



MP3 DECODER in Theory and Practice

Praveen Sripada

Masters Thesis Report

**Blekinge Tekniska Högskola
March 2006**

Supervisors: Josef Ström Bartunek
Jörgen Nordberg

Department of Signal Processing and Telecommunications
Blekinge Institute of Technology
Box 520, SE – 372 25
Ronneby
Sweden

Abstract

MPEG audio coding under the name MP3 has become one of the most popular standards for digital audio broadcasting and videos. High compression ratios offered by MP3 codecs in various stand alone players and hand held devices over the last few years has increased its popularity immensely. Internet users, music lovers who would like to download highly compressed digital audio files at near CD quality are the most benefited. Psychoacoustic model, Modified Discrete Cosine Transform (MDCT) and Huffman coding play a vital role in achieving such magnificent compression ratios. In this thesis, a thorough knowledge of MP3 decoder is obtained by going through the ISO standard and then some of the decoder blocks have been implemented for deeper understanding.

CONTENTS

1 Introduction	- 9 -
1.1 How does MP3 work?	- 9 -
1.2 MPEG Audio Compression.....	- 9 -
2 Overview of audio compression formats	- 11 -
3 Inside an MP3 file.....	- 13 -
4 Overview of MP3 Encoder	15
4.1 Filter bank and Psychoacoustic model	15
4.2 Quantisation	16
4.3 Huffman coding.....	16
4.4 Bitstream formatting	16
5 MP3 Decoder	17
5.1 Audio Frame Header	19
5.1.1 Frame Header in detail.....	19
5.1.2 Frame Length Calculation.....	23
5.2 Decoding Side information	24
5.3 Main data.....	30
5.4 Decoding Scalefactors.....	31
5.5 Decoding Huffman data	31
5.6 Requantizing spectrum	32
5.7 Reordering spectrum	33
5.8 Stereo processing.....	33
5.8.1 Mid/Side stereo	33
5.8.2 Intensity stereo.....	34
5.9 Alias reduction	34
5.10 Inverse Modified Discrete Cosine Transform and Overlapping	35
5.11 Frequency inversion	35
5.12 Synthesis via polyphase filter bank.....	36

6 Implementation.....	37
6.1 Header information for frame 1	37
6.2 Side information details for frame 2	37
6.3 Problems encountered during implementation.....	38
7 Conclusions	41
8 References	- 43 -

Abbreviations

AC 3 – Advanced Codec 3

CCITT – Consultative Committee for International Telephone and Telegraph

CD – Compact Disc

CRC – Cyclic Redundancy Code

DVD – Digital Versatile Disc

FFT – Fast Fourier Transform

GSM – Global System for Mobile communications

IEC – International Electrotechnical Commission

IMDCT – Inverse Modified Discrete Cosine Transform

ISO – International Organization for Standardization

ITU – International Telecommunications Union

kHz – KiloHertz

kbps – Kilo Bits Per Second

MDCT – Modified Discrete Cosine Transform

MPEG – Motion Picture Experts Group

MP3 – MPEG 1 Layer III

MS – Mid Side stereo

PCM – Pulse Code Modulation

SMR – Signal to Masking Ratio

WMA – Windows Media Audio

1 Introduction

MP3 has changed the way people listen to music on the Internet. It was not so long ago that the average pop song converted into a Wav file took hours to download on a 28.8 kbps modem connection and ate up around 50 megabytes of disc space. With the same song converted into an MP3 file, download time gets reduced dramatically to around one-tenth the original size while sounding just as good as before.

1.1 How does MP3 work?

As a form of compression, MP3 is based on a psycho-acoustic model [1], which recognizes that the human ear cannot hear all the audio frequencies in a recording. The human hearing range is between 20 Hz to 20 kHz and it is most sensitive between 2 to 4 kHz. When sound is compressed into an MP3 format, an attempt is made to get rid of the frequencies that cannot be heard. As such, this is known as 'destructive' compression. After compression, the information that is eliminated from the audio signal cannot be replaced [8].

When encoding data into MP3 format, a variety of compression levels can be set. For instance, an MP3 file created with 128 kbit compression will be of a greater quality and larger file size than that of a 56 kbit compression. The greater the compression ratio, the lesser is the sound quality.

Layers in MP3: The complexity of MP3 codec increases while moving from Layer 1 to Layer3.

Layer 1 possesses the lowest complexity and is specifically targeted to applications where the complexity of the encoder plays an important role.

Layer 2 requires a more complex encoder as well as a slightly more complex decoder. Compared to Layer 1, Layer 2 is able to suppress more redundancy in the signal and applies the psychoacoustic model in more efficient way.

Layer 3 is once again of an increased complexity and is targeted to applications needing the lowest data rates, by its suppression of the redundant signal and its improved extraction of feebly audible frequencies using its filter [11].

1.2 MPEG Audio Compression

MPEG is a lossy compression, which means, some audio information is certainly lost using these compression methods. This loss can hardly be noticed because the compression method tries to control it. By using several complicated and demanding mathematical algorithms it will only lose those components of sound that are hard to be heard even in the original form [10].

This leaves more space for information that is important. This way it is possible to compress audio up to 12 times which is really significant. Due to its quality MPEG audio became very popular [10].

MPEG-1 audio (described in ISO/IEC 11172-3) [1] describes three layers of audio coding with the following properties:

- One or two audio channels.
- Sample rate 32 kHz, 44.1 kHz or 48 kHz.
- Bit rates from 32 kbps to 448 kbps.

2 Overview of audio compression formats

MPEG: MPEG (from the Motion Picture Experts Group) is the international standard for multimedia. It incorporates both audio and video encoding at a range of data rates. MPEG audio and video are the standard formats used in Video CDs and DVDs. The lowest data rate supported for MPEG-1 mono audio is 32 kbps. Sample rates of 32 kHz, 44 kHz (audio CD) and 48 kHz (Digital Audio Tape) are supported.

Dolby Digital: Dolby digital created by Dolby labs is used as audio format for movie theatres and DVD's. Dolby Digital is based on "AC-3", a perceptual coding scheme, and can support a wide range of audio stream from a single mono channel to "5.1" surround sound. 5.1 surround sound includes left, center, and right front channels; left and right rear surround channels; and a bass boost channel. Dolby Digital tends to be a specification for high-end applications, while MP3 is better suited to low-end applications.

Mu-Law: mu-law is the international standard telephony encoding format, also known as ITU (formerly CCITT) standard G.711. It packs each 16-bit sample into 8 bits by using a logarithmic table to encode with a 13-bit dynamic range and dropping the least significant 3 bits of precision. Encoding and decoding is very fast and support is universal.

