Adaptive CMOS: From Biological Inspiration to Systems-on-a-Chip

CHRIS DIORIO, MEMBER, IEEE, DAVID HSU, AND MIGUEL FIGUEROA

Encouraged Paper

Local long-term adaptation is a well-known feature of the synaptic junctions in nerve tissue. Neuroscientists have demonstrated that biology uses local adaptation both to tune the performance of neural circuits and for long-term learning. Many researchers believe it is key to the intelligent behavior and the efficiency of biological organizms. Although engineers use adaptation in feedback circuits and in software neural networks, they do not use local adaptation in integrated circuits to the same extent that biology does in nerve tissue. A primary reason is that locally adaptive circuits have proved difficult to implement in silicon. We describe complementary metal–oxide–semiconductor (CMOS) devices called synapse transistors that facilitate local long-term adaptation in silicon. We show that synapse transistors enable self-tuning analog circuits in digital CMOS, facilitating mixed-signal systems-on-a-chip. We also show that synapse transistors enable silicon circuits that learn autonomously, promising sophisticated learning algorithms in CMOS.

Keywords*—*Adaptive, local learning, neural, SOC, synapse transistor.

I. INTRODUCTION

The synapses and neurons in animal brains encode and process information using electrical and chemical signaling, with extraordinary efficiency, under incredibly tight power and supply-voltage constraints [1]. These same synapses and neurons are poorly matched across nerve tissue, degrade over life, and do not even have a common supply voltage or a common ground. These observations have led many researchers (us included) to study biology for inspiration in engineering design. They have also provided impetus for research in artificial neural networks and neuromorphic engineering [2]. We believe that the inspiration provided by biological synapses holds even more promise today than it has in the past, in part because huge opportunities still exist to develop neural-network hardware and in part because of challenges in integrating analog and digital circuitry in modern systems-on-a-chip (SOCs).

Although contemporary implementations of neural networks and machine-learning algorithms are almost entirely software based, there remains the prospect for huge performance gains if engineers could build hardware (silicon) versions of these networks. Because all-digital learning hardware requires costly circuits (in terms of die size and power) such as multipliers, researchers continue exploring analog hardware [3]. Unfortunately, large-scale analog learning has to date eluded researchers. A primary reason is the lack of a simple way to enable local parallel online adaptation in silicon.

The scaling of silicon integrated-circuit processing to deep-submicrometer feature sizes poses significant challenges for SOC design. On the positive side, scaling increases the density and speed of digital complementary metal–oxide–semiconductor (CMOS). On the negative side, scaling burdens analog CMOS with low transistor-breakdown voltages, poor transistor matching, and an absence of high-valued resistors, high-Q inductors, or linear capacitors. SOC applications typically require deep-submicrometer CMOS for the digital circuitry, but have analog inputs and/or outputs. To enable mixed-signal SOC applications, engineers need a simple way to design precision analog circuits side by side with digital logic in standard digital CMOS processes.

So what do neurobiology and silicon learning and SOC have in common? We believe that local adaptation is key. For example, despite our ignorance of how nerve tissue actually performs its computations, we know that it uses long-term local adaptation to tune the performance of its synaptic junctions [4]. As another example, researchers developing precision analog-to-digital or digital-to-analog converters may resort to local on-line calibration to maintain performance

Manuscript received July 17, 2001; revised December 7, 2001. This work was supported by the National Science Foundation under Grant ECS9733425, by Packard and Sloan Foundation Fellowships, by the Office of Naval Research Young Investigator Program, and by Impinj, Inc.

C. Diorio and M. Figueroa are with the Computer Science and Engineering Department, the University of Washington, Seattle, WA 98195 USA and also with Impinj, Inc., Seattle, WA 98103 USA.

D. Hsu with the Computer Science and Engineering Department, the University of Washington, Seattle, WA 98195 USA.

Publisher Item Identifier S 0018-9219(02)02901-8.

[5]. We believe that if engineers had a simple means to incorporate local parallel adaptation in their silicon chips, like neurobiology does in nerve tissue, they could greatly advance learning-network and SOC performance and applications.

We and others [6] have spent years developing silicon devices that mimic the adaptive synapses nervous systems use for memory and learning. The result is a family of single-transistor devices we call *synapse transistors* [7]–[12] that implement long-term nonvolatile analog memory, allow bidirectional memory updates, and learn from an input signal without interrupting the ongoing computation. Although we do not believe that a single transistor can model the complex behavior of a neural synapse completely, our synapse transistors do implement long-term local learning: their output depends not only on the present input, but also on a history of prior inputs.

Synapse transistors allow us to build silicon chips that learn and adapt locally and autonomously in a fashion similar to that used by biology (we believe) to tune its circuits. Using them, we can build both precision analog circuits and artificial learning networks in digital CMOS. We begin this paper by first describing synapse transistors because they are the key enabling technology. Fundamentally, they are metal–oxide–semiconductor (MOS) transistors that tune their performance during normal device operation. We then demonstrate by several working examples how synapse transistors can enable mixed-signal SOC and silicon-learning networks. These examples, all fabricated in digital CMOS, include a digital-to-analog converter (DAC) with 6-bit intrinsic accuracy that trims electrically to 14 bits and an unsupervised competitive-learning circuit that learns to unmix a mixture of Gaussians. We conclude with a discussion of current and future process-related technology issues.

II. pFET Synapse Transistor

We define synapse transistors to be conventional transistors with the following additional attributes: 1) nonvolatile analog weight storage; 2) locally computed bidirectional weight updates; and 3) simultaneous memory reading and writing. We use floating-gate metal–oxide–semiconductor field-effect transistors (MOSFETs) as the basis for all of our synapse transistors. Our synapses use floating-gate charge to represent the nonvolatile analog weight, electron tunneling and hot-electron injection to modify the floating-gate charge bidirectionally, and allow simultaneous memory reading and writing by nature of the mechanisms we use to write the memory. We have developed a family of such devices [10], but primarily use just one, a p-channel MOSFET (pFET) synapse because of its compatibility with standard digital CMOS processing.

We show a conceptual model for a pFET synapse in Fig. 1 and the layout and band diagram [13] in Fig. 2. The synapse comprises two MOSFETs: the first is a readout transistor; the second, with shorted drain and source, forms a tunneling junction. From the control gate's perspective, removing electrons from or adding electrons to the floating gate shifts the readout pFET's threshold voltage bidirectionally. The synapse uses Fowler–Nordheim (FN) tunneling [14] to remove electrons from its floating gate and impact-ionized hot-electron injection (IHEI) [15] to add electrons to the floating gate.

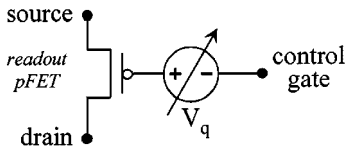


Fig. 1. Simplified circuit model for a pFET synapse. Electron tunneling and injection modify the gate offset voltage V_q.

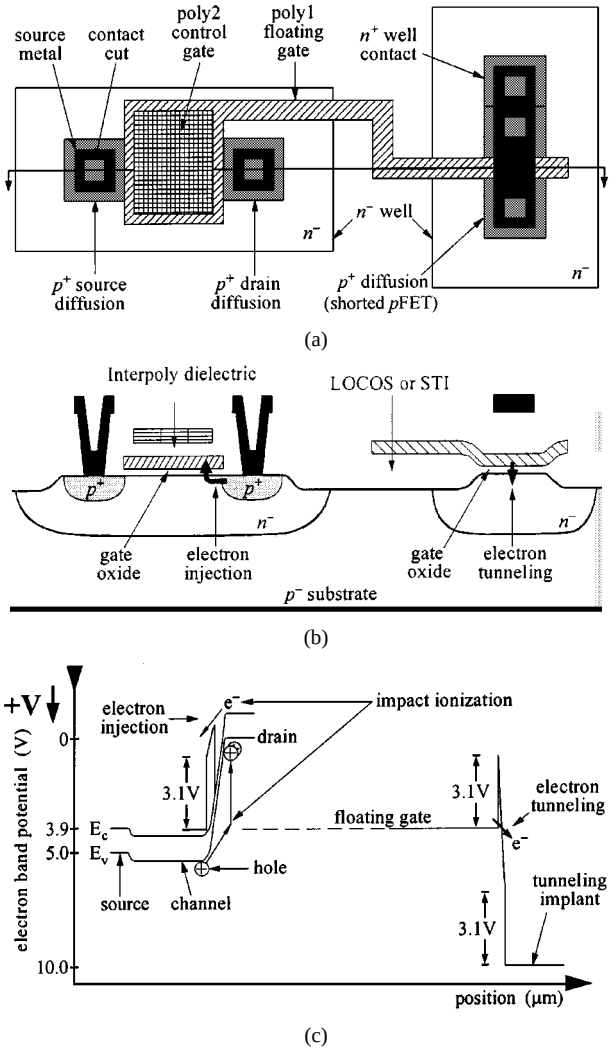


Fig. 2. pFET synapse, showing the electron tunneling and injection locations. (a) Top view. (b) Side view. (c) Electron band diagram. We aligned the three diagrams vertically, drew A and C to scale, exaggerated the vertical scale in B, and assumed subthreshold operation ($I_s < 100$ nA and a 0.35-μm process. Although the gate oxide's band diagram projects vertically, to better illustrate the injection process, we rotated it by 90° and drew it in the channel direction. We decrease the synapse weight by tunneling electrons to the tunneling junction and increase the weight by injecting electrons from the drain region to the floating gate. Our tunneling junction comprises a shorted pFET in an n-well, for two reasons. First, a lightly doped n-well can accommodate high positive voltages without pn-junction breakdown to substrate. Second, a shorted pFET in an n-well is a valid structure (that satisfies design rules) in any CMOS process.

Key features of this synapse are: 1) the readout transistor remains a fully functional pFET; 2) high voltages applied to the tunneling junction tunnel electrons off the floating gate; and 3) large drain-to-source voltages cause IHEI at the drain, injecting electrons onto the floating gate.

