Pipelined Adaptive IIR Filter Architecture

Naresh R. Shanbhag

AT&T Bell Laboratories
600 Mountain Avenue,
Murray Hill, NJ 07974.

Gi-Hong Im

AT&T Bell Laboratories
200 Laurel Avenue,
Middletown, NJ 07748.

Abstract

Presented in this paper are fine-grain pipelined architectures for adaptive infinite impulse response (AIIR) filters. The AIIR filters are equation error based. The proposed architectures are developed by employing a combination of scattered look-ahead and relaxed look-ahead pipelining techniques. The scattered look-ahead technique is applied to the non-adaptive (but time-varying) recursive section. The relaxed look-ahead technique is applied to the adaptive blocks. It is shown via simulations that speed-ups of up to 8 and more can be achieved with marginal or no degradation in performance.

1 Introduction

It is being recognized that in addition to improving performance measures (such as signal-to-noise ratio (SNR) and bit error rate (BER)), design of digital signal processing and communications algorithms also needs to take into account implementation issues such as area, power dissipation and speed. Therefore, in recent years, *algorithm transformation techniques* [1] have been formulated, which transform a given algorithm in order to make it more flexible to implement in silicon. In particular, it can be shown that by combining pipelining [2] with folding [3], it is possible to trade-off area with speed. Thus, speed, power and area can be optimized by the design of pipelined algorithms.

In this paper, we employ *relaxed look-ahead* [4] and *scattered look-ahead* [2] to pipeline the adaptive infinite impulse response (AIIR) filter. The AIIR filters under consideration are based on the *equation error* [5] formulation. In particular, we apply relaxed look-ahead to the adaptive section and scattered look-ahead to the non-adaptive block of the AIIR. The adaptive section is pipelined using the *delay relaxation* and *sum relaxation* [4]. The performance of the proposed pipelined AIIR filter architectures are also verified via simulations.

This paper is organized as follows. In section 2, we describe pipelined system identification scenario. In section 3, we derive the pipelined AIIR (PIPAIIR) filter architecture and in section 4, we present simulation results to demonstrate the performance of the proposed pipelined architecture.

2 Pipelined System Identification

In order to develop pipelined AIIR filter architectures, it is essential to formulate a pipelined system identification setup. In Fig. 1, we show the conventional serial system identification scenario, where $H(z^{-1})$, the unknown plant, is given by

$$H(z^{-1}) = \frac{B(z^{-1})}{1 - A(z^{-1})} = \frac{\sum_{i=0}^{NB-1} b_i z^{-i}}{1 - \sum_{i=1}^{NA} a_i z^{-i}} \qquad (2.1)$$

with numerator polynomial $B(z^{-1})$ and denominator polynomial $1 - A(z^{-1})$, $\eta(n)$ is the additive noise uncorrelated with the input $x(n)$, $y(n)$ is the plant output (also the desired signal) and $H_m(n, z^{-1})$ is the time-varying model. Note that the numerator polynomial $B_m(n, z^{-1})$ and the polynomial $A_m(n, z^{-1})$ are adaptively computed.

We can employ *relaxed look-ahead* to pipelining the adaptive sections in Fig. 1. However, the time-varying recursive section $1/(1 - A_m(n, z^{-1}))$ also needs to be pipelined. From [2], we know that a fixed coefficient recursive filter can be pipelined via scattered look-ahead without effecting the stability. Therefore, in the pipelined system identification scenario (see Fig. 2), we assume the plant to be in a scattered look-ahead form given by

$$H(z^{-1}) = \frac{B(z^{-1}) \prod_{k=1}^{M-1} (1 - \sum_{i=1}^{NA} a_i z^{-i} e^{\frac{-j2\pi k}{M}})}{1 - \sum_{i=1}^{NA} a'_i z^{-Mi}}, \qquad (2.2)$$

where M is the level of pipelining. In practice, given a plant of type (2.1), we can emulate the behavior of an equivalent pipelined plant (2.2) by delaying the output of the plant $y(n)$ by M latches (see Fig. 2).

