FLOATING GATE BASED LARGE-SCALE FIELD-PROGRAMMABLE ANALOG ARRAYS FOR ANALOG SIGNAL PROCESSING

A Dissertation
Presented to
The Academic Faculty

By

Christopher M. Twigg

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy
in
Electrical and Computer Engineering

School of Electrical and Computer Engineering
Georgia Institute of Technology
August 2006

Copyright © 2006 by Christopher M. Twigg

FLOATING GATE BASED LARGE-SCALE FIELD-PROGRAMMABLE ANALOG ARRAYS FOR ANALOG SIGNAL PROCESSING

Approved by:

Dr. Paul E. Hasler, Advisor
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Dr. Aaron D. Lanterman
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Dr. David V. Anderson
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Dr. Mark T. Smith
Professor, School of ICT
Swedish Royal Institute of Technology
Kista, Sweden

Dr. John B. Peatman
Professor, School of ECE
Georgia Institute of Technology
Atlanta, GA

Date Approved: June 2006

ACKNOWLEDGMENTS

I would like to thank everyone who helped me along the way from my advisor, Paul Hasler, to my colleagues within the CADSP research group. I would also like to thank my friends and family who encouraged me throughout the process. However, my greatest thanks goes to my ever loving wife, Shannon, who has endured many things throughout my graduate life that I have no doubt caused. I could never have managed through these last few years without her love, support, and most of all, patience.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Large-scale reconfigurable and programmable analog devices provide a new option for prototyping and synthesizing analog circuits for analog signal processing and beyond. Field-programmable analog arrays (FPAAs) built upon floating gate transistor technologies provide the analog reconfigurability and programmability density required for large-scale devices on a single integrated circuit (IC). A wide variety of synthesized circuits, such as OTA followers, band-pass filters, and capacitively coupled summation/difference circuits, were measured to demonstrate the flexibility of FPAAs. Three generations of devices were designed and tested to verify the viability of such floating gate based large-scale FPAAs. Various architectures and circuit topologies were also designed and tested to explore the trade-offs present in reconfigurable analog systems. In addition, large-scale FPAAs have been incorporated into class laboratory exercises, which provide students with a much broader range of circuit and IC design experiences than have been previously possible. By combining reconfigurable analog technologies with an equivalent large-scale digital device, such as a field-programmable gate array (FPGA), an extremely powerful and flexible mixed signal development system can be produced that will enable all of the benefits possible through cooperative analog/digital signal processing (CADSP).

CHAPTER 1

RECONFIGURABLE AND PROGRAMMABLE ANALOG

With an ever increasing demand to bring new portable devices to market quickly, it would be extremely advantageous to have a reconfigurable and programmable prototyping platform with which to test new ideas. The ideal device would be mixed signal and allow any combination of analog and digital circuitry to be synthesized. With such a device, any analog or digital component could be interfaced with any other analog or digital component. It is not hard to imagine the digital part of this mixed-signal device, since FPGAs have been used to synthesize very complex systems for many years. However, very little is known about large-scale reconfigurable and programmable analog technologies. To build the mixed signal platform of this example, more research needs to be performed on reconfigurable and programmable analog devices. This dissertation attempts to address this research topic through the exploration of field-programmable analog arrays (FPAAs).

1.1 Analog Processing, the Past and the Future

It is hard to imagine in this digital world that there was once a time when everything was analog. Real world signals are analog, so it was only natural to have analog processes controlling these systems. However, digital controllers soon proved to be much easier and quicker to develop new technologies and products. Digital systems quickly replaced their analog predecessors, even though analog systems could perform significantly better in some applications. Compared to digital signal processors (DSPs), analog equivalent circuits could provide significant power savings, as illustrated in Figure 1.1. If the DSP power consumption trend continues as projected, the power savings could be equivalent to a 20 year leap in digital technology. In reality, DSP power consumption is reducing at a significantly lower rate than projected, which only enhances analog's advantage over digital.

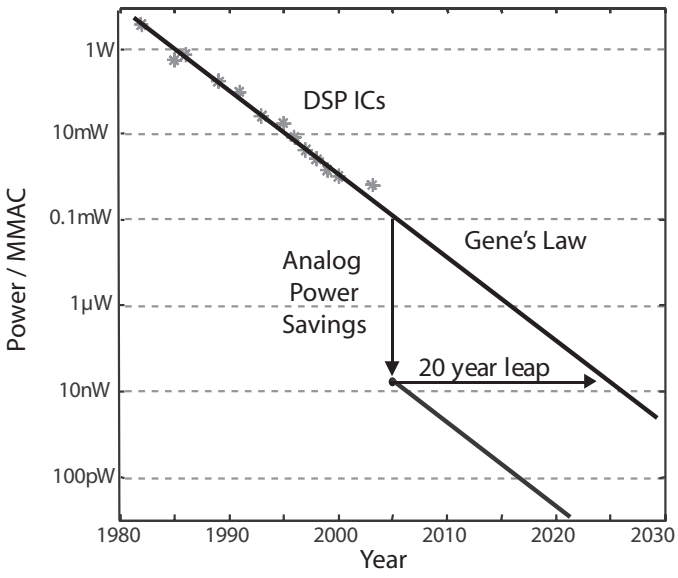


Figure 1.1. DSP power consumption trend compared to power efficient analog equivalent functions [1–5].

Figure 1.2a illustrates how DSPs interact with real world signals. In most of these systems the analog input signal is immediately digitized using an analog to digital converter (ADC). The digital signal is then processed and converted back into the analog domain via a digital to analog converter (DAC). This process has made many current gadgets possible, but it does not take advantage of the benefits possible with analog processing. Figure 1.2b shows an improved system in which the real world signals interface directly to an analog signal processor (ASP). Instead of directly converting to a digital signal, initial processing

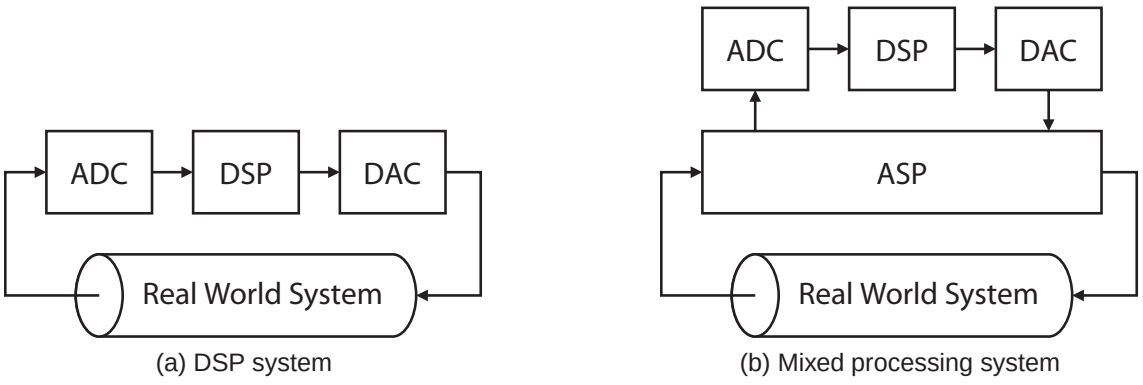
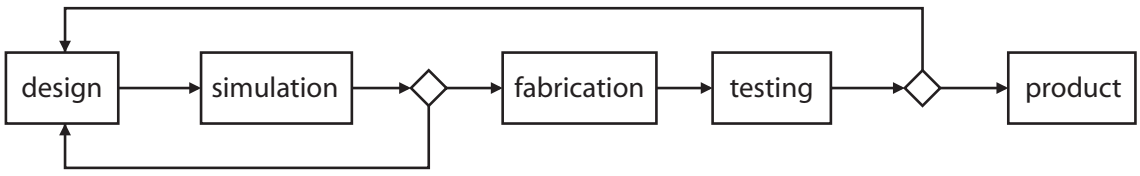


Figure 1.2. Signal processing in the real world.
(a) DSP systems require costly data converters to interact with real world signals.
(b) Mixed signal systems leverage analog and digital processing to increase efficiency.

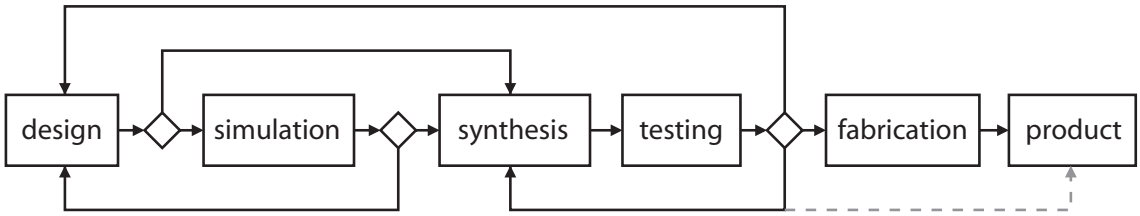
can begin in analog. In some instances the signal may pass only through the ASP and avoid the DSP and data converters altogether. In other cases where the processing would be better suited for digital algorithms, the ASP can preprocess the data before directing the signal to the DSP via the ADC. After passing through the DSP and the DAC, the ASP can then postprocess the signal before it returns to the real world. Although the benefits of a mixed signal system are clear, the difficulty of designing and using analog processing circuits could continue to prevent their widespread acceptance.

1.2 The FPAA Advantage

Figure 1.3a depicts the traditional analog design flow used for custom analog IC development. Typically the design phase will lead to a simulation stage that verifies circuit performance. If the circuit fails to achieve the required specifications in simulation, this process may iterate through design and simulation multiple times. Once a design passes the simulation phase, it progresses through the fabrication and testing steps. Although every attempt is made to perfect the design in earlier stages, it is likely that the fabrication and testing stages will require two or more iterations to obtain a product with the desired performance. Since a single fabrication cycle can requires two or more months to complete,



(a) Custom analog IC design flow



(b) FPAA design flow

Figure 1.3. Comparison of custom analog IC and FPAA design flows.
(a) The traditional analog design flow includes time consuming iterative fabrication loops.
(b) An FPAA flow iterates significantly faster using a synthesis step instead of fabrication.

the final product may require many months or years to develop [6].

The use of a reconfigurable and programmable analog device, such as an FPAA, can significantly reduce the time required for this design cycle. As Figure 1.3b depicts, an FPAA introduces a synthesis phase that allows physical hardware to be generated and tested before fabrication. By taking fabrication out of the iterative loop, a new design or algorithm can be verified in a matter of hours or days. Additionally, a configured FPAA could be used as the final product, thereby skipping the fabrication step, much like FPGAs are sometimes used today.

An FPAA with significant functionality would not only reduce analog system design time, it could also make analog systems easier to use. Proper design tools would abstract most of the actual analog hardware design, much as FPGAs have done for digital systems. This would make it possible for DSP engineers with little or no analog design expertise to quickly synthesize and test analog replacements for traditional digital computation.

1.3 General FPAA Architecture

An FPAA can be generally depicted as in Figure 1.4. Reconfigurability is commonly achieved through some interconnect network, which can be implemented by any number of switch topologies. This network connects the various analog components together to form the desired circuit. Programmability is usually implemented using DACs or ratios of standard components [7], such as transistors, capacitors, and resistors. Figure 1.5a

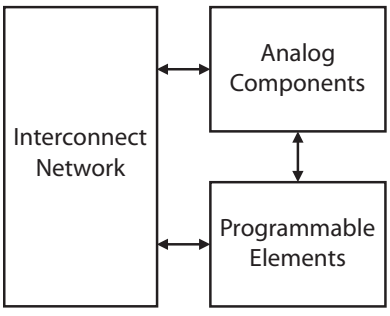
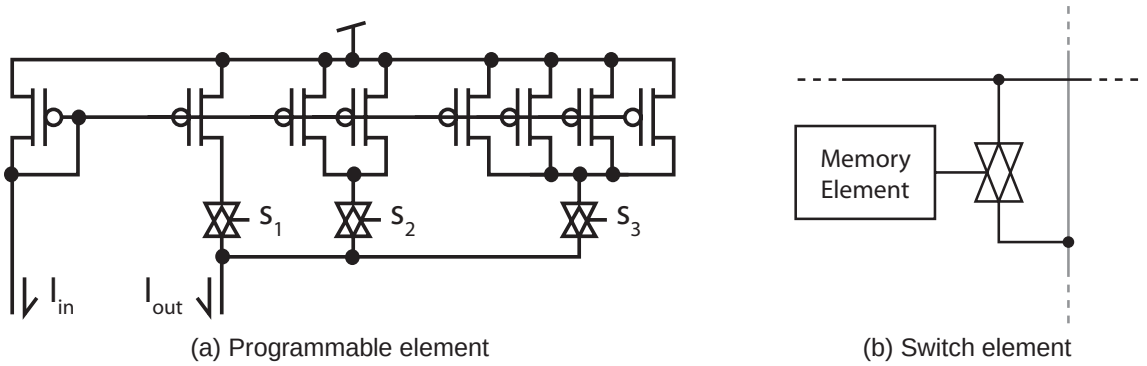


Figure 1.4. Generic FPAA architecture.



(a) Programmable element (b) Switch element

Figure 1.5. Example FPAA switch and programmable element.
(a) Programmability is commonly implemented using arrays of selectable components.
(b) Memory controlled transmission gates are typically used as switches.

demonstrates how an array of binary weighted transistors could be used to implement a current mode multiplier. Programmability implemented in this manner requires a significant amount of area, since the device area approximately doubles for each bit of resolution. This area requirement limits the number of programmable elements that can be integrated on a single chip and therefore reduces the size of the system that can be synthesized, which is the case for many previous FPAAs [8–13].

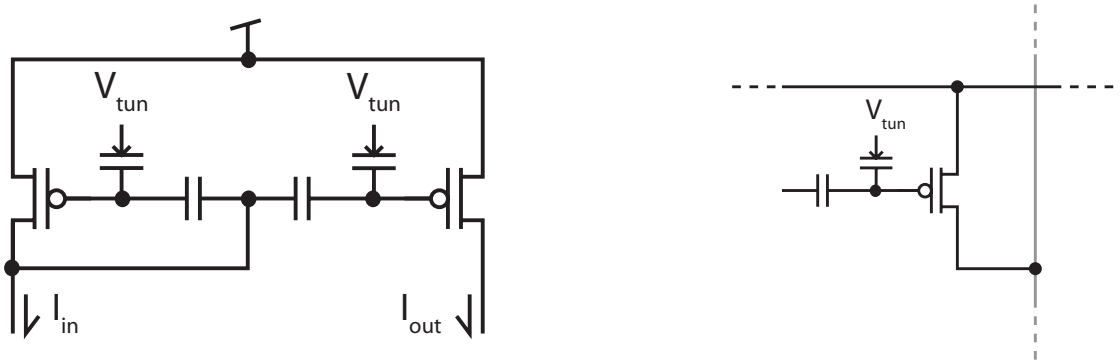
The switches in these devices are commonly transmission gates, each of which is controlled by a memory element in the form of an SRAM or EEPROM cell, as seen in Figure 1.5b. Although these switches allow for great design flexibility, they also contribute a significant amount of parasitic resistance and capacitance. To eliminate these parasitics, various modifications to the general FPAA architecture have been attempted. Fuse based FPAAs attempt to reduce these parasitics by creating or destroying metal wire connections between components. Since these fuses are made of metal instead of a transistor, both the parasitic capacitance and resistance are reduced, but they are generally one-time programmable. Some FPAAs are even built without any switches. These architectures generally have a limited variety of analog components that interconnect directly to each other, such as the hexagonally connected transconductance (G_M) elements of Becker and Manoli's FPAA [9]. In these devices, reconfigurability is achieved through programming.

Since the outputs of G_M stages with no bias current effectively float, connections are made in these G_M elements by digitally controlling the biases.

1.4 Large-Scale FPAAs

Current FPAA offerings are rather small and cannot handle large analog systems on a single IC. These devices are similar in size and complexity to the early digital programmable logic devices (PLDs) rather than modern FPGAs. However, larger or denser FPAAs will be necessary to push reconfigurable and programmable analog technologies into widespread use. The large-scale FPAAs discussed in this work are possible because of the incorporation of floating-gate pFETs as the programmable element.

Instead of area consuming arrays of ratioed devices, a single transistor can be programmed to accurate analog values. This allows the large-scale FPAAs discussed in this work to be significantly denser than previous FPAAs, which makes them capable of synthesizing much larger systems. Figure 1.6a shows a floating gate transistor version of the current mode multiplier example of Figure 1.5a. In addition to programmable parameters, floating-gate transistors can be used as non-volatile switching elements in these FPAAs. By using floating-gate transistors as the switch, the switch and memory elements have been effectively combined in the form of a single transistor, as depicted in Figure 1.6b.



(a) Programmable element (b) Switch element

Figure 1.6. Example FPAA switch and programmable element using floating gate transistors.
(a) Programmability is implemented through charge storage on the floating gate.
(b) A floating gate transistor combines the switch and memory element.

The density of FPAA devices can also be enhanced by correcting offsets and mismatches using floating gate transistors [14–17]. Analog design techniques traditionally deal with offsets and mismatch by increasing device area and redundant layout schemes. These techniques work fairly well, but they consume a large amount of die area. Laser trimming is also commonly used for high accuracy circuits, but this requires costly post fabrication processing. However, floating gate transistors can be used within various circuit topologies and electronically programmed to trim offsets. Since this can be done post fabrication in a cheap manner, a significant area savings can be achieved in the FPAA analog circuit designs in addition to the programmable elements and switches.

CHAPTER 2

FLOATING-GATE TRANSISTORS

For years floating gate transistors have been used in commercial non-volatile digital memories, but only recently have similar devices been considered viable as analog memories [18] and numerous other analog circuit elements. As memories, floating gate transistors have been used to build data converters [17, 19, 20], programmable references [21, 22], and non-volatile switching arrays [23]. Floating gate transistors have been used in analog circuits for such tasks as trimming offsets [14–16] and analog signal processing [24]. Industry has even begun to offer commercial floating gate technologies such as the electronically trimmed zero threshold transistor [25] designed for extremely low power supplies. However, all of these devices share common characteristics and programming techniques despite their dissimilar usage.

2.1 Characteristics

A floating gate transistor is simply a normal transistor except that the gate terminal has no DC path to a fixed potential. Instead, voltages are coupled to the floating gate via coupling capacitors. Figure 2.1 shows the layout for a floating gate pFET with a single drawn coupling capacitor and a special purpose coupling capacitor. The pFET can be seen on the right side of the right N-well. To the left of the pFET is the coupling capacitor made from a poly-poly capacitor. This capacitor is constructed above the N-well of the pFET to reduce the parasitic coupling capacitor to the substrate. Poly-poly capacitors are preferred for coupling because they maintain the same relative capacitance regardless of the voltage across them, unlike MOS capacitors. Although only one of these capacitors is drawn, there can be any number of coupling capacitors attached to the floating node. The special purpose coupling capacitor is made of a MOS capacitor in its own N-well because of the oxide quality needed for tunneling, which will be discussed later.

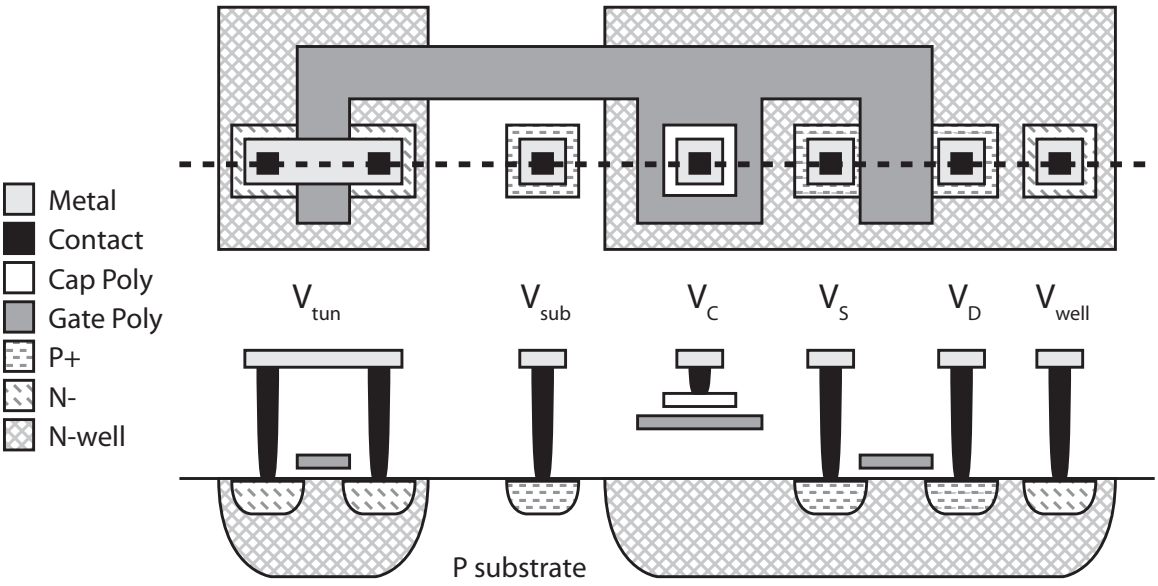


Figure 2.1. Top view and cross-section layout of a floating gate transistor.

Figure 2.2a shows the schematic representation of the floating gate pFET with a tunneling junction, C_{tun}, and one drawn coupling capacitor, C_C. These coupling capacitors allow a ratio of the coupling voltage, V_C, and the tunneling voltage, V_{tun}, to be seen as part of the floating gate voltage, V_{FG}, on the floating node. In addition to the drawn capacitors, there are generally several parasitic capacitors that also couple voltages into the floating node.

Figure 2.2b shows the floating gate pFET schematic with both drawn and parasitic coupling capacitors. The drawn capacitors are the same as the simple case. The well capacitor, C_{well}, is a result of the polysilicon area residing above the N-well of the pFET. Since this polysilicon is mostly above thick field oxide, the capacitance is usually fairly small. However, large poly-poly coupling capacitors can significantly increase the amount of polysilicon area, which increases the significance of the well capacitor. The substrate capacitor, C_{sub}, is formed by the region of polysilicon that crosses above the substrate between the two N-wells, as seen in Figure 2.1. This region of polysilicon is also over field oxide and is thus relatively small. However, its effects can be observed during programming, as will be discussed later. The final two capacitors, C_S and C_D, are the overlap capacitances of the pFET, which means that the source and drain signals can couple into the floating node.

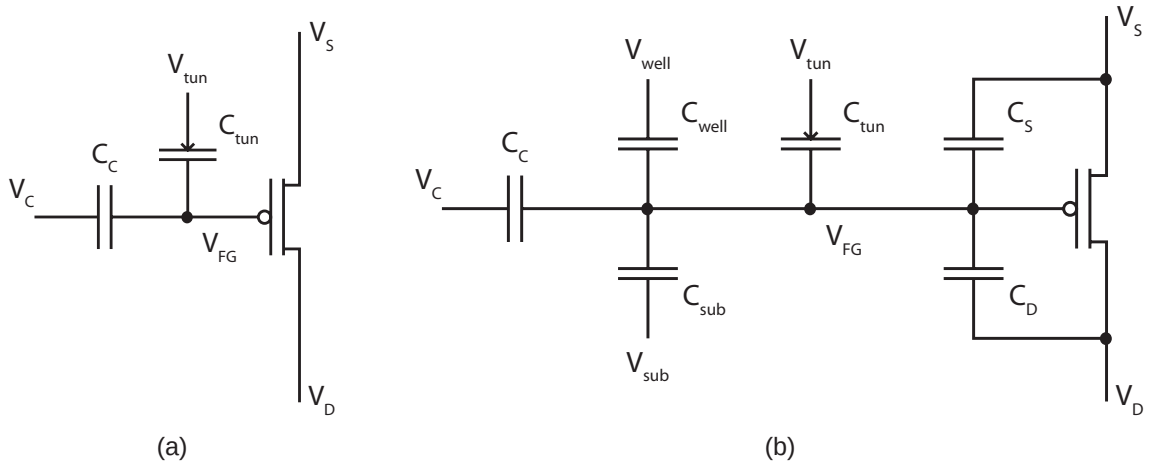


Figure 2.2. Floating gate transistor schematic.
(a) Simplified schematic showing only drawn coupling capacitors.
(b) Elaborated schematic showing all coupling capacitors including parasitic capacitors.

The voltage to current equations used to model ordinary FETs can also be applied to floating gate devices by substituting the floating gate voltage for the gate voltage, as seen in (2.1), the sub-threshold saturation equation. The terminal voltages in (2.1) are all referenced to the bulk material of the FET, which is the well voltage in the case of the floating gate pFET. The floating gate voltage can be expressed by (2.2) where the individual capacitors and voltages correspond to components in Figure 2.2b. The total capacitance, C_T, is the sum of the capacitances seen at the floating node, and the Q term represents the charge stored on the floating gate. If the coupling capacitor and tunneling capacitor comprise a significant portion of the total capacitance, this equation can be simplified to (2.3).

$$I_D = I_0 \, e^{\frac{\kappa V_{FG} - V_S}{U_T}} \, e^{\frac{V_D}{V_A}} \tag{2.1}$$

$$V_{FG} = \frac{C_C V_C + C_{tun} V_{tun} + C_{well} V_{well} + C_{sub} V_{sub} + C_S V_S + C_D V_D + Q}{C_T} \tag{2.2}$$

$$\approx \frac{C_C V_C + C_{tun} V_{tun} + Q}{C_T} \tag{2.3}$$

2.2 Programming

Floating gate transistors can be programmed by modifying the charge term, Q, of (2.2). Since the polysilicon gate of the transistor is completely surrounded by oxide, the charge can be stored on the floating node for long periods of time [15, 16, 26]. Fowler-Nordheim tunneling and hot electron injection are commonly used to move charge across the oxide barrier. The result of these processes can be seen in the gate sweeps of Figure 2.3, which depicts a single floating gate pFET programmed to three different levels of charge.

Tunneling removes electrons from the floating node, so the I-V relationship shifts to the left, which looks like the effective threshold voltage of the pFET has been increased. Hot electron injection adds electrons to the floating node and thereby decreases the effective threshold voltage, which shifts the I-V curve to the right. If a single coupling voltage value is examined for the floating gate pFET example, tunneling can be viewed as reducing the amount of current flowing through the channel, and hot electron injection increases the current. In this manner, a programmable current source can be constructed. Similarly, the same processes can be used to change the conductance of the floating gate pFET, so it can

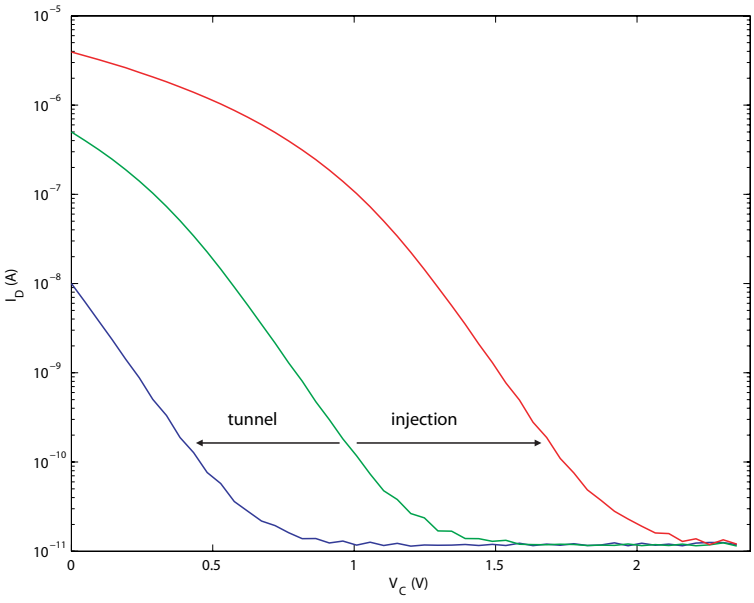


Figure 2.3. Gate sweep measurements showing the programmability of floating gate transistors.

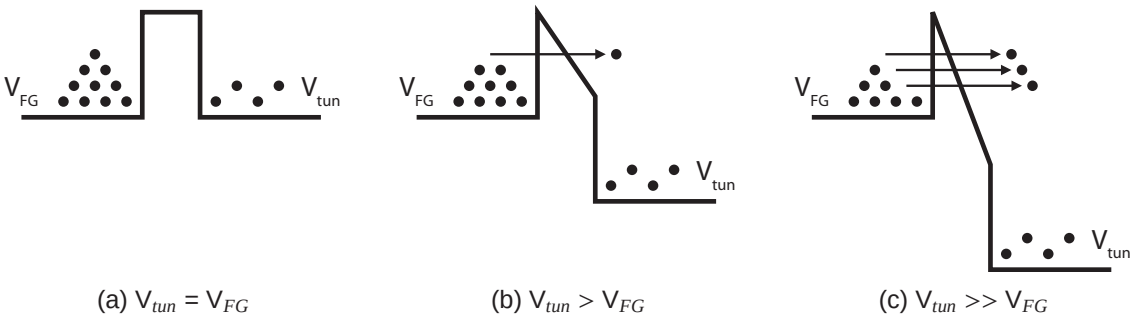


Figure 2.4. Conduction band diagram depicting the tunneling process across an oxide.

be used as a programmable switch.

Under normal circumstances, the oxide barrier significantly reduces the probability of electrons moving to and from the floating node. Figure 2.4a illustrates the conduction band as seen across the tunneling junction capacitor. In the initial state, the floating gate voltage and the tunneling voltage are equal. The phenomenon of Fowler-Nordheim tunneling can then be observed by raising the tunneling voltage, which lowers the conduction band as seen in Figure 2.4b. As the tunneling voltage increases, the probability of an electron crossing the oxide barrier increases. The decreasing conduction band on the tunneling voltage side has the effect of decreasing the barrier width observed by electrons on the floating node. As the effective width of this barrier continues to shrink, more electrons will be able to "tunnel" through the oxide, Figure 2.4c.

The process of hot electron injection is illustrated in Figure 2.5. To inject electrons onto

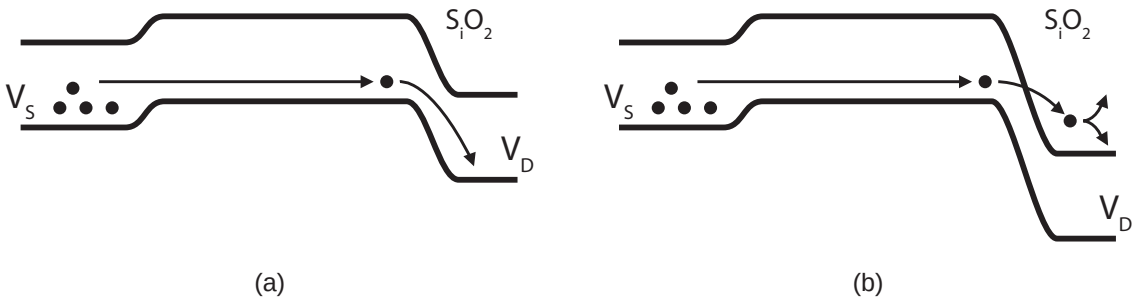


Figure 2.5. Conduction band diagram depicting hot electron injection across an nFET channel.
(a) Moderate field between source and drain.
(b) High field between source and drain.

the floating node, two conditions are required. The first is channel current, and the second is a high field between the source and drain terminals. Under normal conditions, Figure 2.5a, electrons flow through the channel from the source to the drain terminal. Upon reaching the channel-to-drain junction, the electron can be seen to roll down the conduction band where the electric field is highest. If the drain terminal voltage is increased, thereby lowering the conduction band further, the field in this region is significantly increased, Figure 2.5b. In this situation, the electron may now have enough energy to surmount the oxide barrier due to the field. Once in the oxide, the electron will most likely drop back into the drain. However, some of these electrons will cross the oxide and become trapped on the floating node.

Hot electron injection is a positive feedback process in pFETs, since the number of electrons injected onto the floating node is proportional to the amount of current flowing through the channel. As the number of electrons increase on the floating node, the effective floating gate voltage decreases, which increases the amount of current flowing through the channel. In order to accurately program a floating gate transistor, an algorithm of controlled injection pulses, Figure 2.6, has been developed [27–29]. The injection pulse

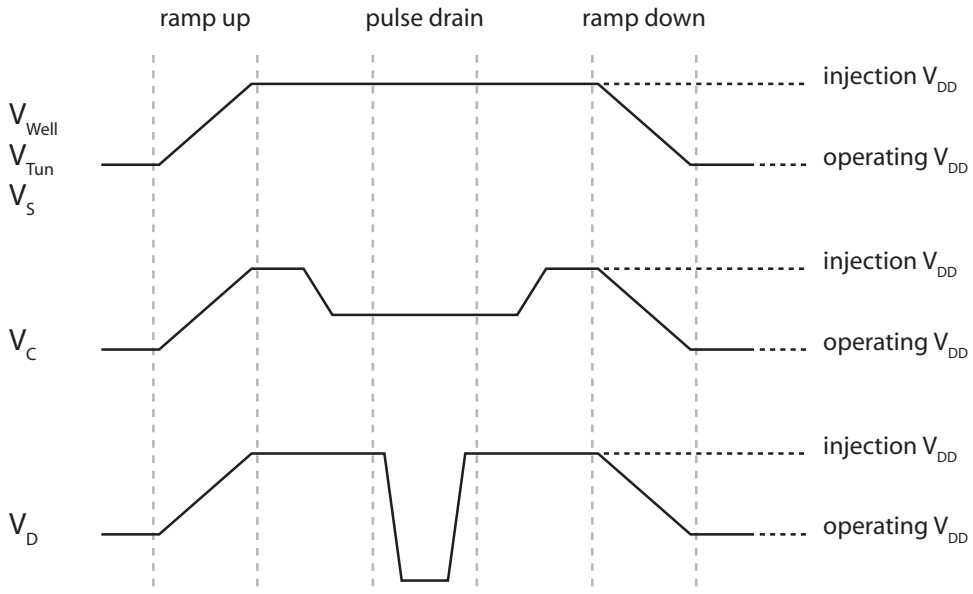


Figure 2.6. Timing diagram showing the steps involved in a hot electron injection pulse.

has two distinct phases, ramp and pulse. During the ramp phases, the coupling voltages of the FET are held constant relative to the bulk potential to prevent accidental injection. To achieve a high field across the transistor during the pulse, the bulk potential is often raised well above operating conditions, as seen in Figure 2.6. Once the terminals have reached the desired supply voltage, the coupling voltages, in this case just V_C, are adjusted to bias the transistor such that current flows through the channel. The drain is then pulsed for a fixed width and returned to the supply voltage. Another ramp phase returns the supply voltage to normal operation levels.

The algorithm in [28] characterizes the injection pulses for a fixed pulse width, a fixed coupling voltage with respect to V_{DD}, and a range of source to drain voltages. A predictive model is then generated such that given the initial current flowing through the transistor and the desired target current, the model can estimate the source-to-drain voltage needed during the pulse to reach the target. Since each floating gate transistor injects at slightly different rates, the algorithm uses a conservative estimate and iterates between current measurement and pulse phases in order to asymptotically approach the target current. This algorithm is capable of .2% accuracy [28] over 3.5 decades of current, but it uses a fixed coupling

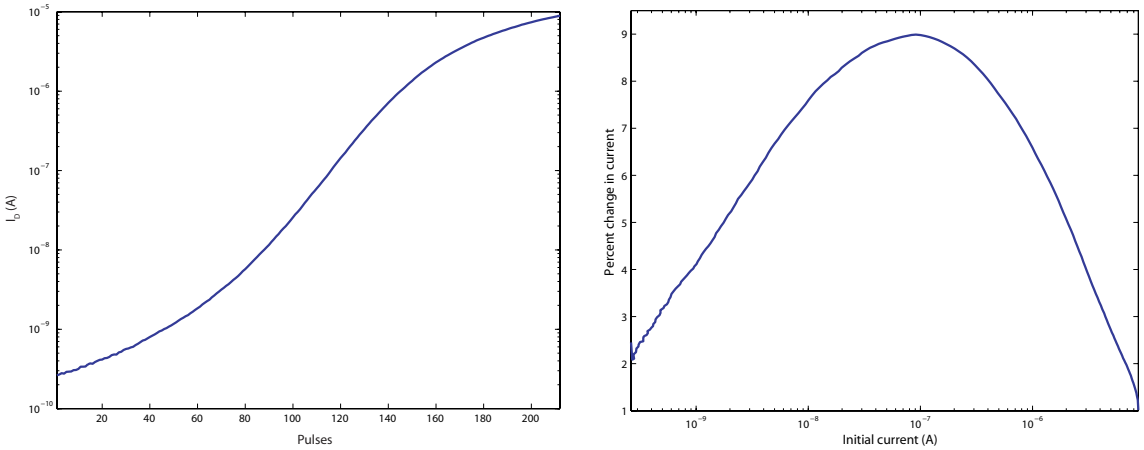


Figure 2.7. Floating gate injection efficiency.
(a) Drain current measured after each injection pulse.
(b) Percent change in current for each initial current.

voltage, which results in a significantly varying injection efficiency per pulse.

Figure 2.7a shows how the drain current of a floating gate pFET changes with each drain pulse given a fixed source to drain voltage and coupling voltage. Figure 2.7b shows the percent change in current for a given initial current. From these measurements it can be seen that the injection efficiency varies significantly over the programmable range of currents. Using this knowledge, the coupling voltage can be adjusted during the pulse phase to maximize the injection efficiency. As a result, this modification should reduce the number of pulses required to program larger target currents.

