Special Session: Testing and Characterization for Large-Scale Programmable Analog Systems

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

***Abstract*—The growth of large-scale analog and mixed-signal configurable computing systems, requires addressing analog and mixed-signal test questions similar to the development of digital IC testing over the past three decades that includes numerous on-chip self-testing mechanisms. *End-to-end* configurable computing systems require verification using an input sensor or the emulation of an input sensor device and measuring the resulting refined digital, or near digital, output. Verifying and calibrating these configurable systems requires a framework for implementing an application in a once-programmed production devices and general procedure for verifying reconfigurable prototype devices.**

Device and system testing remains a critical aspect of any IC design or system design with new IC designs. For digital design, one major issue is if one millions or billions of transistors is not working properly and therefore a digital gate gives incorrect results during a particular operation. Digital IC testing is a stable field over the last three decades (e.g. [1], [2]), although there was a time when large-scale digital testing was an unsolved area. Digital testing can show that devices that can be used for a particular application, show a device can operate using a range of clock frequencies, or enable finding the reasonable distribution of the maximum IC device operation. Digital ICs include test infrastructure and self testing mechanisms as part of the IC (e.g. [1], [2]). A few aspects could be corrected through testing such as eliminating bad memory blocks.

Classical analog and mixed testing (e.g. [3], [4]) addresses similar questions on typically smaller, and yet more challenging test components due to the range of variations as opposed to a logic gate failure. Component mismatch is the great limiter for analog IC design, primarily as a result of threshold voltage (V_{T0}) mismatch (ΔV_{T0}) between devices, as well as geometry mismatch of capacitors and transistors (and resistors when sparingly used). The variations make test vector generation more complex, although significant efforts are enabling the design of waveforms to identify device issues [5], [6], [7], [8], [9]. These analog systems use a few parameters, primarily for choosing between multiple copies of IC components to minimize mismatch, and stored often in nonvolatile storage using embedded EEPROMs & on-chip Floating-Gate (FG) devices. Minimizing mismatch requires using larger analog components, and mismatch correction techniques adds additional component area. The production characterization time on a commercial tester (C, measured in US cents / second) to characterize the system performance (e.g. INL, DNL of an ADC) is often a significant part of the component cost.

978-1-6654-1060-1/22/$31.00

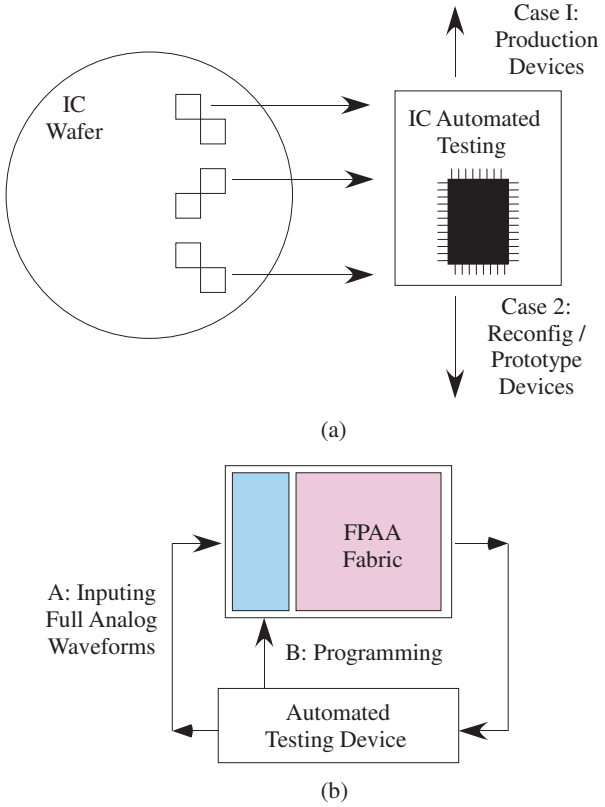

Fig. 1: Testing highly configurable and programmable analog / mixed-signal configurable devices. (a) Testing of Analog / Mixed-Signal devices. includes two important cases, where the first is programming devices for production use (Case I), and the second is testing devices for repeated reconfiguration and devices used for prototyping applications (Case II). (b) Testing a configurable mixed-signal device, such as a large-scale Field-Programmable Analog Array (FPAA) like an SoC FPAA device, requires accounting for the time to input and measure the full analog waveforms for testing the device, as well as accounting for the programming time for setting up a device.

