INVITED PAPER

Large-Scale Field-Programmable Analog Arrays

By JENNIFER HASLER, *Senior Member IEEE*

ABSTRACT | Large-scale field-programmable analog array (FPAA) devices could enable ubiquitous analog or mixed-signal low-power sensor to processing devices similar to the ubiquitous implementation of the existing field-programmable gate array (FPGA) devices. Design tools enable high-level synthesis to gate/transistor design targeting today's FPGA devices and the opportunity for analog or mixed-signal applications with FPAA devices. This discussion will illustrate the FPAA concepts and FPAA history. The development of FPAAs enables the development of multiple potential metrics, and these metrics illustrate future FPAA device directions. The system-on-chip (SoC) FPAA devices illustrate the IC capabilities, computation, tools, and resulting hardware infrastructure. SoC FPAA device generation has enabled analog computing with levels of abstraction for application design.

KEYWORDS | Analog-digital integrated circuits, analog integrated circuits, CMOS integrated circuits, field-programmable analog arrays (FPAAs), field-programmable gate arrays (FPGAs).

I. POTENTIAL OPPORTUNITY OF PROGRAMMABLE AND CONFIGURABLE ANALOG DEVICES

The programmability and configurability of digital computation have been the primary capability enabling the decades rise of digital computation. Programmability and configurability enabled one group to build machines and another group to program those machines. The VLSI revolution [1] enabled further separation of roles to address the increasing complexity resulting from Moore's law scaling [2]–[4]. Digital microprocessors (μP) are ubiquitous from embedded applications to general-purpose (GP) computing. Programmability enables changing parameters or coefficients in a particular algorithm. Changing the stored matrix of weights for a vector–matrix multiplication (VMM) is an example of programmability. Configurability enables changing the data flow, topology, as well as the order or operations. Changing the program for an μP is an example of configurability. Field-programmable gate array (FPGA) devices, programmable and configurable gate-level digital devices, enabled digital designers' design capabilities from gate- to system-level designs. FPGAs are ubiquitous digital computing devices found everywhere over the last two decades, arising from their initial conception (1980) and commercialization (mid-1980s) [5].

Modifying the parameters or control flow requires significant changes, such as soldering new components

In contrast to digital computation, analog functionality is considered to be a fixed function. Although digital computation is considered programmable and configurable, where a user can just sit at their laptop and execute many programs, analog computation is believed to require building custom physical hardware (see Fig. 1). Typically, engineers see that digital computing requires writing code, and analog functions require soldering a printed circuit board (PCB). Analog elements could include physical devices, sensors, actuators, or other devices operating over real or integer values. As these structures are the other in most systems, the components that are not programmed such as digital components. The early neuromorphic design began to change this viewpoint utilizing several hand-tuned parameters (see [6]), and it only practically changed with the invention of the first long-term analog memory element in 1994 [7].

Manuscript received December 31, 2018; revised August 6, 2019; accepted October 17, 2019.
The author is with the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA 30332-250 USA (e-mail: jennifer.hasler@ece.gatech.edu).

Digital Object Identifier 10.1109/JPROC.2019.2950173

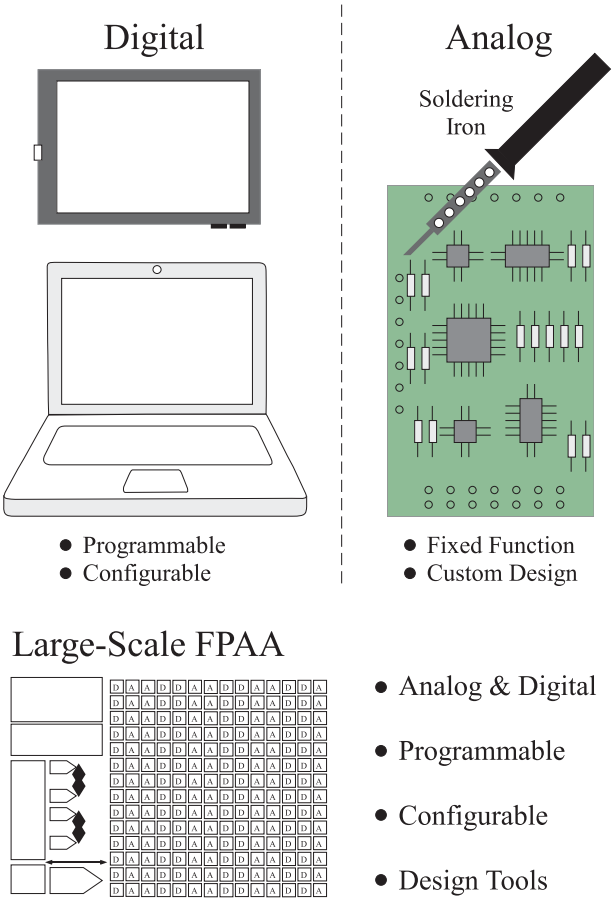


Fig. 1. ***Large-scale FPAAs enable configurable and programmable computation utilizing both analog and digital techniques. Classical digital techniques are ubiquitous and powerful because of parameter programmability and control-flow configurability (e.g., μP and FPGAs). Classical analog techniques are considered fixed and are built custom for a particular application. The typical perception of analog techniques requires significant changes in soldering new components to modifying the parameters or control flow. Dense analog memories enable both programmable and configurable techniques for analog and digital approaches.***

The hope for programmable and configurable analog and mixed-signal devices has been at least as strong if not stronger than the original drive for digital reconfigurability. An equivalent analog and mixed-signal concept could enable the ubiquitous low-power sensor to processing devices. Many perceive analog design as an artistic skill and community; utilizing artists on a large scheduled project requires careful handling. Having design tools, enabling the efficient high-level synthesis of mixed-signal components would greatly improve application opportunities as well as reduce potential schedule risk.

Current programmable and configurable devices (see Fig. 2) have the potential to transform low-power embedded system design, much the way FPGAs transformed physical digital implementations (see Fig. 1). Large-scale field-programmable analog arrays (FPAAs) look toward analog system applications and computation [8], similar to the focus of FPGAs for over 20 years, starting from its introduction in 2002 [9]. Early FPGAs (e.g., 1980s), such as early programmable analog arrays (see [10]–[12]) and early commercial devices (see EPAC [13] or Anadigm [14]), were useful for glue logic and functions and small occasional computations. Analog computing techniques result in 1000× improvement in power or energy efficiency and a 100× improvement in area efficiency, compared to digital computation as Mead originally predicted [15]. For example, an FPAA implemented a command-word acoustic classifier (spectral classification) with hand-tuned weights, achieving command-word recognition in less than 23 μW with standard digital interfaces (see Fig. 2) [16]. The full classification results in less than 1 μJ per classification (or inference), which has 1000× improvement over similar digital neuromorphic solutions requiring roughly 1 mJ or higher for just an inference (see [17]).

This article illustrates the capabilities of these FPAA devices and the opportunities possible with these FPAA devices. Looking over the range, FPAA approaches show a move toward computing and signal processing devices (see Section II), including improving approaches in memory (see Section II-A), architectures (see Section II-B), and process scaling (see Section II-C). One can evaluate a number of metrics for previous, current, and future FPAA device directions, giving a roadmap for scaling these systems to larger architectures (see Section II-D). The system-on-chip (SoC) FPAA concept and family illustrate both the most advanced FPAA family of devices (see Section IV) built to date as well as the most advanced tool and hardware infrastructure (see Section V) developed to date. Any future FPAA device would likely need to build an infrastructure or fit within the existing infrastructure. The existing tool approaches directly extend to larger chips, smaller IC processes, and a number of chips for a given system. The SoC FPAA device family enabling analog system function makes answering questions in analog computing [18] and abstraction, numerics, and architecture complexity (summarized in Section IV) [19]–[21] both possible and necessary for application design.

II. PROGRAMMABLE AND CONFIGURABLE ANALOG DEVICE HISTORY

FPGA and FPAA are combinations of components and connections between these components (see Fig. 3) and off-chip communication. FPGAs could have a number of I/O lines that typically can be programmed to be inputs or outputs (see Fig. 3). FPAA I/O lines could transmit or receive analog or digital signals as well as direct connection lines typical of analog circuits. FPGAs are composed of logic and routing between these devices (see Fig. 3). The logic is referred to as a configurable logic block (CLB) that is typically implemented as lookup tables with flip-flop registers. FPAAs can also have digital logic and routing between digital components. Most FPGAs store the device state using SRAM elements, with a small minority of devices using floating-gate (FG) storage techniques; the

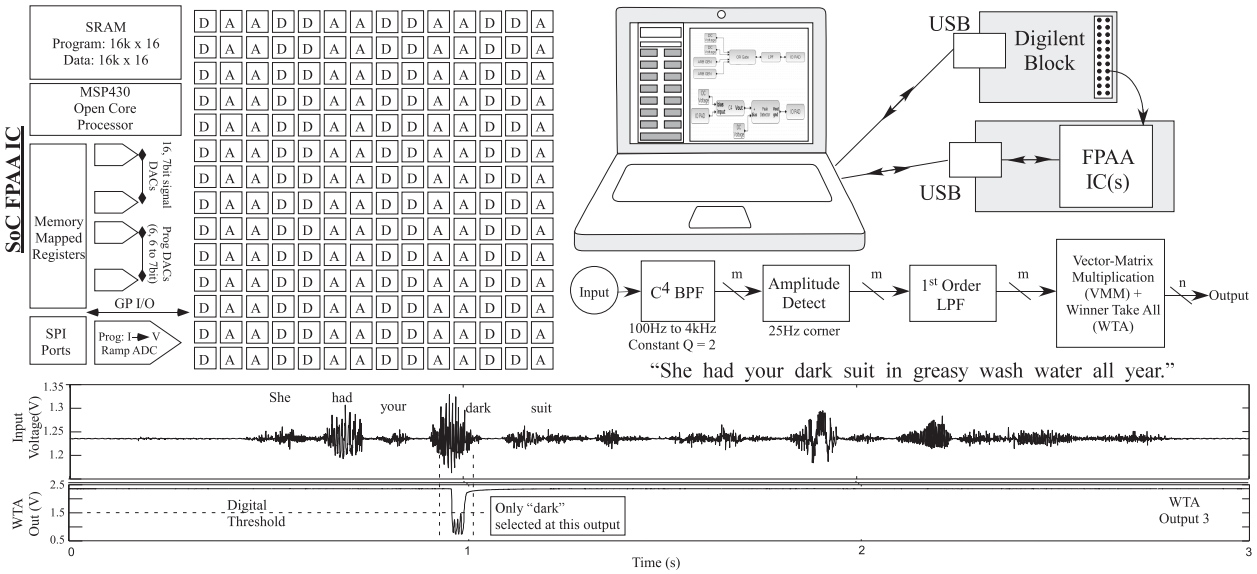


Fig. 2. *SoC large-scale FPAA device showing a command-word speech recognition (spectrum classification). We show the high-level block diagram of the SoC FPAA device (left), a typical measurement setup and computational block diagram for command-word speech recognition, and measured input and classifier output response classifying the word dark in the TIMIT database phrase. The SoC FPAA includes a processor, as well as analog (A) and digital (D) blocks in the routing infrastructure. This analog computation (<23 μW) is radically different from the class of expected analog operations.*

energy required for SRAM storage in modern processes (100 mW–1 W) can limit some FPGA applications.

FPAAs have analog components as well as routing between analog and digital components (see Fig. 3). The routing between analog and digital blocks could occur between the blocks of devices, with converters between these blocks, or more finely connected heterogeneous analog and digital component populations. The components are often organized into regions called computational analog blocks (CABs). CAB components vary considerably between implementations but often include nFET and pFET transistors, transconductance amplifiers [TA or operational transconductance amplifier (OTA)], other amplfiers, passives (e.g., capacitors), as well as more complicated elements (e.g., multipliers), starting from the earliest devices (see [10]–[14] and [22]–[24]). Once FPAA ideas exist, one needs to consider questions of what FPAA memory to use (see Section II-A), of what FPAA architecture to use (see Section II-B), of the impact of FPAA scaling to smaller CMOS linewidths (see Section II-C), and of how FPAA implementations compare (see Section II-D).

A. FPAA Memory Techniques

The memory technology for FPAA devices heavily impacts the application complexity, area, and energy efficiency. The combination of CAB complexity, as well as the memory technology used, categorizes the types of FPAA devices (see Fig. 4). Early FPAA approaches (see [10]–[14] and [22]–[24]), whether from research or commercial sources, utilized SRAMs or similar registers. Analog parameters require a digital-to-analog converter (DAC) for each parameter in one form or another, implemented as ratioed capacitors or transistors. These approaches enabled switching between a few amplifiers or filters, giving the user a few parameters to tune a particular analog function. As a result, these devices rarely reached large-scale configurable systems, and were only used by analog designers because the devices did not have the capabilities to be used at a higher level of integration. These approaches are used as glue components for analog systems and continue to generate commercial interest (e.g., Anadigm) partially as the hope of analog reconfigurable systems.

One SRAM-based FPAA approach reaches a large-scale size for implementing the solutions of nonlinear ordinary differential equations (ODEs) [25]–[27]. The original IC resulted in 400 configurable circuit components (e.g., multipliers and integrators) in 100-mm^2 area (250-nm CMOS) utilizing 16 CABs organized with 25 components in a smaller crossbar array in each CAB. Roughly, one programmable parameter, set by a DAC- or DAC-type structure, is included with each component. An updated CAB with 20 (4×5 array, four integrators) components with additional digital infrastructure to access and reuse the computing infrastructure through DACs, analog-to-digital converters (ADCs), SRAM, and SPI interface in 3.7 mm $\times$ 3.9 mm (65-nm CMOS) area [26]. These devices are used for the solution of general ODEs [25], as well as iterative solutions of linear ODE systems generated from linear partial differential equations (PDEs) [27].

Analog FG devices provide the memory elements for FPAA devices. Section III focuses on FG devices because of their significant impact on FPAA devices, different from the SRAM heavy implementations for FPGA devices. FG elements set the parameters for computational elements

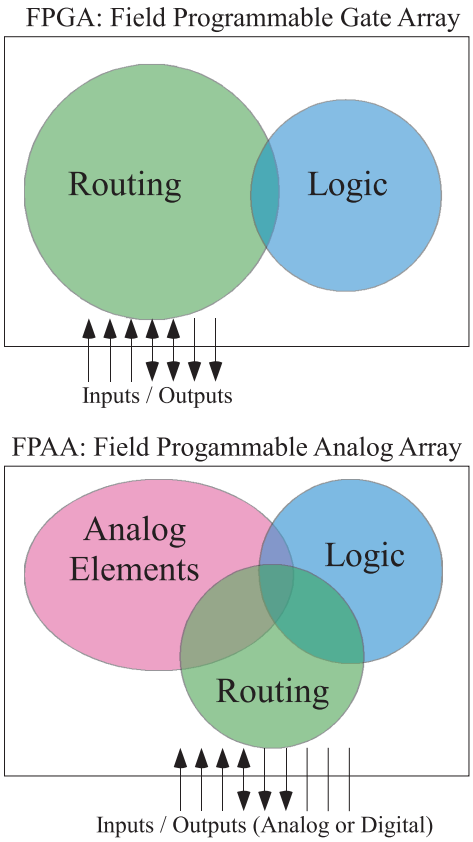


Fig. 3. *FPAA and FPGAs are related forms of configurable technologies, where FPGAs are typically a combination of logic gates and routing, where FPAAs also include analog components in addition to logic gates and routing. These structures can be programmed and reprogrammed several times depending on the user's requirements.*

(e.g., OTAs) while using SRAM or similar digital registers for routing [28]–[32]. Programmable subthreshold and above-threshold current sources are routinely programmed over six orders of magnitude (e.g., 30 pA–30 μA) with better than 1% accuracy at all values, including subthreshold current levels [33]. One recent approach enables FPAA devices targeted for analog at the boundary between analog designers and system designers [30]. FG elements are primarily used for setting current and voltage sources for traditional analog circuits with good performance over the programming range, providing an excellent framework for system-level analog designers to utilize these systems; 80 CABs with roughly 260 circuits are controlled through 296 FG device memories for spectral analysis functions. The required FG programmable voltages are provided through on-chip charge pumps [34].

