The Rise of SoC FPAA Devices

Jennifer Hasler
Electrical and Computer Engineering (ECE)
Georgia Institute of Technology
jennifer.hasler@ece.gatech.edu

Abstract: *This discussion reviews the current capabilities of large-scale Field Programmable Analog Arrays (FPAA) and considers the future potential of these SoC FPAA devices, including techniques to enable ubiquitous use of FPAA devices similar to FPGA devices. Today's FPAA devices include integrated analog and digital fabric as well as specialized processors and infrastructure, becoming a platform of mixed-signal development as well as analog-enabled computing. Investigating the scaling of FPAA devices shows the potential fine-grain capabilities through analyzing the tradeoff between granularity and flexibility as well as the opportunities through CMOS scaling.*

Field Programmable Gate Arrays (FPGA) are ubiquitous in many applications and are becoming embedded into a large number of applications starting from initial demonstrations in the 1980s. Although FPGAs tend to be less efficient than custom design with lower latency (1-3x), larger area (3-10x), and higher energy (30-100x), FPGAs allow near custom digital design through a reconfigurable digital substrate on a single IC. This flexibility has a cost while empowers design opportunities. These opportunities also include device reprogrammability for an IC designed once, standardization and abstraction of components (e.g. look-up tables (LUT), register elements), as well as easily upgradable during fielded operation, with little sacrifice on performance [1].

However, FPGAs are not low power when looking at solutions requiring 10-100mW of power. DSPs (started late 1970s) and microprocessors (μP), the low-power alternatives to FPGAs, have merged into a single ubiquitous technology and lead the low-power market (e.g. cellphones). Most FPGAs require 100s of mW simply to power up the SRAM elements holding the programming variables. Commercial Flash-based FPGAs significantly decrease the starting power requirements (e.g 7-10mW standby power [2,3]) while enabling 350-500MHz signals and 70mW 5G SERDES, although using these techniques still are too high for lower power systems of 10-20mW or less.

The research and development of large-scale Field Programmable Analog Arrays (FPAA), the mixed-signal extension of FPGA devices (Fig. 1,2), show significant potential for mixed-signal computing [4]; [4] provides an extensive review of FPAA history through current times. FPAA devices have provided a programmable and configurable structure to implement the 1000x computational energy efficiency over digital approaches (e.g. [4], [5]) that was originally demonstrated in custom Si [6] to prove Mead's original hypothesis [7]. FPAA devices have experimentally demonstrated a wide range of applications (Fig. 2), sometimes at small scale given the component constraints of a 350nm CMOS SoC FPAA [24]. For example, end-to-end embedded sensor to learning and classification have been demonstrated at 20-30μW on command-word recognition as well as on the full Nzero database [8]. Floating- Gate (FG) circuits enable Programmability, having precise parameters (e.g. 14-bit accuracy [9]), in standard CMOS, as well as configurability through long-term retention of FG charge (0-100 μV over 10 years [10], [11]). SoC FPAAs enable a mixed-signal platform as well as an analog-enabled computing platform, as they include analog elements and signals integrated with potential logic and mixed-signal enabled routing (Figs. 1,2). SoC FPAAs enable user development of the emerging analog computing techniques (e.g. [12]).

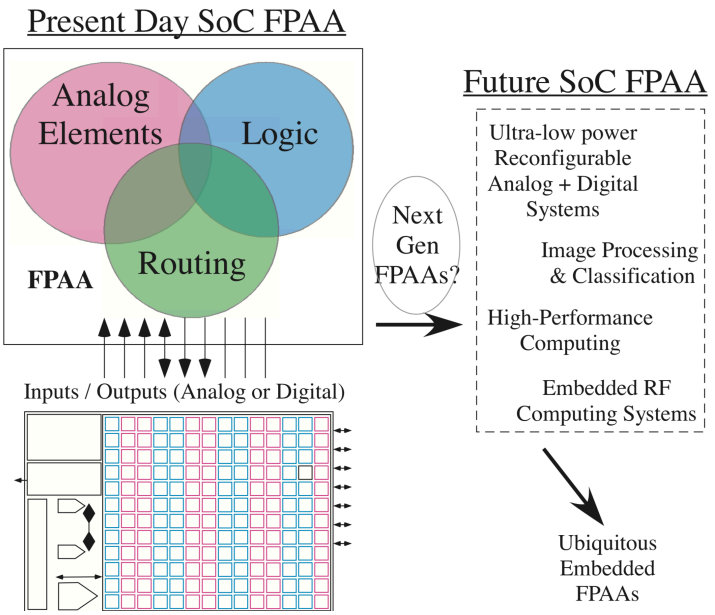
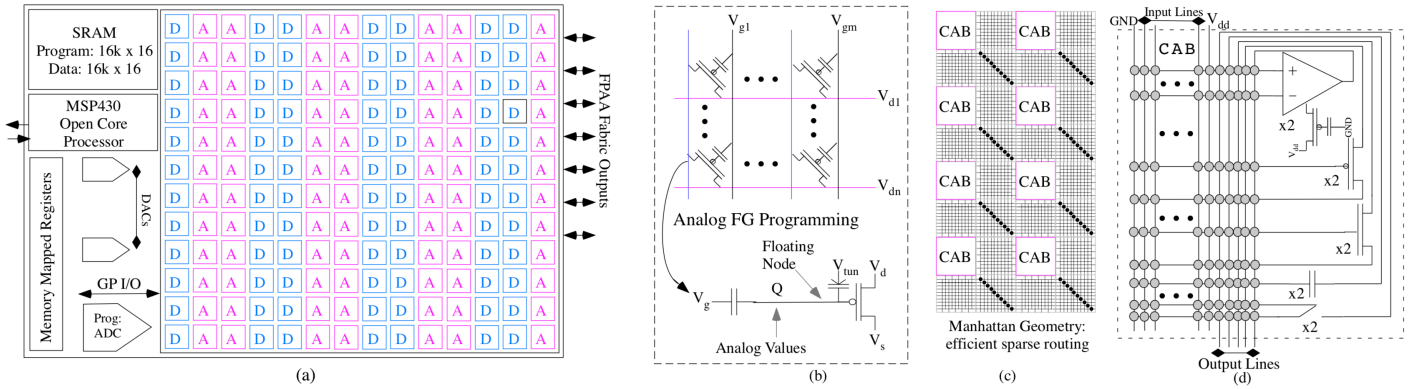


