DEVELOPING LARGE–SCALE FIELD–PROGRAMMABLE ANALOG ARRAYS

Tyson S. Hall, Christopher M. Twigg, Paul Hasler, and David V. Anderson

Georgia Institute of Technology, Atlanta, GA 30332-0250, tyson@ece.gatech.edu

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, flexibility, and usefulness. In this paper, we explore the use of floating–gate devices as the core programmable element in a large–scale FPAA with applications in signal processing emphasized. An FPAA architecture is presented that offers increased functionality and flexibility in realizing analog signal processing systems, and experimental data from a testbed FPAA is shown. In addition, mainstream signal processing systems are discussed that can be effectively implemented on large–scale reconfigurable analog devices thereby realizing dramatic savings in power over traditional digital solutions and improved time–to–market over traditional analog designs.

1. LOW–POWER SIGNAL PROCESSING

The future of field–programmable analog arrays (FPAAs) lies in their ability to speed the implementation of advanced, low–power signal processing systems. In this paper, we present an architecture for achieving flexible, large–scale FPAAs targeted at mainstream signal processing systems. These FPAAs are intended to impact analog signal processing in two ways: first, they perform the function of all rapid prototyping devices in reducing development time. Second, they are a platform for implementing advanced signal processing functions, usually reserved for only digital systems, in analog circuits.

The primary benefit of implementing signal processing systems in analog is the potential for large savings in power consumption. For DSP microprocessors, Gene's law postulates that the power consumption, as measured in mW/MIPS, is halved about every 18 months [1]. These advances largely follow Moore's law, and they are achieved by using decreased feature size and other refinements, such as intelligent clock gating. Unfortunately, a problem looms on the horizon; the power consumption of the analog–to–digital converter (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.

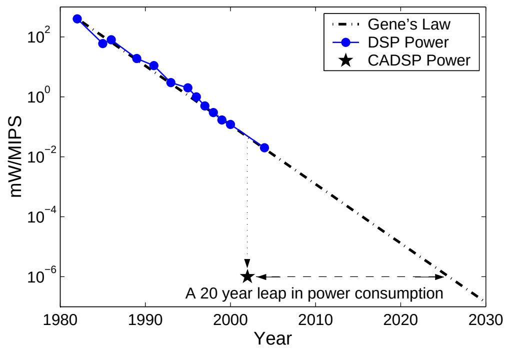


Fig. 1. Data from [1] showing the power consumption trends in DSP microprocessors along with data taken from a recent analog, floating–gate integrated chip developed by the CADSP team.

Functionality	DSP μP	Trad. Analog	Large–scale FPAA
Programmable	●	–	●
Monolithic Filters	○	●	●
Linear Scalar	●	○	●
Nonlinear Scalar	○	●	●
Vector–Matrix	○	○	●
Linear–phase Fltrs	●	–	○
Adaptivity	○	–	○
Tap Delay Lines	●	○	○

Key:
– = No or very limited support
○ = Possible
● = Efficient, well–suited to technology

Table 1. Summary of Signal Processing Functionality

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 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, resulting in both a simpler ADC and a substantially reduced computational load on the digital processors that follow. Experimental data from analog

