A pipelined architecture for DLMS algorithm considering both hardware complexity and output latency

Tadaaki KIMIJIMA   *Kiyoshi NISHIKAWA*   *Hitoshi KIYA*
Dept. of Electronics Eng., Tokyo Metropolitan University
1-1 Minami-Osawa, Hachioji
Tokyo, 192-0397, JAPAN
Tel : +81-426-77-2745; Fax : +81-426-77-2756
e-mail: kiyoshi@eei.metro-u.ac.jp

ABSTRACT

In this paper we propose a new pipelined architecture for the DLMS algorithm which can be implemented with less than half an amount of calculation compared to the conventional architectures. Although the proposed architecture enables us to reduce the required calculation, it can achieve good convergence characteristics, a short latency and high throughput characteristics simultaneously.

1 Introduction

Researches of pipeline processing of gradient-type ADFs(adaptive digital filters) have been based on the DLMS(delayed least mean square) algorithm[1]-[3]. The DLMS is a modified version of the LMS suited for pipelined processing. It has D units of delay in its error feedback path; moving these delays and placing them at every tap of the filter, pipelining of the DLMS is accomplished[1]-[3]. However, the DLMS has a disadvantage that the convergence characteristics worsen as the amount of delay increases although it is required to increase the amount of delay; the finest grain pipelining is realized when D is set as the length of the ADF. Besides, the latency of the architecture increases in proportional to the filter length. We should consider these two issues when implementing ADFs using the pipeline technique.

To improve the convergence characteristics, several architectures were proposed[4]-[6] so far. These architectures are based on the equivalent transformation shown in [8]. Improvement of convergence is achieved by adding a correctional term to the DLMS, which transforms the DLMS into the LMS. Although the correctional term improves the convergence, it requires a large amount of calculations to be implemented. Besides, these architectures cannot reduce the output latency[6].

Recently an architecture based on the LMS was proposed[7] which achieves the identical convergence characteristics to that of the LMS without producing the output latency. Even if we use this architecture however the problem of increase of calculators will be remained since the equivalent expression of the LMS, which is introduced in [7] to achieve pipelining, requires a large amount of calculation.

In this paper we propose an efficient pipelined architecture for the DLMS algorithm which considers both hardware complexity and output latency. The proposed architecture enables us to achieve good convergence characteristics, a short latency and high throughput characteristics simultaneously with less than half an amount of calculation compared to the conventional architectures.

2 Conventional methods

First, we briefly describe the conventional methods, and point out their problems.

2.1 The DLMS algorithm

As a preparation, we briefly describe the DLMS algorithm[1]-[3]. The DLMS has D ($0 \le D \le N$) units of delay in the error feedback path. By moving these delays and placing them at every tap of the filter, pipelining of the DLMS is accomplished.

By assuming that the ADF is an FIR filter, whose impulse response is denoted by $w_k(n)$, the output signal $y(n)$ is given by

$$y(n) = \sum_{k=0}^{N-1} x(n-k)w_k(n), \tag{1}$$

where $x(n)$ and N are the input signal and the filter length respectively. The DLMS is described by

$$\mathbf{W}(n+1) = \mathbf{W}(n) + \mu e(n-D)\mathbf{x}(n-D) \tag{2}$$

$$e(n-D) = d(n-D) - \mathbf{W}^T(n-D)\mathbf{x}(n-D), \tag{3}$$

where $e(n\text{-}D)$ is the error signal of the DLMS, $d(n)$ is the desired signal, $\mathbf{W}(n)$ and $\mathbf{x}(n)$ are the filter coefficient vector and the input vector respectively, and they are given by

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

$$\mathbf{W}(n) = [w_0(n), w_1(n) \cdots w_{N-1}(n)]^T, \tag{5}$$

where $[\mathbf{x}]^T$ indicates transpose of a vector $\mathbf{x}$.

When the DLMS is used to pipeline ADFs, higher throughput can be obtained by increasing the amount of delay D. On the other hand, convergence characteristics become poor as the amount of delay increases. This degradation of the convergence is caused by the time lag between $\mathbf{W}(n)$ in Eq.(2) and $\mathbf{W}(n\text{-}D)$ in Eq.(3), and as the filter length increases, the convergence characteristics become poorer[4],[5].

2.2 Architectures based on the transformation shown in [8]

To improve the convergence characteristics, several architectures were proposed[4]-[6], which are based on the transformation of the DLMS into the LMS shown in [8]. They improve the convergence characteristics by adding a correctional term to the error signal of the DLMS.

