

A Programmable Continuous-Time Floating-Gate Fourier Processor

Matt Kucic, AiChen Low, Paul Hasler, *Member, IEEE*, and Joe Neff

Abstract—We present a programmable continuous-time floating-gate Fourier processor that decomposes the incoming signal into frequency bands by analog bandpass filters, multiplies each channel by a nonvolatile weight, and then recombines the frequency channels. A digital signal processor would take a similar approach of computing a fast Fourier transform (FFT), multiplying the frequency components by a weight and then computing an inverse FFT. We decompose the frequency bands of the incoming signal using the transistor-only version of the autozeroing floating-gate amplifier (AFGA), also termed the capacitively coupled current conveyer (C^4). Each band decomposition is then fed through a floating-gate multiplier to perform the band weighting. Finally, the multiplier outputs are summed using Kirchoff current law to give a band-weighted output of the original signal. We examine many options to reduce second-order harmonic problems inherent in the single-sided C^4 . We present a method for programming arrays of floating-gate devices that are used in the weighting of the bands. All of these pieces fit together to form an elegant and systematic Fourier processor.

Index Terms—Analog floating-gate arrays, floating-gate circuits, programmable analog circuits, programmable analog filters.

I. INTRODUCTION

THIS paper presents a programmable continuous-time floating-gate Fourier processor. The architecture is based on a modified fast Fourier transform implementation in a digital signal processing (DSP) filter. The design is modular, allowing flexibility in the number of taps needed without major additional layout overhead. The design was chosen to allow for a modular system where addition of taps as needed results in very little additional overhead. This filter can be employed anywhere a DSP filter is used without the drawbacks associated with DSP filtering, such as aliasing. This analog filter allows the same level of programming as the DSP filter.

Fig. 1 graphically demonstrates the top-level description of our Fourier processor chip. In this figure, we show four band taps that can be expanded to as many as needed. Each tap consists of a programmable bandpass filter and a weighted multiplier. The input signal is taken as a voltage to allow the signal to be broadcasted to the multiple bandpass filters or band taps. The output of each bandpass filter is also a voltage, which can be broadcasted to several weighted multiplier arrays. Therefore, with one processor, we can output multiple band-weighted ver-

sions of the original signal. The output of each weighted multiplier is a current that allows simple addition via Kirchoff current law to assemble the final output signal.

The multiple bandpass filters produce a frequency decomposition of the incoming signal into multiple bands. We use a transistor-only circuit model of the autozeroing floating-gate amplifier (AFGA) [7], which we will refer to as the capacitively coupled current conveyer (C^4) [2], to achieve a broadly tuned bandpass response. By adding feedback between the stages, we sharpen the filter rolloff response if desired. This approach is another method for getting cochlea-like responses, but it is missing many of the details that are present in the biologically based detailed models [5], [6]. On the other hand, this implementation does not suffer from the noise accumulation and harmonic distortion accumulation typically seen in cochlear cascades [5].

The weighting is performed using floating-gate transistors [1] in a multiplier configuration. The benefit to using a floating gate for the weighting is small size and circuit simplicity. Also, by using floating-gate transistors for the weighting, we are using the actually memory element as part of the computation, which provides for an extremely high chip density. Therefore, in the circuit complexity and power dissipation that one would store an array of 4-bit digital coefficients, we simultaneously compute a parallel vector multiplication with this matrix. We call this technology, which combines computation and memory, analog computing arrays (ACAs). Further, this system only needs to operate at the incoming data speed. This density is desired to realize chips with large number of band taps or chips with several band-weighted outputs. This memory element retains its value even when power is not applied to the device, and eliminates the need for separate nonvolatile memory cells with analog-to-digital and digital-to-analog circuitry to store and reproduce the actual analog weight.

We present analysis and experimental measurements of this analog Fourier processor chip fabricated in a double-poly 2.0- μ m MOSIS process; we have shown that these computations can be performed also in single-poly microprocessor process [4]. We have fabricated these programmable filters in 2.0-, 1.2-, 0.5-, and 0.25- μ m MOSIS processes. Section II describes the circuits for the continuous-time bandpass filters. These circuits are a transistor-only circuit model of the AFGA termed C^4 that we initially described elsewhere [2]. Section III describes the floating-gate circuits for the weighting multipliers. Section IV describes our weight programming scheme. Section V demonstrates the performance of the complete Fourier processor. This processor is a first step to developing all-analog floating-gate adaptive filters.

Manuscript received April 2000; revised November 2000. This paper was recommended by Associate Editor T. Lande.

The authors are with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332-0250 USA (e-mail: phasler@ee.gatech.edu).

Publisher Item Identifier S 1057-7130(01)02023-7.

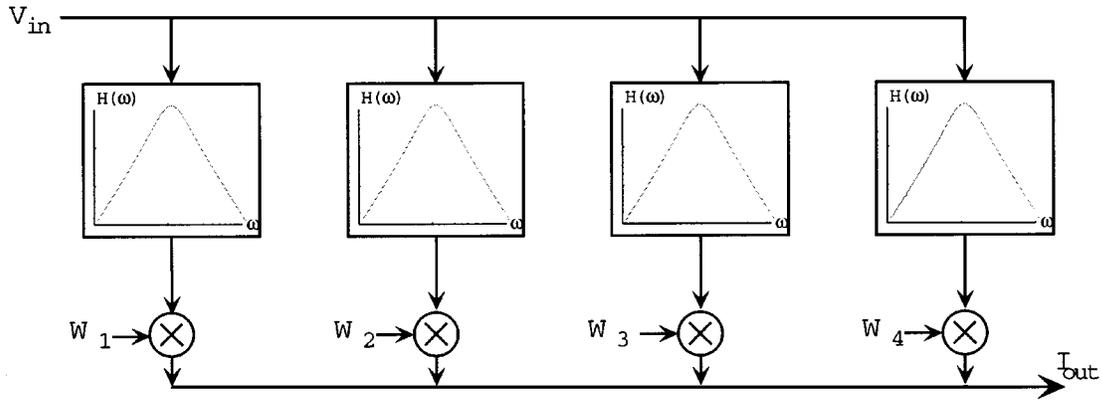


Fig. 1. Top-level pictorial representation of our programmable analog Fourier processor. We separate the signal into frequency bands not by computing a DFT algorithm but by a series of bandpass filters. We can easily divide our frequency space exponentially instead of linearly, as in typical DFT algorithm.