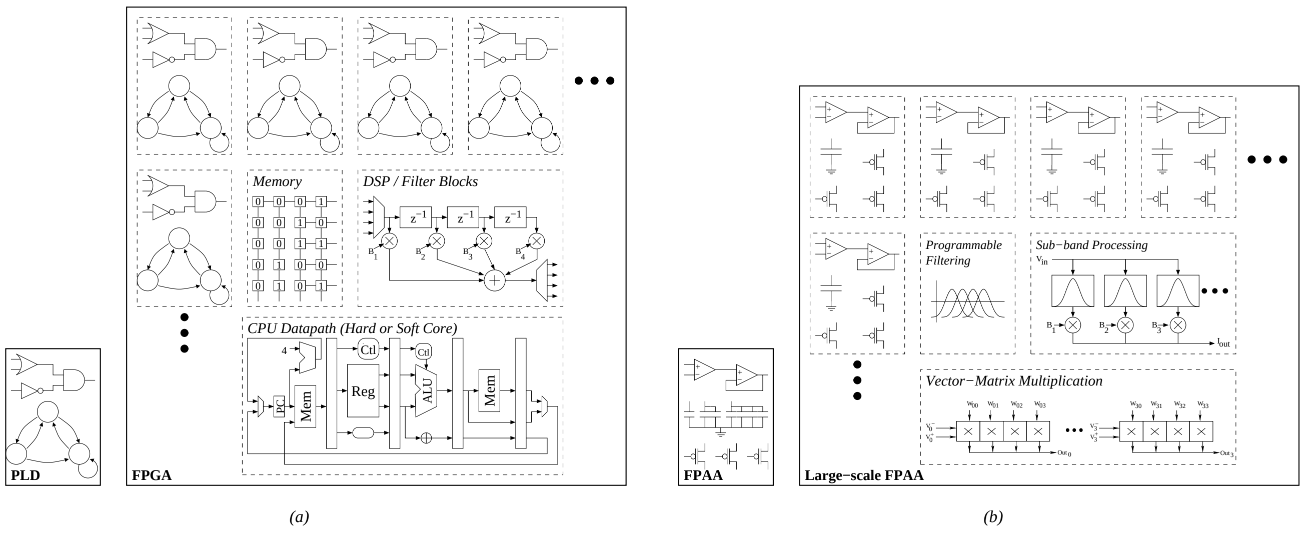


Fig. 2. (a) Digital PLDs can be used to implement small, carefully defined pieces of a complex system, while FPGAs can be used to implement entire systems including processor datapaths, complex DSP functions, and more. Modern FPGAs can be 100 - 10,000 times larger and more complex than the PLDs of the 1970s and 1980s. (b) Analagously, 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 FPAAs based on floating–gate devices presented here 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 programmable op–amp and transistor elements.

signal processing systems has shown that power requirements can be decreased up to five orders of magnitude over typical DSP microprocessor implementations [2, 3]. As illustrated in Fig. 1, this corresponds to a 20 year leap forward on the power curve predicted by Gene's Law [4].

For analog systems to be desirable to the largely digital signal processing commmunity, they not only need to have a significant advantage in terms of size and power but they must be relatively easy to use and easily integrated into a larger digital system. In addition, they must be shown to be accurately programmable and effective at implementing many of the key systems found within digital signal processing (DSP). As shown in Table 1, the functionality desired for any technology focused on signal processing includes monolithic filters, linear and nonlinear scalar functions, vector–matrix operations (i.e., transforms, distance metrics, winner-take-all, principle component analysis, etc.), linear–phase filters, adaptation, and tap delay lines for FIR systems.

2. LARGE–SCALE FPAAS

FPAAs are the analog equivalent of digital field–programmable gate arrays (FPGAs). FPAAs provide a reconfigurable platform that can be used to implement a number of different 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 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 [5, 6, 7, 8, 9]. The next–generation FPAA needs to correct these problems in order to extend the usefulness and acceptance of FPAAs. As shown in Fig. 2, 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 that we are exploring in this paper are more analogous to modern FPGAs. Our proposed 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 medium–grain, programmable analog blocks (e.g., operational transconductance amplifiers (OTAs), transistor elements, capacitors, etc.).

Ideally, one would like to base future FPAAs on a technology that provides a small, easily programmable element that can be configured to act as an ideal switch, variable resistor, and configurable computational element. While such an ideal element is currently unattainable, advances in the area of floating–gate transistors have led to an analog technology that is very small, accurately programmable, and extremely low in power consumption. Previously, we have proposed that the floating–gate transistor can be used as a non-ideal switch, variable resistor, and programmable element within larger computational blocks (e.g., analog multiplier, programmable filter, programmable OTAs, etc.) [10]. A small, testbed FPAA has been fabricated to study the effects of the floating–gate transistor within a reconfigurable environment. Named the Reconfigurable Analog Signal Processor (RASP), this FPAA includes a 64 x 16 crossbar switch interconnect and a computational analog block (CAB) similar to that shown in Fig. 4b. Floating–gate transistors are used as both the crossbar switches and the programmable element within the computational logic.

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) [11]. 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 [11, 12, 13].

