## 6.2 MDCT Coding mode decoding

### 6.2.1 General MDCT decoding

MDCT based codec frames can be either encoded by the MDCT based TCX or the High Quality Coder. Which mode is actually used, is determined by table 76 in subclause 5.3.1. For those configurations where both modes are switched, a signaling bit determines the MDCT mode.

### 6.2.2 MDCT based TCX

A general description of the MDCT based TCX coder module can be found in subclause5.3.3.1. In the following, the configurations are described, the initialization process from the bit stream data and the general decoding process.

#### 6.2.2.1 Rate dependent configurations

The rate dependent configuration of the TCX coder is described for the encoder in subclause 5.3.3.1.2. The configurations are valid for the decoder as well. Depending on the configuration, the initialization order of the modules changes. The following subclause describes the module initialization therefore just on a principle level.

#### 6.2.2.2 Init module parameters

##### 6.2.2.2.1 TCX block configuration

The coding modes TCX20 or TCX10 are signalled within the bit stream. Binary code for the overlap width as defined in subclause 5.3.2.3 is read from the bitstream. Overlap code for the current frame is formed from the short/long transform decision bit and from the binary code for the overlap width as defined in subclause 5.3.2.3. The TCX block is then configured as described in subclause 6.2.4.2.

##### 6.2.2.2.2 LPC parameter

This subclause describes the inverse quantization of LPC.

6.2.2.2.2.1 Low-rate LPC

MDCT based TCX relies on smoothed LPC spectral envelope. Decoding process of LSF is common to that for ACELP except the case for 9.6 kbps (NB/WB/SWB). Decoding process of weighted LSF and conversion of LSF used for 9.6 kbps (NB/WB/SWB) are described in this subclause.

Weighted domain quantized LSF vector is reconstructed with two stage or three stage VQ as described in 5.3.3.2.1. These VQ codebooks are used in two ways. In case of primary decoding, weighted LSF vector is retrieved from two stage VQ and after adding mean vector and the MA predicted contribution vector, reconstructed LSF vector, is obtained. is corresponding to the weighted envelope and can be directly used for inverse shaping of MDCT coefficients. The VQ uses 5 bits for 16 samples at the first stage, 4 bits for the lower 6 LSF parameters, and 4 bits for the higher 10 LSF parameters.

In the secondary decoding, weighted LSF vector is reconstructed without adding MA predicted contribution vector. 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. The conditional third stage VQ with addition of mean vector is applied to the retrieved vector from two stage VQ to get . After reconstruction of LSF vector with two stage VQ and adding mean vector, the first and the second lower position and of the reconstructed LSF vector are checked. If the is expected to have large spectral distortion from , the third stage VQ is applied. The retrieved vector at the third stage VQ with 2 bits is applied to modify and to get , only when, or has smaller values than the threshold. Otherwise, the reconstructed LSF up to the second stage VQ, is used for the final reconstruction LSF, .

For the interpolation of LSF between 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 . In envelope based arithmetic coding, unweighted LSF vector without depending on MA prediction is also necessary to estimate the MDCT envelope. These can be achieved by low-complex direct matrix conversion described in 5.3.3.2.1.1

6.2.2.2.2.2 Mid-rate LPC

The inverse quantization for mid-rate bitrates (13.2 till 32 kbps) is the same as for ACELP (with the quantizer being in AUDIO mode). However there is a different interpolation, the same as described in subclause 5.3.3.2.1.2.

6.2.2.2.2.3 High-rate LPC

Inverse quantization of an LPC filter is performed as described in figure 93. The LPC filters are quantized using the line spectral frequency (LSF) representation. A first-stage approximation is computed by inverse Vector Quantization by a simple table look-up with an 8 bit index. An algebraic vector quantized (AVQ) refinement is then calculated as described in 6.1.1.2.1.4. The quantized LSF vector is reconstructed by adding the first-stage approximation and the inverse-weighted AVQ contribution.

Depending on the frame being coded as a single TCX20 frame or subdivided into TCX10/TCX5 sub-frames one or two sets of LPC have to be de-quantized. In case two sets are transmitted the first set is decoded in the same way a single set would be decoded. For the second set an initial bit signals if it has to be decoded depending on the first set or not. If zero, the first stage approximation of the second set has to be decoded with another 8 bit inverse Vector Quantizer. If one, the inverse quantized first set will be used as first stage approximation. The inverse weights in figure 93 are the reciprocal of the weights used in the encoder (see subclause 5.3.3.2.1.3.3).

The AVQ decoder is described in subclause 6.1.1.2.1.4.



Figure 93: Overview high-rate LPC decoding

The inverse-quantized LSF vector is subsequently converted into a vector of LSF coefficients, then interpolated and converted again into LPC parameters. The interpolation is the same described in 5.3.3.2.1.2.

##### 6.2.2.2.3 PLC Wavefrom adjustment

The waveform adjustment tool reads one bit for initialization where 1 stands for harmonic and 0 for non-harmonic.

##### 6.2.2.2.4 Global Gain

The TCX global gain index is transmitted in the bitstream as a 7 bit unsigned integer.

##### 6.2.2.2.5 Noise fill parameter

The TCX noise factor index is transmitted in the bitstream as a 3 bit unsigned integer.

##### 6.2.2.2.6 LTP

The LTP on/off flag is transmitted in the bitstream as a single bit. If the LTP flag is one, the quantized pitch lag and gain are transmitted after the LTP flag. The pitch lag index is transmitted as unsigned 9 bit integer. The gain index is transmitted as unsigned 2 bit integer.

##### 6.2.2.2.7 TNS parameter

The TNS on/off flat is transmitted in the bitstream as a single bit. If the TNS flag is one and if the configuration (see subclause 5.3.3.2.2) indicates that 2 filters are possible then an additional single bit transmitted in the bitstream indicates if the parameters for 1 or 2 filters are transmitted in the bitstream (nMaxFilters = 1 or nMaxFilters = 2). For each filter, order and coefficients are the parameters transmitted in the bitstream. The order and the coefficients are coded using Huffman coding. 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. If the parameters for 1 filter are transmited in the bitstream, then the second filter is inactive. If the parameters for 2 filters are transmitted in the bitstream, then order 1 and the filter coefficient set to 0 for the first filter indicates that the first filter is disabled and that only the second filter is active.

##### 6.2.2.2.8 Harmonic model

For both context and envelope based arithmetic coding, a harmonic model is used for efficient coding of frames with harmonic content. The harmonic 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.

In the above cases, no further signalling is used.

Otherwise, a single-bit harmonic model flag is read from the bit-stream. When the flag is non-zero, the decoding proceeds by reading the harmonic model interval parameter as follows.

6.2.2.2.8.1 Decoding of Interval of harmonics

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

6.2.2.2.8.1.1 Decoding interval depending on time domain pitch lag

According to the procedure in subclause 5.3.3.2.8.1.8.1.1, , , are set up, is read from the bit-stream, and finally is calculated.

6.2.2.2.8.1.2 Decoding 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 decoding of the interval with un-equal resolution is used. The 8-bit is read from the bit-stream and and are calculated as in subclause 5.3.3.2.8.1.8.1.1.

6.2.2.2.8.2 Decoding of gain

In case of envelope based arithmetic coding and Voiced coder type, a 2-bit gain index is read from the bit-stream.

##### 6.2.2.2.9 IGF bit stream reader

On the decoder side the IGF scale factors, the IGF whitening levels and the IGF temporal flatness indicator flag are extracted from the bit stream and subsequently decoded. The decoding of the IGF scale factors is described in subclause 6.2.2.2.9.2.

6.2.2.2.9.1 IGF whitening level decoding

The IGF whitening levels are decoded according to the following pseudo code:

nT = ;

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

= 0;

}

tmp = -1;

if () {

tmp = 0;

} else {

tmp = read\_bit();

}

if (tmp == 1) {

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

= ;

}

} else {

k = 0;

= decode\_whitening\_level(k);

tmp = read\_bit();

if (tmp == 1) {

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

= decode\_whitening\_level(k);

}

} else {

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

= ;

}

}

}

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

= ;

}

wherein the vector contains the whitening levelsfrom the previous frame and the function decode\_whitening\_level takes care of the decoding the actual whitening level from the bit stream. The function is implemented according to the pseudo code below:

tmp = read\_bit();

if (tmp == 1) {

tmp = read\_bit();

if (tmp == 1) {

return 2;

} else {

return 0;

}

} else {

return 1;

}

Finally the IGF temporal flatness indicator flag is extracted from the bit stream.

In case of a TCX10 frame (), the IGF sideinfo for both sub-frames is extracted from the bit stream prior to the IGF processing. Therefore, the decoded sideinfo for the individual sub-frames is stored in a temporary buffer, so that the sideinfo for the current sub-frame under IGF processing can be accessed via the temporary buffer.

6.2.2.2.9.2 IGF noiseless decoding of scale factors

The noiseless decoding of the IGF scale factor vector is very similar to the noiseless encoding part. The entire encoding and decoding procedures are highly symmetric, and therefore the decoding procedure can be uniquely and unambiguously derived from the encoding procedure.

The module uses the common raw arithmetic decoder functions from the infrastructure, which are available from the core coder. The functions used are , which decodes one bit into , , which decodes one value into from an alphabet of 27 symbols () using the cumulative frequency table , , which initializes the arithmetic decoder. Note that there is no to finalize the arithmetic decoder. Instead, the equivalent function was locally defined, which returns the last 14 bits read back to the bit stream reader.

6.2.2.2.9.2.1 IGF independency flag

The behaviour and processing related to the flag is identical to the encoder side.

6.2.2.2.9.2.2 IGF all-Zero flag

The flag is read from the bit stream first. In case the flag is 1, the decoder state is reset and no further data is read from the bit stream, because the decoded scale factors are all set to zero:

(1762)

Otherwise, if the flag is 0, the arithmetic coded scale factor vector is decoded from the bit stream.

6.2.2.2.9.2.3 IGF arithmetic decoding helper functions

6.2.2.2.9.2.3.1 The reset function

The behaviour and processing related to the reset function is identical to the encoder side.

6.2.2.2.9.2.3.2 The arith\_decode\_bits function

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

arith\_decode\_bits(nBits)

{

x = 0;

for (i = 0; i < nBits; ++i) {

ari\_decode\_14bits\_bit\_ext(&bit);

x = (x << 1) | bit;

}

return x;

}

6.2.2.2.9.2.4 IGF arithmetic decoding

The function decodes an integer valued prediction residual, using the cumulative frequency table , and the table offset .

The behaviour and processing related to the function is very similar to the corresponding function in the encoder.

arith\_decode\_residual(cumulativeFrequencyTable, tableOffset)

{

ari\_decode\_14bits\_s27\_ext(&val, cumulativeFrequencyTable);

if ((val != 0) && (val != SYMBOLS\_IN\_TABLE - 1)) {

x = (val - 1) + MIN\_ENC\_SEPARATE;

x -= tableOffset;

return x;

}

extra = arith\_decode\_bits(4);

if (extra == 15) {

extra\_tmp = arith\_decode\_bits(6);

if (extra\_tmp == 63) {

extra\_tmp = 63 + arith\_decode\_bits(7);

}

extra = 15 + extra\_tmp;

}

if (val == 0) {

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

} else { /\* val == SYMBOLS\_IN\_TABLE - 1 \*/

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

}

x -= tableOffset;

return x;

}

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

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

{

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

if (t == 0) {

if (f == 0) {

ari\_decode\_14bits\_s27\_ext(&pred, cf\_se00);

g[f] = pred << 2;

g[f] += arith\_decode\_bits(2); /\* LSBs as 2 bit raw \*/

}

else if (f == 1) {

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

g[f] = pred + arith\_decode\_residual(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) \*/

g[f] = pred + arith\_decode\_residual(cf\_se02[CTX\_OFFSET + ctx],

cf\_off\_se02[CTX\_OFFSET + ctx]);

}

}

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

if (f == 0) {

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

g[f] = pred + arith\_decode\_residual(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(x[f - 1] - prev[f - 1]); /\* Q(b - c) \*/

g[f] = pred + arith\_decode\_residual(

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

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

}

}

}

}

The cumulative frequency tables and the corresponding table offsets are initialized identically to the encoder side.

##### 6.2.2.2.10 Spectral data

The quantized spectral coefficients are read from the bit-stream by the means of arithmetic decoding.

##### 6.2.2.2.11 Residual bits

The left-over bits (to the target bit budget) after arithmetic decoding are read by the residual decoding module.

#### 6.2.2.3 Decoding process

##### 6.2.2.3.1 Arithmetic decoder

The arithmetic decoder is described by the following pseudo-code. It takes as input arguments the cumulative frequency table *cum\_freq[]* and the size of the alphabet *cfl*.

symbol = ari\_decode(cum\_freq[], cfl) {

if (arith\_first\_symbol()) {

value = 0;

for (i=1; i<=16; i++) {

value = (val<<1) | arith\_get\_next\_bit();

}

low = 0;

high = 65535;

}

range = high-low+1;

cum =((((int) (value-low+1))<<14)-((int) 1))/range;

p = cum\_freq-1;

do {

q = p + (cfl>>1);

if ( \*q > cum ) { p=q; cfl++; }

cfl>>=1;

}

while ( cfl>1 );

symbol = p-cum\_freq+1;

if (symbol)

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

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

for (;;) {

if (high<32768) { }

else if (low>=32768) {

value -= 32768;

low -= 32768;

high -= 32768;

}

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

value -= 16384;

low -= 16384;

high -= 16384;

}

else break;

low += low;

high += high+1;

value = (value<<1) | arith\_get\_next\_bit();

}

return symbol;

}

6.2.2.3.1.1 Context-based arithmetic decoder

The context-based arithmetic decoder reads the following data in the following order:

1. bits for decoding *lastnz/2-1*.
2. The entropy-coded MSBs bits
3. The sign bits
4. The residual quantization bits
5. The LSBs bits are read backwardly from the end of the bitstream buffer.

The following pseudo-code shows how the spectral coefficients *X[]* or are decoded. It takes as input argument the allocated bit budget *target\_bits* and the number of coded samples *lastnz*. The helper functions are given in encoder subsection from 5.3.3.2.8.1.2 to 5.3.3.2.8.1.2.

X=ari\_context\_decode(target\_bits,pi,hi,last\_nz) {

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

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

X[k]=0;

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

a=b=0;

(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);

/\* MSBs decoding \*/

for (lev=esc\_nb=0;;){

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

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

if(r<16) break;

/\*LSBs decoding\*/

a=(a)+read\_bit\_end() <<(lev));

b=(b)+ read\_bit\_end() <<(lev));

lev++;

esc\_nb=min(lev,3);

}

/\*MSBs contributions\*/

b1= r>>2;

a1= r&0x3;

a += (a1)<<lev;

b += (b1)<<lev;

/\*Dectect overflow\*/

if(nbbits>target\_bits){

break;

}

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

/\* Store decoded data \*/

X[a1\_i] = a;

X[b1\_i] = b;

}

/\*decode signs\*/

for (i=0; i<L; i++){

if(X[i]>0){

if ( read\_bit()==1 ){

X[i] = -X[i];

}

}

}

}

6.2.2.3.1.2 Envelope-based arithmetic decoder

The probability model is computed as described in the encoder subclause 5.3.3.2.8.1.2.3.

##### 6.2.2.3.2 Adaptive low frequency de-emphasis

A general description of ALFE can be found in subclause 5.3.3.2.4.1.

6.2.2.3.2.1 Adaptive de-emphasis algorithm 1

ALFE algorithm 1 reverses the encoder-side LF emphasis 1 (see subclause 5.3.3.2.4.2). First, as was done in the encoder, the minimum and maximum of the first nine 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 lowering of the lowest lines in x is performed such that the first line is attenuated by (max/(32 min))0.25 and the 33rd line is not attenuated:

tmp = 32 \* min;

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

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

for (i = 31; i >= 0; i--) { /\* gradual lowering of lowest 32 lines \*/

X[i] \*= fac;

fac \*= tmp;

}

}

Adaptive de-emphasis algorithm 2

ALFE algorithm 2 reverses the encoder-side LF emphasis 2 (see subclause 5.3.3.2.4.3) by checking for modifications to the quantized LF MDCT lines and undoing them. As was done in the encoder, the procedure is split into five steps:

* Step 1: first find first magnitude maximum at index i\_max in lower spectral quarter (*k* = 0 … / 4) for which |Xq[*k*]| ≥ 4 and modify the maximum as follows: Xq[i\_max] += (Xq[i\_max] < 0) ? 2 : -2
* Step 2: then expand value range of all X[*k*] up to i\_max by multiplying all lines at *k* = 0…i\_max–1 with 0.5
* Step 3: again find first magnitude maximum in lower quarter of spectrum if the i\_max found in step 1 is > -1
* Step 4: again expand value range of all X[i] up to i\_max as in step 2, but using the i\_max found in step 3
* Step 5: finish and always expand two lines at the latest i\_max found, i.e. at *k* = i\_max+1, i\_max+2. If the line magnitude at *k* is greater than or equal to 4, move it toward zero by two, otherwise multiply it by 0.5. As in the encoder all i\_max are initialized to –1. For details please see AdaptLowFreqDeemph() in tcx\_utils.c.

##### 6.2.2.3.3 Global gain decoding

The global gain is decoded from the index  transmitted in the bit stream as follows:

 (1763)

##### 6.2.2.3.4 Residual bits decoding

At 13.2 kbps and above the 3 first bits are used for refining the global gain. The variable *n* is initialized to *0*:

The following bits refine the non-zeroed decoded lines. 1 bit per non-zeroed spectral value is read. The rounding offset used in the first quantization stage with dead-zone is taking into account for computing the reconstructed points:

If at least 2 bits are left to read, a zeroed value is refined as:

##### 6.2.2.3.5 TCX formant enhancement

The TCX formant enhancement intends to replicate a behavior similar to that of the ACELP formant enhancement. It operates based on the LPC frequency-band gains, lpcGains[]. First, the square-root of each gain is computed. Then,

fac = 1 / min(sqrtGains[0], sqrtGains[1]);

k = 0;

for (i = 1; i < numGains - 1; i++) {

if ((sqrtGains[i-1] <= sqrtGains[i]) && (sqrtGains[i+1] <= sqrtGains[i])) {

step = max(sqrtGains[i-1], sqrtGains[i+1]);

step = (1 / step - fac) / (i - k);

sqrtGains[k] = 1;

fac += step;

for (j = k + 1; j < i; j++) {

sqrtGains[j] = min(1, sqrtGains[j] \* fac);

fac += step;

}

k = i;

}

}

where sqrtGains[] contains the square-roots of the lpcGains[], and numGains denotes the number of LPC gains. In order to complete the above algorithm for the last gain at *i* = numGains – 1, the following operation is executed,

step = min(sqrtGains[i-1], sqrtGains[i]),

and the above steps inside the if-condition, starting with “step = (1 / step – fac) / (i – k)”, are repeated. Finally, we set sqrtGains[numGains–1] = 1 and multiply the modified set of gains onto the decoded spectrum:

for (i = j = 0; i < j++) {

for (k = 0; k < / numGains; i++, k++) {

[i] \*= sqrtGains[j];

}

}

with being the decoded spectrum. Like its ACELP counterpart, TCX formant enhancement is only used at 9.6 kbps.

##### 6.2.2.3.6 Noise Filling

Noise filling is applied to fill gaps in the MDCT spectrum where coefficients have been quantized to zero. Pseudo-random noise is inserted into the gaps, starting at bin up to bin . The amount of noise inserted is controlled by a noise factor transmitted in the bit stream. To compensate for LPC tilt, a tilt compensation factor is computed. At each side of a noise filling segment a fadeout over bins is applied to the inserted noise to smooth the transition.

The start and stop bins and are determined as described in subclause 5.3.3.2.10.2 . is set to the same value as :

(1764)

Computation of the tilt compensation factor is described in subclause 5.3.3.2.10.1. The transition width is computed as described in 5.3.3.2.10.3.

6.2.2.3.6.1 Decoding of Noise Factor

The dequantized noise factor is obtained from the transmitted index as follows:

(1765)

6.2.2.3.6.2 Noise Filling Seed

The inserted noise is generated as a sequence of pseudo-random numbers, which is computed in a recursive way starting with a seed computed from the quantized MDCT coefficients :

(1766)

6.2.2.3.6.3 Filling Noise Segments

Determining the number and start/stop bins of noise filling segments is described in 5.3.3.2.10.2 and 5.3.3.2.10.4.

For each segment pseudo-random noise is generated and normalized, so that it has an RMS of one. Then tilt compensation, noise factor and transition fadeouts are applied. The resulting coefficients are inserted to the quantized MDCT spectrum and replace the zeroes in the noise filling segments. The following pseudo-code defines the exact procedure:

##### 6.2.2.3.7 Apply global gain and LPC shaping in MDCT domain

The decoded global gain factor is applied to all MDCT coefficients. The LPC shaping of the MDCT spectrum applied on encoder side is inverted by multiplying the spectral coefficients by the LPC shaping gains.

The computation of the shaping gains is performed in the same way as on encoder side, see subclause 5.3.3.2.3.2.

The following pseudo-code defines how global gain and LPC shaping are applied to the MDCT bins corresponding to the CELP frequency range:

For the remaining MDCT coefficients above the CELP frequency range (if any) the last LPC shaping gain is used:

(1767)

##### 6.2.2.3.8 IGF apply

6.2.2.3.8.1 IGF independent noise filling

IGF uses independent noise filling in the IGF range. Through independent noise filling, core coder noise filling is replaced by random noise which is de-correlated from the core coder noise filling. Therefore a vector is filled with either 0 or 1 by evaluating the TCX noise-filling routine from subclause 6.2.2.3.6.3 such that every subband, which is noise-filled by TCX noise-filling, represents a 1 in , all other entries are set to 0 in .

