A Programmable and Configurable Mixed-Mode FPAA SoC

Suma George, Sihwan Kim, Sahil Shah, Jennifer Hasler, Michelle Collins, Farhan Adil, Richard Wunderlich, Stephen Nease, and Shubha Ramakrishnan

***Abstract*— This paper presents a floating-gate (FG)-based, field-programmable analog array (FPAA) system-on-chip (SoC) that integrates analog and digital programmable and configurable blocks with a 16-bit open-source MSP430 microprocessor (μP) and resulting interface circuitry. We show the FPAA SoC architecture, experimental results from a range of circuits compiled into this architecture, and system measurements. A compiled analog acoustic command-word classifier on the FPAA SoC requires 23 μW to experimentally recognize the word dark in a TIMIT database phrase. This paper jointly optimizes high parameter density (number of programmable elements/area/process normalized), as well as high accessibility of the computations due to its data flow handling; the SoC FPAA is 600 000× higher density than other non-FG approaches.**

***Index Terms*— FPAA, floating-gate circuits.**

I. Introduction

THIS paper presents an integrated ultralow-power system-on-chip (SoC) field-programmable analog array (FPAA) IC enabling configurable and programmable analog and digital computation and interfacing. Fig. 1 shows this IC fully integrates a microprocessor (μP, open-source MSP430 [1]) enabling both computing and control with rapid reconfigurable analog–digital computation [2] with configurable fabric of interdigitated analog and digital computing blocks [3] to address a wide range of ultralow-power embedded system computational needs. The integration of these different concepts results in a jointly optimized FPAA performance, both in terms of high parameter density (number of programmable elements/area/normalized to process), as well as high accessibility of each of the resulting computations due to its advanced data flow handling. This IC was fabricated in a 350-nm CMOS process.

This large-scale FPAA enables analog computational energy efficiency [e.g. MMAC(/s)/W] 1000× lower than, and a die area × 100 smaller than, digital solutions, enabling low-power system computation, in the μW levels, empowering a whole range of applications, particularly always-ON context-aware processors. Computational efficiency, not including the energy required for communication, is measured in equivalent multiply and accumulate (MAC) operations per unit time per unit power, and fundamental computing operations found in analog and digital computation. For example, both custom [4] and configurable implementations [5] of vector-matrix multiplication (VMM) demonstrate 1–10-MMAC(/s)/μW power ranges, while the digital MAC energy wall remains roughly at 10 MMAC(/s)/mW [6]. The saturation of computational digital computation energy efficiency [6] influenced this SoC FPAA, a representative of physical computing, and reducing energy requirements for embedded system applications (acoustics, vision, communication, and robotics) through 1000× energy efficiency improvement [7], [8].

This paper focuses on the description of the SoC FPAA IC and the resulting measurements of compiled circuits to show the resulting functionality. In each case, the focus is not necessarily the most optimized circuit design, which would be complete papers unto themselves, but showing good performance for a compiled IP block that can be routinely used. The paper describes this FPAA IC architecture, basic analog and

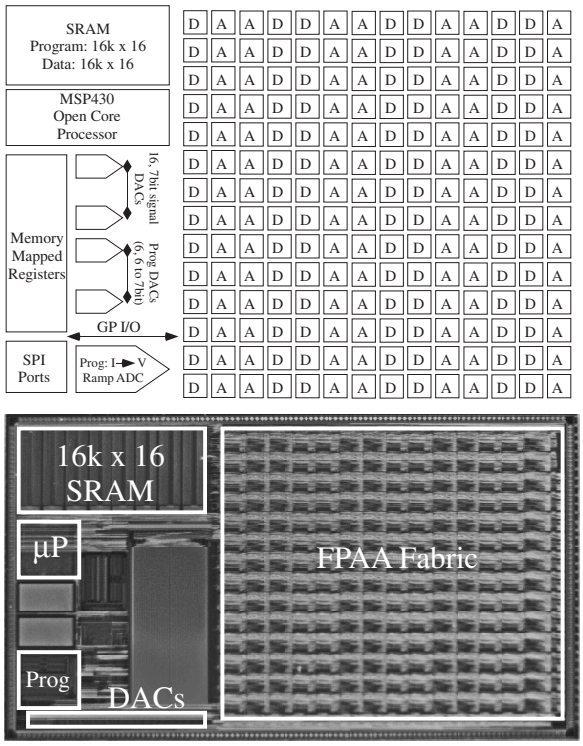

Fig. 1. RASP 3.0 integrates divergent concepts from previous multiple FPAA designs [2], [3], [11] along with low-power digital computation, including a 16-bit microprocessor (μP), interface circuitry, and DACs + ADCs. The FPAA SoC die photo measures 12 mm × 7 mm, fabricated in a 350-nm standard CMOS process. The die photo identifies μP, SRAM memory, DACs, and programming (DACs + ADC) infrastructure; the mixed array of the FPAA fabric is composed of interdigitated analog (A) and digital (D) configurable blocks on a single routing grid. DACs and programming infrastructure are accessed through memory-mapped registers.

Manuscript received June 10, 2015; revised September 16, 2015; accepted October 25, 2015.

The authors are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-250 USA (e-mail: jennifer.hasler@ece.gatech.edu).

