Indirect Programming of Floating-Gate Transistors

David W. Graham, *Member, IEEE*, Ethan Farquhar, *Member, IEEE*, Brian Degnan, *Student Member, IEEE*, Christal Gordon, *Member, IEEE*, and Paul Hasler, *Senior Member, IEEE*

***Abstract*—Floating-gate (FG) transistors are useful for precisely programming a large array of current sources. Present FG programming techniques require disconnection of the transistor from the rest of its circuit while it is being programmed. We present a new method of programming FG transistors that does not require this disconnection. In this indirect programming method, two transistors share a FG allowing one to exist directly in a circuit while the other is reserved for programming. Since the transistor does not need to be disconnected from the circuit to program it, the switch count is reduced, resulting in fewer parasitics and better overall performance. Additionally, the use of these indirectly programmed FG transistors allows a circuit to be tuned such that the effects of device mismatch are negated. Finally, the concept of run-time programming is introduced which allows a circuit to be recalibrated while it is still operating within its system.**

***Index Terms*—Analog programmability, electron tunneling, floating-gate (FG) nFET, FG programming, FG transistor, hot-electron injection, indirect programming.**

I. Introduction

FLOATING-GATE (FG) transistors have been shown to be very useful acting as precise current sources when directly programmed with a combination of hot-electron injection and Fowler–Nordheim tunneling [1]–[5]. Programming a large array of these FGs for analog purposes has previously required disconnection circuitry, such as transmission gates (T-gates), to remove each FG transistor from its circuit for a programming phase and then reconnecting it for a run-time phase [6]. However, the addition of a 2-to-1 multiplexer for every FG to be programmed can be costly. The process of disconnection can decrease the maximum speed of operation and overall accuracy while also increasing the required real estate and necessary supply overhead. To circumvent the problems associated with detaching the FG transistor, we introduce a new, noninvasive method of programming that eliminates the need for disconnection and instead uses an indirect method of programming.

The concept of indirect programming of FG transistors is illustrated in Fig. 1(a) and (b), and an early discussion with preliminary results was included in [7]. With this indirect programming technique, multiple MOSFETs share a common FG.

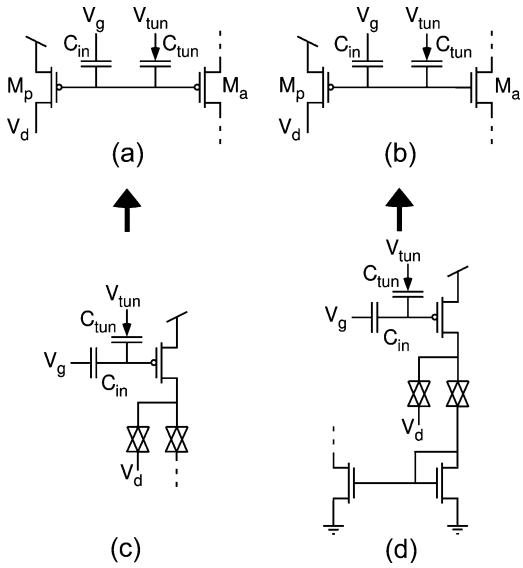

Fig. 1. (a) Programming structure of a pFET indirectly programming another pFET. The programmer transistor M_p is connected to the external programming structure and is actively programmed via hot-electron injection. The agent transistor M_a is connected to its circuit (represented by the dotted lines) and is passively programmed. (b) Programming structure of a pFET indirectly programming an nFET. (c) Direct method of programming a pFET. Direct programming requires disconnecting the pFET from the rest of the circuit with transmission gates (T-gates). This schematic represents a best-case scenario in which only two T-gates are required. For some applications, two T-gates each at the source and gate would also be required. (d) Direct method of programming an nFET. Direct programming requires programming the current in a pFET and then mirroring that current into the nFET that is connected to the circuit. For all shown FG transistors $V_{\rm tun}$ is used for tunneling the FG node or is set to a constant dc voltage in run mode equal to the voltage used when measuring the current.

One pFET is connected to the programming structure while the source and drain of the other transistor are connected to the respective circuit. The first pFET is programmed with hot-electron injection and tunneling using the method of [8]. Since the charge on this "programmer" pFET is modified, the current of the other transistor (the "agent") will also be set.

We present techniques for programming accurate currents with indirect programming and have fabricated several circuits for verification. All data presented in this paper were obtained from 0.5-μm processeses available through MOSIS.

The outline of this paper is as follows. In Section II, an overview of programming FG transistors is provided, and then in Section III, a current-mirror example is used to help illustrate why this new method of indirect programming is so important and useful in circuit design. Then, methods for indirectly programming pFETs and nFETS are given in Sections IV and V, respectively. Capacitive coupling onto the floating node is addressed in Section VI, and a method for resolving this issue when precisely tuning a circuit, along with a circuit example, is given in Section VII. Section VIII shows the benefits of

Manuscript received January 15, 2006; revised September 18, 2006. This paper was recommended by Associate Editor P. Carbone.

D. W. Graham is with the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown, WV 26506-6109 USA (david.graham@mail.wvu.edu).

E. Farquhar is with the Department of Electrical and Computer Engineering, University of Tennessee, Knoxville, TN 37916 USA (farquhar@ece.utk.edu).

B. Degnan and P. Hasler are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (degs@ece.gatech.edu; phasler@ece.gatech.edu).

C. Gordon is with the Department of Electrical and Computer Engineering, North Carolina State University, Raleigh, NC 27607 USA (christal_gordon@ncsu.edu).

Digital Object Identifier 10.1109/TCSI.2007.895521

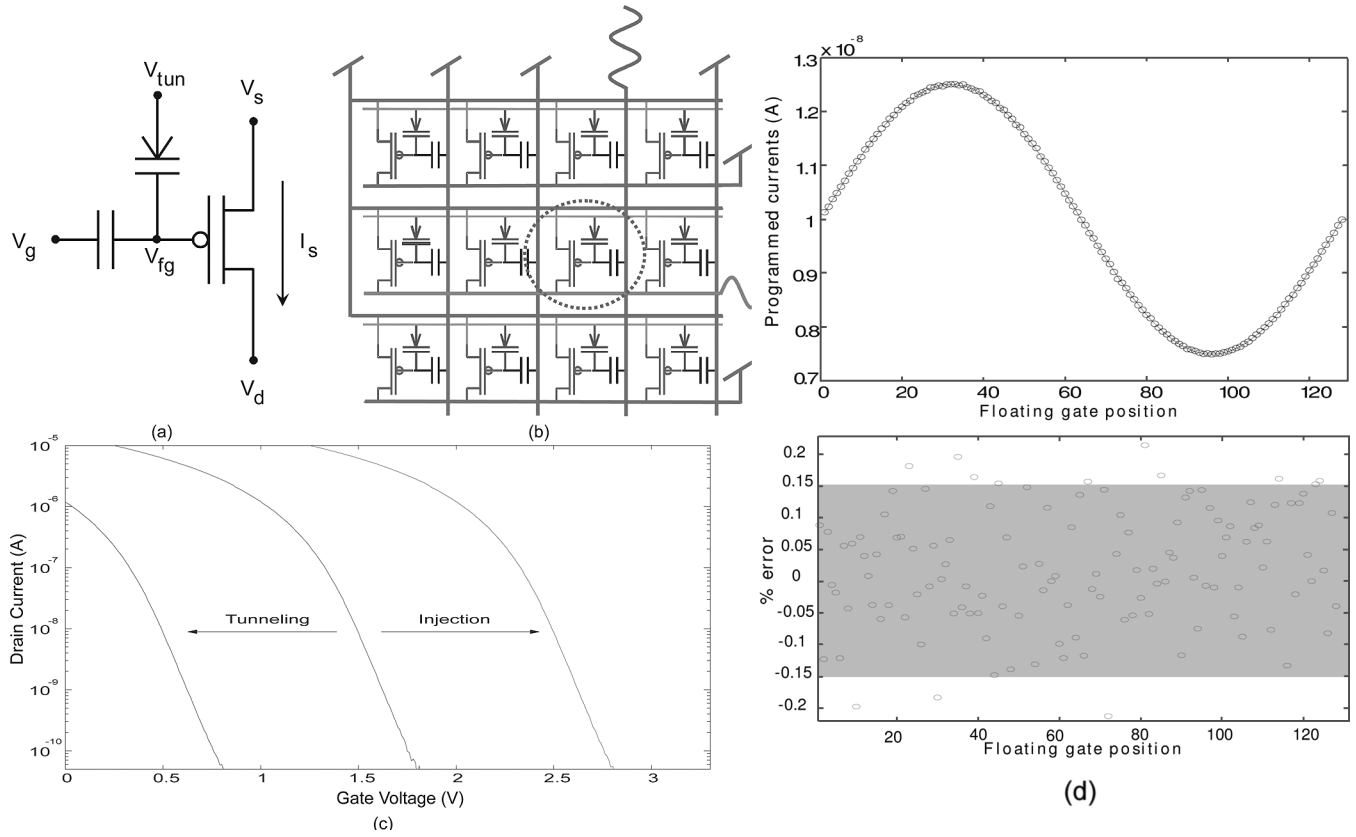

Fig. 2. Overview of FG transistors. A complete description of FG transistors can be found in [9]. (a) Single FG transistor. The gate is electrically isolated from the rest of the circuit by connecting only capacitors to it. The charge on the FG determines the current that flows through the transistor. This charge can be modified by programming using tunneling to remove electrons or hot-electron injection to add electrons to the FG. (b) Array of FG transistors. A large number of FG transistors can be programmed by placing them into an array. Hot-electron injection is used for accurate programming, and tunneling is used for erasure. (c) Movement of the threshold voltage. Tunneling increases the threshold voltage (with respect to V_{dd} since this is a pFET device). Injection decreases the threshold voltages. (d) Accurate programming with an array of FG devices [10].

indirect programming over traditional direct programming methods in terms of supply headroom, parasitics, and speed. The concept of run-time programming, which is enabled by indirect programming, is introduced in Section IX. Finally, this paper is summarized and conclusions are drawn in Section X.

