HIGHLY PARALLEL ADAPTIVE FILTER

Janusz A. Starzyk and Mohammad Eshghi

Department of Electrical and Computer Engineering
Ohio University Athens OH 45701

ABSTRACT

This paper introduces a new parallel algorithm for digital adaptive filters. The algorithm is a look-ahead realization of a transversal filter, where a forgetting factor is employed to diminish the effect of the past errors in the performance index. We have illustrated how such algorithm can be implemented in a regular and modular structure, suitable for VLSI implementation. The maximum sampling rate is limited by time of adding two binary inputs. If the input signal has the sampling rate lower than the maximum sampling rate, then the architecture can be tailored to minimize the hardware costs.

1. INTRODUCTION

Adaptive digital filtering is entering real time applications in high frequency devices. Using VLSI technology, adaptive filters are already being built for sampling rates 10-20 kHz [1]. However, further progress is both expected and possible. Considering limitations of modern digital technology we can improve the processing speed by increasing system parallelism. In our paper we try to address this issue.

In adaptive filtering an effect of estimation error is accumulated in the performance index. The purpose of adaptation process is to minimize this performance index. Many algorithms for this minimization have been developed based on different criteria [2]-[5]. Some algorithms minimize the performance index based on the least mean squares norm [6],[7], some on the least square norm [8], and some on the gradient vector [9]. While these methods differ from each other in their structures, applications and basic theories, all of them are sequential.

An important objective in the adaptive filter design is reduction of time needed to calculate new values of variable tap gains. In spite of improvements in adaptive algorithms [10]-[12] this time is still too long for many practical systems. In some applications, such as image or speech processing or unknown system identification, a real-time response to the high frequency inputs is desired. This goal can not be achieved unless tedious computations for adjusting the tap values of the filter are reduced.

A recent paper [13] proposed to combine the pipelining and parallel processing in order to speed up the tap adjustment. The look-ahead computing was used with the recursive algorithm resulting in a lattice pipelined structure. Consequently a high-speed realization of adaptive filter was achieved with containment in the design area.

This paper follows the same general strategy. The parallel concept is applied to a transversal adaptive filter with a forgetting factor based on the least mean squares method. We show that the parallel structure can be developed for non-recursive algorithms. Consequently a simple realization can be obtained with savings in the hardware and independence of the processing time on the number of look-ahead steps.

2. LEAST MEAN SQUARES ADAPTIVE FILTERS

Consider an adaptive filter shown in Fig. 1.

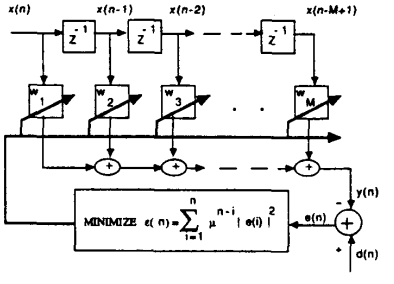

Fig.1. An adaptive filter with minimum sum of weighted squares of errors.

The response of this filter depends on the tap weights vector w(n)

$$y(n) = \mathbf{w}^H(n)\,\mathbf{x}(n), \qquad (1)$$

where H denotes the Hermitian transpose (complex conjugate transpose), x(n) is an M-by-1 input vector at time n

$$\mathbf{x}^T(n) = [\, x(n)\ x(n-1)\ \ldots\ x(n-M+1)\,].$$

and w(n) is an M-by-1 tap weights vector at time n

$$\mathbf{w}^T(n) = [\, w_1(n)\ w_2(n)\ \ldots\ w_M(n)\,].$$

ISCAS '89

CH2692-2/89/0000-2116 $1.00 © 1989 IEEE

M is the order of adaptive filter or the number of adjustable tap weights.

The problem is to find such values of the tap weights which minimize the performance index based on an estimation error

$$e(n) = d(n) - y(n) \qquad (2)$$

where d(n) is a desired output and y(n) is an actual output of the filter at time n. The performance index $\in(n+1)$ is defined as

$$\in(n+1) = \sum_{i=1}^{n+1} \mu^{n+1-i} \| e(i) \|^2 \qquad (3)$$

where μ is a forgetting factor $(0<\mu<1)$.

