A 19.2GOPS, 20mW Adaptive FIR Filter

Miguel Figueroa, Seth Bridges, David Hsu and Chris Diorio

Computer Science and Engineering, University of Washington
Box 352350, Seattle, WA 98195-2350, USA
Phone: +1-206-543-7119, Fax: +1-206-543-2969, Email: {miguel, seth, hsud, diorio}@cs.washington.edu

Abstract

We implemented a 48-tap, mixed-signal adaptive FIR filter with 8-bit input and 10-bit output resolution. The filter stores its tap weights in nonvolatile analog memory cells 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 0.35μm CMOS process. The filter delivers a performance of 19.2GOPS at 200MHz, and consumes 20mW providing a 6mA differential output current.

1. Introduction

Many modern-day electronic systems must deal with unknown or changing environmental variables such as noise levels, interference, and varying input statistics. These systems frequently use adaptive signal-processing techniques to optimize their performance. However, in application domains such as mobile communications or ubiquitous computing, these systems also face severe constraints in power dissipation and circuit die area. In such cases, using programmable digital signal-processing (DSP) chips becomes 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 with the scalability of digital. The filter uses synapse transistors [1] to store its analog weights, provides linear weight updates, and implements a pulse-based version of the Least-Mean-Square (LMS) adaptation algorithm [2]. Each of the 48 taps computes a multiplication and an addition on 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, and consumes 20mW with a 6mA differential output current. The input resolution (delay-line width) is 8 bits, and the LMS circuitry updates the weights with more than 13 bits of accuracy.

Our design improves on past mixed-signal adaptive filters [3] by two orders of magnitude in power/performance ratio and one order of magnitude in die area. Our previous FIR filter design [4] was incapable of on-chip adaptation and provided only 7 bits of output resolution. The current design uses a weight-storage cell [5] with accurate updates and introduces a novel on-chip implementation of the LMS algorithm, thereby enabling closed-loop operation. Our design also introduces new mixed-signal multipliers, achieves an output resolution of 10 bits, and extends the length of the filter to 48 taps.

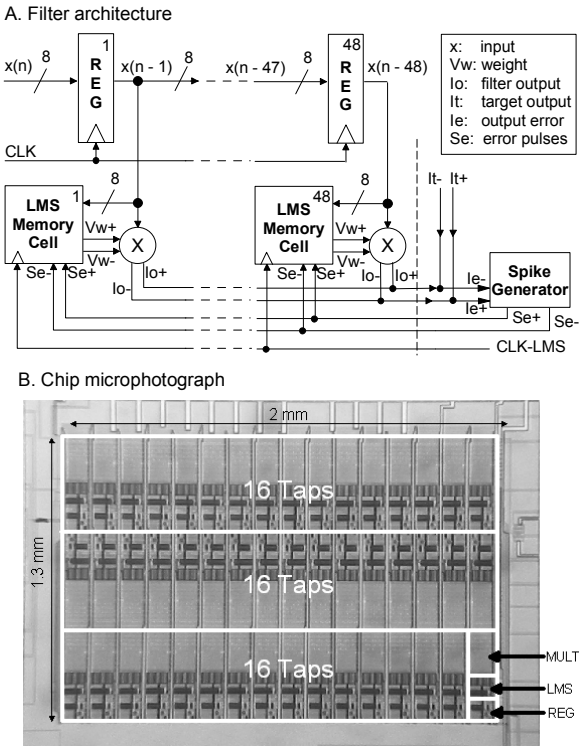

Fig. 1. The adaptive filter. Part A shows the 48-tap filter architecture. Each tap comprises a digital tap register, a mixed-signal multiplier, and a memory cell that stores an analog tap weight and implements LMS adaptation. A spike generator produces a differential frequency-modulated digital pulse train (Se^+ and Se^-), representing the filter error. Part B shows a microphotograph of the chip core in a 0.35μm CMOS process available from MOSIS. The total die area is 2.6mm^2.