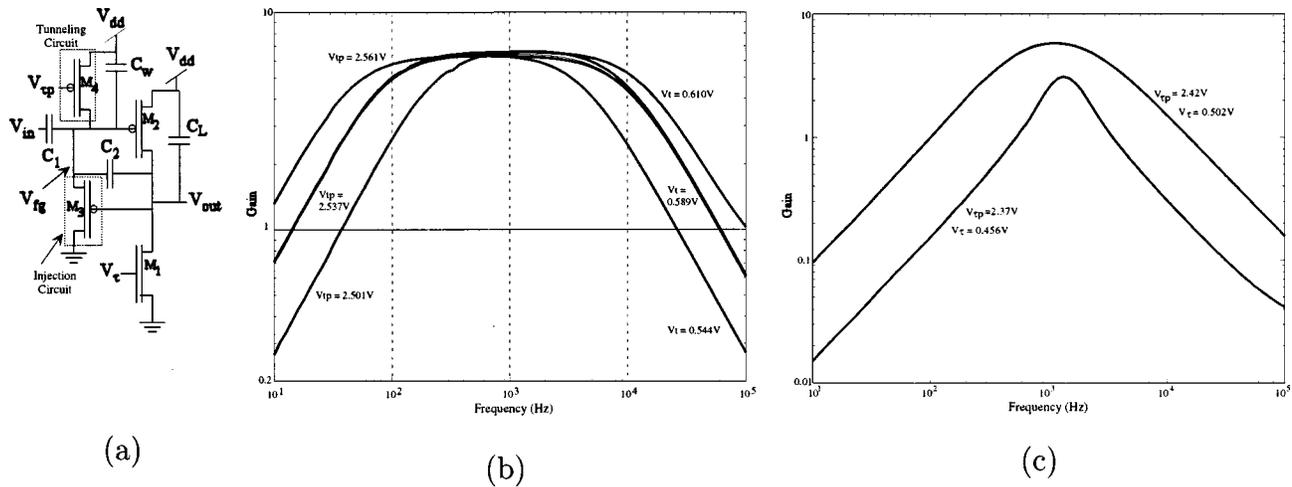


Fig. 2. A single-ended version of our continuous-time bandpass amplifier. This circuit is an all-transistor equivalent of the AFGA. (a) The all-transistor circuit version of the AFGA. The ratio of C_2 to C_1 sets the gain of both inverting amplifiers. The capacitances C_w and C_L represent both the parasitic and the explicitly drawn capacitances. M4 represents the tunneling junction in the AFGA, and M3 represents the injection current (gate current) from M2 in the AFGA. The nFET is a current source that sets the current through the pFET. (b) Frequency response for three different values of V_{τ} and three different values of $V_{\tau p}$. We can independently change the high-frequency corner with V_{τ} and the low-frequency corner with $V_{\tau p}$. The passband gain of this circuit is roughly 6.4. (c) Frequency response of our bandpass circuit when τ_i is near τ_h . When symmetrically decreasing τ_i and τ_h , the center frequency remains nearly unchanged but the bandpass gain decreases. This type of response is our focus in this paper.

II. A DIFFERENTIAL CONTINUOUS-TIME BANDPASS AMPLIFIER

The first element in our Fourier processor is a group of bandpass amplifiers. In addition to band selectivity, the choice of the bandpass amplifier design must take into account die area and quiescent current, as many will be needed on one chip. Certain applications for this processor such as in auditory systems will require 100–200 of these band taps. Because we are designing an on-chip bandpass filter, we are not constrained to building op-amp circuits with feedback. These bandpass amplifiers should also have well-controlled gains, which is easily achieved by capacitive feedback and is ideal for small size.

Fig. 2 shows a single-ended version of our bandpass amplifier. This continuous-time filter allows both the low-frequency and high-frequency cutoffs to be controlled electronically by changing the appropriate bias currents. We have derived analytical models that are in good agreement with experimental data to completely characterize the amplifier [2] and have simulated these circuits using Cadence [3]. We present experimental mea-

surements that were taken using 3.0-V power supplies in this section.

We will model voltage and current swings around the circuit's steady-state values, because the input signals are capacitively coupled to the gate terminals. We describe the subthreshold nFET or pFET channel current in saturation I_s for a change in the field-effect transistor (FET)'s gate voltage ΔV_g and drain-to-source voltage ΔV_{ds} , around a bias current I_{so} as [6]

$$\begin{aligned} \text{nFET: } I_s &= I_{so} \exp\left(\frac{\kappa_n \Delta V_g - \Delta V_s}{U_T}\right) \exp\left(\frac{\Delta V_{ds}}{V_A}\right) \\ \text{pFET: } I_s &= I_{so} \exp\left(\frac{\Delta V_s - \kappa_p \Delta V_g}{U_T}\right) \exp\left(-\frac{\Delta V_{ds}}{V_A}\right) \end{aligned} \quad (1)$$

where

κ_p fractional change in the pFET surface potential due to a change in ΔV_g ;

