A Floating-Gate Memory Cell for Continuous-Time Programming

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

***Abstract*—As an apt choice for long-term analog memory in standard CMOS processes, floating-gate transistors are key enablers for large-scale programmable analog systems. Such systems are often designed for battery-powered—and generally resource-constrained—applications, which require the memory cells to program quickly with low infrastructural overhead. In order to meet these needs, we present a new analog floating-gate memory cell. Our four-transistor memory cell offers both voltage and current outputs and has linear injection and tunneling characteristics. Furthermore, we present a simple programming circuit that forces the memory cell to converge to voltage targets within 100ms and with 8-bit accuracy.**

I. Introduction

In addition to their role as nonvolatile memory elements in Flash memory, floating-gate (FG) transistors are used for programmable voltage/current references, precision analog device matching, and adaptive/learning circuits [1]. An FG transistor (schematic representation shown in Fig. 1) is a MOSFET that has no resistive connection to its gate; instead, a "control gate" couples capacitively onto the transistor's "floating gate." As a result, the FG's charge, which can be modified using Fowler-Nordheim tunneling and hot-electron injection, creates a programmable and nonvolatile threshold-voltage shift from the perspective of the control gate.

In order to modify the charge on the FG, high voltages are applied to the FG transistor's terminals. The charge can be programmed to a desired amount by using either pulsed or continuous methods, as illustrated in Fig. 1. Pulsed methods operate by applying short programming pulses and then measuring the FG after each pulse. In contrast, continuous methods continuously apply the programming voltage and use feedback to force the FG to converge to the target. Such continuous programming promises to be faster and require less peripheral circuitry than pulse-based programming. In this paper, we present a compact FG cell for continuous programming, which, when combined with our simple programmer circuit, converges to target voltages with 8-bit accuracy within 100ms.

Our basic memory cell uses the FG transistor in a source-follower configuration and linearizes injection via negative feedback to the control gate, as in Fig. 1. Such linear source-feedback injection has been used previously in [2], but we accomplish the same characteristics with the smaller current conveyor circuit that we introduce in Section IV. In addition to being smaller, this current conveyor memory cell also offers more flexible control over the injection rate since V_s can be modified using either a voltage or a current input.

This work was supported in part by the United States Army Research Laboratory under Contract W911NF-10-2-0109.

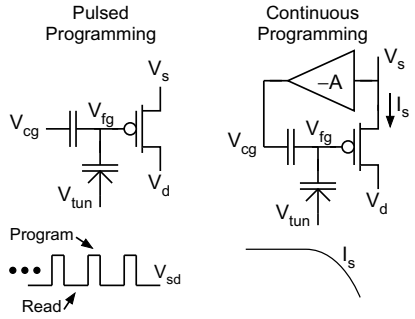

Fig. 1. Pulsed programming and continuous programming. In pulsed programming, the source-to-drain potential is alternately pulsed high for injection, and then placed at a nominal value to measure the floating gate. In continuous programming, injection occurs constantly, and a terminal (in this case the source current) is adjusted to decrease—and eventually shut off—injection as the target is approached.

II. Overview of Floating-Gate Programming

Two phenomena are typically used to program FG transistors: hot-electron injection and Fowler-Nordheim tunneling. Injection occurs when a large source-to-drain potential (V_{sd}>3.5V for 0.35μm) is applied to the FG transistor, thus causing high-energy carriers to impact-ionize at the drain. A fraction of the resulting ionized electrons disperse toward the surface with enough energy to overcome the oxide barrier and inject onto the FG. In the subthreshold region, which is our target operational region, the injection current from V_{fg} to V_d can be approximated as

$$I_{inj} \approx \beta {I_s}^{\alpha} e^{V_{sd}/V_{inj}} \quad (1)$$

where β, α, and V_{inj} are device-dependent fits [3]. Tunneling, on the other hand, requires high voltages (V_{ox}>8V for 0.35μm). In order to avoid write disturbs during tunneling, unselected array elements must either be disconnected from the tunneling voltage using high-voltage switches or the FGs of the unselected elements must be raised to a sufficient voltage that tunneling does not occur. Due to this difficulty in isolating tunneling within an array, tunneling is typically only used for global erasure in analog memory arrays, while injection is used to write to individual elements. Consequently, we focus mainly on injection in this paper.

