1 | Neuromorphic computing architectures based on non-volatile memory

The machine learning-based algorithms that have been discussed thus far for position estimation in Gamma cameras exploit linear algebra, which is involved in virtually every scientific and technical area. Solving matrix equations such as a linear system or an eigenvector equation is accomplished by matrix factorizations or iterative matrix multiplications in conventional digital computers, that can execute advanced operations by a sequence of elementary Boolean functions of 2 or more bits. As a result, solving complex tasks requires a large number of computing steps and extensive use of memory units to store individual bits. To accelerate the execution of such advanced tasks, in-memory computing with analog memories provides a promising avenue, thanks to analog data storage and physical computation in the memory, and has shown high efficiency in terms of time and energy through realizing matrix-vector multiplication in the analog domain with Ohm's law and Kirchhoff's law.

This report therefore contains an introduction to in-memory computing and a review on hardware accelerators for neural networks, which perform computation within a dense memory array in order to overcome major bottlenecks for data-heavy workloads, introducing their figures of merit and possible implementations. The discussion is focused on analog signal accelerators. Lastly, an overview of emerging non volatile memory devices suitable for weight storage is presented, to lay the foundations for a possible comparison between emerging technologies and mature devices.

1.1. Introduction to neuromorphic accelerators

For decades, processor performance has advanced at an inexorable pace by riding on the clever combination of Moore's law, Dennard scaling, and lately by running multiple processor cores in parallel.

In 1965 Gordon Moore, co-founder of Fairchild Semiconductor and Intel, predicted an exponential reduction in the cost per transistor, hence allowing similarly exponential increases in the number of transistors in a dense integrated circuit (IC) [1], that can be observed in Figure 1.1. Despite this observed trend declining from a doubling in the number of components every year to doubling every two years, with an intermediate doubling rate of 18 months [2], Moore's law still allowed an increase in the processing power of microcircuits delivered per square inch, making multi-billion-transistor microprocessors (CPUs) and graphical processing units (GPUs) not just possible, but profitable.

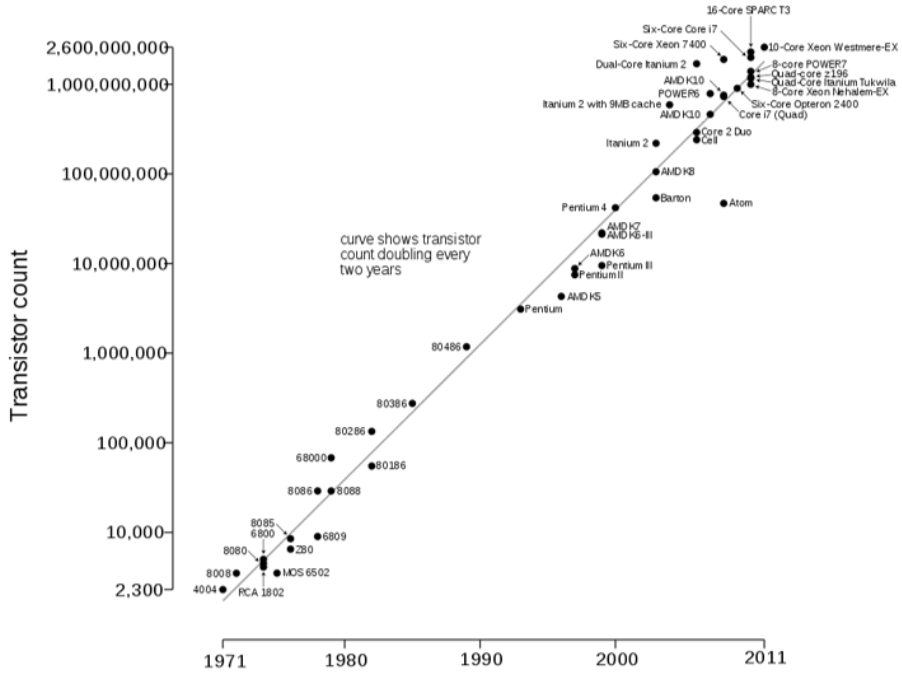


Figure 1.1: *Logarithmic scale of the number of components per integrated circuit, following Moore's law prediction trend.*

1.1.1. Dennard scaling

Moore's law main driving force has been the continuous scaling in MOSFET sizes; in this framework, Robert Dennard's work on scaling theory was pivotal in recognizing that this ongoing reduction was possible. In [3] he pointed out that, if voltages scaled with lithographic dimensions, then faster, lower energy, and cheaper gates were achievable. Table 1.1 lists the changes in integrated circuit performance as reported in Dennard's work, which result from scaling the circuit dimensions, voltages, and substrate doping by a dimensionless scaling factor κ.

Devices are scaled by a transformation in three variables: voltage, dimension, and doping.

Device or Circuit Parameter	Scaling Factor
Device dimension t_{ox}, L, W	$1/\kappa$
Doping concentration N_a	κ
Voltage V	$1/\kappa$
Current I	$1/\kappa$
Capacitance ϵA/t	$1/\kappa$
Delay time/circuit VC/I	$1/\kappa$
Power dissipation/circuit VI	$1/\kappa^2$
Power density VI/A	1

Table 1.1: *Changes in integrated circuit performance following Dennard scaling.*

All linear dimensions, including vertical (such as gate oxide thickness, junction depth, etc.) and horizontal (such as channel length and width) are reduced by a factor κ. The voltages applied to the device are reduced by the same factor; the substrate doping concentration is increased by κ. These scaling relationships were established in [3] by observing that the depletion layer widths in the scaled device are diminished proportionally to the device size due to the reduced potentials and the increased doping; also, the threshold voltage V_t scales down primarily due to the decreased insulator thickness. All circuit elements will therefore see their capacitances reduced by κ because of a decrement by κ^2 in their area, partially compensated by a shrinkage in the electrode spacing by κ due to thinner insulating films and reduced depletion layer widths. These reduced capacitances, multiplied by the unchanged device resistances V/I allow a reduction in the transition times, resulting in a diminished delay time by a factor of κ.

The power dissipation VI of an integrated circuit is then supposed to scale down by κ^2 due to scaled voltage and current levels; since the area of a given device is also reduced by κ^2, the power density, or power consumption per mm^2, remains constant. Therefore as transistors get smaller, they can switch faster and consume less power; the lower energy per switching event is exactly compensated by the energy increase due to having more gates and having them switch faster. As a consequence, computer architects were allowed to drastically raise clock frequencies from one generation to the next one without significantly increasing the overall device's power consumption.

Unfortunately however, no exponential law can last forever and since about 2005-2007 Dennard scaling appears to have broken down, with the scaling relationships diverging from the ideal ones proposed in [3]. The crucial problem is that not all device voltages

can scale, and particularly, since kT/q does not indeed scale linearly with κ, and also because the leakage currents are set by the transistor's threshold voltage (V_t), there is a limit to how low one can make a MOSFET's V_t.

As known, power in complementary metal-oxide semiconductors (CMOS) circuits can be modeled as:

$$P = QfCV^2 + VI_{leakage} \tag{1.1}$$

where N_{sw} is the number of switching transistors, f represents the operating frequencies, while C and V are respectively the capacitance and the operating voltage [4]; this relation shows a dependence of the power on the leakage current. As a matter of fact, at small sizes current leakage also causes the chip to heat up, which poses a threat for thermal runaway, further increasing energy costs.

With a fixed threshold voltage, simply changing the source voltage V_{dd} forces a trade-off between energy and performance; a direct consequence is that from the 130-nm technology node, V_{dd} has been scaling slowly, if at all [5]. This has made power consumption one of the fundamental concerns in modern chip design: it is almost guaranteed that the highest possible performance circuit will dissipate far too much power, therefore designers will have to focus on power efficiency in order to achieve high performance while keeping below power constraints.

1.1.2. The power wall

From Figure 1.2, using microprocessor data to track CMOS scaling from the mid-1980s, it can be noticed that through four technology generations, from the 2-µm (early 1980s) to the 0.5-µm (mid-1990s), the power savings from switching to lower technology nodes were enough that V_{dd} did not need to scale, but could indeed stay constant at 5 V; actually, it started scaling with the 0.5-µm generation up until the 130-nm technology due to power reduction constraints.

A relevant conclusion that can be drawn from Figure 1.2 is that, contrary to Dennard's predictions, while transistors get increasingly smaller with time, voltage scales at a different rate; the implication is that power dissipation per area does not stay constant with the scaling of dimension but it grows instead, thus preventing the subsequent increase in clock frequencies. This led CPU manufacturers to hit the so-called "power wall", increasing power dissipation of CPU chips beyond the capacity of inexpensive cooling techniques.

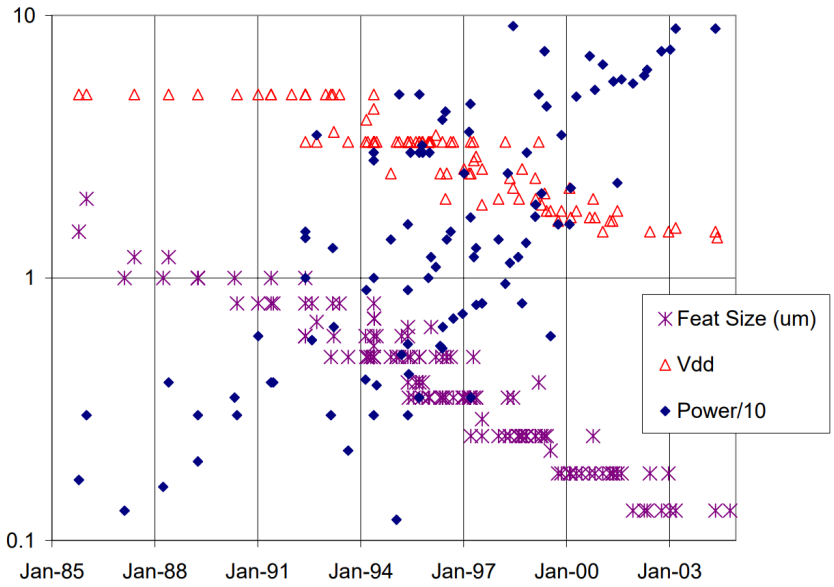


Figure 1.2: *Microprocessor threshold voltage, Power/10, and technology feature size versus year [5].*

1.1.3. Multicore computing

A direct result is that most designers then focused on multicore processors as an alternative approach to improve computing performance. An increased core count benefits many (though by no means all) workloads, but the increase in active switching elements from having multiple cores still results in increased overall power consumption and thus worsens CPU power dissipation issues. The end result is that only some fraction of an integrated circuit can actually be active at any given point in time without violating power constraints, and the remaining inactive area is generally referred to as *dark silicon* [6]. Figure 1.3 provides another interesting insight on the aforementioned trends spanning the last five decades.

Although the introduction of extensive parallelism in the form of increasing the number of logical cores in CPUs drove an increase in processor performance, Amdahl's law [8] demonstrates the diminishing returns from multicore computation using general-purpose processor cores: the theoretical speedup from parallelism is, as a matter of fact, limited by the sequential part of the task; this means that if a portion equal to $1/k$ of the task is serial, the maximum obtainable speedup is $k\times$ the original performance, even if the rest is easily parallel and more processor cores are added. Moreover, applications need to be developed specifically for parallel execution; despite the fact that software developers still managed to exploit parallelism by running different tasks independently on different cores, these independent tasks were mainly not made parallel, thus not truly exploiting

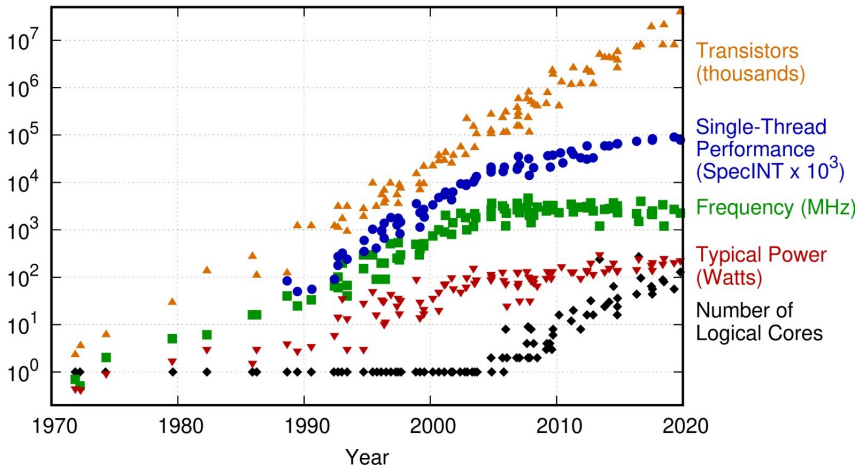


Figure 1.3: *Microprocessor trend data [7]. Orange: Moore's law trend; Green & Red: immediate implications of Dennard scaling breakdown; Blue: slowdown of single-thread performance according to SPECint benchmark specification; Black: number of parallel cores.*

the computational capability introduced by multicore.

Significant future advancements in computation, measured as the raw performance (operations per second), performance per area, and performance per Watt, will progressively come from cutting-edge domain-specific architectures that are specialized toward a narrower set of tasks: heterogeneous computing - a mix of different hardware architectures - seems like the only way to keep pushing computational performances [9].

