Society of Cable Telecommunications Engineers

ENGINEERING COMMITTEE
Data Standards Subcommittee

SCTE STANDARD

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.

140 Philips Road
Exton, PA 19341

Contents

Figures

Tables

1 INTRODUCTION

This document is identical to SCTE 24-23 2012 except for informative components which may have been updated such as the title page, NOTICE text, headers and footers. No normative changes have been made to this document.

This document contains the description of the BV32 speech codec[1]. 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.

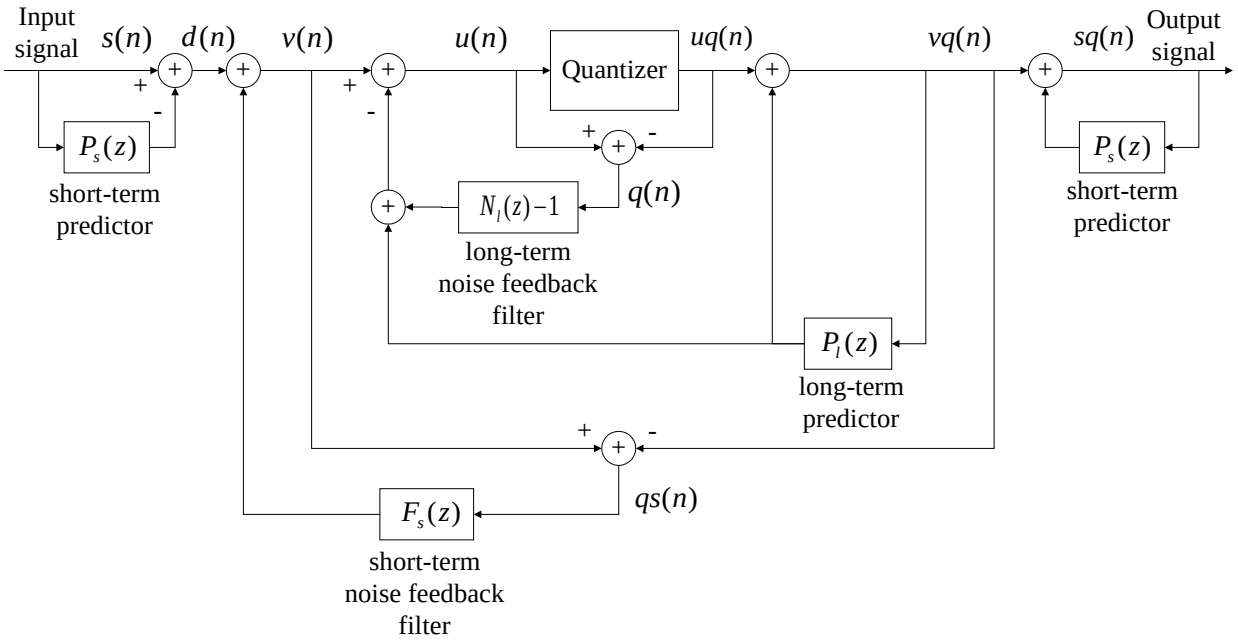

Figure 1 Basic codec structure of Two-Stage Noise Feedback Coding (TSNFC)

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.

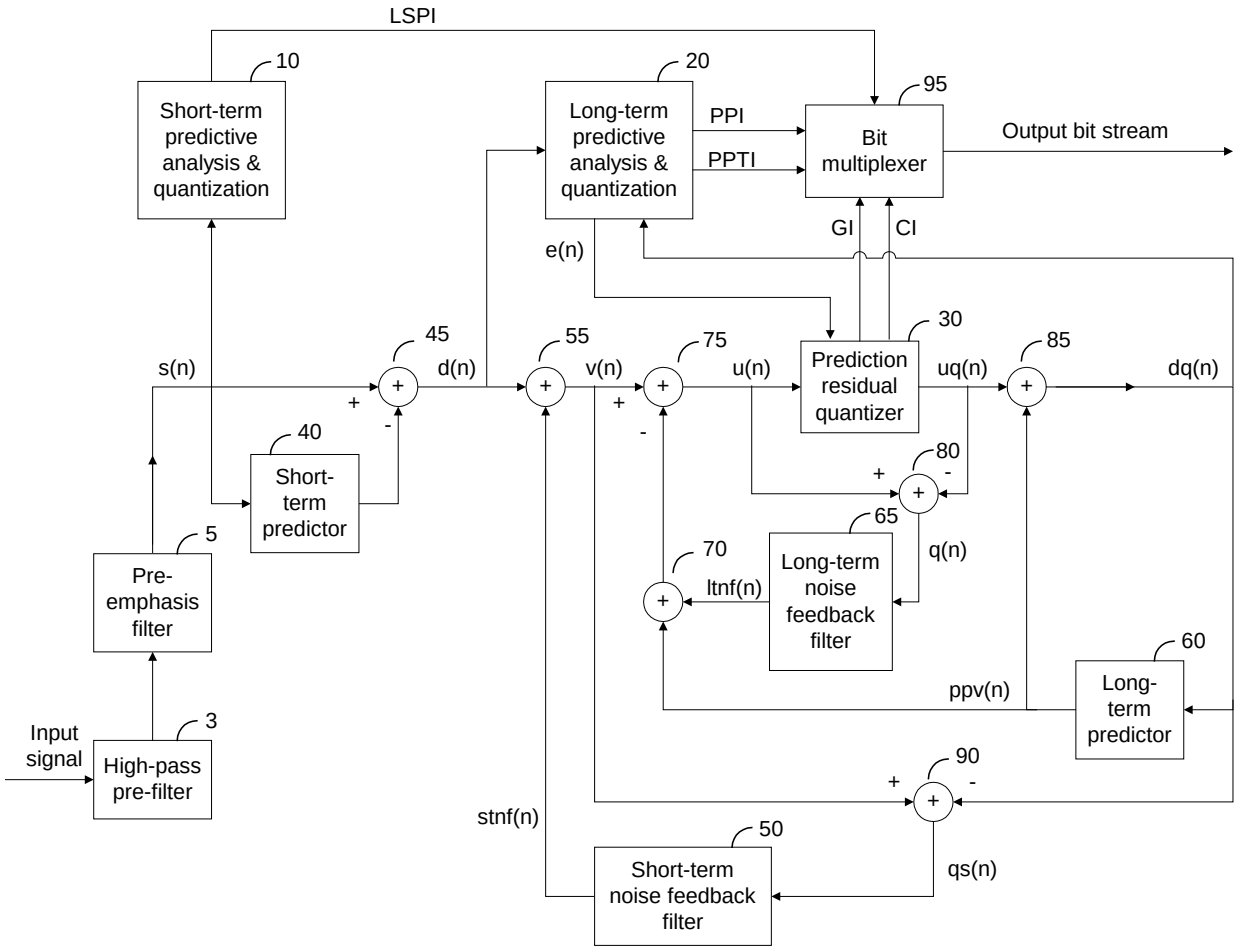

Figure 2 Block diagram of the BV32 encoder

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 8^{th}-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 8^{th}-order moving-average (MA) predictor. The MA predictor coefficients are fixed. The time span that this MA predictor covers is 8 × 5 ms = 40 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 × 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.

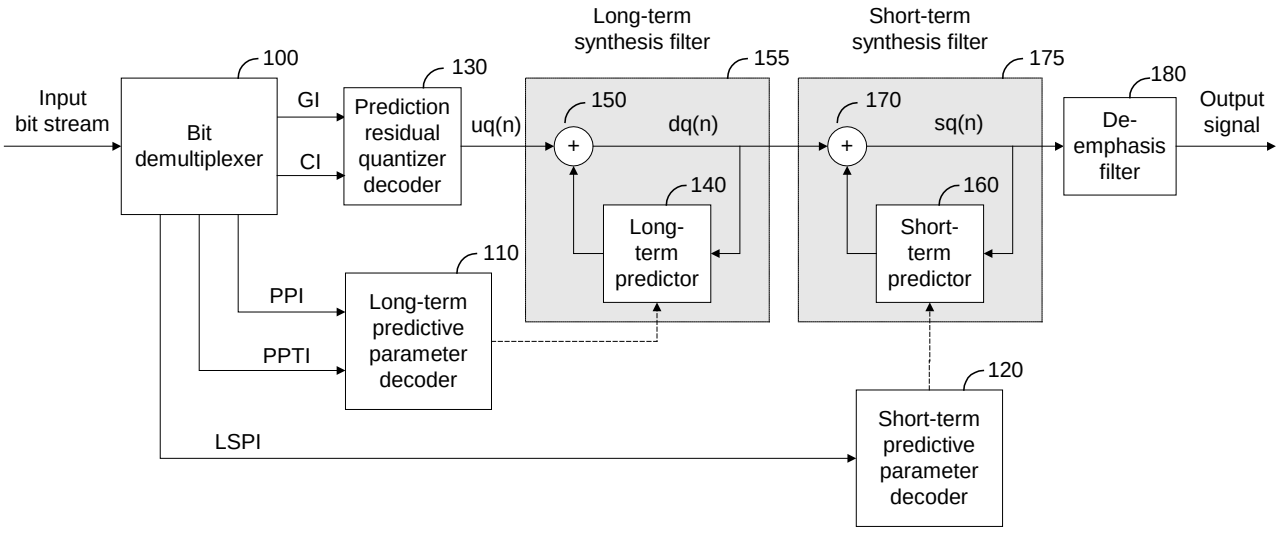

Figure 3 Block diagram of the BV32 decoder

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) × 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.

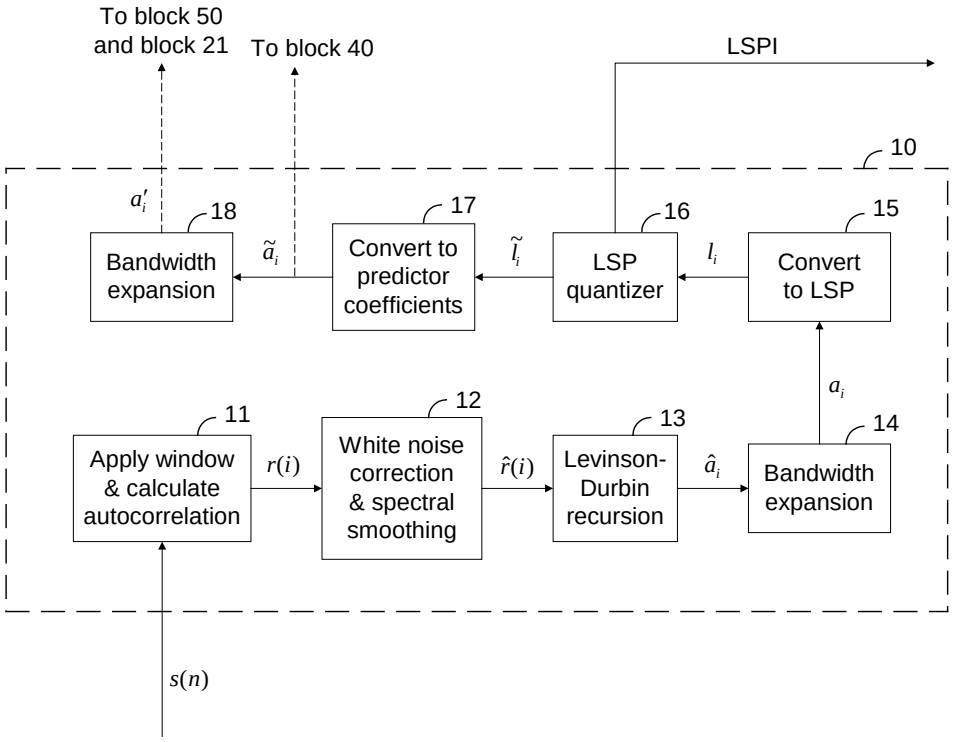

Figure 4 BV32 short-term linear predictive analysis and quantization (block 10)

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

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

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

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

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 $n = 1, 2, \ldots, FRSZ$ 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, …, -1, 0, 1, 2, …, 80. The asymmetrical LPC analysis window function can be expressed as

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

The windowing operation is performed as follows.

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

Next, block 11 calculates the autocorrelation coefficients as follows.

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

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

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

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 ε = 0.0001. In summary, the output of block 12 is given by

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

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, …, 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, …, 8. The Levinson-Durbin recursion is performed in the following algorithm.

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

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

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

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

$$E(i) = (1 - k_i^{\,2})E(i-1)$$

If $E(i) \le 0$, 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

$$\hat{a}_0 = 1$$

$$\hat{a}_i = \hat{a}_i^{\,(8)}, \text{ for } i = 1, 2, \ldots, 8$$

Block 14 performs bandwidth expansion as follows

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

3.4 Conversion to LSP

In Figure 4, block 15 converts the LPC coefficients $a_i, i = 1, 2, \ldots, 8$ 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 $l_i, i = 1, 2, \ldots, 8$. 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, $z = e^{j\omega}, 0 \le \omega \le \pi$, 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

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

in which

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

and

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

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

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

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

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

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, $l_i(k-1), i = 1, 2, \ldots, 8$, 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

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

3.5 LSP Quantization

Block 16 of Figure 4 vector quantizes and encodes the LSP coefficient vector, $\mathbf{l} = [l_1 \quad l_2 \quad \ldots \quad l_8]^T$, 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, $\tilde{\mathbf{l}} = [\tilde{l}_1 \quad \tilde{l}_2 \quad \ldots \quad \tilde{l}_8]^T$, 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.

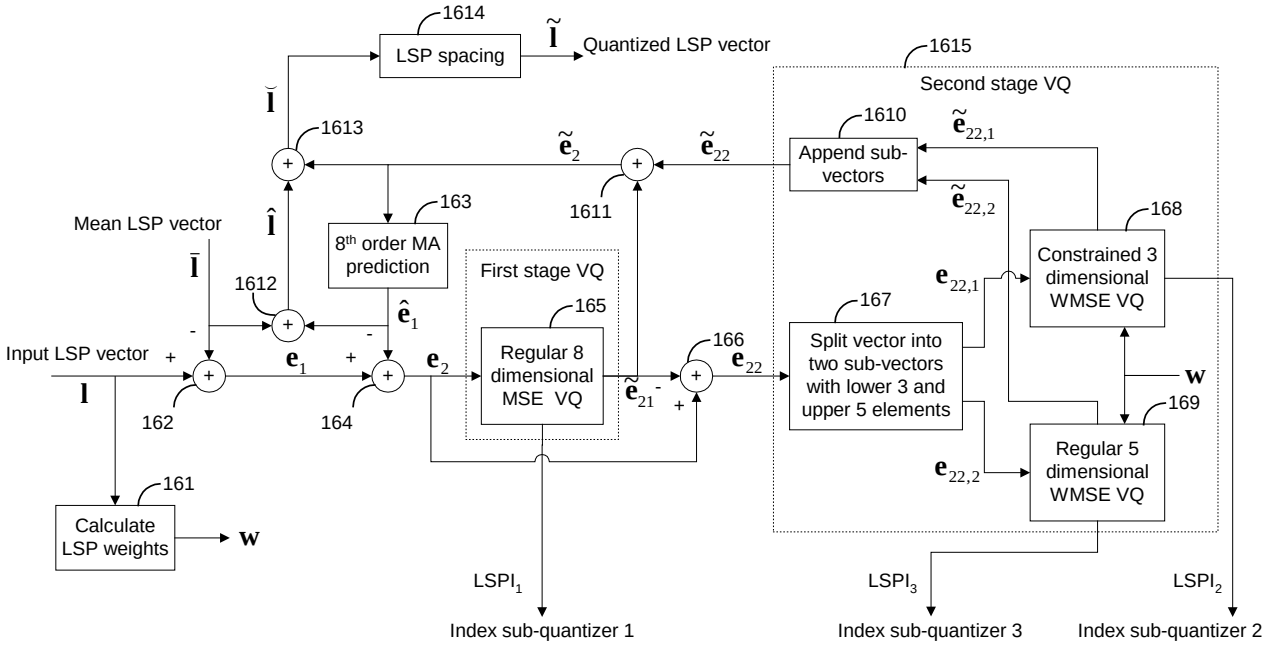

Figure 5 BV32 LSP quantizer (block 16)

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,

$$\bar{\mathbf{l}} = \begin{bmatrix} 0.0551453 & 0.1181030 & 0.2249756 & 0.3316040 & 0.4575806 & 0.5720825 & 0.7193298 & 0.8278198 \end{bmatrix}^T,$$