First, the total noise energy in the IGF source range in the decoded MDCT spectrum is calculated:

, (1768)

where . The noise indicated by is replaced according to the following formula:

(1769)

where contains copies of the spectrum with independent noise per copy, i.e. per IGF tile. For creating pseudo random numbers r(i), the random generator described in subclause 6.2.2.3.6.3 is used.

The energy of the inserted pseudo random numbers is measured with

. (1770)

Now the inserted noise is adjusted to the same energy level as the original noise. Therefore the correction factor is calculated:

(1771)

Using , the replaced noise is rescaled to match the original noise energy level in :

(1772)

6.2.2.3.8.2 IGF whitening generation

In order to remove possible formant structure of the tiled signal and to suppress strong tonal components the routine *IGF\_getWhiteSpectralData()* will be applied to the TCX spectrum if the bitstream element is 1 for any tile . The algorithm is a low complex simplification of the following formula:

(1773)

which is a division of the spectrum by the square root of a moving average calculated on the spectrum. denotes the TCX coefficient of the decoded core signal with index prior to application of the LPC filter.

Since the above formula would need a division and a square root operation per line – two complex operations (18 and 10 OPS) – the operations are done in logarithmic domain, while the logarithm is replaced with a low complex rounded integer logarithm to the basis 2.

(1774)

where

(1775)

The length of the moving average (MA) filter is 15 bins in total.

The range of bins on which this whitening operation has to be carried out is going from to , where is the index of the first scale factor band - 1. Because is always greater 7, the MA filter will be calculated from on. However, the calculation of the MA filter has to be different for the last bins below :

(1776)

If the bitstream element is 2 for any tile no core signal will be copied but a sequence of pseudo-random numbers will be used instead as described in subclause 6.2.2.3.6.2.

The seed for the pseudo random number generator (described in subclause 6.2.2.3.6.3) is derived from the TCX noise filling seed by:

(1777)

6.2.2.3.8.3 IGF envelope reconstruction

The IGF envelope reconstruction tool shapes the noise components filled into the gaps in the IGF range in order to adjust the spectral envelope as a function of the transmitted IGF scale factors.

6.2.2.3.8.3.1 Dequantizing IGF scale factors

For de-quantizing the IGF scale factors , transmitted in the bitstream, to the following mapping is applied:

(1778)

6.2.2.3.8.3.2 Refining IGF scale factor borders

In order to optionally smooth the transmitted scale factors along the frequency axis, a refinement of the IGF scale-factor borders is introduced:

(1779)

The de-quantized IGF scale factors shall be mapped:

(1780)

For simplicity, is also mapped:

(1781)

The IGF envelope refinement is active for bitrates 64 kbps for all operating modes WB, SWB, FB.

6.2.2.3.8.3.3 Collecting energies below

To stabilize energy distribution in the range of , energy below is collected:

(1782)

The energy is later used to adapt the first IGF scale factor band energy as described in subclause 6.2.2.3.8.3.7.

6.2.2.3.8.3.4 Collecting residual energies in IGF range

The residual energy determines the energy of the non-zero subbands in the IGF range:

(1783)

is therefore the energy of the de-quantized subband values above which are not quantized to zero by the tonal mask detection of the encoder described in subclause 5.3.3.2.11.5.

6.2.2.3.8.3.5 Collecting tile energies in IGF range

The tile energy determines the energy of the signal which is filled into the gaps in the IGF range:

(1784)

(1785)

where is the signal after filling the gaps in using the mapping function as described in subclause 5.3.3.2.11.1.8:

(1786)

is therefore the energy of the synthesized subband values above which are quantized to zero by the tonal mask detection of the encoder described in subclause 5.3.3.2.11.5.

6.2.2.3.8.3.6 Rescaling IGF scale factor band energies

The rescaling of the IGF scale factors has to be done in order to bring them on the correct energy level for the subsequent calculation of the IGF gains. In dependency of the refinement rescaling is applied on a scale factor band basis or on a group of scale factor bands. The rescaled IGF scale factor band energy is therefore called IGF destination energy .

In case refinement is not active, the rescaling is applied as follows:

(1787)

In case refinement is active, two subsequent scale factor band energies are mapped:

(1788)

6.2.2.3.8.3.7 Adaption of IGF scale factor band energies

The IGF scale factor band energies have to be adapted to fulfil the signal requirements. The first scale factor band energy is adapted to the energy below using as introduced in subclause 6.2.2.3.8.3.3:

(1789)

where is the adapted IGF scale factor band energy and is the adaption factor for the first scale factor band energy according to table 1:

Table 166: IGF scale factor band energy adaption factors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Bitrate | Mode |  |  |  |
| 9.6 kbps | WB | 0.7 | 0.8 | 0.6 |
| 9.6 kbps | SWB | 0 | 1.0 | 1.0 |
| 13.2 kbps | SWB | 0.2 | 0.93 | 0.85 |
| 16.4 kbps | SWB | 0.2 | 0.93 | 0.85 |
| 24.4 kbps | SWB | 0.2 | 0.965 | 0.85 |
| 32.0 kbps | SWB | 0.2 | 0.965 | 0.85 |
| 48.0 kbps | SWB | 0.2 | 1.0 | 1.0 |
| 64.0 kbps | SWB | 0.2 | 1.0 | 1.0 |
| 16.4 kbps | FB | 0.2 | 0.93 | 0.85 |
| 24.4 kbps | FB | 0.2 | 0.965 | 0.85 |
| 32.0 kbps | FB | 0.2 | 0.965 | 0.85 |
| 48.0 kbps | FB | 0.2 | 1.0 | 1.0 |
| 64.0 kbps | FB | 0.2 | 1.0 | 1.0 |
| 96.0 kbps | FB | 0 | 1.0 | 1.0 |
| 128.0 kbps | FB | 0 | 1.0 | 1.0 |

The last scale factor band energy is adapted as follows:

(1790)

where is the adaption factor for the last scale factor band energy according to table 167.

If refinement is active and , the remaining scale factor band energies are low-pass filtered in order to smooth the frequency envelope:

(1791)

Otherwise the scale factor band energies are not affected by further modifications:

(1792)

6.2.2.3.8.3.8 Calculation of IGF gain factors

The IGF gain factors are used to finally shape the tiled subband values in order to adjust the spectral envelope of the synthesized signal above . First, the target energy level has to be calculated:

(1793)

(1794)

where is the hop-size of the refinement in dependency of and the maximal possible hop-size according to table 168:

(1795)

Table 169: Maximal IGF hop-size

|  |  |  |
| --- | --- | --- |
| Bitrate | mode |  |
| 9.6 kbps | WB | 4 |
| 9.6 kbps | SWB | 2 |
| 13.2 kbps | SWB | 4 |
| 16.4 kbps | SWB | 4 |
| 24.4 kbps | SWB | 4 |
| 32.0 kbps | SWB | 4 |
| 48.0 kbps | SWB | 4 |
| 64.0 kbps | SWB | 4 |
| 16.4 kbps | FB | 4 |
| 24.4 kbps | FB | 2 |
| 32.0 kbps | FB | 2 |
| 48.0 kbps | FB | 2 |
| 64.0 kbps | FB | 2 |
| 96.0 kbps | FB | 1 |
| 128.0 kbps | FB | 1 |

Second, a normalization term for normalizing the target energy is calculated as follows:

(1796)

(1797)

Finally, the IGF gain factors have to be calculated according to the following formula:

(1798)

where is the general adaption factor for all scale factor band energy according to table 170.

If hop-size , the IGF gain factors in between a particular hop are hold beginning at the hop-start:

(1799)

void (1800)

6.2.2.3.8.3.9 IGF envelope adjustment

With the calculated IGF gain factors as described in subclause 6.2.2.3.8.3.8, the envelope of the spectrum above is adjusted as follows:

(1799)

where shall be already mapped with the function see subclause 5.3.3.2.11.1.1, and being the number of bands. is the gap-filled signal in accordance with subclause 6.2.2.3.8.3.5.

##### 6.2.2.3.9 Inverse window grouping (TCX5 separation)

If the configuration, determined as described in subclause 6.2.4.2, indicates that some sub-frames are coded using TCX5 then a sub-frame containing MDCT bins of 2 TCX5 sub-frames is de-interleaved to form 2 consecutive TCX5 sub-frames before the optional Temporal Noise Shaping, the optional IGF temporal flattening and before the transformation with the inverse MDCT:

(1800)

##### 6.2.2.3.10 Temporal Noise Shaping

The decoding process for Temporal Noise Shaping is carried out separately on each window of the current frame by applying the so called lattice filter to selected regions of the spectral coefficients (see in subclause 5.3.3.2.2). The number of noise shaping filters applied to each window is specified by "nMaxFilters ".

For TCX5 the same rearrangement is done as described in subclause 5.3.3.2.2 prior to the TNS filtering and the rearrangement is reverted after the filtering.

First the transmitted filter coefficients have to be decoded, i.e. conversion to signed numbers, inverse quantization. Then the so called lattice filters are applied to the target frequency regions of the spectral coefficients (see subclause 5.3.3.2.2). The maximum possible filter order is defined by the constant TNS\_MAX\_FILTER\_ORDER.

The application of TNS shaping filter is done before the optional IGF temporal flattening and before the transformation with the inverse MDCT

The decoding process for one window can be described as follows pseudo code:

/\* TNS decoding for one window \*/

tns\_decode()

{

set\_zero( state, TNS\_MAX\_FILTER\_ORDER );

for (iFilter = nMaxFilters-1; iFilter >= 0; iFilter--) {

tns\_decode\_coef( order, index[iFilter], parCoeff[iFilter] );

tns\_filter( spectrum, startLine, endLine, parCoeff[iFilter], order, state );

}

}

/\* Decoder transmitted coefficients index[] for one TNS filter \*/

tns\_decode\_coef( order, index[], parCoeff[] )

{

/\* Conversion to signed integer \*/

for (i = 0; i < order; i++)

tmp[i] = index[i] + (1 << (TNS\_COEF\_RES-1));

/\* Inverse quantization \*/

iqfac = ((1 << (TNS\_COEF\_RES-1)) - 0.5) / (/2.0);

iqfac\_m = ((1 << (TNS\_COEF\_RES-1)) + 0.5) / (/2.0);

for (i = 0; i < order; i++) {

parCoeff[i] = sin( tmp[i] / ((tmp[i] >= 0) ? iqfac : iqfac\_m) );

}

}

/\* Lattice filter \*/

tns\_filter( spectrum[], startLine, endLine, parCoeff[], order, state )

{

for (j = startLine; j <= endLine; j++) {

spectrum[j] -= parCoeff[order-1] \* state[order-1];

for (i = order-2; i >= 0; i--) {

spectrum[j] -= parCoeff[i] \* state[i];

state[i+1] = parCoeff[i] \* spectrum[j] + state[i];

}

state[0] = spectrum[j];

}

}

Filter order 1 with the first coefficient equal to 0 identifies disabled filter.

Please note that this pseudo code uses a „C“-style interpretation of arrays and vectors, i.e. if parCoeff describes the coefficients for all filters, parCoeff[iFilter] is a pointer to the coefficients of one particular filter.

##### 6.2.2.3.11 IGF temporal flattening

The reconstructed signal by IGF is temporally flattened in the frequency domain when . The temporal flattening is performed in a frequency-selective manner as follows.

The selection of the spectral contents to be temporally flattened is done by comparing the quantized spectral coefficients with 0 and the contents whose coefficients are quantized to 0 are selected.

In order to maintain the significant spectral contents, they are temporarily replaced by the spectra which are similarly generated to the filled spectra by IGF:

(1803)

where is the quantized MDCT coefficient after arithmetic decoding and is the reconstructed MDCT coefficient by IGF.

The linear prediction of the spectra is done and the linear prediction coefficients are calculated. Then the temporally flattened spectrum is given by the following filtering:

. (1801)

Finally, the significant spectral contents are restored by:

, (1805)

and then the frequency-selectively temporally flattened spectrum is output to IMDCT for getting the time domain signal.

### 6.2.3 High Quality MDCT decoder (HQ)

#### 6.2.3.1 Low-rate HQ decoder

##### 6.2.3.1.1 Mode decoding

Based on the encoded bandwidth and operated bit-rate, mode information is decoded from 1 or 2 bits. Based on the decoded mode information, decoding configurations like band structures are set. The band structure definition for NB, WB, SWB, and FB is the same as encoder presented in table 103 to 108.

##### 6.2.3.1.2 Energy Envelope decoding

From the received low-rate HQ envelope coding method bit and coding mode bit, the low-rate HQ energy decoding mode is determined and the coded quantization differential indices are decoded by the Large symbol decoding method or the Small symbol decoding method. From the received coding mode bits for energy envelope decoding, the envelope coding mode flag is determined, based on the coding mode the transmitted differential indices are decoded. For example, if flag has value 1 the Small symbol decoding method is used otherwise the Large symbol decoding method is used for decoding the differential indices.

The final resulting reconstructed quantized energies are obtained equally as in the encoder, described in subclause 5.3.4.1.3.

6.2.3.1.2.1 Small symbol decoding method

If the flag has value 1, the flag *LCmode* information is extracted from the bit stream. If the *LCmode* has value 1 resized Huffman decoding mode is used otherwise context based Huffman decoding mode is used for decoding the differential indices.

*If IsTransient* is *True*,

The decoded differential indices is extracted either from context based or resized Huffman coding mode according to flag *LCmode* and the differential indices for band *b*=0, are up packed directly with 5 bits. The decoded differential indices are adjusted to extract the original values according to

(1802)

*If IsTransient* is *False*,

The decoded differential indices is extracted either from context based or resized Huffman coding mode according to flag *LCmode* and the differential indices for band *b*=0, are up packed directly with 5 bits. Once the differential indices are extracted, least significant code are up packed directly with 1 bit and the differential indices are reconstructed according to

(1803)

The decoded differential indices are adjusted to extract the original values according to

(1804)

6.2.3.1.2.1.1 Context based Huffman decoding mode

If the context based Huffman decoding mode has been determined, the decoding is performed by referring table 171 and table 172 based on the context described in subclause 5.3.4.1.3.3.1. Four LSBs of entries in table 171 and table 172 indicates how many bits shall be read from bit-stream buffer to decode the next symbol and the signs indicate if the Huffman decoding is terminated or not. The procedure of how to perform Huffman decoding is shown below:

i=0

while( hufftab[i] > 0)

{

read\_bits += hufftab[i] & 0xF

i = (hufftab[i]>>4)+read\_bits(hufftab[i] & 0xF)

}

return hufftab[i]

Table 171: Huffman decoding table for the context based Huffman decoding (*group0*,*group2*)

| Index | Code | Index | Code | Index | Code | Index | Code | Index | Code |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0X13 | 11 | -0X0D | 22 | -0X18 | 33 | 0X41 | 44 | -0X05 |
| 1 | -0X10 | 12 | 0X51 | 23 | -0X16 | 34 | -0X1C | 45 | -0X1E |
| 2 | -0X0F | 13 | 0X62 | 24 | 0X71 | 35 | -0X08 | 46 | -0X04 |
| 3 | -0X11 | 14 | -0X14 | 25 | -0X0B | 36 | 0X31 | 47 | -0X1F |
| 4 | 0X51 | 15 | 0X81 | 26 | 0X71 | 37 | 0X41 | 48 | 0X11 |
| 5 | 0X61 | 16 | -0X0C | 27 | -0X1A | 38 | -0X1D | 49 | -0X03 |
| 6 | -0X0E | 17 | 0X81 | 28 | 0X71 | 39 | -0X06 | 50 | 0X11 |
| 7 | -0X12 | 18 | -0X15 | 29 | -0X09 | 40 | 0X31 | 51 | -0X02 |
| 8 | 0X51 | 19 | -0X17 | 30 | -0X1B | 41 | 0X41 | 52 | 0X11 |
| 9 | 0X61 | 20 | 0X71 | 31 | -0X0A | 42 | -0X07 | 53 | -0X01 |
| 10 | -0X13 | 21 | 0X81 | 32 | -0X19 | 43 | 0X41 | 54 | 0X00 |

Table 172: Huffman decoding table for the context based Huffman decoding (*group1*)

| Index | Code | Index | Code | Index | Code | Index | Code | Index | Code |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0X12 | 12 | -0X12 | 24 | 0X51 | 36 | -0X18 | 48 | -0X1B |
| 1 | 0X41 | 13 | 0X42 | 25 | 0X61 | 37 | -0X05 | 49 | -0X1A |
| 2 | -0X0F | 14 | -0x0C | 26 | -0X16 | 38 | -0X04 | 50 | 0X11 |
| 3 | 0X41 | 15 | 0X61 | 27 | -0X09 | 39 | -0X03 | 51 | 0X00 |
| 4 | -0X10 | 16 | -0X13 | 28 | 0X51 | 40 | 0X51 | 52 | 0X11 |
| 5 | -0X0E | 17 | 0X61 | 29 | 0X61 | 41 | -0X06 | 53 | -0X1D |
| 6 | 0X31 | 18 | 0X71 | 30 | -0X17 | 42 | 0X51 | 54 | 0X11 |
| 7 | -0X11 | 19 | -0X0A | 31 | 0X62 | 43 | -0X19 | 55 | -0X1E |
| 8 | 0X31 | 20 | 0X71 | 32 | -0X08 | 44 | 0X51 | 56 | -0X1F |
| 9 | 0X41 | 21 | -0X0B | 33 | 0X81 | 45 | -0X01 | 57 | -0X1B |
| 10 | -0X0D | 22 | -0X14 | 34 | -0X07 | 46 | -0X1C | - | - |
| 11 | 0X41 | 23 | -0X15 | 35 | 0X81 | 47 | -0X02 | - | - |

6.2.3.1.2.1.2 Resized Huffman decoding mode

If *IsTransient* is *True*

If the frame is Transient, the Huffman decoding is then performed on the transmitted differential indices. The Huffman codes for the differential indices are given in table 111 in subclause 5.3.4.1.3.3.

For Non-Transient frames, the Huffman decoding is then performed on the transmitted differential indices. The Huffman codes for decoding the indices are given in table 115 in subclause 5.3.4.1.3.3.3. The differential indices decoded using table 115 takes the form, the decoded differential indices are reconstructed which is exactly reverse to the encoder described in subclause 5.3.4.1.3.3.3 equation (1040). The way to reconstruct the differential index, which corresponds to the modification in encoder, can be done as shown in the following equation.

 (1809)

6.2.3.1.2.2 Large symbol decoding method

If the Large symbol coding method is determined, the encoded envelope data should be decoded using the reverse process of encoding either the pulse mode or the scale mode as described in subclause 5.3.4.1.3.4

The Huffman data in the encoded envelope data is decoded by a Huffman decoding method described in subclause 6.2.3.1.2.1.1 using table 173.

Table 174: Huffman decoding table for the Large symbol decoding method

| Index | Code | Index | Code | Index | Code |
| --- | --- | --- | --- | --- | --- |
| 0 | 0X11 | 5 | 0X21 | 10 | -0X01 |
| 1 | 0X21 | 6 | -0X02 | 11 | -0X06 |
| 2 | -0X04 | 7 | -0X05 | 12 | 0X11 |
| 3 | 0X21 | 8 | 0X11 | 13 | -0X07 |
| 4 | -0X03 | 9 | 0X21 | 14 | -0X00 |

##### 6.2.3.1.3 Spectral coefficients decoding

6.2.3.1.3.1 Normal Mode

Figure 94 shows the overview of the normal mode decoder.



Figure 94: Block diagram of the Normal mode decoder overview

6.2.3.1.3.1.1 Energy envelope decoding

Details are described in subclause 6.2.3.1.2.

6.2.3.1.3.1.2 Tonality flag decoding

Tonality flags described in subclause 5.3.4.1.4.1.3 are decoded and used for calculation of the bit allocations.

6.2.3.1.3.1.3 Bit allocation

The processing is in the same manner with the one at the encoder side.

Firstly, the fine gain adjustment bits are derived.

Secondly, the bands of the limited-band mode are identified based on the decoded limited-band mode flags, and their corresponding bandwidths are set if the limited-band mode is used in encoding. As is described in 5.3.4.1.4.1.4.4.2, the band is limited to the vicinity of the maximum amplitude spectrum frequency of the previous frame. The position of the maximum amplitude spectrum frequency of the previous frame is stored in a memory, which was searched using the decoded MDCT spectrum in the previous frame. The information of the limited band (i.e. identified vicinity of the maximum amplitude spectrum frequency position) is output to the TCQ decoder along with the bit budget allocated through the following bit allocation process.

Thirdly, bands encoded using PFSC are identified based on the decoded tonality flags among the four highest bands and necessary bits (1 or 2 bits) are allocated to each of the identified bands.

Finally, remaining bits are allocated to other bands based on perceptual importance using the decoded quantized band energies. When there is any band whose assigned bit results in zero in the four bands, such band is re-identified as a PFSC encoding band and the bit allocations are re-calculated.

6.2.3.1.3.1.4 Fine structure decoding

6.2.3.1.3.1.4.1 TCQ decoding

6.2.3.1.3.1.4.1.1 Joint USQ and TCQ

In order to de-quantize the fine structure of the normalized spectrum, the ISC and the information for the selected ISCs in each band are decoded by the position, number, sign and magnitude of the ISCs.

The magnitude information is decoded by the joint USQ and TCQ with an arithmetic decoding, while the position, number and sign are decoded by an arithmetic decoding.

The decoding method is selected at the Selecting Decoding 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 the band are decoded to zero by the zero decoding block. Otherwise, each band is decoded by the selected de-quantizer.

