Critical-Path Analysis and Low-Complexity Implementation of the LMS Adaptive Algorithm

Pramod Kumar Meher, *Senior Member, IEEE*, and Sang Yoon Park, *Member, IEEE*

***Abstract*—This paper presents a precise analysis of the critical path of the least-mean-square (LMS) adaptive filter for deriving its architectures for high-speed and low-complexity implementation. It is shown that the direct-form LMS adaptive filter has nearly the same critical path as its transpose-form counterpart, but provides much faster convergence and lower register complexity. From the critical-path evaluation, it is further shown that no pipelining is required for implementing a direct-form LMS adaptive filter for most practical cases, and can be realized with a very small adaptation delay in cases where a very high sampling rate is required. Based on these findings, this paper proposes three structures of the LMS adaptive filter: (i) Design 1 having no adaptation delays, (ii) Design 2 with only one adaptation delay, and (iii) Design 3 with two adaptation delays. Design 1 involves the minimum area and the minimum energy per sample (EPS). The best of existing direct-form structures requires 80.4% more area and 41.9% more EPS compared to Design 1. Designs 2 and 3 involve slightly more EPS than the Design 1 but offer nearly twice and thrice the MUF at a cost of 55.0% and 60.6% more area, respectively.**

***Index Terms*—Adaptive filters, critical-path optimization, least mean square algorithms, LMS adaptive filter.**

I. Introduction

ADAPTIVE digital filters find wide application in several digital signal processing (DSP) areas, e.g., noise and echo cancellation, system identification, channel estimation, channel equalization, etc. The tapped-delay-line finite-impulse-response (FIR) filter whose weights are updated by the famous Widrow-Hoff least-mean-square (LMS) algorithm [1] may be considered as the simplest known adaptive filter. The LMS adaptive filter is popular not only due to its low-complexity, but also due to its stability and satisfactory convergence performance [2]. Due to its several important applications of current relevance and increasing constraints on area, time, and power complexity, efficient implementation of the LMS adaptive filter is still quite important.

To implement the LMS algorithm, one has to update the filter weights during each sampling period using the estimated error, which equals the difference between the current filter output and the desired response. The weights of the LMS adaptive filter

Manuscript received February 21, 2013; revised June 25, 2013; accepted July 18, 2013. Date of publication October 21, 2013; date of current version February 21, 2014. This paper was recommended by Associate Editor A. Ashrafi.

P. K. Meher is with the School of Computer Engineering, Nanyang Technological University, Singapore (e-mail: aspkmeher@ntu.edu.sg; URL: http://www3.ntu.edu.sg/home/aspkmeher).

S. Y. Park is with the Institute for Infocomm Research, Singapore, 138632 (e-mail: sypark@i2r.a-star.edu.sg).

Corresponding author: S. Y. Park.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2013.2284173

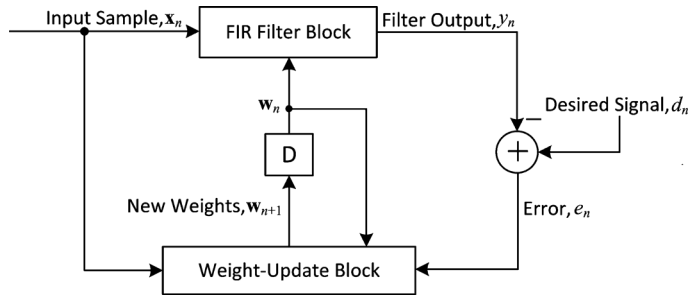


Fig. 1. Structure of conventional LMS adaptive filter.

during the nth iteration are updated according to the following equations.

$$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e_n \mathbf{x}_n \tag{1a}$$

where

$$e_n = d_n - y_n \tag{1b}$$

$$y_n = \mathbf{w}_n^T \mathbf{x}_n \tag{1c}$$

with input vector $\mathbf{x}_n$ and weight vector $\mathbf{w}_n$ at the nth iteration are given by, respectively,

$$\mathbf{x}_n = [x_n, x_{n-1}, \ldots, x_{n-N+1}]^T$$
$$\mathbf{w}_n = [w_n(0), w_n(1), \ldots, w_n(N-1)]^T$$

and where d_n is the desired response, y_n is the filter output of the nth iteration, e_n denotes the error computed during the nth iteration, which is used to update the weights, μ is the convergence factor or step-size, which is usually assumed to be a positive number, and N is the number of weights used in the LMS adaptive filter. The structure of a conventional LMS adaptive filter is shown in Fig. 1.

Since all weights are updated concurrently in every cycle to compute the output according to (1), direct-form realization of the FIR filter is a natural candidate for implementation. However, the direct-form LMS adaptive filter is often believed to have a long critical path due to an inner product computation to obtain the filter output. This is mainly based on the assumption that an arithmetic operation starts only after the complete input operand words are available/generated. For example, in the existing literature on implementation of LMS adaptive filters, it is assumed that the addition in a multiply-add operation (shown in Fig. 2) can proceed only after completion of the multiplication, and with this assumption, the critical path of the multiply-add operation ($T_{\rm MA}$) becomes $T_{\rm MULT} + T_{\rm ADD}$, where $T_{\rm MULT}$ and $T_{\rm ADD}$ are the time required for a multiplication and an addition, respectively. Under such assumption, the critical path of the direct-form LMS adaptive filer (without pipelining) can be

1549-8328 © 2013 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

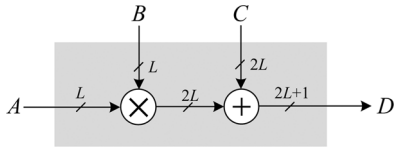


Fig. 2. Example of multiply-add operation for the study of delay in composite operations.

estimated as $T = 2T_{\rm MULT} + (N+1)T_{\rm ADD}$. Since this critical-path estimate is quite high, it could exceed the sample period required in many practical situations, and calls for a reduction of critical-path delay by pipelined implementation. But, the conventional LMS algorithm does not support pipelined implementation. Therefore, it is modified to a form called the delayed LMS (DLMS) algorithm [3], [4], which allows pipelined implementation of different sections of the adaptive filter. Note that the transpose-form FIR LMS adaptive filter is inherently of a delayed LMS kind, where the adaptation delay varies across the sequence of filter weights. Several works have been reported in the literature over the last twenty years [5]–[11] for efficient implementation of the DLMS algorithm.

Van and Feng [5] have proposed an interesting systolic architecture, where they have used relatively large processing elements (PEs) for achieving lower adaptation delay compared to other DLMS systolic structures with critical path of one MAC operation. Yi *et al.* [10] have proposed a fine-grained pipelined design of an adaptive filter based on direct-form FIR filtering, using a fully pipelined binary adder-tree implementation of all the multiplications in the error-computation path and weight-update path to limit the critical path to a maximum of one addition time. This architecture supports high sampling frequency, but involves large pipeline depth, which has two adverse effects. First, the register complexity, and hence the power dissipation, increases. Secondly, the adaptation delay increases and convergence performance degrades. However, in the following discussion, we establish that such aggressive pipelining is often uncalled for, since the assumption that the arithmetic operations start only after generation of their complete input operand words is not valid for the implementation of composite functions in dedicated hardware. Such an assumption could be valid when multipliers and adders are used as discrete components, which is not the case in ASIC and FPGA implementation these days. On the other hand, we can assume that an arithmetic operation can start as soon as the LSBs of the operands are available. Accordingly, the propagation delay for the multiply-add operation in Fig. 2 could be taken to be $T_{\rm MA} = T_{\rm MULT} + T_{\rm FAC} + T_{\rm FAS}$, where $T_{\rm FAC}$ and $T_{\rm FAS}$ are the delays of carry and sum generation in a 1-bit full-adder circuit. Therefore, $T_{\rm MA}$ is much less than $T_{\rm MULT} + T_{\rm ADD}$. In Table I, we have shown the propagation delays of a multiplier, an adder, and carry-and-sum generation in a 1-bit full-adder circuit, and multiply-add circuit in TSMC 90-nm [12] and 0.13-μm [13] processes to validate our assertion in this context. From this table, we can also find that $T_{\rm MA}$ is much less than $T_{\rm MULT} + T_{\rm ADD}$. In Section III, we further show that the critical path of the direct-form LMS adaptive filter is much less than $2T_{\rm MULT} + (N+1)T_{\rm ADD}$, and would amount to nearly $2T_{\rm MULT} + (\lceil \log_2 N \rceil + 2)\Delta$, where $\Delta = T_{\rm FAC} + T_{\rm FAS}$. Besides, we have shown that no pipelining is required for implementing the LMS algorithm for most practical cases, and could be realized with very small adaption delay of one or two samples in cases like radar applications where very high sampling rate is required [10]. The highest sampling rate, which could be as high as 30.72 Msps, supported by the fastest wireless communication standard (long-term evolution) LTE-Advanced [14]. Moreover, computation of the filter output and weight update could be multiplexed to share hardware resources in the adaptive filter structure to reduce the area consumption.

