IEICE TRANS. FUNDAMENTALS, VOL. E81–A, NO. 8 AUGUST 1998

PAPER Special Section on Digital Signal Processing

Pipelined Architecture of the LMS Adaptive Digital Filter with the Minimum Output Latency

Akio HARADA†, *Student Member*, Kiyoshi NISHIKAWA†, *and* Hitoshi KIYA†, *Members*

SUMMARY In this paper, we propose two new pipelined adaptive digital filter architectures. The architectures are based on an equivalent expression of the least mean square (LMS) algorithm. It is shown that one of the proposed architectures achieves the minimum output latency, or zero without affecting the convergence characteristics. We also show that, by increasing the output latency be one, the other architecture can be obtained which has a shorter critical path.

key words: *pipelining, LMS algorithm, output latency, adaptive digital filter*

1. Introduction

In this paper, we propose a pipelined adaptive digital filter (ADF) with the minimum output latency. We also show that the architecture of a proposed filter has flexibility, so that it can be configured according to the required characteristics, namely, throughput or output latency.

ADFs have been used in a wide variety of applications including acoustic echo cancelers. In recent years, ADFs are expected to play an important role in the processing of wider-band signals or in mobile communications. In those applications, ADFs are required to process signals at very high speed or at low power consumption. The pipelined processing of ADF is considered as one of possible solutions for those requirements.

The pipelined processing based on the gradient type algorithm gathers much attention because of their simple structures [1]–[3]. The conventional architectures are based on the delayed LMS (DLMS) algorithm [4], [5], which is known as an algorithm that can be used for pipelining ADFs. By retiming the delays existing in the error feedback loop, the pipelined implementation of the DLMS ADF is achieved. Although the high throughput can be realized independently of the filter length, the DLMS algorithm introduces two disadvantages: (1) the convergence characteristic becomes worse in proportion to the filter length and, (2) the output latency equal to the filter length is produced.

To improve the convergence characteristic without affecting the throughput characteristic, modified algorithms of the DLMS were proposed [2], [3]. Based on these algorithms, we can implement pipelined ADFs which have the same rate of convergence as the LMS ADF. However even if we use these algorithms, we cannot solve the problem that a pipelined ADF produces the output latency equals to the filter length.

In this paper,we propose a new pipelined ADF which produces a constant and the minimum output latency. This advantage is achieved because the proposed filter is based on the LMS algorithm. The pipelined implementation of the LMS ADF has been considered impossible. However, we show that the pipelined implementation of the LMS ADF can be achieved using an equivalent expression of the LMS described in this paper. Although the proposed architecture can achieve the minimum latency, The critical path becomes slightly longer than the conventional methods. Therefore we also propose another architecture whose critical path is almost equal to the conventional methods with increasing one output latency.

This paper is organized as follows. In Sect. 2, a review of the conventional methods based on the DLMS algorithm is given. In Sect. 3, the algorithm based on the equivalent transformation of the LMS algorithm is proposed. Then, in Sect. 4, we propose the architectures based on the derived algorithm. Conclusions are given in Sect. 5.

2. Conventional Pipelined Adaptive Filters

In this section, we describe the conventional pipelined ADFs and their problems.

2.1 Pipelined Processing Based on the DLMS Algorithm

The filter update-formula of the DLMS algorithm is given as [4], [5]

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu e(n-D_d)\mathbf{U}(n-D_d) \quad (1)$$

$$e(n-D_d) = d(n-D_d) - \mathbf{W}^T(n-D_d)\mathbf{U}(n-D_d), \quad (2)$$

where the vector $\mathbf{W}(n)$ and $\mathbf{U}(n)$ are the N adaptive filter coefficients and the N input samples at time n, and they are given as

$$\mathbf{W}(n) = [w_0(n), w_1(n), \cdots, w_{N-1}(n)]^T \quad (3)$$

Manuscript received December 10, 1997.
Manuscript revised March 5, 1998.

†The authors are with the Department of Electrical Engineering, Graduate School of Tokyo Metropolitan University, Hachioji-shi, 192–0397 Japan.

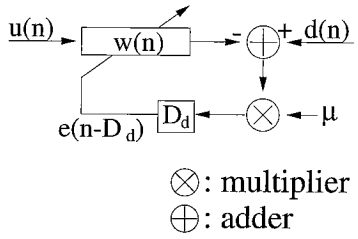


Fig. 1 Block diagram of the DLMS ADF.

