Journal of
Low Power Electronics and Applications

MDPI

Article

The Potential of SoC FPAAs for Emerging Ultra-Low-Power Machine Learning

Jennifer Hasler

Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA 30332, USA; jennifer.hasler@ece.gatech.edu; Tel.: +1-404-894-2944; Fax: +1-404-894-4641

Abstract: Large-scale field-programmable analog arrays (FPAA) have the potential to handle machine inference and learning applications with significantly low energy requirements, potentially alleviating the high cost of these processes today, even in cloud-based systems. FPAA devices enable embedded machine learning, one form of physical mixed-signal computing, enabling machine learning and inference on low-power embedded platforms, particularly edge platforms. This discussion reviews the current capabilities of large-scale field-programmable analog arrays (FPAA), as well as considering the future potential of these SoC FPAA devices, including questions that enable ubiquitous use of FPAA devices similar to FPGA devices. Today's FPAA devices include integrated analog and digital fabric, as well as specialized processors and infrastructure, becoming a platform of mixed-signal development and analog-enabled computing. We address and show that next-generation FPAAs can handle the required load of 10,000–10,000,000,000 PMAC, required for present and future large fielded applications, at orders of magnitude of lower energy levels than those expected by current technology, motivating the need to develop these new generations of FPAA devices.

Keywords: machine learning; FPAA; analog computing

Citation: Hasler, J. The Potential of SoC FPAAs for Emerging Ultra-Low-Power Machine Learning. *J. Low Power Electron. Appl.* **2022**, *12*, 33. https://doi.org/10.3390/jlpea12020033

Academic Editor: Andrea Acquaviva

Received: 31 December 2021
Accepted: 11 March 2022
Published: 6 June 2022

1. Motivating Ultra-Low-Power Embedded Machine Learning

Large-scale field-programmable analog arrays (FPAA) show significant potential for mixed-signal computing [1], including embedded machine learning [2–4], machine learning, and inference on low-power embedded platforms. Although cloud-centric machine learning will be used going forward, the energy constraints at the edge device requires significantly more processing to be computed locally, rather than paying the high energy costs of transmitting data off the device. The energy required for cloud computations is not negligible, seen in the fact that the costs for the production of cloud-based machine learning techniques are becoming a significant fraction of the USA's energy budget (e.g., [5]).

Physical computing techniques enable local, edge-embedded computation, through significantly improved energy efficiency. Mead originally proposed that analog computation would have 1000× computational energy efficiency over digital approaches [6], a factor experimentally demonstrated in custom Si in 2004 [7], and repeatably demonstrated ever since, including multiple FPAA demonstrations (e.g., [1,8]). As a physical neuromorphically engineered Si Cortex could be possible in less than 100 W [9], physical computing devices can eventually go beyond current cloud-based machine learning approaches [5].

FPAA devices enable embedded Machine inference and learning [2–4], that includes neural networks (NN) and neurally inspired datapaths using analog elements and signals integrated with potential logic and mixed-signal-enabled routing (Figure 1). Analog implementations of NN have roots from the earliest demonstrations (e.g., [10,11]). FPAAs include analog elements and signals integrated with potential logic and mixed-signal-enabled routing (Figure 1), enabling full *end-to-end*, sensor to decision—computation that includes machine learning. FPAA devices, such as the SoC FPAAs, integrate analog and digital

components, where FPAA devices that integrate analog and digital elements are a superset of FPGAs by integrating digital elements.

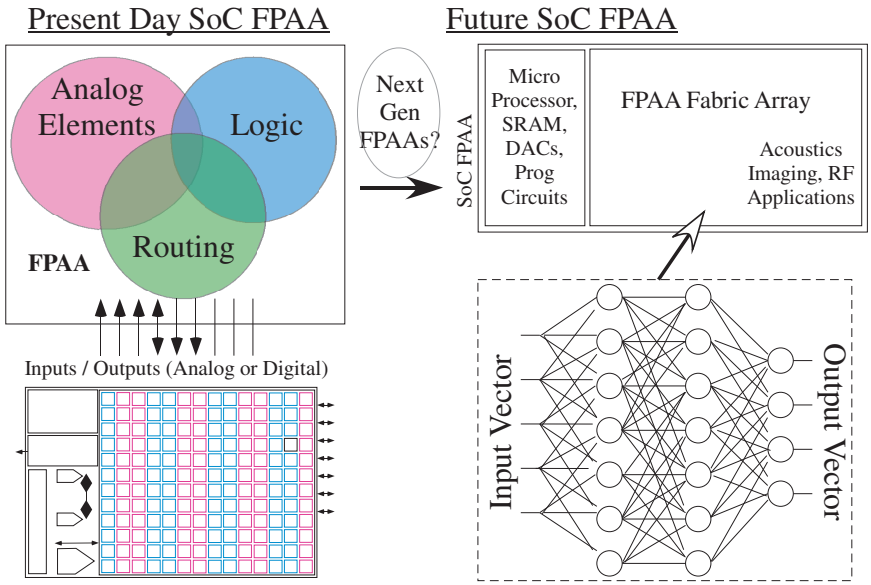


Figure 1. Current large-scale field-programmable analog arrays (FPAAs) extend the concepts of FPGA devices to include analog elements and signals integrated with potential logic and mixed-signal-enabled routing. Current SoC FPAA devices include integrated analog and digital fabric as well as specialized processors and infrastructure, becoming a platform of mixed-signal development as well as analog-enabled computing. *The fundamental question is the following: What is the potential of future FPAA devices in machine learning applications?* Future FPAA devices seem to offer a promise of ubiquitous reconfigurable devices for ultra-low-power machine learning that can directly solve the enormous energy requirements of current fielded machine learning applications, as well as enable what is typically considered cloud-based machine learning in edge devices.

SoC FPAAs enable an analog-enabled mixed-signal computing platform that enables wide user development of the emerging analog computing techniques (e.g., [12]). Many of these techniques were first shown in FPAA devices, or in custom ICs that were on the development path into FPAA devices [1]. Large-scale field-programmable analog arrays were explicitly defined (e.g., [13]) as reconfigurable, mixed-signal devices to be used for computation, rather than *glue* logic devices (e.g., digital CPLDs), typical of early analog reconfigurable approaches (e.g., [14]), while staying consistent with development history.

End-to-end computation requires analog input and computation for physical implementations, and communication and data conversion is part of the system cost (Figure 2). An FPGA can handle analog signal inputs and outputs with the addition of sufficient data converters (Figure 2). FPGA devices are constructed to enable a wide range of digital applications, and have a number of specialized blocks to optimize particular applications [15–23]. Xlinix's Zync FPGA [24], a recent FPGA, includes high-speed (up to 10 GSPS) 14-bit DAC and (up to 4 GSPS) 14-bit ADCs on-chip with 6 ARM µP, as well as digital fabric and interfaces. These ICs show a need for integrated analog components in configurable structures in reducing the overall system complexity and system throughput. The Zync FPGA family, fabricated in 16–10 nm CMOS nodes, can support on-chip RF capability. The overall high energy, area, and complexity costs for the digital computation (e.g., [8] vs. [25]), static FPGA power, and data converters will overwhelm many applications.

FPGAs are not low power when looking at solutions requiring 10–100 mW of power. Most FPGAs require 100 s of mW simply to power up the SRAM elements holding the programming variables. Commercial flash-based FPGAs significantly decrease the starting power requirements (e.g., 7–10 mW standby power [26,27]) while enabling 350–500 MHz signals and 70 mW 5 G SERDES, although the power requirements are too high for lower power systems of 10–20 mW and below. Analog computation (FPAAs) solves these issues energy and area efficient analog computation, as well as requiring far fewer data converters.

Analog co-processors for a digital computation with a bank of data converters create a huge infrastructure that nearly eliminates the benefits of analog processing. Current FPGA-based solutions (Figure 2a) are used because of the lack of commercially available FPAAs, as well as the lack of available FPAA engineering experience.

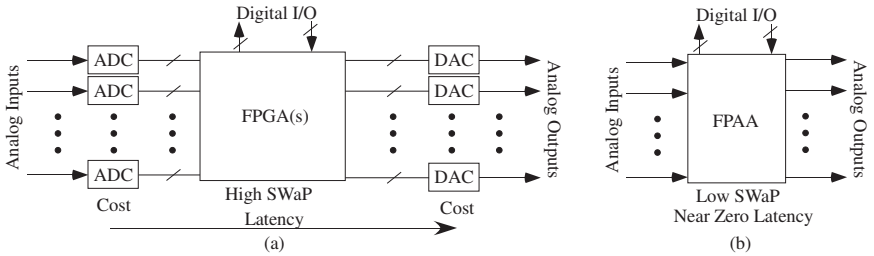


Figure 2. Both FPGA and FPAA are capable of *end-to-end*, analog input and analog output machine learning with different tradeoffs. (**a**) An FPGA or multiple FPGAs can be used for a range of analog approaches assuming there are sufficient ADCs and DACs for the resulting analog signals. In addition to the high energy and complexity costs of these data converters, the FPGA has some latency for its digital computation and is constrained by its static and dynamic power requirements. (**b**) An FPAA device can typically interface to the incoming analog signals. Where necessary (e.g., RF), input signals can be adjusted to the IC's power and supply voltage levels. The FPAA device uses analog techniques where possible in a near-zero latency path utilizing digital control.

Programmable and configurable FPAA devices enable end-to-end embedded machine learning applications (Figures 1 and 3). A machine learning algorithm that ignores other application areas, such as translating sensor data as an input into the network, often shows incremental or negligible system improvements. *End-to-end* embedded machine learning eliminates requiring a bank of FPGAs and interface chips in front of a low-power classifier chip so that the wins from the physical system still provide advantages for the overall system concept. Configurability is essential to have a limited number of commercially relevant ICs to handle a wide application space, particularly the wide range of front-end computations for the end-to-end computation, as well as efficiently implementing local and sparse weight matrices. Building end-to-end architectures typically reduces the amount of raw computations required, further empowering these physical computing approaches.

