

A novel architecture for implementing pipelined FIR ADF by classification of coefficients

James Okello[†], Hirohisa Ohtsuki[†], Hiroshi Ochi^{††}, Yoshio Itoh[†], Yutaka Fukui[†], Masaki Kobayashi^{†††}

[†] Faculty of Engineering, Tottori University

4-101 Koyama-Minami, Tottori 680-8552 Japan

Phone: +81-857-31-5690 E-mail: b95t3014@maxwell.ele.tottori-u.ac.jp

^{††} Faculty of Engineering, Kyushu Institute of Technology, Japan

^{†††} Faculty of Engineering, Chubu University

Kasugai, 487-8501 Japan

Abstract

We propose a novel method for implementing pipelined FIR ADF, with an aim of reducing the maximum delay of the filtering portion of conventional DLMS pipelined ADF. This proposal accomplishes a maximum delay of one by classification of coefficients. The reduction does not depend on the order of the filter, which is an advantage when the order of the filter is very large, and as a result the method can also be applied to IIR filter.

1. Introduction

In certain applications, FIR ADF that is updated using the LMS algorithm may require that the order of the filter be made significantly large in order to achieve a better performance. Under such a condition, it may not be possible to implement this kind of a filter without reducing the sampling rate of the input signal, especially if the initial sampling rate was very high. This is essentially due to the adders within the filter which perform computation in a serial manner. Thus, each adder has to wait until the preceding adder has completed its computation.

In order to try and solve this problem, the delay coefficient adaptation method, otherwise known as the DLMS was proposed [1]. However, it has been noted that the performance of the DLMS decreases considerably as the length of the filter increases. This problem could be overcome by converting the DLMS into the LMS algorithm [2], at the cost of an increase in computational complexity. Another variant of the

pipelining method whose throughput is independent of the length of the filter has also been proposed [3]. It should however be noted that, even though the convergence speed improves to the same level as the LMS algorithm, the latency of the ADF is still large if a high throughput is desired. A DLMS ADF with a high throughput and a reduced latency has also been proposed [4]. In this method, the architecture is implemented using the tree structure, such that latency is significantly reduced. As was indicated in [3] and in comparison to the DLMS implemented using the tree structure, a far much lower latency of zero or one can be obtained. However this will be at the cost of reducing the throughput of the ADF.

An alternative way of tackling the problem of FIR ADF with very long lengths would be to implement FIR ADF using filter banks [5][6]. In this method, the input and the output signal from the unknown system are split into a number of subbands, resulting into the possibility of lowering the lengths of adaptive filter employing in each subband. However, in this paper we deal with only the pipelining technique, with an aim of reducing the latency associated with the conventional method, while the same time improving the convergence speed.

Thus, we propose new pipelining method that reduces the delay of the output signal and the latency of the filter and the algorithm to one and two respectively, independent of the order of the filter. This proposal achieves a high throughput, thus allowing for the operation in systems with high sampling rates. The maximum delay associated with the ADF's coefficients of the proposed method is significantly reduced in

comparison to the transpose type FIR structure, which also has a very low latency [7]. The simulation results of the proposed method show comparable convergence speed with LMS algorithm.

2. Pipelining architecture

2.1 Conventional Pipelining Methods

In this sub-section, the conventional pipelining methods for the FIR ADF are reviewed. The bold lowercase alphabetical letters indicates vectors.

DLMS algorithm [1] is given by

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

where

$$\mathbf{h}(n) = [h_0(n) \ h_1(n) \ \dots \ h_{N-1}(n)]^T$$

$$\mathbf{x}(n) = [x(n) \ x(n-1) \ \dots \ x(n-N+1)]^T$$

$$e(n) = d(n) - \mathbf{h}^T(n)\mathbf{x}(n)$$

μ is the step size of adaptation, $x(n)$ is the input signal, $d(n)$ is the desired signal, and T is the transpose of a vector. h_i (for $i=0, 1, \dots, N-1$) are the coefficients of the FIR ADF. This algorithm includes the FIR ADF latency that is the same as the order of the filter, if the highest throughput is desired. The throughput that can be considered and which involves a maximum of one multiplier and two adders within the critical pass [3], is given by

$$f_{thr} = \frac{1}{t_m + 2t_a} \quad (2)$$

where t_m and t_a is the calculate time of a multiple and an add operation, respectively. In this algorithm, the convergence speed deteriorates, as the order of the ADF is increased [3][4]. This kind of problem has been solved with a correction factor $\Lambda(n)$ into Eq.(1) as shown below [4]