II. Overview of FG Transistors

A single FG transistor, shown in Fig. 2(a), is simply a standard MOSFET device with only capacitors connected to the gate. Since the gate is electrically isolated due to oxide completely surrounding it, the charge on the gate is fixed and is responsible for establishing the amount of current flowing through the transistor. While the charge on the gate will not change on its own, that amount of charge can be modified by processes such as UV photo injection, Fowler–Nordheim tunneling, and hot-electron injection. The last two are the primary means of programming FG transistors to precise currents [6].

Through the process of electron tunneling, a large voltage is placed across a MOS capacitor. As this large tunneling voltage is increased, the effective width of the barrier is decreased, thus allowing electrons to breach the gap without adversely affecting the insulator. Tunneling is used to remove electrons from the FG in a controlled manner and, thus, raises the effective threshold voltage (referenced to V_{dd}), as is shown in Fig. 2(c).

Whereas tunneling is used to remove electrons from the FG, hot-electron injection is used to add electrons in a controlled manner. Hot-electron injection has two requirements. First, an appreciable amount of current must be flowing through the device. Second, a large source-to-drain voltage must be placed across the transistor. When both of these criteria are met, holes in a pFET flowing through the channel can build up sufficiently large energy to impact ionize an electron–hole pair. The resulting electron can have enough energy to pass through the insulator and onto the FG, thus adding electrons to the gate and therefore lowering the effective threshold voltage [Fig. 2(c)]. Since process-control parameters are set to stop injection from occurring in n-channel devices, only p-channel devices are used for FG programming [9].

To program a large amount of FG devices, FG transistors are arranged in an array for ease of programming, as is shown in Fig. 2(b) [6]. While tunneling can be used to program currents accurately, selectivity is not completely controllable in this arrangement. As a result, the tunneling operation is reserved for globally "erasing" the charges stored on the FGs and, therefore, serves as an initialization step. However, hot-electron injection allows complete selectivity of an individual element and is used for precise and accurate programming of FG arrays [6]. Selecting a particular device for injection involves connecting all unselected rows of drain lines to V_{dd} and all unselected columns

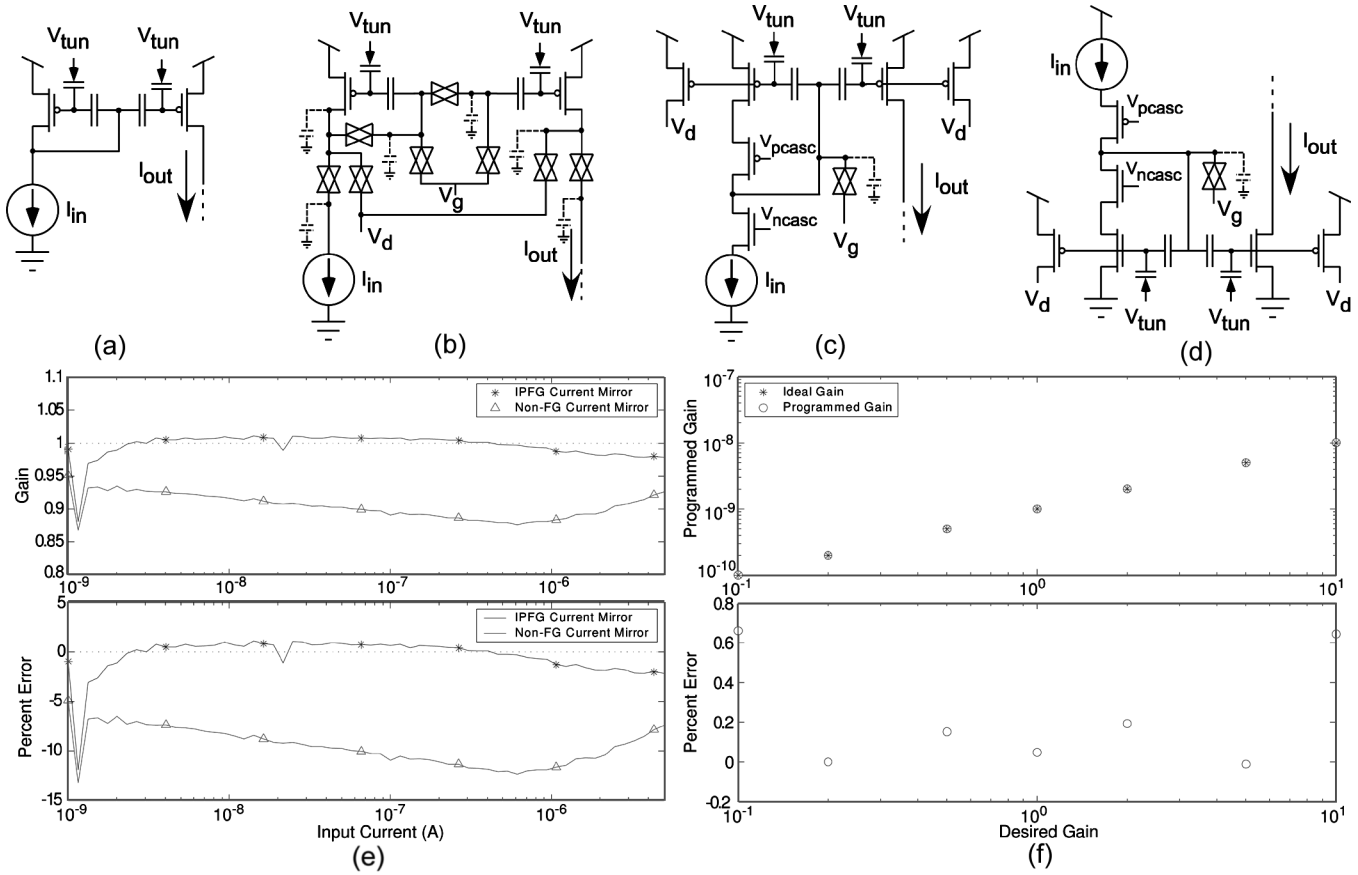

Fig. 3. (a) FG transistors for offset removal in a current mirror. (b) Implementation of the FG current mirror using direct FG programming techniques. To allow complete disconnection of each FG transistor for programming, many T-gate switches must be used which add parasitic capacitances (shown in dashed lines) and resistances. These switches increase the required area and supply headroom while concurrently degrading the operational performance. (c) Implementation of the FG current mirror with the indirect-programming technique. The use of indirectly programmed transistors greatly reduces the complexity of the circuitry and minimizes the parasitics. The two cascode transistors are included for both improved performance and also for isolation of the gate voltage for programming. (d) Implementation of an nFET FG current mirror with indirect programming. This current mirror is a simple design, whereas the construction of an nFET programmable current mirror using the direct programming method is virtually impossible. (e) IPFG nFET current mirror data. The charge on the two FG nodes of (d) were normalized, causing the current gain to be nearly unity for a large range of current values. Data from a non-FG nFET current mirror are also included, and the improvements with the FG version is clearly evident. (f) Various current gains programmed in the IPFG nFET current mirror. Programming with IPFG transistors allow the current gain to be modified after fabrication. These current gains apply to all subthreshold currents. For above threshold current levels, the gains apply only for small signal deviations.

of gate lines also to V_{dd}. As a result, only a single device will meet both criteria for injection to occur. Fig. 2(d) shows that using this array programming procedure, a high-degree of programming accuracy can be achieved [10].

The efficiency of hot-electron injection is not constant over all current ranges, but the efficiency is highest for subthreshold currents [9]. While FG transistors and injection can be used with above-threshold current levels, the high injection efficiency of subthreshold currents coupled with the lower current levels, and hence reduced power consumption, is why subthreshold operation is typically the focus of FG circuit discussions. In consequence, the following discussion will primarily describe the operation of indirect programming with subthreshold current levels, and the concepts can also be extended to above-threshold current levels.

Additional techniques for using FG transistors that do not require programming have also been presented [11]–[13] that rely on techniques such as UV photoinjection for normalizing FG charges across an IC [14] and removing trapped charge at the time of fabrication [15]. However, using the unique trapped charge on each FG as a feature of these FG transistors, these FG transistors can be used as programmable current sources, transistors with tuneable threshold voltages, and perfectly matched transistor pairs (by programming out mismatches between devices). In short, FG programming provides versatility and flexibility to analog circuit design.

III. Motivation for Indirect Programming

To illustrate the usefulness of this indirect programming method, Fig. 3(a) shows the FG current mirror introduced in [16] for perfectly matching the two leg currents. The full schematic of this current mirror is actually given by Fig. 3(b), and the increase in complexity is clearly evident. The additional resistances and capacitances introduced by the eight T-gates, used to break the FG transistors out of the mirror for programming, seriously hamper the performance of the current mirror, especially at high frequencies. The simple two-transistor current mirror becomes a complex 18-transistor circuit.

The use of indirectly programmed FG (IPFG) transistors simplifies the pFET current mirror to that of Fig. 3(c). Now, only a minimal amount of disconnects need to be included. Only two

cascoding transistors and a single T-gate are used, and the cascoding transistors serve the dual purpose of isolating the FG transistor and enhancing the current response of the mirror.

Precise programming of nFETs with hot-electron injection is virtually impossible due to process-control techniques that specifically work to avoid nFET injection [17]. When an nFET is to be used as a precise current source with FGs, a pFET is programmed, and that current is mirrored into the nFET current source, as shown in Fig. 1(d). Therefore, creating a programmable nFET current mirror with the direct method of programming is no simple task.

