International Journal on Applications in Information and Communication Engineering
Volume 1: Issue 5: May 2015, pp 8-13. www.aetsjournal.com ISSN (Online) : 2394-6237

Realisation of Lms Adaptive Algorithm Using Verilog Hdl For Low Complexity

Ashaganga.G.R Suresh.S

Abstract— This paper presents 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).

Index Terms—Adaptive filters, optimization, least mean square algorithms, LMS adaptive filter, delayed least mean square algorithm**.**

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

$\mathbf{w}_{n+1} = \mathbf{w}_n + \mu \mathbf{e}_n \mathbf{x}_n$ (1a)

Where

$\mathbf{e}_n = \mathbf{d}_n - \mathbf{y}_n$ (1b) $\mathbf{y}_n = \mathbf{w}^T_n \mathbf{x}_n$ (1c)

with input vectorxn and weight vector $w_n$at the nth iteration are given by, respectively,

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

Ashaganga.G.R, PG Scholar, M.E VLSI Design
Suresh.S , Assistant Professor

and where $\mathbf{d}_n$is the desired response,$\mathbf{y}_n$ is the filter output of the nth iteration, $\mathbf{e}_n$denotes the error computed during the nth iteration,which is used to update the weights,$\boldsymbol{\mu}$ is the convergence factor or step-size, which is usually assumed to be a positive number, and 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 inputoperand 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 can proceed only after completion of the multiplication, the time required for a multiplication and an addition, respectively. 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.

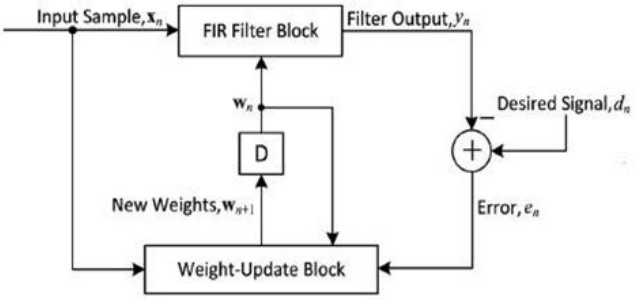


Fig.1. Structure of conventional LMS adaptive filter.

Therefore, it is modified to a form called the delayed LMS (DLMS) algorithm 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 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.

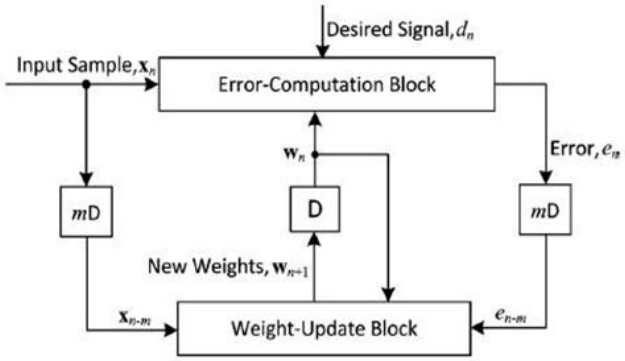


Fig 2 A generalized block diagram of direct-form DLMS adaptivefilter

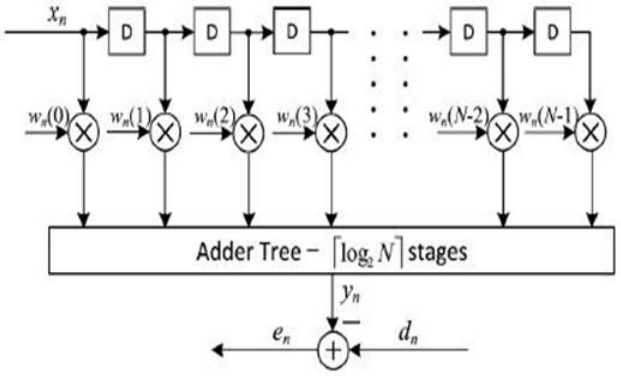


Fig 3 Error computation block

On the other hand, we can assume that an arithmetic operation 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. 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.

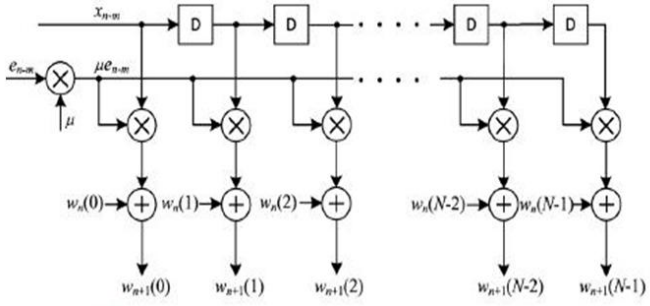


Fig 4 Weight-update block

Due to that, the designs of [8], [9], [15] still consume higher area, which could be substantially reduced. Keeping the aboveobservations 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 behaviour. The proposed low complexity designs is discussed below

