MPEG-1/2 audio layer-3(MP3) ON THE RISC based ARM PROCESSOR (ARM92SAM9263)

E.KALPANA[1] V.SRIDHAR [2] M.RAJENDRA PRASAD[2]

[1] Assistant Professor, ECE, Vidya Jyothi Institute of Technology, Hyderabad
[2] Assistant Professor, ECE, Vidya Jyothi Institute of Technology, Hyderabad
[3] Associate Professor, ECE, Vidya Jyothi Institute of Technology, Hyderabad
[1]kalpanaerla@yahoo.co.in , [2]varadalasri@gmail.com, [3]rajendraresearch@gmail.com

ABSTRACT: MPEG-1/2 audio layer-3(MP3) is a most popular format for playback 0f high quality compressed audio for portable devices such as audio players and mobile phones. Typically these devices are based on DSP architecture or RISC processor. While the DSP architecture is more efficient for implementing the MP3 algorithm, the challenges of RISC implementation are less understood. This project describes the challenges and optimization techniques useful for implementing the MP3 decoder algorithm on the RISC based ARM7DMI processor. Some of these techniques are generic and hence applicable to any audio code implementation on RISC-based platforms. This project gives results which are among the best in the industry, indicate that stereo MP3 at 44.1KHZ and 128KBPS can be decoded using 30-40MIPS on the ARM7TDMI. The target machine runs on ARM92SAM9263. The music data can be compressed into 1/10 size while maintaining CD quality, when the layer-3(MP3) algorithm is applied to the system. In addition the output of this decoder is fully bit complement with the standard on the ISO test vectors.

Keywords: MPEG-1/2 audio layer-3, ARM7DMI, MIPS, CD, MP3,DSP ARCHITECHTURE,RISC,ARM92SAM9263

I . INTRODUCTION

MP3 has changed the way people listen to music on the Internet. Internet users, music lovers who would like to download highly compressed dig ital files at near CD quality are the most benified. Uncompressed digital CD-quality audio signals consume a large amount of data(CD quality audio consumes 10MB of memory space to store 1 minute of song) and are therefore not suited for storage and transmission. The need to reduce this amount without any noticeable quality loss was stated in the late 80's by the International Organization for Standardization (ISO). A working group within the ISO referred to as the Moving Pictures Experts Group (MPEG), developed a standard that contained several techniques for both audio and video compression. The audio part of the standard included three modes with increasing complexity and performance. The third mode, called Layer III, manages to compress CD music from 1.4 Mbit/s to 128 kbit/s with almost no audible degradation. This technique, also known as MP3, has become very popular and is widely used in applications today. The ability to maintain impressive sound quality whilst reducing the data requirements by a factor of 1:10 or more, has led to an explosion of content on the Internet.

Traditionally, a DSP processor may have been specified as an implementation platform for MP3. Most of the devices are battery operated devices the key requirement of these devices are power consumption should be low. However the analysis of the key technical requirements for MP3 shows that a programmable software solution mapped to a RISC based ARM processor and important difficulty that is faced in such cases is that of optimizing DSP intensive algorithms on a general purpose RISC processor that is not explicitly designed for DSP applications.

MP3 decoding is a computationally expensive process, requiring extensive optimizations to be able to run on portable devices. The most critical parts of the decoding process are the *Inverse Modified Discrete Cosine Transform* (IMDCT) and the *Synthesis Polyphone FilterBank*. These two are time consumed processes are optimized by handwritten Assembly code and this master thesis describes the implementation of a system for MP3 audio playback in real-time on an ARM7 based system.

The purpose of this paper is to implement an MP3 decoder that runs on ARM7TDMI based machine. The target machine (AT91SAM9263) runs on an ARM7TDMI processor.

2. 1.THE ANATOMY OF THE MP3 FILE:

All MP3 files are divided into smaller fragments called *frames*. Each frames stores1152audio samples and lasts for 26 ms. This means that the frame rate will be around 38 fps. In addition a frame is subdivided into two *granules* each containing 576 samples. Since the bitrate determines the size of each sample, increasing the bitrate will also increase the size of the frame. The size is also depending on the sampling frequency according to following formula

Frame length = [144 * (bit rate / sampling frequency)] bytes

Padding refers to a special bit allocated in the beginning of the frame. It is used in some frames to exactly satisfy the bitrate requirements. If the padding bit is set the frame is padded with 1 byte. Note that the frame size is an integer

Ex: 144*128000/44100 = 417 bytes

2.1.1. The Frame Layout:

A frame consists of five parts; header, CRC, side information, main data and ancillary data, as shown in figure below

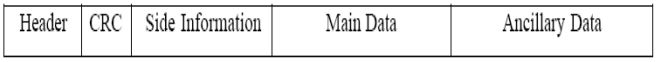

Figure 2.1: The frame layout

1. **Frame header:**

The header is 32 bits long and contains a synchronization word together with a description of the frame. The synchronization word found in the beginning of each frame enables MP3 receivers to lock onto the signal at any point in the stream. This makes it possible to broadcast any MP3 file. A receiver tuning in at any point of the broadcast just have to search for the synchronization word and then start playing. A problem here is that spurious synchronization words might appear in other parts of the frame. A decoder should instead check for valid sync words in two consecutive frames, or check for valid data in the side information, which could be more difficult.

Figure 2.1 shows an illustration of the header

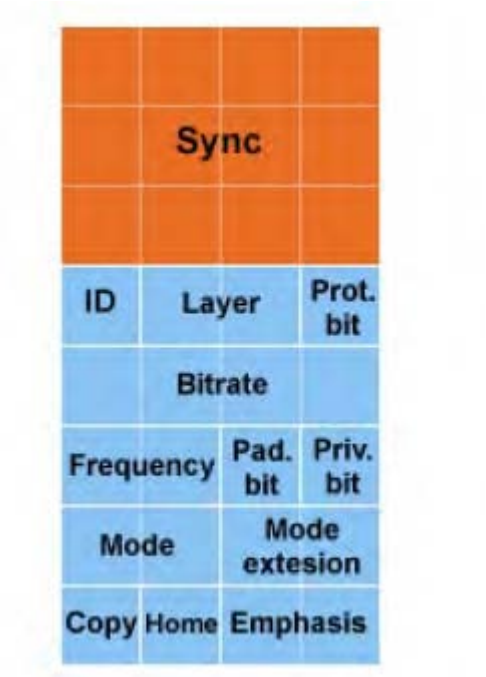

Figure 2.2: The MP3 frame header

Sync (12 bits):

This is the synchronization word described above. All 12 bits must be set, i.e. ‘111111111111’.

Id (1 bit):

Specifies the MPEG version. A set bit means that the frame is encoded with the MPEG-1 Standard, if not MPEG-2 is used.

Some add-on standards only use 11 bits for the sync word in order to dedicate 2 bits for the id.

00	MPEG-2.5 (Later extension of MPEG-2)
01	Reserved
10	MPEG-2
11	MPEG-1

Table 2.1: Bit values when using two id bits

Layer (2 bits)
See Table 5.2

00	reserved
01	Layer III
10	Layer II
11	Layer I

Table 2.2: Definition of layer bits

Protection Bit (1 bit)
If the protection bit is set, the CRC field will be used.
Bitrate(4 bits)*:*
These four bits tells the decoder in what bit rate the frame is encoded. This value will be the same for all frames if the stream is encoded using CBR.
Table 2.3 shows the defined bit values.

Bitrate	MPEG 1			MPEG 2, 2.5 (LSF)	
Index	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 2.3: Bitrate definitions

Frequency (2 bits):
2 bits that give the sampling frequency, see Table 2.4.

Bits	MPEG1	MPEG2	MPEG2.5
00	44100 Hz	22050 Hz	11025 Hz
01	48000 Hz	24000 Hz	12000 Hz
10	32000 Hz	16000 Hz	8000 Hz
11	reserv.	reserv.	reserv.

Table 2.4: Definition of accepted sampling frequencies

Padding bit (1 bit):

An encoded stream with bitrate 128 kbit/s and sampling frequency of 44100 Hz will create frames of size 417 bytes.

To exactly fit the bitrate some of these frames will have to be 418 bytes. These frames set the padding bit.

Private bit (1 bit):

One bit for application-specific triggers.

Mode (2 bits)

Specifies what channel mode is used according to Table 2.5.

