. 7.4). We consider a constant-velocity motion model, resulting in a 27×27 equivalent matrix programmed into the IC. In principle, the state covariance submatrix should be recomputed at each time-step, requiring reprogramming of $s^2 = 36$ memory cells out of 729. To reduce the number of reprogramming and foster data reuse, we exploit the asymptotic formulation of the Kalman gain, allowing us to avoid recomputation of the state covariance matrix at each time-step [72]. Measurement data were synthetically generated from telemetry data of a real rocket launch [80] equipped with a global positioning system (GPS) and barometers for altitude estimation. Fig. 4d shows the reconstructed trajectory in the *xyz* space for our experimental IMC-based testchip, together with fully-digital implementations at FP64 and INT5 precision. The experimental solution closely matches the INT5 digital implementation for all coordinates, for both trajectory (Figs. 4e-g) and velocity (Figs. 4h-j), providing consistent improvement in localization accuracy with respect to the unfiltered sensor data (Extended Data Figs. 5b-d), as well as enabling accurate computation of the rocket's velocity (Extended Data Figs. 5e-g), which is an essential parameter for apogee detection algorithms [81]. These results ensure an accurate KF computation with considerable improvement in both throughput and energy efficiency (Extended Data Fig. 7b) when compared with microcontroller-based implementations [77].

5 In-memory inverse kinematics

Fig. 5a shows a conceptual scheme of a typical robotic arm composed of three *links* L_1, L_2, L_3 interconnected by rotational *joints* described by angles q_1, q_2, q_3 . Different configurations of the joint angles allow the arm terminal or *end effector* to reach different positions (x, y, z) and orientations θ inside a bounded region named *workspace*. Fig. 5b illustrates the main operations involved in robotic arm control. In forward kinematics (FK), the end effector position $\mathbf{s}$ is computed from the joint angles $\mathbf{q}$ by the Denavit-Hartenberg kinematic matrices [82], typically involving repeated MVMs with $\mathcal{O}(dn)$ complexity, where n is the number of links and d is the number of end effectors. On the other hand, inverse kinematics (IK) aims at computing the set of joint angles $\mathbf{q}$ based on a target end effector position $\hat{\mathbf{s}}$. Owing to the problem's intrinsic nonlinearity, iterative algorithms are typically needed to progressively update the joint angles according to:

$$\mathbf{q}^{k+1} = \mathbf{q}^k + \delta \mathbf{q}^k \quad (8)$$

$$\delta \mathbf{q}^k = \mathcal{J}(\mathbf{q})^T \left[\mathcal{J}(\mathbf{q}) \mathcal{J}(\mathbf{q})^T + \lambda^2 \mathbf{I} \right]^{-1} \delta \mathbf{s}^k \quad (9)$$

where $\delta \mathbf{s}^k = \hat{\mathbf{s}} - \mathbf{s}^k$ is the *position error* at the k -th iteration, $\mathcal{J}(\mathbf{q})$ is the Jacobian matrix for the current joint angles, and λ is a regularization parameter. Due to the presence of inverse operations and the iterative nature of the algorithm, IK features a relatively high $\mathcal{O}[(d^3 + d^2n + dn)k]$ computational complexity, where k is the number of iterations. Fig. 5c schematically shows the operations involved in the movement between two target points. At each iteration, the end effector position $\delta \mathbf{s}^k$ is recomputed by FK, thus allowing to update parameters $\mathcal{J}(\mathbf{q})$ and λ . A new estimate for the joint angles is then obtained according to Eqs. (8)-(9). The iteration stops when the error between the corresponding position obtained by FK and the target position becomes smaller than a given tolerance.

For the experimental demonstrations with the integrated circuit, we considered a 4-link robotic arm with a single-end effector ($n = 4$, $d = 1$) operating in a 3D workspace. The corresponding 7×7 equivalent circuit matrix was programmed into the IC with 3 additional rows and columns to provide positive/negative weight mapping for an overall 10×10 active submatrix size. Thanks to the iterative refinement, the precision of IMC-based IK is ultimately limited by the FK precision analogously to mixed-precision computing systems [62]. By executing the lower-complexity FK task in a floating-point digital processor and the high-complexity IK task in the analog IMC circuit, our system can achieve an accuracy comparable to a fully digital computation, as confirmed by the distance error as a function of the iteration number shown in Fig. 5d. Experimental results closely match both circuit simulations and fully digital IK implementations at both INT5 and FP64 precision, with algorithm convergence typically slowing down as precision progressively reduces from FP64 to experimental IMC.

Fig. 5e shows the end effector's full trajectory in the 3D space for an FP64 fully digital implementation and our experimental testchip. Owing to the reduced precision, IMC typically requires a higher mean number of iterations per trajectory point (Extended Data Fig. 6) with respect to fully digital IK. However, the higher number of iterations is compensated by the lower latency of the $\mathcal{O}(1)$ IMC-based computation, potentially achieving the same throughput of a high-performance FPGA-based accelerator (Extended Data Fig. 7c). On the other hand, thanks to the low-power operation, CL-IMC shows potential for $194 \times$ area and $66 \times$ energy efficiency improvement with respect to FPGA-based IK [83], supporting CL-IMC for both industrial automation as well as autonomous robots, where reduced energy consumption is paramount to extend battery life.

6 Conclusions

We present an integrated circuit for closed-loop IMC with two 64×64 40T5R memory arrays enclosed in a feedback loop by on-chip opamps, DACs, and ADCs. The on-chip solution of linear systems with an increasing number of equations up to 512×512 is experimentally demonstrated by implementing dedicated recursive block-inversion algorithms. As real-life toy problems of both academic and industrial interest, we provide experimental demonstrations of sounding rocket trajectory tracking by Kalman filter and acceleration of inverse kinematics in robotic arms. Experimental results closely match fully digital implementations at equivalent precision from the accuracy standpoint while simultaneously providing a consistent advantage in terms of energy efficiency with respect to GPU-, MCU-, and FPGA-based digital solvers. These results provide a tangible demonstration of closed-loop IMC for real-life problems, supporting CL-IMC as a strong candidate for the energy-efficient acceleration of inverse tasks in edge-processing scenarios.

7 Methods

7.1 Chip and memory array fabrication

The IC was designed in ST Microelectronics 90 nm CMOS technology around two memory arrays of 4096 devices each. Memory arrays were organized as matrices of 64 word lines (WLs) and 64 bit lines (BLs), with a 40T5R cell at each WL-BL intersection, where resistors were realized using lightly-doped n-well structures. WL/BL pairs can be connected to dedicated direct memory access (DMA) pins, allowing array characterization by an external circuit. In addition to memory devices, the chip integrates opamps, DACs, and ADCs. Opamps

were designed using a two-stage folded cascode topology with Miller compensation, achieving an open-loop gain of 66 dB and 20 MHz bandwidth. DACs were designed in a 7-bit R-2R topology, where an additional opamp buffered the generated analog value. The comparator-type ADC exploits an 8-bit shared staircase, which is fed as a reference to one comparator per output voltage. Each comparator is connected to an independent register, which latches the content of the staircase register when the corresponding output voltage falls below the staircase voltage. Communication to and from the chip is performed using an 8-bit parallel I/O bus, allowing access to on-chip configuration registers.

7.2 Experimental platform

The IC was interfaced to a hardware platform comprising a custom printed circuit board (PCB) hosting an STM32F429ZI microcontroller unit (MCU) with custom firmware (Extended Data Fig. 1a). The PCB features a serial-to-USB converter to allow interfacing with an external computer running MatLab. The firmware comprised an internal finite-state machine (FSM) allowing execution of pre-programmed routines for matrix programming, circuit reconfiguration, data transmission to the IC, and fetching of IMC results from the IC. Additionally, the PCB comprised an on-board characterization circuit (OBCC) schematically shown in Extended Data Fig. 1b, composed of a 10-bit test voltage generator, a high-side current sense circuit with programmable gain, and a 12-bit ADC, which can be connected to the IC DMA pins through an on-board switching matrix. All power supplies required for IC, MCU, and OBCC operation, namely 1.8 V, 0.9 V, 3.3 V, and 5 V, were generated from an external 7.5 V supply using low-dropout regulators.

7.3 CL-IMC testchip configurations

The proposed CL-IMC core can natively accelerate the IMVM operation and allow computation of a wider spectrum of linear algebra operations by exploiting the block-mapping technique [26]. Using block mapping, any linear matrix operation can be recast as an inverse problem on a suitably defined block matrix. Extended Data Fig. 2 schematically shows the main circuit configurations and block matrices for linear algebra operations used in this work, namely (a) positive-valued, positive-definite IMVM, (b) real-valued, positive-definite IMVM, (c) real-valued, indefinite IMVM, and (d) real-valued matrix-vector multiplication. Different configurations are obtained by corresponding programming of submatrices inside the memory array without additional rerouting or peripheral reconfiguration. Configurations shown in Extended Data Fig. 2a-b provide IMVM capability on positive-definite matrices with INT6 precision, *i.e.*, matrix elements can be mapped in the range $\{-31G_0, +31G_0\}$. Owing to stability limitations, IMVM on indefinite matrices requires exploiting a different mapping procedure for positive and negative weights, as shown in Extended Data Fig. 2c, corresponding to an INT5 matrix precision, *i.e.* matrix elements are mapped in the equivalent range $\{-15G_0, +15G_0\}$. The equivalent block-matrix equation for a $m \times m$ linear system $\mathbf{A}$ is:

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_{out} \\ \mathbf{v}_3 \end{bmatrix} = - \begin{bmatrix} \mathbf{I}_m + k\mathbf{J}_m & -\mathbf{A} + k\mathbf{J}_m & k\mathbf{j}_m \\ \mathbf{A}^T + k\mathbf{J}_m & k\mathbf{J}_m & k\mathbf{j}_m \\ k\mathbf{j}_m^T & k\mathbf{j}_m^T & k \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{i}_{in} \\ \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \quad (10)$$

where $\mathbf{I}_m$, $\mathbf{J}_m$, and $\mathbf{j}_m$ are the identity matrix, all-ones matrix, and all-ones vector of size m , respectively, and $k = 16G_0$. Finally, the IMVM core can be reprogrammed to act as an MVM accelerator by exploiting the block-matrix shown in Fig. 2d, allowing acceleration of MVM with INT6 precision. The equivalent block-matrix equation for a $m \times m$ matrix $\mathbf{A}$ reads:

$$\begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_{out} \end{bmatrix} = - \begin{bmatrix} \mathbf{I}_m & \mathbf{0}_m \\ (\mathbf{A}_+ - \mathbf{A}_-) & \mathbf{I}_m \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{i}_{in} \\ \mathbf{0} \end{bmatrix} \quad (11)$$

where $\mathbf{A}_+$, $\mathbf{A}_-$, and $\mathbf{0}_m$ are the positive and negative parts of $\mathbf{A}$ and the zero matrix of size m , respectively.

7.4 Adaptive precision IMVM algorithm

Extended Data Fig. 3 shows the main experimental routines used in this work, namely (a) on-chip inverse-matrix-vector multiplication, (b) baseline calibration, (c) linear system solution, and (d) linear correction calibration. After programming the memory array, on-chip IMVM (Extended Data Fig. 3a) is performed by first enabling the closed-loop opamps to engage the negative feedback. Input DACs are then enabled, providing input currents to the closed-loop circuit and allowing the system to stabilize into the steady-state solution by waiting for a predefined time t_{IMC} . To mitigate the impact of analog noise, outputs are sampled n_{avg} times and averaged using the onboard microcontroller floating-point unit. To mitigate the impact of opamp and DAC offsets, we calibrate the DAC baseline voltage (Extended Data Fig. 3b), *i.e.* the DAC output voltage corresponding to a '0' input, which nominally lies at $V_{DD}/2 = 0.9\text{ V}$, targeting minimization of the standard deviation of output voltages σ_v . To mitigate quantization error and maximize exploitation of the opamp dynamic range, we used the adaptive precision linear system solution algorithm shown in Extended Data Fig. 3c. The input vector $\mathbf{b}$ is first normalized with respect to its maximum absolute value f_b , and the quantization FSR is reset to 1. A quantized input vector $\mathbf{b}_q$ is obtained as $\mathbf{b}_q = Q(FSR \cdot \mathbf{b})$, where $Q(\cdot)$ is a 7-bit quantizer. To mitigate the impact of opamp and DAC offset, one of three different operation modes can be selected, namely bilinear removal (BILN), baseline removal (BSL), or no removal (NONE). In NONE mode, the quantized input vector $\mathbf{b}_q$ is transmitted to the IC and applied to the circuit, performing on-chip IMVM following the algorithm shown in Extended Data Fig. 3a. Output voltages are then compared with calibrated dynamic range boundaries. If no output voltage exceeds the boundaries, the current solution is stored, the FSR is increased by 1 unit, and the process is repeated. When an output voltage exceeds the boundaries, the current solution is discarded, the FSR increase process is halted, and the last stored solution, corresponding to the last valid FSR, is passed to the next step. In BILN mode, both $\mathbf{b}_q$ and $-\mathbf{b}_q$ are used as two separate inputs, providing the output voltage vector $\mathbf{v}_{out}$ and the baseline voltage vector $\mathbf{v}_{bsl}$. Both output voltage vectors are checked against boundaries, increasing FSR until one output in either of the two sets exceeds the useful dynamic range. Lastly, in BSL mode, vectors $\mathbf{b}_q$ and $\mathbf{0}$ are used as separate inputs to generate the $\mathbf{v}_{out}$ and $\mathbf{v}_{bsl}$ vectors. When the maximum possible FSR is reached, the output voltage is obtained by subtracting the baseline voltages from the output voltages, *i.e.* $\mathbf{v} = \mathbf{v}_{out} - \mathbf{v}_{bsl}$, in the MCU. The original scaling factor of the $\mathbf{b}$ vector f_b is restored at this stage, together with the mapping factor f_A for the programmed matrix, by a floating-point multiplication in the MCU. Lastly, a linear correction is applied to the obtained output vector in the MCU. The linear correction is calibrated each time a new matrix is programmed by performing trivial linear system solutions as shown in Extended Data Fig. 3d. Columns of the programmed matrix are used as input vectors, which must nominally provide columns of the identity matrix as output voltages. Measured output voltage vectors are used to form the output matrix $\mathbf{X}_t$, whose diagonal and out-of-diagonal elements should theoretically be equal to 1 and 0, respectively. Parameters a_{LC}, b_{LC} of the first-order linear correction can be estimated by averaging diagonal and out-of-diagonal values, respectively. One-time calibrated parameters are then reused for multiple linear system solutions on the same programmed matrix.

7.5 Recursive block inversion algorithm

Extended Data Fig. 4 schematically shows the main routines involved in in-memory recursive block-inversion, namely (a) in-memory block IMVM, (b) in-memory block MVM, and (c) in-memory Schur's complement computation. Starting from a linear system $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A}$ is larger than the available core size, the matrix and known term vector are 2×2 partitioned as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_3 \\ \mathbf{A}_2 & \mathbf{A}_4 \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}. \quad (12)$$

If the obtained submatrices are compatible with the core size, the solution vector $\mathbf{x}$ is then computed by sequential execution of (i) $\mathbf{y}_1 = \mathbf{A}_1^{-1}\mathbf{b}_1$, (ii) $\mathbf{c}_2 = \mathbf{A}_2\mathbf{y}_1$, (iii) $\mathbf{c}_3 = \mathbf{b}_2 - \mathbf{c}_2$, (iv) $\mathbf{x}_2 = \mathbf{A}_{4s}^{-1}\mathbf{c}_3$, (v) $\mathbf{c}_1 = \mathbf{A}_3\mathbf{x}_2$, (vi) $\mathbf{c}_4 = \mathbf{b}_1 - \mathbf{c}_1$, (vii) $\mathbf{x}_1 = \mathbf{A}_1^{-1}\mathbf{c}_4$, where $\mathbf{A}_{4s}$ is the Schur complement of $\mathbf{A}_1$. The obtained solution subvectors $\mathbf{x}_1, \mathbf{x}_2$ are finally reassembled to provide the solution vector $\mathbf{x}$. Notably, each IMVM and MVM operation is amenable to in-memory computation by suitable reconfiguration of our core, as discussed in Sec. 7.3. If the

obtained submatrices are not compatible with the core size, the IMVM algorithm is recursively applied, with each IMVM step undergoing matrix partitioning and block-inversion. Extended Data Fig. 4b shows a similar routine for block-MVM, where the original matrix $\mathbf{A}$ is similarly 2×2 partitioned, allowing computation of lower-dimensional MVMs in the IMC core. Block-IMVM and block-MVM routines are similarly exploited for the in-memory computation of the Schur complement required in step (iv) of the block-IMVM algorithm as shown in Extended Data Fig. 4c.

