



***Society of Cable
Telecommunications
Engineers***

**ENGINEERING COMMITTEE
Data Standards Subcommittee**

AMERICAN NATIONAL STANDARD

ANSI/SCTE 24-23 2017

**BV32 Speech Codec Specification for
Voice over IP Applications in Cable Telephony**

NOTICE

The Society of Cable Telecommunications Engineers (SCTE) Standards and Operational Practices (hereafter called “documents”) are intended to serve the public interest by providing specifications, test methods and procedures that promote uniformity of product, interchangeability, best practices and ultimately the long term reliability of broadband communications facilities. These documents shall not in any way preclude any member or non-member of SCTE from manufacturing or selling products not conforming to such documents, nor shall the existence of such standards preclude their voluntary use by those other than SCTE members.

SCTE assumes no obligations or liability whatsoever to any party who may adopt the documents. Such adopting party assumes all risks associated with adoption of these documents, and accepts full responsibility for any damage and/or claims arising from the adoption of such documents.

Attention is called to the possibility that implementation of this document may require the use of subject matter covered by patent rights. By publication of this document, no position is taken with respect to the existence or validity of any patent rights in connection therewith. SCTE shall not be responsible for identifying patents for which a license may be required or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Patent holders who believe that they hold patents which are essential to the implementation of this document have been requested to provide information about those patents and any related licensing terms and conditions. Any such declarations made before or after publication of this document are available on the SCTE web site at <http://www.scte.org>.

All Rights Reserved

© Society of Cable Telecommunications Engineers, Inc. 2017
140 Philips Road
Exton, PA 19341

Contents

1	INTRODUCTION.....	5
2	OVERVIEW OF THE BV32 SPEECH CODEC.....	5
2.1	Brief Introduction of Two-Stage Noise Feedback Coding (TSNFC).....	5
2.2	Overview of the BV32 Codec	7
3	DETAILED DESCRIPTION OF THE BV32 ENCODER.....	11
3.1	High-Pass Pre-Filtering	11
3.2	Pre-emphasis Filtering.....	11
3.3	Short-Term Linear Predictive Analysis.....	11
3.4	Conversion to LSP.....	14
3.5	LSP Quantization	16
3.6	Conversion to Short-Term Predictor Coefficients	22
3.7	Short-Term Linear Prediction of Input Signal	23
3.8	Long-Term Linear Predictive Analysis (Pitch Extraction)	24
3.9	Long-Term Predictor Parameter Quantization	31
3.10	Excitation Gain Quantization	32
3.11	Excitation Vector Quantization	37
3.12	Bit Multiplexing	40
4	DETAILED DESCRIPTION OF THE BV32 DECODER.....	42
4.1	Bit De-multiplexing.....	42
4.2	Long-Term Predictor Parameter Decoding.....	42
4.3	Short-Term Predictor Parameter Decoding	42
4.4	Excitation Gain Decoding	45
4.5	Excitation VQ Decoding and Scaling	48
4.6	Long-Term Synthesis Filtering	48
4.7	Short-Term Synthesis Filtering.....	48
4.8	De-emphasis Filtering	49
4.9	Example Packet Loss Concealment	49
	APPENDIX 1: GRID FOR LPC TO LSP CONVERSION	52
	APPENDIX 2: FIRST-STAGE LSP CODEBOOK	54
	APPENDIX 3: SECOND-STAGE LOWER SPLIT LSP CODEBOOK	57
	APPENDIX 4: SECOND-STAGE UPPER SPLIT LSP CODEBOOK	58
	APPENDIX 5: PITCH PREDICTOR TAB CODEBOOK	59
	APPENDIX 6: GAIN CODEBOOK.....	60

APPENDIX 7: GAIN CHANGE THRESHOLD MATRIX	61
APPENDIX 8: EXCITATION VQ SHAPE CODEBOOK	62

Figures

Figure 1 Basic codec structure of Two-Stage Noise Feedback Coding (TSNFC)	Error!
Bookmark not defined.	
Figure 2 Block diagram of the BV32 encoder	Error! Bookmark not defined.
Figure 3 Block diagram of the BV32 decoder	Error! Bookmark not defined.
Figure 4 BV32 short-term linear predictive analysis and quantization (block 10).....	Error!
Bookmark not defined.	
Figure 5 BV32 LSP quantizer (block 16)	Error! Bookmark not defined.
Figure 6 BV32 long-term predictive analysis and quantization (block 20) ...	Error! Bookmark not defined.
Bookmark not defined.	
Figure 7 Prediction residual quantizer (block 30)	Error! Bookmark not defined.
Figure 8 Filter structure used in BV32 excitation VQ codebook search .	Error! Bookmark not defined.
Bookmark not defined.	
Figure 9 BV32 bit stream format.....	Error! Bookmark not defined.
Figure 10 BV32 short-term predictor parameter decoder (block 120)...	Error! Bookmark not defined.
Bookmark not defined.	
Figure 11 Excitation gain decoder	Error! Bookmark not defined.

Tables

Table 1 Bit allocation of the BV32 codec.....	10
--	-----------

1 INTRODUCTION

This document contains the description of the BV32 speech codec¹. BV32 compresses 16 kHz sampled wideband speech to a bit rate of 32 kb/s (kilobits per second) by employing a speech coding algorithm called Two-Stage Noise Feedback Coding (TSNFC), developed by Broadcom.

The rest of this document is organized as follows. Section 2 gives a high-level overview of TSNFC and BV32. Sections 3 and 4 give detailed description of the BV32 encoder and decoder, respectively. The BV32 codec specification given in Sections 3 and 4 contain enough details to allow those skilled in the art to implement bit-stream compatible and functionally equivalent BV32 encoder and decoder.

2 OVERVIEW OF THE BV32 SPEECH CODEC

In this section, the general principles of Two-Stage Noise Feedback Coding (TSNFC) are first introduced. Next, an overview of the BV32 algorithm is given.