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 vector[2]. 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

$$\hat{e}_{1,i} = \mathbf{p}_{LSP,i}{}^T \cdot \begin{bmatrix} \tilde{e}_{2,i}(1) & \tilde{e}_{2,i}(2) & \tilde{e}_{2,i}(3) & \tilde{e}_{2,i}(4) & \tilde{e}_{2,i}(5) & \tilde{e}_{2,i}(6) & \tilde{e}_{2,i}(7) & \tilde{e}_{2,i}(8) \end{bmatrix}^T, \quad i = 1, 2, \ldots, 8,$$

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

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

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, $\mathbf{CB}_1 = \left\{ \mathbf{cb}_1{}^{(0)}, \mathbf{cb}_1{}^{(1)}, \ldots, \mathbf{cb}_1{}^{(127)} \right\}$, listed in Appendix 2. The codevector minimizing the MSE is denoted $\tilde{\mathbf{e}}_{21}$ and the corresponding index is denoted $LSPI_1$:

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

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

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

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

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

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, $\mathbf{CB}_{21} = \left\{\mathbf{cb}_{21}^{(0)}, \mathbf{cb}_{21}^{(1)}, \ldots, \mathbf{cb}_{21}^{(31)}\right\}$, of Appendix 3. The codevector that minimizes the WMSE subject to the constraint that the 3 first elements of the intermediate quantized LSP vector,

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

preserve the ordering property

$$\begin{matrix} \breve{l}_1 & \geq & 0 \\ \breve{l}_2 & \geq & \breve{l}_1 \\ \breve{l}_3 & \geq & \breve{l}_2 \end{matrix},$$

is selected as $\tilde{\mathbf{e}}_{22,1}$, and the corresponding index is denoted $LSPI_2$. In the inequality above, the symbol $\breve{l}_i$ represents the *i*-th element of the vector $\breve{\mathbf{l}}$. The constrained WMSE VQ is given by

$$LSPI_2 = \underset{k \in \left\{ j \;\middle|\; \breve{l}_1^{(j)} \geq 0, \breve{l}_2^{(j)} \geq \breve{l}_1^{(j)}, \breve{l}_3^{(j)} \geq \breve{l}_2^{(j)}, j \in \{0,1,\ldots,31\} \right\}}{\arg\min} \left\{ \left(\mathbf{e}_{22,1} - \mathbf{cb}_{21}{}^{(k)} \right)^T \mathbf{W}_1 \left(\mathbf{e}_{22,1} - \mathbf{cb}_{21}{}^{(k)} \right) \right\},$$

$$\tilde{\mathbf{e}}_{22,1} = \mathbf{cb}_{21}{}^{(LSPI_2)},$$

where

$$\mathbf{W}_1 = \begin{bmatrix} w_1 & 0 & 0 \\ 0 & w_2 & 0 \\ 0 & 0 & w_3 \end{bmatrix},$$

and $\breve{l}_i^{(j)}$ is the *i*-th element of the reconstructed LSP vector $\breve{\mathbf{l}}$ that is generated by using the *j*-th codevector in $\mathbf{CB}_{21}$. In the highly unlikely, but possible, event that no codevector satisfy the ordering property of the intermediate quantized LSP vector, the quantizer selects the codevector $\mathbf{cb}_{21}{}^{(1)}$ and returns the index $LSPI_2 = 1$.

Block 169 performs an unconstrained WMSE VQ of the 5-dimensional vector, $\mathbf{e}_{22,2}$, using the 32-entry codebook, $\mathbf{CB}_{22} = \left\{ \mathbf{cb}_{22}{}^{(0)}, \mathbf{cb}_{22}{}^{(1)}, \ldots, \underline{\mathbf{cb}}_{22}{}^{(31)} \right\}$, given in Appendix 4, according to

$$LSPI_3 = \underset{k \in \{0,1,\ldots,31\}}{\arg\min} \left\{ \left(\mathbf{e}_{22,2} - \mathbf{cb}_{22}{}^{(k)} \right)^T \mathbf{W}_2 \left(\mathbf{e}_{22,2} - \mathbf{cb}_{22}{}^{(k)} \right) \right\},$$

$$\tilde{\mathbf{e}}_{22,2} = \mathbf{cb}_{22}{}^{(LSPI_3)},$$

where

$$\mathbf{W}_2 = \begin{bmatrix} w_4 & & & 0 \\ & w_5 & & \\ & & \ddots & \\ 0 & & & w_8 \end{bmatrix}.$$

The quantization is complete and the remaining operations of block 16 construct the quantized LSP vector from the codevectors, LSP mean, and MA prediction. Block 1610 concatenates the two quantized split vectors to obtain

$$\tilde{\mathbf{e}}_{22} = \begin{bmatrix} \tilde{\mathbf{e}}_{22,1} \\ \tilde{\mathbf{e}}_{22,2} \end{bmatrix},$$

the quantized version of the quantization error vector of the first stage VQ. Adder 1611 calculates the quantized prediction error vector by adding the stage 1 and stage 2 quantized vectors,

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

Adder 1612 adds the mean LSP vector and the predicted mean-removed LSP vector to obtain the predicted LSP vector,

$$\hat{\mathbf{l}} = \bar{\mathbf{l}} + \hat{\mathbf{e}}_1 .$$

Adder 1613 adds the predicted LSP vector and the quantized prediction error vector to get the intermediate reconstructed LSP vector,

$$\breve{\mathbf{l}} = \hat{\mathbf{l}} + \tilde{\mathbf{e}}_2 .$$

Block 1614 calculates the final quantized LSP coefficients by enforcing a minimum spacing of 100 Hz between adjacent LSP coefficients, as well as an absolute minimum of 12 Hz for the first LSP coefficient and an absolute maximum of 7982 Hz for the eighth LSP coefficient. The spacing constraints are given by

$$\begin{array}{lcll} \tilde{l}_1 & \geq & 0.0015 & \\ \tilde{l}_{i+1} - \tilde{l}_i & \geq & 0.0125 & i = 1, 2, \ldots, 7. \\ \tilde{l}_8 & \leq & 0.99775 & \end{array}$$

The spacing is carried out as follows:

(i) The elements of the intermediate reconstructed LSP vector are sorted such that

$$\breve{l}_1 \leq \breve{l}_2 \leq \ldots \leq \breve{l}_8.$$

(ii) Set $l_{\max} = 0.91025$.

(iii) If $\breve{l}_1 < 0.0015$, set $\tilde{l}_1 = 0.0015$.
else if $\breve{l}_1 > l_{\max}$, set $\tilde{l}_1 = l_{\max}$.
else set $\tilde{l}_1 = \breve{l}_1$.

(iv) for i=2, 3, … , 8 do the following:

1. Set $l_{\min} = \tilde{l}_{i-1} + 0.0125$.
2. Set $l_{\max} \leftarrow l_{\max} + 0.0125$.
3. If $\breve{l}_i < l_{\min}$, set $\tilde{l}_i = l_{\min}$.
   else if $\breve{l}_i > l_{\max}$, set $\tilde{l}_i = l_{\max}$.
   else set $\tilde{l}_i = \breve{l}_i$.

3.6 Conversion to Short-Term Predictor Coefficients

Refer back to Figure 4. In block 17, the quantized set of LSP coefficients $\{\tilde{l}_i\}$, which is determined once a frame, is converted to the corresponding set of linear prediction coefficients $\{\tilde{a}_i\}$, the quantized linear prediction coefficients for the current frame.

With the notation

$$\begin{aligned} x_{p,i} &= \cos(\pi\, \tilde{l}_{2i-1}), \quad i = 1, 2, 3, 4 \\ x_{m,i} &= \cos(\pi\, \tilde{l}_{2i}), \quad i = 1, 2, 3, 4 \end{aligned}$$

the 4 unique coefficients of each of the two polynomials $A_p^{\Delta}(z) = A_p(z)/(1+z^{-1})$ and $A_m^{\Delta}(z) = A_m(z)/(1-z^{-1})$ can be determined using the following recursion:

For $i = 1, 2, 3, 4,$ do the following :

$$a^{\Delta}_{p|m,i} = 2\left(a^{\Delta}_{p|m,i-2} - x_{p|m,i}\, a^{\Delta}_{p|m,i-1}\right)$$

$$a^{\Delta}_{p|m,j} = a^{\Delta}_{p|m,j} + a^{\Delta}_{p|m,j-2} - 2\, x_{p|m,i}\, a^{\Delta}_{p|m,j-1}, \quad j = i-1, i-2, \ldots, 1$$

with initial conditions $a^{\Delta}_{p|m,0} = 1$ and $a^{\Delta}_{p|m,-1} = 0$. In the recursion above, $\{a^{\Delta}_{p,i}\}$ and $\{a^{\Delta}_{m,i}\}$ are the sets of four unique coefficients of the polynomials $A_p^{\Delta}(z)$ and $A_m^{\Delta}(z)$, respectively. Similarly, let the two sets of coefficients $\{a_{p,i}\}$ and $\{a_{m,i}\}$, each of 4 unique coefficients except for a sign on $\{a_{m,i}\}$, represent the unique coefficients of the polynomials $A_p(z)$ and $A_m(z)$, respectively. Then, $\{a_{p,i}\}$ and $\{a_{m,i}\}$ can be obtained from $\{a^{\Delta}_{p,i}\}$ and $\{a^{\Delta}_{m,i}\}$ as

$$\begin{aligned} a_{p,i} &= a^{\Delta}_{p,i} + a^{\Delta}_{p,i-1}, \quad i = 1, 2, 3, 4 \\ a_{m,i} &= a^{\Delta}_{m,i} - a^{\Delta}_{m,i-1}, \quad i = 1, 2, 3, 4 \end{aligned}$$

From $A_p(z)$ and $A_m(z)$, the polynomial of the prediction error filter is obtained as

$$\tilde{A}(z) = \frac{A_p(z) + A_m(z)}{2}.$$

In terms of the unique coefficients of $A_p(z)$ and $A_m(z)$, the coefficients $\{\tilde{a}_i\}$ of $\tilde{A}(z)$ can be expressed as

$$\tilde{a}_i = \begin{cases} 1.0, & i = 0 \\ 0.5\,(a_{p,i} + a_{m,i}), & i = 1, 2, 3, 4 \\ 0.5\,(a_{p,9-i} - a_{m,9-i}), & i = 5, 6, 7, 8 \end{cases}$$

where the tilde signifies that the coefficients correspond to the quantized LSP coefficients. Note that

$$\tilde{A}(z) = 1 - P_s(z) = 1 + \sum_{i=1}^{8} \tilde{a}_i \; z^{-i},$$

where

$$P_s(z) = -\sum_{i=1}^{8} \tilde{a}_i \; z^{-i}$$

is the transfer function of the short-term predictor block 40 in Figure 2.

Block 18 performs further bandwidth expansion on the set of predictor coefficients $\{\tilde{a}_i\}$ using a bandwidth expansion factor of $\gamma_1 = 0.75$. The resulting bandwidth-expanded set of filter coefficients is given by

$$a'_i = \gamma_1^{\,i} \tilde{a}_i \quad , \text{ for } i = 1, 2, \ldots, 8.$$

This bandwidth-expanded set of filter coefficients $\{a'_i\}$ are used to update the coefficients of the short-term noise feedback filter block 50 in Figure 2 and the coefficients of the weighted short-term synthesis filter block 21 in Figure 6 (to be discussed later). This completes the description of short-term predictive analysis and quantization block 10 in Figure 2 and Figure 4.

3.7 Short-Term Linear Prediction of Input Signal

Now refer to Figure 2. The short-term predictor block 40 predicts the input signal sample *s*(*n*) based on a linear combination of the preceding 8 samples. The adder 45 subtracts the resulting predicted value from *s*(*n*) to obtain the short-term prediction residual signal, or the difference signal, *d*(*n*). The combined operation of blocks 40 and 45 is summarized in the following difference equation.

$$d(n) = s(n) + \sum_{i=1}^{8} \tilde{a}_i s(n-i)$$

3.8 Long-Term Linear Predictive Analysis (Pitch Extraction)

In Figure 2, the long-term predictive analysis and quantization block 20 uses the short-term prediction residual signal *d*(*n*) of the current frame and its quantized version *dq*(*n*) in the previous frames to determine the quantized values of the pitch period and the pitch predictor taps. This block 20 is further expanded in Figure 6 below.

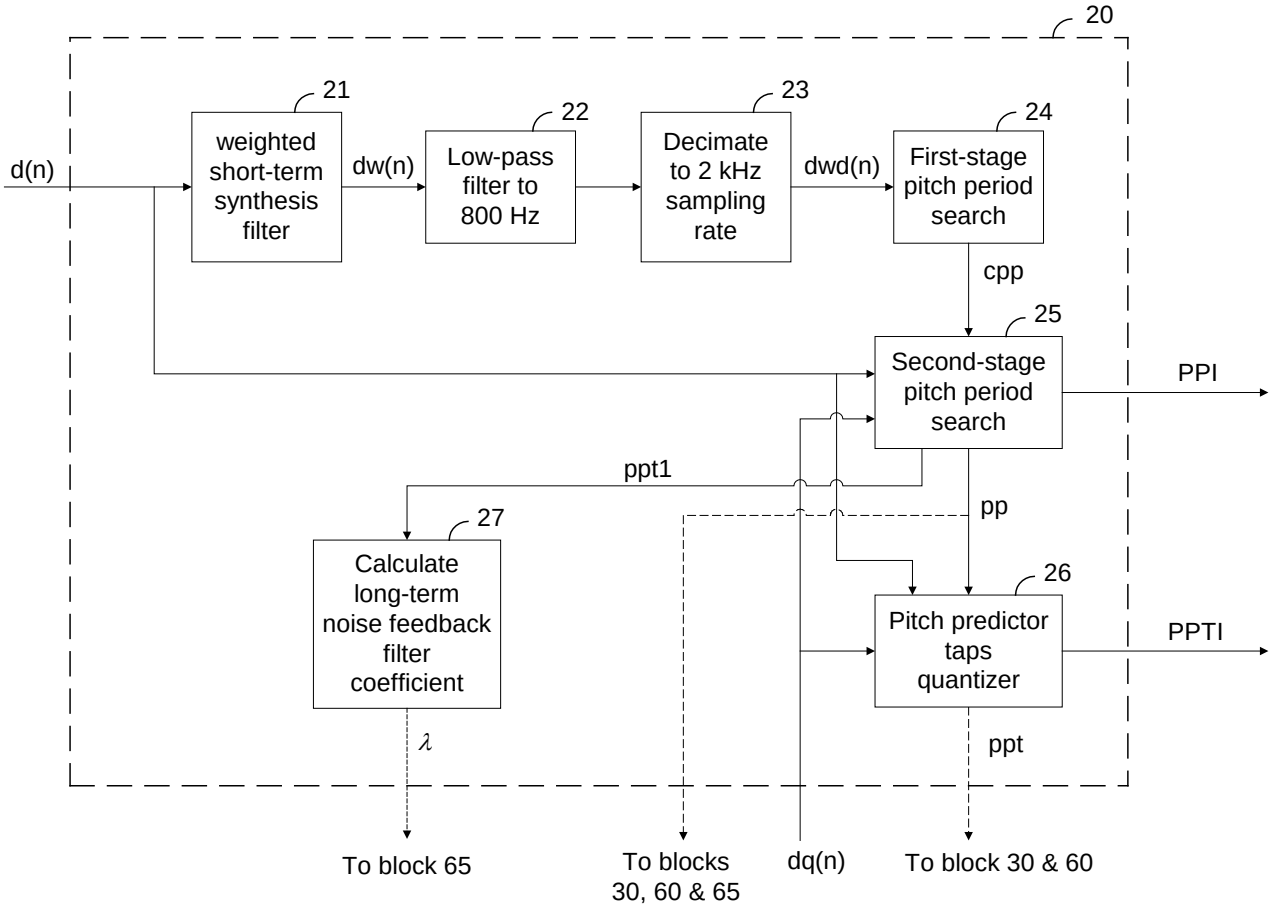

Figure 6 BV32 long-term predictive analysis and quantization (block 20)

Now refer to Figure 6. The short-term prediction residual signal *d*(*n*) passes through the weighted short-term synthesis filter block 21, whose output is calculated as

$$dw(n) = d(n) - \sum_{i=1}^{8} a_i' dw(n-i)$$

The signal *dw*(*n*) is passed through a fixed low-pass filter block 22, which has a –3 dB cut off frequency at about 800 Hz. A 4th-order elliptic filter is used for this purpose. The transfer function of this low-pass filter is

$$H_{lpf}(z) = \frac{0.0322952 - 0.1028824\,z^{-1} + 0.1446838\,z^{-2} - 0.1028824\,z^{-3} + 0.0322952\,z^{-4}}{1 - 3.5602306\,z^{-1} + 4.8558478\,z^{-2} - 2.9988298\,z^{-3} + 0.7069277\,z^{-4}}$$

Block 23 down-samples the low-pass filtered signal to a sampling rate of 2 kHz. This represents an 8:1 decimation for the BV32 codec.

