AUDIO COMPRESSION USING MODIFIED DISCRETE COSINE TRANSFORM: THE MP3 CODING STANDARD

BY

JOEBERT S. JACABA

AN UNDERGRADUATE RESEARCH PAPER
SUBMITTED TO THE
DEPARTMENT OF MATHEMATICS
COLLEGE OF SCIENCE
THE UNIVERSITY OF THE PHILIPPINES
DILIMAN, QUEZON CITY

IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF SCIENCE IN MATHEMATICS

OCTOBER 2001

The University of the Philippines
College of Science
Department of Mathematics

This Undergraduate Research Paper hereto attached, entitled AUDIO COMPRESSION USING MODIFIED DISCRETE COSINE TRANSFORM: THE MP3 CODING STANDARD, prepared and submitted by **Joebert S. Jacaba**, in partial fulfillment of the requirement for the degree of **Bachelor of Science in Mathematics**, examined and recommended for acceptance and approval.

Ricardo C.H. del Rosario, Ph.D.
Adviser

Accepted and approved by the faculty of the Department of Mathematics in partial fulfillment of the requirement for the degree of Bachelor of Science in Mathematics.

Polly W. Sy, Ph.D., D.Sc.
Chairman, Department of Mathematics

Abstract

In this research paper we discuss the application of the modified discrete cosine transform (MDCT) to audio compression, specifically the MP3 standard. MDCT plays a very important role in perceptual audio coding. We also discuss all of the four primary parts of the compression process, namely the filterbank, psychoacoustics, quantization, and bitstream formatting. The use of MDCT in the output of the filterbank and in psychoacoustics will be described in detail. Furthermore, we present the ideas behind the use of the fast Fourier transform (FFT) in psychoacoustics and the role of Huffman coding in quantization.

For Jaica

Acknowledgements

I would like to thank and express my appreciation and gratitude to the following people:

Sir Ric, for accepting me as his advisee despite the fact that he is already overloaded; for letting me use his computer and other resources; and, for giving me too many favors.

Arnold, Wilson and Erwin of AJMM Computer Systems, for printing this paper.

Manang Che, for always reminding me of finishing this paper.

Nanay, Tatay, Joanne, Joemar and Jaica for continuing to inspire me.

Cc, Calm, Orion, Kurt and Dredd Stweirtz, for sleepless nights.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The popularity of the internet has greatly increased over the past few years and it has become a medium for file sharing. High-bandwidth services are available today for connecting to the internet but the fact still remains that the bulk of people hooked on the net uses phone lines as their medium. Phone modems have a maximum connection of only 56 kbps (kilobits per second). This is equivalent to 7 kBps (kilobytes per second). Given this connection speed, a 10 MB (megabyte) file can be downloaded in about 25 minutes. Savings in time and storage space means cheaper costs. This reality paved the way for demands on improvements in data compression.

Data compression is classified into two major categories: lossless and lossy. A lossless compression produces the exact copy of the original after decompression while its lossy counterpart does not. A typical example of a lossless compression is the ZIP format. This form of data compression is effective on a range of files. Compressing images and audio through this format is not as effective since the information in these types of data is less redundant. This is where lossy or *perceptually lossless* compression comes in. Applied to images, a good example would be the JPEG format. The MP3 format used in coding audio data also uses a lossy compression. It is based mainly on psychoacoustics which takes into consideration the perceptive behavior of the human ear. Certain sound frequencies can not be heard by humans, therefore, there is little point in storing information that can not be perceived. Due to the fact that recording equipment like microphones and guitars are much more sensitive to or produces a broader range of audio resolutions than what the human ear can perceive, a high fidelity recording of sound stores a surplus of audio data that are never heard. So by eliminating them, the resulting sound will be indistinguishable from the original. By using this concept, the MP3 format has attained compression ratios of up to 1:24.

Audio CDs use the popular WAV (waveform) format. WAV is under the more general RIFF file format used by Windows. The WAV format is uncompressed and falls into many categories. One of them is the pulse code modulation (PCM) which is the accepted input for MP3 encoding. The size of a WAV file depends on its sampling rate. An 8-bit mono WAV sampled at 22,050 Hz (Hertz) would take 22,050 bytes per second. A 16-bit stereo WAV with a sampling rate of 44.1 kHz (kiloHertz) takes 176,400 bytes per second (44,100/second * 2 bytes * 2 channels). One minute of a CD-quality audio roughly takes 10 MB. These huge requirements in data storage facilitated the popularity of MP3 as one of the most widely used standards in audio compression. Today, ”MP3” is the most-searched-on term on most of the top search engines. That makes the MP3 format the most famous and widely used audio compression.

1.2 The MP3 History

In January 1988 [1], the MPEG (Moving Pictures Expert Group) was born. The committee, formally known as ISO/IEC JTC1/SC29/WG11 worked on standardizing compression algorithms for moving images, audio, and images with audio. As an output of this group, MPEG-1 was standardized in November 1992. In 1994, efforts on perceptual audio was put forth by Fraunhaufer-IIS in a joint cooperation with the University of Erlangen (Prof. Dieter Seitzer). The result of their work became known as MP3 which stands for MPEG-1 Audio Layer III, and not MPEG-3 which

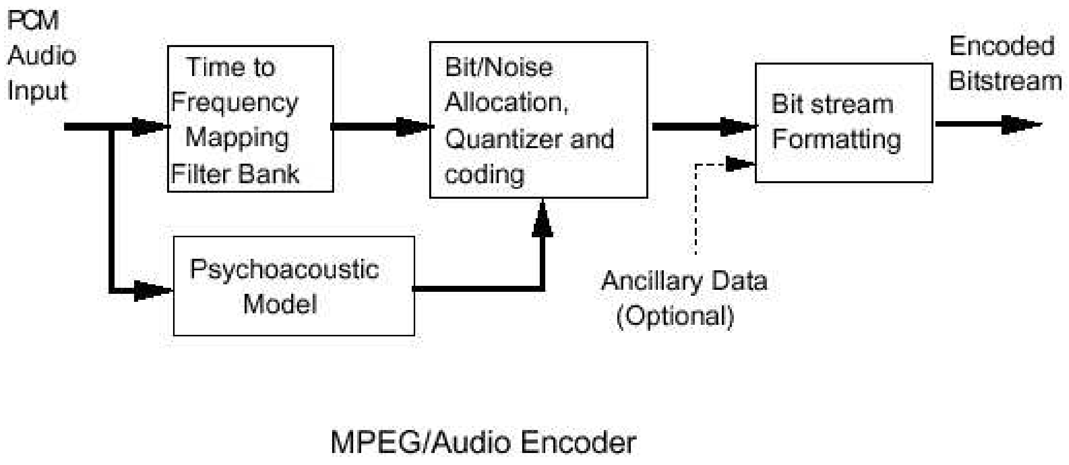

Figure 1.1: Basic Encoder Diagram

is a common misconception.

The MP3 standard does not exactly specify how the encoding process is to be done. It only outlines the techniques and specifies the format of the encoded audio. By doing this, normative elements are minimized and there is much headroom for developers to experiment on. The flow between the four main components of MP3, namely filterbank, psychoacoustics, quantization and bitstream formatting are illustrated in Figure 1.1. We will discuss each part in detail in the succeeding chapters.

1.3 The Algorithm

The MP3 encoding algorithm is divided into stages [4](Figure 1.2) as follows:

1. The first stage consists of dividing the sampled audio signal into smaller components called frames. The signal is made to pass into a time-frequency mapping filterbank which divides it into subbands. This is called a polyphase filterbank and uses a pseudo QMF filter. The output samples are then quantized.

2. The second stage consists of a 1024-point FFT on the sample and then applying the psychoacoustic model. The concepts of masking and thresholds are utilized

to discard data that is inaudible. The final output of the model is a signal-to-mask ratio (SMR) for each group of bands.

An MDCT filter is then performed on the output of first and second stages. This contributes largely to the compression.

3. The third stage consists of quantifying and encoding each sample of each subband by calculating a coefficient required to represent the signal-to-noise ratio (SNR) in scale. This is also known as noise allocation. The allocator looks at both the output samples from the filterbank and the SMRs from the psychoacoustic model, and adjusts the noise allocation in order to simultaneously meet both the bit rate and masking requirements.

   The resulting output will further pass on a traditional lossless compression method known as Huffman coding.

4. The last stage consists of formatting the bitstream. The quantized filterbank outputs, the noise allocation and other required side information are collected and then encoded and formated.

Other specifications for the algorithm are as follows:

- The bit rate will be between 8 kbps to 320 kbps. The bit rate refers to the amount of data (in bits) stored for every second of audio. The default standard bit rate is 128 kbps.
- The sampling rate will be 32 kHz, 44.1 kHz, or 48 kHz. The sampling rate refers to the frequency with which the signal is stored. The default standard sampling rate is 44.1 kHz.
- The bitstream will be encoded either with a constant bit rate (CBR) or a variable bit rate (VBR).
- The mode supported will be mono, dual channel, stereo and joint stereo.

By convention, the term *analysis* is used to refer to processes which involves encoding while *synthesis* is used to describe decoding. The process of encoding and

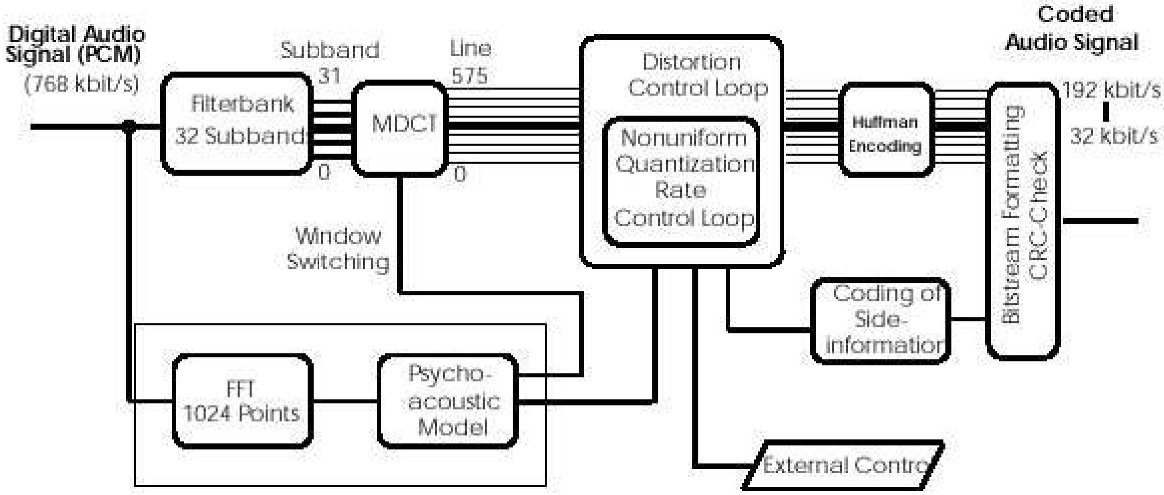

Figure 1.2: Detailed Encoder Diagram

decoding is called *codec* (coined from COmpression/DECompression).

Chapter 2

The Time-Frequency Filterbank

2.1 Input Highpass Filter

The MP3 standard [4] recommends the use of a highpass filter. A highpass filter allows frequencies above a given cutoff frequency to pass and does not allow lower ones to pass. In other words, it attenuates the lower frequencies. The cutoff frequency should be in the range of 2 Hz to 10 Hz.

The application of this kind of filter avoids the unnecessary high bit rate requirement for the lowest subband and increases the overall audio quality.

2.2 Analysis Subband Filter

The analysis subband filterbank is basically a polyphase filter. It is set of band filters covering the entire audio frequency range. It is used to split the input PCM signal with sampling frequency *fs* into subbands. The result will be 32 subbands which are equally spaced with sampling frequencies *fs/32*. The polyphase filter together with the MDCT filter is referred to as the hybrid filterbank.

2.2.1 The polyphase filter

The polyphase filter used in MP3 [8] is adapted from an earlier audio coder named Masking Pattern Adapted Universal Subband Integrated Coding and Multiplexing (MUSICAM). It is a cosine modulated lowpass prototype filter with uniform bandwidth parallel M-channel bandpass filter. This achieves *nearly perfect* reconstruction and has been called a *psuedo QMF* (Quadrature Mirror Filter). M here in the case of MP3 is equal to 32. The advantages of this filter are listed below:

- Constrained design; single FIR (Finite Impulse Response) prototype filter
- Uniform, linear phase channel responses
- Overall linear phase, hence constant group delay
- Low complexity = one filter plus modulation
- Amenable to fast, block algorithms
- Critical sampling

2.2.2 Implementation

The analysis subband filter is implemented in the MP3 algorithm [4] using these following steps:

1. Input 32 audio samples W_i for i=0 to 31.

2. Build an input sample vector X of 512 elements.

$$X_i = X_{i-32}\ ,\ \text{for } i = 511 \text{ down to } 32 \tag{2.1}$$

The 32 audio samples are shifted in at positions 0 to 31, the most recent at position 0, and the 32 oldest elements are shifted out.

$$X_i = W_{31-i}\ ,\ \text{for } i = 31 \text{ down to } 0 \tag{2.2}$$

3. Window vector X by vector C. The coefficients are to be found in Tables 2.1, 2.2, 2.3, 2.4, and 2.5 and represented graphically in Figure 2.1.

$$Z_i = C_i * X_i \text{ , for } i = 0 \text{ to } 511 \tag{2.3}$$

4. Calculate the 64 values of Y_i by the following formula:

$$Y_i = \sum_{j=0}^{7} Z_i + 64j \text{ , for } i = 0 \text{ to } 63. \tag{2.4}$$

5. Calculate the 32 subband samples S_i by matrixing.

$$S_i = \sum_{k=0}^{63} M_{i,k} * Y_k \text{ , for } i = 0 \text{ to } 31 \tag{2.5}$$

The coefficients for the matrix M can be calculated by the following formula:

$$M_{i,k} = \cos\left[\frac{(2i+1)(k-16)\pi}{64}\right] \text{ , for } i = 0 \text{ to } 31,\ k = 0 \text{ to } 63. \tag{2.6}$$

Steps 1-4 calculate the polyphase components of the prototype filter. Step 5 is the polyphase implementation of the subband filters. The vector C in Step 3 contains the coefficients for the prototype filter of the polyphase filter bank.