The quantizer selection information selects between the TCQ and USQ quantizes to get the same results as that of encoder.

The Estimating Number of Pulses block determines the number of pulses per a band using the band length and the bit allocation data R[]. Its principle of operation is same as that of the method which is used in the Scaling Bands module in encoder, see subclause 5.3.4.1.4.1.5.1.1.

The Lossless Decoding and the Decoding Position Info block reconstruct the position information of the ISCs, i.e. the number of ISCs and their positions. This process is similar to encoder side and same probabilities should be used for the proper decoding, see subclause 5.3.4.1.4.1.5.1.

In the Joint USQ and TCQ Decoding block, the magnitudes of the gathered ISCs are decoded by the arithmetic decoding and de-quantized by the joint USQ and TCQ decoding. In this block the non-zero position and the number of the ISC is utilized for the arithmetic decoding. The joint USQ and TCQ have two types of decoding 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 6.2.3.1.3.1.4.1.2 TCQ and USQ with second bit allocation and 6.2.3.1.3.1.4.1.3 LSB TCQ for USQ.

In the Decoding Signs block, the sign information of the selected ISC is decoded by the arithmetic decoding with equal probabilities for the positive and negative signs.

In order to recover the quantized components for each band, the position, sign and magnitude information is added to the quantized components to recover the real components at the Recovering Quantized Components block.

At this point the determined bands with no transmitted data are filled by zeroes. Then the number of pulses in the non-zero bands is estimated and the position information, including the number and position of ISCs, is decoded using this estimated number. After the magnitude information is decoded using the lossless decoder, the joint USQ and TCQ decoding is performed. For non-zero magnitude values the signs and quantized components are finally reconstructed.

In the Inverse Scaling Bands block, the inverse scaling of the quantized components is performed by using the transmitted norm information. The inverse scaled signal is the output of the TCQ decoding.



Figure 95: Block diagram of fine structure decoding using TCQ

6.2.3.1.3.1.4.1.2 TCQ and USQ with second bit allocation

The general de-quantization and decoding scheme of the TCQ and USQ with second bit allocation consists of several main blocks: quantizer decision, TCQ decoder, USQ decoder, lossless decoder, 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 Decoding Method block. Then the selected decoder restores the current band in association with the lossless decoder, based on the arithmetic decoding with the transmitted bit stream.



Figure 95: Block diagram of TCQ and USQ decoding with second bit allocation

The decoding process is started by the recovering the non-zero bands and positions using the transmitted bit-stream and the bit allocation R[] for the selected quantizer. By using this information, the appropriate magnitude for the decoded band is selected. The difference between bit allocation R[] and actual decoded bits per band is accumulated and called the surplus. This surplus will be used while decoding two band determined by second bit allocation procedure described in encoder in subclause 5.3.4.1.4.1.5.1.2.

The magnitude decoding based on binary arithmetic decoding is as follows. First the probability of symbol is calculated by the equations in encoder subclause 5.3.4.1.4.1.5.1.2. Then the number of pulses for each magnitude is decoded by using the probabilities and , where corresponds to last pulse in magnitude and to all other pulses. The magnitude of the pulse probabilities are then modified after this calculation with respect to the trellis code limitation, i.e. magnitudes that are impossible are assigned zero probability.

This algorithm was modified to save complexity for bands with a large number of pulses. The procedure is same as that of encoder subclause 5.3.4.1.4.1.5.1.2.

Location decoding is done based on the same algorithm as that of magnitudes decoding and uses the same complexity reduction technique.

Signs are decoded with the arithmetic decoder, using equal probabilities of positive and negative signs.

6.2.3.1.3.1.4.1.3 LSB TCQ for USQ

The idea of the LSB TCQ for USQ is to use advantages of both quantizers (USQ and TCQ) in one scheme and exclude the path limitation from the TCQ.



Figure 96: Block diagram for LSB TCQ decoding

The decoding process starts from receiving the bit allocation R[] and the decoding of the band information including:

* Number of nonzero positions for ISCs
* Nonzero positions
* USQ magnitude
* Signs for nonzero magnitudes

First the number of nonzero pulses and their positions are decoded using the arithmetic decoder. Then the USQ magnitudes are decoded band by band using bit allocation with surplus control. This generates Delta values in the same manner as the encoder, see subclause 5.3.4.1.4.1.5.1.3. The difference between the bit allocation R[] and actual decoded bits per band is accumulated and called the surplus, which is then used in the next bands.

The algorithms used for decoding positions and magnitudes are the same as those described in subclause 6.2.3.1.3.1.4.1.2 in TCQ and USQ decoder.

After receiving the USQ magnitudes, the TCQ path is decoded from the bit-stream using the arithmetic decoder.

The decoded path is used to reconstruct the residual array according to the decoded trellis state. From each path bit, two LSB bits are generated in the residual array. This process shown in pseudo code:

for( state = 0, i = 0; i < bcount; i++)

{

residualbuffer[2\*i] = dec\_LSB[state][dpath[i]] & 0x1;

residualbuffer [2\*i + 1] = dec\_LSB[state][dpath[i]] & 0x2;

state = trellis\_nextstate[state][dpath[i]];

}

Starting from *state* 0, the decoder moves through the trellis using decoded *dpath* bits, and extracts two bits corresponding to the current trellis edge.

In the Spectrum recovering block the decoded residual array is added to the non-zero spectral components. The output of this block is the reconstructed spectrum.

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

Finally, as described in subclause 5.3.4.1.4.1.4.4.1, fine gain adjustment is performed on the dominant bands. Decoded fine gain adjustment factor is applied to the de-normalized decoded MDCT coefficients.

6.2.3.1.3.1.4.2 Noise-filling

Noise-filling is performed between “De-norm. and Fine gain adj.” and “PFSC decoder” blocks in Figure 94 and the process is the same as the one at the encoder side.

6.2.3.1.3.1.4.3 PFSC decoding

6.2.3.1.3.1.4.3.1 Envelope normalization

This process is the same with the one described in subclause 5.3.4.1.4.1.5.3.2.

6.2.3.1.3.1.4.3.2 Lag information decoding

Lag indices for the last four sub-bands (i.e. *b*=18 to 21 in 13.2 kbps and *b*=20 to 23 in 16.4 kbps) are decoded if the corresponding decoded tonality flag is set to “0”. The starting position is decoded as as described in subclause 5.3.4.1.4.1.5.3.3. Based on the starting position and width of search band, the predicted high-frequency spectrum is generated from the envelope normalized TCQ-decoded low-frequency spectrum.

6.2.3.1.3.1.4.3.3 Scaling and noise smoothing

Scaling factors are calculated for the predicted bands using the decoded band energies. Each scaling factor is calculated as the square root of the quotient of the quantized band energy divided by its corresponding band energy from the predicted high-frequency spectrum. The calculated scaling factors are attenuated by the scaling factor of 0.9 and applied to the predicted high-frequency spectrum.

Inter-frame smoothing process for the noise components are applied as described in subclause 5.3.4.1.4.1.5.3.3.3.

The Normal mode PFSC decoding overview is shown in figure 97.

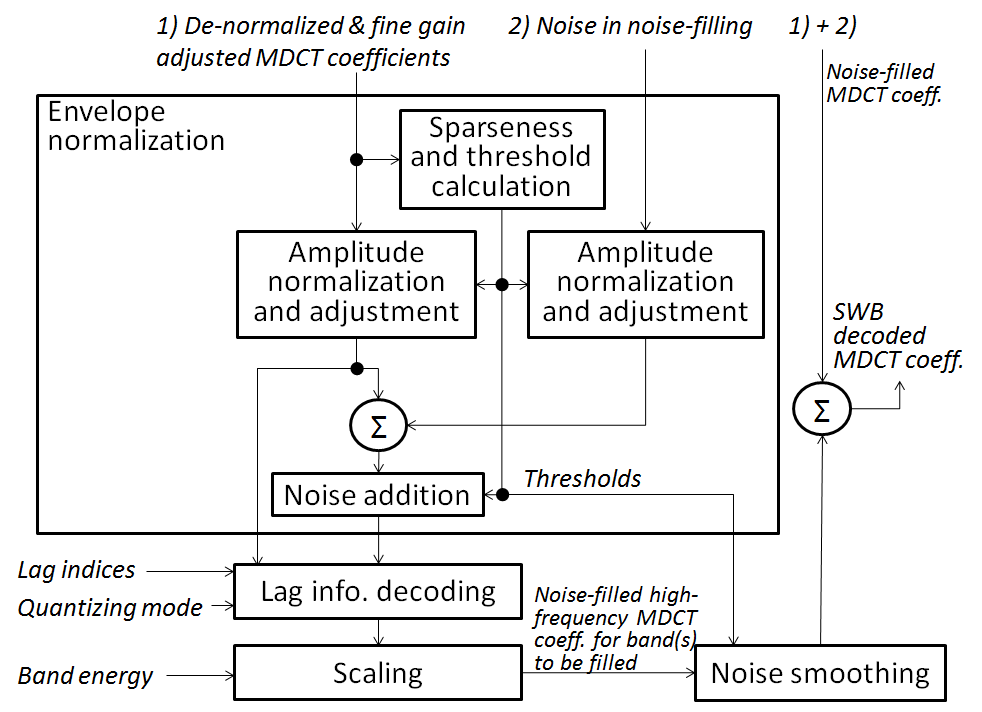


Figure 98: Block diagram of the Normal mode PFSC decoder

6.2.3.1.3.2 Transient Mode

6.2.3.1.3.2.1 Energy envelope decoding

Details are described in subclause 6.2.3.1.2.

6.2.3.1.3.2.2 Bit allocation

The processing is the same as subclause 5.3.4.1.4.2.2

6.2.3.1.3.2.3 Fine structure decoding

TCQ decoding with Transient mode configurations is performed.

6.2.3.1.3.3 Harmonic Mode

6.2.3.1.3.3.1 Overview

The high-level decoder structure of the Harmonic mode is basically the same with the Normal mode. The main difference can be found in its detailed structure of the PFSC block, and it is shown in the following figure.

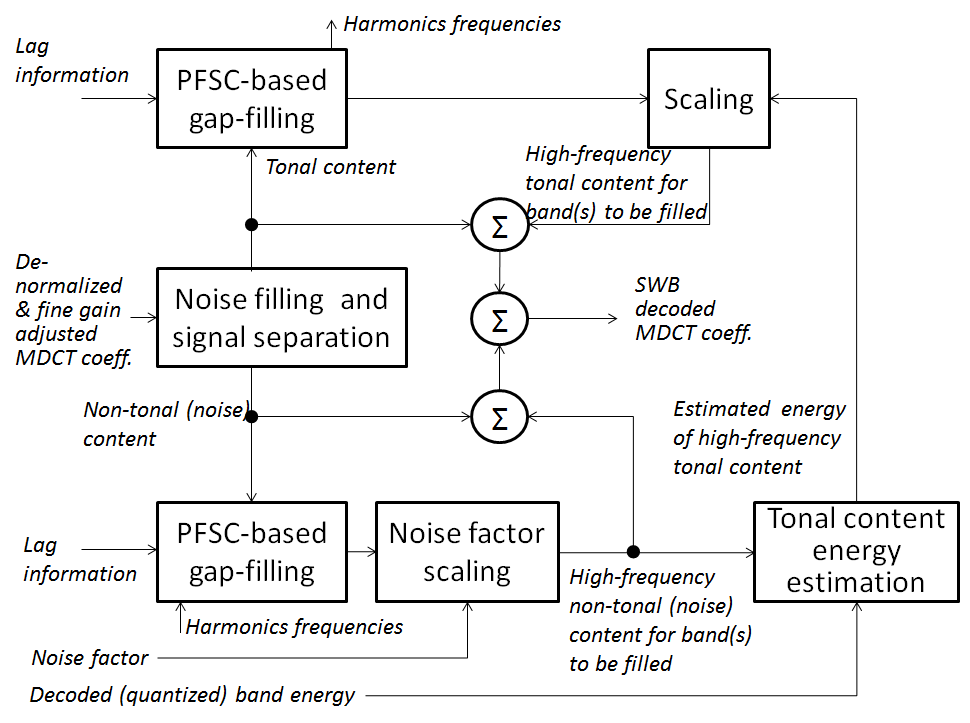


Figure 98: Block diagram of the Harmonic mode decoder overview

6.2.3.1.3.3.2 Energy envelope decoding

Details are described in subclause 6.2.3.1.2.

6.2.3.1.3.3.3 Bit allocation

The processing is in the same manner with the one at the encoder side.

At first, the fine gain adjustment bits are derived, procedure is same as in explained in sub-clause 5.3.4.1.4.3.2.1, and then remaining bits are allocated in an adaptive manner where more bits are allocated to the bands in a perceptually significant group than those in a less significant group. Detailed procedure is same as in explained in sub-clause 5.3.4.1.4.3.2.2.

6.2.3.1.3.3.4 Fine structure decoding

6.2.3.1.3.3.4.1 TCQ decoding

This part is the same with the Normal mode as described in subclause 6.2.3.1.3.1.4.1.

6.2.3.1.3.3.4.2 Noise filling for quantized spectrum

In this subclause noise is filled in the quantized spectrum where coefficients have been quantized to zero when the bit allocation subclause allocates non zero bits to the bands and also fills the un quantized bands up to the transition frequency in the same manner as in the encoder, see subclause 5.3.4.1.4.3.3.2

6.2.3.1.3.3.4.3 PFSC-based gap filling

6.2.3.1.3.3.4.3.1 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 subclause 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 are filled with the new generated spectrum. The predicted spectrum is generated using normalized noise filled quantized spectrum described in subclause 6.2.3.1.3.3.4.3.2.

Based on the bit allocation described in subclause 6.2.3.1.3.3.3, if any of is allocated with zero bits, the corresponding band with start and end positions according to table 108 in thehas a gap and it is filled with the predicted spectrum described in subclause 6.2.3.1.3.3.4.3.5 corresponding to .in

6.2.3.1.3.3.4.3.2 Envelope Normalization

The envelope normalization is performed equally as in the encoder, described in subclause 5.3.4.1.4.3.3.3.2. As a result the envelope normalized signal is obtained, whereis the envelope normalized low frequency quantized spectrum and is the envelope normalized low frequency noise spectrum.

6.2.3.1.3.3.4.3.3 Decoding of lag index

Lag index for sub-bands i=0,1 is decoded from the bit stream. For sub-bands 0 and 1, encoded best match position  is decoded using the starting position and the lag index , is defined in equation (1147).

Based on the best match position the predicted spectrum is generated from the envelope normalized noise filled quantized spectrum. The detailed description of the predicted spectrum generation is described in following subclause 6.2.3.1.3.3.4.3.5.

6.2.3.1.3.3.4.3.4 Structure analysis for Harmonics

The structure analysis for Harmonic mode is performed equally as in the encoder, described in subclause 5.3.4.1.4.3.3.3.4. As a result estimated harmonic is obtained; the estimated harmonic is used for generating the predicted spectrum for the HF region

6.2.3.1.3.3.4.3.5 Predicted spectrum generation

Predicted spectrum is generated for the high frequency region by using the envelope normalized noise-filled quantized spectrum, which is obtained from subclause 6.2.3.1.3.3.4.3.2. Predicted spectrum is generated, first by extracting the desired noise components from the described in subclause 6.2.3.1.3.3.4.3.6 followed by tonal generation using described in subclause 6.2.3.1.3.3.4.3.7.

Noise filled spectrum is used for estimating the tonal energy and the tonal components of the spectrum in the high frequency region, which is obtained from subclause 6.2.3.1.3.3.4.3.7 are normalized using the estimated tonal energy, where is calculated as follows:

(1805)

: is the noise energy obtained using the noise filled spectrum according to

(1806)

The noise energy obtained from equation (1807) is adjusted, when the noise filled spectrum has low level noise and / or when the noise filled spectrum has high level noise. Low noise level is detected using the energy ratio between noise and the total band energy and high noise level is detected when the estimated tonal energy is negative. The adjustment factor is estimated according to

(1808)

For each band, based on the obtained from equation (1809) is used to re-calculate the tonal energy and estimated noise using equations (1810) and (1811). The tonal components of the spectrum in the high frequency region are normalized using the scale factor calculated as follows

(1812)

The calculated scale factor and extracted tonal components are used for injecting the tonal components into the noise filled spectrum according to

(1813)

where, is the tonal positions obtained from subclause 5.3.4.1.4.3.3.3.5

6.2.3.1.3.3.4.3.6 Noise filling for the predicted spectrum

Noise filling for the predicted spectrum is performed equally as in the encoder, described in subclause 5.3.4.1.4.3.3.3.5. As a result noise filled spectrum and tonal positionsis obtained, where *j* is the pulse resolution. The obtained predicted spectrum which contains noise is adjusted using the noise factor according to

(1814)

where is the noise factor which is decoded from the bit stream and the decoded noise factor is converted to linear domain as follows

(1815)

6.2.3.1.3.3.4.3.7 Tonal generation for predicted spectrum

First, the tonal components are extracted from the desired portion of envelope normalized quantized spectrum based on the decoded best match position . The extracted tonal components are used for the spectrum in the high frequency region. As the normalized quantized spectrum characteristics are flat all the values during the normalization process will have similar values, all the non-zero coefficients in the desired region of is identified as follows



where, are defined as follows

is the tonal resolution obtained from the normalized quantized spectrum for sub band *i*=0,1

is the tonal components extracted from the normalized quantized spectrum and used as the spectrum in the high frequency for sub band *i*=0,1

The tonal information, for *i*=0, 1 obtained from normalized quantized spectrum is used for sub band *i*=2, 3. Using the estimated harmonic frequency obtained from subclause 6.2.3.1.3.3.4.3.4 frequency positions of the extracted tonal components are adjusted as described in subclause 6.2.3.1.3.3.4.3.5.

Based on the band definition described in table 108, the high frequency band ranges are defined . Using the band definitions for high frequency region, the extracted tonal components and its corresponding pulse resolutions are restructured, and used for generating predicted spectrum. For example, the restructured information for sub band *i*=0 is equivalent to .

#### 6.2.3.2 High-rate HQ decoder

A high level structural block diagram of the high-rate HQ decoder is in figure 99.



Figure 100: High level structure of the high-rate HQ decoder

Firstly, the High-rate HQ coding mode information is decoded.

##### 6.2.3.2.1 Normal Mode

6.2.3.2.1.1 Envelope decoding

From the received high-rate HQ norm coding mode bits, the high-rate HQ norm coding mode is determined and the transmitted differential indices are decoded using the selected method. The quantization index of the lowest-frequency band, i.e., , is directly decoded in all modes.

6.2.3.2.1.1.1 Context based Huffman decoding mode

If this coding mode is determined for the current frame, the context based Huffman decoding is then performed on the transmitted quantization differential indices using the method described in subclause 6.2.3.1.2.1.1 and the tables shown in 175 and 176.

6.2.3.2.1.1.2 Re-sized Huffman decoding mode

If this coding mode is determined for the current frame, the resized Huffman decoding is then performed on the transmitted quantization differential indices using the method described in subclause 6.2.3.1.2.1.2. The Huffman codes for the differential indices are given in table 105.

6.2.3.2.1.1.3 Normal Huffman decoding and bit-packing mode

If this coding mode is determined for the current frame, the Normal Huffman decoding is then performed on the transmitted differential indices. The Huffman codes for the differential indices are given in subclause 5.3.4.2.1.2.3.

When the bit-packing mode is determined; the adjusted differential indices are un-packed directly with 5 bits.

The actual quantized norms are obtained by lookup table, defined in subclause 5.3.4.2.1.1.

6.2.3.2.1.2 Normal mode fine structure inverse quantization

6.2.3.2.1.2.1 Fine structure inverse PVQ-quantization

The spectral coefficient inverse quantization is done as is described in subclause 6.2.3.2.6

6.2.3.2.1.2.2 Fine gain prediction, inverse quantization and application

The bit allocation for the PVQ shape vector and fine gain adjustment , as well as and are obtained as in subclause 5.3.4.2.1.3a.1. The quantized gain prediction error is obtained by using the assigned bitrateand the fine gain adjustment is obtained by

(1816)

with for. The gain of the synthesis is adjusted by scaling the decoded fine structure with the fine gain.

6.2.3.2.1.3 Spectral filling

This subclause gives a technical overview of the spectrum filling processing which is applied at the decoder in HQ high rate mode.

6.2.3.2.1.3.1 Wideband adaptive noise filling at 24.4/32kbps

Wideband adaptive noise filling at 24.4 and 32 kbps proceeds by calculating the total available bits and the bits variance for the sub-bands in non-transient frames over the index range,

(1817)

(1818)

The average bit allocation threshold is initialized for each coefficient in each sub-band according to the values in table 177.

Table 178: Threshold for average bit allocation

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Band | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|  | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 | 1.5 |
| Band | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|  | 1.5 | 1.5 | 1.5 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.8 | 0.8 |

For the sub-bands in the index range, denotes the number of the sub-bands where the average number of allocated bits for each coefficient is not less than the threshold. The harmonic parameter for those sub-bands is calculated as follows:

(1819)

(1820)

The step length, , is calculated according to:

(1821)

For any sub-band in non-transient frames the following procedure is then followed. If the average number allocated bits for each coefficient in the sub-band is greater than or equal to the threshold 1.5, then the bit allocation for the sub-band is saturated and the un-decoded coefficients of the sub-band are not processed further by the noise filling. Otherwise, the bit allocation to the sub-band is un-saturated, and the un-decoded coefficients of the sub-band are reconstructed by noise filling. For any un-saturated sub-band with zero bits allocated to its coding, the envelope of the un-decoded coefficients in the sub-band are set to the decoded norm for that sub-band. Otherwise, if the un-saturated sub-band has bits allocated to it, the envelope of the un-decoded coefficients is calculated as follows:

The average energy of the sub-band is then calculated using the de-quantized norm as follows:

(1822)

The energy sum of the all decoded non-zero coefficients in this subband is then calculated

(1823)

A search of the maximum magnitude and the minimum magnitude of the decoded coefficients in each subband is also calculated for further processing.