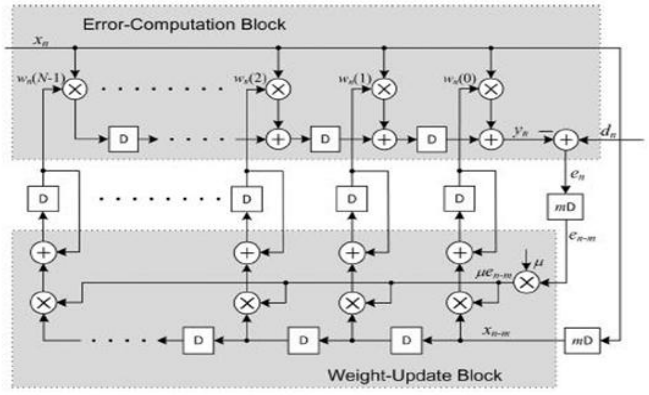


Fig 5 The structure of the transpose-form DLMS adaptive filter.

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.

A. Implementation of Direct-Form Delayed LMS Algorithm

Assuming error computation block is implemented in pipelined stages, the latency of error computation is cycles, so that the error computed by the structure at the nth cycle is , and is used with the input samples delayed by cycles to generate the weight-increment term. The weight-update equation of the

DLMS algorithm is given by

$^{l}w_{n+1} = w_n + \mu e_{n-m} x_{n-m}$ (2a)

Where, $e_{n-m} = d_{n-m} - y_{n-m}$ (2b)and the new updated weight

$y_{n-m} = w^T_n x_n$(3)

The number of delays shown in Fig.2 corresponds to the pipeline delays introduced due to pipelining of the error-computation block. Direct-form adaptive filters with different values of adaptation delay are simulated for a system identification problem, where the system is defined by a band-pass filter with impulse Response.

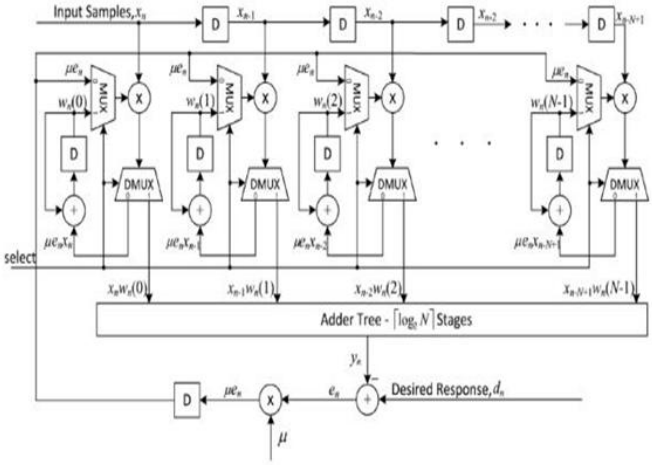


Fig 7 Proposed structure for zero adaptation delay

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:

Where, 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 additional delays are introduced in the error computation at any instant, then the weights are required to be updated according to the following equation

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

but, the equation to compute the filter output remains the same as that of (4a).

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 is updated cycles back,$w_{n-k}(k)$. 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 . We find that the direct from LMS adaptive filter provides much faster convergence than the transpose LMS adaptive filter in all cases. The direct-formDLMS adaptive filter with delay 5 also provides faster convergence compared to the transpose-form LMS adaptivefilter 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.In the error-computation block of the transpose-form LMS adaptive filter (Fig. 7), we can see that all multiplications areperformed simultaneously, which involves time . 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.Thecritical path of the complete transpose-form DLMS adaptive filter is nearly the same as thatof 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 residualMSE. 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..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.

