Detailed Technical description of the Parametric Stereo Tool in the enhanced aacPlus audio codec

8.1 Introduction

This document provides the detailed technical description of the Parametric Stereo Tool, as used in the enhanced aacPlus audio codec as submitted to and tested by 3GPP. The proposal combines HE-AAC and the parametric stereo coding tool. The parametric stereo coding tool is able to capture the stereo image of the audio input signal into a limited number of parameters, requiring only a small overhead ranging from a few kbit/s for medium quality, up to about 9 kbit/s for high quality. Together with a monaural downmix of the stereo input signal generated by the parametric stereo encoding coding tool, the parametric stereo decoding tool is able to regenerate the stereo signal.

The next sections detail the decoding process of the parametric stereo data. In Annex A of this document a normative description of the combination of HE-AAC with the parametric stereo coding tool is provided. In Annex B the Huffman tables used for decoding the parametric stereo payload are provided.

8.2 Terms and definitions

f_s – The sampling frequency in Hertz.

IID - Inter-channel Intensity Differences.

IPD - Inter-channel Phase Differences.

OPD - Overall Phase Differences.

ICC - Inter-channel Coherence.

8.3 Symbols and abbreviations

8.3.1 Arithmetic operators

$\lfloor x \rfloor$ Round x towards minus infinity

$\lceil x \rceil$ Round x towards plus infinity.

mod Modulus operator. Defined only for positive numbers.

8.3.2 Relation operators

x?y:z If x is true then y else z.

8.3.3 Mnemonics

The following mnemonics are defined to describe the different data types used in the coded bit-stream.

uimsbf Unsigned integer, most significant bit first.

simsbf Signed integer, most significant bit first.

bslbf Bitstream left bit first.

8.3.4 Ranges

[0, 10] A number in the range of 0 up to and including 10.

[0, 10> A number in the range of 0 up to but excluding 10.

8.3.5 Number notation

%X Binary number representation (e.g. %01111100).

$X Hexadecimal number representation (e.g. $7C).

X Numbers with no prefix use decimal representation (e.g. 124).

8.4 Payload for the Parametric Stereo object type

8.4.1 ps_data() Payload

Table 8.1 – Syntax of ps_data()

Syntax	Num. bits	Mnemonic
ps_data()		
{		
if (**enable_ps_header**) {	**1**	**uimsbf**
if (**enable_iid**) {	**1**	**uimsbf**
iid_mode	**3**	**uimsbf**
nr_iid_par = nr_iid_par_tab[iid_mode]		
nr_ipdopd_par = nr_ipdopd_par_tab[iid_mode]		
}		
if (**enable_icc**) {	**1**	**uimsbf**
icc_mode	**3**	**uimsbf**
nr_icc_par = nr_icc_par_tab[icc_mode]		
}		
enable_ext	**1**	**uimsbf**
}		
frame_class	**1**	**uimsbf**
num_env_idx	**2**	**uimsbf**
num_env = num_env_tab[frame_class][num_env_idx]		
if (frame_class) {		
for (e=0 ; e<num_env ; e++) {		
border_position[e]	**5**	**uimsbf**
}		
}		
for (e=0 ; e<num_env ; e++) {		
if (enable_iid) {		
iid_dt[e]	**1**	**uimsbf**
iid_data()		
}		
}		
for (e=0 ; e<num_env ; e++) {		
if (enable_icc) {		
icc_dt[e]	**1**	**uimsbf**
icc_data()		
}		
}		
if (enable_ext) {		
cnt = **ps_extension_size**	**4**	**uimsbf**
if (cnt == 15)		
cnt += **esc_count**	**8**	**uimsbf**
num_bits_left = 8 * cnt		
while (num_bits_left > 7) {		
ps_extension_id	**2**	**uimsbf**
num_bits_left -= 2		
ps_extension(ps_extension_id, num_bits_left)		
}		
fill_bits	num_bits_left	
}		
}		
ps_extension(ps_extension_id, num_bits_left){		
if (ps_extension_id == 0) {		

if (**enable_ipdopd**) {	**1**	**uimsbf**
for (e=0 ; e<num_env ; e++) {		
ipd_dt[e]	**1**	**uimsbf**
ipd_data()		
opd_dt[e]	**1**	**uimsbf**
opd_data()		
num_bits_left -= ipd_bits + opd_bits + 2		Note 1
}		
}		
reserved_ps	**1**	**uimsbf**
num_bits_left -= 2		
}		
}		
iid_data() {		
if (iid_dt[e]) {		
for (b=0 ; b<nr_iid_par; b++) {		
iid_par_dt[e,b] = ps_huff_dec(huff_iid_dt[iid_quant],**bs_codeword**);	**1…20**	Note 2
}		
}		
else {		
for (b=0 ; b<nr_iid_par; b++) {		
iid_par_df[e,b] = ps_huff_dec(huff_iid_df[iid_quant],**bs_codeword**);	**1…18**	Note 2
}		
}		
}		
icc_data() {		
if (icc_dt[e]) {		
for (b=0 ; b<nr_icc_par; b++) {		
icc_par_dt[e,b] = ps_huff_dec(huff_icc_dt,**bs_codeword**);	**1…14**	**bslbf**
}		
}		
else {		
for (b=0 ; b<nr_icc_par; b++) {		
icc_par_df[e,b] = ps_huff_dec(huff_icc_df,**bs_codeword**);	**1…13**	**bslbf**
}		
}		
}		
ipd_data() {		
if (ipd_dt[e]) {		
for (b=0 ; b<nr_ipdopd_par; b++) {		
ipd_par_dt[e,b] = ps_huff_dec(huff_ipd_dt,**bs_codeword**);	**1…5**	**bslbf**
}		
}		
else {		
for (b=0 ; b<nr_ipdopd_par; b++) {		
ipd_par_df[e,b] = ps_huff_dec(huff_ipd_df,**bs_codeword**);	**1…4**	**bslbf**
}		
}		
}		
opd_data() {		
if (opd_dt[e]) {		
for (b=0 ; b<nr_ipdopd_par; b++) {		
opd_par_dt[e,b] = ps_huff_dec(huff_opd_dt,**bs_codeword**);	**1…5**	**bslbf**
}		
}		
else {		
for (b=0 ; b<nr_ipdopd_par; b++) {		
opd_par_df[e,b] = ps_huff_dec(huff_opd_df,**bs_codeword**);	**1…5**	**bslbf**
}		

} }		
Note 1: ipd_bits and opd_bits represent the number of bits read by ipd_data() and opd_data() respectively. Note 2: the index iid_quant into huff_iid_df is obtained from Table 8.2.		

8.5 Semantics

8.5.1 Decoding of ps_data() Bitstream Payload

ps_data() – syntactic element that contains the parametric stereo data.

enable_ps_header – If set to %1, the PS header data configuring the PS decoder is transmitted. Otherwise, the latest configuration persists.

enable_iid – If enable_iid is set to %1, Inter-channel Intensity Difference (IID) parameters will be sent from this point on in the bit stream. If enable_iid==%0, no IID parameters will be sent from this point on in the bit stream.

iid_mode - The configuration of IID parameters (number of bands and quantisation grid, iid_quant) is determined by iid_mode. Eight different configurations for IID parameters are supported (see Table 8.2).

Table 8.2 - IID mode configurations

iid_mode	nr_iid_par_tab	nr_ipdopd_par_tab	iid_quant	Index range
0 (000)	10	5	0	-7…7
1 (001)	20	11		-7…7
2 (010)	34	17		-7…7
3 (011)	10	5	1	-15…15
4 (100)	20	11		-15…15
5 (101)	34	17		-15…15
6 (110)	reserved			
7 (111)	reserved			

If no IID data is sent in the bit-stream, all IID parameters are reset to 0 (i.e. index==0).

The default and the fine quantization grids for IID, iid_quant = %0 and iid_quant = %1 are as provided in Table 8.B.4 and Table 8.B.3 respectively.

Table 8.3 - Default quantization grid for IID.

Index	-7	-6	-5	-4	-3	-2	-1	0
IID [dB]	-25	-18	-14	-10	-7	-4	-2	0
Index	1	2	3	4	5	6	7	
IID [dB]	2	4	7	10	14	18	25	

Table 8.4 - Fine quantization grid for IID.

Index	-15	-14	-13	-12	-11	-10	-9	-8
IID [dB]	-50	-45	-40	-35	-30	-25	-22	-19
Index	-7	-6	-5	-4	-3	-2	-1	0
IID [dB]	-16	-13	-10	-8	-6	-4	-2	0
Index	1	2	3	4	5	6	7	8
IID [dB]	2	4	6	8	10	13	16	19
Index	9	10	11	12	13	14	15	
IID [dB]	22	25	30	35	40	45	50	

Note that the configuration of the Inter-channel Phase Difference (IPD) / Overall Phase Difference (OPD) parameters, is strictly coupled to the IID configuration. This is also illustrated in Table 8.2.

enable_icc – If enable_icc is set to %1, Inter-channel Coherence (ICC) parameters will be sent from this point on in the bit stream. If enable_icc==%0, no ICC parameters will be sent from this point on in the bit stream.

icc_mode – The configuration of Inter-channel Coherence parameters (number of bands and quantisation grid) is determined by icc_mode. Eight different configurations are supported for IC parameters (see Table 8.5).

Table 8.5 - ICC mode configuration

icc_mode	nr_icc_par_tab	Index range	Mixing procedure
0 (000)	10	0…7	R_a
1 (001)	20	0…7	
2 (010)	34	0…7	
3 (011)	10	0…7	R_b
4 (100)	20	0…7	
5 (101)	34	0…7	
6 (110)	reserved		
7 (111)	reserved		

If no ICC data is sent in the bit-stream, all ICC parameters are reset to 1 (i.e. index=0). The default quantization grid for ICC is provided in Table 8.B.5.

Table 8.6 - Quantization grid for ICC.

index	0	1	2	3	4	5	6	7
ρ	1	0.937	0.84118	0.60092	0.36764	0	-0.589	-1

enable_ext – The PS extension layer is enabled using the enable_ext bit. If it's set to %1 the IPD and OPD parameters are sent. If it's disabled, i.e. %0, the extension layer is skipped.

frame_class – The frame_class bit determines whether the parameter positions of the current frame are uniformly spaced accross the frame (FIX_BORDERS: frame_class==%0) or they are defined using the positions described by border_position (VAR_BORDERS: frame_class==%1).

num_env_idx – The number of (sets of) parameters (envelopes) per frame is determined using num_env_idx. In case of fixed parameter spacing (frame_class==%0) and a variable parameter spacing (frame_class==%1) this relation is shown in Table 8.7.

num_env – Local variable denoting the number of stereo envelopes (sets of parameters). num_env==0 signals that no new stereo parameters are transmitted and that the last parameters in the previous ps_data() element shall be kept unchanged and applied to the current ps_data() element.