Digital Object Identifier 10.1109/TVLSI.2015.2504119

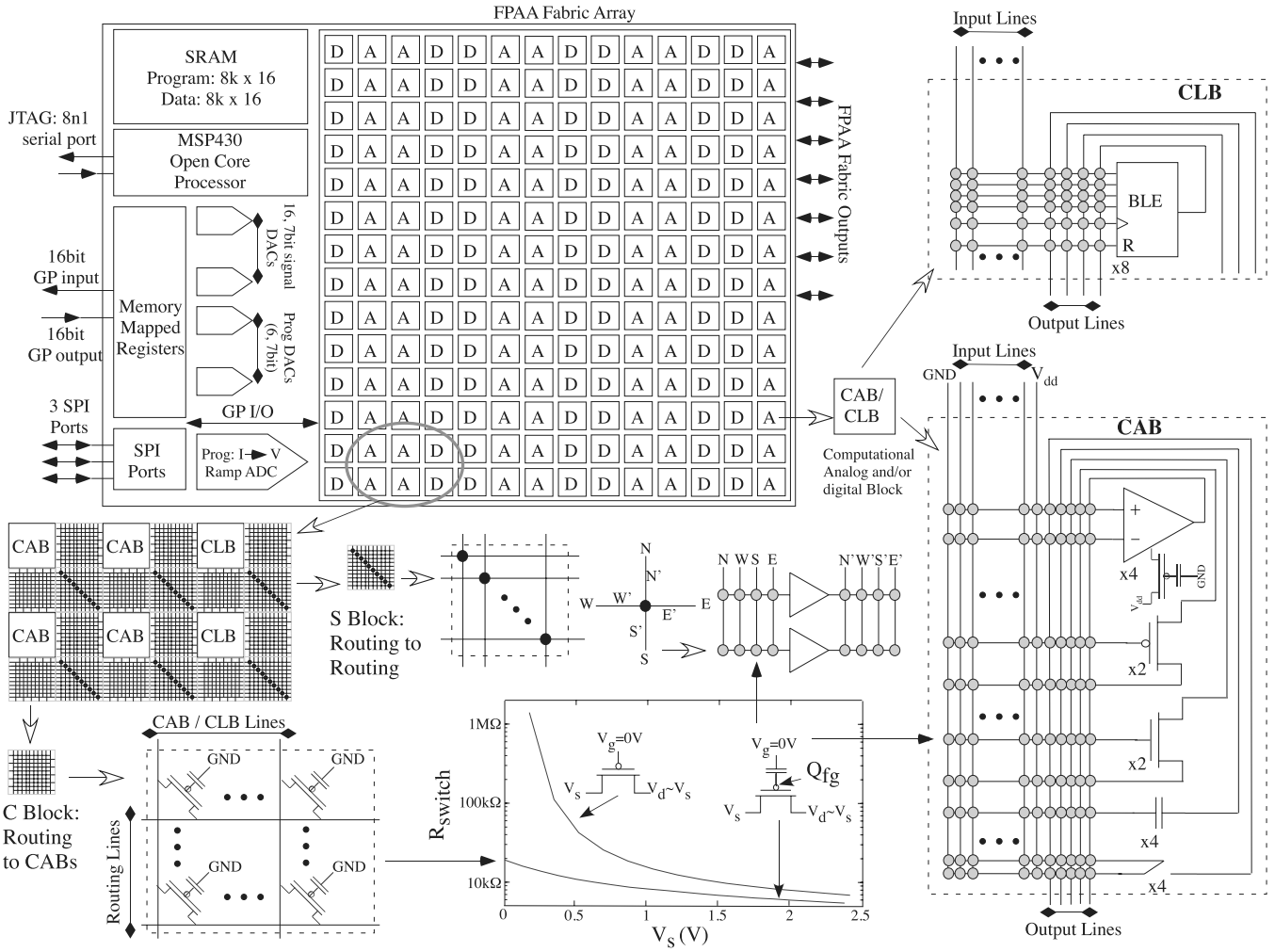

Fig. 2. RASP 3.0 functional block diagram illustrating the resulting computational blocks and resulting routing architecture. The infrastructure control includes a μP developed from an open-source MSP 430 processor [1], as well as on-chip structures include the on-chip DACs, current-to-voltage conversion, and voltage measurement, to program each FG device. The 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 [9]. The CABs and the CLBs are similar to previous approaches [3]. Eight, four input BLE lookup tables with a latch comprise the CLB blocks. Transconductance amplifiers, transistors, capacitors, switches, and other elements comprise the CAB blocks.

digital computational approaches, capacitance, timing, rapid reconfigurability of the routing fabric, implementation of data converters in the mixed-mode fabric, and utilizing the routing fabric as part of the computation.

This paper demonstrates (Section V) the first embedded classifier structure (command-word recognition) compiled onto a single FPAA device, going from sensor input (audio) to classified word, experimentally demonstrated in analog hardware. This demonstration is a small fraction of the overall IC. The SoC FPAA compiled system system power (23 μW) is consistent with the $\times$1000 improvement factor (comparison of MACs) for physical computation over digital approaches, with future opportunities for improved performance in the same IC.

Section VI summarizes the SoC FPAA design, as well as presents the comparison showing the SoC FPAA as the most sophisticated FPAA device built to date. The presented SoC FPAA device maximizes both parameter area normalized to the process node, nearly a factor of 500 improvement in area efficiency as typical of other analog FPAA devices, as well as utilization and accessibility of the resulting computational resources for the data flow. The closest high utilization structure (i.e., PSoC5 [10]) has nearly a 600 000 factor less in parameter density than this SoC FPAA device.

II. Architecture Description of the FPAA SoC IC

Fig. 2 shows the block diagram for the RASP 3.0 FPAA IC based on a Manhattan FPAA architecture, including the array of computation blocks and routing, composed of connection (C) and switch (S) blocks. This configurable fabric effectively integrates analog (A) and digital (D) components in a hardware platform easily mapped toward compiler tools. The switchable analog and digital devices are a combination of the components in the computational analog blocks (CABs), in the computational logic blocks (CLBs), and in the devices in the routing architectures that are programmed to nonbinary levels. The architecture is based on floating-gate (FG) device, circuit, and system techniques; we present the particular FG programming approach elsewhere [12].