i	Value	i	Value	i	Value	i	Value
0	0.000000000	1	-0.000000477	2	-0.000000477	3	-0.000000477
4	-0.000000477	5	-0.000000477	6	-0.000000477	7	-0.000000954
8	-0.000000954	9	-0.000000954	10	-0.000000954	11	-0.000001431
12	-0.000001431	13	-0.000001907	14	-0.000001907	15	-0.000002384
16	-0.000002384	17	-0.000002861	18	-0.000003338	19	-0.000003338
20	-0.000003815	21	-0.000004292	22	-0.000004768	23	-0.000005245
24	-0.000006199	25	-0.000006676	26	-0.000007629	27	-0.000008106
28	-0.000009060	29	-0.000010014	30	-0.000011444	31	-0.000012398
32	-0.000013828	33	-0.000014782	34	-0.000016689	35	-0.000018120
36	-0.000019550	37	-0.000021458	38	-0.000023365	39	-0.000025272
40	-0.000027657	41	-0.000030041	42	-0.000032425	43	-0.000034809
44	-0.000037670	45	-0.000040531	46	-0.000043392	47	-0.000046253
48	-0.000049591	49	-0.000052929	50	-0.000055790	51	-0.000059605
52	-0.000062943	53	-0.000066280	54	-0.000070095	55	-0.000073433
56	-0.000076771	57	-0.000080585	58	-0.000083923	59	-0.000087261
60	-0.000090599	61	-0.000093460	62	-0.000096321	63	-0.000099182
64	0.000101566	65	0.000103951	66	0.000105858	67	0.000107288
68	0.000108242	69	0.000108719	70	0.000108719	71	0.000108242
72	0.000106812	73	0.000105381	74	0.000102520	75	0.000099182
76	0.000095367	77	0.000090122	78	0.000084400	79	0.000077724
80	0.000069618	81	0.000060558	82	0.000050545	83	0.000039577
84	0.000027180	85	0.000013828	86	-0.000000954	87	-0.000017166
88	-0.000034332	89	-0.000052929	90	-0.000072956	91	-0.000093937
92	-0.000116348	93	-0.000140190	94	-0.000165462	95	-0.000191212
96	-0.000218868	97	-0.000247478	98	-0.000277042	99	-0.000307560
100	-0.000339031	101	-0.000371456	102	-0.000404358	103	-0.000438213

Table 2.1: Coefficients of C_i

i	Value	i	Value	i	Value	i	Value
104	-0.000472546	105	-0.000507355	106	-0.000542164	107	-0.000576973
108	-0.000611782	109	-0.000646591	110	-0.000680923	111	-0.000714302
112	-0.000747204	113	-0.000779152	114	-0.000809669	115	-0.000838757
116	-0.000866413	117	-0.000891685	118	-0.000915051	119	-0.000935555
120	-0.000954151	121	-0.000968933	122	-0.000980854	123	-0.000989437
124	-0.000994205	125	-0.000995159	126	-0.000991821	127	-0.000983715
128	0.000971317	129	0.000953674	130	0.000930786	131	0.000902653
132	0.000868797	133	0.000829220	134	0.000783920	135	0.000731945
136	0.000674248	137	0.000610352	138	0.000539303	139	0.000462532
140	0.000378609	141	0.000288486	142	0.000191689	143	0.000088215
144	-0.000021458	145	-0.000137329	146	-0.000259876	147	-0.000388145
148	-0.000522137	149	-0.000661850	150	-0.000806808	151	-0.000956535
152	-0.001111031	153	-0.001269817	154	-0.001432419	155	-0.001597881
156	-0.001766682	157	-0.001937389	158	-0.002110004	159	-0.002283096
160	-0.002457142	161	-0.002630711	162	-0.002803326	163	-0.002974033
164	-0.003141880	165	-0.003306866	166	-0.003467083	167	-0.003622532
168	-0.003771782	169	-0.003914356	170	-0.004048824	171	-0.004174709
172	-0.004290581	173	-0.004395962	174	-0.004489899	175	-0.004570484
176	-0.004638195	177	-0.004691124	178	-0.004728317	179	-0.004748821
180	-0.004752159	181	-0.004737377	182	-0.004703045	183	-0.004649162
184	-0.004573822	185	-0.004477024	186	-0.004357815	187	-0.004215240
188	-0.004049301	189	-0.003858566	190	-0.003643036	191	-0.003401756
192	0.003134727	193	0.002841473	194	0.002521515	195	0.002174854
196	0.001800537	197	0.001399517	198	0.000971317	199	0.000515938
200	0.000033379	201	-0.000475883	202	-0.001011848	203	-0.001573563

Table 2.2: Coefficients of C_i

i	Value	i	Value	i	Value	i	Value
204	-0.002161503	205	-0.002774239	206	-0.003411293	207	-0.004072189
208	-0.004756451	209	-0.005462170	210	-0.006189346	211	-0.006937027
212	-0.007703304	213	-0.008487225	214	-0.009287834	215	-0.010103703
216	-0.010933399	217	-0.011775017	218	-0.012627602	219	-0.013489246
220	-0.014358521	221	-0.015233517	222	-0.016112804	223	-0.016994476
224	-0.017876148	225	-0.018756866	226	-0.019634247	227	-0.020506859
228	-0.021372318	229	-0.022228718	230	-0.023074150	231	-0.023907185
232	-0.024725437	233	-0.025527000	234	-0.026310921	235	-0.027073860
236	-0.027815342	237	-0.028532982	238	-0.029224873	239	-0.029890060
240	-0.030526638	241	-0.031132698	242	-0.031706810	243	-0.032248020
244	-0.032754898	245	-0.033225536	246	-0.033659935	247	-0.034055710
248	-0.034412861	249	-0.034730434	250	-0.035007000	251	-0.035242081
252	-0.035435200	253	-0.035586357	254	-0.035694122	255	-0.035758972
256	0.035780907	257	0.035758972	258	0.035694122	259	0.035586357
260	0.035435200	261	0.035242081	262	0.035007000	263	0.034730434
264	0.034412861	265	0.034055710	266	0.033659935	267	0.033225536
268	0.032754898	269	0.032248020	270	0.031706810	271	0.031132698
272	0.030526638	273	0.029890060	274	0.029224873	275	0.028532982
276	0.027815342	277	0.027073860	278	0.026310921	279	0.025527000
280	0.024725437	281	0.023907185	282	0.023074150	283	0.022228718
284	0.021372318	285	0.020506859	286	0.019634247	287	0.018756866
288	0.017876148	289	0.016994476	290	0.016112804	291	0.015233517
292	0.014358521	293	0.013489246	294	0.012627602	295	0.011775017
296	0.010933399	297	0.010103703	298	0.009287834	299	0.008487225
300	0.007703304	301	0.006937027	302	0.006189346	303	0.005462170

Table 2.3: Coefficients of C_i

i	Value	i	Value	i	Value	i	Value
304	0.004756451	305	0.004072189	306	0.003411293	307	0.002774239
308	0.002161503	309	0.001573563	310	0.001011848	311	0.000475883
312	-0.000033379	313	-0.000515938	314	-0.000971317	315	-0.001399517
316	-0.001800537	317	-0.002174854	318	-0.002521515	319	-0.002841473
320	0.003134727	321	0.003401756	322	0.003643036	323	0.003858566
324	0.004049301	325	0.004215240	326	0.004357815	327	0.004477024
328	0.004573822	329	0.004649162	330	0.004703045	331	0.004737377
332	0.004752159	333	0.004748821	334	0.004728317	335	0.004691124
336	0.004638195	337	0.004570484	338	0.004489899	339	0.004395962
340	0.004290581	341	0.004174709	342	0.004048824	343	0.003914356
344	0.003771782	345	0.003622532	346	0.003467083	347	0.003306866
348	0.003141880	349	0.002974033	350	0.002803326	351	0.002630711
352	0.002457142	353	0.002283096	354	0.002110004	355	0.001937389
356	0.001766682	357	0.001597881	358	0.001432419	359	0.001269817
360	0.001111031	361	0.000956535	362	0.000806808	363	0.000661850
364	0.000522137	365	0.000388145	366	0.000259876	367	0.000137329
368	0.000021458	369	-0.000088215	370	-0.000191689	371	-0.000288486
372	-0.000378609	373	-0.000462532	374	-0.000539303	375	-0.000610352
376	-0.000674248	377	-0.000731945	378	-0.000783920	379	-0.000829220
380	-0.000868797	381	-0.000902653	382	-0.000930786	383	-0.000953674
384	0.000971317	385	0.000983715	386	0.000991821	387	0.000995159
388	0.000994205	389	0.000989437	390	0.000980854	391	0.000968933
392	0.000954151	393	0.000935555	394	0.000915051	395	0.000891685
396	0.000866413	397	0.000838757	398	0.000809669	399	0.000779152
400	0.000747204	401	0.000714302	402	0.000680923	403	0.000646591

Table 2.4: Coefficients of C_i

i	Value	i	Value	i	Value	i	Value
404	0.000611782	405	0.000576973	406	0.000542164	407	0.000507355
408	0.000472546	409	0.000438213	410	0.000404358	411	0.000371456
412	0.000339031	413	0.000307560	414	0.000277042	415	0.000247478
416	0.000218868	417	0.000191212	418	0.000165462	419	0.000140190
420	0.000116348	421	0.000093937	422	0.000072956	423	0.000052929
424	0.000034332	425	0.000017166	426	0.000000954	427	-0.000013828
428	-0.000027180	429	-0.000039577	430	-0.000050545	431	-0.000060558
432	-0.000069618	433	-0.000077724	434	-0.000084400	435	-0.000090122
436	-0.000095367	437	-0.000099182	438	-0.000102520	439	-0.000105381
440	-0.000106812	441	-0.000108242	442	-0.000108719	443	-0.000108719
444	-0.000108242	445	-0.000107288	446	-0.000105858	447	-0.000103951
448	0.000101566	449	0.000099182	450	0.000096321	451	0.000093460
452	0.000090599	453	0.000087261	454	0.000083923	455	0.000080585
456	0.000076771	457	0.000073433	458	0.000070095	459	0.000066280
460	0.000062943	461	0.000059605	462	0.000055790	463	0.000052929
464	0.000049591	465	0.000046253	466	0.000043392	467	0.000040531
468	0.000037670	469	0.000034809	470	0.000032425	471	0.000030041
472	0.000027657	473	0.000025272	474	0.000023365	475	0.000021458
476	0.000019550	477	0.000018120	478	0.000016689	479	0.000014782
480	0.000013828	481	0.000012398	482	0.000011444	483	0.000010014
484	0.000009060	485	0.000008106	486	0.000007629	487	0.000006676
488	0.000006199	489	0.000005245	490	0.000004768	491	0.000004292
492	0.000003815	493	0.000003338	494	0.000003338	495	0.000002861
496	0.000002384	497	0.000002384	498	0.000001907	499	0.000001907
500	0.000001431	501	0.000001431	502	0.000000954	503	0.000000954
504	0.000000954	505	0.000000954	506	0.000000477	507	0.000000477
508	0.000000477	509	0.000000477	510	0.000000477	511	0.000000477

Table 2.5: Coefficients of C_i

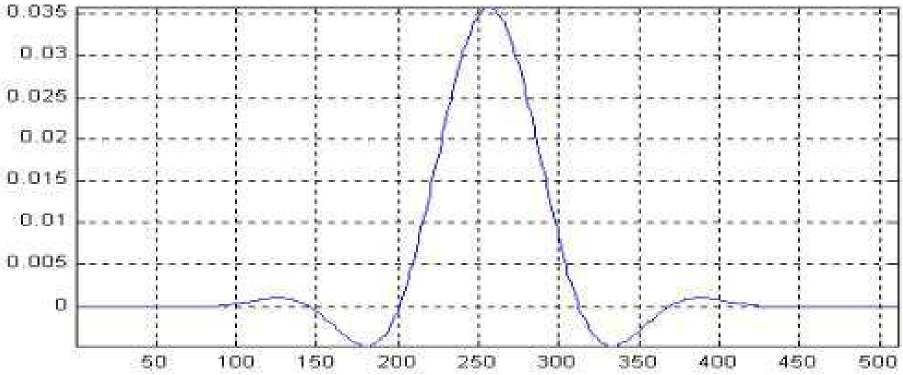

Figure 2.1: Coefficients of C_i

Chapter 3

The Psychoacoustic Model

3.1 Fundamentals

A wave is an aspect of matter. It has properties such as amplitude, wavelength and frequency. The unit of measuring frequency is Hertz (Hz) which means *cycles per second*. Sound waves in particular are mechanical and need a medium to travel and propagate. Only a range of sound frequency is perceptible to human beings (Figure 3.1). Approximately, the *audible range* [6] is between 20 Hz and 20 kHz. We are most sensitive between the frequencies 2 kHz and 4 kHz. This may be explained by the average range of the human voice, which runs roughly from 500 Hz to 2 kHz. Frequencies below the audible range are called *infrasonic* while above those are *ultrasonic*. Musical instruments can create frequencies up to 100 kHz with at least one member of each instrument family (strings, woodwinds, brass and percussion) producing frequencies up to 40 kHz. With respect to the audible range, midrange frequencies are perceived better than high and low frequencies. Sensitivity to higher frequencies deteriorates with age and prolonged exposure to loud volumes. At a certain age, frequencies above 16 kHz is no longer heard. Statistically, women tend to preserve the ability to hear higher frequencies later in life than men do.

Sound pressure level (SPL) is a relative measure of what we hear and its unit is the decibel (dB) and defined as:

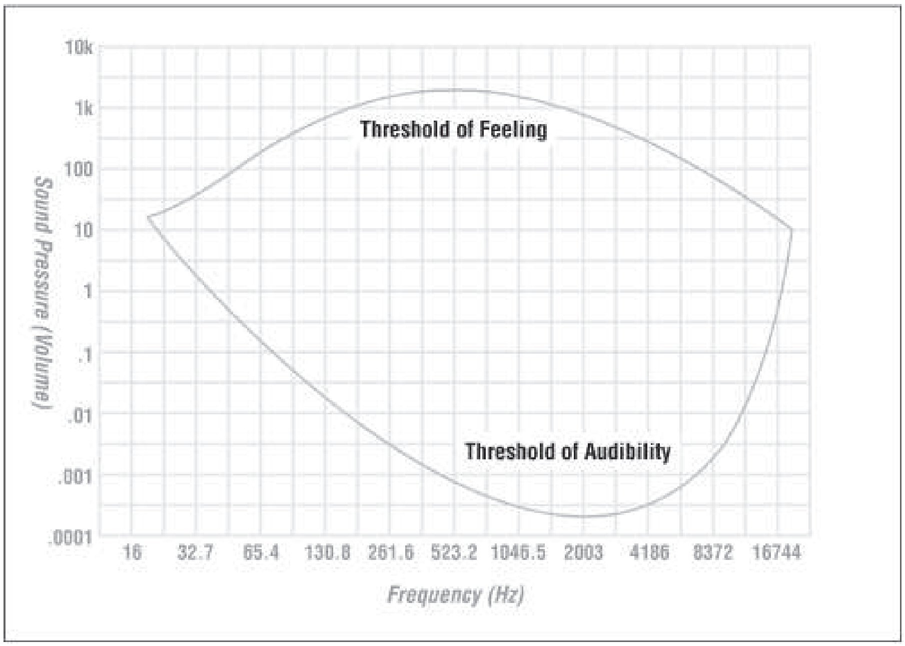

Figure 3.1: Human Hearing Thresholds Curve

$$SPL = 20\,log\left(\frac{\Delta p}{\Delta p_0}\right)\ (dB) \tag{3.1}$$

where Δp is the actual sound pressure and Δp_0 is the reference sound pressure.

Hence 0 dB does not mean absence of sound but denotes a sound level where the sound pressure is equal to that of the reference level. This is a very small pressure close to zero. It is also possible to have negative sound levels. Since the human ear does not respond equally to all frequencies, equal sound pressures do not necessarily indicate equal loudness. For this reason, sound meters are usually fitted with a filter whose response to frequency is a bit like that of the human ear.