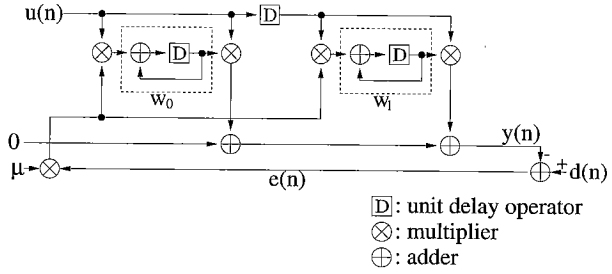


Fig. 2 Signal flow graph for the LMS ADF ($N = 2$).

$$\mathbf{U}(n) = [u(n), u(n-1), \cdots, u(n-N+1)]^T \quad (4)$$

respectively. $d(n)$ and $e(n)$ are the desired and the error signals at time n, and μ is the step size. In (1), D_d is the number of delays inserted into the error feedback path. The block diagram of the DLMS ADF is shown in Fig. 1.

The pipelined implementations [6]–[8] of the DLMS ADF are achieved by replacing the delays D_d using the retiming technique [9], so that a shorter critical path can be achieved as D_d increases. At the same time, however, increasing of D_d narrows the selectable range for μ, namely, the convergence characteristic becomes poor as D_d increases [4], [5].

Besides, there is another problem that the output latency proportional to the filter length will be produced. Let us show this fact using an example under the condition $N = 2$. When the LMS algorithm is used, the output $y(n)$ corresponding to the current input $u(n)$ is generated at the same time n as shown in Fig. 2. On the other hand, when the DLMS algorithm is used, the output is delayed in each processing module by one clock cycle as shown in Fig. 3. In this case, the output is delayed for two clock cycles. This time lag between an input and its corresponding output is called the output latency [1]. In the conventional architectures [6], [7], the output latency becomes N, the filter length. Therefore as the filter length N increases, the filter produces longer output latency.

2.2 Modified Algorithms of the DLMS

Several algorithms [2], [3] were proposed to improve the convergence characteristic of the DLMS ADF. They are based on the expression for converting the DLMS into

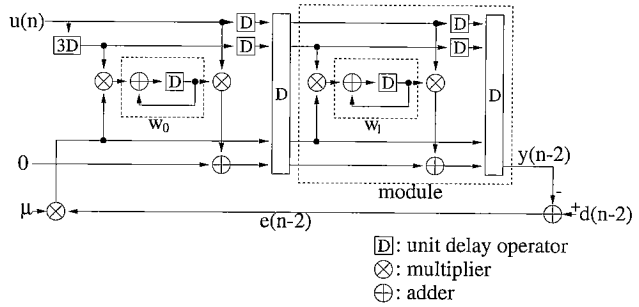


Fig. 3 Signal flow graph for the DLMS ADF ($N = 2$).

the LMS derived in [10]. By adding the difference between the LMS and the DLMS as a correction term to the error signal, the convergence characteristic can be improved without sacrificing the throughput. In mathematical forms, the difference $\Lambda(n)$ is given as

$$\begin{aligned}\Lambda(n) &= e_{(DLMS)}(n-D_d) - e_{(LMS)}(n-D_d) \\ &= (\mathbf{W}^T(n) - \mathbf{W}^T(n-D_d))\mathbf{U}(n-D_d), \end{aligned} \quad (5)$$

where $e_{(DLMS)}(n-D_d)$ is equal to (2) and

$$e_{(LMS)}(n-D_d) = d(n-D_d) - \mathbf{W}^T(n)\mathbf{U}(n-D_d). \quad (6)$$

The convergence characteristic can be improved by subtracting $\Lambda(n)$ from $e_{(DLMS)}(n)$.

The problem of producing the output latency however cannot be solved even if we use these modified algorithms.

3. Proposed Algorithm

Here, we propose an algorithm which can be pipelined. The proposed algorithm is derived from an equivalent transformation of the LMS algorithm.

3.1 Equivalent Transformation of the LMS Algorithm

First, we describe an equivalent transformation of the LMS algorithm which is necessary for deriving the proposed method. The filter update-formula of the LMS algorithm is given as [1]

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu e(n)\mathbf{U}(n) \quad (7)$$

$$e(n) = d(n) - \mathbf{W}^T(n)\mathbf{U}(n). \quad (8)$$

Let us consider a one-step previous relation of (7):

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \mu e(n-1)\mathbf{U}(n-1). \quad (9)$$

By substituting (9) into (8), we obtain

$$\begin{aligned} e(n) = d(n) &- \mu e(n-1)\mathbf{U}^T(n-1)\mathbf{U}(n) \\ &- \mathbf{W}^T(n-1)\mathbf{U}(n). \end{aligned} \quad (10)$$

This operation is called *one-step look-ahead transformation* of (8) [1]. By repeating this operation n times (*n-step look-ahead transformation*), we obtain

$$e(n) = d(n) - y(n), \tag{11}$$

where $y(n)$ is given as

$$y(n) = \sum_{k=1}^{n} \mu e(n-k)\mathbf{U}^T(n-k)\mathbf{U}(n). \tag{12}$$

Note that we assumed the initial condition $\mathbf{W}(0) = \mathbf{0}$.