7.6 Power, performance, area estimation

Power, performance, and area (PPA) were estimated using SPICE simulations of the IC, where PPA characteristics of each analog and digital block were derived from either design values or post-layout simulations. Energy estimates comprise both static contributions due to leakage currents in each block as well as dynamic contributions from active operation energy. We considered non-essential circuitry to be turned off in each phase, *e.g.* power consumption of opamps, DACs, and ADCs were neglected during the memory array programming phase.

In our current 64×64 testchip in 90 nm technology, each array cell occupies an area of $442 \mu\text{m}^2$, with a tunable conductance range from $6.25 \mu\text{S}$ to $193.75 \mu\text{S}$. The conductance of each cell is programmed by writing the cell SRAM latches using two access transistors driven by a suitable addressing logic requiring $12.92 \mu\text{m}^2$ area and 5.4 nW leakage power. Each SRAM write requires 2.5 ns and 3.53 fJ dynamic energy. Each opamp occupies $6292 \mu\text{m}^2$ and requires $547.2 \mu\text{W}$ leakage power. For DACs, we obtain a $5000 \mu\text{m}^2$ area occupation and $130 \mu\text{W}$ leakage power. For ADCs, area contributions are provided by the common capacitive staircase generator ($295\,620 \mu\text{m}^2$) and line comparators and registers ($8196 \mu\text{m}^2$) for an equivalent per-line area occupation of $12\,810 \mu\text{m}^2$, with a total energy requirement of 28 nJ and latency of $51 \mu\text{s}$ for converting all 64 outputs. Finally, for the post-processing unit we reference the on-board STM32F429 microcontroller clocked at 180 MHz frequency, corresponding to an instruction latency and energy of 4.44 ns and 1.69 nJ , respectively, and a leakage power of 69.3 mW . The post-processing floating-point complexity is composed of N divisions (14 instruction cycles), N conversions (1 instruction cycle), N subtractions (1 instruction cycle), N multiplications (3 instruction cycles), and N multiply-accumulate operations (3 instruction cycles), totaling $22N$ instruction cycles. Notably, the microcontroller we used is not limited to digital signal processing but also features additional peripherals, which are unnecessary for our purposes. To estimate the effective area consumption of the computational logic, on-chip memory, and floating-point unit exploited for this work, we consider 10% of the WCLSP143 package area for the area occupation, corresponding to an overall area occupation of 2.74 mm^2 . For a 64×64 linear system solution, we obtain a typical total latency of $84 \mu\text{s}$ ($21 \mu\text{s}$ programming, $57 \mu\text{s}$ computing, $6 \mu\text{s}$ post-processing), total energy of $9.2 \mu\text{J}$ (146 pJ programming, $6.4 \mu\text{J}$ computing, $2.8 \mu\text{J}$ post-processing), and an area occupation of 8.5 mm^2 (5.75 mm^2 IMVM core, 2.75 mm^2 post-processing unit), corresponding to a 3.11 GOPS throughput, 28 GOPS/W energy efficiency and 0.37 GOPS/mm^2 area efficiency.

IMC performance can significantly benefit from technology downscaling through reduced area and power consumption at advanced process nodes. However, the predominantly analog nature of closed-loop IMC limits scalability at aggressively scaled technologies such as 7 nm or 5 nm , where reduced voltage headroom, increased device variability, and transistor sizing quantization critically impair analog performance and circuit reliability [84, 85]. Conversely, FinFET nodes at 14 nm and 16 nm have demonstrated consistent analog performance improvements over older planar technologies, including higher intrinsic gain and reduced noise levels [86, 87]. To assess the potential performance of closed-loop IMC at scaled nodes, we therefore performed projections at 14 nm technology, for a 256×256 macro with scaled-down array cells with area occupation of $5.35 \mu\text{m}^2$, with the same tunable conductance range from $6.25 \mu\text{S}$ to $193.75 \mu\text{S}$. For the programming logic, we consider a scaled-down version with area occupation of $0.32 \mu\text{m}^2$, 0.84 nW leakage power, and parallel programming capability with 1 ns latency [88]. For opamps, we refer to [42] for a 14-nm design with 46 dB gain, 500 MHz GBWP, $12 \mu\text{W}$ leakage power and $50 \mu\text{m}^2$ area occupation. For DACs, we refer to [89], corresponding to a 14-nm current-steering architecture with $1453 \mu\text{m}^2$ area occupation and $400 \mu\text{W}$ leakage power. Finally, we consider the 14-nm parallel ramp ADC reported in [90] with $5 \mu\text{m}^2$ area occupation, 8.78 pJ conversion energy and 128 ns latency. Finally, post-processing is performed by a dedicated floating-point unit with 0.4 ns instruction latency, $573 \mu\text{m}^2$ area occupation, $121 \mu\text{W}$ leakage power and 2.1 pJ dynamic energy [91]. We assume an optimized instruction set capable of executing floating-point division in 10 instruction cycles and fused multiply-add, addition, and type

conversion in 1 instruction cycle, corresponding to reducing the post-processing complexity to $14N$ instruction cycles. For a 256×256 linear system solution, we obtain a typical total latency of $2.8 \mu\text{s}$ (512 ns programming, 860 ns computing, $1.44 \mu\text{s}$ post-processing), total energy of 73.2 nJ (2.3 nJ programming, 63.2 nJ computing, 7.7 nJ post-processing), and an area occupation of 6.8 mm^2 (6.7 mm^2 IMVM core, $573 \mu\text{m}^2$ post-processing unit), corresponding to a 5.98 TOPS throughput, 228.6 TOPS/W energy efficiency and 0.9 TOPS/mm^2 area efficiency.

7.7 RBI benchmark

For recursive block inversion, we benchmarked the performance of our circuit and a commercial GPU, which are typically used to accelerate large-scale linear algebra computations [92–94]. Particularly, we estimate the performance of NVIDIA Jetson Xavier NX 8GB [71] in solving a 1024×1024 linear system. Notably, direct inversion methods such as Gaussian elimination and LU decomposition can not take advantage of the low-precision, high-throughput tensor cores and are typically run on floating-point cores with $\mathcal{O}(N^3)$ complexity. Iterative methods could exploit low-precision tensor cores optimized for matrix-vector multiplication but typically incur an increased complexity in practice ($\mathcal{O}(N^2K)$, where typically $K > N$) and do not allow decomposition reuse to speed up the serial solution of multiple linear systems. Nonetheless, we consider direct inversion at the lowest available INT8 precision, corresponding to a 2.62 TOPS throughput, 59.7 GB/s memory bandwidth, and 15 W thermal design power (TDP), assuming $N^3 = 1.074 \text{ GOPs}$ and $N^2 + 2N = 1 \text{ MB}$ data to be transferred to the memory. We obtain a total latency of $427 \mu\text{s}$ ($409 \mu\text{s}$ computing, $18 \mu\text{s}$ data transfer), estimate the energy consumption as 6.2 mJ (6.1 mJ for computing at full TDP, $26.4 \mu\text{J}$ for data transfer at 10% TDP) and refer to the 450 mm^2 die size as an area estimate, corresponding to an effective 2.512 TOPS throughput, 173.92 GOPS/W energy efficiency, and 7.18 GOPS/mm^2 area efficiency.

Notably, our current testchip features a single 64×64 IMVM core, thus requiring frequent reprogramming when executing RBI to solve multiple linear systems described by the same linear system matrix. To minimize programming operations, multiple identical cores can be colocated on the same chip and interconnected, allowing forwarding of partial results between cores. As such, for benchmarking purposes, we assumed multiple cores to be operated collectively under the RBI algorithm, with mixed parallel/serial operation depending on the RBI step. Owing to the partially serial nature of the RBI algorithm and the reduced computational complexity of MVM steps, we observe a slight reduction in the equivalent performance metrics, which can be recovered by performing multiple linear system solutions in a pipelined architecture. In this case, the Schur complement matrix can be computed once, and stored in on-chip cores to allow on-demand in-memory IMVM and MVM. For the 90 nm implementation, the effective throughput, energy efficiency, and area efficiency when implementing RBI on a 1024×1024 linear system are 1.93 GOPS , 5.98 GOPS/W , and 0.005 GOPS/mm^2 , respectively. For the 14 nm implementation, we instead obtain 3.47 TOPS , 107.18 TOPS/W , and 0.5 TOPS/mm^2 , respectively, as shown in Extended Data Fig. 7a.

7.8 KF benchmark

For Kalman filter, we identify microcontrollers as the more appropriate digital counterpart, owing to their widespread use in embedded systems and sensors with low-power constraints [77, 95–98]. We consider the computational complexity reported in [77] for an ST Microelectronics STM32H7R7 MCU [99], with a reported 0.78 ns instruction cycle, 412.5 mW leakage power, and 0.47 mm^2 area occupation for the core Cortex-M7 processor [100]. We estimate the area of the computational core, accounting also for partial utilization of the on-chip SRAM, flash memory, and clock generation circuitry as 10% of the smallest available package, namely WLCSP101, as 1.47 mm^2 . We obtain an equivalent throughput of 0.91 GOPS , energy efficiency of 3 GOPS/W , and area efficiency of 0.62 GOPS/mm^2 .

For our IMC testchip, we note that for the KF use case, the matrix size is known a-priori, allowing to tailor the circuit size in the design phase and avoid inclusion of unused memory blocks and circuit peripherals. As such, we benchmarked a reduced circuit of size 27×27 , coherently with the block matrix shown in Sec. 4. We obtain 0.3 GOPS , 5 GOPS/W , 0.076 GOPS/mm^2 throughput, energy and area efficiency for the 90 nm implementation, and 45.4 GOPS , 4.25 TOPS/W , 390 GOPS/mm^2 throughput, energy and area efficiency for the 14 nm implementation, respectively, as shown in Extended Data Fig. 7b. Notably, energy consumption benefits

from the sparsity of the Kalman filter block matrix, featuring many zero-valued cells and diagonal submatrices. While this also results in an area overhead, the latter may be eliminated in application-specific contexts by suitably avoiding inclusion of unused memory cells in Fig. 4c, at the cost of reduced circuit flexibility.

7.9 IK benchmark

For inverse kinematics, we benchmarked the performance of our circuit at a reduced 10×10 size as highlighted in Sec. 5, and a high-performance FPGA, which is typically employed in robotic scenarios thanks to their balance between high throughput and limited power consumption [83, 101, 102]. Particularly, we selected an AMD Artix 7 FPGA [103] clocked at 500 MHz. For the FPGA, we estimated the power and latency using the AMD Power Estimator (XPE) [104] and power consumption data reported in [83], considering a damped least-squares (DLS) IK implementation with values for look-up tables (LUTs), flip-flops (FFs), digital signal processing (DSP) slices, and block RAM (BRAM) extracted from [83], corresponding to 62% of the available logic elements [105, 106]. The latter typically occupy 30% of the die area [107], allowing to estimate the area occupation as 20% of the reported die size, corresponding to 31 mm^2 [108]. We obtain an equivalent throughput, energy, and area efficiency of 9 GOPS, 34.46 GOPS/W, and 0.3 GOPS/mm^2 , respectively.

Similarly to the KF case, the a-priori knowledge of the IK algorithm allows tailoring of IMC circuit size at design time, enabling pruning of unnecessary memory blocks and peripherals. We therefore compare digital benchmarks with values for a 10×10 CL-IMC circuit under a $0.75 \times 10^3 = 0.75 \text{ kOPs}$ complexity, accounting for the typically increased number of iterations, obtaining 0.014 GOPS, 0.52 GOPS/W, and 0.004 GOPS/mm^2 for the 90 nm implementation, and 3 GOPS, 0.6 TOPS/W, and 116 GOPS/mm^2 for the 14 nm implementation, as shown in Extended Data Fig. 7c.

8 Supplementary information

8.1 In-memory DE solution

The time dynamics of a prototypical CL-IMC circuit composed of a memory array encoding an $N \times N$ matrix $\mathbf{X}$, enclosed in a negative feedback loop by N opamps with gain α_0 , intrinsic time constant τ_0 , and subject to an $N \times 1$ vector of input currents $\mathbf{i}$ is described by the associated system of differential equations [26]:

$$\tau_0 \frac{d\mathbf{v}(t)}{dt} = -(\alpha_0 \hat{\mathbf{X}} + \mathbf{I})\mathbf{v}(t) - \alpha_0 \mathbf{U}^{-1} \mathbf{i}(t), \quad (13)$$

where $\mathbf{U}$ is the diagonal matrix of row-sums of $\mathbf{X}$, $\hat{\mathbf{X}} = \mathbf{U}^{-1} \mathbf{X}$ is the feedback voltage divider matrix, and $\mathbf{v}(t)$ is the time-dependent $N \times 1$ vector of output voltages. The corresponding finite differences formulation can be written as:

$$\tau_0 \frac{\mathbf{v}[k] - \mathbf{v}[k-1]}{\Delta t} = -(\alpha_0 \hat{\mathbf{X}} + \mathbf{I})\mathbf{v}[k] - \alpha_0 \mathbf{U}^{-1} \mathbf{i}[k], \quad (14)$$

where Δt is the discretization step of the time domain. Defining $\mathbf{A} = -\frac{1}{\tau_0}(\alpha_0 \hat{\mathbf{X}} + \mathbf{I})$ and $\mathbf{B} = -\alpha_0 \mathbf{U}^{-1}$, Eq. (14) can be compactly rewritten in recursive form as:

$$(\mathbf{I} - \Delta t \cdot \mathbf{A})\mathbf{v}[k] = \mathbf{v}[k-1] + \mathbf{B} \cdot \mathbf{i}[k]. \quad (15)$$

Unrolling the recursive relation yields a single high-dimensional linear system of the form:

$$\begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{I} & (\mathbf{I} - \Delta t \cdot \mathbf{A}) & \mathbf{0} & \dots & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & (\mathbf{I} - \Delta t \cdot \mathbf{A}) & \dots & \mathbf{0} & \mathbf{0} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & (\mathbf{I} - \Delta t \cdot \mathbf{A}) & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{I} & (\mathbf{I} - \Delta t \cdot \mathbf{A}) \end{bmatrix} \begin{bmatrix} \mathbf{v}[0] \\ \mathbf{v}[1] \\ \mathbf{v}[2] \\ \vdots \\ \mathbf{v}[K-1] \\ \mathbf{v}[K] \end{bmatrix} = \begin{bmatrix} \mathbf{B} \cdot \mathbf{i}[0] \\ \mathbf{B} \cdot \mathbf{i}[1] \\ \mathbf{B} \cdot \mathbf{i}[2] \\ \vdots \\ \mathbf{B} \cdot \mathbf{i}[K-1] \\ \mathbf{B} \cdot \mathbf{i}[K] \end{bmatrix}, \quad (16)$$

where the first equation encodes the initial condition $\mathbf{v}[0] = \mathbf{0}$, K is the number of time-steps, and the linear system size is $KN \times KN$.