The energy differenceis next calculated. If, the envelope of the un-decoded coefficients is set to zero . Otherwise, the envelope of the un-decoded coefficientsis calculated as follows:

The initial envelope of the un-decoded coefficients in the un-saturated sub-band is calculated by the energy difference,

(1824)

The average norm of the un-saturated sub-bandis calculated

(1825)

If or , then the spectrum of the sub-band is *sharp*. The envelope of the un-decoded coefficients is obtained by modifying the initial envelope as follows:

(1826)

If the spectrum of the sub-band is not *sharp* and , , then the harmonic parameter is added by the step length.

If the envelope is more than the half of the minimum magnitude, then the envelope is set to be equal to the half of the minimum magnitude.

(1827)

If the ratio of the average norms in the current frame to that of the previous frame lies in the range (0.5,…, 2), and the previous frame is a non-transient frame, the the envelope of the un-decoded coefficients for the current frame and the previous frame are weighted as follows.

(1828)

For the un-decoded coefficients in the sub-band, the coefficients are generated using a random noise generator and multiplied by the estimated envelope as described above.

For the last sub-band, a check is made whether the mode of the previous frame was not a transient, and whether the ratio of the decoded norms of the current frame to those of the previous frame are in the range (0.5, …,2.0), and the bit variance is not more than 0.3, and the sub-band of the current frame is bit allocated and the sub-band of the previous frame is not bit allocated or vice versa. If all of the above conditions are fulfilled, then the coefficients in the current frame and the previous frame for the last 20 coefficients in the last sub-band are weighted as follows:

(1829)

In the transient mode, un-decoded coefficients in a sub-band are generated from random noise, and de-normalized by the decoded norm for that sub-band.

6.2.3.2.1.3.2 General spectral filling

Based on the received bit-allocation, the transition frequencyis estimated in the same manner as in the encoder, see subclause 5.3.4.2.1.4. Spectral filling consists of two algorithms. The first algorithm fills the low‑frequency spectrum up to the transition frequency, the second algorithm regenerates the possibly non-coded high-frequency components by using the low-frequency noise-filled spectrum.

The interaction between these two algorithms is shown in figure 101. The resulting spectrum from both the noise-filling algorithm and the high frequency noise fill is a normalized spectrum which is shaped by the received quantized norms.



Figure 102: Spectrum filling block diagram

6.2.3.2.1.3.2.1 Noise filling

The first step of the noise fill procedure relies on the building of the so-called spectral codebook from the received (decoded) normalized transform coefficients. This step is achieved by concatenating the perceptually relevant coefficients of the decoded spectrum. Figure 103  illustrates this procedure. The decoded spectrum has several series of zero coefficients that are called spectral holes of a certain length. This length is the sum of the consecutive lengths of bands which were allocated zero bits.



Figure 104: Building the spectral codebook from the decoded transform signal

Since the length of all spectral holes can be higher than the length of the spectral codebook, the codebook elements might be re-used for filling several spectral holes.



Figure 105: Noise filling from the spectral codebook up to the transition frequency

Figure 106 shows how, based on the spectral codebook C, the non-quantized spectral coefficients are filled. Spectral holes are filled by increasing the codebook index j as much as the index i, used to cover all the spectral holes up to the transition frequency. Reading from the spectral codebook is done sequentially and as a circular buffer according to the following:

*i=*0; *j=*0  
(1:) if then ,  
 increment *i*,*j* (if out of bound, rewind *j* to start of codebook)  
 if *i=*0 then  
 STOP  
 else  
 goto (1:)  
 endif

For low bit rates, many of the quantized bands will contain few pulses and have a sparse structure. For signals which require a more dense and noise-like fill, a set of two anti-sparseness processed codebooks are created instead of the regular spectral codebook as illustrated in figure 107.



Figure 108: Creation of two parallel codebooks to handle sparse coded vectors.

The compression of the coded residual vectors is done according to the following definition:

(1830)

The virtual codebook which constitutes the spectral codebook is built only from “populated” sub-vectors, where each sub-vector has a length of 8. If a coded sub-vector does not fulfill the criterion:

(1831)

it is considered sparse, and is rejected. Since the sub-vector length is 8, this corresponds to a rejection criterion if less than 25% of the vector positions are populated. The remaining compressed sub-vectors are concatenated into Spectral codebook 1, with the length . The final step of the anti-sparseness processing is to combine the codebook samples pair-wise sample-by-sample with a frequency reversed version of the codebook. The combination can be described with the following relation:

(1832)

For SWB processing at 24.4 or 32 kbps in case of low spectral stability, spectral codebook 1 is used below band and the spectral codebook 2 is used above and including band . The spectral filling using these two codebooks is depicted in figure 109.



Figure 110: Creation of two parallel codebooks to handle sparse coded vectors.

6.2.3.2.1.3.2.2 High frequency noise fill

Based on the low-frequency filled spectrum, and prior to noise level attenuation, as described in the previous clause, the last step of the spectral filling consists of the generation of the target bandwidth audio signal. In other words, the process synthesizes a high-frequency spectrum from the filled spectrum by spectral folding based on the value of the transition frequency.

The target bandwidth generation is based on the spectral folding of the spectrum below the transition frequency to the high-frequency spectrum (zeroes above the transition frequency), see figure 111. A first spectral folding is achieved with respect to the point of symmetry defined by the transition frequency. No spectrum coefficients from frequencies below are folded into the high frequencies. In other words, only the upper half of the low frequencies are folded. If there are not enough coefficients in the upper half of the low frequencies to fill the whole spectrum above the transition frequency, the spectrum is folded again around the last filled coefficient. This process is repeated until the last band is filled.



Figure 112: The spectrum above the transition frequency is regenerated using spectral folding from the transition frequency

6.2.3.2.1.3.2.3 Noise level adjustment

After the fine structure of the spectral holes has been determined, the noise-filled part of the spectrum is attenuated according to the received *NoiseLevel* index. In the case of transient mode, the *NoiseLevel* is not estimated in the encoder and is automatically set to the value corresponding to zero index, i.e., 0 dB.

This operation is summarized by the following equation:

(1833)

For SWB processing at 24.4 or 32 kbps in case of low spectral stability, an additional adaptive noise-fill level adjustment is employed. First, an envelope adjustment vector is derived according to the following pseudo-code:

*For* ,  
 *if* ,  
 *if* ,  
 *if*   
 *if* ,  
   
 *else*  
   
 *else*  
   
 *else*  
 *if* *and* ,  
   
 *else*  
   
 *else*

where . Further, denotes the number of pulses for band as described in subclause 5.3.4.2.7, where corresponds to the case when zero bits are assigned to band . In short it permits strong attenuation for short bands where the neighboring bands are quantized, and gradually less when these requirements are not fulfilled. Once has been obtained, attenuation regions of consecutive bands where are identified. The attenuation for each of these regions are adjusted according to

(1834)

where is the number of consecutive bands in the attenuation region. The width-dependent attenuation function is a piece-wise linear function defined as

(1835)

The resulting vector is further combined with a limiting function which prevents attenuation during audio with high spectral stability. The spectral stability is calculated based on a low-pass filtered Euclidian distance between the spectral envelope values of adjacent frames:

(1836)

(1837)

Here denotes the value of the variable for frame . The spectral envelope stability parameter is derived by mapping to the range using a discreetly sampled sigmoid function implemented as a lookup table . Due to the symmetry of the function, the table is mirrored around the mid-point such that the final stability parameter can be obtained by

(1838)

where the quantization index is found by and clamping the index to the range . Finally, the gain adjustment vector is derived as

(1839)

where the envelope stabilityacts as a limiting function for the gain adjustment vector.

For WB processing, a slightly different gain adjustment vector is derived. Here, the is computed as

(1840)

where is a gain attenuation table for index , which in turn is derived by

(1841)

For SWB and 24.4 and 32 kbps, the gain adjustment is applied using a hangover logic which only permits attenuation in case a sequence of 150 frames without transients has been observed. In case this requirement is met for SWB encoded bandwidth or if the encoded bandwidth is WB, the gain adjustment vector is combined with the quantized envelope vector to form the gain adjusted envelope vector .

6.2.3.2.1.3.2.4 Spectral fill envelope shaping

When the full-bandwidth fine spectral structure is generated, the resulting spectrum is shaped by applying the gain adjusted envelope vectors for each band according to:

(1842)

##### 6.2.3.2.2 Transient Mode

6.2.3.2.2.1 Envelope decoding

The envelope is decoded as is described in subclause 6.2.3.2.1.1. In addition to those step the norms are also sorted as is done in the encoder, see subclause 5.3.4.2.2.1.

6.2.3.2.2.2 Fine structure inverse quantization (spectral coefficients decoding)

The spectral coefficients are decoded as for the Normal HQ mode as described in subclause 6.2.3.2.1.2.

6.2.3.2.2.3 Spectral filling

The spectral filling is done as described in 6.2.3.2.1.3, but the bandwidth extension in subclause 6.2.3.2.1.3.2.2 is not done.

##### 6.2.3.2.3 Harmonic Mode

6.2.3.2.3.1 Core decoding

Envelope decoding and the PVQ decoder are described in subclause 6.2.3.2.1.1 and subclause 6.2.3.2.1.2, respectively.

If a sub-band has bits allocated to it, then the decoded coefficients of the sub-band are de-normalized by multiplying the de-quantized norm of the sub-band, and in this way the de-normalized coefficients are obtained. Otherwise, if a sub-band has no bits allocated to it, the de-normalized coefficients of that sub-band are set to 0. And the higher frequency band coefficients with the index of sub-band above are 0 and are reconstructed by bandwidth extension, where is the index of the highest frequency sub-band of the decoded low frequency band signal.

6.2.3.2.3.2 Bandwidth extension decoding for harmonic mode

The start index for the bandwidth extension is adaptively obtained according to the value of.

Firstly preset the start index for bandwidth extension :

(1843)

Then, in order to predict the excitation signal of bandwidth extension, judge whether the index of the highest frequency sub-band of the decoded low frequency band signal is less than the start index for bandwidth extension , i.e. judge whether the highest frequency bin of bit allocation is less than the preset start frequency bin for bandwidth extension,

* if , is then set to . The excitation signal of bandwidth extension is predicted by the preset start index and the chosen excitation signal from the decoded low frequency band signal with the given bandwidth length.
* Otherwise, the excitation signal of bandwidth extension is predicted by the preset start index , the index of the decoded highest frequency sub-band and the chosen excitation signal from the decoded low frequency signal with the given bandwidth length.

Finally, the higher frequency band signal is reconstructed by the predicted excitation signal and the envelopes as described in subclause 6.2.3.2.2.1.

6.2.3.2.3.2.1 Calculate excitation adaptive normalization lengths

The de-normalized coefficients calculated in subclause 6.2.3.2.3.1 need to be recovered to remove the original core envelope effects to give the excitation for bandwidth extension. The normalization length is adaptively obtained according to the signal characteristics. The normalization length of the previous frame is initialized to 8.

208 MDCT coefficients in the 0-5200 Hz frequency range are split into 13 normalization sub-bands with 16 coefficients per sub-band. The peak magnitude and average magnitude in each normalization sub-band are then calculated. The counter is initialized to zero and increased by one if and , where

(1844)

The normalization length is set to , and it is adjusted with reference to the value from the previous frame,

(1845)

6.2.3.2.3.2.2 Calculate envelopes for excitation normalization

The normalization envelopes, for each spectral bin are calculated as follow:

(1846)

The value are then normalized using the normalization envelopes to obtain the normalized coefficients ,

(1847)

6.2.3.2.3.2.3 Adaptive excitation generation

The normalized coefficients in the frequency range 1500-5025Hz, i.e. the coefficients, are selected for the excitation calculation. The starting frequency bin of the excitation, , is calculated as follows,

(1848)

The selected low frequency normalized coefficients from which the re-constructed higher band coefficients are obtained are copied to the high band starting at frequency, , as follows

(1849)

The low frequency normalized coefficients may in practice be copied N times as a circular buffer in order to fill in the re-constructed higher bands, where N can be a decimal fraction.

6.2.3.2.3.2.4 Weighting the re-constructed higher band coefficients and random noise

The envelopes of the re-constructed higher band coefficients are calculated according to the band structure given in table 129, and then the re-constructed higher band signal is weighted and random noise added.

(1850)

Where and .

The weighting factor for the normalized re-constructed higher band signal, , is

(1851)

The weighting value of the normalized re-constructed higher band signal, , is

(1852)

where the noise level is estimated as follows:

(1853)

and the sum of the differences between the consecutive norms and the sum of the norms in the index range, are given by

(1854)

(1855)

##### 6.2.3.2.4 HVQ

First the HVQ decoder extracts from the bitstream number of coded peaks, and reconstructs spectral peaks positions and peak gains. The peaks positions are decoded with either Huffman decode or space coding decoder, based on the received mode decision. The peak shapes vectors are reconstructed from the received VQ indices and further scaled with reconstructed peak gainsfor the corresponding shape region. The low-frequency bands are PVQ decoded, with number of bands determined as described in 5.3.4.2.5

The unquantized coefficients below 5.6 kHz for 24.4 kb/s and 8 kHz for 32 kb/s are grouped into 2 sections and noise filled and scaled. Each of the sections covers half of coded band (of 112 bins at 24.4 kbps and 160 bins at 32 kbps). After the noise fill each of the sections is scaled with the corresponding reconstructed gains and . The gains reconstructed in the current frame are smoothed with the levels from the past frame

(1856)

The reconstructed envelope levels used above 5.6 kHz for 24.4 kb/s and 8 kHz for 32 kb/s are adjusted based on the presence or absence of peak in the low-frequency fine structure used in the noise-fill.

(1857)

##### 6.2.3.2.5 Generic Mode



Figure 107: Generic mode Decoder Block Diagram

6.2.3.2.5.1 Low frequency envelope decoding

This is described in subclause 6.2.3.2.1.1.

6.2.3.2.5.2 High frequency envelope de-quantization

The envelope VQ indices for SWB or for FB are used to de-quantize the high frequency envelope.

At 24.4kbps, the de-quantized high frequency envelope can be determined by:

(1859)

*While at 32kbps, the de-quantized high frequency envelope can be determined by:* (1860)

The final de-quantized envelope is then calculated:

(1858)

In FB case, is further used to generate the de-quantized high frequency envelope.

6.2.3.2.5.3 High frequency envelope refinement

The high frequency envelope refinement is described in subclause 5.3.4.2.6.5. After de-quantizing the high frequency envelope using the VQ described in subclause 6.2.3.2.5.2, the de-quantized high frequency envelope is mapped to one of the HQ high rate normal mode bands. To generate the norms the mapped high frequency envelope is quantized and de-quantized with the scalar quantizer, as shown in subclause 5.3.4.2.6. The de-quantized low frequency envelope and the de-quantized high frequency norms are then combined. Using the combined norms, the bit allocation information per each band is calculated using the fractional bit allocation method. If there are any high bands which have allocated bits, the refinement data is decoded and used to update the high frequency norms. The updated norms are used for de-normalizing the de-quantized spectrum by the PVQ algorithm in subclause 6.2.3.2.5.4 and the noise filling algorithm in subclause 6.2.3.2.5.5.Then the initial bit allocation information is updated, based on the number of bits used for representing the refinement data.

6.2.3.2.5.4 PVQ

This is described in subclause 6.2.3.2.1.2.2

6.2.3.2.5.5 Noise filling

Noise filling is performed described in subclause 6.2.3.2.1.3.2.1. The last band for this noise filling in Generic mode is defined as, where is the last band index where the spectrum was quantized using PVQ. The filled spectrum is further de-normalized to generate as described in subclause 6.2.3.2.1.3.2.4. If *core\_sfm* is higher than *Nband\_LF*-1, then the are the high frequency norms described in subclause 6.2.3.2.5.3.

6.2.3.2.5.6 High frequency excitation spectrum

The high frequency excitation spectrum is based on a copy of the decoded low frequency spectrum. First spectral anti-sparseness processing is applied, and then dynamic range control is applied to depending on the decoded excitation class. Finally, a simple spectral copy is done to create the high frequency excitation spectrum.

6.2.3.2.5.6.1 Spectral anti-sparseness processing

The spectral anti-sparseness processing is performed on the low frequency spectrum by inserting a 0.5 amplitude coefficient, with a random sign, where the normalized spectrum is zero. The end band for the spectral anti-sparseness processing is specified by *Banti* (=max(*core\_sfm*,*Nband\_LF*-1)) and the end frequency is specified by *Lanti*(=*kend*(*Banti*)).

(1859)

where is a random seed and updated by .

After applying this anti-sparseness processing, the energy is further modified by applying the low band dequantized envelope as described in subclause 6.2.3.2.5.1.

6.2.3.2.5.6.2 Control of dynamics based on the excitation class

Following the spectral anti-sparseness processing, the spectrum is further modified by additional processing to control the dynamics.

The spectrum is first normalised by calculating the envelope of the processed spectrum, then dividing the spectrum by this envelope. The window size*,*, for this normalisation depends on the signal characteristics..

The 256 low frequency MDCT coefficients in the 0-6400 Hz frequency range, are split into 16 sharpness bands (16 coefficients per band). In sharpness band *j*, if and , the counter is incremented by one.

The maximum magnitude of the spectral coefficients in a sharpness band, denoted, is:

(1860)

Parameteris initialized to 0 and calculated for every frame. Then the normalization length is obtained:

(1861)

where the current normalization length is calculated as follows:

(1862)

and the current normalization length is preserved as.

The spectrum is then normalized

(1863)

where is the number of bands used in the control of dynamics.

The sign vectors for the spectrum are then removed, leaving just the magnitude, and the mean is then calculated for each band *p*. The bands are 16 frequency bins wide, and start at frequency bin 2. For the SWB case, at 24.4kbps there are 9 bands ending at frequency bin 145, while for 32kbps there are 8 bands, ending at frequency bin 129. For the FB case, at 24.4kbps there are 19 bands ending at frequency bin 305, while for 32kbps there are 18 bands, ending at frequency bin 289

The amplitude of each frequency bin is then reduced by a dynamics control factor of the difference between the bin amplitude and mean of the band.

(1864)

where *drf* is the dynamics control factor depending on the decoded excitation class, (1865)

The original signs are then re-applied for the *HF\_Speech\_excitation\_class* and the *HF\_excitation\_class1*; however random signs are used for the *HF excitation\_class0*. If is higher than 0, the original sign is re-applied, otherwise, a reversed sign of the original is applied. The initial random seed is

(1866)

where is the number of allocated integer bits for each band.

The spectrum is then normalised:

(1867)

The normalised spectrum is then copied, using the mapping in table 179, to create the high frequency excitation spectrum.

Table 180: Frequency mapping to generate high frequency excitation spectrum

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | *l* |  |  |  |  |
| 24.4kbps | 0 | 2 | 129 | 320 | 447 |
| 1 | 2 | 129 | 448 | 575 |
| 2 | 80 | 143 | 576 | 639 |
| 24.4kbps FB | 3 | 144 | 303 | 640 | 799 |
| 32kbps | 0 | 2 | 129 | 384 | 511 |
| 1 | 2 | 129 | 512 | 639 |
| 32kbps FB | 2 | 130 | 289 | 640 | 799 |

Finally the high frequency excitation spectrum is adjusted at the junction boundaries,

(1868)

where ,, and .

If and then

(1869)

where and

At 24.4kbps,

(1870)

where ,, and , and then

(1871)

where, and .

6.2.3.2.5.7 Spectral envelope adjustment

Spectral envelope adjustment is used to generate high frequency spectrum with combining the high frequency excitation spectrum and the interpolated envelope according to the frequency. The low frequency envelope is used in the first band. The interpolated envelope is calculated as follows:

(1872)

where *p* (*p*=1,…,*Nband\_G*-1) is a band index;

*wp*(*k*) is a interpolation function where and ; and

is the initial value of the spectral envelope .

The envelope is then multiplied by the generated high frequency excitation spectrum in subclause 6.2.3.2.5.6.

(1873)

In the FB case, the maximum value of *k* in equations (1874) and (1875) is corrected to 799 and a decision is made on whether or not to interpolate the envelope by comparing the envelope of the first band in FB and the last band in SWB.

(1876)

where *pb* is the index of the first band in FB, 14 at 24.4kbps or 12 at 32kbps.

6.2.3.2.5.8 Spectral combining

The final step of the Generic mode is to combine the noise filled spectrum, obtained from decoding the quantized spectrum in subclauses 6.2.3.2.5.4 and 6.2.3.2.5.5, with the high frequency spectrum, generated in subclause 6.2.3.2.5.7. The noise filled spectrum includes the low frequency spectrum and some bands in the high frequency spectrum where bits were allocated during the spectral quantization. The generated high frequency spectrum includes only the high frequency spectrum.

There are two kinds of overlap bands between these two spectra; one is a partial overlap at the junction between the low frequency and the high frequency (376~400 at 32kbps, 304~328 at 24.4kbps). The other is a full overlap due to the difference between the two band allocations, i.e. due to some bands in the high frequency spectrum being allocated bits during the spectral quantisation.

In the partial overlap band, the spectral combining is performed based on an overlap and add process. If there are any allocated bits from the spectral quantizer, the noise filled spectrum is used directly for the final decoded spectrum. If there were no bits allocated by the spectral quantizer, a overlap and add process between the two spectra is performed:

(1877)

where is the overlapped length at the junction band, 16 at 24.4kbps and 8 at 32kbps.

