A PIPELINED ADAPTIVE LATTICE FILTER ARCHITECTURE[1]

N. R. Shanbhag K. K. Parhi

Dept. of Electrical Engineering
University of Minnesota
Minneapolis, MN 55455

Abstract

The stochastic gradient adaptive lattice filter is pipelined by the application of the *relaxed look-ahead*. This form of look-ahead maintains the functional behavour instead of the input-output mapping. The sum and product relaxations are employed to pipeline the lattice filter. The hardware complexity of the proposed pipelined filter is of the same order as the sequential filter. Thus, the new architecture is attractive from an implementation point of view. Convergence analysis is carried out to illustrate the trade-off offered by relaxed look-ahead. Linear prediction of a speech signal is used as a simulation example.

1 Introduction

Algorithm transformation techniques [1], for increasing the concurrency in digital signal processing architectures, have been well studied. Pipelining [2] and parallel processing [3] are two major techniques for developing high-speed digital signal processing architectures. Pipelining of fixed-coefficient filters via the *look-ahead computation* [2] has already been developed and has also been applied to pipeline adaptive lattice filters [4, 5]. The conventional look-ahead technique transforms a given serial algorithm into an equivalent (in the sense of input-output mapping) pipelined one. This, however, results in an enormous increase in the hardware complexity. In case of adaptive filters, this complexity is even higher due to the coefficient-update loop. For pipelining adaptive filters, it is possible to give up this exact input-output mapping if the average convergence behaviour is not changed substantially. In this paper, we use a *relaxed form of look-ahead* as opposed to the conventional look-ahead for pipelining the stochastic gradient lattice adaptive filter [6]. The relaxed look-ahead is an approximation to the conventional look-ahead which does not alter the convergence behaviour substantially.

The relaxed look-ahead has been succesfully employed to develop a pipelined LMS adaptive filter architecture [7] and the adaptive pulse-code modulation codec [8]. In this paper, we choose a particular stochastic gradient lattice adaptive algorithm [6] to pipeline via this technique. Note that relaxed look-ahead can be employed to pipeline any of the adaptive lattice algorithms in [6].

A brief discussion of the conventional look-ahead and its relaxed form is provided in section 2. In section 3, we derive the pipelined stochastic gradient lattice architecture (PIPSGLA). In section 4 discuss its convergence properties. An example of pipelining and speech prediction is presented in section 5.

2 The Relaxed Form of Look-ahead

We first illustrate the conventional look-ahead [2] with the help of an example. Consider the following first-order recursion,

$$x(n+1) = ax(n) + u(n), \tag{2.1}$$

where $x(n)$ is the output, $u(n)$ is the input and a is a constant. Iterating (2.1) M_1 times, we get

$$x(n+M_1) = a^{M_1}x(n) + \sum_{i=0}^{M_1-1} a^i u(n+M_1-1-i). \tag{2.2}$$

Equation (2.2) represents the application of look-ahead to (2.1). Due to the nature of the transformation both (2.1) and (2.2) represent the same system. However, (2.2) can process data M_1 times faster than (2.1). This is because the M_1 latches introduced into the system can be used to pipeline the multiply-add operation. The second term in (2.2) represents the extra hardware overhead necessary. This overhead can be quite expensive for higher order systems and for fine-grained pipelining (i.e. large values of M_1). However, if (2.1) were an adaptation algorithm, then we may apply the *sum relaxation* to write the second term in (2.2), as follows

$$x(n+M_1) = a^{M_1}x(n) + M_1u(n+M_1-1). \tag{2.3}$$

The system described by (2.3) would be a close approximation of the system in (2.2) if a is close to unity and $u(n)$ were approximately constant. The a^{M_1} term in (2.3) can be precomputed if a is a constant. If a in (2.3) is time-varying (i.e. we have $a(n)$ instead of a), but the magnitude of $a(n)$ is close to unity, then we replace $a(n)$ by $(1-a'(n))$, where $a'(n)$ is close to zero. Then, we apply the *product relaxation* to write (2.3) as

$$x(n+M_1) = (1-M_1a'(n))x(n) + M_1u(n+M_1-1). \tag{2.4}$$

[1]This research was supported by the army research office by contract number DAAL03-90-G-0063.

0-7803-0593-0/92 $3.00 1992 IEEE

Thus, (2.3) and (2.4) represent possible application of relaxed look-ahead to (2.1). Depending on the application at hand, different types of approximations may be employed. Unlike (2.2), (2.3-2.4) are not equivalent to (2.1) with respect to its input-output mapping. Therefore, in case of adaptive algorithms, convergence analysis of (2.3-2.4) needs to be done. In general, the relaxed form of look-ahead is not an algorithm transformation technique in the usual sense [1]. This is because it modifies the algorithm to which it is applied. However, it can be considered as an algorithm transformation technique in the stochastic sense if the average output profile is maintained.

