Run-Time Programming of Analog Circuits Using Floating-Gate Transistors

David W. Graham
Lane Department of Computer Science
and Electrical Engineering
West Virginia University
Morgantown, WV 26506–6109
Email: david.graham@mail.wvu.edu

Paul Hasler
School of Electrical and
Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia 30332–0250
Email: phasler@ece.gatech.edu

***Abstract*— The ability to recalibrate a system is an important feature in allowing that system to maintain optimal performance even in the face of new demands placed on that system by environmental changes or even consumers' desires. The use of floating-gate (FG) transistors provides programmability to analog circuitry and, hence, the ability to recalibrate an analog system. If the FG transistors are programmed indirectly by using a second transistor to perform hot-electron injection, then an analog system can be recalibrated and reprogrammed without ever having to take the circuit out of operation. In this paper, we present a technique for adjusting circuit properties while still in operation as well as example circuits in which this run-time programming is conducted.**

I. Introduction

Designing electronics that are robust enough to withstand varying environmental conditions is a difficult task. As a result, recalibration circuitry is often employed to adjust the system until it performs within a desired tolerance. Additionally, electronics users often desire a change in performance, and creating a design that can account for user-desired settings can be an equally burdensome design challengs. In consequence, electronics with static properties are often less desirable than electronics that offer programmability because programmability translates into versatility.

This desire for programmability and versatility has led the digital domain to the forefront of electronics design because of the ease of design associated with microprocessors, DSPs, and FPGAs. However, the addition of FG transistors to analog circuitry has yielded a significant amount of programmability, and even reconfigurability, to the analog domain [1], [2] where systems are able to operate at extremely low power consumptions.

Even if programmability is a feasible feature of present-day analog systems, many scenarios exist in which recalibration must be conducted without a lengthy recalibration phase in which the system is rendered unusable. Surveillance systems being recalibrated means that an area will be left unprotected temporarily, hearing aids being recalibrated could cause the user to not be able to hear, etc. In certain cases, this temporary unusability of a system is clearly not acceptable.

While the use of FG transistors provides for a means to quickly program and calibrate a system, most FG programming methods require a separate programming phase in which FG transistors are disconnected from the rest of the circuit. However, the use of the indirect method of programming the FG transistors, as was originally presented in [4] requires neither disconnection of the FG transistors nor a separate programming phase. In effect, this method allows programming, or recalibration, to be conducted while the circuit is still operating, as is depicted in Fig. 1. The remainder of this paper elucidates how this run-time recalibration can be achieved using indirectly programmed FG transistors.

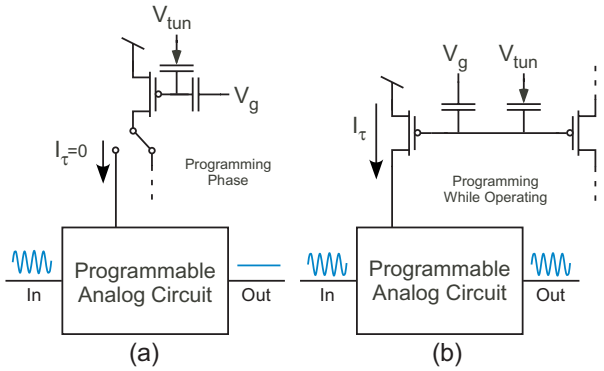

Fig. 1. (a) The direct method of programming floating-gate (FG) transistors, as described in [3], requires a disconnection of each FG transistor for a programming phase. This disconnection causes the circuit to temporarily stop its operation. (b) The indirect method of programming FG transistors does not require a disconnection, as described in [4]. As a result, the indirect method of programming FG transistors allows the circuit to continue operation while it is being reprogrammed or recalibrated. The method of this run-time programming is described in this paper.

All data presented in this paper are from 0.5μm processes available through MOSIS.

II. Indirect Programming of Floating-Gate Transistors

While various methods of normalizing the charge on a FG transistor provide a means to reduce the effects of mismatch between devices (including UV photoinjection and charge normalization at fabrication time) [5], [6], [7], programming of FG transistors allows a circuit to both eliminate the effects of mismatch as well as set a desired current or voltage precisely

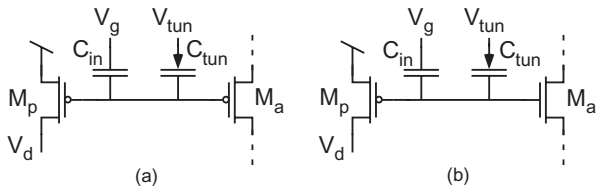

Fig. 2. Schematic of an indirectly programmed (a) pFET and (b) nFET. The programmer transistor, M_p, is connected to the external programming structure and is actively programmed via hot-electron injection. The agent transistor, M_a, is connected to its circuit (represented by the dotted lines) and is passively programmed.

