Example 2: Consider a non-symmetric low-pass complex FIR log filter specified by

\[
20 \log_{10} D(\omega) = \begin{cases}
-30 - j12\omega, & -\pi \leq \omega \leq -0.24\pi, \\
-j12\omega, & -0.24\pi \leq \omega \leq 0.3\pi, \\
-30 + j12\omega, & 0.44\pi \leq \omega \leq \pi,
\end{cases}
\]

which is approximated with \( N = 30 \) by the eigefilter technique. The resultant amplitude response and group delay after 9 iterations are shown in Fig. 2(a) and (b), when \( W_s = 5, W_c = 6, \epsilon_p = \epsilon_c = 0.02 \) and \( \epsilon = 0.03 \) are preassigned. Notice that the peak delay error is larger in the stopband because the delay response is more sensitive when the magnitude is smaller. Fig. 2(c) and (d) present the traces of complex log errors in passband and stopband respectively, which demonstrate that the complex log errors are equiripple. The peak absolute log error in the passband and stopband are 0.04136 and 0.05109, respectively.

IV. CONCLUSION

An effective method has been proposed for the design of linear-phase and nonlinear-phase FIR log filters in this paper. The method is based on the eigenvector computation of an appropriate positive-definite matrix which is real symmetric or complex Hermitian symmetric, hence it is called by the “eigefilter approach.” To design log filters with optimal equiripple real/complex log errors in the real/complex Chebyshev sense, the iterative weighted eigefilter approach is applied. The weighting function is chosen in such a way that the weighted error minimizes the real/complex logarithmic error, and is updated using the result of the previous iteration, which forms a powerful approach for designing FIR log filters. Another alternative least-squares approach, the weighted least squares algorithm [12], can also be used to design FIR log filters in an analogical manner, and the results are similar to those of the given examples.

REFERENCES


Finite-Precision Analysis of the Pipelined ADPCM Coder

Naresh R. Shanbhag and Keshab K. Parhi

Abstract—Roundoff error analysis of a pipelined adaptive differential pulse code modulation (ADPCM) coder is presented. The pipelined coder has been developed by employing the relaxed look-ahead technique. It is shown that the precision of the quantized prediction error and those of the predictor coefficients are critical and analytical expressions for lower bounds on these precisions are derived. In addition, an expression for the excess mean-squared prediction error due to finite-precision is also derived. Based on these expressions, a systematic method for assigning precision to various signals in the coder is developed. Simulation results with real image data confirm the results of this analysis.

I. INTRODUCTION

Pipelined digital signal processing (DSP) algorithms have a much lower hardware overhead as compared to other high-speed techniques. They are, therefore, attractive from an integrated circuit (IC) implementation point of view. Recently, we proposed a fine-grain pipelined adaptive differential pulse code modulation (ADPCM) algorithm in [1] using the technique of relaxed look-ahead [2]. This technique is an approximation of the look-ahead technique [3], which had been proposed for pipelining recursive digital filters. The result of applying relaxed look-ahead is a pipelined algorithm with minimal hardware overhead and a negligible performance degradation. Therefore, this algorithm is very attractive from IC implementation point of view.

Finite-precision analysis is always necessary before an IC implementation is done. Roundoff error analysis of the basic LMS has been presented in [4]. There are, however, significant differences between the basic LMS predictor and the ADPCM. This is because, in addition to the adaptation feedback, the ADPCM has an inherently recursive structure. Additionally, it also contains a quantizer, which is a nonlinear element. Therefore, it is important to analyze the roundoff error of the pipelined ADPCM (or PIPADPCM).

In this paper, we present the roundoff error analysis of the PIPADPCM coder [1] and verify it through simulations on real image data [5]. In section II, we describe the relaxed look-ahead technique and PIPADPCM coder. The roundoff error analysis of the pipelined coder is presented in section III, the results of which are verified in section IV.
II. RELAXED LOOK-AHEAD

Consider the first-order recursion given by

\[ x(n) = x(n-1) + a(n)u(n). \]  

(2.1)

The computation time of (2.1) is lower bounded by a single add time.

