## 5.3 MDCT Coding Mode

### 5.3.1 General description

The decision algorithm for selecting the MDCT Coding mode, instead of CELP, is described in subclause 5.1.16. In general, the MDCT coding mode is selected whenever a signal is not speech or the bit rate for encoding is high enough for the transform coder. It always operates on full input signal independent of sample rate and selected audio bandwidth.

The EVS codec supports two major MDCT coding modes, the MDCT based TCX (TCX) and the High Quality MDCT coder (HQ). Both operate with different configurations depending on the bit rate for encoding. For some operating points, the MDCT coders are switched for optimized efficiency. The two MDCT selector algorithms are described in subclauses 5.1.14.2 and 5.1.14.3. The following table describes the used mode configurations depending on bit rate and bandwidth.

Table 76: Overview of supported MDCT coding modes

|  |  |  |  |
| --- | --- | --- | --- |
| Bitrate  (kbps) | Bandwidth | MDCT Classifier | MDCT mode |
| 7.2, 8 | NB |  | HQ low-rate |
| 9.6 | NB, WB,SWB |  | TCX low-rate |
| 13.2 | NB, WB, SWB | MDCT Selector | Switching HQ low-rate and TCX mid-rate |
| 16.4 | NB, WB, SWB FB |
| 24.4 | NB, WB |  | TCX mid-rate |
| 32 | WB |  | HQ high-rate |
| 24.4 | SWB,FB | MDCT Selector II | Switching HQ high-rate and TCX mid-rate |
| 32 | SWB,FB |
| 48 | WB,SWB,FB |  | TCX high-rate |
| 64 | WB,SWB,FB |  | HQ high-rate |
| 96,128 | WB,SWB,FB |  | TCX high-rate |

The next subclauses describe the common time-to-frequency transformation for both major MDCT modes. In the following the TCX and the HQ coder are described including their modules and configurations.

### 5.3.2 Time-to-frequency transformations

#### 5.3.2.1 Transform sizes and MDCT configurations

The MDCT encoder operates with a frame of length  at the input sampling frequency .

#### 5.3.2.2 Long block transformation (ALDO window)

The window used in long block transformation is an asymmetrical low delay optimized (ALDO) window. This ALDO window is stored in ROM in two versions, one at 48 kHz and another at 25.6 kHz respectively; the ALDO window at other input sampling frequencies (namely 8, 12.8, 16, 32 kHz) is obtained by on-the-fly decimation in the folding process described in subclause 5.3.2.2.1.

The ALDO window has a time support of 40 ms and its definition is the same at 48 and 25.6 kHz. To simplify notations the sampling frequency (48 or 25.6 kHz) associated with the ALDO window is not included in the following text. This window,  of length , is structured in 4 segments:

 (877)

where is the frame length (20 ms) and  is the length of the last segment with a weight of 0 (5.625 ms). The different segments consist of an increasing segment , a constant segment  with a weight of 1, a decreasing segment , a constant segment  with a weight of 0, as illustrated in figure 53. Note that in practice the parts corresponding to weights of 1 and 0 are not stored explicitly.

Figure 53 is also useful to illustrate the time alignment of input signal frames at the MDCT encoder. As the fourth segment of the ALDO window has by definition a weight of 0, the frame of new input samples, , is time-aligned in such a way that its end coincides with the end of the third ALDO window segment ; this alignment allows to save  samples of lookahead (5.625 ms); the previous frame of input signal, , is partially weighted by the first ALDO window segment, .



Figure 53: ALDO window

The ALDO window was obtained at 48 and 25.6 kHz by a two-step process: first an initial window  was obtained, then this initial window  was regularized in order to ensure perfect reconstruction. The initial window  is defined in 4 segments with an increasing segment , a constant segment  with a weight of 1 identical to , a decreasing segment , a constant segment  with a weight of 0 identical to . The ALDO window is therefore defined by spelling out the initial window and the regularization term .

***ALDO window at 48 kHz***

At 48 kHz, the 20 ms frame has  samples. The first segment  of the ALDO window is given by

 (878)

where the first segment of the initial window is

 (879)

and the regularization factor is

, (880)

 is the length of the first segment (690 samples at 48 kHz), is a constant. The second segment of the ALDO window is . The third segment  of the ALDO window is given by

 (881)

where the third segment of the initial window is

 (882)

and the regularization factor is

, (883)

 is the length of the first segment (420 samples at 48 kHz), is a constant. The fourth segment of the ALDO window is  (270 sample at 48 kHz).

***ALDO window at 25.6 kHz***

The ALDO window at 25.6 kHz is defined as the ALDO window at 48 kHz, except the following parameters are used:

* 
* 
* 
* 

It follows that  and .

***ALDO window at other sampling frequencies***

The ALDO window at 48 kHz is used to derive on the fly the ALDO window at 8, 16, and 32 kHz. Similarly, the ALDO window at 25.6 kHz is used to derive on the fly the ALDO window at 12.8 kHz. Details on this on-the-fly decimation are provided in subclause 5.3.2.2.1.

##### 5.3.2.2.1 Folding and on-the-fly window decimation

The folding operations and window decimation (when applicable) are combined in the same process. To achieve perfect reconstruction ALDO windows the 48 and 25.6 kHz are irregularly decimated. The decimation consists in selecting a subset of coefficients from the initial window (48 or 25.6 kHz) at specific indices  to preserve perfect reconstruction; this decimation is combined with the folding process for efficiency.

The folding process is illustrated in figure 54. The MDCT folding is performed by dividing the 40 ms time support of the ALDO window in 4 sections delimited by dotted vertical lines.

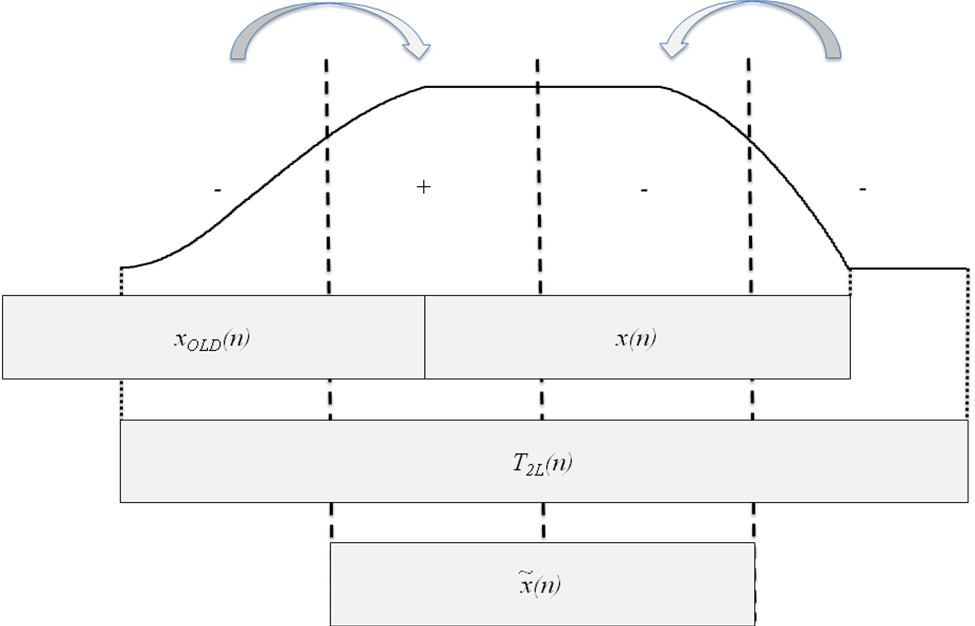


Figure 54: Folding with ALDO window.

For each frame of new input samples, a block  of length  is folded into a block , . The ALDO window at 48 or 25.6 kHz is defined at a sampling frequency corresponding to two frames of length at 48 or 25.6 kHz () . The ratio between  and  is called the decimation factor.

The folded frame is given for  by:

, (884)

, (885)

where the ratio  is the decimation factor and  is an offset that depends on the decimation factor. The ALDO window used for decimation, decimation factor and offset are given in table 77.

Table 77: ALDO window used for decimation, decimation factor and offset parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| MDCT core sampling frequency (kHz) | ALDO window used for decimation (kHz) | Decimation factor | Offset |
| 8 | 48 | 6 | 2 |
| 12.8 | 25.6 | 2 | 0 |
| 16 | 48 | 3 | 1 |
| 25.6 | 25.6 | 1 | 0 |
| 32 | 48 | 3 | 1 |
| 48 | 48 | 1 | 0 |

The 32 kHz sampling frequency is a special case, as the ratio between 48 and 32 kHz is not integer. For this 32 kHz case, in order to achieve perfect reconstruction, the ALDO window  decimated from 48 to 16 kHz is applied to samples with even indices, the samples with odd indices are weighted by a complementary window . For the 32 kHz case, the folded frame is given for  by:

, (886)

, (887)

, (888)

, (889)

where  is the frame length at 16kHz,  is the length of MDCT core frame at 32kHz,  is the decimation factor and  is the offset.

The complementary window of size is stored in ROM. To obtain the window , the ALDO window at 48 kHz is decimated by 3 (with an offset of 1), the resulting decimated window at 16 kHz is considered as if it was obtained from a 32 kHz window decimated by 2 (without any offset) so that half of the samples of the 32 kHz window is known; the other half is obtained doing a linear interpolation between known samples. The combination of the 16 kHz window obtained by decimation and is such that perfect reconstruction is ensured for the overall 32 kHz window.

##### 5.3.2.2.2 eDCT

The ,  are transformed to frequency domain by an eDCT, which is built upon a discrete cosine transform type IV (DCTIV) but the eDCT requires less storage and has lower complexity.

The original L-point DCTIV formula is:

 (890)

where  is the windowed input signal of the current audio frame and is the -th DCT spectral component. This formula can be rewritten as:

 (891)

where the values  are given by

 (892)

and ****, , and .

Hence, the eDCT is computed using the following steps which lead to less storage and lower complexity:

1) Pre-processing

Apply the twiddle factors to the time domain data , so as to obtain twiddled signal :

 (893)

Pre-rotate the twiddled signal  by using a symmetric rotation factor:

, 

where  in the rotation factor may also be expressed in the following form:

 (894)

which satisfies conditions of ,  and , , and therefore, in the implementation, only one data table of cosine values covering L/2-points,  needs to be stored, and  is a constant..

2) Fast Fourier Transform

Perform a Fast Fourier Transform (FFT) of L/2 points on the pre-rotated data .

3) Perform an in-place fixed rotate compensation

The FFT data ,  of  is rotated in-place by multiplying with a fixed rotate compensation factor :

 (895)

4) Post-processing

The data ,  is finally post-rotation processed with a symmetric rotation factor:

, ,

where  in the rotation factor may also be expressed in the following form:

 (896)

In the specific implementation, only one cosine data table of L/2- values needs to be stored i.e. ,  which is the same as the pre-processing, and  is a constant.

5) Obtain frequency domain data

Then the real parts of the post- rotated data are expressed as, , which represent the odd number frequency bins of the frequency domain data; and the frequency reversed imaginary parts of the post-rotated data are expressed as , , which represent the even number frequency bins of the frequency domain data.

 (897)

where , .

#### 5.3.2.3 Transient location dependent overlap and transform length

Beside ALDO window, 3 more overlap shapes are used for windowing:

* FULL: 8.75 milliseconds overlap described in subclause 5.3.2.5, used for transition from transform with long ALDO window to short transform
* HALF: 3.75 milliseconds symmetric sine overlap
* MINIMAL: 1.25 milliseconds symmetric sine overlap

For long MDCT based TCX (TCX20) transformation, that is not after ACELP, 9 combinations of one overlap shape (ALDO, HALF, MINIMAL) on the left side of the window and another overlap shape (ALDO, HALF, MINIMAL) on the right side of the window is possible. For HQ MDCT 4 combinations of one overlap shape (ALDO, FULL, HALF, MINIMAL) on the left side of the window and ALDO on the right side of the window is possible.

For short MDCT based TCX transformation (TCX10 or TCX5), 7 of the 9 possible combinations of one symmetric overlap shape (FULL, HALF, MINIMAL) on the left side of the window and another symmetric overlap shape (FULL, HALF, MINIMAL) on the right side of the window are used (see table 80).

The overlap length and the transform length of the TCX are dependent on the existence of a transient and its location and are chosen so that a transient is mostly contained in only one window as shown in table 78.

Table 78: Coding of the overlap and the transform length based on the transient position

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Attack index | Overlap with the first window of the following frame | Short/Long Transform decision (binary coded)  0 – Long, 1 - Short | Binary code for the overlap width | Overlap code |
| none | ALDO | 0 | 0 | 00 |
| -2 | FULL | 1 | 0 | 10 |
| -1 | FULL | 1 | 0 | 10 |
| 0 | FULL | 1 | 0 | 10 |
| 1 | FULL | 1 | 0 | 10 |
| 2 | MINIMAL | 1 | 10 | 110 |
| 3 | HALF | 1 | 11 | 111 |
| 4 | HALF | 1 | 11 | 111 |
| 5 | MINIMAL | 1 | 10 | 110 |
| 6 | MINIMAL | 0 | 10 | 010 |
| 7 | HALF | 0 | 11 | 011 |

If the transient detector described in the subclause 5.1.8 does not detect a transient, but there is increase of energy at high frequencies in the current frame relative to the previous frame ( is the high frequency energy for the current frame and  is the high frequency energy for the previous frame, as defined in subclause 5.1.2.2):

 (898)

then the short transform is chosen and overlap is dependent on the attack index set in the transient detector or half overlap is chosen if the transient detector didn’t set attack index. The transient detector described in the subclause 5.1.8 basically returns the index of the last attack with the restriction that if there are multiple transients then MINIMAL overlap is preferred over HALF overlap which is preferred over FULL overlap. If an attack at position 2 or 6 is not strong enough then HALF overlap is chosen instead of the MINIMAL overlap.

For bitrates below 48 kbps the short transform length is not allowed and thus the attack index lower than 6 would trigger ALDO window and the overlap code 00.

If the previous frame was coded using ACELP a specific configuration is used as described in subclause 5.4.2.2.

Depending on the right overlap (current overlap code) and the left overlap (previous overlap code) in the current frame the configuration of the transform lengths in the current frame is chosen as presented in table 79.

Table 79: Transform lengths decision table

|  |  |  |  |
| --- | --- | --- | --- |
| Previous overlap code  Current overlap code | 00/10 | 111/011 | 110/010 |
| 00 | TCX20 | TCX20 | TCX20 |
| 10 | 2xTCX5,  TCX10 | 2xTCX5,  TCX10 | 2xTCX5,  TCX10 |
| 111 | 2xTCX5,  TCX10 | 4xTCX5 | 4xTCX5 |
| 110 | 2xTCX5,  TCX10 | 4xTCX5 | 4xTCX5 |
| 010 | TCX20 | TCX20 | TCX20 |
| 011 | TCX20 | TCX20 | TCX20 |

If the current frame is classified as the transition frame (current overlap code is 10, 111 or 110) the detailed configuration of the transform and overlap lengths is presented in table 80.

Table 80: Transform and overlap length configuration for transient frame

|  |  |  |  |
| --- | --- | --- | --- |
| **Prev. code**  Curr. code | 00/10 | 111/011 | 110/010 |
| 10 | FULL,TCX5,MIN.,TCX5,  MIN.,TCX10,FULL | HALF,TCX5,MIN.,TCX5,  HALF,TCX10,FULL | MIN.,TCX5,MIN.,TCX5,  MIN.,TCX10,FULL |
| 111 | FULL,TCX10,HALF,TCX5,  MIN.,TCX5,HALF | HALF,TCX5,MIN.,TCX5,HALF,  TCX5,MIN.,TCX5,HALF | MIN.,TCX5,MIN.,TCX5,MIN.,TCX5,MIN.,TCX5,HALF |
| 110 | FULL,TCX10,MIN.,TCX5,  MIN.,TCX5,MIN. | HALF,TCX5,MIN.,TCX5,MIN.,TCX5,MIN.,TCX5,MIN. | MIN.,TCX5,MIN.,TCX5,MIN.,TCX5,MIN.,TCX5,MIN. |

In table 79 and table 80, the value 00/10 for previous overlap code indicates that the same configuration is used if the previous overlap code is 00 as well as when it is 10. The meaning of values 111/011 and 110/010 follows the same logic.

As example “HALF, TCX5, MIN., TCX5, HALF., TCX10, FULL” from Table 80 indicates that first TCX5 has HALF overlap on the left side and MINIMAL overlap on the right side, followed by second TCX5 with MINIMAL overlap on the left side and HALF overlap on the right side, followed by TCX10 with HALF overlap on the left side and FULL overlap on the right side. The TCX10 with HALF overlap on the left side and FULL overlap on the right is one example of the window described in subclause 5.3.2.5.2.

#### 5.3.2.4 Short block transformation

5.3.2.4.1 Short window transform in TDA domain

5.3.2.4.1.1 Transient detection



Figure 55: Transient detection algorithm

The block diagram of the transient detection algorithm is shown in figure 55. The input signalis first high-pass filtered; the high-pass filter serves as a precaution against undesired low-frequency components. A first order IIR filter is used, and it is given by:

 (899)

The same filter is used for all sampling frequencies.

The output of the high-pass filter  is obtained according to:

 (900)

wheredenotes the 20-ms frame length. The high-pass filtered signal is sectioned into four sub-frames, each corresponding to 5 ms, samples each.

The energy of each sub-frame,, is computed according to:

 (901)

The signal's long-term energy corresponding to each sub-frame,, is updated according to the following equation:

 (902)

In the above equation, the forgetting factor  is set to 0.25, and the convention that from the previous frame is used. The high-pass filter,, states as well as are saved for processing in the next frame.

During switching, when core has changed or extension layer has changed, then is initialized with the energy of.

For each sub-frame , a comparison between the short-term energy  and the long-term energy is performed. A transient is detected whenever the energy ratio is above a certain threshold. Formally, a transient is detected whenever:

 (903)

whereis the energy ratio threshold and is set according to table 81.

Table 81: Threshold table for transient detector

|  |  |  |
| --- | --- | --- |
| Extension layer | Condition | Threshold |
| Generic Mode | Inactive | 10 |
| Not Inactive | 13.5 |
| Otherwise | SWB, |
| Otherwise | 6 |

For the first coded frame with extension layer using the Generic mode then the transient detection, after Equation 903, is set to 0.

In general, the time-frequency transform is applied on a 40-ms frame; therefore, a transient will affect two consecutive frames. To overcome this, a hangover for detected transient is used. A transient detected at a certain frame will also trigger a transient at the next frame.

The output of the transient detector is a flag, denoted *IsTransient*. The flag is set to the logical value *TRUE* if a transient is detected, or *FALSE* otherwise.

If there is a change in extension layer, then the hangover and filter memories are reset before the start of the transient detector.



Figure 56: Transient detection algorithm for NB

The block diagram of the transient detection algorithm for NB is shown in figure 56. Most of algorithm is same as that of WB and SWB. In the *Compute block energy* block, the energy of each sub-frame for ,, is computed according to:

 (904)

where the *EHP(0)* is the energy of the 4th sub-frame in the previous frame.

The *Transient information refinement* block performs an additional verification and checks whether the current frame that has been determined as a transient frame is truly a transient frame. This is to prevent a transient determination error occurring due to the high-pass filtering block removing energy in a low frequency. The operation of the *Transient information refinement* block is described for the case where the current frame is detected to be transient.



Figure 57: Transient information refinement for NB

The position which is detected as a transient is one of 4 sub-frames in frame *n*. The number of blocks included in the region L and the number of blocks included in the second region H will vary depending which of these 4 sub-frames is detected as the transient, as represented in the figure 57.

First, the ratio of the average of the short-term energy in the region H (E\_high) to the average of the short-term energy in the region L (E\_low) is calculated. Region H includes the block which was previously detected to be transient and blocks existing thereafter. Region L includes the blocks prior to region H.

Next, the ratio of the short-term energy of the frame *n* before the high pass filtering (E\_in) to the short-term energy of the frame *n* after the high pass filtering (E\_out) is calculated.

If these ratios meet the following conditions, frame *n* is determined to the normal frame instead of a transient frame.

if( ((E\_high/E\_low<2.0f)&&(E\_high/E\_low>0.7f)) && ((E\_in/E\_out)>Thres) )

IsTransient = 0;

5.3.2.4.1.2 Short window transform

When the flag IsTransient is set then the transform is switched from a long transform to a series of short transforms as is depicted in figure 58. This subclause describes the short transform.



Figure 58: Adaptive time-frequency transform

Windowing and time-domain aliasing are done as described in subclause 5.3.2.2. A reordering of the time-domain aliased signal is performed. In order to keep the temporal coherence of the input signal, the output of the time‑domain aliasing operation needs to be reordered before further processing. The ordering operation is necessary, without ordering the basis functions of the resulting filter-bank will have an incoherent time and frequency responses. The reordering is done according to equation 905, and consists of shuffling the upper and lower half of the TDA output signal .

 (905)

This reordering is only conceptual and in reality no computations are involved. Higher time resolution is obtained by zero-padding the signal and dividing the resulting signal into four overlapped equal length sub-frames. The amount of zero-padding is equal to on each side of the signal. The segments are 50% overlapped and each segment has a length equal to. The two inner segments are post-windowed using a sine window of length. The windows for outer segments are constructed using half a sine window.

This operation is depicted in figure 59; each resulting post-windowed segment is further processed by applying the modified discrete cosine transform (MDCT) (i.e., time aliasing + DCTIV). The output of the MDCT for each segment represents the signal spectrum at different time instants, thus, in case of transients, a higher time resolution is used.

The length of the output of each of the four MDCTs is half the length of the input segment, i.e.,, therefore this operation does not add additional redundancy.



Figure 59: Higher time resolution, division into four segments

5.3.2.4.1.3 Interleaving

After the short transform described in subclause 5.3.2.4.1.2, then the coefficients of the equivalent four 5-ms transforms are interleaved. Table 82 shows the band lengths used for interleaving. Coefficients of interleave band length are read from the first transform, , then the second transform, , and so on until the fourth transform. This is then repeated for all interleave bands. The interleaved bands are written to the vector .

Table 82: Band widths for transient mode interleaving

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Band | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| WB | 16 | 16 | 16 | 16 | 16 |  |  |  |  |
| SWB | 16 | 16 | 16 | 16 | 24 | 24 | 24 | 24 |  |
| FB | 16 | 16 | 16 | 16 | 24 | 24 | 24 | 32 | 32 |

##### 5.3.2.4.2 Short window transform for MDCT based TCX

After choosing the transform and overlap length configuration for transient frame as described in subclause 5.3.2.3 each sub-frame (TCX5 or TCX10) is windowed and transformed using the MDCT, which is implemented using TDA and DCTIV.

After the TNS described in subclause 5.3.3.2.2, the MDCT bins of 2 TCX5 sub-frames are interleaved:

 (906)

#### 5.3.2.5 Special window transitions

The overlap mode FULL is used for transitions between long ALDO windows and short symmetric windows (HALF, MINIMAL) for short block configurations (TCX10, TCX5) described in subclause 5.3.2.3. The transition window is derived from the ALDO and sine window overlaps.

##### 5.3.2.5.1 ALDO to short transition

The left overlap of the transition window has a length of 8.75ms (FULL overlap) and is derived from the ALDO window. The left slope of the ALDO analysis window (long slope) is first shortened to 8.75ms by removing the first 5.625ms. Then the first 1.25ms of the remaining slope are multiplied with the 1.25ms MINIMAL overlap slope to smooth the edge. The resulting 8.75ms slope is depicted in figure 60.

For the right part of the transition window HALF or MINIMAL overlap is used.

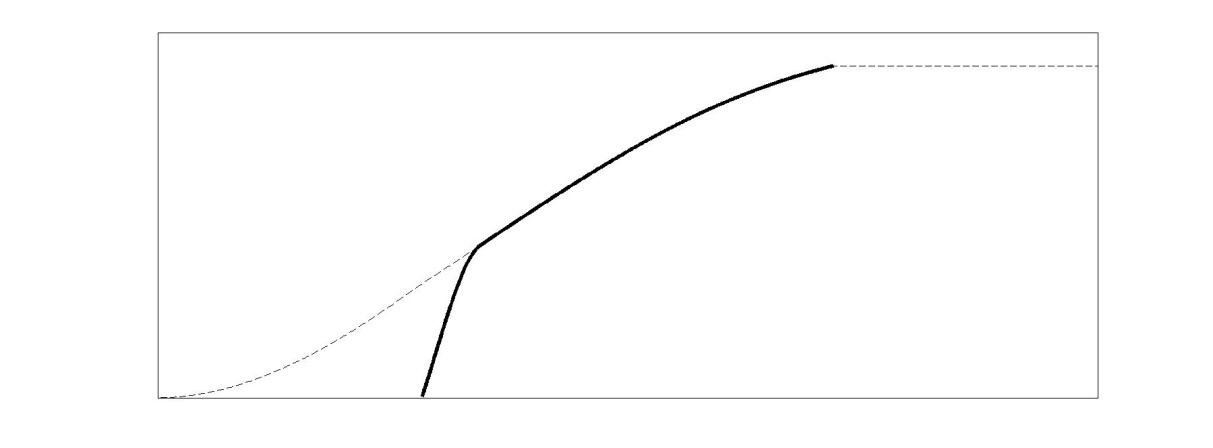


Figure 60: Window shape transition ADLO to Short

##### 5.3.2.5.2 Short to ALDO transition

HALF or MINIMAL overlap is used for the left part of the transition window.

For the right part of the transition window the right slope of the ALDO analysis window (short slope) is used, which has a length of 8.75ms (see figure 61).

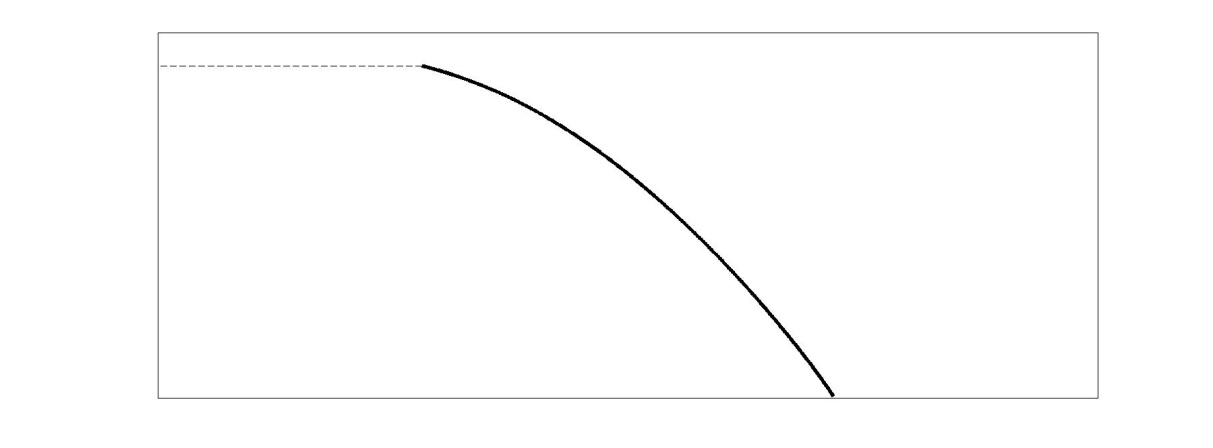


Figure 61: Window shape transition Short to ALDO

#### Modified Discrete Sine Transform

The MDST is computed in the same way as the MDCT defined in the previous subclauses, with two exceptions:

* The TDAC equation is described by:

 (907)

* The DCTIV given by equation (890) is replaced by a DSTIV:

 (908)

The MDST transformation always follows the MDCT parameters such as transform size or windowing.

### 5.3.3 MDCT based TCX

#### 5.3.3.1 General description

##### 5.3.3.1.1 High level overview

The MDCT based TCX (TCX) mode codes the MDCT spectrum using an LPC scheme for noise shaping. The LPC parameters are estimated in time domain and applied in the MDCT domain. The basic mode consists of a 20ms MDCT transformation (TCX20). For higher rates, smaller transform sizes for 10ms (TCX10) and 5ms (TCX5) are supported as well. The TCX mode provides several coding tools such as adaptive low frequency emphasis, temporal noise shaping, intelligent gap filling or noise filling to improve coding efficiency. The spectral data is finally quantized by a uniform quantizer with dead-zone and noiseless codec by an arithmetic coder module with several modes.

For the MDCT based TCX, the total length of the MDCT spectrum  depends on the input sampling rate, the TCX mode and the coding mode of the previous frame. Some of the coding tools used in TCX only work on the lower MDCT bins corresponding to the CELP frequency range, as LP analysis is performed on the CELP sampling rate. This number of bins is designated by. The maximum number of bins  encoded by TCX is determined by the coding bandwidth. If  exceeds, the remaining bins are filled with zeroes.

The following table defines,  and:

Table 83: Overview frame sizes of MDCT-based TCX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TCX mode | | TCX20 | | TCX10/TCX5 |
| Previous coding mode | | MDCT | CELP |
|  | |  |  |  |
|  | |  |  |  |
|  | NB | 160 | 200 | 80 |
| WB | 320 | 400 | 160 |
| SWB | 640 | 800 | 320 |
| FB | 960 | 1200 | 480 |

##### 5.3.3.1.2 Rate dependent configuration

The MDCT-based TCX contains different setups depending on the bit rate for encoding. In general, the setup consists of three configurations: Low-rate, mid-rate and high-rate configuration. The following table lists the main tools used in the various configurations.

Table 84: Overview configurations MDCT-based TCX

|  |  |  |
| --- | --- | --- |
| Bitrate  [kbps] | Bandwidth | Specific configurations |
| 9.6  (Low-rate) | NB,WB,SWB | low-rate LPC, envelope based arithmetic coder, ALFE 1, TCX20 |
| 13.2 - 32  (Mid-rate) | NB, WB, SWB,FB | mid-rate LPC, context based arithmetic coder, ALFE 2, TCX20 |
| 48 – 128 (High-rate) | WB, SWB,FB | high-rate LPC, context based arithmetic coder, ALFE 1, TCX20/10/5 |

One additional bit rate dependent tool is temporal noise shaping (TNS). TNS is activated only for bit rate 24.4 kbps and higher.

#### 5.3.3.2 General encoding procedure

##### 5.3.3.2.1 LPC parameter calculation

5.3.3.2.1.1 Low rate LPC

MDCT based TCX relies on smoothed LPC spectral envelope. Analysis methods are identical to that of ACELP. Quantization of LSF is also common to ACELP except for 9.6 kbps (NB/WB/SWB). Quantization of weighted LSF and conversion of LSF used for 9.6 kbps (NB/WB/SWB) are described in the following subclauses.

5.3.3.2.1.1.1 Quantization of weighted LSF

For MDCT based TCX at 9.6 kbps (NB/WB), envelope based arithmetic coding and low rate quantization of weighted LSF are used. After normal LPC analysis and interpolation in LSP domain, perceptual weighting is applied to the prediction coefficients to get weighted prediction coefficients  as

. (909)

Weighted prediction coefficients  are converted to LSF vector and the LSF vector is quantized with two stage or three stage VQ. These VQ codebooks are used in two ways.

First and primary use of the quantization is intended to represent the weighted LPC envelope to shape the MDCT coefficients with MA inter frame prediction. Secondary use is to tell the estimate of envelope to the arithmetic coding without depending on MA prediction.

In case of primary encoding, input weighted LSF vector is subtracted by the mean vector  and the MA predicted contribution vector  to generate the input of two stage VQ , .

. (910)

For the first stage VQ, 5 bits for 16 samples are allocated. For the second stage VQ, 4 bits are assigned to the lower 6 LSF parameters and 4 bits for the higher 10 parameters. A quantization criterion is to minimize the weighted distortion between input, and output of quantized LSF vectors, . The reconstructed LSF vector after adding mean vector and the MA predicted contribution vector, can be converted to the weighted envelope representing and used for noise shaping of MDCT domain transform coefficients.

In the secondary encoding and decoding, the same LSF vector from the VQ table at the primary encoding,  is retrieved and mean vector is added to reconstruct weighted LSF vector without MA prediction . This can inform the shape of envelope for the envelope base arithmetic coding even when the decoder cannot get the information from the previous frame.

Envelope shape could be distorted due to the lack of MA prediction and a third stage VQ is used to compensate the distortion only when necessary. Compensation is preferable when large spectral envelope or large distortion in spectral envelope is expected. In order to determine the necessity of the third stage VQ, the first and the second lower values of , namely,  and  are checked. Ifor have smaller values than the threshold, namely expected envelope values and distortion are larger than threshold, the third VQ with 2 bits is applied to  and  for the purpose of corrections. The criteria of the selection of the retrieved vector at the third stage VQ,  is to minimize the weighted distortion between input and a final reconstructed LSF vector without MA prediction . If both and values are equal to or larger than threshold, the third VQ is skipped and is used as a final reconstruction LSF vector.

Reconstructed LSF vector is corresponding to the weighted envelopewithout depending on MA prediction. In envelope based arithmetic coding, unweighted LSF vector without depending on MA prediction is also necessary to estimate the MDCT envelope. This can be obtained by low-complex direct matrix conversion described in the next subclause.

5.3.3.2.1.1.2 Direct conversion of LSF

For the interpolation of LSF to the LSF in the possible ACELP at the next frame, the reconstructed LSF vector with MA prediction needs to be converted to unweighted domain LSF vector . Similarly, reconstructed LSF vector without MA prediction needs to be converted to get unweighted domain LSF to assist the envelope base arithmetic coding.

In order to get unweighted LSF corresponding to, weighted LSF  is converted by using matrixand equally spaced LSF as follows.

 (911)

 (912)

 has non-zero elements only in the diagonal position and the adjacent samples as follows and the non-zero elements are shown in table 85.

 (913)

Table 85: Non-zero elements of conversion matrices

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | from  0.92 to 1.0  (12.8 kHz sample) | | | from  0.94 to 1.0  (16 kHz sample) | | |
|  |  |  |  |  |  |  |
| 1 | - | 1.19764 | -0.59173 | - | 0.78925 | -0.38537 |
| 2 | -0.91173 | 1.79182 | -0.80921 | -0.57154 | 1.19486 | -0.54136 |
| 3 | -0.51779 | 1.44703 | -0.81871 | -0.33642 | 0.99096 | -0.56792 |
| 4 | -0.44862 | 1.36777 | -0.75103 | -0.29856 | 0.93785 | -0.51255 |
| 5 | -0.4515 | 1.30719 | -0.7422 | -0.29716 | 0.89303 | -0.50509 |
| 6 | -0.43157 | 1.21326 | -0.68538 | -0.28264 | 0.81530 | -0.46020 |
| 7 | -0.43606 | 1.21317 | -0.69131 | -0.27926 | 0.80997 | -0.46378 |
| 8 | -0.392 | 1.04941 | -0.58674 | -0.25334 | 0.69596 | -0.38969 |
| 9 | -0.45208 | 1.10009 | -0.59175 | -0.29656 | 0.72916 | -0.38888 |
| 10 | -0.42553 | 0.99725 | -0.49992 | -0.27488 | 0.65949 | -0.32999 |
| 11 | -0.50168 | 1.07575 | -0.51401 | -0.32630 | 0.70913 | -0.33659 |
| 12 | -0.498 | 1.06563 | -0.50592 | -0.33069 | 0.70668 | -0.33105 |
| 13 | -0.53101 | 1.16372 | -0.58033 | -0.35437 | 0.77582 | -0.38003 |
| 14 | -0.48744 | 1.07596 | -0.52531 | -0.31771 | 0.70752 | -0.34216 |
| 15 | -0.51899 | 1.04998 | -0.49495 | -0.35066 | 0.70177 | -0.31664 |
| 16 | -0.4773 | 0.90959 | - | -0.33404 | 0.62528 | - |

5.3.3.2.1.2 Mid-rate LPC

For mid-rate bitrates (between 13.2 and 32 kbps) the LP analysis method is the same as for ACELP. The quantizer used is also the same but operated in AUDIO mode only (see subclause 5.2.2.1.2). However the interpolation of quantized LSF coefficients is different. Here each interpolated LSF coefficient  is a weighted sum of the quantized LSF coefficients referring to the current and previous frame:

 (914)

These interpolated coefficients are converted back to LPC domain in the same way as done for ACELP frames.