2. The Filter

An FIR filter computes a convolution between an input data stream and a stored weight vector. Fig. 1(A) shows the architecture of our FIR filter. It comprises a digital delay line, analog weight cells, pulse-based digital LMS adaptation circuitry, and mixed-signal multipliers with differential current outputs (Io^+ and Io^-). We use an 8-bit

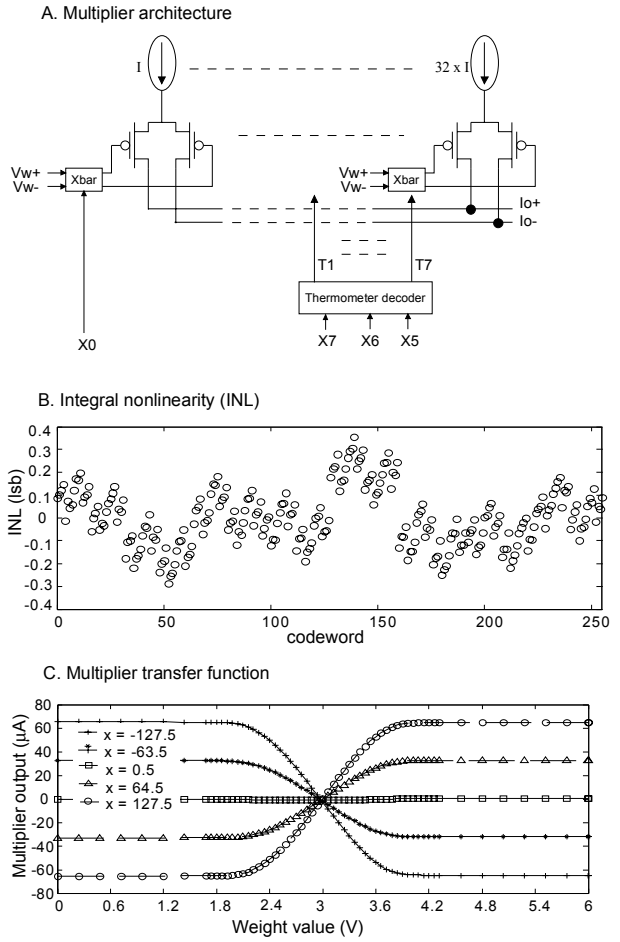

Fig. 2. The mixed-signal multiplier. Part A shows the multiplier cell, which comprises a segmented 8-bit DAC-like circuit with 5 binary and 3 thermometer bits, and an array of differential pairs that multiply the digital input word (x) by the differential output of the weight cell (Vw^+ and Vw^-). Part B shows the measured integral nonlinearity of the digital input in a typical multiplier. The INL and DNL are 0.35 and 0.4 LSBs, respectively. Part C shows the 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 centered at 3V.

200MHz digital delay line to shift the input signal across the filter taps, because offsets and error accumulation make long analog delay lines difficult to implement in VLSI. The error signal is a differential current (Ie^+ and Ie^-). We generate this error signal by subtracting the filter output from the target signal (It^+ and It^-). A spike generator [6] converts the error into a differential frequency-modulated digital pulse train with fixed pulse-width (Se^+ and Se^-) and the filter adapts the tap weights by correlating this error signal with the tap inputs.

Fig. 1(B) shows a microphotograph of the chip core. The multipliers use 50% of the total area, the memory cells and LMS circuitry occupy 25%, and the digital delay line uses the other 25%. The following sections describe the main blocks of the filter in more detail.

3. Mixed-Signal Multiplier

Fig. 2(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 (Vw^+ and Vw^-) represents the analog weight. The digital input x sets the polarity of the weight voltage at each pair.

We use standard current-source sizing techniques to achieve 8-bit intrinsic resolution. Fig. 2(B) shows the measured integral nonlinearity (INL) of a typical multiplier. Both INL and the differential nonlinearity (DNL) are less than 0.5LSB. The current sources occupy 80% of the multiplier area. We can reduce this area in future chips using the on-chip trimming techniques that we demonstrated in [7]. The same techniques can also increase the multiplier resolution.

Fig. 2(C) shows the multiplier output as a function of the weight value, for several digital input codes. The analysis in [8] shows that this multiplier provides adequate linearity for LMS adaptation for a weight range of 1V differential. This is corroborated by our experimental results in Section 6.