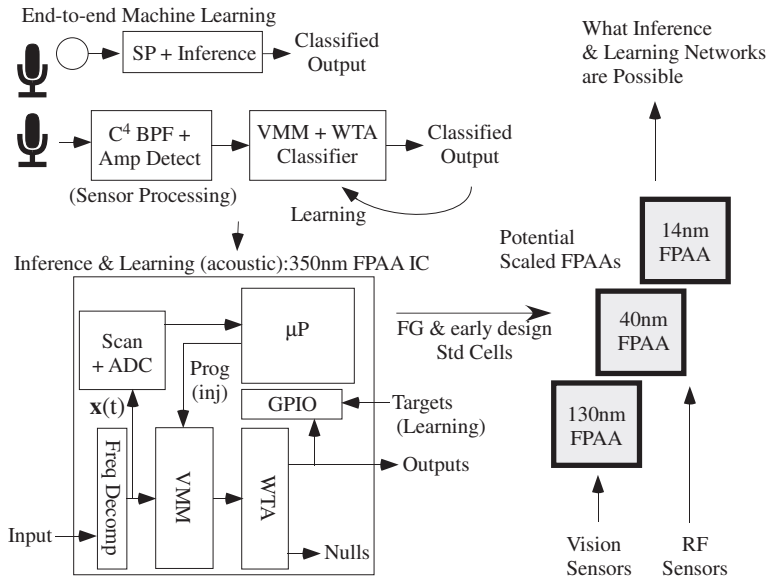


Figure 3. SoC FPAAs (350 nm CMOS) have demonstrated end-to-end machine learning (inference and learning) applications, from microphones to classified result, even without the FPAA designed for machine learning. The question becomes the following: What are the additional capabilities of scaled FPAA devices for end-to-end classifiers for acoustic applications as well as new enabled applications that utilize vision and RF sensors.

The questions are whether FPAA approaches can transform the required load of 10,000 to 10,000,000,000 PMAC required for future fielded applications (e.g., [5]), and do these FPAAs computations perform at far lower energy levels? A positive response motivates the engineering efforts to develop these new generations of FPAA devices. The projected scaling of FPAA devices enables us to directly address these questions, and that, in turn, requires further analysis into the capabilities and flexibilities of FPAA architectures.

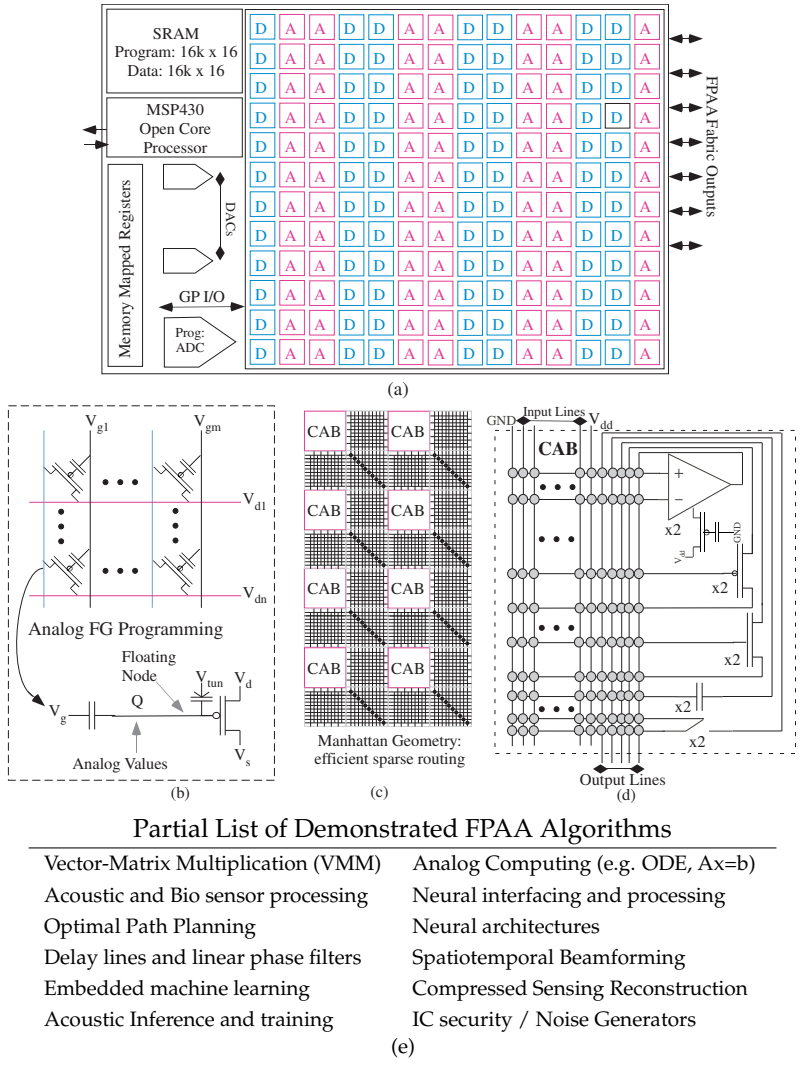
This discussion works through the the opportunities and questions towards building a ubiquitous supply of large-scale field-programmable analog arrays (FPAA), particularly SoC FPAA-type devices [28] for machine learning applications with ultra-low energy requirements. The discussion starts by providing an overview of the FPAA capabilities that enables the wider application for *end-to-end* machine learning (Section 2). Understanding the architectural and granularity tradeoffs in FPAA architectures (Section 3) enables predicting the future capabilities of these devices (Section 4).

2. Configurable Technology, Architecture, and Capabilities

SoC FPAAs enable a mixed-signal *end-to-end* embedded machine platform as they include analog elements and signals integrated with potential logic and mixed-signal-enabled routing (Figure 4). *End-to-end* acoustic embedded learning and classification have been demonstrated at 20–30 μW on command–word recognition, as well as on the full Nzero database [2–4]. FPAA devices have experimentally demonstrated a wide range of computations (Figure 4) that include computations to set up machine learning, as well as the machine learning inference and learning, sometimes at small scale, given the component constraints of a 350 nm CMOS SoC FPAA [28]. Floating gate (FG) techniques enable programmability, having precise parameters (e.g., 14-bit accuracy [29]), in standard CMOS, as well as configurability, through long-term retention of FG charge (0–100 μV over 10 years [30]). SoC FPAAs enable a wide user development of the emerging analog computing techniques (e.g., [12]).

FPAA tools empower a wide application ecosystem through systematic analog design [1,31,32] to enable an engineering team to rapidly develop new embedded machine learning components. These tools give the user the ability to create, model, and simulate analog and digital designs. This early tool enables the development of an FPAA toolset that can start with high-level definitions and automatically generate targeted hardware where the user has the ability to optimize the process at each level. These analog and mixed-signal tools are expanding to analog synthesis using standard cells for custom ICs (e.g., [33]).

FPAA devices can be the solution for analog and mixed-signal security and component obsolescence [34], just as FPGAs solve security and obsolescence issues. Multiple digital techniques can verify an FPGA, allowing for secure and confident FPGA programming for a particular application. An FPAA device can be a completely generic and known device that can be completely verified in a safe location [34], where the secret sauce for the technology can be programmed on the device also in a safe location (Figure 5). The resulting FPAA device layout says nearly nothing about the programmed function, similar to FPGA devices. FPAA devices can directly and discretely map secure functions, such as unique functions and physically unclonable functions (PUF), directly into the FPAA fabric [34]. In general, the more specific the resulting solution, and the fewer levels of software stacks, the fewer security holes; the nonvolatile programming of an FPAA device minimizes the number of mixed-signal security concerns. The use of nonvolative memory (e.g., FG) eliminates SRAM loading vulnerabilities, where analog values are difficult to measure without significantly distorting the measurements where digital computing can be analog encoded, and where low-power circuits provide unique challenges for external measurements [34]. FPAAs can replicate similar analog circuit elements or a combination of analog circuit elements to achieve similar linear and nonlinear dynamics seen in older custom or configurable devices, including some of the unintended dynamics, eliminating the obsolescence issues. Mapping discrete-component (e.g., BJT vs. FET) analog music synthesis to FPAA devices demonstrates translating intended and unintended dynamics [35].



Partial List of Demonstrated FPAA Algorithms

Vector-Matrix Multiplication (VMM)	Analog Computing (e.g. ODE, Ax=b)
Acoustic and Bio sensor processing	Neural interfacing and processing
Optimal Path Planning	Neural architectures
Delay lines and linear phase filters	Spatiotemporal Beamforming
Embedded machine learning	Compressed Sensing Reconstruction
Acoustic Inference and training	IC security / Noise Generators

(e)

Figure 4. FPAAs enable a range of computations in a single programmable and reconfigurable IC. (**a**) Block diagram of a SoC FPAA device, that includes analog (**A**: CAB) and digital (**D**) elements in the routing fabric integrated with a µP and other mixed-signal components. (**b**) The FPAA approach is enabled through analog programmable floating gate (FG) devices, providing non-volatile storage (e.g., <100 µV in 10 years) and routing as well as computing directly through the routing crossbars. (**c**) SoC FPAA architectures use a Manhattan architecture to route between components in a computational analog block (CAB). (**d**) A typical CAB utilizes a number of components that may utilize FG parameters, as well as FG switches that can be used as part of the computation. (**e**) Current FPAA devices are capable of a diverse set of demonstrated computations.

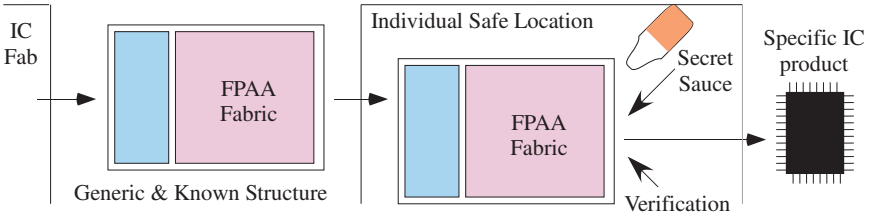
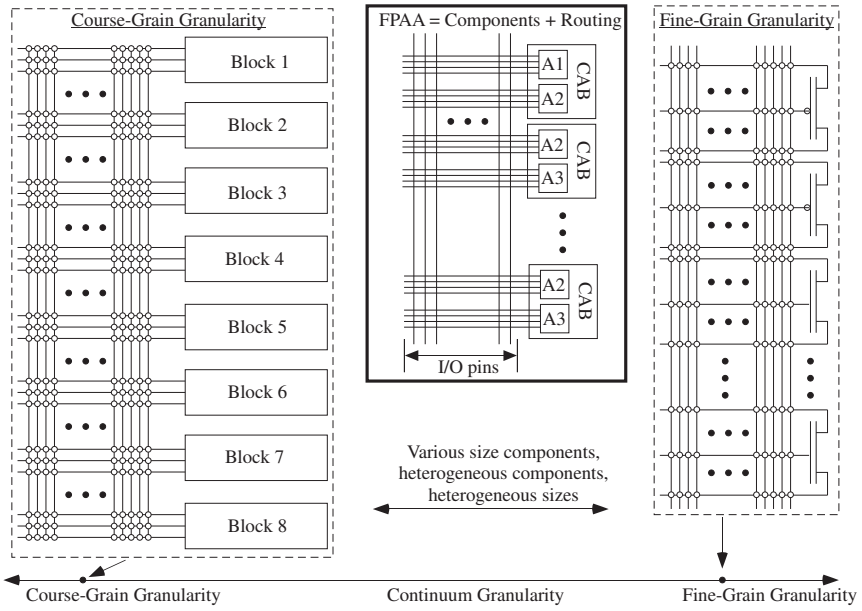


Figure 5. An FPAA device can be a completely generic and known device, completely verified in a safe location, and have the technology secret sauce be programmed in a safe location. The output product is a custom chip due to the nonvolatile device programming.