By doing the *n-step look-ahead transformation*, the tap weight vector $\mathbf{W}(n)$ can be expressed as

$$\mathbf{W}(n) = \sum_{k=1}^{n} \mu e(n-k)\mathbf{U}(n-k). \tag{13}$$

The update-formula of the transformed tap weight vector is given as

$$\begin{aligned}\mathbf{W}(n+1) &= \sum_{k=1}^{n+1} \mu e(n+1-k)\mathbf{U}(n+1-k) \\ &= \sum_{k=1}^{n} \mu e(n-k)\mathbf{U}(n-k) \\ &\quad + \mu e(n)\mathbf{U}(n) \\ &= \mathbf{W}(n) + \mu e(n)\mathbf{U}(n). \end{aligned} \tag{14}$$

By comparing (13), (14) and (8), we find that the update-formula does not change. Therefore, the selectable range for the step size is identical to that of the LMS.

By expanding (12), $y(n)$ is expressed as

$$y(n) = \sum_{k=1}^{n}\sum_{m=0}^{N-1} \mu e(n-k)u(n-k-m)u(n-m). \tag{15}$$

In the following, our objective is to show that $y(n)$ can be calculated in a pipelined fashion. This is because the LMS ADF would be pipelined if the calculation of $y(n)$ can be pipelined. For that purpose, we rewrite (15) as

$$\begin{aligned} y(n) &= \sum_{i=0}^{N-1} c_i(n-i) \\ &= \sum_{i=0}^{N-1} \{h_i(n-i)u(n-i) \\ &\quad + r_i(n-i)\mu e(n-i-1)\} \end{aligned} \tag{16}$$

where

$$h_i(n) = \sum_{j=1}^{n-i} \mu e(n-j)u(n-j-i) \tag{17a}$$

$$r_i(n) = \sum_{j=1}^{N-i-1} u(n-j-i-1)u(n-j) \tag{17b}$$

$$r_{N-1}(n) = 0. \tag{17c}$$

We note that, as will be shown in the following, $h_i(n)$ and $r_i(n)$ are used as basic elements in the proposed method instead of $w_i(n)$.

3.2 Derivation of (16)

In the previous section we showed an equivalent transformation of the LMS algorithm. Here, we give a brief derivation of (16).

Let us consider the LMS ADF with 3 coefficients. The output $y(n)$ is given as

$$\begin{aligned} y(n) &= \mathbf{W}^T(n)\mathbf{U}(n) \\ &= w_0(n)u(n) + w_1(n)u(n-1) \\ &\quad + w_2(n)u(n-2). \end{aligned} \tag{18}$$

By applying *n-step look-ahead transformation* to (18), we obtain

$$\begin{aligned} y(n) &= \sum_{i=1}^{n} \mu e(n-i)u(n-i)u(n) \\ &\quad + \sum_{i=1}^{n} \mu e(n-i)u(n-i-1)u(n-1) \\ &\quad + \sum_{i=1}^{n} \mu e(n-i)u(n-i-2)u(n-2). \end{aligned} \tag{19}$$

Note that the initial condition $\mathbf{W}(0) = \mathbf{0}$.

From (19), we find that the output $y(n)$ is given as a sum of the products of $\{u(n-j) : 0 \leqq j\}$ and $\{e(n-k) : 1 \leqq k\}$. The equation (19) can be divided into two parts: the first part consists of the terms which cannot be calculated before the arrival of $u(n)$ or $e(n-1)$ at time n, and the second one consists of those which can be calculated without $u(n)$ and $e(n-1)$. Namely, the first part is given as

$$\begin{aligned} c_0(n) &= \sum_{i=1}^{n} \mu e(n-i)u(n-i)u(n) \\ &\quad + \sum_{j=1}^{2} \mu e(n-1)u(n-j-1)u(n-j), \end{aligned} \tag{20}$$

and the second part is given as

$$\begin{aligned} c_0'(n) &= y(n) - c_0(n) \\ &= \sum_{i=2}^{n} \mu e(n-i)u(n-i-1)u(n-1) \\ &\quad + \sum_{i=2}^{n} \mu e(n-i)u(n-i-2)u(n-2). \end{aligned} \tag{21}$$

The second part $c_0'(n)$ can be calculated at time $n-1$, because $c_0'(n)$ does not require the knowledge of $u(n)$ and $e(n-1)$.