The first-stage pitch search block 24 then uses the decimated 2 kHz sampled signal *dwd*(*n*) to find a "coarse pitch period", denoted as *cpp* in Figure 6. The time lag represented by *cpp* is in terms of number of samples in the 2 kHz down-sampled signal *dwd*(*n*). A pitch analysis window of 15 ms is used. The end of the pitch analysis window is lined up with the end of the current frame. At a sampling rate of 2 kHz, 15 ms correspond to 30 samples. Without loss of generality, let the index range of $n = 1$ to $n = 30$ correspond to the pitch analysis window for *dwd*(*n*). Block 24 first calculates the following values

$$c(k) = \sum_{n=1}^{30} dwd(n)dwd(n-k),$$

$$E(k) = \sum_{n=1}^{30} [dwd(n-k)]^2,$$

$$c2(k) = \begin{cases} c^2(k), & \text{if } c(k) \geq 0 \\ -c^2(k), & \text{if } c(k) < 0 \end{cases}$$

for all integers from $k = MINPPD - 1$ to $k = MAXPPD + 1$, where *MINPPD* and *MAXPPD* are the minimum and maximum pitch period in the decimated domain, respectively. For BV32, *MINPPD* = 1 sample and *MAXPPD* = 33 samples. Block 24 then searches through the range of $k = MINPPD$, $MINPPD + 1$, $MINPPD + 2$, …, *MAXPPD* to find all local peaks[3] of the array $\{c2(k)/E(k)\}$ for which $c(k) > 0$. Let N_p denote the number of such positive local peaks. Let $k_p(j)$, j =1, 2, …, N_p be the indices where $c2(k_p(j))/E(k_p(j))$ is a local peak and $c(k_p(j)) > 0$, and let $k_p(1) < k_p(2) < ... < k_p(N_p)$. For convenience, the term $c2(k)/E(k)$ will be referred to as the "normalized correlation square".

If $N_p = 0$, the output coarse pitch period is set to $cpp = MINPPD$, and the processing of block 24 is terminated. If $N_p = 1$, block 24 output is set to $cpp = k_p(1)$, and the processing of block 24 is terminated.

[3] A value is characterized as a local peak if both of the adjacent values are smaller.

If there are two or more local peaks ($N_p \geq 2$), then block 24 uses *Algorithms 3.8.1, 3.8.2, 3.8.3,* and *3.8.4* (to be described below), in that order, to determine the output coarse pitch period *cpp*. Variables calculated in the earlier algorithms will be carried over and used in the later algorithms.

Block 24 first uses *Algorithm 3.8.1* below to identify the largest quadratically interpolated peak around local peaks of the normalized correlation square $c2(k_p)/E(k_p)$. Quadratic interpolation is performed for $c(k_p)$, while linear interpolation is performed for $E(k_p)$. Such interpolation is performed with the time resolution for the sampling rate of the input speech (16 kHz for BV32). In the algorithm below, *D* denotes the decimation factor used when decimating *dw*(*n*) to *dwd*(*n*). Thus, *D* = 8 for BV32.

Algorithm 3.8.1 Find largest quadratically interpolated peak around $c2(k_p)/E(k_p)$:

(i) Set *c2max* = -1, *Emax* = 1, and *jmax* = 0.

(ii) For *j* =1, 2, …, N_p, do the following 12 steps:

1. Set $a = 0.5\left[c(k_p(j)+1) + c(k_p(j)-1)\right] - c(k_p(j))$
2. Set $b = 0.5\left[c(k_p(j)+1) - c(k_p(j)-1)\right]$
3. Set $ji = 0$
4. Set $ei = E(k_p(j))$
5. Set $c2m = c2(k_p(j))$
6. Set $Em = E(k_p(j))$
7. If $c2(k_p(j)+1)E(k_p(j)-1) > c2(k_p(j)-1)E(k_p(j)+1)$, do the remaining part of step 7:

   $\Delta = [E(k_p(j)+1) - ei]/D$

   For *k* = 1, 2, … , D/2, do the following indented part of step 7:

   $ci = a\ (k/D)^2 + b\ (k/D) + c(k_p(j))$

   $ei \leftarrow ei + \Delta$

   If $(ci)^2 Em > (c2m)\ ei$, do the next three indented lines:

   $ji = k$

   $c2m = (ci)^2$

   $Em = ei$
8. If $c2(k_p(j)+1)E(k_p(j)-1) \leq c2(k_p(j)-1)E(k_p(j)+1)$, do the remaining part of step 8:

   $\Delta = [E(k_p(j)-1) - ei]/D$

   For *k* = -1, -2, … , -D/2, do the following indented part of step 8:

   $ci = a\ (k/D)^2 + b\ (k/D) + c(k_p(j))$

   $ei \leftarrow ei + \Delta$

   If $(ci)^2 Em > (c2m)\ ei$, do the next three indented lines:

   $ji = k$

$c2m = (ci)^2$

$Em = ei$

9. Set $lag(j) = k_p(j) + ji/D$
10. Set $c2i(j) = c2m$
11. Set $Ei(j) = Em$
12. If $c2m \times Emax > c2max \times Em$, do the following three indented lines:
    jmax = j
    c2max = c2m
    Emax = Em

(iii) Set the first candidate for coarse pitch period as $cpp = k_p(jmax)$.

The symbol $\leftarrow$ indicates that the parameter on the left-hand side is being updated with the value on the right-hand side[4].

To avoid picking a coarse pitch period that is around an integer multiple of the true coarse pitch period, a search through the time lags corresponding to the local peaks of $c2(k_p)/E(k_p)$ is performed to see if any of such time lags is close enough to the output coarse pitch period of block 24 in the last frame, denoted as *cpplast*[5]. If a time lag is within 25% of *cpplast*, it is considered close enough. For all such time lags within 25% of *cpplast*, the corresponding quadratically interpolated peak values of the normalized correlation square $c2(k_p)/E(k_p)$ are compared, and the interpolated time lag corresponding to the maximum normalized correlation square is selected for further consideration. The following algorithm performs the task described above. The interpolated arrays *c2i*(*j*) and *Ei*(*j*) calculated in *Algorithm 3.8.1* above are used in this algorithm.

Algorithm 3.8.2 Find the time lag maximizing interpolated $c2(k_p)/E(k_p)$ *among all time lags close to the output coarse pitch period of the last frame:*

(i) Set index *im* = -1
(ii) Set *c2m* = -1
(iii) Set *Em* = 1
(iv) For *j* =1, 2, …, N_p, do the following:

If $|k_p(j) - cpplast| \le 0.25 \times cpplast$, do the following:

If $c2i(j) \times Em > c2m \times Ei(j)$, do the following three lines:

$im = j$
$c2m = c2i(j)$
$Em = Ei(j)$

[4] An equal sign is not applicable due to a potential mathematical conflict.
[5] For the first frame *cpplast* is initialized to 12.

Note that If there is no time lag $k_p(j)$ within 25% of *cpplast*, then the value of the index *im* will remain at –1 after *Algorithm 3.8.2* is performed. If there are one or more time lags within 25% of *cpplast*, the index *im* corresponds to the largest normalized correlation square among such time lags.

Next, block 24 determines whether an alternative time lag in the first half of the pitch range should be chosen as the output coarse pitch period. Basically, block 24 searches through all interpolated time lags *lag*(*j*) that are less than 16, and checks whether any of them has a large enough local peak of normalized correlation square near every integer multiple of it (including itself) up to 32. If there are one or more such time lags satisfying this condition, the smallest of such qualified time lags is chosen as the output coarse pitch period of block 24.

Again, variables calculated in *Algorithms 3.8.1* and *3.8.2* above carry their final values over to *Algorithm 3.8.3* below. In the following, the parameter *MPDTH* is 0.06, and the threshold array *MPTH*(*k*) is given as *MPTH*(2) = 0.7, *MPTH*(3) = 0.55, *MPTH*(4) = 0.48, *MPTH*(5) = 0.37, and *MPTH*(*k*) = 0.30, for $k > 5$.

Algorithm 3.8.3 Check whether an alternative time lag in the first half of the range of the coarse pitch period should be chosen as the output coarse pitch period:

For $j = 1, 2, 3, \ldots, N_p$, in that order, do the following while *lag*(*j*) < 16:

(i) If $j \neq im$, set *threshold* = 0.73; otherwise, set *threshold* = 0.4.

(ii) If $c2i(j) \times Emax \leq threshold \times c2max \times Ei(j)$, disqualify this *j*, skip step (iii) for this *j*, increment *j* by 1 and go back to step (i).

(iii) If $c2i(j) \times Emax > threshold \times c2max \times Ei(j)$, do the following:

a) For $k = 2, 3, 4, \ldots$, do the following while $k \times lag(j) < 32$:

1. $s = k \times lag(j)$

2. $a = (1 - MPDTH)\, s$

3. $b = (1 + MPDTH)\, s$

4. Go through $m = j+1,\ j+2,\ j+3, \ldots,\ N_p$, in that order, and see if any of the time lags *lag*(*m*) is between *a* and *b*. If none of them is between *a* and *b*, disqualify this *j*, stop step (iii), increment *j* by 1 and go back to step (i). If there is at least one such *m* that satisfies $a < lag(m) \leq b$ and $c2i(m) \times Emax > MPTH(k) \times c2max \times Ei(m)$, then it is considered that a large enough peak of the normalized correlation square is found in the neighborhood of the *k*-th integer multiple of *lag*(*j*); in this case, stop step (iii) a) 4., increment *k* by 1, and go back to step (iii) a) 1.

b) If step (iii) a) is completed without stopping prematurely, that is, if there is a large enough interpolated peak of the normalized correlation square within ±100×*MPDTH*% of every integer multiple of *lag*(*j*) that is less than 32, then stop this algorithm and stop the operation of block 24, and set $cpp = k_p(j)$ as the final output coarse pitch period of block 24.

If *Algorithm 3.8.3* above is completed without finding a qualified output coarse pitch period *cpp*, then block 24 examines the largest local peak of the normalized correlation square around the coarse pitch period of the last frame, found in *Algorithm 3.8.2* above, and makes a final decision on the output coarse pitch period *cpp* using the following algorithm. *Algorithm 3.8.4* performs this final decision. Again, variables calculated in *Algorithms 3.8.1* and *3.8.2* above carry their final values over to *Algorithm 3.8.4* below. In the following, the parameters are *SMDTH* = 0.095 and *LPTH1* = 0.78.

Algorithm 3.8.4: Final decision of the output coarse pitch period

(i) If *im* = -1, that is, if there is no large enough local peak of the normalized correlation square around the coarse pitch period of the last frame, then use the *cpp* calculated at the end of *Algorithm 3.8.1* as the final output coarse pitch period of block 24, and exit this algorithm.

(ii) If *im* = *jmax*, that is, if the largest local peak of the normalized correlation square around the coarse pitch period of the last frame is also the global maximum of all interpolated peaks of the normalized correlation square within this frame, then use the *cpp* calculated at the end of *Algorithm 3.8.1* as the final output coarse pitch period of block 24, and exit this algorithm.

(iii) If *im* < *jmax*, do the following indented part:

If *c2m* × *Emax* > 0.43 × *c2max* × *Em*, do the following indented part of step (iii):

a) If *lag*(*im*) > *MAXPPD*/2, set block 24 output $cpp = k_p(im)$ and exit this algorithm.

b) Otherwise, for *k* = 2, 3, 4, 5, do the following indented part:

1. *s* = *lag*(*jmax*) / *k*
2. *a* = (1 – *SMDTH*) *s*
3. *b* = (1 + *SMDTH*) *s*
4. If *lag*(*im*) > *a* and *lag*(*im*) < *b*, set block 24 output $cpp = k_p(im)$ and exit this algorithm.

(iv) If *im* > *jmax*, do the following indented part:

If $c2m \times Emax > LPTH1 \times c2max \times Em$, set block 24 output $cpp = k_p(im)$ and exit this algorithm.

(v) If algorithm execution proceeds to here, none of the steps above have selected a final output coarse pitch period. In this case, just accept the *cpp* calculated at the end of *Algorithm 3.8.1* as the final output coarse pitch period of block 24.

Block 25 takes *cpp* as its input and performs a second-stage pitch period search in the undecimated signal domain to get a refined pitch period *pp*. Block 25 first converts the coarse pitch period *cpp* to the undecimated signal domain by multiplying it by the decimation factor *D*, where $D = 8$ for BV32. Then, it determines a search range for the refined pitch period around the value $cpp \times D$. The lower bound of the search range is $lb = \max(MINPP, cpp \times D - D + 1)$, where $MINPP = 10$ samples is the minimum pitch period. The upper bound of the search range is $ub = \min(MAXPP, cpp \times D + D - 1)$, where *MAXPP* is the maximum pitch period, which is 264 samples for BV32.

Block 25 maintains a signal buffer with a total of $MAXPP + 1 + FRSZ$ samples, where *FRSZ* is the frame size, which is 80 samples for BV32. The last *FRSZ* samples of this buffer are populated with the open-loop short-term prediction residual signal $d(n)$ in the current frame. The first $MAXPP + 1$ samples are populated with the $MAXPP + 1$ samples of quantized version of $d(n)$, denoted as $dq(n)$, immediately preceding the current frame. For convenience of writing equations later, the symbol $dq(n)$ will be used to denote the entire buffer of $MAXPP + 1 + FRSZ$ samples, even though the last *FRSZ* samples are really $d(n)$ samples. Again, let the index range from $n = 1$ to $n = FRSZ$ denotes the samples in the current frame.

After the lower bound *lb* and upper bound *ub* of the pitch period search range are determined, block 25 calculates the following correlation and energy terms in the undecimated $dq(n)$ signal domain for time lags k within the search range $[lb, ub]$.

$$\tilde{c}(k) = \sum_{n=1}^{FRSZ} dq(n)dq(n-k)$$

$$\tilde{E}(k) = \sum_{n=1}^{FRSZ} dq(n-k)^2$$

The time lag $k \in [lb, ub]$ that maximizes the ratio $\tilde{c}^2(k)/\tilde{E}(k)$ is chosen as the final refined pitch period. That is,

$$pp = \arg\max_{k \in [lb,ub]} \left[\frac{\tilde{c}^2(k)}{\tilde{E}(k)} \right] .$$

Once the refined pitch period *pp* is determined, it is encoded into the corresponding output pitch period index *PPI*, calculated as

$$PPI = pp - 10 \ .$$

Possible values of *PPI* are all integers from 0 to 254 for the BV32 codec. Therefore, the refined pitch period *pp* is encoded into 8 bits, without any distortion. The value of *PPI* = 255 is reserved for signaling purposes and therefore is not used by the BV32 codec.

Block 25 also calculates *ppt*1, the optimal tap weight for a single-tap pitch predictor, as follows

$$ppt1 = \frac{\tilde{c}(pp)}{\tilde{E}(pp)} \ .$$

In the degenerate case where $\tilde{E}(pp) = 0$, *ppt*1 is set to zero. Block 27 calculates the long-term noise feedback filter coefficient λ as follows.

$$\lambda = \begin{cases} 0.5, & ppt1 \geq 1 \\ 0.5 \times ppt1, & 0 < ppt1 < 1 \\ 0, & ppt1 \leq 0 \end{cases}$$

3.9 Long-Term Predictor Parameter Quantization

Pitch predictor taps quantizer block 26 quantizes the three pitch predictor taps to 5 bits using vector quantization. The pitch predictor has a transfer function of

$$P_l(z) = \sum_{i=1}^{3} b_i \ z^{-pp+2-i} \ ,$$

where *pp* is the pitch period calculated in Section 3.8.

Rather than minimizing the mean-square error of the three taps b_1, b_2, and b_3 as in conventional VQ codebook search, block 26 finds from the VQ codebook the set of candidate pitch predictor taps that minimizes the pitch prediction residual energy in the current frame. Using the same *dq*(*n*) buffer and time index convention as in block 25, and denoting the set of three taps corresponding to the *j*-th codevector, $\mathbf{b}_j = [b_{j1} \quad b_{j2} \quad b_{j3}]^T$, as $\{b_{j1}, b_{j2}, b_{j3}\}$, we can express such pitch prediction residual energy as

$$E_j = \sum_{n=1}^{FRSZ} \left[dq(n) - \sum_{i=1}^{3} b_{ji} dq(n - pp + 2 - i) \right]^2 .$$

The codevector is selected from a 3-dimensional codebook of 32 codevectors, $\{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{31}\}$, listed in Appendix 5. The codevector that minimizes the pitch prediction residual energy is selected. The index of the selected codevector is given by

$$PPTI = j^* = \underset{j \in \{0,1,\ldots,31\}}{\arg\min} \{E_j\}$$

and the corresponding set of three quantized pitch predictor taps, denoted as $ppt = \{b_1, b_2, b_3\}$ in Figure 6, is given by

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{b}_{j^*}.$$

This completes the description of block 20, long-term predictive analysis and quantization.