A. Readout Transistor Remains a Fully Functional pFET

We apply signal inputs to the second-level polysilicon (poly2) control gate, which, in turn, couples capacitively to the first-level polysilicon (poly1) floating gate (see Fig. 2). From the control gate's perspective, the transistor remains a conventional pFET, albeit with reduced coupling to the channel because of the intervening poly1 capacitor.

If we operate the MOSFET in its subthreshold regime [16], the synapse is well suited for neural network applications. The reason is that a subthreshold floating-gate pFET performs a multiply operation as follows:

$$\begin{aligned} I_{\rm s} &= I_{\rm o} e^{(\kappa V_{\rm sfg})/U_{\rm t}} = I_{\rm o} e^{\kappa(Q_{\rm sfg}+C_{\rm in}V_{\rm in})/(C_{\rm T}U_{\rm t})} \\ &= I_{\rm o} e^{Q_{\rm sfg}/Q_{\rm T}} e^{(\kappa' V_{\rm in})/U_{\rm t}} \qquad (1) \\ &= W I_{\rm o} e^{(\kappa' V_{\rm in})/U_{\rm t}} \qquad (2) \end{aligned}$$

where $I_{\rm s}$ is the source current, $I_{\rm o}$ is a preexponential current, κ is the coupling coefficient from floating gate to channel, $V_{\rm sfg}$ is the source-to-floating-gate voltage, $Q_{\rm sfg}$ is the floating-gate charge (source referenced), $C_{\rm T}$ is the total capacitance seen by the floating gate, $U_{\rm t}$ is the thermal voltage kT/q, $C_{\rm in}$ is the input (poly1 to poly2) coupling capacitance, $V_{\rm in}$ is the control-gate voltage, $Q_{\rm T} \equiv C_{\rm T}U_{\rm t}/\kappa$, $\kappa' \equiv \kappa C_{\rm in}/C_{\rm T}$, and $W \equiv \exp(Q_{\rm sfg}/Q_{\rm T})$. The synapse weight W is a learned quantity: its value derives from the floating-gate charge, which can change with synapse use. The synapse output is the product of W and the source current of an idealized MOSFET that has a control-gate input $V_{\rm in}$ and a coupling coefficient κ' from the control gate to the channel.

For CMOS processes without poly2, we can use a MOS capacitor as an input capacitor [17] or, for applications that can tolerate the (small) charge leakage that occurs when we add a contact to the floating gate, we can connect the floating gate to a metal–insulator–metal capacitor. Alternatively, we sometimes use no capacitor (i.e., no gate input) at all; in this case, the synapse becomes a tunable current source.

B. Electron Tunneling Decreases the Weight

We decrease the synapse weight W by tunneling electrons from the floating gate to the tunneling junction (the shorted pFET and its associated n-well). Positive high voltages on the tunneling junction cause electron tunneling. We illustrate the FN-tunneling process in the energy-band diagram of Fig. 2. A potential difference between the tunneling junction and the floating gate reduces the effective oxide thickness, facilitating electron tunneling from the floating gate, through the SiO_2 barrier, and into the oxide conduction band. The oxide electric field then sweeps these electrons to the n-well.

In Fig. 3, we show tunneling current (oxide current) versus the reciprocal of the voltage across the oxide for synapses fabricated in 2- and 0.35-μm processes. We fit these data using a simplified FN fit [14], [18]

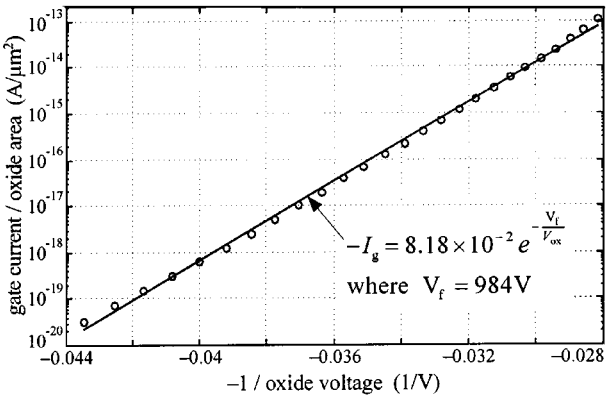


Fig. 3. Tunneling (gate) current $I_{\rm g}$ versus $-1/V_{\rm ox}$, for a synapse fabricated in a 2-μm CMOS process. $V_{\rm ox}$ is the potential between the tunneling junction and the floating gate. We normalized the gate current to the tunneling-junction (gate oxide) area.

$$I_{\rm g} = -I_{\rm tn} e^{-V_{\rm f}/V_{\rm ox}} \qquad (3)$$

where $I_{\rm g}$ is the gate current, $V_{\rm ox}$ is the oxide voltage (well voltage minus floating-gate voltage), $V_{\rm f}$ is a constant that depends primarily on oxide thickness, and $I_{\rm tn}$ is a preexponential current. $I_{\rm g}$ is negative because tunneling reduces the weight W.

C. Electron Injection Increases the Weight

We increase the synapse weight W by injecting electrons onto the floating gate. As shown in the energy-band diagram of Fig. 2, channel holes, accelerated in the transistor's channel-to-drain depletion region, can collide with the semiconductor lattice and liberate additional electron-hole pairs. The ionized electrons, promoted to their conduction band by the collision, are expelled from the drain by the same channel-to-drain electric field. Electrons expelled with more than 3.1 eV of kinetic energy, if scattered upward into the gate oxide, can overcome the 3.1-V difference in electron affinity between the Si and SiO_2 conduction bands, inject into the SiO_2, and be collected by the floating gate.

In Fig. 4, we plot IHEI efficiency (defined as gate current $I_{\rm g}$ divided by source current $I_{\rm s}$) for synapses fabricated in 2- and 0.35-μm processes. We plot the data as efficiency because gate current increases linearly with source current over the entire subthreshold range; predictably, because the gate current derives from the hot-electron population and this population, in turn, increases linearly with the source current.

For a 0.35-μm synapse, when the readout transistor's source-to-drain voltage $V_{\rm sd}$ is less than 3 V, the IHEI gate current is exceedingly small and the weight W remains nonvolatile. When $V_{\rm sd}$ exceeds 3.5 V, the gate current causes measurable changes in the synapse weight W. We approximate the data of Fig. 4 with a simple exponential

$$I_{\rm g} = \beta I_{\rm s} e^{V_{\rm cd}/V_{\rm inj}} \qquad (4)$$

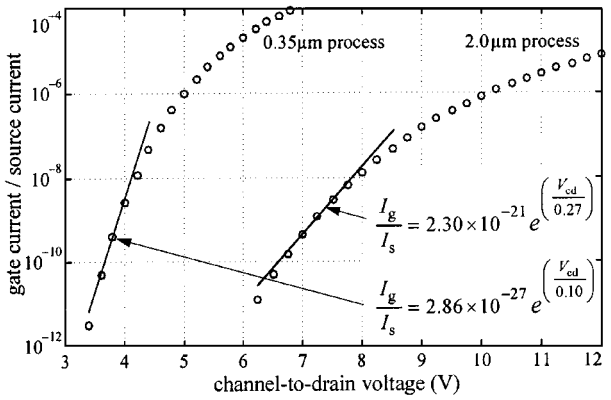


Fig. 4. IHEI efficiency (gate current I_g divided by source current I_s) versus channel-to-drain potential V_{cd}, for synapses fabricated in 2- and 0.35-μm processes. We reference the drain voltage to the channel because the hot-electron population derives from the channel-to-drain electric field. Source-to-drain voltage V_{scd} is a few hundred millivolts smaller than V_{cd}. In the subthreshold regime, I_g increases linearly with I_s; consequently, these data show the IHEI efficiency for the entire subthreshold source-current range.

where I_g is the gate current, I_s is the source current, V_{cd} is the channel-to-drain potential, and β and V_{inj} are fit constants. I_g is positive because IHEI increases the weight W.

III. Gate-Current Equation

In a synapse transistor, we can *simultaneously*: 1) read the channel current; 2) raise the tunneling voltage, causing electrons to tunnel off the floating gate; and 3) lower the drain voltage, causing IHEI. We obtain a final gate-current equation by adding (3) and (4)

$$I_g = \beta I_s e^{V_{cd}/V_{inj}} - I_{tn} e^{-V_f/V_{ox}} \tag{5}$$

assuming subthreshold source currents I_s. The restriction to subthreshold source currents is solely for reasons of mathematical tractability. The synapse is fully functional with above-threshold source currents, but the dynamics are more complicated (and are beyond the scope of this paper).

IV. Synaptic Arrays

In applications that use large numbers of synapse transistors, such as analog memories or neural networks, we use arrays of synapses rather than isolated devices. Although arrays provide dense synapse packing and simple addressing, they must not compromise the isolation between individual synapses and must provide a means for writing and erasing synapses easily. We fabricated the array shown in Fig. 5 to: 1) verify synapse isolation and 2) demonstrate a self-convergent technique for writing individual synapses.

A. Synapse Isolation

Array synapses share tunneling and drain wires; consequently, tunneling or injecting one synapse can cause undesired tunneling or injection at another synapse. To measure synapse isolation, we tunneled and injected the {1,1} synapse in Fig. 5 over a three-decade range, while measuring the crosstalk to the other synapses. We define crosstalk to

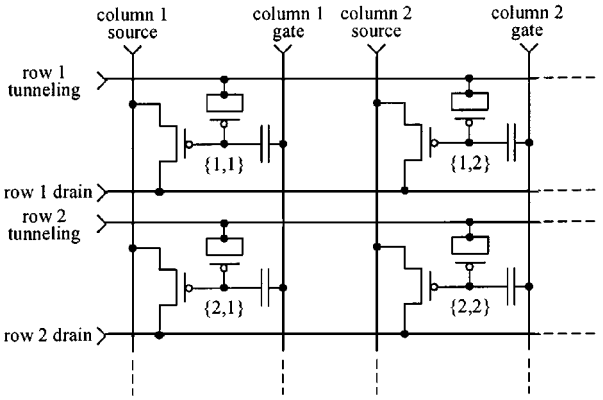


Fig. 5. Synaptic array. Column synapses share a common tunneling wire, meaning that they share a common tunneling well.