The process of programming an nFET is more explicit with indirect programming. Since an nFET and pFET can share the same FG, the nFET current is set by programming the pFET. This technique allows the construction of a programmable nFET current mirror [Fig. 3(d)] that is completely analogous to the pFET version of Fig. 3(c).

Fig. 3(e) and (f) shows the benefits of using not only a FG programmable current mirror, but also an indirectly programmed version. The data from these plots were all obtained from an nFET version of the programmable current mirror. Data from a pFET version of an indirectly programmed current mirror has similar results, but only the nFET version, which was not previously capable of being built, is shown here for simplicity.

Fig. 3(e) shows that the result of normalizing the charge on the two floating nodes in the current mirror allows the current mirror to perform very close to the ideal. This nFET version of an IPFG current mirror was constructed using identically sized FG transistors. Therefore, normalizing the charge on the two floating nodes resulted in identical currents flowing through both legs of the IPFG current mirror. Since the subthreshold current flowing through an FG transistor in saturation is

$$I = I_0 e^{\kappa V_{\mathrm{FG}}/U_T} e^{-V_s/U_T} e^{V_d/V_A} \tag{1}$$

the current gain $I_{\mathrm{out}}/I_{\mathrm{in}}$ is

$$\frac{I_{\mathrm{out}}}{I_{\mathrm{in}}} = \frac{I_0 e^{\kappa V_{\mathrm{FG,out}}/U_T} e^{-V_s/U_T} e^{V_d/V_A}}{I_0 e^{\kappa V_{\mathrm{FG,in}}/U_T} e^{-V_s/U_T} e^{V_d/V_A}} = \frac{e^{\kappa V_{\mathrm{FG,out}}/U_T}}{e^{\kappa V_{\mathrm{FG,in}}/U_T}} \approx 1 \tag{2}$$

assuming that the drains are at similar potentials. In these subthreshold equations, U_T is the thermal voltage, κ is the capacitive ratio coupling from the gate to the surface potential, and V_A is the Early voltage [18].

Fig. 3(e) shows that the gain can, indeed, be made very close to unity by programming identical charges to the two floating nodes. Also included are data from a standard two-transistor current mirror showing that the percent error from the input to the output is 7–10% over a wide range of input currents. This degree of mismatch is not unexpected for small-sized transistors ($W/L = 2.4\ \mu\mathrm{m}/1.2\ \mu\mathrm{m}$) such as these [19].

In addition to normalizing the FG charge for a unity-gain current mirror, this IPFG current mirror allows the gain to be set after fabrication by programming different charges to the two floating nodes. Fig. 3(f) shows measurements of the current mirror programmed to a variety of gains. These gains were well within 1% accuracy. While (2) allows the IPFG current mirror to achieve unity gain over a wide range of current levels, this same relationship will only allow the current mirror to achieve the desired nonunity gains while both transistors stay in the subthreshold region. Once one transistor enters moderate or strong inversion, the exponential relationship of (1) no longer holds, and the gains will degrade from their programmed values. Therefore, the baseline current for measurement in Fig. 3 was a subthreshold current (1 nA).

This current mirror example shows several distinct advantages of indirect programming over previous methods which required disconnection circuitry. These advantages, and others that have not yet been mentioned, are summarized as follows. Indirect programming of FG transistors:

- allows nFET programming;
- decreases the number of poles/parasitic capacitances for faster operational speeds;
- decreases resistance;
- decreases minimum supply headroom;
- reduces transistor count/real estate;
- permits run-time time programming/calibration.

IV. Indirect Programming of pFET Transistors

The most basic method of indirect programming uses injection in the programming pFET to set the current in the agent pFET and tunneling for erasing that current. The programming pFET can be placed in a large FG array similar to that shown in Fig. 2(b) and selected and programmed in the fashion of [6]. The output of the agent will be a scaled version of the programmer, assuming the drain and source potentials of the two devices are similar. Scaling is due to W/L ratios and any mismatch between the two devices. Fig. 4(a) shows the *I–V* characteristics for a gate sweep of both the programmer and the agent, which are identically sized devices ($W/L = 2$). Typically, the agent current is unobservable, but these data are from an isolated pFET–pFET pair sharing the same FG that will be used for characterization purposes.

Assuming that the sources and drains of the two transistors are at similar potentials is not always valid. Fig. 4(b) shows the effects of varying the source potential of the agent. With both transistors in the subthreshold regime, varying the programmer current yields approximately a 1:1 change in the agent current. The exact relationship is a ratio of the subthreshold slope, κ/U_T, of the two transistors, which should be very closely matched due to their same orientation and close proximity in layout.

When programming the agent current to a desired value, only the programmer current is observable. Therefore, measurement of the programmer current is used to predict the current flowing through the agent. Using characterization curves such as the ones shown in Fig. 4(b) (which account for the subthreshold slopes and the differences in current due to differing source potentials), the agent current can be accurately programmed. Using these characterization curves to set the programmer current that will yield the desired agent current, we show in Fig. 4(c) that this technique can be used to accurately set the agent current within tolerance for two different values of the agent's source potential. While achieving high precision on

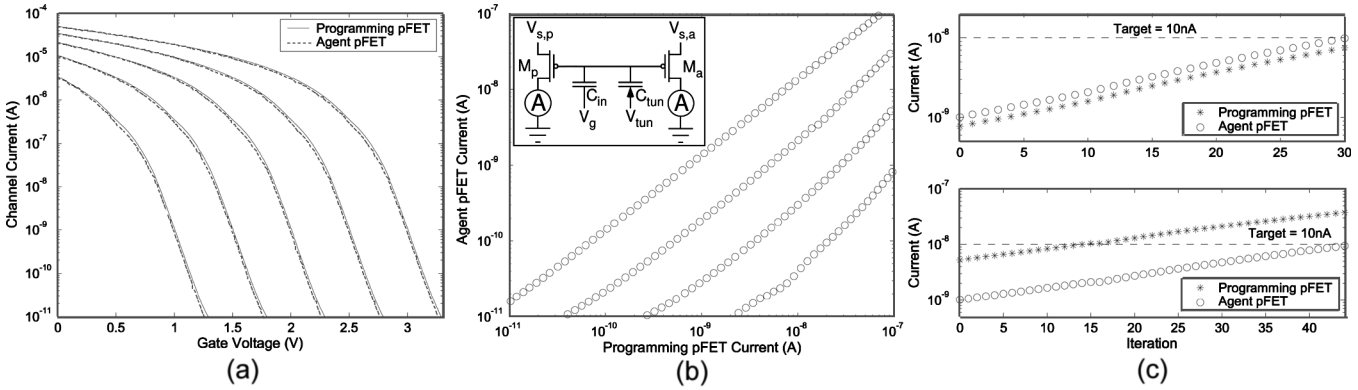

Fig. 4. (a) *I–V* characteristics of an indirectly programmed pFET ($W/L = 2$) and its programming pFET ($W/L = 2$). The currents were measured simultaneously through two identical picoammeters using the schematic shown in this figure. Typically, the current through the agent is unobservable, but these data are from an isolated pFET–pFET pair used for characterization. (b) Schematic for the measurements and the ratio of the programming pFET current to the agent pFET current for various values of $V_{s,a}$. The slope of each trace begins to differ from unity at low current levels due to measurement limitations. At high current levels, the slope differs from unity since the programming pFET leaves subthreshold sooner than the agent pFET as $V_{s,a}$ is increased. (c) Programming the agent pFET to a target. (Top) Programming when the sources are at similar potentials. (Bottom) Programming when the agent pFET's source has been lowered below the programmer pFET's source potential.