3.10 Excitation Gain Quantization

In BV32 coding, there are two 2.5 ms sub-frames in each 5 ms frame, with one residual gain for each sub-frame. The sub-frame size is therefore *SFRSZ* = *FRSZ*/2 = 40 samples. The unquantized residual gains are based on the pitch prediction residuals of the respective sub-frames and are quantized open-loop in the base-2 logarithmic domain. The quantization of the residual gain is part of the prediction residual quantizer block 30 in Figure 2. Block 30 is further expanded in Figure 7. All the operations in Figure 7 are performed sub-frame by sub-frame.

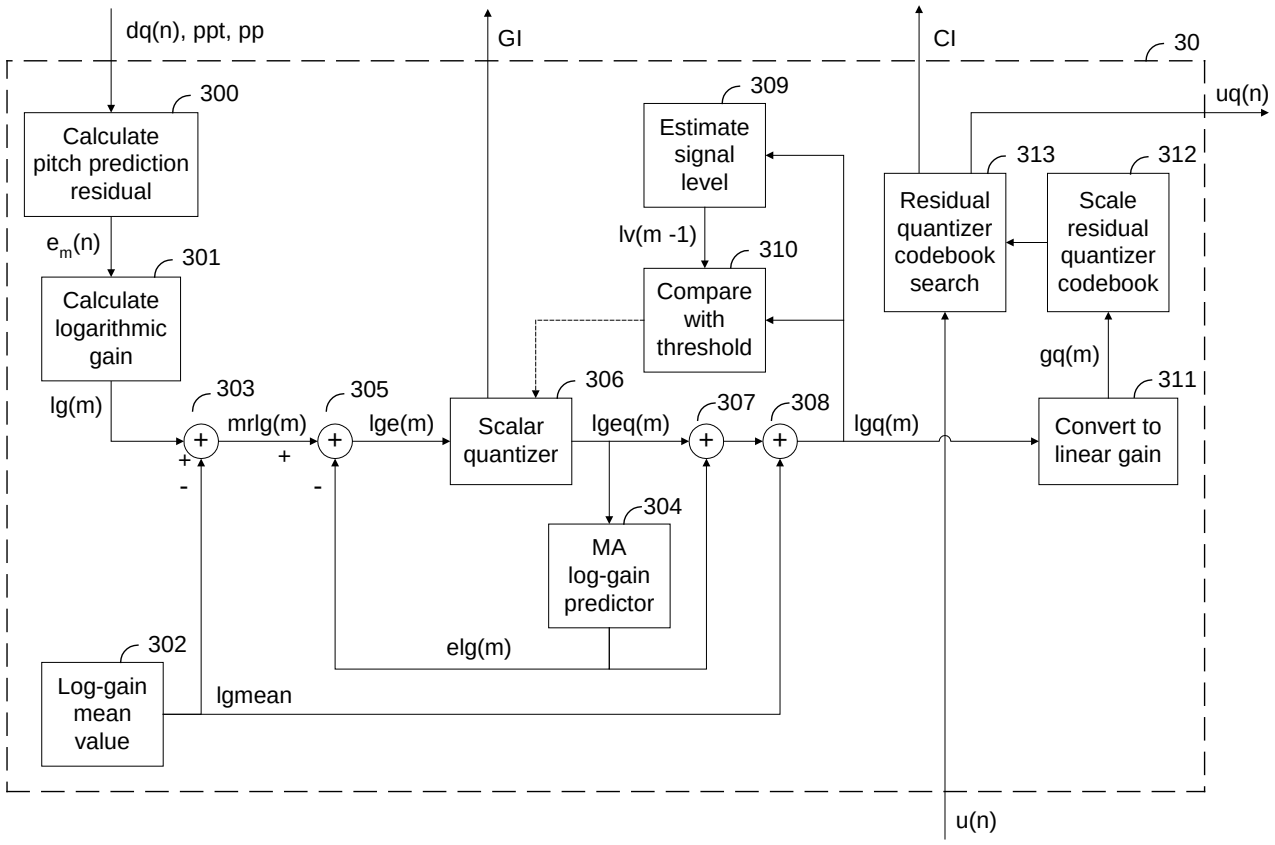

Figure 7 Prediction residual quantizer (block 30)

Block 300 in Figure 7 calculates the pitch prediction residual signal. For the first sub-frame the pitch prediction residual signal is given by

$$e_1(n) = dq(n) - \sum_{i=1}^{3} b_i dq(n - pp + 2 - i), \quad n = 1,2,...,SFRSZ ,$$

where the same $dq(n)$ buffer and time index convention of block 25 is used. Hence, the first sub-frame of $dq(n)$ for n = 1, 2, …, *SFRSZ* is the unquantized open-loop short-term prediction residual signal $d(n)$. For the second sub-frame the pitch prediction residual signal is given by

$$e_2(n) = dq(SFRSZ + n) - \sum_{i=1}^{3} b_i dq(SFRSZ + n - pp + 2 - i), \quad n = 1, 2, ..., SFRSZ .$$

Again, the second sub-frame of $dq(n)$, n = *SFRSZ*+1, *SFRSZ*+2, …, *FRSZ*, is the unquantized open-loop short-term prediction residual signal $d(n)$. However, at the time the pitch prediction residual of the second sub-frame is calculated, the excitation of the first sub-frame is fully quantized and is located at $dq(n)$, n = 1, 2, …, *SFRSZ*, which is then no longer the unquantized open-loop short-term prediction residual signal.

Block 301 calculates the residual gain in the base-2 logarithmic domain. First, the average power of the pitch prediction residual signal in the *m*-th sub-frame is calculated as

$$P_e(m) = \frac{1}{SFRSZ} \sum_{n=1}^{SFRSZ} e_m{}^2(n),$$

where m = 1 and 2 for the first and second sub-frames of the current frame, respectively. Then, to avoid taking logarithm of zero or a very small number, the logarithmic gain (log-gain) of the *m*-th sub-frame is calculated as

$$lg(m) = \begin{cases} \log_2 P_e(m), & \text{if } P_e(m) > 1/4 \\ -2, & \text{if } P_e(m) \le 1/4 \end{cases}.$$

The long-term mean value of the log-gain is calculated off-line and stored in block 302. This log-gain mean value for BV32 is *lgmean* = 11.82031. The adder 303 calculates the mean-removed version of the log-gain as *mrlg*(*m*) = *lg*(*m*) - *lgmean*. The MA log-gain predictor block 304 is a 16th-order FIR filter with its memory initialized to zero at the very first frame. The coefficients of this log-gain predictor *lgp*(*k*), *k* = 1, 2, 3, …, 16, are fixed, as given below:

lgp(1) = 0.5913086
lgp(2) = 0.5251160
lgp(3) = 0.5724792
lgp(4) = 0.5977783
lgp(5) = 0.4800720
lgp(6) = 0.4939270
lgp(7) = 0.4729614
lgp(8) = 0.4158936
lgp(9) = 0.3805847
lgp(10) = 0.3395081
lgp(11) = 0.2780151
lgp(12) = 0.2455139
lgp(13) = 0.1916199
lgp(14) = 0.1470032
lgp(15) = 0.1138611
lgp(16) = 0.0664673

Block 304 calculates its output, the estimated log-gain, as

$$elg(m) = \sum_{k=1}^{GPO} lgp(k)\, lgeq(m-k),$$

where *GPO* = 16 is the gain predictor order for BV32, and *lgeq*(*m* - *k*) is the quantized version of the log-gain prediction error at sub-frame *m* – *k*. Here it is assumed that the sub-frame indices of

different frames form a continuous sequence of integers. For example, if the sub-frame indices in the current frame is 1 and 2, then the sub-frame indices of the last frame is –1 and 0, and the sub-frame indices of the frame before that is –3 and –2.

The adder 305 calculates the log-gain prediction error as

$$lge(m) = mrlg(m) - elg(m).$$

The scalar quantizer block 306 performs 5-bit scalar quantization of the resulting log-gain prediction error $lge(m)$. The codebook entries of this gain quantizer, along with the corresponding codebook indices, are listed in Appendix 6. The operation of this quantizer is controlled by block 310, whose purpose is to achieve a good trade-off between clear-channel performance and noisy-channel performance of the excitation gain quantizer. The operation of block 310 will be described later.

For each temporarily quantized $lgeq(m)$, the adders 307 and 308 together calculate the corresponding temporarily quantized log-gain as

$$lgq(m) = lgeq(m) + elg(m) + lgmean$$

Block 309 estimates the signal level based on the final quantized log-gain, to be determined later subject to the constraint imposed by block 310. Let $lv(m)$ denote the output estimated signal level of block 309 at sub-frame m. Since the final value of $lgq(m)$ has not been determined yet at this point, block 310 can only use the estimated signal level at the last sub-frame, namely, $lv(m-1)$. One way to think of this situation is that block 309 has a one-sample delay unit for its input $lgq(m)$.

At sub-frame m, block 310 controls the quantization operation of block 306 based on $lv(m-1)$, $lgq(m-1)$, and $lgq(m-2)$[6]. It uses an $NG \times NGC$ gain change threshold matrix $T(i, j)$, $i = 1, 2, \ldots, NG$, $j = 1, 2, \ldots, NGC$ to limit how high $lgq(m)$ can go. For BV32, the parameter values are $NG = 18$ and $NGC = 11$. The threshold matrix $T(i, j)$ is given in Appendix 7.

Block 310 and block 306 work together to perform the quantization of $lge(m)$ in the following way. First, the row index into the threshold matrix $T(i, j)$ is calculated as

$$i = \left\lceil \frac{lgq(m-1) - lv(m-1) - GLB}{2} \right\rceil,$$

where $GLB = -24$, and the symbol $\lceil . \rceil$ means "take the next larger integer" or "rounding to the nearest integer toward infinity". If $i > NG$, i is clipped to NG. If $i < 1$, i is clipped to 1.

Second, the column index into the threshold matrix $T(i, j)$ is calculated as

$$j = \left\lceil \frac{lgq(m-1) - lgq(m-2) - GCLB}{2} \right\rceil,$$

[6] The initial value of $lgq(m-1)$ and $lgq(m-2)$ is –2, i.e. $lgq(0) = -2$ *and* $lgq(-1) = -2$.

where *GCLB* = –8. If $j > NGC$, j is clipped to *NGC*. If $j < 1$, j is clipped to 1.

Third, with the row and column indices i and j calculated above, a gain quantization limit is calculated as

$$GL = lgq(m-1) + T(i, j) - elg(m) - lgmean$$

Fourth, block 306 performs normal scalar quantization of *lge*(*m*) into its nearest neighbor in the quantizer codebook. If the resulting quantized value is not greater than *GL*, this quantized value is accepted as the final quantized log-gain prediction error *lgeq*(*m*), and the corresponding codebook index is the output gain index GI_m. On the other hand, if the quantized value is greater than *GL*, the next smaller gain quantizer codebook entry is compared with *GL*. If it is not greater than *GL*, it is accepted as the final output *lgeq*(*m*) of block 306, and the corresponding codebook index is accepted as GI_m. However, if it is still greater the *GL*, then block 306 keeps looking for the next smaller quantizer codebook entry (in descending order of codebook entry value), until it finds one that is not greater than *GL*. In such a search, the first one (that is, the largest one) that it finds to be no greater than *GL* is chosen as the final output *lgeq*(*m*) of block 306, and the corresponding codebook index is accepted as GI_m. In the rare occasion when all the gain quantizer codebook entries are greater than *GL*, then the smallest gain quantizer codebook entry is chosen as the final output *lgeq*(*m*) of block 306, and the corresponding codebook index (0 in this case) is chosen as the output GI_m. The final gain quantizer codebook index GI_m is passed to the bit multiplexer block 95 of Figure 2.

Once the quantized log-gain prediction error *lgeq*(*m*) is determined in this way, adders 307 and 308 add *elg*(*m*) and *lgmean* to *lgeq*(*m*) to obtain the quantized log-gain *lgq*(*m*) as

$$lgq(m) = lgeq(m) + elg(m) + lgmean$$

After this final quantized log-gain *lgq*(*m*) subject to the constraint imposed by block 310 is calculated, it is used by block 309 to update the estimated signal level *lv*(*m*). This value *lv*(*m*) is used by block 310 in the next sub-frame (the (*m* + 1)-th sub-frame).

At sub-frame *m*, after the final quantized log-gain *lgq*(*m*) is calculated, block 309 estimates the signal level using the following algorithm. The parameter values used are $\alpha = 8191/8192$, $\beta = 1023/1024$, and $\gamma = 511/512$. At codec initialization, the related variables are initialized as: *lmax*(*m* - 1) = -100, *lmin*(*m* - 1) = 100, *lmean*(*m* - 1) = 8, *lv*(*m* - 1) = 13.5, and *x*(*m* - 1) =13.5.

Algorithm for updating estimated long-term average signal level:

(i) If *lgq*(*m*) > *lmax*(*m* - 1), set *lmax*(*m*) = *lgq*(*m*);
otherwise; set *lmax*(*m*) = *lmean*(*m* - 1) + α [*lmax*(*m* - 1) - *lmean*(*m* - 1)].

(ii) If *lgq*(*m*) < *lmin*(*m* - 1), set *lmin*(*m*) = *lgq*(*m*);

otherwise; set $lmin(m) = lmean(m - 1) + \alpha\,[lmin(m - 1) - lmean(m - 1)]$.

(iii) Set $lmean(m) = \beta \times lmean(m - 1) + (1 - \beta)\,[lmax(m) + lmin(m)]/2$.

(iv) Set $lth = lmean(m) + 0.2\,[lmax(m) - lmean(m)]$.

(v) If $lgq(m) > lth$, set $x(m) = \gamma \times x(m - 1) + (1- \gamma) lgq(m)$, and set $lv(m) = \gamma \times lv(m - 1) + (1- \gamma)\, x(m)$; Otherwise, set $x(m) = x(m - 1)$ and $lv(m) = lv(m - 1)$.

Block 311 converts the quantized log-gain $lgq(m)$ to the quantized gain $gq(m)$ in the linear domain as follows.

$$gq(m) = 2^{\frac{lgq(m)}{2}}$$

Block 312 scales the residual vector quantization (also called excitation VQ) codebook by simply multiplying every element of every codevector in the excitation VQ codebook by $gq(m)$. The resulting scaled codebook is then used by block 313 to perform Excitation VQ codebook search, as described in the next section.

3.11 Excitation Vector Quantization

The excitation VQ codebook of BV32 has a sign-shape structure, with 1 bit for sign and 5 bits for shape. The vector dimension is 4. Thus, there are 32 independent shape codevectors stored in the codebook, but the negated version of each shape codevector (i.e., the mirror image with respect to the origin) is also a valid codevector for excitation VQ. The 32 shape codevectors, along with the corresponding codebook indices, are listed in Appendix 8.

Block 313 in Figure 7 performs the excitation VQ codebook search using the filter structure shown in Figure 8, which is essentially a subset of the BV32 encoder shown in Figure 2. The only difference is that the prediction residual quantizer (block 30) in Figure 2 is replaced by block 48 in Figure 8, which is labeled as “scaled VQ codebook”. This scaled VQ codebook is calculated in Section 3.10 above.