Table 8.7 - Number of parameter sets num_env as a function of num_env_idx in case of fixed and variable spacing.

	num_env_tab[frame_class][num_env_idx]	
num_env_idx	frame_class==0	frame_class==1
0	0	1
1	1	2
2	2	3
3	4	4

border_position[e] – In case of variable parameter spacing the parameter positions are determined by border_position[e]. It contains the QMF sample index n_e for parameter set e of the current ps_data() element.

iid_dt[e] – This flag describes for envelope index n, whether the IID parameters are coded differentially over time (iid_dt==%1) or over frequency (iid_dt==%0). In the case iid_mode is different from the previous envelope (e-1), iid_dt[e] shall have the value 0% forcing frequency differential coding.

iid_data() – syntactic element containing IID data.

icc_dt[e] – This flag describes for envelope index e, whether the ICC parameters are coded differentially over time (icc_dt==%1) or over frequency (icc_dt==%0). In the case icc_mode is different from the previous envelope (e-1), icc_dt[e] shall have the value 0% forcing frequency differential coding.

icc_data() – syntactic element containing ICC data.

cnt – Local variable denoting the number of bytes used for the ps_extension() element.

ps_extension_size – The length of the PS extension layer is ps_extension_size, measured in bytes. If the extension size makes use of the escape code (ps_extension_size == 15) the length of the extension layer is extended by an additional amount of bytes.

esc_count – In case the escape code (ps_extension_size == 15) is used, esc_count describes the additional length of the PS extension layer measured in bytes.

num_bits_left – Local variable describing the number of bits left for reading in the ps_extension() element.

ps_extension_id – The identification tag (version) of the PS extension layer is given by ps_extension_id. Currently only one version is supported (see Table 8.8).

Table 8.8 - Description of ps_extension_id

ps_extension_id	Version
00 (0)	v0
01 (1)	reserved
10 (2)	reserved
11 (3)	reserved

fill_bits – These fill_bits accpomplish byte-alignment of the ps_extension() data.

enable_ipdopd – The application of IPD and OPD parameters in the bit-stream is denoted by enable_ipdopd. If set (enable_ipdopd==%1) IPD and OPD parameters are sent, if disabled (enable_ipdopd==%0) no IPD and OPD parameters are sent for the current frame in the bit-stream. The quantization grid for both IPD and OPD is provided in Table 8.9. If no IPD or OPD data is sent in the bit-stream, all IPD and OPD parameters are set to 0 (i.e. index=0).

Table 8.9 - Quantization grid for IPD/OPD.

Index	0	1	2	3	4	5	6	7
Representation level	0	$\frac{\pi}{4}$	$\frac{\pi}{2}$	$\frac{3\pi}{4}$	π	$\frac{5}{4}\pi$	$\frac{3}{2}\pi$	$\frac{7}{4}\pi$

ipd_dt[e] – This flag describes for envelope index e, whether the IPD parameters are coded differentially over time (ipd_dt==%1) or over frequency (ipd_dt==%0). In the case iid_mode is different from the previous envelope (e-1), ipd_dt[e] shall have the value 0% forcing frequency differential coding.

ipd_data() – syntactic element containing IPD data.

opd_dt[e] – This flag describes for envelope index e, whether the OPD parameters are coded differentially over time (opd_dt==%1) or over frequency (opd_dt==%0). In the case iid_mode is different from the previous envelope (e-1), opd_dt[e] shall have the value 0% forcing frequency differential coding.

opd_data() – syntactic element containing OPD data.

reserved_ps – This bit is reserved and has a value %0.

iid_par_dt[e,b] – In case of differential coding of IID parameters over time (iid_dt[e]==%1), iid_par_dt[e,b] describes the IID index difference with respect to the b^{th} parameter position for envelope e-1. If no previous parameter is available, iid_par_dt[e,b] represents the IID index difference with respect to the decoded value 0 (i.e. index=0). The IID index iid_par[e,b] is determined as:

$$\mathrm{iid\_par}[e,b]=\mathrm{iid\_par}[e-1,b]+\mathrm{iid\_par\_dt}[e,b]$$

where iid_par[e-1,b] represents the IID index of the previous envelope, e-1. The IID value *iid(b)* is obtained by using iid_par[e,b] as an index to Table 8.3 or Table 8.4, depending on iid_mode.

iid_par_df[e,b] – In case of differential coding of IID parameters over frequency (iid_dt[e]==%0), iid_par_df[e,b] describes the IID difference with respect to the (b-1)th parameter in envelope e. If no previous parameter is available, iid_par_df[e,b] represents the IID difference with respect to the decoded value 0 (i.e. index=0). The IID index iid_par[e,b] is determined as:

$$\mathrm{iid\_par}[e,0]=\mathrm{iid\_par\_df}[e,0]$$
$$\mathrm{iid\_par}[e,b]=\mathrm{iid\_par}[e,b-1]+\mathrm{iid\_par\_df}[e,0] \qquad \text{for} \quad b>0$$

where iid_par[e,b-1] represents the IID index of the previous IID value for envelope e. The IID value *iid(b)* is obtained by using iid_par[e,b] as an index to Table 8.3 or Table 8.4, depending on iid_mode.

icc_par_dt[e,b] – In case of differential coding of ICC parameters over time (icc_dt[e]==%1), icc_par_dt[e,b] describes the difference with respect to the bth parameter position for envelope e-1. If no previous parameter is available, icc_par_dt[e,b] represents the ICC difference with respect to the decoded value 1 (i.e. index=0). The ICC index icc_par[e,b] is determined as:

$$\mathrm{icc\_par}[e,b]=\mathrm{icc\_par}[e-1,b]+\mathrm{icc\_par\_dt}[e,b]$$

where icc_par[e-1,b] represents the ICC index of the previous envelope, e-1. The ICC value *ρ(b)* is obtained by using icc_par[e,b] as an index to Table 8.6.

icc_par_df[e,b] – In case of differential coding of ICC parameters over frequency (icc_dt[e]==%0), icc_par_df[e,b] describes the ICC difference with respect to the (b-1)th parameter for envelope e. If no previous parameter is available, icc_par_df[e,b] represents the ICC difference with respect to the decoded value 1 (i.e. index=0). The ICC index icc_par[e,b] is determined as:

$$\mathrm{icc\_par}[e,0]=\mathrm{icc\_par\_df}[e,0]$$
$$\mathrm{icc\_par}[e,b]=\mathrm{icc\_par}[e,b-1]+\mathrm{icc\_par\_df}[e,0] \qquad \text{for} \quad b>0$$

where icc_par[e,b-1] represents the ICC index of the previous ICC value for envelope e. The ICC value *ρ(b)* is obtained by using icc_par[e,b] as an index to Table 8.6.

ipd_par_dt[e,b] – In case of differential coding of IPD parameters over time (ipd_dt[e]==%1), ipd_par_dt[e,b] describes the IPD difference with respect to the bth parameter position for envelope e. If no previous parameter is available, ipd_par_dt[e,b] represents the IPD difference with respect to the decoded value 0 (i.e. index=0). Note: for IPD parameters modulo-8 differential coding is applied. The IPD index ipd_par[e,b] is determined as:

$$\mathrm{ipd\_par}[e,b]=\mathrm{mod}(\mathrm{ipd\_par}[e-1,b]+\mathrm{ipd\_par\_dt}[e,b],8)$$

where ipd_par[e-1,b] represents the IPD index of the previous envelope, e-1. The IPD value *ipd(b)* is obtained by using ipd_par[e,b] as an index to Table 8.9.

ipd_par_df[e,b] – In case of differential coding of IPD parameters over frequency (ipd_dt[e]==%0), ipd_par_df[e,b] describes the IPD difference with respect to the (b-1)th parameter for envelope e. If no previous parameter is available, ipd_par_df[e,b] represents the IPD difference with respect to the decoded value 0 (i.e. index=0). Note: for IPD parameters modulo-8 differential coding is applied. The IPD index ipd_par[e,b] is determined as:

$$\mathrm{ipd\_par}[e,0]=\mathrm{ipd\_par\_df}[e,0]$$
$$\mathrm{ipd\_par}[e,b]=\mathrm{mod}\left(\mathrm{ipd\_par}[e,b-1]+\mathrm{ipd\_par\_df}[e,0],8\right) \qquad \text{for} \quad b>0$$

where ipd_par[e,b-1] represents the IPD index of the previous IPD value for envelope e. The IPD value *ipd(b)* is obtained by using ipd_par[e,b] as an index to Table 8.6.

opd_par_dt[e,b] – In case of differential coding of OPD parameters over time (opd_dt[e]==%1), opd_par_dt[e,b] describes the OPD difference with respect to the b^{th} parameter position for envelope (e-1). If no previous parameter is available, opd_par_dt[e,b] represents the OPD difference with respect to the decoded value 0 (i.e. index=0). Note: for OPD parameters modulo-8 differential coding is applied. The OPD index opd_par[e,b] is determined as:

$$\mathrm{opd\_par}[e,b] = \mathrm{mod}(\mathrm{opd\_par}[e-1,b] + \mathrm{opd\_par\_dt}[e,b], 8)$$

where opd_par[e-1,b] represents the OPD index of the previous envelope, e-1. The OPD value *opd(b)* is obtained by using opd_par[e,b] as an index to Table 8.9.

opd_par_df[e,b] – In case of differential coding of OPD parameters over time (opd_dt[e]==%0), opd_par_dt[e,b] describes the OPD difference with respect to the $(b\text{-}1)^{th}$ parameter position for envelope e. If no previous parameter is available, opd_par_df[e,b] represents the OPD difference with respect to the decoded value 0 (i.e. index=0). Note: for OPD parameters modulo-8 differential coding is applied. The OPD index opd_par[e,b] is determined as:

$$\mathrm{opd\_par}[e,0] = \mathrm{opd\_par\_df}[e,0]$$
$$\mathrm{opd\_par}[e,b] = \mathrm{mod}\left(\mathrm{opd\_par}[e,b-1] + \mathrm{opd\_par\_df}[e,0], 8\right) \qquad \text{for} \quad b > 0$$

where opd_par[e,b-1] represents the OPD index of the previous OPD value for envelope e. The OPD value *opd(b)* is obtained by using opd_par[e,b] as an index to Table 8.6.

8.6 Decoding processes

The basic parametric stereo decoder is illustrated in Figure 8.1. After de-formatting the bit-stream, the monaural signal M is reconstructed. Subsequently the stereo parameters are used to re-create the left and right signal from the monaural encoded signal.

