A MITE-Based Translinear FPAA

Craig R. Schlottmann, *Student Member, IEEE*, David Abramson, and Paul E. Hasler, *Senior Member, IEEE*

***Abstract*—While the development of reconfigurable analog platforms is a blossoming field, the tradeoff between usability and flexibility continues to be a major barrier. Field Programmable Analog Arrays (FPAAs) built with translinear elements offer a promising solution to this problem. These FPAAs can be built to use previously developed synthesis procedures for translinear circuits. Furthermore, large-scale translinear FPAAs can be built using floating-gate transistors as both the computational elements and the reconfigurable interconnect network. An FPAA built using Multiple Input Translinear Elements (MITEs) has been designed, fabricated in 0.35 μm CMOS, and tested. These devices have been programmed to implement various circuits including multipliers, squaring circuits, RMS-to-DC converters, and filters. In addition, synthesis, place-and-route, and programming tools have been created in order to implement a reconfigurable system where the circuits implemented are described only by equations. The continued development of translinear FPAAs will lead to a reconfigurable analog system that allows for a large portion of the design to be abstracted away from the user.**

***Index Terms*—FPAA, field-programmable analog array, programmable analog, MITE, translinear.**

I. Analog Reconfigurability and Design Abstraction

ONE of the biggest breakthroughs in the field of digital integrated circuits has been the field-programmable gate array (FPGA). This is not only because they enable rapid prototyping, but also because they open up the use of digital circuits to those without expertise in the field. While field-programmable analog arrays (FPAAs) are attempting to fill a similar void in the analog field, they have not been developed to a point where they are being adopted by designers. FPAAs are being developed at a time when analog signal processing is on the rise due to the power savings it offers over traditional digital solutions in certain situations [1]. In addition to offering significant power savings, a reconfigurable analog platform would allow the user to prototype designs, cutting down on the fabrication cycle and facilitating a faster time to market.

In this paper, we present the MITE FPAA (MFPAA), which utilizes Multiple Input Translinear Elements (MITEs) as the core computational unit. We have developed a novel MITE unit which takes advantage of typically fixed nodes while still fitting into a reconfigurable framework. By carefully designing this hardware structure, we were able to fully utilize existing synthesis algorithms for large-scale MITE systems. This novel architecture allows for a synthesis that is elegant in its simplicity and lets us fully abstract the circuit design from the user. Thus, by using this full-system approach to FPAA design, we have created a complete tool chain: the abstracted software design environment, the place-and-route and programming tools, and the analog hardware. This entire platform will open up MITEs to new audiences as a design tool for implementing low-power signal-processing systems.

A. Questions of Analog Reconfigurability

While FPGAs have been developed for commercial use, FPAAs have not had the same success. The chief reasons for this is the lack of an universal block from which analog circuits could be systematically built, as gates are to digital circuits [2]. This can be seen by comparing FPAAs currently on the market or under development. Anadigm's FPAA and their software package, Anadigm Designer, use switched-capacitor circuits to realize the users desired circuit [3]. On the other end of the granularity spectrum are Field Programmable Transistor Arrays (FPTAs), which use transistors that must be connected together with switches to realize the user's circuit [4]. In addition, there are FPAAs that are built using only $g_m - C$ filters [5] and transconductors [6]. Recently, Reconfigurable Analog Signal Processor (RASP) has been trying to solve this problem by using a mixture of analog blocks to realize circuits [7].

This inherent tradeoff between flexibility and the appropriate level of abstraction is limiting the usefulness of FPAAs. Most of the current FPAAs tend to one of the extremes in this tradeoff. For example, FPTAs are highly flexible but offer almost no real level of abstraction, whereas the $g_m - C$ based FPAAs that have high abstraction levels, filter designs, but do not have any true flexibility.

This tradeoff is also seen in the tools used to interface with the reconfigurable platform. For example, platforms without an appropriate level of abstraction struggle to incorporate any type of synthesis into their tools, while platforms with high levels of abstraction and limited flexibility can include synthesis in their tool packages but for very narrow scopes. This lack of synthesis tools is painfully clear in the current state of FPAA systems. While there have been a couple design environments reported, such as the RASP Simulink tool [8] and Anadigm Designer, which allow you to graphically configure the FPAA's components, the majority of FPAA systems report no such tools. Thus, in most cases, the user is forced to manually route their system with the use of fuse charts.

The use of translinear circuits as the universal analog block to reduce the tradeoff between flexibility and abstraction level has been gaining a lot of recent attention [9]–[11]. Using translinear circuits for which known network synthesis procedures exist [12], [13], it is possible to build a system in which the only input necessary is the set of equations that describe the system to be

Manuscript received March 24, 2010; revised July 04, 2010, September 27, 2010; accepted September 30, 2010. Date of publication November 29, 2010; date of current version December 14, 2011.

C. Schlottmann and P. Hasler are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-250 USA (e-mail: cschlott@gatech.edu phasler@ece.gatech.edu).

D. Abramson is with Texas Instruments Inc, Manchester, NH 03101-3105 USA (e-mail:abramson@ece.gatech.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2010.2089705

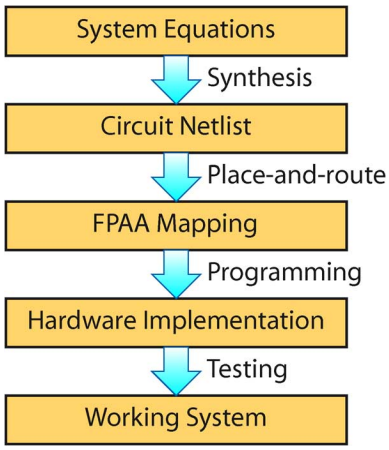

Fig. 1. Design flow using a translinear FPAA. Using translinear circuits allows the user to enter a set of equations which is then netlisted using existing synthesis procedures. The circuit is then place-and-routed, and the system is programmed onto the FPAA.

implemented. The translinear FPAA will be able to implement a wide range of circuits, including all linear static equations and most differential equations, while requiring the user to perform no actual analog design. This idea is illustrated by the translinear FPAA design flow, shown in Fig. 1. Unlike the traditional FPAA design flow, there are no design or simulation steps required to create the working system. This will allow users with a background in math, controls, physics, or many other fields to easily interact with the FPAA.

