THE BELL SYSTEM TECHNICAL JOURNAL
Vol. 60, No. 7, September 1981
Printed in U.S.A.

Digital Signal Processor:

Sub-Band Coding

By R. E. CROCHIERE

(Manuscript received July 8, 1980)

This paper explores the use of the Bell Laboratories digital signal processing integrated circuit for digitally encoding speech or audio signals based on the sub-band coding technique. Sub-band coding represents a next level in algorithmic complexity over that of adaptive differential pulse-code modulation, discussed in a companion paper, and it has a corresponding advantage in performance. We discuss the details of a real-time, two-band sub-band coding implementation on the digital signal processor. We then comment on how this approach can be extended to more than two band designs for greater bit rate compression capability. In connection with this, we also consider some general issues involved in implementing multirate signal processing algorithms of this type on the digital signal processor.

I. INTRODUCTION

Digital encoding of speech and audio has been a topic of long-standing interest for purposes of digital communications and digital storage[1-3]. The efficiency of such encoding techniques depends strongly on the degree to which the bit rate can be reduced (compressed) without impairing the quality of the decoded signal. Typically, signals such as speech and audio have a high degree of redundancy that can be used to reduce this bit rate. Also, properties of human perception can be used to reduce the bit rate without impairing the quality of the decoded signal.

To take advantage of these properties, a considerable amount of signal processing is necessary. Thus, in the past many of these techniques have only been implemented by non-real-time computer simulations or with the aid of highly specialized digital hardware. This

picture is now rapidly changing, as is exemplified by the recent Bell Laboratories digital signal processing integrated circuit (DSP).[4,5] With this device it is possible to conveniently implement, in real time, signal processing algorithms of low to medium complexity. Thus, a single DSP integrated circuit can be used to implement many of the simpler encoding algorithms and multiple DSPs can be used for some of the more complex algorithms.

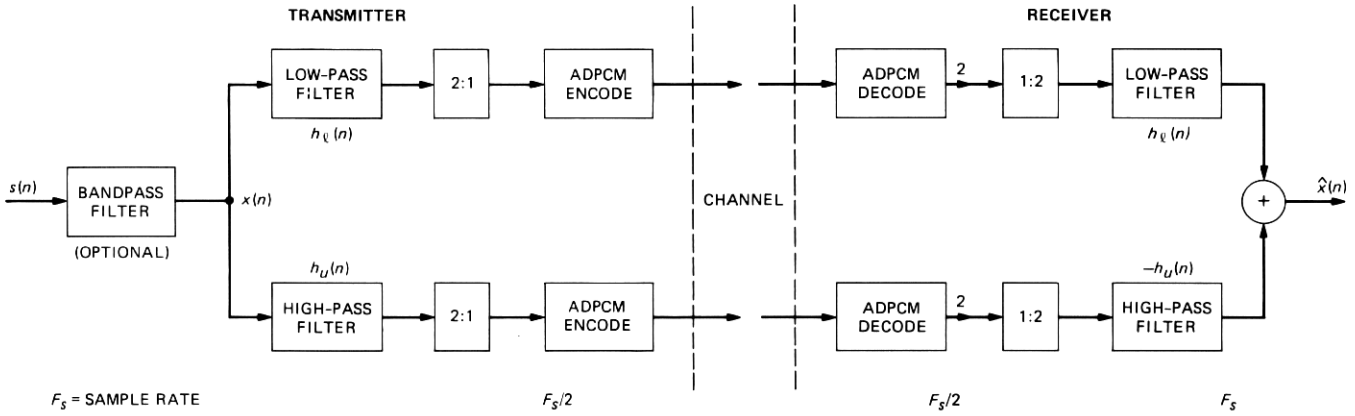
In a companion paper,[6] it is shown that the ADPCM (adaptive differential PCM) encoding algorithm, which offers a bit rate reduction factor of approximately two over conventional logarithmic companded PCM encoding (for speech), can be efficiently implemented on the DSP, and that it uses only about one-quarter of the processing capability of the device. In this paper, we report on continuing efforts towards a next level of complexity of encoding techniques on the DSP. In particular, we discuss the technique of sub-band coding (SBC).[3,7,8] Our efforts focus primarily on a two-band sub-band coder design which demonstrates the capability of the DSP for this class of algorithms. By extension of these same techniques, it is expected that more complex SBC designs on the DSP (e.g., four or more bands) with greater bit-rate compression capability will also be possible and efforts are continuing in this direction.

II. THE SUB-BAND CODER ALGORITHM

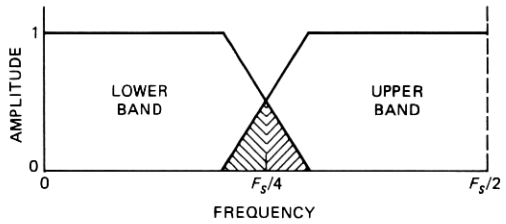
Figure 1 reviews the basic conceptual configuration for a two-band SBC design. The input signals $s(n)$ is assumed to be in digital (linear PCM) form and it may be (optionally) filtered with a bandpass prefilter for reasons to be discussed later. The output signal $x(n)$ is then divided into two equally spaced frequency bands by low-pass and high-pass filters, $h_l(n)$ and $h_u(n)$, respectively. Each sub-band signal is reduced in sampling rate by a factor of two, i.e. if F_s is the sampling rate of the input signal, $F_s/2$ is the sampling rate of the sub-band signals. The sub-band signals are then encoded with ADPCM encoders and the output bits are multiplexed for storage or transmission.

In the receiver, the sub-band signals are decoded and interpolated back to their original sampling rates with the aid of similar low-pass and high-pass filters. The sum of the two interpolated sub-band signals, $\hat{x}(n)$, is the reconstructed version of the input signal, $x(n)$, (see Fig. 1).

This process of dividing the signal into sub-bands permits each band to be encoded with a different number of bits per sample and with a independent adaptive step-size in order to obtain a better perceived quality. For telephone band speech (200 to 3200 Hz) sampled at 8 kHz, the two-band technique provides about a 3- or 4-kb/s advantage over ADPCM in the bit-rate range of 24 kb/s.[9] In another study,[3,10] it has been shown that the two-band SBC design is useful for encoding of



(a)



(b)

Fig. 1—(a) General block diagram of a two-band sub-band coder. (b) A spectral description of the sub-bands.

wider bandwidth (7 kHz) signals at bit rates which are commensurate with digital transmission rates commonly used for telephony (56 or 64 kb/s). The "commentary quality" obtained from this design is suitable for applications such as broadcast services for news, correspondence, sports, and AM radio music transmission.