8.2 In-memory Kalman Filter circuit

Both the prediction and correction steps of a prototypical Kalman filter can be accelerated through dedicated in-memory circuits. The corresponding circuit topology is shown in Extended Data Fig. 5a, consisting of three main stages, namely (a) the prediction stage (blue), (b) the measurement stage (green), and (c) the Kalman gain stage (brown). The prediction stage operates as an MVM accelerator with respect to the previous *a-posteriori* estimate, which is fed to the circuit as an input voltage vector $\mathbf{v}_{\text{in}} = \boldsymbol{\theta}_{(n-1|n-1)}$. Correspondingly, the prediction stage output reads:

$$\mathbf{v}_{\text{out,p}} = -\frac{\mathbf{A}}{f_0} \mathbf{v}_{\text{in}}, \quad (17)$$

representing the *a-priori* state estimation $\boldsymbol{\theta}_{(n|n-1)}$ when $f_0 = 1$. The measurement stage prepares the estimation error by multiplying the *a-priori* estimate to the observation model matrix $\mathbf{B}$, and combining the output current with measurement data, fed as an input current vector $\mathbf{i}_1 = \mathbf{x}_n$, such that the overall output current reads:

$$\mathbf{i}_\epsilon = \mathbf{i}_1 - \frac{\mathbf{B}\mathbf{A}}{f_0} \mathbf{v}_{\text{in}}, \quad (18)$$

corresponding to the estimation error $\boldsymbol{\epsilon}_n$. Lastly, the current output from the measurement stage is fed to the Kalman gain stage, consisting of four memory arrays storing replicas of the observation model $\mathbf{B}$, the *a-posteriori* state covariance $\mathbf{C}_{\boldsymbol{\theta}(n-1|n-1)}$, and the measurement uncertainty covariance $\mathbf{C}_{\boldsymbol{\beta}\boldsymbol{\beta}}$, and three inverting

OA sets closing the feedback loop. Particularly, two OA sets are used in TIA configuration, with local feedback conductances f_1, f_2 . The output voltages then read:

$$\mathbf{v}_{\text{out,c}} = -\mathbf{C}_\theta \mathbf{B}^T \left(\frac{\mathbf{B} \mathbf{C}_\theta \mathbf{B}^T}{f_1 f_2} + \mathbf{C}_{\beta\beta} \right)^{-1} \mathbf{i}_\varepsilon, \quad (19)$$

thus matching Eq. (6) for $f_1 f_2 = 1$. The *a-posteriori* estimate is finally obtained by summing the prediction and correction outputs through either an additional analog stage or inside a digital signal processor with $\mathcal{O}(p)$ complexity, where p is the number of state variables.

8.3 Constant-velocity 3D motion model

Consider an object moving in a three-dimensional space with position and velocity vectors at the k -th timestep:

$$\mathbf{r}^k = \begin{bmatrix} r_x^k \\ r_y^k \\ r_z^k \end{bmatrix} \quad \mathbf{v}^k = \begin{bmatrix} v_x^k \\ v_y^k \\ v_z^k \end{bmatrix}. \quad (20)$$

If the velocities are assumed constant in every direction, then at the k -th timestep:

$$\begin{cases} r_x^{k+1} = r_x^k + v_x^k \cdot \Delta t \\ v_x^{k+1} = v_x^k \\ r_y^{k+1} = r_y^k + v_y^k \cdot \Delta t \\ v_y^{k+1} = v_y^k \\ r_z^{k+1} = r_z^k + v_z^k \cdot \Delta t \\ v_z^{k+1} = v_z^k \end{cases} \quad (21)$$

where Δt is the timestep width. Rewriting the latter system in matrix notation yields:

$$\boldsymbol{\theta}^{k+1} = \begin{bmatrix} \mathbf{r}^{k+1} \\ \mathbf{v}^{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{r}^k \\ \mathbf{v}^k \end{bmatrix} = \mathbf{A} \boldsymbol{\theta}^k \quad (22)$$

where $\mathbf{A}$, $\boldsymbol{\theta}$ are thus the physical model matrix and state vector of the Kalman filter implementation, respectively. Since a constant-velocity was assumed, any unwanted accelerations, *e.g.* caused by uncontrolled forces, will cause the system state to deviate from the ideal behavior described by $\mathbf{A}$. Accounting for these inaccuracies, the system state will read:

$$\boldsymbol{\theta}^{k+1} = \mathbf{A} \boldsymbol{\theta}^k + \boldsymbol{\alpha}^k \quad (23)$$

where $\boldsymbol{\alpha}^k$ is the *prediction noise*, with $\boldsymbol{\alpha}^k \sim \mathcal{N}(0, \mathbf{C}_{\alpha\alpha})$.

Consequently, if one assumes to sample the xyz position of the object at the k -th timestep, the measurement vector is given by:

$$\mathbf{x}^k = \begin{bmatrix} r_x^k \\ r_y^k \\ r_z^k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{r}^k \\ \mathbf{v}^k \end{bmatrix} = \mathbf{B} \boldsymbol{\theta}^k \quad (24)$$

where $\mathbf{B}$ is thus the measurement model matrix of the Kalman filter implementation. In the presence of perturbations, *e.g.* sensor inaccuracy, the measurement vector is corrupted as:

$$\mathbf{x}^k = \mathbf{B} \boldsymbol{\theta}^k + \boldsymbol{\beta}^k \quad (25)$$

557 where $\boldsymbol{\beta}^k$ is the *measurement noise*, with $\boldsymbol{\beta}^k \sim \mathcal{N}(0, \mathbf{C}_{\boldsymbol{\beta}\boldsymbol{\beta}})$. Covariance matrices $\mathbf{C}_{\alpha\alpha}$, $\mathbf{C}_{\boldsymbol{\beta}\boldsymbol{\beta}}$ can then be tuned
558 to the accuracy of the prediction and measurement setup, respectively.

References

- [1] von Neumann, J. First draft of a report on the EDVAC. *IEEE Annals of the History of Computing* **15**, 27–75 (1993). <https://doi.org/10.1109/85.238389>.
- [2] Lee, D. U. *et al.*, Lee, Dong Uk and Kim, Kyung Whan and Kim, Kwan Weon and Kim, Hongjung and Kim, Ju Young and Park, Young Jun and Kim, Jae Hwan and Kim, Dae Suk and Park, Heat Bit and Shin, Jin Wook and Cho, Jang Hwan and Kwon, Ki Hun and Kim, Min Jeong and Lee, Jaejin and Park, Kun Woo and Chung, Byongtae and Hong, Sungjoo (2014). 25.2 A 1.2V 8Gb 8-channel 128GB/s high-bandwidth memory (HBM) stacked DRAM with effective microbump I/O test methods using 29nm process and TSV. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (IEEE, San Francisco, CA, USA, 2014), 432–433, <https://doi.org/10.1109/ISSCC.2014.6757501>.
- [3] Mathur, R. *et al.*, Mathur, R. and Sisodia, R. and Chen, A. and Singh, A. and Thyagarajan, S. and Cubeta, A. and Andrieux, C. and Sowden, A. and Chong, Y. (2024). A 7GHz High-Bandwidth 1R-1RW SRAM for Arm HPC Processor in 3nm Technology. In *2024 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)* (IEEE, Honolulu, HI, USA, 2024), 1–2, <https://doi.org/10.1109/VLSITechnologyandCir46783.2024.10631403>.
- [4] Chen, Y.-H., Yang, T.-J., Emer, J. S. & Sze, V. Eyeriss v2: A Flexible Accelerator for Emerging Deep Neural Networks on Mobile Devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **9**, 292–308 (2019). <https://doi.org/10.1109/JETCAS.2019.2910232>.
- [5] Bankman, D., Yang, L., Moons, B., Verhelst, M. & Murmann, B. An Always-On 3.8 μ J/86% CIFAR-10 Mixed-Signal Binary CNN Processor With All Memory on Chip in 28-nm CMOS. *IEEE Journal of Solid-State Circuits* **54**, 158–172 (2019). <https://doi.org/10.1109/JSSC.2018.2869150>.
- [6] Geens, R., Shi, M., Symons, A., Fang, C. & Verhelst, M., Geens, Robin and Shi, Man and Symons, Arne and Fang, Chao and Verhelst, Marian (2024). Energy Cost Modelling for Optimizing Large Language Model Inference on Hardware Accelerators. In *2024 IEEE 37th International System-on-Chip Conference (SOCC)* (IEEE, Dresden, Germany, 2024), 1–6, <https://doi.org/10.1109/SOCC62300.2024.10737844>.
- [7] Horowitz, M., Horowitz, Mark (2014). 1.1 Computing’s energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (IEEE, San Francisco, CA, USA, 2014), 10–14, <https://doi.org/10.1109/ISSCC.2014.6757323>.
- [8] Ielmini, D. & Wong, H.-S. P. In-memory computing with resistive switching devices. *Nature Electronics* **1**, 333–343 (2018). <https://doi.org/10.1038/s41928-018-0092-2>.
- [9] Mittal, S., Verma, G., Kaushik, B. & Khanday, F. A. A survey of SRAM-based in-memory computing techniques and applications. *Journal of Systems Architecture* **119**, 102276 (2021). <https://doi.org/10.1016/j.sysarc.2021.102276>.
- [10] Sebastian, A., Le Gallo, M., Khaddam-Aljameh, R. & Eleftheriou, E. Memory devices and applications for in-memory computing. *Nature Nanotechnology* (2020). <https://doi.org/10.1038/s41565-020-0655-z>.
- [11] Zidan, M. A., Strachan, J. P. & Lu, W. D. The future of electronics based on memristive systems. *Nature Electronics* **1**, 22–29 (2018). <https://doi.org/10.1038/s41928-017-0006-8>.
- [12] Mannocci, P. *et al.* In-memory computing with emerging memory devices: Status and outlook. *APL Machine Learning* **1**, 010902 (2023). <https://doi.org/10.1063/5.0136403>.
- [13] Lepri, N. *et al.* In-Memory Computing for Machine Learning and Deep Learning. *IEEE Journal of the Electron Devices Society* **11**, 587–601 (2023). <https://doi.org/10.1109/JEDS.2023.3265875>.

- [14] Krishnan, G. *et al.* SIAM: Chiplet-based Scalable In-Memory Acceleration with Mesh for Deep Neural Networks. *ACM Transactions on Embedded Computing Systems* **20**, 1–24 (2021). <https://doi.org/10.1145/3476999>.
- [15] Wan, W. *et al.* A compute-in-memory chip based on resistive random-access memory. *Nature* **608**, 504–512 (2022). <https://doi.org/10.1038/s41586-022-04992-8>.
- [16] Le Gallo, M. *et al.* A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nature Electronics* **6**, 680–693 (2023). <https://doi.org/10.1038/s41928-023-01010-1>.
- [17] Li, C. *et al.* Analogue signal and image processing with large memristor crossbars. *Nature Electronics* **1**, 52–59 (2018). <https://doi.org/10.1038/s41928-017-0002-z>.
- [18] Song, W. *et al.* Programming memristor arrays with arbitrarily high precision for analog computing. *Science* **383**, 903–910 (2024). <https://doi.org/10.1126/science.adi9405>.
- [19] Cai, F. *et al.* Power-efficient combinatorial optimization using intrinsic noise in memristor Hopfield neural networks. *Nature Electronics* **3**, 409–418 (2020). <https://doi.org/10.1038/s41928-020-0436-6>.
- [20] Bhattacharya, T. *et al.* Computing high-degree polynomial gradients in memory. *Nature Communications* **15**, 8211 (2024). <https://doi.org/10.1038/s41467-024-52488-y>.
- [21] Sun, Z. & Huang, R. Time Complexity of In-Memory Matrix-Vector Multiplication. *IEEE Transactions on Circuits and Systems II: Express Briefs* **68**, 2785–2789 (2021). <https://doi.org/10.1109/TCSII.2021.3068764>.
- [22] Horn, R. A. & Johnson, C. R. *Matrix analysis* 2nd ed edn (Cambridge University Press, Cambridge ; New York, 2012).
- [23] Sun, Z. *et al.* Solving matrix equations in one step with cross-point resistive arrays. *Proceedings of the National Academy of Sciences* **116**, 4123–4128 (2019). <https://doi.org/10.1073/pnas.1815682116>.
- [24] Golub, G. H. & Van Loan, C. F. *Matrix computations* (The Johns Hopkins University Press, Baltimore, 2013).
- [25] Sun, Z. *et al.* Time Complexity of In-Memory Solution of Linear Systems. *IEEE Transactions on Electron Devices* **67**, 2945–2951 (2020). <https://doi.org/10.1109/TED.2020.2992435>.
- [26] Mannocci, P. & Ielmini, D. A Generalized Block-Matrix Circuit for Closed-Loop Analog In-Memory Computing. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* **9**, 47–55 (2023). <https://doi.org/10.1109/JXCDC.2023.3265803>.
- [27] Mannocci, P., Melacarne, E. & Ielmini, D. An Analogue In-Memory Ridge Regression Circuit With Application to Massive MIMO Acceleration. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **12**, 952–962 (2022). <https://doi.org/10.1109/JETCAS.2022.3221284>.
- [28] Zeng, Q. *et al.* Realizing In-Memory Baseband Processing for Ultrafast and Energy-Efficient 6G. *IEEE Internet of Things Journal* **11**, 5169–5183 (2024). <https://doi.org/10.1109/JIOT.2023.3307405>.
- [29] Wang, S. *et al.* In-memory analog solution of compressed sensing recovery in one step. *Science Advances* **9**, eadj2908 (2023). <https://doi.org/10.1126/sciadv.adj2908>.

- [30] Chen, Z. *et al.*, Chen, Zirui and Zhou, Yue and Xu, Hanxi and Fu, Yaoyao and Li, Yi and He, Y. and Miao, X.-S. and Mannocci, Piergiulio and Ielmini, Daniele (2023). An Ultrafast (< 200 ns) Sparse Solution Solver made by $\text{HfWO}_x/\text{VO}_y$ Threshold Tunable Neurons. In *2023 International Electron Devices Meeting (IEDM)* (IEEE, San Francisco, CA, USA, 2023), 1–4, <https://doi.org/10.1109/IEDM45741.2023.10413700>.
- [31] Mannocci, P. *et al.*, Mannocci, P. and Melacarne, E. and Pezzoli, A. and Pedretti, G. and Villa, C. and Sancandi, F. and Spagnolini, U. and Ielmini, D. (2023). An SRAM-based reconfigurable analog in-memory computing circuit for solving linear algebra problems. In *2023 International Electron Devices Meeting (IEDM)* (IEEE, San Francisco, CA, USA, 2023), 1–4, <https://doi.org/10.1109/IEDM45741.2023.10413724>.
- [32] Sun, Z., Ambrosi, E., Pedretti, G., Bricalli, A. & Ielmini, D. In-Memory PageRank Accelerator With a Cross-Point Array of Resistive Memories. *IEEE Transactions on Electron Devices* **67**, 1466–1470 (2020). <https://doi.org/10.1109/TED.2020.2966908>.
- [33] Sun, Z., Pedretti, G., Ambrosi, E., Bricalli, A. & Ielmini, D. In-Memory Eigenvector Computation in Time $O(1)$. *Advanced Intelligent Systems* **2**, 2000042 (2020). <https://doi.org/10.1002/aisy.202000042>.
- [34] Korkmaz, A. *et al.* Spectral Ranking in Complex Networks Using Memristor Crossbars. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **13**, 357–370 (2023). <https://doi.org/10.1109/JETCAS.2023.3237836>.
- [35] Sun, Z., Pedretti, G., Bricalli, A. & Ielmini, D. One-step regression and classification with cross-point resistive memory arrays. *Science Advances* **6**, eaay2378 (2020). <https://doi.org/10.1126/sciadv.aay2378>.
- [36] Chen, X. & Han, Y., Chen, Xiaoming and Han, Yinhe (2022). Solving Least-Squares Fitting in $\$O(1)\$$ Using RRAM-based Computing-in-Memory Technique. In *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)* (IEEE, Taipei, Taiwan, 2022), 339–344, <https://doi.org/10.1109/ASP-DAC52403.2022.9712568>.
- [37] Zoppo, G. *et al.*, Zoppo, Gianluca and Korkmaz, Anil and Marrone, Francesco and Yi, Su-in and Palermo, Samuel and Corinto, Fernando and Williams, R. Stanley (2023). Gaussian Process for Nonlinear Regression via Memristive Crossbars. In *2023 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, Monterey, CA, USA, 2023), 1–5, <https://doi.org/10.1109/ISCAS46773.2023.10181785>.
- [38] Mannocci, P., Giannone, E. & Ielmini, D. In-Memory Principal Component Analysis by Analogue Closed-Loop Eigendecomposition. *IEEE Transactions on Circuits and Systems II: Express Briefs* **71**, 1839–1843 (2024). <https://doi.org/10.1109/TCSII.2023.3334958>.
- [39] Bao, H. *et al.* Energy Efficient Memristive Transiently Chaotic Neural Network for Combinatorial Optimization. *IEEE Transactions on Circuits and Systems I: Regular Papers* **71**, 3708–3716 (2024). <https://doi.org/10.1109/TCSI.2024.3406167>.
- [40] Shang, L., Adil, M., Madani, R. & Pan, C. Memristor-Based Analog Recursive Computation Circuit for Linear Programming Optimization. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* **6**, 53–61 (2020). <https://doi.org/10.1109/JXCDC.2020.2995123>.
- [41] Mannocci, P. *et al.* A Universal, Analog, In-Memory Computing Primitive for Linear Algebra Using Memristors. *IEEE Transactions on Circuits and Systems I: Regular Papers* **68**, 4889–4899 (2021). <https://doi.org/10.1109/TCSI.2021.3122278>.
- [42] Feinberg, B. *et al.*, Feinberg, Ben and Wong, Ryan and Xiao, T. Patrick and Bennett, Christopher H. and Rohan, Jacob N. and Boman, Erik G. and Marinella, Matthew J. and Agarwal, Sapan and Ipek, Engin