3 Pipelined AIIR Filter Architectures

In this section, we first present the serial AIIR (SAIIR) filter architecture and then derive the PIPAIIR architecture. This architecture is developed via the application of the relaxed look-ahead [4] technique

0-7803-2570-2/95 $4.00 ©1995 IEEE

to the pipelined system identification scenario, which was developed using scattered look-ahead [2] (see Fig. 2).

3.1 The SAIIR Architecture

The serial AIIR filter is described by the following equations

$$\mathbf{W}(n) = \mathbf{W}(n-1) + \mu e(n)\mathbf{U}(n) \quad (3.1)$$

$$e(n) = y(n) - \mathbf{W}^T(n-1)\mathbf{U}(n) \quad (3.2)$$

$$\hat{y}(n) = \mathbf{W}^T(n-1)\hat{\mathbf{U}}(n) \quad (3.3)$$

$$e_1(n) = y(n) - \hat{y}(n), \quad (3.4)$$

where

$$\mathbf{W}(n) = \begin{bmatrix} \mathbf{B}(n) \\ \mathbf{A}(n) \end{bmatrix}, \quad (3.5)$$

$$\mathbf{B}^T(n) = [b_0(n), b_1(n), \ldots, b_{N_B-1}(n)] \quad (3.6)$$

$$\mathbf{A}^T(n) = [a_1(n), a_2(n), \ldots, a_{N_A}(n)] \quad (3.7)$$

$$\mathbf{U}^T(n) = [x(n), ., x(n-N_B+1), y(n-1), ., y(n-N_A)] \quad (3.8)$$

$$\hat{\mathbf{U}}^T(n) = [x(n), ., x(n-N_B+1), \hat{y}(n-1), ., \hat{y}(n-N_A)]. \quad (3.9)$$

Note that $\mathbf{W}(n)$ is the coefficient vector with $\mathbf{B}(n)$ and $\mathbf{A}(n)$ being the coeffcient vectors of the numerator and denominator polynomials, respectively. In addition, μ is the adaptation step-size, $e(n)$ is the adaptation error, $\hat{y}(n)$ is an estimate of the plant output generated by the adaptive filter and $e_1(n)$ is the estimation error.

The serial AIIR (SAIIR) filter architecture is shown in Fig. 3, where **FB** block and **FA** block do the filtering operation (see (3.3)) associated with $B_m(n, z^{-1})$, and $A_m(n, z^{-1})$, respectively. Weight-update blocks **WUDB** and **WUDA** compute the coefficients of **FB** block and **FA** block, respectively (see (3.1)). The **FC** block computes the denominator polynomial $1 - A_m(n, z^{-1})$ according to (3.3).

Due to the recursive structure of the adaptation and filtering operations, the SAIIR architecture in Fig. 3 has a throughput bottleneck. In particular, it can be seen that the critical path for SAIIR has a computation time of

$$T_{serial} = [max(N_B, N_A) + 1]T_{A_1} + 2T_m + T_{A_2}, \quad (3.10)$$

where N_B and N_A are the number of taps in **FB** and **FA** blocks respectively, $max()$ generates the maximum of its arguments, T_{A_1} is the computation time of a two operand adder in the filtering blocks (i.e., **FB**, **FA** and **FC** blocks) and T_{A_2} is the computation time of the adder in the weight-update blocks (i.e., **WUDB** and **WUDA**). From (3.10), it is clear that for applications that require large values of N_B and N_A (such as echo cancellation), T_{serial} can be quite large and it may not be possible to meet the input sample rate requirements. This problem is solved in the next sub-section, where a pipelined AIIR architecture is presented.

3.2 The PIPAIIR Architecture

In order to derive the PIPAIIR architecture, we start with the SAIIR equations (3.1-3.4). The process of pipelining the SAIIR proceeds in two steps. First, we transform (3.1-3.4) such that it is applicable to the scattered look-ahead based pipelined system identification model of Fig. 2. This step will result in the pipelining of the **FC** block. Next, we apply relaxed look-ahead to the adaptive sections, which will result in the PIPAIIR architecture. Details of this derivation are omitted here and instead we present the PIPAIIR architecture.