TABLE I
PROPAGATION DELAY (NS) BASED ON SYNTHESIS OF TSMC 0.13-μM AND 90-NM CMOS TECHNOLOGY LIBRARIES

Component	0.13-μm		90-nm	
	$L=8$	$L=16$	$L=8$	$L=16$
$T_{\rm FAC}$	0.21$\sim$ 0.23		0.10	
$T_{\rm FAS}$	0.14$\sim$ 0.18		0.1$\sim$ 0.16	
$T_{\rm ADD}$	1.08	2.03	0.74	1.42
$T_{\rm MULT}$	2.43	4.51	1.54	3.00
$T_{\rm MA}$	2.79	4.90	1.83	3.29

$T_{\rm MULT}$: computation time of delay-optimized flexible Booth Wallace multiplier. $T_{\rm FAC}$: delay to produce the carry in a 1-bit full-adder circuit. $T_{\rm FAS}$: propagation delay for generation of sum in a 1-bit full adder, which is the same as that of 3-input XOR gate. $T_{\rm MA}$: computation time of multiplier-add circuit. Propagation delay is measured in ns and can change with temperature and output load capacitance.

Further effort has been made by Meher and Maheswari [15] to reduce the number of adaptation delays as well as the critical path by an optimized implementation of the inner product using a unified pipelined carry-save chain in the forward path. Meher and Park [8], [9] have proposed a 2-bit multiplication cell, and used that with an efficient adder tree for the implementation of pipelined inner-product computation to minimize the critical path and silicon area without increasing the number of adaptation delays. But, in these works, the critical-path analysis and necessary design considerations are not taken into account. Due to that, the designs of [8], [9], [15] still consume higher area, which could be substantially reduced. Keeping the above observations in mind, we present a systematic critical-path analysis of the LMS adaptive filter, and based on that, we derive an architecture for the LMS adaptive filter with minimal use of pipeline stages, which will result in lower area complexity and less power consumption without compromising the desired processing throughput.

The rest of the paper is organized as follows. In the next section, we review the direct-form and transpose-form implementations of the DLMS algorithm, along-with their convergence behavior. The critical-path analysis of both these implementations is discussed in Section III. The proposed low-complexity designs of the LMS adaptive filter are described in Section IV. The performance of the proposed designs in terms of hardware requirement, timings, and power consumption is discussed in Section V. Conclusions are presented in Section VI.

II. REVIEW OF DELAYED LMS ALGORITHM AND ITS IMPLEMENTATION

In this section, we discuss the implementation and convergence performance of direct-form and transpose-form DLMS adaptive filters.

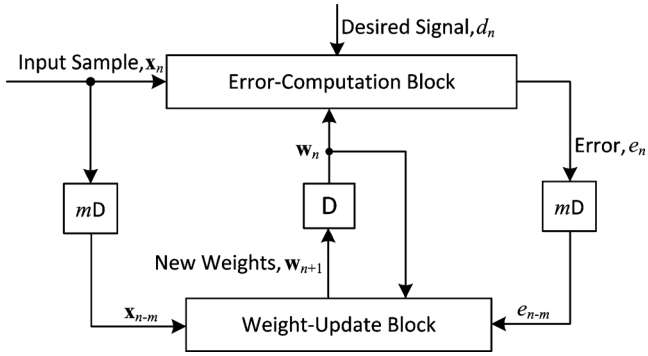


Fig. 3. Generalized block diagram of direct-form DLMS adaptive filter.

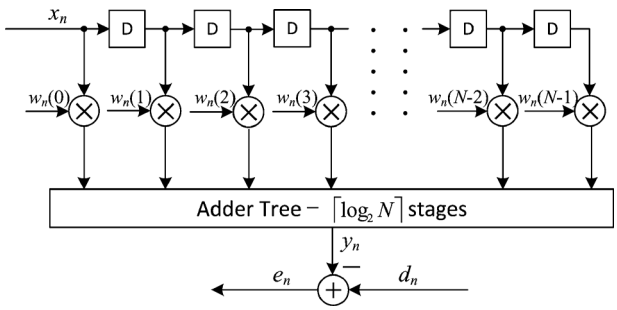


Fig. 4. Error-computation block of Fig. 3.

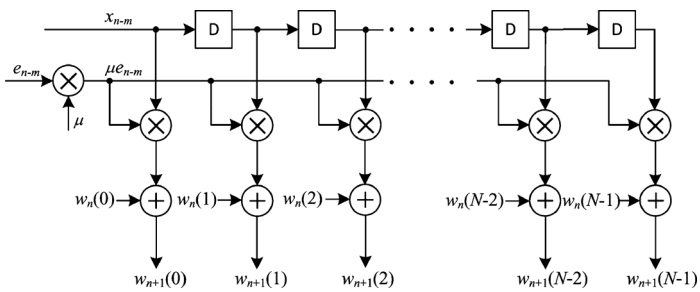


Fig. 5. Weight-update block of Fig. 3.

A. Implementation of Direct-Form Delayed LMS Algorithm

Assuming that the error-computation path is implemented in m pipelined stages, the latency of error computation is m cycles, so that the error computed by the structure at the nth cycle is e_{n-m}, and is used with the input samples delayed by m cycles to generate the weight-increment term. The weight-update equation of the DLMS algorithm is given by

$$l\mathbf{w}_{n+1} = \mathbf{w}_n + \mu e_{n-m}\mathbf{x}_{n-m} \tag{2a}$$

where

$$e_{n-m} = d_{n-m} - y_{n-m} \tag{2b}$$

and

$$y_n = \mathbf{w}_n^T \mathbf{x}_n. \tag{2c}$$

A generalized block diagram of direct-form DLMS adaptive filter is shown in Fig. 3. It consists of an error-computation block (shown in Fig. 4) and a weight-update block (shown in Fig. 5). The number of delays m shown in Fig. 3 corresponds to the pipeline delays introduced due to pipelining of the error-computation block.

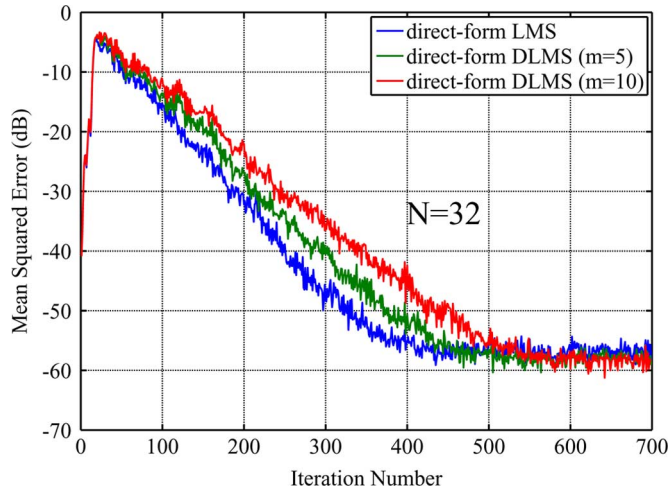


Fig. 6. Convergence of direct-form delayed LMS adaptive filter.