2.1 Brief Introduction of Two-Stage Noise Feedback Coding (TSNFC)

In conventional Noise Feedback Coding (NFC), the encoder modifies a prediction residual signal by adding a noise feedback signal to it. A scalar quantizer quantizes this modified prediction residual signal. The difference between the quantizer input and output, or the quantization error signal, is passed through a noise feedback filter. The output signal of this filter is the noise feedback signal added to the prediction residual. The noise feedback filter is used to control the spectrum of the coding noise in order to minimize the perceived coding noise. This is achieved by exploiting the masking properties of the human auditory system.

Conventional NFC codecs typically only use a short-term noise feedback filter to shape the spectral envelope of the coding noise, and a scalar quantizer is used universally. In contrast, Broadcom's Two-Stage Noise Feedback Coding (TSNFC) system uses a codec structure employing two stages of noise feedback coding in a nested loop: the first NFC stage performs short-term prediction and short-term noise spectral shaping (spectral envelope shaping), and the second nested NFC stage performs long-term prediction and long-term noise spectral shaping (harmonic shaping). Such a nested two-stage NFC structure is shown in Figure 1 below.

¹ The "BV32 speech codec" specification is based on Broadcom Corporation's BroadVoice®32 speech codec. The BroadVoice® open source software is provided under the GNU Lesser General Public License ("LGPL"), version 2.1, as published by the Free Software Foundation.

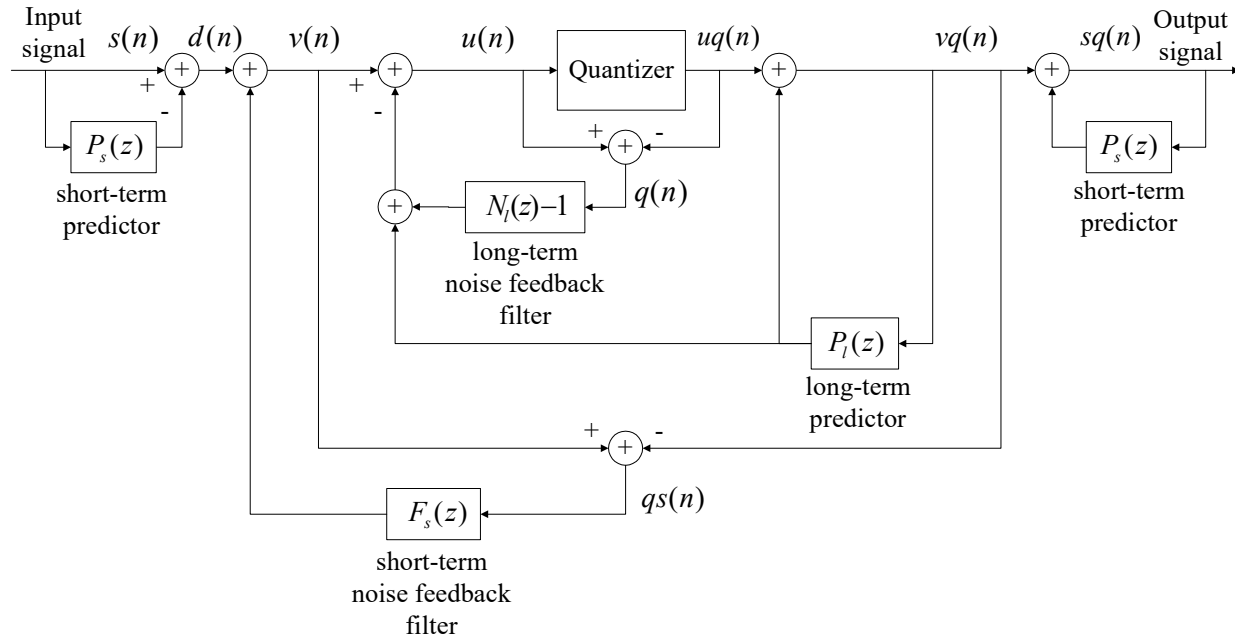


Figure 1 Basic codec structure of Two-Stage Noise Feedback Coding (TSNFC)

In Figure 1 above, the outer layer (including the two short-term predictors and the short-term noise feedback filter) follows the structure of the conventional NFC codec. The TSNFC structure in Figure 1 is obtained by replacing the simple scalar quantizer in the conventional (single-stage) NFC structure by a “predictive quantizer” that employs long-term prediction and long-term noise spectral shaping. This “predictive quantizer” is represented by the inner feedback loop in Figure 1, including the long-term predictor and long-term noise feedback filter. This inner feedback loop uses an alternative but equivalent conventional NFC structure, where $N_l(z)$ represents the filter whose frequency response is the desired noise shape for long-term noise spectral shaping. In the outer layer, the short-term noise feedback filter $F_s(z)$ is usually chosen as a bandwidth-expanded version of the short-term predictor $P_s(z)$. The choice of different NFC structures in the outer and inner layers is based on complexity consideration. By combining two stages of NFC in a nested loop, the TSNFC in Figure 1 can reap the benefits of both short-term and long-term prediction and also achieve short-term and long-term noise spectral shaping at the same time.

It is natural and straightforward to use a scalar quantizer in Figure 1. However, to achieve better coding efficiency, a vector quantizer is used in BV32. In the Vector Quantization (VQ) codebook search, the $u(n)$ vector cannot be generated before the VQ codebook search starts. Due to the feedback structure in Figure 1, the elements of $u(n)$ from the second element on will depend on the vector-quantized version of earlier elements. Therefore, the VQ codebook search is performed by trying out each of the candidate codevectors in the VQ codebook (i.e. fixing a candidate $uq(n)$ vector first), calculating the corresponding $u(n)$ vector and the corresponding VQ error $q(n) = u(n) - uq(n)$. The VQ codevector that minimizes the energy of $q(n)$ within the current

vector time span is chosen as the winning codevector, and the corresponding codebook index becomes part of the encoder output bit stream for the current speech frame.

The TSNFC decoder structure is simply a quantizer decoder followed by the two feedback filter structures involving the long-term predictor and the short-term predictor, respectively, shown on the right half of Figure 1. Thus, the TSNFC decoder is similar to the decoders of other predictive coding techniques such as Adaptive Predictive Coding (APC), Multi-Pulse Linear Predictive Coding (MPLPC), and Code-Excited Linear Prediction (CELP).

