A PIPELINED ARCHITECTURE FOR LMS ADAPTIVE FIR FILTERS WITHOUT ADAPTATION DELAY

Q. Zhu, S.C. Douglas, and K.F. Smith

Department of Electrical Engineering
University of Utah
Salt Lake City, UT 84112 USA

ABSTRACT

Past methods for mapping the least-mean-square (LMS) adaptive finite-impulse-response (FIR) filter onto parallel and pipelined architectures either introduce delays in the coefficient updates or have excessive hardware requirements. In this paper, we describe a pipelined architecture for the LMS adaptive FIR filter that produces the same output and error signals as would be produced by the standard LMS adaptive filter architecture without adaptation delays. Unlike existing architectures for delayless LMS adaptation, the new architecture's throughput and hardware complexity are independent of and linear with the filter length, respectively.

1. INTRODUCTION

Advancements in VLSI technology have spurred efforts to map complex algorithms onto regular architectures with computations that can be parallelized and/or pipelined. Although it is relatively straightforward to synthesize high-speed architectures for feedforward-only signal processing structures such as finite-impulse-response (FIR) filters, it is considerably more difficult to synthesize similar architectures for feedback structures [1, 2]. A case in point is the least-mean-square (LMS) adaptive FIR filter [3], in which the error of the adaptive filter is used to adjust the filter coefficients in real time. The need for a high-speed architecture for this ubiquitous signal processing system has long been recognized, as it is used in numerous high-data-rate systems in communications [4, 5, 6].

Without exception, previously-proposed methods to parallelize and/or pipeline the LMS adaptive FIR filter introduce either delays in the coefficient updates [2, 7, 8, 9, 10, 11] or a large hardware overhead [1]. In [7, 8, 9, 10], each coefficient of the system receives an equal delay in the coefficient update, resulting in the delayed LMS adaptive filter. The delayed LMS algorithm is generalized in [2] to allow different coefficient delays for both the filter output computation and the calculation of the coefficient updates. Alternate algorithms employing a transposed form of the FIR filter structure are proposed and described in [11], in which the update delay is different for each LMS adaptive filter coefficient. Analyses of these delayed adaptation algorithms indicate that the performance of these systems is inferior to that of the LMS adaptive filter in some cases, and the performance loss is particularly severe as the amount of delay is increased [2, 7, 8, 9, 11]. This loss in performance is problematic for adaptive FIR filters with hundreds of coefficients, as the update delay usually increases in proportion to the number of filter coefficients.

A second difficulty in employing LMS adaptive filters with delayed updates is in choosing the step size to obtain fast and accurate adaptation behavior. The best step size choice for these algorithms is a complicated function of the input statistics and the delays within the adaptation loop [8, 9]. Step size normalization for the delayed LMS adaptive filter has been proposed [12]; however, the performance of the resulting system still depends on the input signal correlation statistics and the adaptation delays. Clearly, it would be desirable to obtain a low-complexity pipelined architecture whose coefficient updates contain no adaptation delay, so that normalized LMS adaptation or other variable step size strategies can be reliably employed.

Recently, a method has been introduced for correcting the error produced by the delayed LMS adaptive filter so that standard LMS or normalized LMS adaptation can be employed [12, 13]. In this method, a correction term is computed using products of past corrected errors and certain input signal correlation terms that can be computed recursively. Extensions of the method to allow delayless filtered-X LMS adaptation for adaptive control applications are presented in [14]. It is not clear, however, how the computation of the correction term in these methods can be paralleled and/or pipelined. No high-speed architecture employing these methods has been presented.

In this paper, we describe a pipelined implementation of the LMS adaptive FIR filter. The implementation applies the correction term method as described in [13] to a transpose-form FIR filter structure, where the correction term is also computed using a transpose-form structure. The system can be implemented using regularly-connected processing modules, making it amenable to VLSI implementation, and the overall complexity is linear in the number of filter coefficients. The new architecture's throughput is always greater than that of existing delayless LMS adaptive filter architectures. Moreover, its throughput does not depend on the filter length to first order. Simulations verify the computational equivalence of the new architecture to that of the standard LMS adaptive filter.

