|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A black and white logo  Description automatically generated | INTERNATIONAL TELECOMMUNICATION UNION  **TELECOMMUNICATION STANDARDIZATION SECTOR**  STUDY PERIOD 2025-2028 | | **SG21-C0020** | | |
| **STUDY GROUP 21** | | |
| **Original: English** | | |
| **Question(s):** | | 6/21 | | Geneva, 13-24 January 2025 | |
| **CONTRIBUTION** | | | | | |
| **Source:** | | Fraunhofer HHI, Dolby Laboratories | | | |
| **Title:** | | H.BWC draft algorithm description | | | |
| **Contact:** | | Jonathan Pfaff  Fraunhofer HHI Germany | | | Tel:  E-mail: jonathan.pfaff@hhi.fraunhofer.de |
| **Contact:** | | Christof Fersch Dolby Laboratories United States | | | Tel:  E-mail: christof.fersch@dolby.com |

|  |  |
| --- | --- |
| **Abstract:** | This document contains the draft algorithm description for H:BWC |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Introduction

This document contains an algorithm description of a first test model for H.BWC. The first version of this document treats the low-level part of the test model only.

# Description of the codec

The present section describes the codec for the coding of a single channel group with M channels and N samples per channel. The corresponding sample values shall be denoted by x[i][j], where and For notational simplicity, it is assumed that the channel group is coded as an independent frame. For a dependent frame, the architecture is the same but reference samples of previously coded frames in bitstream order might as well be invoked in the prediction process as described above.

## Block partitioning and sample processing order

The sample values are partitioned into a sequence of blocks denoted as . Each is described by the position of its first sample, denoted by , and by its length in sample direction, denoted by . The length is always an integral power of two. The block is defined as

Furthermore, adjacent blocks and contain adjacent samples. More precisely, one has

(1)

The blocks are sequentially coded. Thus, one starts with coding and then codes the block after having coded the block until one has reached the final position in the channel group.

For coding a block , one partitions it into its channel-wise sub-blocks with . The block is defined as

The block is coded by sequentially coding the blocks . This means that for the coding of , one starts with coding and, until , codes the block after having coded the block .

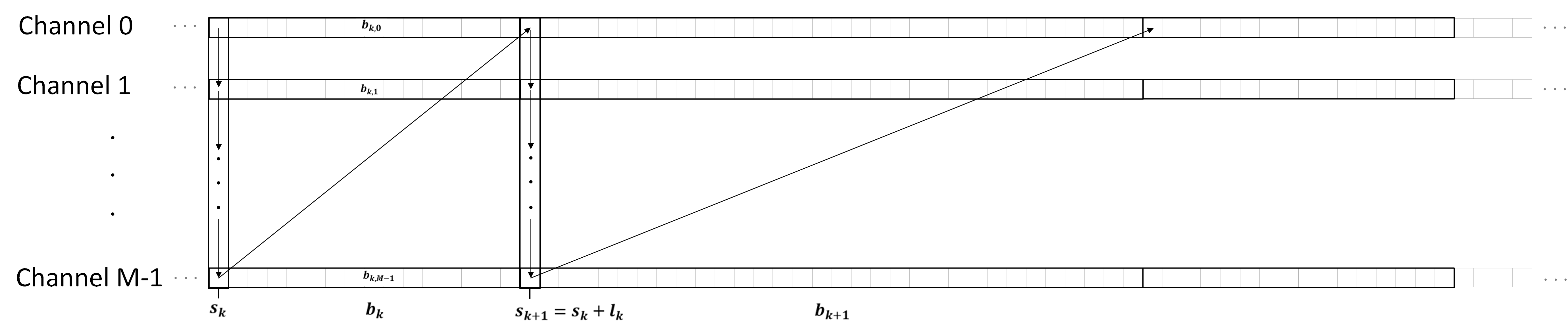


Figure 1: Illustration of the block partitioning and sample processing order

The value of a given block length is derived from the syntax element block\_split\_log2 and the variable Log2MaxBlockSize from the independent frame coding tool parameter set according to

).

## Switching between Least Mean Squares and Linear Predictive Coding based signal reconstruction, and predictive transform coding

For each block , two coding modes are supported. The switching between these coding modes is determined by the syntax element lms\_lpc\_block\_mode\_flag.

If lms\_lpc\_block\_mode\_flag is equal to 1, the reconstructed samples are computed by the process of Section 2.3.

## LMS- and LPC – based signal reconstruction

When lms\_lpc\_block\_mode\_flag is equal to 1, independent, non-overlapping frames of the input signal channels are processed at block size . *L* can be chosen as any power of two from 32 to 2048 samples. The processing is based on an integer DCT (see Section 2.3.5) in combination with linear prediction and quantization in the transform domain, and a straightforward entropy codec based on combination of multi-dimensional Huffman and Golomb-Rice coding (see Section 2.3.4). It is a single harmonized and conceptually simple structure that supports both lossless and lossy coding, where for the lossless operation both DCT-domain (Section 2.3.2) and time-domain coding (Section 2.3.3) can be applied.

Figure 1 and Figure 2 present the LMS-based encoding and decoding processes, respectively, when lms\_lpc\_block\_mode\_flag = 1 and IsLossy = 1.

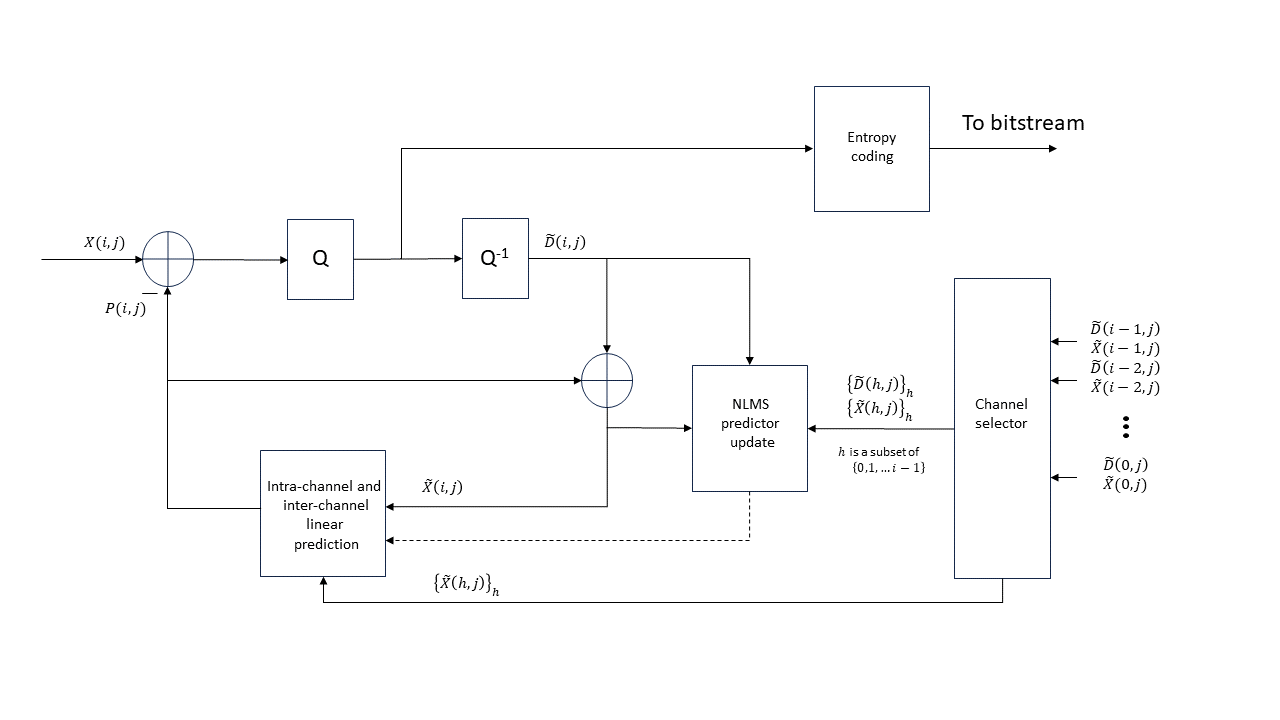


Figure 1. Block diagram presenting encoder-side intra-channel and inter-channel prediction in DCT domain when lms\_lpc\_block\_mode\_flag is equal to 1. The dashed line corresponds to sending the updated prediction coefficients to the predictor. The channel selector selects all or a subset of the *V* most recent coded channels.