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mu \mathbf{x}(n-D)\{e(n-D) - \Lambda(n)\} \quad (3)$$

where

$$\Lambda(n) = \sum_{i=1}^{D-1} \mu \mathbf{x}(n-2D-i)\mathbf{x}^T(n-2D-i)\mathbf{x}(n-D)$$

However, since there are large amount of the delay D for implementing high order filters, the latency becomes very high. Due to reduce the latency, the DLMS with the

tree structure was proposed. This method reduced the delay D with the result being a reduction in the computational complexity of the algorithm with the correction factor. It should however be realized that whereas the delay has been significantly reduced, its value still depends on the length of the FIR ADF.

2.2 Proposed Architecture

In this section, a novel architecture for implementing pipelined ADF is proposed. The proposed method involves, delaying the output of the FIR filter by one sample before dividing the coefficients into the groups of odd and even numbered coefficients. It can then be shown that by retiming, the architecture can be implemented as shown in Fig.1. In this figure, it can be seen that maximum frequency otherwise known as the throughput, at which the input samples $x(n)$ may appear at the input will be given by $1/(t_m + t_a)$. This is the same as the transpose FIR digital filter. However, from Fig.1, it can be seen that the effect of a change of the coefficient h_{N-1} will be felt at the output $y(n-1)$ after $\text{ceil}\{(N+1)/2\}$ samples, where $\text{ceil}(y)$ is the smallest integer not less than y . If we consider the adaptive algorithm for this architecture, we will still have to add two more delay elements so as to maintain a throughput of $1/(t_m + t_a)$.

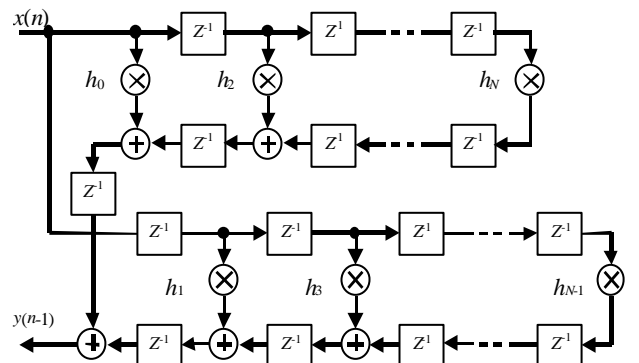


Fig.1. Proposed architecture obtained by classification of coefficient into odd and even group and then re-timing.

This adaptive algorithm is give by

$$\mathbf{h}(n+1) = \mu \mathbf{x}(n-3)e(n-3) \quad (4)$$

With the inclusion of this algorithm, the effect of a change in the coefficient h_{N-1} will still be felt at

estimation error after $\text{ceil}\{(N+1)/2\}$ samples, a value which is less than that of the transpose FIR ADF. In other words, this structure will exhibit a faster convergence speed when compared with the transpose algorithm, which is known to have better performance in comparison to the DLMS.

For comparison purpose, we will consider the throughput defined in Eq.2. It is also possible to reduce significantly this delay which is maximum at the coefficient h_{N-1} , by implementing the first three even coefficients using the tree structure. The remaining even coefficients are then renumbered from zero, so as to allow a re-classification into even and odd coefficients. This procedure will then repeated until there are at most four coefficients left in each path. The final architecture will then be in the form of the tree structure as shown in Fig.2 (a), but completely different from the conventional method of implementing tree structured FIR digital filter.

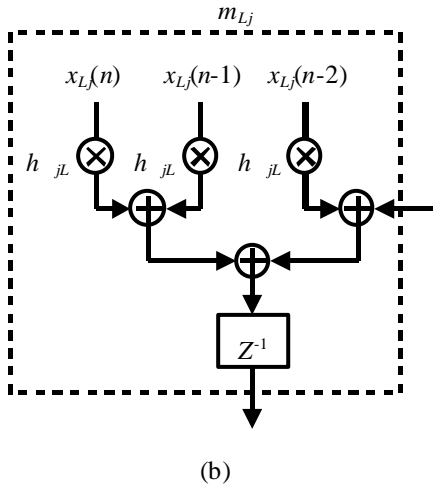
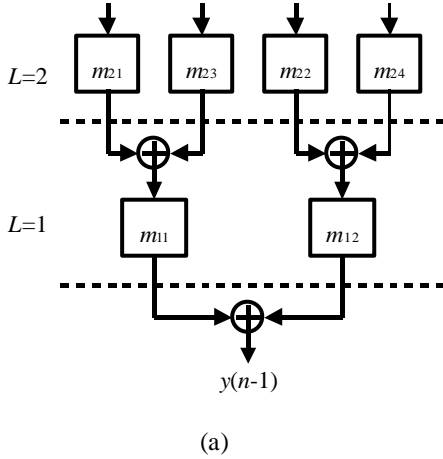