2. THE ARCHITECTURE

The LMS adaptive filter coefficient updates can be described in vector form as

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu e(n)\mathbf{X}(n) \qquad (1)$$
$$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{X}(n), \qquad (2)$$

[0]Portions of this research have been supported by the Federal Bureau of Investigation under Contract No. JFBI95017 and by the National Science Foundation under Grant No. MIP-9501680.

$$r_{L-i+1}(n+L-i+1) = r_{L-i+1}(n+L-i) + x(n+L-i+1)x(n) - x(n-i+1)x(n-L) \quad (17)$$

$$\begin{aligned} y_i(n-L) &= y_{i-1}(n-L-1) + w_{L-i}(n-L-i)x(n-L) + r_{L-i+1}(n-i)e_\mu(n-L-1) && (18)\\ r_{L-i+1}(n-i+1) &= r_{L-i+1}(n-i) + x(n-i+1)x(n-L) - x(n-L-i+1)x(n-2L) && (19)\\ w_{L-i}(n-L-i+1) &= w_{L-i}(n-L-i) + e_\mu(n-L-i)x(n-2L). && (20)\end{aligned}$$

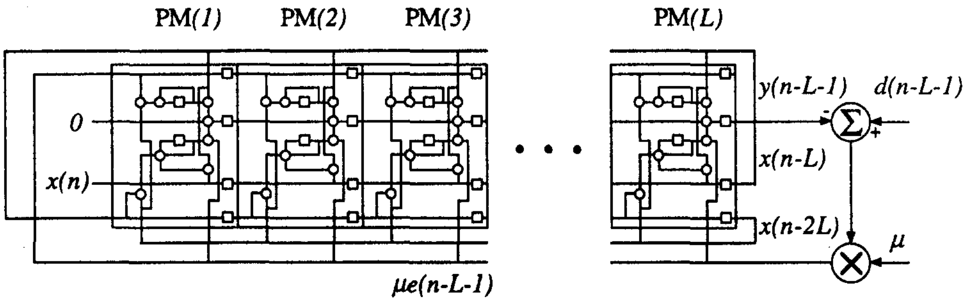

Fig. 3: The pipelined LMS adaptive filter architecture.

Table 1: Transpose-form algorithm for LMS adaptation.

Equation	×'s
$\mathbf{Y}(n-L) = \begin{bmatrix} 0 \\ \overline{\mathbf{Y}}(n-L-1) \end{bmatrix} + x(n-L)\widetilde{\mathbf{W}}(n-L)$ $+e_\mu(n-L-1)\widetilde{\mathbf{R}}(n)$	$2L$
$\widetilde{\mathbf{W}}(n-L+1) = \widetilde{\mathbf{W}}(n-L) + x(n-2L)\mathbf{E}_\mu(n-L-1)$	L
$e_\mu(n-L) = \mu(n-L)(d(n-L) - y_L(n-L))$	1
$\mathbf{E}_\mu(n-L) = \begin{bmatrix} e_\mu(n-L) \\ \overline{\mathbf{E}}_\mu(n-L-1) \end{bmatrix}$	0
$\widetilde{\mathbf{R}}(n+1) = \widetilde{\mathbf{R}}(n) + x(n-L)\mathbf{X}(n) - x(n-2L)\mathbf{X}(n-L)$	$2L$
Total ($\mu(n)$ arbitrary): $5L+1$	

Comparing (16) and (14), we see that the two equations have the same form. Therefore, we can use a transpose-form structure to compute the correction term in (16). Moreover, since $c(n)$ is added to the output of the delayed coefficient filter, we can use a single transpose-form structure to compute both the delayed coefficient filter output and the correction term. Since the correlation terms $r_i(n)$ are delayed within the transpose-form structure, the ith processing module implements the update relation given by (17) at the top of this page, where the time indices for this equation are analogous to those in (15).