00	Stereo
01	Joint Sereo
10	Dual Channel
11	Single Channel

Table 2.5: Channel Modes and respective bit values

Mode Extension (2 bits):

These 2 bits are only usable in joint stereo mode and they specify which methods to use. The joint stereo mode can be changed from one frame to another, or even switched on or off. To interpret the mode extension bits the encoder needs the information in Table 2.6.

Bits	Intensity stereo	MS stereo
00	Off	Off
01	On	Off
10	Off	On
11	On	On

Table 2.6: Definition of mode extension bits

Copyright Bit (1 bit):

If this bit is set it means that it is illegal to copy the contents.

Home (Original Bit) (1 bit)

The original bit indicates, if it is set, that the frame is located on its original media.

Emphasis (2 bits):

The emphasis indication is used to tell the decoder that the file must be de-emphasized i.e. The decoder must 're-equalize' the sound after a Dolby- like noise suppression. It is rarely used.

00	None
01	50/15 ms
10	Reserved
11	CCITT J.17

Table 2.7: Noise suppression model

2. CRC *(0 bytes, 16 bytes)**:***
This field will only exist if the protection bit in the header is set and makes it possible check the most sensitive data for transmission errors. Sensitive data is defined by the standard to be bit 16 to 31 in both the header and the side information. If these values are incorrect they will corrupt the whole frame whereas an error in the main data only distorts a part of the frame. A corrupted frame can either be muted or replaced by the previous frame.

3. Side information:
The side information part of the frame consists of information needed to decode the main data. The size depends on the encoded channel mode. If it is a single channel bitstream the size will be 17 bytes, if not, 32 bytes are allocated. The different parts of the side information are presented in Table 2.8 and described in detail below.
The length of each field will be specified in parenthesis together with the fieldname above the actual description. If one length value is written the field size is constant. If two values are specified the first value will be used in mono mode and the second will be used for all other modes, thus these fields are of variable length. All tables below will assume a mono mode. The tables will change depending on mode since separate values are needed for each channel

main_data_begin	private_bits	scfsi	Side_info gr. 0	Side_info gr. 1

Table 2.8: Side information

main_data_begin (9 bits):
Using the layer III format there is a technique called the *bit reservoir* which enables the left over free space in the main data area of a frame to be used by consecutive frames. To be able to find where the main data of a certain frame begins the decoder has to read the main_data_begin value. The value is as a negative offset from the first byte of the synchronization word. Since it is 9 bits long it can point (2^9 – 1) * 8 = 4088 bits. This means that data for one frame can be found several previous frames. Note that static parts of a frame like the header, which is always 32 bytes, are not included in the offset. If main_data_begin =0 the main data starts directly after the side information.

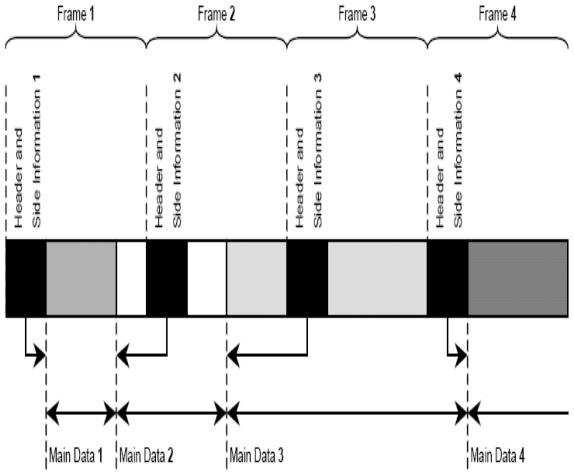

Figure 2.3: Use of the bit reservoir

private bits (5 bits, 3 bits):
Bits for private use, these will not be used in the future by ISO.
scfsi (4 bits, 8 bits):
The Scale Factor Selection Information determines whether the same scale factors are transferred for both granules or not. Here the scale factor bands are divided into 4 groups According to Table 2.9.

group	scalefactor bands
0	0,1,2,3,4,5
1	6,7,8,9,10
2	11,12,13,14,15
3	16,17,18,19,20

Table 2.9: Scale factor groups

4 bits per channel are transmitted, one for each scalefactor band. If a bit belonging to aScalefactor band is zero the scalefactors for that particular band are transmitted for each
granule. A set bit indicates that the scalefactors for granule0 are also valid for granule1e1.This means that the scalefactors only need to be transmitted in granule0, the gained bits can be used for the Huffman coding.If short windows are used (block_type = 10) in any granule/channel, the scalefactors arealways sent for each granule for that channel.

Side info for each granule:

The last two parts of a frame have the same anatomy and consists of several subparts as shown in . These two parts store particular information for each granule respectively

part2_3_length	big_values	global_gain	scalefac_compress
windows_switching_flag	block_type	mixed_block_flag	table_select
subblock_gain	region0_count	region1_count	preflag
scalefac_scale	count1table_select		

Table 2.10: Fields for side information for each granule

par2_3_length (12 bits, 24 bits):

States the number of bits allocated in the main data part of the frame for scale factors (part2) and Huffman encoded data (part3). 12 bits will be used in a single channel mode whereas in stereo modes the double is needed. This field can be used to calculate the location of the next granule and the ancillary information (if used).

big values (9 bits, 18 bits):

The 576 frequency lines of each granule are not coded with the same Huffman code table.
These frequencies range from zero to the Nyquist frequency and are divided into five *regions* (see Figure 3.4). The purpose of this partitioning is to allow different Huffman tables to different parts of the spectrum in order to enhance the performance of the Huffman encoder.

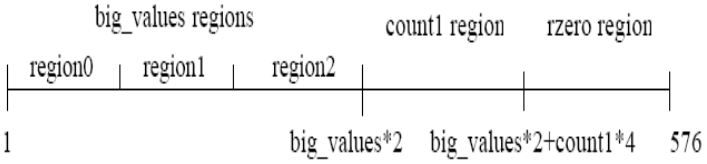

Figure 2.4: Regions of the frequency spectrum

Partitioning is done according to the maximum quantized values. This is done with the assumption that values at higher frequencies are expected to have lower amplitudes or does not need to be coded at all.

The zero regions represent the highest frequencies and contain pairs of quantized values equal to zero. In the count1 region quadruples of quantized values equal to -1, 0 or 1 reside. Finally, the big values region contains pairs of values in representing the region of the spectrum which extends down to zero. The maximum absolute value in this range is constrained to 8191. The big values field indicates the size of the big values partition hence the maximum value is 288.

global gain (8 bits, 16 bits):

Specifies the quantization step size, this is needed in the requantization block of
the decoder.

scalefac_compress (4 bits, 8 bits):
Determines the number of bits used for the transmission of scale factors. A granule can be divided into 12 or 21 scale factor bands. If long windows are used (block_type = {0,1,3}) the granule will be partitioned into 21 scale factor bands. Using short windows (block_type = 2) will partition the granule into 12 scale factor bands. The scale factors are then further divided into two groups, 0-10, 11-20 for long windows and 0-6, 7-11 for short windows.
The scalefac_compress variable is an index to a defined table (see Table 5.11). slen1 and slen2 gives the number of bits assigned to the first and second group of scalefactor bands respectively.