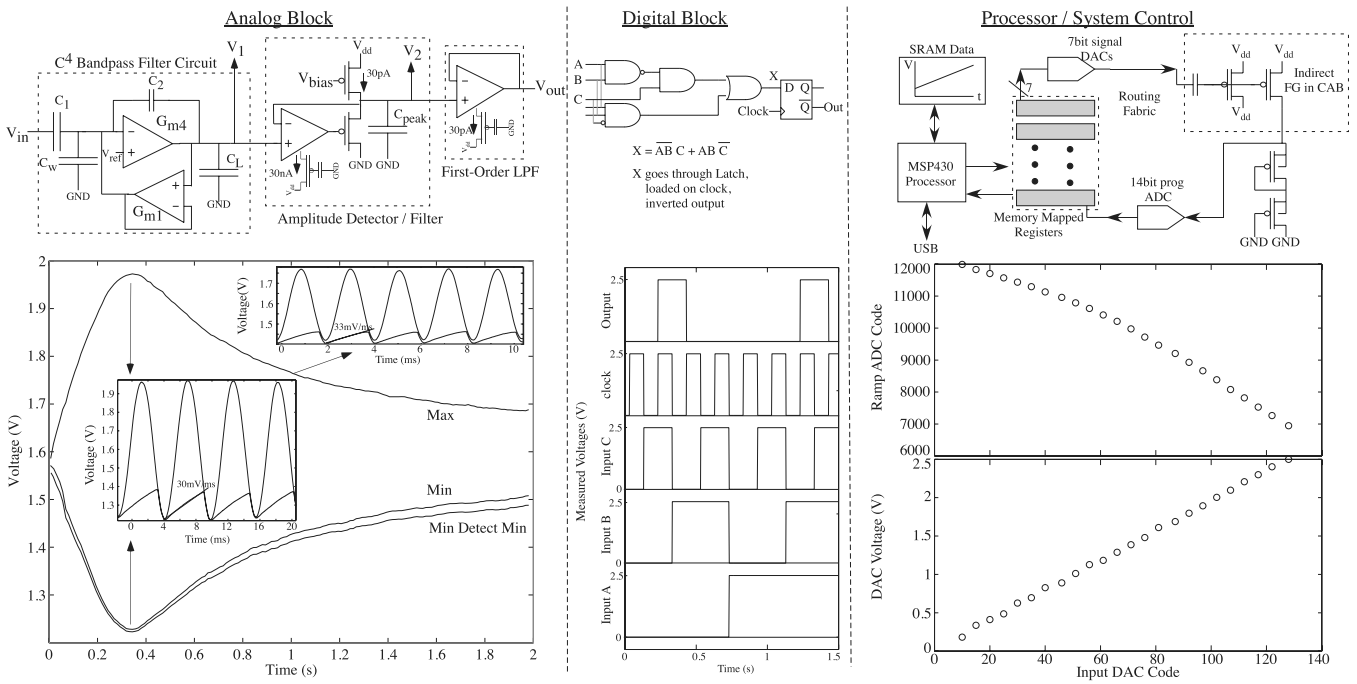

Fig. 3. SoC FPAA IC enables integration of analog and digital blocks in the routing fabric, as well as standard digital computation (i.e., μP) and infrastructure. Experimental measurements of heterogeneous programmable components from this FPAA IC are shown. Analog block: circuit diagram, compilation, and experimental measurement of a representative single signal processing chain—second-order bandpass filter (BPF), amplitude detector, and smoothing filter. Digital block: circuit diagram, compilation, and experimental measurement of a representative digital function using the lookup tables in a CLB; it illustrates the basic capability in a single BLE element and register. Digital computation/infrastructure: block diagram, compilation, and experimental measurement demonstrating a complete loop using a CAB device, the μP, a signal using (7 bit) DACs, the ramp ADC used in programming (14 bit), and a memory-mapped GP I/O. Instrumenting and measuring analog and digital blocks requires similar loops, employing all these capabilities as part of the FPAA computation.

The interaction of analog computation, digital field-programmable gate array (FPGA)-like components, and a μP infrastructure coming together creates a significant codesign space between these three domains (analog, digital, and μP). The analog computation combines significant innovations, enabling integration of previous heterogeneous concepts [2], [3], [11], from our earlier FPAA designs in ways not allowed or envisioned by the previous architectures. What is unique is the addition of digital low-power programmable and configurable FPGA fabric, first attempted in [3] (and fully integrated in this paper), to fully streamline the routing of analog and digital signals through a continuous fabric. Section III further describes our routing fabric and characterization. Furthermore, integrating these capabilities with an on-chip μP component and a range of digital communication ports completes the picture that this FPAA is an SoC computing device, not just an analog signal-conditioning device.

This FPAA further employs an open-source MSP 430 microprocessor (μP) with on-chip structures for 7-bit signal DACs, a ramp analog-to-digital converter (ADC), memory-mapped general purpose (GP) I/O, and related components. 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. The processor supplements the processing power of the digital portion of the system and increases overall implementation flexibility; portions of a problem can be mapped to reconfigurable analog, reconfigurable digital, or a GP digital processor.

Fig. 3 shows that our SoC FPAA approach enables integrated analog interfacing and computation with digital blocks, both FPGA and μP blocks. The analog components show the compilation of an auditory processing chain for subband signal detection. Where possible, one wants to compile key blocks into a single CAB to minimize parasitic capacitances and minimize global routing requirements.

III. SoC FPAA Routing Fabric Computation

But, our approach further moves away from the classical FPGA approach in a radical perspective, because the FG devices are programmed to analog levels; our routing fabric is no longer dead weight, as we hypothesized previously [13], and fully implemented in our SoC FPAA.

Our 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; 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 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. 4 shows an added routing structure component that enables rapid reconfigurability in the FPAA fabric. These