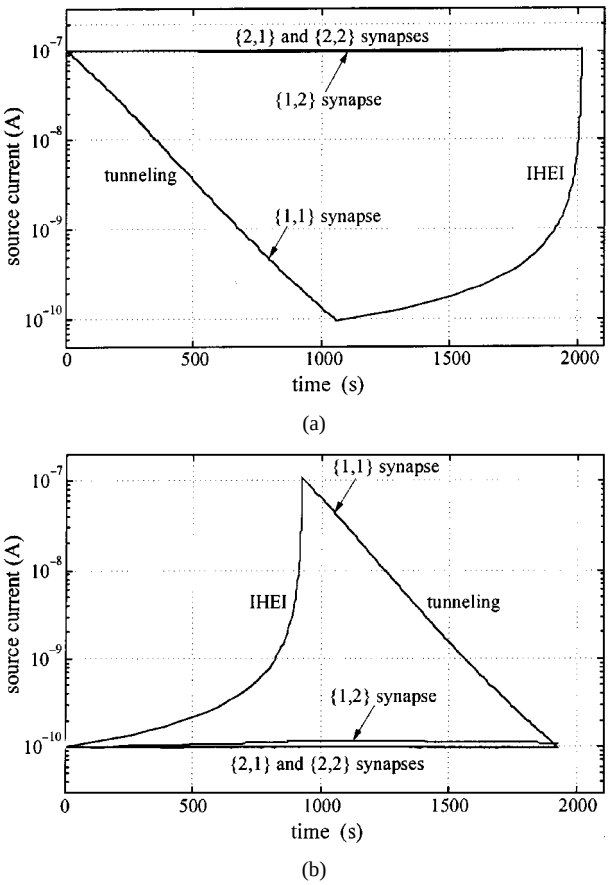


Fig. 6. Synapse isolation in the array of Fig. 5, fabricated in a 2 μm CMOS process. (a) Tunneling down, then injecting backup. We first initialized all four synapses to $I_s = 100$ nA. We tunneled synapse {1,1} down to 100 pA, then injected it back up to 100 nA, while measuring the source currents of the other three synapses. Crosstalk to the {1,2} synapse, defined as the fractional change in the {1, 2} synapse's source current divided by the fractional change in the {1,1} synapse's source current, was 0.004% during tunneling and was 0.005% during injection. (b) Injecting up, then tunneling back down. We first initialized all four synapses to $I_s = 100$ pA. We injected the {1,1} synapse up to 100 nA, then injected it back down to 100 pA. Crosstalk to the {1,2} synapse was 0.016% during injecting and 0.007% during tunneling. In both experiments, the crosstalk to the row 2 synapses was negligible.

be the fractional change in a deselected synapse's source current divided by the fractional change in the selected synapse's source current. The data in Fig. 6 show that the

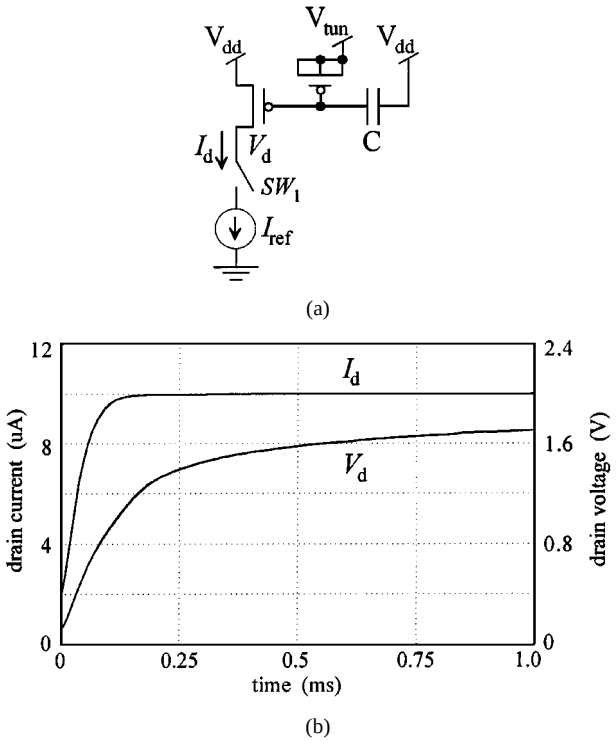


Fig. 7. Self-convergent memory writes. (a) Circuit. (b) SPICE simulation showing the pFET's drain voltage $V_{\rm d}$ and drain current $I_{\rm d}$ during a write. We first tunnel electrons off the floating gate so $I_{\rm d} < I_{\rm ref}$ (not shown in the simulation), turn off tunneling, then begin writing. We close switch SW_1 at $t = 0$, causing $V_{\rm d}$ to drop, electrons to inject onto the floating gate, and $I_{\rm d}$ to rise. As $I_{\rm d}$ approaches $I_{\rm ref}$, $V_{\rm d}$ rises, turning off the injection. $I_{\rm d}$ reaches 99% of its final value in 140 μs. We read the memory by applying $V_{\rm d} = 1.7$ V and measuring $I_{\rm d}$, with an accuracy that depends on the circuit details but can be better than 1%. Simulation parameters were $V_{\rm dd} = 6$ V, $C = 5$ fF, $I_{\rm ref} = 10$ μA.

crosstalk between selected and deselected synapses is less than 0.01% during tunneling and is less than 0.02% during IHEI. The reason for the good isolation can be seen from (5) and from the data in Figs. 3 and 4: both tunneling and IHEI are steep exponentials. Consequently, we can store precise analog values in a synaptic array without significant degradation due to crosstalk.

B. Self-Convergent Memory Writes

Because synapse transistors allow simultaneous memory reading and writing, we can use negative feedback to store accurate memory values. As an example, Fig. 7 shows a self-convergent memory write. We store the memory values as drain current $I_{\rm d}$. The write process works as follows: assume that, initially, $I_{\rm d}$ is smaller in magnitude than the programming current $I_{\rm ref}$. To write, we apply $I_{\rm ref}$ using switch SW_1. As long as $I_{\rm ref}$ exceeds $I_{\rm d}$, the synapse's drain voltage will be held low, causing electrons to inject onto the floating gate and thereby increasing $I_{\rm d}$. As $I_{\rm d}$ approaches $I_{\rm ref}$, the synapse's drain voltage will rise, turning off the injection. IHEI closes a negative feedback loop around the inverting amplifier formed by the pFET and the $I_{\rm ref}$ current source. This intrinsic feedback mechanism adapts the floating-gate charge to equalize the programming and pFET-drain currents, storing $I_{\rm ref}$ in the synapse transistor.

Notice that the synapse in Fig. 7 (comprising the two pFETs and the gate capacitor) is identical to an array element in Fig. 5. Consequently, we can use self-convergent mechanisms to write array synapses, by placing switches and current sources in the row-drain wires and by using the column-gate wires to select a column for writing. We monitor the row-drain voltages using sense amplifiers and open each switch to stop the write when its corresponding drain rises to a predetermined voltage. To read a column, we lower the appropriate column-gate wire and read the drain currents of all the transistors in the column.

V. Implications for SOC

Developing single-chip solutions to mixed-signal problems poses difficult engineering challenges. Contemporary integrated circuits (ICs) such as a scanner-on-a-chip or a Bluetooth transceiver-on-a-chip require integrating digital, analog, radio frequency (RF), and, often, nonvolatile memory (NVM) on a single die. These systems need large amounts of digital logic, necessitating fabrication in deep-submicrometer digital CMOS. And therein lies the problem. The low supply voltages, poor transistor matching, and absence of linear capacitors and resistors make analog and RF design in digital CMOS difficult. Furthermore, digital CMOS with embedded NVM typically lags two process generations behind digital CMOS without NVM. Contemporary goals such as a cellular transceiver-on-a-chip require huge amounts of digital logic, RF with 100-dB dynamic range, 16-bit analog baseband with 5–50 MHz bandwidth [19], and, ideally, NVM.

Synapse transistors afford significant benefit to SOC design. We have used them to store direct currents and voltages, match multiple current sources to a common reference, set operating points for capacitive-feedback operational amplifiers, balance mixers for improved image rejection, and store nonvolatile memories. By using onchip feedback to slowly and carefully adjust a synapse transistor's floating-gate charge, we can trim analog circuits to 16-bit accuracy [20]. The possibilities appear limitless. Consequently, rather than trying to describe all the possibilities here, we will instead illustrate with three examples some of the benefits of this technology. These examples are: 1) a mixed-signal finite-impulse response (FIR) filter; 2) a precision DAC; and 3) an autonulling amplifier.

A. Mixed-Signal FIR Filter

FIR filters are standard building blocks in signal-processing systems. Although digital-in/digital-out filters are the norm, analog-in/digital-out and digital-in/analog-out filters are not uncommon. One example of the latter is a pulse-shaping filter used to limit out-of-band spectral energy in communications systems [21]. The typical approach uses a digital signal processor (DSP)-based FIR filter followed by a DAC.

DSP chips are reconfigurable and easy to use, but tend to be large and power hungry. Applications that require both high throughput and low power use full-custom digital

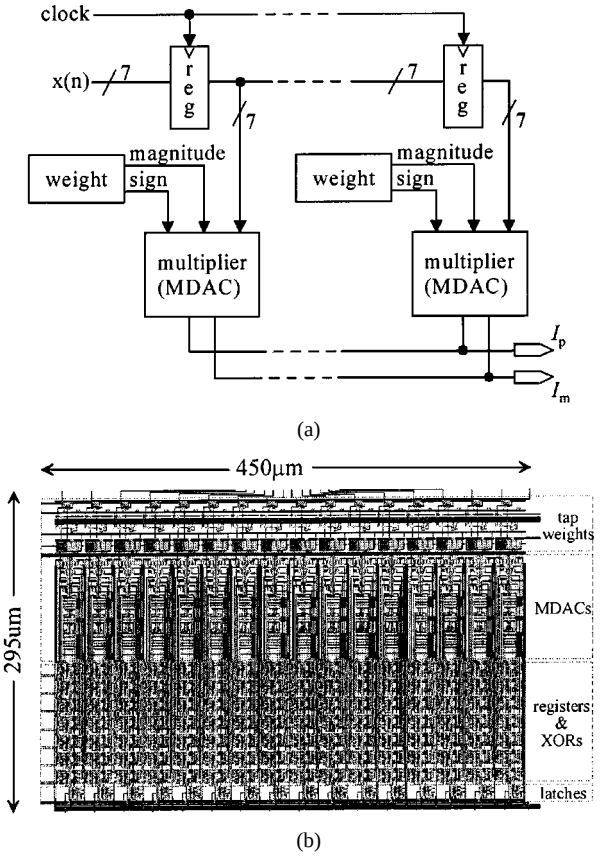


Fig. 8. (a) FIR filter architecture. We use a 7-bit digital delay line and store the tap weights on synapse-based analog memory cells. Multipliers are differential MDACs. Chip output is a differential current, comprising the sum of the currents from the 16 MDACs. (b) Filter layout. Digital components account for 48% of the die area, MDACs 35%, and analog memory cells 17%.