Analog FG devices provide both the FPAA memory elements and the FPAA routing elements. The chip stores the parameters as well as configuration as nonvolatile analog and digital values. Starting from the earliest of these approaches in 2002 [9], these techniques continued to develop (see [35]–[38]) to the current SoC FPAA [16], utilizing 600 000 programmable parameters in 350-nm CMOS. The fabric switches use a single FG pFET device that can be programmed in an analog manner, enabling computation in routing fabric as well as CAB elements [39]. These techniques enable 1000× improvement in computational energy efficiency compared to custom digital [40], retaining the custom analog computation improvement [41] even though systems are compiled on an FPAA. Section IV discusses further the SoC capabilities and infrastructure.

B. FPAA Architecture Designs

The capability of memory devices for configurable systems has enabled a wide range of FPAA architectures (see Fig. 5). Component and routing architectures distinguish between different FPAA devices, following some similar paths and lessons learned from FPGA devices. Memory elements' crossbar can enable selectivity similar to digital configurability (see Fig. 5), where the routing might be an extended crossbar between all the components and outputs. Early FPAA approaches typically used these schemes (see [9]–[12]).

Another approach uses the computational elements for routing to decrease the area and load capacitance from large crossbar arrays. The classic approach utilized OTAs as the computation and routing in a hexagonal routing pattern [42]–[45]. These approaches have resulted in the highest frequency response for a given IC process by this simple routing structure [42], [44]. Using switches in a crossbar as computation seems to take a related approach to these concepts, utilizing switches for computation as needed [39].

Fig. 5 also shows the routing architecture for a Manhattan routing scheme. Manhattan routing is typical of FPGAs, as well as in some form for modern FPAAs (see [16], [25], and [32]). Manhattan FPAA architecture connects CABs and CLBs through connection (C) and switch (S) blocks. The CABs or CLBs are the buildings, the C blocks enable

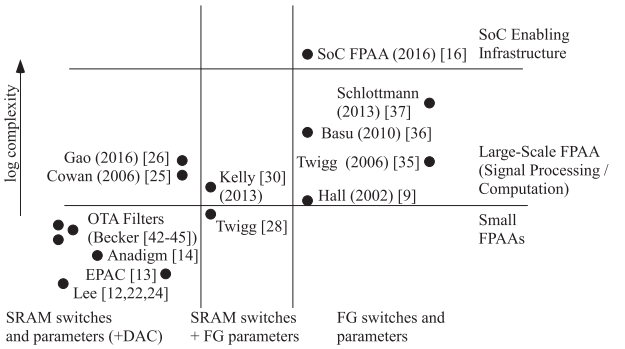


Fig. 4. *Complexity comparison between FPAA implementations. FG parameters and switches allow larger and more complex FPAA structures. Enabling SoC infrastructure with FPAA devices utilizes and manages the larger computation, assisting access to most of the available computation.*

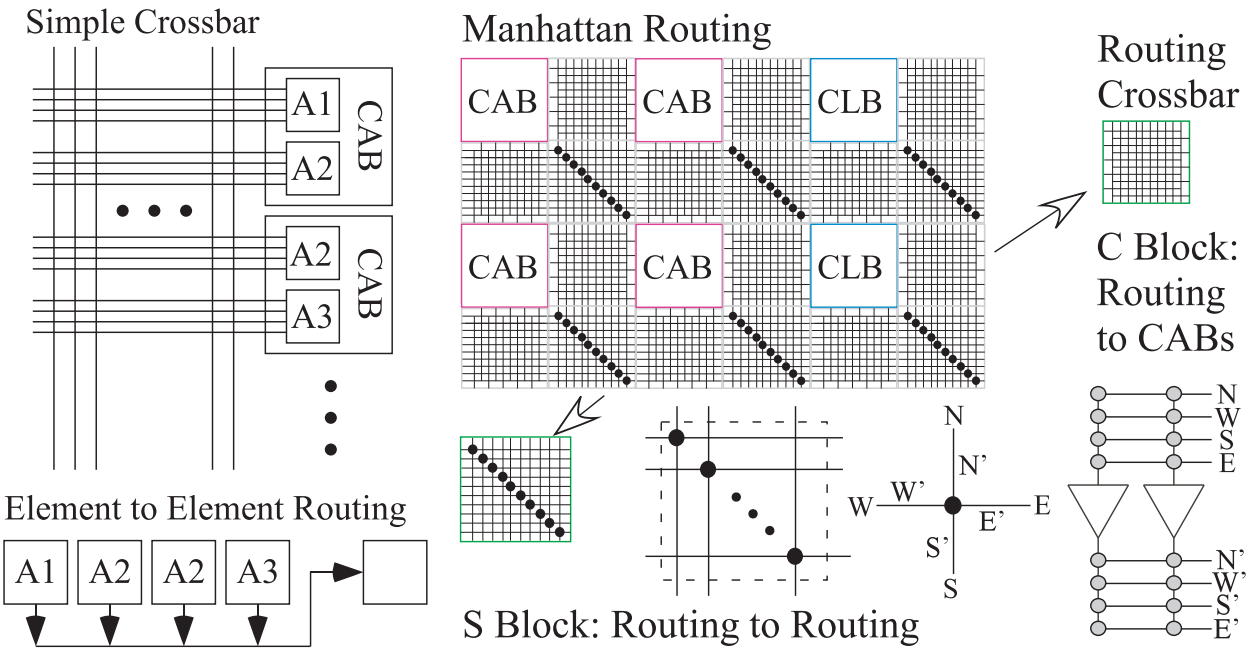


Fig. 5. *FPAA (and FPGA) routing architectures. Simple crossbar: CABs are simply components made up of different configurable elements (A1, A2, ...) (Manhattan, simple crossbar internally). Element-to-element routing: routing through the computational devices, such as OTAs, to potentially decrease routing parasitics. Element-to-element routing is typically arranged in geometrical patterns, hexagonal or rectangular. Manhattan routing: multilevel routing architecture connecting CABs and/or CLBs, which have their own connection patterns (e.g., crossbar), through a set of connection blocks (C blocks) to the higher level interconnection which are routed through the (S blocks).*

the street and street access for the routing, and the S blocks are the intersections between these routing, allowing to go straight or make turns. This routing scheme is typical of the earlier Xlinix architectures (e.g., Virtex 2 or 3). VPR/VTR [46] can place and route for these architectures. These techniques can allow for a number of unique CAB components, such as detailed biologically modeled neurons [47], a technique roughly repeated recently for simpler neurons using digital computation [48].

C. FPAA IC Process and Frequency Scaling

FPAA operating frequency for a particular Manhattan architecture, similar to other architectures, is ideally an inverse as a quadratic function of the minimum IC process linewidth. Decreasing the minimum linewidth quadratically decreases the capacitance, typically resulting in an inversely proportional improvement in energy efficiency and ideally the same improvement in fabric operating frequency. One expects the number of parameters to increase by inverse of the square of the process linewidth. FG-based routing follows the ideal scaling as the FG switches are typically programmed to the maximum conductance value [49]. Fig. 6 shows the operating bandwidth of a Manhattan architecture, similar to the SoC FPAA architecture, based on modeling and experimental data (350, 130, and 40 nm) [49]. This FPAA architecture in the 350-nm process operates at 50-MHz bandwidth, while in 45 nm, this architecture is capable of 4-GHz bandwidth. A 7-, 10-, or 14-nm FPAA design would enable very wide bandwidth RF computation. FG approaches have no apparent limitations in FinFET or silicon on insulator (SOI) although the capacitor structures modify, given the technology capabilities. Therefore, although an FPAA can have a significant performance at a large process node (e.g., 350-nm CMOS), the opportunities only improve in terms of improved bandwidth, higher energy efficiency,

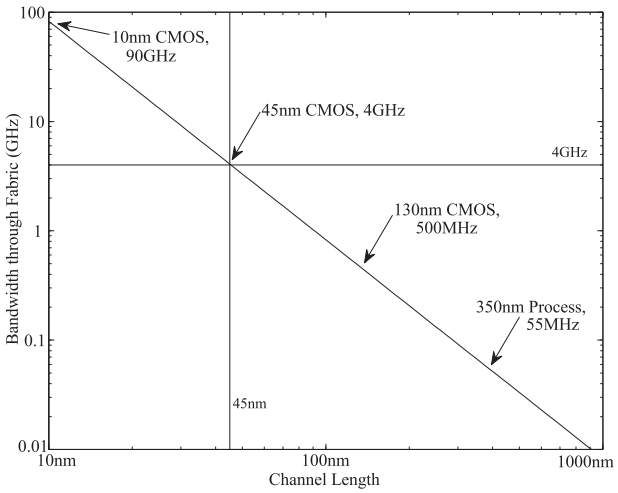


Fig. 6. *Scaling of FPAA architectures using FG device fabric to anchored from data in 350-, 130-, and 40-nm CMOS, bandwidth is from dc to −3-dB corner frequency. Bandwidth of FPAA architectures is a quadratic function of minimum process dimension.*

Table 1 FPAA Comparison Table for Significant and Updated FPAA Devices

Ref.	Process	Area	Num of CAB/CLB	CAB / CLB elements	CAB lines	Num. of parameters	Architecture	Capability
[16]	350nm	84mm^2	98/98	OTAs, Transistors, FG, T-gate multiply, 8, 4-input BLEs	55	359,014	FG, rapid reconfig 16-bit μP, Manhattan	Analog / Digital Systems Full SP processing
[38]	350nm	25mm^2	108/108	2 OTA, 2 Transistor, 2 Tgate 4, 3-input BLEs	20	80,000	FG, Manhattan	Analog / Digital Circuits
[37]	350nm	25mm^2	78/ 0	OTAs, Transistors, FG,	30	76,000	FG crossbar	analog circuits analog SP
[36]	350nm	9mm^2	32 / 0	OTAs, Transistors, FG	40	50,000	FG crossbar	analog circuits, systems
[32]	350nm	25mm^2	64/16	OTAs, BPF, Mult, FF		296	FG, Manhattan	Analog Circuits, Digital interfacing
[25]	250nm	100mm^2	16 / 0 (25 circuits)			416	log-domain (I mode) SRAM, $\approx$ Manhattan	ODE simulation
[44], [42]	130nm	1mm^2	7 / 0	1 Digitally tuned OTAs	4	58	Minimal OTA routing	Basic OTA circuits

FF = Flip Flop, OTA = Transconductance Amplifier, BPF = BandPass Filter, Mult = Multiplier

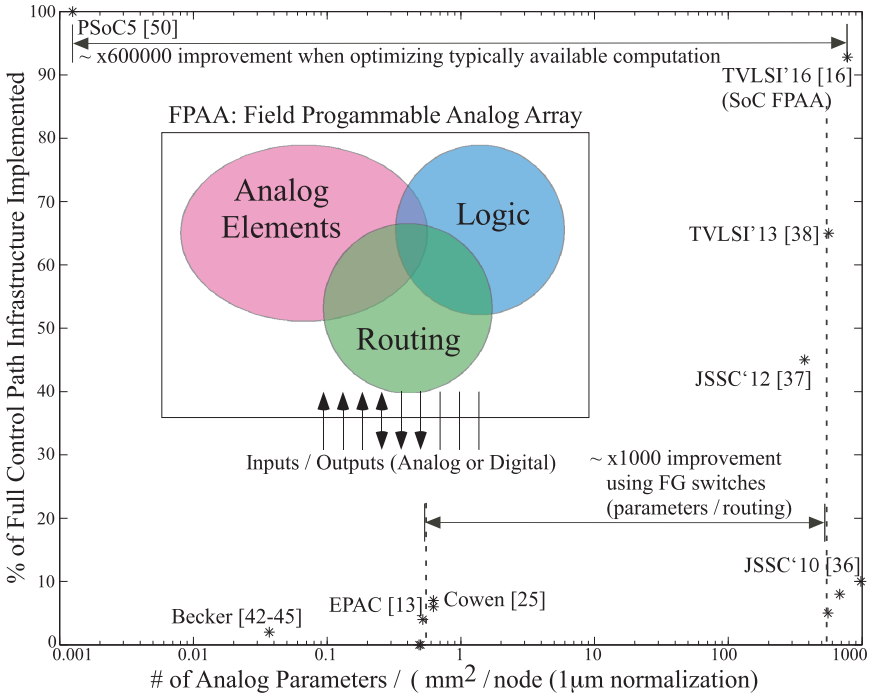


Fig. 7. ***FPAA devices are plotted as the percentage of control path implemented versus analog parameter density. Recent FPAA ICs effectively maximize both parameters. Analog parameter density is the number of analog parameters per mm^2, normalized to a 1-μm process. Analog parameters directly set the complexity possible by the particular FPAA device.***

and higher density similar to the improvements seen in FPGAs.

D. Metric Comparisons of FPAA Devices

Comparing FPAA devices shows the computational possibilities for multiple architectures from experimental data from the current generation of FPAA devices. Table 1 shows another comparison among FPAA devices in a table form, and Fig. 7 plots various FPAA devices showing the percentage of control path implemented versus analog parameter density. Fig. 7 shows the two metrics from many published FPAA devices [9]–[14], [16], [22]–[25], [28]–[32], [35]–[38], [42]–[45], [50]–[58].

Because physical implementations of these FPAA devices show the quadratic scaling of operating frequency with the inverse of minimum channel linewidth for devices from 2.0-μm to 40-nm CMOS (see Section II-C), we normalize the metrics by this parameter. We define analog parameter density as the number of programmable parameters per mm^2, normalized to a 1-μm CMOS node. Analog parameter density determines critically the IC computation complexity, particularly when using routing as computation. Fig. 7 shows that the FG-based FPAAs enable $\approx$1000 parameter density improvement, particularly when used for routing, as will be discussed further in Section III providing increased computation on a single device.

