Floating-Gate Devices, Circuits, and Systems

Paul Hasler
Georgia Institute of Technology, Atlanta, Georgia 30332–0250, Email: phasler@ece.gatech.edu

***Abstract*— This paper describes our programmable analog technology based around floating-gate transistors that allow for non-volitile storage as well as computation through the same device. We describe the basic concepts for floating-gate devices, capacitor-based circuits, and the basic charge modification mechanisms that makes this analog technology programmable. We describes the techniques to extend these techniques to program an nonhomogenious array of floating-gate devices.**

I. Impact of Programmable Analog Circuits

Over the last decade floating-gate circuit approaches have progressed from a few foundational academic results [1], [2] to a stable circuit and system technology with both academic and industrial applications. This programmable analog technology empowers analog signal processing approaches programmable precision analog low-power techniques. An analog technology that is programmable can enable analog components to be seen as nearly user-friendly as configurable digital options. This approach allows power efficient computing for analog signal processing of 1000 to 10,000 times more efficient than custom digital computation, making a range of portable applications not possible for over a decade a possibility today.

The goal of this paper and the corresponding tutorial session develop general understanding of these programmable analog techniques. The next few sections detail the basic concepts of programmable analog technology. Section II discusses the basic concepts for Floating-Gate devices. Chapter III describes capacitor-based circuits, which are the basis of Floating-Gate Circuit approaches. Chapter IV describes the basic mechanisms for modifying the charge on a floating-gate device, and therefore making this analog technology programmable. Chapter V describes the techniques to extend these techniques to program an array of floating-gate devices, where each could be performing different computations. Chapter VI describes

In particular, many of these techniques are valuable given the development of large-scale Field Programmable Analog Arrays (FPAAs) which could allow applications for non -IC designers by having ICs available..

II. Floating-Gate Circuit Basics

Figure 1a shows the layout, cross–section, and circuit symbol for the floating–gate pFET device [3]. A floating gate is a polysilicon gate surrounded by silicon–dioxide. Charge on the floating gate is stored permanently, providing a long–term memory, because it is completely surrounded by a high–quality insulator. From the layout, we see that the floating gate is a polysilicon layer that has no contacts to other layers. This floating gate can be the gate of a MOSFET and can be capacitively connected to other layers. In circuit terms, a floating gate occurs when we have no DC path to a fixed potential. No DC path implies only capacitive connections to the floating node, as seen in Fig. 1.

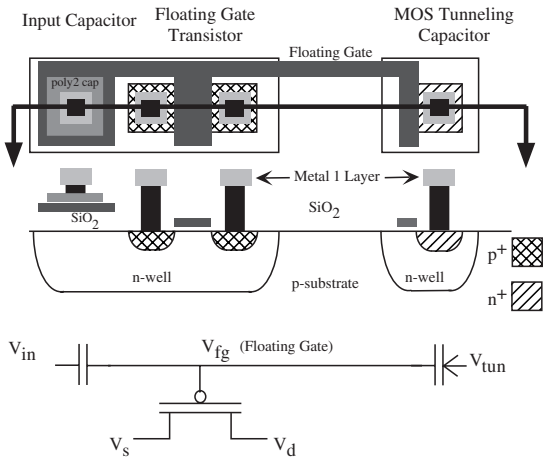

Fig. 1. Layout, cross section, and circuit diagram of the floating–gate pFET in a standard double–poly, n-well MOSIS process: The cross section corresponds to the horizontal line slicing through the layout view. The pFET transistor is the standard pFET transistor in the n-well process. The gate input capacitively couples to the floating–gate by either a poly–poly capacitor, a diffused linear capacitor, or a MOS capacitor, as seen in the circuit diagram (not explicitly shown in the other two figures). Between V_{tun} and the floating–gate is our symbol for a tunneling junction—a capacitor with an added arrow designating the charge flow.

The floating–gate voltage, determined by the charge stored on the floating gate, can modulate a channel between a source and drain, and therefore, can be used in computation. As a result, the floating-gate device can be viewed as a single transistor with one or several control gates where the designer controls the coupling into the surface potential. Floating–gate devices can compute a wide range of static and dynamic translinear functions by the particular choice of capacitive couplings into floating–gate devices [4].