Due to their ability to provide dense, low-power, analog biases, FGs are elemental in large-scale programmable analog systems—such as filter banks, classifiers, and field-programmable analog arrays. In these systems, circuit param-

978-1-4673-2527-1/12/$31.00 ©2012 IEEE

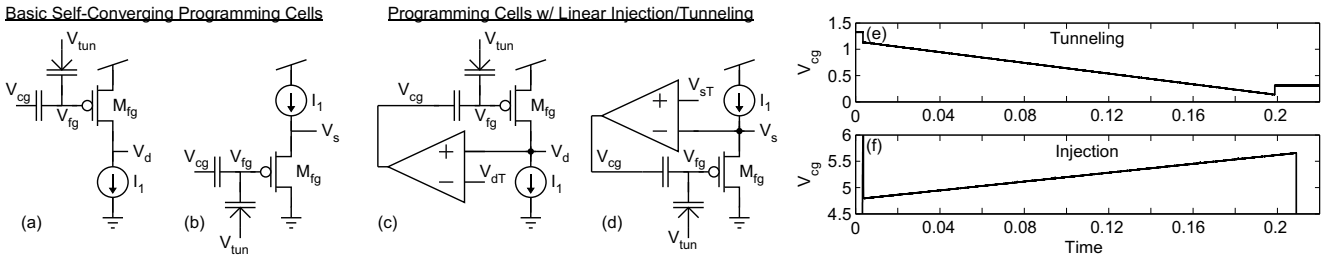

Fig. 2. Canonical forms of programming cells. (a) and (b) are single-branch circuits without explicit feedback to keep all of the terminals of M_{fg} constant, thereby permitting situations that result in slow injection. (c) and (d) employ negative feedback to the gate to hold the terminals and current of M_{fg} constant, thus resulting in linear injection and tunneling. (e) and (f) demonstrate the linear programming characteristics of the circuit in (d).

eters (e.g. corner frequencies) are controlled by the charge on the FGs; as a result, system performance depends strongly on the programming accuracy. Prior pulse-based programming techniques have achieved high accuracy [2], [4]. One advantage that pulse-based techniques have in terms of accuracy is that the FG is measured in a state that is similar to run-mode: with no high program voltages applied to the cell, and with the same current levels that will be used in run-mode. Unfortunately, pulsing is inherently slow due to the time spent reading, during which the high program voltages are stepped down and the FG is allowed to settle before the measurement is taken; if measuring low currents, then the read time further increases due to the long integration time that is necessary for accurate measurement. Methods to increase the programming speed rely on precise knowledge of each FG's characteristics, so that each pulse can move more aggressively towards the target [4]; but this adds to the complexity. Additionally, pulsing techniques require high-precision data conversion and pulse timing, and possibly large-range current measurement, all of which complicate the inclusion of analog FG memory into simple, resource-constrained, systems. Thus, there is a need for fast, compact, low-overhead, and accurate programming: we posit that continuous-time programming is more appropriate for resource-constrained systems.

III. Continuous-Time Floating Gate Programming

Continuous-time FG programming is accomplished by using feedback to stop programming when the memory cell reaches its target value. A variety of continuous programming circuits have been presented: ranging from a single-transistor circuit [3] which self-converges due to the negative feedback of injection current from the FG to the drain, to more complex circuits with improved speed and accuracy. In order to linearize the characteristics of injection/tunneling, most programming circuits use feedback to maintain a constant FG voltage, though the details vary from circuit to circuit: [5] presents a three-transistor memory cell plus a comparator to stop injection once the target is reached; [6] presents a programming circuit which uses a differential FG amplifier to achieve linear tunneling and also to cancel out the tunneling junction's capacitive coupling; [7] builds upon the basic cell in [5] to create a fully integrated continuous-time FG programming system; [8] presents a memory cell which uses both hot-electron and hot-hole injection in order to converge bidirectionally toward the target. In these prior linearized techniques, the programming rate is held constant, and once the target is reached, programming is stopped; such programming faces a severe tradeoff between programming speed and accuracy [7]. In contrast, we adjust the source current to reduce the programming rate as the target is approached in order to achieve a better tradeoff between programming speed and accuracy.

