

A 19.2GOPS Mixed-Signal Filter with Floating-Gate Adaptation

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

Abstract—We have built a 48-tap, mixed-signal adaptive FIR filter with 8-bit digital input and an analog output with 10 bits of resolution. The filter stores its tap weights in nonvolatile analog memory cells using synapse transistors, and adapts using the Least-Mean-Square (LMS) algorithm. We run the input through a digital tapped delay line, multiply the digital words with the analog tap weights using mixed-signal multipliers, and adapt the tap coefficients using pulse-based feedback. The accuracy of the weight updates exceeds 13 bits. The total die area is 2.6mm^2 in a $0.35\mu\text{m}$ CMOS process. The filter delivers a performance of 19.2GOPS at 200MHz, and consumes 20mW providing a 6mA differential output current.

Index Terms—Adaptive signal processing, FIR filter, mixed-signal VLSI, floating-gate MOSFET.

I. INTRODUCTION

PORTABLE electronic systems normally operate under unknown or changing environmental conditions such as noise levels, interference, and varying input statistics. To combat these problems, they frequently employ adaptive signal-processing techniques to optimize their performance. Adaptive filtering is the most popular of these techniques: it is widely used in applications such as noise canceling, adaptive modeling, system identification, equalization, and linear predictive coding.

In application domains such as mobile communications or ubiquitous computing, adaptive systems also face severe constraints in power dissipation and circuit die area. In such cases, software implementations using programmable digital signal-processing (DSP) chips become infeasible. Even custom digital circuits can be prohibitively large and power-hungry, mainly due to the need for fast adders and multipliers. Although analog circuits can implement moderate-resolution arithmetic at low power and area, these circuits are limited by other problems such as charge leakage, signal offsets, circuit mismatch, error accumulation, and noise sensitivity.

We have built a mixed-signal, adaptive finite impulse-response (FIR) filter that combines the power and area benefits of analog circuits with the scalability of digital technology. The filter features digital inputs, analog weights with linear updates, and implements a pulse-

based version of the ubiquitous Least-Mean-Square (LMS) adaptation algorithm [1]. Each of the 48 taps computes a multiplication and an addition at every clock cycle, for an aggregated throughput of 19.2 Giga-Operations Per Second (GOPS) at 200MHz. The filter uses 2.6mm^2 of die area, consumes 20mW, and provides a 6mA differential output current. The input resolution is 8 bits, the output resolution is 10 bits, and the LMS circuits update the weights with more than 13 bits of accuracy. Our design improves upon other mixed-signal adaptive filters [2] by two orders of magnitude in power/performance ratio and one order of magnitude in die area. Our own previous 16-tap FIR filter [3] was small and low-power, but it was incapable of on-chip adaptation and provided only 7 bits of output resolution. The current design uses a nonvolatile weight-storage cell [4] based on *synapse transistors* [5], provides accurate weight-updates, and introduces a novel on-chip implementation of the LMS algorithm, thereby enabling closed-loop operation. Our design also features new mixed-signal multipliers, achieves an output resolution of 10 bits, and extends the length of the filter to 48 taps.

The remainder of this paper is organized as follows: first, we introduce synapse transistors as an enabling technology for compact and accurate weight-storage in adaptive hardware systems. Then, we describe our filter and the implementation of its fundamental building blocks. Finally, we discuss our experimental results and conclude.

II. SYNAPSE TRANSISTORS

A synapse transistor is a conventional MOSFET with three additional properties: nonvolatile analog weight storage, local weight update mechanisms, and simultaneous read/write operations. We build synapse transistors using floating-gate pFETs [5], where the charge stored on the floating gate represents the analog weight. Researchers have used synapse transistors to compute and store correlations [6, 7], perform unsupervised vector quantization [8], trim digital-to-analog converters [9] and null input offsets in a capacitive-feedback operational amplifier [10].

Fig. 1 shows a layout view of a synapse transistor in a double-poly process. The gate and field oxides isolate the poly 1 gate and provide nonvolatile charge-storage. The control gate capacitively couples to the floating gate, and

Manuscript received Xxxx xx, xxxx; revised Xxxx xx, xxxx

The authors are with the Department of Computer Science and Engineering, University of Washington, Box 352350, Seattle, WA 98105, USA (e-mail: miguel@cs.washington.edu)