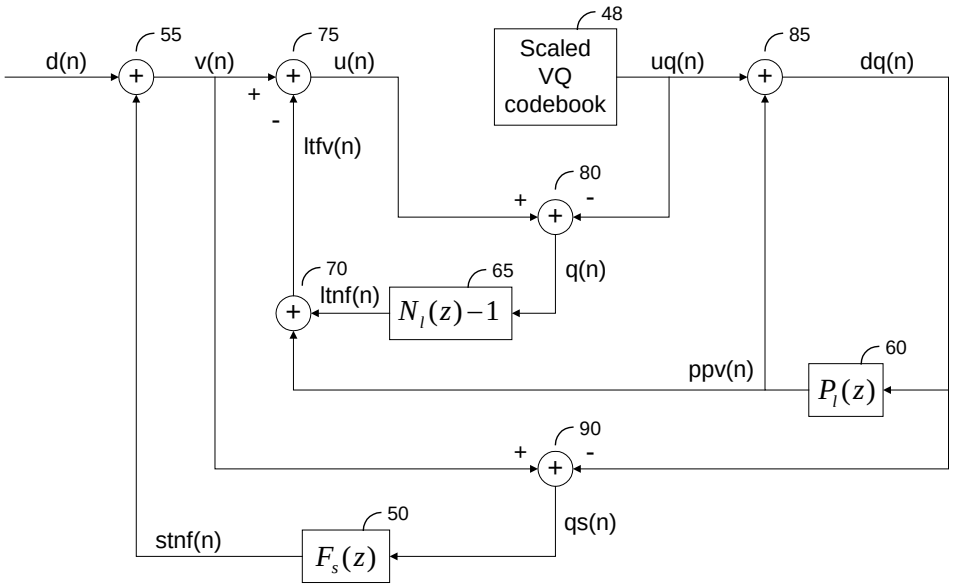

Figure 8 Filter structure used in BV32 excitation VQ codebook search

The three filters of blocks 50, 60, and 65 have transfer functions given by

$$F_s(z) = -\sum_{i=1}^{8} a_i' \, z^{-i},$$

where $a_i' = (0.75)^i \tilde{a}_i$, and $\tilde{a}_i$ is the *i*-th coefficient of the short-term prediction error filter;

$$P_l(z) = \sum_{i=1}^{3} b_i \, z^{-pp+2-i},$$

where *pp* is the pitch period, and b_i is the *i*-th long-term predictor coefficient;

$$N_l(z) - 1 = \lambda \, z^{-pp},$$

where λ is the long-term noise feedback filter coefficient calculated in Section 3.8.

Using the filter structure in Figure 8, block 313 in Figure 7 performs excitation VQ codebook search one excitation vector at a time. Each excitation vector contains four samples. The excitation gain *gq*(*m*) is updated once a sub-frame. Each sub-frame contains 10 excitation vectors. Therefore, for each sub-frame, the same scaled VQ codebook is used in 10 separate VQ codebook searches corresponding to the 10 excitation vectors in that sub-frame.

Let n = 1, 2, 3, 4 denote the sample time indices corresponding to the current four–dimensional excitation vector. Before the excitation VQ codebook search for the current excitation vector starts, the open-loop short-term prediction residual $d(n)$, n = 1, 2, 3, 4 has been calculated in Section 3.7. In addition, before the VQ codebook search starts, the initial filter states (also called "filter memory") of the three filters in Figure 8 (blocks 50, 60, and 65) are also known. All the other signals in Figure 8 are not determined yet for n = 1, 2, 3, 4.

The basic ideas of the excitation VQ codebook search are explained below. Refer to Figure 8. Block 48 stores the N scaled shape codevectors, where N = 32. Counting also the negated version of each scaled shape codevector, it is equivalent to having 2N scaled codevectors available for excitation VQ. From these 2N scaled codevectors, block 48 puts out one scaled codevector at a time as $uq(n)$, n = 1, 2, 3, 4. With the initial filter memories in blocks 50, 60, and 65 set to what were left after vector-quantizing the last excitation vector, this $uq(n)$ vector then "drives" the rest of the filter structure until the corresponding quantization error vector $q(n)$, n = 1, 2, 3, 4 is obtained. The energy of this $q(n)$ vector is calculated and stored. This process is repeated for each of the 2N scaled codevectors, with the filter memories reset to their initial values before the process is repeated each time. After all 2N codevectors have been tried, we have calculated 2N corresponding quantization error energy values. The scaled codevector that minimizes the energy of the quantization error vector $q(n)$, n = 1, 2, 3, 4 is the winning scaled codevector and is used as the VQ output vector. The corresponding output VQ codebook index is a 6-bit index consisting of a sign bit as the most significant bit (MSB), followed by 5 shape bits. If the winning scaled codevector is a negated version of a scaled shape codevector, then the sign bit is 1, otherwise, the sign bit is 0. The 5 shape bits are simply the binary representation of the codebook index of the winning shape codevector, as defined in Appendix 8. Note that there are 20 such excitation codebook indices in a frame, since each frame has 20 excitation vectors. These 20 indices are grouped in an excitation codebook index array, denoted as $CI = \{CI(1), CI(2), ..., CI(20)\}$, where $CI(k)$ is the excitation codebook index for the k-th excitation vector in the current frame. This excitation codebook index array CI is passed to the bit multiplexer block 95.

Given a $uq(n)$ vector (taking the value of one of the 2N scaled codevectors), the way to derive the corresponding energy of the $q(n)$ vector is now described in more detail below. First, block 60 performs pitch prediction to produce the pitch-predicted vector $ppv(n)$ as

$$ppv(n) = \sum_{i=1}^{3} b_i \; dq(n - pp + 2 - i) \;\; , n = 1, 2, 3, 4$$

Adder 85 then updates the $dq(n)$ vector as

$$dq(n) = uq(n) + ppv(n) \;\; , n = 1, 2, 3, 4$$

Next, block 50 and adders 90 and 55 work together to update the $v(n)$ vector as

$$v(n) = d(n) - \sum_{i=1}^{8} a_i' \; [v(n-i) - dq(n-i)] \;\; , n = 1, 2, 3, 4.$$

Finally, the corresponding $q(n)$ vector is calculated as

$$q(n) = v(n) - ppv(n) - \lambda\, q(n - pp) - uq(n)\,,\ n = 1, 2, 3, 4.$$

The energy of the $q(n)$ vector is calculated as

$$E_q = \sum_{n=1}^{4} q^2(n)\,.$$

Such calculation from a given $uq(n)$ vector to the corresponding energy term E_q is repeated $2N$ times for the $2N$ scaled VQ codevectors. After the winning scaled codevector that minimizes the E_q term is selected, the filter memories of blocks 50, 60, and 65 are updated by using the filter memories that were left after the calculation of the E_q term for that particular winning codevector was done. Such updated filter memories become the initial filter memories used for the excitation VQ codebook search for the next excitation vector.

3.12 Bit Multiplexing

The bit multiplexer block 95 in Figure 2 packs the five sets of indices *LSPI*, *PPI*, *PPTI*, *GI*, and *CI* into a single bit stream. This bit stream is the output of the BroadVoice32 encoder. It is passed to the communication channel.

Figure 9 shows the BV32 bit stream format in each frame. In Figure 9, the bit stream for the current frame is the shaded area in the middle. The bit stream for the last frame is on the left, while the bit stream for the next frame is on the right. Although the bit stream of different frames may not be sent next to each other in a packet voice system, this illustration is meant to show that time goes from left to right, and the 40 side information bits consisting of *LSPI*, *PPI*, *PPTI*, and *GI* goes before the excitation codebook indices $CI(k)$, k =1, 2, …, 20 when the bit stream is transmitted in a serial manner. Note that for each index, the most significant bit (MSB) goes first (on the left), while the least significant bit (LSB) goes last.

This completes the detailed description of the BV32 encoder.

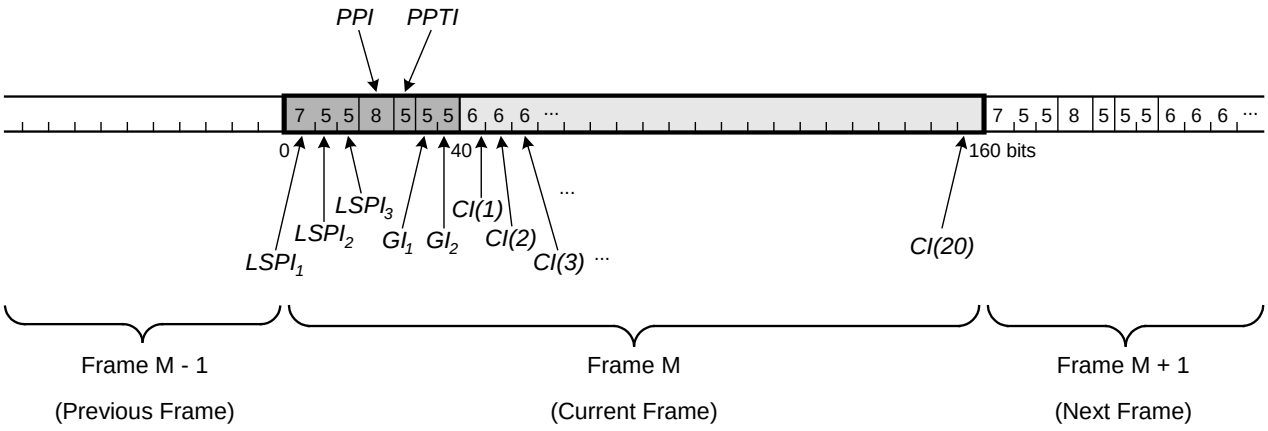

Figure 9 BV32 bit stream format

4 DETAILED DESCRIPTION OF THE BV32 DECODER

This section gives a detailed description of each functional block in the BV32 decoder shown in Figure 3. Those blocks or signals that have the same labels as their counterparts in the encoder of Figure 2 have the same meaning as those counterparts.

4.1 Bit De-multiplexing

The bit de-multiplexer block 100 takes one frame of input bit stream at a time, and de-multiplexes, or separates, the five sets of indices *LSPI, PPI, PPTI, GI,* and *CI* from the current frame of input bit stream. As described in Section 3 above, *LSPI* contains three indices: a 7-bit first-stage VQ index, a 5-bit second-stage lower VQ index, and a 5-bit second-stage upper VQ index. *PPI* is an 8-bit pitch period index. *PPTI* is a 5-bit pitch predictor tap VQ index. *GI* contains two 5-bit gain indices, and *CI* contains twenty 6-bit excitation VQ indices, each with 1 sign bit and 5 shape bits.

4.2 Long-Term Predictor Parameter Decoding

The long-term predictor parameter decoder (block 110) decodes the indices *PPI* and *PPTI*. The pitch period is decoded from *PPI* as

$$pp = PPI + 10$$

Let $\{\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_{31}\}$ be the 3-dimensional, 32-entry codebook used for pitch predictor tap VQ, as listed in Appendix 5. Let $\mathbf{b}_j$ be the *j*-th codevector in this codebook, where the subscript *j* is the codebook index listed in the first column of the table in Appendix 5. The three pitch predictor taps b_1, b_2, and b_3 are decoded from *PPTI* as

$$\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \mathbf{b}_{PPTI} \ .$$

4.3 Short-Term Predictor Parameter Decoding

The short-term predictor parameter decoding takes place in block 120 of Figure 3. Block 120 receives the set of decoded LSP indices, $LSPI = \{LSPI_1, LSPI_2, LSPI_3\}$, from the bit de-multiplexer, block 100 in Figure 3. First, block 120 reconstructs the LSP coefficients, $\{\tilde{l}_i\}$, from the LSP indices, and then it produces the coefficients of the short-term prediction error filter, $\{\tilde{a}_i\}$, from the LSP coefficients according to the conversion procedure specified in Section 3.6.

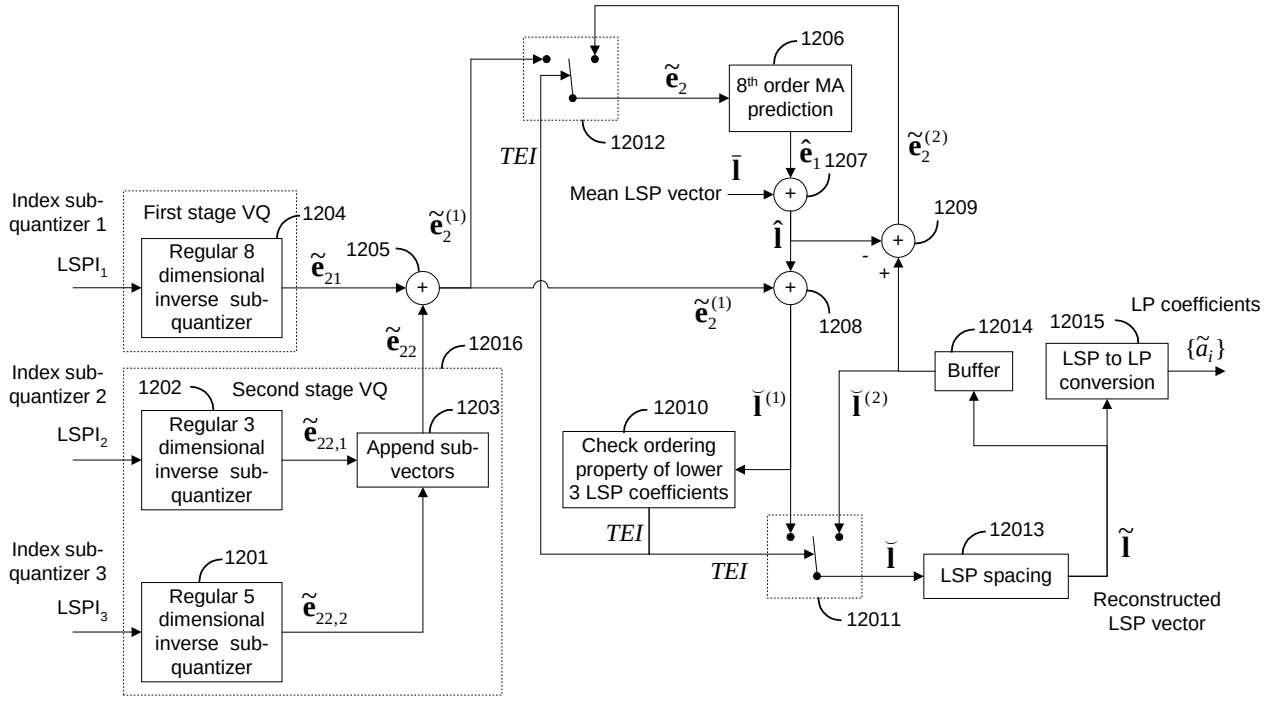

Figure 10 BV32 short-term predictor parameter decoder (block 120)

Block 120 of Figure 3 is expanded in Figure 10. The reconstruction of the LSP coefficients from the LSP indices is the inverse of the LSP quantization, and many operations have equivalents in Section 3.5 and Figure 5. The first-stage VQ is decoded in block 1204, and the second-stage split VQ is decoded in block 12016.

Based on the index for the second-stage upper split VQ, block 1201 looks up the quantized upper split vector from the codebook $\mathbf{CB}_{22} = \left\{ \mathbf{cb}_{22}{}^{(0)}, \mathbf{cb}_{22}{}^{(1)}, \ldots, \mathbf{cb}_{22}{}^{(31)} \right\}$, producing

$$\tilde{\mathbf{e}}_{22,2} = \mathbf{cb}_{22}{}^{(LSPI_3)}.$$

Similarly, based on the index for the second-stage lower split VQ, block 1202 looks up the quantized lower split vector from the codebook $\mathbf{CB}_{21} = \left\{ \mathbf{cb}_{21}{}^{(0)}, \mathbf{cb}_{21}{}^{(1)}, \ldots, \mathbf{cb}_{21}{}^{(31)} \right\}$, producing

$$\tilde{\mathbf{e}}_{22,1} = \mathbf{cb}_{21}{}^{(LSPI_2)}.$$

Block 1203 performs the identical operation of block 1610 in Figure 5 and appends the two second-stage sub-vectors to form the second-stage output vector,

$$\tilde{\mathbf{e}}_{22} = \begin{bmatrix} \tilde{\mathbf{e}}_{22,1} \\ \tilde{\mathbf{e}}_{22,2} \end{bmatrix}.$$

From the index for the first stage VQ, block 1204 looks up the quantized first stage vector from the codebook $\mathbf{CB}_1 = \left\{\mathbf{cb}_1^{(0)}, \mathbf{cb}_1^{(1)}, \ldots, \mathbf{cb}_1^{(127)}\right\}$,

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

Adder 1205 performs the equivalent operation of Adder 1611 in Figure 5. It adds the first-stage and second-stage vectors to obtain a first reconstructed prediction error vector,

$$\tilde{\mathbf{e}}_2^{(1)} = \tilde{\mathbf{e}}_{21} + \tilde{\mathbf{e}}_{22}.$$

Equivalent to block 163 in Figure 5, block 1206 performs the 8th-order MA prediction of the mean-removed LSP vector according to

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

where $\tilde{e}_{2,i}(k)$ and $\mathbf{p}_{LSP,i}$ are defined in Section 3.5. Adder 1207, equivalent to Adder 1612 in Figure 5, generates the predicted LSP vector by adding the mean LSP vector and the mean-removed predicted LSP vector,

$$\hat{\mathbf{l}} = \bar{\mathbf{l}} + \hat{\mathbf{e}}_1.$$

Subsequently, adder 1208 adds the predicted LSP vector to the first reconstructed prediction error vector to obtain a first intermediate reconstructed LSP vector,

$$\breve{\mathbf{l}}^{(1)} = \hat{\mathbf{l}} + \tilde{\mathbf{e}}_2^{(1)}.$$

Adder 1209 subtracts the predicted LSP vector from a second intermediate reconstructed LSP $\breve{\mathbf{l}}^{(2)}$, to calculate a second reconstructed prediction error vector

$$\tilde{\mathbf{e}}_2^{(2)} = \breve{\mathbf{l}}^{(2)} - \hat{\mathbf{l}} \quad ,$$

to be used to update the MA predictor memory in the presence of bit-errors. Block 12010 determines the ordering property of the first 3 first intermediate reconstructed LSP coefficients,

$$\begin{aligned} \breve{l}_1^{(1)} &\geq 0 \\ \breve{l}_2^{(1)} &\geq \breve{l}_1^{(1)}, \\ \breve{l}_3^{(1)} &\geq \breve{l}_2^{(1)} \end{aligned}$$

This ordering property was enforced during the encoding operation of the constrained 3-dimensional VQ of the lower split vector, block 168 of Figure 5. If the ordering is found to be preserved, the *Transmission-Error-Indicator*, *TEI* , is set to 0 to indicate that no bit-errors in the LSP bits have

been detected. Otherwise, if it is not preserved, the *Transmission-Error-Indicator* is set to 1 to indicate the likely presence of bit-errors in the LSP bits.