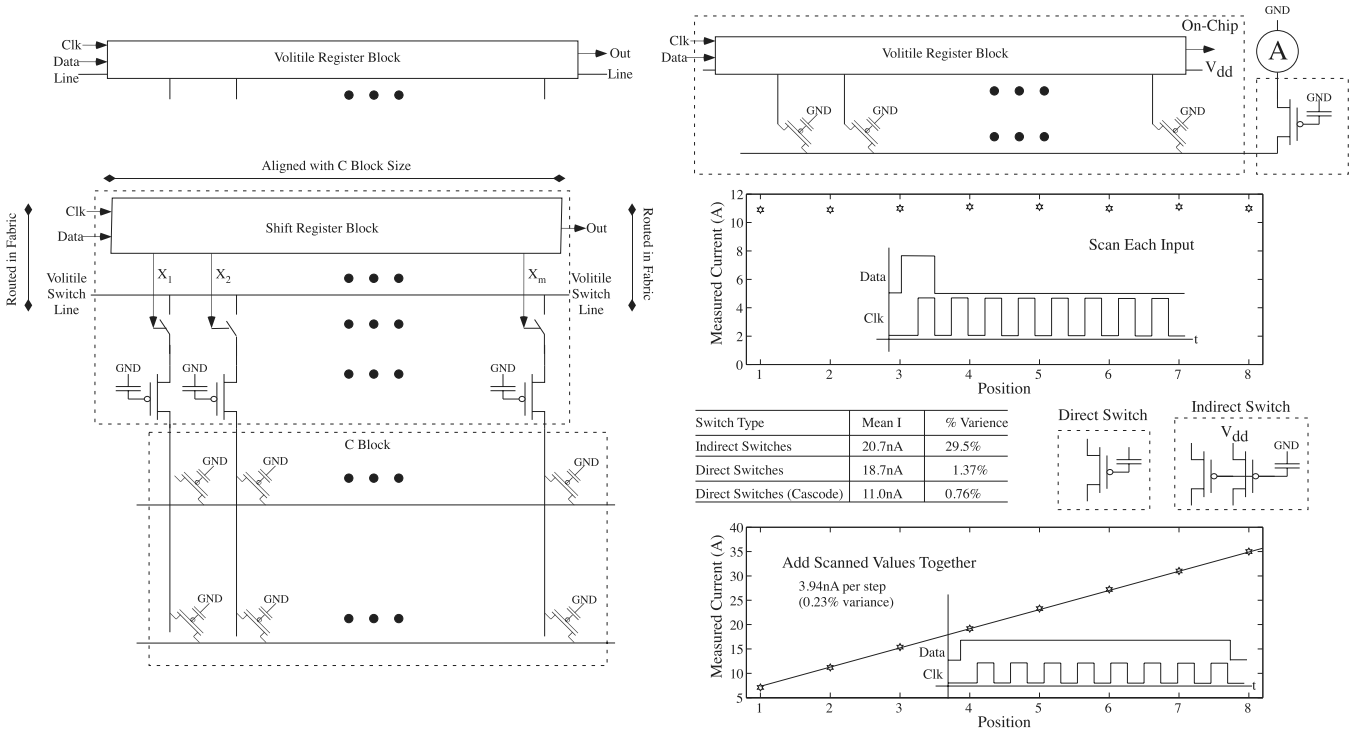

Fig. 4. FPAA SoC includes a set of T-gate-based switch elements in the routing fabric to empower rapid reconfigurability. These switches are accessed through a shift register that enables rapid change of configuration on a single clock cycle; different lines of the resulting *C* block and/or local routing store the different configurations. The resulting switches, resulting shift register, and switches connecting the block to the routing fabric are represented as a single volatile routing block. Utilizing routing elements programmed as precise current source elements illustrates both using them as an input to the shift register to scan through the individual signals, and using them as an input to the shift register to accumulate the resulting outputs through the individual signals. It is straightforward to imagine a range of arbitrary waveform generation based on patterns stored in routing fabric. This measurement gives a metric of programming accuracy in an operational mode. The accuracy for these switches was within 0.2%–0.76% for programmed subthreshold currents for uncorrected FG values; the resulting accuracy can be improved after such an initial measurement. Furthermore, some switches in the routing fabric use only a single pFET transistor (direct switches), while some use two pFET transistors (indirect switches), where one device is used for computation and one device is used for programming. The indirect switches show characteristically higher mismatch for uncorrected FG programming due to the threshold voltage mismatch of the two pFET devices. GND is signal GND; we bias the gate terminal for the FG devices at 0.6 V.

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 (CK) and data signals. 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. Fig. 4 shows using the routing fabric elements, this time as a bank of parallel current sources, as well as a cascading transistor. One easily sees an arbitrary waveform generator that could be compiled into the fabric; the circuit also becomes the nonvolatile memory for the function, eliminating outside memory and resulting complexity and energy requirements. The measurements show the accuracy of the FG transistor programming (FG voltage or resulting channel current). The measured accuracy is tighter than 1% for subthreshold currents (<250-μV).

Fig. 4 discusses the programming accuracy for the direct and indirect switches, where both are available within our routing fabric, sometimes in the same *C* block or local interconnect block. The difference between directly programmed and indirectly programmed FG devices is whether or not current measurements are made on the circuit transistor or the injection transistor during the programming algorithm. In the direct case, both the circuit and injection transistors are the same transistor. In the indirect case, they are two separate transistors. The indirect FG device leads to a more efficient switch (fewer parasitics), but one must account for the V_{T0} mismatch between the two pFET devices. The direct FG device uses the same pFET to program, measure, and compute, eliminating any V_{T0} mismatch, but requiring additional transmission gates in the signal path for programming.

Computing VMM solidifies the radical use of routing fabric as a computational element. Fig. 5 shows the implementation of a VMM in the routing fabric of our FPAA structure. We implement this functionality either in the *C* block or in the local CAB/CLB routing fabric, being that both structures are naturally crossbar arrays. Longer discussion on VMMs in early FPAA routing fabric is described elsewhere [5]. The VMM computation occurs through the memory device, using nonvolatile voltage storage, directly in routing fabric; other approaches, including traditional FPGA approaches, typically utilize memory separated from required computations. Integrated VMM and rapid reconfigurability enables switching