Psychoacoustics [6] is the study of the inter-relation between the ear, the mind, and vibratory audio signal. The field has made significant progress toward characterizing human auditory perception and particularly the time-frequency analysis capabilities of the inner ear. Irrelevant information is identified during signal analysis by incorporating into the encoder some of its principles such as absolute threshold of hearing, critical band frequency analysis, auditory masking, and temporal masking. Combining these notions with basic properties of signal quantization has also led to the development of perceptual entropy, a quantitative estimate of the fundamental limit of transparent audio signal compression.

3.1.1 Absolute threshold of hearing

The absolute threshold of hearing [8] characterizes the amount of energy needed in a pure tone such that it can be detected by a listener in a noiseless environment. The absolute threshold is measured in terms of dB Sound Pressure Level (dB SPL) and is approximated with the following non-linear function:

$$\begin{aligned} T_q(f) &= 3.64 \left(\tfrac{f}{1000}\right)^{-0.8} - 6.5 \exp\left(-0.6 \left(\tfrac{f}{1000} - 3.3\right)^2\right) \\ &+ 10^{-3} \left(\tfrac{f}{1000}\right)^4 \ (dB). \end{aligned} \qquad (3.2)$$

3.1.2 Critical bands

The cochlea [8] acts as a bandpass filterbank of non-uniform bandwidth. Bandwidth increases with increasing frequency. The critical band is a function of frequency that quantifies the bandpass filterbank. A Bark is a unit distance of one critical band. A frequency in Hz can be converted to Bark using the following formula:

$$z(f) = 13 \arctan(.00076\, f) + 3.5 \arctan\left[\left(\frac{f}{7500}\right)^2\right] \ (Bark). \qquad (3.3)$$

A typical critical band filter can be found in Table 3.1.

3.1.3 Auditory masking

Auditory masking or simultaneous masking [6] is achieved if two different frequencies are close to each other and the other is weaker. An example is demonstrated in Figure 3.2. The figure shows two audio signals consisting of a perfect sine wave, the first

Band No.	Center Frequency (Hz)	Bandwidth (Hz)
1	50	-100
2	150	100-200
3	250	200-300
4	350	300-400
5	450	400-510
6	570	510-630
7	700	630-770
8	840	770-920
9	1000	920-1080
10	1175	1080-1270
11	1370	1270-1480
12	1600	1480-1720
13	1850	1720-2000
14	2150	2000-2320
15	2500	2320-2700
16	2900	2700-3150
17	3400	3150-3700
18	4000	3700-4400
19	4800	4400-5300
20	5800	5300-6400
21	7000	6400-7700
22	8500	7700-9500
23	10500	9500-12000
24	13500	12000-15500
25	19500	15500-

Table 3.1: Critical Band Filter

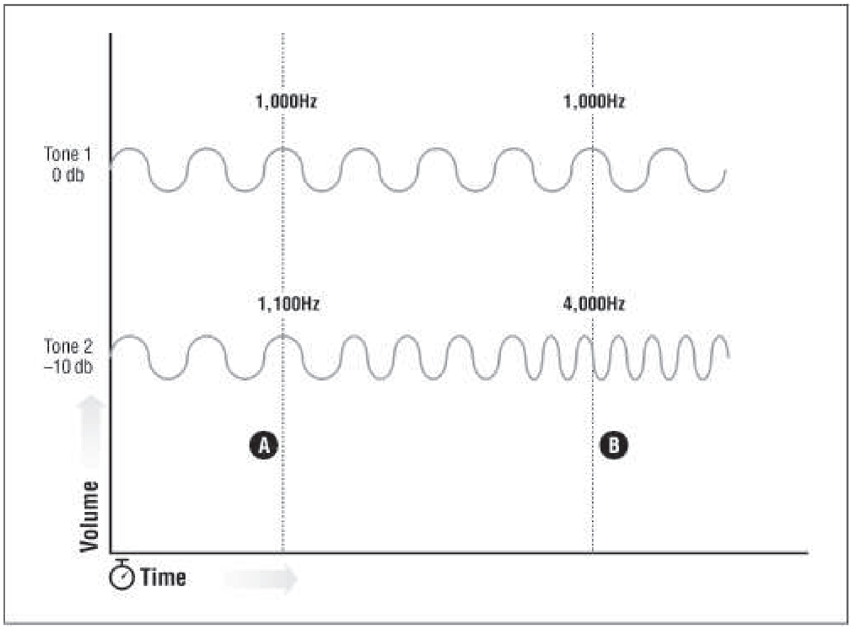

Figure 3.2: Auditory Masking Example

one fluctuating at 1,000 Hz at 0 dB and the second one at 1,100 kHz at -10 dB. It can be seen in the figure that the second will be inaudible at point A. By varying the frequency of the second signal until it reaches 4,000 kHz while maintaining its SPL at -10 dB, then both signals can be heard simultaneously at point B. This demonstrates that when the frequencies of two signals are close to each other the weaker signal will be inaudible or masked. If two frequencies are apart, even if one of them is weaker then both would still be heard. The strong tonal signal masks a region of weaker signals under a curve. This is depicted in Figure 3.3.

3.1.4 Temporal masking

While auditory masking is dependent on the relationship between frequencies and their relative volumes, temporal masking [6] is based on time rather than on frequency. Temporal masking is achieved if a loud sound and a quiet sound is played simultaneously. By placing a sufficient delay between the two sounds the softer sound

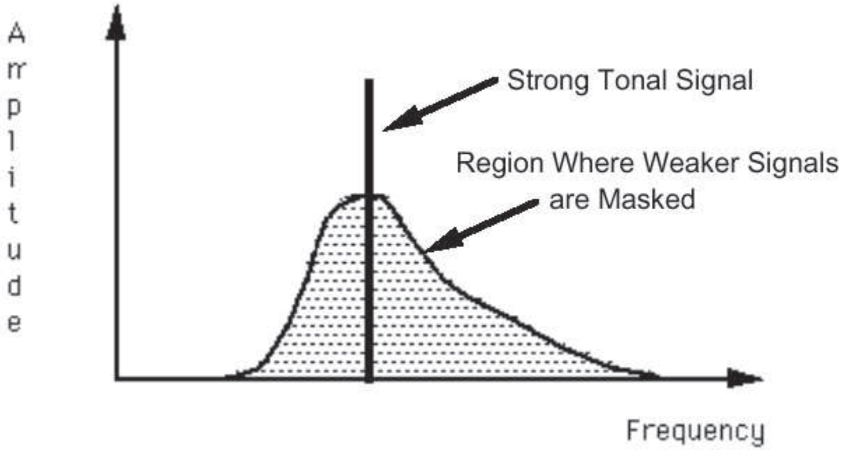

Figure 3.3: Auditory Masking Curve

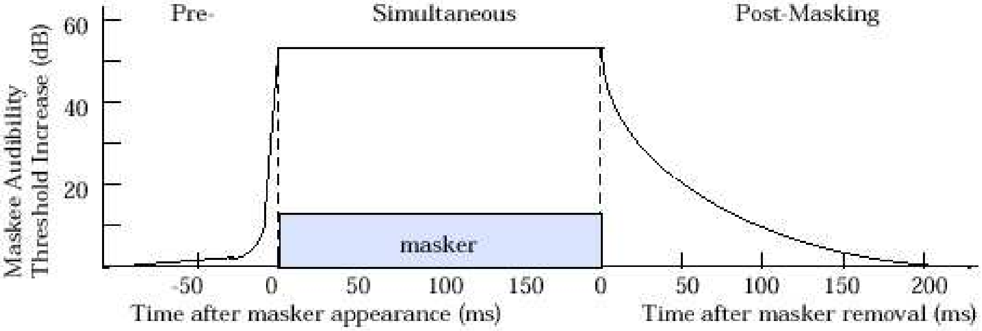

Figure 3.4: Temporal Masking Curve

will be heard. By determining or quantifying the length of time between two tones at which both tones would be audible, temporal masking therefore contributes to data reduction in audio compression. Whether a loud tone or a quiet tone comes first does not matter. Temporal masking applies to premasking and postmasking (Figure 3.4). The average time gap for both tone to be heard is 5 ms when working with pure tones, though it varies up and down in accordance with different audio passages.

3.2 Implementation

The following implementation [4] is based on what the standard refers to as the ”Psychoacoustic Model 1.”

The bit allocation of the 32 subbands is calculated on the basis of the SMRs of all the subbands. Therefore it is necessary to determine, for each subband the maximum signal level and the minimum masking threshold. The minimum masking threshold is derived from an FFT of the input PCM signal, followed by a psychoacoustic model calculation.

The FFT in parallel with the subband filter compensates for the lack of spectral selectivity obtained at low frequencies by the subband filterbank. This technique provides both a sufficient time resolution for the coded audio signal (polyphase filter with optimized window for minimal pre-echoes) and a sufficient spectral resolution for the calculation of the masking thresholds.

The frequencies and levels of aliasing distortions can be calculated. This is necessary for calculating a minimum bit rate for those subbands which need some bits to cancel the aliasing components in the decoder. The additional complexity to calculate the better frequency resolution is necessary only in the encoder, and introduces no additional delay or complexity in the decoder.

The calculation of the SMR is based on the following steps which will be further discussed. A sampling frequency of 44.1 kHz is assumed throughout. Appropriate scaling should be made to the other two sampling frequencies.

1. Calculation of the FFT for time to frequency conversion.

2. Determination of the sound pressure level in each subband.

3. Determination of the threshold in quiet (absolute threshold).

4. Finding of the tonal (sinusoid-like) and non-tonal (noise-like) components of the audio signal.

5. Decimation of the maskers, to obtain only the relevant maskers.

6. Calculation of the individual masking thresholds.

7. Determination of the global masking threshold.

8. Determination of the minimum masking threshold in each subband.

9. Calculation of the SMR in each subband.

3.2.1 FFT analysis

Incoming audio samples, $s(n)$, are normalized [8] according to FFT length, N, and the number of bits per sample, b, using the equation:

$$x(n) = \frac{s(n)}{N\left(2^{b-1}\right)}. \tag{3.4}$$

The masking threshold is derived from an estimate of the power density spectrum, $P(k)$ that is calculated by a 1024-point FFT,

$$P(k) = PN + 10\,\log\left|\sum_{n=0}^{N-1} h(n)\,x(n)\,\exp\left(-j\frac{2\pi\,k\,n}{N}\right)\right|^2 (dB),\ 0 \leq k \leq N/2 \tag{3.5}$$

where $h(n)$ is a Hann window computed from

$$h(n) = 0.5\left(1 - \cos\frac{2\,\pi\,n}{N-1}\right)\ , 0 \leq i \leq N-1 \tag{3.6}$$

and PN is the power normalization term. A normalization to the reference level of 96 dB SPL has to be done in such a way that the maximum value corresponds to 96 dB.

For a coincidence in time between the bit allocation and the corresponding subband samples, the PCM samples entering the FFT have to be delayed:

Transform length	1024 samples
Window size if $fs = 48$ kHz	21.3 ms
Window size if $fs = 44.1$ kHz	23.2 ms
Window size if $fs = 32$ kHz	32 ms
Frequency resolution	$fs/1024$

Table 3.2: Technical Data of the FFT

1. The delay of the analysis subband filter is 256 samples, corresponding to 5.8 ms at the 44.1 kHz sampling rate. This corresponds to a window shift of 256 samples.

2. The Hann window must coincide with the subband samples of the frame. This means an additional window shift of minus 64 samples.

The window size depending on sampling frequency fs is listed in Table 3.2.

3.2.2 SPL determination

The sound pressure level L_{SB} in subband n is computed by:

$$L_{SB}(n) = max[P(k), 20\log\left(SCF_{max}(n) * 32768\right) - 10]\ (dB) \tag{3.7}$$

where $P(k)$ is the sound pressure level of the spectral line with index k of the FFT with the maximum amplitude in the frequency range corresponding to subband n. The expression $SCF_{max}(n)$ is the maximum of the three scalefactors of subband n within a frame. The -10 dB term corrects for the difference between peak and RMS (root-mean-square) level. The sound pressure level $L_{SB}(n)$ is computed for every subband n.

3.2.3 Treshhold in quiet

The threshold in quiet [4] $T_q(k)$, or the absolute threshold of hearing (Equation 3.2, is computed in Tables 3.3, 3.4, 3.5, 3.6 and 3.7. An offset depending on the overall bit rate is used for the absolute threshold. This offset is -12 dB for bit rates $\leq$ 96 kbps and 0 dB for bit rates $<$ 96 kbps per channel.

3.2.4 Tonal and non-tonal components

The tonality of a masking component [4] has an influence on the masking threshold. For this reason, it is worthwhile to discriminate between tonal and non-tonal components. For calculating the global masking threshold it is necessary to derive the tonal and the non-tonal components from the FFT spectrum.

This step starts with the determination of local maxima, then extracts tonal components (sinusoids) and calculates the intensity of the non-tonal components within a bandwidth of a critical band. The boundaries of the critical bands are given in Table 3.8.

The bandwidth of the critical bands varies with the center frequency with a bandwidth of about only 0.1 kHz at low frequencies and with a bandwidth of about 4 kHz at high frequencies. It is known from psychoacoustic experiments that the ear has a better frequency resolution in the lower than in the higher frequency region. To determine if a local maximum may be a tonal component a frequency range *df* around the local maximum is examined. The frequency range *df* is given by Table 3.9.