And yet, with the significant interest and growth in large-scale configurable systems, systems involving analog computing often within a mixed-signal infrastructure. Large-scale analog programmability enables a range of large-scale analog computing concepts including Large-Scale Field Programmable Analog Arrays (FPAA) [10]. Currently systems utilize 1 million parameters (350nm CMOS) and scaled systems (40nm, 14nm CMOS) are expected to have billions of parameters [11]. Testing of these fine-grained configurable mixed-signal devices is essential for their use and development, although it might seem to be an overwhelming problem.

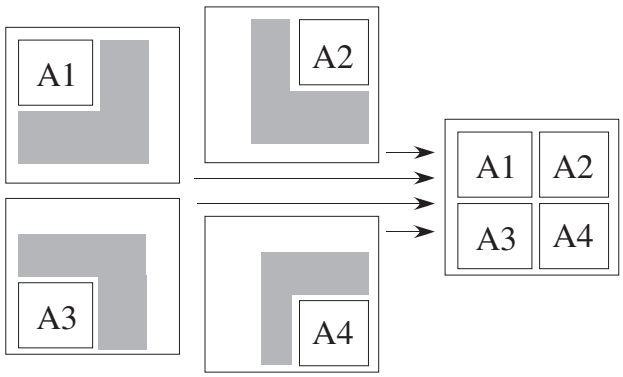

Fig. 2: Several self-test structures (shaded regions) can be initially targeted and measured on an FPAA device for core FPAA components (e.g. A1, A2, A3, A4), then target the full system modifying the system parameters based on these initial tests. The deployed area cost of these test-structures is negligible.

Although most of the current development does not utilize commercial testing equipment, one expects significant interest to integrate the integrated FPAA testing [12] and tool [13], [14] infrastructure with commercial testing (Fig. 1). This discussion will work through a framework for verifying and calibration of configurable analog computing systems (Sec. I), then overview FG and FPAA devices (Sec. II) to move towards optimizing the programming time for large-scale FG systems to minimize one aspect of commercial testing time (Sec. III).

These configurable systems are often used for *end-to-end* computing systems that start from a sensor device and output a refined representation, such as a classified result. Testing these systems requires an electrical representation of the input sensor that can be compiled into the IC (e.g. a capacitor to represent a sensor capacitor) as part of the test structure, or simply require one or a vector of analog input voltages from a stored signal played through a DAC (e.g. acoustic, image sensors). End-to-end systems provide efficient platforms for analog computation as the inputs typically start as an analog result, and the digital or near digital outputs minimize the number and complexity of output data converters. Having an analog system as a co-processor with a huge bank of data converters tends to not provide most of the benefits of analog computation.

I. Framework for Verification and Calibration of Large-Scale Configurable Analog Systems

Valid configurable (e.g. FPAA) requires testing and programming for both once-programmed production devices as well as reconfigurable prototype devices (Fig. 1). In the first case (Fig. 1a, Case 1), the measurement and calibration is limited to what is necessary for that application, and in the second case (Fig. 1b, Case 2), the measurement and calibration requires a general procedure to verify the entire device. Both cases require the setup time for the equipment connected to a production digital tester.

The first case (Case 1) stamps a product into the configurable device. Creating a set of single programmed devices (Case I) requires testing a single application with the required input signal and comparisons to the desired output signals. Testing cost (C) is proportional to testing time that is the combination of the time for Inputing Full Analog Waveforms (A, Fig. 1b) into the device under test for each iteration, and the time for Programming (B, Fig. 1b) or reprogramming the device for each iteration. Testing time is correlated with the application frequency range. An acoustic processing device requires time for the lowest frequency (e.g. 20-100Hz) signals to be part of the computed output, where seconds of testing time is not unusual for many of these applications. The tester time, and resulting tester cost, is highly dependent on the application. The golden system was generated likely with a Case 2 tested device with the associated design tools.

