11-1-2005

Large–scale field–programmable analog arrays for analog signal processing

Tyson S. Hall
Southern Adventist University, tyson@southern.edu

Christopher M. Twigg
Georgia Institute of Technology - Main Campus, ctwigg@ece.gatech.edu

Jordan D. Gray

Paul Hasler
Georgia Institute of Technology - Main Campus, phasler@ece.gatech.edu

David V. Anderson
Georgia Institute of Technology - Main Campus, dva@ece.gatech.edu

Large-Scale Field-Programmable Analog Arrays for Analog Signal Processing

Tyson S. Hall, *Member, IEEE*, Christopher M. Twigg, *Student Member, IEEE*, Jordan D. Gray, *Student Member, IEEE*, Paul Hasler, *Senior Member, IEEE*, and David V. Anderson, *Senior Member, IEEE*

***Abstract*—Field-programmable analog arrays (FPAAs) provide a method for rapidly prototyping analog systems. Currently available commercial and academic FPAAs are typically based on operational amplifiers (or other similar analog primitives) with only a few computational elements per chip. While their specific architectures vary, their small sizes and often restrictive interconnect designs leave current FPAAs limited in functionality and flexibility. For FPAAs to enter the realm of large-scale reconfigurable devices such as modern field-programmable gate arrays (FPGAs), new technologies must be explored to provide area-efficient accurately programmable analog circuitry that can be easily integrated into a larger digital/mixed-signal system. Recent advances in the area of floating-gate transistors have led to a core technology that exhibits many of these qualities, and current research promises a digitally controllable analog technology that can be directly mated to commercial FPGAs. By leveraging these advances, a new generation of FPAAs is introduced in this paper that will dramatically advance the current state of the art in terms of size, functionality, and flexibility. FPAAs have been fabricated using floating-gate transistors as the sole programmable element, and the results of characterization and system-level experiments on the most recent FPAA are shown.**

***Index Terms*—Analog arrays, field-programmable analog arrays (FPAAs), floating gate, reconfigurable.**

I. Low-Power Signal Processing

THE GROWING demand for complex information processing on portable devices has motivated significant research in the design of power efficient signal processing systems. One method for achieving low-power designs is to move processing on system inputs from the digital processor to analog hardware situated before the analog-to-digital converter (ADC). However, for analog systems to be desirable to digital signal processing engineers, they need to provide a significant advantage in terms of size and power and yet still remain relatively easy to use and integrate into a larger digital system. Reconfigurable analog arrays, dubbed field-programmable analog arrays (FPAAs), can speed the transition of systems from digital to analog by providing the ability to rapidly implement advanced, low-power signal processing systems.

Manuscript received August 8, 2004; revised April 3, 2005. This work was supported in part by the National Science Foundation under Grant 0347792, and Grant 0083172.

T. S. Hall is with the School of Computing, Southern Adventist University, Collegedale, TN 37315-0370 USA (e-mail: tyson@southern.edu).

C. M. Twigg, J. D. Gray, P. Hasler, and D. V. Anderson are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: ctwigg@ece.gatech.edu; jgray@ece.gatech.edu; phasler@ece.gatech.edu; dva@ece.gatech.edu).

Digital Object Identifier 10.1109/TCSI.2005.853401

Gene's law postulates that the power consumption in digital signal processing microprocessors, as measured in milliwatts per million multiply-accumulate (mW/MMAC) operations, is halved about every 18 months. These advances largely follow Moore's law, and they are achieved by using decreased feature size and other refinements, such as intelligent clock gating. Myriad applications only dreamed of a few years ago are possible because of these gains, and they have increased the demand for more advanced signal processing systems. Unfortunately, a problem looms on the horizon: the power consumption of the ADC does not follow Gene's law and will soon dominate the total power budget of digital systems. While ADC resolution has been increasing at roughly 1.5 bits every five years, the power performance has remained the same, and soon, physical limits will further slow progress.

Most current signal processing systems that generate digital output place the ADC as close to the analog input signal as possible to take advantage of the computational flexibility available in digital processors. However, the development of large-scale FPAAs and the computer-aided design (CAD) tools needed for their ease of use would allow engineers the option of performing some of the computations in reconfigurable analog hardware prior to the ADC. This results in both a simpler ADC and a substantially reduced computational load on the digital processors that follow. Furthermore, the analog processor and ADC may be combined to form a specialized ADC tailored to the application at hand.

FPAAs have been of interest for some time, but historically, these devices have had very few programmable elements and limited interconnect capabilities, making them limited in their usefulness and versatility. Currently available commercial and academic FPAAs are typically based on op-amp circuits with only relatively few op-amps per chip [1]–[9]—see [10] for a more exhaustive discussion of previous FPAA designs. By building larger, more flexible FPAAs, reconfigurable analog devices will become more analogous to today's high-density FPGA architectures. This will enable a very useful rapid prototyping system for analog circuit development.

Recent advances in analog floating-gate technologies have shown it to be a viable alternative to traditional FPAA designs [15]. As shown in Fig. 1, analog floating-gate circuits have shown tremendous gains in efficiency (a factor of as much as 10 000) compared with custom digital approaches for the same applications. A new FPAA design, dubbed the Reconfigurable Analog Signal Processor (RASP) 1.5, has been fabricated that is based on floating-gate technology (Fig. 2).

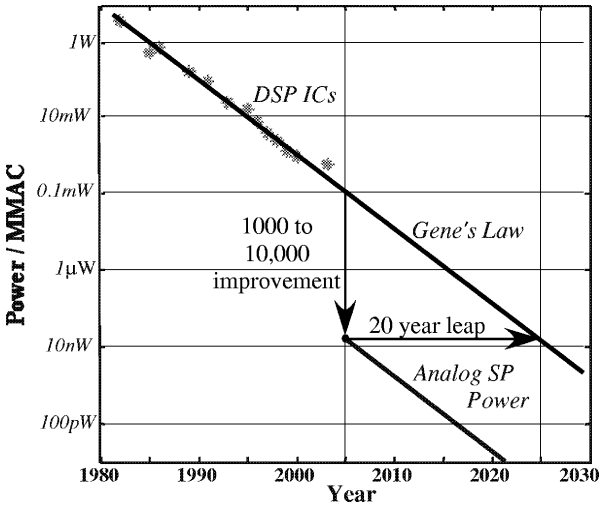

Fig. 1. Data from [11] showing the power consumption trends in DSP microprocessors along with data taken from a recent analog, floating-gate integrated chip developed by the CADSP team [12]–[14].

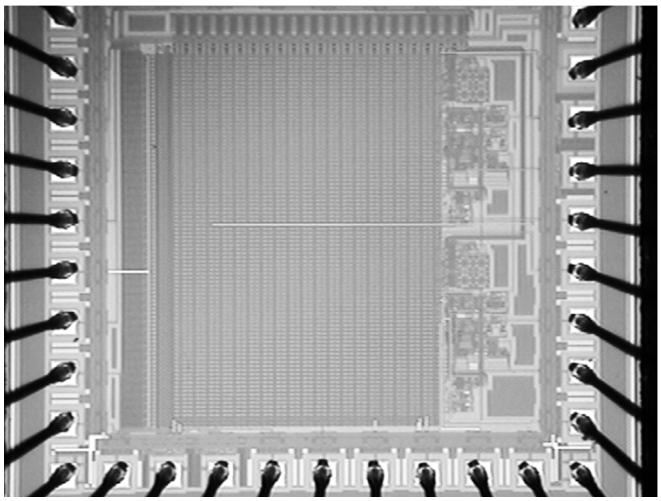

Fig. 2. Die photo for the RASP 1.5 FPAA. It is used to characterize the large-scale FPAA architecture explored in this paper.

This paper proceeds with a discussion of the challenges of building reconfigurable analog devices in Section II. The architecture for our large-scale FPAAs is described in Section III. In Section IV, the fabricated FPAA is discussed, and experimental data is shown that characterizes some of the low-level components of this device. In Section V, the FPAA system performance is analyzed by looking at results from experimental systems that have been prototyped on the FPAA.