II. Multiple Input Translinear Elements

Ideal translinear elements have infinite input impedance and an exponential voltage to current relationship independent of the current level at which they are operating. In addition, any translinear element can be made to have multiple inputs by simply applying resistive or capacitive division at the voltage input. MITEs can thus be built using either subthreshold MOSFETs or BJTs, each of which is stronger in one of the two above specifications [12]. In order to allow for the practical implementation of our FPAAs in a simple digital process, we have chosen to use subthreshold pFETs. This pFET has a current that is exponentially related to its gate voltage given by

$$I_d = I_0 e^{\frac{V_s - \kappa V_g}{U_T}} \left(1 - e^{\frac{V_d - V_s}{U_T}}\right) \tag{1}$$

where I_0 is a pre-exponential constant term, κ is the capacitive division between the oxide capacitance and the depletion capacitance, and U_T is the thermal voltage, kT/q. Note that all voltages are referenced to the bulk, which is the well voltage for the pFET. Furthermore, as long as the device is in saturation, $V_{\rm sd} > 100$ mV, the second exponential term can be neglected.

Fig. 2(a) shows the subthreshold pFET realization of a MITE, with capacitive division is used for the introduction of multiple inputs. The current-voltage relationship for this element is given by

$$I_d = I_0 e^{\frac{V_s - \kappa \sum (w_i V_i)}{U_T}} \tag{2}$$

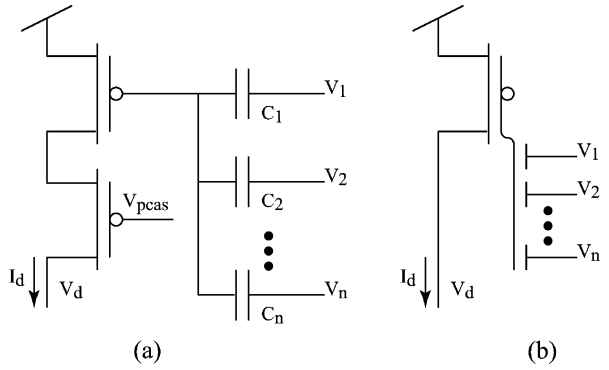

Fig. 2. Subthreshold pFET realization of a MITE. (a) Components used to realize a MITE in a standard CMOS process. (b) Symbol used to represent a MITE.

where w_i, the dimensionless weight applied to an input, is given by $(C_i)/(C_T)$ where C_T is the total capacitance at the gate of the pFET. Fig. 2(b) shows the symbol that will be used for this realization of a MITE. Note that while the subthreshold MOSFET does have nearly infinite input impedance, the range in which the relationship between current and voltage is exponential is limited. However, by making the W/L ratio of the MITEs larger, this range can be increased.

To precisely set the charge on the floating node of the floating-gate pFET, two methods of programming are used: Fowler-Nordheim tunneling and hot-election injection. This method of programming is vastly superior to simply removing the charge with UV radiation, because the charge can be precisely set, thus removing any offset between devices. Historically, gain errors induced by charge mismatch between devices have had a crippling affect on large-scale MITE systems [14]. In order for the MITE to be compatible with the FPAA programming core, we have developed a specialized MITE structure as described in [15]. Of particular importance, the use of this on-chip programming core comes at no additional overhead as it is already built in to program the floating-gate switches [16].

A. Building Blocks of MITE Systems

In order to build complex systems using MITEs, it is necessary to explore what higher level components are commonly used. Translinear loops and log-domain filters will be emphasized because they are commonly used as core elements in most synthesis procedures.

1) Translinear Loops: Translinear loops are well documented building blocks of almost every translinear system [12], [17]. In a reconfigurable system, fixed loops are used to reduce the amount of reconfigurability needed. For our reconfigurable system we will use the translinear loop shown in Fig. 3(a), which can be analyzed by simply solving for each MITE's diode connected voltage. For our analysis, we can assume that the floating gates have an equal amount of charge on them and that both of the MITE's input capacitors are equal ($\kappa w_1 = \kappa w_2 \equiv w$). Under these assumptions (with $V_{\rm ref} \equiv V_0$), the equations are

$$V_i = \frac{U_T}{w} \log \frac{I_i}{I_0} - V_{i-1}, \quad i = 1, 2, 3 \tag{3}$$

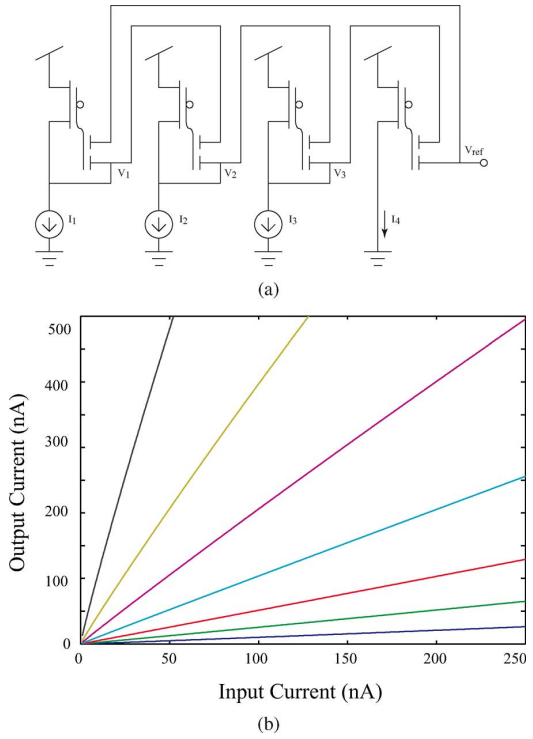

Fig. 3. MITE implementation of a 2nd-order translinear loop. (a) Schematic of a 2nd-order translinear loop. (b) Simulation results of the translinear loop. The multiplication coefficients were chosen to be $(1/10)$, $(1/4)$, $(1/2)$, 1, 2, 4, and 10.

$$V_3 + V_0 = \frac{U_T}{w} \log \frac{I_4}{I_0}. \tag{4}$$

Substituting (3) into (4) gives

$$\log \frac{I_1}{I_0} + \log \frac{I_3}{I_0} = \log \frac{I_2}{I_0} + \log \frac{I_4}{I_0} \tag{5}$$

which can also be written as

$$I_1 I_3 = I_2 I_4. \tag{6}$$

This circuit is most often used as a multiplier with

$$I_{\text{out}} = \frac{I_a I_b}{I_c}. \tag{7}$$

Simulation results of the translinear loop are shown in Fig. 3(b). Data was taken as I_a was swept and the coefficient I_b/I_c was held constant. For higher coefficients, the trace is not completely straight because the MITEs leave the subthreshold region due to the higher current levels. The dynamic range (DR) for such a system follows the discussion given in [18].