Let us explain this point by using equations. By applying the transformation, the DLMS is modified as

$$\mathbf{W}(n+1)=\mathbf{W}(n)+\mu\varepsilon(n-D)\mathbf{x}(n-D) \quad (6)$$

$$\varepsilon(n-D)=d(n-D)-\mathbf{W}^T(n-D)\mathbf{x}(n-D)-\Lambda(n) \quad (7)$$

$$\Lambda(n)=\sum_{i=1}^{D-1}\mu e(n-D-i)\mathbf{x}^T(n-D-i)\mathbf{x}(n-D), \quad (8)$$

where $\Lambda(n)$ is the correctional term[4],[5]. Note that when $\Lambda(n)$=0, this algorithm is reduced to the DLMS. By adding a correctional term $\Lambda(n)$ to the error signal of the DLMS, we can cancel the influence of the time lag without reducing the inserted delays D[4]-[6].

Although these approach improve the convergence, a large amount of calculation is needed to implement those architectures: the amount of calculation needed to implement $\Lambda(n)$ depends on D as is shown in Eq.(8). However these delays are required to maintain the high throughput characteristics. Besides, these architectures cannot reduce the output latency[6].

2.3 An architecture based on the LMS

Independently from the architectures based on [8], an architecture based on the LMS was proposed[7]. In [7], it is shown that the LMS can be pipelined by using an equivalent expression of the LMS although it was considered to be impossible. This architecture has identical convergence characteristics and latency equal to that of the LMS. But it has the same problem as the architectures mentioned in **2.2**: it requires large amount of calculation to be implemented.

3 The proposed architecture

Here we describe an outline of the proposed architecture. We show that the proposed architecture enables us to achieve the following characteristics simultaneously: (1)High throughput characteristics same as that of the conventional architectures[4]-[6]; (2)A short latency; (3)Good convergence property equivalent as that of the LMS; (4)Composed with less than half an amount of calculators compared to conventional architectures[4]-[7].

3.1 Derivation of the proposed architecture

The main idea of the proposed architecture is to use binary-tree adder in calculating $y(n)$, insert the minimum amount of delay enough to achieve the same critical path (CP) as that of the conventional architectures[4]-[6]. Derivation of the proposed architecture is divided in two steps. In the following, we explain a method to decrease the critical path as the *Step1*, and to cancel the influence of the time lag as the *Step2*.

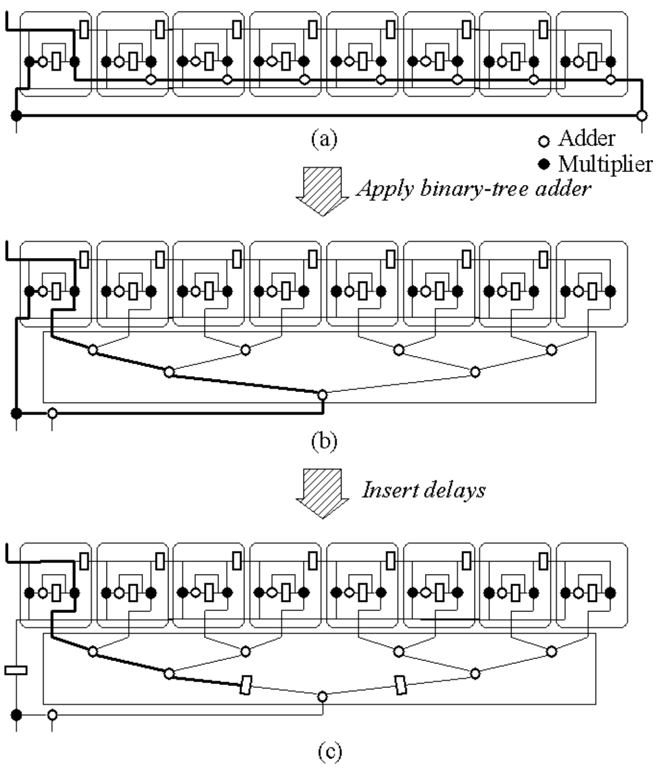

Fig.1 Modification of SFG of example of N=8taps
(a)an architecture of LMS
(b)after applying binary-tree adders
(c)after inserting the minimum amount of delay

Step 1. Decrease of the Critical path

Let us start considering the architecture of the LMS. The CP of the LMS is the path for calculating the output signal $y(n)$ and its length is $(N+1)t_{add}+3t_{mlt}$, where t_{add} and t_{mlt} are required time for an addition and a multiplication respectively. On the other hand, the shortest CP of the architectures proposed so far is $t_{add}+2t_{mlt}$[4]-[6].