Direct-form adaptive filters with different values of adaptation delay are simulated for a system identification problem, where the system is defined by a bandpass filter with impulse response given by

$$h_n = \frac{\sin[w_{\mathrm{H}}(n-7.5)]}{\pi(n-7.5)} - \frac{\sin[w_{\mathrm{L}}(n-7.5)]}{\pi(n-7.5)} \tag{3}$$

for $n = 0, 1, \ldots, 15$, and $h_n = 0$ otherwise. Parameters w_{H} and w_{L} represent the high and low cutoff frequencies of the passband, and are set to $w_{\mathrm{H}} = 0.7\pi$ and $w_{\mathrm{L}} = 0.3\pi$, respectively. Fig. 6 shows the learning curves for identification of a 32-tap filter with Gaussian random input x_n of zero mean and unit variance, obtained by averaging 50 runs for $m = 0, 5$, and 10. The step-size μ is set to 1/40, 1/50, and 1/60 for $m = 0,\ 5,$ and 10, respectively, so that they provide the fastest convergence. In all cases, the output of the known system is of unity power, and contaminated with white Gaussian noise of -60 dB strength. It can be seen that as the number of delays increases, the convergence is slowed down, although the steady-state mean-square-error (MSE) remains almost the same in all cases.

B. Implementation of Transpose-Form Delayed LMS Algorithm

The transpose-form FIR structure cannot be used to implement the LMS algorithm given by (1), since the filter output at any instant of time has contributions from filter weights updated at different iterations, where the adaptation delay of the weights could vary from 1 to $(N-1)$. It could, however, be implemented by a different set of equations as follows:

$$y_n = \sum_{k=0}^{N-1} x_{n-k} w_{n-k}(k) \tag{4a}$$

$$w_{n+1}(k) = w_n(k) + \mu e_n x_{n-k} \tag{4b}$$

where $e_n = y_n - d_n$, and the symbols have the same meaning as those described in (1). In (4), it is assumed that no additional delays are incorporated to reduce the critical path during computation of filter output and weight update. If m additional delays are introduced in the error computation at any instant, then the

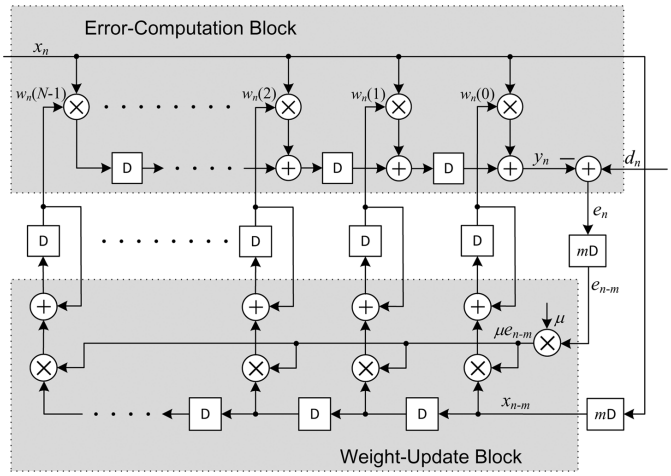


Fig. 7. Structure of transpose-form DLMS adaptive filter. The additional adaptation delay m could be at most 2 if no more delays are incorporated within the multiplication unit or between the multipliers and adders. If one delay could be placed after the computation of y_n and another after the computation of e_n, then $m = 2$.

weights are required to be updated according to the following equation

$$w_{n+1}(k) = w_n(k) + \mu e_{n-m} x_{n-m-k}, \tag{5}$$

but, the equation to compute the filter output remains the same as that of (4a). The structure of the transpose-form DLMS adaptive filter is shown in Fig. 7.

It is noted that in (4a), the weight values used to compute the filter output y_n at the nth cycle are updated at different cycles, such that the $(k+1)$th weight value $w_{n-k}(k)$ is updated k cycles back, where $k = 0, 1, \ldots N-1$. The transpose-form LMS is, therefore, inherently a delayed LMS and consequently provides slower convergence performance. To compare the convergence performance of LMS adaptive filters of different configurations, we have simulated the direct-form LMS, direct-form DLMS, and transpose-form LMS for the same system identification problem, where the system is defined by (3) using the same simulation configuration. The learning curves thus obtained for filter length $N = 16, 32$, and 48 are shown in Fig. 8. We find that the direct from LMS adaptive filter provides much faster convergence than the transpose LMS adaptive filter in all cases. The direct-form DLMS adaptive filter with delay 5 also provides faster convergence compared to the transpose-form LMS adaptive filter without any delay. However, the residual mean-square error is found to be nearly the same in all cases.

From Fig. 7, it can be further observed that the transpose-form LMS involves significantly higher register complexity over the direct-form implementation, since it requires an additional signal-path delay line for weight updating, and the registers on the adder-line to compute the filter output are at least twice the size of the delay line of the direct-form LMS adaptive filter.

III. Critical-Path Analysis of LMS Adaptive Filter and Implementation Strategy

The critical path of the LMS adaptive filter of Fig. 1 for direct implementation is given by

$$T = C_{\text{ERROR}} + C_{\text{UPDATE}} \tag{6}$$

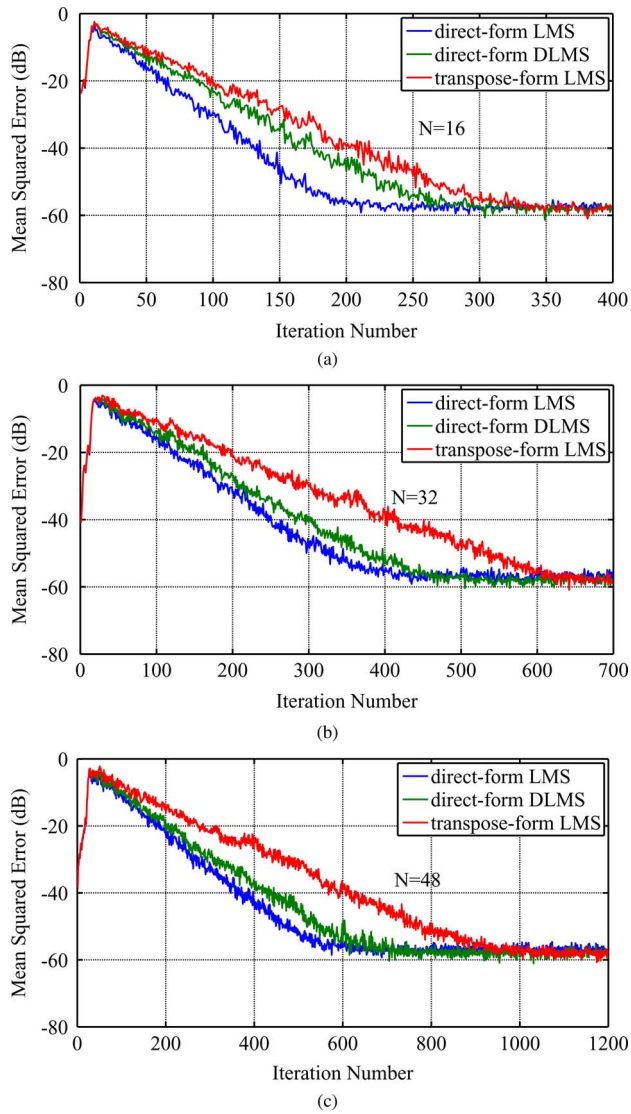


Fig. 8. Convergence comparison of direct-form and transpose-form adaptive filters. (a) $N = 16$. (b) $N = 32$. (c) $N = 48$. Adaptation delay is set to 5 for the direct-form DLMS adaptive filter.

where C_{ERROR} and C_{UPDATE} are, respectively, the time involved in error computation and weight updating. When the error computation and weight updating are performed in two separate pipeline stages, the critical path becomes

$$T = \max\{C_{\text{ERROR}}, C_{\text{UPDATE}}\}. \tag{7}$$

Using (6) and (7), we discuss in the following the critical paths of direct-form and transpose-form LMS adaptive filters.