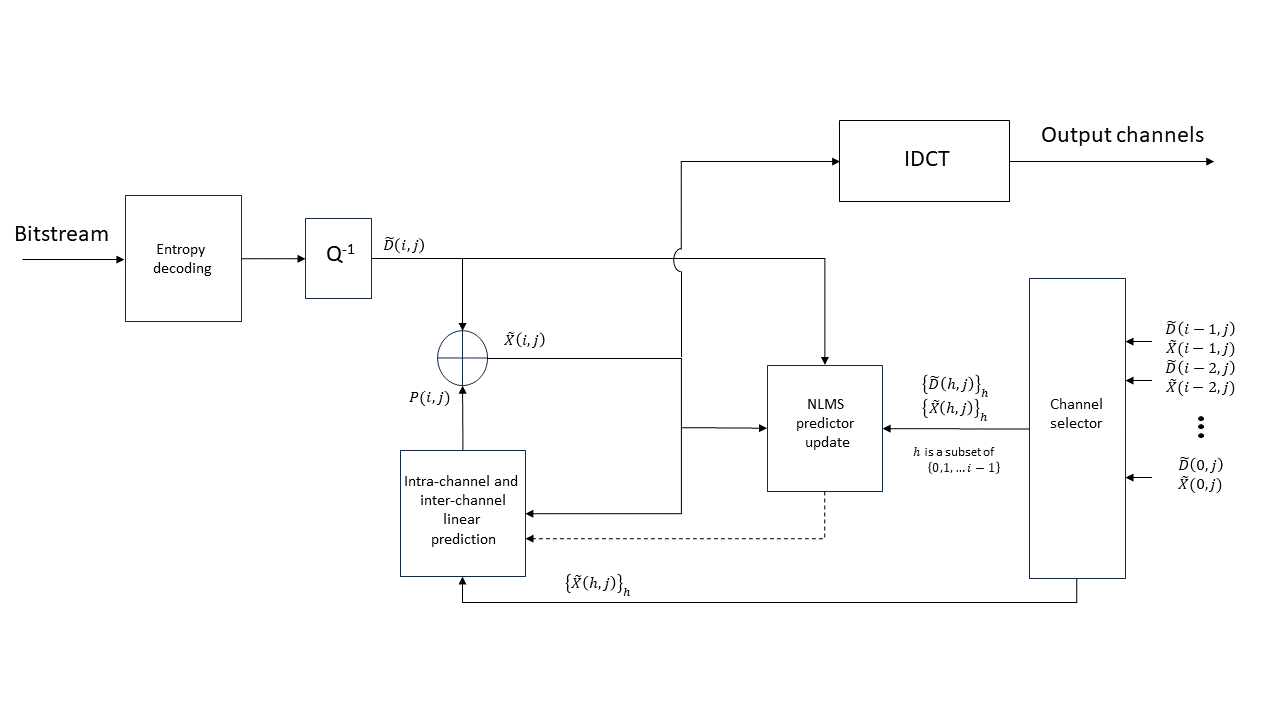


Figure 2. Block diagram presenting decoder-side intra-channel and inter-channel prediction in DCT domain when lms\_lpc\_block\_mode\_flag is equal to 1. The dashed line corresponds to sending the updated prediction coefficients to the predictor. The channel selector selects all or a subset of *V* most recent coded channels.

### LMS prediction and quantization in the transform domain

Prediction is applied in the DCT domain in two ways: prediction from reconstructed samples in the current channel and frame (intra-channel prediction) and/or from reconstructed samples of coded channels of the same frame (inter-channel prediction). There is no (temporal) inter-frame dependency for prediction. Two flags per frame and channel are transmitted in the bitstream to signal which type of prediction is active: when b\_use\_lpc equals to 1, intra-channel prediction is active, and when b\_use\_mcp equals to 1, inter-channel prediction is active. Prediction and adaptation of the prediction coefficients run from high to low frequencies.

Given a reconstructed spectral sample for channel *i* and frequency bin *j*, the prediction process starts at *j = L −* 1 where *L* is the transform length and proceeds until *j =* 0 (the lowest frequency bin). For channel *i* the prediction of sample is computed as

where *U* is the prediction order, *V* is the number of prediction channels and is initialized as

and the prediction coefficients are initialized as

Prediction order *U* is dependent on the transform length *L* as described in Table 1. All channels are coded in a successive manner such that for the first coded channel there is no prediction from other channels, and for the *i*th channel, .

Table 1. Default prediction order *U* corresponding to the supported transform lengths *L*.

|  |  |
| --- | --- |
| *L* | *U* |
| 2048 | 40 |
| 1024 | 20 |
| 512 | 10 |
| 256 | 5 |
| 128 | 4 |
| 64 | 4 |
| 32 | 4 |

In the encoder, the prediction residual is first computed as the difference between the signal and the prediction, as shown in Eq. (4).

The prediction residual is then quantized with a dead-zone scalar quantizer using a quantization scale factor (aka global gain) as shown in Eq. (5).

Where, is a quantization control parameter that ranges between 0 and 1023 and is transmitted to the decoder. The choice of the value of α is an encoder-only matter that does not affect interoperability, and ordinarily, *α* would be in the range of 0.0 to 0.5; as tested, *α* was set to 0.4054.

Samples are reconstructed from the residual using uniform reconstruction quantization (URQ), and the reconstruction is added to the prediction as

After each prediction and reconstruction step, prediction coefficients and are updated as

where the update step gain is computed as

and the input vector norm is computed as

Here, the predictor input total energy is computed as

In a fixed-point implementation, the term is efficiently approximated by a right-bit-shift of bits. Note that the encoder and decoder LMS implementations must perform the exact same calculations, thus, both the encoder and decoder must be either fixed point, or both must be floating point. It should be stressed that in the case of floating point, the encoder and decoder must have the exact same implementation of floating-point calculations.

Since prediction is done in a backwards-adaptive manner, the amount of side information describing the transform and predictors is minimized.

### Integer invertible DCT-domain LMS predictor

Lossless encoding and decoding follow the same principles as the lossy encoding and decoding, although in this case the transform must be integer invertible and the DCT-domain LMS predictor must also be integer invertible. Furthermore, the encoder considers both DCT-domain and time-domain prediction for the frame that is being coded and chooses the operation mode that results in a lower bitrate for that frame. The integer invertible transform is described in Section 2.3.5. The modifications to the LMS predictor equations are detailed in Section 2.3.2.

Both the lossy (IsLossy equals to 1) and lossless (IsLossy equals to 0) encoding and decoding can share the same DCT-domain LMS predictor (as described in Figure 1 and Figure 2), however, for lossless coding the quantization of the prediction must be explicit to ensure integer invertible behavior. Furthermore, the update step for the LMS predictor is explicitly fixed point.

The modification to prediction calculation is provided in the following equation.

where is 1 << 14 and is 15.

As for the lossy case, the signal value for the *i*th channel and *j*th sample can be reconstructed according to Eq. (4) from the residual transmitted in the bitstream to the decoder and the prediction . The signal energy is also computed according to Eq. (11).

The integer approximation of of the square root of the signal energy is then computed as shown in the following equation,

where means rounding up to nearest integer.

The update gain is then computed using the following equation.

The value of is then limited to the range shown in the following equation.

Finally, the predictors are updated as shown in the following equations.

where

### LPC Prediction in the time domain

The visualizations in Figure 3 and Figure 4 illustrate the operation of the predictors in the time domain, for the encoder and decoder, when IsLossy equals to 0 and enable\_dct equals to 0.