Figure 2 shows four fundamental continuous programming configurations. The two circuits shown in Fig. 2(a) and (b) do not have feedback to linearize programming, but they do have inherent feedback that causes them to self-converge. In both circuits, the injection of electrons onto the FG causes V_{fg} to decrease, which decreases V_{sd} and thereby decreases I_{inj} according to (1). The circuit in Fig. 2(a) [3] can be programmed to different targets either by using different values of I_1 for a constant V_{cg}, or by using different values of V_{cg} for a constant I_1. While this circuit is very compact and produces repeatable results, its convergence time depends on the initial condition: if the initial V_{fg} is too high to supply I_1, then the small initial drain current yields little injection. As a result, convergence can take several seconds. The circuit in Fig. 2(b) has the opposite problem: whereas the circuit in (a) begins slow and finishes fast, the circuit in (b) begins fast and then takes minutes to slowly converge.

The long convergence times of the simple configurations can be addressed by using negative feedback to V_{cg}—shown in Fig. 2(c) and (d)—so that all terminals of M_{fg} are constant, and thus the rate of injection/tunneling is held constant; however, these memory cells no longer converge on their own, but require additional programming circuitry. In both of these circuits, V_{fg} is constant and V_{cg} ramps linearly up during injection, or down during tunneling, to compensate for the change in charge on the FG—see Fig. 2(e) and (f). V_{cg} thus provides our measure of the charge on the FG. We have found the high gain around the loop of the circuit in (c) to cause stability problems, and so we will not consider it any further. The source follower circuit in (d) is the same configuration that has been used in pulse-based source-feedback injection to achieve 13-bit precision with program times on the order of 50sec/200mV [2]. This circuit has good stability and offers good control over injection and tunneling through the manipulation of both V_{sT} (which sets V_s) and I_1. Our memory cell has the same basic characteristics as this circuit, but is smaller, which is important for large array applications.

IV. Current-Conveyor-Based Memory Cell

In order to achieve the good characteristics of the circuit in Fig. 2(d), but reduce the size, we have developed the circuit in Fig. 3(a). For simplicity, current sources are shown for I_1 and I_2, but in the actual implementation, each source is implemented by a single transistor. In this memory cell, the inverting amplifier M_1–I_2 replaces the op-amp in Fig. 2(d). The resulting circuit structure is the current-controlled current conveyor, the details of which can be found in [9]. In this circuit, the negative feedback adjusts V_{cg} in order to force both V_{fg} and V_s to fixed voltages. The equilibrium point for V_s is controlled by both the voltage V_X and the current I_2. The equilibrium point of V_{fg} depends on both V_s and I_1. Thus we maintain independent control of the source current and drain-to-source potential (the two main injection parameters) with this four-transistor circuit.

This memory cell offers three control terminals for modifying injection: two currents (I_1 and I_2) and one voltage (V_X). Using the subthreshold injection approximation in (1), we can solve for the injection current as a function of the control terminals in subthreshold operation

$$I_{inj} \approx \beta {I_1}^{\alpha} \left(\frac{I_2}{I_0}\right)^{-\frac{U_T}{\kappa V_{inj}}} e^{\frac{V_x-(1-\kappa)V_{dd}}{\kappa V_{inj}}} \qquad (2)$$