To make lists of the spectral lines $P(k)$ that are tonal or non-tonal, the following three operations are performed:

- Labelling of local maxima

  A spectral line, $X(k)$, is labelled as a local maximum if

  $$P(k) > P(k-1)$$

  and

  $$P(k) \geq P(k+1).$$

- Listing of tonal components and calculation of the sound pressure level

Index Number (i)	Frequency (Hz)	Critical Band Rate (z)	Absolute Threshold (dB)
1	43.07	.425	45.05
2	86.13	.850	25.87
3	129.20	1.273	18.70
4	172.27	1.694	14.85
5	215.33	2.112	12.41
6	258.40	2.525	10.72
7	301.46	2.934	9.47
8	344.53	3.337	8.50
9	387.60	3.733	7.73
10	430.66	4.124	7.10
11	473.73	4.507	6.56
12	516.80	4.882	6.11
13	559.86	5.249	5.72
14	602.93	5.608	5.37
15	646.00	5.959	5.07
16	689.06	6.301	4.79
17	732.13	6.634	4.55
18	775.20	6.959	4.32
19	818.26	7.274	4.11
20	861.33	7.581	3.92
21	904.39	7.879	3.74
22	947.46	8.169	3.57
23	990.53	8.450	3.40
24	1033.59	8.723	3.25
25	1076.66	8.987	3.10
26	1119.73	9.244	2.95
27	1162.79	9.493	2.81
28	1205.86	9.734	2.67
29	1248.93	9.968	2.53
30	1291.99	10.195	2.39

Table 3.3: Frequencies, Critical Bands, and Absolute Threshold

Index Number (i)	Frequency (Hz)	Critical Band Rate (z)	Absolute Threshold (dB)
31	1335.06	10.416	2.25
32	1378.13	10.629	2.11
33	1421.19	10.836	1.97
34	1464.26	11.037	1.83
35	1507.32	11.232	1.68
36	1550.39	11.421	1.53
37	1593.46	11.605	1.38
38	1636.52	11.783	1.23
39	1679.59	11.957	1.07
40	1722.66	12.125	.90
41	1765.72	12.289	.74
42	1808.79	12.448	.56
43	1851.86	12.603	.39
44	1894.92	12.753	.21
45	1937.99	12.900	.02
46	1981.05	13.042	-.17
47	2024.12	13.181	-.36
48	2067.19	13.317	-.56
49	2153.32	13.578	-.96
50	2239.45	13.826	-1.38
51	2325.59	14.062	-1.79
52	2411.72	14.288	-2.21
53	2497.85	14.504	-2.63
54	2583.98	14.711	-3.03
55	2670.12	14.909	-3.41
56	2756.25	15.100	-3.77
57	2842.38	15.284	-4.09
58	2928.52	15.460	-4.37
59	3014.65	15.631	-4.60
60	3100.78	15.796	-4.78

Table 3.4: Frequencies, Critical Bands, and Absolute Threshold

Index Number (i)	Frequency (Hz)	Critical Band Rate (z)	Absolute Threshold (dB)
61	3186.91	15.955	-4.91
62	3273.05	16.110	-4.97
63	3359.18	16.260	-4.98
64	3445.31	16.406	-4.92
65	3531.45	16.547	-4.81
66	3617.58	16.685	-4.65
67	3703.71	16.820	-4.43
68	3789.84	16.951	-4.17
69	3875.98	17.079	-3.87
70	3962.11	17.205	-3.54
71	4048.24	17.327	-3.19
72	4134.38	17.447	-2.82
73	4306.64	17.680	-2.06
74	4478.91	17.905	-1.32
75	4651.17	18.121	-.64
76	4823.44	18.331	-.04
77	4995.70	18.534	.47
78	5167.97	18.731	.89
79	5340.23	18.922	1.23
80	5512.50	19.108	1.51
81	5684.77	19.289	1.74
82	5857.03	19.464	1.93
83	6029.30	19.635	2.11
84	6201.56	19.801	2.28
85	6373.83	19.963	2.46
86	6546.09	20.120	2.63
87	6718.36	20.273	2.82
88	6890.63	20.421	3.03
89	7062.89	20.565	3.25
90	7235.16	20.705	3.49

Table 3.5: Frequencies, Critical Bands, and Absolute Threshold

Index Number (i)	Frequency (Hz)	Critical Band Rate (z)	Absolute Threshold (dB)
91	7407.42	20.840	3.74
92	7579.69	20.972	4.02
93	7751.95	21.099	4.32
94	7924.22	21.222	4.64
95	8096.48	21.342	4.98
96	8268.75	21.457	5.35
97	8613.28	21.677	6.15
98	8957.81	21.882	7.07
99	9302.34	22.074	8.10
100	9646.88	22.253	9.25
101	9991.41	22.420	10.54
102	10335.94	22.576	11.97
103	10680.47	22.721	13.56
104	11025.00	22.857	15.31
105	11369.53	22.984	17.23
106	11714.06	23.102	19.34
107	12058.59	23.213	21.64
108	12403.13	23.317	24.15
109	12747.66	23.415	26.88
110	13092.19	23.506	29.84
111	13436.72	23.592	33.05
112	13781.25	23.673	36.52
113	14125.78	23.749	40.25
114	14470.31	23.821	44.27
115	14814.84	23.888	48.59
116	15159.38	23.952	53.22
117	15503.91	24.013	58.18
118	15848.44	24.070	63.49
119	16192.97	24.125	68.00
120	16537.50	24.176	68.00

Table 3.6: Frequencies, Critical Bands, and Absolute Threshold

Index Number (i)	Frequency (Hz)	Critical Band Rate (z)	Absolute Threshold (dB)
121	16882.03	24.225	68.00
122	17226.56	24.271	68.00
123	17571.09	24.316	68.00
124	17915.63	24.358	68.00
125	18260.16	24.398	68.00
126	18604.69	24.436	68.00
127	18949.22	24.473	68.00
128	19293.75	24.508	68.00
129	19638.28	24.542	68.00
130	19982.81	24.574	68.00

Table 3.7: Frequencies, Critical Bands, and Absolute Threshold

A local maximum is put in the list of tonal components if

$$P(k) - P(k+j) \geq 7\ dB$$

where j is chosen according to Table 3.10.

If $P(k)$ is found to be a tonal component, then the following parameters are listed:

- Index number k of the spectral line.
- Sound pressure level $P_{TM}(k) = P(k-1) + P(k) + P(k+1)$, in dB
- Tonal flag.

Next, all spectral lines within the examined frequency range are set to -8 dB.

- Listing of non-tonal components and calculation of the power

The non-tonal (noise) components are calculated from the remaining spectral lines. To calculate the non-tonal components from these spectral lines $P(k)$, the critical bands $z(k)$ are determined using Table 3.8. The 24 critical bands are used for 32 kHz sampling rate, and 26 critical bands are used for 44.1 kHz and 48 kHz sampling rate. Within each critical band,

Number	Index of Table	Frequency (Hz)	Bark (z)
0	1	43.066	.425
1	2	86.133	.850
2	3	129.199	1.273
3	5	215.332	2.112
4	7	301.465	2.934
5	10	430.664	4.124
6	13	559.863	5.249
7	16	689.063	6.301
8	19	818.262	7.274
9	22	947.461	8.169
10	26	1119.727	9.244
11	30	1291.992	10.195
12	35	1507.324	11.232
13	40	1722.656	12.125
14	46	1981.055	13.042
15	51	2325.586	14.062
16	56	2756.250	15.100
17	62	3273.047	16.11
18	69	3875.977	17.079
19	74	4478.906	17.904
20	79	5340.234	18.922
21	85	6373.828	19.963
22	92	7579.688	20.971
23	99	9302.344	22.074
24	105	11369.531	22.984
25	117	15503.906	24.013
26	130	19982.813	24.573

Table 3.8: Critical Band Boundaries

Sampling rate	*df*	Range
38 kHz	62.5 Hz	$0\,kHz < f \leq 3.0\,kHz$
38 kHz	93.75 Hz	$3.0\,kHz < f \leq 6.0\,kHz$
38 kHz	187.5 Hz	$6.0\,kHz < f \leq 12.0\,kHz$
38 kHz	375 Hz	$12.0\,kHz < f \leq 24.0\,kHz$
44.1 kHz	86.133 Hz	$0\,kHz < f \leq 2.756\,kHz$
44.1 kHz	129.199 Hz	$2.756\,kHz < f \leq 5.512\,kHz$
44.1 kHz	258.398 Hz	$5.512\,kHz < f \leq 11.024\,kHz$
44.1 kHz	516.797 Hz	$11.024\,kHz < f \leq 19.982\,kHz$
48 kHz	93.750 Hz	$0\,kHz < f \leq 3.0\,kHz$
48 kHz	140.63 Hz	$3.0\,kHz < f \leq 6.0\,kHz$
48 kHz	281.25 Hz	$6.0\,kHz < f \leq 12.0\,kHz$
48 kHz	562.50 Hz	$12.0\,kHz < f \leq 24.0\,kHz$

Table 3.9: Frequency Range of *df*

j	Range
-2, +2	$2 < k < 63$
-3, -2, +2, +3	$63 \leq k < 127$
-6,..., -2, +2,..., +6	$127 \leq k < 255$
-12,..., -2, +2,..., +12	$255 \leq k \leq 500$

Table 3.10: Tonal Component Conditions

the power of the spectral lines are summed to form the sound pressure level of the new non-tonal component corresponding to that critical band. The following parameters are listed:

- Index number k of the spectral line nearest to the geometric mean of the critical band.
- Sound pressure level $P_{NM}(k)$, in dB.
- Non-tonal flag.

3.2.5 Decimation of the masking components

Decimation [4] is a procedure that is used to reduce the number of maskers which are considered for the calculation of the global masking threshold.

(i) Tonal $P_{TM}(k)$ or non-tonal components $P_{NM}(k)$ are considered for the calculation of the masking threshold only if:

$$P_{TM}(k) >= T_q(k)$$

or

$$P_{NM}(k) >= T_q(k)$$

where, $T_q(k)$ is the absolute threshold at the frequency of index k. These values are given in Tables 3.3, 3.4, 3.5, 3.6 and 3.7.

(ii) Two or more tonal components within a distance of less then 0.5 Bark should be decimated. The component with the highest power must be kept, and the smaller components from the list of tonal components should be removed. A sliding window in the critical band domain will be used with a width of 0.5 Bark for this operation.

In the following, the index j is used to indicate the relevant tonal or non-tonal masking components from the combined decimated list.

Sampling rate	i
32 kHz	132
44.1 kHz	130
48 kHz	126

Table 3.11: Sampling Rate vs. Number of Samples

3.2.6 Calculation of masking thresholds

Of the original $N/2$ frequency domain samples, indexed by k, only a subset of the samples, indexed by i, are considered for the global masking threshold [4] calculation. The samples used are shown in Tables 3.3, 3.4, 3.5, 3.6 and 3.7.

For the frequency lines corresponding to the frequency region which is covered by the first three subbands no subsampling is used. For the frequency region which is covered by next three subbands every second spectral line is considered. For the frequency region corresponding to the next six subbands every fourth spectral line is considered. Finally, in the case of 44.1 and 48 kHz sampling rates, in the remaining subbands every eighth spectral line is considered up to 20 kHz. In the case of 32 kHz sampling rate, in the frequency region corresponding to the remaining subbands, every eighth spectral line is considered up to 15 kHz (Tables 3.3, 3.4, 3.5, 3.6 and 3.7).

The number of samples, i, in the subsampled frequency domain is different depending on the sampling rates (Table 3.11).

To every tonal and non-tonal component the index i in the subsampled frequency domain is assigned, which is closest in frequency to the original spectral line $P(k)$. This index i is given in Tables 3.3, 3.4, 3.5, 3.6 and 3.7.

The individual masking thresholds of both tonal and non-tonal components are given by the following expression:

$$T_{TM}[z(j), z(i)] = P_{TM}[z(j)] + AV_{TM}[z(j)] + VF[z(j), z(i)] \ (dB) \tag{3.8}$$

$$T_{NM}[z(j), z(i)] = P_{NM}[z(j)] + AV_{NM}[z(j)] + VF[z(j), z(i)] \ (dB). \tag{3.9}$$

In this formula T_{TM} and T_{NM} are the individual masking thresholds at critical band rate, z, in Bark of the masking component at the critical band rate, z, in Bark. The values in dB can be either positive or negative. The term $P_{TM}[z(j)]$ is the sound pressure level of the masking component with the index number j at the corresponding critical band rate $z(j)$. The term AV is called the masking index and VF the masking function of the masking component $P_{TM}[z(j)]$. The masking index AV is different for tonal (AV_{TM}) and non-tonal masker (AV_{NM}).

For tonal maskers it is given by

$$AV_{TM} = -1.525 - 0.275\, z(j) - 4.5\ (dB) \tag{3.10}$$

and for non-tonal maskers

$$AV_{NM} = -1.525 - 0.175\, z(j) - 0.5\ (dB). \tag{3.11}$$

The masking function VF of a masker is characterized by different lower and upper slopes, which depend on the distance in Bark $dz = z(i) - z(j)$ to the masker. In this expression i is the index of the spectral line at which the masking function is calculated and j is that of the masker. The critical band rates $z(j)$ and $z(i)$ can be found in Tables 3.3, 3.4, 3.5, 3.6 and 3.7. The masking function, which is the same for tonal and non-tonal maskers, is given by:

$$VF = \begin{cases} 17\,(dz+1) - (0.4\,P[z(j)] + 6)\ (dB) & , for -3 \leq dz < -1\ (Bark) \\ (0.4\,P[z(j)] + 6)\,dz\ (dB) & , for -1 \leq dz < 0\ (Bark) \\ -17\,dz\ (dB) & , for 0 \leq dz < 1\ (Bark) \\ -(dz-1)\,(17 - 0.15\,P[z(j)]) - 17\ (dB) & , for 1 \leq dz < 8\ (Bark) \end{cases} \tag{3.12}$$

In these expressions $P[z(j)]$ is the sound pressure level of the jth masking component in dB. If $dz < -3\ Bark$, or $dz \geq 8\ Bark$, the masking is no longer considered (T_{TM} and T_{NM} are set to -8 dB outside this range).

3.2.7 Global masking threshold

The global masking threshold $T_g(i)$ (Eq. 3.13) at the ith frequency sample [4] is derived from the upper and lower slopes of the individual masking threshold of each of the j tonal and non-tonal maskers, and in addition from the threshold in quiet $T_q(i)$. This is also given in Tables 3.3, 3.4, 3.5, 3.6 and 3.7. The global masking threshold is found by summing the powers corresponding to the individual masking thresholds and the threshold in quiet.

$$T_g(i) = 10 \log \left(10^{(0.1\, T_q(i))} + \sum_{l=1}^{L} 10^{(0.1\, T_{TM}(i,l))} + \sum_{m=1}^{M} 10^{(0.1\, T_{NM}(i,m))} \right) (dB) \tag{3.13}$$

The total number of tonal maskers is given by m, and the total number of non-tonal maskers is given by n. For a given i, the range of j can be reduced to just encompass those masking components that are within -8 to +3 Bark from i. Outside of this range T_{TM} and T_{NM} are -8 dB.

3.2.8 Minimum masking threshold

The minimum masking level $T_{min}(n)$ in subband n is determined [4] by the following expression:

$$T_{min}(n) = min[T_g(i)] \ (dB) \tag{3.14}$$

where $T_g(i)$ is the frequency of the ith frequency sample in subband n. The $T_g(i)$ are tabulated in Tables 3.3, 3.4, 3.5, 3.6 and 3.7. A minimum masking level $T_{min}(n)$ is computed for every subband.

3.2.9 Calculation of the SMR

The SMR [4] is computed for every subband n (Eq. 3.15).

$$SMR_{SB}(n) = L_{SB}(n) - T_{min}(n) \ (dB) \tag{3.15}$$

Chapter 4

Modified Discrete Cosine Transform

4.1 Theory

The output of the pseudo QMF filterbank [8] is prone to audible artifacts since it does not achieve perfect reconstruction (PR). To circumvent this, MDCT is added and together they are called the hybrid filterbank. MDCT constrains the sources of output distortion to the quantization stage. MDCT is a special case of perfect reconstruction cosine modulated filterbank with $L = 2M$. The MDCT analysis filter is given by

$$h_k(n) = w(n)\sqrt{\frac{2}{M}}\cos\left[\frac{(2n+M+1)(2k+1)\pi}{4M}\right] \tag{4.1}$$

and the synthesis filter is given by

$$g_k(n) = h_k(2M-1-n). \tag{4.2}$$

The synthesis filter is used to satisfy the overall linear phase constraint and obtained through a time reversal. This perspective is useful for visualizing individual channel characteristics in terms of their impulse and frequency responses. In practice, however, the filterbank is realized as a block transform.