2.2 Overview of the BV32 Codec

The BV32 codec is a purely forward-adaptive TSNFC codec. It operates at an input sampling rate of 16 kHz and an encoding bit rate of 32 kb/s, or 2 bits per sample. The BV32 uses a frame size of 5 ms, or 80 samples. There is no look ahead. Therefore, the total algorithmic buffering delay is just the frame size itself, or 5 ms. The main design goal of BV32 is to make the coding delay and the codec complexity as low as possible, while maintaining essentially transparent output speech quality.

Due to the small frame size, the parameters of the short-term predictor (also called the “LPC predictor”) and the long-term predictor (also called the “pitch predictor”) are both transmitted and updated once a frame. Each 5 ms frame is divided equally into two 2.5 ms sub-frames (40 samples each). The gain of the excitation signal is transmitted once every sub-frame. The excitation VQ uses a vector dimension of 4 samples. Hence, there are 10 excitation vectors in a sub-frame, and 20 vectors in a frame. Figure 2 shows a block diagram of the BV32 encoder. More detailed description of each functional block will be given in Section 3.

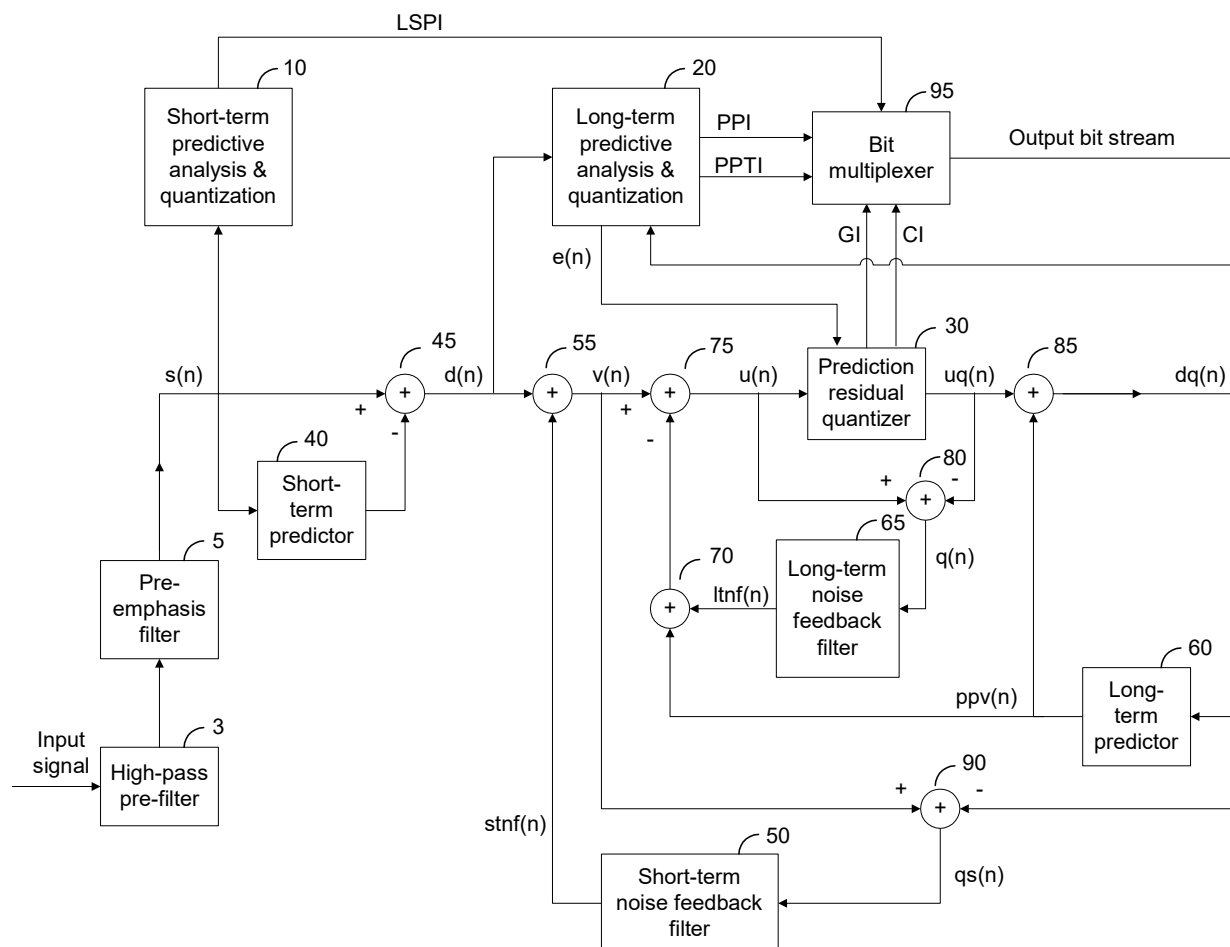


Figure 2 Block diagram of the BV32 encoder

The BV32 encoder first passes the input signal through a fixed pole-zero high-pass pre-filter to remove possible DC bias or low frequency rumble. The filtered signal is then further passed through a fixed pole-zero pre-emphasis filter that provides a general high-pass spectral tilt. The resulting pre-emphasized signal is then used to derive the LPC predictor coefficients.

To keep the complexity low, the BV32 uses a relatively low LPC predictor order of 8, and the LPC analysis window is only 10 ms (160 samples) long. The LPC analysis window is asymmetric, with the peak of the window located at the center of the current frame, and the end of the window coinciding with the last sample of the current frame. Autocorrelation LPC analysis based on Levinson-Durbin recursion is used to derive the coefficients of the 8th-order LPC predictor. The derived LPC predictor coefficients are converted to Line-Spectrum Pair (LSP) parameters, which are then quantized by an inter-frame predictive coding scheme.

The inter-frame prediction of LSP parameters uses an 8th-order moving-average (MA) predictor. The MA predictor coefficients are fixed. The time span that this MA predictor covers is $8 \times 5 \text{ ms} = 40 \text{ ms}$. The inter-frame LSP prediction residual is quantized by a two-stage vector quantizer. The first stage employs an 8-dimensional vector quantizer with a 7-bit codebook. The second stage uses a

split vector quantizer with a 3-5 split and 5 bits each. That is, the first three elements are vector quantized to 5 bits, and the remaining 5 elements are also quantized to 5 bits.