1.1.4. Von Neumann bottleneck

Most of today's computing systems are based on von Neumann's architecture, a stored-program structure in which data must be moved towards and from a computational unit in order to be processed, and where an instruction and a data fetch cannot occur at the same time because they share a common bus; the vast majority of modern computers also use the same memory for both data and program instructions.

Therefore during the execution of different tasks, large amounts of data need to travel back and forth between the processing and memory units, producing significant costs in terms of energy and latency, both key performance factors for computing hardware. Notably, the latency associated with accessing data from memory units is critical for a wide range of applications such as the increasingly popular and data-intensive artificial intelligence (AI) workloads. There is a growing disparity between the speed of processing and memory units, generally referred to as the *memory wall*: since data movement is

much more expensive than computation, with slower improvements in data transfer rate, the higher-speed processor spends more time idle, waiting for data to be fetched from memory[10].

Another challenge is the energy cost associated with data movement considering that, as mentioned in Section 1.1.2, computing systems are severely limited in terms of power due to cooling constraints. Even at 45-nm CMOS technology node, dating back to 2008, the cost of multiplication between two numbers is orders of magnitude lower than that of fetching those numbers from memory, as seen in Table 1.2[11].

Integer		
Add	8 bit	0.03 pJ
	32 bit	0.1 pJ
Mult	8 bit	0.2 pJ
	32 bit	3.1 pJ

Floating-Point		
Add	16 bit	0.4 pJ
	32 bit	0.9 pJ
Mult	16 bit	1.1 pJ
	32 bit	3.7 pJ

Memory		
Cache	8 KB	10 pJ
	32 KB	20 pJ
	1 MB	100 pJ
DRAM		1.3 nJ
		2.6 nJ

Table 1.2: *Rough energy costs for different operations in* 45-nm *0.9 V.*

These numbers make two things very clear: first, the programmable nature of a processor has high energy overhead due to the fact that fetching the instruction and clocking the state registers that keep the data organized add a significant cost. Second, if high energy efficiency is required,the application must have very good data locality [11]. Conventional computing paradigms based on the von Neumann model that separates computational and memory structures have become outdated and less efficient for this increased demand.

1.2. In-memory computing

The current approaches, such as the use of hundreds of parallel processing cores or custom application-specific processors are not likely to fully overcome the challenge of data movement overhead; this means that innovative hardware architectures are to be explored, seeking a radical departure from traditional systems. One such non-von Neumann approach consists of performing computational tasks in the memory itself by exploiting simultaneously the physical properties of memory devices, their array-level organization, the control logic, and the peripheral circuitry. Solving tasks within a memory unit is typically referred to as *in-memory computing* (IMC), which presents an opportunity for neuromorphic accelerators to thrive.

Neuromorphic computing, as a matter of fact, is leveraged for analog hardware acceler-

ators that can perform computations directly inside the memory array where the operational parameters are stored, seeing that it distributes both computation and storage among a vast amount of relatively primitive and low-power units referred to as *neurons*, each communicating with hundreds or thousands of other neurons through adaptive connections called *synapses* [12]. Researchers are currently exploring neuromorphic computing at a large scale, inspired by mammalians' nervous system, in which the boundaries between processing and memory units are blurred [13], in order to overcome the shortfalls of CPUs and GPUs for data-centric applications due the end of Dennard scaling and von Neumann bottleneck, cited respectively in paragraphs 1.1.1 and 1.1.4. Compared with digital computers, the human brain only needs an extremely low power (about 20 W) and frequency (usually in the few Hz range) for information processing [14], thus providing a living biological example to help introduce novel energy-efficient computing paradigms to tackle data-intensive workloads such as machine learning and AI.

Besides mitigating the latency and energy costs associated with data movement, IMC offers significant advantages in reducing computational time complexity as well; this is mostly due to the massive task parallelism achievable with the employment of dense arrays with potentially millions of memory devices performing computation [15].

Figure 1.4 [15] offers a schematic illustration of a comparison between conventional computing systems and IMC. In conventional structures, computation takes place in the CPU according to instructions that are fetched from a working memory according to the von Neumann architecture [16]; such working memory, most commonly a Dynamic Random Access Memory (DRAM) is usually located on a physically separate chip, resulting in cumbersome latency and energy consumption for data-intensive tasks.

IMC instead performs data processing in situ within a dedicated memory array, suppressing the latency for data and instructions fetch and output upload in the memory, thus overcoming the von Neumann bottleneck. Analog processing inside such an array can inherently parallelize the matrix algebra computational primitives that underlie many machine learning algorithms, thanks to the fact that, due to its specific architecture, computation can take place along several current paths at the same time.

These intrinsic advantages can potentially translate into dramatic improvements in both the computational capacity and energy efficiency of neuromorphic hardware [17]. IMC also benefits from the high density of the memory arrays, translating in excellent scalability and the capability of 3D integration. Thanks to these many advantages the number of projects that are being described as neuromorphic computing, albeit being utilized in very different applications, has grown very large as demonstrated by recent surveys

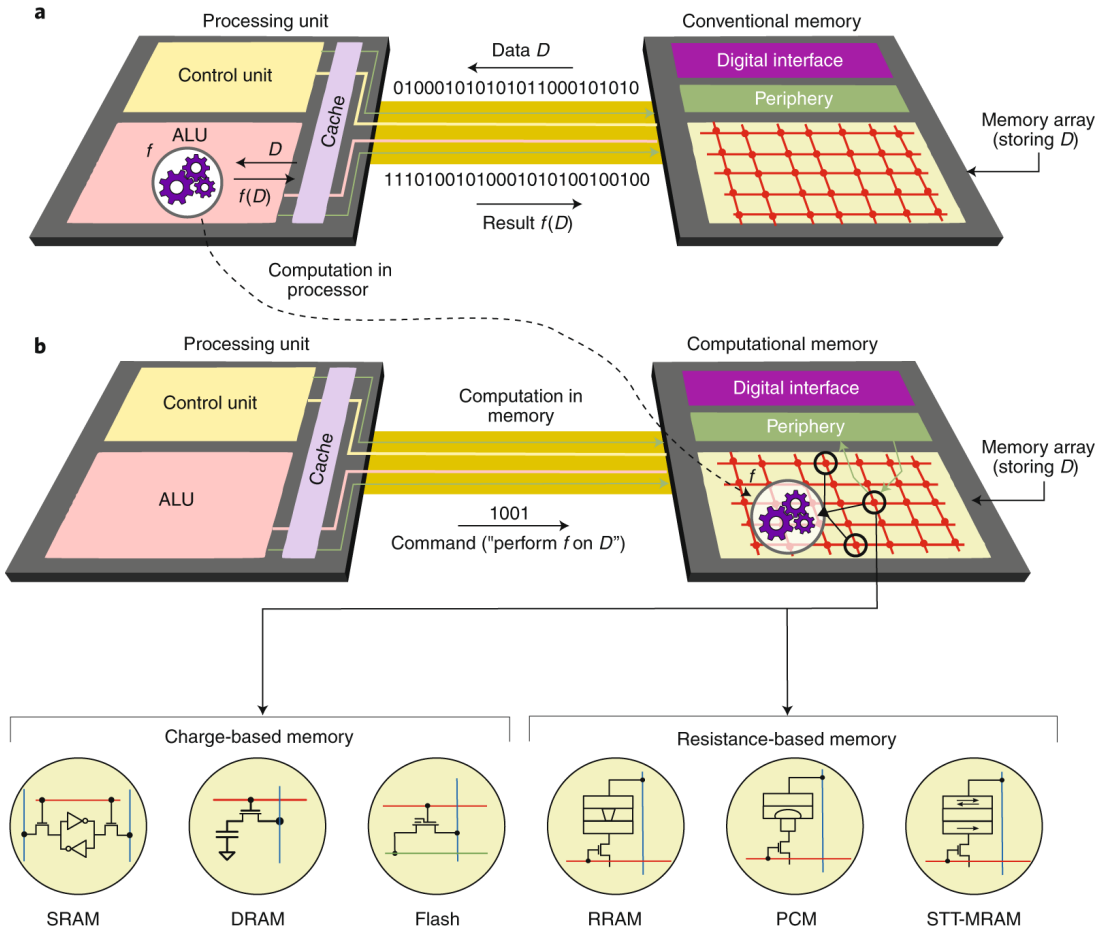


Figure 1.4: ***(a)*** *In a conventional computing system, when an operation f is performed on data D, D has to be moved into a processing unit, leading to significant costs in latency and energy.* ***(b)*** *With IMC systems, the operation f(D) is performed within a computational memory unit by exploiting the physical attributes of the memory devices, obviating the need to move D to the processing unit. Tasks are performed within the memory array and its peripheral circuitry without reading the content of each individual memory element.*

of neuromorphic hardware that end up listing hundreds or even thousands of different citations [18, 19].

Many of these works involve circuits that attempt to carefully mimic something that we can currently observe with moderate accuracy within the brain, usually at the scale of a few neurons, such as the exact synaptic response, local connection patterns between neurons, or local learning rules such as spiking time-dependent plasticity. Other applications instead draw inspiration from the human brain in order to design architectures that accelerate computational kernels, such as solving linear systems with in-memory matrix-vector multiplication (MVM) and vector-matrix multiplication (VMM) [17].

Both digital and analog architectures targeted towards solving linear systems that pertain to machine learning algorithms will be discussed. The analysis will focus on analog systems aimed at performing neural network inference.
An advantage of a fully analog ASIC in the field of gamma imaging is related to the fact that input signals from SiPM readout circuits are analog voltages, therefore performing analog computations exploiting Ohm's law and Kirchhoff's law allows a fast and accurate evaluation bypassing the need for analog-to-digital conversion, effectively enabling ADC-less position sensitivity.
Analog computation proved to be also more agile than FPGA implementations of neural networks: the high data processing parallelism due to the memory array architecture enables faster inference on the same dataset, significantly fewer clock cycles, and lower energy consumption with respect to the conventional digital approach. This is particularly useful in the field of medical imaging and notably multimodal imaging, where implementing agile computation and curtailing power consumption are of utmost importance.

1.3. Accelerators for Neural Network Inference

Deep neural networks (DNNs) provide state-of-the-art performance for a multitude of various tasks (including but not limited to image classification, speech recognition, and natural language processing) despite being particularly complex to solve, seeing as typical networks usually comprise up to tens of layers and millions of weights that translate into billions of multiply-accumulate (MAC) or multiply-add (MAD) operations required for a single inference, making it difficult to run them on resource-constrained platforms. Moreover, DNNs and in particular convolutional neural networks (CNNs) layers are often referred to as *embarrassingly parallel*, making them particularly suited for hardware acceleration. Embracing domain-specific hardware accelerators could therefore improve speed and energy consumption by orders of magnitude relative to general purpose processors [20]. However, it is important to note that progress in this field heavily relies on the simultaneous design of algorithms and application-specific hardware [21].

1.3.1. Figures of merit for DNN inference accelerators

Assessing the system-level advantages of NN hardware accelerators is a challenging effort, debated for instance in [22] and [23]. The basic aspects will be discussed in this section.

Efficiency Benchmarking: the most objective way to compare the efficiency of DNN inference accelerators would be to measure their total energy expenditure per inference for the same task and accuracy. However, this is difficult for the analog

and mixed signal design framework, where many applications are only partial in hardware realizations and emulate the full network only in software. A compromise is then to quantify these subsystems by their number of performed operations per Watt: this is typically expressed in tera-operations per second and watt, or TOPS/W. One TOPS/W corresponds to 1 pJ per operation and one MAC is defined as two operations. It is also frequent practice to compute the area efficiency in TOPS/mm^2 considering that energy efficiency alone is not useful without high compute density. The latest advancements on these benchmarks are tracked in Figure 1.5 from the online survey available at [24]. However, it must be noted that comparisons in terms of these metrics alone are inherently flawed: for instance, it is fairly easy to achieve high efficiency in scaled technologies and for small bitwidths; the relationship between total inference energy and the operands' bitwidth is unknown [25], although it must be noted that low bit resolution typically requires more network weights. Moreover, smaller arrays have inherently higher efficiency, but it is unclear whether this array-level performance will translate to a higher efficiency when the whole accelerator is concerned, considering that limited array size might lead to an increased memory access burden: as a matter of fact, many works report only partial, array-level energy and area efficiency. Some impressive results are reported for analog architectures, such as the design presented in [26], that performs image classification on a CIFAR-10 dataset [27] with an 86% accuracy and an expense of only 3.8 µJ in 28-nm CMOS, and the one in [28] which achieves a 192 TOPS/W efficiency at $V_{DD} = 1.2$ V and a 400 TOPS/W efficiency at $V_{DD} = 0.85$ V with an SRAM IMC in 65-nm CMOS, albeit both applications being referred to binarized networks. Operand scaling in the design of Jia *et al.* [28] from 1 bit to 8 bits comes at a significant cost, as both energy and area efficiency scale linearly with the number of bits. The 4-bit analog fabric in [29] reports a 351 TOPS/W result, even though for a very small array (64×64) and in 7-nm FinFET technology.