Publisher Item Identifier X xxxx-xxxx(xx)xxxxx-x

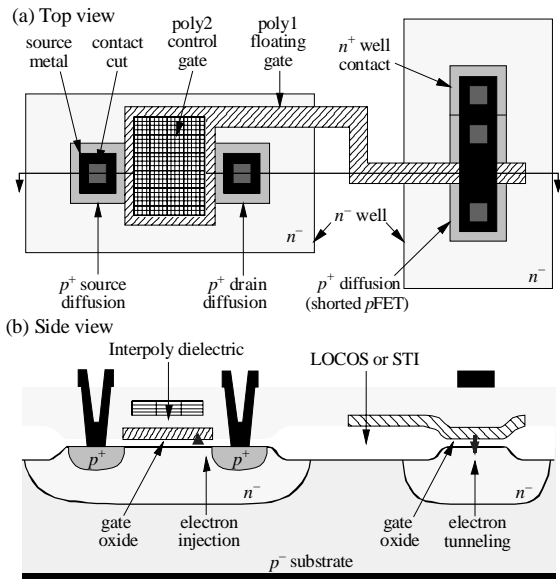
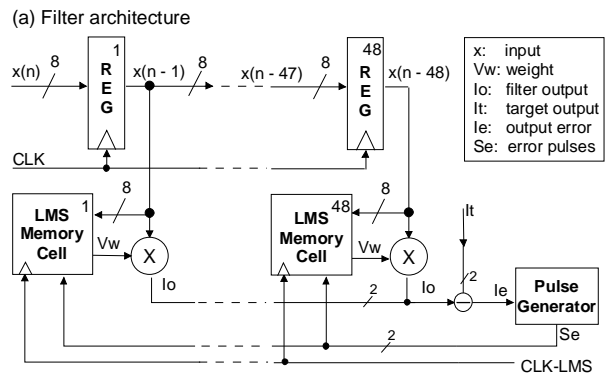


Fig. 1. (a) Top and (b) side layout view of a synapse transistor in a double-poly process, using an exaggerated vertical scale. The device stores a nonvolatile analog weight as charge on the floating gate. We increase the amount of charge by tunneling electrons off the floating-gate through the small pFET on the right. We decrease the charge by injecting electrons at the drain terminal.

can be built from poly 2 as in the figure, or using a MOS capacitor in a digital process. We increase the amount of charge stored on the floating gate using Fowler-Nordheim tunneling [11], with a small pFET as a tunneling junction. We decrease the amount of charge via impact-ionized hot-electron injection (IHEI) [12] by lowering the voltage at the drain terminal of the device. Synapse transistors and their charge-update dynamics are discussed in [5, 13]. In our filter, synapse transistors provide accurate and compact analog weight storage without the charge-injection and leakage problems commonly associated with VLSI capacitors.

III. THE FILTER

Fig. 2(a) shows the architecture of our filter and Fig. 2(b) shows a microphotograph of the chip core. We use a digital delay line to shift the 8-bit input signal across the filter taps, because offsets and error accumulation make long analog delay lines difficult to implement in VLSI. The delay line runs at 200MHz and uses standard master-slave flip-flops. Each tap contains a nonvolatile memory cell that stores an analog weight, and a mixed-signal multiplier that computes the product between the digital tap input x and the local tap weight voltage V_w , generating an analog differential current output I_o . These currents are summed across the filter on common wires to create the filter output. To minimize noise coupling, we isolate the analog section from the digital circuits with double guard rings and maintain separate analog and digital grounds.



(b) Chip microphotograph

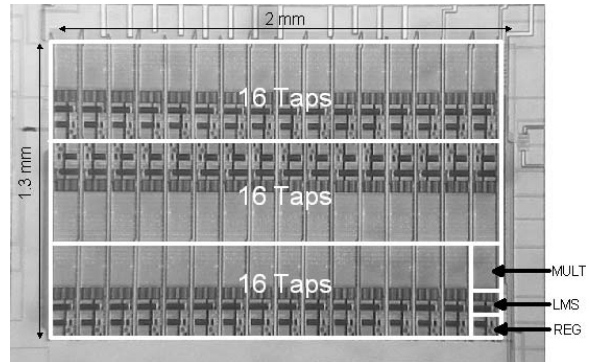


Fig. 2. (a) Adaptive 48-tap filter architecture. Each tap comprises a digital tap register, a mixed-signal multiplier, and a memory cell that stores an analog tap weight and implements LMS adaptation. A pulse generator produces a differential frequency-modulated digital pulse train (S_e), representing the filter error. (b) Microphotograph of the chip core in a $0.35\mu\text{m}$ CMOS process. The total die area is 2.6mm^2 . The mixed-signal multipliers use almost 50% of the area. The memory cells and update circuitry account for 25%. The delay line registers use the other 25%.

We encode the output target I_t as an analog differential current and use it to generate a differential error signal I_e . Current-driven pulse generators [14] (off-chip in the present implementation) generate a differential digital pulse-train S_e of fixed pulse-width, which encodes the instantaneous value of the error as the difference between the pulse frequencies of the differential components S_e^+ and S_e^- . At each tap, LMS circuits adapt the weight values by correlating this error signal with the tap inputs. The following subsections describe the implementations of our memory cells, LMS-update circuits and mixed-signal multipliers.