5.3.3.2.1.3 High-rate LPC

This subclause describes the LPC analysis and quantization scheme for high-rate bitrates (48 kbps and higher). The description in the following subclauses refers to the quantization of either a single set of LPC parameters for the whole frame or two sets if the frame is divided into sub-frames. As described in subclause 5.3.2.3 a frame can be subdivided into TCX5 or TCX10 sub-frames. In this case the LPC quantizer however treats two adjacent TCX5 sub-frames as a single TCX10 sub-frame.

The LP coefficients are obtained by calculating autocorrelation on a windowed and pre-emphasized time domain signal with  set to 0.9 for SWB and to 0.72 for WB and the input signal to the pre-emphasis filter is  for WB,  for 48 kbps SWB and for 96 kbps and 128 kbps SWB (see subclause 5.1.4). The window used here is the window used for the TCX MDCT transformation (described in subclause 5.3.2.3) with the only exception, the ALDO windows will be replaced by FULL overlap windows (see subclause 5.3.2.5). After autocorrelation adaptive lag-windowing (see subclause 5.1.9.3) is applied to the autocorrelation coefficients which are finally converted to LP coefficients by Levinson recursion. The sample rate used for adaptive lag-windowing is the TCX sample rate (25.6 or 32 kHz) and the information of OL pitch estimation of current frame is used. For quantization the LP coefficients are converted to LSF domain.

The first set of LSF coefficients is always quantized without inter-frame dependency. In case two sets of LSF coefficients are quantized the second set may be quantized with or without dependency to the first set. This dependency (explained below) is signalled with one bit.

5.3.3.2.1.3.1 General Principle

The general principle for the quantization of a given LPC filter is represented in figure 62. A first-stage approximation is computed, and then subtracted from the input LSF vector to produce a residual LSF vector.

A LSF weighting function is derived from the first-stage approximation and applied to the residual LSF vector. The purpose and calculation of the weighting function are described in subclause 5.3.3.2.1.2.3.

The resulting weighted residual LSF vector is finally fed to the algebraic VQ encoder described in subclause 5.3.3.2.1.2.4.

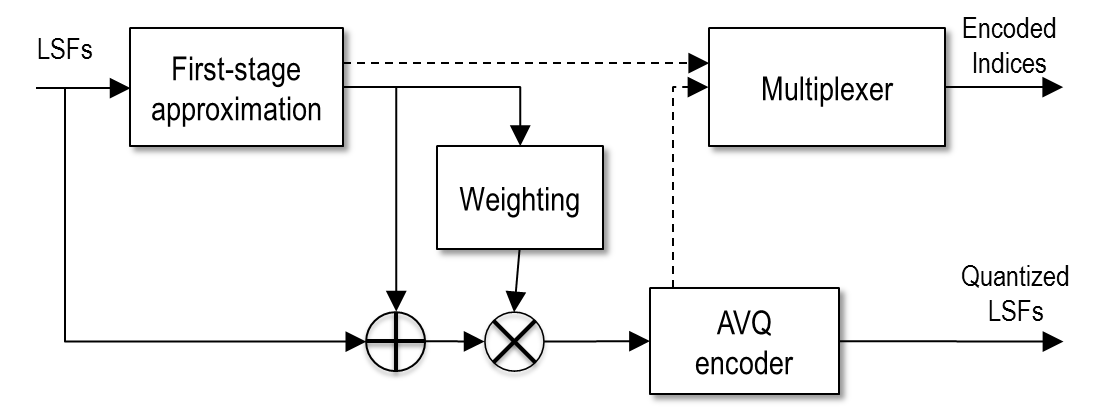


Figure 62: Principle of weighted algebraic LPC quantizer

5.3.3.2.1.3.2 First-stage approximation

The first stage approximation is a 16-dimensional, 8-bits stochastic vector quantizer applied to the input LSF vector. The codebook search uses a weighted Euclidian distance in which each component of the squared difference between the input LSF vector and the codebook entry is multiplied by:

 (915)

with:

 (916)

where LSF is the input LSF vector to be quantized and is the internal sampling frequency of the TCX-based codec (25600 for 48 kbps and 32000 for 96 kbps and above).

The difference between the input LSF vector and the first-stage approximation is called the residual LSF vector.

5.3.3.2.1.3.3 LSF weighting

Initially a weighted sum of the residual LSF vector is compared to a threshold:

 (917)

where  are the first stage approximation of the LSF coefficients and  are the weights according to equation (918) with the factor  being . If the above relation is true, a special codebook is selected and no further quantization done. Otherwise the residual LSF vector is quantized with stochastic vector quantization.

The principle of stochastic vector quantization is to search a codebook of vectors for the nearest neighbour (in terms of Euclidian distance) of the vector to be quantized. When quantizing LPC filters in the LSF (line spectral frequency) domain, a weighted Euclidian distance is generally used, each component of the vector being weighted differently depending on its value and of the value of the other components. The purpose of this weighting is to make the minimization of the Euclidian distance behave as close as possible to a minimization of the spectral distortion.

Unlike a stochastic quantizer, a uniform algebraic vector quantizer does not perform an exhaustive search of the codebook. It is therefore very complex to introduce a weighting function in the distance computation.

The solution used here is to warp the residual LSF vector (i.e. the difference between the input LSF vector and a first stage approximation) using a weighting function computed from the first-stage LSF approximation. By warping we mean applying different weights to the components of the LSF residual vector. Because the first-stage LSF approximation is also available at the decoder, the inverse weighting factors can also be computed at the decoder and the inverse warping can be applied to the quantized residual LSF vector. Warping the residual LSF vector according to a model that minimizes the spectral distortion is especially useful when the quantizer is uniform.

The weights applied to the components of the residual LSF vector are:

 (918)

with

 (919)

where  is the internal sampling frequency of the TCX-based codec (25600 for 48 kbps and 32000 for 96 kbps and above). The factor  is always  for the first set of LSF coefficients.

5.3.3.2.1.3.4 Algebraic vector quantizer

The algebraic VQ used for quantizing the refinement is described in subclause 5.2.3.1.6.9.

5.3.3.2.1.3.5 Quantization of second set of LSF

In case a second set of LSF coefficients needs to be quantized, the quantization scheme for the second set may be dependent on the first set or not. Two approaches are tried and the one with the lower bit consumption will be chosen. The first approach is identical to the quantization of the first set. In the second approach the first stage approximation described in subclause 5.3.3.2.1.3.2 is replaced by the quantized first set of LSF. The factor for calculating the weights is changed to. Finally the bit consumption of both approaches is compared and the one yielding the lower amount of bits is chosen and signalled with one bit.

##### 5.3.3.2.2 Temporal Noise Shaping

Temporal Noise Shaping (TNS) is used to control the temporal shape of the quantization noise within each window of the transform.

If TNS is active in this encoder, up to two filters per MDCT-spectrum will be applied. The steps in the TNS encoding are described below. TNS is always calculated on a per subwindow basis, so in case of an 4 TCX5 window sequence these steps have to be applied once for each of the 4 subwindows.

Table 86: TNS configurations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bitrate  [kbps] | Bandwidth | Number of TNS filters for TCX20 and start and stop frequency | Number of TNS filters for TCX10 and start and stop frequency | Number of TNS filters for TCX5 and start and stop frequency |
| 24.4, 32 | SWB | 1 (600Hz-16000Hz) | TCX10 not used | TCX5 not used |
| 48,96,128 | WB | 1 (600Hz-8000Hz) | 2 (800Hz-4400Hz, 4400-8000Hz) | 1 (800Hz-8000Hz) |
| 48,96,128 | SWB | 2 (600Hz-4500Hz, 4500-16000Hz) | 2 (800Hz-8400Hz, 8400-16000Hz) | 1 (800Hz-16000Hz) |
| 24.4,32 | FB | 1 (600Hz-20000Hz) | TCX10 not used | TCX5 not used |
| 48,96,128 | FB | 2 (600Hz-4500Hz, 4500-20000Hz) | 2 (800Hz-10400Hz, 10400-20000Hz) | 1 (800Hz-20000Hz) |

The number of filters for each configuration and the start and the stop frequency of each filter are given in table 86.

The MDCT bins of 2 TCX5 sub-frames are rearranged prior to TNS filtering:

 (920)

For such rearanged and concatanated 2 TCX5 frames, special TNS configuration with 2 filters is used, which effectively functions as one filter per TCX5. The rearrangement is reverted after the filtering described below, to again have 2 separated TCX5 subwindows.

5.3.3.2.2.1 TNS detection

The spectral coefficients between the start and stop frequency are divided into  equal consecutive portions and for each portion the normalized autocorrelation function is calculated. The values of the autocorrelation function of all portions are then summed up and lag windowed. An exception is the first filter for 48/96/128 kbps SWB from 600 Hz to 4500 Hz which has only one portion ( = 1).

The next step is an LPC calculation using the Levinson-Durbin algorithm (defined in the subclause 5.1.9.4). The filter oder  is limited to 8 and to quarter of the number of the bins that a TNS filter covers.. As a result so called PARCOR or reflection coefficients  (where is defined in subclause 5.1.9.4 and ) and the prediction gain  are available, where  is the residual error energy as defined in subclause 5.1.9.4.

The TNS parcor coefficients will be quantized with a resolution of 4 bits.

The filter order is reduced such that the last PARCOR coefficient is non-zero.

TNS filter will be used only if the prediction gain is greater than a given threshold, which is dependent on the filter and varies between 1.35 and 1.85, or if the average of the squared filter coefficients is greater than a given threshold, which is dependent on the filter and varies between 0.03 and 0.075. For configurations with 2 filters per spectrum, each filter can be independently disabled or enabled.

5.3.3.2.2.2 TNS filtering

The spectral coefficients will be replaced by the spectral coefficients filtered with the TNS filter. In the following text  refers to the TNS filtered or to the non-filtered spectral coefficients, depending on the configuration, where the configurations where TNS filtered spectral coefficients are used are listed in table 86. The filtering is done with the help of a so called lattice filter, no conversion from parcor coefficients to linear prediction coefficients is required.

5.3.3.2.2.3 Coding of the TNS parameters

The TNS filter order and the quantized parcor coefficients are coded using Huffman code. Which Huffman code will be used for the filter order depends on the frame configuration (TCX20/TCX10/TCX5) and on the bandwidth (SWB,WB). Which Huffman code will be used for a parcor coefficient depends on the frame configuration (TCX20/TCX10/TCX5), on the bandwidth (SWB,WB) and on the parcor coefficent’s index.

##### 5.3.3.2.3 LPC shaping in MDCT domain

5.3.3.2.3.1 General Principle

LPC shaping is performed in the MDCT domain by applying gain factors computed from weighted quantized LP filter coefficients to the MDCT spectrum. The input sampling rate , on which the MDCT transform is based, can be higher than the CELP sampling rate , for which LP coefficients are computed. Therefore LPC shaping gains can only be computed for the part of the MDCT spectrum corresponding to the CELP frequency range. For the remaining part of the spectrum (if any) the shaping gain of the highest frequency band is used.

5.3.3.2.3.2 Computation of LPC shaping gains

To compute the 64 LPC shaping gains the weighted LP filter coefficients  are first transformed into the frequency domain using an oddly stacked DFT of length 128:

 (921)

The LPC shaping gains  are then computed as the reciprocal absolute values of :

 (922)

5.3.3.2.3.3 Applying LPC shaping gains to MDCT spectrum

The MDCT coefficients  corresponding to the CELP frequency range are grouped into 64 sub-bands. The coefficients of each sub-band are multiplied by the reciprocal of the corresponding LPC shaping gain to obtain the shaped spectrum . If the number of MDCT bins corresponding to the CELP frequency range  is not a multiple of 64, the width of sub-bands varies by one bin as defined by the following pseudo-code:

, 

if  then

, 

else if  then

, , 

else

, , 



for 

{

if  then



else



for 

{





}

}

The remaining MDCT coefficients above the CELP frequency range (if any) are multiplied by the reciprocal of the last LPC shaping gain:

 (923)

For the configurations 9.6 kbit/s and 13.2 kbit/s SWB, the remaining spectral coefficients above the CELP frequency range are postprocessed:

First, the highest amplitudes of the MDCT spectrum below and above are determined. The search procedure returns the following values:   
a) max\_low\_pre: The maximum MDCT coefficient below , evaluated on the spectrum of absolute values before the application of reciprocal LPC shaping gains  
b) max\_high\_pre: The maximum MDCT coefficient above , evaluated on the spectrum of absolute values before the application of reciprocal LPC shaping gains

max\_low\_pre = 0;  
for(i=0; i<; i++)  
{  
 tmp = fabs();  
 if(tmp > max\_low\_pre)  
 {  
 max\_low\_pre = tmp;  
 }  
}  
  
max\_high\_pre = 0;  
for(i=0; i<; i++)  
{  
 tmp = fabs( ( + i));  
 if(tmp > max\_high\_pre)  
 {  
 max\_high\_pre = tmp;  
 }  
}

Second, a peak-distance metric analyzes the impact of spectral peaks above above  on the arithmetic coder. Thus, the maximum amplitude of the MDCT spectrum below and above  are searched on the MDCT spectrum after the application of reciprocal LPC shaping gains, i.e. in the domain where also the arithmetic coder is applied. In addition to the maximum amplitude, also the distance from  is evaluated. The search procedure returns the following values:  
a) max\_low: The maximum MDCT coefficient below , evaluated on the spectrum of absolute values after the application of reciprocal LPC shaping gains  
b) dist\_low: The distance of max\_low from   
c) max\_high: The maximum MDCT coefficient above , evaluated on the spectrum of absolute values after the application of reciprocal LPC shaping gains  
d) dist\_high: The distance of max\_high from 

max\_low = 0;  
dist\_low = 0;  
for(i=0; i<; i++)  
{  
 tmp = fabs((– 1-i));  
 if(tmp > max\_low)  
 {  
 max\_low = tmp;  
 dist\_low = i;  
 }  
}  
  
max\_high = 0;  
dist\_high = 0;  
for(i=0; i<- ; i++)  
{  
 tmp = fabs(( + i));  
 if(tmp > max\_high)  
 {  
 max\_high = tmp;  
 dist\_high = i;  
 }  
}

Third, the peak-amplitudes in psycho-acoustically similar spectral regions are compared. Thus, the maximum amplitude of the MDCT spectrum below and above  is searched on the MDCT spectrum after the application of reciprocal LPC shaping gains. The maximum amplitude of the MDCT spectrum below  is not searched for the full spectrum, but only starting at . This is to discard the lowest frequencies, which are psycho-acoustically most important and usually have the highest amplitude after the application of reciprocal LPC shaping gains, and to only compare components with a similar psycho-acoustical importance. The search procedure returns the following values:  
a) max\_low2: The maximum MDCT coefficient below , evaluated on the spectrum of absolute values after the application of reciprocal LPC shaping gains starting from   
b) max\_high: The maximum MDCT coefficient above , evaluated on the spectrum of absolute values after the application of reciprocal LPC shaping gains

max\_low2 = 0;  
for(i=; i<; i++)  
{  
 tmp = fabs((i));  
 if(tmp > max\_low2)  
 {  
 max\_low2 = tmp;  
 }  
}  
  
max\_high = 0;  
for(i=0; i<- ; i++)  
{  
 tmp = fabs(( + i));  
 if(tmp > max\_high)  
 {  
 max\_high = tmp;  
 }  
}

Finally, the relation of max\_low, dist\_low, max\_high, dist\_high, max\_low2, max\_high is evaluated and – if applicaple – a correction factor for determined and applied:

if( (16.0 \* max\_low\_pre > max\_high\_pre) &&  
 (4.0 \* dist\_high \* max\_high > dist\_low \* max\_low) &&  
 (max\_high > max\_fac \* max\_low2)  
 )  
{  
 fac = mac\_fac \* max\_low2 / max\_high;  
  
 for(i = ; i< ; i++)  
 {  
 (i) = (i) \* fac;  
 }  
}

where max\_fac is set to 1.5 in case the Envelope based arithmetic coder is used, or max\_fac is set to 3.0 for all other cases.

##### 5.3.3.2.4 Adaptive low frequency emphasis

5.3.3.2.4.1 General Principle

The purpose of the adaptive low-frequency emphasis and de-emphasis (ALFE) processes is to improve the subjective performance of the frequency-domain TCX codec at low frequencies. To this end, the low-frequency MDCT spectral lines are amplified prior to quantization in the encoder, thereby increasing their quantization SNR, and this boosting is undone prior to the inverse MDCT process in the internal and external decoders to prevent amplification artifacts.

There are two different ALFE algorithms which are selected consistently in encoder and decoder based on the choice of arithmetic coding algorithm and bit-rate. ALFE algorithm 1 is used at 9.6 kbps (envelope based arithmetic coder) and at 48 kbps and above (context based arithmetic coder). ALFE algorithm 2 is used from 13.2 up to incl. 32 kbps. In the encoder, the ALFE operates on the spectral lines in vector x[] directly before (algorithm 1) or after (algorithm 2) every MDCT quantization, which runs multiple times inside a rate-loop in case of the context based arithmetic coder (see subclause 5.3.3.2.8.1).

5.3.3.2.4.2 Adaptive emphasis algorithm 1

ALFE algorithm 1 operates based on the LPC frequency-band gains, lpcGains[]. First, the minimum and maximum of the first nine gains – the low-frequency (LF) gains – are found using comparison operations executed within a loop over the gain indices 0 to 8.

Then, if the ratio between the minimum and maximum exceeds a threshold of 1/32, a gradual boosting of the lowest lines in x is performed such that the first line (DC) is amplified by (32 min/max)0.25 and the 33rd line is not amplified:

tmp = 32 \* min

if ((max < tmp) && (max > 0))

{

fac = tmp = pow(tmp / max, 1/128)

for (i = 31; i >= 0; i--)

{ /\* gradual boosting of lowest 32 lines \*/

x[i] \*= fac

fac \*= tmp

}

}

5.3.3.2.4.3 Adaptive emphasis algorithm 2

ALFE algorithm 2, unlike algorithm 1, does not operate based on transmitted LPC gains but is signaled by means of modifications to the quantized low-frequency (LF) MDCT lines. The procedure is divided into five consecutive steps:

* Step 1: first find first magnitude maximum at index i\_max in lower spectral quarter (*k* = 0 … / 4) utilizing invGain = 2/*gTCX* and modifying the maximum: xq[i\_max] += (xq[i\_max] < 0) ? -2 : 2
* Step 2: then compress value range of all x[i] up to i\_max by requantizing all lines at *k* = 0 … i\_max–1 as in the subclause describing the quantization, but utilizing invGain instead of *gTCX* as the global gain factor.
* Step 3: find first magnitude maximum below i\_max (*k* = 0 …/ 4) which is half as high if i\_max > –1 using invGain = 4/*gTCX* and modifying the maximum: xq[i\_max] += (xq[i\_max] < 0) ? -2 : 2
* Step 4: re-compress and quantize all x[i] up to the half-height i\_max found in the previous step, as in step 2
* Step 5: finish and always compress two lines at the latest i\_max found, i.e. at *k* = i\_max+1, i\_max+2, again utilizing invGain = 2/*gTCX* if the initial i\_max found in step 1 is greater than –1, or using invGain = 4/*gTCX* otherwise. All i\_max are initialized to –1. For details please see AdaptLowFreqEmph() in tcx\_utils\_enc.c.

##### 5.3.3.2.5 Spectrum noise measure in power spectrum

For guidance of quantization in the TXC encoding process, a noise measure between 0 (tonal) and 1 (noise-like) is determined for each MDCT spectral line above a specified frequency based on the current transform’s power spectrum. The power spectrum  is computed from the MDCT coefficients and the MDSTcoefficients on the same time-domain signal segment and with the same windowing operation:

 (924)

Each noise measure in  is then calculated as follows. First, if the transform length changed (e.g. after a TCX transition transform following an ACELP frame) or if the previous frame did not use TCX20 coding (e.g. in case a shorter transform length was used in the last frame), all  up to  are reset to zero. The noise measure start line is initialized according to the following table 87.

Table 87: Initialization table of in noise measure

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bitrate (kbps) | 9.6 | 13.2 | 16.4 | 24.4 | 32 | 48 | 96 | 128 |
| bw= NB, WB | 66 | 128 | 200 | 320 | 320 | 320 | 320 | 320 |
| bw=SWB,FB | 44 | 96 | 160 | 320 | 320 | 256 | 640 | 640 |

For ACELP to TCX transitions, is scaled by 1.25. Then, if the noise measure start line is less than , the  at and above  are derived recursively from running sums of power spectral lines:

 (925)

 (926)

Furthermore, every time is given the value zero in the above loop, the variable *lastTone* is set to *k*. The upper 7 lines are treated separately since  cannot be updated any more (, however, is computed as above):

 (927)

The uppermost line at is defined as being noise-like, hence  . Finally, if the above variable *lastTone* (which was initialized to zero) is greater than zero, then . Note that this procedure is only carried out in TCX20, not in other TCX modes ().

##### 5.3.3.2.6 Low pass factor detector

A low pass factor is determined based on the power spectrum for all bitrates below 32.0 kbps. Therefore, the power spectrum  is compared iteratively against a threshold for all , where for regular MDCT windows and for ACELP to MDCT transition windows. The iteration stops as soon as.

The low pass factor determines as , where is the last determined low pass factor. At encoder startup, is set to 1.0. The low pass factor is used to determine the noise filling stop bin (see subclause 5.3.3.2.10.2).

##### 5.3.3.2.7 Uniform quantizer with adaptive dead-zone

For uniform quantization of the MDCT spectrum after or before ALFE (depending on the applied emphasis algorithm, see subclause 5.3.3.2.4.1), the coefficients are first divided by the global gain (see subclause 5.3.3.2.8.1.1), which controls the step-size of quantization. The results are then rounded toward zero with a rounding offset which is adapted for each coefficient based on the coefficient’s magnitude (relative to ) and tonality (as defined by  in subclause 5.3.3.2.5). For high-frequency spectral lines with low tonality and magnitude, a rounding offset of zero is used, whereas for all other spectral lines, an offset of 0.375 is employed. More specifically, the following algorithm is executed.

Starting from the highest coded MDCT coefficient at index , we set  and decrement  by 1 as long as condition  and  evaluates to true. Then downward from the first line at index  where this condition is not met (which is guaranteed since ), rounding toward zero with a rounding offset of 0.375 and limiting of the resulting integer values to the range –32768 to 32767 is performed:

 (928)

with . Finally, all quantized coefficients of  at and above  are set to zero.

##### 5.3.3.2.8 Arithmetic coder

The quantized spectral coefficients are noiselessly coded by an entropy coding and more particularly by an arithmetic coding.

The arithmetic coding uses 14 bits precision probabilities for computing its code. The alphabet probability distribution can be derived in different ways. At low rates, it is derived from the LPC envelope, while at high rates it is derived from the past context. In both cases, a harmonic model can be added for refining the probability model.

The following pseudo-code describes the arithmetic encoding routine, which is used for coding any symbol associated with a probability model. The probability model is represented by a cumulative frequency table *cum\_freq[].* The derivation of the probability model is described in the following subclauses.

/\* global varibles \*/

low

high

bits\_to\_follow

ar\_encode(symbol, cum\_freq[])

{

if (ari\_first\_symbol()) {

low = 0;

high = 65535;

bits\_to\_follow = 0;

}

range = high-low+1;

if (symbol > 0) {

high = low + ((range\*cum\_freq[symbol-1])>>14) - 1;

}

low += ((range\*cum\_freq[symbol-1])>>14) - 1;

for (;;) {

if (high < 32768 ) {

write\_bit(0);

while ( bits\_to\_follow ) {

write\_bit(1);

bits\_to\_follow--;

}

}

else if (low >= 32768 ) {

write\_bit(1)

while ( bits\_to\_follow ) {

write\_bit(0);

bits\_to\_follow--;

}

low -= 32768;

high -= 32768;

}

else if ( (low >= 16384) && (high < 49152) ) {

bits\_to\_follow += 1;

low -= 16384;

high -= 16384;

}

else break;

low += low;

high += high+1;

}

if (ari\_last\_symbol()) /\* flush bits \*/

if ( low < 16384 ) {

write\_bit(0);

while ( bits\_to\_follow > 0) {

write\_bit(1);

bits\_to\_follow--;

}

} else {

write\_bit(1);

while ( bits\_to\_follow > 0) {

write\_bit(0);

bits\_to\_follow--;

}

}

}

}

The helper functions *ari\_first\_symbol()* and *ari\_last\_symbol()* detect the first symbol and the last symbol of the generated codeword respectively.

5.3.3.2.8.1 Context based arithmetic codec

5.3.3.2.8.1.1 Global gain estimator

The estimation of the global gain  for the TCX frame is performed in two iterative steps. The first estimate considers a SNR gain of 6dB per sample per bit from SQ. The second estimate refines the estimate by taking into account the entropy coding.

The energy of each block of 4 coefficients is first computed:

 (929)

A bisection search is performed with a final resolution of 0.125dB:

**Initialization:** Set *fac* = *offset = 12.8* and *target = 0.15(target\_bits – L/16)*

**Iteration:** Do the following block of operations 10times

1- *fac=fac/2*

2- *offset = offset – fac*

2- 

3- *if(ener>target) then offset=offset+fac*

The first estimate of gain is then given by:

 (930)

5.3.3.2.8.1.2 Rate-loop for constant bit rate and global gain

In order to set the best gain within the constraints of , convergence process of and is carried out by using following valuables and constants:

 and  denote weights corresponding to the lower bound the upper bound,

and denote gain corresponding to the lower bound the upper bound, and

 and denote flags indicating and is found, respectively.

 and  are variables with  and .

 and are constants, set as 10 and 0.96.

After the initial estimate of bit consumption by arithmetic coding,  is set 0 when is larger than, while is set as  when  is larger than .

If  is larger than 0, that means is larger than ,

needs to be modified to be larger than the previous one and is set as TRUE,  is set as the previous.  is set as

, (931)

When was set, that means was smaller than , is updated as an interpolated value between upper bound and lower bound. ,

, (932)

Otherwise, that means is FALSE, gain is amplified as

, (933)

with larger amplification ratio when the ratio of (= and  is larger to accelerate to attain .

If  equals to 0, that means is smaller than,

 should be smaller than the previous one and is set as 1,  is set as the previous and  is set as

, (934)

If has been already set, gain is calculated as

, (935)

otherwise, in order to accelerate to lower band gain , gain is reduced as,

, (936)

with larger reduction rates of gain when the ratio of  and  is small.

After above correction of gain, quantization is performed and estimation of  by arithmetic coding is obtained. As a result,  is set 0 when is larger than, and is set as  when it is larger than . If the loop count is less than 4, either lower bound setting process or upper bound setting process is carried out at the next loop depending on the value . If the loop count is 4, the final gain and the quantized MDCT sequence are obtained.

5.3.3.2.8.1.3 Probability model derivation and coding

The quantized spectral coefficients X are noiselessly encoded starting from the lowest-frequency coefficient and progressing to the highest-frequency coefficient. They are encoded by groups of two coefficients a and b gathering in a so-called 2-tuple {a,b}.

Each 2-tuple {a,b} is split into three parts namely, MSB, LSB and the sign. The sign is coded independently from the magnitude using uniform probability distribution. The magnitude itself is further divided in two parts, the two most significant bits (MSBs) and the remaining least significant bitplanes (LSBs, if applicable). The 2-tuples for which the magnitude of the two spectral coefficients is lower or equal to 3 are coded directly by the MSB coding. Otherwise, an escape symbol is transmitted first for signalling any additional bit plane.

The relation between 2-tuple, the individual spectral values *a* and *b* of a 2-tuple, the most significant bit planes *m* and the remaining least significant bit planes, *r,* are illustrated in the example in figure 63. In this example three escape symbols are sent prior to the actual value m, indicating three transmitted least significant bit planes

Figure 63: Example of a coded pair (2-tuple) of spectral values *a* and *b*   
and their representation as *m* and *r*.

The probability model is derived from the past context. The past context is translated on a 12 bits-wise index and maps with the lookup table *ari\_context\_lookup []* to one of the 64 available probability models stored in *ari\_cf\_m[]*.

The past context is derived from two 2-tuples already coded within the same frame. The context can be derived from the direct neighbourhood or located further in the past frequencies. Separate contexts are maintained for the peak regions (coefficients belonging to the harmonic peaks) and other (non-peak) regions according to the harmonic model. If no harmonic model is used, only the other (non-peak) region context is used.

The zeroed spectral values lying in the tail of spectrum are not transmitted. It is achieved by transmitting the index of last non-zeroed 2-tuple. If harmonic model is used, the tail of the spectrum is defined as the tail of spectrum consisting of the peak region coefficients, followed by the other (non-peak) region coefficients, as this definition tends to increase the number of trailing zeros and thus improves coding efficiency. The number of samples to encode is computed as follows:

 (937)

The following data are written into the bitstream with the following order:

1. *lastnz/2-1* is coded on  bits.
2. The entropy-coded MSBs along with escape symbols.
3. The signs with 1 bit-wise code-words
4. The residual quantization bits described in section when the bit budget is not fully used.
5. The LSBs are written backwardly from the end of the bitstream buffer.

The following pseudo-code describes how the context is derived and how the bitstream data for the MSBs, signs and LSBs are computed. The input arguments are the quantized spectral coefficients *X[]*, the size of the considered spectrum *L*, the bit budget *target\_bits*, the harmonic model parameters (*pi, hi)*, and the index of the last non zeroed symbol *lastnz*.

ari\_context\_encode(X[], L,target\_bits,pi[],hi[],lastnz)

{

c[0]=c[1]=p1=p2=0;

for (k=0; k<lastnz; k+=2) {

ari\_copy\_states();

(a1\_i,p1,idx1) = get\_next\_coeff(pi,hi,lastnz);

(b1\_i,p2,idx2) = get\_next\_coeff(pi,hi,lastnz);

t=get\_context(idx1,idx2,c,p1,p2);

esc\_nb = lev1 = 0;

a = a1 = abs(X[a1\_i]);

b = b1 = abs(X[b1\_i]);

/\* sign encoding\*/

if(a1>0) save\_bit(X[a1\_i]>0?0:1);

if(b1>0) save\_bit(X[b1\_i]>0?0:1);

/\* MSB encoding \*/

while (a1 > 3 || b1 > 3) {

pki = ari\_context\_lookup[t+1024\*esc\_nb];

/\* write escape codeword \*/

ari\_encode(17, ari\_cf\_m[pki]);

a1>>=1; b1 >>=1; lev1++;

esc\_nb = min(lev1,3);

}

pki = ari\_context\_lookup[t+1024\*esc\_nb];

ari\_encode(a1+4\*b1, ari\_cf\_m[pki]);

/\* LSB encoding \*/

for(lev=0;lev<lev1;lev++){

write\_bit\_end((a>>lev)&1);

write\_bit\_end((b>>lev)&1);

}

/\*check budget\*/

if(nbbits>target\_bits){

ari\_restore\_states();

break;

}

c=update\_context(a,b,a1,b1,c,p1,p2);

}

write\_sign\_bits();

}

The helper functions *ari\_save\_states()* and *ari\_restore\_states()* are used for saving and restoring the arithmetic coder states respectively. It allows cancelling the encoding of the last symbols if it violates the bit budget. Moreover and in case of bit budget overflow, it is able to fill the remaining bits with zeros till reaching the end of the bit budget or till processing *lastnz* samples in the spectrum.

The other helper functions are described in the following subclauses.

5.3.3.2.8.1.4 Get next coefficient

(a,p,idx) = get\_next\_coeff(pi, hi, lastnz)

If ((ii[0] ≥ lastnz – min(#pi, lastnz)) or

(ii[1] < min(#pi, lastnz) and pi[ii[1]] < hi[ii[0]])) then

{

p=1

idx=ii[1]

a=pi[ii[1]]

}

else

{

p=0

idx=ii[0] + #pi

a=hi[ii[0]]

}

ii[p]=ii[p] + 1

The ii[0] and ii[1] counters are initialized to 0 at the beginning of *ari\_context\_encode()* (and *ari\_context\_decode()* in the decoder).

5.3.3.2.8.1.5 Context update

The context is updated as described by the following pseudo-code. It consists of the concatenation of two 4 bit-wise context elements.

if ()

{

if ()

{



If ()





}

if ()

{



if ()





}

}

else

{



if ()



else



}

5.3.3.2.8.1.6 Get context

The final context is amended in two ways:



if then



if  then



The context *t* is an index from 0 to 1023.

5.3.3.2.8.1.7 Bit consumption estimation

The bit consumption estimation of the context-based arithmetic coder is needed for the rate-loop optimization of the quantization. The estimation is done by computing the bit requirement without calling the arithmetic coder. The generated bits can be accurately estimated by:

cum\_freq= arith\_cf\_m[pki]+m

proba\*= cum\_freq[0]- cum\_freq[1]

nlz=norm\_l(proba) /\*get the number of leading zero \*/

nbits=nlz

proba>>=14

where *proba* is an integer initialized to 16384 and *m* is a MSB symbol.

5.3.3.2.8.1.8 Harmonic model

For both context and envelope based arithmetic coding, a harmonic model is used for more efficient coding of frames with harmonic content. The model is disabled if any of the following conditions apply:

- The bit-rate is not one of 9.6, 13.2, 16.4, 24.4, 32, 48 kbps.

- The previous frame was coded by ACELP.

- Envelope based arithmetic coding is used and the coder type is neither Voiced nor Generic.

- The single-bit harmonic model flag in the bit-stream in set to zero.

When the model is enabled, the frequency domain interval of harmonics is a key parameter and is commonly analysed and encoded for both flavours of arithmetic coders.

5.3.3.2.8.1.8.1 Encoding of Interval of harmonics

When pitch lag and gain are used for the post processing, the lag parameter is utilized for representing the interval of harmonics in the frequency domain. Otherwise, normal representation of interval is applied.

5.3.3.2.8.1.8.1.1 Encoding interval depending on time domain pitch lag

If integer part of pitch lag in time domain  is less than the frame size of MDCT , frequency domain interval unit (between harmonic peaks corresponding to the pitch lag) with 7 bit fractional accuracy is given by

 (938)

where  denotes the fractional part of pitch lag in time domain,denotes the max number of allowable fractional values whose values are either 4 or 6 depending on the conditions.

Since  has limited range, the actual interval between harmonic peaks in the frequency domain is coded relatively to using the bits specified in table 88. Among candidate of multiplication factors, given in the table 89 or table 90, the multiplication number is selected that gives the most suitable harmonic interval of MDCT domain transform coefficients.