- (2021). An Analog Preconditioner for Solving Linear Systems. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (IEEE, Seoul, Korea (South), 2021), 761–774, <https://doi.org/10.1109/HPCA51647.2021.00069>.
- [43] Korkmaz, A. *et al.*, Korkmaz, Anil and Zoppo, Gianluca and Marrone, Francesco and Corinto, Fernando and Williams, R. Stanley and Palermo, Samuel (2022). Analog Acceleration of the Power Method using Memristor Crossbars. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)* (IEEE, Austin, TX, USA, 2022), 1194–1198, <https://doi.org/10.1109/ISCAS48785.2022.9937543>.
- [44] Lin, J., Barrows, F. & Caravelli, F. Memristive Linear Algebra (2024). URL <https://arxiv.org/abs/2407.20539>. Version Number: 1.
- [45] Kalantzis, V. *et al.*, Kalantzis, Vasileios and Gupta, Anshul and Horesh, Lior and Nowicki, Tomasz and Squillante, Mark S. and Wah Wu, Chai and Gokmen, Tayfun and Avron, Haim (2021). Solving sparse linear systems with approximate inverse preconditioners on analog devices. In *2021 IEEE High Performance Extreme Computing Conference (HPEC)* (IEEE, Waltham, MA, USA, 2021), 1–7, <https://doi.org/10.1109/HPEC49654.2021.9622816>.
- [46] Boniardi, M., Baldo, M., Allegra, M. & Redaelli, A. Phase Change Memory: A Review on Electrical Behavior and Use in Analog In-Memory-Computing (A-IMC) Applications. *Advanced Electronic Materials* 2400599 (2024). <https://doi.org/10.1002/aelm.202400599>.
- [47] Mehonic, A. *et al.* Roadmap to neuromorphic computing with emerging technologies. *APL Materials* **12**, 109201 (2024). <https://doi.org/10.1063/5.0179424>.
- [48] Verma, N. *et al.* In-Memory Computing: Advances and Prospects. *IEEE Solid-State Circuits Magazine* **11**, 43–55 (2019). <https://doi.org/10.1109/MSSC.2019.2922889>.
- [49] Ielmini, D. & Pedretti, G. Device and Circuit Architectures for In-Memory Computing. *Advanced Intelligent Systems* **2**, 2000040 (2020). <https://doi.org/10.1002/aisy.202000040>.
- [50] Milo, V. *et al.* Accurate Program/Verify Schemes of Resistive Switching Memory (RRAM) for In-Memory Neural Network Circuits. *IEEE Transactions on Electron Devices* **68**, 3832–3837 (2021). <https://doi.org/10.1109/TED.2021.3089995>.
- [51] Sangwan, V. K. *et al.* Multi-terminal memtransistors from polycrystalline monolayer molybdenum disulfide. *Nature* **554**, 500–504 (2018). <https://doi.org/10.1038/nature25747>.
- [52] Pistolesi, L. *et al.*, Pistolesi, L. and Glukhov, A. and De Gracia Herranz, A. and Lopez-Vallejo, M. and Carissimi, M. and Pasotti, M. and Rolandi, P. and Redaelli, A. and Martín, I. Muñoz and Bianchi, S. and Bonfanti, A. and Ielmini, D. (2024). Drift Compensation in Multilevel PCM for in-Memory Computing Accelerators. In *2024 IEEE International Reliability Physics Symposium (IRPS)* (IEEE, Grapevine, TX, USA, 2024), 1–4, <https://doi.org/10.1109/IRPS48228.2024.10529438>.
- [53] Zoppo, G. *et al.* Analog Solutions of Discrete Markov Chains via Memristor Crossbars. *IEEE Transactions on Circuits and Systems I: Regular Papers* **68**, 4910–4923 (2021). <https://doi.org/10.1109/TCSI.2021.3126477>.
- [54] Chataut, R. & Akl, R. Massive MIMO Systems for 5G and beyond Networks—Overview, Recent Trends, Challenges, and Future Research Direction. *Sensors* **20**, 2753 (2020). <https://doi.org/10.3390/s20102753>.
- [55] Jolliffe, I. in *Principal Component Analysis* In Everitt, B. S. & Howell, D. C. (eds) *Encyclopedia of Statistics in Behavioral Science* bsa501 (John Wiley & Sons, Ltd, Chichester, UK, 2005). URL <https://onlinelibrary.wiley.com/doi/10.1002/0470013192.bsa501>.

- [56] Tisseur, F. & Dongarra, J. A Parallel Divide and Conquer Algorithm for the Symmetric Eigenvalue Problem on Distributed Memory Architectures. *SIAM Journal on Scientific Computing* **20**, 2223–2236 (1999). <https://doi.org/10.1137/S1064827598336951>.
- [57] Pan, L., Zuo, P., Luo, Y., Sun, Z. & Huang, R., Pan, Lunshuai and Zuo, Pushen and Luo, Yubiao and Sun, Zhong and Huang, Ru (2024). BlockAMC: Scalable In-Memory Analog Matrix Computing for Solving Linear Systems. In *2024 Design, Automation & Test in Europe Conference & Exhibition (DATE)* (IEEE, Valencia, Spain, 2024), 1–6, <https://doi.org/10.23919/DATE58400.2024.10546501>.
- [58] Bridarolli, D. *et al.*, Bridarolli, D. and Zucchelli, C. and Mannocci, P. and Ricci, S. and Farronato, M. and Pedretti, G. and Sun, Z. and Ielmini, D. (2024). High-Density Multilevel 3D Vertical Resistive Switching Memory (VRRAM) for Massively Parallel in-Memory Computing. In *2024 IEEE International Electron Devices Meeting (IEDM)* (IEEE, San Francisco, CA, USA, 2024), 1–4, <https://doi.org/10.1109/IEDM50854.2024.10873537>.
- [59] Zhang, F. (ed.) *The Schur Complement and Its Applications* Vol. 4 of *Numerical Methods and Algorithms* (Springer-Verlag, New York, 2005). URL <http://link.springer.com/10.1007/b105056>.
- [60] Lepri, N., Glukhov, A. & Ielmini, D., Lepri, Nicola and Glukhov, Artem and Ielmini, Daniele (2022). Mitigating read-program variation and IR drop by circuit architecture in RRAM-based neural network accelerators. In *2022 IEEE International Reliability Physics Symposium (IRPS)* (IEEE, Dallas, TX, USA, 2022), 3C.2–1–3C.2–6, <https://doi.org/10.1109/IRPS48227.2022.9764486>.
- [61] Higham, N. Iterative refinement for linear systems and LAPACK. *IMA Journal of Numerical Analysis* **17**, 495–509 (1997). <https://doi.org/10.1093/imanum/17.4.495>.
- [62] Le Gallo, M. *et al.* Mixed-Precision In-Memory Computing. *Nature Electronics* **1**, 246–253 (2018). <https://doi.org/10.1038/s41928-018-0054-8>.
- [63] Evans, L. C. *Partial differential equations* Second edition edn. No. 19 in Graduate studies in mathematics (American Mathematical Society, Providence, Rhode Island, 2022).
- [64] Ding, Z., Li, X. & Lin, L. Simulating Open Quantum Systems Using Hamiltonian Simulations. *PRX Quantum* **5**, 020332 (2024). <https://doi.org/10.1103/PRXQuantum.5.020332>.
- [65] Sato, Y., Kondo, R., Hamamura, I., Onodera, T. & Yamamoto, N. Hamiltonian simulation for hyperbolic partial differential equations by scalable quantum circuits. *Physical Review Research* **6**, 033246 (2024). <https://doi.org/10.1103/PhysRevResearch.6.033246>.
- [66] Guo, Z. *et al.* Diffusion models in bioinformatics and computational biology. *Nature Reviews Bioengineering* **2**, 136–154 (2023). <https://doi.org/10.1038/s44222-023-00114-9>.
- [67] Selvadurai, A. P. S. *Partial Differential Equations in Mechanics 1: Fundamentals, Laplace’s Equation, Diffusion Equation, Wave Equation* (Springer, Berlin Heidelberg, 2000).
- [68] Dhatt, G., Touzot, G. & Lefrançois, E. *Finite Element Method* 1 edn (Wiley, 2012). URL <https://onlinelibrary.wiley.com/doi/book/10.1002/9781118569764>.
- [69] Januszewski, M. & Kostur, M. Accelerating numerical solution of stochastic differential equations with CUDA. *Computer Physics Communications* **181**, 183–188 (2010). <https://doi.org/10.1016/j.cpc.2009.09.009>.
- [70] Wu, C. W., Squillante, M. S., Kalantzis, V. & Horesh, L. Stable iterative refinement algorithms for solving linear systems (2023). URL <https://arxiv.org/abs/2309.07865>. Version Number: 2.

- [71] NVIDIA Jetson Xavier (2024). URL <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/>.
- [72] Spagnolini, U. *Statistical signal processing in engineering* (Wiley, Hoboken, 2018).
- [73] Zarchan, P. & Musoff, H. *Fundamentals of Kalman filtering: a practical approach* Progress in astronautics and aeronautics (American Institute of Aeronautics and Astronautics, Reston, Va. [Great Britain], 2000).
- [74] Kalita, D. & Lyakhov, P. Moving Object Detection Based on a Combination of Kalman Filter and Median Filtering. *Big Data and Cognitive Computing* **6**, 142 (2022). <https://doi.org/10.3390/bdcc6040142>.
- [75] Li, W. *et al.* A Survey on Multisensor Fusion and Consensus Filtering for Sensor Networks. *Discrete Dynamics in Nature and Society* **2015**, 1–12 (2015). <https://doi.org/10.1155/2015/683701>.
- [76] Ghysels, E. & Marcellino, M. *Applied economic forecasting using time series methods* (Oxford University Press, New York, NY, 2018).
- [77] Büyükeşmeli, H., Masazade, E. & Ünsalan, C. Probabilistic object tracking by low power micro-controllers. *Journal of Real-Time Image Processing* **18**, 2539–2550 (2021). <https://doi.org/10.1007/s11554-021-01139-2>.
- [78] Madjarov, N. & Mihaylova, L., Madjarov, N. and Mihaylova, L. (1993). Sensitivity analysis of linear optimal stochastic observers. In *Proceedings of IEEE Systems Man and Cybernetics Conference - SMC* (IEEE, Le Touquet, France, 1993), 482–487, <https://doi.org/10.1109/ICSMC.1993.384919>.
- [79] Saab, S. S. & Nasr, G. E. Sensitivity of discrete-time Kalman filter to statistical modeling errors. *Optimal Control Applications and Methods* **20**, 249–259 (1999). [https://doi.org/10.1002/\(SICI\)1099-1514\(199909/10\)20:5<249::AID-OCA659>3.0.CO;2-2](https://doi.org/10.1002/(SICI)1099-1514(199909/10)20:5<249::AID-OCA659>3.0.CO;2-2).
- [80] Skyward, E. R. Gemini Flight Logs (2023). URL <https://git.skywarder.eu/skyward/flight-logs/gemini-flight-logs>.
- [81] Slawecki, P., Slonka, P., Zebrowski, W. & Wesolowska, Z., Slawecki, Piotr and Slonka, Piotr and Zebrowski, Wojciech and Wesolowska, Zuzanna (2024). Development of a distributed embedded apogee prediction and control system with sensor fusion for sounding rockets. In *AIAA SCITECH 2024 Forum* (American Institute of Aeronautics and Astronautics, Orlando, FL, 2024), <https://doi.org/10.2514/6.2024-0963>.
- [82] Cubero, S. (ed.) *Industrial robotics: theory, modelling and control* ARS, Advanced robotic systems international (plV, pro-Literatur-Verl, Mammendorf, 2007).
- [83] Sau, C. *et al.* Feasibility Study and Porting of the Damped Least Square Algorithm on FPGA. *IEEE Access* **8**, 175483–175500 (2020). <https://doi.org/10.1109/ACCESS.2020.3025367>.
- [84] Loke, A. L. S. *et al.*, Loke, Alvin L. S. and Yang, Da and Wee, Tin Tin and Holland, Jonathan L. and Isakanian, Patrick and Rim, Kern and Yang, Sam and Schneider, Jacob S. and Nallapati, Giri and Dundigal, Sreeker and Lakdawala, Hasnain and Amelifard, Behnam and Lee, Chulkyu and McGovern, Betty and Holdaway, Paul S. and Kong, Xiaohua and Leary, Burton M. (2018). Analog/mixed-signal design challenges in 7-nm CMOS and beyond. In *2018 IEEE Custom Integrated Circuits Conference (CICC)* (IEEE, San Diego, CA, 2018), 1–8, <https://doi.org/10.1109/CICC.2018.8357060>.
- [85] Wei, A. *et al.*, Wei, A. and Singh, J. and Bouche, G. and Zaleski, M. and Augur, R. and Senapati, B. and Stephens, J. and Lin, I. and Rashed, M. and Yuan, L. and Kye, J. and Woo, Y. and Zeng, J. and Levinson, H. and Wehbi, A. and Hang, P. and Ton-That, V. and Kanagala, V. and Yu, D.

- and Blackwell, D. and Beece, A. and Gao, S. and Thangaraju, S. and Alapati, R. and Samavedam, S. (2014). Challenges of analog and I/O scaling in 10nm SoC technology and beyond. In *2014 IEEE International Electron Devices Meeting* (IEEE, San Francisco, CA, USA, 2014), 18.3.1–18.3.4, <https://doi.org/10.1109/IEDM.2014.7047076>.
- [86] Loke, A. L. S. *et al.* in *Analog/Mixed-Signal Design in FinFET Technologies* In Harpe, P., Makinwa, K. A. A. & Baschiroto, A. (eds) *Hybrid ADCs, Smart Sensors for the IoT, and Sub-1V & Advanced Node Analog Circuit Design* 259–280 (Springer International Publishing, Cham, 2018). URL http://link.springer.com/10.1007/978-3-319-61285-0_14.
- [87] Singh, J. *et al.*, Singh, J. and Bousquet, A. and Ciavatti, J. and Sundaram, K. and Wong, J. S. and Chew, K. W. and Bandyopadhyay, A. and Li, S. and Bellaouar, A. and Pandey, S. M. and Zhu, B. and Martin, A. and Kyono, C. and Goo, J.-S. and Yang, H. S. and Mehta, A. and Zhang, X. and Hu, O. and Mahajan, S. and Geiss, E. and Yamaguchi, S. and Mittal, S. and Asra, R. and Balasubramaniam, P. and Watts, J. and Harame, D. and Todi, R. M. and Samavedam, S. B. and Sohn, D. K. (2017). 14nm FinFET technology for analog and RF applications. In *2017 Symposium on VLSI Technology* (IEEE, Kyoto, Japan, 2017), T140–T141, <https://doi.org/10.23919/VLSIT.2017.7998154>.
- [88] Gul, W., Shams, M. & Al-Khalili, D. SRAM Cell Design Challenges in Modern Deep Sub-Micron Technologies: An Overview. *Micromachines* **13**, 1332 (2022). <https://doi.org/10.3390/mi13081332>.
- [89] Kim, J. *et al.* Design and Analysis of a 12-b Current-Steering DAC in a 14-nm FinFET Technology for 2G/3G/4G Cellular Applications. *IEEE Transactions on Circuits and Systems I: Regular Papers* **66**, 3723–3732 (2019). <https://doi.org/10.1109/TCSI.2019.2913174>.
- [90] Marinella, M. J. *et al.* Multiscale Co-Design Analysis of Energy, Latency, Area, and Accuracy of a ReRAM Analog Neural Training Accelerator. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* **8**, 86–101 (2018). <https://doi.org/10.1109/JETCAS.2018.2796379>.
- [91] Salehi, S. & DeMara, R. F., Salehi, Soheil and DeMara, Ronald F. (2015). Energy and area analysis of a floating-point unit in 15nm CMOS process technology. In *SoutheastCon 2015* (IEEE, Fort Lauderdale, FL, USA, 2015), 1–5, <https://doi.org/10.1109/SECON.2015.7132972>.
- [92] Benner, P., Ezzatti, P., Quintana-Ortí, E. & Remón, A. Matrix inversion on CPU–GPU platforms with applications in control theory. *Concurrency and Computation: Practice and Experience* **25**, 1170–1182 (2013). <https://doi.org/10.1002/cpe.2933>.
- [93] Ezzatti, P., Quintana-Orti, E. S. & Remon, A., Ezzatti, Pablo and Quintana-Orti, Enrique S. and Remon, Alfredo (2011). High Performance Matrix Inversion on a Multi-core Platform with Several GPUs. In *2011 19th International Euromicro Conference on Parallel, Distributed and Network-Based Processing* (IEEE, Ayia Napa, Cyprus, 2011), 87–93, <https://doi.org/10.1109/PDP.2011.66>.
- [94] Zhou, G., Feng, Y., Bo, R. & Zhang, T. GPU-accelerated sparse matrices parallel inversion algorithm for large-scale power systems. *International Journal of Electrical Power & Energy Systems* **111**, 34–43 (2019). <https://doi.org/10.1016/j.ijepes.2019.03.074>.
- [95] Valade, A., Acco, P., Grabolosa, P. & Fourniols, J.-Y. A Study about Kalman Filters Applied to Embedded Sensors. *Sensors* **17**, 2810 (2017). <https://doi.org/10.3390/s17122810>.
- [96] Ferdinando, H., Khoswanto, H. & Purwanto, D., Ferdinando, Hany and Khoswanto, Handry and Purwanto, Djoko (2012). Embedded Kalman Filter for Inertial Measurement Unit (IMU) on the ATmega8535. In *2012 International Symposium on Innovations in Intelligent Systems and Applications* (IEEE, Trabzon, Turkey, 2012), 1–5, <https://doi.org/10.1109/INISTA.2012.6246978>.