3. Granularity for End-To-End Machine Learning: Flexibility vs. Switch Cost

To predict FPAA devices scaling to advanced process nodes, one needs to understand how the configurable fabric might scale to larger technology nodes. An effective config-

urable fabric for embedded machine learning applications empowers the user's creativity through flexible opportunities while minimizing the added cost for that flexibility, particularly with increasing application size. Flexibility (ϕ) enables more computations in a single architecture, where Φ quantifies the possible combinations available. Flexibility affects the types, sparsity, and energy efficiency of machine learning algorithms, as well as the processing circuitry before and after the machine learning operations. Flexibility requires switches, and more switches result in higher area, circuit, and interconnect cost (Figure 6).



(a)

Switch	R Loss?	Nonvol?	Maturity	Application
CMOS FET	Y	N	Y	Rapid
	(≈1-10kΩ/μm)	(On power)	(CMOS)	Reconfig
FG FET	Y	Y	Y	Analog &
	(≈1-10kΩ/μm)	(> 10 years)	(CMOS)	Digital Storage
MESFETs	less	N	Y	III-V material
	(≈0.1-1kΩ/μm)	(On power)	(not CMOS)	systems
RRAM	Y	≈ Y?	???	????
	(≈1kΩ/μm)	(≈ 1 month?)		???
Calcoginide	low	Y	ok	Antenna and
	(≈1-10Ω/μm)	(months)		RF switches

(b)

Figure 6. Impact of switches on routing architecture. (**a**) Continuum of FPAA routing granularity. (**b**) Switch types compared considering significant resistive (R) losses nonvolatile capabilities technology maturity for large-scale capabilities, and applications for these switches.

A configurable architecture will always be a factor higher cost (K) in area, however small, compared with the area of a fully custom architecture (A_1). The custom block area (A_1) is fully custom, explicitly including in K any configurability or parameters. Each switch linearly increases K by a factor a that is the ratio of the size of a switch compared with an individual selection block. For n switches for configuring a custom block area, the resulting area (A) is

$$A = KA_1, K = 1 + an, A = (1 + an)A_1. \quad (1)$$

Typical values of a for small to moderate cells connected to this switch would be between 0.1 and 0.01; switches selecting a single transistor element would have a closer to 1. Switches connecting individual transistors offers significant opportunities [36,37]

with significant additional cost ($a > 1$). Switch implementation in a particular technology (Figure 6b) directly affects a.

Switches add circuit costs to the flexible fabric. With the increase in the area by (K), the custom computation has a total load capacitance (C_L), and the configurable computation has an increased capacitance roughly scaled by the cost factor (K). Area efficiency due to configurability is the inverse of cost (1/K). The custom computation power-delay product (E_1) would be proportional to C_L that is proportional to A_1:

$$E_1 \propto C_L \propto A_1. \tag{2}$$

For subthreshold operation, near-threshold operation, and some other situations, E_1 is constant with frequency. The size, weight, and power (SWaP) metric, a product of the area and power-delay product for a custom and configurable system, are

$$\text{Custom : SWaP} \propto E_1 A_1,$$

$$\text{Configurable} \propto K^2 E_1 A_1. \tag{3}$$

Furthermore, CMOS switches have a resistive loss (Figure 6b), although other technologies (e.g., III–V transistors and Chalcogenides) potentially can reduce the signal loss for an on switch. Switch approaches include SRAM-driven transmission gates [38–47], memristor elements [48–51], and phase-change memories [52–62], as well as FG devices.

Switch granularity is typically pictured as a continuum between course-grain granularity that has a minimum of switches between a menu of items, and fine-grain granularity, that has switches between the lowest level of components (Figure 6a). Different architectures create a different K in their attempt to achieve their desired flexility. Over the following subsections, we will develop the cost of configurability (K), which trades off with the resulting increase in flexibility (ϕ), described as increased functionality when connecting n blocks together, including course-grain architectures (Section 3.1), Manhattan architectures (Section 3.2), and fine-grain architectures (Section 3.3).

3.1. Course-Grain Architectures

Given the concern about switches, many FPAAs (e.g., [14,38–47,63]) utilize course-grain architectures (Figure 6a), minimizing the number of switches (Figure 6b) and associated parasitics required for any particular computation. Course-grain architectures attempt to minimize the effect of additional switches by only switching between large fixed components, and the loss of opportunity by this strategy is incorporated into the flexibility metric (ϕ). In a simple crossbar network (Figure 7a), Φ is

$$\text{Selection :} \phi \approx n,$$

$$\text{Parallel connections : } \phi \approx n^2, \tag{4}$$

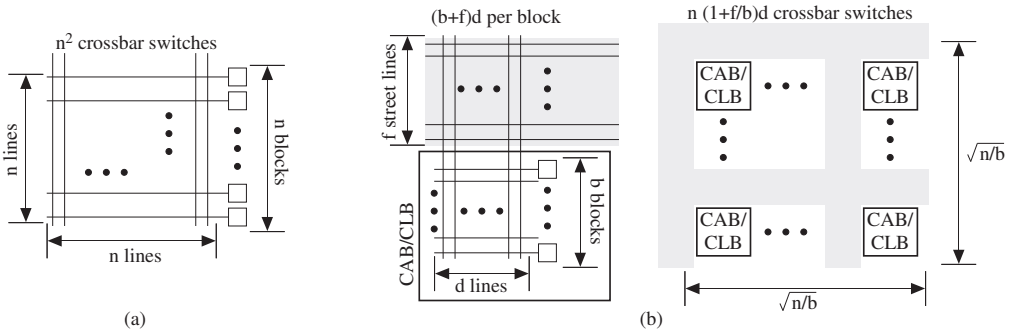
where each block could have a selection connection to the $n-1$ other blocks or could have parallel connections to each of the $n-1$ blocks (Figure 7c).

3.2. Manhattan Architectures Improve Flexibility

Manhattan architectures utilize a multilevel routing scheme to reduce the scaling of K with the number of elements while still achieving significant flexibility. Manhattan architectures enable reconfigurable and *efficient routing of local and sparse interconnections*, a critical issue for many neural network algorithms. FPGAs significantly improve their granularity through Manhattan architectures [64]. The evolution of configurable digital from fully connected structures to Manhattan-type approaches enabled the production of FPGAs with a routing structure that enabled a level of granularity beyond typical LUTs [64]. More flexible, efficient, and fine-grained granularity enables creativity by the designer; although, these approaches require placement and routing design tools [64]. The wide

range of configurations, as well as the portability of high-level code, enabled FPGAs to solve the digital design legacy as well as enabled secure FPGAs. These techniques require optimization algorithms to place and route an application into this architecture.

The Manhattan routing approach improves the effective granularity by assuming more connections are effectively local—typical of digital and analog designs. Manhattan routing structures (Figure 7b) assume a starting element size significantly larger than the crossbar determined by the local switch matrix parameters—b (CAB/CLB lines), d (lines into CAB/CLB), and f (lines in the connection block)—resulting in an improved scaling metric. The values of b, d, and f would increase weakly for increasing total number of nodes. K scales with local routing (b, d, f) within each module (e.g., CLB or CAB) instead of the entire array (Figure 7b). Other multilevel routing schemes have similar scaling properties. Manhattan architectures utilize these crossbar arrays in each of their local regions (CLB/CAB) typically having n = 8 to 64 block elements, where one wants to maximize Φ in each local region. A local region is defined as a large block where the routing architecture focuses on local computation, enabling these bus connections to only weakly grow with increased number of local regions and number of components.



Architecture	K	SWaP	ϕ
Simple Selection Crossbar (a)			$a\,n^2$
Simple Parallel Connection	$1 + a\,n$	$(1+an)^2$	$a\,n^3/a$
Parallel FG Analog (m-bit) Switch Parallel			$2^m an^3$
Selection Manhattan (b)			$ad(f+b)$
Parallel Switch Manhattan	$1 + a(1+\frac{f}{b})d$	$(1+a(1+\frac{f}{b})d)^2$	$ad^2(f+b)$
Parallel FG Analog (m-bit) Switch Manhattan			$a2^m d^2(f+b)$

(c)

Figure 7. Architecture scaling for interconnecting n-processors. (**a**) Individual crossbar array to fully connect n processors requiring O(n^2) switches. (**b**) Manhattan routing geometry to connect n processors having b processors in a CAB with d lines running out of the CAB onto the street (or **C**-block) of f lines. The number of switches for n processors becomes O($n(1+\frac{f}{b})d$). (**c**) Cost (K) and flexibility (ϕ) for n blocks as a function of typical architectures.

3.3. Fine-Grain Architectures

CMOS devices using non-volatile switches FG elements enables analog granularity. Analog parameters improve the resulting density and resulting system flexibility (Figure 7c). For analog m-bit switch elements, the increased parallel flexibility increases by a 2^m factor. Having analog parameters with parallel connections enables using routing fabric as computing fabric [65]. In this computing in memory approach [1,66], the number of additional switches, and therefore K, for a particular computation decreases significantly.

Manhattan architectures with fine-grain analog storage provide an energy- and area-efficient implementation of reconfigurable routing of local and sparse interconnections, where the neural network weight computation occurs directly through the weight fabric routing. In these cases, the network complexity scales as the number of neurons, and only weakly on the number of synapses within reasonable neuron sparsity.

Fine-grain granularity, particularly analog programmable granularity, greatly improves the tradeoff between configurable architecture efficiency (1/K) and flexibility (Φ),

requiring fewer nodes for similar flexibility as well as having a lower cost (K) of that flexibility (Figure 8). The higher granularity by parallel analog connections significantly decreases the number of components (n >> 1000 to n = 15), and the resulting SWaP efficiency (1/K → 0.1% to 80%) illustrates the potential advantage using switch elements as programmable transistors. Decreasing the required number of blocks for the same Φ illustrates the potential system-level reduction to achieve a range of potential applications.

