# A Mixed-Signal Approach to High-Performance Low-Power Linear Filters

Miguel Figueroa, *Student Member, IEEE*, David Hsu, and Chris Diorio, *Member, IEEE*

***Abstract*—We present a new approach to the design of high-performance low-power linear filters. We use p-channel synapse transistors as analog memory cells, and mixed-signal circuits for fast low-power arithmetic. To demonstrate the effectiveness of our approach, we have built a 16-tap 7-b 200-MHz mixed-signal finite-impulse response (FIR) filter that consumes 3 mW at 3.3 V. The filter uses synapse pFETs to store the analog tap coefficients, electron tunneling and hot-electron injection to modify the coefficient values, digital registers for the delay line, and multiplying digital-to-analog converters to multiply the digital delay-line values with the analog tap coefficients. The measured maximum clock speed is 225 MHz; the measured tap-multiplier resolution is 7 b at 200 MHz. The total die area is 0.13 mm$^2$. We can readily scale our design to longer delay lines.**

***Index Terms*—Adaptive filters, delay lines, high-speed integrated circuits, hot carriers, mixed analog–digital integrated circuits, signal processing, tunneling, very large-scale integration.**

## I. Introduction

THERE IS an ever-present need for low-power high-performance signal-processing chips. For example, to support higher data rates and provide better channel separation, wireless communications devices implement signal-processing algorithms that require high computational throughput. At the same time, these devices require low-power compact circuits to maximize battery life and minimize off-chip communication and system size. Digital-signal processing (DSP) chips, although immensely popular, tend to be large and power-hungry; thus applications that require both high throughput and low power employ special-purpose digital VLSI circuitry [1]. However, even custom digital solutions are inadequate for ultralow-power applications, mainly due to the area and power cost associated with digital multipliers and adders [2]. With the introduction of new, more sophisticated communications standards, in addition to the proliferation of mobile multimedia devices, the design of digital signal-processing subsystems in wireless devices will become increasingly difficult.

Although analog circuitry can implement arithmetic functions with low power dissipation and small area, these circuits present other problems such as offsets, error accumulation, and noise sensitivity, limiting their scalability and resolution. In particular, offsets and signal attenuation make it difficult to implement long tapped delay lines, which are common in filtering applications [3]. Additionally, adaptive systems in mobile communications employ circuits that periodically compute and store coefficient values on chip [4]. The lack of a means for accurate long-term analog coefficient storage complicates the implementation of these systems, forcing designers to use digital memories and implement computations in the digital domain [5].

Manuscript received August 11, 2000; revised December 1, 2000.

The authors are with the Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-2350 USA (e-mail: miguel@cs.washington.edu; hsud@cs.washington.edu; diorio@cs.washington.edu).

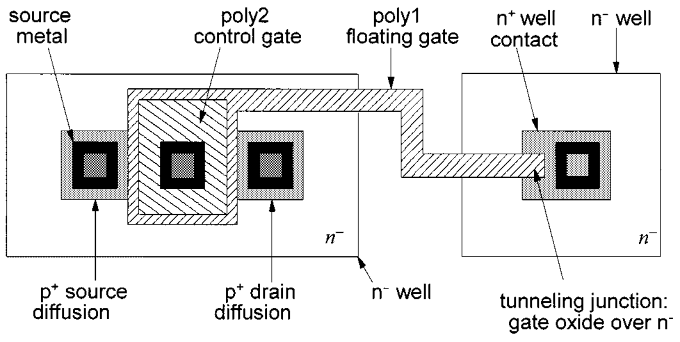
Publisher Item Identifier S 0018-9200(01)03025-6.

The Silicon Learning Group at the University of Washington, Seattle, studies the integration of adaptation and computation in VLSI circuits. Our approach is based on a new silicon primitive which we call a synapse transistor [6], [7]. This device is a floating-gate MOSFET that provides permanent analog weight storage, bidirectional updates for weight adaptation, and simultaneous adaptation and computation. We have developed analog and mixed-signal circuits around these devices that enable small high-throughput low-power VLSI systems. We are currently investigating the application of these devices in machine learning [8] and adaptive signal processing applications [9]. In this paper, we present a finite-impulse response (FIR) filter that employs digital delay lines, synapse transistors for weight storage and updates, and mixed-signal hardware for compact low-power four-quadrant arithmetic. Our 16-tap filter operates in open loop; although there is no on-line adaptation, we can write arbitrary tap-coefficient values at any point during operation. The filter runs at 200 MHz with 7-b accuracy, dissipating less than 3 mW. The die area is 0.13 mm$^2$ in a 0.35-$\mu$m CMOS process. We have an extended version of this architecture in fabrication which features 10-b resolution and 64 taps at comparable power and area cost, as well as an intrinsic silicon implementation of the least mean squares (LMS) [10] adaptation algorithm.

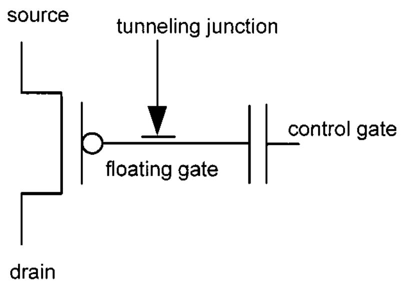
The rest of this paper is organized as follows. In Section II we discuss synapse transistors in more detail. In Section III we introduce the analog memory cell employed in our filter. We present the design of the filter in Section IV, including details on the delay line, weight storage, and arithmetic units. Finally, we discuss our experimental results and conclusions.