In order to reduce this lower bound, we apply an M-step look-ahead transformation [3] to (2.1), which leads to the following equation:

\[ x(n) = x(n-M) + \sum_{i=0}^{M-1} a(n-D_l-i)u(n-D_l-i). \]

(2.2)

This transformation results in M latches being introduced into the recursive loop, which in turn can be retimed [6] to pipeline the last operation to any desired level. Note that this transformation did not alter the input-output behavior and therefore resulted in a hardware overhead (the second term in (2.2)).

Relaxed look-ahead involves the formulation of various approximations (or relaxations) to reduce the overhead due to look-ahead. The delay relaxation involves approximating (2.2) by

\[ x(n) = x(n-M) + \sum_{i=0}^{M-1} a(n-D_l-i)u(n-D_l-i), \]

(2.3)

which is based on the assumption that the product \( a(n)u(n) \) is more or less constant over \( D_l \) samples. Under the same assumption, we can approximate (2.2) as

\[ x(n) = x(n-M) + \frac{M}{LA} \sum_{i=0}^{LA-1} a(n-i)u(n-i), \]

(2.4)

where we have kept \( LA (LA \leq M) \) terms from the summation in (2.2). This approximation is referred to as sum relaxation. Note that the additional factor \( M/LA \) is necessary to make the average output profile of (2.4) identical to that of (2.1) for stationary \( a(n) \) and \( u(n) \).

The delay relaxation and the sum relaxation have been employed to pipeline the least mean-squared (LMS) algorithm [7]-[8] in [1]. The resulting pipelined LMS algorithm was then analyzed for convergence and also employed to develop the PIPADPCM coder. The interested reader is referred to [1] for details of this derivation. The PIPADPCM coder (see Fig. 1), with an \( N^{th} \) order predictor and operating on a \( N_{cols} \times N_{rows} \) image frame, is described by

\[ \hat{s}(n+M_1) = W^T(n-M_2)\hat{S} \left( n - N_{cols} + M_1 + \frac{N-1}{2} \right) \]

(2.5a)

\[ W(n) = W(n-M_2) + \mu \sum_{i=0}^{LA-1} e_q(n-i) \hat{s}(n - N_{cols} + \frac{N-1}{2} - i) \]

(2.5b)

\[ e(n) = s(n) - \hat{s}(n) \]

(2.5c)

\[ e_q(n) = Q[e(n)] = e(n) + q(n) \]

(2.5d)

\[ \hat{s}(n) = \hat{s}(n) + e_q(n) \]

(2.5e)

where \( s(n) \) is the raster scan input formed from the image pixels, \( \hat{s}(n) \) is the reconstructed signal, \( s(n) \) is the predicted value of \( s(n) \), \( e(n) \) is the prediction error, \( e_q(n) \) is the quantized value (Q[.] representing the quantization operator with \( R \) bits), \( q(n) \) is the quantization error, \( W(n) \) is the predictor coefficient vector and \( \mu \) is the adaptation step-size. \( M_1 \) and \( M_2 \) are the pipelining levels of the ADPCM and the adaptation recursions, respectively. From (2.5a) and (2.5b), we observe that the least mean-squared (LMS) algorithm [7]-[8] is being employed to update the predictor coefficients.

III. ROUNDOFF ERROR ANALYSIS

The roundoff error analysis presented in this section proceeds along lines similar to that of the LMS algorithm in [4]. Employing the notation in [4], we denote the quantized version of a variable \( x \) by \( x' \), i.e.,

\[ x' = x + \delta x \]

(3.1)

where \( \delta x \) is the quantization error. In addition, the number of bits employed in the finite-precision implementation of a variable \( x \) is denoted by \( B(x) \). Therefore, the quantization error power \( (\sigma_{x'}^2) \) equals \( 2^{-2B(x)} / 12 \).