To obtain a causal implementation given the computation of the correlation terms in (17), we delay the input and desired response signals by L time samples. Figures 3 and 4 show the set of interconnected modules and the ith module, respectively, for the pipelined LMS adaptive filter. The set of equations implemented in the ith module are shown in (18)–(20) at the top of this page. The value of $y_L(n-L)$ is equal to $y(n-L) = \mathbf{W}^T(n-L)\mathbf{X}(n-L)$, where $\mathbf{W}(n-L)$ evolves according to the LMS algorithm in (1). Thus, although the system adapts the coefficients using the LMS algorithm, the system output is only available after an L-sample delay. Table 1 depicts the algorithm in vector form, where we have defined $\widetilde{\mathbf{W}}(n) = [w_{L-1}(n-1)\ w_{L-2}(n-2) \cdots w_0(n-L)]^T$, $\widetilde{\mathbf{R}}(n) = [r_L(n-1)\ r_{L-1}(n-2) \cdots r_1(n-L)]^T$, and $\mathbf{Y}(n) = [y_1(n)\ y_2(n) \cdots y_L(n)]^T$, respectively, and where the vectors $\overline{\mathbf{Y}}(n)$ and $\overline{\mathbf{E}}_\mu(n)$ contain the first $L-1$ elements of $\mathbf{Y}(n)$ and $\mathbf{E}_\mu(n)$, respectively. In the table, we have listed the algorithm with a variable step size $\mu(n)$, as this modification does not alter the fundamental form of the algorithm.

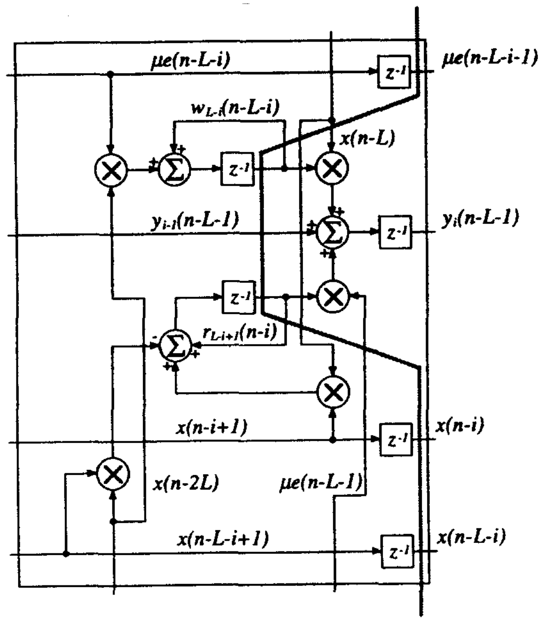

Fig. 4: A single processing module for a pipelined LMS adaptive filter architecture.

3. IMPLEMENTATION ISSUES

We now compare the throughput of the pipelined LMS architecture with more conventional architectures for delayless LMS adaptive filtering. It is well known that the limiting factor in the throughput of the LMS adaptive filter is the number of additions necessary to form the error signal [11]. Assuming that all of the multiplies in the conventional LMS adaptive filter are computed in parallel and that the step size μ is a power-of-two value such that the product $\mu e(n)$ can be computed via simple bit shifts, the throughput of the conventional architecture is given by

$$f_{conv} = \frac{1}{2T_{mult} + (L+1)T_{add}}, \quad (21)$$

where T_{mult} and T_{add} are the times necessary to compute a single multiply and add, respectively. This throughput can be increased if a binary adder architecture is used, in which

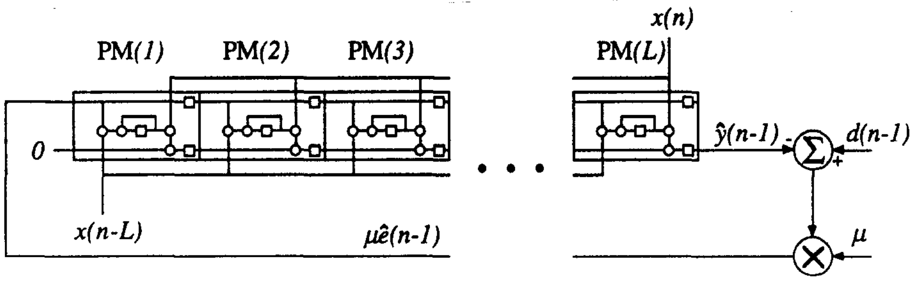

Fig. 1: A pipelined delayed LMS adaptive filter architecture.