 (939)

 (940)

Table 88: Number of bits for specifying the multiplier depending on 

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| NB: | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 3 | 3 | 3 | 3 | 2 | 2 | 2 | 2 | 2 |
| WB: | 5 | 5 | 5 | 5 | 5 | 5 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 2 | 2 | 2 |

Table 89: Candidates of multiplier in the order of  depending on  (NB)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 30 | 32 | 34 | 36 | 38 | 40 |
| 1 | 0.5 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 16 | 20 | 24 | 30 |
| 2 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 14 | 16 | 18 | 20 | 24 | 30 |
| 3 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 14 | 16 | 18 | 20 | 24 | 30 |
| 4 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 14 | 16 | 18 | 20 | 24 | 30 |
| 5 | 1 | 2 | 2.5 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 14 | 16 | 18 | 20 |
| 6 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 16 |
| 7 | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 10 | - | - | - | - | - | - | - | - |
| 8 | 1 | 2 | 3 | 4 | 5 | 6 | 8 | 10 | - | - | - | - | - | - | - | - |
| 9 | 1 | 1.5 | 2 | 3 | 4 | 5 | 6 | 8 | - | - | - | - | - | - | - | - |
| 10 | 1 | 2 | 2.5 | 3 | 4 | 5 | 6 | 8 | - | - | - | - | - | - | - | - |
| 11 | 1 | 2 | 3 | 4 | - | - | - | - | - | - | - | - | - | - | - | - |
| 12 | 1 | 2 | 4 | 6 | - | - | - | - | - | - | - | - | - | - | - | - |
| 13 | 1 | 2 | 3 | 4 | - | - | - | - | - | - | - | - | - | - | - | - |
| 14 | 1 | 1.5 | 2 | 4 | - | - | - | - | - | - | - | - | - | - | - | - |
| 15 | 1 | 1.5 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - |
| 16 | 0.5 | 1 | 2 | 3 | - | - | - | - | - | - | - | - | - | - | - | - |

Table 90: Candidates of multiplier in the order of depending on (WB)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 30 | 32 | 34 | 36 | 38 | 40 |
| 1 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 14 | 16 | 18 | 20 | 22 |
| 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 44 | 48 | 54 | 60 | 68 | 78 | 80 |
| 2 | 1.5 | 2 | 2.5 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 14 | 16 | 18 | 20 |
| 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 38 | 40 | 42 | 44 | 48 | 52 | 54 | 68 |
| 3 | 1 | 1.5 | 2 | 2.5 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| 15 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 30 | 32 | 34 | 36 | 40 | 44 | 48 | 54 |
| 4 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 5.5 | 6 | 6.5 | 7 | 7.5 | 8 | 9 |
| 10 | 11 | 12 | 13 | 14 | 15 | 16 | 18 | 20 | 22 | 24 | 26 | 28 | 34 | 40 | 41 |
| 5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
| 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22.5 | 24 | 25 | 27 | 28 | 30 | 35 |
| 6 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 5.5 | 6 | 7 | 8 | 9 | 10 |
| 7 | 1 | 2 | 2.5 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 12 | 15 | 16 | 18 | 27 |
| 8 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 5 | 6 | 8 | 10 | 15 | 18 | 22 | 24 | 26 |
| 9 | 1 | 1.5 | 2 | 2.5 | 3 | 3.5 | 4 | 5 | 6 | 8 | 10 | 12 | 13 | 14 | 18 | 21 |
| 10 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 4 | 5 | 6 | 8 | 9 | 11 | 12 | 13.5 | 16 | 20 |
| 11 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 4 | 5 | 6 | 7 | 8 | 10 | 11 | 12 | 14 | 20 |
| 12 | 0.5 | 1 | 1.5 | 2 | 2.5 | 3 | 4 | 4.5 | 6 | 7.5 | 9 | 10 | 12 | 14 | 15 | 18 |
| 13 | 0.5 | 1 | 1.25 | 1.5 | 1.75 | 2 | 2.5 | 3 | 3.5 | 4 | 4.5 | 5 | 6 | 8 | 9 | 14 |
| 14 | 0.5 | 1 | 2 | 4 | - | - | - | - | - | - | - | - | - | - | - | - |
| 15 | 1 | 1.5 | 2 | 4 | - | - | - | - | - | - | - | - | - | - | - | - |
| 16 | 1 | 2 | 3 | 4 | - | - | - | - | - | - | - | - | - | - | - | - |

5.3.3.2.8.1.8.1.2 Encoding interval without depending on time domain pitch lag

When pitch lag and gain in the time domain is not used or the pitch gain is less than or equals to 0.46, normal encoding of the interval with un-equal resolution is used.

Unit interval of spectral peaks  is coded as

, (941)

and actual interval  is represented with fractional resolution of  as

. (942)

Each paramter is shown in table 91, where “small size” means when frame size is smaller than 256 of the target bit rates is less than or equal to 150.

Table 91: Un-equal resolution for coding of (0<= index < 256)

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Res* | *base* | *bias* |
|  | 3 | 6 | 0 |
|  | 4 | 8 | 16 |
|  | 3 | 12 | 80 |
| “small size” or | 1 | 28 | 208 |
|  | 0 | 188 | 224 |

5.3.3.2.8.1.8.2 Void

5.3.3.2.8.1.8.3 Search for interval of harmonics

In search of the best interval of harmonics, encoder tries to find the index which can maximize the weighted sum  of the peak part of absolute MDCT coefficients. denotes sum of 3 samples of absolute value of MDCT domain transform coefficients as

 (943)

 (944)

where *num\_peak* is the maximum number that  reaches the limit of samples in the frequency domain.

In case interval does not rely on the pitch lag in time domain, hierarchical search is used to save computational cost. If the index of the interval is less than 80, periodicity is checked by a coarse step of 4. After getting the best interval, finer periodicity is searched around the best interval from -2 to +2. If index is equal to or larger than 80, periodicity is searched for each index.

5.3.3.2.8.1.8.4 Decision of harmonic model

At the initial estimation, number of used bits without harmonic model, , and one with harmonic model, is obtained and the indicator of consumed bits is defined as

, (945)

, (946)

, (947)

where denotes the additional bits for model parameters of periodic harmonic structure, and and indicate the consumed bits when they are larger than the target bits. Thus, the larger, the more preferable to use harmonic model. Relative periodicity is defined as the normalized sum of absolute values for peak regions of the shaped MDCT coefficients as

, (948)

where is the harmonic interval that attain the max value of. When the score of periodicity of this frame is larger than the threshold as

, (949)

this frame is considered to be coded by the harmonic model. The shaped MDCT coefficients divided by gain are quantized to produce a sequence of integer values of MDCT coefficients, , and compressed by arithmetic coding with harmonic model. This process needs iterative convergence process (rate loop) to get  and with consumed bits . At the end of convergence, in order to validate harmonic model, the consumed bits by arithmetic coding with normal (non-harmonic) model for  is additionally calculated and compared with . If  is larger than, arithmetic coding of is revert to use normal model.  can be used for residual quantization for further enhancements. Otherwise, harmonic model is used in arithmetic coding.

In contrast, if the indicator of periodicity of this frame is smaller than or the same as the threshold, quantization and arithmetic coding are carried out assuming the normal model to produce a sequence of integer values of the shaped MDCT coefficients,  with consumed bits . After convergence of rate loop, consumed bits by arithmetic coding with harmonic model for is calculated. If  is larger than, arithmetic coding of  is switched to use harmonic model. Otherwise, normal model is used in arithmetic coding.

5.3.3.2.8.1.9 Use of harmonic information in Context based arithmetic coding

For context based arithmetic coding, all regions are classified into two categories. One is peak part and consists of 3 consecutive samples centered at  ( is a positive integer up to the limit) peak of harmonic peak of,

. (950)

The other samples belong to normal or valley part. Harmonic peak part can be specified by the interval of harmonics and integer multiples of the interval. Arithmetic coding uses different contexts for peak and valley regions.

For ease of description and implementation, the harmonic model uses the following index sequences:

, (951)

, (952)

, the concatenation of  and . (953)

In case of disabled harmonic model, these sequences are , and .

5.3.3.2.8.2 Envelope based arithmetic coder

In the MDCT domain, spectral lines are weighted with the perceptual model  such that each line can be quantized with the same accuracy. The variance of individual spectral lines follow the shape of the linear predictor  weighted by the perceptual model, whereby the weighted shape is .  is calculated by transforming  to frequency domain LPC gains as detailed in subclauses 5.3.3.2.4.1 and 5.3.3.2.4.2. is derived from  after conversion to direct-form coefficients, and applying tilt compensation , and finally transforming to frequency domain LPC gains. All other frequency-shaping tools, as well as the contribution from the harmonic model, shall be also included in this envelope shape . Observe that this gives only the relative variances of spectral lines, while the overall envelope has arbitrary scaling, whereby we must begin by scaling the envelope.

5.3.3.2.8.2.1 Envelope scaling

We will assume that spectral lines  are zero-mean and distributed according to the Laplace-distribution, whereby the probability distribution function is

 (954)

The entropy and thus the bit-consumption of such a spectral line is . However, this formula assumes that the sign is encoded also for those spectral lines which are quantized to zero. To compensate for this discrepancy, we use instead the approximation

 (955)

which is accurate for . We will assume that the bit-consumption of lines with  is  which matches the bit-consumption at . For large  we use the true entropy  for simplicity.

The variance of spectral lines is then If  is the th element of the power of the envelope shape  then  describes the relative energy of spectral lines such that  where  is scaling coefficient. In other words,  describes only the shape of the spectrum without any meaningful magnitude and  is used to scale that shape to obtain the actual variance .

Our objective is that when we encode all lines of the spectrum with an arithmetic coder, then the bit-consumption matches a pre-defined level , that is, . We can then use a bi-section algorithm to determine the appropriate scaling factor  such that the target bit-rate  is reached.

Once the envelope shape  has been scaled such that the expected bit-consumption of signals matching that shape yield the target bit-rate, we can proceed to quantizing the spectral lines.

5.3.3.2.8.2.2 Quantization rate loop

Assume that  is quantized to an integer  such that the quantization interval is  then the probability of a spectral line occurring in that interval is for 

 (956)

and for 

 (957)

It follows that the bit-consumption for these two cases is in the ideal case

 (958)

By pre-computing the terms  and , we can efficiently calculate the bit-consumption of the whole spectrum.

The rate-loop can then be applied with a bi-section search, where we adjust the scaling of the spectral lines by a factor , and calculate the bit-consumption of the spectrum , until we are sufficiently close to the desired bit-rate. Note that the above ideal-case values for the bit-consumption do not necessarily perfectly coincide with the final bit-consumption, since the arithmetic codec works with a finite-precision approximation. This rate-loop thus relies on an approximation of the bit-consumption, but with the benefit of a computationally efficient implementation.

When the optimal scaling  has been determined, the spectrum can be encoded with a standard arithmetic coder. A spectral line which is quantized to a value  is encoded to the interval

 (959)

and  is encoded onto the interval

 (960)

The sign of  will be encoded with one further bit.

Observe that the arithmetic coder must operate with a fixed-point implementation such that the above intervals are bit-exact across all platforms. Therefore all inputs to the arithmetic coder, including the linear predictive model and the weighting filter, must be implemented in fixed-point throughout the system

5.3.3.2.8.2.3 Probability model derivation and coding

When the optimal scaling  has been determined, the spectrum can be encoded with a standard arithmetic coder. A spectral line which is quantized to a value  is encoded to the interval

 (961)

and  is encoded onto the interval

 (962)

The sign of  will be encoded with one further bit.

5.3.3.2.8.2.4 Harmonic model in envelope based arithmetic coding

In case of envelope base arithmetic coding, harmonic model can be used to enhance the arithmetic coding. The similar search procedure as in the context based arithmetic coding is used for estimating the interval between harmonics in the MDCT domain. However, the harmonic model is used in combination of the LPC envelope as shown in figure 64. The shape of the envelope is rendered according to the information of the harmonic analysis.

Harmonic shape at  in the frequency data sample is defined as

, (963)

, otherwise , where  denotes center position of  harmonics.

 (964)

 and are height and width of each harmonics depending on the unit interval as shown,

 (965)

 (966)

Height and width get larger when interval gets larger.

The spectral envelopeis modified by the harmonic shape at  as

**,** (967)

where gain for the harmonic components  is always set as 0.75 for Generic mode, and  is selected from {0.6, 1.4, 4.5, 10.0} that minimizes for Voiced mode using 2 bits,

, (968)

. (969)



Figure 64: Example of harmonic envelope combined with LPC envelope used in envelope based arithmetic coding.

##### 5.3.3.2.9 Global gain coding

5.3.3.2.9.1 Optimizing global gain

The optimum global gain  is computed from the quantized and unquantized MDCT coefficients. For bit rates up to 32 kbps, the adaptive low frequency de-emphasis (see subclause 6.2.2.3.2) is applied to the quantized MDCT coefficients before this step. In case the computation results in an optimum gain less than or equal to zero, the global gain  determined before (by estimate and rate loop) is used.

 (970)

 (971)

5.3.3.2.9.2 Quantization of global gain

For transmission to the decoder the optimum global gain  is quantized to a 7 bit index :

 (972)

The dequantized global gain  is obtained as defined in subclause 6.2.2.3.3).

5.3.3.2.9.3 Residual coding

The residual quantization is a refinement quantization layer refining the first SQ stage. It exploits eventual unused bits *target\_bits-nbbits*, where *nbbits* is the number of bits consumed by the entropy coder. The residual quantization adopts a greedy strategy and no entropy coding in order to stop the coding whenever the bit-stream reaches the desired size.

The residual quantization can refine the first quantization by two means. The first mean is the refinement of the global gain quantization. The global gain refinement is only done for rates at and above 13.2kbps. At most three additional bits is allocated to it. The quantized gain  is refined sequentially starting from *n=0* and incrementing *n* by one after each following iteration:



The second mean of refinement consists of re-quantizing the quantized spectrum line per line. First, the non-zeroed quantized lines are processed with a 1 bit residual quantizer:



Finally, if bits remain, the zeroed lines are considered and quantized with on 3 levels. The rounding offset of the SQ with deadzone was taken into account in the residual quantizer design:



##### 5.3.3.2.10 Noise Filling

On the decoder side noise filling is applied to fill gaps in the MDCT spectrum where coefficients have been quantized to zero. Noise filling inserts pseudo-random noise into the gaps, starting at bin  up to bin . To control the amount of noise inserted in the decoder, a noise factor is computed on encoder side and transmitted to the decoder.

5.3.3.2.10.1 Noise Filling Tilt

To compensate for LPC tilt, a tilt compensation factor is computed. For bitrates below 13.2 kbps the tilt compensation is computed from the direct form quantized LP coefficients , while for higher bitrates a constant value is used:

 (973)

 (974)

5.3.3.2.10.2 Noise Filling Start and Stop Bins

The noise filling start and stop bins are computed as follows:

 (975)

 (976)

5.3.3.2.10.3 Noise Transition Width

At each side of a noise filling segment a transition fadeout is applied to the inserted noise. The width of the transitions (number of bins) is defined as:

 (977)

where denotes that the harmonic model is used for the arithmetic codec and denotes the previous codec mode.

5.3.3.2.10.4 Computation of Noise Segments

The noise filling segments are determined, which are the segments of successive bins of the MDCT spectrum between  and  for which all coefficients are quantized to zero. The segments are determined as defined by the following pseudo-code:



where  and  are the start and stop bins of noise filling segment *j*, and  is the number of segments.

5.3.3.2.10.5 Computation of Noise Factor

The noise factor is computed from the unquantized MDCT coefficients of the bins for which noise filling is applied.

If the noise transition width  is 3 or less bins, an attenuation factor is computed based on the energy of even and odd MDCT bins:

 (978)

 (979)

 (980)

For each segment an error value is computed from the unquantized MDCT coefficients, applying global gain, tilt compensation and transitions:

 (981)

A weight for each segment is computed based on the width of the segment:

 (982)

The noise factor is then computed as follows:

 (983)

5.3.3.2.10.6 Quantization of Noise Factor

For transmission the noise factor is quantized to obtain a 3 bit index:

 (984)

##### 5.3.3.2.11 Intelligent Gap Filling

The *Intelligent Gap Filling* (IGF) tool is an enhanced noise filling technique to fill gaps (regions of zero values) in spectra. These gaps may occur due to coarse quantization in the encoding process where large portions of a given spectrum might be set to zero to meet bit constraints. However, with the IGF tool these missing signal portions are reconstructed on the receiver side (RX) with parametric information calculated on the transmission side (TX). IGF is used only if TCX mode is active.

See table 92 below for all IGF operating points:

Table 92: IGF application modes

|  |  |
| --- | --- |
| Bitrate | Mode |
| 9.6 kbps | WB |
| 9.6 kbps | SWB |
| 13.2 kbps | SWB |
| 16.4 kbps | SWB |
| 24.4 kbps | SWB |
| 32.2 kbps | SWB |
| 48.0 kbps | SWB |
| 16.4 kbps | FB |
| 24.4 kbps | FB |
| 32.0 kbps | FB |
| 48.0 kbps | FB |
| 96.0 kbps | FB |
| 128.0 kbps | FB |

On transmission side, IGF calculates levels on scale factor bands, using a complex or real valued TCX spectrum. Additionally spectral whitening indices are calculated using a spectral flatness measurement and a crest-factor. An arithmetic coder is used for noiseless coding and efficient transmission to receiver (RX) side.

5.3.3.2.11.1 IGF helper functions

5.3.3.2.11.1.1 Mapping values with the transition factor

If there is a transition from CELP to TCX coding () or a TCX 10 frame is signalled (), the TCX frame length may change. In case of frame length change, all values which are related to the frame length are mapped with the function :



 (985)

where  is a natural number, for example a scale factor band offset, and  is a transition factor, see table 97.

5.3.3.2.11.1.2 TCX power spectrum

The power spectrum of the current TCX frame is calculated with:

 (986)

where  is the actual TCX window length,  is the vector containing the real valued part (cos-transformed) of the current TCX spectrum, and  is the vector containing the imaginary (sin-transformed) part of the current TCX spectrum.

5.3.3.2.11.1.3 The spectral flatness measurement function 

Let  be the TCX power spectrum as calculated according to subclause 5.3.3.2.11.1.2 and  the start line and  the stop line of the SFM measurement range.

Thefunction, applied with IGF, is defined with:



 (987)

where  is the actual TCX window length and  is defined with:

 (988)

5.3.3.2.11.1.4 The crest factor function 

Let  be the TCX power spectrum as calculated according to subclause 5.3.3.2.11.1.2 and the start line and  the stop line of the crest factor measurement range.

The  function, applied with IGF, is defined with:



 (989)

where  is the actual TCX window length and  is defined with:

 (990)

5.3.3.2.11.1.5 The mapping function 

The mapping function is defined with:



 (991)

where  is a calculated spectral flatness value and is the noise band in scope. For threshold values ,  refer to table 93 below.

Table 93: Thresholds for whitening for , and 

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bitrate | Mode | nT | ThM | ThS |
| 9.6 kbps | WB | 2 | 0.36, 0.36 | 1.41, 1.41 |
| 9.6 kbps | SWB | 3 | 0.84, 0.89, 0.89 | 1.30, 1.25, 1.25 |
| 13.2 kbps | SWB | 2 | 0.84, 0.89 | 1.30, 1.25 |
| 16.4 kbps | SWB | 3 | 0.83, 0.89, 0.89 | 1.31, 1.19, 1.19 |
| 24.4 kbps | SWB | 3 | 0.81, 0.85, 0.85 | 1.35, 1.23, 1.23 |
| 32.2 kbps | SWB | 3 | 0.91, 0.85, 0.85 | 1.34, 1.35, 1.35 |
| 48.0 kbps | SWB | 1 | 1.15 | 1.19 |
| 16.4 kbps | FB | 3 | 0.63, 0.27, 0.36 | 1.53, 1.32, 0.67 |
| 24.4 kbps | FB | 4 | 0.78, 0.31, 0.34, 0.34 | 1.49, 1.38, 0.65, 0.65 |
| 32.0 kbps | FB | 4 | 0.78, 0.31, 0.34, 0.34 | 1.49, 1.38, 0.65, 0.65 |
| 48.0 kbps | FB | 1 | 0.80 | 1.0 |
| 96.0 kbps | FB | 1 | 0 | 2.82 |
| 128.0 kbps | FB | 1 | 0 | 2.82 |

5.3.3.2.11.1.6 Void

5.3.3.2.11.1.7 IGF scale factor tables

IGF scale factor tables are available for all modes where IGF is applied.

Table 94: Scale factor band offset table

|  |  |  |  |
| --- | --- | --- | --- |
| Bitrate | Mode | Number of bands (nB) | Scale factor band offsets (t[0],t[1],…,t[nB]) |
| 9.6 kbps | WB | 3 | 164, 186, 242, 320 |
| 9.6 kbps | SWB | 3 | 200, 322, 444, 566 |
| 13.2 kbps | SWB | 6 | 256, 288, 328, 376, 432, 496, 566 |
| 16.4 kbps | SWB | 7 | 256, 288, 328, 376, 432, 496, 576, 640 |
| 24.4 kbps | SWB | 8 | 256, 284, 318, 358, 402, 450, 508, 576, 640 |
| 32.2 kbps | SWB | 8 | 256, 284, 318, 358, 402, 450, 508, 576, 640 |
| 48.0 kbps | SWB | 3 | 512, 534, 576, 640 |
| 16.4 kbps | FB | 9 | 256, 288, 328, 376, 432, 496, 576, 640, 720, 800 |
| 24.4 kbps | FB | 10 | 256, 284, 318, 358, 402, 450, 508, 576, 640, 720, 800 |
| 32.0 kbps | FB | 10 | 256, 284, 318, 358, 402, 450, 508, 576, 640, 720, 800 |
| 48.0 kbps | FB | 4 | 512, 584, 656, 728, 800 |
| 96.0 kbps | FB | 2 | 640, 720, 800 |
| 128.0 kbps | FB | 2 | 640, 720, 800 |

The table 94 above refers to the TCX 20 window length and a transition factor 1.00.

For all window lengths apply the following remapping

 (992)

where is the transition factor mapping function described in subclause 5.3.3.2.11.1.1.

5.3.3.2.11.1.8 The mapping function 

Table 95: IGF minimal source subband, 

|  |  |  |
| --- | --- | --- |
| Bitrate | mode |  |
| 9.6 kbps | WB | 30 |
| 9.6 kbps | SWB | 32 |
| 13.2 kbps | SWB | 32 |
| 16.4 kbps | SWB | 32 |
| 24.4 kbps | SWB | 32 |
| 32.2 kbps | SWB | 32 |
| 48.0 kbps | SWB | 64 |
| 16.4 kbps | FB | 32 |
| 24.4 kbps | FB | 32 |
| 32.0 kbps | FB | 32 |
| 48.0 kbps | FB | 64 |
| 96.0 kbps | FB | 64 |
| 128.0 kbps | FB | 64 |

For every mode a mapping function is defined in order to access source lines from a given target line in IGF range.

Table 96: Mapping functions for every mode

|  |  |  |  |
| --- | --- | --- | --- |
| Bitrate | Mode | nT | mapping Function |
| 9.6 kbps | WB | 2 |  |
| 9.6 kbps | SWB | 3 |  |
| 13.2 1kbps | SWB | 2 |  |
| 16.4 kbps | SWB | 3 |  |
| 24.4 kbps | SWB | 3 |  |
| 32.2 kbps | SWB | 3 |  |
| 48.0 kbps | SWB | 1 |  |
| 16.4 kbps | FB | 3 |  |
| 24.4 kbps | FB | 4 |  |
| 32.0 kbps | FB | 4 |  |
| 48.0 kbps | FB | 1 |  |
| 96.0 kbps | FB | 1 |  |
| 128.0 kbps | FB | 1 |  |

The mapping function  is defined with:

 (993)

The mapping function  is defined with:

 (994)

The mapping function  is defined with:

 (995)

The mapping function  is defined with:

 (996)

The mapping function  is defined with:

 (997)

The mapping function  is defined with:

 (998)

The mapping function  is defined with:

 (999)

The mapping function  is defined with:

 (1000)

The value is the appropriate transition factor, see table 97 and is described in subclause 5.3.3.2.11.1.1.

Please note, that all values  shall be already mapped with the function as described in subclause 5.3.3.2.11.1.1. Values for  are defined in table 94.

The here described mapping functions will be referenced in the text as “mapping function m” assuming, that the proper function for the current mode is selected.

5.3.3.2.11.2 IGF input elements (TX)

The IGF encoder module expects the following vectors and flags as an input:

: vector with real part of the current TCX spectrum 

: vector with imaginary part of the current TCX spectrum 

: vector with values of the TCX power spectrum 

: flag, signalling if the current frame contains a transient, see subclause 5.3.2.4.1.1

: flag, signalling a TCX 10 frame

: flag, signalling a TCX 20 frame

: flag, signalling CELP to TCX transition; generate flag by test whether last frame was CELP

: flag, signalling that the current frame is independent from the previous frame

Listed in table 97, the following combinations signalled through flags ,  and  are allowed with IGF:

Table 97: TCX transitions, transition factor , window length

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Bitrate / Mode |  |  |  | Transition factor | Window length |
| 9.6 kbps / WB | false | true | false | 1.00 | 320 |
| false | true | true | 1.25 | 400 |
| 9.6 kbps / SWB | false | true | false | 1.00 | 640 |
| false | true | true | 1.25 | 800 |
| 13.2 kbps / SWB | false | true | false | 1.00 | 640 |
| false | true | true | 1.25 | 800 |
| 16.4 kbps / SWB | false | true | false | 1.00 | 640 |
| false | true | true | 1.25 | 800 |
| 24.4 kbps / SWB | false | true | false | 1.00 | 640 |
| false | true | true | 1.25 | 800 |
| 32.0 kbps / SWB | false | true | false | 1.00 | 640 |
| false | true | true | 1.25 | 800 |
| 48.0 kbps / SWB | false | true | false | 1.00 | 640 |
| false | true | true | 1.00 | 640 |
| true | false | false | 0.50 | 320 |
| 16.4 kbps / FB | false | true | false | 1.00 | 960 |
| false | true | true | 1.25 | 1200 |
| 24.4 kbps / FB | false | true | false | 1.00 | 960 |
| false | true | true | 1.25 | 1200 |
| 32.0 kbps / FB | false | true | false | 1.00 | 960 |
| false | true | true | 1.25 | 1200 |
| 48.0 kbps / FB | false | true | false | 1.00 | 960 |
| false | true | true | 1.00 | 960 |
| true | false | false | 0.50 | 480 |
| 96.0 kbps / FB | false | true | false | 1.00 | 960 |
| false | true | true | 1.00 | 960 |
| true | false | false | 0.50 | 480 |
| 128.0 kbps / FB | false | true | false | 1.00 | 960 |
| false | true | true | 1.00 | 960 |
| true | false | false | 0.50 | 480 |

5.3.3.2.11.3 IGF functions on transmission (TX) side

All function declaration assumes that input elements are provided by a frame by frame basis. The only exceptions are two consecutive TCX 10 frames, where the second frame is encoded dependent on the first frame.

5.3.3.2.11.4 IGF scale factor calculation

This subclause describes how the IGF scale factor vector  is calculated on transmission (TX) side.

5.3.3.2.11.4.1 Complex valued calculation

In case the TCX power spectrum  is available the IGF scale factor values  are calculated using :

 (1001)

and let be the mapping function which maps the IGF target range into the IGF source range described in subclause 5.3.3.2.11.1.8, calculate:

 (1002)

 (1003)

where  shall be already mapped with the function see subclause 5.3.3.2.11.1.1, and  are the number of IGF scale factor bands, see table 94.

Calculate  with:

 (1004)

and limit  to the range  with

 (1005)

The values  will be transmitted to the receiver (RX) side after further lossless compression with an arithmetic coder described in subclause 5.3.3.2.11.8.

5.3.3.2.11.4.2 Real valued calculation

If the TCX power spectrum is not available calculate:

 (1006)

where  shall be already mapped with the function see subclause 5.3.3.2.11.1.1, and  are the number of bands, see table 94.

Calculate  with:

 (1007)

and limit  to the range  with

 (1008)

The values  will be transmitted to the receiver (RX) side after further lossless compression with an arithmetic coder described in subclause 5.3.3.2.11.8.

5.3.3.2.11.5 IGF tonal mask

In order to determine which spectral components should be transmitted with the core coder, a tonal mask is calculated. Therefore all significant spectral content is identified whereas content that is well suited for parametric coding through IGF is quantized to zero.

5.3.3.2.11.5.1 IGF tonal mask calculation

In case the TCX power spectrum  is not available, all spectral content above  is deleted:

 (1009)

where  is the real valued TCX spectrum after applying TNS and  is the current TCX window length.

In case the TCX power spectrum  is available, calculate:

 (1010)

where  is the first spectral line in IGF range and  is 1.0 for 9.6 and 13.2 kbit/s SWB and 2.0 for all other configurations.

Given , apply the following algorithm:

Initialize  and :





for (i = ; i <  ; i++) {

if () {







} else if() {







}

}

if , set 

5.3.3.2.11.6 IGF spectral flatness calculation

Table 98: Number of tiles  and tile width 

|  |  |  |  |
| --- | --- | --- | --- |
| Bitrate | Mode |  |  |
| 9.6 kbps | WB | 2 |  |
| 9.6 kbps | SWB | 3 |  |
| 13.2 kbps | SWB | 2 |  |
| 16.4 kbps | SWB | 3 |  |
| 24.4 kbps | SWB | 3 |  |
| 32.2 kbps | SWB | 3 |  |
| 48.0 kbps | SWB | 1 |  |
| 16.4 kbps | FB | 3 |  |
| 24.4 kbps | FB | 4 |  |
| 32.0 kbps | FB | 4 |  |
| 48.0 kbps | FB | 1 |  |
| 96.0 kbps | FB | 1 |  |
| 128.0 kbps | FB | 1 |  |

For the IGF spectral flatness calculation two static arrays, and , both of size are needed to hold filter-states over frames. Additionally a static flag is needed to save the information of the input flag  from the previous frame.

5.3.3.2.11.6.1 Resetting filter states

The vectors  and  are both static arrays of size  in the IGF module and both arrays are initialised with zeroes:

 (1011)

This initialisation shall be done

* with codec start up
* with any bitrate switch
* with any codec type switch
* with a transition from CELP to TCX, e.g. 
* if the current frame has transient properties, e.g.

5.3.3.2.11.6.2 Resetting current whitening levels

The vector  shall be initialised with zero for all tiles,

 (1012)

* with codec start up
* with any bitrate switch
* with any codec type switch
* with a transition from CELP to TCX, e.g. 

5.3.3.2.11.6.3 Calculation of spectral flatness indices

The following steps 1) to 4) shall be executed consecutive:

1. Update previous level buffers and initialize current levels:

 (1013)

In case  or  is true, apply

 (1014)

else, if the power spectrum  is available, calculate

 (1015)

with

 (1016)

where  is a spectral flatness measurement function, described in subclause 5.3.3.2.11.1.3 and  is a crest-factor function described in subclause 5.3.3.2.11.1.4.

Calculate:

 (1017)

After calculation of the vector , the filter states are updated with:

 (1018)

1. A mapping function  is applied to the calculated values to obtain a whitening level index vector  The mapping function  is described in subclause 5.3.3.2.11.1.5.

 (1019)

1. With selected modes, see table 99, apply the following final mapping:

 (1020)

Table 99: modes for step 4) mapping

|  |  |  |
| --- | --- | --- |
| Bitrate | mode | mapping |
| 9.6 kbps | WB | apply |
| 9.6 kbps | SWB | apply |
| 13.2 kbps | SWB | NOP |
| 16.4 kbps | SWB | apply |
| 24.4 kbps | SWB | apply |
| 32.2 kbps | SWB | apply |
| 48.0 kbps | SWB | NOP |
| 16.4 kbps | FB | apply |
| 24.4 kbps | FB | apply |
| 32.0 kbps | FB | apply |
| 48.0 kbps | FB | NOP |
| 96.0 kbps | FB | NOP |
| 128.0 kbps | FB | NOP |

After executing step 4) the whitening level index vector  is ready for transmission.

5.3.3.2.11.6.4 Coding of IGF whitening levels

IGF whitening levels, defined in the vector , are transmitted using 1 or 2 bits per tile. The exact number of total bits required depends on the actual values contained in  and the value of the  flag. The detailed processing is described in the pseudo code below:

 = 1;

nTiles = ;

k = 0;

if (

 = 0;

} else {

for (k = 0; k < nTiles ; k++) {

if ( != ) {

 = 0;

break;

}

}

}

if () {

write\_bit(1);

} else {

if (!) {

write\_bit(0);

}

encode\_whitening\_level();

for (k = 1; k < nTiles ; k++) {

 = 1;

if ( != ) {

 = 0;

break;

}

}

if (!) {

write\_bit(1);

for (k = 1; k < nTiles ; k++) {

encode\_whitening\_level();

}

} else {

write\_bit(0);

}

}

wherein the vector  contains the whitening levelsfrom the previous frame and the function encode\_whitening\_level takes care of the actual mapping of the whitening level to a binary code. The function is implemented according to the pseudo code below:

if (

write\_bit(0);

} else {

write\_bit(1);

if (

write\_bit(0);

} else {

write\_bit(1);

}

}

5.3.3.2.11.7 IGF temporal flatness indicator

The temporal envelope of the reconstructed signal by the IGF is flattened on the receiver (RX) side according to the transmitted information on the temporal envelope flatness, which is an IGF flatness indicator.

The temporal flatness is measured as the linear prediction gain in the frequency domain. Firstly, the linear prediction of the real part of the current TCX spectrum is performed and then the prediction gain  is calculated:

 (1021)

where  = *i*-th PARCOR coefficient obtained by the linear prediction.

From the prediction gain  and the prediction gain  described in subclause 5.3.3.2.2.3, the IGF temporal flatness indicator flag  is defined as

 (1022)

5.3.3.2.11.8 IGF noiseless coding

The IGF scale factor vector  is noiseless encoded with an arithmetic coder in order to write an efficient representation of the vector to the bit stream.

The module uses the common raw arithmetic encoder functions from the infrastructure, which are provided by the core encoder. The functions used are , which encodes the value , , which encodes  from an alphabet of 27 symbols () using the cumulative frequency table , , which initializes the arithmetic encoder, and , which finalizes the arithmetic encoder.

5.3.3.2.11.8.1 IGF independency flag

The internal state of the arithmetic encoder is reset in case the  flag has the value . This flag may be set to  only in modes where TCX10 windows (see table 97) are used for the second frame of two consecutive TCX 10 frames.

5.3.3.2.11.8.2 IGF all-Zero flag

The IGF all-Zero flag signals that all of the IGF scale factors are zero:

 (1023)

The  flag is written to the bit stream first. In case the flag is , the encoder state is reset and no further data is written to the bit stream, otherwise the arithmetic coded scale factor vector  follows in the bit stream.

5.3.3.2.11.8.3 IGF arithmetic encoding helper functions

5.3.3.2.11.8.3.1 The reset function

The arithmetic encoder states consist of , and the  vector, which represents the value of the vector  preserved from the previous frame. When encoding the vector , the value 0 for  means that there is no previous frame available, therefore  is undefined and not used. The value 1 for  means that there is a previous frame available therefore  has valid data and it is used, this being the case only in modes where TCX10 windows (see table 97) are used for the second frame of two consecutive TCX 10 frames. For resetting the arithmetic encoder state, it is enough to set .

If a frame has  set, the encoder state is reset before encoding the scale factor vector . Note that the combination  and  is valid, and may happen for the second frame of two consecutive TCX 10 frames, when the first frame had . In this particular case, the frame uses no context information from the previous frame (the  vector), because , and it is actually encoded as an independent frame.

5.3.3.2.11.8.3.2 The arith\_encode\_bits function

The  function encodes an unsigned integer , of length  bits, by writing one bit at a time.

arith\_encode\_bits(x, nBits)

{

for (i = nBits - 1; i >= 0; --i) {

bit = (x >> i) & 1;

ari\_encode\_14bits\_sign(bit);

}

}

5.3.3.2.11.8.3.2 The save and restore encoder state functions

Saving the encoder state is achieved using the function , which copies  and  vector into  and  vector, respectively. Restoring the encoder state is done using the complementary function , which copies back  and  vector into  and  vector, respectively.

