Open access • Journal Article • DOI:10.1109/TNN.2002.1000139

Competitive learning with floating-gate circuits — Source link

D. Hsu, Miguel Figueroa, Christopher J. Diorio

Institutions: University of Washington

Published on: 01 May 2002 - IEEE Transactions on Neural Networks (IEEE)

Topics: Competitive learning, Electronic circuit, Cluster analysis and Artificial neural network

Competitive Learning With Floating-Gate Circuits

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

***Abstract*—Competitive learning is a general technique for training clustering and classification networks. We have developed an 11-transistor silicon circuit, that we term an automaximizing bump circuit, that uses silicon physics to naturally implement a similarity computation, local adaptation, simultaneous adaptation and computation and nonvolatile storage. This circuit is an ideal building block for constructing competitive-learning networks. We illustrate the adaptive nature of the automaximizing bump in two ways. First, we demonstrate a silicon competitive-learning circuit that clusters one-dimensional (1-D) data. We then illustrate a general architecture based on the automaximizing bump circuit; we show the effectiveness of this architecture, via software simulation, on a general clustering task. We corroborate our analysis with experimental data from circuits fabricated in a 0.35-μm CMOS process.**

***Index Terms*—Analog very large scale integration (VLSI), competitive learning.**

I. Introduction

COMPETITIVE learning (Fig. 1) comprises a style of neural learning algorithms that has proved useful for training many classification and clustering networks [1]. In a competitive-learning network, a neuron's synaptic weight vector typically represents a set of related data points. Upon receiving an input, each neuron adapts, decreasing the difference between its weight vector and the input based on the following rule:

$$\Delta\boldsymbol{\mu}_i = \rho \times \boldsymbol{\sigma}(N_i) \times (\mathbf{x} - \boldsymbol{\mu}_i) \tag{1}$$

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 (we follow the convention that variables denoted in boldface correspond to vectors or matrices). The activation depends on the similarity between a neuron's synaptic weights and the input and can be inhibited by other neurons; hence neurons compete for input data. An example of $\sigma(N_i)$ is a hard winner-take-all (WTA) [2], where $\sigma(N_i) = 1$ if $\boldsymbol{\mu}_i$ is the weight vector most similar to the input, or zero otherwise.

Different kinds of inhibition lead to different learning rules. A hard WTA leads to the basic competitive learning rule where the most similar neuron updates its weight vector according to the rule

$$\Delta\boldsymbol{\mu} = \boldsymbol{\rho} \times (\mathbf{x} - \boldsymbol{\mu}) \tag{2}$$

Manuscript received July 31, 2000. This work was supported by the NSF under Grants BES 9720353 and ECS 9733425 and by a Packard Foundation Fellowship.

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

Publisher Item Identifier S 1045-9227(02)04434-X.

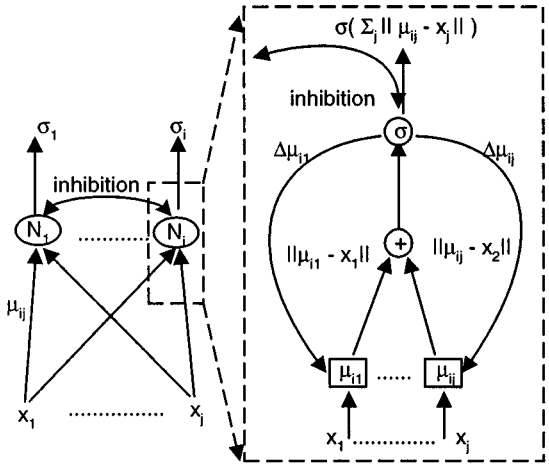

Fig. 1. A framework for competitive learning. Each neuron computes the difference between the input vector and the values stored in its synapses. Each synapse computes the distance between its input and a stored value. The neuron aggregates its synaptic outputs and updates its synaptic weights in an unsupervised fashion. The adaptation typically decreases the difference between the neuron's input and weight vector. Competition among neurons ensures that only neurons that are close to the input adapt.

and the other neurons do not adapt. A soft WTA [3], [4] leads to an online version of maximum likelihood competitive learning [5]. Imposing topological constraints on the inhibition leads to learning rules appropriate for self-organizing feature maps [6]. These learning routines can be used to train nearest neighbor style classifiers [7], [8], adaptive vector quantizers, ART networks [1], mixtures of experts and radial basis functions [9].

The synapses in a competitive-learning network typically follow a common adaptation dynamic, increasing the similarity between the synaptic weight vector and the present input. Consequently, a silicon synapse that exhibits this behavior can be combined with external circuitry to implement many neural learning algorithms.

Very large scale integration (VLSI) implementations of competitive-learning synapses have been reported in the literature [10]–[13]. These synapses typically use digital or capacitive weight storage. Digital storage is expensive in terms of die area and limits the precision of synaptic weight updates. Capacitive storage requires a refresh scheme to prevent weight decay. In addition, these implementations all require separate weight-update and computation phases, adding complexity to the control circuitry. More importantly, neural networks built with these synapses do not typically adapt during normal operation. A notable exception is the analog synapse designed by Fusi *et al.* [14], which integrates the capacitive refresh into the weight update dynamics. However, their synapse does not perform competitive learning.

Synapse transistors [15]–[18] address the problems raised in the previous paragraph. These devices use a floating-gate technology similar to that found in digital EEPROMs to provide nonvolatile analog storage and local adaptation in silicon. The adaptation mechanisms do not significantly perturb the operation of the device, thus enabling simultaneous adaptation and computation. Despite these advantages, the adaptation mechanisms provide dynamics that are difficult to translate into existing neural-network learning rules. Thus, so far, this technology has not been used to build competitive learning networks in silicon.

To avoid the difficult task of mimicking existing competitive-learning rules in silicon, we instead constructed a circuit that naturally exhibited competitive learning dynamics and then derived a learning rule directly from the physics of the component synapse transistors. Our 11-transistor silicon circuit, termed an automaximizing bump circuit, computes a similarity measure, provides nonvolatile storage and local adaptation and performs simultaneous adaptation and computation. As we show in this paper, our circuit provides the functionality we desire in a competitive-learning primitive.

By employing different feedback error signals to our bump circuit, we can develop a large class of competitive-learning networks in silicon. Consequently, we envision this circuit as a fundamental building block for many large-scale clustering and classification networks. As a first example, we have fabricated a circuit that clusters one-dimensional (1-D) data.

We begin this paper by reviewing synapse transistors. In Section III, we describe the automaximizing bump circuit. In Section IV, we show data from a 1-D competitive learning network, fabricated in a 0.35-μm CMOS process, that learns to cluster data drawn from a mixture of Gaussians. The network architecture is readily scalable to n-dimensional inputs. The later sections discuss issues related to this architecture and demonstrate, via software simulation, that the competitive learning rule derived from the bump synapses can perform effective clustering. Finally we provide some discussion and conclusions.

II. Synapse Transistors

Because the properties of the automaximizing bump circuit depend on the storage and adaptation mechanisms of synapse transistors, we begin by reviewing these mechanisms. Fig. 2 illustrates the four-terminal *p*FET synapse transistor that we use in the bump circuit. The synapse comprises a single MOSFET, (with a poly2 control gate capacitively coupled to a poly1 floating gate) and an associated n-well tunneling implant. It uses floating-gate charge to represent a nonvolatile analog memory and outputs a source current that varies with both the stored charge and the control-gate input voltage. The synapse transistor uses two mechanisms for adaptation: Fowler-Nordheim (FN) tunneling [19] increases the stored charge; impact-ionized hot-electron injection (IHEI) [20] decreases the charge. Because tunneling and IHEI can both be active during normal transistor operation, the synapse enables simultaneous adaptation and computation.

A voltage difference between the floating gate and the tunneling implant causes electrons to tunnel from the floating gate,

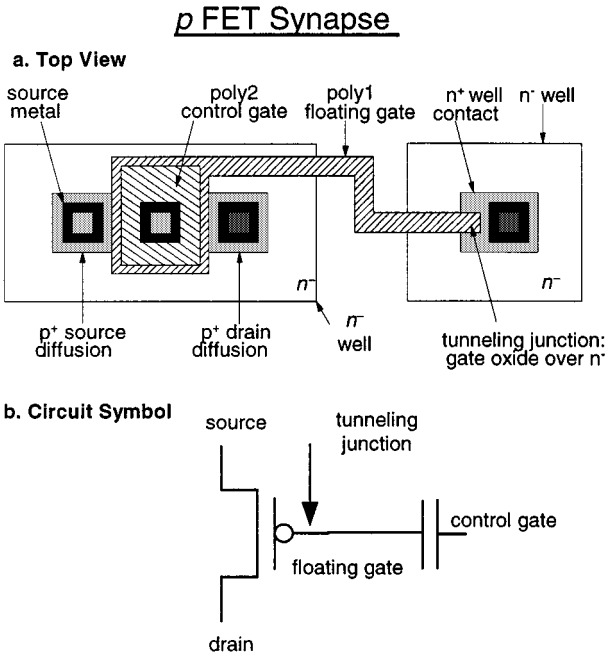

Fig. 2. (a) Layout view of a *p*FET synapse. The synapse comprises a single MOSFET, with poly1 floating gate and poly2 control gate and an associated n-well tunneling implant. (b) Circuit symbol for a *p*FET synapse.

through gate oxide, to the tunneling implant. The magnitude of the tunneling current depends on the voltage across the gate oxide, defined as the difference between the floating-gate and tunneling-implant voltages. We approximate this gate current by

$$I_{\rm tun} = I_t e^{-V_f/(V_{\rm tun}-V_{\rm fg})} \tag{3}$$

where I_t is a pre-exponential constant and V_f depends on oxide thickness [17]. We substitute $V_{\rm tun} = V_{\rm tun0} + \delta V_{\rm tun}$ and $V_{\rm fg} = V_{\rm fg0} + \delta V_{\rm fg}$ into (3), where $V_{\rm tun0}$ and $V_{\rm fg0}$ are quiescent voltage levels around which the (small-signal) tunneling and floating-gate voltages vary. We represent these variations as $\delta V_{\rm tun}$ and $\delta V_{\rm fg}$. We then approximate $1/(1+x) \approx 1-x$ for small x and solve [18]

$$I_{\rm tun} = I_{\rm tun0} e^{(\delta V_{\rm tun} - \delta V_{\rm fg})/V_\chi} \tag{4}$$

where $I_{\rm tun0}$ is the tunneling current when $\delta V_{\rm tun} = 0$ and $\delta V_{\rm fg} = 0$ and V_χ is a constant defined by

$$V_\chi = \frac{(V_{\rm tun0} - V_{\rm fg0})^2}{V_f}. \tag{5}$$

Fig. 3 shows tunneling data for the 0.35 μm process, including a fit according to (4).