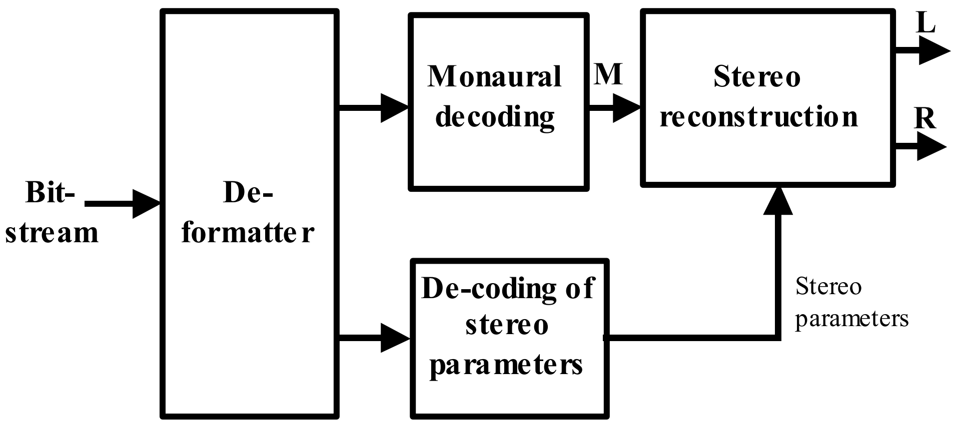

Figure 8.1 - Illustration of the integration of the stereo coding tools into a monaural decoder

8.6.1 Parametric stereo

8.6.1.1 Stereo parameters

Three different types of stereo parameters are used in representing the stereo image. For in total a maximum of 34 bands, one set of stereo parameters is available per band.

1) The inter-channel intensity difference, or IID, defined by the relative levels of the band-limited signal.

2) The inter-channel and overall phase differences, IPD and OPD, defining the phase behaviour of the band-limited signal.

3) The inter-channel Coherence ICC, defining the (dis)similarity of the left and right band-limited signal.

Figure 8.2 diagrams the processing chain of the parametric stereo decoder.

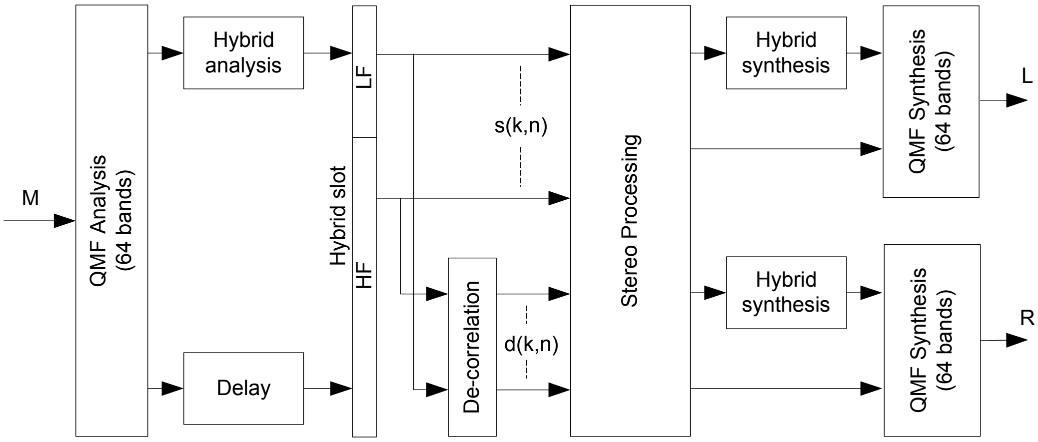

Figure 8.2 - QMF based parametric stereo synthesis

The input to the parametric stereo decoder consists of the monaural parametric generated signal M as obtained through transient, sinusoidal and noise synthesis. The output consists of the left and right stereo representation respectively. In the next paragraphs each block will be detailed.

8.6.1.2 QMF analysis filterbank

This filterbank is identical to the 64 complex QMF analysis filterbank as defined in ISO/IEC 14496-3/AMD1:2003, subclause 4.B.18.2. However, in the equation for matrix M(k,n) and in Figure 4.B.20, the term "(2*n+1)" has to be substituted by "(2*n-1)". The input to the filterbank are blocks of 64 samples of the monaural synthesized signal M. For each block the filterbank outputs one slot of 64 QMF samples.

8.6.1.3 Low frequency filtering

The lower QMF subbands are further split in order to obtain a higher frequency resolution enabling a proper stereo analysis and synthesis for the lower frequencies. Depending on the number of stereo bands, two hybrid configurations have been defined. See Table 8.10 for an overview of the splits and the type of filter that is used to make the split.

Table 8.10 - Overview of low frequency split for the available configurations.

Configuration, number of stereo bands	QMF subband p	Number of bands Q^p	Filter
10, 20	0	8	Type A
	1	2	Type B
	2	2	
34	0	12	Type A
	1	8	
	2	4	
	3	4	
	4	4	

$$TypeA: G_q^p = g^p(n)\exp\left(j\frac{2\pi}{Q^p}\left(q+\frac{1}{2}\right)(n-6)\right),$$

$$TypeB: G_q^p = g^p(n)\cos\left(\frac{2\pi}{Q^p}q(n-6)\right),$$

where g^p represents the prototype filters in QMF subband p. Q^p represents the number of sub-subbands in QMF subband p, q the sub-subband index in QMF channel p and n the time index. The prototype filters are all of length 13 and have a delay of 6 QMF samples. The prototype filters are listed in Table 8.11 and Table 8.12 for the 10,20 and the 34 stereo bands configuration respectively.

Table 8.11 - Prototype filter coefficients for the filters that split the lower QMF subbands for the 10 and 20 stereo bands configuration.

n	$g^0(n)$, Q^0 =8	$g^{1,2}(n)$, $Q^{1,2}$ =2
0	0.00746082949812	0
1	0.02270420949825	0.01899487526049
2	0.04546865930473	0
3	0.07266113929591	-0.07293139167538
4	0.09885108575264	0
5	0.11793710567217	0.30596630545168
6	0.125	0.5
7	0.11793710567217	0.30596630545168
8	0.09885108575264	0
9	0.07266113929591	-0.07293139167538
10	0.04546865930473	0
11	0.02270420949825	0.01899487526049
12	0.00746082949812	0

Table 8.12 - Prototype filter coefficients for the filters that split the lower QMF subbands for the 34 stereo bands configuration.

n	$g^0(n)$, Q^0 =12	$g^1(n)$, Q^1 =8	$g^{2,3,4}$, $Q^{2,3,4}$ =4
0	0.04081179924692	0.01565675600122	-0.05908211155639
1	0.03812810994926	0.03752716391991	-0.04871498374946
2	0.05144908135699	0.05417891378782	0
3	0.06399831151592	0.08417044116767	0.07778723915851
4	0.07428313801106	0.10307344158036	0.16486303567403
5	0.08100347892914	0.12222452249753	0.23279856662996
6	0.08333333333333	0.12500000000000	0.25000000000000
7	0.08100347892914	0.12222452249753	0.23279856662996
8	0.07428313801106	0.10307344158036	0.16486303567403
9	0.06399831151592	0.08417044116767	0.07778723915851
10	0.05144908135699	0.05417891378782	0
11	0.03812810994926	0.03752716391991	-0.04871498374946
12	0.04081179924692	0.01565675600122	-0.05908211155639

Figure 8.3 and Figure 8.4 illustrated the hybrid analysis and synthesis filterbank for the 10 and 20 stereo bands configuration respectively. Figure 8.5 and Figure 8.6 illustrated the hybrid analysis and synthesis filterbank for the 34 stereo bands configuration respectively. Note that for the 10 and 20 stereo bands configuration, sub-subbands have been combined into a single sub-subband.

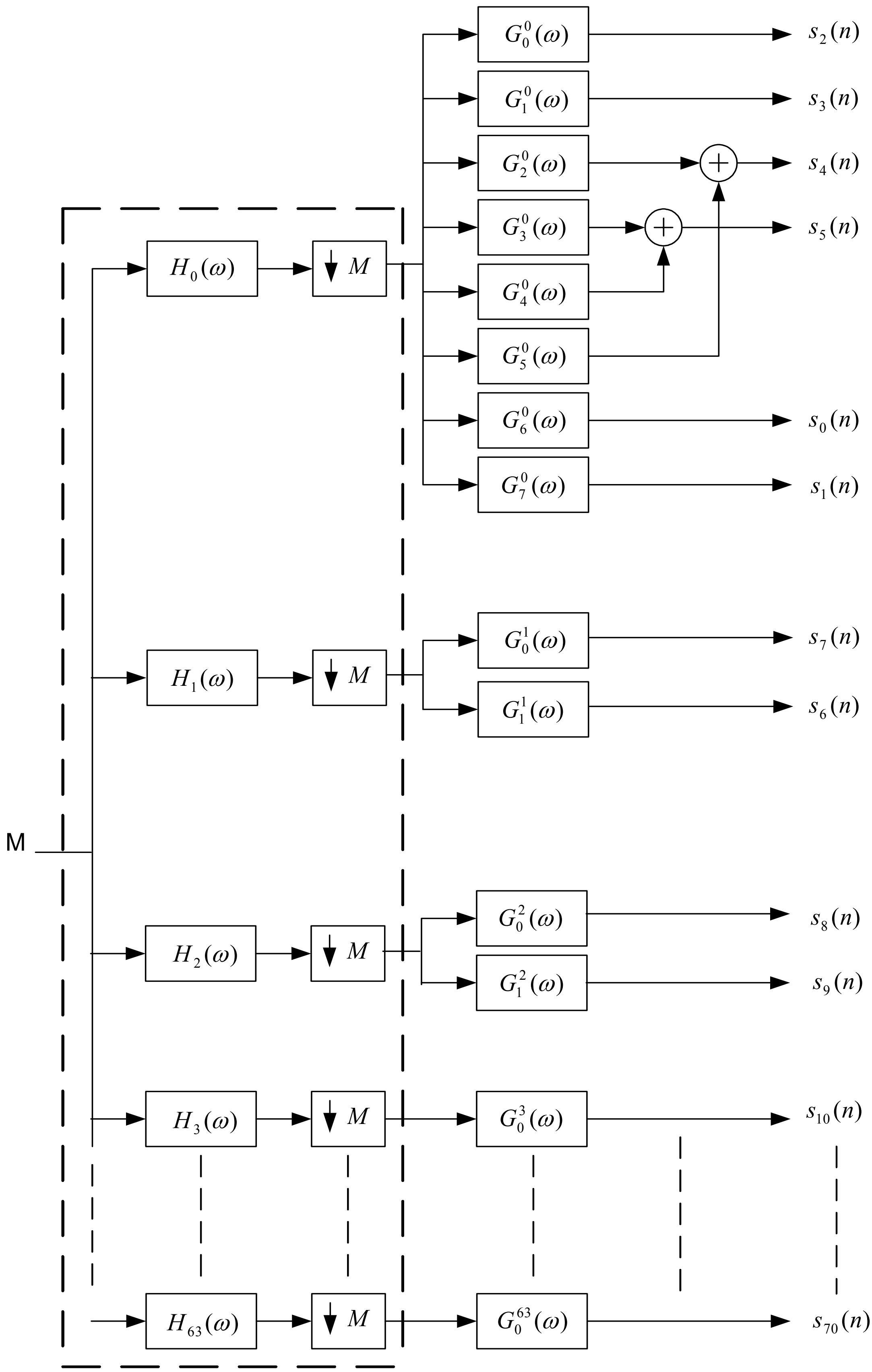

Figure 8.3 - Hybrid QMF analysis filterbank for the 10 and 20 stereo-bands configuration. The lower subbands of the 64 QMF (see dashed box) are further split to provide for increased resolution for the lower frequencies