Real Audio: RealAudio is a proprietary encoding format created by Progressive Networks. It was the first compression format to support live audio over the Internet and thus gained considerable support. Later versions could support higher sampling rates ranging from 11 kHz to 44 kHz.

GSM 06.10: GSM 06.10 is the international standard digital mobile telephony encoding format. It uses linear predictive coding to substantially compress the data by predicting the likely shape of the sound wave and recording the differences between the actual sound and the prediction. Compression and decompression are slow and the quality is not great, but the algorithm is freely available resulting in widespread use in various products.

WMA: Windows Media Audio, developed by Microsoft, has been pushing their own audio compression format as an alternative to MP3. Even though it is said that it has good compression quality, but the predominance of Microsoft ensured that WMA is a popular format and is supported by almost all digital music players.

OggSquish: One of the most interesting compression algorithms is being implemented by the "Xiph.org Foundation". It is a group named "Ogg" to create public-domain, non-proprietary, free and open compression and multimedia specifications. They have created an audio compression scheme

known as 'Vorbis'. It will provide a range of compression factors from 5:1 up to 18:1 plus a "lossless" compression mode. It is optimized for very high sound quality (source material at 30-48 kHz sample rates).

3 Inside an MP3 file

The bit stream inside an MP3 file (see Figure 1) contains frames with the following parts [1].

- Header
- Side information
- Main data
- Ancillary data

Header	Side information	Main data	Ancillary data
--------	------------------	-----------	----------------

Figure 1: Organisation of bit stream in MPEG 1 Layer III

Header is always 32 bits or 4 bytes and the information in the header confirms the authenticity of an MP3 file. Location of each header does not always need to be at the beginning of the frame. Therefore each header starts with a syncword to mark its position in an MP3 file.

Side information can either be 17 bytes if it is a single channel or 32 bytes if it is a dual channel. Side information always immediately follows the header. Basically, it contains all the relevant information to decode the main data. For example it contains the main data begin pointer, scale factor selection information, Huffman table information for both the granules etc.

Main data need not always follows the side information. It can be divided in such a way that a part of the main data can be located in the current frame and the other part can be found in the previous frame. Details on how the main data is organised can be known only after extracting the side information. It is always advised to look at least three frames at a time to get some clear understanding of the bitstream.

Ancillary data can be defined by the user and the exact number of bits is not explicitly mentioned. It starts after the Huffman coded bits. The distance between the end of the Huffman coded bits and the location in the bitstream, where the next frame's main data begin pointer points to, is the number of ancillary bits [1].

4 Overview of MP3 Encoder

Block diagram (see Figure 2) of an MP3 Encoder along with a brief description of it is given below.

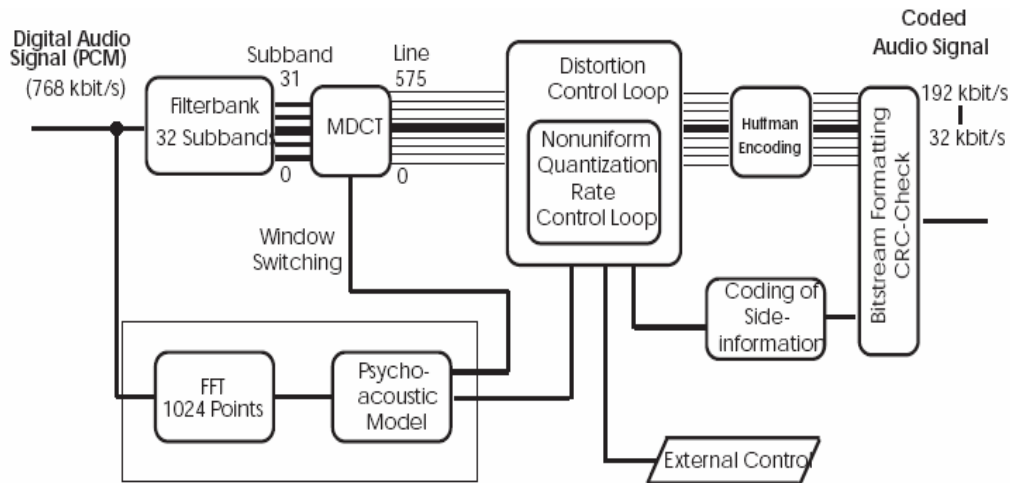


Figure 2: Block diagram of MPEG-1 Layer 3 Encoder (Source [4])

4.1 Filter bank and Psychoacoustic model

There are two filter banks in a MPEG audio algorithm, namely filterbank and a hybrid polyphase / MDCT filterbank (see Figure 2). The input PCM samples are simultaneously fed into a filterbank and a psychoacoustic model. Filter bank splits the signal into 32 equal subbands in frequency domain where as psychoacoustic model takes the signal spectrum as input and determines the ratio of signal energy to masking threshold for each subband. To obtain better frequency resolution the 32 subbands are further divided into 576 frequency lines by the MDCT. MDCT used is 12 point (short) or 36 point (long) with 50 % overlap and the type of MDCT (long or short point) is determined by the window switching algorithm [6]. The output of the psychoacoustic model consists of masking threshold values or allowed noise for each coder partition. In Layer-3, these coder partitions are roughly equivalent to the critical bands of human hearing. If the quantization noise can be kept below the masking threshold for each coder partition, then the compression result should be indistinguishable from the original signal.

4.2 Quantisation

The signal energy to masking ratio (SMR) which is calculated by the psychoacoustic model is used by the quantizer to determine the number of code bits that should be allocated for the quantization of subband coefficients. Quantization is done via a power-law quantizer [3].

4.3 Huffman coding

The quantized values are coded by Huffman coding. To get even better adaptation to signal statistics, different Huffman code tables can be selected for different parts of the spectrum. Huffman coding is basically a variable code length method and noise shaping has to be done to keep the quantization noise below the masking threshold. So a global gain value (determining the quantization step size) and scalefactors (determining noise shaping factors for each scalefactor band) are applied before actual quantization [3]. The process to find the optimum gain and scalefactors for a given block, bit-rate and output from the perceptual model is usually done by two nested iteration loops namely the rate control loop and the noise or distortion control loop.

The order of the Huffman encoded data depends on the block type of the granule. If the block type is 0, 1, or 3 the Huffman encoded data is ordered in terms of increasing frequency. If the block type is 2, then the Huffman encoded data is ordered in the same order as the scalefactor values for that granule.

4.4 Bitstream formatting

Finally the Huffman coded values are formed into a bitstream. A bitstream formatter is used to assemble the bitstream. The encoded bitstream consists of quantized and coded spectral coefficients along with some side information like bit allocation information, quantiser step size information etc.