2) Filters: Log-domain filters were included in our system as higher level blocks because they are a building block of almost every dynamic system and are commonly utilized in synthesis procedures. The synthesis of the circuit, found in [12], is similar to the synthesis of the loop, but first the constraint equations are needed. The differential equation for a first-order low-pass filter is

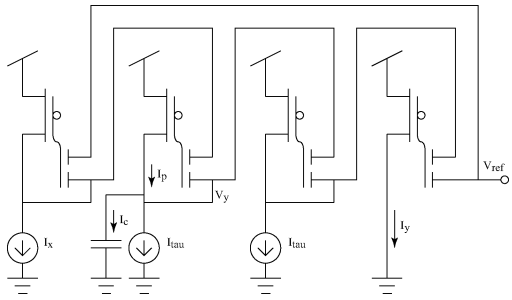

Fig. 4. MITE implementation of a 1st-order low-pass log-domain filter. The I_τ bias current connected to the capacitor is used to set the corner frequency of the filter. The second bias current is set to I_τ in order to maintain unity gain.

$$\tau \frac{d}{dt} I_y + I_y = I_x \tag{8}$$

where I_x is the input current, I_y is the output current, and τ is the time constant of the filter. The chain rule can be applied to the derivative of the current giving

$$\tau \frac{\partial I_y}{\partial V_y} \frac{dV_y}{dt} + I_y = I_x \tag{9}$$

where V_y is the log compressed voltage associated with I_y. Taking the derivative of the current through the 2-input MITE with respect to a single controlling voltage results in

$$-\tau \frac{w I_y}{U_T} \frac{dV_y}{dt} + I_y = I_x \tag{10}$$

where w is the weight of the controlling voltage V_y. Noting that $C(dV_y)/(dt)$ is a capacitive current (I_c) and $(\tau w)/(U_T C)$ can be written as a reciprocal of a bias current (I_τ^{-1}) we can arrange (10) as

$$I_\tau - I_c = \frac{I_x I_\tau}{I_y}. \tag{11}$$

This equation is implemented by the circuit in Fig. 4, where the right hand side is the same as the loop derived in (7), and the left hand side is simply the KCL of I_p. In addition, a gain term can be added to the transfer function by multiplying the second I_τ, the bias current for the MITE without the capacitor on its drain, by the coefficient desired.

III. Reconfigurable Architecture

The MFPAA utilizes the base architecture developed for the general RASP 2.8 line of FPAAs [19]. This results in a system which is a vast advancement over the Reconfigurable Analog Array of MITEs [9] by using a more computationally efficient MITE element, incorporating a more complex routing scheme in order to reduce the parasitic capacitance of the switch matrixes, and utilizing on-chip programming [16].

A. System Architecture

The architecture of the MFPAA is shown in Fig. 5. The FPAA is laid out with 18 CABs in a 6×3 array, with 17 being MITE CABS and one being the I/O CAB. The RASP infrastructure incorporates a cross-bar switch matrix for connecting the elements

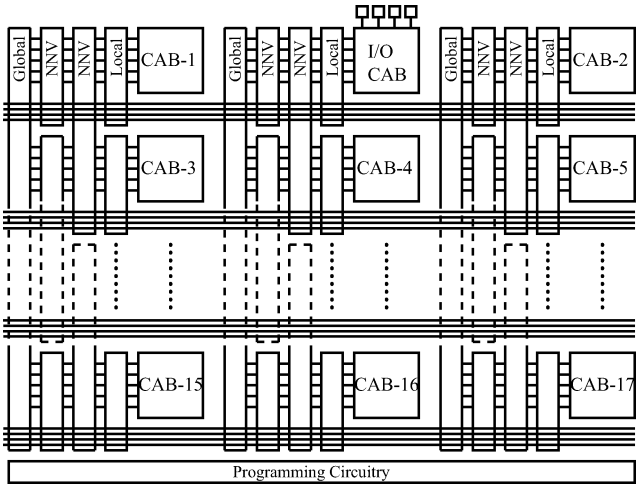

Fig. 5. System architecture of the improved MITE FPAA. The FPAA consists of 17 MITE CABs and a single IO CAB. The vertical routing between CABs is organized into local, nearest-neighbor vertical (NNV), and global. The horizontal routing is only global.

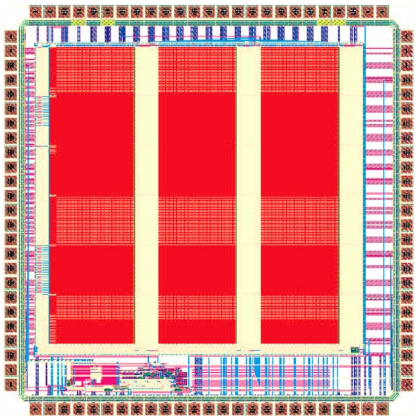

Fig. 6. Layout of the MFPAA. The FPAA was fabricated in a 0.35 μm standard CMOS process on a 3 mm $\times$ 3 mm die.

to one another. The connection between the horizontal and vertical lines is controlled by a single floating-gate switch, which allows it to store its own value without a separate memory.

Within each CAB, the vertical routing is organized into 10 global, 20 nearest neighbor (10 up, 10 down) and 10 local lines. The shorter lines are used whenever possible to reduce parasitic line capacitance. Each CAB also has 10 global horizontal lines. At the lower end of the IC is the on-chip programming structure, which selects and programs all necessary floating-gate switches and MITEs. The layout of the MFPAA is shown in Fig. 6, which was fabricated in 0.35 μm standard CMOS with a $V_{\rm dd}$ of 2.4 V.

B. The MITE CAB

The most significant advancements in the architecture of the MFPAA are within the MITE CAB. In order to improve the density of computation elements to switch elements, single MITEs must be replaced with computational blocks with less reconfigurability. In order to avoid losing flexibility, the new computation element, shown in Fig. 7, was chosen by trying to maximize the number of equations the element could implement while minimizing the reconfigurability needed. This structure

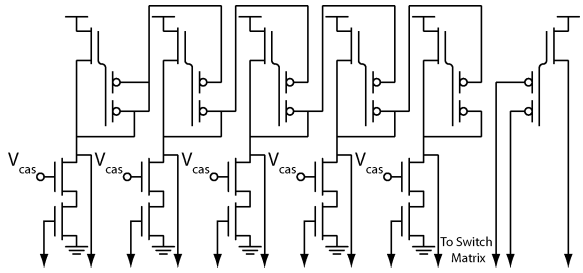

Fig. 7. Basic MITE computation element of the improved MITE FPAA. The computation element consists of 5 input MITEs in a translinear loop configuration and 1 output MITE. The gates of the output MITE are sent into the switch matrix where they are connected to any of the input MITE gate voltages.

is similar to the one analyzed in Section II.A.1, with $V_{\rm ref}$ taken from V_1. Two of these elements, called MITE Computational Elements (MCEs), are contained in each CAB.

The CAB also includes a first-order log-domain filter, shown in Fig. 14(a). This is the same structure discussed in Section II.A.2, with $V_{\rm ref}$ taken from $V_{\rm tau}$. Again, this was done to increase the density of the computational elements without losing too much reconfigurability. This also lends itself to implementing previously developed synthesis procedures on the MFPAA, as dynamic functions can be implemented by combining static functions with first-order filters [13]. Both the MCE and filter were drawn with $w/l = (48\ \mu m)/(1.2\ \mu m)$ to increase the subthreshold range.