II. Rapid Prototyping of Analog Systems

The traditional analog IC design process can be lengthy, lasting for over a year if multiple iterations of a design must be fabricated. Thus, the benefits of rapid prototyping for analog circuits would be significant in the design and testing of analog systems. FPAAs provide a viable platform for rapid prototyping of analog systems, and in design and function, they are the analog equivalent to digital reconfigurable devices such as programmable logic devices (PLDs) and field-programmable gate arrays (FPGAs).

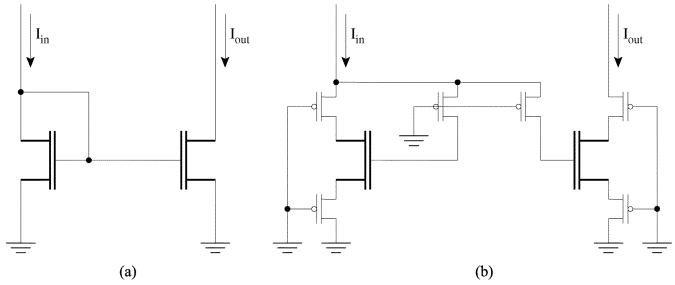

Fig. 3. (a) Simple two-transistor current mirror illustrates the challenges of designing reconfigurable ICs with fine-grain building blocks. (b) Parasitic capacitance that results from using switches to form the current mirror will reduce the bandwidth of the system.

Traditional FPAAs resemble the early PLDs in that they are focused on small systems such as low-order filtering, amplification, and signal conditioning. However, the class of large-scale FPAAs explored in this paper are more analogous to modern FPGAs. These FPAAs are much larger devices with the functionality needed to implement high-level system blocks such as programmable high-order filtering and fourier processing in addition to having a large number of basic analog blocks [e.g., operational transconductance amplifiers (OTAs), transistor elements, capacitors, etc.].

Rapid prototyping of analog systems is not completely analogous to its digital counterpart. Developing robust, programmable analog circuits presents a number of challenges not found in the digital world. In particular, the noise sensitivity (and effects of the switch network on the results of the computation) and the design space to which programmable devices are applicable are more critical factors in designing FPAAs.

A. Noise Sensitivity

Analog circuits tend to be more sensitive to noise than digital designs. Because of the quantization and resulting representation of ones and zeros as discrete voltages, digital designs can tolerate a relatively large amount of noise in the system without changing the precision of the result. Problems arise only when noise levels are high enough to move a signal from a logical one to a logical zero or vice versa. In the analog domain, however, values are represented as continuous voltages or currents. Any noise in the system will directly affect the precision of the result. For reconfigurable analog systems that rely on networks of switches to set the internal signal paths, this means that the parasitics of the switches in a signal's path can affect the result and are a critical factor in the performance of the FPAA.

Adding switches in the signal path can have several effects including the addition of parasitic capacitance, resistance, and transistor leakage currents to the path. Increased capacitance and resistance on a signal line will lower the bandwidth of the system. For example, a simple two-transistor current mirror is shown in Fig. 3(a). The same circuit is shown in Fig. 3(b) with switches added to the signal paths as they would need to be if the current mirror were synthesized on the FPAA using the MOSFET transistors in the computational analog block (CAB). In this case, there should not be any current flow between the gate nodes, so the voltage should remain equal on the two gate nodes even with the switches in the signal path. Other circuits

that have switches in a signal path with current flowing through them will have a voltage drop across the transistor that can vary nonlinearly. As the number of switches in a given design increases, the performance and functionality will degrade significantly.

B. Design Space

Another difference between reconfigurable analog and digital devices is the design space that each must encompass. Functionality in the digital domain can be reduced to a basic set of combinational and sequential primitives. For example, a NAND gate can be configured to implement any of the other Boolean logic gates. Thus, with a sufficiently large number of NAND gates, any combinational logic function can be achieved. Similiarly, an asynchronous read-only memory (ROM) primitive can be used to implement any combinational function. For sequential functions, any basic storage element (e.g., flip-flop or latch) can be used to provide the necessary memory. Most modern FPGAs use asynchronous ROMs to synthesize the combinational logic and D-type flip-flops for implementing the memory/sequential logic. Thus, by replicating these two basic primitives thousands of times across a chip (and a sufficient routing network), an FPGA can be created that synthesizes a very large number of different digital systems. It is tempting to think that one might be able to do the same thing in the analog domain. However, there has not been a sufficiently generic set of medium-grained building blocks (on the same order of complexity as flip-flops or asynchronous ROMs) proposed for synthesizing a wide-range of analog circuits. To get the desired generality, one must use fine-grain building blocks, such as transistors, resistors, diodes, and capacitors. Indeed, a large number of analog systems can be built with these basic blocks; even digital systems could be synthesized with such a device. However, these primitives are so fine-grained that it would require such a large number of components—and thus a large number of switches—to implement a design that the switch parasitics would significantly degrade the performance. For example, the circuit diagram for a basic 9-transistor OTA is shown in Fig. 4(a). In Fig. 4(b), the same OTA is shown with the switches necessary to synthesize this circuit on a fine-grained FPAA with transistors only. The FPAA design requires at least 27 switches, in addition to the nine transistors, to implement the OTA. The switches will drastically affect the performance and functionality of the OTA and may cause the circuit to break. To mitigate these effects, coarser-grained blocks must be used. The task then is to do so while still maintaining sufficient flexibility, functionality, and generality.

Using coarse-grain blocks can be appealing given their increase in performance and robustness over fine-grain blocks. However, if the basic building blocks in an FPAA are of too high a level, then the flexibility is greatly diminished. To be as flexible as possible, an FPAA needs to have a wide range of fine-grained, medium-grained, and coarse-grained components. This means that there will often be more than one way of synthesizing the same system on the FPAA. This provides the most flexibility to end-users, because they can vary the levels of performance, utilization, flexibility, and complexity.

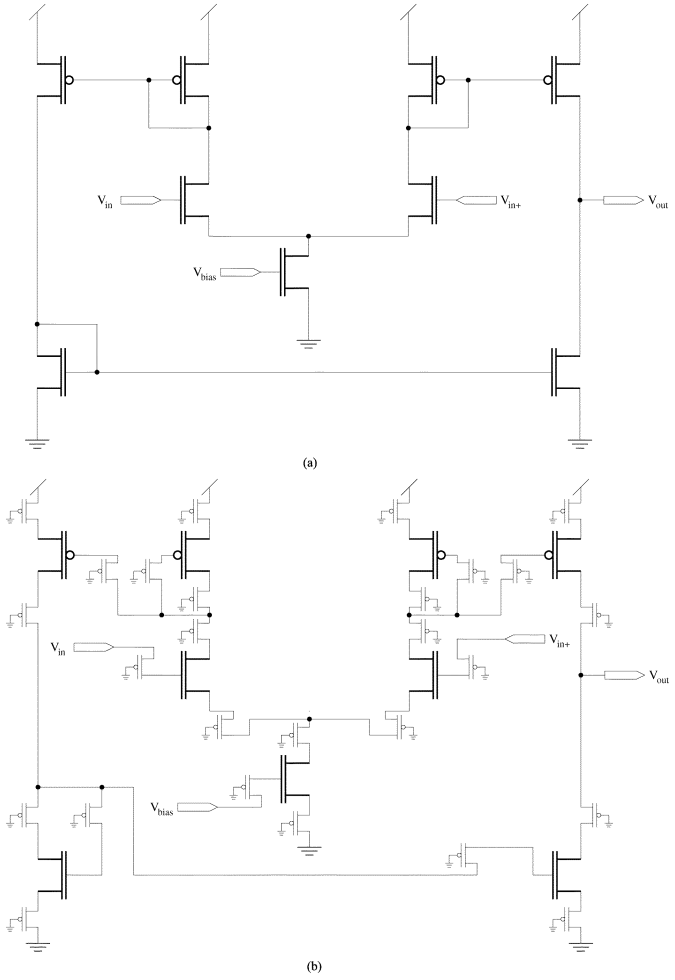

Fig. 4. (a) Circuit diagram of a basic 9-transistor OTA. (b) Circuit diagram of the same OTA with the switches needed as implemented on a fine-grain FPAA with only transistors. The addition of the 27 switches will dramatically reduce the performance and functionality of this circuit.