5 MP3 Decoder

Decoding of MP3 audio is carefully defined in the ISO standard [1]. Each frame is made up of 1152 samples and there is always a header attached to each of the frames associated in the MP3 file. Content in the header and side information for a particular frame is necessary so that decoding is done correctly.

The first and foremost thing in the decoding procedure is the synchronisation of the decoder to the incoming bitstream. Synchronisation is the process of finding the position of the first header and the subsequent ones. Once this is done, the organisation of the encoded data is completely known and the decoding procedure can be performed smoothly. The block diagram in Figure 3 and flow chart in Figure 4 gives an idea on the procedure.

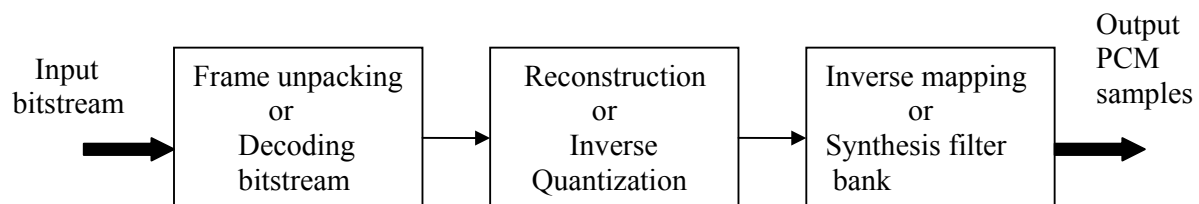


Figure 3: Basic sketch of a decoder

Frame unpacking constitutes finding the bitstream header, decoding side information, decoding scale factors and decoding the Huffman data. Reconstruction block constitutes requantizing and reordering the spectrum. Inverse mapping constitutes joint stereo processing if applicable, alias reduction, synthesis via IMDCT and polyphase filter bank, and out comes the PCM samples.

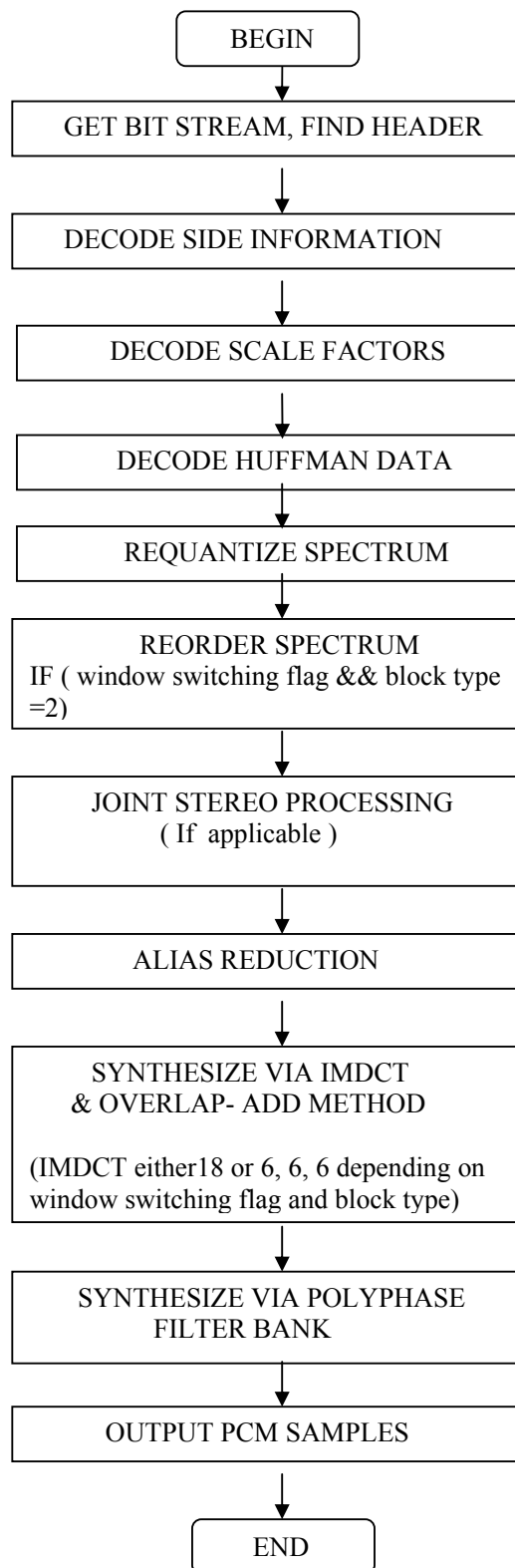


Figure 4: Block diagram of MPEG 1 Layer III decoder

5.1 Audio Frame Header

An MPEG audio file is built up from smaller parts called frames. Generally, frames are independent items i.e. any part of the file can be cut and played correctly. For Layer III, this is not totally true. Due to internal data organization in MPEG version 1 Layer III files, frames are often dependent on each other [12].

When information about an MPEG file has to be read, it is usually enough to find the first frame, read its information and assume that all the frames would be following the same pattern [12].

But MP3 supports variable bitrate, which means bitrate changes according to the content of each frame. This way lower bitrates may be used in frames where it will not reduce sound quality. Thus it allows better compression rates while keeping high quality of sound.

5.1.1 Frame Header in detail

The frame header consists of four bytes or 32 bits and the proper way to read the header is given in Table 1. The first 12 bits of the frame header are always set to one and are called '*frame sync*'. The remaining bits contain information about the MPEG version, bit rate, sampling rate etc. Remember, this is not enough, frame sync can be easily (and very frequently) found in any binary file. Also it is likely that MP3 file contains lot of additional information on its beginning which also may contain false sync. Thus, two or more frames in a row have to be checked to assure that it really is an MP3 file.

Frames may have a CRC check. The CRC is 16 bits long and, if it exists, it follows the frame header. After the CRC comes the audio data [12]. The length of the frame may be calculated to read other headers. It can also be used to calculate the CRC of the frame and can be compared with the audio file under consideration. This is actually a very good method to check the MPEG header validity.