where I_0 is the pre-exponential current scaler for M_1, κ is the subthreshold slope for M_1, and U_T is the thermal voltage. Figure 3(c)–(d) shows measured injection rates as a function of each of these control terminals. The floating-gate transistor was fabricated in a 0.35μm standard CMOS process, and has dimensions $\frac{W}{L}$=$\frac{1.6\mu m}{0.6\mu m}$ and C_g=60fF. All other transistors were ALD1105 FETs. The injection rate was measured by determining the slope of V_{cg} during injection experiments that were similar to Fig. 2(f); this slope is equal to the injection current normalized by the control gate capacitance C_g. When not being swept, V_X, I_1, and I_2 were held fixed at 5V, 860nA, and 2nA, respectively. Additionally, since the feedback holds V_{fg} constant, this cell has linear tunneling characteristics. Figure 3(b) shows the dependence of the tunneling current on V_X while all other terminals were held fixed. The experiments shown in Fig. 3(b)–(d) demonstrate the ability to adjust the cell's programming rate over a large range using either voltage or current inputs. Additionally, the weak dependence on I_2—approximately an inverse fifth root dependence—makes I_2 appropriate for fine rate adjustment. Furthermore, the cell works well in the subthreshold region, where power consumption is low and (2) holds true.

V. Programming Infrastructure

The combination of control terminals makes the memory cell very flexible in terms of programming circuits. Figure 4(a) illustrates one possible programming circuit, which uses I_1 as the control terminal. The transconductor converts the difference between V_{cg} and the target value, V_{targ}, into a current. This current is rectified by the current mirror M_2–M_3 and is forced into the source terminal of the FG transistor.

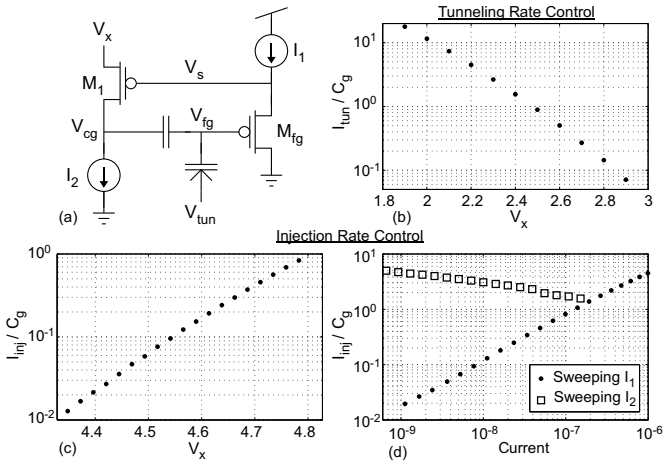

Fig. 3. (a) Our floating-gate memory cell, which is based on the current-controlled current conveyor circuit. (b) Measured dependence of tunneling current on terminal V_X. (c)–(d) Measured dependence of injection current on the three control terminals of the circuit: V_X, I_1, and I_2.

As the target is approached, the injection rate is reduced, and eventually stopped, by the reducing I_1. Figure 4(b) shows a timing diagram of the programming process. During the pre-injection interval (i), the supply voltage is at the run-level value (3V), and the FG has been tunneled to the point that V_{cg} is at ground. When the injection interval (ii) first starts, the supply voltage is ramped up (5.4V), which pulls V_{cg} up, and there is a short time during which the capacitance of node V_{cg} is discharged through I_2; the significant duration of this discharge time is due to the fact that the circuit was prototyped on a breadboard with significant parasitic capacitance. Once V_{cg} is discharged, we observe linear injection while the transconductor's output current is saturated. As V_{cg} approaches the target, I_1 is reduced (see the bottom pane of the plot), and once V_{cg} reaches the target, I_1 is zero. During interval (iii), the current conveyor structure has stopped operating due to I_1 being shut off, and as a result, V_{cg} is pulled high. Then for the read mode interval (iv), the supply is ramped down to run level, and the cell is configured as a voltage reference. The cell's voltage output is read from V_{cg}, and the cell is configured by removing the transconductor from the loop and supplying a constant voltage to the gate of M_2 (I_2 remained constant throughout the experiment). Alternatively, the FG can be disconnected from the memory cell and placed into a separate circuit for a current output.