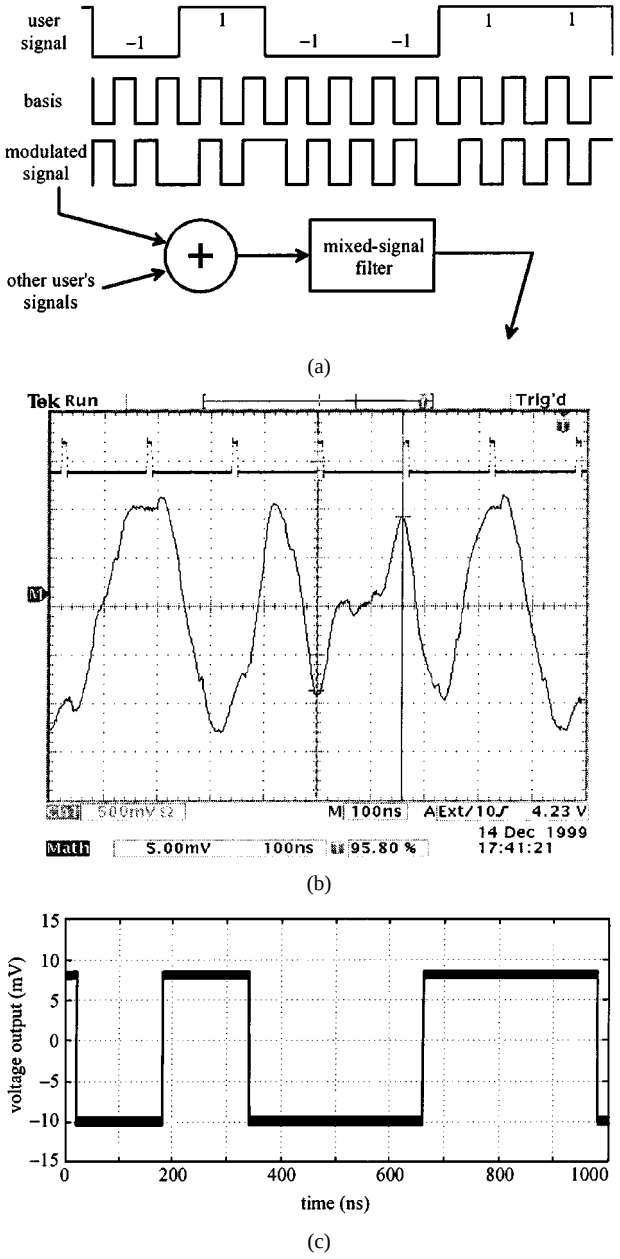


Fig. 9. FIR filter in a DS-CDMA despreading application. We applied as input a 100-Mb/s CDMA-like input, comprising two bitstreams encoded using orthogonal bases. We set the tap weights to decode the indicated basis. (a) Input bitstream and the basis we used to encode it. (b) Filter output and the strobe pulse we used to recover the data. (c) Reconstructed data for 64 (superimposed) experiments, showing reconstruction noise.

VLSI. However, even custom digital cannot overcome the area and power cost associated with an FIR filter's multipliers and adders [22]. Furthermore, systems with analog outputs still must integrate a precision DAC on the same chip as the digital circuitry. All-analog solutions are rarely viable because implementing analog delay lines and storing analog tap weights is difficult.

Although we cannot yet implement analog delay lines using synapse transistors, we can store analog tap values. We have developed a digital-in/analog-out FIR filter that comprises a 16-tap digital delay line, 16 synapse transistors to store the 16 (analog) tap coefficients and 16 multiplying DACs (MDACs) to multiply the digital data by the tap coefficients. The chip consumes 3 mW from a 3.3-V supply at a 225-MHz clock rate and occupies 0.13 mm^2 of die area in 0.35-μm CMOS. Although the present chip employs uncalibrated 7-bit MDACs, for precision applications we can substitute the 14-bit synapse-calibrated DAC described in Section V-B. Similarly, although the present chip uses 16 taps, we can easily scale the approach to 64 or more taps.

Fig. 8(a) shows the filter architecture and Fig. 8(b) shows the chip layout. A 7-bit 16-tap digital delay line shifts the input signal across the filter. We store each tap coefficient's magnitude in a synapse-based memory cell that we can individually erase and write. We store each tap coefficient's sign bit in a digital latch. Each MDAC generates a differential current; we sum the currents from the 16 MDACs to create a differential current output for the entire chip. For more details, see [23].

To verify performance, we tested the filter in a simple direct-sequence code-division multiple-access (DS-CDMA) despreading application [24]. We encoded two user bitstreams with orthogonal signatures and added them to form a combined signal at a 100-Mb/s chip rate. We programmed the synapse tap coefficients with one of the users' signatures and used the filter to recover the original bitstream. Fig. 9 shows the experimental results. From the measured

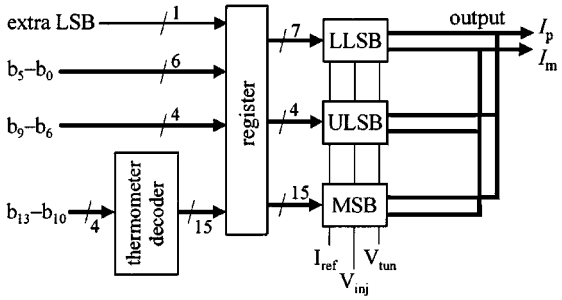


Fig. 10. DAC block diagram. A digital register latches the input. Ten LSBs are binary decoded; four MSBs are thermometer decoded. Extra LSB input is for trimming. LLSB segment is an untrimmable current-source array; the ULSB and MSB segments are trimmable arrays. Output is a differential current. We generate the injection and tunneling voltages offchip; future designs will generate these voltages using onchip charge pumps.

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-bit resolution of the input data stream.

B. Precision Digital-to-Analog Converter

Many applications, including those for emerging communications systems, require DACs with sample rates in the tens to hundreds of megasamples per second, at resolutions of 10–18 bits [19]. SOC integration poses the additional constraint of compatibility with standard digital CMOS. Current-steering DACs [25] are ideal for these applications because they are fast and can drive an output load without a voltage buffer. Their linearity, however, is limited by mismatch in the current-source transistors.

To reduce mismatch, DAC designers use large transistors, randomized layout, laser trimming, continuous electrical trimming, or other approaches [5], [26], [27]. These techniques increase die area and power dissipation substantially. What DAC designers need is a small nonvolatile electrically trimmable current source. Synapse transistors fit the bill perfectly.

Our 14-bit DAC, fabricated in a 0.25-μm digital CMOS process, uses synapse transistors to trim its current sources. Fig. 10 shows a block diagram. The DAC die area is 0.17 mm^2; the calibration circuitry occupies less than 10% of this area. Because synapse transistors store nonvolatile analog trim values, we do not need large current-source transistors or continuous trimming. To ensure calibration, we force the DAC to periodically self-trim during idle states or during powerup.

The DAC architecture is segmented, comprising six binary-decoded lower least significant bits (LLSBs), 4 binary-decoded upper least significant bits (ULSBs) and four thermometer-decoded most significant bits (MSBs). The six lower bits (LLSB segment) rely on intrinsic matching and are untrimmable; the eight upper bits (ULSB and MSB segments) employ synapse transistors for calibration. The digital circuitry comprises a static register to latch the 14-bit input word. The lower ten flip-flops drive the binary-weighted current-source arrays (six LLSB and four ULSB); the upper 4 bits are thermometer decoded to drive 15 identical current sources.

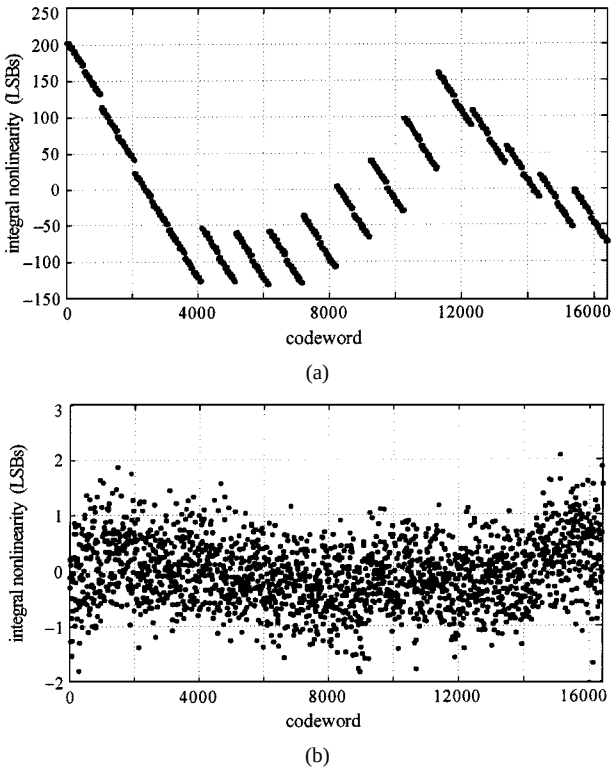


Fig. 11. (a) DAC INL before and (b) after trimming. We trimmed the ULSBs and MSBs to 0.5 LSB INL; the distribution in the posttrim data is due to noise in the circuit and measurement setup and to mismatch in an untrimmable LLSB bit.