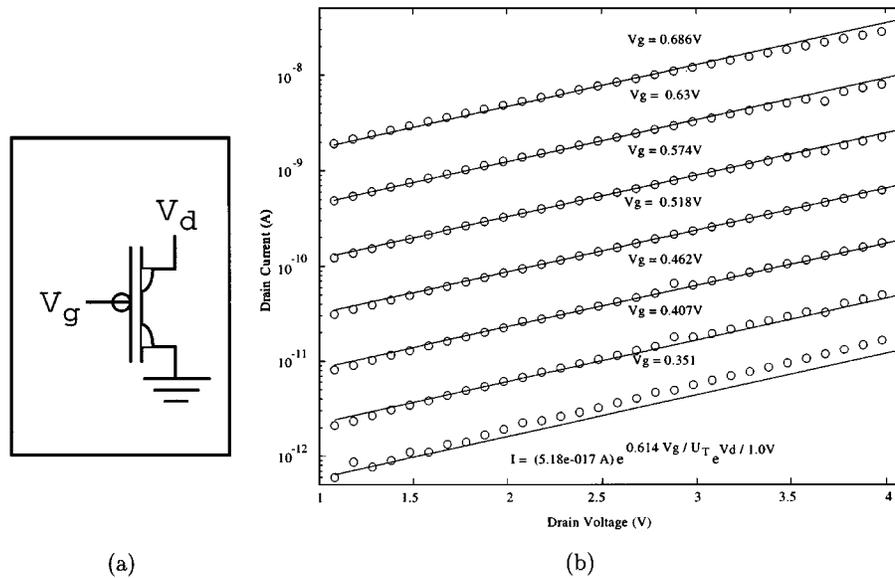


Fig. 3. Circuit diagram and experimental data from a DIBL pFET. (a) Circuit diagram of this DIBL pFET. We build a DIBL FET from a traditional FET by using a very short channel length for a given process. (b) These data were taken from an $L = 1.75 \mu\text{m}$ pFET built in a $2\text{-}\mu\text{m}$ CMOS process. The DIBL FET results in an exponentially increasing current due to linear changes in drain voltages for subthreshold biases; we see a similar effect for above-threshold currents. Good agreement is seen to a single curve fit of this data to (1), with $\kappa = 0.614$ and $V_A = 1 \text{ V}$. We can use this device to increase the linear range of on-chip FET circuits without resistors.

κ_n fractional change in the nFET surface potential due to a change in ΔV_g ;

V_A Early voltage of the nFET or pFET;

U_T thermal voltage kT/q .

In (1), we use a modified form of the Early voltage expression that is consistent with classical formulations for large V_A and more closely models the behavior for small V_A . Fig. 3(b) shows that the channel current through this pFET is an exponential function of both gate and drain voltage for a very short channel-length device. We call a device that exhibits this exponential relationship between channel current and drain voltage a drain-induced barrier lowering (DIBL) FET. The symbol used for the DIBL FET is shown in Fig. 3(a).

Fig. 2(b) shows this circuit's bandpass transfer function. The nFET current source **M1** sets the amplifier's bias current and therefore the resulting high corner frequency. The pFET current source **M4** sets the bias current and the resulting low corner frequency. Fig. 2(b) shows the measured pAFGA frequency response for various inputs and V_τ , $V_{\tau p}$ bias voltages. We can obtain the following transfer function [2]:

$$\frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = -\frac{C_1}{C_2} \frac{1 - A_h \tau_h s}{1 + \tau_l s + \frac{1}{\tau_l s}} \quad (2)$$

where

τ_l high-pass corner that is a function of the bias current flowing through **M4** and C_2 ;

τ_h low-pass corner that is a function of the bias current flowing through **M1** and all the circuit's capacitors;

A_h high-frequency feedthrough gain.

We can simplify (2) when $\tau_l \gg \tau_h$, that is, when the time constants are sufficiently separated to form an amplifier region.

In this regime, V_τ and $V_{\tau p}$ independently alter the corner frequencies [2]. At low frequencies, the circuit in Fig. 2 behaves as a high-pass filter. We approximate (2) as

$$\frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = -\frac{C_1}{C_2} \frac{s \tau_l}{1 + s \tau_l}. \quad (3)$$

The corner frequency is set by $V_{\tau p}$. At high frequencies, the circuit in Fig. 2 behaves as a low-pass filter. In this case, we approximate (2) as

$$\frac{V_{\text{out}}(s)}{V_{\text{in}}(s)} = -\frac{C_1}{C_2} \frac{1 - A_h \tau_h s}{1 + \tau_h s}. \quad (4)$$

At frequencies much higher than the integrating regime, this circuit exhibits capacitive feedthrough, which can be reduced by an increase in either C_w or C_L . The circuit in Fig. 2(a) can also operate as a bandpass filter with a narrow passband, that is, V_τ and $V_{\tau p}$ can affect the entire transfer function. Fig. 2(c) shows the frequency response for two values τ_l and τ_h that are close together; this experiment shows this bandpass behavior.

This bandpass circuit was originally developed as an transistor-only version of our AFGA [7], [1]. We use one subthreshold transistor to model the behavior of an electron-tunneling device and another to model the behavior of pFET hot-electron injection. This circuit behaves similarly to the AFGA with different operating parameters. This filter has a low-frequency cutoff at frequencies above 1 Hz, and therefore complements the operating regimes of the AFGA ($<1 \text{ Hz}$ cutoff). The close connection to the AFGA allows for direct applications of existing results.

- 1) We can increase the filter's linear (minimum) range by increasing C_w .
- 2) A voltage input at the filter's linear range corresponds to -26 dB second-harmonic dominated distortion.