Greater bit-rate compression, (or higher quality performance for the same bit rate) over that of a two-band scheme is possible with more bands.[7,8] This can be accomplished, for example, by further subdividing each of the two sub-band signals into two more sub-bands at one-quarter of the original sampling rate to produce a four-band SBC design (with equally spaced frequency bands). Alternatively, designs with octavely spaced bands are possible by successively subdividing the lower bands in a "tree structure."[11] Such designs are a logical extension of the two-band approach, and they can have a performance advantage of up to about 8 kb/s over ADPCM (in the range of 16 to 24 kb/s) for speech.[7-9,11]

In the following sections, we discuss in detail an implementation of the two-band SBC design on the DSP. In Section III, we first review some of the theoretical aspects of the design, particularly the structure of the quadrature mirror approach to the filter bank. Then, in Section IV, we consider some of the programming aspects of the design on the DSP and show how features of the DSP are used in the implementation.

III. DESIGN CONSIDERATIONS

3.1 The quadrature mirror filter bank

An important and critical aspect of the SBC design is that of the filter bank and its interaction with the sampling rate reduction (decimation) and the subsequent sampling rate increase (interpolation) of the sub-band signals. The approach used in this design is that of the quadrature mirror filter bank (QMFB).[12] In this section, we will summarize the basic concepts of the QMFB and consider a practical filter design. Later, in Section 4.4, we will discuss the implementation of the QMFB on the DSP.

The reduction of the sub-band sampling rates is necessary in order to maintain a minimal overall bit-rate in encoding these signals. This sampling rate reduction introduces aliasing terms in each of the sub-band signals. For example, in the lower band the signal energy in the frequency range above $F_s/4$ is folded down into the range 0 to $F_s/4$ and appears as aliasing in this signal, as illustrated by the shaded region in Fig. 1b. Similarly, for the upper band any signal energy in the frequency range below $F_s/4$ is folded upward into its Nyquist band $F_s/4$ to $F_s/2$. This mutual aliasing of signal energy between the upper and lower sub-bands is sometimes called interband "leakage." The amount of leakage that occurs between sub-bands is directly dependent

on the degree to which the filters $h_l(n)$ and $h_u(n)$ approximate ideal low-pass and high-pass filters, respectively.

In the reconstruction process the sub-band sampling rates are increased by filling in zero valued samples between each pair of sub-band samples. This introduces a periodic repetition of the signal spectra in the sub-band. For example, in the lower band the signal energy from 0 to $F_s/4$ is symmetrically folded around the frequency $F_s/4$ into the range of the upper band. This unwanted signal energy, referred to as an "image" is filtered out by the low-pass filter $h_l(n)$ in the receiver. This filtering operation effectively interpolates the zero valued samples that have been inserted between the sub-band signals to values that appropriately represent the desired waveform.[13] Similarly, in the upper sub-band signal an image is reflected to the lower sub-band and filtered out by the filter $-h_u(n)$.

The degree to which the above images are removed by the filters $h_l(n)$ and $-h_u(n)$ is determined by the degree to which they approximate ideal low-pass and high-pass filters. Because of the quadrature relationship of the sub-band signals in the QMFB the remaining components of the images can be *exactly* canceled by the aliasing terms introduced in the analysis (in the absence of coding errors). In practice, this cancellation is obtained down to the level of the quantization noise of the coders.

To obtain this cancellation property in the QMFB, the filters $h_l(n)$ and $h_u(n)$ must be symmetrical finite impulse response (FIR) designs[14] with even numbers of taps, i.e.,

$$h_l(n) = h_u(n) = 0 \qquad \begin{matrix}\text{for } n < 0,\\ \text{and } n \geq N\end{matrix} \tag{1}$$

where N, even, is the number of taps. The symmetry property implies that

$$h_l(n) = h_l(N-1-n), \qquad n = 0, 1, 2, \cdots, \quad \frac{N}{2}-1, \quad \text{and} \tag{2a}$$

$$h_u(n) = -h_u(N-1-n), \qquad n = 0, 1, 2, \cdots, \quad \frac{N}{2}-1. \tag{2b}$$

The QMFB further requires that the filter in Fig. 1a satisfy the condition.[12]

$$h_u(n) = (-1)^n h_l(n) \qquad n = 0, 1, \cdots N-1, \tag{3}$$

which is the mirror image relationship of the filters.

With the above constraints, the aliasing cancellation property of the QMFB can be easily verified.[12] A derivation is given in the Appendix. As seen from this derivation, the filters $h_l(n)$ and $h_u(n)$ must also ideally satisfy the condition

$$|H_l(e^{j\omega})|^2 + |H_u(e^{j\omega})|^2 = 1, \tag{4}$$

where $H_l(e^{j\omega})$ and $H_u(e^{j\omega})$ are the Fourier transforms of $h_l(n)$ and $h_u(n)$, respectively.

The above filter requirement of eq. (4) cannot be met exactly except when $N = 2$ and when N approaches infinity. However, it can be very closely approximated for modest values of N. Filter designs which satisfy eq. (2a) and approximate the condition of eq. (4) and the lowpass characteristic can be obtained with the aid of an optimization program. Reference 15 describes a procedure based on the Hooke and Jeeves optimization algorithm and presents a set of filter designs for values of $N = 8$, 12, 16, 24, 32, 48, and 64. Also, useful but less optimal designs can be obtained from conventional Hanning window designs.[3]

Figure 2 shows the frequency response characteristics for an $N = 32$-tap filter design that was used in the DSP implementation and Table I gives the filter coefficients. Fig. 2a shows the magnitude of $H_l(e^{j\omega})$ and $H_u(e^{j\omega})$ expressed in dB as a function of ω and Fig. 2b shows the magnitude of the expression

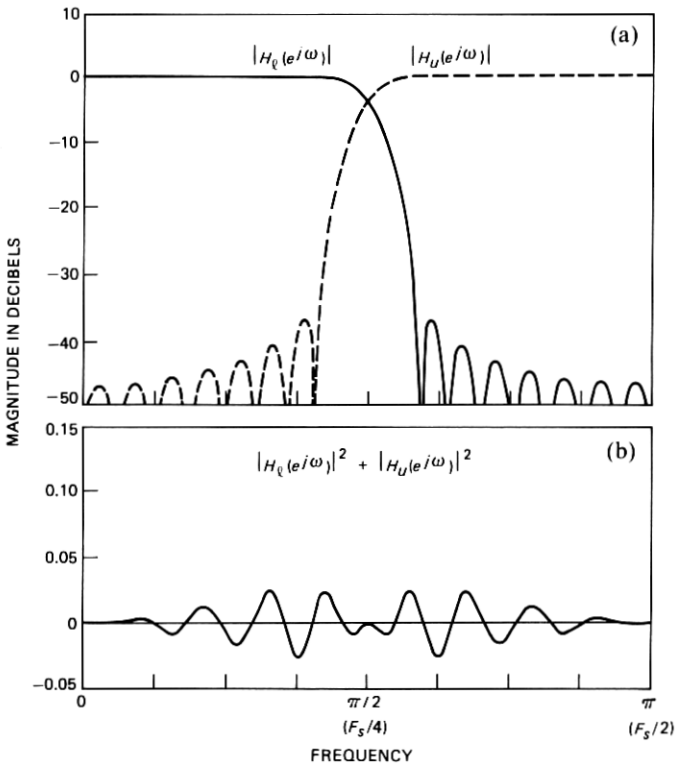


Fig. 2—Frequency response for a 32-tap quadrature mirror filter design. (a) Magnitude responses of the individual filters. (b) Magnitude response of the composite system.