III. Floating-Gate Circuits enabling Capacitive Circuits

Floating–gate circuits provide IC designers with a practical, capacitor–based technology; since capacitors, rather than resistors, are a natural result of a MOS process. Figure 2 shows key basic capacitor circuit elements. Figure 2a shows the capacitive equivalent to a resistive voltage divider. The resulting expression is expected from a resistive voltage divider, except that we have an additional voltage, V_{charge}, that is set by the charge (Q) at the output node, as V_{charge} = Q / ($C_1 + C_2$). Figure 2b shows the capacitor circuit for feedback around an

Proceedings of the 9th International Database Engineering & Application Symposium (IDEAS'05)
1098-8068/05 $20.00 © 2005 **IEEE**

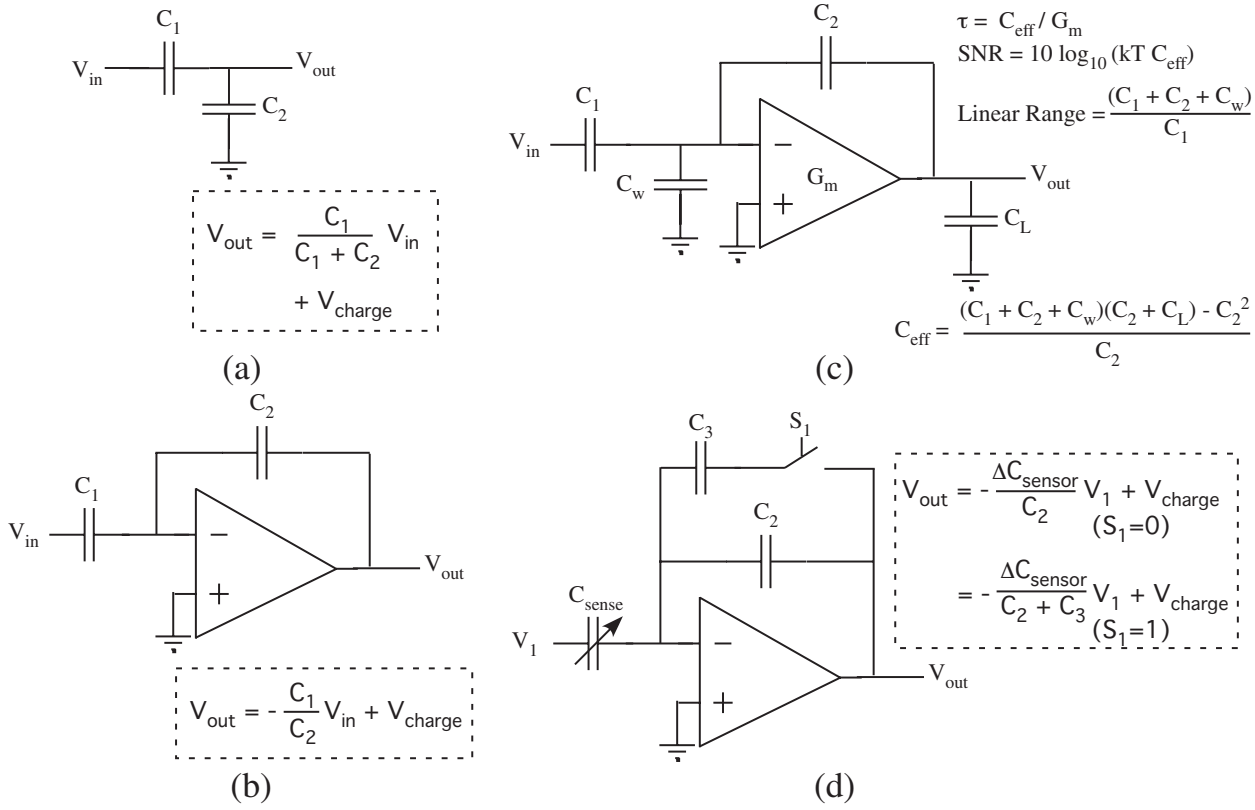

Fig. 2. Basic capacitor circuits. In all cases, we have a charge term. (a) Capacitive Divider Circuit. (b) Capacitive Feedback around an amplifier. We assume the amplifier has MOS inputs; therefore inputs are practically only capacitive. (c) Assuming amplifier has a finite Gm, we identify the key parameters (bandwidth, SNR, and input linear range) for this amplifier (d) Circuit Modification for direct measurement of capacitive sensors.

amplifier. As expected, the closed loop for this amplifier is $-C_1/C_2$; the output of this amplifier also has a voltage term due to the charge stored (Q) at the '-' input terminal, where $\mathrm{V}_{charge} = Q/C_2$.