With the notation of (3.1), we can describe the finite-precision PIPADPCM (see Fig. 2) coder as follows

\[ \hat{s}'(n+M_1) = W'^T(n-M_2)\hat{S}'(n_1 + M_1) + \delta 1(n) \]

(3.2a)

\[ W'(n) = W'(n-M_2) \]

(3.2b)

\[ + \mu \sum_{i=0}^{LA-1} e_q'(n-i)\hat{s}'(n_1 - i) + \delta 2(n) \]

(3.2c)

\[ e'_q(n) = Q[e'(n)] = e'(n) + q(n) \]

(3.2d)

\[ \hat{s}'(n) = \hat{s}'(n) + e'_q(n) + \delta 4(n) \]

(3.2e)

where \( n_1 = n - N_{cols} + \frac{N-1}{2} \) and \( \delta 1(n), \delta 2(n), \delta 3(n) \) and \( \delta 4(n) \) are the roundoff errors made in the computation of (3.2a), (3.2b), (3.2c) and (3.2e), respectively.

To make the analysis tractable, we approximate (3.2b) as follows

\[ W'(n) = W'(n-M_2) + \mu LA e'_q(n+M_1)\hat{s}'(n_1 + M_1) + \delta 2(n), \]

(3.3)

which is a reasonable approximation to make if the gradient changes slowly over \( M_1 + LA - 1 \) samples.

Substituting for \( \hat{s}'(n) \) from (3.2c) and \( e'_q(n) \) from (3.2d) into (3.2e) and after some manipulation, we obtain

\[ \hat{s}'(n) = \hat{s}'(n) + \gamma(n) \]

(3.4)
where \( y(n) = s(n) + \delta s(n) + \delta 3(n) + \delta 4(n) \) represents the additional quantization noise in the reconstructed output due to finite-precision implementation. In order to minimize \( \sigma^2_q \) we have to assign sufficient precision to \( e_r(n) \). Clearly, it is necessary to have \( \sigma^2_q \) much smaller than \( \sigma^2_s \). In other words

\[
\Delta Q = 2^{-\alpha} \sigma^2 = 2^{-\alpha} \left[ \frac{1}{2} \Delta Q \right]^2,
\]

(3.5)

where \( \Delta Q \) is the dynamic range of \( Q \), \( 2^N \) is the number of quantizer levels and \( \alpha \) is a sufficiently large positive constant. Note that we have assumed a uniform quantizer in (3.5). This results in the following lower bound on \( B(e_r) \)

\[
B(e_r) \geq -\log_2 \left( \frac{\Delta Q}{2^{2\alpha}-1} \right) + \frac{\alpha}{2} - 1.
\]

(3.6)

To determine the precision of the predictor coefficients \( W(n) \), consider (3.3). For the weights to continue to adapt, it is necessary that the term being added to \( W(n-M_2) \) (in (3.3)) be greater than \( 2^{-B(W)-1} \). Otherwise, due to roundoff, this term will be quantized to zero. Hence, we get the following condition

\[
\mu^2 LA^2 E[e^2_r(n)] E[\hat{s}^2(n_1)] \geq 2^{-B(W)-1},
\]

(3.7)

where \( E[\cdot] \) is the expectation operator. From (3.4), we have

\[
E[\hat{s}^2(n_1)] = \sigma^2_s + \sigma^2_q + \sigma^2_s + \sigma^2_s.
\]

(3.8)

The computation of \( E[e^2_r(n)] \) is done as follows

\[
E[e^2_r(n)] = \frac{1}{2R} \sum_{i=0}^{2R-1} c^2_i,
\]

(3.9)

where \( c_i \) are the quantizer codewords. Substituting (3.8) and (3.9) into (3.7) and neglecting \( \sigma^2_s \), we obtain the following lower bound on \( B(W) \)

\[
B(W) \geq -\log_2(\mu LA) - 2^{-B(W)-1} \log_2(2^{2R}-1).
\]

(3.10)