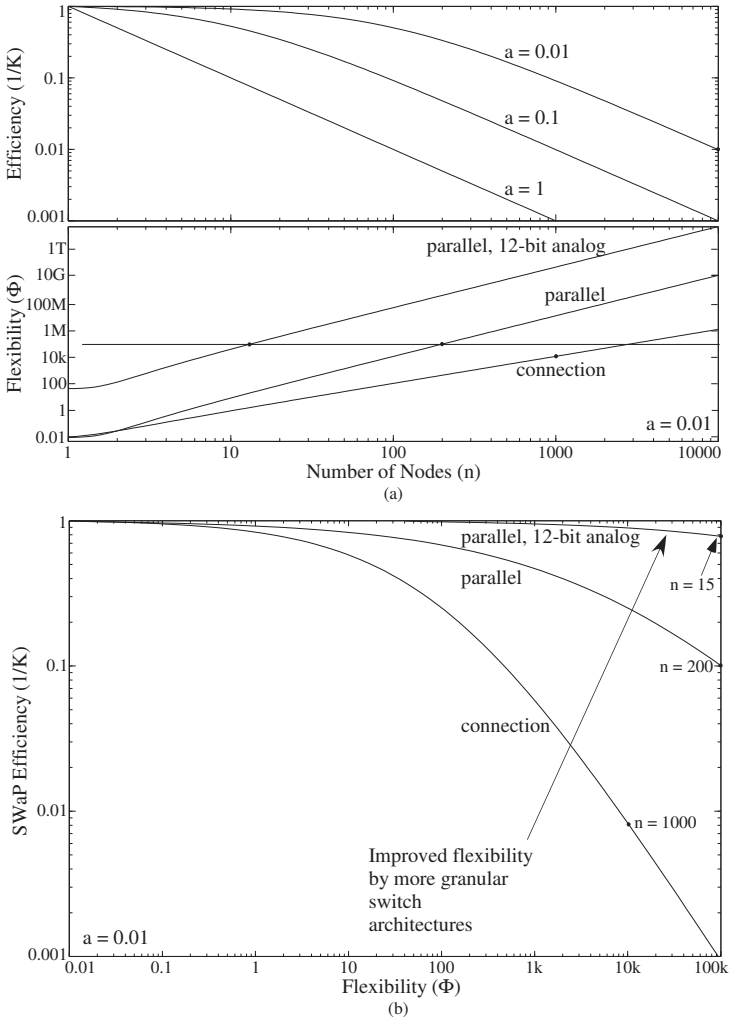


Figure 8. Impact of fine-grain granularity with analog programmability. (**a**) Configurable architecture efficiency (1/K) and flexibility for simple crossbar networks (Figure 7a) as a function of the number of blocks (n) for connection architectures, parallel switch architectures, and parallel analog (12-bit) switch architectures. (**b**) Efficiency (1/K) vs. flexibility (Φ) for these three architectures. Higher granularity, particularly analog-programmed granularity, enables significant flexibility with fewer resources (e.g., n), and at lower cost.

Fine-grain, analog switch architectures provides a favorable tradeoff between configurable architecture efficiency (1/K) and flexibility (Φ), and further research to enable these techniques should yield many significant opportunities. These advantages are consistent with the demonstrated orders of magnitude advantage of FPAA devices using analog switching matrices (e.g., SoC FPAA [1,28]). As the the high Φ makes the computational routing fabric more important, additional fabric infrastructure, such as partial high-speed in-circuit reconfigurability [28], further empowers the range of potential applications.

The difference in Φ between digital connection switches and parallel analog switches, enabling computing through the switches structured in a memory configuration, can be seen by the capabilities of a typical CLB and CAB (Figure 9). Where a typical CLB can impressively enable a state machine per CLB, a CAB could potentially implement a small

acoustic classifier stage in a single CAB. These differences in capabilities are almost entirely due to the fine-grain routing vs. an efficient traditional routing approach; course-grain routing techniques leave even more unused Φ and capability.

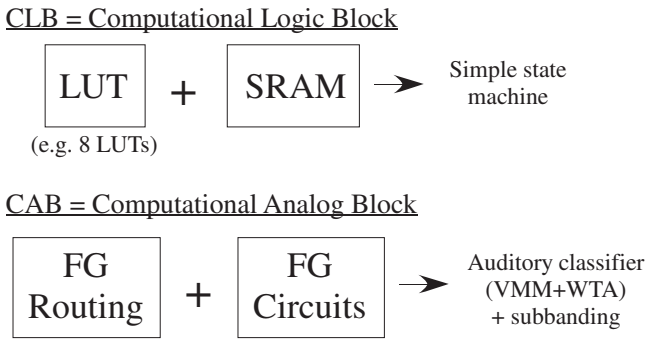


Figure 9. Comparison of the computation possible in a single local region, such as a computational logic block (CLB) or a computational analog block (CAB). A CLB typically uses binary connection switches to form multiple (e.g., 8) lookup tables and some selection RAM, enabling small state machines in a single CLB. A CAB typically uses analog parallel switches for its computation, that includes FG routing that can be used for programmable and configurable computation, as well as programmable FG-based circuits. Within such a structure, a small auditory classifier could be compiled in a single CAB.

4. Scaled FPAA Devices Opportunities towards Low-Energy Machine Learning

We want to *understand the scaling opportunities for new machine-learning-capable FPAA devices*, given the current FPAA capabilities. Configurable mixed-signal devices—-having the opportunity of flexibility in a reasonably granular solution—can justify the IC design cost for new embedded devices (Figure 10). As mask costs exponentially increase with decreasing processing node, the resulting design costs to obtain value from these investments exponentially increases, requiring a significantly higher expected market return from the effort (Figure 10). Only a few applications (e.g., cell phone processors) can have the market impact that are necessary to justify the cost of advanced IC nodes (e.g., 10 nm, 14 nm), where configurable solutions can utilize a single IC design across a number of applications to justify the investment cost.

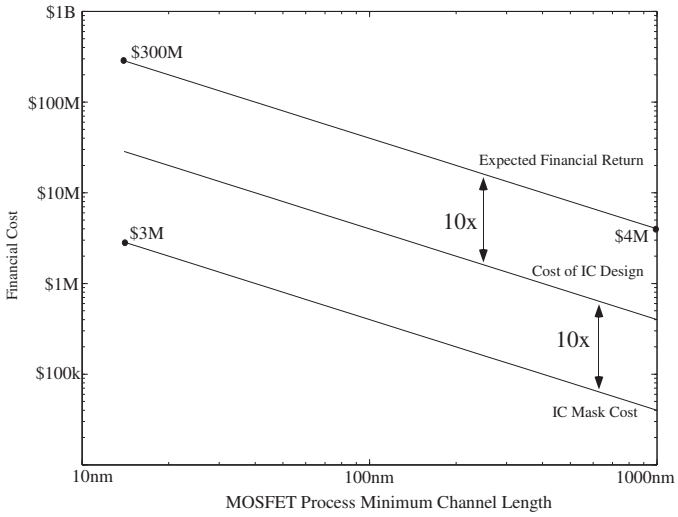


Figure 10. Configurable devices such as FPGA and FPAA devices provide a cost effective machine learning end-to-end solution, because of the ever-increasing cost of IC design for scaled-down processes. The costs for making a set of IC masks scales inversely as a power law of the CMOS minimum channel length, and typically the design cost for a new design is at least 10× the mask cost, typically requiring a 10× the expected financial return to even attempt such a venture. The resulting cost for designing an IC is often far too high for most engineering applications to hope to reach these financial returns. A configurable device can spread this resulting engineering cost over a wide number of designs.

4.1. Machine Learning Computation Opportunities from Scaled CMOS FPAAs

Given the demonstration of high flexibility of fine-grain capabilities compared with architectural cost, as well as demonstration of SoC FPAAs for embedded machine inference and learning [2,28], we ask the following question: What is the potential of these FPAA devices given the existing understanding of these techniques? Scaling allows for a higher signal bandwidth in the FPAA fabric architectures (Figure 11a), roughly with an inverse quadratic scaling on the minimum channel length, enabling some RF bandwidths (e.g., 4 GHz) at 40–45 nm CMOS [67,68].

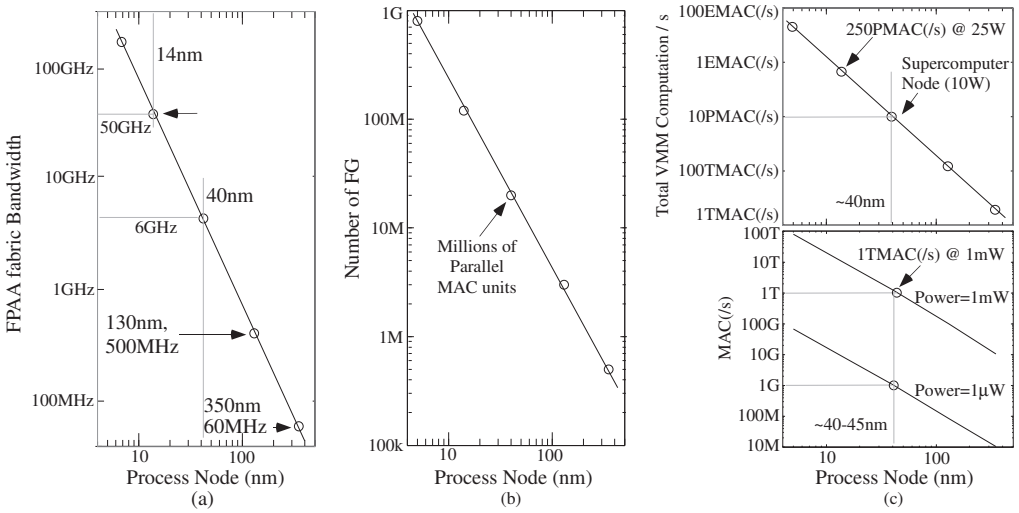


Figure 11. Bandwidth, parallel computational units, computational capability, and computational efficiency for scaled-down FPAA devices, extrapolated from experimental measurements and early studies of scaled-down FG and FPAA devices. (**a**) Typical FPAA fabric bandwidth for scaled process nodes. (**b**) Number of FG elements for scaled process nodes that directly relates to the number of parallel computations (e.g., MAC). (**c**) Computational capability and computational efficiency for scaled-down FPAA devices.

FG devices scale to a number of CMOS processes (e.g., 40 nm, 14 nm) [67], and do not limit any expected scaling opportunities. FG devices have been demonstrated across a number of IC technologies from 2.0 µm to 40 nm CMOS [9,67–69], with designs awaiting measurement in 14 nm CMOS. Issues around different insulators [67,68], temperature issues on circuit operation [70,71], handling of voltage levels on-chip for programming [1,72,73], and long-term reliability and multiple writing [30] have all been carefully studied and their capabilities across standard CMOS processes have been established. Future IC processes from 14 nm and below show no significant constraint on further scaling of these devices, as CMOS continues to scale to smaller channel lengths.