Flexibility: while analog signal processing arrays, which will be introduced in Section 1.3.3, can be studied to optimize memory access amortization, their size is fixed during the design stage. This hinders the flexibility that is sought after to support the latest developments in DNN architectures: if the network's architecture is modified, the hardware array utilization becomes sub-optimal, potentially introducing issues with reduced ADC or input buffer amortization, extra analog to digital (A/D) conversions, post-ADC digital accumulation, and data movement penalties due to potential weight reloads. This issue is seen in [30], which reports a $\sim 4.5\times$ efficiency

degradation when external buffer accesses are added to the energy budget. Flexibility is also difficult to manage for dense IMC fabrics, where the tight pitch may not allow the inclusion of any static or dynamic reconfiguration circuitry. A possible solution might be a middle-ground architecture, which does not push density to the limit, but in return offers better reconfigurability for flexible DNN mapping and execution.

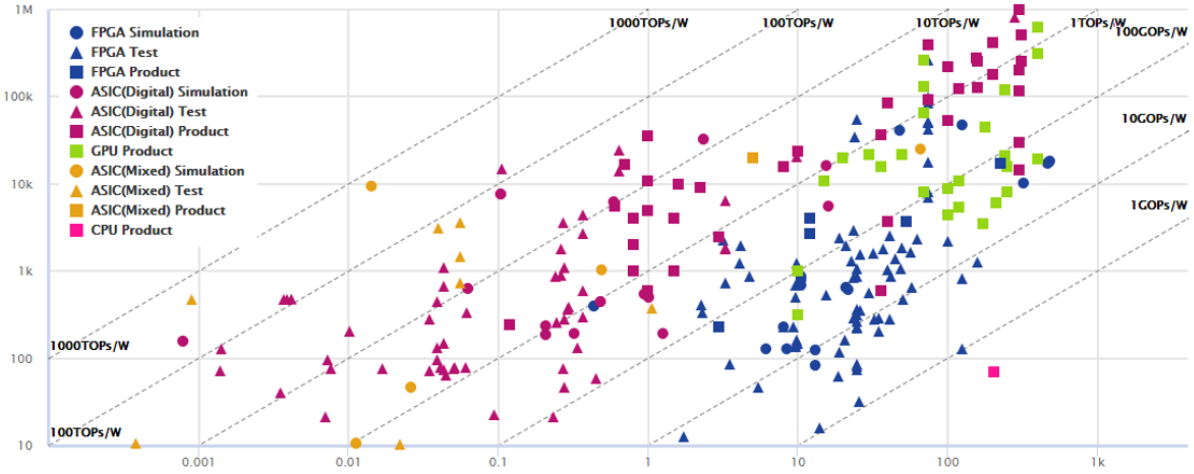


Figure 1.5: *Neural network accelerator comparison in terms of speed (expressed in GOPS) over power.*

1.3.2. Digital Neural Network accelerators

Digital neural network accelerators include both systems based on field-programmable gate arrays (FPGAs) and application-specific integrated circuits (ASICs); most solutions aim to accelerate a neural network inference, with training still largely performed offline by GPUs. Surveys for these types of applications can be found in [31] and [32].

The FPGA is a popular platform for the development of flexible hardware for various applications, such as DNNs. Its reconfigurability enables a high degree of hardware and software co-design and the possibility of energy-efficient inference operations [17]. Nevertheless, reconfigurability needs to be supported by a significant overhead, which in turn limits the available processing resources on-chip, and multiple FPGAs might be needed for the implementation of DNN inference, thus increasing area and power consumption and jeopardizing the benefits of hardware accelerators altogether. A trade-off can be achieved by balancing a collection of FPGAs interconnected among them by a high-speed reconfigurable network which allows an efficient inference with low latency; for such a system to work efficiently, the network must be partitioned across the FPGAs such that each partition requires only those weights that are stored on the single FPGA on-chip

memory (see Microsoft's Project Brainwave at [33]).

ASIC-based digital hardware accelerators for DNNs realize stateful logic in a memory circuit by comparing the voltages among two or more devices and conditionally inducing switching in one of these devices or in an additional one, by the threshold-dependent set/reset operations [34]. This approach improves the density of logic gates and suppresses the latency associated with data transfer for digital computation. On the other hand, digital IMC suffers from an increased energy per operation due to the need to change the state of a device during computation. The state switching also increases the time for logic operation and critically limits the lifetime of the circuit due to endurance constraints.

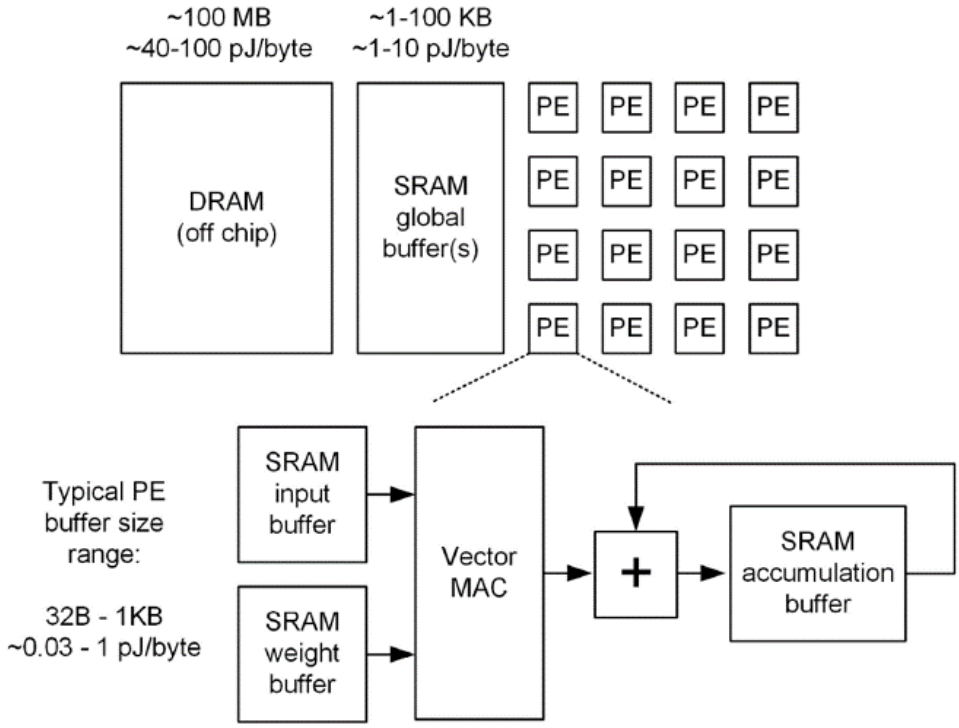


Figure 1.6: *Simplified block diagram of a digital DNN accelerator [20]. A typical design consists of three memory levels: an external main DRAM, a global SRAM, and local buffers in the single PEs.*

However, in the digital world there exist many efficient architectures that handle the workload well; generally, these systems have a hierarchical structure based on elementary units referred to as *processing elements* (PEs) that measure up to a few hundred micrometers on a side and include a small amount of buffer memory plus strictly separated MAC arithmetic. Most digital architectures are organized in a PE array that performs a large number of MAC operations in parallel, as shown in a generic diagram in Figure 1.6 [20].

The main memory, often off-chip, stores all the network's weights, since a capacity of ∼10 MB/100 MB is often needed. Data access from memory is highly inefficient from energy consumption and latency point of view, but when a weight is fetched from memory it is temporarily cached in a global Static Random Access Memory (SRAM) buffer or in the local PE; this proves especially useful in tackling CNNs, where weight reuse is a main advantage, and filter weights can be kept while running through the input image. Similarly, partial sums from a previous filter position may be reused for the computation of the next output pixel. In Figure 1.7 the architecture of a PE is shown, as seen in the state-of-the-art design presented in [35], where an array of 16 PEs computes a total of 1024 8-bit MACs per clock cycle achieving a 9.5 TOPS/W energy efficiency and a 1.29 TOPS/mm^2 area efficiency.

In such a design, data reutilization is critical for maximizing the chip's efficiency and a carefully crafted memory hierarchy exploits all data-reuse opportunities.

As a matter of fact, it is estimated that a 16-bit MAC operation performed in a digital 28-nm CMOS architecture costs only about 75 fJ, with a possibility for this value to drop below 10 fJ for lower bitwidths. Access to a memory buffer (for instance a 1 KB SRAM) requires roughly 1 pJ, so it must be amortized across numerous computations [20].

While data reuse is pivotal, it is important to note that a large amount of other parameters can be optimized, such as dataflow, resource allocation (mostly the sizing of the memory hierarchy), and spatial or temporal reuse. The combination of these options leads to an exceedingly large optimization space that was explored in [22], where it was observed that most dataflow options can be near-optimal, as long as a high degree of data reuse and resource utilization is assured, and that the accelerator's energetic efficiency is most closely related to the hierarchical design of the memory system. For instance, it was illustrated that the size of the PE buffers in particular should be minimized to balance the energy across the system [22], considering that they are typically accessed in every compute cycle; however, even with small PE buffer memories, each MAC operation will come with significant memory access overhead, typically making it difficult to improve digital PEs beyond ∼150 fJ/MAC-200 fJ/MAC for 8-bit operands (see the state-of-the-art design of Zimmer *et al.* [35]). These 200 fJ/MAC are spent just to deal with the architecture's inner memory, posing a ceiling at 10 TOPS/W; this line (seen in Figure 1.5) is a frontier for digital designers, and the ones that do go above it tend to do some form of mixed computing.

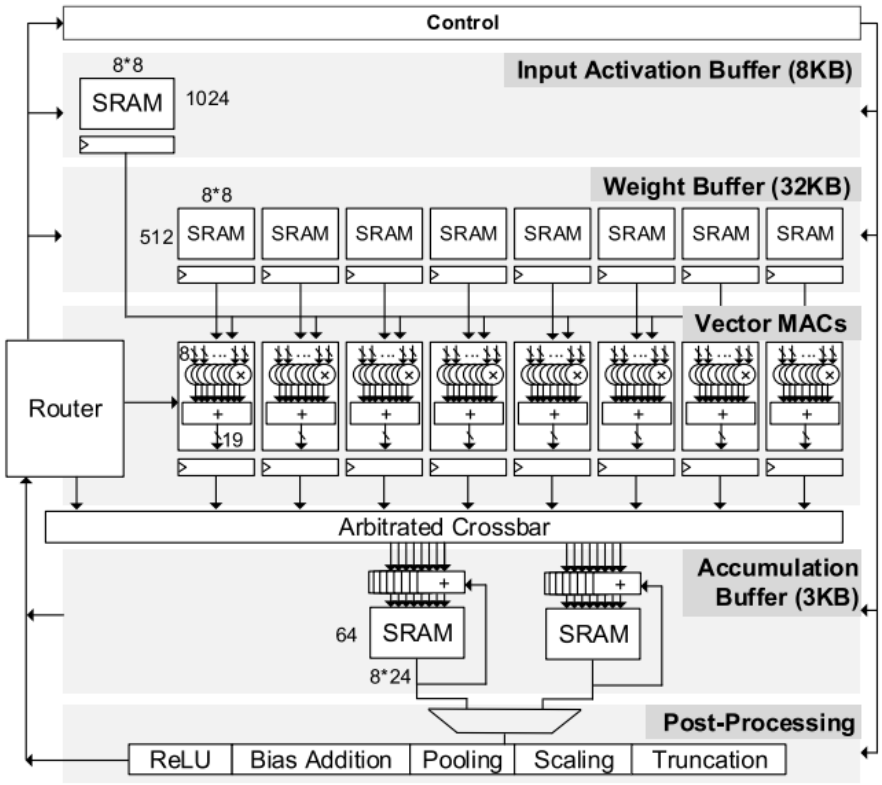


Figure 1.7: *Architecture of a PE in [35], which executes convolutional layers, fully-connected layers, and postprocessing functions like bias addition, rectified linear unit (ReLU), and pooling. A PE has eight lanes, each using a different weight tensor to generate elements for a separate output channel. Within each lane, an 8-bit precision vector MAC multiplies eight input elements from separate input channels with eight weight elements and sums them to calculate a single output value. Local input activation, output activation, and accumulation SRAMs buffer data for the datapath. Minimizing accesses to these SRAMs is critical to maximizing energy efficiency.*

1.3.3. Mixed-signal and analog Neural Network accelerators

Looking at the digital implementation, it is evident that there is no need for a more efficient compute element (such as an adder or a multiplier), because the most relevant issue is moving the data to and from those elements and most of the energy is spent in inputs, outputs, and weights transfer; moreover, each MAC operation triggers several accesses to the memory, and once the local buffers have been filled, data movement to a larger global buffer is mandatory, thus introducing unwanted energy overhead. Therefore a better fabric is needed in order to reduce data movement overhead, such as the one shown in Figure 1.8. In analog and mixed-signal applications there are new degrees of freedom

because analog MAC operations are implementable; these architectures can have denser granularity than digital systems, intersecting compute kernels with memory elements and thus shortening the distance from memory to compute element and thus the latency.