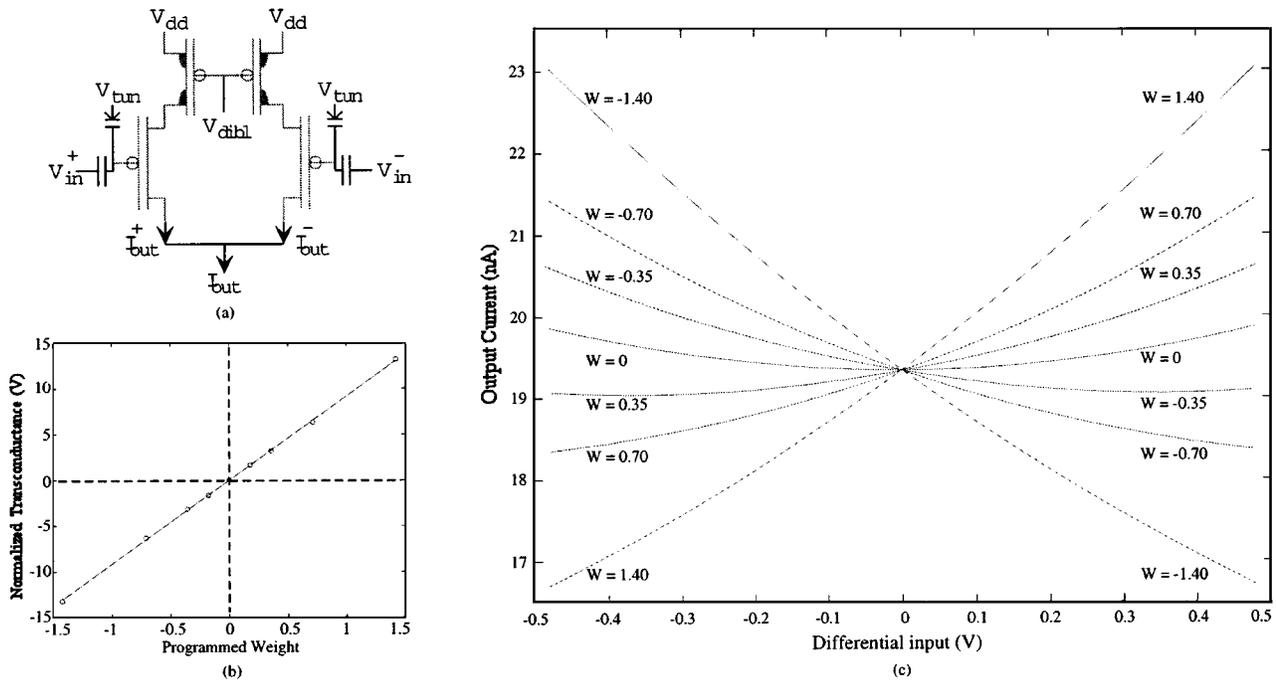


Fig. 5. Four-quadrant weighted multiplication using floating-gate devices. Shown are experimental data of the transfer characteristics of these devices. Between V_{tun} and V_{fg} is our symbol for a tunneling junction, which is a capacitor between the floating gate and an n-well.

is a four-quadrant product of the input by the synapse weights for fast timescales.

IV. FLOATING-GATE PROGRAMMING

We present a systematic method for programming an array of floating-gate devices, which are a critical part of this single-chip system. The e-pot (electronically programmable voltage source) approach is explicitly designed to program single voltage biases in a user-friendly manner [14], [15]. However, these cells require significant circuit complexity and therefore are inefficient for large floating-gate arrays. Instead, we will trade off user-friendly but complex circuits for simpler circuits programmed by well-controlled computer algorithms that can be implemented in industrial testers. We also desire a programming algorithm that is based on output values actually used during operation; therefore, compensation circuitry is not needed to produce a smooth transition from operation to program mode. As a result, we have developed a programming scheme where we perform injection over a fixed time window based on injection physics, and then measure the results after returning the cell in operating mode. Fig. 6 shows the control logic we use to automate the programming of an array of floating-gate devices. Once programmed, these floating-gate devices retain their channel current in a nonvolatile manner like the e-pot circuits.

A. Physics of Programming pFET Floating-Gate Devices

Developing an efficient algorithm for pFET programming requires discussing the dependencies of the gate currents and the ability to modify a single device with high selectivity. We program a device by increasing the output current of a pMOS transistor using hot-electron injection and decrease the output cur-

rent using electron tunneling. In our scheme, devices are programmed using hot-electron injection and are reset by tunneling the devices below the level to which they are to be programmed. This is chosen due to device selectivity for each method, which will be described. We use a time response of 0.5 s or greater because of the instruments needed to carefully characterize this measurement. The floating-gate devices could easily handle responses in 1 ms by simply increasing the tunneling and drain-to-source voltages used during programming.

Before deciding on a particular programming scheme, we first considered how the synapses (floating-gate pFET with DIBL device) interact when coupled into an array. The device interactions are due entirely to the nonlinear dependence of the terminal voltages on the floating-gate current. We choose the tunneling and drain terminals to be common along a row; therefore, when programming one row, the other rows remain unaffected. We need to establish how to selectively modify the charge on a particular floating gate without affecting the other floating gates along the same row.

Fig. 7 illustrates how we can program a single floating-gate device along a row; we originally showed the selectivity data on nFET floating-gate devices [16]. The change in source current in the selected synapse is much less than the corresponding change in the nonselected synapse, and is nearly independent of drain voltage. Tunneling selectivity along a row in this array is entirely a function of how far apart the two floating gates are pushed by the gate inputs. This is due to the fact that the amount of tunneled current is based on the voltage across the tunneling capacitor. We obtain exponentially more tunneling current for each linear increase across the capacitor due to the probability of electrons' tunneling through the barrier. To select a particular synapse, we bring that floating gate to as low a voltage as possible, while at the same time bringing all the remaining floating gates to as high