The impact of scaled-down FPAA devices for end-to-end machine learning (e.g., neural network) applications builds from a number of experimental results of FPAA devices at the 350 nm CMOS node. Initial early experimental measurements in smaller IC processes (e.g., [67,68]) and efforts in analog system automation derived from FPAA tools (e.g., [1,31,32]) give significant grounding of these next generation FPAAs. The FPAA CAB/CLB density should be similar to existing FPGA densities. The complexity of a CLB in the 350 nm SoC FPAA [1] is the same size and complexity (8 LUTs + registers) as the CLBs used by Xlinix Zynq RF-enabled devices. The 200 CAB+ CLBs in existing 350 nm FPAAs could be optimized to improve the density by at least a factor of 2. Scaled sizes show similar number of CLBs in a similar area 20–40 k (40 nm CMOS) vs. 50 k in Zynq RF (12–16 nm CMOS) [24]. Many different features might be possible in each case, and yet, these approaches are of a similar order of magnitude.

Scaling creates smaller switches and processing elements, resulting in higher density and lower energy consumption (Figure 11b,c). One expects a significantly increased number of FG devices with CMOS scaling depending on process capabilities (Figure 11b). The

number of vector–matrix multiplications (VMM) on a 5 mm × 5 mm die grows rapidly with decreasing process node (Figure 11b); these results assume that 1/8 of the total routing fabric are VMM computations. The 45 nm and 14 nm devices are capable of PMAC(/s) level computation on a single die, computation levels typically requiring a large supercomputer (Figure 11c). A neural network or similar machine learning problem could utilize PMAC(/s) computations for inference and learning. As one expects 10,000–10,000,000,000 PMAC for current and future large fielded applications (e.g., [5]), a single 10 W, 10 PMAC(/s) device at 40 nm and a single 25 W, 250 PMAC(/s) device at 14 nm dramatically decreases the computing time and power requirements, as well as decreases significantly the overall energy requirements (Table 1).

Table 1. Potential scaled FPAA computation time, energy, and power for machine learning tasks.

CMOS process	10,000 PMAC Time, 1 device	Power/ Energy	10,000,000,000 PMAC Time, 100,000 devices	Power/ Energy
40 nm	20 min	10 W → 10 kJ	3.3 h	1 MW → 10 GJ
14 nm	2 min	25 W → 1 kJ	20 min	2.5 MW → 1 GJ

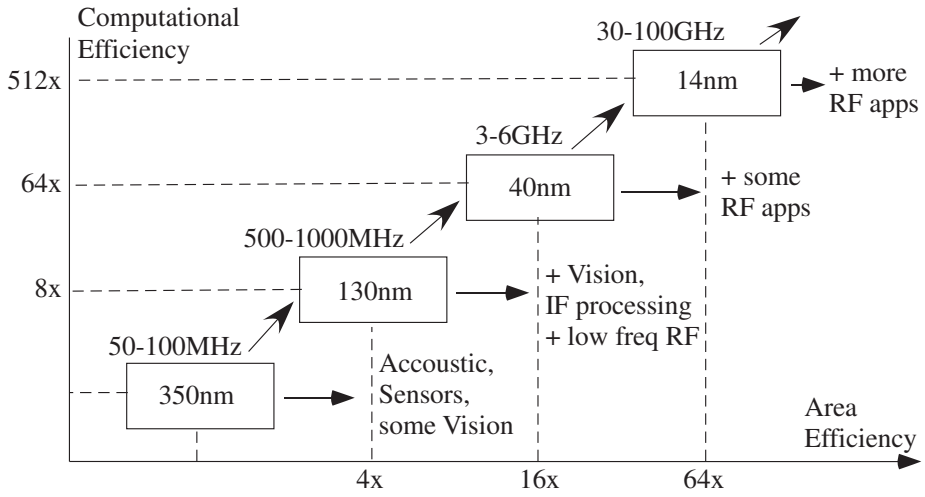
Another important aspect of these devices would be the significantly lower required energy consumption for these operations, particularly the possible computation at 1 μW and 1 mW levels (Figure 11c). The range of machine learning (inference and learning) demonstrations have been demonstrated with the 350 nm CMOS FPAA device [2–4]. In this application, a range of acoustic microphone-to-classification machine learning (inference and training) techniques have been demonstrated with inference in 350 nm CMOS at 20–30 μW levels. Improved circuit design in 350 nm CMOS would already move these devices to 1 μW levels [2]; therefore, scaled devices certainly would certainly implement machine learning for similar applications at 1 μW levels. Further improvements by utilizing more neuromorphic physical algorithms in FPAA devices, such as neurons, synapses, and dendrites (e.g., [9,74–76]), with their improved energy efficiency over analog matrix–vector multiplication used in NN (e.g., [9]), further illustrate the opportunities for energy-efficient neural network and machine learning computing at 1 μW energy levels.

A device requiring 1 mW average energy could be easily supplied by a battery, enabling months of continuous fielded use, and a device requiring 1 μW average energy could easily be supplied by small (<1 cm^2) energy-harvesting devices. Energy levels of 1 mW can possibly be supplied by moderate-sized (e.g., 10 cm × 10 cm) devices. A 40 nm CMOS structure enables 1GMAC(/s), around the level of a fully capable laptop computer, and around 1 TMAC(/s), around the level of a small GPU or FPGA cluster. Embedded low-power *end-to-end* embedded machine learning powered through energy harvesting eliminates the need for external energy sources, eliminating part of the machine learning energy crisis [5].

4.2. Algorithm Opportunities from Scaled CMOS FPAAs

The current [1] and future (Figure 12) FPAA directions towards machine learning show opportunities for embedded *end-to-end* system algorithms. Increased density, decreased energy consumption, and increased bandwidth at each CMOS node (Figure 12) directly impact the range of computations; although, the same computations are possible at each CMOS node at a lower operating frequency and problem size.

Developing *end-to-end* machine learning systems require building the computation before the machine learning, as well as the computing for the machine learning operations (Figure 13). The optimal operating frequency matches the input data rate to eliminate the need for any internal storage or related infrastructure. For a given operating frequency, the objective is to minimize energy consumption, as well as the classification latency (small for analog computing). Analog numerical analysis [77], analog architecture theory [78], and real-valued computing theory [12] provides the framework for analog computations.



(a)

	350nm	130nm	45nm	14nm
Fabric voltage	2.5V	1.5V	1V	0.8V
CAB/CLB:5mm×5mm	200	1000	10000	100000

(b)

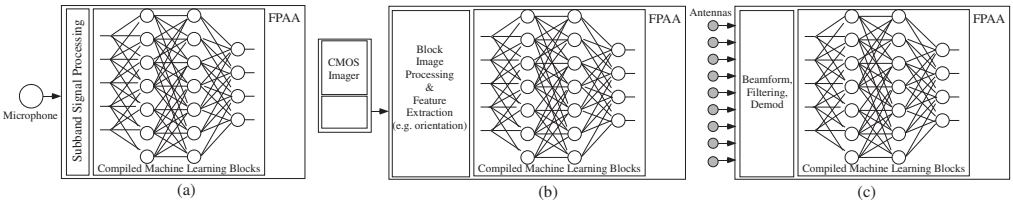
Figure 12. Scaling FPAA devices enables a range of new applications at scaled-down nodes. (**a**) Scaling FPAA devices from 350 nm CMOS to 130 nm, 40 nm, and 14 nm processes predictably increases the computational efficiency, area per operation, and fabric bandwidth, as well as opening new application spaces at each node. (**b**) Fabric operating conditions and CAB/CLB sizes for scaled FPAA devices.

A single 40 nm FPAA device (Figure 12) could potentially be used for an acoustic NN classifier, an image NN classifer, and an RF NN classifer. These examples provide a good comparison with a single high-end (e.g., RF Zync [24]) FPGA IC (12–16 nm CMOS), solving these applications (Figure 13). The 5 GHz FPAA bandwidth (dc to 5 GHz [67]) compares to the highest-end 10 GSPS DACs on the RF Zynq [24]. The FG elements that are not programmed are initialized to accumulation, and the negligible current (e.g., [67]) results in little static power required in a 40 nm node. An FPGA has significant initial static power (100's of mW to W) because of the on-chip SRAM storage. A 40 nm CMOS FPAA would have roughly 20 k CABs and CLBs with roughly 50 M multiplication elements in the FPAA fabric that are likely to be typically accessible. The highest end Zync processor has roughly 50 k CLBs and 4 k DSP (with single multiply) units. This 40 nm FPAA device would be similar to the 350 nm SoC FPAA, likely with multiple embedded processors (MSP430 or RISC V) in an open design architecture; Zync includes 6 ARM cores with a 1–2 GHz maximum clock rate. A 14 nm FPAA device should have improved metrics; a 40 nm FPAA device requires considerably lower building costs and design costs.

End-to-end solutions requires different front-end solutions for different applications (e.g., acoustic, imaging, RF) to set the data before the network inference and training (Figure 13). Acoustic or speech processing or classification scaling increases the problem size (command word to small vocabulary to speech classification), while potentially further improving the energy efficiency. Acoustic applications often require front-end filterbanks, delay lines, and other sub-banding processes before the machine learning computation, as well as asynchronous event processing to encode the machine learning process (Figure 13a).

A 40 nm FPAA, similar to the process which is already carried out in 350 nm SoC FPAA, could directly implement the front-end computation before the classifier would include BPF over acoustic frequencies, amplitude detection, and continuous time delay approximations in each parallel channel. One expects energy costs of 1–20 μW for this entire front-end infrastructure (e.g., [3,28]) with an upper-end output frequency of 1 kHz for each output. A speech recognizer using multiple neural layers (phonemes, syllables, words) in realistic SNR (<10 dB) environments with a moderate number of weights (10 M) that might use single-layer VMM+WTA blocks (e.g., [3]). As the NN computation requires 10 μW

(10 GMAC(/s)), one easily expects this application to fit in a 50 µW budget, including the additional potential overheads.



Application	NN Compute	FPGA Number	FPGA Power	FPAA Number	FPAA Power
Acoustic	10 GMAC(/s)	1	≈ 1W	1	50μW
Image	10 TMAC(/s)	2-3	≈ 100W	1	40mW
RF	10 PMAC(/s)	2500	> 100kW	1	10W
Accelerator	10 EMAC(/s)	2.5M	> 100MW	1000	20kW

(d)

Figure 13. A single FPAA device could enable a range of embedded machine learning applications (**a**) A microphone sensor (acoustic)-embedded machine learning problem. (**b**) A CMOS imager-sensor-embedded machine learning problem. (**c**) A multi-antenna (RF)-embedded machine learning problem. (**d**) Summary of potential resources for acoustic, image, RF, and NN training accelerator applications comparing against a high-end RF Zynq Xlinix FPGA (12–16 nm CMOS) and potential 40 nm FPAA (40 nm CMOS).

