A Pipelined LMS Adaptive Filter Architecture[1]

N. R. Shanbhag K. K. Parhi

Dept. of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455

Abstract

A fine-grain pipelined architecture for least mean-square (LMS) filtering is developed by employing a stochastic form of look-ahead. The new architecture offers a trade-off between a variable output latency and the adaptation accuracy. Analytical expressions describing the convergence properties are provided. A comparison with previous work indicates that the new architecture has the least increase in hardware requirements and at the same time has the highest convergence speed in seconds. Simulation results confirm the analytical expressions derived in this paper.

1 Introduction

In the quest for high-speed architectures for recursive signal processing, algorithm transformation techniques based on *look-ahead computation* have been successfully applied to *create* concurrency in non-concurrent signal processing algorithms [1]. These include the techniques of pipelining [2] and parallel processing [3]. While making it feasible to pipeline recursive signal processing algorithms, the *look-ahead computation* [2] requires significant hardware increase. This hardware increase becomes extremely prohibitive when pipelining of adaptive filters is concerned. Existing pipelined adaptive filter architectures [4, 5] maintain exact input-output mapping with the accompanying explosion in hardware complexity. Therefore, a lot of attention has been directed towards block-processing and parallel processing [6, 7, 8] approaches. As suggested before [2], it is more natural to design a new algorithm which is inherently concurrent than creating concurrency in a traditional non-concurrent one. In designing new algorithms, we seek new topologies or architectures, which contain more loop delays, and can operate in functionally correct manner. Thus, the emphasis here is on correct *functionality*, as opposed to preserving the input-output characteristics.

[1] This research was supported by the army research office by contract number DAAL03-90-G-0063.

In this paper, we introduce a new pipelined LMS adaptive filter architecture, referred to as the PIPLMS (to be read as Pipe LMS) filter. The PIPLMS filter is designed on the basis of correct functionality assumption, and is developed using a *stochastic form of look-ahead prediction*. With the stochastic form of look-ahead we look for acceptable convergence behavior rather than invariance with respect to the input-output mapping. It is shown that the use of this approach requires only additional pipeline latches.

In section 2, we derive the PIPLMS architecture. The convergence properties of the PIPLMS are presented in section 3. Comparison of PIPLMS, in terms of hardware requirements and speed, with the corresponding serial architecture (SLMS), the block architecture (BLMS) [6] and the parallel architecture (PARLMS) [8] is carried out in section 4. In section 5, we present simulation results to verify the theoretical analysis done on PIPLMS and compare its performance with the SLMS, BLMS and PARLMS. The conclusions and future research directions are presented in section 6.

2 The Pipelined LMS Architecture

Given a desired speed-up of M, the clock period of PIPLMS should be less than or equal to $1/M$ times that of SLMS.

2.1 Pipelining the Sub-blocks

The block diagram of SLMS in (Fig.1) shows the two major sub-blocks denoted as the input tap filter block (**F**) and the weight update block (**WUD**). We can estimate the required level of pipelining by comparing the computation times of **F** and **WUD** with the desired clock-period. Let M_1 and M_2 represent the level to which the sub-blocks **F** and **WUD** need to be pipelined respectively. Pipelining **F** by M_1 levels is trivial as it is a FIR block.

Instead of employing the look-ahead transformation technique [1] to pipeline the recursive loop in **WUD**, we simply place the desired number (say M_{20}) latches in it. The remaining M_{21} latches, where

$$M_2 = M_{20} + M_{21}, \tag{2.1}$$

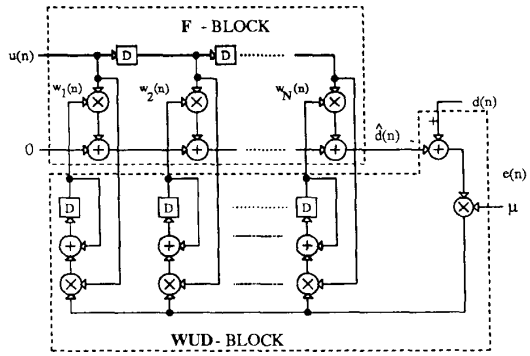

Figure 1: The SLMS architecture.