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 2.11: scalefac_compress table

windows_switching_flag (1 bit, 2 bits):
Indicates that another window than the normal is used block type,mixed_block_flag and subblo16 Also when windows_switching_flag is set all remaining values not being in region0 are contained in region1 thus region2 is not used.
block type (2 bits, 4 bits):
This field is only used when windows_switching_flag is set and indicates the type of window used for the particular granule (see Table 5.12, all values but 3 are long windows). The value 00 is forbidden since block type is only used when other than normal windows are used.

block_type	window type
00	forbidden
01	start
10	3 short windows
11	end

Table 2.12: block type definition

mixed_blockflag (1 bit, 2 bits):
This field is only used when windows_switching_flag is set.
The mixed_block_flag indicates that different types of windows are used in the lower and higher frequencies. If mixed_block_flag is set the two lowest sub bands (see 6.1) are transformed using a normal window and the remaining 30 sub bands are transformed using the window specified by the block type variable.

table select ((10 bits, 20 bits) or (15 bits, 30 bits):
There are 32 possible Huffman code tables available in the standard. The value of this field gives the Huffman table to use when decoding and its size is 5 bits, i.e. 32 different values, for each region, granule and channel. The table select only specifies the tables to use when decoding the big values partition. The table specified is dependent on the local statistics of the signal and by the maximum quantization allowed to quantize the 576 frequency lines in the granule. As stated above, when the windows_switching_flag is set region2 is empty so only two regions are coded. This implies that in mono mode 5*2*1 = 10 bits are needed and in stereo mode5*2*2 = 20 bits are needed if windows_switching_flag = 1. Using all regions (windows_switching_flag = 0) the bits needed will be 5*3*1 = 15 and 5*3*2 = 30 respectively.
subblock_gain (9 bits, 18 bits):
This field is only used when windows_switching_flag is set and when block type = 10, although it is transmitted independently of block type. This 3 bit variable indicates the gain offset from global gain for each short block.
region0_count (4 bits, 8 bits), region1_count (3 bits, 6 bits):
region0_count and region1_count contains one less than the number of scale factor bands in region0 and region1 respectively. The region boundaries are adjusted to the partitioning of the frequency spectrum into scale factor bands. If short windows are used the number of each windows is counted.
Preflag (1 bit, 2bits):
This is a shortcut for additional high frequency amplification of the quantized values. If preflag is set, the values of a defined table are added to the scale factors. If block type = 10, i.e. short blocks, preflag is never used.
scalfac_scale (1 bit, 2bits):
The scale factors are logarithmically quantized with a step size of 2 or v2 according to Table 2.13.

scalfac_scale	step size
0	v2
1	2

Table 2.13: Quantization step size applied to scale factors

count1table_select (1 bit, 2bits):
Two possible Huffman code tables are available for the count1 region. This field specifies which table to apply.

4. Main Data:
The main data part of the frame consists of scale factors, Huffman coded bits and ancillary data.

- ***Scale factors***

  Scalefactors for 44.1 kHz, long windows (576 frequency lines)

scalefactor band	width	start index	end index
0	4	0	3
1	4	4	7
2	4	8	11
3	4	12	15
4	4	16	19
5	4	20	23
6	6	24	29
7	6	30	35
8	8	36	43
9	8	44	51
10	10	52	61
11	12	62	73
12	16	74	89
13	20	90	109
14	24	110	133
15	28	134	161
16	34	162	195
17	42	196	237
18	50	238	287
19	54	288	341
20	76	342	417

Table 2.14 scalefactor bands

The purpose of scale factors is to reduce the quantization noise. If the samples in a particular scale factor band are scaled the right way the quantization noise will be completely masked. One scale factor for each scale factor band is transmitted. The scfsi field determines if the scale factors are shared between the granules or not. The actual bits allocated for each scale factor depends on the scalefac_compress field.
The subdivision of the spectrum into scale factor bands is fixed for every window length and sampling frequency and stored in tables in the coder and decoder (a table of scale factor bands used for long windows with a sampling frequency of 44.1 kHz is presented in(table no 2.15).

- ***Huffman code bits***

This part of the frame contains the Huffman code bits. Information on how to decode these is found in the side information. For the three sub regions in the big values region always pairs are encoded. For instance, the Huffman table no. 7 (table3.14)may be applied to big values. *X* and *y* are the pair of values to be coded, *hen* specifies the length of the Huffman code for x and y and hcode is the actual Huffman code representing x and y. In the case of count1 values they are encoded in quadruple les, v,w,x,y. Only tables A and B support coding in this region. The zero region is not Huffman coded but run length coded since all values are zero.
Depending on whether long or short blocks are used, the order of the Huffman data differs. If long blocks are used, the Huffman encoded data is ordered in terms of increasing frequency.

Huffman code table 7

x y	hlen	hcod
0 0	1	1
0 1	3	010
0 2	6	001010
0 3	8	00010011
0 4	8	00010000
0 5	9	000001010
1 0	3	011
1 1	4	0011
1 2	6	000111
1 3	7	0001010
1 4	7	0000101
1 5	8	00000011
2 0	6	001011
2 1	5	00100
2 2	7	0001101
2 3	8	00010001
2 4	8	00001000
2 5	9	000000100
3 0	7	0001100
3 1	7	0001011
3 2	8	00010010
3 3	9	000001111
3 4	9	000001011
3 5	9	000000010
4 0	7	0000111
4 1	7	0000110
4 2	8	00001001
4 3	9	000001110
4 4	9	000000011
4 5	10	0000000001
5 0	8	00000110
5 1	8	00000100
5 2	9	000000101
5 3	10	0000000011
5 4	10	0000000010
5 5	10	0000000000

Table2.15 huffmantable no.7

5. Ancillary Data:

The ancillary data can hold user-defined information. The decoder does not process the information, more likely it is sent to the application using the decoder. The ancillary data can hold the song title or optional song information. The drawback is that the ancillary data is placed in the frame by the encoding process and can not easily changed at run-time. If it is being streamed over the network, live broadcasting information cannot be included in the ancillary data. Today, Internet radio stations often sends broadcasting information with a continuous, predefined interval. Streaming MP3 decoders must be able to handle this case correctly.

2.2. OVERVIEW OF THE MP3 ENCODER:

The encoding process is very complex and not fully described here:

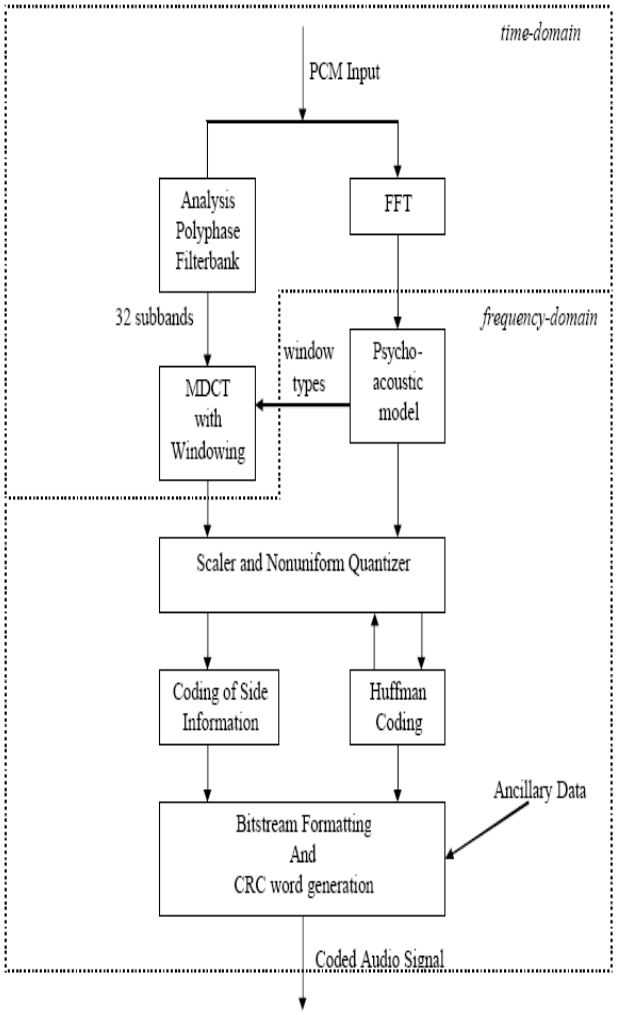

Figure 2.5. MPEG-1 Layer III encoding scheme

The input to the encoder is PCM audio signal(CD quality audio at bitrate of 1.4Mbits/s).The output from the encoder is a MP3 bitstream further this is given to the decoder to get the original PCM output.

The PCM input is given to the analysis filter bank. The output from this is given to psychoacoustic and MDCT. MDCT is used for the conversion of time domain to frequency domain. The masking thresholds are used to iteratively determine how many bits are needed in each critical band to code the samples so that the quantization noise is not audible. This process is done in quantization block. The quantized samples are Huffman coded and stored in the bit stream along with the scale factors and side information. The output from the encoder is a MP3 further this is given to the decoder to get the original PCM output.

2.2.1. Analysis Polyphase Filter bank:

A sequence of 1152 PCM samples are filtered into 32 equally spaced frequency sub bands depending of the Nyquist frequency of the PCM signal. If the sample frequency of the PCM signal is 44.1 kHz the Nyquist frequency will be 22.05 kHz. Each sub band will be approximately 22050/32 = 689 Hz wide. The lowest sub band will have a range from 0-689 Hz, the next sub band 689 – 1378 Hz, etc. Every sample (might) contain signal components from 0 – 22.05 kHz that will be filtered into appropriate sub band.

