2005

Developing large-scale field-programmable analog arrays for rapid prototyping

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

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

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

Developing large-scale field-programmable analog arrays for rapid prototyping

Tyson S. Hall*

School of Computing,
Southern Adventist University,
PO Box No. 370, Collegedale, TN 37315-0370, USA
E-mail: tyson@southern.edu

Christopher M. Twigg, Paul Hasler and David V. Anderson

Georgia Institute of Technology,
Atlanta, GA 30332-0250, USA
E-mail: ctwigg@ece.gatech.edu E-mail: phasler@ece.gatech.edu
E-mail: dva@ece.gatech.edu

Abstract: Field-programmable analog arrays (FPAAs) provide a method for rapidly prototyping analog systems. While currently available FPAAs vary in architecture and interconnect design, they are often limited in size and flexibility. For FPAAs to be as useful and marketable as modern digital reconfigurable devices, 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. By leveraging recent advances in floating gate transistors, a new generation of FPAAs are achievable that will dramatically advance the current state of the art in terms of size, functionality, and flexibility.

Keywords: FPAA; field programmable analog arrays; reconfigurable; analog; array.

Reference to this paper should be made as follows: Hall, T.S., Twigg, C.M., Hasler, P. and Anderson, D.V. (2005) 'Developing large-scale field-programmable analog arrays for rapid prototyping', *Int. J. Embedded Systems*, Vol. 1, Nos. 3/4, pp.179–192.

Biographical notes: Tyson S. Hall received the PhD, MSECE, and BSCMPE degrees in electrical and computer engineering from the Georgia Institute of Technology in 2004, 2001, and 1999. He is currently an Assistant Professor in the School of Computing at Southern Adventist University in Collegedale, Tennessee. His research interests include rapid prototyping of mixed–signal systems, cooperative analog/digital signal processing, reconfigurable computing, and embedded systems education.

Christopher M. Twigg received his BSEE and BSCPE from West Virginia University in 2000 and his MSECE from the Georgia Institute of Technology in 2002. He is currently pursuing his PhD at the Georgia Institute of Technology. His research interests include Reconfigurable and Programmable Mixed signal Systems, Cooperative Analog/Digital Signal Processing, and Floating Gate Circuits.

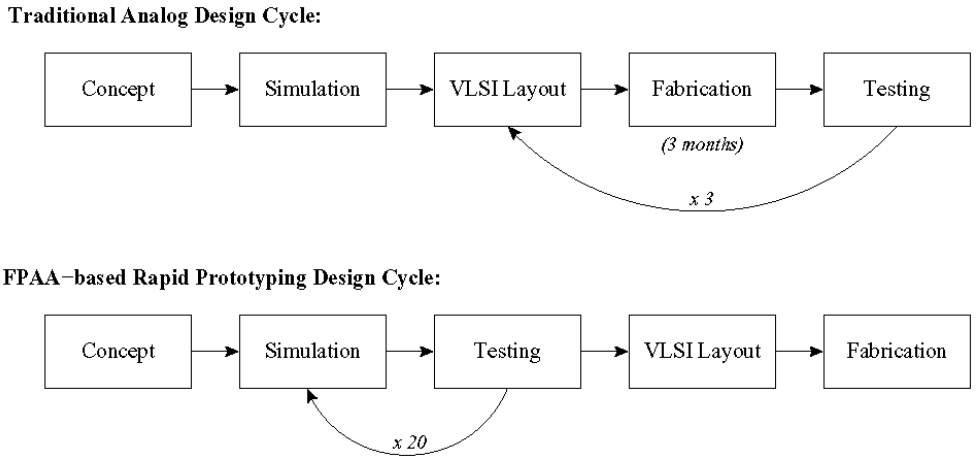
Paul Hasler received his MS and BSE Degrees in Electrical Engineering from Arizona State University, Tempe, in 1991, and his PhD Degree from the California Institute of Technology, Pasadena in Computation and Neural Systems in 1997. He is an Associate Professor in the School of Electrical and Computer Engineering at Georgia Tech. His current research interests include low-power electronics, mixed-signal system ICs, floating gate MOS transistors, adaptive information processing systems, smart sensor interfaces, and device physics related to submicron and floating gate devices. He received the NSF CAREER Award in 2001, and the ONR YIP Award in 2002.

David V. Anderson received the BS and MS Degrees from Brigham Young University, Provo, UT and the PhD Degree from Georgia Institute of Technology (Georgia Tech) Atlanta, GA, in 1993, 1994, and 1999, respectively. He is an Associate Professor in the School of Electrical and Computer Engineering at Georgia Tech and an Associate Director of the Center for Research in Embedded Systems Technology. His research interests include low-power signal processing techniques using both analog and digital hardware. He is a recipient of the 2004 Presidential Early Career Awards for Scientists and Engineers (PECASE).

1 Rapid-prototyping analog systems

The process of designing, fabricating, and testing an analog chip requires a certain level of expertise and is often long and expensive. As shown in Figure 1, the process is not unlike designing digital ASICs (application specific integrated circuits), except that there are fewer tools and libraries available to the designer. The traditional analog design cycle often requires several iterations of the fabrication process, which with the simulation, VLSI layout, and testing phases can easily consume a year or more for typical IC designs. However, an FPAA dramatically reduces the design cycle by removing the fabrication stage from the iterative process. Thus, many designs may be tested and modified within a single day.

Figure 1 This figure illustrates the advantages of designing analog ICs using an FPAA based rapid prototyping technology as opposed to the traditional design cycle of VLSI layout and fabrication. The traditional analog design cycle often requires 3 or more iterations of the fabrication process which extends the development process to over a year. With an FPAA based system, designs can be synthesised, tested and modified 20 or more times within a matter of days, instead of years



Programmable analog devices have benefits and design similar to FPGAs. Like FPGAs, the analog arrays, dubbed field programmable analog arrays (FPAAs), are not optimal for all solutions. They are, however, very useful for many situations and a solution can be found for many problems not requiring tight specifications. Relative to custom designed analog circuits, a design implemented on an FPAA results in higher parasitics as well as increased die area for a given design; therefore, the design always possesses some inefficiencies (measured in lower bandwidth and higher consumed power). On the other hand, analog circuit design is often time consuming and these adverse tradeoffs are well balanced by decreased time to market.

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 (Sivilotti, 1991; Lee and Gulak, 1995, 1991a; Chang et al., 1996; Anderson et al., 1997; ispPAC Overview, 2001; Quan et al., 1998; Totally Reconfigurable Analog Circuit – TRAC, 1999; Looby and Lyden, 1997). By building larger, more flexible FPAAs, reconfigurable analog devices will become more analogous to today's high density FPGA architectures. This will allow a very useful rapid prototyping system to be built for analog circuit development. Furthermore, an integrated, rapid prototyping system can be built with an FPAA and an FPGA on board that will allow engineers to prototype analog, digital and mixed signal systems, all on one station.

The FPAA explored in this paper leverages the recent advances in floating gate technology to provide computational logic that is programmable within a compact, scalable architecture (Hall et al., 2002, 2004a). The computational logic includes a mix of coarse, medium, and fine grain blocks to provide a balance between flexibility and performance. In addition, floating gate transistors are used to set the biases, coefficients and other parameters of the computational analog blocks so that the analog elements are highly configurable, unlike most traditional designs. Floating gate transistors are also used as the switches.

This provides a very compact switch that can be programmed 'on', 'off', or as an in-circuit resistive element. The ability to use the switches as in-circuit elements results in both greater flexibility and a smaller area than traditional designs.

This paper proceeds with an overview of the past FPAA work in Section 2. In Section 3 the power savings demonstrated in analog circuits will be discussed and in Section 4, a brief overview of the advances of floating gate technologies will be presented. In Section 5, a new architecture is presented for large-scale FPAAs, including experimental data from a testbed FPAA fabricated with this new architecture.