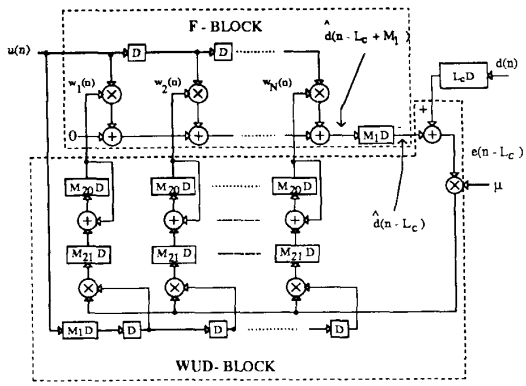

Figure 2: The PIPLMS architecture.

are placed outside the loop (see Fig.2).

Since the PIPLMS structure is not a retimed version of SLMS, a thorough convergence anaylsis needs to be carried out in order to determine the conditions under which the stochastic form of look-ahead prediction, as mentioned earlier, preserves the *overall* convergence of the weight vector and the output. In this way, apart from the pipelining latches, no other hardware overhead is required. This is a remarkable feature considering that an adaptive filter has been pipelined.

2.2 Modified Weight-update Equations

With the pipelining latches introduced in the previous sub-section, it is necessary to derive new weight-update equations (in a fashion similar to that of SLMS). In other words, we start with the expression for the expected value of the output error. We then find its gradient with respect to the weight vector ($\nabla_{\mathbf{W}}(n)$) and substitute its noisy estimate into a modified form of the steepest-descent algorithm shown below,

$$\mathbf{W}(n + M_{20}) = \mathbf{W}(n) + \frac{\mu}{2}(-\nabla_{\mathbf{W}}(n - L_c - M_{21})), \tag{2.2}$$

where μ represents the adaptation constant, L_c is the desired output latency, and we have employed the notation $\mathbf{W}(n)$ to denote the weight vector in block **F** i.e.

$$\mathbf{W}(n) = [w_1(n), w_2(n), \ldots, w_N(n)]^T, \tag{2.3}$$

where T is the transpose operator and N is the filter order.

The following final weight-update equation is obtained

$$\mathbf{W}(n+M_{20}) = \mathbf{W}(n)+\mu\mathbf{U}(n-M_1-M_{21})\epsilon^*(n-L_c-M_{21}). \tag{2.4}$$

where $*$ represents the complex conjugation operation.

2.3 Variable Latency Feature

The introduction of the L_c latches in the PIPLMS not only gives rise to a unique variable latency feature but also plays an important role in determining the adaptation accuracy. As the output of the pipelined **F** block is $\hat{d}(n - L_c)$ (Fig.2), it is clear that the output latency of PIPLMS equals L_c, which may be varied from zero to any reasonably large value.

Depending on whether L_c is less than, equal to, or greater than M_1, block **F** behaves as a $(M_1 - L_c)$-step forward, a $(L_c - M_1)$-step backward, or a zero-step predictor, respectively. Hence, only when L_c equals M_1, the adaptation accuracy is guaranteed to be equal to that of SLMS.

3 Convergence Properties of PIPLMS

In order to make the analysis tractable, we consider the following assumption which is general enough to cover all possible situations,

$$M_1 + M_{21} = M_{20}. \tag{3.1}$$

It is necessary to point out that the assumption of (3.1) can cover all possible values of M_1 and M_2. This can be seen to be true if we let $M_{21} = 0$ and $M_{20} = M_2$. Thus, we may move as many as $M_{20} - 1$ latches from the recursive loop in the **WUD** block to pipeline the rest of **WUD** and if necessary the **F** block.

3.1 Bounds on μ for Convergence

The bounds on μ to guarantee the convergence of PIPLMS was found to be tighter by a factor of 8 as compared to those for SLMS. In particular, the bounds on μ for PIPLMS were

$$0 \leq \mu \leq \frac{1}{4\lambda_{max}}, \tag{3.2}$$

TABLE I. Convergence Time-Constants

	$\tau_{k,J}$	$t_{k,J}$
SLMS	$1/(2\mu\lambda_k)$	$T_c/(2\mu\lambda_k)$
BLMS	$L/(2\mu\lambda_k)$	$LT_c/(2\mu\lambda_k)$
PARLMS	$\lceil 1/(2\mu\lambda_k M_B N_1)\rceil M_B N_1$	$\lceil 1/(2\mu\lambda_k M_B N_1)\rceil N_1 T_c$
PIPLMS	$(M_1 + M_{21})/(2\mu\lambda_k)$	$(M_1 + M_{21})T_c/(2(M_1 + M_2)\mu\lambda_k)$

while those for SLMS were

$$0 \leq \mu \leq \frac{2}{\lambda_{max}}, \tag{3.3}$$

where λ_{max} is the largest eigenvalue of the input correlation matrix $\mathbf{R}$.