For long-term prediction, a three-tap pitch predictor with an integer pitch period is used. To keep the complexity low, the pitch period and the pitch taps are both determined in an open-loop fashion. The three pitch predictor taps are jointly quantized using a 5-bit vector quantizer. The distortion measure used in the codebook search is the energy of the open-loop pitch prediction residual. The 32 codevectors in the pitch tap codebook have been “stabilized” to make sure that they will not give rise to an unstable pitch synthesis filter.

The excitation gain is also determined in an open-loop fashion to keep the complexity low. The average power of the open-loop pitch prediction residual within the current sub-frame is calculated and converted to the logarithmic domain. The resulting log-gain is then quantized using inter-subframe MA predictive coding. The MA predictor order for the log-gain is 16, corresponding to a time span of $16 \times 2.5 = 40$ ms. Again, the log-gain MA predictor coefficients are fixed. The log-gain prediction residual is quantized by a 5-bit scalar quantizer.

The 4-dimensional excitation VQ codebook has a simple sign-shape structure, with 1 bit for sign, and 5 bits for shape. In other words, only 32 four-dimensional codevectors are stored, but the mirror image of each codevector with respect to the origin is also a codevector.

In the BV32 decoder, the decoded excitation vectors are scaled by the excitation gain. The scaled excitation signal passes through a long-term synthesis filter and a short-term synthesis filter, and finally through a fixed pole-zero de-emphasis filter which is the inverse filter of the pre-emphasis filter in the encoder. Figure 3 shows the block diagram of the BV32 decoder.

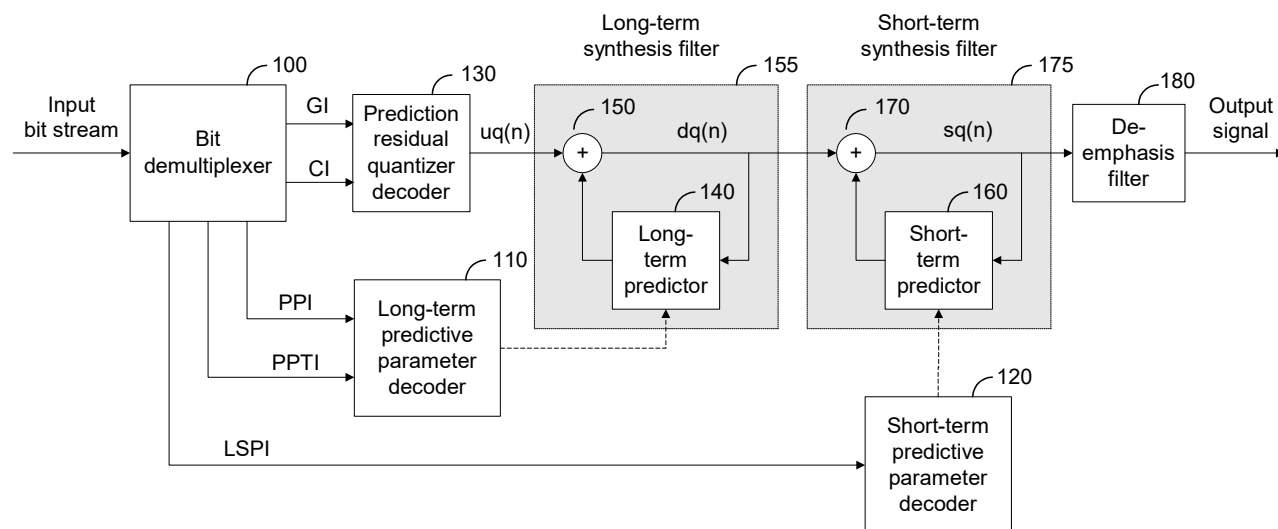


Figure 3 Block diagram of the BV32 decoder

Table 1 shows the bit allocation of BV32 in each 5 ms frame. The LSP parameters are encoded into 17 bits per frame, including 7 bits for the first-stage VQ, and $5 + 5 = 10$ bits for the second-stage split VQ. The pitch period and pitch predictor taps are encoded into 8 and 5 bits, respectively. The two excitation gains in each frame are encoded into $5 + 5 = 10$ bits. The 20 excitation vectors are each encoded with 1 bit for sign and 5 bits for shape, resulting in 120 bits per frame for excitation VQ. Including the other 40 bits of side information, the grand total is 160 bits per 80-sample frame, which is 2 bits/sample, or 32 kb/s.

Parameter	Bits per frame (80 samples)
LSP	$7 + (5 + 5) = 17$
Pitch Period	8
3 Pitch Predictor Taps	5
2 Excitation Gains	$5 + 5 = 10$
20 Excitation Vectors	$(1 + 5) \times 20 = 120$
Total	160

Table 1 Bit allocation of the BV32 codec

3 DETAILED DESCRIPTION OF THE BV32 ENCODER

In this section, detailed description of each functional block of the BV32 encoder in Figure 2 is given. When necessary, certain functional blocks will be expanded into more detailed block diagrams. The description given in this section will be in sufficient detail that will allow those skilled in the art to implement a mathematically equivalent BV32 encoder.

3.1 High-Pass Pre-Filtering

Refer to Figure 2. The input signal is assumed to be represented by 16-bit linear PCM. Block 3 is a high-pass pre-filter with fixed coefficients. It is a first-order pole-zero filter with the following transfer function.

$$H_{hpf}(z) = \frac{(255/256)(1 - z^{-1})}{1 - (127/128)z^{-1}}$$

This high-pass pre-filter filters the input signal to remove undesirable low-frequency components, and passes the filtered signal to the pre-emphasis filter (block 5).