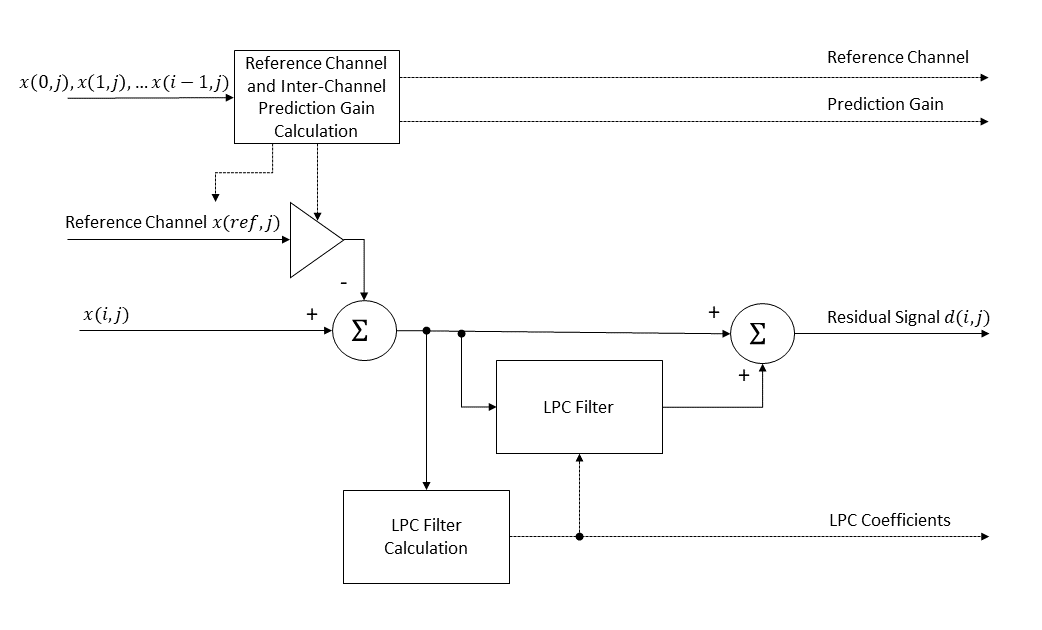


Figure 3. LPC encoder

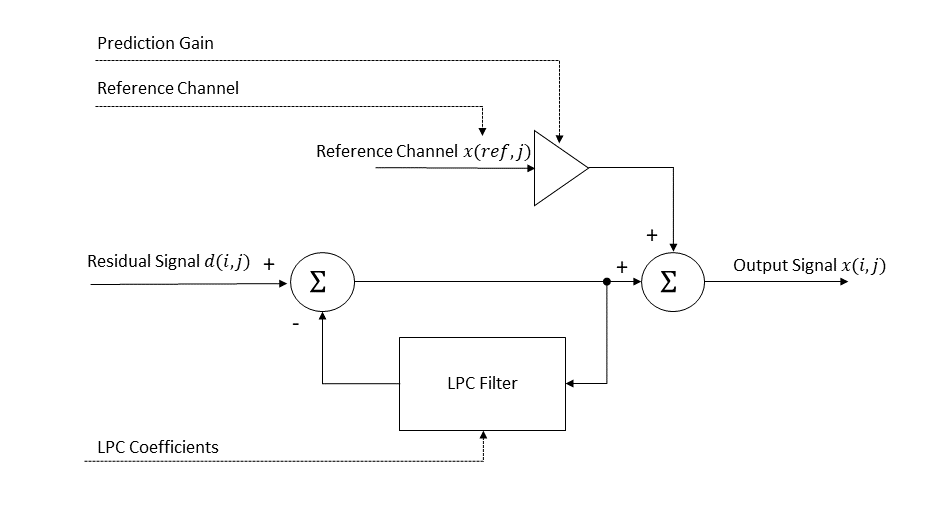


Figure 4. LPC decoder

The lossless encoder and decoder can switch on a frame-by-frame basis from the DCT domain to the time domain. As such it supports time-domain linear predictive coding (LPC), as well as inter-channel prediction in the time domain. Similarly to the DCT-domain operation, there is no temporal inter-frame dependency for prediction.

If time-domain LPC is enabled, the encoder calculates prediction coefficients and transmits the coefficients to the decoder. The time-domain LPC mode allows the update of the prediction coefficients at 32 sample intervals within a frame. The LPC coefficients are transmitted as reflection coefficients [1], which ensures the prediction filter is stable even with quantization. The lossless encoder and decoder support LPC filter orders ranging from 0th order (no LPC) to 31st order. The LPC filter order is transmitted in the bitstream to the decoder.

The following equation shows the reconstruction of the signal from the residual signal transmitted to the decoder.

where is the signal for the *i*th channel and *j*th sample in a frame,

is the residual signal transmitted to the decoder for the *i*th channel and *j*th sample in a frame,

are the direct-form LPC coefficients,

is the LPC order, and

and are (1<<22) and 23 respectively for LPC coding.

the LPC coefficients are transmitted as reflection coefficients, the direct-form LPC coefficients must be calculated at the decoder. The conversion of the reflection coefficients is recursive and shown in the following equation.

where the recursion index ,

is the direct-form LPC coefficients for the *m*th recursion, and

is the reflection coefficients transmitted to the decoder.

The LPC filter buffer is reset at frame boundaries, however, if the LPC coefficients are updated within a frame then the filter buffer is not reset, and the filter coefficients are swapped at the update boundary. As truncation of the reflection coefficients to any order *M*, where *M* ≤ *K* yields the optimal *M*thorder predictor, the predictor can be built up from 0th order to *K*th order during buffer resets at frame boundaries. This is achieved by recursing through equation 18 for the first *K* samples of the frame. Thus, the optimal filter is used during buffer resets at frame boundaries.

If inter-channel prediction in the time-domain is enabled, the encoder transmits a reference channel and a prediction gain, for a given target channel. Computation of the original signal from a residual signal is given by the following equation.

where

* is the signal for the *i*th channel and *j*th sample in a frame,
* is the residual signal transmitted to the decoder for the *i*th channel and *j*th sample in a frame,
* is the inter-channel prediction coefficient for the *i*th channel transmitted to the decoder,

is the reference channel used to predict the *i*th channel transmitted to the decoder.

### Entropy coding of transform coefficients for LMS- and LPC-based signal reconstruction

The entropy coding of the residual signal is performed using a simple combination of multi-dimension Huffman coding and Golomb-Rice coding. The signal is first divided into sub-regions, where the length of each sub-region is dependent on the frame length as shown in Table 2. The sub-regions are of uniform length except the last sub-region that is twice the previous sub-region lengths. In each sub-region the signal is encoded with one of 22 possible codebooks, that are either Huffman or Golomb-Rice codes. The 22 possible codebooks that can be selected for each sub-region are specified in Table 3. The codebooks are selected in each sub-region such that the codebook introduces no distortion (the maximum absolute value in a sub-region must be less than the largest absolute value (LAV) specified in Table 3), and the overall bitrate is minimized including the cost to transmit the codebook selection in each sub-region. The codebook index for the codebook selected in the 0th sub-region is transmitted as a 5bit number, while the codebook index for the subsequent sub-regions is transmitted as a Huffman encoded difference relative to the previous sub-region.