Figure 1: Fig 1: Large-scale Field Programmable Analog Arrays (FPAA) extend the concepts of FPGA devices to include Analog elements and signals integrated with potential logic and mixed-signal enabled routing. Today's FPAA devices include integrated analog and digital fabric as well as specialized processors and infrastructure, becoming a platform of mixed- signal development as well as analog-enabled computing. The fundamental question is what is the potential of future FPAA devices? Future FPAA devices seem to offer a promise of ubiquitous reconfigurable devices as part of ultra- low power mixed-signal applications, image processing & classification, near- zero latency RF computational applications, and low-power high-performance computing.

End-to-end computation requires analog input and outputs as well as computation for these physical implementations, with the system cost for data conversion and communication whether an FPGA or an FPAA system (Fig. 3). An FPGA can handle any analog signal inputs and outputs with the addition of sufficient data converters (Fig. 3), although the overall high energy, area, and complexity costs for the digital computation (e.g. [5] vs [13,14]), static FPGA power, and data converters will overwhelm many energy budgets. The significant energy and area efficiencies of analog computation, as well as the reduced system issues by significantly reducing the sampled data converters as well as the required data-converter SNR.

On the otherhand, an analog co-processor block for a digital computation with a bank of data converters creates a huge amount of infrastructure that nearly eliminates the benefits of the analog processing. The analog computation potentially provides near-zero latency for the computation compared to the latency required for the FPGA digital computation (e.g. [15]). And yet, today's FPGA-based solutions (Fig. 3a) are used because of the lack of commercially available FPAAs, as well as engineers having sufficient experience with FPAA solutions.

Like FPGA devices, FPAA tools can empower a wide application ecosystem, providing tools to enable an engineering team to rapidly develop new applications. The initial approach at systematic analog design tools arose through enabling FPAA designers to target designs to hardware from higher level abstractions as well as simulate these abstractions [4], [16], [17], [18]. These tools give the user the ability to create, model, and simulate analog and digital designs. This early tool effort enables the development for an FPAA toolset a toolset that can start with high-level definitions and automatically generate targeted hardware where as the user has



Partial List of Demonstrated FPAA Algorithms

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

(e)

Figure 2: FPAAs enable a range of computations in a single programmable and reconfigurable IC. (a) Block diagram of a SoC FPAA device, that includes analog (A: CAB) and digital (D) elements in the routing fabric integrated with a µP and other mixed-signal components. (b) The FPAA approach is enabled through analog programmable Floating-Gate (FG) devices, providing non-volatile storage (e.g. < 100µV in 10 years) and routing as well as computing directly through the routing crossbars. (c) SoC FPAA architectures use a manhattan architecture to route between components in a Computational Analog Block (CAB). (d) A typical CAB utilizes a number of components that may utilize FG parameters, as well as FG switches that can be used as part of the computation. (e) Current FPAA devices are capable of a diverse set of computations, as seen by the partial list of demonstrated FPAA algorithms.

the ability to optimize the process at each level. Recent efforts are expanding these analog and mixed-signal tools towards analog synthesis using standard cells for custom ICs (e.g. [19], [20], [21]).

FPAA devices, can be the solution for analog and mixed- signal security and component obselecence [30], just as FP- GAs soluve security and obsolescence issues. Multiple digital techniques can verify an FPGA, allowing for secure and confident FPGA programming for a particular application. An FPAA device can be a completely generic and known device that can be completely verified in a safe location [30], where the secret-sauce for the technology can be programmed on the device also in a safe location (Fig. 4). The resulting FPAA device layout says nearly nothing about the programmed function, similar to FPGA devices. FPAA devices can directly and discretely map secure functions, like Unique functions and Physically Unclonable Functions (PUF), directly into the FPAA fabric [30]. The nonvolatile programming of an FPAA to a specific hardware code removes most security holes from multi-level hardware stacks. FG memory eliminates the security issue of loading SRAM values. Analog values are difficult to measure without the measurements significantly distorting the values, and low-power circuits provide unique challenges for external measurements [30]. Analog and digital computing can be analog encoded. FPAAs can replicate similar analog circuit elements or a combination of analog circuit elements to achieve similar linear and nonlinear dynamics seen in older custom or configurable devices, including some of the unintended dynamics, eliminating the obsolescence issues. These techniques have been initially demonstrated for mapping analog music synthesis, even though the circuit techniques (e.g. BJT vs. FET) were used for these components [31], [32].

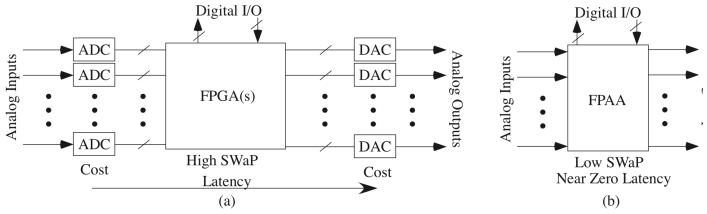


Figure 3: input and analog output computation. (a) An FPGA or multiple FPGAs can be used for a range of analog approaches assuming there are sufficient ADCs and DACs for the resulting analog signals. In addition to the high energy and complexity costs of these data converters, the FPGA has some latency for its digital computation and is constrained by the static and dynamic power issues for digital systems. (b) An FPAA device can typically handle the incoming analog signals directly, and where necessary (e.g. RF), those signals can be transformed to the power and supply voltage levels, transformations required for the FPGA devices when used. The FPAA device also computes with the high energy analog efficiency where possible in a near-zero latency path that can utilize digital control through the structure.