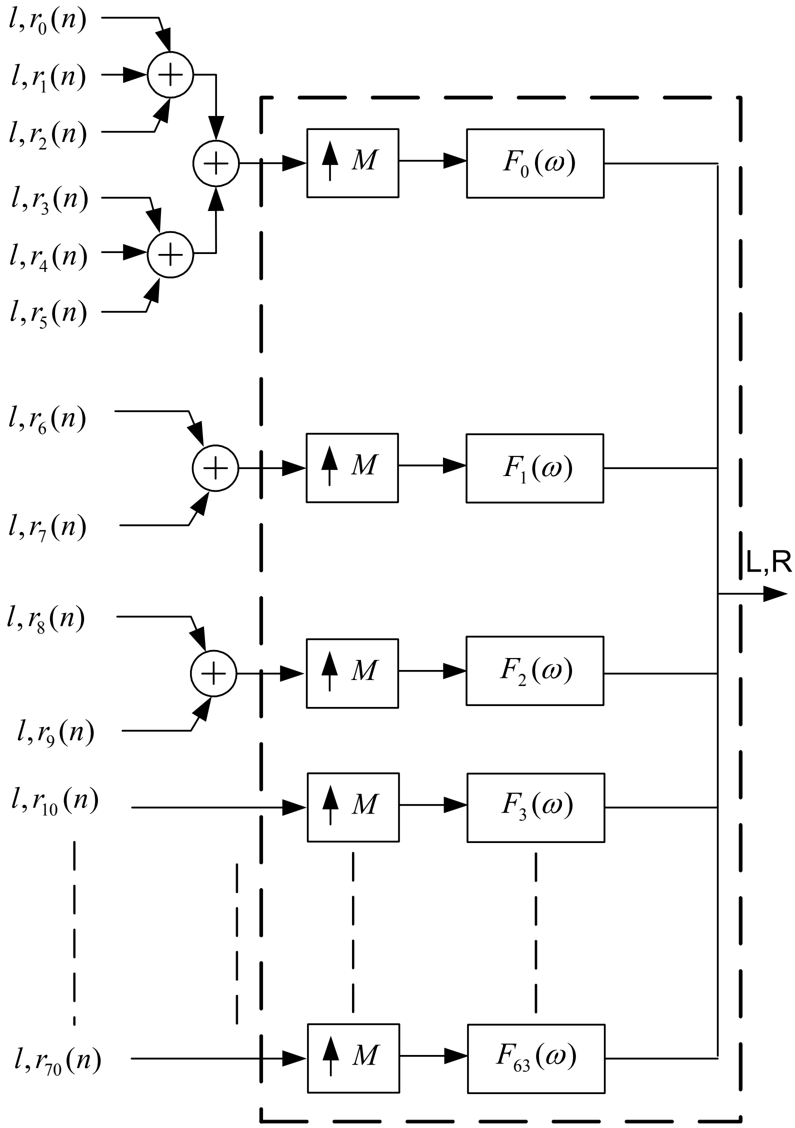

Figure 8.4 - Hybrid QMF synthesis filterbank for the 10 and 20 stereo-bands configuration. The coefficients offering higher resolution for the lower QMF channel are simply added prior to the synthesis with the 64 subbands QMF (see dashed box)

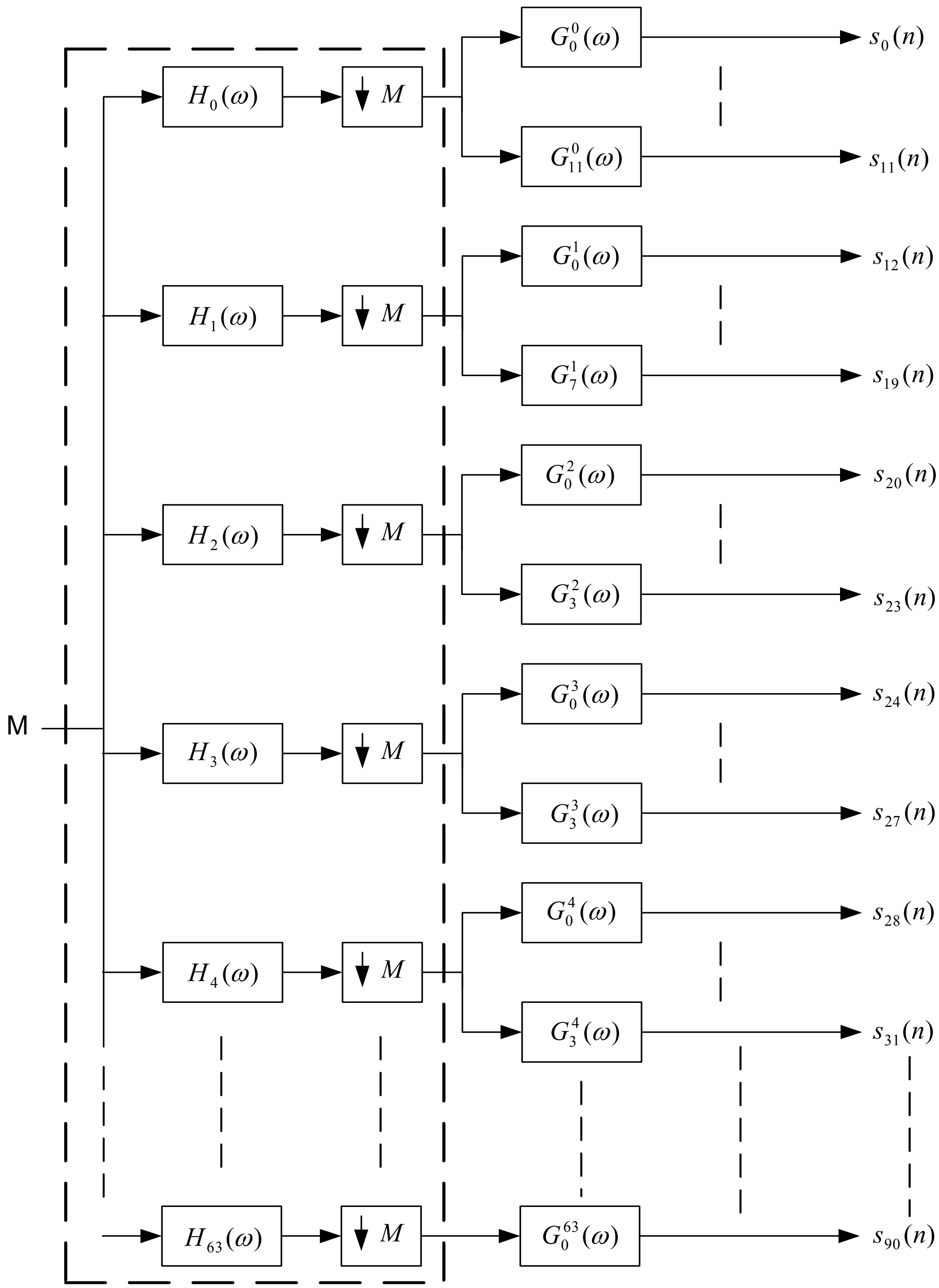

Figure 8.5 - Hybrid QMF analysis filterbank for the 34 stereo-bands configuration. The lower subbands of the 64 QMF (see dashed box) are further split to provide for increased resolution for the lower frequencies

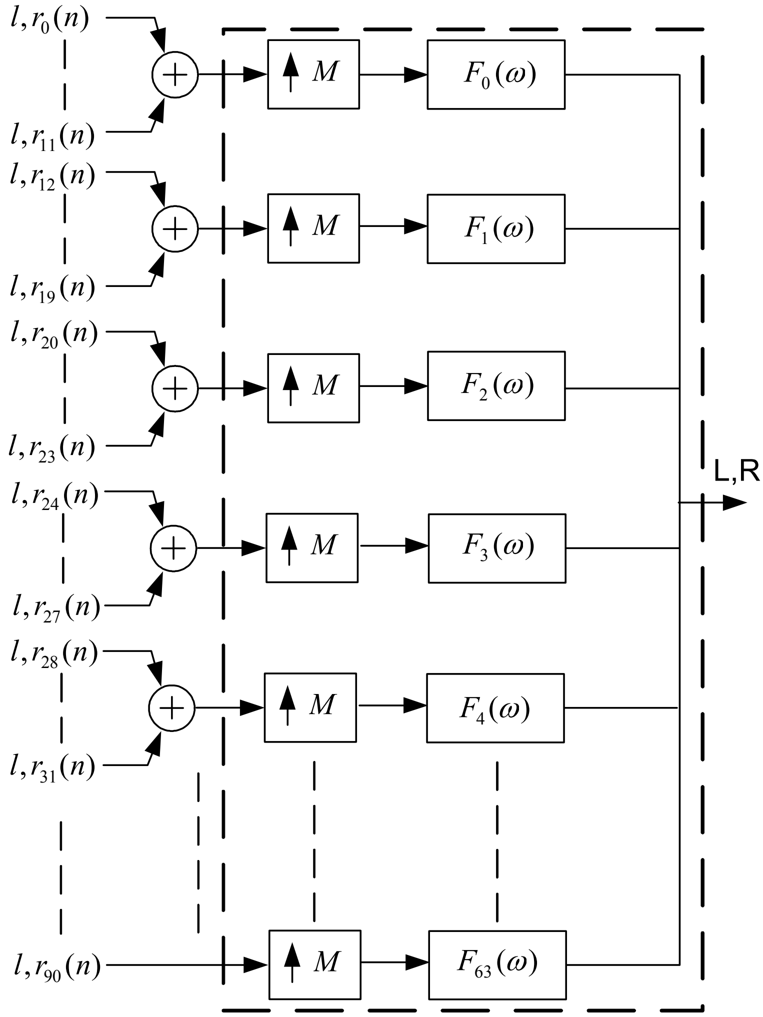

Figure 8.6 - Hybrid QMF synthesis filterbank for the 34 stereo-bands configuration. The coefficients offering higher resolution for the lower QMF subbands are simply added prior to the synthesis with the 64 subbands QMF (see dashed box)

In order to time align all the samples originating from the hybrid filterbank, the remaining QMF subbands that have not been filtered are delay compensated. This delay amounts to 6 QMF subband samples. This means $G_0^k(z) = z^{-6}$ for k=3...63 (10,20 stereo bands) or k=5...63 (34 stereo bands). In order to compensate for the overall delay of the hybrid analysis filterbank, the first 10 sets (6 from delay and 4 from QMF filter) of hybrid subbands are flushed and therefore not taken into account for processing.

The resultant of this operation is a slot of hybrid subband samples consisting of a LF (low frequency) sub QMF subband portion and HF (high frequency) QMF subband portion. As an illustration, see the hybrid slot in Figure 8.7.