Table 2. Sub-region lengths for each available frame length

|  |  |  |
| --- | --- | --- |
| Frame length | Sub-region lengths | Sub-region count |
| 32 | [4,4,…,4,8] | 7 |
| 64 | [4,4,…,4,8] | 15 |
| 128 | [4,4,…,4,8] | 31 |
| 256 | [4,4,…,4,8] | 63 |
| 512 | [8,8,…,8,16] | 63 |
| 1024 | [16,16,…,16,32] | 63 |
| 2048 | [32,32,….,32,64] | 63 |

Table 3. Codebook type, largest absolute value (LAV), signed/unsigned codebook, and codebook size for each codebook index

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Codebook index** | **Codebook type** | **LAV** | **Dimension** | **Signed** | **Codebook size** |
| 0 | N/A | 0 | N/A | N/A | 0 |
| 1 | Huffman | 1 | 4 | Yes | 81 |
| 2 | Huffman | 1 | 4 | Yes | 81 |
| 3 | Huffman | 2 | 4 | No | 81 |
| 4 | Huffman | 2 | 4 | No | 81 |
| 5 | Huffman | 4 | 2 | Yes | 81 |
| 6 | Huffman | 4 | 2 | Yes | 81 |
| 7 | Huffman | 7 | 2 | No | 64 |
| 8 | Huffman | 7 | 2 | No | 64 |
| 9 | Huffman | 12 | 2 | No | 169 |
| 10 | Huffman | 12 | 2 | No | 169 |
| 11 | GR (2) | Inf | 1 | No | N/A |
| 12 | GR (3) | Inf | 1 | No | N/A |
| 13 | GR (4) | Inf | 1 | No | N/A |
| 14 | GR (5) | Inf | 1 | No | N/A |
| 15 | GR (6) | Inf | 1 | No | N/A |
| 16 | GR (7) | Inf | 1 | No | N/A |
| 17 | GR (8) | Inf | 1 | No | N/A |
| 18 | GR (9) | Inf | 1 | No | N/A |
| 19 | GR (10) | Inf | 1 | No | N/A |
| 20 | GR (11) | Inf | 1 | No | N/A |
| 21 | GR (12) | Inf | 1 | No | N/A |
| 22 | GR (13) | Inf | 1 | No | N/A |

### Lifting based transform for LMS-based signal reconstruction

The transform is an integerized DCT of type II and the IDCT is its inverse. Nonoverlapping frames of the input signal channels are processed at stride , which can be chosen as any power of two from 32 to 2048 samples. The normalization of the DCT is chosen to achieve an orthogonal transform of its input frame of samples to frequency bins according to

An integer invertible DCT of size is obtained from an integer invertible DFT of size by adapting the rotation and twiddling steps described in [2] to orthogonality and subsequently factorizing those to integer invertible lifting steps. The integer invertible DFT is derived by a generalization of the multidimensional lifting method of [3]. More precisely, a dual input integer invertible DFT is implemented by the flow diagram depicted in Figure 5. For a single input, a reduction to the dual input case with half the block size is obtained by splitting the input into even and odd samples and factorizing the Cooley-Tukey FFT butterfly into integer invertible lifting steps. With this implementation, the number of lifting steps per sample has a constant upper bound independent of the transform size . Since each lifting step is followed by rounding, this leads to a good approximation of the target DCT of Equation (20), even for large .



Figure 5. Calculation flow diagram for dual integer invertible DFTs with lifting steps.

## Blockwise predictive transform coding

If the value of the syntax element lms\_lpc\_block\_mode\_flag is equal to zero, the following transform coding scheme is applied.

The prediction signal sample values

are generated on each block out of already reconstructed sample values

that belong to the same channel or out of already reconstructed sample values

that belong to a previously coded channel in channel order.

A blockwise transform is applied to the original prediction residual sample values

at the encoder. The resulting transform coefficients are quantized to obtain the transform coefficient levels c[ j ]. The levels c[ j ] are entropy coded using the syntax below. At the decoder, the levels c[ j ] are multiplied with an appropriate quantization step size to obtain the intermediate reconstructed residual sample values

A sample wise prediction process can be applied to the intermediate reconstructed residual sample values resImd[ j ], where also the extended left residual sample values resExt[ j ] from the previous section are needed as an input. This yields the final reconstructed residual sample values

and the final reconstructed sample values

### DC Prediction

For the DC prediction mode, the predictions signal is defined as the mean value of the four preceding already reconstructed sample value. Thus, one puts

and sets

The DC prediction mode is used if the value of the syntax element block\_matching\_or\_cross\_channel\_pred\_flag is equal to zero and if the value of the syntax element block\_pred\_mode is equal to 0.

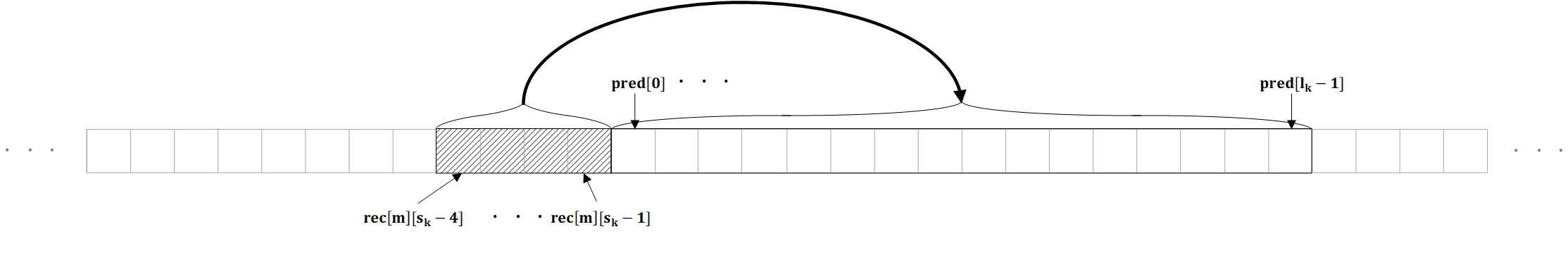


Figure 2: Illustration of the setup for the DC- and the Line-Fitting Prediction modes

### Line-Fitting Prediction

In the line fitting prediction, the prediction signal is defined by a line with a damped slope where the slope is determined by the four reconstructed sample values preceding the current block. Thus, with

one sets

The line fitting prediction mode is used if the value of the syntax element block\_matching\_or\_cross\_channel\_pred\_flag is equal to zero and if the value of the syntax element block\_pred\_mode is equal to 1.

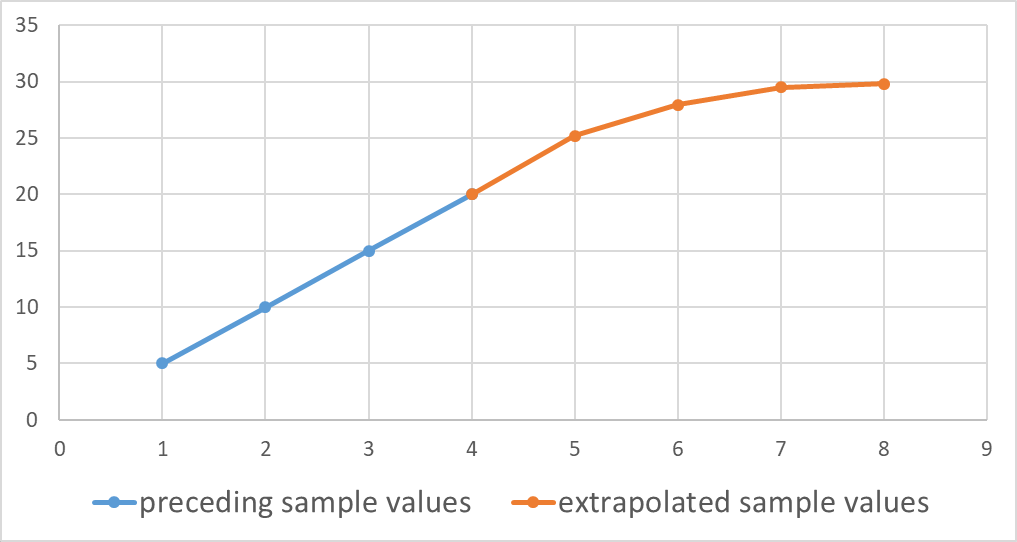


Figure 3: Illustration of the extrapolation method for the line fitting prediction.

### Cross-Channel Prediction

In thecross channel prediction process, the prediction signal is generated by a linear model using collocated reconstructed sample values from different channels. The parameters of the linear model are derived from already reconstructed samples on a left adjacent template of size tSize = 16.

The cross channel prediction mode is invoked if the syntax elements block\_matching\_of\_cross\_channel\_pred\_flag and cross\_channel\_pred\_flag are equal to 1. Cross channel prediction with a single hypothesis (cc\_pred\_mult\_hyp\_flag equal to zero) and cross channel prediction with two hypothesis (cc\_pred\_mult\_hyp\_flag equal to one) are supported.