The second case (Case 2) enables an often-reconfigurable product for prototyping or in-field reprogramming. A generally programmed highly configurable device requires initial verification and calibration for proper functioning of the programming infrastructure, as well as calibrating any device that has configurable parameters different from the programming expectations. Mapping potential component mismatches requires testing programmable devices. Initial development using this device would optimize the development for Case 1.

Both cases benefit from compiled self-test approaches are compiled into the device for measurement and calibration (Fig. 2). Initial self-test algorithms and measurements [15], including an optimized linear Vector-Matrix Multiplier (VMM) [16], show the opportunities and potential algorithms for these demonstrations. Repeatable and reliable FG programming allows initial components can be compiled and optimized, and enables the entire system to be compiled using these new optimized parameters with no additional required area cost. These steps do require additional tester time (and cost) for the additional programming and measurement, further requiring fast FG programming infrastructure. These approaches utilize programmable circuits with low temperature sensitivities even using subthreshold transistors [17], [18], enabling robust analog computing engines not constrained to a 2-3 degree range (e.g. [19]).

II. Overview of Floating-Gates (FG) and FPAAs

Reviewing the basic properties of FG devices sets the framework for the verification opportunities and challenges. A FG device (Fig. 3) is a combination of a capacitive divider and pFET transistor. The capacitive divider is modeled as

$$V_{fg} = \frac{C_1}{C_T} V_g + V_{offset} \tag{1}$$

where C_T is the total capacitance at the floating-gate node ($C_T = C_1 + C_w$), and V_{offset} is due to charge at the floating-gate node. A Saturated pFET transistor as a function of the gate (V_{fg}) and source (V_s) nodes and the well voltage at V_{dd}) is (ignoring drain effects)

$$I = I_{th} e^{(\kappa(V_{dd} - V_{fg} - V_{T0}) - (V_{dd} - V_s))/U_T} \tag{2}$$

where κ is the V_{fg} to surface potential coupling, and U_T is the thermal voltage (kT/q $\approx$ 25mV at 300K). The combined

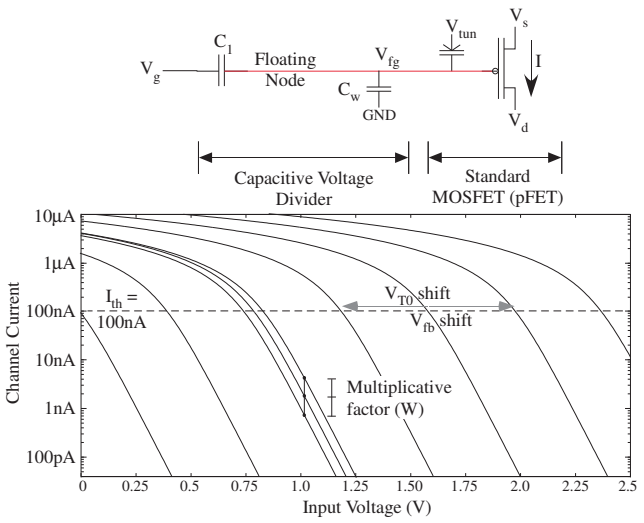

Fig. 3: Illustration of a FG device that is a combination of a capacitive voltage divider and a standard pFET transistor in a standard CMOS process. The current-voltage curves for this device are similar to a typical transistor except that the gate to surface potential (κ) decreases by the larger capacitive divider, and the stored charge provides a free parameter that can be viewed as a flatband voltage (V_{fg}) or V_{T0} shift) or as a multiplicative subthreshold factor.

model for the FG pFET device is

$$I = I_{th} e^{(\kappa(V_{dd} - V_{offset} - V_{T0}) - \kappa_{eff} V_g - (V_{dd} - V_s))/U_T} \quad (3)$$