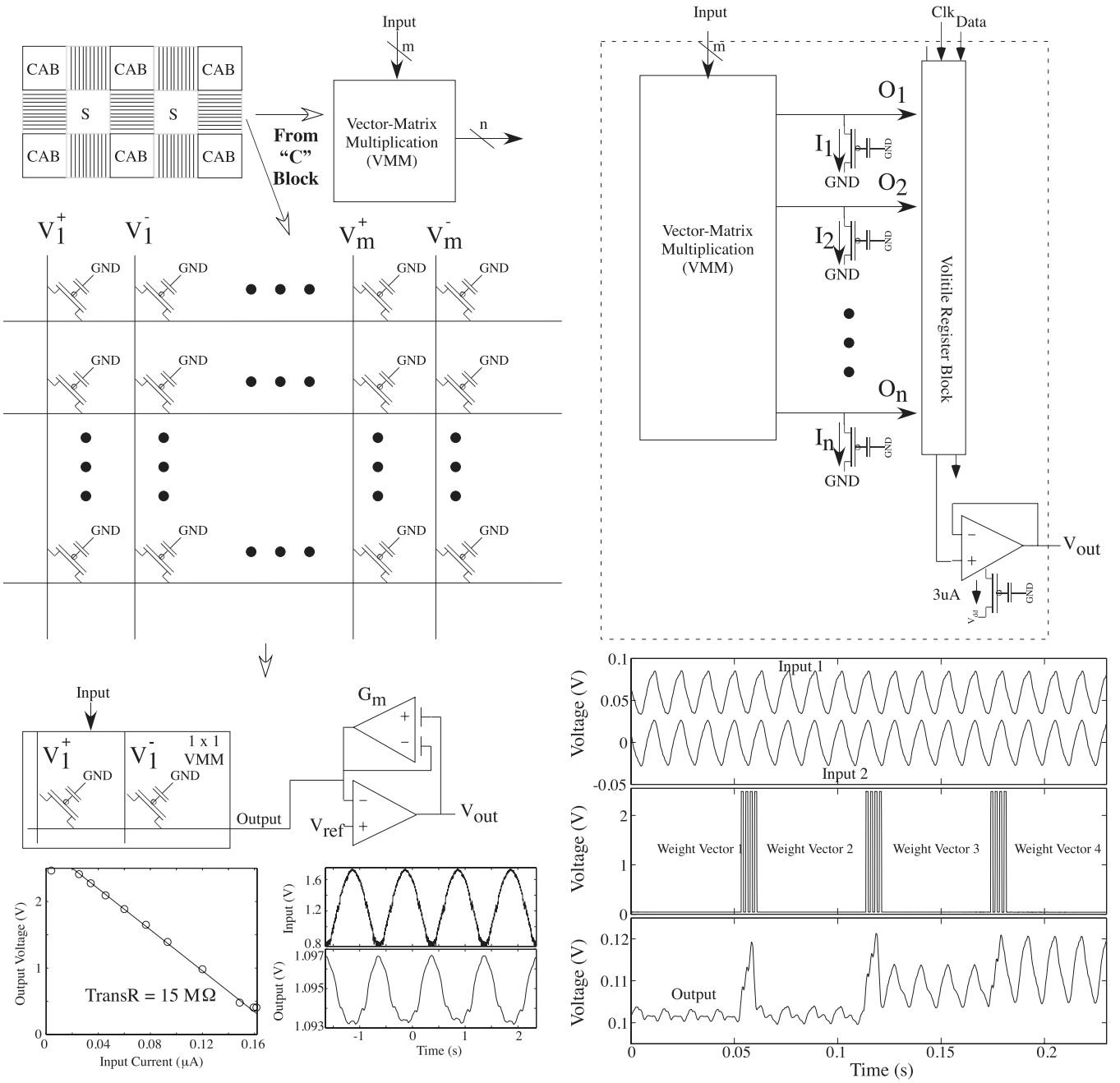

Fig. 5. VMM as a computational block instantiated in *C* block routing fabric. The *C* block forms a natural crossbar network typical for a VMM computation. The basic behavior is illustrated by the data for a single VMM element in routing fabric; two pFET transistors are required for source-input four-quadrant multiplication. We independently measure the resulting transresistance as 15 MΩ. Furthermore, we show an application of VMM integrated with the volatile switch register block to enable rapid (single clock) switching between weight vectors.

between metrics in the FPAA architecture. This feature permits data flow architectures to do a particular computation when data arrives, reducing the need for short-term storage.

IV. Representative SoC FPAA Processing

This section considers the behavior for some basic mixed-signal processing circuits compiled and experimentally measured in this system, illustrating basic analog–digital codesign approaches. The first example is compiling two basic ADC devices in the routing fabric. Compilation refers to the user starting with a high-level description of the IC and ending with an experimentally measured IC utilized in the same places as a commercial IC. Fig. 6 shows circuit compilation at the analog–digital boundary through compilation of multiple forms of ADCs as an example of integration of the capabilities. Being able to both compile an ADC and the particular needed ADC allows for optimal power computation and heavy IP block reuse, blurring system lines between analog and digital for more effective approaches of classifying raw analog data. The design of the routing fabric was not a block of analog components and a block of digital components with hard-build data converters in between, but rather a mixed fabric explicitly allowing blurred lines as the application requires.

The second example is a basic FPAA classifier using a single-layer VMM + winner-take-all (WTA) as a non-ADC

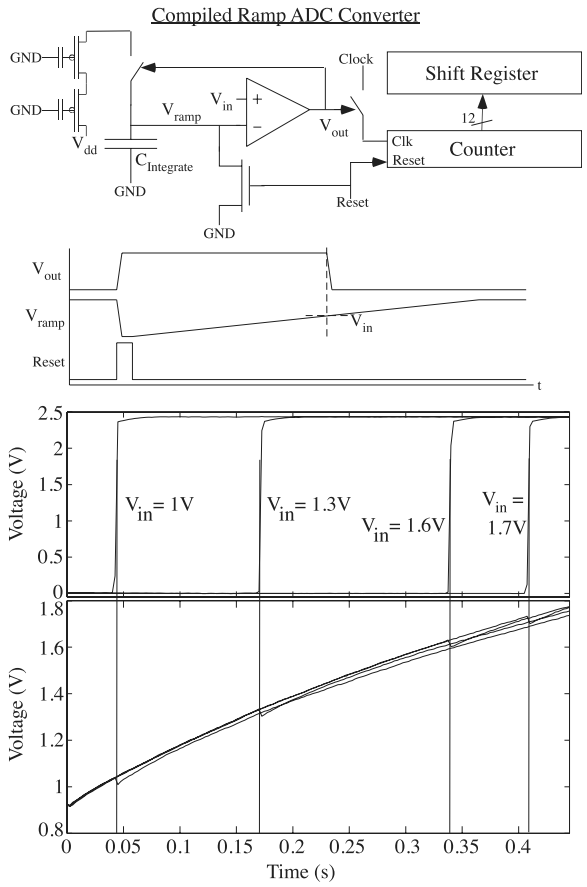

Fig. 6. Compiled and measured ramp ADCs on the SoC FPAA in the analog and digital enabled routing fabric. Ramp ADC: circuit diagram, timing diagram, and experimentally measured data for a compiled ramp ADC. The circuit requires one CAB, three BLEs, and four CLBs (24 registers for 12-bit counter and shift register). When the ramp crosses the input value, the open-loop OTA output switches. The first plot shows the OTA output voltage as a function of time for multiple different input voltages (1, 1.3, 1.6, and 1.7 V), and the second plot shows the measured V_{ramp} voltage (through a buffer). The voltage switches nearly when the ramp equals the input voltage but with a 45-mV offset. We present experimental data showing the comparison points for an approximately linear ramp input voltage. The largest systematic error is the curvature in the input ramp, generated by two FG routing switches charging up a single capacitor. The overlap capacitance to the FG reduces the effective early voltage of an FG device.

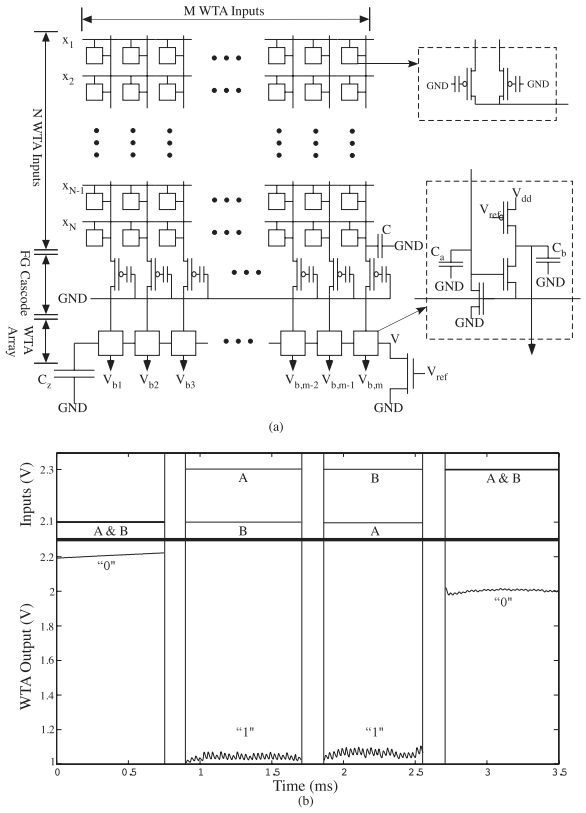

Fig. 7. Instantiated FPAA classifier block based on a combination of a VMM with a WTA block. (a) Circuit diagram for an N-input, M-output VMM + WTA classifier block, including typical circuits compiled for the individual VMM and individual WTA blocks. The WTA circuit is operated as a single winner circuit or as a k-WTA circuit, where up to k winners are possible if their metric is above a basic threshold as originally described in [14]. (b) Experimental measurements for a compiled three-input, three-output one-layer VMM + WTA classifier verifying the XOR functionality programmed into this classifier.