In addition to the two MCEs and the filter, the CAB includes six bias current generators, six nFET current mirrors, and a cascode-bias generator. The bias currents are programmed with floating-gate current sources and are used for implementing coefficients and scaling currents in the input equations. The current mirrors are used for adding and subtracting as well as signal routing. The cascode-bias generator, based on Brad Minch's design [20], creates all of the cascode biases needed.

C. The I/O CAB

The input/output (I/O) CAB is the CAB that interfaces the MITE systems to the outside world. This CAB contains input voltage-to-current (VI) converters, output drivers, and broadcast drivers for inputs. The chip was designed with banks of 10 of each of these components. The VI converter is necessary because MITEs are mainly current-mode elements, but it is much easier to generate voltage-mode signals off chip, via DACs or function generators. The output driver is a current mirror with a gain factor of 10 to help off-chip current meters read the subthreshold MITE currents. In this system, we chose not to incorporate a IV-ADC because it was easy enough for us to read currents with off-the-shelf instruments. We will pursue adding this capability to future systems to allow interfacing with a programmable processor. The broadcast driver is equivalent to half of a current mirror to log-compress the current into a gate voltage by a diode connected nFET, which can then be broadcast to many input nFET devices.

The VI converter on the MFPAA was designed for both accuracy and speed considerations. The VI must be able to convert currents on the order of nanoamps without sacrificing the speed of the entire system. This requires an extremely low input resistance to compensate for the large capacitance of the bonding

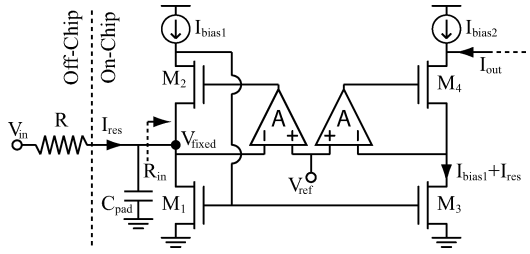

Fig. 8. VI converter used in the improved MITE FPAA. The amplifier on the input side provides an extremely low input resistance allowing for high speed and good accuracy. The amplifier on the output side reduces mismatch between the input and output currents by matching the drain voltages of the mirror transistors. The bias currents are provided by floating gates.

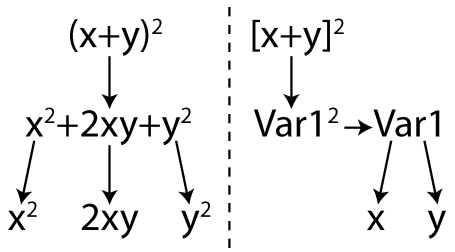

Fig. 9. A representation of how equations are parsed for use in the MFPAA. Equations are split at addition and subtraction signs to create units that will be implemented by MCEs. The user's expression is expanded first in order to create a simple parsing tree (left). However, the user can define sub-blocks by using brackets to replace an expression with an intermediate variable (right).

pad. This is accomplished by using active feedback, shown in Fig. 8, which is similar to the one presented in [21]. The speed of the VI can be written as

$$f_{-3\,\mathrm{dB}} = \frac{1}{2\pi(R \parallel R_{\mathrm{in}})C_{\mathrm{pad}}} \approx \frac{1}{2\pi R_{\mathrm{in}} C_{\mathrm{pad}}} \tag{12}$$

and its accuracy can be written as

$$\mathrm{Error} \approx \frac{R_{\mathrm{in}}}{R} \tag{13}$$

where

$$R_{\mathrm{in}} \approx \frac{1}{g_{m1}[g_{m2} r_{\mathrm{ds2}}(A+1)+1]}. \tag{14}$$

The amplifiers used are simple pFET-input 5 transistor OTAs with a voltage gain of approximately $g_m r_{\mathrm{ds}}$. V_{ref} is usually set to 0.4 V and R is usually 10 MΩ.

IV. The Design Flow

We have developed an entire software chain in order to effectively utilize the MFPAA. The collective purpose of this chain is to implement, in hardware, the equation entered by the user. The main components of the chain are network synthesis, place-and-route, visualization and programming.

A. Network Synthesis

The first step in the software chain is the synthesis of a circuit topology from the input equation. This topic was thoroughly explored in [13]. In order to take advantage of this work, a set of MATLAB functions were written to parse the input

TABLE I
Exponent Patterns Generated With Different Gate Connections

Gate Connections	Input Exponent Pattern
1, 3	+1, -1, +1, 0, 0
1, 5	+1, -1, +1, -1, +1
2, 2	-1, +2, 0, 0, 0
2, 4	-1, +2, -1, +1, 0
3, 3	+1, -2, +2, 0, 0
3, 5	+1, -2, +2, -1, +1
4, 4	-1, +2, -2, +2, 0
5, 5	+1, -2, +2, -2, +2

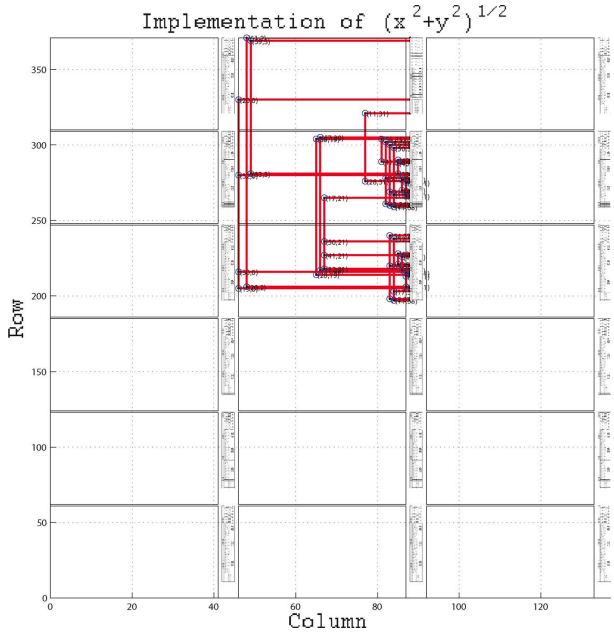

Fig. 10. Sample of the GUI for interfacing with the FPAAA is shown. The GUI output is shown for a vector magnitude circuit.

equation into modules capable of being processed by the MCE. First, the expression is prepared for parsing by expanding it using MATLAB's symbolic toolbox. Since expanding the expression blindly may not lead to optimal use of components in the MFPAA, an option for the user to create sub-blocks was included. This is done by using '[' and ']' instead of parenthesis while entering the equation. Anything included in brackets is treated as its own expression and is replaced by a new variable in the original expression. Once expanded, each expression is split at the '+' and '−' signs in order to break it into units containing only multiplication, division, and powers. These ideas are illustrated in Fig. 9.