4.1.1 Forward and inverse MDCT

The analysis filter bank [8] is realized as a block transform of length $2M$ samples, while using a block advance of only M samples. Thus, the MDCT basis functions extend across two blocks in time, leading to virtual elimination of the blocking artifacts that plague the reconstruction of non-overlapped transform coders. Despite the 50% overlap, however, the MDCT is still critically sampled, and only M coefficients are generated by the forward transform for each 2M-sample input block. Given an input block, $x(n)$, the transform coefficients, $X(k)$, for $0 \leq k \leq M-1$, are obtained by means of the forward MDCT, defined as

$$X(k) = \sum_{n=0}^{2M-1} x(n)h_k(n). \tag{4.3}$$

Clearly, the forward MDCT performs a series of inner products between the M analysis filter impulse responses, $h_k(n)$, and the input, $x(n)$. On the other hand, the inverse MDCT obtains a reconstruction by computing a sum of the basis vectors weighted by the transform coefficients from two blocks. The first M-samples of the kth basis vector, $h_k(n)$, for $0 \leq n \leq M-1$, are weighted by the kth coefficient of the current block, $X(k)$. Simultaneously, the second M-samples of the kth basis vector, $h_k(n)$, for $M \leq n \leq 2M-1$, are weighted by kth coefficient of the previous block, $X^P(k)$. Then, the weighted basis vectors are overlapped and added at each time index, n . Note that the extended basis functions require that the inverse transform maintain an M-sample memory to retain the previous set of coefficients. Thus, the reconstructed samples, $x(n)$, for $0 \leq n \leq M-1$, are obtained via the inverse MDCT, defined as

$$x(n) = \sum_{k=0}^{M-1} [X(k)h_k(n) + X^P(k)h_k(n+M)] \tag{4.4}$$

where $X^P(k)$ denotes the previous block of transform coefficients. The overlapped analysis and overlap-add synthesis processes are illustrated in Fig. 4.1.

Given the forward (Eq. 4.3) and inverse (Eq. 4.4) transform definitions, a suitable

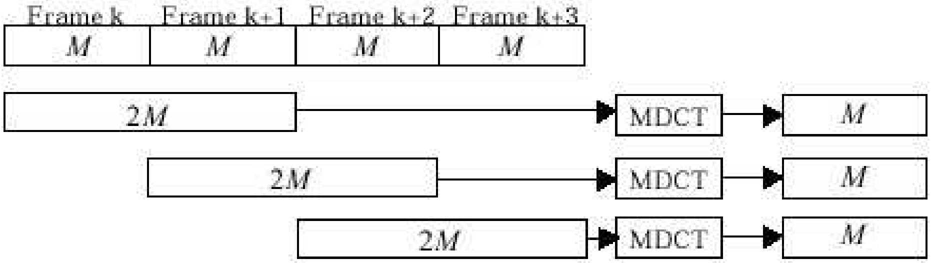

Figure 4.1: Lapped Forward Transform

FIR prototype filter, $w(n)$, should be designed. For the MDCT, the generalized PR conditions can be reduced to linear phase and Nyquist constraints on the window, namely,

$$w(2M-1-n) = w(n)w^2(n) + w^2(n+M) = 1 \tag{4.5}$$

for the sample indices $0 \leq n \leq M-1$.

It is possible to modify these constraints and reformulate the MDCT with unique analysis and synthesis windows using a biorthogonal construction. Several general purpose orthogonal and biorthogonal windows have been proposed, while still other orthogonal and biorthogonal windows are optimized explicitly for audio coding.

4.1.2 Modulated lapped transform

There are many kinds of MDCT windows [8]. The MP3 MDCT filterbank makes use of the sine window referred to as the MLT (Modulated Lapped Transform). It is defined as,

$$w(n) = \sin\left[\frac{\pi}{2M}(n+\frac{1}{2})\right] \tag{4.6}$$

for $0 \leq n \leq M-1$. This particular window is perhaps the most popular in audio coding. This is also used in the time-frequency filterbank of MPEG-2 AAC, MPEG-4 and many experimental coders. The sine window has several unique properties that make it advantageous. In particular, DC energy is concentrated in a single coefficient, the filterbank channels have 24 dB sidelobe attenuation, and it can be shown that the MLT is asymptotically optimal in terms of coding gain. Optimization criteria other than coding gain or DC localization have also been investigated.

4.1.3 Time-varying MLT windows

In perceptual audio coders [8], the characteristics of the best filterbank for audio are signal-specific and therefore time-varying. This must be considered in MDCT window design. It is very common for encoders using the MDCT to change the window length to match the signal properties of the input. MP3 in particular, uses a *long* window to maximize coding gain and achieve good channel separation during segments identified as stationary and a *short* window to localize time-domain artifacts when pre-echoes are likely. As a consequence, a time overlap between basis vectors emerges. To preserve perfect reconstruction when window switching occurs, either special transitional windows or boundary filters are required. Other schemes are also available but for practical reasons these are not typically used. This is called mode switching which is employed by the MPEG MDCT-based encoders and the Dolby AC-3 algorithm. MP3 opt to use transitional windows called *start* window and *stop* window. Unlike MPEG, AC-3 [3] do not use transitional windows to maintain perfect reconstruction. The spectral and temporal analysis tradeoffs involved in transitional window designs are well illustrated in the MP3 filterbanks.

4.1.4 Fast algorithms and complexity issues

One of the attractive properties that has contributed to the widespread use of the MDCT, particularly in the standards, is the availability of FFT-based fast algorithms that make the filterbank viable for real-time applications. For example, a unified fast algorithm is available for the MPEG-1, MPEG-2, MPEG-4, and AC-3 long block MDCT, the AC-3 short block MDCT, and the MPEG-1 pseudo QMF filterbank. A regressive structure suitable for parallel VLSI implementation of the Eq. 4.3 MDCT was also proposed. As far as quantization sensitivity is concerned, there are available expressions for the reconstruction error of the quantized system in terms of signal-correlated and uncorrelated components that can be used to identify perceptually disturbing reconstruction artifacts.

4.2 Implementation

4.2.1 Window switching

Window switching [4] is implemented to contribute to the pre-echo suppression. Window switching works by changing the analysis block length from *long* duration (e.g., 25 ms) during stationary segments to *short* duration (e.g., 4 ms) when transients are detected. At least two considerations motivate this method. First, a *short* window applied to the frame containing the transient will tend to minimize the temporal spread of quantization noise such that temporal premasking effects might preclude audibility. Second, it is desirable to constrain the high bit rates associated with transients to the shortest possible temporal regions.

Although window switching has been successful, it also has significant drawbacks. For one, the perceptual model and lossless coding portions of the coder must support multiple time resolutions. Furthermore, most modern coders use the MLT. To satisfy PR constraints, window switching typically requires transition windows between the long and short blocks. Even when suitable transition windows satisfy the PR constraints, they do so at the expense of poor time and frequency localization properties, resulting in reduced coding gain. Other difficulties inherent to window switching schemes are increased coder delay, undesirable latency for closely spaced transients (e.g., long-start-short-stop-start-short), and impractical overusage of short windows for pitched signals.

The decision whether the filterbank should be switched to short windows is derived from the calculation of the masking threshold by calculating the estimate of the

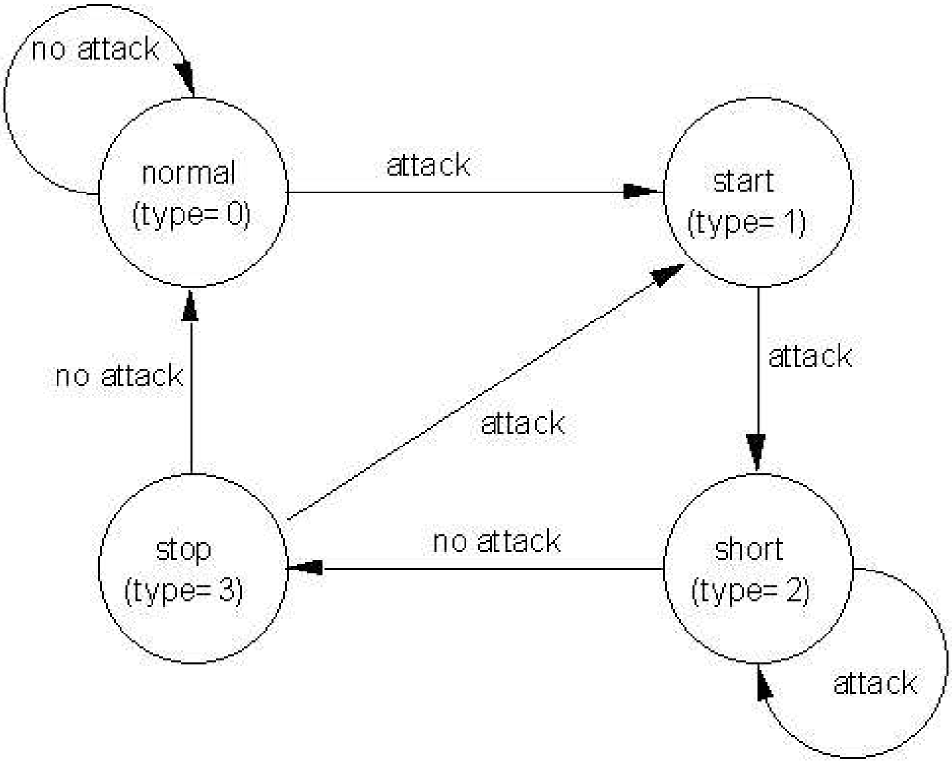

Figure 4.2: Window Switching Logic

psychoacoustic entropy (PE) and switching when the PE exceeds the value 1800. If this condition is met, the sequence start (`block_type`=1), short (`block_type`=2), short, stop (`block_type`=3) is started. Fig. 4.2 shows the possible state changes for the window switching logic.

4.2.2 Window types

The output of the polyphase filterbank is the input to the subdivision using the MDCT. According to the output of the psychoacoustic model [4] the window and transform types normal (long), start, short or stop are used.

The 18 consecutive output values of one granule and 18 output values of the granule before are assembled to one block of 36 samples.

Block type *normal*

$$z_i = x_i \sin\left(\frac{\pi}{36}(i + \frac{1}{2})\right) \text{ , for i = 0 to 35.} \tag{4.7}$$

Block type *start*

$$z_i = \begin{cases} x_i \sin\left(\frac{\pi}{36}(i + \frac{1}{2})\right) & \text{, for i = 0 to 17} \\ x_i & \text{, for i = 18 to 23} \\ x_i \sin\left(\frac{\pi}{12}(i - 18 + \frac{1}{2})\right) & \text{, for i = 24 to 29} \\ 0 & \text{, for i = 30 to 35} \end{cases} \tag{4.8}$$

Block type *stop*

$$z_i = \begin{cases} 0 & \text{, for i = 0 to 5} \\ x_i \sin\left(\frac{\pi}{12}(i - 6 + \frac{1}{2})\right) & \text{, for i = 6 to 11} \\ x_i & \text{, for i = 12 to 17} \\ x_i \sin\left(\frac{\pi}{36}(i + \frac{1}{2})\right) & \text{, for i = 18 to 35} \end{cases} \tag{4.9}$$

Block type *short*

The block of 36 samples is divided into three overlapping blocks:

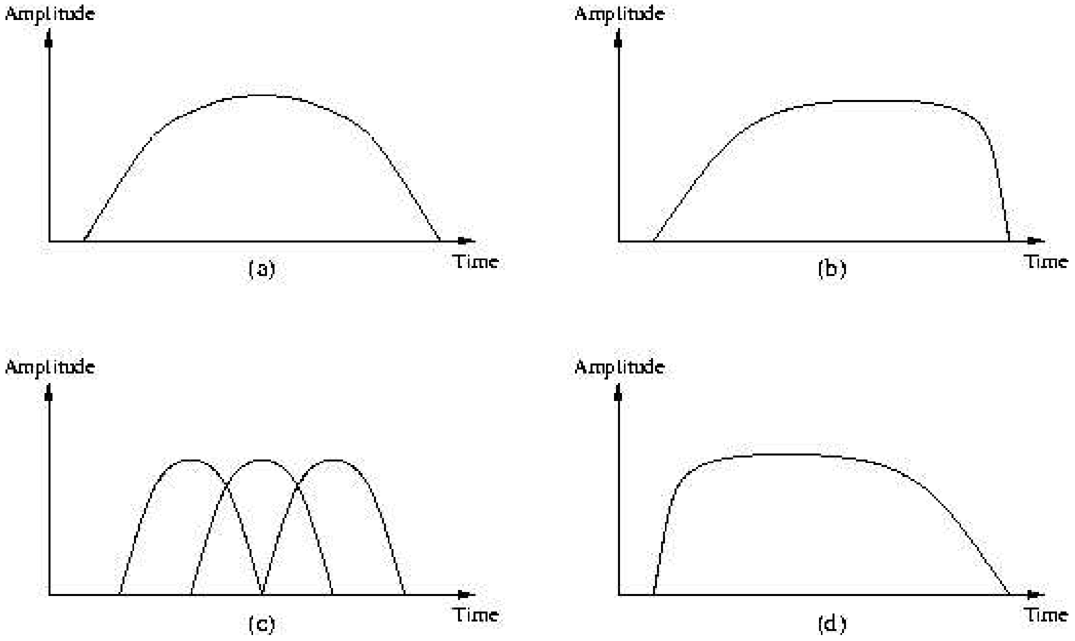

Figure 4.3: (a) Normal, (b) Start, (c) Short, and (d) Stop

$$y_i^{(k)} = x_{i+6(k+1)} \text{ , for i = 0 to 11, k = 0 to 2.} \tag{4.10}$$

Each of the three small blocks is windowed separately:

$$z_i^{(k)} = y_i^{(k)} \sin\left(\frac{\pi}{12}(i+\frac{1}{2})\right) \text{ , for i = 0 to 11, k = 0 to 2.} \tag{4.11}$$

In the following n is the number of windowed samples. For short blocks n is 12, for long blocks n is 36. The analytical expression of the MDCT is:

$$X_i = \sum_{k=0}^{n-1} z_k \cos\left(\frac{\pi}{2n}\left(2k+1+\frac{n}{2}\right)(2i+1)\right). \tag{4.12}$$

These windows are graphically depicted in Fig. 4.3.