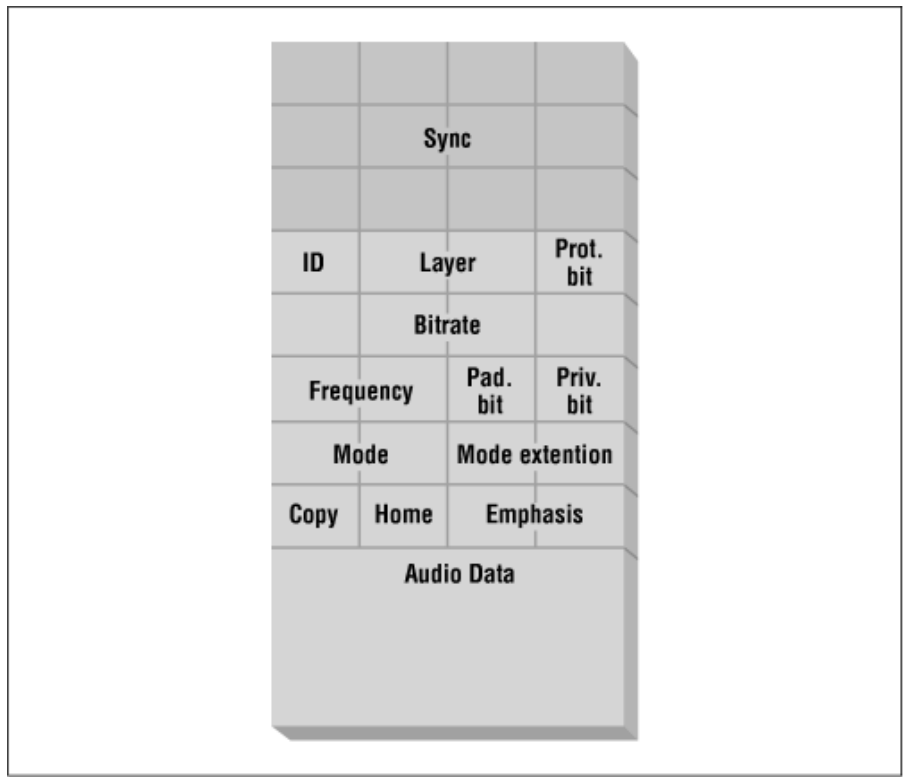


Figure 5: Terms contained in the header (Source [9], [13])

From Figure 5, those details represented above the audio data constitute the header part of an MP3 file. When Figure 5 is observed closely, the header part is divided into 32 small blocks which is analogous to the 32 bits in the header. So each block represents a bit and the description of the terms is given in Table 1.

Position of bits in the header	Number of bits	Definition	Example
1 to 12	12	Sync word : Frame sync which should always be set	'1111 1111 1111 1111'
13	1	ID: Denotes the MPEG version 1 or 2	'1'
14 to 15	2	Layer: Denotes which layer is used.	'01' - Layer III '10' - Layer II '11' - Layer I '00' - Reserved
16	1	Protection bit: Indicates if the bitstream is protected by CRC following the header	'0' - Protected '1' - Unprotected
17 to 20	4	Bitrate: Different bitrates can be used while encoding	'1001' – 128 kbps
21 to 22	2	Frequency : MP3 supports 32, 44.1, 48 kHz frequencies	'00' - 44.1 kHz
23	1	Padding bit: If set, then the data is padded with one extra slot.	'0' - Not set '1' - Set
24	1	Private bit: Only informative	'1'
25 to 26	2	Mode: Denotes either single, dual, joint stereo or stereo channels	'11' - Single Channel
27 to 28	2	Mode extension: Used only in joint stereo mode	'00' - M/S and Intensity stereo are off
29	1	Copyright: Indicates if the bitstream is copyrighted or not	'0' - No copyright '1' - Copyrighted
30	1	Copy: Indicates if bitstream is a copy or original	'0' - Copy '1' - Original
31 to 32	2	Emphasis: Indicates the type of emphasis used	'00' - No emphasis

Table 1: Organisation of bits in the header of an MP3 file

Bitrate: Bitrates ranging from 32 to 328 kbps are supported in MP3. Layer III supports variable bit rate by switching the bitrate index between the frames. Table 2 gives the bit rates for MPEG versions 1, 2 and 2.5.

Bitrate Index	MPEG 1			MPEG 2, 2.5 (LSF)	
	Layer I	Layer II	Layer III	Layer I	Layer II & III
0000	Free	Free	Free	Free	Free
0001	32	32	32	32	8
0010	64	48	40	48	16
0011	96	56	48	56	24
0100	128	64	56	64	32
0101	160	80	64	80	40
0110	192	96	80	96	48
0111	224	112	96	112	56
1000	256	128	112	128	64
1001	288	160	128	144	80
1010	320	192	160	160	96
1011	352	224	192	176	112
1100	384	256	224	192	128
1101	416	320	256	224	144
1110	448	384	320	256	160
1111	Reserved	Reserved	Reserved	Reserved	Reserved

Table 2: Bitrates (Source [12])

Sampling frequency: Table 3 indicates the sampling frequency according to the ISO specifications. From Table 3, it can be seen that the sampling frequencies are getting halved in the next versions of MPEG which means that it can support wider range of applications.

Sampling Rate Index	MPEG 1	MPEG 2 (LSF)	MPEG 2.5 (LSF)
00	44100 Hz	22050 Hz	11025 Hz
01	48000 Hz	24000 Hz	12000 Hz
10	32000 Hz	16000 Hz	8000 Hz
11	Reserved	Reserved	Reserved

Table 3: Sampling frequency for MPEG versions 1, 2 and 2.5

Mode: Four types of modes are supported by MP3. Mode indicates the various types of modes used according to Table 4.

Bit value	Type of mode
00	Stereo
11	Joint Stereo
10	Dual Channel
01	Single Channel

Table 4: Bit values and mode type

Mode extension: If joint stereo coding is applied in the bitstream then it is important to know if the intensity stereo and mid side (MS) stereo are on or off. It can be known from Table 5.

Bit value	Intensity stereo	MS stereo
00	Off	On
01	On	Off
10	Off	On
11	On	On

Table 5: Bit values and mode extension

5.1.2 Frame Length Calculation

There are two terms that most people are confused with an MP3 file. They are the frame size and the frame length. Frame size is the number of samples contained in a frame. It is constant and is always 384 samples for Layer I and 1152 samples for Layer II and Layer III.

Frame length is the length of a frame when compressed. It is calculated in slots. One slot is 4 bytes long for Layer I, and one byte long for Layer II and Layer III. When reading a MPEG audio file, length of each frame has to be calculated to be able to find each consecutive frame. Remember, frame length may change from frame to frame due to padding or bitrate switching.

For Layer II & III files, frame length is calculated by formula 1.

$$FLB = 144 * \frac{Bitrate}{Samplerate} + Padding \quad (1)$$

where FLB is the frame length in bytes.

If the padding bit is set, then the frame contains an additional slot to adjust the mean bitrate to the sampling frequency. Padding is necessary when the sampling frequency is 44.1 kHz and the frame length should always be an integer.