## II. Synapse Transistors

A synapse transistor is a floating-gate MOSFET with the following attributes: 1) nonvolatile analog weight storage, 2) locally computed bidirectional weight updates, and 3) simultaneous memory reads and writes. Fig. 1 illustrates the four-terminal pFET synapse that we use to store coefficient values in our filter. It comprises a single MOSFET (with poly1 floating gate and poly2 control gate) and an associated n$^-$ well tunneling implant. It uses floating-gate charge to represent a nonvolatile analog value, and outputs a source current that depends on both the control-gate input and the stored charge. The synapse transistor uses two mechanisms for adaptation: Fowler–Nordheim (FN) tunneling [11] increases the stored charge, and impact-ionized hot-electron injection (IHEI) [12] decreases the charge.



(a)



(b)

Fig. 1. (a) Layout view of a pFET synapse. The synapse comprises a single MOSFET, with poly1 floating gate and poly2 control gate, and an associated $\mathrm{n}^-$ well tunneling implant. (b) Circuit symbol for a pFET synapse.

Because tunneling and IHEI can both be active during normal transistor operation, the synapse enables simultaneous adaptation and computation. We can fabricate synapse transistors in any standard CMOS process, although double-poly processes are preferred because the second poly layer simplifies the implementation of a linear control gate. The reader can consult [13] for a discussion of synapse transistors in single-poly digital CMOS processes.

We operate the synapse pFET in the subthreshold regime [14], where the device shows an exponential relationship between gate voltage and source current. Consequently, we can substantially alter the source current with small changes in the floating-gate charge [see Fig. 1(b)]:

$$I_{\mathrm{s}} = I_0 e^{(\kappa(V_{\mathrm{s}}-V_{\mathrm{fg}}))/U_t} = I_0 \exp\left(\frac{\kappa}{U_t}\left(V_{\mathrm{s}} - \frac{(C_{\mathrm{in}}V_{\mathrm{in}}+Q_{\mathrm{fg}})}{C_T}\right)\right) \tag{1}$$

where

| | |
|---|---|
| $I_0$ | pre-exponential current; |
| $\kappa$ | coupling coefficient from floating gate to channel; |
| $V_{\mathrm{s}}$ | source voltage; |
| $V_{\mathrm{fg}}$ | floating-gate voltage; |
| $U_{\mathrm{t}}$ | thermal voltage $KT/q$; |
| $C_{\mathrm{in}}$ | input (poly1 to poly2) coupling capacitance; |
| $V_{\mathrm{in}}$ | control-gate voltage; |
| $Q_{\mathrm{fg}}$ | floating-gate charge; |
| $C_{\mathrm{T}}$ | total capacitance seen by the floating gate. |

We define the synapse weight $W_{\mathrm{s}}$ as the transistor's source current, and tie the source and the control gate to $V_{dd}$, so that the weight value depends only on $Q_{\mathrm{fg}}$:

$$W_{\mathrm{s}} = I_{\mathrm{s}} = I_1 e^{-(Q_{\mathrm{fg}}/Q_1)} \tag{2}$$

where $Q_1 = C_{\mathrm{T}}U_{\mathrm{t}}/\kappa$ and $I_1 = I_0 \exp(V_{dd}(C_{\mathrm{T}} - C_{\mathrm{in}})/Q_1)$. Equation (2) shows that we can control the value of $W_{\mathrm{s}}$ just by varying the amount of charge on the floating gate. The rest of this section discusses the mechanisms that we use to update the weight.

## A. Electron Tunneling Decreases the Weight

To decrease the value of $W_{\mathrm{s}}$, we apply a positive high voltage (~9.5 V in a 0.35-$\mu$m CMOS process) to the $\mathrm{n}^-$ well of the tunneling junction. The potential difference between the well and the floating gate reduces the effective oxide thickness, facilitating electron tunneling from the floating gate into the $\mathrm{n}^-$ well. This process increases $Q_{\mathrm{fg}}$, decreasing $W_{\mathrm{s}}$. From (2), the weight update rate is given by

$$\left(\frac{dW_{\mathrm{s}}}{dt}\right)_{\mathrm{tun}} = \left(\frac{dW_{\mathrm{s}}}{dQ_{\mathrm{fg}}}\right) \times \left(\frac{dQ_{\mathrm{fg}}}{dt}\right)_{\mathrm{tun}} = -\frac{W_{\mathrm{s}}}{Q_1} I_{\mathrm{tun}} \tag{3}$$

where $I_{\mathrm{tun}}$ is the gate current due to FN electron tunneling. We approximate $I_{\mathrm{tun}}$ using a simplified FN fit [15] [see Fig. 1(b)]

$$I_{\mathrm{tun}} = I_t e^{-(V_f/(V_{\mathrm{tun}}-V_{\mathrm{fg}}))} = I_t e^{-(V_f/V_{\mathrm{ox}})} \tag{4}$$

where

| | |
|---|---|
| $I_t$ | pre-exponential constant; |
| $V_{\mathrm{tun}}$ | voltage applied to the tunneling junction; |
| $V_{\mathrm{ox}}$ | oxide voltage, defined as $(V_{\mathrm{tun}} - V_{\mathrm{fg}})$. |

$I_{\mathrm{tun}}$ is positive because tunneling increases the floating-gate charge. Fig. 2(a) shows experimental data for $I_{\mathrm{tun}}$ versus $1/V_{\mathrm{ox}}$, along with a fit according to (4).