Table I—Coefficients for 32-tap FIR quadrature mirror filter

$h_l(0) =$	0.002245139	$= h_l(31)$	$h_l(8) =$	−0.007961731	$= h_l(23)$
$h_l(1) =$	−0.003971152	$= h_l(30)$	$h_l(9) =$	−0.034964400	$= h_l(22)$
$h_l(2) =$	−0.001969672	$= h_l(29)$	$h_l(10) =$	0.019472180	$= h_l(21)$
$h_l(3) =$	0.008181941	$= h_l(28)$	$h_l(11) =$	0.054812130	$= h_l(20)$
$h_l(4) =$	0.000842683	$= h_l(27)$	$h_l(12) =$	−0.044524230	$= h_l(19)$
$h_l(5) =$	−0.014228990	$= h_l(26)$	$h_l(13) =$	−0.099338590	$= h_l(18)$
$h_l(6) =$	0.002069470	$= h_l(25)$	$h_l(14) =$	0.132972500	$= h_l(17)$
$h_l(7) =$	0.022704150	$= h_l(24)$	$h_l(15) =$	0.463674100	$= h_l(16)$

$$|H_l(e^{j\omega})|^2 + |H_u(e^{j\omega})|^2$$

expressed in dB as a function of ω. As can be seen from Fig. 2b, the requirement of eq. (4) is satisfied to within ± 0.025 dB which is more than satisfactory for good SBC performance. The above filter design is based on the "32 D" design.[15]

This concludes our discussion of the QMFB conditions and the filter design. In Section 4.4 we discuss how the mirror image relationship of eq. (3) is used to advantage in the DSP implementation.

3.2 The ADPCM coders

The adaptive differential PCM (ADPCM) coders in the two-band SBC are based on the algorithm by Cummiskey, Jayant, and Flanagan[16] and they use the robust form of the step-size adaptation by Goodman and Wilkinson.[17]

A detailed description of this algorithm is given in a companion paper.[6] Therefore, in this section we will only briefly outline the form of the algorithm to identify relevent parameters and refer the reader to Ref. 6 for specifics.

Figure 3a shows a simplified block diagram of the ADPCM algorithm. The input (decimated) sub-band signal is denoted as $y(n)$. A predicted estimate of this signal, $p(n)$, is subtracted from $y(n)$ to produce the difference signal

$$e(n) = y(n) - p(n). \tag{5}$$

This difference signal is then quantized with an adaptive step-size quantizer to produce the code word $I(n)$ and the decoded difference signal $\hat{e}(n)$.

The step-size of the quantizer $\Delta(n)$ is adaptively varied according to the relation

$$\Delta(n) = (\Delta(n-1))^{\gamma} \cdot M(I(n-1)), \tag{6}$$

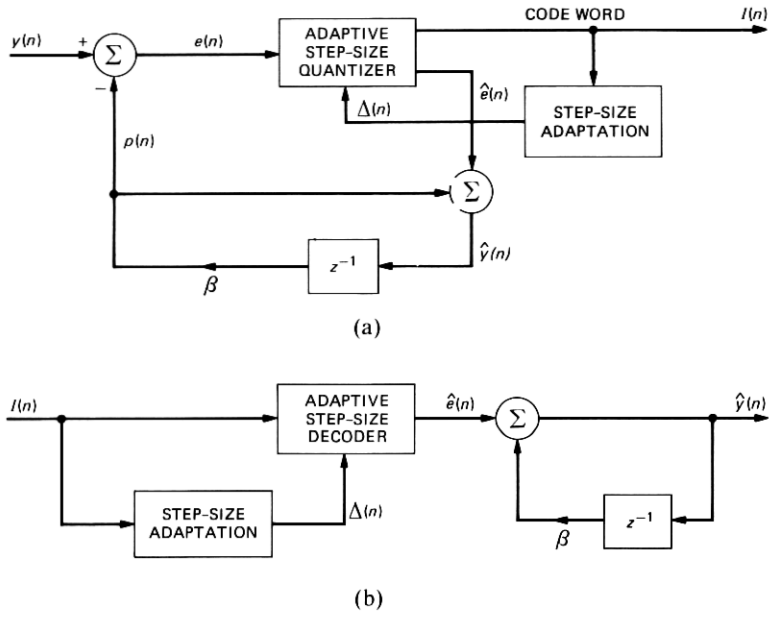


Fig. 3—General block diagram of the ADPCM coders. (a) Encoder. (b) Decoder.

where $\Delta(n-1)$ is the step-size and $I(n-1)$ is the code word at the previous sample time $n-1$. The parameter γ is a number in the range.

$$0 \leq \gamma \leq 1, \tag{7}$$

and, typically, has a value of $\gamma = 0.98$. It is used to introduce a limited memory to the step-size adaptation algorithm to mitigate the effects of channel errors.[17] The scale factor $M(I(n))$ is a number that depends on the code word $I(n)$. If an outermost positive or negative quantizer level is used at time $n-1$ a value of $M(\cdot)$ greater than one (typically $M(\cdot) = 2$) is used to increase the step-size for the sample time n. If a lower quantizer magnitude level is used at time $n-1$, a value of $M(\cdot)$ less than one (typically $M(\cdot) = 0.77$) is used to reduce the step-size at time n. In this way, the step-size is dynamically varied in an attempt to match the center of the quantizer characteristic to that of the rms level of the difference signal $e(n)$. The values of $M(\cdot)$ can be tailored to modify the adaptation characteristics of the quantizer. Typically, a faster attack (step-size increase) and a slower decay (step-size decrease) is preferred[10,16] for best subjective performance.