2.2.2. FFT:

Simultaneously as the signal is processed by the polyphase filter bank it is also transformed to the frequency domain by a Fast Fourier Transform. Both a 1024 and a 256 point FFT are performed on 1152 PCM samples at the time to give higher frequency resolution and information on the spectral changes over time.

2.2.3. Psychoacoustic Model:

This block retrieves the input data from the FFT output. Since the samples are in the frequency domain they can be applied to a set of algorithms. These algorithms will model the human sound perception and hence

they can provide information about which parts of the audio signals that is audible and which parts are not. This information is useful to decide which window types the MDCT should apply and also to provide the Nonuniform Quantization block with information on how to quantize the frequency lines.(frequently appearing bit patterns by short bit patterns and infrequently appearing bit patterns by longer bit patterns.)

To know which window type to send to the MDCT block the two presently FFT spectra and the two previous spectra are compared. If certain differences are found a change to short windows requested. As soon as the differences fades away the MDCT block will be informed to change back to long (normal) windows (see Figure 2.6).

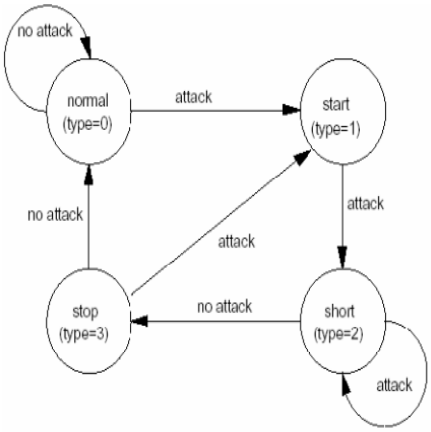

Figure 2.6: Window switching decision

The Psychoacoustic Model also analyzes the FFT spectrum to detect dominant tonal components and for each critical band masking thresholds are calculated. Frequency components below this threshold are masked out. Recall that the scalefactor bands are roughly equivalent to the critical bands of human hearing. The thresholds limits for each scalefactor band are used by the quantization block to keep the quantization noise below these limits.

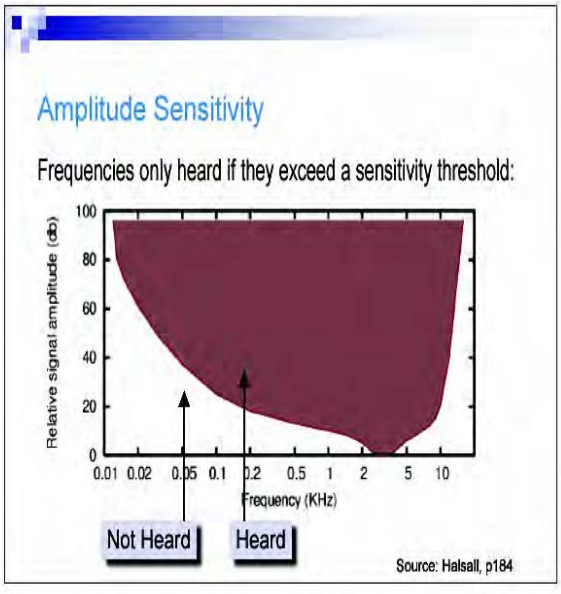

Figure 2.7 Masking threshold of each subband

2.2.4. Modified Discrete Cosine Transform (MDCT):

By applying a modified discrete cosine transform to each time frame of sub band samples the 32 sub bands will be split into 18 finer sub bands creating a granule with a total of 576 frequency lines. But prior to the MDCT each sub band signal has to be windowed.

Windowing is done to reduce artifacts caused by the edges of the time- limited signal segment. For steady state signals long window is used and for transient signals short windows are used. The aliasing introduced by the polyphase filter bank is now removed to reduce the amount of information that needs to be transmitted. This is achieved using a series of butterfly computations that add weighted, mirrored versions of adjacent sub bands to each other

2.2.5. Nonuniform Quantization :
Nonuniform Quantization block with information on how to qua ntize the frequency lines.(frequently appearing bit patterns by short bit patterns and infrequently appearing bit patterns by longer bit patterns.) The frequency lines are quantized using power law computation.
2.2.6. Huffman Encoding:
The quantized values are Huffman coded. Each division of the frequency spectrum can be coded using different tables. The Huffman coding is one of the major reasons why the MPEG-1 Layer III retains a high quality at low bit rate.
2.2.7. Coding of Side Information:
All parameters generated by the encoder are collected to enable the decoder to reproduce the audio signal. These are the parameters that reside in the side information part of the frame.
2.2.8. Bitstream Formatting CRC word generation
In this final block the defined bitstream is generated The frame header, side information, CRC, Huffman coded frequency lines etc are put together to form frames.

3. MP3 DECODING:
3.1. THE MP3 DECODING PROCESS:
The decoding process creates output samples from an MP3 bitstream. MP3 is based on small packages, or *frames*, each producing a few milliseconds of sound output. A frame contains information about how it should be properly decoded, as well as compressed audio data.

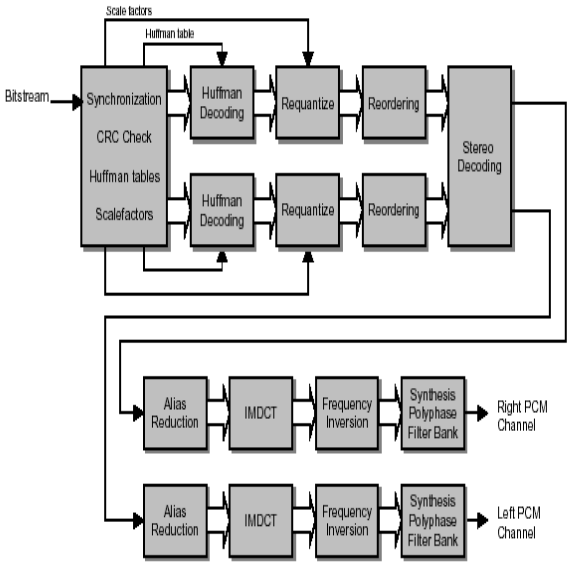

Figure 3.1. The MP3 decoding process

The audio data is first Huffman decoded, producing symbols that represent 576 scaled frequency lines. The next step in the process is to *descale* the symbols to normal frequency lines, using the *scalefactors* that are provided in the frame. If the frame contains more than one channel of audio information, the *stereo decoder* recreates the frequencies for the left and right channels. The last decoder steps produce time samples from the frequency lines using complex calculations by running it through an *Inverse Discrete Cosine Transform* (IDCT) and a *synthesis polyphase filter bank*. An overview of the decoding process can be found in Figure 3.1.
3.1.1. Frame synchronization:

Before any decoding can take place, the start of the frame must be found. A synchronization word containing a bit pattern of 12 consecutive ones, '1111 1111 1111', is placed at the beginning of each frame header. It is possible that the same bit pattern exists in other parts of the frame as well, and the decoder cannot tell if it has found a synchronization word or just some arbitrary data. If a CRC is present, it can be used to confirm that a new valid header has been found. If the CRC is not present, the decoder can still make modest attempts to confirm that it is a valid synchronization word by searching for invalid combinations of fields in the header. The header contains enough information to calculate the size of the frame, which is also the start of the next frame, using the bitrate and frequency fields. If a frame is corrupted, it is no longer possible to find the exact position of the next frame. This is why the synchronization word is present in every frame.

3.1.2. Huffman decoding:

Huffman encoding is a loss-less coding scheme that produces Huffman codes from input symbols. The mapping of symbols to Huffman codes is based on the statistic contents of the input sequence. Symbols that occur more frequently are coded with a short code, while symbols that occur less frequently are coded with longer codes. In average, this will reduce redundancy, but is only possible if some combinations of input sequences are more likely to appear than others.

The decoding procedure is based on several Huffman tables that are used for mapping Huffman codes to symbols. The decoder compares the input sequence with the information in the Huffman table. If it finds a match, the corresponding output symbol is fetched. There is one and only one symbol for every possible input sequence.

In MP3, there are 32 different Huffman tables. The tables are predefined and based on statistics suitable for compressing audio information. The side information specifies which table to use for decoding the current frame.