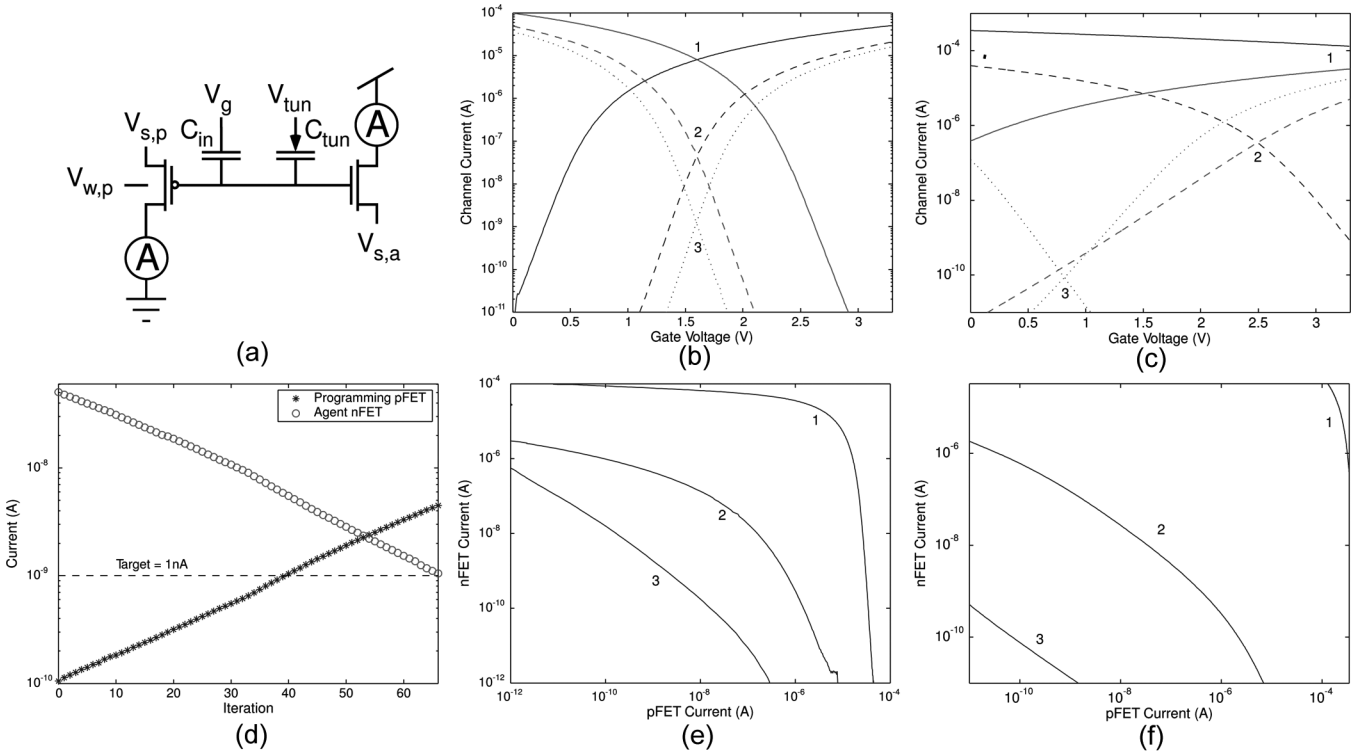

Fig. 5. Indirect programming of an nFET transistor. (a) Testing setup for the following measurements using two identical picoammeters. These data are from an isolated nFET–pFET pair that has been used for characterization purposes. (b) *I–V* characteristics of an nFET–pFET pair ($W/L = 2$ for both). Curve 1 shows the *I–V* relationships when $V_{w,p} = V_{s,p} = V_{dd}$ and $V_{s,a} = gnd$. Curves 2 and 3 show the *I–V* relationships attained by increasing $V_{s,a}$ above gnd and by lowering $V_{w,p} = V_{s,p}$ below V_{dd}. Changing the source potentials of the nFET and pFET allows the two transistors to operate with subthreshold currents simultaneously, which is adventageous for accurately programming the nFET transistor. (c) *I–V* characteristics attained by raising $V_{w,p}$ and lowering $V_{s,p}$. The pFET is much (10 times) larger than the nFET. This is an exagerated example that would not typically be used but has been used here to illustrate the ability to achieve subthreshold operation in both transistors even in an undesirable case. Curve 1 shows the *I–V* relationships when $V_{w,p} = V_{s,p} = V_{dd}$ and $V_{s,a} = gnd$. Both transistors have very large above threshold currents when they cross (if the gate voltage increased above V_{dd}) and require significant movements to both achieve subthreshold operation. Curves 2 and 3 show that by raising the pFET's well potential above V_{dd} and by lowing the pFET's source potential below V_{dd}, the operation of both transistors can be placed into the subthreshold regime. Even though the terminals of the nFET are not altered in this example, the current flowing through the nFET still changes. This change in current is due to capacitive coupling onto the FG node through the parasitic capacitances (C_{gs} and C_{gw}) of the pFET. This capacitive coupling, as well as the change in the subthreshold slope, will be explained in Section VI. (d) Programming the agent nFET to a target current within accuracy. Either the method of (b) or (c) can be used to place both transistors into subthreshold operation for programming. (e) Current-to-current relationships for each of the three curves shown in (b). (f) Current-to-current relationships for the method used in (c). By placing both transistors into subthreshold operation, a given percentage change the pFET's current translates into a linear percentage change in the nFET, therefore making the programming algorithm easier to implement and predict correct pFET current for the desired nFET current.

the actual programmed current in the agent is important, the ultimate goal of accurate programming is to achieve precise control over the operation of the overall circuit, and this procedure will be described in detail in Section VII.

V. Indirect Programming of nFET Transistors

As stated previously, an important advantage of indirect programming is that it provides a simple mechanism for programming nFETs, whereas low injection efficiency makes direct nFET programming difficult. In this section, a pFET and an nFET share a common FG, as was shown in Fig. 1(b). Fig. 5(b) shows the *I–V* characteristics of both the nFET and pFET. If the transistors are not properly sized, then the current level at which both transistors have equal currents will be very high, as is illustrated in Fig. 5(c). Unlike the pFET–pFET case, a direct relationship between the two transistors is not easily obtained. When the two transistor currents are not in subthreshold simultaneously, a current-to-current relationship like that in curve 1 of Fig. 5(b) is the result. Small changes in pFET current yield large changes in nFET current. Therefore, restricting the operation to strictly subthreshold is desirable because it linearizes the current-to-current ratio.

Two methods are available to ensure that both transistors are simultaneously in the subthreshold regime. The first method requires moving the sources of both transistors. Decreasing the pFET source (referenced to $V_{\rm well}$) and increasing the nFET source (referenced to $V_{\rm bulk}$) reduces the current in each transistor. This moves the threshold voltages to a point in which it is possible to operate both transistors in subthreshold at the same time [Fig. 5(a)]. Fig. 5(b) relates the pFET-to-nFET current for each set of curves in Fig. 5(a). Lowering the crossover point increases the linear range of the current-to-current ratio.

A linear current-to-current relationship makes predicting the agent current trivial. However, any reasonable current-to-current relationship [like curve in 2 Fig. 5(b)] allows accurate programming of the nFET.

As the source of the agent may not always be accessible or is set to a given potential due to placement within the circuit, the previous method is not always possible. The second method of ensuring that both transistors are in subthreshold requires that the programming pFET is in a well isolated from the operational circuit and that the well can be accessed. By raising the potential of the programmer's well and also lowering its source potential, the current flowing through the pFET is reduced. By using this procedure, the currents flowing through the nFET and pFET can be made to cross each other in the subthreshold regime. Fig. 5(c) shows the operation of this process for the worst case scenario in which the pFET is much larger than the nFET $(W_p/L_p = 10(W_n/L_n))$. Since the pFET is so much larger than the nFET, larger voltage differences from V_{dd} must be used in this example to bring the currents simultaneously into subthreshold operation. Typically, a nearly minimum-sized pFET programmer would be used, and the voltage differences would not be as large, but for illustrative purposes, we have shown that this operation is still possible under the worst-case scenario.

The movement of the nFET's current is due to capacitive coupling onto the FG, which will be explained in detail in the next section. Again, this movement is maximized in this example due to the large size of the pFET. Additionally, the change in the subthreshold slope, as seen in Fig. 5, is another result of the pFET's large size and would be minimized for smaller transistor sizes. This effect will also be discussed in the following section.

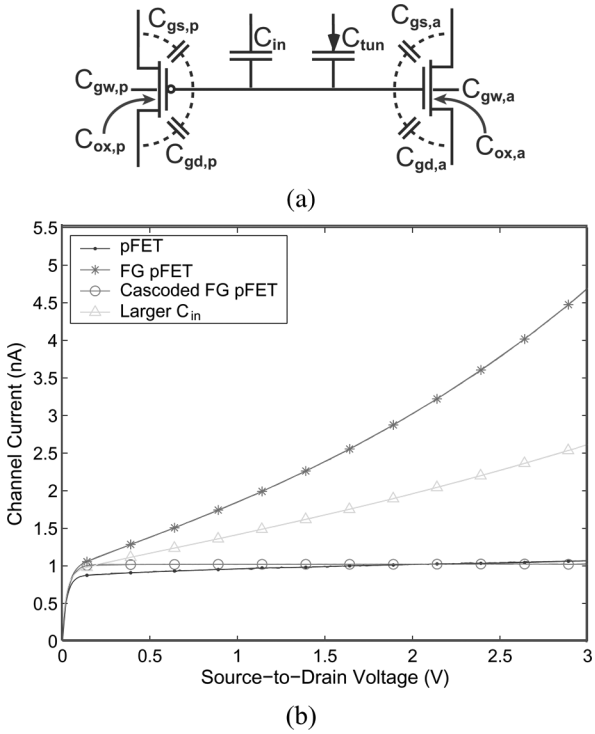

Fig. 6. (a) Schematic of a pair of transistors sharing the same FG and the parasitic capacitances that allow coupling of voltages onto the floating node. (b) Transistor drain sweeps. Due to capacitive coupling through $C_{gd,a}$, $I_{\rm sat}$ in the FG pFET increases exponentially for larger $V_{ds,a}$ values. Increasing $C_{\rm in}$ increases the effective Early voltage. Cascoding the agent transistor eliminates the exponential current increase and flattens $I_{\rm sat}$ more than the $I_{\rm sat}$ of the identically sized non-FG pFET.

Either of these two methods can be used to accurately program a current in the nFET. By keeping both transistors in subthreshold and measuring the pFET's current, the linear relationship of either Fig. 5(e) or Fig. 5(f) can be used to predict the nFET's current. Fig. 5(d) shows an example of accurately programming a current in the nFET where only the pFET's current is observable during the programming routine.

VI. Capacitive Coupling With Indirect Programming

As has been shown previously, the difference between source potentials of the programming pFET and the agent transistor need to be taken into account when programming so that the correct current flows through the agent. The drain potentials of the two transistors are also of concern, especially the drain of the agent since the operation of its connected circuit can affect the potential at the drain. The terminals of the programming pFET is held constant when not programming, thus eliminating all transient coupling effects from it.

The voltage on any FG node is set by a combination of the FG charge and a sum of the inputs to the gate through capacitive dividers [20]. The extension of the the FG voltage for the indirect programming case is depicted in Fig. 6(a) and described by

$$V_{\rm FG}=\frac{Q_{\rm FG}}{C_T}+\frac{C_{\rm in}}{C_T}V_g+\frac{C_{\rm tun}}{C_T}V_{\rm tun}+\frac{C_{gd,p}}{C_T}V_{d,p}+\frac{C_{gs,p}}{C_T}V_{s,p}+\frac{C_{gw,p}}{C_T}V_{w,p}+\frac{C_{ox,p}}{C_T}\psi_p+\frac{C_{gd,a}}{C_T}V_{d,a}+\frac{C_{gs,a}}{C_T}V_{s,a}+\frac{C_{gb,a}}{C_T}V_{b,a}+\frac{C_{ox,a}}{C_T}\psi_a \quad (3)$$

where C_T is the total capacitance connected to the FG node, the p and a subscripts indicate the programmer and the agent, and ψ represents the surface potential of each transistor (constant ψ in subthreshold). Since $C_{gd,a}$ is a small parasitic capacitance, the drain of the transistor acts as an input to the gate. As the drain voltage of the agent is swept, a subthreshold current through the device changes exponentially, as is shown in Fig. 6(b). This is a significant alteration from the small slope due to the Early voltage of an identically sized transistor, which is also shown.