The sum of the decoded difference signal $\hat{e}(n)$ and the predictor signal $p(n)$ gives the decoded version of the input signal, denoted as $\hat{y}(n)$, i.e.

$$\hat{y}(n) = \hat{e}(n) + p(n). \tag{8}$$

This is used in the ADPCM receiver, (see Fig. 3b) to produce the decoded output signal. It is also used in the ADPCM transmitter (see Fig. 3a) to generate the predictor signal according to the relation

$$p(n) = \beta \cdot \hat{y}(n-1). \tag{9}$$

The parameter β determines the fraction of the signal $\hat{y}(n-1)$ that is used to predict the next incoming sample $y(n)$. Ideally it should be equal to the sample-to-sample correlation that exists in the signal $y(n)$.[16]

For speech, sampled at 8 kHz, it has been suggested that values of $\beta_l = 0.7$ (lower band) and $\beta_u = -0.45$ (upper band) are appropriate.[9] The negative correlation in the upper sub-band is because the frequency scale of the spectrum is inverted in the decimation process of the QMFB. For audio signals, sampled at 14 kHz, values of $\beta_l = 0.16$ and $\beta_u = -0.82$ have been suggested[3] (note that β_l and β_u are referred to as α_1 and α_2 in Ref. 3).

The number of bits per sample used to encode each sub-band is dependent on the overall bit rate of the coder. For speech, sampled at 8 kHz, a choice of 4 bits/sample for the low band and 2 bits/sample for the upper band leads to a 24-kb/s design. For audio, sampled at 14 kHz, a choice of 4 bits/sample was used in each sub-band for the 56-kb/s commentary grade coder.[3]

3.3 Prefiltering

It is sometimes desirable in SBC coding to band-limit the input signal prior to encoding. For example, in speech a substantial amount of signal energy may be present in the frequency range from 0 to 200 Hz. This energy contributes to an increased step-size and, therefore, more quantization noise in the lower sub-band. If telephone band speech (200 to 3200 Hz) is of interest, then band-limiting the input signal to this range before encoding removes the signal energy below 200 Hz and above 3200 Hz. This permits the use of a lower step-size in the bottom band and, therefore, produces less quantization noise. For audio, a similar advantage is gained from prefiltering by removing low frequency hum and turntable rumble components in the signal prior to encoding.

Figure 4 shows an example of a cascade filter structure for a sixth-order infinite impulse response (IIR) filter[14] that was used in the DSP implementation for this purpose. The coefficients for a 200- to 3200-Hz bandpass elliptic filter design (assuming an 8-kHz sampling rate) are given in Table II and the frequency response for this design is shown in Fig. 5.

In the design of IIR filters, some caution must be observed in minimizing the effects of roundoff noise, limit cycles, and dynamic

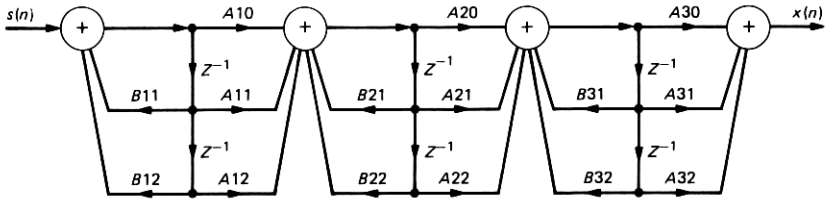


Fig. 4—Block diagram of the cascade structure for the IIR prefilter.

range constraints within the filter. This can be accomplished by observing several rules of thumb for pairing and ordering the arrangement of poles and zeroes within the cascade filter structure and also by appropriately scaling the signal from section to section within the filter structure to control the internal dynamic range. Further information on this subject can be found in Refs. 14 and 18. In general, these procedures are more critical for high-order high Q filters and less critical for low-order low Q designs.

IV. IMPLEMENTING THE SBC ALGORITHM ON THE DSP

4.1 Some general programming considerations

As seen from the above discussion there are a number of different aspects to consider in the implementation of an algorithm such as SBC on the DSP. In this section, we discuss some of these issues and point out some general programming techniques that were used. Principles such as modularization, stream processing, block processing, and double buffering will be introduced. In Sections 4.2 to 4.5 we discuss more specifically how these principles are used in the SBC software. We will assume in the following discussion that the reader is generally familiar with the DSP software.

The software development for the SBC and similar signal processing algorithms is greatly simplified by recognizing the fact that there are several well-defined operations that are being performed in the algorithm, such as filtering, coding, and sampling rate conversion. By identifying these operations and modularizing the software around them, the problem can be subdivided into a series of smaller problems.

Table II—Coefficients for sixth-order IIR bandpass elliptic filter
(200- to 3200-Hz BW, 8000-Hz sampling rate)

A10	1.0	B21	−1.44837372
A11	−1.99730706	B22	−0.746318392
A12	1.0	A30	1.0
B11	0.470839023	A31	1.95459080
B12	0.2281607545	A32	1.0
A20	0.459738676	B31	1.9053369
A21	0.0	B32	−0.92630251
A22	−0.459738676		

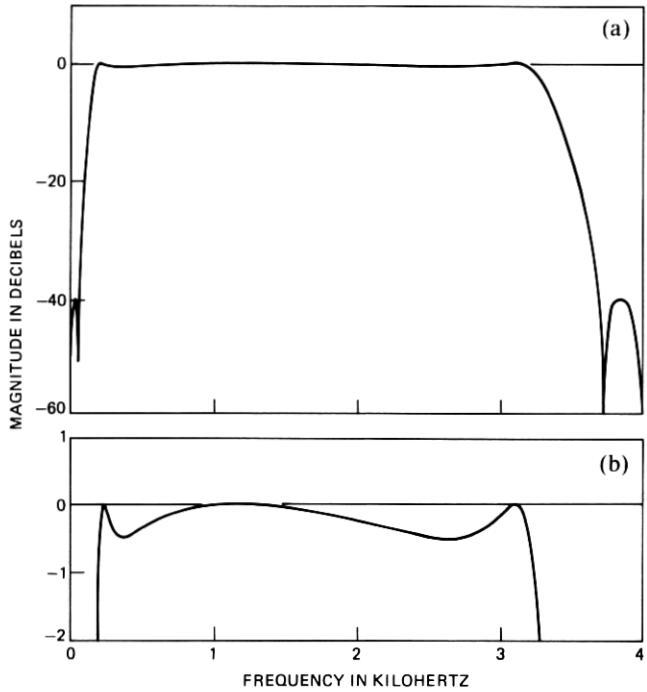


Fig. 5—Frequency response of the 200- to 3200-Hz (8000-Hz sampling rate) bandpass elliptic filter. (a) Overall response. (b) Expanded view of the passband ripples.

This modularization process consists of assigning and labeling separate parts of the DSP RAM memory for each block in the algorithm. For example, in the IIR filter the six internal (state) variables necessary for the filter, denoted as BPI[0], BPI[1], ⋯, BPI[5], can be assigned to RAM locations using the RAM-variable definition statement

ram BPI[6].