The PIPAIIR architecture is described by

$$\mathbf{W}(n) = \mathbf{W}(n-D_2) + \mu \sum_{i=0}^{LA-1} e(n-D'_{11}-i)\mathbf{U}_1(n-D'_{11}-i), \quad (3.11)$$

$$e(n) = y(n) - \mathbf{W}^T(n-D_2)\mathbf{U}_1(n). \quad (3.12)$$

$$\hat{y}(n) = \mathbf{W}^T(n-D_2)\hat{\mathbf{U}}_1(n). \quad (3.13)$$

and (3.4), where $D'_{11} = D_1 + D'_1$ and furthermore

$$\mathbf{W}(n) = \begin{bmatrix} \mathbf{B}(n) \\ \mathbf{A}(n) \end{bmatrix} \quad (3.14)$$

$$\mathbf{B}^T(n) = [b_0(n), b_1(n), \ldots, b_{N_{B'}-1}(n)] \quad (3.15)$$

$$N_{B'} \leq N_A(D_3 - 1) + N_B \quad (3.16)$$

$$\mathbf{A}^T(n) = [a_1(n), a_2(n), \ldots, a_{N_A}(n)] \quad (3.17)$$

$$\mathbf{U}_1^T(n) = [x(n), ., x(n-N_{B'}+1), y(n-D_3), ., y(n-N_A D_3)] \quad (3.18)$$

$$\hat{\mathbf{U}}_1^T(n) = [x(n), ., x(n-N'_B+1), \hat{y}(n-D_3), ., \hat{y}(n-N_A D_3)] \quad (3.19)$$

In Fig. 4, we show the PIPAIIR architecture, where the D'_1 delays have been retimed such that they have been placed at the output of the **FB** block and **FA** block. This requires delaying $e_1(n)$ by an additional D'_1 delays. Note that the plant output $y(n)$ is delayed by $M = D'_1 + D_3$. This is because (3.19) introduces a latency of D_3 samples, which is now added to the D'_1 delays due to retiming. In a practical implementation, we would employ the D'_1 latches to pipeline the filter blocks **FB** and **FA**, while the D_1 latches would be used to pipeline the **WUDA** and **WUDB** blocks. As mentioned before, the D_3 latches can be employed to pipeline the **FC** block. Thus, we have achieved fine-grain pipelining of all the computational blocks in the AIIR filter.

3.3 Convergence Analysis

Convergence analysis of the PIPAIIR filter was also done in the same fashion as that for the pipelined LMS

algorithm [4]. The bound on μ for PIPAIIR is given by

$$0 \leq \mu \leq \frac{\alpha P + 2K - \sqrt{(\alpha P + 2K)^2 - 8K(K+1)}}{2K(K+1)\lambda_{av}}. \tag{3.20}$$

where $\alpha = \lambda_{rms}^2/\lambda_{av}^2$, $P = N_A + N_{B'} + 2$, $K = D_{11}'/D_2$, λ_{rms} and λ_{av} are the rms and average values of the eigenvalues of the covariance matrix of $\mathbf{U}_1(n)$. As K is increased from 1 to D_1, the bound on μ decreases. This fact has been confirmed via simulations. The misadjustment for PIPAIIR is given by

$$\mathcal{M} = \frac{\alpha N b}{2 - (\alpha P + 2K)b + K(K+1)b^2}. \tag{3.21}$$

As K is increased from unity towards D_1, the misadjustment would increase. In actual practice, the misadjustment does not change substantially as K varies and therefore can be considered to be approximately constant.

4 Simulation Results

In this section, we present simulation results to demonstrate the utility of PIPAIIR. In Experiment A, we consider the identification of a first-order plant using PIPAIIR and demonstrate the impact of pipelining on the SNR. In Experiment B, we demonstrate the use of sum relaxation to improve the convergence speed as the level of pipelining increases.