An FPAA should have a large number of programmable parameters, as well as having the infrastructure to get data communicated to these processing devices. Our second metric describes the amount of control flow (mostly digital) relative to the amount of analog and digital data

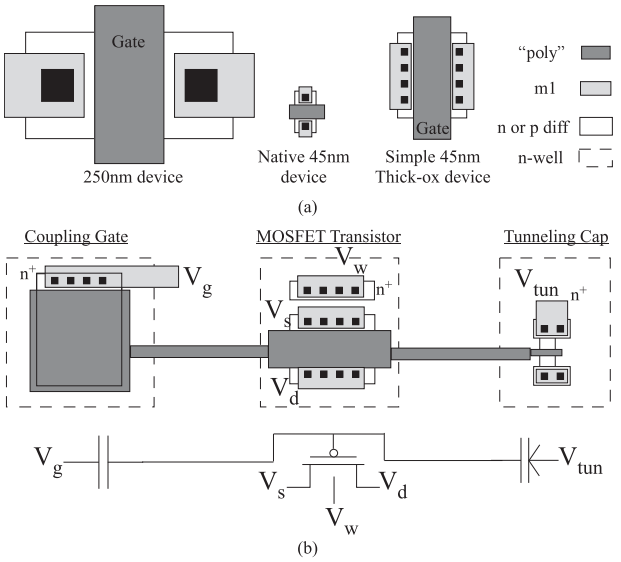


Fig. 8. ***Single-poly cross-section typical for FG devices. Double poly is used as available, but this device is available in a wide range of IC processes (e.g., 130 and 45 nm [49]). Practical devices often have additional process-dependent modifications.***

flow capability. Getting data to all the processors can be a primary limitation for a series of application spaces, such as image processing, where data do not always arrive in the desired order for the computation. Recent RASP-based FPAA designs (see [16], [26], [37], and [38]) have started to focus on improving this second metric. The SoC FPAA is a strong example (discussed further in Section IV) optimizing both metrics; the SoC FPAA is nearly 600 000× parameter density improvement to the closest high utilization structure (i.e., PSoC5) [50].

III. FG DEVICES AS MEMORY AND COMPUTING ELEMENTS FOR FPAA DESIGN

FG devices will be considered briefly in the context for FPAA device design, given its huge impact on these architectures. The original FG circuit concept in the standard CMOS process [7] enabled nonvolatile parameter storage, computed using that parameter, and used the computation potentially modifying (or learning) the long-term parameter. These concepts were built on a long history on FG device development, starting from the initial discovery of an MOS FG device [59], [60]) and early uses of FG devices to store analog quantities (see [61] and [62]). These devices were the original crossbar computation. Other nonvolatile devices with the same analog programmability, selectivity, and density could make a similar impact, although non-Si substitute technology seems unlikely to satisfy these requirements in the near-term future (see [63]).

Fig. 8 shows the top-level (e.g., layout) generic view of a single-poly (standard CMOS) FG device. This core structure is used in every FG test structure since it characterizes baseline performance of these devices, starting from its initial introduction [64]. The transistor gate is capacitively coupled by one or multiple capacitors. Recent ferroelectric capacitors further enable FG circuit capabilities (see [65]); FG techniques can utilize any process improvements. The FG charge is modified by the combination of transistor hot-electron injection and electron tunneling through a separate tunneling capacitor (see [33]). The tunneling and hot-electron injection voltages (e.g., 12 and 6 V, respectively, for 350-nm CMOS) are easily handled through the process (see [33]) and can be generated on-chip using charge-pump circuits [34]. Handling high-voltage signals, signals higher than the applied external power supply, can be easily handled on-chip. High-voltage handling can sit with precision analog circuits, including not affecting the long-term behavior of these FG analog circuits. Decades of FG circuit designs that include memory devices demonstrate this high-voltage design.

With typical ESD protection I/O pads, even if higher voltages for non-FG programming pins are applied, the overall IC can be designed to not be affected by these higher voltages. Practical devices have additional improvements; some are process dependent (e.g., covered by NDAs). Process nodes below 350-nm CMOS use the thicker insulator for all devices, including pFETs. Process nodes below 65 nm use thicker HfO_2 insulators for MOSFETs, including pFETs, including for bulk, SOI, and FinFET

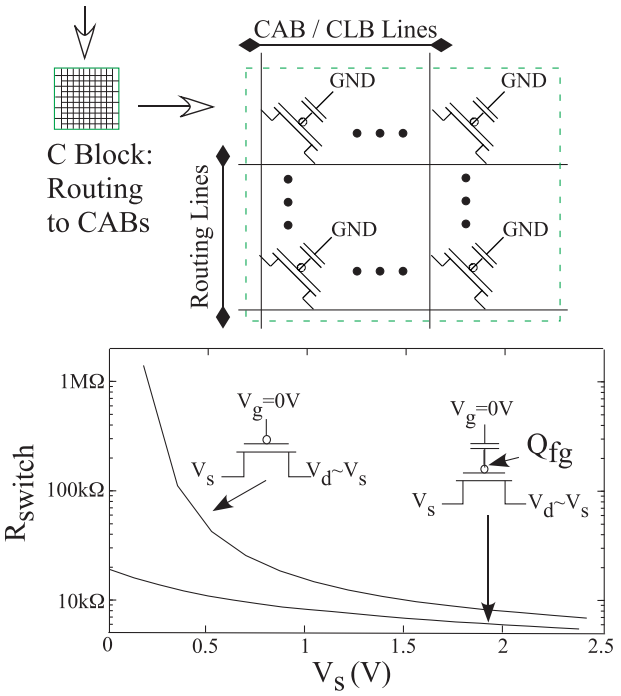


Fig. 9. ***FG switches in the connection (C) blocks, the switch (S) blocks, and the local routing are a single pFET FG transistor programmed to be a closed switch over the entire fabric signal swing of 0–2.5 V [58].***

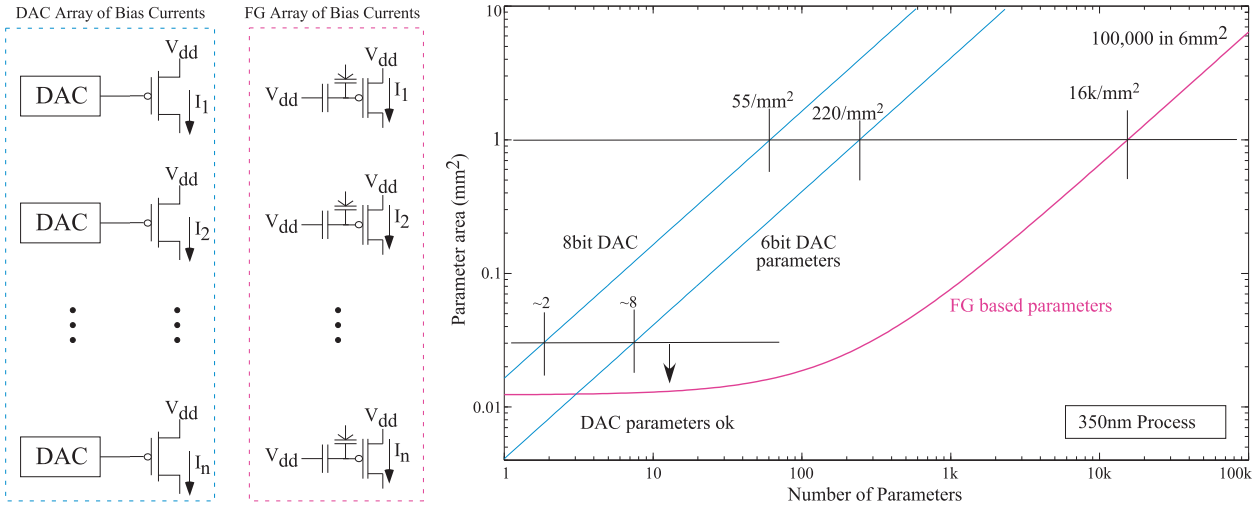


Fig. 10. ***Using FG parameters results in significantly higher parameter density (100× or larger). We compare between FG parameters and its next closest solution, having an n-bit DAC at every device. Optimistically, a DAC grows by a factor of 2 for an increase of 1 bit. At 8-bit DAC precision, 100 FG parameters are smaller than 1 DAC for 350-nm CMOS. We assume an increase for a DAC of 2× for 1 bit. Typically, the cost will increase at a higher level. Handling mismatch is a key risk for any analog (as well as digital) system; only programmability makes analog computation practical in a system (including high-precision ADCs).***

devices. Sometimes this thicker insulator device is used for I/O devices, so the IC directly interfaces to board-level infrastructure. Economic constraints strongly encourage multiple gate insulator thicknesses. One should never put contacts on the FG node to avoid lower retention rates.

FG devices have demonstrated long-term (ten-year lifetime) across multiple IC processes from 2-μm to 40-nm linewidths [66]–[68] as well as have shown precision targeted (re)programming of heterogeneous arrangements of FG devices (see [33]); FG devices have been used for high-matching analog circuits [69], including references [67], amplifiers [66], sensor interfaces [70], filters [71], [72], and data converters [73]–[75], enabling dense high signal-to-noise ratio (SNR) devices. FG devices have demonstrated multiple commercially qualified and sold devices. FG devices have been used in acoustic [76] and imaging [77], [78], utilizing energy-efficient (1000× versus custom digital) signal processing [79] as VMM [41], filterbanks [71], Gaussian mixture models (GMMs) [80], support vector machine [81], VMM+Winner-Take-All (WTA) classifiers [82], and adaptive filters [83].

A single FG pFET can approximate an ideal switch in FPAA routing crossbar (see Fig. 9). One might be surprised that a single pFET operates as a good switch, because of traditional wisdom states that an nFET can only pass lower signal values, while a pFET can only pass higher signal values. Transmission (T)-gates use a parallel combination of an nFET and pFET with a CMOS inverter for a good switch throughout all power supply rails. A T-gate is programmed digitally, either ON or OFF, controlled by a stored digital value. Digital storage limits the computing opportunities of this T-gate switch.

A single pFET device is a good switch over the entire operating range, because the FG voltage can exist above or below the power supply rail. The FG pFET is a standard pFET device whose gate terminals are not connected to signals except through capacitors (e.g., no dc path to a fixed potential). With no dc path to a fixed potential, stored FG charge results in an FG voltage that can be inside or outside the power supply rails. The maximum FG voltage is limited by the programming scheme for the device [58]. To program an OFF-switch, the pFET FG gate is set well above V_{dd}, setting the transistor in accumulation, effectively conducting no current (e.g., <1 pA) throughout the entire operating range [84]. To program an ON switch, the pFET FG gate is set well below GND, setting the transistor above threshold throughout the entire operating range [84]. Fig. 9 shows the measured ON-switch resistance for a 2.5-V supply for a pFET with its gate at GND and an FG pFET with the FG programmed below GND. This maximum ON conductance is roughly independent on process minimum channel length. The conductance is set by velocity saturation of electrons/holes for the MOSFET channel [49]. These devices allow for nearly ideal switches, including in a crossbar array configuration (see Fig. 9), and do not constrain the FG voltage that can be programmed between the ON and OFF states. Although one might consider the nonlinear behavior if using this switch as a resistor, typically the resistance is significantly smaller than other circuit elements. One typically can ignore these nonlinear behaviors and often can ignore the resistances entirely.

FG elements provide a dense analog nonvolatile parameter integrated within a computational fabric, a memory element for both parameters and routing. The alternative to using FG memory elements is using a DAC or DAC device (e.g., capacitor bank) for every parameter. Dynamically storing voltages on a capacitor and refreshing from a

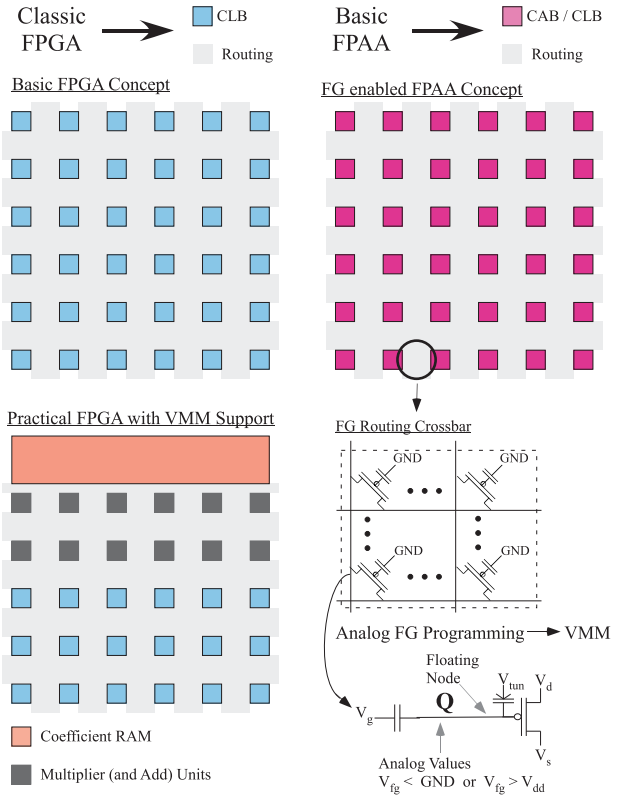


Fig. 11. *FPGA and FPAA computing architectures. A basic FPGA includes CLBs and routing to connect the CLBs and I/O for a particular computation. Larger FPGAs are more structured forms of these blocks. Practical FPGAs are only efficient with VMM support for signal processing and matrix algebra computations (e.g., deep NN), implemented through specialized rows of multipliers (often with adders) and specialized local memory for cycling through required VMM coefficients. These additions improve many FPGA computations while eliminating area for additional CLBs. A basic FPAA also includes CABs or CLB and routing. At a high level, the two approaches look similar. A closer look shows that the FG-enabled routing crossbar is excellent switches, as well as enabling VMM in the routing as a result of analog programming. Unlike FPGAs, all switches in some FG-enabled FPAA devices are potential places of computation. For FPAAs, switches are not dead weight. FG stores a charge, Q, at the floating node, allowing storage of analog voltages that can be inside or outside the power supplies (GND and V_{dd}).*

single DAC results in higher complexity and energy for these operations. Fig. 10 shows the significant opportunities of programmable FG analog concepts compared to alternative approaches, a DAC for every parameter. The DAC approach is a set of ratioed current source transistors (and similar to ratioed capacitors), where the number of devices doubles for each increasing bit to minimize mismatch; in practice, the situation is practically less favorable to the DAC design. The FG area requires infrastructure [33] for FG programming. If one only needs 2–8 parameters, then 6–8-bit DAC area is similar to the resulting FG area. FPAA devices practically require thousands, if eventually not millions to billions of parameters. For 1-mm^2 die area in 350-nm CMOS, 220 6-bit DACs take a similar area to 16 000 FG parameters (see Fig. 10). The FG device in this process is capable of far higher precision (>14 bit [33]). FG device comparison only improves for scaled-down technologies. Fig. 7 shows that the FG-based FPAAs enable ≈1000 parameter density improvement, providing increased computation on a single device.

Using FG parameters explains the 1000× advantage of using FG devices for parameters and routing (see Fig. 7). Only using FG for parameters, and not routing, improves the density metric from 0.55 parameters per normalized mm^2 using DACs [25] to 2.8 parameters per normalized mm^2 using FG parameters [32]. The full advantage is not apparent because of the large amount of resulting routing that only connects devices but is not useful for computation. On average, utilizing FG routing improves parameter density by 200× in area.

These FG pFET retains its analog programming range, unlike T-gate switches, not constraining analog computation in the crossbar array (see Fig. 9). Because the array FG pFET can be programmed anywhere between the OFF and ON states, the resulting device still is an operational transistor for multiple uses, including a current source, cascode element, or a resistor. These FG pFETs are integrated into a crossbar network, typical of VMM topologies [40]. One effectively gets crossbar computations, originally described in FG devices [7]. for free in an FPAA approach. The FPAA computes in the colocated memory space of switches.