3 The PIPSGLA Architecture

As usual, the indices n and m denote the time instance and the lattice stage number. The serial stochastic gradient lattice algorithm (SSGLA) chosen for pipelining is described as follows [6]

$$k_m(n+1) = [1-\beta_m(n)P(n|m-1)]k_m(n) + 2\beta_m(n)Q(n|m-1) \quad (3.1)$$

$$S(n+1|m-1) = (1-\beta)S(n|m-1) + P(n|m-1) \quad (3.2)$$

$$\beta_m(n) = \frac{1}{S(n|m-1)} \quad (3.3)$$

$$\epsilon_f(n|m) = \epsilon_f(n|m-1) - k_m(n)\epsilon_b(n-1|m-1) \quad (3.4)$$

$$\epsilon_b(n|m) = \epsilon_b(n-1|m-1) - k_m(n)\epsilon_f(n|m-1) \quad (3.5)$$

where

$$P(n|m) = \epsilon_f^2(n|m) + \epsilon_b^2(n-1|m) \quad (3.6)$$

$$Q(n|m) = \epsilon_f(n|m)\epsilon_b(n-1|m). \quad (3.7)$$

The reflection coefficient $k_m(n)$ is updated according to (3.1), while the input power estimate $S(n|m)$ is updated using (3.2). Thus, (3.1) and (3.2) are the two recursive loops in the algorithm. The equations (3.3) and (3.4) are the order-updates. The SSGLA architecture (Fig.1) shows a computation time (T_c) of

$$T_c = 4(T_m + T_a), \quad (3.8)$$

where T_m and T_a are two operand multiply and add times. The loop computation time T_L is

$$T_L = T_m + T_a. \quad (3.9)$$

The pipelining of SSGLA is done by first inserting interstage pipelining latches. This step is trivial as the stages are connected in a non-recursive fashion. If the desired speed-up is not achieved, then the two recursive loops need to be pipelined. Let M_S and M_L represent the number of interstage and loop pipelining latches, respectively. To do this we apply relaxed look-ahead to (3.1) and (3.2). Like (2.1), (3.2) is a first order recursion. Therefore, applying the sum relaxation (see (2.3)) we can simply write down the final equation by inspection

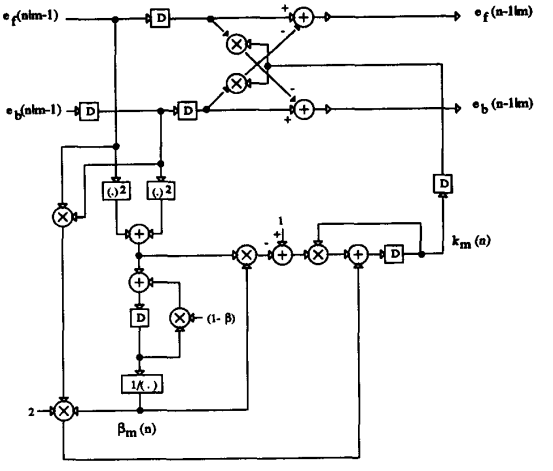

Figure 1: The SSGLA architecture.

$$S(n'+M_L) = (1-\beta)^{M_L}S(n'|m-1) + P(n'|m-1). \quad (3.10)$$

where $n' = n - M_S$. Similarly, we can show that application of the product and sum relaxations (see (2.4)) to (3.1) results in the following

$$k_m(n'+M_L) = [1 - M_L\beta_m(n')P(n'|m-1)]k_m(n') + 2M_L\beta_m(n')Q(n'|m-1) \quad (3.11)$$

Note that the loop multiplier coefficients in (3.1) and (3.2) are close to unity. Therefore, the form of relaxation used to derive (3.10) and (3.11) is justified. The rest of the equations describing PIPSGLA are identical to SSGLA and hence are not provided.

The complete PIPSGLA is shown in Fig.2, where it can be seen that the hardware increase is two multipliers per stage and the pipelining latches (see Fig.2). This is remarkably lower than that obtained by conventional look-ahead [4, 5].