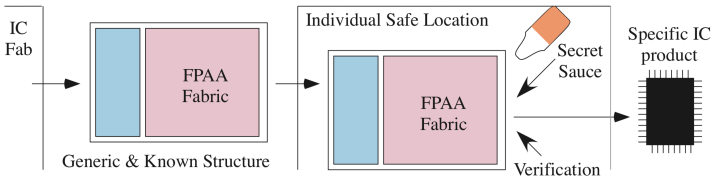


Figure 4: An FPAA device can be a completely generic and known device, completely verified in a safe location, and have the technology secret-sauce be programmed in a safe location. The output product is a custom chip due to the nonvolatile device programming.

The primary question is the CMOS node scaling opportunities for new FPAA devices given the current FPAA sys- tem capabilities. Scaling directly depends on on architectural and granularity tradeoffs in FPAA architectures (Sec. I). Configurable mixed-signal devices, having the opportunity of flexibility in a reasonably granular solution, can justify the IC design cost for new embedded devices (Fig. 5). As mask costs exponentially increase with decreasing processing node, the resulting design costs to obtain value from these investments increases exponentially, requiring a significantly higher expected market return from the effort (Fig. 5). Only a few applications (e.g. cell phone processors) can have the market impact that are necessary to justify the cost of advanced IC nodes (e.g. 10nm, 14nm), where configurable solutions can utilize a single IC design across a number of applications to justify the investment cost. Given the fine-grain and highly flexible opportunities of

configurable FPAA devices, the next step is predicting future mixed-signal computing opportunities (Sec. II) using FPAA devices (Fig. 1). This discussion works through the opportunities and challenges on the path to building a ubiquitous supply of SoC FPAAs.

I. Granularity: Flexibility vs. Switch Cost

An effective configurable fabric empowers the user's creativity through flexible opportunities while minimizing the added cost for that flexibility. Flexibility (φ) enables more computations in a single architecture, where φ quantifies the possible combinations available. Flexibility requires switches, and more switches result in higher area, circuit and interconnect cost (Fig. 6). A configurable architecture will always be a factor higher cost (K) in area, however small, compared to the fully custom architecture (A_1). The custom block area is fully custom, explicitly including in K any configurability or parameters. Each switch linearly increases K by a factor a, which is the ratio of the size of a switch compared to an individual selection block. Typical values for small to moderate cells connected to this switch would be between 0.1 to 0.01; switches selecting a single transistor element would have a closer to 1. Switch implementation in a particular technology Fig. 6b) directly affects a.

Switches add circuit costs to the flexible fabric. With the increase in area by (K), the custom computation has a total capacitance (C_L), and the configurable computation has an increased capacitance roughly scaled by the cost factor (K). Area efficiency due to configurability is the inverse of cost (1/K). The custom computation power-delay product (E_1) would be proportional to C_L that is proportional to A_1. For subthreshold operation and near-threshold operation, E_1 is constant with frequency. The Size, Weight, and Power (SWaP) metric, a product of the area and Power-delay product, for a custom system is proportional to E_1A_1, and for a configurable system is proportional to K^2 E_1A_1. Further, CMOS switches have a resistive loss Fig. 6b), although other technologies (e.g. III-V transistors and Calcoginides) potentially can reduce the signal loss for an on-switch.

Switch granularity is typically pictured as a continuum be- tween coarse-grain granularity that has a minimum of switches between a menu of items, and fine-grain granularity that has switches between the lowest level of components (Fig. 6a). Different architectures create a different K in their attempt to achieve their desired flexibility. Over the following subsections, we will examine how the cost of configurability (K) trades off with the resulting increase in flexibility (φ) described as increased functionality when connecting n blocks together, including coarse-grain architectures (Sec. I-A), manhattan architectures (Sec. I-B), and fine-grain architectures (Sec. I-C).

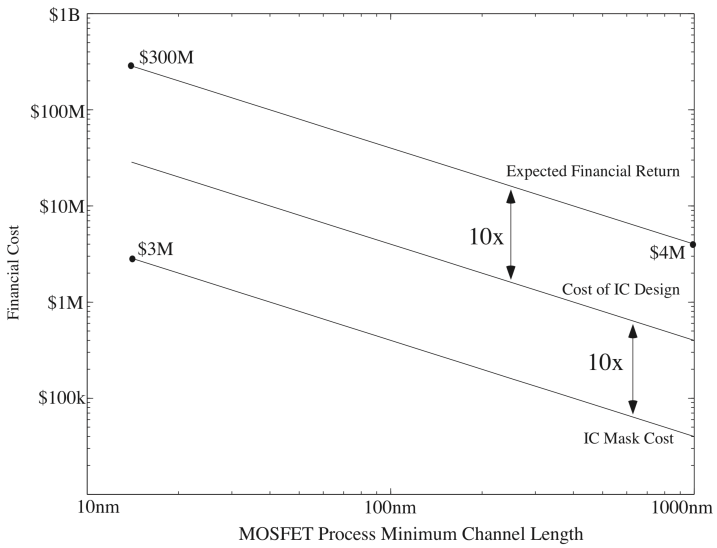
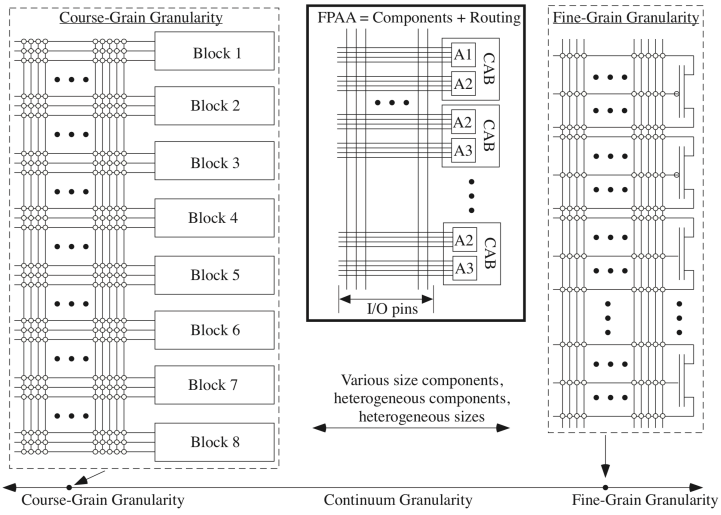