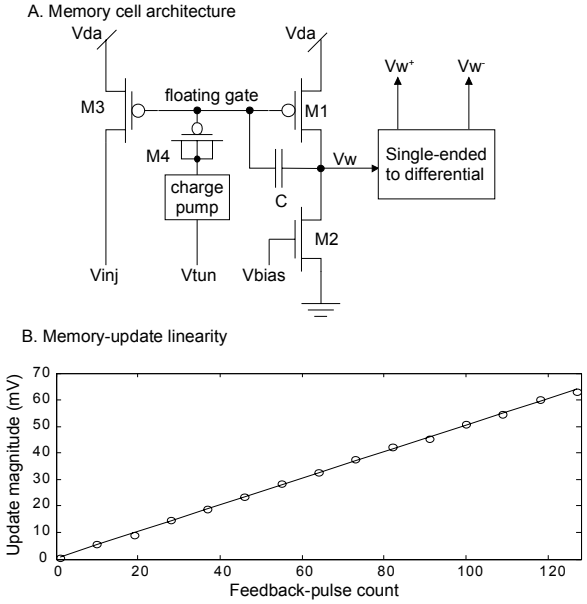

Fig. 3. Part A shows the memory-cell architecture. We store a weight as a nonvolatile analog charge on the floating gate. We update the charge using Fowler-Nordheim tunneling and hot-electron injection, controlled by pulses on *Vtun* and *Vinj*, respectively. M2 is a current source that forces a constant current through M1, thereby pinning the floating-gate voltage. Capacitor C integrates the charge updates, causing Vw to change by an amount $\Delta Vw = \Delta Q/C$. Because the floating-gate voltage is constant, feedback pulses of fixed width and amplitude change the charge on the floating gate by constant amounts, causing fixed updates in Vw. Part B shows the measured linearity of the memory updates with respect to the frequency of *Vinj*. We obtain similar results for pulses on *Vtun*.

4. Analog Memory Cell

Fig. 3(A) shows our analog weight cell, based on the design we presented in [5]. We store each filter coefficient as charge on a floating gate, and update the charge using Fowler-Nordheim tunneling and impact-ionized hot-electron injection [1]. Tunneling and injection naturally produce weight updates that are highly nonlinear and dependent on the weight value [9].

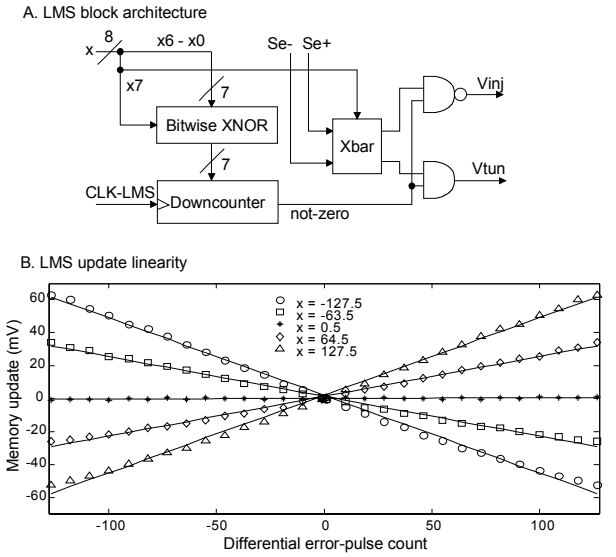

Fig. 4. Part A shows the LMS-update architecture. The filter preloads a downcounter with the magnitude (7 lower bits) of the digital tap-input x. While the counter counts down, the error pulses (Se^+ and Se^-) are transformed into update pulses to drive tunneling and injection in the memory cell. The sign of x, given by the MSB $x7$, determines the polarity of the weight update. Therefore, the polarity and number of pulses updating the weight value depends on both the present input x and the present error (Se^+ and Se^-). Part B shows measured updates versus error pulse frequency e. The update is the 4-quadrant multiplication of the error signal e by the input value x.

Our weight cell provides weight-independent linear updates as needed for LMS adaptation. We use feedback in an amplifier-like circuit to pin the floating-gate voltage, and integrate the floating-gate updates (i.e., charge) on the feedback capacitor C. The charge updates modify the weight value Vw.