κ_{eff} is an effective κ equal to $\kappa C_1/C_T$. A FG device is a free parameter, where that parameter could be either a variable V_{T0} (effectively flatband voltage (V_{fb})), or a multiplicative weight in subthreshold operation (FIg. 3). FG elements enable the wide programmability in the FPAA devices [10], [11], including computing in the routing achieved by enabling programming of FG nodes outside of the operating power supply as the floating-node does not have a DC path to GND. FPAA arrays include combinations of analog and digital components (Fig. 4) within the same fabric as well as additional edge devices (e.g. μP and DACs). The crossbar routing components can be utilized as analog computing elements, further expanding the verification space [20].

A device is considered not working when it is out of specification, and yet, if there is a way to bring that device into useful operation, the overall yeilds and performance improve. For example, CMOS imagers are concerned about dead pixels that are effectively pixel elements with parameters operating out of specification. Imagers with analog programmable FG devices can access the entire array of pixels, eliminating effectively any identified dead pixel [21]. The analog computation, and the direct nonvolatile analog correction of the primary source of mismatch (ΔV_{T0}), results in near ideal yields if only there is sufficient time available to measure and correct these devices. In general, we rarely find non-functional devices in a configurable array even after considerable classroom student stress on these devices; devices are more likely to fail because of damage to other board components or damage to pin components. The programmed FG elements hold charge within 1 to 100μV accuracy over 10 year lifetimes [22], [23]. These questions directly lead to asking the speed of programming and of the measurements to improve the programming.

III. Optimizing Large-Scale FG Programming Time

On-chip FG Programming and Measurement time directly impacts the device cost (Fig. 1b); all digital FG programming interface both minimizes tester complexity as well as minimizes tester time (e.g. [24]). Generalized FG programming of heterogeneous components transforms the general configuration of components (Run mode) into a crossbar or island of crossbars enabling individual measurement and programming selectivity of any element in the array (Fig. 4) [25], [24]. Hot-electron injection programming enables precise and nearly ideal selectivity measuring and programming FG devices, while the typical mismatch for electron tunneling tends to reserve this function for block erase of these devices. To minimize the number of erasing and full reprogram cycles, programming iterations should be designed to undershoot programming targets to enable fast programming of tuned parameters.

Device current measurement requires the longest amount of time for a programming iteration. The measured output device current is transformed into a voltage and then converted into a digital value to compare with a desired target result (Fig. 4). The measurement system (Fig. 4) can either measure each FG device in a serial manner, typical to existing SoC FPAA devices [10], [24], or could parallelize the current to digital measurement by measuring each row or measuring blocks in parallel [25] (Fig. 4). The added area for the parallel measurement infrastructure is a small additional cost that is offset by the significantly reduced tester time. Typically the cost of one-second of tester time is in the ballpark of 250μm x 250μm of die are in a typical 350nm CMOS process.

FG devices can correct for device mismatch either in production programming or for an often reprogrammable device given a calibration step. This calibration step requires tester time, as well as user storage of the mismatch parameter. One FG calibration example is the use of direct vs. indirect FG configurations (Fig. 5). Indirect programming, where a different pFET device is used for operation as compared with measurement and programming, significantly decreases the number of reconfiguration switches, including switches in the datapath, while incurring ΔV_{T0} mismatch, and to a lesser extent κ mismatch, between the two pFETs that must be calibrated. Direct programming eliminates this ΔV_{fg} mismatch, and resulting calibration, while requiring additional switches. This issue directly illustrates the tradeoff between the cost of die area and the cost of tester time. Several circuits currently utilize indirect programming (Fig. 5), such as the SoC FPAA switches [10] enabling a wide programming range (pA to 100s fo μA), differential pairs with FG input transistors [10], and realistic neuron models [26]. Going forward, one might expect to minimize the use of indirect programming except where the resulting die cost or loss of circuit performance makes using direct programming prohibitive. With CMOS

