Historical perspective and opportunity for computing in memory using floating-gate and resistive non-volatile computing including neuromorphic computing

Hasler, Jennifer; Basu, Arindam

Published in:
Neuromorphic Computing and Engineering

Published: 01/03/2025

Document Version:
Final Published version, also known as Publisher's PDF, Publisher's Final version or Version of Record

Published version (DOI):
10.1088/2634-4386/ad9b4a

Publication details:
Hasler, J., & Basu, A. (2025). Historical perspective and opportunity for computing in memory using floating-gate and resistive non-volatile computing including neuromorphic computing. *Neuromorphic Computing and Engineering*, *5*(1), Article 012001. https://doi.org/10.1088/2634-4386/ad9b4a

Neuromorphic
Computing and
Engineering

TOPICAL REVIEW • OPEN ACCESS

Historical perspective and opportunity for computing in memory using floating-gate and resistive non-volatile computing including neuromorphic computing

To cite this article: Jennifer Hasler and Arindam Basu 2025 *Neuromorph. Comput. Eng.* **5** 012001

IOP Publishing Neuromorph. Comput. Eng. 5 (2025) 012001 https://doi.org/10.1088/2634-4386/ad9b4a

NEUROMORPHIC
Computing and Engineering

OPEN ACCESS

RECEIVED
27 May 2024

REVISED
27 October 2024

ACCEPTED FOR PUBLICATION
6 December 2024

PUBLISHED
8 January 2025

TOPICAL REVIEW

Historical perspective and opportunity for computing in memory using floating-gate and resistive non-volatile computing including neuromorphic computing

Jennifer Hasler[1,*] and Arindam Basu[2]

[1] Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA 30332-250, United States of America
[2] Department of Electrical Engineering, City University of Hong Kong, Kowloon Tong, Hong Kong Special Administrative Region of China, People's Republic of China
* Author to whom any correspondence should be addressed.

E-mail: jennifer.hasler@ece.gatech.edu

Keywords: floating-gate devices, circuits and systems, memristors, RRAM

Abstract

The effort addresses the research activity around the usage of non-volatile memories (NVM) for storage of 'weights' in neural networks and the resulting computation through these memory crossbars. In particular, we focus on the CMOS implementations of, and comparisons between, memristor/resistive random access memory (RRAM) devices, and floating-gate (FG) devices. A historical perspective for illustrating FG and memristor/RRAM devices enables comparison of nonvolatile storage (addressing issues related to resolution, lifetime, endurance etc), feedforward computation (different variants of vector matrix multiplication, tradeoffs between power dissipation and signal to noise ratio etc), programming (addressing issues of selectivity, peripheral circuits, charge trapping etc), and learning algorithms (continuous time LMS or batch update), in these systems. We believe this historical perspective is necessary and timely given the increasing interest in using computation in memory with NVM for a wide variety of memory bound applications.

1. Motivating the device comparison

The tremendous success of deep learning to train a variety of neural networks (NNs) [1] utilized for applications ranging from face recognition [2] to speech recognition [3] to game playing [4] has rekindled a huge interest in NNs. While initial research focused more on new algorithm development using graphics processing units (GPUs) and field programmable gate arrays (FPGAs) as the underlying hardware, the need for new dedicated hardware to implement key computational kernels used in these NNs became evident to both industry [5] and academia [6, 7] for both server scale and edge node deployments. In this context, computing using non-volatile memories (NVMs) has generated a significant amount of research activity as well as the potential of commercial activity. Digital NVM are ubiquitous, being found in portable computing platforms from laptops, to cell-phones, to USB memory sticks, to embedded devices etc. NVM for mixed-signal and analog computation rely on NVM with the potential of commercially competitive computing module and application solutions.

This effort provides a framework and comparison of analog and mixed-signal techniques organized in computational crossbars of potential NVM elements. Computing in memory [8, 9] (CiM), with elements similarly arranged to a digital memory, enables vector-matrix multiplication (VMM) operations (figure 1),

$$I_l = g_0 \sum_{k=1}^{m} w_{l,k} v_{in,k}, \ \mathbf{i} = g_0 \mathbf{W} \mathbf{v}_{\text{in}} \tag{1}$$

where the input voltages ($\mathbf{v}_{\text{in}} : v_{\text{in},1}, \ldots v_{\text{in},m}$) are applied at one side of the memory, the output currents ($\mathbf{i} : i_1, \ldots i_n$), a sum of the currents from each element, are measured at an orthogonal side of the memory, the

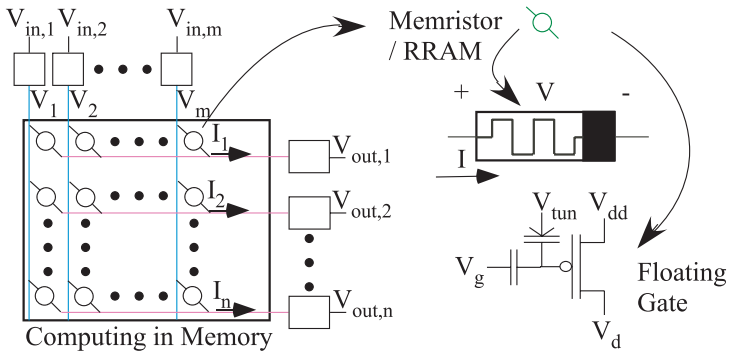


Figure 1. Computing in memory using non-volatile memories (NVM), which include memristor/resistive RAM (RRAM) and floating-gate (FG) devices. These NVM could be implemented entirely in, or at least integrated on, a standard CMOS process.

array elements store a $m \times n$ matrix of weights ($\mathbf{W}$), and g_0 is the baseline or normalizing conductance value arising from the particular device. VMM is the fundamental operation for signal processing and computing for NNs. CiM eliminates the need for addressing and moving the weight values to the computation [9]. The entire computation is performed with the complexity required to locally read a couple of rows from a memory element, drastically reducing the architectural costs of this module's operation [10]. CiM has been revisited several times recently with the new interest in crossbar memories [11], including overcoming the performance limits of memory intensive NN computations imposed by von-Neumann bottleneck [10, 12], CiM techniques rely on using analog or mixed-signal computing within a memory array conventionally used for digital storage and often are utilized in 'neuromorphic' approaches [13–15].

The focus then shifts to addressing the device or devices could be used for these memory computing elements. This paper will focus on the CMOS compatible implementations of *Memristor* devices (such as Resistive RAM (RRAM)) and of floating-gate (FG) or flash devices, as well as the technical comparison between these elements (figure 1). We will start by discussing CiM through crossbar architectures in a historical perspective to illustrate key operations for comparison and to enable the discussion of the individual properties: nonvolatile storage (section 3), feedforward computation (section 4), programming (section 5) and learning algorithms (section 6). We hope the scientific community engaged in designing NVM structures of various CiM implementations, whether memristive, FG, or other techniques, would gain a better historical perspective and also potentially get new ideas from the early efforts of FG-based and related CiM systems. While resistive memory devices have garnered tremendous interest recently [12], FG device arrays have historically preceded RRAM in implementing similar CiM functions, and are generating new interest because of its research and commercial implementation opportunities (e.g. [16]).

This discussion will primarily focus on the crossbar-architecture comparisons with relatively homogeneous operations in the crossbar (e.g. SIMD: Single Instruction Multiple Data architecture), while acknowledging other opportunities, such as handling a large number of heterogeneous operations (e.g. [17]), result from the potential programmable and configurable opportunities of these devices. The development of FG NVM elements have enabled demonstration of large mixed-signal systems, such as large-scale field programmable analog arrays [17], encapsulating high-matching analog circuits [18], including [19], amplifiers [20], sensor interfaces [21], filters [22, 23], and data-converters [24–26]. These structures have shown energy-efficient signal processing applications in acoustics [27, 28], wavelet filterbanks [22], imaging [29, 30]. and RF [31]. Although FG NVM elements enabled connections to neurobiological computing (e.g. [15, 32]), this effort will primarily not address biological similarities since it has been addressed carefully in multiple FG based areas. Further, even though neurobiologically inspired solutions enable a wide range of neuromorphically plausible computations that are outside of the crossbar architecture (e.g. [33]), those discussions are beyond the scope of this discussion.

2. CiM through crossbar architectures: a historical perspective

A historical perspective of FG and memristor/RRAM devices sets the stage for crossbar architectures (figure 1). NVM digital memories, starting with the initial discovery of a MOS FG device [34], has its own unique history leading to EEPROM and flash-based devices (e.g. [35, 36]). The classical FG digital memory device uses two gate layers to create a capacitor on-top of a typical MOSFET device (figure 2(a)). These structures typically modify the FG charge through electron tunneling [37] handling the higher voltage control through modified transistors (e.g. high voltage nFET in a CMOS process, figure 2(b) as well as

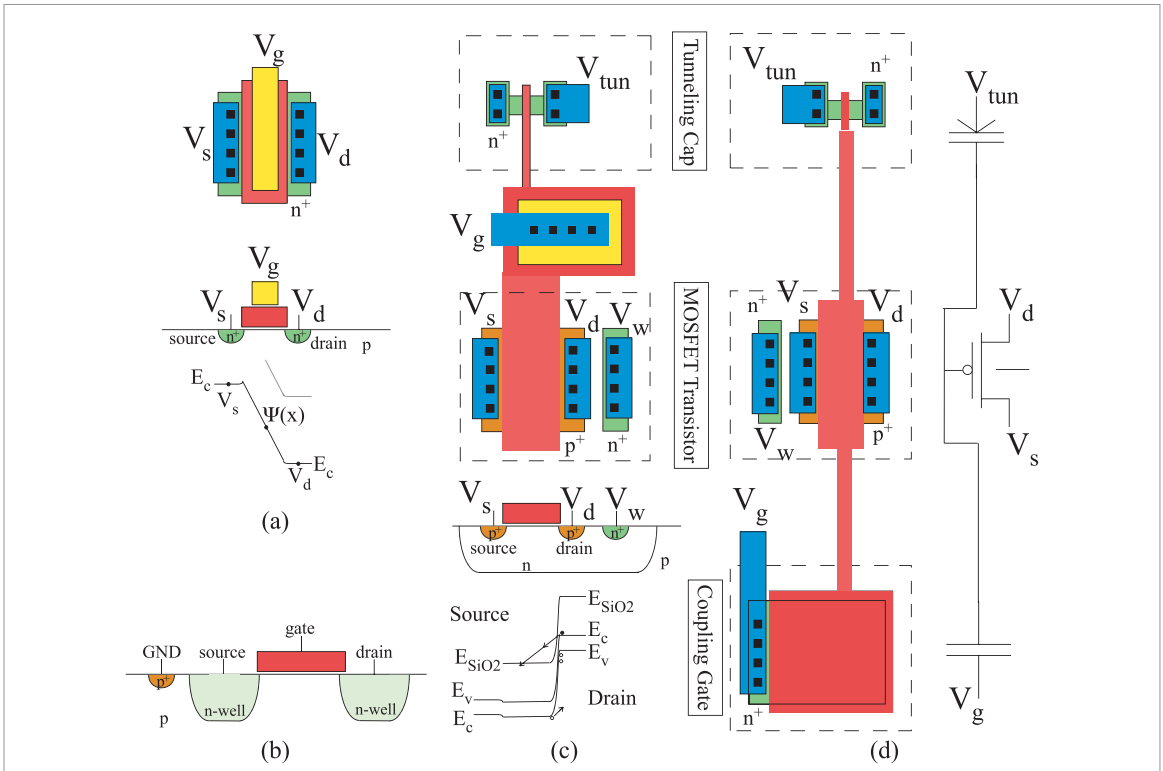


Figure 2. Cross-section and layout for FG devices. (a) Typical single EEPROM or flash cell layout utilizing a double-poly stack over the gate insulator. V_s, V_d, V_g refer to the source, drain, and gate voltages, respectively. (b) Cross-section of a standard CMOS high-voltage capable transistor for controlling tunneling voltages. V_w and V_{tun} refer to the well and tunneling voltages, respectively. (c) Double-poly cross-section and representative layout typical for FG circuits when double-poly is available in standard CMOS. (d) Single-poly cross-section and representative layout typical for FG circuits.

through on-chip charge pumps to generate the higher voltages. Digital memories get configured in either NAND approaches [38] used for mass-storage such as hard-drive replacements, or NOR approaches [39] used for random-access ROM blocks. NOR approaches fit within a crossbar architectures (figure 1) and are mostly used for CiM approaches (though see [40] for NAND based CiM). Current EEPROM devices already store 4 bits (16 levels) in a single transistor of 100 nm × 100 nm area in 32 nm CMOS, or smaller CMOS IC processes, for the last decade [41, 42]. These NAND based structures continued to scale to smaller technology nodes (e.g. 15 nm, 19 nm [43–45]), and have been long in production in these nodes. Flash memory structures continue to scale to higher and higher densities, both in lateral dimensions with continued CMOS scaling, as well as in vertical dimensions (stacks of memory transistors). The primary focus for digital EEPROM and flash elements is density often on optimized IC process lines producing arrays of the NVM ICs.

The history of NVM computing technologies outside of digital memories is directly linked to the development of adaptive filters and NN, as evidenced by the effective naming of 'memistor' elements by Widrow in the 1960s as a three terminal element in his adaptive filter implementations [46]. Implementations of NN effectively build multiple adaptive filters in a complete classifier using multiple VMM operations. During the research into NN IC implementations in the 1980s and early 1990s, the lack of an analog NVM was *the struggle* that defined the field. Mesh crossbar architectures (figure 1), modifying theory from systolic and wavefront arrays [47, 48], were already understood as the optimal method for implementing synaptic arrays (e.g. [49]). In a typical crossbar array performing VMM computation, it is the weight storage and computation that consumes most of the area (O(n m)) as well as dominates other complexity measures. A k-bit DAC at every node requires O(2^k) area and energy, and a multiplexed set of weight values temporarily stored on capacitors requires significant complexity in storing and generating the resulting weights (e.g. see [17]). These solutions were costly even for toy problems. Heroic implementations of refreshable analog dynamic memories were developed with some success [50–52].

Only by solving the analog NVM problem would the field of NN implementations, as well as analog computation, become competitive with digital applications. The following subsections follow the development of analog/mixed-signal FG approaches (section 2.1) and the development of memristor approaches (section 2.2).

2.1. Mesh-based synaptic architectures and single-transistor learning synapses (STLSs)

The definitive answer to an analog NVM came from Caltech in 1994–1995 [53, 54] with the STLS. The STLS was a FG element (figure 1) implemented in a commonly available CMOS process (through MOSIS) that satisfied the five requirements for a synapse element required by a VMM array, adaptive filter, or NN computation:

(i) Non-volatile storage of weight values → FG device.
(ii) Enables a multiplication or related nonlinear function of inputs and the stored weight value → $e^{-\Delta V_{fg}/U_T}$ where ΔV_{fg} denotes charge change on the FG voltage (V_{fg}) and $U_T = kT/q \approx 25$ mV at room temperature.
(iii) Has small area to minimize the overall mesh area → 1 transistor and 2+ capacitors.
(iv) Requires very low energy to minimize the overall mesh energy → subthreshold operation (e.g. nA, pA)
(v) Program the weights as well as modify the weights corresponding to outer-product rules → combination of electron tunneling and hot-carrier injection, where hot-carrier injection is roughly proportional to channel current.

The STLS device became the prime example of a solution to this approach. These requirements are essential for any structure operating in a two-dimensional array of mesh processors, particularly for a two-dimensional crossbar array, and from its original definition, provided the comparison point for any resulting VMM and NN approaches. CiM and VMM crossbar computation had begun!

The STLS FG device was slightly modified from a classical EEPROM transistor (figure 2(a)) to enable additional functionality as well as to utilize capabilities available in standard CMOS processes. The FG transistor gate is capacitively coupled by one or multiple capacitors to the first level gate material (e.g. polysilicon). The STLS device often modified the classical EEPROM transistor stack (figure 2(a)) by having one or more of the capacitors not above the transistor channel, and yet still connected to the same gate layer (figure 2(c)). As many standard CMOS processes do not have two gate level materials, a FG device using MOS capacitors [55] provides similar performance to its double-gate version (figure 2(d)). Recent ferroelectric capacitors further enable FG circuit capabilities (e.g. [56]); FG techniques can utilize many CMOS process improvements. Transistor density compared to a NOR digital memory cell could be slightly relaxed for added functionality, particularly analog functionality.

The FG charge is modified by the combination of transistor hot-electron injection to decrease the charge (equivalently increase electrons), and electron tunneling through a separate tunneling capacitor to increase the charge [53, 54, 57] (equivalently decrease electrons). Hot-electron injection operating with subthreshold and near-threshold currents require a channel current and high drain-to-channel voltages, resulting in a near-perfect programming selectivity for NOR configurations, as well as having FG charge updates that are nearly proportional to the signal current in the channel. The first property results in extremely high control of FG programming (e.g. 14 bit accuracy over near 1 million FG devices on an IC [58]). The second property enables continuous-time (CT) adaptation and learning using local signals. The tunneling and hot-electron injection voltages (e.g. 9–12 V and 4–6 V, respectively for 350 nm CMOS) are easily handled through a CMOS process (e.g. high-voltage transistors in figure 2(b), and can be generated on-chip using charge-pump circuits [59]. High-voltage handling circuits can be placed on the same chip next to precision analog circuits without affecting the long-term behavior of these FG analog circuits.