Finally, in Appendix A, we derive an estimate of the excess mean squared error due to the quantization of various signals. Following the development in [4], we get the following expression for the mean-squared prediction error for the finite-precision PIPADPCM:

\[
E(e^2(n)) = E(\hat{s}^2(n)) + \left( \left| W_{opt} \right|^2 \sigma^2_s + \frac{1}{2} \mu LA \sigma^2 \right),
\]

\[
+ \mu^2 LA^2 \left[ (\left| W_{opt} \right|^2 \sigma^2 + \sigma^2) \mu(R) + N \sigma^2 (J_{min} + \sigma^2) \right],
\]

\[
2\mu LA - \mu^2 LA^2 \mu(R).
\]
where $W_{opt}$ is the optimal predictor coefficient values, $J_{min}$ is the mean-squared prediction error when $W = W_{opt}$, $\sigma_j^2 = \sigma_{e_j}^2 + \sigma_{t_j}^2 + \sigma_{s_j}^2$ and $\text{tr}(R)$ denotes the trace of the autocorrelation matrix of $\hat{s}(n)$.

The terms being added to $E(\hat{e}^2(n))$ in (3.11) are the additional noise terms due to signal quantization. By satisfying (3.6) and providing sufficient precision to $\hat{s}(n)$, $\hat{s}(n)$ and $\hat{e}(n)$, we can make the second and the last three terms in (3.11) negligible. More interesting are the third and fourth terms. As $\mu$ decreases the third term is also decreased, while the fourth term increases. Clearly, an optimal value of $\mu$ exists at which the sum of these two terms is minimized. This optimal value of $\mu$ can be shown to be equal to

$$\mu_{opt} = \sqrt{\frac{N \sigma_{e_j}^2}{\text{tr}(R)}} [1 + \sigma_{e_j}^2 + \text{tr}(R)] + N \sigma_{e_j}^2 (J_{min} + \sigma_{e_j}^2)$$

(3.12)

On the basis of (3.6), (3.10) and (3.12), we now provide a systematic method to assign precision to various signals in a finite-precision implementation of the ADPCM codec.

**Step 1** Given values of $\Delta Q$, $R$ and $\alpha$ employ (3.6) to obtain $B_1(\hat{e})$. Let $B(\hat{s})$, $B(\hat{s})$, and $B(e)$ be equal to $B_1(e)$.

**Step 2** Employ (3.12) to obtain $\mu_{opt}$.

**Step 3** Choose a value of $\mu$ close to $\mu_{opt}$ and employ (3.10) to obtain $B(W)$.

**IV. SIMULATION RESULTS**

In this section, we present simulation results to verify the analysis of the previous section. For this purpose, we consider the coding of the image of 'Lenna' (see Fig. 3) with a frame size of 256 x 256 and with 256 gray levels scaled to lie between -1 and +1.

By extensive simulations, we found that $\alpha = 4$ (in (3.6)) is sufficient. In Fig. 4, we plot the output signal-to-noise ratio (SNR) for different values of $B(e)$. The two plots are for $R = 3$ and $R = 5$ with $\Delta Q = 0.25$ for both. The dashed lines in Fig. 4 represent the signal-to-noise ratios (SNRs) of the corresponding infinite-precision implementations. With the values of $R = 3, 5$, $\Delta Q = 0.25$ and $\alpha = 8$, (3.6) predicts $B(e_3) \geq 5.8$ and $B(e_5) \geq 7.95$ for $R = 3$ and $5$, respectively. This is also indicated quite accurately by the corresponding SNR plots in Fig. 4.

Next, we compare the output SNR for different values of predictor coefficient precision. The values of $R = 3$ and $\Delta Q = 0.25$ were employed. The precision of all other signals was fixed at 8 as suggested by (3.6) and Fig. 4. In Fig. 5, we plot the output SNRs for $\mu = 2^{-6}$ and $\mu = 2^{-5}$. Again, the dashed lines represent the SNR of the corresponding infinite-precision implementation. For $\mu = 2^{-5}$ and $2^{-6}$, (3.10) predicts a lower bound of 10 and 11, respectively. This is verified experimentally in Fig. 5. The importance of the predictor coefficient precision is evident.