Example:

$$Framelength = 144 * \frac{128}{44.1} + 1 = 418 \text{ bytes}$$

5.2 Decoding Side information

As stated earlier, the side information is 17 bytes in length for a single channel encoded file and 32 bytes for dual channel mode. Information in side information allows decoding the main data correctly. Basic structure of side information is given in Figure 6.

main_data_begin	Private_bits	scfsi	side_info for gr.1	side_info for gr.2
-----------------	--------------	-------	-----------------------	-----------------------

Figure 6: View of side information

Table 6 gives a clear picture of how the side information is organised both for single and dual channel modes. Organisation of side information for block type 2 is presented in Table 7. Lengths of each term mentioned in the tables are indicated in bits and terms involved in the tables 6 and 7 are explained following the tables.

Name	Single channel	Dual channel
<i>main_data_begin</i>	9	9
<i>private_bits</i>	5	3
<i>share</i>	4	4 + 4
Information for first granule:		
<i>part2_3_length</i>	12	12 + 12
<i>big_values</i>	9	9 + 9
<i>global_gain</i>	8	8 + 8
<i>scalefac_compress</i>	4	4 + 4
<i>window_switching</i>	1	1 + 1
For normal blocks:		
<i>table_select</i>	3*5	3*5 + 3*5
<i>region0_count</i>	4	4 + 4
<i>region1_count</i>	3	3 + 3
Subtotal for normal blocks	22	44
<i>preflag</i>	1	2
<i>scalefac_scale</i>	1	2
<i>count1table_select</i>	1	2
Subtotal for first granule	59	118
Subtotal for second granule	59	118
Total number of bits	136	256
Total number of bytes	17	32

Table 6: Organisation of side information for block types 0, 1 and 3(Source [2])

Name	Single channel	Dual channel
For start, stop and short blocks: <i>block_type</i>	2	2 + 2
<i>mixed_block_flag</i>	1	1 + 1
<i>Table selection for two regions</i>	2*5	2*5 + 2*5
<i>subblock gain</i>	2*5	3*3 + 3*3
Subtotal for not normal blocks	22	44

Table 7: Organisation of side information for block type 2

main_data_begin

It is a pointer that points to the beginning of the main data. The variable has nine bits and specifies the location of the main data as a negative offset (jumping backwards) in bytes from the first byte of the audio sync word. The number of bytes of the header and side information are not taken into account while calculating the location of the main data. This is called bit reservoir technique and it allows the encoder to use some extra bits while encoding a difficult frame. Since it is nine bits long, it can point upto $2^9 - 1 = 511$ bytes in front of the header. If the value of *main_data_begin* is zero, then the main data follows immediately the side information.

private_bits

These bits are for private use and will not be used by ISO in the future. But one may wonder as to why to waste three to five precious bits, while fighting for every single bit in other places. The reason is, these private bits round up the size of side information to a sequence of full bytes and equalize it, making a fixed size (which being 17 for single and 32 for dual channels) as required [2].

scfsi

Layer III contains two granules and the encoder can specify separately for each group of scale factor bands whether the second granule will reuse the scale factor information of the first granule or not. If the value of *scfsi* is one, then sharing of scale factors is allowed between the granules.

scfsi_band

Layer III there has one scale factor for each frequency band and the 21 frequency bands are separated into 4 groups according to Table 8. If block type is 2 then scale factors are transmitted for each granule and channel.

Group	Scalefactor band
0	0 - 5
1	6 - 10
2	11 - 15
3	16 - 20

Table 8: Scalefactor bands

part2_3_length

This value contains the number of main_data bits used for scale factors and Huffman coded data. The main data is divided into two or four parts, for each granule and channel, depending on single or dual channel respectively. The size of each of these sections is the first item in the side information which is 12 bit unsigned integer.

big_values

The total frequency spectrum from zero to Nyquist frequency is divided into several regions depending on the maximum quantized values. It is broadly classified into three regions namely big values, count1 and rzero, which is shown in Figure 7. Once they are classified they are coded with different Huffman code tables.

Rzero: It is assumed that higher frequency values are expected to possess lower amplitudes and need not be coded. So starting from higher frequencies pairs of quantized values equal to zero are counted and are termed as 'rzero' [1].

Count1: These are quadruples of quantized values which has only three quantized values containing -1, 0 and 1.

Big values: The first part of the frequency spectrum contains the big values. Big values are the number of pairs of quantized values, in the region of the spectrum which extend down to zero. The maximum absolute value in this range is 8191.

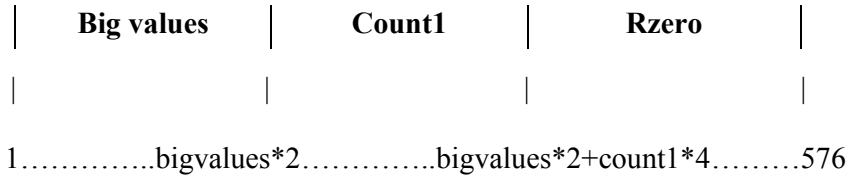


Figure 7: Frequency spectrum division

global_gain

The quantizer step size information is known through this variable and the formula for requantization is given in the requantization block.

scalefac_compress

Determines the number of bits used for the transmission of the scalefactors. The number of bits that has to be transferred to scale factor bands is defined by two variables called 'slen1' and 'slen2'. Depending on the block type, the transmission of slen1 and slen2 for the scalefactor bands vary and is presented in Table 9.

scalefac_compress	slen1	slen2
0	0	0
1	0	1
2	0	2
3	0	3
4	3	0
5	1	1
6	1	2
7	1	3
8	2	1
9	2	2
10	2	3
11	3	1
12	3	2
13	3	3
14	4	2
15	4	3

Table 9: scalefac_compress

If the block type is 0,1or 3 then - slen1 is transferred for the scalefactor bands 0 to 10 and and slen2 for the bands 11 to 20.

If the block type is 2 and mixed block flag is 0 then - slen1 is transferred for the scalefactor bands 0 to 5 and and slen2 for the bands 6 to 11.