For the cross channel mode, which is only applicable if , a reference channel index with is transmitted. In the case of multi-hypothesis cross channel prediction, also a second reference channel index with is transmitted.

If cc\_pred\_mult\_hyp\_flag is equal to zero and , is not signaled and inferred to be 0. Similarly, if cc\_pred\_mult\_hyp\_flag is not equal to zero and , and are not signaled and are inferred to be 0 and 1, respectively.

#### Cross channel prediction with a single prediction hypothesis

For the cross channel prediction with a single hypothesis, integral values and are derived from the preceding already reconstructed sample values

in the reference channel and from the preceding already reconstructed sample values

in the current channel by solving a linear equation that is motivated by the minimization of the quadratic prediction error

If cc\_pred\_offset\_only\_flag is equal to one, is set to 1 which results in the simplified equation

whose integral solution is given as

In this case, an intermediate prediction is computed as

Otherwise, if cc\_pred\_offset\_only\_flag is not equal to one, the parameters of the aforementioned linear equation

are determined as follows:

The linear equation is solved at the decoder by invoking a specified process for the solution of linear equations in fixed point arithmetic which is described in Section 5.10.4.5 of the attached Draft Specification Text. Here, the precision shift is set as ccShift = 16.

After the determination of andan intermediate prediction signal is computed as

If cc\_pred\_filter\_flag is equal to zero, the final prediction signal is

Otherwise (cc\_pred\_filter\_flag is equal to one) the intermediate prediction is extended to the left by

where fPdL equals 3. The intermediate prediction is extrapolated to the right by setting fPdR=4 and invoking the extrapolation process from section 8.3.1 of the attached Draft Specification Text, which results in the sample values Then, the final prediction signal is

with . Here, depending on the value of the syntax element cc\_pred\_filter\_idx, the filter coefficients CCFiltCoeffs[ k ] either represent a fixed smoothing filter or a fixed half-pel interpolation filter as described in the following table:

|  |  |
| --- | --- |
| cc\_pred\_filter\_idx | CCFiltCoeffs |
| 0 | {-3, 0, 19, 32, 19, 0, -3, 0} |
| 1 | {-1, -4, 8, 29, 29, 8, -4, -1} |

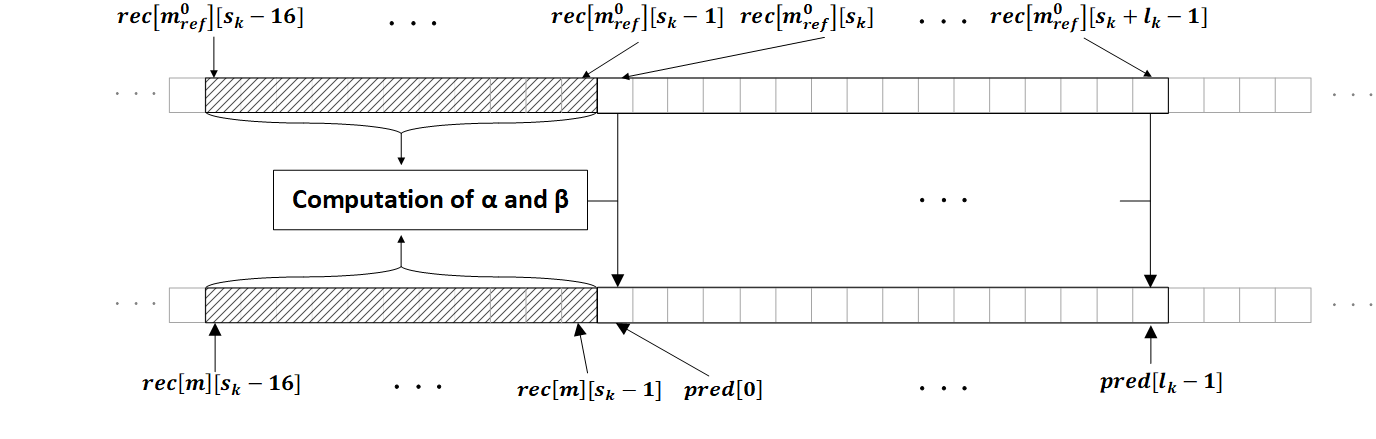


Figure 4: Ilustration of the Inter-Channel Prediction with one reference channel

#### Cross channel prediction with two prediction hypotheses

For the cross channel prediction with two hypotheses, integral values and are derived from the preceding already reconstructed sample values

in the reference channels and and from the preceding already reconstructed sample values

in the current channel by solving a linear equation that is motivated by the minimization of the quadratic prediction error

If cc\_pred\_offset\_only\_flag is equal to one, and are set to 0.5. This results in the simplified equation

which is specified to be solved by

In this case, an intermediate prediction is computed as

Otherwise, if cc\_pred\_offset\_only\_flag is not equal to one, the parameters of the aforementioned linear equation

are determined as follows:

As for the case of a single hypothesis, the latter 3x3linear equation is solved at the decoder by invoking the process of Section 5.10.4.5 of the attached Draft Specification Text with precision ccShift=16.

After the determination of , andthe intermediate prediction signal is computed as

If cc\_pred\_filter\_flag equal to zero, the final prediction signal pred is equal to .

Otherwise, the same filtering process as for the case of cross channel prediction with a single hypothesis is applied for the intermediate prediction signal to obtain the final prediction signal .

### Block-Matching Prediction

In the block matching prediction mode, the prediction signal is generated by copying the sample values of already reconstructed reference blocks of the same channel. The locations of these blocks are transmitted. Block matching prediction with a single hypothesis and block matching prediction with two hypotheses are supported.

Additionally, an offset can be derived at the decoder from adjacent already reconstructed sample values on a template of size tSize=16. A half-pel accurate representation of the location of the reference block and a smoothing prediction filter are also supported.

The block matching prediction mode is invoked if the value of the syntax element block\_matching\_or\_cross\_channel\_pred\_flag is equal 1 and if the value of the syntax element cross\_channel\_pred\_flag is not equal to one. If bm\_pred\_mult\_hyp\_flag is equal to 0, block matching prediction with a single hypothesis is used. If bm\_pred\_mult\_hyp\_flag is equal to 1, block matching prediction with two hypothesis is used.

The values of the block matching offsets are coded predictively by invoking the corresponding values from previous blocks as a prediction. Here, the values of the current channel are invoked for prediction if the value of the syntax element bm\_pred\_off\_pred\_prev\_ch\_flag is equal to 0 and the values of the preceding channel are invoked if the value of the syntax element bm\_pred\_off\_pred\_prev\_ch\_flag is equal to 1.

#### Block-Matching Prediction with a single hypothesis

For block matching prediction with a single hypothesis and offset value , the unfiltered prediction signal is

If the value of the syntax element bm\_pred\_filter\_flag is equal to zero, the prediction signal before offset addition is

Otherwise, if bm\_pred\_filter\_flag is equal to 1, the intermediate prediction is extended to the left by fPdL=3 samples and it is extended to the right by fPdR=4 samples according to

and the filtered prediction signal before offset calculation is computed as

with . Here, depending on the value of the syntax element bm\_pred\_filter\_idx, the filter coefficients BMFiltCoeffs[ k ] either represent a fixed smoothing filter or a fixed half-pel interpolation filter as described in the following table:

|  |  |
| --- | --- |
| bm\_pred\_filter\_idx | BMFilterCoeffs |
| 0 | {-3, 0, 19, 32, 19, 0, -3, 0} |
| 1 | {-1, -4, 8, 29, 29, 8, -4, -1} |

If bm\_pred\_add\_offset\_flag is equal to zero, the final prediction sample values are

Otherwise, if bm\_pred\_add\_offset\_flag is equal to one, an offset value is calculated on the tSize = 16 left adjacent already reconstruct boundary samples as