A. Critical Path of Direct Form

To find the critical path C_{ERROR} of the direct-form LMS adaptive filter, let us consider the implementation of an inner product $w(0)x(0)+w(1)x(1)+w(2)x(2)+w(3)x(3)$ of length 4. The implementation of this inner product is shown in Fig. 9, where all multiplications proceed concurrently, and additions of product words start as soon as the LSBs of products are available. Computations of the first-level adders (ADD-1 and ADD-2) are completed in time $T_{\text{MULT}} + T_{\text{FAC}} + T_{\text{FAS}}$, where

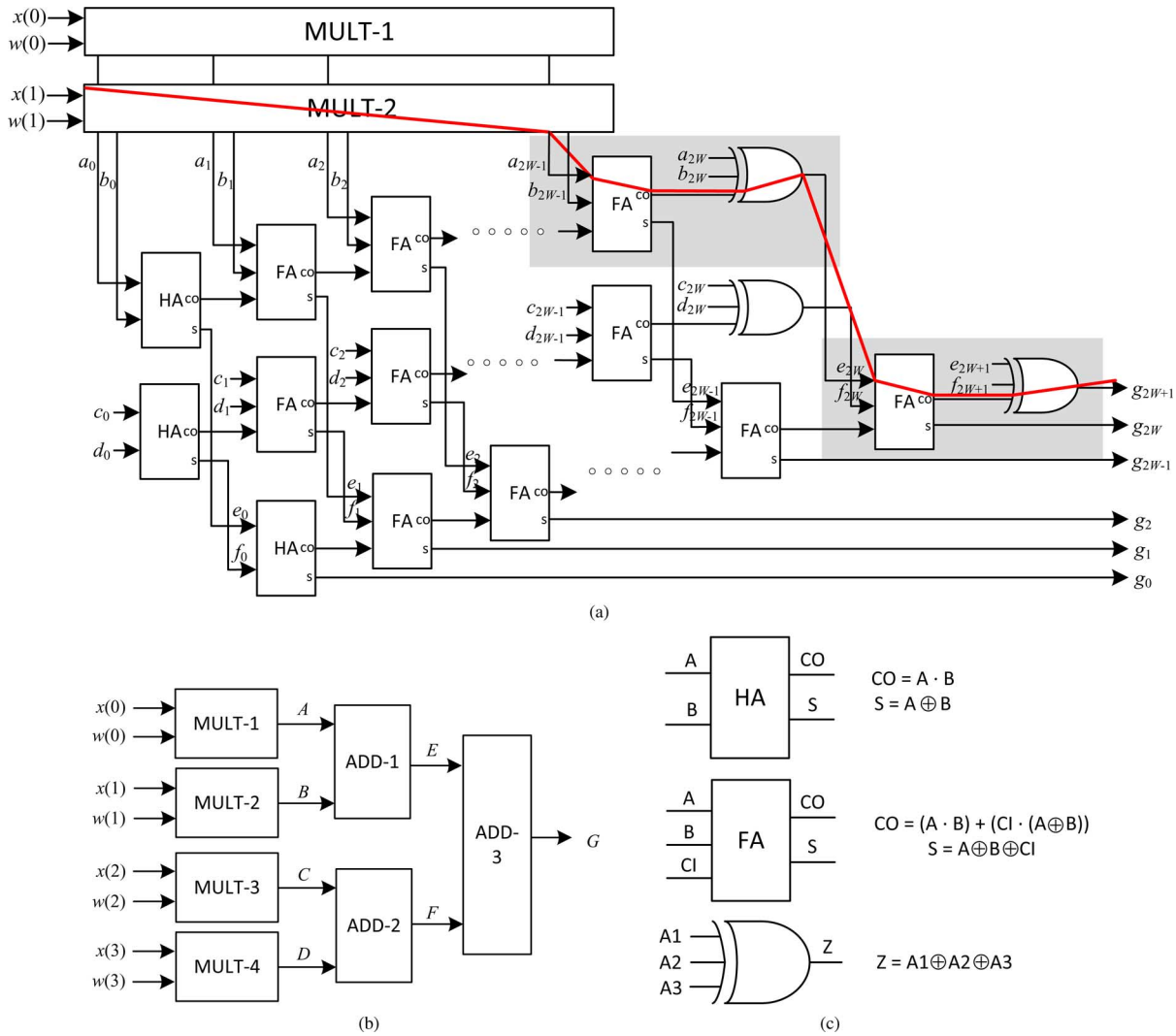


Fig. 9. Critical path of an inner product computation. (a) Detailed block diagram to show critical path of inner product computation of length 4. (b) Block diagram of inner product computation of $w(0)x(0) + w(1)x(1) + w(2)x(2) + w(3)x(3)$. (c) HA, FA, and 3-input XOR gate.

$T_{\rm FAS}$ is the delay due to the 3-input XOR operation for the addition of the last bits (without computing the carry bits), and $T_{\rm FAC} = T_{\rm AND} + T_{\rm XOR}$, where $T_{\rm AND}$ and $T_{\rm XOR}$ are the propagation delays of AND and XOR operations, respectively. For convenience of representation, we take

$$\Delta = T_{\rm FAC} + T_{\rm FAS}. \tag{8}$$

Similarly, the addition of the second-level adder (ADD-3) (and hence the inner-product computation of length 4) is completed in time $T_{\rm MULT} + 2\Delta$. In general, an inner product of length N (shown in Fig. 4) involves a delay of

$$T_{\rm IP} = T_{\rm MULT} + \lceil \log_2 N \rceil \Delta. \tag{9}$$

In order to validate (9), we show in Table II the time required for the computation of inner products of different length for word-length 8 and 16 using TSMC 0.13-μm and 90-nm process libraries. Using multiplication time and time required for carry-and-sum generation in a 1-bit full-adder, obtained from Table I, we find that the results shown in Table II are in conformity with those given by (9). The critical path of the error-computation block therefore amounts to

$$C_{\rm ERROR} = T_{\rm MULT} + (\lceil \log_2 N \rceil + 1)\Delta. \tag{10}$$

TABLE II
SYNTHESIS RESULT OF INNER PRODUCT COMPUTATION TIME (NS) USING TSMC 0.13-μM AND 90-NM CMOS TECHNOLOGY LIBRARIES

Inner Product Length	0.13-μm		90-nm	
	$L = 8$	$L = 16$	$L = 8$	$L = 16$
2	2.79	4.90	1.83	3.29
4	3.19	5.30	2.10	3.56
8	3.60	5.71	2.37	3.82
16	4.00	6.11	2.64	4.09
32	4.40	6.51	2.90	4.36

For computation of the weight-update unit shown in Fig. 5, if we assume the step-size μ to be a power of 2 fraction, i.e., of the form $1/2^k$, then the multiplication with μ can be implemented by rewiring, without involving any hardware or time delay. The

critical path then consists of a multiply-add operation, which can be shown to be

$$C_{\mathrm{UPDATE}} = T_{\mathrm{MULT}} + \Delta. \tag{11}$$

Using (6), (10), and (11), we can find the critical path of the non-pipelined direct-form LMS adaptive filter to be

$$T = 2T_{\mathrm{MULT}} + (\lceil \log_2 N \rceil + 2)\Delta. \tag{12}$$

If the error computation and weight updating are performed in two pipelined stages, then from (7), we can find the critical path to be

$$\begin{aligned} T &= \max\{T_{\mathrm{MULT}} + (\lceil \log_2 N \rceil + 1)\Delta, T_{\mathrm{MULT}} + \Delta\} \\ &= T_{\mathrm{MULT}} + (\lceil \log_2 N \rceil + 1)\Delta. \end{aligned} \tag{13}$$

This could be further reduced if we introduce delays in the error-computation block to have a pipelined implementation.

B. Critical Path of Transpose Form

In the error-computation block of the transpose-form LMS adaptive filter (Fig. 7), we can see that all multiplications are performed simultaneously, which involves time T_{MULT}. After multiplications, the results are transferred through preceding registers to be added with another product word in the next cycle. Since the addition operation starts as soon as the first bit of the product word is available (as in the direct-form LMS), the critical path of the error-computation block is

$$C_{\mathrm{ERROR}} = T_{\mathrm{MULT}} + \lceil \log_2(N+1) \rceil \Delta. \tag{14}$$

If one delay is inserted after the computation of y_n, then the critical path given by (14) will change to $T_{\mathrm{MULT}} + \lceil \log_2 N \rceil \Delta$. We have assumed here that the critical path is comprised of the last multiply-add operation to compute the filter output. Note that as the sum of product words traverses across the adder line, more and more product words are accumulated, and the width of the accumulated sum finally becomes $2L + \lceil \log_2(N+1) \rceil$, where L is the width of the input as well as the weight values.