A. Analog Memory Cell

The charge-update dynamics of synapse transistors naturally lead to exponential adaptation rules and weight-dependent updates. While it is possible to design adaptive systems using these update dynamics [13], they are not suitable for adaptation using the LMS algorithm because, as shown in Eqn. (1) in subsection C, the LMS rule updates each tap weight by the (linear) product between the error and each tap input. Nonlinearities are acceptable if they are mild and the adaptation rule is a smooth, odd,

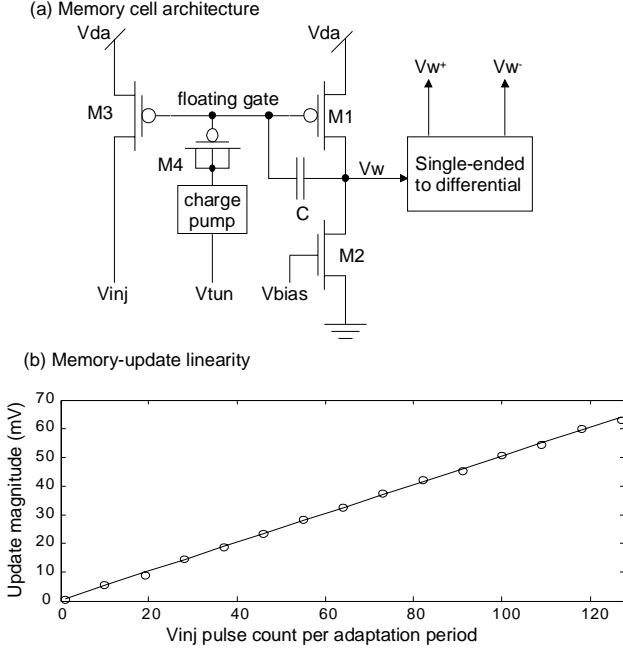


Fig. 3. (a) Memory-cell architecture. We update the floating-gate charge with pulses on V_{tun} and V_{inj} . M2 forces a constant current through M1, thereby pinning the floating-gate voltage. Capacitor C integrates the charge updates, causing V_w to change by $\Delta V_w = \Delta Q/C$. Because the floating-gate voltage is constant, feedback pulses of fixed width and amplitude change the floating-gate charge by constant amounts, causing fixed updates in V_w . (b) Measured linearity of the memory updates with respect to the frequency of V_{inj} . We obtain similar results for V_{tun} .

and monotonic function. In general, nonlinearities and weight dependences affect the stability of the adaptation and constrain the maximal adaptation rate. In addition, our system-level simulations showed that the resolution of the weight-updates directly affect the error performance of the filter. Therefore, our filter requires a memory cell that supports weight updates that are accurate, quasi-linear, and weight-independent.

Fig. 3(a) shows our analog weight cell, based on the design we presented in [4]. We store the weight as charge on the floating gate of a synapse pFET. Because the dynamics of tunneling and injection depend on the floating-gate voltage, we use negative feedback in an amplifier-like circuit (transistors M1-M2 and capacitor C) to keep this voltage at a constant value. M3 acts as an injection device: voltage pulses applied to its drain (V_{inj}) activate injection and add electrons to the floating gate. We remove electrons from the floating gate by applying pulses (V_{tun}) to the tunneling junction M4. The voltage at V_{bias} adjusts the relative strengths of tunneling and injection, and is tuned to yield symmetric weight updates at each tap in the presence of device mismatch.

The feedback capacitor C integrates the charge updates, modifying the output voltage V_w by $\Delta V_w = \Delta Q/C$. Because the floating-gate voltage is constant, update voltage-pulses of fixed width and amplitude change the output by a fixed amount.

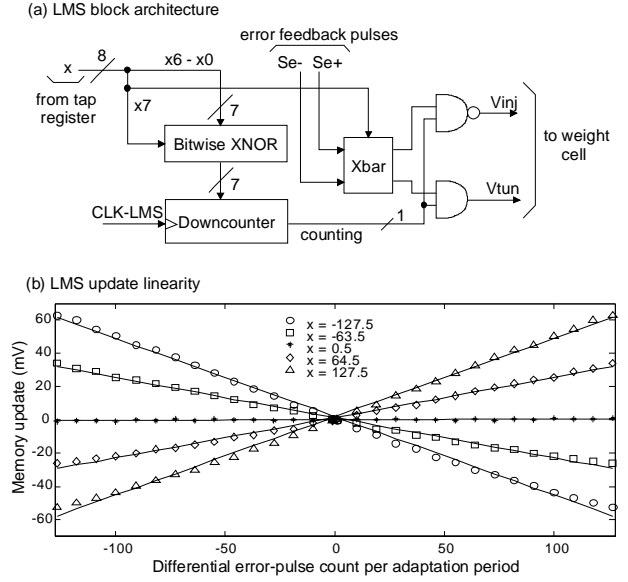


Fig. 4. (a) LMS-update architecture. The filter preloads a downcounter with the magnitude of the digital tap-input x . While the counter counts down, the error pulses (Se^+ and Se^-) drive tunneling and injection in the memory cell. The sign of x determines the polarity of the weight update. Therefore, the polarity and number of pulses updating the weight value during one adaptation period is the 4-quadrant product between the present input x and the present error Se . (b) Measured updates versus error pulse frequency e .