8.6.1.4 Framing

Stereo parameters within a stereo frame can be assigned to one or more slots. The stereo frame boundaries and the positions n_e of the slots that have been assigned stereo parameters to, define so called regions. Stereo parameters are defined for the last slot in a region. By means of these regions transients can be effectively handled. As illustrated by the bold solid lines in Figure 8.7, stereo parameters for non-assigned slots are obtained by means of interpolation. For the very first region (region_0), interpolation is performed with respect to the default parameters for index=0 (i.e. IID=0, ICC=1 and IPD=OPD=0). Any existing slots after the last assigned slot in the stereo frame obtain the same stereo parameters from this last assigned slot (region_2). At stereo-frame borders (region_3), interpolation for the first region is performed with respect to the last slot in the previous frame. The stereo reconstruction is applied per region. For simplicity, the envelope index is only indicated when applicable.

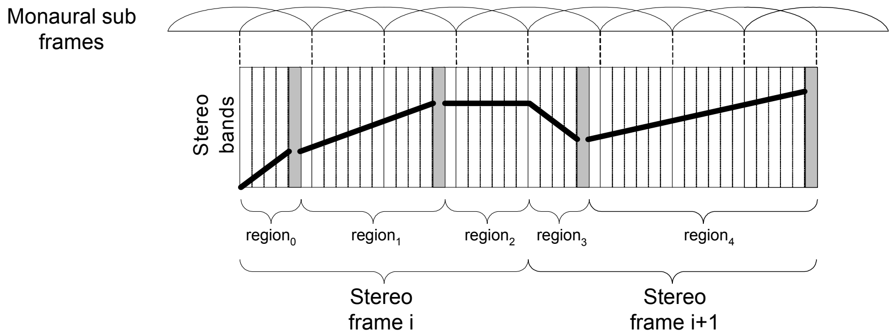

Figure 8.7 – Illustration of stereo frames of data. The solid line illustrates the interpolation between stereo parameters for slots that have not been assigned stereo parameters to

8.6.1.5 De-correlation

By means of all-pass filtering and delaying, the sub subband samples $s_k(n)$ are converted into de-correlated sub subband samples $d_k(n)$, where k represents the frequency in the hybrid spectrum and n the time index.

8.6.1.5.1 constants:

F_s is the output sampling rate.

$DECAY\_SLOPE = 0.05$ is the all-pass filter decay slope.

$NR\_ALLPASS\_LINKS = 3$ is the number of filter links for the all-pass filter.

NR_PAR_BANDS is the number of frequency bands that can be addressed by the parameter index, b(k) (see Table 8.22 and Table 8.23).

$$NR\_PAR\_BANDS = \begin{cases} 20 & ,10 \text{ or } 20 \text{ stereo bands} \\ 34 & ,34 \text{ stereo bands} \end{cases}$$

NR_BANDS is the number of frequency bands that can be addressed by the sub subband index, k.

$$NR\_BANDS = \begin{cases} 71 & ,10 \text{ or } 20 \text{ stereo bands} \\ 91 & ,34 \text{ stereo bands} \end{cases}$$

$DECAY\_CUTOFF$ is the start frequency band for the all-pass filter decay slope.

$$DECAY\_CUTOFF = \begin{cases} 10 & ,10 \text{ or } 20 \text{ stereo bands} \\ 32 & ,34 \text{ stereo bands} \end{cases}$$

$NR\_ALLPASS\_BANDS$ is the number of all-pass filter bands.

$$NR\_ALLPASS\_BANDS = \begin{cases} 53 & ,F_s < 32kHz, 10 \text{ } or \text{ } 20 \text{ } stereo \text{ } bands \\ 73 & ,F_s < 32kHz, 34 \text{ } stereo \text{ } bands \\ 30 & ,F_s \geq 32kHz, 10 \text{ } or \text{ } 20 \text{ } stereo \text{ } bands \\ 50 & ,F_s \geq 32kHz, 34 \text{ } stereo \text{ } bands \end{cases}$$

$SHORT\_DELAY\_BAND$ is the first stereo band using the short, one sample delay.

$$SHORT\_DELAY\_BAND = \begin{cases} 71 & ,F_s < 32kHz, 10 \text{ } or \text{ } 20 \text{ } stereo \text{ } bands \\ 91 & ,F_s < 32kHz, 34 \text{ } stereo \text{ } bands \\ 42 & ,F_s \geq 32kHz, 10 \text{ } or \text{ } 20 \text{ } stereo \text{ } bands \\ 62 & ,F_s \geq 32kHz, 34 \text{ } stereo \text{ } bands \end{cases}$$

$a_{Smooth} = \begin{cases} 0.6 & ,F_s < 32kHz \\ 0.25 & ,F_s \geq 32kHz \end{cases}$ is the smoothing coefficient.

8.6.1.5.2 Calculate decorrelated signal, $d_k(z)$

The decorrelation process for the first $NR\_ALLPASS\_BANDS$ frequency bands of $s_k(n)$ is based on an all-pass filter described in the Z-domain below. Its transfer function for each band, k is defined by:

$$\mathbf{H}_k(z) = z^{-2} \cdot \varphi_{Fract}(k) \cdot \prod_{m=0}^{NR\_ALLPASS\_LINKS-1} \frac{\mathbf{Q}_{Fract\_allpass}(k,m) z^{-\mathbf{d}(m)} - \mathbf{a}(m)\mathbf{g}_{DecaySlope}(k)}{1 - \mathbf{a}(m)\mathbf{g}_{DecaySlope}(k)\mathbf{Q}_{Fract\_allpass}(k,m) z^{-\mathbf{d}(m)}} \quad ,$$

for $0 \leq k < NR\_ALLPASS\_BANDS$.

The fractional delay length matrix, $\mathbf{Q}_{Fract\_allpass}(k,m)$ and the fractional delay vector, $\varphi_{Fract}(k)$ are defined, by:

$$\mathbf{Q}_{Fract\_allpass}(k,m) = \exp\left(-i\pi \mathbf{q}(m) \mathbf{f}_{center}(k)\right) \quad , \begin{cases} 0 \leq k < NR\_ALLPASS\_BANDS \\ 0 \leq m < NUM\_OF\_LINKS \end{cases} ,$$

and

$$\varphi_{Fract}(k) = \exp\left(-i\pi q_{\varphi} \mathbf{f}_{center}(k)\right) \quad , 0 \leq k < NR\_ALLPASS\_BANDS ,$$

where $i = \sqrt{-1}$ denotes the imaginary unit. For the frequency vector, $\mathbf{f}_{center}$, the fractional delay length vector, see Table 8.14 and Table 8.15. The vector $\mathbf{q}(k)$ is defined in Table 8.16. The fractional delay length constant, $q_{\varphi} = 0.39$.

For the filter coefficient vector $\mathbf{a}(m)$ and the delay length vector $\mathbf{d}(m)$ see Table 8.13.

The vector $\mathbf{g}_{DecaySlope}$ contains time invariant factors for making the all-pass filter frequency variant. It is defined by:

$$\mathbf{g}_{DecaySlope}(k) = \begin{cases} \max\langle 0, 1 - DECAY\_SLOPE \cdot (k - DECAY\_CUTOFF)\rangle & , k > DECAY\_CUTOFF \\ 1 & , otherwise \end{cases}$$

for $0 \le k < NR\_ALLPASS\_BANDS$.

For the upper bands, i.e. for $NR\_ALLPASS\_BANDS \le k < NR\_BANDS$ the transfer function, $\mathbf{H}_k(z)$ equals a delay according to:

$\mathbf{H}_k(z) = z^{-D(k)}$, where $D(k)$ is defined by:

$$D(k) = \begin{cases} 14 & , NR\_ALLPASS\_BANDS \le k < SHORT\_DELAY\_BAND \\ 1 & , SHORT\_DELAY\_BAND \le k < NR\_BANDS \end{cases}.$$

8.6.1.5.3 Perform transient detection

To be able to handle transients and other fast time-envelopes, the all-pass filter has to be attenuated at those signals. It is done by the following scheme:

First define the input power matrix, $\mathbf{P}(i,n)$ that contains the sum of the squared sub subband samples of each parameter band. As indicated in Figure 8.8, *n* runs from n_0 to n_{L-1}.

$$\mathbf{P}(i,n) = \sum_{i=b(k)} |s_k(n)|^2 \quad , 0 \le i < NR\_PAR\_BANDS,$$

where b(k) is defined in Table 8.22 and Table 8.23. Apply peak decay on the input power signal according to:

$$\mathrm{P}_{PeakDecayNrg}(i,n) = \begin{cases} \mathrm{P}(i,n) & , \alpha \mathrm{P}_{PeakDecayNrg}(i,n-1) < \mathrm{P}(i,n) \\ \alpha \mathrm{P}_{PeakDecayNrg}(i,n-1) & , otherwise \end{cases}.$$

for $0 \le i < NR\_PAR\_BANDS$. α is the peak decay factor defined in Table 8.17.

Subsequently, filter the input power and peak decay power signals with the Z-domain transfer function,

$H_{Smooth}(z)$:

$$\mathbf{P}_{SmoothNrg}(i,z) = H_{Smooth}(z) \cdot \mathbf{P}(i,z),$$

$$\mathbf{P}_{SmoothPeakDecayDiffNrg}(i,z) = H_{Smooth}(z) \cdot (\mathbf{P}_{PeakDecayNrg}(i,z) - \mathbf{P}(i,z)),$$

for $0 \le i < NR\_PAR\_BANDS$, where

$$H_{Smooth}(z) = \frac{a_{Smooth}}{1+(a_{Smooth}-1)\cdot z^{-1}}.$$

The transient attenuator, $\mathbf{G}_{TransientRatio}$ is then calculated as follows:

$$\mathbf{G}_{TransientRatio}(i,n) = \begin{cases} \dfrac{\mathbf{P}(i,n)}{\gamma \cdot \mathbf{P}_{SmoothPeakDecayDiffNrg}(i,n)} & , \gamma \cdot \mathbf{P}_{SmoothPeakDecayDiffNrg}(i,n) > \mathbf{P}(i,n) \\ 1 & , otherwise \end{cases},$$

for $0 \le i < NR\_PAR\_BANDS$, where $\gamma = 1.5$ is a transient impact factor.

Finally map the transient attenuator, $\mathbf{G}_{TransientRatio}$ to bands according to:

$$\mathbf{G}_{TransientRatioMapped}(k,n) = \mathbf{G}_{TransientRatio}(b(k),n) \quad ,0 \le k < NR\_BANDS.$$

8.6.1.5.4 Apply transient reduction to decorrelated signal

Let $d_k(z)$ be the decorrelated signal and $s_k(z)$ the mono input signal in the Z-domain for each band. Then $d_k(z)$ is defined according to:

$$d_k(z) = \mathbf{G}_{TransientRatioMapped}(k,z)\cdot \mathbf{H}_k(z)\cdot s_k(z) \text{, where } 0 \le k < NR\_BANDS.$$

Table 8.13 - Filter coefficient vector, delay length vectors $\mathbf{d}_{24kHz}(m)$ and $\mathbf{d}_{48kHz}(m)$.

m	$\mathbf{a}(m)$	$\mathbf{d}_{24kHz}(m)$	$\mathbf{d}_{48kHz}(m)$
0	0.65143905753106	1	3
1	0.56471812200776	2	4
2	0.48954165955695	3	5

Delay length vector, $\mathbf{d} = \begin{cases} \mathbf{d}_{24kHz} & ,F_s < 32kHz \\ \mathbf{d}_{48kHz} & ,F_s \ge 32kHz \end{cases}$.