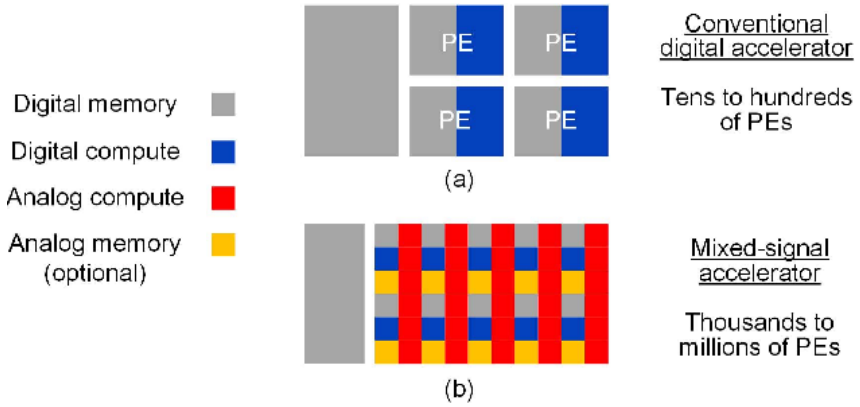


Figure 1.8: ***(a)*** *Typical digital DNN accelerator* ***(b)*** *Mixed-signal accelerator, mixing memory and computation.*

Figure 1.8(b) schematically outlines the pursued approach. Digital compute is kept for operations that are difficult to mirror in the analog domain such as non-linear activations, as well as dense digital memory for fast data caching. Analog arithmetic instead is included to enable low-energy spatial accumulation: instead of writing a MAC result to a local memory buffer, it can be accumulated via current or charge summation on a wire thanks to Kirchhoff's current law, achieving an immediate computation. Multiplication can instead be implemented by exploiting the characteristics of non-volatile memory (NVM) elements, useful for both weight storage and synaptic operations thanks to Ohm's law (current = voltage $\times$ stored conductance). Analog circuit fabrics are motivated by additional advantages: first, analog inference is inherently robust against small computation errors [20] such as those introduced by relatively low electronics noise, and it can be further desensitized by injecting noise during the training phase [36, 37], which in turn allows bitwidth reductions below 4 bits[25]. Second, analog computation appears to be more efficient than digital for low bit precisions due to their structural simplicity [38–40], as shown in Figure 1.9 [20]. For higher bitwidths instead, the analog energy cost rises abruptly and non-idealities are dominated by thermal noise; since its rms value is proportional to $\sqrt{kT/C}$, achieving an extra bit of precision in analog domain requires quadrupling the capacitance and hence the energy, as a direct consequence of (1.1).

However, typically below 8-bit precision, analog architectures appear preferable than digital in particular for edge inference applications that require only small networks. With the recent advancements in dense NVM structures, computation can be performed locally

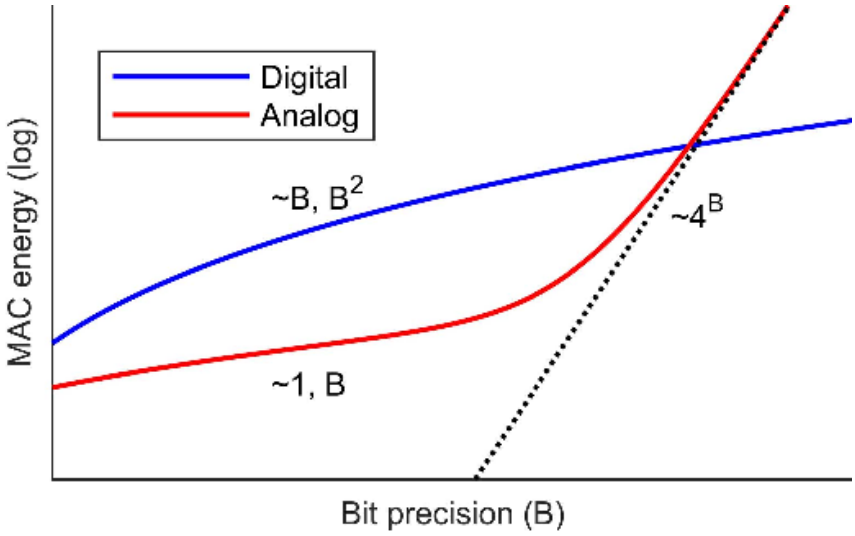


Figure 1.9: *Multiply-accumulate (MAC) operations energy cost versus bit precision for both digital (blue) and analog (red) compute fabrics [20].*

within the memory elements, therefore further savings in power consumption are achieved by the reduction of off-chip memory access, as anticipated in Section 1.3.2 [41]: the reasoning behind the development of analog compute fabrics is not limited to the reduction of compute energy cost, but extends to introducing new degrees of freedom in positioning and density of compute elements, playing a critical role in a DNN's data flow and thus potentially decreasing data movement and memory access. Close attention must always be paid to effectively amortize potential D/A and A/D conversions.

A variety of unit cell sizes are possible within this generic framework. One extreme, more closely related to IMC (Section 1.2), prioritizes array density and makes do with whatever compute function can be integrated on the scale of a memory cell. The basic processing element in such a system is an array (or crossbar) of typically resistive memory devices. Many of the candidate devices for this type of application are emerging NVM technologies, although it is also feasible to leverage mature technologies such as flash memory.

Another plausible idea is to employ larger, memory-like analog or mixed-signal cells that are still substantially smaller than a digital PE but offer enhanced capabilities relative to the denser arrays. These memory-like cells (Figure 1.10 [20]) are referred to as *unit elements* (UEs), and can be viewed as synapses due to their inherent capability to perform analog MAC operations with stored weights that remain stationary for many compute cycles. The fundamental advantage of having stored weights is that data movement and weight access energy overhead are both reduced.

For low-energy accumulation, an analog charge or current bus collects column-wise partial

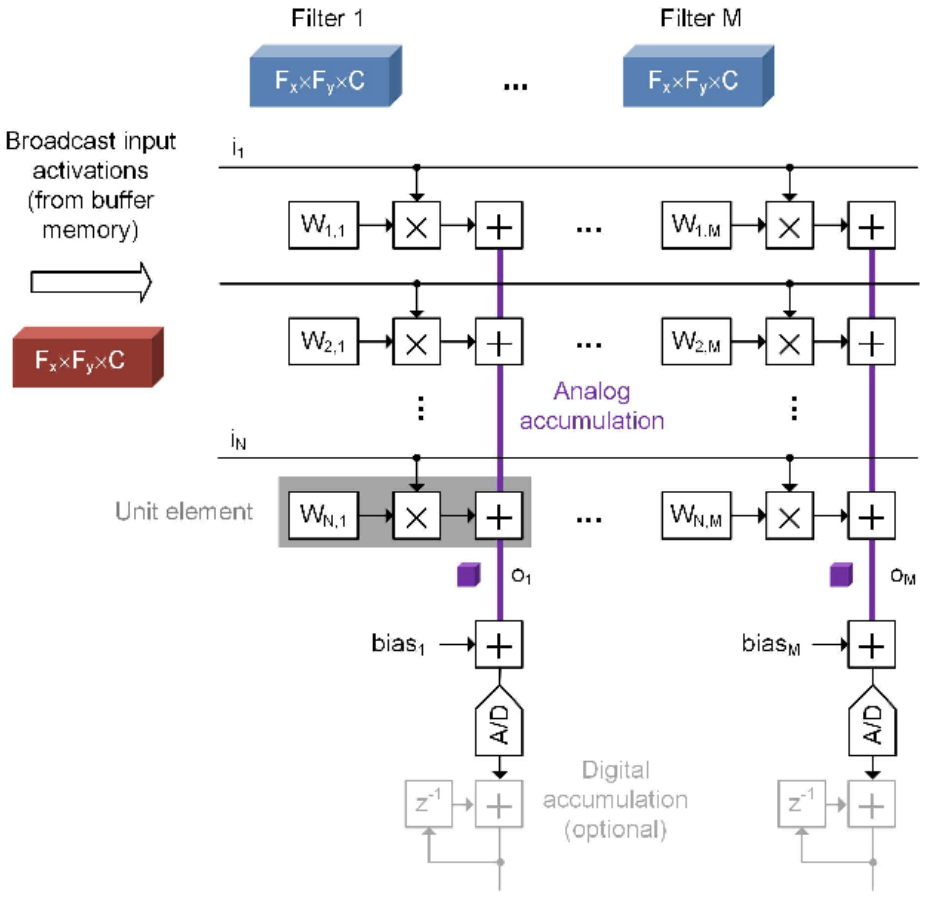


Figure 1.10: *Generic layout of mixed-signal array [20] showing a crossbar of small unit elements (UEs). The input is an activation tensor (in red, on the left of the crossbar array) unrolled into a vector and forwarded to the array from the left. The columns store the filter weight tensors (in blue, upper side of the crossbar), also unrolled into vectors. Then, a dot product is computed between the input vector and the weight vectors stored in each column. Potentially, when involving CNNs the array can be sized such that an entire filter is held in a single column, turning each column into a complete neuron. If this is not possible (or desirable), the accumulations can be completed in the digital domain, as shown in the bottom part of the figure. Also, the biases that must be added to the output vector are shown. The number of columns in the array can potentially be chosen to process all the filters of a DNN layer in parallel to attain maximum spatial reuse of global input activation buffer access. Alternatively, the weights can be reloaded after several array computations, still offering substantial amortization of global activation and weight buffer access energy for a reasonably large array.*

products without involving digital logic or memory. Since most applications of mixed-signal compute arrays involve digital inputs and outputs, the cost for the analog summation is that energetically cumbersome memory access and a D/A conversion are required for each row, while an A/D conversion is required for each column output; even with low-resolution ADCs, an energy often higher than a pJ is required and without heavy amortization it might be challenging to perform inference with less than 10 fJ/MAC. However, the fundamental advantage of these fabrics is that memory access overhead is amortized across many columns (exploiting the spatial reuse of input activations), while the A/D conversions are amortized across many rows: this is the reason why very large arrays are essential. Implementations of the array concept in Figure 1.10 are discussed below, focusing on a subset of options that have proven more attractive from an efficiency maximization point of view [20], and omitting for instance approaches that rely on devices such as SRAM cells, which seem less performative for this type of application due to high variability [42].

Binary Switched Capacitor Arrays

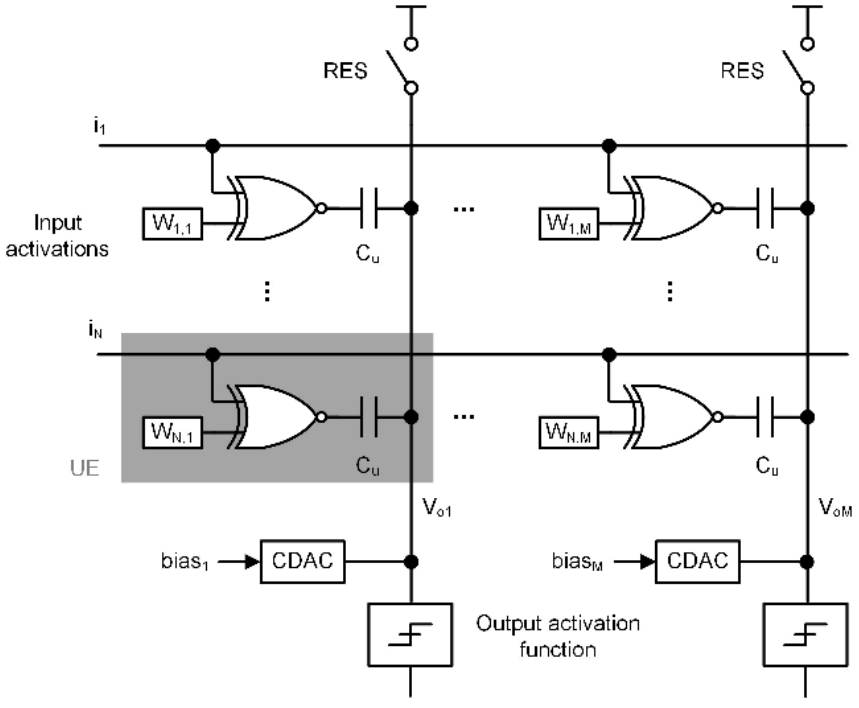


Figure 1.11: *Mixed-signal array with binary weights and activations [20]. Binarization reduces multiplications to XNOR operations [43].*

Figure 1.11 shows a schematic of a switched capacitor (SC) MAC array for binary DNNs. The appeal of this approach mainly stems from the limited mismatch of metal-oxide-metal (MOM) capacitors in modern CMOS technology: even for subfemtofarad capacitors,

standard deviations of around 1% or less are achievable [44, 45] enabling compact, low-energy, and robust accumulation at high precision.

An implementation of a capacitive 1024×64 fully-binary array is reported in [26], which integrates fully analog accumulation with a high cell density and achieves, as said, a 3.8 fJ energy overhead per inference with a 700 TOPS/W efficiency. It is important to note how charge sharing on the accumulation bus is implemented: an advantage of the circuit depicted in Figure 1.11 is that dynamic dissipation occurs only if the input activations switch, similarly to standard CMOS circuits. However, a charge reset (RES) must be periodically performed to overcome leakage, posing the need for additional transistors or dedicated NOR gates inside the UE [26]. Other applications, such as the ones presented in [28] and [46], do not require the RES signal nor extra transistors, but dynamic energy is dissipated even if the input activations do not switch, inducing a significant energy overhead, since the activity factors of DNN input activations tend to be below 10% [26].