Computation in FPAA fabric represents a dramatic departure from classical FPGA architectures. (see Fig. 11).

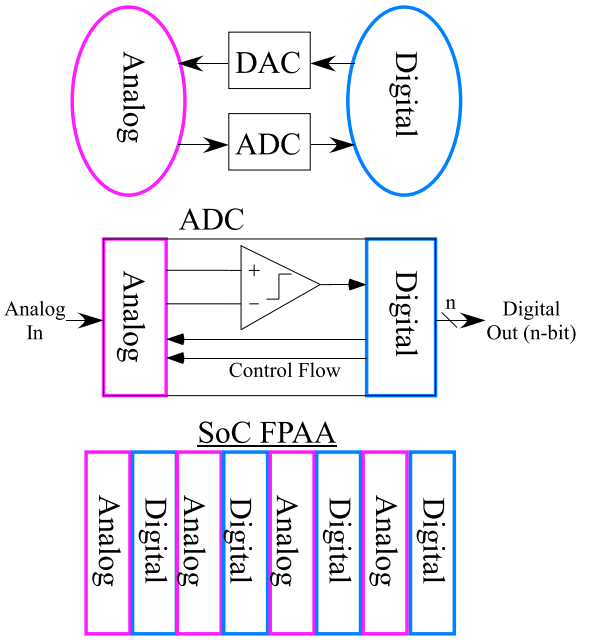
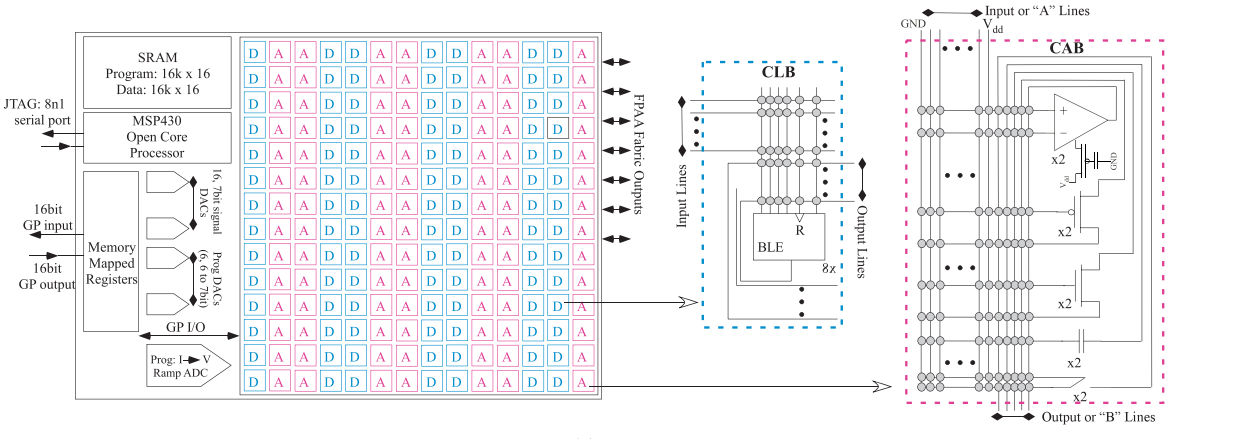


Fig. 12. *Analog and digital computation are typically separated through data converters. Reinvestigating this assumptions shows many systems that are a combination of analog and digital, including data converters. Recent FPAA ICs, including the SoC FPAA IC, enable utilization of analog and digital configurable components, where components are positioned near each other.*



(a)

(b) SoC FPAA IC Parameters

Parameter	Value	Parameter	Value
Number of CABs	98	Number of CLBs	98
On Chip μP	Open Source MSP430	μP clock frequency	0 - 50MHz
C block Line Capacitance	160fF	S Block Line Capacitance	38fF
V_{dd} (analog)	2.5V	V_{dd} (digital)	2.5V, 3.3V
V_{dd} Injection	6.0V	V_{dd} Tunneling	12V
Program Memory	16k x 16	Data Memory	16k x 16
IC Process	350nm CMOS	Die Size	12mm x 7mm
General Digital I/O	16 (in), 16(out)	SPI ports	5
General Analog I/O	125	Analog Parameters	359,014

(c) Summary of Analog and Digital Complexity

Measured System	CAB (devices)	CLB (devices)
C^4 + LPF + Amplitude Detect	1 CAB (4 OTAs, 1 cap)	
Digital block	0	1 CLB (1 BLE)
Ramp ADC	1 CAB (1 OTA, 1nFET, 1 switch, 2 fabric pFETs)	3 CLBs (24 registers, 1 BLE)
Sigma-Delta Modulator	1 CAB (4 OTAs, 1 switch)	(no digital reconstruct)
VMM + WTA (XOR)	3 CABs (1 per WTA input, VMM in local routing)	
Command Word Classifier	12 CABs (filterbank) + 8 CAB (WTA, 3 transistors, routing))	

Fig. 13. ***SoC FPAA IC. (a) Functional block diagram illustrating the resulting computational blocks and resulting routing architecture. The infrastructure control includes a μP developed from an open-source MSP 430 processor [86], as well as on-chip structures that include the on-chip DACs, current-to-voltage conversion, and voltage measurement, to program each FG device. Eight, four-input Boolean logic element (BLE) lookup tables with a latch comprise the CLB blocks. Transconductance amplifiers, transistors, capacitors, switches, as well as other elements comprise the CAB blocks. (b) Table of important SoC FPAA IC parameters. (c) Summary of application complexity of analog and digital elements. The chip has 98 CLBs and 98 CABs.***

Many FPGA architectures are optimized for VMM operations as a fundamental computation and signal processing operation. GPU architectures are optimized for similar computations. These FPGAs are specialized with multiplier units as well as specialized memory blocks to enable efficient VMM computations, cycling through memory for different matrix coefficients. In the FPAA with FG switches, the switches are not the dead weight to connect components [39], but they are computing memory elements greatly increasing the potential computation available in an FPAA. The FG FPAA does not require these specializations, but rather VMM computations are computed in routing fabric. The boundary computations of the VMM computation primarily set the architecture complexity, and the VMM is nearly free in as a unique routing pattern. Other computations can be implemented in the routing infrastructure (see [85]). Further routing infrastructure improvements can enable additional computation advantages.

IV. SoC FPAA IC AND REPRESENTATIVE APPLICATIONS

The SoC FPAA IC represents the most complex FPAA and the first FPAA to be a complete SoC [16]. The SoC FPAA interdigitates analog and digital computation at a fine-grain level in the same routing fabric. Classically, one assumes that analog and digital computations are widely separated, communicating through ADCs and DACs (see Fig. 12). Many operations, such as classifiers, require analog and digital logic together. An N-bit ADC is a specialized simple classifier identifying an incoming signal with one of 2^N levels. This classification occurs by utilizing one or multiple comparators using analog inputs and digital outputs (see Fig. 12). Analog computation often requires digital control flow, again typical in ADC concepts. These examples show the need and simplicity of integrating these spaces. From the experience compiling a number of systems in earlier FPAA designs, some FPAA designs started integrating digital infrastructure control [37], as well as interdigitated analog and digital fabric [38]. The SoC FPAA fully integrated analog and digital columns, enabling analog and digital signals routed on the same fabric (see Fig. 12).

Fig. 13 shows and summarizes the SoC FPAA IC [16]. This SoC FPAA utilizes a configurable fabric integrating analog (A → CAB) and digital (D → CLB) components, specialized blocks, as well as an on-chip μP, SRAM memory, and digital I/O communication ports [see Fig. 13(a)].

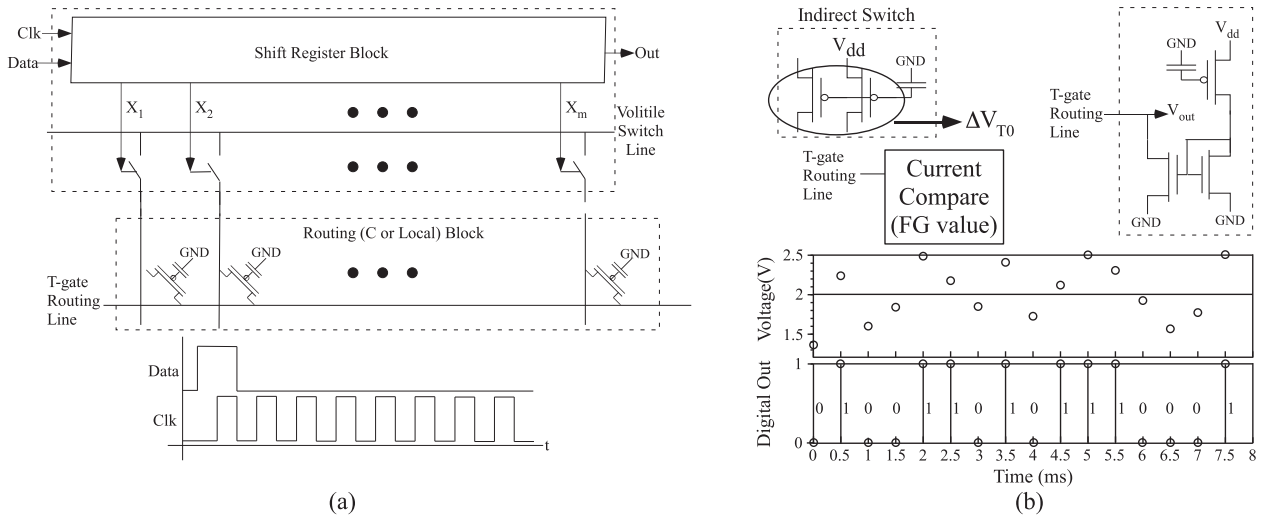


Fig. 14. *Use of T-gates in FPAA fabric for rapid reconfigurability. (a) Circuit diagram of the digital-enabled routing fabric using a set of T-gate switches to dynamically reconfigure the SoC FPAA fabric. (b) Simple circuit compilation using this SoC FPAA fabric for a compiled physically unclonable function (PUF) element. Mismatch in the indirect FG programming infrastructure creates the unique code.*

Fig. 13(b) shows the table of parameters for the resulting SoC FPAA. Analog FG enables both analog and digital functionality within the Manhattan geometry. The CAB devices are typical of earlier CAB designs, being combinations of transistors, OTAs, FG transistors, OTAs, capacitors, T-gates, current mirrors, and signal-by-signal multipliers (see [35]–[37] and [84]). The low-power programmable and configurable FPGA fabric, streamlines the routing of analog and digital signals through a continuous fabric. Fig. 13(c) shows the representative circuits compiled and experimentally measured [16], as well as a summary of the resources used in each case. The hardware platform directly maps to compiler tools (see Section V).

The processor supplements the digital processing system capability and increases overall implementation flexibility; portions of a problem can be mapped to reconfigurable analog, reconfigurable digital, or a GP digital processor. The FPAA employs an open-source MSP 430 microprocessor (μP) [86] with on-chip structures for 7-bit signal DACs, a ramp ADC, memory-mapped GP IO, and related components [see Fig. 13(a)]. The processor is able to send information to and from the array through memory-mapped I/O special-purpose peripherals. These peripherals include 16 memory-mapped 7-bit signal DACs for the architecture, allowing measurements to be performed on-chip, with the data taken by and stored in the processor, as well as additional DACs (and one 14-bit ramp ADC) for the FG programming.

Additional digital control infrastructure in the routing fabric enables rapid reconfiguration of the analog data path. The routing fabric is capable of partial rapid reconfigurability, while using mostly FG devices, by adding an additional set of switch configuration into the fabric. This rapid reconfigurability comes by adding a row of T-gate switches set by a shift register into the switch fabric, originally started in [37]. The I/O lines for the added T-gate row and the shift register signals are available through the routing fabric. These volatile switches are found directly at the interface between the C block and the local interconnect; depending on the desired higher level of abstraction, these switches may be considered as part of either block. One simple application of this technique is enabling a scan chain for either digital or analog circuit debugging. Fig. 14 shows the added routing structure component that enables rapid reconfigurability in the FPAA fabric. These techniques minimize the amount of intermediate data storage required for many computations, enabling data flow techniques for analog processing. Intermediate data storage often requires the largest power and complexity system cost. The rapid fabric reconfigurability can change between programmed aspects in a single clock cycle or asynchronous request–acknowledge loop. SoC FPAA shift register control signals are directed by locally routed signals in the fabric, thus determining the controlling clock (Clk) and data signals [see Fig. 14(a)]. Data stored in the FG fabric would be as optimal as data stored in an off-chip nonvolatile memory without the complexity of loading the resulting computation.

The SoC FPAA, as well as earlier families of FG-enabled FPAAs, demonstrated a number of core concepts. FPAA temperature behavior, modeling, and design [87]–[89] are an essential issue for computation. FG circuits can be programmable and can have weak functions of temperature. FG devices enable directly eliminating mismatch or setting desired targeting values in the configurable structure. These FPAA devices demonstrated many on-chip classifiers sensor-to-output ultralow power classification [16], [82], [90]–[92], embedded machine learning for sensor-to-output classifiers [90], [91], robotics and path planning [93]–[95], image processing [37], and

Table 2 Measured Power Numbers for Compiled Command-Word Classifier Function

Computational Block	Average Power
C^4 block power (12 blocks, exp spaced from 100Hz to 4kHz)	4.86μW
Amplitude detection	3 μW
LPF (12 blocks)	3μW
VMM + WTA block	12.3μW
Total computation Power	23μW
Output signal buffers to outside IC measurement	30μW

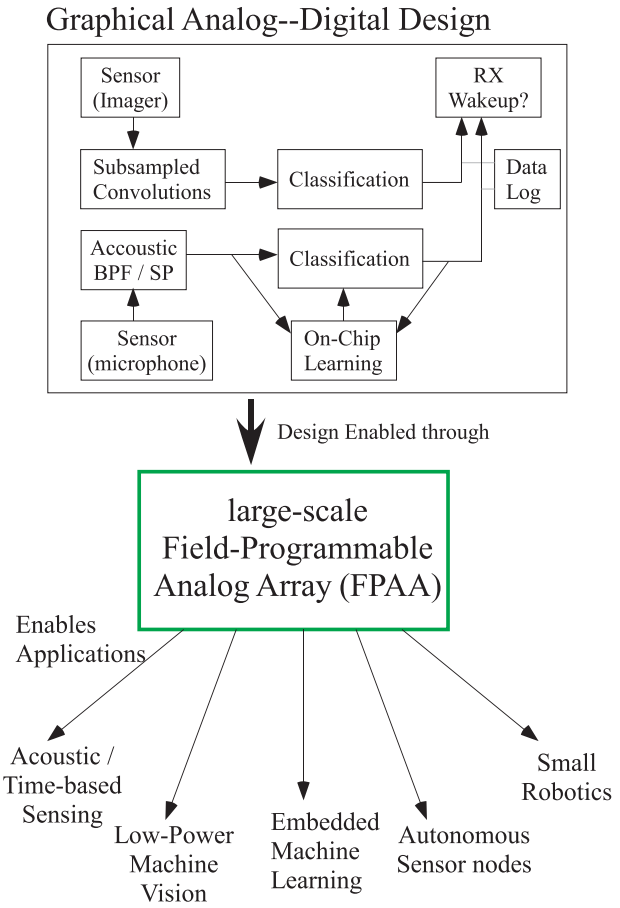


Fig. 15. *FPAA devices provide the ultralow energy/power capability for future embedded system applications.*

low-power biomedical implementations [96]–[98]. The largest existing signal processing functions take a small percentage of the available IC (≈12 CABs and minimal CLB resources). Table 2 shows the energy breakdown for an FPAA device for the application in Fig. 2. FPAA devices as potential applications for embedded Internet of Things (IoT) and remote sensor nodes are capable of secure operation, both in currently demonstrated capabilities such as the implementation of PUF circuits [see Fig. 14(b)] and implementable functions for secure embedded FPAA devices [99].