4.1 Experiment A

In this experiment, we consider the problem of identifying the plant

$$H(z^{-1}) = \frac{0.5z^{-1}}{1 - 0.86z^{-1}}. \tag{4.1}$$

The plant SNR, defined as the power of the input signal $x(n)$ to the power of additive noise $\eta(n)$ was 32 dB. All simulation results were averaged over 100 independent trials. Consider the problem of identification with an SAIIR architecture having $N_A = 1$ and $N_B = 1$.

Furthermore, assume that a two operand multiply takes $T_m = 32$ time units, a two operand filter block addition time is $T_{A1} = 20$ units and the weight-update block addition time is $T_{A2} = 40$ units. From (3.10), we can see that the SAIIR architecture can be clocked with a minimum clock-period of $T_{serial} = 144$ time units. Employing PIPAIIR with $D_1 = 5$, $D_1' = 5$, $D_2 = 2$, $D_3 = 3$, and $LA = 2$, results in an architecture that can be clocked with a minimum period of 20 time units.

In Table I, we summarize the effect of pipelining on $SNRE$ and $SNRE1$, where $SNRE$ and $SNRE1$ are the SNRs with respect to the adaptation error $e(n)$ and the estimation error $e_1(n)$ respectively. From Table I, it is clear that as speed-up increases both $SNRE$ and $SNRE1$ change slightly. Even for a speed-up of 8, the degradation in $SNRE1$ is less than 1 dB. Similar results were observed for higher speed-ups. Therefore, we can conclude that there is negligible degradation in performance for substantially high levels of pipelining.

4.2 Experiment B

As mentioned in the previous sub-section, the convergence speed of PIPAIIR (with $LA = 1$) slows down as D_2 increases from unity to $D_1 + D_1'$. This shown in Fig. 5, where the mean-squared adaptation error (MSE) for different speed-ups have been plotted.

As speed-up increases the number of samples required to converge also increases. The slower convergence of PIPAIIR can be rectified by using sum relaxation with $LA = 3$. As $LA = 3$ implies an increase in the computation time of the **WUDA** and **WUDB** blocks, hence we need to increase the value of D_1 from 5 to 10. From Experiment B, we know that the degradation in SNR is negligible for these parameter values. In Fig. 5, we have plotted the MSE curves for the a speed-up of unity (first row in Table I), 8 with $LA = 1$ (fourth row in Table I) and 8 with $LA = 3$ (fifth row in Table I). Note the remarkable improvement in convergence speed of PIPAIIR. Thus, PIPAIIR with $D_1 = 10$, $D_1' = 4$, $D_2 = 3$, $D_3 = 3$ and $LA = 3$ achieves a speed-up of 8 with no degradation in the misadjustment and some loss in the convergence speed.

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, C.-Y. Wang, and A. P. Brown, "Synthesis of control circuits in folded pipelined DSP architectures," *IEEE J. of Solid-state Circuits*, vol. 27, no. 1, pp. 29-43, January 1992.

[4] N. R. Shanbhag and K. K. Parhi, *Pipelined Adaptive Digital Filters*, Kluwer Academic Publishers, Boston, Massachusetts, 1994.

[5] C. R. Johnson, Jr., "Adaptive IIR filtering: Current results and open issues", *IEEE Trans. on Information Theory*, vol. IT-30, no. 2, pp. 237-250, March 1984.

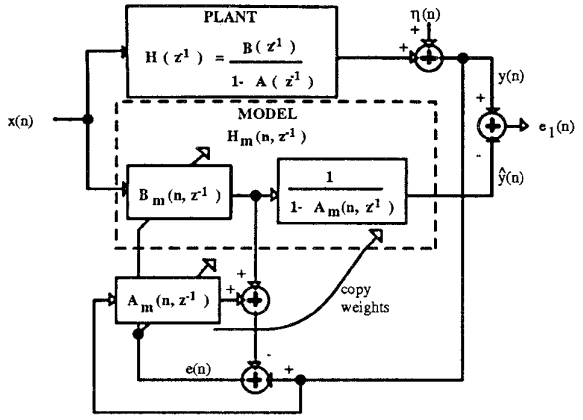

Figure 1. Conventional system identification set-up.