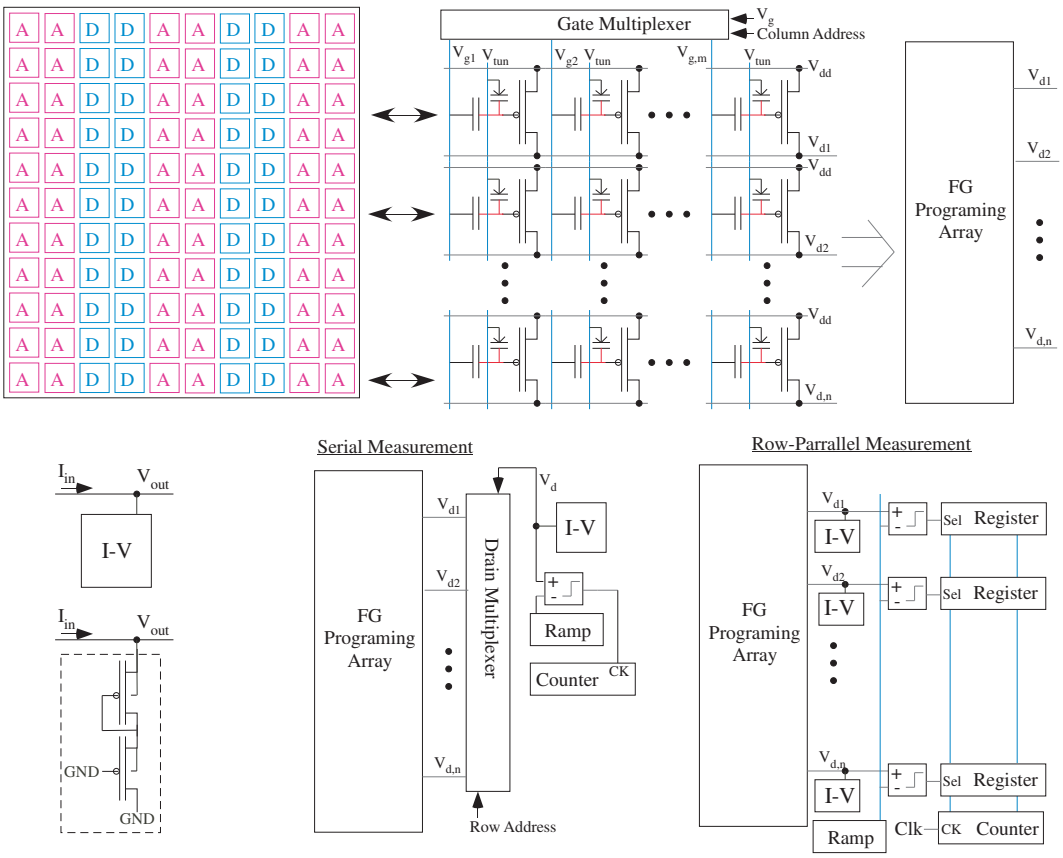

Fig. 4: Programming a configurable analog (A) and digital (D) array, typical of an SoC FPAA, requires reconfiguring the FG array elements to a crossbar of devices or islands of crossbar devices. At the row-wise drain terminals for the programming array, one can have a *serial measurement* of each FG device or a *row-parallel measurement* of the FG devices. Measuring the FG device requires the longest time of any programming step, even with the on-chip current-to-voltage (I-V) transistor element.