IHEI (impact-ionized hot-electron injection) adds electrons to the floating gate, decreasing its stored charge. The magnitude of the IHEI current varies with the transistor's source current and channel-to-drain voltage ($V_{\rm cd}$) [18]; over a small drain-voltage range, we can model this dependence as [15]

$$I_{\rm inj} = \alpha I_s e^{V_{\rm cd} V_\gamma} \tag{6}$$

where α and V_γ are constants. In Fig. 4, we illustrate the injection efficiency, defined as the injection current divided by the

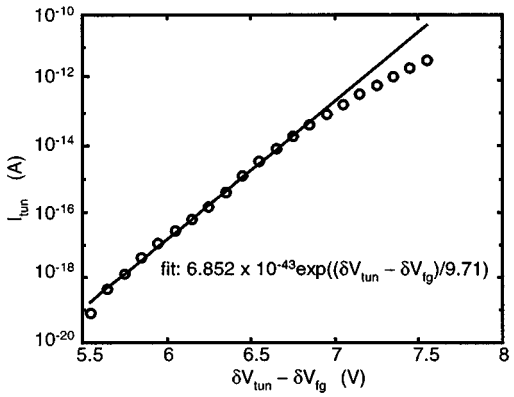

Fig. 3. Tunneling current $I_{\rm tun}$ versus $(V_{\rm tun} - V_{\rm fg})$. Equation (3) fits the experimental data over the range of oxide-current values that we use in the automaximizing bump circuit.

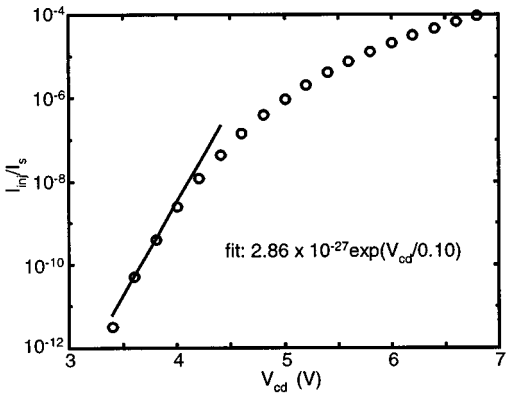

Fig. 4. Injection efficiency, defined as $I_{\rm inj}$ divided by I_s, versus channel-to-drain voltage, with a fit from (5). Our bump-circuit experiments (e.g., Figs. 7–9) use channel-to-drain voltages ranging from 3.25 V to 3.75 V.

source current, along with a fit according to (5). If we operate the MOSFET in its subthreshold regime, $V_{\rm cd}$ is essentially constant across the channel and we can rewrite (6) in terms of the transistor's source-to-drain voltage, $V_{\rm sd}$ and its source current, I_s [15]

$$I_{\rm inj} = \alpha I_s^{1-U_t/V_\gamma} e^{V_{\rm sd}/V_\gamma} \tag{7}$$

where U_t is the thermal voltage kT/q (~ 25.7 mV at room temperature). Equation (7) illustrates the dependence of the IHEI current on the transistor's source-to-drain voltage and source current.

III. The Automaximizing Bump Circuit

The automaximizing bump circuit is an adaptive version of the classic bump-antibump circuit [21]. The circuit uses synapse transistors to implement the three necessary functions of a competitive-learning synapse: 1) storage of a weight value μ; 2) computation of a similarity measure between the input and μ; and 3) adaptation that decreases the difference between μ and the present input.

The bump-antibump circuit of Fig. 5(a) provides an ideal starting point for our discussion. It comprises only five transistors and computes a similarity measure between two input voltages, V_1 and V_2. The bump circuit generates three output

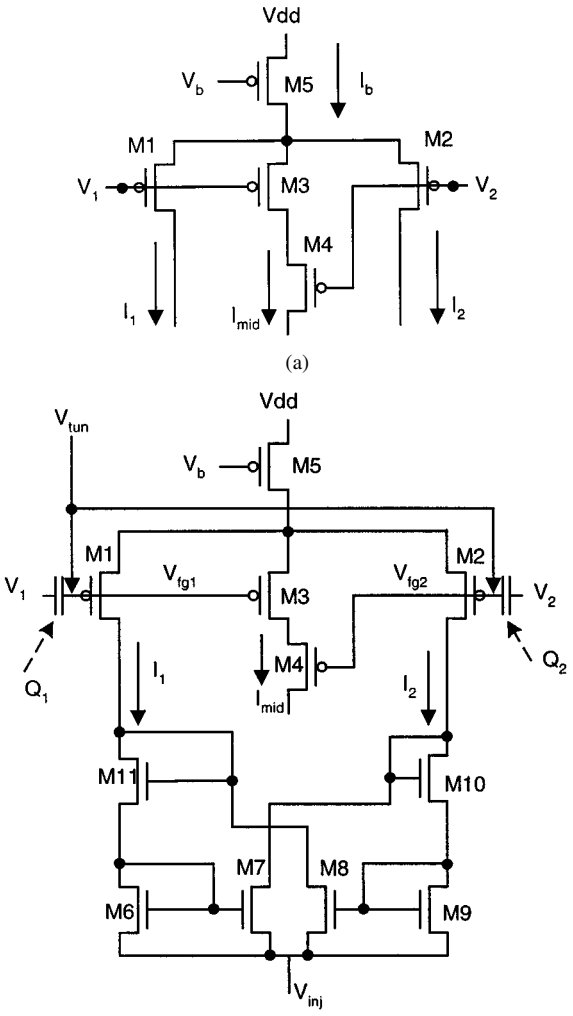

Fig. 5. (a) Bump–antibump circuit. $I_{\rm mid}$ computes a similarity measure between V_1 and V_2. I_1 or I_2 increase when $V_1 \gg V_2$ or $V_2 \gg V_1$, respectively. (b) Automaximizing bump circuit. We replace M_1, M_2, M_3 and M_4 with synapse transistors. M_1 and M_3 share a common floating gate, as do M_2 and M_4. $I_{\rm mid}$ computes a similarity measure between $V_{\rm fg1}$ and $V_{\rm fg2}$ where $V_{\rm fg1}$ depends on V_1 and Q_1 and $V_{\rm fg2}$ depends on V_2 and Q_2, respectively. A high voltage on $V_{\rm tun}$ causes electron tunneling from both floating gates. A low voltage on $V_{\rm inj}$ enables injection through the *n*FET current mirrors. The circuit adapts Q_1 and Q_2 to maximize $I_{\rm mid}$.

currents. The two outside currents, I_1 and I_2, are a measure of the dissimilarity between the two inputs; the center current is a measure of the similarity. The center current $I_{\rm mid}$ follows the equation

$$I_{\rm mid} = \frac{I_b}{1 + \left(\frac{4}{S}\right)\cosh^2\left(\left(\frac{\kappa}{2U_t}\right)(V_1 - V_2)\right)} \tag{8}$$

where S is the ratio of the strength of the middle transistors (M_3 and M_4) to the outer transistors (M_1 and M_2), κ is the back-gate coefficient (approx $0.6 \sim 0.8$) and I_b is the bias current sourced by M_5. $I_{\rm mid}$ is symmetric with respect to the voltage difference between V_1 and V_2 and approximates a Gaussian centered at $V_1 - V_2 = 0$.

Part (b) of Fig. 5 shows the automaximizing bump circuit that, for convenience, we refer to as a *bump circuit*. We replace M_1, M_2, M_3 and M_4 with synapse transistors. M_1 and M_3 share

a common floating gate and tunneling junction, as do M_2 and M_4. The charge stored on the bump circuit's floating gates shift I_{mid}'s peak away from $V_1 = V_2$. We interpret this shift as the weight, μ, stored by the circuit and interpret I_{mid} as a similarity measure between the differential input, $V_1 - V_2$ and the weight μ.

Alternately, we could use the antibump outputs as a distance measure. I_1 and I_2 are large when $V_1 \gg V_2$ or $V_2 \gg V_1$, respectively. In addition to providing a (saturating) distance measure, the antibump outputs also provide the direction of the inequality. Unfortunately, the antibump outputs saturate and therefore only provide distance information for inputs close to the stored weight. Although I_{mid} also saturates, $\log(I_{\text{mid}})$ does not. Therefore, I_{mid} is a more informative similarity measure (for more details see Section V). Furthermore, when computing distances between two objects, we are typically unconcerned with direction, only with magnitude. Consequently, even though I_{mid} does not provide direction information, this is not a concern. Direction is important for computing weight updates and, as we show in Section III-B2, we use the antibump outputs to perform adaptation.

The circuit is *automaximizing* because tunneling and injection naturally tune I_{mid}'s peak to coincide with the present input $(V_1 - V_2)$, decreasing the difference between the stored weight and the input. We enable adaptation by activating both tunneling and injection and disable adaptation by shutting off both mechanisms. A high $V_{\text{tun}}(\sim 10\text{ V})$ causes tunneling and a low $V_{\text{inj}}(\sim 0\text{ V})$ causes injection. The *n*FET current mirrors $(M_6 - 9)$ and diodes $(M_{10} - M_{11})$ control the amount of injection at each synapse transistor (we defer details until Section III-B2). A low $V_{\text{tun}}(< 8\text{ V})$ and high $V_{\text{inj}}(> 2\text{ V})$, deactivate adaptation. We can achieve a wide range of adaptation rates by choosing appropriate values for V_{tun} and V_{inj}.

By itself, the bump circuit does not implement competitive learning. The circuits we construct around it, that select bumps for adaptation, enable competitive behavior. Different selection mechanisms can implement a wide variety of competitive learning rules; we show one such rule in Section IV. We conclude this section by describing the bump circuit in more detail.