Earlier groups saw the hope for having a STLS analog NVM device in their initial research directions. Synaptics, a new startup company looking at NN-implementations, was experimenting with FG elements for weights for classifiers, seeing the potential for such embedded structures in commercial applications [36]. Intel created the ETANN IC using a specialized EEPROM process to enable programmable current sources for Gilbert-style multipliers in a NN chip [60]. Most importantly, Brooke's group (e.g. [61]) considered creating FG devices in standard CMOS processes using two-level polysilicon for bidirectional electron-tunneling to modify the FG charge for analog applications. Each of these approaches showed the desire for a STLS type analog NVM.

Analog computation enables highly energy-efficient computing (computations per unit energy) as well as area-efficient computing (computations per unit area). In 1990, Carver Mead hypothesized that analog computation would be at least 1000× lower energy compared to digital computation based on counting the transistors required for a fundamental operation (e.g. multiplication) [62]. A FG VMM experimentally demonstrated the analog computing efficiency compared to digital efficiency (e.g. digital energy efficiency wall [63]) in 2004 [64]. These results would be followed by other similarly efficient crossbar algorithms for adaptive filters [65], Gaussian mixture models (GMM) [66, 67], support vector machine [68], and VMM+winner-take-all (WTA) classifiers [69]. These computing approaches also experimentally verify Mead's hypothesis of 100X area improvements, packing more, smaller processors in the same area. These

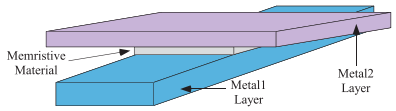


Figure 3. Basic concept and location of an *ideal* memristor structure. In this case, the routing pitch of two metal line levels sets the array density when not considering a separate selector device.

techniques also allowed for dense crossbar implementation of biological synapses including generation of biological post-synaptic potential (PSP) outputs [70], as well as CT biological synaptic learning rules, such as long term potentiation (LTP), long term depression (LTD), and pairwise [71, 72] or triplet [73] spike time dependent plasticity (STDP). Large arrays (30 000) of such learning synapses operating in a crossbar with a set of 100 reconfigurable neurons have also been demonstrated [74].

2.2. How SyNAPSE fueled nanodevice synaptic element research

Getting new technology opportunities into the consciousness of the wider technical community requires some significant event, and today's widespread interest in two-terminal nanodevices for crossbar computations started from the DARPA SyNAPSE proposal formulation [75]. Todd Hylton, a distinguished researcher in nanodevices, became a program manager at DARPA and arrived arriving at DARPA at a time when interest in neuromorphic systems was growing; for example Neovision [76], CT2WS [77] programs were already in place and some programs involved FG devices. He developed SyNAPSE [75] program in 2007 seeing that synaptic arrays might be an important first application for many two-terminal nano-devices where slow electrically-controlled movement of ions allow changes of a device property like its resistance.

At this time, one could find multiple researchers considering nano-devices having long and short-timescale behaviors as a result of ion movement (e.g. Kozicki in the 1990s [78–81]). One of the individuals Dr Hylton consulted was Stan Williams, who had been working on theoretical and experimental aspects of ion-movement devices for over a decade by this point, bringing this research directly to show the opportunities for these memristive devices [82]. The first memristor described and put into practice was by Widrow (1960s) [46] for his core adaptive filter element [83] (he referred to it as 'memistor'). In somewhat unrelated spaces, Chua formulated a memristive linear circuit theory element in the 1970s [84, 85]. As the program formulation continued, the connections to Chua's circuit theory became more apparent, dusted off, and became part of the story [86].

The DARPA SyNAPSE program fueled the great interest in nano-devices as synaptic devices [75], effectively echoing many of the synaptic requirements from the STLS efforts described a decade earlier [53, 54]. When the program was launched towards the end of 2008, every research group primarily looked towards these nanodevices as the path towards making large-scale networks of synapses and neurons; only one team (from HRL) had a single research group looking at an all CMOS solution to solve the proposal goals. DARPA SYNAPSE promised electronic systems (by 2015) to have the number of neurons and synapses found in the mouse brain in 1-liter volume.

An entire community embraced these new opportunities empowered by these new formulations. This program was instrumental in the demonstration of the first two-dimensional memristor crossbar arrays (2012) [87] three years into the start of the program, build on innovative device research for digital [88], synapse [89] and analog applications [90]. Memristor at its core is typically a nonlinear resistor (has some linear range) such that the resistance changes on a slower timescale (usually) as a function of the voltages applied. Ideally a memristor could just be the crossover of two metal wires with memristive material in between these two layers (figure 3). Theoretically the density might only be limited by pitch of metal lines (figure 3). At a simple level, a memristor can be primarily considered as a linear resistive element where the resistance (applied voltage (V) over applied current (I)) changes on a slow timescale:

$$I = G_0 W V,\ \tau \frac{\mathrm{d}W}{\mathrm{d}t} = f(W, V) \tag{2}$$

where τ is a normalization factor indicating the fundamental time constant typically related to the transport of ions, G_0 would be the nominal maximum conductance, which is around the typical operating conductance, for this device, and W is the normalized *weight* of the effective resistance. Multiple models have been produced, and yet, the complex transport dynamics in these devices is still incomplete and a current research question (e.g. [91]).

Although SyNAPSE would eventually move in different directions, these initial memristor efforts spawned a number of research efforts including VMM crossbar operations simulated and measured [92–95],

aspects of programming a two-dimensional array of devices [96–99], and discussing some implications of nonlinear device properties and circuit noise [93, 95, 100], particularly the high $1/f$ noise [95, 101]. RRAM are one form of memristors typically more amenable to integration into a CMOS process with memristive VMM computations and adaptive properties [102].

Recently, RRAM devices were built using electrochemical intercalation of guest ions into host structures rather than electrodeposition, an electrochemical memory (ECRAM) and in the process enables a potential third terminal for modulating the RRAM conductance and learning dynamics [103–113], similar to the original Widrow memistor [46, 83] that used liquid electrolytes. ECRAM is a solid state device, partially resulting from Li- ion battery advances [106]. ECRAM devices are fabricated in CMOS compatible materials and process with linear and symmetric switching, and scaled to sub-micron dimensions [105, 111].

For these comparisons, we will consider RRAM as the representative device from the memristive device family. Magnetic memory devices such as STT-MRAM typically exhibit a small range of resistances and is likely usable only for binary states [114]. Spin wave based neuromorphic computing [115] has been also proposed but no measured results are available and such devices are not considered here. Phase change memory (PCM) can be an option for analog NVM and CiM, often due to initial success in such devices commercially (i.e. by Samsung [116, 117]). Two-dimensional crossbar synaptic arrays can be built [118], and Micron started production of 1 Gbit memories in 2012. The PCM devices still remain slightly larger and require higher energy than their RRAM counterparts, primarily because a transistor per cell is often required for programming. Therefore, we will not consider these devices (or other devices that lack experimental measurements such as [119–121]) in details in these comparisons but rather focus on RRAM as the representative memristive device [115] (though see Comparison tables in Discussion section for some recent work using these devices).

The following sections compare between FG and memristor/RRAM nonvolatile storage (3), feedforward computation (4), programming (5), and learning algorithms (5), before summarizing and discussing the resulting opportunities for these techniques.

3. CiM requirement 1: nonvolatile storage

The first critical aspect of a crossbar NVM element is long-term storage of the resulting state over a reasonably wide range of user operating conditions. FG devices have demonstrated long-term (10-year lifetime) across multiple IC processes from 2 μm to 40 nm linewidths [19, 20, 31, 122]. The 10 year lifetime metric comes from the 10 year lifetime metric for the *normal* operation of a CMOS IC. A chip programmed in the factory would have roughly the same stored charge at its end of life.

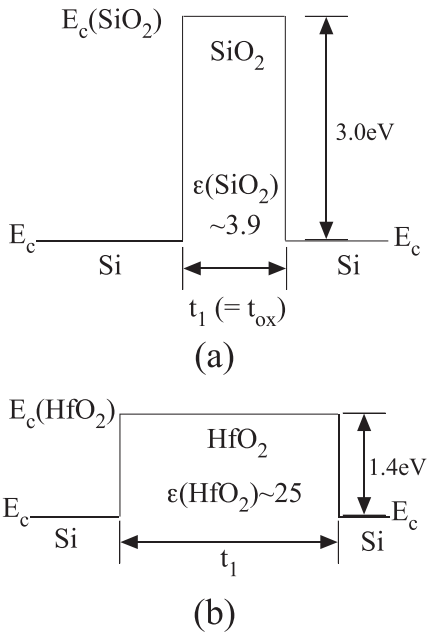
The focus of this section is to understand the NVM storage physics for FG and for memristor/RRAM devices. The questions center around the lifetime of stored charge in a crossbar array, as well as the number experimentally demonstrated stored levels in a crossbar array. NVM write stressing related questions are addressed in later sections (section 5).

FG devices achieve high charge retention over months and years by using high-quality gate insulators with sufficient thickness. The quality of these gate insulators needs to remain as ideal as possible, and therefore one should not utilize contacts and other routing on the FG node to avoid any process defects. The band-diagram between the FG and Si devices is effectively a large barrier of a particular thickness (t_1) for high-quality material (figure 4). The primary transport for electrons through this barrier is through electron tunneling. Tunneling arises from the fact that an electron wavefunction has finite spatial extent [37, 123], and therefore the electron has a finite probability of existing on the other side of the barrier. This situation leads to the classical quantum-mechanical expression for electron-tunneling current through a square barrier (width t_1 and size E_{barrier}) (figure 4) [37, 123, 124]:

$$I_{\text{tun}} = I_{\text{tun0}} \exp\left(-\frac{2\sqrt{2m^*}}{\hbar}\sqrt{E_{\text{barrier}}}t_1\right), \tag{3}$$

where m^* is the effective mass of an electron, and I_{tun0} is an experimentally determined constant for the particular insulator.

A Si-SiO_2 t_1 between 5 and 7 nm (figure 4) is sufficient to achieve 10-year lifetime metrics [37, 122–124], depending on the quality of the insulator. A typical FG device to change 1–100 μV at room temperature over a 10 year device (e.g. 500 nm, 350 nm CMOS) lifetime as characterized by accelerated temperature measurements [19, 20]. Process nodes below 350 nm CMOS have at least one additional a thicker transistor gate insulator, driven by economic constraints such as having I/O devices so the IC directly interfaces to board level infrastructure. High applied temperatures enables accelerating the electron leakage, allowing for estimates of 10-year lifetime measurements. Assuming a thermionic emission as a temperature extrapolation



C_T	1 e$^-$ V
1pF	100nV
100fF	1μV
10fF	10μV
1fF	100μV
100aF	1mV

(c)

C_T	%I
1pF	3ppm
100fF	30ppm
10fF	0.03%
1fF	0.3%
100aF	3%

(d)

Figure 4. Banddiagrams for a FG (a) conduction-band diagram from silicon subtrate through the SiO_2 insulator to the FG layer node. The barrier between Si and SiO_2 is roughly 3.0 eV. A typical insulator thickness, t_1 would be 6–7 nm. (b) For modern processes that use HfO_2 for their gate dielectric, the barrier energy decreases (3.0 eV–1.4 V), the thickness increases (e.g. 14 nm from 7 nm), and the dielectric (ϵ) constant significantly increases (25 from 3.9), resulting in better gate coupling as well as having a higher effective barrier for charge storage. The gate material might be Si or might be a different conductor. (c) Voltage change for a single electron on different size FG nodes described by C_T. (d) Percentage change for a subthreshold bias current ($\kappa = 0.76$), the highest resulting current error, for this single electron change.

model for the electron-tunneling and thermionic emission, one can model the FG charge with time ($Q(t)$) by the simple model [125–128]

$$Q(t) = Q(0)\exp\left(-tve^{-q\phi_b/U_T}\right) \tag{4}$$

where $Q(0)$ is the initial charge on the FG, v is the relaxation frequency of electrons in polysilicon, and $q\phi_b$ is the effective Si-insulator barrier potential (Volts). Typical numbers for a 500 nm CMOS process are 0.9 eV and 60 s^{-1} [20] and for a 350 nm CMOS process are 0.618 eV and 55 ms^{-1} [19], showing the influence of intermediate trap states in these long-term lifetimes. This classical model over-estimates the charge loss [20]. These FG lifetime measurements agree with measurements showing negligible FG change on remeasured analog FG devices (e.g. column of voltage sources) shelved for 10 years or longer.

Scaled down FG devices using non-SiO_2 materials, such as HfO_2 (figure 4(b)) in 40 nm CMOS show similar FG charge lifetime (e.g. [122]). The change in insulators do enable a thicker insulator (e.g. 14–15 nm) but with a smaller barrier potential (1.4 eV [129] versus 3.0 eV [124]); therefore for a square barrier we would expect lower leakage than the 350 nm device. Process nodes below 65 nm use thicker HfO_2 insulators for MOSFETs including pFETs including for bulk, SOI, and FinFET devices.

A related question is the number of experimentally demonstrated stored levels in a crossbar array. As FG programming has achieved near single-electron programming accuracies (e.g. [58]), the number of potential levels in a FG device is the voltage change due to a single electron on the total FG capacitance (C_T), or q/C_T. As the useful FG voltage range extends by more than 1 V from accumulation to far above threshold, the useful question is the granularity of the resulting voltage (figure 4(c)), as well as the percentage subthreshold current change that relates to the largest change for a given gate voltage change (figure 2(d)). For a small 100 aF C_T, smaller than a typical 14 nm CMOS device, one expects to have over a thousand voltage levels resulting in greater than 10 bit granularity, and a 3% granularity in setting a subthreshold current using these devices.

The FG voltage does not need to be biased within the power supply rails. The FG pFET is a standard pFET devices whose gate terminals are not connected to signals except through capacitors (i.e. no dc path to a fixed potential). With no DC path to a fixed potential, stored FG charge results in a FG voltage that can be inside or outside the power supply rails. The maximum FG voltage is limited by the programming scheme for the device [130]. Because the FG voltage (V_{fg}) is not limited by power-supply ranges, a crossbar FG pFET can be an analog component as well as act like an ideal switch (e.g. [17, 122]). For example, field programmable analog array (FPAA) devices utilize analog components in crossbar routing infrastructure any

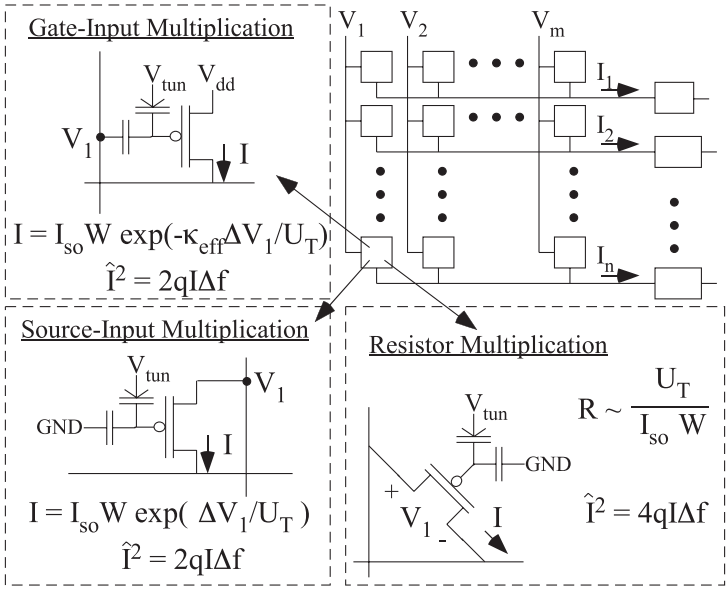


Figure 5. Several FG circuits for VMM computation. Gate-input multiplication: the broadcast input voltage can be coupled into the gate terminal. Source-input multiplication: the broadcast input voltage can be coupled into the source terminal. Resistor multiplication: a FG device operated in the ohmic region approximates as weighted conductance ($1/R$). The noise power is double the noise for a circuit configuration using saturated MOSFET devices.

element in a FG pFET array can be programmed to an *off* state (accumulation, <1 pA), to an *on state* (well above V_{T0} for all signals in the supply), or anywhere between the *off* and *on states* [17]. One effectively gets crossbar computations for free in an FPAA approach [131].