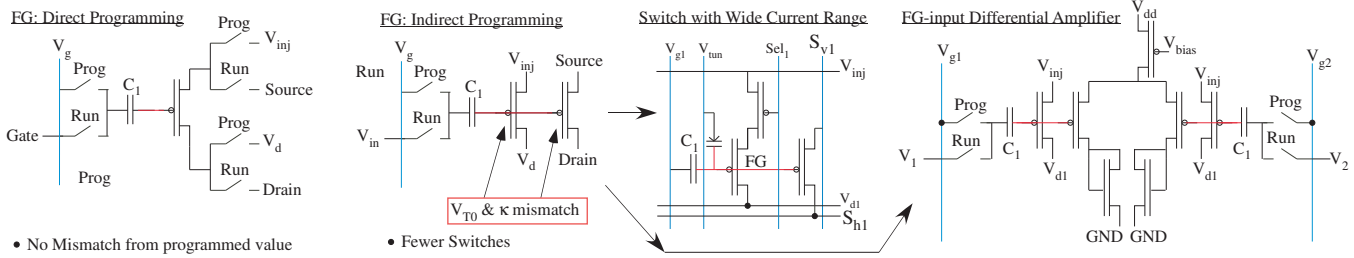

Fig. 5: The tradeoff between Direct and Indirect FG programming. Direct FG programming results in little to no mismatch between the programmed and operating FG values for a single device, although requiring more switches in a general configuration to switch between the operating mode (Run) and programming mode (Prog). Many configurations could move these switches to the edge of an array or island of devices. Indirect FG programming significantly reduces the number of switches required, including switches directly in the device datapaths that could affect performance, but must account for mismatch in the threshold voltage (V_{T0}) and gate to surface-potential coupling (κ). Indirect FG programming enables the FG *switch with wide current range* and *FG-input differential amplifier* used throughout FG FPAA implementations.

scaling, additional switches are less costly, as well as future architectures using islands of FG elements benefit less from indirect programming architectures.

IV. Summary and Discussions

This discussion showed a framework to address analog and mixed-signal verification for large-scale analog and mixed-signal configurable computing systems, potentially enabling similar test methods to digital IC testing. *End-to-end* configurable computing systems require verification from an input sensor or emulated input sensor compiled on the configurable device through the refined digital or near-digital output. Two frameworks enable verification both for implementing once-programmed production devices as well as for calibrating reconfigurable prototype devices. This discussion showed a framework for verifying and calibration of configurable analog computing systems, as well as the impact of FG elements in these configurable (e.g. FPAA) devices, and the aspects required to optimize the programming time to help minimize commercial testing cost.

Testing of end-to-end programmable and configurable systems opens a number of potential opportunities. The high-level of flexibility of these configurable analog devices [11] enables a number of new applications such as secure applications

that are defined at testing time. Assuming the verification of the initial programming structure, the entire structure can be directly measured post-fabrication using existing FPAA programming and measurement techniques. Further, having regions with mismatch that can be calibrated enables a range of security applications (e.g. PUF type circuits)

REFERENCES

[1] B. Konemann, G. Zwiehoff, and J. Mucha, "Built-in test for complex digital integrated circuits," *IEEE Journal of Solid-State Circuits*, vol. 15, no. 3, pp. 315–319, 1980.

[2] V. D. Agrawal, C.-J. Lin, P. W. Rutkowski, S. Wu, and Y. Zorian, "Built-in self-test for digital integrated circuits," *AT T Technical Journal*, vol. 73, no. 2, pp. 30–39, 1994.

[3] L. Scheffer, L. Lavagno, and G. Martin, *EDA for IC System Design, Verification, and Testing (Electronic Design Automation for Integrated Circuits Handbook)*. USA: CRC Press, Inc., 2006.

[4] G. Roberts, F. Taenzler, and M. Burns, *An Introduction to Mixed-signal IC Test and Measurement*, ser. The Oxford series in electrical and computer engineering. Oxford University Press, 2012. [Online]. Available: https://books.google.com/books?id=sZZUXwAACAAJ

[5] A. Chatterjee, S. Deyati, B. Muldrey, S. Devarakond, and A. Banerjee, "Validation signature testing: A methodology for post-silicon validation of analog/mixed-signal circuits," in *International Conference on Computer Aided Design*. ACM, 2012, pp. 553–556.

[6] B. Muldrey, S. Deyati, M. Giardino, and A. Chatterjee, "Ravage: Post-silicon validation of mixed signal systems using genetic stimulus evolution and model tuning," in *VLSI Test Symposium (VTS), 2013 IEEE 31st*, 2013, pp. 1–6.

[7] S. Deyati, B. J. Muldrey, A. Banerjee, and A. Chatterjee, "Atomic model learning: A machine learning paradigm for post silicon debug of rf/analog circuits," in *2014 IEEE 32nd VLSI Test Symposium (VTS)*, 2014, pp. 1–6.

