

ACCURATE PROGRAMMING OF ANALOG FLOATING-GATE ARRAYS

Paul D. Smith, Matt Kucic, Paul Hasler *

Georgia Institute of Technology
Department of Electrical and Computer Engineering
Atlanta, GA 30332

ABSTRACT

This paper presents an accurate method for programming analog values into an array of floating-gate pFETs. Channel hot-electron (CHE) injection is used to program the devices and the programming algorithm is based upon a representation of the physical injection equation. The programming accuracy is dependent upon the physical parameters fitting of this equation and the algorithm updates these physical parameters to correspond with the current region of operation. The control circuits used to access each element and program the array are contained on a custom programming board which interfaces to a computer via a serial connection.

The use of floating-gate devices as a memory element is not new in the circuits community [1]. Since their discovery much effort has gone into making them a viable non-volatile memory element which we now find in digital cameras and mp3 players. More recently the beneficial use of floating-gates in analog circuits has been realized and it has been published that not only can they be used as memory devices but function as programmable, compact, computational elements [2]. Floating-gate devices have been recently touted as almost a magical elements in analog circuits, storing analog values [3], and performing computations [4].

Floating-gate analog circuits are currently capable of performing various computational tasks such as multiplication [5], filtering [6] and automatic gain control [7]. The floating-gates in each of these circuits store an analog value as well as perform the analog computation. These implementations are highly dependent upon the ability to accurately programming multiple floating-gate elements. This approach to floating-gate programming is extendable to a wide range of circuits. Two examples applications include: the C^4 programmable bandpass filters [6], and diff-pair mismatch cancellation.

The floating-gate core used in the computational arrays differ from those used in other analog memory circuits such

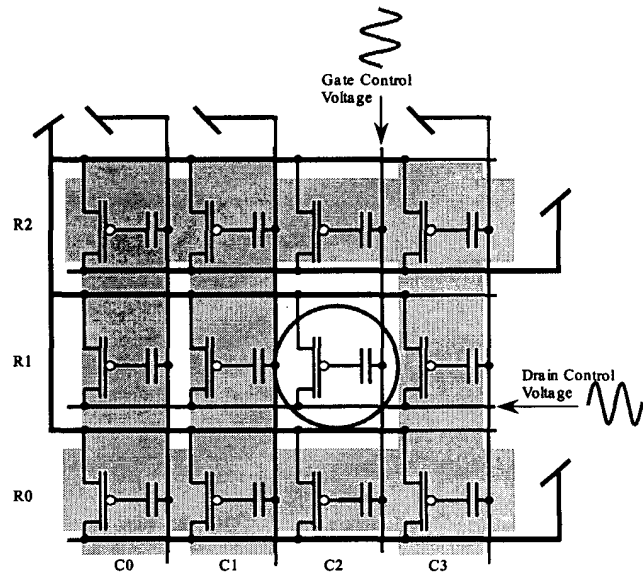


Fig. 1. Floating-gate array demonstrating element isolation by controlling the gate and drain voltage of each column and row. Selection of gate and drain voltages are controlled by on-chip mux circuitry.

as Epots [3]. Epots are made "User-Friendly" by the addition of several control circuits around each floating-gate element such that the overall circuit block is one order of magnitude larger than our floating-gate elements. The benefit of large floating-gate computational arrays is the compactness of each core element and therefore support circuits are moved to the arrays periphery.

As with any new technology, this device presents new challenges that must be addressed before it can be a viable solution to mainstream applications. Many of these headaches are very similar to the digital counterparts, EEPROMs [8]. The major similarity is how to modify the information in the device to a desired value. Yet, while similar in physics, the desired application and use of the device lead us to different schemes to alter the stored information. The goal of this research is to deliver a system that will allow anyone to realize the benefits of floating-gates in analog

*This work was partially supported by grants National Science Foundation (CISE-1068549, ECS (CAREER): 0093915, ECS-9988905) and by corporate donations to the Georgia Tech Analog Consortium by Texas Instruments and Motorola, Inc.

systems, even without a deep understanding of floating-gate devices.

We present a custom programming board and a physics based algorithm that is able to quickly and accurately program the floating gates in large arrays. Hopefully the user friendly analog programming scheme places analog floating-gates in the cookbook of circuit designers. Eventually making them as easy to understand and easy to use as digital EEPROMs which is currently an accepted technology in modern circuit design [4].