3.2 Pre-emphasis Filtering

Block 5 is a first-order pole-zero pre-emphasis filter with fixed coefficients. It has the following transfer function.

$$H_{pe}(z) = \frac{1 + 0.5z^{-1}}{1 + 0.75z^{-1}}$$

It filters the high-pass pre-filtered signal (the output signal of block 3) and gives an output signal denoted as $s(n)$ in Figure 2, where n is the sample index.

3.3 Short-Term Linear Predictive Analysis

The high-pass filtered and pre-emphasized signal $s(n)$ is buffered at block 10, which performs short-term linear predictive analysis and quantization to obtain the coefficients for the short-term predictor 40 and the short-term noise feedback filter 50. This block 10 is further expanded in Figure 4.

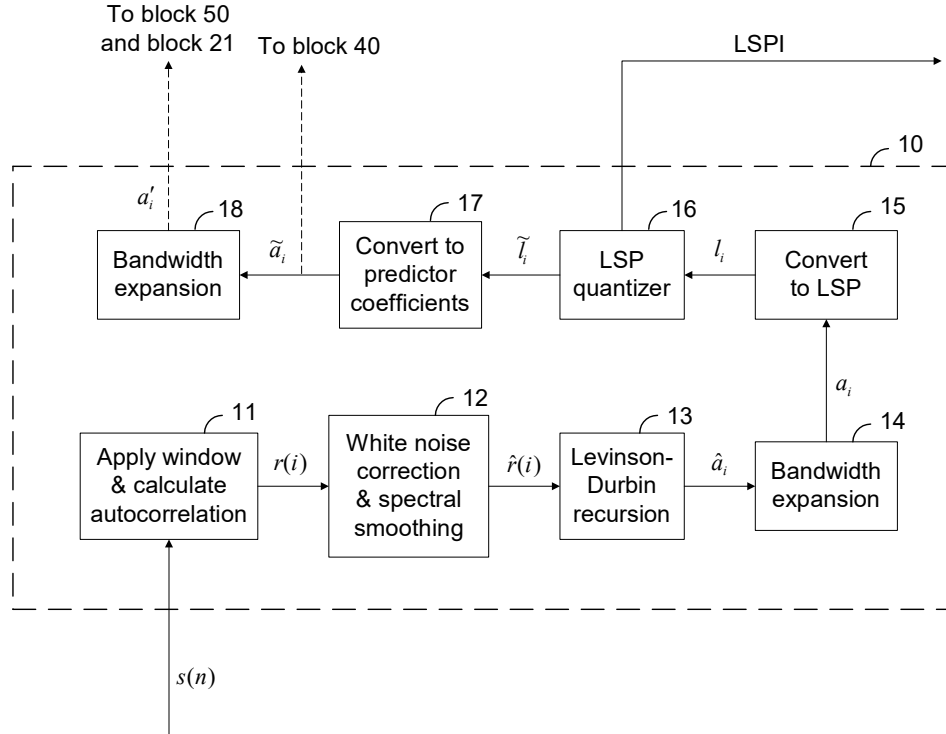


Figure 4 BV32 short-term linear predictive analysis and quantization (block 10)

Refer to Figure 4. The input signal $s(n)$ is buffered in block 11, where a 10 ms asymmetric analysis window is applied to the buffered $s(n)$ signal array. The “left window” is 7.5 ms long, and the “right window” is 2.5 ms long. Let $LWINSZ$ be the number of samples in the left window ($LWINSZ = 120$ for 16 kHz sampling), then the left window is given by

$$wl(n) = \frac{1}{2} \left[1 - \cos \left(\frac{n\pi}{LWINSZ + 1} \right) \right], n = 1, 2, \dots, LWINSZ.$$

Let $RWINSZ$ be the number of samples in the right window. Then, $RWINSZ = 40$ for 16 kHz sampling. The right window is given by

$$wr(n) = \cos \left(\frac{(n-1)\pi}{2RWINSZ} \right), n = 1, 2, \dots, RWINSZ.$$

The concatenation of $wl(n)$ and $wr(n)$ gives the 10 ms asymmetrical analysis window, with the peak of the window located at the center of the current frame. When applying this analysis window, the last sample of the window is lined up with the last sample of the current frame. Therefore, the codec does not use any look ahead.

More specifically, without loss of generality, let the sampling time index range of $n = 1, 2, \dots, FRSZ$ corresponds to the current frame, where the frame size $FRSZ = 80$ for BV32. Then, the $s(n)$ signal

buffer stored in block 11 is for $n = -79, -78, \dots, -1, 0, 1, 2, \dots, 80$. The asymmetrical LPC analysis window function can be expressed as

$$w(n) = \begin{cases} wl(n+80), & n = -79, -78, \dots, 40 \\ wr(n-40), & n = 41, 42, \dots, 80 \end{cases}.$$

The windowing operation is performed as follows.

$$s_w(n) = s(n)w(n), \quad n = -79, -78, \dots, -1, 0, 1, 2, \dots, 80.$$

Next, block 11 calculates the autocorrelation coefficients as follows.

$$r(i) = \sum_{n=-79+i}^{80} s_w(n)s_w(n-i), \quad i = 0, 1, 2, \dots, 8.$$

The calculated autocorrelation coefficients are passed to block 12, which applies a Gaussian window to the autocorrelation coefficients to perform spectral smoothing. The Gaussian window function is given by

$$gw(i) = e^{\frac{-(2\pi i \sigma / f_s)^2}{2}}, \quad i = 1, 2, \dots, 8,$$

where f_s is the sampling rate of the input signal, expressed in Hz, and σ is 40 Hz.

After multiplying the $r(i)$ array by such a Gaussian window, block 12 then multiplies $r(0)$ by a white noise correction factor of $WNCF = 1 + \varepsilon$, where $\varepsilon = 0.0001$. In summary, the output of block 12 is given by

$$\hat{r}(i) = \begin{cases} 1.0001 \times r(0), & i = 0 \\ gw(i)r(i), & i = 1, 2, \dots, 8 \end{cases}$$

Block 13 performs the Levinson-Durbin recursion to convert the autocorrelation coefficients $\hat{r}(i)$ to the short-term predictor coefficients $\hat{a}_i$, $i = 0, 1, \dots, 8$. If the Levinson-Durbin recursion exits prematurely before the recursion is completed (for example, because the prediction residual energy $E(i)$ is less than zero), then the short-term predictor coefficients of the last frame is also used in the current frame. To do the exception handling this way, there needs to be an initial value of the $\hat{a}_i$ array. The initial value of the $\hat{a}_i$ array is set to $\hat{a}_0 = 1$ and $\hat{a}_i = 0$ for $i = 1, 2, \dots, 8$. The Levinson-Durbin recursion is performed in the following algorithm.