where the vector $\mathbf{W}(n) = [w_0(n) \cdots w_{L-1}(n)]^T$ contains the L adaptive filter coefficients at time n, $\mathbf{X}(n) = [x(n) \cdots x(n-L+1)]^T$ contains the L input signal samples in filter memory at time n, $d(n)$ and $e(n)$ are the desired response and error signals at time n, respectively, and μ is the step size. In contrast, the delayed LMS adaptive filter is defined as

$$\widehat{\mathbf{W}}(n+1) = \widehat{\mathbf{W}}(n) + \mu\widehat{e}(n)\mathbf{X}(n) \quad (3)$$

$$\widehat{e}(n) = d(n) - \widehat{\mathbf{W}}^T(n-D)\mathbf{X}(n), \quad (4)$$

where $\widehat{\mathbf{W}}(n) = [\widehat{w}_0(n) \cdots \widehat{w}_{L-1}(n)]^T$ and D is an integer value of delay.

Define the error signal $e_m(n)$ as

$$e_m(n) = d(n) - \mathbf{W}^T(m)\mathbf{X}(n). \quad (5)$$

Then, it is shown in [13] that the error signal $e(n) = e_n(n)$ can be computed from the error signal with delayed coefficient values $e_{n-D}(n)$ via a correction term $c(n)$ as

$$e_n(n) = e_{n-D}(n) - c(n) \quad (6)$$

$$c(n) = \sum_{i=1}^{D} r_i(n)(\mu e_{n-i}(n-i)) \quad (7)$$

$$= \mathbf{R}^T(n)\mathbf{E}_\mu(n-1), \quad (8)$$

where $\mathbf{R}(n) = [r_1(n)\ r_2(n) \cdots r_D(n)]^T$ is a D-dimensional vector of input signal correlations given by

$$r_j(n) = \sum_{m=0}^{L-1} x(n-m)x(n-m-j), \quad (9)$$

and $\mathbf{E}_\mu(n) = [\mu e_n(n)\ \mu e_{n-1}(n-1) \cdots \mu e_{n-D+1}(n-D+1)]^T$ is a vector of past errors. The correlation terms $r_i(n)$ can be computed recursively as

$$r_j(n) = r_j(n-1) + x(n)x(n-j) - x(n-L)x(n-L-j). \quad (10)$$

Thus, the error signal can be computed using delayed coefficient values, and the delayless error signal $e_n(n)$ can be used to update the filter coefficients. However, since the update in (10) is marginally stable, errors in $r_j(n)$ due to finite-precision calculations will grow linearly over time. In such cases, the value of $r_j(n)$ must be periodically re-initialized to its correct value, or the approximate leaky integrator update given by

$$r_j(n) = \lambda r_j(n-1) + x(n)x(n-j) - \lambda^L x(n-L)x(n-L-j), \quad (11)$$

can be used in place of (10), where $0 \ll \lambda < 1$.

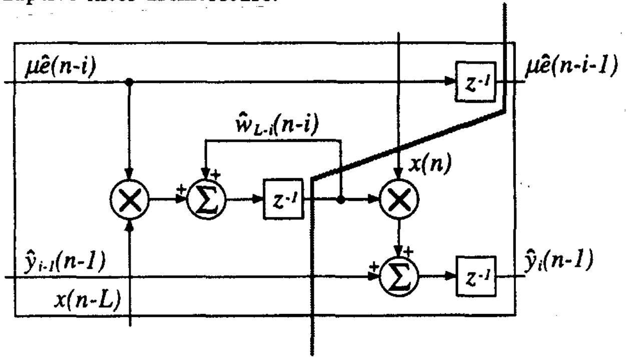

Fig. 2: A single processing module for a pipelined delayed LMS adaptive filter architecture.

To obtain a pipelined implementation of the LMS adaptive filter, we first consider a transposed-form version of the delayed LMS adaptive filter. Figure 1 shows the complete structure containing L processing modules, and Figure 2 shows the detail of the ith processing module. Also shown in Figure 2 are the data wavefronts in the pipelined array, as indicated by the grey cut lines. In this case, the intermediate outputs $\widehat{y}_i(n)$ for $1 \le i \le L$ are defined as

$$\widehat{y}_i(n) = \sum_{j=1}^{i} \widehat{w}_{L-j}(n-i)x(n-i+j), \quad (12)$$

and they can be computed using the relation

$$\widehat{y}_i(n) = \widehat{y}_{i-1}(n-1) + \widehat{w}_{L-i}(n-i)x(n) \quad (13)$$

with $y_0(n-1) = 0$. The output of the system is

$$\widehat{y}(n) = \widehat{y}_L(n) = \sum_{i=0}^{L-1} \widehat{w}_i(n-L)x(n-i). \quad (14)$$