2 Overview of past and present FPAAs

Reconfigurable hardware has long been of interest to circuit designers and engineers. In the digital domain, programmable logic devices (PLDs) have made a large impact on the development of custom digital chips by enabling a designer to try custom designs on easily reconfigurable hardware. Since their conception in the late 1960s and early 1970s, PLDs have evolved into today's high density field programmable gate arrays (FPGAs) (Wakerly, 1999; Birkner and Chua, 1978; Chow et al., 1999). Modern FPGAs are widely used in the laboratory for rapidly prototyping digital hardware, as well as in production goods to decrease time to market and to allow products to be easily upgraded after being deployed.

In the analog domain, however, progress has been much slower. While early analog integrated circuits (ICs) were often tuneable with adjustable biases, truly reconfigurable analog circuitry in the form of field programmable analog arrays (FPAAs) did not emerge until the late 1980s (Sivilotti, 1991; Gulak, 1995), and commercial offerings did not reach the market until 1996 (Marsh, 2001).

Field programmable analog arrays (FPAAs) can be broadly classified into two categories: continuous time devices and discrete time devices (Ganesan and Vemuri, 2001). There are academic and commercial examples of both categories as well as advantages unique to each design methodology. In addition, previous FPAAs have varied greatly in terms of computational granularity and capability, interconnect structure, performance, and application focus.

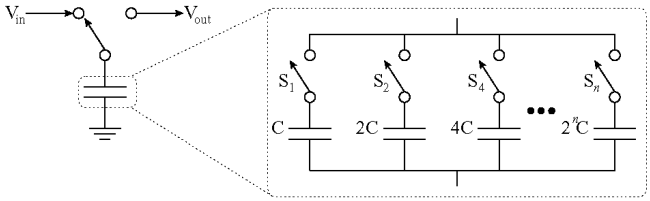
2.1 Discrete time FPAAs

Discrete time FPAAs are typically switched-capacitor designs. For these circuits, the incoming voltage is sampled by opening and closing a switch that connects the input to an initial capacitor as shown in Figure 2. The switch and capacitor form a type of analog register and the system's signal path is partitioned by these registers. The basic computational elements are usually operational amplifiers and analog registers, which synthesise a linear resistor whose value is determined by the switching rate and capacitor value. The synthesis of linear variable resistors gives switched-capacitor FPAAs greater flexibility than traditional continuous time FPAAs; however, they can also be harder to design well, because the switches and capacitors can introduce noise and nonlinearities into the system that must be overcome (Marsh, 2001). In addition, these designs have a limited bandwidth based on the sampling rate, are complicated by the need for continuous time antialiasing and reconstruction filters at the input and output, and can be large if programmable capacitor arrays are included (Gulak, 1995; Marsh, 2001).

In the late 1990s, several switched-capacitor FPAAs were introduced by both academic and commercial entities. In the academic arena, basic computational elements vary from the simple operational amplifier (Koneru et al., 1999; Edwards et al., 2000) to more complex blocks such as a lossless integrator and lossy integrator connected in a loop (Kutuk and Kang, 1996). These devices also can have programmable capacitor and/or programmable resistor arrays to add programmability (Edwards et al., 2000; Gulak, 1995). In the commercial arena, Motorola was one of the first companies to bring a general purpose FPAA to market with their MPAA020 and MPA1000 series (Bratt, 1998; Anderson et al., 1997; Marsh, 2001); however, since then, a spinoff company named Anadigm has marketed these switched-capacitor FPAAs (Anadigm Company Fact Sheet, 2005). The newest Anadigm devices have Computational Analog Blocks (CABs) with two differential operational amplifiers, programmable capacitor banks, a successive approximation register (for implementing an A/D converter), and a highspeed comparator (Anadigm FPAA Family Overview, 2005). However, even the latest devices are relatively limited, with only four CABs per chip and are targeted at basic signal conditioning and filtering applications.

Switched-capacitor designs are not the only discrete time FPAAs. Switched-current circuits can be used to build a FPAA. The advantages of this technique include not requiring operational amplifiers, capability of being fabricated on standard digital CMOS processes, and elimination of distortion on the signals due to parasitic resistances; however, these designs can produce less accuracy than switched-capacitor circuits, and since the signals are all currents, a given output stage can only drive one input stage (Chang et al., 1996).

Figure 2 Most discrete time FPAAs use switched-capacitor designs. The programmability within switched-capacitor designs is usually achieved using an array of capacitors as shown. The effective capacitance at each switch can be varied by setting the *n* digital memory cells controlling switches S_1 to S_n. In essence, this amounts to a Digital-to-Analog Converter (DAC) being included in each computational block



2.2 Continuous time FPAAs

Continuous time FPAAs typically use an array of fixed components (often operational amplifiers and/or transistors) that are interconnected by a switching matrix. The switches are usually controlled by digital registers, which can be loaded by an external controller, thus allowing the FPAA to be configured to implement a number of different designs. This type of FPAA is advantageous because potential sampling artefacts are avoided; antialiasing filters are not needed; common, relatively easy design processes can be used (e.g., standard CMOS processes); and large signal bandwidths can be supported with predictable performance (Marsh, 2001). However, the switching networks introduce parasitics into

the signal path that can limit the bandwidth and add noise to the system. Some of the literature has focused on minimising the number of switches in the signal path of the FPAA (Pierzchala et al., 1995; Embabi et al., 1996; Lee and Gulak, 1991a, 1991b).

The granularity of the computational logic that forms the basis of the FPAA's design is an important design characteristic. The computational logic is usually arranged within a Computational Analog Block (CAB) on the FPAA, and then the CABs are dispersed across the FPAA with some form of an interconnect network tying them together. As summarised in Table 1, the finest grain architectures typically use transistors as the core computational cell. While these designs offer the most flexibility and generality, synthesising a sufficiently complex system requires a large number of transistors to be wired together. Thus, a large number of switches are introduced into the signal path. The switch parasitics and finite resistance increases the noise within the system and limits the performance/bandwidth (Pierzchala et al., 1995; Embabi et al., 1996; Klein, 1996). Fine grain FPAAs have primarily been relegated to research in evolvable hardware (Keymeulen et al., 2000; Santini et al., 2001; Stoica et al., 2001), where the lowest level building blocks are desirable for generating unique designs using nontraditional design methodologies. Systems that are designed using genetic algorithms are not as negatively affected by the parasitics and nonideal resistances of switches, since these parameters are taken into account and even exploited throughout the evolutionary design process.

Table 1 Summary of FPAA Granularity: The granularity of the computational logic used in an FPAA impacts the size, performance, flexibility, and functionality of the device

	Typical computational elements	*Advantages*	*Disadvantages*	*Primary applications*
Fine	Transistors	Small Simple CAB design Generic building blocks	Large no. of switches Large parasitics	Evolvable hardware
Medium	Op-amp OTA Current conveyor	Semi-generic building blocks Moderate CAB design Large variety of CAB/interconnect designs	Limited size Severe functionality/performance tradeoff	Filters Amplifiers Signal conditioning Low level Signal processing
Coarse	Fourier processor 'Expert cell'	Higher performance Easier user interface	Limited flexibility Limited functionality	Filters Signal conditioning

On the coarse grain extreme, one finds FPAAs such as IMP's EPAC™devices, which contain an 'expert cell' as the core computational block (Klein, 1996). For the IMP50E10 device, this cell is a very high level block with limited interconnects that is aimed directly at signal conditioning applications. The logic within the cell can be configured to function as an amplifier with an optional low pass filter or as a comparator with optional hysteresis. There is also a dedicated D/A converter for defining the reference point for the comparator. These coarse grain designs sacrifice flexibility and generality in favour of increased, more predictable performance (Klein, 1996).