within that circuit. Using FG programming methods such as [3], a current or voltage can be set accurately and then reprogrammed to a new, accurate value repeatedly. However, the method described in [3] requires that FG transistors be isolated from the rest of the circuit for a specified programming phase because accurate programming is carried out by using hot-electron injection. The FG transistor can then be reconnected to the circuit for a separate run-time phase.

Alternatively, the method of indirect programming, as was introduced in [4], does not require a disconnection of the FG transistor. The schematic of Fig. 2 shows the basic configuration of FG transistors programmed via this method. With indirect programming, two transistors share a common FG. In consequence, if the charge on the FG is altered, then the current flowing through both of the transistors will both change. Therefore, one transistor in the pair can be reserved for injection because it will need to be connected to programming logic, as described in [3]. The other transistor can be connected to the circuit directly; it will never need to be disconnected for programming because the FET reserved for injection (called the "programmer") performs all the injection.

Using the indirect method of FG programming provides several distinct advantages over the direct method, including reduced parasitics (because of no disconnection circuitry), the ability to program nFETs (which is difficult to do with the direct method because of process-control techniques), and the freedom to program without a specified programming phase.

III. Capacitive-Coupling Effects with Floating-Gate Transistors

When programming an indirectly programmed floating-gate (IPFG) pair during run-time conditions, care must be taken when modifying the terminal voltages of both transistors because the voltage on the FG node is set by a combination of the FG charge and a sum of the inputs to the gate through capacitive dividers [8]. The extension of the the FG voltage for the indirect programming case is depicted in Fig. 3(a) and described by

$$V_{FG} = \frac{Q_{FG}}{C_T} + \frac{C_{in}}{C_T}V_g + \frac{C_{tun}}{C_T}V_{tun} + \frac{C_{gd,p}}{C_T}V_{d,p} + \frac{C_{gs,p}}{C_T}V_{s,p} + \frac{C_{gw,p}}{C_T}V_{w,p} + \frac{C_{ox,p}}{C_T}\psi_p + \frac{C_{gd,a}}{C_T}V_{d,a} + \frac{C_{gs,a}}{C_T}V_{s,a} + \frac{C_{gb,a}}{C_T}V_{b,a} + \frac{C_{ox,a}}{C_T}\psi_a \quad (1)$$

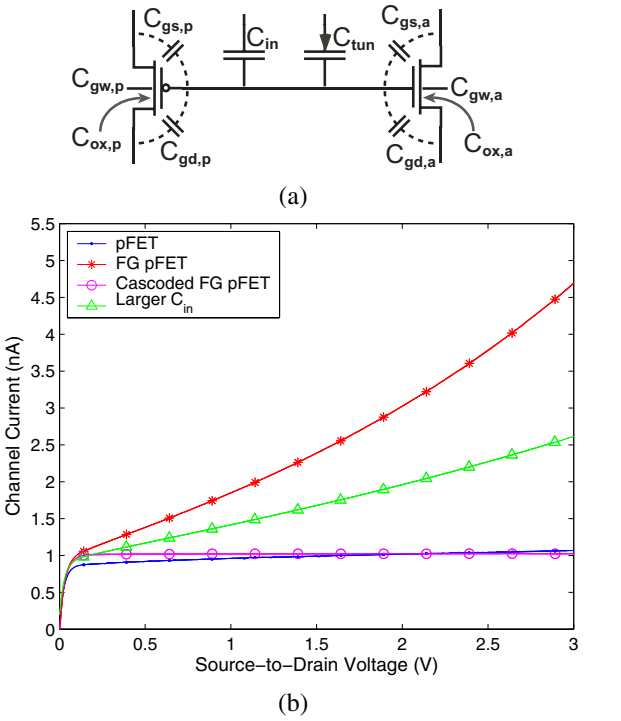

Fig. 3. (a) Schematic of a pair of transistors sharing the same floating gate and the parasitic capacitances that allow coupling of voltages onto the floating node. (b) Transistor drain sweeps. Due to capacitive coupling through $C_{gd,a}$, I_{sat} in the FG pFET increases exponentially for larger $V_{ds,a}$ values. Increasing C_{in} increases the effective Early voltage. Cascoding the agent transistor eliminates the exponential current increase and flattens I_{sat} more than the I_{sat} of the identically sized non-FG pFET.

where C_T is the total capacitance connected to the FG node, the p and a subscripts indicate the programmer (the transistor that performs the injection) and the agent (the transistor that resides within the circuit), and ψ represents the surface potential of each transistor (constant ψ in subthreshold). Therefore, each of the listed nodes acts as an input to the gate and can therefore modify the FG voltage and, accordingly, the currents flowing through the transistors.