Another key design consideration relates to the dynamic range of the charge accumulation and the analog circuit errors. Unfortunately, the requirements depend on the DNN algorithm, inference accuracy, signal statistics, and the employed training methods, which can be skillfully tuned to make the network more robust [47, 48].

It is generally important to keep in mind that the voltage noise on the accumulation wire tends to be relatively small if compared to the supply voltage V_{DD}. In terms of random errors, the DNN-level simulations in [26] suggest tolerable standard deviations in the order of V_{LSB} for binary networks, where $V_{LSB} = V_{DD}/N_r$ represents a unit increment in the accumulation voltage and N_r denotes the number of rows in the crossbar array. Assuming random uncorrelated binary inputs and weights with $Pr(0) = Pr(1) = 0.5$, the accumulation result has mean $N_r/2$ and standard deviation $\sqrt{N_r}/2$. Therefore, if $V_{DD} = 1$ and $N_r = 256$ as in [20], an easily tolerable noise of $\sim 4\,\mathrm{mV}_{rms}$ is observed on the accumulation lines. The kT/C noise at the charge summation node and the capacitor mismatch are also negligible [20].

Thanks to these relatively mild circuit requirements, chips that implement binary MAC operations with subfemtofarad switched capacitors easily achieve near 4 fJ/MAC, with all peripheral memory access considered. Fully digital architectures with a comparable design and dataflow can, however, get close to these figures [49]. Therefore, mixed-signal processing does not appear to be a game changer for binarized DNN accelerators,. As a result, it makes sense to look at multibit implementations, which can cover a larger range of applications.

Multibit Switched Capacitor Arrays

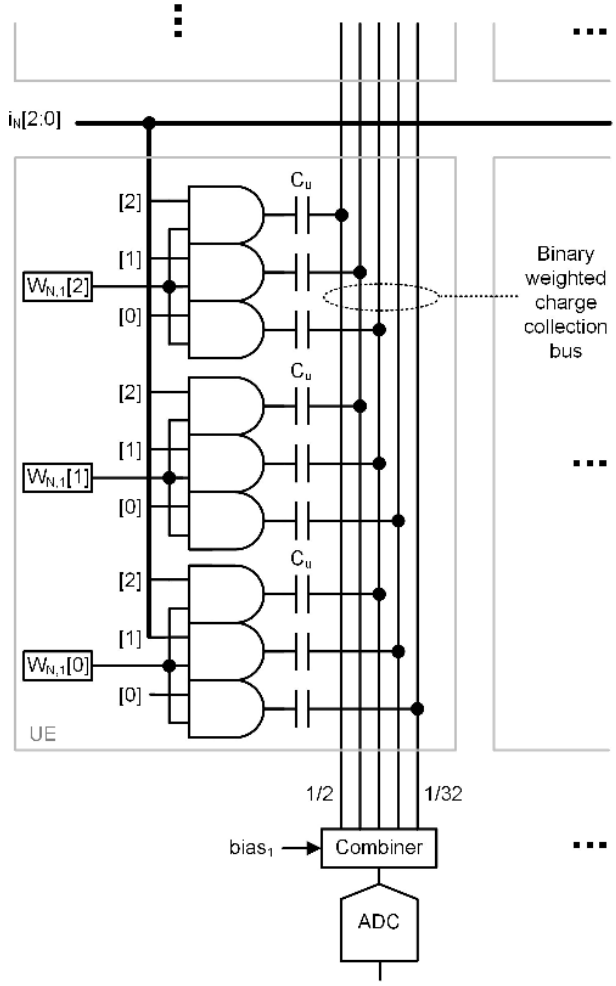


Figure 1.12: *Mixed-signal unit element (UE) implementing a 3-bit MAC with switched capacitors [20].*

Figure 1.12 depicts an extension of the concept represented in Figure 1.11 to a multibit operation. The circuit assumes unsigned operands for simplicity, but it may be adjusted to perform signed operations. The topology is similar to that of a digital multiplier, but instead of employing digital full adders, the local partial products are collected in the charge domain. As a result, the charge collection bus serves two purposes: local accumulation and array-level accumulation. The voltages on the charge collection bus must be combined in a binary-weighted manner before or during A/D conversion.

To reduce overhead, the combiner might be included within the ADC's input network (e.g., using binary splitting of the sampling capacitance). Instead of allocating nine AND gates for parallel 3-bit operations, it is also viable to process the input activations serially, as proposed in [28]. This solution splits the weight bits into distinct columns and

serializes the input activation bits into the array, with one A/D conversion per time step and column; the final result is then obtained in the digital domain. This system has the advantage of higher density (one AND gate per UE) and programmability, but it comes at the cost of lower throughput and additional A/D conversions. This serial approach is preferred when obtaining high density and flexible bitwidth is the primary goal, while a more parallel topology may enhance throughput and energy efficiency.

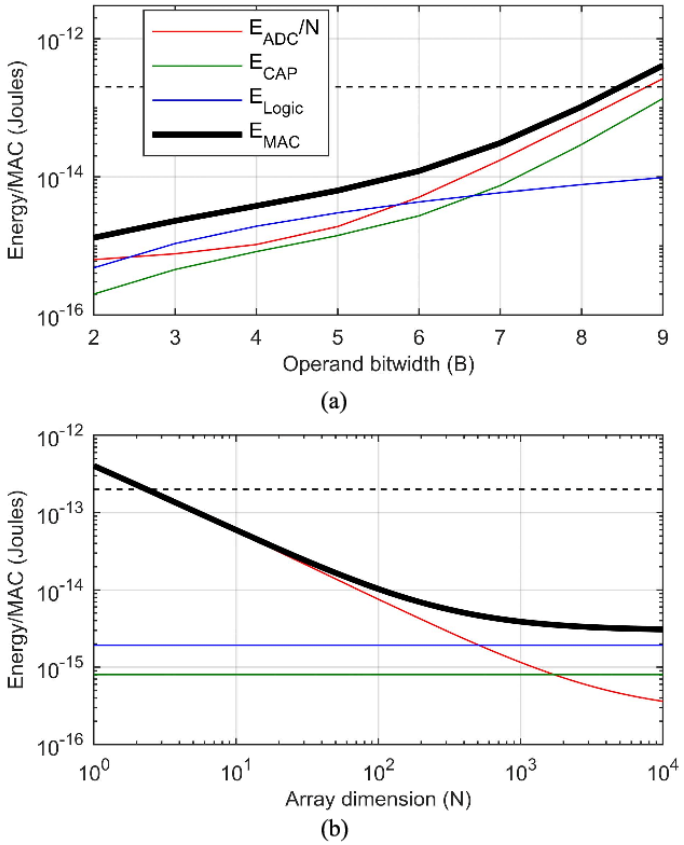


Figure 1.13: *MAC energy estimate for the multibit array in Figure 1.12 assuming* 28-nm *CMOS.* ***(a)*** *Energy versus operand bitwidth* (B) *for fixed array dimension* $N_r = 3 \times 3 \times 128 = 1152$. ***(b)*** *Energy versus array dimension* (N_r) *for fixed bitwidth* $B = 4$.

To get a sense of the fully parallel configuration's energy efficiency, Figure 1.13 shows a quantitative analysis on the operands bitwidth B (assumed equal for input activations and weights), and the vertical array dimension N_r.

The energy per MAC operation E_{MAC} is equal to

$$E_{MAC} = \frac{E_{ADC}}{N} + E_{CAP} + E_{Logic} \tag{1.2}$$

where E_{ADC} energy required for each ADC conversion, which is amortized over the number

of rows in the array (N_r); E_{CAP} and E_{Logic} instead measure the energy consumption due to unit capacitances (C_u in Figure 1.12, also referred to as C_{LSB} throughout this work) and logic gates in each UE. In Figure 1.13 it is evident that, assuming a large array with $N_r = 1152$, high amortization is achieved for low bitwidths ($B = 2 - 6$), where the ADC energy contribution is comparable to that of logic gates activity. However, due to the severe tradeoff between energy and thermal noise requirements, the ADC eventually dominates the energy consumption at higher bitwidths. Beyond $B = 8$, the estimate exceeds 200 fJ/MAC, which is also achievable with purely digital implementations [35]. Therefore, competing with digital topologies above 6-7 bits is challenging and will likely yield poor results.

The second plot in Figure 1.13 represents the energy against the number of rows N_r with $B = 4$. It demonstrates that, in order to attain greater efficiency, the A/D conversion should be amortized across at least a few hundred rows. For very large N, however, the amortization benefits stop, and the ADC energy flattens.

Resistive memory arrays

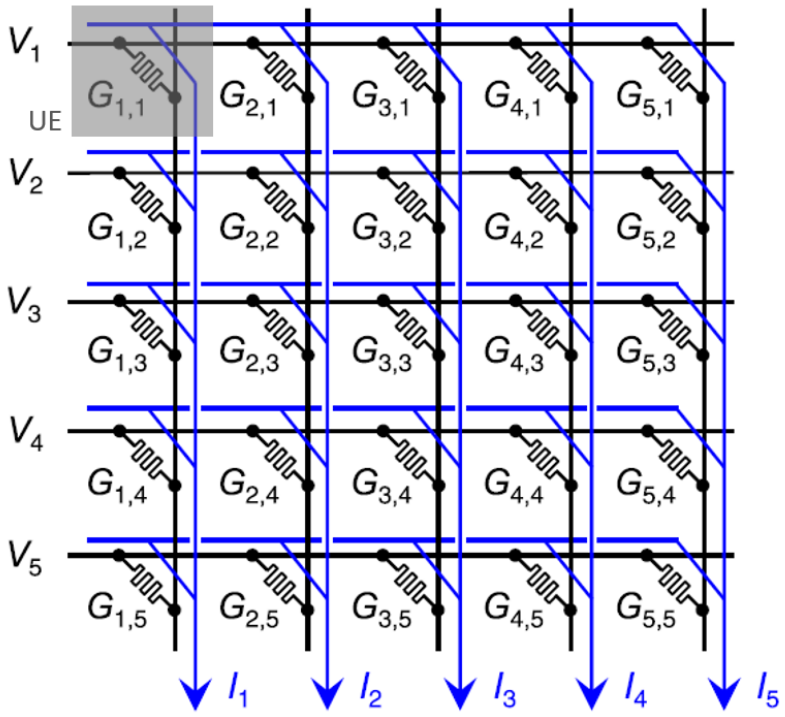


Figure 1.14: *Basic concept of a massively parallel analog VMM within a resistive memory crossbar. The currents accumulated by the columns are given by* (1.3)

Under reasonable chip area constraints, none of the processing arrays explored so far are

dense enough to store all the weights of a large DNN (for instance $> 10\,\text{MB}$) on a single die. However, in-memory computing and emerging dense NVM technologies may make this viable. Because their resistive memories are non-volatile, power cycling is possible without having to reload the weights from external memory. Analog neuromorphic accelerators based on non volatile memory arrays can dramatically minimize the energy and latency costs associated with data movement by incorporating neural network calculations directly into the memory components that hold the weights. Computation within these arrays is also inherently parallel, and thus can be leveraged to accelerate neural network primitives such as VMM and MVM, allowing for more efficient topologies.

Figure 1.14 shows the basic structure of a fully resistive memory array in a 1R configuration; some configurations are also feasible, and often preferable, typically referred to as 1T1R, where a transistor is in series to the resistive element in each UE. The series transistor grants better control on the crossbar array, proving useful during weight programming since a single cell can be activated by selecting the corresponding row and column; the access transistor can also be used to more efficiently pass VMM inputs to the memory elements during inference: by applying the input as a voltage on a high-impedance access gate, the signal can remain free from distortion caused by parasitic resistance. Another advantage is that the transistor works as a current limiter during programming phase. Unfortunately though the UE results larger, thus partially sacrificing the array density of 1R configurations.

To perform a VMM within the array, all the rows (also referred to as *wordlines*) are activated simultaneously, with voltage V_i applied on row i. The total current collected by jth column (or *bitline*), which is assumed to be held at a fixed voltage, is given by

$$I_j = \sum_{i=0}^{N_r-1} G_{ij} V_i = 1 \qquad 0 < j < N_c - 1 \tag{1.3}$$

where G_{ij} is the conductance of the memory element at the crossbar position (i, j), N_r is the number of wordlines and N_c the number of bitlines. (1.3) implements a vector dot product where the multiplications are realized by Ohn'ms law and accumulation occurs thanks to Kirchoff's current law, as previously stated in Section 1.3.3. Since the currents flow through all the columns in parallel, the crossbar executes a full VMM in a single operation; the MVM operation instead, which uses the transpose of the same weight matrix, can likewise be executed by driving the columns and reading the currents on the rows. The bias vector is usually added as an extra row in the array. The sum is then usually processed in the current domain via a transimpedance amplifier, integrator [50]

or current-mode comparator [51]; voltage-domain readouts are also feasible.