In essence, this drain coupling of the agent can be viewed as reducing the effective Early voltage, which is undesirable if using the transistor as a current source. By rewriting (3) as

$$V_{\rm FG} = \frac{C_{\rm in}}{C_T} V_g + \frac{C_{gd,a}}{C_T} V_{d,a} + V_{\rm offset} \tag{4}$$

where $V_{\rm offset}$ represents all the other terms in (3), replacing the gate term in the subthreshold equation (1) with (4), and dropping the a subscript for the agent, the saturation current becomes

$$I = I_0 e^{\kappa\left(\frac{C_{\rm in}}{C_T}V_g + \frac{C_{gd}}{C_T}V_d + V_{\rm offset}\right)/U_T} e^{-V_s/U_T} e^{V_d/V_A}. \tag{5}$$

Rearranging, this expression takes the form

$$\begin{aligned} I &= I_0 e^{\kappa V_{\rm offset}/U_T} e^{\kappa \frac{C_{\rm in}}{C_T} V_g/U_T} e^{-V_s/U_T} e^{V_d\left(\frac{1}{V_A} + \frac{\kappa C_{gd}}{C_T U_T}\right)} \\ &= I_0 e^{\kappa V_{\rm offset}/U_T} e^{\kappa \frac{C_{\rm in}}{C_T} V_g/U_T} e^{-V_s/U_T} e^{V_d\left(V_A \| \frac{C_T U_T}{\kappa C_{gd}}\right)}. \end{aligned} \tag{6}$$

The effective Early voltage is thus

$$V_{A,\rm eff} = V_A \| \frac{C_T U_T}{\kappa C_{gd}}. \tag{7}$$

With typical capacitance values, the effective Early voltages for FG transistors can easily fall into the range of 1 V, much like the the FG transistors shown in Fig. 6(b).

If supply headroom issues are important, then the drain-coupling effect can be minimized by increasing the input gate capacitance. Increasing $C_{\rm in}$ increases C_T, thereby reducing the effects of coupling through the parasitic capacitances, such as $C_{gd,a}$. While the saturation current still has an exponential increase with drain potential, the effective Early voltage is increased, as is shown in Fig. 6(b).

If supply headroom issues are not a concern, then this drain-coupling effect can be completely removed by adding a cascode transistor at the drain of the agent. The saturation current received by the circuit is flatter than even a standard transistor, as is shown in Fig. 6(b).

Coupling through the gate-to-drain capacitances is not the only source of coupling into the floating node. In fact, all the terminals affect the drain currents of the two transistors to varying degrees by coupling into the floating node, as was shown in (3). These varying degrees depend on both the total capacitance, C_T, connected to the FG and also the size of the capacitor through which the voltage couples, which is typically a small parasitic capacitance. Increasing C_T decreases the capacitive coupling affects, as does decreasing the parasitic capacitances through which the coupling takes place. For example, simply increasing the drawn $C_{\rm in}$ and using a minimum sized transistor will reduce the effect of the overlap capacitance C_{gd} coupling into the FG.

For this reason, when programming a nFET–pFET pair and altering the pFET's source and well potentials, these voltages alter the charge on the floating node. This is the reason that the nFET's curve shifts in Fig. 5(c) because the pFET is a large device, and the parasitic capacitances between the gate and source and the gate and well are comparable to C_T. For this reason, nearly minimum sized programmer pFETs should be used when using an nFET–pFET pair to reduce the parasitic capacitances.

The second reason for making the programmer pFET small in an nFET–pFET pair is because of the change in the subthreshold slope when modifying the pFET's source and well potentials. The parasitic capacitances of a transistor are different depending on which mode of operation the transistor is in (subthreshold or above threshold). To minimize the changes in the coupling affects between modes of operation, the transistors should be made small so that the input capacitance, $C_{\rm in}$, dominates the total capacitance, C_T.

VII. Precise Tuning of Circuits

If the dc operating point of the agent transistor is not known, and the drain current has an exponential dependance upon all of its terminals, how can an indirectly programmed transistor accurately bias a circuit, especially in the case where no cascode is used to protect the drain terminal? Even though the current through the agent transistor is unobservable, the overall operation of the circuit can be tuned so precisely that the effects of device mismatches can be negated. In the following, we will give a circuit example that shows the method for programming both a pFET and an nFET for correct circuit operation.

A. Overview of the Capacitively Coupled Current Conveyor

The circuit we will use to demonstrate the indirect-programming algorithm is an indirectly programmed version of the capacitively coupled current conveyor (C^4), as shown in Fig. 7(a). The C^4, which is a bandpass filter typically used in audio applications, serves as a useful example of indirect programming because the two corner frequencies are each set solely by the current flowing through a single transistor, where one must be an nFET and the other a pFET. Further information on the C^4 can be found in [10], [21], [22] which cover direct FG and non-FG versions of the C^4. A brief summary elucidating the pertinent points of the C^4 with regards to the indirect-programming algorithm is as follows.

The C^4 is a capacitively based bandpass filter with electronically tunable corner frequencies that are independent of each other. The frequency response of the C^4 is given by

$$\frac{V_{\rm out}}{V_{\rm in}} = -\frac{C_1}{C_2} \frac{s\tau_l}{s^2\tau_h\tau_l + s\left(\tau_l + \frac{C_2 U_T}{\kappa I_{\tau_h}}\left(\frac{C_o}{\kappa C_2} - 1\right)\right) + 1} \tag{8}$$

where the time constants are

$$\tau_l = \frac{C_2 U_T}{\kappa I_{\tau_l}} \quad \tau_h = \frac{C_T C_O - C_2^2}{C_2} \frac{U_T}{\kappa I_{\tau_h}} \tag{9}$$

and a high-frequency zero, which occurs at sufficiently high-frequencies, has been neglected. The total capacitance, C_T, and the output capacitance, C_O, are defined as $C_T = C_1 + C_2 + C_W$ and $C_O = C_2 + C_L$. The currents I_{τ_h} and I_{τ_l} are the bias currents

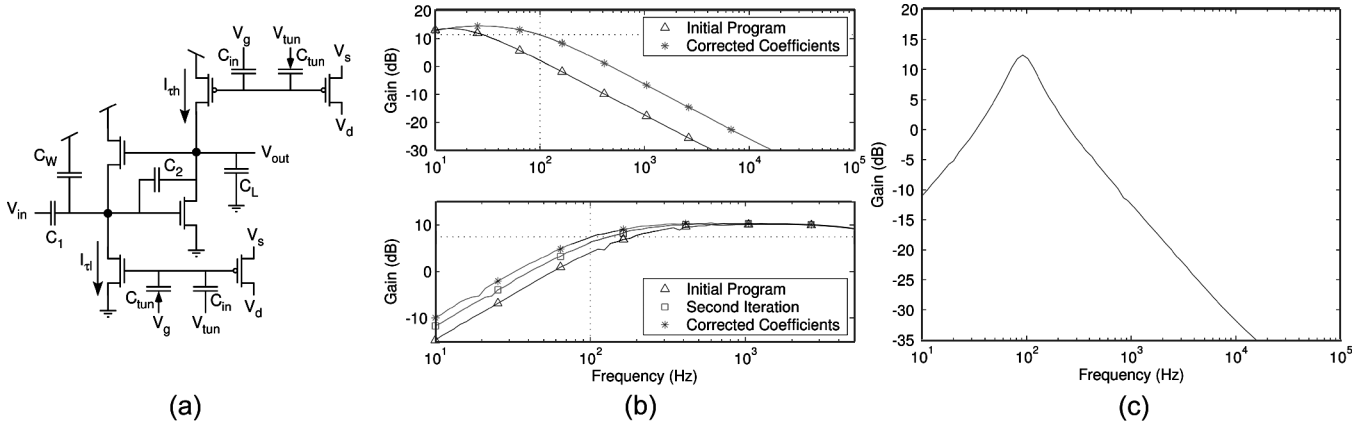

Fig. 7. (a) Schematic of an indirectly programmed version of the capacitively coupled current conveyor (C^4). This bandpass filter is used as an example of programming a circuit to a desired performance because the two corner frequencies are each tuned solely by altering the current through a given transistor. The high corner frequency is tuned by programming the current through a pFET ($I_{\tau h}$). The low corner frequency is tuned by programming the current through an nFET ($I_{\tau l}$). Thus, the C^4 is a good example for showing the operation of programming both a pFET and an nFET indirectly for a desired circuit performance. (b) Indirectly programming both corner frequencies of a C^4 to a target of 100 Hz. (Top) Programming the high corner frequency, or the current through a pFET, requires only two steps to achieve a desired value within tolerance. The crosshairs show the location (frequency and gain) of the desired -3-dB frequency. As can be seen, the actual -3-dB frequency matches the target well within the allotted tolerance. This programming technique essentially eliminates the effects of mismatch of device parameters. (Bottom) Programming the low corner frequency requires three steps. The additional step is used to determine an exact relationship between the programmer current and the agent current. (c) By programming both corner frequencies of the C^4, to the desired value, the circuit takes on the form of a bandpass filter.

through the pFET and nFET agents, respectively, as shown in Fig. 7(a). In consequence, the current flowing through a pFET alone controls the high corner frequency, and the current flowing through an nFET alone controls the low corner frequency. As a result, the C^4 is a good circuit example for showing the proper programming procedure since the filter's operation is directly affected by both a pFET and an nFET.