The weight updates have a linear dependency on the frequency of the digital feedback pulses. We activate tunneling by applying pulses to the *Vtun* terminal; a simple charge pump boosts the pulse voltage to 11V and causes electron tunneling through M4's gate oxide. Active-low *Vinj* pulses inject hot electrons from M3's drain onto the floating gate. The voltage at *Vbias* sets the floating-gate voltage, and thereby adjusts the relative strengths of tunneling and injection. We tune this bias to yield symmetric weight-update rates at each tap.

Fig. 3(B) shows measured values of Vw updates as a function of the frequency of *Vinj* pulses. A single-ended to differential converter transforms Vw into the differential voltage that drives the multiplier.

5. LMS Block

The least-mean-square (LMS) algorithm [2] 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 the equation

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

where w_i is the weight at tap i, λ is the learning rate, x_i is the value of the input at tap i, and e is the error.

Fig. 4(A) shows our implementation of the LMS algorithm. We represent the error signal e as a differential frequency-modulated train of digital pulses

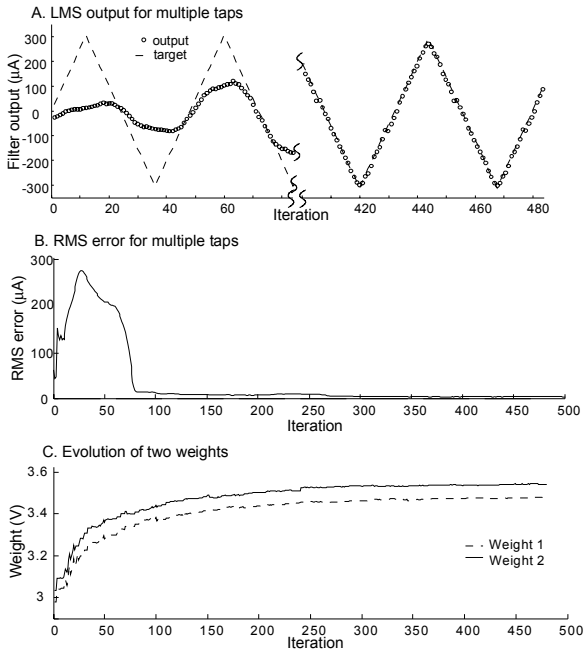

Fig. 5. LMS performance with 24 taps. Part A shows the target and the output during the first 80 and last 80 iterations. Part B shows the RMS error. After 480 iterations, the error is 5μA, settling at 2μA (equivalent to 10-bit output resolution) after 300 additional iterations. Part C shows the outputs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

(Se^+ and Se^-). We use these pulses to update the value stored in the weight cell. At the beginning of each LMS iteration, we preload a digital downcounter with the magnitude (the lower 7 bits) of the current tap input x. An external clock signal (*CLK-LMS*) drives the downcounter. This clock is independent of the delay-line clock and its frequency modulates the learning rate λ of Eqn. (1). The countdown defines a time window proportional to the magnitude of x, and the error pulses update the weight memory for the duration of that window. Hence, the number of update pulses that the weight cell receives during each LMS iteration is proportional to both the magnitude of x and the difference between the frequencies of Se^+ and Se^-. The sign bit of x determines the sign of the updates (i.e., tunneling or injection). We control the learning rate λ by adjusting either the gain of the current-to-spike frequency generators or the frequency of the LMS clock.

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 Se^+ as a positive count and Se^- as a negative count. Fig. 4(B) demonstrates that the memory-update magnitude is a linear function of the product of x_i and e, as required by Eqn. (1).

6. Experimental results

For our first on-line adaptation experiment, we enabled a single tap in the filter, set a DC-valued digital input and target, and let the filter adapt. The purpose of this experiment was to evaluate the resolution of the weight updates, isolated from the effect of input quantization errors. After 15 iterations, the error settled