Aliasing Butterfly Encoder

The calculation of aliasing reduction [4] in the encoder is performed. Figure 4.4 shows the general procedure while Figure 4.5 details the butterfly definition to be

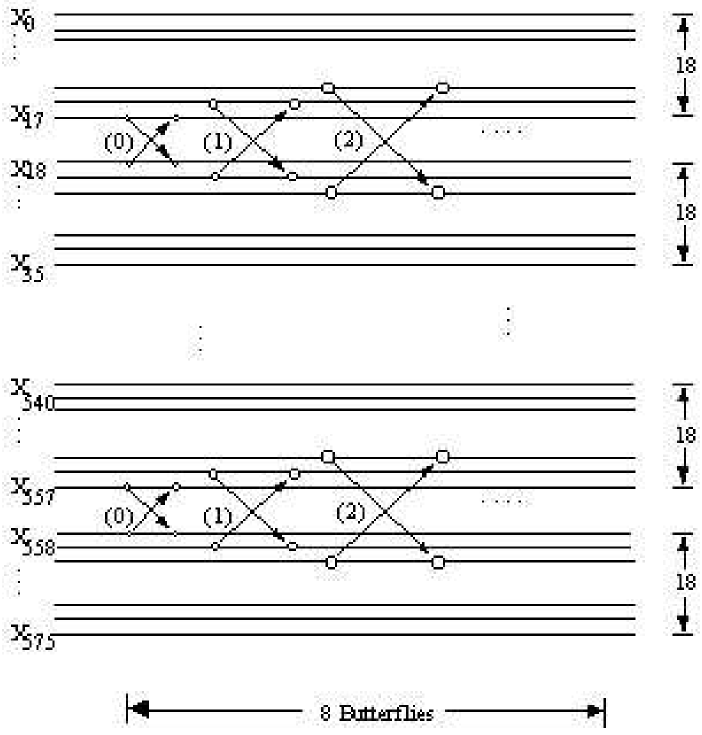

Figure 4.4: Aliasing Butterfly

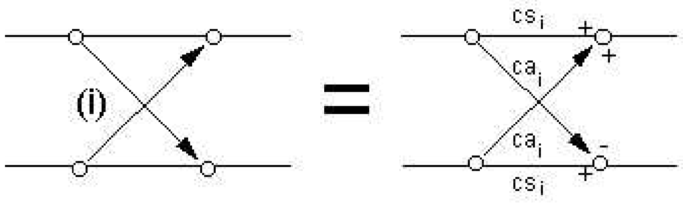

Figure 4.5: Butterfly Definition

i	c_i
0	-0.6
1	-0.535
2	-0.33
3	-0.185
4	-0.095
5	-0.041
6	-0.0142
7	-0.0037

Table 4.1: Aliasing Reduction c_i Coefficients

used. The butterfly coefficients cs_i and ca_i are calculated using Eq. 4.13 and 4.14, respectively. The coefficients c_i can be found in Table 4.1.

$$cs_i = \frac{1}{\sqrt{1 + c_i^2}} \tag{4.13}$$

$$ca_i = \frac{c_i}{\sqrt{1 + c_i^2}} \tag{4.14}$$

Chapter 5

Quantization

5.1 Noise Allocation

The following subsections are done in preparation for the iteration loops which is the topic of the next section.

5.1.1 Average available bits

The average number of bits per granule [4] is calculated from the frame size. The bit rate 64 kbps is used as an example. At bit rate 64 kbps at 48000 samples per second,

$$\frac{(64000*0.24\ bits\ per\ frame)}{(2\ granules\ per\ frame)} = 768\ bits\ per\ granule$$

As the header takes 32 bits and the side information takes 17 bytes (136 bits) in `single_channel mode`, the average amount of available bits for `the main_data` for a granule is given by

$$\texttt{mean\_bits} = 768\ bits\ per\ granule - \frac{(32+136\ bits\ per\ frame)}{(2\ granules\ per\ frame)} = 684\ bits\ per\ granule$$

5.1.2 Reset of all iteration variables

The following variables are all set to zero [4]:

- `scalefac[cb]` - the scalefactors of the coder partitions
- `qquant` - the counter for the quantizer step size

- preflag
- scalefac_scale

The inital value of quantanf is set as follows:

$$\texttt{quantanf} = \texttt{system\_const} * ln\ \texttt{sfm} \tag{5.1}$$

where sfm is the spectral flatness measure and quantanf depends on the computational implementation of the encoder.

The spectral flatness measure sfm is given by

$$\texttt{sfm} = \frac{e^{\frac{1}{n}(\sum_{i=0}^{n} \log xr(i)^2)}}{\frac{1}{n}(\sum_{i=0}^{n} xr(i)^2)}. \tag{5.2}$$

The value of system_const is chosen so that for all signals the first iteration of the inner loop for all signals comes out with a bit sum higher than the desired bitsum. By that it is ensured that the first call of the inner loop results in the solution which uses as many of the available bits as possible. In order to spare computing time it is desirable to minimize the number of iterations by adapting the value of quantanf to the bit rate and the signal statistics.

5.1.3 Bit reservoir control

Bits are saved to the reservoir [4] when fewer than the mean_bits are used to code one granule. If bits are saved for a frame, the value of main_data_end is increased accordingly. The bitstream organization is found in Fig. 5.1.

The number of bits which are made available for the main_data (max_bits) is derived from the actual estimated threshold (the PE as calculated by the psychoacoustic model), the average number of bits (mean_bits) and the actual content of the bit reservoir. The number of bytes in the bit reservoir is given by main_data_end.

The actual rules for the control of the bit reservoir are given below:

- If a number of bytes available to the inner iteration loop is not used for the Huffman encoding or other main_data, the number is added to the bit

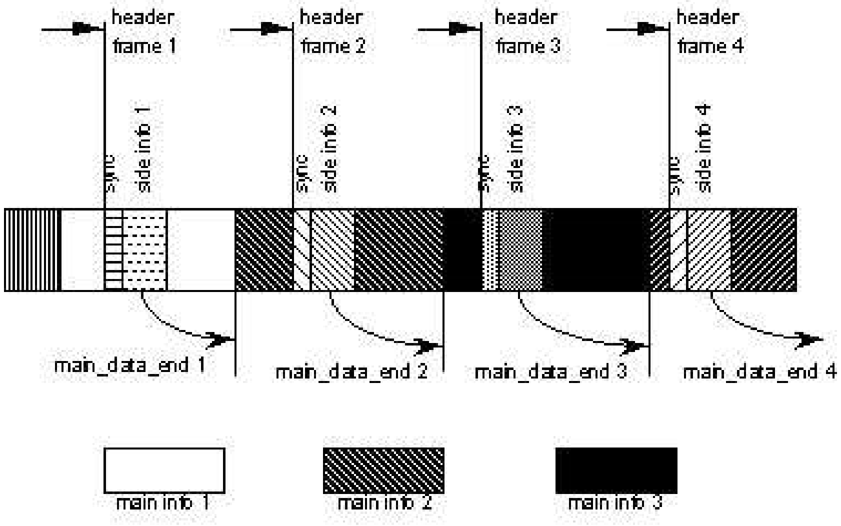

Figure 5.1: Bitstream Organization

reservoir.

- If the bit reservoir contains more than 0.8 times the maximum allowed content of the bit reservoir, all bytes exceeding this number are made available for `main_data` (in addition to `mean_bits`).
- If more_bits is greater than 100 bits, then max(`more_bits`/8, 0.6 * `main_data_end`) bytes are taken from the bit reservoir and made available for `main_data` (in addition to `mean_bits`).
- After the actual loops computations have been completed, the number of bytes not used for `main_data` is added to the bit reservoir.
- If after the step above the number of bytes in the bit reservoir exceeds the maximum allowed content, stuffing bits are written to the bitstream and the content of the bit reservoir is adjusted accordingly.

5.1.4 Scalefactor select information (scfsi)

The scfsi [4] contains the information, which scalefactors (grouped in the scfsi_bands) of the first granule can also be used for the second granule. These scalefactors are therefore not transmitted, the gained bits can be used for the Huffman coding. To determine the usage of the scfsi, the following information of each granule must be stored:

1. The block type.

2. The total energy of the granule:

$$\texttt{en\_tot} = \left\lceil \log_2 \left(\sum_{i=1}^{n} |xr(i)|^2 \right) \right\rceil \tag{5.3}$$

   where n is the total number of spectral values.

3. The energy of each scalefactor band:

$$en(cb) = \left\lceil \log_2 \left(\sum_{i=lbl(cb)}^{lbl(cb)+bw(cb)-1} |xr(i)|^2 \right) \right\rceil \tag{5.4}$$

   where $lbl(cb)$ is the number of the first coefficient belonging to scalefactor band cb and $bw(cb)$ is the number of coefficients within scalefactor band cb.

4. The allowed distortion of each scalefactor band:

$$xm(cb) = \lceil \log_2 |xmin(i)| \rceil \tag{5.5}$$

   $xmin(cb)$ is calculated by the psychoacoustic model.

The scalefactors of the first granule are always transmitted. When coding the second granule, the information of the two granules is compared. There are four criteria to determine if the scfsi can be used in general. If one of the four is not fulfilled, the scfsi is disabled (that means it is set to 0 in all scfsi_bands). The criteria are (index 0 means first granule, index 1 second):

1. The spectral values are not all zero.
2. None of the granules contains short blocks.
3. $|\texttt{en\_tot}_0 - \texttt{en\_tot}_1| < \texttt{en\_tot}_{krit}$.
4. $$\sum_{all\ scalefactor\ bands} |en(cb)_0 - en(cb)_1| < \texttt{en\_dif}_{krit}.$$

If the `scfsi` is not disabled after the tests above, there are two criterias for each `scfsi_band`, which have both to be fulfilled to enable `scfsi` (that means to set it to 1 in this `scfsi_band`):

1. $$\sum_{all\ cb's\ in\ \texttt{scfsi\_band}} |en(cb)_0 - en(cb)_0| < en(\texttt{scfsi\_band})_{krit}.$$
2. $$\sum_{all\ cb's\ in\ \texttt{scfsi\_band}} |xm(cb)_0 - xm(cb)_0| < xm(\texttt{scfsi\_band})_{krit}.$$

The constants (with the index *krit*) have to be chosen so, that the `scfsi` is only enabled in case of similar energy and distortion. Suggested values are:

- `en_tot` = 10
- `en_dif` = 100
- en(`scfsi_band`) = 10, for each `scfsi_band`
- xm(`scfsi_band`) = 10, for each `scfsi_band`

5.2 Iteration Loops

The frequency domain data are quantized and coded within two nested iteration loops [4].

The description of the loop module is subdivided into three levels. The top level is called *loops frame program*. The loops frame program calls a subroutine named *outer iteration loop* which calls the subroutine *inner iteration loop*. For each level a corresponding flow diagram is shown.

The loops module quantizes an input vector of spectral data in an iterative process according to several demands. The inner loop quantizes the input vector and increases the quantizer step size until the output vector can be coded with the available amount of bit. After completion of the inner loop an outer loop checks the distortion of each scalefactor band and, if the allowed distortion is exceeded, amplifies the scalefactor band and calls the inner loop again.

Loops module input

1. vector of the magnitudes of the spectral values $xr(0..575)$
2. $xmin(cb)$, the allowed distortion of the scalefactor bands
3. `blocksplit_flag` which in conjunction with `switch_point` determines the number of scalefactor bands
4. `mean_bits` (bit available for the Huffman coding and the coding of the scalefactors)
5. `more_bits`, the number of bits in addition to the average number of bits, as demanded by the value of the psychoacoustic entropy for the granule:

   `more_bits` $= 3.1 * PE - (average\ number\ of\ bits)$

Loops module output

1. vector of quantized values $ix(0..575)$
2. $ifq(cb)$, the scalefactors
3. `qquant` (quantizer step size information)
4. number of unused bit available for later use

5. preflag (loops preemphasis on/off)

6. Huffman code related side information

   - big_values (number of pairs of Huffman coded values, excluding count1)
   - count1table_select (Huffman code table of absolute values ≤ 1 at the upper end of the spectrum
   - table_select[0..2](Huffman code table of regions)
   - region_address1 and region_address2 (used to calculate boundaries between regions)
   - part2_3_length

5.2.1 Outer iteration loop

The outer iteration loop (Fig. 5.3) functions as a distortion control loop [4]. It controls the quantization noise which is produced by the quantization of the frequency domain lines within the inner iteration loop. The colouration of the noise is done by multiplication of the lines within scalefactor bands with the actual scalefactors before doing the quantization. The following pseudo code illustrates the multiplication.

```
do for each scalefactor band
  do from lower index to upper index of scalefactor band
    xr(i) = xr(i) * $\sqrt{2}^{[(1+\texttt{scalefac\_scale})*\texttt{ifq}(\texttt{scalefactor band})]}$
  end do
end do
```

In the actual system the multiplication is done incrementally with just the increase of the scalefactors applied in each distortion control loop.

The distortion loop is always starting with scalefac_scale = 0. If after some iterations the maximum length of the scalefactors would be exceeded, then scalefac_scale is increased to the value 1 thus increasing the possible dynamic range

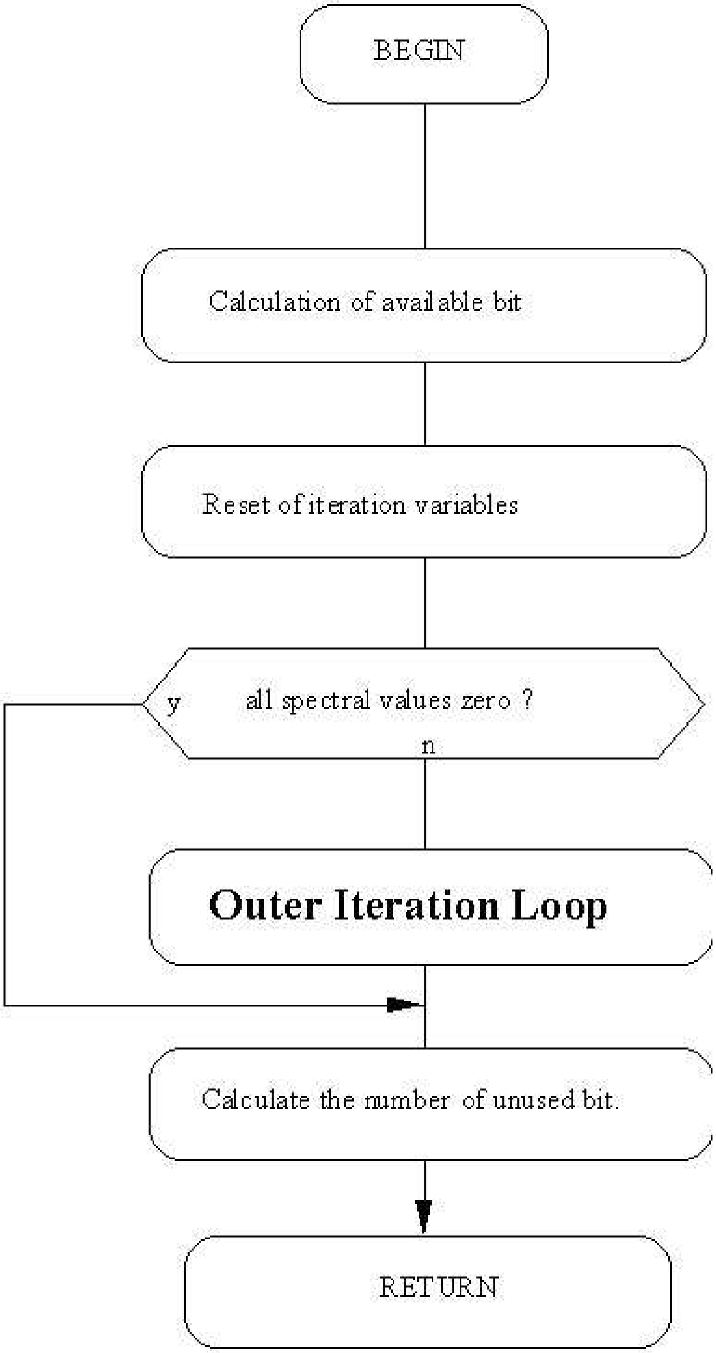
BEGIN
Calculation of available bit
Reset of iteration variables
y
all spectral values zero ?
n
Outer Iteration Loop
Calculate the number of unused bit.
RETURN