Programming a C^4 to have precisely tuned time constants requires finding an estimate of the effective mismatch of the devices involved in each time constant. This method of estimation was presented for a directly programmed C^4 in [23], and the modifications to the procedure for indirect programming are presented here.

B. pFET Programming

Programming an agent pFET to yield a desired circuit performance is a straightforward procedure. This process involves two steps in which a current is programmed into the pFET, and the effects of mismatch are then calibrated out.

The programmer is initially programmed to the current that should yield correct circuit performance if all devices were ideal. Using the designed values and a rubric for the correct circuit operation, an initial current is programmed into the programmer. However, all device parameters will deviate from the ideal, and since the dc operating point of the agent will likely differ from the programmer, the actual performance of the circuit will not equal the idealized performance. Nevertheless, once the programmed current and the resulting circuit performance are known, the function relating the two can be calculated. This function incorporates both the deviations from the ideal device parameters and also the difference in dc operating points of the programmer and agent, and, thus, the circuit can be reprogrammed to any desired performance.

Using the example of the C^4, the pFET agent exclusively controls the high corner frequency. The rubric for knowing correct circuit operation is thus the placement of the high corner frequency, which is given by

$$f_h = \frac{1}{2\pi}\frac{C_2}{C_T C_O - C_2^2}\frac{\kappa_{p,\mathrm{eff}} I_{\tau_h}}{U_T} \tag{10}$$

where $\kappa_{p,\mathrm{eff}}$ is the effective coupling onto the surface potential including the input capacitor, C_{in}. An initial current is programmed into the the programmer assuming ideal values for the capacitors and $\kappa_{p,\mathrm{eff}}$ such that the resulting corner frequency should be the target value. Since these idealized values are not the actual values, and since the drain of the agent is not the same as that of the programmer, the actual programmed corner frequency does not fall within tolerance of the target value. However, the function relating the corner frequency and the current only involves a single coefficient since the currents are remaining in subthreshold and (10) applies.

Equating all the coefficients of the programmed current into a single coefficient, (10) becomes

$$f_h = K_{\mathrm{high}} I_{\tau_h}. \tag{11}$$

Since the programmed current and the circuit output, the corner frequency, are known, the true value for K_{high} can be calculated. Now,

$$K_{\mathrm{high}} = \frac{1}{2\pi}\frac{C_2}{C_T C_O - C_2^2}\frac{\kappa_{p,\mathrm{eff}}}{U_T} k_{\mathrm{dc}} \tag{12}$$

where all the device parameters represent their actual values, and k_{dc} represents the shift in the bias current between the agent and the programmer due to differences in the dc operating point. Using (11) with the measured value of K_{high} allows a second programming step to be used to produce the desired corner frequency.

Fig. 7(b) shows an example programming the C^4's high corner frequency using this method, and the high degree of

accuracy with this approach is clear. The crosshairs indicate the location of the ideal −3-dB frequency. Further improvements in accuracy can be achieved by improving the accuracy of the FG programming algorithm, as is described in [6].

C. nFET Programming

Since the current through an nFET agent follows an inverse relationship to the current through its pFET programmer, programming a precise current is a more complicated procedure than a pFET–pFET case. A high degree of characterization of the nFET–pFET combination will ease the programming procedure. However, this characterization is not required, and through the following example, we will show how to achieve accuracy even when an exact relationship between the nFET and pFET is not initially known.

The procedure starts by programming an initial current into the programmer that will translate as closely as possible to an nFET agent current that will yield the desired circuit operation. The translation from programmer current to agent current can be estimated by a characterization nFET–pFET pair on the periphery of the die area or even by simulation. A circuit measurement is taken to determine the deviation from the ideal performance. This difference will be due to deviations in parameter sizes and values as well as differences in the agent current from the expected value.

Whereas simply finding the estimate of the device and current mismatch for a given parameter was sufficient for the pFET–pFET case, this method is no longer sufficient for the nFET–pFET case. Placing both the programmer and agent transistors into subthreshold simultaneously greatly eases the programming procedure since the relationship is linearized (on a logarithmic scale), as is shown in Fig. 5(e) and (f). Two calibration steps are required to accurately program an nFET. The first calibration step allows the effective mismatch to be found, and the second step allows the slope of the relationship between the programmer and the agent to be determined. Then the current can be programmed so that the circuit accurately performs the desired action.

Again, we will use the C^4 as a circuit example, and since the low corner frequency of the C^4 requires only the current through the nFET agent to be modified, the low corner frequency will be the system parameter under study. The low corner frequency is given by

$$f_l = \frac{1}{2\pi}\frac{\kappa_{n,\text{eff}} I_{\tau l}}{C_2 U_T}. \tag{13}$$

Using an estimate for the nFET agent's current and the ideal values for the device parameters, a current is programmed such that the low corner frequency should hit its target. However, the actual corner frequency will likely deviate from the desired value due to both device mismatch and the difference from the desired nFET current. The actual corner frequency will have a value of

$$f_1 = \frac{1}{2\pi}\frac{\kappa_{n,\text{eff}} I_{n1}}{C_2 U_T} = K_{\text{low}} I_{n1} \tag{14}$$

where f_1 is the initial measured corner frequency, K_{low} is the estimated multiplicative coefficient, and I_{n1} is the unknown and unobservable agent current.

In addition to the unknown agent current, the relationship between the programmer and agent currents is also not yet known. An alternative way of viewing this problem is that the slope of the curve in Fig. 5(f) is not known, even when assuming subthreshold operation. A second current must be programmed into the programmer using (14) such that

$$f_2 = K_{\text{low}} I_{n2} \tag{15}$$

where I_{n2} is the agent current and f_2 is the resulting corner frequency. However, this new corner frequency will likely not fall within tolerance because the exact value of I_{n2} is unobservable.

Nevertheless, there is now enough information to program the circuit accurately on a third iteration, and this is done by finding the slope of Fig. 5(f), assuming subthreshold operation. To find this slope, (14) is divided through by (15).

$$\frac{f_1}{f_2} = \frac{K_{\text{low}} I_{n1}}{K_{\text{low}} I_{n2}} = \frac{I_{n1}}{I_{n2}}. \tag{16}$$

Then, letting m represent the slope of Fig. 5(f) and using the ratios of (16), the slope is given by

$$m = \frac{\ln(I_{n2}) - \ln(I_{n1})}{\ln(I_{p2}) - \ln(I_{p1})} = \frac{\ln\left(\frac{I_{n2}}{I_{n1}}\right)}{\ln\left(\frac{I_{p2}}{I_{p1}}\right)} = \frac{\ln\left(\frac{f_2}{f_1}\right)}{\ln\left(\frac{I_{p2}}{I_{p1}}\right)} \tag{17}$$

Now, letting k represent the programming iteration number, the slope can be written as

$$m = \frac{\ln\left(\frac{f_{k+1}}{f_k}\right)}{\ln\left(\frac{I_{p,k+1}}{I_{p,k}}\right)} \tag{18}$$

where knowlege of only the programmer currents and resulting corner frequencies are required. Rewriting (18) and letting f_{k+1} represent the desired corner frequency, the exact current that must be programmed into the programmer is given by

$$I_{p,k+1} = I_{p,k}\left(\frac{f_{k+1}}{f_k}\right)^{1/m}. \tag{19}$$

Thus, in three steps, the relationship between the nFET and pFET has been determined, the effects of mismatch have been calibrated out, and the circuit has been programmed to the desired corner frequency.

Fig. 7(b) shows data from this programming procedure for the C^4's low corner frequency. On the third iteration, the corner frequency fell well within the tolerance of the programming algorithm, as is indicated with the ideal −3-dB point depicted with the crosshairs. Again, this percentage error could be improved even further by increasing the accuracy of the programming algorithm.

D. Generalized Indirect Programming Algorithm

While the C^4 served as a good example of indirectly programming a circuit for precise operation criteria, the C^4 is by no means an exclusive case. In fact, this indirect programming

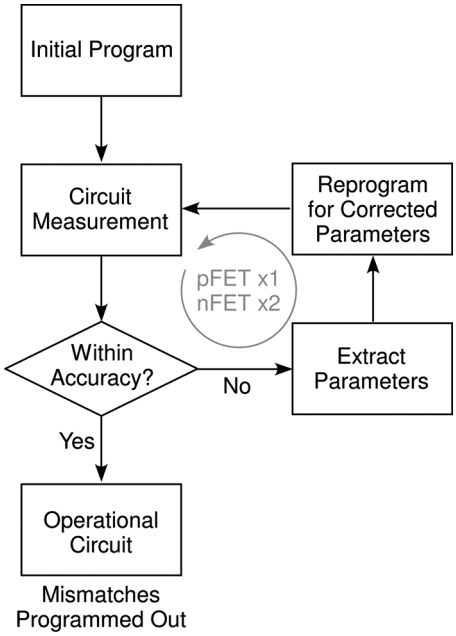

Fig. 8. Flow diagram of the programming algorithm used in tuning a circuit to the desired performance. Programming a pFET indirectly requires a single iteration through the loop, whereas programming an nFET indirectly requires two iterations through the loop. The additional iteration for the nFET results from the need to determine an exact relationship between the programmer pFET current and the agent nFET current.

algorithm can be applied to a wide variety of circuits, and it can be viewed in its generalized form to be as that described in the flow diagram of Fig. 8.