The physical latency of a crossbar VMM is determined by the time required to charge each row of the array, scaling as $O(N^2)$ for an $N \times N$ array. However, this RC delay tends to be much smaller than the latency of the peripheral circuits such as the ADC and the DAC [17]. Energy scaling is favorable too, as by charging all the rows in parallel it scales as $O(N^2)$. While in-memory matrix computations are highly efficient, it is often the case in analog accelerators that the area and energy consumption is dominated not by the crossbar but by the peripheral circuitry. Therefore, as discussed previously, to achieve high energy efficiency the column readout circuit must be adequately amortized. Another issue is fitting the interface circuits to the exceedingly fine pitch of these dense arrays.

For multibit activations, the inputs can be analog voltages generated by a DAC. However, for low bitwidths, the DAC overhead can be avoided by serializing [51] or pulsewidth modulating the input activations [52]. Another important consideration is the energy dissipation caused by the resistors alone. Applying 0.5 V across 10 kΩ for 10 ns (typical readout time) corresponds to 250 fJ/MAC. Working with considerably larger resistance values is thus desired, although this often conflicts with maintaining low variability [53].

Many of these tiles might be combined on a single chip to make a full accelerator, as suggested in [54]. All weights remain stationary in this design, while data is processed throughout the fabric with massively parallel computation. At present, the art of designing entire DNN processors with emerging memory devices is still in its infancy. Access to process technology, as well as obstacles with variability, retention, and endurance, are all major concerns.

Despite the error tolerance of neural networks, programming the resistance values appropriately and reading consistent values across operating conditions and device lifespan is a significant difficulty. As a result, the majority of demonstrators are merely subsystems made of relatively small arrays [51, 55]. Furthermore, to simplify the design, some implementations activate just a limited number of rows at a time (e.g., $3 \times 3 = 9$ rows in [51]), resulting in declining performance gains. However, the great amount of continuing research will likely change this, and arrays functioning at near 0.1 fJ/op may one day be possible [56].

1.4. Candidate memory devices

1.4.1. Device requirements

To be useful for neuromorphic computing and DNN inference in particular, NVM devices must fulfill a number of standards that are far more demanding than those for storage memory. Despite this, numerous alternative forms of memory have been proposed as synaptic weights for neuromorphic computing. The device must have at least two consistently identifiable conductance states for inference-only applications. While architectures that perform training require more analog conductance states, seeing as it involves more precise weight tuning, devices with even smaller variability in both read and write operations are mandatory. However, such devices are beyond the scope of this work and will not be discussed further.

Although not strictly necessary, memory components with minimal cycle-to-cycle fluctuation or noise in their stored weight value are sought after in order to realize many bits of weight information in a single device.

Provided that the read non idealities represented in Figure 1.15(a) are minimal, multibit devices can significantly increase the area and energy efficiency of the accelerator. The device must also have a long retention time: any drift in the stored state caused by charge leakage, ion migration, structural relaxation, or another process must be absent or occur over a sufficiently long timescale [17]. The variability in drift rates causes read noise to rise with time, which is an auxiliary but nonetheless significant side consequence of drift. Low conductance is also desirable to prevent parasitic voltage drops along interconnects, allowing for larger VMMs within a crossbar [57]. Cycle-to-cycle noise, incorrect weight programming, and drift can all add up to pose a significant challenge to the signal-to-noise ratio (SNR) of VMM operations in large crossbar arrays. The device should also have high endurance and low write latency so that the weight can be reliably stored during a write operation. The write current and write voltage are also individually important, although less so for inference than for training applications: a low current is desirable to avoid write errors caused by array parasitic resistance, while a low write voltage is required to minimize the CV^2 energy needed to charge the array. Furthermore, regardless of the initial state, the memory device should have a gradual and linear response to the programming pulses used to update the weight, such as the same magnitude of conductance change for the same intensity of the pulse. Generally asymmetric nonlinearity, meaning a distinct response to positive and negative update pulses, degrades accuracy more than symmetric nonlinearity. Some device flaws, notably low accuracy and asymmetry, can be compensated for at the architectural level [17].

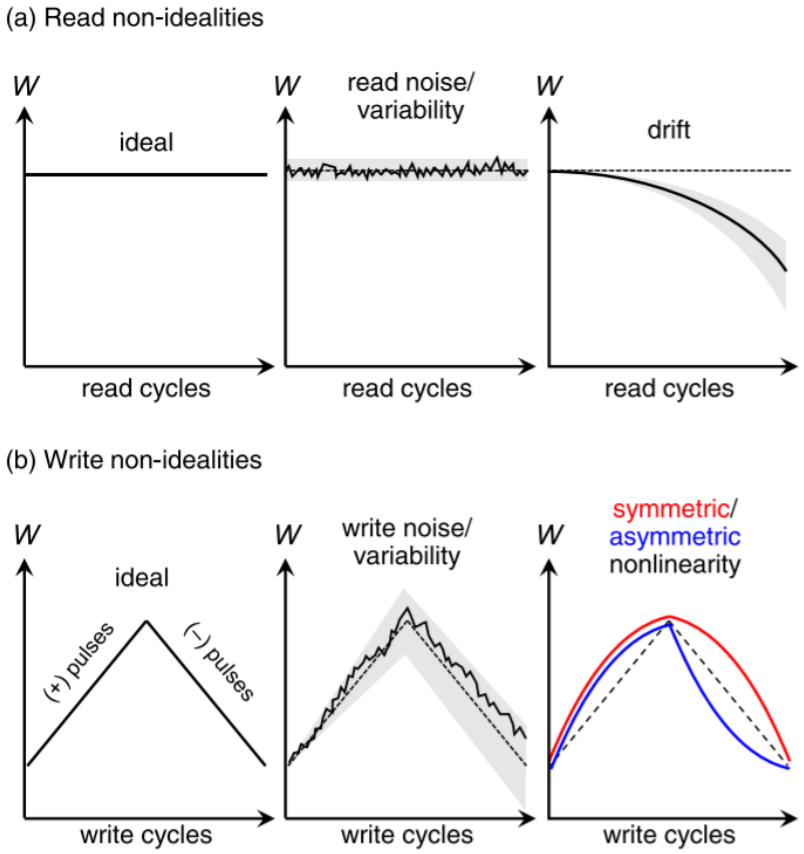


Figure 1.15: ***(a)*** *For reliable and accurate inference, memory devices with low read noise and small drift are desired.* ***(b)*** *The response of a memory device, with and without non-idealities, to a sequence of identical positive update pulses, followed by a sequence of identical negative updates of the same magnitude. Image from [17].*

1.4.2. Emerging non volatile memories

Table 1.3 [17] offers a coarse comparison of the resistive NVM technologies that are suitable for analog synapses, where endurance refers to the number of program/erase (P/E) cycles that can be applied to the device before the stored value becomes unreliable. Device area is expressed by F, defined as the feature size of a technology node.

Below is a brief review of different device alternatives.

Resistive RAM (RRAM): Figure 1.16a shows the RRAM device, consisting of a metallic top electrode, an insulating metal-oxide layer, and a metallic bottom electrode stacked on top of each other. Because of the insulating nature of the oxide layer, the resultant metal-insulator-metal structure has a relatively high resistance. The device is initially formed by a soft breakdown operation, which results in a local

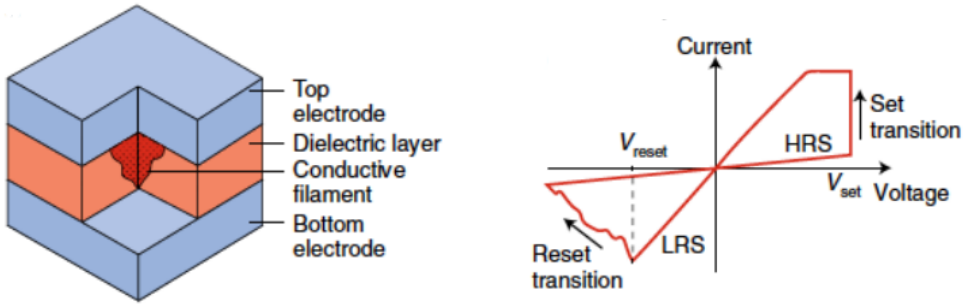
	RRAM	PCM	FGMOS	Redox transistor	FeFET
Maximum resistance	∼1 MΩ	∼1 MΩ	∼1 GΩ	∼10 MΩ [58]	∼1 GΩ
Device area [59]	4 F^2	4 F^2	4 F^2-10 F^2	Large	4 F^2 [60]
Endurance	10^{12} cycles [61]	10^{12} cycles [62]	10^5 cycles	$>10^9$ cycles [58]	10^9 cycles [60]
Programmable resolution	8 bits [63]	5 bits [64]	7 bits [65]	9 bits [66]	5 bits [67]
Write current	1 µA [68]	100 µA [62]	1 pA	10 nA [58]	...
Write speed	ns	ns	ms	ms, µs [58]	ns

Table 1.3: *Comparison of presently available non volatile analog memory technologies.*

modification in the material composition or an increase in the defect concentration, such as oxygen vacancies in the metal oxide. The forming operation typically results in the formation of a conductive filament with a higher conductance than the original insulating layer, resulting in a low resistance state (LRS) of the device. A reset operation can then electrically diminish the conductivity of the conductive filament, resulting in a high resistance state (HRS); the conductivity can also be increased by the set transition in order to recover the LRS state. There also exist uniform-switching RRAM devices, in which the oxide layer alteration spreads over the entire surface rather than just a confined filament region [34].

RRAM is a popular option for crossbar accelerators, as it is generally extremely scalable (with a footprint possibly limited only by the metal pitch), back-end-of-line compatible, and can have a continuously variable conductance, low write energy, low write latency, and high endurance. RRAM devices suffer from high cycle-to-cycle write noise, high device-to-device variability, relatively high read conductance, and random telegraph noise (RTN) in the stored weight value owing to electron trapping, which impacts the HRS more intensely than the LRS [17].

Phase change memory (PCM): in a PCM device (Figure 1.17a) the microstructure of a phase-change material, typically a chalcogenide glass such as $Ge_2Sb_22Te_5$(GST), may be reversibly switched between a crystalline phase and an amorphous phase using the energy delivered by a current pulse. In contrast to the low resistivity of the crystalline phase, the amorphous phase exhibits disorder-induced high resistivity; consequently, the PCM state can be determined using simple voltage/current monitoring. Opposite to the RRAM's filamentary switching mechanism, the PCM depends on the bulk characteristics of the active material, resulting in a broader resistance window: as a matter of fact, different regions of the device volume can be locally amorphous or crystalline, enabling many intermediate conductance levels. A significant Joule heating, on the other hand, is often required to accelerate



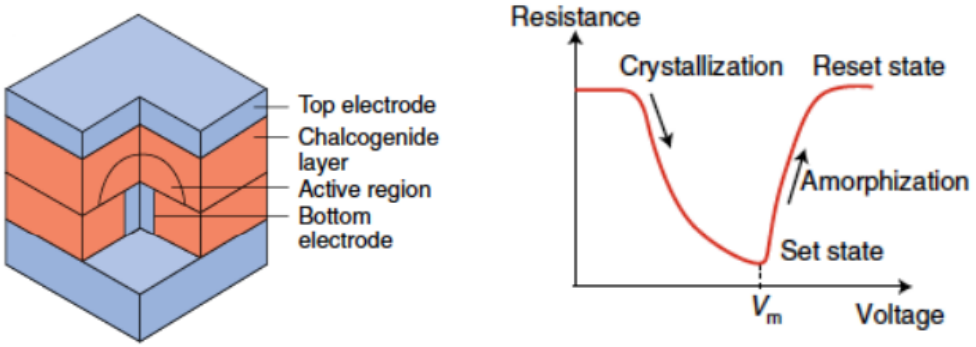
(a) RRAM structure (b) RRAM I-V characteristic

Figure 1.16: *Resistive switching random access memory (RRAM) structure and current-voltage (I-V) characteristic of a bipolar switching device. The set transition from the high resistance state (HRS) to the low resistance state (LRS) occurs at positive voltage due to the formation of a filament shunting the top and bottom electrodes, while the reset transition from LRS to HRS under negative voltage indicates the voltage-induced filament disconnection. [69]*

phase transitions such as melting and crystallization, which result in a rather abrupt phase transition and relatively large currents for programming/erasing the device. Scaling the active zone of the PCM is then required to reduce the programming current [15]. Another serious issue for PCM is resistance drift, in which the device resistance increases with time after programming owing to structural relaxation of the amorphous phase.

Floating gate MOSFET: because of their established manufacturing technology, the floating-gate transistor [70] and the closely related charge-trap memory, which serve as the foundation for commercial flash memory, are also appealing possibilities for neuromorphic computing. In these devices, the analog synaptic weight is stored as charge in an electrically isolated internal electrode or within trap states in an insulator. The charge is added or removed via Fowler–Nordheim tunneling or hot-electron injection through the surrounding oxide. Their advantages include minimal variability, multiple bits per cell, and access to a subthreshold domain of operation with very low conductance, allowing scalability to larger crossbars. However, especially for training applications, a significant obstacle lies in the large voltage (for tunneling), large current (for hot-electron injection), and long write times incurred by the programming pulses.