2.3 Floating Gate Transistor Arrays

In general, floating gate transistors are arranged into a two dimensional array during programming, as seen in Figure 2.8. In this configuration, all of the transistor source terminals are connected to V_{DD}. The coupling voltages and drain voltages are switched between a fixed potential, usually V_{DD}, and a DAC voltage. A decoder, shift register, or combination

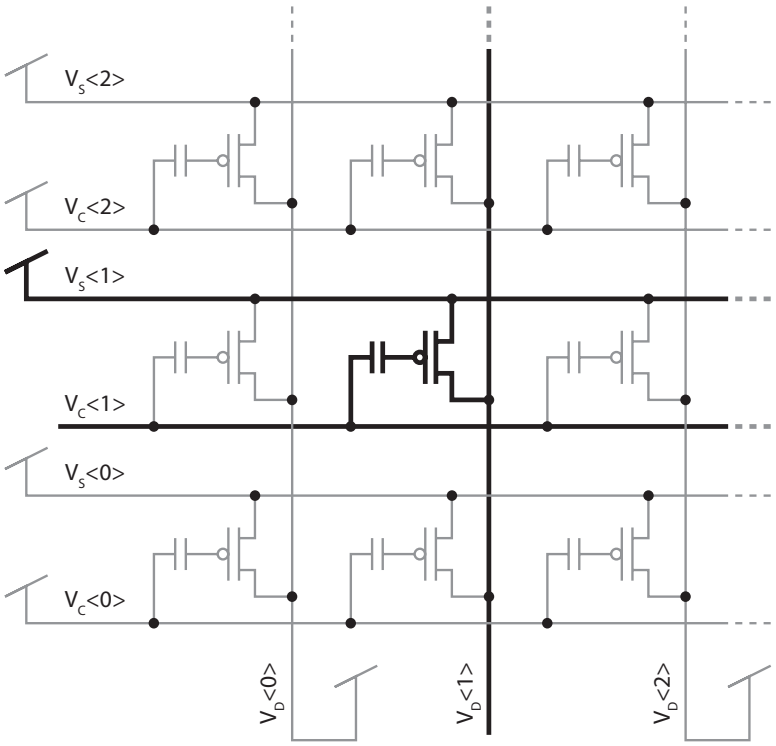
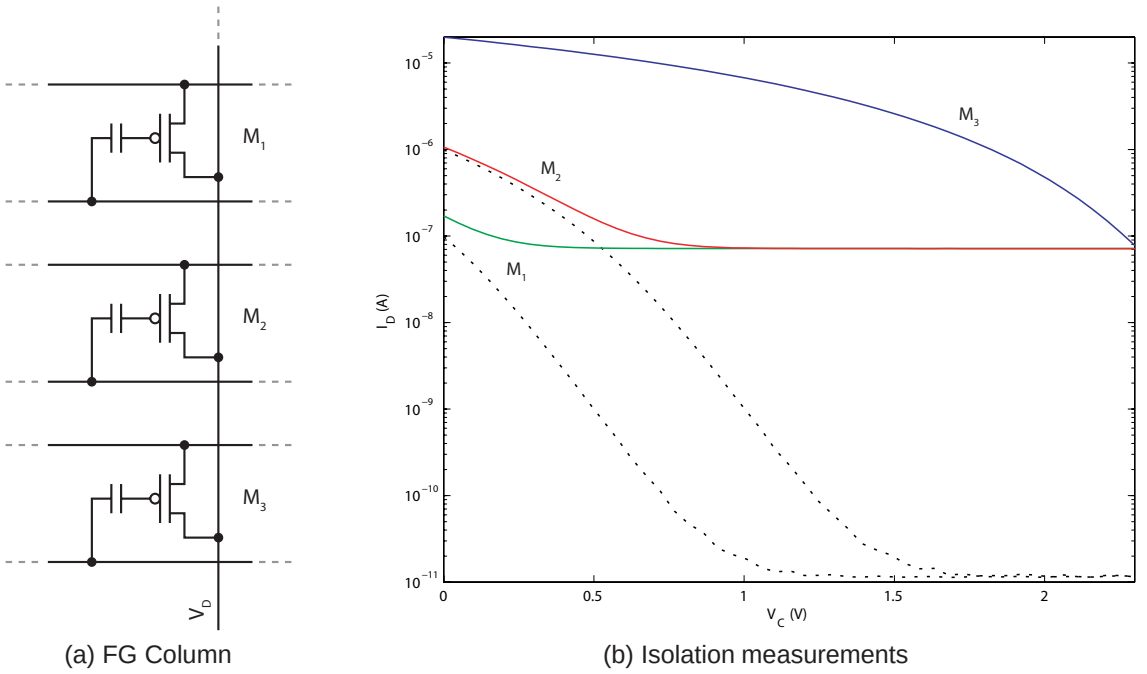


Figure 2.8. Floating gate transistors arranged into an array for programming.

of the two control which row and which column are selected at a time. In Figure 2.8, the center column and center row are selected. The coupling voltages of the unselected rows are connected to V_{DD}, and the unselected drain lines are also connected to V_{DD}. Since channel current and high source-to-drain field are both required for hot electron injection, this configuration allows individual transistor selection within the matrix without additional isolation hardware. The high field is applied to the selected column, but only the selected row allows a significant channel current to flow through the desired transistor.

The selectivity of individual devices within the array is only valid when the coupling and drain voltages used for the unselected devices in the array can maintain isolation. Figure 2.9 graphically shows the conditions required to maintain isolation during programming. Transistors M_1 and M_2 were programmed to have drain currents of 100 nA and 1 μA respectively at a coupling voltage of 0 V, as seen by the dotted lines in Figure 2.9b. The gate sweep for each of these transistors shows that the transistor current can still be shut



(a) FG Column (b) Isolation measurements

Figure 2.9. Floating gate transistor isolation in arrays.
(a) A column of floating gate transistors.
(b) Gate sweep measurements depicting the concept of device isolation within floating gate transistor arrays. The dotted lines show what M_1 and M_2 would look like if device isolation was maintained.

off with the coupling voltage at V_{DD}, which is 2.4 V in Figure 2.9b. A third transistor, M_3, was then programmed to 20 μA at a coupling voltage of 0 V. This transistor can no longer be turned off with the coupling voltage at V_{DD}, so it will conduct whenever the column is selected. The current contribution from this transistor now interferes with the current measurements of M_1 and M_2, but it would completely mask currents less than 100 nA, which makes accurately programming such currents impossible once isolation has been violated. Even if another transistor along the column could be programmed given the state of M_3, the current flowing through M_3 during the injection pulse would cause it to further inject.

2.4 Switch Characteristics

In addition to programmable biases, floating gate transistors can also be used as programmable conductance switches. An ideal switch is characterized by infinite impedance, or no conductivity, in the "off" state and zero impedance, or infinite conductivity, in the "on" state. Of course, no such device exists in reality. Figure 2.10 depicts the "on" and "off" states of floating gate pFETs using gate sweeps. An "off" switch is tunneled such that no measurable current flows through it. Although some minimum level of current is

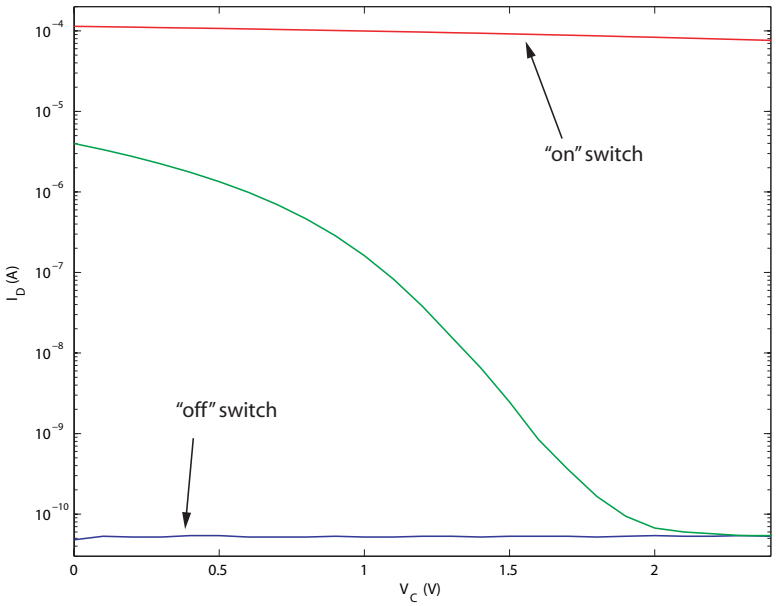


Figure 2.10. Gate sweeps depicting the "on" and "off" states of floating gate switches.

observed in the figure, this is a result of the combined leakage currents of all the transistors attached to a single column in the array plus the leakage current of the reverse biased diodes formed by the ESD structure and the drain to well junctions. In the "on" state, the floating gate transistor is injected as strongly as possible to provide the best conductivity. However, floating gate transistor switches are not limited to simply "on" and "off" states like their pass FET and transmission gate alternatives generally are. Instead, the floating gate transistor can be programmed to any intermediate state, as seen in Figure 2.10.

Generally, pass FETs or transmission gates are used as switches, depending upon the requirements of the system. Floating gate pFET switches are simply a form of pass pFET in which the gate biasing is controlled through charge programming and coupling capacitors. However, a significant difference can be observed in Figure 2.11. The circuit in Figure 2.11a was used to evaluate and compare the resistance of a pFET, a transmission gate, and a floating gate pFET when used as a switch. Each switch element was biased

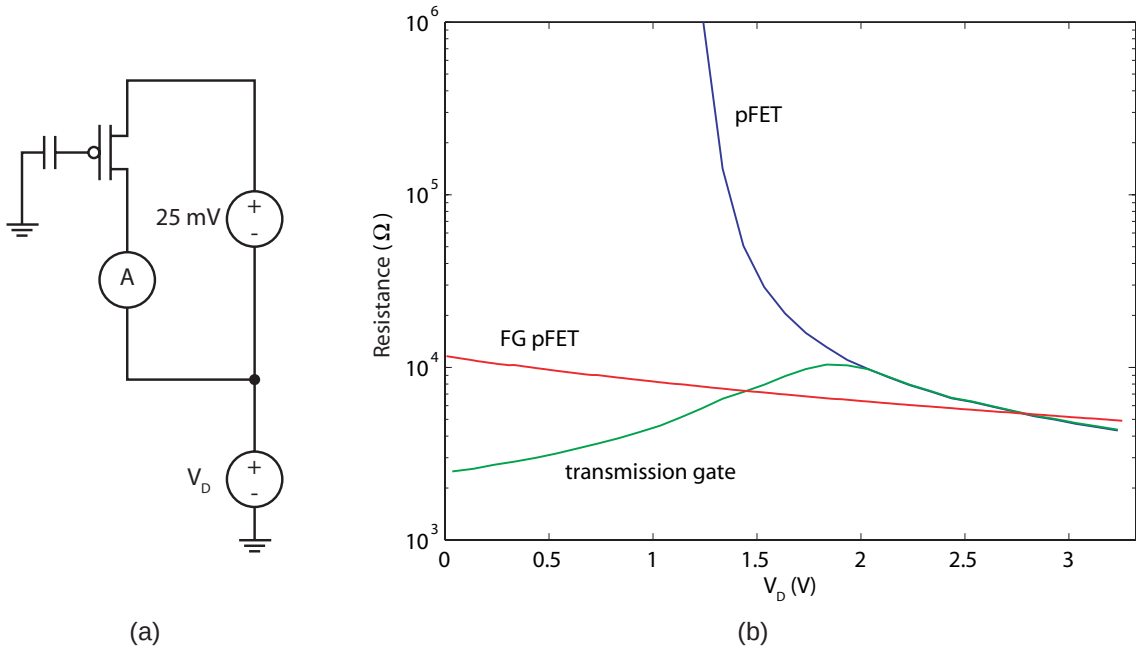


Figure 2.11. Comparison of switch resistance for various devices.
(a) The resistance was measured with the depicted circuit.
(b) Measurements were taken for a pFET, a transmission gate, and a floating gate pFET of similar sizes in a .5 μm process.

such that a constant 25 mV drop was observed across the switch terminals. One end of the switch was swept across the supply voltage range, and the current was measured at each step. The resistance was then estimated using Ohm's law, Figure 2.11b. As the figure shows, the pFET resistance increases dramatically as the signal passing through it is reduced. This means that the pFET will have trouble passing low voltage signals when connected to a low impedance. To solve this problem, transmission gates are generally employed to cover the entire signal range at a cost of extra capacitance, which reduces the switch bandwidth. The pFET in the transmission gate reduces the switches resistance for the upper end of the signal range, and the nFET handles the lower end.

The floating gate pFET's resistance in Figure 2.11b is on the same order as that of the transmission gate, but it has significantly less parasitic capacitance because there is only one transistor, which improves signal bandwidth. The floating gate pFET's resistance is monotonic with decreasing signal voltage and varies less than the transmission gate over the entire signal range. For any DC signal bias, the relative change in resistance for an AC signal is approximately the same. Although the floating gate pFET resistance curve looks significantly different, it is actually very similar, except shift to the left. The charge on the floating gate has been programmed such that the effective floating gate voltage is below the negative supply. If a negative gate voltage were applied to the pFET, the resistance curve would look very similar to the floating gate pFET. However, the floating gate pFET's floating gate voltage is also affected by the signal passing through it. The parasitic overlap capacitors from the source and drain couple into the floating node. For higher signal voltages, this coupling increases the resistance slightly. For lower signal voltages, this coupling decreases the resistance a bit. The effect is basically a horizontal stretching of the negatively biased pFET resistance curve, which explains why the resistance of the floating gate pFET is slightly higher than the pFET or the transmission gate for signals near the supply rail.

2.5 Switch Programming

Programming a floating gate transistor as a switch occurs in much the same manner as discussed before. For the intermediate switch conductance values, the precision programming schemes [27–29] work in exactly the same fashion. However, the "on" state of Figure 2.10 requires programming the individual transistors beyond the point of isolation. This means that multiple switches cannot be reliably turned "on" in a column by programming them sequentially.

The current masking effect observed in Figure 2.9b would prevent additional transistors along a column from being measured during the iterative programming scheme. However, this is not crucial given that the transistor is not being accurately programmed to a specific point. Rather, it is being injected as hard as possible to make the best possible switch. The real problem is the amount of current flowing through the selection circuitry attached to the drain terminal [30]. The selection circuitry has a finite conductance and therefore has an associated voltage drop across its terminals when a significant amount of current flows through it. This voltage drop reduces the field across the floating gate pFET during the injection pulse, which also reduces the injection efficiency and maximum conduction level of the switch. As such, it is very difficult to program multiple "on" transistors in a column using this method.

The algorithm proposed in [30] attempts to solve this problem by incrementally injecting each "on" transistor. The algorithm pulses each transistor to be turned "on" sequentially before returning to pulse the first transistor a second time. After each iteration, the coupling voltage is increased slightly, which decreases the current flowing through the devices during the injection pulse thereby reducing the field dropped across the selection circuitry. By repeating this method, each device is slowly injected up to and beyond the point of isolation. However, the first transistor to breach the isolation point will then continue to inject for every drain pulse on the column. If a large number of transistors are to be turned "on" in a given column, this could result in the same diminishing field problem as before.

An alternate method for programming “on” switches utilizes the substrate coupling capacitor, C_{sub}, from Figure 2.2b. Figure 2.12 shows the effect of increasing the supply voltage during the injection pulse. Although all of the other terminals are adjusted with respect to the supply voltage during an injection pulse, the substrate voltage cannot be. Therefore, the substrate couples in an effectively lower voltage, which increases the current flowing through the transistor when in the high supply voltage state. Therefore, the isolation point is actually lower than observed during normal supply voltage operation. The shift of the I-V curve is linearly dependent upon the difference between the operating and injection supply voltages. To account for this shift, most programming algorithms will ramp the supply voltage to a consistent injection supply voltage independent of the source to drain voltage of the injection pulse. By characterizing the injection pulses under this constraint, the substrate coupling has little affect upon the algorithm.

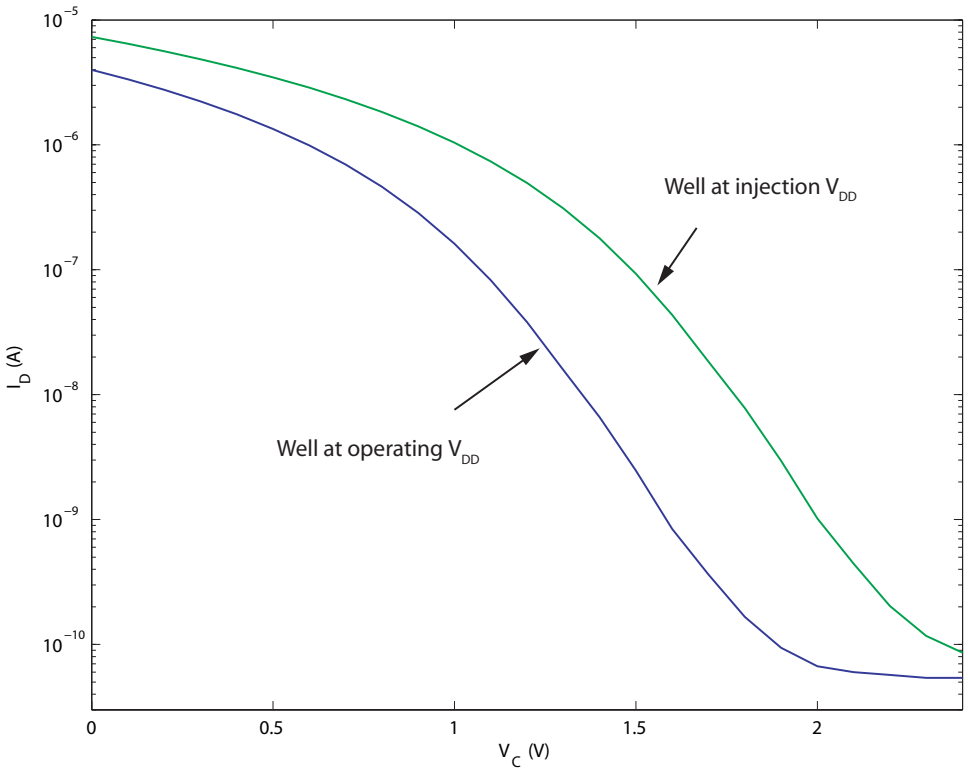


Figure 2.12. Exploiting the substrate coupling capacitor for switch programming.

For switches, this substrate coupling can be used to inject all "on" devices simultaneously beyond the point of isolation. The first step programs each "on" transistor along the column up to the point of breaching isolation. The coupling voltage is then held at the supply voltage, and the injection supply voltage is increased beyond the characterized value. By doing this, all of the transistors programmed to the isolation point will be pushed slightly beyond isolation and conduct current with their coupling voltages at the supply voltage. A drain pulse under these conditions will thus cause all of these devices to inject at the same time, which means they all see the same source to drain field. In this manner, any number of transistors should be programmable to the "on" state. Although all of the "on" devices should be programmed to the same relative level, they may not conduct as well as a single "on" switch, since the field will be reduced significantly the the selection circuitry.

Floating-gate nFET transistors would seem like a more appropriate choice for a switch, since the conductance of an nFET is significantly higher than that of an equally sized pFET. However, most processes now use spacers around the gates of transistors to reduce the amount of hot electron injection in order to lower the power consumption caused by gate leakage in high-speed digital systems. This means a drastically reduced injection efficiency in nFETs, since the injection mechanism occurs near the drain junction. In pFETs, injection occurs further away from the drain than in the nFET case, which means the spacers have less of an effect. Therefore, floating gate pFETs have been the floating gate transistor of choice for this work.

2.6 Indirect Programming

In some instances, the selection circuitry necessary to pull a floating gate transistor out of an analog circuit is undesirable. Figure 2.13a shows an example circuit using a floating gate transistor to set a bias voltage. Although the switches in this example are not detrimental to the operation of the diode-connected nFET, the simplicity better illustrates the mechanisms

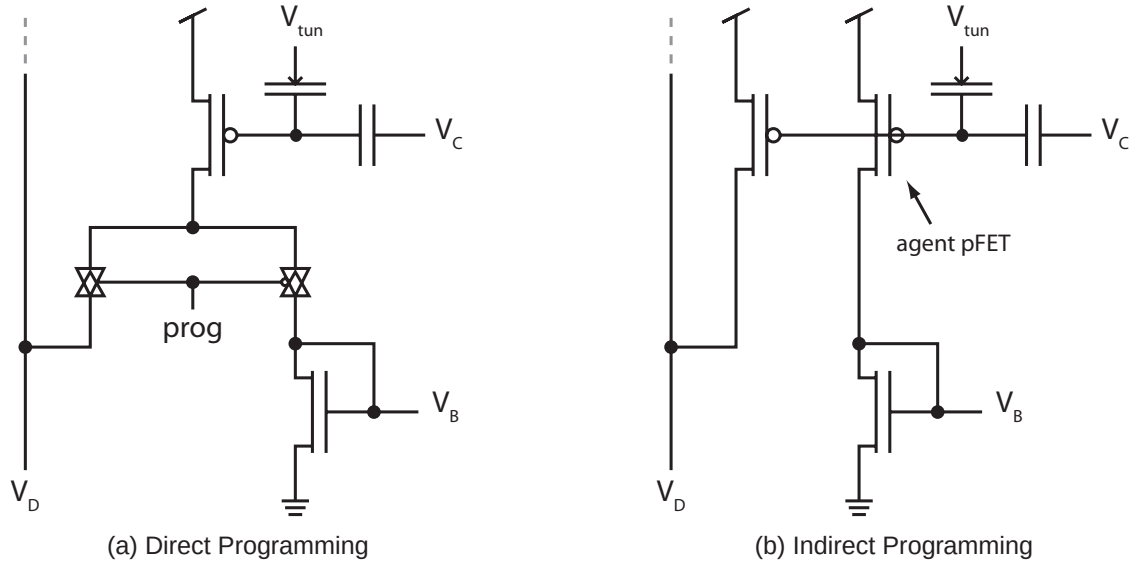


Figure 2.13. Direct versus indirect floating gate transistor programming.
(a) Direct programming requiring selection switches.
(b) Indirect programming requiring no additional switches.

required for programming floating gate transistors within a circuit. During program mode, the "prog" signal, as seen in Figure 2.13a, is driven high. This switches the drain of the floating gate pFET from the circuit to the programming drain line. In run mode, the "prog" signal is held low, which switches the floating gate pFET back into the circuit.

A way to avoid extra selection switches can be seen in Figure 2.13b. Instead of a single pFET, there are now two pFETs that share a common floating node [31]. One of these pFETs is attached directly to the circuit of interest. The other is wired into the programming circuitry. In this configuration, the charge on the floating node can be modified using the programmer pFET, the pFET connected to the programming circuitry, without disturbing the terminals of the agent pFET, the pFET connected to the circuit of interest.

Indirect programming also provides an effective way to use floating gate nFETs in circuits. By replacing the agent pFET with an agent nFET, the programmer pFET can now be used to modify the charge controlling the floating gate of the nFET, which avoids the low injection efficiency problem of the nFET. However, this makes programming the nFET a little trickier than in the pFET case. Tunneling removes electrons from the floating node,

which increases the effective floating gate voltage. This turns off the pFET, but it turns on the nFET. This is not a significant problem, but it does require injection to turn "off" the nFET. In a large system, such as an FPAA, this can be disadvantageous because of the time required to program "off" these devices. Tunneling is generally a global procedure in that it affects all of the devices simultaneously. For pFETs, this can be viewed as a global erase, but for nFETs, all of the devices are turned on very hard. In a reconfigurable system, global erasure makes sense, since a majority of the devices will not be used in any given system. The process of injecting these nFET devices "off" makes this impractical on a large scale, such as would be the case for switches. However, using these devices in a few special case circuits or biases within large-scale devices may be practical.

2.7 Modified Tunneling Junctions

In most reconfigurable systems, the switching fabric consumes a significant portion of the die area, which is also the case for the FPAAs described in this work. Most of the space in floating gate pFET switches is consumed by the tunneling junction. Since this junction is formed by a MOS capacitor residing in its own well, it requires an isolating substrate space between its well and the well of the pFET, as seen in Figure 2.1. Because the tunneling voltage is significantly higher than the operating voltage of the process, this substrate space between the wells cannot be used for active components, so it is mostly wasted space.

One way to conserve space in floating gate transistor switch networks is to eliminate the special tunneling junction, as seen in Figure 2.14. In this switch topology, the well potential can be raised to a high enough voltage to cause the tunneling phenomenon across the capacitor formed between the pFET's gate and the well. Using this and the hot electron injection mechanism discussed earlier, the charge on these well tunneled devices can be modified, as seen in Figure 2.15.

Although this structure functionally works, it may not be practical for large-scale systems. When increasing the well potential during tunneling, it is also necessary to increase

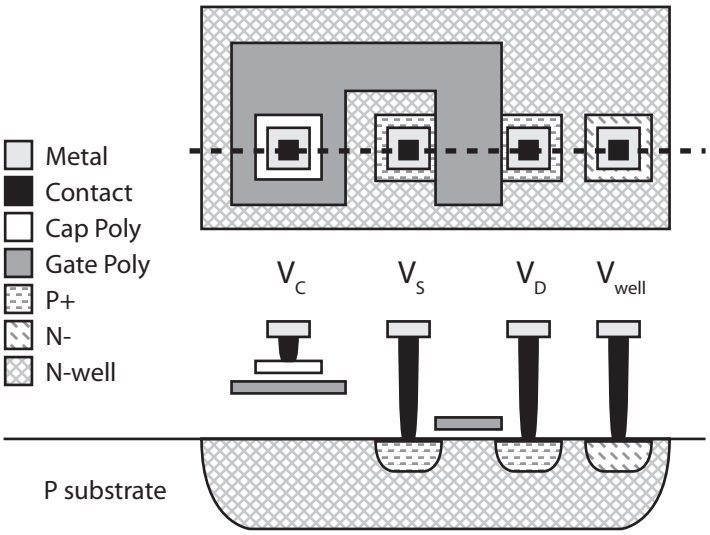


Figure 2.14. Layout for a floating gate transistor using well tunneling.

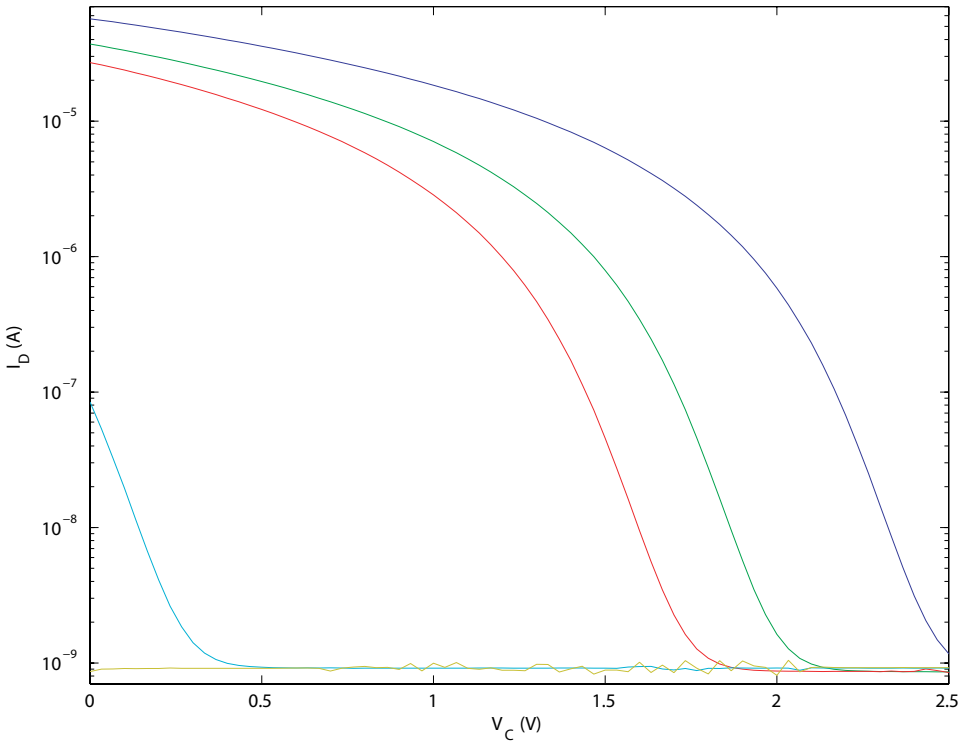


Figure 2.15. Gate sweep data showing well tunneling results.

the source and drain voltages at the same time. The junctions between the source/drain active regions and the well form parasitic diodes. During normal operation, the well is biased at or above the highest potential of the pFET's source terminal. In this manner, these diodes are reverse biased and conduct only leakage current. However, raising the well voltage significantly higher than the process voltage causes these diodes to break down and conduct high levels of current, which can cause damage to the device or IC. To compensate for this issue, the source and drain terminals were raised along with the well to prevent breakdown. In an array of floating gate transistors, this would require special level shifting capabilities and other high-voltage interface circuits to handle the higher source and drain potentials. The raising of the source and drain potentials also couples into the floating node thereby increasing the effective floating gate voltage. This reduces the effectiveness of the tunneling voltage, but increasing the coupling capacitor size and driving the coupling voltage to ground can help compensate for this effect.

The possibility of replacing the MOS tunneling capacitor with a poly-poly capacitor was also investigated. Figure 2.16 depicts the layout of such a floating gate transistor. In this topology, the tunneling capacitors work much in the same manner as the MOS capacitor used in the standard floating gate pFET design. The charge stored on the floating node was

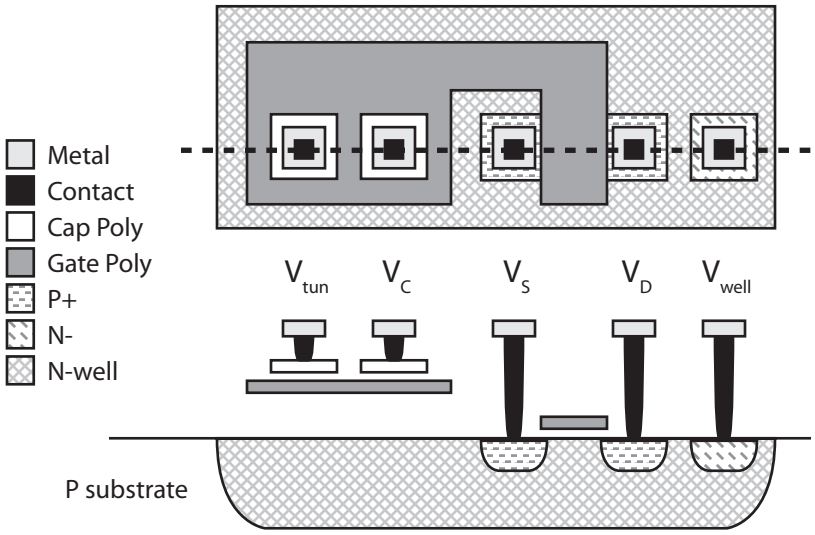


Figure 2.16. Layout for a floating gate transistor using poly-poly cap tunneling.

modified using the same techniques as with the standard floating gate transistor. However, the voltage required to tunnel the device was significantly higher than the standard pFET case, since the oxide thickness of the poly-poly capacitor is greater than that of the gate oxide used in MOS capacitors. Interestingly, this structure also allows electrons to be tunneled onto the floating node by decreasing the tunneling voltage. A negative voltage is usually required to do this, and a MOS capacitor would not be able to handle such voltages, since it would forward bias the junction diodes. However, this is not the case with the poly-poly cap, and a negative tunneling voltage can be used.

2.8 Improving Isolation

Indirect programming may be helpful in dealing with the isolation issues involved with programming switches. Figure 2.17 shows a new switch topology based upon indirect injection. In this circuit, the injection mechanism occurs within the programmer pFET, so an extra selection pFET can be added just below the programmer pFET, which allows the current to be shut off for unselected rows. This configuration allows "on" switches to be injected individually and ensures that all of the "on" transistors along a column are injected to the maximum possible level.

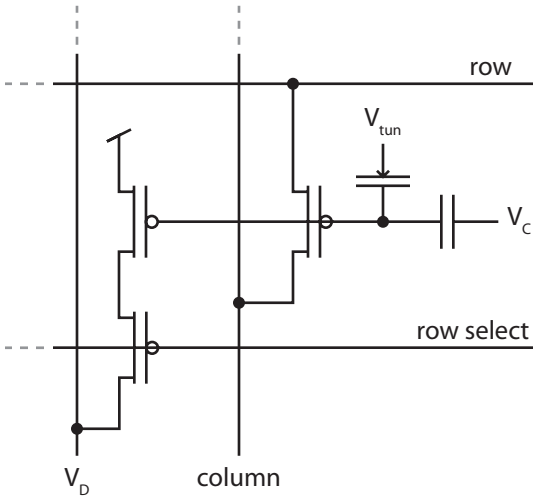


Figure 2.17. Switch element using indirect programming.

Figure 2.18 shows some measured gate sweep results for this switch topology. With the selection pFET shut off, no current from the programmer pFET can be observed. After turning on the selection pFET, current can be seen to flow through the programmer pFET. The initial current flowing through the pFET is in between an “off” and “on” transistor switch, as seen in Figure 2.18. After injecting the programmer pFET, the current seen flowing through the programmer pFET looks more like an “on” switch. However, the observability of this current is being limited by the selection pFET, because it is within the signal path. Performing a gate sweep on the switch pFET, or agent pFET as described earlier, results in the much higher current levels expected of “on” switches, as seen in Figure 2.18.

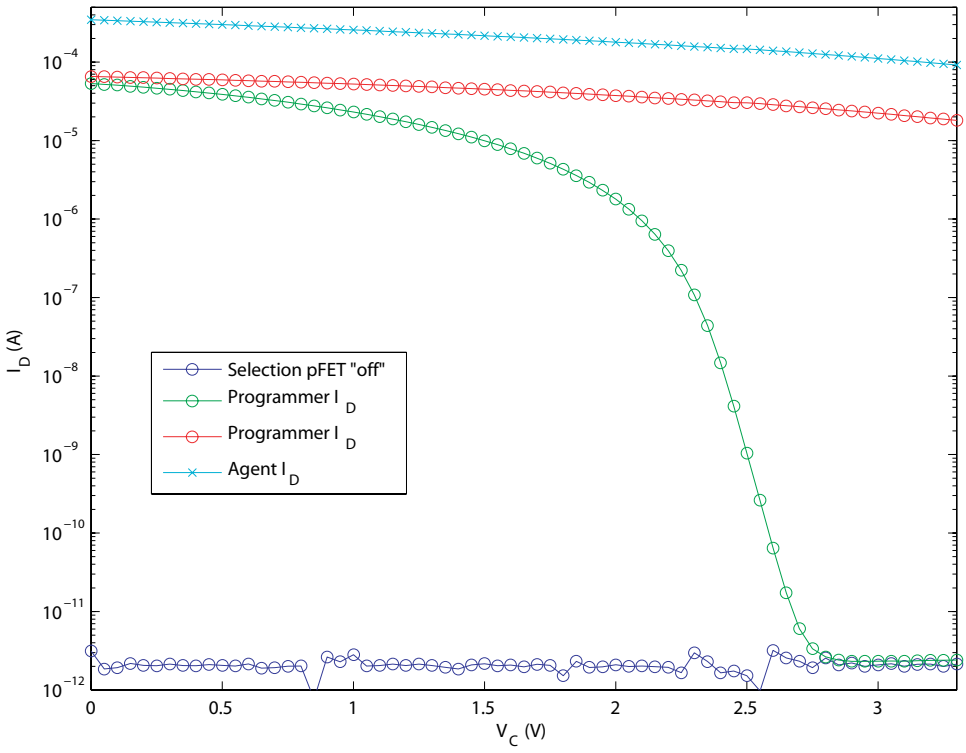


Figure 2.18. Gate sweep results from the indirectly programmed switch topology.

CHAPTER 3

PROGRAMMABLE VOLTAGE / CURRENT REFERENCE

Analog systems generally require many different bias voltages and currents to set various circuit parameters. For large scale analog systems, such as those synthesizable with large-scale FPAAs, the number of biases could easily number in the 100s to 1000s. It is impractical to dedicate IC pins for this level of biasing, so an on-chip solution is required. Since many process parameters are temperature dependent, the analog circuit characteristics will also be temperature dependent unless the biasing structures compensate for these temperature effects. For programmable and reconfigurable systems, such as FPAAs, the synthesized analog circuits must maintain their performance characteristics over a reasonable range of temperatures to gain commercial acceptance.

Although temperature independent circuit topologies, such as the very popular bandgap reference [32–34], are in common use, they tend to consume a large die area and have a fixed reference value. The fixed reference is obviously a problem for programmable systems, but this can be overcome through the use of selectable references, which are effectively DACs. The real issue is the area requirement for the large number of on-chip biases. Accuracy requirements and large DAC array structures force the significant area consumption in standard reference designs. High accuracy often also comes at the expense of post fabrication trimming using lasers or other techniques. However, floating gate transistors can be introduced to increase the initial accuracy and decrease the die area through programming.

3.1 Architecture and Theory

A floating gate based programmable voltage reference has been developed[1] [19] based upon the common beta-multiplier reference circuit [35]. As seen in Figure 3.1a, the pFET

[1]This work was done in collaboration with Venkatesh Srinivasan and Guillermo Serrano. It was partially funded by the JPL Self-Reconfigurable Electronics for Extreme Environments (SREE) project.

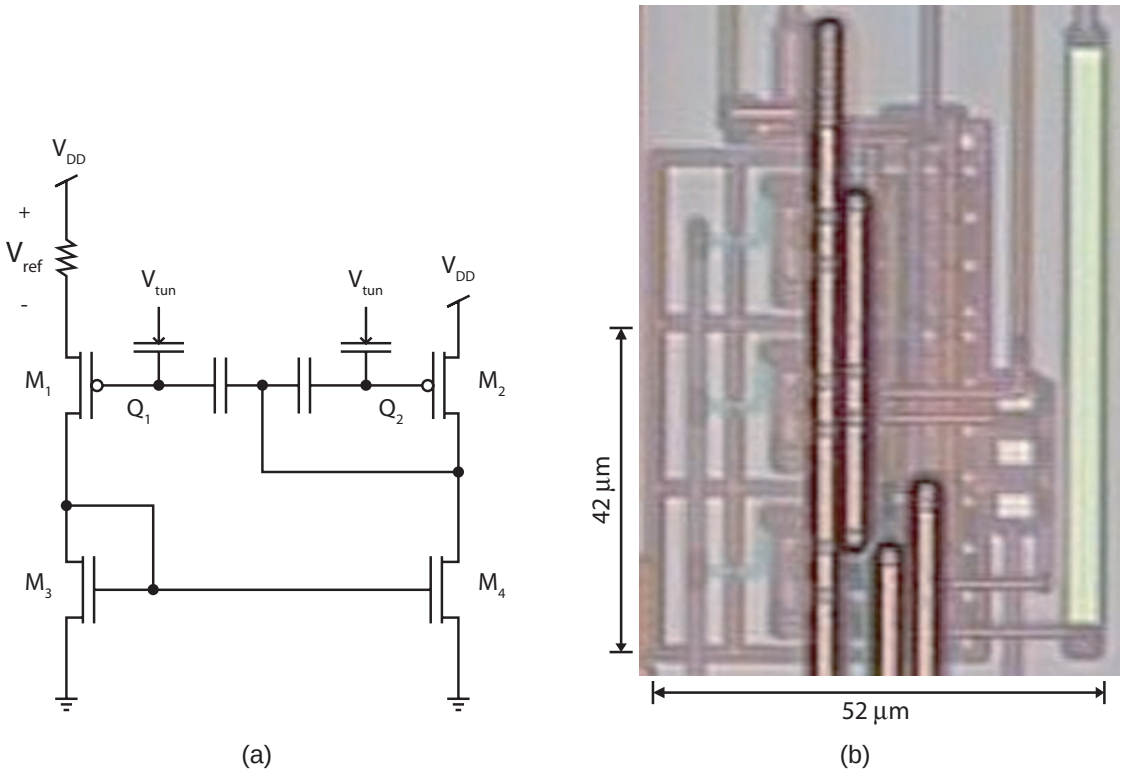


Figure 3.1. Programmable floating gate based reference.
(a) Schematic of the floating gate reference.
(b) Die photo of the floating gate reference with relevant dimensions.

transistors M_1 and M_2 have been replaced with floating gate pFETs. An on-chip resistor, in conjunction with the programmed reference voltage, is used to set the current flowing through each leg of the circuit. Assuming M_1 and M_2 are sized the same and M_3 and M_4 are matched, the current flowing through M_1 and M_2 should be the same. Figure 3.1b shows a die photo of the programmable reference designed in a .35 μm process using a 2.5 V supply. The dimensions for the circuit in Figure 3.1a are given in the die photo. The long rectangle along the right edge of the photo is the on-chip resistance, which can be significantly larger or smaller, depending upon the region of operation, above or sub-threshold. The extra circuitry shown in the photo was simply used for testing purposes and is not required by the reference.