4 Convergence Analysis

The convergence analysis, for a single lattice stage, was carried out under the independence assumptions [6]. These assumptions become truer as the number of lattice stages increases. The inputs to the stage under consideration is assumed to be stationary. The results of this analysis are presented below. The bounds on β for convergence of weights is

$$0 \le \beta \le \frac{2}{M_L^2}, \quad (4.1)$$

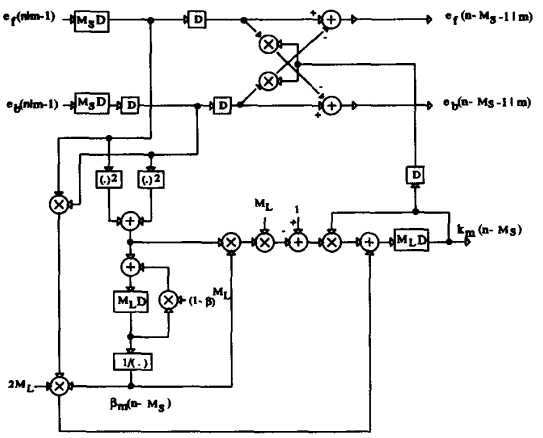

Figure 2: The PIPSGLA architecture.

which is tighter than that of SSGLA (for which the upper bound is 2). However, for most applications the value of β is smaller than the upper bound.

The convergence time-constant for PIPSGLA (τ_{PIP}) equals

$$\tau_{PIP} = \frac{1}{2\beta M_L} \tag{4.2}$$

and this value is $1/M_L$ times that of SSGLA. This is to be expected as (3.11) has an effective adaptation constant which is M_L times that of SSGLA in (3.1).

The misadjustment for PIPSGLA ($\mathcal{M}_{PIP}$) is given by

$$\mathcal{M}_{PIP} = \frac{M_L\gamma(1-k_{m,opt}^2)}{(2-M_L\gamma)(2+k_{m,opt}^2)} \tag{4.3}$$

where $\gamma = 1-(1-\beta)^{M_L}$. Assuming β is much smaller than unity, (4.3) implies that $\mathcal{M}_{PIP}$ is M_L^2 times that of SSGLA.

5 Examples

In this section, we first present a pipelining example to demonstrate the increase in speed. Then, we employ speech prediction to compare the performance of SSGLA and PIPSGLA.

5.1 Pipelining Example

To illustrate the speed-up due to pipelining, we assume $T_a = 20$, $T_m = 40$ and the filter order $N = 2$. Therefore, from (3.8), it follows that $T_c = 240$. For a speed-up of $M = 6$, the clock-period of PIPSGLA should be 40. This can be achieved with $M_L = 2$ and $M_S = 6$. Retiming these latches (Fig.3) results in the desired clock-period.

5.2 Speech Prediction Example

A speech signal corresponding to the word 'zoos', was sampled at 8 kHz for a duration of 1 s. The scaled version

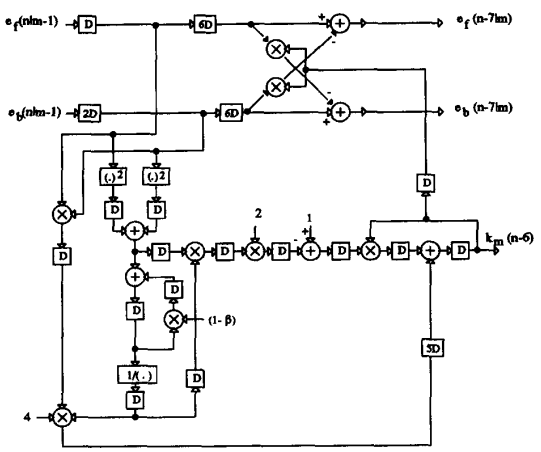

Figure 3: The pipelining example.

of this signal (see Fig.4) was input to SSGLA and PIPSGLA (the same as in Fig.3). The value of β for SSGLA and PIPSGLA was 0.005.

The mean-squared error (MSE) plots for SSGLA (Fig.5) and PIPSGLA (Fig.6) are very similar. However, due to the faster convergence time constant of PIPSGLA, it tracks the highly non-stationary sections (sample numbers 1300-1500 and 2900-3100) better than SSGLA. The signal-to-prediction noise ratio (SNR) for SSGLA was 12.4087 dB, while that for PIPSGLA was 12.7047 dB. Note that PIPSGLA operates 6 times than SSGLA. The prediction MSE for a simulation with $M_S = 18$ and $M_L = 5$, which corresponds to a speed-up of $M = 20$, is shown in Fig.7. The SNR in this case was 12.6757 dB.

Thus, we see that large speed-ups are achievable without

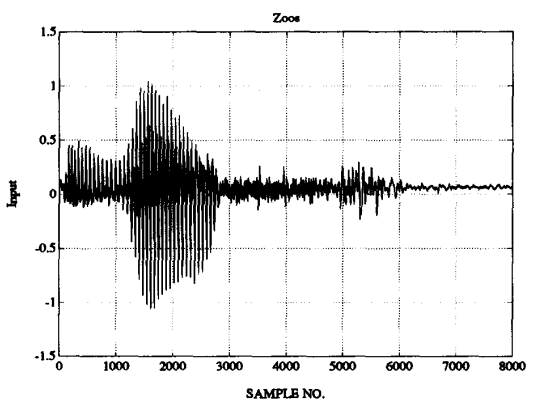

Figure 4: The input speech signal.