The architecture of an FPAA based on floating–gate transistors is such that a single programmable device is used throughout the design. The floating–gate transistor is used as the switch within the crossbar network, and it is used within the computational analog blocks (CABs) to program the characteristics and functionality of the computational elements.

The floating–gate switch network has been characterized in [4]. The switches were found to exhibit similar characteristics to standard pFET switches with an "on" resistance as low as 11 kΩ and an "off" resistance in the low gigaohm range. They have also been shown to be accurately programmable and capable of implementing a variable resistance. As shown in Fig. 3, the floating–gate switch can be programmed in between the "on" and "off" extremes and used as a nonlinear impedance. This allows for a very compact architecture in contrast to many traditional FPAA designs, which require large resistor or capacitor arrays to achieve this same functionality. In addition, routing complexity is reduced because a separate resistance or capacitance element does not have to be included in the signal path.

To increase the quality of a switch, the floating–gate transistors are programmed to the far extremes of their range. In this case, one of the limiting factors is the ability of the measurement equipment to measure the very small currents present as the switch is programmed "off." To extend the viable programming range, current measurements are taken at larger V_{DD}'s as shown in Figure 3. Measuring the currents with $V_{DD} = 6.5\ V$, allows the I–V curves to be visible to

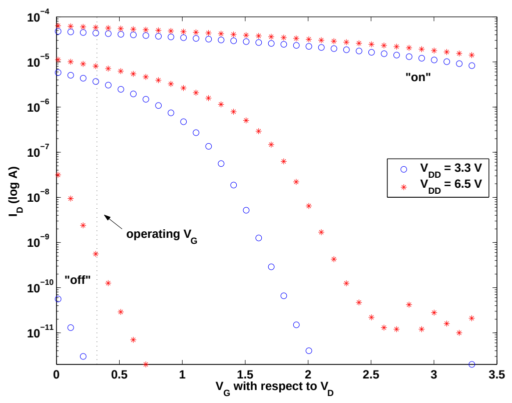


Fig. 3. Floating–gate switches can be programmed within a wide range. Here, examples of an "on", "off", and mid–position device are shown. During programming, currents are measured with $V_{DD} = 3.3V$ for large currents and $V_{DD} = 6.5V$ for small currents. This effectively extends the programming range of the device.

the programming infrastructure 1 V below the point visible when $V_{DD} = 3.3\ V$.

In the operating mode of this FPAA, the voltage on the gate capacitor for all switches is the same. From Figure 3, 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_{DD}. Thus, an operating gate voltage of 0.3 V is deemed optimal for the current programming scheme.

Within the CABs, floating–gate transistors are used to set bias voltages for the OTAs (see Fig. 5a), adjust the corner frequencies on the capacitively coupled current conveyors (C^4s), and set muliplier coefficients 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.

3. COMPUTATIONAL ANALOG BLOCKS

The computational logic is organized in a compact computational analog block (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 Figure 4a. A sample FPAA with 64 CABs on a single chip fabricated in TSMC 0.35-micron is estimated to cover an area of approximately 36 mm^2. Of course, commercially viable FPAAs are forseen that have 100s if not 1000s of CABs based on this same architecture.

The programmable elements in the FPAA are floating–gate transistors, and they are used for both the switch interconnects and the computational logic as characterized in [4]. Thus, a single programming infrastructure will be used since all programmable elements will be floating–gate