If the *Transmission-Error-Indicator* is 0, the switches 12011 and 12012 are in the left position, and they route the first reconstructed prediction error vector $\tilde{\mathbf{e}}_2^{(1)}$ and the first intermediate reconstructed LSP vector $\breve{\mathbf{l}}^{(1)}$ to the reconstructed prediction error vector $\tilde{\mathbf{e}}_2$ and the intermediate reconstructed LSP vector $\breve{\mathbf{l}}$, respectively. Otherwise, if the *Transmission-Error-Indicator* is 1, the switches 12011 and 12012 are in the right position, and they route the second reconstructed prediction error vector $\tilde{\mathbf{e}}_2^{(2)}$ and the second intermediate reconstructed LSP vector $\breve{\mathbf{l}}^{(2)}$ to the reconstructed prediction error vector $\tilde{\mathbf{e}}_2$ and the intermediate reconstructed LSP vector $\breve{\mathbf{l}}$, respectively. Hence, the reconstructed prediction error vector and the intermediate reconstructed LSP vector are obtained as

$$\tilde{\mathbf{e}}_2 = \begin{cases} \tilde{\mathbf{e}}_2^{(1)}, & \text{if } TEI = 0 \\ \tilde{\mathbf{e}}_2^{(2)}, & \text{if } TEI = 1 \end{cases}$$

and

$$\breve{\mathbf{l}} = \begin{cases} \breve{\mathbf{l}}^{(1)}, & \text{if } TEI = 0 \\ \breve{\mathbf{l}}^{(2)}, & \text{if } TEI = 1 \end{cases},$$

respectively. Block 12013 enforces LSP spacing; it is functionally identical to block 1614 in Figure 5, as specified in Section 3.5. Block 12014 buffers the reconstructed LSP vector for future use in the presence of bit-errors. The reconstructed LSP vector of the current frame becomes the second intermediate reconstructed LSP vector of the next frame,

$$\breve{\mathbf{l}}^{(2)}(k+1) = \tilde{\mathbf{l}}(k),$$

where the additional parameter k here represents the frame index of the current frame. For the very first frame the second intermediate reconstructed LSP vector is initialized to

$$\breve{\mathbf{l}}^{(2)} = \begin{bmatrix} 1/9 & 2/9 & \dots & 8/9 \end{bmatrix}^T$$

The final step of the short-term predictor parameter decoding is to convert the reconstructed LSP coefficients to linear prediction coefficients. This operation takes place in block 12015, which is functionally identical to block 17 of Figure 4, described in Section 3.6.

4.4 Excitation Gain Decoding

The excitation gain decoder is shown in Figure 11. It is part of block 130 in Figure 3. It decodes the two gain indices in *GI* into the two corresponding decoded sub-frame excitation gains *gq*(*m*), *m* = 1, 2 in the linear domain. All operations in Figure 11 are performed sub-frame by sub-frame.

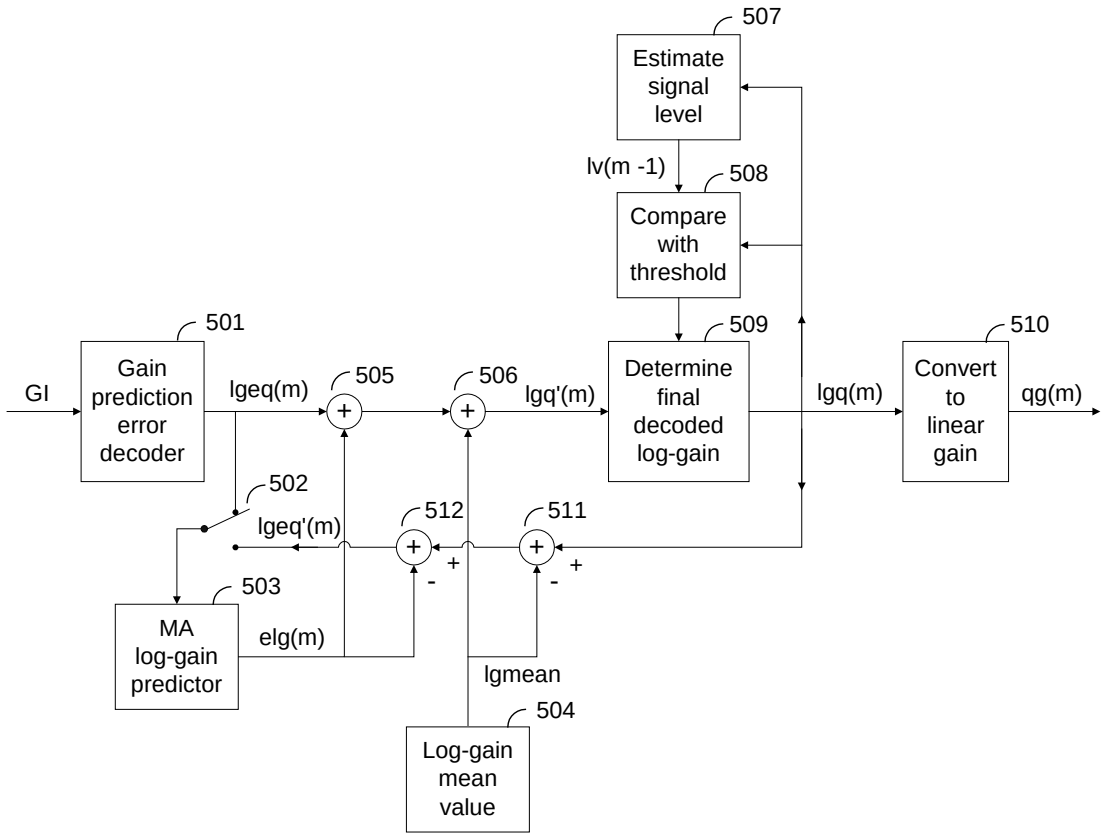

Figure 11 Excitation gain decoder

Refer to Figure 11. Let *m* be the sub-frame index of the current sub-frame, and assume the same convention for the sub-frame index *m* as in Section 3.10. Block 501 decodes the 5-bit gain index GI_m into the log-gain prediction error *lgeq*(*m*) using the codebook in Appendix 6. Switch 502 is normally in the upper position, connecting the output of block 501 to the input of block 503. Then, the MA log-gain predictor (block 503) calculates the estimated log-gain for the current sub-frame as

$$elg(m) = \sum_{k=1}^{GPO} lgp(k)lgeq(m-k) \ ,$$

where *GPO* = 16, and *lgp*(*k*), *k* = 1, 2, …, *GPO* are the MA log-gain predictor coefficients given in Section 3.10.

Block 504 holds the long-term average log-gain value *lgmean* = 11.82031. Adders 505 and 506 adds *elg*(*m*) and *lgmean*, respectively, to *lgeq*(*m*), resulting in the temporarily decoded log-gain of

$$lgq'(m) = lgeq(m) + elg(m) + lgmean \ .$$

Block 507 is functionally identical to block 309 in Figure 7, described in Section 3.10. It is important to note that equivalently to the encoder, the log-gain value passed to block 507 for updating its estimate of the long-term average signal level is the final value of the decoded log-gain *lgq*(*m*), i.e. after the threshold check of block 508 and potential log-gain extrapolation and substitution of block 509, respectively, as described below.

Block 508 calculates the row and column indices *i* and *j* into the threshold matrix *T*(*i, j*) in the same way as block 310 in Figure 7. Namely, the row index is calculated as

$$i = \left\lceil \frac{lgq(m-1) - lv(m-1) - GLB}{2} \right\rceil ,$$

where *GLB* = –24. If $i > NG$, i is clipped to *NG*. If $i < 1$, i is clipped to 1. The column index is calculated as

$$j = \left\lceil \frac{lgq(m-1) - lgq(m-2) - GCLB}{2} \right\rceil ,$$

where *GCLB* = –8. If $j > NGC$, j is clipped to *NGC*. If $j < 1$, j is clipped to 1.

Block 508 controls the actions of block 509 and switch 502 in the following way. If $GI_m = 0$ or $lgq'(m) \le T(i, j) + lgq(m-1)$, then switch 502 is in the upper position, block 509 determines the final decoded log-gain as

$$lgq(m) = lgq'(m) \ ,$$

and the filter memory in the MA log-gain predictor (block 503) is updated by shifting the old memory values by one position, and then assigning *lgeq*(*m*) to the newest position of the filter memory.

If, on the other hand, $GI_m > 0$ and $lgq(m) > T(i, j) + lgq(m-1)$, then the temporarily decoded log-gain $lgq'(m)$ is discarded, block 509 determines the final decoded log-gain as

$$lgq(m) = lgq(m-1)$$

(by extrapolating the decoded log-gain of the last sub-frame); furthermore, switch 502 is moved to the lower position, adders 511 and 512 subtract *lgmean* and *elg*(*m*), respectively, from $lgq(m)$ to get

$$lgeq'(m) = lgq(m) - lgmean - elg(m) ,$$

and this $lgeq'(m)$ is used to update the newest position of the filter memory of block 503, after the old memory values are shifted by one position.

Once the final decoded log-gain *lgq*(*m*) subject to the constraint imposed by block 509 is determined as described above, it is used by block 508 to update the estimated signal level *lv*(*m*). This value *lv*(*m*) is then used by block 509 in the next sub-frame (the (*m* + 1)-th sub-frame).

Block 510 converts final decoded log-gain *lgq*(*m*) to the linear domain as

$$gq(m) = 2^{\frac{lgq(m)}{2}} .$$

4.5 Excitation VQ Decoding and Scaling

The excitation codebook index array *CI* of each frame contains 20 excitation codebook indices, *CI*(*k*), *k* = 1, …, 20, each containing 1 sign bit and 5 shape bits. The excitation vectors are decoded vector-by-vector, and then sub-frame-by-sub-frame, since the excitation gain is updated once a sub-frame.

Suppose the current excitation vector that needs to be decoded is in the *m*-th sub-frame and has a corresponding excitation codebook index of *CI*(*k*). This index assumes a value between 0 and 63. The most significant bit of this index is the sign bit. Therefore, if *CI*(*k*) < 32, the sign bit is 0; otherwise, the sign bit is 1. Let $c_j(n)$, $n = 1, 2, 3, 4$ represent the *j*-th shape codevector in Appendix 8, with a shape codebook index of *j*. Furthermore, without loss of generality, let *n* = 1, 2, 3, 4 correspond to the sample time indices of the current vector. Then, in Figure 3, the decoded and scaled excitation vector, or *uq*(*n*), *n* = 1, 2, 3, 4, is obtained as

$$uq(n) = \begin{cases} gq(m)\, c_{CI(k)}(n), \ n = 1, 2, 3, 4, & \text{if } CI(k) < 32 \\ -gq(m)\, c_{CI(k)-32}(n), \ n = 1, 2, 3, 4, & \text{if } CI(k) \geq 32 \end{cases}$$

4.6 Long-Term Synthesis Filtering

Let *n* = 1, 2, …, *FRSZ* correspond to the sample time indices of the current frame. In Figure 3, the long-term synthesis filter (block 155, consisting of block 140 and adder 150 in a feedback loop) performs sample-by-sample long-term synthesis filtering as follows.

$$dq(n) = uq(n) + \sum_{i=1}^{3} b_i \, dq(n - pp + 2 - i), \ n = 1, 2, \ldots FRSZ.$$

4.7 Short-Term Synthesis Filtering

The short-term synthesis filter (block 175, consisting of block 160 and adder 170 in a feedback loop) performs sample-by-sample short-term synthesis filtering as follows.

$$sq(n) = dq(n) - \sum_{i=1}^{8} \tilde{a}_i \; sq(n-i)\,,\; n = 1, 2, \ldots FRSZ.$$

4.8 De-emphasis Filtering

The de-emphasis filter (block 180) is a first-order pole-zero filter with fixed coefficients. It is exactly the inverse filter of the pre-emphasis filter $H_{pe}(z)$ described in Section 3.2. This de-emphasis filter has the following transfer function.

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

Block 180 filters the short-term synthesis filter output signal *sq*(*n*) to produce the output signal of the entire decoder in Figure 3.

This completes the detailed description of the BV32 decoder.

4.9 Example Packet Loss Concealment

The packet loss concealment is not a mandatory component of this BV32 Codec Specification, since packet lost concealment does not affect bit-stream compatibility or encoder-decoder inter-operability. However, an example packet loss concealment technique is described in this section for reference purposes only. An implementer of BV32 can utilize other packet loss concealment techniques without affecting inter-operability.

The example packet loss concealment technique utilizes the synthesis model of the decoder. In principle, all side information of the previous frame is repeated while the excitation of the cascaded long-term and short-term synthesis filters is from a random source, scaled to a proper level. Hence, with the additional index *m* denoting the m-th frame, during packet-loss:

- The pitch period, *pp*, is set to the pitch period of the last frame[7]:
  $pp = pp_{m-1}$.
- The pitch taps, b_1 b_2 and b_3, are set to the pitch taps of the last frame[8].
  $b_i = b_{m-1,i}$, i=1,2,3.
- The short-term synthesis filter coefficients, $\tilde{a}_i, i = 1,...,8$, are set to those of the last frame[9]:

[7] If the first frame is lost a value of 100 is used for the pitch period.
[8] If the first frame is lost the pitch taps are set to zero.
[9] If the first frame is lost the short-term filter coefficients are set to zero.

$\tilde{a}_i = \tilde{a}_{m-1,i}$, i=1,…,8.

- A properly scaled random sequence is used as long-term synthesis filter excitation, $uq(n)$, n = 1, 2, … *FRSZ*.

The speech synthesis of the bad frame (part of lost packet) now takes place exactly as specified in Sections 4.6, 4.7, and 4.8.

The random sequence is scaled according to

$$uq(n) = g_{plc} \cdot \sqrt{\frac{E_{m-1}}{\sum_{n=1}^{FRSZ}[r(n)]^2}} \cdot r(n)\,,\ n = 1, 2, \ldots FRSZ,$$

where $r(n)$, n = 1, 2, … *FRSZ*, is a random sequence, E_{m-1} is in principle the energy of the long-term synthesis filter excitation of the previous frame[10], and the scaling factor, g_{plc}, is calculated as detailed below.

During good frames an estimate of periodicity is updated as

$$per_m = 0.5\ per_{m-1} + 0.5\,bs\,,$$

where *bs* is the sum of the three pitch taps clipped at a lower threshold of zero and an upper threshold of one[11], while it is maintained during bad frames: $per_m = per_{m-1}$. Based on the periodicity the scaling factor is calculated as

$$g_{plc} = -2\ per_{m-1} + 1.9$$

with g_{plc} clipped at a lower threshold of 0.1 and an upper threshold of 0.9.

After synthesis of the signal output of a lost frame, memories of predictive quantizers are updated.