3.2 Convergence Speed

The *convergence time-constant* (denoted as $\tau_{k,J}$) and the *convergence time in seconds*, $t_{k,J}$, where index k refers to the k^{th} $(k = 1, 2, \ldots, N)$ mode of convergence, for all the architectures under consideration, are compared in Table I. In Table I, L is the block-length in BLMS, and M_B is the number of filters operating in parallel on contiguous blocks of data of length N_1 for PARLMS.

Thus, in spite of having a higher $\tau_{k,J}$ than that of SLMS, PIPLMS has a lower $t_{k,J}$. This is due to its faster clock-speed.

3.3 Adaptation Accuracy

The adaptation accuracy, which is quantified in terms of the misadjustment, is defined as follows

$$\mathcal{M} = \frac{E(J(\infty)) - J_{min}}{J_{min}}, \tag{3.4}$$

where $J(n)$ is the MSE at time instant n, and $E(J(n))$ is its average. The notation J_{min} refers to the minimum mean-squared error, which would be obtained if the filter weight vector $\mathbf{W}(n)$ equalled the optimum Weiner solution $\mathbf{W}_o$.

If $L_c \neq M_1$, then the optimal solution $\mathbf{W}_o$ and hence J_{min}, for PIPLMS, would be different from that of SLMS. Therefore, we need two definitions for the misadjustment of PIPLMS ($\mathcal{M}_{1,PIP}$ and $\mathcal{M}_{2,PIP}$), as shown below

$$\mathcal{M}_{1,PIP} = \frac{E(J(\infty)_{PIP}) - J_{min,S}}{J_{min,S}} \tag{3.5}$$

$$\mathcal{M}_{2,PIP} = \frac{E(J(\infty)_{PIP}) - J_{min,PIP}}{J_{min,PIP}}, \tag{3.6}$$

where $J_{min,S}$ and $J_{min,PIP}$ are the minimum mean-squared errors for SLMS and PIPLMS respectively,

TABLE II. Adaptation accuracy.

	Misadjustment
SLMS	M_S
BLMS	M_S/L
PARLMS	M_S
PIPLMS	$M_{1,\mathrm{PIP}} = [(M_S + 1)\,\alpha - 1]$
	$M_{2,\mathrm{PIP}} = M_S$

while $E(J(\infty)_{PIP})$ is the *average steady-state mean-squared error* of PIPLMS.

In Table II, we present the misadjustments for all the architectures under consideration, in terms of the misadjustment for SLMS ($\mathcal{M}_S$). The misadjustment $\mathcal{M}_{1,PIP}$ is a linearly increasing function of the factor α, which is the ratio of $J_{min,PIP}$ to $J_{min,S}$. It can be seen that $\mathcal{M}_{1,PIP} = \mathcal{M}_S$, if $\alpha = 1$ and the latter would be true if $L_c = M_1$. The other measure of misadjustment $\mathcal{M}_{2,PIP}$, can be seen to be equal to $\mathcal{M}_S$. Thus, PIPLMS may have a higher misadjustment if we desire an output latency less than M_1.

4 Hardware and Clock-Speed Comparisons

In this section, we present the hardware requirements and the clock speed.

4.1 Hardware Requirements

The hardware requirements for all the four architectures are tabulated in Table III, from which we can see that the extra hardware requirements for PIPLMS is restricted to the $2M_1 + NM_2 + L_c$ latches. This is the only hardware penalty for using PIPLMS to achieve high processing speed. The hardware requirements in BLMS increase linearly with L. The penalty on PARLMS is quite high even for small values of M_B. This is due to the fact that the hardware overhead, which is needed to compute the weight error vector, is proportional to N^2. Therefore, PIPLMS represents an architecture with negligible hardware penalty.

TABLE III. Hardware requirements.

HARDWARE	SLMS	BLMS	PARLMS	PIPLMS
MULTIPLIERS	2N+1	L(2N+1)	$M_B(N^2+3N+4)$	2N+1
ADDERS	2N	2LN	$M_B(N^2+3N+2)$	2N
LATCHES	2N-1	N(L+1)-L	$M_B(2N-1)$	$2N-1+2M_1+NM_2+L_c$

TABLE IV. Speed.