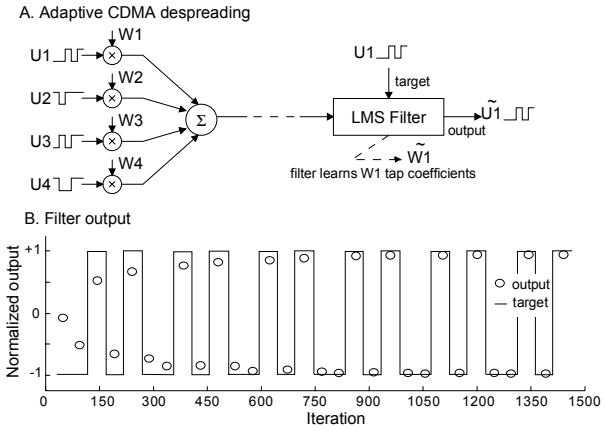

Fig. 6. LMS performance on an adaptive CDMA despreading application. Part A: We generated bit streams for 4 simultaneous CMDA users (*U1–U4*) and encoded them using 16-chip orthogonal Walsh codes (*W1–W4*). We combined the user chip streams, oversampled the combined signal by a factor of 3, and fed the resulting signal to the 48-tap filter. We provided *U1*'s bit stream as the target and let the filter learn the appropriate despreading code *W1*. Part B: The 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.

to 10nA RMS (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 result shows that, as predicted by [8], the error performance of the filter is not limited by the weight linearity of the multipliers.

For our second experiment, we enabled 24 taps and trained the filter to output a triangle wave given a square-wave input. Fig. 5(A) shows the target and output during the first 80 and the last 80 iterations. Fig. 5(B) shows the RMS error during adaptation. The filter performance is limited by the input quantization, with an RMS error of about 5μA (full output range is 24×128μA) after 480 iterations. As the filter continues to adapt, the error settles to 2μA RMS after an additional 300 iterations (corresponding to an output resolution of 10 bits). Fig. 5(C) illustrates an attractive benefit of adaptation: Two weight cells learning the same nominal value converge to different voltages because the LMS algorithm naturally compensates for the effects of process mismatch (offsets in the weight representation and variations in the multiplier gain).

As a final experiment, we enabled all 48 taps and used the adaptive filter in a direct-sequence code-division multiple-access (DS-CDMA) despreading application. Fig. 6(A) shows the experiment, where four users share a CDMA channel. We encoded each user's bit stream *Ui* with orthogonal 16-chip Walsh spreading codes, and added the chip streams to form a composite signal. We input this signal (oversampled by a factor of 3) to the 48-tap adaptive filter, and provided it with *U1*'s bit stream as a reference. The task of the filter is to learn the spreading code *W1* and produce *U1*'s original bit stream. An adaptive matched filter like this could be used in decision-feedback CDMA despreading with blind multiuser detection [10]. Fig. 6(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 bits stream from the filter's analog output. 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 matched filter's interference- and noise-rejection characteristics to a resolution of 10 bits.

7. Conclusion

We built a 48-tap, 19.2GOPS, 2.6mm^2 adaptive FIR filter using a digital delay line, nonvolatile analog weights, and mixed-signal multipliers. Analog storage and pulse-based feedback allow us to store and adapt the tap weights with an accuracy of 13 bits and achieve an output resolution of 10 bits with a power consumption of only 20mW. Also, nonvolatile analog weights enable us to stop adaptation and operate in open loop without the charge-leakage problems associated with capacitor-based analog storage. Because we use a digital delay line, we can readily scale the design to add more taps.

Future work includes reducing the area and increasing the resolution of the mixed-signal multipliers using the on-chip trimming techniques we demonstrated in [7]. Additionally, we will accelerate the convergence of the LMS algorithm by adding a decorrelating input stage as shown in [11].

References

[1] 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.

[2] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1985.

[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] M. Q. Le, P. J. Hurst, and J. P. Keane, "An Adaptive Analog Noise-Predictive Decision-Feedback Equalizer," in Symposium on VLSI Circuits, Honolulu, Hawaii, 2000.

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

[6] 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 0.25μm CMOS," in Symposium on VLSI Circuits, Honolulu, Hawaii, 2002.

[7] 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.

[8] C. Diorio, S. Mahajan, P. Hasler, B. A. Minch, and C. Mead, "A High-Resolution Nonvolatile Analog Memory Cell," in IEEE Intl. Symp. on Circuits and Systems, 1995.

[9] P. Hasler and J. Dugger, "Correlation Learning Rule in Floating-Gate pFET Synapses," *IEEE Transactions on Circuits and Systems II*, vol. 48, pp. 65-73, 2001.

[10] M. Honig, U. Madhow, and S. Verdú, "Blind Adaptive Multiuser Detection," *IEEE Transactions on Information Theory*, vol. 41, pp. 944-960, 1995.

[11] 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.