[8] S. Deyati, B. J. Muldrey, and A. Chatterjee, "Adaptive testing of analog/rf circuits using hardware extracted fsm models," in *2016 IEEE 34th VLSI Test Symposium (VTS)*, 2016, pp. 1–6.

[9] B. Muldrey, "Mixed signal design validation using reinforcement learning guided stimulus generation for behavior discovery," in *Proceedings IEEE VLSI Test Symposium*. IEEE, 2019.

[10] J. Hasler, "Large-scale field programmable analog arrays," *IEEE Proceedings*, vol. 108, no. 8, pp. 1283–1302, 2020.

[11] ——, "The rise of soc fpaa devices," in *CICC*, 2022.

[12] J. Hasler, S. Kim, S. Shah, F. Adil, M. Collins, S. Koziol, and S. Nease, "Transforming mixed-signal circuits class through SoC FPAA IC, PCB, and Toolset," in *IEEE European Workshop on Microelectronics Education*, Southampton, May 2016.

[13] M. Collins, J. . Hasler, and S. George, "An open-source toolset enabling analog–digital software codesign," *Journal of Low Power Electronics Applications*, vol. 6, no. 1, pp. 1–15, 2016.

[14] J. Hasler and A. Natarajan, "An open-source toolset for fpaa design," in *WOSET*, 2020.

[15] S. Shah and J. Hasler, "Tuning of multiple parameters with a bist system," *Journal of Low Power Electronics Applications*, vol. 64, no. 7, pp. 1772–1780, 2017.

[16] A. Natarajan and J. Hasler, "Built-in self-test of vector matrix multipliers on a reconfigurable device," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2020, pp. 1–5.

[17] S. Shah, H. Toreyin, J. Hasler, and A. Natarajan, "Models and techniques for temperature robust systems on a reconfigurable platform," *Journal of Low Power Electronics Applications*, vol. 7, no. 21, pp. 1–14, 2017.

[18] ——, "Temperature sensitivity and compensation on a reconfigurable platform," *IEEE Transactions VLSI*, vol. 26, no. 3, pp. 604–607, 2018.

[19] E. Kauderer-Abrams, A. Gilbert, A. Voelker, B. Benjamin, T. C. Stewart, and K. Boahen, "A population-level approach to temperature robustness in neuromorphic systems," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4.

[20] C. Twigg, J. Gray, and Hasler, "Programmable floating-gate FPAA switches are not dead weight," in *2007 IEEE International Symposium on Circuits and Systems*, 2007, pp. 169–72.

[21] A. Bandyopadhyay, J. Lee, R. Robucci, and Hasler, "Matia: a programmable 80 μwframe cmos block matrix transform imager architecture," *IEEE Journal of Solid-State Circuits*, vol. 41, no. 3, pp. 663–672, 2006.

[22] V. Srinivasan, G. J. Serrano, J. Gray, and P. Hasler, "A precision CMOS amplifier using floating-gate transistors for offset cancellation,," *IEEE JSSC*, vol. 42, no. 2, pp. 280–291, 2007.

[23] V. Srinivasan, G. Serrano, C. Twigg, and P. Hasler, "Floating-gate-based programmable cmos reference," *IEEE Transactions CAS I*, vol. 55, no. 11, pp. 3448 – 3456, 2008.

[24] S. Kim, J. Hasler, and S. George, "Integrated floating-gate programming environment for system-level ics," *IEEE Transactions VLSI*, vol. 24, no. 6, pp. 2244–2252, 2016.

[25] M. Kucic, *Analog Computing Arrays*. Ph. D. thesis. Georgia Institute of Technology, 2004.

[26] S. Brink, S. Nease, Hasler, S. Ramakrishnan, R. Wunderlich, A. Basu, and B. Degnan, "A learning-enabled neuron array ic based upon transistor channel models of biological phenomena," *IEEE Transactions on Biomedical Circuits and Systems*, vol. 7, no. 1, pp. 71–81, 2013.