III. Large-Scale FPAAs

Previous FPAAs [2], [5], [6], [16] are somewhat limited due to their small size and lack of generality. By addressing these problems, we hope to extend the usefulness and acceptance of FPAAs. In addition, large-scale FPAA designs must address the complex design space that analog designs entail (including a wide-range of linear and nonlinear functions) while keeping switch parasitics minimized as discussed in the previous section.

The switches used in FPAAs are very important, because the signals are affected by any nonideal characteristics. An ideal switch switch has zero impedance when ON and infinite impedance when OFF. For practical reasons, the switch should also be small and easily controlled. The switches used in our FPAA are based on floating-gate transistors and can be used to approximate the ideal switch. Additionally, the floating-gate switches can be programmed to states between ON and OFF, synthesizing a finite resistance. Thus, switches can be used as a resistive circuit element within the design [15].

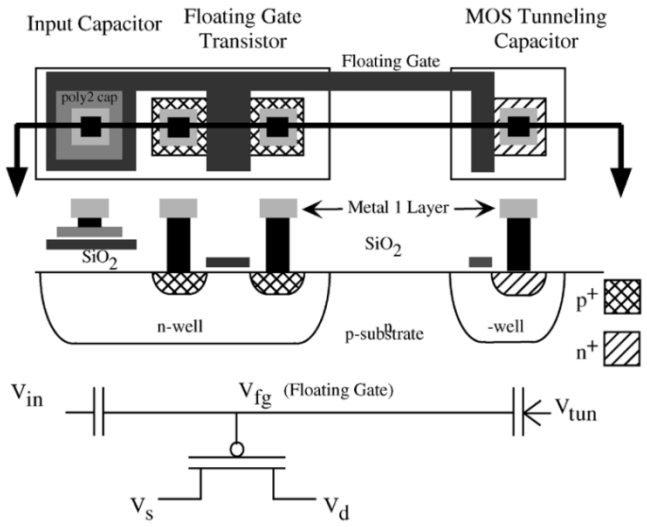

Fig. 5. Layout, cross section, and circuit diagram of the floating-gate pFET in a standard double-poly, n-well MOSIS process: The cross section corresponds to the horizontal line slicing through the layout view. The pFET transistor is the standard pFET transistor in the n-well process. The gate input capacitively couples to the floating-gate by either a poly–poly capacitor, a diffused linear capacitor, or a MOS capacitor, as seen in the circuit diagram (not explicitly shown in the other two figures). Between $V_{\rm tun}$ and the floating-gate is our symbol for a tunneling junction-a capacitor with an added arrow designating the charge flow.

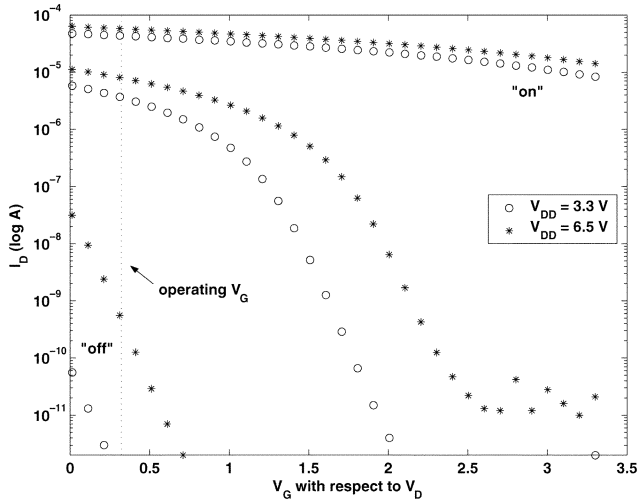

Fig. 6. Floating-gate switches can be programmed within a wide range. Here, examples of an ON, OFF, and midposition device are shown. During programming, currents are measured with $V_{\rm DD} = 3.3$ V for large currents and $V_{\rm DD} = 6.5$ V for small currents. This effectively extends the programming range of the device.

A. Building Compact Floating-Gate Switches

The floating-gate transistors used in these FPAAs are standard pFET devices whose gate terminals are not connected to signals except through capacitors (e.g., no dc path to a fixed potential) [17]. Fig. 5 shows the layout, cross-section, and circuit symbol for the floating-gate pFET device. Because the gate terminal is well insulated from external signals, it can maintain a permanent charge, and thus, it is an analog memory cell similar to an EEPROM cell. With a floating gate, the current through the pFET channel is dependent on the charge of the floating-gate node. By using hot-electron injection to decrease the charge on the floating-gate node and electron tunneling to increase the charge on the floating-gate node, the current flow through the pFET channel can be accurately controlled [17], [18].

To increase the quality of a switch, the floating-gate transistors are programmed to the far extremes of their range. When switches are being programmed OFF, currents in the low picoampere range must be measured. These measurements are near the limits of standard laboratory equipment; therefore, to extend the viable programming range, current measurements are taken at a larger drain-to-source voltage. Typically, V_{DS} is set to the supply voltage, $V_{\rm DD}$, and an increase in V_{DS} is achieved by increasing $V_{\rm DD}$. As shown in Fig. 6, measuring the currents with $V_{\rm DD} = 6.5$ V, allows the I–V curves to be visible to the programming infrastructure 1 V below the point visible when $V_{\rm DD} = 3.3$ V [19].

For simplicity, the voltages on the gate capacitors of all the switches are set to a constant potential. This means that the voltage driving the gate capacitors will be the same for both ON and OFF switches. To determine the appropriate gate voltage for run mode, the relative quality of ON and OFF switches must be balanced. From Fig. 6, it is clear that the OFF switches do not pose a problem, since any gate voltage selected at or above 0.3 V should provide a sufficiently high impedance. However, the ON switch exhibits a decrease in quality as the gate voltage is increased to $V_{\rm DD}$. Thus, an operating gate voltage of 0.3 V is deemed optimal for the current programming scheme.

B. Switch as a Computational Element

When used as a switch, the floating gate should be as transparent a part of the circuit as possible. However, Fig. 6 shows that the floating-gate transistor can also be used as an in-circuit element [20]. By adjusting the charge on the floating-gate node between the extremes used for ON and OFF, the impedance of the switch can be varied over several orders of magnitude. Thus, a variable nonlinear resistor can be synthesized by the floating-gate switch.

Using the floating-gate switches as in-circuit elements allows for a very compact architecture. The physical area needed for the CABs is reduced greatly, because resistors, which consume relatively large amounts of space on CMOS processes, are not needed as separate components. Also, by reducing the number of individual circuit elements, signal routing is simplified, while retaining functionality.

C. Floating-Gate Transistors Within Computational Logic

Current FPAA designs rely on switches as the primary or sole programmable element on the chip. Biases, multiplier coefficients, resistances, and similar elements are set via off-chip components or with low-resolution capacitor banks or current-mirror banks. Thus, the ability to modify or program the actual analog computational logic is severely limited. By using floating-gate transistors within the computational logic, circuit characteristics can be directly modified.