4. CiM requirement 2: effective VMM and related feedforward computations

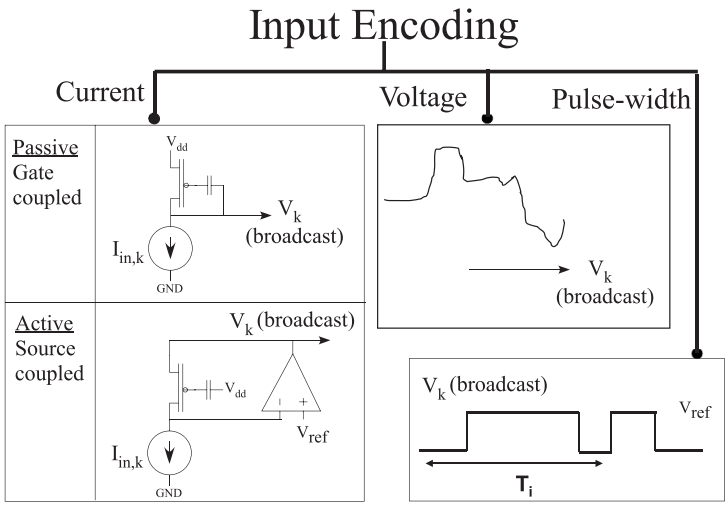
VMM computations arise from crossbar arrays of FG and RRAM/memristor devices. CiM minimizes the cost and complexity of communication for VMM and related computation [8, 10]; the cost of a computation is much less than digital approaches and similar to accessing a few rows of data from a memory, not to mention the cost of transporting that data to the processor [9]. The approach works best when continuously operating on a streaming input. Voltages are broadcast to the array elements (along columns), and output currents are summed on the output lines (along rows) using KCL (figure 6). The VMM operation is a *read* operation through this memory resulting in an important computation for multiple architectures (section 4.1). All these cases satisfy the required constraint for an area dense operation by using a single device, or only a few devices, depending on the computation. Achieving this requirement enables comparison of this computation along other lines that include questions of low-power & low-energy operation (section 4.2), questions of signal-to-noise ratio (SNR) & noise (section 4.3), as well discussions of density and scaling (section 4.4).

4.1. VMM CiM architecture

FG based VMM structures have multiple architectural choices (figure 5). Voltages can be broadcast along the columns to the gate terminals (Gate-coupled) [16, 64] or along the source terminals (Source-coupled) [132, 133] of the FG device. A FG device typically is biased in saturation [16, 64, 132, 133], although it can be biased in the ohmic regime [134], paralleling the VMM operation with resistive elements (e.g. [135]). The VMM input encoding using voltage input broadcast along the column (figure 6) can encode the input as a purely analog signal (amplitude coding) or as a pulse-width modulated digital signal [136]. The input signal can be encoded in current amplitude [64, 132, 133] through either a passive (e.g. diode connected FG device) or an active (e.g. a transconductance-amplifier (TA) based current mirror input stage) preconditioning circuit that generates a broadcast voltage. For a gate-coupled VMM, the output saturated subthreshold current ($I_{\text{out},ij}$) for the ith element on the jth row (assuming saturated) is given by:

$$I_{\text{out},ij} = I_{s0} W_{ij} e^{-\kappa_{\text{eff}} \Delta V_i / U_T}, W_{ij} = e^{-\Delta V_{fg}/U_T} \tag{5}$$

where ΔV_i represents the ith voltage input change from a reference input voltage ($V_{i,0}$), ΔV_{fg} represents the FG voltage change from the bias level ($V_{\text{fg},0}$), and κ_{eff} the capacitive control gate (C_g) the FG node that in turn, couples to the surface potential (κ) for the subthreshold device ($\kappa_{\text{eff}} = (C_g/C_T)\kappa$). This approach is



(a)

	Gate Coupled	Source Coupled
Ohmic	FG	FG, RRAM
Saturation	FG	FG

(b)

Figure 6. Multiple VMM input encodings and operating modes for NVM devices, both FG and RRAM devices. (a) The VMM input can be encoded in terms of current, voltage or pulse-width. Both passive and active (OTA based) circuits (e.g. current inputs) can be used to create the voltage to be broadcasted. (b) The VMM devices can use either ohmic or saturated, gate- or source-coupled devices (FG). Ohmic RRAM devices also use the ohmic, source-coupled topology.

often a prefered approach for FG devices given the input gate flexibility. While this topology was quite popular initially [64] and is used often in today's designs (e.g. [16, 137, 138]), one must be aware of the κ and coupling capacitance shifts and mismatch (e.g. 1%), issues that are not corrected by weight programming. For a source-coupled VMM [132, 133], the output subthreshold current for the ith element on the jth row (assuming saturated) is given by:

$$I_{\text{out},ij} = I_{s0} W_{ij} e^{\Delta V_i / U_T}, \tag{6}$$

This method is free of gate-coupling mismatch as the signal couples to the source of the FG transistor. Related to the source-coupled VMM topology (figure 5), we can have the FG biased in ohmic operation [134] and parallels the memristive VMM architectures [139–142]. The resistance (R_{ij}, subthreshold bias) of the ith element on the jth row is

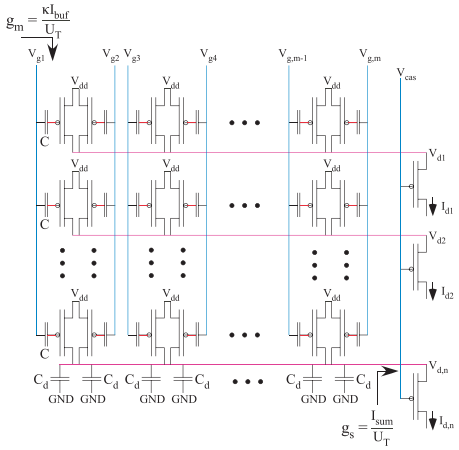
$$R_{ij} = \frac{U_T}{I_{s0}} W_{ij}, W_{ij} = e^{\Delta V_{fg}/U_T} \tag{7}$$

where the linear range of operation is limited to $\approx \pm U_T = 25$ mV of ΔV_i.

4.2. Low-power & low-energy crossbar operation

The next discussion is to consider the low-power/low-energy operation for a NVM crossbar computation, in particular the fundamental VMM operation discussed from its innovation [53, 54]. A VMM operation in an FG crossbar (figure 7) is experimentally known to have higher energy efficiencies ($\approx$1000$\times$ [64]) than similar digital computations, and has been mathematically modeled in multiple circuit topologies (e.g. [132]). A VMM computation can be modeled (figure 7) for the computation throughput (in Multiply–ACcumulate/s = MMAC(/s)) and computational efficiency (MMAC(s)/W). A per FG input gate capacitance (C) and source-drain capacitance (C_d) of 0.5 fF would be typical of a 130–65 nm CMOS IC process. A 256 $\times$ 256 array with 19 mW could enable 1 GHz, 134 TMAC(/s) operation and with 120 μW could enable 0.83 TMAC(/s) operation (figure 7). FG enabled CMOS transistors allow bias currents from sub-pA levels near accumulation to high above-threshold bias currents (e.g. 100's μA) from the same device that is programmed for a particular application. The desired input linear range (V_L) requires changing C_T of a given device relative to the input capacitor (C).

A 100 000 VMM element crossbar is typical for a range of acoustic applications [143], and the number of FG elements is an order of magnitude larger for the SoC FPAA [144]. Significantly larger VMM networks can



(a)

	VMM: n = 256, m=256 Theoretical Expression	Case I	Case II
τ_{in}	C n / g_m = n κ C U_T/ I_{buf}	77ps	12.8ns
τ_{out}	C_d m / g_s = m C_d U_T / I_{sum}	78ps	12.5ns
Power	V_{dd} (m I_{buf} + n I_{sum})	18.9mW	117μW
f_c	1 / (2 $\pi(\tau_{in} + \tau_{out})$)	1 GHz	6.3MHz
TMAC(/s)	2 n m f_c	134	0.83
TMAC(/s)/W	2 n m f_c / Power	7080	7060

(b)

Figure 7. FG VMM gate-coupled array. (a) VMM circuit structure illustrating the important analysis variables. (b) Analysis of a general and 256 × 256 subthreshold FG array (Voltage vector input, current-vector output) with $\kappa = 0.8$, $C = 0.5$ fF, $C_d = 0.5$ fF. Case I: $I_{buf} = 33$ μA, $I_{sum} = 41$ μA (160 nA/device). Case II: $I_{buf} = 200$ nA, $I_{sum} = 256$ nA (1 nA/device). $\kappa = 0.8$, $I_{dev} = 400$ nA, $I_{sum} = 100$ μA, $I_{buf} = 33$ μA, C = 0.5 fF, $C_d = 0.25$ fF).

be synthesized [145] using analog standard-cells formulated in multiple IC processes [137, 138, 146]. For a gate-coupled 200 × 500 VMM, the same computations show the change in energy/power as a function of frequency when scaling all bias currents. One expects that the operating frequency is proportional to the scaling of the bias currents.

An embedded application requiring significant 100 000 MAC elements (figure 8) provides a method to compare between FG and memristor crossbar approaches. The low memristor/RRAM computational resistance values sets the feedforward energy and power comparisons with FG devices. Memristor/RRAM devices tend to have measured *on* computational resistances, the VMM computation values, in the range of 10 kΩ or less. For example, handling the low resistance memristor values of 10 kΩ when building hybrid memristor + FPAA devices required some routing creativity as the configurable fully-*on* switches had a 5–10 kΩ resistance [147]; larger resistances were far more difficult to fabricate (this is an important future direction as mentioned in the Discussion section). A memristor/RRAM element required current and power is set by its average *on*-resistance (R_{on}) and V_{dd}; $V_{dd} = 1$ V is a typical comparison voltage and one can optimistically use an average $R_{on} = 10$ kΩ. For an $N = 100\,000$ acoustic application (10 kHz signals), a 350 nm CMOS requires 1.5 mW as compared with 5 W required for a memristive/RRAM array (figure 8). Such an application would typically require some preprocessing that have been efficiently implemented in programmable and configurable CMOS (e.g. [144]).

Memristor/RRAM devices require *on*-resistances greater than 1 MΩ to be competitive for crossbar compute-in-memory embedded spaces, requiring that *off*-devices are a few orders of magnitude higher resistance. The small conductances make memristor/RRAM crossbar array computations sensitive to parasitic resistances the routing lines, something rarely seen in transistor crossbar arrays. These issues were predicted from the formulation of the synapse project as the power required for a mouse-brain (1 kW, 50× a human brain) provided a related metric to accommodate higher-power devices [75]. The synapse project did hope to have synapses that would only require 10 pJ per feedforward synaptic operation, although it was not the most important metric used. A pFET transistor operating with 1 nA peak current over a 1–2 ms window

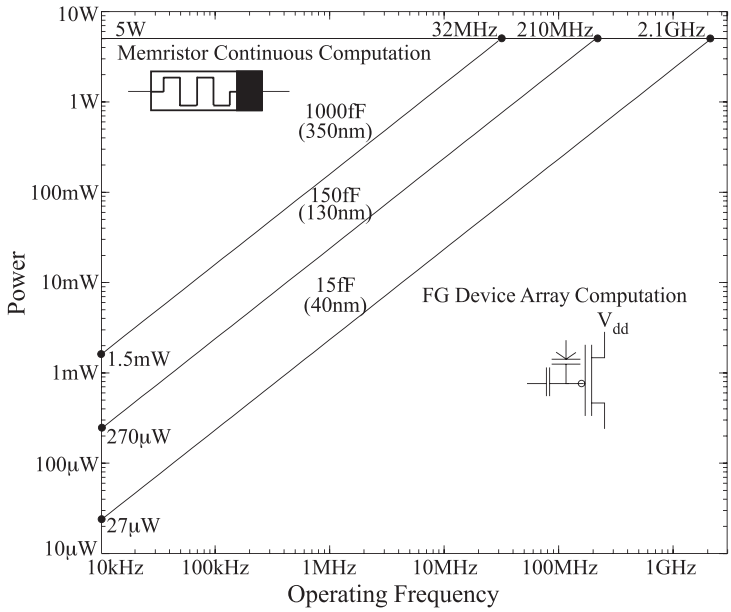


Figure 8. Energy comparison for a 100 000 MAC element VMM (200 × 500) FG (Gate-Coupled) and memristor/RRAM crossbar computations. The memristor device has an average computational *on*-resistance of 10 kΩ. The V_{dd} for the computation is 1 V. Half of the supply is applied across the resistance for the RRAM devices. The FG crossbar elements are considered using a total line capacitance, *C*, at 1 pF, 150 fF, and 15 fF levels, approximate estimates for routing capacitance in the VMM architecture for a 350 nm, 130 nm, and 40 nm CMOS IC implementations. Capacitance scales roughly quadratically with minimum channel length [122].

(<1 pJ per operation) easily achieves this metric even with the resulting control circuitry. Obtaining nA level memristive *on*-currents requires significantly higher resistances (e.g. 100 MΩ).

The comparison focused on average power efficiency, effectively energy efficiency, for CT computation as a fair comparison between two technology nodes. A technology can also be designed to run faster than the required frequency (e.g. FG programmed to μA currents for an application requiring kHz of bandwidth), and only turning the computation *On* for a fraction of the sampling duration, and otherwise turning the computation *Off*. This discrete time (DT) duty-cycled approach might make a higher power technology (1–10 kΩ on resistances) applicable to a lower power application and increase energy efficiency [148, 149]. The resulting infrastructure for turning the computation *On* and *Off* is rarely trivial and needs to be accounted in the application. Volatile SRAM based CiM for NNs [150] or image processing [151, 152] uses this methodology.

The earlier CT and DT approaches can be mapped to different applications. In general, we can classify the applications requiring embedded NNs into two categories depending on the throughput requirements: (a) real time (RT) and (b) As fast as possible (AFAP). RT can refer to always-on applications running on edge devices such as keyword spotting with continuous time feature detectors like neuromorphic cochleas [153], or epileptic seizure detection using analog filter bank based spectral analysis [154, 155] etc. These applications require high energy efficiency as well as low absolute power dissipation, due to constraints on battery and powering sources in wearables, implantables or sensor nodes in the Internet of Things (IoT) [156]. Moreover, the input data rate is fixed by a physical signal such as 10–20 kHz for audio processing or 100–200 Hz for EEG etc and hence there is no benefit in running the processor at a faster rate. CT computation directly matches this mode. AFAP applications can use these DT approaches, working with the earlier caveats, as well as requiring extra memory overhead for adaptation. AFAP cases are not limited by any data availability allowing the NN needs to be run at its maximum throughput and fastest speed. Server side computing is a good example of such cases; alternatively, running a generative ML application to produce an image or video at the edge is another example. Both CT and DT modes could be used in such applications.

4.3. Crossbar SNR considerations

SNR in crossbar array structures rests on the noise generated in each device as well as the topologies utilized with each device. Noise is composed of two components, thermal noise and $1/f$ noise, related to noise power spectrum for each approach. The thermal noise power from a saturated transistor with I_{bias} current is

$$\hat{I}^2 = 2qI_{bias}\Delta f.$$

An ohmic transistor biased around zero drain-to-source voltage would have twice the noise power, as would a resistive element. A FG device could operate biased with ohmic or saturated currents. In a crossbar, the saturated devices always have lower noise, and as these devices can operate over larger voltages, result in higher SNR, than the ohmic devices. Often individuals will want to use ohmic FG devices in a crossbar array inspired by the parallels of memristive/RRAM devices, unaware of the historical background in FG devices. A group reported using two FG devices to emulate and model a memristive device that is a very useful model of memristive behavior while completely missing the historical advantages originally developed for FG devices.

Typically FG devices are limited at the balance between thermal and $1/f$ noise, where memristive/RRAM devices are primarily limited by high $1/f$ noise, likely due to the lower thermal noise for the resulting bandwidth because of the low sl on-resistances. The low memristor/RRAM resistance and resulting high operating currents results in lower noise over the smaller bandwidth, lower noise than expected by kT/C as it would be spread over the wider bandwidth (not operating bandwidth). For a FG array, the bias current used is set to the level required for the desired output signal bandwidth to optimize energy efficiency. For memristive devices, the *on-resistance* sets the bias current and noise levels independent of the bandwidth during continuous operation. Memristor/RRAM devices tend to have considerable 1/f noise levels [95, 101].

Nonlinearities can restrict the largest input signal to be safely used. Both FG and RRAM/memristive devices would alter their stored values if sufficiently high signals were applied as those operations are part of writing these devices. Both FG transistors operating at or near subthreshold bias currents as well as memristor/RRAM devices have significant nonlinearities depending on the particular topology. Nonadapting operating signals for both approaches tend to be 1 V or less. Classical circuit techniques allow using multiple devices with differential signals to cancel out even-harmonic nonlinear behavior [65, 157].

4.4. Crossbar density considerations

Mesh-element density is part of the argument for memristive/RRAM crossbar devices over FG-based crossbar devices. A memristive/RRAM device could simply exist between the crossing of two metal interconnection lines resulting in very high density numbers. The SyNAPSE project expected these devices could easily fit into a 100 nm $\times$ 100 nm pitch. And yet at a similar time, FG flash devices were already commercially available (e.g. 32 nm) at that pitch (e.g. [41, 42]). Flash structures at 11 nm CMOS already reach 30 nm $\times$ 30 nm pitch. We continue to see the reporting of small memristive devices (e.g. 12–15 nm $\times$ 12–15 nm) [158], as well as smaller FG flash and EEPROM devices. FG crossbar devices are somewhat larger for good circuit properties and are organized in NOR configurations for highly selective programming; one expects that similar performance memristive/RRAM devices will likely be larger as well. In the end, one expects that memristive/RRAM devices will be smaller than their FG device counterparts, although not by a huge factor (e.g. 2$\times$), achieved through additional fabrication complexity and cost. These comparisons assume the memristive/RRAM device does not require a transistor for programming or selectivity; when a transistor is added, one finds little difference in the density comparison.