Figure 4(c) shows the results of performance experiments on the memory cell and programmer combination. A standard wide output-range transconductor was used. The memory cell was programmed to linearly spaced values of V_{targ}, and the value of V_{cg} was measured after the circuit was placed in read mode—for which the supply voltage was ramped down, the transconductor was disconnected, and a fixed current was applied for I_1. The top pane shows that the memory cell has a linear relationship between V_{targ} and the ramped down V_{cg} (with a slope of 1.003 and an offset of 122mV). The deviation from a straight line is shown in the bottom pane. For every fourth data point, the memory cell was programmed 100 times

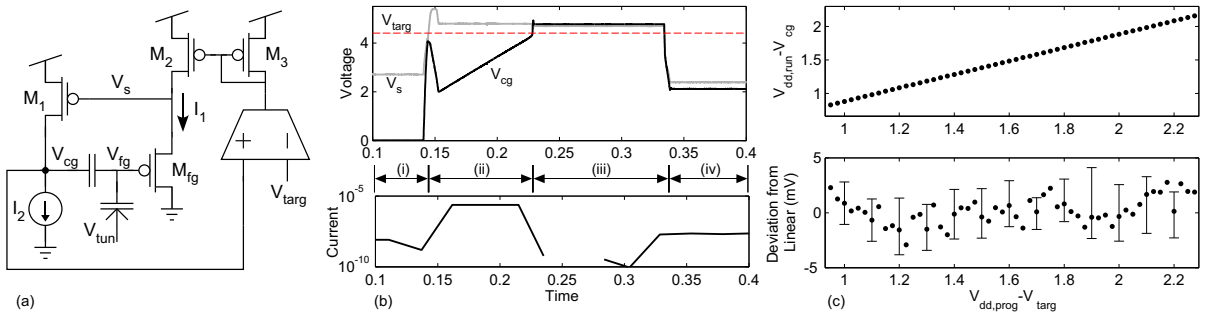

Fig. 4. (a) Our memory cell programming circuit. (b) Measured timing diagram of the programming circuit, showing the control gate, source, and target voltages as well as the source current. While V_{targ} is greater than V_{cg}, the floating-gate injects and V_{cg} rises. As the target is approached, I_1 is reduced, slowing injection until the target is reached, at which point the current is shut off. Afterwards, the supply voltage is reduced to a run-mode level, a constant current is applied for I_1 in order to bias the current conveyor, and the programmed value can then be read from V_{cg}. (c) Measured programming accuracy.

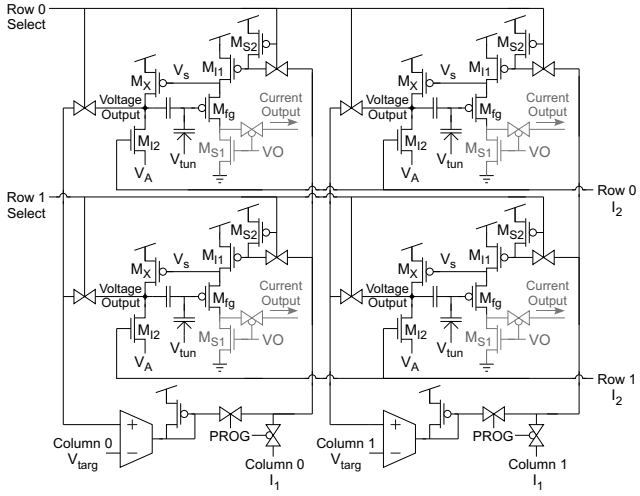

Fig. 5. Array architecture for the memory cell and programmer.

in order to verify repeatability; from these data are derived the error bars which show maximum and minimum values. Over a range of 1.36V, the maximum deviation is 4.2mV, thus yielding an accuracy of over 8-bits. Note that this prototype was built on a breadboard with discrete components, and for the low current levels that were used, a noise floor on the order of millivolts was observed.