From (3), the coefficients of the filter at the ith stage can be updated as

$$\widehat{w}_{L-i}(n-i+1) = \widehat{w}_{L-i}(n-i) + \mu\widehat{e}(n-i)x(n-L). \quad (15)$$

The transpose structure delays these intermediate coefficient values such that the filter output is given by (14). The samples $x(n)$ and $x(n-L)$ are passed to every processing module; thus, an additional L-length shift register is needed to store the input signal values for the system.

To develop the pipelined architecture for delayless LMS adaptation, we consider the correction term $c(n)$ defined in (7) for the case $D = L$. We note that we can write this correction term as

$$c(n) = \sum_{i=1}^{L} r_i((n+L)-L)(\mu e(n-i)). \quad (16)$$

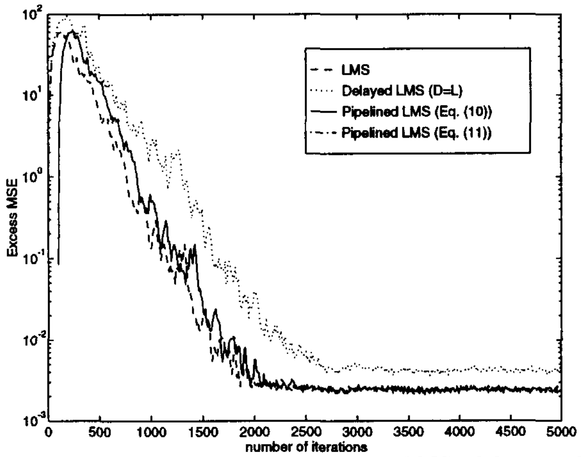

Fig. 5: Convergence of excess MSE: LMS, delayed LMS ($D = L$), and pipelined LMS adaptive filters, $\mu = 2^{-8}$, $L = 100$.

case the throughput is

$$f_{bin} = \frac{1}{2T_{mult} + \lceil \log_2 L + 1 \rceil T_{add}}, \qquad (22)$$

where $\lceil \log_2 L + 1 \rceil$ denotes the next largest integer value of $\log_2 L + 1$. The structure of this binary adder is not regular, however, and is difficult to implement in VLSI.

For the pipelined architecture, we assume an adaptive noise canceller configuration for which only the error signal $e(n)$ is of interest. If we set $y_0(n-1) = -d(n-1)$, the structure produces the error signal $y_L(n-L-1) = -e(n-L-1)$ directly. From Figure 3, it is seen that the elemental computation within the structure is a single multiply and two adds if the bit shift of $e(n)$ by μ is ignored, so that the throughput is given by

$$f_{pipe} = \frac{1}{T_{mult} + 2T_{add}}. \qquad (23)$$

Unlike the other structures, the new architecture has a throughput that is independent of the filter length, implying that LMS adaptive FIR filters with hundreds of filter coefficients can be implemented without a significant decrease in the sample rate of the system.

Assuming that $T_{mult} = 2T_{add}$ for a pipelined multiplier, the speedups of the new architecture over the conventional and binary tree adder LMS architectures are

$$S_{pipe/conv} = \frac{5+L}{4} \qquad (24)$$

$$S_{pipe/bin} = \frac{5 + \lceil \log_2 L \rceil}{4}, \qquad (25)$$

respectively. Thus, for all filter lengths, the new architecture is faster than the existing architectures. For $L = 128$, the new architecture is 3 and 33.2 times as fast as the binary tree adder and conventional architectures, respectively.