Figure 2c shows a more realistic circuit model for the amplifier circuit in 2b. This circuit description, which includes parasitic capacitances, is described as a single pole and zero system, assuming the amplifier sets a single pole (i.e. the amplifier has a frequency independent transconductance. In general, we describe the amplifier as a transconductance amplifier, assuming the gain set by the capacitors is lower than the amplifier gain. Increasing C_w increases the input linear range; typically, C_w is larger than C_1 and C_2, where C_w models the input capacitance of the amplifier, explicitly drawn capacitance, and parasitic capacitances. Increasing the function $C_w C_L/C_2$ proportionately increases SNR (in signal power); therefore, unlike many filters where output noise and SNR is set by the load capacitance (kT/C thermal noise), this function allows for lower drawn capacitances for a given noise floor. When we improve the linear range of this amplifier, we simultaneously improve the SNR of the amplifier. These circuit approaches extend Transconductance-C filter approaches to allow some parameters to be set by the ratio of capacitors [5], such as bandpass gain, the linear range, the noise level, and bandwidth. These approaches, coupled with the array programming discussions in Section IV, results in a highly accurate low-power filter technology.

Figure 2d shows a slight extention of the other circuits towards measuring changes in capacitive sensors (i.e. MEMs sensors). Analyzing this circuit with an ideal amplifier with gain A_v, we get

$$V_{out} = V_1 \frac{\Delta C_{sensor} + \frac{1}{A_v}(C_{sensor} + C_w)}{C_2},$$

$$\Delta V_{out} = V_1 \frac{\Delta C_{sensor}}{C_2}, \qquad (1)$$

where C_w is the capacitance at the '-' amplifier input, including the amplifier input capacitance. This circuit configuration attenuates the effect of of sensor capacitance and C_w by the amplifier gain, an effect that is only a dc term assuming V_1 remains constant. For example, for C_{sensor} = 1pF, maximum ΔC_{sensor} = 2fF (typical of some MEMs sensors), and A_v = 1000, we choose C_2 = 20fF for a maximum V_{out} change of 1V, resulting in an output offset voltage of 0.25V. The constant part of C_{sense}, as well as C_w, increases the linear range of the closed-loop circuit.

The dynamic performance of this amplifier is similar to techniques for Fig. 2. The circuit is still a first-order system (assuming a frequency independent amplifier Gm) described as

$$\frac{V_{out}}{\Delta C_{sensor}} = \frac{V_1}{C_2} \frac{1 - s(C_2/G_m)}{1 + s\tau} \qquad (2)$$

with the same timeconstant (τ) as the amplifier in Fig. 2c, and the zero, due to capacitive feedthrough, is typically at much higher frequency responses than the amplifier bandwidth. Typically, C_{sensor} and C_L are roughly equal size (C_L might be larger) and (the output load capacitance, as in Fig. 2c) are larger than C_2. For the example above, with $C_L = C_{sensor} = 1pF$, the resulting bandwidth

Transconductance (G_m)	Bias Current	Bandwidth
1 (kΩ)$^{-1}$	30μA	3MHz
10 (kΩ)$^{-1}$	300μA	30MHz

(3)