5. CiM requirement 3: effective array programming

The next two sections focus on the last component for a NVM crossbar memory that the weights can be programmed accurately, as well as that the weights can be adapted based on the applied signals in an outer-product rule (section 6). Programming is essential in setting devices to their desired starting values, enabling methods to remove the effect of mismatch on the feedforward computation, changing these values based on calculations, NVM programming requires discussions about the programming mechanisms (section 5.1), programming selectivity and resulting accuracy (section 5.2), changes in programming mechanisms (e.g. currents) after repeated programming (section 5.3), and programming infrastructure (section 5.4).

5.1. NVM programming mechanisms

This section reviews further details of NVM programming mechanisms that are required for the following subsections. The discussion starts with overviewing more details on typical FG programming, electron-tunneling and hot-electron injection, from its initial description (section 2.1). Then we continue to overview the relevant on memristor programming mechanisms.

Electron tunneling and hot-electron injection are used in different ways for FG crossbar arrays. Digital flash or EEPROM memories typically use electron-tunneling for programming devices, and may also use electron tunneling for erasing devices. In most cases, hot-electron injection is only used to erase a FG device using the FG nFET transistor operating in ohmic above-threshold operation. Analog & mixed-mode

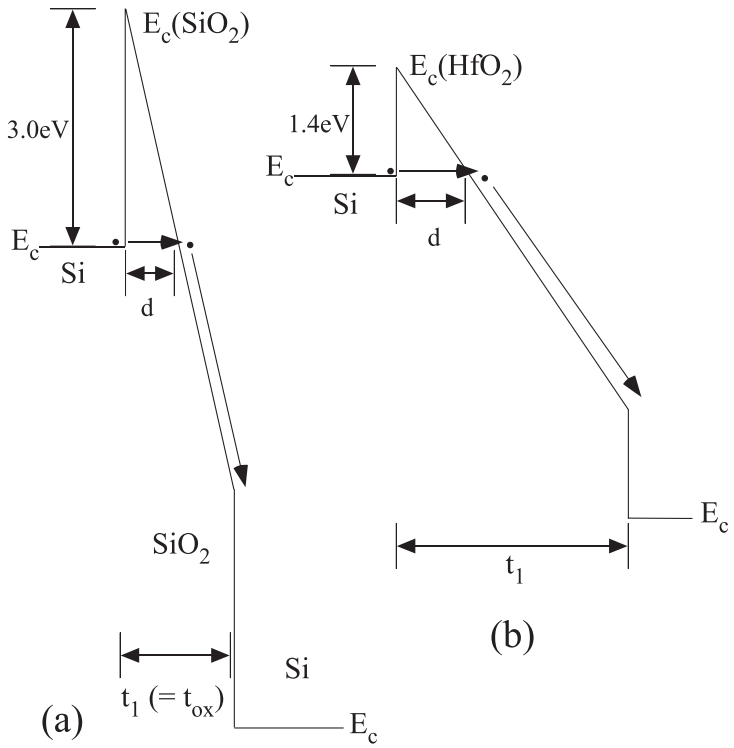


Figure 9. Band-diagram for FN-tunneling in silicon. Applying a voltage across the capacitor decreases the width the confined electron experiences from t_1 to d, exponentially increasing the probability the electron will tunnel through the barrier. The effective barrier change differs for a same 3 V applied across a (a) Si-SiO_2 capacitor, and across a (b) Si-HfO_2 capacitor.

crossbar arrays use pFET hot-electron injection for precision programming by remove charge from the FG, and electron tunneling for erasing devices by adding charge to the FG [54, 159].

Electron tunneling enables changing the charge at the FG dependent on the voltage applied across the barrier. Fowler–Nordheim tunneling, or tunneling through a triangle barrier, created by this applied voltage (figure 9). An electric field across the insulator, created by the voltage difference, reduces the thickness of the barrier to the electrons on the floating gate, allowing some electrons to move through the oxide. For a thin enough barrier (d), this spatial extent is sufficient for an electron to pass through the barrier (figure 9). FN electron tunneling current is modeled as [37]

$$\begin{aligned} I_{\text{tun}} &= I_{\text{tun0}} \exp\left(-\frac{4\sqrt{2m^*}}{3\hbar}\frac{E_{\text{barrier}}^{3/2}}{q\mathcal{E}}\right) \\ &= I_{\text{tun0}} \exp\left(-\frac{V_{\text{o}}}{V_{\text{tun}} - V_{\text{fg}}}\right) \end{aligned} \tag{8}$$

where q is the charge of an electron, and $\mathcal{E}$ is the electric field in the insulator, $\mathcal{E}t_1 = V_{\text{tun}} - V_{\text{fg}}$, and $V_{\text{o}} = \frac{4\sqrt{2m^*}}{3q\hbar}E_{\text{barrier}}^{3/2}t_1$, typically an experimentally measured parameter. Several experimental measurements verify this functional form (e.g. [159]), although the specific values for I_{tun0} and resulting barrier parameters often require detailed analysis to achieve good agreement between theory and experiment.

Hot-electron injection typically occurs when carriers in a MOSFET channel gain significant energy from the conduction band to then enable electrons to move into, and then through, the insulator layer to the FG node (figure 10). Although one could create other high-field regions to accelerate electrons above the Si-insulator barrier, these techniques are directly a function of the channel current as well as applied transistor voltages. Operating for subthreshold currents directly shows hot-electron injection is proportional to channel current where the high-field drain-to-channel region is minimally effected by the channel potential [160]. One can achieve nFET subthreshold and near-threshold injection (e.g. [53, 160, 161]) when one can operate the nFET device with subthreshold or near-threshold currents with the FG voltage operating above the drain voltage to move injected electrons to the FG node (e.g. $V_{\text{T0}} \approx 6$ V in [160]). Using nFETs operating with ohmic above-threshold currents also allows for the FG voltage to be larger than the drain

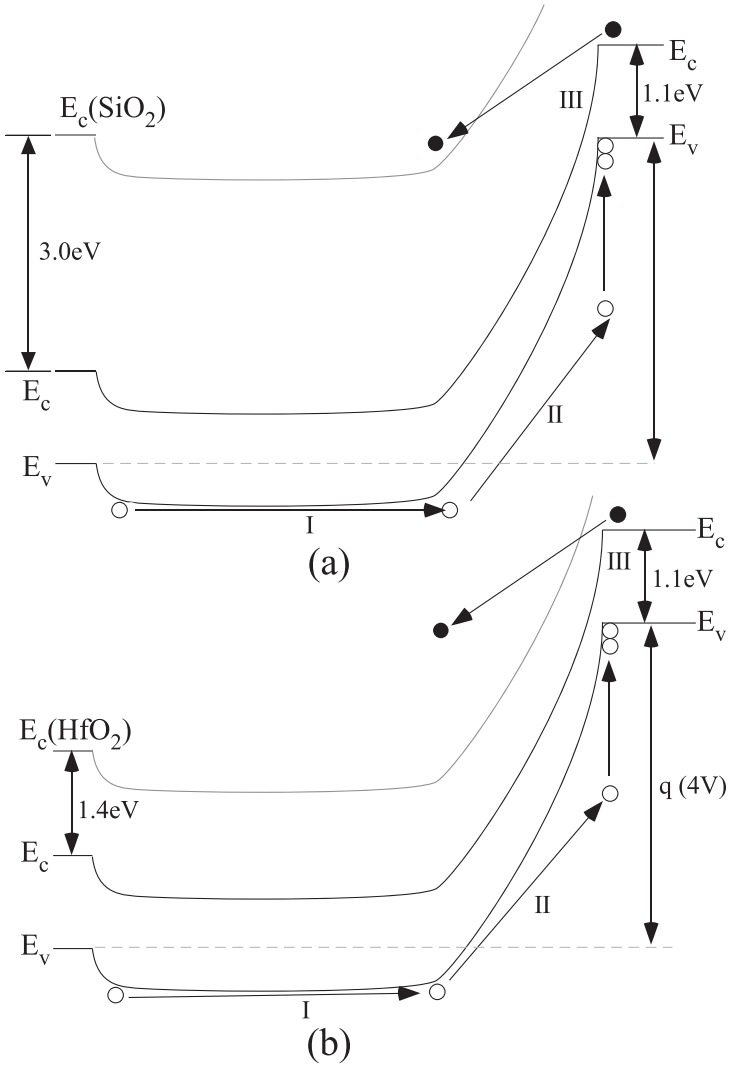


Figure 10. Band diagram for pFET hot-electron injection with (a) SiO_2 gate insulator and with (b) HfO_2 gate insulator. The hole diffuses across the MOSFET channel (I), gaining energy while potentially impact ionizing into two holes and one electron (II), and the resulting electrons correspondingly may gain sufficient energy traveling towards the channel region to surmount the insulator barrier (III).

voltage but with little fine-control of these mechanisms (figure 2), leading classically to the usage of these structures for erasing digital blocks.

In practice, a pFET transistor can both create sufficiently high-energy carriers (figure 10) and transport these carriers through the insulator using typically applied transistor voltages on typical pFET devices [162]. The fundamental model for hot-electron injection current (I_{inj}) is [160]

$$I_{\text{inj}} = I_s e^{f\left(V_{\text{fg}}, \Phi_{\text{dc}}\right)}, \tag{9}$$

where I_s is the channel current, V_{fg} is the FG voltage, and Φ_{dc} is the drain-to-channel potential for the pFET device; we often use a linearized exponential function for injection current

$$I_{\text{inj}} \approx I_{\text{inj0}} \left(\frac{I_s}{I_{s0}}\right) e^{\Phi_{\text{dc}}/V_{\text{inj}}} \tag{10}$$

where V_{inj} represents the one parameter for this linearization. The exponential dependence of drain voltage on injection current will be utilized to enable a wide dynamic range of programming step sizes with linearly-scaled, lower-precision gate and drain voltages. The percentage of flow of electrons arriving on the FG compared with the channel current in a 350 nm CMOS process for a 6 V drain voltage tends to be around 100 ppm or 1 in 10 000. Scaled down IC processes (e.g HfO_2 insulators), as well as operation at cryogenic temperatures tend to increase these efficiencies; hot-electron injection becomes a function of impact ionization with a significant percentage (e.g. 1%–10%) then transporting into the gate.

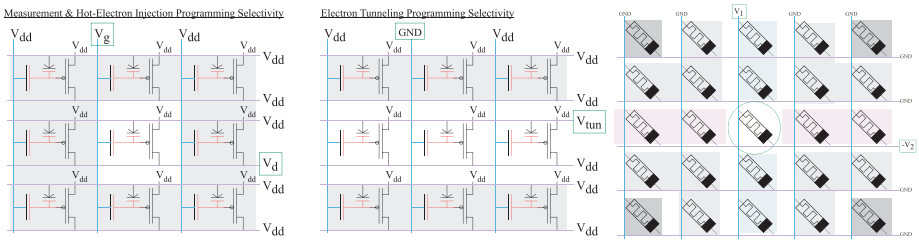


Figure 11. Illustration of methods and effect of FG selectivity for (a) measurement and hot-electron-injection programming, as well as (b) electron-tunneling programming. (c) Illustration of memristor/RRAM programming selectivity by applying a single voltage (V_1) along one column, and applying a single voltage ($-V_2$) along a single row. The primary effect of applying these voltages will be for the selected device, and yet, nearby columns and rows will also be impacted by these voltages.

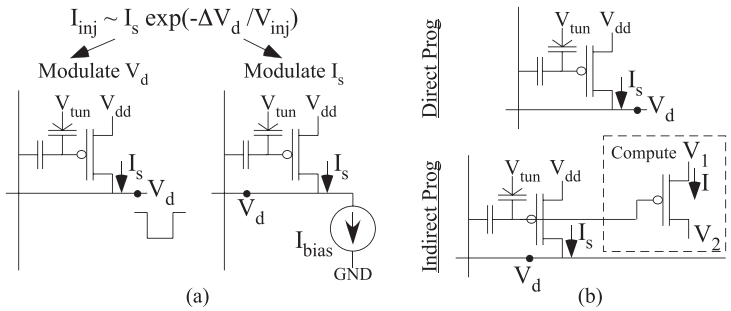


Figure 12. Multiple forms of hot-electron injection FG programming. (a) Hot-electron injection FG programming is a product of the channel current and applied voltage. Programming can occur by applying a voltage to the drain (V_d) line or applying a current to the drain line. (b) FG programming either can use direct programming, where the same FG transistor is used for programming and computationi, or indirect programming, where different FG transistors are used for programming and computation. Indirect programming is utilized primarily when programming a heterogeneous set of functions in a crossbar array (e.g. FPAA), where direct programming is utilized primarily when programming a homogeneous set of functions in a crossbar array (e.g. custom VMM).

5.2. Selectivity and accuracy

Selecting and accurate programming single devices is essential to crossbar computation as one typically expects arrays of thousands of NVM per mm^2 die area. Lack of selectivity requires significant complexity, cost, and resources to attempt to overcome this selectivity, typically overwhelming any advantages gained in other areas (e.g. energy efficiency). Other metrics can be compared once accuracy and selectivity are established.

Digital memories can achieve excellent individual device measurement and good programming selectivity through NOR structures; NAND flash's higher density requires block programming. EEPROM NOR structures find that coupling between adjacent lines and gate conductors can effect the selectivity [163]; the analog FG crossbars typically have fixed voltage lines between the crossbar signal lines, effectively eliminating this effect by shielding. Analog and mixed-signal crossbar programming utilizes a NOR configuration (figure 11), enabling near perfect device selectivity for measurement and for hot-electron injection programming [159]. Because hot-electron injection requires the product of current and applied drain voltage from (10), enabling only one column with channel current and only one row with applied drain voltage results in only the one pFET FG node to change its charge (figure 11). Routinely on the order of 1M individual FG devices (e.g. [17] are programmed over a wide dynamic range (multiple V on FG node) at a high precision (>14 bit [58]). A range of configurations enable hot-electron injection programming (figure 12). These techniques involve hot-electron injection programming involving subthreshold and above threshold saturated pFET transistors [122].

Electron tunneling potentially allows for good although not excellent selectivity. In a crossbar FG array, the tunneling voltage line is broadcast along a row and the gate coupling line are broadcast orthogonally along columns (figure 11). By applying a more positive voltage on a tunneling line and a more negative voltage on a gate line, the resulting FG to tunneling voltage will be largest for a single device, resulting in the highest tunneling current for that device. All rows where a tunneling capable voltage is not applied results in no tunneling and perfect selectivities among other rows (figure 11). Along the row with the tunneling capable voltage, we will have the largest tunneling current, with other devices along the row having lower, although not zero tunneling current (figure 11). The very sharp exponential dependence of tunneling

current with applied voltage results in a high tunneling current ratio between selected and unselected row devices, but the other row devices will disturbed [53, 54]. The mismatch between tunneling junctions typically results in 2–3× difference between two nearby devices for identical applied voltages, with a mismatch that can easily be higher for less well controlled insulator layers. This mismatch only increases the probability and extent of disturbing other devices along the row to be tunneled. This lower selectivity is a primary reason that digital EEPROM memories are limited to programming 16 levels (4 bits) in a single memory cell [41–45]. Tunneling devices do not have perfect selectivity because of the larger infrastructure requirements for a high-voltage amplifier in a particular cell requiring high-voltage handling devices (e.g. figure 2(b)). Because of these concerns, electron tunneling is often used as a block erase mechanism in analog/mixed-signal crossbar applications (e.g. [17]).

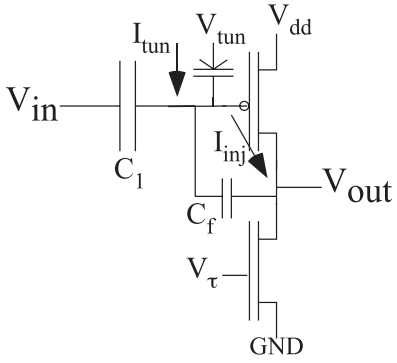
Two-terminal devices memristors/RRAM devices tend to have poor crossbar programming selectivity. The situation for a memristor device parallels tunneling junction programming with a shallower nonlinear function compared to electron tunneling. Memristor/RRAM crossbar arrays have significantly higher challenges As the memristor/RRAM crossbar is a mesh of interconnected resistances, a voltage applied on the column and row will not only affect other devices on that column or row, but will also have weaker effects on nearby columns and rows as the voltage diffuses between these local nodes (figure 11). A precise pattern of voltages could be applied along the columns and rows after solving a partial differential equation (PDE) for the resistive network requiring a very significant amount of computational cost as well as additional infrastructure. Although sometimes one or more FETs are used for programming to obtain some level of selectivity, it is no better than using FG type devices in a crossbar array, a structure requiring only a standard CMOS process.