At the beginning of each module, the address pointers of the DSP can then be initialized for that module. For example, the statement

rya = &BPI[0];

sets the DSP address pointer RYA to the first state variable in the filter. The automatic increment or decrement feature of the address pointers in the DSP can then be used to step RYA through the state variables within the module. Although it may cost a few extra lines of code, this simple, perhaps obvious, principle goes a long way toward simplifying the programming and debugging process in the DSP by unlinking the address connections between modules and generally producing more readable code.

Signal processing operations, such as filtering and coding, generally require a single input sample and produce a single output sample for each sample time. They also perform essentially the same signal processing operations for each sample time. Such operations will be referred to as stream processing operations and they can be conveniently defined and implemented in the modular fashion discussed above.

When more than one sampling rate is involved in an algorithm, the operations to be performed within a module, or the entire module itself, may be different from sample to sample. For example, if there is a 2-to-1 difference in sampling rate within the algorithm it may be necessary to perform one set of operations at the odd sample times (cycle zero) and a different set of operations (cycle one) at the even sample times in certain parts of the algorithm. For this, it is necessary to introduce the concepts of block processing in which different program modules are used for different cycles. Furthermore, interfacing stream processing modules with block processing modules may require double buffering operations and care must be used to distribute the amount of processing performed in each cycle to avoid I/O synchronization problems. These concepts will become more clear when we discuss the implementation of the QMFB and sampling rate conversion in Sections 4.4 and 4.5.

We first consider the implementation of the IIR prefilter and the ADPCM coder module. Then, we show how these modules fit within the general multirate processing framework of the SBC coder.

4.2 IIR prefilter

The IIR filter can be implemented in a straightforward stream processing manner on the DSP. Beause the DSP is especially suited for linear filtering algorithms of this type, the filter structure of Fig. 4 can be very efficiently realized with approximately one DSP instruction for each combination of a shift, multiply, and add in the structure.

Assuming that the internal state variables are stored in RAM locations BPI[0], BPI[1], $\cdots$, BPI[5], and the input signal $s(n)$ is in the P register, the following DSP instructions compute $x(n)$ according to Fig. 4 (see Table II for the filter coefficients).

```
                rya = &BPI[0];
                rda = &BPI[0];;
                                  a = p       p = B12**rya++;
                                  a = p + a   p = B11**rya--;
                                  a = p + a   p = A12**rya++;
*rda++ = y      w = a             a = p       p = A11**rya++;
*rda++ = w                        a = p + a   p = A10*w;
                                  a = p + a   p = B22**rya++;
```

```
                                a = p + a   p = B21**rya--;
                                a = p + a   p = A22**rya++;
*rda++ = y      w = a           a = p       p = A21**rya++;
*rda++ = w                      a = p + a   p = A20*w;
                                a = p + a   p = B32**rya++;
                                a = p + a   p = B31**rya--;
                                a = p + a   p = A32**rya++;
*rda++ = y      w = a           a = p       p = A31**rya++;
*rda++ = w                      a = p + a   p = A30*w;
                                a = p + a;
```

The output signal $s(n)$ appears in the A register. The filter coefficients are stored in the beginning of the program using # define statements and are inserted into the code by the assembler.

4.3 ADPCM encoder and decoders

The ADPCM encoders and decoders are also stream processing algorithms in the sense that they receive a single input sample and produce a single output sample for each sample time. However, as seen in Fig. 1, the sampling rate at which they operate in the SBC algorithm is one half of the input sampling rate. As will be discussed in the next section, this can be accomplished by a two-cycle computational structure in which the encoder and decoder for the lower sub-band are computed in one cycle time (cycle 0) and the encoder and decoder for the upper sub-band are computed in the second cycle time (cycle 1). Since each cycle time is associated with one-half of the input sampling rate, the ADPCM coders operate in a stream processing manner within this framework.

See Ref. 6 for a detailed discussion of the ADPCM algorithm and its implementation on the DSP, since essentially the same design and code have been used for the SBC algorithm.

4.4 Quadrature mirror filter bank and the multirate computational structure

Perhaps the most subtle aspect of the SBC algorithm is that it is a multirate system;[13] i.e., it has more than one sampling rate. This imposes a block processing framework on the computational structure of the system. In the next two sections, we will discuss these issues in more detail and present a computational structure for the two-band SBC. First, we will discuss the polyphase structure for the QMFB which takes advantage of its mirror image and multirate properties and results in a more efficient realization than the one implied by Fig. 1.

From the mirror image property of the QMFB described by eq. (3), note that the coefficients used for the upper and lower sub-band filters are identical, except for the signs of the odd-numbered coefficients.

This property can be used to save a factor of two in computation by sharing the computation between the filters in the manner described in Fig. 6.[12] The partial sums of products are accumulated separately for the even- and odd-filter coefficient values. The sum of these two partial sums then gives the lower sub-band signal, and their difference produces the upper sub-band signal. Since the sampling rates are one-half of the input sampling rate, an additional factor of two is gained by computing the sums of products indicated in Fig. 6 once for every other input sample. Thus, each sample is shifted two delays in the shift register of Fig. 6 before being used.

Because of this sample rate reduction, the filter structure of Fig. 6 can be divided into two parts as shown in Fig. 7. This structure is a two-band version of a more general class of multirate structures sometimes referred to as polyphase structures.[13,19] As Fig. 7 shows, the input signal is separated into two sets by a commutator. Assuming that the commutator is in the upper position at time $n = -1$, the upper branch receives odd values of $x(n)$, i.e. $x(-1)$, $x(1)$, $x(3)$, $x(5)$ $\cdots$, and the lower branch receives even values of $x(n)$, i.e. $x(0)$, $x(2)$, $x(4)$, $\cdots$. Both branches now operate at one-half of the original sampling rate. Odd values of $x(n)$ are filtered at odd sample times in the upper branch with an $N/2$ tap filter of odd valued filter coefficients. Similarly, even valued samples of $x(n)$ are filtered in the lower branch with an $N/2$ tap filter of even filter coefficients.

At the end of the even sample times, the sums and differences of the two filter outputs are taken to produce the (decimated) lower and upper sub-band signals respectively. This sum and difference amounts to a two-point DFT (a discrete Fourier transform butterfly) in the two-band polyphase framework. The purpose of the double buffer will be discussed in more detail in connection with the timing and control structure in the next section.