Substituting (1) and (2) in (3), we get

$$\in(n+1) = \sum_{i=1}^{n+1} \mu^{n+1-i} [d\, d^* - d^* \mathbf{w}^H \mathbf{x} - d\mathbf{x}^H \mathbf{w} + \mathbf{w}^H \mathbf{x}\mathbf{x}^H \mathbf{w}] \qquad (4)$$

In order to minimize the performance index we differentiate $\in(n+1)$ with respect to $\mathbf{w}$. The minimum is obtained when this derivative is equal to zero, which yields the normal equation

$$\Phi(n+1)\, \mathbf{w}(n+1) = \theta(n+1) \qquad (5)$$

where $\Phi(n+1)$ denotes the M-by-M correlation matrix at time n+1

$$\Phi(n+1) = \sum_{i=1}^{n+1} \mu^{n+1-i}\, \mathbf{x}(i)\, \mathbf{x}^H(i) \qquad (6)$$

and $\theta(n+1)$ denotes the M-by-1 cross correlation vector at time n+1

$$\theta(n+1) = \sum_{i=1}^{n+1} \mu^{n+1-i}\, \mathbf{x}(i)\, d^*(i) \qquad (7)$$

From the normal equation the optimum tap gains are

$$\underline{w}(n+1) = \Phi^{-1}(n+1)\, \theta(n+1) \qquad (8)$$

Different methods are used to solve the normal equation. In [13] a recursive algorithm was successfully implemented in a pipelined parallel lattice structure. In our work we use look-ahead computation to develop a parallel architecture for a direct algorithm, based for example on LU-factorization.

3. PARALLEL ALGORITHM FOR TRANSVERSAL FILTERS

Solution of the normal equation (5) gives the optimal tap gains that minimizes the performance index. We rewrite (6) as

$$\Phi(n+1) = \mu \left[\sum_{i=1}^{n} \mu^{n-i} \mathbf{x}(i)\mathbf{x}^H(i) \right] + \mathbf{x}(n+1)\mathbf{x}^H(n+1) \qquad (9)$$

and similarly (7) as

$$\theta(n+1) = \mu \left[\sum_{i=1}^{n} \mu^{n-i}\, \mathbf{x}(i)\, d^* \right] + \mathbf{x}(n+1)\, d^*(n+1) \qquad (10)$$

Two previous equations result in

$$\Phi(n+1) = \mu\, \Phi(n) + \mathbf{x}(n+1)\mathbf{x}^H(n+1) \qquad (11)$$

and

$$\theta(n+1) = \mu\, \theta(n) + \mathbf{x}(n+1)d^*(n+1). \qquad (12)$$

Using (11) and (12), we can derive a look-ahead formulation of the normal equation. At the time step n+2 the two components of the normal equation are as follows

$$\Phi(n+2) = \mu^2\, \Phi(n) + \mu\mathbf{x}(n+1)\mathbf{x}^H(n+1) + \mathbf{x}(n+2)\mathbf{x}^H(n+2) \qquad (13)$$

and

$$\theta(n+2) = \mu^2\, \theta(n) + \mu\mathbf{x}(n+1)d^*(n+1) + \mathbf{x}(n+2)d^*(n+2) \qquad (14)$$

A general formula for the correlation matrix and the cross-correlation vector for the next s steps can be obtained as

$$\Phi(n+s) = \mu^s\, \Phi(n) + \sum_{i=n+1}^{n+s} \mu^{n+s-i}\, \mathbf{x}(i)\mathbf{x}^H(i) \qquad (15)$$

and

$$\theta(n+s) = \mu^s\, \theta(n) + \sum_{i=n+1}^{n+s} \mu^{n+s-i}\, \mathbf{x}(i)d^*(i) \qquad (16)$$

Having the correlation matrix and the cross-correlation vector at time n as well as inputs and desired outputs for the next s steps of time, we can formulate the normal equation up to time (n+s)

$$\Phi(n+s)\, \mathbf{w}(n+s) = \theta(n+s). \qquad (17)$$

We illustrate this algorithm with a simple example, where we describe an organization of a circuit designed to generate the normal equation.

Example:

Consider a transversal filter with 3 taps (M=3). We want to calculate the tap weights with 5-step parallel look-ahead algorithm (s=5). This means that having the correlation matrix and the cross-correlation vector at time n we want to formulate the normal equation for time (n+5). In order to formulate this normal equation, we calculate the elements of the correlation matrix $\Phi(n+5)$ and the cross-correlation vector $\theta(n+5)$ using (15) and (16).

All products of input signals

$$x(i)x^*(j) \qquad n-1 \le i,j \le n+5$$

are required in order to compute the Φ matrix. All these products are obtained in a regular, well organized structure as illustrated in Fig. 2. In this figure vertical lines are used to supply the original input signals while the horizontal lines supply the complex conjugate inputs. Products are calculated at intersections of rows and columns in the shaded area only.