## B. Electron Injection Increases the Weight

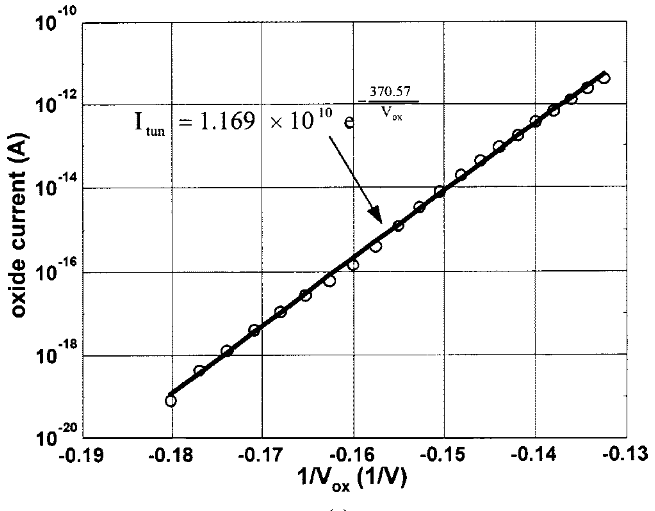
To increase the value of $W_{\mathrm{s}}$, we inject electrons onto the floating gate. We increase the source-to-drain voltage (to ~4 V in a 0.35-$\mu$m CMOS process) to create a large electric field between channel and drain. This field accelerates channel holes in the transistor's channel-to-drain depletion region, which collide with the semiconductor lattice and ionize free electrons. Because the floating-gate voltage is considerably higher than the drain voltage (the device is operating in the subthreshold regime), when the ionized electrons are expelled from the drain they can scatter upward into the gate oxide and be collected by the floating gate. The accumulation of electrons on the floating gate decreases $Q_{\mathrm{fg}}$, thereby increasing $W_{\mathrm{s}}$. Similarly to (3), we can express the weight update rate due to IHEI as

$$\left(\frac{dW_{\mathrm{s}}}{dt}\right)_{\mathrm{inj}} = \left(\frac{dW_{\mathrm{s}}}{dQ_{\mathrm{fg}}}\right) \times \left(\frac{dQ_{\mathrm{fg}}}{dt}\right)_{\mathrm{inj}} = -\frac{W_{\mathrm{s}}}{Q_1} I_{\mathrm{inj}} \tag{5}$$

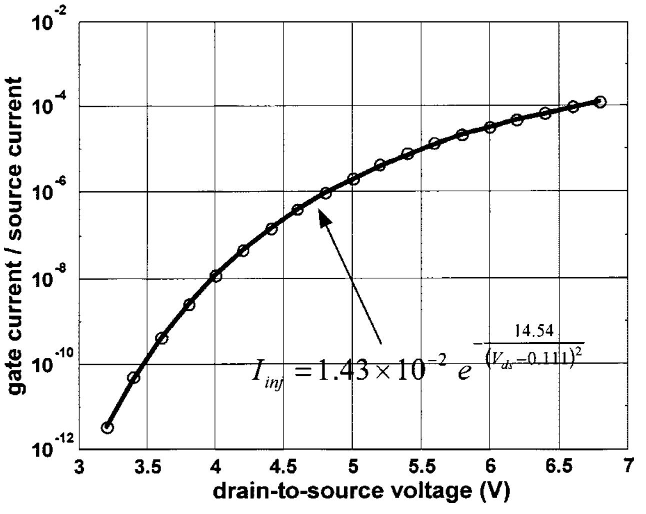
where $I_{\mathrm{inj}}$ is the gate current due to IHEI. We approximate $I_{\mathrm{inj}}$ as

$$I_{\mathrm{inj}} = -KI_{\mathrm{s}}e^{V_1/(V_{\mathrm{sd}}-V_2)^2} = -KW_{\mathrm{s}}e^{V_1/(V_{\mathrm{sd}}-V_2)^2} \tag{6}$$

where $K$, $V_1$, and $V_2$ are fit constants, $I_{\mathrm{s}}$ is the transistor's source current, and $V_{\mathrm{sd}}$ is the source-to-drain voltage. $I_{\mathrm{inj}}$ is negative because IHEI reduces the floating-gate charge.



(a)



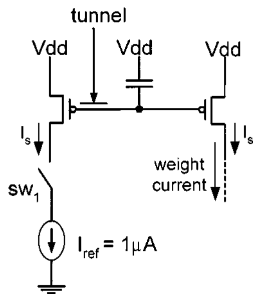
(b)

Fig. 2. (a) Tunneling (gate) current $I_g$ versus $-1/V_{ox}$. We define $V_{ox}$ to be the potential difference between the n$^-$ well contact and the floating gate. We show a fit according to a Fowler–Nordheim expression. (b) IHEI efficiency versus source-to-drain voltage. For this experiment, we held the floating-gate voltage, $V_{fg}$, and the source current, $I_s$, fixed, and we measured the gate current $I_g$ for different source-to-drain voltages.

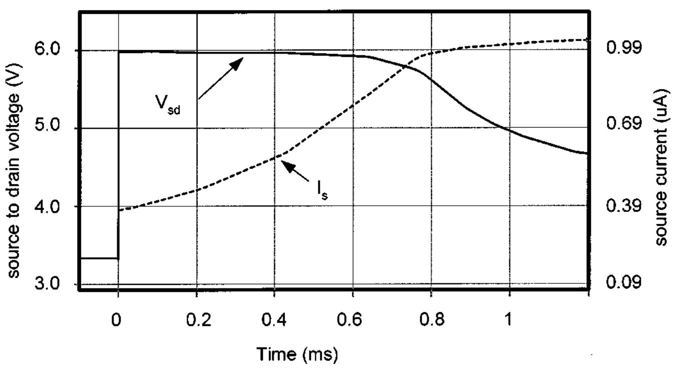
Fig. 2(b) shows experimental data for injection efficiency $(I_{inj}/I_s)$ versus $V_{sd}$, along with a fit from (6).