In the similar way, $c_0'(n)$ can be divided into two parts $c_1(n-1)$ and $c_2(n-2)$ defined as

$$\begin{aligned} c_1(n-1) &= \sum_{i=2}^{n} \mu e(n-i)u(n-i-1)u(n-1) \\ &\quad + \mu e(n-2)u(n-4)u(n-2) \end{aligned} \tag{22}$$

$$c_2(n-2) = \sum_{i=3}^{n} \mu e(n-i)u(n-i-2)u(n-2). \quad (23)$$

$c_1(n-1)$ can be calculated using the information available at time $n-1$. Using (20)–(23), $y(n)$ can be expressed as

$$y(n) = \sum_{i=0}^{2} c_i(n-i), \quad (24)$$

and Fig. 4 shows the calculation of $y(n)$ and an explanation of this figure is given in the following.

3.3 Pipelining of Calculation of $y(n)$

Let us show that calculation of $y(n)$ can be pipelined. First, we define $c_i(n)$ as

$$c_i(n) = h_i(n)u(n) + r_i(n)\mu e(n-1). \quad (25)$$

Then, (16) is rewritten as

$$y(n) = \sum_{i=0}^{N-1} c_i(n-i). \quad (26)$$

By defining $y_i(n)$ as

$$y_i(n) = y_{i+1}(n-1) + c_i(n), \quad (27)$$

we have the following relation:

$$y(n) = y_0(n). \quad (28)$$

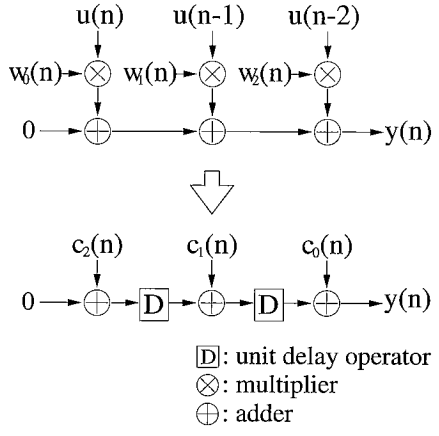


Fig. 4 Pipelined implementation of the LMS ADF by using the proposed algorithm.

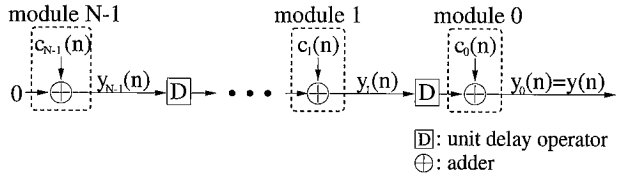


Fig. 5 Pipelined implementation of the LMS ADF by connecting N processing modules.

The form of the Eq. (27) suggests that the calculation of $y_i(n)$ can be regarded as a processing-module. Therefore by connecting N modules in series as shown in Fig. 5, we can implement the LMS ADF in a pipelined fashion.

Each processing module is composed of $h_i(n)$ and $r_i(n)$ instead of the filter coefficient $w_i(n)$. This is the major difference from the conventional methods and it is the key concept of the proposed method.

Note that we can calculate $h_i(n)$ and $r_i(n)$ recursively as

$$h_i(n) = h_i(n-1) + \mu e(n-1)u(n-i-1) \quad (29a)$$

$$r_i(n) = r_i(n-1) + u(n-2-i)u(n-1) - u(n-1-N)u(n-(N-i)). \quad (29b)$$

The conventional pipelined implementations are achieved by inserting the delays between the filter coefficients $w_i(n)$. However in this strategy, all the coefficients $w_i(n)$ cannot be calculated before the arrival of the signals $u(n)$ or $e(n-1)$ and hence, we cannot insert the delays between the filter coefficients. We can achieve the pipelined implementation of the LMS ADF by using $h_i(n)$ and $r_i(n)$ instead of using $w_i(n)$.

3.4 Processing Procedure at Each Time

Here we summarize the procedure at each time n.

First, we define the initial conditions as

$$h_i(0) = r_i(0) = 0. \quad (30)$$

Then, the following procedure is proceeded at time n:

1. Using (29), $h_i(n)$ and $r_i(n)$ are calculated in each processing modules.
2. Using (27), the output $y_i(n)$ is calculated in each processing modules.
3. From (28), the output of ADF $y(n)$ is obtained as the output of 0-th processing module $y_0(n)$.
4. The error $e(n)$ is calculated using (11).