Then, the final prediction sample values are

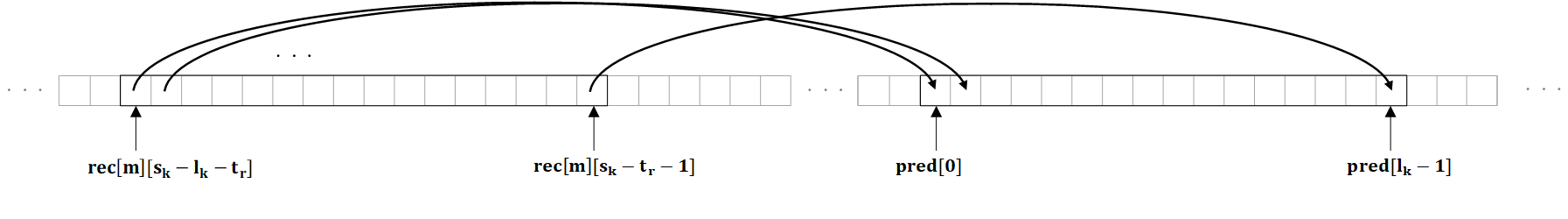


Figure 5: Illustration of the single hypothesis Block-Matching Prediction (without filtering)

#### Block-Matching Prediction with two hypotheses

For the block matching prediction with two hypotheses and offset values , two intermediate prediction signals before offset value addition, and , are computed in the same way as in the case of block matching prediction with a single hypothesis. Each prediction hypothesis can individually be filtered as in the case of a single prediction hypothesis, where the applicability of a filtering depends on the value of the syntax elements bm\_pred\_filter\_flag[ 0 ], bm\_pred\_filter\_flag[ 1 ] and where the selection of the filter depends on the syntax elements bm\_pred\_filter\_idx[ 0 ] and bm\_pred\_filter\_idx[ 1 ].

If the value of the syntax element bm\_pred\_add\_offset\_flag is equal to zero, the final prediction signal sample values are

Otherwise ( bm\_pred\_add\_offset\_flag is equal to one), an offset value is calculated on the tSize = 16 left adjacent boundary values according to

and the final prediction signal sample values are

### Extension of prediction signals the left adjacent samples

In order to obtain an extension of the prediction residual to the left, which is invoked in the sample wise prediction of reconstructed residuals described in Section 2.4.12 below, for each prediction signal, an extension

to tSize=16 many left adjacent prediction values is supported. Here, to determine the extended prediction, essentially the same process that was used in the respective prediction generation on the given block is invoked. Further details can be found in clause 8.3 of the attached Draft Specification Text. Given the extended prediction signal, an extended residual signal is defined as

Here, a specified padding process for sample values at unavailable sample locations is invoked.

### Overview of Transform coding of prediction residuals

The following transform coding scheme is applied. A blockwise transform is applied to the original prediction residual sample values

at the encoder. The resulting transform coefficients are quantized to obtain the transform coefficient levels c[ j ]. The levels c[ j ] are entropy coded using the syntax below. At the decoder, the levels c[ j ] are multiplied with an appropriate quantization step size to obtain the intermediate reconstructed residual sample values

A sample wise prediction process can be applied to the intermediate reconstructed residual sample values resImd[ j ], where also the extended left residual sample values resExt[ j ] from the previous section are needed as an input. This yields the final reconstructed residual sample values

and the final reconstructed sample values

### Block wise transforms

The identity transform, the Discrete Sine Transform DST-II and the Discrete Cosine Transform DCT-II are supported as block wise transforms. The DST-II is only supported if the blocksize satisfies and the DCT-II is only supported if the blocksize satisfies . The block wise transform is determined by the syntax elements transform\_skip\_flag and the syntax element transform\_dst\_flag as follows. If transform\_skip\_flag is equal to 1, the identity transform is used. If transform\_skip\_flag is equal to 0 and transform\_dst\_flag is equal to 1, the DST-VII transform is used. If transform\_skip\_flag is equal to 0 and transform\_dst\_flag is equal to 0, the DCT-II transform is used. The forward and inverse transforms are implemented in fixed point arithmetic with 32 bit precision.

At the encoder, the transform coefficients are quantized and the resulting quantization indices

also referred to as transform coefficient levels, are entropy coded. Two separate paths of level coding are supported depending on whether a trigonometric transform or whether the identity transform is used for the current block.

### Level coding for the identity transform

First, in the special case that the identity transform is used and that no block-based prediction and that no sample wise prediction are used, the levels c[ j] of the identity transform are coded in a fixed length representation depending on the bit-depth, starting from j=0 until , where all bins are bypass coded.

Otherwise, for the identity transform, the coefficient levels are always sequentially coded in decreasing order starting from by the following procedure.

For a coefficient level c[ j ], the syntax element abs\_tskip\_coeff\_gt0\_flag[ k ] indicates whether c[ j ] is zero or not. If , the absolute value is transmitted as follows. First, the syntax elements abs\_tskip\_coeff\_offset[ j ] is coded, with

0<= abs\_tskip\_coeff\_offset[ j ] < 5.

If abs\_tskip\_coeff\_offset[ j ]<4, one sets

Otherwise, a truncated Rice whose prefix is specified by the syntax element abs\_tskip\_coeff\_rem\_pref[ j ] is invoked, where

0<= abs\_tskip\_coeff\_rem\_pref[ j ] < 32.

The Rice parameter R of this Rice code is determined from the sum

of absolute values of all previously coded coefficient levels of the same block. If abs\_tskip\_coeff\_rem\_prefix[ j ] < 32, the suffix abs\_tskip\_coeff\_rem\_fl\_suffix[ j ] is coded with R bits and one sets

Otherwise, the suffix abs\_tskip\_coeff\_rem\_eg0\_suffix[ j ] is coded using an Exponential-Golomb code of order zero and one sets

Finally, the sign of c[ j ] is coded using the syntax element tskip\_coeff\_sign\_flag[ j ],

The syntax element abs\_tskip\_coeff\_gt0\_flag[ k ] is context coded using a single context model per channel.

The bins of the truncated unary binarization of abs\_tskip\_coeff\_offset[ j ] are context coded, where a dedicated context model per channel is used for each bin position. For each channel, 5 context models are supported for the coding of these bins.

In the same way, the bins of the truncated unary binarization of abs\_tskip\_coeff\_rem\_prefix[ j ] are context coded, where a dedicated context model per channel is used for each bin position. For each channel, 33 context models are supported for the coding of these bins.

The bins that belong to the prefix in the Exponential-Golomb binarization of abs\_tskip\_coeff\_rem\_eg0\_suffix[ j ] are context coded where the context model selection depends on the position of a bin. For each channel, 32 context models are supported for the coding of these bins.

All other bins that occur in the supported coding scheme for the identity transform are coded in bypass mode.

### State machine for level coding and coefficient reconstruction process

The level coding and the coefficient reconstruction process for transform coefficient levels of a trigonometric transform invoke a specified state machine which can have 1 state, 4 states or 8 states.

As further described in Section 2.4.11 below, the state machine determines the switching between two scalar quantizers and : For a given quantization step size , the reconstruction levels of are the even integral multiples of while the reconstruction levels of are the odd multiples of and the value 0.

The state transition from a state to a next state given a value q is determined by the respective state transition table invoking the state and the parity of q. The number of states and the invoked state-transition table are determined by the syntax element if\_residual\_quant\_mode as follows.

If if\_residual\_quant\_mode is equal to 0, a single state is used. This case corresponds to scalar quantization with uniform reconstruction quantizer .