FPAA integration with sensors, including sensors fabricated on the FPAA fabric [57], [100], opens additional new possibilities. FPAA devices can enable parallel development of sensors and their integration, greatly improving the speed of sensor integration. Custom analog circuit design tends to be the primary limitation for integrated sensor development.

V. SoC FPAA TOOLS AND HARDWARE INFRASTRUCTURE

Ubiquitous use of FPAA devices requires an IC infrastructure and tools, as well as a user base who can utilize these capabilities. The user of these technologies is more likely to be system design and application engineers, individuals who have not obtained a higher degree in analog IC design. A set of user-friendly, high-level tools (e.g., graphical) is enabling chip design to compile FPAA IC for a wide range of energy-efficient applications integrating a number of sensory modalities (acoustic and imaging), potentially for context-aware applications (see Fig. 15). System-level FPAA development requires these capabilities, particularly because designers are used to similar tools for digital-based design. Although these capabilities might have seemed mostly theoretical a decade ago, a reasonable system design time by a wide community requires these capabilities. The success of these tools requires a framework for analog computing [18].

This discussion will focus heavily on the SoC FPAA design tools, infrastructure, and implications [16], because it has developed and reported, by far, the largest amount of tool and infrastructure discussion, to date (see Fig. 16). Fig. 16 shows a high-level view of demonstrated SoC FPAA infrastructure and tools, including FG programming, device scaling, and PCB infrastructure, through system enabling technologies as calibration and built-in self-test methodologies and through high-level tools for design as well as education (see [101]). Sections V-A–V-C discuss

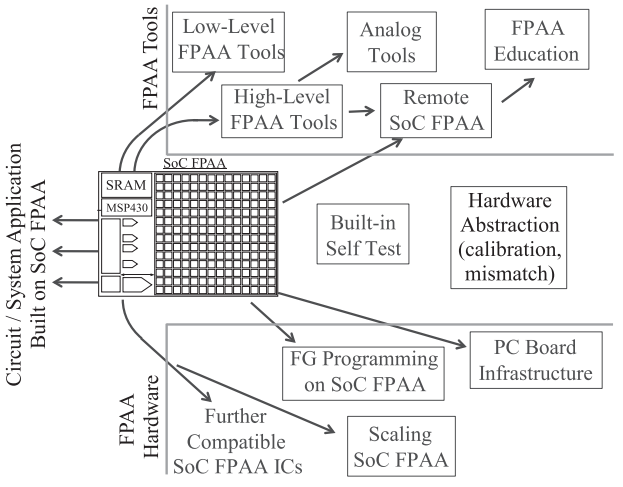


Fig. 16. *SoC FPAA approach consists of key innovations in FPAA hardware, innovations and developments in FPAA tool structure, as well as innovations in the bridges between them. One typically focuses on what circuit and system applications can be built on the FPAA platform, but every solution is built up for a large number of components ideally abstracted away from the user.*

these capabilities, including FPAA tool framework (see Section V-A), FPAA infrastructure (see Section V-B), and FPAA education (see Section V-C). This tool framework is directly applicable to other FPAA devices (see [25], [32], and [45]), and we encourage an open community in these directions. The SoC FPAA tools and PCB infrastructure are openly available as open-source tools.[1] A standard tool and infrastructure platform enables faster utilization and development of next-generation FPAA applications.

A. Analog Tools and the SoC FPAA Toolset

Tools are essential for FPAA system design. While identifying every switch is easier than IC layout, design, verification, fabrication, and testing for analog IC engineers, system designers will expect higher level capabilities. Most of these designers do not want to know about transistors and analog transistor circuits, and yet, the tools must enable efficient use by analog IC designers to enable blocks for application designers. Tools are essential for application-based system design using physical systems, given the modern comfort with structured and automated digital design from code to working application.

Digital design tools for FPGAs are widely accessible. Well-established FPGA design tools include Simulink [102] compilation for Xlinix [103] and Altera FPGAs [104], [105] devices. FPGA manufacturers also have their own toolset for Verilog/VHDL compilation to hardware. Simulink, and to a lesser extent some open-source tools (see [106]), provides the framework to input into Xlinix/Altera compilation tools, completely abstracting away the details from the user, by allowing both standard Simulink blocks to compile to Verilog blocks to targetable hardware, as well as support for specific blocks on that hardware platform.

In contrast, analog design tools have a brief history, including theoretical analog automation tools, as well as early FPAA ICs [107]–[111]. Macromodeling techniques, making a simplified algebraic or numerically simple ODE circuit model, remain a key framework for analog design (see [112] and [113]). Some techniques are coupled with digital tools for joint analog and digital system verification [114], [115]. Labview is a nonopen-source approach for representing analog and digital components that do have some aspects to connect to physical instruments (see [116]) and with an infrastructure that might be adapted to create a similar flow. Custom analog IC design has little additional tool support than low-level IC design tools. Many companies (see [117]) have tried to automate the analog design process but failed because the solution was aimed for analog IC designers who are artistic critics of other analog designs.

Physical FPAA implementations drove the development for analog and mixed-signal design tools, particularly the SoC FPAA implementation, as well as FPAA ICs leading up to the SoC FPAA devices [87], [118], [119]. These tools give the user the ability to create, model, and simulate analog and digital designs. High-level design tools (see Fig. 16) have been essential to the SoC FPAA development [118]. High-level design tools, implemented in Scilab/Xcos, enable automated compilation to a switch list, the description of the programmed FPAA hardware [118]. The tools designed to enable a noncircuits expert, such as a system applications engineer, to investigate particular algorithms. Tools enable system-level design (level $=$ 1) and circuit-level design (level $=$ 2) (see [87]), including both FPAA targeting and simulation [120]. Tools enable physical noise modeling (see [87]) allowing for simulated prediction of the effect of noise on a compiled system as well as the resulting system SNR. The chip details are specified in architecture files for analog-to-digital SoC.

Analog block library is similar to a high-level software definition or library. The graphical high-level tool uses a palette for available blocks that compile down to a combination of digital and analog hardware blocks, as well as software blocks on the resulting processor. The analog Scilab/Xcos system is a visual programming language in the same tradition as Simulink, building on aspects of visual programming languages [121], [122], and data flow languages [123], [124]. Abstracting analog design for system designers increases the chance of automation to be utilized. Graphical algorithms are popular for graphical FPGA tools, such as the recent and independently developed open-source tool, Icestudio [125]. The result is a rich set of analog and digital blocks similar to FPGAs when using graphical design tools (see [102]).

This open-source tool platform creates an integrated environment running in Scilab/Xcos integrating multiple tools, such as x2c, with modified open-source digital place and route (VPR [46]) tools. x2c converts high-level block description by the user to blif format, the input to the modified VPR tool, utilizing vpr2swcs (scilab $\rightarrow$ blif), as well as modified architecture file. The resulting tool uses analog, as well as mixed-signal, library of components. A single Ubuntu 12.04 Virtual Machine (VM) abstracts the entire tool flow from the user, from Scilab/Xcos, device library files, through sci2bliff, vpr2swcs, and modified VPR tools, by simply requiring pressing one button to bring up the entire graphical working toolset.

Tools open the space for abstraction. The multiple levels of analog abstraction in a typical implementation (see Fig. 17) can be abstracted from the designer who only needs to use higher level blocks (blocks in measurement setup of Fig. 17). Fig. 17 shows a typical use of the C^4 block in an acoustic front end for creating subbanded outputs. The core computational chain, C^4 Bandpass filter $+$ Amp Detect $+$ LPF, all compiles into a single CAB. FG elements (e.g., FG-enabled OTA elements), as well as tunable capacitor banks, enable this abstraction and can be tuned around mismatches (see [126]). The abstraction includes computation and testing instrumentation blocks into a single complete compiled system. This measurement

[1]Tools can be downloaded at hasler.ece.gatech.edu.

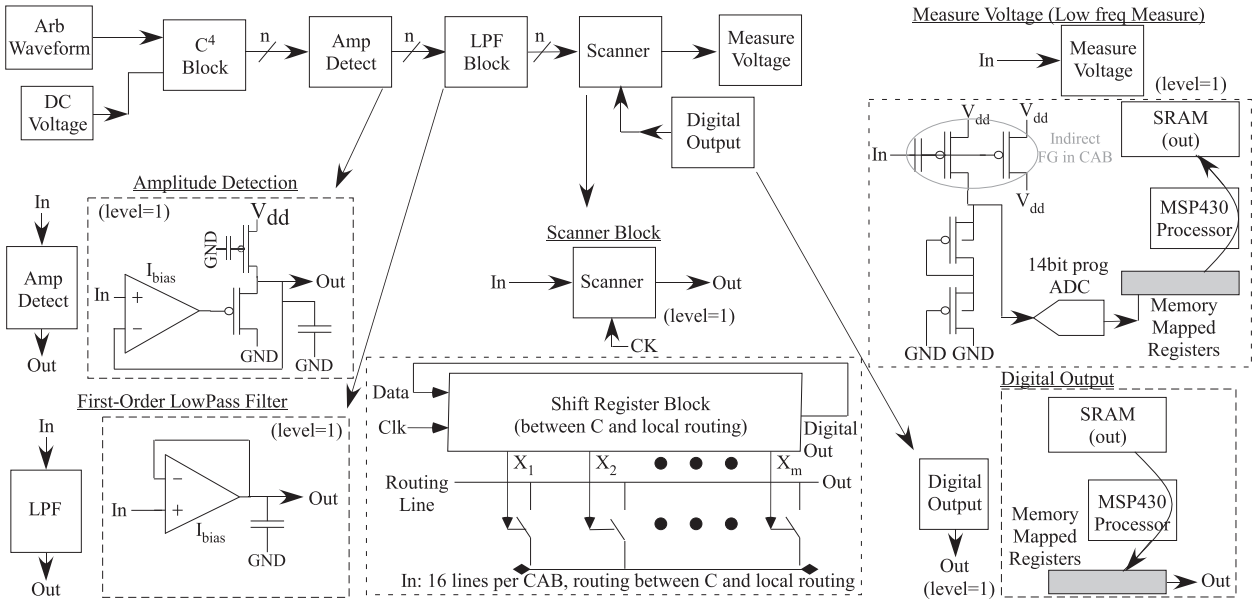


Fig. 17. ***Tool blocks for acoustic subband computation: Bandpass filter bank, amplitude detection, and time window filtering for later processing. Measuring this block introduces the amplitude detection and first-order LPF blocks, both requiring one OTA each. These three elements make up a subband compute block. The structure requires the scanner block, targeted as a set of T-gate switches and shift register in routing between the CABs and C block routing, to multiplex the multiple signals. The scanner is controlled by digital output block taking μP signals into the CAB. The measure voltage block is a low-frequency (200 SPS) block utilizing the 14-bit ramp ADC in the programming infrastructure [33] to connect with the digital system. The approximate gain from In through the ADC is nearly 1 and calibrated on-chip [126].***

illustrates the measure voltage block, effectively a slow-speed (200 SPS), high-resolution (14-bit) voltage measurement. The structure uses the FG programming circuitry, including the 14-bit measurement ramp ADC, still available in the run mode. These blocks abstract further as the subband processing stage as the front end of an acoustic classifier (see [16]).

With the higher level of abstraction, handling the codesign issue between at least analog code, digital code, and μP code becomes immediately apparent. Tools should enable designers to effectively and efficiently design through the large number of open questions in this analog-to-digital codesign space. Digital-only Hardware-Software CoDesign is an established, although unsolved and currently researched, discipline (see [127]–[130]); incorporating analog computation and signal processing adds a new dimension to codesign.

B. SoC FPAA Hardware Infrastructure

Infrastructure is essential for FPAA system design, as a critical aspect to use the IC developments and tool capabilities. Treating infrastructure design with the same attention as IC design eliminates unnecessary bottleneck for a user base, as well as the original designers, to innovate using these devices. The goal of this section is to illustrate the range of infrastructure possible with the existing FPAA devices (see Fig. 18), primarily SoC FPAA devices (see [131]), to encourage additional creativity for FPAA users to innovate using these devices.

Programming the FPAA IC sets the software and hardware infrastructure for the FPAA board [see Fig. 18(a)]. A simple structure facilitates user integration and adoption. The SoC FPAA programs the FG elements using the μP. A single data stream is downloaded to the processor that includes processor object code and programming data (e.g., the switch list from tool compilation); the final processor data are its operational code after programming. The structure is simple enough to enable simple downloading code libraries (e.g., python and Java) to stream the device data as well as the TCL framework used by the design tools. The primary hardware infrastructure is μP IC controlled 6- and 12-V charge-pump ICs, as shown in Fig. 18(a). Charge pumps for FPAAs, as already demonstrated [30], reduces the board infrastructure further. A history of on-chip FG programming development is presented in [33], some of it connected with programming FPAA devices.

The rest of the infrastructure should be simple to minimize user challenges as well as complexities, and yet functional enough for the desired range of applications. Fig. 18(a) shows a GP test infrastructure based on earlier GP boards for earlier FPAA devices (see [132]). The board uses a single USB interface for power [see Fig. 18(a)], a simple serial (8n1) debug interface into the chip, and an interface to other serial standards (e.g., SPI). Boards for specific applications will be optimized along with required specifications while benefiting from a GP open reference design.[2] The FPAA board looks like simple digital peripheral using a standard digital

[2]The boards developed at the Georgia Institute of Technology are openly available at http://hasler.ece.gatech.edu.