The output from the Huffman decoder is 576 scaled frequency lines, represented with integer values. The frequency lines are divided into three partitions: *Big-values*, *count1* and *rzero*, as can be seen in Figure 3.2.*Big-values* contains the lowest frequency lines and are coded with the highest precision. The range of *big-values* is 15. The next partition, *count1*, can only be coded with the values 0, 1 or -1. The last partition, *rzero*, represents the highest frequencies and is not a part of the bitstream. Instead, *rzero* represents the frequency lines that have been removed by the encoder, and it should be filled with zeros by the decoder. The boundaries of the partitions are specified in the side information, and are selected by the psycho acoustic model during the encoding process.

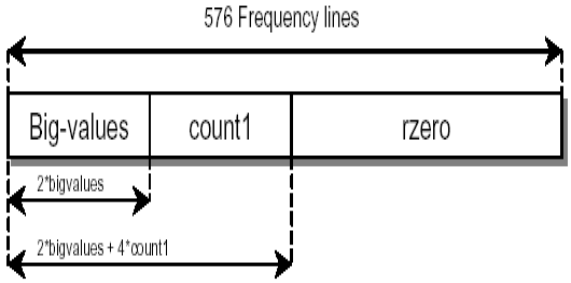

Figure 3.2 The frequency lines are divided into three partitions

When decoding *big-values*, the huffman table will produce two frequency lines for each input sequence and the *count1* partition will produce four frequency lines for each input sequence. It is possible to send larger *big-values* than 15 if extra precision is needed. It is done with a so-called escape sequence, and is only available for *big-values*. If the Huffman decoder finds the value 15, it assumes an escape sequence is present and reads out a number of bits from the stream. The escape value is directly added to the original value 15. The number of bits to read
for each escape sequence is specified by the Huffman table that is currently in use, and is called *linbits*. An example of the 576 frequency lines from the Huffman decoder is shown in Figure 3.3 The horizontal lines represent the maximum range from -15 to 15. All exceeding values are coded with escape sequences.

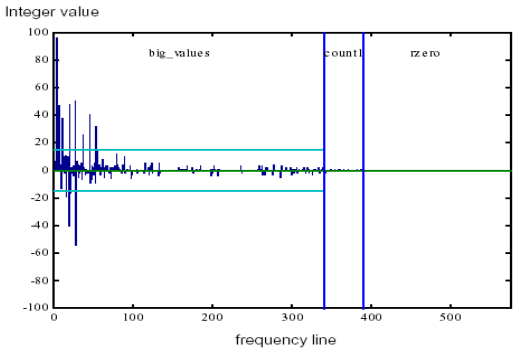

Figure 3.3 Frequency lines generated by the Huffman decoder

3.1.3. Descaling:

The symbols from the Huffman decoder can reconstruct the original frequency lines by using the scalefactors in the side information. In the encoder process, the frequency lines were divided into a scalefactor and a small integer number, normally in the range –15 to 15. The scalefactors were saved separately and the integer number were Huffman encoded. Together, the scalefactors and the Huffman coded data forms the main data. An overview of the descaling process can be found in Figure 3.4.

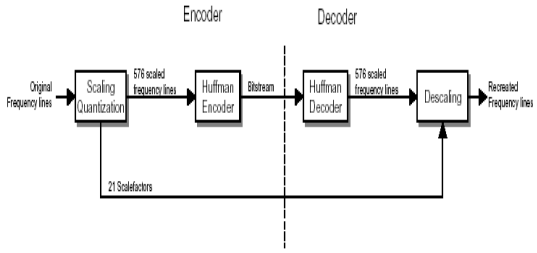

Figure 3.4 The descaling process

There is not one scalefactor for each frequency line. Instead the frequency lines are divided into 21 groups, called *scalefactor bands*, each using its own scalefactor. The number of frequency lines in each group depends on the frequency band that it is representing. A low frequency scalefactor band contains fewer values than a scalefactor band representing higher frequencies.

The descaling process uses global scaling information from the side information that should be applied to all frequency groups, as well as the 21 scalefactors that should be applied to each group independently. The complete descaling equations for short and long blocks respectively are presented below:

$$xr_i = sign(is_i) \cdot |is_i|^{\frac{4}{3}} \cdot 2^{\frac{1}{4}(global\_gain[gr]-210-8\cdot subblock\_gain[window][gr])} \cdot 2^{-(scalefac\_multiplier \cdot scalefac\_s[gr][ch][sfb][window])}$$

$$xr_i = sign(is_i) \cdot |is_i|^{\frac{4}{3}} \cdot 2^{\frac{1}{4}(global\_gain[gr]-210)} \cdot 2^{-(scalefac\_multiplier \cdot (scalefac\_l[gr][ch][sfb]+preflag[gr]\cdot pretab[sfb]))}$$

In the equations above, *is* represents the input values generated by the huffman decoder and *xr* represent the output sequence from the descaling process. *global_gain*, *subblock_gain*, *scalefac_multiplier* and *preflag* can all be found in the side information. *Scalefac_s* and *scalefac_l* are the scalefactors for long and short blocks respectively, found in the main data. The *pretab* is a predefined table in the ISO/IEC 11172-3 standard.

3.1.4 . Reordering:

During the encoding process, the MDCT can arrange the output in two different ways.
Normally the output from the MDCT is sorted by subbands in increasing frequency. When a short block is decoded, a short window will be used. Figure 3.5 shows short and long windows, applied on short and long blocks respectively. The output will in this case be sorted on subbands, then on windows and then on increasing frequency.

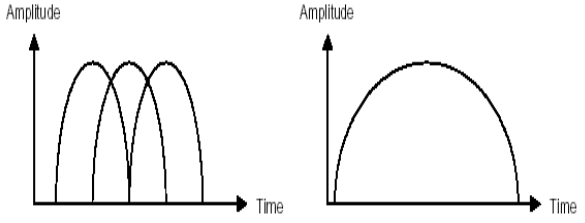

Figure 3.5 Short and long windows used by the MDCT and the IMDCT

Reordering is only applied on short blocks, sorting the frequency lines first by subbands and then by frequency.

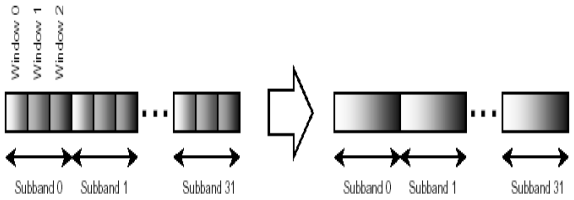

Figure 3.6. Reordering of short block types into continuous frequency lines

Figure 3.6 shows the reordering of a short block into subbands with increasing frequency.
The shading in the subband represents the frequencies. A darker color corresponds to a higher frequency.

3.1.5. Stereo processing:

There are four different channel modes in MP3: *Single channel*, *dual channel*, *MS stereo* and *Intensity stereo*. When using single channel and dual channel, no stereo processing is
required. Dual channel means that the two channels are coded independently.

3.1.6. MS stereo:

Middle/side stereo, or MS stereo for short, is a method to remove redundancy between
channels. This is achieved by coding the sum and difference of the signal instead of the left and right channels.

$$L_i = \frac{M_i + S_i}{\sqrt{2}} \quad \text{and} \quad R_i = \frac{M_i - S_i}{\sqrt{2}}$$

Mi is the sum of the two channels, and Si is the difference. Li and Ri are the output samples.

3.1.7. Intensity stereo:

Intensity stereo is a more complex method of stereo coding that is divided in a magnitude part and a stereo position part. Intensity stereo only affects the coding of the higher frequency lines, and the lower frequency lines can still be coded as left/right channel or in middle/side stereo. The calculation steps for producing the left and right channel is:

- $is\_ratio = \tan\left(is\_pos_{sb} \cdot \frac{\pi}{12}\right)$
- $L_i = L_i \cdot \frac{is\_ratio}{1 + is\_ratio}$
- $R_i = L_i \cdot \frac{1}{1 + is\_ratio}$

The intensity stereo position, is_possb, is read from the scalefactor of the right channel.
Together these techniques of removing irrelevant difference between channels are called *joint stereo coding*.

3.1.8. Alias reconstruction:

The encoder applies an alias reduction after the subband synthesis, because alias effects are always introduced after a non-ideal bandpass filtering. The filter merges the frequency lines using eight butterfly calculations for each subband.