5.3. Effects of repeated programming and charge trapping

A reprogrammable memory implies writing to that device many times with more or less the same properties. When programming precise values, device properties may change over time reducing the ability to program at the same initial precision as well as having reasonable confidence when developing adaptive systems. These issues relate to the material quality available for these operations, and engineering design using the available materials to minimize change during a write operation. where one designing these systems.

As one sees in FG approaches, the charge moving through the insulator can be temporarily stored in the insulator due to nonidealities in the material. Charge traveling through the insulator frees up trapped ions (e.g. H_2) that leaves wells for electrons temporarily to be trapped in that location. Most of these imperfections are at the insulator interface with Si in a typical gate insulator, although they will be more homogeneous through a lower quality insulator (e.g. source-drain spacer SiO_2). First, these traps change the current–voltage characteristics, such as changing the resulting tunneling current for the same applied tunneling voltage. Second, these traps hold charge that were intended to modify the FG charge and can move either towards the FG or other nodes over a timeframe of hours changing the resulting measured electrostatics. Sometimes these changes are incorrectly perceived as the FG node leaking charge as the effect does stop over a few hours to 1–2 days, and yet, the charge shift must be considered as it affects the overall accuracy of the solution. The programming sequence must take into account the operational voltages after programming to make sure insulator charge movement has settled near zero to minimize the effect on the resulting computation. Charge trapped near the transistor channel–insulator boundary should be pushed back into the transistor channel. If one does not address this issue, one might believe incorrectly that one requires reprogramming FG devices at a regular intervals of 1–2 days for high-precision storage.

Characterizing how NVM writing mechanisms (e.g. FG hot-electron injection and electron tunneling) change over time for particular devices directly impacts the device design and choice of materials. Although a FG cell could be written and rewritten many times to characterize its writing behavior, one only gets an indirect measure of the writing device effect. The autozeroing FG amplifier (AFGA) circuit (figure 13) provides a direct measurement by considering the FG node as a circuit node [164]. At this node, the tunneling current acts like a current source setting the bias current and equilibrium point for the balance of electron-tunneling and hot-electron injection currents. The AFGA circuit shows high-pass characteristics, where large positive changes from equilibrium are restored through the constant tunneling current source, and large negative changes from equilibrium are restored through the hot-electron injection dynamics. An occasional upgoing and downgoing step response allows direct measurement of the FG programming mechanisms while effectively keeping a constant high programming current flowing through the devices. This data can be used to directly characterize as well as design effective long-term memory writing strategies (figure 13). These measurements show a small change (e.g. 10%–20%) in the tunneling current (I_{tun}) over significant total charge (Q_T) through the insulator (e.g. 0.1–1 μC), corresponding to writing/erasing a device trillions of times [164]. The amount of change in tunneling current decreases with continued characterization time. The pFET hot-electron injection has nearly no change (figure 13), as any created



(a)

Item	Value
I_{tun}	0.5pA
C_f	3pF
Time	145hours
Q_T	0.26μC
Writes	16.1M

Item	Shift
I_{tun}	14.3%
V_{inj}	≈ 0

(b)

Figure 13. A FG autozeroing FG amplifier (AFGA). (a) An AFGA enables continuous gate-insulator currents while characterizing the effects of these gate currents. The feedback capacitor, C_f goes between the FG and output node. (b) Measured parameter shifts for an AFGA circuit operating for two weeks [164]. This data was measured in a 1200 nm CMOS process; values are typical with other IC processes. The FG transistor was untouched before the start of this experiment. Writes are normalized to moving 1 V of charge on a 16 fF capacitor (16 fC), typical for a 1200–350 nm CMOS process.

electron traps have no impact on the channel current or applied fields in Si or the insulator [164]. Results in other IC processes qualitatively follow these measurements, as well as years of repeated programmability on ICs on the order of one million of FG devices (e.g. [17]). As a result, one can expect billions of write operations using FG devices for programming operations using typical analog programming techniques (e.g. [58]), as well as using continuous insulator currents for long sustained periods of time (e.g. months).

One might wonder why there is any concern about changes in writing to FG devices. Typical EEPROM devices often state their cells allow a maximum of 100 000 writes to a particular cell. What is the difference? EEPROMs use lower quality insulators for their tunneling junctions that are used for device programming. In a double-poly CMOS process, one can tunnel through the two polysilicon layers (e.g. [61]), but the tunneling junction current changes by orders of magnitude after a few pC of charge through a nominal insulator (e.g. 10–100 fF capacitor), requiring significantly higher programming voltages, voltages typically out of range for designed supply voltages for the IC. The insulators in an EEPROM are similar to the double-poly CMOS devices. In practice, most EEPROM devices are used for block-memory storage, similar to a hard-drive, resulting in only a few thousand writes ever occurring for most devices, and error correction being used to mask any remaining errors. The devices used for the AFGA measurements as well as other applications (e.g. [17]) utilize pFET injection for programming, and electron tunneling occurs through a MOS capacitor gate insulator. Using charge modification through gate insulators minimizes tunneling junction shifts as gate insulator is tightly controlled by the fabrication process.

5.4. NVM programming infrastructure

Circuits to interface, control, and program NVM elements in an analog or mixed-mode system directly impacts the overall system area, cost, and performance. On-chip NVM Programming and Measurement time directly impacts all system integration costs and therefore eventual adoption of these approaches. As the FG devices have the most advanced programming infrastructures (e.g. [58]), the discussion focuses first on these techniques.

A generic structure, such as a mixed-signal FPAA, transforms into an FG crossbar array for programming (figure 14) as FG elements have near infinite measurement and programming (through Hot-electron Injection) selectivity [8, 58, 159], Hot-electron injection programming enables precise and nearly ideal selectivity when programming FG devices, while the typical mismatch for electron tunneling (lower typical mismatch than RRAM devices) tends to reserve this function for block erase of these devices. This transformation of heterogeneous circuits between operation, or *Run* mode, and programming, or *Program* Mode, allows for a single, or multiple parallel, FG crossbar arrays for programming.

This transformation is the start of the FG programming approach. The FG crossbar isolation (section 5.2) effectively allows one to abstract the entire programming process to measuring and programming a single device through its selected gate and drain voltages. This on-chip infrastructure (figure 14) requires supplying the resulting gate/drain multiplexer for the device, supplying DACs to control these selected gate and drain voltages, supplying the current-to-voltage as well as analog to digital conversion components, as well as supplying the digital control infrastructure (e.g. microprocessor, state machine) for controlling the programming. The programming algorithm using this infrastructure must iterate to

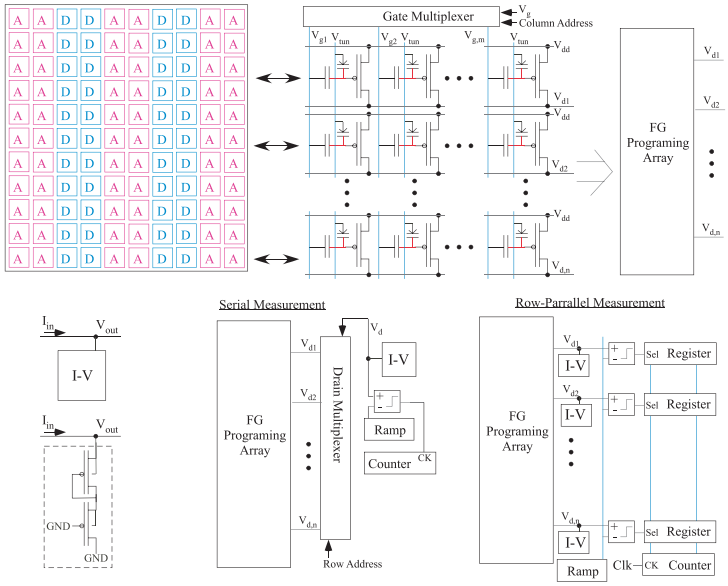


Figure 14. Programming a configurable analog (A) and digital (D) array, typical of an SoC FPAA, requires reconfiguring the FG array elements to a crossbar, or multiple crossbars, of devices. At the row-wise drain terminals for the programming array, one can have a *serial measurement* of each FG device or a *row-parallel measurement* of the FG devices. Measuring the FG device requires the longest time of any programming step, even with the on-chip current-to-voltage (I)–(V) transistor element.

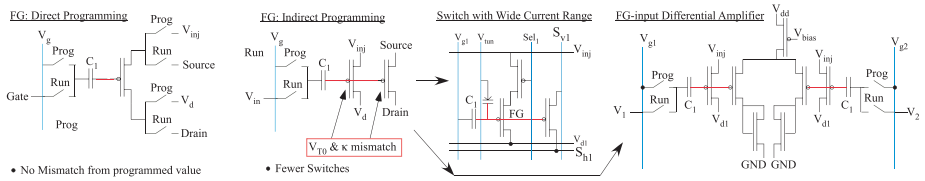


Figure 15. The tradeoff between direct and indirect FG programming. Direct FG programming results in little to no mismatch between the programmed and operating FG values for a single device, although requiring more switches in a general configuration to switch between the operating mode (Run) and programming mode (Prog). Many configurations could move these switches to the edge of an array or island of devices. Indirect FG programming significantly reduces the number of switches required, including switches directly in the device datapaths that could affect performance, but introduces V_{T0} mismatch and gate to surface-potential coupling (κ). Indirect FG programming enables the FG *switch with wide current range* and *FG-input differential amplifier* used throughout FG FPAA implementations.

approach the programmed values without overshooting any of the targeted values. The measurement system (figure 14) can either measure each FG device in a serial manner, typical to existing SoC FPAA devices [17, 58], or could parallelize the current to digital measurement by measuring each row or measuring blocks in parallel [165]. Heterogeneous FG circuits with around 1 million FG devices are routinely programmed completely on-chip to high accuracy (e.g. 14 bits) [58].

The transformation between *Run* and *Program* creates multiple circuit choices on how to make this transformation. This transformation could require multiple transmission (T-) Gate switches for a particular configuration, although these transformations can make use of multiple transistors connected to the FG node. Direct programming (figure 15), where the same pFET device is used for operation and programming, can require additional switches, while being certain the programmed behavior directly translates to the operational device behavior. Indirect programming (figure 15), where a different pFET device is used for operation as compared with measurement and programming, significantly decreases the number of reconfiguration switches, including switches in the datapath, while incurring ΔV_{T0} mismatch, and to a lesser extent κ mismatch, between the two pFETs that should be calibrated. FG devices can correct for device mismatch either in production programming or for an often reprogrammable device given a calibration step [166]. Several circuits currently utilize indirect programming (figure 15), such as the SoC FPAA switches [17] enabling a wide programming range (pA to 100 s to μA), differential pairs with FG input transistors [17], and realistic neuron models [74].

Handling the higher-voltage tunneling lines as well as generating the higher voltages for electron-tunneling and hot-electron injection completes the FG infrastructure questions. The best approach is generating tunneling voltages and injection voltages using charge-pump circuits [59, 167, 168], and often

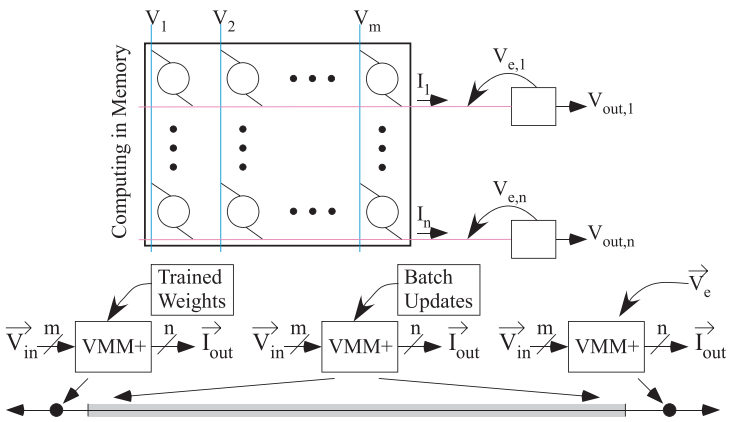


Figure 16. Potential crossbar NVM learning approaches based on an a function of the input voltages ($V_1, \ldots, V_m$) and the error voltages (V_e). These approaches potentially allow for capabilities between programming of the crossbar network, through a whole range of batch training of the weights, to full continuous-time update of the crossbar weights as a dynamical systems.

having multiple charge-pump circuits to independently program blocks of FG devices. Charge pumps require charging and discharging capacitors at a particular frequency; distributing charge pumps tends to require smaller capacitors at these blocks with the same total capacitance as one has for a single charge pump element. Routing these high voltage lines (e.g. 12 V lines through a 1 V–2.5 V process) is straight-forward with near-zero risk. The primary issue is local substrate or well inversion under polysilicon or metal lines that typically results in some initial junction breakdown with its resulting higher junction currents to supply the tunneling voltage. For some initial FG implementations in 2 μm CMOS that required 40 V for electron tunneling, the polysilicon routing that is at a higher thickness than gate insulator thickness would invert the channel resulting in higher impact ionization currents at the well to substrate junction. In IC processes from 800 nm and below, these issues are not observed. Metal layers, starting from the first level metal layer, typically would not create any breakdown problems due to an inverted substrate. High-voltage transistors (e.g. nFETs) using wells as source-drain terminals enables some high-voltage circuitry (e.g. [169]), although they often require significant bias current. The required electron tunneling current tends to be small for an entire array, and therefore the power required for a tunneling erase during programming tends to be small, and effectively negligible for the rest of the programming or system operation.

6. CiM requirement 4: learning algorithms on physical mesh architectures

The last capability of an NVM crossbar network requires that the mesh element can adapt its parameter (weight) value through a function based on the locally available broadcast input and error voltage signals. Both FG and memristor/RRAM devices are capable of this operation, and yet the next question is whether either or both devices are capable of useful operations and how do these devices compare when performing these useful operations. FG based adaptation (e.g. figure 13 [164]) provides the foundation for FG learning, including least mean square (LMS) case [65], VQ type learning [68, 170], and biological neuron + learning synapses in a range of crossbar networks [71, 74, 171]. Everything previously required (e.g. low energy/power) for the feedforward networks still applies in this context, while only requiring at most a minimal change to the mesh circuitry.

Adaptation in a NVM crossbar array builds additional infrastructure from the voltage inputs and output current outputs by modulating the voltage of the output line (figure 16). CT crossbar learning (figure 16), based on LMS learning, one must have a current out ($\mathbf{y}$) without being affected by the applied error voltage ($\mathbf{e}$):

$$\mathbf{y} = \mathbf{W}\mathbf{x}, \mathbf{e} = \hat{\mathbf{y}} - \mathbf{y}$$

$$\tau \frac{\mathrm{d}\mathbf{W}}{\mathrm{d}t} = \mathbf{x}\mathbf{e}^T \approx E\left[\mathbf{x}\mathbf{e}^T\right]. \tag{11}$$

Training and learning occurs at a slower and ideally non-overlapping timescale than the feedforward computation aggregating statistics to update the weights. Analog numerics utilizing continuum of real-value representations are well suited for these low-pass filtering computations [172].

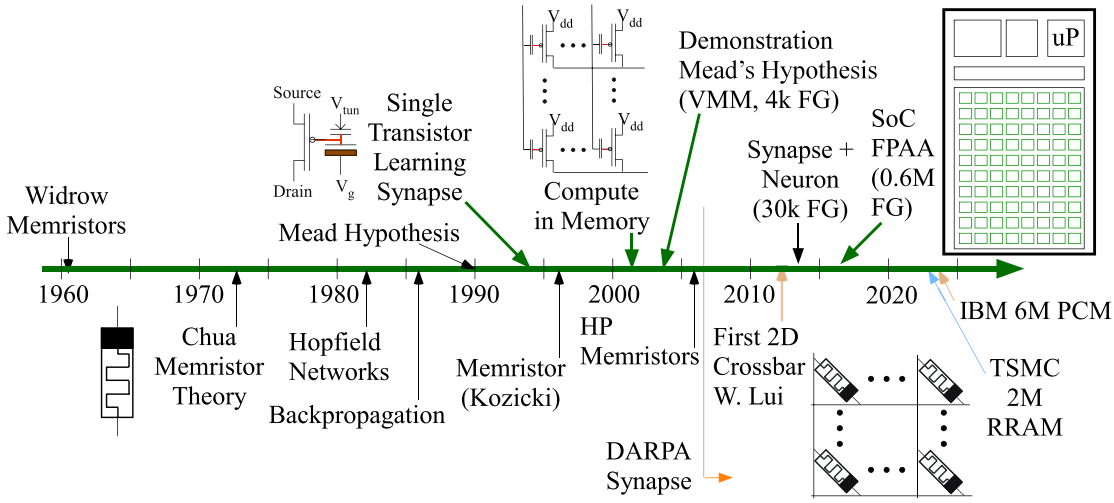


Figure 17. Timeline showing important events in FG, RRAM, and related crossbar efforts as well as related achievements in this area.