Fig.2. Proposed architecture based on classification and realization using the tree structure.

In this method, the coefficients appear under levels, where the maximum number of levels Max_L is given by

$$\begin{aligned} Max_L &= l(N) \\ &= \text{ceil}\{\log_2\{1+0.5*\text{ceil}(N/3)\}\} \end{aligned} \quad (5)$$

Each level L has 2^L modules, whereas each module m_{jL} with number j has the filter coefficients given by Fig.2 (b). The numbering of the modules is such that their first coefficients are in an increasing order. The coefficients \mathbf{h}_{jL} in each module j ($j=0,1,\dots,2^L$) can be given in a more general form as

$$\mathbf{h}_{jL} = [h_{jL} \ h_{jL} \ h_{jL}] \quad (6)$$

where

$$\mathbf{a}_{jL} = j + \sum_{l=1}^{L-1} 3(2)^l, \mathbf{b}_{jL} = \mathbf{a}_{jL} + 2^L \text{ and } \mathbf{g}_{jL} = \mathbf{a}_{jL} + 2^{L+1}$$

The adaptive algorithm which is obtained by calculating the gradient of $e^2(n-1)$ and then delaying at an appropriate location the resulting term by a single sample so as to achieve the same throughput as in Eq.2 is given by

$$\mathbf{h}(n+1) = \mathbf{h}(n) + \mathbf{m}\mathbf{x}(n-2)e(n-2) \quad (7)$$

It can be seen that with the proposed architecture, the effect of an update on the coefficient h_{N-1} will be seen at the output $y(n)$ after only $Max_L + 1$ samples. This delay is much less than transpose FIR ADF with the same throughput. Hence, the proposed algorithm and architecture is expected to outperform completely the transpose FIR ADF.

Furthermore, it is well known that the DLMS implemented using the tree architecture reduces the number of delay elements. This essentially results in a reduction in the computational complexity in comparison to other architectures and the transpose architecture with the correction factor. In this paper, we therefore compare the complexity of the proposed architecture in comparison to the tree structure. The total number of delay elements will be the values indicated in Fig.3, added to the delay elements in the direct form FIR LMS algorithm. From this result we observe that the extra number of delay elements is less than that of the tree

structure for certain order of FIR filter, such as 4000. For the same length of FIR filter, and with the technique of re-using adders, the number of adder used in the filtering section of the ADF is also less than those of the tree structure.

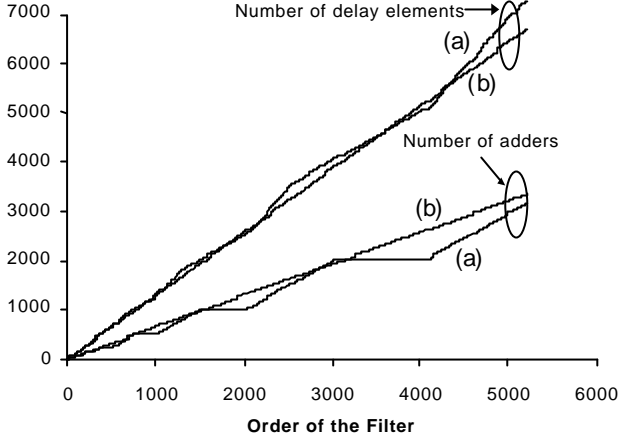


Fig.3. Comparison of the number of delay elements and adders between (a) the proposed and (b) the tree architecture: without the correction factor.

3. Conversion of the algorithm to LMS algorithm

In this section, we shall present a correction factor $e(n)$ that can be used with the algorithm of the proposed architecture so as to transform it into a LMS algorithm. The basic concept is essentially the same as in [4].