conversion between analog and digital signals. Fig. 7 shows a one-layer classifier approach based on the combination of a VMM and a k-winner WTA circuit [14] that elegantly compiles into routing fabric [15]. The one-layer architecture can perform standard one-layer hyperplane classifiers, while also performing tasks considered impossible for typical one-layer neural network architectures (i.e., XOR). Fig. 7 shows experimental measurements for both of these cases: 1) an XOR function and 2) a linear approximator function. The result experimentally verifies the universal approximator behavior of this one-layer VMM + WTA architecture compiled on this RASP 3.0 FPAA structure.

V. Representative SoC FPAA System Application

We next show a complete analog signal processing application compiled in an SoC FPAA; this system is the first compiled sequence of signal processing algorithms shown on any FPAA/configurable device. The signal processing circuits compiled and measured on this SoC FPAA show measured data for performing command-word recognition. We expect the SoC FPAA could be used for a variety of potential applications for sound/acoustics/speech, image processing and vision sensors, robotics, and wireless communication.

We show an example application of auditory/speech classification looking at detecting a command word in a sentence. Fig. 8 shows the first application example of an auditory classifier structure for a limited phrase, such as a command word, that can be classified through features in the averaged signal spectrum. Continuous-time spectrum decomposition used a bank of constant Q filters at the first processing stage, using a bank of amplitude detection and filtering operations, and then a VMM + WTA classifier block to classify each of the resulting spectrum into simple symbols. In a more complex speech recognition system, one might have the spectrum correspond to phonemes or part of phonemes and build ups the temporal representations using the temporal classification (i.e., HMM classification) to word spot the resulting phonemes, syllables, and words. A simple command-word application, required only to distinguish between a few simple symbols, can be directly computed as a state machine on the MSP430 processor; a next level of computation, say a simple Viterbi decoder, could be directly implemented on the MSP430 processor as well.

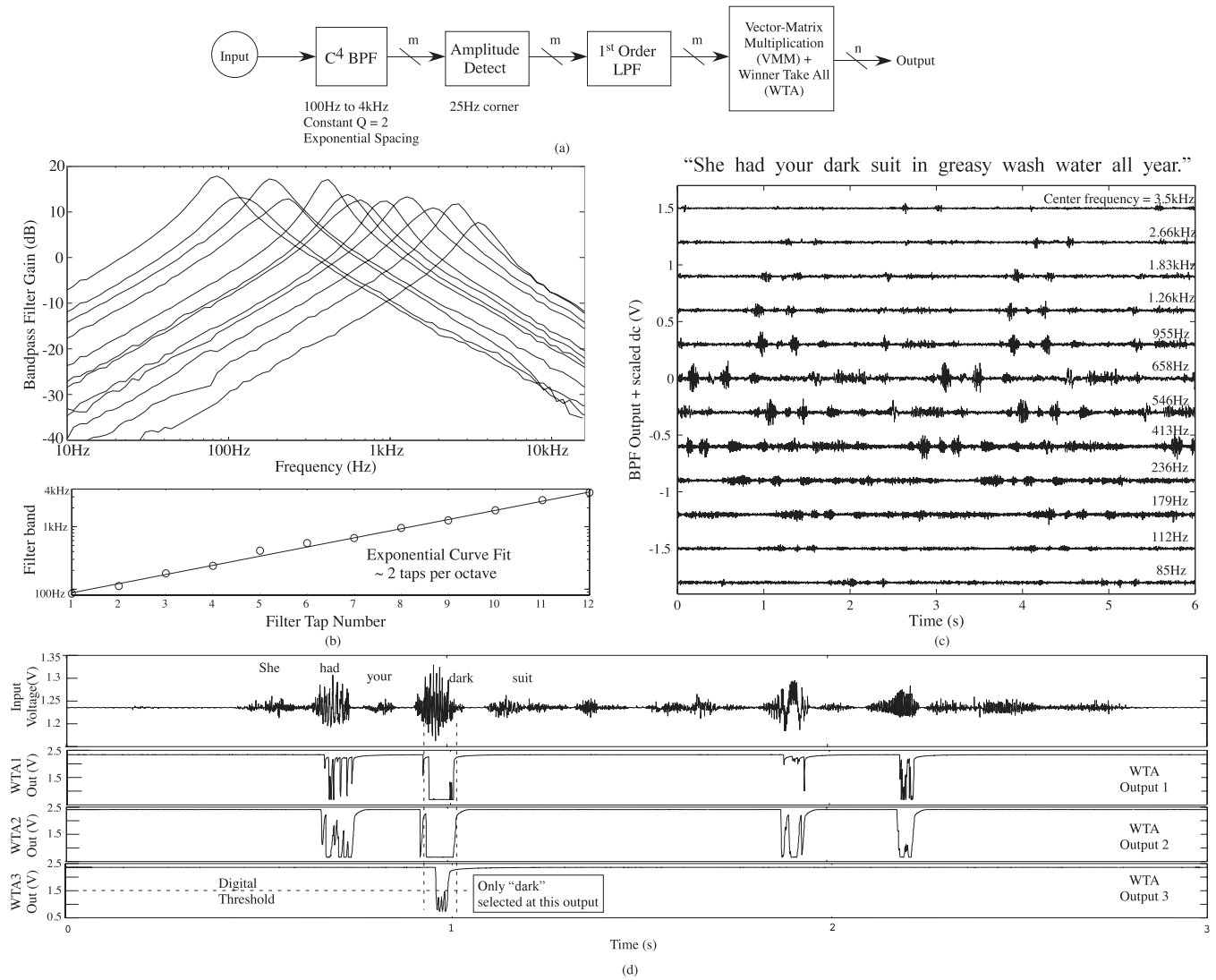

Fig. 8. Analog auditory word classification application compiled into the RASP 3.0, showing the experimental waveforms from the IC. (a) Block diagram for the classifier algorithm, in a similar representation used for our tool framework. (b) BPF center frequencies that are scaled evenly on a log-frequency scale between 100 Hz and 4 kHz with a nearly constant Q ($Q \approx 2$). The programming approach only accounted for part of the threshold voltage mismatch variations (indirect FG devices) and did not consider effects of capacitor mismatch; additional techniques can be used for the existing IC for tighter measurements. The peak gain was the largest variation between the filters; Q was roughly 2 with some variation. The center frequency for each of the filters, as shown, was monotonic and fairly close to an ideal exponential spacing. (c) BPF outputs and amplitude detection for a single phrase from the TIMIT database. (d) Classification of word and components for a TIMIT waveform, using a k-WTA with three outputs to detect the word dark in the resulting phrase.