NVM approaches for implementation of trained crossbar networks could include a continuum between programming a previously trained set of values to full CT update of the crossbar values (figure 16). The intermediate forms are methods where the feedforward computation is paused for a time to update these values and/or methods where statistics are aggregated at some timescale outside of the crossbar network (figure 16). If the weight update dynamics are on a slower timescale than the feedforward computation, there are advantages to only updating the weight occasionally. The implementation choice depends highly on the application as well as the available hardware capabilities. For example, when using an FG FPAA device that did not enable continuous adaptation [144], an on-chip learning algorithm utilized batch programming to efficiently use the FG programming infrastructure already available [173, 174]; the approach might look different for an FPAA device that did enable adaptation [130]. FG devices can implement the entire continuum of learning approaches. Hardware efficient adaptive algorithms for WTA terminations for a VMM (or related synapse) block [173, 174] enables using a wide range of output elements at the output (e.g. [175–177]) including allowing for selecting these particular output elements.

Implementing CT adaptation using two-terminal memristive/RRAM devices faces challenges as modulating the error-side voltage effectively creates an additive term on the output computation from every weight element. Most memristor/RRAM approaches apply learning updates in-between feedforward computation. For LMS type learning algorithms, the ideal circuit element is a current source that outputs the same current independent of output voltage (e.g. drain of a saturated FET) versus a resistive element. For both devices, an important future opportunity is developing architectures for learning algorithms with second order momentum terms which are common in most state-of-the-art algorithms today. Also, introducing short-term plasticity along with long-term ones in future will help mimic biological synapses better and provide interesting filtering properties.

7. Summary and discussion on FG and memristor/RRAM learning devices

This discussion addressed the capabilities and opportunities of NVM crossbar in-memory computation, primarily focused between FG and memristor/RRAM devices. A historical perspective (figure 17) on the rise of NVM crossbar in-memory computation, starting from the STLS (1994) [53, 54] provided the foundation to enable the discussion of nonvolatile storage, feedforward computation, programming, and learning algorithms. The potential of scaled FPAA devices shows immense opportunities for FG-enabled systems [143, 178]. The historical foundation is necessary otherwise we might find research groups trying to use FG devices as a two-terminal device to model a memristor so they can use them in a crossbar network [134], not realizing FG devices started our current understanding of crossbar networks. These discussions focused on framing the each computational aspect as well as experimental measurements of system results. In addition to the academic interest, commercial approaches have found interest in FG (Synpaptics, GTronix, Impinj, Mythic [179]) and memristor techniques.

In the following subsections, we will summarize some comparisons between these NVM devices (section 7.1) as well as discuss some of the peripheral and edge circuit issues (section 7.2), and then finish with some concluding thoughts.

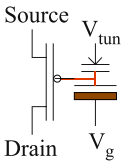
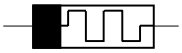
VMM & Related Operations	 FG	 Memristor (RRAM)
Retention	> 10 year lifetime $\Delta V_{fg} = 1 - 100\mu$V (500, 350, 180nm) [16, 19, 20, 134]	some $> 10year$ measurements [180] Programmed level measurements [181]
Number of Write Operations	> 1 Trillion [164]	Large number of cycles on materials [182]
Prog. Precision (Measure)	10bit [64, 74] -14bit [16, 144] $< 0.2\%$ - 1% Wide subVT / above VT range [16, 58, 64, 134]	1bit [183]- 5bit [184]
Current (A) / R(kΩ) Range	< 1pA to $> 10\mu$A [58]	1-4000 [141,183–186]
Configuration	1-2 Transistors, 1-3 Capacitors [144]	1 Transistor, 1 Resistor (1T-1R), Processing Interface [141,183–186]
Fully Continuous Adaptive Operation	Adaptive Filters [65], STDP [74]	Only Sampled Adaptation On-Chip Learning
Energy Efficiency (VMM, Peak)	3.7 - 14.7 TMAC(/s)/W (500nm, 350nm)	0.187 - 11 TMAC(/s)/W

Figure 18. Summary of FG and memristor (RRAM) properties and relevant comparison metrics.

7.1. NVM summary comparison

After addressing the NVM based VMM systems, comparing FG and memristor/RRAM devices shows a number of important trends (figure 18. The efficiency of memristor) systems can reach the two-decade FG-enabled energy efficiency demonstrations at least at one operating frequency, although FG-transistor approaches achieve the high energy efficiency over many orders of magnitude of operating frequencies, resulting in a true power-delay product. Most of the resistive implementations use a transistor to allow for programming selection (1R-1T), increasing the device size to be at least the size of FG elements, particularly given the area required to interface between the transistor and Memristor/RRAM element, typical of other systems. In general, FG based systems store more number of bits per cell than their resistive memory counterparts (figure 18). Having 6–8 or more bits of programmed precision is essential for most analog computing, particularly to remove issues of device mismatch, an issue more acute in RRAM/memristor implementations. For example, recent CiM implementations using RRAM devices [187] showed effective resolution of 4.5 bits being limited by conductance drift. Designing better devices or algorithmic solutions to this problem remains an important future aspect of research. The FG-device long-term retention (10 year) and number of write cycles (1 Trillion) continues to be seen using these only CMOS-process devices across process nodes (e.g. [122, 138], although explicit 10-year lifetime characterization of advanced process nodes would further add to this story. While 10-year lifetime for binary RRAM has been demonstrated [180], characterizing and improving retention for multi-state RRAM remains an important focus of future work. Although very few FG systems demonstrate CT learning, some approaches use these concepts (LMS [65], STDP learning [74, 134]) and continuous adaptation remains an important opportunity as learning in memristor/RRAM systems require sampled adaptive systems. FG-devices are still possible in some form across every IC process node, from 2 μm CMOS to 350 nm [144]/65 nm [138]/28 nm CMOS and FinFET CMOS process nodes (e.g. 16 nm/14 nm CMOS). The FG device scaling and characterization will continue for the FPAA and analog/mixed-signal standard cell directions [122, 137, 138, 146, 178]. Memristor devices still require additional process steps and only now are being integrated as research (non-production) elements.

A more detailed look at representative FG and additive device implementations shows this wider gulf between FG & memristor/RRAM, and shows PCM and magneto-resistive random-access memory (Magnetic RAM or MRAM) compare positively to RRAM implementations (figure 19). In terms of encoding, analog

FG Crossbar Implementations

	CICC '04 [64]	JSSC '07 [188]	JETCAS '11 [132] TVLSI '16 [144]	TBCAS '13 Spiking [74]	TNNLS '17 [16]	Nature Electronics '19 [134]
Process (nm)	500	500	350	350	180	180
VMM Topology	Gate coupled, Saturated	Source coupled, Saturated	Source coupled, Saturated	Gate coupled, Saturated	Gate coupled, Saturated	Source coupled, Ohmic
VMM Input encoding	Current	Current	Current	Voltage	Voltage, Current	Voltage
Array size	128×32	14×720, 720×24	2×2 >600k FG	300×100	785×128, 64×20	12×8
Programming mechanism	HEI, FN tunnelling (erase)	HEI, FN tunnelling (erase)	HEI, FN tunnelling (erase)	HEI, FN tunnelling	HEI, FN tunnelling (erase)	HEI, BTB tunneling
Programming topology	Direct, constant voltage	Indirect, constant current	Indirect, constant voltage	Direct, constant voltage	Direct, constant voltage	Indirect, constant voltage
Cell Area (μm^2)	> 25	-	25	< 100 (bio synapse)	1.5	~ 3
Bits/cell	~ 10	~ 7	-	~ 10	~ 14	~ 6
Energy Efficiency VMM (TMAC/s)/W)	3.7	1.3	14.7		5	-

(a)

Memristor/ RRAM Crossbar Implementations / PCM / MRAM

	Nat Commun '17 [185]	SoVC '19 [183]	IEDM '18 [186]	Nat Electron '19 [184]	Nature '20 [141]	Nat Electron '20 [187]	Nat Electron '23 [149]	Nature '22 [189]
Process (nm)	1200	130	-	2000	130	180	14	28
Technology	RRAM (TiN/$HfAl_yO_x$/ TaO_x/TiN)	RRAM (HfO_2/Ti/TiN)	RRAM (TiN/TEL/ HfO_x/TiN)	RRAM (Pd/HfO_2/Ta)	RRAM (TiN/TaO_x/ HfO_x/TiN)	RRAM(WO_x)	PCM	MRAM
Array size	128×8	128×128	1Kb	128×64	$8 \times 128 \times 16$	54×108	$64 \times 256 \times 256$	64×64
Cell structure	1T1R	1T1R	1T1R	1T1R	1T1R	0T1R	8T4R	1T1R
Programming mechanism	SET: V_{WL} = 2.3V, V_{BL}=2.1V@50ns, V_{SL}=0V RESET: V_{WL} = 8V, V_{BL}=0V, V_{SL}=2V@50ns	Forming:4V@40μs SET:2V@100ns RESET:2.5V@100ns	SET & RESET	SET & RESET	SET: V_{WL} = 1.8V, V_{BL}=2.0V@50ns, V_{SL}=0V RESET: V_{WL} = 4.7V, V_{BL}=0V, V_{SL}=1.8V@50ns	SET: 1.8V & RESET: -1.8V	SET: 125μA@250ns RESET: 700μA@125ns	V_{WRITE}=1.5V@200ns
VMM Input encoding	voltage pulse at top electrode	voltage pulse at WL (transistor gate)	voltage pulse at top electrode	voltage	pulse number	pulse width	pulse width	voltage pulse at transistor gate
Cell Area (μm^2)	-	7.32	-	16	0.25	-	4.08	0.933
Resistance range (kΩ)	25-150	1.6-4000	40-1000	1.1-10	50-500	12-61	100-10000	13-26
Bits/cell	5% error programming	1	~ 3.2	4-5	5	-	3-4	1
On-chip learning	No	No	Yes	No	Partial	Yes	No	No
Energy Efficiency VMM (TMAC/s)/W)	-	0.5	-	12.5	11	0.187	9.76	405

(b)

Figure 19. Summary of recent FG and RRAM/memristor crossbar implementations. (a) Summary of recent FG VMM and related crossbar implementations. (b) Summary of recent RRAM VMM implementations.

voltage or currents are directly possible for FG systems, while resistive solutions typically rely on pulse width modulation. Among the newer technology options, PCM devices [149] have demonstrated the largest resistance values (compared with Memristor/RRAM devices) resulting in smaller power dissipation values and ability to turn on many rows simultaneously. MRAM systems potentially can have high energy efficiency for smaller systems with low-resolution (e.g. 1–2 bit) weights. Investigating RRAM devices with higher range of resistances is an important direction of future RRAM research that can enable lower power-dissipation (e.g. suitable for systems operating on harvested energy) as well as reduced issues with voltage drop along bitlines.

7.2. Peripheral circuits

The analog architecture [10] around the VMM and other computations for a NVM CIM system typically requires some additional circuits both in operation and for programming these systems. Programming circuitry can often use a few circuits to multiplex to the entire array or have some parallel operation for faster programming [165, 166]. During typical operation, such as for an inference system, the programming infrastructure is typically disabled; the analog architecture around the inference system must still be considered. Programing infrastructure is required even for continuous learning systems (e.g. [65, 74]) to set the initial conditions and to set the required precision components (there is always something) for the operation.

Good analog architecture designs minimize the total number of data converters (analog to digital (ADC), digital to analog (DAC)) as well as signal communication, including memory access. Some architectures do not require any data converters particularly if the inputs or outputs go into, or transmit out towards, sensors or transducers, respectively. Analog processing (and to some extent digital processing) should operate at the speed of the incoming data to minimize intermediate data storage [10]. Some NN implementations are built using significant intermediate digital memory (& data conversion) and intermediate digital processing (e.g. data staging for deep convolutional networks, digital nonlinear operations for long short term memory networks), incurring a heavy cost for these analog architecture choices. On the other hand, some architectures use an analog VMM co-processor for a digital system, often incurring a large cost due to the periphery input DACs and output ADCs (figure 20) for that operation, Some intermediate representations, such as analog pulse widths for digital systems (e.g. [148]) can provide some efficiencies.

Journal	Process (nm)	ADC resolution (bit/ENOB)	ADC sample rate/ Effective sample rate (fs, MHz)	Mx(# of column per ADC)	Power (W)	FOM (pJ)	Area(m^2)	Area proportion (ADC/chip)
Low Speed ADC								
Science'23 [190]	130	8 / 8	0.78 / 0.78	1	11.9	0.06	8597080	0.394
Nature Elect'23 [149] (CCO-based)	16	12 / 8.7	7.9 / 7.9	1	_	_	400	0.27
Med Speed ADC								
Nature Elect'19 [191]	180	9 / 9	16.4 / 16.4	1	_	_	_	_
Nature'20 [141]	130	8 / 8	2.5 / 0.625	4	51	0.01	48000	0.75
Nature Elect'22 [192]	55	8 / 8	1.024 / 0.016	64	33.18	0.129	_	_
Flash ADC								
TCASI'22 [193]	40	4 / 4	25 / 3.125	8	1.4	0.0035	2300	0.072
TED '20 [194]	90	3 / 3	150 / 18.75	8	_	_	_	_
SSCL'20 [195]	90	1 / 1	140 / 17.5	8	_	_	_	_
JSSC'22 [196]	40	3 / 3	100 / 12.5	8	_	_	11500	0.046

Figure 20. Summary of ADC details in recent CIM work. Mx denotes the number of elements multiplexed into the data converter, resulting in a lower effective sample rate.

As the ADCs can take a major percentage of energy and area of the system, particularly a co-processor type system, the tradeoffs (figure 20) in ADCs for CiM systems are a significant part of the system design. In a coprocessor system, one needs as many DACs and ADCs as there are input and output data lines, requiring the DACs and ADCs to be pitch matched to the rows and columns of the crossbar network, creating a difficult constraint on the comparators and architecture of the data converters. Ramp ADC architectures parallelize along a row or column [165], as it typical for parallel data conversion on commercial CMOS imagers. Ramp ADCs can create analog pulse widths [148, 197] which can be transmitted to the later stages to replace the DACs at the input of the next stage; this results in significant savings in energy and area. To enable a good matching of the ADC characteristics with the MAC output, replica biasing methods should be used for generating ADC references (e.g. [196]). More complicated ADC architectures (e.g. Flash) that require exponentially more number of comparators require a high degree of multiplexing that must be considered when comparing these architectures.

Spiking networks used as coprocessors can simplify this approach as some neurons make a 1 bit encoding of the analog signals, effectively 1-bit sigma-delta ADCs that are downfiltered by the resulting synapse circuitry. Further, careful choice over the neuron model (e.g. Hodgkin-Huxley Circuit models [32]) result in far more efficient neural codes. Efficient spike codes and event address representations for these sparse codes makes for an efficient and cascade-able system block (e.g. [74]) Further research into peripheral circuits, particularly ADCs are an important direction of future research.

7.3. NVM conclusion and way forward

FG devices continue to outperform memristive devices as predicted over 10 years ago [15], even with far more efforts and funding focused towards memristive/RRAM approaches. Nothing fundamentally limits FG devices on standard CMOS processes as all processes have an insulator thickness that has sufficient memory retention. FG approaches have been demonstrated across two orders of CMOS IC fabrication processes and one expects it to be possible on any future CMOS process, and given its great potential, should be in the conversations for future IC fabrication development.

Data availability statement

All data that support the findings of this study are included within the article (and any supplementary files).

Acknowledgments

Jennifer Hasler would like to acknowledge the multiple discussions from her GT research group as well as discussions arising in her Neuromorphic Analog VLSI course in Spring 2020, particularly the in depth discussions with Siri Narla. Jennifer Hasler also appreciate discussions with Mika Laiho in the areas around learning and computation for neural structures between FG and memristor technologies.

Arindam Basu was supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 11200922] and a Start-up grant from City University of Hong Kong.