1. FLOATING-GATE STRUCTURE AND PROGRAMMING SCHEME

The programming scheme for analog cells must be different than that of digital memory due to the difference in their objectives. For digital or binary circuits the desired solution is only interested in resolving two distinct values for each cell, that is one or zero; newer digital systems store information using multiple levels but fundamentally use approaches similar to binary valued cells. Analog programming systems will require a continuous range of programmable values. Also in digital systems, design considerations demand a device that can continuously and quickly store and read information. In an analog system we are generally more interested in programming a device once and then reading or, as we use them, computing continuously afterwards. Therefore in the analog system, instantaneous programming of each cell is not as critical while accuracy of programming is far more critical. Speed is still relatively important as large arrays are to be programmed, but our approach prioritizes accuracy considerations over speed as evident by the iterative approach.

Programming a floating-gate element involves being able to adjust multiple control voltages of a single element. As shown in Fig. 1 it is possible to isolate individual elements in a large matrix using peripheral control circuitry. Using the control circuitry, one is able to access and isolate the gate and drain voltage of a single element. This orthogonal isolation allows a single device to be programmed through correctly applied control voltages.

Any circuit containing programmable floating-gate elements must also have various switching circuitry to access each floating-gate element. The programming method uses gate isolation per column and drain isolation per row. Individual circuit blocks are designed assuming access by rows and columns and the control circuitry is designed accordingly. This method of device isolation ensures that 1.) there will only be sufficient drain to source voltage for injection of elements in a given row, and 2.) the only element in that given row with any current flowing will be in the selected column. Row selection switches the drain lines of a row of circuit blocks to the appropriate control signal while all

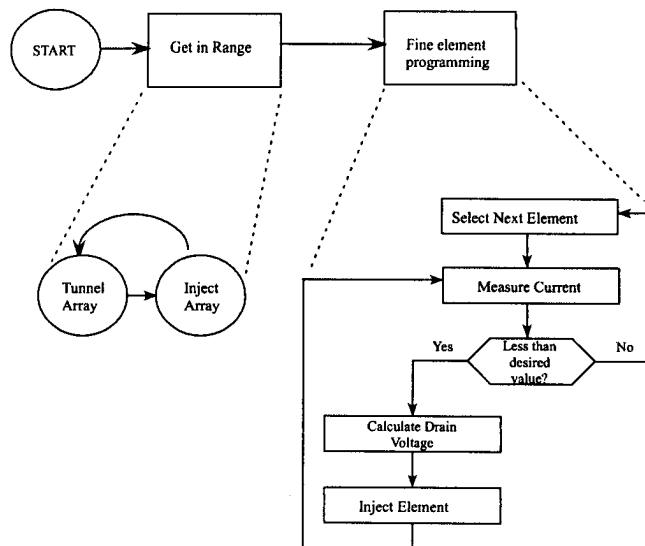


Fig. 2. Flow chart of programming algorithm.

other drains are connected to a different control line which is typically connected to VDD for the chip. Column selection switches the gate lines of an entire column to a single control line while all other gates are switched to another line which is typically connected to VDD. This configuration ensures that control voltages for the selected element are available externally for injection control, while at the same time elements not being programmed are off during the process. Because of the simplicity of the switching scheme, which requires a single pFET in most cases, or a complementary set of full transmission gates in the most complex case, we are able to minimize the overall size of individual circuit blocks.

The isolation circuitry is made of muxes that switch the drain and gate voltages of the desired element onto a common bus for each signal. All other elements are switched to a separate voltage which ensures that those devices will not inject. The external voltages are routed off-chip and controlled by an external programming board[2]. A typical programming scenario is shown in Fig. 1.

2. HARDWARE

We designed a custom floating-gate programming board to control the floating-gate programming scheme. The board is controlled by a serial port through MATLAB. The board controls the power supply, gate voltage, tunnelling voltage, and drain voltage. The board is also able to measure current while varying the drain voltage, which is essential during injection.

Part of the system involved accessing the array is on chip with the devices. Future revisions of this system are being

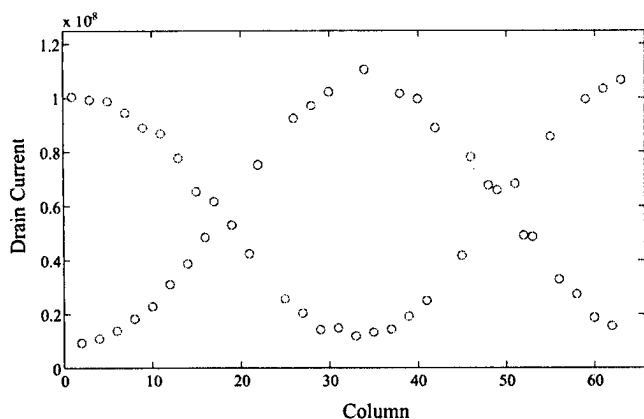


Fig. 3. A single row of floating-gate multiplier blocks programmed to cosine coefficients. These blocks are essential to performing analog frequency transform functions. Because the values are arbitrary, one can also set these linearly or to increase or decrease logarithmically.

developed that move more of the external circuitry on chip. This will provide cleaner data and decrease programming speed. Speed improvements are also possible by utilizing parallelism that can only be obtained, at reasonable costs of time, money, and overhead, by moving the control on chip.