Now that expressions containing only multiplication, division, and powers have been obtained, a few special cases must be checked for and taken care of. One of these cases is an expression that contains fractional exponents. Since MITEs with only two gate capacitors can only implement powers with magnitudes of 1 or 2, the final expression that will be implemented can only have integer exponents. This is accomplished by raising the expression to the lowest integer power that will result in all integer exponents. While the new expression is now capable of being implemented, the output now has an exponent other than

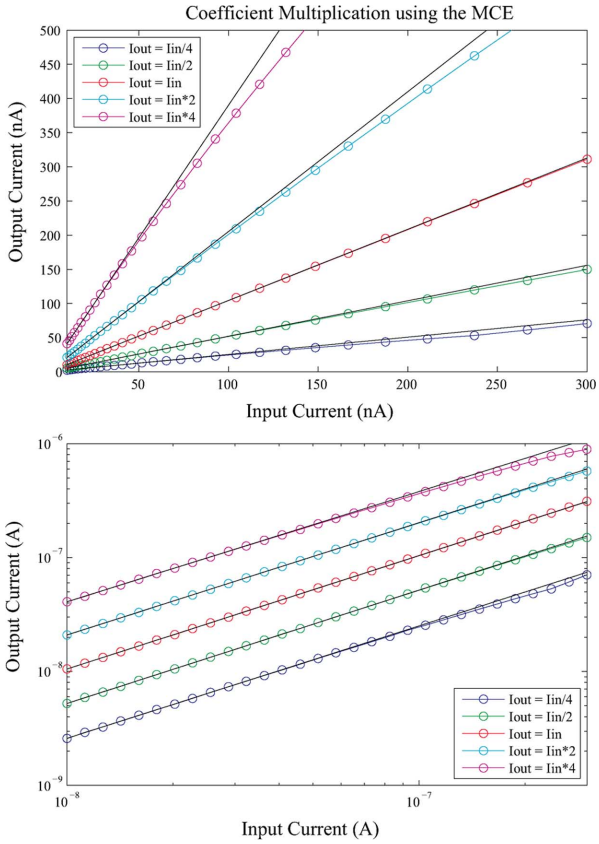

Fig. 11. Results of a coefficient multiplication implemented with the MCE. The results are shown in a linear plot (top) and a log plot (bottom) to show both the accuracy and the dynamic range of the computation.

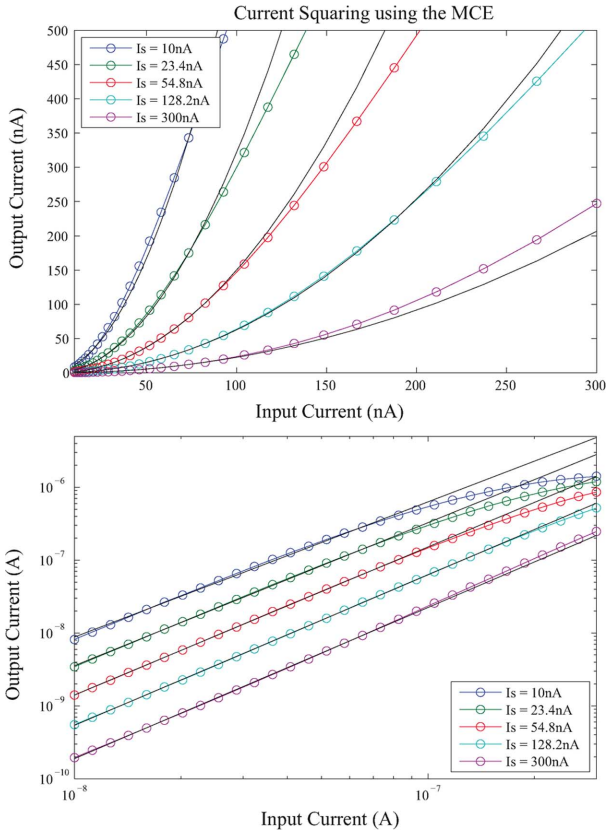

Fig. 12. Results of a squaring circuit implemented with the MCE. The results are shown in a linear plot (top) and a log plot (bottom) to show both the accuracy and the dynamic range of the computation. Note that the inaccuracy at high output currents is due to devices leaving subthreshold operation.

one. To correct this, the output signal will be fed back to produce an equation that results in the intended output. An example of this process is shown here

$$I_{\text{out}} = I_1^{1/2} I_2^{1/4} I_3^{1/4} \Rightarrow I_{\text{out}}^4 = I_1^2 I_2 I_3 \Rightarrow I_{\text{out}} = \frac{I_1^2 I_2 I_3}{I_{\text{out}}^3}. \tag{15}$$

Once functions capable of being implemented with the MITEs are obtained, previous work can be leveraged to map the functions onto a MCE. As described in [13], the fixed gate connections of the 5 input MITEs contained in each MCE produces a set pattern in the exponents of the expression implemented. This pattern can be altered by changing where the gates of the output MITE are connected. The possible patterns are shown in Table I. Exponents with a magnitude greater than two must be realized by connecting the input signal to multiple MITEs. For example

$$I_{\text{out}} = \frac{I_1^3 I_2^2}{I_3^4} = \frac{I_1 I_1^2 I_2^2}{I_3^2 I_3^2}. \tag{16}$$

In addition, expressions that cannot be implemented in a single MITE Computation Element must be broken up into multiple elements. For example

$$I_{\text{out}} = \frac{I_1^4 I_2^3}{I_3^5 I_4} = \left[\frac{I_1^2 I_1^2 I_2}{I_3^2 I_3^2}\right] \frac{I_2^2}{I_3 I_4}. \tag{17}$$

While the MITE elements realize the multiplication, division, and powers found in the user's expression, addition and subtraction is done through the use of KCL. Intermediate expressions are summed by simply connecting the current-mode output of each MITE together, and subtracted by connecting the appropriate output of each MITE to different sides of a current mirror.

B. Place-and-Route

While place-and-route algorithms are an area of active research in both FPGAs [22] and FPAAs [23], the simple algorithm used here is meant to show the possibilities of using a translinear FPAA in simplifying the software algorithms needed. The algorithm, which uses the output of the synthesis function, can be broken into two distinct functions-placement of the components used and routing of the signals between them.

The placement function breaks the input structure into five main categories-inputs, outputs, loops, scaling currents, and mirrors. They are placed in that order by searching for closest available elements to the I/O CAB. The current biases and mirrors are placed in the same CAB as the MCE they are operating on. The routing is then performed by picking the shortest line between elements. The local lines have the lowest cost and the globals have the highest cost, to reduce parasitic capacitance.