In the full overlap bands, the spectrum is combined in a selective way. If there are any allocated bits from the spectral quantizer, the noise filled spectrum is used directly for the final decoded spectrum. If there were no bits allocated by the spectral quantizer, the high frequency spectrum is used for generating the final decoded spectrum .

##### 6.2.3.2.6 PVQ decoding and de-indexing

6.2.3.2.6.1 High dynamic range arithmetic decoding

The PVQ-codewords are extracted from the bit stream using the Range decoder.

6.2.3.2.6.2 Split-PVQ decoding approach

The PVQ-split parameters are obtained as inverse of the functions in subclause 5.3.4.2.7.2

6.2.3.2.6.2.1 Split-PVQ Decoder band splitting calculation

The initial number of segments (parts) is computed according to the first equation in subclause 5.3.4.2.7.2.1. In case and the band bit rate is high, the flag is read from the bit stream. Finally, is computed as

(1878)

6.2.3.2.6.2.2 PVQ sub vector gain decoding

The decoded Split-PVQ angles are converted into sub-vector gains.

6.2.3.2.6.3 PVQ sub-vector MPVQ de-indexing

First the, values for the sub vector to be decoded are used in a ‘FindSizeAndOffsets(N,K)’ function which pre-computes the row of the MPVQ offset matrix [, ] and also calculates the integer size of the MPVQ-index *MPVQ-size* using this last row. The last four equation in subclause 5.3.4.2.7.4.1 are employed for the offset and size calculations.

Secondly the 1 bit leading sign index and the MPVQ-indexis obtained from the Range decoder using the calculated *MPVQ-size* information. The leading signis decoded from the 1 bit leading sign index, where a zero sign index yields a positive value of “+1”, and a non-zero sign index yields a negative value of “-1”.

The third step is the actual MPVQ-de-indexing scheme, converting the leading sign and the to a valid integer vector .

The MPVQ de-indexing loop is carried out according to figure 113, where is the MPVQ-index, is the current row number of the MPVQ offset matrix, is the pointer into the samples/coefficients of the received PVQ-vector . The function “FindAmplitudeAndOffset” obtains the amplitude and the MPVQ indexing offset for the current number of accumulated pulses, by searching in the current row in the MPVQ offset matrix. Further the function “UpdateOffsetsBwd” iteratively updates the required MPVQ-offsets for the next larger dimension using combinations of the last four equations in subclause 5.3.4.2.7.4.1. The function “GetLeadSign” obtains the next leading sign value from the LSB of , and shifts the one bit to the right. On the decoder side the MPVQ recursion is run in the order of position 0 to position, with a dimension decreasing from to 1.



Figure 113: Detailed MPVQ-de-indexing

The calculation of the indexing offset matrix is optimized to use direct calculations up to row for any combination of and , further if the number of unit pulses are low enough and the dimension is 5 or lower, a direct row initialization of the offset matrix is used for the offset determination, where the last column in row is calculated using the low dynamic “row-only” relation:

(1879)

### 6.2.4 Frequency-to-time transformation

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

##### 6.2.4.1.1 eDCT

The IDCTIV is identical to the DCTIV and is given by the following equation, with omitted normalization:

(1880)

##### 6.2.4.1.2 Unfolding and windowing

The frame , , coming from the inverse eDCT transform is unfolded in order to obtain two frames that can be used for overlap- add with the previous unfolded frame to remove the aliasing introduced by the folding process at the encoder.

Similar to the folding done at the encoder, unfolding and window decimation operations are combined in the same process to automatically resample the ALDO windows at 48 and 25.6 kHz while keeping perfect reconstruction conditions. The decimation factor and offset parameters are the same are the one used in the encoder.

The frame issued from the eDCT inverse transform is unfolded into a block of length . The ALDO window is stored at a sampling rate corresponding to two frames of length (). The ratio between and is called the decimation factor (). The unfolding and windowing process is illustrated in figure 114.

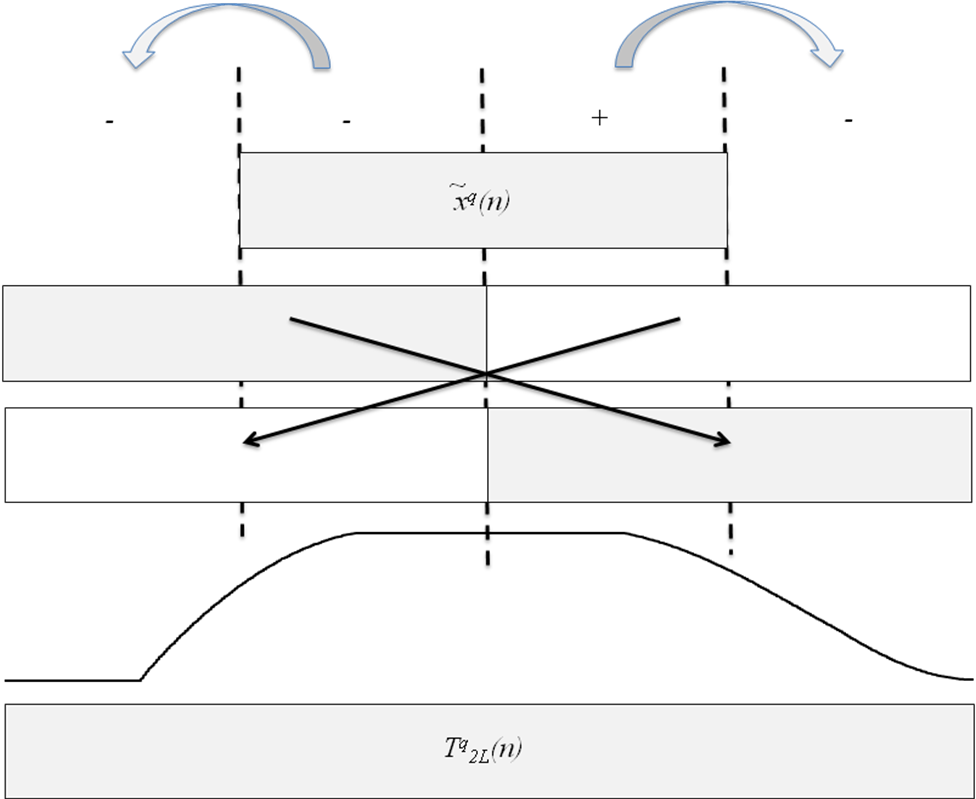


Figure 115: Unfolding and windowing with ALDO window.

The unfolded frame is obtained for :

, (1882)

, (1883)

, (1884)

, (1885)

where is the time-reversed version of the ALDO window used in the encoder

, (1886)

is the decimation factor and is the offset.

For the 32 kHz case, to have perfect reconstruction, the ALDO window decimated from 48 to 16 kHz applied on one sample over 2, the other samples are weighted by a complementary window . For this 32 kHz case, the unfolded frames are given by:

, (1887)

, (1888)

, (1889)

, (1890)

, (1891)

, (1892)

, (1893)

, (1894)

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

##### 6.2.4.1.3 Overlap-add

Finally, the output full-band signal is constructed by overlap-adding the signals for two successive frames:

(1881)

##### 6.2.4.1.4 Pre-echo attenuation

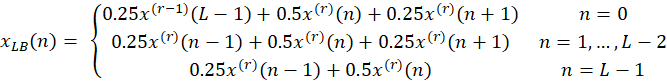
A typical artefact in transform coding known as pre-echo is observed especially when the signal energy grows suddenly, like speech onsets or music percussions. The origin of pre-echoes is explained below. The quantization noise in the frequency domain is translated into the time domain by an inverse MDCT transform and an add/overlap operation. Thus the quantization noise is spread uniformly in the MDCT synthesis window. In case of an onset, the part of the input signal preceding the onset often has a very low energy compared to the energy of the onset part. Since the quantization noise level depends on the mean energy of the frame, it can be quite high in the whole synthesis window. In this case, the signal to noise ratio (SNR) is very low (often negative) in the low energy part. The quantization noise can be audible before the onset as an extra artificial signal called pre-echo. To prevent the pre-echo artefact, an attenuation scheme is necessary when there is a significant energy increase (attack or onset) in some part of the synthesis window, and the pre-echo reduction has to be performed in the low energy part of the synthesis window preceding the onset. In the following, this low energy part preceding an onset will be referenced as "pre-echo zone". On the other hand the signal energy after pre-echo reduction should not lower than the mean energy in the preceding frames. However, if the preceding frame have low frequency spectrum, knowing that the pre-echo has often white noise like spectrum, even if the energy of the pre-echo zone is reduced to the level of the previous frames the pre-echo is still audible in the higher frequencies.

To improve the pre-echo reduction, an adaptive spectral shaping filtering is applied in the pre-echo zone up to the detected attack or onset to eliminate undesirable higher frequency pre-echo noise. This adaptive spectral shaping filter is realized by a two-band filterbank: the decoded signal is decomposed into two sub-signals according to a frequency criterion to obtain two sub-bands and a pre-echo attenuation factor is calculated in the determined pre-echo zone for each sample in both sub-bands. The attenuation factors of the sub-bands that determinate the spectral response of the filter are computed in function of several parameters of the full-band and sub-band signals as detailed below. The pre-echo attenuation is made in the sub-bands by applying these attenuation factors in the pre-echo-zone. Finally the two attenuated sub-bands are combined to obtain the pre-echo attenuated decoded signal. The pre-echo attenuation is activated for received frames, when the previous frame was also received, and when the bitrate is not higher than 32 kbit/s.

A pre-echo in the current frame can be caused by a sharp onset in the current or the next frame, as the MDCT analysis window covers these two consecutive frames. An onset in the next frame can be detected by analysing the memory of inverse MDCT that will be used in the next frame in the overlap-add operation. A discrimination of pre-echo/non-pre-echo zones and the attenuation factor computation are based on two signals of the inverse MDCT transform: on the decoded output full-band signal , and on the first un-windowed memory of inverse MDCT , that will be used in the next frame in the overlap-add operation to synthesize the output content for the next frame and the pre-echo reduction is done in echo zones preceding the onsets.

**Decomposition in two sub-bands**

The decoded signal is decomposed in a lower and an upper frequency band sub-signals. These signals are computed by applying an adaptive zero-delay FIR filter with transfer function  in low-band, with  = 0.25 in the current frame and 0 otherwise; the high-band is given by the complementary filter. The first, lower band sub-signal is obtained by a first filtering of the full-band signal by the low-pass filter

 (1896)

and the second, higher band sub-signal is obtained by subtracting the lower band sub-signal from the decoded signal:

(1882)

For the memory part only the higher-band component is computed as

(1883)

**Discrimination procedure of pre-echo/non-pre- echo zones**

The discrimination procedure between pre-echo zones and non-pre-echo zones is based on the concatenated signal formed from , and , . This signal is divided in sub-blocks and its temporal envelope is computed.

The current frame part of the concatenated signal, , is divided into sub-blocks of samples where =8 (2.5 ms sub-blocks). The temporal envelope of this signal is computed as successive sub-block energies.

, (1884)

The memory part of the concatenated signal forms one sub-block, its energy is computed as

(1885)

The energy of the first half and the first ¾ samples of each sub-blocks of the current frame are also memorized:

, (1886)

, (1887)

The temporal envelope of the higher band in the current frame is also computed:

, (1888)

Then, , is then modified as follows:

(1889)

In this paragraph, index is used for samples, and index is used for sub-blocks.

In the concatenated signal the sub-block with maximal energy, including the memory sub-block, is also searched:

(1890)

The transition of the temporal envelope to a high-energy zone is detected in the sub-block with the index given by:

(1891)

Note that when =0 either no pre-echo attenuation is made or the pre-echo attenuation of the previous frame is finished on the first samples of the current frame.

The zero-crossing rate *,* is also computed for each sub-block. A zero-crossing is detected when the product of two consecutive samples is smaller or equal to 0. The parameter *,* is defined as the number of times when the following condition is verified:

, (1892)

The zero crossings between two consecutive sub-blocks count for the next sub-block. The zero crossing rate of the memory part is also computed.

The maximum length without zero crossing *,* is also stored for each sub-block. A period without zero crossing that covers a sub-block border is taken into account for the previous sub-block.

The maximal energy is compared to that of the preceding sub-blocks:

, (1893)

The low energy sub-blocks preceding the sub-block in which a transition has been detected with > 16 are determined as pre-echo zone. However in the following cases the sub-block is considered as non-pre-echo zone:

if

or

if and

or

if and

where, computed in the previous frame and memorised,

(1894)

and = 10 for narrowband signals and 16 otherwise.

Even the previous sub-blocks are considered as non-pre-echo zone if their energy is higher than *.*

The pre-echo attenuation of low energy sub-block determined as pre-echo zone is made by multiplying the two sub-band signals, the lower band and the higher band by attenuation factors and respectively, where and are determined as a function of the temporal envelope of the concatenated signal *,*.

For each sample of the pre-echo zone sub-blocks, these gains are set to 0.01 if > 32 and to 0.1 otherwise. For the other sub-blocks, the initial gains are set to 1, they form the non-pre-echo zone. Following this is set as the index of the first non-echo sub-block (where the initial gain is equal to 1).

A false alarm detection is made at this point. If the last pre-echo attenuation gain in the previous frame is higher than 0.5 and in the current frame only one sub-block has attenuation gain of 0.1 and the other gains are 0, is set to 0.

The initial pre-echo attenuation gains depend also on the energy of the previous frame: a minimal attenuation value for each pre-echo zone sub-block and for both sub-bands are also fixed as a function of the temporal envelope of the reconstructed signal of the previous frame. This value is fixed in a way that the attenuated sub-block energy in the sub-band cannot be lower than the pre-echo attenuation gain compensated mean energy of the previous frame in that sub-band, to preserve background noise energy. In the lower band:

(1895)

for and where was computed in the previous frame as:

(1896)

However is set to 1 if or .

In a similar way the initial pre-echo attenuation gain for the higher band signal is computed as:

(1897)

for where

(1898)

Note that the initial attenuation gain in both the lower band and the higher band are identical for each samples of a sub-block.

Before applying the pre-echo attenuation gains the position of the onset is refined. If the onset was detected in the current frame, each sub frames from index  to are divided into sub-sub-blocks where =4 if the sampling frequency is 8 kHz and =8 otherwise. If = 0 only the first sub-block is considered.

The energy of these sub-sub-blocks is computed:

, (1899)

where

(1900)

and

(1901)

When the onset was detected in the future memory part , only the first samples are examined and and

, (1902)

The maximum of these values is searched:

, (1903)

The values are compared to adaptive thresholds. The first one is independent of the sub-sub-block index:

(1904)

The second one is computed as:

(1905)

where

(1906)

If and this value is modified as:

(1907)

Initially the starting position of the onset for both the lower and the higher band is the beginning of the sub-block . This position is delayed by samples by sub-sub-blocks as long as *.*  The pre-echo attenuation gain of these samples moved from the non-pre-echo zone to the pre-echo-zone is set equal to the gain of last sample of the original pre-echo zone and respectively in the 2 sub-bands. In the following these new samples in the pre-echo zone are considered as the part of the last pre-echo zone sub-block (index ), the length of this sub-block can be longer than .

To avoid false pre-echo detection, the energies of the last 2 or 3 sub-blocks preceding the onset is verified for both the full-band and the high-band signals: the regression coefficient for these sub-blocks energies is computed by the least squares estimation technique and compared to thresholds. If at least one regression coefficient is lower to its threshold the pre-echo attenuation is inhibited. In fact it is checked whether the sub-blocks preceding the onset have stable or increasing energy, this is always true for pre-echos. For easy comparison to threshold the regression coefficients are normalised by the sub-band energies when the threshold is different to 0. If the threshold is 0, only the sign of the regression coefficient is checked, no normalisation is needed.

When the onset is detected in the first or second sub-block this verification is not possible.

When the onset is detected in the third sub-block only the high-band regression coefficient is computed and compared to the threshold . As only the sign is checked here no normalization is needed for the regression coefficient:



If  <  the pre-echo attenuation in the pre-echo zone is inhibited.

When the onset is detected in the fourth or later sub-block both the full-band regression coefficient  and the normalized high-band regression coefficient  are computed on the last 3 sub-blocks preceding the onset and they are compared to the thresholds  and  respectively. Let’s note the index of the sub-block where the onset is detected .

In the full-band only the sign is checked, no normalization of the regression coefficient is needed:



In the higher band the normalized regression coefficient is estimated as:



The comparison  is equivalent to :



If the  <  or  <  the pre-echo attenuation in the pre-echo zone is inhibited.

The pre-echo attenuation functions and are stair-like, the gain is constant within a sub-block. To avoid annoying noise due to this discontinuity, the final pre-echo attenuation gain for the lower band is obtained by linear smoothing of the initial pre-echo attenuation gain introducing intermediate levels between the gains of consecutive sub-blocks. For narrow band signals = 20, for other bandwidths = 4. This smoothing is done before the detected onset position and at the beginning of each sub-block. For the first sub-block the smoothing is done between the memorized last gain value of the previous frame and the gain of the first sub-block of the current frame. If the onset position is detected in the next frame no smoothing is done at the end of the frame, this will be done at the beginning of the next frame. For example at the beginning sub-block *,*  and if the gains determined for the sub-blocks and are and respectively, for wide band signals (= 4) the gains are smoothed in the following way:

Before smoothing:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | … |  |  |  |  |  |  |  |  | … |
| gain | … |  |  |  |  |  |  |  |  | … |

After smoothing:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | … |  |  |  |  |  |  |  |  | … |
| gain | … |  |  |  |  |  |  |  |  | … |

In the higher band no smoothing is necessary, .

In both sub-band, the pre-echo is attenuated in the pre-echo-zone by applying these gains to the sub-signals:

(1908)

(1909)

The final pre-echo attenuated synthesized signal is obtained by combining the two attenuated sub-signals:

(1910)

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

The configuration for the overlap and transform lengths is depends on the overlap code for the current frame and on the overlap code for the previous frame as described in subclause 5.3.2.3, where the overlap code is obtained as described in subclause 6.2.2.2.1.

#### 6.2.4.3 Short block transformation

##### 6.2.4.3.1 Short window transform in TDA domain

This processing is done when the Transient mode is selected*,* the spectrum is first de-interleaved into four spectra with m = 0,...,3. This operation is the inverse of the interleaving performed in the encoder, see subclause 5.3.2.4.1.3.

The four spectra corresponding to short 5-ms transforms are first transformed to the time-aliased domain using the short inverse DCTIV transforms. The obtained signals are denoted  and are each of a length.

The obtained time-domain aliased signals are further expanded into the time domain by using the inverse time-domain aliasing operation. This operation can be seen as a pseudo-inverse of the matrix used in equation (8) (with replaced by).

Formally, this is performed according to:

(1911)

The length of the resulting signal for each sub-frame index is equal to double the length of the input spectrum, i.e., .



Figure 116: Algorithm for inverse transform in the case of transient mode.

The resulting time domain aliased signals for each sub-frame are windowed using the same configuration of windows as those in the encoder. The resulting windowed signals are overlap‑added. Note that the window for the first m = 0 and last m = 3 sub-frame is zero. This is due to the zero padding that is used in the encoder. These two frame edges do need to be computed and are effectively dropped. The resulting signal of the overlap-add operations of all sub-frames is reordered using the inverse operation performed in the encoder, which leads to the signal , . An overview of these operations is shown in figure 117. Then windowing and overlap-add are performed on the same as for the long window transform in subclause 5.3.2.2.

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

After choosing the transform and overlap length configuration for transient frame as described in subclause 6.2.4.2 each sub-frame (TCX5 or TCX10) is windowed and transformed using the inverse MDCT, which is implemeted using IDCTIV and inverse TDA.

#### 6.2.4.4 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 6.2.4.2. The transition window is derived from the ALDO and sine window overlaps.

##### 6.2.4.4.1 ALDO to short transition

The left part of the transition window uses the left slope of the ALDO synthesis window (short slope), which has a length of 8.75ms (see figure 118).