3. ALGORITHM

Using large floating-gate arrays on the order of 1K to 10K elements, it becomes obvious that programming each element by hand would be a very time intensive process. Fig. 2 shows a block diagram of the programming algorithm. The programming algorithm has been automated into a single MATLAB function comprised of three lower level functions. These functions include:

1. TunnelToRange()

Tunnels entire array until all current levels are below their desired values. This function is required because tunnelling is used for global erase and injection is used for fine programming but only works in a single direction.

2. InjectToRange()

Ensures that all elements have a sufficient current at a given gate voltage to increase the speed of injection. Injection requires drain current. Tunnelling is a global function so some elements may have significantly lower current levels than others. Future revisions will allow isolation of tunnelling as well.

3. InjectArray()

Controls the injection of an entire array by calculating the optimal drain voltage to ensure that each injection

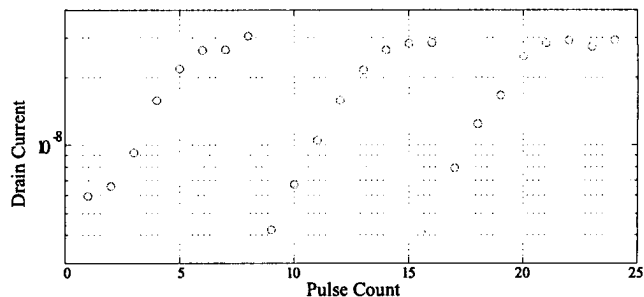


Fig. 4. Injection of a single element over 3 different trials. Each trial has a desired current of 30 nA. After the desired current is reached, the gate voltage is increased to lower the current and the programming algorithm is repeated.

pulse brings the element closer to the desired current without overshooting.

Based on transistor physics we are able to solve for the injection drain-to-source voltage for a desired current. Hot-electron injection in a pFET is described by (1)

$$I_{inj} = I_{inj0} \left(\frac{I_S}{I_{S0}} \right)^\alpha e^{-\Delta V_d / V_{inj}} \quad (1)$$

where I_{S0} is the initial current, I_S is the final current, and V_{inj} and I_{inj} are physical device parameters. The injection current is also dependent upon the change in the floating-gate voltage V_{fg} [9]. Solving for I_S we get

$$I_S^{-\alpha} = \frac{-\alpha \cdot I_{inj0}}{C_0 \cdot I_{S0}^\alpha} \cdot e^{-\Delta V_d / V_{inj}} \cdot (t) + I_{S0}^{-\alpha} \quad (2)$$

where $C_0 = U_T \cdot C_T / \kappa$ [5, 10].

Rearranging this solution into a single equation for ΔV_d , these equations are simplified into a solution of a single equation for the required drain voltage to reach a desired current. A change in drain voltage is calculated around a quiescent drain voltage that gives a 10 percent change in current for each injection pulse.

$$\Delta V_d = -\ln \left[\frac{-C_0}{\alpha \cdot I_{inj0} \cdot (t)} \left[\left(\frac{I_{S0}}{I_{S_{desired}}} \right)^\alpha - 1 \right] \right] \cdot V_{inj} \quad (3)$$

The required drain voltage is calculated for each element in the array given their present value of current and the final desired current. These voltage values are then applied to each element in the array [5].

Using floating-gate elements in analog computation requires that each element be programmed accurately to a desired current. Fig. 3 shows a row of floating-gate multipliers that have been programmed as a differential cosine weighting function. These elements perform a scale multiplication on a differential input signal.