By careful analysis of the receiver structure of Fig. 1 a similar efficient polyphase structure can be generated for the QMFB synthesis. Alternatively, it can be generated by applying concepts of multirate

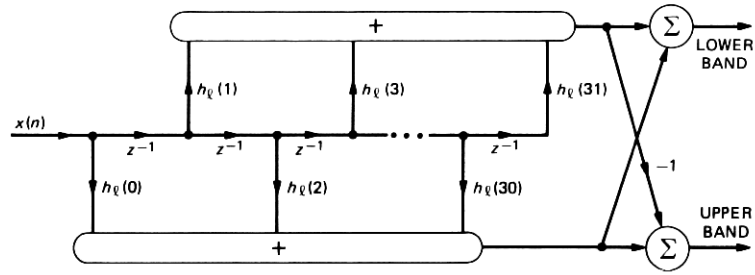


Fig. 6—Quadrature mirror filter bank structure that shares computation between upper and lower filters.

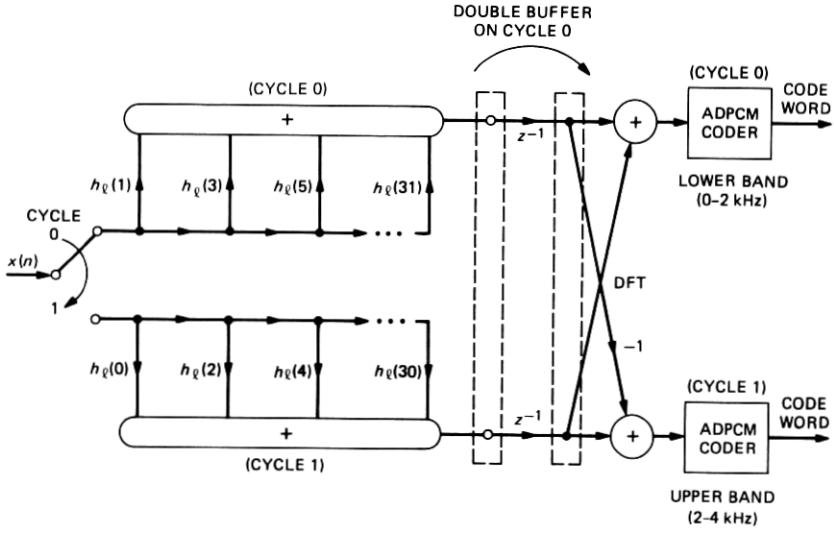


Fig. 7—Sub-band coding transmitter structure using a polyphase QMFB.

network transposition[20] to the structure of Fig. 7. The resulting structure, with either approach is shown in Fig. 8. The sum and difference (an inverse DFT) of the ADPCM decoder output signals are first computed to produce inputs for the odd and even FIR filters, respectively. At odd sample times (cycle 0) the even FIR filter coefficients (upper branch) are used to compute odd sample values of the output $\hat{x}(n)$, i.e.

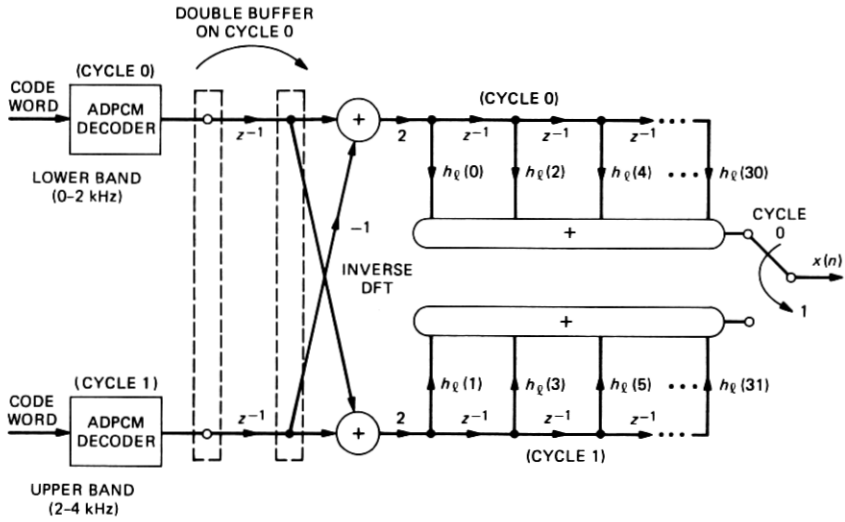


Fig. 8—Sub-band coding receiver structure using a polyphase QMFB for synthesis.

$\hat{x}(-1)$, $\hat{x}(1)$, $\hat{x}(3)$, $\cdots$. At even sample times (cycle 1), the odd FIR filter coefficients (lower branch) are used to compute even samples of the output $\hat{x}(n)$, i.e. $\hat{x}(0)$, $\hat{x}(2)$, $\hat{x}(4)$, $\cdots$.

As in the analysis structure of Fig. 7, note that one-half of the filter computation is performed at even sample times and the other half is performed at the odd sample times. Thus, the computational load is evenly distributed between even and odd time cycles. This will be discussed in more detail in the next section on the control structure.

The DSP can be very efficiently used to perform the above operations. For example, the FIR filters can be implemented with essentially one line of code per tap in the filter, plus a few setup instructions. Assuming that the state variables of the 16-tap odd coefficient filter in the upper branch of Fig. 7 are stored in RAM locations $XO[0]$, $XO[1]$, $\cdots$, $XO[15]$, and the filter input is stored in the A register, the following DSP instructions compute the filter output.

```
                rya = &XO[15];
                rda = &XO[15];;
                w = a           a = p       p = H31**rya--;
*rda-- = y                      a = p       p = H29**rya--;
*rda-- = y                      a = p + a   p = H27**rya--;
*rda-- = y                      a = p + a   p = H25**rya--;
*rda-- = y                      a = p + a   p = H23**rya--;
*rda-- = y                      a = p + a   p = H21**rya--;
*rda-- = y                      a = p + a   p = H19**rya--;
*rda-- = y                      a = p + a   p = H17**rya--;
*rda-- = y                      a = p + a   p = H15**rya--;
*rda-- = y                      a = p + a   p = H13**rya--;
*rda-- = y                      a = p + a   p = H11**rya--;
*rda-- = y                      a = p + a   p = H9**rya--;
*rda-- = y                      a = p + a   p = H7**rya--;
*rda-- = y                      a = p + a   p = H5**rya--;
*rda-- = y                      a = p + a   p = H3**rya--;
*rda = w                        a = p + a   p = H1*w;
                                a = p + a;
```

The coefficients $H1$, $H3$, $\cdots$, $H31$ correspond to the filter coefficients $h_l(1)$, $h_l(3)$, $\cdots$ $h_l(31)$, respectively in Table I. They are stored at the beginning of the program using # define statements and inserted into the code using the assembler. Note that the filter state variables are addressed in reverse order, i.e. $XO[15]$, $XO[14]$, $\cdots$, $XO[2]$, $XO[1]$, $XO[0]$ and that the filter input is held in the W register until coefficient $H1$ is reached, and then it is stored in location $XO[0]$. The output of the filter appears in the A register.