Therefore, the magnitude of the updates depends linearly on the frequency of the pulses V_{inj} and V_{tun} . Additionally, frequency-modulated digital updates are immune to jitter noise and signal degradation because the update pulses are restored and integrated at each tap. The experimental data in Fig. 3(b) shows the linear relationship between the weight updates and the number of pulses in V_{inj} over one adaptation period. Sweeping the frequency of V_{tun} yields similar results.

A single-ended to differential voltage converter transforms V_w into a differential signal (V_w^+ and V_w^-) with a fixed common mode. This differential output drives the mixed-signal multiplier that scales the weight by the tap input, as described in Section C.

B. LMS Block

The least-mean-square (LMS) algorithm [1] uses a gradient-descent method to update the weights of a linear filter or neural network. At each iteration, the algorithm updates the weights according to its adaptation rule:

$$w_i(n+1) = w_i(n) + \lambda x_i(n)e(n) \quad (1)$$

where w_i is the weight at tap i , λ is the adaptation rate, x_i is the value of the input at tap i , and e is the error. The LMS block in our filter computes this update by integrating pulses of frequency proportional to the error e , during a time window of length proportional to x_i .

Fig. 4(a) shows the implementation of the LMS-update block at each tap. We preload a digital downcounter with the magnitude of the tap input x (the lower 7 bits or their

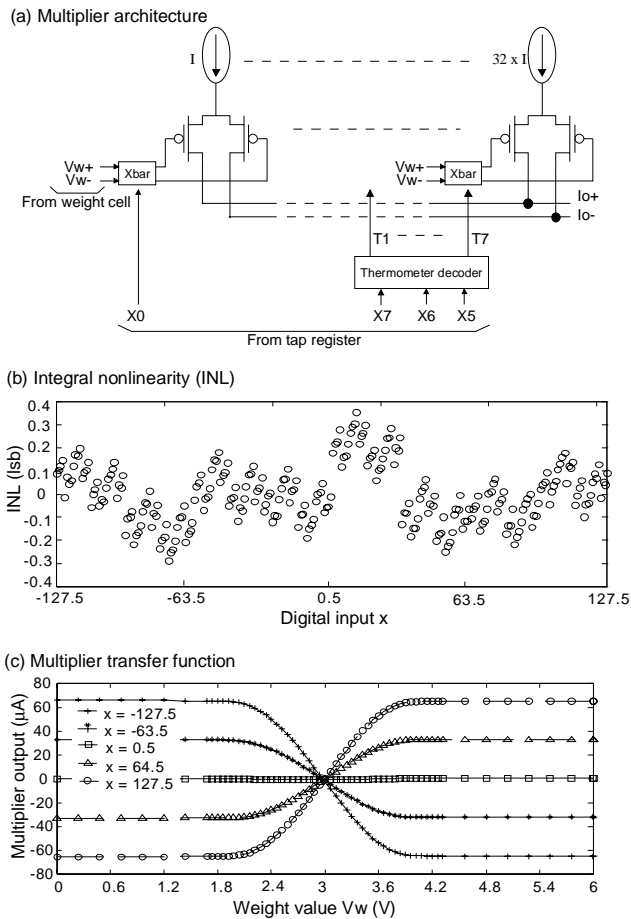


Fig. 5. (a) Mixed-signal multiplier cell, which comprises a segmented 8-bit DAC-like circuit with 5 binary and 3 thermometer bits, and an array of differential pairs that multiply the digital input word (x) by the differential output of the weight cell (Vw^+ and Vw^-). (b) Measured integral nonlinearity of the digital input in a typical multiplier. The INL and DNL are 0.35 and 0.4 LSBs, respectively. (c) Measured linearity of the weight in a typical multiplier. We do not have access to the differential weight, so we measured the single-ended representation Vw centered at 3V.

complement, because x uses an offset-binary code, which defines the time window during which we let the error pulses (S_e^+ and S_e^-) update the value of the weight memory cell (V_{inj} and V_{run}). The number of update pulses received by the memory cell is proportional to the product between the duration of the time window (the magnitude of the tap input x), and the frequency of the error pulses (the value of the error e). Finally, the crossbar $Xbar$ optionally inverts the sign of the updates based on the MSB of x , completing the full four-quadrant multiply of Eqn. (1). An external clock signal ($CLK-LMS$), independent of the delay-line clock (CLK), drives the downcounter and thereby modulates the adaptation rate. We can also adjust the gain of the pulse generators to control the adaptation rate.