In the FPAA explored here, floating-gate transistors are used within the CABs to set bias voltages for the OTAs [see Fig. 9(a), shown later], adjust the corner frequencies on the capacitively coupled current conveyors (C^4s), and set muliplier coefficients

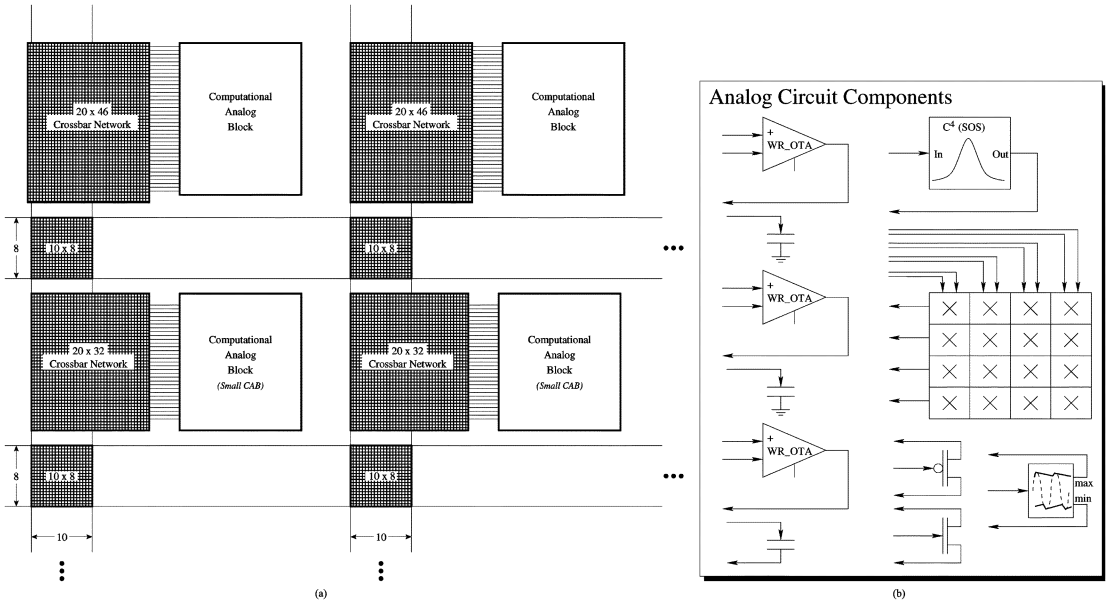

Fig. 7. (a) Overall block diagram for a large-scale FPAA. The switching interconnects are fully connectable crossbar networks built using floating-gate transistors. (b) There are two slightly different CABs on this FPAA. The large or regular CAB contains a four-by-four matrix multiplier, three wide-range OTAs, three fixed-value capacitors, a capacitively coupled current conveyor (C^4) SOS, a peak detector, and two FET transistors. The small CAB is the same except it does not include the four-by-four matrix multiplier. This design includes large CABs at the top and bottom of each column and small CABs in-between.

in the vector-matrix multipliers. In this manner, the floating-gate transistors allow the characteristics of the computational elements to be programmed on chip while still maintaining a compact CAB. Thus, by allowing both the switch networks and the computational logic to be programmable, the flexibility and usability of these FPAAs are greatly enhanced over previous designs.

D. Programmability

By using floating-gate devices as the only programmable element on the chip, configuring the chip is greatly simplified. Additionally, all of the floating-gate transistors are clustered together to aid in the programming logic and signal routing. Decoders on the periphery of the circuit are connected to the drain, source, and gate (through a capacitor) terminals of the floating-gate matrix. During programming mode, these decoders allow each floating-gate transistor to be individually programmed using hot-electron injection [18].

Part of the previous work has been the development of a systematic method for programming arrays of floating-gate transistors [18], [20], [21]. A microprocessor-based board has been built to interface a PC to these analog floating-gate arrays for the purposes of programming and testing. With a PC controlling the programming of these devices, the details of using hot-electron injection and tunneling to program individual floating-gate switches have been abstracted away from the end-user. The programming algorithms have been optimized for accuracy and speed, while giving the end-user an easy-to-use interface for configuring arrays of floating-gate devices.

E. Switch Networks

As shown in Fig. 7(a), the routing architecture of our large-scale FPAAs is a combination of global and local switch networks. Each CAB has an associated local switch network for making connections within a single CAB. The switches' source lines are routed along the rows and connect the inputs and outputs of each CAB to the switch network. The drain lines of the switches are connected along the columns. By turning a switch on, a single row (source) can be connected through the switch to a single column (drain).

The size of the switch network is dependent on the number of I/O lines in each CAB. For the design shown in Fig. 7, there are two types of CABs. For the larger CABs, the local switch networks are comprised of a 10×42 matrix of switches, and for the smaller CABs, the local switch network is a 10×32 matrix of switches. Each local switch network is integrated into a matching global routing switch network. The global routing switch network allows local signals from a CAB to be connected to the global routing busses and be routed off the chip or to another CAB. There are also 10×8 switch networks at each junction of the horizontal and vertical global routing busses.

The performance of circuit designs implemented on the FPAA are dependent on the routing patterns used and the number of switches needed in a given path. Frequency analysis of floating-gate circuits and the effects of switch networks has been discussed in [22] and placement and routing algorithms for large-scale FPAAs have been discussed in [32].

F. Computational Analog Blocks

The computational logic is organized in a compact CAB providing a naturally scalable architecture. CABs are tiled across the chip in a regular mesh-type architecture with busses and local interconnects in-between as shown in Fig. 7(a).

Many example CABs can be imagined using this technology. Fig. 7(b) shows one example CAB, whose functionality is enhanced by a mixture of fine-grained, medium-grained, and coarse-grained computational blocks similar to many modern

FPGA designs. The computational blocks were carefully selected to provide a sufficiently flexible, generic architecture while optimizing certain frequently used signal processing blocks. For generality, three OTAs are included in each CAB. OTAs have already been shown to be effective at implementing a large class of systems including amplification, integration, filtering, multiplication, exponentiation, modulation, and other linear and nonlinear functions [23]–[26]. In addition, the two FET devices provide the ability to perform logarithmic and exponential functions as well as convert back and forth between current and voltage. The three capacitors are fixed in value to minimize the size of the CAB and are primarily used on the outputs of the OTAs; however, they will be available for any purpose. The variable capacitor and/or current mirror banks found in some designs are not needed here, because the use of floating-gate transistors in the OTAs will give the user sufficient control in programming the transconductance of the amplifiers [24], [27]. Eliminating the capacitor banks creates a large savings in the area required for each CAB.

The high-level computational blocks used in this design are a second-order section (SOS) bandpass filter module comprised of two capacitively coupled current conveyors (C^4) and the 4×4 vector-matrix multiplier block. In general, the C^4 SOS module provides a straightforward method of subbanding an incoming signal. This allows Fourier analysis analagous to performing a fast Fourier transform (FFT) in the digital domain. The vector-matrix multiplier block allows the user to perform a matrix transformation on the incoming signals. Together these blocks can be used like a Fourier processor [18], [28]. In addition, a peak detector is added to each CAB. The peak detector allows the amplitude to be extracted from the incoming waveform and is useful for doing static or dynamic gain adjustments on individual subbands of the incoming signal.

The architecture illustrated in Fig. 7(a) is nonhomogeneous in that there are two different CABs tiled across the chip. The small CAB is identical to the large CAB except it does not include the vector-matrix multiplier module. Since the vector-matrix multiplier takes four inputs and each input will often be derived in a separate CAB (from a separate subband created by the C^4 SOS module), designs will typically only utilize one vector-matrix multiplier for every four CABs. Thus, FPAAs that have 50–100 CABs can be made more compact by removing the vector-matrix multiplier from all the CABs except those on the top and bottom rows (assuming the FPAA is more or less square in design). Alternatively, the vector-matrix multipliers in some of the CABs could be replaced with other specialized circuits to increase functionality and performance for a targeted application.