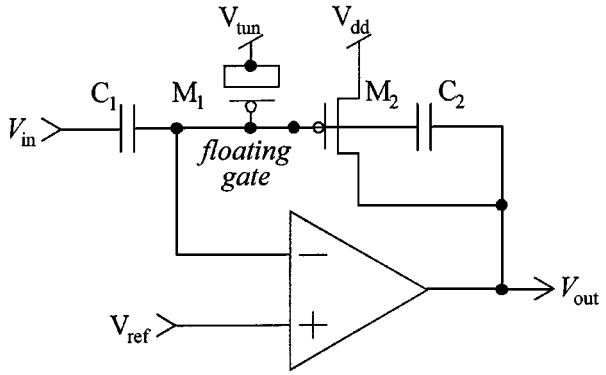
An offchip state machine controls IHEI and tunneling in the present chip; future designs will incorporate this state machine onchip. As we show in Fig. 11, trimming improves static DAC linearity by roughly two orders of magnitude. Because we trim the current sources, we avoid large current-source transistors and their large parasitic capacitances. Consequently, our DAC runs at 100 MSPS with a $-$ 10-dBm differential output, dissipating only 11 mW from a 3.3-V supply. For more details, see [28].

C. Autonulling Amplifier

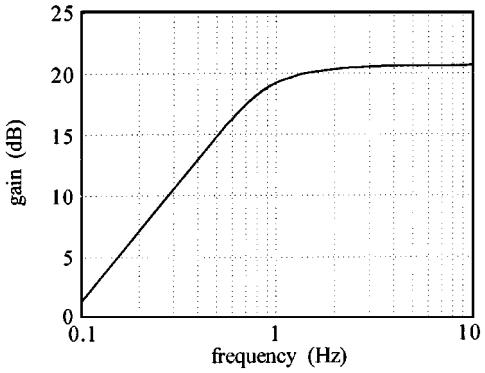
Most digital CMOS processes lack high-value linear resistors. For this and other reasons, CMOS designers tend to build operational amplifiers using switched-capacitor techniques.[1] We present an alternative continuous-time approach that uses a synapse transistor, rather than switches and capacitors, to set a direct current (dc) value on a floating node. We show the concept in Fig. 12.

Capacitors C_1 and C_2 set the op-amp's closed-loop gain. We apply a fixed high voltage to $V_{\rm tun}$, causing a small electron current (typically femtoamps) to tunnel off the floating gate. The op-amp compensates by lowering $V_{\rm out}$, causing transistor M_2 to inject electrons back onto the floating gate. $V_{\rm out}$ stabilizes when the tunneling and injection currents are

[1]Low-noise and wideband amplifiers that employ low-value resistors are counterexamples to this general claim.



(a)



(b)

Fig. 12. (a) Autonulling amplifier. Frequency response is high pass, with the low-frequency corner set by $V_{\rm tun}$. (b) Output amplitude versus frequency. Reasonable corner frequencies range from about a microhertz to about a kilohertz. Data show a 1-Hz corner.

equal and opposite and the floating-gate voltage is equal (in a dc sense) to $V_{\rm ref}$. $V_{\rm tun}$ sets both the quiescent value of the op-amp's output and its adaptation rate: if we raise $V_{\rm tun}$, we lower $V_{\rm out}$ and increase the adaptation rate. We describe the low-frequency response using the term "adaptation" rather than "time constant" because tunneling and injection are nonlinear processes, so the adaptation does not follow typical time-constant dynamics [29]. Other researchers have used this autozeroing concept in circuits such as filters [30] and silicon retinas [31].

VI. Implications for Silicon Learning

Because synapse transistors adapt locally and in parallel, we believe they can provide huge performance gains in artificial neural networks and machine learning. These gains arise from building systems in hardware rather than in software and from analog computation using innate features of the silicon-oxide physics rather than digital computation using transistors as switches. We do *not* claim that synapse transistors enable systems we could not build in software or in digital VLSI. Nonetheless, we envision extraordinary applications of synapse transistors to silicon learning. We demonstrate the approach using two example circuits, which implement key neural-network and machine-learning algorithms: 1) competitive learning and 2) correlational learning.

A. Competitive-Learning Circuit

Competitive learning comprises a class of algorithms that are useful for training classification and clustering networks [32]. In a typical competitive-learning neural network, a neuron's synaptic weight vector represents a set of related data points. Upon receiving an input, the neuron adapts, decreasing the difference between its weight vector and the input based on the following rule:

$$\Delta\boldsymbol{\mu}_i = \rho \times \sigma\left(N_i\right) \times \left(\mathbf{x} - \boldsymbol{\mu}_i\right) \tag{6}$$

where $\boldsymbol{\mu}_i$ is the weight vector of the ith neuron, ρ is the learning rate, $\sigma(N_i)$ is the activation of the ith neuron, and $\mathbf{x}$ is the input vector (boldface variables represent vectors or matrices). A given neuron's activation depends on the similarity between the input and that neuron's synaptic weights. A neuron's activation can be inhibited by other neurons; hence, neurons compete for inputs.

Different kinds of inhibition lead to different network learning rules. One example of an activation function is a hard winner-take-all (WTA) [33], where $\sigma(N_i) = 1$ if $\boldsymbol{\mu}_i$ is the weight vector most similar to the input and is zero otherwise. The hard WTA leads to the most basic form of competitive learning in which the neuron most similar to the input updates its weight vector according to the rule

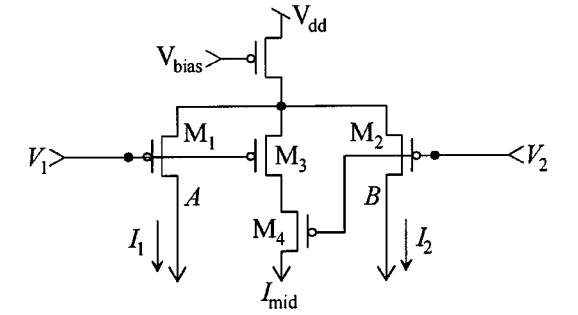
$$\Delta\boldsymbol{\mu} = \rho \times \left(\mathbf{x} - \boldsymbol{\mu}\right) \tag{7}$$

and the other neurons do not adapt. A soft WTA [34], [35] leads to an online version of maximum-likelihood competitive learning [36]. Imposing topological constrains on the inhibition leads to learning rules that are appropriate for self-organizing feature maps [37].

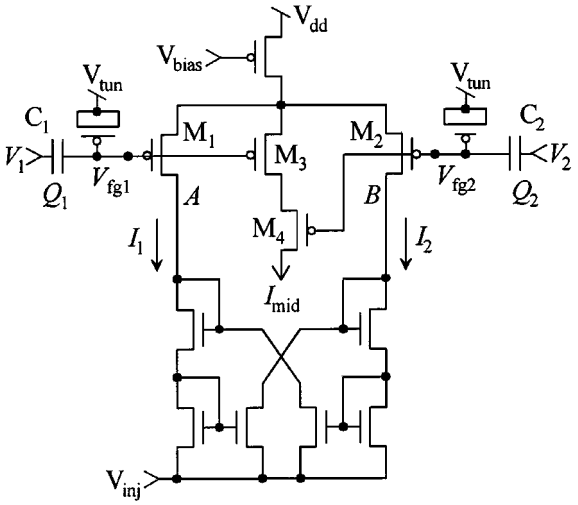
Synapses in competitive-learning networks adapt to increase the similarity between their weight vector $\boldsymbol{\mu}_i$ and the input $\mathbf{x}$. In Fig. 13 we show a circuit, termed an *automaximizing bump circuit* for reasons that will become clear shortly, which exhibits this behavior and can implement a variety of clustering and classification networks based on competitive learning.

The automaximizing bump circuit is an adaptive version of a classic circuit called the bump-antibump [see Fig. 13(a)] [38]. The classic bump computes a similarity and dissimilarity between two input voltages V_1 and V_2 and generates three output currents. The center current $I_{\rm mid}$ is a measure of the similarity between the inputs; the outside currents I_1 and I_2 are a measure of the dissimilarity. In the classic bump, $I_{\rm mid}$ is large if $V_1 \cong V_2$ and approximates a Gaussian centered at $V_1 - V_2 = 0$. The term "bump" comes from $I_{\rm mid}$'s Gaussian-like response. I_1 or I_2 are large when $V_1 \gg V_2$ or $V_2 \gg V_1$, respectively.

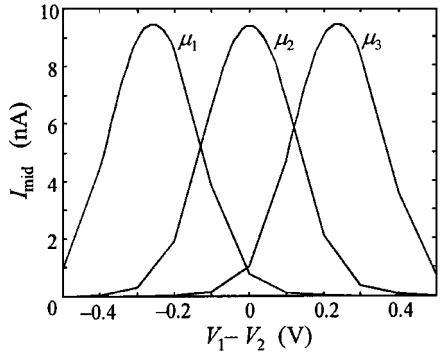
We revise the classic bump by replacing M_1/M_3 and M_2/M_4 with synapses and adding cross-coupled n-channel MOSFET (nFET) current mirrors [see Fig. 13(b)]. In the revised bump, $I_{\rm mid}$ is large if $V_{\rm fg1} \cong V_{\rm fg2}$, where $V_{\rm fg1} = V_1 + Q_1/C_1$ and $V_{\rm fg2} = V_2 + Q_2/C_2$ (Q_1 and Q_2 are the charge on floating gates 1 and 2, respectively). $I_{\rm mid}$ computes a similarity between the differential input $V_1 - V_2$ and a stored weight μ, where $\mu = Q_2/C_2 - Q_1/C_1$. If



(a)



(b)



(c)

Fig. 13. (a) Classic bump–antibump circuit. (b) Automaximizing bump circuit. To enable adaptation, we set $V_{\rm tun} \sim 10$ V and $V_{\rm inj} \sim 0$ V. To disable enable adaptation, we set $V_{\rm tun} < 8$ V and $V_{\rm inj} > 2$ V (these values are typical for a 0.35 μm process). (c) Measured $I_{\rm mid}$ versus $V_1 - V_2$ for three stored weights μ_1, μ_2, and μ_3. Stored weight changes the location of the bump peak, but not its shape. Circuit allows a wide range of adaptation rates, depending on $V_{\rm tun}$ and $V_{\rm inj}$.

the mean value of $V_1 - V_2$ is a nonzero real number μ, the circuit shifts $I_{\rm mid}$'s peak away from $V_1 - V_2 = 0$ and toward $V_1 - V_2 = \mu$, centering $I_{\rm mid}$'s Gaussian-like response around μ. We interpret this shift as a weight μ stored by the circuit and interpret $I_{\rm mid}$ as a similarity between the differential input $V_1 - V_2$ and the stored weight μ. We say that the circuit "learns" μ.