In the proposed algorithm, $e(n-2)$ is defined as

$$e(n-2) = d(n-2) - \{\mathbf{h}'(n-2)\}^T \mathbf{x}(n-2) \quad (8)$$

where

$$\begin{aligned} \mathbf{h}'(n) = & [h_0(n) \cdots h_5(n) h_6(n-1) \cdots h_{17}(n-1) \\ & \cdots h_{i-1}(n-x_i) \cdots h_{N-1}(n-x_N)]^T \\ x_i = & 1 - l(i) \end{aligned}$$

The aim of the correction factor $e(n)$ is to transform the error signal from $e(n-2)$ to $e'(n-2)$ such that

$$\begin{aligned} e'(n-2) = & e(n-2) - e(n) \\ = & d(n-2) - \mathbf{h}^T(n) \mathbf{x}(n-2) \end{aligned} \quad (9)$$

Under this condition, $e(n)$ will be given by

$$e(n) = \{\mathbf{h}(n) - \mathbf{h}'(n-2)\}^T \mathbf{x}(n-2)$$

$$\begin{aligned} & \begin{bmatrix} \sum_{i=1}^2 x(n-2-i)e'(n-2-i) \\ \vdots \\ \sum_{i=1}^{2+x_k} x(n-2-k-i)e'(n-2-i) \\ \vdots \\ \sum_{i=1}^{2+x_N} x(n-2-N-i)e'(n-2-i) \end{bmatrix}^T \mathbf{x}(n-2) \\ & = \mathbf{m} \begin{bmatrix} e'(n-3) \\ e'(n-4) \\ \vdots \\ e'(n-4-x_N) \end{bmatrix}^T \begin{bmatrix} \sum_{i=1}^N x(n-2-i)x(n-1-i) \\ \vdots \\ \sum_{i=I_k}^N x(n-2-k-i)x(n-1-i) \\ \vdots \\ \sum_{i=I_{x_N+2}}^N x(n-2-i)x(n-1-i) \end{bmatrix} \end{aligned} \quad (10)$$

where

$$I_k = \begin{cases} \sum_{i=3}^k 3(2)^{k-2} & \text{if } k > 2 \\ 1 & \text{otherwise} \end{cases}$$

and k ($k=1, 2, \dots, x_N+2$) is the column number.

Even though I_k and $l(N)$ seems complex to calculate, their values are constants which can be pre-calculated during the design, and as a result their computation does not contribute to the computational load of the proposed architecture for FIR ADF. If we consider this fact, it will then be realized that transpose FIR ADF with a correction factor is far much more computationally complex in comparison to the proposed architecture. Figure 4 shows the circuitry that is used to calculate the correction factor. In this figure,

$$R_k(z) = \begin{cases} \sum_{i=0}^{N-1} z^{-i} & \text{if } k < 2, \\ \sum_{i=0}^{N-1-I_k} z^{-i} & \text{otherwise} \end{cases} \quad (11)$$

The filters with the transfer function $R_k(z)$ can either be implemented using the recursive [3] or the direct form structure. The recursive structure has an advantage of utilizing only two adders, but a disadvantage of being susceptible to error propagation. In the case of the FIR structure, and depending on the order of this structure, we can reduce the number of adders by using the

proposed architecture.

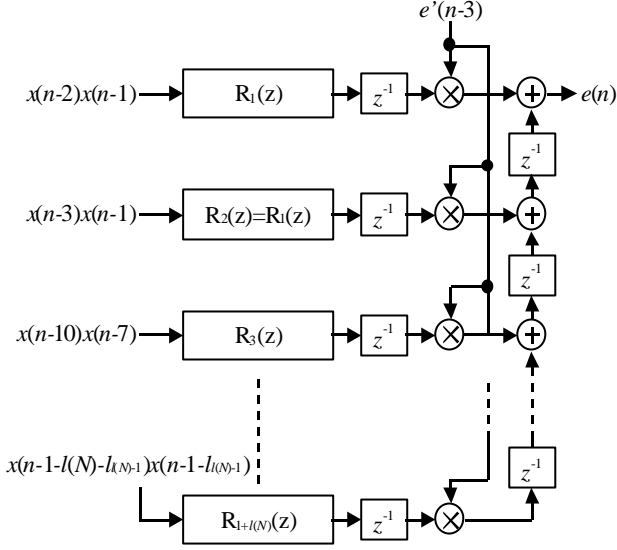


Fig.4. Architecture for implementing the correction factors.