The resulting output noise (entire band) is

$$\hat{V}_{out} = \sqrt{qV_{IC}n\frac{(C_{sensor} + C_w)}{C_2 C_L}} \tag{4}$$

where n is the equivalent number of devices contributing noise for the amplifier, and V_{IC} is the ratio of the transconductance (G_m) and the differential pair transistor's bias current. For the example above, with a typical low-noise amplifier stage with input transistors operating with subthreshold currents results in 0.5mV total noise, resulting in an SNR of roughly 66dB between the maximum capacitor deflection and the minimum deflection. For our example circuit, the maximum capacitance change of 2fF gives an output of 1V, where a 1aF change is at the 0dB SNR level; by restricting the bandwidth of interest or making the amplifier bandwidth larger than the bandwidth of interest, the resulting sensitivity will increase. In practice, we often allow for a bank of capacitors that can be switched into the circuit to alter C_2, and therefore the dynamic range and noise of these signals. Figure 2d shows the switching between gain levels; the switch is not at the charge storage node because a MOS switch on the '-' terminal increases the leakage current at this node, decreasing hold time. All of these results have been experimentally verified through use of variable MEMs capacitor devices. In one particular system, one sees 100aF capacitor change resulting in 37.5mV change for an amplifier with noise significantly less than 1mV; therefore, 3fF change resulted in a 1.13V output swing, and 3aF change resulted in a 1mV output swing (0dB SNR point). The bandwidth of the amplifier was greater than 1MHz.

IV. Modifying Floating-Gate Charge

transistor I-V curve; therefore transistor elements

We modify the floating–gate charge by applying large voltages across a silicon–oxide capacitor to tunnel electrons through the oxide or by adding electrons using hot–electron injection. Although we extensively discuss the physics of floating-gate circuits elsewhere [2], [6], [7], we briefly review the necessary physics below.

A. Electron Tunneling

We add charge to the floating gate by removing electrons using electron tunneling. Increasing the voltage across this tunneling capacitor, either by increasing the tunneling voltage (V_{tun}) or decreasing the floating-gate voltage, increases the effective electric field across the oxide, thereby increasing the probability of the electron tunneling through the barrier. Starting from the classic model of electron tunneling, given as

$$I_{tun} = I_0 e^{(\mathcal{E}_0 t_{ox})/V_{ox}}, \tag{5}$$

where $\mathcal{E}_0$ is a fundamental parameter derived from a WKB solution of Schrodingers equation, I_0 is an emperically measured parameter, t_{ox} is the thickness of the oxide dielectric, and V_{ox} is the voltage across the dielectric, we can derive an approximate model for the electron tunneling current around a given voltage acorss the oxide (tunneling voltage minus floating-gate voltage) as [2], [6]

$$I_{tun} = I_{tun0} e^{(V_{tun} - V_{fg})/V_x}, \tag{6}$$

where V_x is a tunneling device-dependent parameter that is a function of the bias voltage across the oxide.

V. pFET Hot-Electron Injection

We use pFET hot-electron injection to add electrons (remove charge) to the floating-gate. We use pFET hot-electron injection, because pFET hot-electron injection can not be eliminated from a CMOS process without adversely affecting basic transistor operation, and therefore will be available in all commercial CMOS processes. One might wonder how pFETs, where the current carriers are holes, inject hot electrons onto the floating gate. Figure 1b shows the band diagram of a pFET operating under bias conditions that are favorable for hot-electron injection. The hot-hole impact ionization creates electrons at the drain edge of the drain-to-channel depletion region, due to the high electric fields there. These electrons travel back into the channel region, gaining energy as they go. When their kinetic energy exceeds that of the silicon–silicon-dioxide barrier, they can be injected into the oxide and transported to the floating gate. If we are to inject an electron onto a floating gate, the MOSFET must have a high-electric-field region ($> 10\text{V}/\mu m$) to accelerate channel electrons to energies above the silicon–silicon-dioxide barrier, and in that region the oxide electric field must transport the electrons that surmount the barrier to the floating gate. As a result, the subthreshold MOSFET injection current is proportional to the source current, and is the exponential of a smooth function of the drain-to-channel potential (Φ_{dc}); the product of these two circuit variables is the key aspect necessary to build outer-product learning rules. We present a first-principles model derived from basic physics that shows the resulting exponential functions elsewhere [8].

A simplified model for pFET injection that is useful for hand calculations relates the hot-electron injection current for a channel current (I_s) and drain-to-source (ΔV_{ds}) voltage as

$$I_{inj} = I_{inj0} \left(\frac{I_s}{I_{s0}}\right)^{\alpha} e^{-\Delta V_{ds}/V_{inj}}, \tag{7}$$

where I_{inj0} is the injection current when the pFET is operating with a channel current reference (I_{s0}), where $I_s = I_{so}$ at this reference current, and a drain-to-source voltage, V_{inj} is a device and bias dependent parameter, and α is $1 - \frac{U_T}{V_{inj}}$.