It if\_residual\_quant\_mode is equal to 1, the following state transition table is used:

|  |  |  |
| --- | --- | --- |
|  | Next state | |
| Current state | even parity | odd parity |
|  | 0 | 1 |
|  | 2 | 3 |
|  | 1 | 0 |
|  | 3 | 2 |

If if\_residual\_quant\_mode is equal to 2, the following state transition table is used:

|  |  |  |
| --- | --- | --- |
|  | Next state | |
| Current state | even parity | odd parity |
|  | 0 | 2 |
|  | 5 | 7 |
|  | 1 | 3 |
|  | 6 | 4 |
|  | 2 | 0 |
|  | 4 | 6 |
|  | 3 | 1 |
|  | 7 | 5 |

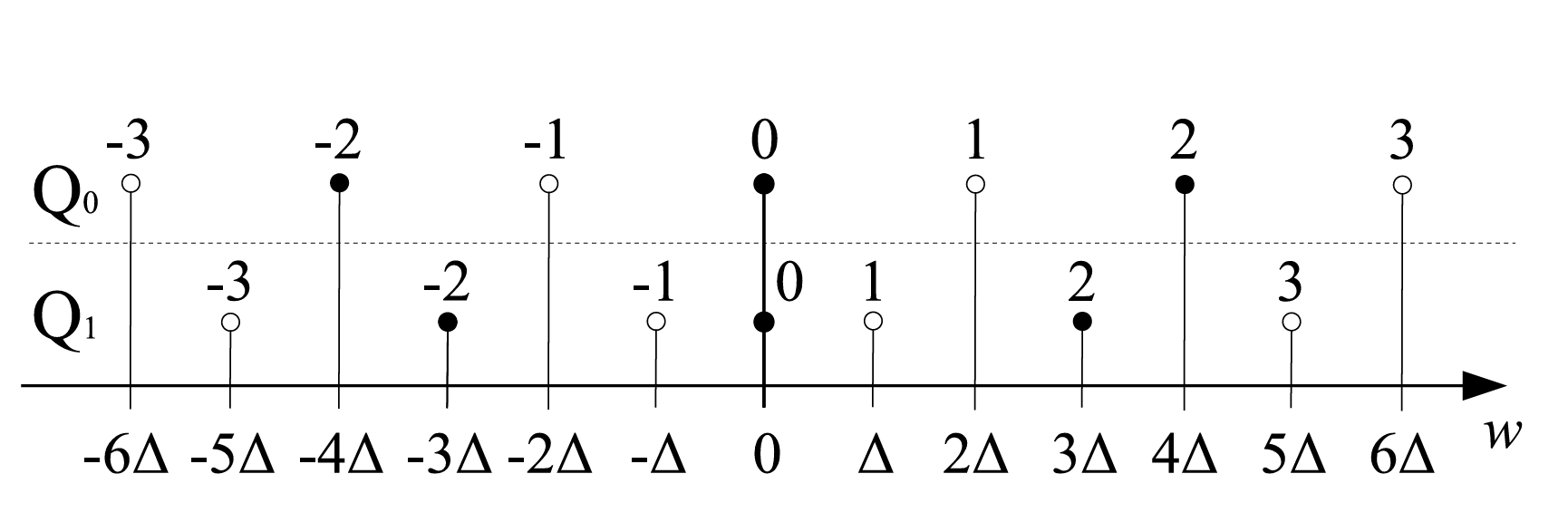


Figure 6: Illustration of the reconstruction points of the quantizers and

### Level coding for trigonometric transforms

If a trigonometric transform is used, a value last\_scan\_pos with is transmitted first. Here, for all with one has

and, if , one always has

(2)

Then, starting with last\_scan\_pos, the coefficients are sequentially coded in a backwards scan and in a single pass. Thus, one first codes and then, until , codes after having coded .

The state Qstate[ j ] of the used state machine S that holds for the coefficient level c[ j ] is determined as follows. For j=last\_scan pos, one initializes Qstate[ j ] = 0 and then, after the reconstruction of c[ j ], one updates Qstate[ j-1 ] according to the parity c[ j ]&1 of c[ j ] by invoking the respective state-transition table from the previous section.

The value of last\_scan\_pos is derived from the syntax elements last\_sbb\_index\_gt0\_flag, last\_sbb\_index\_rem and last\_index\_offset, where

0 <= last\_sbb\_index\_rem < (>>1) – 1 and 0 <= last\_index\_offset < 2

as follows: If last\_sbb\_index\_gt0\_flag is equal to zero, one sets last\_scan\_pos = last\_index\_offset. Otherwise, one sets

last\_scan\_pos = ((last\_sbb\_index\_rem+1)<<1) + last\_index\_offset.

The value of last\_sbb\_index\_rem is binarized using a limited Exponential-Golomb code.

For the coding of a coefficient , its significance flag , defined as

is transmitted first except for the case that j = last\_scan\_pos and last\_scan\_pos>0 , where the value of abs\_trafo\_coeff\_gt0\_flag[j] is inferred to be in accordance to (2).

Next, if , the absolute value and the sign of are transmitted. The absolute value is determined by the two syntax elements abs\_trafo\_coeff\_offset[ j ] and abs\_trafo\_coeff\_remainder[ j ], where

0 <= abs\_trafo\_coeff\_offset[ j ] < 20 and 0 <= abs\_trafo\_coeff\_remainder[ j ],

by using the relation

(5)

The value abs\_trafo\_coeff\_offset[ j ] is transmitted first using truncated unary coding. Then, if abs\_trafo\_coeff\_offset[ j ] = 19, the value abs\_trafo\_coeff\_remainder[ j ] is transmitted using Exponential Golomb coding. Finally, the sign is transmitted by the syntax element trafo\_coeff\_sign\_flag[ j ].

The bins that belong to the prefix in the Exponential-Golomb binarization of last\_sbb\_index\_rem are context coded where the context model selection depends on the position of a bin. For each channel, 15 context models are supported for the coding of these bins.

The flag is context coded. The selection of the context model depends on the position j within the block, the template sum

of absolute values of up to three neighboring previously coded transform coefficient levels, and the parity of the state Qstate[ j ]. The reason for the latter dependency is that, from an encoder point of view, the reconstruction interval around zero is larger if Qstate[ j ] is even than it is if Qstate[ j ] is odd. For each channel, 54 context models for are supported.

The bins of the truncated unary binarization of abs\_trafo\_coeff\_offset[j] are context coded. The same context model is used for all bins. However, this context model is selected out of a set of 9 context models per channel, where the selection depends on the position j.

The bins that belong to the prefix in the Exponential-Golomb binarization of abs\_trafo\_coeff\_remainder[ j ] are context coded where the context model selection depends on the position of a bin. For each channel, 31 context models are supported for the coding of these bins.

All other bins that occur in the coding scheme for the trigonometric transforms are coded in bypass mode.

### Reconstruction process for transform coefficients

Each transform coefficient level value c[ j ] is reconstructed by invoking a quantization stepsize that is transmitted in the bit stream. For the trigonometric transforms, this reconstruction process involves the state Qstate[ j ] at the current position j with respect to the specified state machine S which is determined as in the previous section. To ease notation, if the identity transform is used, it is assumed that Qstate[ j ] is equal to 0. Then, putting

the reconstruction of the intermediate residual coefficients can always be realized as

### Sample wise prediction of reconstructed residual sample values

If the identity transform is used on the current block , a sample wise prediction is supported for the residual samples for the case that the prediction mode is, the Inter-Channel Prediction, the Block-Matching Prediction or the Zero Prediction mode.

#### General form of sample wise prediction

The general sample wise prediction works as follows.

Input are

* The reconstructed sample values   
    
  of the adjacent blocks from up to three previously coded channels
* The reconstructed residual sample values   
    
  on the left adjacent template of size tSize which are obtained as in Section 2.4.5.
* The intermediate reconstructed residual sample values   
    
  which are obtained as in Section 2.4.11.
* Two set of filter coefficients  
    
  together with a filter precision pRF and the corresponding rounding offset offRnd = (1<<pRF).

The array of sample values u[ j ] with is initialized for negative values of j according to

The values u[ j ] are sequentially computed starting with j=0 and proceeding until by setting for each j:

Here the sum invoking the sample values of the previous channels is understood to be zero in the case that .