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

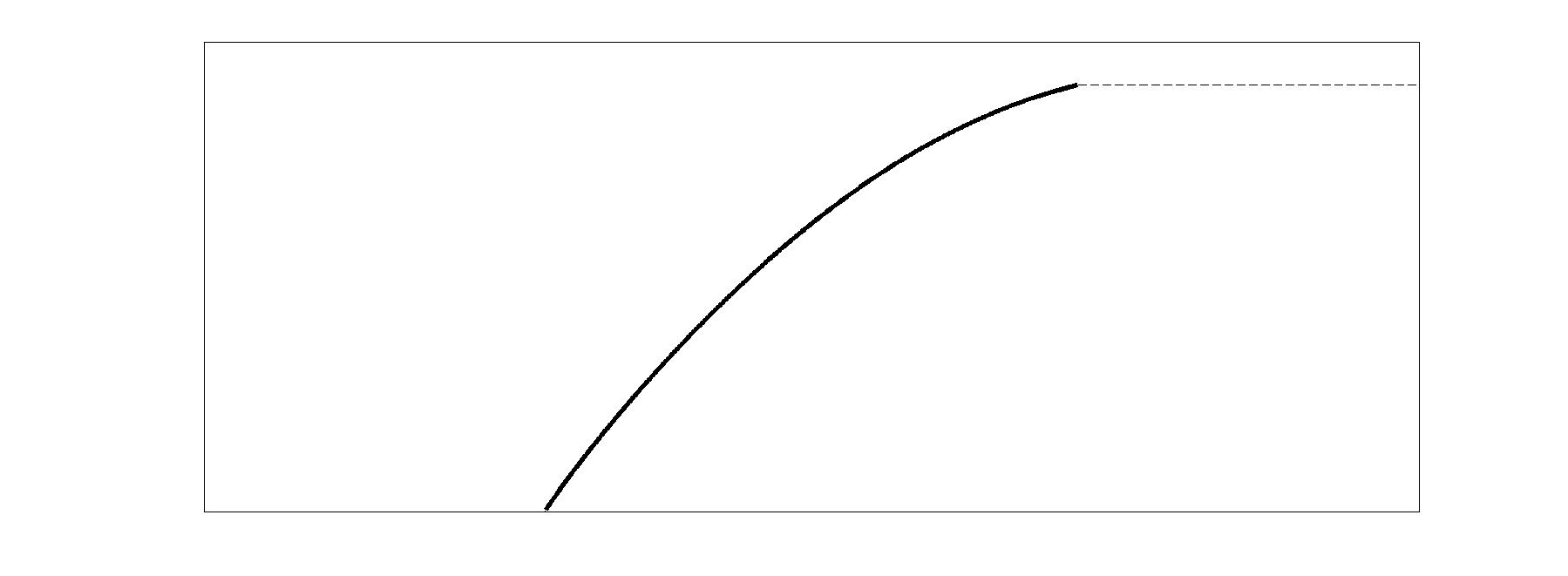


Figure 119: ALDO to short transition

##### 6.2.4.4.2 Short to ALDO transition

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

The right overlap of the transition window has a length of 8.75ms (FULL overlap) and is derived from the ALDO window. The right slope of the ALDO synthesis window (long slope) is first shortened to 8.75ms by removing the last 5.625ms. Then the last 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 figure120 .

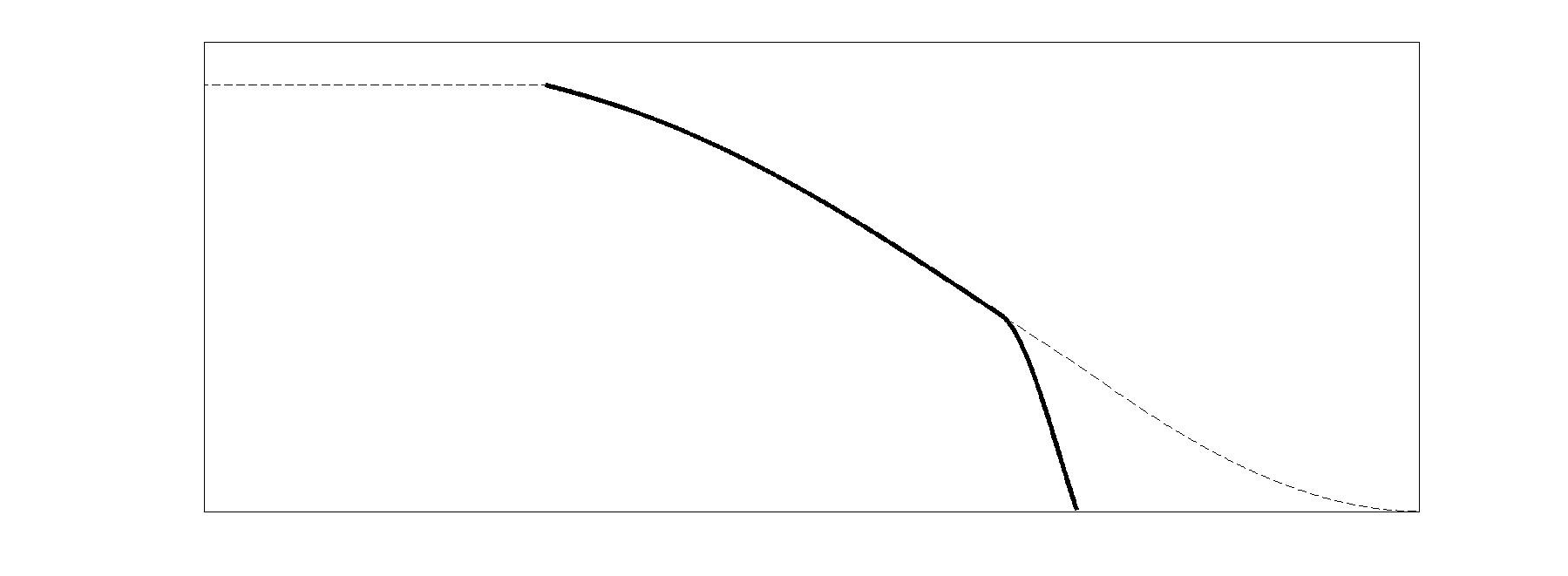


Figure 121: Short to ALDO transition

#### 6.2.4.5 Low Rate MDCT Synthesis

In addition to the full MDCT synthesis of length , which gives an output signal at the configured output sampling rate , a further MDCT synthesis of the lower part of the spectrum is performed to obtain an output signal at the CELP sampling rate . The low rate output is required for switching from TCX to ACELP. An MDCT synthesis of length is performed on the lowest MDCT coefficients. In case the output sampling rate is set lower than the CELP sampling rate (so that ), the spectrum is padded with zeroes to the required length . The low rate synthesis transform is performed as defined above with the only difference being the transform length.

## 6.3 Switching coding modes in decoding

### 6.3.1 General description

This clause describes all transitions between coding modes including changes for sample rates, bit rates and audio bandwidths for the decoding process. The transitions between CELP coding mode and MDCT coding mode within the same bit rate and audio bandwidth are described in 6.3.2 and 6.3.3.

The handling of sample rate changes within the CELP or LP-based coding mode and MDCT-based TCX mode is described in 6.3.4.

The switching between primary and AMR-WB IO modes is described in 6.3.5.

The handling of transitions in the context of bit rate switching is described in 6.3.6.

Finally, the transition between NB, WB, SWB and FB are described in 6.3.7.

### 6.3.2 MDCT coding mode to CELP coding mode

When a CELP encoded frame is preceded by a MDCT based encoded frame, the memories of the CELP encoded frame have to be updated before starting the decoding of the CELP frame, similarly to the encoder case (see clause 5.4.2).

Additionally, cross-fading is applied in the time-domain at the output sampling rate to avoid any discontinuities between the MDCT based output and the CELP based output including bandwidth extension.

The CELP memories update and the cross-fading are performed depending on the bitrate and the previous encoding mode. In general three different MDCT to CELP (MC1 to MC3) transitions are supported. The table in clause 5.4.2 describes which transition mode is used for a specific configuration. The different decoding cases are described in detail in the following sub-clauses.

#### 6.3.2.1 MDCT to CELP transition 1 (MC1)

MC1 is used when the previous frame was decoded with HQ MDCT and the current frame is decoded with CELP. The CELP state variables are reset in the current frame to predetermined (fixed) values. In particular the following memories are reset to 0 in the CELP decoder:

* Resampling memories of the CELP synthesis
* Pre-emphasis and de-emphasis memories
* LPC synthesis memories
* Past excitation (adaptive codebook memory)
* Bass post-filter memories

The old LPC coefficients and associated representations (LSP, LSF) and CELP gain quantization memories are reset to predetermined (fixed) values. The CELP decoder in the current frame is forced to operate in Transition coding (TC), i.e. without using an adaptive codebook from the previous frame. Since the LPC coefficients from the previous frame are not available, only one set of LPC coefficients corresponding to the end of frame are decoded and using for all subframes in the current frame.

To avoid discontinuities at the output sampling rate between the decoded HQ MDCT signal in the previous frame and the decoded CELP signal (including time-domain BWE) in the current frame, cross-fading (i.e. overlap-add) is used.

The decoded HQ MDCT signal could be windowed to compensate for the MDCT synthesis window, however in practice this compensation is not done. Note that the synthesis of the CELP decoder is more delayed than the synthesis of the MDCT decoder as shown in Figure 112a; the time difference is denoted here *D*. In the current CELP frame, the first samples are replaced by the HQ MDCT synthesis from the previous frame (D samples). The unweighted HQ MDCT aliased memory is overlap-added with the CELP output.



Figure 122a: Overlap-add in MDCT to CELP transition.

#### 6.3.2.2 MDCT to CELP transition 2 (MC2)

As described in clause 6.2.4.5, the MDCT based TCX decoder generates two time-domain signals, one at the output sampling rate and one at the CELP sampling rate . The signal at the CELP sampling rate is used to update the CELP memories, similarly to the encoder case (see clause 5.4.2.2). It is also used at the decoder side to update the memories of the CLDFB based resampler that is used to resample the decoded CELP signal. The signal at the output sampling rate is the signal that is actually sent to the decoder output. To avoid discontinuities between this MDCT based TCX signal and the decoded CELP signal at the output sampling rate, cross-fading is used.

The decoded CELP signal at the output sampling rate is obtained after CELP decoding at the CELP sampling rate, CLDFB based resampling and time-domain bandwidth extension decoding (see clause 6.1). Both the CLDFB based resampling and the time-domain bandwidth extension decoding introduce a delay. Consequently, the decoded CELP frame is delayed compared to the MDCT based TCX frame and an overlap between the two frames is then introduced. This overlap is used to perform a cross-fade and thus to avoid any discontinuities between the two frames. The output signal is given by

(1927)

with is the output signal, is the MDCT based TCX signal, is the CELP signal, is the delay of the TCX LTP postfilter (0.25ms), is the delay introduced by the CLDFB resampler and the time-domain BWE (1.25ms if and 2.3125ms otherwise), and is the frame length at the output sampling rate (20ms).

#### 6.3.2.3 MDCT to CELP transition 3 (MC3)

Similarly to MC1, the CELP memories are either reset or extrapolated similarly as in the encoder (see clause 5.4.2.3).

To avoid discontinuities between the decoded MDCT based TCX signal and the decoded CELP signal (including time-domain BWE) at the output sampling rate, cross-fading is used. The same approach as used in clause 6.3.2.1 is used.

### 6.3.3 CELP coding mode to MDCT coding mode

When a MDCT encoded frame is preceded by a CELP encoded frame, a beginning portion of the MDCT encoded frame cannot be reconstructed properly due to the aliasing introduced by the missing previous MDCT encoded frame. As already described in clause 5.4.3, two approaches are used to solve this problem, depending on the MDCT based coding mode (either MDCT based TCX or HQ MDCT). The decoder part of these two approaches is described in detail in the following sub-clauses.

#### 6.3.3.1 CELP coding mode to MDCT based TCX coding mode

If MDCT based TCX is used in the current frame and if the previous frame was encoded with CELP, the MDCT based TCX frame length is increased and the left part of the MDCT window is modified, as already described in clause 5.4.3.1.

At the decoder side, the MDCT coefficients are decoded as described in clause 6.2.2. The decoded MDCT coefficients are then transformed back to the time-domain as described in clause 6.2.4. Two inverse transforms are performed and two time-domain signals are generated, as described in clause 6.2.4.5. One signal is generated at the CELP sampling rate and follows the previous decoded CELP signal at the CELP sampling rate. The other signal is generated at the output sampling rate and follows the previous decoded CELP signal at the output sampling rate (after CLDFB resampling and time-domain BWE). The decoding of the CELP signal (including the time-domain BWE) was described in clause 6.1.

To avoid any discontinuities that could be introduced in the decoded time-domain signal at the border between the two frames, a smoothing mechanism is applied. This algorithm is described in detail in the following.  
Let’s first assume the following notations. The frame length at the CELP sampling rate is noted. The decoded CELP signal at the CELP sampling rate is noted with . The decoded MDCT signal (including the windowed portion overlapping with the previous CELP frame) is noted with , is the length of the segment where both the MDCT signal and the CELP signal overlap (it is also equal to the length of the sine window used in the left part of the transition window as described in clause 5.4.3.1), and is the CELP sampling rate. The LPC synthesis filter used in last subframe of the previous CELP frame is noted with and is the LPC filter.  
A modified CELP signal is first computed by performing an overlap-add operation with the decoded MDCT signal on the overlap region and by artificially compensating the aliasing introduced by the decoded MDCT signal using the decoded CELP signal. The modified CELP signal can be defined as follow

(1928)

with is the sine window used in the left-part of the transition window as described in clause 5.4.3.1. Contrary to the non-modified CELP signal case, discontinuities are significantly reduced (or even completely supressed in most cases) in the modified CELP signal preceding the MDCT signal, due to the overlap-add operation. However, the modified CELP signal cannot be used directly to generate the decoder output signal of the current frame, because it would introduce an additional decoder delay equal to the overlap length. Instead, the modified CELP signal is used only to generate a zero-input-response (ZIR) of the LPC synthesis filter. This ZIR is then used to modify the decoded MDCT signal, reducing significantly (or even removing in most cases) the possible discontinuity, and without introducing any additional decoder delay. The ZIR of the LPC synthesis filter is generated by first computing the memory of the LPC synthesis filter as follow

(1929)

and then computing the zero-input-response as follow

(1930)

with is the number of generated ZIR samples. The ZIR is then windowed such that its amplitude always decreases to 0, producing the windowed ZIR

(1931)

Finally, the windowed ZIR is added to the beginning portion of the decoded MDCT signal, corresponding to the time samples .

The same smoothing mechanism is applied to the decoded signals at the output sampling rate, with the exception that the ZIR is not re-computed but obtained by resampling the ZIR computed at the CELP sampling rate. The resampling is performed using linear interpolation as described in clause 5.4.4.4. The resampled ZIR is then added to the decoded MDCT signal at the output sampling rate.

The smoothing mechanism described above ensures a smooth transition between the CELP and MDCT signals at the output sampling rate, but only in the CELP bandwidth part. Due to the delay introduced by the time-domain BWE, a gap is introduced in the high frequency region as explained in clause 6.1.5.1.13.1. To fill this gap, a transition signal is generated as described in clause 6.1.5.1.13.1. This transition signal is long enough to cover not only the gap but also an additional signal portion following the gap. This additional signal portion is then used to perform a cross-fading with the decoded MDCT signal, ensuring smooth transition at the output sampling rate.

#### 6.3.3.2 CELP coding mode to HQ MDCT coding mode

When the previous frame is CELP and the current frame is to be coded by HQ MDCT, the current frame is a transition frame in which two types of decoding are used:

* Constrained CELP coding and (when required) simplified time-domain BWE coding
* HQ MDCT coding with a modified window

Constrained CELP decoding means here that CELP is restricted to decode only a subset of CELP parameters, to reuse parameters (LPC coefficients) from the previous CELP frame, and to cover only the first subframe of the current frame. These constraints are set to minimize the bit budget taken by continuing CELP decoding in the current frame, this bit budget being taken out of HQ MDCT decoding.

As shown in Figure 112b, the transition frame includes at the decoder side a gap between the previous output frame (decoded by CELP) and the decoded signal with only the contribution from HQ MDCT. The length of this gap at the decoder is 4.375 ms, which corresponds to 10-5.625 ms (10 ms for ¼ of MDCT window support – 5.625 ms which is the length of the zero segment at the beginning of the ALDO synthesis window). In addition, an overlap period of 1,825 ms is used to attenuate discontinuities between CELP and HQ MDCT decoded signal. The total transition region between CELP and HQ MDCT in the decoder (grey zone decoder in Figure 112b) is 4.375+1.825 = 6.25 ms..



Figure 112b: Modified MDCT synthesis window in the transition frame (CELP to HQ MDCT).

##### 6.3.3.2.1 Constrained CELP decoding and simplified BWE decoding

The bit budget for CELP and BWE in the current (transition) frame is determined depending on the CELP coder used in the previous frame (12.8 kHz or 16 kHz) and decoded audio bandwidth in the current frame, as described in the pseudo-code in clause 5.4.3.2.1. Note that the current frame being a transition frame, one bit is used to indicate the type of CELP coding (12.8 kHz or 16 kHz); this bit is allows decoding the transition frame even when the information about the previous CELP coding type was lost due to frame erasures..

LPC coefficients from the end of the previous frame are reused, constrained CELP decoding only relies on decoding an extra subframe with the same CELP core decoder (12.8 kHz or 16 kHz) as in the previous frame; subframe decoding similar to clause 6.1 is applied (without LSF decoding). The length of the decoded subframe is 5 ms if CELP is at 12.8 kHz and 4 ms if CELP is at 16 kHz. The decoded CELP synthesis is normally delayed by 1.25 ms at the decoder due to the FIR resampling/delay operation. However, to have enough samples to cover the transition region of 6.25 ms, the resampling memory is resampled with 0-delay using the optimized cubic interpolation described in clause 6.3.3.2.1.1.

Note that the cubic interpolation requires to have two future samples, which are estimated by simply repeating the last value of the FIR memory.

Hence, if CELP is at 12.8 kHz,, the CELP synthesis (after FIR resampling and 0-delay resampling) is 6.25 ms long; if CELP is at 16 kHz, the subframe is 1 ms shorter and the CELP synthesis is extended by ringing to get a decoded signal of 6.25 ms; the ringing is used only in the overlap region and its influence is perceptually minimal.

When the coded audio bandwidth is higher than the bandwidth of the core CELP coder, BWE decoding is applied. The previous frame is high-pass FIR filtered to obtain the high-band, and the decoded pitch lag and gain are decoded to repeat 6.25 ms of high-band signal which is added to the 6.25 ms of decoded CELP signal.

###### 6.3.3.2.1.1 Optimized cubic interpolation

The missing signal at the output sampling frequency is partly available in the memory buffer at the internal sampling frequency, 12.8 kHz or 16 kHz. By doing low delay resampling like interpolation method of this memory a good estimation of the missing signal can be obtained. Third order cubic interpolation is used here, where cubic curves are used to interpolate the output values within 3 input interval delimited by 4 input samples. Respectively, in each input interval the interpolation can be made by using 3 different cubic curves. To further improve the quality of this estimation the interpolated samples are obtained by computing a weighted mean value of the possible cubic interpolated values computed on the plurality intervals covering the time position of the sample to interpolate.

The length of the resampling buffer (input to cubic interpolation) is 1.25 ms (16 samples at 12.8 kHz sampling rate or 20 samples at 16 kHz sampling rate) plus 2 past samples used as memory for the first cubic interpolations of the first 2 intervals. In cubic interpolation, 4 consecutive input samples determinate a cubic curve, the general equation of this curve is . To simplify the computations of the coefficients the temporal index of the 4 consecutive input samples are always considered as , , and and so they define 3 intervals, [-1, 0], [0, 1] et [1, 2]. Noting the values of these 4 input samples, , and, the coefficients , , and can be computed as:

(1932a)

(1933b)

(1934c)

(1935d)

To get the output resampled signal often the value of the output is needed to be determined between two input samples, in the interval limited by these input samples. As mentioned above, in cubic interpolation one cubic curve covers 3 intervals and respectively each interval can be covered by 3 different cubic curves: by the interval central [0, 1] of the central cubic curve or by the interval [1,2] of the previous cubic curve or the interval [-1, 0] of the next cubic curve. In the following the index corresponds to the beginning of the input interval where the output interpolated sample is computed. Let’s note the coefficients of the cubic curve of which the central interval is used , , , , the coefficients of the previous cubic curve , , , and the coefficients of the next cubic curve , , , . This gives 3 possible values for a given time instant ,

(1931e)

(1931f)

(1931g)

The interpolated output value for a given instant is computed as the weighted mean value of these 3 possible interpolated values:

(1936h)

The weights used are same for each interpolated value, ===1/3. To reduce the complexity the values of x/3, x2/3, x3/3, (x-1)/3, (x-1)2/3, (x-1)3/3, (x+1) /3, (x+1)2/3, and (x+1)3/3, are tabulated for all possible values of needed for the interpolations. So the weighting by 1/3 is integrated in these tables, only the coefficients , and are needed with a multiplication by 1/3 when the output value is computed. For example to upsample from 12.8 kHz to 32 kHz the required values of are 0.2, 0.4, 0.6 and 0.8.

The last 2intervals cannot be covered by 3 cubic curves as future samples are not available to compute all curves. Here simplified interpolation is used. For the last but one input interval the central interval of the last possible cubic curve is used to compute the interpolated signal:

(1937i)

and for the last input interval the interval [1,2] of the same last cubic curve is used to compute the interpolated signal

(1938j)

In case of subsampling, the output samples after the last input sample cannot be interpolated, that causes a small delay of up to 3 output samples.

##### 6.3.3.2.2 HQ MDCT decoding with a modified synthesis window

HQ MDCT decoding in the transition frame is identical to clause 6.2.3, except the MDCT synthesis window is modified and the bit budget in the current frame is decreased as described in clause 6.3.3.2.1.

The modified MDCT window is designed to avoid aliasing in the first part of the frame. Its shape also allows cross-fading between the synthesis from constrained CELP and simplified BWE and the synthesis from HQ MDCT.

##### 6.3.3.2.3 Cross-fading

As shown in Figure 112b, the CELP and HQ MDCT decoded signals are overlapping; the length of this overlapping region is 1,825 ms. The HQ MDCT synthesis is already windowed by the modified MDCT window at the decoder. The CELP decoded signal is windowed by the complementary window and added to the HQ MDCT output in the overlap region.

### 6.3.4 Internal sampling rate switching

When changing the internal sampling rate in CELP or MDCT-based TCX, a number of memory and buffer updates needs to be done. These are described in subsequent sub-clauses.

#### 6.3.4.1 Reset of LPC memory

Same as subclause 5.4.4.1.

#### 6.3.4.2 Conversion of LP filter between 12.8 and 16 kHz internal sampling rates

Same as subclause 5.4.4.2.

#### 6.3.4.3 Extrapolation of LP filter

Same as subclause 5.4.4.3.

#### 6.3.4.4 Update of CELP synthesis memories

Same as subclause 5.4.4.7.