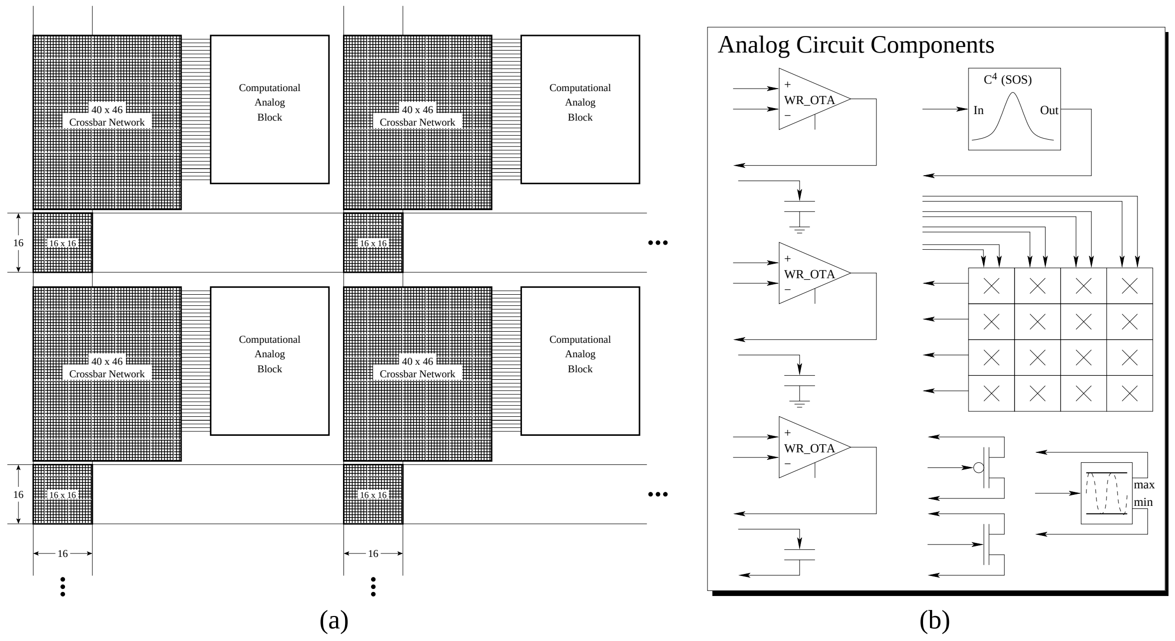


Fig. 4. (a) This is the overall block diagram for a large–scale FPAA. The switching interconnects are fully connectable crossbar networks built using floating–gate transistors. (b) This is a Computational Analog Block (CAB) for an FPAA based on floating–gate devices. Here, each CAB contains a four-by-four matrix multiplier, three wide–range operational transconductance amplifiers (OTAs), three fixed–value capacitors, a capcatively coupled current conveyor (C^4), a peak detector, and two FET transistors. The input and output signals shown in this figure are routed to the rows of the switch matrix.

transistors.

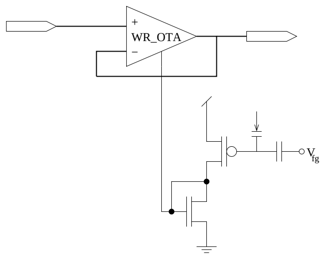
Many example CABs can be imagined using this technology. Figure 4b shows one example CAB, whose functionality is enhanced by a mixture of medium– and coarse–grained computational blocks similar to many modern FPGA designs. The computational blocks were carefully selected to provide a sufficiently flexible, generic architecture while optimizing certain frequently used signal processing blocks. For generality, three operational transconductance amplifiers (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 non-linear functions [14, 7, 15, 9]. 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 [4, 7]. 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 capacitively coupled current conveyor (C^4) used as a band–pass filter module and the 4 x 4 vector–matrix multiplier block. In general, the C^4 module provides a straightforward method of sub–banding 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 [16, 12]. In addition, a peak detector is added to each CAB.

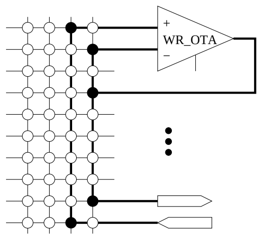
4. TESTBED FPAA

The testbed FPAA based on floating–gate devices was fabricated in a 0.5-micron, standard CMOS process. This FPAA contains two CABs with a floating–gate crossbar switch network connecting them [10]. The CAB design was slightly smaller than the one outlined in Section 3 having a C^4 bandpass filter module, 4 x 4 vector–matrix multiplier, and three wide–range OTAs. However, this design is more than sufficient to test the concept of FPAAs with floating–gate devices and characterize the elements of the CAB.

As an initial example of the testbed system, a first–order filter is implemented using an OTA in one of the CABs. Figure 5 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 cor-



(a)



(b)

Fig. 5. (a) The 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) is fixed. 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 Computational Analog Blocks (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.