A. Stored Weight and Input Representation

We now express the bump circuit's weight as a function of its floating-gate charge. This charge has the same effect as a voltage in series with the control gate, of value Q/C_{in}, where Q is the floating-gate charge and C_{in} is the control-gate to floating-gate coupling capacitance. We define the input V_{in} to the bump circuit to be a differential signal $V_{\text{in}} = V_1 - V_2$. To ensure symmetric adaptation, we constrain the common mode of V_1 and V_2 to be a fixed voltage V_0 (how this ensures symmetric adaptation will become clear shortly) and express $V_1 = V_0 + V_{\text{in}}/2$ and $V_2 = V_0 - V_{\text{in}}/2$. I_{mid} computes the similarity between the floating-gate voltages $V_{\text{fg1}} = Q_1/C_{\text{in}} + V_{\text{in}}/2 + V_0$ and $V_{\text{fg2}} = Q_2/C_{\text{in}} - V_{\text{in}}/2 + V_0$, where Q_1 and Q_2 are M_1's and M_2's floating-gate charge. We define the circuit's weight μ as

$$\mu = \frac{Q_2 - Q_1}{C_{\text{in}}}. \tag{9}$$

Because of the differential input encoding and weight definition, $|V_{\text{in}} - \mu| = |V_{\text{fg1}} - V_{\text{fg2}}|$. Therefore, I_{mid} computes the similarity between V_{in} and μ. Fig. 6 shows the bump circuit's I_{mid} output for three weight values, as a function of the differential input V_{in}. We see that different stored values change the location of the peak, but do not change the bump shape.

The differential encoding of V_{in} in terms of V_1 and V_2 also leads to symmetric adaptation dynamics, because the values of the two floating-gate voltages only depend on the magnitude of $V_{\text{in}} - \mu$ and not on the sign. Other encodings (e.g., setting $V_1 = V_{\text{in}} + V_0$ and $V_2 = V_0$) do not have this property.

B. Adaptation

We now examine the bump circuit's adaptation. We start by defining $\Delta V_{\text{fg}} = |V_{\text{fg1}} - V_{\text{fg2}}|$. Because $\Delta V_{\text{fg}} = |V_{\text{in}} - \mu|$, the magnitude of $d\mu/dt$ is equivalent to $d\Delta V_{\text{fg}}/dt$. We begin by considering the effect of tunneling on ΔV_{fg}.

1) The Tunneling Learning Rule: Tunneling decreases the difference between V_{fg1} and V_{fg2}. In practice, tunneling increases the voltage of both floating gates, but, because tunneling increases exponentially the lower the floating-gate voltage [see (4)], tunneling decreases the difference.

Assuming that M_1's floating-gate voltage is lower than M_2's, the change in ΔV_{fg} due to electron tunneling is

$$\frac{d\Delta V_{\text{fg}}}{dt} = -\frac{I_{\text{tun1}} - I_{\text{tun2}}}{C_{\text{in}}} \tag{10}$$

where I_{tun1} and I_{tun2} are the tunneling currents at M_1 and M_2, respectively. Equation (4) describes the tunneling current as a function of the deviation of the floating gate voltage from a fixed voltage level. Consequently, we express variations in the bump circuit's two floating-gate voltages as $\delta V_{\text{fg1}} = \delta V_0 - \Delta V_{\text{fg}}/2$ and $\delta V_{\text{fg2}} = \delta V_0 + \Delta V_{\text{fg}}/2$, where $\delta V_0 = (\delta V_{\text{fg1}} + \delta V_{\text{fg2}})/2$ is the small-signal variation in the common-mode voltage. We substitute (4) into (10) and solve

$$I_{\text{tun1}} = I_{\text{tun0}} e^{(\delta V_{\text{tun}} - \delta V_0 + \Delta V_{\text{fg}}/2)/V_\chi} \tag{11}$$

$$I_{\text{tun2}} = I_{\text{tun0}} e^{(\delta V_{\text{tun}} - \delta V_0 - \Delta V_{\text{fg}}/2)/V_\chi}. \tag{12}$$

We substitute (11) and (12) into (10) and solve

$$\frac{d\Delta V_{\text{fg}}}{dt} = -I_{t0} e^{(\delta V_{\text{tun}} - \delta V_0)/V_\chi} \sinh\left(\frac{\Delta V_{\text{fg}}}{2V_\chi}\right) \tag{13}$$

where $I_{t0} = 2I_{\text{tun0}}/C_{\text{fg}}$. $d\Delta V_{\text{fg}}/dt$ depends on three factors: a controllable learning rate, δV_{tun}; the difference between the input and the stored weight, ΔV_{fg}; and the average floating-gate voltage, δV_0.

Unfortunately, tunneling currents are not well matched, even for two synapse transistors on the same chip. Consequently, the two tunneling currents equalize at a slight offset from $\Delta V_{\text{fg}} = 0$. We model the mismatch by adding an offset term to the sinh from (13)

$$\frac{d\Delta V_{\text{fg}}}{dt} = -I_{t0} e^{(\delta V_{\text{tun}} - \delta V_0)/V_\chi} \sinh\left(\frac{\Delta V_{\text{fg}} - \phi}{2V_\chi}\right). \tag{14}$$

Fig. 7 shows measured $d\Delta V_{\text{fg}}/dt$ versus ΔV_{fg} due to tunneling, including a fit from (14). In our experiments, the measured tunneling offset is about 18 mV.

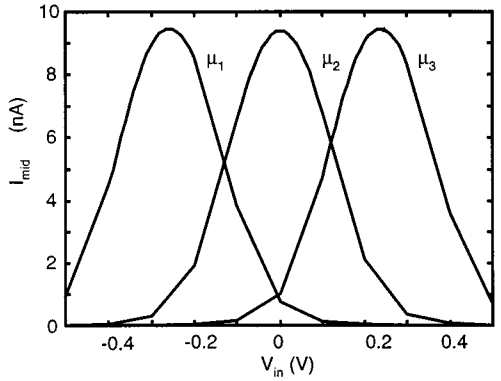

Fig. 6. Experimental measurements of I_{mid} versus V_{in} for a single automaximizing bump circuit, for three stored values labeled μ_1, μ_2 and μ_3. The stored weight changes the location of the bump peak, but not the bump shape.

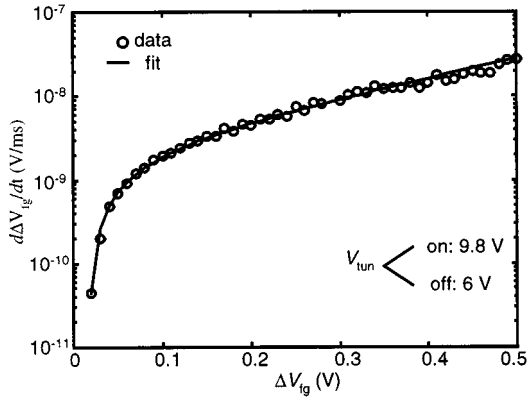

Fig. 7. Derivative of ΔV_{fg} plotted versus ΔV_{fg} due to electron tunneling. We fit these data using (14). We measured the change in the location of I_{mid}'s peak due to a short tunneling pulse when the floating gates were ΔV_{fg} apart. Different values for δV_0 merely change the magnitude of adaptation, not the general shape. We followed the same measurement procedure for the experiments of Figs. 8 and 9.

2) The Injection Learning Rule: IHEI decreases the difference between V_{fg1} and V_{fg2}. We bias the circuit so that only M_1 and M_2 experience IHEI. From (7), we see that IHEI varies sublinearly with the transistor's source current, but exponentially with its source-to-drain voltage. Consequently, we decrease ΔV_{fg} by controlling the drain voltages of M_1 and M_2.

We use cross-coupled current mirrors ($M_6 - M_7$ and $M_8 - M_9$) at the drains of M_1 and M_2, to raise the drain voltage of the floating-gate transistor that is sourcing a larger current and to lower the drain voltage of the floating-gate transistor that is sourcing a smaller current. The transistor with the smaller source current experiences a larger V_{sd} and thus exponentially more IHEI, causing its source current to rapidly increase. Diodes ($M6, M_9 - M_{11}$) raise the drain voltage of the transistor with the larger current yet further. The net effect is that IHEI equalizes the source currents of the floating-gate transistors and, likewise, their floating-gate voltages. Negative feedback from the cross-coupled current mirrors is necessary for the automaximizing behavior; circuits that do not incorporate such feedback exhibit runaway IHEI [18]. Appendix A provides a small-signal stability analysis of this feedback.

To see this behavior more clearly, let us first examine the case where the two floating-gate voltages are equal. If I_1 and I_2 are equal, then the drain voltages of M_1 and M_2 will likewise be equal. If we set V_{inj} low enough to cause injection, both transistors will inject at the same rate, because their source currents and drain voltages will be the same. Therefore, injection does not change ΔV_{fg}. In addition, because transistors M_6 and M_8 have the same gate voltage, I_1 will split evenly at the drain of M_{11}. The same is true for I_2 at the drain of M_{10}.

Now consider the case where V_{fg1} is lower than $V_{\mathrm{fg2}} (I_1 > I_2)$. M_7 now tries to sink a current that is larger than $I_2/2$. Consequently, $M_9 - M_{10}$ sink less than $I_2/2$, causing M_2's drain voltage to fall and, in turn, decreasing the current that M_8 sinks. This further increases the amount of current that flows through M_7. The positive feedback process causes M_7 to sink all of I_2 and M_{11} and M_6 to sink all of I_1. As a result, M_2's drain voltage drops to V_{inj} and M_1's drain voltage rises to $2U_t/\kappa \log(I_1/I_0) + V_{\mathrm{inj}}$. This process will exhibit hysteresis if the gain of either current mirror exceeds unity (see Appendix A).