There are different options for integrating floating-gate devices in a dense array for VMM acceleration, as shown in Figure 1.18. Floating-gate devices can be integrated into a crossbar in the same way as two-terminal variable resistors are (Fig-



(a) PCM structure (b) PCM I-V characteristic

Figure 1.17: *Phase change memory (PCM) structure and resistance change characteristic, showing the resistance measured after a voltage pulse is applied to a PCM device in the amorphous state. The decrease of resistance indicates increasing crystalized volume in the active material, while the increase of resistance above the melting point indicates increasing amorphous volume. [69]*

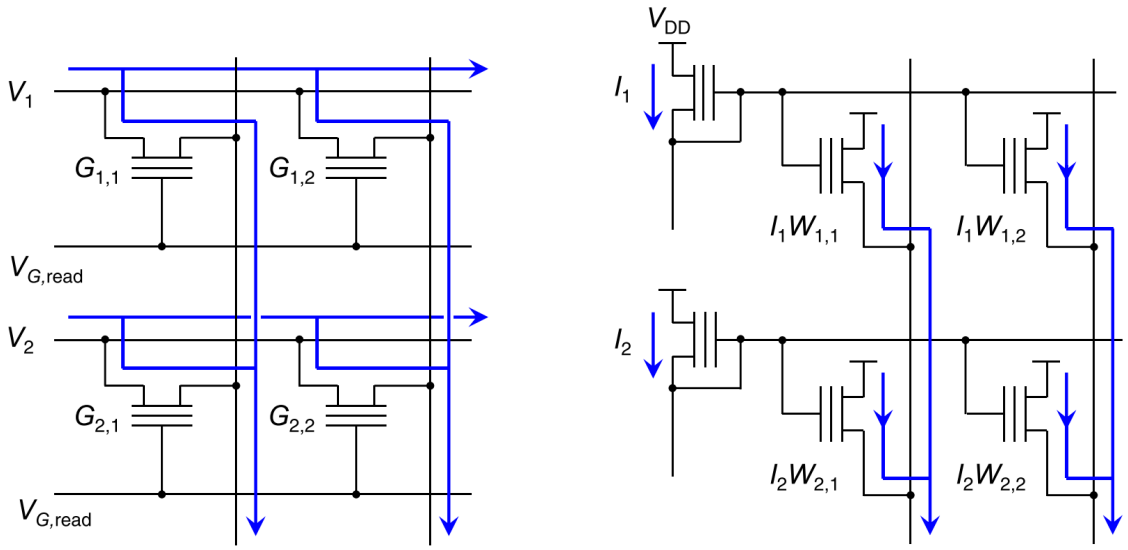
ure 1.18a), with the input voltage applied across the source and drain terminals to perform a VMM through Ohm's law [58, 71]. Depending on the gate voltage provided during read, this layout allows the device to operate in multiple operating regimes—subthreshold, triode, or saturation. Alternatively, by operating the transistors as programmable current mirrors (Figure 1.18b), the inputs to a VMM can be applied via the gate terminal [72–74]. In the subthreshold regime, rather than through Ohm's law, the weight can be implemented as a scaling factor between two currents,

$$W_{ij} = \frac{I_{ij}}{I_i} = exp\left(\kappa \frac{V_{fg,ij} - V_{fg,ref}}{kT/q}\right) \tag{1.4}$$

where I_{ij} is the drain current through the floating-gate synapse, I_i is the input current on the ith row that is injected through a reference transistor, V_{fg} is the voltage on the floating gate, κ is the gate efficiency, and kT/q is the thermal voltage.

The transmission of a gate voltage across the rows requires a minimal amount of input current, reducing the input signal sensitivity to parasitic voltage drops, but the device drain currents must still flow across both the rows and columns. The temperature dependence in (1.4) can be compensated using external circuitry [72].

Redox transistor: electrochemical devices that perform extremely linear and symmetric synapses with a large number of finely spaced states (pivotal for training) have recently been presented [66]. Mobile ions can be injected or withdrawn from the conductive channel of a redox transistor by sending voltage pulses of the proper



(a) Array of FGMOS as programmable resistors

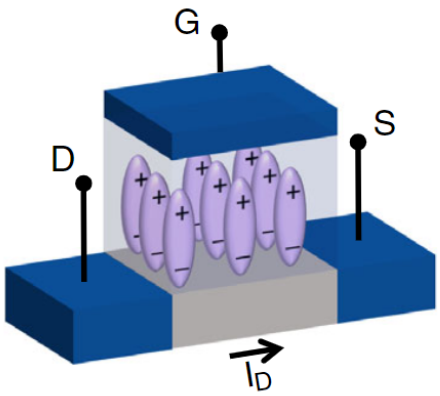
(b) Array of FGMOS as gate-coupled current mirrors

Figure 1.18: ***(a)*** *VMM using a 1T array of floating-gate devices operated as programmable resistors.* ***(b)*** *VMM using a 1T array of floating-gate devices operated as gate-coupled programmable current mirrors. In this case, a voltage input must first be converted into a current; fully current-mode operation can also be used. The floating gate voltage on the row drivers is set to a uniform reference,* $V_{fg,ref}$.

polarity to its gate, modulating its carrier density and hence its conductance.

The advantages of these devices, making them particularly suited for training, are low channel conductance (100 nS and sub-microsecond write times, while the main drawback is the difficulty of integration within CMOS technology due to the added fabrication complexity and the temperature incompatibility of polymer device processing.

Ferroelectric field-effect transistor (FeFET): the FeFET, which permits threshold-voltage programming by integrating a ferroelectric layer into the MOS gate stack, is a non-volatile memory with some similarities to flash memory. The electric field provided by brief voltage pulses with very small current can be used to set the degree of electric polarization of the ferroelectric layer. Training with these devices is complicated by the fact that different ferroelectric domains within the layer require different switching voltages, requiring therefore a complex write scheme. FeFETs are potentially more attractive as VMM engines for inference, but may be practically limited in this case by their comparatively short retention, which arises from charge leakage through the gate and the presence of a parasitic depolarization field.



(a) FeFET structure

Figure 1.19: *FeFET, where the threshold voltage is controlled by the orientation of the ferroelectric dipoles within the gate insulator. [34]*

1.5. Conclusions

To wrap up this report, some conclusions are bound to be drawn. The design of domain-specific hardware for DNNs is an intriguing and fast-paced research topic. As new algorithms, network topologies, and accelerators advance concurrently, the ability to provide hardware flexibility is of major importance, and completely digital architectures are likely to be a fitting solution. However, fully digital DNN accelerators are limited by memory access and data movement overhead, and as the field matures it is realistic to assume that, in some application scenarios, the need for the highest possible efficiency will outweigh the yearning for ample flexibility.

With this expectation in mind, it is worth considering the benefits of mixed-signal compute fabrics for DNN acceleration. While there is no shortage of low-energy analog/mixed-signal demonstrators, the present difficulty lies in combining these macros into a whole accelerator architecture without compromising their efficiency. Data transfer and layer reconfigurability should be a priority in future development on mixed-signal systems, instead of an afterthought left for software simulations. Future DNN accelerators will likely leverage emerging memory technologies to store all DNN weights on moderate-size chips, removing the overhead of external DRAMs. Using in-memory computing may then become a more logical choice in this scenario with dense, weight stationary processing elements and efficient array accumulation using charge or current. However, a significant amount of work remains to be done in terms of understanding the impact of analog nonidealities in DNN computation and how to properly manage them, besides solving challenges such as maintaining architecture flexibility, maximizing efficiency with varying

kernel sizes, managing resolution and mismatch requirements, and designing efficient and dense array interface circuits. Additionally, more research is needed in particular to better understand and mitigate the effects of ADC thermal noise and network weight variability. Developing such an understanding, as with many parts of this fascinating field, requires combining insights from algorithms, network topologies, and circuit design.

Bibliography

[1] Gordon E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965.

[2] David Burg and Jesse H. Ausubel. Moore's law revisited through Intel chip density. *PLOS ONE*, 16(8):1–18, 08 2021.

[3] Robert H. Dennard, Fritz H. Gaensslen, Hwa nien Yu, V. Leo Rideout, Ernest Bassous, Andre, and R. Leblanc. Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE J. Solid-State Circuits*, page 256, 1974.

[4] Christian Märtin. Post-Dennard scaling and the final years of Moore's Law consequences for the evolution of multicore-architectures, 2014.

[5] M. Horowitz, Elad Alon, Dinesh Patil, Sam Naffziger, Rajesh Kumar, and K. Bernstein. Scaling, power, and the future of CMOS, 01 2006.

[6] Hadi Esmaeilzadeh, Emily Blem, Renee St. Amant, Karthikeyan Sankaralingam, and Doug Burger. Dark silicon and the end of multicore scaling. *SIGARCH Comput. Archit. News*, 39(3):365–376, jun 2011.

[7] Samuel H. Fuller and Lynette I. Millett. Computing performance: Game over or next level? *Computer*, 44:31–38, 2011.

[8] János Végh. How Amdahl's law limits the performance of large artificial neural networks. *Brain Informatics*, 6, 2019.

[9] Norman P. Jouppi, Cliff Young, Nishant Patil, and David Patterson. A domain-specific architecture for deep neural networks. *Commun. ACM*, 61(9):50–59, aug 2018.

[10] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. Processing data where it makes sense: Enabling in-memory computation. *CoRR*, abs/1903.03988, 2019.

[11] Mark Horowitz. Computing's energy problem (and what we can do about it).

2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), pages 10–14, 2014.

[12] Don Monroe. Neuromorphic computing gets ready for the (really) big time. *Commun. ACM*, 57(6):13–15, jun 2014.

[13] Giacomo Indiveri and Shih-Chii Liu. Memory and information processing in neuromorphic systems. *Proceedings of the IEEE*, 103(8):1379–1397, 2015.

[14] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, Bernard Brezzo, Ivan Vo, Steven K. Esser, Rathinakumar Appuswamy, Brian Taba, Arnon Amir, Myron D. Flickner, William P. Risk, Rajit Manohar, and Dharmendra S. Modha. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[15] Abu Sebastian, Manuel Gallo, Riduan Khaddam-Aljameh, and Evangelos Eleftheriou. Memory devices and applications for in-memory computing. *Nature Nanotechnology*, 15, 03 2020.

[16] Mohammed A. Zidan, John Paul Strachan, and Wei D. Lu. The future of electronics based on memristive systems. *Nature Electronics*, 1(1):22–29, Jan 2018.

[17] T. Patrick Xiao, Christopher H. Bennett, Ben Feinberg, Sapan Agarwal, and Matthew J. Marinella. Analog architectures for neural network acceleration based on non-volatile memory. *Applied Physics Reviews*, 7(3):031301, 2020.

[18] Robert A. Nawrocki, Richard M. Voyles, and Sean E. Shaheen. A mini review of neuromorphic architectures and implementations. *IEEE Transactions on Electron Devices*, 63:3819–3829, 2016.

[19] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A survey of neuromorphic computing and neural networks in hardware. *CoRR*, abs/1705.06963, 2017.

[20] Boris Murmann. Mixed-signal computing for deep neural network inference. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29:3–13, 2021.

[21] Lei Deng, Guoqi Li, Song Han, L.P. Shi, and Yuan Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, PP:1–48, 03 2020.

[22] Xuan Yang, Mingyu Gao, Qiaoyi Liu, Jeff Setter, Jing Pu, Ankita Nayak, Steven

Bell, Kaidi Cao, Heonjae Ha, Priyanka Raina, and et al. Interstellar: Using Halide's scheduling language to analyze DNN accelerators. *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, Mar 2020.

[23] Yannan N. Wu, Joel S. Emer, and Vivienne Sze. Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs. In *IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2019.

[24] K. Guo, W. Li, K. Zhong, Z. Zhu, S. Zeng, S. Han, Y. Xie, P. Debacker, M. Verhelst, and Y. Wang. Neural network accelerator comparison. Available: https://nicsefc.ee.tsinghua.edu.cn/projects/neural-network-accelerator/, 2021.

[25] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization, 2020.

[26] Daniel Bankman, Lita Yang, Bert Moons, Marian Verhelst, and Boris Murmann. An always-on 3.8μj/86% CIFAR-10 mixed-signal binary CNN processor with all memory on chip in 28nm CMOS. *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, pages 222–224, 2018.

[27] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research).

[28] Hongyang Jia, Hossein Valavi, Yinqi Tang, Jintao Zhang, and Naveen Verma. A programmable heterogeneous microprocessor based on bit-scalable in-memory computing. *IEEE Journal of Solid-State Circuits*, PP:1–1, 04 2020.

[29] Qing Dong, Mahmut E. Sinangil, Burak Erbagci, Dar Sun, Win-San Khwa, Hung-Jen Liao, Yih-Chau Wang, and Jonathan Chang. A 351TOPS/W and 372.4GOPS compute-in-memory SRAM macro in 7nm FinFET CMOS for machine-learning applications. *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*, pages 242–244, 2020.

[30] Jinshan Yue, Zhe Yuan, Xiaoyu Feng, Yifan He, Zhixiao Zhang, Xin Si, Ruhui Liu, Meng-Fan Chang, Xueqing Li, Huazhong Yang, and Yongpan Liu. 14.3 a 65nm computing-in-memory-based CNN processor with 2.9-to-35.8 TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse. In *2020 IEEE International Solid- State Circuits Conference, ISSCC 2020, San Francisco, CA, USA, February 16-20, 2020*, pages 234–236. IEEE, 2020.