In all cases, the circuit should be initially programmed so that it would perform perfectly if all device parameters were ideal. Some circuit measurement should then be taken, be it a frequency response, a step response, etc., to determine how far from ideal the circuit's performance was. If at any time, this circuit operates within system tolerance, then no more steps are needed. However, the initial program will not likely produce the desired results, but it can be used to extract certain parameters about the circuit's operation. These parameters may include a variety of contributions, including capacitor sizes and transistor currents. These extracted parameters can then be used to reprogram the circuit, and the circuit is then tested again to find whether or not it operates within the desired tolerances.

This loop, as shown in Fig. 8, can be repeated as many times as necessary. Typically, only a single time through the loop is required for programming a pFET since both the programmer and the agent follow the same current trends. A second iteration through the loop is required for indirectly programming nFETs since not only device parameters must be determined but also the exact relationship between the currents in the nFET agent and the pFET programmer. Keeping both the agent and the programmer in subthreshold simultaneously aids this procedure since the relationship is linearized, as is shown in Fig. 5(e) and (f).

VIII. Benefits of Indirect Programming

In addition to the ability to program out mismatches in a circuit and set precise current sources, which are both advantages available with FG circuits and direct programming methods, the noninvasive nature of indirect programming has several benefits over traditional FG programming methods. These benefits are largely related to the removal of the transmission gates that are needed for disconnecting FG transistors for a programming phase.

The addition of at least one T-gate for every FG transistor, and often more T-gates for certain circuit configurations [16], adds both resistance and capacitance to the FG circuit. The added resistance and capacitance can have several harmful effects. These extra parasitics will slow down the operation of the circuit and, thus, limit the speed at which the circuit can operate. Also, when using large currents in the FG transistors, the added resistance, which is approximately 10 kΩ for small devices [24], will cause a significant voltage drop to form across the switch. The voltage drop could cause problems with the operation of the circuit and it could be large enough to alter the required voltage headroom of the circuit. Thus, the circuit would have to run on a larger supply voltage.

Since indirect programming of FG transistors does not require this disconnection via T-gate switches, many of the parasitics are removed. Therefore, IPFG circuits have the ability to operate at higher frequencies than do directly programmed FG circuits. The increase in speed with IPFG transistors was demonstrated with an IPFG inverter using an ad hoc programming method [25]. Moreover, this IPFG inverter was able to operate at faster speeds than an identically sized non-FG inverter. Furthermore, the removal of the selection switches removes the added resistance. Circuit applications requiring very low supply voltages can now utilize the programmability of FG transistors without concerns of headroom loss due to parasitic resistances.

IX. Run-Time Programming

Since the use of indirect programming does not require disconnection of the agent transistor from its circuit, there is now no need for a separate programming phase to set the charge on the FG nodes. In fact, programming can occur during normal operation of the circuit so that data acquisition does not need to be stopped in order to reprogram the device. This new "run-time programming," which will be introduced in this section, allows a circuit to be recalibrated while it is still operating so that the circuit can respond to changes in its environment (e.g., temperature) or new desires of the circuit's user (e.g., changing the gain of a certain band of frequencies in a hearing aid). This run-time programming, unlike adaptation techniques, allows programming to be turned on temporarily whenever recalibration is desired.

While typical methods of programming FG transistors [8] would work even in this run-time programming, these methods are not ideal since they involve large, instantaneous movements of the transistor's terminal voltages in order to cause injection to occur. Since, with indirect programming, the programmer and the agent share the same FG node and the movements on the programmer's terminals capacitively couple onto the FG node, these methods of programming can cause large instantaneous changes in the agent's current that could seriously alter the operation of the circuit. Therefore, when recalibrating a circuit while it is still operating, care must be taken so that the operation of

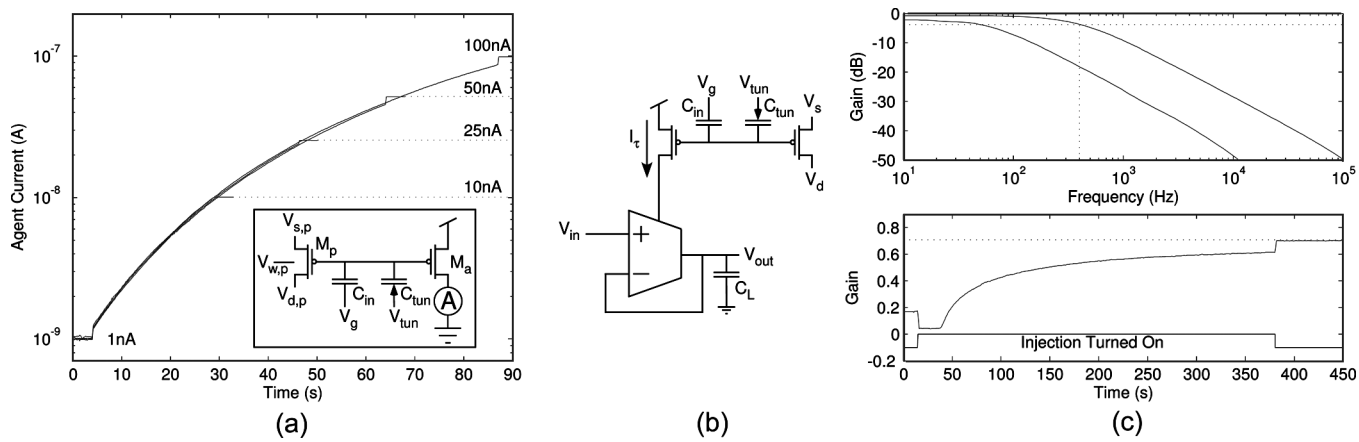

Fig. 9. Run-time programming using IPFG transistors. (a) Programming the agent current using run-time programming. At $t = 5$ s, the injection was turned on by symmetrically changing the source, well, and drain potentials of the programmer such that the contribution of all coupling terms negated each other and the FG voltage remained stationary. Once injection started, electrons were added to the FG, and the agent current started to increase. Injection was turned off (all the programmer terminals were symmetrically brought back to their initial position) when the agent current reached its target value. The curvature to the slope shows that the injection efficiency decreases as the currents near threshold operation. Programming speeds can be increased to the microsecond timescale by increasing the source-to-drain potentials. (b) Simple G_m–C first-order low-pass filter using an indirectly programmed tail current. (c) The filter initially had a corner frequency below 10 Hz and was to be reprogrammed to 400 Hz without stopping the operation of the filter. By looking at the output of the filter for an input of a 400-Hz sinusoidal waveform, injection was turned on and then turned off again when the amplitude of the filter reached the desired value. (Top) The frequency responses at the beginning and end of the run-time programming. The crosshairs show that the -3-dB point is at the target frequency. (Bottom) The output of the filter while the filter was actively being programmed. The small change in amplitude at the onset and termination of injection was due to slightly unsymmetric coupling onto the FG node. A large current was required for these frequencies due to the size of the load capacitance. The slightly unsymmetric coupling was used because, at the large currents required, the injection efficiency was very low, and the unsymmetric coupling allowed a more efficient current level to be used.

the circuit will not be temporarily rendered useless (and thus negating the benefits of using run-time programming).

To recalibrate an FG agent transistor in run-time operation using injection, the actual charge on the floating node should remain unaltered by any process except for injection. Therefore, any voltages that couple onto the floating node should always be balanced by an equal voltage coupling onto the floating node in the opposite direction. Referring back to (3) and Fig. 6(a), if one terminal of the programmer is moved, then another terminal must also be moved in the opposite direction such that the two voltages couple identical, but opposite amounts. The current flowing through the agent will thus not be moved at all. By "symmetrically" raising the programmer's source potential and lowering the drain potential with respect to $V_{\rm FG}$, the source-to-drain potential is increased until it reaches the potential at which the desired injection occurs. When injection occurs, the charge on the floating node is altered, and the current flowing through the agent is modified (increased for a pFET and decreased for an nFET). When the current flowing through the agent has reached the desired value, then the injection can be turned off by symmetrically returning the source and drain potentials to their normal operating values. As a result, this process of symmetrically modifying the programmer's terminal potentials allows the entire circuit to go back and forth between normal operating potentials and the larger injection potentials without an appreciable effect on the circuit's output.

Fig. 9(a) shows the operation of injecting the current to the desired value with this process. The small discontinuities in the current levels at the onset and termination of injection are a result of parasitic-capacitance estimates not being perfectly calibrated. Additionally, the larger jump at the termination of injection is a result of the higher current levels (near or above threshold) and the resulting changes in capacitor values since the parasitic capacitances have different values when the transistor is operating in either subthreshold or above threshold. These discontinuities can be accounted for, and, thus, injection can be turned off in anticipation that the final current will be the desired value. These discontinuities can also be calibrated out and compensated in a manner similar to [26].

To test the operation of run-time programming within a circuit, the circuit of Fig. 9(b) was built to show that by viewing the output of the circuit, the operation of the circuit can be recalibrated by using run-time programming. This circuit is simply a G_m–C element constructed to act as a first-order low-pass filter in which the time constant is set by an indirectly programmed transistor. The G_m element is simply a five-transistor operational transconductance amplifier (OTA) [18] in which the bias current is set with an IPFG transistor.

In this simple experiment to show how run-time programming can be used, the corner frequency of the filter was programmed to below 10 Hz. However, it was desired that this corner frequency should be moved to exactly 400 Hz without stopping the operation of the circuit. As a result, the output of the filter was viewed when injection was turned on in the programmer pFET. Injection was then turned off when the circuit was observed to be operating at the desired corner frequency. Fig. 9 shows frequency responses before and after the run-time programming as well as the observed output of the circuit while injection was occurring. The final output of the filter had the desired corner frequency.

This run-time approach to programming FG transistors has promising new possibilities for circuits needing frequent updates due to environmental changes and consumer needs. Additionally, a circuit using a similar approach has been used for adaptive applications by continuously updating the stored charge on the FG node [27].