Fig. 4(b) shows the weight update as a function of the error e (represented as a pulse count) for several values of the input x . We represent S_e^+ as a positive count and S_e^- as a negative count. Fig. 4(b) demonstrates that the memory-

update magnitude is an approximately linear function of the product of x and e , as required by Eqn. (1). The mild nonlinearity of the large updates is due to limitations of the amplifier used in the memory cell (transistors M1-M2 in Fig. 3), and can be corrected at a modest area cost by replacing it with a small operational amplifier. However, our system simulations and the theoretical analysis presented in [15] show that the error-performance of the filter is not affected by these mild nonlinearities, and therefore we kept the more compact design presented here.

C. Mixed-Signal Multiplier

The mixed-signal multiplier outputs a differential analog current that represents the product between the digital tap input and the analog tap weight. This current is summed across the taps to produce the filter output. Unlike the LMS block, this multiplier runs at the full speed of the tapped delay-line (200MHz) and ultimately determines the bandwidth of the filter.

Fig. 5(a) shows the 4-quadrant multiplier cell. We use a circuit that resembles a current-steered digital-to-analog converter (DAC), with an array of scaled current sources. The scaled currents pass through differential pairs that implement a saturating multiply. The differential input voltage to each pair (V_w^+ and V_w^-) represents the analog weight. The digital input x sets the polarity of the weight voltage at each pair. The circuit computes the expression:

$$I_o = k \times \sum_{i=0}^7 (-1)^{x_i+1} [x] V_w \times (2^{x_i}) \quad (2)$$

where $V_w = (V_w^+ - V_w^-)$ is the differential weight voltage, x_i are the bits of the digital tap input x , $I_o = (I_o^+ - I_o^-)$ is the output current, the operator $[x]$ represents the saturating multiply transfer function of the differential pair, and k is a scaling and unit-adjustment constant. Eqn. (2) is the offset-binary representation of the tap input, where the contribution of each bit is additionally multiplied by the analog weight.

We use standard current-source sizing techniques [16] to achieve 8-bit intrinsic resolution in the digital-to-analog conversion. We also use a thermometer decoder for the upper 3 bits to reduce glitch energy and achieve higher bandwidth. Fig. 5(b) shows the measured integral nonlinearity (INL) of a typical multiplier as a function of the digital input x . Both the INL and the differential nonlinearity (DNL, not shown) are less than 0.5LSB. The current sources and differential pairs occupy 80% of the multiplier area. We can reduce this area by nearly an order of magnitude and increase the multiplier resolution in future implementations using the on-chip trimming techniques that we demonstrated in [9].

Fig. 5(c) shows the multiplier output as a function of the weight value, for several digital input codes. The

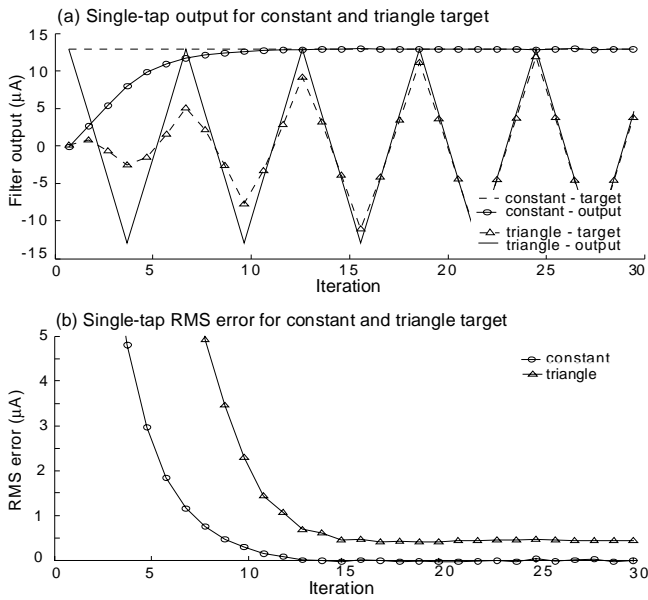


Fig. 6. Single-tap error performance for a constant and triangle-wave target and input. (a) Target and filter output for both experiments. (b) RMS value of the filter error. With a constant target, the error settles to 10nA (13-bit output resolution). With the triangular target, the error settles to 0.5µA (8-bit output resolution). As discussed in the text, the experiment shows that the error performance of the filter is limited by the resolution of the mixed-signal multipliers.

analysis in [15] and our own system simulations show that this multiplier provides adequate linearity for LMS adaptation for a weight range of up to 1V differential. This is corroborated by our experimental results in Section IV.