## III. Analog Memory Cell With Self-Convergent Writes

As (3)–(6) show, the weight-update rules defined by tunneling and IHEI depend exponentially on the control voltages ($V_{tun}$ and $V_{sd}$, respectively). In addition, the tunneling rule is proportional to $W_s$, and the injection rule is proportional to $W_s^2$ [substitute (6) into (5)]. Because the rules are multiplicative and exponential, writing an accurate value to the synapse transistor is difficult. Consequently, we have devised a writing mechanism that uses injection and negative feedback to accurately write a synapse pFET. The writing mechanism is *self-convergent* [15], meaning that an intrinsic self-limiting feedback path ensures that an analog value is stored accurately on the transistor. This feedback path is possible because the weight-update mechanisms can be active during normal device operation.



(a)



(b)

Fig. 3. (a) An analog memory cell with self-convergent writes. We apply a reference current $I_{ref}$ to the drain of the *injection transistor* (left) to cause IHEI onto the common floating gate. A self-limited feedback path (through the drain voltage) stops the injection when the transistor is able to source $I_{ref}$. The *weight transistor* (right) shares the floating gate with the injection transistor, thus it also sources $I_{ref}$ once the writing process has stopped. (b) Source-to-drain voltage and source current during the writing process (we close switch $sw_1$ at time $t = 0$). At first, the pFET is not able to source $I_{ref}$, and therefore its drain voltage drops to ground. As IEHI causes the floating-gate voltage to decrease, the source current $I_s$ rises, exponentially approaching $I_{ref}$. When $I_s = I_{ref}$, the drain voltage rises, stopping the injection process.

Fig. 3(a) depicts our analog memory cell, comprising two synapse transistors with a common floating gate and tunneling junction. The *weight transistor* provides a current (the weight value), whereas the *injection transistor* controls the writing process. Because the transistors share floating-gate, control-gate, and source terminals, the two devices source nominally identical currents. To write a memory, we first erase the value stored in the memory cell by applying a high voltage to the tunneling junction. We then close switch $sw_1$, and apply a reference current $I_{ref}$ to the drain of the injection transistor. If $I_{ref}$ is larger than the present source current $I_s$ in the device, the drain voltage in the injection transistor will drop, activating the injection process. Because injection is exponential in the source-to-drain voltage, electrons will rapidly accumulate on the floating gate, increasing $I_s$ [see (5) and (6)]. When $I_s$ reaches $I_{ref}$, the drain voltage will rise, turning off the injection process. This process effectively writes the current $I_{ref}$ on the memory cell. Fig. 3(b) shows the source current and source-to-drain voltage of the weight transistor during a simulated memory write.

It is possible to write a synapse transistor with a resolution exceeding 12 b [16]. However, the exponential dynamics of the self-convergent memory-write mechanism translate into fast moderate-accuracy writes, and slow high-accuracy writes. In practice, we find the self-convergent mechanism attractive

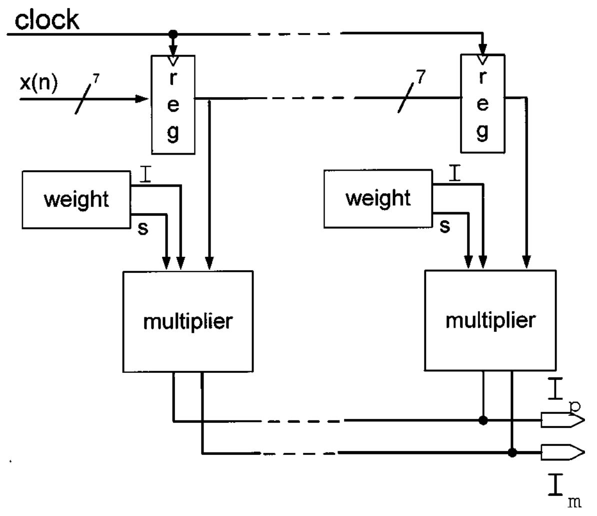


Fig. 4. Filter architecture. We use a 7-b delay line and store the tap coefficients on analog weight cells. We implement the multipliers using differential multiplying digital-to-analog converters (MDACs). The chip output is a differential current, comprising the sum of the currents from the 16 MDACs.

for writing weights with a resolution on the order of 6 b. For higher resolutions, a pulse-writing mechanism provides a better speed/accuracy tradeoff.

## IV. Filter

Fig. 4 shows the filter architecture. Because scaleable analog delay lines are difficult to implement in VLSI, we use a 7-b digital delay line to shift the input signal across the filter taps. We store each tap weight in an analog memory cell, which we can selectively erase and write. We implement the multipliers with multiplying digital-to-analog converters (MDACs) [17] which provide a differential current output. The chip output is also a differential current, comprising the sum of the currents from the 16 MDACs. We implement the sum by connecting the outputs of all the MDACs in the filter to common wires. The rest of this section describes the design of the key modules.

### *A. Tap Coefficient Storage*

Fig. 5 shows the coefficient-storage cell. We store the coefficient magnitude in an analog memory cell based on the designed presented in Section III. We store the coefficient sign in a static digital latch. Because our multiplier comprises a two-segment MDAC that requires two identical bias currents, the analog memory cell contains two weight transistors with a common floating gate. We cascode a pFET source follower to the drain of each weight transistor, to prevent these devices from injecting. We tie the control gate to a bias line that is common to the entire filter. By varying the voltage on this line, we control the output-current range of the memory cell. This feature lets us trade power for performance and resolution. In all the experiments presented in this paper, we tied the global bias line to $V_{dd}$. A digital select line controls the switches at the drain of the injection transistor and the input of the latch, enabling us to write a new tap-coefficient value into the memory cell. We can selectively erase each cell via independent tunneling terminals.