In other cases, the increased problem size opens new architectural solutions, such as an in-image classification, where more processing can occur on the incoming streamed image from a sensor (Figure 13b). Image classification on larger nodes might take a standard database with an on-board compression (e.g., compressed DCT [79]), where a scaled-down system would compute and classify sub-images in parallel.

Typical image processing and NN classification architectures would build around an image IC that transmits an image one pixel value at a time (Figure 14a). Matching data and computing speed results in optimal computing efficiency and minimal overhead minimizing the expensive requirement for buffering or caching data. An alternate path could enable a CMOS imager to have direct interconnections between a pixel and reconfigurable Si processing on another IC that would enable significantly higher computing opportunities (Figure 14b). Unlike the trajectories for single wafer imagers (e.g., [80,81]), through-hole wafer connections and die stacking (e.g., [82,83]) enable a wide range of opportunities for 3D die-stacked imagers [84–89]. Image processing would have CABs or groups of CABs for handling different symbols. Data moves throughout the IC and utilizes local computing in the memory. A 10 M Pixel imager requires an average of 10 TMAC (/s) at roughly 10–20 mW in the FPAA device, similar cost as the CMOS imager, and similar cost in transmitting the data between the two chips on a PCB. If an imager has vertical connections from one wafer to another, then the vertical connection would change the algorithm and more parallelism would occur due to parallel input (naturally from source).

An FPAA device could be a common module for RF related machine learning (Figure 13c) at 45 nm and smaller CMOS nodes [68]. CMOS scaling enables this application, such as beamforming and demodulation, that could be 40 MHz at 350 nm CMOS, while improving to 400 MHz at 130 nm CMOS, 4 GHz at 40 nm CMOS, and higher for smaller CMOS processes [67]. The device, depending on process node, might include some specialized LNA at input, configuration for initial signal processing (e.g., VMM for beamforming), as well as demodulation for classifying modulated signals. The 40 nm FPAA device can be directly built to enable 10–20 GHz signals with 4–5 GHz signal bandwidths (dc to 4–5 GHz) in the routing fabric [68], including FG tunable delay elements for spatiotemporal filtering through the routing fabric. These inputs could directly be used to classify spectrum dynamics, utilizing a large classifier network (again, 10 M NN), as well as utilizing those dynamics with minimal system latency [78]. A 10 M NN operating at 1 GHz bandwidth puts the computation at 10 PMAC(/s) range operating at 10 W of average power.

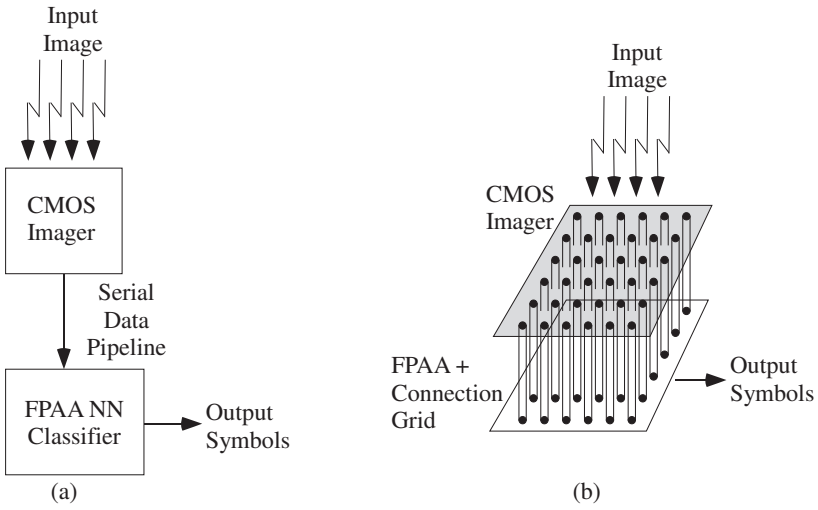


Figure 14. CMOS imager interfacing options (**a**) Interfacing between two ICs. Single data input with single pixels arriving on each sample. Different imagers (e.g., event) have different approaches, but still have a single pixel per time scale. (**b**) Vertical stacking of a CMOS imager and a FPAA device with multiple inputs for each pixel allowing parallel processing of input signals.

Using multiple FPAA devices could enable a platform for larger algorithms and an accelerator for training a network. Training current NN models requires multiple parallel networks training for different initial conditions and other parallel-type structures. One would expect many FPAA chips compiled to the desired NN structure with the same signal and training inputs with the system recording the near-digital classified outputs, and eventually reading the converged weight values. Rewriting the networks consumes a small percentage of time compared with the training algorithm. Supplying an arbitrary network with vectors of analog signals requires a large number of DACs; therefore, compiling and using the preprocessing algorithms on the FPAA with the NN would drastically reduce the amount of DACs (e.g., acoustic classification), system complexity (e.g., input data movement), and energy requirements by utilizing the end-to-end computing. Again using the same 40 nm FPAA devices with an application of 10 M weights with the bandwidth accelerated (where possible) to 1 GHz speeds, one expects 10P MAC(/s) range operating at 10 W of average power. A parallel system of 1000 devices would occupy a rack infrastructure with 10 kW of computing power. The infrastructure to control the system (input DAC signals) and store the resulting inputs likely requires similar complexity for this 10 EMAC (/s) system. FPGAs used in accelerators for training follow a similar path, although with higher energy and complexity (data movement) requirements than those seen in the other applications. An FPGA system would also benefit from implementing front-end processing, reducing the input complexity. The FPAA system would require roughly 20–40 kW of power, and the FPGA system would require on the order of 100 MW of power, typical of a data-computing node. The large 10,000,000,000 PMAC(/s) production training problems would require roughly 2 weeks of compute time, assuming the FPAA reconfiguration/reloading time is a small fraction of the computing time.

5. Summary and Further Directions

We have shown that FPAAs have the potential to handle machine inference and learning applications with significantly lower energy requirements, potentially alleviating the high cost experienced today even in cloud-based systems. FPAA devices enable embedded machine learning—one form of physical mixed-signal computing—enabling machine learning and inference on low-power embedded platforms, particularly edge platforms. The SoC FPAA device uses fine-grain analog programmability and therefore minimizes the high cost of fine-grain switch networks. Current FPAA devices are platform of mixed-signal development as well as analog-enabled computing, and future FPAA devices will signifi-

cantly increase the size, area, and energy efficiency of these capabilities. Next-generation FPAAs can handle loads of 10,000–10,000,000,000 PMAC required for current and future large fielded applications at orders of magnitude of lower energy levels than expected by current technology, motivating the need to develop these new generations of FPAA devices.

An *end-to-end* solution perspective tends to take the computing communication issues into account, from sensor to classified result. As part of the configurability items, the related architecture constraint requires avoiding large external memories, because they will significantly reduce performance without significantly increasing Φ. The local computing eliminates the deep communication to memories, as well as difference in learning architecture efficiently computed, etc. [78]. Manhattan architectures with fine-grain analog storage provide an energy- and area-efficient implementation of reconfigurable routing of local and sparse interconnections, where the neural network weight computation occurs directly through the weight fabric routing. In these cases, the network complexity scales as the number of neurons, and only weakly on the number of synapses within reasonable neuron sparsity.

FPAAs for online learning may want to incorporate direct FG on-chip learning algorithms into the architecture. FPAA networks that both use FG elements for routing as well as for adaptation have been demonstrated, requiring use of nFET FG elements to route the pFET adaptive paths while retaining their immediate state. These architectures should be considered in the next generations of FPAA devices where specializations towards machine learning would be used.

One might question the long period of research and development before a commercial SoC FPAA device will be available. The development of FPGA devices took considerable time in an environment where digital computing was well established with a clear framework, and other system design issues were happening in parallel [64]. The success of the SoC and earlier FPAA devices [1] led to the development of an initial complete toolset [33], a toolset that shows the next steps in FPAA automation. These successes to develop 350 nm FPAA devices to a significant size (nearly 1 M FG parameters, $\approx 200CAB + CLB$, µP) with design tools capable of designing an entire FPAA target, led to development of analog computing techniques [12,90] that included foundational work on analog numeric [77], architectures [78], and abstraction [91]. Like the SoC FPAA devices, these techniques are still relatively new, and are expanding towards new applications (e.g., [92]).

From a technical perspective, the current and projected SoC FPAA capabilities could impact a range of applications. Commercial success often requires an alignment of a combination of technical and non-technical factors, and the path for platform technologies, such as FPGAs, is rarely a linear path [64]. The decades of market interest in Anadigm FPAAs, as well as recent interest in the FG-based Aspinity FPAAs, shows a continued market interest for FPAAs given its limited capabilities. Even though it is hard to predict commercial opportunities, the technical capabilities potentially open new opportunities in low-power end-to-end computing for machine learning applications.

Funding: This research received no external funding.

Data Availability Statement: Data sharing not applicable.

Conflicts of Interest: The author declares no conflict of interest.

References

1. Hasler, J. Large-Scale Field Programmable Analog Arrays. *Proc. IEEE* **2020**, *108*, 1283–1302. [CrossRef]
2. Hasler, J.; Shah, S. Learning for VMM + WTA Embedded Classifiers. In Proceedings of the GOMAC, Orlando, FL, USA, 14–17 March 2016.
3. Hasler, J.; Shah, S. SoC FPAA Hardware Implementation of a VMM+WTA Embedded Learning Classifier. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2018**, *8*, 28–37.
4. Hasler, J.; Shah, S. VMM + WTA Embedded Classifiers Learning Algorithm implementable on SoC FPAA devices. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2018**, *8*, 65–76. [CrossRef]