The critical path of the weight-updating block is similarly found to be

$$C_{\mathrm{UPDATE}} = T_{\mathrm{MULT}} + \Delta. \tag{15}$$

However, for $m = 1$, i.e., the delay is inserted after y_n only, the critical path will include the additional delay introduced by the subtraction for the computation of the error term, and $C_{\mathrm{UPDATE}} = T_{\mathrm{MULT}} + 2\Delta$. Without any adaptation delay, the critical path would be

$$T = 2T_{\mathrm{MULT}} + [\lceil \log_2(N+1) \rceil + 1]\Delta. \tag{16}$$

Interestingly, the critical paths of the direct-form and transpose-form structures without additional adaptation delay are nearly the same. If the weight updating and error computation in the transpose-form structure happen in two different pipeline stages, the critical path of the complete transpose-form adaptive filter structure with adaptation delay $m = 2$, amounts to

$$T = T_{\mathrm{MULT}} + \lceil \log_2 N \rceil \Delta. \tag{17}$$

From (13) and (17), we can find that the critical path of the transpose-form DLMS adaptive filter is nearly the same as that of direct-form implementation where weight updating and error computation are performed in two separate pipeline stages.

C. Proposed Design Strategy

We find that the direct-form FIR structure not only is the natural candidate for implementation of the LMS algorithm in its original form, but also provides better convergence speed with the same residual MSE. It also involves less register complexity and nearly the same critical path as the transpose-form structure. Therefore, we have preferred to design a low-complexity direct-form structure for implementation of the LMS adaptive filter.

From Tables I and II, we can find that the critical path of the direct-implementation LMS algorithm is around 7.3 ns for filter length $N \sim 100$ with 16-bit implementation using the 0.13-μm technology library, which can be used for sampling rate as high as 100 Msps. The critical path increases by one full-adder delay (nearly 0.2 ns) when the filter order is doubled. So, for filter order $N \sim 1000$, the critical path still remains within 8 ns. On the other hand, the highest sampling frequency of LTE-Advanced amounts to 30.72 Msps [14]. For still higher data rates, such as those of some acoustic echo cancelers, we can have structures with one and two adaptation delays, which can respectively support about twice and thrice the sampling rate of the zero-adaptation delay structure.

IV. Proposed Structure

In this section, we discuss area- and power-efficient approaches for the implementation of direct-form LMS adaptive filters with zero, one, and two adaptation delays.

A. Zero Adaptation Delay

As shown in Fig. 3, there are two main computing blocks in the direct-form LMS adaptive filter, namely, i) the error-computation block (shown in Fig. 4) and ii) the weight-update block (shown in Fig. 5). It can be observed in Figs. 4 and 5 that most of the area-intensive components are common in the error-computation and weight-update blocks: the multipliers, weight registers, and tapped-delay line. The adder tree and subtractor in Fig. 4 and the adders for weight updating in Fig. 5, which constitute only a small part of the circuit, are different in these two computing blocks. For the zero-adaptation-delay implementation, the computation of both these blocks is required to be performed in the same cycle. Moreover, since the structure is of the non-pipelined type, weight updating and error computation cannot occur concurrently. Therefore, the multiplications of both these phases could be multiplexed by the same set of multipliers, while the same registers could be used for both these

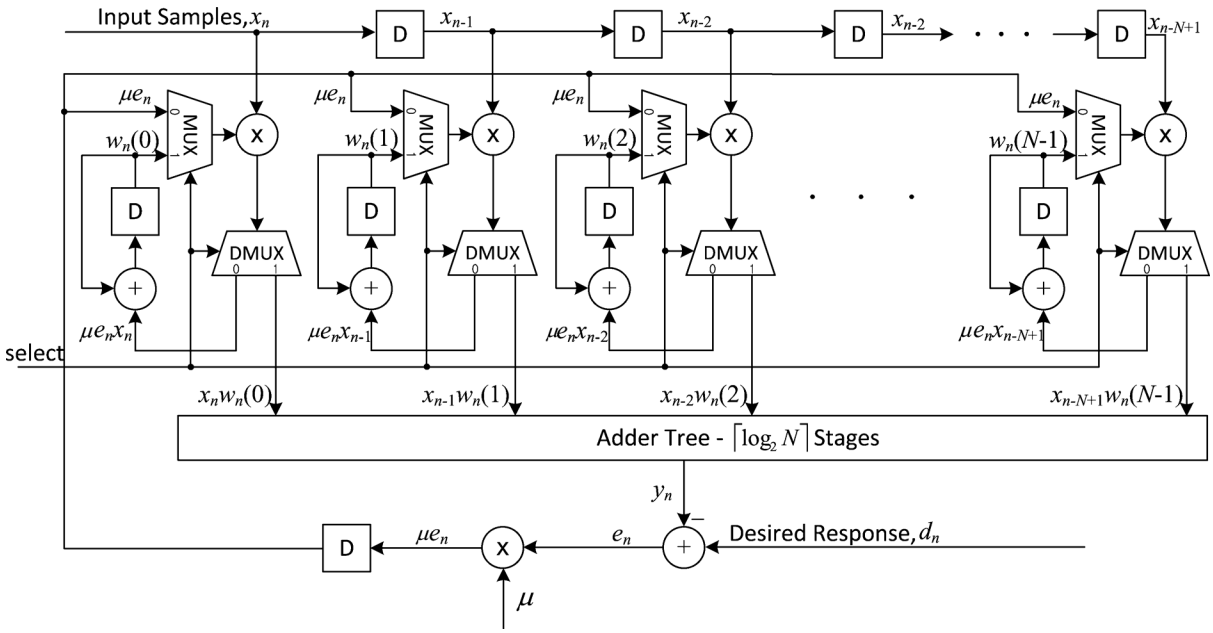


Fig. 10. Proposed structure for zero-adaptation-delay time-multiplexed direct-form LMS adaptive filter.

phases if error computation is performed in the first half cycle, while weight update is performed in the second-half cycle.

The proposed time-multiplexed zero-adaptation-delay structure for a direct-form N-tap LMS adaptive filter is shown in Fig. 10, which consists of N multipliers. The input samples are fed to the multipliers from a common tapped delay line. The N weight values (stored in N registers) and the estimated error value (after right-shifting by a fixed number of locations to realize multiplication by the step size μ) are fed to the multipliers as the other input through a 2:1 multiplexer. Apart from this, the proposed structure requires N adders for modification of N weights, and an adder tree to add the output of N multipliers for computation of the filter output. Also, it requires a subtractor to compute the error value and N 2:1 de-multiplexors to move the product values either towards the adder tree or weight-update circuit. All the multiplexors and de-multiplexors are controlled by a clock signal.

The registers in the delay line are clocked at the rising edge of the clock pulse and remain unchanged for a complete clock period since the structure is required to take one new sample in every clock cycle. During the first half of each clock period, the weight values stored in different registers are fed to the multiplier through the multiplexors to compute the filter output. The product words are then fed to the adder tree though the de-multiplexors. The filter output is computed by the adder tree and the error value is computed by a subtractor. Then the computed error value is right-shifted to obtain μe_n and is broadcasted to all N multipliers in the weight-update circuits. Note that the LMS adaptive filter requires at least one delay at a suitable location to break the recursive loop. A delay could be inserted either after the adder tree, after the e_n computation, or after the μe_n computation. If the delay is placed just after the adder tree, then the critical path shifts to the weight-updating circuit and gets increased by $T_{\rm ADD}$. Therefore, we should place the delay after computation of e_n or μe_n, but preferably after μe_n computation to reduce the register width.

The first half-cycle of each clock period ends with the computation of μe_n, and during the second half cycle, the μe_n value is fed to the multipliers though the multiplexors to calculate $\mu e_n x_n$ and de-multiplexed out to be added to the stored weight values to produce the new weights according to (2a). The computation during the second half of a clock period is completed once a new set of weight values is computed. The updated weight values are used in the first half-cycle of the next clock cycle for computation of the filter output and for subsequent error estimation. When the next cycle begins, the weight registers are also updated by the new weight values. Therefore, the weight registers are also clocked at the rising edge of each clock pulse.

The time required for error computation is more than that of weight updating. The system clock period could be less if we just perform these operations one after the other in every cycle. This is possible since all the register contents also change once at the beginning of a clock cycle, but we cannot exactly determine when the error computation is over and when weight updating is completed. Therefore, we need to perform the error computation during the first half-cycle and the weight updating during the second half-cycle. Accordingly, the clock period of the proposed structure is twice the critical-path delay for the error-computation block $C_{\rm ERROR}$, which we can find using (14) as

$$C_{\rm ERROR} = 2[T_{\rm MULT} + (\lceil \log_2 N \rceil + 1)\Delta + 2T_{\rm MUX}] \quad (18)$$

where $T_{\rm MUX}$ is the time required for multiplexing and de-multiplexing.