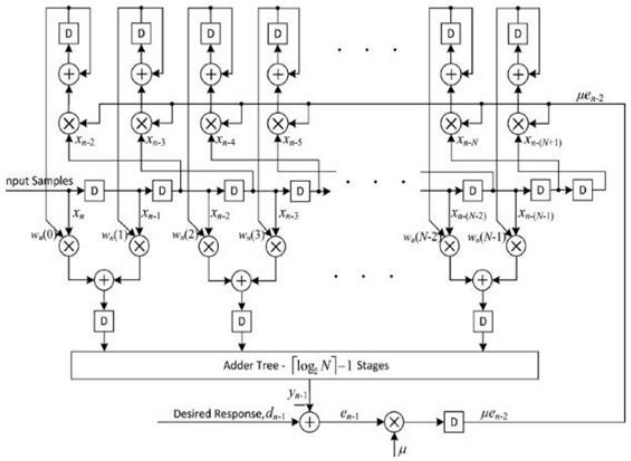
III. Proposed Strucuture

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

There are two main computing blocks in the direct-form LMS adaptive filter, namely, i) the error-computationblock (shown in Fig. 4) and ii) the weight-update block(shown in Fig. 5). It can be observed in Figs. 4 and 5 that mostof the area-intensive components are common in the error-computationand weight-update blocks: the multipliers, weight registers, and tapped-delay line. The adder tree and sub tractor in Fig. 4 and the adders for weight updating in Fig. 5, which constituteonly a small part of the circuit, are different in these twocomputing 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 isof the non-pipelined type, weight updating and error computation cannot occur concurrently. Therefore, themultiplications ofboth these phases could be multiplexed by the same set of multipliers, while the same registers could be used for both thesephases 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 structurefor a direct-form -tap LMS adaptive filter is shown in Fig. 7, which consists of multipliers. The input samples arefed to the multipliers from a common tapped delay line. Theweight values (stored in registers) and the estimated error value (after right-shifting by a fixed number of locations to realizemultiplication by the step size) are fed to the multipliersas the other input through a 2:1 multiplexer. Apart from this, the proposed structure requires adders for modification ofweights, and an adder tree to add the output of multipliers forcomputation of the filter output. Also, it requires a subtractor to compute the error value and 2:1demultiplexors to move theproduct values either towards the adder tree or weight-updatecircuit. All the multiplexors and de-multiplexors are controlled by a clock signal. The registers in the delay line are clocked at the rising edgeof the clock pulse and remain unchanged for a complete clockperiod 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 multiplierthrough the multiplexors to compute the filter output. Theproduct words are then fed to the adder tree though the demultiplexors. The filter output is computed by the adder tree andthe error value is computed by a subtractor. Then the computederror value is right-shifted to obtain and is broadcasted toall multipliers in the weight-update circuits. Note that theLMS adaptive filter requires at least one delay at a suitable locationto break the recursive loop. A delay could be insertedeither after the adder tree, after the computation, or after thecomputation. If the delay is placed just after the adder tree, then the critical path shifts to the weight-updating circuit andgets increased. Therefore, we should place the delay after computation of or, but preferably after computationto reduce the register width. The first half-cycle of each clock period ends with thecomputation of , and during the second half cycle, thevalue is fed to the multipliers though the multiplexors to calculate and de-multiplexed out to be added to thestored weight values to produce the new weights according to. The computation during the second half of a clock period is completed once a new set of weight values is computed. Theupdated weight values are used in the first half-cycle of thenext clock cycle for computation of the filter output and forsubsequent error estimation. When the next cycle begins, theweight registers are also updated by the new weight values. Therefore, the weight registers are also clocked at the risingedge of each clock pulse. The time required for error computation is more than thatof weight updating. The system clock period could be less ifwe just perform these operations one after the other in every cycle. This is possible since all the register contents also changeonce at the beginning of a clock cycle, but we cannot exactly determine when the error computation is over and when weightupdating is completed. Therefore, we need to perform the error computation during the first half-cycle and the weight updatingduring the second half-cycle. Accordingly, the clock period ofthe proposed structure is twice the critical-path delay for the error-computation block.

B. One Adaptation Delay

The proposed structure for a one-adaptation-delay LMSadaptive filter consists of one error-computation unit as shownin Fig. 4 and one weight-update unit as shown in Fig. 5. A pipeline latch is introduced after computation of μe_n . Themultiplication with requires only a hardwired shift, sinceis assumed to be a power of 2 fraction. So there is no register overhead in pipelining. Also, the registers in the tapped delayline and filter weights can be shared by the error-computationunit and weight-updating unit.

C. Two Adaptation Delays

The proposed structure for a two-adaptation-delay LMSadaptive filter is shown in Fig.8, which consists of threepipeline stages, where the first stage ends after the first level of the adder tree in the error-computation unit, and the rest of theerror-computation block comprises the next pipeline stage. The weight-update block comprises the third pipeline

stage. Thetwo-adaptation-delay structure involves N/2additional registers over the one-adaptation-delay structure. The critical pathof this structure is the same as either that of the weight-update unit

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

We find that in many popular applications like channelequalization and channel estimation in wireless communication, noise cancellation in speech processing, and power-line interference cancellation, removal of muscle artefacts, and electro demotion artefacts for ECG the filter order couldvary from 5 to 100. However, in some applications like acoustic echo cancellation and seismic signal acquisition, the filter orderrequirement could be more than 1000. Therefore, wediscuss here the impact of increase in filter order on criticalpath along with the design considerations for implementationof large order filters for high-speed applications. Therefore, in order to support input sampling rates higher than 273 Msps, additional delays could be incorporated at the tail-end of the adder treeusing only a small number of registers. Note that if a pipelinestage is introduced just before the last level of addition in theadder tree, then only one pipeline register is required. If we introduce the pipeline stage at levels up from the last adder inthe adder tree, then we need additional registers. The delayof the adder block however does not increase fast with the filterorder since the adder tree is only increased one level when the filter length is doubled, and introduces only one extra delay of in the critical path. The critical path could be reduced only incrementally if wepipeline the adaptive filter after every addition, which will involveenormous register complexity. For a further increase inclock rate, one can use the block-LMS adaptive filter [26]. Ablock-LMS adaptive filter with block length would support times higher sampling rate without increasing the energy persample (EPS). Therefore, pipelining of the multiplication blockor adder tree after every addition is not a preferable option toimplement adaptive filters for high-sampling rate or for largefilter orders.