IV. RASP 1.5 FPAA

Several FPAAs have been fabricated in 0.5-μm, standard CMOS process to characterize the switches, computational logic, and programming infrastructure. The first chip fabricated was the RASP, and it is discussed in [15], [19], [27]. A second version of RASP, dubbed RASP 1.5, has been fabricated with a number of small circuit and architectural improvements over the previous chip. Experimental results that characterize the switch network and demonstrate system-level functionality

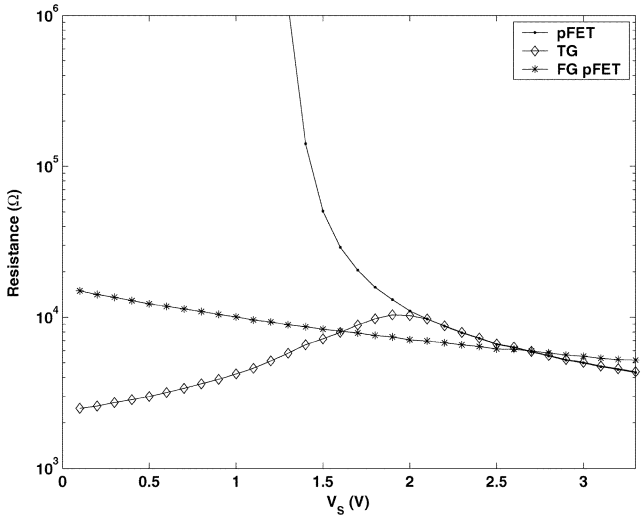

Fig. 8. Switch resistances for a floating-gate pFET, standard pFET, and a standard two-transistor T-gate . The floating-gate switch has been programmed to an extremely *on* position such that the high-impedance region (at 1.5 V for the standard pFET shown) has effectively been shifted below the power rail. This results in a relatively flat resistance similar to the larger T-gate.

are discussed in the next two sections. RASP 1.5 contains two CABs with a floating-gate crossbar switch network connecting them. Both CABs are identical to the large CAB illustrated in Fig. 7(b).

As discussed earlier, the resistance and capacitance of the floating-gate switch are important characteristics. The ON switch resistance is plotted in Fig. 8. For reference, this figure also shows the resistance of a standard pFET (with an SRAM memory bit setting the gate) and transmission gate (T-gate) (both an nFET and a pFET passing the signal). When programmed to a point that is not extremely *on*, the floating-gate switch exhibits a resistance that is very similar to the standard pFET shown here (as seen in [27]. However, by injecting the floating-gate switch further, the voltage on the isolated gate node is pushed lower and thus the resistance curve shifts to the left. This figure shows that by programming the switch far enough, the resistance through the switch can maintain a more consistent level through the operating range (power rails) of the switch. This allows a single floating-gate pFET to exhibit a resistive characteristic that is similar to the resistance of a standard T-gate with two transistors. As shown, the resistance of the floating-gate switches is approximately 10 kΩ, which is about what is expected for relatively small ($W/L = 3$) pFETs.

The OFF resistance is harder to measure given the limitations of standard test equipment. Even at a $V_{DS} = 3.3$ V, current through the OFF switches is below the measureable range of standard picoammeters. Given this, the OFF resistance should be in the gigaohm range and in the worst case hundreds of megaohms. Likewise, the parasitic capacitance of the switches is difficult to measure when they are embedded in the switch network and accessible only through the programming infrastructure. A theoritical estimate based on the layout and fabrication parameters yields a value of 1 fF for each switch on each column and row. Thus, for the RASP 1.5, each column is estimated to contribute 96 fF of parasitic capacitance and 46 fF for each row.

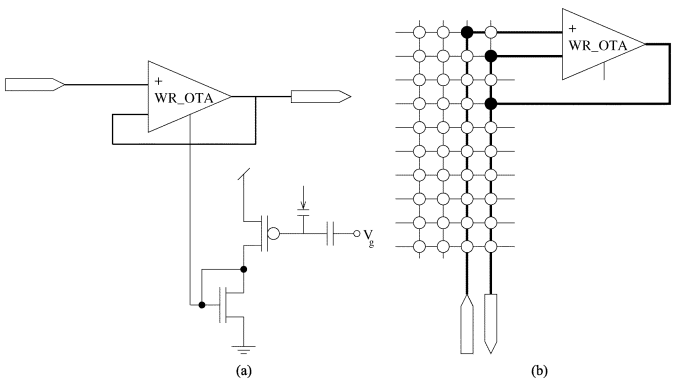

Fig. 9. (a) Source-follower configured using a floating-gate current source. By programming the floating gate charge, the current is set in the current mirror (the other half of the current mirror is internal to the wide-range OTA). Thus, the effective conductance can be modified for each of the OTAs on chip. (b) Using the switch matrix, an OTA located in one of the CABs is connected in a source-follower configuration, and two external pins are routed to the OTA as the input and output signals. The programmable biases illustrated in (a) are not shown here for simplicity, but each OTA has a current mirror and floating-gate current source that sets its bias.

V. System Results

A number of different analog systems have been synthesized on the RASP 1.5. These systems vary from simple one and two element systems to more complex systems with as many as seven on-chip components. These systems also use a range of different CAB components including fine-grained (transistors and capacitors), medium-grained (OTAs), and coarse-grained (C^4 SOS and peak detector). In each of these examples, floating-gate transistors are used as current sources to set biases. Depending on the circuit, these programmable biases are shown to control filter corner frequencies, Q-peaks, and time constants.

A. Low-Order Filtering With OTAs

A first-order filter can be implemented in the RASP 1.5 FPAA using an OTA in one of its CABs. Fig. 9 shows how the circuit is mapped onto the FPAA using five floating-gate switches. Once the switch network is configured, the biasing floating-gate transistor is programmed to vary the corner frequency of this first-order filter. The frequency response is shown for several programmed corner frequencies in Fig. 10(a). The plot in Fig. 10(b) shows the correlation between programmed bias current and measured corner frequency. By fitting a curve to this data, the ability to predict the necessary bias current for a desired corner frequency can be achieved. This is important, because the user will typically want to specify the system parameters in terms of corner frequency, Q-peak, time constants, offsets, etc. and then let the programming interface make the translation to the appropriate bias currents to generate these paramete these currents while programming the floating-gate transistors. Experimental results from Fig. 6 show a measurement threshold of 1 pA using present measurement techniques. An important consideration here is the relative sizing of the transistors that set the bias rs.

In Fig. 11, a SOS filter is shown along side the FPAA implementation. The fixed-value capacitors and OTAs from a single CAB are used to synthesize this circuit. Using the floating-gate