Figure 5.2: The Iteration Loops

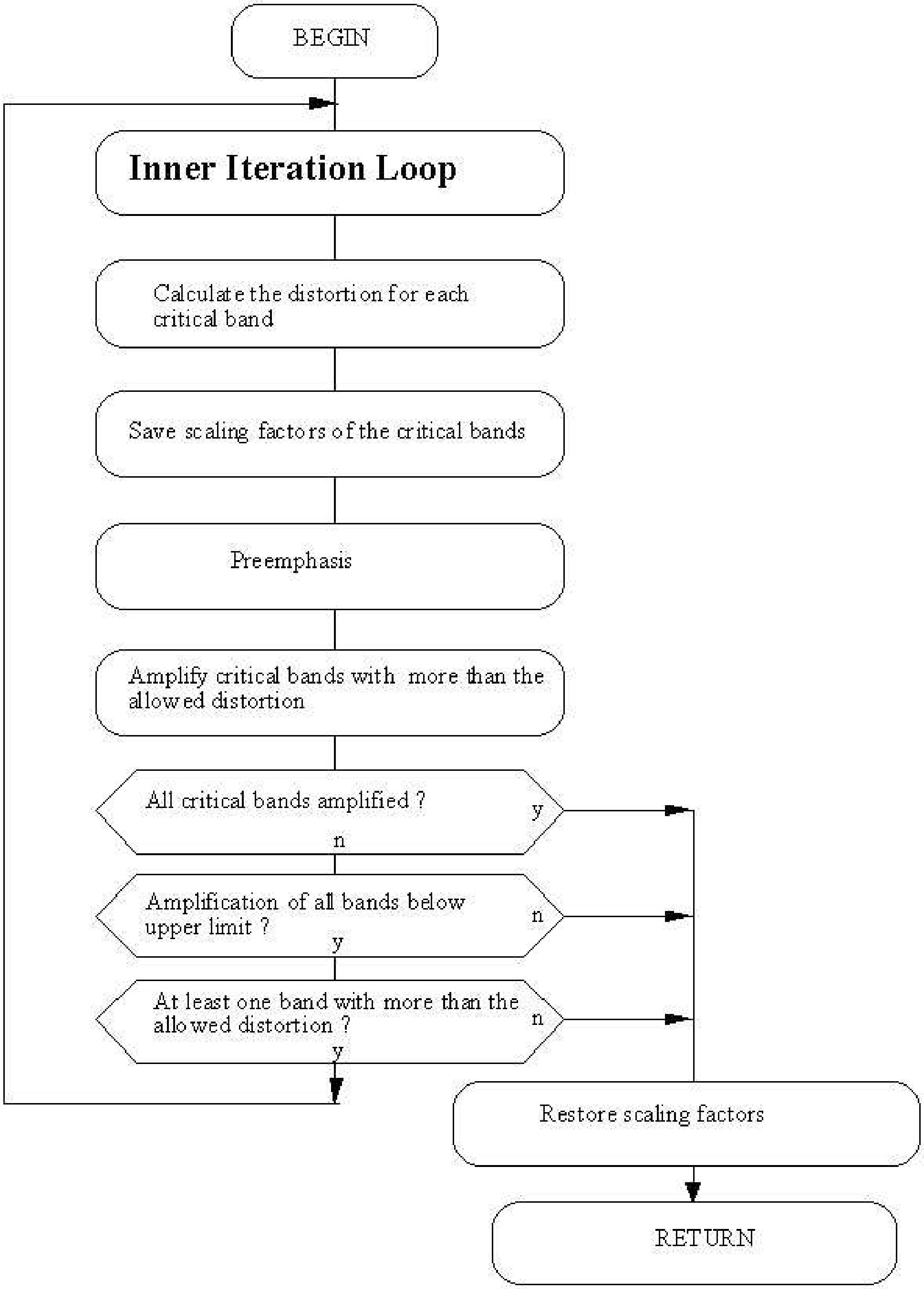
BEGIN
Inner Iteration Loop
Calculate the distortion for each critical band
Save scaling factors of the critical bands
Preemphasis
Amplify critical bands with more than the allowed distortion
All critical bands amplified ?
y
n
Amplification of all bands below upper limit ?
n
y
At least one band with more than the allowed distortion ?
n
y
Restore scaling factors
RETURN

Figure 5.3: Outer Iteration Loop

of the scalefactors. In this case the actual scalefactors and frequency lines have to be corrected accordingly.

Saving of the scalefactors

The scalefactors of all scalefactor bands $ifq(cb)$ as well as the quantizer step size `qquant` are saved. If the computation of the outer loop is cancelled without having reached a proper result this values together with the quantized spectrum give an approximation and can be transmitted.

Call of inner iteration loop

For each outer iteration loop (distortion control loop) the inner iteration loop (rate control loop) is called. The parameters are the frequency domain values (hybrid filterbank output) with the scalefactors applied to the values within the scalefactor bands and the number of bits which are available to the rate control loop. The result is the number of bits actually used and the quantized frequency lines $ix(i)$.

Calculation of the distortion of the scalefactor bands

For each scalefactor band the actual distortion is calculated according to:

$$xfsf(cb) = \sum_{i=lbl(cb)}^{lbl(cb)+bw(cb)-1} \frac{(|xr(i)| - ix(i)^{\frac{4}{3}} * \sqrt[4]{2}^{\texttt{qquant+quantanf}})^2}{bandwidth(cb)} \tag{5.6}$$

where $lbl(cb)$ is the number of the coefficient representing the lowest frequency in a scalefactor band and $bw(cb)$ is the number of coefficients within this band.

Preemphasis

The preemphasis option (switched on by setting `preflag` to a value of 1) provides the possibility to amplifiy the upper part of the spectrum.

```
if preflag==1
{
   $xmin(j) = xmin(j) * ifqstep2 * prefact(j)$
   for(i = lower limit of scalefactor band j;i ≤ upper limit of
      scalefactor band j;i++)
   {
    xr(i) = xr(i) * ifqstepprefact(j)
  }
}
```

The condition to switch on the preemphasis is up to the implementation. For example preemphasis could be switched on if in all of the upper 4 scalefactor bands the actual distortion exceeds the threshold after the first call of the inner loop. If the second granule is being coded and `scfsi` is active in at least one `scfsi_band`, the preemphasis in the second granule is set equal to the setting in first granule.

Amplification of scalefactor bands which violate the masking threshold

All spectral values of the scalefactor bands which have a distortion that exceeds the allowed distortion are amplified by a factor of `ifqstep`. The value of `ifqstep` is transmitted by `scalefac_scale`.

```
if (xmin - xfsf) of scalefactor band j < 0
{
  xmin(j) = xmin(j) * ifqstep2
  ifq(j) = ifq(j) + 1
  for (i = lower limit of scalefactor band; i ≤ upper limit of
    scalefactor band; i++)
  {
    xr(i) = xr(i) * ifqstep
  }
}
```

If the second granule is being coded and `scfsi` is active in at least one `scfsi_band`, the following steps have to be done:

1. `ifqstep` has to be set similar to the first granule.
2. If it is the first iteration, the scalefactors of scalefactor bands in which `scfsi` is enabled have to be taken over from the first granule. The corresponding spectral values have to be amplified:

```
if (scfsi according to scalefactor band j = 1)
{
  ifq(j) = ifq(j) first granule
  for (i = lower limit of scalefactor band; i ≤ upper limit of
    scalefactor band; i++)
  {
    xr(i) = xr(i) * ifqstepifq(j)
  }
}
```

3. If it is not the first iteration, the amplification must be prevented for scalefactor bands in which `scfsi` is enabled.

Conditions for the termination of the loops processing

Normally the loops processing terminates if there is no scalefactor band with more than the allowed distortion. However this is not always possible to obtain. In this case there are other conditions to terminate the outer loop:

1. All scalefactor bands are already amplified.
2. The amplification of at least one band exceeds the upper limit which is determined by the transmission format of the scalefactors. The upper limit is a scalefactor of 15 for scalefactor bands 0 through 10 and 7 for scalefactors 11 through 20.

The loops processing stops and by restoring the saved $ifq(cb)$ a useful output is available. For realtime implementation there might be a third condition added which terminates the loops in case of a lack of computing time.

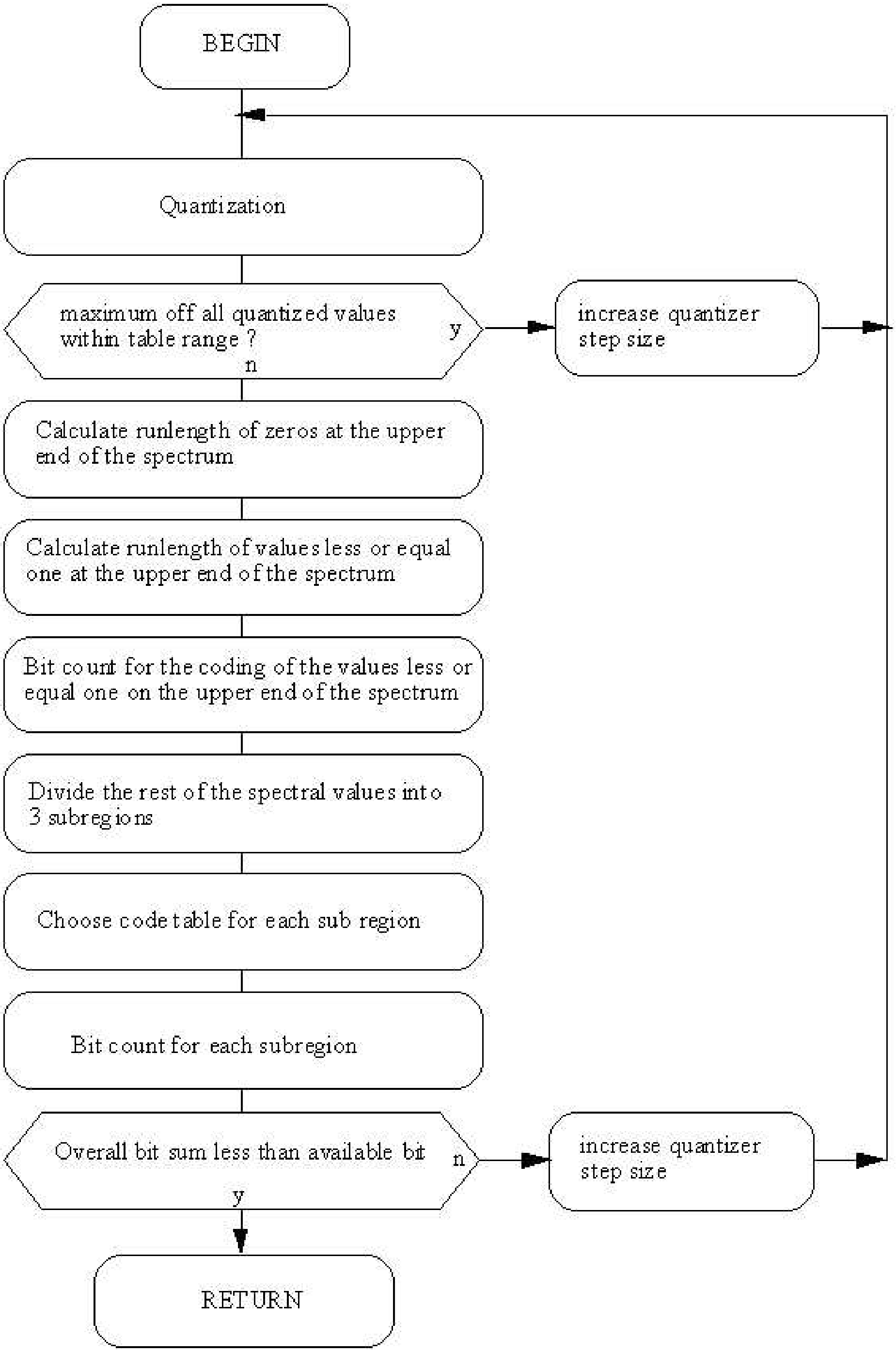
BEGIN
Quantization
maximum off all quantized values within table range ?
y
n
increase quantizer step size
Calculate runlength of zeros at the upper end of the spectrum
Calculate runlength of values less or equal one at the upper end of the spectrum
Bit count for the coding of the values less or equal one on the upper end of the spectrum
Divide the rest of the spectral values into 3 subregions
Choose code table for each sub region
Bit count for each subregion
Overall bit sum less than available bit
n
y
increase quantizer step size
RETURN

Figure 5.4: Inner Iteration Loop

5.2.2 Inner iteration loop

The inner iteration loop functions as a nonuniform quantization rate control loop. It does the actual quantization of the frequency domain data and prepares the formatting. The table selection, subdivision of the `big_values` range into regions and the selection of the quantizer step size takes place here. A flowchart of the process is illustrated in Fig. 5.4.

5.3 Huffman Coding

The quantization of the complete vector of spectral values is done according to

$$ix(i) = \left\lceil \left(\frac{|xr(i)|}{\sqrt[4]{2}^{qquant+quantanf}} \right)^{\frac{3}{4}} - 0.0946 \right\rceil . \tag{5.7}$$

Test of the maximum of the quantized values

The maximum allowed quantized value is limited. This limit is set to constraint the table size if a table lookup is used to requantize the quantized frequency lines. The limit is given by the possible values of the length identifier, `linbits`, of values flagged with an ESC code. Therefore before any bit counting is done the quantizer stepsize is increased by

```
qquant = qquant + 1
```

until the maximum of the quantized values is within the range of the largest Huffman code table.

Calculation of the run length of zeros

The run length of pairs of spectral coefficients quantized to zero on the upper end of the spectrum is counted and called `rzero`.

Calculation of the run length of values less or equal one

The run length of quadrupels of spectral coefficients quantized to one or zero, following the `rzero` pairs of zeros, is calculated and called `count1`.

Counting the bit necessary to code the values less or equal one

One Huffman code word is used to code one of the `count1` quadrupels. There are two different Huffman code books with corresponding code length tables. The number of bits to code all the `count1` quadrupels is given by:

`bitsum_count1 = min(bitsum_table0, bitsum_table1)`
where `count1table_0` is used to point to Table 5.1,

$$\begin{aligned} \texttt{bitsum\_table0} &= \textstyle\sum_{k=\texttt{firstcount1}}^{\texttt{firstcount1+count1}-1} \texttt{count1table\_0}(ix(4k) + 2 * ix(4k+1) \\ &+ 4 * ix(4k+2) + 6 * ix(4k+3)) \end{aligned} \tag{5.8}$$

and `count1table_1` is used to point to Table 5.2.

$$\begin{aligned} \texttt{bitsum\_table1} &= \textstyle\sum_{k=\texttt{firstcount1}}^{\texttt{firstcount1+count1}-1} \texttt{count1table\_1}(ix(4k) + 2 * ix(4k+1) \\ &+ 4 * ix(4k+2) + 6 * ix(4k+3)) \end{aligned} \tag{5.9}$$

The information which table is used is transmitted by `count1table_select`, which is 0 for Table 5.1 or 1 for Table 5.2.