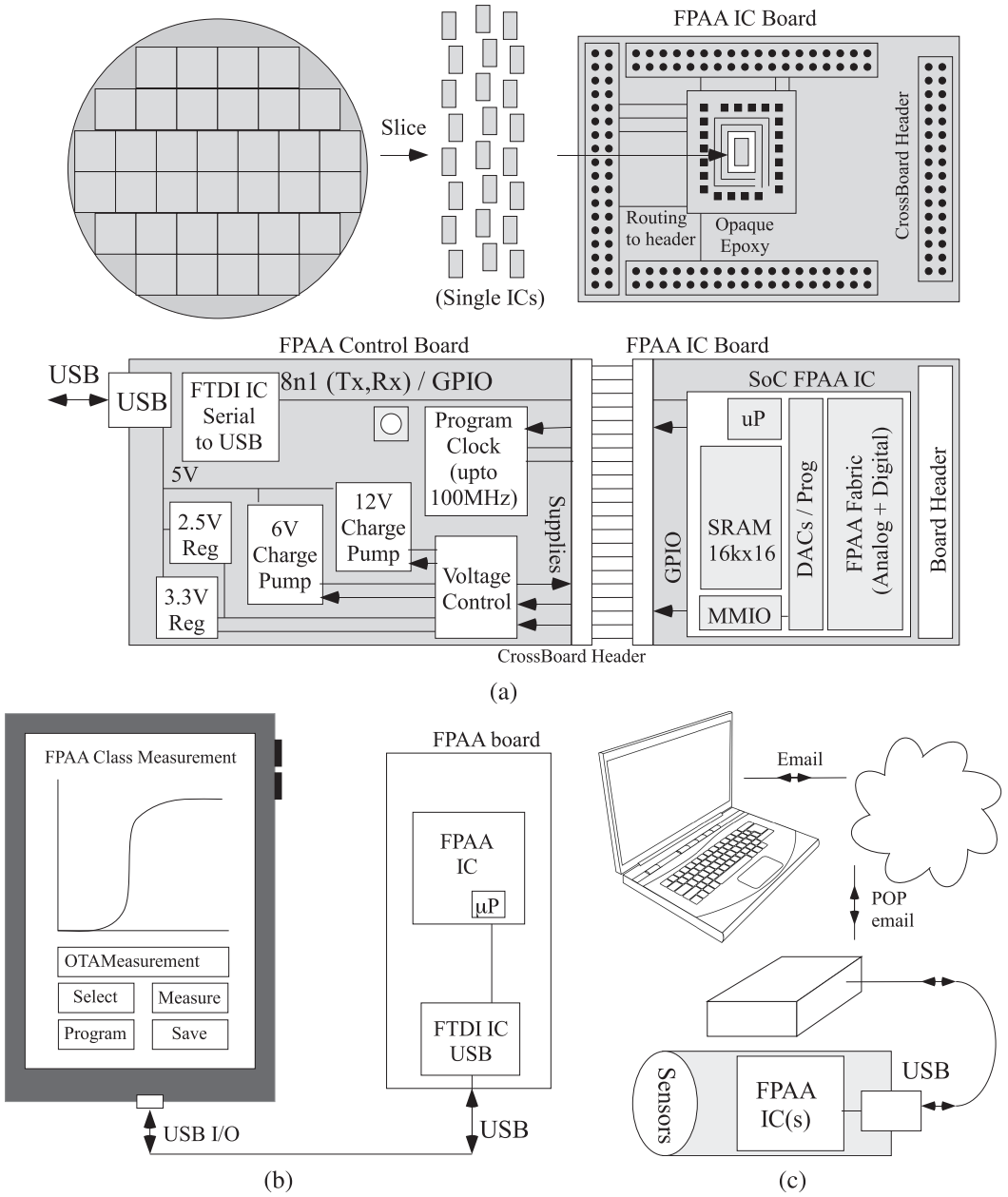


Fig. 18. ***(a) Detailed picture of the SoC FPAA board. Top: process flow for chip-on-board (CoB) build for the resulting IC board. Bottom: detailed block diagram for the SoC FPAA board. The FPAA control board primarily handles the USB to serial communication interface, the programmable clock generator, as well as the multiple supply voltages, controlled by the FPAA, required for operation and programming of the FPAA device. (b) Connecting the FPAA board to Android tablet. An OTG cable is used to handle USB I/O, allowing the device to recognize the FTDI IC's serial port. The tools described in this article allow programs to communicate at a high level with the FPAA µP. (c) Remote test system based on FPAA devices that can be used within our current framework of high-level, open-source Xcos/Scilab tools. With a single button click in the graphical tool, the system will e-mail the resulting targeting code for the FPAA device to a server location, to be picked up by the remote system, which compiles, runs, and then e-mails back the target results.***

communication interface (i.e., USB), allowing minimal (Linux) code for programming and operation for continuous data processing.

The simple structure and range of coding languages to program the FPAA enable a range of user experiences with this FPAA device. FPAA devices can be powered, programmed, and controlled through Android devices [see Fig. 18(b)] and therefore through a device app, using a developed Java code library [133]. The package makes use of the Android API to access the device's serial port, making it easily portable to other devices. FPAA devices can be available as a remote device, controlled through a simple Unix platform using a Python code library [see Fig. 18(c)], operating using a POP e-mail server [134]. The high-level tools are the same for the remote system or in-hand system, where a user simply needs to choose a different button

to "e-mail" the compiled structure, verses to "program" the local device with the compiled structure. An e-mail server enables a relatively stable remote platform capable with nearly zero administrative overhead. These systems go beyond simple one-way updating of FPGA software for fielded devices [135]–[139], enabling user interaction of programming and data analysis. This device illustrates using an FPAA as a small IoT block.

C. SoC FPAA Hardware Impacting Engineering Education

The availability of FPAA devices with an abstracted tool flow and user infrastructure enabled educational opportunities. Education becomes essential to the long-term viability of FPAA opportunities, in much the same way that education was essential to the viability of FPGA of DSP processor opportunities over two decades ago, by empowering generations of students with the knowledge and framework to use these devices. These same students greatly benefit using advanced technology to enable learning about mixed-signal computation at several levels, learning in a physically real system. Using FPAAs in the classroom is a huge application area for any viable FPAA technology. The focus of this section is to overview the FPAA use in educational experiences.

Commercial FPAA devices open a number of academic institutions to develop their own FPAA educational applications. Anadigm's FPAA devices have found their way into a range of applications over two decades. Anadigm FPAA devices have been used for class development (see [140]), as well as part of academic development (see [141] and [142]). Anadigm has sufficient graphical tools for their designs for their small but effective FPAA devices, obtaining functional blocks with 90 dB of SNR [143]. As additional commercial FPAAs are available, the capabilities and opportunities will increase exponentially.

FPAA hardware platforms have been central to hands-on circuit courses at Georgia Tech (GT) for over a decade. Neuromorphic Analog VLSI Circuits (ECE 6435)[3] has utilized FPAA devices since 2005 for its hands-on laboratory experiences (starting with [35]). These devices initially significantly reduced the required significant bench infrastructure and before-class IC fabrication [144], reducing the required generation equipment every few years [145], and eventually eliminated the need for any additional test equipment other than an FPAA board using the SoC FPAA devices [131], [146]. The simple and accessible laboratory framework is a student laptop-based setup using an SoC FPAA device (through USB) and/or remote SoC FPAA device designed through graphical Scilab/Xcos-based tools. The SoC FPAA, and resulting infrastructure, creates a portable student user experience different from any typical laboratory.

[3]The course information is openly available at the website: http: users.ece.gatech.edu/phasler/ECE6435.

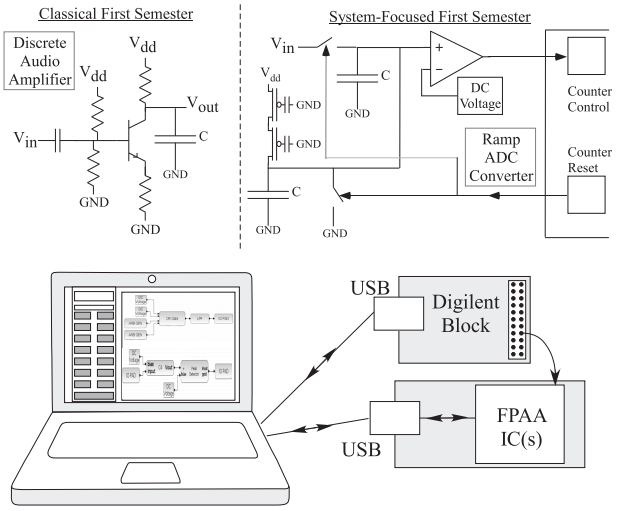


Fig. 19. ***Moving from classical discrete circuit concept toward system-level design, empowered by FPAA devices. A typical classical first junior-level transistor circuits course focuses on learning many forms of a traditional audio amplifier. The focus is on design with discrete parts, where the transistor is the difficult element. The system-focused first junior-level course covers many of the same circuit fundamentals and results in students designing a system component (e.g., a ramp ADC). FPAA concepts empower the ability to move toward a system-level course as well as a hands-on circuits course. The FPAA structure just requires a board connected (through USB) to a student laptop with our open-source VM.***

These techniques moved to other courses, including a first transistor circuit class (see Fig. 19) taken by undergraduate students in their third year (ECE 3400)[4] in Fall 2016 [101], [147]. The course became a hands-on, design, devices-to-systems course, resulting in a significant difference in how students approached circuit design. The classroom implementation only required FPAA boards, without any other technology, specialized laboratory spaces or additional human resources. This study opened a number of undergraduate curriculum questions due to the change in circuit implementation medium from discrete circuit design to configurable mixed-signal hardware [147]. The students heavily utilized remote FPAA system during this course to characterize and design a number of circuits (see Fig. 19). Several groups successfully designed and characterized a ramp ADC, including full low-frequency linearity (INL and DNL). Previous remote test systems that have to spend considerable time in developing their hand-tailored configurable system for educational directions [148]–[153]; the FPAA tools eliminate this issue.

VI. FPAA SUMMARY, NEXT DIRECTIONS, AND LONG-TERM IMPLICATIONS

Current FPAAs have the potential to, as well as started demonstrating the ability to, transform low-power

[4]The course information is openly available at the website: http: users.ece.gatech.edu/phasler/ECE3400.

embedded system design, much the way FPGAs transformed physical–digital implementations. The history of FPAA approaches shows a move toward computing and signal processing applications with sufficient metrics to open these opportunities toward computing applications. FPAAs, such as the SoC FPAA family, illustrate the system-level capabilities, as well as tool and hardware infrastructure opportunities. Design tools enable that design from high-level synthesis to gate/transistor design is the reality for digital applications today with FPGA devices and is appearing for analog or mixed-signal applications with FPAA devices. Large-scale FPAA devices already have the potential to empower ubiquitous analog or mixed-signal low-power sensor to processing devices similar to the ubiquitous implementation of the existing FPGA devices.

A usable programmable and configurable technology (e.g., SoC FPAA [16]) requires maturing analog/physical computing capabilities empowering this disruptive FPAA technology toward commercial opportunities. Physical computing, computing over real values versus integers, includes the space of analog, neuromorphic, quantum, and optical computing, unifying the mutual opportunities between these areas. Physical computing framework's recent development has focused on analog techniques, including starting the analog framework [18], demonstrating and developing analog abstraction and hierarchy [19], the development of analog architecture theory and algorithmic complexity [21], and the development of analog numerical analysis [20].

Related to the development of a physical computing framework, one asks about the SNR of FPAA components and the resulting computation. For simple FPAA devices of isolated components of nontunable SNR with switch matrix connections (e.g., Anadigm), SNR metrics can be reasonably specified and estimated (see [142]). These simple estimates are no longer applicable with advanced FPAA devices (e.g., SoC FPAA) due to programmable currents, potential of fine-grain (e.g., transistor level) compilation, and use of routing fabric for computing and passive elements. A circuit's load capacitance is configurable to a wide range of possible sizes [16]. One can compile an FG OTA amplifier with greater than 1-V linear range (2.5-V supply) with a load capacitance greater than 10 pF, resulting in an SNR (thermal noise) greater than 100 dB ($>$16 bit). One can choose even parameters for some circuits to achieve even larger values. Such a response does not say that every circuit will achieve 16 bit or higher SNR, but rather the SNR of the circuits compiled will depend upon the particular design and only have slight limitations based on the infrastructure. A careful understanding of the question is essential to set reasonable expectations and communication. Typically, the issues that limit the operation of a particular FPAA design involve higher input voltage levels ($V_{\mathrm{dd}} = 2.5$ V) or handling of high-power levels using a particular design; these issues can be handled by analog circuits at the I/O pins or by more specialized FPAA design at the edges or throughout the routing fabric.

One might wonder about the future of FPAA devices given current development, which we will discuss some potential perspectives in the rest of this section as well as directions to reach these opportunities. One can visualize a world where analog and mixed-signal reconfigurability in various forms will be as common as digital reconfigurability today in all of its various forms (e.g., μP, FPGAs, and GPUs). Even when custom analog and mixed-signal ICs are designed in this future, some aspects of configurability will still be used, just as custom digital designs still utilize reconfigurability today.

FPAAs require commercial sources to fully unleash their technological impact. Anadigm's steady commercial market shows the extremely high interest and hope in configurable analog and mixed-signal opportunities, particularly given their heroic efforts with a very limited configurable chip that could be replaced with less than $1 in parts. These chips have already used for initial prototyping and educational projects. Commercialization of SoC-type FPAA ICs represents a generational improvement over the existing capabilities. Many designs can be developed into industrially hardened IC products. Early use of system-level FPAA devices in education and research directions shows numerous initial promising directions.

Families of FPAA devices, similar to FPGA devices, are expected with a number of die sizes and optimizations for energy consumption. Some FPAA devices should have specialized fabrics and components for particular computations, such as embedded machine learning or neuromorphic approaches (e.g., as in early work [47], [154]). Specialized FPAA devices for special voltage and power conditions, such as neural stimulators or RF transmitters, could integrate with general FPAA devices. These generic FPAA devices that can be electronically measured at every node enable a trusted, secure, and legacy-resistant computational platform. The existing FPAA devices can be adapted to secure small embedded network devices [99].

Scaling FPAA designs to state-of-the-art processes enables computational problems beyond what is imagined by current digital computing structures. Scaling opens up a range of new architecture approaches between GPUs and FPGAs in a uniquely mixed-signal manner. Scaling opens FPAA designs to the complexity sizes of current GPUs (e.g., video and image processing), while offering far greater energy efficiency per parallel operation, or the complexity to directly compute a number of PDE computations (e.g., charged particle computations) on a single device. Scaling FPAA designs enable higher density (100$\times$ expected from 350 to 40 nm), lower energy and power (100$\times$ lower from 350 to 40 nm), as well as lower commercial cost per computation from the existing designs. Smaller CMOS linewidths (e.g., 40 nm) also enable RF signals through

routing fabric [68]. The improvements assume the design iterations to optimize the resulting designs. The technology risk in scaling is low, as devices and initial routing fabrics for FG and non-FG structures have been demonstrated in the 40-nm CMOS range (see [26] and [68]). Programmable techniques enable scaled-down analog design, avoiding the difficulties that device mismatch creates for analog IC design in scaled processes. Efforts and resources enabling scaled CMOS FPAA implementations will have many significant future applications.

Finally, one can envision a large community utilizing these FPAA devices, contributing to a number of commercial and open-source communities, empowered through common tool frameworks. These directions require the developing an FPAA user and developer community working around common tools, particularly tools that enable system compilation including handling analog-to-digital codesign through device programming and system computation. One can imagine a large user community developing a wide library of analog and digital components for these new devices. Analog component and system reuse will become a standard practice. The tools need to be flexible enough to handle a wide range of architectures, including multiple ICs. These communities will be heavily built through integrating FPAA devices into educational environments, whether in the classroom or in the research laboratory, following along the inspiration from the rise of DSPs and FPGA devices. Educational directions enable a community who can utilize analog signal processing and computing effectively toward commercial opportunities. Even IC design tools should be optimized to rapidly generate FPAA fabric and related infrastructure, as well as optimization tools to take an existing FPAA design to optimize hardware solutions when necessary. Improved tools decrease cost and open opportunities throughout the commercial process. ■

REFERENCES

[1] C. Mead and L. Conway, *Introduction to VLSI System Design*. Reading, MA, USA: Addison-Wesley, 1980. [Online]. Available: http://ai.eecs.umich.edu/people/conway/VLSI/VLSIText/VLSIText.html

[2] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, p. 114, Apr. 1965.