5.3.3.2.11.8.4 IGF arithmetic encoding

Please note that the arithmetic encoder should be capable of counting bits only, e.g., performing arithmetic encoding without writing bits to the bit stream. If the arithmetic encoder is called with a counting request, by using the parameter  set to , the internal state of the arithmetic encoder shall be saved before the call to the top level function  and restored and after the call, by the caller. In this particular case, the bits internally generated by the arithmetic encoder are not written to the bit stream.

The  function encodes the integer valued prediction residual , using the cumulative frequency table , and the table offset . The table offset  is used to adjust the value  before encoding, in order to minimize the total probability that a very small or a very large value will be encoded using escape coding, which slightly is less efficient. The values which are between  and , inclusive, are encoded directly using the cumulative frequency table , and an alphabet size of .

For the above alphabet of SYMBOLS\_IN\_TABLE symbols, the values 0 and  are reserved as escape codes to indicate that a value is too small or too large to fit in the default interval. In these cases, the value  indicates the position of the value in one of the tails of the distribution. The value  is encoded using 4 bits if it is in the range , or using 4 bits with value 15 followed by extra 6 bits if it is in the range {15, or using 4 bits with value 15 followed by extra 6 bits with value 63 followed by extra 7 bits if it is larger or equal than . The last of the three cases is mainly useful to avoid the rare situation where a purposely constructed artificial signal may produce an unexpectedly large residual value condition in the encoder.

arith\_encode\_residual(x, cumulativeFrequencyTable, tableOffset)

{

x += tableOffset;

if ((x >= MIN\_ENC\_SEPARATE) && (x <= MAX\_ENC\_SEPARATE)) {

ari\_encode\_14bits\_ext((x - MIN\_ENC\_SEPARATE) + 1, cumulativeFrequencyTable);

return;

} else if (x < MIN\_ENC\_SEPARATE) {

extra = (MIN\_ENC\_SEPARATE - 1) - x;

ari\_encode\_14bits\_ext(0, cumulativeFrequencyTable);

} else { /\* x > MAX\_ENC\_SEPARATE \*/

extra = x - (MAX\_ENC\_SEPARATE + 1);

ari\_encode\_14bits\_ext(SYMBOLS\_IN\_TABLE - 1, cumulativeFrequencyTable);

}

if (extra < 15) {

arith\_encode\_bits(extra, 4);

} else { /\* extra >= 15 \*/

arith\_encode\_bits(15, 4);

extra -= 15;

if (extra < 63) {

arith\_encode\_bits(extra, 6);

} else { /\* extra >= 63 \*/

arith\_encode\_bits(63, 6);

extra -= 63;

arith\_encode\_bits(extra, 7);

}

}

}

The function  encodes the scale factor vector , which consists of  integer values. The value  and the  vector, which constitute the encoder state, are used as additional parameters for the function. Note that the top level function  must call the common arithmetic encoder initialization function  before calling the function , and also call the arithmetic encoder finalization function  afterwards.

The function  is used to quantize a context value , by limiting it to , and it is defined as:

quant\_ctx(ctx)

{

if (abs(ctx) <= 3) {

return ctx;

} else if (ctx > 3) {

return 3;

} else { /\* ctx < -3 \*/

return -3;

}

}

The definitions of the symbolic names indicated in the comments from the pseudo code, used for computing the context values, are listed in the following table 100:

Table 100: Definition of symbolic names

|  |  |
| --- | --- |
| the previous frame (when available) | the current frame |
|  | (the value to be coded) |
|  | (when available) |
|  | (when available) |

encode\_sfe\_vector(t, prev, g, nB)

for (f = 0; f < nB; f++) {

if (t == 0) {

if (f == 0) {

ari\_encode\_14bits\_ext(g[f] >> 2, cf\_se00);

arith\_encode\_bits(g[f] & 3, 2); /\* LSBs as 2 bit raw \*/

}

else if (f == 1) {

pred = g[f - 1]; /\* pred = b \*/

arith\_encode\_residual(g[f] - pred, cf\_se01, cf\_off\_se01);

} else { /\* f >= 2 \*/

pred = g[f - 1]; /\* pred = b \*/

ctx = quant\_ctx(g[f - 1] - g[f - 2]); /\* Q(b - e) \*/

arith\_encode\_residual(g[f] - pred, cf\_se02[CTX\_OFFSET + ctx)],

cf\_off\_se02[IGF\_CTX\_OFFSET + ctx]);

}

}

else { /\* t == 1 \*/

if (f == 0) {

pred = prev[f]; /\* pred = a \*/

arith\_encode\_residual(x[f] - pred, cf\_se10, cf\_off\_se10);

} else { /\* (t == 1) && (f >= 1) \*/

pred = prev[f] + g[f - 1] - prev[f - 1]; /\* pred = a + b - c \*/

ctx\_f = quant\_ctx(prev[f] - prev[f - 1]); /\* Q(a - c) \*/

ctx\_t = quant\_ctx(g[f - 1] - prev[f - 1]); /\* Q(b - c) \*/

arith\_encode\_residual(g[f] - pred,

cf\_se11[CTX\_OFFSET + ctx\_t][CTX\_OFFSET + ctx\_f)],

cf\_off\_se11[CTX\_OFFSET + ctx\_t][CTX\_OFFSET + ctx\_f]);

}

}

}

}

There are five cases in the above function, depending on the value of  and also on the position  of a value in the vector :

* when  and , the first scalefactor of an independent frame is coded, by splitting it into the most significant bits which are coded using the cumulative frequency table , and the least two significant bits coded directly.
* when  and , the second scale factor of an independent frame is coded (as a prediction residual) using the cumulative frequency table .
* when  and , the third and following scale factors of an independent frame are coded (as prediction residuals) using the cumulative frequency table , determined by the quantized context value .
* when  and , the first scalefactor of a dependent frame is coded (as a prediction residual) using the cumulative frequency table .
* when  and , the second and following scale factors of a dependent frame are coded (as prediction residuals) using the cumulative frequency table  determined by the quantized context values  and .

Please note that the predefined cumulative frequency tables , , and the table offsets ,  depend on the current operating point and implicitly on the bitrate, and are selected from the set of available options during initialization of the encoder for each given operating point. The cumulative frequency table  is common for all operating points, and cumulative frequency tables  and , and the corresponding table offsets  and  are also common, but they are used only for operating points corresponding to bitrates larger or equal than 48 kbps, in case of dependent TCX 10 frames (when ).

5.3.3.2.11.9 IGF bit stream writer

The arithmetic coded IGF scale factors, the IGF whitening levels and the IGF temporal flatness indicator are consecutively transmitted to the decoder side via bit stream. The coding of the IGF scale factors is described in subclause 5.3.3.2.11.8.4. The IGF whitening levels are encoded as presented in subclause 5.3.3.2.11.6.4. Finally the IGF temporal flatness indicator flag, represented as one bit, is written to the bit stream.

In case of a TCX20 frame, i.e. (), and no counting request is signalled to the bit stream writer, the output of the bit stream writer is fed directly to the bit stream. In case of a TCX10 frame (), where two sub-frames are coded dependently within one 20ms frame, the output of the bit stream writer for each sub-frame is written to a temporary buffer, resulting in a bit stream containing the output of the bit stream writer for the individual sub-frames. The content of this temporary buffer is finally written to the bit stream.

##### 5.3.3.2.12 Memory updates

5.3.3.2.12.1 Internal decoder

Subsequent to the MDCT based TCX encoding, a simplified decoding at  is performed to enable and update the filter memories for CELP coding. Based on the quantized MDCT spectral coefficients, the following decoding steps are performed:

* Adaptive low frequency deemphasis (subclause 6.2.2.3.2)
* Global gain decoding (subclause 6.2.2.3.3)
* Residual dequantizer (subclause 6.2.2.3.4)
* Formant enhancement (subclause 6.2.2.3.5)
* Noise filling (subclause 6.2.2.3.6)
* Application of global gain and LPC shaping in MDCT domain (subclause 6.2.2.3.7)
* Inverse window grouping (subclause 6.2.2.3.9)
* Temporal noise shaping (subclause 6.2.2.3.10)

The resulting MDCT spectrum is transformed to a time-domain signal at  using the corresponding frequency-to-time transformation, as described in subclause 6.2.4, resulting in the synthesized TCX decoder output .

5.3.3.2.12.2 Update of filter memories

The updating of the states of the filter is performed by applying the perceptually weighted LPC coefficients on the synthesized TCX decoder output  and subtract this from the perceptually weighted speech signal  The LPC coefficients correspond to the 4th subframe for and to the 5th subframe for 

 (1024)

The pre-emphasis-filter  is applied to  for updating the synthesis buffers used in the target signal computation to calculate the residual signal ( subclause 5.2.3.1.2), and for LPC synthesis filter states used in the CELP encoder. For updating the filter states of LP residual signal computation, the pre-emphasized synthesis buffer is filtered by the quantized LPC coefficients (see subclause 5.2.3.1.1).

##### 5.3.3.2.13 Global Gain Adjuster

For bitrates less than 13.2 kbps the optimum global gain  is partially recomputed after noise filling and formant enhancement have been applied in the internal decoder:

 (1025)

The new global gain  is quantized again for transmission in the bit stream to an index , replacing the previously computed one.

### 5.3.4 High Quality MDCT coder (HQ)

#### 5.3.4.1 Low-rate HQ coder

The structure of the Low-rate HQ MDCT core coder is presented in figure 65. Encoding is performed in the MDCT domain. Based on the input signal bandwidth, operational bitrates and signal characteristics the coding modes are decided. For example, an input signal with the sampling frequency of 32 kHz or 48 kHz with operational bitrates of 13.2 kbps and 16.4 kbps will be encoded using any one of the three alternative modes: Transient mode, Normal mode, and Harmonic mode. For an input signal with sampling frequency of 8 kHz or 16 kHz, the tonality estimation block is not used, so the input signal is either encoded under Transient or Normal mode. The mode classification decision is described in subclause 5.3.4.1.1. The following table 101 summarizes the supported modes per bit rate and bandwidth. Mode information (one bit for Transient/non-Transient, and one bit for Normal/Harmonic) is encoded and transmitted to the decoder side.

Table 101: Supported modes for low-rate HQ coder

|  |  |  |
| --- | --- | --- |
| Bitrate [kbps] | Bandwidth | Supported modes |
| 7.2, 8 | NB | Normal, Transient |
| 13.2, 16.4 | NB, WB | Normal, Transient |
| SWB | Normal, Transient, Harmonic |

Based on the mode, the obtained spectral coefficients are grouped into bands of unequal lengths. The energy of each band is estimated and the resulting spectral envelope consisting of the energies of all bands is quantized and encoded using Huffman coding. The quantized energies are used as input for bit allocation. The spectral coefficients are quantized using TCQ and USQ and encoded based on the allocated bits for each frequency band. The encoded spectral coefficients are adjusted using the quantized energies. The level of the most significant spectral coefficients is adjusted using an estimated gain which is coded and transmitted to the decoder. The spectral bands which are not quantized in the HF region are identified and coded with relatively few bits using the quantized spectral coefficients information.

The parameters transmitted from encoder to decoder are the mode selection, energy envelope information, the quanitzed spectral coefficients, LF and the HF parameters.

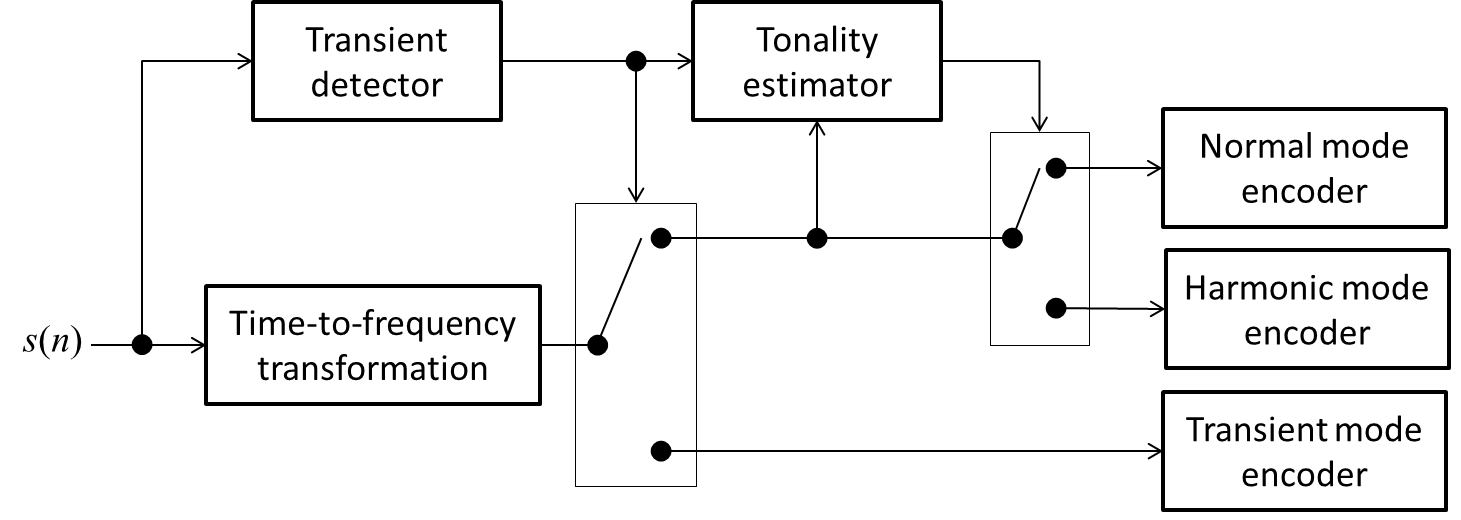


Figure 65: Structural block diagram of the Low-rate HQ coder

##### 5.3.4.1.1 Tonality Estimation

Non transient mode is referred as Normal mode for NB and WB inputs, whereas for SWB and FB inputs at 13.2 and 16.4 kbps, Non-transient signals are further classified as Normal and Harmonic mode. The detailed description for Normal and Harmonic mode classification is as follows,

448 MDCT coefficients in the 2400-13600 Hz frequency range are used in order to perform this classification. 14 sub-bands are split, which is equally 32 coefficients per sub-band. The frequency sharpness  is defined as the ratio between the peak and the average magnitudes in each sharpness band:

 (1026)

where the peak magnitude of spectral coefficients in a sharpness band, denoted, is:

 (1027)

The counter  denotes the number of sub-bands which have harmonic characteristics corresponding to the first 5 sub-bands.  is initialized to zero and increased by one if . The counter  denotes the number of sub-band which have harmonic characteristics corresponding to the remaining 9 sub-bands.  is initialized to zero and increased by one if  and .

The mode counters  and  are introduced for harmonic mode detection and are initialized to zero.  is increased by one and  is decreased by one if  and . Otherwise, the mode counter  is decreased by one and  is increased by one.  and  are constrained to only hold values between 0 and 8. The calculations of  and  are summarized as follows,

 (1028)

The current frame mode is classified as harmonic if ,  and ; or if .

##### 5.3.4.1.2 Grouping of spectral coefficients

The spectral coefficients are divided to obtain bands of variable lengths; the total number of bands varies depending on the signal bandwidth (NB, WB, SWB, and FB), operational bitrates and the classifier. Tables 103 to 108 describe the band structure for different signal bandwidths and operational bitrates. The NB band structure is used for NB signals, the WB band structure is used for WB signals, and the SWB band structure is used for SWB and FB signals.

The total number of bands and the corresponding bandwidth used for NB, WB and SWB is presented in table 102. The band structure and the number of bands for FB are same as SWB for 13.2 and 16.4 kbps.

In case of the Transient mode, the coefficients of the equivalent four 5-ms transforms are consecutively joined, and the bandwidth varies based on signal bandwidth. table 103, 104, 105 shows the detailed band structure for the NB, WB, and SWB (and FB) Transient frames based on the operational bitrates. In each table, *b* denotes the index of the band, *kwidth*(*b*) is the corresponding band length, and *kstart*(*b*) and *kend*(*b*) denote the start and end index of the spectral coefficients forming the band.

Table 102: Number of bands and its corresponding bandwidth

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Bitrate (kbps) | Nbands | | Bandwidth |
| Transient | Normal/Harmonic |
| NB | 7.2, 8 | 16 (4\*4) | 13 | 160 |
| 13.2 | 20(4\*5) | 19 |
| WB | 13.2 | 28(4\*7) | 18 | 320 |
| 16.4 | 20 |
| SWB, FB | 13.2 ,16.4 | 32(4\*8) | 22 | 568 |
| 24 | 640 |

Table 103: Band structure for NB Transient frames

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Transient mode | | | | | |
|  | 7.2,8 kbps | | | 13.2 kbps | | |
| b |  |  |  |  |  |  |
| 0 | 6 | 0 | 5 | 6 | 0 | 5 |
| 1 | 8 | 6 | 13 | 7 | 6 | 12 |
| 2 | 11 | 14 | 24 | 7 | 13 | 19 |
| 3 | 15 | 25 | 39 | 9 | 20 | 28 |
| 4 | 6 | 40 | 45 | 11 | 29 | 39 |
| 5 | 8 | 46 | 53 | 6 | 40 | 45 |
| 6 | 11 | 54 | 64 | 7 | 46 | 52 |
| 7 | 15 | 65 | 79 | 7 | 53 | 59 |
| 8 | 6 | 80 | 85 | 9 | 60 | 68 |
| 9 | 8 | 86 | 93 | 11 | 69 | 79 |
| 10 | 11 | 94 | 104 | 6 | 80 | 85 |
| 11 | 15 | 105 | 119 | 7 | 86 | 92 |
| 12 | 6 | 120 | 125 | 7 | 93 | 99 |
| 13 | 8 | 126 | 133 | 9 | 100 | 108 |
| 14 | 11 | 134 | 144 | 11 | 109 | 119 |
| 15 | 15 | 145 | 159 | 6 | 120 | 125 |
| 16 | - | - | - | 7 | 126 | 132 |
| 17 | - | - | - | 7 | 133 | 139 |
| 18 | - | - | - | 9 | 140 | 148 |
| 19 | - | - | - | 11 | 149 | 159 |

Table 104: Band structure for WB Transient frames

|  |  |  |  |
| --- | --- | --- | --- |
|  | Transient mode | | |
|  | 13.2, 16.4 kbps | | |
| *b* |  |  |  |
| 0 | 6 | 0 | 5 |
| 1 | 7 | 6 | 12 |
| 2 | 8 | 13 | 20 |
| 3 | 10 | 21 | 30 |
| 4 | 12 | 31 | 42 |
| 5 | 16 | 43 | 58 |
| 6 | 21 | 59 | 79 |
| 7 | 6 | 80 | 85 |
| 8 | 7 | 86 | 92 |
| 9 | 8 | 93 | 100 |
| 10 | 10 | 101 | 110 |
| 11 | 12 | 111 | 122 |
| 12 | 16 | 123 | 138 |
| 13 | 21 | 139 | 159 |
| 14 | 6 | 160 | 165 |
| 15 | 7 | 166 | 172 |
| 16 | 8 | 173 | 180 |
| 17 | 10 | 181 | 190 |
| 18 | 12 | 191 | 202 |
| 19 | 16 | 203 | 218 |
| 20 | 21 | 219 | 239 |
| 21 | 6 | 240 | 245 |
| 22 | 7 | 246 | 252 |
| 23 | 8 | 253 | 260 |
| 24 | 10 | 261 | 270 |
| 25 | 12 | 271 | 282 |
| 26 | 16 | 283 | 298 |
| 27 | 21 | 299 | 319 |

Table 105: Band structure for SWB, FB Transient frames

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Transient mode | | | | | | |
|  | 13.2 kbps | | | 16.4 kbps | | |
| *b* |  |  |  |  |  |  |
| 0 | 7 | 0 | 6 | 8 | 0 | 7 |
| 1 | 8 | 7 | 14 | 9 | 15 | 8 |
| 2 | 10 | 15 | 24 | 11 | 25 | 17 |
| 3 | 11 | 25 | 35 | 13 | 36 | 28 |
| 4 | 15 | 36 | 50 | 17 | 51 | 41 |
| 5 | 21 | 51 | 71 | 23 | 71 | 58 |
| 6 | 29 | 72 | 100 | 32 | 99 | 81 |
| 7 | 41 | 101 | 141 | 47 | 140 | 113 |
| 8 | 7 | 142 | 148 | 8 | 160 | 167 |
| 9 | 8 | 149 | 156 | 9 | 168 | 176 |
| 10 | 10 | 157 | 166 | 11 | 177 | 187 |
| 11 | 11 | 167 | 177 | 13 | 188 | 200 |
| 12 | 15 | 178 | 192 | 17 | 201 | 217 |
| 13 | 21 | 193 | 213 | 23 | 218 | 240 |
| 14 | 29 | 214 | 242 | 32 | 241 | 272 |
| 15 | 41 | 243 | 283 | 47 | 273 | 319 |
| 16 | 7 | 284 | 290 | 8 | 320 | 327 |
| 17 | 8 | 291 | 298 | 9 | 328 | 336 |
| 18 | 10 | 299 | 308 | 11 | 337 | 347 |
| 19 | 11 | 309 | 319 | 13 | 348 | 360 |
| 20 | 15 | 320 | 334 | 17 | 361 | 377 |
| 21 | 21 | 335 | 355 | 23 | 378 | 400 |
| 22 | 29 | 356 | 384 | 32 | 401 | 432 |
| 23 | 41 | 385 | 425 | 47 | 433 | 479 |
| 24 | 7 | 426 | 432 | 8 | 480 | 487 |
| 25 | 8 | 433 | 440 | 9 | 488 | 496 |
| 26 | 10 | 441 | 450 | 11 | 497 | 507 |
| 27 | 11 | 451 | 461 | 13 | 508 | 520 |
| 28 | 15 | 462 | 476 | 17 | 521 | 537 |
| 29 | 21 | 477 | 497 | 23 | 538 | 560 |
| 30 | 29 | 498 | 526 | 32 | 561 | 592 |
| 31 | 41 | 527 | 567 | 47 | 593 | 639 |

In the Normal mode of operation, the bands have different sizes that increase with increasing frequency. This subdivision allows a consistent representation of the spectrum which closely resembles that of the human ear. Higher frequency resolution is used for low frequencies, while lower frequency resolution is used for high frequencies. The detailed allocation of spectral coefficients to bands for NB, WB, SWB and FB signals are presented in tables 106, 107, 108

Table 106: Band structure for NB Normal mode frames

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Normal mode | | | | | | |
|  | 7.2,8 kbps | | | 13.2 kbps | | |
| *b* |  |  |  |  |  |  |
| 0 | 6 | 0 | 5 | 6 | 0 | 5 |
| 1 | 6 | 6 | 11 | 6 | 6 | 11 |
| 2 | 6 | 12 | 17 | 6 | 12 | 17 |
| 3 | 6 | 18 | 23 | 6 | 18 | 23 |
| 4 | 7 | 24 | 30 | 6 | 24 | 29 |
| 5 | 8 | 31 | 38 | 6 | 30 | 35 |
| 6 | 9 | 39 | 47 | 7 | 36 | 42 |
| 7 | 10 | 48 | 57 | 7 | 43 | 49 |
| 8 | 13 | 58 | 70 | 8 | 50 | 57 |
| 9 | 15 | 71 | 85 | 8 | 58 | 65 |
| 10 | 19 | 86 | 104 | 9 | 66 | 74 |
| 11 | 24 | 105 | 128 | 10 | 75 | 84 |
| 12 | 31 | 129 | 159 | 11 | 85 | 95 |
| 13 | - | - | - | 13 | 96 | 108 |
| 14 | - | - | - | 15 | 109 | 123 |
| 15 | - | - | - | 17 | 124 | 140 |
| 16 | - | - | - | 19 | 141 | 159 |

Table 107: Band structure for WB Normal mode frames

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Normal mode | | | | | | |
|  | 13.2 kbps | | | 16.4 kbps | | |
| *b* |  |  |  |  |  |  |
| 0 | 6 | 0 | 5 | 6 | 0 | 5 |
| 1 | 6 | 6 | 11 | 6 | 6 | 11 |
| 2 | 6 | 12 | 17 | 6 | 12 | 17 |
| 3 | 6 | 18 | 23 | 6 | 18 | 23 |
| 4 | 6 | 24 | 29 | 6 | 24 | 29 |
| 5 | 7 | 30 | 36 | 6 | 30 | 35 |
| 6 | 7 | 37 | 43 | 7 | 36 | 42 |
| 7 | 8 | 44 | 51 | 8 | 43 | 50 |
| 8 | 10 | 52 | 61 | 8 | 51 | 58 |
| 9 | 11 | 62 | 72 | 9 | 59 | 67 |
| 10 | 13 | 73 | 85 | 11 | 68 | 78 |
| 11 | 16 | 86 | 101 | 12 | 79 | 90 |
| 12 | 19 | 102 | 120 | 14 | 91 | 104 |
| 13 | 24 | 121 | 144 | 17 | 105 | 121 |
| 14 | 30 | 145 | 174 | 20 | 122 | 141 |
| 15 | 37 | 175 | 211 | 23 | 142 | 164 |
| 16 | 47 | 212 | 258 | 28 | 165 | 192 |
| 17 | 61 | 259 | 319 | 34 | 193 | 226 |
| 18 | - | - | - | 42 | 227 | 268 |
| 19 | - | - | - | 51 | 269 | 319 |

Table 108: Band structure for SWB and FB Normal and Harmonic frames

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Normal and Harmonic mode | | | | | | |
|  | 13.2 kbps | | | 16.4 kbps | | |
| *b* |  |  |  |  |  |  |
| 0 | 6 | 0 | 5 | 6 | 0 | 5 |
| 1 | 6 | 6 | 11 | 6 | 6 | 11 |
| 2 | 6 | 12 | 17 | 6 | 12 | 17 |
| 3 | 6 | 18 | 23 | 6 | 18 | 23 |
| 4 | 6 | 24 | 29 | 6 | 24 | 29 |
| 5 | 6 | 30 | 35 | 6 | 30 | 35 |
| 6 | 7 | 36 | 42 | 7 | 36 | 42 |
| 7 | 8 | 43 | 50 | 7 | 43 | 49 |
| 8 | 9 | 51 | 59 | 8 | 50 | 57 |
| 9 | 10 | 60 | 69 | 9 | 58 | 66 |
| 10 | 11 | 70 | 80 | 10 | 67 | 76 |
| 11 | 13 | 81 | 93 | 11 | 77 | 87 |
| 12 | 16 | 94 | 109 | 13 | 88 | 100 |
| 13 | 19 | 110 | 128 | 15 | 101 | 115 |
| 14 | 23 | 129 | 151 | 18 | 116 | 133 |
| 15 | 28 | 152 | 179 | 21 | 134 | 154 |
| 16 | 34 | 180 | 213 | 26 | 155 | 180 |
| 17 | 42 | 214 | 255 | 32 | 181 | 212 |
| 18 | 55 | 256 | 310 | 39 | 213 | 251 |
| 19 | 68 | 311 | 378 | 48 | 252 | 299 |
| 20 | 84 | 379 | 462 | 59 | 300 | 358 |
| 21 | 105 | 463 | 567 | 74 | 359 | 432 |
| 22 | - | - | - | 92 | 433 | 524 |
| 23 | - | - | - | 115 | 525 | 639 |

##### 5.3.4.1.3 Energy Envelope coding

Energy envelope coding module is applied for all types of signal i.e., from NB, WB, SWB and FB for various bitrates as described in table 101. In this module, the spectrum energy of a band is computed the differential indices of scalar quantized band energies are encoded using either a Large symbol coding method or a Small symbol coding method. The coding method is selected according to the range required to represent all the differential indices and the bit consumption. The encoding of band energies is detailed below.

The spectrum energy of a band, *EM(b)* is computed as follows:

 (1029)

In case of transient mode, the energies to be quantized are first reordered such that energy corresponding to even sub-frame index *m* = 0, 2 are in frequency-increasing order while the energy of odd sub-frame index *m* = 1, 3 are in frequency decreasing order which allows for an efficient differential energy encoding

In each frame, the energies are scalar quantized with a uniform scalar quantizer *qint*. The value of *qint* varies and it is selected based on table 109.The index of the quantized energy, *IM*(*b*), can easily be obtained as:

 (1030)

Table 109: Scalar quantizer values for NB, WB, SWB, FB modes

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BW | Mode | 7.2 kbps | 8 kbps | 13.2 kbps | 16.4kbps |
| NB | Transient | 1.8 | 2.2 | 1.4 | - |
| Normal | 1 | 1 | 0.8 | - |
| WB | Transient | - | - | 1.8 | 1.8 |
| Normal | - | - | 0.8 | 0.8 |
| SWB, FB | Transient | - | - | 3 | 1.2 |
| Normal, Harmonic | - | - | 0.6 | 0.6 |

The quantized indices of the band energies are differentially coded by computing

 (1031)

where lowest frequency band quantized index is differentially coded using a reference band energy .

The differential indices ∆*IM*(*b*) are constrained into the range of [–256, 255]. This is performed by first adjusting the negative differential indices and then adjusting the positive differential indices as follows:

 (1032)

The constrained differential indices are used for selecting the more efficient mode from either the Small symbol coding method or the Large symbol coding method. The method selection between the Small symbol coding mode and the Large symbol coding mode is described in subclause 5.3.4.1.3.2.

Using the coding method information obtained from subclause 5.3.4.1.3.2 differential indices are coded using the respective modes as indicated below.

A flag bitDENG\_CMODE which was obtained from subclause 5.3.4.1.3.2 used to indicate the type of encoding method between the Small symbol coding method and the Large symbol coding method and transmitted as side information to the decoder. The flag DENG\_CMODE is set to 1 when the Small coding method is used and it is set to zero for the Large symbol coding method and it is described in table 110.

If the flag DENG\_CMODE is set to 1, in the Small symbol coding the band energies are either coded by resized or context based Huffman coding. A flag bitLC\_MODE is used to indicate the mode selection between resized or context based Huffman coding and it is transmitted as side information to the decoder. The mode selection between resized and context based coding is done based on the estimated number of bits consumed by the respective coding modes. The resized Huffman coding described in subclause 5.3.4.1.3.3.2 is used for coding the differential indices when LC\_MODE is set to 1 and LC\_MODE is set to 0 for coding the differential indices using context based coding which is described in subclause 5.3.4.1.3.3.1.

5.3.4.1.3.1 Reconstruction of quantized energies

The quantized differential indices are reconstructed according to

 (1033)

where is the reference band energy The final resulting reconstructed quantized energies are obtained as follows

 (1034)

where the value of *qint* varies and it is selected based on table 109.

If band energies are quantized under transient mode i.e., *IsTransient* is True, the energies are reordered back to original.

5.3.4.1.3.2 Energy envelope coding mode selection

The differential quantization indices are encoded by one of two coding methods. The coding method is selected according to the range required to represent all the differential indices and the bit consumption. The range of the large symbol coding method enables it to represent larger number of bits. The large symbol coding method consists of a scale mode and a pulse mode. The small symbol coding method uses an upper bit coding method which consists of a context based Huffman coding mode and a re-sized Huffman coding mode as well as bit packing for the lower bit.

Table 110: Low-rate HQ envelope coding modes

| Coding Method index  *DENG\_CMODE*  (1bit) | Coding Method | Coding Mode index  (1bit) | Description |
| --- | --- | --- | --- |
| 0 | Large symbol method | 0 | Pulse mode |
| 1 | Scale mode |
| 1 | Small symbol method | 0 | Context based Huffman coding mode |
| 1 | Re-sized Huffman coding mode |

If at least one of the differential quantization indices in all the bands of a frame cannot be represented in [-32, 31]([-46,17] for the first index), the large symbol coding method is always used. If this larger range is not required the bits consumption for both large and small symbol coding modes are compared and the coding mode with the least bits is selected. The corresponding coding method information is transmitted for each frame.

The Small symbol coding method and the large symbol method are used for estimating the bits consumption for coding the differential indices which are obtained in equation (1031). The detailed descriptions of the respective coding modes are given in the following subclause 5.3.4.1.3.3.

The number of estimated bits hcode1 and flag LCmode information obtained from subclause 5.3.4.1.3.3. is used for selecting the best mode with the Large symbol coding method bits ULbits obtained from subclause 5.3.4.1.3.4, as shown below.



5.3.4.1.3.3 Small symbol coding method

If *IsTransient* is *True*,

In this module, the quantized indices obtained from equation (1030) are constrained into the range of [–15, 16] by calculating the differences as in equation (1031) and constraining the range. This is performed by first adjusting the negative differential indices and then adjusting the positive differential indices as follows:

1. Compute the differential indices for lowest frequency band using reference band energy as in equation (1031)
2. For b = 0, if  and 
3. For the rest of the bands, compute the differential indices defined in equation (1031) in order from the highest-frequency band to the lowest-frequency band.
4. If 
5. Re-compute the differential indices in order from the lowest-frequency sub-vector from band b=1 to the highest-frequency sub-vector.
6. If 
7. The adjusted differential indices in the range [0, 31] are obtained by adding an offset of 15 to *∆IM*(*b*).

Context based Huffman coding mode is used for estimating the bit consumption, if the range of differential indices lies in between [10, 22] resized Huffman coding [b-Huffman] mode is enabled for estimating the bits consumption. The Huffman codes for the differential indices for resized Huffman coding mode when *IsTransient* is *True* are given in table 111. In the table 111, *Hi* denotes the index of the Huffman code, *Hc* is the Huffman code corresponding to index *Hi* and *Hb* denotes the bits required for representing the Huffman code corresponding to index *Hi*.

Table 111: Huffman code for Transient frames

| Hi | Hc | Hb | Hi | Hc | Hb | Hi | Hc | Hb | Hi | Hc | Hb |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 8 | 0 | 0 | 16 | 11 | 2 | 24 | 0 | 0 |
| 1 | 0 | 0 | 9 | 0 | 0 | 17 | 0010 | 4 | 25 | 0 | 0 |
| 2 | 0 | 0 | 10 | 0 | 0 | 18 | 011010 | 6 | 26 | 0 | 0 |
| 3 | 0 | 0 | 11 | 1111010 | 7 | 19 | 00111010 | 8 | 27 | 0 | 0 |
| 4 | 0 | 0 | 12 | 01010 | 5 | 20 | 010111010 | 9 | 28 | 0 | 0 |
| 5 | 0 | 0 | 13 | 110 | 3 | 21 | 110111010 | 9 | 29 | 0 | 0 |
| 6 | 0 | 0 | 14 | 01 | 2 | 22 | 0 | 0 | 30 | 0 | 0 |
| 7 | 0 | 0 | 15 | 00 | 2 | 23 | 0 | 0 | 31 | 0 | 0 |

The estimated bits for context based coding which was obtained from subclause 5.3.4.1.3.3.1 is represented as *,*whilefor resized coding it is represented as,a best coding mode is selected based on

 .

If *IsTransient* is *False*,

The differential quantization indices are adjusted to have positive values by adding 46 in the first band and 32 in the other bands. The differential quantization indices are split into 5 upper bits and 1 lower bit. The 5 upper bits are encoded by either a context based Huffman coding mode described in subclause 5.3.4.1.3.3.1 or a re-sized Huffman coding mode described in subclause 5.3.4.1.3.3.2 and the 1 lower bit is packed.

In more detail, the context based coding mode or the resized Huffman coding is used for estimating the bits. The differential indices which are obtained in equation (1031) ∆*IM*(*b*) are constrained into the range of [0, 63] by adding an offset of 32 to *∆IM*(*b*) for *b* = 1,….,*Nbands*-1 and for *b*=0 an offset of 46 is used for *∆IM*(*0*).If the constrained differential indices exceed [0 63] when *IsTransient* is *False* and [0 31] when *IsTransient* is *True, hcode1* is set to -1 and the differential indices are coded using the Large symbol coding method.