ORCID iD

Jennifer Hasler https://orcid.org/0000-0002-6866-3156

References

[1] LeCun Y, Bengio Y and Hinton G 2015 Deep learning *Nature* **521** 436–44
[2] Taigman Y, Yang M, Ranzato M and Wolf L 2014 DeepFace: closing the gap to human-level performance in face verification *IEEE CVPR* pp 1701–8
[3] Deng L *et al* 2013 Recent advances in deep learning for speech research at Microsoft *IEEE ICASSP* pp 8604–8
[4] Silver D *et al* 2016 Mastering the game of Go with deep neural networks and tree search *Nature* **529** 484–9
[5] Jouppi N P *et al* 2017 In-datacenter performance analysis of a tensor processing unit *Proc. 44th Annual Int. Symp. on Computer Architecture (2017)* (ISCA) pp 1–12
[6] Chen Y-H, Krishna T, Emer J and Sze V 2016 Eyeriss: an energy-efficient reconfigurable accelerator for deep convolutional neural networks *IEEE Int. Solid-State Circuits Conf. (ISSCC 2016) (Digest of Technical Papers)* pp 262–3
[7] Moons B *et al* 2017 Envision: a 0.26-to-10TOPS/W subword-parallel dynamic-voltage-accuracy-frequency-scalable convolutional neural network processor in 28nm FDSOI *IEEE Int. Solid-State Circuits Conf. (ISSCC 2017) (Digest of Technical Papers)* pp 246–7
[8] Kucic M, Hasler P, Dugger J and Anderson D 2001 Programmable and adaptive analog filters using arrays of floating-gate circuits *Advanced Research in VLSI* pp 148–62
[9] Hasler P and Anderson D V 2002 Cooperative analog-digital signal processing *IEEE ICASSP (Orlando, FL)* pp IV-3972–75
[10] Hasler J 2019 Analog architecture and complexity theory to empowering ultra-low power configurable analog and mixed mode SoC systems *J. Low Power Electron. Appl.* **9** 1–37
[11] Indiveri G and Liu S-C 2015 Memory and information processing in neuromorphic systems *Proc. IEEE* **103** 1379–97
[12] Zidan M A, Strachan J P and Lu W 2018 The future of electronics based on memristive systems *Nat. Electron.* **1** 22–29
[13] Mead C 2020 How we created neuromorphic engineering *Nat. Electron.* **3** 434–5
[14] Bose S K, Acharya J and Basu A 2019 Is my neural network neuromorphic? Taxonomy, recent trends and future directions in neuromorphic engineering *2019 53rd Asilomar Conf. on Signals, Systems and Computers* pp 1522–7
[15] Hasler J and Marr H B 2013 Finding a roadmap to achieve large neuromorphic hardware systems *Front. Neurosci.* **7** 1–29
[16] Merrikh-Bayat F, Guo X, Klachko M, Prezioso M, Likharev K K and Strukov D B 2017 High-performance mixed-signal neurocomputing with nanoscale floating-gate memory cell arrays *IEEE Trans. Neural Netw. Learn. Syst.* **29** 4782–90
[17] Hasler J 2020 Large-scale field programmable analog arrays *IEEE Proc.* **108** 1283–302
[18] Srinivasan V, Graham D and Hasler P 2005 Floating-gates transistors for precision analog circuit design: an overview *Midwest Symp. on Circuits and Systems* vol 1 pp 71–74
[19] Srinivasan V, Serrano G, Twigg C and Hasler P 2008 A floating-gate-based programmable CMOS reference *IEEE Trans. Circuits Syst.* I **55** 3448–56
[20] Srinivasan V, Serrano G J, Gray J and Hasler P 2007 A precision CMOS amplifier using floating-gate transistors for offset cancellation *IEEE J. Solid-State Circuits* **42** 280–91
[21] Peng S Y, Qureshi M S, Hasler P E, Basu A and Degertekin F L 2008 A charge-based low-power high-SNR capacitive sensing interface circuit *IEEE Trans. Circuits Syst.* I **55** 1863
[22] Graham D W, Hasler P E, Chawla R and Smith P D 2007 A low-power, programmable bandpass filter section for higher-order filter applications *IEEE Trans. Circuits Syst.* I **54** 1165–76
[23] Chawla R, Adil F, Serrano G and Hasler P E 2007 Programmable $G_m - C$ filters using floating-gate operational transconductance amplifiers *IEEE Trans. Circuits Syst.* I **54** 481–91
[24] Serrano G, Kucic M and Hasler P 2002 Investigating programmable floating-gate digital-to-analog converter as single element or element arrays *IEEE Midwest CAS* vol 1 pp 75–77
[25] Brady P and Hasler P 2005 Offset compensation in flash ADCs using floating-gate circuits *IEEE ISCAS* vol 6 pp 6154–7
[26] Pereira A W, Allen D J and Hasler P E 2004 A 0.5μm CMOS programmable discrete-time Delta-Sigma modulator with floating gate elements *ISCAS* vol 1 pp 213–6
[27] Hasler P 2005 Low-power programmable signal processing *Int. Workshop on System-on-Chip for Real-Time Applications* pp 413–8
[28] Hasler P, Smith P D, Graham D, Ellis R and Anderson D V 2005 Analog floating-gate, on-chip auditory sensing system interfaces *IEEE Trans. Sens.* **5** 1027–34
[29] Bandyopadhyay A, Hasler P and Anderson D V 2005 A CMOS floating-gate matrix transform imager *IEEE Trans. Sens.* **5** 455–62
[30] Bandyopadhyay A, Lee J, Robucci R and Hasler P 2006 MATIA: a programmable 80 W/frame CMOS block matrix transform imager architecture *IEEE J. Solid-State Circuits* **41** 663–72
[31] Hasler J and Wang H 2015 A fine-grain FPAA fabric for RF + baseband *GOMAC*
[32] Farquhar E and Hasler P 2005 A bio-physically inspired silicon neuron *IEEE Trans. Circuits Syst.* I **52** 477–88
[33] George S, Hasler J, Koziol S, Nease S and Ramakrishnan S 2013 Low-power dendritic computation for wordspotting *J. Low Power Electron. Appl.* **3** 78–98
[34] Kahng D and Sze S M 1967 A floating-gate and its application to memory devices *Bell Syst. Tech. J.* **46** 1288–95
[35] Harari E 2012 Flash memory—the great disruptor! *ISSCC* pp 10–15
[36] Gilder G 2005 *The Silicon Eye* (Norton)
[37] Lenzlinger M and Snow E H 1969 Fowler-Nordheim tunneling into thermally grown SiO_2 *J. Appl. Phys.* **40** 278–83
[38] Masuoka F, Momodomi M, Iwata Y and Shirota R 1984 New ultra high density EPROM and flash EEPROM with NAND structure cell *IEEE IEDM* pp 552–5
[39] Masuoka F, Asano M, Iwahashi H, Komuro T and Tanaka S 1984 A new flash E^2PROM cell using triple polysilicon technology *IEEE IEDM* pp 464–7
[40] Wang P, Xu F, Wang B, Gao B, Wu H, Qian H and Yu S 2019 Three-dimensional NAND flash for vector–matrix multiplication *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **27** 988–91
[41] Marotta G G *et al* 2010 A 3bit/cell 32Gb NAND flash memory at 34nm with 6MB/s program throughput and with dynamic 2b/cell blocks configuration mode for a program throughput increase up to 13MB/s *Int. Solid-State Circuits Conf.*
[42] Li Y, Lee S, Fong Y, Pan F, Kuo T C, Park J, Samaddar T, Nguyen H, Mui M and Htoo K 2008 A 16Gb 3b/Cell NAND flash memory in 56nm with 8MB/s write rate *Int. Solid-State Circuits Conf.*
[43] Shibata N *et al* 2012 A 19nm 112.8mm^2 64Gb multi-level flash memory with 400Mb/s/pin 1.8V toggle mode interface *ISSCC* pp 422–3
[44] Li Y *et al* 2012 128Gb 3b/Cell NAND flash memory in 19nm technology with 18MB/s write rate and 400Mb/s toggle mode *ISSCC* pp 436–7
[45] Lee D *et al* 2012 A 64Gb 533Mb/s DDR interface MLC NAND flash in sub-20nm technology *ISSCC* pp 430–1

[46] Widrow B 1960 An adaptive ADALINE neuron using chemical memistors *Technical Report* No. 1553-2 (Stanford University)
[47] Kung S Y 1987 *VLSI Array Processors* (Prentice-Hall)
[48] JaJa J 1992 *An Introduction to Parallel Algorithms* (Addison-Wesley)
[49] Hasler P and Akers L 1991 Implementation of analog neural networks *Annual Int. Conf. on Computers and Communications* pp 32–38
[50] Cauwenberghs G and Yariv A 1994 Fault-tolerant dynamic multi-level storage in analog VLSI *IEEE Trans. Circuits Syst.* II **41** 827–9
[51] Cauwenberghs G 1995 A micropower CMOS algorithmic A/D/A converter *IEEE Trans. Circuits Syst.* I **42** 913–9
[52] Hasler P and Akers L 1991 A continuous time synapse employing a refreshable multilevel memory *Int. Joint Conf. on Neural Networks* pp 563–8
[53] Hasler P, Diorio C, Minch B A and Mead C A 1994 Single transistor learning synapses *Advances in Neural Information Processing Systems* vol 7, ed G Tesauro, D S Touretzky and T K Leen (MIT Press) pp 817–24
[54] Hasler P, Diorio C, Minch B and Mead C 1995 Single transistor learning synapse with long term storage *IEEE Int. Symp. on Circuits and Systems* vol 3 pp 1660–3
[55] Minch B and Hasler P 1999 A floating-gate technology for digital CMOS processes *IEEE ISCAS* vol 2 pp 400–3
[56] Li X *et al* 2017 Enabling energy-efficient nonvolatile computing with negative capacitance FET *IEEE Trans. Electron Devices* **64** 3452–8
[57] Hasler P, Minch B and Diorio C 1999 Adaptive circuits using pFET floating-gate devices *Advanced Research in VLSI* pp 215–29
[58] Kim S, Hasler J and George S 2016 Integrated floating-gate programming environment for system-level ICs *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **24** 2244–52
[59] Rumberg B, Graham D W and Navidi M M 2017 A regulated charge pump for tunneling floating-gate transistors *IEEE Trans. Circuits Syst.* I **64** 516–27
[60] Holler M, Tam S, Castro H and Benson R 1989 An electrically trainable artificial neural network (ETANN) with 10240 'floating gate' synapses *Proc. Int. Joint Conf. on Neural Networks (Washington, DC)* vol II pp 191–6
[61] Thomsen A and Brooke M A 1991 A floating-gate MOSFET with tunneling injector fabricated using a standard double-polysilicon CMOS process *IEEE Electron Device Lett.* **12** 111–3
[62] Mead C 1990 Neuromorphic electronic systems *Proc. IEEE* **78** 1629–36
[63] Marr B, Degnan B, Hasler P and Anderson D 2013 Scaling energy per operation via an asynchronous pipeline *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **21** 147–51
[64] Chawla R, Bandyopadhyay A, Srinivasan V and Hasler P 2004 A 531 nW/MHz, 128 × 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity *CICC* p 651
[65] Hasler P and Dugger J 2005 An analog floating-gate node for supervised learning *IEEE Trans. Circuits Syst.* I **52** 834–45
[66] Peng S-Y, Hasler P and Anderson D 2007 An analog programmable multi-dimensional radial basis function based classifer *VLSI—SoC IFIP Int. Conf. on Very Large Scale Integration* pp 13–18
[67] Lu J *et al* 2014 A 1TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13μm CMOS *IEEE Int. Solid-State Circuits Conf.* pp 504–5
[68] Peng S-Y, Minch B A and Hasler P 2008 Analog VLSI implementation of support vector machine learning and classification *ISCAS*
[69] Ramakrishnan S and Hasler J 2014 Vector-matrix multiply and winner-take-all as an analog classifier *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **22** 353–61
[70] Gordon C, Farquhar E and Hasler P 2004 A family of floating-gate adapting synapses based upon transistor channel models *Int. Symp. on Circuits and Systems* vol 1 pp 317–20
[71] Ramakrishnan S, Hasler P and Gordon C 2011 Floating gate synapses with spike-time-dependent plasticity *IEEE Trans. Biomed. Circuits Syst.* **5** 244–52
[72] Gopalakrishnan R and Basu A 2015 On the non-STDP behavior and its remedy in a floating-gate synapse *IEEE Trans. Neural Netw. Learn. Syst.* **26** 2596–601
[73] Gopalakrishnan R and Basu A 2015 Triplet spike time dependent plasticity in a floating-gate synapse *IEEE Trans. Neural Netw. Learn. Syst.* **28** 778–90
[74] Brink S, Nease S, Hasler P, Ramakrishnan S, Wunderlich R, Basu A and Degnan B 2013 Learning-enabled neuron array IC based upon transistor channel models of biological phenomena *IEEE Trans. Biomed. Circuits Syst.* **7** 71–81
[75] SyNAPSE DARPA Program 2008 Systems of neuromorphic adaptive plastic scalable electronics *DSO, DARPA-BAA* 80-28 (available at: https://aeromotores.wordpress.com/wp-content/uploads/2011/03/darpa-systems-of-neuromorphic-adaptive-plastic-scalable-electronics.pdf) (Accessed 24 May 2020)
[76] Neovision DARPA Program (available at: www.federalgrants.com/Neovision2-18664.html)
[77] CT2WS DARPA Program (available at: https://en.wikipedia.org/wiki/Cognitive_Technology_Threat_Warning_System)
[78] Swaroop B, West W C, Martinez G, Kozicki M N and Akers L A 1998 Programmable current mode Hebbian learning neural network using programmable metallization cell *1998 IEEE ISCAS (Monterey)* vol 3 pp 33–36
[79] Kozicki M N, Yun M, Hilt L and Singh A 1999 Applications of programmable resistance changes in metal-doped chalcogenides *Electrochemical Society* vol 99 pp 298–309
[80] Kozicki M N and West W C 1998 Programmable metallization cell structure and method of making same *US Patent* 5761115
[81] Kozicki M N, Park M and Mitkova M 2005 Nanoscale memory elements based on solid-state electrolytes *IEEE Trans. Nanotechnol.* **4** 331–8
[82] Strukov D B, Snider G S, Stewart D R and Williams R S 2008 The missing memristor found *Nature* **453** 80–83
[83] Widrow B and Hoff M E 1960 Adaptive switching circuits *IRE Wescon Convention Record* pp 96–104
[84] Chua L 1971 Memristor-the missing circuit element *IEEE Trans. Circuit Theory* **18** 507–19
[85] Chua L O and Kang S M 1976 Memristive devices and systems *Proc. IEEE* **64** 209–23
[86] Chua L O 2011 Resistance switching memories are memristors *Appl. Phys.* A **102** 765–83
[87] Kim K-H, Gaba S, Wheeler D, Cruz-Albrecht J M, Hussain T, Srinivasa N and Lu W 2012 A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications *Nano Lett.* **12** 389–95
[88] Jo S H, Kim K-H and Lu W 2009 Programmable resistance switching in nanoscale two-terminal devices *Nano Lett.* **9** 496–500
[89] Jo S H, Chang T, Ebong I, Bhadviya B B, Mazumder P and Lu W 2010 Nanoscale memristor device as synapse in neuromorphic systems *Nano Lett.* **10** 1297–301
[90] Jo S H, Kim K-H and Lu W 2011 Short-term memory to long-term memory transition in a nanoscale memristor *ACS Nano* **5** 7669–76