#### 6.3.4.5 Update of CELP decoded past signal

When switching from CELP coding mode to MDCT-base TCX coding mode, the CELP decoded past signal with is needed as described in subclause 6.3.3.1. The past signal is resampled with the method described in subclause 5.4.4.4 in case of internal sampling rate switching before proceeding as described in subclause 6.3.3.1.

#### 6.3.4.6 Post-processing

In case of sampling rate switching, the post-processing module described in clause 6.1.4 has a specific behavior during the transition.

#### 6.3.4.6.1 Adaptive post-filtering

The memories of the adaptive post-filtering are resampled with the linear interpolation described in subclause 5.4.4.4 in case of sampling rate switching and in case post-filtering was applied in the previous frame.

If the post-filtering was not employed in the previous frame, the adaptive post-filter is not employed in the first frame after the transition. In such a case, only the memories of the post-filter are populated and updated for the next frame.

Moreover, in case the adaptive post-filter was activated in the previous frame and is switched off in the current frame, a smoothing mechanism is applied for avoiding any discontinuities. It is achieved by computing the zero impulse response of the post-filter states and adding it to the zero memory response of the current decoded frame. First, the first subframe of the current decoded frame is filtered by the corresponding set of LPC analysis filter and using as memory the previously decoded frame samples before being post-processed. The past non post-processed samples are eventually resampled as described in clause 5.4.4.4 in case of internal sampling rate switching. The residual is re-synthesized with using this time as memory the past decoded frame sampled computing after being post-processed. As stated above, the past non post-processed samples are resampled in case of internal sampling rate switching. The rest the current decoded frame is not processed further.

#### 6.3.4.6.2 Bass post filter

At 9.6, 16.4 and 24.4 kbps, the past memory of signal defined in clause 6.1.4.2 is reset in case of sampling rate switching. That means that the Bass post-filter has no effect during the transition.

#### 6.3.4.7 CLDFB

In case of internal sampling rate switching, the states of the analysis CLDFB needs to be resampled for both the decoded signal coming from either CELP or MDCT-based TCX decoded module, and also for the error signal provided by the Bass post-filter as defined in clause 6.1.4.2.

The resampling of the two set of states is performed with the help of the linear interpolation described in subclause 5.4.4.4.

### 6.3.5 EVS primary modes and AMR-WB IO

#### 6.3.5.1 Switching from primary modes to AMR-WB IO

In addition to processing described Subclause 5.4.5.1, the following is done:

* Reset the unvoiced/audio signal improvement memories
* Reset AMR-WB BWE memories
* bass post-filter is not employed in the first AMR-WB IO frame
* formant post-filter is not employed in the first AMR-WB IO frame

#### 6.3.5.2 Switching from AMR-WB IO mode to primary modes

Same as Subclause 5.4.5.2.

### 6.3.6 Rate switching

When the bit-rate is changing, the different coding tools are reconfigured at the beginning of the frame. The different bit-rate dependent setups of each tool are described in each corresponding clause. Rate switching doesn’t require any specific handling, except in the following scenarios.

#### 6.3.6.1 Rate switching along with internal sampling rate switching

In case the internal sampling rate changes when switching the bit-rate, the processing described in clause 6.3.4 is performed at first.

#### 6.3.6.2 Rate switching along with coding mode switching

In case the internal sampling rate changes when switching the bit-rate, the processing described in clause 6.3.3 is performed. If the internal sampling rate is also changing, the processing of clause 6.3.4 is performed beforehand.

#### 6.3.6.3 Adaptive post-filter reset and smoothing

The adaptive post-filter can be reset and it effect smoothed when it is switched on or off from frame to frame, respectively. The procedure is described in suclause 6.3.4.6.1, where the buffer resampling is performed only in case of internal sampling rate switching.

### 6.3.7 Bandwidth switching

When rate switching happens and the bandwidth of the output signal is changed from WB to SWB and from SWB to WB, bandwidth switching post-processing is performed in order to improve the perceptual quality for the end users. The smoothing is applied for switching from WB to SWB and the blind bandwidth extension is employed for switching from SWB to WB.

#### 6.3.7.1 Bandwidth switching detector

Firstly, bandwidth switching detector is employed to detect if there is bandwidth switching or not.

Initialize the counter of bandwidth switching from WB to SWB , and initialize the counter of bandwidth switching from SWB to WB .

The counter is calculated as follows:

1. If the counter, the counter is reset to 0;
2. else if the output bandwidth of the current frame is SWB, the total bit rate of the current frame is large than 9.6kbps, and the bandwidth of the previous frame is WB, the total bit rate of the previous frame is not large than 9.6kbps, the counter is incremented 1;
3. else if the counter, the countersand will be reset as follows:

(1939)

(1940)

The counter is calculated as follows:

1. If the counter, the counter is reset to 0;
2. else if the conditions are all satisfied, the counter is incremented 1
3. else if the counter, the countersand will be reset as follows:

(1941)

(1942)

Finally, the counterindicates the switching from super wideband to wideband; and the counter indicates the switching from wideband to super wideband.

#### 6.3.7.2 Super wideband switching to wideband

TBE mode, multi-mode FD BWE or MDCT based scheme will be employed to generate the SHB signal when switching to wideband.

If the following conditions

(1943)

are satisfied, TBE mode is applied to reconstruct the SHB signal.

Otherwise, if the conditions are satisfied, multi-mode FD BWE algorithm is applied;

If the core is MDCT coding, the MDCT coefficients of the upper band are predicted.

##### 6.3.7.2.1 TBE mode

The following steps are performed when TBE mode is used to generate the SHB signal for wideband output:

1. Estimate the high band LSF, gain shape according to the corresponding parameters of the previous frame or by looking for the pre-determined tables
2. Reconstruct an initial SHB signal according to the TBE algorithm described in subclause 5.2.6.1.
3. Predict a global gain of the initial SHB signal according to the spectral tilt parameter of the current frame and the correlation of the low frequency signal between the current frame and the previous frame
4. Modify the initial SHB signal by the predicted global gain to obtain a final SHB signal
5. Finally, the final SHB signal and low frequency signal are combined to obtain the output signal.

The spectral tilt parameter can be calculated as described by the algorithm described in equations (800) and (801), and the correlation of the low frequency signals of the current frame and the previous frame can be the energy ratio between the current frame and the previous frame.

In detail, the algorithm of predicting the global gain is described as follows:

1. Classify the signal of the current frame to fricative signal or non- fricative signal according to the spectral tilt parameter and the correlation of the low frequency signal between the current frame and the previous frame:   
   When the spectral tilt parameter of the current frame is larger than 5 and the FEC class of the low frequency signal is UNVOICED\_CLAS, or the spectral tilt parameter is larger than 10. And if the signal of the previous frame is non-fricative signal, and the correlation parameter is larger than a threshold, or if the signal of the previous frame is fricative signal and the correlation parameter is less than a threshold, the current frame is classified as fricative signal . Otherwise, the current frame is classified as fricative signal .
2. For non-fricative signal, the spectral tilt parameter is limited to the range [0.5, 1.0]. For fricative signal, the spectral tilt parameter is limited to not larger than 8. The limited spectral tilt parameter is used as the global gain of the SHB signal.

For some speacial cases: If the energy of the SHB signal(calculated by the global gain and ) is larger than the energy of the signal with the frequency range in [3200, 6400] , the global gain of the SHB signal is calculated as follows:

(1944)

where is the energy of the initial SHB signal.

If the energy of the SHB signal is less than 0.05 times of the energy of the signal with the frequency range in [3200, 6400] , the global gain of the SHB signal is calculated as follows:

(1938)

For non-fricative signal, the global gain is multiplied by 2; and for fricative signal, the global gain is multiplied by 8. And then the global gain of the SHB signal will be smoothed further as follows:

* If the signal of the current frame and the previous frame are both fricative signal and

(1939)

* else if the energy ratio of the low frequency signal between the current frame and the previous frame is in the range of [0.5, 2], and the modes of the signal of the current frame and the previous frame are both fricative or are both non-fricative

(1945)

* Otherwise

(1946)

where is the energy ratio between the final SHB signal of the previous frame and the initial SHB signal of the current frame.

At the end, fade out the global gain of the SHB signal frame by frame as follows:

(1947)

##### 6.3.7.2.2 Multi-mode FD BWE mode

Predict the SHB signal of the current frame, and weight the SHB signal of the current frame and the previous frame to obtain the final SHB signal of the current frame. Then the SHB signal and the low frequency signal are combined to obtain the output signal.

The SHB signal of the current frame is generated as follows:

1) Predict the fine structure of the SHB signal of the current frame as described in subclause 6.1.5.2.1.5.

2) Predict the envelope of the SHB signal of the current frame, and weight the envelopes of the current frame and the previous frame to obtain the final envelope of the current frame.

3) Reconstruct the SHB signal of the current frame by the predicted fine structure and the weighted envelope.

In detail, the algorithm of predicting and weighting the envelopes is described as follows:

1. The 224 MDCT coefficients in the 800-6400 Hz frequency range are split into 7 sharpness bands (32 coefficients per band). Calculate the peak of the magnitudes and the average magnitude in each sharpness band.
2. The low frequency signal is classified into NORMAL or HARMONIC according to the ratios of the peak of the magnitudes and the average magnitude.
3. Predict the initial envelope of the SHB signal according to the average magnitudes, the maximum value of the average magnitudes, the minimum value of the average magnitudes and the tilt parameter of the low frequency signal.
4. An energy ratio between the same parts of the low frequency signal of the current frame and the previous frame is introduced to control the weighted value of the predicted envelope. This ratio reflects the correlation of the current frame and the previous frame.
5. Factor is the weighting factor for the spectral envelope of the current frame, and is the one for the previous frame. is initialized to 0.1. Note that is reset to 0.1 if equals to zero.
6. If , the predicted spectral envelope is weighted according to the energy ratio

(1948)

and , when

where is the envelope of the previous frame. Factor is incremented by 0.05 and saved for next frame.

Otherwise, the predicted envelope is weighted as follows:

(1949)

Then, fade out the predicted envelope of the SHB signal frame by frame as follows:

(1950)

##### 6.3.7.2.3 MDCT core

For low bit rate core: predict the envelopes of the upper band from the 2 decoded highest frequency envelopes and the average envelope of all the decoded envelopes, and reconstruct he normalized coefficients by random noise. And then the MDCT coefficients of the upper band are reconstructed by the envelopes and the normalized coefficients.

For high bit rate core: for transient mode, predict the envelopes by the 20 decoded highest frequency coefficients; and for non-transient mode, predict the envelopes by the 2 decoded highest frequency envelopes. The normalized coefficients are predicted by random noise or by weighting the random noise and the decoded low frequency coefficients. And then the MDCT coefficients of the upper band are reconstructed by the envelopes and the normalized coefficients.

#### 6.3.7.3 Wideband switching to super wideband

In the case of switching from wideband to super wideband for multi-mode FD BWE or the MDCT core the coefficients in the frequency domain are faded using the factor. When operating with TBE, the global gain in the temporal domain is faded using the same factor .

## 6.4 De-emphasis

The reverse of the filtering process described in subclause 5.1.4 is performed at the output of the decoder.

(1951)

where is the de-emphasis factor which is set according to the sample rate and core type as described in subclause 5.1.4.

## 6.5 Resampling to the output sampling frequency

The CELP and MDCT decoder output signals are sample rate converted, depending on the selected output sample rate, i.e. 8 kHz, 16 kHz, 32 kHz or 48 kHz. The CELP signal is resampled by the CLDFB, while the MDCT part is resampled by its frequency to time transformation.

## 6.6 Decoding of frame erasure concealment side information

The parameters described in subclause 5.5 are decoded to aid in the error concealment procedure.

These parameters are;

* Signal classification parameter
* Energy information
* Phase control information
* Pich lag information
* Spectral envelope diffuser

For more information how these parameters are employed, refer to subclause 5.3.3 of [6].

## 6.7 Decoding in DTX/CNG operation

### 6.7.1 Overview

When the decoder is in the CNG operation, the first bit in the SID frame is decoded to point to the CNG mode in which the decoder will be operating. In the LP-CNG mode, excitation(s) and synthesis filter(s) are calculated from the decoded CN parameters, and the comfort noise is synthesized by a LP synthesis approach.

In the FD-CNG mode, a spectral envelope is generated with the help of the energy level information decoded from the SID data. The spectral envelope is refined by a noise estimator running during active frames. The resulting envelope determines the actual comfort noise which is rendered inside different frequency domains, such as MDCT, FFT or CLDFB and finally transformed to time-domain.

Subsequent subclauses describe the respect LP-CNG decoding and FD-CNG decoding in details.

### 6.7.2 Decoding for LP-CNG

#### 6.7.2.1 LP-CNG decoding Overview

When the decoder is in the LP-CNG operation, a procedure to synthesize a comfort noise signal is applied.

For each received SID frame, the one bit indicating the bandwidth type of the SID frame is first decoded. WB SID frame is received if the bandwidth bit equals “0”, otherwise the SWB SID frame is received. The LP-CNG decoder only operates in WB mode if no SWB SID frame has been received, in which case the comfort noise is only generated for low-band. Otherwise, the LP-CNG decoder will switch to SWB mode upon the receiving of the SWB SID frame. Since the transmission of high-band CN parameter is not synchronized with the transmission of the low-band CN parameters, WB SID frames can be received even the LP-CNG decoder operates in SWB mode. In which case, the energy parameter for high-band CN synthesis is extrapolated from the low-band CN synthesis signal. The low-band excitation energy is decoded from each LP-CNG SID frame based on which a smoothed low-band excitation energy used for low-band CNG synthesis is computed, as described in subclause 6.7.2.1.3. The low-band LSF vector is decoded from each LP-CNG SID frame then converted to LSP vector based on which a smoothed LSP vector is computed then converted to LP coefficients to obtain the low-band CNG synthesis filter, as described in subclause 6.7.2.1.4. If WB LP-CNG SID frame is received, the residual spectral envelope is decoded based on which a smoothed residual spectral envelope is computed, as described in subclause 6.7.2.1.5. A random excitation signal is generated from the smoothed low-band excitation energy which is combined with a second excitation signal generated from the smoothed residual spectral envelope to form the final excitation signal for the low-band CNG synthesis, as described in subclause 6.7.2.1.5. Low-band comfort noise is synthesized by filtering the low-band final excitation signal through the low-band CNG synthesis filter, as described in subclause 6.7.2.1.6.

In subclause 6.7.2.1.7, high-band decoding and synthesis is described if the decoder is operating in SWB mode. When SWB LP-CNG SID frame is received, the high-band energy of the frame is decoded from the SID frame. For other types of received frames, that is the WB LP-CNG SID frames and the NO\_DATA frames, the high-band energy of the frame is generated locally at the decoder by extrapolating from the smoothed low-band energy of the frame which is obtained from the low-band CNG synthesis together with a high-band to low-band energy ratio calculated at the last received SWB LP-CNG SID frame. The high-band energy of the frame is further smoothed in each frame to be used for final high-band CNG synthesis. For each CN frame, the high-band LSF spectrum used to obtain the high-band CNG synthesis filter for each CN frame is interpolated from the LSF spectrum of the hangover frames. The high-band comfort noise is synthesized for each CN frame by filtering a random excitation through the high-band CNG synthesis filter, then scaled to the level corresponding to the smoothed high-band energy. The scaled high-band synthesis signal is finally spectral flipped to the bandwidth from 12.8 kHz to 14.4kHz, as described in subclause 6.1.5.1.12. The resulting spectral flipped high-band synthesis signal is added to the low-band synthesis signal so to form the final SWB comfort noise synthesis signal.

##### 6.7.2.1.1 CNG parameter updates in active periods

During actively encoded periods without comfort noise parameters, four buffers of the fixed predetermined size are kept updated with the current actively encoded frame’s LSPs, an LSP domain flag memory, the frame’s excitation energy(in the LP-residual domain) and the current low frequency spectral envelope of the excitation as:

(1947)

(1948)

(1949)

(1950)

where andare, respectively, the real and the imaginary parts of the-th frequency bin as outputted by the FFT of the LP excitation signal, = 256 is the size of FFT analysis. The attenuation factor is given by

(1950a)

where is determined by the latest bitrate used for actively encoded frames , not including the current frame, according to Table 172a.

Table 172a: Attenuation factor selection

|  |  |
| --- | --- |
| Latest active bitrate [kbps] |  |
|  | 1.7938412 |
|  | 1.3952098 |
|  | 1.0962363 |
|  | 0.9965784 |
|  | 0.9965784 |

These buffers are implemented as circular FIFO (First in First Out) buffers of sizeto save complexity.

##### 6.7.2.1.2 DTX-hangover based parameter analysis in LP-CNG mode

To provide smoother sounding comfort noise synthesis in transitions from active to inactive (CNG) coding, the 3 bit parameter isdecoded from the bit stream and used as the indicator for determining the initial subset of the sized buffers with () parameters from the last active frames. The most recent number of frames of the stored parameters (), are used for an additional comfort noise parameter analysis in the very first SID frame after an active speech segment.

Before copying the most recent vectors to the CNG-analysis buffer , the past LSP’s which were analysed with a different sampling frequency than the current SID frame’s sampling frequency is converted to the current frames sampling frequency according to the information available in the flag vector . The most recent values are copied into the analysis vector and the most recent values are copied into the analysis vector.

An age weighted average energy of the entries which are less than 103% of the most recent energy value and greater than 70 % of the most recent energy value, is computed as, further the number of entries in used for this average calculation is stored as. The age weights for the computation are:

(1951)

Further the vectors corresponding in time to the past residual energies in used for are saved in a buffer. The buffer is converted into the LSF domain in buffer .

Two outlier vector indices [] in the buffer among the vector entries are found, by analysing the maximum average LSF deviation to a uniform LSF spectrum. An average LSP-vector is calculated, by computing the LSP-average with exclusion of zero, one or two of the found outlier vectors, depending on the value of.

The sum of the LSP-deviations with respect to the received SID-frame’s quantized LSPs , is computed as:

(1952)

Further the maximum individual distortion contribution in the summation above is saved as.   
  
If there were no past CN-parameters to analyse or an residual energy step was detected, the received vector is used as the final vector right away, on the other hand if there were some past active SAD hangover frames to analyse and there was no energy step detected , the and are now used to control the vector update over , of the final CNG LSPs using the average LSP-vector as follows:

(1953)

The energy step is detected if it is the first CN frame after an active frame and the energy quantization index decoded from the current SID frame is greater than the previous energy quantization index by more than 1, where . Additionally, if there were past CN-parameters, an energy step is detected if the most recent energy value in is more than four times larger than the smoothed quantized excitation energy . Further the vectors that originate from active or WB SID frames among the vectors corresponding in time to the past residual energies in used for are saved in a buffer. The average envelope of is computed and from which two times of the smoothed residual spectral envelope of the previous frame, , calculated in equation (1953) is subtracted. The resulting average envelope is used to initialize the smoothed residual spectral envelope if there is no energy step detected.

When a SID frame is received and there was no energy step detected, first the received and decoded LSP vector is added to the CNG-analysis buffer in a FIFO manner for a buffer size of up to, and secondly the decoded residual energy value in the SID frame is added to the CNG analysis buffer in a FIFO manner for a buffer size of up to, then thirdly, if applicable (depending on if the SID frame is of WB type or not), the decoded low frequency envelope of the excitation from the SID frame is added to the CNG analysis buffer for a buffer size of up to.

During actively encoded periods, i.e. not including SID frames, the currently least recent buffer element (firstly added) in the buffer  and the corresponding element in the buffer  are excluded from the buffers with a period of number of consecutive actively encoded frames given by the decrement factor . As circular FIFO buffers are implemented the elements do not actually have to be deleted but the variable  representing the number of valid buffer elements, i.e. elements used for determination of  and , is given by:

 (1953a)

where  is the number of valid buffer elements in the very beginning of the actively encoded period,  is a non-negative integer and  is a counter of consecutive actively encoded frames. The variable  does together with a pointer to the most recently added buffer element determine the valid buffer elements.

##### 6.7.2.1.3 LP-CNG low-band energy decoding

The quantized low-band excitation energy in logarithmic domain is decoded from each LP-CNG SID frame and converted to linear domain using the procedure described in subclause 5.6.2.1.5. The resulting linear domain low-band excitation energy is used to obtain the smoothed low-band excitation energy used for low-band CNG synthesis in the same way as described in subclause 5.6.2.1.6.

##### 6.7.2.1.4 LP-CNG low-band filter parameters decoding

The quantized LSF vector is found in the same way as described in subclause 6.1.1.1.1. For the two stage quantizer there are two indexes that define the LSF vector. The index of the first stage codevector is retrieved and the codevector components are obtained from the 16-dimensional codebook of 16 codevectors. The second stage index is interpreted like in subclause 6.1.1.1.1. and the corresponding multiple scale lattice codevector is obtained. If the codevector index from the first stage has one of the values 0, 1, 2, 3, 7, 9, 12, 13, 14, 15 the permutations specified in subclause 5.6.2.1.3 are applied to the decoded codevector. The resulting codevector is added to the codevector obtained in the first stage and the result corresponds to the decoded LSF vector. The sampling frequency of the LP-CNG frame can be determined by checking the value of the highest order LSF coefficient (last coefficient). If the last decoded LSF coefficient is larger than 6350 the decoded frame has sampling rate of 16 kHz, otherwise it is sampled at 12.8kHz and contains either NB or WB LSF data. The smoothed LP synthesis filter, , is then obtained in the same way as described in subclause 5.6.2.1.4.

##### 6.7.2.1.5 LP-CNG low-band excitation generation

The low-band excitation signal used for CNG synthesis is generated by combining a random excitation and an excitation representing the low frequency spectral details of the excitation signal.