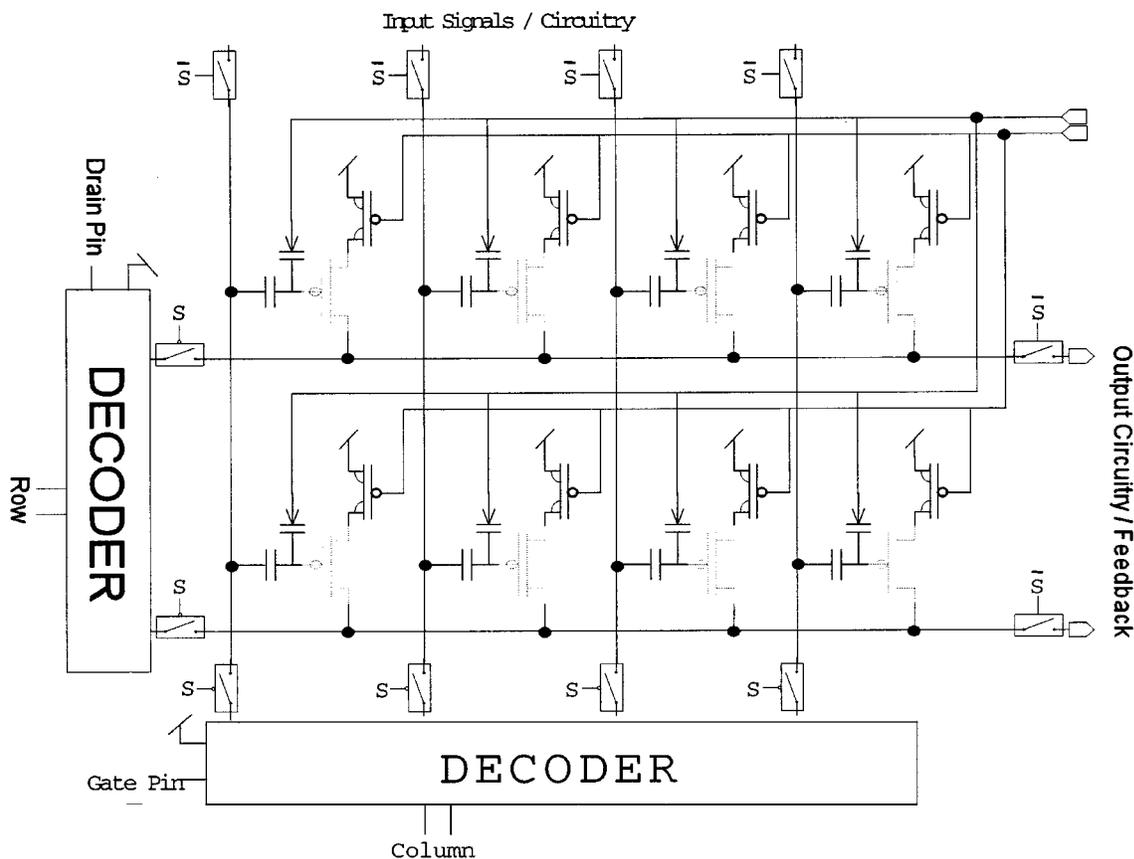


Fig. 6. Circuit diagram of chip design to allow dual programming/operation. When the control signal S is one, then we close the switches to the decoder circuitry, enabling programming, and open the switches to the normal operating circuitry. Both decoders either set their outputs to V_{dd} if zero or select an output to an external pin. When the control signal S is zero, then we open the switches to the decoder circuitry and close the switches to the normal operating circuitry.

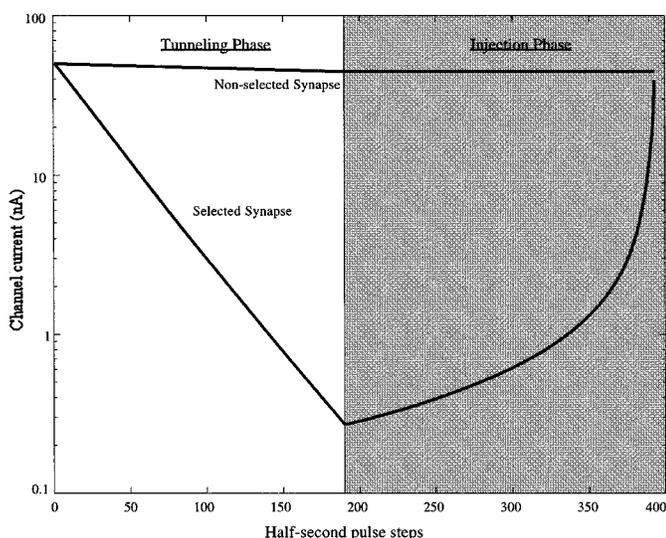


Fig. 7. Output currents from two elements on the same row of a floating-gate pFET array, showing 115 tunneling operations followed by 200 injection operations. We modified the floating-gate charge using several 0.5-s pulses.

a voltage as possible. Our selectivity ratio for the pFETs in Fig. 7 on the same row is roughly 40 for a 5-V supply. The tunneling selectivity can be increased by increasing the input voltage steps

or by increasing the gate coupling to the floating gate. Because of the poorer selectivity, we use tunneling primarily for erasing and for rough programming steps.

We simplified our initial circuit by tying all tunneling rows together to consume only one pin on the package. This approach avoids the need for high-voltage transistor switches to tunnel along an individual row. Using this simplified approach, we can only select down to a column of synapses to be modified via tunneling because in our implementation, the gates are tied together in columns. However, this approach works successfully in our scheme because tunneling is used primarily for erasing. In future revisions, we will add row decoding to tunneling, which will add an additional level of programming control.

Fig. 8 shows experimental measurements of pFET injection versus drain-to-channel voltage. Injection current in the transistor is modeled by

$$I_{inj} = I_{inj0} \left(\frac{I}{I_{so}} \right)^\alpha e^{-\Delta V_d / V_{inj}} \quad (8)$$

where $\alpha = 0.93$ and $V_{inj} = 400$ mV. For injection to occur in a device, there are two controlling parameters: the source-to-drain voltage to create the high field and the gate voltage to create the MOS channel. Therefore, a device in the array is selected to be programmed by lowering the column voltage, containing