Figure 5: impact because of the ever-increasing cost of IC design for scaled down processes. The costs for making a set of IC masks scales inversely as a power law of the CMOS minimum channel length, and typically the design cost for a new design is at least 10x the mask cost, typically requiring a 10x the expected financial return to even attempt such a venture. The resulting cost for designing an IC is often far too high for most engineering applications to hope to reach these financial returns. A configurable device can spread this resulting engineering cost over a wide number of designs, enabling a market case for the engineering effort for an FPGA or FPAA device, as well as those directly using FPGA or FPAA devices not having to invest in the heavy IC design efforts.



(a)

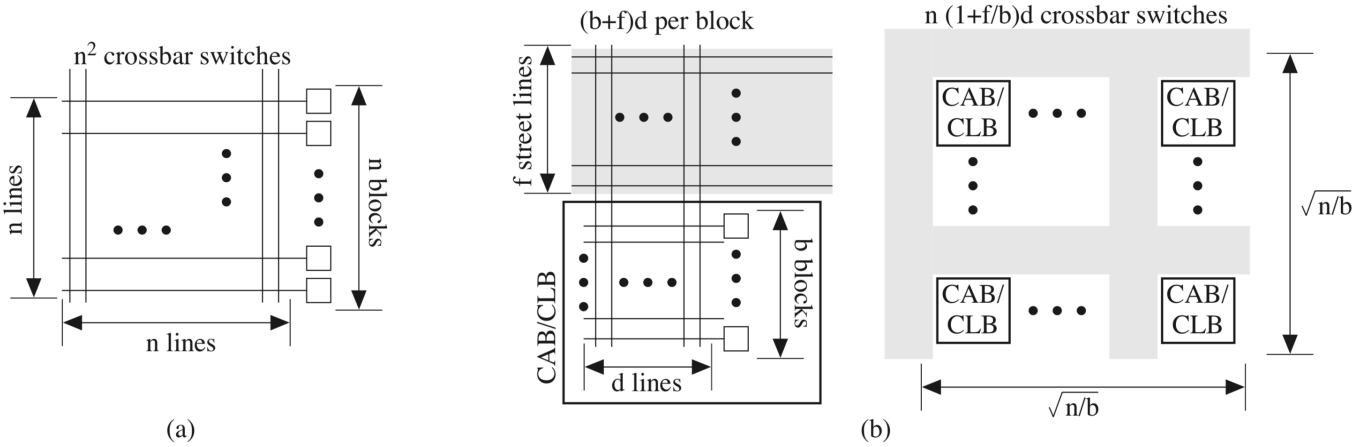
Switch	R Loss?	Nonvol?	Maturity	Application
CMOS FET	Y (≈1-10kΩ/μm)	N (On power)	Y (CMOS)	Rapid Reconfig
FG FET	Y (≈1-10kΩ/μm)	Y (>10 years)	Y (CMOS)	Analog & Digital Storage
MESFETs	less (≈0.1-1kΩ/μm)	N (On power)	Y? (not CMOS)	III-V materials
RRAM	Y (≈1kΩ/μm)	≈ Y? (≈month?)	???	???? ???
Calcoginide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

(b)

Figure 6: Impact of Switches on Routing Architecture. (a) Continuum of FPAA routing granularity. (b) Switch types compared considering significant Resistive (R) losses nonvolatile capabilities technology maturity for large-scale capabilties, and applications for these switches.

A. Coarse-grain architectures

Given the concern around switch cost for potential applications, many FPAA designs utilize coarse-grain architectures (Fig. 6a), minimizing the number of switches (Fig. 6b) and associated parasitics required for any particular computation. Coarse-grain architectures attempt to minimize the effect of additional switches by only switching between large fixed components, and the loss of opportunity by this strategy is incorporated into the flexibility metric (φ). Consider a system with N-blocks connected through a crossbar network (Fig. 7a); each block could have a selection connection to the n-1 other blocks resulting in a flexibility ≈ n or could have parallel connections to each of the n-1 blocks with a flexibility ≈ n^2 (Fig. 7c).



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

(c)

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

B. Manhattan architectures improve Flexibility

Manhattan architectures utilize a multilevel routing scheme to reduce the scaling of K with the number of elements while still achieving significant flexibility. FPGAs significantly improve their granularity through Manhattan architectures [1]. The evolution of configurable digital from fully connected structures to Manhattan type approaches enabled FPGAs with a routing structure that enabled a level of granularity beyond typical LUTs. The more flexible, efficient, and fine grain granularity enables creativity by the designer and such approaches requires significant device targeting tools[1].

Manhattan routing improves the effective granularity by assuming more connections are local or sparse, typical of digital and analog designs. Manhattan routing structures assume the number of elements or nodes to be routed are significantly larger than the crossbar determined by b, d, and f switch matrix (Fig. 7b), therefore having an improved scaling metric; the values of b, d, and f would increase weakly for increasing total number of nodes (= N). K scales with local routing (b, d, f) within each module (e.g. CLB or CAB) instead of the entire array (Fig. 7b). Other multilevel routing schemes have similar scaling properties.

Manhattan architectures utilize these crossbar arrays in each of their local regions (CLB/CAB) typically having n=8 to n=64 block elements, where one wants to maximize φ in each local region. A local region is defined as a large block where the routing architecture focuses on local computation, enabling these bus connections to only weakly grow with increased number of local regions and number of components.