The least significant bit is extracted from the constrained differential indices *∆IM*(*b*) for *b* = 0,….,*Nbands*-1 using

 (1035)

The updated differential indices are used for estimating the bits consumed by the two different coding modes. Based on the estimated bits obtained from context based coding, , and resized Huffman coding mode,  , the best coding mode is selected as shown below.



The differential index ∆*IM*(*0*) is usually transmitted as is for both *IsTransient* is *True* or *False*, i.e., 5 bits per norm index is required and these bits are updated to the estimated bits which was shown below.

 (1036)

The least significant bit extracted from the constrained differential indices is transmitted as is, if the Small symbol coding method is selected and is updated as shown below.

 (1037)

By default flag LCmode is set to 0 and represents context based coding, while if flag LCmode is reset to 1, it indicates resized Huffman coding mode consumes less bits compared to context based coding, and the estimated bits is used for selecting the best mode with the Large symbol coding method bits ULbits.

5.3.4.1.3.3.1 Context based Huffman coding mode

If the this coding mode is selected for the current frame, the context based Huffman coding is applied to the adjusted differential indices. These indices are encoded using a context model which corresponds to the adjusted differential index in a previous band. The first band must be handled separately, so the context for encoding is adjusted by subtracting 

 (1038)

There are three groups depending on the and two probability models as shown in table 112,  and share the Huffman tables depending on the probability model.

Table 112: The groups and the probability models

| Group index | Lower bound | Upper bound | Probability model |
| --- | --- | --- | --- |
| 0 | - | 12 | 0 |
| 1 | 13 | 17 | 1 |
| 2 | 18 | - | 0 |

A Huffman table for  and is defined in table 113 and a Huffman table for is defined in table 114.

If is located in, is reversed toand then the reversed value is encoded.

Table 113: Huffman coefficient table for the group0 Huffman coding (group0,group2)

| Index | Code | Index | Code | Index | Code | Index | Code |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 11100111101111 | 8 | 11100110 | 16 | 000 | 24 | 111011 |
| 1 | 11100111101110 | 9 | 1110100 | 17 | 010 | 25 | 0110011 |
| 2 | 1110011110110 | 10 | 0110010 | 18 | 110 | 26 | 1110010 |
| 3 | 111001111010 | 11 | 100100 | 19 | 0111 | 27 | 1110101 |
| 4 | 1001010001 | 12 | 01101 | 20 | 1111 | 28 | 1001011 |
| 5 | 1110011111 | 13 | 1000 | 21 | 10011 | 29 | 10010101 |
| 6 | 111001110 | 14 | 101 | 22 | 011000 | 30 | 1001010000 |
| 7 | 100101001 | 15 | 001 | 23 | 111000 | 31 | 11100111100 |

Table 114: Huffman coefficient table for the context based Huffman coding (group1)

| Index | Code | Index | Code | Index | Code | Index | Code |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0010000100110 | 8 | 001000101 | 16 | 11 | 24 | 0010000101 |
| 1 | 001000100110 | 9 | 00100110 | 17 | 100 | 25 | 00100001000 |
| 2 | 00100111010 | 10 | 0010010 | 18 | 1011 | 26 | 001000010010 |
| 3 | 00100010010 | 11 | 101000 | 19 | 10101 | 27 | 00100111011 |
| 4 | 00100010001 | 12 | 00101 | 20 | 101001 | 28 | 001000100111 |
| 5 | 00100010000 | 13 | 0011 | 21 | 00100000 | 29 | 00100001001110 |
| 6 | 0010011100 | 14 | 000 | 22 | 00100011 | 30 | 001000010011110 |
| 7 | 001001111 | 15 | 01 | 23 | 001000011 | 31 | 001000010011111 |

5.3.4.1.3.3.2 Resized Huffman coding mode

Resized Huffman coding is applied to the adjusted differential indices. In this method, the span of the differential indices is reduced while being able to perfectly reconstruct the differential indices. Based on the newly modified differential indices obtained from equation (1040), number of bits consumed for coding the new differential indices  is estimated as shown below.

 (1039)

5.3.4.1.3.3.3 Differential Indices Modification

The modification of differential indices is done according to the value of the differential index for the preceding sub band and a threshold. Equation (1040) is used for modifying the span of differential indices. It should be noted that this modification is not applied to the first differential index, i.e. the case of "b = 1" and the differential indices which are not true for the both of the “if” conditions.

(1040)



Based on the new differential indices obtained from equation (1040), resized Huffman coding is applied, if any of the new differential indices lies outside [0 31] range, resized Huffman coding is not used for coding the differential indices.

The range of the new differential indices for Huffman coding is identified as shown below.

 (1041)

where *b* = 1, ... *Nbands*-1

Based on the range obtained from equation (1041), range difference is calculated as shown below.

 (1042)

Resized Huffman coding is used for coding the new differential indices if the *RangeDiff* values lies below 11, otherwise resized Huffman coding is not used. The Huffman codes and its corresponding bits consumption for coding of the new differential indices are given in table 115.

Table 115: Huffman code for Non-Transient frames

| Hi | Hc | Hb | Hi | Hc | Hb | Hi | Hc | Hb | Hi | Hc | Hb |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 8 | 001111111 | 9 | 16 | 10 | 2 | 24 | 1011111111 | 10 |
| 1 | 0 | 0 | 9 | 00111111 | 8 | 17 | 101 | 3 | 25 | 1111111111 | 11 |
| 2 | 0 | 0 | 10 | 0011111 | 7 | 18 | 1011 | 4 | 26 | 0 | 0 |
| 3 | 0 | 0 | 11 | 001111 | 6 | 19 | 10111 | 5 | 27 | 0 | 0 |
| 4 | 0 | 0 | 12 | 00111 | 5 | 20 | 101111 | 6 | 28 | 0 | 0 |
| 5 | 01111111111 | 11 | 13 | 0011 | 4 | 21 | 1011111 | 7 | 29 | 0 | 0 |
| 6 | 0111111111 | 10 | 14 | 001 | 3 | 22 | 10111111 | 8 | 30 | 0 | 0 |
| 7 | 0011111111 | 10 | 15 | 00 | 2 | 23 | 101111111 | 9 | 31 | 0 | 0 |

5.3.4.1.3.4 Large symbol coding method

If the large symbol coding method is used then either the pulse mode or the scale mode is selected to encode the differential quantization indices. The pulse mode is adequate when no differential quantization index is over [-4,3]. If this range is exceeded, the pulse mode cannot be used, and instead the scale mode is always used. Additionally, if the first quantization differential index is over [-64,63], the scale mode is always used. In the large symbol coding method a Huffman coding with 8 symbols shown in table 116 is used.

Table 116: Huffman coefficient table in the large symbol coding method

| Index | Code |
| --- | --- |
| -4 | 0001011 |
| -3 | 00011 |
| -2 | 001 |
| -1 | 01 |
| 0 | 1 |
| 1 | 0000 |
| 2 | 000100 |
| 3 | 0001010 |

5.3.4.1.3.4.1 Pulse mode

In the pulse mode, there are two indicators; an indicator to show whether the first index is transmitted separately and an indicator  to show if there is a differential quantization index exceeding the range [-4,3].

If the first index is within [-4,3], is set to 0 and the first index is then encoded by the Huffman coding defined in table 116 with the other indices. Otherwise, is set to 1 and the first index is then packed using 7 bits after adding 64.

If a pulse exists in the current frame, is set to 1 and the pulse position and amplitude are transmitted using 5 bits and 7 bits respectively. All the other indices are then encoded by the Huffman coding in table 116. If no pulse exists, all indices are encoded by the Huffman coding in table 116.

Table 117: bit allocation for the scale mode

|  |  |  |  |  |  |  |  | Huffman bits |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| bits | 1 | 1 | 1 | 1 | 7 | 5 | 7 | - |

5.3.4.1.3.4.2 Scale mode

In the scale mode, all the indices are split into 3 upper bits and a few lower bits depending on the minimum and maximum of all the indices. The 3 upper bits are encoded by the Huffman coding in table 116 and the lower bits are packed. The number of lower bits is defined as . Theis calculated to make all the differential quantization indices fit within the range [-4,3] by scaling down the indices, and is represented by three bits.

##### 5.3.4.1.4 MDCT coefficients quantization

5.3.4.1.4.1 Normal Mode

5.3.4.1.4.1.1 Overview

The figure below shows the overview of the normal mode encoder.

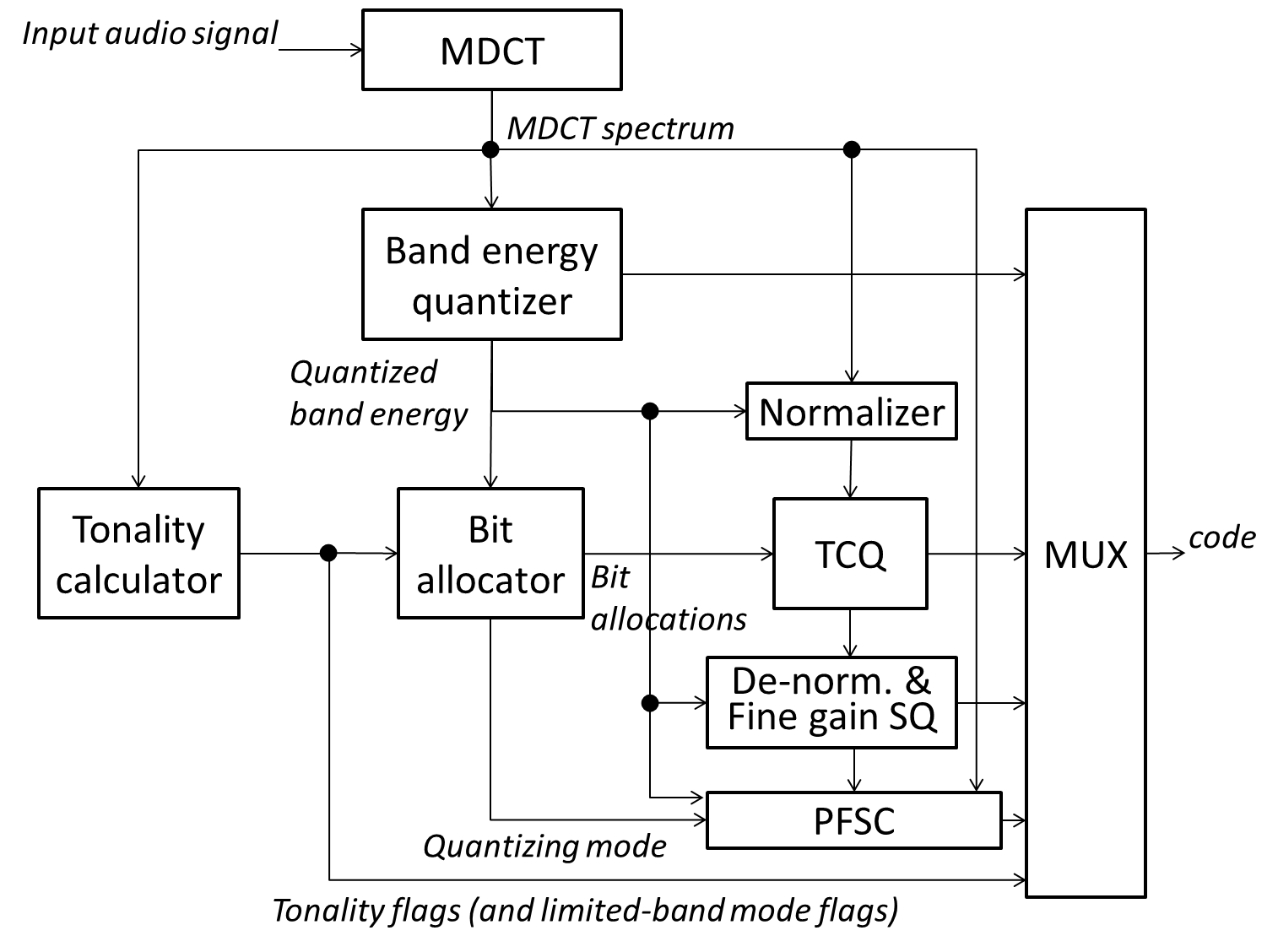


Figure 66: Block diagram of the Normal mode encoder overview

5.3.4.1.4.1.2 Energy envelope coding

Details are described in subclause 5.3.4.1.3.

5.3.4.1.4.1.3 Tonality flag calculation

Tonality flags are calculated for the high bands *b*= *Nbands*- *hb*, ..,*Nbands*-1 as described in table 118. For example, for the last (highest) five bands, i.e. b=17 to 21 for 13.2 kbps and b=19 to 23 for 16.4kbps in table 108, and the last three bands b=15 to 17 for 13.2 kbps and b=17 to 19 for 16.4kbps in case of WB as in table 107 and last two bands b=15 to 16 for 13.2 kbps for NB inputs as in table 106, peak-to-average ratios are calculated and compared with a threshold. Flags indicating whether the peak-to-average ratios are greater than the threshold are sent to the decoder side using one bit per band. If the peak-to-average ratio is greater than the threshold, the tonality flag is set to “1”, otherwise it is set to “0”. For the SWB and FB case, a limited-band mode flag is further sent to the decoder side as described in subclause 5.3.4.1.4.1.4.4.2, if the tonality flag is set to “1”.

Table 118: Total number of high bands for tonality calculation

|  |  |  |
| --- | --- | --- |
| Bandwidth | Bitrate (kbps) | High bands , *hb* |
| NB | 13.2 | 2 |
| WB | 13.2,16.4 | 3 |
| SWB, FB | 13.2,16.4 | 5 |

5.3.4.1.4.1.4 Bit allocation

5.3.4.1.4.1.4.1 Bit allocation overview

In the Normal Mode, band spectra are encoded by either Trellis Coding Quantization (TCQ) or Pitch Filtering Spectrum Coding (PFSC) with assigned bits for the bands. TCQ is used for encoding peaky/tonal spectrum bands for NB, WB, SWB and FB, while PFSC is mainly applied for encoding other spectrum bands in a high-frequency region other than NB. This switching principle is analogous to the coding mode switching between Generic mode and Sinusoidal mode in G.718 Annex B [25].

Bit-allocation process is performed in the following manner. Firstly, bands encoded using PFSC are identified based on the tonality flags among the four highest bands in case of SWB ,FB and peak-to-average ratio is calculated for last band in the WB and necessary bits (1 or 2 bits) are allocated to each of the identified bands. Secondly, remaining bits are allocated to other bands based on perceptual importance. When there is any band whose assigned bit results is zero in the four bands for SWB and FB signals, such band is re-identified as a PFSC encoding band and the bit allocations are re-calculated.

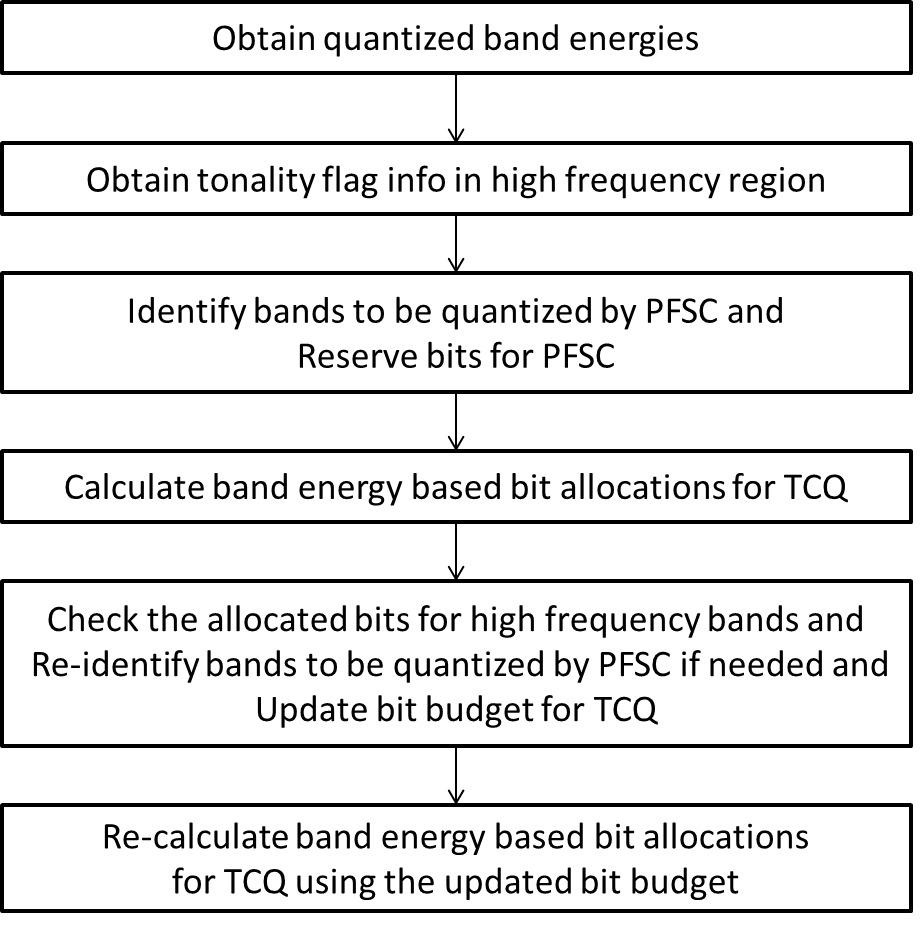


Figure 67: A flowchart of bit allocation processing for LR-HQ Normal mode.

5.3.4.1.4.1.4.2 The adjustment of quantized energy envelope prior to bit allocation

For the Non-Transient mode of NB (the bit rate is less than or equal to 13.2kbps) and WB cases, in order to make the inter-frame reconstruction more continuous and allocate more bits to perceptual important bands, the quantized energy envelopes of the highest  bands are adjusted prior to bit allocation. Some of the quantized energy envelopes of the high frequency bands and low frequency bands are adjusted. Then perform the bit allocation to bands according to the adjusted energy envelopes. Finally, the coefficients of the bit allocated bands are quantized and written to the bit-stream.

To adjust the quantized energy envelope of high frequency bands at NB, the following steps are performed:

1. Two bits are encoded to indicate whether the highest two bands of the previous frame is encoded. Initialize a band boundary  to 6 and initialize adjustment factors , respectively:

 (1043)

1. For each band, calculate the magnitude envelope:

 (1044)

where  denotes the bandwidth of each band, and is computed as follows:

 (1045)

where  denotes the quantized energy envelope of each band.

1. Then, calculate the sum of the differences between the consecutive two magnitude envelopes and the sum of the magnitude envelopes for high frequency bands:

 (1046)

 (1047)

1. Search the peak of magnitude envelopes and calculate the sum of the magnitude envelopes for low frequency bands:

 (1048)

 (1049)

1. Initialize an adjustment factor  for each band to 1, and then adjust the quantized energy envelopes of high frequency bands as follows:
2. if is satisfied at the bit-rates below 13.2kbps or

is satisfied at 13.2kbps,

the quantized energy envelopes of the last  bands are adjusted as follows:

 (1050)

1. Otherwise, the adjustment factors of the last 2 bands are calculated:

 (1051)

Then, the adjustment factors of the last 2 bands are updated further according to :

 (1052)

Finally, the quantized energy envelopes of the last 2 bands are adjusted: ,

where  is the tonality flag which is calculated in subclause 5.3.4.1.4.1.3, i.e. the classification mode of the band, and is obtained from the previous frame and indicates whether the bits are allocated to the highest two bands of the previous frame or not. If is equal to 1, the band of the previous frame is allocated bits; Otherwise, if is equal to 0, the -band of the previous frame is not allocated bits -. After bit allocation, the flag of the current frame is preserved for the next frame.

To adjust the quantized energy envelops of low frequency bands at NB, the following steps are performed:

1. Initialize  to 3, select low frequency bands from bands. For each band, calculate the magnitude envelope:

 (1053)

1. Then, calculate the sum of the magnitude envelopes for high frequency bands:

 (1054)

the variable is assigned to different value for different bit rates, specifically it is assigned to 13,14,15,17 for 7.2kbps, 8kbps, 9.6kbps and 13.2kbps respectively.

1. Search the peak of magnitude envelopes and calculate the sum of the magnitude envelops for low frequency bands:

 (1055)

 (1056)

1. Initialize a flag  to 0.  is a flag to indicate whether second stage bit allocation algorithm is used in TCQ module. When  is equal to 0, second stage bit allocation algorithm will be used. When  is equal to 1, the second stage bit allocation algorithm will not be used. The flag is utilized in subclause 5.3.4.1.4.1.5.1.2.
2. Determining whether to modify the quantized energy envelops of the six low frequency band according to their energy characteristics and spectral characteristics. The energy characteristics denote the ratio between the energy of six low frequency bands and the energy of the other bands which are determined by  and ; The spectral characteristics denote the degree of spectrum fluctuation which are determined by.
3. If the following conditions are satisfied,

, the flag  is set to 1, and the quantized energy envelops of the three low frequency bands are adjusted as follows:

 (1057)

To adjust the quantized energy envelopes of high frequency bands at WB, the following steps are performed:

1. Encode two bits to indicate whether the highest two bands of the previous frame is encoded. Define two band boundaries and. If the bit rate is 13.2kbps, set and to 8 and 15, respectively; Otherwise, set and to 8 and 16, respectively. The bandwidths of low frequency bands and high frequency bands are obtained as follows:

 (1058)

1. For each band, calculate the magnitude envelope:

 (1059)

Then, calculate the sum of the differences between the consecutive 2 magnitude envelopes and the sum of the magnitude envelopes for part of the high frequency bands:

 (1060)

 (1061)

1. The energies of low frequency bands and high frequency bands are computed as follows:

 (1062)

1. Obtain adjustment factors  for the highest  bands according to the tonality flag and the energies of low frequency bands and high frequency bands:

 (1063)

and for the highest  bands, update the adjustment factors  according to :

 (1064)

and then adjust the quantized energy envelopes of the highest  bands .

To adjust the quantized energy envelopes of low frequency bands at WB, the following steps are performed:

1. Initialize a low frequency band boundary  to 6. For each band, calculate the magnitude envelope:

 (1065)

1. Then, calculate the sum of the magnitude envelopes for high frequency bands:

 (1066)

the variableis assigned to different value for different bit rate, specifically it is assigned to 18, 20 for 13.2kbps and 16.4kbps respectively.

1. Search the peak of magnitude envelopes and calculate the sum of the magnitude envelops for low frequency bands:

 (1067)

 (1068)

1. Determining whether to modify the quantized energy envelops of the six low frequency band according to their energy characteristics and spectral characteristics. The energy characteristics denote the ratio between the energy of six low frequency bands and the energy of the other bands which are determined by  and ; The spectral characteristics denote the degree of spectrum fluctuation which are determined by .

If the conditions  or are satisfied, the flag, and the quantized energy envelops of the six low frequency bands are adjusted as follows:

 (1069)

Finally, the adjusted quantized energy envelopes and the initial quantized energy envelopes are used in the first bit allocation module.

5.3.4.1.4.1.4.3 Bit allocation for PFSC

Based on the tonality flags, whether TCQ is used for encoding the band in the high frequency region is determined. When the tonality flag is set to “0”, such band is excluded from target bands of the TCQ, i.e. no bit is assigned to the band for TCQ.

In the Normal Mode, the four bands in the high-frequency region for SWB/FB and one band for WB are assumed to be quantized by TCQ in default operation. However, when the tonality flag is set to “0”, such band is quantized with the PFSC scheme using a similar procedure with the “sub-band search” (called as “band search” in this specification) in [25], whereas for WB cases such band is filled with noise. This means the peaky/tonal bands are encoded by the TCQ while the PFSC scheme is used for other bands.

In Figure 66, whether the PFSC is used for quantizing the high-frequency bands is indicated by ‘Quantizing mode’.

In the PFSC scheme, the band search is based on a pitch filter based prediction (, where *T* is a pitch coefficient, and low-frequency spectrum is used as its filter state (filter memory)), and the pitch coefficient (i.e. lag information as a filter parameter) is encoded. The lag information is encoded with 2 bits or 1 bit. Lower two bands among the four bands are encoded with 2 bits, while higher two bands among the four are encoded with 1 bit. When the PFSC is selected for encoding some of the four bands, necessary bits for encoding those bands are reserved and assigned for the bands before starting the bit-allocation process for TCQ encoding bands.

5.3.4.1.4.1.4.4 Bit allocation for TCQ

5.3.4.1.4.1.4.4.1 Allocating bits for fine gain adjustment

In the TCQ based MDCT coefficients quantization, fine gain adjustment is applied to several bands whose energies are larger than the others. The number of those bands is configured based on encoding bit-rate and signal bandwidth as shown in table 119.

Table 119: Bits reserved for fine gain adjustments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **7.2 kbit/s** | **8.0 kbit/s** | **13.2 kbit/s** | **16.4 kbit/s** |
| NB | 2 bands, | 2 bands, | 4 bands, | 2 bands, |
| WB | - | - | 6 bands, | 6 bands, |
| SWB | - | - | 4 bands, | 4 bands, |

One-bit scalar quantization is used for the fine gain adjustment. Therefore the number of reserved bits  for NB, WB, SWB, and FB is shown in the table 119. The bits for the TCQ are obtained by subtracting the fine gain bits, and expressed by, where is the available bit budget for spectrum coding, *ebits* is the bits consumed for quantizing the band energies which is obtained from subclause 5.3.4.1.3, and *R* is the total bits. It should be noted that the mode bits for switching Transient/non-Transient and Normal/Harmonic are included in the available bit budget. Therefore the necessary bits (1 or 2) are subtracted from the available bit budget in the following subclauses,   
i.e.  (NB and WB cases) or  (SWB and FB cases).

5.3.4.1.4.1.4.4.2 Limited-band mode

For the SWB case, bandwidth, , is more than 50 bins for the last four bands (i.e. *b*=18 to 21 in 13.2 kbps and *b*=20 to 23 in 16.4 kbps). At low bit-rates, such wide bandwidth would result in insufficient quantization performance. Therefore, a limited-band mode is introduced for achieving efficient encoding. In the limited-band mode, only the vicinity of a perceptually important spectrum in each band is targeted to be quantized, i.e., outside the vicinity in each band is not targeted to be quantized. This is realized by the following principle. A bandwidth is adaptively shortened if the maximum amplitude spectrum frequency falls into a range of frequencies around the maximum amplitude spectrum frequency in the previous frame. The difference between the maximum amplitude spectrum frequencies for the current and previous frame is calculated. When it is smaller than a threshold, the limited-band mode is used for such band. The frequency position of the maximum amplitude spectrum is searched for each of the four bands. The position for the previous frame is stored in a memory, which was searched using the quantized MDCT spectrum (i.e. decoded MDCT spectrum) in the previous frame. The position for the current frame is searched using the MDCT spectrum calculated from the input signal in the current frame. When the limited-band mode is selected, the targeted band is limited to the vicinity of the maximum amplitude spectrum frequency of the previous frame. The range of the vicinity is 15 or 31 spectrum bins depending on coding bit-rate and band index. One bit is used for indicating whether the limited band mode is used when tonality flag is set to “1”.

The limited-band mode can be used only in the case where the corresponding previous band was quantized by TCQ.

5.3.4.1.4.1.4.4.3 Final bit allocation for TCQ and PFSC

Band widths and quantized band energies are used for bit allocation. The band widths are basically configured by Table 108. When the limited-band mode is selected, corresponding band widths are shortened. Bit allocation for TCQ is as follows. Firstly, bits are distributed according to the quantized band energy. Secondly, if there is a band whose assigned bits are less than a minimum number of bits, no bit is assigned to the band and the assigned bits will be re-allocated to the other bands. Thirdly, if there is a noise-like band whose assigned bits are not sufficient in comparison with its bandwidth, no bit is assigned to the band and the assigned bits will be re-allocated to the other bands. Furthermore, assigned bits are compared with predefined band-based threshold, and no bit is re-assigned if the assigned bits are less than the threshold for the band.

In case no bit is assigned to any band among the highest four bands for a SWB and FB signals, PFSC is used for quantizing such band. This case can happen when the tonality flag is set to “1” but assigned bit results in zero. In this case, necessary bits for the PFSC encoding for the band is extracted from the bit budget for TCQ,  . Therefore  is updated by subtracting the necessary bits, and the bit-allocation is re-calculated.

5.3.4.1.4.1.5 Fine structure encoding

5.3.4.1.4.1.5.1 Trellis Coding Quantization (TCQ)

5.3.4.1.4.1.5.1.1 Joint USQ and TCQ

Trellis Coded Quantization (TCQ) quantizes the fine structure of normalized spectrum, selecting the Important Spectral Components (ISCs). The information for the selected ISCs in each band is coded as the position, number, sign and magnitude of the ISCs. The magnitude information is quantized by the joint Uniform Scalar Quantization (USQ) and TCQ with arithmetic coding, while the information on position, number and sign is coded by arithmetic coding. A block diagram of the fine structure encoding using TCQ is depicted by the figure 68.



Figure 68: Block diagram of fine structure encoding using TCQ

The encoding method is selected at the Selecting Encoding Method block by the bit allocation and the information for each band. If a bit allocated for a band is zero, all the samples in that band are coded to zero by the zero encoding block. Otherwise, each band is quantized by the selected quantizer.

The quantizer selection information selects the most efficient quantizer between the USQ and TCQ by considering the input signal characteristics, i.e. the bit allocation and the length of each band. If the average number of bits for each sample in a band is greater or equal to 0.75, the band is of high importance and USQ is used; for all other bands TCQ is used. The quantizer selection flag, USQ\_TCQ[i], is set to 1 when USQ is used.

The Scaling Bands block performs scaling at each band to control the bit rate, using the bit allocations and the normalized spectrum for each band. The scaling is done by considering the average bit allocation for each spectral component in the band. If the average bit allocation is bigger than the number specified by the bit allocation, then more scaling is done.

The detailed scaling process is as follows. First the estimation of the number of pulses for the current band is obtained using the length and the bit allocation information for each band. Then the number of nonzero positions is obtained by the following equation, which is based on the probabilities.

 (1070)

where *b* is the number of bits and calculated as:

 (1071)

and the number of required bits for the positions is estimated as:

 (1072)

where *n* is the band length, *m* is the number of pulses, *i* is number of non-zero positions which have an ISC, and *b* is the number of bits required for the given size of band and number of pulses. Finally the number of pulses is selected by the *b* value which is the closest to the value of the allocated bits for the band.

The initial scaling factor is decided by the estimation of the number of pulses and the absolute value of the input signal. The input signal for each band is scaled by this factor. If the estimated number of pulses is not the same as the summation of the number of pulses for the quantized signal after scaling, a pulse redistribution process will be performed using the updated scaling coefficient. The redistribution process is as follows: if the number of selected pulses is smaller than the number of estimated pulses, the scaling coefficient will be decreased, otherwise the scaling coefficient will be increased.

The distortion function for the TCQ is sum of squared distance of each quantized and un-quantized value in each band. It is similar to the Euclidean distance but avoids the square root, since only the relative magnitudes of the values are needed rather than the exact distance.

 (1073)

where  is actual value and  is quantized value.

For the USQ module the Euclidean distance is used to determine the best quantized values. To minimize the computational complexity, the modified equation including the scaling factor is used, which is done by calculating the *d1* as:

 (1074)

If the number of pulses per band does not match the required value, it is necessary to add or delete some pulses while preserving the minimal metric. This procedure is done in an iterative manner by adding or deleting a single pulse, and then repeating until the number of pulses reaches the required value.

To add or delete one pulse it is necessary to calculate *n* values of distortions in order to select best one For example, the distortion value *j* corresponds to adding pulse at *j*-th position in a band:

 (1075)

To avoid the full calculation of this formula *n* times, the following derivation can be used:

 (1076)

where values can be calculated just once, *n* is the band length (number of coefficients in a band), *p* is the input signal of the quantizer , *q* is the quantized signal, and *g* is the scaling factor. Finally the position *j*, which minimized the distortion *d*, is selected and *qj* is updated.

To control the bit rate, an appropriate ISC has to be selected using the scaled spectral coefficients at the Selecting Important Spectral Components (ISCs) block. The spectral component to quantize is selected by using the bit allocations for each band. This selection can have various combinations which depend on the distribution and variance of the spectral component. The actual non-zero position is then calculated. The non-zero position is obtained by analyzing the amount of scaling and redistribution operations. The non-zero position will be the ISC.

If the number of pulses is not controlled by the scaling and the redistribution operations, the selected pulse will be quantized by TCQ and the surplus adjusted with the results of the quantization. Hence, if the number of non-zero position is greater than 1, and not equal to the estimated number of the pulse, and the quantizer selection information indicates the TCQ, the surplus redistribution process will be used. If this condition is satisfied, the surplus will be redistributed by the TCQ quantization operation in advance. If the real number of the pulses from the TCQ quantization is smaller than the estimated number of pulses which was derived for each band in advance, the scale factor will be multiplied by 1.1. However, if the number of pulses from the TCQ quantization is bigger than the estimated number of pulses, the scale factor will be multiplied by 0.9.

The final selected non-zero position is called by ISC whose information is encoded at the Encoding Position Info block. The information consists of the number of the selected ISC and the non-zero positions. In order to enhance the efficiency of the information encoding arithmetic coding is used.

Given a stream of symbols and their probabilities, the arithmetic coder produces a space efficient bit-stream to represent these symbols and, given the bit stream and the probabilities, the arithmetic decoder reverses the process.

In the arithmetic coding algorithm two 16 bits integers are taken as the numerators of fractions called “low” and “high”.  These fractions have a common denominator equal to, such that all the fractions fall in the range . These integers define a range so that a single number stores all the symbols of the message. This number is saved bit by bit to the bit stream during the arithmetic coding process. To avoid precision loss, at each step of coding process renormalization is performed when difference between ‘low’ and ‘high’ is less than 0.5.

In the decoder, 16 bits will first be read and stored in the arithmetic decoder accumulator. The decoder can then replicate the coding process, but instead of encoding the values, it uses the bit-stream to produce symbols. These symbols will be reconstructed correctly, if their probabilities are the same in both the encoder and decoder.

In the Gathering ISCs block, the new buffer is constructed by the selected ISCs, as shown in figure 69. The zero-band and the position which is not selected are both excluded from this buffer.

In the Joint USQ and TCQ Coding block, the magnitudes of the gathered ISCs are quantized by the joint USQ and TCQ, and the quantized information is additionally coded by arithmetic coding. In order to enhance the efficiency of the arithmetic coding, the non-zero position and the number of ISCs is utilized for the arithmetic coding. The joint USQ and TCQ have two types of coding methods. One is TCQ and USQ with 2nd bit allocation for the NB and WB, and the other one is the LSB TCQ for USQ for the SWB and FB. These methods are described in subclause 5.3.4.1.4.1.5.1.2 TCQ and USQ with second bit allocation and subclause 5.3.4.1.4.1.5.1.3 LSB TCQ for USQ.



Figure 69: Concept for the gathering ISCs

In the Encode Signs block, the sign information of the selected ISC is coded by the arithmetic coding. In order to recover the quantized components, the position, sign and magnitude information is added to the quantized components to recover the real components at the Recovering Quantized Coefficients block. In this block, the zero is allocated to the zero positions.

In the Inverse Scaling Bands block, the inverse scaling of the quantized components is performed. The inverse scaling factor can be extracted by using the scaling factor in the Scaling Bands block. The inverse scaled signal is same level as that of the normalized input signal and the signal is the output of the TCQ quantization.