As an example, if the gate-to-drain capacitance of the agent, $C_{gd,a}$, is not sufficiently smaller than the total capacitance, C_T, then the drain of the transistor acts as an input to the gate. As the drain voltage of the agent is swept, the saturation current through the device changes exponentially, as is shown in Fig. 3(b). This is a significant alteration from the small slope due to the Early voltage of an identically sized transistor, which is also shown.

Coupling through the gate-to-drain capacitances is not the only source of coupling onto the floating node. In fact, all the terminals affect the drain currents of the two transistors to varying degrees by coupling onto the floating node, as was shown in Eq. (1). These varying degrees depend on both the total capacacitance, 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.

IV. 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 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.

A. Basic Operation of Run-Time Programming

While typical methods of programming FG transistors using hot-electron injection [9] 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 since 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 the circuit will not be temporarily rendered useless (and thus negating the benefits of using of run-time programming).

To recalibrate an FG agent 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 Eq. (1) and Fig. 3(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 pulling the source and drain apart "symmetrically" about V_{FG}, the source-to-drain potential is increased until the point at which 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 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.

Figure 4 shows the operation of injecting the current to a desired value with this process. The small discontinuities 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 capacitance values due to differing parasitic capacitances in weak and strong inversion. These discontinuities can be accounted for, and 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 [10].

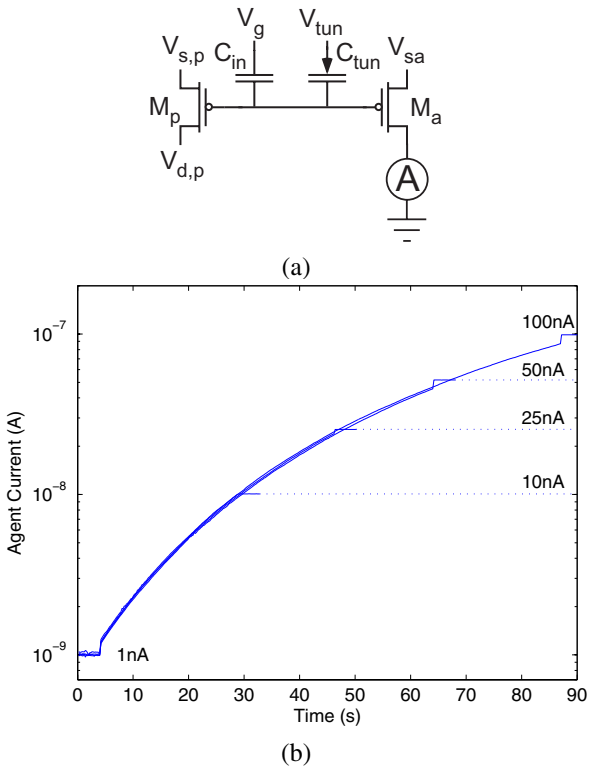

Fig. 4. Run-time programming using indirectly programmed FG transistors. (a) Schematic for programming the agent current using run-time programming. (b) At $t = 5$s, injection was turned on by symmetrically changing the source, well, and drain potentials of the programmer such that the contribution of 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. Run-Time Programming of a Lowpass Filter

To test the operation of run-time programming within a circuit, the circuit of Fig. 5(a) 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 lowpass 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) [11] in which the bias current is set with an IPFG transistor.