B. One Adaptation Delay

The proposed structure for a one-adaptation-delay LMS adaptive filter consists of one error-computation unit as shown in Fig. 4 and one weight-update unit as shown in Fig. 5. A pipeline latch is introduced after computation of μe_n. The

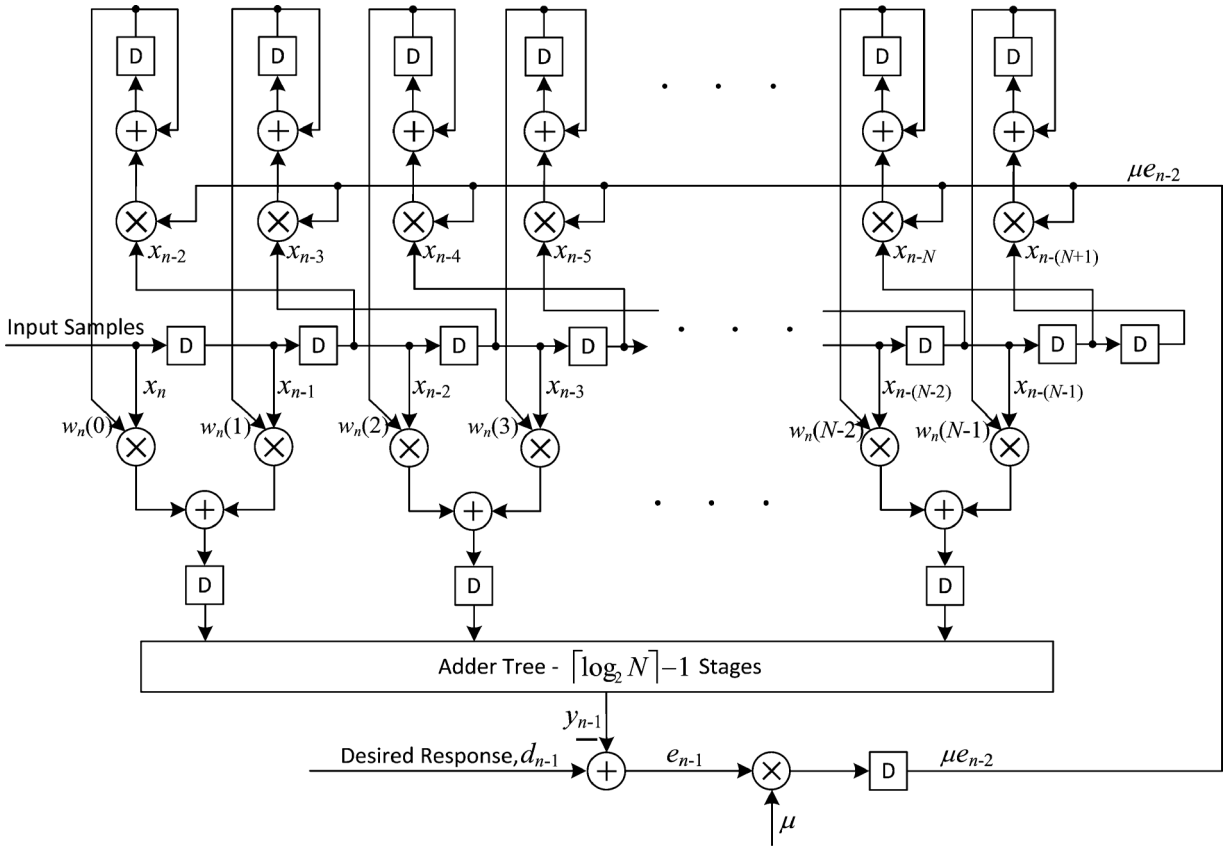


Fig. 11. Proposed structure for two-adaptation-delay direct-form LMS adaptive filter.

multiplication with μ requires only a hardwired shift, since μ is assumed to be a power of 2 fraction. So there is no register overhead in pipelining. Also, the registers in the tapped delay line and filter weights can be shared by the error-computation unit and weight-updating unit. The critical path of this structure is the same as $C_{\rm ERROR}$ [derived in (10)], given by

$$T = T_{\rm MULT} + (\lceil \log_2 N \rceil + 1)\Delta. \tag{19}$$

C. Two Adaptation Delays

The proposed structure for a two-adaptation-delay LMS adaptive filter is shown in Fig. 11, which consists of three pipeline stages, where the first stage ends after the first level of the adder tree in the error-computation unit, and the rest of the error-computation block comprises the next pipeline stage. The weight-update block comprises the third pipeline stage. The two-adaptation-delay structure involves $N/2$ additional registers over the one-adaptation-delay structure. The critical path of this structure is the same as either that of the weight-update unit $C_{\rm UPDATE}$ [derived in (11)] or the second pipeline stage, given by

$$T = \max\{T_{\rm MULT} + \Delta, T_{\rm AB}\} \tag{20}$$

where $T_{\rm AB}$ refers to the adder-tree delay of $\lceil \log_2 N \rceil - 1$ stages to add $N/2$ words along with the time required for subtraction in the error computation.

D. Structure for High Sampling Rate and Large-Order Filters

We find that in many popular applications like channel equalization and channel estimation in wireless communication, noise cancellation in speech processing, and power-line interference cancellation, removal of muscle artifacts, and electrode motion artifacts for ECG [16]–[22], the filter order could vary from 5 to 100. However, in some applications like acoustic echo cancellation and seismic signal acquisition, the filter order requirement could be more than 1000 [23]–[25]. Therefore, we discuss here the impact of increase in filter order on critical path along with the design considerations for implementation of large order filters for high-speed applications.

For large-order filters, i.e., for large N, the critical-path delay for 1-stage pipeline implementation in (19) increases by Δ when the filter order is doubled. For 2-stage pipeline implementation, $T_{\rm AB}$ in (20) could be larger than $T_{\rm MULT} + \Delta$, and could be the critical-path delay of the structure. $T_{\rm AB}$ also increases by Δ when the filter order is doubled. When 90-nm CMOS technology is used, the critical-path delay could be nearly 5.97 ns and 3.66 ns for 1 and 2-stage pipeline implementations, respectively, when $N \sim 1000$ and $L = 16$. Therefore, in order to support input sampling rates higher than 273 Msps, additional delays could be incorporated at the tail-end of the adder tree using only a small number of registers. Note that if a pipeline stage is introduced just before the last level of addition in the adder tree, then only one pipeline register is required. If we introduce the pipeline stage at k levels up from the last adder in the adder tree, then we need 2^k additional registers. The delay of the adder block however does not increase fast with the filter

TABLE III
COMPARISON OF HARDWARE AND TIME COMPLEXITIES OF DIFFERENT ARCHITECTURES

Design	Critical-Path Delay	Adaptation Delay (m)	Hardware Elements		
			# adders	# multipliers	# delays
Direct-form LMS (Figs.4, 5)	$2T_{\text{MULT}} + (\lceil \log_2 N \rceil + 2)\Delta$	0	$2N$	$2N$	$2N-1$
Transpose-form LMS (Fig. 7)	$2T_{\text{MULT}} + [\lceil \log_2(N+1) \rceil + 1]\Delta$	–	$2N$	$2N$	$3N-2$
Long et al. [3]	$T_{\text{MULT}} + \Delta$	$\log_2 N + 1$	$2N$	$2N$	$3N + 2\log_2 N + 1$
Yi et al. (TDF-RDLMS) [10]	T_{MULT}	$\log_2 N + 3$	$2N$	$2N$	$6N + \log_2 N + 2$
Yi et al. (TF-RDLMS) [10]	T_{MULT}	–	$2N+1$	$2N$	$7N+2$
Van and Feng [5]	$T_{\text{MULT}} + \Delta$	$N/4 + 3$	$2N$	$2N$	$5N+3$
Meher and Park [9]	T_{ADD}	5	$10N+2$	0	$2N + 14 + E^{\dagger}$
Proposed Design 1	$2[T_{\text{MULT}} + 2T_{\text{MUX}} + (\lceil \log_2 N \rceil + 1)\Delta]$	0	$2N$	N	$2N$
Proposed Design 2	$T_{\text{MULT}} + (\lceil \log_2 N \rceil + 1)\Delta$	1	$2N$	$2N$	$2N+1$
Proposed Design 3	$\max\{T_{\text{MULT}} + \Delta, T_{\text{AB}}\}$	2	$2N$	$2N$	$2.5N+2$

$^{\dagger}E = 24,\ 40$ and 48 for $N = 8,\ 16$ and 32, respectively. Proposed Design 1 needs additional N MUX and N DMUX. It is assumed in all these structures that multiplication with the step-size does not need a multiplier.

order since the adder tree is only increased one level when the filter length is doubled, and introduces only one extra delay of Δ in the critical path.