Tunneling and IHEI together adjust $I_{\rm mid}$ to decrease the difference between μ and the mean value of the input $V_1 - V_2$. A high $V_{\rm tun}$ causes symmetric tunneling from both floating gates. A low $V_{\rm inj}$ causes selective IHEI at either M_1 or M_2, depending on the relative values of I_1 and I_2. If $I_1 > I_2$, the nFET current mirrors pull node B low and let node A rise, causing IHEI at M_2 and thereby increasing I_2. If $I_2 > I_1$, the nFET current mirrors pull node A low and let node B rise, causing IHEI at M_1 and thereby increasing I_1. The circuit adapts both Q_1 and Q_2 to equalize I_1 and I_2, thereby equalizing $V_{\rm fg1}$ and $V_{\rm fg2}$ (for any input $V_1 - V_2$). The circuit is *automaximizing* because $V_{\rm tun}$ and $V_{\rm inj}$ tell the circuit when to adapt, but the circuit itself decides the direction and magnitude of the adaptation. For more details on this circuit see [39].

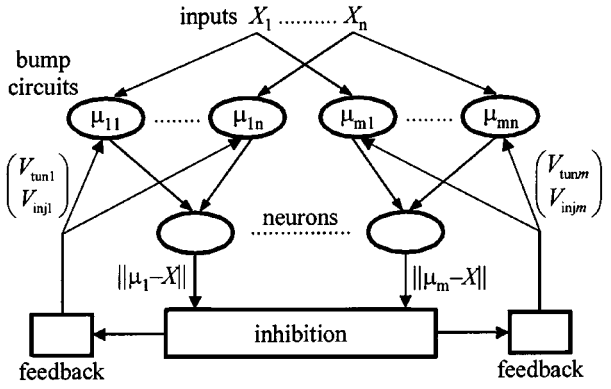
Although $I_{\rm mid}$ does not tell us whether $V_1 > V_2$ or $V_2 > V_1$, when we are computing the distance between the input and μ, we are typically concerned with magnitude, not with direction. Direction is important for computing weight updates and we can use the antibump outputs to provide a direction metric.

Fig. 14 illustrates onchip learning using the automaximizing bump. Fig. 14(a) shows a general architecture for competitive learning. Each neuron is a cluster center in an n-dimensional input space, with a bump circuit (a weight) for each dimension of that space. Each bump computes the similarity between the component of the input in that dimension and its stored weight. The inhibitory circuit decides the extent to which each neuron adapts to the input. Fig. 14(b) shows a one-dimensional (1-D) circuit implementation comprising two bump circuits with a common input. Each bump representing a 1-D cluster and each computes a similarity to the input. A WTA decides which bump is closest to the input (i.e., which bump has the larger $I_{\rm mid}$) and onchip feedback generates $V_{\rm tun}$ and $V_{\rm inj}$ signals to update the selected bump.

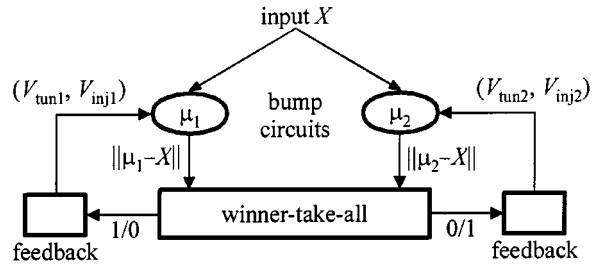
Fig. 14(c) shows measured test data from the circuit in Fig. 14(b). We applied as input a mixture of two Gaussians; the circuit autonomously unmixed them by adapting a bump to the mean of each of the Gaussians. For comparison, we show the output of a software learning network in which the neuron whose weight vector was closest to the input updated its weight using the learning rule of (7). We applied identical inputs to the simulated network and to the chip. The data show clearly that the chip learns to cluster its inputs.

B. Correlational-Learning Circuit

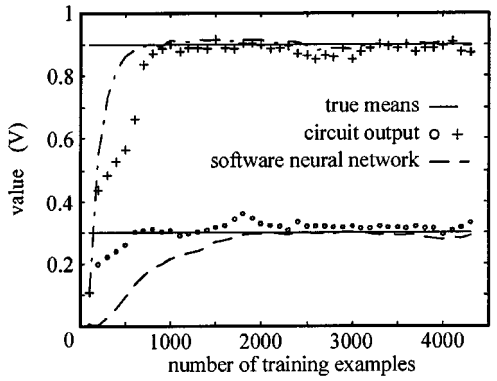
Learning correlations or conditional probabilities between pairs of variables underlies a wide variety of machine-learning algorithms. Outer-product rules (with weight decay) learn correlations between synaptic inputs and feedback signals. Maximum-likelihood classifiers base their classification on conditional probabilities between inputs and class labels. In sequence-learning models, such as that proposed by Levy [40], synapses store the log-conditional probability that a presynaptic spike occurred given that the postsynaptic neuron spiked at a later time. Other algorithms, such as temporal-difference learning [41] and temporally asymmetric Hebbian learning [42], are based on time-separated correlations. Consequently, a silicon circuit



(a)



(b)

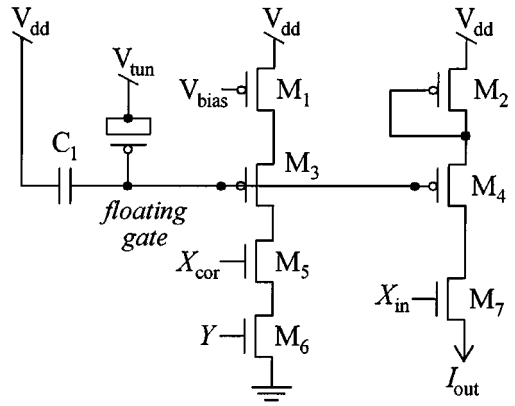


(c)

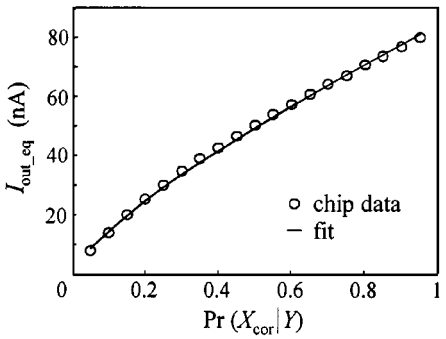
Fig. 14. (a) Architecture for competitive learning in n dimensions. Each neuron represents a cluster, with a bump circuit for each input and a single feedback block for the entire neuron (all bumps in a neuron have common $V_{\rm tun}$ and $V_{\rm inj}$). (b) Competitive-learning network comprising two bump circuits, a WTA and feedback that derives $V_{\rm inj}$ and $V_{\rm tun}$ from the WTA output. (c) Comparison between a chip that implements the network from (b) and an equivalent software network that implements the learning rule from (7). We drew the input data from a mixture of two Gaussians, with 80% of the data drawn from the higher-mean Gaussian. We plot the means learned by the circuit and the software simulation, compared to the true means of the Gaussians. We set the learning rate ρ in (7) to 0.01.

that learns correlations and conditional probabilities can implement a wide variety of learning algorithms.

We have developed a 0.35-μm CMOS circuit that can learn either a correlation or a conditional probability between binary-valued input and feedback signals. Our circuit builds upon previous work [15], [43], [44] and extends it in several ways. First, the circuit implements a general learning principle rather than a particular learning rule. Second, it can implement learning rules that correlate temporally separate pre- and postsynaptic activity. Third, it employs self-calibration



(a)



(b)

Fig. 15. (a) Correlational-learning circuit. (b) Equilibrium current $I_{\rm out\_eq}$ versus the conditional probability $P(X_{\rm cor}|Y)$, including a fit from (10) with $\alpha = 0.7664$.

to remove mismatch between the learning rates of individual synapses.

We show the circuit in Fig. 15(a). The circuit stores an analog current $I_{\rm out}$, multiplies $I_{\rm out}$ by a binary input $X_{\rm in}$, and adapts $I_{\rm out}$ to form either a conditional probability $P(X_{\rm cor}|Y)$ or a correlation $P(X_{\rm cor}, Y)$. $X_{\rm in}$ is analogous to a presynaptic input, $X_{\rm cor}$ is a presynaptic adaptation signal that typically has some relationship with $X_{\rm in}$, Y is analogous to a postsynaptic signal or error feedback, and $I_{\rm out}$ is the circuit's weight. We can implement different learning rules by altering the relationship between $X_{\rm cor}$, $X_{\rm in}$ and Y.

Transistor M_4 sources the output current $I_{\rm out}$. Because M_4's control gate is tied to $V_{\rm dd}$, $I_{\rm out}$ depends solely on capacitor C_1's stored charge. We adjust M_4's drain voltage to prevent IHEI. A second floating-gate transistor M_3, whose gate is connected to C_1, enables IHEI. Simultaneously high input signals $X_{\rm cor}$ and Y pull M_3's drain low, injecting electrons onto the floating gate and increasing $I_{\rm out}$. A high $V_{\rm tun}$ tunnels electrons off the floating gate, decreasing $I_{\rm out}$. We switch $I_{\rm out}$ on or off using transistor M_7; this switching corresponds to multiplying the output $I_{\rm out}$ (the circuit's weight) by a binary input signal $X_{\rm in}$.

We describe qualitatively how our circuit learns correlations or conditional probabilities using an approach similar to that in [43]. We begin by defining the circuit's equilibrium output $I_{\rm out\_eq}$ as that value of $I_{\rm out}$ for which the expected values of the tunneling and IHEI currents are equal

and opposite. We show that $I_{\text{out\_eq}}$ comprises a correlation or a conditional probability over the statistics of X_{cor} and Y, the circuit learns $I_{\text{out\_eq}}$, and, consequently, the circuit learns a correlation or conditional probability that depends on X_{cor} and Y.