4. Proposed Architecture

Here we propose two pipelined architectures of the equivalent transformation of the LMS algorithm described in Sect. 3. As will be shown in the following, there exists flexibility in constructing each module although the basic equations are given in (27)–(29). We show two different architectures as examples. The first one enables us to produce zero output latency, and the other has a shorter critical path and a constant output latency.

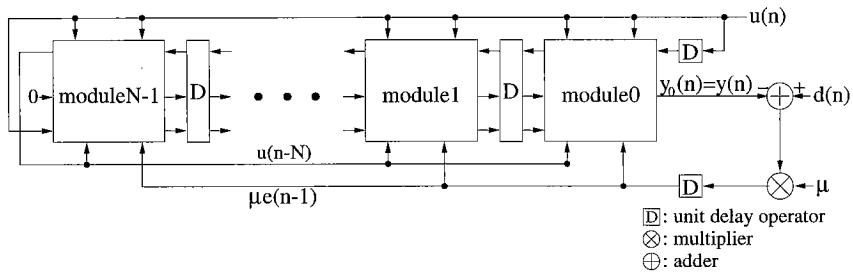


Fig. 6 Architecture with zero output latency (Arc1).

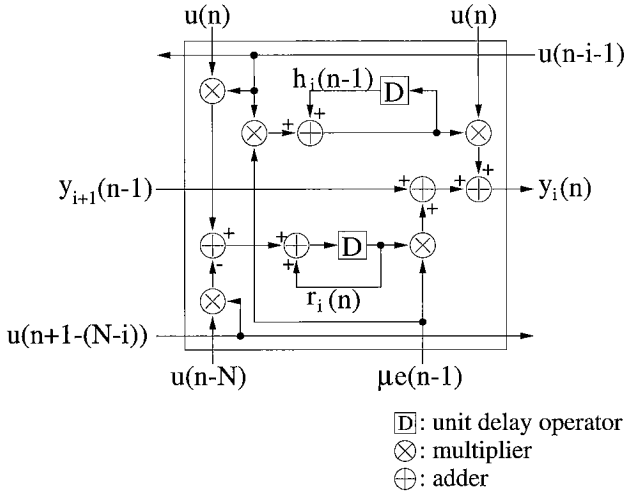


Fig. 7 Structure of the i-th processing module for architecture with zero output latency (Arc1).

4.1 Architecture with Zero Output Latency (Arc1)

First we show an architecture producing zero output latency in Fig. 6, and the structure of the i-th processing module in Fig. 7. In this architecture, the output latency is always 0, because there is no delay between the input $u(n)$ and the output $y(n) = y_0(n)$.

Let us consider the critical path of this structure. In 0-th processing module, we use the equation given as

$$\begin{aligned} y_0(n) &= d(n) - y_1(n-1) - c_0(n) \\ &= e(n), \end{aligned} \tag{31}$$

instead of (27). The critical path t_{cpArc1} of Arc1 is given as

$$t_{cpArc1} = 3t_{mult} + 2t_{add}, \tag{32}$$

where t_{mult} and t_{add} are the times required for one multiplication and one addition respectively.

Besides, we note that by selecting the value for the step size μ as a power of two, we will regard the calculation time at the error feedback path as the time required for one addition. Under this assumption, the critical path t_{cpArc1} of Arc1 is given as

$$t_{cpArc1} = 2t_{mult} + 3t_{add}, \tag{33}$$

4.2 Architecture with Shorter Critical Path (Arc2)

In Arc1, the output latency becomes 0, however its critical path is longer than that of conventional architectures, as will be shown later. Here we give an architecture that has shorter critical path than Arc1.

From Fig. 7, we notice that two multiplications exist in the critical path of Arc1. Accordingly, let us consider reducing one of the multiplications in the critical path by inserting a delay between them.

First we transform (25) as

$$\begin{aligned} c_i(n) &= \mu e(n-1)u(n-i-1)u(n) \\ &\quad + h_i(n-1)u(n) + r_i(n)\mu e(n-1), \end{aligned} \tag{34}$$

and insert a delay into each the input $u(n)$ and the desired signal $d(n)$. Then we apply the retiming technique to these delays. As a result, the architecture shown in Fig. 8 is obtained and the structure of i-th processing module is shown in Fig. 9.

By using (31) instead of (27) in 0-th processing-module as is done in Arc1. The critical path t_{cpArc2} of Arc2 is given as

$$t_{cpArc2} = 2t_{mult} + 2t_{add}. \tag{35}$$

Besides, by assuming the calculation time in the critical path can be regarded as same as the time required for one addition in a similar way as Arc1, the critical path t_{cpArc2} of Arc2 is given as

$$t_{cpArc2} = t_{mult} + 3t_{add}. \tag{36}$$

Note that the output latency of this architecture is always 1.