The critical path could be reduced only incrementally if we pipeline the adaptive filter after every addition, which will involve enormous register complexity. For a further increase in clock rate, one can use the block-LMS adaptive filter [26]. A block-LMS adaptive filter with block length B would support B times higher sampling rate without increasing the energy per sample (EPS). Therefore, pipelining of the multiplication block or adder tree after every addition is not a preferable option to implement adaptive filters for high-sampling rate or for large filter orders.

V. COMPLEXITY CONSIDERATIONS

The hardware and time complexities of the proposed and existing designs are listed in Table III. A transpose-form fine-grained retimed DLMS (TF-RDLMS), a tree direct-form fine-grained retimed DLMS (TDF-RDLMS) [10], the best of systolic structures [5], and our most recent direct-form structure [9] are compared with the proposed structures. The proposed design with 0, 1, and 2 adaptation delays (presented in Section IV) are referred to as proposed Design 1, Design 2, and Design 3, in Table III. The direct-form LMS and transpose-form LMS algorithm based on the structure of Figs. 4, 5, and 7 without any adaptation delays, e.g., $m = 0$, and the DLMS structure proposed in [3] are also listed in this table for reference. It is found that proposed Design 1 has the longest critical path, but involves only half the number of multipliers of other designs except [9], and does not require any adaptation delay. Proposed Design 2 and Design 3 have less adaption delay compared to existing designs, with the same number of adders and multipliers, and involve fewer delay registers.

We have coded all the proposed designs in VHDL and synthesized them using the Synopsys Design Compiler with the TSMC 90-nm CMOS library [12] for different filter orders. The structures of [10], [5], and [9] were also similarly coded, and synthesized using the same tool. The word-length of input samples and weights are chosen to be 12, and internal data are not truncated before the computation of filter output y_n to minimize quantization noise. Then, e_n is truncated to 12 bits, while the step size μ is chosen to be $(1/2^k)$ to realize its multiplication without any additional circuitry. The data arrival time (DAT), maximum usable frequency (MUF), adaptation delay, area, area-delay product (ADP), power consumption at maximum usable frequency (PCMUF), normalized power consumption at 50 MHz, and energy per sample (EPS) are listed in Table IV. Note that power consumption increases linearly with frequency, and PCMUF gives the power consumption when the circuit is used at its highest possible frequency. All the proposed designs have significantly less PCMUF compared to the existing designs. However, the circuits need not always be operated at the highest frequency. Therefore, PCMUF is not a suitable measure for power performance. The normalized power consumption at a given frequency provides a relatively better figure of merit to compare the power-efficiency of different designs. The EPS similarly does not change much with operating frequency for a given technology and given operating voltage, and could be a useful measure.

The transpose-form structure of [10], TF-RDLMS provides the relatively high MUF, which is 8.1% more than that of proposed Design 3, but involves 19.4% more area, 10.4% more ADP, and 59.3% more EPS. Besides, the transpose-form structure [10] provides slower convergence than the proposed direct-form structure. The direct-form structure of [10], TDF-RDLMS, has nearly the same complexity as the transpose-form counterpart of [10]. It involves 13.8% more area, 8.0% more ADP and 35.6% more EPS, and 5.4% higher MUF compared with Design 3. Besides, it requires 4, 5, and 6 more adaptation delays than the proposed Design 3 for filter length 8, 16, and 32, respectively. The structure of [5] provides nearly the same MUF as that of proposed Design 3, but requires 19.0% more area, 17.6% more ADP, and 20.4% more EPS. The structure of [9] provides the highest MUF since the critical-path delay is only T_{ADD}, however, it requires more adaptation delay than the proposed designs. Also, the structure of [9] involves 4.7% less ADP, but 12.2% more area and 26.2% more EPS than the proposed Design 3. Proposed Design 1 has the minimum MUF among all the structures, but that is adequate to support the highest data rate in current communication systems. It involves the minimum area and the minimum EPS of all the designs. The direct-form structure of [10] requires 82.8% more area and 52.4% more EPS

TABLE IV
PERFORMANCE COMPARISON OF DLMS ADAPTIVE FILTER CHARACTERISTICS BASED ON SYNTHESIS USING TSMC 90-NM LIBRARY

Design	Filter Length, N	DAT (ns)	MUF (MHz)	Adaptation Delay	Area (sq.um)	ADP (sq.um×ns)	PCMUF (mW)	Normalized Power (mW)	EPS (mW×ns)
Yi et al. (TDF-RDLMS) [10]	8	3.14	318	6	48595	152588	7.16	1.21	22.25
	16	3.14	318	7	97525	306228	14.21	2.42	44.16
	32	3.14	318	8	196017	615493	28.28	4.82	87.89
Yi et al. (TF-RDLMS) [10]	8	3.06	326	–	50859	155628	8.53	1.40	25.88
	16	3.06	326	–	102480	313588	17.13	2.82	51.94
	32	3.06	326	–	206098	630659	34.35	5.65	104.13
Van and Feng [5]	8	3.27	305	5	50717	165844	6.49	1.16	20.97
	16	3.27	305	7	102149	334027	12.39	2.23	40.00
	32	3.27	305	11	205270	671232	22.23	4.04	71.66
Meher and Park [9]	8	2.71	369	5	50357	136467	8.51	1.25	22.88
	16	2.81	355	5	95368	267984	14.27	2.19	39.70
	32	2.91	343	5	185158	538809	26.44	4.22	76.05
Proposed Design 1 (no adaptation delay)	8	7.54	132	0	26335	198565	1.99	0.79	14.48
	16	8.06	124	0	53088	427889	3.73	1.58	28.84
	32	8.60	116	0	108618	934114	7.00	3.16	58.37
Proposed Design 2 (one adaptation delay)	8	3.75	266	1	41131	154241	4.00	0.83	14.74
	16	4.01	249	1	82639	331382	7.43	1.65	29.23
	32	4.27	234	1	166450	710741	13.96	3.31	58.42
Proposed Design 3 (two adaptation delays)	8	3.31	302	2	42729	141432	5.02	0.91	16.42
	16	3.31	302	2	85664	283547	9.92	1.81	32.39
	32	3.31	302	2	171979	569250	19.93	3.65	65.10

DAT: data arrival time, MUF: maximum usable frequency, ADP: area-delay product, PCMUF: power consumption at maximum usable frequency, normalized power: power consumption at 50 MHz, and EPS: energy per sample.

compared to proposed Design 1. Similarly, the structure of [5] involves 91.3% more area and 35.4% more EPS compared with proposed Design 1. Proposed Design 2 and Design 3 involve nearly the same (slightly more) EPS than the proposed Design 1 but offer nearly twice and thrice the MUF at a cost of 55.0% and 60.6% more area, respectively.