Assuming perfect device matching and ignoring the Early effect, the current flowing through M_1 and M_2 are equal. In the saturated sub-threshold region, a bulk referenced

expression comparing the currents flowing through M_1 and M_2 can be seen in (3.1) and simplified in (3.2).

$$\begin{aligned} I_2 &= I_1 \\ I_0\, e^{\left(\frac{\kappa V_{FG2}}{U_T}\right)} &= I_0\, e^{\left(\frac{\kappa V_{FG1} - V_{ref}}{U_T}\right)} \end{aligned} \tag{3.1}$$

$$\kappa V_{FG2} = \kappa V_{FG1} - V_{ref} \tag{3.2}$$

In a similar fashion, the same result can be seen in (3.3) assuming above-threshold saturated operation and using a bulk referenced model.

$$\begin{aligned} \frac{K}{2\kappa}\left[\kappa\left(V_{FG2} - V_{th}\right)\right]^2 &= \frac{K}{2\kappa}\left[\kappa\left(V_{FG1} - V_{th}\right) + V_{ref}\right]^2 \\ \kappa V_{FG2} &= \kappa V_{FG1} - V_{ref} \end{aligned} \tag{3.3}$$

With the results of (3.2) and (3.3), the floating gate voltages can be expanded using (2.3) to form (3.4) and simplified to (3.5).

$$\begin{aligned} V_{ref} &= \kappa\left(V_{FG1} - V_{FG2}\right) \\ &= \kappa\left[\left(\frac{C_C V_C + C_{tun} V_{tun} + Q_1}{C_T}\right) - \left(\frac{C_C V_C + C_{tun} V_{tun} + Q_2}{C_T}\right)\right] \qquad (3.4) \\ &= \kappa\left(\frac{Q_1 - Q_2}{C_T}\right) \\ &= \kappa\frac{\Delta Q}{C_T} \qquad (3.5) \end{aligned}$$

From (3.5) it is easily seen that the reference voltage is proportional to the charge difference, ΔQ, between floating gate transistors M_1 and M_2.

3.2 Programmability

Using the programming techniques described in Chapter 2, the reference voltage can be set according to (3.5). Figure 3.2 shows the programmable reference circuit with switches needed to take the floating gate transistors out of the circuit and connect them to the programming lines. The source terminals are tied to V_{DD}, the drain terminals to independent

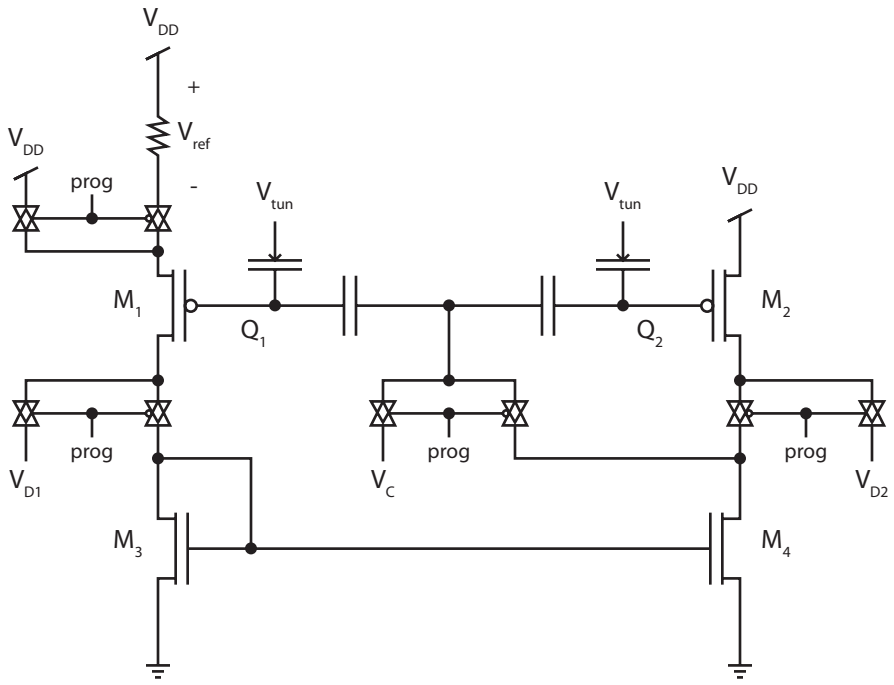


Figure 3.2. Programmable reference schematic showing programming switches.

drain lines, and the coupling voltage is switched from its normal diode connection to a control voltage.

In addition to providing connectivity to the programming lines, the switches in Figure 3.2 eliminate the need for a start-up circuit. This particular reference topology has two stable operating points. One stable point is the desired operating condition in which a current flows through both legs of the circuit as set by the resistor. The second stability point occurs when no current flows through either leg. In this case, the current through each leg is still equal, just zero. To ensure proper reference operation, a start-up circuit is often used to inject a current into one of the circuit legs until the reference reaches the desired operating point. The programmable reference utilizes the selection switches to ensure a correct start-up condition by connecting the drains and the coupling voltage of the floating gate transistors to a low potential, which causes current to flow. A power-on-reset circuit is used to keep the circuit in program mode for a short period of time before switching to run mode.

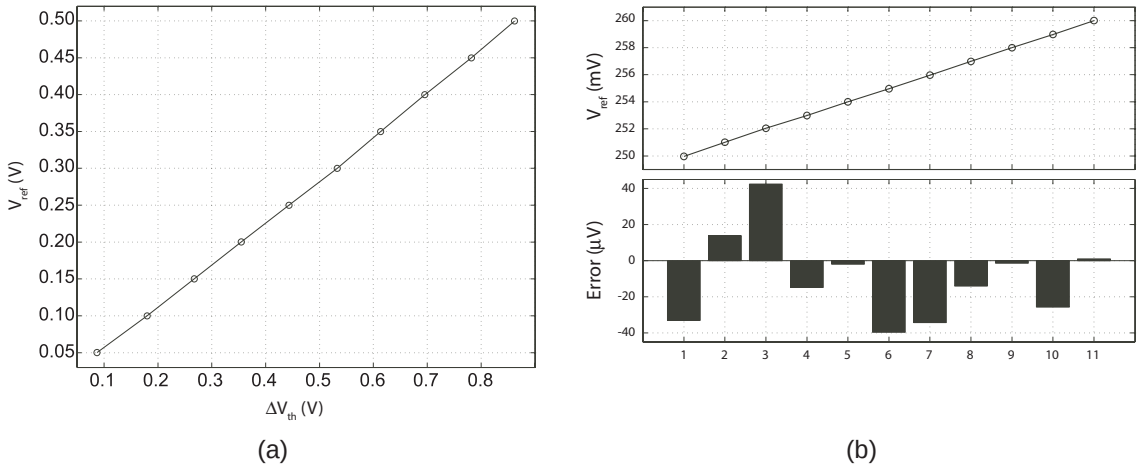


Figure 3.3. Reference programmability and accuracy.
(a) The voltage reference is a linear function of the threshold difference.
(b) Initial accuracy of programmed reference voltages.

Figure 3.3a shows the reference circuit programmed to output values between 50 mV and 500 mV. The charge difference is estimated as the difference of the effective threshold voltages of M_1 and M_2. As predicted, the reference voltage is linearly proportional to the charge difference. Taking a closer examination of the programming, Figure 3.3b shows the reference programmed between 250 mV and 260 mV and the initial accuracy observed. The worst case initial offset was measured to be approximately 40 μV. However, a better characterized programming algorithm should be capable of even better.

3.3 Temperature Dependence

From (3.5) the dominant temperature dependence of this circuit is primarily due to the κ term, which is expanded in (3.6).

$$\kappa = 1 - \frac{\gamma}{2\sqrt{V_{FG} - V_{th} + \left(\frac{\gamma}{2+\sqrt{\phi_0}}\right)^2}} \tag{3.6}$$

The various parameters of (3.6) are given by (3.7) through (3.10).

$$V_{th} = V_{t0} + \alpha\,(T - T_0) \tag{3.7}$$

$$U_T = 8.62 \cdot 10^{-5}\,T \tag{3.8}$$

$$\phi_0 = -2U_T\,ln\left(\frac{n_i}{N_A}\right) \tag{3.9}$$

$$n_i = 3.1 \cdot 10^{16}\,T^{\,3/2} e^{-7000/T} \tag{3.10}$$

The capacitance will also have some dependence upon temperature due to the physical expansion and contraction of the oxide, but this should be relatively low compared to that of κ.

Figure 3.4a shows measured temperature data between -60° C and 140° C for six different reference voltages between 100 mV and 600 mV. Figure 3.4b shows a detailed view of the temperature data for a reference voltage of 400 mV. From this data, a maximum temperature dependence of 110 μV/°C was observed with the reference programmed to 600 mV, and a minimum of 10 μV/°C was measured for the 100 mV case.

The first-order temperature dependence observed in the reference circuit could have been significantly reduced by connecting the source and bulk of M_1, assuming it resides in

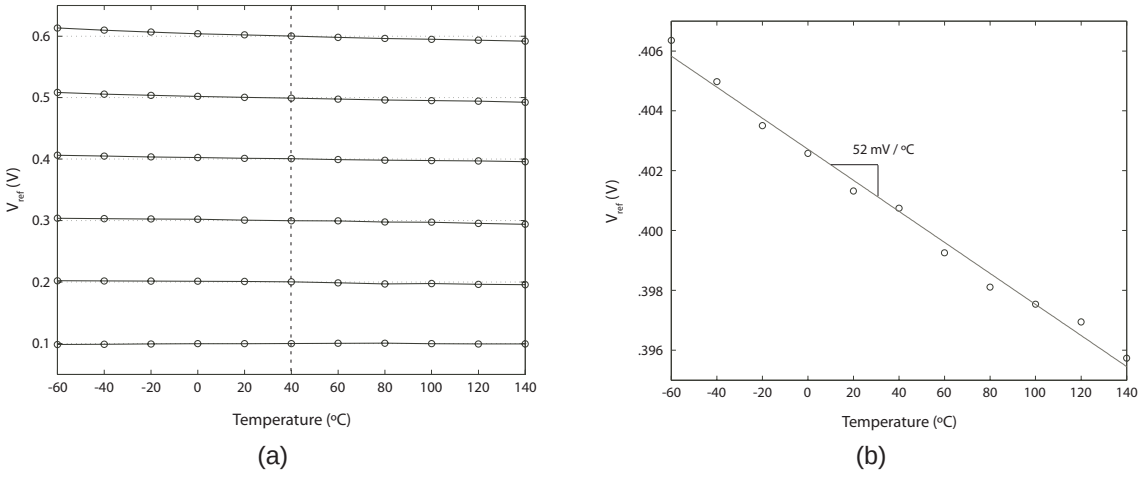


Figure 3.4. Reference voltage change as a function of temperature.
(a) Reference circuit programmed to several output voltages.
(b) A closer view of a single programmed voltage.

its own well, as seen in the following sub-threshold derivation.

$$\begin{aligned} I_2 &= I_1 \\ I_0\, e^{\left(\frac{\kappa V_{FG2}}{U_T}\right)} &= I_0\, e^{\left(\frac{\kappa V_{FG1}}{U_T}\right)} \\ V_{FG2} &= V_{FG1} \end{aligned} \tag{3.11}$$

As expected, the effective floating gate voltages referenced to the bulk of the transistor are equal (3.11). Substituting (2.2a) into (3.11) results in (3.12).

$$\frac{C_C V_C + C_{tun} V_{tun} + Q_2}{C_T} = \frac{C_C\left(V_C - V_{ref}\right) + C_{tun}\left(V_{tun} - V_{ref}\right) + Q_1}{C_T} \tag{3.12}$$

Since the bulk potential for transistors M_1 and M_2 are different in this case, the coupling voltage V_C and V_{tun} have been referred to the bulk of M_2 in (3.12). The coupling terms of M_1 have been adjusted to accommodate the difference between the bulk terminals, which is the reference voltage. Simplifying (3.12) results in (3.13).

$$\left(C_C + C_{tun}\right) V_{ref} = Q_1 - Q_2 \tag{3.13}$$

This equation can be further simplified by recognizing that the total capacitance at the floating node is a summation of the coupling capacitor and tunneling capacitor in the simplified case, which results in (3.14).

$$\begin{aligned} V_{ref} &= \frac{Q_1 - Q_2}{C_T} \\ &= \frac{\Delta Q}{C_T} \end{aligned} \tag{3.14}$$

Thus the reference voltage is only dependent upon the charge difference and the total capacitance seen at the floating node. The charge is not directly dependent upon temperature, although it can affect the long-term drift as discussed in the following section. Therefore, the temperature dependence of this modified circuit is primarily due to the temperature coefficient of the capacitors.

3.4 Long-Term Retention

From (3.5) it is easy to see that the long term drift of the reference voltage is proportional to the change in the charge difference. The change in charge over time is believe to be primarily due to thermionic emission [36]. The fractional change in the charge difference, and therefore the fractional change in the reference voltage, can be expressed as a function of time and temperature as seen in (3.15).

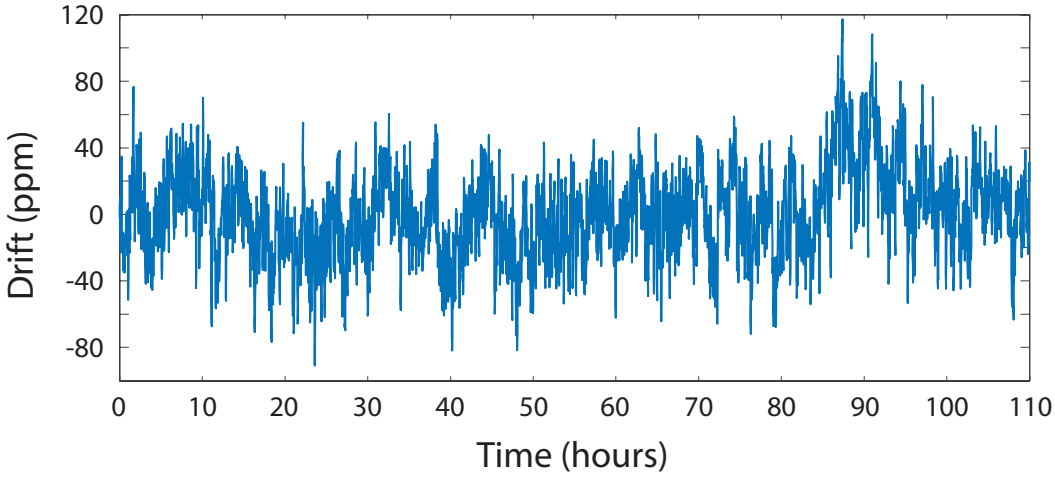
$$\frac{V_{ref}(t)}{V_{ref}(0)} = \frac{\Delta Q(t)}{\Delta Q(0)} = e^{-tve^{\frac{-\phi_B}{kT}}} \tag{3.15}$$

$V_{ref}(0)$ is the initial programmed reference voltage, and $V_{ref}(t)$ is the reference voltage at time t. Likewise, ΔQ(0) is the initial charge difference, and ΔQ(t) is the charge difference at time t. υ is the relaxation frequency of electrons in polysilicon, ϕ_B is the S_i / S_iO_2 barrier potential, k is Boltzmann's constant, and T is the absolute temperature in Kelvin.

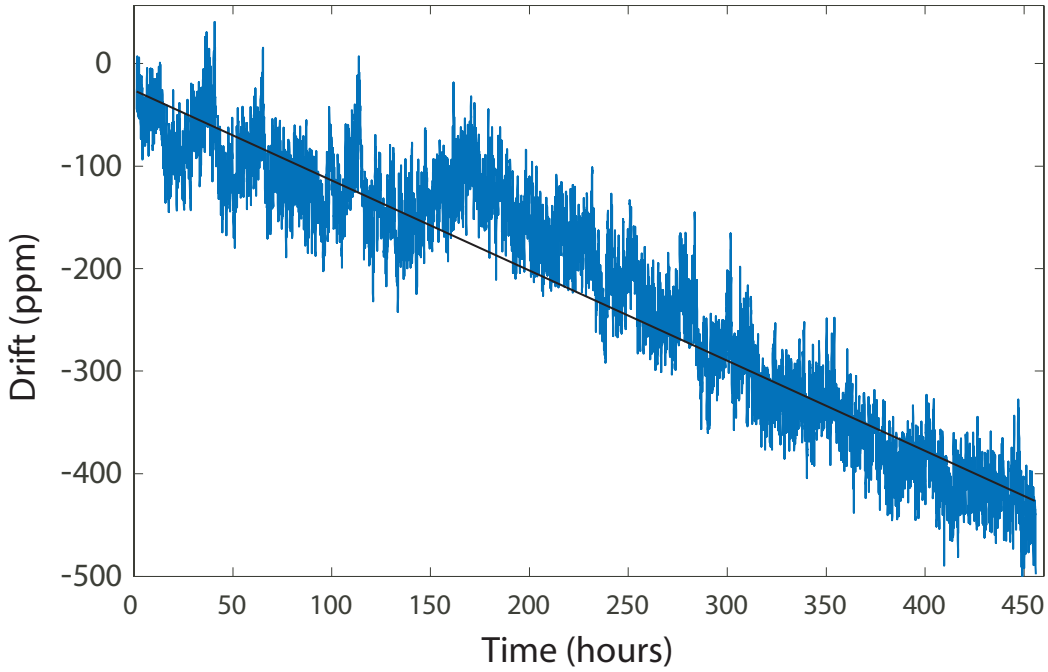
Accelerated lifetime retention data was measured according to the method previously used in [16] and is summarized in Table 3.1. Figure 3.5a shows negligible drift in the reference voltage for over 100 hours at a temperature of 25° C. At 125° C, the reference voltage changed by 400 μV over a period of 450 hours, as seen in Figure 3.5b. Using this data, the high temperature data from Table 3.1, and (3.15), υ and ϕ_B were found to be 55 m/s and .618 eV, respectively. This results in a 10 year drift of 400 μV using (3.15) at 25° C.

Table 3.1. Reference Voltage Drift Data

Temperature (°C)	325	325	125
Time (hours)	24	48	400
$\frac{V_{ref}(t)}{V_{ref}(0)}$	.967	.953	.998



(a) 25° C



(b) 125° C

Figure 3.5. Long term reference voltage drift at low and high temperatures.

CHAPTER 4

FIRST GENERATION FLOATING GATE FPAA

The RASP 1.x ICs were the first attempt at an FPAA based upon floating gate transistors[1]. The RASP 1.0 and RASP 1.5, as seen in Figure 4.1, were fabricated on 1.5 mm x 1.5 mm dies in a .5 μm process available through MOSIS. Before developing large-scale FPAAs, it was necessary to demonstrate the feasibility of using floating gate transistors as both the programmable and switching elements within an array structure. The RASP 1.x FPAAs therefore served as characterization chips for potentially larger devices. Although they are fairly small in size, the RASP 1.x FPAAs have nearly the same functionality as commercially available FPAA ICs [6, 23].

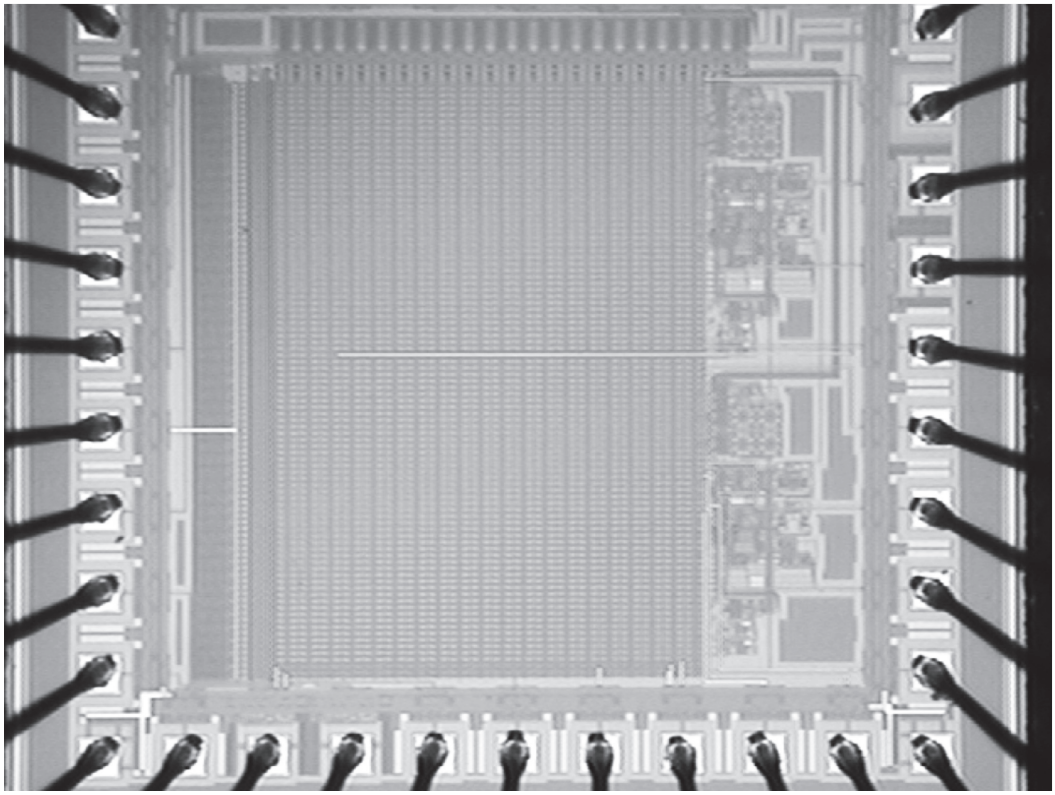


Figure 4.1. RASP 1.5 die photograph.

[1]This work was done in collaboration with Tyson S. Hall and Jordan D. Gray

4.1 Architecture

As seen in Figure 4.2, the RASP 1.x FPAA [23] is composed of two vertically aligned general purpose configurable analog blocks (CABs) connected via a single crossbar switching network. This switch matrix (SM) allows any CAB component in either CAB to be connected to any other CAB component with just two switches. Input / Output (I/O) pins connect directly to several of the rows in the switch network of the RASP 1.0. Some I/O lines also contain dedicated output buffers for driving signals off chip. The later revision of the chip, RASP 1.5, also included I/O pin connections to some of the column routing lines. This provided another level of I/O that had half the routing impedance of the original IC's I/O lines, since it requires only one switch to make a connection instead of two.

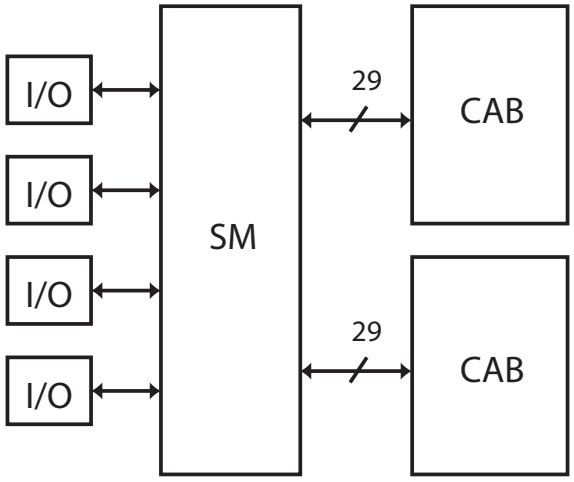


Figure 4.2. RASP 1.x FPAA architecture and CAB components.

4.1.1 CAB Component Selection

In the case of most FPGAs, the core elements used to synthesize digital circuits are look-up tables (LUTs) and D-type flip-flops (DFFs). Using a cascade of asynchronous LUTs, any combinational logic chain can be implemented. The addition of DFFs enable a wide variety of synchronous circuits, such as simple state machines and soft core processors. Since these two components are rather simple, they can be arrayed in a regular manner to create a large-scale reconfigurable and programmable digital device.

Determining a core set of components for reconfigurable analog ICs, such as FPAAs, is somewhat more difficult. Some FPAAs, such as the field programmable transistor array [37], utilize fine-grain components such as transistors and capacitors as the core elements. This is very advantageous from the perspective of flexibility, since any circuit can be built from a large enough array of transistors. However, this flexibility comes at the cost of performance. By building the circuit from these primitive elements, switch parasitics are introduced at every junction in the circuit. Figure 4.3 shows an example five transistor OTA topology and the corresponding circuit as synthesized within an FPAA using only transistor primitives. Floating gate transistor switches used to connect the individual CAB transistors are depicted in Figure 4.3b as light grey pFETs. Each of these switches adds a little impedance in the signal path and a little capacitance at each junction. To help improve performance, commonly used circuits, such as OTAs or OPAMPs, are also included

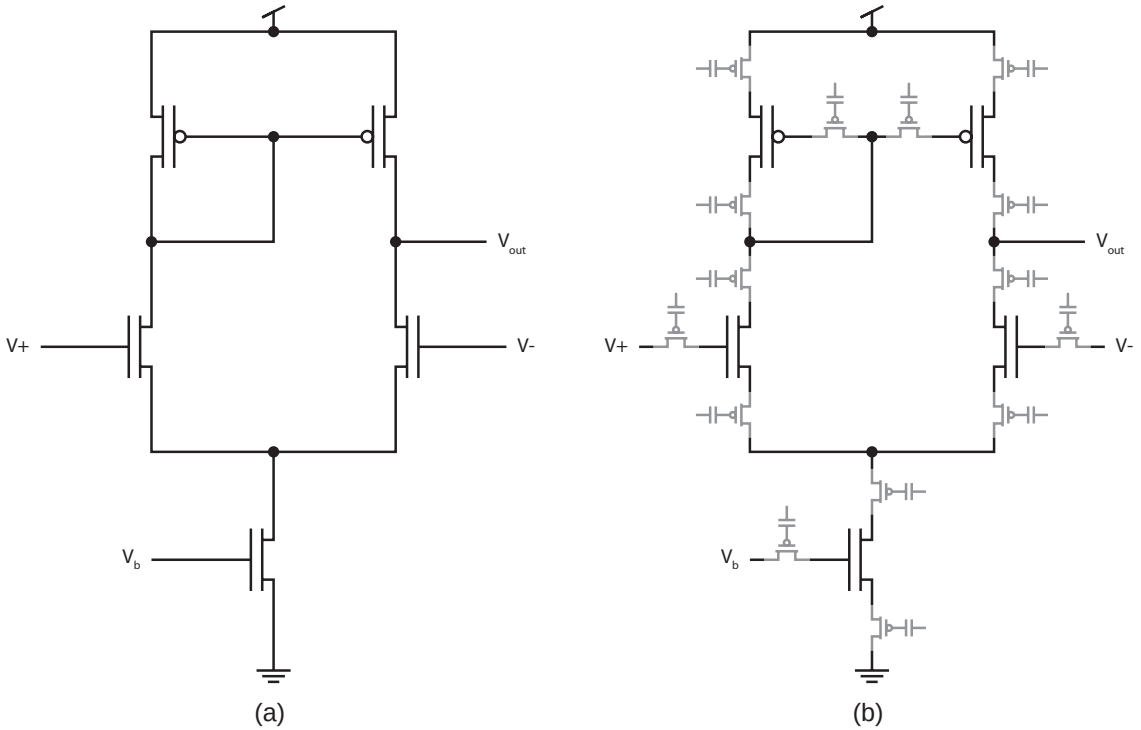


Figure 4.3. Example OTA implemented using transistors within an FPAA.

as medium-grain components. Medium-grain components provide improved circuit parameters without a significant loss of generality or flexibility. Finally, course-grain components are generally added as specialized circuits, which have been highly optimized to perform a specific task.

The CAB components, as seen in Figure 4.4, were chosen to provide a balanced mixture of granularity [6] in order to achieve an effective trade-off between performance and flexibility. The transistors and capacitors provide fine-grain flexibility, which allows almost any circuit to be synthesized with a sufficiently large CAB array. OTAs and C^4 band-pass elements were included as medium-grain components, since these elements can be used in a significant number of circuit topologies. Special purpose course-grain components, such as the min/max detectors and the vector-matrix multiplier (VMM) are also included.

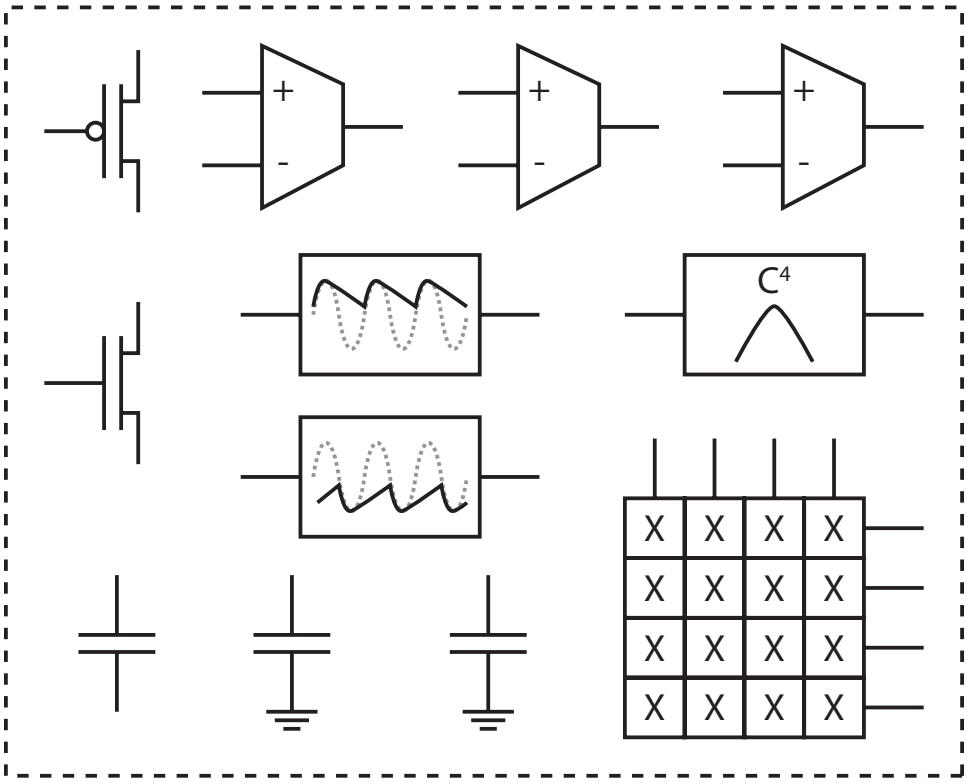


Figure 4.4. RASP 1.x FPAA CAB components.

4.1.2 Floating Gate Transistor Array Structure

Figure 4.5 shows the inner details of the FPAA architecture, which has two modes of operation, run and program. In program mode, indicated by the "prog" signal going high, the floating gate transistors used for switches and biases are configured into one large matrix for global addressing. As seen in Figure 4.5, pull-up transistors drive the sources of floating gate switch transistors, and drain lines are connected to the column programming logic. Bias transistors are disconnected from the circuits they control and are connected to

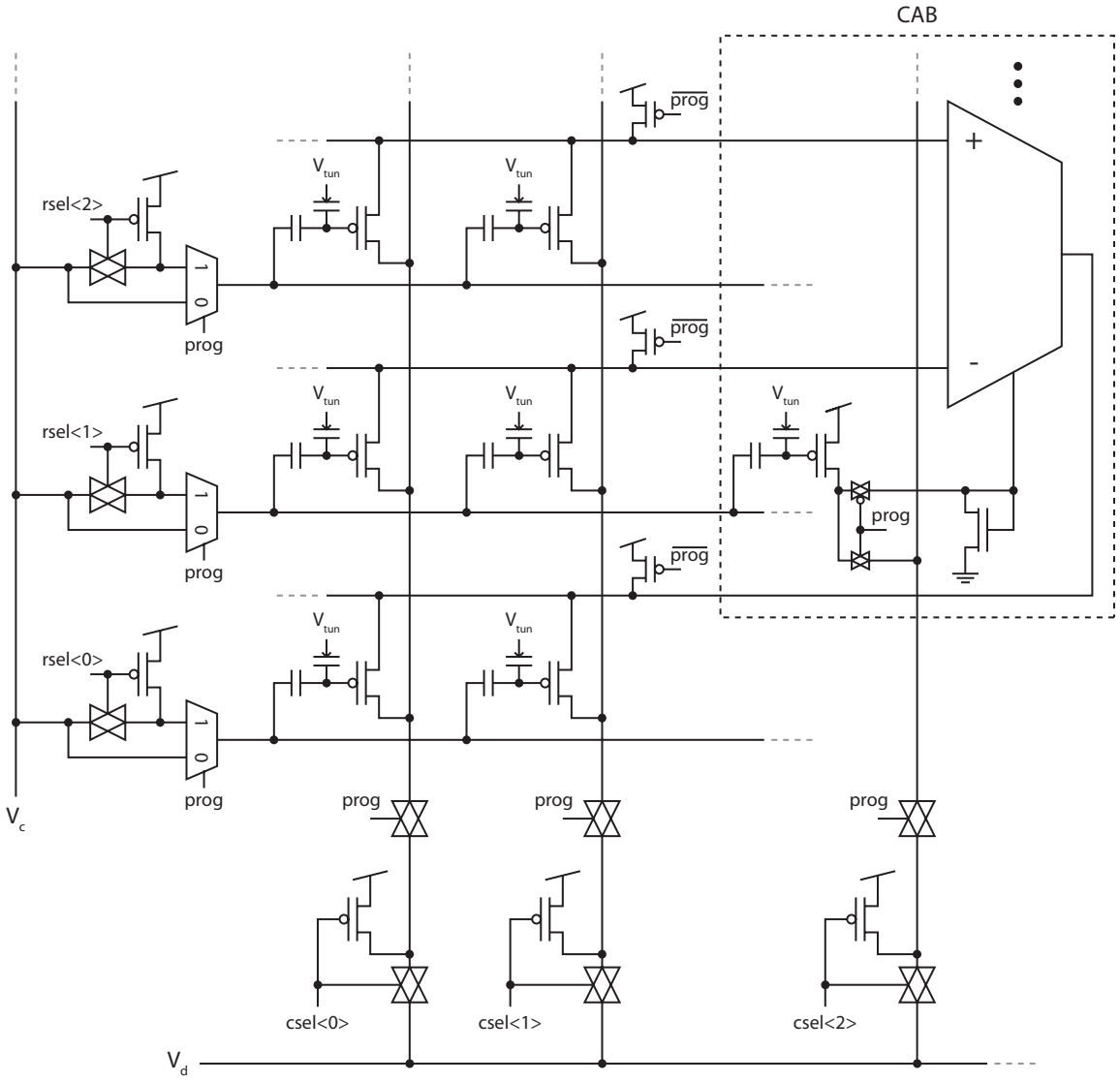


Figure 4.5. Floating gate transistor array architecture for programming.

the same programming lines used for the switches. The row selection circuitry, left side of Figure 4.5, is used to switch the external coupling voltage, V_C, to the selected row. All unselected rows have their coupling capacitors pulled up to V_{DD}. A decoder is used to generate the row select signals, rsel<x>. In a similar fashion, the column selection circuitry, bottom of Figure 4.5, connects the selected column's drain line to the external drain signal, V_D. All unselected drain lines are tied to V_{DD}. Another decoder generates the column select signals, csel<x>.

Run mode is defined when the "prog" signal in Figure 4.5 is low. In this configuration, the source and drain terminals of the floating gate switches are left floating as the rows and columns of the routing fabric, and the bias floating gate transistors are switched into their corresponding CAB components. In this example, the floating gate bias controls the tail current of an OTA. The current flowing through the transistor is set during program mode. A diode-connected nFET converts this current into a voltage that drives the bias transistor of the OTA. The terminals of the OTA connect to the routing rows. Likewise, all other CAB component terminals connect to rows of the routing matrix. By programming the switch transistors, connections can be made between these component terminals via the routing columns.

4.2 Synthesized Circuits and Results

Testing this first generation FPAA began by characterizing the floating gate transistors. Since this was the first time that floating gate transistors had been used as switches, not much was known about how to program them. Initial attempts used pulse width modulated schemes for programming both biases and switches. Although this worked fairly well for biases, switch conductivity was limited by standard array isolation techniques. These initial tests illustrated the need for a new programming scheme designed specifically for switches. The methods described in Chapter 2 were developed in response to this limitation and allowed for significantly better switches.

4.2.1 Follower, Low-Pass Filter

With the bias and switch programming characterized, the first circuit synthesized was a simple follower acting as a G_M-C low-pass filter, as seen in Figure 4.6a. The OTA is a nine-transistor topology with nFET inputs and a tail current set by a bias voltage. Figure 4.6a shows the floating gate pFET and diode-connected nFET that sets the OTA bias. The 3 dB corner frequency, (4.1), is directly determined by the OTA transconductance, which is determined by the bias current, and the load capacitor.

$$f_{3dB} = \frac{G_M}{2\pi C} \tag{4.1}$$

By programming the tail current of the OTA, the corner frequency of the circuit can be set.