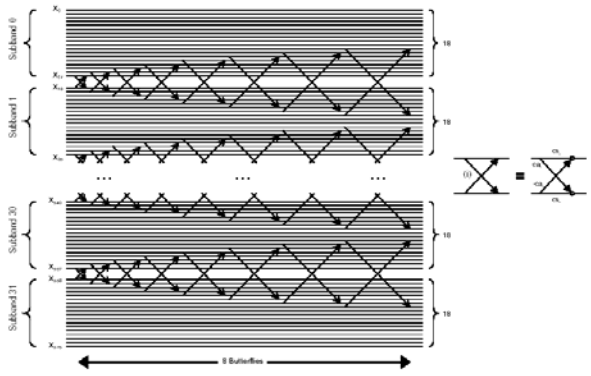

Figure 3.7 Alias reduction diagram

The diagram of the alias reduction can be found in Figure 2.13. The adjacent subbands are weighted in all frequency lines except the two centermost frequencies in every subband. The coefficient in the butterfly filter is defined in the ISO/IEC 11172-3 standard.

3.1.9. Inverse modified DCT (IMDCT):

The *Inverse Modified Discrete Cosine Transform* (IMDCT) used in MP3 is an 18-point DCT that produces 36 output values from 18 input values. It does not produce more information than it is provided with, but it disperses the result from the calculation on a larger number of values. This is why the term *modified* is being used. The DCT is *inverse* because it produces time samples from frequency lines.
The transformation from the frequency domain to the time domain is done in conjunction with the *synthesis polyphase filter bank* described in section 2.3.10. Instead of directly producing time samples, the IMDCT generates *polyphase filter subband samples* from the input frequency lines.These will be later be used for creating time samples.
The 36 calculated values from the IMDCT must be multiplied with a 36-point window before it can be used by the next step in the decoding process. There are four different window types that can be applied to the output samples. The window to use is based on the block type, and the block type can be found in the side information.
The reason for generating 36 output values is that the IMDCT uses a 50% overlap. The lower 18 values are added with the higher 18 values from the previous frame, and used as output. The higher 18 values are then stored and used the same way when the next frame is being decoded.

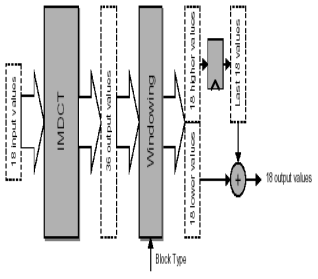

Figure 3.8. IMDCT with windowing and overlap add

A full view of IMDCT including windowing and overlap add can be found in Figure 3.7. The IMDCT calculation is based on the following equation:

$$x_i = \sum_{k=0}^{\frac{n}{2}-1} X_k \cos\left(\frac{\pi}{2n}\left(2i+1+\frac{n}{2}\right)(2k+1)\right) \qquad 0 \le i < n$$

One of four different windows can be applied to the output from the IMDCT. Each input value is scaled with the corresponding value in the window. The window type is based on the behavior of the input sequence and can be a normal, short, start or stop window.

3.1.10. Frequency inversion:

Before proceeding to the synthesis polyphase filter bank, all odd subsamples in all odd subbands must be negated. This is done to compensate for the frequency inversion in the synthesis polyphase filter bank.

3.1.11. Synthesis polyphase filter bank:

The synthesis polyphase filter bank, or the subband synthesis, is the final step in the decoding process. It produces 32 PCM samples at a time using the input samples from the filter bank.

3.2. Software implementation:

The goal has been to create a decoder that can be used by any application on any platform, and the decoding software can execute in any stand-alone application It is always an advantage to be able to compile and run the software in different environments.

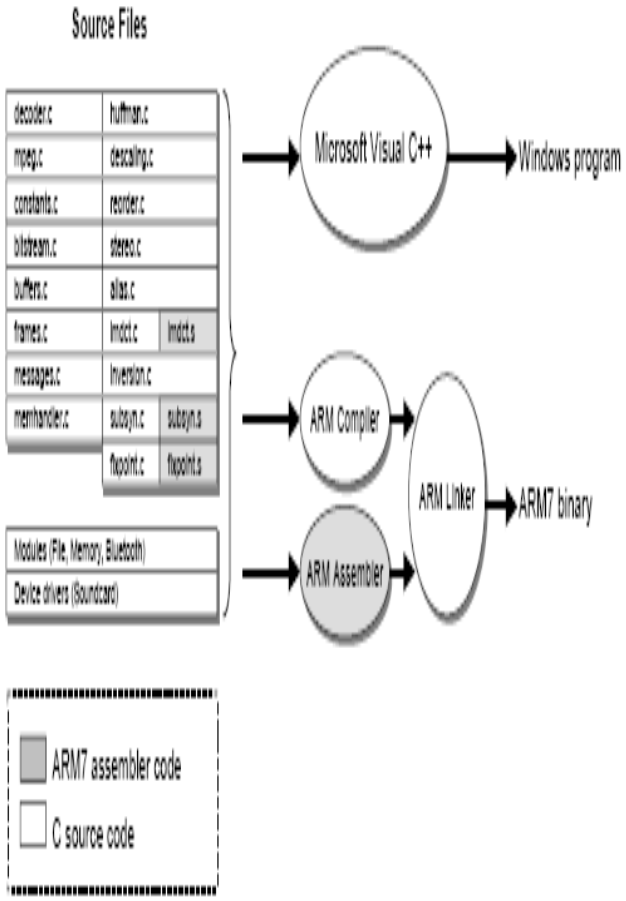

Figure 3.9 Developing of mp3 decoder for arm platform

The decoder software was initially developed using Microsoft Visual C++, writing only ANSI compatible code. Later on, the same code was compiled using a cross compiler from ARM that produces binaries for the ARM7 processors. The ARM Developer Suite TM (ADS) provides the software developer with a complete kit of tools including a GUI, C and C++ compilers and linkers in an integrated environment, which can be used to develop application code for the ARM.

Initially decoder software coded in ANSI C, for this code corresponding optimized assembly code is generated by the arm compiler (provided by ADS).With this decoder implementation the MP3 decoder is consuming around 2269 MIPS(millions of instructions per second, it is measured by profiling,(profile is one tool provided by ADS). it is not at all useful for real time playing .The observed things in this profile is IMDCT and subband synthesis are consuming large amount of time, because these two functions IMDCT and polyphase subband synthesis are DSP processes ,no specific
instructions are available for arm processor, so these two functions are code in handwritten Assembly, with this MP3decoder the processor time consumption is reduced to 32 MIPS

3.3. Experiment:

Development board:

The platform for this master thesis is the AT91SAM9263 Target board. It contains a micro controller, static memory, flash memory and an FPGA1. Expansion ports are available for interfacing with other hardware devices.The software is developed on a workstation and then downloaded and executed on theTarget board. The development board communicates with the workstation using a Interface.i.e.In _Circuit Emulator.
The microcontroller on the Target board is based on a CPU core from ARM, known as ARM7. The ARM7 core is a 32-bit microprocessor using RISC technology with a 3-stage pipeline.

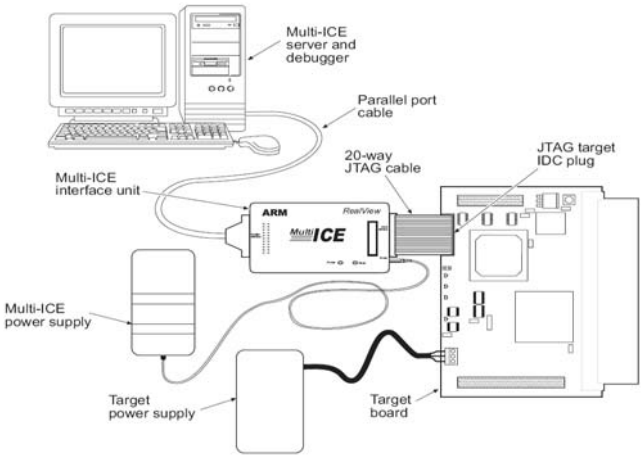

Figure 3.10 MP3 DECODERON ARM SET_UP DIAGRAM

The ARM7 binary mp3 decoder is loaded on to the AT91SAM9263 board, through interface debugger ICE. The input to this MP3 decoder is **.mp3** extension file ,the MP3 decoder will decodes compressed MP3 audio and generates decompressed format of MP3 i.e. PCM format will be stored in the target board memory.