C. Fine-grain architectures

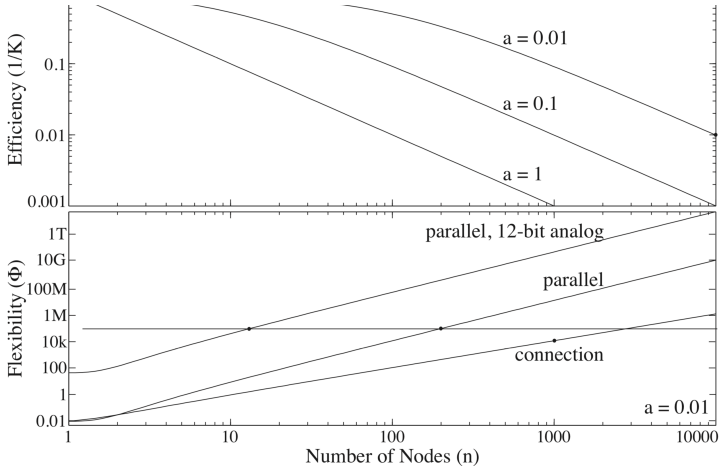
CMOS devices using FG elements allow for non-volatile switches potentially enabling analog granularity. Analog parameters improve the resulting density and resulting system flexibility (Fig. 7c). For

analog m-bit switch elements, the increased parallel flexibility increases by a 2^m factor, as the number of possibilities for a single switch increases by 2^m. Having analog parameters with parallel connections enables using routing fabric as computing fabric [22]. In this computing in memory approach [4], [23], the number of additional switches, and therefore K, for a particular computation decreases significantly.

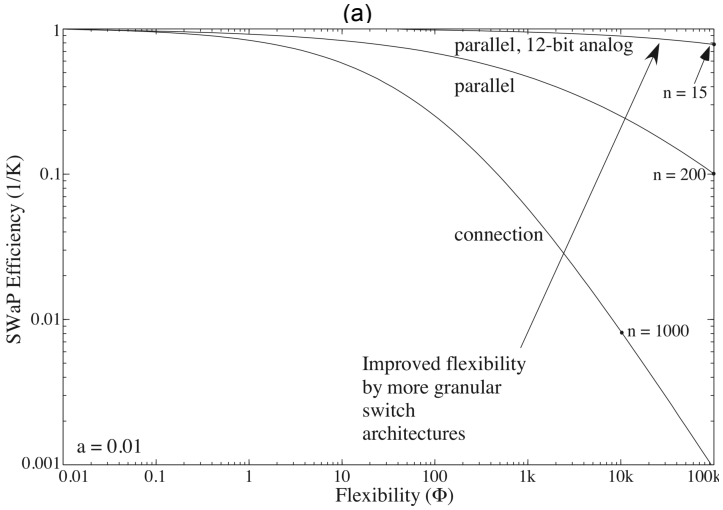
Fine-grain granularity, particularly analog programmable granularity, greatly improves the tradeoff between configurable architecture efficiency (1/K) and flexibility (Φ), requiring fewer nodes for similar flexibility as well as having a lower cost (K) of that flexibility (Fig. 8). The higher granularity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

Fine-grain, analog switch architectures provides a favorable tradeoff between configurable architecture efficiency (1/K) and flexibility (Φ), and further research to enable these techniques should yield many significant opportunities. These advantages are consistent with the demonstrated orders of magnitude advantage of FPAA devices using analog switching matrices, such as the SoC FPAA [4], [24]. As the computational routing fabric becomes more

important because of the high flexibility (Φ), additional fabric infrastructure, such as partial high-speed in-circuit reconfigurability [24], [25], further empowers the range of potential targeted applications.



(a)



(b)

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

The difference in φ between digital connection switches and parallel analog switches, enabling computing through the switches structured in a memory configuration, can be seen by the capabilities of a typical CLB and CAB (Fig. 9). Where a typical CLB can impressively enable a state machine per CLB, a CAB could potentially implement a small acoustic classifier stage in a single CAB. These differences in capabilities are almost entirely due to the fine-grain routing vs. an efficient traditional routing approach; coarse-grain routing techniques leave even more φ and capability unused.

Fine-grain programmability enables sufficient flexibility for security measures, as well as the infrastructure for programming and using fine-grain infrastructure enables verifying nearly every node on a given device.

II. OPPORTUNITIES OF SCALED FPAA DEVICES

Given that fine-grain capabilities have been demonstrated for their high flexibility compared to architectural cost, as well as initial SoC FPAAs have been experimentally demonstrated [24], what is the potential of these FPAA devices given existing understanding of these techniques? The scaling of FG devices to current processes (e.g. 40nm, 14nm) was experimentally shown [26], although further data will continue to provide confidence in these areas. Scaling improves the potential FPAA fabric bandwidth, enabling some RF bandwidths (e.g. 4GHz) at 40-45nm CMOS [26], [27]. What should one build as well as expect to be built for the next generation of FPAA devices particularly given recent possibilities for analog IC synthesis [20]? What is needed for a common module (Fig. 12) or image processing or remote sensor node application?

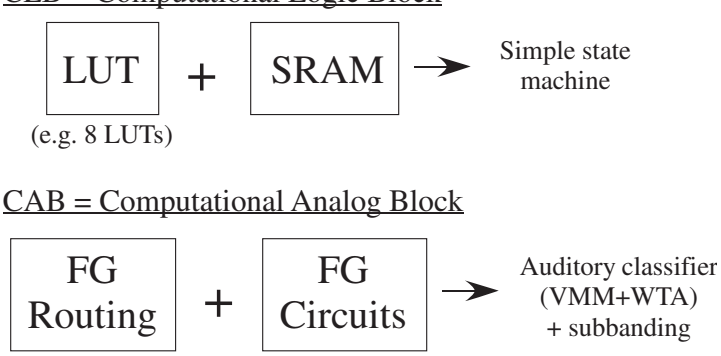


Figure 9: Comparison of the computation possible in a single local region, such as a Computational Logic Block (CLB) or a Computational Analog Block (CAB). A CLB typically uses binary connection switches to form multiple (e.g. 8) lookup tables and some selection RAM, enabling small state machines in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

A. Technical Opportunities from CMOS Scaling

Scaling provides two primary opportunities. First, scaling creates smaller switches and processing elements resulting in higher density and lower energy consumption. One would expect a significantly increased number of FG devices with CMOS scaling depending on process capabilities (Fig. 11a). Second, scaling allows for a higher signal bandwidth in the fabric architectures (Fig. 11a), roughly with an inverse quadratic scaling on the minimum channel length [26], [27].