[31] Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, and Joel Emer. Efficient processing of deep neural networks: A tutorial and survey, 2017.

[32] Albert Reuther, Peter Michaleas, Michael Jones, Vijay Gadepally, Siddharth Samsi, and Jeremy Kepner. Survey and benchmarking of machine learning accelerators. *CoRR*, abs/1908.11348, 2019.

[33] Eric S. Chung, Jeremy Fowers, Kalin Ovtcharov, Michael Papamichael, Adrian M. Caulfield, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, Michael Haselman, Maleen Abeydeera, Logan Adams, Hari Angepat, Christian Boehn, Derek Chiou, Oren Firestein, Alessandro Forin, Kang Su Gatlin, Mahdi Ghandi, Stephen Heil, Kyle Holohan, Ahmad El Husseini, Tamás Juhász, Kara Kagi, Ratna Kovvuri, Sitaram Lanka, Friedel van Megen, Dima Mukhortov, Prerak Patel, Brandon Perez, Amanda Rapsang, Steven K. Reinhardt, Bita Darvish Rouhani, Adam Sapek, Raja Seera, Sangeetha Shekar, Balaji Sridharan, Gabriel Weisz, Lisa Woods, Phillip Yi Xiao, Dan Zhang, Ritchie Zhao, and Doug Burger. Serving DNNs in real time at datacenter scale with Project Brainwave. *IEEE Micro*, 38:8–20, 2018.

[34] D. Ielmini and Giacomo Pedretti. Device and circuit architectures for in-memory computing. *Advanced Intelligent Systems*, 2, 05 2020.

[35] Brian Zimmer, Priyanka Raina, Stephen Tell, Yanqing Zhang, William Dally, Joel Emer, C. Gray, Stephen Keckler, Brucek Khailany, Rangharajan Venkatesan, Yakun Shao, Jason Clemons, Matthew Fojtik, Nan Jiang, Ben Keller, Alicia Klinefelter, and Nathaniel Pinckney. A 0.32-128 TOPS, scalable multi-chip-module- based deep neural network inference accelerator with ground-referenced signaling in 16 nm. *IEEE Journal of Solid-State Circuits*, PP:1–13, 01 2020.

[36] Vasisht Duddu, D Vijay Rao, and Valentina Balas. Towards enhancing fault tolerance in neural networks. In *MobiQuitous 2020 - 17th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, MobiQuitous '20, page 59–68, New York, NY, USA, 2020. Association for Computing Machinery.

[37] César Torres-Huitzil and Bernard Girau. Fault and error tolerance in neural networks: A review. *IEEE Access*, 5:17322–17341, 2017.

[38] E.A. Vittoz. Future of analog in the VLSI environment. pages 1372 – 1375 vol.2, 06 1990.

[39] Rahul Sarpeshkar. Analog versus digital: extrapolating from electronics to neurobiology. *Neural computation*, 10(7):1601–1638, 1998.

[40] Boris Murmann, Daniel Bankman, Elaina Chai, Daisuke Miyashita, and Lita Yang. Mixed-signal circuits for embedded machine-learning applications. In *2015 49th Asilomar conference on signals, systems and computers*, pages 1341–1345. IEEE, 2015.

[41] Shih-Chii Liu, John Paul Strachan, and Arindam Basu. Prospects for analog circuits in deep networks, 2021.

[42] Xuezhu Zhang, Jian Zhou, Simon R Cherry, Ramsey D Badawi, and Jinyi Qi. Quantitative image reconstruction for total-body PET imaging using the 2-meter long EXPLORER scanner. *Physics in Medicine and Biology*, 62(6):2465–2485, feb 2017.

[43] Matthieu Courbariaux and Yoshua Bengio. BinaryNet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830, 2016.

[44] Vaibhav Tripathi and Boris Murmann. Mismatch characterization of small metal fringe capacitors. volume 61, pages 1–4, 09 2013.

[45] Hesham Omran, Hamzah Alahmadi, and Khaled Salama. Matching properties of femtofarad and sub-femtofarad MOM capacitors. *IEEE Transactions on Circuits and Systems I: Regular Papers*, pages 1–10, 04 2016.

[46] Hossein Valavi, Peter Ramadge, Eric Nestler, and Naveen Verma. A 64-tile 2.4-Mb in-memory-computing CNN accelerator employing charge-domain compute. *IEEE Journal of Solid-State Circuits*, PP:1–11, 03 2019.

[47] Angad Rekhi, Brian Zimmer, Nikola Nedovic, Ningxi Liu, Rangharajan Venkatesan, Miaorong Wang, Brucek Khailany, William Dally, and C. Gray. Analog/mixed-signal hardware error modeling for deep learning inference. pages 1–6, 06 2019.

[48] Bonan Zhang, Lung Yen Chen, and Naveen Verma. Stochastic data-driven hardware resilience to efficiently train inference models for stochastic hardware implementations. In *2019 IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2019 - Proceedings*, ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing - Proceedings, pages 1388–1392, United States, May 2019. Institute of Electrical and Electronics Engineers Inc. 44th IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2019 ; Conference date: 12-05-2019 Through 17-05-2019.

[49] Phil C. Knag, Gregory K. Chen, Huseyin Ekin Sumbul, Raghavan Kumar, Steven K. Hsu, Amit Agarwal, Monodeep Kar, Seongjong Kim, Mark Anders, Himanshu Kaul,

and Ram K. Krishnamurthy. A 617-TOPS/W all-digital binary neural network accelerator in 10-nm FinFET CMOS. *IEEE Journal of Solid-State Circuits*, 56:1082–1092, 2021.

[50] Qi Liu, Bin Gao, Peng Yao, Dong Wu, Junren Chen, Yachuan Pang, Wenqiang Zhang, Yan Liao, Cheng-Xin Xue, Wei-Hao Chen, Jianshi Tang, Yu Wang, Meng-Fan Chang, He Qian, and Huaqiang Wu. A fully integrated analog ReRAM based 78.4TOPS/W compute-in-memory chip with fully parallel MAC computing. pages 500–502, 02 2020.

[51] Cheng-Xin Xue, Wei-Hao Chen, Je-Syu Liu, Jia-Fang Li, Wei-Yu Lin, Wei-En Lin, Jing-Hong Wang, Wei-Chen Wei, Chang Ting-Wei, Tung-Cheng Chang, Tsung-Yuan Huang, Hui-Yao Kao, Shih-Ying Wei, Yen-Cheng Chiu, Chun-Ying Lee, Chung-Chuan Lo, Frederick Chen, Chorng-Jung Lin, Ren-Shuo Liu, and Meng-Fan Chang. 24.1 a 1Mb multibit ReRAM computing-in-memory macro with 14.6ns parallel MAC computing time for CNN based AI edge processors. pages 388–390, 02 2019.

[52] D. Bankman, J. Messner, A. Gural, and B. Murmann. Rram-based in-memory computing for embedded deep neural networks. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, pages 1511–1515, 2019.

[53] Binh Q. Le, Alessandro Grossi, Elisa Vianello, Tony F. Wu, Giusy Lama, Edith Beigné, H. S. Philip Wong, and Subhasish Mitra. Resistive RAM with multiple bits per cell: Array-level demonstration of 3 bits per cell. *IEEE Transactions on Electron Devices*, 66:641–646, 2019.

[54] Martino Dazzi, Abu Sebastian, Pier Andrea Francese, Thomas Parnell, Luca Benini, and Evangelos Eleftheriou. 5 Parallel Prism: A topology for pipelined implementations of convolutional neural networks using computational memory, 2019.

[55] Shihui Yin, Xiaoyu Sun, Shimeng Yu, and Jae-Sun Seo. High-throughput in-memory computing for binary deep neural networks with monolithically integrated RRAM and 90-nm CMOS. *IEEE Transactions on Electron Devices*, 67(10):4185–4192, Oct 2020.

[56] Stefan Cosemans, Bram-Ernst Verhoef, J. Doevenspeck, I. Papistas, F. Catthoor, Peter Debacker, Arindam Mallik, and D. Verkest. Towards 10000TOPS/W DNN inference with analog in-memory computing – a circuit blueprint, device options and requirements. pages 22.2.1–22.2.4, 12 2019.

[57] Chuan-Sen Yang, Da-Shan Shang, Nan Liu, Elliot J. Fuller, Sapan Agrawal, A. Alec Talin, Yong-Qing Li, Bao-Gen Shen, and Young Sun. All-solid-state synaptic tran-

sistor with ultralow conductance for neuromorphic computing. *Advanced Functional Materials*, 28(42):1804170, 2018.

[58] Elliot J. Fuller, Scott T. Keene, Armantas Melianas, Zhongrui Wang, Sapan Agarwal, Yiyang Li, Yaakov Tuchman, Conrad D. James, Matthew J. Marinella, J. Joshua Yang, Alberto Salleo, and A. Alec Talin. Parallel programming of an ionic floating-gate memory array for scalable neuromorphic computing. *Science*, 364(6440):570–574, 2019.

[59] An Chen. A review of emerging non-volatile memory (NVM) technologies and applications. *Solid-State Electronics*, 125:25–38, 2016. Extended papers selected from ESSDERC 2015.

[60] Yun Long, Taesik Na, Prakshi Rastogi, Karthik Rao, Asif Islam Khan, Sudhakar Yalamanchili, and Saibal Mukhopadhyay. A ferroelectric FET based power-efficient architecture for data-intensive computing. In *Proceedings of the International Conference on Computer-Aided Design*, ICCAD '18, New York, NY, USA, 2018. Association for Computing Machinery.

[61] Chung-Wei Hsu, I-Ting Wang, Chun-Li Lo, Ming-Chung Chiang, Wen-Yueh Jang, Chen-Hsi Lin, and Tuo-Hung Hou. Self-rectifying bipolar TaOx/TiO2 RRAM with superior endurance over 1012 cycles for 3D high-density storage-class memory. pages T166–T167, 01 2013.

[62] Scott W. Fong, Christopher M. Neumann, and H. S. Philip Wong. Phase-change memory—towards a storage-class memory. *IEEE Transactions on Electron Devices*, 64:4374–4385, 2017.

[63] Emmanuelle Merced-Grafals, Noraica Dávila, Ning Ge, Stan Williams, and John William Strachan. Repeatable, accurate, and high speed multi-level programming of memristor 1T1R arrays for power efficient analog computing applications. *Nanotechnology*, 27:365202, 08 2016.

[64] Abu Sebastian, Manuel Le Gallo, and Evangelos Eleftheriou. Computational phase-change memory: beyond von Neumann computing. *Journal of Physics D: Applied Physics*, 52(44):443002, aug 2019.

[65] F. Merrikh Bayat, X. Guo, H. A. Ommani, N. Do, K. K. Likharev, and D. B. Strukov. Redesigning commercial Floating-Gate memory for analog computing applications, 2014.

[66] Yoeri van de Burgt, Ewout Lubberman, Elliot J. Fuller, Scott T. Keene, Grégorio C.

Faria, Sapan Agarwal, Matthew J. Marinella, A. Alec Talin, and Alberto Salleo. A non-volatile organic electrochemical device as a low-voltage artificial synapse for neuromorphic computing. *Nature Materials*, 16(4):414–418, Apr 2017.

[67] Maximilian Lederer, Thomas Kämpfe, T. Ali, Franz Müller, Ricardo Revello Olivo, Raik Hoffmann, Nellie Laleni, and K. Seidel. Ferroelectric Field Effect Transistors as a synapse for neuromorphic application. *IEEE Transactions on Electron Devices*, 68:2295 – 2300, 03 2021.

[68] Yi Wu, Byoungil Lee, and H.-S. Philip Wong. Al2O3-based RRAM using atomic layer deposition (ALD) with 1-μa reset current. *Electron Device Letters, IEEE*, 31:1449 – 1451, 01 2011.

[69] Daniele Ielmini and H.-S. Philip Wong. In-memory computing with resistive switching devices. *Nature Electronics*, 1:333–343, 2018.

[70] Paul Hasler, Chris Diorio, Bradley A. Minch, and Carver Mead. Single transistor learning synapses. In *Proceedings of the 7th International Conference on Neural Information Processing Systems*, NIPS'94, page 817–824, Cambridge, MA, USA, 1994. MIT Press.

[71] Hyun-Seok Choi, Yu Jeong Park, Jong-Ho Lee, and Yoon Kim. 3-D synapse array architecture based on charge-trap flash memory for neuromorphic application. *Electronics*, 9(1), 2020.

[72] Craig Schlottmann and Paul E. Hasler. A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, 1:403–411, 2011.

[73] Ravi Chawla, Abhishek Bandyopadhyay, Venkatesh Srinivasan, and Paul E. Hasler. A 531 nW/MHz, 128/spl times/32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity. *Proceedings of the IEEE 2004 Custom Integrated Circuits Conference (IEEE Cat. No.04CH37571)*, pages 651–654, 2004.

[74] Laura Fick, David Blaauw, Dennis Sylvester, Skylar Skrzyniarz, M. Parikh, and David Fick. Analog in-memory subthreshold deep neural network accelerator. *2017 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–4, 2017.