The screen (PC workstation) displays the frame numbers while decoding until it reaches the End of stream. On the completion find the copy of the file with **.pcm** extension which is the decompressed or decoded version of mp3 .This is standard audio format which can be played on any audio device (using cool edit pro tool).

C.1 Target Board:

The target board that is used to run the MP3 decoder is AT91SAM9263. It is Atmel 32- bit microcontroller. The target processor includes ARM9 platforms.

AT91SAM9263 BLOCK DIAGRAM

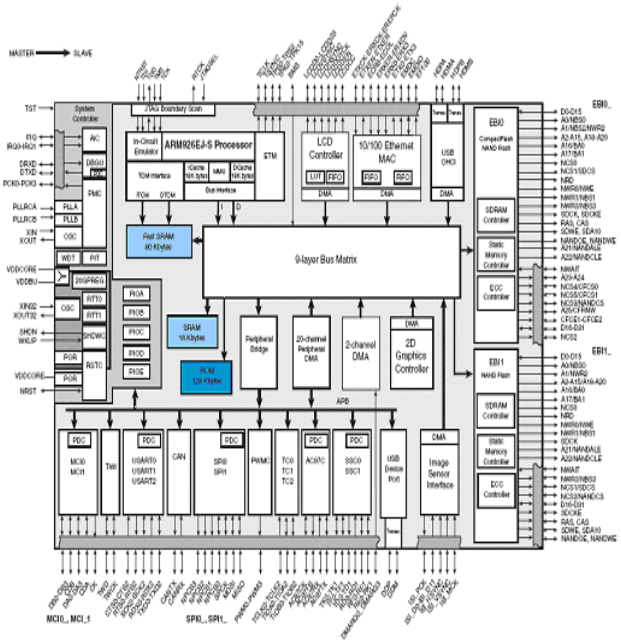

Figure 4. AT91SAM9263 Block Diagram

DESCRIPTION:

The AT91SAM9263 32-bit microcontroller, based on the ARM926-EJ-S processor, is architectured on a 9-layer matrix, allowing a maximum internal bandwidth of nine 32-bit buses.

It also features two independent external memory buses, EBI0 and EBI1, capable of interfacing with a wide range of memory devices and an IDE hard disk. Two external buses prevent bottlenecks, thus guaranteeing maximum performance.

The AT91SAM9263 embeds an LCD Controller supported by 2D Graphics Controller and a 2- channel DMA Controller, and one Image Sensor Interface. It also integrates several standard peripherals, such as USART, SPI, TWI, Timer Counters, PWM Generators, Multimedia Card interface and one CAN Controller. When coupled with an external GPS engine, the AT91SAM9263 provides the ideal solution for navigation systems.

Features:

- Incorporates the ARM926EJ-S™ ARM® Thumb® Processor
  - DSP Instruction Extensions, Jazelle® Technology for Java® Acceleration
  - 16 Kbyte Data Cache, 16 Kbyte Instruction Cache, Write Buffer
  - 220 MIPS at 200 MHz
  - Memory Management Unit
  - EmbeddedICE™, Debug Communication Channel Support
  - Mid-level Implementation Embedded Trace Macrocell™

- Bus Matrix
  - Nine 32-bit-layer Matrix, Allowing a Total of 28.8 Gbps of On-chip Bus Bandwidth
  - Boot Mode Select Option, Remap Command
- Embedded Memories
  - One 128 Kbyte Internal ROM, Single-cycle Access at Maximum Bus Matrix Speed
  - One 80 Kbyte Internal SRAM, Single-cycle Access at Maximum Processor or Bus
  - Matrix Speed
  - One 16 Kbyte Internal SRAM, Single-cycle Access at Maximum Bus Matrix Speed
- Dual External Bus Interface (EBI0 and EBI1)
  - EBI0 Supports SDRAM, Static Memory, ECC-enabled NAND Flash and
  - CompactFlash®
  - EBI1 Supports SDRAM, Static Memory and ECC-enabled NAND Flash
  -
- Image Sensor Interface

  - ITU-R BT. 601/656 External Interface, Programmable Frame Capture Rate
  - 12-bit Data Interface for Support of High Sensibility Sensors
  - SAV and EAV Synchronization, Preview Path with Scaler, YCbCr Format
- USB 2.0 Full Speed (12 Mbits per second) Host Double Port
  - Dual On-chip Transceivers
  - Integrated FIFOs and Dedicated DMA Channels
- USB 2.0 Full Speed (12 Mbits per second) Device Port
  - On-chip Transceiver, 2,432-byte Configurable Integrated DPRAM
- Ethernet MAC 10/100 Base-T
  - Media Independent Interface or Reduced Media Independent Interface
  - 28-byte FIFOs and Dedicated DMA Channels for Receive and Transmit
- Fully-featured System Controller, including
  - Reset Controller, Shutdown Controller
  - Twenty 32-bit Battery Backup Registers for a Total of 80 Bytes
  - Clock Generator and Power Management Controller
  - Advanced Interrupt Controller and Debug Unit
  - Periodic Interval Timer, Watchdog Timer and Double Real-time Timer
- Reset Controller (RSTC)
  - Based on Two Power-on Reset Cells, Reset Source Identification and Reset Output Control
- Shutdown Controller (SHDWC)
  - Programmable Shutdown Pin Control and Wake-up Circuitry
- Clock Generator (CKGR)
  - 32768Hz Low-power Oscillator on Battery Backup Power Supply, Providing a Permanent Slow Clock
  - 3 to 20 MHz On-chip Oscillator and Two Up to 240 MHz PLLs
- Power Management Controller (PMC)
  - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
  - Four Programmable External Clock Signals
- Advanced Interrupt Controller (AIC)
  - Individually Maskable, Eight-level Priority, Vectored Interrupt Sources
  - Two External Interrupt Sources and One Fast Interrupt Source, Spurious Interrupt Protected
- Debug Unit (DBGU)
  - 2-wire UART and Support for Debug Communication Channel, Programmable ICE Access Prevention
- Periodic Interval Timer (PIT)
  - 20-bit Interval Timer plus 12-bit Interval Counter
- Watchdog Timer (WDT)
  - Key-protected, Programmable Only Once, Windowed 16-bit Counter Running at Slow Clock
- Two Real-time Timers (RTT)
  - 32-bit Free-running Backup Counter Running at Slow Clock with 16-bit Prescaler
- Five 32-bit Parallel Input/Output Controllers (PIOA, PIOB, PIOC, PIOD and PIOE)
  - 160 Programmable I/O Lines Multiplexed with Up to Two Peripheral I/Os
  - Input Change Interrupt Capability on Each I/O Line
  - Individually Programmable Open-drain, Pull-up Resistor and Synchronous Output
- One Part 2.0A and Part 2.0B-compliant CAN Controller
  - 16 Fully-programmable Message Object Mailboxes, 16-bit Time Stamp Counter
- Two Multimedia Card Interface (MCI)
  - SDCard/SDIO and MultiMediaCard™ Compliant
  - Automatic Protocol Control and Fast Automatic Data Transfers with PDC
  - Two SDCard Slots Support on eAch Controller
- Two Synchronous Serial Controllers (SSC)
  - Independent Clock and Frame Sync Signals for Each Receiver and Transmitter
  - I²S Analog Interface Support, Time Division Multiplex Support
  - High-speed Continuous Data Stream Capabilities with 32-bit Data Transfer
- One AC97 Controller (AC97C)
  - 6-channel Single AC97 Analog Front End Interface, Slot Assigner
- Three Universal Synchronous/Asynchronous Receiver Transmitters (USART)
  - Individual Baud Rate Generator, IrDA® Infrared Modulation/Demodulation, Manchester Encoding/Decoding