4.3 Comparison of Each Architecture

Let us compare the proposed architectures with the conventional ones in terms of the critical path, the produced output latency and the number of operators. We show the comparison in Table 1.

We can see from the table, Arc1 has zero output latency, although its critical path is twice as long as the DLMS ADF and the LDLMS ADF [3]. On the other hand, Arc2 can achieve the critical path that is almost equal to that of the conventional architectures with producing 1 output latency and slight increase of

Table 1 Comparison of each architecture where N is filter length, t_{mult} and t_{add} are the times required for one multiplication and one addition, δ is the parameter for adjusting the convergence characteristic ($0 \leqq \delta \leqq D_d - 3$) used in LDLMS[3].

Architecture	Critical path	Output latency	Number of operators		
			Multiplier	Adder	Delay
LMS[11]	$2t_{mult} + (N+2)t_{add}$	0	$2N+1$	$2N+1$	$2N-1$
DLMS[6]	$t_{mult} + 2t_{add}$	N	$2N+1$	$2N+1$	$8N-2$
LDLMS[3]	$t_{mult} + 2t_{add}$	N	$2(N+\delta)+1$	$(\delta+2)N+\delta+1$	$(11+\delta)N+2\delta-7$
Zhu et al.[2]	$t_{mult} + 2t_{add}$	N	$5N+1$	$5N+1$	$6N$
Arc1	$2t_{mult} + 3t_{add}$	0	$5N-2$	$5N-2$	$5N-2$
Arc2	$t_{mult} + 3t_{add}$	1	$6N-1$	$6N-2$	$7N-1$

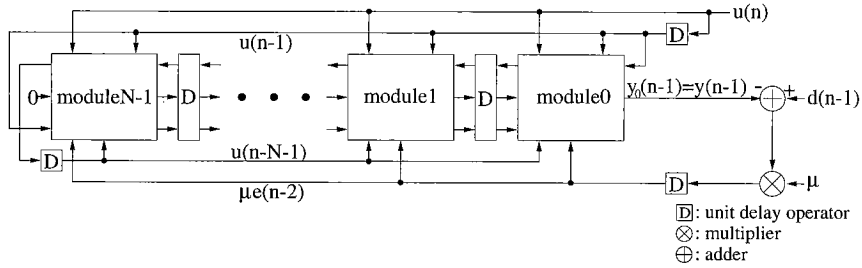


Fig. 8 Architecture with a shorter critical path (Arc2).

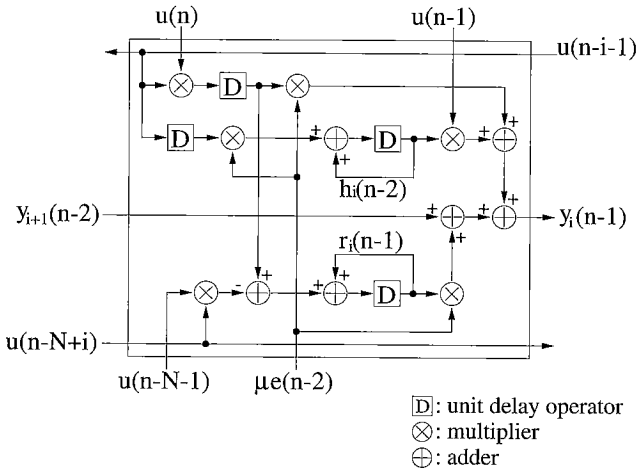


Fig. 9 Structure of i-th processing module for architecture with shorter critical path (Arc2).

required operations. Note that the conventional architectures produce the output latency proportional to the filter length N.

4.4 Simulation

Finally we show some simulation results to demonstrate the effectiveness of the proposed architecture. We simulated system identification problems with an unknown system of the filter length 200 on 3 conditions:

(1) The input signal was a zero-mean Gaussian random process with a variance of 1 as a white process, the unknown system was stationary.

(2) The input signal was an AR process of order 1 as a colored process, the unknown system was stationary.

(3) The input signal was the same signal as (1), the unknown system was nonstationary, namely, when the number of iterations became 2000, the unknown system was changed.

The observation noise signal was zero-mean Gaussian random process with a variance of 0.0001 which is independent of the input. The results are ensemble averages of ten independent processes.