[3] B. Hoeneisen and C. A. Mead, "Fundamental limitations in microelectronics—I. MOS technology," *Solid-State Electron.*, vol. 15, no. 7, pp. 819–829, Jul. 1972.

[4] B. Hoeneisen and C. A. Mead, "Current-voltage characteristics of small size MOS transistors," *IEEE Trans. Electron Devices*, vol. ED-19, no. 3, pp. 382–383, Mar. 1972.

[5] B. Santo, "25 microchips that shook the world," *IEEE Spectr.*, vol. 46, May 2009.

[6] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA, USA: Addison Wesley, 1989.

[7] P. Hasler, C. Diorio, B. A. Minch, and C. A. Mead, "Single transistor learning synapses," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA, USA: MIT Press, 1994, pp. 817–824.

[8] S. Bains, "Analog's answer to FPGA opens field to masses," *EE Times*, no. 1510, Feb. 21, 2008.

[9] T. S. Hall, D. V. Anderson, and P. Hasler, "Field-programmable analog arrays: A floating—Gate approach," in *Proc. Int. Conf. Field Program. Logic Appl.*, Montpellier, France, Sep. 2002, pp. 424–433.

[10] M. A. Brooke, "A reconfigurable general purpose analog integrated circuit," Ph.D. dissertation, Dept. Elect. Eng., Univ. Southern California, Los Angeles, CA, USA, 1988.

[11] M. A. Sivilotti, "Wiring considerations in analog VLSI systems, with application to field-programmable networks," Ph.D. dissertation, California Inst. Technol., Pasadena, CA, USA, 1991.

[12] E. K. F. Lee and P. G. Gulak, "A CMOS fieldprogrammable analog array," *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 1860–1867, Dec. 1991.

[13] H. W. Klein, "The EPAC architecture: An expert cell approach to field programmable analog circuits," in *Proc. IEEE Midwest CAS*, vol. 1, Aug. 1992, pp. 169–172.

[14] "Anadigm: Specifically generic analog functions for FPAAs Anadigm says," *EE Times*, Sep. 28, 2004.

[15] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.

[16] S. George *et al.*, "A programmable and configurable mixed-mode FPAA SoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2253–2261, Jun. 2016.

[17] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," Apr. 2019. *arXiv:1812.01739*. [Online]. Available: https://arxiv.org/abs/1812.01739

[18] J. Hasler, "Opportunities in physical computing driven by analog realization," in *Proc. IEEE ICRC*, San Diego, CA, USA, Oct. 2016, pp. 1–8.

[19] J. Hasler, A. Natarajan, and S. Kim, "Enabling energy-efficient physical computing through analog abstraction and IP reuse," *J. Low Power Electron. Appl.*, vol. 8, no. 4, pp. 1–23, Dec. 2018.

[20] J. Hasler, "Starting framework for analog numerical analysis for energy-efficient computing," *J. Low Power Electron. Appl.*, vol. 7, no. 17, pp. 1–22, Jun. 2017.

[21] J. Hasler, "Analog architecture complexity theory empowering ultra-low power configurable analog and mixed mode SoC systems," *J. Low Power Electron. Appl.*, vol. 9, no. 1, pp. 1–37, Jan. 2019.

[22] E. K. F. Lee and P. G. Gulak, "Field programmable analogue array based on MOSFET transconductors," *Electron. Lett.*, vol. 28, no. 1, pp. 28–29, 1992.

[23] E. K. F. Lee, "Reconfigurable pipelined data converter architecture," in *Proc. IEEE Midwest CAS*, vol. 1, Aug. 1996, pp. 162–165.

[24] S. Koneru, E. K. F. Lee, and C. Chu, "A flexible 2-D switched-capacitor FPAA architecture and its mapping algorithm," in *Proc. IEEE Midwest CAS*, vol. 1, Aug. 1999, pp. 296–299.

[25] G. E. R. Cowan, R. C. Melville, and Y. P. Tsividis, "A VLSI analog computer/digital computer accelerator," *IEEE J. Solid-State Circuits*, vol. 41, no. 1, pp. 42–53, Jan. 2006.

[26] N. Guo *et al.*, "Energy-efficient hybrid analog/digital approximate computation in continuous time," *IEEE J. Solid-State Circuits*, vol. 51, no. 7, pp. 1514–1524, Jul. 2016.

[27] Y. Huang, N. Guo, M. Seok, Y. Tsividis, K. Mandli, and S. Sethumadhavan, "Hybrid analog-digital solution of nonlinear partial differential equations," in *Proc. Micro-50*, Oct. 2017, pp. 665–678.

[28] C. M. Twigg and P. E. Hasler, "An OTA-based large-scale field programmable analog array (FPAA) for faster on-chip communication and computation," in *Proc. IEEE ISCAS*, May 2007, pp. 177–180.

[29] C. Twigg and P. Hasler, "Configurable analog signal processing," *Digit. Signal Process.*, vol. 19, no. 6, pp. 904–922, 2009.

[30] B. M. Kelly, B. Rumberg, D. W. Graham, and V. Kulathumani, "Reconfigurable analog signal processing for wireless sensor networks," in *Proc. IEEE Midwest CAS*, Columbus, OH, USA, Aug. 2013, pp. 221–224.

[31] B. Rumberg *et al.*, "RAMP: Accelerating wireless sensor hardware design with a reconfigurable analog/mixed-signal platform," in *Proc. ACM/IEEE Conf. Inf. Process. Sensor Netw.*, Seattle, WA, USA, Apr. 2015, pp. 47–58.

[32] B. Rumberg and D. W. Graham, "A low-power field-programmable analog array for wireless sensing," in *Proc. ISQED*, Mar. 2015, pp. 542–546.

[33] S. Kim, J. Hasler, and S. George, "Integrated floating-gate programming environment for system-level ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 24, no. 6, pp. 2244–2252, Jun. 2016.

[34] B. Rumberg, D. W. Graham, and M. M. Navidi, "A regulated charge pump for tunneling floating-gate transistors," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 3, pp. 516–527, Mar. 2017.

[35] C. M. Twigg and P. Hasler, "A large-scale reconfigurable analog signal processor (RASP) IC," in *Proc. IEEE CICC*, Sep. 2006, pp. 5–8.

[36] A. Basu *et al.*, "A floating-gate-based field-programmable analog array," *IEEE J. Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, Sep. 2010.

[37] C. Schlottmann, S. Shapero, S. Nease, and P. Hasler, "A digitally-enhanced reconfigurable analog platform for low-power signal processing," *IEEE J. Solid State Circuits*, vol. 47, no. 10, pp. 2174–2184, Sep. 2012.

[38] R. B. Wunderlich, F. Adil, and P. Hasler, "Floating gate-based field programmable mixed-signal array," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 8, pp. 1496–1505, Aug. 2013.

[39] C. M. Twigg, J. D. Gray, and P. E. Hasler, "Programmable floating gate FPAA switches are not dead weight," in *Proc. IEEE ISCAS*, May 2007, pp. 72–169.

[40] C. R. Schlottmann and P. E. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 1, no. 3, pp. 403–411, Sep. 2011.

[41] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nW/MHz, 128 × 32 current-mode programmable analog vector-matrix

multiplier with over two decades of linearity," in *Proc. CICC*, Oct. 2004, pp. 651–654.
[42] J. Becker and Y. Manoli, "A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable G_M-cells," in *Proc. ISCAS*, May 2004, pp. I.1092–I.1095.
[43] J. Becker, F. Henrici, S. Trendelenburg, M. Ortmanns, and Y. Manoli, "A continuous-time hexagonal field-programmable analog array in 0.13 μm CMOS with 186 MHz GBW," in *Proc. IEEE ISSCC*, Feb. 2008, pp. 595–596.
[44] J. Becker, F. Henrici, S. Trendelenburg, M. Ortmanns, and Y. Manoli, "A field-programmable analog array of 55 digitally tunable OTAs in a hexagonal lattice," *IEEE J. Solid-State Circuits*, vol. 43, no. 12, pp. 2759–2768, Dec. 2008.
[45] F. Henrici, J. Becker, S. Trendelenburg, D. DeDorigo, M. Ortmanns, and Y. Manoli, "A field programmable analog array using floating gates for high resolution tuning," in *Proc. ISCAS*, May 2009, pp. 265–268.
[46] J. Luu *et al.*, "VTR 7.0: Next generation architecture and CAD system for FPGAs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 7, no. 2, pp. 6:1–6:30, Jul. 2014.
[47] S. Ramakrishnan, R. Wunderlich, J. Hasler, and S. George, "Neuron array with plastic synapses and programmable dendrites," *IEEE Trans. Biomed. Circuits Syst.*, vol. 7, no. 5, pp. 631–642, Oct. 2013.
[48] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Jan. 2018.
[49] J. Hasler, S. Kim, and F. Adil, "Scaling floating-gate devices predicting behavior for programmable and configurable circuits and systems," *J. Low Power Electron. Appl.*, vol. 6, no. 3, p. 13, Jul. 2016.
[50] *PSoC5 Data Sheet*, Cyprus Semi, San Jose, CA, USA, 2011.
[51] C. A. Looby and C. Lyden, "A CMOS continuous-time field programmable analog array," in *Proc. FPGA*, 1997, pp. 137–141.
[52] V. Gaudet and G. Gulak, "10 MHz field programmable analog array prototype based on CMOS current conveyors," in *Proc. Micronet*, 1999, p. 1.
[53] D. Keymeulen, R. S. Zebulum, Y. Jin, and A. Stoica, "Fault-tolerant evolvable hardware using field-programmable transistor arrays," *IEEE Trans. Rel.*, vol. 49, no. 3, pp. 305–316, Sep. 2000.
[54] A. Stoica, R. Zebulum, D. Keymeulen, R. Tawel, T. Daud, and A. Thakoor, "Reconfigurable VLSI architectures for evolvable hardware: From experimental field programmable transistor arrays to evolution-oriented chips," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 9, no. 1, pp. 227–232, Feb. 2001.
[55] B. Pankiewicz, M. Wojcikowski, S. Szczepanski, and Y. Sun, "A field programmable analog array for CMOS continuous-time OTA-C filter applications," *IEEE J. Solid-State Circuits*, vol. 37, no. 2, pp. 125–126, Feb. 2002.
[56] P. Lajevardi, A. P. Chandrakasan, and H.-S. Lee, "Zero-crossing detector based reconfigurable analog system," *IEEE J. Solid-State Circuits*, vol. 46, no. 11, pp. 2478–2487, Nov. 2011.
[57] S.-Y. Peng *et al.*, "A large-scale reconfigurable smart sensory chip," in *Proc. IEEE ISCAS*, May 2009, pp. 2145–2148.
[58] S. Brink, J. Hasler, and R. Wunderlich, "Adaptive floating-gate circuit enabled large-scale FPAA," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 11, pp. 2307–2315, Nov. 2014.
[59] D. Kahng and S. M. Sze, "A floating gate and its application to memory devices," *Bell Syst. Tech. J.*, vol. 46, no. 6, pp. 1288–1295, 1967.
[60] G. Gilder, *The Silicon Eye*. New York, NY, USA: Norton, 2005.
[61] M. Holler, S. Tam, H. Castro, and R. Benson, "An electrically trainable artificial neural network (ETANN) with 10240 'floating gate' synapses," in *Proc. Int. Joint Conf. Neural Netw.*, Washington, DC, USA, vol. 2, 1989, pp. 191–196.
[62] A. Thomsen and M. A. Brooke, "A floating-gate MOSFET with tunneling injector fabricated using a standard double-polysilicon CMOS process," *IEEE Electron Device Lett.*, vol. 12, no. 3, pp. 111–113, Mar. 1991.
[63] J. Hasler and B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers Neurosci.*, vol. 7, no. 118, pp. 1–29, 2013.
[64] B. A. Minch and P. Hasler, "A floating-gate technology for digital CMOS processes," in *Proc. IEEE ISCAS*, vol. 2, May/Jun. 1999, pp. 400–403.
[65] X. Li *et al.*, "Enabling energy-efficient nonvolatile computing with negative capacitance FET," *IEEE Trans. Electron Devices*, vol. 64, no. 8, pp. 3452–3458, Aug. 2017.
[66] V. Srinivasan, G. J. Serrano, J. Gray, and P. Hasler, "A precision CMOS amplifier using floating-gate transistors for offset cancellation," *IEEE J. Solid-State Circuits*, vol. 42, no. 2, pp. 280–291, Feb. 2007.
[67] V. Srinivasan, G. Serrano, C. M. Twigg, and P. Hasler, "A floating-gate-based programmable CMOS reference," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 11, pp. 3448–3456, Dec. 2008.
[68] J. Hasler and H. Wang, "A fine-grain FPAA fabric for RF + baseband," in *Proc. GOMAC*, 2015.
[69] V. Srinivasan, D. W. Graham, and P. Hasler, "Floating-gates transistors for precision analog circuit design: An overview," in *Proc. 48th Midwest Symp. Circuits Syst.*, vol. 1, Aug. 2005, pp. 71–74.
[70] S.-Y. Peng, M. S. Qureshi, P. E. Hasler, A. Basu, and F. L. Degertekin, "A charge-based low-power high-SNR capacitive sensing interface circuit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 7, pp. 1863–1872, Aug. 2008.
[71] D. W. Graham, P. Hasler, R. Chawla, and P. D. Smith, "A low-power programmable bandpass filter section for higher order filter applications," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1165–1176, Jun. 2007.
[72] R. Chawla, F. Adil, G. Serrano, and P. E. Hasler, "Programmable G_m-C filters using floating-gate operational transconductance amplifiers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 3, pp. 481–491, Mar. 2007.
[73] G. Serrano, M. Kucic, and P. Hasler, "Investigating programmable floating-gate digital-to-analog converter as single element or element arrays," in *Proc. IEEE Midwest CAS*, vol. 1, Aug. 2002, pp. 75–77.
[74] P. Brady and P. Hasler, "Offset compensation in flash ADCs using floating-gate circuits," in *Proc. IEEE ISCAS*, vol. 6, May 2005, pp. 6154–6157.
[75] A. W. Pereira, D. J. Allen, and P. E. Hasler, "A 0.5 μm CMOS programmable discrete-time Δ-Σ modulator with floating gate elements," in *Proc. ISCAS*, vol. 1, May 2004, pp. 213–216.
[76] P. Hasler, P. D. Smith, D. Graham, R. Ellis, and D. V. Anderson, "Analog floating-gate, on-chip auditory sensing system interfaces," *IEEE Sensors J.*, vol. 5, no. 5, pp. 1027–1034, Oct. 2005.
[77] A. Bandyopadhyay, P. Hasler, and D. Anderson, "A CMOS floating-gate matrix transform imager," *IEEE Sensors J.*, vol. 5, no. 3, pp. 455–462, Jun. 2005.
[78] A. Bandyopadhyay, J. Lee, R. W. Robucci, and P. Hasler, "MATIA: A programmable 80 μw/frame CMOS block matrix transform imager architecture," *IEEE J. Solid-State Circuits*, vol. 41, no. 3, pp. 663–672, Mar. 2006.
[79] P. Hasler, "Low-power programmable signal processing," in *Proc. Int. Workshop Syst.-Chip Real-Time Appl.*, Jul. 2005, pp. 413–418.
[80] S.-Y. Peng, P. E. Hasler, and D. Anderson, "An analog programmable multi-dimensional radial basis function based classifier," in *Proc. VLSI-SoC IFIP Int. Conf. Very Large Scale Integr.*, Oct. 2007, pp. 13–18.
[81] S.-Y. Peng, B. A. Minch, and P. Hasler, "Analog VLSI implementation of support vector machine learning and classification," in *Proc. ISCAS*, May 2008, pp. 860–863.
[82] S. Ramakrishnan and J. Hasler, "Vector-matrix multiply and winner-take-all as an analog classifier," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 2, pp. 353–361, Feb. 2014.
[83] P. Hasler and J. Dugger, "An analog floating-gate node for supervised learning," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 834–845, May 2005.
[84] T. S. Hall, C. M. Twigg, J. D. Gray, P. Hasler, and D. V. Anderson, "Large-scale field-programmable analog arrays for analog signal processing," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 11, pp. 2298–2307, Nov. 2005.
[85] S. George, J. Hasler, S. Koziol, S. Nease, and S. Ramakrishnan, "Low power dendritic computation for wordspotting," *J. Low Power Electron. Appl.*, vol. 3, no. 2, pp. 78–98, 2013.
[86] *OpenMSP430 Project: Open Core MSP430*. [Online]. Available: http://opencores.org/projectopenmsp430
[87] C. R. Schlottmann and J. Hasler, "High-level modeling of analog computational elements for signal processing applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 9, pp. 1945–1953, Sep. 2014.
[88] S. Shah, H. Toreyin, J. Hasler, and A. Natarajan, "Temperature sensitivity and compensation on a reconfigurable platform," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 26, no. 3, pp. 604–607, Mar. 2018.
[89] S. Shah, H. Toreyin, J. Hasler, and A. Natarajan, "Models and techniques for temperature robust systems on a reconfigurable platform," *J. Low Power Electron. Appl.*, vol. 7, no. 21, pp. 1–14, Aug. 2017.
[90] J. Hasler and S. Shah, "SoC FPAA hardware implementation of a VMM+WTA embedded learning classifier," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 28–37, Mar. 2018.
[91] S. Shah and J. Hasler, "VMM + WTA embedded classifiers learning algorithm implementable on SoC FPAA devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 8, no. 1, pp. 65–76, Mar. 2018.
[92] S. Shah and J. Hasler, "Low power speech detector on a FPAA," in *Proc. IEEE ISCAS*, May 2017, pp. 1–4.
[93] S. Koziol, S. Brink, and J. Hasler, "A neuromorphic approach to path planning using a reconfigurable neuron array IC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 12, pp. 2724–2737, Dec. 2014.
[94] S. Koziol and P. Hasler, "Reconfigurable analog VLSI circuits for robot path planning," in *Proc. NASA/ESA Conf. Adapt. Hardw. Syst.*, Jun. 2011, pp. 36–43.
[95] S. Koziol, D. Lenz, S. Hilsenbeck, S. Chopra, P. Hasler, and A. Howard, "Using floating-gate based programmable analog arrays for real-time control of a game-playing robot," in *Proc. IEEE Syst. Man Conf.*, Oct. 2011, pp. 3566–3571.
[96] S. Shah, H. Toreyin, O. T. Inan, and J. Hasler, "Reconfigurable analog classifier for knee-joint rehabilitation," in *Proc. IEEE Eng. Med. Biol. Soc.*, Aug. 2016, pp. 4784–4787.
[97] S. Shah, C. N. Teague, O. T. Inan, and J. Hasler, "A proof-of-concept classifier for acoustic signals from the knee joint on a FPAA," in *Proc. IEEE Sensors Conf.*, Orlando, FL, USA, Oct./Nov. 2016, pp. 1–3.
[98] H. Toreyin, S. Shah, C. Gungor, and J. Hasler, "Real-time vital-sign monitoring in the physical domain on a mixed-signal reconfigurable platform," *IEEE Trans. Biomed. Circuits Syst.*, to be published.
[99] J. Hasler and S. Shah, "Security implications for ultra-low power configurable SoC FPAA embedded systems," *J. Low Power Electron. Appl.*, vol. 8, no. 2, pp. 1–17, Jun. 2018.
[100] M. Laiho *et al.*, "FPAA/memristor hybrid computing infrastructure," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 62, no. 3, pp. 906–915, Mar. 2015.
[101] J. Hasler, A. Natarajan, S. Shah, and S. Kim, "SoC FPAA immersed junior level circuits course," in