4.5 Control and data flow structure

So far we have discussed block diagrams and programs for individual modules within the SBC code. In this section, we consider the overall control structure and flow of data within the program. Because the SBC algorithm is a multirate system with a sampling rate ratio of two, it requires a two-cycle control structure (for a sampling rate ratio of 4, a four-cycle control structure would be needed, ect). Cycle zero is associated with odd sample times $n = 1, 1, 3, 5, \cdots$, and cycle 1 is associated with even sample times $n = 0, 2, 4, \cdots$. As seen in Figs. 7 and 8, most of the operations in the upper branches of these structures are computed in cycle 0 and most of the operations in the lower branches are computed in cycle 1. Thus, the ADPCM coder and decoder for the lower sub-band are computed in cycle 0 and the ADPCM coder and decoder for the upper sub-band are computed in cycle 1. In this way, the computational load is shared equally between both cycles. Note that the bandpass prefilter must be computed in both cycles since it is implemented at the high (input) sampling rate.

Double buffering is required in the multirate structure when data computed in one cycle is required in another cycle. For example, in Fig. 7 the outputs of the even and odd filters are computed and stored in the left buffer for cycles 0 and 1, while the DFT uses available data from the right buffer which was computed in the previous 0 and 1 cycles. At the beginning of cycle 0 (or the end of the last cycle) the data is transferred from the left buffer to the right buffer. This transfer can be accomplished with essentially no extra overhead by using the features of the DSP.

For example the DFT sum in the upper branch of Fig. 7 is computed in cycle 0 (the difference is computed in cycle 1). This is accomplished by reading data from the left buffer (RAM memory $XB[0]$ and $XB[1]$) and simultaneously transfering it to the right buffer (RAM memory $XBB[0]$ and $XBB[1]$), while the DFT sum is being performed. The following DSP instructions perform this operation:

```
                rya = &XB[0];
                rya = &XBB[0];;
*rda++ = y                                  p = *rya++;
*rda = y                          a = p     p = *rya;
                                  a = p + a;
```

The DFT sum appears in the A register. In cycle 1, the DFT difference output is computed from data in the left buffer (read in reverse order to accommodate the difference operation in the DSP).

Table III summarizes the sequence of operations that are performed in cycle 0 and cycle 1 of the two-band SBC software. Both the SBC transmitter and receiver are implemented in the same DSP for the sake

Table III—Control structure for two-band SBC

I. Cycle 0	A. Transmitter 1. Double buffer 2. DFT sum 3. ADPCM encoder (lower band) 4. Output (or store) code word 5. Input one sample of $x(n)$ 6. BP prefilter 7. FIR filter (upper branch) B. Receiver 1. Double buffer 2. IDFT sum 3. FIR filter (upper branch) 4. Output one sample of $\hat{x}(n)$ 5. Input code word 6. ADPCM decode (lower band)
II. Cycle 1	A. Transmitter 1. DFT difference 2. ADPCM encoder (upper band) 3. Output (or store) code word 4. Input one sample of $x(n)$ 5. BP prefilter 6. FIR filter (lower branch) B. Receiver 1. IDFT difference 2. FIR filter (lower branch) 3. Output one sample of $\hat{x}(n)$ 4. Input code word 5. ADPCM decode (upper band)
Return to I. Cycle 0.	

of demonstration and the code words are simply stored in memory and read back in the receiver. One input sample of the signal $x(n)$ is used in each cycle and one output sample of $\hat{x}(n)$ is generated in each cycle. Although the code for the transmitter and receiver are interlaced in this control structure it can be easily separated, because of the modular design, so that the transmitter and receiver can be realized in separate DSPs.

V. DISCUSSION

We have discussed an implementation of a two-band sub-band coder using the DSP. Both the transmitter and receiver have been implemented on the same DSP including a sixth-order bandpass prefilter. The algorithm was implemented according to the signal processing structures in Figs. 4, 7, 8, and the ADPCM encoder and decoder structures discussed in Ref. 6. Table III outlines the overall control structure for the algorithm.

The program uses approximately 98 percent of the 8-kHz real-time capability of the DSP running with a 5-MHz clock. It uses approximately 78 percent of the RAM and 73 percent of the ROM. If the transmitter and receiver are separated to two DSPs (in a practical situation) approximately one-half of the above processing capability and memory will be required for each DSP.

Based on the above figures the two-band commentary grade coder for audio ecoding at a 14-kHz sample rate[3] is possible using a single DSP for the transmitter and another DSP for the receiver.

Also these figures suggest that a four-band sub-band coder design which further subdivides each of the above two sub-bands into two more sub-bands by similar quadrature mirror filter techniques is realizable using one DSP for the transmitter and a second DSP for the receiver. Such designs offer greater bit-rate compression capability for speech than the two-band SBC algorithm discussed here.[7-9,11] Efforts are continuing in this direction

VI. ACKNOWLEDGMENTS

The author wishes to thank C. A. McGonegal, J. D. Johnston, R. V. Cox, and J. W. Upton for their comments and discussion in the course of this work.

APPENDIX

Aliasing Cancellation Property of the Quadrature Mirror Filter Bank[12]

Let $X_l(e^{j\omega})$ and $X_u(e^{j\omega})$ be the Fourier transforms of the lower and upper sub-band signals, respectively before decimation and $X(e^{j\omega})$ be the transform of $x(n)$. Then

$$X_l(e^{j\omega}) = X(e^{j\omega})\, H_l(e^{j\omega}), \qquad \text{and} \tag{10}$$

$$X_u(e^{j\omega}) = X(e^{j\omega})\, H_u(e^{j\omega}), \tag{11}$$

where $H_l(e^{j\omega})$ and $H_u(e^{j\omega})$ are the Fourier transforms of $h_l(n)$ and $h_u(n)$, respectively. After decimation the lower and upper sub-band signals will be defined as $Y_l(e^{j\omega})$ and $Y_u(e^{j\omega})$, respectively and can be expressed as[13]

$$Y_l(e^{j\omega}) = \tfrac{1}{2}\,[X_l(e^{j\omega/2}) + X_l(e^{j(\omega+2\pi)/2})], \qquad \text{and} \tag{12}$$

$$Y_u(e^{j\omega}) = \tfrac{1}{2}\,[X_u(e^{j\omega/2}) + X_u(e^{j(\omega+2\pi)/2})]. \tag{13}$$