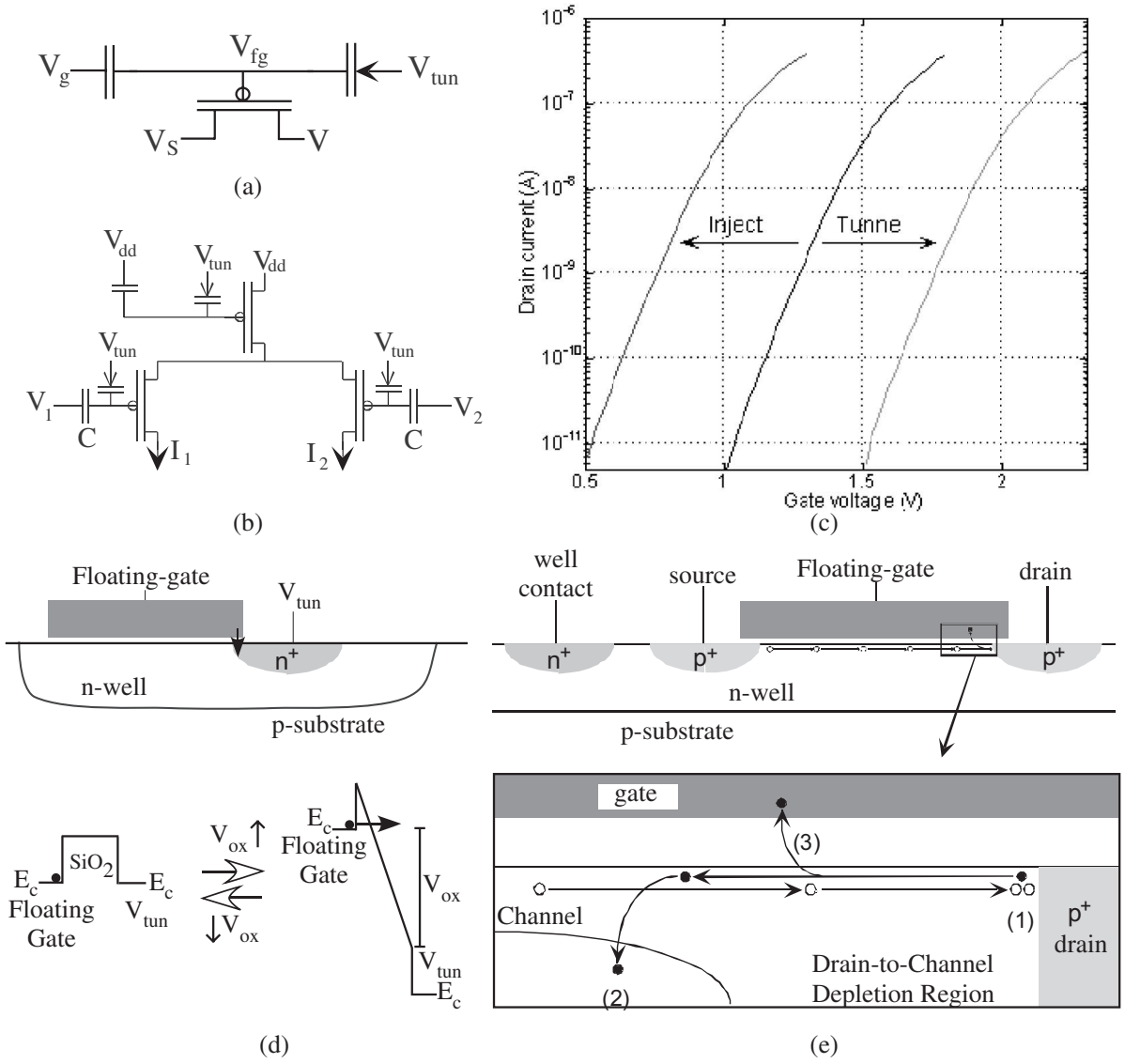

Fig. 3. Approach to modifying floating-gate charge. (a) Basic Circuit representation for a floating-gate device. (b) A programmable floating-gate differential pair. Both the input transistors as well as the current source transistor are programmed. Practically, we want to develop approaches to program the offset voltage for the amplifier, as well as the value for the current source. (c) Current–Voltage curves from a programmed pFET transistor. We modify charge by a complimentary combination of electron tunneling (weaker pFET transistor) and hot-electron injection (stronger pFET transistor). (d) Basic picture of electron tunneling in Si-SiO_2 system (e) Basic picture of pFET hot-electron injection. Some holes moving through the channel gain sufficient energy to create an impact ionization event, generating electrons that increase in energy moving towards the channel. Some of these electrons will have sufficient energy to surmount the Si-SiO_2 barrier and arrive at the gate terminal.

Typical values for V_{inj} in a 0.5μm CMOS process are $100mV$ to $250mV$.

Choosing the appropriate model for simulation is critical for these devices. For example, when simulating a set of floating-gate devices that will be programmed, one typically does not need to implement the tunneling and injection currents, but rather make sure at the beginning of the simulation that the floating-gate voltages / bias currents are set correctly based upon the behavior of the particular programming scheme. In this mode of operation, one can set the floating-gate voltage through a very large resistor; for the total capacitance at a floating-gate node of 100fF (a small device), a resistor of 10^{26} is consistent with the typical room temperature voltage drop of 4μV over 10 year period for a 10nm oxide [9]. In some cases, transistor equivalent circuits can be used to simulate adaptive floating-gate elements, such as the C^4 circuit and the ATS synapse element [5], [10]; these techniques tend to be useful circuits in their own right for applications requiring fast adaptation rates.

VI. Accurate Programming of Programmable Analog

The charge modification schemes, along with their detailed modeling, opens the door for accurate programming of a large number of floating-gate devices being utilized by a diverse set of circuits. Figure 4 shows the starting point for the story