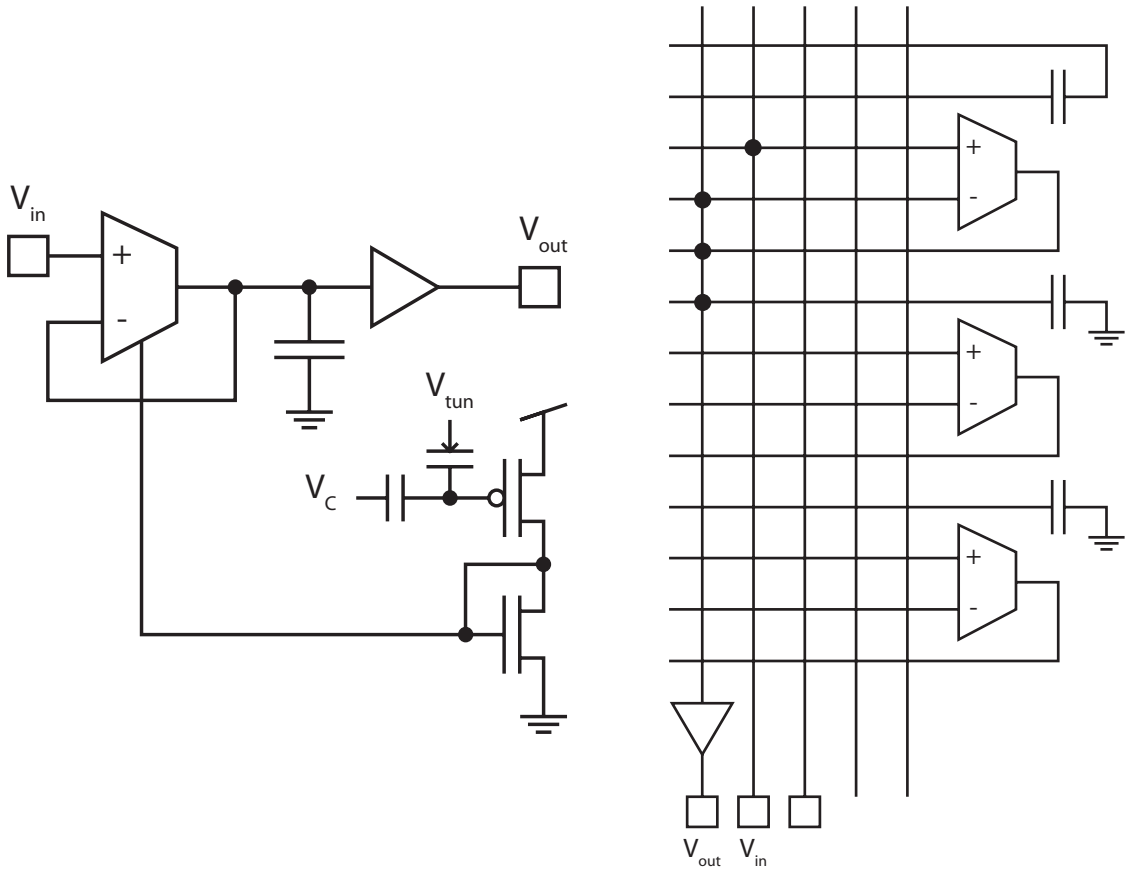


Figure 4.6. G_M-C low-pass filter implemented on the RASP 1.5.
(a) Low-pass filter circuit schematic.
(b) FPAA implementation showing "on" switches.

The follower circuit was routed in the RASP 1.5 as depicted in Figure 4.6b. As discussed earlier, the terminals of CAB components are connected to the rows of the switch matrix. Connections between CAB components are made by turning "on" the transistors, represented in this diagram by filled circles at the intersections of rows and columns. For this circuit, two column I/O lines were used for the input and output waveforms. The output of this circuit was isolated from external parasitic capacitances by using one of the column I/O lines with a dedicated output buffer.

The bias transistor, which is not shown in Figure 4.6b, was programmed and swept from 10 nA to 20 μA. For each bias current, a frequency response was measured, as seen in Figure 4.7. For the programmed current range, the 3 dB corder frequency range was between 700 Hz and 20 kHz, which nearly covers the spectrum typically used for audio applications.

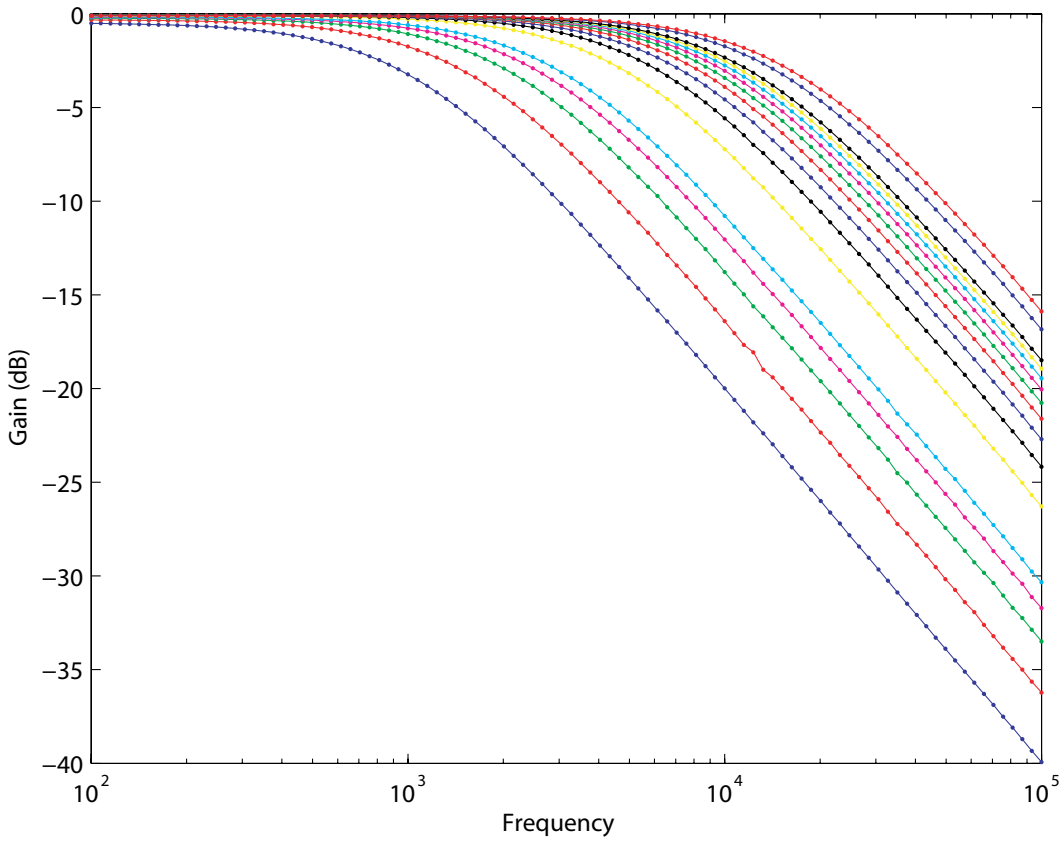


Figure 4.7. Frequency response of G_M-C low-pass filter.

4.2.2 Second-Order Section

A second-order section was next synthesized on the RASP 1.5, as seen in Figure 4.8a. The transfer function of this circuit topology is given by (4.2).

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{\omega_n^2}{s^2 + \frac{\omega_n}{Q}s + \omega_n^2} \tag{4.2}$$

The natural frequency of the circuit is determined by (4.3).

$$\omega_n = \frac{G_{M1}}{C} \tag{4.3}$$

The quality factor, Q, is given by (4.4).

$$Q = \frac{1}{2 - \frac{G_{M3}}{G_{M1}}} \tag{4.4}$$

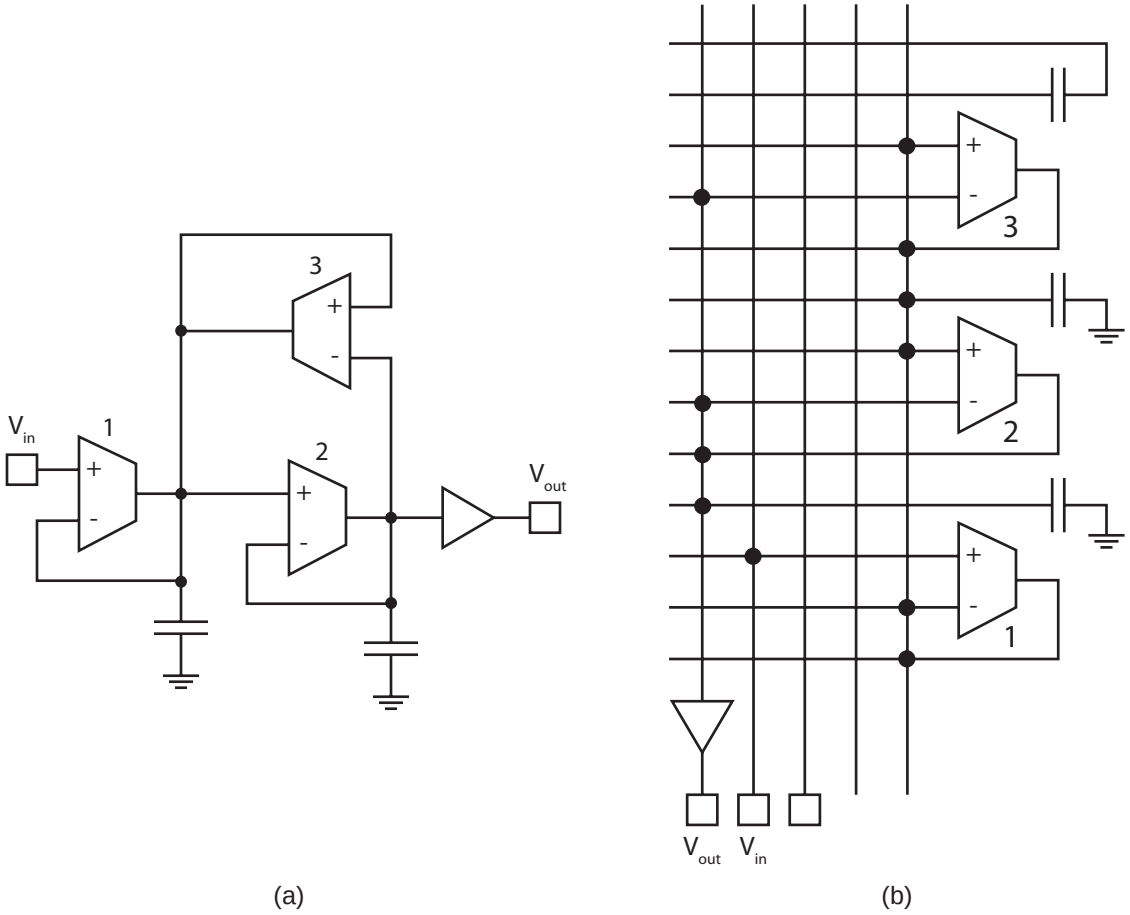


Figure 4.8. Second-order section implemented on the RASP 1.5.
(a) Circuit schematic.
(b) FPAA implementation showing "on" switches.

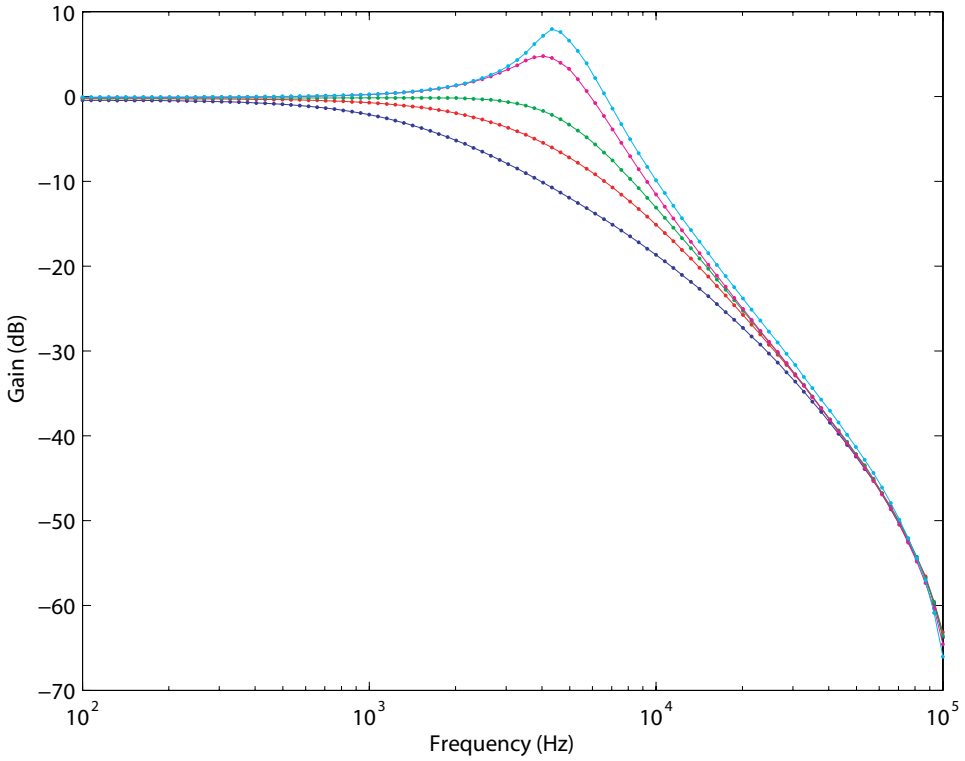


Figure 4.9. Frequency response of the second-order section circuit.

These equations assume that $G_{M2} = G_{M1}$ and the load capacitances are equal. Therefore, the bias currents of OTA 1 and 2 set the corner frequency, and the bias current of OTA 3 sets the Q of the circuit. Figure 4.8b shows the circuit routed within a single CAB of the RASP 1.5 using three OTAs and two drawn capacitors. Frequency response data was also taken for this circuit and is shown in Figure 4.9. The bias currents of OTAs 1 and 2 were programmed to approximately 100 pA, and the bias current was programmed to various values around 100 pA to change the Q.

4.2.3 Capacitively Coupled Current Conveyor

The C^4 CAB component is commonly used as a compact band-pass filter. As seen in Figure 4.10a, the C^4 is a 4-transistor circuit with capacitively coupled inputs. A 5^{th} transistor, which is biased with the circuit depicted in Figure 4.10b, was added to increase the input linear range using source degeneration [38]. A drain induced barrier lowering (DIBL) transistor was chosen for this 5^{th} transistor to provide a strong exponential relationship the

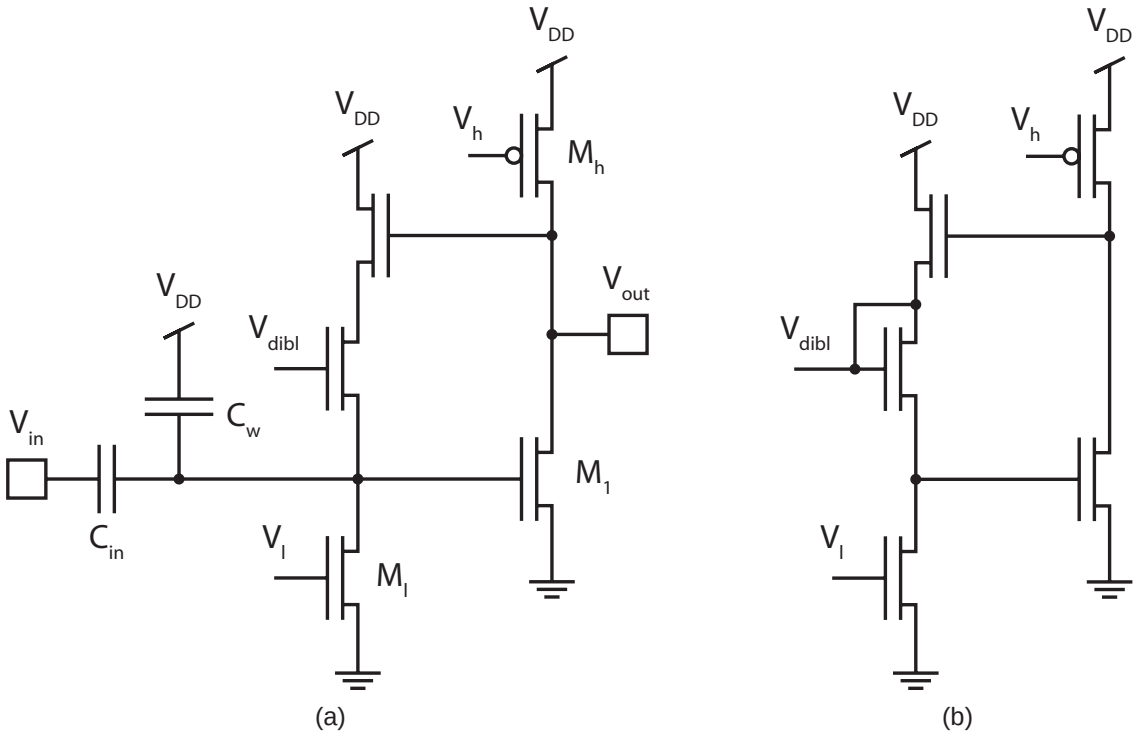


Figure 4.10. Capacitively coupled current conveyor (C^4) band-pass element.
(a) Schematic of C^4 circuit.
(b) Schematic of automatic DIBL bias generator.

drain potential and the current flowing through it. This is achieved by reducing the length of this transistor just below the minimum feature size of the process.

The transfer function of this C^4 circuit [38] is given by (4.8).

$$\frac{V_{out}(s)}{V_{in}(s)} = -\left(\frac{C_{in}}{C_{ov}}\right)\left[\frac{s\tau_l\,(1-s\tau_f)}{s^2\tau_l\tau_h + s\left[\tau_l + \tau_f\left(\frac{C_{ov}+C_L}{\kappa C_{ov}} - 1\right)\right] + 1}\right] \tag{4.5}$$

The time constants are defined by (4.6) through (4.5).

$$\tau_l = \frac{C_{ov}}{g_{Ml}} \tag{4.6}$$

$$\tau_h = \frac{C_{ov}}{g_{Mh}} \tag{4.7}$$

$$\tau_f = \left(\frac{C_T\,(C_{ov}+C_L) - C_{ov}^2}{C_{ov}\,g_{Mh}}\right) \tag{4.8}$$

where C_T is the total capacitance seen at the gate of M_1, C_L is the load capacitance seen at the output of the circuit, and C_{ov} is the overlap capacitance between the gate and drain of

M_1. The center frequency of the band-pass transfer function is then defined by (4.9).

$$\tau = \sqrt{\tau_l \tau_h} \tag{4.9}$$

The high and low corners are independently tunable by adjusting the V_h and V_l biases, respectively. This also allows the circuit to be used as an effective low-pass or high-pass filter by programming the opposite corner frequency outside the relevant frequency band.

The C^4 block used in the RASP 1.5 is actually a cascade of two C^4 elements, which yields a second-order section. Frequency response results for one of the C^4 elements was obtained, as seen in Figure 4.11. One C^4 element was programmed with a very wide bandwidth, which effectively nullifies its affect on the output in the observable band. This allows the center frequencies of the second C^4 element to be adjusted and observed independently, as seen in Figure 4.11. For each center frequency, the high and low bias corners were programmed such that they overlap at the desired center frequency.

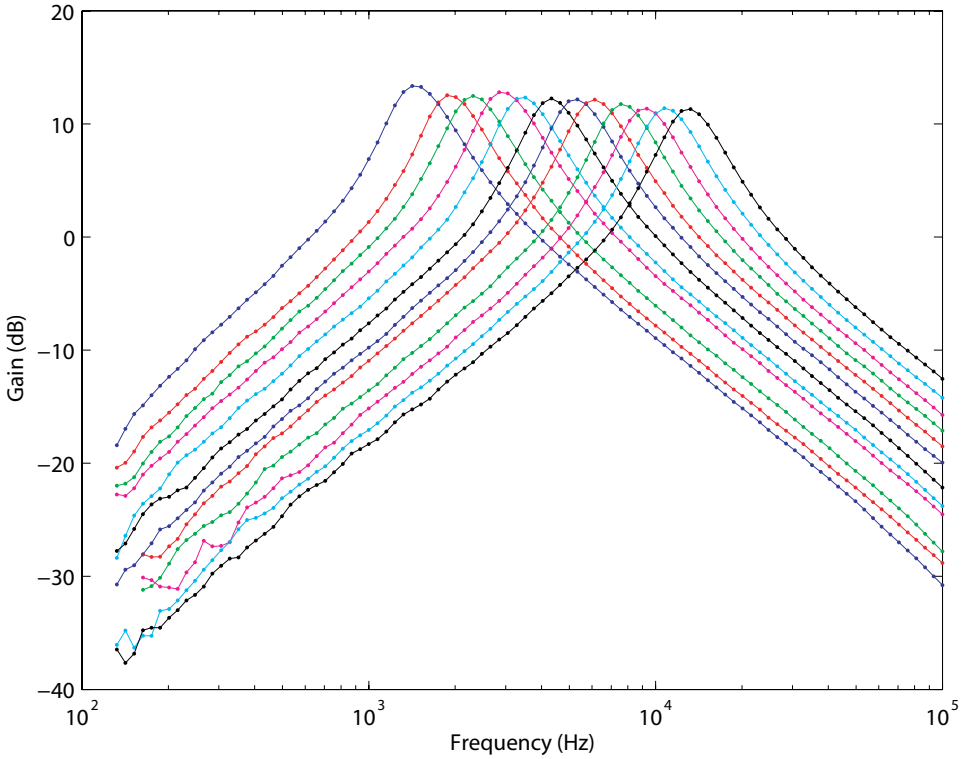


Figure 4.11. Frequency response of the C^4 band-pass element.

4.2.4 Third-Order Ladder Filter

A third-order ladder filter, as seen in Figure 4.12a, was also synthesized to demonstrate a multi-CAB circuit. The circuit was routed in the RASP 1.5 across two CABs, as seen in Figure 4.12b. This topology produces a low-pass third-order Butterworth filter, which was chosen for its sharp roll-off and flat passband region. The transfer function of this circuit is given by (4.10).

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{G_{M1}\, G_{M3}\, G_{M4}}{b_3\, s^3 + b_2\, s^2 + b_1\, s + b_0} \tag{4.10}$$

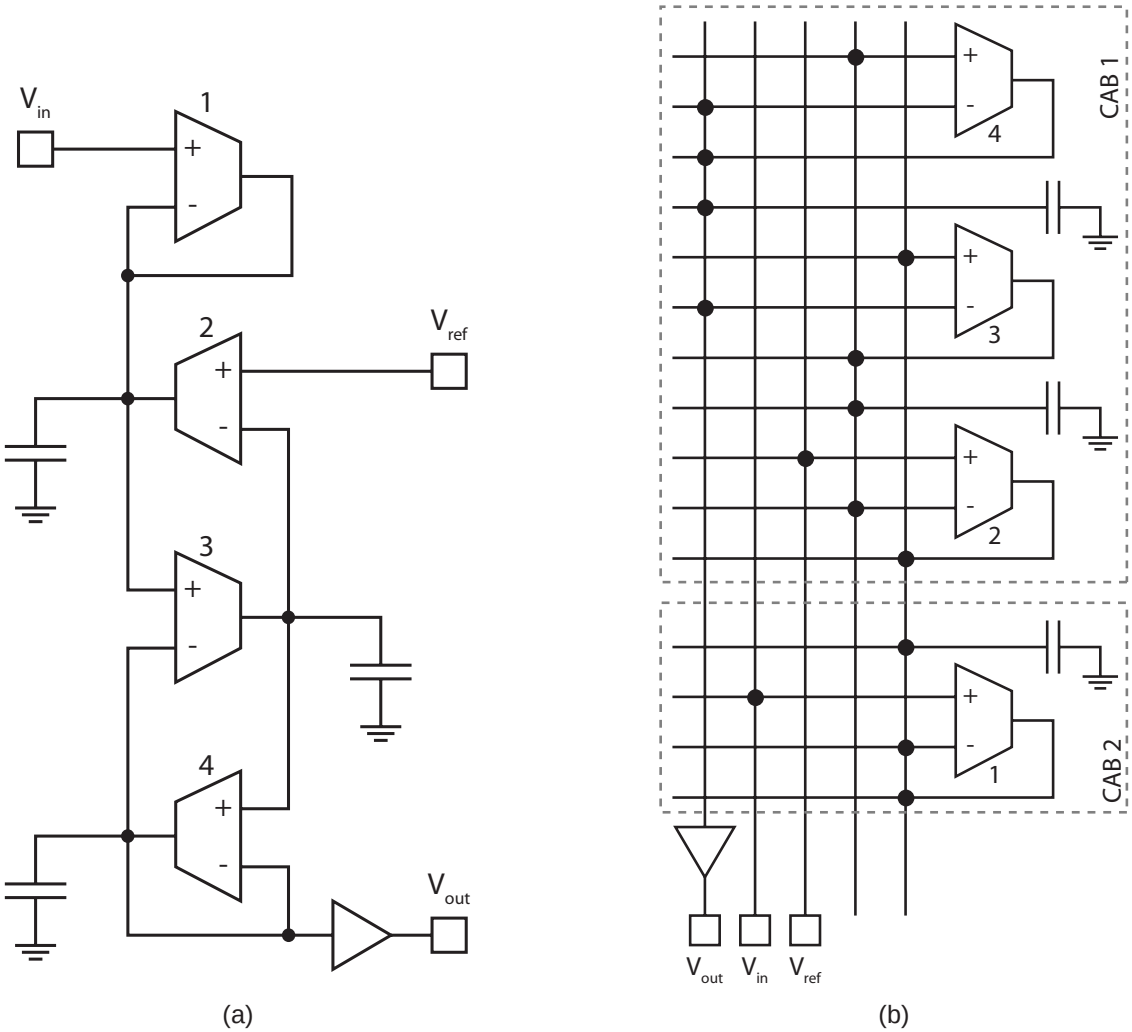


Figure 4.12. Third-order ladder circuit implemented on the RASP 1.5.
(a) Circuit schematic.
(b) FPAA implementation showing "on" switches.

The coefficients are defined by

$$\begin{aligned} b_3 &= C^3 \\ b_2 &= (G_{M1} + G_{M4})\, C^2 \\ b_1 &= (G_{M1}\, G_{M4} + G_{M2}\, G_{M3} + G_{M3}\, G_{M4})\, C \\ b_0 &= G_{M3}\, G_{M4}\, (G_{M1} + G_{M2}) \end{aligned}$$

where the transconductance, G_{Mx}, corresponds to OTA x and C is a single drawn capacitor. To meet the Butterworth filter conditions, the OTAs are biased such that $G_{M1} = G_{M2} = G_{M4} = 2\, G_{M3}$. With these constraints, (4.10) simplifies to (4.11), where $G_M = G_{M1}$.

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{G_M^3}{2\, C^3\, s^3 + 4\, G_M\, s^2 + 5\, G_M^2 C\, s + 2\, G_M^3} \tag{4.11}$$

Frequency response data was also obtained for several corner frequencies, as seen in Figure 4.13.

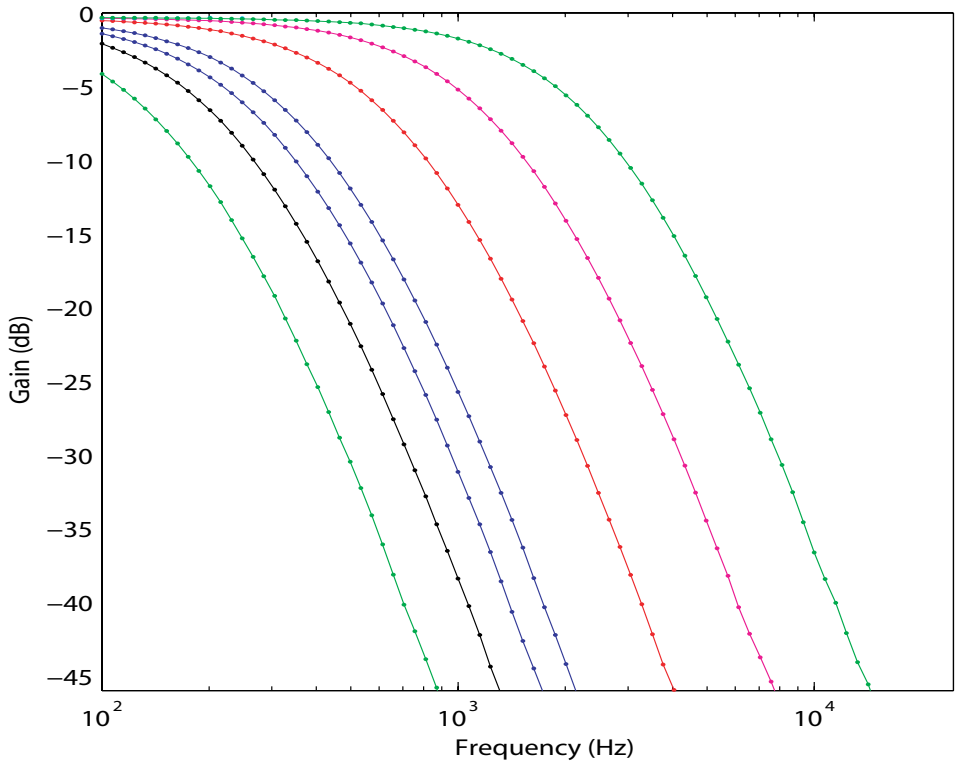


Figure 4.13. Frequency response of the third-order ladder circuit.

4.3 Observations and Conclusions

The results of testing the RASP 1.0 and RASP 1.5 demonstrated the feasibility of using floating gate transistors as both the programmable and reconfigurable elements within an FPAA. A single floating gate pFET switch performed as well as a standard transmission gate of comparable size, and required no additional memory elements, such as SRAM, to maintain connection information. As programmable elements, floating gate biases provided a method for setting analog circuit parameters over several orders of magnitude without requiring a significant amount of die area. With both programmable and reconfigurable elements tested, it seemed likely that a large-scale FPAA based upon this technology would be possible.

CHAPTER 5

SECOND GENERATION FLOATING GATE FPAA

The RASP 2.x ICs[1] are the first large-scale FPAAs based upon floating gate transistors, and the only reconfigurable analog device comparable in scale is the analog coprocessor discussed in [39]. These ICs were all fabricated in .35 μm processes available through MOSIS. The RASP 2.5 contains 56 CABs and fills a 3 mm x 3 mm die. The RASP 2.7 occupies a 3 mm x 4.5 mm die and consists of 72 CABs, as seen in Figure 5.1. Although the results of the RASP 1.x line demonstrated the feasibility of creating dense analog components, the RASP 2.x FPAAs would illustrate the issues involved in programming and using components located across the chip in a synthesized analog system.

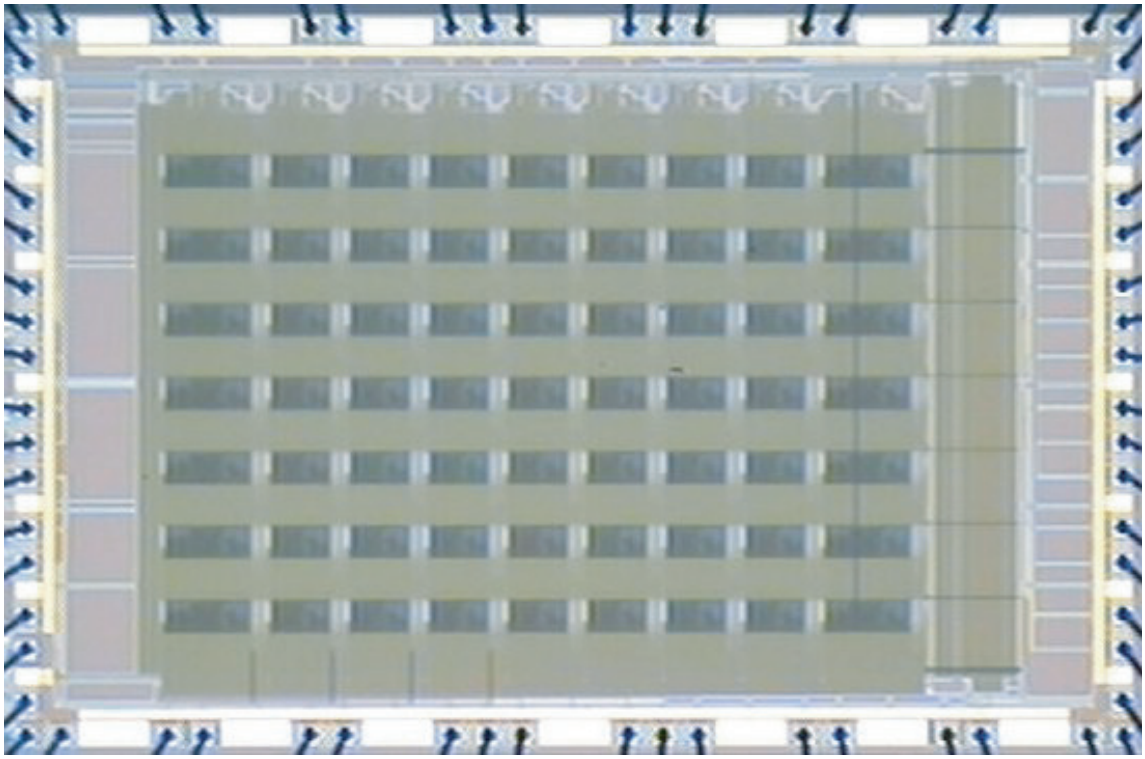


Figure 5.1. RASP 2.7 die photograph.

[1]This work was done in collaboration with Tyson S. Hall and Jordan D. Gray

5.1 Architecture

The RASP 2.x line of floating gate FPAAs is composed of a two-dimensional array of CABs, as seen in Figure 5.2. Instead of a single crossbar network connecting the CABs, there are now multiple routing options. The first layer of routing consists of the local crossbar switch matrix, which allows CAB component terminals to be connected within a single CAB. This switch matrix also connects to vertical global routing lines, which connect all of the local switch matrices along a column. Global horizontal routing lines are located just below each CAB row and are connected to the vertical global routing lines via a smaller switch matrix located at the intersection of the global horizontal and vertical routing lines. These global routing lines make up the second routing layer, which also connects to the I/O pins. Level 1 I/O pins, those that require only a single switch to connect to a CAB component terminal, are connected to select vertical global routing lines. Level 2 I/O pins, which require two switches to reach CAB component terminals, are connected to select horizontal global routing lines.

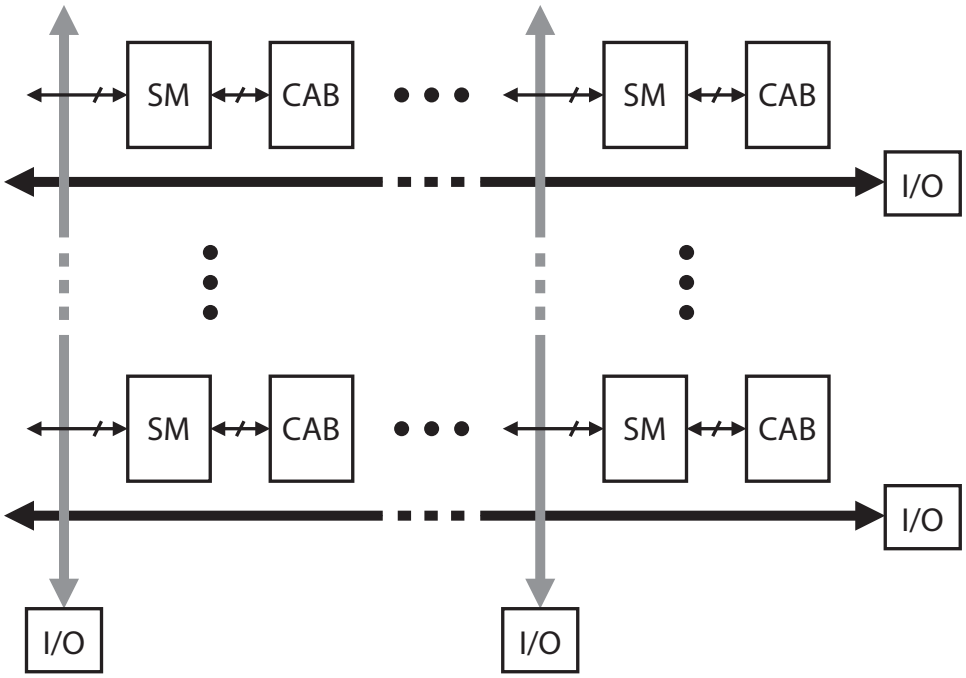


Figure 5.2. The two-dimensional CAB array, RASP 2.x FPAA architecture.

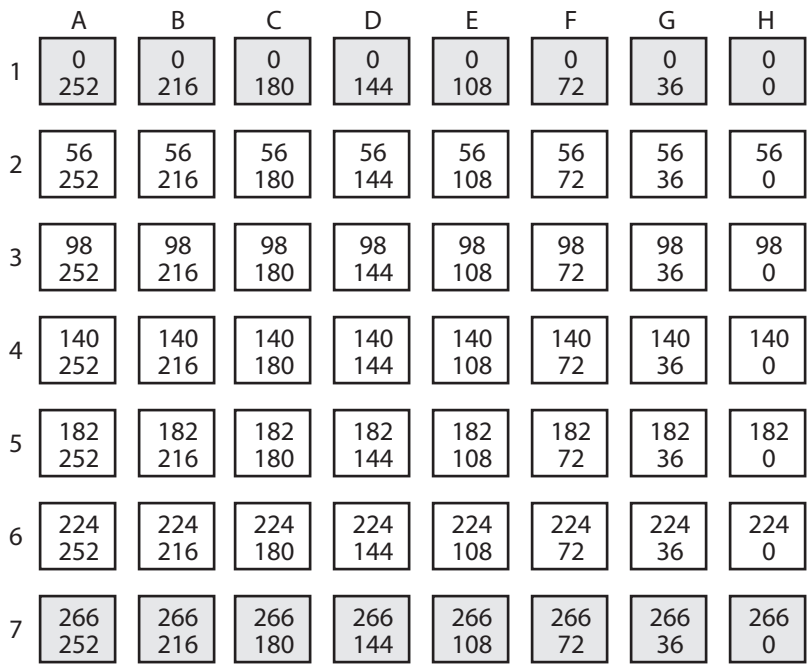


Figure 5.3. The RASP 2.5 CAB array with row and column addressing offsets.

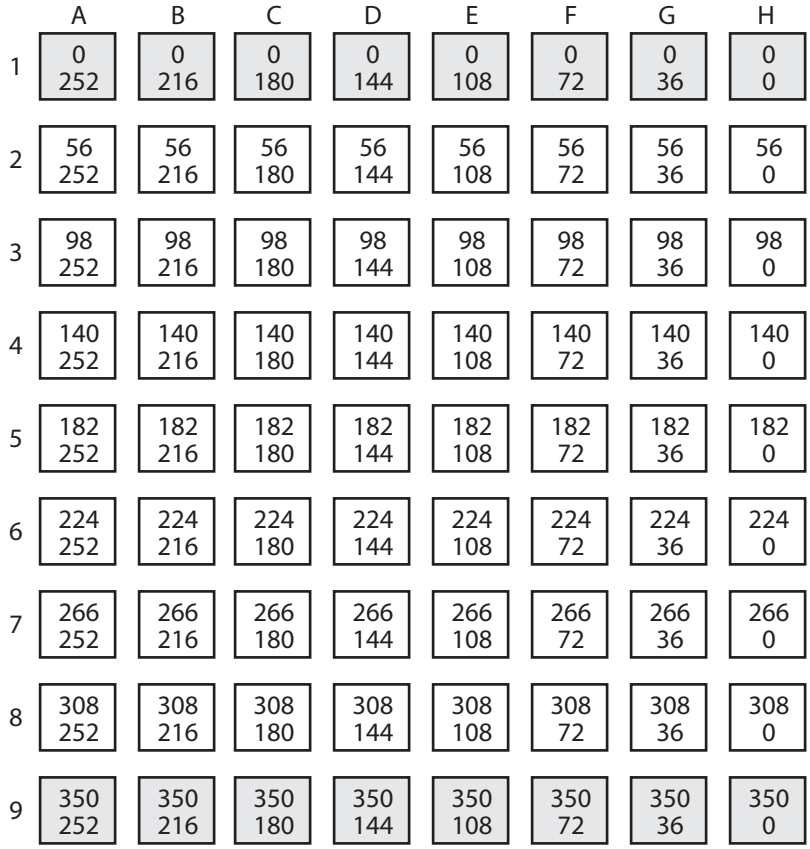


Figure 5.4. The RASP 2.7 CAB array with row and column addressing offsets.