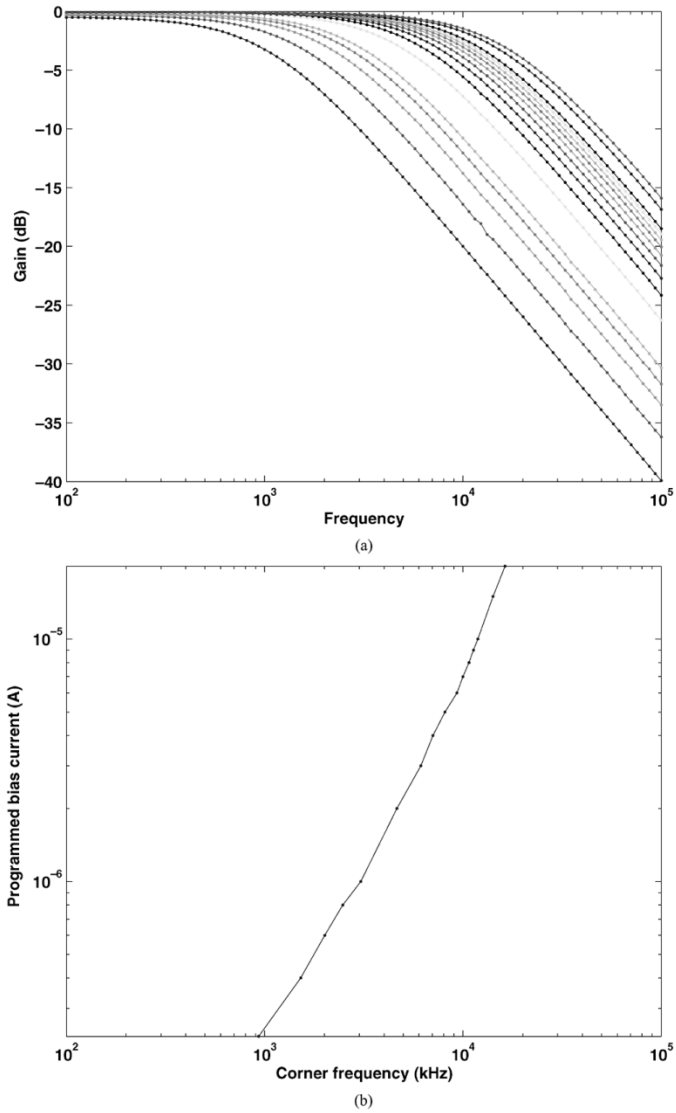

Fig. 10. (a) Frequency response of the source-follower circuit for several bias currents. An internal floating-gate transistor is used as a current source to set the OTAs bias. Injecting the floating-gate device increases the current and thus the bandwidth of this first order filter. (b) Plot shows the correlation between bias current and programmed corner frequency. This data can be used in future experiments to select the appropriate bias current for the desired corner frequency.

programmble biases, the two OTAs in a source–follower configuration were biased to the same level and the the third OTAs bias current was increased to adjust the Q-peak of the system. The frequency response for this circuit is shown in Fig. 11(c). As expected, the Q-peak increases as the third bias current (e.g., conductance) increases.

For second-order functions such as the SOS and diff2 circuit, reasonable Q-peaks and filter bandwidths require small bias currents (in the picoampere to femtoampere range). While the floating-gate transistors can set bias currents this low, the constraint becomes the ability to accurately measure these currents while programming the floating-gate transistors. Experimental results from Fig. 6 show a measurement threshold of 1 pA using present measurement techniques. An important consideration here is the relative sizing of the transistors that set the bias currents. The floating-gate transistor shown in Fig. 9(a) sets

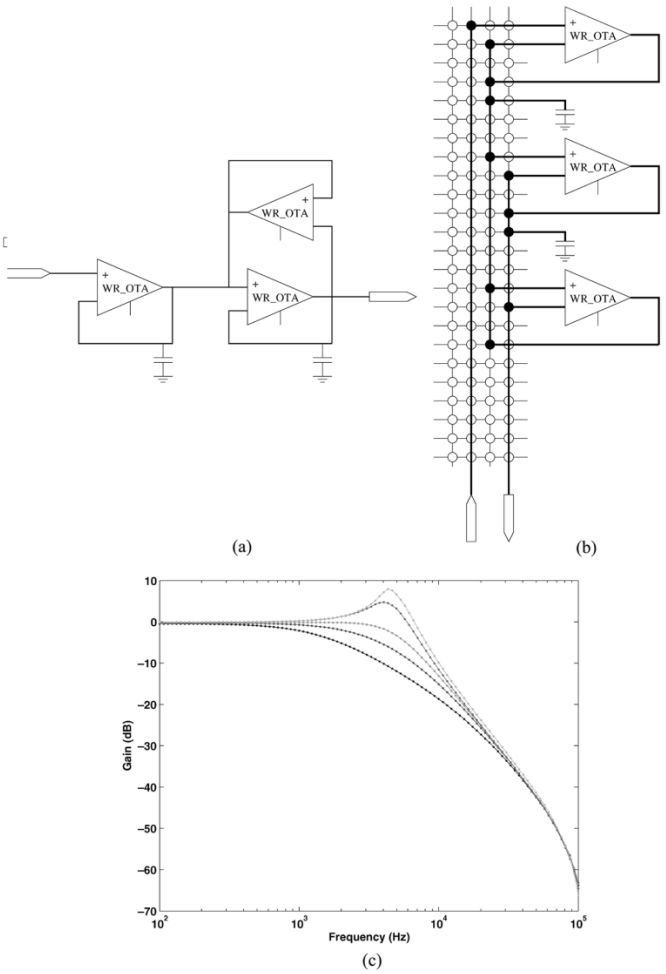

Fig. 11. (a) SOS filter can be implemented with two OTAs in a source-follower configuration and a third OTA that creates postive feedback. (b) Using the switch matrix, two OTAs within the CABs are connected in a SOS configuration. The programmable biases shown in Fig. 9(a) are not included here for simplicity, but each OTA has a current mirror and floating-gate current source that sets its bias. (c) The experimental frequency response of a Access is not allowed to one of your selected collections: Rates SOS filter is shown here. The Q parameter is adjusted by increasing the bias current of the positive feedback amplifier via a floating-gate current source.

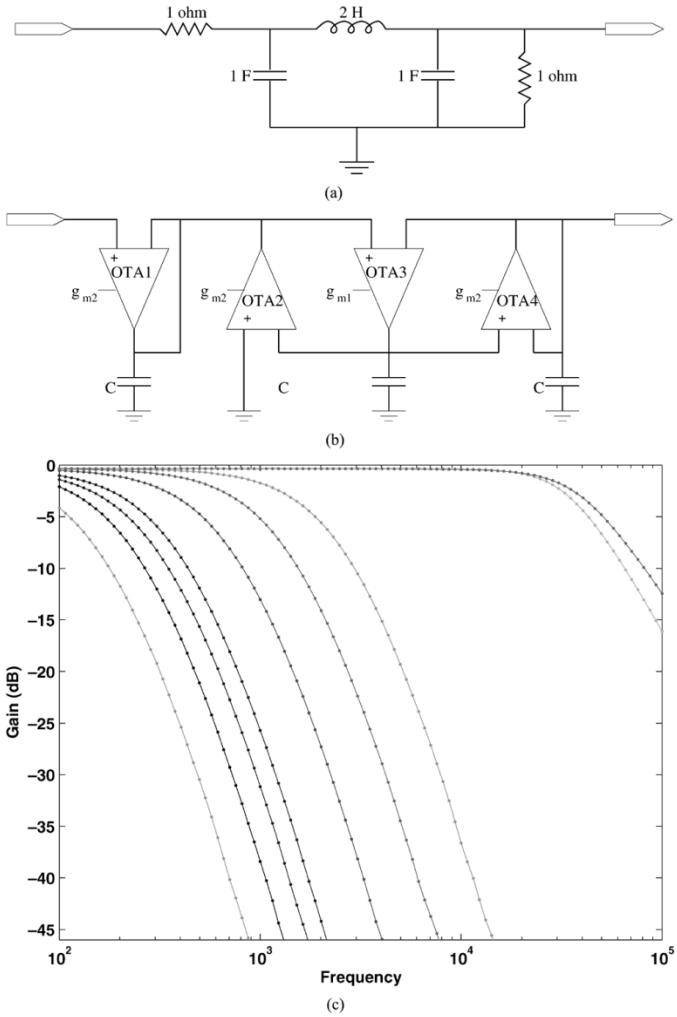

Fig. 12. (a) Canonical prototype of a third-order Butterworth double-resistance terminated *LC* filter. (b) The Gm–C implementation of the same filter. This form of the filter can be realized on the RASP 1.5 FPAA. (c) The experimental frequency response of a third-order Gm–C filter is shown here. The corner frequency is adjusted by programming the bias currents of the four OTAs.

the current through the nMOS current mirror (the other half of the current mirror is internal to the OTA module). To set small bias currents, it is preferrable to have the nFET and floating-gate transistor sized larger than the current mirror nFET internal to the OTA. In this configuration, the current mirror functions as a current divider, and thus, very low bias currents can be set by programming the floating-gate transistor to generate currents in the picoampere range.