5. Thompson, N.C.; Greenewald, K.; Lee, K.; Manso, G.F. Deep Learning's Diminishing Returns: The Cost of Improvement is Becoming Unsustainable. *IEEE Spectr.* **2021**, *58*, 50–55. [CrossRef]
6. Mead, C. Neuromorphic electronic systems. *Proc. IEEE* **1990**, *78*, 1629–1636. [CrossRef]
7. Chawla, R.; Bandyopadhyay, A.; Srinivasan, V.; Hasler, A. 531 nW/MHz, 128 × 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity. In Proceedings of the IEEE Custom Integrated Circuits Conference, Orlando, FL, USA, 6 October 2004; pp. 651–654.
8. Schlottmann, C.; Hasler, P. A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation. *IEEE J. Emerg. Sel. Top. Circuits Syst.* **2011**, *1*, 403–411. [CrossRef]
9. Hasler, J.; Marr, H.B. Finding a Roadmap to achieve Large Neuromorphic Hardware Systems. *Front. Neuromorphic Eng.* **2013**, *7*, 1–29. [CrossRef]
10. Rosenblatt, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychol. Rev.* **1958**, *65*, 386–408. [CrossRef]
11. Widrow, B.; Hoff, M.E. Adaptive switching circuits. *IRE WESCON Conv.* **1960**, *4*, 96–140.
12. Hasler, J.; Black, E. Physical Computing: Unifying Real Number Computation. *J. Low Power Electron. Appl.* **2021**, *11*, 1. [CrossRef]
13. Hall, T.; Twigg, C.; Gray, J.; Hasler, P.; Anderson, D. Large-scale field-programmable analog arrays for analog signal processing. *IEEE Trans. Circuits Syst. I* **2005**, *52*, 2298–2307. [CrossRef]
14. EE Times. Specifically Generic Analog Functions for FPAAs Anadigm Says. Available online: https://www.eetimes.com/specifically-generic-analog-functions-for-fpaas/ (accessed on 15 February 2022).
15. Kuon, I.; Tessier, R.; Rose, J. FPGA Architecture: Survey and Challenges. *Found. Trends Electron. Des. Autom.* **2007**, *2*, 135–253. [CrossRef]
16. Nguyen, M.; Serafin, N.; Hoe, J.C. Partial Reconfiguration for Design Optimization. In Proceedings of the IEEE International Conference on Field-programmable Logic and Applications (FPL), Gothenburg, Sweden, 31 August–4 September 2020.
17. Chung, E.S.; Milder, P.A.; Hoe, J.C.; Mai, K. Single-chip Heterogeneous Computing: Does the future include Custom Logic, FPGAs, and GPUs? In Proceedings of the 2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture, Atlanta, GA, USA, 4–8 December 2010; pp. 53–64.
18. Chung, E.S.; Hoe, J.C.; Mai, K. CoRAM: An In-Fabric Memory Architecture for FPGA-based Computing. In Proceedings of the ACM Field-Programmable Gate Arrays (FPGA), Monterey, CA, USA, 27 February–1 March 2011; pp. 97–106.
19. Farooq, U.; Marrakchi, Z.; Mehrez, H. FPGA Architectures: An Overview. In *Tree-Based Heterogeneous FPGA Architectures*; Springer: New York, NY, USA, 2012; pp. 7–48.
20. Nguyen, M.; Tamburo, R.; Narasimhan, S.; Hoe, J.C. Quantifying the Benefits of Dynamic Partial Reconfiguration for Embedded Vision Applications. In Proceedings of the Field-Programmable Logic and Applications (FPL), Barcelona, Spain, 8–12 September 2019.
21. Boutros, A.; Nurvitadhi, E.; Ma, R.; Gribok, S.; Zhao, Z.; Hoe, J.C.; Betz, V.; Langhammer, M. Beyond Peak Performance: Comparing the Real Performance of AI-Optimized FPGAs and GPUs. In Proceedings of the Field-Programmable Technology (FPT), Maui, HI, USA, 9–11 December 2020.
22. Boutros, A.; Betz, V. FPGA Architecture: Principles and Progression. *IEEE Circuits Syst. Mag.* **2021**, *21*, 4–29. [CrossRef]
23. Nikoli, S.; Tkei, Z.; Lenne, P.; Catthoor, F. Global Is the New Local: FPGA Architecture at 5 nm and Beyond. In Proceedings of the 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Virtual, 28 February–2 March 2021.
24. Zynq UltraScale + RFSoC Data Sheet. *Xlinix*. DS889 v. 1.13. 7 January 2022. Available online: https://www.xilinx.com/support/documentation/data_sheets/ds889-zynq-usp-rfsoc-overview.pdf (accessed on 20 January 2022).
25. Marr, B.; Degnan, B.; Hasler, P.; Anderson, D. Scaling energy per operation via an asynchronous pipeline. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2013**, *21*, 147–151. [CrossRef]
26. IGLOO2 FPGA, Microsemi. Available online: https://www.microsemi.com/product-directory/fpgas/1688-igloo2 (accessed on 8 May 2021).
27. Available online: https://www.microsemi.com/product-directory/antifuse-fpgas/1700-axcelerator (accessed on 8 May 2021).
28. George, S.; Kim, S.; Shah, S.; Hasler, J.; Collins, M.; Adil, F.; Wunderlich, R.; Nease, S.; Ramakrishnan, S. A Programmable and Configurable Mixed-Mode FPAA SoC. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2016**, *24*, 2253–2261. [CrossRef]
29. Kim, S.; Hasler, J.; George, S. Integrated Floating-Gate Programming Environment for System-Level Ics. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2016**, *24*, 2244–2252. [CrossRef]
30. Srinivasan, V.; Serrano, G.; Twigg, C.; Hasler, P. A Floating-Gate-Based Programmable CMOS Reference. *IEEE Trans. Circuits Syst. I* **2008**, *55*, 3448–3456. [CrossRef]
31. Collins, M.; Hasler, J.; George, S. An Open-Source Toolset Enabling Analog–Digital–Software Codesign. invited paper. *J. Low Power Electron. Appl.* **2016**, *6*, 3. [CrossRef]
32. Kim, S.; Shah, S.; Wunderlich, R.; Hasler, J. CAD Synthesis Tools for Large-Scale Floating-Gate FPAA System. *J. Des. Autom. Embed. Syst.* **2021**, *25*, 1–16.
33. Hasler, J. A CMOS Programmable Analog Standard Cell Library in Skywater 130 nm Open-Source Process. In Proceedings of the Workshop on Open-Source EDA Technology, WOSET, Virtual, 4 November 2021.
34. Hasler, J.; Shah, S. Security Implications for Ultra-Low Power Configurable Analog and Mixed Mode SoC Systems. *J. Low Power Electron. Appl.* **2018**, *8*, 17. [CrossRef]

35. Nease, S.; Lanterman, A.; Hasler, J. A Transistor Ladder Voltage-Controlled Filter Implemented on a Field Programmable Analog Array. *JAES* **2014**, *62*, 611–618. [CrossRef]
36. Keymeulen, D.; Zebulum, R.S.; Jin, Y.; Stoica, A. Fault-tolerant evolvable hardware using field-programmable transistor arrays. *IEEE Trans. Reliab.* **2000**, *49*, 305–316. [CrossRef]
37. Stoica, A.; Zebulum, R.; Keymeulen, D.; Tawel, R.; Daud, T.; Thakoor, A. Reconfigurable VLSI architectures for evolvable hardware: From experimental field programmable transistor arrays to evolution-oriented chips. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2001**, *9*, 227–232. [CrossRef]
38. Gaudet, V.; Gulak, G. 10 MHz field programmable analog array prototype based on CMOS current conveyors. In Proceedings of 1999 Micronet Annual Workshop, Ottawa, ON, Canada, 26–17 April 1999.
39. Lee, E.K.F.; Gulak, P.G. Field programmable analogue array based on MOSFET transconductors. *Electron. Lett.* **1992**, *28*, 28–29. [CrossRef]
40. Klein, H.W. The EPAC architecture: An expert cell approach to field programmable analog circuits. In Proceedings of the 39th Midwest Symposium on Circuits and Systems, Ames, IA, USA, 21 August 1996; Volume 1, pp. 169–172.
41. Becker, J.; Manoli, Y. A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable GM-cells. In Proceedings of the 2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No. 04CH37512), Vancouver, BC, Canada, 23–26 May 2004; pp. I.1092–I.1095.
42. Cowan, G.E.R.; Melville, R.C.; Tsividis, Y.P. A VLSI analog computer/digital computer accelerator. *IEEE J. Solid-State Circuits* **2006**, *41*, 42–53. [CrossRef]
43. Guo, N.; Huang, Y.; Mai, T.; Patil, S.; Cao, C.; Seok, M.; Sethumadhavan, S.; Tsividis, Y. Energy-efficient hybrid analog/digital approximate computation in continuous time. *IEEE J. Solid-State Circuits* **2016**, *51*, 1514–1524. [CrossRef]
44. Huang, Y.; Guo, N.; Seok, M.; Tsividis, Y.; Mandli, K.; Sethumadhavan, S. Hybrid analog-digital solution of nonlinear partial differential equations. In Proceedings of the 2017 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), Boston, MA, USA, 14–17 October 2017; pp. 665–678.
45. Kelly, B.M.; Rumberg, B.; Graham, D.W.; Kulathumani, V. Reconfigurable analog signal processing for wireless sensor networks. In Proceedings of the 2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS), Columbus, OH, USA, 4–7 August 2013; pp. 221–224.
46. Rumberg, B.; Graham, D.W.; Clites, S.; Kelly, B.M.; Navidi, M.M.; Dilello, A.; Kulathumani, V. RAMP: Accelerating wireless sensor hardware design with a reconfigurable analog/mixed-signal platform. In Proceedings of the 14th International Conference on Information Processing in Sensor Networks, Seattle, WA, USA, 13–16 April 2015; pp. 47–58.
47. Rumberg, B.; Graham, D.W. A low-power field-programmable analog array for wireless sensing. In Proceedings of the Sixteenth International Symposium on Quality Electronic Design, Santa Clara, CA, USA, 2–4 March 2015; pp. 542–546.
48. Kim, K.H.; Gaba, S.; Wheeler, D.; Cruz-Albrecht, J.M.; Hussain, T.; Srinivasa, N.; Lu, W. A functional hybrid memristor crossbar-array/CMOS system for data storage and neuromorphic applications. *Nano Lett.* **2012**, *12*, 389–395. [CrossRef]
49. Jo, S.H.; Kim, K.H.; Lu, W. Short-term memory to long-term memory transition in a nanoscale memristor. *ACS Nano* **2011**, *9*, 7669–7676.
50. Jo, S.H.; Lu, W. Programmable resistance switching in nanoscale two-terminal devices. *Nano Lett.* **2009**, *9*, 496–500. [CrossRef]
51. Snider, G.; Amerson, R.; Carter, D.; Abdalla, H.; Qureshi, M.S.; Leveille, J.; Versace, M.; Ames, H.; Patrick, S.; Chandler, B.; et al. From synapses to circuitry: Using memristive memory to explore the electronic brain. *IEEE Comput.* **2011**, *44*, 21–28. [CrossRef]
52. Pohm, A.; Sie, C.; Uttecht, R.; Kao, V.; Agrawal, O. Chalcogenide glass bistable resistivity (Ovonic) memories. *IEEE Trans. Magn.* **1970**, *6*, 592. [CrossRef]
53. Lee, B.C.; Ipek, E.; Mutlu, O.; Burger, D. Architecting phase change memory as a scalable dram alternative. In Proceedings of the 36th Annual International Symposium on Computer Architecture, Austin, TX, USA, 15 June 2009.
54. Karpov, I.V.; Kencke, D.; Kau, D.; Tang, S.; Spadini, G. Phase Change Memory with Chalcogenide Selector (PCMS): Characteristic Behaviors, Physical Models and Key Material Properties. *MRS Online Proc. Libr.* **2010**, *1250*, 70.
55. Burr, G.W.; Breitwisch, M.J.; Franceschini, M.; Garetto, D.; Gopalakrishnan, K.; Jackson, B.; Kurdi, B.; Lam, C.; Lastras, L.A.; Padilla, A.; et al. Phase change memory technology. *J. Vac. Sci. Technol. B* **2010**, *28*, 223–262. [CrossRef]
56. Chung, H.; Jeong, B.H.; Min, B.J.; Choi, Y.; Cho, B.H.; Shin, J.; Kim, J.; Sunwoo, J.; Park, J.; Wang, Q.; et al. A 58 nm 1.8V 1Gb PRAM with 6.4MB/s program BW. In Proceedings of the 2011 IEEE International Solid-State Circuits Conference, San Francisco, CA, USA, 20–24 February 2011; pp. 500–502.
57. Xu, Z.; Sutaria, K.B.; Yang, C.; Chakrabarti, C.; Cao, Y. Hierarchical modeling of Phase Change memory for reliable design. In Proceedings of the 2012 IEEE 30th International Conference on Computer Design (ICCD), Montreal, QC, Canada, 30 September–3 October 2012; pp. 115–120.
58. Choi, Y.; Song, I.; Chung, M.H.P.H.; Chang, S.; Cho, B.; Kim, J.; Oh, Y.; Kwon, D.; Sunwoo, J.; Shin, J.; et al. A 20 nm 1.8V 8Gb PRAM with 40 MB/s program bandwidth. In Proceedings of the 2012 IEEE International Solid-State Circuits Conference, San Francisco, CA, USA, 19–23 February 2012.
59. di Ventra, M.; Pershin, Y.V. On the physical properties of memristive, memcapacitive and meminductive systems. *Nanotechnology* **2013**, *24*, 25. [CrossRef]
60. Raoux, S.; Xiong, F.; Wuttig, M.; Pop, E. Phase change materials and phase change memory. *MRS Bull. Mater. Res. Soc.* **2014**, *39*, 703–710. [CrossRef]