Table 8.14 - Delay length vector $\mathbf{f}_{center\_20}$.

k	$\mathbf{f}_{center\_20}(k)$	k	$\mathbf{f}_{center\_20}(k)$
0	-3/8	5	7/8
1	-1/8	6	2.5/2
2	1/8	7	3.5/2
3	3/8	8	4.5/2
4	5/8	9	5.5/2

Table 8.15 - Delay length vector $\mathbf{f}_{center\_34}$.

k	$\mathbf{f}_{center\_34}(k)$	k	$\mathbf{f}_{center\_34}(k)$
0	1/12	16	9/8
1	3/12	17	11/8
2	5/12	18	13/8
3	7/12	19	15/8
4	9/12	20	9/4
5	11/12	21	11/4
6	13/12	22	13/4
7	15/12	23	7/4
8	17/12	24	17/4
9	-5/12	25	11/4
10	-3/12	26	13/4
11	-1/12	27	15/4
12	17/8	28	17/4
13	19/8	29	19/4
14	5/8	30	21/4
15	7/8	31	15/4

$$\mathbf{f}_{center\_20}(k) = k + \frac{1}{2} - 7 \quad ,10 \le k < NR\_ALLPASS\_BANDS,$$

$$\mathbf{f}_{center\_34}(k) = k + \frac{1}{2} - 27 \quad ,32 \le k < NR\_ALLPASS\_BANDS,$$

$$\mathbf{f}_{center} = \begin{cases} \mathbf{f}_{center\_20} & ,NR\_PAR\_BANDS = 20 \\ \mathbf{f}_{center\_34} & ,NR\_PAR\_BANDS = 34 \end{cases}.$$

Table 8.16 - Fractional delay length vector $\mathbf{q}(m)$.

m	$\mathbf{q}(m)$
0	0.43
1	0.75
2	0.347

Table 8.17 - Peak Decay Factors $\alpha_{Decay24kHz}$ and $\alpha_{Decay48kHz}$.

$\alpha_{Decay24kHz}$	$\alpha_{Decay48kHz}$
0.58664621951003	0.76592833836465

Peak decay factor, $\alpha = \begin{cases} \alpha_{Decay24kHz} & ,F_s < 32kHz \\ \alpha_{Decay48kHz} & ,F_s \geq 32kHz \end{cases}$.

8.6.1.6 Stereo Processing

The sets of sub subband samples $s_k(n)$ and $d_k(n)$ are now processed according to the stereo cues. These cues are defined per stereo band. All the hybrid subband samples within a stereo band are processed according to the cues in that respective stereo band. Table 8.22 and Table 8.23 indicate the hybrid subband samples that fall into each stereoband for the (10,20) and 34 stereoband configuration. *k* traverses the range from 0…70 or 0…90 for the (10,20) or 34 stereoband configuration respectively.

8.6.1.6.1 Mapping

The number of stereo bands that is actually used for the processing of the cues depends on the number of available parameters for IID and ICC according to the relation given in Table 8.18.

Table 8.18 - The number of stereo bands depends on the number of parameters for IID and ICC.

Number of IID parameters	number of ICC parameters	number of stereo bands
10	10	20 (i.e., 10,20 stereo band configuration)
10	20	
20	10	
20	20	
10,20	34	34
34	10,20	

In the case the number of parameters for IID and ICC differs from the number of stereo bands, as dictated in Table 8.18, a mapping from the lower number to the higher number of parameters is required. For the mapping from 10 to 20 parameters this is realised by duplicating every parameter as shown in Table 8.19. For the mapping from 20 to 34 parameters this is realized according to Table 8.19. For the mapping from 10 to 34 parameters, the 10 parameters are first mapped to 20 parameters and subsequently to 34 parameters. Table 8.20 provides the inverse mapping from 34 to 20 parameters.

Table 8.19 - Mapping from 10 to 20 to 34 parameters.

parameter grid			parameter grid		
34	20	10	34	20	10
idx_0	idx_0	idx_0	idx_{17}	idx_{11}	idx_5
idx_1	$(idx_0+idx_1)/2$	idx_0	idx_{18}	idx_{12}	idx_6
idx_2	idx_1	idx_0	idx_{19}	idx_{13}	idx_6
idx_3	idx_2	idx_1	idx_{20}	idx_{14}	idx_7
idx_4	$(idx_2+idx_3)/2$	idx_1	idx_{21}	idx_{14}	idx_7
idx_5	idx_3	idx_1	idx_{22}	idx_{15}	idx_7
idx_6	idx_4	idx_2	idx_{23}	idx_{15}	idx_7
idx_7	idx_4	idx_2	idx_{24}	idx_{16}	idx_8
idx_8	idx_5	idx_2	idx_{25}	idx_{16}	idx_8
idx_9	idx_5	idx_2	idx_{26}	idx_{17}	idx_8
idx_{10}	idx_6	idx_3	idx_{27}	idx_{17}	idx_8
idx_{11}	idx_7	idx_3	idx_{28}	idx_{18}	idx_9
idx_{12}	idx_8	idx_4	idx_{29}	idx_{18}	idx_9
idx_{13}	idx_8	idx_4	idx_{30}	idx_{18}	idx_9
idx_{14}	idx_9	idx_4	idx_{31}	idx_{18}	idx_9
idx_{15}	idx_9	idx_4	idx_{32}	idx_{19}	idx_9
idx_{16}	idx_{10}	idx_5	idx_{33}	idx_{19}	idx_9

Table 8.20 – Mapping of IID, ICC, IPD and OPD parameters from 34 stereo bands to 20 stereo bands. For IPD and OPD parameters, this mapping applies up to and including idx_{10} and idx_{16} for 20 and 34 stereo bands respectively

20 stereo bands	34 stereo bands
idx_0	$(2*idx_0+idx_1)/3$
idx_1	$(idx_1+2*idx_2)/3$
idx_2	$(2*idx_3+idx_4)/3$
idx_3	$(idx_4+2*idx_5)/3$
idx_4	$(idx_6+idx_7)/2$
idx_5	$(idx_8+idx_9)/2$
idx_6	idx_{10}
idx_7	idx_{11}
idx_8	$(idx_{12}+idx_{13})/2$
idx_9	$(idx_{14}+idx_{15})/2$
idx_{10}	idx_{16}
idx_{11}	idx_{17}
idx_{12}	idx_{18}
idx_{13}	idx_{19}
idx_{14}	$(idx_{20}+idx_{21})/2$
idx_{15}	$(idx_{22}+idx_{23})/2$
idx_{16}	$(idx_{24}+idx_{25})/2$
idx_{17}	$(idx_{26}+idx_{27})/2$
idx_{18}	$(idx_{28}+idx_{29}+idx_{30}+idx_{31})/4$
idx_{19}	$(idx_{32}+idx_{33})/2$

The averaging process denoted by e.g. $(2*idx_0 + idx_1)/2$ in Table 8.19 and Table 8.20 is carried out for the integer index representation idx_k of the IID or ICC parameters prior to dequantization, according to ANSI-C integer arithmetic.

The IPD/OPD parameters follow the mapping for the IID parameters, taking into account the relative amount of parameters for IPD/OPD as given by Table 8.2. Consequently the same mapping as for IID is applied, but only for a lower number of parameters for IPD/OPD. For the upper stereo bands, where no IPD/OPD data is transmitted, the IPD/OPD parameters are set to zero.

If the number of stereo bands changes from 10,20 in the previous frame to 34 in the current frame, the stereo parameters from the previous frame are mapped to 34 stereo bands according to Table 8.19 prior to further processing of the current. The frequency resolution of the hybrid QMF analysis filterbank (see subclause 8.6.1.3) is changed instantaneously to the 34 stereo band configuration. The state variable of the decorrelation process are mapped from *NR_BANDS*==71 to *NR_BANDS*==91 configuration according to Table 8.21. The state variables for sub subbands in *NR_BANDS*==91 configuration not listed in Table 8.21 are reset to zero.

If the number of stereo bands changes from 34 in the previous frame to 10,20 in the current frame, the stereo parameters from the previous frame are mapped to 20 stereo bands according to Table 8.20 prior to further processing of the current. The frequency resolution of the hybrid QMF analysis filterbank (see subclause 8.6.1.3) is changed instantaneously to the 20 stereo band configuration. The state variable of the decorrelation process are mapped from *NR_BANDS*==91 to *NR_BANDS*==71 configuration according to Table 8.21. The state variables for sub subbands in *NR_BANDS*==91 configuration not listed in Table 8.21 are discarded.

Table 8.21 - Mapping of state variables of decorrelation process between 10,20 and 34 stereo band configurations

NR_BANDS==91	*NR_BANDS*==71	
Sub subband index *k*	Sub subband index *k*	
9	0	Sub QMF
11	1	
0	2	
2	3	
3	4	
5	5	
16	6	
18	7	
20	8	
21	9	
26	10	
28	11	
32	12	QMF (only)
[33, 89]	[13, 69]	
90	70	

8.6.1.6.2 Mixing

In order to generate the QMF subband signals for the subband samples $n = n_e + 1 \ldots n_{e+1}$ the parameters at position n_e and n_{e+1} are required as well as the subband domain signals $s_k(n)$ and $d_k(n)$ for $n = n_e + 1 \ldots n_{e+1}$ (see Figure 8.8). For IPD/OPD, the parameters at position n_{e-1} are needed in addition. n_e represents the start position for envelope e. In the case frameclass == %1 (VAR_BORDERS), the border positions n_e are obtained by borderposition[e]. In the case frameclass == %0 (FIX_BORDERS), the border positions n_e are obtained by means of

$$n_e = \left\lfloor \frac{numQMFSlots * (e+1)}{num\_env} \right\rfloor - 1, \qquad e = [0, \ldots, num\_env - 1].$$

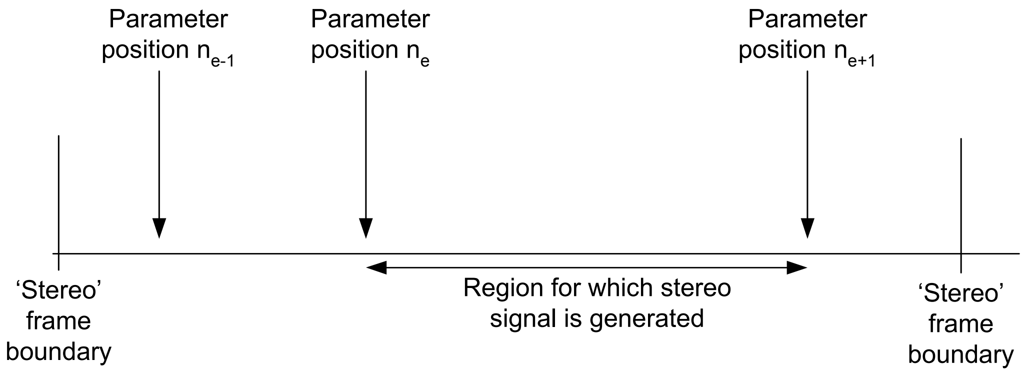

Figure 8.8 - Region for which the stereo subband signals are generated

The stereo sub subband signals are constructed as:

$$
\begin{aligned}
l_k(n) &= H_{11}(k,n)s_k(n) + H_{21}(k,n)d_k(n) \\
r_k(n) &= H_{12}(k,n)s_k(n) + H_{22}(k,n)d_k(n)
\end{aligned}.
$$

In order to obtain the matrices $H_{11}(k,n)$, $H_{12}(k,n)$, $H_{21}(k,n)$, $H_{22}(k,n)$, the vectors $h_{11}(b)$, $h_{12}(b)$, $h_{21}(b)$, $h_{22}(b)$ need to be calculated first, where the parameter *b* is used as parameter index. First for the parameter position n_{e+1} the intensity differences (IID) are transformed to the linear domain.

$$
c(b) = 10^{\frac{iid(b)}{20}},
$$

where *iid*(*b*) represents the decoded IID value for stereo band *b* in dB. Depending on the ICC mode configuration, either mixing procedure R_a or R_b is used, see Table 8.5. For both mixing procedures, the parameters for parameter position n_{e+1} are used.