The majority of FPAAs fall in between these two extremes. A number of FPAAs use an operational transconductance amplifier (OTA) as the basic computational element (Ray et al., 2000; Pankiewicz et al., 2001, 2002; Sanchez-Sinencio et al., 1989; Totally Re-configurable Analog Circuit – TRAC®, 1999; Pierzchala et al., 1995). OTAs work well as the basic building block because their transconductance can be programmed either by varying the analog bias voltage or by changing the gain of the output current mirrors (Adams et al., 1989; Pankiewicz et al., 2002). In addition, it has been shown that OTAs can implement a wide range of linear and nonlinear circuits. Several FPAA designs have focused on synthesising linear circuits and use OTAs to implement amplification, integration, and filtering functions (Pankiewicz et al., 2001, 2002; Pierzchala et al., 1995). Ray et al. (2000) has proposed a more generalised scheme in which linear circuits are synthesised using an OTA based lossless integrator and an OTA based lossy integrator as the basic functional blocks. He also uses an OTA based multiplier and OTA based integrator as the basic functional block for synthesising nonlinear circuits such as amplitude and frequency modulation. Sanchez-Sinencio et al. (1989) have used OTAs to implement nonlinear functions such as multiplication, division, square root, exponentiation, and piecewise linear operations.

Similarly, several FPAA designs have been proposed using a current conveyor structure as the basic building block. Current conveyors are similar to OTAs; however, when used as an amplifier they exhibit a constant bandwidth that is independent of gain. They also do not require compensation circuitry to insure stability and thus they can operate at higher frequencies (Gaudet and Gulak, 1997). Gaudet and Gulak (1997) have proposed a current conveyor based FPAA in which each CAB contains a second generation current conveyor, two programmable capacitors, and two programmable resistors (transconductors). This CAB is shown to implement amplification and first order filtering functions as well as log and antilog functions with the addition of switchable diodes. Premont

et al. (1996) also describe an FPAA based on current conveyors. Their core cell includes tuneable resistors and a current conveyor. It is demonstrated that this cell can be configured to implement a tuneable capacitor, and thus is suitable for amplification and filtering functions.

Other medium grain computational blocks have been used in FPAAs as well. Pierzchala et al. (2001) used an OTA based amplifier/integrator cell that does not require switches in the signal path. Quan et al. (1998) proposed a current mode FPAA that uses a cascode current-mode integrator as the basic building block. This core cell can implement amplification, integration, and attenuation with a minimum number of switches in the signal path (Embabi et al., 1996).

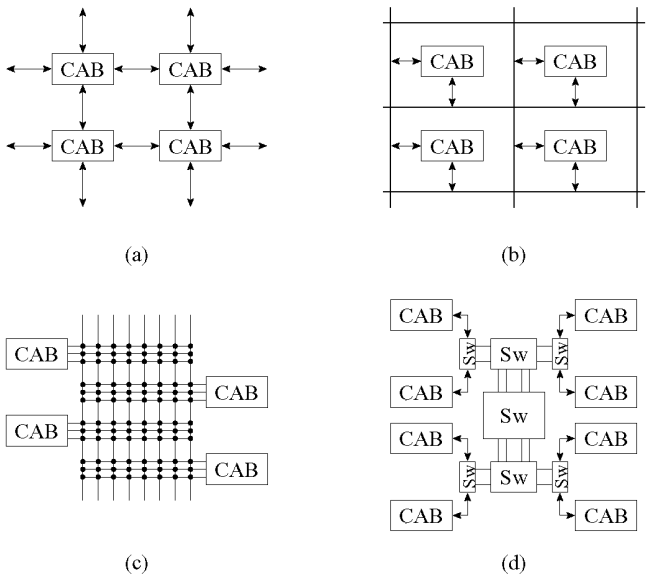
2.3 Interconnect structure

Aside from CAB components, a number of different interconnect structures have also been proposed for FPAAs. The most common method of interconnect switch is a MOS transistor controlled by a digital memory (Lee and Gulak, 1991b; Pankiewicz et al., 2002; Quan et al., 1998; Anderson et al., 1997; Chang et al., 1996; Gaudet and Gulak, 1997). Unfortunately, switch resistance in these devices can be high and can vary dramatically, based on the DC bias of the signal making them a limiting factor in designing large, complex FPAAs (Edwards et al., 2000). Lee and Gulak used this type of pass transistor switch in Lee and Gulak (1991a, 1991b); however, the parasitic effects of these switches greatly limited the performance and capability of their FPAA (Lee and Gulak, 1995). So, they replaced the pass transistors with four transistor transconductors, which increased the performance and functionality of their FPAA (Lee, 1995). The new design reduced the parasitic effects, increased the linearity, and increased the noise immunity. In addition, a transconductor switch exhibits a linear resistance, so each switch can also be used as a variable resistor by driving the gate voltage with a multivalued memory (or other internal or external signal). However, the large transistors needed for low frequency operation and the addition of a multivalued memory for each switch greatly increases the area required for the interconnects (Lee and Gulak, 1995).

Other switch designs have been proposed as well. Premont et al. (1996) used a current conveyor as the switching element. This was particularly novel, because they used the current conveyor for both the switching element and the active computational element. In an effort to provide a radiation tolerant FPAA for space applications, Edwards et al. (2000) proposed the use of metal to metal antifuses for the switches. The antifuse design they used also has the benefit of a relatively low resistance (in the 15–25 ohm range).

Besides the use of different switches, interconnect schemes are also varied in overall architecture (see Figure 3). In the Premont et al. (1996) FPAA discussed above, the use of a current conveyor for both the switching element and active computational element leads to the CABs and the switching interconnects becoming indistinguishable. The overall architecture becomes a homogeneous mesh of logic with a minimum number of switches introduced into the signal path. Various other approaches have also been tried to minimise the number of switches. Quan et al. proposed the use of local interconnects. In their architecture, each CAB can be connected to its eight neighbours and itself (Embabi et al., 1996). This would seem to be a severe limitation on the flexibility of this FPAA; however, they focus on the large number of analog circuits with mostly local interconnections (Quan et al., 1998). Pierzchala et al. (1995) tried an even more limiting architecture in which no electronic switches were included in the signal paths. While these designs may provide benefits in bandwidth and signal to noise ratio (SNR), they lack the flexibility and generality needed in a truly general purpose FPAA.

Figure 3 A number of different interconnect schemes have been used in FPAAs including (a) local connections; (b) global connections; (c) cross bar networks and (d) fat tree interconnects



In another design, Pierzchala et al. (1994) introduced an interconnect scheme with both local and global signal paths. This configuration provided local routing paths for a cell's four neighbours (north, south, east and west), as well as connections to global busses that run horizontally, vertically, and diagonally. This two tiered hierarchy increases the routing flexibility within the FPAA. An even more flexible interconnection network is the crossbar switch (Ray et al., 2000). The crossbar switch offers a nonblocking, fully connectable architecture; however, for a large number of inputs and outputs its size can be too big ($O(N^2)$ growth rate) (Lee and Gulak, 1991b). Lee and Gulak (1991a) tried to solve this problem by using an area-universal fat tree network. They used a hierarchical fat tree network of small crossbar switches where the CABs were connected as the leaves of the tree. In an additional effort to minimise the size required by the switch networks,

the number of connections was constrained (Lee and Gulak, 1991b). Unfortunately, their prototype was too small to really test this interconnect concept.