[91] Sundqvist K, Ferry D K and Kish L B 2017 Memristor equations: incomplete physics and undefined passivity/activity *Fluct. Noise Lett.* **16** 1771001
[92] Chien W-C *et al* 2011 Multi-level 40nm WO_X resistive memory with excellent reliability *IEEE IEDM*
[93] Jang J-W, Park S, Burr G W, Hwang H and Jeong Y-H 2015 Optimization of conductance change in Pr_{1-x} Ca_x MnO_3-based synaptic devices for neuromorphic systems *IEEE Electron Device Lett.* **36** 457–9
[94] Yu S, Chen P, Cao Y, Xia L, Wang Y and Wu H 2015 Scaling-up resistive synaptic arrays for neuro-inspired architecture: challenges and prospect *IEEE IEDM (Washington, DC)* pp 17.3.1–4
[95] Agarwal S *et al* 2016 Resistive memory device requirements for a neural algorithm accelerator *IJCNN (Vancouver, BC)* pp 929–38
[96] Xu C, Niu D, Muralimanohar N, Jouppi N P and Xie Y 2013 Understanding the trade-offs in multi-level cell ReRAM memory design *IEEE DAC (Austin, TX)* pp 1–6
[97] Kadetotad D *et al* 2014 Neurophysics-inspired parallel architecture with resistive crosspoint array for dictionary learning *IEEE BioCAS* pp 536–9
[98] Chen P *et al* 2015 Mitigating effects of non-ideal synaptic device characteristics for on-chip learning *IEEE ICCAD (Austin, TX)* pp 194–9
[99] Chen P *et al* 2015 Technology-design co-optimization of resistive cross-point array for accelerating learning algorithms on chip *IEEE DATE (Grenoble)* pp 854–9
[100] Chang C *et al* 2018 Mitigating asymmetric nonlinear weight update effects in hardware neural network based on analog resistive synapse *IEEE J. Emerg. Sel. Top. Circuits Syst.* **8** 116–24
[101] Yi W, Savel'ev S E, Medeiros-Ribeiro G, Miao F, Zhang M-X, Yang J J, Bratkovsky A M and Williams R S 2016 Quantized conductance coincides with state instability and excess noise in tantalum oxide memristors *Nat. Commun.* **7** 11142
[102] Park S *et al* 2012 RRAM-based synapse for neuromorphic system with pattern recognition function *IEEE IEDM*
[103] Fuller E J *et al* 2017 Li-ion synaptic transistor for low power analog computing *Adv. Mater.* **29** 1604310
[104] van De Burgt Y, Lubberman E, Fuller E J, Keene S T, Faria G C, Agarwal S, Marinella M J, Talin A A and Salleo A 2017 A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing *Nat. Mater.* **16** 414–8
[105] Li Y *et al* 2020 Filament-free bulk resistive memory enables deterministic analogue switching *Adv. Mater.* **32** 2003984
[106] Whittingham M S 2021 Solid-state ionics: the key to the discovery and domination of lithium batteries: some learnings from β-alumina and titanium disulfide *MRS Bull.* **46** 168–73
[107] Fuller E J *et al* 2019 Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing *Science* **364** 570–4
[108] Melianas A *et al* 2020 Temperature-resilient solid-state organic artificial synapses for neuromorphic computing *Sci. Adv.* **6** eabb2958
[109] Melianas A, Kang M-A, Mohammadi A V, Quill T J, Tian W, Gogotsi Y, Salleo A and Hamedi M M 2022 High-speed ionic synaptic memory based on 2D titanium carbide MXene *Adv. Funct. Mater.* **32** 2109970
[110] Tang J *et al* 2018 ECRAM as scalable synaptic cell for high-speed, low-power neuromorphic computing *IEEE IEDM* (IEEE) pp 13–11
[111] Kim S *et al* 2019 Metal-oxide based, CMOS-compatible ECRAM for deep learning accelerator *IEEE IEDM* pp 35–37
[112] Onen M, Emond N, Li J, Yildiz B and Del Alamo J A 2021 CMOS-compatible protonic programmable resistor based on phosphosilicate glass electrolyte for analog deep learning *Nano Lett.* **21** 6111–6
[113] Kwak H, Lee C, Lee C, Noh K and Kim S 2021 Experimental measurement of ungated channel region conductance in a multi-terminal, metal oxide-based ECRAM *Semicond. Sci. Technol.* **36** 114002
[114] Sebastian A, Le Gallo M, Khaddam-Aljameh R and Eleftheriou E 2020 Memory devices and applications for in-memory computing *Nat. Nanotechnol.* **15** 529–44
[115] Basu A, Acharya J, Karnik T, Liu H, Li H, Seo J-S and Song C 2018 Low-power, adaptive neuromorphic systems: recent progress and future directions *IEEE J. Emerg. Sel. Top. Circuits Syst.* **8** 6–27
[116] Jeong B H *et al* 2011 A 58nm 1.8V 1Gb PRAM with 6.4MB/s program BW *ISSCC* pp 500–2
[117] Choi Y *et al* 2012 A 20nm 1.8V 8Gb PRAM with 40MB/s Program Bandwidth *ISSCC*
[118] Burr G W *et al* 2015 Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element *IEEE Trans. Electron Devices* **62** 3498–507
[119] John R A *et al* 2018 Ionotronic halide perovskite drift-diffusive synapses for low-power neuromorphic computation *Adv. Mater.* **30** 1805454
[120] John R A *et al* 2018 Ultralow power dual-gated subthreshold oxide neuristors: an enabler for higher order neuronal temporal correlations *ACS Nano* **12** 11263–73
[121] John R A *et al* 2020 Optogenetics inspired transition metal dichalcogenide neuristors for in-memory deep recurrent neural networks *Nat. Commun.* **11** 3211
[122] Hasler J, Kim S and Adil F 2016 Scaling floating-gate devices predicting behavior for programmable and configurable circuits and systems *J. Low Power Electron. Appl.* **6** 13
[123] Mead C 1994 Scaling of MOS technology to sub micrometer feature sizes *J. VLSI Signal Process.* **8** 9–25
[124] Nicollian E H and Brews J R 1982 *MOS Physics and Technology* (Wiley)
[125] Carley L R 1989 Trimming analog circuits using floating-gate analog MOS memory *IEEE J. Solid-State Circuits* **24** 1569–75
[126] Sackinger E and Guggenbuhl W 1988 An analog trimming circuit based on a floating-gate device *IEEE J. Solid-State Circuits* **23** 1437–40
[127] Bleiker C and Melchior H 1987 A four-state EEPROM using floating-gate memory cells *IEEE J. Solid-State Circuits* **22** 460–3
[128] Nozawa H and Kohyama S 1992 A thermionic electron emission model for charge retention in SAMOS structures *Jpn. J. Appl. Phys.* **21** 111–2
[129] Puthenkovilakam R and Chang J P 2004 An accurate determination of barrier heights at the HfO_2/Si interfaces *J. Appl. Phys.* **96** 2701
[130] Brink S, Hasler J and Wunderlich R 2014 Adaptive floating-gate circuit enabled large-scale FPAA *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **22** 2307–15
[131] Twigg C, Gray J and Hasler P 2007 Programmable floating-gate FPAA switches are not dead weight *Int. Symp. on Circuits and Systems* pp 169–72
[132] Schlottmann C R and Hasler P E 2011 A highly dense, low power, programmable analog vector-matrix multiplier: the FPAA implementation *IEEE J. Emerg. Sel. Top. Circuits Syst.* **1** 403–11
[133] Basu A *et al* 2010 A floating-gate-based field-programmable analog arrays *IEEE J. Solid-State Circuits* **45** 1781–94

[134] Danial L, Pikhay E, Herbelin E, Wainstein N, Gupta V, Wald N, Roizin Y, Daniel R and Kvatinsky S 2019 Two-terminal floating-gate transistors with a low-power memristive operation mode for analogue neuromorphic computing *Nat. Electron.* **2** 596–605
[135] Graf H P and deVegvar P G N 1987 A CMOS implementation of a neural network model *ARVLSI* (Stanford MIT Press) pp 351–67
[136] Sheridan P M, Cai F, Du C, Ma W, Zhang Z and Lu W D 2017 Sparse coding with memristor networks *Nat. Nanotechnol.* **12** 784–90
[137] Hasler J, Ayyappan P R, Ige A and Mathew P 2024 A 130nm CMOS programmable analog standard cell library *IEEE Trans. Circuits Syst.* I **71** 2497–510
[138] Mathews P O, Ayyappan P R, Ige A, Bhattacharyya S, Yang L and Hasler J 2024 A 65 nm and 130 nm CMOS programmable analog standard cell library for scalable system synthesis *IEEE CICC*
[139] Chen W-H *et al* 2019 CMOS-integrated memristive non-volatile computing-in-memory for AI edge processors *Nat. Electron.* **2** 420–8
[140] Wang Z *et al* 2019 *In situ* training of feed-forward and recurrent convolutional memristor networks *Nat. Mach. Intell.* **1** 434–42
[141] Yao P, Wu H, Gao B, Tang J, Zhang Q, Zhang W, Yang J J and Qian H 2020 Fully hardware-implemented memristor convolutional neural network *Nature* **577** 641–6
[142] Merikh-Bayat F *et al* 2018 Implementation of multilayer perceptron network with highly uniform passive memristive crossbar circuits *Nat. Commun.* **9** 2331
[143] Hasler J and Hao C 2024 Programmable analog system benchmarks leading to efficient analog computation synthesis *ACM Trans. Reconfigurable Technol. Syst.* **17** 1–25
[144] George S, Kim S, Shah S, Hasler J, Collins M, Adil F, Wunderlich R, Nease S and Ramakrishnan S 2016 A programmable and configurable mixed-mode FPAA SoC *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **24** 2253–61
[145] Ige A, Yang L, Yang H, Hasler J and Hao C 2023 Analog system high-level synthesis for energy-efficient reconfigurable computing *J. Low Power Electron. Appl.* **13** 58
[146] Hasler J 2024 Scalable analog standard cells for mixed-signal processing and computing *GOMAC*
[147] Laiho M, Hasler J, Zhou J, Du C, Lu W, Lehtonen E and Poikonen J H 2015 FPAA/memristor hybrid computing infrastructure *IEEE Trans. Circuits Syst.* I **62** 906–15
[148] Ambrogio S *et al* 2023 An analog-AI chip for energy-efficient speech recognition and transcription *Nature* **620** 768–75
[149] Le Gallo M *et al* 2023 A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference *Nat. Electron.* **6** 680–93
[150] Jhang C J *et al* 2021 Challenges and trends of SRAM-based computing-in-memory for AI edge devices *IEEE Trans. Circuits Syst.* I **68** 1773–86
[151] Bose S K, Singla D and Basu A 2021 A 51.3-TOPS/W, 134.4-GOPS in-memory binary image filtering in 65-nm CMOS *IEEE J. Solid-State Circuits* **57** 323–35
[152] Zhang X and Basu A 2022 A 915-1220 TOPS/W, 976-1301 GOPS hybrid in-memory computing based always-on image processing for neuromorphic vision sensors *IEEE J. Solid-State Circuits* **58** 589–99
[153] Kim K, Gao C, Graca R, Kiselev I, Yoo H-J, Delbruck T and Liu S-C 2022 A 23-μW keyword spotting IC with ring-oscillator-based time-domain feature extraction *IEEE J. Solid-State Circuits* **57** 3298–311
[154] Verma N, Shoeb A, Bohorquez J, Dawson J, Guttag J and Chandrakasan A P 2010 A micro-power EEG acquisition SoC with integrated feature extraction processor for a chronic seizure detection system *IEEE J. Solid-State Circuits* **45** 804–16
[155] Sukumaran D *et al* 2012 A low-power, reconfigurable smart sensor system for EEG acquisition and classification *2012 IEEE Asia Pacific Conf. on Circuits and Systems (APCCAS)* pp 9–12
[156] Lin L, Jain S and Alioto M 2018 A 595pW 14pJ/cycle microcontroller with dual-mode standard cells and self-startup for battery-indifferent distributed sensing *IEEE Intl. Solid-State Circuits Conf. (ISSCC)* pp 44–46
[157] Cauwenberghs G, Neugebauer C F and Yariv A 1991 An adaptive CMOS matrix-vector multiplier for large scale analog hardware neural network applications *IJCNN (Seattle)* vol 1 pp 507–11
[158] Pi S, Li C, Jiang H, Xia W, Xin H, Yang J J and Xia Q 2019 Memristor crossbar arrays with 6-nm half-pitch and 2-nm critical dimension *Nat. Nanotechnol.* **14** 35–39
[159] Hasler P, Minch B and Diorio C 1999 Adaptive circuits using pFET floating-gate devices *Proc. IEEE 20th Advanced Research in VLSI (Atlanta)* pp 215–29
[160] Hasler P, Andreou A, Diorio C, Minch B A and Mead C 1998 Impact ionization and hot-electron injection derived consistently from Boltzman transport *VLSI Des.* **8** 455–61
[161] Hasler P, Basu A and Kozoil S 2007 Above threshold pFET injection modeling intended for programming floating-gate systems *IEEE Int. Symp. on Circuits and Systems* pp 1557–60
[162] Duffy C and Hasler P 2004 Scaling pFET hot-electron injection *Int. Workshop on Computational Electronics* vol 2004 pp 149–50
[163] Park D and Lee J 2011 Floating-gate coupling canceller for multi-level cell NAND flash *IEEE Trans. Magn.* **47** 624–8
[164] Hasler P, Minch B and Diorio C 2001 An autozeroing floating-gate amplifier *IEEE Trans. Circuits Syst.* II **48** 74–82
[165] Kucic M R 2004 Analog computing arrays *PhD Thesis* Georgia Institute of Technology
[166] Hasler J 2022 Special session: testing and characterization for large-scale programmable analog systems *IEEE VLSI Test Symp. (San Diego, USA)* pp 1–5
[167] Hooper M, Kucic M and Hasler P 2004 Characterization of charge-pump rectifiers for standard submicron CMOS processes *IEEE ISCAS (Vancouver, Canada)*
[168] Hooper M, Kucic M and Hasler P 2005 Integration of high voltage charge-pumps in a submicron standard CMOS process for programming analog floating-gate circuits *ISCAS (Kobe, Japan)* pp 125–8
[169] Harrison R, Bragg J, Hasler P, Minch B and Deweerth S 2001 A CMOS programmable analog memory-cell array using floating-gate circuits *IEEE Trans. Circuits Syst.* II **48** 4–11
[170] Hasler P, Smith P, Duffy C, Gordon C, Dugger J and Anderson D 2002 A floating-gate vector-quantizer *Midwest Symp. on Circuits and Systems* vol 1 pp 196–9
[171] Ramakrishnan S, Wunderlich R, Hasler J and George S 2013 Neuron array with plastic synapses and programmable dendrites *IEEE Trans. Biomed. Circuits Syst.* **7** 631–42
[172] Hasler J 2017 Starting framework for analog numerical analysis for energy efficient computing *J. Low Power Electron. Appl.* **7** 1–22
[173] Hasler J and Shah S 2018 SoC FPAA hardware implementation of a VMM+WTA embedded learning classifier *IEEE J. Emerg. Sel. Top. Circuits Syst.* **8** 28–37

[174] Hasler J and Shah S 2018 VMM + WTA embedded classifiers learning algorithm implementable on SoC FPAA devices *IEEE J. Emerg. Sel. Top. Circuits Syst.* **8** 65–76
[175] Lazzaro J, Ryckebusch S, Mahowald M A and Mead C A 1989 Winner-take-all networks of O(N) complexity *Advances in Neural Information Processing Systems* vol 1 (Morgan Kaufmann Publishers)
[176] Indiveri G, Horiuchi T, Niebur E and Douglas R 2001 A competitive network of spiking VLSI neurons *Proc. World Congress Neuroinformatics (Vienna, Austria)*
[177] Chicca E 2006 A neuromorphic VLSI system for modeling spike–based cooperative competitive neural networks *PhD Thesis* ETH Zurich, Zurich, Switzerland
[178] Hasler J 2022 The potential of SoC FPAAs for emerging ultra-low-power machine learning *J. Low Power Electron. Appl.* **12** 33
[179] Demler M 2018 *Mythic Multiplies in a Flash: Analog In-Memory Computing Eliminates Dram Read/Write Cycles* (The Linley Group)
[180] Jiang H, Han L, Lin P, Wang Z, Jang M H, Wu Q, Barnell M, Yang J J, Xin H L and Xia Q 2016 Sub-10 nm Ta channel responsible for superior performance of a HfO_2 memristor *Sci. Rep.* **6** 28525
[181] Jiang M, Shan K, He C and Li C 2023 Efficient combinatorial optimization by quantum-inspired parallel annealing in analogue memristor crossbar *Nat. Commun.* **14** 5927
[182] Lee M-J *et al* 2011 A fast, high-endurance and scalable non-volatile memory device made from asymmetric Ta_2O_{5-x}/TaO_{2-x} bilayer structures *Nat. Mater.* **10** 625–30
[183] Zha Y, Nowak E and Li J 2019 Liquid silicon: a nonvolatile fully programmable processing-in-memory processor with monolithically integrated ReRAM for big data/machine learning applications *2019 Symp. on VLSI Circuits* pp C206–7
[184] Wang Z *et al* 2019 Reinforcement learning with analogue memristor arrays *Nat. Electron.* **2** 115–24
[185] Yao P *et al* 2017 Face classification using electronic synapses *Nat. Commun.* **8** 15199
[186] Lin Y *et al* 2018 Demonstration of generative adversarial network by intrinsic random noises of analog RRAM devices *2018 IEEE Int. Electron Devices Meeting (IEDM)* pp 3.4.1–4
[187] Wan W *et al* 2022 A compute-in-memory chip based on resistive random-access memory *Nature* **608** 504–12
[188] Chakrabartty S and Cauwenberghs G 2007 Sub-microwatt analog VLSI trainable pattern classifier *IEEE J. Solid-State Circuits* **42** 1169–79
[189] Cai F *et al* 2019 A fully integrated reprogrammable memristor CMOS system for efficient multiply accumulate operations *Nat. Electron.* **2** 290–9
[190] Jung S *et al* 2022 A crossbar array of magnetoresistive memory devices for in-memory computing *Nature* **601** 211
[191] Zhang W *et al* 2023 Edge learning using a fully integrated neuro-inspired memristor chip *Science* **381** 1205–11
[192] Huo Q *et al* 2022 A computing-in-memory macro based on three-dimensional resistive random-access memory *Nat. Electron.* **5** 469–77
[193] Jing Z *et al* 2022 VSDCA: a voltage sensing differential column architecture based on 1T2R RRAM array for computing-in-memory accelerators *IEEE Trans. Circuits Syst.* I **69** 4028–41
[194] Yin S, Sun X, Yu S and Seo J-S 2020 High-throughput in-memory computing for binary deep neural networks with monolithically integrated RRAM and 90-nm CMOS *IEEE Trans. Electron Devices* **69** 4185–92
[195] He W, Yin S, Kim Y, Sun X, Kim J-J, Yu S and Seo J-S 2020 2-bit-per-cell RRAM-based in-memory computing for area-/energy-efficient deep learning *IEEE Solid-State Circuits Lett.* **3** 194–7
[196] Li W, Sun X, Huang S, Jiang H and Yu S 2022 A 40-nm MLC-RRAM compute-in-memory macro with sparsity control, on-chip write-verify and temperature-independent ADC references *IEEE J. Solid-State Circuits* **57** 2868–77
[197] Jiang H *et al* 2022 A 40nm analog-input ADC-free compute-in-memory RRAM macro with pulse-width modulation between sub-arrays *IEEE Symp. on VLSI Circuits (SOVC)*