If the block type is 2 and mixed block flag is 1 then - slen1 is transferred for the scalefactor bands 0 to 7 (long window scale factor band) and 3 to 5 (for short window scale factor band). Slen2 is transferred for the bands 6 to11.

window_switching_flag

Indicates that other than normal window is used. If window_switching_flag is set then variables block_type, mixed_block_flag, subblock_gain are also set. If window_switching_flag is not set then the value of block_type is zero.

block_type

Indicates which type of window to be used for each granule. The different types of windows along with block type are provided in Table 10.

block_type	window type
0	reserved
1	start block
2	3 short windows
3	end

Table 10: Block type and window type

mixed_block_flag

Indicates that different frequencies are transformed with different window types. If mixed_block_flag is not set then all the frequency lines are transformed as specified by block_type. If it is not set, then the two lowest polyphase subbands are transformed with normal window and the remaining 30 subbands as block_type.

table_select

As the name states, different Huffman coded tables are selected depending on the maximum quantized value and local statistics of the signal. There are 32 different Huffman tables given in the ISO standard. The table_select specifies the Huffman table to decode only the big_values.

subblock_gain

This variable is used only when window_switching_flag is set and for short windows (i.e, block_type=2). It indicates the gain offset from the global

gain for one subblock and the values of the subblock have to be divided by $4^{(subblock_gain [window])}$.

region0_count and region1_count

Big_values that were mentioned earlier is further subdivided into three regions namely, region0, region1 and region2. This partition of the spectrum is used to enhance the performance of Huffman coder while also attaining better error robustness and better coding efficiency. The values region0_count and region1_count are used to indicate the boundaries of the regions. The region boundaries are aligned with the partitioning of the spectrum into scale factor bands. Region0_count and region1_count contains one less than the number of scalefactor bands in the regions 0 and 1 respectively [1].

preflag

This field is never used for short blocks (i.e, block_type = 2). If it is set, then the values of the Table 11 are added to the scale factors. This is equivalent to multiplication of the requantized scalefactors with the Table 11 values which also means additional high frequency amplification of the quantized values.

scalefac_scale	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
pretab	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	2	2	3	3	3	2

Table 11: Preflag table only for block_type 2 windows

scalefac_scale

The scalefactors are logarithmically quantized with a step size of $\sqrt{2}$ or 2 depending on the value of scalefac_scale. In the requantization equation of each step size the scalefac_scale is multiplied by a factor 0.5 if the value of scalefac_scale is 0, else multiplied by 1 if the value of scalefac_scale is 1.

count1table_select

This variable selects which of the two possible Huffman tables will be used for quadruples of quantized values with magnitude not exceeding 1.

5.3 Main data

The main data in the bitstream is split into two granules. Each granule contains scalefactors, Huffman coded data and ancillary information which

has to be read. The start of the main data, whether it immediately follows the side information or its location as negative offset, is known from the main_data_begin pointer. The decoder has to skip the header (4 bytes) and

side information (17 or 32 bytes) while decoding the main data. Main data is allocated in such a way that all main data is resident in the input buffer when the header of the next frame is arriving in the input buffer. Organisation of main data in granules is shown in Figure 8.

Scale factors	Huffman coded data	Ancillary information
---------------	--------------------	-----------------------

Figure 8: Organisation of main data in granules

5.4 Decoding Scalefactors

For each granule the bitstream contains first the scalefactors and then the Huffman coded raw samples. Sharing of scalefactors has to be checked before reading the scalefactors. They are definitely not shared in the first granule of a frame. If sharing of scalefactors is allowed, then the scalefactors of the first granule are used for the second granule as well and they will not be transmitted for the second granule. Further, a short block in either the first or the second granule prevents sharing.

From the bitstream only the scale factor indices are found but not the scalefactors. The index along with the maximum scalefactor index is stored into two arrays. Most of it is reused for the second granule. Reading of scale factor indices are done according to *slen1* and *slen2*, which themselves are decoded from the values of *scalefac_compress*.

The number of bits used to encode the scalefactors is called *part2_length* and is calculated by the formulas 2, 3 and 4 depending on the window types used in the granules.

If *block_type* = 0, 1, 3, then

$$\text{part2_length} = 11 * \text{slen1} + 10 * \text{slen2}. \quad (2)$$

If *block_type*= 2 and *mixed_block_flag*=0, then

$$\text{part2_length} = 18 * \text{slen1} + 18 * \text{slen2}. \quad (3)$$

If *block_type*= 2 and *mixed_block_flag*=1, then

$$\text{part2_length} = 17 * \text{slen1} + 18 * \text{slen2}. \quad (4)$$

5.5 Decoding Huffman data

First, the frequency lines of the three regions in the big values are decoded and then the small values. Decoding is done by using the tables specified in

the standard and details on which table to choose is given by the value `table_select` in the side information. Once the big values are decoded, the remaining Huffman coded bits are decoded by the value `count[table_select]`. Decoding is done until all Huffman code bits have been decoded or until quantized values representing 576 frequency lines have been decoded, whichever comes first. If there are more Huffman code bits than necessary to decode 576 values they are regarded as stuffing bits and discarded. When there are less than 576 frequency lines, Huffman code has to initiate a zero padding to compensate the lack of data.

5.6 Requantizing spectrum

Quantization is the process of converting a real number (of almost) infinite precision, taken from an infinite and continuous set of possible values, into an integer number [2]. This is done during the encoding process. In the decoding process the quantization process is reversed to obtain the frequency lines. The raw integer sample values for all 576 frequency lines that are obtained after Huffman decoding are first requantized and scaled. Requantization is done separately for both short and long blocks by using a power law and is given in formulas 5 and 6. Scaling which follows requantization, is done by multiplying the values by the corresponding scalefactors and are stored as scaled frequencies.

The Huffman decoded value at buffer index i is called $is[i]$ and the input to synthesis filter bank at index i is called $xr[i]$.

For short blocks,

$$xr[i] = \text{sign}(is[i]) * |is[i]|^{\frac{4}{3}} * 2^A * 2^B. \quad (5)$$

The terms A and B are defined according to 5.1 and 5.2 respectively.

$$A = \frac{1}{4} * (\text{global_gain}[gr] - 210 - 8 * \text{subblock_gain}[\text{window}][gr]) \quad (5.1)$$

$$B = -(\text{scalefac_multiplier} * \text{scalefac_s}[gr][ch][sfb][window]). \quad (5.2)$$

For long blocks,

$$xr[i] = \text{sign}(is[i]) * |is[i]|^{\frac{4}{3}} * 2^C * 2^D . \quad (6)$$

The terms C and D are defined according to 6.1 and 6.2 respectively.

$$C = \frac{1}{4} * (\text{global_gain}[gr] - 210) \quad (6.1)$$

$$D = -(\text{scalefac_multiplier} * (\text{scalefac_l}[sfb][ch][gr][window] + \text{preflag}[gr] * \text{pretab}[sfb])) \quad (6.2)$$

If the block type is 0,1,or 3 then the formula of long blocks is used.If the block type is 2 then the formula of short blocks is used. When the difference between the present time frame and the previous time frame is very less then long block/window is used. Alternatively if the subband signal shows considerable difference between the time frames, then short block/ window is used. Short windows consists of three short overlapped windows and will improve the time resolution given by the MDCT [5].

5.7 Reordering spectrum

Reordering of spectrum is dependent on the block type used prior to the IMDCT operation. If short window is used (block_type=2) then the requantization block would produce frequency lines ordered first by subband, then by window and at last by frequency. This ordering of frequency lines for short windows is done in such a way so as to increase the Huffman coding efficiency. If long windows are used, then the frequency lines are ordered first by subband and then by frequency [5].