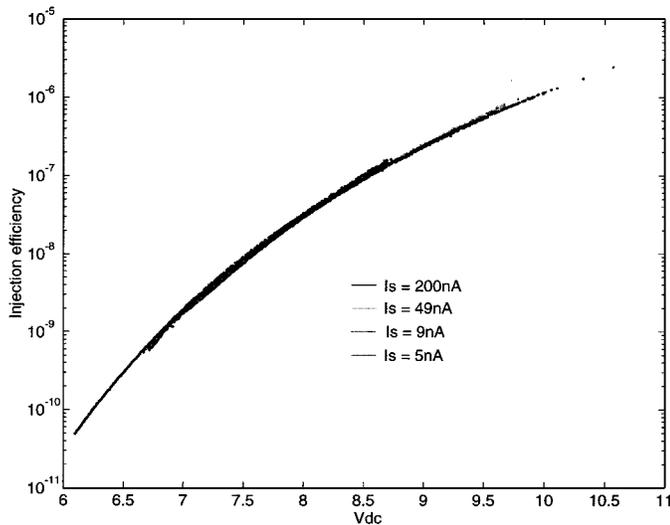


Fig. 8. Measured data of pFET injection efficiency versus the drain-to-channel voltage for four source currents. Injection efficiency is the ratio of injection current to source current. The injection efficiencies are nearly identical for the different source currents; therefore, they appear to be indistinguishable on the plot. At drain-to-channel voltages equal to 8.2 V, the injection efficiency e-folds (increases by a factor of e) for a 250-mV increase in drain-to-channel voltage.

the gate of the device, to around threshold (optimal injection voltage) and the row voltage, containing the drain of the device, to a voltage to produce injection, while all other rows and columns are tied to V_{dd} . Because both conditions described earlier must be met, we have the ability to select the device to program out of the array. For larger source-to-drain voltages, we will get exponentially more injection current, as shown in (8). To control the amount of injection in the device, the source-to-drain voltage is modulated by raising or lowering the rows' voltage. During programming, the system voltage is raised to values to allow injection in that process. All current calculations for the device are performed with the same drain-to-source voltage as during operation at the specified gate voltage. This allows the device to be programmed for its operational voltages.

B. The Programming Method

The programming of the floating gate is performed over multiple iterations. This iterative approach uses mathematical models and parameter extraction to zero in on the target current. We are interested in a change in the floating-gate voltage, which is dependent on the current being injected onto the floating node, as shown in (9). For this, we will choose a fixed time period for T , which will be used during every injection. Also, I_{inj} is assumed roughly constant with time for small changes and is pulled out as such

$$\Delta V_{fg} = \frac{1}{C} \int_0^T I_{inj} dt \rightarrow -\frac{T}{C} I_{inj}. \quad (9)$$

We solve for hot-electron injection programming using (8) as

$$\frac{I_{next}}{I_{now}} = 1 + A \left(\frac{I_{next}}{I_{so}} \right)^\alpha e^{-\Delta V_d / V_{inj}} \quad (10)$$

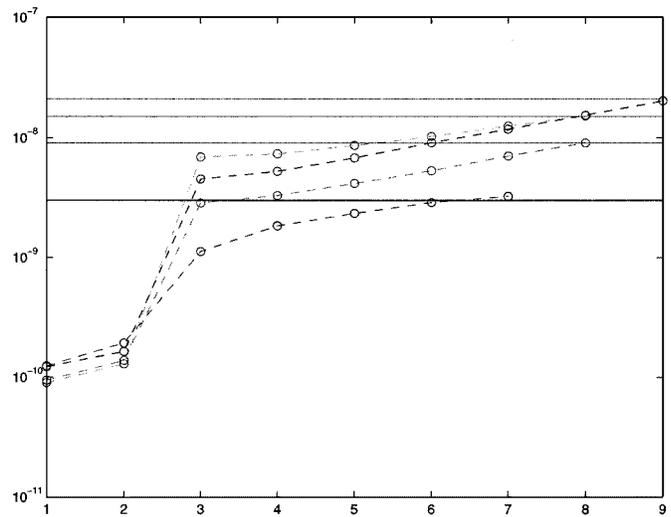


Fig. 9. Plot showing the programming of four current values. All four values converged within nine steps.

where we define $A = (\kappa T / C * U_T) * I_{inj0}$. This equation models the current that a given ΔV_d will produce during injection for T time. We have defined I as the channel current in the transistor. I_{so} is defined as the current in the device when we choose a V_{dref} , or can be thought of as where we start from (original state). I_{now} is used to define the current in the device before injection in every iterative step, or can be thought of as where we currently are. I_{next} is used to define the current that is desired in the device after the injection pulse is performed.

In the first step, we calculate the term A from (10). Also, we are attempting to find the reference V_d (V_{dref}) with this step. The process involves a loop that takes the drain voltage and slowly lowers it incrementally while measuring the before and after currents. We take the reference V_d to be the voltage at which some percent change threshold is exceed. For the experiments we have used 30%, but this is an arbitrary value. Therefore, in this first step, ΔV_d is equal to zero by choosing this voltage as a reference, making the exponential term equal to one. In the term I_{start} / I_{so} , we take the I_{so} term to be the current at V_{dref} , so the I_{start} is also equal to I_{so} . With α being close to one, we assume it to be one. The equation then reduces down to

$$\frac{I_{next}}{I_{now}} = 1 + A. \quad (11)$$

Therefore, we can calculate A by knowing the before and after current in the channel. We then choose this V_d used for injection to be our reference voltage for future injections. Also, the "before" injection current becomes our I_{so} term for the future calculations. The threshold mentioned earlier is a given value of A , which is used to bring us to a valid injection V_d for the device.