The OTAs on the RASP 1.5 FPAA are standard nine-transistor wide-range OTAs with $W/L = 1.8/1.8$ and a bias transistor W/L of 8.1/1.8. Although the OTAs used in this example are fairly generic, the FPAA fabric is designed to be highly flexible, and the CABs can be enhanced with any OTA flavor desired. Thus, the overall architecture can remain fixed, while the specific components and their respective performance characteristics (signal-to-noise ratio, dynamic range, distortion, input linear range, etc.) can be modified for different target markets. Additionally, the use of floating-gate transistors to set the bias currents allows a large degree of freedom in adjusting the circuit characteristcs. Floating-gate transistors similar to those that set the bias current in the OTAs have been shown to be programmable over at least three and half orders of magnitude [29] and more recent efforts are yielding programmability over seven orders of magnitude [30].

B. Third-Order Gm–C Ladder Filter

The availability of OTAs and grounded capacitors makes the RASP ideal for implementing Gm–C filters. One way to realize a particular filter is by modeling it with resistors, inductors, and capacitors, and then synthesize the design using Gm–C filters. In this example, a third-order Butterworth filter is implemented. The canonical prototype of the filter, a double-resistance terminated *LC* filter, is shown in Fig. 12(a). By using the signal simulation method outlined in [31], the Gm–C filter shown in Fig. 12(b) is generated. In order to maintain a maximally flat response, the following must hold: $2 * g_{m1} = g_{m2}$. Accordingly, the bias current of OTA-3 was set to half of the other OTA bias

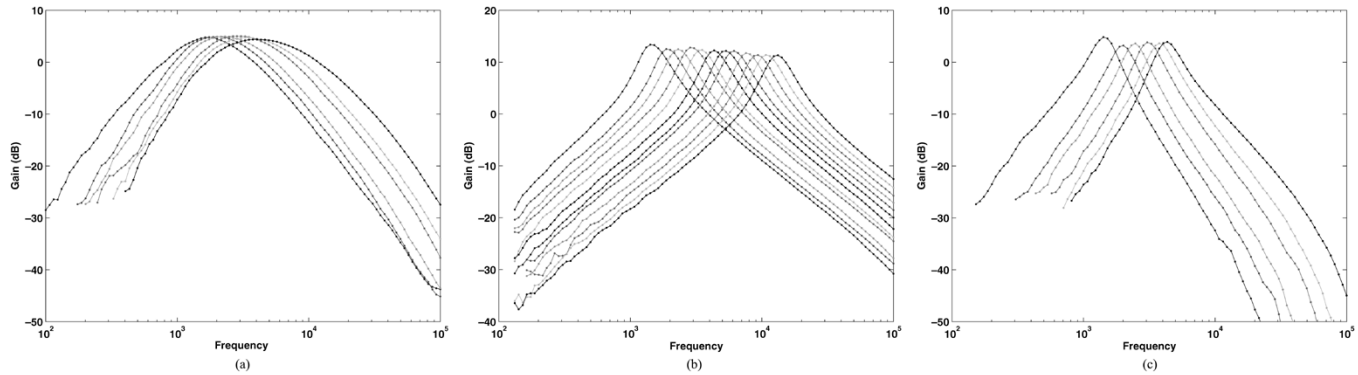

Fig. 13. The C^4 SOS block is comprised of two C^4 circuits with a buffer in between. By spreading the corner frequencies of one of the C^4 circuits to be far apart, the frequency response of the other C^4 can be measured. This method is used here to generate the frequency response plots for the first (a) and the second (b) C^4 circuits. Then, the frequency response for the SOS is generated by programming both C^4 to the same corner frequencies as shown in (c).

currents. A range of bias currents was used to create the frequency response shown in Fig. 12(c). As expected, the corner frequency of the filter is proportional to the bias currents of the OTAs. The lower corners were obtained by using a bias current in the range of hundreds of pico-amps, while the highest corners required currents of up to 1 μA.

C. Coarse-Grain CAB Components

As mentioned earlier, the CABs on this FPAA have several special-purpose components that have been designed to optimize specific functions. In particular, these CABs include programmable peak detectors and programmable bandpass filter modules (C^4 SOS circuits).

There is a wide range of systems that can be implemented and configured on FPAAs with many of these CABs on them. In particular, differentiators, cascaded SOSs, bandpass filters, matrix transforms (including DCTs and wavelet transforms), and frequency decomposition are all well suited for this architecture. In the audio arena alone, designs could be prototyped to implement forms of noise suppression, audio enhancement, feature extraction, auditory modeling, and simple audio array processing. Other potential interest areas include communications signal conditioning (modulation, mixing, etc.), transform coding, and neural networks (with external training). Many of these systems rely on efficient subband processing; therefore, each CAB has been designed with a C^4 SOS bandpass filter module to optimize this operation.

The C^4 SOS module is comprised of two C^4 modules cascaded with a buffer in-between them. Either C^4 module can be used alone by spreading apart the corner frequencies of the other module. To characterize this module, frequency response plots of each of the individual C^4 modules are shown in Fig. 13(a) and (b). The bandwidth and Q-peak of the C^4 modules are quite different. This is due to the difference in output capacitance of each module. The output of the first C^4 is tied to the input of a buffer, which results in a relatively small capacitance. However, the output of the second C^4 is tied into the switch network. Therefore, the output load capacitance for this device will be much higher due to the parasitics of the switches and the capacitance of the next circuit in the path. In these experiments, the next stage was a relatively large buffer in an output pad.

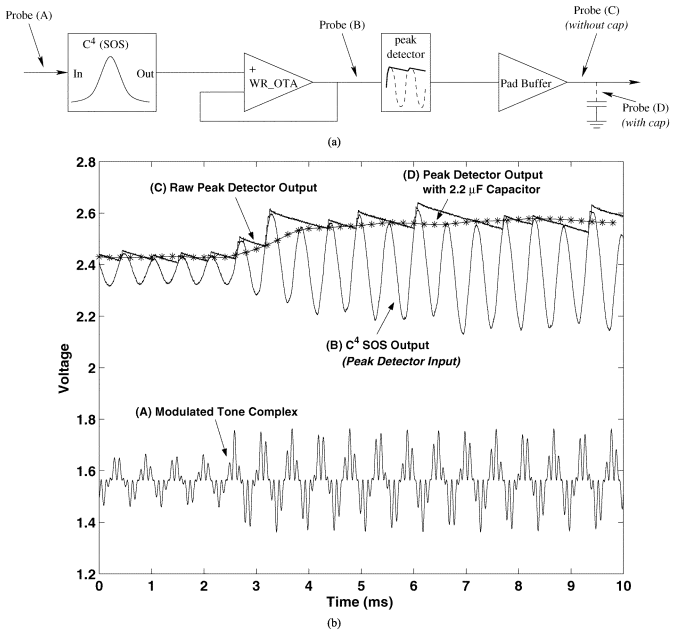

Fig. 14. Circuit diagram for a typical subband system. The incoming signal is bandpass filtered and then the magnitude of the subband is output from the peak detector. This is analagous to taking a discrete Fourier transform. (b) Experiment data from the FPAA for the system in part (a). The input waveform is an amplitude modulated signal with 1.8 kHz and 10.0 kHz components. The output of the peak detector is shown with and without an integrating capacitor added to the output stage.

When both of the cascaded C^4s are set to the same corner frequencies, the output of the module shows the desired second-order roll-off as shown in Fig. 13(c). In all of these plots, the corner frequencies are shown to be programmable over a wide range of frequencies. The bias current to corner frequency correlation is different for each of the cascaded devices. However, all of the bias currents for these plots were within the range of 25 pA to 200 nA.

The coarse-grain components are most useful when they can be combined to form a larger system. In Fig. 14(a), a circuit is shown that uses a C^4 SOS block, an OTA, and a peak detector in series. This configuration is very powerful when it is replicated 64, 128, or more times on the FPAA with the center frequencies of the bandpass filters varying over the desired frequency range. The outputs of the different subbands are analagous to

the magnitudes of the discrete Fourier transform. As a test of this system, data was taken from RASP 1.5 for a single subband. As shown in Fig. 14(b), the input is an amplitude-modulated signal with 1.8 and 10.0 kHz frequency components. The C^4 SOS module is biased to have a center frequency near 1.8 kHz, and the OTA is configured to be a noninverting buffer. The output of the system is shown in Fig. 14(b). Also, the output of the system is shown after an external 2.2 μF capacitor has been added at the output of the FPAA. This change has the effect of smoothing (i.e., low-pass filtering) the output, thus creating a longer effective time constant for the system.