The last major functions in the software chain are visualization and hardware programming. While programming floating-gate transistors has been developed previously [24], functions

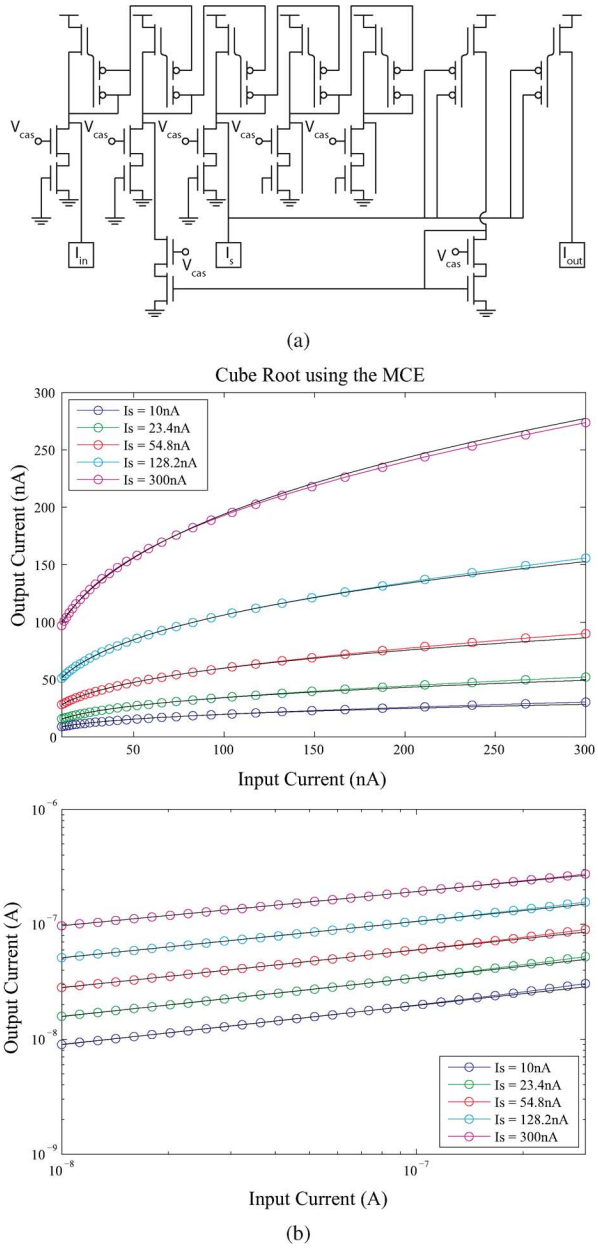

Fig. 13. Cube root circuit. (a) Circuit which implements a cube root on the MFPAA. A second output MITE, from the other MCE in the CAB, is used to gain access to the output current. In addition, a current mirror is used to feed back the output current to create the cube root. (b) Results of a cube root circuit on the MFPAA. The results are shown in a linear plot (top) and a log plot (bottom) to show both the accuracy and the dynamic range of the computation.

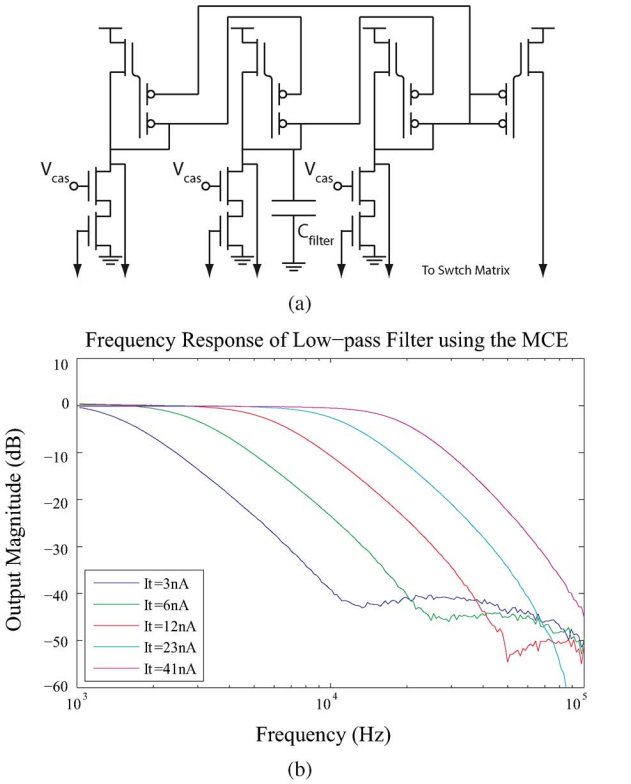

Fig. 14. Log-domain filter of the improved MITE FPAA. (a) The MITE FPAA uses a standard first-order MITE log-domain filter in order to implement dynamic functions. (b) The transfer function of a first-order low-pass filter for various bias currents is shown. The bias currents used were logarithmically spaced between 3 nA and 41 nA. Note that the highest achievable corner frequency is 200 KHz.

have been added to make interfacing with an FPAA much easier. Most importantly, a GUI has been created to show the output of the synthesis and place-and-route functions. This GUI shows the FPAA and draws the switches which will be turned on and the connections between them. It also includes diagrams of the CABs so the user can easily understand what is being connected. A sample of the GUI is in Fig. 10. In addition to allowing the user to easily understand how the equation is being implemented on the FPAA, the GUI also allows the user to modify the implementation if they desire.

Once the equation has been synthesized and routed, the list of switches and programmable elements are programmed on the chip. The setup that allows for this to happen includes a printed circuit board (PCB), a microcontroller, and a computer for communication [25]. Routines for selecting devices, programming switches, and programming computational elements are stored on the microprocessor and initiated by communication for the computer. The computer communicates, over either serial or USB, directly from MATLAB allowing easy interfacing between the synthesis, place-and-route, and programming code.

V. Results

In order to the test the MFPAA, a wide range of circuits were compiled onto it. First, some static functions were tested including circuits for multiplying, squaring, and cube root. Next, dynamic functions were tested. These included a low-pass filter, a high-pass filter, and a RMS-to-DC converter. The circuits were compiled using the synthesis procedures previously discussed.

A. Static Examples

The first static example compiled onto the MFPAA implements the equation

$$I_{\text{out}} = \frac{I_a I_b}{I_c}. \tag{18}$$

In order to test this circuit, I_a was swept while I_b and I_c were held constant. In addition, I_b/I_c, was set to produce a variety of coefficients. The results are shown in Fig. 11.