Obviously the forgetting factor adds different weights to each of these products. In order to reduce the number of multipliers in our structure we submit original inputs pre-multiplied by different powers of the forgetting factor. Consequently we design a regular, modular and

(217) 333-4789 Further information can be obtained from Dr. W. Kenneth Jenkins.

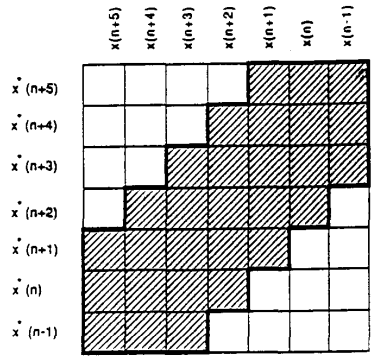

Fig. 2. Array of input products.

cellular structure to calculate the correlation matrix and the cross-correlation vector. Fig. 3 illustrates this structure for our example. Fig. 3a) shows a general floor plan of the subsystem generating the Φ matrix. On the left side of this figure we have an array of cells performing necessary operations. The internal structure of each cell is shown in Fig. 3b). Each black square in the cell in Fig.3a) indicates the presence of a corresponding multiplier and adder from Fig. 3b). On the right side of Fig. 3a) we have the accumulator matrix, which has cells as shown in Fig. 3d). Vectors $v(n-i)$ in Fig. 3a) represent input signals $x(n+i)$ premultiplied by different powers of μ.

$$v(n+i) = x(n+i) \begin{bmatrix} \mu^{s-i-2} \\ \mu^{s-i-1} \\ \mu^{s-i} \end{bmatrix}$$

Each accumulator cell adds a result supplied by a horizontal line to the value stored in the register R_1 (initially all registers R_1 are set to zero). The result of addition is stored in the register R_2 and can be used in the solution of the *normal equation*. *The normal equation is solved* in an efficient pipelined architecture, using for example a hexagonal array of processors discussed in [15]. Every T_1 units of time a new normal equation is formulated and fed into the solution pipeline, where T_1 is a loading time necessary to calculate $\Phi(n+s)$ and $\theta(n+s)$ from (15) and (16). At the beginning of the next loading period the value stored in R_2 is multiplied by μ^s and stored in R_1.

The sampling period T_{in} of the input signal must satisfy the following inequality

$$s\, T_{in} > T_1$$

It is clear that for a given T_1 we can increase the sampling frequency by increasing s.

The cross-correlation vector can be obtained without difficulties during the loading period since it requires only two multiplications and one addition per each component. A floor plan of the subsystem used to generate the θ vector is shown in Fig. 4. Cells on the left side of this figure are the same as in Fig. 3b), however each of them contains only one multiplier and one adder as indicated by the black squares. Each cell on

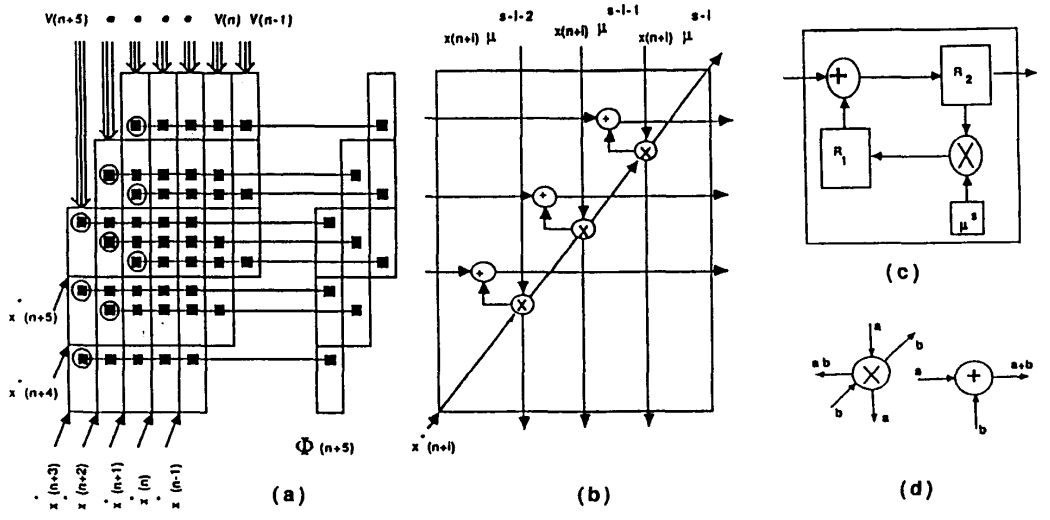

Fig. 3. a) Subsystem generating Φ matrix, b) Cell structure in the array in Fig.3.a), c) Definition of processing elements used, d) Accumulator cell.

the right side of Fig. 4 have the structure shown in Fig. 3d).