4. Simulation and result

In this section we verify through simulation the performance of proposed pipelining architecture. The length of the FIR ADF used in the simulation was 2500. This was the same as that of unknown system. An additive white gaussian noise was added to the output of the unknown system such that the signal to noise ratio (S/N) was 40 dB. The measure of performance was the impulse response error ration (IRER) defined as

$$\text{IRER} = 10 \log_{10} \frac{\sum_{i=0}^N \{\hat{h}_i - h_i(n)\}^2}{\sum_{i=0}^N \{\hat{h}_i\}^2} \text{ dB} \quad (12)$$

where $\hat{h}_i \{i = 0, 1, 2, \dots, N\}$ is the impulse response of the unknown system. The results were compared with the LMS algorithm and the transpose-FIR ADF.

In this simulation, the proposed algorithm defined in Eq.(7), which does not have the correction factor, was used. Figure 5 shows the result obtained when the step size of the transpose FIR ADF was adjusted so as to give it its fastest convergence. The step size of the LMS and the proposed algorithm were then adjusted so as to achieve a final IRER that is the same as the transpose algorithm (about 80 dB). From this result we observe that the proposed method achieve better result than the

transpose-FIR ADF. The performance of the proposed architecture without the correction factor is also fairly close to the LMS algorithm. This is due to the reduced delay.

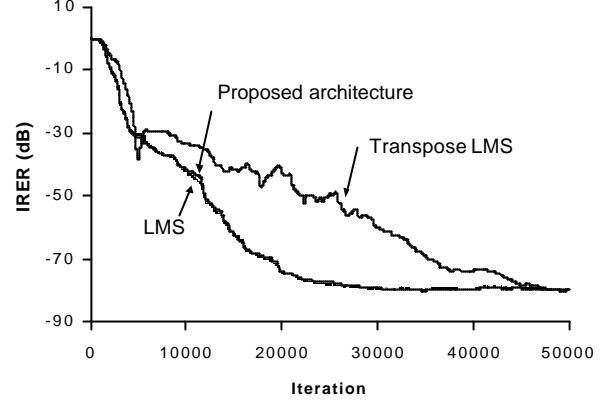


Fig.5. Comparison of the convergence characteristic with the step size adjusted to achieve the fastest convergence for the transpose type and a final steady state IRER of about 80dB.

5. Conclusion

We have proposed a novel pipelining method for the FIR ADF. The proposed method achieves high throughput with a latency of only two. In comparison to the transpose-FIR ADF, the maximum delay after the effect of the change in the last coefficient will be seen at the output of the FIR ADF has been significantly reduced. The number of delay elements utilized has likewise been reduced, making the proposed architecture far much more advantageous than the transpose-FIR ADF.

References

- [1] G. Long, F. Ling and J. G. Proakis, "The LMS algorithm with delayed coefficient adaptation". *IEEE Trans. Acoust., Speech, Signal Processing*. vol. 37, no. 9, pp.1397-1405, Sept. 1989.
- [2] R. D. Poltmann, "Conversion of the delayed LMS Algorithm into the LMS algorithm". *IEEE Signal Processing Letters*. vol. 2, no. 12, pp. 223, Dec. 1995.
- [3] S. C. Douglas, Q. Zhu, and K. F. Smith, "A pipelined LMS adaptive FIR filter architecture without adaptation delay". *IEEE Trans. Signal Processing*. vol. 46, no. 3, March 1998.
- [4] T. Kimijima, K. Nishikawa and H. Kiya, "An effective architecture of the Pipelined LMS adaptive filters". *IEICE Trans. Fundamentals*. vol. E82-A. no. 8, August 1999.
- [5] Breining C., et. al. "Acoustic echo control: An application of very-high-order adaptive filters". *IEEE Signal Processing Magazine*, vol. 16, no. 4, pp.42-69, July 1999.
- [6] Strang G. and Nguyen T. "Wavelets and filter banks". *Wellesley-Cambridge Press* 1996, pp.104.
- [7] D. J. Jones, "Learning characteristics of Transpose-Form LMS adaptive filters". *IEEE Trans. Circuit and Sys.* - vol. 39, no. 10, Oct. 1992.