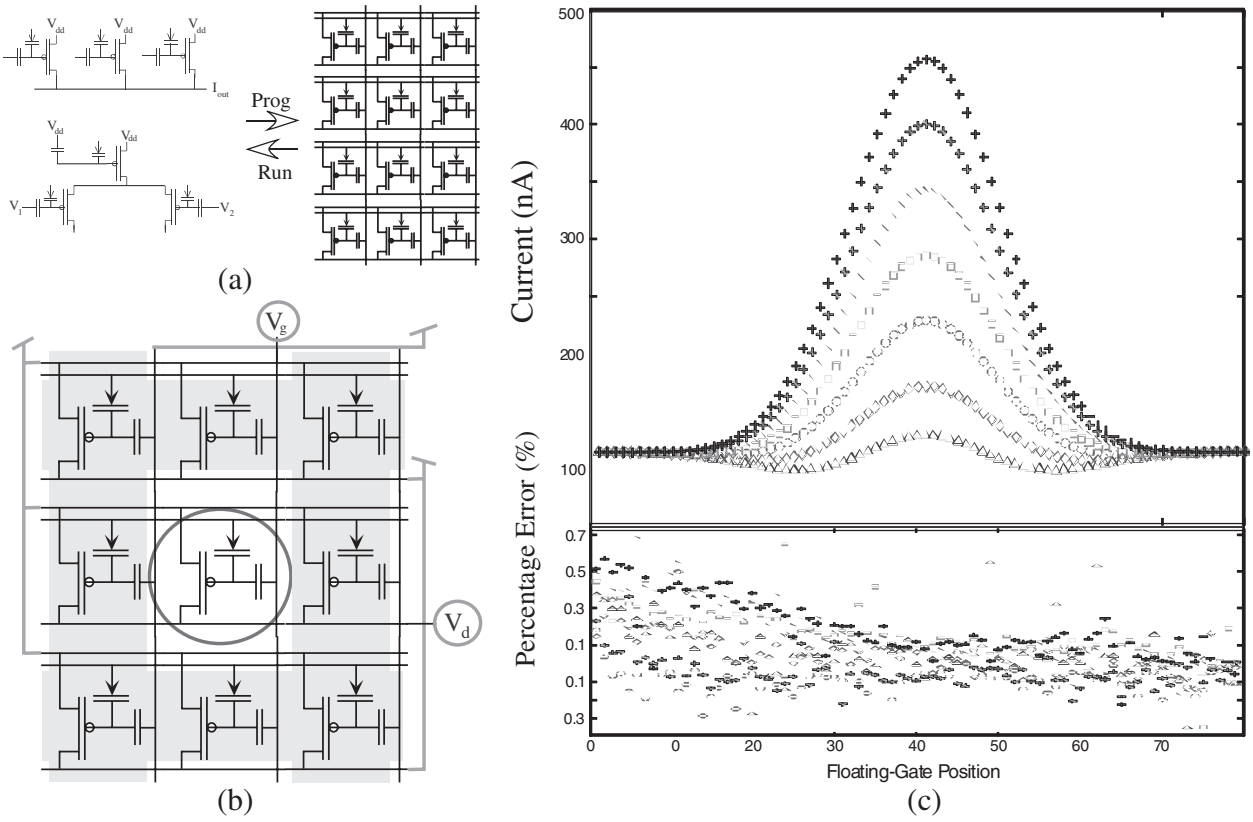

Fig. 4. Programming of Large Number of Floating-Gate elements. (a) Infrastructure takes any number of floating-gate elements on a chip during **run** mode, and reconfigures these devices into a regular array of floating-gate elements. (b) Hot electron injection requires both channel current (subthreshold) and high-electric field; therefore in an array of devices we can access a single element using an AND scheme, both for measurement and for programming. (c) Experimental measurements for programming a sequence of functions of different amplitudes. The corresponding percentage error is plotted below the data; the experimental error (for this example bounded between 0.7 percent and 0.3 percent) does not correlate with the experimental waveform.

of automatically programming a large array of floating-gate elements. Figure 4a illustrates how we access programmable devices, which we define as **Prog** or program mode, and how we we compute using these elements, which we define as **Run** mode. When we go from **Run** mode to **Prog** mode, we electrically reconfigure all circuits such that each floating-gate device is configured into a two-dimensional mesh array of devices with the drain and gate lines moving in orthogonal directions. We isolate individual elements (access to an individual gate and drain line) in a large matrix using peripheral control circuitry [3], [7]. A standard technique is necessary when working with thousands and millions of floating-gate elements on a single die.

Our programming scheme minimizes interaction between floating–gate devices in an array during the programming operation. Other elements are switched to a separate voltage to ensure that those devices will not inject. We program a device by increasing the output current using hot-electron injection, and erase a device by decreasing the output current using electron tunnelling. Because of the poorer selectivity, we use tunnelling primarily for erasing and for rough