We now compare perceptually the reconstructed outputs of the finite-precision and the infinite-precision PIPADPCM coders. As suggested by (3.6) and (3.10), we fix the precision of predictor coefficients $B(W) = 15$ and that of all other signals at 8. The values of $\mu = 2^{-6}$, $\Delta Q = 0.25$ and $R = 3$ were chosen for all the simulations in this experiment. In Figs. 6 (a)-(b), we show the reconstructed outputs of the PIPADPCM with $M_1 = 0$, $M_2 = 1$ and $LA = 1$. These values of $M_1$ and $M_2$ result in a speed-up of approximately 40. Again, the reconstructed outputs (see Figs. 7 (a)-(b)) of the finite and infinite precision PIPADPCM algorithms are perceptually identical. Thus, we verify that the performance...
of the finite-precision PIPADPCM is close to its infinite-precision counterpart if the signal precisions are obtained from (3.6) and (3.10).

Note also that the PIPADPCM outputs in Fig. 7 are similar to the corresponding ones in Fig. 6. This indicates that the increase in the speed has been achieved without any loss of performance.

APPENDIX A: DERIVATION OF THE EXPRESSION FOR THE MSE

In the derivation below, we neglect second and higher order error terms. Also, we assume that all quantization errors are uncorrelated and independent of the signals themselves.

Substituting for $\delta'(n)$ from (3.2(a)) into (3.2(c)), we obtain

$$e'(n) = e(n) + \delta s(n) - W^T(n-M)\hat{S}(n),$$

where $M = M_1 + M_2$. Next, we substitute for $W'(n)$ and $\delta'(n)$ in terms of their infinite-precision variables (see (3.1)) to obtain

$$e'(n) = e(n) - W^T(n-M)\Gamma(n_1) - \delta W^T(n-M)\hat{S}(n),$$

where $\Gamma(n_1) = [\gamma(n_1), \gamma(n_1-1), ..., \gamma(n_1-N+1)]^T$ and $\gamma(n)$ (see (3.4)) is the quantization error in $\delta(n)$. Squaring and taking the expected value of (A.2) we get

$$E[\epsilon'^2(n)] = E[\epsilon^2(n)] + E[W^T(n-M)W(n-M)]\sigma^2$$

In (A.3), the first term is the MSE of the infinite-precision PIPADPCM and the rest are the excess MSE due to the finite-precision effects. We now derive analytic expressions for the second and the third terms in (A.3).

The second term in (A.3) can be written as

$$E[W^T(n-M)W(n-M)]\sigma^2$$

$$= E[V(n-M) + W_{opt}^T(V(n-M) + W_{opt})]\sigma^2$$

where $V(n)$ is the weight error vector. Employing the analysis in [8, Appendix D(B)], it is easy to show that

$$E[V(n)T\hat{V}(n)] = \frac{1}{2}\mu LA_{min}N. \quad (A.5)$$

Thus, we can write (A.4) as

$$E[W^T(n-M)W(n-M)]\sigma^2 = (E[V(n-M)T\hat{V}(n-M)] + |W_{opt}|^2)\sigma^2$$

$$= \left(\frac{1}{2}\mu LA_{min}N + |W_{opt}|^2\right)\sigma^2$$

(A.6)

To obtain an analytic expression for the third term in (A.3) we rewrite it as follows

$$E[6W^T(n-M)\hat{S}(n)\hat{S}^T(n)6W(n-M)]$$

$$= tr[6W^T(n-M)\hat{S}(n)\hat{S}^T(n)6W(n-M)]\hat{R}]$$

(A.7)

where $tr[.]$ represents the trace of a matrix.

Next, we substitute for $\epsilon'(n)$ from (A.2) into (3.2(d)) to get

$$e'(n) = e(n) - W^T(n-M)\Gamma(n_1) - \delta W^T(n-M)\hat{S}(n)$$

$$+ \delta s(n) - \delta(n-M_1) + \delta s(n) + q(n) + \delta q(n)$$

$$= \epsilon(n) - W^T(n-M)\Gamma(n_1) - \delta W^T(n-M)\hat{S}(n)$$

$$+ \beta(n). \quad (A.8)$$