5.8 Stereo processing

The reconstructed values after requantization, are now processed for MS or intensity stereo modes or both. Details on which mode to process is known by the mode extension value of the header specification.

The two channels of typical stereo signal are not independent and joint stereo tries to exploit the existing similarities. Joint stereo processing is complicated because short blocks are handled differently than long blocks. Also, granules can contain a mixture of long and short blocks and the bands in the granule can be combined with different stereo modes. There are two types of joint stereo namely Mid/Side (MS) stereo and Intensity stereo [2].

5.8.1 Mid/Side stereo

Mid/side stereo is only an option in Layer III, otherwise joint stereo is always intensity stereo. In mid/side stereo mode, instead of transmitting the left and right channel separately, the mid signal M(i) is derived by adding

the left and the right channel. The side signal $S(i)$ is derived by subtracting the right from the left channel. So in order to reconstruct the left and right channel values we reverse the process and are given by the formulas 7 and 8.

$$\text{Left channel } L(i) = \frac{1}{\sqrt{2}} * [M(i) + S(i)]. \quad (7)$$

$$\text{Right channel } R(i) = \frac{1}{\sqrt{2}} * [M(i) - S(i)]. \quad (8)$$

5.8.2 Intensity stereo

In intensity stereo mode, both channels share the same signal, only the intensity in both the channels differ [2]. Sounds coming from the side reach one ear of the listener faster than the other ear. Therefore the signal will be louder in the ear towards the sound source than in the other ear. Intensity stereo is more compact coding than normal stereo. It is done by specifying the magnitude via the scale factors of the right channel and a stereo position variable named `is_pos(sb)`. This variable is transmitted instead of scalefactors for the right channels.

5.9 Alias reduction

Aliasing reduction is done for long block types (ie, `block_type!=2`). The `antialias` block reduces the aliasing that is introduced by the use of ideal non-band pass filter. The frequency lines in the granule are arranged in the increasing order with 0 being the index of lowest frequency line and 576 being the highest. Aliasing reduction is done by merging the frequency lines using eight butterfly calculations for each subband [5]. The coefficients for the butterfly calculations are calculated using the values from the Table 12 and substituting them in the formulas 9 and 10.

$$cs(i) = \frac{1}{\sqrt{(1 + (c(i))^2)}}. \quad (9)$$

$$ca(i) = \frac{c(i)}{\sqrt{(1 + (c(i))^2)}}. \quad (10)$$

i	0	1	2	3	4	5	6	7
C(i)	-0.6	-0.535	-0.33	-0.185	-0.095	-	-	-
						0.041	0.00142	0.0037

Table 12: Coefficients for alias reduction

5.10 Inverse Modified Discrete Cosine Transform (IMDCT) and Overlapping

The IMDCT in co-operation with the synthesis filter bank produces, time samples $x(i)$ from frequency lines $X(k)$. 'n' is the number of windowed samples and for short blocks 'n' is 12 and for long blocks 'n' is 36. The IMDCT is calculated using the formula 11.

For $i = 0$ to $n-1$

$$x(i) = \sum_{k=0}^{(n/2)-1} X(k) \cos\left(\frac{\pi}{2n}\left(2i+1+\frac{n}{2}\right)(2k+1)\right). \quad (11)$$

For $n = 36$, the IMDCT takes 18 frequency lines as input and generates 36 polyphase filter sub-band samples. These samples are multiplied with a 36-point window before they can be passed on to the next step in the decoding process [7].

Windowing contains four different types of windows namely, normal, short, start and stop. Information on what type to use is found in the side information part of each frame.

Producing 36 samples from 18 frequency lines means that only 18 of the samples are unique. Therefore IMDCT is said to use a 50% overlap [7]. The 36 values from the windowing operation are divided into two groups. The first half of the block of 36 values is overlapped with the second half of the previous block. The second half of the actual block is stored to be used in the next block. Overlapping is carried out by interleaving (adding) values from the lower group with corresponding values from the higher group from the previous frame.

5.11 Frequency inversion

The output of overlap add consists of 18 time samples for each of 32 polyphase subbands. Before processing the time samples into synthesis polyphase filter bank, every odd time sample of every odd subband should be multiplied by -1 to compensate for frequency inversion.

5.12 Synthesis via polyphase filter bank

The final step in the decoding process is to synthesize the 18 time samples for each of the 32 subbands in each granule, into 18 blocks of 32 PCM samples.

6 Implementation

The header and side information are the most important blocks to get the details of an audio file. So they have been considered and were successfully implemented in Matlab. The practical values obtained for header are given under Header information for frame 1. Side information values are presented under Side information details for frame 2 and in Table 13. The MP3 file that has been taken for implementation is 'fg_nufolk_snippet_mix.mp3'.

6.1 Header information for frame 1

From the Matlab program that is written the total bits present in the test file under consideration could be extracted. The header starts at bit number 32769 and the total number of bits are 37055216. The total number of headers that has been obtained is 11749.

As expected, the first 12 bits found in the header are '1111 1111 1111' which means that it is the syncword. After finding the syncword, remaining bits are examined according to the ISO specifications. After examining the remaining bits, we could confirm that the test file is an MP3 file.

Result: The audio file under test is found to be MP3 encoded in joint stereo mode with 44.1 kHz sampling frequency which has a bitrate of 128kbps.

6.2 Side information details for frame 2

It is always advised to look at three frames before confirming whether the file is MP3 encoded or not. So frame 2 has been considered for side information details.

From the `main_data_begin` pointer it is found that, the main data for frame 2 starts at 198 bytes before the header and side information. For frame 1 the `main_data_begin` pointer has the value zero, which means that the side information follows immediately the header. So this confirms the fact that the main data need not always follow the side information and the main data can be placed in any of the previous frames using the bit reservoir technique. Side information results for granule one and channel one that are obtained after Matlab implementation are presented in Table 13. A graphical representation of the results is presented in Figure 10.

Variables	Values obtained in bits
Part2_3_length	188
Big_values	90
Global_gain	90
Scalefac_compress	9
Window_switching_flag	1
Block_type	2
Mixed_block_flag	1
Table_select	25
Subblock_gain	4
Preflag	1
Scalefac_scale	1
Count1table	1

Figure 9: Side information values obtained after implementation

6.3 Problems encountered during implementation

Implementation was done in Matlab in the signal processing department at BTH on a P4, 1.5 GHz, 512 MB RAM, Dell machine. The header and side information results obtained show that they exactly match the standard, which means that implementation has been successful.

During the implementation, the following are the problems encountered.