PARAMETERS	SLMS	BLMS	PARLMS	PIPLMS
ITERATION PERIOD	T_c	T_c/L	T_c/M_B	$T_c/(M_1+M_{20})$
CLOCK PERIOD	T_c	T_c	T_c	$T_c/(M_1+M_{20})$

4.2 Speed

For speed comparisons, we employ the iteration period as a measure. The iteration period is simply the ratio of the clock period of the architecture to the number of outputs produced in each clock cycle [3]. The iteration period and the clock period for different architectures are shown in Table IV, where T_c is the time required to compute one serial output.

It can be seen that for each of the high-speed architectures, i.e. BLMS, PARLMS and PIPLMS, the iteration period is a fraction of T_c. However, hardware complexity limits the achievable speed for PARLMS and BLMS. Since the only hardware overhead for PIPLMS are the latches, the increase in hardware complexity would not limit the level of pipelining M.

5 Simulation Results

In this section, we present simulation results to verify the theoretical expressions for adaptation speed and adaptation accuracy for PIPLMS (section 3).

5.1 Experiment A

In this experiment, we attempt to identify a system composed of two delay elements (Fig.3). The noise variances are $\sigma_1^2 = 1$ and $\sigma_2^2 = 0.5$, and the filter order is $N = 4$. The mean squared error expression (eq.(29), [6]), which can be easily derived, is averaged over 32 independent trials and 400 iterations. This experiment is conducted for $\mu = 0.0234$ (experiment A1) and $\mu = 0.0468$ (experiment A2). The results of experiments A1 and A2 are tabulated in Table V. It can be seen (Table V) that the time-constants $\tau_{k,J}$ of BLMS and PIPLMS are nearly the same and that these are greater by a factor of two than the corresponding SLMS time-constants. In this experiment, as was discussed in section 3.2, the convergence time in seconds $t_{k,J}$ for PIPLMS would be one-half and one-fourth of the corresponding values for SLMS and BLMS respectively.

TABLE V. Experiment A.

	Exp.	μ	$\tau_{k,J}$		Misadjustment (%)	
			measured	theoretical	measured	theoretical
SLMS	A1	0.0234	24.0	21.37	4.68	4.68
	A2	0.0468	12.0	10.68	10.45	9.36
BLMS	A1	0.0234	43.0	42.74	2.57	2.34
L = 2	A2	0.0468	23.0	21.37	4.65	4.68
PARLMS	A1	0.0234	25.0	25.0	4.92	4.68
$M_B = 2$	A2	0.0468	25.0	25.0	10.45	9.36
$N_1 = 12$						
PIPLMS	A1	0.0234	44.0	42.74	5.08	4.68
$M_1 = 2$	A2	0.0468	22.0	21.37	10.96	9.36
$M_2 = 2$						

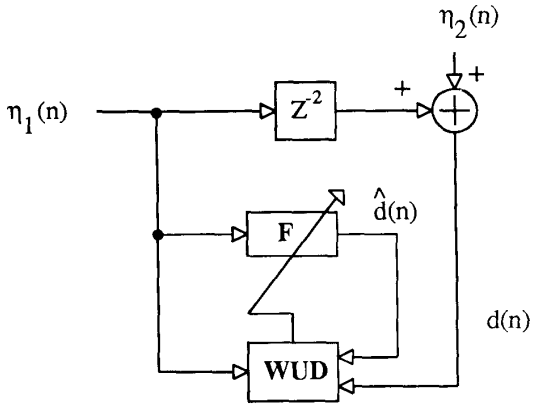

Figure 3: System identification example.

The misadjustment of BLMS is the lowest as expected. It is interesting to note that in this experiment the misadjustment ($\mathcal{M}_{1,PIP}$) for PIPLMS is the same as that of SLMS inspite of having $L_c = 0$. The plots (for experiment A1) depicting the variation of the mean-squared error of PIPLMS and SLMS, with the number of iterations, are shown in Fig.4.