The weight change for injection follows a similar form to that of tunneling

$$\frac{d\Delta V_{\mathrm{fg}}}{dt} = -\frac{I_{\mathrm{inj2}} - I_{\mathrm{inj1}}}{C_{\mathrm{in}}}. \tag{15}$$

The final IHEI weight update rule follows by substituting the expressions for I_{inj1} and I_{inj2} from (7) into (15). To determine values for the drain voltages of M_1 and M_2, we assume that all of I_1 flows through M_{11} and all of I_2 flows through M_7. We derive the update rule from a large-signal analysis in Appendix B. The final form is

$$\frac{d\Delta V_{\mathrm{fg}}}{dt} = -I_{j0} e^{\varsigma \delta V_0} \left(e^{\tau V_{\mathrm{inj}}} \Phi_1 (\Delta V_{\mathrm{fg}}) - e^{\eta V_{\mathrm{inj}}} \Phi_2 (\Delta V_{\mathrm{fg}}) \right) \tag{16}$$

where $I_{j0} = \left(\alpha I_0^{1-U_t/V_\gamma} \right) / C_{\mathrm{in}}$, $\zeta = \kappa/V_\gamma$, $\tau = -2/\kappa V_\gamma$ and $\eta = -1/V_\gamma$. I_0 is the transistor's pre-exponential current. The two functions, Φ_1 and Φ_2, are

$$\Phi_1 (\Delta V_{\mathrm{fg}}) = \Phi (\Delta V_{\mathrm{fg}})^{1-2U_t/\kappa V_\gamma} e^{-\omega \Delta V_{\mathrm{fg}}} \tag{17}$$

$$\Phi_2 (\Delta V_{\mathrm{fg}}) = \Phi (\Delta V_{\mathrm{fg}}) e^{-\sigma \Delta V_{\mathrm{fg}}} \cdot \left(1 - e^{-\kappa \Delta V_{\mathrm{fg}}/U_t} \right)^{-U_t/V_\gamma} \tag{18}$$

where $\sigma = (1 - U_t/V_\gamma)^{\kappa/2U_t}$, $\omega = \kappa/2U_t - \kappa/2V_\gamma - 1/V_\gamma$ and Φ is

$$\Phi (\Delta V_{\mathrm{fg}}) = \frac{I_b - I_{\mathrm{mid}}}{2 \cosh \left(\frac{\kappa \Delta V_{\mathrm{fg}}}{2U_t} \right)}. \tag{19}$$

Like tunneling, $d\Delta V_{\mathrm{fg}}/dt$ due to IHEI depends on three factors: A controllable learning rate, V_{inj}; the difference between the stored weight and the input, ΔV_{fg}; and δV_0.

Fig. 8 shows experimental measurements of $d\Delta V_{\mathrm{fg}}/dt$ versus ΔV_{fg} due to IHEI, along with a fit from (16). As ΔV_{fg} increases, the weight-update magnitude reaches its peak between 0.1 V and 0.2 V and then decreases afterwards. The increase in magnitude between 0 V and $\sim$ 0.14 V occurs because the difference between the drain voltages of M_1 and M_2 increases as ΔV_{fg} increases. At $\Delta V_{\mathrm{fg}} > 0.14$, the difference between M_1's and M_2's drain voltages has reached its maximum value; therefore, further increasing ΔV_{fg} only causes the source current for the

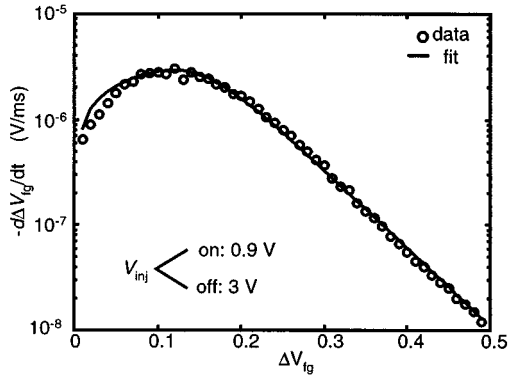

Fig. 8. Derivative of $\Delta V_{\rm fg}$ plotted versus $\Delta V_{\rm fg}$ due to electron injection. We fit these data using (16). Different values for δV_0 merely change the magnitude of adaptation, not the shape.

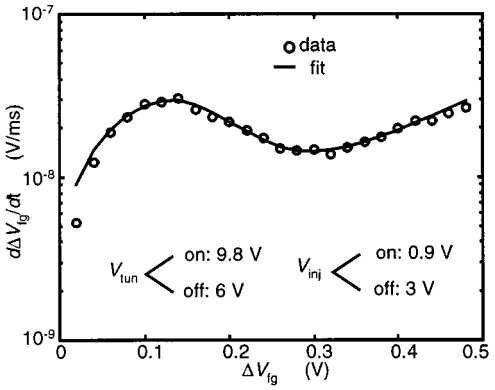

Fig. 9. Derivative of $\Delta V_{\rm fg}$ versus $\Delta V_{\rm fg}$, due to electron injection and tunneling. We simultaneously pulsed $V_{\rm tun}$ and $V_{\rm inj}$ on for short periods of time and measured the resulting change in $\Delta V_{\rm fg}$. We fit these data to the learning rule given in (20).

transistor with the lower floating gate voltage to increase and the source current of the other transistor to decrease, decreasing the magnitude of the IHEI weight update. Finally, we note that because the bump circuit's output is symmetric with respect to I_1 and I_2, the IHEI weight-update rule takes the same form when $I_2 > I_1$.

3) Composite Learning Rule: Injection and tunneling cause adaptation by adding and removing charge from the bump circuit's floating gates, respectively. In isolation, either mechanism will eventually drive the bump circuit out of its useful operating range. For effective adaptation, we need to activate both. Interestingly, Figs. 7 and 8 show that tunneling acts more strongly for larger values of $\Delta V_{\rm fg}$, whereas injection shows the opposite behavior. The two mechanisms complement each other to provide effective adaptation over a large range of $\Delta V_{\rm fg}$.

Combining (14) and (16), we obtain the general bump circuit weight-update rule

$$\frac{d\Delta V_{\rm fg}}{dt} = -I_{t0} e^{-\delta V_{\rm tun}/V_\chi} e^{-\delta V_0/V_\chi} \sinh\left(\frac{\Delta V_{\rm fg} - \phi}{2V_\chi}\right) - I_{j0} e^{\varsigma \delta V_0} \left(e^{\tau V_{\rm inj}} \Phi_1\left(\Delta V_{\rm fg}\right) - e^{\eta V_{\rm inj}} \Phi_2\left(\Delta V_{\rm fg}\right)\right). \quad (20)$$

The data and fit in Fig. 9 illustrates this learning rule. When $\Delta V_{\rm fg}$ is small, adaptation is primarily driven by injection, whereas tunneling dominates for larger values of $\Delta V_{\rm fg}$. The dip in adaptation rate around $\Delta V_{\rm fg} = 0.2$ V occurs because, as injection decreases, the contribution of tunneling has not yet increased enough to compensate. We can change the learning dynamics by modifying the sizes of transistor sizes of $M_1 - M_4$, the topology of the *n*FET current mirrors and the magnitudes of $V_{\rm inj}$ and $V_{\rm tun}$.

The weight-update rule in (20) is unlike any learning rule that we have found in the literature. Nevertheless, it exhibits several desirable properties. First, it increases the similarity between the bump circuit's weight and the present input. Second, the weight update is symmetric with respect to the difference between the weight and the present input. Third, the update rule decomposes into a product of 1) a weight-independent update rate; 2) update rates set by controllable terminal voltages ($V_{\rm inj}$ and $V_{\rm tun}$); 3) an autozeroing common-mode voltage (δV_0); and 4) rates dependent on the difference between the weight and the input ($\Delta V_{\rm fg}$). Finally, the learning rule derives from the physics of the silicon. This allows for a compact and efficient VLSI implementation.

C. Layout Area and Power Consumption

Recently optimized bump-circuit layouts occupy 20 μm $\times$ 20 μm in a 0.35 μm process. We share the n-well that the tunneling junctions occupy among multiple bump circuits, saving layout area (especially for large arrays).

We operate our bump circuit with subthreshold currents in the range of 10 nA to 100 nA. Consequently, the power consumption is typically less than a microwatt.

D. Common-Mode Rejection

$I_{\rm mid}$ varies with the common mode voltage, δV_0 (recall $\delta V_0 = (\delta V_{\rm fg1} + \delta V_{\rm fg2})/2$). Parasitic capacitive coupling from $V_{\rm tun}$ and $V_{\rm inj}$ to the floating gates alters the common-mode voltage, thereby altering $I_{\rm mid}$. A few simple circuit techniques can reduce this problem. Increasing the length of the bias transistor, cascoding the bias transistor, or increasing the sizes of the floating-gate capacitors increases the common-mode rejection. The first two solutions do not appreciably increase the layout area. Finally, Harrison *et al.* have described compensatory circuits for canceling parasitic coupling to the floating gates [22]. We can adapt these techniques to the bump circuit.

Variations in δV_0 during adaptation also affect the circuit's learning dynamics, because altering δV_0 alters the relative contribution of injection and tunneling in the bump circuit's learning rule (20). Our circuit compensates by *autozeroing* δV_0 over time [23]. Indeed, notice that a high δV_0 strengthens injection and weakens tunneling, lowering δV_0. Conversely, a low δV_0 strengthens tunneling and weakens injection, raising δV_0.

Theoretically, δV_0 can reach a state in which both the injection and tunneling currents are small, leading to slow adaptation. This state can occur when the common-mode voltage is so high that there is little tunneling and the difference between $V_{\rm in}$ and μ is so large that there is little injection. However, this only occurs if the bump circuit only adapts to inputs that are very distant from it. The likelihood of this occurrence is very small in classification and clustering problems because the bump circuit typically adapts to a distribution of inputs.

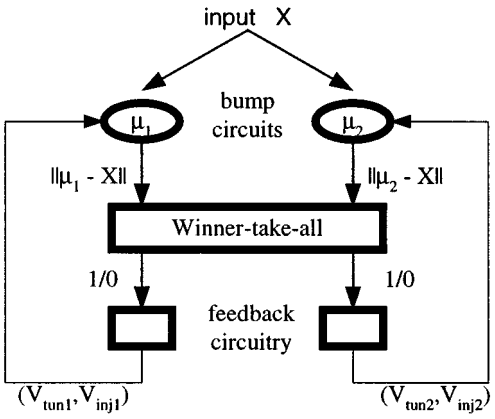

Fig. 10. A CLC, comprising two bump circuits, a WTA and feedback circuitry that derive local $V_{\rm inj}$ and $V_{\rm tun}$ signals from the WTA output.

IV. Competitive Learning Circuit

Fig. 10 shows a 1-D competitive learning circuit (CLC) that clusters 1-D data. The CLC comprises multiple bump circuits with a common input. Each bump circuit computes the similarity between the input and its weight; a WTA [13] selects the bump circuit that produces the largest output. On-chip feedback circuitry transforms the WTA's outputs into $V_{\rm tun}$ and $V_{\rm inj}$ signals for each bump circuit, causing the closest bump circuit to adapt to the input and inhibiting other bump circuits from adapting.