Based on this memory cell and the requirements of the multiplier, we define the tap-coefficient $W$ using a sign-magnitude

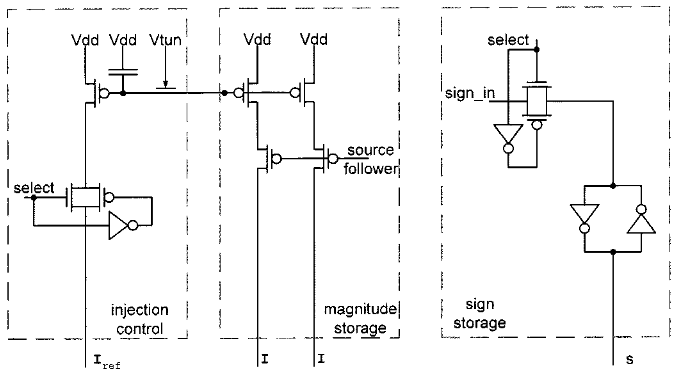


Fig. 5. Weight-storage cell. We store the tap-coefficient magnitude on a synapse transistor-based memory cell, and the sign on a static latch. We can change the coefficient magnitude using selectable tunneling and injection circuitry.

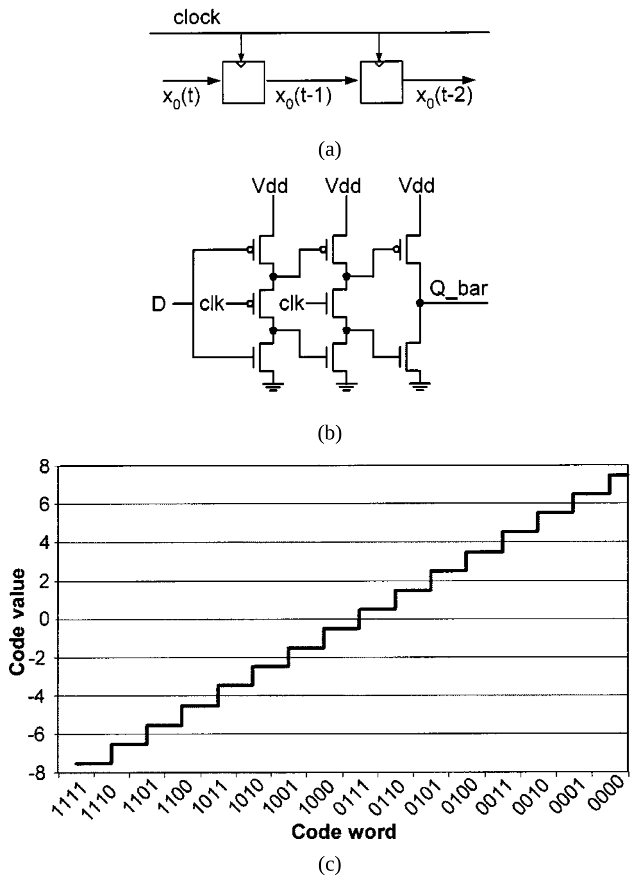


Fig. 6. (a) We used a digital shift-register to implement the delay line. (b) True-single-phase flip-flop used to build the shift register. This circuit is a dynamic positive-edge-triggered D-flip-flop with split outputs. (c) Offset-binary code used to represent the input (only four bits shown). In this code, each bit represents a signed contribution to the word value. The code is symmetric, so if the tap coefficient is negative, we only need to invert every bit of the input prior to computing the product of the input and the tap-coefficient magnitude.

representation:

$$W = \tfrac{1}{4} \times (-1)^s \times I \tag{7}$$

where

| | |
|---|---|
| $W$ | tap coefficient; |
| $s$ | sign bit stored in the latch; |
| $I$ | output current from the synapse transistors. |

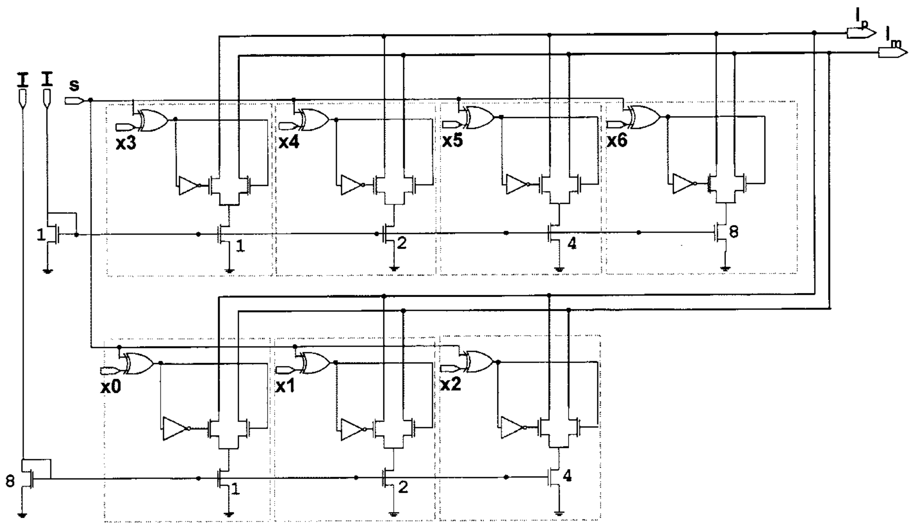


Fig. 7. The multiplier is based on a two-segment differential MDAC with scaled current mirrors. We compute the sign arithmetic using XOR gates: If the sign of the coefficient is negative (a binary value of 1), we invert every bit of the input word. The MDAC computes the product of the sign-corrected input word and the magnitude of the weight, given by the bias currents provided by the analog memory cell. The number next to each nFET in the MDAC represents the relative width of the devices. To minimize the effects of process variations and device mismatch, we used multiple identical parallel transistors, instead of scaling their width.