2.4 Application focus

A general purpose, commercially viable FPAA, similar to commercial FPGAs, remains elusive. Many FPAA designs have sacrificed size and generality in favour of better performance for a constrained set of circuit designs. FPAAs have been proposed for evolvable hardware (Keymeulen et al., 2000; Santini et al., 2001; Stoica et al., 2001), neural networks (Lee and Gulak, 1991a, 1991b), signal conditioning (Klein, 1996), programmable filters (Embabi et al., 1996; Quan et al., 1998), fuzzy logic (Pierzchala et al., 1994), and high frequency applications (Gaudet and Gulak, 1997). Others FPAA designs have attempted to focus on a broader class of systems including both linear and nonlinear elements (Bratt, 1998; Ray et al., 2000). However, these efforts have failed to produce a suitably generic, user friendly FPAA. In addition, all of the FPAAs to date have been very small. The number of CABs on a given device remains under 50 with many of the devices having less than ten CABs. While several companies currently sell FPAA devices, the market remains relatively small, and no single device or technology has received widespread acceptance.

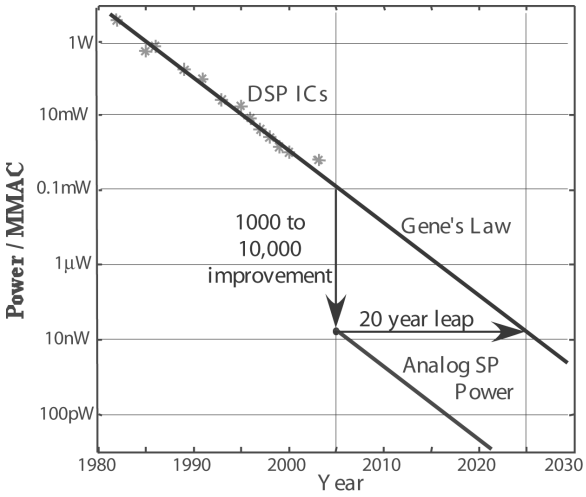
3 The analog advantage

The future of FPAAs lies in their ability to speed the implementation of advanced, low power signal processing systems. Growing demand for complex information processing on portable devices has motivated a lot of contemporary research in the design of power efficient signal processing systems. For analog systems to be desirable to the largely digital signal processing community, 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.

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 Power/MMAC, is halved about every 18 months (Frantz, 2000). 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.

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 signal processing systems has shown that power requirements can be decreased up to five orders of magnitude over typical DSP microprocessor implementations (Ellis et al., 2002; Smith et al., 2002b). As illustrated in Figure 4, this corresponds to a 20 year leap forward on the power curve predicted by Gene's Law (Hall et al., 2004b).

Figure 4 Data from Frantz (2000) showing the power consumption trends in DSP microprocessors along with data taken from a recent analog, floating-gate integrated chip



Source: Ellis et al. (2002), Hasler et al. (2002) and Smith et al. (2002b)

For analog systems to be desirable to the largely digital signal processing community 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 2, 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.

Table 2 Summary of signal processing functionality

Functionality	*DSP μP*	*Trad. analog*	*Large – scale FPAA*
Programmable	•	–	•
Monolithic filters	o	•	•
Linear scalar	•	o	•
Nonlinear scalar	o	•	•
Vector-matrix	o	o	•
Linear phase filters	•	–	o
Adaptivity	o	–	o
Tap delay lines	•	o	o

– = No or very limited support

o = Possible

• = Efficient, well suited to technology.

Recent advances in analog floating gate technologies have shown it to be a viable alternative to traditional FPAA designs (Hall et al., 2002). As shown in Figure 4, 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, and when used in the ADC, they result in more efficient biasing.

4 Floating gate technology in programmable analog circuits

Previous FPAAs have suffered from their small size and lack of functionality/generality. The next generation FPAA needs to correct these problems in order to extend the usefulness and acceptance of FPAAs. Ideally, one would like a small, easily programmable element that can be configured to act as an ideal switch, variable resistor, and configurable computational element. While such a device is indeed ideal, floating gate transistors do offer some of these qualities. Previously, we have shown that the floating gate transistor can be used as a (nonideal) switch, variable resistor, and programmable element within larger computational blocks (e.g., analog multiplier, programmable filter, programmable OTAs, etc) (Hall et al., 2002).

In addition, the small size of the floating gate structure will allow larger, more functional FPAAs to be built, using this technology. One example of the capability/area improvement that can be achieved with floating gate transistors is the programmable current mirror. Pankiewicz et al. have presented one of the most recent FPAA designs. Their FPAA is based on OTAs in which the current mirrors on the differential outputs can be programmed. They use a bank of current mirrors as shown in the simplified form in Figure 5. Each current mirror requires 64 MOS transistors, 31 digitally controlled switches, and five memory cells to hold the configuration of the switches. The entire structure can be replaced with two programmable floating gate transistors. The area savings in this case are quite dramatic. Furthermore, the resolution of the bank of current mirrors is set at five bits; whereas the floating gate current mirror's resolution can be varied based on the need of a given application with a maximum resolution of approximately ten bits (Sarpeshkar, 1997).

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) (Hasler et al., 1999). Figure 6 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 tunnelling to increase the charge on the floating gate node, the current flow through the pFET channel can be accurately controlled (Hasler et al., 1999; Kucic et al., 2001a).

Figure 5 Layout, cross section, and circuit diagram of the floating gate pFET in a standard double-poly, n-well MOSIS process: The pFET transistor is the standard pFET transistor in the n-well process. The gate input capacitively couples to the floating gate by a capacitor. Between V_{tun} and the floating gate is our symbol for a tunneling junction – a capacitor with an added arrow designating the charge flow

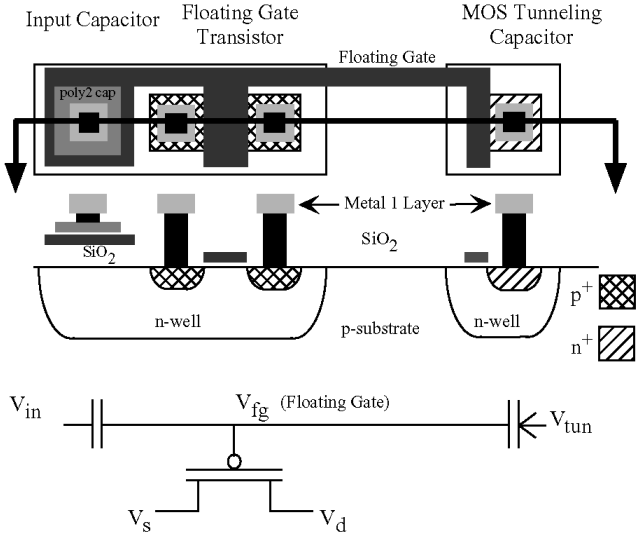
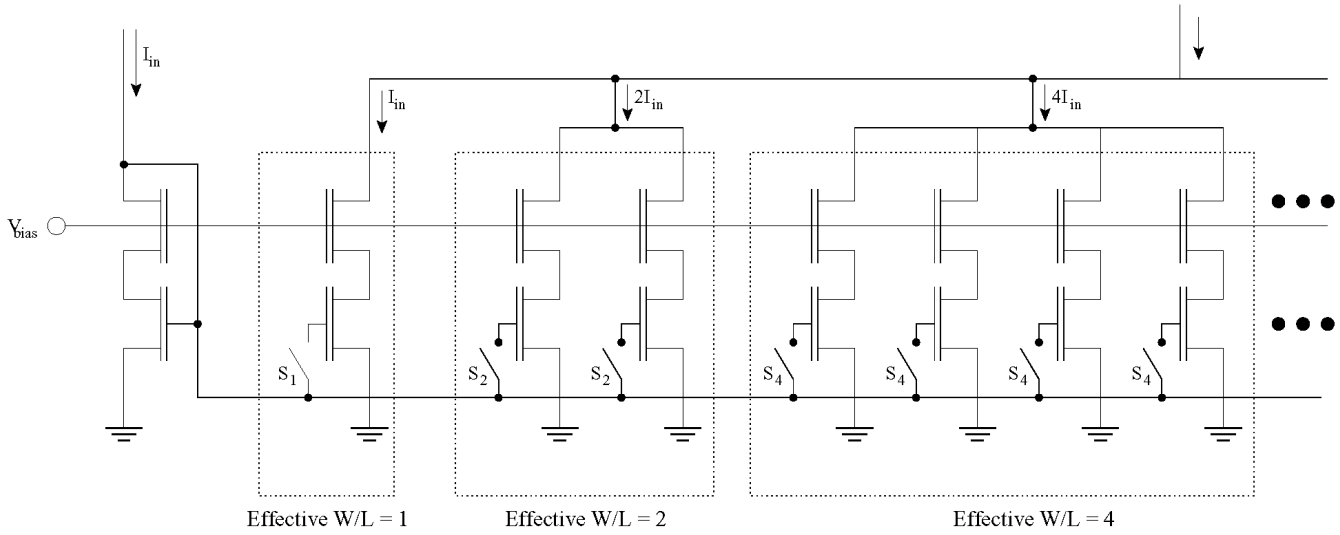