Fig. 9 shows the power required for the compiled command-word classifier computation by a functional block, as well as the entire system, which requires 23 μW for the current implementation. The implementation is not optimal, particularly in the LPF and classifier blocks, but shows the functional capability of the system at a very low (23 μW) power level; the LPF did not filter at a low enough rate, resulting in 5-kHz bandwidth signals into the VMM, while the VMM could have been operated at lower frequencies. Optimal biasing for these blocks could have resulted in a factor of 100 in the overall power budget, reducing the power by a factor of 3. We used buffers to characterize the resulting output signals off-chip; routing the signals into the digital μP would nearly eliminate the need for the buffer power. This operation could be optimized for lower power, which is the focus of another discussion (and beyond the scope of this paper). When programming the resulting system, we only partially accounted for the threshold voltage mismatch due to indirect programming mechanisms and did not program around capacitance variations. A closer look at Fig. 8(b) shows the center frequency for each of the filters is monotonic and fairly close to an ideal exponential spacing, and the Q is roughly 2 with some variations. The peak gain shows the largest variations due to capacitance (C_2, in Fig. 3) mismatches, where this capacitance is the small parasitic capacitance between the OTA terminals resulting in more device-to-device variation. The gain variation can be eliminated by modifying the weights of the VMM block. This level of programming is shown sufficient for this application, while further programming accuracy possible in the system might be required in further applications.

This classifier system, compiled on this FPAA, is consistent with the 1000$\times$ improvement factor in computation (measured in equivalent multiplication and accumulate, or MAC, operations), is similar to systems developed for VMMs (cus-

(a)

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

(b)

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)

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, 3transitors, routing))	

Fig. 9. SoC FPAA values. (a) Measured power numbers for compiled command-word classifier function. (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.

tom and compiled) [5], as well as other custom classifier networks [15]–[17], [19]. The VMM + WTA classifier, being a universal approximator [15], can generate the same classification functions as other radial basis function networks or Guassian mixture model networks [16], [17], as well as related algorithms [18]. This case shows a full system for an embedded classifier structure, going from sensor input (audio) to classified word, and further, is experimentally demonstrated in configurable analog hardware utilizing high-level design tools. The focus in this paper is to demonstrate the operation of this classifier structure; further papers will describe the optimal design, classification accuracy metrics, robust behavior over data sets, and power consumption optimization.

VI. Summary Discussion and Comparisons

We presented an IC that integrates divergent concepts from multiple previous FPAA designs along with low-power digital computation and interface circuitry (i.e., DACs and ADCs). We showed through discussion and measured data that this unified structure enables a wide range of SoC computing options that can be optimized for multiple parameters, showing the most sophisticated FPAA capability built to date; we hope that the success of this IC inspires additional devices built in

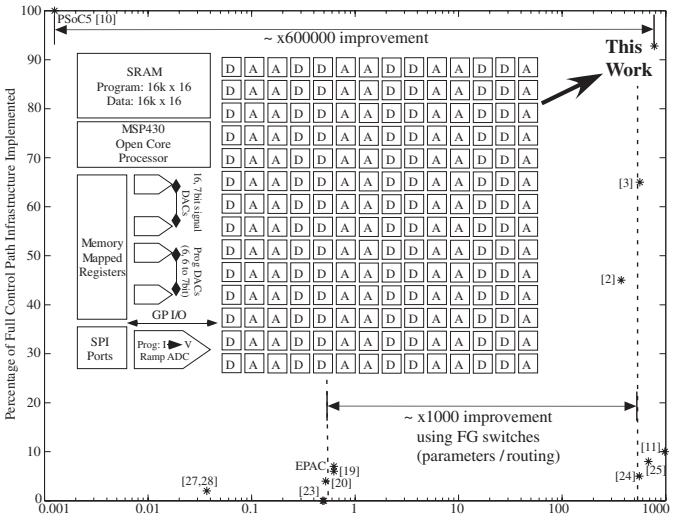

Fig. 10. Using data from generations of FPAA devices, various FPAA devices are plotted as the percentage of control path implemented versus analog parameter density. Recent FPAA ICs, such as the dynamically reconfigurable FPAA or FPAADD device, begin to effectively maximize both parameters. Analog parameter density is the number of analog parameters per mm^2, normalized to a 1-μm process (or analog parameter density). Analog parameters directly set the complexity possible by the particular FPAA device.

the near future. Fig. 9 shows most of the circuits presented, compiled, and experimentally measured in this paper, as well as a summary of the resources used in each case. Fig. 9 shows the table of parameters for the resulting SoC FPAA. Largest signal processing functions shown to date [2], [3], [19] take a small percentage of the available IC.

Fig. 10 plots various FPAA devices showing the percentage of control path implemented versus analog parameter density. Fig. 10 shows two key metrics for FPAA approaches based on a collection of published FPAA devices [20]–[23], [25], [26], [28]. 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. 10 shows FG-based FPAAs enable $\approx$1000 parameter density improvement, providing increased computation on a single device; FG alternatives require a DAC at every node or dynamic techniques.

Table I shows another comparison among FPAA devices in table form, as an updated table following along the comparison made for one of Georgia Tech's first GP FPAA device [11]. Previous papers have more detailed discussion on early FPAA work [11], [26]. Table I shows the impact of the FG-based FPAA devices, compared against other FPAAs, such as the most advanced non-FG FPAA [19]. The highest frequency response devices in [30] and [31] operate at expected frequencies given the IC process, but otherwise, are extremely simple in structure and capability. The maximum analog frequency response is directly related to process technology for devices from 1μm CMOS to 40-nm CMOS.