We first describe IHEI and tunneling in terms of I_{out}, X_{cor}, and Y. Consider IHEI. A joint binary event (X_{cor}, Y) closes nFET switches M_5 and M_6, pulling M_3's drain to ground and injecting electrons onto the floating gate. The IHEI probability is the probability of the joint event (X_{cor}, Y). Looking now at tunneling, if we hold V_{tun} high during adaptation, the tunneling current will be approximately constant. If we instead use Y to gate a high-voltage signal onto the tunneling implant, then tunneling will depend on the probability of the event Y.

Consider the latter case in which tunneling depends on $P(Y)$. The injection and tunneling gate currents have power-law relationships to I_{out} and are given by [45]

$$I_{\text{g\_inj}} = \lambda I_{\text{out}}^{-\phi} P(X_{\text{cor}}, Y) \tag{8}$$

$$I_{\text{g\_tun}} = \eta I_{\text{out}}^{\upsilon} P(Y) \tag{9}$$

where λ, η, ϕ, and ν are constants that derive from the IHEI and tunneling processes. We solve for $I_{\text{out\_eq}}$ by finding a value for I_{out} that equalizes $I_{\text{g\_inj}}$ and $I_{\text{g\_tun}}$

$$I_{\text{out\_eq}} = I_0 \left(\frac{\lambda}{\eta} P(X_{\text{cor}}|Y) \right)^{\alpha} \tag{10}$$

where $\alpha = 1/(\nu + \phi)$ and I_0 is a preexponential current. When tunneling depends on $P(Y)$, the equilibrium output approximates $P(X_{\text{cor}}|Y)$.

If we hold V_{tun} high rather than gating it with Y, the $P(X_{\text{cor}}|Y)$ in (10) changes to $P(X_{\text{cor}}, Y)$. The reason is that tunneling becomes a quasiconstant leak off the floating gate, but IHEI still depends on $P(X_{\text{cor}}, Y)$, so $I_{\text{out\_eq}}$ depends only on $P(X_{\text{cor}}, Y)$. In this case, the circuit learns a correlation.

M_1 forces a constant current through injecting transistor M_3, so adaptation naturally moves I_{out} toward $I_{\text{out\_eq}}$. If $I_{\text{out}} > I_{\text{out\_eq}}$, the tunneling gate current will exceed M_3's IHEI gate current, decreasing I_{out}. If $I_{\text{out}} < I_{\text{out\_eq}}$, the IHEI current will exceed the tunneling current, increasing I_{out}. Because adaptation naturally moves I_{out} toward $I_{\text{out\_eq}}$, the circuit learns the approximation to $P(X_{\text{cor}}|Y)$ given by (10). Fig. 15 shows measured $I_{\text{out\_eq}}$ versus the conditional $P(X_{\text{cor}}|Y)$, including a fit from (10). Although (10) only approximates a conditional probability, simulations show that the mild nonlinearity does not degrade the performance of typical learning networks appreciably.

1) Calibrated Correlational-Learning Circuit: Fabrication-induced mismatch among nominally identical synapse transistors can prevent learning circuits from learning meaningful values. Matching data, from synapses fabricated in a 0.35-μm process, show that the oxide currents due to IHEI vary by roughly 2:1 across a chip and those due to tunneling by 1.5:1. We can model the effect as mismatch in the coefficients I_{tn} and β of (3) and (4). To illustrate the problem, Fig. 16(b) shows equilibrium outputs from an array of six

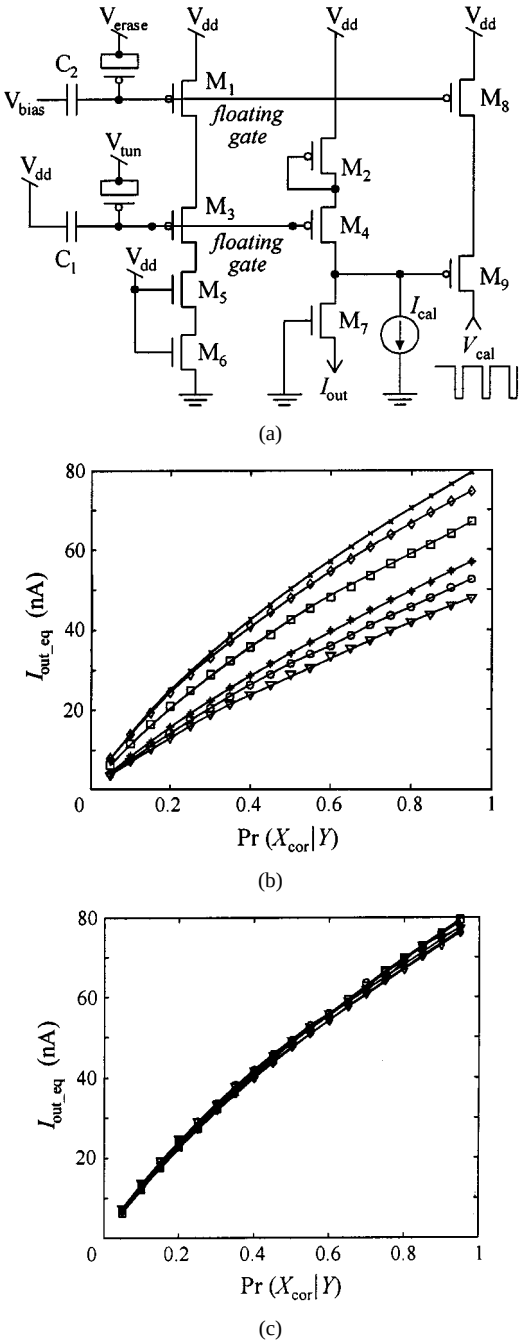


Fig. 16. (a) Schematic of calibrated correlational-learning circuit, with inputs that we use for calibration. (b) Equilibrium outputs for an array of circuits identical to that in Fig. 15. (c) Postcalibration equilibrium outputs for an array of circuits from (a).

correlational-learning circuits, all exposed to identical inputs. The mismatch-induced variation in the outputs is of the same order of magnitude as that due to the input data, rendering the array all but useless. Other researchers have noted this same problem in other floating-gate systems [31], [44].

In our correlational-learning circuit, mismatch in β and I_{tn} leads to mismatched λ/η in (10). We solve this problem using synapse transistors to match λ/η across multiple circuits. Notice from (4) that IHEI varies linearly with a synapse's source current I_{s}. By altering M_3's source current, we can alter λ/η, thereby matching $I_{\text{out\_eq}}$ across

multiple circuits. We illustrate the approach using an array of correlational-learning circuits and calibrate the circuits using a variant of the self-convergent write mechanism from Section IV-B.

We show the modified correlational-learning circuit in Fig. 16(a) and postcalibration matching data in Fig. 16(c). The primary change from the circuit in Fig. 15 is that we converted M_1 into a synapse transistor, allowing us to adjust M_3's source current. We use self-convergence to force $I_{\text{out\_eq}}$ to match a global calibration current I_{cal}, with $P(X_{\text{cor}}|Y) = 1$. If I_{cal} is larger in magnitude than $I_{\text{out\_eq}}$ (i.e., if I_{cal} exceeds M_4's drain current), then switch M_9 will be turned on and active-low calibration pulses will cause IHEI at transistor M_8. When $I_{\text{out\_eq}}$ exceeds I_{cal}, M_9 will turn off, disabling IHEI.

In practice, we first erase the floating gates of all transistors M_1 using tunneling to V_{erase}. We then turn off V_{erase}, enable I_{cal} and V_{tun}, apply a pulse train V_{cal}, and wait until the gates of all transistors M_9 rise. V_{cal} comprises narrow active-low pulses, giving M_3 time to adapt to the statistics of its inputs when V_{cal} is high and adapting M_1 quickly when V_{cal} is low. Fig. 16(c) demonstrates that we can reduce mismatch to a small fraction of the circuit's output.

VII. Technology Issues

Synapse transistors have technological and reliability issues similar to other NVM technologies, of which the most critical are tunneling- and injection-induced damage to the gate oxide, and charge leakage off the floating gate. Oxide damage limits the number of read/write cycles in digital flash memory and electrically erasable programmable read-only memory (EEPROM) [46]. Although synapse transistors are subject to the same damage mechanisms, their analog-valued weight updates are typically much slower and smaller than digital memory writes, so their oxide currents are three to six orders of magnitude smaller than in flash memories or EEPROMs. Consequently, in our experience, oxide damage has not been an issue, even for synapse-based circuits that use continuous tunneling and injection. Oxide trapping does decrease a synapse's weight-update rates, forcing us to regulate the tunneling and injection voltages. Synapse-based regulation circuits allow us to control these voltages precisely.

The scaling of gate oxides to less than about 70Å thickness causes floating gates to leak. This problem is not unique to synapse transistors—it affects all NVM devices that use floating gates. If anything, synapse transistors are far more tolerant of oxide leakage because, in most situations, we use them in circuits that adapt the stored charge on an ongoing basis. If, however, we must store a memory for years without updating, we use the 70Å oxide available in most dual gate-oxide CMOS processes.

VIII. Conclusion

We and other researchers are developing neurally inspired chips that compute and adapt using innate features of the silicon-MOS and silicon-oxide physics. Our rationale for this research is that we believe local adaptation can be instrumental in modern IC design. We have demonstrated, by a few simple circuits, how our synapse-based approach can advance SOC design and silicon learning. However, our discussion only hints at the broad promise of this technology. As digital circuits shrink to ever smaller feature sizes, the value in the analog portion, which does not scale well, continues to rise. Analog circuits that are simple to design because they self-tune, are small because the transistors themselves compensate for mismatch and degradations, and learn from their inputs can bypass many of the bottlenecks in contemporary IC design.

The circuits that we have built to date are small and simple, primarily because they represent our baby steps in exploring a new technology. As we and others learn how to use local adaptation effectively, the circuits we build will mature. Our confidence in this technology is rooted not in what we have built to date, but rather in the existence proof provided by neurobiology. Nature demonstrates good solutions to hard engineering problems, using a neural substrate that continuously adapts and learns and changes. We are merely copying Nature.