- [97] Zhang, F. *et al.*, Zhang, Fan and Ur Rehman, M. Muneeb and Wang, Hongjie and Levron, Yoash and Plett, Gregory and Zane, Regan and Maksimovic, Dragan (2015). State-of-charge estimation based on microcontroller-implemented sigma-point Kalman filter in a modular cell balancing system for Lithium-Ion battery packs. In *2015 IEEE 16th Workshop on Control and Modeling for Power Electronics (COMPEL)* (IEEE, Vancouver, BC, 2015), 1–7, <https://doi.org/10.1109/COMPEL.2015.7236525>.
- [98] Dilys, J., Stankevič, V. & Luksza, K. Implementation of Extended Kalman Filter with Optimized Execution Time for Sensorless Control of a PMSM Using ARM Cortex-M3 Microcontroller. *Energies* **14**, 3491 (2021). <https://doi.org/10.3390/en14123491>.
- [99] STM32H7R7 32-bit Microcontroller (2024). URL <https://www.st.com/en/microcontrollers-microprocessors/stm32h7r7-7s7.html>.
- [100] ARM Cortex-M7 Processor Datasheet (2025). URL <https://www.arm.com/-/media/Arm%20Developer%20Community/PDF/Processor%20Datasheets/Arm-Cortex-M7-Processor-Datasheet.pdf>.
- [101] Kung, Y.-S., Tseng, K.-H., Chen, C.-S., Sze, H.-Z. & Wang, A.-P., Kung, Ying-Shieh and Tseng, Kuan-Hsuan and Chen, Chia-Sheng and Sze, Hau-Zen and Wang, An-Peng (2006). FPGA-Implementation of Inverse Kinematics and Servo Controller for Robot Manipulator. In *2006 IEEE International Conference on Robotics and Biomimetics* (IEEE, Kunming, China, 2006), 1163–1168, <https://doi.org/10.1109/ROBIO.2006.340093>.
- [102] Gac, K., Karpiel, G. & Petko, M., Gac, Konrad and Karpiel, Grzegorz and Petko, Maciej (2012). FPGA based hardware accelerator for calculations of the parallel robot inverse kinematics. In *Proceedings of 2012 IEEE 17th International Conference on Emerging Technologies & Factory Automation (ETFA 2012)* (IEEE, Krakow, Poland, 2012), 1–4, <https://doi.org/10.1109/ETFA.2012.6489717>.
- [103] AMD Artix 7 FPGA (2024). URL <https://www.amd.com/en/products/adaptive-socs-and-fpgas/fpga/artix-7.html>.
- [104] AMD Power Estimator. URL <https://www.amd.com/en/products/adaptive-socs-and-fpgas/technologies/power-efficiency/power-estimator.html>.
- [105] Xilinx XC7A100T FPGA datasheet (2020). URL <https://www.xilinxsemi.com/datasheet/xilinxsemi/XC7A100T-3FTG256E.pdf>.
- [106] 7 Series DSP48E1 Slice User Guide (2018). URL https://docs.amd.com/v/u/en-US/ug479_7Series_DSP48E1.
- [107] Eldafrawy, M., Boutros, A., Yazdanshenas, S. & Betz, V. FPGA Logic Block Architectures for Efficient Deep Learning Inference. *ACM Transactions on Reconfigurable Technology and Systems* **13**, 1–34 (2020). <https://doi.org/10.1145/3393668>.
- [108] 7 Series FPGAs Packaging and Pinout Product Specification (UG475) (2024). URL https://docs.amd.com/v/u/en-US/ug475_7Series_Pkg_Pinout.
- [109] Khaddam-Aljameh, R. *et al.* HERMES-Core—A 1.59-TOPS/mm² PCM on 14-nm CMOS In-Memory Compute Core Using 300-ps/LSB Linearized CCO-Based ADCs. *IEEE Journal of Solid-State Circuits* **57**, 1027–1038 (2022). <https://doi.org/10.1109/JSSC.2022.3140414>.
- [110] Wang, H. *et al.* A Charge Domain SRAM Compute-in-Memory Macro With C-2C Ladder-Based 8-Bit MAC Unit in 22-nm FinFET Process for Edge Inference. *IEEE Journal of Solid-State Circuits* **58**, 1037–1050 (2023). <https://doi.org/10.1109/JSSC.2022.3232601>.

882 [111] Zhang, X., Jo, Y.-J. & Kim, T. T.-H. A 65-nm 55.8-TOPS/W Compact 2T eDRAM-Based Compute-
883 in-Memory Macro With Linear Calibration. *IEEE Transactions on Very Large Scale Integration (VLSI)*
884 *Systems* 1–5 (2025). <https://doi.org/10.1109/TVLSI.2024.3520588>.

885 **Acknowledgments.**

886 **Declarations**

887 **Funding** This work was supported by the European Research Council (ERC) under the European Union’s
888 Horizon Europe Research and Innovation Programme under Grant ERC-2021-AdG-101054098-ANIMATE.

889 **Competing interests** The authors declare no competing interests.

890 **Ethics approval** Not applicable.

891 **Consent to participate** Not applicable.

892 **Consent for publication** Not applicable.

893 **Availability of data and materials** Experimental data shown in figures in this work are available on
894 Figshare at <https://figshare.com/s/919f8adeabc759323691?file=51036278> (doi: 10.6084/m9.figshare.27984260).
895 Additional characterization data are available from the authors upon reasonable request.

896 **Code availability** All program codes used in this work are available from the authors upon reasonable request.

897 **Authors’ contributions** D.I., P.M., and G.P. conceived the project and constructed the research frame. P.M.,
898 I.A., A.P., and E.M. designed the circuits for Kalman filter and inverse kinematics. P.M., C.Z., I.A., A.P.,
899 and E.M. performed the circuit simulations. P.M., C.Z., I.A., and A.P. performed the experiments. C.V. and
900 F.S. designed the integrated circuit. P.M. realized the experimental setup. P.M., C.Z., and I.A. contributed
901 to performance benchmarking. D.I. directed the project. All authors supported the analysis of the results and
902 commented on the article. P.M. and D.I. co-wrote the manuscript based on the input from all authors.

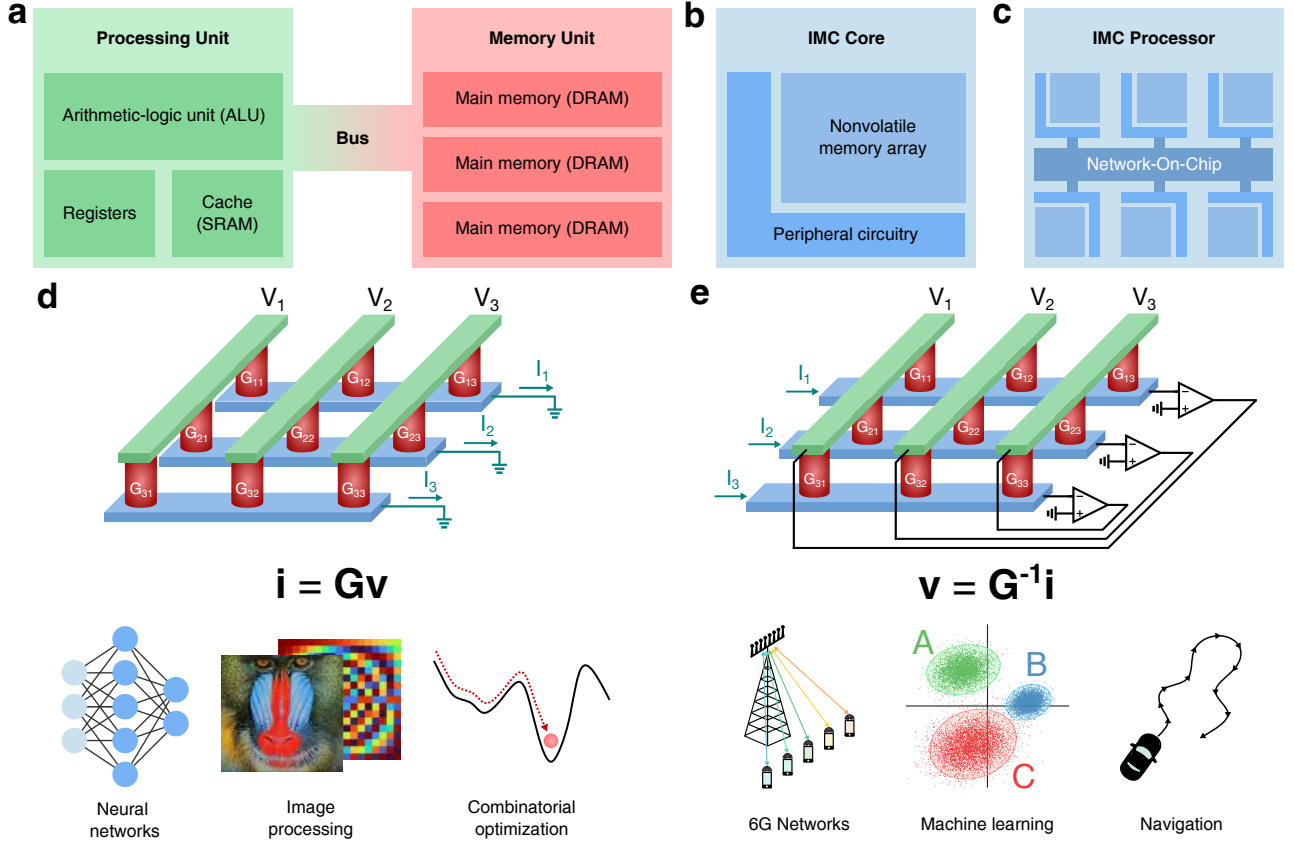


Fig. 1 From von Neumann to in-memory computing. (a) Prototypical von Neumann architecture, where memory and processing units are physically separated and interconnected by a data bus. (b) IMC core, where computation happens directly within the embedded nonvolatile memory array. (c) IMC processor, composed of several IMC cores interconnected by a dedicated Network-On-Chip. (d) Open-loop IMC-based MVM circuit and selected applications. (e) Closed-loop IMC-based IMVM circuit and selected applications.

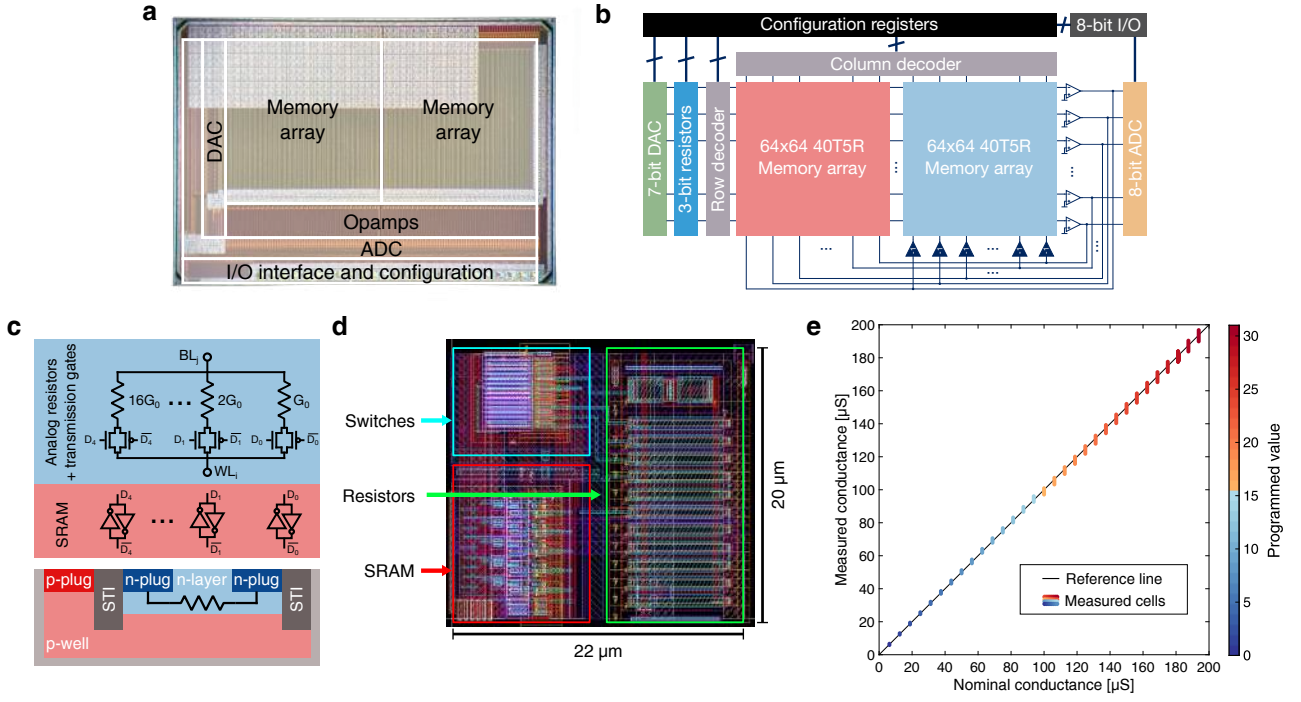


Fig. 2 Closed-loop IMC testchip. (a) Die photograph highlighting the main circuit blocks. (b) Testchip schematic, comprising two 64×64 memory arrays enclosed in a feedback loop by on-chip OAs and inverting buffers, driven by on-chip DACs. A parallel 8-bit I/O bus allows chip reconfiguration and memory programming. (c) 40T5R memory cell schematic, consisting of 5 n-well resistors with scaled values from G_0 to $16G_0$, each driven by a 2T transmission gate enabled by a nearby 6T SRAM. (d) 40T5R memory cell layout in the EDA software, highlighting an almost equal area subdivision between resistors and selecting circuitry for an overall $442 \mu\text{m}^2$ cell size. (e) Measured conductances as a function of nominal conductances for 4096 memory cells.