Table.1 Comparison for area, power and ADP for proposed design

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

IV. COMPLEXITY CONSIDERATIONS

A transpose-form fine-grained retimed DLMS (TF-RDLMS), a tree direct-form fine-grained retimed DLMS(TDF-RDLMS) [10], the best of systolicstructures [5], and our most recent direct-form structure [9] arecompared 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 algorithmbased on the structure of Figs. 4, 5, and 7 without any adaptation delays, e.g., m=0 , and the DLMS structure proposedin [3] are also listed in this table for reference. It is foundthat proposed Design 1 has the longest critical path, but involvesonly 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 andsynthesized them using the Synopsys Design Compiler withthe 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 dataare not truncated before the computation of filter outputto minimize quantization noise. Then, is truncated to 12 bits, while the step size is chosen to be to realize itsmultiplication without any additional circuitry. The data arrivaltime (DAT), maximum usable frequency (MUF), adaptation delay, area, area-delay product (ADP), power consumptionat maximum usable frequency (PCMUF), normalized powerconsumption at 50MHz, and energy per sample (EPS) with frequency, and PCMUF gives the power consumptionwhen the circuit is used at its highest possible frequency. Allthe proposed designs have significantly less PCMUF comparedto the existing designs. However, the circuits need not alwaysbe operated at the highest frequency. Therefore, PCMUF is not a suitable measure for power performance. The normalizedpower consumption at a given frequency provides a relativelybetter figure of merit to compare the power-efficiency of differentdesigns. The EPS similarly does not change much withoperating 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 proposedDesign 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 6more adaptation delays than theproposed Design 3 for filter length 8, 16, and 32, respectively. The structure of [5] provides nearly the same MUF as that ofproposed Design 3, but requires 19.0% more area, 17.6% more ADP, and 20.4% more EPS. The structure of [9] provides thehighest MUF since the

critical-path delay is only , however,it requires more adaptation delay than the proposed designs. Also, the structure of [9] involves 4.7% less ADP, but12.2% more area and 26.2% more EPS than the proposed Design3. Proposed Design 1 has the minimum MUF among all the structures, but that is adequate to support the highest datarate in current communication systems. It involves the minimum area and the minimum EPS of all the designs. The direct-formstructure of [10] requires 82.8% more area and 52.4% more EPScompared to proposed Design 1. Similarly, the structure of [5]involves 91.3% more area and 35.4% more EPS compared withproposed Design 1. Proposed Design 2 and Design 3 involve nearly the same (slightly more) EPS than the proposed Design1 but offer nearly twice and thrice the MUF at a cost of 55.0% and 60.6% more area, respectively

V.CONCLUSION

We have shown that the direct-form and transpose-form LMS adaptive filters have nearly the same critical-path delay. Thedirect-from LMS adaptive filter, however, involves less registercomplexity and provides much faster convergence than its transpose-form counterpart since the latter inherently performsdelayed weight adaptation. We have proposed three differentstructures of direct-form LMS adaptive filter with i) zero adaptationdelay, ii) one adaptation delay, and iii) two adaptationdelays. Proposed Design 1 does not involve any adaptationdelay. It has the minimum of MUF among all the structures, but that is adequate to support the highest data rate in currentcommunication systems. It involves the minimum area and theminimum EPS of all the designs. The direct-form structure of[10] requires 82.8% more area and 52.4% more EPS comparedto 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 comparedwith proposed Design 1. Similarly, the structure of [9] involves80.4% more area and 41.9% more EPS than proposed Design 1.Proposed Design 3 involves relatively fewer adaptation delaysand provides similar MUF as the structures of [10] and [5]. Itinvolves slightly less ADP but provides around 16% to 26% of savings in EPS over the others. Proposed Design 2 and Design 3involve nearly the same (slightly more) EPS than the proposed Design 1 but offer nearly twice or thrice the MUF at the cost of55.0% and 60.6% more area, respectively.

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 algorithmwith 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, ―Amodular 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, andC. F. N. Cowan, ―VirtexFPGAimplementation 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, ―Lowadaptation-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 788 IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—I: REGULAR PAPERS, VOL. 61, NO. 3, MARCH 2014

[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 m 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*, vol63, no. 12, pp. 1692–1716, Dec. 1975.