5.3.4.1.4.1.5.1.2 TCQ and USQ with second bit allocation

The following information is needed for TCQ and USQ with second bit allocation:

1. Determine the total number of bits which will be allocated to the corresponding bands to be processed in current frame in subclause 5.3.4.1.4.1.4.4.1.
2. In order to obtain the number of bits for each band by first bit allocation, implement the first bit allocation to the bands based on the total number of bits to be allocated in subclause 5.3.4.1.4.1.4.4.3.
3. Based on the number of bits for each band by first bit allocation, the first detailed scaling process in subclause 5.3.4.1.4.1.5.1.1 is implemented on each band which is allocated bits by first bit allocation. Then, the total number of redundant bits of the current frame and the number of pulses of each band are obtained.

The general quantization and coding scheme of the TCQ and USQ with second bit allocation consists of several main blocks: quantizer decision, TCQ quantizer, USQ quantizer, lossless coder, and second bit allocation. In the quantizer decision module the quantization mode of the current band is selected by using the results of the Selecting Encoding Method block. Then the selected quantizer quantizes the current band, and the lossless encoder based on the arithmetic coding transmits the data to the bit stream. The second bit allocation block distributes surplus bits from the previously coded bands. The second bit allocation procedure detects two bands that will be encoded separately.



Figure 70: Block diagram of TCQ and USQ encoding with second bit allocation

Based on the second bit allocation parameters, two bands are selected from those bands allocated bits during the first bit allocation that will be allocated more bits. The second bit allocation parameters include at least one of the total number of redundant bits and the characteristics of each band. The characteristics of each band include the harmonic characteristics and the bit allocation state of each band. The tonality flags for the bands of current frame which are calculated in subclause 5.3.4.1.4.1.3 represent the harmonic characteristics of each band, and whether the highest two bands of the previous frame is quantized represents the bit allocation state of each band. If the tonality flag is equal to one, the signal type of corresponding band is harmonic. Otherwise, the signal type of corresponding band is not harmonic. If the tonality flags are not all zero or any one of the highest two bands of the previous frame is quantized, the two bands to be quantized last will be finally selected in the highest few bands. Otherwise, they will be selected in other bands than the highest few ones.

The first of these two bands is selected using the analysis of the average number of bits per bin in the band by the first bit allocation:

For each band, calculate the average number of bits per bin as follows:

 (1077)

where *R(b)* is the number of bits allocated to each band by the first bit allocation based on the total number of bits in the subclause 5.3.4.1.4.1.4.4.3, and  is the bandwidth of each band.

In order to determine the first band, the average number of bits per bin of each band will be compared. And the bands should satisfy the following conditions:

 (1078)

where  is the tonality flag, and  indicates whether the highest two bands of the previous frame is quantized. If is equal to 1, the band of the previous frame is quantized, else if is equal to 0, the band of the previous frame is not quantized.

If the average number of bits per bin in band  is the least one in the bands which satisfy the above conditions, the band  will be determined as the first band for a second bit allocation.

When there are no bands satisfy the above conditions or the encoded bandwidth is NB, the first band for a second bit allocation will be selected during the bands which satisfy the following conditions:

 (1079)

If the average number of bits per bin in band  is the least one in the bands which satisfy the above conditions, the band  will be determined as the first band for a second bit allocation.

For the first band  for a second bit allocation, if  or, then the band  or the band  will be selected as the second band for a second bit allocation. Otherwise, for the band  and the band, if the average number of bits per bin of the band  is less than the band, the band  will be selected as the second band for a second bit allocation, if not, the band  will be selected as the second band for a second bit allocation.

Thus implement the second bit allocation on the two selected bands to be allocated bits once again. When the total number of redundant bits is replenished, the redundant bits are allocated to the two bands selected above, with the first one being allocated the majority, or all, of the surplus. The number of bits allocated to the two bands is obtained respectively and the proportion of the surplus allocated to each band is shown in the following:

When the encoded bandwidth is NB:

 (1080)

When the encoded bandwidth is WB:

 (1081)

where  denotes the bit surplus,  and  denotes the number of bits allocated to the first and the second band for a second bit allocation respectively.

At last, based on the number of bits allocated to the two selected bands by the first bit allocation and the number of bits allocated to the two selected bands by the second bit allocation, implement the second detailed scaling process on each of the two selected bands same as the first detailed scaling process in subclause 5.3.4.1.4.1.5.1.1. So, the number of pulses for each of the two selected bands is obtained again.

In the TCQ quantization module a trellis is used with 8 states, 4-coset (subsets) with 2 zero levels. The quantization indexes are derived from the TCQ codebook, which consists of the branch information of a trellis state (path information) and the information for quantization level allocated to the selected coset (subset). The quantization indexes are always positive integers and are coded by arithmetic coding. The detailed magnitude encoding is as follows.



Figure 71: Trellis structure with 8 states 4-coset with 2 zero levels

Each quantized band starts from state (0), because in other cases it is required to transmit two additional bits per band. Then, by using the Viterbi algorithm, the optimal path through trellis is selected, in order to have best possible SNR when comparing the original and quantized sequences. To calculate the SNR, it is necessary to have quantization scale before starting TCQ quantization, so the band is initially quantized by USQ and the optimal scale is calculated. After the lossless encoding the bit surplus is accumulated.

The accumulated surplus is used as an additional resource for the encoding of the last two bands determined by the second bit allocation procedure. Depending on the coding mode and surplus value, the surplus is shared between these two bands in different proportions. Then the encoding of these two bands is performed in the same way as described above.

The magnitude coding based on binary arithmetic coding is carried out as follows. First the probability of a symbol is calculated as

 (1082)

 (1083)

where  is the number of magnitudes left to transmit in the band,  is the number of pulses left to transmit in the band,  is the current coded pulse in magnitude and  is the set of existing magnitudes at trellis state .

Each magnitude pulse is encoded by using probabilities  and , where corresponds to last pulse in magnitude and  to all other pulses.

The magnitude pulse probabilities are modified after this calculation with respect to the trellis code limitation. This information is determined by the trellis structure and is available to both the encoder and decoder. Thus any magnitude values that are impossible are encoded using zero probability and hence do not require any bits.

The encoding algorithm was modified to save complexity for bands with a large number of pulses. The idea is to introduce a non-binary arithmetic coder, where the probabilities of a coded symbol are calculated as the mutual probability of the binary symbols for the current magnitude. To avoid very low probabilities and hence loss of precision, escape symbols are utilized. After an escape symbol transmitted, the rest of the magnitude pulses transmitted are started from initial probabilities condition.

The location coding is carried out based on same algorithm as used for the magnitude coding and uses the same complexity reduction technique. The signs and LSB path vector are transmitted as is, because the distributions are random.

5.3.4.1.4.1.5.1.3 LSB TCQ for USQ

The aim of the LSB TCQ for USQ is to use the advantages of both quantizers (USQ and TCQ) in one scheme and exclude the path limitation from the TCQ. Conceptually the LSB coding of the quantized data is shown in figure 72.



Figure 72: Concept of LSB coding

Each quantized value that is greater than one contains an LSB which can be zero or one. The sequence of LSBs can then be quantized by TCQ to find the best match between that sequence and the available trellis paths. In terms of the SNR criteria, it does not matter where the error occurs. Thus at the cost of some errors in the quantized sequence, the length of the sequence is decreased.

The encoding of the spectral data is done by the TCQ and USQ quantizers and lossless coding based on the binary arithmetic coder. Before processing the norms, the extraction and spectrum normalization are done. The combined quantizer scheme is presented at figure 73.



Figure 73: Block diagram for LSB TCQ for USQ encoding

The spectral data in each band are quantized by the USQ quantizer with the number on bits *R[]* determined by the bit allocation module. In order to fit the bit requirement, the number of bits which will be used for TCQ data is extracted from each non-zero band evenly, and then the bands are quantized by the USQ. The quantization procedure is the same as described above for the TCQ and USQ algorithm. All bands that have non-zero data after quantization are collected as the difference between the quantized and un-quantized data and are called the residual. If some frequencies are quantized as zero in a nonzero band, they are not included into residual.



Figure 74: Concept for constructing the residual array

The residual array is quantized by TCQ with code rate ½ known (7,5)8 code:

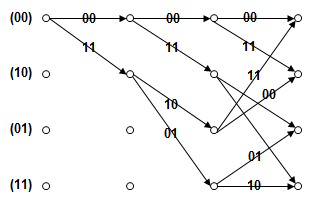


Figure 75: Trellis structure for the 4 states TCQ

Quantization using TCQ is performed for the first  magnitudes. After quantization the path metrics are checked and the best one selected. For lossless coding the data for the best trellis path is stored in separate array while the trace back procedure is performed. The constant  was defined as 10, which allows up to 20 magnitudes per frame to be encoded.

The trellis path data is encoded by the arithmetic encoder as equi-probable symbols. The path data is binary sequence which is encoded using an arithmetic encoder with a uniform probability model.

The quantized spectral data produced by the USQ are encoded by the same method as described in subclause 5.3.4.1.4.1.5.1.2

The quantized MDCT coefficients are de-normalized using the quantized band energies.

Finally, as described in subclause 5.3.4.1.4.1.4.4.1, quantization of fine gain adjustment is performed on the dominant bands. The inner product between target MDCT coefficients and the de-normalized quantized MDCT coefficients is calculated, and fine gain adjustment factor is calculated by dividing the inner product by the energy of the de-normalized quantized MDCT coefficients. The fine gain adjustment factor is quantized by a 1-bit scalar quantizer.

5.3.4.1.4.1.5.2 Noise-filling for 0-bit assigned sub-bands

Decode the MDCT coefficients of each sub-band from the received bit stream. In order to reconstruct the un-decoded MDCT coefficients, the sub-bands are classified into the bit allocation saturated sub-bands and the bit allocation un-saturated sub-bands according to the average number of bit allocation to a coefficient in a sub-band. The noise filling module is applied to the un-saturated sub-bands. Firstly the noise gain for noise filling of each sub-band is calculated. Then, fill the appropriate noise into the un-decoded MDCT coefficients of each sub-band, and the un-decoded MDCT coefficients are reconstructed. At last, the whole frequency domain signal is obtained based on the decoded MDCT coefficients and the reconstructed MDCT coefficients.

5.3.4.1.4.1.5.2.1 Parameters calculation for noise gain

Initialize a critical number of the sub-band  to. If the encoded bandwidth is SWB and the encoding mode is harmonic or normal,  is updated to 19 at 16.4kbps and  is updated to 17 at 13.2kbps.

Calculate the average number of allocated bits for each coefficient in each sub-band:

 (1084)

where  denotes the number of bits allocated to each sub-band, and  denotes the bandwidth of each sub-band.

Calculate the average envelope of each sub-band:

 (1085)

where  denotes the quantized energy of each sub-band.

The maximum magnitude of the decoded coefficients in each sub-band  and the number of the non-zero decoded coefficients in each sub-band  are determined by the decoded coefficients.

Suppose  denotes the number of the encoded pulses of each sub-band and  means the average number of bit allocation to a coefficient in a sub-band is larger than 0. And the energy difference  is obtained as follows:

 (1086)

If , compute the initial noise gain of each sub-band  by the energy difference  and sub-band bandwidth :

 (1087)

Otherwise, .

And, if , compute the parameter of harmonic character . The parameter of harmonic character  is used to calculate the noise gain, then fill noise to the not decoded coefficients of bit allocation un-saturated sub-band.

5.3.4.1.4.1.5.2.2 Noise gain calculation

If the average number of bit allocation to a coefficient in a sub-band  is not less than 0.8, the bit allocation of the sub-band is defined as saturated; otherwise, the bit allocation of the sub-band is defined as un-saturated.

When the bit allocation of sub-band  is un-saturated and , based on the envelope and decoded coefficients the noise gain  is calculated as follows:

(a) First estimate the overall noise factors under different conditions:

(a1) When  and the encoded bandwidth is SWB:

If the encoding mode is harmonic and  :

 (1088)

If the encoding mode is harmonic and  :

 (1089)

If the encoding mode is normal and  :

 (1090)

If the encoding mode is normal and  :

 (1091)

If the encoding mode is transient:

 (1092)

(a2) When  and the encoded bandwidth is WB or NB:

Initialize  as:

 (1093)

While  and,  is updated as follows:

 (1094)

While  and the encoded bandwidth is WB,  is updated as follows. Where  is a number of sub-bands at the highest frequency which have a flag  to indicate whether the sub-band is harmonic or not. If  is equal to 1, the sub-band have harmonic character; otherwise, the sub-band  do not have harmonic character.

 (1095)

where  and  denote the energy of high frequency bands and low frequency bands, respectively.  and  denote the numbers of correspond high frequency coefficients and correspond low frequency coefficients respectively.

(a3) When,  is determined as follows:

If the encoded bandwidth is SWB and the encoding mode is harmonic; Otherwise, .

(b) Based on the overall noise factor, the noise gain is calculated:

 (1096)

When the sub-band  is a bit allocation saturated sub-band or  is less than zero, the noise gain  is set to zero.

(c) In order to smooth the noise gain between the current frame and the previous frame, a sub-band boundary  is defined. If the encoded bandwidth is SWB, . Otherwise, . Here  denotes the last sub-band with encoded pulses in current frame,  denotes the last sub-band with encoded pulses in previous frame.

For the sub-band, if, then the noise gain  is smoothed as follows:

When  is less than:

 (1097)

When  is equal to:

 (1098)

where  denotes the average envelope of the previous frame,  denotes the noise gain of the previous frame.

5.3.4.1.4.1.5.2.3 Noise filling

First, if the encoded bandwidth is not SWB and, modify the decoded coefficients:

 (1099)

For sub-band, if the conditions: bit allocation is unsaturated, the encoded bandwidth is not SWB, , are all satisfied, the decoded coefficients is filled noise as follows:

Initialize a counter  to zero. Based on the smoothed noise gain, calculate a noise filling factor:

 (1100)

If the decoded coefficient is equal to zero,

 (1101)

where  is generated by random noise,  is increased by 1.

Otherwise, the noise filling of the decoded coefficients is described as follows:

 (1102)

Finally, update the memories as follows:

,, (1103)

5.3.4.1.4.1.5.3 Pitch filtering spectrum coding (PFSC)

5.3.4.1.4.1.5.3.1 PFSC overview

Pitch filtering spectrum coding is applied only for the SWB and FB signals.

To encode and decode high frequency MDCT coefficients, PFSC utilizes both decoded MDCT coefficients which is the output of “TCQ”, “De-norm and Fine gain SQ” blocks in Figure 66 and noise-filled low-frequency decoded MDCT coefficients. The band which is encoded by PFSC is determined through the bit-allocation process described in the previous subclauses. In the PFSC, the four highest bands are the high-frequency bands, and the other bands lower than the high-frequency bands are the low-frequency bands. For example, in Table 108, the low-frequency bands are =0 to 17, and the high-frequency bands are =18 to 21, for the 13.2 kbps.

For each band, the most similar match with the selected similarity criteria is searched from the TCQ-quantized and envelope normalized low-frequency content (i.e., the envelope normalized version of the decoded MDCT coefficients in the low-frequency bands). The most similar match is scaled with a scaling factor calculated using the quantized band energy to obtain the synthesized high frequency content.

5.3.4.1.4.1.5.3.2 Envelope normalization

In a manner similar to G.718 Annex B, quantized low-frequency content is normalized with its envelope. However, the TCQ-quantized low-frequency content (which is the decoded low-frequency content before the noise filling) is a sparse pulse sequence, and a normalization process is therefore used to flatten the low-frequency content.

The low-frequency content is normalized by dividing it by the maximum amplitude value in each sub-band. Here, the sub-band configuration is special and used only for this normalization. Each sub-band consists of 12 MDCT coefficients. By performing this process, each sub-band will have the same maximum amplitude value, and the low-frequency content can therefore be converted to an MDCT coefficient sequence whose spectral characteristic is flat and smoothed.

The envelope normalization (or smoothing) process is separately performed on the TCQ-quantized low-frequency content and the filled low-frequency noise content, respectively. And the amplitude of the normalized noise content is adaptively scaled according to the sparseness of the TCQ-quantized low-frequency content. The sparseness is calculated by dividing the number of non-zero spectrum in the TCQ-quantized low-frequency content by the bandwidth of the low-frequency content. A threshold is calculated using the sparseness and used for removing low amplitude TCQ-quantized low-frequency content and for scaling the maximum amplitude of the noise content. The normalized TCQ-quantized low-frequency content is further modified so that its non-zero content has larger amplitude than the maximum amplitude of the normalized noise content thus dynamic range of the normalized TCQ-quantized low-frequency content is modified for better matching with a targeted high-frequency spectrum. Finally, the scaled low-frequency noise content and the modified TCQ-quantized low-frequency content are added for generating envelope normalized spectrum (MDCT coefficients). If the generated spectrum becomes zero, such spectrum component is replaced with a randomly generated noise whose maximum amplitude is limited to the half of the maximum amplitude of the scaled noise component.

A block diagram of this envelope normalization is shown in Figure 98 of subclause 6.2.3.1.3.1.4.3.3. In Figure 98, all processing blocks except ‘Lag info. decoding’ are common to the both sides of encoder and decoder.

5.3.4.1.4.1.5.3.3 Band search

5.3.4.1.4.1.5.3.3.1 Selection of representative MDCT coefficients

Similarly to G.718 Annex B, a band search approach is used. The last four bands (i.e. *b*=18 to 21 in 13.2 kbps and *b*=20 to 23 in 16.4 kbps) are then subject to be encoded with PFSC. As shown in the table 108, the widths of the bands are 55, 68, 84, and 105 for 13.2 kbps, and 59, 74, 92, and 115 for 16.4 kbps.

To reduce computational load for calculating correlation, only limited number of input (target) MDCT coefficients are selected as representative MDCT coefficients and used for correlation calculation. The selection of the MDCT coefficients is performed by amplitude threshold process, i.e. an MDCT coefficient is selected if its absolute value is greater than a threshold. The threshold is determined using the average and standard deviation of the absolute values of the MDCT coefficients in a subjected high-frequency band.

 (1104)

Here is the initial threshold for the *i*-th high-frequency band,  is the average of the absolute values of the MDCT coefficients in the *i*-th high-frequency band,  is the standard deviation of the absolute values of the MDCT coefficients in the *i*-th high-frequency band, and  is a factor for controlling the selected number of the MDCT coefficients. is chosen so that a calculated threshold becomes higher than a threshold which is expected to be appropriate for selecting the limited number of the MDCT coefficients.

If the number of selected MDCT coefficients is less than a pre-determined number, the threshold is updated by the following equation, and an additional selection process is performed.

 (1105)

Here,  is the weakest attenuation factor and is the strongest attenuation factor, and .  is the pre-determined number of the MDCT coefficients to be selected in the end, and  is remaining number of MDCT coefficients to be selected.  is the updated threshold. By using this equation, the threshold is calculated according to the number of non-selected MDCT coefficients, i.e. the larger the number of non-selected MDCT coefficients is, the lower the threshold become. The above equation is equivalent to the following equation.  is the number of already selected MDCT coefficients.

 (1106)

This threshold update is performed twice using a different set of  and  unless the number of selected MDCT coefficients does not reach the pre-determined number.

The selected MDCT coefficients (target MDCT coefficients for the band search) are stored in a memory as their frequency positions and used for the band search process.

5.3.4.1.4.1.5.3.3.2 Matching process

Once the representative MDCT coefficients are selected from the j-th band input MDCT coefficients , a matching process is performed by calculating the correlation between the representative MDCT coefficients and normalized low-frequency MDCT coefficients derived from the envelope normalized MDCT coefficients calculated in subclause 5.3.4.1.4.1.5.3.2. Since the correlation is calculated only using the selected MDCT coefficients, required computational complexity can be saved.

The task of the matching process is to find *k'* which maximizes *S*(*k'*).

 (1107)

where , and  denote the following.

 : correlation between representative MDCT coefficients and normalized low-frequency MDCT coefficients for the *k*’-th lag candidate,

 : energy of normalized low-frequency MDCT coefficients for the *k’*-th lag candidate  
 : number of lag candidates for the *j*-th band.

 and  are calculated by the following equations.

(1108)

(1109)

, ,  and  denote the followings.

: selected number of representative MDCT coefficients in the *j*-th band,  
: frequency position of the *k*-th representative MDCT coefficient in the *j*-th band,  
: the *k*’-th lag candidate for the *j*-th band,  
: starting frequency position of normalized low-frequency MDCT coefficients for the *j*-th band.

The lag candidates are defined as the frequency positions of non-zero normalized low-frequency spectrum. Therefore  means the *k*’-th non-zero normalized low-frequency MDCT coefficients frequency position in the *j*-th band search range. The *j*-th sub-band search range is started at, which is defined as offsets from the zero frequency point of the low-frequency spectrum. = {0, 0, 64, 64}

By using this lag candidate representation, even when the bit budget for the lag information is small, actual lag search range can be wide, and it enables to guarantee to generate candidate spectra which have at least one non-zero spectrum.

Figure 76 shows a conceptual block diagram of the matching process. For the lag search, only the TCQ quantized component is used as the low-frequency MDCT coefficients while both of the TCQ-quantized and noise-filled low-frequency MDCT coefficients are used for generating high-frequency spectrum and calculating the scaling factors.

The best *k'*, which maximizes *S*(*k'*), is packed into the bit stream as an encoded parameter of the best lag candidate.

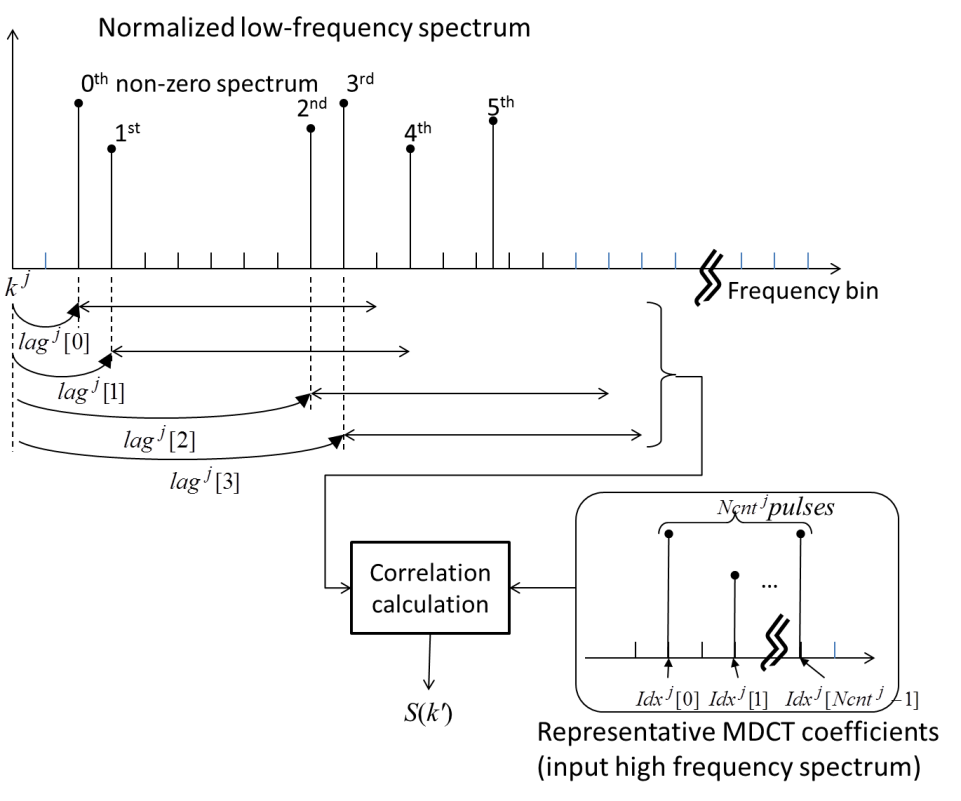


Figure 76: Conceptual block diagram of the matching process

5.3.4.1.4.1.5.3.3.3 Scaling and noise smoothing

Once the best match has been found through the matching process, scaling factors are calculated for the searched bands using the quantized band energies. Each scaling factor is calculated as the square root of the quotient of each quantized band energy divided by its corresponding band enegy of a generated high-frequency spectrum. The band energy of the generated high-frequency spectrum is equal to  for the selected  and is calculated according to equation (1109). The calculated scaling factors are attenuated by the scaling factor of 0.9.

Inter-frame smoothing process is applied on the generated high-frequency noise components. The generated MDCT coefficients whose amplitudes are below a threshold calculated using the sparseness of the TCQ-quantized low-frequency components are targeted for the smoothing process. When the energy of the targeted components in the previous frame is sufficiently less than current band energy, relatively strong smoothing is applied, while relatively weak smoothing is applied in other cases (i.e. current band energy is not sufficiently larger than the noise energy in the previous frame). The smoothing is performed on the energy of the targeted (noise) components.Then the smoothened energy of the noise components is divided by the energy of the noise components for calculating the final scaling factor in the smoothing process. The final scaling factor is applied to the noise components to obtain smoothened MDCT coefficients for the noise components.

Local decoding at the encoder side is necessary for switching to non-MDCT-based modes. For the local decoding, the noise-filling process is performed between the “De-norm. & Fine gain SQ” and the “PFSC” blocks in Figure 66.

5.3.4.1.4.2 Transient mode

5.3.4.1.4.2.1 Energy envelope coding

The energy envelope coding is performed as described in subclause 5.3.4.1.3.

5.3.4.1.4.2.2 Bit allocation

The bit allocation to bands based on quantized band energies is performed as described in subclause 5.3.4.1.4.1.4.4. However, for transient frames the Limited band mode described in subclause 5.3.4.1.4.1.4.4.2 is not used. For SWB, no bits are allocated at the 6th and 7th bands.

5.3.4.1.4.2.3 Fine structure encoding

5.3.4.1.4.2.3.1 Trellis Coding Quantization (TCQ)

The spectral coefficient quantization is done as is described in subclause 5.3.4.1.4.1.5.1.

5.3.4.1.4.2.4 Noise-filling for 0-bit assigned bands

The noise filling is done as is described in subclause 5.3.4.1.4.1.5.2.

5.3.4.1.4.3 Harmonic Mode

5.3.4.1.4.3.1 Energy envelope coding

Overall framework of the Harmonic mode is the same as the Normal mode as shown in Figure 66, but the PFSC block is called as PFSC-based gap filling in the Harmonic mode.

The energy envelope coding is performed as described in subclause 5.3.4.1.3.

5.3.4.1.4.3.2 Bit allocation

5.3.4.1.4.3.2.1 Allocating bits for fine gain adjustment

Bit-allocation process is performed in the following manner when the signal is classified as harmonic. Firstly, two bits are reserved for transmitting the noise factor information (cf. Sec. 5.3.4.1.4.3.3.3.6) followed by four bits are allocated for performing gap filling using PFSC based approach. Then some bits are reserved for applying fine gain quantization to the band energies that are larger than the others.

The bit budget used for the spectrum quantization is obtained using

 (1110)

where *ebits* is the bits consumed for quantizing the band energies which is obtained from subclause 5.3.4.1.3,  is the available bit budget for spectrum quantization. is the number of bits reserved for fine gain quantization and described in table 119. For SWB and FB at the bit rates 13.2 and 16.4 kbps, .

5.3.4.1.4.3.2.2 Adaptive bit allocation to bands

Based on the available bit budget, bits are allocated to the bands using the adaptive bit allocation. The adaptive bit-allocation scheme uses the quantized band energiesto allocate the available bits in a frame among the bands. In this scheme, bits are allocated by adaptively grouping the band energies into a number of groups; each group is allocated variable number of bits based on characteristics between the groups followed by bits allocation to bands within groups based on the available bit budget to the groups. Figure 77 illustrates the overview of the adaptive bit allocation. The detailed procedure is described in the following subclause.

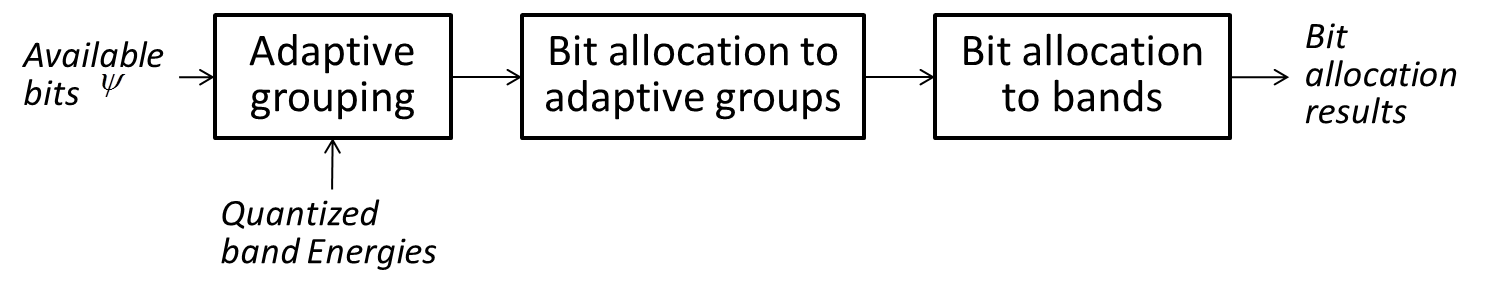


Figure 77: Overview of the Adaptive bit allocation scheme

5.3.4.1.4.3.2.2.1 Bits allocation to groups.

The adaptive bit-allocation scheme uses the quantized band energies,  to allocate the available bits  in a frame among the bands.

First, the bit-allocation vector entries, i.e., bit allocation of each bands in bits per sample, are set to one, and followed by temporary storage of band energies in a buffer. This is done according to:

 (1111)

In order to allocate bits to bands using band energies, firstly bands are adaptively grouped and each group contains variable number of bands, the grouping is to separate the dominant frequency band from non-dominant frequency band. In total, four groups are formed, while the widths of the first three groups are adaptively varied and the last group has a fixed width which has last four bands. The maximum variable width  of each adaptive group is thresholded and it is given in table 120

Table 120: Maximum widths of the groups

|  |  |  |
| --- | --- | --- |
|  | 13.2kbps | 16.4kbps |
|  | 7 | 7 |
|  | 9 | 10 |
|  | 2 | 3 |

The grouping of bands is done by identifying the dominant frequency vectors (the frequency bands with large and local maximum energy factor values in the spectrum). Using the dominant frequency band as center, include the descending slope at both sides to one group; the group width is adaptively determined corresponding to the input signal characteristic. If the dominant frequency band is at the edge, only one side of the descending slope would be included in the group. In order to reduce the complexity sensitive ness of human ear is considered when searching

Once width of each group  is decided, bits are allocated to each group and it is done in the following steps.

1. Calculate the energy of each group according to

 (1112)

where  is the start and end of each group and width of each group is defined according to

. (1113)

1. Allocate bits to the last group which has a fixed width based on the following steps.
   1. Calculate average number of bits that can be allocated for a group

 (1114)

* 1. Calculate mean for the group energies according to

 (1115)

* 1. Calculate energy difference for last group according to

 (1116)

* 1. Calculate group energy ratio between the last group and average group energy of the remaining groups according to

 (1117)

* 1. Calculate the energy difference between maximum energy of a subvector in the last group and with average sub vector energy of the rest of the groups.

 (1118)

where  is the maximum energy in the last group according to equation (1119) and  is the average energy of the rest of the groups band and it is calculated according to .

 (1119)

 (1120)

* 1. Allocate bits to the last group according to

 (1121)

where *bw1* is the bit allocation weight, which takes value based on



1. Once bits are allocated to last group, remaining bits are calculated according to

 (1122)

1. Based on the remaining bits obtained from c), bits are allocated to the rest of the groups using following steps.
   1. Calculate the band energy difference between bands.

 (1123)

* 1. Identify the maximum band energy difference according to

 (1124)

* 1. Identify the group which has maximum band energy difference  and allocate more bits to the group .

Ifbelongs to group *g*=1, bits allocation to the groups will be according to

 (1125)

If belongs to group *g*=0 or group *g*=2, bits allocation to the groups will be according to

 (1126)

where *Scale1* and *Scale2* values are based on table 121

Table 121: Scale factors for groups

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | 13.2 kbps | 16.4 kbps |
|  | *Scale1* | 1.05 | 1.1 |
| *Scale2* | 0.97 | 0.92 |
|  | *Scale1* | 0.92 | 0.97 |
| *Scale2* | 1.0 | 1 |

Bits are allocated to individual bands in a group based on the available bit budget for the groups and it is done according to subclause 5.3.4.1.4.3.2.2.2. If 0 bits are available to any of the groups the corresponding bands in the group are allocated 0 bits.

5.3.4.1.4.3.2.2.2 Bits allocation to bands in a group

The bits allocation to individual bands in a group is applied using the steps given below

1. Calculate sum of band energies in a group

 (1127)

1. Calculate Mean energy for the group

 (1128)

1. Set *scale3* value according to

 (1129)

where is a bitallocation weight constant which is used for controlling the amount of bits being allocated to the bands.

 is the number of bands whose difference between the Mean energy for the group and band energy of the sub band exceeds 5 dB.

1. For each iteration in a group,
   1. Calculate Mean energy for the group

 (1130)

* 1. Allocate bits to the individual subvectros in a group according to .

 (1131)

* 1. Once the bits are allocated to indvidual bands in a group *g*, identify the band which has least energy in the group and verify whether bits allocated to the identfied band has under allocation using a threshold, if the band has under allocation. update the sum of energies using



where *Gmin* is the index corresponding to minimum energy *Rmin* of a band in the group *g*

* 1. Repeat Step g, until there is no under allocation.

The process described in subclause 5.3.4.1.4.3.2.2.2, is used for allocating bits to individual bands for groups *g=0,1,2,3.*

5.3.4.1.4.3.3 Fine structure encoding

5.3.4.1.4.3.3.1 Trellis Coding Quantization (TCQ)

The spectral coefficient quantization is done as is described in subclause 5.3.4.1.4.1.5.1.

5.3.4.1.4.3.3.2 Noise-filling for 0-bit assigned bands:

This subclause gives a technical overview of the noise filling processing. Noise filling consists of two algorithms. The first algorithm described in this subclause 5.3.4.1.4.1.5.2 fills the gaps in the quantized spectrum where coefficients have been quantized to zero when the bit allocation clause allocates non zero bits to the bands and fills the un quantized bands up to the transition frequency where the transition frequencyis estimated based on the received bit allocation. The second algorithm is described as follows

5.3.4.1.4.3.3.3 PFSC-based gap filling

5.3.4.1.4.3.3.3.1 Overview

This subclause is only applied to SWB and FB input signals. The spectral coefficients which belong to bands which are assigned zero bits from the bit‑allocation clause are not quantized. This means that not all transform coefficients are transmitted to the decoder. From the noise filled quantized spectrum, the gaps in the high frequency region  which has zero bit allocation are identified and a predicted spectrum information is generated and the missing frequency bands (gaps) are filled with the predicted spectrum information.

In order to perform gap filling, the most similar match with the selected similarity criteria is searched from the coded and envelope normalized content and encodes the lag index parameter followed by encoding of the noise factor. The lag index and noise factor parameters are used in the decoder for generating the predicted spectrum, which will be used for filling the gaps. Figure 78 illustrates the overview of PFSC based gap filling for Harmonic mode.



Figure 78: Overview of the Harmonic mode PFSC based gap filling

5.3.4.1.4.3.3.3.2 Envelope Normalization

Low frequency content is extracted from the quantized noise filled spectrum, which is stored temporarily in a buffer and supplies the buffer information to noise separator, which separates noise and quantized content. The envelope of the quantized and noise contents is normalized before further processing. The normalization is performed in logarithmic domain and it is according to

Peak to average information is calculated in log domain for the quantized content.

 (1132)

where  is the maximum absolute amplitude of the quantized spectrum and  is the mean of the absolute amplitude of the quantized spectrum.