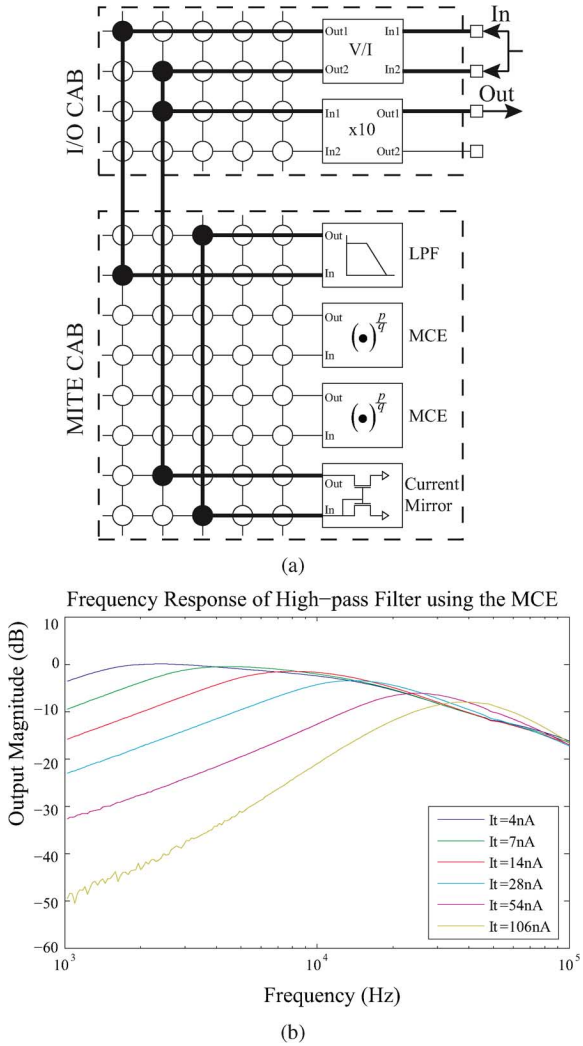

Fig. 15. Log-domain high-pass filter. (a) The log-domain high-pass filter can easily be complied into a single MITE CAB. To implement the high-pass filter a low-pass version of the signal is subtracted from the original signal using the current mirror. (b) The transfer function of the filter for various bias currents is shown. The bias currents used were logarithmically spaced between 4 nA and 106 nA.

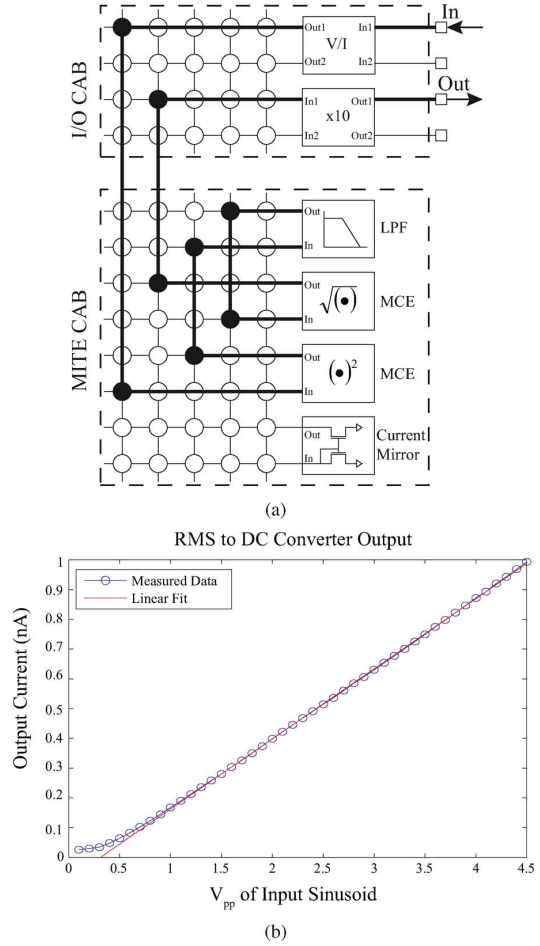

Fig. 16. RMS-to-DC converter. (a) The RMS-to-DC converter as it is complied into a single MITE CAB. The three computational stages are: square, filter, and square-root. These three functions can each be performed by a single MCE. (b) The output characteristic of the RMS-to-DC converter. The amplitude of the input sinusoid was swept from 0.1–4.5 V. The frequency of the input was held at 500 Hz.

Next, a squaring circuit was compiled onto the MFPAA. The circuit uses a scaling current, I_s, that determines the value of unity in the system. This idea is illustrated in the equation

$$I_{\text{out}} = \frac{I_{\text{in}}^2}{I_s} \tag{19}$$

which describes the system's input-output relationship. The results of the squaring circuit are shown in Fig. 12. The most important feature of the output characteristic is its inaccuracy for large input to scaling current ratios. This causes currents larger than the subthreshold range to flow through the output MITE.

A cube root circuit was also compiled on the MFPAA. The circuit is shown in Fig. 13(a). The output MITE of another MITE Computational Element (MCE) is used to gain access to the output current. Again, a scaling current is used set the value of unity in the system. The equation that describes the system is

$$I_{\text{out}} = I_{\text{in}}^{1/3} I_s^{2/3}. \tag{20}$$

The results of the cube root are shown in Fig. 13(b). In contrast to the squaring circuit, the cube root results are more accurate because of its compressive nature.

B. Dynamic Examples

The first dynamic circuit compiled onto the MFPAA was a first-order low-pass filter. The filter is included as one of the CAB components on the MFPAA, shown in Fig. 14(a). The filter was tested by adjusting the bias currents that set the corner frequency of the filter and measuring the transfer function. The results are shown in Fig. 14(b).

Next, a first-order high-pass filter was compiled onto the MFPAA. The filter was built by subtracting a low-pass filtered version of the input from the original signal. The MFPAA implementation of this design is shown in Fig. 15(a). Again, the filter was tested by measuring the transfer function for multiple bias currents. The frequency response of the entire system is more apparent here than in the low-pass filter case. Here, the pass-band shows the effects of the mismatch due to the current mirror. The results are shown in Fig. 15(b).

An RMS-to-DC converter was also compiled onto the MFPAA. A combination of three static and dynamic circuits, in

TABLE II
SYSTEM PARAMETERS

	This work	[11]
Process	$0.35\mu m$ CMOS	$0.35\mu m$ CMOS
Die Size	$9mm^2$	$.92mm^2$
Power Supply	$2.4V$	not given
Number of CABs	18	25
No. of T-L elements	272	25
Largest order filter	17^{th}	4^{th}
Bandwidth	$200KHz$ (measured)	$7MHz$ (simulated)
Current Range	$1nA - 1\mu A$	$1nA - 10\mu A$
Synthesis tools	Complete	none reported

addition to the VI converter, are needed in order to realize the converter. First, the input, which has been rectified by the input VI structure, is squared. Second, it is passed through a low-pass filter to find the mean. Third, the square root of the mean is found. The MFPAA implementation of this design is shown in Fig. 16(a). The converter was tested by varying the input amplitude of a sine wave and measuring the output current. The results are shown in Fig. 16(b).