Increased density, decreased energy consumption, and increased bandwidth directly impact the range of computations. The amount of traditional computation, expressed by the number of Vector-Matrix Multiplications (VMM) (5mm x 5mm) die grows rapidly with decreasing process node, where one assumes that the VMM elements are roughly 1/8 of the total routing fabric (Fig. 11b). From this modeling, one expects 45nm and 14nm devices are capable of PMAC(/s) level computation on a single die, computation levels typically requiring a large supercomputer (Fig. 11c).

These FPAA enable ultra-low power and energy harvesting operations given the possible computation at 1μW and 1mW levels (Fig. 11b). 1μW average energy could easily be supplied by small (< 1cm^2) energy harvesting devices. 1mW average energy could be supplied by a battery enabling months of continuous fielded use or by moderate sized (e.g. 10cm x 10cm) energy harvesting device. A 40nm CMOS structure enables 1GMAC(/s), around the level of a fully capable laptop computer, and around 1TMAC(/s), around the level of a small GPU or FPGA cluster.

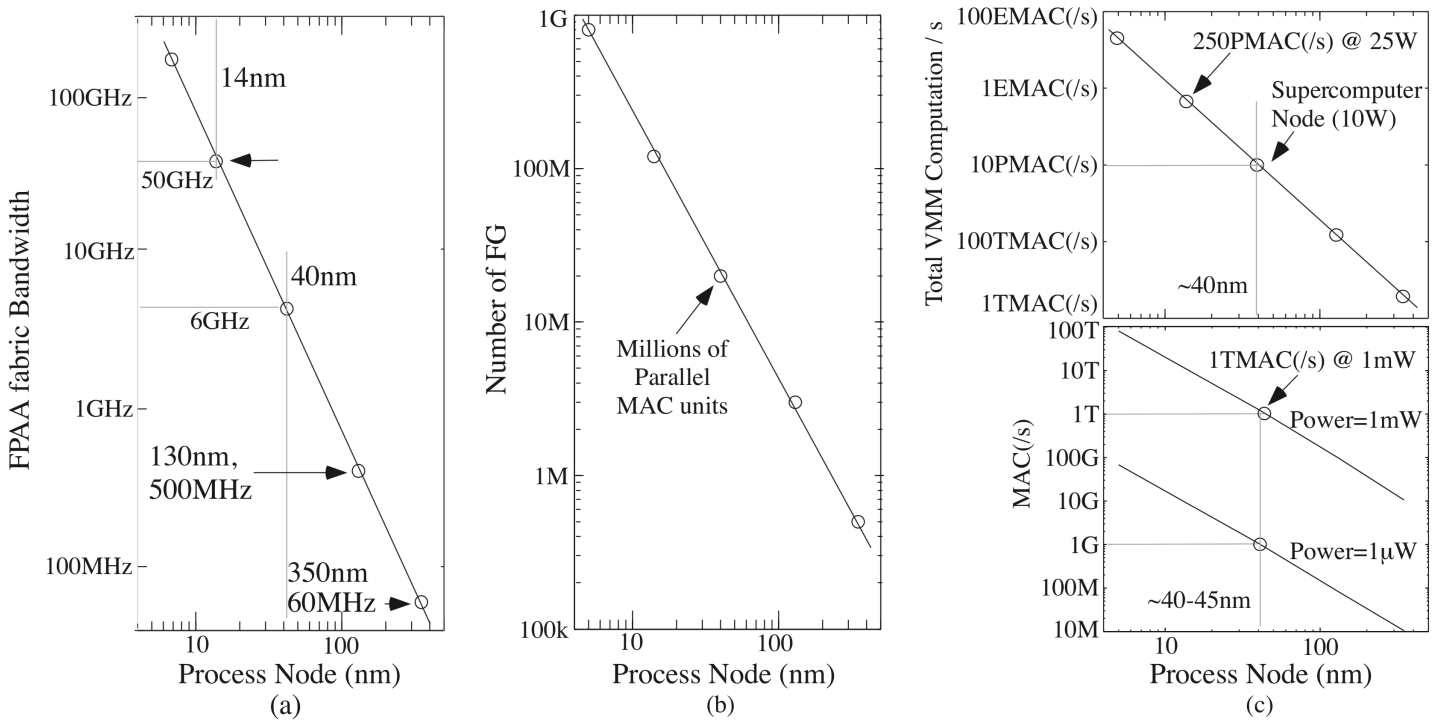


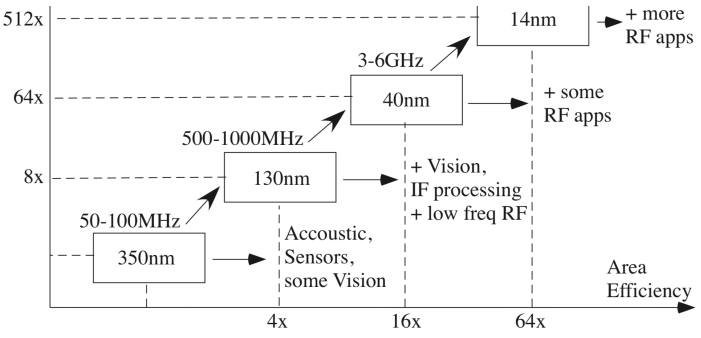
Figure 10: Bandwidth, parallel computational units, computational capability, and computational efficiency for scaled down FPAA devices extrapolated from experimental measurements and early studies of scaled down FG and FPAA devices. (a) Typical FPAA fabric bandwidth for scaled process nodes. (b) Number of FG elements for scaled process nodes that directly relates to the number of parallel computations (e.g. MAC). (c) Computational capability as well as computational efficiency for scaled down FPAA devices.