8.6.1.6.2.1 Mixing procedure R_a

In the case mixing procedure R_a is used the following method is applied.

From the intensity differences two scale-factor vectors c_1 and c_2 are calculated.

$$
\begin{aligned}
c_1(b) &= \frac{\sqrt{2}}{\sqrt{1+c^2(b)}} \\
c_2(b) &= \frac{\sqrt{2}c(b)}{\sqrt{1+c^2(b)}}
\end{aligned}.
$$

From these and the ICC parameter $\rho(b)$, the coefficients $h_{xy}(b)$ are calculated according to

$$
\begin{aligned}
\alpha(b) &= \frac{1}{2}\arccos(\rho(b)) \\
\beta(b) &= \alpha(b)\frac{c_1(b)-c_2(b)}{\sqrt{2}} \\
h_{11}(b) &= \cos(\alpha(b)+\beta(b))c_2(b) \\
h_{12}(b) &= \cos(\beta(b)-\alpha(b))c_1(b) \\
h_{21}(b) &= \sin(\alpha(b)+\beta(b))c_2(b) \\
h_{22}(b) &= \sin(\beta(b)-\alpha(b))c_1(b)
\end{aligned}.
$$

8.6.1.6.2.2 Mixing procedure R_b

In the case mixing procedure R_b is used the following method is applied.

In order to prevent instabilities, in the case the value of $\rho(b)$ is smaller than 0.05, $\rho(b)$ is set to 0.05. In the case *c*(*b*) is not equal to 1

$$
\alpha(b) = \frac{1}{2}\arctan\left(\frac{2c(b)\rho(b)}{c^2(b)-1}\right),
$$

otherwise $\alpha(b)=\frac{\pi}{4}$. After modulo correction of $\alpha(b)$, again the value of $c(b)$ and $\rho(b)$ are used to derive the coefficients $h_{xy}(b)$.

$$
\begin{aligned}
&\alpha(b)=\alpha(b)-\left\lfloor\frac{\alpha(b)}{\frac{1}{2}\pi}\right\rfloor\frac{1}{2}\pi,\\
&\mu(b)=1+\frac{4\rho^2(b)-4}{(c(b)+c^{-1}(b))^2},\\
&\gamma(b)=\arctan(\sqrt{\frac{1-\sqrt{\mu(b)}}{1+\sqrt{\mu(b)}}}),\\
&h_{11}(b)=\sqrt{2}\cos(\alpha(b))\cos(\gamma(b)),\ .\\
&h_{12}(b)=\sqrt{2}\sin(\alpha(b))\cos(\gamma(b)),\\
&h_{21}(b)=-\sqrt{2}\sin(\alpha(b))\sin(\gamma(b)),\\
&h_{22}(b)=\sqrt{2}\cos(\alpha(b))\sin(\gamma(b)).
\end{aligned}
$$

8.6.1.6.3 Phase parameters

8.6.1.6.3.1 Phase parameters disabled

In the case IPD and OPD are disabled as indicated by (**enable_ipdopd**==0) the following procedure is applied.

In order to obtain $H_{11}(k,n_{e+1})$, $H_{12}(k,n_{e+1})$, $H_{21}(k,n_{e+1})$ and $H_{22}(k,n_{e+1})$ we use the following equations

$$
\begin{aligned}
H_{11}(k,n_{e+1})&=h_{11}(b(k))\\
H_{12}(k,n_{e+1})&=h_{12}(b(k))\\
H_{21}(k,n_{e+1})&=h_{21}(b(k))\\
H_{22}(k,n_{e+1})&=h_{22}(b(k))
\end{aligned},
$$

where $b(k)$ is defined in Table 8.22 and Table 8.23.

8.6.1.6.3.2 Phase parameters enabled

In the case IPD and OPD are enabled as indicated by (enable_ipdopd==1) the following procedure is applied.

First the IPD and OPD values are smoothed over time according to

$$
\begin{aligned}
\varphi_{opd}(b)&=\angle\left\{\frac{1}{4}\exp(j\cdot opd(b,n_{e-1}))+\frac{1}{2}\exp(j\cdot opd(b,n_e))+\exp(j\cdot opd(b,n_{e+1}))\right\}\\
\varphi_{ipd}(b)&=\angle\left\{\frac{1}{4}\exp(j\cdot ipd(b,n_{e-1}))+\frac{1}{2}\exp(j\cdot ipd(b,n_e))+\exp(j\cdot ipd(b,n_{e+1}))\right\}
\end{aligned}.
$$

In the case the number of IPD/OPD parameters for parameter position n_{e-1} and/or n_e are different from the number of IPD/OPD parameters for parameter position n_{e+1}, these are mapped to the number of IPD/OPD parameters for parameter position n_{e+1} using Table 8.19 and Table 8.20.

$$\varphi_1(b) = \varphi_{opd}(b)$$
$$\varphi_2(b) = \varphi_{opd}(b) - \varphi_{ipd}(b).$$

Finally, in order to get to $H_{11}(k, n_{e+1})$, $H_{12}(k, n_{e+1})$, $H_{21}(k, n_{e+1})$ and $H_{22}(k, n_{e+1})$, the following equations are applied.

$$\begin{aligned}
H_{11}(k, n_{e+1}) &= h_{11}(b(k)) \cdot \exp(j\varphi_1(b(k))) \\
H_{12}(k, n_{e+1}) &= h_{12}(b(k)) \cdot \exp(j\varphi_2(b(k))) \\
H_{21}(k, n_{e+1}) &= h_{21}(b(k)) \cdot \exp(j\varphi_1(b(k))) \\
H_{22}(k, n_{e+1}) &= h_{22}(b(k)) \cdot \exp(j\varphi_2(b(k)))
\end{aligned},$$

where Table 8.22 and Table 8.23 are used to translate from parameter indexing to subband indexing. For indices denoted with a *, we use:

$$\begin{aligned}
H_{11}(k, n_{e+1}) &= h_{11}(b(k)) \cdot \exp(-j\varphi_1(b(k))) \\
H_{12}(k, n_{e+1}) &= h_{12}(b(k)) \cdot \exp(-j\varphi_2(b(k))) \\
H_{21}(k, n_{e+1}) &= h_{21}(b(k)) \cdot \exp(-j\varphi_1(b(k))) \\
H_{22}(k, n_{e+1}) &= h_{22}(b(k)) \cdot \exp(-j\varphi_2(b(k)))
\end{aligned}.$$

Table 8.22 - Mapping of parameters from 20 bands to 71 sub subbands.

sub subband index *k*	QMF channel	Parameter index *b*(*k*)	
0	0	1*	Sub QMF
1	0	0*	
2	0	0	
3	0	1	
4	0	2	
5	0	3	
6	1	4	
7	1	5	
8	2	6	
9	2	7	
10	3	8	QMF (only)
11	4	9	
12	5	10	
13	6	11	
14	7	12	
15	8	13	
16-17	9-10	14	
18-20	11-13	15	
21-24	14-17	16	
25-29	18-22	17	
30-41	23-34	18	
42-70	35-63	19	

Table 8.23 - Mapping of parameters from 34 bands to 91 sub subbands.

Sub subband index *k*	QMF channel	Parameter index *b*(*k*)	
0	0	0	Sub QMF
1	0	1	
2	0	2	
3	0	3	
4	0	4	
5	0	5	
6-7	0	6	
8	0	7	
9	0	2*	
10	0	1*	
11	0	0*	
12-13	1	10	
14	1	4	
15	1	5	
16	1	6	
17	1	7	
18	1	8	
19	1	9	
20	2	10	
21	2	11	
22	2	12	
23	2	9	
24	3	14	
25	3	11	
26	3	12	
27	3	13	
28	4	14	
29	4	15	
30	4	16	
31	4	13	
32	5	16	QMF (only)
33	6	17	
34	7	18	
35	8	19	

36	9	20	
37	10	21	
38-39	11-12	22	
40-41	13-14	23	
42-43	15-16	24	
44-45	17-18	25	
46-47	19-20	26	
48-50	21-23	27	
51-53	24-26	28	
54-56	27-29	29	
57-59	30-32	30	
60-63	33-36	31	
64-67	37-40	32	
68-90	41-63	33	

8.6.1.6.4 Interpolation

The intermediate values for $H_{11}(k,n)$, $H_{12}(k,n)$, $H_{21}(k,n)$ and $H_{22}(k,n)$ at positions $n = n_e + 1 \ldots n_{e+1}$ are obtained by means of linear interpolation conforming to.

$$
\begin{aligned}
H_{11}(k,n) &= H_{11}(k,n_e) + (n-n_e)\frac{H_{11}(k,n_{e+1}) - H_{11}(k,n_e)}{n_{e+1}-n_e} \\
H_{12}(k,n) &= H_{12}(k,n_e) + (n-n_e)\frac{H_{12}(k,n_{e+1}) - H_{12}(k,n_e)}{n_{e+1}-n_e} \\
H_{21}(k,n) &= H_{21}(k,n_e) + (n-n_e)\frac{H_{21}(k,n_{e+1}) - H_{21}(k,n_e)}{n_{e+1}-n_e} \\
H_{22}(k,n) &= H_{22}(k,n_e) + (n-n_e)\frac{H_{22}(k,n_{e+1}) - H_{22}(k,n_e)}{n_{e+1}-n_e}
\end{aligned}
.
$$

8.6.1.7 Hybrid QMF synthesis filterbank

The stereo processed hybrid subband signals $l_k(n)$ and $r_k(n)$ are fed into the hybrid synthesis filterbanks, which are implemented as adders of sub QMF samples. This is illustrated in Figure 8.4 and Figure 8.6. The two synthesis filterbanks are identical to the 64 complex QMF synthesis filterbank as defined in ISO/IEC 14496-3/AMD1:2003 subclause 4.6.18.4.2. The input to the filterbank are slots of 64 QMF samples. For each slot the filterbank outputs one block of 64 samples of the one channel of the reconstructed stereo signal. There are two independent instances of the QMF synthesis filterbank for the left and right channel, respectively.