The feedback circuitry that converts the WTA output into $V_{\rm tun}$ and $V_{\rm inj}$ signals is quite simple—it comprises three inverters and a modified transconductance amplifier. The inverters shown in Fig. 11(a) take the WTA output ($V_{\rm wta}$) and generate $V_{\rm inj} = V_{\rm low}$ when $V_{\rm wta}$ is high, or $V_{\rm inj} = V_{\rm hi}$ when $V_{\rm wta}$ is low. These voltages activate or deactivate injection, respectively. We used three inverters to increase the overall gain in the circuit. V_b sets the first inverter's threshold. In our experiments, we used $V_{\rm low} = 0$ V and $V_{\rm hi} = 2$ V.

The modified transconductance amplifier in Fig. 11(b) takes as input $V_{\rm tunin}$, the voltage generated by the second inverter in the $V_{\rm inj}$ generator and sets $V_{\rm tun}$ equal to $V_{\rm tunh}$ when $V_{\rm tunin}$ is higher than $V_{\rm comp}$ and four diode drops below $V_{\rm tunh}$ otherwise. The voltage swing on $V_{\rm tun}$ activates or deactivates tunneling. Because the drain voltages of M_1 and M_2 are the tunneling voltage (~ 10 V), we make these devices high-voltage nFETS by using a n-well for the drain. For a layout, see [22]. Together, the two circuits in Fig. 11 simultaneously activate or deactivate injection and tunneling.

We fabricated 2-bump and 8-bump versions of the CLC. Although both circuits are functional, for ease of testing we have performed most of our experiments on the 2-bump version. In Fig. 12, we show competitive learning in the 2-bump CLC. We applied a single input drawn from a mixture of 2 Gaussians. The CLC clearly adapts to the Gaussians—each bump selects and adapts to the mean of one of the two Gaussians.

For comparison, we also show in Fig. 12 the results of a software neural network in which the most similar neuron updates its value using the standard competitive learning rule from (2) where we set the learning rate ρ to 0.01. We apply the same data to the simulation that we send to the chip. The data show that the

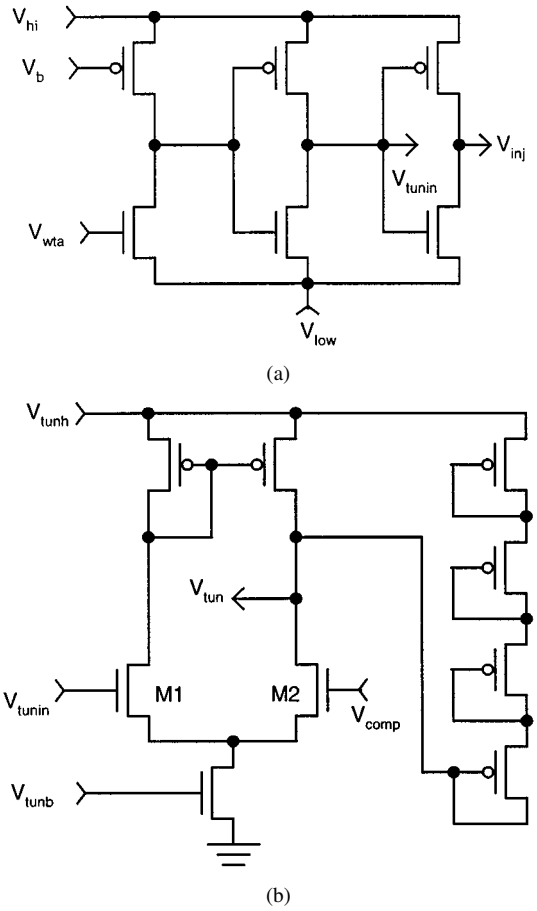

Fig. 11. (a) Circuit that generates $V_{\rm inj}$ from the WTA output. (b) Circuit that generates $V_{\rm tun}$ from the circuit in part (a).

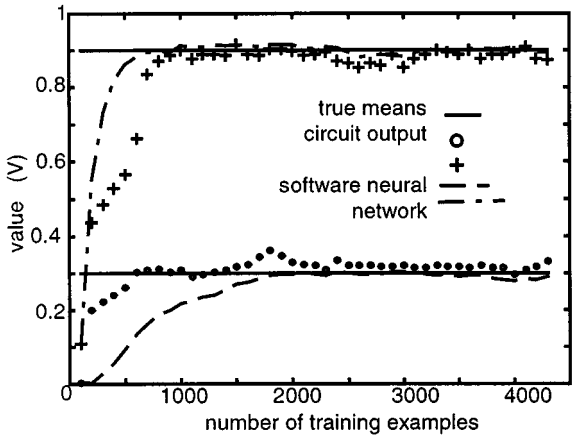

Fig. 12. Comparison between the competitive learning circuit and an equivalent software neural network that used a standard competitive learning rule. The input data were drawn from a mixture of two Gaussians, with 80% of the data drawn from the higher Gaussian. We plot the means learned by the circuit and the software simulation, compared to the true means of the two Gaussians.

bump circuit's weight-update rule exhibits the same functionality as the basic competitive-learning rule, but with different adaptation dynamics.

The bump circuits' weights exhibit a consistent offset from the values learned by the neural network. We believe the cause is a turn-on delay in the well-tunneling junctions—electron tun-

neling does not actually occur until several seconds *after* we raise the tunneling voltage. We have observed a similar effect previously (see [17]: the section on bowl-shaped tunneling junctions), although not in the tunneling junction that we show in Fig. 2 (n^+ in n^- well). If the latency between the two synapses in a bump circuit differs, the weight may drift away from the input; only when both tunneling junctions finally turn on does the weight adapt toward the input. The tunneling latency may also explain why one bump circuit adapts so quickly to one of the true means, while the other bump circuit exhibits slower adaptation. Each bump circuit, because of the latency, may be strongly biased toward adaptation in one direction. We have developed an alternative tunneling junction that tunnels over p^+ rather than over n^+; in experiments on isolated tunneling junctions, the revised design does not exhibit a turn-on delay. We are currently incorporating the revised design in future versions of the automaximizing bump circuit.

Although both the software neural network and the CLC drift about the true Gaussian centers, the CLC shows fluctuations of greater magnitude. We believe the reason is that the CLCs learning rate is greater than the software neural network's learning rate Unfortunately, because the CLC operates completely asynchronously, it is hard to quantify the learning rate that the circuit uses.

Our competitive learning architecture can be scaled to n-dimensional inputs and any number of neurons. Each neuron requires one bump circuit per synapse. Most important, each neuron requires only one feedback block, because all the synapses receive the same V_{tun} and V_{inj}. We illustrate the approach in Fig. 13. Each bump circuit corresponds to a synapse of a neuron; the WTA from Fig. 10 is an example of an inhibitory circuit; and the circuits of Fig. 11 are examples of feedback circuits.

V. Neuron Design

Because a bump circuit computes a similarity measure, the method we should use to combine bump outputs is not obvious. The bump circuit output is a current; since addition of currents is particularly easy to implement in analog VLSI, we might surmise that addition is the correct way to combine bump outputs. In fact, in at least one previous hardware neural network, with synapses that compute a similarity measure, the neurons add these similarity measures [24]. However, the way we combine bump similarities implicitly defines a distance measure. If we want to approximate some natural distance measures like Manhattan distance or squared distance[1] in our network, then we will show that multiplication provides a more sensible way to combine bump similarities.

Intuitively, the bump circuit's similarity measure approximates the probability that an input x was generated by a 1-D Gaussian $N(\mu, \sigma^2)$ (with mean μ and variance σ^2), where μ corresponds to the bump circuit's stored weight

$$prob(x) = \left(\frac{1}{2\pi\sigma^2}\right)^{1/2} e^{-(x-\mu)^2/2\sigma^2}. \tag{21}$$

[1]For two vectors $\mathbf{x}$ and $\mathbf{y}$, the Manhattan distance is $\Sigma_i |x_I - y_i|$ and the squared distance is $\Sigma_i (x_I - y_i)^2$.

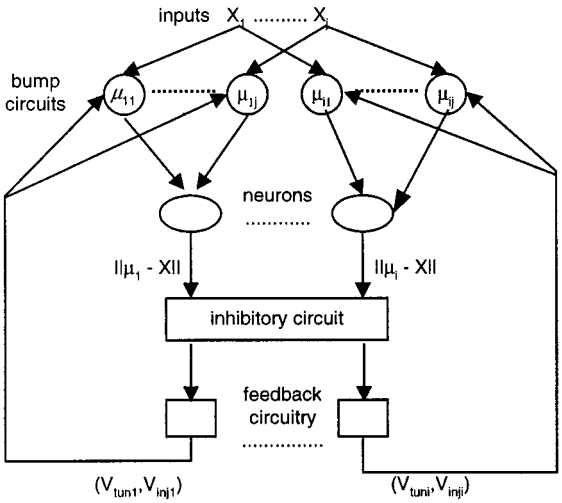

Fig. 13. Generalized competitive learning architecture.

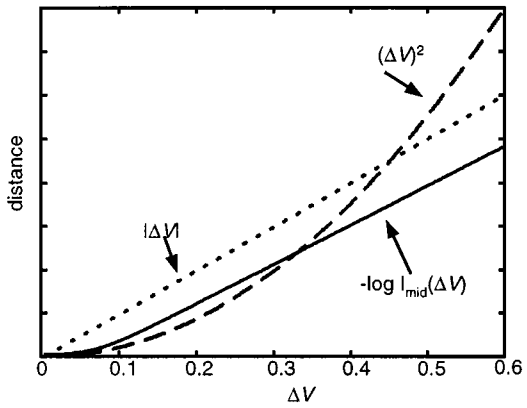

Fig. 14. Comparison of the bump distance measure to Manhattan and squared distance. Distances are scaled to facilitate a comparison.