The quantized content  is divided into  sub-vectors and the total number of the sub-vectors is based on the table 122. Each sub-vector  consists of eight spectral coefficients: This sub-vector division is only available for envelope normalization.

 (1133)

The arithmetic means of the 1st and 2nd half of each sub-band  and  are calculated as follows:

 (1134)

 (1135)

Next, the geometric mean of  and  in logarithmic domain of each sub-band,  is obtained as:

 (1136)

Here, the square root calculation in linear domain is performed by multiplying 0.5.

The , *l* = 0, …, *Nsv*, is smoothed using moving average method with a span of seven samples. The envelope smoothed values  are obtained as:

 (1137)

The smoothed envelope  is transformed from logarithmic domain to linear domain as follows:

 (1138)

The quantized contents are normalized by multiplying the smoothed envelope  in order to obtain the normalized contents:

 (1139)

Table 122: Number of sub-vectors for energy normalization

|  |  |
| --- | --- |
| Bitrate in kbps | *Nsv* |
| 13.2 | 32 |
| 16.4 | 38 |

From the normalized content, thresholds are calculated according to

 (1140)

where  value is according to.

 (1141)

is calculated according to

 (1142)

Absolute values of the normalized coefficients are compared to the thresholdsand , which are adaptively calculated by equation (1140). The normalized coefficients with absolute amplitudes above  are clipped to to avoid excessive spectral peaks, and the normalized coefficients with absolute amplitudes below  are suppressed to 0 to enhance the harmonic structure, according to



Above process i.e., normalization and clipping of the normalized contents is also applied for normalizing the low frequency noise contents and the obtained normalized noise contents is represented as  the normalized noise contents are further adjusted according to

 (1143)

where  is the sparseness ratio and it is calculated according to

 (1144)

is the number of non-zero spectrum in the TCQ-quantized low-frequency content.

The normalized low frequency quantized noise filled spectrum is obtained according to

 (1145)

5.3.4.1.4.3.3.3.3 Band Search

The high-frequency region of with width *hfw* is divided into four bands as follows

 (1146)

where  is the input MDCT spectrum,  is the starting position of high frequency, high frequency width is represented as  and its corresponding values are according to table 123. The values of band widths  corresponding to index *i* and offsets  are according to tables 124 and 125.

Table 123: High frequency width and starting position

|  |  |  |
| --- | --- | --- |
| Bitrate in kbps |  |  |
| 13.2 | 256 | 312 |
| 16.4 | 300 | 340 |

Table 124: High frequency band width and offsets for 13.2 kbps

|  |  |  |
| --- | --- | --- |
| Index *i* |  |  |
| 0 | 56 | 0 |
| 1 | 100 | 56 |
| 2 | 100 | 156 |
| 3 | 56 | 256 |

Table 125: High frequency band width and offsets for 16.4 kbps

|  |  |  |
| --- | --- | --- |
| Index *i* |  |  |
| 0 | 60 | 0 |
| 1 | 110 | 60 |
| 2 | 110 | 170 |
| 3 | 60 | 280 |

For sub-band , *i*=0,1there is a search band  which defines from where the best match is searched.

The start position  and the width  of each sub-band are set as follows:

 (1147)

 (1148)

where is the number of search positions and it is calculated according to

 (1149)

where the number of bits  is reserved for the band search for index *i.*

The best match is searched for every band *i* ,in order to reduce computational load for calculating correlation, only limited number of input spectral coefficients are selected and used for the calculation of correlation. The selection of the MDCT coefficients is described in subclause 5.3.4.1.4.1.5.3.3.1.

Once the input spectral coefficients are selected, searching process is performed by calculating the correlation between the input spectral coefficients and normalized low-frequency coefficients derived from the envelope normalized coefficients calculated in subclause 5.3.4.1.4.3.3.3.2. Since the correlation is calculated only using the selected coefficients, required computational complexity can be reduced.

The best match is searched for every band *i* as follows. The correlation for index *k’* is computed as

 (1150)

where ,,  and  denote the following

 is the correlation between input spectral coefficients  and search band  for the *k’-th* lag candidate,

 selected number of representative spectral coefficients in the *i*-th band.: frequency position of the *k*-th representative spectral coefficient in the *i*-th sub-band,  
: is the start and end position of search lag and its values are calculated according to

 (1151)

where is the previous frame best match position for band *i*  
similarly, the energy for index *k’* is obtained as

 (1152)

where  is the energy of the search band for the k’-th lag candidate

The actual similarity measure used is of the form

 (1153)

The task is to find *k’* which maximizes *S’*(*k’*). The complexity of the implementation can be reduced by squaring *S’*(*k*’), which removes the absolute value signs as well as square root from the denominator as equation (1107). The best match is now searched efficiently for band *i* using



where  is the best match position for band *i* and if there is no spectral coefficients in the search band then  value takes according to

 (1154)

The best match index  is packed into the bit stream as the parameter. Once the best match index is obtained, past frame best match index  is updated with current frame best match index according to

 (1155)

5.3.4.1.4.3.3.3.4 Structure analysis for Harmonics

This subclause gives a technical overview of the structure analysis for harmonics which is applied for SWB and FB Harmonic signals. The purpose of this subclause is to estimate the harmonics from the quantized spectrum and the estimated harmonic is used for generating the new tonal spectrum for the HF region at the decoder, where the HF region is calculated using the frequency transition which is estimated based on the received bit allocation.

The first step of the structure analysis is to extract a portion from the quantized spectrum. A portion from 2 kHz to 6.4 kHz for 13.2 kbps and from 2- 7.5 kHz for 16.4 kbps is used for structure analysis.

From the extracted portion, structure for harmonics is analysed. The detail procedure of this subclause comprises the following steps:

1. Grouping of spectral coefficients into a number of blocks of equal length; in total 16 and 19 blocks (*Nblock*) are formed for 13.2 and 16.4. Kbps with a length of 16 coefficients per block.
2. Extract the spectral peak magnitude  and their corresponding positions  within each block; as shown

 (1156)

1. From the extracted spectral peaks identify the spectral peaks whose spacing lie closely and discard the spectral peak from analysis which is not perceptually important.
2. Split the extracted portion into regions with cut-off at 140, 200, 16*Nblock*-1 spectral coefficients and count the peaks in each region according to

 (1157)

Before the process counteris set to zero.

1. Calculate the spacing between the identified peak positions and identify the minimum and maximum spacing between spectral peaks.

 (1158)

6) Calculate sum of spacing according to

where are the sum of spacing and the counter increments when the respective conditions are satisfied.

7) Estimate the harmonic frequency based on the spacing between the identified peak positions according to the following.



where are the estimated harmonic frequencies, which can be used for generating the missing bands.

In order to improve the stability of present frame, previous frame estimated frequency  information is used.

if  and the calculated estimated frequencies  has a zero value it indicates that there is no harmonics extracted from *0* to *Nblock -1.*usually this happens when decoded spectrum is sparse. For this case, is set to previous frame estimated frequencies if it not equal to zero or it is set to a default value of 80. In the encoder, estimated harmonic  is used for generating the noise spectrum for the HF region. As the estimated harmonic  is determined using the quantized spectrum, there is no need to transmit the  parameter to the decoder. Under rate switching conditions this structure analysis for harmonics is also used for bitrates above 16.4 kbps for extracting the harmonics.

5.3.4.1.4.3.3.3.5 Noise filling for the predicted spectrum

First high-frequency region  with width *hfw* is divided into four bands with same band configuration described in subclause 5.3.4.1.4.3.3.3.3

From the envelope normalized noise spectrum, a desired portion of noise spectrum is extracted. The start position  and the end  of each desired portion in the normalized noise spectrum  are set as follows:

 (1159)

 (1160)

here *lagsi* is the number of search positions and it is according to equation (1149), lag index value is obtained from subclause 5.3.4.1.4.3.3.3.3.

Estimate the position of harmonics for band *i* =0, 1 in the predicted spectrum, according to

 (1161)

where  represents the first tonal position of the HF spectrum estimated based on the end tonal frequency position in the low frequency spectrum. represents pulse resolution for predicted spectrum and  is obtained from subclause 5.3.4.1.4.3.3.3.4.

Fill the bands,*i*=0,1 using extracted noise from the normalized spectrum  except in the positions obtained in equation (1161) and fill the remaining bands,*i*=2,3 by copying the information obtained from the lower bands *i*=0,1 reversely.

5.3.4.1.4.3.3.3.6 Noise factor

On the decoder side a predicted spectrum  is generated for the high frequency region by using the envelope normalized noise-filled quantized spectrum, which is obtained from subclause 5.3.4.1.4.3.3.3.5. The predicted spectrum is generated for the high frequency region, first by extracting the desired noise components from the described in subclause 5.3.4.1.4.3.3.3.5 followed by tonal generation using the envelope normalized quantized spectrum . To control the amount of noise inserted in the predicted spectrum at the decoder, a noise factor is computed on the encoder side and quantized using a 2-bit scalar quantizer. The procedure for noise factor calculation is described in this subclause.

From the input spectral coefficients , extract the high-frequency region  according to

(1162)

where the starting position of high frequency is represented as and high frequency width is represented as. The corresponding values of  is same as in table 123.

Tonal components in the  are selected using the values  obtained in subclause 5.3.4.1.4.3.3.3.3. Calculate energy of the selected coefficients according to

 (1162a)

Calculate noise factor using  and  according to

 (1163)

where  is the energies obtained using,  according to

 (1164)

The calculated noise factor *Nfac* is encoded using a two bit scalar quantizer.

#### 5.3.4.2 High-rate HQ coder

Table 126: High rate HQ supported modes

|  |  |  |
| --- | --- | --- |
| Bitrate [kbps] | Bandwidth | Supported modes |
| 32 | WB | Normal, Transient |
| 24.4, 32 | SWB | Transient, Harmonic, HVQ, Generic |
| 24.4, 32 | FB | Transient, Generic |
| 64 | SWB, FB | Normal, Transient |

The high level structure of the high-rate HQ coder is in figure 79. The different operating modes of the high-rate HQ coder are outlined in table 126, the high-rate HQ is used at WB, SWB, and FB, at bit-rates 24.4, 32, and 64 kb/s. The coding is done in the MDCT domain. There are 5 different modes: Transient mode handles transient signals by using shorter transforms, Harmonic mode handles harmonic signals and HVQ takes care of strongly harmonic signals. Normal and Generic mode are used for all other signals.



Figure 79: High level structure of the high-rate HQ encoder

The whole frequency spectrum is divided into bands, there are 3 different band structures used: Default, harmonic, and wideband, see tables 127, 128, and 129. Harmonic band structure is used for HVQ and Harmonic mode, wideband band structure is used for WB. Otherwise the default band structure is used.

Table 127: Normal band structure used in high rate HQ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|  | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 |
|  | 7 | 15 | 23 | 31 | 39 | 47 | 55 | 63 | 71 | 79 | 87 | 95 | 103 |
|  | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|  | 104 | 112 | 120 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 | 256 | 280 |
|  | 111 | 119 | 127 | 143 | 159 | 175 | 191 | 207 | 223 | 239 | 255 | 279 | 303 |
|  | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 |
|  | 304 | 328 | 352 | 376 | 400 | 424 | 448 | 472 | 496 | 520 | 554 | 576 | 608 |
|  | 327 | 351 | 375 | 399 | 423 | 447 | 471 | 495 | 519 | 553 | 575 | 607 | 639 |
|  | 39 | 40 | 41 | 42 | 43 |  |  |  |  |  |  |  |  |
|  | 640 | 672 | 704 | 736 | 768 |  |  |  |  |  |  |  |  |
|  | 671 | 703 | 735 | 767 | 799 |  |  |  |  |  |  |  |  |

Table 128: Wideband band structure used in high rate HQ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|  | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 |
|  | 7 | 15 | 23 | 31 | 39 | 47 | 55 | 63 | 71 | 79 | 87 | 95 | 103 |
|  | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|  | 104 | 112 | 120 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 240 | 256 | 288 |
|  | 111 | 119 | 127 | 143 | 159 | 175 | 191 | 207 | 223 | 239 | 255 | 287 | 319 |

Table 129: Harmonic band structure used in high rate HQ

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|  | 0 | 8 | 16 | 24 | 32 | 40 | 48 | 56 | 64 | 72 | 80 | 88 | 96 |
|  | 7 | 15 | 23 | 31 | 39 | 47 | 55 | 63 | 71 | 79 | 87 | 95 | 103 |
|  | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|  | 104 | 112 | 120 | 128 | 144 | 160 | 176 | 192 | 208 | 224 | 256 | 288 | 320 |
|  | 111 | 119 | 127 | 143 | 159 | 175 | 191 | 207 | 223 | 255 | 287 | 319 | 367 |
|  | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 |
|  | 368 | 416 | 464 | 512 | 576 |  |  |  |  |  |  |  |  |
|  | 415 | 463 | 511 | 575 | 639 |  |  |  |  |  |  |  |  |

The number of bands used in WB is 26, and for FB it is 44. For SWB 39 bands are used, except if Harmonic or HVQ is used. For Harmonic mode 31 bands are used. The HVQ will use bands 21-30 for 24.4 kb/s, or 24-30 for 32 kb/s.

Based on the different modes, the low frequency band signal is encoded with the different bandwidths. In addition, the envelopes of the higher frequency band are encoded differently according to the different modes.

Then, the mode information, the indices of the low frequency band signal and the envelopes of the higher frequency band signal are written to the bitstream.

##### 5.3.4.2.1 Normal Mode

5.3.4.2.1.1 Envelope calculation and quantization

The normor spectrum energy **of a band is defined as the root-mean-square (*rms*) value of the band and computed as follows:

 (1165)

where is the length of the band .

Table 130: Envelope quantization table

| Index | Code | Index | Code | Index | Code | Index | Code |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 |  | 10 |  | 20 |  | 30 |  |
| 1 |  | 11 |  | 21 |  | 31 |  |
| 2 |  | 12 |  | 22 |  | 32 |  |
| 3 |  | 13 |  | 23 |  | 33 |  |
| 4 |  | 14 |  | 24 |  | 34 |  |
| 5 |  | 15 |  | 25 |  | 35 |  |
| 6 |  | 16 |  | 26 |  | 36 |  |
| 7 |  | 17 |  | 27 |  | 37 |  |
| 8 |  | 18 |  | 28 |  | 38 |  |
| 9 |  | 19 |  | 29 |  | 39 |  |

In each frame, the norms are scalar quantized with a uniform logarithmic scalar quantizer with 40 steps of 3 dB.

The index of the quantized norm,, can easily be obtained as:

 (1166)

and is saturated such that it is limited to the range of [0,31] for 0 and [0, 39] for the others.

The quantization index of the lowest-frequency band, i.e., , is directly transmitted to the decoder. The quantization indices of the norms in the remaining bands are differentially coded by computing

 (1167)

The differential indices  are constrained into the range of [–15, 16]. This is performed by first adjusting negative differential indices and then adjusting positive differential indices as follows:

1) Compute the differential indices defined in equation (1167) in order from the highest-frequency band to the lowest-frequency.

2) 

3) Recompute the differential indices in order from the lowest-frequency band to the highest-frequency.

4) 

5) The adjusted differential indices in the range [0, 31] are obtained by adding an offset of 15 to.

5.3.4.2.1.2 Envelope coding

The first index of the quantized norm  is transmitted by packing the bits directly using 5 bits. The adjusted quantization differential indices are coded by one of four high-rate HQ norm coding modes shown in table 131. The mode requiring the least bits is selected and used for high-rate HQ norm coding. This HQ norm coding mode information is signalled with 2 bits. The final resulting reconstructed quantized norms are denoted by, .

Table 131: High-rate HQ norm coding modes

| Mode index | Description |
| --- | --- |
| 0 | Context based Huffman coding |
| 1 | Resized Huffman coding |
| 2 | Normal Huffman coding |
| 3 | Bit-packing |

5.3.4.2.1.2.1 Context based Huffman coding mode

If this coding mode is selected for the current frame, the context based Huffman coding is applied to the quantization differential indices described in subclause 5.3.4.1.3.3.1. In this case equation (1038) shall be replaced with

 (1168)

When the bit-packing mode is selected the adjusted differential rates are packed directly with 5 bits.

5.3.4.2.1.2.2 Re-sized Huffman coding mode

If this coding mode is selected for the current frame, the Re-sized Huffman coding is applied to the adjusted differential indices as same as the low-rate HQ coder. Details are described in subclause 5.3.4.1.3.3.2. The Huffman codes for the differential indices are given in table 115. When the bit-packing mode is selected the adjusted differential rates are packed directly with 5 bits.

5.3.4.2.1.2.3 Normal Huffman coding and bit-packing mode

If this coding mode is selected for the current frame, the Normal Huffman coding is then applied to the adjusted differential indices. The Huffman codes for the differential indices are given in table 132.

Table 132: Huffman coefficient table

| Index | Code | Index | Code | Index | Code | Index | Code |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0011010 | 8 | 001100 | 16 | 000 | 24 | 0011110 |
| 1 | 0111010 | 9 | 011100 | 17 | 010 | 25 | 0111110 |
| 2 | 1011010 | 10 | 101100 | 18 | 1010 | 26 | 1011110 |
| 3 | 1111010 | 11 | 111100 | 19 | 1110 | 27 | 1111110 |
| 4 | 0011011 | 12 | 0010 | 20 | 001110 | 28 | 0011111 |
| 5 | 0111011 | 13 | 0110 | 21 | 011110 | 29 | 0111111 |
| 6 | 1011011 | 14 | 100 | 22 | 101110 | 30 | 1011111 |
| 7 | 1111011 | 15 | 110 | 23 | 111110 | 31 | 1111111 |

When the bit-packing mode is selected the adjusted differential rates are packed directly with 5 bits.

5.3.4.2.1.3 Bit allocation

5.3.4.2.1.3.1 Envelope adjustment before bit allocation

In order to account for psycho-acoustical weighting and masking effects, the quantized norms are adjusted prior to bit allocation. The algorithm consists of mapping the quantized norms by using spectral weighting functions. This algorithm is only used at FB.

First, the quantized norms are mapped to the spectral domain. This is equivalent to copying the quantized norms in case of stationary signals and averaging the time-dependent quantized norms of the four spectra in case of transients. This is performed according to:

 (1169)

The obtained spectrum is afterwards mapped to a function which is similar to the ear's output of auditory filters; this gives a representation of the psycho-acoustical importance of the input signal. This operation is performed according to the following linear operation:

 (1170)

where the constants, and the summation interval are given in table 133. when the bit-rate is 24.4 kb/s,  at 32 kb/s, and at 64 kb/s.

Table 133: Spectrum mapping variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 3 | 8 |
| 1 | 1 | 1 | 3 | 6 |
| 2 | 2 | 1 | 3 | 3 |
| 3 | 3 | 1 | 3 | 3 |
| 4 | 4 | 1 | 3 | 3 |
| 5 | 5 | 1 | 3 | 3 |
| 6 | 6 | 1 | 3 | 3 |
| 7 | 7 | 1 | 3 | 3 |
| 8 | 8 | 1 | 3 | 3 |
| 9 | 9 | 1 | 3 | 3 |
| 10 | 10, 11 | 2 | 4 | 3 |
| 11 | 12, 13 | 2 | 4 | 3 |
| 12 | 14, 15 | 2 | 4 | 3 |
| 13 | 16, 17 | 2 | 5 | 3 |
| 14 | 18, 19 | 2 | 5 | 3 |
| 15 | 20, 21, 22, 23 | 4 | 6 | 3 |
| 16 | 24, 25, 26 | 3 | 6 | 4 |
| 17 | 27, 28, 29 | 3 | 6 | 5 |
| 18 | 30, 31, 32, 33, 34 | 5 | 7 | 7 |
| 19 | 35, 36, 37, 38, 39, 40, 41, 42, 43 | 9 | 8 | 11 |

The mapped spectrum is forward-smoothed according to the following:

 (1171)

and the resulting in-place function is backward-smoothed according to:

 (1172)

After the smoothing operation, the resulting function is thresholded in order to take into account an average level of absolute threshold of hearing. Thresholding and renormalization are performed according to:

 (1173)

where is given by table 133 and represents a pseudo-threshold of hearing.

The resulting function is further adaptively mapped, companded or expanded depending on the dynamic range of the spectrum, to the range of [–0,…,3] according to the following linear mapping:

 (1174)

The resulting spectrum is mapped back to bands according to:

 (1175)

If the variable *IsTransient* is set to *TRUE*, i.e., transient mode, the resulting spectrum is further smoothed according to:

 (1176)

Finally, the norms used for bit allocation are computed as:

 (1177)

5.3.4.2.1.3.2 Envelope based bit allocation

5.3.4.2.1.3.2.1 Wideband bit allocation of non-transient mode at 24.4/32kbps

A group based bit allocation scheme has been introduced for wideband signal coding at 24.4kbps and 32kbps to avoid zero bit allocations to some of the sub-bands when they may be perceptually important.

5.3.4.2.1.3.2.1.1 Group based bit allocation

The sub-bands are divided into three groups. Firstly, the initial number of allocated bits to each group is determined according to the sum or the average of the norms in each group. Based on the initial number of allocated bits to each group, the second stage group based bit allocation is performed according to the characteristics and the energy information of the signal.

To achieve the group based bit allocation, firstly the average of the quantized norms in the index range is calculated,

 (1178)

For the first 4 sub-bands, if the norm for bit allocation is less than, then the norm is set to .

As a first step in the bit allocation, 1 bit is allocated to code each sub-band, and then the sub-bands are divided into 3 groups, i.e. [0,…, 15], [16,…, 23], [24,…, 25].

Then calculate the following parameters which indicate the characteristics and the energy information of the signal for group based bit allocation.

The factors are initialized. These factors are then employed to adjust the norms, and then the average of the norms in each group is calculated along with the sum of the averages.

 (1179)

 (1180)

Table 134: Spectrum mapping group structure

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
| 0 | 0 | 15 | 16 |
| 1 | 16 | 23 | 8 |
| 2 | 24 | 25 | 2 |

The differences of the consecutive averages and calculated:

 (1181)

If  is greater than or equal to 6, or  is greater than or equal to 3.75, then the number of bits allocated to each group is described as follows,

 (1182)

where is the total available bit budget. Otherwise, the bit allocation for each group is detailed as follows:

Initialize the number of bits allocated to each group using the following equation,

 (1183)

If the average envelope of the third groupis more than 12, then adjust the number of bits of each group further:

 (1184)

where is the number of the saved bits and initialized to zero. If, the saved bits are re-allocated to each group according to:

 (1185)

and reset the factors.

If the number of bits allocated to the third group is less than 3, then the number of allocated bits in the third group are moved to the second group , and the number of bits allocated to the third group is set to zero, .

5.3.4.2.1.3.2.1.2 Bit allocation in each group

For the 3 sub-band groups, the smallest thresholds for the number of allocated bits are 5, 6 and 7, respectively, and the largest number of the sub-bandswhich are bit-allocated is calculated according to:

 (1186)

The norms in each group are re-ordered and the re-ordered norms are , where  is the number of the sub-bands in each group.

The step length  is initialized, and the re-ordered norms are adjusted according to the step length,  and :

 (1187)

The bits allocated to each sub-band are set according to the adjusted norms in each group as follows:

If, initialize the allocated bit of each sub-band to 1. If the adjusted norm is less than 0, reset it to 0. Then the following 4 steps are performed :

1. Initialize the counter.
2. Calculate the sum of the first norms in the group, , and allocate the bits of each sub-band in the index range :

 (1188)

1. If the number of bits allocated to the last sub-band is fewer than thresholds, , then the number of allocated bits in the  sub-band is set to zero and the counter  is decremented by one. Processing now returns to step 2.
2. If the number of bits allocated to the  sub-band is not fewer than thresholds, then the bit-allocation of the sub-bands in each group has been completed.

Otherwise, if , initialize the number of allocated bits of the first  sub-bands to 1, and initialize the bit of the last  sub-bands to 0. If the adjusted norms of the first  sub-bands are less than 0, set them to 0. Then the following 4 steps are performed:

1. Initialize the counter.
2. Calculate the sum of the first norms in the group, , and allocate the bits of each sub-band in the index range :

 (1189)

1. If the number of the bits allocated to sub-band is fewer than thresholds, , then the number of allocated bits in the last  sub-bands is set to zero and the counter  is set to to . Processing now returns to step 2.
2. If the number of bits allocated to all  sub-bands is not fewer than thresholds, then the bit-allocation of the sub-bands in each group has been completed.

5.3.4.2.1.3.2.2 General envelope based bit allocation

The adaptive bit-allocation scheme uses the adjusted quantized norms , to allocate the available bits in a frame among the bands.

The maximum number of bits assigned to each normalized transform coefficient is by default set tobits/coefficient.

First, the bit-allocation vector entries, i.e., bit allocation of each band in bits per sample, are set to zero. This is done according to:

 for  (1190)

The number of remainder bits, denoted , is set to the total available bit budget. The latter is calculated after subtraction of the number of signalling bits, the bits used by the envelope coding and possibly the noise level bits (see subclause 5.3.4.2.1.4) from the total available bits for the frame.

The vector of bit-allocations,, remainder and fulfil:

 (1191)

At each iteration, the index of the band which has the largest norm among number of bands used is found:

 (1192)

For this band, the algorithm allocates 1 bit for each spectral coefficient, i.e.,is incremented by 1. The norm will, on the other hand, be decremented by 6 dB, i.e., the norm index is decreased by two. These two operations are performed according to:

 (1193)

The remainder is updated as well to take into account the updated bit-allocation vector:

 (1194)

Whenever the bit-allocationreaches the maximum allowable bit-rate*,* its norm is set to minus infinity (in reality, to ) such that this vector is not taken into account in the next iterations. This procedure is repeated until is less than.

When the iterative procedure stops and, depending on the value of the remainder bits, the remaining bits are allocated to bands of a lower dimension than which caused the stop of the bit‑allocation loop.

The last step is to convert the bit allocation from bits per coefficients to total bits per band:

 (1195)

5.3.4.2.1.3a Fine structure quantization

The normalized MDCT spectrum is encoded in bands using the bit allocation .

5.3.4.2.1.3.a1 Fine gain bit allocation

The spectral envelope coefficients have been quantized in 3 dB steps. For higher rates, this resolution becomes too coarse and additional fine gain adjustments are made. The bits assigned for the band is split into a number of bits for the PVQ shape quantizer and bits for the fine gain quantizer . The assignment is done using a pre-trained look-up table based on the assigned band bitsand the bandwidth according t

 (1196)

where is a lookup-table for fine gain bits (see table 135) and  is the number of bits per sample rounded down. The bits for the PVQ shape quantization are obtained by subtracting the fine gain bits, .

Table 135: Fine gain bits table

|  |  |
| --- | --- |
| 0 | 0 |
| 1 | 0 |
| 2 | 0 |
| 3 | 1 |
| 4 | 2 |
| 5 | 2 |
| 6 | 4 |
| 7 | 5 |

5.3.4.2.1.3a.2 Fine structure quantization using PVQ

The quantization of the PVQ shape vector is performed as described in subclause 5.3.4.2.7. A decoded version of the PVQ shape vector is also obtained.

5.3.4.2.1.3a.3 Fine gain prediction, quantization and application

The fine gain adjustment is based on a predicted value, . It is formed using an accuracy measure  which combines the band bitrate , band size and the largest absolute integer pulse valueof the synthesized band.

 (1197)

where is the actual number of pulses used for encoding and depends on the bitrate and the PVQ encoding process.The accuracy measure  is then translated into a gain prediction  following this relation:

 (1198)

where approximates the optimal MMSE gain. In case there are bits allocated, further refinement of the fine gain may be done by encoding the gain prediction error , defined as

 (1199)

where the optimal MMSE gain is defined as

 (1200)

and the normalization factor to RMS=1.0 is

 (1201)

The gain prediction error is encoded with a non-uniform scalar quantizer in log domain using bits.

5.3.4.2.1.4 Noise level adjustment

The spectral coefficients which belong to bands which are assigned zero bits from the bit‑allocation procedure are not encoded. This means that not all transform coefficients are transmitted to the decoder.

The level of these non-coded spectral coefficients is estimated and quantized in the encoder. This is not done in WB at rates 24.4 kb/s and 32 kb/s.

The estimation of the non-quantized signal level is done directly in the normalized spectrum domain. Prior to estimating the noise level, a transition frequency between the noise fill region and high frequency noise fill region is estimated. This transition frequency is identically estimated in the encoder and decoder and marks the start of high frequency noise fill and the end of noise fill.

The transition frequency is estimated according to the criterion of the last quantized band. The general method consists in looping through all the bands, starting at down to 0. If there are no quantized coefficients in the current band, it will be flagged as filled by high frequency noise fill. If there are quantized coefficients in the band, the holes of this band as well as the following bands are filled using noise fill. Figure 80 illustrates such a procedure, where the transient frequency separates between noise-fill and high frequency noise-fill.

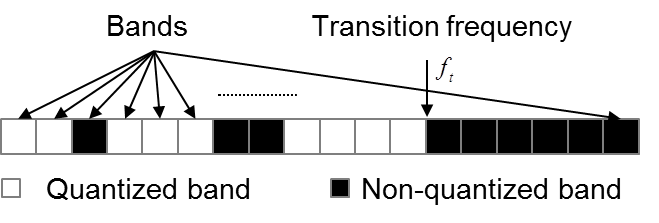


Figure 80: Estimation of transition frequency



Figure 81: Noise level estimation done below the transition frequency

The noise level is in turn estimated by measuring the average level of the normalized non-coded signal below the transition frequency, see figure 81. Formally this is obtained by:

 (1202)

where is the set of indices of coefficients allocated zero bits below the transition frequency *ft*.

The above equation always returns a value which is below zero because of the convexity of the logarithm. The noise level is quantized using a two-bit scalar quantizer according to table 136.

Table 136: Codebook entries for the uniform noise level quantizer

| Index | Output quantized NoiseLevel (dB) |
| --- | --- |
| 0 | 0 |
| 1 | –6 |
| 2 | –12 |
| 3 | –18 |

##### 5.3.4.2.2 Transient Mode

5.3.4.2.2.1 Envelope calculation and quantization

The bands are sorted according to equation (1203), where  is defined in tables 137, 138, and 139. Then the envelope calculation is done as in subclause 5.3.4.2.1.1 but using  instead of . After the quantization then the sorting is inverted by doing the inverse step of equation (1203).

 (1203)

Table 137: Envelope sorting table in transient mode for WB

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 0 | 1 | 8 | 9 | 16 | 20 | 24 | 21 | 17 | 11 | 10 |
|  | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|  | 3 | 2 | 4 | 5 | 12 | 13 | 18 | 22 | 25 | 23 | 19 |
|  | 22 | 23 | 24 | 25 |  |  |  |  |  |  |  |
|  | 15 | 14 | 7 | 6 |  |  |  |  |  |  |  |

Table 138: Envelope sorting table in transient mode for SWB

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 0 | 1 | 8 | 9 | 16 | 20 | 24 | 28 | 32 | 36 | 37 |
|  | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|  | 33 | 29 | 25 | 21 | 17 | 11 | 10 | 3 | 2 | 4 | 5 |
|  | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|  | 12 | 13 | 18 | 22 | 26 | 30 | 34 | 38 | 35 | 31 | 27 |
|  | 33 | 34 | 35 | 36 | 37 | 38 |  |  |  |  |  |
|  | 23 | 19 | 15 | 14 | 7 | 6 |  |  |  |  |  |

Table 139: Envelope sorting table in transient mode for FB

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|  | 0 | 1 | 8 | 9 | 16 | 20 | 24 | 28 | 32 | 36 | 40 |
|  | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 |
|  | 41 | 37 | 33 | 29 | 25 | 21 | 17 | 11 | 10 | 3 | 2 |
|  | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 |
|  | 4 | 5 | 12 | 13 | 18 | 22 | 26 | 30 | 34 | 38 | 42 |
|  | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 |
|  | 43 | 39 | 35 | 31 | 27 | 23 | 19 | 15 | 14 | 7 | 6 |

5.3.4.2.2.2 Bit allocation

For most bit rates bit allocation is done as is described in subclause 5.3.4.2.1.2.1 and 5.3.4.2.1.2.2. For SWB at 24.4 and 32 kbps the Generic mode bit allocation is used, as described in subclause 5.3.4.2.5.6.

5.3.4.2.2.3 Fine structure quantization using PVQ

The spectral coefficient quantization is done as is described in subclause 5.3.4.2.1.4.

##### 5.3.4.2.3 Generic, Harmonic and HVQ mode detector

Non-transient signals are further classified as Generic mode, Harmonic mode or HVQ at 24.4 kb/s and 32 kb/s for SWB inputs. The detailed classification is described in subclause 5.3.4.1.1, with slight differences in the final decision logic as follows.

At 24.4 kb/s the current frame mode is classified as harmonic if ,  and .

At 32 kb/s the current frame mode is classified as harmonic if ,  and ; or if .

5.3.4.2.3.1 HVQ classifier

Instantaneous noise-level and peak-level are estimated from the absolute values of transform coefficients, where at 24.4 kb/s and at 32 kb/s. The noise-level is calculated as:

 (1204)

where

 (1205)

The peak-level is calculated as

 (1206)

where

 (1207)

and both and are initialized to 800.

The per-band averages of noise-level  and peak-level  are calculated by averaging instantaneous level in a band (every 32 bins). The number of bands is at 24.4 kb/s and at 32 kb/s.

The average of the elements of the first half of gives the noise-floor gain, while the second half of coefficients forms. In a similar way the per-band peak-levels  produce two peak energy gains and.

The decision to switch to HVQ mode is based on the threshold in table 140, and three variables, , , and , defined below.

The threshold for selecting peak candidates is calculated as:

 (1208)

Absolute values of transform coefficients are compared to the threshold, and the ones with amplitude above it, form a vector of peak candidates. Elements from the peaks candidate vector are extracted in decreasing order, and when a peak is extracted the threshold over the neighboring peaks is adjusted with {0.7071068, 0.5000000, 0.2500000, 0.5000000, 0.7071068}. This procedure produces a set of spectral peaks, with number.

A measure of frequency sharpness per-band , similar to the one in 5.3.4.1.1, is defined as peak to noise-floor ratio in each band:

 (1209)

Variable is calculated as the number of bands for each 

Variable is calculated as

 (1210)

Table 140: Thresholds for HVQ mode decision

|  |  |  |  |
| --- | --- | --- | --- |
| rate |  |  |  |
| 24.4 kb/s | 4 | 20 | 22 |
| 32 kb/s | 7 | 23 | 22 |

The current frame is encoded in HVQ mode if:, , and 

##### 5.3.4.2.4 Harmonic Mode

The harmonic mode is used to code the harmonic-like signal. For harmonic signals, usually less noise filling is performed. It is also important to mitigate any discontinuity in the higher sub-band due to changes in core coding and for the most part it is preferable at 24.4 kb/s and 32kb/s, when there are insufficient bits to encode the full signal, to stop short of coding right up to the Nyquist frequency.

5.3.4.2.4.1 Envelope calculation and quantization

Envelope calculation and quantization is performed as described in subclause 5.3.4.2.1.1.

In order to reconstruct harmonic characteristics of the higher frequency band signal, the widths of the bands are larger than the ones for non-harmonic mode.

5.3.4.2.4.2 Bit allocation

The band-energy limitation factor  is introduced before bit allocation to mitigate discontinuous core coding in the higher sub-band.

Initialize the band-energy limitation factor

 (1211)

Calculate the energy of the first 10 sub-bands  and the energy of the first 28 sub-bands , and add the norms  to when . When ,  is the index of the highest encoded sub-band.