8.7 References

[Breebaart] Binaural processing model based on contralateral inhibition I. Model setup, J. Breebaart, S. van de Par and A. Kohlrausch, J. Acoust. Soc. Am., 110:1074–1088, 2001.

Annex A
(normative)
Combination of the SBR tool with the parametric stereo tool

8.A.1 Overview

The parametric stereo coding tool (PS tool) can be used in combination with the SBR tool as defined in subclause 4.6.18. In this case, a 1-channel audio signal is conveyed by AAC+SBR (i.e., HE-AAC) and the PS tool is used to reconstruct a 2-channel stereo signal from this monaural signal. The bitstream element ps_data() as defined in subclause 8.4.1 conveys the information needed by the PS tool and is carried in the sbr_extension() container of the SBR bitstream.

The usage of this parametric stereo extension to HE-AAC is signalled implicitly in the bitstream. Hence, if an sbr_extension() with bs_extension_id==EXTENSION_ID_PS is found in the SBR part of the bitstream, a decoder supporting the combination of SBR and PS shall operate the PS tool to generate a stereo output signal. If no ps_data() element is available in the SBR part of a monaural HE-AAC bitstream, the normal monaural signal is generated by the SBR tool.

8.A.2 Bitstream syntax and semantics

The bitstream element ps_data() as defined in subclause 8.4.1 is carried in the sbr_extension() container (see Table 8.A.1 below) provided by the SBR bitstream defined in subclause 4.4.2.8. The semantics of the bs_extension_id field are given in Table 8.A.2, which replaces Table 4.97 "bs_extension_id."

Table 8.A.1 – Syntax of sbr_extension()

Syntax	No. of bits	Mnemonic
sbr_extension(bs_extension_id, num_bits_left)		
{		
switch (bs_extension_id) {		
case EXTENSION_ID_PS:		
num_bits_left -= ps_data();		Note 1
break;		
default:		
bs_fill_bits;	num_bits_left	**bslbf**
num_bits_left = 0;		
break;		
}		
}		
Note 1: ps_data() returns the number of bits read.		

Table 8.A.2 – Values of the bs_extension_id field

Symbol	Value	Purpose
EXTENSION_ID_PS	2	Parametric Stereo Coding
	all other values	reserved

8.A.3 Decoding process

Semantics and decoding process for the PS tool are defined in subclauses 8.5.1 and 8.6.1, respectively. When the PS tool is combined with SBR, a stereo frame is identical to an SBR frame and consists of 32 complex samples per QMF band for 1024 framing of the AAC (30 samples for 960 framing). The initial 64-band QMF analysis filterbank of the PS tool is removed and the 64-band QMF representation of the monaural signal generated by the SBR tool as available directly prior to SBR's 64-band QMF synthesis filterbank is used as input to the PS tool. The monaural 64-band QMF synthesis filterbank of the SBR tool is obsolete and hence removed. Instead, the two 64-band QMF synthesis filterbanks at the output of the PS tool are used to generate the stereo audio signal.

As shown in Figure 4.46, "Synchronization and timing" in the SBR tool description, there are 6 QMF samples look-ahead available in the low-band buffer, i.e., $\mathbf{X}_{Low}(k,l)$ with 34<=l<40 for 1024 framing. Hence, the hybrid filter structure of the PS tool, where the lowest 3 or 5 QMF bands (all of which are in the low-band) are split up further with help of 13-tap linear-phase FIR filters, does not introduce any algorithmic delay when the PS tool is inserted between SBR processing and the final 64-band QMF synthesis filterbanks. This means that the delay as depicted in Figure 8.2 is obsolete and thus removed.

Hence, the input to the PS tool is a matrix $\mathbf{X}_{input}(k,l)$ defined according to:

$$\mathbf{X}_{input}\left(k,l\right)=\begin{cases}\mathbf{X}_{Low}\left(k,l+t_{HFAdj}\right) & ,\ 0 \le k < 5, numTimeSlots\cdot RATE \le l < numTimeSlots\cdot RATE+6\\ \mathbf{X}\left(k,l\right) & ,\ 0 \le k < 64, 0 \le l < numTimeSlots\cdot RATE\end{cases}$$

where $\mathbf{X}_{Low}(k,l)$, $\mathbf{X}(k,l)$, t_{HFAdj} , *numTimeSlots*, and *RATE* are defined in subclause 4.6.18 SBR Tool.
Furthermore, *numQMFSlots=numTimeSlots*RATE.*

The PS tool uses a complex-valued QMF representation and therefore cannot be used in combination with the low power version of the SBR tool. If DRC is used in combination with SBR as defined in subclause 4.5.2.7.5, DRC is applied in the QMF domain to the output of the PS tool immediately prior to the QMF synthesis filterbanks. The same *factor(k,l)* is applied to both the left and the right audio channel.

8.A.4 Baseline version of the parametric stereo coding tool

In order to facilitate implementation of the PS decoder tool on platforms with very limited computational resources, a baseline version of the PS tool is defined. A PS decoder implementing this baseline version always uses the hybrid filter structure for 20 stereo bands and does not implement IPD/OPD synthesis. This results in a reduction of the computational complexity by approximately 25% when compared to unrestricted PS tool. The baseline version of the PS tool supports the complete bitstream syntax for ps_data(). However, IPD/OPD data is ignored and reset to IPD=OPD=0 prior to stereo synthesis. If a 34 stereo band configuration is used for IID or ICC parameters in the bitstream, the decoded parameters are mapped to 20 stereo bands according to Table 8.20. The averaging process denoted by e.g. $(2*idx_0+idx_1)/3$ in this table is carried out according to ANSI-C integer arithmetic for the integer index representation idx_k of the IID or ICC parameters prior to dequantization.

Annex B
(normative)
Normative Tables

8.B.1 Huffman tables ps_data()

The function ps_huff_dec() is used as:

data = ps_huff_dec (t_huff, codeword),

where *t_huff* is the selected Huffman table and *codeword* is the word read from the bitstream. The return value *data*, is a Huffman table index corresponding to a specific code word.

Huffman table overview:

Table 8.B.3 – huff_iid_df[0] and huff_iid_dt[0]

Index	huff_iid_df[0]	huff_iid_dt[0]	Index	huff_iid_df[0]	huff_iid_dt[0]
-30	011111111010110100	0100111011010100	1	010	00
-29	011111111010110101	0100111011010101	2	0001	01011
-28	011111110101110110	0100111011001110	3	01101	010010
-27	011111110101110111	0100111011001111	4	011101	0100001
-26	011111110101110100	0100111011001100	5	0111101	01001100
-25	011111110101110101	0100111011010110	6	01111101	010011011
-24	011111111010001010	0100111011011000	7	011111100	0100111010
-23	011111111010001011	0100111101000110	8	0111111100	01001111001
-22	011111111010001000	0100111101100000	9	01111111100	01001110000
-21	01111111010000000	010011100011000	10	01111110100	010011101111
-20	011111111010110110	010011100011001	11	011111101011	010011100010
-19	01111111010000010	010011101100100	12	0111111101010	0100111101010
-18	01111111010111000	010011101100101	13	01111111101010	0100111011000
-17	0111111101000010	010011101101101	14	01111111010110	01001111010111
-16	0111111110101110	010011110110001	15	011111111010000	01001111010000
-15	011111110101111	01001110110111	16	0111111110101111	010011110110010
-14	01111111010001	01001111010110	17	0111111101000011	010011110100010
-13	01111111101001	0100111000111	18	01111111010111001	010011100011010
-12	0111111101001	0100111101001	19	01111111010000011	010011100011011
-11	011111101010	0100111101101	20	011111111010110111	0100111101100110
-10	011111111011	010011101110	21	01111111010000001	0100111101100111
-9	01111111011	010011110111	22	011111111010001001	0100111101100001
-8	0111111011	01001111000	23	011111111010001110	0100111101000111
-7	0111111111	0100111001	24	011111111010001111	0100111011011001
-6	01111100	010011010	25	011111111010001100	0100111011010111
-5	0111100	010011111	26	011111111010001101	0100111011001101
-4	011100	0100000	27	011111111010110010	0100111011010010
-3	01100	010001	28	011111111010110011	0100111011010011
-2	0000	01010	29	011111111010110000	0100111011010000
-1	001	011	30	011111111010110001	0100111011010001
0	1	1			

Table 8.B.4 - huff_iid_df[1] and huff_iid_dt[1]

Index	huff_iid_df[1]	huff_iid_dt[1]
-14	11111111111111011	1111111111111111001
-13	11111111111111100	1111111111111111010
-12	11111111111111101	1111111111111111011
-11	11111111111111010	11111111111111111000
-10	1111111111111100	11111111111111111001
-9	111111111111100	11111111111111111010
-8	1111111111101	11111111111111101
-7	1111111110	111111111111110
-6	111111110	111111111110
-5	1111110	1111111110
-4	111100	11111110
-3	11101	111110
-2	1101	1110
-1	101	10
0	0	0
1	100	110
2	1100	11110
3	11100	1111110
4	111101	111111110
5	111110	11111111110
6	11111110	1111111111110
7	11111111110	11111111111110
8	1111111111100	11111111111111100
9	11111111111100	1111111111111111000
10	11111111111101	11111111111111111011
11	111111111111101	11111111111111111100
12	11111111111111110	11111111111111111101
13	111111111111111110	11111111111111111110
14	111111111111111111	11111111111111111111

Table 8.B.5 – huff_icc_dt and huff_icc_df

Index	huff_icc_df	huff_icc_dt
-7	11111111111111	11111111111110
-6	11111111111110	1111111111110
-5	111111111110	11111111110
-4	1111111110	111111110
-3	1111110	1111110
-2	11110	11110
-1	110	110
0	0	0
1	10	10
2	1110	1110
3	111110	111110
4	11111110	11111110
5	111111110	1111111110
6	11111111110	111111111110
7	1111111111110	11111111111111

Table 8.B.6 – huff_ipd_df and huff_ipd_dt

Index	huff_ipd_df	huff_ipd_dt
0	1	1
1	000	010
2	0110	0010
3	0100	00011
4	0010	00010
5	0011	0000
6	0101	0011
7	0111	011

Table 8.B.7 - huff_opd_df and huff_opd_dt

Index	huff_opd_df	huff_opd_dt
0	1	1
1	001	010
2	0110	0001
3	0100	00111
4	01111	00110
5	01110	0000
6	0101	0010
7	000	011