Letting $U_l(e^{j\omega})$ and $U_u(e^{j\omega})$ be the interpolated lower and upper sub-band signals, respectively in the receiver, and ignoring effects of quantization because of the coders we get

$$U_l(e^{j\omega}) = 2\, Y_l(e^{j2\omega}) H_l(e^{j\omega}), \qquad \text{and} \tag{14}$$

$$U_u(e^{j\omega}) = -2\, Y_u(e^{j2\omega}) H_u(e^{j\omega}). \tag{15}$$

Finally the output signal $\hat{X}(e^{j\omega})$, the transform of $\hat{x}(n)$ in Fig. 1a, can be expressed as

$$\hat{X}(e^{j\omega}) = U_l(e^{j\omega}) + U_u(e^{j\omega}). \tag{16}$$

Combining eqs. (10) to (16) gives the input to output relation

$$\hat{X}(e^{j\omega}) = X(e^{j\omega})[H_l^2(e^{j\omega}) - H_u^2(e^{j\omega})] + X(e^{j(\omega+\pi)})[H_l(e^{j\omega})H_l(e^{j(\omega+\pi)}) - H_u(e^{j\omega})H_u(e^{j(\omega+\pi)})]. \quad (17)$$

The first term in this expression expresses the desired signal component of $\hat{X}(e^{j\omega})$ and the second term expresses the undesired aliasing component. The cancellation of this aliasing component can be observed by transforming eq. (3) to get

$$H_l(e^{j\omega}) = H_u(e^{j(\omega+\pi)}), \quad (18)$$

and applying this condition to eq. (17). It can be easily verified that the second term cancels leaving

$$\hat{X}(e^{j\omega}) = X(e^{j\omega})[H_l^2(e^{j\omega}) - H_l^2(e^{j(\omega+\pi)})]. \quad (19)$$

From the symmetry property in eq. (2), it can be shown that the frequency response of $H_l(e^{j\omega})$ can be expressed in the form

$$H_l(e^{j\omega}) = |H_l(e^{j\omega})|e^{j\omega(N-1)/2}. \quad (20)$$

Recalling that N is even and applying this condition to eq. (19), leads to the expression

$$\hat{X}(e^{j\omega}) = X(e^{j\omega})[|H_l(e^{j\omega})|^2 + |H_l(e^{j(\omega+\pi)})|^2]e^{j\omega(N-1)}. \quad (21)$$

In the above expression, the term $e^{j\omega(N-1)}$ implies that there is an $N-1$ sample delay between $\hat{x}(n)$ and $x(n)$. Furthermore, it can be seen from eq. (21) that if $\hat{x}(n)$ is to be a (delayed) replica of $x(n)$ then $H_l(e^{j\omega})$ must satisfy the requirement that

$$|H_l(e^{j\omega})|^2 + |H_l(e^{j(\omega+\pi)})|^2 = 1, \quad (22)$$

or equivalently

$$|H_l(e^{j\omega})|^2 + |H_u(e^{j\omega})|^2 = 1. \quad (23)$$

REFERENCES

1. J. L. Flanagan et al., "Speech Coding," IEEE Trans. Commun., *COM-27,* No. 4 (April 1979), pp. 710–37.
2. J. M. Tribolet and R. E. Crochiere, "Frequency Domain Coding of Speech," IEEE Trans. ASSP, *ASSP-27,* No. 5 (October 1979), pp. 512–30.
3. J. D. Johnston and R. E. Crochiere, "An All Digital Commentary Grade Sub-band Coder," J. of the Audio Eng. Society, *27,* No. 11 (November 1979), pp. 855–65.
4. J. R. Boddie et al., "Digital Signal Processor: Architecture and Performance," B.S.T.J., this issue.
5. J. S. Thompson and J. R. Boddie, "An LSI Digital Signal Processor," Proc. 1980 IEEE Int. Conf. ASSP (April 1980), pp. 383–5.
6. J. R. Boddie et al., "Digital Signal Pressure: ADPCM Coding," B.S.T.J., this issue.
7. R. E. Crochiere, S. A. Webber, and J. L. Flanagan, "Digital Coding of Speech in Sub-Bands," B.S.T.J., *55,* No. 7 (October 1976), pp. 1069–85.
8. R. E. Crochiere, "On the Design of Sub-Band Coders for Low-Bit-Rate Speech Communications," B.S.T.J., *56,* No. 5 (May–June 1977), pp. 747–70.

9. R. V. Cox, "A Comparison of Three Coders to be Implemented on the Digital Signal Processor," B.S.T.J., this issue, Part 2.
10. J. D. Johnston and D. J. Goodman, "Digital Transmission of Commentary-Grade (7 kHz) Audio at 56 or 64 kb/s," Proc. IEEE Int. Conf. ASSP. Proc. (1979), pp. 442–4.
11. A. J. Barabell and R. E. Crochiere, "Sub-band Coder Design Incorporating Quadrature Filters and Pitch Prediction," in Proc. IEEE Int. Conf. ASSP (1979), pp. 530–3.
12. D. Esteban and C. Galand, "Application of Quadrature Mirror Filters to Split Band Voice Coding Schemes," Proc. IEEE Int. Conf. ASSP (1977), pp. 191–5.
13. R. E. Crochiere and L. R. Rabiner, "Interpolation and Decimation of Digital Signals—A Tutorial Review," Proc. IEEE, *69,* No. 3 (March 1981), pp. 300–31.
14. L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing,* New York: Prentice Hall Inc., 1975.
15. J. D. Johnston, "A Filter Family Designed for use in Quadrature Mirror Filter Banks," Proc. IEEE Int. Conf. ASSP (April 1980), pp. 291–4.
16. P. Cummiskey, N. S. Jayant, and J. L. Flanagan, "Adaptive Quantization in Differential PCM Coding of Speech," B.S.T.J., *52,* No. 7 (September 1973), pp. 1105–18.
17. D. J. Goodman and R. M. Wilkinson, "A Robust Adaptive Quantizer," IEEE Trans. Commun., *COM-23* (November 1975), pp. 1362–5.
18. L. B. Jackson, "Roundoff-Noise Analysis for Fixed-Point Digital Filters Realized in Cascade or Parallel Form," IEEE Trans. Audio Electroacoust., *AU-18* (June 1970), pp. 107–22.
19. M. G. Bellanger, G. Bonnerot, and M. Coudreuse, "Digital Filtering by Polyphase Network: Application to Sample Rate Alteration of Filter Banks," IEEE Trans. ASSP, *ASSP-24,* No. 2 (April 1976), pp. 109–14.
20. T. A. C. M. Claasen and W. F. G. Mecklenbrauker, "On the Transposition of Linear Time-Varying Discrete-Time Networks and its Application to Multirate Digital Systems," Philips J. Res. *23* (1978), pp. 78–102.