Designing SoC FPAA devices requires maximizing both metrics, so that we have a large number of programmable parameters and resulting computation, as well as having the infrastructure to get data communicated to these processing devices. Our second metric is to describe the amount of

TABLE I
FPAA COMPARISON TABLE CONSISTENT WITH [11] FOR SIGNIFICANT AND UPDATED FPAA DEVICES

Ref.	Process	Area	Num of CAB/CLB	CAB / CLB elements	CAB lines	Num. of parameters	Architecture	Capability
This Work	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 Circuits Full SP processing
[3]	350nm	25mm^2	108/108	2 OTA, 2 Transistor, 2 Tgate 4, 3-input BLEs	20	80,000	FG, manhattan	Analog / Digital Circuits
[2]	350nm	25mm^2	78/ 0	OTAs, Transistors, FG,	30	76,000	FG crossbar	analog circuits 1 stage analog SP
[11]	350nm	9mm^2	32 / 0	OTAs, Transistors, FG	40	50,000	FG crossbar	analog circuits
[20]	250nm	100mm^2	16 / 0			416	log-domain (I mode) SRAM crossbar	ODE simulation
[31], [30]	130nm	1mm^2	7 / 0	1 Digitally tuned OTAs	4	58	Minimal OTA routing	Basic OTA circuits

control flow (mostly digital) relative to the amount of analog and digital data flow capability. Practically, the ability to get data to all of the processors can be a primary limitation for a series of application spaces, such as image processing, where data does not always arrive in the desired order for the computation. Recent RASP-based FPAA designs [2], [3] have started to focus on improving this second metric while not losing the analog parameter density efficiency. The presented SoC FPAA device maximizes both metrics, being nearly a factor of 500 improvement in area efficiency as typical of other analog FPAA devices, but with high utilization of the resulting computational resources; the closest high utilization structure (i.e., like PSoC5) is nearly a 600 000 factor improvement.

ACKNOWLEDGMENT

The authors thank the efforts developing the open-source MSP 430 development, as well as thank all of the efforts put into the open-source VPR/VTR place and route project.

REFERENCES

[1] *OpenMSP430 Project: Open Core MSP430*. [Online]. Available: http://opencores.org/projectopenmsp430, accessed Nov. 1, 2015.
[2] C. R. Schlottmann, S. Shapero, S. Nease, and P. E. Hasler, "A digitally enhanced dynamically reconfigurable analog platform for low-power signal processing," *IEEE J. Solid-State Circuits*, vol. 47, no. 9, pp. 2174–2184, Sep. 2012.
[3] R. B. Wunderlich, F. Adil, and P. E. 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.
[4] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. E. Hasler, "A 531 nW/MHz, 128 $\times$ 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," in *Proc. IEEE CICC*, Oct. 2004, pp. 651–654.
[5] C. R. Schlottmann and P. E. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: The FPAA implementation," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 1, no. 3, pp. 403–411, Sep. 2012.
[6] B. Marr, B. Degnan, P. E. Hasler, and D. V. Anderson, "Scaling energy per operation via an asynchronous pipeline," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 1, pp. 147–151, Jan. 2013.
[7] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, no. 10, pp. 1629–1636, Oct. 1990.
[8] J. Hasler and B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers Neurosci.*, vol. 7, p. 118, Sep. 2013.
[9] 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.
[10] *PSoC5 Data Sheet*, Cypress Semicond., San Jose, CA, USA, 2011.
[11] 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.
[12] S. Kim, J. Hasler, and S. George, "Integrated floating-gate programming environment for system-level ICs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2015, doi: 10.1109/TVLSI.2015.2504118.
[13] C. M. Twigg, J. D. Gray, and P. E. Hasler, "Programmable floating gate FPAA switches are not dead weight," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2007, pp. 169–172.
[14] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of $O(N)$ complexity," in *Advances in Neural Information Processing Systems 1*. San Francisco, CA, USA: Morgan Kaufmann, 1989.
[15] 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.
[16] P. E. Hasler, P. Smith, C. Duffy, C. Gordon, J. Dugger, and D. V. Anderson, "A floating-gate vector-quantizer," in *Proc. 45th Midwest Symp. CAS*, vol. 1. 2002, pp. I-196–I-199.
[17] S.-Y. Peng, P. E. Hasler, and D. V. Anderson, "An analog programmable multidimensional radial basis function based classifier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 10, pp. 2148–2158, Oct. 2007.
[18] J. Lu, S. Young, I. Arel, and J. Holleman, "A 1 TOPS/W analog deep machine-learning engine with floating-gate storage in 0.13 μm CMOS," *IEEE J. Solid-State Circuits*, vol. 50, no. 1, pp. 270–281, Jan. 2015.
[19] G. E. R. Cowan, R. C. Melville, and Y. Tsividis, "A VLSI analog computer/math co-processor for a digital computer," in *Proc. IEEE ISSCC*, Feb. 2005, pp. 82–83.
[20] 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.
[21] 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. 125–136, Feb. 2002.
[22] E. K. F. Lee and P. G. Gulak, "FPAA based on MOSFET transconductors," *Electron. Lett.*, vol. 28, no. 1, pp. 28–29, 1992.
[23] V. Gaudet and G. Gulak, "10 MHz FPAA prototype based on CMOS current conveyors," in *Proc. Micronet Annu. Workshop*, 1999.
[24] T. S. Hall, C. M. Twigg, P. E. Hasler, and D. V. Anderson, "Application performance of elements in a floating-gate FPAA," in *Proc. ISCAS*, May 2004, pp. II-589–II-592.
[25] C. M. Twigg and P. E. Hasler, "A large-scale reconfigurable analog signal processor (RASP) IC," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sep. 2006, pp. 5–8.
[26] T. S. Hall, C. M. Twigg, J. D. Gray, P. E. 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.
[27] J. Becker and Y. Manoli, "A continuous-time field programmable analog array (FPAA) consisting of digitally reconfigurable G_M-cells," in *Proc. ISCAS*, 2004, pp. I-1092–I-1095.
[28] J. Becker, F. Henrici, S. Trendelenburg, M. Ortmanns, and Y. Manoli, "A FPAA of 55 digitally tunable OTAs in a hexagonal lattice," *IEEE J. Solid-State Circuits*, vol. 43, no. 12, pp. 2759–2768, Dec. 2008.

Suma George, Sihwan Kim, Sahil Shah, Jennifer Hasler, Michelle Collins, Farhan Adil, Richard Wunderlich, Stephen Nease, and Shubha Ramakrishnan photograph and biography not available at the time of publication.