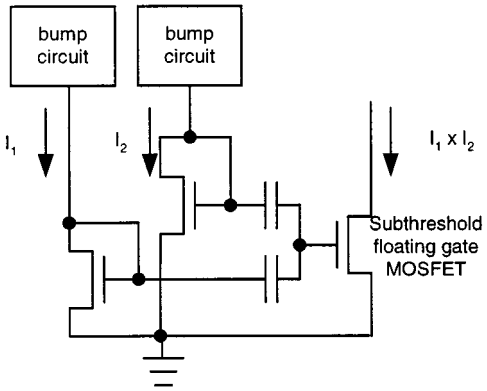

Fig. 15. Circuit that multiplies the output of two bump circuits. Each diode transforms its input current (I_1 or I_2) into a voltage that is logarithmic with the current. The *n*FET on the right is a floating-gate transistor with two control gates; one for each diode voltage. If the floating gate *n*FET is biased in its subthreshold regime, its output current will be proportional to the product of the two bump-circuit currents.

A Gaussian is an exponential function of the squared Euclidean distance between input x and mean μ. Consequently, multiplying a set of Gaussian similarities is equivalent to adding the corresponding Euclidean distances. Conversely, adding Gaussian similarities does not correspond to any sensible

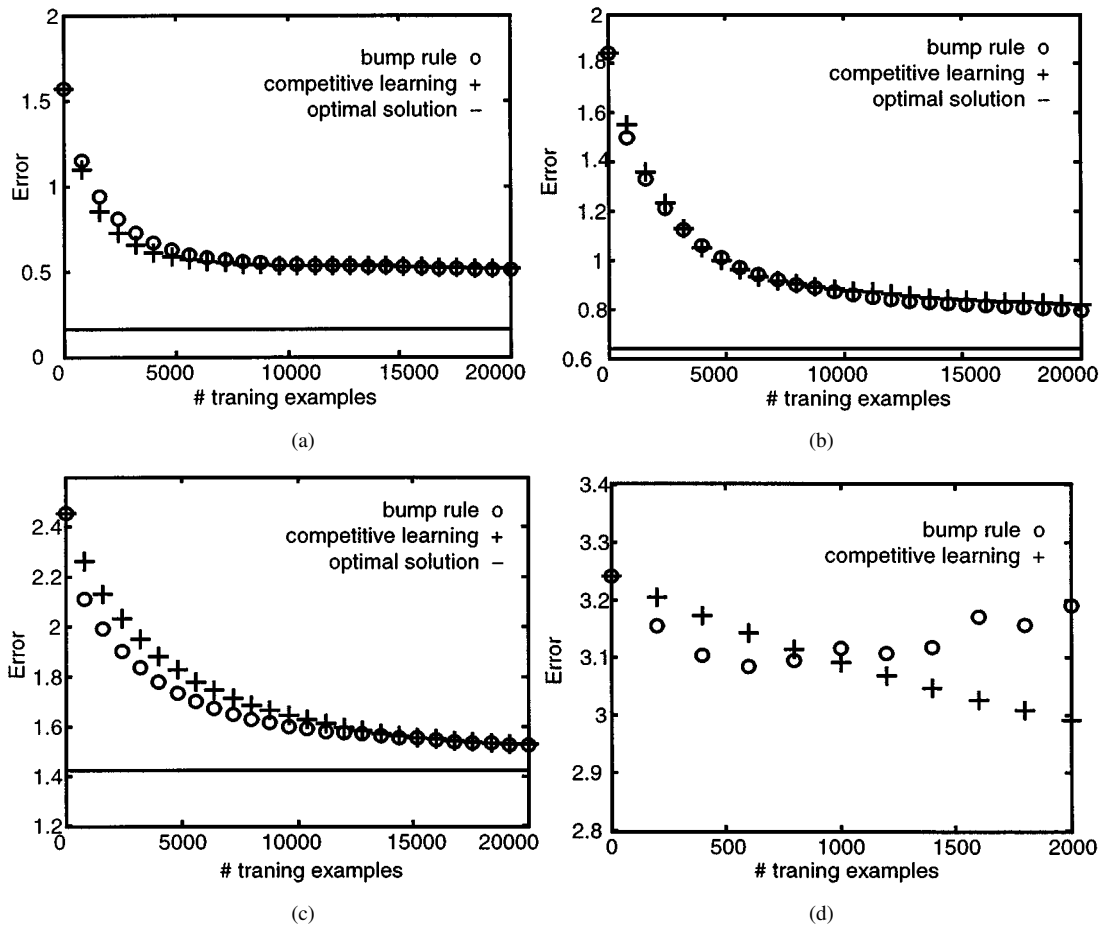

Fig. 16. Comparison of the standard competitive learning rule (+), bump-learning rule (o) and optimal solution (−) for learning the means of a mixture of Gaussians where each Gaussian had a diagonal covariance matrix and the standard deviation of each dimension was (a) 0.1 V, (b) 0.2 V, (c) 0.3 V, and (d) 0.4 V.

operation on Euclidean distance. Consequently, we believe that the correct way to combine similarities that have a Gaussian shape is through multiplication.

We express I_{mid} as a negative exponential of a distance function (similar to a Gaussian), by rewriting (8) as follows:

$$I_{\text{mid}} = I_b e^{-\Gamma(\Delta V)} \tag{22}$$

where $\Delta V = V_{\text{in}} - \mu$. Γ is given by

$$\Gamma(\Delta V) = \log\left(1 + \left(\frac{4}{S}\right)\cosh^2\left(\frac{\kappa \Delta V}{2U_t}\right)\right). \tag{23}$$

Fig. 14 compares this function with Manhattan distance and squared distance. Γ looks like squared distance when the input and bump weight are similar and like Manhattan distance when the two values are dissimilar. In contrast, there is no obvious way to extract a distance function from the addition of bump similarities.

Finally, we consider the qualitative behavior of adding bump similarities versus multiplying bump similarities. Because I_{mid} has a Gaussian-like shape, it only provides an informative distance measure for inputs close to the weight μ. However, $\log(I_{\text{mid}})$ provides an informative distance measure for any distance between the input and μ.

Fig. 15 illustrates one way to implement a multiplying neuron for our competitive-learning architecture. We use a subthreshold floating-gate MOSFET to multiply the currents from two bump circuits. We capacitively couple a floating gate to diode voltages derived from the bump-circuit currents. The MOSFET connected to this floating-gate implements a translinear multiply [25]. This approach requires an extra diode and capacitor for each bump.

VI. Performance of the Bump Learning Rule

We performed computer simulations to investigate the performance the bump learning rule on an unsupervised clustering task. We compared two versions of a standard competitive learning network. The first used the competitive learning rule of (2), with the learning rate set to 0.005. The second was a simulated version of the competitive architecture in Fig. 13, with synapses that both computed a bump similarity and adapted according to the bump learning rule of (20) and neurons that multiplied their synaptic outputs. We parameterized the bump learning rule with values from the fit in Fig. 9. Both networks performed hard competitive learning—only the neuron most similar to the input adapted.

We drew our input from a 16-dimensional input space with the data generated uniformly from a mixture of 16 Gaussians (i.e., for each data point, we randomly selected one Gaussian and generated the data point from its probability distribution function). For each Gaussian, we drew its mean's components

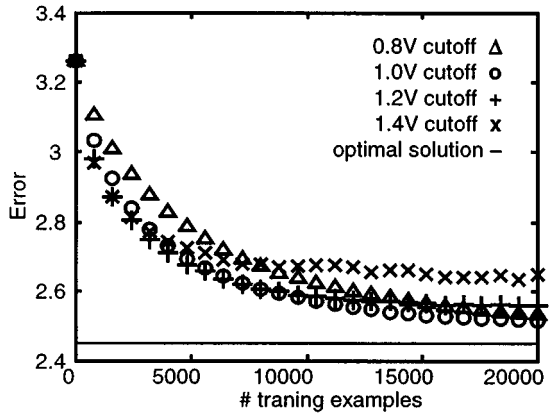

Fig. 17. To measure the effect of outliers on the bump circuit's performance, we altered the bump learning rule so that when $|x - \mu|$ exceeded a certain threshold, the magnitude of the bump adaptation did not increase. The results show that for values of $|x - \mu| < 1$, outliers did not degrade the bump circuit's performance.

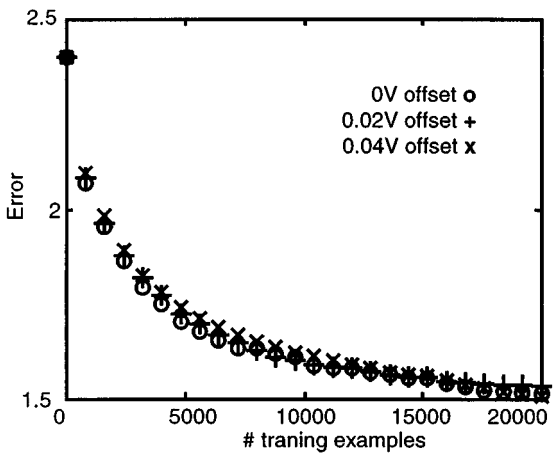

Fig. 18. Effect of tunneling mismatch on the bump learning rule for three mismatch values. We generated the training and test data from a mixture of Gaussians where each Gaussian has a covariance of $0.3^{*}I$. The results show that tunneling mismatch has a negligible effect on the bump learning rule's performance.

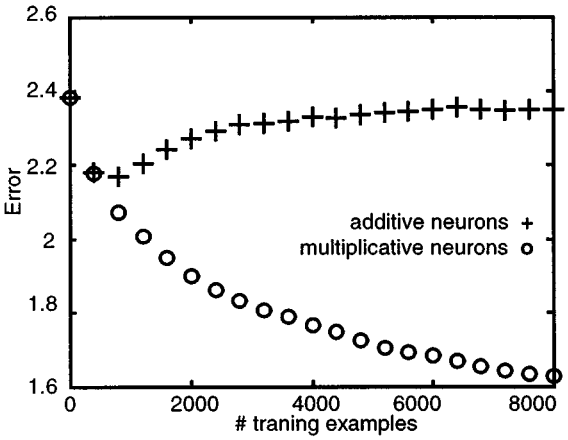

Fig. 19. Performance of a competitive learning network with bump circuit synapses using multiplicative or additive neurons.

randomly from the interval [0 V, 1 V]. Each Gaussian had a diagonal covariance matrix. We initialized the weight vectors for each network to the same starting point. Upon presenting each data point, we updated each network based on its respective learning rule.