In step two, we know the A , the I_{so} , and the I_{now} terms from the previous step. We can then plug these terms into (10) with a guess of V_{inj} to calculate a ΔV_d for the next injection. This should result in a value of V_d to inject the device to I_{next} , which is the target current to program. The lower the guess of V_{inj} ,

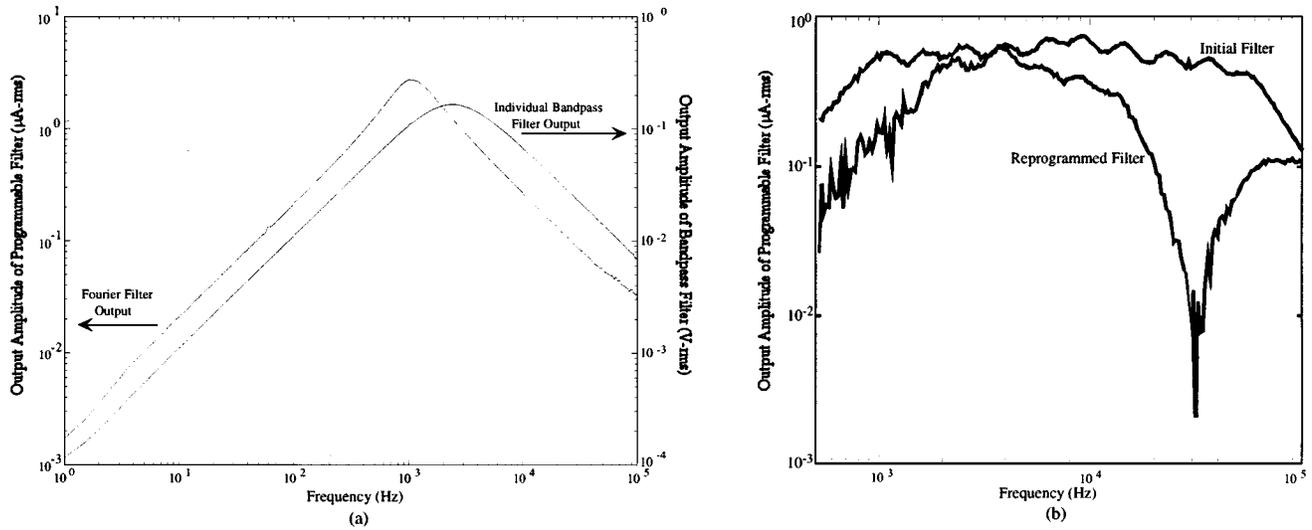


Fig. 10. Experimental measurements of frequency response of our programmable bandpass filters. (a) Frequency response for a single bandpass filter and for the array bandpass filters with programmed weighting function. We set zero tilt on the frequency line; therefore, all corner frequencies should be identical aside from mismatch. The result of this programmable filter is a tighter bandpass filter, with a corner frequency roughly half of the original corner frequency. (b) Frequency response for this programmable filter with ten bandpass elements exponentially spaced in frequency. The ripples on both curves show the location of these bandpass elements. We show an initial programmed frequency response, where the weights are nearly equal, and a second programmed frequency response to program an additional notch in the filter's response. We obtain similar frequency responses generated by our Spectre simulation model used for simulating floating-gate circuits.

TABLE I
WEIGHT VALUES USED IN PROGRAMMABLE FILTER TO OBTAIN THE RESPONSE IN FIG. 10(a)

Position	1	2	3	4	5
$W^+(\mu A)$	1.56	0.85	1.23	2.38	0.59
$W^-(\mu A)$	1.11	0.99	1.24	0.73	1.45
$W(\mu A)$	0.45	-0.14	-0.01	1.65	-0.86
Position	6	7	8	9	10
$W^+(\mu A)$	1.55	1.06	0.62	2.39	0.90
$W^-(\mu A)$	1.11	2.08	0.60	0.64	0.75
$W(\mu A)$	0.44	-1.02	0.02	1.75	0.15

the more conservative the next injection will be and the further the device will be from the target current. However, choosing a V_{inj} too high will cause the calculation to give a V_d that will overshoot the target value.

After the injection pulse in step two is performed, we can then extract the actual V_{inj} for the device. Because we know where the device ended up (I_{next}) and we knew where the device started (I_{now}), the term V_{inj} can be derived from (10). With this, we can now calculate the correct V_d to take us to the wanted programmed channel current. In the programming of the device, we backed the calculated V_d off slightly to guarantee that the target current was not overshoot. The last step is repeated until we reach the desired current in the device.

C. Experimental Data Programming for Memory Cells

The programming results from a 1.2- μm MOSIS process. A 2×4 array of floating gates was used for this experiment. The operation voltage for the chip was 3 V. For programming, 8 V was used to allow significant injection in this process to occur. During programming, the drain voltage was held at 5 V during the current measurements for system operation with a 3-V supply. The timing T used for injection was 2 s. This value was chosen only to insure no timing issues in the test environment. There is no reason that fast T values cannot be used, and future revisions of the test setup include on-board timing circuits to insure constant timing at faster speeds. These fast speeds are critical to programming mass

production or large arrays of floating gates. Fig. 9 shows an attempt at programming four devices in the array to different values.

V. THE FOURIER PROCESSOR

In this section, we show an example of this programmable filter's operation. First, we look at the frequency response of a single C^4 bandpass element, which is shown in Fig. 10(a). We obtain this bandpass response by adjusting the bias voltages V_{τ} and $V_{\tau p}$ such that the high-pass and low-pass corner frequencies are nearly equal.

Next, we set all the filter's frequency taps to identical bias voltages and program the floating-gate weights to generate an interesting bandpass response. With this topology and constant bias voltages, we can program many arbitrary filters of second order; using different topologies and using feedback connections will allow any desired filter function. Experimentally, we found a 15% difference in corner frequencies due to mismatch in the bias transistors. This current chip has a linear change in bias voltages because we connect neighboring biases with resistive connections. We programmed the weights to the pattern shown in Table I.

The floating-gate values did not change throughout our experiment. Typically, we find a small initial change in the first 24 h due to detrapping (approximately an identical 10 mV for each device), after which the gate charge remains constant over the entire duration of these experiments. Since we use differential weights, any constant detrapping will not affect the filter's transfer function.

We show the frequency response of individual bands multiplied by their weighted outputs for constant effective weights. Fig. 10(a) shows experimental measurements from our programmable Fourier filter. The result of this programmable filter is a tighter bandpass filter, with a corner frequency roughly half of the original corner frequency. The floating-gate devices used for the multiplication were built with $W/L = 10$; therefore, the devices were biases near threshold with an average bias current of $1 \mu\text{A}$ (total current was $20.58 \mu\text{A}$). We found that dynamics of the multipliers did not affect the filter transfer function, and that the harmonic distortion is limited by the multipliers, and not the bandpass filters.