VI. Conclusion

Large-scale FPAAs based on floating-gate technologies provide the necessary levels of programmability and functionality to implement complex signal processing systems. The floating-gate transistors are shown to provide a compact switch that exhibits relatively flat resistance characteristics across the full operating voltage and can be programmed to be an active circuit element (variable resistor). FPAAs based on floating-gate circuits have been built and characterized, and system-level results have been shown. Systems implemented on these FPAAs are demonstrated to be programmable over a wide range of frequencies, Q-peaks, bandwidths, and/or time constants. With orders of magnitude power consumption savings over traditional digital approaches, this reconfigurable analog technology offers an attractive alternative for implementing advanced signal processing systems in low-power embedded systems.

References

[1] M. A. Sivilotti, "Wiring Considerations in Analog VLSI Systems, With Application to Field-Programmable Networks (VLSI)," Ph.D., California Institute of Technology, Pasadena, CA, 1991.

[2] K. Lee and P. Gulak, "A transconductor-based field-programmable analog arry," in *Dig. Tech. Papers IEEE Int. Solid-Sate Conf.*, Feb. 1995, pp. 198–199.

[3] ——, "A CMOS field-programmable analog array," in *Dig. Tech. Papers IEEE Int. Solid-Sate Conf.*, Feb. 1991, pp. 186–188.

[4] S. Chang, B. Hayes-Gill, and C. Paul, "Multi-function block for a switched current field programmable analog array," in *Proc. Midwest Symp. Circuits and Systems*, vol. 1, Aug. 1996, pp. 158–161.

[5] D. Anderson, C. Marcjan, D. Bersch, H. Anderson, P. Hu, O. Palusinski, D. Gettman, I. Macbeth, and A. Bratt, "A field programmable analog array and its application," in *Proc. IEEE Custom Integrated Circuits Conf.*, May 1997, pp. 555–558.

[6] "ispPAC Overview," Lattice Semiconductor Corporation, Hillsboro, OR, 1999.

[7] X. Quan, S. Embabi, and E. Sanchez-Sinencio, "A current-mode based field programmable analog array architecture for signal processing applications," in *Proc. IEEE Custom Integrated Circuits Conf.*, Santa Clara, CA, May 1998, pp. 277–280.

[8] "Totally re-configurable analog circuit-TRAC," Fast Analog Solutions, Ltd, Oldham, U.K., 1999.

[9] C. A. Looby and C. Lyden, "A CMOS continuous-time field programmable analog array," in *Proc. 5th Int. ACM Symp. Field-Programmable Gate Arrays*, 1997, pp. 137–141.

[10] T. S. Hall, C. M. Twigg, P. Hasler, and D. V. Anderson, "Developing large-scale field-programmable analog arrays for rapid prototyping," *Int. J. Embed. Syst.*, 2004.

[11] G. Franz, "Digital signal processor trends," *IEEE Micro*, vol. 20, no. 6, pp. 52–59, Nov.–Dec. 2000.

[12] R. Ellis, H. Yoo, D. Graham, P. Hasler, and D. Anderson, "A continuous-time speech enhancement front-end for microphone inputs," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 2, Phoenix, AZ, May 2002, pp. II.728–II.731.

[13] P. Hasler, P. Smith, R. Ellis, D. Graham, and D. V. Anderson, "Biologically inspired auditory sensing system interfaces on a chip," in *Proc. IEEE Sensors Conf.*, Orlando, FL, Jun. 2002, invited Paper.

[14] P. D. Smith, M. Kucic, R. Ellis, P. Hasler, and D. V. Anderson, "Mel-frequency cepstrum encoding in analog floating-gate circuitry," in *Proc. Int. Symp. Circuits and Systems*, vol. 4, 2002, pp. 671–674.

[15] T. S. Hall, P. Hasler, and D. V. Anderson, "Field-programmable analog arrays: A floating-gate approach," in *Proc. 12th Int. Conf. Field Programmable Logic and Applications*, Montpellier, France, Sep. 2002, pp. 424–433.

[16] H. W. Klein, "The EPAC architecture: An expert cell approach to field programmable analog circuits," in *Proc. IEEE Midwest Symp. Circuits and Systems*, vol. 1, Aug. 1996, pp. 169–172.

[17] P. Hasler, B. A. Minch, and C. Diorio, "Adaptive circuits using pFET floating-gate devices," in *Proc. 20th Anniversary Conference on Advnaced Research in VLSI*, Atlanta, GA, Mar. 1999, pp. 215–229.

[18] M. Kucic, A. Low, P. Hasler, and J. Neff, "A programmable continuous-time floating-gate Fourier processor," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 90–99, Jan. 2001.

[19] T. S. Hall, C. M. Twigg, P. Hasler, and D. V. Anderson, "Developing large-scale field-programmable analog arrays," in *Proc. 18th Int. Parallel and Distributed Processing Symp.*, Santa Fe, NM, Apr. 2004.

[20] M. Kucic, P. Hasler, J. Dugger, and D. V. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *Proc. Conf. Advanced Research in VLSI*, Mar. 2001, pp. 148–162.

[21] P. Smith, M. Kucic, and P. Hasler, "Accurate programming of analog floating-gate arrays," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 5, Phoenix, AZ, May 2002, pp. V.489–V.492.

[22] J. D. Gray, C. M. Twigg, D. N. Abramson, and P. Hasler, "Characteristics and programming of floating-gate pFET switches in an FPAA crossbar network," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 1, May 2005, pp. 468–471.

[23] B. Ray, P. P. Chaudhuri, and P. K. Nandi, "Design of OTA based field programmable analog array," in *Proc. 13th Int. Conf. VLSI Design*, Jan. 2000, pp. 494–498.

[24] 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, pp. 125–136, Feb. 2002.

[25] E. Sanchez-Sinencio, J. Ramirez-Angulo, B. Linares-Barranco, and A. Rodriguez-Vazquez, "OTA-based nonlinear function approximations," *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 1, pp. 96–99, May 1989.

[26] (1999) Totally re-configurable analog circuit-TRAC. Fast Analog Solutions Ltd., Manchester, U.K.. [Online]. Available: http://www.zetex.com

[27] T. S. Hall, C. M. Twigg, P. Hasler, and D. V. Anderson, "Application performance of elements in a floating-gate FPAA," in *Proc. 2004 IEEE Int. Symp. Circuits and Systems*, May 2004, pp. II.589–II.592.

[28] 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.

[29] A. Bandyopadhyay, G. J. Serrano, and P. Hasler, "Programming analog computational memory elements to 0.2% accuracy over 3.5 decades using a predictive method," in *Proc. IEEE Int. Symp. Circuits and Systems*, May 2005, pp. 2148–2151.

[30] P. D. Smith and P. Hasler, "A kappa projection algorithm (KPA) for programming to femtoampere currents in standard CMOS floating-gate elements," in *Proc. Midwest Symp. Circuits and Systems*, Cincinnati, OH, 2005.

[31] Y. E. Sun, *Design of High Frequency Integrated Analogue Filters*. London, U.K.: The Institution of Electrical Engineers, 2002.

[32] F. Baskaya, S. Reddy, S. K. Lim, T. Hall, and D. Anderson, "Mapping algorithm for large-scale field programmable analog array," in *Proc. 2005 ACM Int. Symp. Physical Design*, San Fransisco, CA, Apr. 2005.

Tyson S. Hall, photograph and biography not available at the time of publication.

Christopher M. Twigg, photograph and biography not available at the time of publication.

Jordan D. Gray, photograph and biography not available at the time of publication.

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