programming steps. Our programming scheme performs injection over a fixed time window (from 1μs to 10s and larger) using drain-to-source voltage based on the actual and target currents. Most rapid programming techniques use pulse widths in the 10μs to 100μs range. which potentially enable programming large arrays of floating-gate elements in mass production. Developing an efficient algorithm for pFET programming requires discussing the dependencies of the gate currents as a function of the drain-to-source voltage. This scheme also measures results at the circuit's operating condition for optimal tuning of the operating circuit (no compensation circuitry needed). Once programmed, the floating-gate devices retain their channel current in a non-volatile manner. We designed a custom programming board (PCB board) to program large floating-gate arrays around these standards [7], [11], and have been developing approaches to move all of these blocks on-chip using row-parallel programming techniques [12]. These approaches have been used by over 40 researchers on hundreds of IC projects. The limiting factor for rapid programming of these devices is the speed and accuracy of our current measurements; the hot-electron injection physics does not limit the speed of the programming with current approaches.

Often, one is asked how accurately can these devices be programmed. In the end, the accuracy is limited by change in voltage (ΔV) due to one electron (q) moving off of the total capacitance (C_T) at the floating node, expressed as

$$\Delta V = q/C_T. \tag{8}$$

For C_T of 16fF (a small device), the voltage change from one electron is 10μV. For a voltage source based on the floating-gate voltage for a 2V swing, the accuracy is roughly 0.0005 percent or 17 - 18 bits. For a subthreshold current source, the accuracy is roughly 0.05 percent (11 bit) over the entire subthreshold current range (typically 6 to 8 orders of magnitude). The error decreases inversely proportional to an increasing C_T.