The current values used for run mode were I_1=20nA and I_2=2nA, yielding a power consumption of 66nW/cell when configured as a voltage reference. These currents do not need to be exact and do not require precise matching across cells—but the currents should be stable. During programming, the transconductor bias is set to 2μA, and the maximum program power consumption is 43μW/cell.

Since an advantage of FGs is that they allow for dense analog memory arrays, an FG memory cell should be suitable for placement within an array. In Fig. 5, we show a two-by-two array of our memory cell. In order to save space, the program control circuit is shared amongst the cells in a column; this facilitates simultaneous programming of all cells in a row. The procedure for programming row 0 is as follows. The programming circuits are connected to the array by setting PROG to high; voltage output is selected by setting VO to high; and row 0 is selected by setting "Row 0 Select" to high, thus connecting row 0's voltage output and control input (gate of M_{I1}) to the column programming circuit. For the unselected rows, the gate of M_{I1} is pulled to V_{dd} to prevent injection. In run mode, the memory cell can be used either with voltage outputs or with optional current outputs. The current outputs are accessed by pulling VO low so that the drain of M_{FG} goes out through the current output and also by pulling the source of M_{FG} up to V_{dd} by lowering the Column I_1 lines. With V_s high, M_X is off and the control gates of the cells are connected to V_A by raising the I_2 biases.

VI. Conclusion

We have presented a compact analog FG memory cell. We have also presented a continuous-time programmer for the cell which achieves 8-bit accuracy with 100ms program times. This circuit was prototyped on a breadboard, and better performance is expected for an integrated implementation.

References

[1] P. Hasler, B. Minch, and C. Diorio, "Floating-gate devices: They are not just for digital memories anymore," in *Proc. IEEE Int. Symp. Circuits Syst.*, 1999, pp. 388–391.

[2] C. Huang, P. Sarkar, and S. Chakrabartty, "Rail-to-rail, linear hot-electron injection programming of floating-gate voltage bias generators at 13-bit resolution," *IEEE J. Solid-State Circuits*, vol. 46, no. 11, pp. 2685–2692, Nov. 2011.

[3] C. Diorio, "A p-channel MOS synapse transistor with self-convergent memory writes," *IEEE Trans. Electron Dev.*, vol. 47, no. 2, pp. 464–472, Feb. 2000.

[4] A. Bandyopadhyay, G. Serrano, and P. Hasler, "Adaptive algorithm using hot-electron injection for programming analog computational memory elements within 0.2% of accuracy over 3.5 decades," *IEEE J. Solid-State Circuits*, vol. 41, no. 9, pp. 2107–2114, Sept. 2006.

[5] C. Diorio, S. Mahajan, P. Hasler, B. Minch, and C. Mead, "A high-resolution nonvolatile analog memory cell," in *Proc. IEEE Int. Symp. Circuits Syst.*, vol. 3, Seattle, WA, April 1995, pp. 2233–2236.

[6] K.-H. Kim, K. Lee, T.-S. Jung, and K.-D. Suh, "An 8-bit-resolution, 360-μs write time nonvolatile analog memory based on differentially balanced constant-tunneling-current scheme (DBCS)," *IEEE J. Solid-State Circuits*, vol. 33, no. 11, pp. 1758–1762, Nov. 1998.

[7] H. Román and G. Serrano, "A system architecture for automated charge modifications of analog memories," in *IEEE Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 1069–1072.

[8] Y.-D. Wu, K.-C. Cheng, C.-C. Lu, and H. Chen, "Embedded analog nonvolatile memory with bidirectional and linear programmability," *IEEE Trans. Circuits Syst. II*, vol. 59, no. 2, pp. 88–92, Feb. 2012.

[9] A. Andreou, K. Boahen, P. Pouliquen, A. Pavasović, R. Jenkins, and K. Strohbehn, "Current-mode subthreshold MOS circuits for analog VLSI neural systems," *IEEE Trans. Neural Networks*, vol. 2, no. 2, pp. 205–213, March 1991.