References

[1] P. Churchland and T. Sejnowski, *The Computational Brain*. Cambridge, MA: MIT Press, 1993.

[2] C. Mead, "Neuromorphic electronic systems," *Proc. IEEE*, vol. 78, pp. 1629–1636, Oct. 1990.

[3] G. Cauwenberghs and M. Bayoumi, *Learning in Silicon*, G. Cauwenberghs and M. Bayoumi, Eds. Norwell, MA: Kluwer, 1999.

[4] G. Barrionuevo, S. R. Kelso, D. Johnston, and T. H. Brown, "Conductance mechanism responsible for long-term potentiation in mono-synaptic and isolated excitatory synaptic inputs to hippocampus," *J. Neurophysiol.*, vol. 55, no. 3, pp. 540–550, 1986.

[5] A. R. Bugeja and S. Bang-Sup, "A self-trimming 14-b 100-MS/s CMOS DAC," *IEEE J. Solid-State Circuits*, vol. 35, pp. 1841–1852, Dec. 2000.

[6] P. Hasler, "Foundations of learning in analog VLSI," Ph.D. dissertation, Dept. Comput. Neural Syst., California Inst. Technol., Pasadena, CA, 1997.

[7] C. Diorio, P. Hasler, B. A. Minch, and C. Mead, "A Three-terminal silicon synaptic device," U.S. Patent 5 825 063, July 1998.

[8] ——, "Floating-gate MOS synapse transistors," in *Neuromorphic Systems Engineering: Neural Networks in Silicon*, T. S. Lande, Ed. Norwell, MA: Kluwer, 1998, pp. 315–337.

[9] ——, "A complementary pair of four-terminal silicon synapses," *Analog Integr. Circuits Signal Processing*, vol. 13, no. 1/2, pp. 153–166, 1997.

[10] C. Diorio, "Neurally inspired silicon learning: From synapse transistors to learning arrays," Ph.D. dissertation, Dept. Elect. Eng., California Inst. Technol., Pasadena, CA, 1997.

[11] C. Diorio, P. Hasler, B. A. Minch, and C. Mead, "A Semiconductor structure for long-term learning," U.S. Patent 5 627 392, May 1997.

[12] ——, "A single-transistor silicon synapse," *IEEE Trans. Electron Devices*, vol. 43, pp. 1972–1980, Nov. 1996.

[13] A. S. Grove, *Physics and Technology of Semiconductor Devices*. New York: Wiley, 1967.

[14] M. Lenzlinger and E. H. Snow, "Fowler-Nordheim tunneling into thermally grown $Sio_2 2$," *J. Appl. Phys.*, vol. 40, no. 6, pp. 278–283, 1969.

[15] C. Diorio, P. Hasler, B. A. Minch, and C. Mead, "A floating-gate MOS learning array with locally computed weight updates," *IEEE Trans. Electron Devices*, vol. 44, pp. 2281–2289, Dec. 1997.

[16] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.

[17] B. A. Minch and P. Hasler, "A floating-gate technology for digital CMOS processes," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 2, Orlando, FL, 1999, pp. 400–403.

[18] S. M. Sze, *Physics of Semiconductor Devices*. New York: Wiley, 1981.
[19] A. K. Salkintzis, H. Nie, and P. T. Mathiopoulos, "ADC and DSP challenges in the development of software radio base stations," *IEEE Pers. Commun.*, vol. 6, pp. 47–55, Aug. 1999.
[20] C. Diorio, S. Mahajan, P. Hasler, B. A. Minch, and C. Mead, "A high-resolution nonvolatile analog memory cell," in *Proc. IEEE Int. Symp. Circuits and Systems*, vol. 3, Seattle, WA, 1995, pp. 2233–2236.
[21] B. Sklar, *Digital Communications*. Englewood Cliffs, NJ: Prentice-Hall, 1988.
[22] N. Zhang, C. Teuscher, H. Lee, and B. Brodersen, "Architectural implementation issues in a wideband receiver using multiuser detection," in *Proc. 36th Allerton Conf. Communications, Control, and Computing*, Urbana-Champaign, IL, 1998, pp. 765–771.
[23] M. Figueroa, D. Hsu, and C. Diorio, "A mixed-signal approach to high-performance, low-power linear filters," *IEEE J. Solid-State Circuits*, vol. 36, pp. 816–822, June 2001.
[24] 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 Allerton Conf. Communications, Control, and Computing*, Urbana-Champaign, IL, 1996, pp. 623–632.
[25] 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.
[26] J. Bastos *et al.*, "A 12-bit intrinsic accuracy high-speed CMOS DAC," *IEEE J. Solid-State Circuits*, vol. 33, pp. 1959–1969, Dec. 1998.
[27] G. A. M. Van-Der-Plas, J. Vandenbussche, W. Sansen, M. S. J. Steyaert, and G. G. E. Gielen, "A 14-bit intrinsic accuracy Q^2 random walk CMOS DAC," *IEEE J. Solid-State Circuits*, vol. 34, pp. 1708–1718, Dec. 1999.
[28] M. Figueroa, J. Hyde, T. Humes, and C. Diorio, "A floating-gate trimmable high-resolution DAC in standard 0.25 μm CMOS," in *Proc. Nonvolatile Semiconductor Memory Workshop*, Monterey, CA, 2001, pp. 46–47.
[29] P. Hasler, B. A. Minch, and C. Diorio, "An autozeroing floating-gate amplifier," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 74–82, Jan. 2001.
[30] P. Hasler, T. Stanford, and B. A. Minch, "A second-order section built from autozeroing floating-gate amplifiers," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 116–120, Jan. 2001.
[31] A. Pesavento, T. Horiuchi, C. Diorio, and C. Koch, "Adaptation of current signals with floating-gate circuits," *Analog Integr. Circuits Signal Processing*, vol. 30, no. 2, pp. 137–147, Feb. 2002.
[32] M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995.
[33] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of O(n) complexity," in *Advances in Neural Information Processing Systems 1*, D. Touretzky, Ed. San Mateo, CA: Morgan Kauffman, 1989, pp. 703–711.
[34] I. M. Elfadel and J. L. Wyatt Jr., "The softmax nonlinearity: Derivation using statistical mechanics and useful properties as a multiterminal analog circuit element," in *Advances in Neural Information Processing Systems 6*, J. Cowan, G. Tesauro, and C. L. Giles, Eds. San Mateo, CA: Morgan Kaufmann, 1994, pp. 882–887.
[35] S.-C Liu, "A winner-take-all circuit with controllable soft max property," in *Advances in Neural Information Processing Systems 12*, S. A. Solla, T. K. Leen, and K. R. Muller, Eds. Cambridge, MA: MIT Press, 2000, pp. 717–723.
[36] S. J. Nowlan, "Maximum likelihood competitive learning," in *Advances in Neural Information Processing Systems 2*, D. Touretzky, Ed. San Mateo, CA: Morgan Kauffman, 1990, pp. 574–582.
[37] T. Kohonen, *Self Organizing Feature Maps*, 2nd ed. Berlin, Germany: Springer-Verlag, 1997.
[38] T. Delbruck, "Bump circuits for computing similarity and dissimilarity of analog voltages," California Inst. Technol., Pasadena, CA, CNS Memo 26, 1993.
[39] D. Hsu, M. Figueroa, and C. Diorio, "A silicon primitive for competitive learning," in *Advances in Neural Information Processing Systems 13*, T. K. Leen, T. Dietterich, and V. Tresp, Eds. Cambridge, MA: MIT Press, 2001, pp. 713–719.
[40] W. B. Levy, "A computational approach to hippocampal function," in *Computational Models of Learning in Simple Neural Systems, The Psychology of Learning and Motivation* , R. D. Hawkins and G. H. Bower, Eds. New York: Academic, 1989, vol. 23, pp. 243–305.
[41] R. Sutton, "Learning to predict by the method of temporal difference," *Mach. Learn.*, vol. 3, pp. 9–44, 1988.
[42] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs," *Science*, vol. 275, no. 5297, pp. 213–215, 1997.
[43] P. Hasler, B. A. Minch, J. Dugger, and C. Diorio, "Adaptive circuits and synapses using pFET floating-gate devices," in *Learning in Silicon*, G. Cauwenberghs and M. Bayoumi, Eds. Norwell, MA: Kluwer, 1999, pp. 33–65.
[44] P. Hafliger, "A Spike-Based Learning Rule and its Implementation in Analog Hardware," Ph.D. dissertation, ETH Zurich, Switzerland, 1999.
[45] A. Shon, D. Hsu, and C. Diorio, "Learning spike-based correlations and conditional probabilities in silicon," in Advances in Neural Information Processing Systems 14, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds. Cambridge, MA: MIT Press, 2002.
[46] S. Aritome, R. Shirota, G. Hemink, T. Endoh, and F. Masuoka, "Reliability issues of flash memory cells," *Proc. IEEE*, vol. 81, pp. 776–787, May 1993.

Chris Diorio (Member, IEEE) 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 Associate Professor of Computer Science and Engineering with the University of Washington, Seattle. He is a founder of Impinj, Inc. and 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 current research interests include building electronic circuits and systems that mimic the computational and organizational principles found in the nervous systems of living organizms.

Dr. Diorio received the Office of Naval Research Young Investigator Award in 2001, the Alfred P. Sloan Foundation Research Fellowship in 2000, the Presidential Early Career Award in Science and Engineering (PECASE) in 1999, the Packard Foundation Fellowship in 1998, the National Science Foundation CAREER Award in 1998, and the Electron Devices Society's Paul Rappaport Award in 1996.

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

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

Miguel Figueroa received the B.Sc. degree, an engineering degree, and M.S. degree in electrical engineering from the University of Concepción, Chile, in 1988, 1991, and 1997, respectively, and the M.S. degree in computer science from the University of Washington, Seattle, in 1999. He is currently working towards the Ph.D. degree at the same university.

His current research interests include mixed-signal VLSI, neurobiology-inspired computation, and reconfigurable architectures.