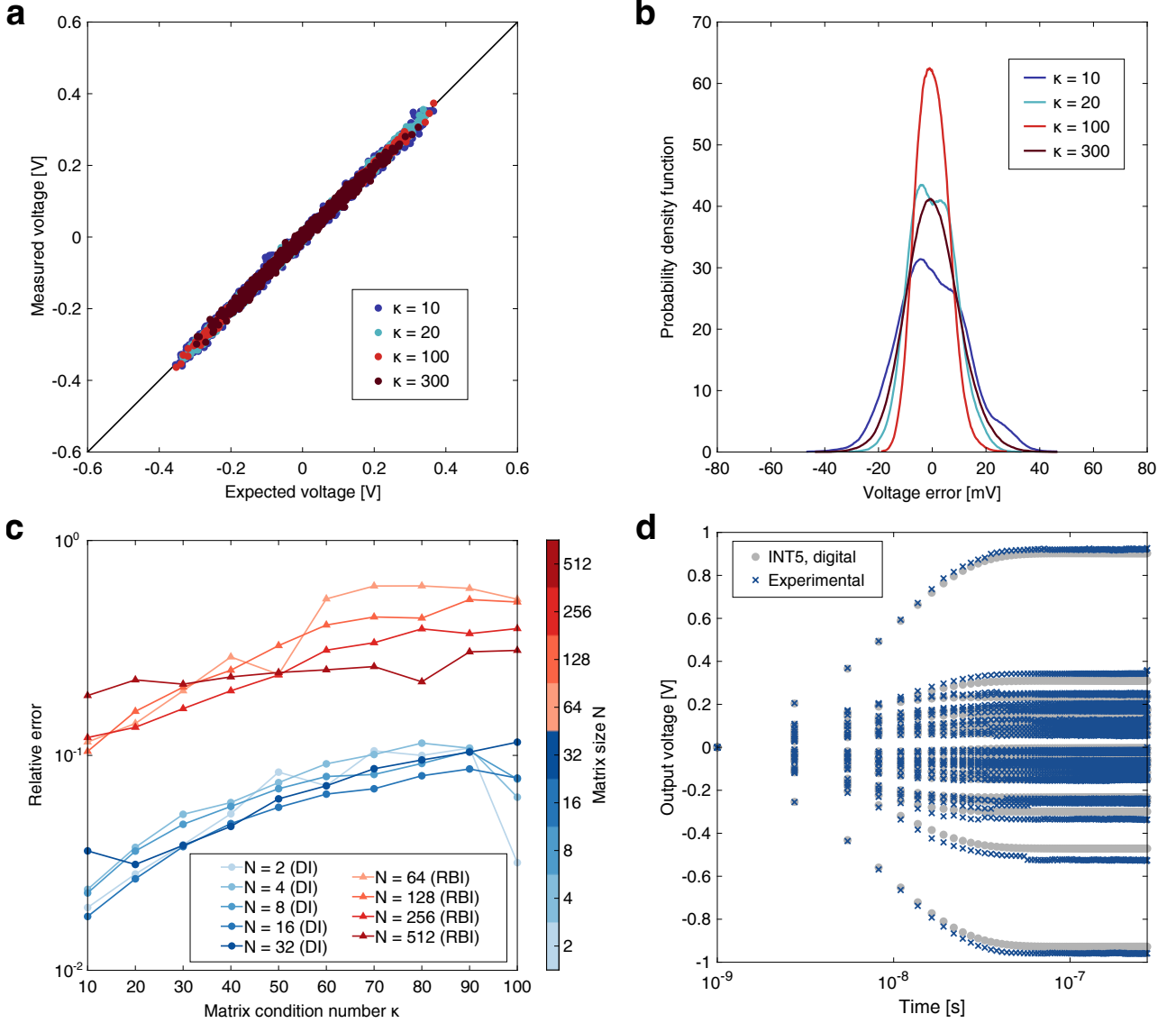


Fig. 3 On-chip linear system solution. (a) Measured solution voltages as a function of expected solution voltages for positive-definite 38×38 linear systems with increasing condition number κ , namely $\kappa = 10$ (blue), $\kappa = 20$ (cyan), $\kappa = 100$ (red), $\kappa = 300$ (dark red). (b) Absolute voltage error histograms for the solution of positive-definite linear systems with increasing condition number κ , namely $\kappa = 10$ (blue), $\kappa = 20$ (cyan), $\kappa = 100$ (red), $\kappa = 300$ (dark red), highlighting a κ -independent standard deviation $\sigma \simeq 10$ mV. (c) Relative error as a function of matrix condition number κ , for linear systems of increasing size N from $N = 2$ to $N = 512$. Systems whose size was compatible with the CL-IMC memory array, namely from $N = 2$ to $N = 32$ (blue points), were solved by DI. Conversely, linear systems where the number of equations exceeded the available array size, namely from $N = 64$ to $N = 512$ (red triangles), were solved by RBI, where each sub-operation was performed on-chip. (d) Output voltage transients for a 32×32 CL-IMC circuit solving a covariance matrix-based linear system, obtained by numerical integration of the associated DE in a 5-bit digital solver (grey) and by on-chip RBI of the equivalent 3200×3200 linear system (blue).

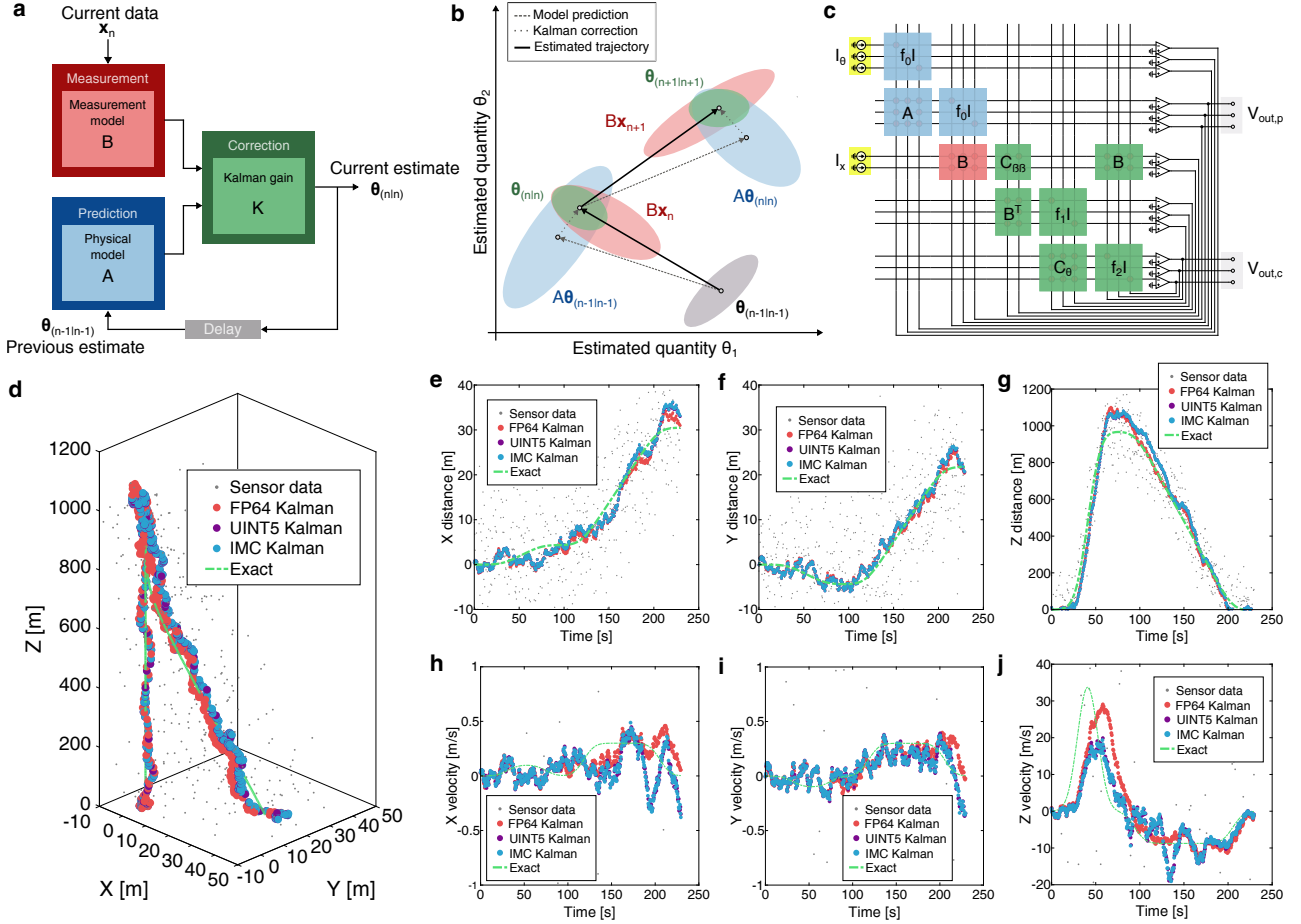


Fig. 4 In-memory Kalman filter. (a) Concept scheme of the KF, where predictions from a physical model A are combined with measurements from a setup B with weight matrix K to provide a refined estimate $\theta_{(n|n)}$ of some variables at the current timestep. (b) In the space of the θ_i variables, KF estimates the centroid of the joint distribution of predictions and measurements at each timestep. (c) Testchip memory array configuration for KF circuit. (d) Rocket trajectory in the 3D space, as estimated by IMC-based KF (blue) and digital KF implementations at FP64 (red) and INT5 (purple) precision, together with sensor data (grey) and exact trajectory (green) for reference. (e-j) Estimated values of (e) x -, (f) y -, (g) z -coordinates and (h) x -, (i) y -, (j) z -velocities over time, for IMC-based KF (blue), digital implementations at FP64 (red) and UINT5 (purple) precision, together with unfiltered sensor data (grey) and exact parameters (green) for reference.

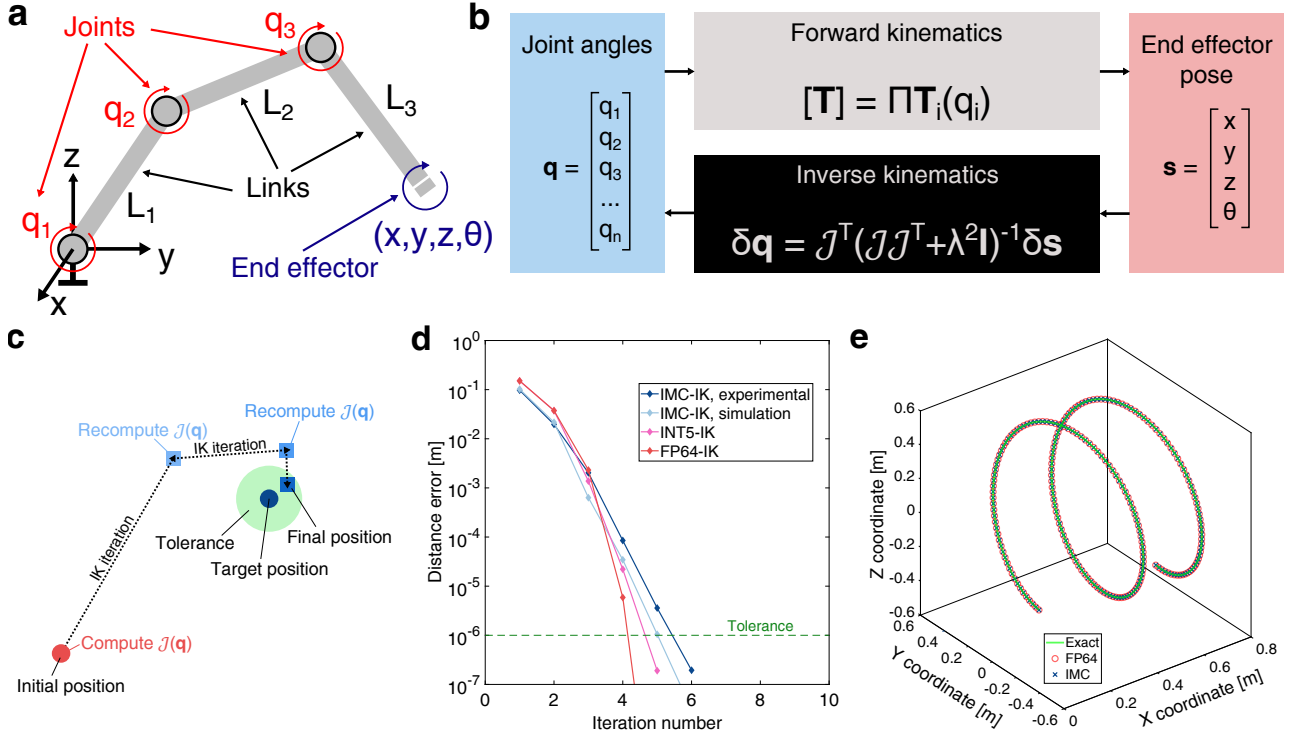
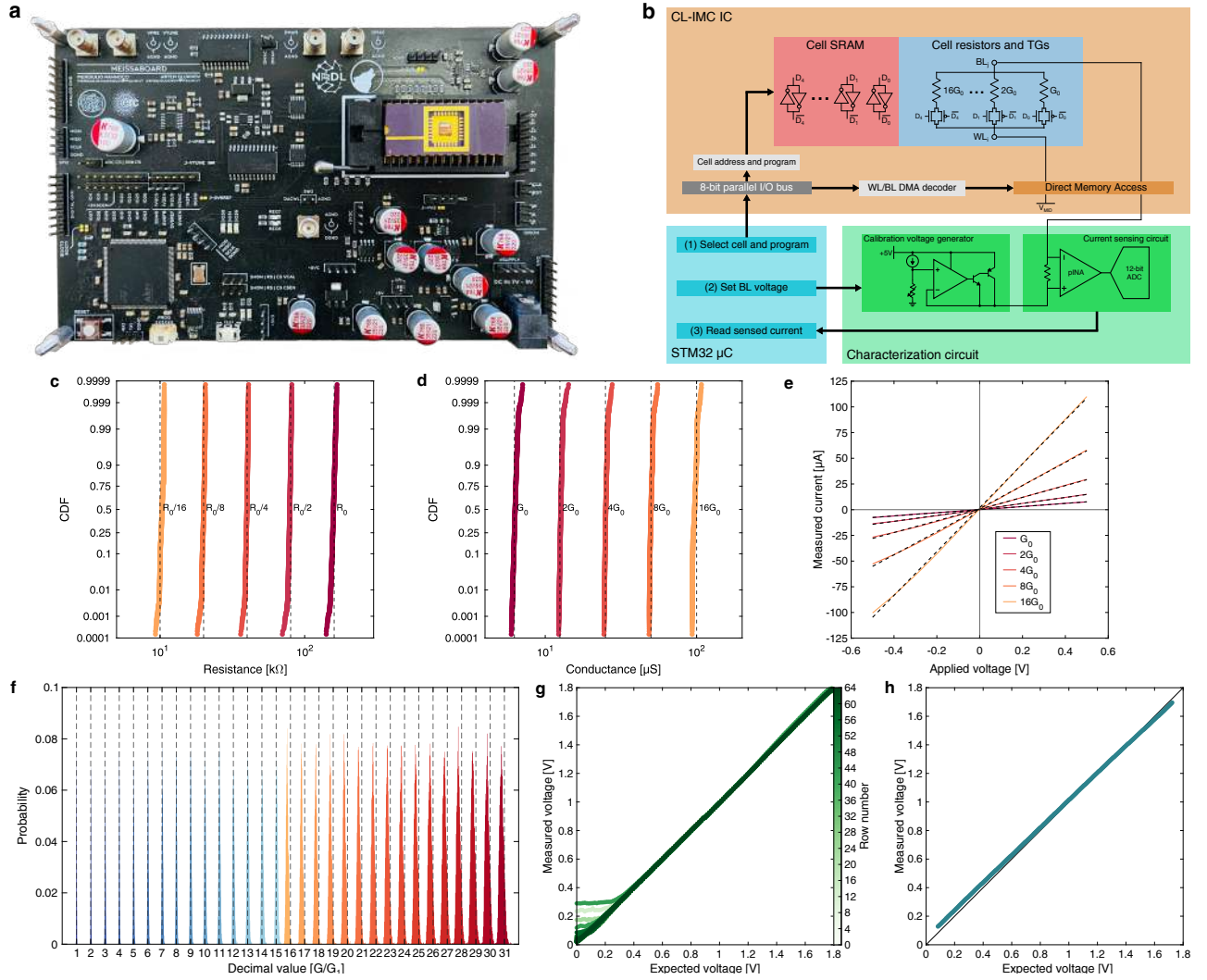
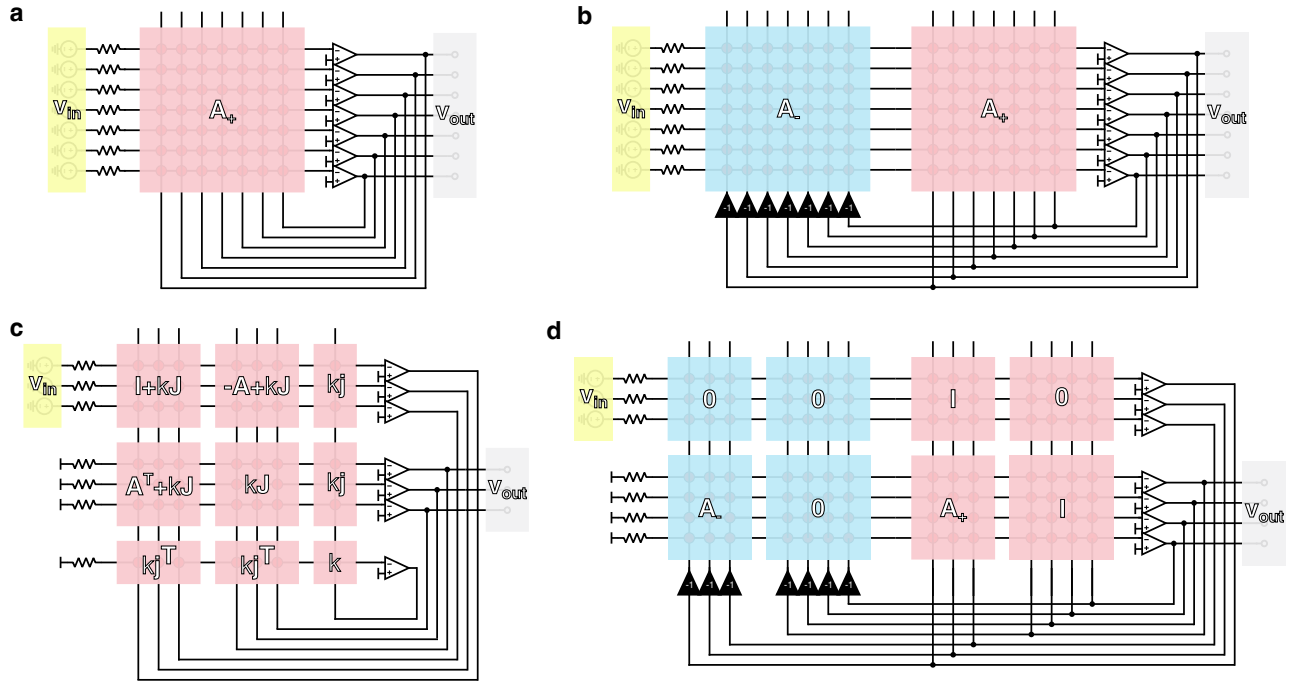