Figure 6 Standard designs often achieve circuit programmability by embedding switchable arrays of elements (such as transistors or capacitors) within the logic cells. Here, a conceptual version of Pankiewicz et al. (2002)'s programmable current mirror is shown. In their case, 5 bits were used to set the switches. This requires 64 MOS transistors, 31 digitally controlled switches, and five memory cells to hold the configuration of the switches. Using floating gate technology, this entire structure can be replaced with two programmable floating gate transistors



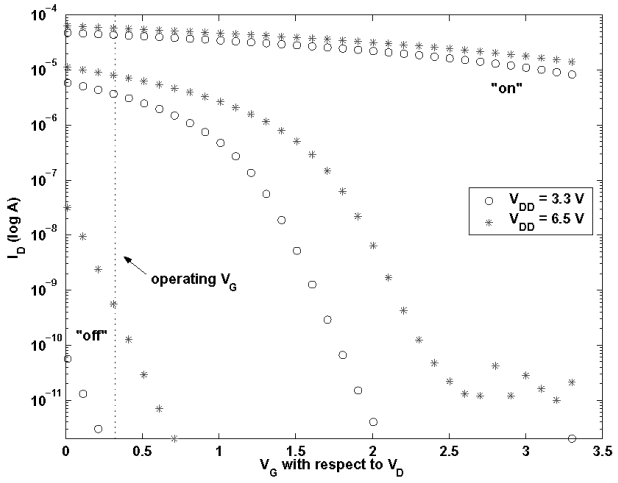
4.1 Floating gate switches

Using a floating gate transistor as a switch requires that the device be turned 'on' or turned 'off'. Ideally, the 'on' state corresponds to the free flow of current through the device or equivalently, zero impedance between the source and the drain. Likewise, the 'off' state is characterised by zero current flowing through the device – an infinite impedance between the source and the drain nodes. A floating gate transistor, however, does not act as a perfect switch. The 'on' state is characterised by an impedance greater than zero, and the 'off' state has an impedance less than infinity. Therefore, the quality of a floating gate transistor as a switch is determined by measuring the 'on' and 'off' impedances.

The floating gate switch network has been characterised in Hall et al. (2004b). 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 Figure 7, the floating gate switch can be programmed in between the 'on' and 'off' extremes.

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 7. Measuring the currents with V_{DD} = 65 V, allows the I–V curves to be visible to the programming infrastructure 1–V below the point visible when V_{DD} = 3.3 V.

Figure 7 Floating gate switches can be programmed within a wide range. Here, examples of an 'on', 'off', and midposition device are shown. To extend the effective programming range of the device, large currents are measured with V_{DD} = 3.3 V and small currents are measured with V_{DD} = 6.5 V during programming



In the operating mode of this FPAA, the voltage on the gate capacitor for all switches is the same. From Figure 7, 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.

4.2 Switch as computational element

When used as a switch, the floating gate should be as transparent a part of the circuit as possible. However, Figure 7 shows that the floating gate transistor can be used

as an in-circuit element (Kucic et al., 2001b; Hasler and Minch, 2002). 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 linear resistor can be synthesised 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 not losing functionality.

4.3 Floating gate transistors within computational logic

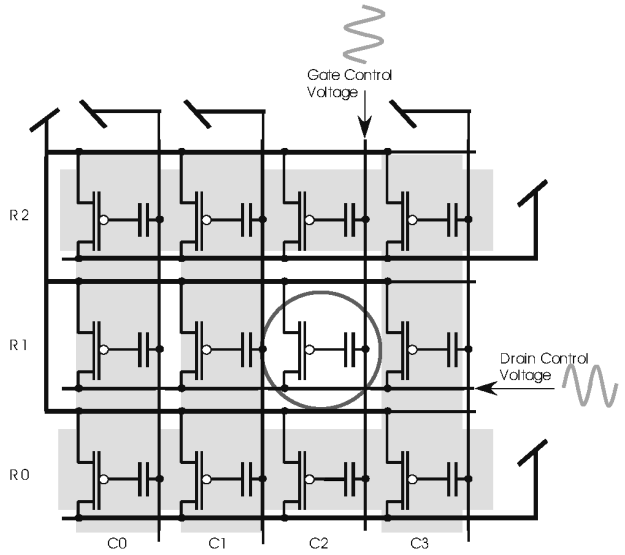
Current FPAA designs rely on switches as the primary or sole programmable element on the chip. Biases, multiplier coefficients, resistances, and similar elements are set via offchip components or with low resolution capacitor banks or current mirror banks. Thus, the ability to modify or program the actual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

In the FPAA explored here, floating gate transistors are used within the computational analog blocks (CABs) to set bias voltages for the OTAs (see Figure ll(a)), adjust the corner frequencies on the capacitively coupled current conveyors (C^4s), and set multiplier coefficients in the vector-matrix multipliers. In this manner, the floating gate transistors allow the characteristics of the computational elements to be programmed onchip 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.

4.4 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 (see Figure 8) (Kucic et al., 2001a).

Figure 8 By selectively setting the gate and drain voltages of the columns and rows, a single floating gate transistor can be programmed using hot electron injection without affecting the neighbouring devices

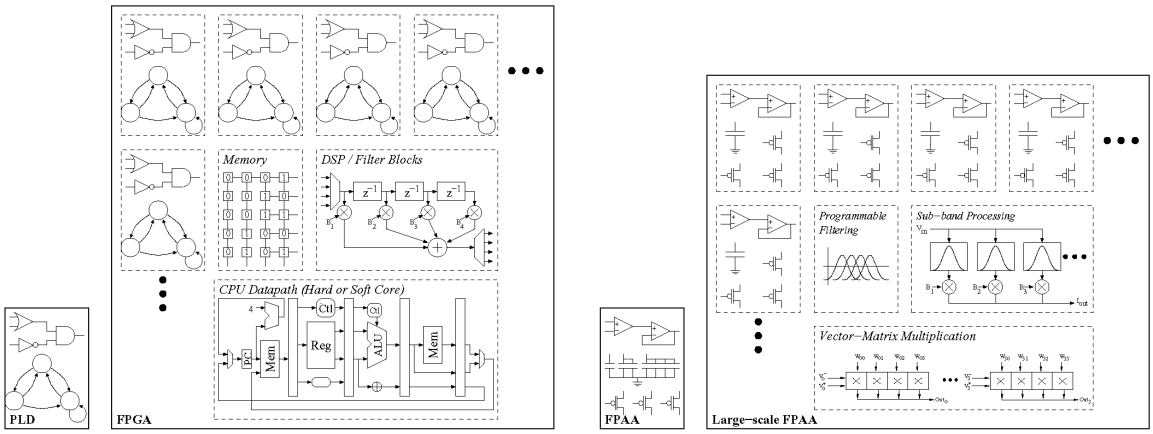


Part of the previous work has been the development of a systematic method for programming arrays of floating gate transistors (Kucic et al., 2001a, 2001b; Smith et al., 2002a). A microprocessor based board has been built to interface a PC to these analog floating gate arrays for the purposes of programming and testing. With a PC controlling the programming of these devices, the details of using hot electron injection and tunnelling to program individual floating gate switches have been abstracted away from the enduser. The programming algorithms have been optimised for accuracy and speed, while giving the enduser an easy to use interface for configuring arrays of floating gate devices.