We consider decreasing the CP of the LMS by using binary-tree adder. Fig.1(a) shows an 8taps example of an architecture of the LMS. In this figure, the CP is shown by the bold line. By applying binary-tree adder, we can decrease the CP from $(N+1)t_{add}+3t_{mlt}$ to $(\log_2 N+2)t_{add}+3t_{mlt}$ as is shown in Fig.1(b). Note that this CP is still longer than that of the conventional one. To make the CP equal to that of the shortest one, let us insert the minimum amount of delay. It can be achieved by inserting one delay per three calculators. Fig.1(c) shows an 8taps example of inserting the minimum amount of delay.

As a result of inserting delays, a time lag will be generated, whose length is given by

$$D'=\left\lceil\frac{\log_2 N}{3}\right\rceil+1, \quad (9)$$

where $\lceil x\rceil$ is the smallest integer larger than x, and the equations of the LMS[9] are changed as follows

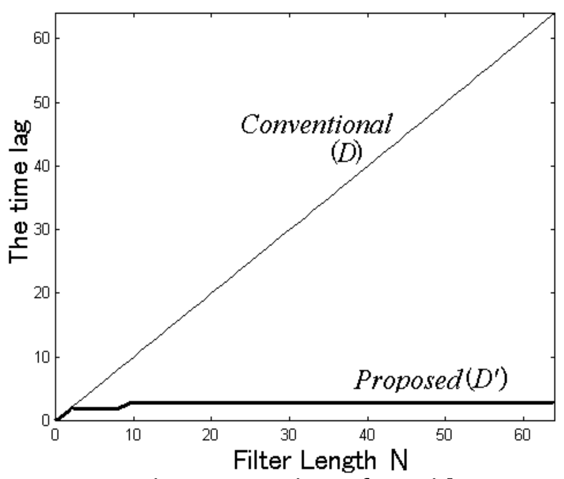

Fig.2 Comparison of *D'* with *D*

$$\mathbf{W}(n+1)=\mathbf{W}(n)+\mu e(n-D')\mathbf{x}(n-D') \quad (10)$$

$$e(n-D')=d(n-D')-\mathbf{W}^T(n-D')\mathbf{x}(n-D') \quad (11)$$

Note that Eqs.**(10)** and **(11)** are correspond to Eqs.(2) and (3) with $D'=D$. This shows that the time lag between $\mathbf{W}(n)$ in Eqs.**(10)** and **(11)** becomes *D'* instead of *D*. Fig.2 shows the comparison of *D'* with *D*. This figure clearly shows that the time lag of the proposed architecture *D'* become very small compared to that of the conventional architectures, *D*.

Since this time lag causes the degradation of the convergence, we need to cancel the influence of this time lag as the conventional architectures do.

Step 2. A method of canceling the influence of the time lag

In order to cancel the influence of this time lag, we add a correctional term $\Lambda'(n)$ to the error signal as the conventional architectures do[4]-[6]. In this case, $\Lambda'(n)$ is described by

$$\Lambda'(n)=\sum_{i=1}^{D'-1}\mu e(n-D'-i)\mathbf{x}^T(n-D'-i)\mathbf{x}(n-D') . \quad (12)$$

Note that the difference of the upper limit between Eqs.(8) and (12); it changed from *D* to *D'*. This difference contributes the reduction of the required amount of calculation. A comparison with the conventional architectures will be given in **4.2**.

3.2 Formula and Architecture

Finally the basic formula, on which the proposed architecture is constructed, is described by

$$\mathbf{W}(n+1)=\mathbf{W}(n)+\mu\varepsilon(n-D')\mathbf{x}(n-D') \quad (13)$$

$$\varepsilon(n-D')=d(n-D')-\mathbf{W}^T(n-D')\mathbf{x}(n-D')-\Lambda(n) \quad (14)$$

$$\Lambda'(n)=\sum_{i=1}^{D'-1}\mu e(n-D'-i)\mathbf{x}^T(n-D'-i)\mathbf{x}(n-D') . \quad (15)$$

Fig.4 shows the signal flow graph (SFG) of the proposed architecture. Fig.4(a) shows the whole architecture, and (b) shows the structure of *k*th module which corresponds to one tap of the ADF. The structure of $\Lambda'(n)$ is depicted in (c), and (d) shows that of binary-tree adder, where $y_k(n)$ is a part of output signal $y(n)$. The relation between $y(n)$ and $y_k(n)$ are given as