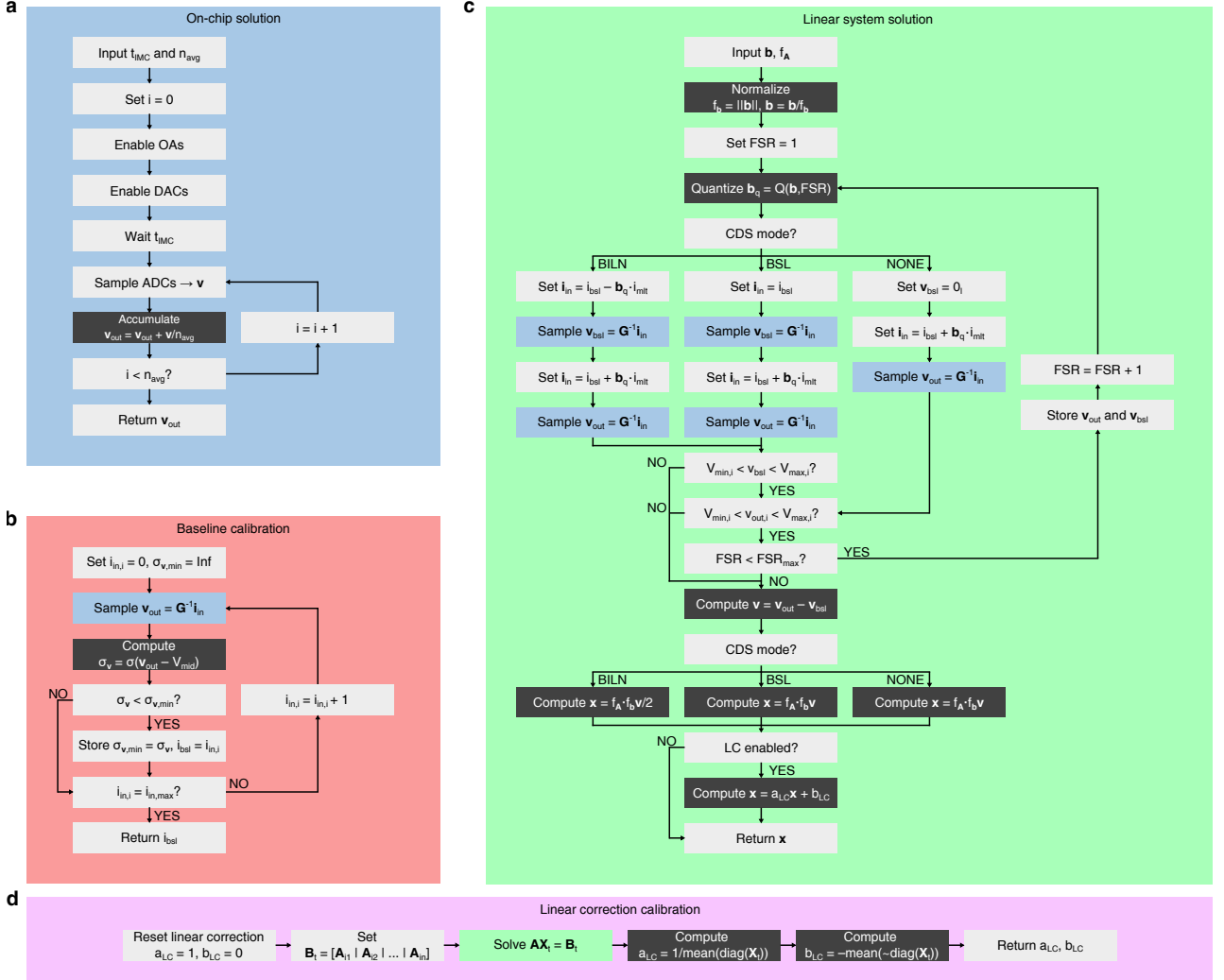
Fig. 5 In-memory inverse kinematics. (a) Prototypical robot arm, where several links are interconnected by rotating joints described by angles q_i . By tuning the joint angles, the arm terminal or end effector position and orientation can be changed inside a 3D workspace. (b) Concept scheme of kinematic operations for a robotic arm. Starting from the joint angles, the pose of the end effector, *i.e.* position and orientation, in the Cartesian space is computed by Denavit-Hartenberg transforms in the forward kinematics step. From the Cartesian space, inverse kinematics techniques are used to compute angle updates to reach a desired target position. (c) Conceptual scheme of IK-based arm movement. Starting from the initial position (red), the Jacobian matrix is computed and used to calculate a first-order angle update in the first IK iteration. The Jacobian matrix is recomputed with the new angle values and used to assess the new position (light blue) of the robotic arm, allowing computation of a second-order angle update in the second IK iteration. The process is repeated until the arm position lies inside a tolerance region (green) with respect to the target position (dark blue). (d) Distance error as a function of the iteration number for IMC experiments (blue) and circuit simulations (cyan), as compared to FP64 (red) and INT5 (magenta) IK. Thanks to the iterative scheme, all implementations progressively reduce the distance error with respect to the target position, with the overall number of required iterations typically increasing with reduced computation precision. (e) Helix trajectory in the 3D space for FP64 (red) and IMC-based (blue) IK. Thanks to the iterative procedure, both implementations exactly match the ideal trajectory (green).



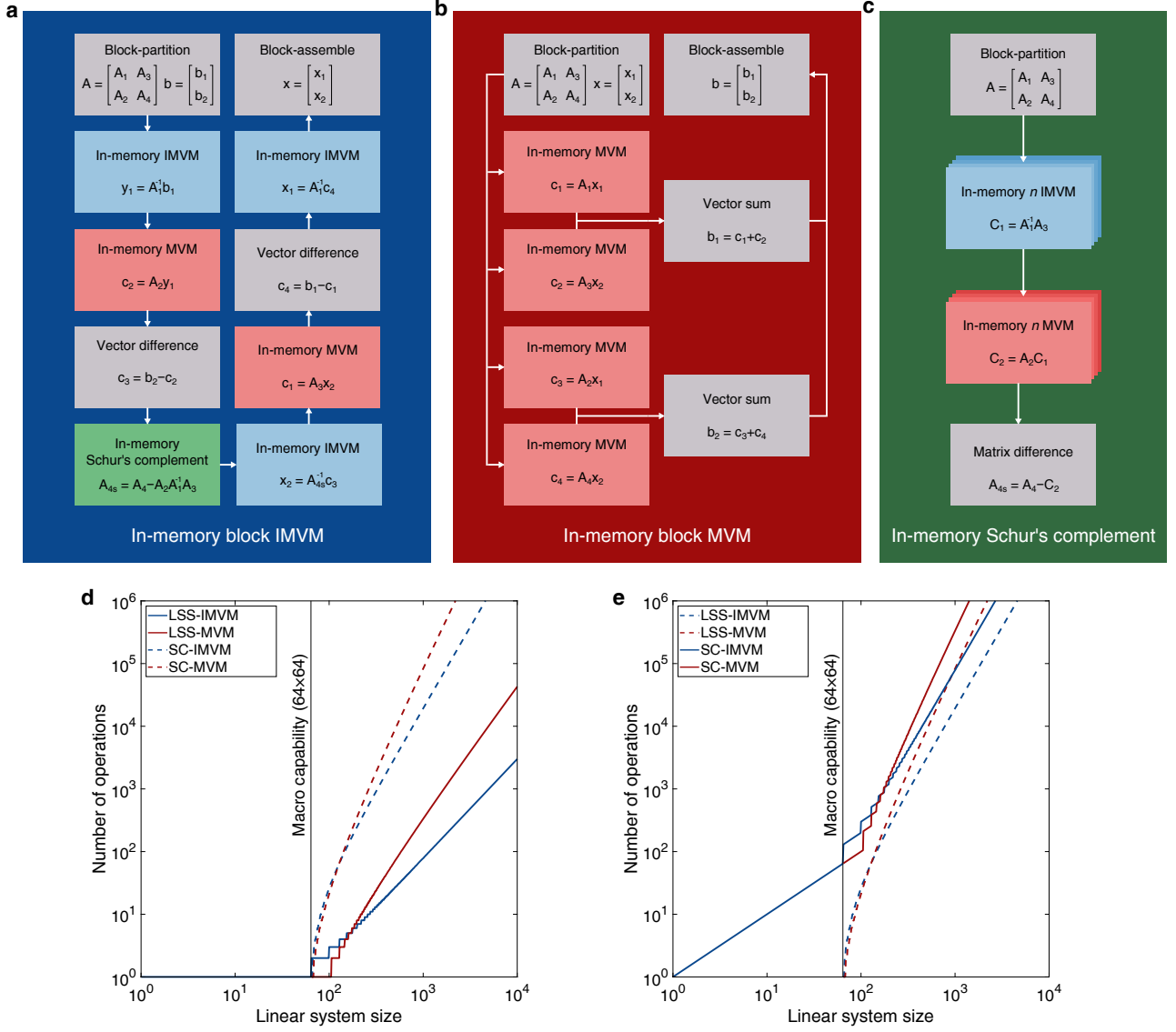
Extended Data Fig. 1 IC characterization and setup. (a) Custom PCB for IC characterization, comprising an onboard MCU for IC interfacing and circuitry orchestration. (b) Memory cell characterization schematic. The MCU first programs a memory cell in the testchip and connects the corresponding row and column to the DMA interface. A calibration voltage generated by a 10-bit onboard circuit is applied to the column line while the row line is kept at a mid-supply voltage, mimicking closed-loop operation. Current flowing into the cell is read by a programmable-gain instrumentation amplifier (INA), converted to the digital domain by a 12-bit onboard ADC, and transferred to the MCU. (c) Resistance and (d) conductance cumulative distribution functions for 8192 memory cells measured on two different testchips. (e) Measured memory cell current-voltage (I-V) curve for digital values corresponding to activation of one out of five integrated resistors, highlighting the linearity of the memory cell. (f) Array characterization results for 4096 programmed cells with values from 1 to 31. Depending on the circuit configuration used, conductance values from G_0 to $31G_0$ can encode values from 1 to 31 (UINT5/INT6 configuration) or from -15 to 15 (INT5 configuration). In the latter case, conductance values from G_0 to $15G_0$ (blue) encode values from -15 to -1, and conductance values from $17G_0$ to $31G_0$ encode values from +1 to +15 (red), with $16G_0$ correspondingly encoding a 0 value. Higher conductive states show a wider spread owing to the cumulative variability of the selected resistors. Mean shift to lower conductances for decimal values from 16 to 31 is induced by the additional 1.6 k Ω series resistance contribution of the TGs to the overall cell resistance, which is instead negligible for decimal values from 1 to 15. (g) Measured output voltage of 64 on-chip voltage DACs used to provide circuit inputs, highlighting saturation effects at low voltages. (h) Measured characteristic of the on-chip ADC for output voltage readout, highlighting a small linearity error.



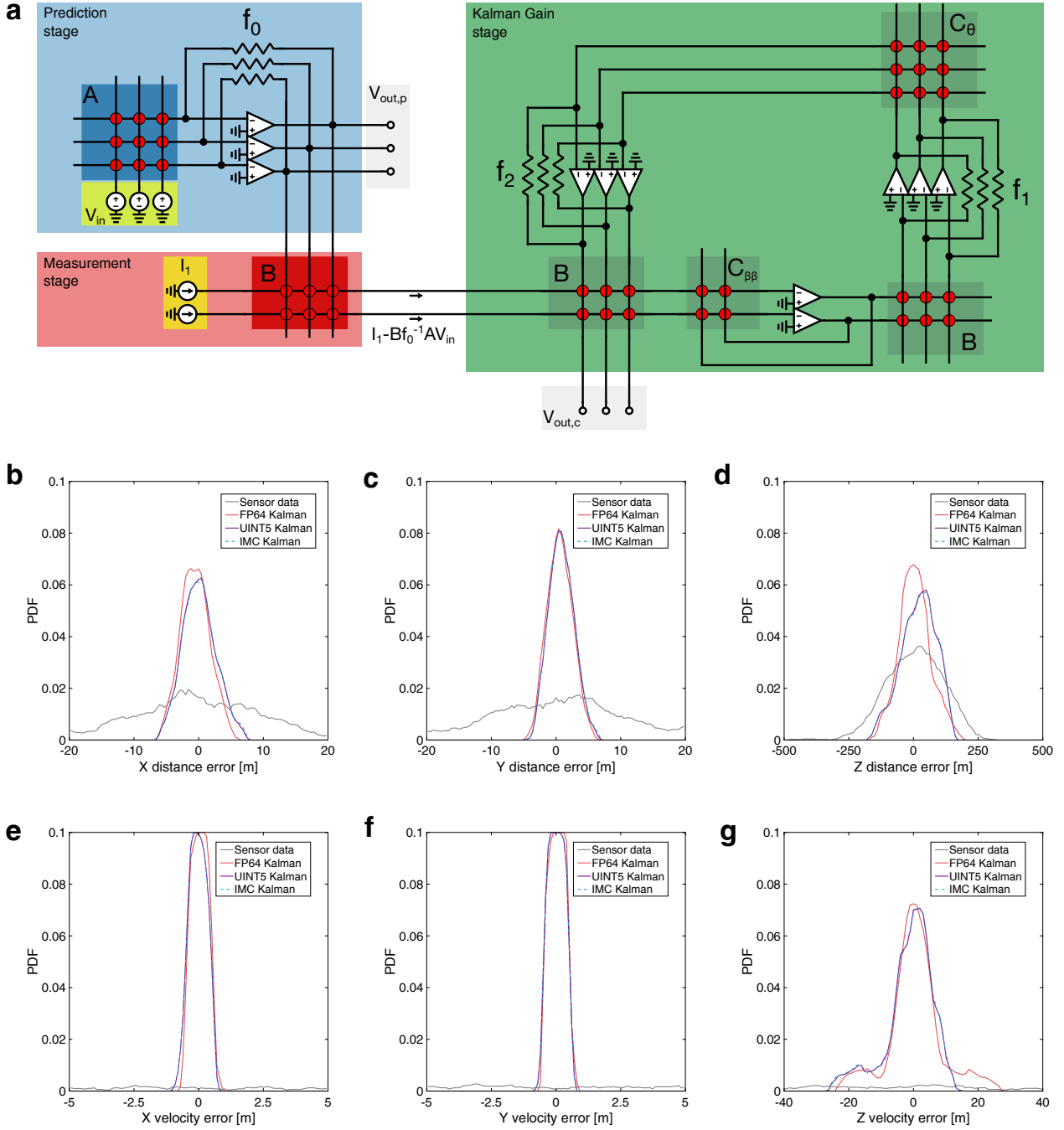
Extended Data Fig. 2 Circuit configurations. (a) Single-matrix circuit (SMC) configuration, allowing solution of positive-valued, positive-definite linear systems. (b) Double-matrix circuit (DMC) configuration for solving real-valued, positive-definite linear systems. (c) Linear regression circuit (LRC) with reference topology (RT) configuration for solving real-valued linear systems of arbitrary definiteness. (d) Real-valued matrix-vector multiplication circuit configuration.