1. If $\hat{r}(0) \leq 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson-Durbin recursion.
2. $E(0) = \hat{r}(0)$
3. $k_1 = -\hat{r}(1) / \hat{r}(0)$
4. $\hat{a}_1^{(1)} = k_1$

5. $E(1) = (1 - k_1^2)E(0)$
6. If $E(1) \leq 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson-Durbin recursion.
7. For $i = 2, 3, 4, \dots, 8$, do the following

$$k_i = \frac{-\hat{r}(i) - \sum_{j=1}^{i-1} \hat{a}_j^{(i-1)} \hat{r}(i-j)}{E(i-1)}$$

$$\hat{a}_i^{(i)} = k_i$$

$$\hat{a}_j^{(i)} = \hat{a}_j^{(i-1)} + k_i \hat{a}_{i-j}^{(i-1)}, \text{ for } j = 1, 2, \dots, i-1$$

$$E(i) = (1 - k_i^2)E(i-1)$$

If $E(i) \leq 0$, use the $\hat{a}_i$ array of the last frame, and exit the Levinson - Durbin recursion.

If the recursion is exited pre-maturely, the $\hat{a}_i$ array of the last frame is used as the output of block 13. If the recursion is completed successfully (which is normally the case), then the final output of block 13 is taken as

$$\begin{aligned} \hat{a}_0 &= 1 \\ \hat{a}_i &= \hat{a}_i^{(8)}, \text{ for } i = 1, 2, \dots, 8 \end{aligned}$$

Block 14 performs bandwidth expansion as follows

$$a_i = (0.96852)^i \hat{a}_i, \text{ for } i = 0, 1, \dots, 8.$$

3.4 Conversion to LSP

In Figure 4, block 15 converts the LPC coefficients $a_i, i = 1, 2, \dots, 8$ of the prediction error filter given by

$$A(z) = 1 + \sum_{i=1}^8 a_i z^{-i}$$

to a set of 8 Line-Spectrum Pair (LSP) coefficients $l_i, i = 1, 2, \dots, 8$. The LSP coefficients, also known as the Line Spectrum Frequencies (LSF), are the angular positions normalized to 1, i.e. 1.0 corresponds to the Nyquist frequency, of the roots of

$$A_p(z) = A(z) + z^{-9} A(z^{-1})$$

and

$$A_m(z) = A(z) - z^{-9} A(z^{-1})$$

on the upper half of the unit circle, $z = e^{j\omega}$, $0 \leq \omega \leq \pi$, less the trivial roots in $z = -1$ and $z = 1$ of $A_p(z)$ and $A_m(z)$, respectively. Due to the symmetry and anti-symmetric of $A_p(z)$ and $A_m(z)$, respectively, the roots of interest can be determined as the roots of

$$G_p(\omega) = \sum_{i=0}^4 g_{p,i} \cos(i\omega)$$

and

$$G_m(\omega) = \sum_{i=0}^4 g_{m,i} \cos(i\omega)$$

where

$$g_{p|m,i} = \begin{cases} f_{p|m,4} & i = 0 \\ 2f_{p|m,4-i} & i = 1, \dots, 4 \end{cases}$$

in which

$$f_{p,i} = \begin{cases} 1.0 & i = 0 \\ a_i + a_{9-i} - f_{p,i-1} & i = 1, \dots, 4 \end{cases}$$

and

$$f_{m,i} = \begin{cases} 1.0 & i = 0 \\ a_i - a_{9-i} + f_{p,i-1} & i = 1, \dots, 4 \end{cases}$$

The subscript "p|m" means dual versions of the equation exist, with either subscript "p" or subscript "m". The roots of $A_p(z)$ and $A_m(z)$, and therefore the roots of $G_p(\omega)$ and $G_m(\omega)$, are interlaced, with the first root belonging to $G_p(\omega)$. The evaluation of the functions $G_p(\omega)$ and $G_m(\omega)$ are carried out efficiently using Chebyshev polynomial series. With the mapping $x = \cos(\omega)$,

$$\cos(m\omega) = T_m(x)$$

where $T_m(x)$ is the m^{th} -order Chebyshev polynomial, the two functions $G_p(\omega)$ and $G_m(\omega)$ can be expressed as

$$G_{p|m}(x) = \sum_{i=0}^4 g_{p|m,i} T_i(x).$$

Due to the recursive nature of Chebyshev polynomials the functions can be evaluated as

$$G_{p|m}(x) = \frac{b_{p|m,0}(x) - b_{p|m,2}(x) + g_{p|m,0}}{2}$$

where $b_{p|m,0}(x)$ and $b_{p|m,2}(x)$ are calculated using the following recurrence

$$b_{p|m,i}(x) = 2x b_{p|m,i+1}(x) - b_{p|m,i+2}(x) + g_{p|m,i}$$

with initial conditions $b_{p|m,5}(x) = b_{p|m,6}(x) = 0$.