The final reconstructed residual sample res[ j ] are set as

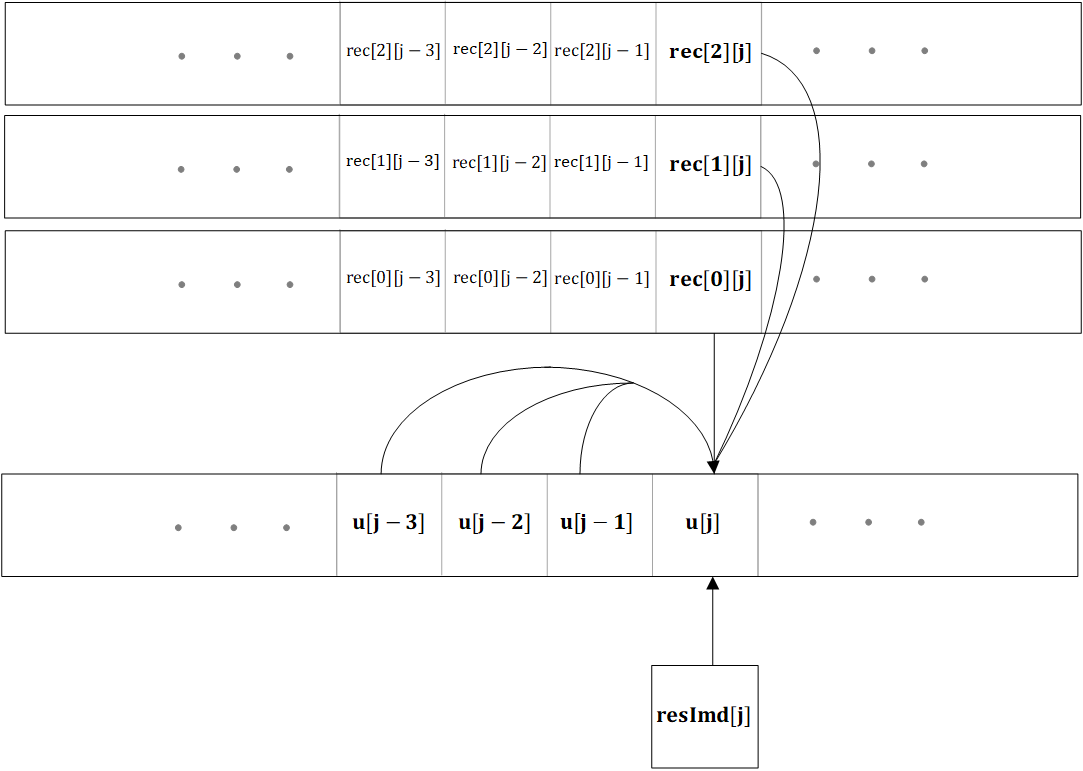


Figure 7: Illustration of the sample wise prediction

#### Sample wise prediction with fixed weights

Three sample wise prediction modes with fixed weights are supported. All three modes do not invoke the samples from previous channels, which means that the weights are always zero for these modes.

First, the difference sample wise prediction mode, with

is supported.

Second, the slope sample wise prediction mode, with

is supported.

Third, the half slope sample wise prediction mode with

is supported.

The selection of the sample wise prediction mode with fixed weights is determined by the syntax elements spred\_lpf\_or\_diff\_flag, spred\_lpf\_flag and spred\_rem\_mode\_idx as follows. If spred\_lpf\_or\_diff\_flag is equal to 1 and spred\_lpf\_flag is equal to 0, the difference sample wise prediction mode is used. If spred\_lpf\_or\_diff\_flag is equal to 0 and spred\_rem\_mode\_idx is equal to 0, the slope sample wise prediction mode is used. If spred\_lpf\_or\_diff\_flag is equal to 0 and spred\_rem\_mode\_idx is equal to 1, the half slope sample wise prediction mode is used

A complete bypassing of the sample wise prediction process is also supported, in which case the final residual sample values res[ j ] are set to the intermediate residual values resImd[ j ]. This is invoked if spred\_lpf\_or\_diff\_flag is equal to 0 and spred\_rem\_mode\_idx is equal to 2.

#### Sample wise prediction with adaptive weights

The sample wise prediction with adaptive weights is used if the syntax elements spred\_lpf\_or\_diff\_flag and spred\_lpf\_flag are equal to one. If the sample wise prediction with adaptive weights is used, the values of the coefficients and are coded in the bitstream.

The maximal number of non-zero coefficients is determined by the syntax element lpf\_num\_weights\_idx with 0 < = lpf\_num\_weights\_idx < 8 by setting

and inferring

The filter precision is set to 14 for the case of sample wise prediction with adaptive weights.

#### Transmission of filter coefficients for prediction from the current channel only

If the syntax element lpf\_prev\_ch\_flag is equal to zero, no sample values from previous channels and no adaptive offset value are used for the sample wise prediction and thus is set to 0 for all c. In this case, for transmission and predictive coding, the values of the coefficients with 0 <= p < fltrSz are represented by the corresponding reflection-coefficients (sometimes also referred to as partial correlation or PARCOR coefficients) . The coefficients are derived from the coefficients by invoking the following algorithm:

Set p = 0

do

set q = 0

do

while( q < p)

set q = 0

do

while ( q < p )

while( p < fltrSz )

For transmission, the values are represented as a quadratic polynomial in a value . To determine the value , a value is transmitted in the bitstream as follows. If abs\_lpf\_weight\_greater0\_flag[ p ] is equal to zero, one sets Otherwise, the absolute value is inferred from the value of the syntax element abs\_lpf\_weight\_minus1[ p ] by setting

while the sign of is inferred from the syntax element lpf\_weight\_sign\_flag[ p ].

If the value of the syntax element lpf\_prev\_ch\_flag is equal to 0, one sets .

If the value of the syntax lpf\_prev\_ch\_flag is equal to 1, one sets

,

where is a corresponding value of from a previous block of the same channel.

After the reconstruction of the , one updates:

#### Transmission of filter coefficients for prediction invoking a previous channel

If the syntax element lpf\_prev\_ch\_flag is equal to 1, it is inferred that samples from previous channels are to be invoked for the adaptive predictive filtering process. In this case, fltrSz many coefficients as well as many filter coefficients are coded. All coefficients are coded directly invoking the syntax elements abs\_lpf\_weight\_greater0\_flag[  ], and lpf\_weight\_sign\_flag[  ], which means that the reflection-coefficient representation as well as the predictive transmission of filter coefficients is disabled for this case. The reflection representation of filter coefficients is not used since, from an encoder point of view, the correlation matrix which is typically invoked to determine the filter coefficients by solving a system of linear equations is generally a Toeplitz matrix only if no sample values from previous channels are employed. On the other hand, the reflection coefficient representation can only be meaningfully derived for the solution of a linear equation when A is a Toeplitz matrix.

### Arithmetic coding engine

For entropy coding, the state-based probability estimator and arithmetic coding engine of the Neural Network Coding Standard (NNC, ISO/IEC 15938-17) with a few minor modifications are supported. First, the two state variables per context model are represented with 10 and 13 bit, respectively, and the adaptation rates are set to 2 and 5, respectively. Next, the first 63 bins of each context model are encoded with an adaptation rate of 1 in order to allow a fast initial adaptation to the received bin sequence. The coding of syntax elements of a channel employs separate context models per channel, as indicated for example in Section 2.4.8 and Section 2.4.10. However, for initialization, the channel dependency is neglected and the same initialization probability is used for a context model across all channels. Here, the initialization probability depends on the quantization parameter of the sequence, which is represented by the syntax element if\_indep\_init\_block\_qp. Initialization values have been trained on a set of bitstreams of all sequences of all three datasets and all working points.

# References

[1] J. G. Proakis and B. G. Manolakis, Digital Signal Processing Principles, Algorithms, and Applications (3rd Ed.). 1996, Prentice-Hall Inc. Upper Saddle River, N.J. pp 863-864.

[2] J. Makhoul, “A fast cosine transform in one and two dimensions,” in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 28, no. 1, pp. 27-34, February 1980.

[3] R. Geiger, Y. Yokotani and G. Schuller, “Improved integer transforms for lossless audio coding,” *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers*, 2003, Pacific Grove, CA, USA, 2003, pp. 2119-2123 Vol.2.