First, we show the results under the condition (1). In this case, we compared the proposed architecture (Arc1), the LMS ADF, the DLMS ADF ($D_d = N$) and LDLMS ADF ($D_d = N, \delta = D_d - 3$). We set the filter length N of ADFs as $N = 200$. The step size μ was set as $\mu = 0.001953125 = 2^{-9}$ for the DLMS ADF, and $\mu = 0.00390625 = 2^{-8}$ for the others. The simulation results are shown in Fig. 10. From Fig. 10, we can see that the proposed architecture realizes the identical convergence characteristic as that of the LMS ADF. Besides, the proposed architecture has no output latency, therefore, it is superior than DLMS ADF and LDLMS ADF.

Then, we show the results under the conditions (2) and (3). In these cases, we compared the proposed architecture (Arc1) with the LMS ADF. We set the filter length N of ADFs as $N = 200$. The step size μ was set as $\mu = 0.0000152587890625 = 2^{-16}$ for both ADF under condition (2), $\mu = 0.001953125 = 2^{-9}$ for both ADF under condition (3). The simulation results are shown in Figs. 11 and 12. From the figures, we can see that the proposed architecture can realize identical con-

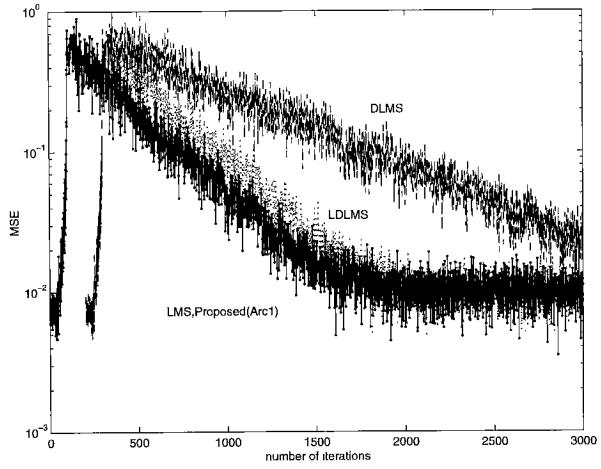


Fig. 10 Convergence characteristics of the proposed architecture (Arc1), the LMS ADF, the DLMS ADF ($D_d = N$) and the LDLMS ADF ($D_d = N, \delta = D_d - 3$) when the input signal was the white process.

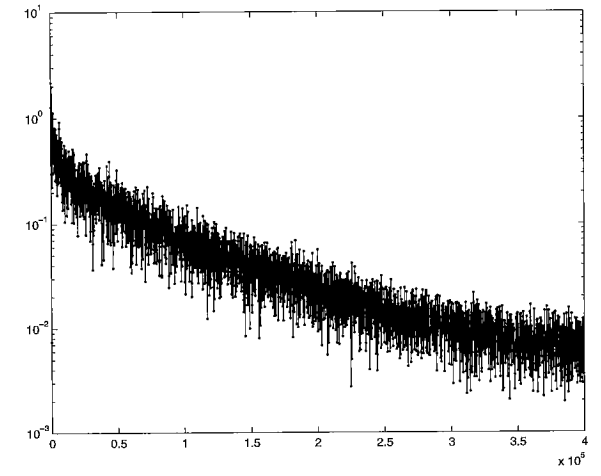


Fig. 11 Convergence characteristics of the proposed architecture (Arc1) and the LMS ADF when the input signal was the colored process.

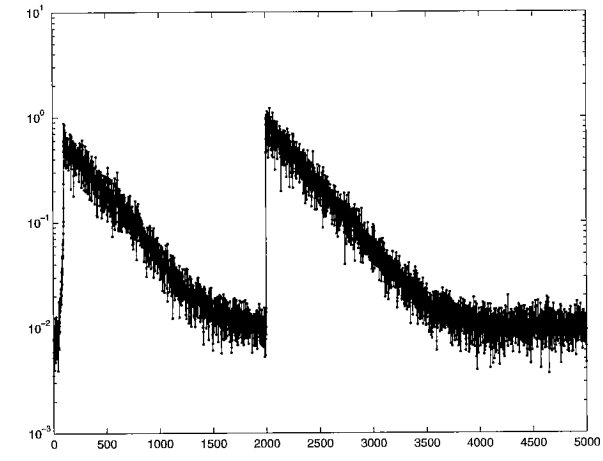


Fig. 12 Convergence characteristics of the proposed architecture (Arc1) and the LMS ADF when the unknown system was unstationary.

vergence characteristic as the LMS ADF as in the white input case.