61. Le Gallo, M.; Sebastian, A. An overview of phase-change memory device physics. *J. Phys. D Appl. Phys.* **2020**, *53*, 1–27. [CrossRef]
62. Song, S.; Mutlu, O.; Das, A.; Kandasamy, N. Improving Phase Change Memory Performance with Data Content Aware Access. In Proceedings of the 2020 ACM SIGPLAN International Symposium on Memory Management, London, UK, 16 June 2020.
63. Kushner, L.J.; Sliech, K.W.; Flewelling, G.M.; Cali, J.D.; Grens, C.M.; Turner, S.E.; Jansen, D.S.; Wood, J.L.; Madison, G.M. The MATRICs RF-FPGA in 180 nm SiGe-on-SOI BiCMOS. In Proceedings of the IEEE RFIC Symposium, Phoenix, AZ, USA, 17–19 May 2015; pp. 283–286.
64. Trimberger, S. Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology. *Proc. IEEE* **2015**, *103*, 3. [CrossRef]
65. Twigg, C.; Gray, J.D.; Hasler, P. Programmable floating gate FPAA switches are not dead weight. In Proceedings of the IEEE International Symposium on Circuits and Systems, New Orleans, LA, USA, 27–30 May 2007; pp. 169–172.
66. Kucic, M.; Hasler, P.; Dugger, J.; Anderson, D. Programmable and adaptive analog filters using arrays of floating-gate circuits. In Proceedings of the 2001 Conference on Advanced Research in VLSI. ARVLSI 2001, Salt Lake City, UT, USA, 14–16 March 2001; pp. 148–162.
67. Hasler, J.; Kim, S.; Adil, F. Scaling Floating-Gate Devices Predicting Behavior for Programmable and Configurable Circuits and Systems. *J. Low Power Electron. Appl.* **2016**, *6*, 1–3. [CrossRef]
68. Hasler, J.; Wang, H. A Fine-Grain FPAA fabric for RF + Baseband. In Proceedings of the GOMAC, St. Louis, MO, USA, 23–26 March 2015.
69. Hasler, P.; Minch, B.; Diorio, C. Adaptive circuits using pFET floating-gate devices. In Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI, Atlanta, GA, USA, 21–24 March 1999; pp. 215–229.
70. Shah, S.; Toreyin, H.; Hasler, J.; Natarajan, A. Models and Techniques for Temperature Robust Systems on a Reconfigurable Platform. *J. Low Power Electron. Appl.* **2017**, *7*, 21. [CrossRef]
71. Shah, S.; Toreyin, H.; Hasler, J.; Natarajan, A. Temperature Sensitivity and Compensation on A Reconfigurable Platform. *IEEE Trans. VLSI* **2018**, *26*, 604–607. [CrossRef]
72. Hooper, M.; Kucic, M.; Hasler, P. 5v-only, standard 0.5 µm CMOS programmable and adaptive floating-gate circuits and arrays using CMOS charge pumps. In Proceedings of the 2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No. 04CH37512), Vancouver, BC, Canada, 23– 26 May 2004; pp. 832–835.
73. Rumberg, B.; Graham, D.W.; Navidi, M.M. A regulated charge pump for tunneling floating-gate transistor. *IEEE Trans. Circuits Syst. I* **2017**, *64*, 516–527. [CrossRef]
74. George, S.; Hasler, J.; Koziol, S.; Nease, S.; Ramakrishnan, S. Low-power dendritic computation for wordspotting. *J. Low Power Electron. Appl.* **2013**, *3*, 78–98. [CrossRef]
75. Koziol, S.; Brink, S.; Hasler, J. A neuromorphic approach to path planning using a reconfigurable neuron array IC. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2014**, *22*, 2724–2737. [CrossRef]
76. Natarajan, A.; Hasler, J. Implementation of Synapses with Hodgkin-Huxley Neurons on the FPAA. In Proceedings of the 2019 IEEE International Symposium on Circuits and Systems (ISCAS), Sapporo, Japan, 26–29 May 2019.
77. Hasler, J. Starting Framework for Analog Numerical Analysis for Energy Efficient Computing. *J. Low Power Electron. Appl.* **2017**, *7*, 1–22. [CrossRef]
78. Hasler, J. Analog Architecture and Complexity Theory to Empowering Ultra-Low Power Configurable Analog and Mixed Mode SoC Systems. *J. Low Power Electron. Appl.* **2019**, *9*, 4. [CrossRef]
79. Moreno, D.G.; del Barrio, A.A.; Botella, G.; Hasler, J. A Cluster of FPAAs to Recognize Images Using Neural Networks. *IEEE TCAS II* **2021**, *68*, 3391–3395. [CrossRef]
80. Theuwissen, A. CMOS image sensors: State-of-the-art and future perspectives. In Proceedings of the ESSDERC 2007—37th European Solid State Device Research Conference, Munich, Germany, 11–13 September 2007; pp. 21–27.
81. Theuwissen, A. CMOS image sensors: State-of-the-art. *Solid-State Electron.* **2008**, *52*, 1401–1406. [CrossRef]
82. Beyne, E.; de Moor, P.; Ruythooren, W.; Labie, R.; Jourdain, A.; Tilmans, H.; Tezcan, D.; Soussan, P.; Swinnen, B.; Cartuyvels, R. Through-silicon via and die stacking technologies for microsystems-integration. In Proceedings of the 2008 IEEE International Electron Devices Meeting, San Francisco, CA, USA, 15–17 December 2008; pp. 1–4.
83. Lhostis, S.; Farcy, A.; Deloffre, E.; Lorut, F.; Mermoz, S.; Henrion, Y.; Berthier, L.; Bailly, F.; Scevola, D.; Guyader, F.; et al. Reliable 300 mm Wafer Level Hybrid Bonding for 3D Stacked CMOS Image Sensors. In Proceedings of the IEEE Electronic Components and Technology Conference (ECTC), Las Vegas, NV, USA, 31 May–3 June 2016; pp. 869–876.
84. Fontaine, R. The State-ofthe-Art of Mainstream CMOS Image Sensors. In Proceedings of the International Image Sensors Workshop, Vaals, The Netherlands, 8–11 June 2015.
85. Tsugawa, H.; Takahashi, H.; Nakamura, R.; Umebayashi, T.; Ogita, T.; Okano, H.; Iwase, K.; Kawashima, H.; Yamasaki, T.; Yoneyama, D.; et al. Pixel/DRAM/logic 3-layer stacked CMOS image sensor technology. In Proceedings of the 2017 IEEE International Electron Devices Meeting (IEDM), San Francisco, CA, USA, 2–6 December 2017; pp. 3.2.1–3.2.4.
86. Kagawa, Y.; Iwamoto, H. 3D Integration Technologies for the Stacked CMOS Image Sensors. In Proceedings of the International 3D Systems Integration Conference (3DIC), Sendai, Japan, 8–10 October 2019; pp. 1–4.
87. Lu, M. Enabling Packaging Architectures and Interconnect Technologies for Image Sensors. In Proceedings of the ASME 2020 International Technical Conference and Exhibition on Packaging and Integration of Electronic and Photonic Microsystems, Virtual, 27–29 October 2020.

88. Oike, Y. Evolution of Image Sensor Architectures with Stacked Device Technologies. *IEEE Trans. Electron Devices* **2021**, *69*, 2757–2765. [CrossRef]
89. Takahashi, T.; Kaji, Y.; Tsukuda, Y.; Futami, S.; Hanzawa, K.; Yamauchi, T.; Wong, P.W.; Brady, F.T.; Holden, P.; Ayers, T.; et al. A Stacked CMOS Image Sensor With Array-Parallel ADC Architecture. *IEEE J. Solid-State Circuits* **2018**, *53*, 1061–1070. [CrossRef]
90. Hasler, J. Opportunities in physical computing driven by analog realization. In Proceedings of the 2016 IEEE International Conference on Rebooting Computing (ICRC), San Diego, CA, USA, 17–19 October 2016; pp. 1–8.
91. Hasler, J.; Kim, S.; Natarajan, A. Enabling Energy-Efficient Physical Computing through Analog Abstraction and IP Reuse. *J. Low Power Electron. Appl.* **2018**, *8*, 47. [CrossRef]
92. Hasler, J.; Natarajan, A. Continuous-time, Configurable Analog Linear System Solutions with Transconductance Amplifiers. *IEEE Circuits Syst. I* **2021**, *68*, 765–775. [CrossRef]