The roots of $G_p(x)$ and $G_m(x)$ are determined in an alternating fashion starting with a root in $G_p(x)$. Each root of $G_p(x)$ and $G_m(x)$ is located by identifying a sign change of the relevant function along a grid of 60 points, given in Appendix 1. The estimation of the root is then refined using 4 bisections followed by a final linear interpolation between the two points surrounding the root. It should be noted that the roots and grid points are in the cosine domain. Once the 8 roots

$$x_i = \cos(\omega_i), \quad i = 1, 2, \dots, 8$$

are determined in the cosine domain, they are converted to the normalized frequency domain according to

$$l_i = \cos^{-1}(x_i)/\pi, \quad i = 1, 2, \dots, 8$$

in order to obtain the LSP coefficients. In the rare event that less than 8 roots are found the routine returns the LSP coefficients of the previous frame, $l_i(k-1), i = 1, 2, \dots, 8$, where the additional parameter k represents the frame index of the current frame. The LSP coefficients of the previous frame at the very first frame are initialized to

$$l_i(0) = i/9, \quad i = 1, 2, \dots, 8.$$

3.5 LSP Quantization

Block 16 of Figure 4 vector quantizes and encodes the LSP coefficient vector, $\mathbf{l} = [l_1 \ l_2 \ \dots \ l_8]^T$, to a total of 17 bits. The output LSP quantizer index array, $LSPI = \{LSPI_1, LSPI_2, LSPI_3\}$, is passed

to the bit multiplexer (block 95), while the quantized LSP coefficient vector, $\tilde{\mathbf{I}} = [\tilde{l}_1 \quad \tilde{l}_2 \quad \dots \quad \tilde{l}_8]^T$, is passed to block 17.

The LSP quantizer is based on mean-removed inter-frame moving-average (MA) prediction with two-stage vector quantization (VQ) of the prediction error. The quantizer enables bit-error detection at the decoder by constraining the codevector selection at the encoder. It should be noted that the encoder must perform the specified constrained VQ in order to maintain interoperability properly. The first-stage VQ is searched using the simple mean-squared error (MSE) distortion criterion, while both lower and upper splits of the second-stage split VQ are searched using the weighted mean-square error (WMSE) distortion criterion.

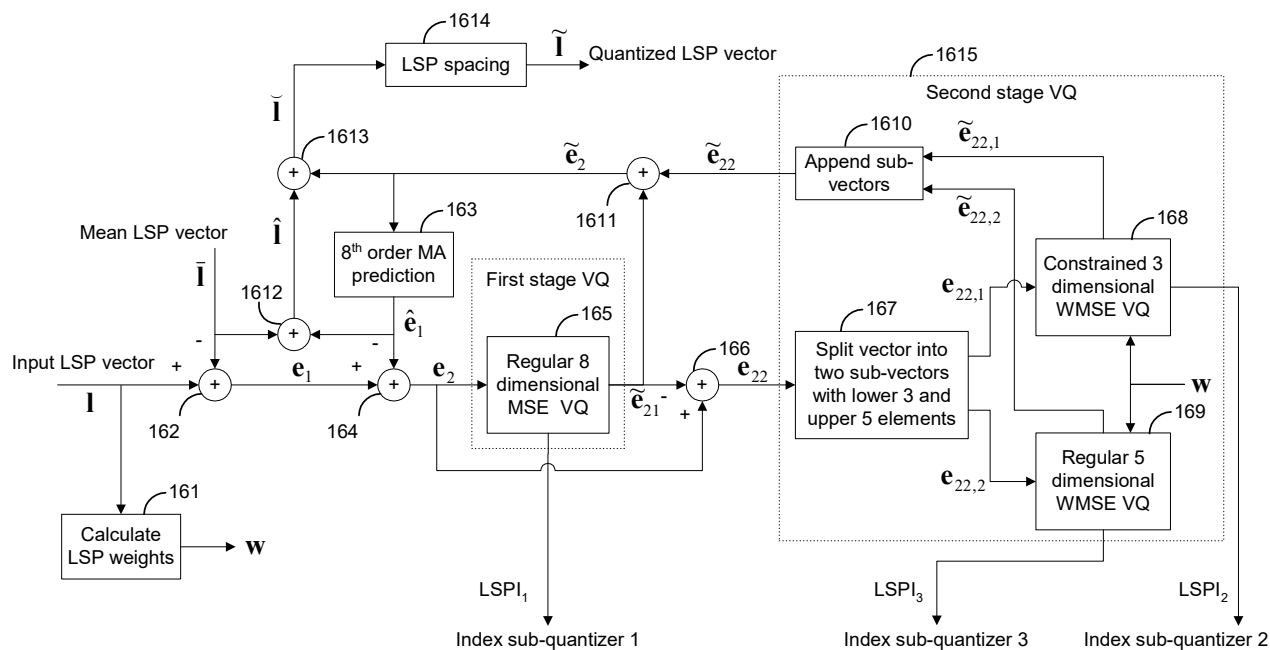


Figure 5 BV32 LSP quantizer (block 16)

Block 16 is further expanded in Figure 5. The first stage VQ takes place in block 165, and the second stage split VQ takes place in block 1615. Except for the LSP quantizer indices $LSPI_1, LSPI_2, LSPI_3$ and split vectors, all signal paths in Figure 5 are for vectors of dimension 8. Block 161 uses the unquantized LSP coefficient vector to calculate the weights to be used later in the second-stage WMSE VQs. The weights are determined as

$$w_i = \begin{cases} 1/(l_2 - l_1), & i = 1 \\ 1/\min(l_i - l_{i-1}, l_{i+1} - l_i), & 1 < i < 8 \\ 1/(l_M - l_{M-1}), & i = 8 \end{cases}.$$

Basically, the i -th weight is the inverse of the distance between the i -th LSP coefficient and its nearest neighbor LSP coefficient.

Adder 162 subtracts the constant LSP mean vector,

$$\bar{\mathbf{l}} = [0.0551453 \ 0.1181030 \ 0.2249756 \ 0.3316040 \ 0.4575806 \ 0.5720825 \ 0.7193298 \ 0.8278198]^T,$$

from the unquantized LSP coefficient vector to get the mean-removed LSP vector,

$$\mathbf{e}_1 = \mathbf{l} - \bar{\mathbf{l}}.$$

In Figure 5, block 163 performs 8th order inter-frame MA prediction of the mean-removed LSP vector $\mathbf{e}_1$ based on the $\tilde{\mathbf{e}}_2$ vectors in the previous 8 frames, where $\tilde{\mathbf{e}}_2$ is the quantized version of the inter-frame LSP prediction error vector². Let $\tilde{e}_{2,i}(k)$ denote the i -th element of the vector $\tilde{\mathbf{e}}_2$ in the frame that is k frames before the current frame. Let $\hat{e}_{1,i}$ be the i -th element of the inter-frame-predicted mean-removed LSP vector $\hat{\mathbf{e}}_1$. Then, block 163 calculates the predicted LSP vector according to