The RASP 2.x FPAAs feature multiple CAB types, since some components are used less often than others for typical circuits. The VMM CAB is the same as in Figure 4.4, and the general purpose CAB contains all of the components in the VMM CAB except for the vector-matrix multiplier. For both the RASP 2.5 and 2.7, the top and bottom rows of CABs consist of VMM CABs, while the remaining CABs are general purpose. As depicted in Figure 5.3, the RASP 2.5 has a 7 x 8 array of CABs with the VMM CABs highlighted by a grey background. The RASP 2.7 contains a 9 x 8 array of CABs, as seen in Figure 5.4.

Each floating gate transistor within the FPAA is addressed using a global addressing scheme, which gives each transistor a unique row and column address. For routing purposes, it is often useful to refer the floating gate transistors within a local switch matrix to the CAB to which it is attached. For this purpose, each CAB within the array is given the address of the first floating gate transistor contained within the CAB, as seen in Figures 5.3 and 5.4. Using this CAB address, the floating gate transistors are given offset addresses based upon the routing lines, as seen in Figure 5.5.

With proper CAD tools, routing would be performed by software based upon some description of the hardware, whether schematics or HDL. However, the tools for reconfigurable and programmable hardware are still in early development, so much of the routing has been performed manually. Figure 5.5 depicts a condensed version of the RASP 2.x switch plot diagram, similar to fuse plots used in the early days of programmable digital devices. Connections between routing lines are graphically illustrated by drawing bubbles over the intersections, similar to that seen earlier in Figure 4.6b. The switch locations are then determined by adding the CAB address to the routing line offsets. For example, a simple buffer circuit has been routed using CAB A2 in Figure 5.5. To connect an I/O pin to the positive OTA terminal, switch (56 + 12, 252 + 23) is selected to be turned “on” during programming. The CAB offset (56, 252) was added to the switch’s relative position within the CAB, (12, 23). Similarly, the second I/O pin is connected to the OTA’s negative and output terminals via switches (56 + 13, 252 + 22) and (56 + 14, 252 + 22).

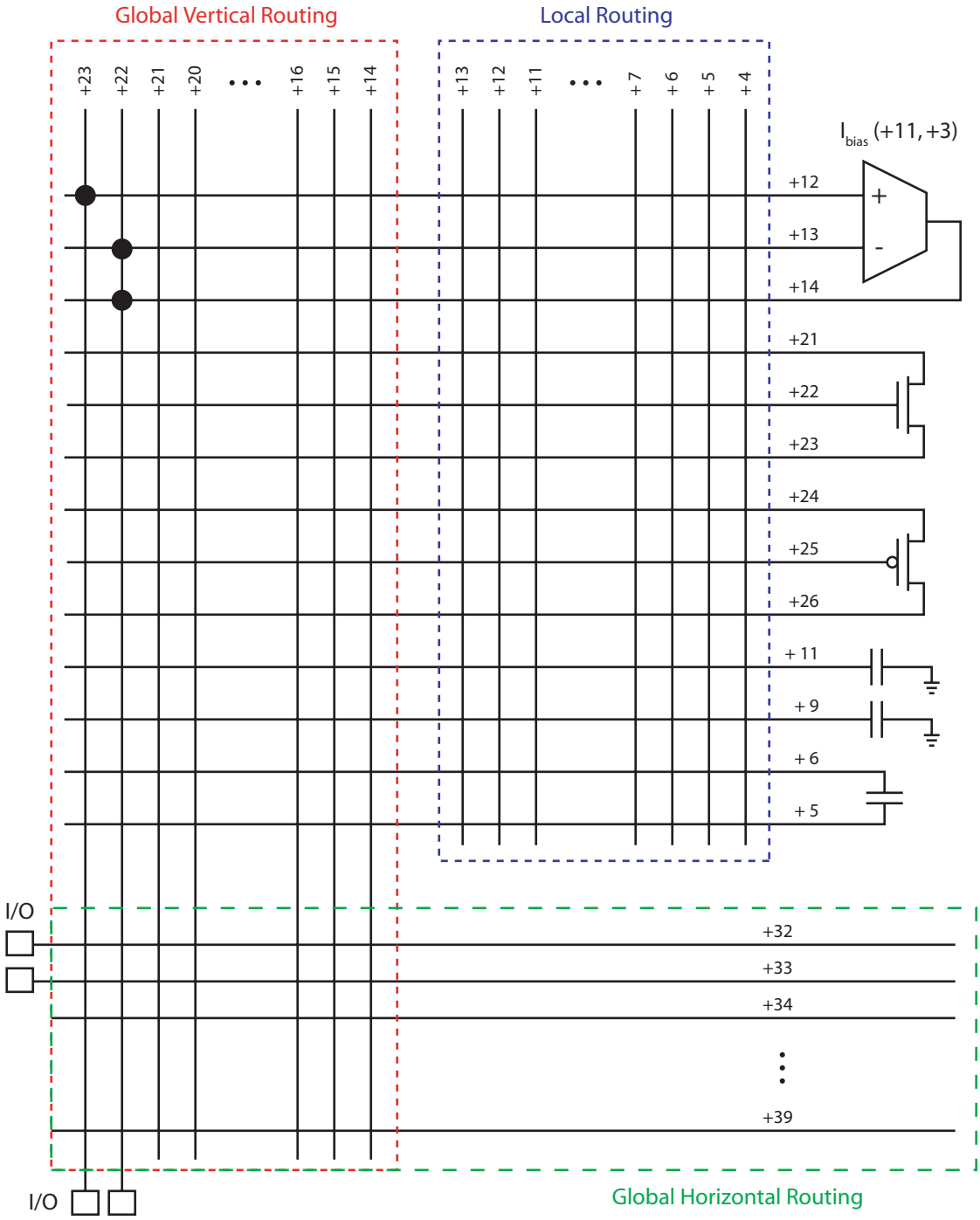


Figure 5.5. Switch plot diagram used to map circuits to FPAA CABs. For clarity, only selected CAB components are shown.

5.2 Characterization

As was the case with the first generation floating gate FPAA, testing began by characterizing the floating gate transistors in this process. Switch programming was achieved by utilizing the methods described in Chapter 2. A modified version of the algorithm described in [28] was used to characterize and program biases. During the characterization and programming steps, the coupling voltage was also modulated in an attempt to increase injection efficiency by biasing the floating gate pFET to a specific point during the drain pulse phase. Although this was marginally successful, it is not clear that this method is better than that described in [28]. Injection efficiency was increased, but accuracy decreased, most likely because of the additional approximation made while biasing the coupling voltage. A better solution would be to use gate modulation during initial pulses to quickly program the pFET close to the desired value. Then the programming scheme of [28] could be used to accurately tweak the pFET to the exact value. However, this significantly complicates the circuitry required to implement on-chip programming and may prove more difficult than beneficial.

Besides the floating gate transistors, the drawn CAB capacitors and parasitic routing capacitances were also characterized. Figure 5.6a depicts the circuit used to accomplish this task. OTA 2 in combination with the drawn and parasitic capacitances form a G_M-C element, which has a time constant proportional to the load capacitance. OTA 1 is used simply as a buffer to isolate the global routing capacitance from OTA 2. The black bubbles indicate the initial connections made, and the grey bubbles represent incremental connections made to observe the various capacitive load conditions. OTA 2 was biased with a small sub-threshold current, and OTA 1 was biased with a significantly larger current such that it would not affect the time constant of OTA 2.

Select step responses for this circuit are shown in Figure 5.6b to illustrate the effect of adding additional routing and drawn capacitance to the output of OTA 2. From these step responses and others, the individual capacitances were extracted, as summarized in

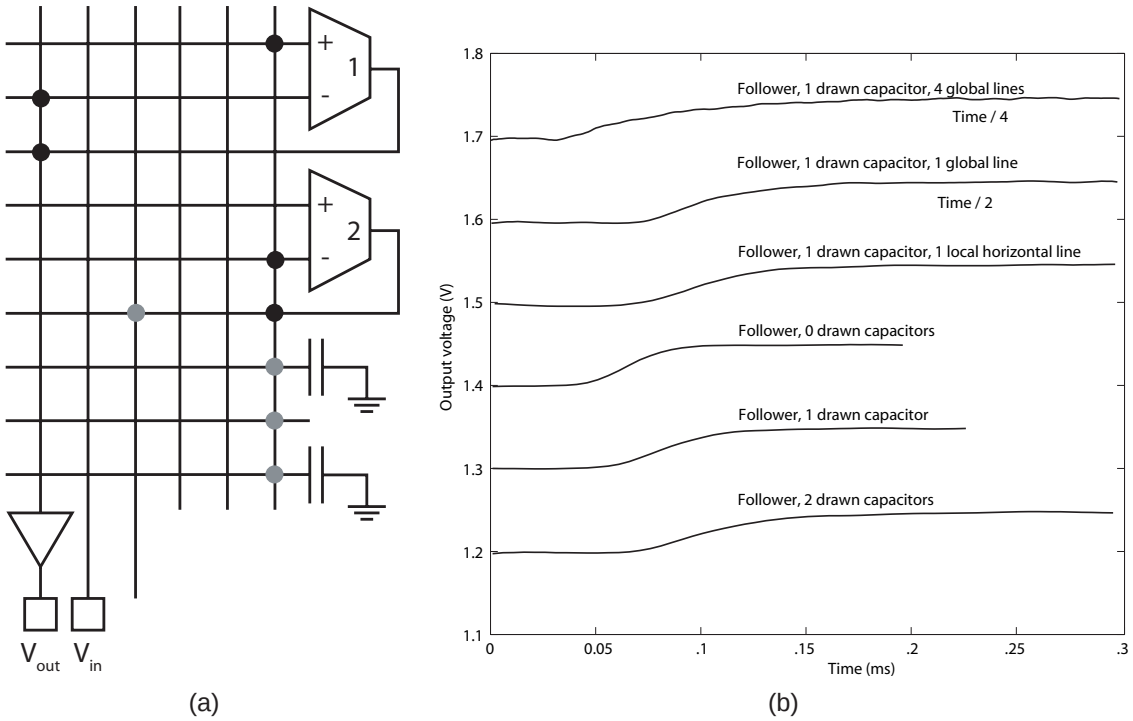


Figure 5.6. Characterization of drawn capacitors and parasitic routing capacitance.
(a) Circuit used to characterize capacitance.
(b) Step response data for various configurations. DC levels were shifted for clarity.

Table 5.1. From the measured routing capacitance values, the parasitic capacitance contribution of a single switch transistor was also estimated.

Table 5.1. Extracted Parasitic and Drawn Capacitances.

Global line	640 fF
Drawn Capacitor	130 fF
Local Line	75 fF
Closed Switch	10 fF
Open Switch	2.5 fF

5.3 Synthesized Circuits and Results

After the various characterization procedures were completed, many of the same circuits synthesized during the RASP 1.x testing were also analyzed on the RASP 2.x FPAAs, such as the follower circuit. However, additional capabilities were also explored when desired circuit elements were not available within the CABs. Capacitively coupled circuits are

often desirable for linear voltage operations. Unfortunately, none of these devices were included within the CAB, but they can be synthesized with some success. Another interesting area involves computational uses for switch fabric transistors. In most reconfigurable devices, the routing switches are mostly wasted space as far as signal processing. However, the floating gate switch provides a unique opportunity to utilize these transistors within circuit topologies, as will be discussed in the following sections.

5.3.1 Follower, Low-Pass Filter

The low-pass follower circuit design was migrated from the RASP 1.5 to the RASP 2.7 and synthesized in the same manner as depicted in Figure 4.6b. Instead of using the drawn capacitor, the load capacitance was dominated by the parasitic routing capacitance. In these large-scale device, the routing capacitance was significantly larger than in the RASP 1.x case, as would be expected. However, the drawn capacitor in the RASP 2.x devices is significantly smaller than the parasitic capacitance of the global routing lines. Figure 5.7 shows the results of using this parasitic capacitance of a single global routing line as the load capacitor of the G_M-C low-pass filter.

The corner frequency was programmed by adjusting the OTA transconductance according to (4.1). For sub-threshold bias currents, the transconductance is defined by (5.1).

$$G_M = \frac{\kappa I_{bias}}{U_T} \tag{5.1}$$

Substituting (5.1) into (4.1) yields a proportional relationship between the programmed OTA bias current and the 3 dB corner frequency, (5.2).

$$f_{3dB} = \frac{\kappa I_{bias}}{2\pi C U_T} \tag{5.2}$$

The corner frequency was extracted for each bias current in Figure 5.7 and plotted against the programmed bias current, as seen in Figure 5.8. A linear fit is also plotted in Figure 5.8 to show the conformance to (5.2).

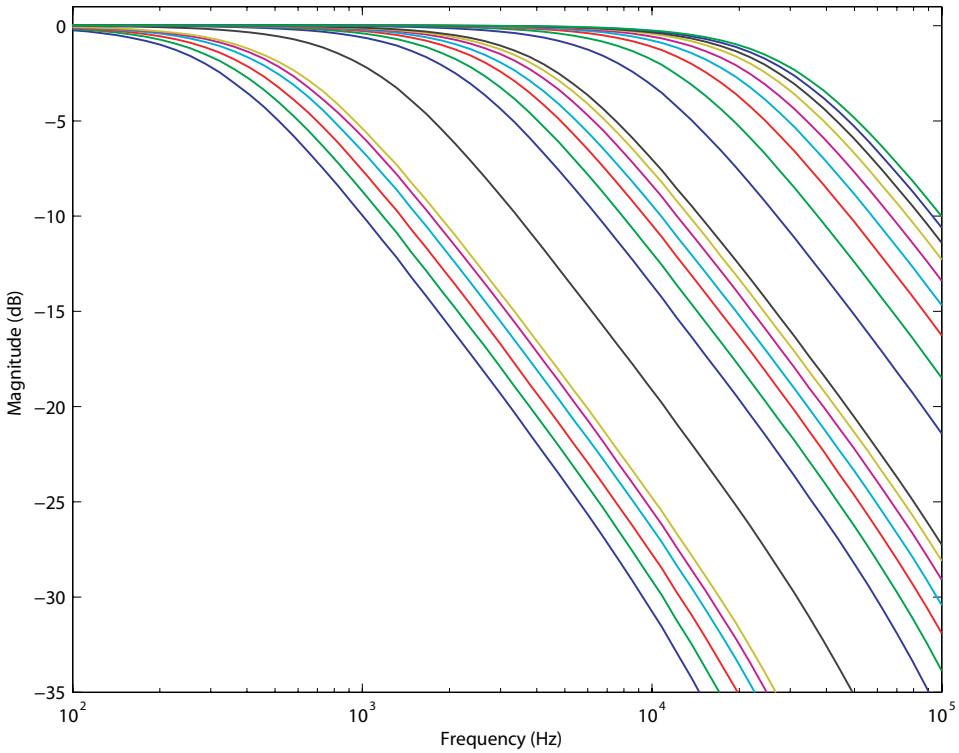


Figure 5.7. Low-pass follower data from the RASP 2.7.

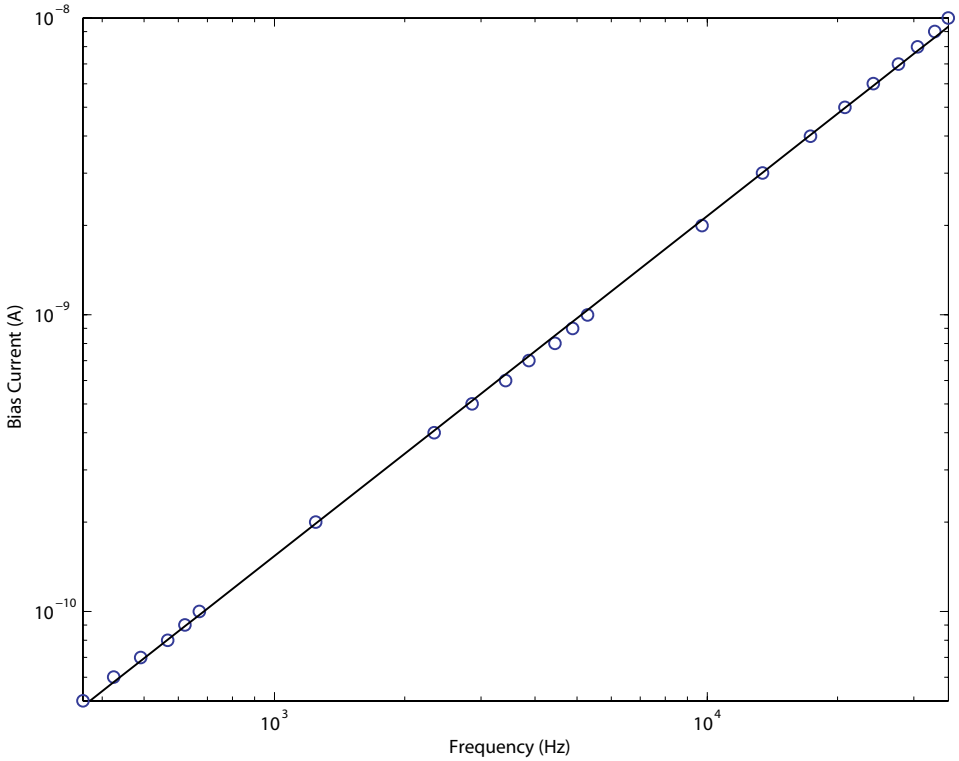


Figure 5.8. Corner frequency relationship to sub-threshold OTA bias currents.

5.3.2 Capacitively Coupled Summation

Some of the new systems intended for the RASP 2.x FPAAs required circuits and components that were not explicitly present in the CABs, so equivalent circuits were synthesized from the available components. One example is the summation block illustrated in Figure 5.9. Voltage summation is a fairly trivial task with resistors and operational amplifiers, but resistors are not commonly fabricated in ICs due to the large area requirements and variation between identically drawn resistors. However, capacitively coupled circuits, such as that depicted in Figure 5.9, can also perform this task.

Each input of this circuit is capacitively coupled to the negative terminal of the OTA. Another capacitor provides feedback while the pFET across this capacitor establishes the DC operating point of the output. If this was not a reconfigurable implementation of the circuit, the negative terminal of the OTA would be floating. The charge on this floating node would then need to be programmed in order to set the DC point of the circuit. An alternate topology uses a high resistance between a reference voltage and the pseudo floating node to set the DC point. However, a pFET, like the one in Figure 5.9, weakly biased in the sub-threshold region allows a leakage path from the output back to the negative terminal. This also creates a pseudo floating node and allows the OTA to set its own DC point to the reference voltage applied to the positive terminal of the OTA.

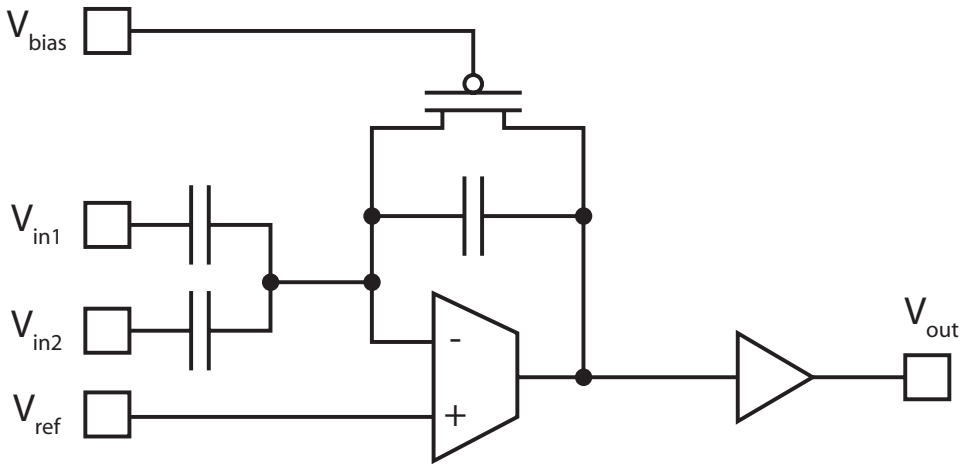


Figure 5.9. A capacitively coupled summation circuit using a pFET leakage resistance.

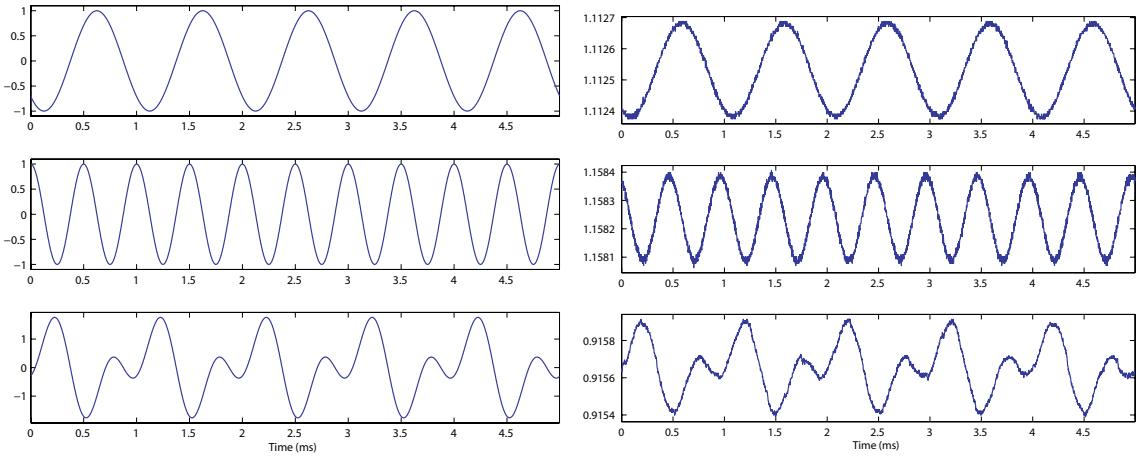


Figure 5.10. Summation circuit ideal and measured results.
(a) Ideal inverting summation with normalized amplitudes.
(b) Measured data from audio card inputs and circuit output.

To test the circuit, two sinusoids of frequencies 1 kHz and 2 kHz were generated using MATLAB and played over the PC's left and right audio channels. These signals were used as the input to the summation circuit and the output was captured using an oscilloscope. Figure 5.10 depicts the theoretical and measured responses of the summation circuit. The output is clearly the inverted summation of the two input signals. If desired, the original signal polarity could be recovered by simply using a single input version of this same circuit cascaded to the output. A frequency response of the circuit confirms the expected high-pass behavior, with the corner frequency determined by the size of the capacitors and the conductivity of the leakage pFET. For this circuit, high-pass corner frequencies below 100 Hz were observed, which is adequate for audio spectrum signals.

An interesting modification can be made to the circuit to reduce the component count and eliminate the I/O pin required for the leakage pFET's bias voltage. Instead of using one of the CAB's pFETs as the leakage path, switch transistors can be used, as seen in Figure 5.11. Since these are floating gate transistors, the conductivity of these switches can be adjusted to control the amount of leakage and therefore the high-pass corner frequency.

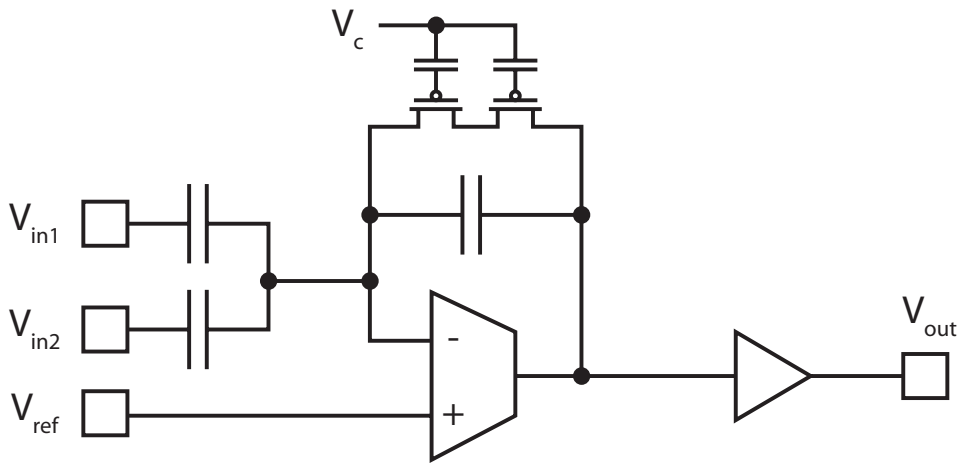


Figure 5.11. Summation circuit using the switch fabric as a resistance.

5.3.3 Capacitively Coupled Difference

The subtraction of two signals could also have been done with the inverting summation circuit of Figure 5.11. All of the positive terms would be summed and inverted using the structure and then fed into a second circuit cascaded to the output of the first. By attaching the negative signals to the second summation amplifier, the output would be the difference of the signals in the correct polarity. However, the circuit of Figure 5.12 can perform the same task in a more compact form. The form of this amplifier is very similar to the summation amplifier except that signals are now coupled into the positive terminal of the OTA as well. The positive terminal is also a pseudo floating node, which is set by another switch fabric leakage resistor to a reference voltage.

An interesting issue arose when examining the frequency response of this circuit, as seen in Figure 5.13. An imbalance in the capacitances at the pseudo floating nodes can be seen as a gain error for the individual inputs. In this case, the positive input, Figure 5.13b, has a slightly higher gain than that of the negative input, Figure 5.13a. By tweaking the routing a bit, the parasitic capacitance contribution from the routing can be used to help balance these nodes using the capacitive characterization data discussed earlier.

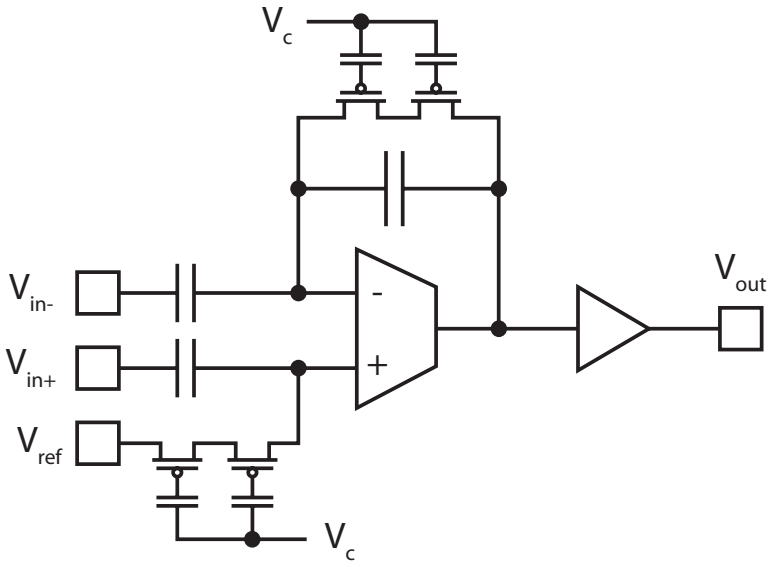


Figure 5.12. Capacitively coupled difference circuit.

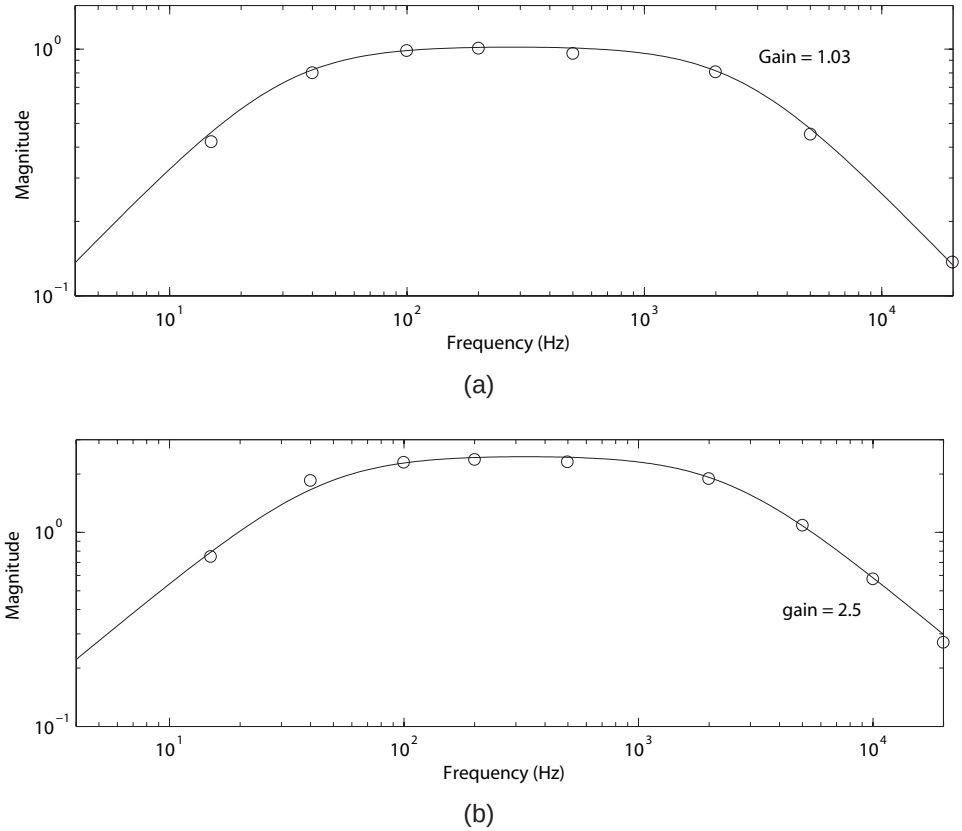


Figure 5.13. Frequency response of the capacitive difference amplifier.
(a) Negative input.
(b) Positive input.

5.3.4 Programmable Switch Fabric Current Source

Although no current sources were explicitly included as CAB components, the occasional need for them did arise. The transistors included in each CAB could easily serve as a current source, but this requires several I/O pins for each current value desired. However, another solution can be found within the routing fabric. Figure 5.14a depicts the routing configuration used to generate a current source. The only external signal required is V_{DD}, which can be shared across any number of current sources.

The drain of this device was swept to determine how good a current source the switch fabric can provide. For the first sweep, M_1 was programmed to the desired current, and M_2 was programmed as an "on" switch. M_2 was then programmed to a bias level slightly higher than that of M_1 in order to act as a cascode for M_1. This increases the output resistance of the current source, thereby making it more ideal. Figure 5.14b shows the results of these sweeps. When M_2 was programmed as an "on" switch, the current flowing

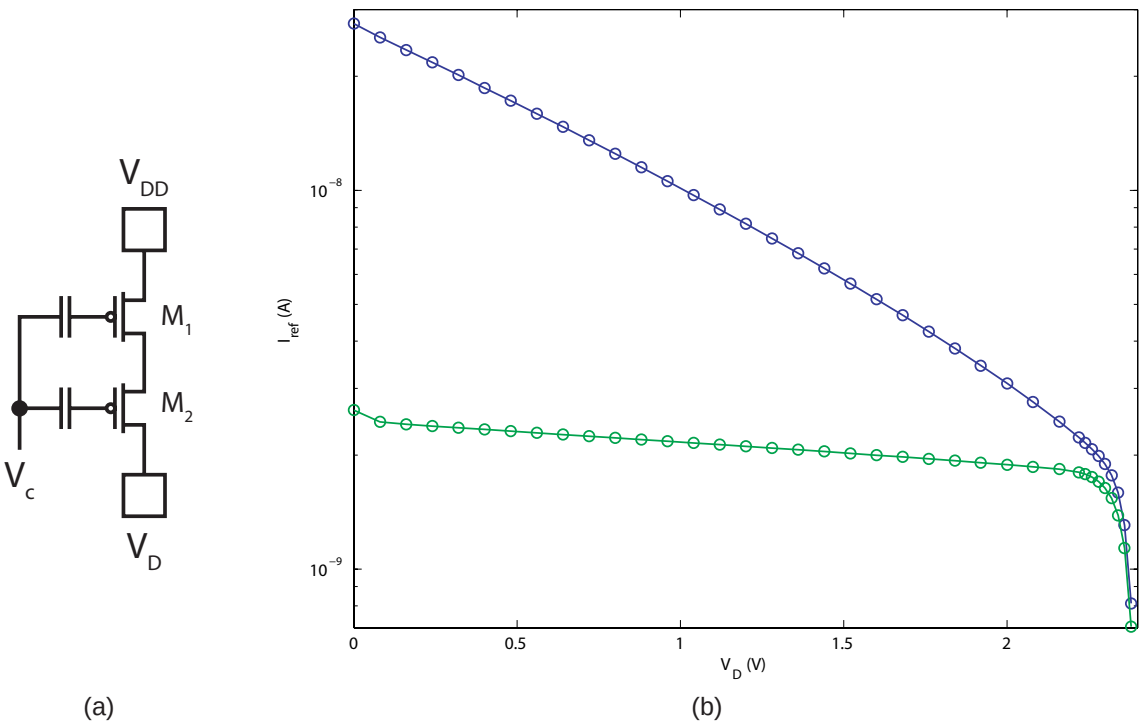


Figure 5.14. Current source/reference built within switch fabric.
(a) Circuit schematic showing a pair of switch transistors.
(b) Sweep showing current source value as a function of drain voltage.

through M_1 changed significantly with V_D. Cascoding the current source with another switch dramatically reduce the current dependence upon V_D. Additional switches could be added as extra cascode stages to further improve the current source at the cost of headroom.

5.3.5 Programmable Voltage Reference

In addition to current sources, voltage references can also be generated on-chip. Figure 5.15 shows one example of a voltage reference that uses a switch fabric current source to set the output voltage via a diode connect nFET. This voltage is then buffered by an OTA configured as a follower to isolate the reference circuit from any current loading.

For characterization purposes, the switch fabric current source was replaced by a pFET in order to quickly sweep the device over the operating range. Figure 5.16 shows the transfer function between the bias voltage applied to the pFET and the measured output voltage. The current flowing through the device was also measured during this sweep in order to derive a relationship between the programmed switch fabric current source and the output voltage, as shown in Figure 5.17. For verification, several points along the curve were programmed via the switch fabric current source and were also plotted.

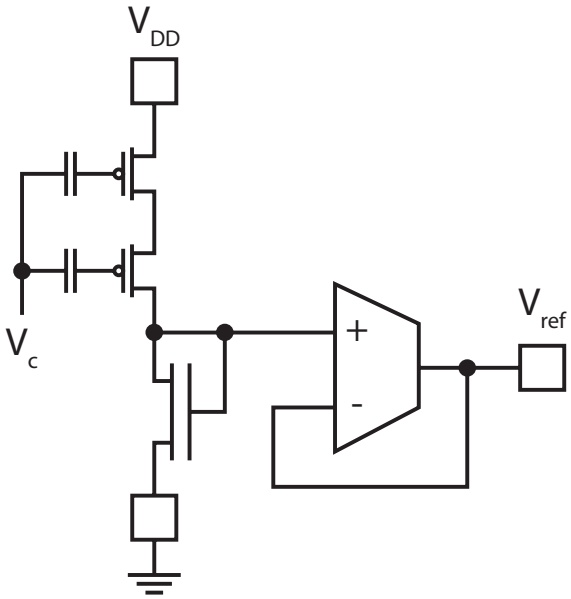


Figure 5.15. Reference voltage constructed using switch fabric current source.

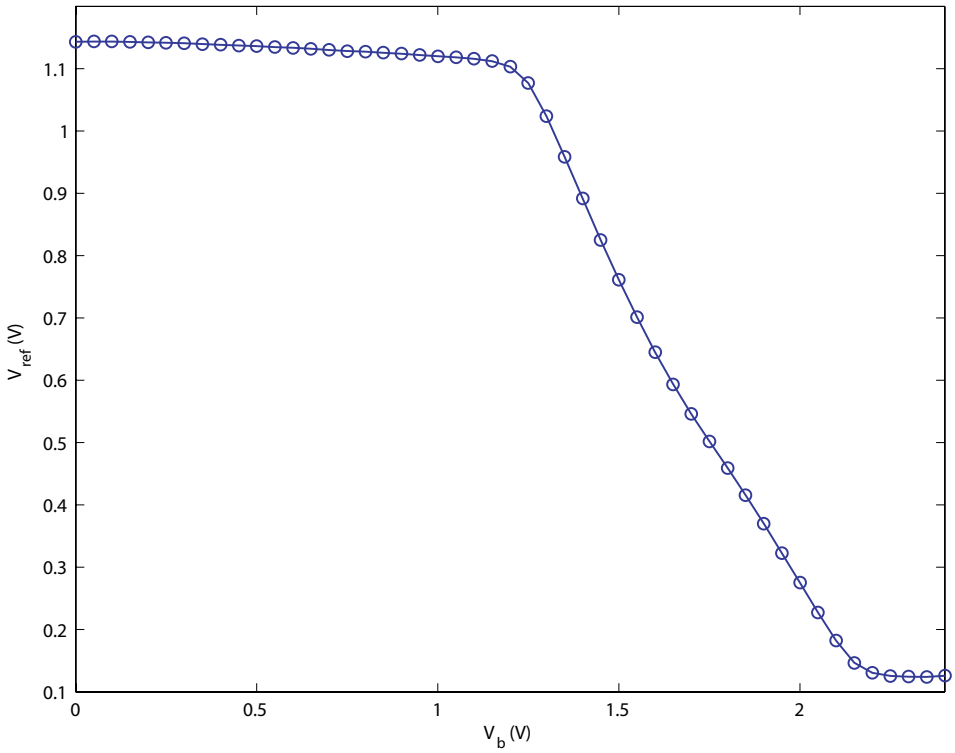


Figure 5.16. Voltage reference characterization using a voltage biased pFET.