IV. EXPERIMENTAL RESULTS

A. Single-Tap Performance

In our first experiment, we seek to evaluate the impact on the error performance of the filter due to the resolution of the memory cell and LMS adaptation circuits versus the impact of the mixed-signal multipliers. First, we enable a single tap in the filter, set a DC-valued digital input and target, and let the filter adapt. In this setup, the current-source mismatch in the mixed-signal multiplier has no influence on the output error because the inputs to the differential pairs do not switch. Next, we use a triangle-wave input and target and again let the filter adapt. In this case, because the digital input switches at 200MHz, the current-source mismatch in the multiplier introduces a high-frequency error that the adaptation is unable to cancel. Fig. 6(a) shows the evolution output and reference for both setups, and Fig. 6(b) shows the RMS value of the error. After a few adaptation periods, the error settles to 10nA RMS for the first case (this measured accuracy is limited by our experimental setup). For a 128µA single-tap output range, this error corresponds to an output resolution better than 13 bits. This resolution is ultimately limited by the tradeoff

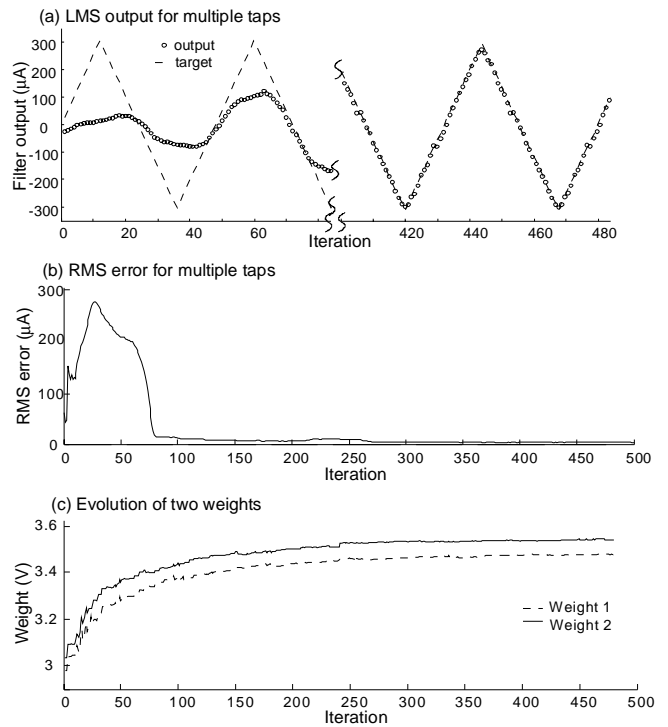


Fig. 7. LMS performance with 24 taps. (a) Target and output during the first 80 and last 80 iterations. (b) RMS error. After 480 iterations, the error is 5µA, settling at 2µA (equivalent to 10-bit output resolution) after 300 additional iterations. (c) Outputs of two memory cells learning the same theoretical weight value. The LMS algorithm compensates for mismatch across cells, so that each cell converges to a voltage that represents the same nominal weight.

between the dynamic range of the weights and the magnitude of the update performed by a single pulse. In the second case, the error settles to 0.5µA, corresponding to a resolution of 8 bits. This is consistent with the 8-bit matching of the current sources in the mixed-signal multiplier, and demonstrates that the error-performance of our current implementation is limited by the multipliers and not by the linearity of the memory cell and weight updates. As stated in Section III.C, we can improve the error-performance of the filter at negligible additional cost in power and area, using the on-chip trimming techniques that we demonstrated in [9].

B. Adapting with Multiple Taps

In our second experiment, we enable 24 taps and train the filter to output a triangle wave given a square-wave input. Fig. 7(a) shows the target and output during the first 80 and the last 80 iterations. Fig. 7(b) shows the RMS error during adaptation. After 480 iterations, the error is about 5µA (full output range is $24 \times 128 \mu\text{A}$). As stated before, the error performance is limited by the resolution of the mixed-signal multipliers. Because the effects of device mismatch in the multipliers are largely uncorrelated across the filter, the output picks up two more bits of resolution, settling at slightly more than 10 bits (2µA RMS error) after an additional 300 iterations.