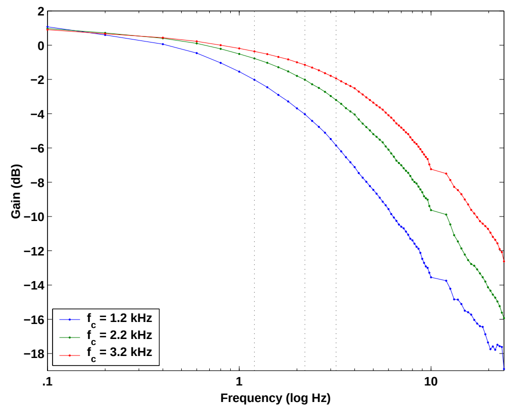
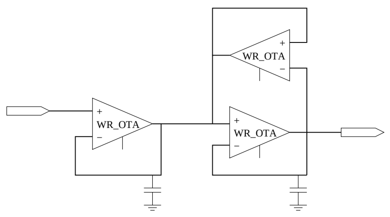


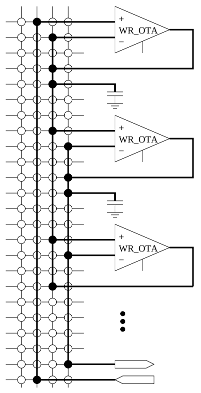
Fig. 6. Here, the frequency response of the source–follower circuit is shown for several bias currents. An internal floating–gate transistor is used as a current source to set the OTA's bias. Injecting the floating–gate device, increases the current and thus the bandwidth of this first order filter.

ner frequencies in Figure 6. The moderate gain in the lower frequencies is due to the switches in the feedback loop of the OTA. Ideally, the output node and the negative input node would be directly connected. However, in the FPAA, this path must be routed via the switch network, which means that a minimum of two floating–gate switches will be in the feedback loop. The gain can be minimized by injecting the floating–gates of these switches to a lower charge, or if gain is desired for a given application, then it can be set by programming these switches to a higher charge.

In Figure 7, a second–order section filter is shown along side the FPAA implementation. Once again, explicit capacitors are eliminated since the switch parasitics provide the necessary capacitance. Using the floating–gate programmble biases, the two OTAs in a source–follower configuration were biased to the same level and the the third OTA's bias current was increased to adjust the Q–peak of the system.



(a)



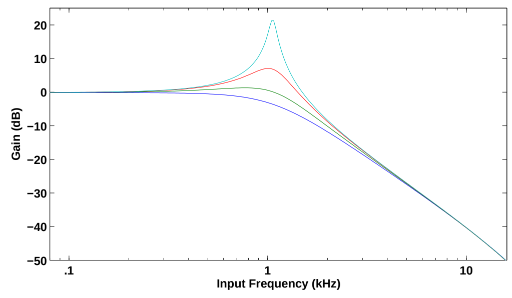
(b)

Fig. 7. (a) A second–order section 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 second–order section configuration. The programmable biases shown in Figure 5(a) are not included here for simplicity, but each OTA has a current mirror and floating–gate current source that sets its bias.

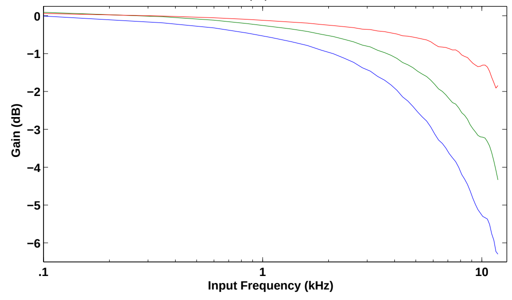
The frequency response for this circuit is shown in Figure 8. As expected, the Q–peak increases as the bias current (e.g., conductance) increases.

For second–order functions such as the second–order section and diff2 circuit, reasonable Q–peaks and filter bandwidths require small bias currents (in the picoamp to femtoamp 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. 3 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. 5a sets 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 picoamp range.