Call of subroutine `SUBDIVIDE`

The number of pairs of quantized values not counted in `count1` or `rzero` is called `bigvalues`. `SUBDIVIDE` splits the scalefactor bands corresponding to this values into three groups. The last one, incomplete generally, counts as a complete one. The `region_adress1` and `region_adress2` contains the number of scalefactor bands in the first and the last region, respectively. The number of scalefactor bands in the second region can be calculated using `bigvalues`. If `bigvalues` comprises only two scalefactor bands `region_adress2` is set to zero. If there are less than two also `region_adress1` is zero. The split strategy is up to the implementation. A very simple one for instance is to assign $\frac{1}{3}$ of the scalefactor bands to the first and $\frac{1}{4}$ to the last region.

`SUBDIVIDE` in case of blocksplit is done analogous but only two subregions. The

Value	`hlen`	`hcod`
0000	1	1
0001	4	0101
0010	4	0100
0011	5	00101
0100	4	0110
0101	6	000101
0110	5	00100
0111	6	000100
1000	4	0111
1001	5	00011
1010	5	00110
1011	6	000000
1100	5	00111
1101	6	000010
1110	6	000011
1111	6	000001

Table 5.1: Huffman Code Table for Quadruples (A)

Value	`hlen`	`hcod`
0000	4	1111
0001	4	1110
0010	4	1101
0011	4	1100
0100	4	1011
0101	4	1010
0110	4	1001
0111	4	1000
1000	4	0111
1001	4	0110
1010	4	0101
1011	4	0100
1100	4	0011
1101	4	0010
1110	4	0001
1111	4	0000

Table 5.2: Huffman Code Table for Quadruples (B)

`region_address1` is set to a default in this case. This default is 8 in the case of `split_point = 0` and 9 in the case of `split_point = 1`. Both this values point to the same absolute frequency.

Calculation of the code book for each subregion

There are 32 different Huffman code tables for the coding of pairs of quantized values available. They differ from each other in the maximum value that can be coded and in the signal statistic they are optimized for. Only codes for values < 16 are in the table. For values ≥ 16 there are two tables provided where the largest value 15 is an escape character. In this case the value 15 is coded in an additional word using a linear PCM code with a word length called `linbits`. The `linbits` can be calculated by taking the base 2 logarithm of the PCM code, which is x - `max_table_entry`.

A simple way to choose a table is to use the maximum of the quantized values in a subregion. Tables which have the same size are optimized for different signal statistics. Therefore additional coding gain can be achieved for example by trying all of this tables.

Counting of the bit necessary to code the values in the subregions

The number of bits necessary to code the quantized values of one subregion is given by:

$$\begin{aligned}\texttt{bitsum}(j) \quad &= \quad \textstyle\sum_{k=0}^{np(j)-1} bitz(\texttt{tableselect}(j), \\ & \qquad min(15, ix(2k+fe(j))), min(15, ix(2k+fe(j)+1))) \\ &+ \quad \textstyle\sum_{k=0}^{np(j)-1}(s(ix(2k+fe(j)) - 15) \\ &+ \quad s(ix(2k+fe(j)+1) - 15)) * \texttt{linbits}(j)\end{aligned} \tag{5.10}$$

where $np(j)$ is the number of pairs in a subregion, $fe(j)$ is the number of the first quantized value in a subregion, $bitz$ is the table with Huffman code length, $s(x)$ is the step function defined as:

$is(i), i = 0...575$	quantized frequency domain values
`table_select[region]`	Huffman code table used for regions
`region_adress1`	defines the border between region 0 and 1
`region_adress2`	defines the border between region 1 and 2
`max_value[region]`	maximum absolute value of quantized data in regions

Table 5.3: Inner Iteration Variables

$$s(x) = \begin{cases} 1 & , if\ x \geq 0 \\ 0 & , if\ x < 0 \end{cases} \tag{5.11}$$

The Huffman codewords [4] are in sequence from low to high frequencies. Table 5.3 contains the calculated variables used in encoding the Huffman codewords in the iteration loops.

The actual assembly of the Huffman code for the `big_values` part is described in the following pseudo high level language:

```
for region number from 0 to 2
  if table_select for this region is 0
    nothing to do, all values in region are zero
  else
    if table_select for this region is > 15
      an ESC-table is used:  look up linbits value connected to the
        table used
      for i = begin of region to end of region, count in pairs
        x = is(i), y = is(i+1)
        if x > 14
          linbitsx = x - 15, x = 15
        end if
        signx = sign(x), x = abs(x)
        if y > 14
          linbitsy = y - 15, y = 15
        end if
        signy = sign(y), y = abs(y)
        look for codeword = hcod([x][y]]) in table table_seletc
        write hcod([x][y]), beginning with the leftmost bit, number
          of bits is hlen([x][y])
        if x > 14
          write linbitsx to the bitstream, number of bits is linbits
        end if
        if x != 0
          write signx to bitstream
        end if
        if y > 14
          write linbitsy to the bitstream, number of bits is linbits
        end if
        if y != 0
          write signy to bitstream
        end if
      end for
    else
      no ESC-words are used in this region:
      for i = beginning of region to end of region, count in pairs
        x = is(i), y = is(i+1)
        signx = sign(x), x = abs(x)
        signy = sign(y), y = abs(y)
        look for codeword = hcod([x][y]) in table table_seletct
        write hcod([x][y]), beginning with the leftmost bit, number
          of bits is hlen([x][y])
        if x != 0
          write signx to bitstream
        end if
```

```
        if y != 0
          write signy to bitstream
        end if
      end for
    end if
  end if
end for
```

A possible application for the `private_bits` is to use them as frame counter.

Chapter 6

Bitstream Formatting

6.1 Audio Frame

An audio frame [4] is a part of the bitstream that is decodable with the use of previously acquired side and main information. It contains information for 1152 samples. Although the distance between the start of consecutive syncwords is an integer number of slots (one byte), the audio information belonging to one frame is generally not contained between two successive syncwords. The audio frame is composed of four main components:

- `header` - part of the bitstream containing synchronization and state information.
- `error_check` - part of the bitstream containing information for error detection.
- `audio_data` - part of the bitstream containing information on the audio samples.
- `ancillary_data` - part of the bitstream that may be used for ancillary data.

00	MPEG Version 2.5 (unofficial)
01	*reserved* (=something is wrong)
10	MPEG Version 2 (ISO/IEC 13818-3)
11	MPEG Version 1 (ISO/IEC 11172-3)

Table 6.1: The `ID` bit assignment

00	reserved
01	Layer III
10	Layer II
11	Layer I

Table 6.2: The `Layer` Bit Assignment

6.1.1 Header

The first 32 bits (four bytes) are header information which is common to all layers.

- `syncword` - the bit string '111 1111 1111'.
- `ID` - two bits to indicate the ID (Table 6.1) of the algorithm.

  MPEG Version 2.5 is unofficial standard. It is an extension of the standard used for very low bit rate files.
- `Layer` - 2 bits to indicate which layer is used (Table 6.2).

  To change the layer, a reset of the decoder is required.
- `protection_bit` - one bit to indicate whether redundancy has been added in the audio bitstream to facilitate error detection and concealment. Equals 1 if no redundancy has been added or not protected, 0 if redundancy has been added or protected by CRC (16 bit crc follows header).
- `bit_rate_index` - indicates the bit rate (Table 6.3). The all zero value indicates the *free format* condition, in which a fixed bit rate which does not need to be in the list can be used. The all one value indicates the *bad* condition, which means that this is not an allowed value (=something is wrong). Fixed means that a frame contains either N or $N + 1$ slots, depending on the value of the padding bit. The `bit_rate_index` is an index to a table, which is different for the different Layers. The `bit_rate_index`

`bit_rate_index`	Layer I	Layer II	Layer III
0000	*free format*	*free format*	*free format*
0001	32 kbps	32 kbps	32 kbps
0010	64 kbps	48 kbps	40 kbps
0011	96 kbps	56 kbps	48 kbps
0100	128 kbps	64 kbps	56 kbps
0101	160 kbps	80 kbps	64 kbps
0110	192 kbps	96 kbps	80 kbps
0111	224 kbps	112 kbps	96 kbps
1000	256 kbps	128 kbps	112 kbps
1001	288 kbps	160 kbps	128 kbps
1010	320 kbps	192 kbps	160 kbps
1011	352 kbps	224 kbps	192 kbps
1100	384 kbps	256 kbps	224 kbps
1101	416 kbps	320 kbps	256 kbps
1110	448 kbps	384 kbps	320 kbps
1111	*bad*	*bad*	*bad*

Table 6.3: The `bit_rate_index` Bit Assignment

00	44.1 kHz
01	48 kHz
10	32 kHz
11	*reserved*

Table 6.4: The `sampling_frequency` Bit Assignment

indicates the total bit rate irrespective of the mode (`stereo, joint_stereo, dual_channel, single_channel`).

In order to provide the smallest possible delay and complexity, the decoder supports variable bit rate by switching the `bit_rate_index`. However, in *free format,* fixed bit rate is required.

- `sampling_frequency` - indicates the sampling frequency (Table 6.4).

  A reset of the decoder is required to change the sampling rate.

- `padding_bit` - if this bit equals 1 the frame contains an additional slot to adjust the mean bit rate to the sampling frequency, otherwise this bit will

00	stereo
01	joint_stereo (intensity_stereo and/or ms_stereo)
10	dual_channel
11	single_channel

Table 6.5: The mode Bit Assignment

00	subbands 4-31 in intensity_stereo, bound=4
01	subbands 8-31 in intensity_stereo, bound=8
10	subbands 12-31 in intensity_stereo, bound=12
11	subbands 16-31 in intensity_stereo, bound=16

Table 6.6: The Layer I and II mode_extension Bit Assignment

be 0. Padding is only necessary with a sampling frequency of 44.1 kHz. Padding is used to fit the bit rates exactly.

- private_bit - bit for private use. This bit will not be used in the future by ISO.
- mode - Indicates the mode according to Table 6.5. In Layer I and II the joint_stereo mode is intensity_stereo, in Layer III it is intensity_stereo and/or ms_stereo.
- mode_extension - these bits (Table 6.6) are used in joint_stereo mode. In Layer I and II they indicate which subbands are in intensity_stereo. All other subbands are coded in stereo.
  In Layer III (Table 6.7), they indicate which type of joint stereo coding method is applied. The frequency ranges over which the intensity_stereo and ms_stereo modes are applied are implicit in the algorithm.

	intensity_stereo	ms_stereo
00	off	off
01	on	off
10	off	on
11	on	on

Table 6.7: The Layer III mode_extension Bit Assignment

00	*no emphasis*
01	50/15 $\mu\,s$ emphasis
10	*reserved*
11	CCITT J.17

Table 6.8: The `emphasis` Bit Assignment

- `copyright` - if this bit equals 0 there is no copyright on the coded bitstream, 1 means copyright protected.
- `original/home` - this bit equals 0 if the bitstream is a copy, 1 if it is an original.
- `emphasis` - indicates the type of de-emphasis (Table 6.8) that shall be used.

6.1.2 Audio data

`audio_data` - the quantized audio data from the filterbank and psychoacoustics.

6.1.3 Error check

`crc_check` - a 16 bit parity-check word is used for optional error detection within the encoded bitstream.

6.1.4 Ancillary data

The most common and perhaps the most popular ancillary data used in MP3 coding is the ID3 tag. We will discuss this in the following section.

6.2 ID3 Tags

The ID3 tag [4] is used to describe the MPEG Audio file. It contains information about artist, title, album, publishing year and genre. There are two major versions of ID3 tags: ID3v1 and ID3v2. The major difference is that the latter can hold more

Length (byte)	Description
3	Tag identification. Must contain "TAG" if tag exists and is correct.
30	Title
30	Artist
30	Album
4	Year
30	Comment (if ID3v1.1 then it's only 28 followed by a "\0")
1	In most cases this is a part of the 30 character Comment string, but in some files(ID3v1.1), this number represents the track number.
1	Genre

Table 6.9: ID3v1 Tag Format

side information and of different variety than the former. ID3v2 can even contain a thumbnail image and the song lyrics or voice transcription synchronized with its timestamp. ID3v1 is exactly 128 bytes long and is always located at the end of the audio data although placing it at the beginning is allowed. This often gives problems to the decoder. ID3v2 is located between the header and the audio data and can vary in size. The format of ID3v1 is found in Table 6.9.

Chapter 7

Conclusion

The MP3 algorithm has been extended [5] to lower sampling frequencies, namely 16 kHz, 22.05 kHz and 24 kHz. Separate groups have devoted research in optimizing algorithms for each of the four primary parts. A high performance filterbank implementation was discussed [7]. Research in psychoacoustics are continuing in hope to fully understand the behavior of the human ear. MDCT is commonly used in various audio compression algorithms [9]. Many MPEG algorithms are still undergoing research. One of them is MPEG-4, the coding algorithm that will be used in future mobile communications, encompasses low bit rate speech coding (2 kbps and below) and high-quality audio coding (64 kbps and above) per channel.

MP3 players are becoming common. Even portable versions [2] are already available in the market. The downside of MP3 is the legal issues that haunt it. Aside from the complicated patent licensing scheme of the algorithm, recording companies are complaining of rampant violations of copyright laws. This is beyond the scope of this research. One certain thing is that MDCT will be central to future compression algorithms, whether standalone or combination of audio and still or moving images.

List of References

[1] K. Brandenburg and H. Popp, *An introduction to mpeg layer-3*, tech. report, Fraunhofer Institut fr Integrierte Schaltungen (IIS), Erlangen, Germany.

[2] D. M. Carey, *Mp3, now portable*, tech. report.

[3] Fraunhofer Institute for Integrated Circuits FhG-IIS A, *MP3 And AAC Explained*, AES 17th International Conference on High Quality Audio Coding, Erlangen, Germany.

[4] B. Grill, B. Edler, R. Funken, M. Hahn, K. Iijima, N. Iwakami, Y. Lee, T. Moryia, J. Ostermann, S. Nakajima, M. Nishiguchi, T. Nomura, W. Oomen, H. Purnhagen, E. Scheirer, N. Tanaka, A. Tan, and R. Taori, *Coding of moving pictures and associated audio for digital storage media at up to about 1.5 mbps part 3 audio*, tech. report, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, November 22, 1991. ISO/IEC 11172-3 International Standard.

[5] ——, *Information technology - generic coding of moving pictures and associated audio: Audio*, Tech. Report NO803, ISO/IEC JTC1/SC29/WG11, Geneva, Switzerland, November 11, 1994. ISO/IEC 13818-7 International Standard.

[6] S. Hacker, *MP3: The Definitive Guide*, O'Reilly and Associates, Inc, 2000.

[7] M. Kumar and M. Zubair, *A high performance software implementation of mpeg audio encoder*, tech. report, IBM T.J. Watson Research Center, NY, USA.

[8] T. Painter and A. Spanias, *Perceptual coding of digital audio*, Tech. Report

85287-7206, Department of Electrical Engineering, Telecommunications Research Center, Arizona State University, Tempe, Arizona.

[9] ——, *A review of algorithms for perceptual coding of digital audio signals*, Tech. Report 85287-7206, Department of Electrical Engineering, Telecommunications Research Center, Arizona State University, Tempe, Arizona.