To verify the operation of the proposed architecture, Figure 5 shows the convergence of the excess mean-squared error (MSE), defined as $E\{(e(n) - \eta(n))^2\}$ for the LMS, delayed LMS ($D = L$), and pipelined LMS adaptive filters where $r_j(n)$ is as computed in (10) and in (11), respectively, for a $L = 100$-coefficient system identification task, where $\eta(n)$ is the observation noise corrupting the desired response signal and $\lambda = 1 - 2^{-12}$. In these simulations, the input and observation noise signals were zero-mean joinly Gaussian random processes with variances of 1 and 0.01, respectively, and the unknown system coefficients were all set to unity. In this case, we have chosen $\mu = 2^{-8} \approx 0.0039$ for all three algorithms. The LMS and pipelined LMS architectures perform similarly for these signals, whereas the delayed LMS adaptive filter suffers from slower convergence and a greater level of error in steady-state for this step size value. Note that the pipelined LMS algorithm's output is an exact delayed version of the LMS algorithm's output, up to finite precision errors in the two systems. Thus, the L-sample latency in the pipelined LMS algorithm's output has no effect on the overall adaptation behavior of the system. In addition the errors introduced via the leaky update of $r_j(n)$ in (11) are negligible.

4. CONCLUSION

In this paper, we have described a pipelined implementation of the LMS adaptive filter. Unlike existing architectures, the proposed architecture computes the LMS adaptive filter output exactly and therefore does not suffer from poor performance due to adaptation delays. However, the output of the filter is delayed by L samples, where L is the filter length. The new architecture is regular, thus making it amenable to VLSI implementation. Comparisons with existing delayless LMS adaptive filter architectures show that the new architecture achieves a higher throughput than existing architectures for all FIR filter lengths. Simulations show the equivalence of the new architecture's adaptation behavior and that of the conventional LMS adaptive filter.

REFERENCES

[1] T.H.-Y. Meng and D.G. Messerschmitt, "Arbitrarily high sampling rate adaptive filters," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 35, no. 4, pp. 455-470, Apr. 1987.

[2] N.R. Shanbhag and K.K. Parhi, *Pipelined Adaptive Digital Filters* (Boston, MA: Kluwer, 1994).

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

[4] V.B. Lawrence and S.K. Tewksbury, "Multiprocessor implementation of adaptive digital filters," *IEEE Trans. Comm.,* vol. 31, no. 6, pp. 826-835, June 1983.

[5] T.K. Miller, S.T. Alexander, and L.J. Faber, "An SIMD multiprocessor ring architecture for the LMS adaptive algorithm," *IEEE Trans. Comm.,* vol. 34, no. 1, pp. 89-92, Jan. 1986.

[6] F. Lu and H. Samueli, "A 60-MBaud, 480-Mb/s, 256-QAM decision-feedback equalizer in 1.2μ CMOS," *IEEE J. Solid State Circ.,* vol. 28, no. 3, pp. 330-338, May 1993.

[7] P. Kabal, "The stability of adaptive minimum mean square error equalizers using delayed adjustment," *IEEE Trans. Comm.,* vol. 31, no. 2, pp. 430-432, Mar. 1983.

[8] G. Long, F. Ling, and J.G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust., Speech, Signal Processing,* vol. 37, no. 9, pp. 1397-1405, Sept. 1989; vol. 40, no. 1, pp. 230-232, Jan. 1992 (corrections).

[9] H. Herzberg, R. Haimi-Cohen, and Y. Be'ery, "A systolic array realization of an LMS adaptive filter and the effects of delayed adaptation," *IEEE Trans. Signal Processing,* vol. 40, no. 11, pp. 2799-2803, Nov. 1992.

[10] M.D. Meyer and D.P. Agrawal, "A high sampling rate delayed LMS filter architecture," *IEEE Trans. Circ. Syst. II: Anal. Dig. Sig. Proc.,* vol. 40, no. 11, pp. 727-729, Nov. 1993.

[11] D.L. Jones, "Learning characteristics of transpose-form LMS adaptive filters," *IEEE Trans. Circ. Syst. II: Anal. Dig. Sig. Proc.,* vol. 39, no. 10, pp. 745-749, Oct. 1992.

[12] M. Rupp and R. Frenzel, "The behavior of LMS and NLMS algorithms with delayed coefficient update in the presence of spherically invariant processes," *IEEE Trans. Signal Processing,* vol. 42, no. 3, pp. 668-672, Mar. 1994.

[13] R.D. Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm," *IEEE Signal Processing Lett.,* vol. 2, no. 12, p. 223, Dec. 1995.

[14] S.C. Douglas, "Fast exact filtered-X LMS and LMS algorithms for multichannel active noise control," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing,* Munich, Germany, Apr. 1996.