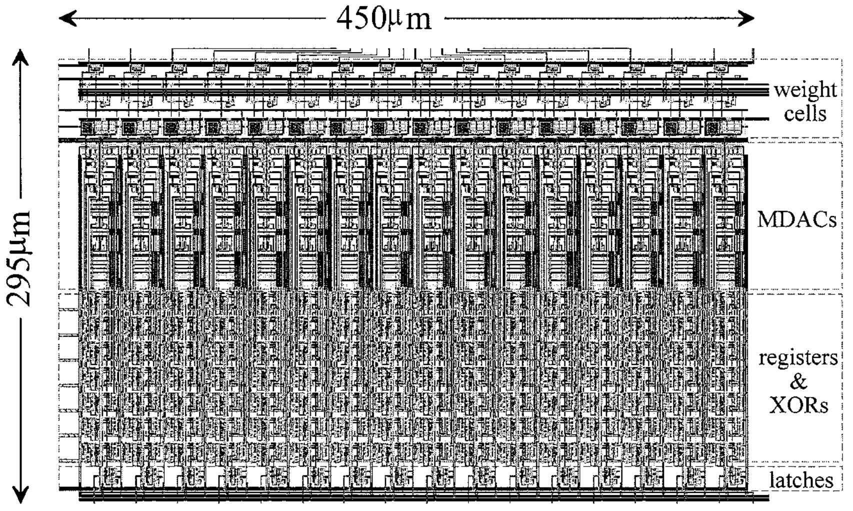


Fig. 8. Filter layout. The filter is 450 $\mu$m wide and 295 $\mu$m wide in a triple-metal double-poly 0.35-$\mu$m CMOS process available from MOSIS. The digital circuitry takes 48% of the die area, while 35% and 17% are allocated to the MDACs and analog memory cells, respectively.

We use a scale factor of 1/4 in our representations, for reasons that will become clear when we discuss the design of the multiplier in subsection C.

### B. Delay Line

As shown in Fig. 6(a) and (b), we implemented the digital delay line as a 7-b shift register composed of true-single-phase-clock (TSPC) D-flip-flops [18]. We used an offset-binary code for the digital input. This is a positional code, where each bit contributes to the value of the word with a power of two, depending on the bit position. The sign of the contribution of each bit is given by its value:

$$X = \tfrac{1}{2}\sum_{i=0}^{6}(-1)^{x_i} \times 2^i \tag{8}$$

where $X$ is the word value and $x_i$ is the value of the $i$th bit of the digital representation of $X$ ($x_0$ is the LSB). Fig. 6(c) shows the coding scheme. We chose seven bits for our representation because this resolution is reasonable for untrimmed DACs in a digital CMOS process. For higher bit resolutions, we can use synapse transistors to implement on-chip DAC calibration.

### C. Multiplier

Each filter tap multiplies the tap weight and the input. From (7) and (8), we can derive an expression for the multiplication:

$$Y = W \times X = \tfrac{1}{4} \times (-1)^s \times I \times \tfrac{1}{2}\sum_{i=0}^{6}(-1)^{x_i} \times 2^i$$

$$Y = \frac{I}{8}\sum_{i=0}^{6}(-1)^{x_i \ominus s} \times 2^i. \tag{9}$$

Fig. 7 shows the multiplier implementation. The circuit is basically a two-segment MDAC [17], which accepts a digital input from the tap register ($x[6\ldots0]$), and two identical bias currents ($I$) from the tap-coefficient memory cell. The first segment (bottom) of the MDAC divides $I$ by 8, and multiplies this

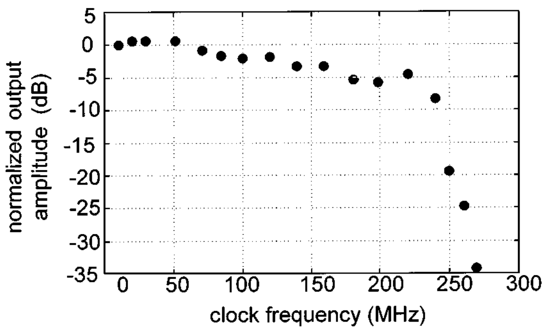


Fig. 9. Filter output amplitude versus clock frequency. We set all the tap weights to the same positive value, and applied a one-clock-cycle impulse input waveform, so that the output shows the contribution of only one tap at any time. We measured the output amplitude as we clocked the pulse from tap to tap. The graph shows the filter output normalized to a base clock speed of 10 MHz. With clock speeds up to 220 MHz, we measured an output resolution of 7 b. For higher clock speeds, the output amplitude quickly decays.

current by 1, 2, and 4 using scaled current mirrors. The second segment takes $I$ and multiplies it by powers of 2 from 1 through 8, thereby computing each one of the terms of the sum in (9). The sign of each term is determined by an XOR function between the corresponding bit of the input word ($x_0$ through $x_6$), and the tap-coefficient sign from the memory cell. The XOR gates drive differential pairs that route the current sunk by the mirrors to the positive or negative terminals of the multiplier's differential output ($I_p$ and $I_m$). In this way, we add the DAC-bit currents at $I_p$ and $I_m$, respectively. We use the same technique to add the outputs of all the multipliers in the filter.

## V. Experimental Results

We fabricated the filter in a 0.35-$\mu$m double-poly three-metal CMOS process available from MOSIS. The die area of the entire circuit is 0.13 mm$^2$, and each tap measures 28 $\mu$m by 295 $\mu$m. Fig. 8 shows the filter layout. The digital components account for about 48% of the die area, 35% is devoted to the MDACs, and the remaining 17% is occupied by the analog memory cells. We tested the chip using a digital oscilloscope as a 50-$\Omega$ differential transimpedance amplifier and deglitcher.