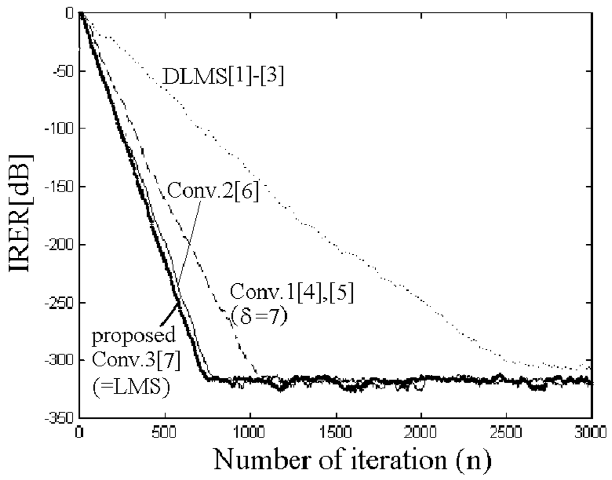

Fig.3 Learning Curves of the proposed algorithm

$$y(n)=\sum_{k=0}^{N-1}y_k(n) , \quad (16)$$

where

$$y_k(n)=x(n-k)w_k(n) . \quad (17)$$

4 Comparison of the proposed architecture with the conventional architecture

Here, we show the characteristics of the proposed architecture and compare them with the conventional ones. We show results of computer simulations, and compare characteristics of the proposed with the conventional ones.

4.1 Simulation results

To verify the validity of the proposed architecture, we show the results of computer simulation of system identification. The optimum system was a low-pass finite impulse response(FIR) filter of length 10. The order of the ADF was selected as the same size of the optimum system. The amount of delay *D* was 10 for the conventionals, and $D'=3$ for the proposed. The input signal $\mathbf{x}(n)$ was a white Gaussian process with mean 0 and variance 1. The step size parameter μ was selected to be the maximum value which guarantees the convergence of the mean-squared error (MSE)[9]. As the reference of comparison we used the impulse response error ratio (IRER).

Fig.3 shows the learning curves of the proposed architecture and the conventional ones, where Conv.1 is for the LDLMS[4],[5], Conv.2 is for the algorithm proposed in [6], and Conv.3 is for the algorithm proposed in [7]. Note that Conv.1 has a parameter δ and here we choose δ=7[4],[5]. From the figure we can see that the learning curves of the proposed architecture is equivalent to that of the LMS.

4.2 Comparison of the characteristics

Table.1 shows a comparison of the characteristics of the proposed architecture with the conventional ones. We can see, from this table, that the proposed architecture requires less than half an amount of calculations compared with the conventional architectures[4]-[7]. Besides, the latency of the proposed architecture is approximately same as the LMS.

Table.1 A comparison of the characteristics

where N is the filter length, t_{add} and t_{mlt} are required time for an addition and a multiplication respectively, $D'=\left\lceil \frac{\log_2 N}{3} \right\rceil+1$, Conv.1 is for the LDLMS[4],[5], Conv.2 is for the algorithm proposed in [6], and Conv.3 is for the algorithm proposed in [7].

	Critical path	Latency	Number of calculator		
			Adder	Multiplier	Delay
The LMS[9]	$(N+1)t_{add}+3t_{mlt}$	0	$2N+1$	$2N+1$	$2N$
The DLMS[1]-[3]	$2t_{add}+2t_{mlt}$	N	$2N+1$	$2N+1$	$8N-5$
Conv.1($\delta=N$-3) [4],[5]	$2t_{add}+t_{mlt}$	N	N^2+2	$4N-5$	$N^2+10N-4$
Conv.2[6]	$2t_{add}+t_{mlt}$	N	$5N+1$	$5N+1$	$6N$
Conv.3[7]	$2t_{add}+2t_{mlt}$	0	$5N-2$	$5N-1$	$5N$
Proposed	$2t_{add}+t_{mlt}$	D'-1	$2N+3D'$	$2N+3D'+1$	$2N+(2+D')D'-1+\sum_{k=1}^{D'-1}\left\lceil \frac{N}{2^{3k-1}} \right\rceil$