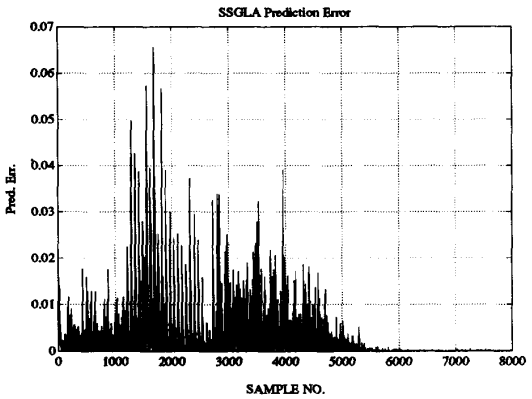

Figure 5: Prediction MSE for SSGLA.

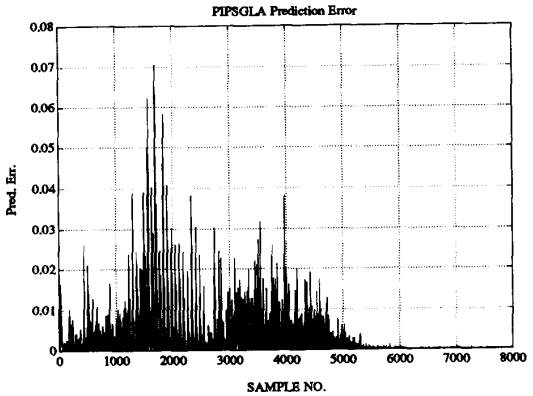

Figure 6: Prediction MSE for PIPSGLA with speed-up $M = 6$.

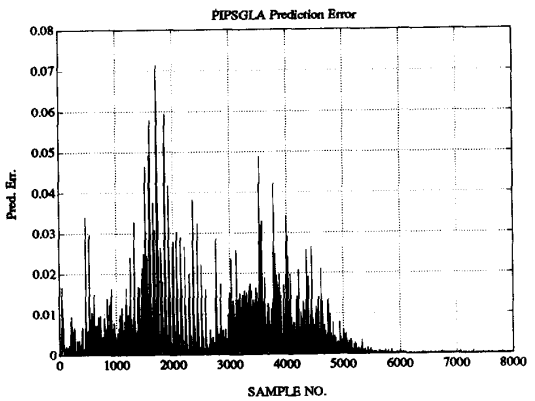

Figure 7: Prediction MSE for PIPSGLA with speed-up $M = 20$.

any appreciable degradation in the convergence behavior.

6 Summary and Conclusions

The stochastic gradient lattice filter has been pipelined via the application of *relaxed look-ahead* [7]. The relaxed look-ahead results in an enormous savings in hardware. As it is not a one-to-one mapping like its deterministic counterpart, the relaxed look-ahead offers a multitude of architectural possibilities. The PIPSGLA is one such choice but the user may select any other form of relaxation depending on the application at hand. Future work is being directed towards the design of inherently pipelinable adaptive decision feedback equalizers and adaptive pulse code modulation [8].

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 and D.G. Messerschmitt, "Pipeline interleaving and parallelism in recursive digital filters - Part II : Pipelined incremental block filtering", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 37, pp. 1118-1134, July 1989.

[4] K.K. Parhi and D.G. Messerschmitt, "Concurrent cellular VLSI adaptive filter architectures", *IEEE Trans. on Circuits and Systems*, vol. 34, pp. 1141-1151, Oct. 1987.

[5] T. H.-Y. Meng and D.G. Messerschmitt, "Arbitrarily high sampling rate adaptive filters", *IEEE Trans. on Acoustics, Speech and Signal Proc.*, vol. 35, pp. 455-470, April 1987.

[6] M.L. Honig and D.G. Messerschmitt, *Adaptive Filters : Structures, Algorithms, and Applications*. Boston: Kluwer, 1984.

[7] N.R. Shanbhag and K.K. Parhi,"A pipelined LMS adaptive filter architecture", *Proc. Asilomar Conf. on Sig., Syst. and Comput.*, Nov. 1991, Pacific Grove (CA).

[8] N.R. Shanbhag and K.K. Parhi,"A high-speed architecture for ADPCM codec", *Proc. IEEE Intl. Symp. on Circuits and Systems*, San Diego, 1992.