B. Application Opportunities from Scaling

FPAA algorithm opportunities can be mapped using the current [4] and future FPAA capabilities (Fig. 11), These end- to-end, sensor-to-refined result computations are possible at each CMOS process node, although the operating frequency and problem size will be lower for 350nm CMOS node com- pared with a 130nm, 40nm, and 14nm CMOS node. Analog numerical analysis [28], analog architecture theory [15], and real-valued computing theory [12] provides the analog computing framework.

For embedded applications, the optimal operating frequency matches the input data rate to eliminate the need for any internal storage or related infrastructure. In these particular situations, such as acoustic or speech processing or classification, scaling increases the problem size (command word to small vocabulary to speech classification) while potentially further improving the energy efficiency. In other cases, the increased problem size opens new architectural solutions, such as an image sensor classification where processing occurs on the incoming streamed image from an image sensor. Image classification on larger nodes might take a standard database with an on-board compression (e.g. Compressed DCT [29]), where a scaled down system would compute and classify subimages in parallel. A 45nm CMOS FPAA has enough CABs to locally store and configurably process images similar to GPU image reconstruction.

One can imagine an FPAA device being a common module for a range of application directions, including an FPAA device as a common module for RF related applications (Fig. 12) or a common module for front-end image processing and classification. A multi-input common-module (Fig. 12) is possible in 45nm and smalller CMOS [27]. CMOS scaling enables some applications, such as beamforming and demoduation that could be 40MHz at 350nm CMOS, while improving to 400MHz at 130nm CMOS, 4GHz at 40nm CMOS, and higher for smaller CMOS processes.



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

(b)

Figure 11: Scaling FPAA devices enables a range of new applications at scaled down nodes. (a) Scaling FPAA devices from 350nm CMOS to 130nm, 40nm ,and 14nm processes predictably increases the computational efficiency, area per operation, and fabric bandwidth, as well as opens new application spaces at each node. (b) Fabric perating conditions and CAB/CLB sizes for scaled FPAA devices.

III. SUMMARY AND FURTHER DIRECTIONS

Today's FPAA devices include integrated analog and digital fabric as well as specialized processors and infrastructure, and scaled FPAA devices show an increased potential on a single device, particularly high computing applications in ultra-low power constraints. This discussion presented a short summary of the FPAA device capability to date, including two primary threads of current FPAA development: components being connected together in a menu of functions, or a fine-grain interconnected network. The SoC FPAA device utilizes a form of this fine-grain network through analog programmability without paying the high cost of fine-grain switch networks.

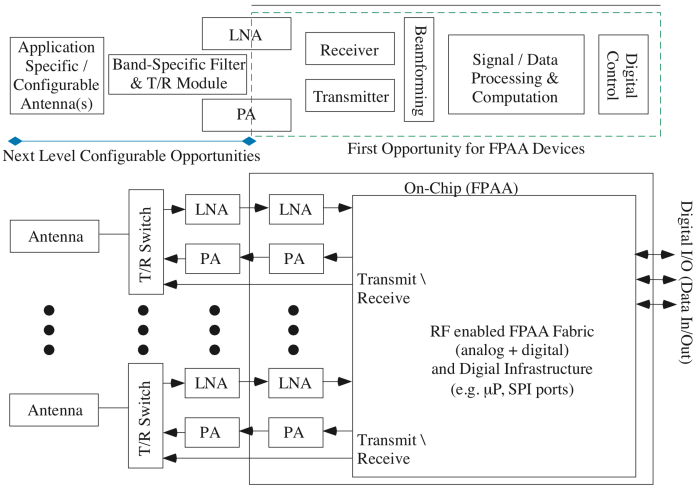


Figure 12: Configurable devices could encompass the entire core common-module for systems operating at RF frequencies, where the mixed-signal computation up to part of the LNA and PA devices through the back end control could be enabled in a single low-latency device. These systems would likely include additional devices in package or in the integrated product to handle some functions, including antenna devices and other specialized (e.g. high power for Power Amplifier (PA)) components. These components can be reconfigurable and controlled through the common module.

Given the limitations of CMOS devices, particularly scaled down CMOS devices, applications will continue to have a need for higher-voltage and higher-power external devices, and one would want some of these devices to be programmable and configurable. This approach enables configurability beyond the common module, say in an RF application (Fig. 12), to other structures either in the same package, on the same board, or in the system depending on the particular appli- cation requirements. Extending these concepts to III-V and GaN, potentially as configurable modules, require significant technology development and likely would be a next layer opportunity. The Si structure can provide the initial core tunability where needed for these ICs and chiplets including required I/O pins, where eventually some configurability can be developed in the native technology (e.g. III-V or GaN). The integration of these additional modules extends the capability of the common module.

And yet. as of today, FPAA devices are not ubiquitous. The community does not have a source of FPAA devices to enable the ubiquitous use of these devices as is seen for digital devices. The continued life of circuits like Anadigm's early FPAA devices [33] and related devices [34] demonstrates a constant hope for these configurable techniques, in spite of these devices very limited capabilities. Moving forward in this space requires a source of devices, and a source of devices requires building a bridge towards compelling application opportunities.