VI. CONCLUSION

Based on a precise critical-path analysis, we have derived low-complexity architectures for the LMS adaptive filter. We have shown that the direct-form and transpose-form LMS adaptive filters have nearly the same critical-path delay. The direct-from LMS adaptive filter, however, involves less register complexity and provides much faster convergence than its transpose-form counterpart since the latter inherently performs delayed weight adaptation. We have proposed three different structures of direct-form LMS adaptive filter with i) zero adaptation delay, ii) one adaptation delay, and iii) two adaptation delays. Proposed Design 1 does not involve any adaptation delay. It has the minimum of MUF among all the structures, but that is adequate to support the highest data rate in current communication systems. It involves the minimum area and the minimum EPS of all the designs. The direct-form structure of [10] requires 82.8% more area and 52.4% more EPS compared to proposed Design 1, and the transpose-form structure of [10] involves still higher complexity. The structure of [5] involves 91.3% more area and 35.4% more EPS compared with proposed Design 1. Similarly, the structure of [9] involves 80.4% more area and 41.9% more EPS than proposed Design 1. Proposed Design 3 involves relatively fewer adaptation delays and provides similar MUF as the structures of [10] and [5]. It involves slightly less ADP but provides around 16% to 26% of savings in EPS over the others. Proposed Design 2 and Design 3 involve nearly the same (slightly more) EPS than the proposed Design 1 but offer nearly twice or thrice the MUF at the cost of 55.0% and 60.6% more area, respectively. However, proposed Design 1 could be the preferred choice instead of proposed Design 2 and Design 3 in most communication applications, since it provides adequate speed performance, and involves significantly less area and EPS.

REFERENCES

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

[2] S. Haykin and B. Widrow, *Least-Mean-Square Adaptive Filters*. Hoboken, NJ, USA: Wiley-Interscience, 2003.

[3] G. Long, F. Ling, and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 9, pp. 1397–1405, Sep. 1989.

[4] M. D. Meyer and D. P. Agrawal, "A modular pipelined implementation of a delayed LMS transversal adaptive filter," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 1990, pp. 1943–1946.

[5] L. D. Van and W. S. Feng, "An efficient systolic architecture for the DLMS adaptive filter and its applications," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 48, no. 4, pp. 359–366, Apr. 2001.

[6] L.-K. Ting, R. Woods, and C. F. N. Cowan, "Virtex FPGA implementation of a pipelined adaptive LMS predictor for electronic support measures receivers," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 1, pp. 86–99, Jan. 2005.

[7] E. Mahfuz, C. Wang, and M. O. Ahmad, "A high-throughput DLMS adaptive algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2005, pp. 3753–3756.

[8] P. K. Meher and S. Y. Park, "Low adaptation-delay LMS adaptive filter Part-II: An optimized architecture," in *Proc. IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2011.

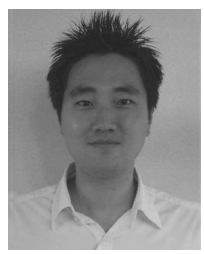
[9] P. K. Meher and S. Y. Park, "Area-delay-power efficient fixed-point LMS adaptive filter with low adaptation-delay," *Trans. Very Large Scale Integr. (VLSI) Signal Process.* [Online]. Available: http://ieeexplore.ieee.org

[10] Y. Yi, R. Woods, L.-K. Ting, and C. F. N. Cowan, "High speed FPGA-based implementations of delayed-LMS filters," *Trans. Very Large Scale Integr. (VLSI) Signal Process.*, vol. 39, no. 1–2, pp. 113–131, Jan. 2005.
[11] S. Y. Park and P. K. Meher, "Low-power, high-throughput, and low-area adaptive FIR filter based on distributed arithmetic," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 6, pp. 346–350, Jun. 2013.
[12] "TSMC 90 nm general-purpose CMOS standard cell libraries—tcbn90ghp" [Online]. Available: www.tsmc.com/
[13] TSMC 0.13 μm General-Purpose CMOS Standard Cell Libraries - tcb013ghp [Online]. Available: www.tsmc.com/
[14] 3GPP TS 36.211, Physical Channels and Modulation, ver. 10.0.0 Release 10, Jan. 2011.
[15] P. K. Meher and M. Maheshwari, "A high-speed FIR adaptive filter architecture using a modified delayed LMS algorithm," in *Proc. IEEE Int. Symp. Circuits Syst.*, May 2011, pp. 121–124.
[16] J. Vanus and V. Styskala, "Application of optimal settings of the LMS adaptive filter for speech signal processing," in *Proc. IEEE Int. Multiconf. Comput. Sci. Inf. Technol.*, Oct. 2010, pp. 767–774.
[17] M. Z. U. Rahman, R. A. Shaik, and D. V. R. K. Reddy, "Noise cancellation in ECG signals using computationally simplified adaptive filtering techniques: Application to biotelemetry," *Signal Process. Int. J. (SPIJ)*, vol. 3, no. 5, pp. 1–12, Nov. 2009.
[18] M. Z. U. Rahman, R. A. Shaik, and D. V. R. K. Reddy, "Adaptive noise removal in the ECG using the block LMS algorithm," in *Proc. IEEE Int. Conf. Adaptive Sci. Technol.*, Jan. 2009, pp. 380–383.
[19] B. Widrow, J. R. Glover, Jr., J. M. McCool, J. Kaunitz, C. S. Williams, R. H. Hearn, J. R. Zeidler, E. Dong, Jr., and R. C. Goodlin, "Adaptive noise cancelling: Principles and applications," *Proc. IEEE*, vol. 63, no. 12, pp. 1692–1716, Dec. 1975.
[20] W. A. Harrison, J. S. Lim, and E. Singer, "A new application of adaptive noise cancellation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 1, pp. 21–27, Feb. 1986.
[21] S. Coleri, M. Ergen, A. Puri, and A. Bahai, "A study of channel estimation in OFDM systems," in *Proc. IEEE Veh. Technol. Conf.*, 2002, pp. 894–898.
[22] J. C. Patra, R. N. Pal, R. Baliarsingh, and G. Panda, "Nonlinear channel equalization for QAM signal constellation using artificial neural networks," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 29, no. 2, pp. 262–271, Apr. 1999.
[23] D. Xu and J. Chiu, "Design of a high-order FIR digital filtering and variable gain ranging seismic data acquisition system," in *Proc. IEEE Southeastcon*, Apr. 1993.
[24] M. Mboup, M. Bonnet, and N. Bershad, "LMS coupled adaptive prediction and system identification: A statistical model and transient mean analysis," *IEEE Trans. Signal Process.*, vol. 42, no. 10, pp. 2607–2615, Oct. 1994.
[25] C. Breining, P. Dreiseitel, E. Hansler, A. Mader, B. Nitsch, H. Puder, T. Schertler, G. Schmidt, and J. Tilp, "Acoustic echo control," *IEEE Signal Process. Mag.*, vol. 16, no. 4, pp. 42–69, Jul. 1999.
[26] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block implementation of adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-29, no. 3, pp. 744–752, Jun 1981.

Pramod Kumar Meher (SM'03) received the B.Sc. (Honours) and M.Sc. degree in physics, and the Ph.D. degree in science from Sambalpur University, India, in 1976, 1978, and 1996, respectively.

Currently, he is a Senior Research Scientist, with the School of Computer Engineering, Nanyang Technological University, Singapore. Previously, he was a Senior Scientist with the Institute for Infocomm Research, Singapore, and Senior Fellow with the School of Computer Engineering, Nanyang Technological University, Singapore. He was a Professor of Computer Applications with Utkal University, India, from 1997 to 2002, and a Reader in electronics with Berhampur University, India, from 1993 to 1997. His research interest includes design of dedicated and reconfigurable architectures for computation-intensive algorithms pertaining to signal, image and video processing, communication, bio-informatics and intelligent computing. He has contributed nearly 200 technical papers to various reputed journals and conference proceedings.

Dr. Meher has served as a speaker for the Distinguished Lecturer Program (DLP) of IEEE Circuits Systems Society during 2011 and 2012 and Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART II: EXPRESS BRIEFS during 2008 to 2011. Currently, he is serving as Associate Editor for the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—PART I: REGULAR PAPERS, the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, and the *Journal of Circuits, Systems, and Signal Processing*. Dr. Meher is a Fellow of the Institution of Electronics and Telecommunication Engineers, India. He was the recipient of the Samanta Chandrasekhar Award for excellence in research in engineering and technology for 1999.



Sang Yoon Park (S'03–M'11) received the B.S. degree in electrical engineering and the M.S. and Ph.D. degrees in electrical engineering and computer science from Seoul National University, Seoul, Korea, in 2000, 2002, and 2006, respectively.

He joined the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, as a Research Fellow in 2007. Since 2008, he has been with Institute for Infocomm Research, Singapore, where he is currently a Research Scientist. His research interest includes design of dedicated and reconfigurable architectures for low-power and high-performance digital signal processing systems.