Therefore, the real question is how accurately can the programming algorithm achieve this theoretical limit. First, the limitation is the accuracy of measuring the quantity we desire to program; for example, if we only have 8bit accuracy to measure the output current we want to program, then we can not expect to achieve significantly higher programming performance through that measurement. Second, is the limitation of the programming algorithm and associated circuitry, including parasitic elements; our approaches are designed to minimizes the effect of parasitic elements by design. On the other hand, due to factors like capacitor mismatch, we can usually have fine-tuning programming steps to improve the effects due to these mismatches. [5], [13] Finally, C_T can set the thermal noise (kT/C noise) for the previous stage; a 16fF capacitor will set a total noise for a single subthreshold device as 0.25mV, an error if not addressed in the resulting circuit, will be 25 times greater than the programming accuracy. Figure 4c shows measurements of programming accuracy from an array of floating-gate devices ($C_T \approx 100$fF). Typical experimental results show that we can program within 0.5 to 0.1 percent accuracy over 3.5 decades of target currents, we can program at least as well as 1 percent accuracy over 6 decodes of target currents, in CMOS processes ranging from 0.5μm to 0.25μm CMOS processes [14].

VII. Acknowledgements

This paper has been heavily based upon the research in the CADSP laboratory at Georgia Tech, founded by David Anderson and myself. In particular, I greatly appreciate the many discussions with graduate students that have helped shape the focus of this paper: Farhan Adil, Abhishek Bandyopadyady, Ravi Chawla, Chris Duffy, Jeff Dugger, David Graham, Jordan Gray, Tyson Hall, Matt Kucic, Guillermo Serrano, Paul Smith, Venkatesh Srinivasan. and Chris Twigg.

References

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

[2] P. Hasler, C. Diorio, B. A. Minch, and C. A. Mead, *Advances in Neural Information Processing Systems 7*. Cambridge, MA: MIT Press, 1995, ch. Single transistor learning synapses, pp. 817–824.

[3] P. Hasler and T. S. Lande, "Special issue on floating-gate devices, circuits, and systems," *IEEE Journal of Circuits and Systems*, vol. 48, no. 1, Jan. 2001.

[4] B. A. Minch, P. Hasler, and C. Diorio, "Multiple-input translinear element networks," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, pp. 20–28, Jan. 2001.

[5] D. Graham, P. D. Smith, R. Ellis, R. Chawla, and P. Hasler, "Programmable bandpass array using floating-gate elements," in *Proceedings of the International Symposium on Circuits and Systems*, Vancouver, BC, May 2004.

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

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

[8] C. Duffy and P. Hasler, "Modeling hot-electron injection in pfet's," *Journal of Computational Electronics*, vol. 2, pp. 317–322, Jan. 2001.

[9] V. Srinivasan, G. J. Serrano, J. Gray, and P. Hasler, "A precision cmos amplifier using floating-gates for offset cancellation," in *Custom Integrated Circuits Conference*, San Jose, Sept. 2005.

[10] V. Srinivasan, J. Dugger, and P. Hasler, "An adaptive analog synapse circuit that implements the least-mean-square learning algorithm," in *Proceedings of the International Symposium on Circuits and Systems*, Kobe, Japan, May 2005.

[11] G. Serrano, P. Smith, H. Lo, R. Chawla, T. Hall, C. Twigg, and P.Hasler, "Automatic rapid programming of large arrays of floating-gate elements," in *Proceedings of the International Symposium on Circuits and Systems*, Vancouver, May 2004.

[12] M. Kucic, Ed., *Analog Computing Arrays*. Atlanta, GA: Ph.D. Dissertation, Georgia Institute of Technology, 2004.

[13] A. Bandyopadhyay, J. Lee, R. Robucci, and P. Hasler, "A 80uw/frame 104x128 cmos imager front-end for jpeg compression," in *Proceedings of the International Symposium on Circuits and Systems*, Kobe, Japan, May 2005.

[14] A. Bandyopadhyay, G. Serrano, and P. Hasler, "Programming analog computational memory elements to 0.2 percent accuracy over 3.5 decades of current," in *Proceedings of the International Symposium on Circuits and Systems*, Kobe, Japan, May 2005.