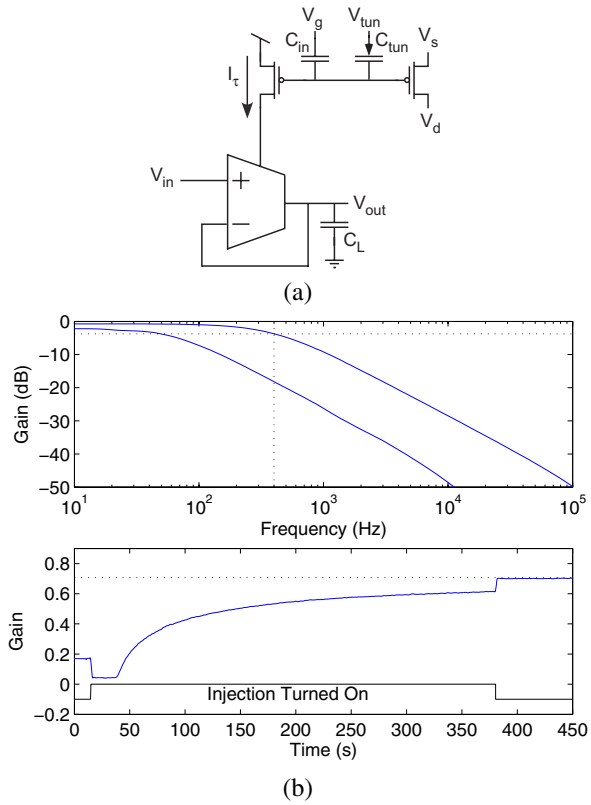

Fig. 5. Run-time programming of a lowpass filter. (a) Simple G_m-C first-order lowpass filter using an indirectly programmed tail current. (b) The filter initially had a corner frequency below 10Hz and was to be reprogrammed to 400Hz without stopping the operation of the filter. By looking at the output of the filter for an input of a 400Hz 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 -3dB 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.

In this simple experiment showing how run-time programming can be used, the corner frequency of the filter was programmed to below 10Hz. However, it was desired that this corner frequency should be moved to exactly 400Hz without stopping the operation of the circuit. As a result, the output of the filter was viewed as 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. Figure 5 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.

V. Conclusion

This run-time approach to programming FG transistors has promising new possibilities for circuits needing frequent updates due to environmental changes and consumer needs. As long as the voltages associated with programming the FG transistors move symmetrically about the FG so that equal but opposite charges couple onto the floating node, then no significant disturbances to the output will be made; the only change in the circuit's operation is due solely to the effect of hot-electron injection.

We have presented the case in which only a single indirectly programmed FG pair is used at a single time. However, much like the case of directly programmed FG transistors, these indirectly programmed FG transistors may also be placed into a large array for large system applications in which each IPFG is individually programmed, and by extension, can be reprogrammed during a run-time phase without adversely harming the operation of the large system.

Additionally, this type of approach can be extended to provide a mechanism for adaptive applications in which weights are continuously updated by a constant modification of the stored charge on the FG node. Such a circuit, which has been presented elsewhere [12], and also the techniques presented here on run-time programming provide a new means to continuously modify a circuit's operation while it continues to operate.

References

[1] D. Graham, P. Smith, R. Chawla, and P. Hasler, "A programmable bandpass array using floating-gate transistors," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, May 2004, pp. I–97 I–100.

[2] T. Hall, C. Twigg, J. Gray, P. Hasler, and D. Anderson, "Large-scale field-programmable analog arrays for analog signal processing," *IEEE Transactions on Circuits and Systems I*, vol. 52, p. 22982307, Nov. 2005.

[3] P. Smith, M. Kucic, and P. Hasler, "Accurate programming of analog floating-gate arrays," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 5, May 2002, pp. 489–492.

[4] D. Graham, E. Farquhar, B. Degnan, C. Gordon, and P. Hasler, "Indirect programming of floating-gate transistors," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 3, May 2005, pp. 2172–2175.

[5] Y. Berg, T. S. Lande, and Ø. Næss, "Programming floating-gate circuits with UV-activated conductances," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 12–19, Jan 2001.

[6] E. Rodriguez-Villegas and H. Barnes, "Solution to trapped charge in FGMOS transistors," *Electronics Letters*, vol. 39, no. 19, pp. 1416–1417, Sept. 2003.

[7] 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 Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 106–110, Jan. 2001.

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

[9] G. Serrano, P. Smith, H. J. Lo, R. Chawla, T. Hall, C. Twigg, and P. Hasler, "Automatic rapid programming of large arrays of floating-gate elements," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol. 1, May 2004, pp. 1373–1376.

[10] R. Harrison, J. Bragg, P. Hasler, B. Minch, and S. Deweerth, "A CMOS programmable analog memory cell array using floating-gate circuits," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 4–11, Jan. 2001.

[11] C. Mead, *Analog VLSI and Neural Systems*. Massachusetts: Addison-Wesley, 1989.

[12] P. Hasler and J. Dugger, "An analog floating-gate node for supervised learning," *IEEE Transactions on Circuits and Systems I*, vol. 52, no. 5, pp. 834–845, May 2005.