Fig. 10(b) shows the frequency response for a programmable filter with ten bandpass elements that are now exponentially spaced in frequency instead of nearly identical frequencies. For the initial frequency response, we used an exponential spacing with nearly identical weight values at each tap. We can see the exponential spacing of these bandpass elements by looking at the ripples on the initial frequency response. We also show a second programmed frequency response, where we programmed an additional notch in the spectrum of this initial filter. In current versions of this programmable filter, the parameters of each bandpass element are programmable, and therefore we can build our programmable filters utilizing bandpass filters with an arbitrary spacing.

We have developed simulation models of this circuit to assist in developing future revisions of this circuit. We recently

presented a methodology for simulating floating-gate circuits in Cadence's simulator, Specter, using a modified EKV MOSFET model [3]. The frequency responses that we obtain from this simulation model agree closely with experimental results.

REFERENCES

- [1] P. Hasler, B. A. Minch, and C. Diorio, "Adaptive circuits using pFET floating-gate devices," in *Proc. 20th Anniv. Conf. Advanced Research in VLSI*, Atlanta, GA, March 1999, pp. 215–229.
- [2] P. Hasler, M. Kucic, and B. A. Minch, "A transistor-only circuit model of the autozeroing floating-gate amplifier," in *Proc. Midwest Conf. Circuits and Systems*, Las Cruces, NM, 1999.
- [3] A. Low and P. Hasler, "Cadence-based simulation of floating-gate circuits using the EKV model," in *Proc. Midwest Symp. Circuits and Systems*, Las Cruces, NM, 1999.
- [4] B. A. Minch and P. Hasler, "A floating-gate technology for digital CMOS processes," in *Proc. Int. Symp. Circuits and Systems*, Orlando, FL, 1999.
- [5] R. Sarpeshkar, R. F. Lyon, and C. Mead, "A low-power wide-linear-range transconductance amplifier," *Analog Integr. Circuits Signal Process.*, 1997.
- [6] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.
- [7] P. Hasler, B. A. Minch, and C. Diorio, "An autozeroing floating-gate bandpass filter," in *Proc. IEEE Int. Symp. Circuits and Systems*, Monterey, CA, 1998.
- [8] P. Hasler, *Foundations of Learning in Analog VLSI*. Pasadena: California Institute of Technology, Feb. 1997.
- [9] P. Hasler and J. Dugger, "Correlation learning rule in floating-gate pFET synapses," in *Proc. Int. Symp. Circuits and Systems*, Orlando, FL, 1999.
- [10] B. A. Minch, "Multiple-input translinear-element log-domain filters," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. xxx–xxx, Jan. 2001.
- [11] P. Hasler, C. Diorio, and B. A. Minch, "A four-quadrant floating-gate synapse," in *Proc. IEEE Int. Symp. Circuits and Systems*, Monterey, CA, 1998.
- [12] Y. Tsvividis, M. Banu, and J. Khaury, "Continuous-time MOSFET-C filters in VLSI," *IEEE Trans. Circuits Syst.*, vol. CAS-33, no. 2, 1986.
- [13] S. T. Dupuie and M. Ismail, "High frequency CMOS transconductors," in *Analogue IC Design: The Current-Mode Approach*, C. Toumazou, F. J. Lidgey, and D. G. Haigh, Eds. London, U.K.: Peregrinus, 1990, pp. 181–238.
- [14] R. Harrison, P. Hasler, and B. A. Minch, "Floating-gate CMOS analog memory cell array," in *Proc. Int. Symp. Circuits and Systems*, Monterey, CA, 1998.
- [15] R. R. Harrison, J. A. Bragg, P. Hasler, S. Deweerth, and B. A. Minch, "A CMOS programmable analog memory cell array using floating-gate circuits," *IEEE Trans. Circuits Syst. II*, vol. 48, no. 1, pp. xxx–xxx, 2000.
- [16] P. Hasler, C. Diorio, B. A. Minch, and C. A. Mead, "Single transistor learning synapses," in *Advances in Neural Information Processing Systems 7*, G. Tesauro, D. S. Touretzky, and T. K. Leen, Eds. Cambridge, MA: MIT Press, 1995, pp. 817–824.

Matt Kucic received the B.S. degree in electrical engineering from the Georgia Institute of Technology (Georgia Tech), Atlanta, in 1999 and the M.S. degree in electrical engineering from the Integrated Computational Electronics Laboratory of Georgia Tech in 2000. He is currently pursuing the Ph.D. degree in electrical and computer engineering from the same university.

His research interests include cooperative analog–digital signal processing, floating-gate devices, circuits, and systems, and analog signal processing for RF design.

AiChen Low received the M.S. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, in 2000.

She is currently employed by Texas Instruments, Incorporated, NH.



Paul Hasler (S'87–A'97–M'01) received the B.S.E. and M.S. degrees in electrical engineering from Arizona State University, Tempe, in 1991 and the Ph.D. degree in computation and neural systems from the California Institute of Technology, Pasadena, in 1997.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering at the Georgia Institute of Technology. His research interests include low-power electronics; mixed-signal integrated circuits and systems; the use of floating-gate MOS transistors to build adaptive information processing systems and “smart” sensor interfaces; the physics of deep submicrometer devices or floating-gate MOS devices; and analog VLSI models of neurobiological learning and sensory information processing.

Dr. Hasler received an NSF Career Award in 2001 and the IEEE Electron Devices Society's Paul Rappaport Award in 1996. He is active in the IEEE as a Cochair of the Atlanta section of the IEEE Electron Devices Society, as a Reviewer for IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS and IEEE TRANSACTIONS ON NEURAL NETWORKS, and as Cochair for special sessions in the IEEE International Symposium on Circuits and Systems in both 1998 and 1999.

Joe Neff, photograph and biography not available at the time of publication.