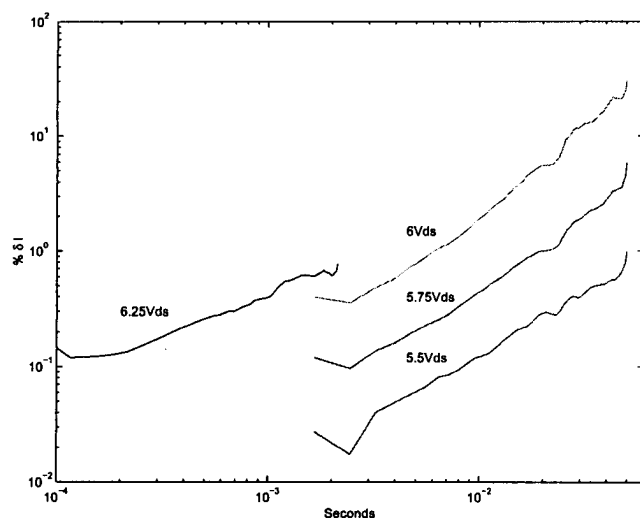


Fig. 5. Plot of injection rate versus injection pulse width for different drain-to-source voltages. The injection pulse width was limited by programming hardware and is shown to occur at a pulse width of $100\mu\text{s}$. The minimum injection pulse width can be decreased by moving the control circuitry on chip.

4. SPEED ISSUES

With large arrays of floating gates, on the order of 1K to 10K, speed is of paramount importance for this technology to be viable. Programming speed is limited by four factors:

1. Injection pulse width.
2. External current measurement.
3. Serial communications for element selection.
4. Parallelism exploited within the system.

Injection has been shown to occur with pulses down to $10\mu\text{s}$. The rate of change at a given drain-to-source voltage increases with pulse width and drain-to-source voltage. Fig. 5 shows injection rates as the injection pulse width increases. Fig. 4 shows convergence in eight to ten iterations, which results in an individual programming time of 80 - $100\mu\text{s}$. For an array with 2K floating-gate elements, the total programming time could be as low as 80 - 100 ms.

External current measurements are limited due to the huge line capacitance for wires running off chip and the equipment used to perform the current measurement. The current measurement circuitry can typically provide one current measurement between $10\mu\text{s}$ for large currents and 100ms for very small currents. Off chip also requires additional filtering of the data to ensure accurate results which can make the measurements even slower. Future versions will have on-chip current measurement circuitry.

Programming board serial port communication is limited by port speed and also the operating system running

the algorithm. Using windows machines has shown to add additional serial port timeout delays. We are currently moving the algorithm onto a programmable integrated circuit controller (PIC) or onto a field-programmable gate array (FPGA), thereby removing the OS delays.

Currently, chips isolate individual elements but future versions will select multiple elements in parallel. Selecting and programming multiple elements will become possible as more of the control circuitry is moved on-chip. Programming an entire row or column at a time will significantly reduce overall programming time.

5. REFERENCES

- [1] D.Kahng and S.M. Sze, "A floating gate and its application to memory devices," *Bell Syst. Tech. J.*, vol. 46, pp. 1288, 1967.
- [2] M. Kucic, J. Dugger, P. Hasler, and D. Anderson, "Programmable and adaptive analog filters using arrays of floating-gate circuits," in *Proceedings of the 21st Conference on Advanced Research in VLSI*, Atlanta, GA, March 2001, pp. 148 -162.
- [3] R.R. Harrison, J.A. Bragg, P. Hasler, B.A. Minch, , and S.P. Deweerth, "A cmos programmable analog memory-cell array using floating-gate circuit," in *IEEE Transactions on Circuits and Systems Special Issue*, January 2001, vol. 48, p. 4.
- [4] P. Hasler, B. A. Minch, C. Diorio, and C. A. Mead, "An autozeroing floating-gate amplifier," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 2000, in Press.
- [5] 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.
- [6] Matt Kucic, AiChen Low, Paul Hasler, and Joe Neff, "Programmable continuous-time floating-gate fourier processor," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 90-99, January 2001.
- [7] Paul Hasler and Paul D. Smith, "An autozeroing floating-gate amplifier with gain adaptation," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, Orlando, 1999, vol. II, pp. 412-415.
- [8] William D. Brown and Joe E. Brewer, *Nonvolatile Semiconductor Memory Technology, A Comprehensive Guide to Understanding and Using NVSM Devices*, IEEE Press, 345 East 47th Street, New York, NY 10017-2394, 1998.
- [9] Paul Hasler, Chris Diorio, Bradley A. Minch, and Carver A. Mead, "Single transistor learning synapses," in *Advances in Neural Information Processing Systems 7*, Gerald Tesauro, David S. Touretzky, and Todd K. Leen, Eds., pp. 817-824. MIT Press, Cambridge, MA, 1995.
- [10] Paul Hasler, "Continuous-time feedback in floating-gate mos circuits," in *IEEE Journal of Circuits and Systems II*, January 2001, pp. 56-64.