Reorder the quantized norms with the index range to obtain the reordered norms, and adjust the quantized norms as follows:

 (1212)

Then, the bit allocation is performed based on the adjusted norms to the sub-bands with the index range as described in subclause 5.3.4.2.1.2.2.

5.3.4.2.4.3 PVQ

The spectral coefficient quantization and coding is done according to the number of bits allocated to each sub-band as is described in subclause 5.3.4.2.1.3.

##### 5.3.4.2.5 HVQ

The HVQ mode is used only for SWB signals at 24.4 kb/s and 32 kb/s. At 24.4 kb/s it initially codes the first 224 MDCT coefficients (this corresponds to frequency range up to 5.6 kHz), while at 32 kb/s it initially codes the first 320 coefficients (this corresponds to frequency range up to 8 kHz). The major algorithmic steps at the encoder are: detect and code spectral peaks regions, code low-frequency spectral coefficients (the size of coded region depends on the remaining bits after peaks coding), code noise-floor gains for spectral coefficients outside the peaks regions, code high-frequency spectrum envelope to be used with the high-frequency noise-fill.

The input to the HVQ mode is the set of MDCT coefficients, the noise-floor gains, and the spectral peaks (both noise-floor gains and spectral peaks are calculated in the classification module, described in subclause 5.3.4.2.3.1). Each peak is normalized to unit energy and the surrounding 4 neighbours are normalized to with the peak gain. The peaks position, gain and sign are quantized. A VQ is applied to the four MDCT bins surrounding each peak. The coded number of peaks, peaks position, gain and sign, as well as the surrounding shape vectors are quantized and quantization indices transmitted to the decoder.

First sorted by energy peaks are arranged by position. Then peaks amplitudes  are differentially coded by 5 bits SQ on a log domain, and the indices Huffman coded to form a quantized peak gains. Prior to quantization the peak gains are multiplied by 0.25 and after the quantization multiplied by 4.

The spectral peak positions  are coded by choosing between two alternative lossless coding schemes, where the coding scheme that requires the least number of bits is selected, and explicitly indicated to the decoder.

The first lossless spectral peak position coding scheme is delta Huffman coding, suitable for periodic or semi-periodic spectral peak position distributions. The second lossless spectral peak position coding scheme is or-ing, suitable for sparse spectral peak position distributions.

The first scheme consists of the following steps: deltas (differences)  between consecutive elements (positions)  are created. Then these deltas are Huffman coded. Since the peaks are selected in a way that they cannot be positioned closer than 3 positions apart, is subtracted from the peak differences.

 (1213)

This eliminates the need of keeping codewords in the Huffman table, corresponding to unused deltas.

The second scheme consists of the following steps: the vector representing the spectral peak positions (absence of peak is indicated with 0) is divided into consecutive equal size (5 elements) bit groups, and the bits in each group are OR-ed forming a group bit vector (second layer), which is 5 time shorter. Each bit in this second layer indicates presence or absence or peak in the 5-dim group from layer below. In this way the only 5-dim groups from the first layer, which are not indicated as all-zero by the second layer have to be transmitted. The second (control) layer is always transmitted. The non-zero bit groups from the first layer also are mapped to exploit the constraints the fact that peaks cannot be closer than 3 positions (not all possible 5-dim vectors are allowed).

The selected coding scheme is explicitly signaled to the decoder with one bit: indicates sparse coding scheme, while indicates usage of delta coding scheme. Here is the largest distance between two consecutive peaks, which is compared to the largest difference possible to code with the pre-stored Huffman tables,  is the total number of bits consumed by the delta coding scheme, and  is the total number of bits consumed by the sparse coding scheme

 (1214)

Sign of the spectral peaks is coded separately with 1 bit per-peak, with 0 indicating negative sign and 1 indicating positive sign.

The peak regions to be coded are 5-dim MDCT vectors; a bin corresponding to the spectral peak and 2 MDCT bins on each side of the peak. The peak amplitude  of the central bin is used to normalize the entire peak region. In this way the central bin is scaled to unit energy, while surrounding 4 bins are normalized relative to the central one. The shape vector of the peak region, centered at bin  is defined as:

 (1215)

Each shape vector is quantized with 9 bits; 8 bits for the VQ index and 1 bit for classification. The numbers of peak regions vary over frames; this means different number bits will be required for coding the shape vectors, which will result in variable number bits used in the PVQ coding of low-frequency MDCT bands (except for the first low-frequency band, which has reserved bits).

Variations in the number of peaks per-frame results in different number of VQs, which leads to large variation in complexity. To keep the complexity nearly constant, while achieving low quantization error, the following approach is used: the search for each is performed in a structured CB, with dynamically selected offset and size of the search region. The starting point for the search is determined by initial classification of the input shape vector, while the length of the search region depends on the number of received shape vectors.

The codewords in the CB used for quantization of the shape vectors are order based on their distance to two pre-demined classes, with centroids and. The CB is structured in a way that the codewords closest to and most distanced to are in one side of the CB, while codewords closest to  and most distanced to  are clustered in the other side of the CB. The distance between the input vectors to each of the classes determines the starting point for the search. Since the codevectors in the codebook are sorted according to a distortion measure reflecting the distance between each codevectors and the centroids, the search procedure goes first over set of vectors that is likely to contain the best match.

The search space is dynamically adjusted to the number of input vectors. The maximum search space is used with 8 peaks or less at 24.4 kb/s, and 12 peaks or less at 32 kb/s. When larger numbers of peaks are to be quantized in the current frame, the search space is reduced to limit the peak complexity.

Table 141: Adaptive search space in HVQ at 24.4 kb/s

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |  | 1 |
|  | 128 | 136 | 145 | 155 | 167 | 181 | 197 | 217 | 241 | 256 |  | 256 |

Table 142: Adaptive search space in HVQ at 32 kb/s

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 |  | 1 |
|  | 128 | 134 | 141 | 149 | 158 | 168 | 179 | 192 | 206 | 224 | 244 | 256 |  | 256 |

The target shape vector exhibits certain symmetries (the MDCT coefficients on the both sides of the spectral peak have similar statistics) that can be used to optimize centroids for the class selection and the CB. A “flipped” version of the centroids used in the initial classification, “flipped” version of the CB (not pre-stored, but through modified search) can capture this symmetry in the shape vectors.

First the input shape vector is compared to 4 centroids (each centroid representing a respective class of codevectors in a codebook), which determines a starting point to the search. If or are selected, the same logic is used, but the search is performed in a “flipped” CB.

Table 143: Centroids for the class selection in HVQ search

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Class |  | | | |
|  | -0.2324457 | -0.4390556 | 0.0651793 | 0.2109977 |
|  | 0.1471332 | -0.1351437 | 0.4312476 | -0.1384814 |
|  | | | | |
|  | 0.2109977 | 0.0651793 | -0.4390556 | -0.2324457 |
|  | -0.1384814 | 0.4312476 | -0.1351437 | 0.1471332 |

In case of close peaks with overlapping shape vectors, a weighted minimum-mean-squared error is used in the VQ search. Zero weights are assigned to the overlapping shape elements that belong to the peak with lowest amplitude. In this way the CB entries are matched against the meaningful coefficients only.

After the peak regions are extracted and quantized, all remaining bits that are not reserved for signalling are used to quantize the low frequency MDCT coefficients. This is done by grouping the remaining un-quantized MDCT coefficients into 24-dimensional bands (not including already coded peak regions). A bit budget for coding the first band always exists, but the total number of coded bands depends on the bits left after peak coding. The number of bands to be coded  with remaining bits is determined by dividing the number of available bits by the maximum number allowed per-band, which is set to at 24.4 kb/s and at 32 kb/s.

 (1216)

Then the selected bands are gain-shape quantized. Gains are SQ on log domain with 5 bits, and shape are PVQ quantized as described in subclause 5.3.4.2.7.

A coded band is introduced above 5.6 kHz for 24.4 kb/s and 8 kHz for 32 kb/s if energy in the high band is relatively high compared to the peak coded region in the lower band, the band has high energy compared to the neighbouring high-frequency bands, and there is sufficient number of bits for encoding band of that size.

To encode the high-frequency band **with band log energy**and bandwidth ** following conditions have to be simultaneously met:

 (1217)

where an estimate of the low band energy  is obtained through summation over all peak amplitudes, coded at low-frequencies:

 (1218)

and  denotes the number of bits required to encode one pulse in a band of width . The average band log energy at high-frequencies is denoted by **and the amount available bits by **.

void (1219)

The two noise floor gains use in the low-frequency noise-fill and are scalar quantized with 5 bits on a log domain. The high-frequency gains and are transmitted to the decoder for the high-frequency noise-fill. First two variables: the noise-floor and peak-energy gains  and  are calculated, in similar way as described in subclause 5.3.4.2.3.1, but over high-frequency MDCT coefficients (coefficients above 224 at 24.4 kb/s and above 320 at 32 kb/s). That is the summation is done over for 24.4 kb/s and for 32 kb/s. Then the two gains are calculated as:

 (1220)

These gains are quantized with 2 bit uniform SQ to form quantized high-frequency gains and transmitted and used in the high-frequency noise-fill.

##### 5.3.4.2.6 Generic Mode

The first stage in this mode is the selection of one of three types of high frequency excitation class which is followed by the separate encoding of the low and high frequency envelopes. The low frequency envelope is quantized and coded in the same manner as the normal high rate mode of the HQ coder. The high frequency envelope is first quantised in the generic mode domain before being mapped onto the HQ normal mode domain, re-quantized and then combined with the low frequency envelope. An initial bit allocation is determined, and then delta’s are calculated and coded. The coded values are used to update the combined envelope, before the final bit allocation is determined.



Figure 82: Generic mode Encoder Block Diagram

5.3.4.2.6.1 Band allocation for the Generic Mode

The band allocation for the low frequency envelope at 24.4 and 32kbps is the same as default mode band allocation defined in table 127. The band allocation for the high frequency envelope at 24.4 and 32kbps is shown in table 144.

Table 144: Band Allocation for the high frequency envelope

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Band index | SWB, FB @ 24.4kbps | | | SWB, FB @ 32kbps | | |
|  |  |  |  |  |  |
| 0 | 320 | 335 | 16 | 384 | 399 | 16 |
| 1 | 336 | 359 | 24 | 400 | 423 | 24 |
| 2 | 360 | 375 | 16 | 424 | 439 | 16 |
| 3 | 376 | 399 | 24 | 440 | 463 | 24 |
| 4 | 400 | 415 | 16 | 464 | 479 | 16 |
| 5 | 416 | 439 | 24 | 480 | 503 | 24 |
| 6 | 440 | 455 | 16 | 504 | 519 | 16 |
| 7 | 456 | 479 | 24 | 520 | 543 | 24 |
| 8 | 480 | 503 | 24 | 544 | 567 | 24 |
| 9 | 504 | 527 | 24 | 568 | 591 | 24 |
| 10 | 528 | 551 | 24 | 592 | 615 | 24 |
| 11 | 552 | 575 | 24 | 616 | 639 | 24 |
| 12 | 576 | 607 | 32 | 640 (FB) | 679 (FB) | 40 (FB) |
| 13 | 608 | 639 | 32 | 680 (FB) | 719 (FB) | 40 (FB) |
| 14 | 640 (FB) | 679 (FB) | 40 (FB) | 720 (FB) | 799 (FB) | 80 (FB) |
| 15 | 680 (FB) | 719 (FB) | 40 (FB) | - | - | - |
| 16 | 720 (FB) | 799 (FB) | 80 (FB) | - | - | - |

5.3.4.2.6.2 High frequency Excitation Class

There are three different high frequency excitation classes, one each for speech, tonal music and non-tonal music. The *HF\_Speech\_excitation\_class* is determined by the instantaneous result of the speech/music classifier, i.e. by applying the output of the first speech/music classifier without adding any hang-over in subclause 5.1.13.6.3.

 (1221)

If the output of the classifier indicates music then a tonality measurement is calculated.

 (1222)

where  is the number of bands for calculating tonality, 10 at 24.4kbps and 8 at 32kbps.

This tonality value is then thresholded to further subdivide the excitation into *HF\_excitation\_class0* for noisy signals or *HF\_excitation\_class1* for tonal signals. The bit allocations are shown in table 145.

Table 145: Bit Allocation for the HF Excitation Classes

|  |  |  |
| --- | --- | --- |
| High Frequency Excitation Classes | Code | Num of bits |
| HF\_excitation\_class0 | 00 | 2 |
| HF\_Speech\_excitation\_class | 1 | 1 |
| HF\_excitation\_class1 | 01 | 2 |

5.3.4.2.6.3 Low Frequency Envelope Quantization and Coding

The low frequency envelope is quantized and coded in the same manner as described for the normal high rate mode of the HQ coder, subclause 5.3.4.2.1.1

For both SWB and FB the transmitted low frequency envelope is approximately 8 kHz at 24.4 kbps and increases to approximately 9.6 kHz at 32kbps. The number of bands transmitted by the low frequency envelope quantization and coding is defined as 27 at 24.4kbps and 30 at 32kbps.

5.3.4.2.6.4 High Frequency Envelope Quantization

The high frequency envelope is quantized in a similar manner as described in subclause 5.2.6.2.1.5, but with different starting frequencies. For SWB at 24.4kbps the frequency range is 8 to16 kHz with 14 dimensions, while at 32kbps the range is 9.6 to 16 kHz with 12 dimensions. For FB at 24.4kbps the frequency range is 8 to 20 kHz, while at 32kbps the range is 9.6 to 20 kHz with 3 additional bands.

First the high frequency envelope is calculated, and then the energy control tool is applied as described in subclause 5.2.6.2.1.5 at SWB bands (14 bands at 24.4kbps, 12 bands at 32kbps) for both SWB and FB. When performing energy control, the variable from equation (715) in subclause 5.2.6.2.1.5, is set to 0.55. However, the copied spectrum used to generate the simulated spectrum is generated by using the frequency mapping as defined in the following table, rather than the mapping used in subclause 5.2.6.2.1.5 and defined in table 60.

Table 146: Frequency mapping to generate base excitation spectrum

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| BW, Bit-rate | *l* |  |  |  |  |
| SWB, FB @ 24.4kbps | 0 | 2 | 239 | 320 | 447 |
| 1 | 2 | 239 | 448 | 575 |
| 2 | 80 | 143 | 576 | 639 |
| SWB, FB@ 32kbps | 0 | 2 | 239 | 384 | 511 |
| 1 | 2 | 239 | 512 | 639 |

VQ is then applied as described in subclause 5.2.6.2.1.5. At 24.4kbps the VQ is identical to that used at Non-TRANSIENT mode, but at the 32kbps, the VQ is modified as shown in figure 83:



Figure 83: VQ for HQ generic mode at 32kbps

In the first stage, three candidate indices are chosen using the weighted mean squared error minimization criterion. The 6 values in even positions, as well as the last (11th) position are selected and quantized using a 7 dimensional VQ with 5 bits.

 (1223)

The quantization error is calculated:

 (1224)

where is the de-quantized value.

Then the errors are split into  and and quantized:

 (1225)

The two quantized and de-quantized values  and  are then combined:

 (1226)

At odd positions (excluding position 11), an interpolation using boundary values is applied for intra-frame prediction and the predicted error is calculated and quantized:

 (1227)

is then split into and  and each is then quantised to 5 bits with a 3 dimensional VQ.

For FB, the energies for three additional bands are calculated. These are then quantised using 5 bits, after subtracting the mean vector (shown in table 147)

Table 147: Mean vector in FB

|  |  |
| --- | --- |
| *j* | FB |
| 0 | 13.75 |
| 1 | 6.29 |
| 2 | 3.70 |

The final selected set of indices  for SWB, or  for FB are then transmitted.

5.3.4.2.6.5 High Frequency Envelope Refinement and Fine structure quantization

After de-quantisation the value in each band of the de-quantized high frequency envelope  is mapped to one of the HQ high rate normal mode bands in order to match the frequencies. The following energy displacement factors are used:



The mapping for SWB at 24.4kbps is:

 (1228)

The mapping for SWB at 32kbps is:

 (1229)

The mapping for FB is:

 (1230)

The resulting mapped envelope is then re-quantized using the same method as is used for the HQ high rate normal mode, as described in subclause 5.3.4.2.1.1.

The quantized high and low frequency envelopes are then combined, and an initial fractional bit allocation is carried out. The initial bit allocation for the SWB generic mode is described in 5.3.4.2.6.6. However, in the first step, equation (1235) is replaced with the following equation:

 (1231)

The initial bit allocation used for FB generic mode is the same as the bit allocation used for the Normal mode as is described in 5.3.4.2.1.3.

The information obtained from the bit allocation indicates whether or not the envelope refinement is required for the current frame. If there are any high bands which have allocated bits, delta coding needs to be done to refine the high frequency envelope. In other words, if there are any important spectral components in the higher bands, the refinement is performed to provide a finer spectral envelope. If there are no bits allocated to the higher bands during the initial bit allocation, the envelope refinement is not required and the initial bit allocation is used. In this case the envelope requires no further modification so the following steps are not required and the fine structure quantization is applied to spectrum to quantize the coefficients as is described in subclause 5.3.4.2.1.3.

The quantized and de-quantized norms are calculated with the scalar quantizer as shown in subclause 5.3.4.2.1.1 by using the original spectrum, which it is defined to. The scalar quantized and de-quantized norms are also calculated using the mapped norms with the vector quantization, which is defined to.

Deltas are calculated at every band with allocated bits:

 (1232)

A possible bit for representing allspans 2,3,4 and 5 bits, and they are encoded as  using 2 bits. So, the possible bit is calculated with  and it then is adjusted to fit within the maximum number of bits (5). Any  which exceeds the maximum value is corrected to 31 or -32 depending on a sign of .

 (1233)

The corrected  is transmitted with .

The value of is then used to compute a final update to the envelope.

 (1234)

where is the updated norm.

The fine structure quantization is then applied to the resulting spectrum to quantize the coefficients as is described in subclause 5.3.4.2.1.3.

Finally the initial bit allocation information is updated, based on the number of bits used for representing the deltas. This is done by reducing the bits allocated to some sub bands, to provide enough bits to code the deltas. If during the initial bit allocation a sub band was allocated more than 3 bits, its allocation is reduced by one bit until all the bits required for the deltas have been accounted for. This is shown in more detail in the pseudo code below.

while (Bits\_needed\_for\_deltas > 0)

{

Current\_band = number\_of\_bands-1

while(bits\_needed\_for\_deltas >0 and Current\_band >= 0)

{

if (Bits\_allocated\_to\_current\_band > 3)

{

Bits\_allocated\_to\_current\_band --

Bits\_needed\_for\_deltas --

}

Current\_band --

}

}

This procedure provides the final bit allocation.

5.3.4.2.6.6 Bit Allocation

In the Generic mode, a fractional bit allocation is used for the spectral quantizer bit allocation. This permits the allocation of bits with three bits fractional parts. Initially bits for each band are estimated by:

 (1235)

where is a total bit budget.

The fully allocated bits are calculated as a starting point and the first-stage iterations are done to re-distribute the allocated bits to the bands with non-zero bits until the number of fully allocated bits is equal to the total bit budget.

 (1236)

where is the number of spectral lines in all bands with allocated bits after *k* iterations.

If too few bits are allocated, this can cause a quality degradation due to the reduced SNR. To avoid this problem a minimum bit limitation is applied to the allocated bits. The first minimums for the bits consist of constant values depending on the band index and bit-rate.

 (1237)

In the second-stage iterations, the re-distribution of bits is done again to allocate bits to the bands with more than  bits. The value  bits indicates the number of bits that corresponds to 1 bit/sample in band  and is the second minimum required bits for each band. Initially, the allocated bits are calculated based on the result of the first-stage iteration and the first and second mininum required bits for each band.

 (1238)

where is the allocated bits after the first-stage iterations andis 2 at 24.4kbps and 3 at 32kbps.

The  is updated by subtracting the number of bits in bands with  bits , and the band index  is updated to  which indicates the band indices with higher bits than  bits. is updated to  which is the number of bands for . The second-stage iterations are then done until the updated () is equal to the number of bits in bands with more than  bits.

 (1239)

where is the number of spectral lines in all bands with more than  bits after *k* iterations.

During the second-stage iterations, if there are no bands with more than  bits, the bits in bands with non-zero allocated bits from the highest bands are set to zero until  is equal to zero. Then the iterations are terminated.

Then, a final re-distribution of over-allocated and under-allocated bits is performed and finally, the fractional parts of bit allocation are adjusted to have three bits.

 (1240)

##### 5.3.4.2.7 Pyramid Vector Quantization (PVQ) and indexing

The PVQ is a lattice quantizer, with complexity growing linearly with the vector dimension. The PVQ quantizes a dimensional vector by allocating signed pulses to match the shape of that vector. In this way the PVQ codebook is defined as a combination of signed pulses in adimensional space (with pulses at the same position having the same sign). The maximum size of PVQ codebooks is set to 32 bits, and if allocated bits are more than 32, the vector is split into sub-vectors, bits re-distributed, and a gain parameter is quantized to represent relative energy between these sub-vectors. Largest allowed target vector is of dimension 64.

5.3.4.2.7.2 PVQ split methodology

When the bits assigned for the band are above a pre-determined threshold, as described in subclause 5.3.4.2.7.2.1, an algorithm for band splitting is activated. First the input vector is split into uniform (or close to uniform) segments in a non-recursive way. Then angles, which represent the ratio between energies andof a left and a right level segment, are calculated recursively. At each iteration, the level segments consist of one or several of the pre-determined segments from the initial split of the input vector. The angles are calculated from the top level (full size of the band to be quantized), and continuing towards the levels of shorter sub-vectors, i.e. shorter level segments.

The angle is determined from the energy ratio between one left and one right level segment. In case of an even number of splits, the left and right level segments will consist of an equal number of segments. In case of odd number of splits, the angle calculated during the first iteration of the recursion will be with the right segment having a larger number of segments than the left level segment, and the recursion will require more steps for the right level segment.

For example if the bit budget  for the band size require split in 3 segments (here assuming that a split into equal sized segments is possible), the angle calculation and the bit distribution is done in the following way:

Step 0: the angle and the shape bits are, the left level segment is of length, while the right level segment is of length, and bits for the two level segments are and.

Step 1: the right level segment from the previous step is split in a left and a right level segment, both with the length, the angle and the shape bits to be distributed are , the two level segments (here equal to the second and third segment of the initial split of the input vector) are allocated the bits  and.

The angles are used to distribute bits recursively to the already determined segments. At each iteration, the number of bits  and  for a left and a right level segment are derived from the available number of bits for shape coding of these segments , lengths of the level segments and , and the angle :

 (1241)

 (1242)

where  and  are compensation factors for differences in segment lengths within the level segments, defined as

, (1242a)

,  (1242b)

where the function  gives the number of unit pulses that giving the segment lengths  and  of the segments  within the left and the right level segment respectively, can be represented using  bits.  is the total number of pre-determined segments/splits within the left and right level segments. The function  gives the number of bits used to represent the by the function  determined number of unit pulses for the minimum dimension among the pre-determined segments.

The angle, which captures the relation between the energies of the left and right segment at certain split level, is defined as:

 (1243)

These angles are calculated recursively, starting with the top level, and continuing towards the levels of shorter sub-vectors. The angles are quantized by a range coder with asymmetric triangular PDF.

5.3.4.2.7.2.1 Band splitting analysis

The number of segments (parts) in the split PVQ vector is determined in two steps. First, an initial number of segments  is computed as

 (1244)

where is the maximum bitrate for each PVQ vector segment. If  and the band bit rate is high, an additional split is considered. Using segments, the energies of each PVQ target vector segment is computed.

 (1245)

If the maximum absolute deviation from the mean energy ,

 (1246)

is larger than the difference between the maximum number of bits for each segment and the average bit rate for each segment, an additional split will be added. That is,

 (1247)

Depending on if the conditions for the additional split are met, the split increment flag  or is signalled in the bitstream using 1 bit.

5.3.4.2.7.3 PVQ sub-vector shape search and shape normalization

5.3.4.2.7.3.1 PVQ-search introduction

The goal of the search procedure is to find the vector , which is defined as:

 (1248)

whereis a point on the surface of an -dimensional hyper-pyramid and the L1 norm of is . I.e. is the selected integer shape code vector of size ***N*** according to:

 (1249)

i.e.  is the unit energy normalized integer sub vector  scaled with a sub vector splitting gain , (if the band is not split in sub vectors the gain  is *)*.

The best vector is the one minimizing the mean squared shape error between the target vector and the scaled normalised quantized output vector*.* The target vector can be the time domain or in the frequency domain. Finding the best is achieved by minimizing the following search distortion:

 (1250)

By squaring the numerator and denominator and eliminating the predetermined constant gain scale factor, we may maximize the quotient:

 (1251)

The L1-norm structured PVQ-quantizer, allows for several search optimisations, where the primary optimization is to move the target to the all positive “quadrant” in -dimensional space and the second optimization is to use an L1-norm projection as a starting approximation for. A third optimization is to iteratively update the  quotient, instead of re-computing equation (1251) over the whole vector space for every candidate change to the vector  in pursuit of reaching the L1-norm , which is required for the subsequent PVQ-indexing step .

In the search of the optimal PVQ vector shape with L1-norm , iterative updates of the variables are made in the all positive “quadrant” in -dimensional space according to:

 (1252)

 (1253)

where signifies the correlation achieved so far by placing the previous unit pulses, and  signifies the accumulated energy achieved so far by placing the previous unit pulses, and  signifies the amplitude of at position  from the previous placement of unit pulses. To further speed up the in-loop iterative processing the energy term is scaled down by 2, thus saving one multiplication in the inner-loop.

 (1254)

 (1255)

The best position for the ’th unit pulse, is iteratively updated by iterating  over .

 (1256)

To avoid divisions the  maximization update decision is performed using a cross-multiplication of the saved best squared correlation numerator  and the saved best energy denominator  so far.

 (1257)

The iterative maximization of may start from a zero number of placed unit pulses or from an adaptive lower cost pre-placement number of unit pulses, based on an integer projection to a point below the ’th-pyramid’s surface, with a guaranteed undershoot of unit pulses in the target L1 norm .

Due to the structured nature of the PVQ integer sub vector, where all possible sign combinations are allowed and it is possible to encode all sign combinations, as long as the resulting vector adheres to the L1 norm of  unit pulses, the search is performed in the all positive first *“quadrant”*. Further to achieve as a high accuracy as possible for a limited precision implementation the maximum absolute value of the input signalis pre-analysed for future use in the setup of the inner search loop precision.

 (1258)

 (1259)

In case the input vector is an all zero vector or the sub-vector gain is very low, the PVQ-search is bypassed, and a valid PVQ-vector  is deterministically created by assigning half of the unit pulses to the first position ( ), and the remaining unit pulses to the last position (  ). (With this approach PVQ-search complexity is reduced and the indexing complexity is spread between encoder indexing and decoder de-indexing.)

5.3.4.2.7.3.4 PVQ pre-search projection

If the pulse density ratio is larger than 0.5 unit pulses per coefficient and is larger than *1*, a projection to thesub pyramid is made and used as a starting point for ***y***, on the other hand if the pulse density is less than 0.5 or is *1*, the iterative PVQ-search will start off from *zero* pre-placed unit pulses. The projection is performed as:

 (1260)

 (1261)

If no projection is made the starting point is an all zeroed vector. In preparation for the fine search to reach the’th-pyramid’s surface the accumulated number of unit pulses, the accumulated correlation and the accumulated energy for the starting point  is computed as:

 (1262)

 (1263)

 (1264)

 (1265)

5.3.4.2.7.3.5 PVQ fine search

The final integer shape vector  must adhere to the L1 norm of pulses. The fine search starts from a lower point in the pyramid and iteratively finds its way to the surface of the ***N***-dimensional ’th-hyperpyramid. The -value in the fine search ranges from 1 pulse to 512 unit pulses, to keep the complexity of the search at a reasonable level, the search is split into two main branches, one branch is used when the in-loop energy representation of will stay within a signed 16 bit word, and another branch is used if the in-loop energy may or may not exceed the dynamic range of a 16 bit word.

When the final is lower than or equal to 127 unit pulses, the dynamics of the 1 bit upshifted will always stay within 15 bits, allowing efficient use of a signed 16 bit word for representing every within all the fine pulse search inner loop iterations up to .

In preparation for the next unit pulse addition, the near optimal maximum possible upshift of the next loop’s accumulated in-loop correlation value in a signed 32 bit word is pre-analysed using the previously calculated maximum absolute input value as:

 (1266)

To make the critical equation (1257) update as efficient as possible, the numeratoris represented by a 16 bit signed word, by the following approach:

 (1267)

 (1268)

where the function “Round16” extracts the top 16 bits of a signed 32 bit variable with rounding, this near optimal upshift and the use of 16 bit representation of the squared correlation ***bestCorrSq16*** enables a very fast inner-loop search for performing the equation (1268) test and variable updates.

The location of the next unit pulse is now determined by iterating over the possible positions, while employing equations (1267), (1253) and (1268).

When the best position  of the unit pulse has been determined , the accumulated correlationthe accumulated inloop energy and the number of accumulated unit pulses are updated. If there are further unit pulses to add , a new inner-loop is started with a new near optimal  analysis (equation (1266)) for the next unit pulse.

When the final  is higher than 127 unit pulses, the dynamics of the 1 bit upshifted may exceed 15 bits, the fine search will adaptively choose between 16 bit representation and 32 bit representation of the pair  . The fine search will keep track of the maximum pulse amplitude in achieved so far, this information is used in a pre-analysis step before entering the optimized inner dimension loop, to determine the precision to use for the inner unit pulse addition loop. If the pre-analysis indicates that more than a signed 16 bit word is needed to represent the in-loop energy without losing any energy information, a high precision unit pulse addition loop is employed, where both the saved best squared correlation term and the saved best accumulated energy term are represented by 32 bit words.

 (1269)

 (1270)

If is *FALSE* the lower precision inner search loop in equations (1267), (1253) and (1268) is employed, on the other hand if is *TRUE*, the location of the next unit pulse is determined by iterating over the possible positions, using equations (1271), (1253) and (1272).

 (1271)

 (1272)

When the best position  of the unit pulse has been determined, the accumulated correlation, the accumulated inloop energy and the number of accumulated unit pulses are updated. Further the maximum amplitude  in the best integer vector so far, is kept up to date for the next unit pulse addition loop.

 (1273)

If there are further unit pulses to add , a new inner-loop is started with a new near optimal  analysis equation (1266) and a new energy precision analysis equations (1269) and (1270) and then commencing the next unit pulse loop equations (1271), (1253) and (1272)).

The effect of the in-loop accumulated energy based inner loop precision selection is that target sub vectors that have a high peakiness, or have very fine shape granularity (finalis high) will be using the higher precision loop, while non-peaky or lower shape granularity sub vectors will more often use the lower precision loop.

5.3.4.2.7.3.6 PVQ sub-vector finalization and normalization

After shape search each non-zero PVQ-sub-vector element is assigned its proper sign and the vector is L2-normalized to unit energy. Additionally if the band was split, it is further scaled with the previously determined sub-vector gain.

 (1274)

 (1275)

 (1276)

5.3.4.2.7.4 PVQ short codeword indexing

The PVQ short codeword indexing assigns a unique index to any possible vector in. The incoming vector, which represents a point on the *K*’th hyper-pyramid in *N*-dimensional space is indexed into two short codeword(s) using a leading sign sections Modular PVQ recursive indexing method denoted. After the MPVQ indexing procedure, the integer codewords and the size of each codeword are sent to a Range encoder, an arithmetic encoder, which in turn provides a stream of arithmetically encoded bits to the to the bit stream.

5.3.4.2.7.4.1 MPVQ modular leading sign recursion definition

The first codeword represents a leading sign, which is the sign of the first non-zero position in, the second codeword is a recursive representation of the amplitudes (including zeroes) and leading signs of the remaining vector  after the first occurring non-zero position sign in has been extracted. The modular leading sign extraction is performed recursively until all amplitudes (including zeroes) and signs have been consumed.

The number of possible combinations ofis denoted. The number of possible combinations of, anvector with unit pulses in dimensionwith an L1-norm of  and an initial positive sign is denoted. The elimination of the initial leading sign from yields the number of possible entries in as:

 (1277)

The MPVQ-index for vector  is based on the recursive scheme where we:

Define as the number of integer vectors of dimension and L1-norm of, that does not have a leading zero, and does not have the leading value *k***,** has a leading positive value, and has a positive leading sign, (where the leading sign is the first sign encountered after the current value in the direction of the recursion.

Define as the number of integer vectors of dimension and L1-norm of , that has a positive leading value and that does not have a leading zero.

See table 148 for a structured view of  and its relation to the total number of vectors in the structure with vectors.

Table 148: MPVQ recursion overview: the three subsections for theleading sign based recursion

| Subsection  ( initial value for the subsection ) | Number of entries for each subsection |
| --- | --- |
|  | 1 |
| All unit pulses consumed.  Remainingvalues all zeroes, recursion may stop |
| (non-zero with a positive or negative next leading sign) |  |
| First encountered non-zero position requires 1 bit of next leading sign information,  stored as the LSB in this even sized sub-section.  The recursion consumes the  amplitude value and moves on to |
|  |  |
| Recursion moves on to  without any additional next leading sign information |

The recursive definition of thetotal number of entries in becomes:

 (1278)

By applying Fischer’soriginal recursion fortogether with equation (1277) and (1278) , we establish that:

 (1279)

, from the definition (and table 148) one can find that the relation betweenandis:

 (1280)

where ( the “1” comes from the single initial-“*k*” valued vector, and the factor “2” is due to positive and negative sign possibilities of the coming next leading sign.

On the encoder side recursive relations are used for the forward update of the MPVQ indexing offsets and as the dimension increases up to the size of the vector to be indexed:

 (1281)

 (1282)

where the low dynamic range calculation equation (1282) is used for the last column  forward update of the MPVQ offset matrix, a column which is needed for the calculation of the exact size. Anbased recursion is used for the offset matrix columns, as these columns are known in advance to have a low dynamic range. To keep the dynamic range within a signed 32 bit word, the finalsize calculation is performed as:

 (1283)

5.3.4.2.7.4.2 Detailed MPVQ indexing approach

The maximum index range of the found PVQ-vector ’s is [0.. 232-1 ] and the total resulting index would require an unsigned 32 bit word, however with the introduction of the modular leading sign approach the maximum index range is reduced to [0.. 231-1 ], which enables fast implementation using signed 32 bit word arithmetic.

The MPVQ-indexing scheme on the encoder side is run in the order of position  to position 0, (on the decoder side the de-indexing will start from position 0 and end at position). The MPVQ indexing loop is carried out according to figure 84, where  is the row number of the MPVQ offset matrix,is the pointer into the samples/coefficients of the PVQ-vector , “>>1” denotes a 1 bit right shift and further the function “UpdateOffsetsFwd” iteratively updates the required MPVQ-offsets for the next larger dimension using combinations of equations (1281), (1280) and (1282).



Figure 84: Detailed MPVQ-indexing, leading sign index extraction and size calculation

To speed up the indexing when the number of unit pulses is high (and the dimensionis low), direct functions are used to calculate both the MPVQ-offsets and the MPVQ-size.

5.3.4.2.7.5 High Dynamic Range Arithmetic Encoding

The Split PVQ scheme relies on a Range encoder to encode symbols with higher granularity than bits. A high dynamic range implementation, where of up to 24 bits resolution may be used for the CDF is used to encode short codewords in three distinct ways:

* Direct bit encoding
* 1/8 bit resolution uniform PDF encoding
* Asymmetric triangular PDF encoding of band split energy angles, using tailor made CDFs.

Further special range encoder functions are used to determine the remaining bit budget in the frame and band quantization loops, and used to correct the bit budget for coming frames or coming bands.