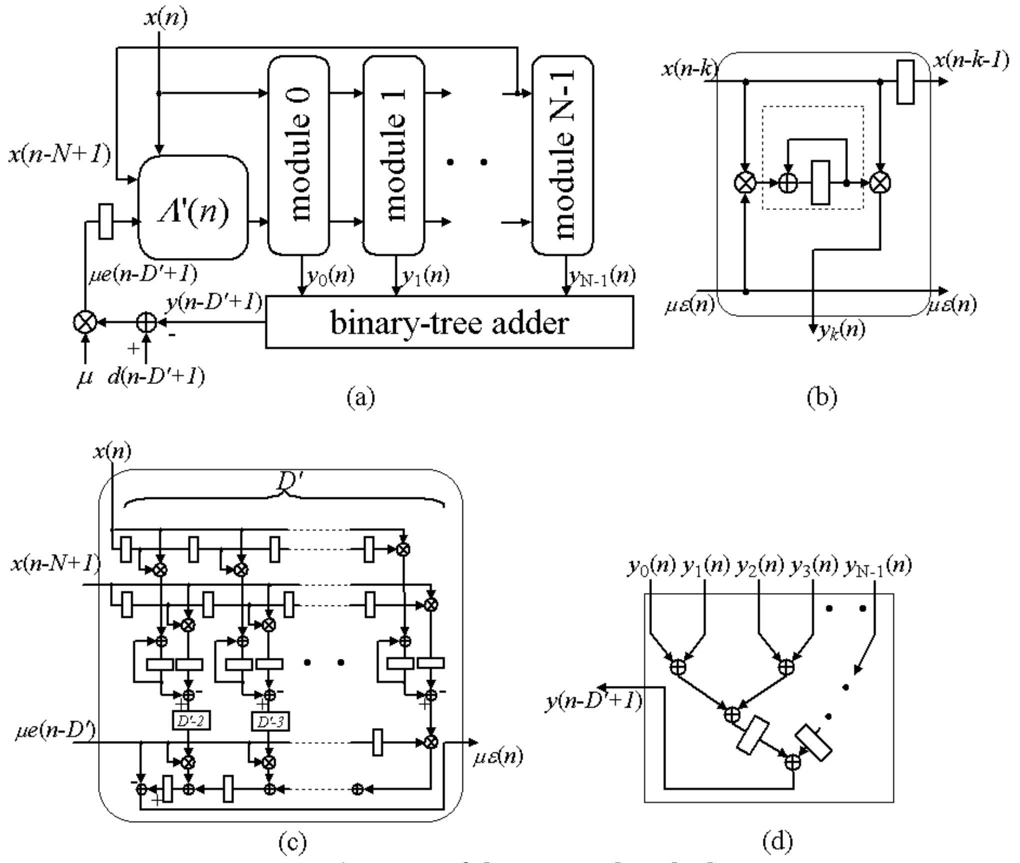

Fig.4 SFG of the proposed method
(a)whole architecture (b)module k (c)correctional term Λ'(n) (d)binary-tree adder

5 Conclusion

In this paper we proposed a new pipelined architecture for the DLMS algorithm. It enables us simultaneously to achieve good convergence characteristics, a short latency and high throughput characteristics with less than half an amount of calculation compared to the conventional architectures. To verify the validity of the proposed architecture, we showed the results of computer simulation of system identification and a comparison of characteristics of the proposed architecture with the conventional ones.

References

[1] R. H.-Cohen, H. Herzberg, and Y. Be'ery, "Delayed adaptive LMS filtering: current results," Proc. IEEE ICASSP '90, pp.1273-1276, April 1990

[2] G. Long, F.Ling, and J.G.Proakis, "The LMS algorithm with delayed coefficient adaptation", IEEE Trans. Accoust., Speech, and Signal Process., Vol.37, No.9, pp.1397-1405, Sept. 1989

[3] G. Long, F.Ling, and J.G.Proakis, "Corrections to "The LMS algorithm with delayed coefficient adaptation",", IEEE Trans. Signal Process., Vol.40, No.1, pp.230-232, Sept. 1989

[4] K.Matsubara, K.Nishikawa, and H.Kiya, "A New PipelinedI Architecture Of The LMS Algorithm Without Degradation Of Convergence Characteristics", Proc. IEEE ICASSP '97, pp.4125-4128, Apr. 1997

[5] K.Matsubara, K.Nishikawa, and H.Kiya, "Pipelined LMS adaptive filter using a new look-ahead transformation", IEEE ISCAS '97, pp.2309-2312, June 1997

[6] S. C. Douglas, Q. Zhu, and K. F. Smith, "A pipelined LMS Adaptive FIR Filter Architecture Without Adaptation Delay", IEEE Trans. Signal Process., Vol.46, No.3, pp.775-779, Mar. 1998

[7] A.Harada, K.Nishikawa, and H.Kiya, "Pipelined Architecture of the LMS Adaptive Digital Filter with the Minimum Output Latency", IEICE Trans. Fundamentals, Aug. 1998 (to be published)

[8] R.D.Poltmann, "Conversion of the delayed LMS algorithm into the LMS algorithm", IEEE Signal Proc. Letters, Vol.2, No.12, p.233, Dec. 1995

[9] N. R. Shanbhag and K. K. Parhi, "Pipelined Adaptive Digital Filters", Kluwer Academic Publishers ,1994