5 Large-scale FPAAs

As shown in Figure 9, 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 in that they 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).

Figure 9 (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 and (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 the 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

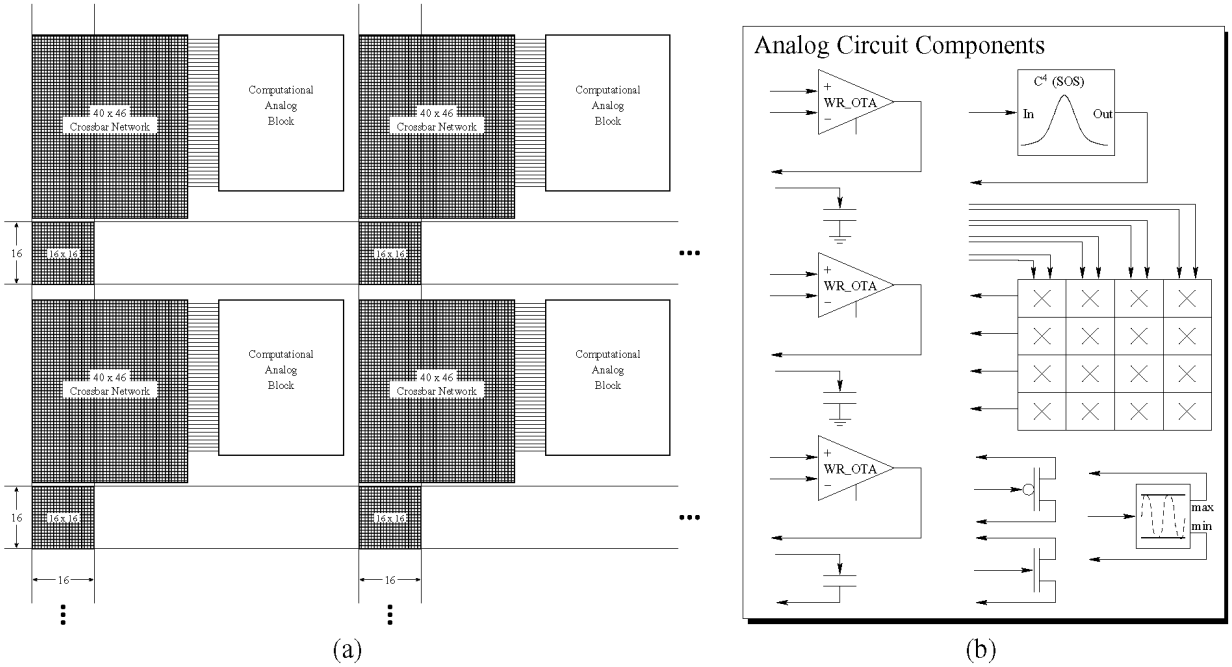


(a) (b)

Large-scale FPAAs are possible using floating gate technology. In the previous section, floating gate transistors have been shown to be viable analog programmable elements within the switch network and the computational logic. This enhances the functionality and flexibility of the FPAA while simplifying the programming infrastructure and creating a very compact, scalable architecture.

In our large-scale FPAAs, the computational logic is organised 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 global busses and local interconnects inbetween as shown in Figure 10(a). Early designs typically have 50–100 CABs on a single chip and often consume 9–36 mm^2 in the TSMC 0.35 micron CMOS process as the size of the switch networks and number and complexity of the CABs is varied.

Figure 10 (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 and (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 widerange 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



(a) (b)

5.1 Computational analog blocks

Many example CABs can be imagined using this technology. Figure 10(b) shows one example CAB, whose functionality is enhanced by a mixture of medium and coarse grain computational blocks similar to many modern FPGA designs. The computational blocks were carefully selected to provide a sufficiently flexible, generic architecture while optimising 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 nonlinear functions (Ray et al., 2000; Pankiewicz et al., 2002; Sanchez-Sinencio et al., 1989; Totally Re-configurable Analog Circuit – TRAC Datasheet, 1999). 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 minimise 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 (Hall et al., 2004b; Pankiewicz et al., 2002). 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 bandpass filter module and the 4×4 vector-matrix multiplier block. In general, the C^4 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 (Hasler et al., 2001; Kucic et al., 2001a),. In addition, a peak detector is added to each CAB.

5.2 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 64×16 floating gate crossbar switch network connecting them (Hall et al., 2002). The CAB design was slightly smaller than the one outlined in Section 3 having a C^4 bandpass filter module, 4×4 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 characterise 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 11 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 Figure 12. 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 minimised 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.

Figure 11 (a) The source-follower is configured by programming the floating gate charge on the floating gate device. (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 and (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

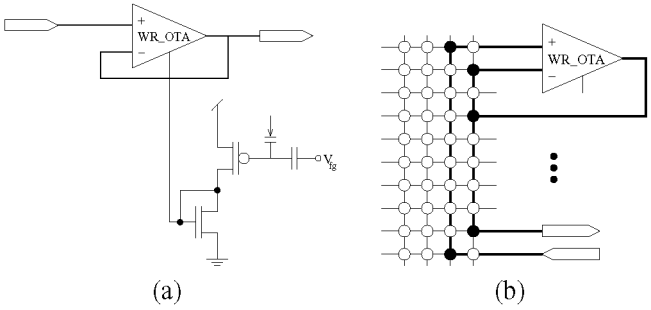
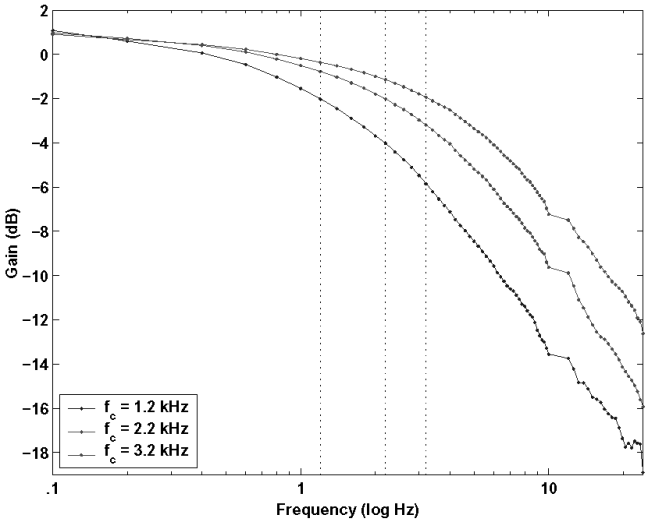


Figure 12 The frequency response of the source-follower circuit is shown for several bias currents. An internal floating gate transistor is used as a current source to set the OTA's bias and thus the bandwidth of this first order filter



In Figure 13, 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 programmable biases, the two OTAs in a source-follower configuration were biased to the same level and the third OTA's bias current was increased to adjust the *Q* peak of the system. The frequency response for this circuit is shown in Figure 14. As expected, the *Q* peak increases as the bias current (e.g., conductance) increases.