Proc. MSE, May 2017, pp. 7–10.
[102] Accessed: Nov. 14, 2019. [Online]. Available: http://it.mathworks.com/solutions/fpga-design/
[103] (2012). *Zynq: All Programmable SoC Architecture*. [Online]. Available: http://www.xilinx.com/products/silicon-devices/soc/index.htm
[104] (2012). *SoC FPGAs: Integration to Reduce Power, Cost, and Board Size*. [Online]. Available: http://www.altera.com/devices/processor/soc-fpga/proc-soc-fpga.html
[105] Accessed: Nov. 14, 2019. [Online]. Available: http://www.altera.com/products/software/products/dsp/dsp-builder.html
[106] *Scilab: Free and Open Source Software for Numerical Computation*, Scilab Enterprises, Orsay, France, 2012.
[107] S. Granesan and R. Vemuri, "FAAR: A router for field-programmable analog arrays," in *Proc. Int. Conf. VLSI Design*, Jan. 1999, pp. 556–563.
[108] S. Ganesan and R. Vemuri, "A methodology for rapid prototyping of analog systems," in *Proc. Int. Conf. Comput. Design*, Oct. 1999, pp. 482–488.
[109] S. Ganesan and R. Vemuri, "Analog-digital partitioning for field-programmable mixed signal systems," in *Proc. ARVLSI*, Mar. 2001, pp. 172–185.
[110] S. Ganesan and R. Vemuri, "Behavioral partitioning in the synthesis of mixed analog-digital systems," in *Proc. IEEE DAC*, Jun. 2001, pp. 133–138.
[111] A. Doboli and R. Vemuri, "Exploration-based high-level synthesis of linear analog systems operating at low/medium frequencies," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 22, no. 11, pp. 1556–1568, Nov. 2003.
[112] G. R. Boyle, B. M. Cohn, D. O. Pederson, and J. E. Solomon, "Macromodeling of integrated circuit operational amplifiers," *IEEE J. Solid-State Circuits*, vol. SSC-9, no. 6, pp. 353–363, Dec. 1974.
[113] G. Casinovi and A. Sangiovanni-Vincentelli, "A macromodeling algorithm for analog circuits," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 10, no. 2, pp. 150–160, Feb. 1991.
[114] J. Kim, M. Jeeradit, B. Lim, and M. A. Horowitz, "Leveraging designer's intent: A path toward simpler analog CAD tools," in *Proc. IEEE CICC*, Sep. 2009, pp. 613–620.
[115] S. Liao and M. Horowitz, "A verilog piecewise-linear analog behavior model for mixed-signal validation," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 8, pp. 2229–2235, Aug. 2014.
[116] C. Elliott, V. Vijayakumar, W. Zink, and R. Hansen, "National Instruments LabVIEW: A programming environment for laboratory automation and measurement," *J. Assoc. Lab. Autom.*, vol. 12, no. 1, pp. 17–24, Feb. 2007.
[117] (1999). *Barcelona Design*. [Online]. Available: http://www.barcelonadesign.com
[118] M. Collins, J. Hasler, and S. George, "An open-source tool set enabling analog-digital-software co-design," *J. Low Power Electron. Appl.*, vol. 6, no. 1, pp. 13–15, Feb. 2016.
[119] C. R. Schlottmann, C. Petre, and P. E. Hasler, "Simulink framework for design to and automated conversion on large-scale FPAA devices," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, to be published.
[120] A. Natarajan and J. Hasler, "Modeling, simulation and implementation of circuit elements in an open-source tool set on the FPAA," *Analog Integr. Circuits Signal Process.*, vol. 91, no. 1, pp. 119–130, 2017.
[121] T. Weis, M. Knoll, A. Ulbrich, G. Muhl, and A. Brandle, "Rapid prototyping for pervasive applications," *IEEE Pervasive Comput.*, vol. 6, no. 2, pp. 76–84, Apr./Jun. 2007.
[122] M. Boshernitsan and M. Downes, "Visual programming languages: A survey," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/CSD-04-1368, Dec. 2004.
[123] W. M. Johnston, J. R. P. Hanna, and R. J. Millar, "Advances in dataflow programming languages," *ACM Comput. Surv.*, vol. 36, no. 1, pp. 1–34, Mar. 2004.
[124] W. W. Wadge and E. A. Ashcroft, *Lucid, the Dataflow Programming Language*. New York, NY, USA: Academic, 1985.
[125] E. Evenchick, "ICESTUDIO: An open source graphical FPGA tool," *Hackaday*, Feb. 23, 2016.
[126] S. Kim, S. Shah, and J. Hasler, "Calibration of floating-gate SoC FPAA system," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 9, pp. 2649–2657, Sep. 2017.
[127] W. H. Wolf, "Hardware-software co-design of embedded systems," *Proc. IEEE*, vol. 82, no. 7, pp. 967–989, Jul. 1994.
[128] Q. Zhao, M. Amagasaki, M. Iida, M. Kuga, and T. Sueyoshi, "An automatic FPGA design and implementation framework," in *Proc. IEEE DAC*, Sep. 2013, pp. 1–4.
[129] M. Weinhardt, A. Krieger, and T. Kinder, "A framework for PC applications with portable and scalable FPGA accelerators," in *Proc. IEEE DAC*, Dec. 2013, pp. 1–6.
[130] D. Rossi, C. Mucci, M. Pizzotti, L. Perugini, R. Canegallo, and R. Guerrieri, "Multicore signal processing platform with heterogeneous configurable hardware accelerators," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 9, pp. 1990–2003, Sep. 2014.
[131] J. Hasler *et al.*, "Transforming mixed-signal circuits class through SoC FPAA IC, PCB, and toolset," in *Proc. IEEE Eur. Workshop Microelectron. Educ.*, Southampton, U.K., May 2016, pp. 1–6.
[132] S. Koziol *et al.*, "Hardware and software infrastructure for a family of floating-gate based FPAAs," in *Proc. IEEE ISCAS*, May/Jun. 2010, pp. 2794–2797.
[133] B. Bolte, S. Shah, S. Kim, P. Hwang, and J. Hasler, "Live demonstration: FPAA demonstration controlled through android-based device," in *Proc. IEEE ISCAS*, May 2016, p. 1442.
[134] J. Hasler, S. Kim, S. Shah, I. Lal, M. Kagle, and M. Collins, "Remote system setup using large-scale field programmable analog arrays (FPAA) to enabling wide accessibility of configurable devices," *J. Low-Power Electron. Appl.*, vol. 6, no. 3, p. 14, 2016.
[135] J. F. Kurose and K. W. Ross, *Computer Networking: A Top-Down Approach*. Boston, MA, USA: Pearson Education, 2010.
[136] K. Park and H. Kim, "Remote FPGA reconfiguration using MicroBlaze or PowerPC processors," Xilinx, San Jose, CA, USA, Appl. Note XAPP441 (v1.1), Sep. 2006.
[137] R. Kuramoto, "QuickBoot method for FPGA design remote update," Xilinx, San Jose, CA, USA, Appl. Note XAPP1081 (v1.3), Mar. 2014.
[138] J. Vliegen, N. Mentcns, and I. Verbauwhede, "A single-chip solution for the secure remote configuration of FPGAs using bitstream compression," in *Proc. IEEE ReConFig*, Cancun, Mexico, Dec. 2013, pp. 1–6.
[139] M. Surratt, H. H. Loomis, A. A. Ross, and R. Duren, "Challenges of remote FPGA configuration for space applications," in *Proc. IEEE Aerosp. Conf.*, Big Sky, MT, USA, Mar. 2005, pp. 1–9.
[140] T. J. Freeborn and B. Maundy, "Incorporating FPAAs into laboratory exercises for analogue filter design," *Int. J. Elect. Eng. Educ.*, vol. 50, no. 2, pp. 188–200, 2013.
[141] E. Strasnick, M. Agrawala, and S. Follmer, "Scanalog: Interactive design and debugging of analog circuits with programmable hardware," in *Proc. UIST*, Québec City, QC, Canada, Oct. 2017, pp. 321–330.
[142] A. Malcher and Z. Kidoń, "Some properties of FPAA-based analog signal processing applications," in *Proc. IFAC Workshop Program. Devices Embedded Syst.*, 2009, pp. 184–189.
[143] (Jul. 2018). *AnadigmDesigner2 EDA Software*. Accessed: Nov. 14, 2019. [Online]. Available: http://www.anadigm.com/anadigmdesigner2.asp and http://www.anadigm.com/fpaa.asp
[144] C. M. Twigg and P. E. Hasler, "Incorporating large-scale FPAAs into analog design and test courses," *IEEE Trans. Educ.*, vol. 51, no. 3, pp. 319–324, Aug. 2008.
[145] P. Hasler, C. Scholttmann, and S. Koziol, "FPAA chips and tools as the center of an design-based analog systems education," in *Proc. IEEE MSE*, San Diego, CA, USA, Jun. 2011, pp. 47–51.
[146] M. Collins, J. Hasler, and S. George, "Analog systems education: An integrated toolset and FPAA SoC boards," in *Proc. IEEE MSE*, May 2015, pp. 32–35.
[147] J. Hasler, "Circuit implementations teaching a junior level circuits course utilizing the SoC FPAA," in *Proc. ISCAS*, Florence, Italy, May 2018, pp. 1–5.
[148] A. Maiti, A. D. Maxwell, A. A. Kist, and L. Orwin, "Merging remote laboratories and enquiry-based learning for STEM education," *Int. J. Online Biomed. Eng.*, vol. 10, no. 6, pp. 50–57, 2014.
[149] V. J. Harward *et al.*, "The iLab shared architecture: A Web services infrastructure to build communities of Internet accessible laboratories," *Proc. IEEE*, vol. 96, no. 6, pp. 931–950, Jun. 2008.
[150] D. Lowe, S. Murray, E. Lindsay, and D. Liu, "Evolving remote laboratory architectures to leverage emerging Internet technologies," *IEEE Trans. Learn. Technol.*, vol. 2, no. 4, pp. 289–294, Oct./Dec. 2009.
[151] N. Sousa, G. R. Alves, and M. G. Gericota, "An integrated reusable remote laboratory to complement electronics teaching," *IEEE Trans. Learn. Technol.*, vol. 3, no. 3, pp. 265–271, Jul./Sep. 2010.
[152] M. A. Bochicchio and A. Longo, "Hands-on remote labs: Collaborative Web laboratories as a case study for IT engineering classes," *IEEE Trans. Learn. Technol.*, vol. 2, no. 4, pp. 320–330, Oct./Dec. 2009.
[153] M. Cooper and J. M. M. Ferreira, "Remote laboratories extending access to science and engineering curricular," *IEEE Trans. Learn. Technol.*, vol. 2, no. 4, pp. 342–353, Oct./Dec. 2009.
[154] S. Ramakrishnan, "A system design approach to neuromorphic classifiers," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, GA, USA, 2013.