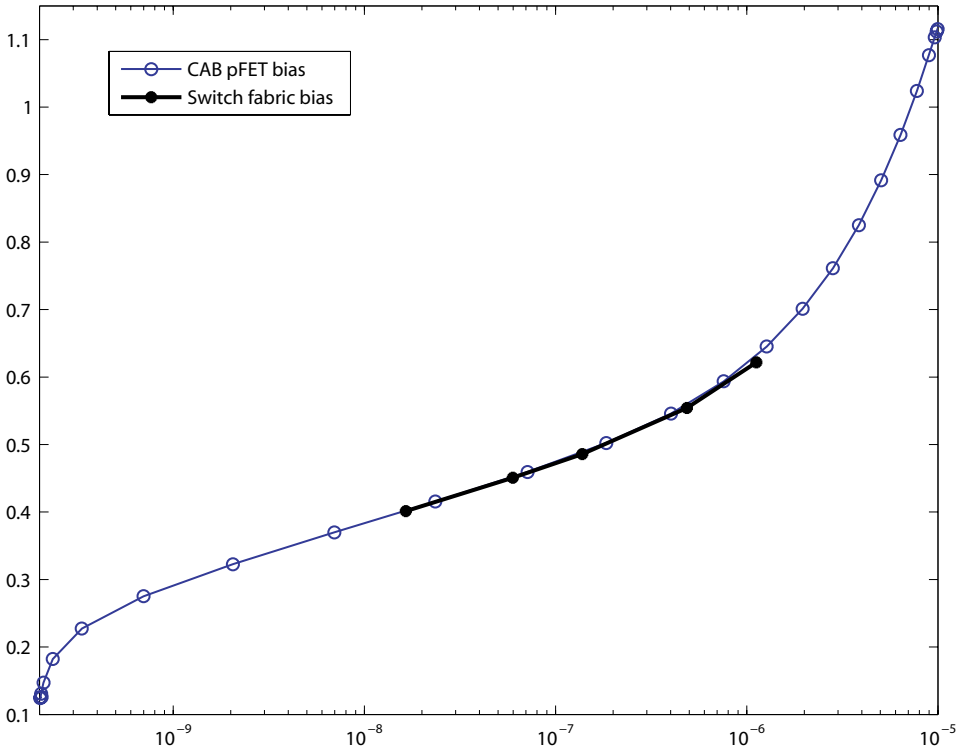


Figure 5.17. Voltage reference output set by a switch fabric current source.

5.3.6 Envelope Detector

Another interesting circuit is the synthesized minimum envelope detector of Figure 5.18. The minimum detector included within the CABs was functional, but somewhat difficult to use as a result of a design issue. The synthesized circuit was a bit more flexible and significantly easier to bias. The design is fairly simple, as seen in Figure 5.18a, but it does require components from two CABs as drawn. However, the second pFET transistor can be replaced with a switch fabric current source, as seen in Figure 5.18b, which reduces the circuit requirements to a single CAB.

The time constant programmability is demonstrated in Figure 5.19. The input signal is presented as a dotted line, while the output responses are given as thicker solid lines. By adjusting the bias voltage or programming a different switch fabric current, the slope of the rising edge can be significantly altered. Figure 5.20 shows the output response of the envelope detector to frequencies of 100, 200, 400, and 800 Hz. Again, the circuit input signals are given as dotted lines, while the output responses are thicker solid lines. As seen in Figure 5.20, the detector tracks the falling signals and slowly rises at the programmed rate on rising signals. By adjusting the current source, the circuit can be tuned for the desired frequency band.

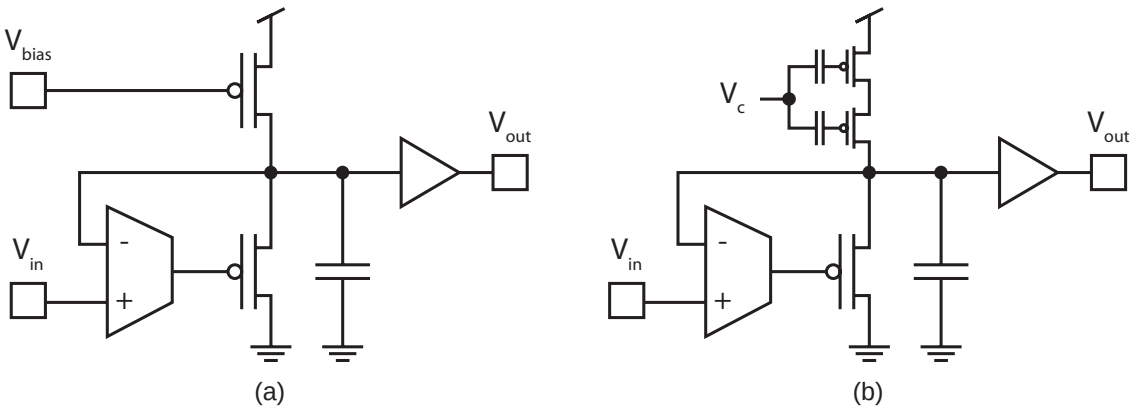


Figure 5.18. Synthesized envelope detector circuit.
(a) Minimum detector using an OTA, two pFETs, and a capacitor.
(b) Minimum detector circuit using switch fabric as a current source.

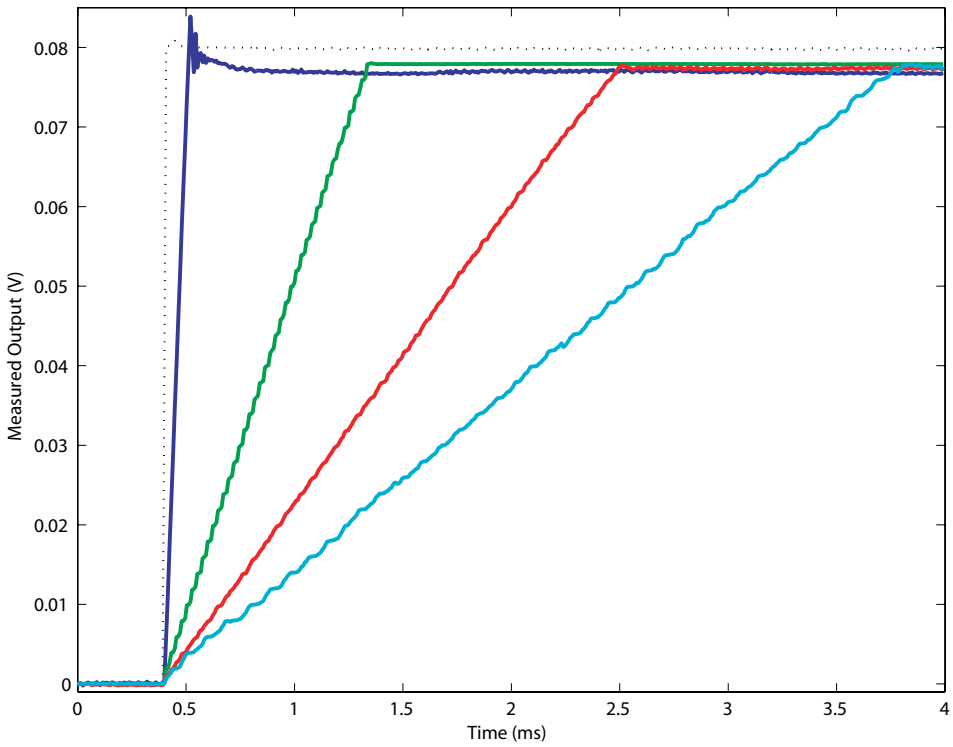


Figure 5.19. Programming the synthesized envelope detector's time constant.

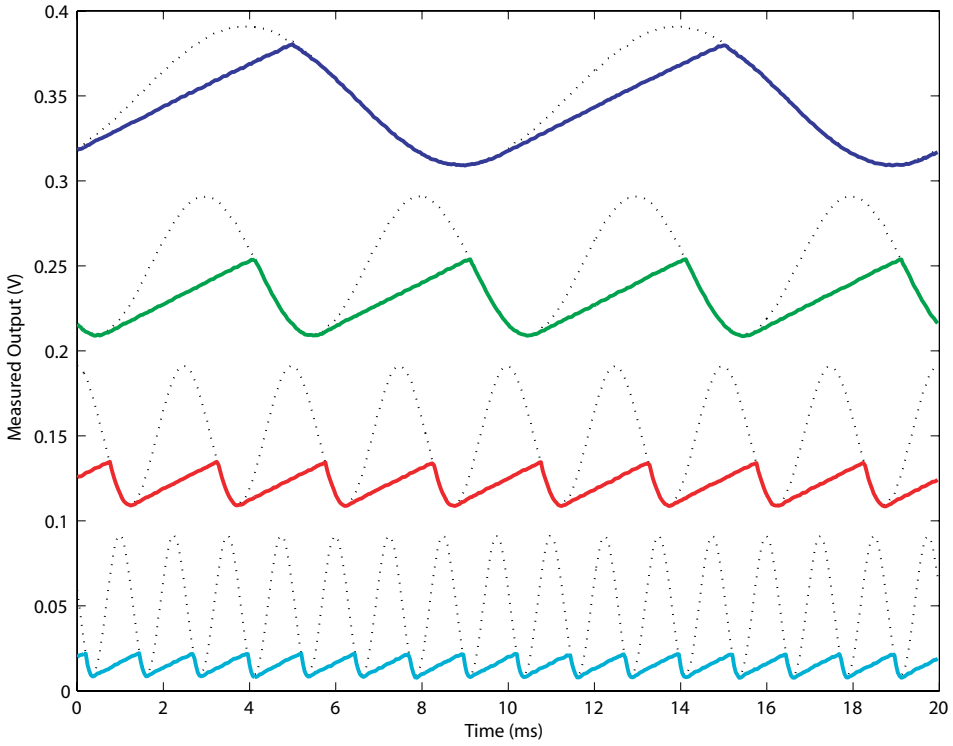


Figure 5.20. Measured minimum detector's response to various frequencies.

5.3.7 Band-Pass Resonator

A multiple CAB band-pass circuit was also synthesized using a resonator topology, as seen in Figure 5.21. The C^4 circuit topology discussed in Chapter 4 is generally preferable due to its compact size. However, the biasing for this particular implementation can sometimes be difficult when trying to obtain lower center frequencies, such as those needed for the audio spectrum. Fortunately, this is not a problem, since another band-pass topology can be easily synthesized as shown in Figure 5.21. The transfer function of this circuit is given by (5.3).

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{G_{M1}\, C\, s}{C^2\, s^2 + G_{M2}\, C\, s + G_{M3}\, G_{M4}} \tag{5.3}$$

The gain of the circuit is then controlled by the ratio of G_{M1} and G_{M2} as seen in (5.4).

$$Gain = \frac{G_{M1}}{G_{M2}} \tag{5.4}$$

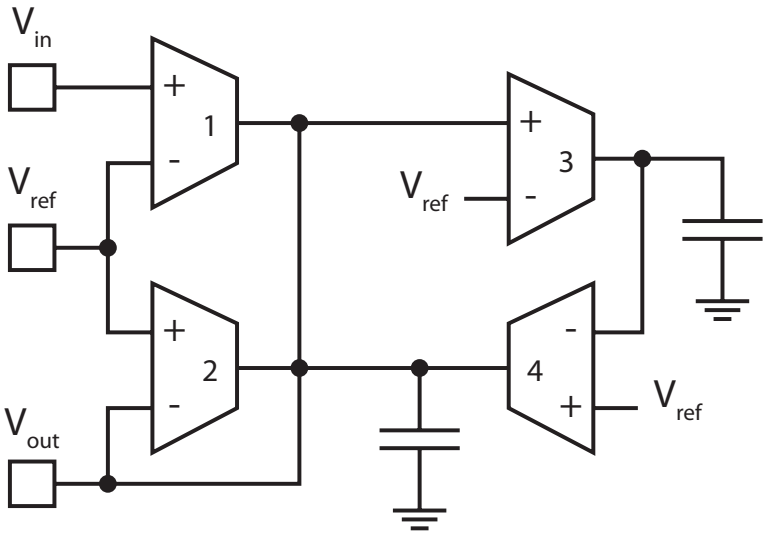


Figure 5.21. Synthesized band-pass filter using an OTA resonator topology.

5.4 Observations and Conclusions

The many successfully synthesized circuits have demonstrated the viability of large-scale FPAAs based upon floating gate transistors. These devices provide a new option for prototyping and designing large analog systems based upon reconfigurable and programmable

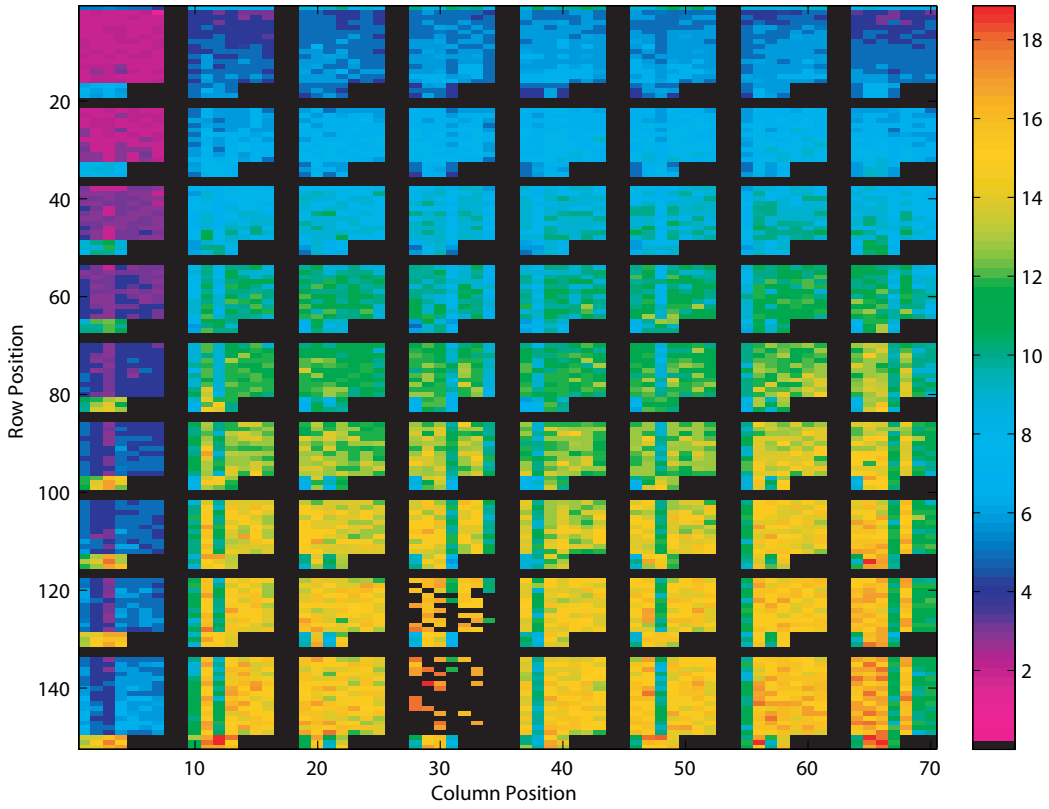


Figure 5.22. Switch isolation variation across die.

technologies that have been previously unavailable. Not only were floating gate transistors useful as programmable biases for the CAB components, but they proved useful as biases and programmable conductances within the switch fabric. This feature eliminates the notion of switches as "wasted" space and moves them closer to computational elements.

Although all of the results were encouraging, there were a few areas that could be improved. One issue revolves around the programming of the switches. As discussed in Chapter 2, the switch programming can be a bit tricky with the given architecture. The point of isolation for the floating gate transistors can vary significantly across the die, as seen in Figure 5.22. Transistors with a higher value have a higher isolation point, which requires that they be programmed to a higher level in order to program them as an "on" switch in parallel with other switches.

Part of this isolation variation across the die could be caused by processing variation,

which affects the properties of transistors as a gradient across the wafer. As seen in Figure 5.22, there is definitely a gradient from top to bottom present. However, there may also be an architectural and layout component involved as well. The external drain line connects to a pin in the upper left corner of the die. This also corresponds to the area where the switches have the lowest isolation points. Unfortunately, the on-chip routing lines for the drain signal were made fairly thin, which results in an increased line resistance over long distances. This line resistance appears directly in the programming path and increases in value as the locations get further away from the upper left corner. The affect of adding a series resistance to the drain has been studied [30] and is known to degrade the quality of floating gate switches. However, this does not account for the significant difference between the first column and the rest.

Fabrication defects may also be responsible for some of the variance in Figure 5.22. A couple of the CABs in the bottom center had extreme difficulties that make the switches nearly unusable. This could be the result of wafer defects, which drastically affect the transistor parameters. A fabrication defect could also be responsible for the significant difference between the first column and the rest. The external drain line runs across the top of the IC from left to right. Since this was routed as a fairly thin wire, it is conceivable that the wire was overly etched in the section between the first and second columns, which would significantly increase the resistance seen by the later columns, thereby decreasing the effectiveness of the drain pulses.

Figure 5.23 shows a histogram of the isolation points observed across the die. The values are fairly evenly distributed across a wide range of currents. Because of this, a single programming isolation point cannot be chosen, which makes programming multiple switches along a column difficult. To account for this, the isolation point used for programming “on” switches may need to reflect the gradients shown in Figure 5.22. Another potential and perhaps more practical solution to this problem may be the indirectly programmed switches discussed in Chapter 2. Since these switches can be isolated using row

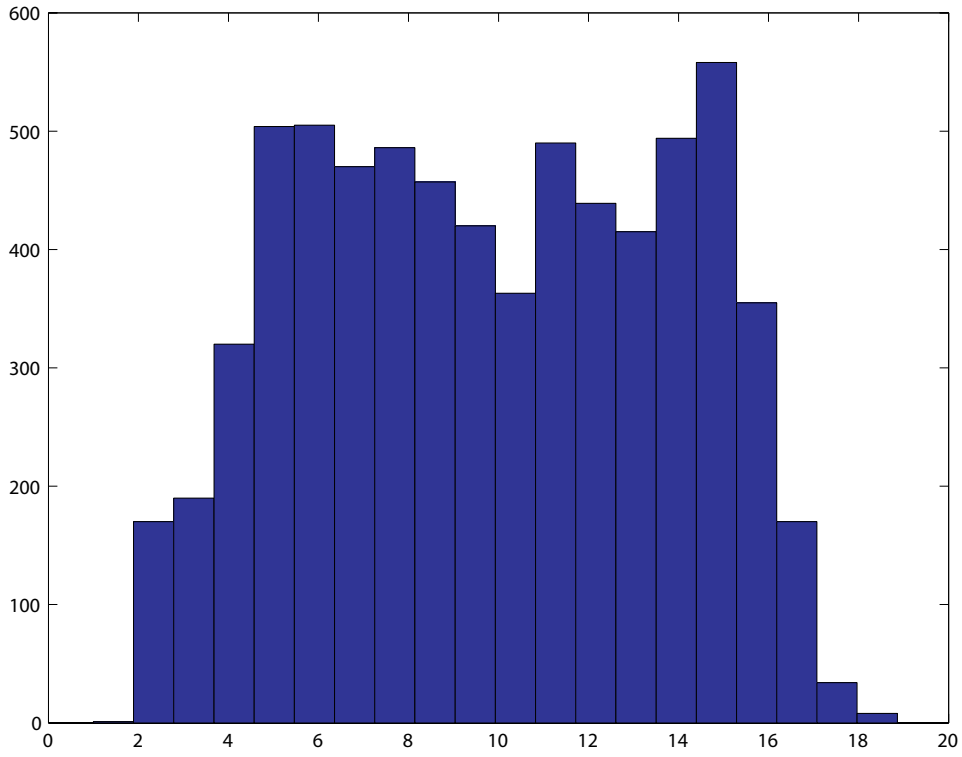


Figure 5.23. Switch isolation breakpoint histogram.

selection circuitry, they do not require complicated or coordinated programming schemes, such as those currently being used.

CAB component selection could also be improved. Quite often circuits synthesized in the RASP 2.x ICs could have benefited from additional transistors or two terminal capacitors within the CABs. Several components included within the general purpose CABs were not used very often, such as the min and max detectors and the C^4 band-pass elements. Instead of including these elements, more transistors and capacitors should be included on future revisions. Another useful item would be a dynamic switch, such as a transmission gate. Discrete time circuits would significantly benefit from such switches, since synthesizing these elements utilizes a significant amount of CAB resources for a fairly primitive and useful device. Capacitor sizing could also be improved. Although it was convenient to use the parasitic routing capacitance, it would be easier to design circuits based upon a drawn dominant capacitor within the CABs.

Signal routing quickly became a problem within the RASP 2.x FPAAs. The inclusion of only local and global routing severely limited the utilization possible with these devices. Most circuits tended to route from one CAB to an adjacent CAB, which meant that the global routing lines being used to connect these two CABs were being mostly wasted. A more efficient routing scheme would also include nearest neighbor routing, which should significantly improve the routing density as well as the component utilization across the IC.

CHAPTER 6

HIGH PERFORMANCE FPAA

Since the maximum bandwidth of a system degrades with the number of switches in the signal path, it is desirable to reduce or somehow eliminate these switches in high-performance applications. Many approaches have been suggested, such as the digitally controlled G_M cell architecture described in [9] or the metal-to-metal antifuses [7]. Most of these devices use redundant components with controllable biases, such as Becker's G_M cells, to make connections to other components. This redundancy can lead to significant space requirements and diminished utilization efficiency. However, floating gate transistors may provide an advantage in such architectures by saving a significant amount of area for programmability. The high-performance FPAA[1], as seen in Figure 6.1, was developed to investigate this possibility.

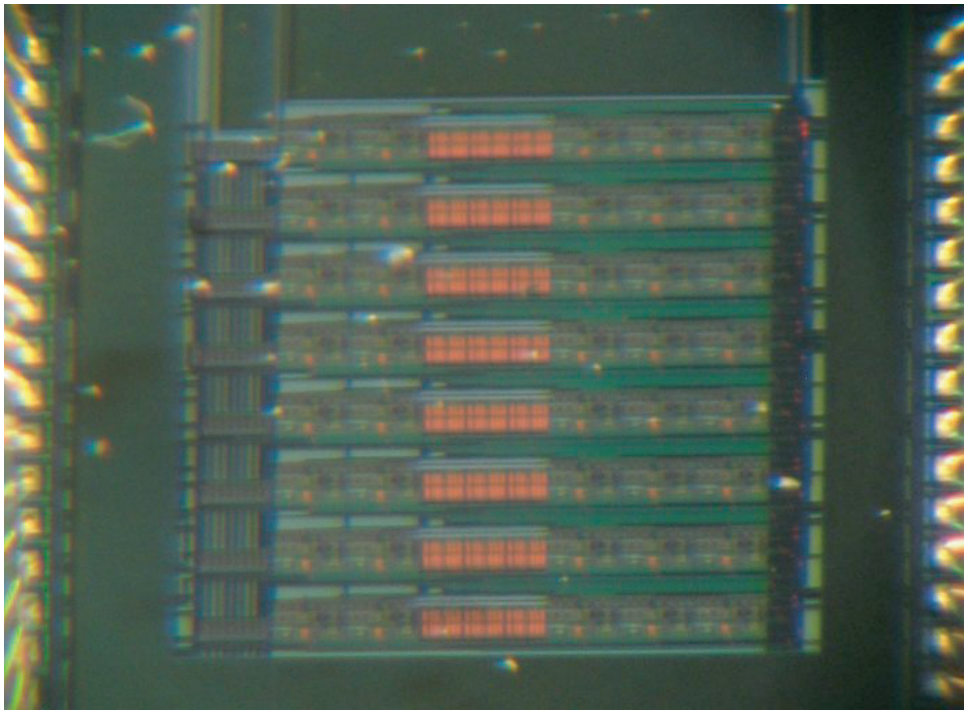


Figure 6.1. High-performance FPAA die photograph.

[1]This work was partially funded by the JPL Self-Reconfigurable Electronics for Extreme Environments (SREE) project.

6.1 Architecture

The high-performance FPAA architecture, as seen in Figure 6.2, contains a number of CABs with direct input and output connections to pins. This allows high bandwidth circuits to have a degree of programmability and reconfigurability without any switches in the signal path. However, more complex computations may require multiple CABs. For this reason, rows 1 through 4 in the high-performance FPAA have switched inputs. The output of each CAB is routed back to a switch matrix connected to the inputs of the CABs. By disconnecting the switch between a pin and one of the switched input CAB rows, the output of any other CAB can then be routed to the input of this CAB through a single switch. Although there is now a switch in the path, the bandwidth of this signal is still fairly high since it is only a single switch, as opposed to the switch pairs required in the previous FPAA architectures.

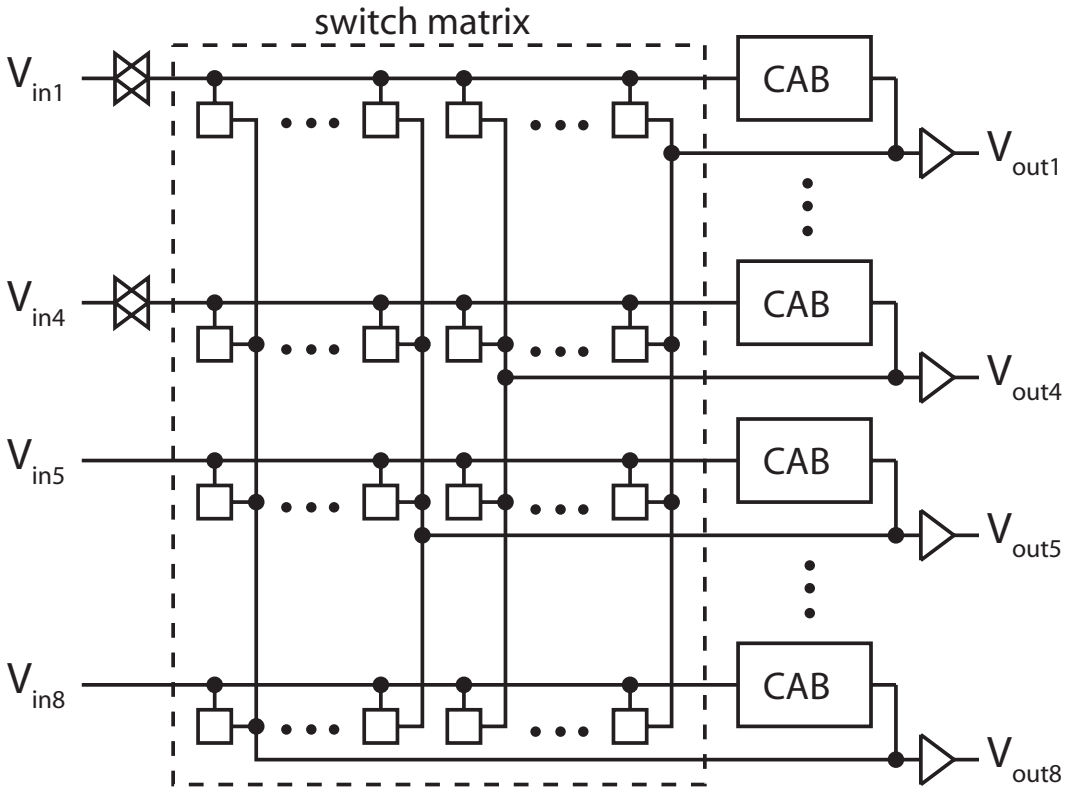


Figure 6.2. High-performance FPAA architecture.

The switches on this IC, represented by squares inside the switch matrix block of Figure 6.2, were composed of transmission gates controlled by a shift register. This was done to provide a very fast way to switch between several pre-programmed transfer functions. Floating gate transistor switches would also work in this situation, except that the programming time would have been slower than desired. As the programming circuitry moves on-chip, the programming time should dramatically decrease and make floating gate switches a more viable option for high-speed dynamic reconfigurability.

The CAB of this FPAA was chosen to contain a single biquad circuit, as depicted in Figure 6.3. This particular biquad configuration allows any two-pole, two-zero system to be synthesized simply by controlling the size of the capacitors and the transconductance of each OTA in the biquad. The transfer function is given by (6.1).

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{C_x\, C_f\, s^2 + G_{M2}\, C_x\, s + G_{M1}\, G_{M4}}{C_x\, (C_L + C_f)\, s^2 + G_{M5}\, C_x\, s + G_{M3}\, G_{M4}} \tag{6.1}$$

The transconductances, G_{Mx}, correspond to OTA x. As was the case with the OTAs in the general purpose CABs, floating gate transistors are used to set the bias current in the

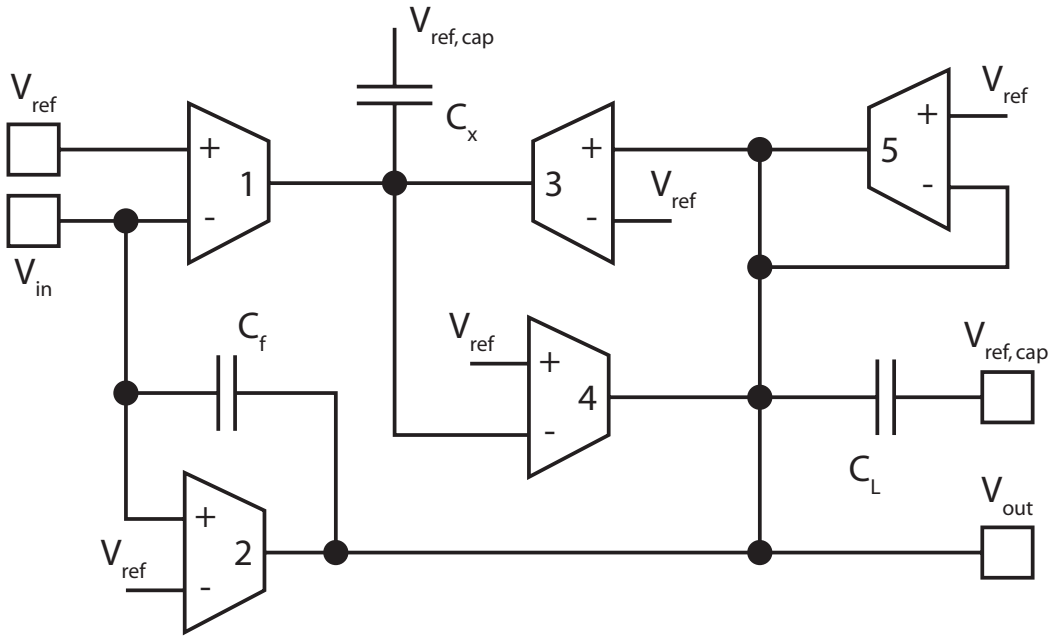


Figure 6.3. Biquad circuit topology used for the high-performance FPAA CAB.

OTAs, thereby determining the transconductance. Although the OTA transconductances are the primary programmable parameters in this design, the capacitors were also made to be selectable, as seen in Figure 6.4a, to extend the possible range of operation. The fabricated IC contains the three capacitors illustrated in Figure 6.4a and is digitally selectable using a shift register. In this manner, any combination of these capacitors can be selected. If none are selected, the dominant capacitance will be parasitic.

Each OTA in the biquad is actually a parallel combination of various OTA architectures each with its own programmable floating gate bias, as seen in Figure 6.4b. Since each OTA design is independently biased, individual OTAs within the block can be shut down by simply depriving them of bias current. This provides another degree of reconfigurability by allowing the user to select the OTA with specifications appropriate for the application. The user can even mix the output responses from the different OTA designs within a block by programming their respective biases.

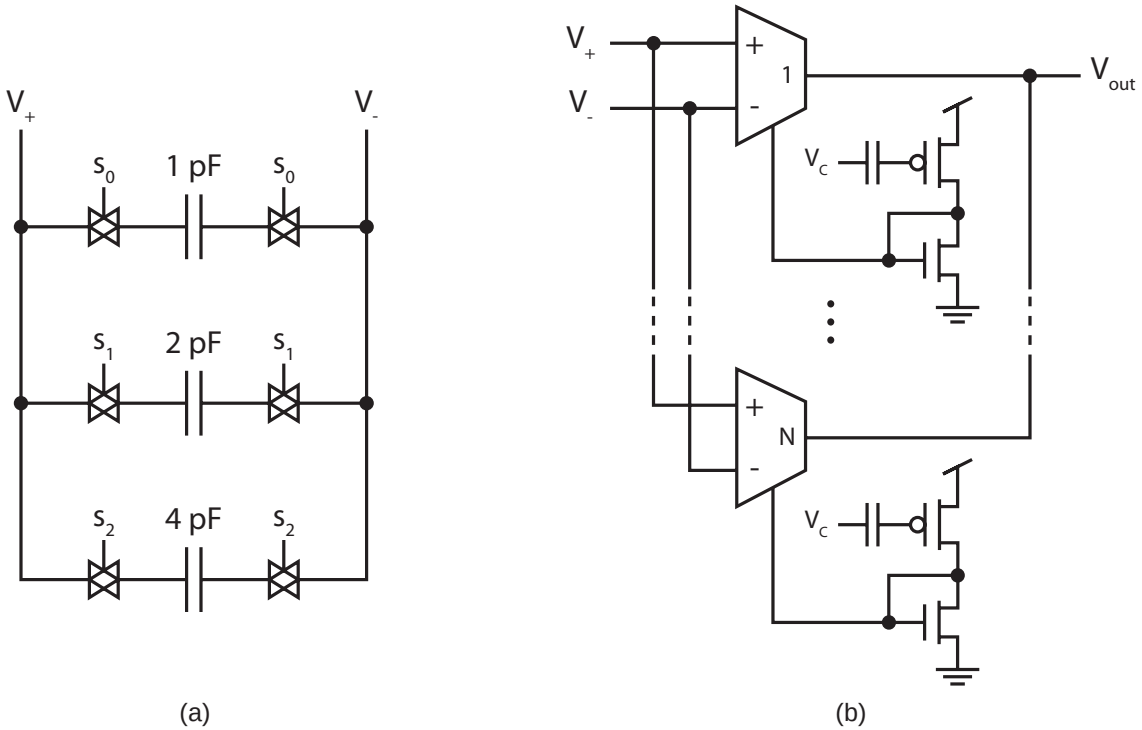


Figure 6.4. Reconfigurability in the high-performance FPAA biquad.

6.2 Low-Pass Filter Implementation and Results

A low-pass filter was synthesized using a single biquad, as shown in Figure 6.5. This configuration is achieved by programming the biases of OTAs 1, 3, and 4 as low as possible and unselecting capacitors C_x and C_f. This reduces (6.1) to (6.2).

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{G_{M2}}{s\,C_L + G_{M5}} \tag{6.2}$$

The gain of this circuit it set by the ratio of the transconductances, as seen in (6.3).

$$Gain = \frac{G_{M2}}{G_{M5}} \tag{6.3}$$

The time constant is then given by (6.4).

$$\tau = \frac{C_L}{G_{M5}} \tag{6.4}$$

Frequency response data was taken for various values of C_L as seen in Figure 6.6. The biases of OTA 2 and OTA 5 were programmed to the same level, which should have resulted in a gain of 1 given (6.3). However, a slight mismatch between the diode connected nFETs used to convert the programmed floating gate pFET currents into bias voltages and the nFETs within the OTAs that set the tail current. This mismatch would multiply the programmed bias current by the ratio of the fabricated device geometries, W/L. As seen in

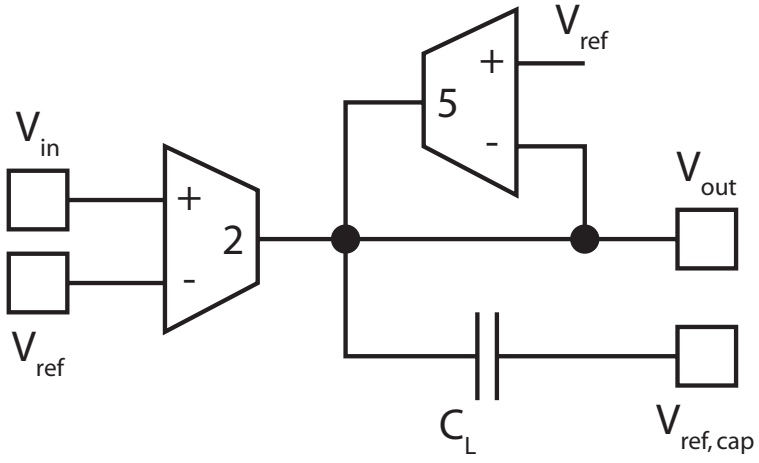


Figure 6.5. Low-pass filter synthesized using the biquad CAB.

the data of Figure 6.6, G_{M2} seems to be about twice the magnitude of G_{M5}, which results in a gain of approximately 3 dB. Most likely the current mirrors were mismatched such that G_{M5} was lower than programmed, and G_{M2} was higher than programmed giving the net factor of 2 difference. Programming error could also have contributed a small amount of this gain error.

The results in Figure 6.6 also deviate from the ideal transfer function, (6.2), in that they exhibit a resonance peak at the corner frequency. The first-order circuit in Figure 6.5 should have a relatively flat passband with no peak, but the biquad synthesizes a pseudo first-order circuit by canceling some of the parameters in (6.1). This peak is most likely a result of the various other biquad parameters not being equal to precisely zero. For instance, the capacitor C_x is assumed to be 0, but the parasitic capacitance seen at that node in the biquad circuit will contribute to the output response.