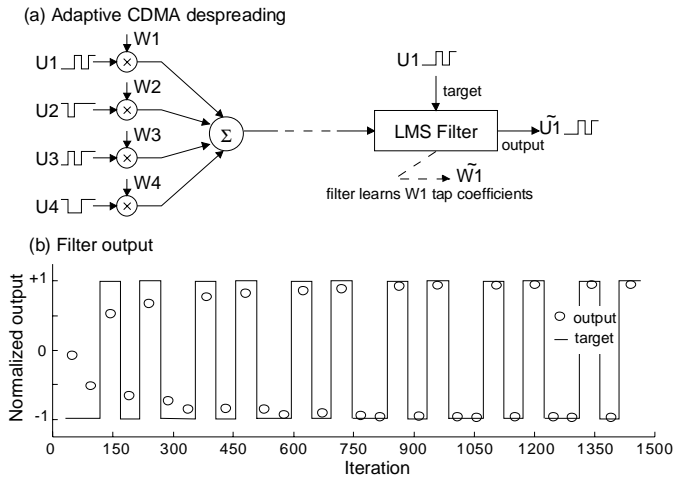


Fig. 8. LMS performance on an adaptive CDMA despadding application. (a) The experiment. We generated bit streams for 4 simultaneous CDMA users ($U1-U4$) and encoded them using 16-chip orthogonal Walsh codes ($W1-W4$). We combined the user chip streams, oversampled the combined signal by a factor of 3, and fed the resulting signal to the 48-tap filter. We provided $U1$'s bit stream as the target and let the filter learn the appropriate despadding code $W1$. (b) Evolution of the output normalized to the amplitude of the reference. The filter learns to correctly discriminate the user's bit stream after only a few iterations.

The convergence speed is limited by the properties of the LMS algorithm, and can be accelerated using an adaptive input decorrelating stage as shown in [17]. We can easily build this input stage using the same adaptation and computational primitives that we developed for the LMS filter. Fig. 7(c) illustrates an attractive benefit of adaptation. Device mismatch causes variations in memory-cell offsets and multiplier gains, therefore two weights of identical nominal value have different single-ended voltage representations in their memory cell. This would normally impact the resolution of an open-loop system, but the LMS adaptation naturally compensates for the mismatch so that the memory cells converge to the different voltage outputs that represent the correct nominal value.

C. Adaptive-DS-CDMA Despadding

As a final experiment, we enable all 48 taps and use the adaptive filter in a direct-sequence code-division multiple-access (DS-CDMA) adaptive despadding application [18]. Fig. 8(a) shows the experiment, where four users share a CDMA channel. We encode each user's bit stream U_i with orthogonal 16-chip Walsh spreading codes W_i (signatures), and add the chip streams to form a composite signal. Traditionally, a receiver recovers the user's bit stream by correlating the composed chip stream with the user's signature. However, in the presence of multipath fading and multiple-user interference, the optimal signature to recover the original bit stream deviates from the original user's spreading code, consists of non-binary values and

is unknown to the receiver [19]. An effective approach to solve this problem is to learn the optimal signature using an adaptive filter to recover the bit stream. The target can be externally provided via a training sequence, or it can be estimated using a decision-feedback approach. We input the composite signal (oversampled by a factor of 3) to the 48-tap adaptive filter, and provide $U1$'s bit stream as a reference. The task of the filter is to learn the optimal spreading code \tilde{w}_1 and produce $U1$'s original bit stream.

Fig. 8(b) shows the reference bit stream and the filter's output, sampled after each complete bit frame. Because the reference is a binary sequence, a simple comparator can generate the user's bitstream from the filter's analog output and feed it back as the reference. The figure shows that the filter learns to discriminate the bit stream after only a few iterations. Furthermore, as the adaptation progresses, the amplitude of the output becomes larger, improving the receiver's interference- and noise-rejection characteristics to the maximal filter resolution of 10 bits.

To provide a reference for comparison, we designed a fully-digital version of the adaptive filter in the same CMOS process and with similar performance. The digital design occupies more than twice the die area and consumes more than two orders of magnitude more power than our mixed-signal implementation based on synapse transistors. Other hardware implementations of CDMA despadding [20, 21] use less power and area than our filter, but they rely fundamentally on non-adaptive, strictly binary user signatures to avoid the use of multipliers. These implementations can only operate in open loop and thus are not suitable for the blind adaptive detection application that we have illustrated here.

V. CONCLUSION

We built a 48-tap, 19.2GOPS, 20mW, 2.6mm² filter that adapts its weights using the LMS algorithm with an output resolution of 10 bits. Synapse transistors enable compact, low-power analog memory cells without charge leakage. This is particularly important in supervised-learning applications, where the filter operates in open loop after a training period. Moreover, because the mechanisms that we use to update the weights are immune to charge injection, we can use digital pulse-based adaptation, which results in accurate weight updates. As a result, our filter can adapt with more than 13 bits of accuracy. In fact, the error-performance of the filter is limited by the current-source mismatch in our mixed-signal multipliers, which we can improve at low cost using proven on-chip trimming techniques. The same techniques can also drastically reduce the die area of the filter. Combining analog and digital technology in the implementation of this filter enables high performance, low power consumption, good scalability and compatibility with standard digital CMOS processes.