Figure 13 (a) a second order section filter can be implemented with two OTAs in a source-follower configuration and a third OTA that creates positive feedback and (b) using the switch matrix, three OTAs within the CABs are connected in a second order section configuration

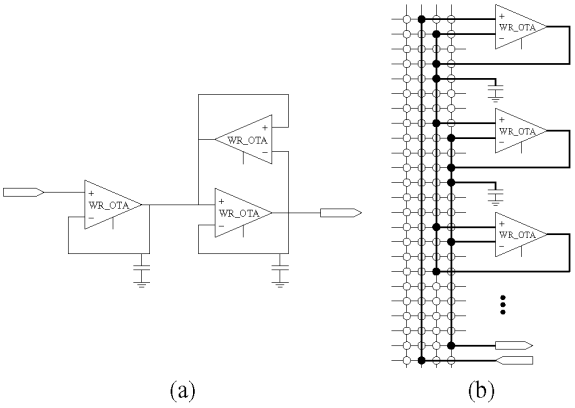
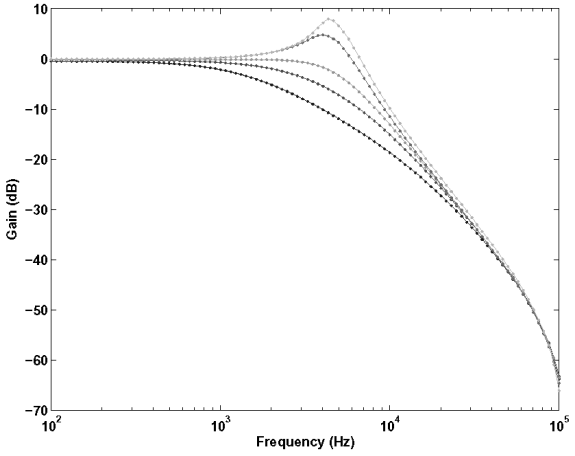


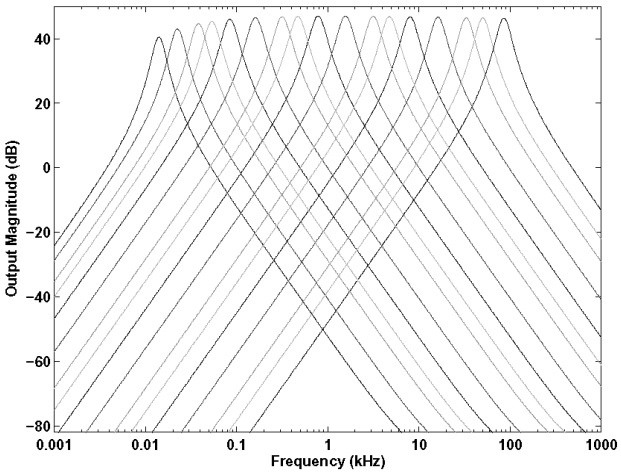
Figure 14 The experimental frequency response of a second order section filter is shown here. The *Q* parameter is adjusted by increasing the bias current of the positive feedback amplifier via a floating gate current source.



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 Figure 7 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 Figure 11(a) 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 preferable 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 conditioning (modulation, mixing, etc), transform coding, and neural networks (with external training). Most of these systems rely on efficient subband processing; so, each CAB has been designed with a C^4 bandpass to optimise this operation. As shown in Figure 15, the centre frequency of the C^4 filter can be moved over a large range of frequencies.

Figure 15 Frequency decomposition (subband processing) can be achieved on the test bed FPAA by using the C^4 bandpass filter block in each CAB. In this simulation of the FPAA, the centre frequency of the C^4 is shown to be programmable over a wide range of frequencies



6 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 initial results have been shown.

References

Adams, W.J., Nedungadi, A. and Geiger, R.L. (1989) 'Design of a programmable OTA with multi-decade transconductance adjustment', *Proceedings of the International Symposium on Circuits and Systems*, Vol. 1, May, pp.663–666.

Anadigm Company Fact Sheet (2005) HTML Page, Anadigm, http://www.anadigm.com/Prs_15.asp/, September, p.1.

Anadigm FPAA Family Overview (2005) PDF File, Anadigm, http://www.anadigm.com/Supp_06.asp?tab=lit, September.

Anderson, D., Marcjan, C., Bersch, D., Anderson, H., Hu, P., Palusinski, O., Gettman, D., Macbeth, I. and Bratt, A. (1997) 'A field programmable analog array and its application', *Proc. IEEE Custom Integrated Circuits Conference*, May, pp.555–558.

Birkner, J.M. and Chua, H-T. (1978) *Programmable Array Logic Circuit*, US Patent No. 4, pp.1–24.

Bratt, A. (1998) 'Motorola field programmable analogue arrays, present hardware and future trends', *IEE Half – day Colloquium on Evolvable Hardware Systems*, March, pp.1/1–1/5.

Chang, S., Hayes-Gill, B. and Paul, C. (1996) 'Multi-function block for a switched current field programmable analog array', *1996 Midwest Symposium on Circuits and Systems*, August, pp.158–161.

Chow, P., Seo, S.O., Rose, J., Chung, K., Paez-Monzon, G. and Rahardja, I. (1999) 'The design of an SRAM-based field-programmable gate array – part I: architecture', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 7, No. 2, June, pp.191–197.

Edwards, R.T., Strohbehn, K. and Jaskulek, S.E. (2000) 'A field-programmable mixed-signal array architecture using antifuse interconnects', *Proceedings of the International Symposium on Circuits and Systems*, Vol. 3, May, pp.319–322.

Ellis, R., Yoo, H., Graham, D., Hasler, P. and Anderson, D. (2002) 'A continuous-time speech enhancement from-end for microphone inputs', *Proceedings of the IEEE International Symposium on Circuits and Systems*, Phoenix, AZ, Vol. 2, pp.II.728–II.731.

Embabi, S., Quan, X., Oki, N., Manjrekar, A. and Sanchez-Sinencio, E. (1996) 'A field programmable analog signal processing array', *IEEE 39th Midwest Symposium on Circuits and Systems*, Vol. 1, August, pp.151–154.

Frantz, G. (2000) 'Digital signal processor trends', *IEEE Micro*, Vol. 20, No. 6, November–December, pp.52–59.

Ganesan, S. and Vemuri, R. (2001) 'Behavioral partitioning in the synthesis of mixed analog-digital systems', *Proc. Design Automation Conference*, June, pp.133–138.

Gaudet, V.C. and Gulak, P.G. (1997) 'CMOS implementation of a current conveyor-based field-programmable analog array', *Conference Record of the 31st Asilomar Conference on Signals, Systems and Computers*, Vol. 2, November, pp.1156–1159.

Gulak, P.G. (1995) 'Field-programmable analog arrays: past, present and future perspectives', *IEEE Region 10th International Conference on Microelectronics and VLSI*, November, pp.123–126.

Hall, T.S., Hasler, P. and Anderson, D.V. (2002) 'Field-programmable analog arrays: a floating-gate approach', *Proc. 12th International Conference on Field Programmable Logic and Applications*, Montpellier, France, September, pp.424–433.

Hall, T.S., Twigg, C.M., Hasler, P. and Anderson, D.V. (2004a) 'Developing large-scale field-programmable analog arrays', *Proc. 18th International Parallel and Distributed Processing Symposium*, Santa Fe, New Mexico, April, p.6.

Hall, T.S., Twigg, C.M., Hasler, P. and Anderson, D.V. (2004b) 'Application performance of elements in a floating-gate FPAA', *Proceedings of the International Symposium on Circuits and Systems*, May, pp.II.589–II.592.