X. Conclusion

Direct programming of FGs has been proven to be a highly accurate tool for analog designers. Difficulties with programming nFETs and the parasitics associated with the isolation circuitry exist with the direct programming method but can be overcome by the indirect programming method we have just introduced.

An early, ad hoc method of indirectly programming FG transistors has been shown to be a useful means of tuning a circuit [28]. Additionally, in this paper we have presented a systemmatic approach to programming both pFETs and nFETs indirectly. This systemmatic approach can easily be extended to large arrays of FG devices so that a large number of current sources can be programmed without invasively disconnecting them. In fact, directly and IPFG transistors can coexist in the same large array so that each might be used for its particular advantage.

Indirect programming also allows certain circuits to be transformed into a programmable version that would not have been previously possible. The aforementioned programmable nFET current mirror is now possible, and a neuron circuit [29] that cannot properly operate due to the parasitics of the isolation circuitry can now be made.

New possibilities with FG programming also exist. Since the agent transistor is never removed from its circuit, indirect programming removes the necessity of a separate programming phase and a operational phase. This allows the possibility of run-time recalibration and adaption to be carried out by the programming pFET.

Indirect programming offers solutions to many of the problems of direct programming while also providing new and unique capabilities to augment the analog designer's toolbox.

References

[1] T. Shibata and T. Ohmi, "A functional MOS transistor featuring gate-level weighted sum and threshold operations," *IEEE Trans. Electron Dev.*, vol. 39, no. 6, pp. 1444–1455, Jun. 1992.

[2] P. Hasler, C. Diorio, B. Minch, and C. Mead, "Single transistor learning synapses," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. Touretzky, and T. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 817–824.

[3] W. P. Millard, Z. Kalayjian, and A. G. Andreou, "Calibration and matching of floating gate devices," in *Proc. IEEE Int. Symp. Circuits Syst.*, Geneva, Switzerland, Jun. 2000, vol. IV, pp. 121–124.

[4] A. J. Montalvo and J. J. Paulos, "Improved floating-gate devices using standard CMOS technology," *IEEE Electron Dev. Lett.*, vol. 14, no. 8, pp. 372–374, 1993.

[5] J. Killens, "Utilizing standard CMOS process floating gate devices for analog design," Master's thesis, Dept. Elect. Comp. Eng., Mississippi State University, Mississippi, 2001.

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

[7] D. Graham, E. Farquhar, B. Degnan, C. Gordon, and P. Hasler, "Indirect programming of floating-gate transistors," in *Proc. IEEE Int. Symp. Circuits Syst.*, Kobe, Japan, May 2005, vol. 3, pp. 2172–2175.

[8] G. Serrano, P. D. Smith, H. J. Lo, R. Chawla, T. S. Hall, C. M. Twigg, and P. Hasler, "Automatic rapid programming of large arrays of floating-gate elements," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 2004, vol. 1, pp. 1373–1376.

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

[10] R. Chawla, D. Graham, P. Smith, and P. Hasler, "A low-power, programmable bandpass filter section for higher-order filter-bank applications," in *Proc. IEEE Int. Symp. Circuits Syst.*, Kobe, Japan, May 2005, vol. 3, pp. 1980–1983.

[11] F. Munoz, A. Torralba, R. G. Carvajal, J. Tombs, and J. Ramirez-Angulo, "Floating-gate-based tunable CMOS low-voltage linear transconductor and its application to HF G_m–C filter design," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 106–110, Jan. 2001.

[12] B. Minch, "Construction and transformation of multiple-input translinear element networks," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 12, pp. 1530–1537, Dec. 2003.

[13] B. Ahuja, H. Vu, C. Laber, and W. Owen, "A very high precision 500-nA CMOS floating-gate analog voltage reference," *IEEE J. Solid-State Circuits*, vol. 40, no. 12, pp. 2364–2372, Dec. 2005.

[14] Y. Berg, T. S. Lande, and Ø. Næss, "Programming floating-gate circuits with UV-activated conductances," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 12–19, Jan. 2001.

[15] E. Rodriguez-Villegas and H. Barnes, "Solution to trapped charge in FGMOS transistors," *Electron. Lett.*, vol. 39, no. 19, pp. 1416–1417, Sep. 2003.

[16] F. Adil, G. Serrano, and P. Hasler, "Offset removal using floating-gate circuits for mixed-signal systems," in *Proc. IEEE Southwest Symp. Mixed-Signal Design*, Las Vegas, NV, Feb. 2003, pp. 190–195.

[17] P. Hasler, A. G. Andreou, C. Diorio, B. A. Minch, and C. A. Mead, "Impact ionization and hot-electron injection derived consistently from boltzmann transport," *VLSI Design*, vol. 8, no. 1–4, pp. 455–461, 1998.

[18] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[19] P. Kinget, "Device mismatch and tradeoffs in the design of analog circuits," *IEEE J. Solid-State Circuits*, vol. 40, no. 6, pp. 1212–1224, Jun. 2004.

[20] K. Yang and A. G. Andreou, "Subthreshold analysis of floating-gate MOSFET's," in *Proc. 10th Biennial University/Government/Industry Microelectronics Symposium*, Research Triangle Park, NC, May 1993, pp. 141–144.

[21] P. Hasler, M. Kucic, and B. A. Minch, "A transistor-only circuit model of the autozeroing floating-gate amplifier," in *Proc. IEEE Midwest Symp. Circuits Syst.*, Las Cruces, NM, Aug. 1999, pp. 157–160.

[22] P. Smith, D. Graham, R. Chawla, and P. Hasler, "A five-transistor bandpass filter element," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 2004, vol. 1, pp. I-97–I-100.

[23] D. W. Graham, P. D. Smith, R. Chawla, and P. Hasler, "A programmable bandpass array using floating-gate transistors," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 2004, vol. 1, pp. I-97–I-100.

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

[25] B. Degnan, R. Wunderlich, and P. Hasler, "Programmable floating-gate techniques for CMOS inverters," in *Proc. IEEE Int. Symp. Circuits Syst.*, Kobe, Japan, May 2005, vol. 3, pp. 2441–2444.

[26] R. Harrison, J. Bragg, P. Hasler, B. Minch, and S. Deweerth, "A CMOS programmable analog memory cell array using floating-gate circuits," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 1, pp. 4–11, Jan. 2001.

[27] P. Hasler and J. Dugger, "An analog floating-gate node for supervised learning," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 5, pp. 834–845, May 2005.

[28] A. Low and P. Hasler, "Basics of floating-gate low-dropout voltage regulators," in *Proc. IEEE Midwest Symp. Circuits Syst.*, 2000, vol. 3, pp. 1048–1051.

[29] E. Farquhar and P. Hasler, "A bio-physcially inspired silicon neuron," in *Proc. IEEE Int. Symp. Circuits Syst.*, Vancouver, BC, Canada, May 2003, vol. 1, pp. I-309–I-312.

David W. Graham (S'00–M'07) received the B.A. degree in natural science from Covenant College, Lookout Mountain, TN, in 2001, and the B.S. degree in electrical engineering, and the M.S. and Ph.D. degrees in electrical and computer engineering all from Georgia Institute of Technology (Georgia Tech), Atlanta, in 2001, 2003, and 2006, respectively.

He is an Assistant Professor in the Lane Department of Computer Science and Electrical Engineering, West Virginia University, Morgantown. His research interests are in developing biologically inspired electronics, cooperative analog and digital signal-processing systems, and programmable analog devices.

Ethan Farquhar (S'99–M'06) received the B.A. degree in natural science from Covenant College, Lookout Mountain, TN, in 1999, the B.S. degree in computer engineering from Clemson University, Clemson, SC, and the M.S. and Ph.D. degrees in electrical and computer engineering from Georgia Institute of Technology (Georgia Tech), Atlanta, in 2003, and 2005, respectively.

He is an Assistant Professor in the Department of Electrical and Computer Engineering, University of Tennessee, Knoxville. His current research interests include modeling complex neuron structures with analog VLSI technology.

Brian Degnan (S'05) received the B.S. degree in mechanical engineering and the M.S. degree in electrical engineering from Rose-Hulman Institute of Technology. He is working toward the Ph.D. degree at Georgia Institute of Technology, Atlanta.

He also holds a research certificate from Kanazawa Institute of Technology, Kanazawa, Japan. His current areas of research are floating-gate inspired digital circuits.

Christal Gordon (S'00–M'07) received the dual B.S. degree in electrical and computer engineering from Polytechnic University, Brooklyn, NY, in 1999 and the M.S. degree in electrical engineering from the Georgia Institute of Technology, Atlanta, in 2000.

She is a faculty member in the School of Electrical and Computer Engineering, North Carolina State University, Raleigh. Her interests revolve around creating bio-inspired systems for use by the general public, engineers, and neuroscientists. Applications of these bio-inspired systems include efficient consumer electronics, neural processors, and cochlear implants.

Paul Hasler (S'87–M'01–SM'03) received the B.S. and M.S. degrees in electrical engineering from Arizona State University, Tempe, both in 1991, and the Ph.D. degree in computation and neural systems from California Institute of Technology, Pasadena, in 1997.

He is an Associate Professor in the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta. His current research interests include low-power electronics, mixed-signal system integrated circuits, floating-gate MOS transistors, adaptive information-processing systems, "smart" interfaces for sensors, cooperative analog–digital signal processing, device physics related to submicron devices or floating-gate devices, and analog VLSI models of on-chip learning and sensory processing in neurobiology.

Dr. Hasler received the National Science Foundation CAREER Award in 2001, and the Office of Naval Research YIP award in 2002. He received the Paul Raphorst Best Paper Award, IEEE Electron Devices Society, 1997, Best student paper award, CICC, 2005, Best student paper award, IEEE Ultrasound Symposium, 2006, Best Sensors Paper, ISCAS 2005, and a Best Paper Award at SCI 2001.