$$\hat{e}_{1,i} = \mathbf{p}_{LSP,i}^T \cdot [\tilde{e}_{2,i}(1) \ \tilde{e}_{2,i}(2) \ \tilde{e}_{2,i}(3) \ \tilde{e}_{2,i}(4) \ \tilde{e}_{2,i}(5) \ \tilde{e}_{2,i}(6) \ \tilde{e}_{2,i}(7) \ \tilde{e}_{2,i}(8)]^T, \ i = 1, 2, \dots, 8,$$

where $\mathbf{p}_{LSP,i}$ holds the 8 prediction coefficients for the i -th LSP coefficient and is given by

$$\begin{aligned} \mathbf{p}_{LSP,1}^T &= [0.7401123 \ 0.6939697 \ 0.6031494 \ 0.5333862 \ 0.4295044 \ 0.3234253 \ 0.2177124 \ 0.1162720] \\ \mathbf{p}_{LSP,2}^T &= [0.7939453 \ 0.7693481 \ 0.6712036 \ 0.5919189 \ 0.4750366 \ 0.3556519 \ 0.2369385 \ 0.1181030] \\ \mathbf{p}_{LSP,3}^T &= [0.7534180 \ 0.7318115 \ 0.6326294 \ 0.5588379 \ 0.4530029 \ 0.3394775 \ 0.2307739 \ 0.1201172] \\ \mathbf{p}_{LSP,4}^T &= [0.7188110 \ 0.6765747 \ 0.5792847 \ 0.5169067 \ 0.4223022 \ 0.3202515 \ 0.2235718 \ 0.1181030] \\ \mathbf{p}_{LSP,5}^T &= [0.6431885 \ 0.6023560 \ 0.5112305 \ 0.4573364 \ 0.3764038 \ 0.2803345 \ 0.2060547 \ 0.1090698] \\ \mathbf{p}_{LSP,6}^T &= [0.5687866 \ 0.5837402 \ 0.4616089 \ 0.4351196 \ 0.3502808 \ 0.2602539 \ 0.1951294 \ 0.0994263] \\ \mathbf{p}_{LSP,7}^T &= [0.5292969 \ 0.4835205 \ 0.3890381 \ 0.3581543 \ 0.2882080 \ 0.2261353 \ 0.1708984 \ 0.0941162] \\ \mathbf{p}_{LSP,8}^T &= [0.5134277 \ 0.4365845 \ 0.3521729 \ 0.3118896 \ 0.2514038 \ 0.1951294 \ 0.1443481 \ 0.0841064] \end{aligned}$$

Adder 164 calculates the prediction error vector

$$\mathbf{e}_2 = \mathbf{e}_1 - \hat{\mathbf{e}}_1,$$

which is the input to the first-stage VQ. In block 165 the 8-dimensional prediction error vector, $\mathbf{e}_2$, is vector quantized with the 128-entry, 8-dimensional codebook, $\mathbf{CB}_1 = \{\mathbf{cb}_1^{(0)}, \mathbf{cb}_1^{(1)}, \dots, \mathbf{cb}_1^{(127)}\}$, listed in Appendix 2. The codevector minimizing the MSE is denoted $\tilde{\mathbf{e}}_{21}$ and the corresponding index is denoted $LSP I_1$:

² At the first frame, the previous, non-existing, quantized interframe LSP prediction error vectors are set to zero-vectors.

$$LSPI_1 = \arg \min_{k \in \{0,1,\dots,127\}} \left\{ \left(\mathbf{e}_2 - \mathbf{cb}_1^{(k)} \right)^T \left(\mathbf{e}_2 - \mathbf{cb}_1^{(k)} \right) \right\},$$

$$\tilde{\mathbf{e}}_{21} = \mathbf{cb}_1^{(LSPI_1)},$$

where the notation $I = \arg \min_i \{D(i)\}$ means that I is the argument that minimizes the entity $D(i)$, i.e.

$$D(I) \leq D(i) \text{ for all } i.$$

Adder 166 subtracts the first-stage codevector from the prediction error vector to form the quantization error vector of the first stage,

$$\mathbf{e}_{22} = \mathbf{e}_2 - \tilde{\mathbf{e}}_{21}.$$

This is the input to the second-stage VQ, which is a two-split VQ. Block 167 splits the quantization error vector of the first stage into a lower sub-vector, $\mathbf{e}_{22,1}$, with the first 3 elements of $\mathbf{e}_{22}$,

$$\mathbf{e}_{22,1} = [e_{22,1} \quad e_{22,2} \quad e_{22,3}]^T,$$

and an upper sub-vector, $\mathbf{e}_{22,2}$, with the last 5 elements of $\mathbf{e}_{22}$,

$$\mathbf{e}_{22,2} = [e_{22,4} \quad e_{22,5} \quad e_{22,6} \quad e_{22,7} \quad e_{22,8}]^T.$$

The two sub-vectors are quantized independently into $\tilde{\mathbf{e}}_{22,1}$ and $\tilde{\mathbf{e}}_{22,2}$, respectively.

Block 168 performs a constrained VQ of the 3-dimensional vector, $\mathbf{e}_{22,1}$, using the 32-entry codebook, $\mathbf{CB}_{21} = \{\mathbf{cb}_{21}^{(0)}, \mathbf{cb}_{21}^{(1)}, \dots, \mathbf{cb}_{21}^{(31)}\}$, of Appendix 3. The codevector that minimizes the WMSE subject to the constraint that the 3 first elements of the intermediate quantized LSP vector,

$$\begin{aligned} \tilde{\mathbf{l}} &= \hat{\mathbf{l}} + \tilde{\mathbf{e}}_2 \\ &= \bar{\mathbf{l}} + \hat{\mathbf{e}}_1 + \tilde{\mathbf{e}}_{21} + \begin{bmatrix} \tilde{\mathbf{e}}_{22,1} \\ \tilde{\mathbf{e}}_{22,2} \end{bmatrix}, \end{aligned}$$

preserve the ordering property