VI. DISCUSSION

In this paper, we have discussed the design of a reconfigurable MITE system, the MFPAA. This MITE-based FPAA was designed, fabricated in 0.35 μm CMOS, and tested. A summary of this technology and comparison to another translinear FPAA is given in Table II. It was designed using the floating-gate switch matrix framework of the RASP 2.8 line of FPAAs. Floating-gate switches are a natural choice for MITE systems because they can share the programming overhead that is already required to program the MITEs. Along with the MFPAA IC, we also presented an entire chain of design tools: a synthesis tool, a place-and-route tool, a routing visualization GUI, an evaluation board, and the programming system. This complete system allows the user to go from a system of equations all the way to a working hardware MITE implementation. In addition to presenting the hardware and design tools, we demonstrated several working circuits. Static systems such as multipliers and squaring circuits, as well as dynamic systems such as filters and an RMS-to-DC converter were successfully tested on the hardware system.

REFERENCES

[1] S. C. Liu, *Analog VLSI: Circuits and Principles*. Montgomery, VT: Bradford Books, 2002.

[2] U. M. O'Reilly, "Potential uses of dynamically reconfigurable analog circuits," MIT, Tech. Rep. [Online]. Available: http://people.csail.mit.edu/unamay/research-abstracts/grace-abstract/grace-abstract.html

[3] Application note: Developing a state-driven embedded system using the Atmega128," Anadigm, Tech. Rep., Jul. 2009 [Online]. Available: http://www.anadigm.com/doc/AN221009-U209.pdf

[4] A. Stoica, D. Keymeulen, R. Zebulum, A. Thakoor, T. Daud, Y. Klimeck, R. Tawel, and V. Duong, "Evolution of analog circuits on field programmable transistor arrays," in *Proc. NASA/DoD Workshop on Evolvable Hardware*, 2000, pp. 99–108.

[5] J. Becker, F. Henrici, S. Trendelenburg, M. Ortmanns, and Y. Manoli, "A field-programmable analog array of 55 digitally tunable otas in a hexagonal lattice," *IEEE J. Solid-State Circuits*, vol. 43, no. 12, pp. 2759–2768, Dec. 2008.

[6] E. Lee and P. Gulak, "A transconductor-based field-programmable analog array," in *Proc. IEEE Int. Solid-State Circuits Conf.*, 1995, pp. 198–199.

[7] A. Basu, S. Brink, C. Schlottmann, S. Ramakrishnan, C. Petre, S. Koziol, F. Baskaya, C. Twigg, and P. Hasler, "A floating-gate-based field-programmable analog array," *IEEE J. Solid-State Circuits*, vol. 45, no. 9, pp. 1781–1794, Sep. 2010.

[8] C. Schlottmann, C. Petre, and P. Hasler, "A high-level simulink-based tool for FPAA configuration," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2011, accepted for publication.

[9] D. Abramson, J. Gray, S. Subramanian, and P. Hasler, "A field-programmable analog array using translinear elements," in *Proc. Int. Workshop System-on-Chip for Real-Time App.*, 2005, pp. 425–428.

[10] D. Abramson, "A MITE based translinear FPAA and its practical implementation," Ph.D. Thesis, Georgia Tech, Atlanta, Nov. 2008.

[11] L. Martinez-Alvarado, J. Madrenas, and D. Fernandez, "Translinear signal processing circuits in standard CMOS FPAA," in *Proc. IEEE Int. Conf. Electron., Circuits Syst.*, 2009, pp. 715–718.

[12] B. A. Minch, "Synthesis of static and dynamic multiple-input translinear element networks," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 51, no. 2, pp. 409–421, Feb. 2004.

[13] S. Subramanian, "Methods for synthesis of multiple-input translinear element networks," Ph.D. Thesis, Georgia Tech, Atlanta, 2007.

[14] E. McDonald and B. A. Minch, "Synthesis of a translinear analog adaptive filter," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2002, pp. 321–324.

[15] C. Schlottmann, B. Degnan, D. Abramson, and P. Hasler, "Reducing offset errors in mite systems by precise floating gate programming," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 1340–1343.

[16] A. Basu and P. Hasler, "A fully integrated architecture for fast and accurate programming of floating gates over six decades of current," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, 2010, to be published.

[17] J. Mulder, W. A. Serdijn, A. C. van der Woerd, and A. H. M. van Roermund, *Dynamic Translinear and Log-Domain Circuits: Analysis and Synthesis*. Norwell, MA: Kluwer, 1999.

[18] P. Hasler, B. A. Minch, and C. Diorio, "An autozeroing floating-gate amplifier," *IEEE Trans. Circuits Syst. II: Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 74–82, Jan/ 2001.

[19] A. Basu, C. Twigg, S. Brink, P. Hasler, C. Petre, S. Ramakrishnan, S. Koziol, and C. Schlottmann, "Rasp 2.8: A new generation of floating-gate based field programmable analog array," in *Proc. IEEE Custom Integr. Circuits Conf.*, Sept. 2008, pp. 213–216.

[20] B. A. Minch, "A low-voltage mos cascode bias circuit for all current levels," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2002, pp. 619–622.

[21] V. Srinivasan, R. Chawla, and P. Hasler, "Linear current-to-voltage and voltage-to-current converters," in *IEEE Midwest Symp. Circuits Syst.*, 2005, pp. 675–678.

[22] S. K. Nag and R. A. Rutenbar, "Performance-driven simultaneous placement and routing for fpga's," *IEEE Trans. Computer-Aided Design Integr. Circuits Syst.*, vol. 17, no. 6, pp. 499–518, 1998.

[23] F. Baskaya, S. Reddy, S. K. Lim, and D. V. Anderson, "Placement for large-scale floating-gate field-programable analog arrays," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 14, no. 8, pp. 906–910, Aug. 2006.

[24] G. Serrano, P. Smith, H. J. Lo, R. Chawla, T. Hall, C. Twigg, and P. Hasler, "Automatic rapid programming of large arrays of floating-gate elements," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2004, pp. 373–376.

[25] S. Koziol, C. Schlottmann, A. Basu, S. Brink, C. Petre, B. Degnan, S. Ramakrishnan, P. Hasler, and A. Balavoine, "Hardware and software infrastructure for a family of floating-gate based FPAAs," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2010, pp. 2794–2797.

Craig R. Schlottmann, photograph and biography not available at the time of publication.

David Abramson, photograph and biography not available at the time of publication.

Paul E. Hasler, photograph and biography not available at the time of publication.