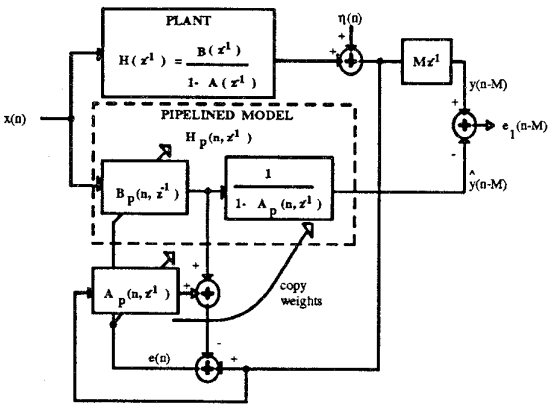

Figure 2. Pipelined system identification set-up.

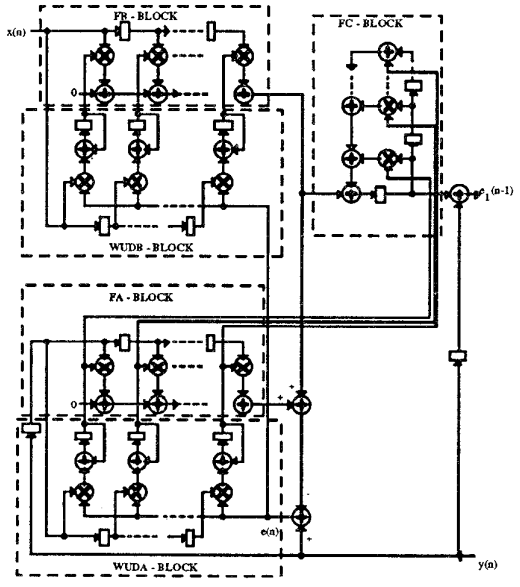

Figure 3. Serial AIIR filter architecture.

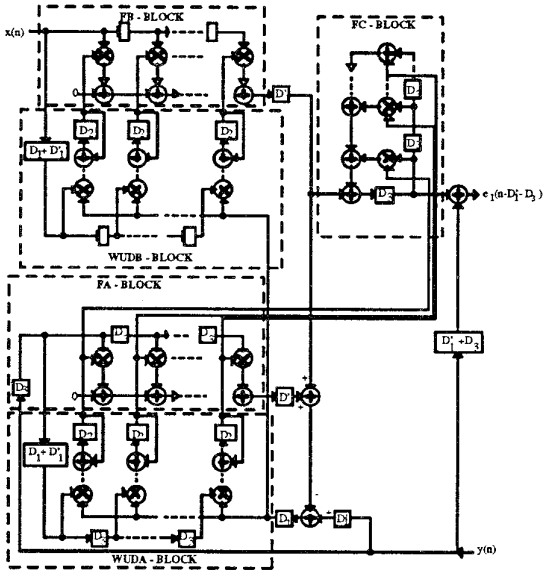

Figure 4. Pipelined AIIR filter architecture.

TABLE I

SU	D_1	D_1'	D_2	D_3	LA	N_B'	SNRE (dB)	SNRE1 (dB)
1	0	0	1	1	1	1	29.36	31.39
2	1	0	1	1	1	1	29.46	31.37
4	3	1	2	2	1	2	29.84	31.17
8	5	4	3	3	1	3	30.18	31.12
8	10	4	3	3	3	3	29.99	30.84

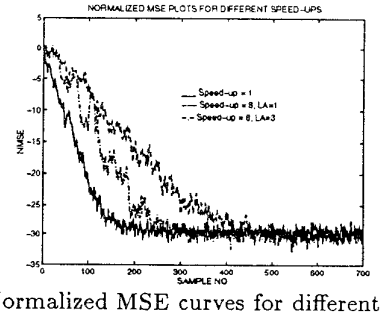

Figure 5. Normalized MSE curves for different speed-ups.