5.2 Experiment B

The linear predictor is used to demonstrate the effect of the variable latency L_c on the misadjustment of the PIPLMS. In this experiment, we perform 1-step forward prediction of a second-order autoregressive (AR) process, whose generating filter has poles at $0.4875 \pm j0.8440$. The variance of the white noise at the input to the generating filter was chosen to be 0.0731 so as to get an output variance of 1. Plots of the mean-squared error for PIPLMS with $L_c = 0$ and $L_c = 2$ along with the corresponding plot for SLMS are shown in Fig.5, where $M_1 = M_{20} = 2$. As $M_1 = 2$, the misadjustment for $L_c = 2$ is the same as $\mathcal{M}_S$. On the other hand for $L_c = 0$, $J_{min,PIP}$ can be calculated to be equal to 0.1427, which results in α being equal to 1.95. When the measured value of 6.29%

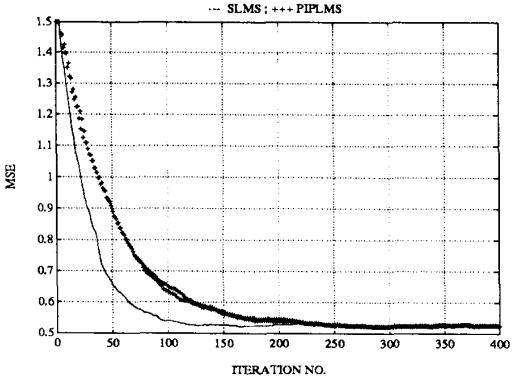

Figure 4: Experiment A : MSE plots for PIPLMS and SLMS.

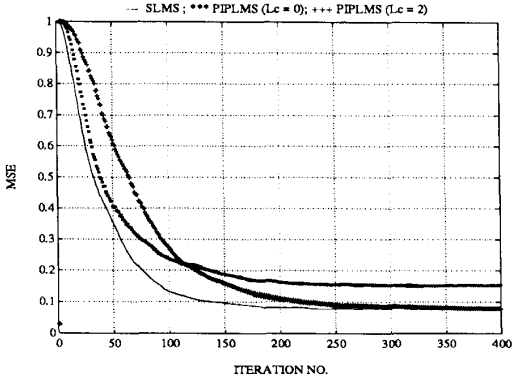

Figure 5: Experiment B : MSE plots for PIPLMS and SLMS.

for $\mathcal{M}_S$ is substituted along with α into the expression for $\mathcal{M}_{1,PIP}$ (Table II), we get a misadjustment of 107.48%. This compares well with the measured value of 110.26% for $\mathcal{M}_{1,PIP}$. Thus, the validity of the analytical expression for $\mathcal{M}_{1,PIP}$ is verified. As expected, the measured value of $\mathcal{M}_{2,PIP}$ was found to be close to $\mathcal{M}_S$. Hence, we may conclude that output latency can be traded-off with the misadjustment in case of PIPLMS.

6 Summary and Conclusions

A novel pipelined LMS architecture (PIPLMS) has been introduced. A stochastic form of look-ahead prediction resulted in enormous hardware savings. In particular, the increase in the hardware requirements are restricted to only pipeline latches. A unique feature of PIPLMS is its ability to trade-off the output latency for the adaptation accuracy.

Current work is being directed towards design of inherently concurrent algorithms for differential pulse code modulation (DPCM), adaptive decision feedback equalizers, and adaptive lattice filters.

References

[1] K.K. Parhi, "Algorithm transformation techniques for concurrent processors", *Proceedings of the IEEE*, vol. 77, pp. 1879-1895, Dec. 1989.

[2] K.K. Parhi and D.G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters - Part I : Pipelining using scattered look-ahead and decomposition", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 37, pp. 1099-1117, July 1989.

[3] K.K. Parhi and D.G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters - Part II : Pipelined incremental block filtering", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 37, pp. 1118-1134, July 1989.

[4] K.K. Parhi and D.G. Messerschmitt, "Concurrent cellular VLSI adaptive filter architectures", *IEEE Trans. on Circuits and Systems*, vol. 34, pp. 1141-1151, Oct. 1987.

[5] T. H.-Y. Meng and D.G. Messerschmitt, "Arbitrarily high sampling rate adaptive filters", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 35, pp. 455-470, April 1987.

[6] G.A. Clark, S.K. Mitra and S.R. Parker, "Block implementation of adaptive digital filters", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 29, pp. 744-752, June 1981.

[7] J.M. Cioffi, "The block-processing FTF adaptive algorithm", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 34, pp. 77-90, Feb. 1986.

[8] A. Gatherer and T. H.-Y. Meng, "High sampling rate adaptive decision feedback equalizer", *Proc. ICASSP '90*, pp. 909-912.