The random excitation is generated for each subframe using a random integer generator, the seed of which is updated by

(1954)

where is the seed value, initially set to 21845, and short[.] limits the value to the interval [–32767; 32768]. The generated random sequence, denoted as , is scaled for each subframe by

(1955)

where denotes the length of subframe, is the smoothed quantized excitation energy, as described in subclause 6.7.2.1.3, with some random variation between subframes added. The random variation is added to the smoothed quantized excitation energy by

(1956)

where is a random integer number generated for each subframe using the same equation (1957) with the initial value of 21845. The purpose of which is to better model the variance of background noise during inactive signal periods. The scaled random sequence in each subframe, , is concatenated to form the random excitation signal for the whole frame, , where is the frame length.

The excitation representing the low frequency spectral details of the excitation signal is generated from the quantized residual spectral envelope. The quantized residual spectral envelope is recovered from each WB SID frame in the inverse way as described in subclause 5.6.2.1.6 that

(1958)

where is the quantized residual spectral envelope, is the entry of the residual spectral envelope codebook found with the index decoded from the SID frame, is the quantized total excitation energy calculated using the similar equation in subclause 5.6.2.1.6,. A smoothed residual spectral envelope is updated at each CN frame by through an AR filtering

(1959)

where denotes the smoothed residual spectral envelope from the previous frame. If the current frame is of NO\_DATA type, the from the last received SID frame is used. The FFT spectrum of the random excitation, , generated in equation (1960) is computed and based on this the low frequency spectral envelope, , corresponding to the one transmitted in the SID frame is calculated in a similar manner. The difference envelope between the smoothed spectral envelope and the random excitation envelope is calculated

(1961)

where is the smoothed quantized excitation energy obtained in subclause 6.7.2.1.3, 2 times of is compensated to the before the difference envelope is calculated. Slight random variation is added to the difference envelope.

(1962)

where is a series of random integer numbers generated for each envelope band using the same equation (1963) with the initial value of 21845. A series of 256-point random-phase FFT coefficients are generated where its low frequency spectral envelope is made equal to the difference envelope and the coefficients corresponding to other frequencies are set to 0. An IFFT is performed to the above FFT coefficients and a 256-point time domain sequence is outputted. is re-sampled to 320 points if operating in 16kHz core. is scaled for each subframe in a similar way to equation (1964) that

(1965)

where is the scaled with random variation added, denotes the length of subframe, is a random integer number generated for each subframe using the same random generator as used in equation (1966), is the average energy of calculated as

(1967)

where is the frame length. Energy increasing is not allowed if the current frame is the first SID frame after an active burst in which case, for subframe with , is limited to . The is the excitation representing the low frequency spectral details of the excitation signal whichis attenuated and combined with the earlier calculated random excitation

(1968)

where is the frame length, is the combined excitation signal. The combined excitation is scaled if its average energy is higher than the smoothed quantized excitation energy obtained in subclause 6.7.2.1.3.

(1969)

where is the frame length, is the scaled combined excitation which is the final excitation signal for low-band CNG synthesis.

##### 6.7.2.1.6 LP-CNG low-band synthesis

The low-band comfort noise is synthesized by filtering the scaled combined excitation signal, , obtained in previous subclause 6.7.2.1.5 through the smoothed LP synthesis filter, , obtained in subclause 6.7.2.1.4.

##### 6.7.2.1.7 LP-CNG high-band decoding and synthesis

To enable high perceptual quality in the inactive portions of speech on the decoder side, during SWB mode operation of the codec, a high band comfort noise synthesis (SHB-CNG) (12.8 - 14.4 kHz) is added to the low bandwidth (0-12.8 kHz) LP-CNG synthesis output. This also helps to ensure smooth transitions between active and inactive speech.

However, this is being done without transmitting any extra parameters from the encoder to decoder to model the high-band spectral characteristics of the inactive frames. Instead, to model the high band spectrum (12.8 - 14.4 kHz) of the comfort noise, the high band LSF parameters of the active speech frames preceding the current inactive frames are used after interpolation as described below. The hangover setting in the SAD algorithm ensures the active speech segments used for the spectral characteristics estimation of the inactive frames, sufficiently capture the background noise characteristics without significant impact from the talk spurt.

The quantized LSF vectors of order 10 corresponding to active speech high band (subclauses 5.2.4.1.3.1 and 6.1.5.1.3.1) received at the decoder are buffered up to two past active frames (N-1) and (N), denoted by and where N+M is the current inactive frame. Using these, the LSF vector corresponding to SHB-CNG of (N+M) th inactive frame is interpolated as

(1970)

where interpolation factor *T* is computed as

(1971)

using the number of inactive frames M leading up to the current inactive frame (N+M) since the last active frame N.

This interpolated LSF vector is then converted to LPC coefficients and used as the coefficients of LP synthesis filter to generate a synthesized signal. The energy of the high-band signal is obtained for each CN frame by either directly decoding from the SID frame if the SID frame is a SWB SID frame or by extrapolating for other received frame types. If SWB SID frame is received, the high-band energy of the frame which is the quantized high-band log average energy, , is recovered by

(1972)

where is the high-band energy index decoded from the SWB SID frame. If is 0, is set to -15 for a lower noise floor. If WB SID frame or NO\_DATA frame is received, the high-band energy of the frame is generated locally at the decoder by extrapolating from the smoothed low-band energy of the frame together with the high-band to low-band energy ratio at the last received SWB LP-CNG SID frame. The smoothed low-band energy of the frame is a weighted average of the low-band energy of the current frame and the smoothed low-band energy of the previous frame. The low-band energy of the current frame which is the log average energy of the low-band signal is calculated from the low-band synthesis signal

(1973)

where is the low-band synthesis signal as obtained in subclause 6.7.2.1.6, = 640 is the length of the low-band synthesis signal. If the low-band energy of the current frame is deviating from the smoothed low-band energy of the previous frame by more than 12 dB, a step update flag is set to 1 indicating the permission of step update, otherwise is set to 0. If the flag is set to 1, the smoothed low-band energy at the current frame, , is set to the current frame’s low-band energy . Otherwise, if the flag is set to 0, the smoothed low-band energy is updated at the current frame as

(1969)

where denotes the smoothed low-band energy of the previous frame. The high-band energy of the frame,, for received WB SID or NO\_DATA frame is thus extrapolated as the sum of and, where denotes the high-band to low-band energy ratio at the last received SWB SID frame *i* frames ago. The high-band energy of the frame is then smoothed for final use according to

(1970)

where is the smoothed high-band energy of the current frame, denotes the smoothed high-band energy of the previous frame, is the forgetting factor which is set to 0 if is set to 1 or the current frame is the first frame after an active burst, otherwise is set to 0.75. The high-band comfort noise is synthesized by filtering a 320-point white noise excitation signal through the LP synthesis filter derived earlier in this subclause. The synthesized comfort noise signal is then level adjusted to match the calculated smoothed high-band energy . A smoothing period is setup for the first 5 frames after an active burst of more than 3 frames and if the core technology used in the last active frame is not HQ-core. Within the smoothing period, the synthesized comfort noise is not level adjusted to the calculated smoothed high-band energy , but to an interpolated energy between and the high-band log average energy calculated at the last active frame. The interpolated energy is calculated as

(1971)

where denotes the interpolated high-band energy of the -th CN frame in the smoothing period, denotes the high-band log average energy of the last active frame, denotes the sine function. Finally, the level adjusted synthesized high-band comfort noise is spectral flipped to the bandwidth from 12.8kHz to 14.4kHz as described in subclause 6.1.5.1.12. The resulting high-band synthesis signal is later added to the low-band synthesis signal to form the final SWB comfort noise synthesis signal.

#### 6.7.2.2 Memory update

When an inactive signal frame is processed, the following updates are performed:

– MA memory of the ISF quantizer is set to zero;

– AR memory of the ISF quantizer is set to mean values (UC mode, WB case);

– phase dispersion memory is set to zero;

– synthesis excitation spectrum tilt is set to zero;

– noise enhancer memory is set to zero;

– class of last received good frame for FEC is set to UNVOICED\_CLAS;

– floating point pitch for each subframe is set to the subframe length;

– the low-pass filtered pitch gain for FEC is set to zero;

– the filtered algebraic codebook gain for FEC is set to the square root of the smoothed quantized CNG excitation energy, , from subclause 6.7.2.1.3;

– the excitation buffer memory is updated;

– previous pitch gains are all set to zero;

– previous codebook gain is set to zero;

– active frame counter is set to zero;

– voicing factors used by the bandwidth extension are all set to 1;

– bass post-filter is tuned off;

– synthesis filter memories are updated.

### 6.7.3 Decoding for FD-CNG

In FD-CNG, the comfort noise is generated in the frequency domain. Based on the information provided by the SID frames, the amplitude of the random sequences can be individually computed in each band such that the spectrum of the generated comfort noise resembles the spectrum of the actual background noise in the input signal.

Unfortunately, the limited number of parameters transmitted in the SID frames allows only to reproduce the smooth spectral envelop of the background noise. Hence, it cannot capture the fine spectral structure of the noise. At the output of a DTX system, the discrepancy between the smooth spectrum of the reconstructed comfort noise and the spectrum of the actual background noise can become very audible at the transitions between active frames (involving regular coding and decoding of a noisy speech portion of the signal) and the comfort noise frames. Since the fine spectral structure of the background noise cannot be transmitted efficiently from the encoder to the decoder, it is highly desirable to recover this information directly at the decoder side. This can be carried out using a noise estimator.

Note that noise-only frames are considered as inactive frames in a DTX system. Therefore, the noise estimation in the decoder must operate during active phases only, i.e., on noisy speech contents. In FD-CNG, the decoder uses in fact the same noise estimation algorithm as in the encoder, but applying a significantly higher spectral resolution at the decoder than at the encoder.

#### 6.7.3.1 Decoding SID frames in FD-CNG

The decoded SID parameters describe the energy of the background noise in the spectral partitions defined in subclause 5.6.3.6. The first parameters capture the spectral energy of the noise in FFT bins covering the core bandwidth. The remainingparameters capture the spectral energy of the noise in CLDFB bands above the core bandwidth.

##### 6.7.3.1.1 SID parameters decoding

The SID parameters are decoded using MSVQ decoding and global gain adjustment.

Seven indices are decoded from the bitstream. The first six indices are used for MSVQ decoding. They correspond to the six stages of the MSVQ. The first index is encoded on 7 bits and the five next indices are encoded on 6 bits. The last index, encoded on 7 bits, is used for decoding the global gain.

The MSVQ decoder output is given by

, (1974)

where is the -th coefficient of the -th vector in the codebook of stage .

The decoded global gain is given by

. (1975)

The SID parameters are then obtained

. (1976)

Finally the last band parameter is adjusted in case the encoded last band size is different from the decoded last band size

##### 6.7.3.1.2 SID parameters interpolation

The SID parameters are interpolated using linear interpolation in the log domain. The interpolation is carried out separately for the FFT partitions () and CLDFB partitions ().

, (1977)

where

(1978)

denotes the centre bin in each spectral partition, and

(1979)

is the multiplicative increment.

##### 6.7.3.1.3 LPC estimation from the interpolated SID parameters

A set of LPC coefficients is estimated from the SID spectrum in order to update excitation and LPC related memories.

A noise floor is first added to the interpolated SID parameters and a pre-emphasis function is then applied in the frequency domain

, (1980)

with is the pre-emphasis factor (0.68 at 12.8 kHz and 0.72 at 16 kHz).

The noise estimates are then transformed using an inverse FFT, producing autocorrelation coefficients

. (1981)

Then the first autocorrelation coefficient is adjusted

. (1982)

And finally LPC coefficients are estimated from the autocorrelation coefficients using Levinson-Durbin (see subclause 5.1.9.4).

#### 6.7.3.2 Noise tracking during active frames in FD-CNG

At the decoder side, the noise estimator is applied at the output of the core decoder during active frames. To achieve a trade-off between spectral resolution and computational complexity, spectral energies are averaged among groups of spectral bands called partitions, just like in the encoder (see subclause 5.6.3.1). However, the size of each partition is significantly smaller in the decoder compared to the encoder, yielding thereby a finer quantization of the frequency axis in the decoder. Moreover, the decoder-side noise estimation operates solely in the FFT domain and covers only the core bandwidth. Hence FFT partitions are formed, but no CLDFB partitions.

##### 6.7.3.2.1 Spectral partition energies

The output of the core decoder is first transformed by an FFT of size, whererefers to the sampling rate of the core decoder output. Then partitions are formed as follows:

, (1983)

where is the FFT transform of the core output signal. The following table lists the number of partitions and their upper boundaries for the different FD-CNG configurations at the decoder, as a function of bandwidths and bit-rates.

Table 173: FD-CNG decoder parameters

|  |  |  |  |
| --- | --- | --- | --- |
|  | Bit-rates (kbps) |  | (Hz) |
| NB |  | 56 | 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1075, 1175, 1275, 1375, 1475, 1600, 1725, 1850, 2000, 2150, 2325, 2500, 2700, 2925, 3150, 3400, 3975 |
| WB/ SWB/ FB |  | 62 | 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1075, 1175, 1275, 1375, 1475, 1600, 1725, 1850, 2000, 2150, 2325, 2500, 2700, 2925, 3150, 3375, 3700, 4050, 4400, 4800, 5300, 5800, 6375 |
|  | 61 | 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 325, 350, 375, 400, 425, 450, 475, 500, 525, 550, 575, 600, 625, 650, 675, 700, 725, 750, 775, 800, 825, 850, 875, 900, 925, 950, 975, 1000, 1075, 1175, 1275, 1375, 1475, 1600, 1725, 1850, 2000, 2150, 2325, 2500, 2700, 2925, 3150, 3400, 3700, 4400, 5300, 6400, 7700, 7975 |

For each partition,corresponds to the frequency of the last band in the *i*-th partition. The indicesand of the first and last bands in each spectral partition can be derived as a function of the FFT size and the sampling rate of the core decoder as follows:

, (1984)

, (1985)

where is the frequency of the first band in the first spectral partition. Hence the FD-CNG generates some comfort noise above 50Hz only.

##### 6.7.3.2.2 FD-CNG noise estimation

In FD-CNG, encoder and decoder rely on the same noise estimator, except that the number of partitions differs. As in subclause 5.6.3.2, the input partition energies are first processed by a non-linear transform before applying the noise tracking algorithm on the inputs. The inverse transform is then used to recover the original dynamic range. In the sequel, the resulting decoder-side noise estimates are referred to as . They are used as shaping parameters in the next subclause.

##### 6.7.3.2.3 Noise shaping in FD-CNG

Note that the shaping parameterscomputed directly at the decoder differ from the SID parameters which are transmitted via SID frames. Both sets are computed from FFT partitions covering the core bandwidth, but the decoder benefits from a significantly higher number of spectral partitions, i.e., . In fact, the high-resolution noise estimates obtained at the decoder capture information about the fine spectral structure of the background noise. However, the decoder-side noise estimates cannot adapt to changes in the actual background noise during inactive phases. In contrast, the SID frames deliver new information about the spectral envelop at regular intervals during inactive phases. The FD-CNG therefore combines these two sources of information in an effort to reproduce the fine spectral structure captured from the background noise present during active phases, while updating only the spectral envelop of the comfort noise during inactive parts with the help of the SID information.

6.7.3.2.3.1 Conversion to a lower spectral resolution

Interpolation is first applied to the shaping parametersto obtain a full-resolution FFT power spectrum as follows:

, (1986)

where

(1987)

denote the center FFT bins in each spectral partition, and is the multiplicative increment for the interpolation. The above corresponds in fact to a linear interpolation in the log domain of the FFT shaping partitions.

The full-resolution spectrum is subsequently converted again to a lower resolution based on the SID spectral partitions (see subclause 5.6.3.6). The resulting noise energy spectrum exhibits therefore the same spectral resolution as the SID parameters. Hence, both sets are comparable and can be combined in the next subclause.

6.7.3.2.3.2 Combining SID and shaping parameters

Comparing the low-resolution noise estimatesandobtained from the encoder (via SID frames) and decoder, respectively, the full-resolution noise spectrum can now be scaled to yield a full-resolution noise power spectrum as follows:

, (1988)

whereand refer to the first and last FFT bin of the *i*-th SID partition (see subclause 5.6.3.6). The full-resolution noise power spectrumis recomputed for each active frame. It can be used to accurately adjust the level of comfort noise in each FFT bin during SID frames or zero frames, as shown in the next subclause.

#### 6.7.3.3 Noise generation for SID or zero frames in FD-CNG

##### 6.7.3.3.1 Update of the noise levels for FD-CNG

During the first non-active frame following an active frame, the low-resolution shaping parameters are recomputed from by averaging over the SID partitions as in subclause 5.6.3.1.1.

If an SID frame occurs while the noise estimator (subclause 6.7.3.2.2) is still in its initialization phase, the interpolated SID parameters (see subclause 6.7.3.1.2) are used as comfort noise levels.

If an SID frame occurs once the noise estimator left the initialization phase, the comfort noise levels are computed for FFT bins by combining the noise estimates from encoder and decoder as described in subclause 6.7.3.2.3.2, while the CLDFB levels are obtained directly by interpolating the SID parameters corresponding to the CLDFB partitions, i.e.,

, (1989)

whereare the SID parameters decoded from the SID frames,are computed during the first non-active frame following an active frame, as explained just above, andis the full-resolution noise spectrum obtained by interpolating the decoder-side noise estimates during active frames (see subclause 6.7.3.2.3.1).

##### 6.7.3.3.2 Comfort noise generation in the frequency domain

The FFT noise spectrum corresponding to the first parameters in the array can finally be used to generate some random Gaussian noise of zero mean and variance separately for the real and imaginary parts of the FFT coefficients.

The second part of the CNG spectrum, i.e., corresponds to the CLDFB noise levels for frequencies above the core bandwidth. For each CLDFB time slot, some random Gaussian noise of zero mean and variance is generated separately for the real and imaginary parts of the CLDFB coefficients corresponding to frequencies above the core bandwidth.

##### 6.7.3.3.3 Comfort noise generation in the time domain

The FFT coefficients obtained after comfort noise generation in the frequency domain are transformed by an inverse FFT, producing a CNG time-domain signal of length. This signal is then windowed using a sine-based window that can be defined as follow

, for . (1990)

, for . (1991)

, for . (1992)

, for . (1993)

, for . (1994)

An overlap-add method is finally applied on the current windowed CNG signal and the previous windowed CNG signal. The final FD-CNG frame corresponds to.

To avoid discontinuities at transitions from active frames to inactive frames, a cross-fading mechanism is employed. Several approaches are described in the following subclauses, depending on the bitrate and the previous encoding mode.

6.7.3.3.3.1 Transitions from MDCT to FD-CNG at 9.6kbps, 16.4kbps and 24.4kbps

At 9.6 kbps, 16.4 kbps and 24.4 kbps and when the previous frame was active and encoded with an MDCT-based coding mode, a cross-fading operation is performed using the MDCT window.

First, the left part of the FD-CNG frame is not windowed after the inverse FFT and there is no overlap-add operation with the previous (missing) FD-CNG frame.

Instead, the left part of the FD-CNG frame is windowed using the complementary version of the MDCT window used in the right part of the previous MDCT-based frame (see subclause 6.2.4). An overlap-add method is then applied on the MDCT-windowed FD-CNG frame and the previous MDCT-based frame.

6.7.3.3.3.2 Transitions from ACELP to FD-CNG at 9.6kbps, 16.4kbps and 24.4kbps

At 9.6 kbps, 16.4 kbps and 24.4 kbps and when the previous frame was active and encoded with ACELP, a cross-fading operation is performed using an extrapolation of the previous ACELP frame.

A random excitation with the same energy as the last half of the excitation of the previous ACELP frame is computed

(1995)

where is the total excitation of the previous ACELP frame (see subclause 6.1.1.2), is a random Gaussian noise with zero mean and is the length of the ACELP frame.

The random excitation is then filtered through the same LPC synthesis filter and de-emphasis filter as used in the last subframe of the previous ACELP frame (see subclause 6.1.3), producing the extrapolated ACELP signal.

Finally the extrapolated ACELP signal is windowed and the overlap-add method is applied as if the extrapolated ACELP signal was the previous FD-CNG frame.

6.7.3.3.3.3 Transitions from active to FD-CNG at bitrates<=8kbps and at 13.2kbps

At bitrates<=8 kbps and at 13.2 kbps, a FD-CNG frame is converted to a combination of an excitation signal and a set of LPC coefficients. It then becomes very similar to a LP-CNG frame, and can be further processed as such.

First, the left part of the FD-CNG frame is not windowed after the inverse FFT and there is no overlap-add operation with the previous (missing) FD-CNG frame.

The time-domain FD-CNG signal is then pre-emphasized and filter through the LP analysis filter (see subclause 6.7.3.1.3), producing an excitation signal.

The set of LPC coefficients associated with the excitation is. These LPC coefficients are then used to synthesize the final CNG signal using the excitation and the filter memories from the previous frame, as it is done in LP-CNG.

##### 6.7.3.3.4 FD-CNG decoder memory update

The LPC coefficients are converted to LSP and to LSF. The LPC, LSP and LSF are then used to update all LPC/LSP/LSF-related memories.

The FD-CNG time-domain output is used to update all signal domain memories.

The FD-CNG time-domain signal is pre-emphasized and the pre-emphasized signal is then used to update all pre-emphasized signal domain memories.

The pre-emphasized signal is filtered through the LP analysis filter and the obtained residual is then used to update all excitation domain memories.

All the other memories are updated similarly to LP-CNG (see subclause 6.7.2.2).