REFERENCES

- [1] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.
- [2] M. Q. Le, P. J. Hurst, and J. P. Keane, "An Adaptive Analog Noise-Predictive Decision-Feedback Equalizer," presented at Symposium on VLSI Circuits, Honolulu, Hawaii, 2000.
- [3] M. Figueroa, D. Hsu, and C. Diorio, "A Mixed-Signal Approach to High-Performance, Low-Power Linear Filters," *IEEE Journal of Solid-State Circuits*, vol. 36, pp. 816-822, 2001.
- [4] C. Diorio, S. Mahajan, P. Hasler, B. A. Minch, and C. Mead, "A High-Resolution Nonvolatile Analog Memory Cell," presented at IEEE Intl. Symp. on Circuits and Systems, 1995.
- [5] C. Diorio, P. Hasler, B. Minch, and C. Mead, "A Complementary Pair of Four-Terminal Silicon Synapses," *Analog Integrated Circuits and Signal Processing*, vol. 13, pp. 153-166, 1997.
- [6] P. Hasler and J. Dugger, "Correlation Learning Rule in Floating-Gate pFET Synapses," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 65-73, 2001.
- [7] A. Shon, D. Hsu, and C. Diorio, "Learning Spike-Based Correlations and Conditional Probabilities in Silicon," presented at Neural Information Processing Systems (NIPS), Vancouver, BC, 2001.
- [8] D. Hsu, M. Figueroa, and C. Diorio, "Competitive Learning with Floating-Gate Circuits," *IEEE Transactions on Neural Networks*, vol. 13, pp. 732-744, 2002.
- [9] J. Hyde, T. Humes, C. Diorio, M. Thomas, and M. Figueroa, "A Floating-Gate Trimmed, 14-bit, 250 MS/s Digital-to-Analog Converter in Standard 0.25 μ m CMOS," presented at Symposium on VLSI Circuits, 2002.
- [10] P. Hasler, B. Minch, and C. Diorio, "An Autozeroing Floating-Gate Amplifier," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 48, pp. 74-82, 2001.
- [11] M. Lenzlinger and E. H. Snow, "Fowler-Nordheim Tunneling into Thermally Grown SiO₂," *Journal of Applied Physics*, vol. 40, pp. 278-283, 1969.
- [12] E. Takeda, C. Yang, and A. Miura-Hamada, *Hot Carrier Effects in MOS Devices*. San Diego, CA: Academic Press, 1995.
- [13] C. Diorio, D. Hsu, and M. Figueroa, "Adaptive CMOS: from Biological Inspiration to Systems-on-a-Chip," *Proceedings of the IEEE*, vol. 90, pp. 345-357, 2002.
- [14] C. Mead, *Analog VLSI and Neural Systems*. Reading, MA: Addison-Wesley, 1989.
- [15] B. K. Dolenko and H. C. Card, "Tolerance to Analog Hardware of On-Chip Learning in Backpropagation Networks," *IEEE Transactions on Neural Networks*, vol. 6, pp. 1045-1052, 1995.
- [16] M. J. M. Pelgrom, A. C. J. Duinmaijer, and A. P. G. Welbers, "Matching Properties of MOS Transistors," *IEEE Journal of Solid-State Circuits*, vol. 24, pp. 1433-1440, 1989.
- [17] F. Palmieri, J. Zhu, and C. Chang, "Anti-Hebbian Learning in Topologically Constrained Linear Networks: A Tutorial," *IEEE Transactions on Neural Networks*, vol. 4, pp. 748-761, 1993.
- [18] R. Lupas and S. Verdu, "Linear Multiuser Detectors for Synchronous Code-Division Multiple Access Channels," *IEEE Transactions on Information Theory*, vol. IT-35, 1989.
- [19] M. Honig, U. Madhow, and S. Verdú, "Blind Adaptive Multiuser Detection," *IEEE Transactions on Information Theory*, vol. 41, pp. 944-960, 1995.
- [20] D. Senderowicz, S. Azuma, H. Matsui, K. Hara, S. Kawama, Y. Ohta, M. Miyamoto, and K. Iizuka, "A 23mW 256-Tap 8MSample/s QPSK Matched Filter for DS-CDMA Cellular Telephony Using Recycling Integrator Correlators," presented at International Solid-State Circuits Conference (ISSCC), San Francisco, CA, 2000.
- [21] T. Yamasaki, T. Fukuda, and T. Shibata, "A Floating-Gate-MOS-Based Low-Power CDMA Matched Filter Employing Capacitance Disconnection Technique," presented at IEEE Symposium on VLSI Circuits, Kyoto, Japan, 2003.