We tested the two networks by evaluating the average coding error for each test instance, on test data drawn from the same set of Gaussians as the training data. The coding error is the squared distance between each test data point and the closest Gaussian mean. Fig. 15 shows the results averaged over ten trials, for Gaussians where the standard deviation of each dimension was 0.1 V (a), 0.2 V (b), 0.3 V (c), and 0.4 V (d). We also illustrate the coding error that the optimal mixture of Gaussians would produce. The data show that for mixtures of Gaussians with reasonable standard deviation Fig. 16(a)–(c), the bump rule's performance compares favorably with the basic competitive learning rule of (2). In the first three cases, both networks approach the error achieved by the correct mixture but asymptote before reaching the best solution. We believe that with a smaller learning rate or a slowly decaying learning rate, both learning rules can achieve the optimal solution.

Because the bump circuit's learning rule is exponential with respect to $(x-\mu)$, outliers can have a large effect on the learning dynamics. In our simulations, outliers did not become a problem until the standard deviation became larger than 0.3 V. Fig. 16(d) shows that the bump rule fails to converge if the standard deviation exceeds 0.4 V.

To test the sensitivity of the bump rule to outliers, we performed further tests on data drawn from Gaussians with standard deviation 0.4 V. We altered the rule so that if $|x - \mu|$ exceeded a preset threshold, the magnitude of the weight update did not increase. We used the same learning rate for these experiments as for the experiments in Fig. 16(a)–(c). We tested different thresholds using the same experimental procedure as in Fig. 16. We report these results in Fig. 17. The data show that outliers that are within 1.0 V of the bump circuit's weight do not degrade the bump learning rule. Therefore, if we restrict the input data to a 1.0 V interval, the exponential dynamics of bump adaptation do not degrade learning performance.

Our second experiment shows the effect of tunneling mismatch on the generalization performance of the bump learning rule. We used the same methodology from the experiment of Fig. 16 to compare bump learning with tunneling mismatches of 0 mV, 20 mV, and 40 mV (recall that the intrinsic tunneling offset in Fig. 7 was 18 mV). Fig. 18 shows the results for data generated from Gaussians with a standard deviation of 0.3 V. The results show that the bump rule is virtually unaffected by tunneling offsets.

Our third experiment compared the performance of two competitive learning architectures, one using multiplicative neurons and the other using additive neurons. Again, we used the same methodology in this experiment as in the previous two. We tested the networks on data drawn from Gaussians with standard deviation of 0.3 V. The results in Fig. 19 show that, as predicted in Section V, competitive-learning rules that use bump synapses perform better with multiplicative neurons than with additive neurons.

These experiments demonstrate that the bump circuit's learning rule is well suited for performing competitive learning

in silicon. In addition, the asymmetry due to tunneling mismatch does not greatly affect the performance of the learning rule.

VII. Conclusion

We have demonstrated an 11-transistor circuit primitive that incorporates a similarity function, nonvolatile storage and local adaptation; we can use this circuit in analog VLSI implementations of competitive learning. This circuit leverages synapse transistors to afford nonvolatile storage and to perform simultaneous adaptation and computation. The circuit's learning rule originates from the physics of the synapse transistors; consequently, it is unlike any rule that we know of in the literature. Even so, the learning rule provides the correct dynamics for competitive learning and exhibits symmetric adaptation. More importantly, software simulations show that the learning rule is as effective as the traditional competitive learning rule for clustering data drawn from a mixture of Gaussians.

To test our circuit primitive, we used it in a competitive learning circuit that clustered 1-D data. The circuit learned to discriminate data drawn from a mixture of Gaussians, providing evidence that this learning rule, while natural to the physics of our devices, is also useful in a learning context. This circuit is readily extensible to n-dimensional inputs and any number of neurons. We intend to use this circuit primitive and variations of the competitive learning architecture as building blocks for silicon chips that perform clustering and classification tasks.

Appendix I

The sizing of current-mirror transistors M_6-M_{11} determines whether injection increases or decreases the similarity between the floating-gate voltages of transistors M_1 and M_2 in Fig. 5. We performed a small-signal analysis of injection in M_1 and M_2, to determine sizing constraints on M_6-M_{11} that ensure injection will increase the similarity. Fig. 20 shows a small-signal model. We study the change in injection due to a small increase δI in M_1's source current (and the symmetric decrease δI in M_2's source current). The values for the circuit elements in Fig. 20 are

$$r_1 = \frac{1}{I_0\left(\frac{\kappa}{U_t}+\frac{1}{V_e}\right)} \tag{24}$$

$$r_2 = \frac{r_1}{\alpha} \tag{25}$$

$$g_1 = \frac{\alpha I_0 \kappa}{2U_t} \tag{26}$$

$$g_2 = \frac{-\left((1+\alpha)I_0\right)^{1-U_t/V_{\text{inj}}}}{V_{\text{inj}}} \tag{27}$$

$$g_3 = \frac{1-\frac{U_t}{V_{\text{inj}}}}{\left((1+\alpha)I_0\right)^{U_t/V_{\text{inj}}}} \tag{28}$$

where I_0 is the equilibrium current in diodes M_6 and M_9, α is the current-mirror gain, V_e is the Early voltage and v_1 and v_2 are the drain voltages of floating-gate transistors M_1 and M_2,

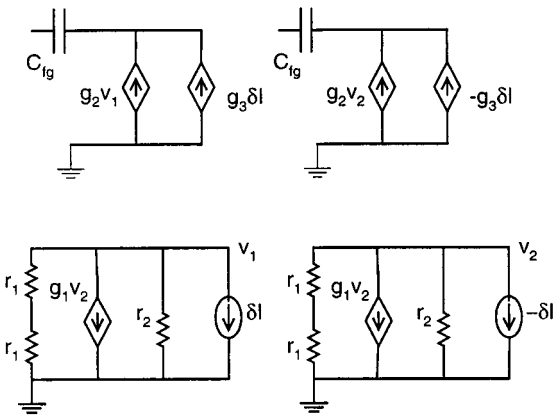

Fig. 20. Small signal model for injection in the bump circuit (transistors M_1-M_2, M_6-M_{11} in Fig. 5). Resistors r_1 model the diode connected nFETS. g_1 and r_2 model the current mirrors (M_7-M_8). The top level circuits model the IHEI dynamics of (7). We study the perturbation of the small signal model by an increase of δI in current on transistor M_1 and a decrease of δI on transistor M_2.

respectively. By inspection, $v_1 = -v_2$ and the small-signal injection in M_1 and M_2 is also equal in magnitude and opposite in sign (i.e., $g_2 v_1 + g_3\delta I = -g_2 v_2 + g_3\delta I$). To show that injection is stable, we must show that a positive δI in M_1 causes a decrease in M_1's injection. Stability requires

$$-g_2 v_1 > g_3 \delta I \tag{29}$$

where v_1 is given by

$$v_1 = \frac{\delta I}{\left(r_2 + \frac{r_1}{2} - g_1\right)}. \tag{30}$$

Substituting (24)–(29) into (30) and solving, we obtain the current-mirror gain that ensures stability

$$\alpha > \frac{\frac{V_{\text{inj}}}{2V_e} + \frac{\kappa V_{\text{inj}}}{2U_t} - \frac{1}{1-\frac{U_t}{V_{\text{inj}}}}}{\frac{1}{1-\frac{U_t}{V_{\text{inj}}}} - \frac{V_{\text{inj}}}{V_e} + \frac{\kappa V_{\text{inj}}}{2U_t}}. \tag{31}$$

If we substitute realistic values for V_{inj} and κ, the right-hand side of (32) is less than 0.1, ensuring that injection is stable for reasonable early voltages ($V_e > 1$ V).

Cross-coupled current mirrors $M_6 - M_{11}$ can exhibit hysteresis if the current-mirror gain is too high. To see why this can occur, consider the following scenario: Suppose the current-mirror gain is much greater than unity and, initially, I_1 is greater than I_2. Diodes M_6 and M_{11} will sink I_1 and mirror M_7 will sink I_2. In fact, M_7 can sink αI_1 of current and, therefore, if $\alpha > 1$, the mirror exhibits hysteresis. Hysteresis will not occur if $\alpha < 1$ because, in this case, if I_2 is greater than αI_1, diodes M_9-M_{10} will sink part of I_2, providing stabilizing feedback to the mirrors. A small signal analysis confirms that the circuit is stable and will not exhibit hysteresis if the mirror gain satisfies

$$\alpha < \frac{\frac{\kappa}{U_t} + \frac{1}{V_e}}{\frac{\kappa}{U_t} - \frac{2}{V_e}}. \tag{32}$$

APPENDIX II

We use a large-signal analysis to derive the bump circuit's IHEI weight-update rule. To begin, assume that M_1 sources more current than M_2 (the analysis is symmetric for M_2 sourcing more current than M_1). $d\Delta V_{\text{fg}}/dt$ due to IHEI derives from the difference between the injection currents in transistors M_1 and M_2 [see (11)]. We obtain the injection currents by substituting (7) into (15) and solving for the 1) source currents; 2) source voltages; and 3) drain voltages of M_1 and M_2.