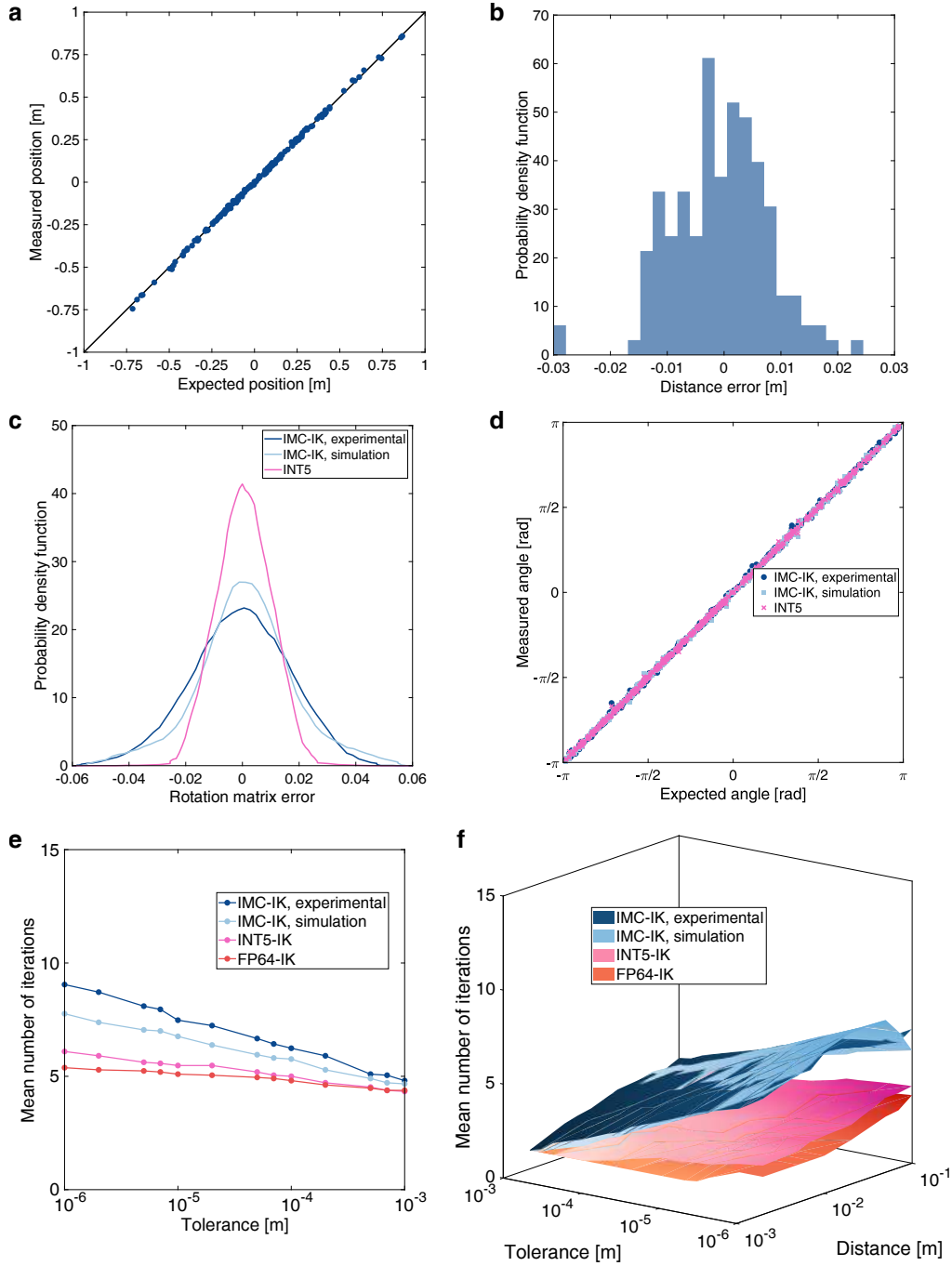
Extended Data Fig. 3 Calibration and error mitigation algorithms. (a) Algorithm for on-chip inverse-matrix-vector multiplication. To mitigate the impact of analog noise, the output voltages are sampled n_{avg} times and averaged. (b) Baseline calibration algorithm. To mitigate the impact of OA offset, the baseline voltage of DACs is calibrated to reduce the spread of output voltages. (c) Algorithm for linear system solution. The input vector $\mathbf{b}$ is normalized and quantized with a progressively increasing full-scale range (FSR) as long as OA output voltages remain inside a confidence window. To eliminate the impact of residual offsets, two IMVM operations are performed with either $+\mathbf{b}$ and $-\mathbf{b}$ (BILN mode) or $+\mathbf{b}$ and $\mathbf{0}$ (BSL mode). (d) Low-complexity linear correction calibration. Columns of the programmed matrix $\mathbf{A}$ are fed as circuit inputs, theoretically providing $\mathbf{A}^{-1}\mathbf{A} = \mathbf{I}$. Measured diagonal and out-of-diagonal values of the output matrix $\hat{\mathbf{I}}$ can then be used to estimate parameters $\mathbf{a}_{LC}, \mathbf{b}_{LC}$ of the linear correction $c(\mathbf{v}) = \mathbf{a}_{LC}\mathbf{v} + \mathbf{b}_{LC}$.



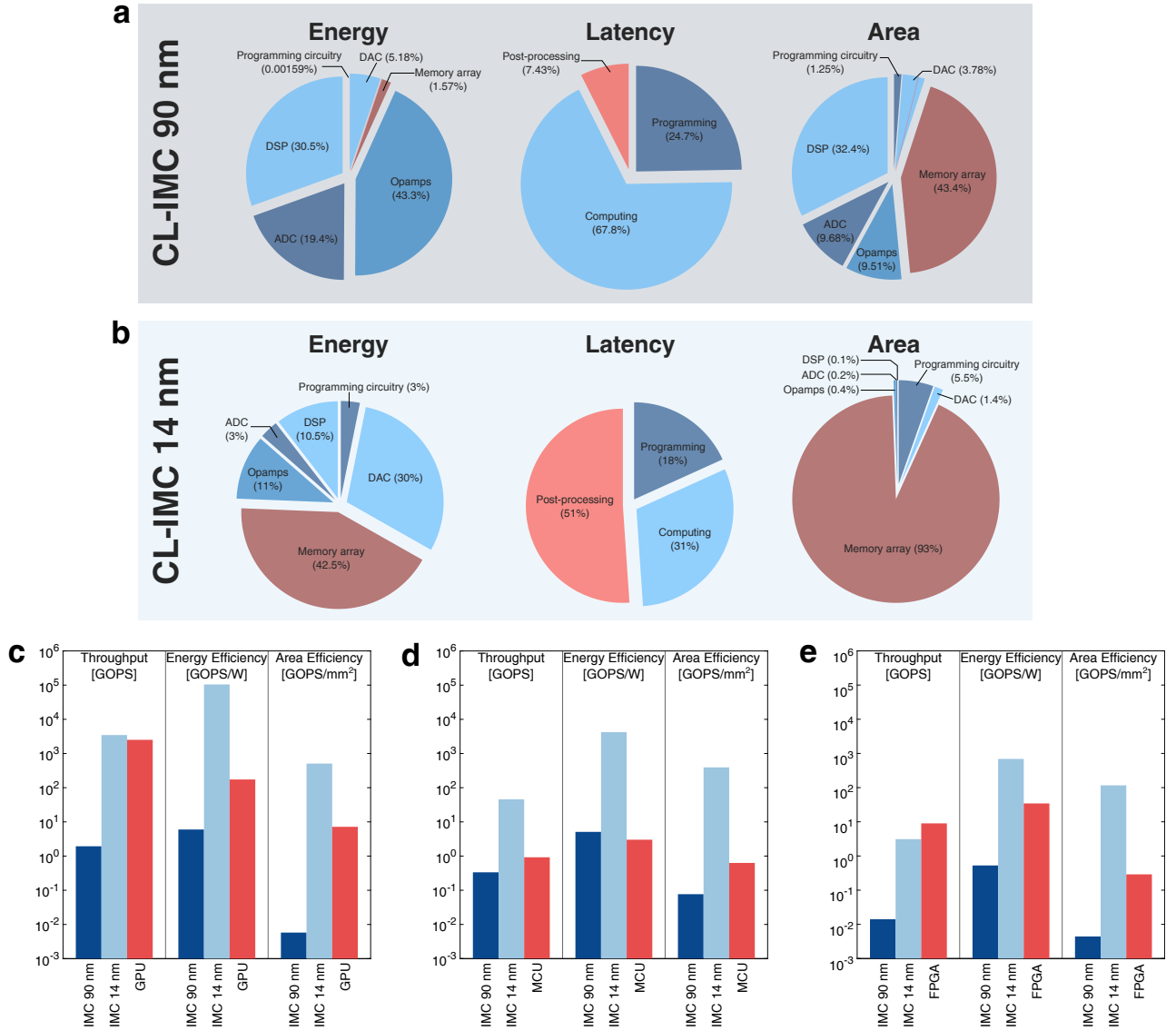
Extended Data Fig. 4 Recursive block-inversion algorithm. (a-c) Workflows for in-memory (a) block IMVM, (b) block MVM, and (c) Schur complement computation. Inner MVMs and IMVMs are computed directly in hardware when the matrix size is compatible with the memory array. Otherwise, the block-IMVM or block-MVM workflow is recursively applied. (d,e) Number of IMVMs and MVMs for Schur complement computation (SC-IMVM, SC-MVM) and linear system solution (LSS-IMVM, LSS-MVM) as a function of the linear system size, for an analog macro with 64×64 IMVM/MVM capability solving (d) $L = 1$ and (e) $L = N$ linear systems, where N is the linear system size. When solving multiple linear systems described by the same matrix $\mathbf{A}$ and different known terms $\mathbf{b}_1, \mathbf{b}_2, \dots$, SC computation is performed once and reused, allowing amortization of the additional overhead.



Extended Data Fig. 5 Accuracy of in-memory Kalman filter. (a) Circuit schematic, comprising in-memory circuits for accelerating prediction, measurement, and Kalman gain blocks of the discrete Kalman filter. (b-g) Error histograms for (b) x , (c) y , and (d) z distances, and (e) x , (f) y , and (g) z velocities, for unfiltered sensor data (grey), FP64-KF (red), INT5-KF (purple), and IMC-KF experiments (cyan), together with exact parameters from the synthetically generated data (green), highlighting consistent accuracy improvement for all coordinates.



Extended Data Fig. 6 Accuracy of in-memory kinematic operations. (a) Measured position as obtained by IMC-based forward kinematics as a function of expected position obtained by FP64 forward kinematics for 50 random configurations of joint angles in a 4-link robotic arm. (b) Distance error histogram for IMC measurements reported in (a). (c) Absolute error histograms for rotation matrix elements to compute end effector orientation for IMC experiments (blue), IMC circuit simulations (cyan), and a fully digital INT5 implementation (pink). (d) Measured end effector orientation as a function of target orientation for IMC experiments (blue), IMC circuit simulations (cyan), and a fully digital INT5 implementation (pink). (e) Mean number of iterations as a function of target distance error tolerance for IMC-IK experiments (blue), IMC-IK circuit simulations (cyan), INT5-IK (pink), and FP64-IK (red) for the 3D helix trajectory. (f) Mean number of iterations as a function of distance error tolerance and mean distance between trajectory points for 50 random evenly-spaced points for IMC-IK experiments (blue), FP64-IK (brown), and INT5-IK (pink).



Extended Data Fig. 7 Testchip performance benchmarks. (a,b) Energy, latency, and area breakdowns for (a) the 90 nm testchip presented in this work, and (b) a 14 nm testchip projection with optimized circuit peripherals. (c) Recursive block inversion, (d) 3D trajectory and velocity tracking by Kalman filter, and (e) Inverse kinematics benchmark on throughput, energy, and area efficiency, for the 90 nm IC in this work (dark blue), a simulated 14 nm scaling of the IC in this work with optimized peripherals and architecture (light blue), and (c) an 8 nm commercial GPU [71], (d) a high-performance 40 nm commercial microcontroller (MCU) [77, 99], and (e) a 28 nm commercial FPGA [83, 103]. While the current testchip is unoptimized for performance, by appropriately tailoring the circuit peripherals and taking advantage of technological scaling, CL-IMC provides significant throughput, energy, and area efficiency improvements compared to GPUs, MCUs, and FPGAs.

Component	Technology	Number	Energy	Latency	Area
Programming circuit	90 nm	2×64×64	1.5 nJ	20.8 μ s	0.1 mm ²
Memory array	90 nm	2×64×64	4.15 μ J	6.06 μ s	3.67 mm ²
Opamps	90 nm	2×64			0.8 mm ²
DACs	90 nm	64	478 nJ	20 ns	0.32 mm ²
ADCs	90 nm	64	1.8 μ J	51 μ s	0.82 mm ²
DSP	90 nm	1	2.8 μ J	6.25 μ s	2.74 mm ²
Total	90 nm		9.2 μJ	84 μs	8.5 mm²
Programming circuit	14 nm	2×256×256	2.3 nJ	512 ns	0.37 mm ²
Memory array	14 nm	2×256×256			6.3 mm ²
Opamps	14 nm	2×256	39 nJ	604 ns	0.03 mm ²
DACs	14 nm	256	22 nJ	1 ns	0.09 mm ²
ADCs	14 nm	256	2.24 nJ	256 ns	0.01 mm ²
DSP	14 nm	1	7.7 nJ	1.44 μ s	0.001 mm ²
Total	14 nm		73.2 nJ	2.8 μs	6.8 mm²

Extended Data Tab. 1 Summary of proposed and projected CL-IMC testchip on-chip peripherals latency, energy and area consumption.

	JSSC '22 [109]	JSSC '23 [110]	TVLSI '25 [111]	This work (current)	This work (projected)
Technology	14 nm	22 nm	65 nm	90 nm	14 nm
Cell type	8T4R PCM	9T3C SRAM	2T eDRAM	40T5R CMOS	40T5R CMOS
Chip area	0.63 mm ²	0.25 mm ²	0.017 mm ²	8.5 mm²	6.8 mm²
Array size	256×256	64×16	128×128	64×64	256×256
In/w/out precision	8b/Analog/8b	8b/8b/8b	4b/4b/6b	7b/5b/8b	7b/5b/8b
Operations	MVM	MVM	MVM	MVM + IMVM	MVM + IMVM
Throughput	1 TOPS	0.6 TOPS	1.63 TOPS	3.1 GOPS	5.98 TOPS
Energy efficiency	10.5 TOPS/W	32.2 TOPS/W	55.8 TOPS/W	28.5 GOPS/W	228.6 TOPS/W
Area efficiency	1.6 TOPS/mm ²	2.4 TOPS/mm ²	1.5 TOPS/mm ²	0.37 GOPS/mm²	0.9 TOPS/mm²

Extended Data Tab. 2 Comparison of proposed and projected CL-IMC testchip with other analog IMC macros demonstrating linear algebra acceleration.