Based on these testbed systems, one can start to imagine a wide class of systems that can be implemented and configured on FPAAs with many of these CABs on them. In particular, differentiators, cascaded second–order sections, 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 modelling, and simple audio array processing. Other potential interest areas include communications signal con-



(a)



(b)

Fig. 8. The simulated (a) and experimental (b) frequency response of a second–order section 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.

ditioning (modulation, mixing, etc.), transform coding, and nerual networks (with external training). Most of these systems rely on efficient sub–band processing; so, each CAB has been designed with a C^4 bandpass to optimize this operation. As shown in Fig. 9, the center frequency of the C^4 filter can be moved over a large range of frequencies.

5. CONCLUSION

Large–scale FPAAs based on floating–gate technologies provide the necessary levels of programmability and functionality to implement complex signal processing systems. 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 devices. A testbed FPAA based on floating–gate circuits has been built and inital results have been shown.

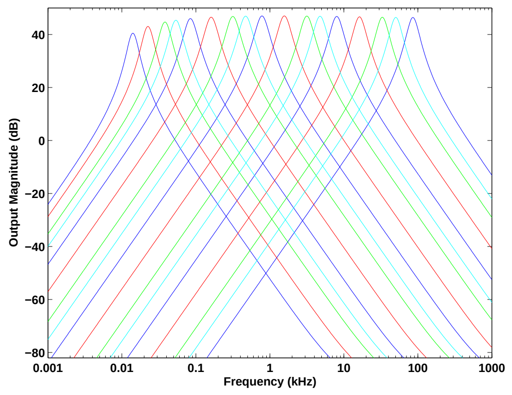


Fig. 9. Frequency decomposition (sub–band processing) can be achieved on the test–bed FPAA by using the C^4 bandpass filter block in each CAB. Here the center frequency of the C^4 is shown to be programmable over a wide range of frequencies.

6. REFERENCES

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

[2] R. Ellis, H. Yoo, D. Graham, P. Hasler, and D. Anderson, "A continuous-time speech enhancement front-end for microphone inputs," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Phoenix, AZ, 2002, vol. 2, pp. II–728 – II–731.

[3] P. D. Smith, M. Kucic, R. Ellis, P. Hasler, and D. V. Anderson, "Mel–frequency cepstrum encoding in analog floating-gate circuitry," in *Proceedings of the International Symposium on Circuits and Systems*, Phoenix, AZ, May 2002, pp. IV–671 – IV–674.

[4] T. S. Hall, C. M. Twigg, P. Hasler, and D. V. Anderson, "Application performance of elements in a floating–gate FPAA," in *Accepted to ISCAS*, 2004.

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

[6] K.F.E. Lee and P.G. Gulak, "A transconductor-based field-programmable analog array," in *IEEE International Solid–State Conference Digest of Technical Papers*, Feb. 1995, pp. 198–199.

[7] B. Pankiewicz, M. Wojcikowski, S. Szczepanski, and Y. Sun, "A field programmable analog array for cmos continuous–time ota-c filter applications," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 2, pp. 125–136, Feb. 2002.

[8] Anadigm, http://www.anadigm.com/Supp_05.asp/, *Anadigm FPAA Family Overview*, Mar. 2003.

[9] Fast Analog Solutions Ltd., http://www.zetex.com, *Totally reconfigurable analog circuit – TRAC®*, Mar. 1999.

[10] T. S. Hall, P. Hasler, and D. V. Anderson, "Field–programmable analog arrays: A floating–gate approach," in *Proc. 12th International Conference on Field Programmable Logic and Applications*, Montpellier, France, Sept. 2002, pp. 424–433.

[11] P. Hasler, B. A. Minch, and C. Diorio, "Adaptive circuits using pfet floating-gate devices," in *Proc. of the 20th Anniv. Conference on Advnaced Research in VLSI*, Atlanta, GA, March 1999, pp. 215–229.

[12] M. Kucic, A. Low, P. Hasler, and J. Neff, "A programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 90–99, Jan. 2001.

[13] 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 *Accepted to ISCAS*, 2004.

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

[15] E. Sanchez-Sinencio, J. Ramirez-Angulo, B. Linares-Barranco, and A. Rodriguez-Vazquez, "OTA-based non-linear function approximations," in *ISCAS Proc.*, May 1989, vol. 1, pp. 96–99.

[16] P. Hasler, B. A. Minch, and C. Diorio, "An autozeroing floating-gate amplifier," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 74–82, Jan. 2001.