- 1.They are some false sync words, which actually felt that it could be the start of the sync word. Proper care should be taken to verify the authenticity of the sync word. False sync words can be avoided by taking three or four frames into consideration.
2. The computer reads the bits in little endian format where as big endian is the general way of bit representation around the world. There are specific commands in Matlab, which are to be considered during implementation, so that the big endian way of representing the bits is computed.

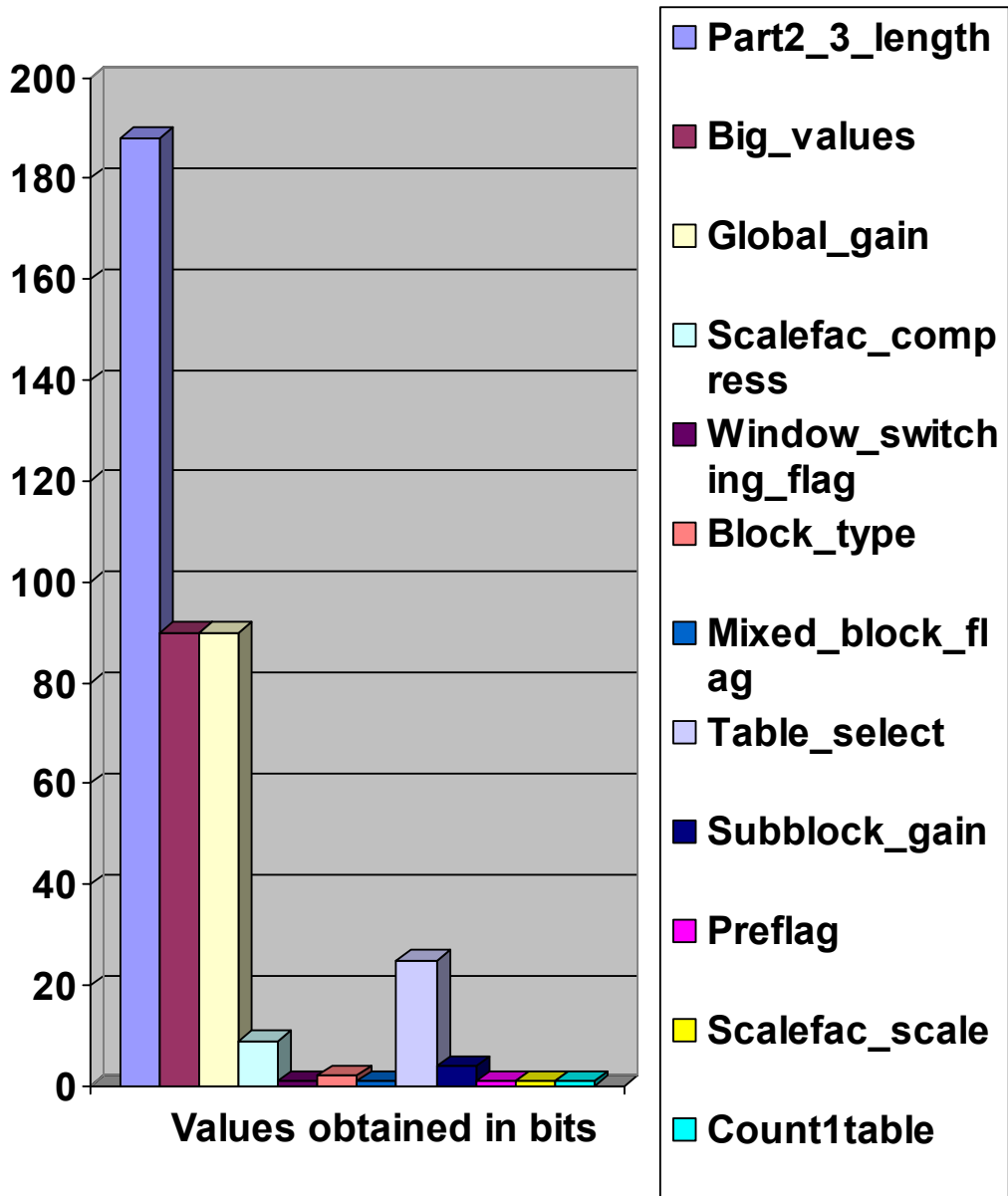


Figure 10: Graphical representation of side information values

7 Conclusions

An MP3 decoder has been thoroughly studied and some of the decoder blocks have been successfully implemented in Matlab for deeper understanding. As expected, implementation of a decoder has been found difficult. The standard does not provide a clear picture of the decoder implementation. Parts of the standard are unclear for which reference of other existing works is a must.

Bearing in mind, that decoder only has to decode the bitstream that has been encoded in some sort, it can be said that decoder is relatively easier to implement than an encoder. With time and proper resources, it is advised to implement an entire decoder and real time implementation of it would be challenging.

Better compression ratios can be obtained by using good IMDCT algorithms and transforms. This thesis basically provides good information for those who are interested in either software or hardware implementation of an MP3 decoder.

8 References

- [1] ISO / IEC 11172-3: Information technology – Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s – Part3: Audio, ISO/ IEC 1993.
- [2] M. Ruckert, *Understanding MP3*, Vieweg, 2005, ISBN 3-528-05905-2.
- [3] Bradenburg K. and Popp H., “*An introduction to MPEG Layer-3*”, EBU Technical review, June 2000.
- [4] Bradenburg K., “*MP3 and AAC explained*”, in *Proc. of the AES 17th Int. Conf. on high quality audio coding*, 1999.
- [5] Raissi R., “*Theory behind MP3*”, 5th October 2005, <<http://www.mp3-tech.org>>.
- [6] Mathew M., Bhat V., Thomas S.M., Yim C., “*Modified MP3 Encoder using complex modified discrete cosine transform*”, ICME 2003.
- [7] Fältmann I, Hast M, Lundgren A, Malki S, Montnemery E, Rångevall A, Sandvall J, Stamenkovic M., “*A Hardware implementation of an MP3 decoder*”, Digital IC project, LTH, Sweden, May 2003.
- [8] Geoff Nicholson, “*MP3 Explained: A Beginners guide*” 18th January 2006, <http://www.hitsquad.com/smm/news/9903_109/?nl9905>.
- [9] S. Haker, *MP3: The Definitive guide*, O’Reilly, 2000, ISBN 1-56592-661-7.
- [10] Predrag Supurovic, “*MPEG script*”, 10th January 2006, <<http://www.dv.co.yu/mpgscript/mpeghdr.htm>>.
- [11] Gabriel Bouvigne, “*MP3 Tech*”, 5th October 2005, <<http://www.mp3-tech.org>>.
- [12] “*MP3 converter*”, 15th December 2005, <<http://www.mp3-converter.com>>.
- [13] “*Id3 tags*”, 28th December 2005, <<http://www.id3.org>>.