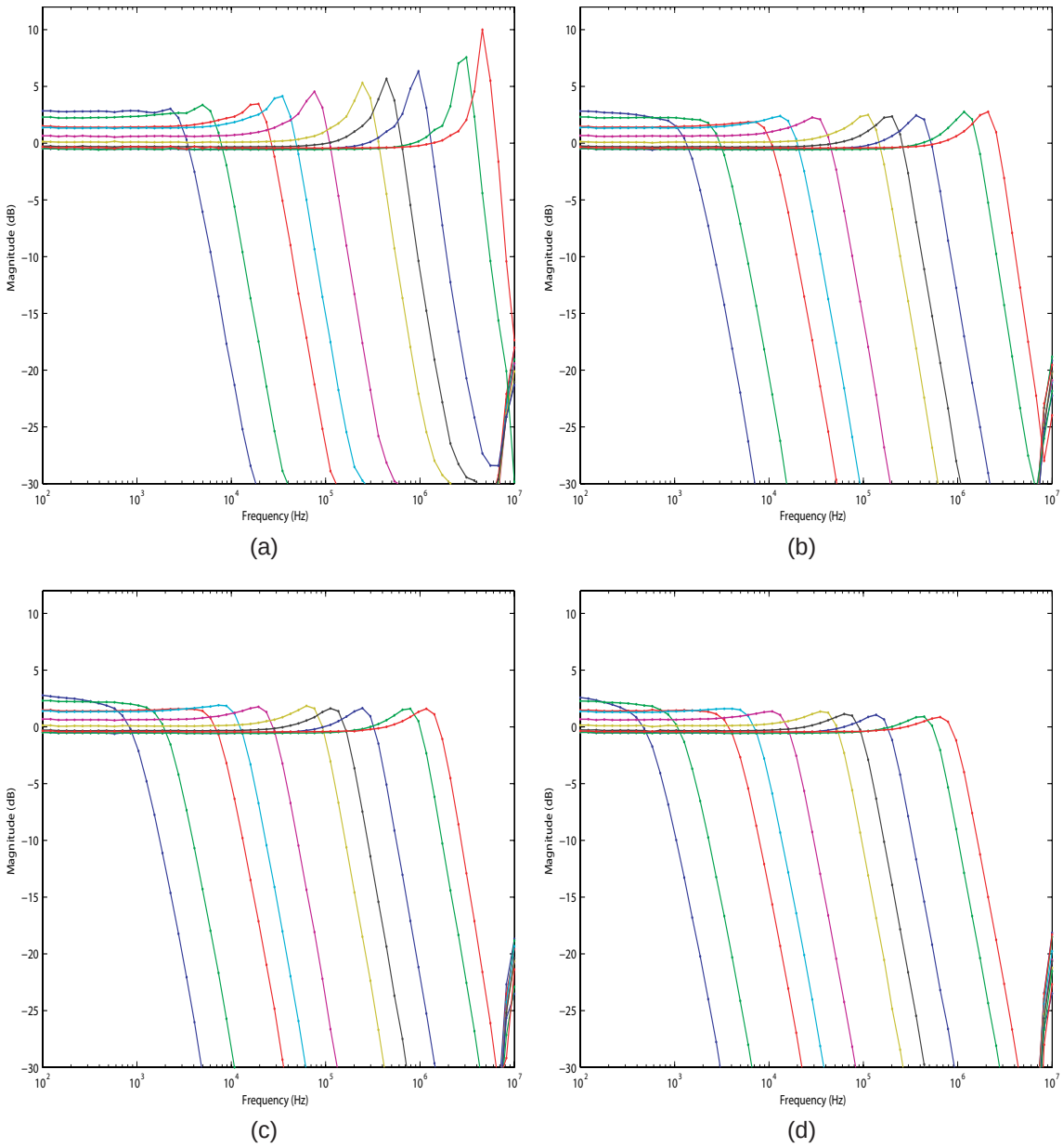


Figure 6.6. Biquad synthesized low-pass filter frequency responses for several load capacitances.
(a) Drawn load capacitance = 0 pF
(b) Drawn load capacitance = 1 pF
(c) Drawn load capacitance = 2 pF
(d) Drawn load capacitance = 4 pF

CHAPTER 7

LARGE-SCALE FPAAS, THE NEXT GENERATION

The third-generation RASP FPAAs, RASP 3.x, combine aspects of the general purpose RASP 2.x line with the higher bandwidths possible in the high-performance FPAA. These devices retain much of the generality and flexibility of the RASP 2.x FPAAs by including similar general purpose CABs. Specialized computational blocks have been added at key points in the array to provide higher functionality and bandwidth. The RASP 3.0, as seen in Figure 7.1, is the first IC of this class and was fabricated on a 4.5 mm x 8 mm die in a .35 μm process available through MOSIS. The specialized blocks within the RASP 3.0 were designed for audio and other similar signal processing algorithms.

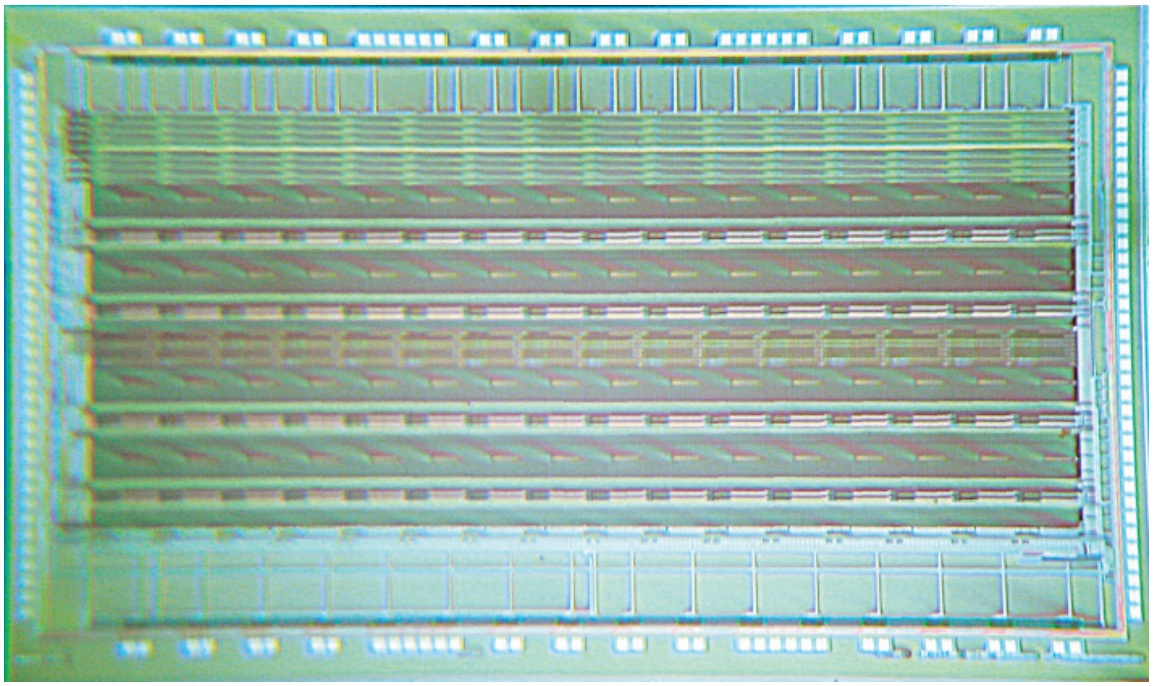


Figure 7.1. RASP 3.0 die photograph.

7.1 Architecture

The RASP 3.0 design began by examining common audio signal processing systems such as equalizers, feature extraction front-ends, and hearing aids. Most of these and other algorithms shared a common set of operations, as illustrated in Figure 7.2. The first operation is typically a frequency decomposition, which is commonly performed by an FFT in a DSP. The frequency decomposition step can be viewed as a filter bank operation, which is depicted as a parallel set of band-pass filters in Figure 7.2. The output of each filter bank element is then passed through a series of transforms such as envelope detectors, expansive or contractive power laws, and signal-by-signal multipliers. Some algorithms also include a measure of interaction between these signal bands, which is not depicted in Figure 7.2. Typcally, the end result of each column is then recombined by some weighted summation to generate one or more outputs.

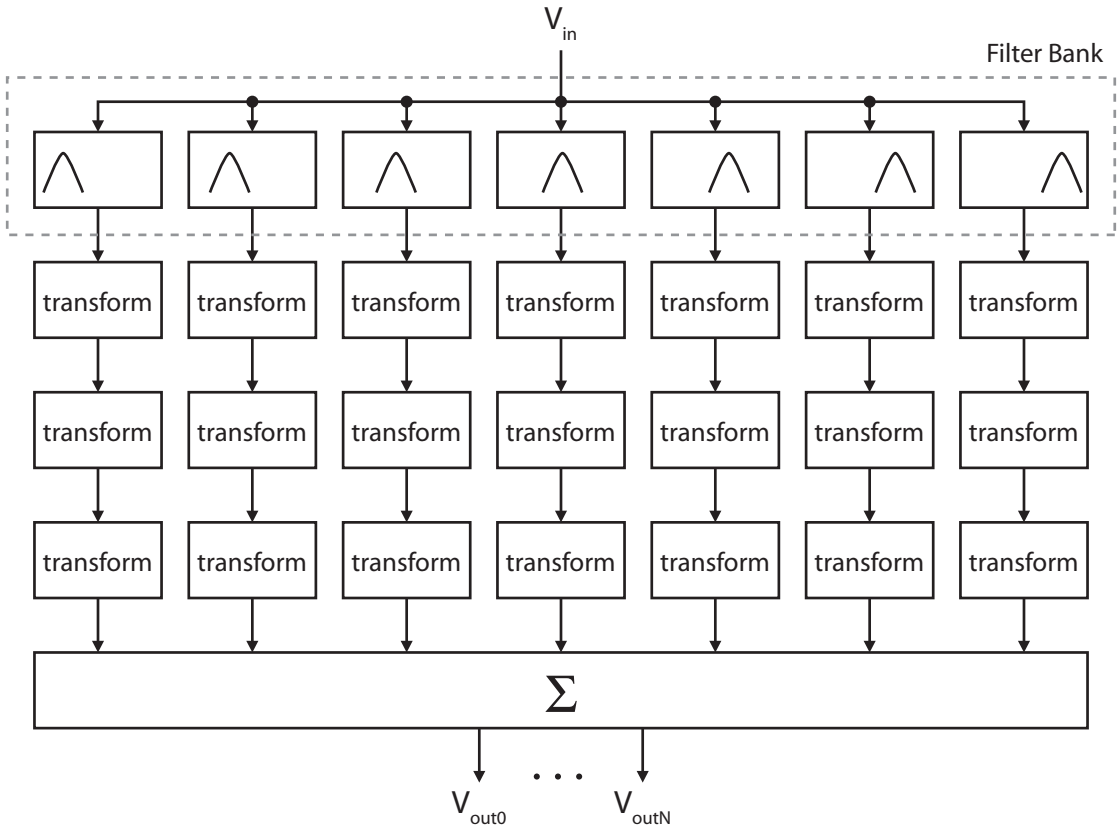


Figure 7.2. Common algorithm steps in audio signal processing.

Considering Figure 7.2, the RASP 3.0 was designed specifically for frequency decomposition and channel based processing. Each band, or channel, in Figure 7.2 appeared to be mostly independent of the surrounding channels. The structure is somewhat analogous to the bit slice concept in digital processing. Since the architecture for a single channel, or bit in the digital equivalent, is identical to the architecture of any other channel, a single processing slice can be designed, laid out, and tiled to produce a processor of arbitrary channel length, or register length in the digital case. Utilizing this concept, the core of the RASP 3.0 is comprised of sixteen fully differential channel slices, as depicted in Figure 7.3.

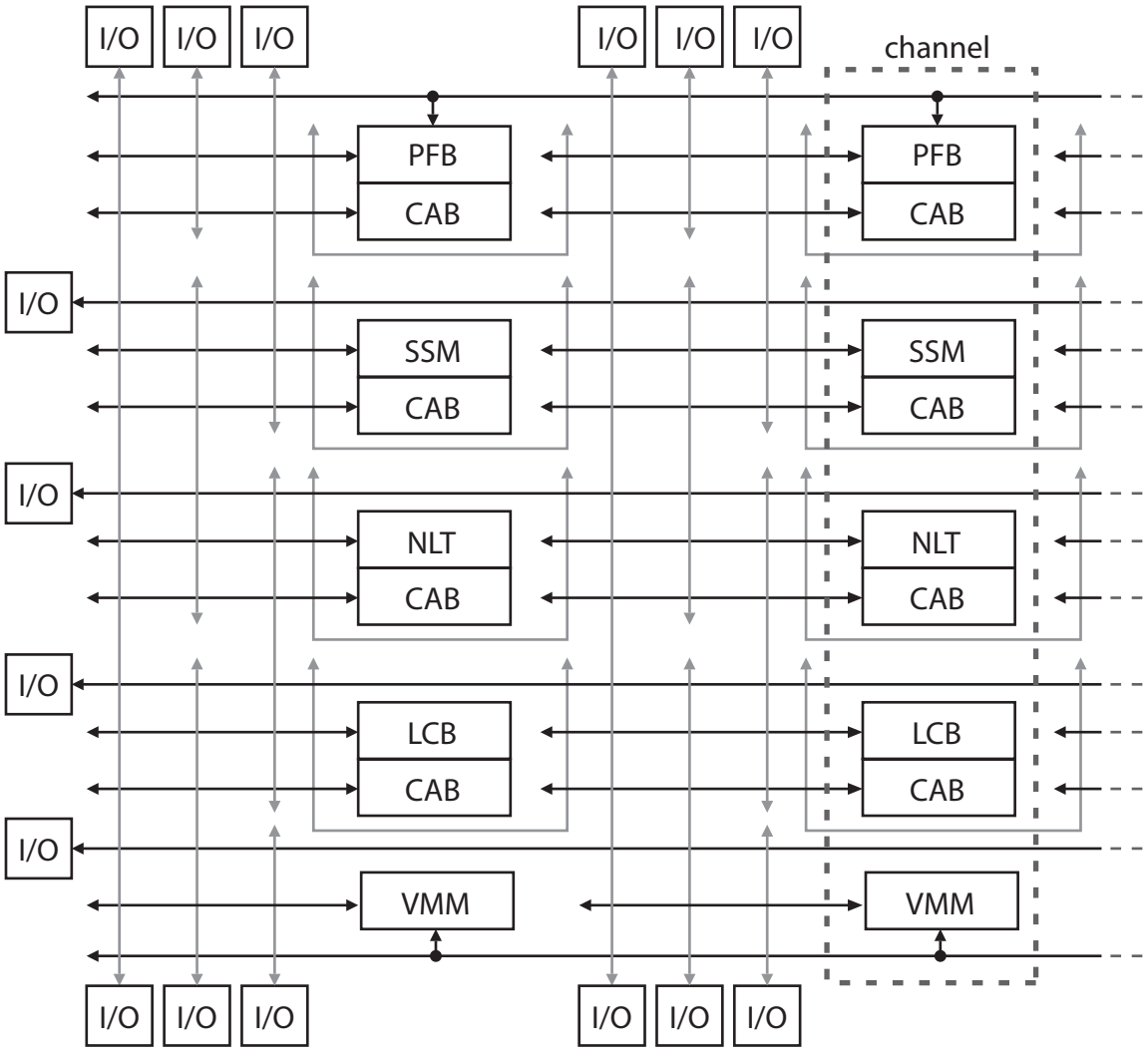


Figure 7.3. The RASP 3.0 FPAA architecture.

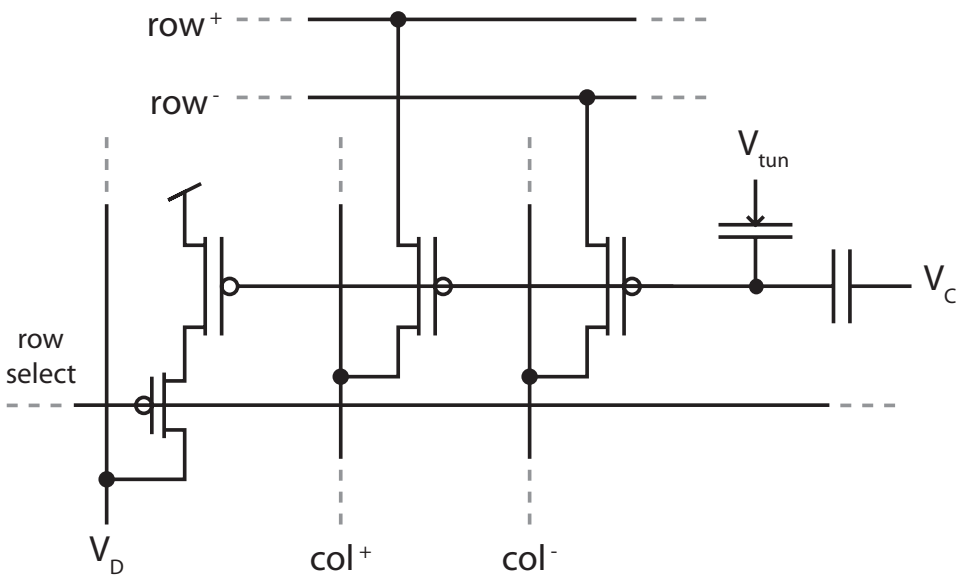


Figure 7.4. Indirectly programmed differential switch used in the RASP 3.0.

Each channel slice contains several general purpose CABs and special purpose functional blocks. The signal flow starts at the top of the IC and flows down along the columns. An improved switch fabric interconnects the various CABs and specialized blocks. Since the components and therefore the routing within this FPAA were differential, an interesting architectural issue arose. The switches used within the previous generations were all directly programmed, but an indirectly programmed switch would allow both positive and negative routing to be controlled by a single floating node, as depicted in Figure 7.4. This would save considerable programming time by simplifying the programming to a single pFET instead of two. To improve switch isolation and observability, a row selection switch was added to the programmer pFET.

The routing network of the RASP 3.0 has also been updated to reflect lessons learned while working with the RASP 2.x line of FPAAs. Instead of relying upon global routing for CAB-to-CAB connections, the RASP 3.0 contains local nearest neighbor routing, as seen in Figure 7.3. Most signals routed on the RASP 2.x FPAAs were not broadcast to multiple destinations across the chip. Instead, many signals simply required point-to-point connections with an adjacent CAB. On the RASP 2.x, signals such as these would have to

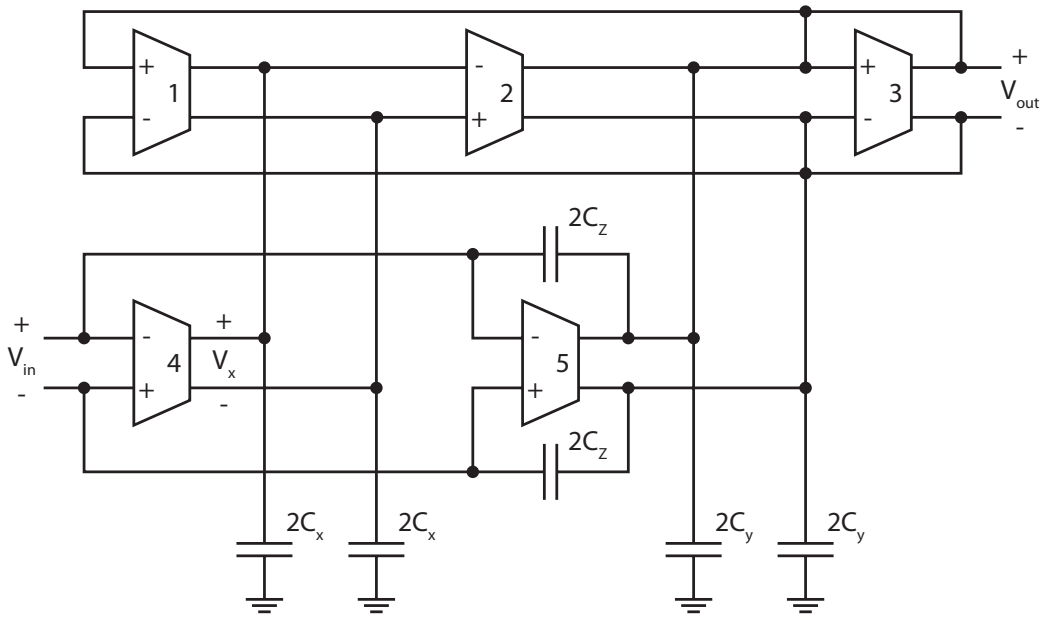


Figure 7.5. Differential biquad circuit topology.

be routed along global vertical and horizontal lines, which were very limited in number. Very quickly the global routing lines would become completely utilized and limit implementable system size. However, local routing lines that connect adjacent blocks allow for more routing lines in the same amount of space as before. Local horizontal routing lines also enable intermixing between adjacent channels.

This biquad is a two-pole, two-zero topology similar to the high-performance FPAA's biquad, except that it is fully differential [40], as seen in Figure 7.5. The transfer function of this circuit is given by (7.1).

$$\frac{V_{out}(s)}{V_{in}(s)} = \frac{a\,s^2 + b\,s + c}{s^2 + \frac{\omega_n}{Q}s + \omega_n^2} \tag{7.1}$$

The various parameters of (7.1) are given by (7.2) through (7.6).

$$a = \frac{C_z}{C_y + C_z} \tag{7.2}$$

$$b = \frac{G_{M5}}{C_y + C_z} \tag{7.3}$$

$$c = \frac{G_{M4}\,\omega_n}{C_x} \tag{7.4}$$

$$Q = \sqrt{\left(\frac{G_{M1}G_{M2}}{G_{M3}}\right)\left(\frac{C_y + C_z}{C_x}\right)} \tag{7.5}$$

$$\omega_n = \sqrt{\frac{G_{M1}G_{M2}}{C_x\,(C_y + C_z)}} \tag{7.6}$$

The biquad terminals, V_{in} and V_{out}, are routed to rows of the local switch matrix. V_x is also routed to a row to provide access to individual OTAs. As was the case with the high-performance FPAA, the bias currents of all other OTAs would be shut down.

In addition to the biquad circuit, the RASP 3.0 CAB is also composed of transistors and capacitors, which come in pairs to accommodate the differential signals. In addition to standard nFETs and pFETs, these CABs also include floating gate pFETS. There are also more capacitors in the RASP 3.0 CAB than there were in the RASP 2.x line. With the successful implementation of numerous capacitively coupled circuits in the RASP 2.x ICs, it was clearly evident that more capacitors would be needed to implement large systems based upon these circuits on future generations of FPAAs.

7.2 The Channel Slice

As seen in Figure 7.3, the channel signal processing begins with a common input routed on a dedicated line to each channel, which contains a single element of the programmable filter bank (PFB) [41]. This element band-pass filters the input signal, using a C^4 second-order section similar to those described in Chapter 4, into the specific frequency range for the particular channel slice. The signal is then routed out to a local or global vertical routing line or passed through a min/max detector. Flowing down the channel stack, the signal passes through or by the first general purpose CAB.

Just below the PFB/CAB block sits the SSM/CAB block in the channel stack. A signal-by-signal multiplier (SSM) is located here and provides a way to mix two signals. Below this block resides the nonlinear transform (NLT) circuit in the NLT/CAB block. This circuit is composed of a specialized multiple-input translinear element (MITE) network [42–44] which performs compressive or expansive power law transforms on the signal. The next block in the stack, LCB/CAB, contains part of the linear combiner block. This capacitively coupled structure allows the individual signals from each channel to be recombined into a single output signal. This is useful for audio applications that perform frequency dependent transformations on the individual bands and recombines them as an output audio signal, such as an equalizer.

The final block in the stack is the vector-matrix multiplier slice. The full vector-matrix multiplier is composed of the individual slices from each channel. The outputs of these slices are tied to dedicated global lines that are shared by each slice. Since the output of each individual multiplier is a current, summation is achieved through simple KCL. A current-to-voltage converter restores the individual signals to a voltage that is transmittable throughout the FPAA.

CHAPTER 8

THE FPAA IN EDUCATION

FPAAs could significantly impact analog design and embedded systems education. A common part of analog design educational laboratory exercises is the selection and wiring of discrete components, which is rather similar to the way digital design used to be taught using discrete TTL or CMOS ICs. However, PLDs and FPGAs were eventually introduced to the class laboratory as another design option. They provided greater design freedom and quicker circuit implementation, which is something FPAAs could also do for analog. At the IC design level, FPAAs could be used as functional prototyping or development platforms, just as FPGAs have done. Instead of fabricating and testing individual analog circuits on individual analog ICs, one mass produced FPAA could be used to synthesize circuits for an unlimited number of laboratory exercises.

Although the effect on analog design seems impressive, the impact on embedded systems could be even greater. Embedded systems generally involve the integration of various sensors, displays, and mechanical systems using an FPGA, DSP, or microcontroller because of the design flexibility they afford. As such, many analog sensors require data converters to produce a signal that can be processed by the digital device. These data converters can consume a significant amount of power, which is a concern for portable embedded systems running on batteries. However, the incorporation of an FPAA would provide another option, analog signal processing. An FPAA would not eliminate the need for a programmable digital device in all of these systems, but it could be used to enhance the capabilities of such devices. Systems with both analog inputs and outputs could potentially be synthesized completely within the FPAA. However, systems with digital outputs, such is the case with digital displays and wireless transmitters, could use the FPAA to perform analog preprocessing on the input signal before digitizing it. By incorporating these devices into embedded systems laboratory exercises, students would be able to explore these analog

possibilities in addition to the digital integration already being used.

8.1 First-Generation Educational FPAA Board

The first attempt at introducing a large-scale FPAA into the class environment was using the conceptual setup depicted in Figure 8.1. The class was a senior level analog IC design course, and this laboratory bench configuration is fairly typical for analog IC testing, except for the FPGA and FPAA boards. MATLAB or a similar data manipulation software package is commonly used to control bench test equipment or PC instrumentation cards, which are used to generate input signals for the device-under-test (DUT) and to observe the outputs. The simplest way to incorporate an FPAA into this existing laboratory environment was as the DUT. A board was thus designed to handle some of the programming and basic I/O interfacing for the RASP 2.5 FPAA. A commercial FPGA board was also used to precisely control the analog programming circuitry timing on the FPAA board and to provide local digital signals to the FPAA IC.

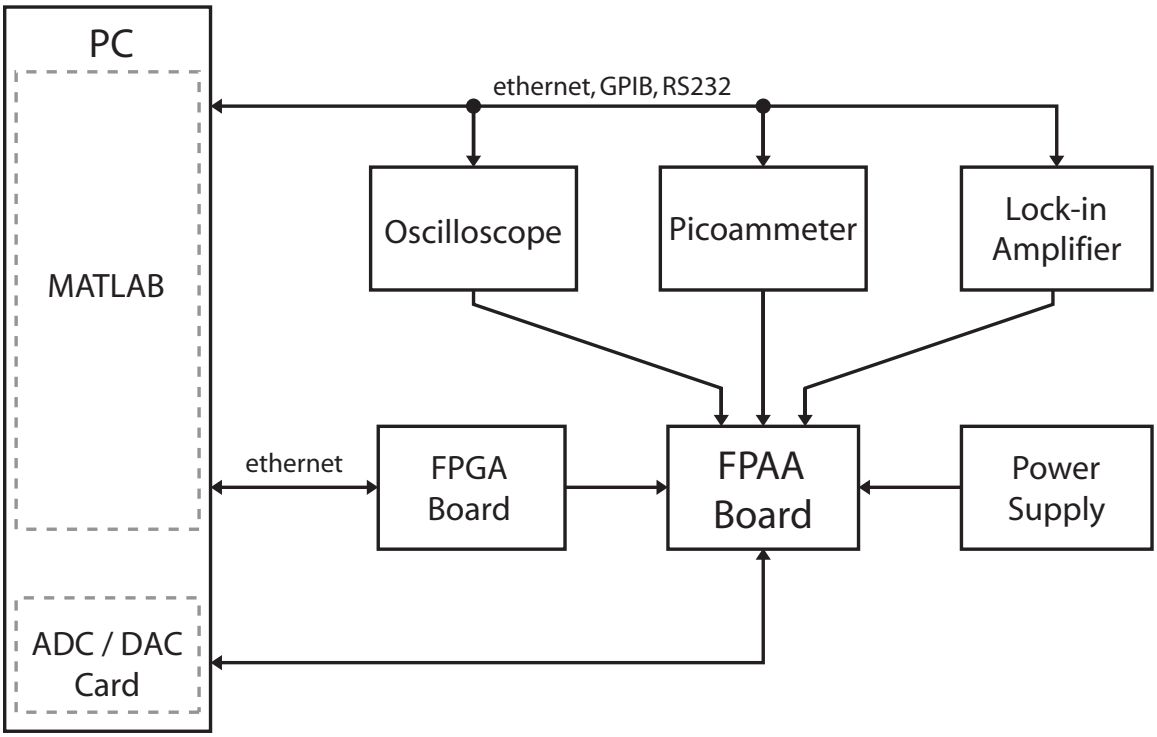


Figure 8.1. Educational laboratory setup using the RASP 2.5 FPAA.

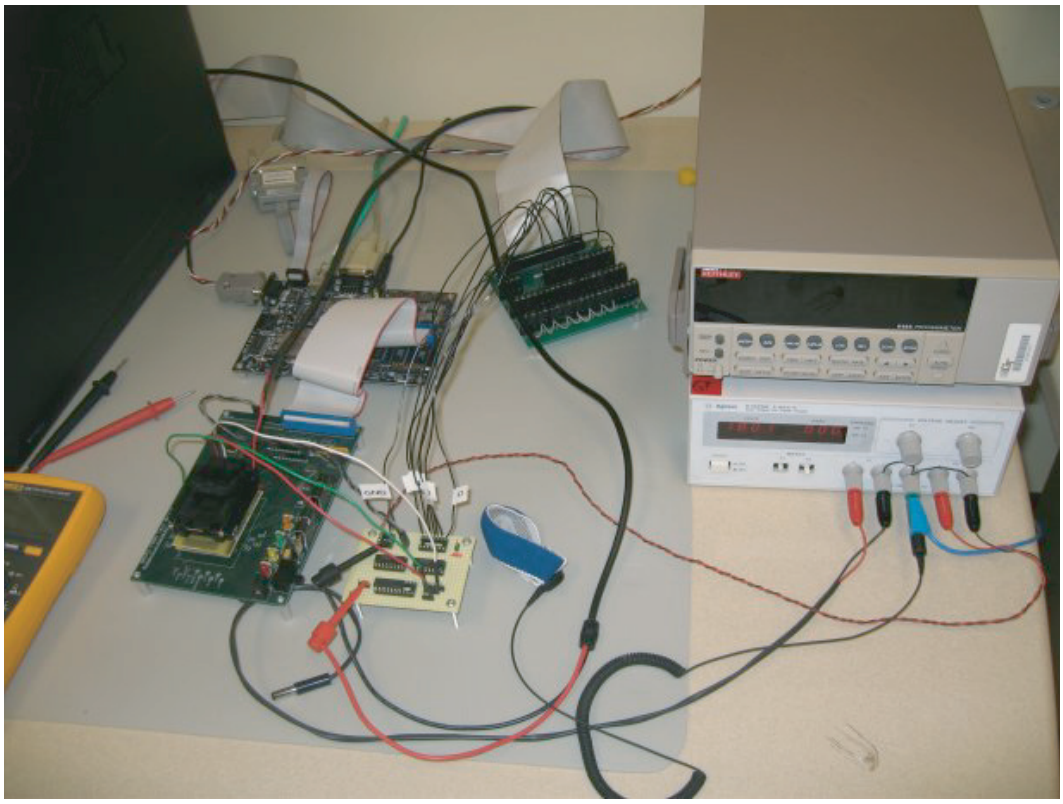


Figure 8.2. Laboratory setup used to prototype and design analog circuits on an FPAA.

Figure 8.2 depicts the FPAA laboratory setup used for various workshops and a class at Georgia Tech. The FPGA board can be seen at the top left of Figure 8.2. A synthesized Nios processor core was used to interface with the FPAA boards and the picoammeter, located on top of the power supply, as seen on the right side of Figure 8.2. Communication between MATLAB running on the PC and the Nios processor was achieved using TCP/IP over a direct ethernet cable between the PC and FPGA board. The FPAA board is located in the lower left of Figure 8.2 and is connected to the FPGA board via a ribbon cable.

In addition to bench test equipment, a PC DAC/ADC card was used for test signal generation and measurement. The interface board is seen at the top of the figure towards the center and connects to the PC via a large ribbon cable. During initial experiments involving students, several RASP 2.5 ICs were damaged because they lacked disconnection circuitry for programming. Test signals applied to the I/O pins connected directly to the array drain lines, which interfered with programming. In response to this, the small wire-wrap board

```
prog(1);
erase;

select(56 + 11, 252 + 3);
recover;
program(1e-9);

sprogram_col(56 + [13 14], 252 + 23);
sprogram_col(56 + [12], 252 + 5);

prog(0)
```

Figure 8.3. RASP 2.5 FPAA board interface commands.

seen at the bottom of Figure 8.2 was added as a buffer between the FPAA board and the DAC/ADC interface board. This slowed down the rate at which FPAA ICs were damaged, but did not completely resolve the issue, since the students still had to make the connections correctly between the FPAA and interface boards.

To control the FPAA board and program the RASP 2.5 IC, MATLAB commands, such as that seen in Figure 8.3, are used. This code example programs the follower circuit from Figure 4.6. The user is given a higher level of commands which erase, select, and program the individual floating gate biases and switches. These high level commands call low level routines in MATLAB and on the Nios processor. This abstracts the details of the floating gate transistor programming such that the user can concentrate on the circuit and system levels.

An early version of the setup depicted in Figure 8.2 was tested at the 2005 Telluride Workshop on Neuromorphic Engineering located in Telluride, Colorado. Participants came from diverse backgrounds including engineering, biology, and computational neuroscience. Most of these individuals had not designed or tested an analog IC before, so they were ideal candidates for learning this new approach to analog design. Within three weeks this group of individuals had completed many of the lab exercises characterizing the various CAB components. Using the fuse-plot sheets depicted in Figure 5.5, these individuals

routed circuits given to them in schematic form and tested them using the laboratory bench equipment. By the third week, several of the participants had begun to compile their own circuits from the characterized CAB components in an attempt to integrate these FPAAs within their own research.

With the success at the Telluride workshop, the FPAA was then integrated into a senior level analog IC design class. Instead of analyzing discrete components and circuits, the class was given FPAA synthesized circuits to characterize for laboratory exercises, like the transistor characterization circuit in Figure 8.4. Example data sweeps of the pFET are depicted in Figure 8.5. Using the synthesized pFET, students are able to extract the same parameters as they could with a custom IC or discrete component. Figure 8.5a shows a drain sweep with the pFET biased in the sub-threshold region. The thermal voltage, U_T, can be extracted from the sub-threshold slope of the source sweep, as seen in Figure 8.5b.

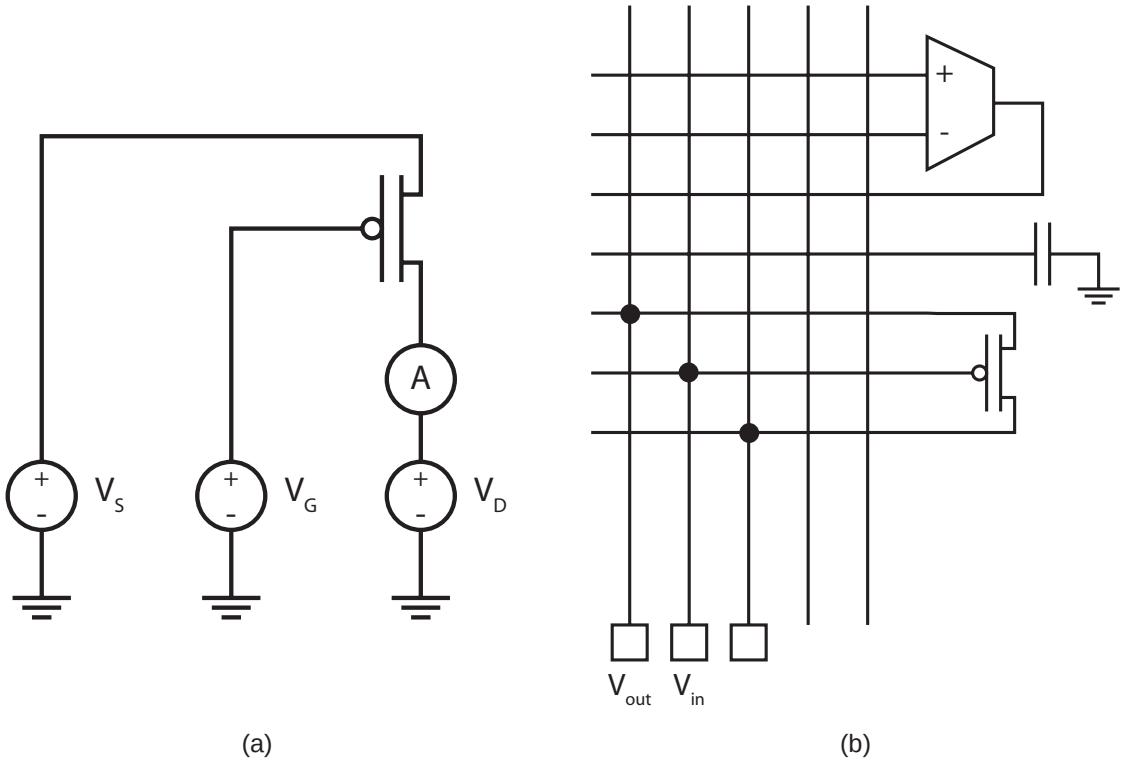


Figure 8.4. Transistor characterization using a pFET within an FPAA CAB.
(a) Circuit used to measure and characterize the pFET.
(b) Routing used to connect the pFET within the RASP FPAA.

From gate sweeps in Figures 8.5c and 8.5d, the sub-threshold κ and the above-threshold V_{th} can be calculated. In a similar manner, many basic circuit components were characterized.

One interesting effect observed in Figure 8.5d is the switch resistance current limiting the gate sweep. At low gate voltages, those below .5 V in Figure 8.5d, the switches clamp the current that would normally flow through the pFET being tested. Since most circuits in these FPAAs were designed to operate in the sub-threshold region or just above threshold, this current limitation should not be a problem in the general case.