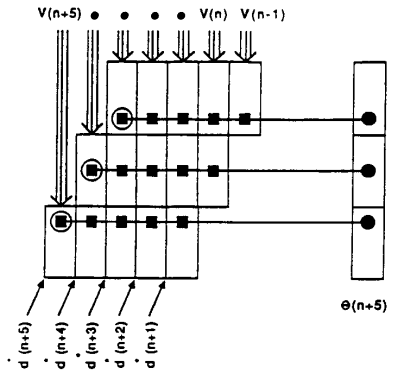

Fig. 4. Subsystem generating θ vector.

4. TIME AND AREA REQUIRED

As discussed in the previous section, the proposed algorithm formulates and solves the normal equation (17). Our discussion in Example was concentrated on formulation of this equation i.e. computation of the correlation matrix and the cross-correlation vector. A circuit to generate the normal equation, can be built using structures presented in Example. On this basis we can estimate time and area required to built such a circuit. The result is summarized in Table 1.

Table 1. Time and area required to calculate Φ and θ

	multipliers	adders	registers	time
Φ	$(s+1)M^2$	$(s+1)M^2$	$2M^2+(s+1)M+s-1$	$t_1=t_m+st_a$
θ	$(s+1)M$	$(s+1)M$	$2M+s$	$t_1=t_m+st_a$
TOTAL	$M(M+1)(s+1)$	$M(M+1)(s+1)$	$2M^2+(s+3)M+2s-1$	$t_1=t_m+st_a$

In Table 1 t_m and t_a denote time of one word multiplication and one word addition, respectively.

A modification of this algorithm allows for a real-time response implementation with the frequency of tap adjustment limited only by the solution time of the normal equation. The input frequency will be limited by time needed to calculate and add one product term. This time can be reduced to as little as time needed for addition of two bits, if a parallel structure is used to calculate product terms in (15) and (16).

CONCLUSION

We have demonstrated that using a look-ahead concept we can built a highly parallel and modular architecture for adaptive filters based on non-recursive algorithm. Due to its regularity the architecture is suitable for VLSI implementation. As a result we expect that a real time implementation of adaptive algorithms is possible for much higher frequency that can be reached by presently known techniques.

REFERENCES

[1] P. Denyer, and D. Renshow, "VLSI Signal Processing : a Bit-Serial Approach," Addison-Wesley, 1985.

[2] S. Haykin, "Adaptive Filter Theory," Englewood Clifton, NJ: Prentice-Hall, 1986.

[3] S. T. Alexander, "Adaptive Signal Processing," New York, Springer-Verlag, 1986.

[4] L. H. Sibul, "Adaptive Signal Processing," IEEE Press, 1987.

[5] J. Makhoul, "Linear Prediction: a Tutorial Review," Proceedings of the IEEE, April 1975.

[6] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, Jr., "Stationary and Nonstationary Learning Characteristics of LMS Adaptive Filters," Proceedings of the IEEE, August 1976.

[7] B. Widrow, and M. E. Hoff, Jr., "Adaptive Switching Circuits," IRE WESCON Conv. Rec., Part 4, pp. 96-104, 1960.

[8] W. Murray, "Numerical Methods for Unconstrained Optimization," New York, Academic Press, 1972.

[9] C. L. Lawson, and R. J. Hanson, "Solving Least Squares Problems," Englewood Cliffs, N.J., 1974.

[10] G. A. Clark, S. K. Mitra, and S. R. Parker, "Block Implementation of Adaptive Digital Filters," IEEE Trans. Circuits Syst., June 1981.

[11] G. Carayannis, D. G. Manolakis, and N. Kalouptsidis, "A fast sequential algorithm for least squares filtering and prediction," IEEE Trans. Acoust. Speech Signal Process., Dec. 1983.

[12] J. M. Cioffi, and T. Kailath, "Fast, recursive least squares transversal filters for adaptive filtering," IEEE Trans. Acoust. Speech Signal Process., Apr. 1984.

[13] K. K. Parhi, and D. G. Messerschmitt, "Concurrent Cellular VLSI Adaptive Filter Architechture," IEEE Trans. Circuits Syst. Oct. 1987.

[14] C. Mead, and L. Conway, "Introduction to VLSI Systems," Addison-Wesley, 1980.

(217) 333-4789 Further information can be obtained from Dr. W. Kenneth Jenkins.