The filter runs at clock speeds to 225 MHz, while maintaining a resolution of 7 b. At 200 MHz, the filter dissipates 3 mW with a maximum tap-weight current of 1 $\mu$A and a 3.3-V power supply. At this speed, the digital circuits dissipate about 70% of the total power. Running at 100 MHz, the filter dissipates 2 mW, distributed evenly between the digital and analog circuitry. We tested the response of the filter to different clock frequencies: We set all the taps weights to their maximum value, and applied an impulse waveform as input, so that only one of the taps would be on at any time. Fig. 9 shows the output of the filter versus clock frequency, normalized to its response at a reference clock frequency of 10 MHz. As the figure shows, the output rapidly decays at clock speeds above 225 MHz. Our post-layout simulations suggest that the filter should maintain a 7-b resolution at clock speeds to 500 MHz. Package and test-setup limitations (we used a 40-pin ceramic dual in-line package (DIP) mounted on a chip socket) likely constrained our performance results.

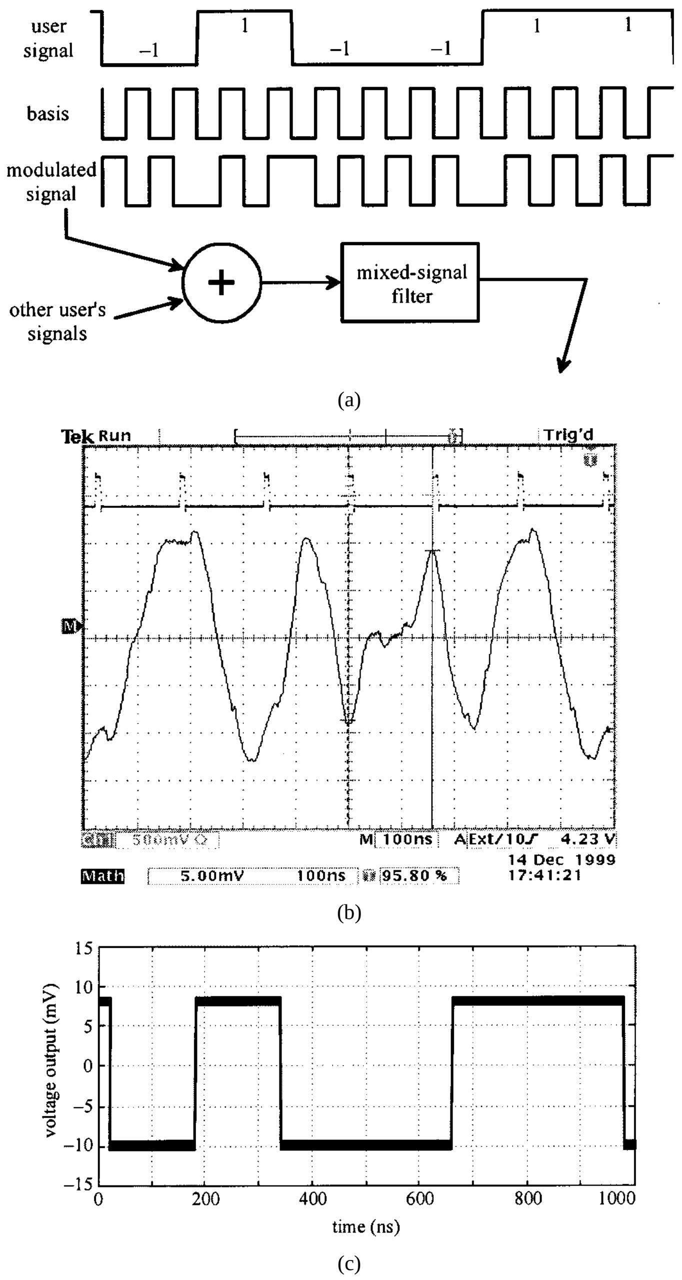


Fig. 10. Application to DS-CDMA despreading. We applied a 100 Mb/s CDMA-like input, comprising two bit streams encoded using orthogonal bases, to the filter. We set the tap coefficients to decode the shown basis. (a) Input bit stream and the basis used to encode it. (b) Filter output and the strobe pulse used to recover the data. (c) Reconstructed data, for 64 (superimposed) experiments, showing the logic-level variance at the output. We used the oscilloscope as a differential 50-$\Omega$ transimpedance amplifier, low-pass filter and track–hold, and reconstructed the output data in software using the oscilloscope measurements. In this application, the filter supports an input dynamic range of 42.6 dB (7 b).

Finally, we tested the filter in a simple direct-sequence multiple access (DS-CDMA) despreading application. We encoded two user bit streams with orthogonal signatures, and added them to form a combined signal at a chip rate of 100 Mb/s. We programmed the filter coefficients with one of the users' signatures, and used it to recover the original bit stream. Fig. 10 shows the results of this experiment. From the measured signal-to-noise ratio at the output, we determined that the filter supports an input dynamic range of 42.6 dB, which is consistent with the 7-b resolution of the input bit stream.

## VI. Conclusion

We have built a 16-tap FIR filter that uses synapse transistors for analog weight storage and weight updates. This approach allows us to use mixed-signal arithmetic units, resulting in a compact high-speed low-power design. Because we use a digital delay line, we can scale our solution to a larger number of taps ($\gg 16$). On a DS-CDMA decoding application, our filter demonstrated an input dynamic range of 42.6 dB, enabling us to scale the design up to 128 taps (up to 128 users). Future and ongoing research include an enhanced version of the filter, which supports more taps and higher bit resolution, as well as a compact mixed-signal implementation of the LMS adaptation algorithm that optimizes the filter response by modifying the coefficients on-line.