The memory of the inverse LSP quantizer is updated with

$$\tilde{e}_{2,i} = \tilde{I}_{m-1,i} - \hat{e}_{1,i} - \bar{I}_i\,,\ \text{i=1,2,…,8,}$$

where $\hat{e}_{1,i}$ is given in Section 4.3, $\bar{I}_i$ in Section 3.5, and $\tilde{I}_{m-1,i}$ denotes the i-th LSP coefficients of the (m-1)-th frame (as decoded according to Section 4.3 for a good frame, or repeated for a bad frame).

The memory of the inverse gain quantizer is updated with

[10] The energy is initialized to zero, i.e. E_0=0.
[11] The estimate of periodicity is initialized to zero, i.e. per_0=0.

$$lgeq(m) = lgq(m) - lgmean - elg(m),$$

where $elg(m)$ is given in Section 4.4, *lgmean* in Section 3.10, and *lgq*(*m*) is calculated as

$$lgq(m) = \begin{cases} \log_2 \frac{E_{m-1}}{FRSZ}, & \text{if } \frac{E_{m-1}}{FRSZ} > 1/4 \\ -2, & \text{if } \frac{E_{m-1}}{FRSZ} \le 1/4 \end{cases}.$$

The level estimation for a bad frame is updated exactly as for a good frame, see Section 4.4.

At the end of a good frame (after synthesis of the output) the estimate of periodicity is estimated as explained above, and the energy of the long-term synthesis filter excitation is updated as

$$E_m = \sum_{n=1}^{FRSZ} [uq(n)]^2 .$$

At the end of the processing of a bad frame (after synthesis of the output and update of predictive quantizers), the energy of the long-term synthesis filter excitation and the long-term synthesis filter coefficients are scaled down when 8 or more consecutive frames are lost:

$$E_m = \begin{cases} E_{m-1} & Nclf < 8 \\ (\beta_{Nclf})^2 \, E_{m-1} & Nclf \ge 8 \end{cases},$$

$$b_{m,i} = \begin{cases} b_{m-1,i} & Nclf < 8 \\ \beta_{Nclf} \, b_{m-1,i} & Nclf \ge 8 \end{cases}, \quad \text{i=1,2,3,}$$

where *Nclf* is the number of consecutive lost frames, and the scaling, β_{Nclf}, is given by

$$\beta_{Nclf} = \begin{cases} 1 - 0.02\,(Nclf - 7) & 8 \le Nclf \le 57 \\ 0 & Nclf > 57 \end{cases}.$$

This will gradually mute the output signal when consecutive packets are lost for an extended period of time.

APPENDIX 1: GRID FOR LPC TO LSP CONVERSION

Grid point	Grid value
0	0.9999390
1	0.9935608
2	0.9848633
3	0.9725342
4	0.9577942
5	0.9409180
6	0.9215393
7	0.8995972
8	0.8753662
9	0.8487854
10	0.8198242
11	0.7887573
12	0.7558899
13	0.7213440
14	0.6853943
15	0.6481323
16	0.6101379
17	0.5709839
18	0.5300903
19	0.4882507
20	0.4447632
21	0.3993530
22	0.3531189
23	0.3058167
24	0.2585754
25	0.2109680
26	0.1630859
27	0.1148682
28	0.0657349
29	0.0161438
30	-0.0335693
31	-0.0830994
32	-0.1319580
33	-0.1804199
34	-0.2279663
35	-0.2751465
36	-0.3224487
37	-0.3693237
38	-0.4155884
39	-0.4604187
40	-0.5034180
41	-0.5446472
42	-0.5848999
43	-0.6235962
44	-0.6612244
45	-0.6979980
46	-0.7336731
47	-0.7675781
48	-0.7998962
49	-0.8302002
50	-0.8584290
51	-0.8842468
52	-0.9077148
53	-0.9288635
54	-0.9472046
55	-0.9635010
56	-0.9772034
57	-0.9883118
58	-0.9955139

59	-0.9999390

APPENDIX 2: FIRST-STAGE LSP CODEBOOK

Index	Element 1	Element 2	Element 3	Element 4	Element 5	Element 6	Element 7	Element 8
0	-0.00384521	-0.00849915	-0.01591492	-0.00360107	-0.00013733	0.00610352	0.01640320	-0.00166321
1	-0.00511169	-0.01313782	-0.01698303	-0.00103760	-0.01216125	-0.00427246	-0.00271606	0.00846863
2	-0.00367737	-0.00166321	0.00045776	-0.00309753	0.01814270	-0.00053406	0.00256348	-0.00833130
3	-0.00312805	-0.00488281	0.00282288	-0.00173950	0.00004578	-0.00094604	-0.01976013	0.00306702
4	-0.00250244	-0.00323486	0.00154114	0.00422668	-0.00964355	-0.01895142	0.01704407	0.00219727
5	-0.00090027	-0.00347900	-0.00909424	-0.00746155	-0.00656128	-0.02726746	-0.00769043	-0.00224304
6	0.00399780	0.01086426	0.00677490	0.00090027	0.00244141	-0.00988770	0.00549316	-0.00628662
7	-0.00151062	-0.00581360	-0.00186157	-0.00430298	-0.01788330	-0.01603699	-0.03099060	-0.00659180
8	-0.00547791	-0.00958252	0.00094604	0.01203918	0.00695801	0.02105713	0.00720215	0.00140381
9	-0.00393677	-0.00848389	-0.01943970	-0.01473999	0.01364136	-0.00468445	-0.00344849	0.00566101
10	-0.00331116	-0.00723267	0.00175476	0.03128052	0.00772095	-0.00163269	0.00566101	-0.00460815
11	-0.00222778	-0.00709534	-0.00581360	0.01132202	-0.00482178	-0.00050354	-0.01037598	-0.01887512
12	-0.00325012	-0.00445557	0.00651550	0.00497437	-0.01744080	0.01000977	0.01194763	-0.00160217
13	-0.00054932	-0.00219727	-0.00631714	-0.01139832	-0.01916504	-0.00711060	0.00106812	-0.01481628
14	-0.00546265	0.00070190	0.02934265	0.01412964	0.00656128	0.00003052	0.01229858	0.00367737
15	-0.00254822	0.00099182	0.02000427	-0.00164795	-0.01643372	-0.00813293	-0.00671387	-0.01013184
16	-0.00204468	0.00265503	-0.00135803	-0.02322388	0.00332642	0.01715088	0.01350403	0.00199890
17	-0.00289917	-0.00740051	-0.01710510	-0.02655029	-0.01350403	0.00151062	-0.00038147	0.00778198
18	-0.00028992	0.00064087	0.00022888	-0.00819397	0.00061035	0.02536011	-0.00822449	-0.02096558
19	-0.00028992	0.00001526	-0.00805664	-0.02310181	-0.00082397	-0.00106812	-0.02081299	-0.01762390
20	-0.00030518	0.00170898	-0.00651550	-0.01683044	0.00083923	-0.00955200	0.02677917	0.00958252
21	0.00292969	0.00251770	-0.00447083	-0.01782227	-0.02940369	-0.02981567	0.00372314	-0.00421143
22	0.01701355	0.02578735	-0.00593567	0.00595093	0.01370239	0.01223755	0.00622559	-0.00111389
23	0.00061035	-0.00015259	0.00686646	0.00129700	-0.00637817	-0.02079773	-0.05078125	-0.01544189
24	-0.00398254	0.00350952	0.01591492	-0.00076294	0.02429199	0.02890015	0.01559448	0.00701904
25	-0.00177002	-0.00981140	-0.03118896	-0.01042175	-0.00013733	0.00044250	-0.00659180	-0.01545715
26	0.00256348	0.01017761	0.01966858	0.01533508	0.01405334	0.01646423	-0.00257874	-0.01338196
27	0.00088501	-0.00016785	-0.00163269	-0.00199890	-0.00700378	-0.00726318	-0.02569580	-0.03907776
28	0.00035095	0.00717163	0.00427246	0.00279236	0.02046204	0.00689697	0.02848816	0.01043701
29	0.00041199	0.00004578	-0.01815796	-0.03132629	-0.00378418	-0.02220154	0.00140381	-0.00294495
30	0.01571655	0.02601624	0.01066589	0.03164673	0.03356934	0.02770996	0.01812744	0.00709534
31	0.00881958	0.02149963	0.01010132	0.00360107	0.00122070	-0.00657654	-0.01893616	-0.02380371
32	-0.00672913	-0.01612854	-0.02481079	-0.00184631	0.00761414	0.01754761	0.00720215	0.01480103
33	-0.00515747	-0.01365662	-0.01542664	-0.01049805	-0.01742554	0.02040100	-0.00880432	-0.00152588
34	-0.00303650	-0.00975037	-0.02221680	0.01498413	0.02423096	0.00935364	-0.00544739	-0.00675964
35	-0.00221252	-0.00933838	-0.02006531	0.00033569	0.00292969	-0.01268005	-0.02940369	-0.00543213
36	-0.00231934	-0.00257874	0.00263977	-0.00134277	-0.00151062	-0.00566101	0.00665283	0.03112793
37	-0.00123596	-0.00584412	-0.01034546	-0.01982117	-0.02880859	-0.02052307	-0.01663208	0.00572205
38	0.00738525	0.02700806	0.01812744	0.02203369	0.00323486	-0.00514221	0.01075745	0.00660706
39	0.00349426	0.00294495	-0.00387573	-0.01075745	-0.02171326	-0.03224182	-0.02403259	-0.02343750
40	-0.00619507	-0.01358032	-0.01676941	0.01498413	0.02687073	0.02645874	0.01818848	0.01010132
41	-0.00459290	-0.00839233	-0.02026367	-0.02606201	0.02151489	0.02061462	-0.00651550	-0.00538635
42	-0.00405884	-0.00538635	0.00645447	0.03422546	0.03749084	0.02166748	0.00497437	-0.00592041
43	-0.00209045	-0.00204468	-0.00219727	0.00228882	0.02597046	0.00415039	-0.02684021	-0.01873779
44	-0.00489807	-0.00955200	-0.00572205	0.00482178	-0.00778198	0.01531982	0.03317261	0.01727295
45	-0.00341797	-0.00909424	-0.00500488	-0.00860596	-0.04263306	-0.00547791	0.00357056	0.00357056
46	-0.00016785	0.01191711	0.03486633	0.03454590	0.02195740	0.01472473	0.03034973	0.02073669
47	-0.00109863	0.00473022	0.01737976	0.00859070	-0.00253296	-0.03044128	-0.00776672	-0.01104736
48	-0.00527954	-0.00999451	-0.00939941	-0.00805664	-0.00268555	0.04862976	0.01870728	0.00442505
49	-0.00317383	-0.00744629	-0.00877380	-0.02050781	-0.03236389	0.01905823	0.01884460	0.00524902
50	0.00453186	0.01782227	0.00762939	-0.00749207	0.03543091	0.01852417	-0.00367737	-0.01086426
51	0.00018311	-0.00355530	-0.01539612	-0.02656555	-0.00277710	-0.01931763	-0.03083801	0.00360107
52	-0.00143433	0.00292969	0.01277161	0.00936890	0.00128174	-0.00985718	0.04154968	0.02775574
53	0.00213623	0.00561523	0.00642395	-0.00889587	-0.03330994	-0.05546570	0.00897217	0.00265503
54	0.01060486	0.05717468	0.03829956	0.03216553	0.02561951	0.02203369	0.01969910	0.00923157
55	0.00221252	0.00817871	0.01704407	-0.00007629	-0.00616455	-0.04737854	-0.03558350	0.00561523
56	-0.00749207	-0.00627136	0.02369690	0.02711487	0.03462219	0.04241943	0.02859497	0.01635742
57	-0.02087402	-0.04931641	0.00619507	0.00404358	0.01080322	0.00926208	0.00779724	0.00225830
58	-0.00173950	0.01293945	0.04112244	0.03024292	0.03976440	0.03063965	0.00881958	-0.00358582
59	-0.00424194	-0.00158691	0.02459717	0.01078796	0.00611877	0.00105286	-0.02471924	-0.02410889

60	-0.00451660	-0.00415039	0.00253296	0.01228333	0.02276611	0.02371216	0.05001831	0.02963257
61	-0.00369263	-0.01776123	-0.03298950	-0.01219177	-0.03230286	-0.02035522	-0.01049805	-0.00700378
62	0.01309204	0.03527832	0.04226685	0.04809570	0.04991150	0.04533386	0.03337097	0.01974487
63	0.00236511	0.01925659	0.04072571	0.02778625	0.01647949	-0.01173401	-0.02360535	-0.01696777
64	-0.00433350	-0.01188660	-0.02235413	0.01066589	0.01145935	-0.00656128	0.02409363	0.01565552
65	-0.00448608	-0.01176453	-0.02374268	-0.01464844	-0.01629639	-0.01852417	0.01446533	0.01126099
66	-0.00320435	0.00030518	0.00944519	0.01014709	0.03031921	0.00007629	-0.00328064	0.01599121
67	-0.00141907	-0.00477600	-0.00032043	-0.00436401	-0.00563049	-0.02128601	-0.03314209	0.02626038
68	-0.00105286	-0.00151062	-0.00180054	-0.00811768	-0.02941895	-0.01837158	0.03617859	0.01126099
69	-0.00238037	-0.00828552	-0.00988770	0.00376892	-0.02708435	-0.03489685	-0.00431824	0.00047302
70	0.01274109	0.02935791	0.00981140	-0.00921631	-0.01629639	-0.00587463	0.00247192	0.00064087
71	0.00193787	-0.00151062	-0.00468445	-0.01261902	-0.02470398	-0.03384399	-0.04949951	-0.00338745
72	-0.00361633	-0.00816345	0.00148010	0.03401184	0.01333618	0.01911926	0.02272034	0.01939392
73	-0.00132751	-0.00799561	-0.02526855	-0.03221130	0.00328064	0.00810242	0.00950623	0.01345825
74	-0.00360107	-0.00260925	0.03428650	0.04959106	0.01815796	0.00881958	-0.00042725	-0.00680542
75	-0.00175476	-0.00695801	0.00030518	0.02726746	-0.00277710	-0.01660156	-0.02694702	-0.01084900
76	-0.00105286	0.00723267	0.02352905	0.00462341	-0.01211548	0.02276611	0.02523804	0.01177979
77	0.00044250	-0.00314331	-0.00833130	-0.02253723	-0.03590393	0.00534058	-0.01576233	-0.01797485
78	0.00265503	0.02203369	0.05549622	0.02410889	0.00866699	0.00965881	0.00958252	-0.00190735
79	0.00163269	0.00399780	0.01559448	-0.00083923	-0.03933716	-0.01277161	-0.02479553	-0.01690674
80	-0.00450134	-0.00598145	-0.01719666	-0.02134705	0.02500916	0.02310181	0.02972412	0.01644897
81	-0.00181580	-0.00743103	-0.02114868	-0.03652954	-0.03193665	-0.00167847	0.00451660	0.00935364
82	-0.00350952	-0.00422668	-0.00115967	0.00111389	0.03088379	0.04490662	-0.00390625	-0.01063538
83	0.00189209	-0.00177002	-0.01432800	-0.02612305	-0.01161194	-0.01190186	-0.04681396	-0.02130127
84	-0.00234985	-0.00527954	-0.01350403	-0.01855469	-0.00726318	-0.00196838	0.04997253	0.02980042
85	0.00024414	-0.00541687	-0.01794434	-0.02980042	-0.03829956	-0.04582214	0.01480103	0.01237488
86	0.05004883	0.03166199	0.02220154	0.01562500	0.00930786	0.00764465	0.00833130	0.00251770
87	0.00430298	0.00444031	0.00691223	-0.00653076	-0.01719666	-0.04112244	-0.09020996	-0.00898743
88	0.00041199	0.00929260	0.00347900	0.00259399	0.05375671	0.03878784	0.02937317	0.01449585
89	-0.00491333	-0.02757263	-0.06730652	-0.02465820	-0.00869751	-0.00566101	-0.00590515	0.00354004
90	0.01916504	0.03500366	0.02929688	0.03329468	0.02725220	0.01902771	-0.00694275	-0.01644897
91	0.00370789	0.00387573	0.00061035	-0.00419617	-0.01568604	-0.02262878	-0.05206299	-0.04679871
92	0.01432800	0.03143311	0.01612854	0.00932312	0.01620483	0.02969360	0.03417969	0.02700806
93	0.00350952	0.00082397	-0.03111267	-0.06707764	-0.02024841	-0.01860046	-0.00958252	-0.00173950
94	0.05522156	0.04231262	0.04219055	0.03793335	0.03443909	0.03150940	0.02209473	0.01277161
95	0.03974915	0.03291321	0.01431274	0.00024414	0.00086975	-0.01142883	-0.03588867	-0.01281738
96	-0.00840759	-0.02593994	-0.04820251	0.00361633	0.01782227	0.03044128	0.02810669	0.02386475
97	-0.00660706	-0.02162170	-0.03446960	-0.01261902	-0.02426147	0.01382446	0.01550293	0.01689148
98	-0.00529480	-0.00663757	-0.01538086	-0.00068665	0.05569458	0.01844788	0.00303650	0.00178528
99	-0.00257874	-0.00895691	-0.00971985	-0.00666809	0.00314331	-0.00125122	-0.05572510	0.00030518
100	-0.00048828	0.00041199	-0.00028992	-0.00938416	-0.00831604	-0.03677368	0.01962280	0.03395081
101	0.00151062	-0.00248718	-0.01004028	-0.02021790	-0.03338623	-0.05041504	-0.02108765	0.00358582
102	0.01791382	0.06657410	0.02952576	0.01698303	0.00154114	-0.00361633	0.00166321	-0.00538635
103	0.00483704	0.01110840	0.01173401	-0.00868225	-0.03361511	-0.06233215	-0.03771973	-0.02009583
104	-0.00576782	-0.01533508	-0.01855469	0.03782654	0.04870605	0.04002380	0.02944946	0.01617432
105	-0.00526428	-0.01495361	-0.03617859	-0.05659485	0.00096130	0.01994324	0.01362610	0.00981140
106	-0.00202942	-0.00065613	0.02978516	0.07008362	0.05216980	0.03585815	0.01889038	0.00332642
107	-0.00016785	0.00070190	0.00903320	0.03486633	0.03285217	0.00875854	-0.03100586	-0.03533936
108	-0.00460815	-0.00889587	-0.00535583	-0.00605774	-0.01533508	0.03904724	0.05235291	0.02601624
109	0.00132751	-0.00230408	-0.00685120	-0.03175354	-0.06939697	-0.02249146	-0.01206970	-0.00375366
110	-0.00267029	0.01330566	0.07746887	0.05206299	0.03462219	0.02912903	0.02680969	0.01100159
111	0.00160217	0.01959229	0.05360413	0.01110840	-0.02023315	-0.03753662	-0.01402283	-0.01716614
112	-0.00639343	-0.01054382	-0.00729370	0.00303650	0.02307129	0.07855225	0.04028320	0.01892090
113	-0.00352478	-0.01866150	-0.02894592	-0.03585815	-0.06474304	-0.00245667	0.02102661	0.01512146
114	-0.00077820	0.01048279	0.01577759	0.02217102	0.07875061	0.04440308	0.00306702	-0.00975037
115	0.00099182	-0.01753235	-0.02914429	-0.03764343	-0.03930664	-0.04081726	-0.06845093	-0.01873779
116	-0.00256348	-0.00033569	0.00053406	0.00070190	-0.00105286	0.00088501	0.08264160	0.03260803
117	0.00869751	0.00711060	-0.00775146	-0.03376770	-0.06314087	-0.08934021	-0.01795959	0.00088501
118	0.02838135	0.09507751	0.06649780	0.05419922	0.04470825	0.03926086	0.02592468	0.01179504
119	0.04316711	0.05152893	0.01130676	-0.02204895	-0.04406738	-0.06632996	-0.08439636	-0.05546570
120	-0.00311279	0.00325012	0.02406311	0.04458618	0.07960510	0.07987976	0.04357910	0.01593018
121	-0.02593994	-0.07890320	-0.02648926	-0.02957153	-0.01586914	-0.01681519	-0.00686646	-0.00469971
122	0.00602722	0.03062439	0.07060242	0.06431580	0.04623413	0.02545166	-0.00128174	-0.01522827
123	0.00640869	0.01506042	0.02645874	0.01609802	0.00148010	-0.01939392	-0.05572510	-0.06500244
124	-0.00239563	0.00444031	0.01907349	0.03089905	0.03352356	0.05075073	0.07539368	0.03486633