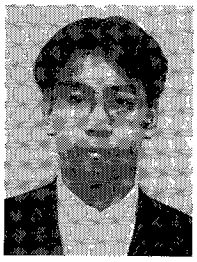
5. Conclusion

In this paper, we proposed the pipelined architectures of the LMS ADF with the minimum output latency. The proposed architectures are based on an equivalent expression of the LMS algorithm. We showed that by altering the filter architecture, the pipelined implementation is achieved at the high throughput while keeping the convergence characteristic and small output latency. Finally, we showed that the proposed architecture could achieve pipelined processing with the minimum output latency and that the identical convergence characteristic as that of the LMS ADF through computer simulations.

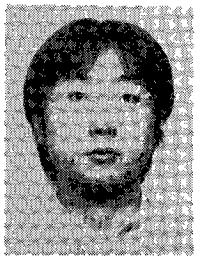
Consideration on the effect of the finite precision calculations on the algorithm is considered as a future work.

References

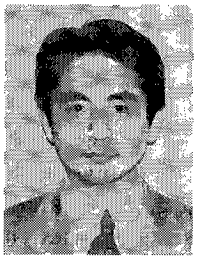
[1] N.R. Shanbhag and K.K. Parhi, "Pipelined Adaptive Digital Filters," Kluwer Academic Publishers, 1994.
[2] Q. Zhu, S.C. Douglas, and K.F. Smith, "A pipelined architecture for LMS adaptive FIR filters without adaptation delay," Proc. IEEE ICASSP'97, III, pp.1933–1936, April 1997.
[3] K. Matubara, K. Nishikawa, and H. Kiya, "A new pipelined architecture of the LMS algorithm without degradation of convergence characteristics," Proc. IEEE ICASSP'97, III, pp.4125–4128, April 1997.
[4] 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, Sept. 1989.
[5] G. Long, F. Ling, and J.G. Proakis, "Corrections to "the LMS algorithm with delayed coefficient adaptation"," IEEE Trans. Signal Processing, vol.40, no.1, pp.230–232, Jan. 1992.
[6] H. Herzberg, R. Haimi-Cohen, and Y. Be'ery, "A systolic array realization of an LMS adaptive filter and the effects of delayed adaptation," IEEE Trans. Signal Processing, vol.40, no.11, pp.2799–2803, Nov. 1992.
[7] M.D. Meyer and D.P. Agrawal, "A high sampling rate delayed LMS filter architecture," IEEE Trans. Circuits & Syst. II, vol.40, no.11, pp.727–729, Nov. 1993.
[8] K. Matubara, K. Nishikawa, and H. Kiya, "Pipelined adaptive filters based on delayed LMS algorithm," IEICE Trans (in Japanese), J79-A, pp.1050–1057, May 1996.
[9] C.E. Leiserson, F. Rose, and J. Saxe, "Optimizing synchronous circuitry by retiming," Proc. of the Third Caltech Conference on VLSI, pp.87–116, March 1983.
[10] R.D. Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm," IEEE Signal Processing Letters, vol.2, no.12, p.223, Dec. 1995.
[11] B. Widrow and S.D. Stearms, "Adaptive Signal Processing," Prentice-Hall, Inc., 1985.



Akio Harada was born in Tokyo, Japan, on October 31, 1974. He received the B.E. degrees in electronics and information engineering from Tokyo Metropolitan University in 1997. He is currently a candidate for the M.E. degree at Tokyo Metropolitan University. His research interest is in adaptive signal processing.



Kiyoshi Nishikawa was born in Tokyo, Japan, on December 12, 1966. He received the B.E., the M.E., and the D.E. degrees in electrical engineering from Tokyo Metropolitan University in 1990, 1992 and 1996, respectively. From 1992 to 1993, he was at the Computer Systems Laboratory, Nippon Steel Corp. as a researcher. Since 1993, he has been with Tokyo Metropolitan University as a research associate. His research interest includes the adaptive signal processing. He is a member of IEEE SP, CAS, COM, and Computer Societies.



Hitoshi Kiya was born in Yamagata, Japan, on November 16, 1957. He received the B.E. and M.E. degrees in electrical engineering from Nagaoka University of Technology, Niigata, Japan, and the D.E. degree in electrical engineering from Tokyo Metropolitan University, Tokyo, Japan, in 1980, 1982, and 1987, respectively. In 1982, he joined Tokyo Metropolitan University, where he is currently an Associate Professor of Electrical Engineering, Graduate School of Engineering. He was a visiting researcher of the University of Sydney in Australia from Oct. 1995 to Mar. 1996. His research interests are in digital signal processing, multirate systems, adaptive filtering, image processing, and efficient algorithms for VLSI implementation. Dr. Kiya is a Member of the IEEE, the Image Electronics Engineers of Japan and the Institute of Television Engineers of Japan. He is a Associate Editor of IEEE Transactions on Signal Processing of IEEE.