## References

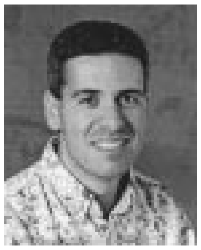
[1] C. Teuscher, S. Sheng, I. O'Donnell, K. Stone, and R. Brodersen, "Design and implementation issues for a wideband indoor DS-CDMA system providing multimedia access," in *Proc. 34th Annu. Allerton Conf. Communication, Control, and Computing*, Urbana-Champaign, IL, Oct. 1996, pp. 623–632.
[2] N. Zhang, C. Teuscher, H. Lee, and B. Brodersen, "Architectural implementation issues in a wideband receiver using multiuser detection," in *Proc. 36th Annu. Allerton Conf. Communication, Control, and Computing*, Urbana-Champaign, IL, Sept. 1998, pp. 765–771.
[3] K. K. Onodera and P. R. Gray, "A 75-mW 128-MHz DS-CDMA baseband demodulator for high-speed wireless applications," *J. Solid-State Circuits*, vol. 33, pp. 753–761, May 1998.
[4] R. Lupas and S. Verdu, "Linear multiuser detectors for synchronous code-division multiple access channels," *IEEE Trans. Information Theory*, vol. IT-35, pp. 123–136, Jan. 1989.
[5] M. Q. Le, P. J. Hurst, and J. P. Keane, "An adaptive analog noise-predictive decision-feedback equalizer," in *Proc. Symp. VLSI Circuits*, Honolulu, HI, 2000, pp. 216–217.
[6] C. Diorio, P. Hasler, B. Minch, and C. Mead, "A complementary pair of four-terminal silicon synapses," *Analog Integrated Circuits and Signal Processing*, vol. 13, no. 1/2, pp. 153–166, 1997.
[7] C. Diorio, "Neurally inspired silicon learning: From synapse transistors to learning arrays," Ph.D. dissertation, California Institute of Technology, Pasadena, CA, 1997.
[8] D. Hsu, M. Figueroa, and C. Diorio, "A silicon primitive for competitive learning," Univ. of Washington, Seattle, WA, Tech. Rep. 2000-07-01, July 2000.
[9] M. Figueroa, "A mixed-signal adaptive correlator for low-power signal processing," Univ. of Washington, Seattle, WA, Tech. Rep. 2000-08-02, Aug. 2000.
[10] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
[11] M. Lenzlinger and E. H. Snow, "Fowler–Nordheim tunneling into thermally grown SiO2," *J. Appl. Phys.*, vol. 40, no. 1, pp. 278–283, 1969.
[12] E. Takeda, C. Yang, and A. Miura-Hamada, *Hot Carrier Effects in MOS Devices*. San Diego, CA: Academic, 1995.
[13] B. A. Minch and P. Hasler, "A floating-gate technology for digital CMOS processes," in *Proc. IEEE Intl. Symp. Circuits and Systems*, vol. 2, 1999, pp. 400–403.
[14] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.
[15] C. Diorio, "A p-channel MOS synapse transistor with self-convergent memory writes," *IEEE Trans. Electron Devices*, vol. 47, pp. 464–472, Feb. 2000.
[16] C. Diorio, S. Mahajan, P. Hasler, B. A. Minch, and C. Mead, "A high-resolution nonvolatile analog memory cell," in *Proc. IEEE Intl. Symp. Circuits and Systems*, vol. 3, 1995, pp. 2233–2236.
[17] E. L. Zuch, "Principles of data acquisition and conversion," in *Data Acquisition and Conversion Handbook*, E. L. Zuch, Ed. Mansfield, MA: DATEL, 1979, pp. 13–18.
[18] J. M. Rabaey, "Dynamic sequential circuits," in *Digital Integrated Circuits: A Design Perspective, Electronics and VLSI*. Englewood Cliffs, NJ: Prentice-Hall, 1996, pp. 359–362.

**Miguel Figueroa** (S'01) received the B.S., professional, and M.S. degrees in electrical engineering from the University of Concepción, Chile, in 1988, 1991, and 1997, respectively. He is currently working toward the Ph.D. degree in the Department of Computer Science and Engineering, University of Washington, Seattle, where he received the M.S. degree in computer science in 1999.

His research interests include VLSI design, neurobiology-inspired computation, and reconfigurable architectures.

**David Hsu** received the B.S. degree in electrical engineering and computer science from the University of California, Berkeley, in 1996. He is currently working toward the Ph.D. degree in computer science at the University of Washington, Seattle.

His research interests are in the area of machine learning, neural architectures, and analog VLSI.



**Chris Diorio** (M'88) received the B.A. degree in physics from Occidental College, Los Angeles, CA, in 1983, and the M.S. and Ph.D. degrees in electrical engineering from the California Institute of Technology, Pasadena, CA, in 1984 and 1997, respectively.

He is currently an Assistant Professor of Computer Science and Engineering at the University of Washington, Seattle. He has worked as a Senior Staff Engineer at TRW, Inc., as a Senior Staff Scientist at American Systems Corporation, and as a Technical Consultant at The Analytic Sciences Corporation. His research focuses on building electronic circuits and systems that mimic the computational and organizational principles found in the nervous systems of living organisms.

Dr. Diorio received an Alfred P. Sloan Foundation Research Fellowship in 2000, a Presidential Early Career Award in Science and Engineering (PECASE) in 1999, a Packard Foundation Fellowship in 1998, a National Science Foundation CAREER Award in 1998, and the Electron Devices Society's Paul Rappaport Award in 1996.