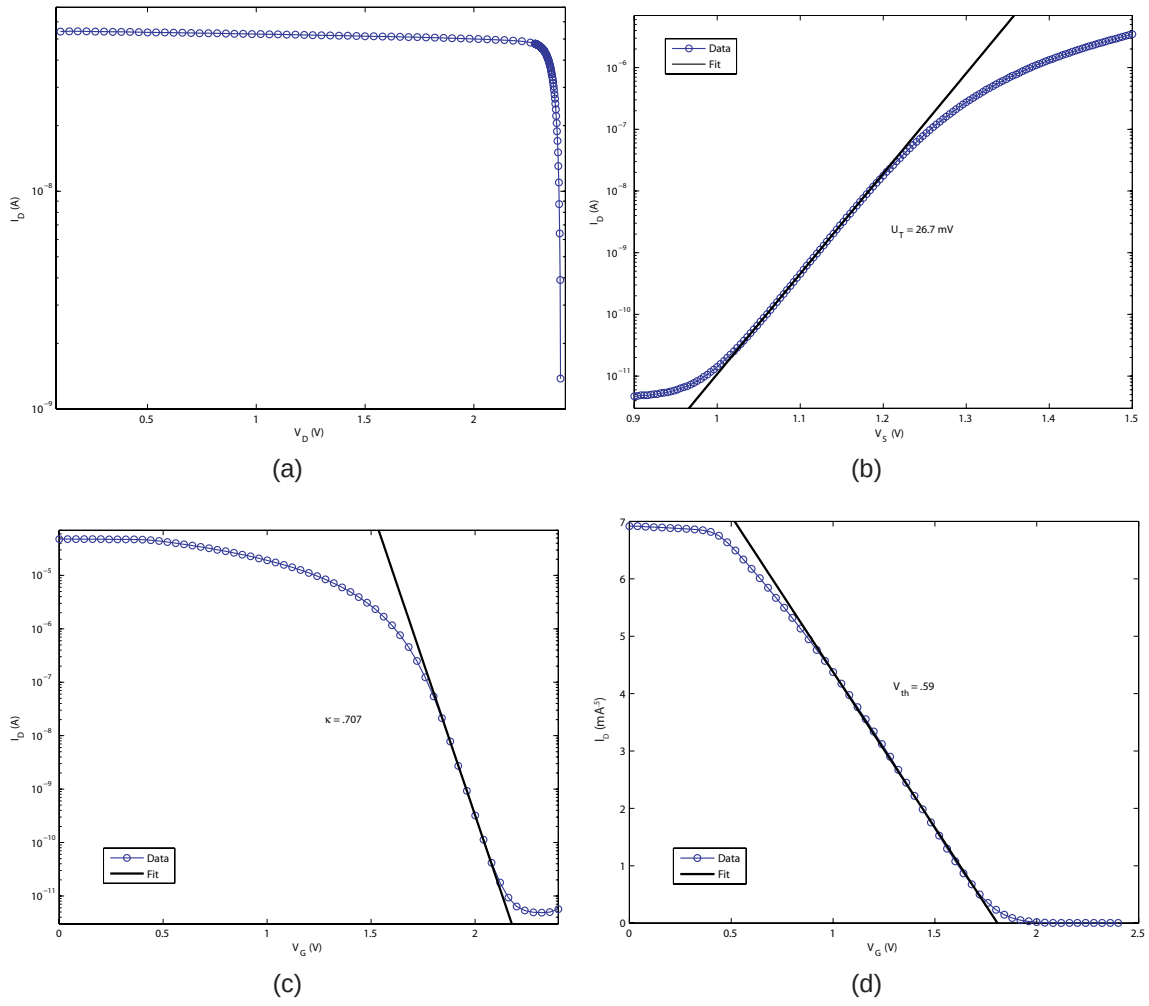


Figure 8.5. Characterizing a pFET using the educational setup.
(a) Drain sweep
(b) Source sweep
(c) Gate sweep showing logarithmic sub-threshold region
(d) Gate sweep showing square root above-threshold region

8.2 Second-Generation Educational FPAA Board

A second-generation FPAA board was designed around the RASP 2.7 IC to reduce the dependence upon expensive laboratory bench equipment and PC instrumentation boards, as seen in Figure 8.6. In the previous setup, current measurements for programming were handled by a picoammeter residing on the test bench. On the new board, these current measurements are made using a logarithmic amplifier IC and an ADC chip. Additional DACs and an ADC were added to eliminate the need for the PC DAC/ADC cards. Audio coupling circuitry was also added to aid in audio signal processing applications. As an additional precaution, isolation circuitry was added to the I/O pins of the FPAA to prevent external signals from interfering with the programming signals and thereby damaging the RASP IC.

For portability, the FPAA and FPGA boards were integrated within a box, as seen in Figure 8.7. The FPGA board is located in the top left corner, and the FPAA board, located in the center of the figure, connects directly to one of the FPGA board's I/O headers. A power supply board, located in the top right of Figure 8.7, was designed and fabricated to minimize the number of connections required to operate this portable laboratory setup.

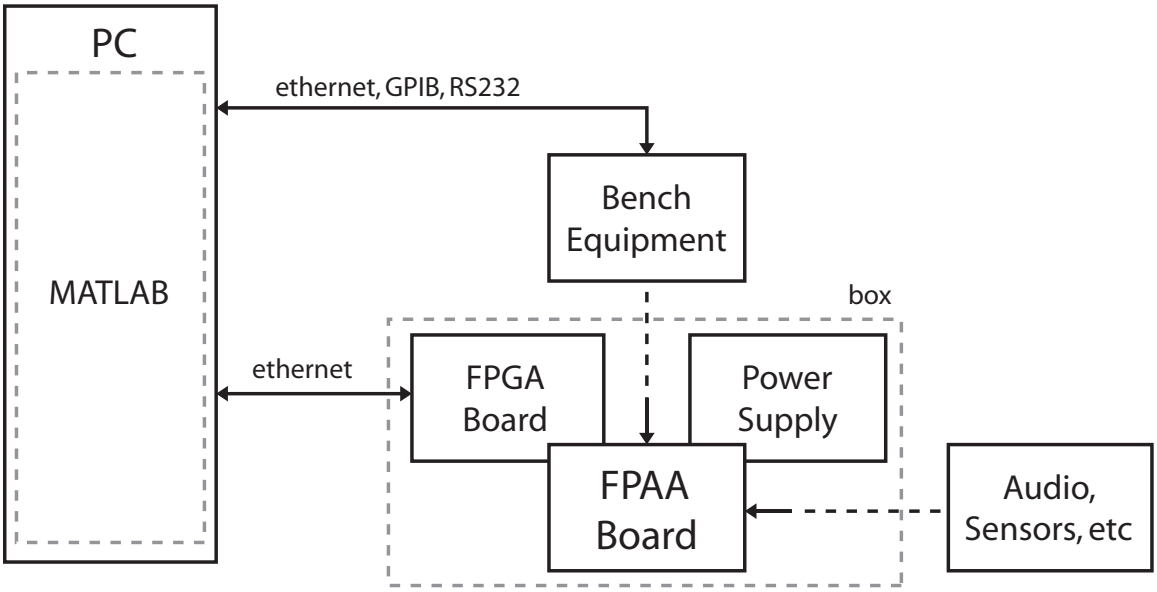


Figure 8.6. Educational laboratory setup using the RASP 2.7 FPAA.

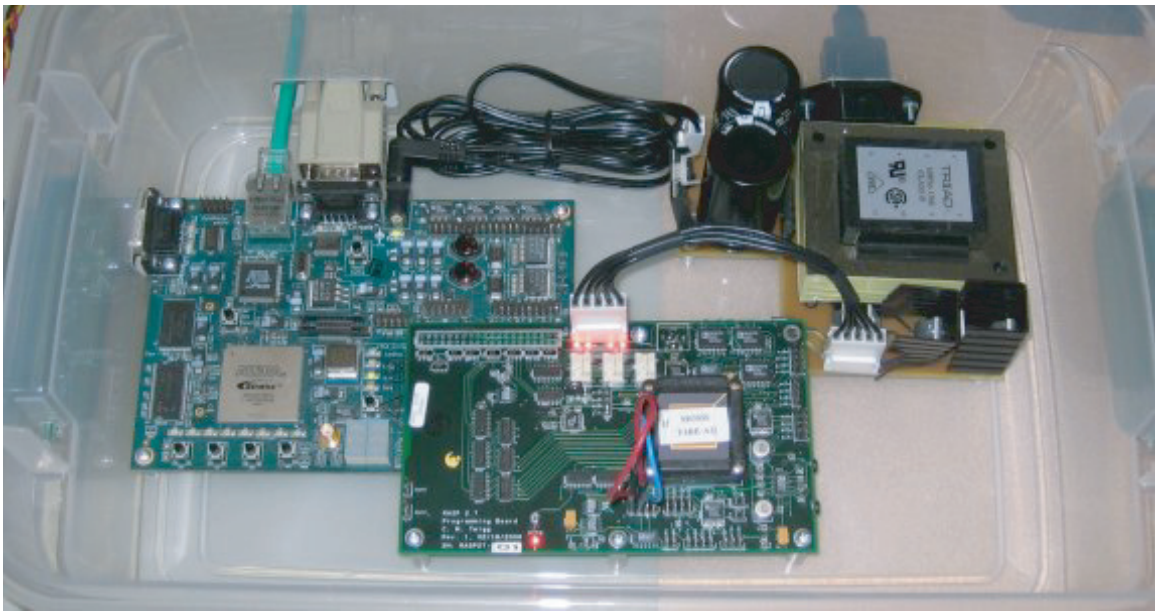


Figure 8.7. Portable FPAA laboratory in a box.

With this setup, most laboratory exercises can be performed with only an ethernet cable, a power cable, and a PC running MATLAB. No external bench equipment is required for programming or general testing, but some items, like lock-in amplifiers and spectrum analyzers, may still be used for measurements requiring higher accuracy or special functions than are available with the on-board instrumentation ICs.

The high level MATLAB commands were also updated for this new setup, as seen in Figure 8.8. Instead of issuing individual programming commands, the user now only needs to point the programming algorithm to a configuration file. As seen in Figure 8.8b, the program file contains no MATLAB commands, only floating gate transistor locations and bias currents, unless the transistor is being programmed as an "on" switch. This format is particularly useful when translating the fuse-plot drawings into global addresses for programming. Further improvements to the programming interface have also been made by collaborative researchers at Georgia Tech, who have developed routing tools to map netlists to the RASP architecture [45–47]. An open source schematic capture tool has even been modified to support the RASP 2.7 functional blocks, as seen in Figure 8.9.

In the laboratory and workshop environments, many circuits have been synthesized

```
>> program('follower');
```

(a)

```
% follower.prg
67 255 10e-9   % A2 OTA 1 bias

68 274         % A2 OTA 1 vp
69 275         % A2 OTA 1 vn
70 275         % A2 OTA 1 vout
```

(b)

Figure 8.8. RASP 2.7 FPAA board interface commands.
(a) MATLAB command window executing the 'follower' program.
(b) Contents of the 'follower' program file.

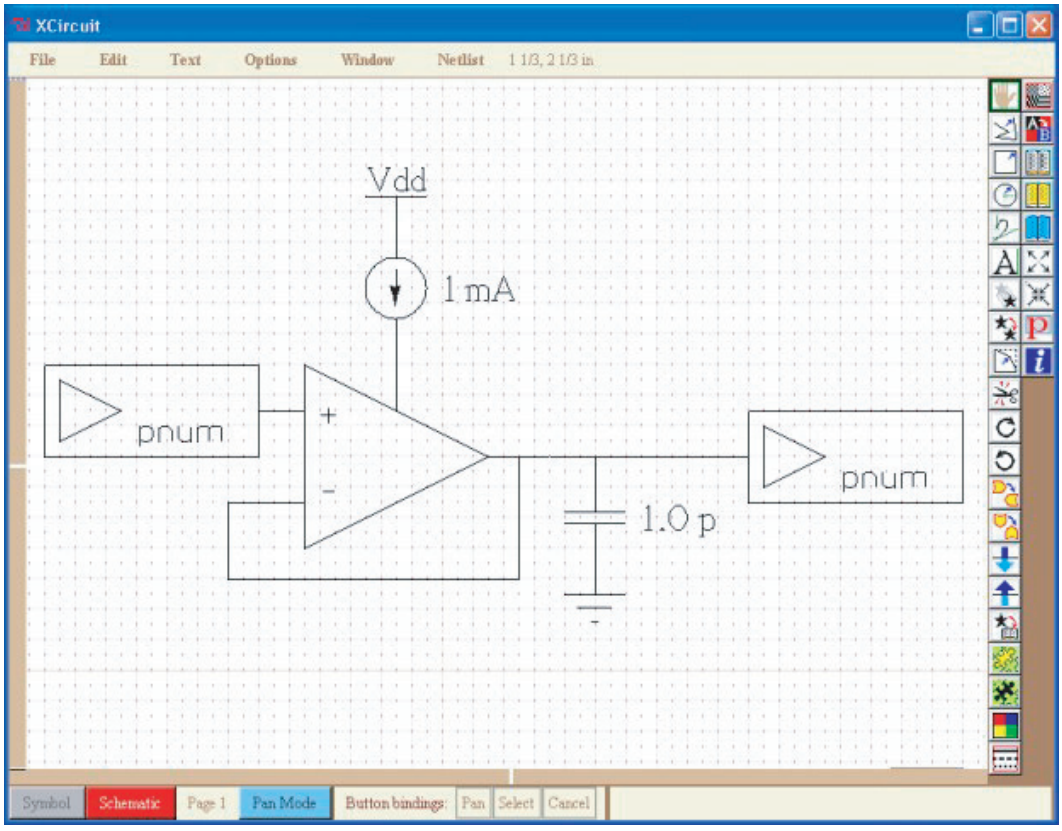


Figure 8.9. Xcircuit schematic capture tool.

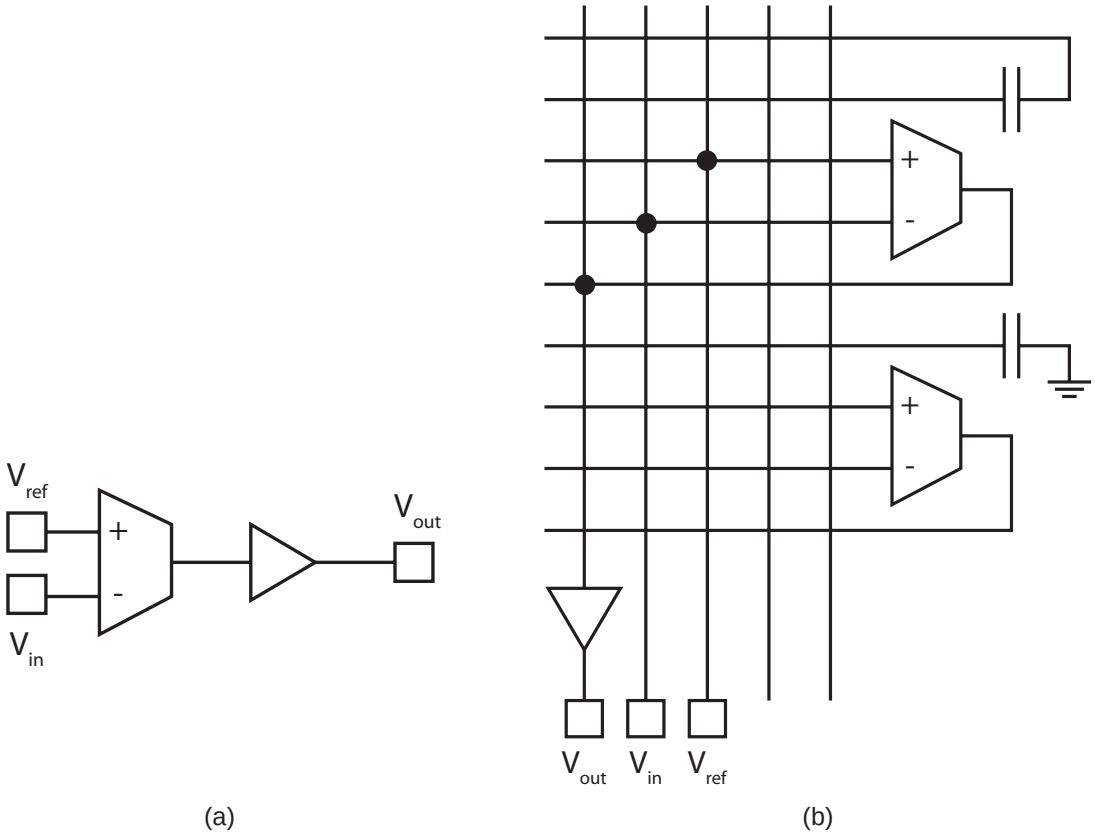


Figure 8.10. Comparator circuit synthesized on the RASP 2.7.
(a) Circuit schematic.
(b) FPAA implementation showing "on" switches.

and measured using only the resources available on the portable FPAA laboratory setup. Figure 8.10 shows the schematic and FPAA mappings used to compile a simple inverting comparator structure. The reference terminal sets the comparator trip point, which can also be set by the switch fabric voltage reference of Figure 5.15 to save an I/O pin. For characterization purposes, the reference voltage was set using one of the on-board DACs. The input voltage was swept using another on-board DAC while the output was measured using the on-board ADC, as seen in Figure 8.11. The follower circuit of Figure 4.6 was also synthesized and measured. Figure 8.12a shows the result of sweeping the input and measuring the output using the on-board DAC and ADC. From this transfer function, the gain of this OTA topology was calculated, as seen in Figure 8.12b. These results demonstrate that typical analog laboratory exercises can be performed using only the portable FPAA

setup.

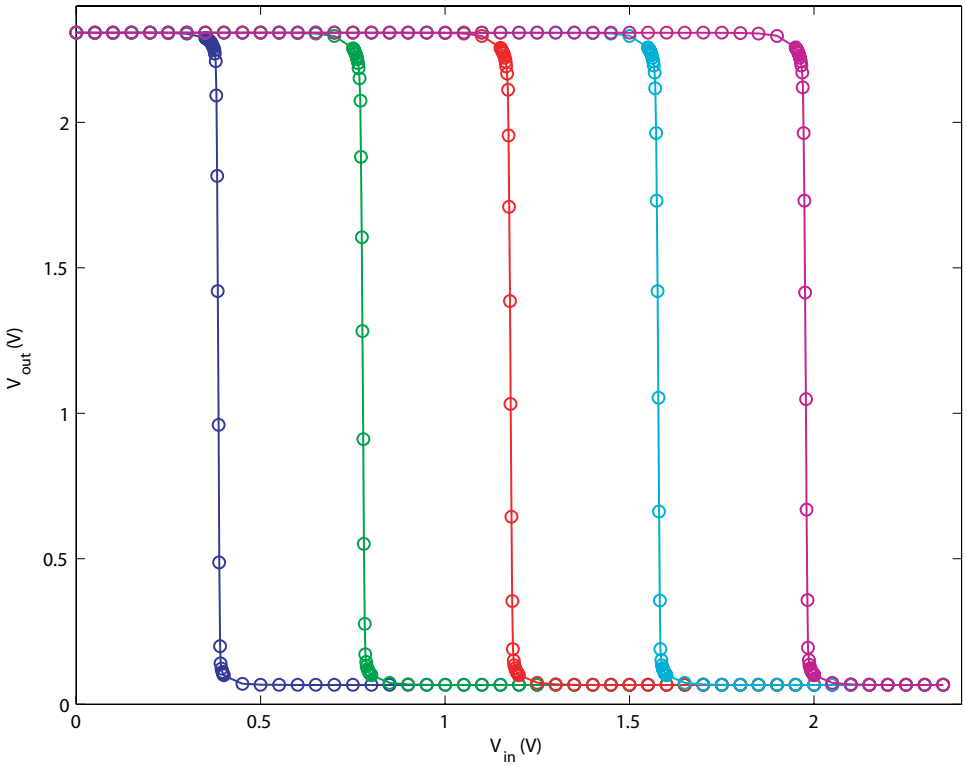


Figure 8.11. Results from a simple comparator synthesized using the RASP 2.7 FPAA board.

8.3 Next-Generation Educational FPAA Board

The next-generation educational FPAA board is planned to integrate even more functionality onto a single board, as seen in Figure 8.13. The planned RASP 2.8 FPAA will have integrated much of the programming circuitry currently on the RASP 2.7 board onto the IC. This will free up a significant portion of the board area and enable more instrumentation circuits to be integrated on-board. This board will also feature a much more power efficient microcontroller with USB or ethernet support built directly into the chip. This will dramatically reduce the power budget of the entire laboratory setup and should enable the possibility of using battery power. Battery operation would significantly improve the portability of the system, which could lead to in-class laboratory exercises using physical hardware.

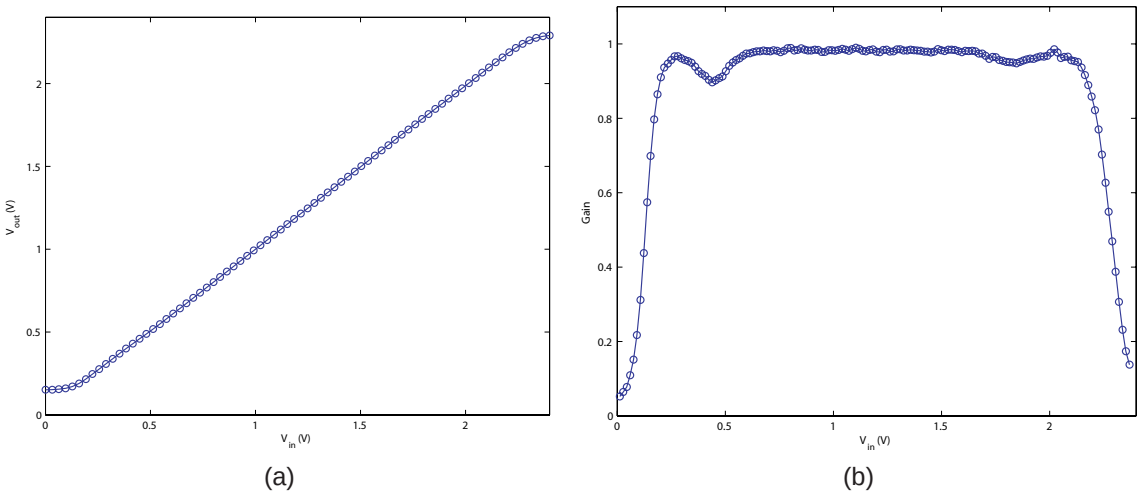


Figure 8.12. Results from a follower synthesized on the RASP 2.7 IC.
(a) DC transfer function.
(b) Gain measured from the output of the DC sweep.

A logarithmic amplifier and ADC combination have been included for on-board current measurement, as was the case for the RASP 2.7 board. However, this circuit would be dedicated to testing circuits synthesized on the FPAA rather than floating gate transistor programming, since the programming circuitry will be integrated within the FPAA. A direct

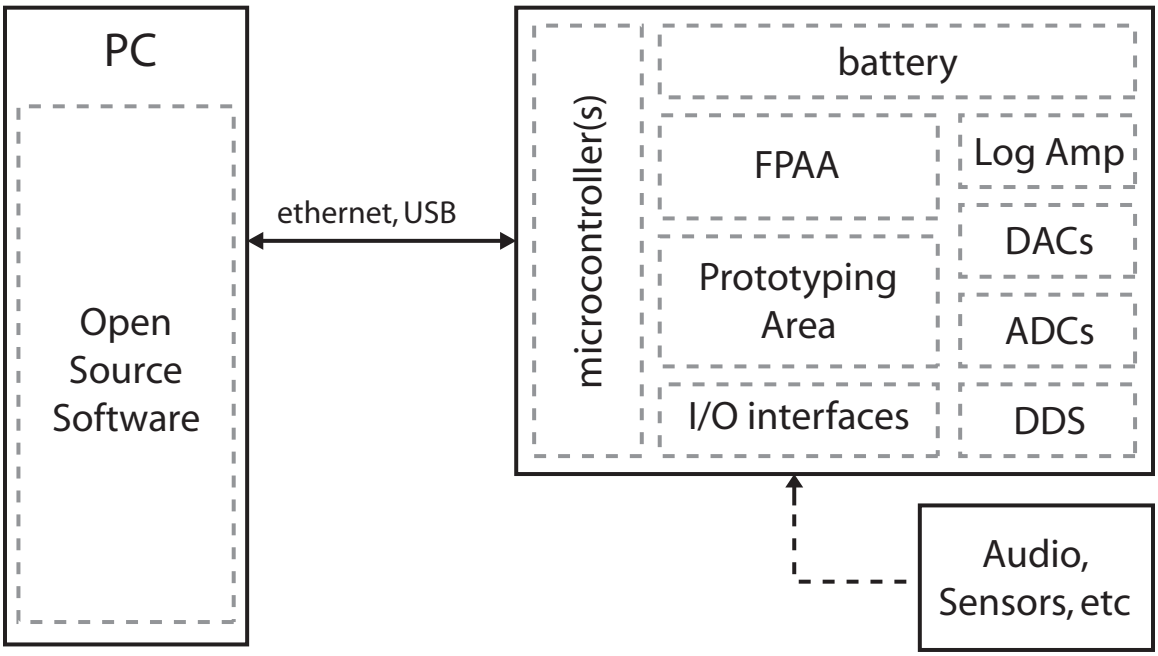


Figure 8.13. Future educational laboratory setup using the planned RASP 2.8 FPAA.

digital synthesis (DDS) chip will also be included along with another couple of ADCs on this board in order to replace the bench lock-in amplifier for generating frequency/phase data. Additional I/O interface circuits will also be included, such as the audio circuitry used on the RASP 2.7 board design. Unlike previous generations, this new board will also feature a small prototyping area. This will allow the functionality of the board to be expanded using ICs and wires soldered directly to the board, which is a common feature of embedded systems development boards.

CHAPTER 9

FPAA DIRECTIONS

The FPAAs developed and discussed within this dissertation have demonstrated the feasibility of large-scale reconfigurable and programmable analog devices. However, the long-term goal for maximum flexibility and programmability is the mixed signal development system depicted in Figure 1.2b. In a cooperative analog/digital signal processing (CADSP) system such as this, analog and digital devices, such as sensors and displays, can be integrated using a single programmable device. Figure 9.1 shows the road map that links the FPAAs developed to date to the mixed signal IC of the future.

The first-generation devices proved the viability of FPAAs based upon floating gate transistors. These devices were fairly low in complexity and functionality, much like the commercial devices currently available. However, these commercial ICs are significantly

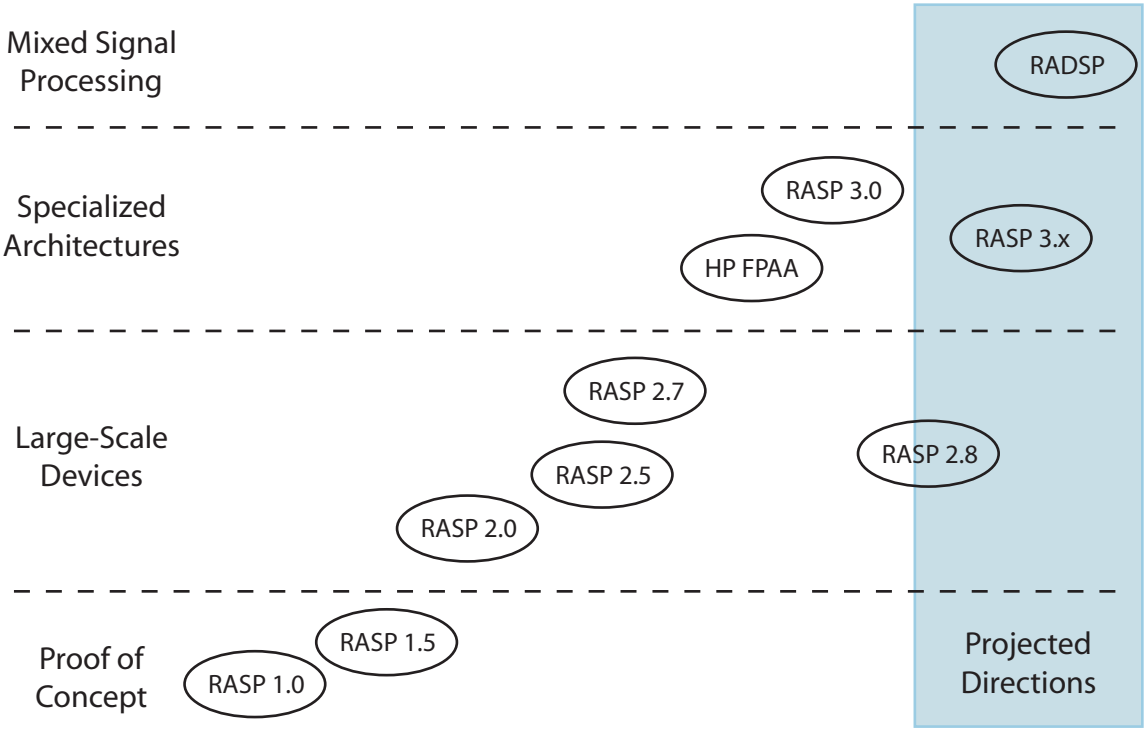


Figure 9.1. Road map for the RASP FPAA and beyond.

larger in die area than the RASP 1.x FPAAs. Scaling up the CAB array architecture led to the second-generation RASP FPAA. As a result of the analog component density provided by floating gate transistors, this generation of large-scale FPAAs is capable of synthesizing larger analog systems. Although this generation of FPAAs is a significant step toward large-scale mixed signal reconfigurable systems, there are still a few issues to resolve.

The RASP 2.8 FPAA is the next revision of the second-generation RASP FPAA. This IC will contain a number of improvements over the RASP 2.5 and 2.7. The number of different CAB types will be increased to better reflect component utilization observed while working with systems on the RASP 2.7 IC. A general purpose CAB containing OTAs, transistors, and capacitors will still comprise the majority of the FPAA. More specialized CABs containing optimized circuits, such as signal-by-signal multipliers and vector-matrix multipliers, will be distributed throughout the CAB array. The programming structures currently found on the RASP 2.7 educational board will be integrated on the IC to improve accuracy and speed. On-chip data converters allow direct interactions between the FPAA and an FPGA or microcontroller. This will provide a mixed signal development system integrated at the board level, which is another step closer to a mixed signal reconfigurable IC.

The third-generation RASP FPAAs are just entering the testing phase and are distinguished from previous generations by the specialized hardware blocks included within the CAB array. The inspiration for these ICs again came from digital parallels. In modern FPGAs, it is fairly common for highly specialized hardware blocks, such as memory cells, DSP functions, and microprocessors to be included on the same die connected to the interconnect network. Some FPGAs even include basic analog blocks on the periphery of the IC for very simple analog preprocessing [48]. Using this idea, specialized components, such as the programmable filter bank and capacitively coupled linear combiner, have been included in the RASP 3.0 for audio signal processing applications. Several future RASP 3.x FPAAs for adaptive signal processing and neuron interconnection modeling are currently

being planned.

The fourth-generation device will no longer be part of the RASP family of FPAAs. Instead, this IC is planned to be the first mixed signal large-scale reconfigurable and programmable device based upon floating gate transistors. The reconfigurable analog/digital signal processor (RADSP) line will contain all the features of a large-scale FPAA coupled via data converters to an integrated FPGA structure. The initial design will most likely be two separate architectures, FPAA and FPGA, which will have their own I/O lines. Future revisions of the IC will investigate integrating the two devices within the same array architecture, which should provide greater coupling between analog and digital as well as more I/O pin flexibility. The successful demonstration of this device will mark a significant achievement in mixed signal processing hardware and could fundamentally alter the way in which hardware design is approached.

REFERENCES

[1] G. Frantz, "Digital signal processor trends," *IEEE Micro*, vol. 20, pp. 52–59, Nov–Dec 2000.

[2] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531nw/mhz, 128 x 32 current–mode programmable analog vector–matrix multiplier with over two decades of linearity," in *IEEE Proceedings of the Custom Integrated Circuits Conference*, pp. 651–654, 2004.

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

[4] P. Hasler, P. Smith, R. Ellis, D. Graham, and D. V. Anderson, "Biologically inspired auditory sensing system interfaces on a chip," in *Proceedings of IEEE Sensors*, pp. 669–674, 2002.

[5] P. D. Smith, M. Kucic, R. Ellis, P. Hasler, and D. V. Anderson, "Mel-frequency cepstrum encoding in analog floating–gate circuitry," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, vol. 4, pp. 671–674, 2002.

[6] T. S. Hall, C. M. Twigg, P. Hasler, and D. V. Anderson, "Developing large–scale field–programmable analog arrays for rapid prototyping," *International Journal of Embedded Systems*, vol. 1, no. 3/4, pp. 179–192, 2005.

[7] R. T. Edwards, K. Strohbehn, and S. E. Jaskulek, "A field–programmable mixed–signal array architecture using antifuse interconnects," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, vol. 3, pp. 319–322, 2000.

[8] 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*, pp. 198–199, Feb 1995.

[9] J. Becker and Y. Manoli, "A continuous–time field programmable analog array (FPAA) consisting of digitally reconfigurable G_M–cells," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 1092–1095, 2004.

[10] D. Anderson, C. Marcjan, D. Bersch, H. Anderson, P. Hu, O. Palusinski, D. Gettman, I. Macbeth, and A. Bratt, "A field programmable analog array and its application," in *IEEE Proceedings of the Custom Integrated Circuits Conference*, May 1997.

[11] Fast Analog Solutions Ltd., http://www.zetex.com, *Totally re-configurable analog circuit – TRAC®*, Mar 1999.

[12] Anadigm, http://www.anadigm.com/_doc/DS030100–U008.pdf, *Dynamically Reconfigurable FPAA With Basic I/O*, Nov 2005.

[13] H. W. Klein, "The EPAC architecture: an expert cell approach to field programmable analog circuits," in *IEEE Proceedings of the Midwest Symposium on Circuits and Systems*, vol. 1, pp. 169–172, Aug. 1996.

[14] E. Sackinger and W. Guggenbuhl, "An analog trimming circuit based on a floating–gate device," *IEEE Journal of Solid–State Circuits*, vol. 23, pp. 1437–1440, Dec 1988.

[15] L. R. Carley, "Trimming analog circuits using floating–gate analog MOS memory," *IEEE Journal of Solid–State Circuits*, vol. 24, pp. 1569–1574, Dec 1989.

[16] V. Srinivasan, G. J. Serrano, J. Gray, and P. Hasler, "A precision CMOS amplifier using floating–gates for offset cancellation," in *IEEE Proceedings of the Custom Integrated Circuits Conference*, pp. 739–742, 2005.

[17] P. Brady and P. Hasler, "Offset compensation in flash ADCs using floating–gate circuits," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 6154–6157, 2005.

[18] P. Hasler, B. A. Minch, and C. Diorio, "Floating–gate devices: they are not just for digital memories any more," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, vol. 2, pp. 388–391, 1999.

[19] V. Srinivasan, G. J. Serrano, C. M. Twigg, and P. Hasler, "A compact programmable CMOS reference with $\pm 40\mu$V accuracy," in *IEEE Proceedings of the Custom Integrated Circuits Conference*, p. accepted, 2006.

[20] E. Ozalevli, C. M. Twigg, and P. Hasler, "10–bit programmable voltage–output digital–analog converter," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 5553–5556, 2005.

[21] A. Thomsen and M. A. Brooke, "A temperature stable current reference source with programmable output," in *IEEE Midwest Symposium on Circuits and Systems*, vol. 2, pp. 831–834, 1992.

[22] E. Farquhar, C. Duffy, and P. Hasler, "Practical issues using e–pot circuits," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, vol. 5, pp. 493–496, 2002.

[23] T. S. Hall, C. M. Twigg, J. D. Gray, P. Hasler, and D. V. Anderson, "Large–scale field–programmable analog arrays for analog signal processing," *IEEE Transactions on Circuits and Systems I*, vol. 52, pp. 2298–2307, Nov. 2005.

[24] S. Vlassis and S. Siskos, "Design of voltage–mode and current–mode computational circuits using floating–gate mos transistors," *IEEE Transactions on Circuits and Systems I*, vol. 51, pp. 329–341, Feb. 2004.

[25] Advanced Linear Devices Inc., http://www.aldinc.com/ald_przerothreshold.htm, *ALD revolutionizes analog design with zero threshold™MOSFETS*, May 2006. Press Release.

[26] C. Bleiker and H. Melchior, "A four–state EEPROM using floating–gate memory cells," *IEEE Journal of Solid–State Circuits*, vol. 22, pp. 460–463, Jun 1987.

[27] G. Serrano, P. Smith, H. J. Lo, R. Chawla, T. Hall, C. M. Twigg, and P. Hasler, "Automatic rapid programming of large arrays of floating–gate elements," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 373–376, 2004.

[28] A. Bandyopadhyay, G. J. Serrano, and P. Hasler, "Programming analog computational memory elements to 0.2% accuracy over 3.5 decades using a predictive method," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 2148–2151, 2005.

[29] P. D. Smith, M. Kucic, and P. Hasler, "Accurate programming of analog floating–gate arrays," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 489–492, 2002.

[30] J. D. Gray, C. M. Twigg, D. N. Abramson, and P. Hasler, "Characteristics and programming of floating-gate pFET switches in an FPAA crossar network," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 468–471, 2005.

[31] D. W. Graham, E. Farquhar, B. Degnan, C. Gordon, and P. Hasler, "Indirect programming of floating–gate transistors," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 2172–2175, 2005.

[32] A. E. Buck, C. L. McDonald, S. H. Lewis, and T. R. Viswanathan, "A CMOS bandgap reference without resistors," *IEEE Journal of Solid–State Circuits*, pp. 81–83, Jan 2002.

[33] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw–Hill Series in Electrical and Computer Engineering, McGraw–Hill, 2001.

[34] B. S. Song and P. R. Gray, "A precision curvature–compensated CMOS bandgap reference," *IEEE Journal of Solid–State Circuits*, pp. 634–643, Dec 1983.

[35] R. J. Baker, *CMOS Circuit Design, Layout, and Simulation*. IEEE Press Series on Microelectronic Systems, John Wiley & Sons, second ed., 2005.

[36] H. Nozama and S. Kokyama, "A thermionic electron emission model for charge retention in SAMOS structures," in *Japanese Journal of Applied Physics*, vol. 21, pp. L111–L112, Feb 1992.

[37] A. Stoica, R. Zebulum, D. Keymeulen, R. Tawel, T. Daud, and A. Thakoor, "Reconfigurable VLSI architectures for evolvable hardware: from experimental field programmable transistor arrays to evolution–oriented chips," *IEEE Transactions on Very Large Scale Integration*, vol. 9, pp. 227–232, Feb 2001.

[38] P. D. Smith, D. W. Graham, R. Chawla, and P. Hasler, "A five–transistor bandpass filter element," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, vol. 1, pp. 861–864, 2004.

[39] G. E. R. Cowan, R. C. Melville, and Y. P. Tsividis, "A VLSI analog computer/digital computer accelerator," *Journal of Solid–State Circuits*, vol. 41, no. 1, pp. 42–53, 2006.

[40] J. Choma, "Transconductor applications." EE533 Lecture 07 Powerpoint Presentation, 2002.

[41] D. W. Graham, P. D. Smith, R. Ellis, R. Chawla, and P. Hasler, "A programmable bandpass array using floating–gate transistors," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 97–100, 2004.

[42] B. A. Minch, "Synthesis of static and dynamic multiple–input translinear element networks," in *IEEE Transactions on Circuits and Systems I*, vol. 51, pp. 409–421, Feb 2004.

[43] B. A. Minch, "Construction and transformation of multiple–input translinear element networks," in *IEEE Transactions on Circuits and Systems I*, vol. 50, pp. 1530–1537, Dec 2003.

[44] H. J. Lo, G. Serrano, P. Hasler, D. V. Anderson, and B. Minch, "Programmable multiple input translinear elements," in *IEEE Proceedings of the International Symposium on Circuits and Systems*, pp. 757–760, 2004.

[45] F. Baskaya, S. Reddy, S. K. Lim, and D. V. Anderson, "Placement for large–scale floating–gate field programable analog arrays," *IEEE Transactions on Very Large Scale Integration Systems*, vol. 14, no. 8, 2006.

[46] F. Baskaya, S. Reddy, S. K. Lim, T. S. Hall, and D. V. Anderson, "Mapping algorithm for large–scale field programmable analog array," in *ACM International Symposium on Physical Design*, pp. 152–158, 2005.

[47] F. Baskaya, S. Reddy, S. K. Lim, and D. V. Anderson, "Hierarchical placement for large–scale fpaa," in *International Conference on Field Programmable Logic and Applications*, pp. 421–426, 2005.

[48] Actel, http://www.actel.com/products/fusion/, *Actel: Flash Devices: Fusion Programmable System Chip*, July 2006. Product Description.