Hasler, P. and Minch, B.A. (2002) *Floating-gate Devices, Circuits, and Systems*, in press.

Hasler, P., Minch, B.A. and Diorio, C. (1999) 'Adaptive circuits using PFET floating-gate devices', *Proceedings of the 20th Anniversary Conference on Advanced Research in VLSI*, Atlanta, GA, March, pp.215–229.

Hasler, P., Minch, B.A. and Diorio, C. (2001) 'An autozeroing floating-gate amplifier', *IEEE Transactions on Circuits and Systems II*, Vol. 48, No. 1, January, pp.74–82.

Hasler, P., Smith, P., Ellis, R., Graham, D. and Anderson, D.V. (2002) 'Biologically inspired auditory sensing system interfaces on a chip', *2002 IEEE Sensors Conference*, Orlando, FL, June, pp.669–674, invited Paper.

ispPAC Overview (2001) PDF File, *Lattice Semiconductor Corporation*, http://www.latticesemi.com/, August, pp.1–3.

Keymeulen, D., Zebulum, R.S., Jin, Y. and Stoica, A. (2000) 'Fault-tolerant evolvable hardware using field-programmable transistor arrays', *IEEE Transactions on Reliability*, Vol. 49, No. 3, September, pp.305–316.

Klein, H.W. (1996) 'The EPAC architecture: an expert cell approach to field programmable analog circuits', *IEEE 39th Midwest Symposium on Circuits and Systems*, Vol. 1, August, pp.169–172.

Koneru, S., Lee, E.K.F. and Chu, C. (1999) 'A flexible 2-d switched-capacitor FPAA architecture and its mapping algorithm', *42nd Midwest Symposium on Circuits and Systems*, Vol. 1, August, pp.296–299.

Kucic, M., Low, A., Hasler, P. and Neff, J. (2001a) 'A programmable continuous-time floating-gate fourier processor', *IEEE Transactions on Circuits and Systems II*, Vol. 48, No. 1, January, pp.90–99.

Kucic, M., Hasler, P., Dugger, J. and Anderson, D.V. (2001b) 'Programmable and adaptive analog filters using arrays of floating-gate circuits', in Brunvand, E. and Myers, C. (Eds.): *2001 Conference on Advanced Research in VLSI*, IEEE Computer Society, March, pp.148–162.

Kutuk, H. and Kang, S-M. (1996) 'A field-programmable analog array (FPAA) using switched-capacitor techniques', *Proceedings of the International Symposium on Circuits and Systems*, Vol. 4, May, pp.41–44.

Lee, E.K.F. (1995) *Field-programmable Analog Arrays on MOS Transconductors*, PhD Dissertation, University of Toronto, p.182.

Lee, K. and Gulak, P. (1991a) 'A CMOS field-programmable analog array', *IEEE International Solid – State Conference Digest of Technical Papers*, February, pp.186–188.

Lee, E.K.F. and Gulak, P.G. (1991b) 'A CMOS field-programmable analog array', *IEEE Journal of Solid-State Circuits*, Vol. 26, No. 12, February, pp.1860–1867.

Lee, K. and Gulak, P. (1995) 'A transconductor-based field-programmable analog array', *IEEE International Solid – State Conference Digest of Technical Papers*, February, pp.198–199.

Looby, C.A. and Lyden, C. (1997) 'A CMOS continuous-time field programmable analog array', *Proc. 5th International ACM Symposium on Field – Programmable Gate Arrays*, ACM Press, pp.137–141.

Marsh, D. (2001) 'Programmable analogue ICs challenge spice-and-breadboard designs', *EDN Europe*, http:// www.ednmag.com: Reed Business Information, October, pp.30–36.

Pankiewicz, A., Wojcikowski, M., Szczepanski, S. and Sun, Y. (2001) 'A CMOS field programmable analog array and its application in continuous-time OTA-C filter design', *Proceedings of the International Symposium on Circuits and Systems*, Vol. 1, May, pp.5–8.

Pankiewicz, B., Wojcikowski, M., Szczepanski, S. and Sun, Y. (2002) 'A field programmable analog array for cmos continuous-time OTA-C filter applications', *IEEE Journal of Solid-State Circuits*, Vol. 37, No. 2, February, pp.125–136.

Pierzchala, E., Perkowski, M.A. and Grygiel, S. (1994) 'A field programmable analog array for continuous, fuzzy, and multi-valued logic applications', *24th International Symposium on Multiple – Valued Logic*, May, pp.148–155.

Pierzchala, E., Perkowski, M.A., Halen, P.V. and Schaumann, R. (1995) 'Current-mode amplifier/integrator for a field-programmable analog array', *IEEE International Solid–State Conference Digest of Technical Papers*, February, pp.196–197.

Premont, C., Grisel, R., Abouchi, N. and Chante, J-P. (1996) 'Current-conveyor based field programmable analog array', *IEEE 39th Midwest Symposium on Circuits and Systems*, Vol. 1, August, pp.155–157.

Quan, X., Embabi, S. and Sanchez-Sinencio, E. (1998) 'A current-mode based field programmable analog array architecture for signal processing applications', *IEEE 1998 Custom Integrated Circuits Conference*, Santa Clara, CA, May, pp.277–280.

Ray, A., Chaudhuri, P.P. and Nandi, P.K. (2000) 'Design of OTA based field programmable analog array', *Proc. 13th International Conference on VLSI Design*, January, pp.494–498.

Sanchez-Sinencio, E., Ramirez-Angulo, J., Linares-Barranco, B. and Rodriguez-Vazquez, A. (1989) 'Ota-based non-linear function approximations', *Proceedings of the International Symposium on Circuits and Systems*, Vol. 1, May, pp.96–99.

Santini, C.C., Zebulum, R., Pacheco, M.A.C., Vellasco, M.M.R. and Szwarcman, M.H. (2001) 'Evolution of analog circuits on a programmable analog multiplexer array', *Proc. IEEE Aerospace Conference*, Vol. 5, March, pp.2301–2308.

Sarpeshkar, R. (1997) *Efficient Precise Computation with Noisy Components: Extrapolating from an Electronic Cochlea to the Brain*, PhD Thesis, California Institute of Technology, Pasadena, CA, p.234.

Sivilotti, M.A. (1991) *Wiring Considerations in Analog VLSI Systems, with Application to Field-programmable Networks (VLSI)*, PhD Dissertation, California Institute of Technology, Pasadena, CA, p.210.

Smith, P., Kucic, M. and Hasler, P. (2002a) 'Accurate programming of analog floating-gate arrays', *Proceedings of the International Symposium on Circuits and Systems*, Phoenix, AZ, May, Vol. 5, pp.V.489–V.492.

Smith, P.D., Kucic, M., Ellis, R., Hasler, P. and Anderson, D.V. (2002b) 'Mel–frequency cepstrum encoding in analog floating–gate circuitry', in *Proceedings of the International Symposium on Circuits and Systems*, Phoenix, AZ, May, Vol. 4, pp.IV671–IV674.

Stoica, A., Zebulum, R., Keymeulen, D., Tawel, R., Daud, T. and Thakoor, A. (2001) 'Reconfigurable VLSI architectures for evolvable hardware: from experimental field programmable transistor arrays to evolution-oriented chips', *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, Vol. 9, No. 1, February, pp.227–232.

Totally Re-configurable Analog Circuit – TRAC Datasheet (1999) PDF File, Zetex Semiconductors, http://www.zetex.com, March.

Wakerly, J.F. (1999) *Digital Design: Principles and Practices*, 3rd ed., Chapter 5, Combinational logic design practices, New Jersey, Prentice-Hall, pp.311–455.