A. Source Current

To derive I_s for M_1 and M_2, recall that transistors M_1 and M_2 form a differential pair on that fraction of I_b that is not part of I_{mid}. I_1 and I_2 in terms of ΔV_{fg} are

$$I_1 = I_0 \Phi\left(\Delta V_{\text{fg}}\right) e^{\kappa \Delta V_{\text{fg}}/2U_t} \tag{33}$$

$$I_2 = I_0 \Phi\left(\Delta V_{\text{fg}}\right) e^{-\kappa \Delta V_{\text{fg}}/2U_t} \tag{34}$$

where $\Phi(\Delta V_{\text{fg}})$ is

$$\Phi\left(\Delta V_{\text{fg}}\right) = \frac{I_b - I_{\text{mid}}}{I_0 2 \cosh\left(\frac{\kappa \Delta V_{\text{fg}}}{2U_t}\right)}. \tag{35}$$

B. Source Voltage

M_1 and M_2 share their source node, so V_s is the same for both. We can calculate V_s by equating (33) and (34) with the equivalent subthreshold expressions for a transistor's source current in terms of its source and gate voltages. We solve for e^{V_s/V_γ}, [see (7)]

$$e^{V_s/V_\gamma} = \Phi\left(\Delta V_{\text{fg}}\right)^{U_t/V_\gamma} e^{\kappa \delta V_0/V_\gamma}. \tag{36}$$

C. Drain Voltage

We solve for the drain voltages of M_1 and M_2 separately. We make the approximation that all of I_1 flows through M_6 and M_{11} and all of I_2 flows through M_7. Therefore, to solve for V_{d1} and V_{d2} (the drain voltages of M_1 and M_2, respectively), we equate (33) and (34) with the equivalent subthreshold expressions for an *n*FETs source current in terms of its gate and source voltages. The current in M_6 and M_{11} are

$$I_1 = I_0 e^{\kappa V_{d1}/2U_t} e^{-V_{\text{inj}}/U_t}. \tag{37}$$

We solve for e^{-V_{d1}/V_γ} by equating (33) and (37)

$$e^{-V_{d1}/V_\gamma} = \Phi\left(\Delta V_{\text{fg}}\right)^{2U_t/\kappa V_\gamma} e^{\Delta V_{\text{fg}}/V_\gamma} e^{-2V_{\text{inj}}/\kappa V_\gamma}. \tag{38}$$

We solve for V_{d2} using the following relationship:

$$I_2 = I_0 e^{\kappa V_{d1}/2U_t} \left(e^{-V_{\text{inj}}/U_t} - e^{-V_{d2}/U_t}\right). \tag{39}$$

We substitute I_1 into (39)

$$I_2 = I_1 e^{V_{\text{inj}}/U_t} \left(e^{-V_{\text{inj}}/U_t} - e^{-V_{d2}/U_t}\right). \tag{40}$$

Solving for e^{-V_{d2}/V_γ} from (40) we obtain

$$e^{-V_{d2}/V_\gamma} = \left(1 - e^{-\kappa \Delta V_{\text{fg}}/U_t}\right)^{-U_t/V_\gamma} e^{-V_{\text{inj}}/V_\gamma}. \tag{41}$$

Substituting (36), (38) and (41) into (7), we obtain

$$I_{\text{inj1}} = \alpha I_0^{1-U_t/V_\gamma} \Phi\left(\Delta V_{\text{fg}}\right)^{1-2U_t/\kappa V_\gamma} e^{\omega \Delta V_{\text{fg}}} e^{\varsigma \delta V_0} e^{-2V_{\text{inj}}/\kappa V_\gamma} \tag{42}$$

$$I_{\text{inj2}} = \alpha I_0^{1-U_t/V_\gamma} \Phi\left(\Delta V_{\text{fg}}\right) e^{-\sigma \Delta V_{\text{fg}}} e^{\varsigma \delta V_0} \times \left(1 - e^{-\kappa \Delta V_{\text{fg}}/U_t}\right)^{-U_t/V_\gamma} e^{-V_{\text{inj}}/V_\gamma} \tag{43}$$

where $\sigma = (1 - U_t/V_\gamma)^{\kappa/2U_t}$, $\omega = \kappa/2U_t - \kappa/2V_\gamma - 1/V_\gamma)$ and $\zeta = \kappa/V_\gamma$. Substituting (38) and (39) into (15), we obtain (16)–(18)

$$\frac{-d\Delta V_{\text{fg}}}{dt} = I_{j0} e^{\varsigma \delta V_0} \left(e^{\chi V_{\text{inj}}} \Phi_1\left(\Delta V_{\text{fg}}\right) - e^{\eta V_{\text{inj}}} \Phi_2\left(\Delta V_{\text{fg}}\right)\right) \tag{44}$$

where $I_{j0} = \left(\alpha I_0^{1-U_t/V_\gamma}\right)/C_{\text{in}}$, $\chi = -2/\kappa V_\gamma$ and $\eta = -1/V_\gamma$. The two functions, Φ_1 and Φ_2 are

$$\Phi_1\left(\Delta V_{\text{fg}}\right) = \Phi\left(\Delta V_{\text{fg}}\right)^{1-2U_t/\kappa V_\gamma} e^{-\omega \Delta V_{\text{fg}}} \tag{45}$$

$$\Phi_2\left(\Delta V_{\text{fg}}\right) = \Phi\left(\Delta V_{\text{fg}}\right) e^{-\sigma \Delta V_{\text{fg}}} \left(1 - e^{-\kappa \Delta V_{\text{fg}}/U_t}\right)^{-U_t/V_\gamma}. \tag{46}$$

This result is the bump circuit's weight-update rule (due to injection) shown in (16).

ACKNOWLEDGMENT

The authors would like to thank J. Nichols for help with layout and A. Doan, K. Partridge, M. Richardson, P. Tressel, A. Schon, and E. Vee for their suggestions and constructive criticism. Finally, the authors thank J. Dugger for initial discussions that led to this research.

REFERENCES

[1] M. A. Arbib, *The Handbook of Brain Theory and Neural Networks*, M. A. Arbib, Ed. Cambridge, MA: MIT Press, 1995.

[2] 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*. San Mateo, CA: Morgan Kauffman, 1989, vol. 1, pp. 703–711.

[3] 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*. San Mateo, CA: Morgan Kaufmann, 1994, vol. 6, pp. 882–887.

[4] S.-C. Liu, "A winner-take-all circuit with controllable soft max property," in *Advances in Neural Information Processing Systems*, S. A. Solla, T. K. Leen, and K. R. Muller, Eds. Cambridge, MA: MIT Press, 2000, vol. 12, pp. 717–723.

[5] S. J. Nowlan, "Maximum likelihood competitive learning," in *Advances in Neural Information Processing Systems*. San Mateo, CA: Morgan Kauffman, 1990, vol. 2, pp. 574–582.

[6] T. Kohonen, *Self Organizing Feature Maps*, 2nd ed. Berlin, Germany: Springer-Verlag, 1997.

[7] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[8] R. Cole *et al.*, *Survey of the State of the Art in Human Language Technology*, R. Cole *et al.*, Eds. Cambridge, U.K.: Cambridge Univ. Press, 1998.

[9] C. M. Bishop, *Neural Networks for Pattern Recognition*. Oxford, U.K.: Clarendon, 1995.

[10] H. C. Card, D. K. McNeill, and C. R. Schneider, "Analog VLSI circuits for competitive learning networks," *Analog Integrated Circuits and Signal Processing*, vol. 15, pp. 291–314, 1998.
[11] J. Lubkin and G. Cauwenberghs, "A learning parallel analog-to-digital vector quantizer," *IEEE J. Circuits, Syst., Comput.*, vol. 8, pp. 604–614, 1998.
[12] D. Macq, M. Verleysen, P. Jespers, and J. D. Legat, "Analog implementation of a Kohonen map with on-chip learning," *IEEE Trans. Neural Networks*, vol. 4, pp. 456–461, May 1993.
[13] Y. He and U. Cilingiroglu, "A charge-based on-chip adaptation Kohonen neural network," *IEEE Trans. Neural Networks*, vol. 4, pp. 462–469, May 1993.
[14] S. Fusi, M. Annunziato, D. Badoni, A. Salamon, and D. J. Amit, "Spike-driven synaptic plasticity: Theory, simulation, VLSI implementation," Neural Comput., 2000.
[15] C. Diorio, "A p-Channel MOS synapse transistor with self-convergent memory writes," *IEEE Trans. Electron Devices*, vol. 47, 2000.
[16] C. Diorio, P. Hasler, B. A. 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.
[17] C. Diorio, "Neurally Inspired Silicon Learning: From Synapse Transistors to Learning Arrays," Ph.D. dissertation, California Inst. Technol., Pasadena, 1997.
[18] P. Hasler, "Foundations of Learning in Analog VLSI," Ph.D. dissertation, California Inst. Technol., Pasadena, 1997.
[19] M. Lenzlinger and E. H. Snow, "Fowler-Nordheim tunneling into thermally grown SiO_2," *J. Appl. Phys.*, vol. 40, no. 1, pp. 278–283, 1969.
[20] E. Takeda, C. Yang, and A. Miura-Hamada, *Hot Carrier Effects in MOS Devices*. San Diego, CA: Academic, 1995.
[21] T. Delbruck, "Bump Circuits for Computing Similarity and Dissimilarity of Analog Voltages," California Inst. Technol., Pasadena, CNS Memo 26, 1993.
[22] R. R. Harrison, J. A. Bragg, P. Hasler, B. A. Minch, and S. P. Deweerth, "A CMOS programmable analog memory-cell array using floating-gate circuits," *IEEE Trans. Circuits. Syst. II*, vol. 48, pp. 4–11, 2001.
[23] P. Hasler, B. A. Minch, and C. Diorio, "An autozeroing floating-gate amplifier," *IEEE Trans. Circuits Syst. II*, vol. 48, pp. 65–73, 2001.
[24] J. Anderson, J. C. Platt, and D. B. Kirk, "An analog VLSI chip for radial basis functions," in *Advances in Neural Information Processing Systems*. San Mateo, CA: Morgan Kaufmann, 1995, vol. 7, pp. 765–772.
[25] B. A. Minch, C. Diorio, P. Hasler, and C. A. Mead, "Translinear circuits using subthreshold floating-gate MOS transistors," *Analog Integrated Circuits and Signal Processing*, vol. 9, pp. 167–179, 1996.

David Hsu received the B.S. degree from the University of California, Berkeley, in 1996 and the M.S. degree in computer science from the University of Washington, Seattle, in 2001. He is currently pursuing the Ph.D. degree at the same university.

His research includes VLSI design, machine learning and neural networks.

Miguel Figueroa received the B.S. degree, a professional degree, and the M.S. degree in electrical engineering from the University of Concepcion, Chile, in 1988, 1991, and 1997, respectively. He received the M.S. degree in computer science from the University of Washington, Seattle, in 1999 and is currently pursuing the Ph.D. degree at the same university.

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

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, in 1984 and 1997, respectively.

He is an Associate Professor of Computer Science and Engineering at the University of Washington. His research includes the interface of computing and biology and involves both building electronic circuits that mimic the computational and organizational principles used by nerve tissue and implanting electronic circuits into nerve tissue.

Dr. Diorio received a University of Washington Distinguished Teaching Award in 2001, an ONR Young Investigator Award in 2001, 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, an NSF CAREER Award in 1998 and the Electron Devices Society's Paul Rappaport Award in 1996. 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.