References:

[1] S. Trimberger, "Three ages of FPGAs: A retrospective on the first thirty years of FPGA technology," IEEE Proceedings, vol. 103, no. 3, 2015.
[2] IGLOO2 FPGA, Microsemi, https://www.microsemi.com/product-directory/fpgas/1688-igloo2. Last visited, May 8, 2021.
[3] https://www.microsemi.com/product-directory/antifuse-fpgas/1700-accelerator, last visited, May 8, 2021.
[4] J. Hasler, "Large-Scale Field Programmable Analog Arrays," IEEE Proceedings, vol. 108. no. 8. August 2020. pp. 1283-1302.
[5] C. Schlottmann and P. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 3, 2011, pp. 403–411.
[6] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and Hasler, "A 531 nW/MHz, 128 x 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," *IEEE Custom Integrated Circuits Conference*, October 2004, pp. 651 – 654.
[7] C. Mead, "Neuromorphic electronic systems," Proceedings of IEEE, vol. 78, 1990, pp. 1629-1636.
[8] J. Hasler and S. Shah, "SoC FPAA Hardware Implementation of a VMM+WTA Embedded Learning Classifier," IEEE Journal on Emerging and Selected Topics in Circuits and Systems, vol. 8, no. 1, March 2018. pp. 28-37.
[9] S. Kim, J. Hasler, and S. George, "Integrated Floating-Gate Program- ming Environment for System-Level Ics," IEEE Transactions on VLSI, vol. 24, no. 6, 2016. pp. 2244-2252.
[10] V. Srinivasan, G. J. Serrano, J. Gray, and P. Hasler, "A precision CMOS amplifier using floating-gate transistors for offset cancellation," *IEEE Journal of Solid-State Circuits*, vol. 42, no. 2, pp. 280-291, Feb. 2007.
[11] V. Srinivasan, G. Serrano, C. Twigg, and P. Hasler, "A Floating-Gate- Based Programmable CMOS Reference," *IEEE Transactions on Circuits and Systems I*, Vol. 55, No. 11, pp. 3448 - 3456, Dec. 2008.
[12] J. Hasler and E. Black, "Physical Computing: Unifying Real Number Computation," Journal of Low Power Electronics Applications, vol. 11, March 2021. pp. 1-21.
[13] B. Marr, B. Degnan, P. Hasler, and D. Anderson, "Scaling energy per operation via an asynchronous pipeline," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 1, pp. 147–151, 2013.
[14] B. Degnan, B. Marr, and J. Hasler, "Assessing Trends in Performance per Watt for Signal Processing Applications," IEEE Transactions on Very Large Scale Integration Systems, vol. 24, no. 1, 2016, pp. 58-66.
[15] J. Hasler, "Analog Architecture and Complexity Theory to Empowering Ultra-Low Power Configurable Analog and Mixed Mode SoC Systems," JPLEA, 2019.
[16] M. Collins, J. Hasler, and S. George, "An Open-Source Toolset Enabling Analog–Digital–Software Codesign," invited paper Journal of Low Power Electronics Applications, Vol. 6, no. 1, February 2016, pp. 1-15.
[17] J. Hasler and A. Natarajan, "An Open-Source ToolSet for FPAA Design," WOSET, November 2020.
[18] S. Kim, Sahil Shah, Richard Wunderlich, and Jennifer Hasler, "CAD Synthesis Tools for Large-Scale Floating-Gate FPAA System," Journal Design Automation for Embedded Systems, March 2021. pp. 1-16.
[19] J. Hasler, "Circuit Implementations Teaching a Junior Level Circuits Course Utilizing the SoC FPAA," ISCAS 2018, Florance, Italy, May 2018. pp. 1-5.
[20] J. Hasler, "Defining Analog Standard Cell Libraries for Mixed-Signal Computing enabled through Educational Directions," IEEE ISCAS, 2020. pp. 1-5.
[21] J. Hasler, "A CMOS Programmable Analog Standard Cell Library in Skywater 130nm Open-Source Process," WOSET, 2021.
[22] C. M. Twigg, J. D. Gray, and P. Hasler, "Programmable floating gate FPAA switches are not dead weight," IEEE

International Symposium on Circuits and Systems, May 2007, pp. 169 - 172.
[23] M. Kucic, P. Hasler, J. Dugger, and D. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," *Advanced Research in VLSI*, 14-16 March 2001, pp. 148 – 162.
[24] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R, Wunderlich, S. Nease, and S. Ramakrishnan, "A Programmable and Configurable Mixed-Mode FPAA SoC," IEEE Transactions on VLSI, vol. 24, no. 6, 2016, pp. 2253-2261.
[25] C. Schlottmann, S. Shapero, S. Nease, and P. Hasler, "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 9, 2012. pp. 2174–2184.
[26] J. Hasler, S. Kim, and F. Adil, "Scaling Floating-Gate Devices Predicting Behavior for Programmable and Configurable Circuits and Systems," Journal of Low Power Electronics Applications, vol. 6, no. 13, 2016, pp. 1-19.
[27] J. Hasler and H. Wang, "A Fine-Grain FPAA fabric for RF + Baseband," GOMAC, March 2015.
[28] J. Hasler, "Starting Framework for Analog Numerical Analysis for Energy Efficient Computing," Journal of Low Power Electronics Applications, vol. 7, no. 17, June 2017. pp. 1-22.
[29] D. G. Moreno, A. A. Del Barrio, G. Botella, and J. Hasler, "A Cluster of FPAAs to Recognize Images Using Neural Networks, IEEE TCAS II, Vol. 68, no. 11, Nov. 2021. pp. 3391-3395.
[30] J. Hasler and S. Shah, "Security Implications for Ultra-Low Power Configurable Analog and Mixed Mode SoC Systems," Journal of Low Power Electronics Applications, June 2018, pp. 1-17.
[31] S. Nease, A. Lanterman, J. Hasler, "A Transistor Ladder Voltage- Controlled Filter Implemented on a Field Programmable Analog Array," JAES , Vol. 62, no. 9, Sept 2014. pp. 611-618.
[32] S. H. Nease, A. D. Lanterman, and J. Hasler, "Applications of Current- Starved Inverters to Music Synthesis on Field Programmable Analog Arrays," Journal of Audio Engineering Society, Vol. 66, No. 1/2, Jan- uary/February 2018.
[33] Anadigm: Specifically generic analog functions for FPAAs Anadigm says, EE Times, Sep. 28, 2004.
[34] L. J. Kushner, K. W. Sliech, G. M. Flewelling, J. D Cali, C. M. Grens, S. E. Turner, D. S. Jansen, J. L. Wood, G. M. Madison, " The MATRICs RF-FPGA in 180nm SiGe-on-SOI BiCMOS," IEEE RFIC Symposium, May 2015, pp. 283-286.