125	-0.00079346	-0.03021240	-0.05854797	-0.07080078	-0.06494141	-0.05015564	-0.02285767	-0.00508118
126	0.00588989	0.03402710	0.08795166	0.09323120	0.07124329	0.05776978	0.03340149	0.01075745
127	0.02717590	0.07472229	0.08680725	0.03575134	0.00018311	-0.03523254	-0.05368042	-0.04931641

APPENDIX 3: SECOND-STAGE LOWER SPLIT LSP CODEBOOK

Index	Element 1	Element 2	Element 3
0	0.00281525	0.00292778	0.00433731
1	-0.00021553	-0.00037766	-0.00252151
2	0.00709152	-0.00558853	-0.00040245
3	-0.00341034	-0.00456047	0.00535393
4	-0.00196075	0.00144005	0.01340103
5	-0.00179482	-0.00482559	-0.00926208
6	-0.00576019	0.00680923	0.00318718
7	-0.00498390	-0.01045990	-0.00181580
8	0.00724030	0.00892258	-0.00010681
9	-0.00100517	0.00750542	-0.01124763
10	0.01622772	0.00503349	-0.00928497
11	-0.01317978	-0.00148201	-0.00485039
12	0.00139236	0.01294518	0.01284790
13	0.00160599	-0.00276566	-0.02051735
14	0.00048065	0.02153206	-0.00239372
15	0.00121498	-0.01841927	0.00706482
16	0.01221657	0.00114632	0.01258469
17	0.00564766	0.00059319	-0.00907707
18	0.02144051	-0.01291847	-0.00042725
19	-0.01160431	-0.01168442	0.01208878
20	-0.00497437	-0.00429916	0.02562332
21	-0.00357437	-0.01308441	-0.01529694
22	-0.01611328	0.01459503	0.00725365
23	-0.01193810	-0.02121544	-0.00399017
24	0.01710129	0.01618958	0.00624657
25	0.00753784	0.01832008	-0.02398491
26	0.03960609	0.01548195	-0.00556374
27	-0.03484535	0.00230217	0.00053406
28	-0.00045013	0.01565170	0.03667641
29	-0.00150681	-0.01651573	-0.03601646
30	0.00778198	0.04269028	0.00644302
31	-0.01263237	-0.04002953	0.00638008

APPENDIX 4: SECOND-STAGE UPPER SPLIT LSP CODEBOOK

Index	Element 1	Element 2	Element 3	Element 4	Element 5
0	0.00223160	-0.00800133	-0.00899124	0.00006485	0.00058365
1	0.00498199	0.00384903	-0.00713539	-0.00961494	-0.00307274
2	-0.00000954	0.00230217	0.00827026	0.00367355	0.00186920
3	0.00362587	0.01415634	0.00111580	0.00265884	-0.00458145
4	-0.01116562	0.00059700	-0.01137161	0.00316811	-0.00823975
5	0.00366402	0.00034904	-0.00654984	0.00271797	-0.01940155
6	-0.00282288	-0.00809288	0.00408554	-0.00595474	-0.00964355
7	0.01284599	0.00154495	0.00731087	0.00330925	-0.00998116
8	-0.00849152	-0.00714302	0.00018120	0.00532913	0.00732613
9	-0.00639915	0.00654030	-0.00492859	-0.00344276	0.01243401
10	-0.00438499	0.00685120	-0.00248146	0.01663589	0.00031281
11	0.01028252	0.00627327	-0.00315285	0.00683403	0.00990868
12	-0.01620674	0.00895309	0.00953102	0.00367737	-0.00362778
13	-0.00172234	0.00682259	0.00998497	-0.01184273	0.00318718
14	-0.00300217	-0.00821686	0.00954819	0.01287270	-0.00807762
15	0.01217651	-0.00773621	0.00847435	-0.00031281	0.00645638
16	-0.00471497	-0.01052666	-0.02195930	-0.01058769	0.00412560
17	0.00894547	-0.00356674	-0.00493240	-0.02550888	-0.00962448
18	-0.00122452	0.00730324	0.01606369	0.01205063	0.01569366
19	-0.00556946	0.02675247	-0.00582695	-0.00326729	0.00189209
20	-0.01784134	0.00078583	-0.00429535	-0.01312637	-0.00244522
21	-0.00508881	0.00965881	0.00708389	-0.01148987	-0.02126884
22	-0.00472450	-0.01339912	0.00592613	-0.01262474	0.00816154
23	0.02260780	0.01769447	0.00827408	-0.00707054	-0.00349998
24	-0.00877571	-0.00870895	-0.01420212	0.01482201	0.01783562
25	0.00730515	0.00027847	-0.00198555	-0.01367950	0.02097321
26	0.00929070	-0.00706673	-0.00564384	0.01904678	0.00018692
27	0.01049805	0.01000977	-0.02177620	0.00494194	0.00013351
28	-0.02701187	-0.01168251	0.01052856	0.00321388	0.00094223
29	0.00286293	-0.00534248	0.02644157	-0.00658035	-0.00415039
30	0.00362587	-0.02618980	0.00177765	0.00383186	-0.00398064
31	0.02854538	-0.00962830	-0.00597000	-0.00085640	-0.00148964

APPENDIX 5: PITCH PREDICTOR TAB CODEBOOK

Index	Element 1	Element 2	Element 3
0	-0.1450195	0.2992860	0.1412050
1	-0.1288755	0.6889955	0.4095765
2	-0.0344850	0.2010195	0.1164550
3	-0.0050965	0.5024415	0.3855895
4	-0.0058290	0.2820435	0.0539245
5	0.2434080	0.4494935	0.2871705
6	-0.0683900	-0.2305605	0.0762635
7	0.0208435	0.3207705	0.2186585
8	0.0831910	0.3331300	0.0210265
9	0.1194460	0.6207885	0.2395935
10	0.0783080	0.1853335	0.0917360
11	0.1614380	0.4213255	0.3347170
12	0.1701965	0.2445375	0.0487670
13	0.3446655	0.4873655	0.1487425
14	0.1160280	-0.2495730	0.0732725
15	0.2894895	0.3384705	0.2445070
16	-0.1333315	0.3481750	-0.0697630
17	-0.0087890	0.8484800	0.1400755
18	-0.0806275	0.2053835	-0.0161745
19	-0.0332030	0.4768980	0.1187745
20	0.0019225	0.2650755	-0.1307375
21	0.2192995	0.7223815	0.0311585
22	0.0077210	0.0712585	0.0126040
23	0.1473085	0.2614745	0.1786500
24	0.1340940	0.4643860	-0.1044920
25	0.4174500	0.6805420	-0.1161805
26	0.0624085	0.1940615	-0.0299375
27	0.1651305	0.4480895	0.1498110
28	0.1640320	0.2704775	-0.1030580
29	0.3888245	0.5142820	-0.0249940
30	0.0041200	-0.2192080	-0.0823975
31	0.2724305	0.3651735	0.0523375

APPENDIX 6: GAIN CODEBOOK

Index	Element
0	-4.91895
1	-3.75049
2	-3.09082
3	-2.59961
4	-2.22656
5	-1.46240
6	-0.88037
7	-0.34717
8	-1.93408
9	-1.25635
10	-0.70117
11	-0.16650
12	0.20361
13	0.82568
14	1.59863
15	2.75684
16	-1.68457
17	-1.06299
18	-0.52588
19	0.01563
20	0.39941
21	1.05664
22	1.91602
23	3.34326
24	0.60693
25	1.31201
26	2.29736
27	4.11426
28	5.20996
29	6.70410
30	8.74316
31	10.92188

APPENDIX 7: GAIN CHANGE THRESHOLD MATRIX

Relative log-gain of previous frame, [dB2]			Log-gain change of previous frame, [dB2]										
			-8 to -6 j=1	-6 to -4 j=2	-4 to -2 j=3	-2 to 0 j=4	0 to 2 j=5	2 to 4 j=6	4 to 6 j=7	6 to 8 j=8	8 to 10 j=9	10 to 12 j=10	12 to 14 j=11
	-24 to –22	i=1	0.00000	0.13477	2.26563	2.94336	4.71875	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
	-22 to –20	i=2	0.00000	0.64453	4.90039	3.38281	4.58203	5.69336	0.00000	0.00000	0.00000	0.00000	0.00000
	-20 to –18	i=3	0.00000	0.33594	7.27734	5.82422	11.66211	11.66211	0.00000	0.00000	0.00000	0.00000	0.00000
	-18 to –16	i=4	6.31250	5.50977	4.83984	6.99023	8.22852	11.49805	1.89844	0.00000	0.00000	0.00000	0.00000
	-16 to –14	i=5	0.00000	5.04883	5.09180	5.91406	6.92188	7.38086	4.13867	0.00000	0.00000	0.00000	0.00000
	-14 to –12	i=6	-0.36523	6.15625	8.26953	5.40430	5.88477	11.53906	5.31836	-4.97070	0.00000	0.00000	0.00000
	-12 to –10	i=7	5.51172	6.31641	9.66602	7.58594	10.63281	12.03906	8.79297	3.06836	0.00000	0.00000	0.00000
	-10 to –8	i=8	3.95703	10.51172	8.42969	7.62891	11.45703	11.95898	10.85352	2.83008	1.50000	0.00000	0.00000
	-8 to –6	i=9	7.37305	8.93945	8.57422	6.85742	9.67773	11.54492	10.98242	10.43359	2.53320	5.05859	0.00000
	-6 to –4	i=10	7.37305	8.12109	6.66406	5.87891	7.59766	10.67969	10.42578	9.46875	6.85938	3.06445	0.00000
	-4 to –2	i=11	4.39844	5.94336	5.73047	5.10742	5.69531	8.31641	10.05273	8.23047	7.11328	3.04102	-1.27930
	-2 to 0	i=12	0.58789	5.10938	5.41602	4.55273	4.32813	5.75586	7.42383	6.63867	6.81055	4.14258	3.31641
	0 to 2	i=13	0.14453	5.64844	5.05859	4.06836	3.51758	4.07617	4.56055	4.99219	5.51953	4.82227	5.19141
	2 to 4	i=14	0.00000	5.54688	5.15625	3.37891	2.90430	2.74805	2.82422	3.37500	4.02930	4.49805	3.42188
	4 to 6	i=15	0.00000	0.39258	3.92188	2.67383	2.66602	2.40039	4.65039	3.29883	2.16016	2.95703	0.40820
	6 to 8	i=16	0.00000	0.00000	1.15039	2.56641	3.98438	3.61133	4.66797	0.58398	-0.26563	0.09570	0.00000
	8 to 10	i=17	0.00000	0.00000	0.37695	4.30664	7.07031	0.81641	2.86914	1.19336	0.69922	-1.23242	0.00000
	10 to 12	i=18	0.00000	0.00000	0.07617	1.46875	3.49219	3.16992	-0.84180	3.81250	-0.50781	0.00000	0.00000

APPENDIX 8: EXCITATION VQ SHAPE CODEBOOK

Index	Element 1	Element 2	Element 3	Element 4
0	-0.537476	0.974976	-0.631104	-0.617920
1	1.145142	1.222534	-1.252441	0.616211
2	1.174194	1.399414	0.330933	0.823120
3	2.946899	0.798096	-0.274658	-0.027344
4	-1.704102	0.098755	-0.526001	-0.395508
5	-0.889038	-0.337402	0.784546	0.298462
6	-0.756958	-0.061890	0.558960	-0.907227
7	1.373169	-0.413330	0.690552	-0.794067
8	-0.573364	-0.463745	-0.606934	-0.623535
9	1.058716	-0.566040	-1.677246	0.752563
10	0.432617	0.441895	-0.630493	-1.445801
11	2.542358	0.207031	-1.611450	0.313354
12	-2.572266	-2.758423	-0.499390	-0.020142
13	0.432251	-2.303711	-2.016479	0.228638
14	0.701538	-1.355591	-0.861572	-0.243042
15	0.857056	-1.842285	-0.006348	1.216919
16	-1.474365	1.636108	-0.683838	0.362915
17	-0.361694	0.711914	-0.136353	1.619873
18	0.407104	1.661255	0.566406	-0.559937
19	1.670288	1.159668	1.760254	0.524780
20	-1.860596	0.592285	1.213379	0.719482
21	-0.845703	0.081421	2.197754	1.654785
22	0.425415	0.641357	1.210205	-1.444580
23	0.208374	0.481567	1.808472	0.685913
24	-1.022583	0.425781	-0.168945	-1.642700
25	0.502075	-0.491455	-0.296631	-0.068359
26	0.270630	0.005981	0.257813	-0.466309
27	2.266357	-1.128540	-0.399414	0.438477
28	-1.876343	-0.895142	-0.012207	0.886841
29	-0.389771	-1.818604	1.185791	0.913452
30	-0.040771	-1.141968	0.364258	-0.283691
31	0.448242	-0.755127	1.767578	-0.691406