    - Support for ISO7816 T0/T1 Smart Card, Hardware Handshaking, RS485 Support
- Two Master/Slave Serial Peripheral Interface (SPI)
    - 8- to 16-bit Programmable Data Length, Four External Peripheral Chip Selects
    - Synchronous Communications at Up to 90Mbits/sec
- One Three-channel 16-bit Timer/Counters (TC)
    - Three External Clock Inputs, Two Multi-purpose I/O Pins per Channel
    - Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
- One Four-channel 16-bit PWM Controller (PWMC)
- One Two-wire Interface (TWI)
    - Master Mode Support, All Two-wire Atmel® EEPROMs Supported
- IEEE® 1149.1 JTAG Boundary Scan on All Digital Pins
- Required Power Supplies
    - 1.08V to 1.32V for VDDCORE and VDDBU
    - 3.0V to 3.6V for VDDOSC and VDDPLL
    - 2.7V to 3.6V for VDDIOP0 (Peripheral I/Os)
    - 1.65V to 3.6V for VDDIOP1 (Peripheral I/Os)
    - Programmable 1.65V to 1.95V or 3.0V to 3.6V for VDDIOM0/VDDIOM1 (Memory I/Os)
    - periodic interval Timer, Watchdog Timer and Double Real-time Timer
- Reset Controller(RSTC)
    - Based on Two Power-on Reset Cell, Reset source identification and Reset Output control
- Shut down controller (SHDWC)
    - Programmable Shutdown Pin Control and wake-up Circuitry
- Clock Generator (CKGR)
    - 32768Hz Low-power Oscillator on Battery Back-up Power Supply, providing a permanent slow clock
    - 3 to 20 MHz On-chip oscillator and two up to 240 MHz PLLs
- Power Management Controller(PMC)
    - Very Slow Clock Operating Mode, Software Programmable Power Optimization Capabilities
    - Four Programmable External Clock Signals
- Advanced Interrupt Controller(AIC)
    - Individually Mask able, Eight-level Priority, Vectored interrupt Sources
    - Two External interrupt sources and one fast interrupt source, spurious interrupt protected
- Debug Unit (DBGU)
    - 2-wire UART and support for debug communication channel, programmable ICE Access Prevention
- Periodic Interval Timer(PIT)
    - 20-bit interval timer plus 12-bit interval counter
- Watchdog Timer(WDT)
    - Key-protected, programmable only once, windowed 16-bit counter running at slow clock
- Two Real-time timers(RTT)
    - 32-bit free-running backup counter running at slow clock with 16-bit Prescaler
- Five 32-bit parallel Input/output Controller (PIOA,PIOB,PIOC,PIOD and PIOE)
    - 160 Programmable I/O Lines Multiplexed with Up to Two Peripheral I/Os
    - Input Change interrupt Capability on Each I/O line
    - Individually Programmable Open-drain, Pull-up Resistor and Synchronous Output
- One part 2.0A and Part 2.0B-compliant CAN controller
    - 16 Fully-programmable Message Object mailboxes, 16-bit Time Stamp Counter
- Two Multimedia Card Interface(MCI)
    - SDCard/SDIO and MultiMediaCard™ Compliant
    - Automatic Protocol Control and Fast Automatic Data Transfer with PDC
    - Two SDCard Slots Support on each controller
- Two Synchronous Serial Controller(SSC)
    - Independent Clock and Frame Sync Signal for each receiver and transmitter
    - I2S Analog interface support, Time Division Multiplex Support
    - High-speed Continuous data stream capabilities with 32-bit data transfer
- One AC97 Controller (AC97C)
    - 6-channel Single AC97 Analog Front End Interface, Slot Assigner
- Three Universal Synchronous/Asynchronous Receiver Transmitter(UASRT)

  - Individual Baud Rate Generator, IrDA Infrared Modulation/Demodulation, Manchester Encoding/Decoding
- Two Master/Slave Serial Peripheral Interface(SPI)
  - 8 to 16-bit Programmable data Length, Four External Peripheral Chip Selects
  - Synchronous Communication at Up to 90 Mbits/sec
- One Three-channel 16-bit Timer/Counter (TC)
  - Three External Clock inputs, two Multi-purpose I/O Pins per channel
  - Double PWM Generation, Capture/Waveform Mode, Up/Down Capability
- One Four-Channel 16-bit PWM Controller (PWMC)
- One Two-wire Interface(TWI)

4. SIMULATION RESULTS:

SCREEN SHOTS:

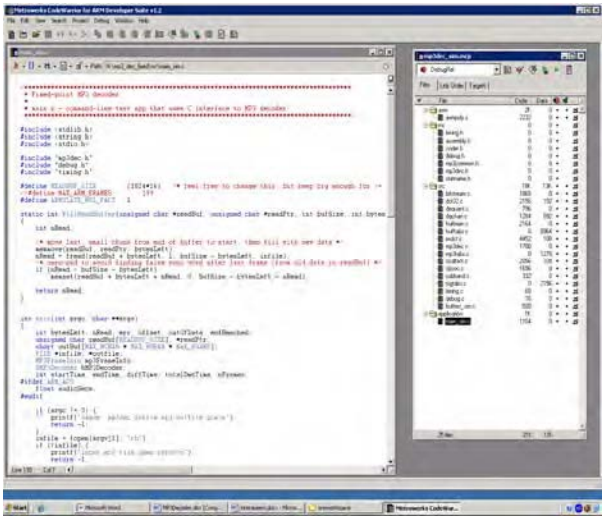

Figure4.1 MP3decoder software (on arm tools i.e.metroworkscode warrior)

- Metroworks code warrior is one of the window of the arm tools on this window MP3 decoder software is weitten in c.

AXD DEBUGGER

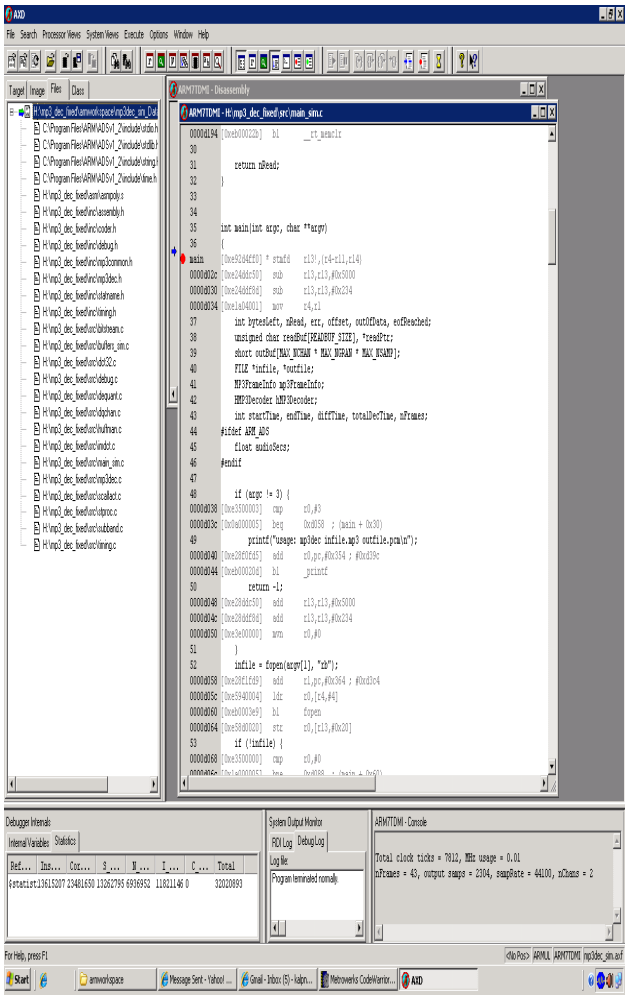

Figure 4.2 Each function in the MP3decoder

- Here every function in MP3 decoder (i.e. Frame synchronizatuin,huffman decoding, requantization. descaling,reording, stereoprocessing, aliasreconstruction, IMDCT, frequency inversion and systhesis polyphase filter bank)

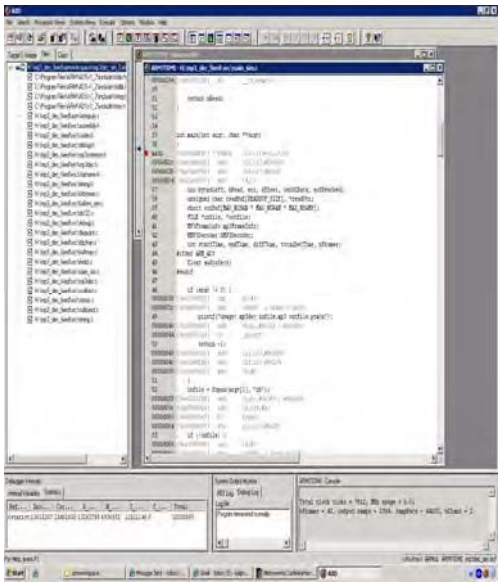

Figure4.3 Generated assembly code for corresponding c code

- Here assembly code is generated for corresponding c code by compiler id provided by ARM tools.

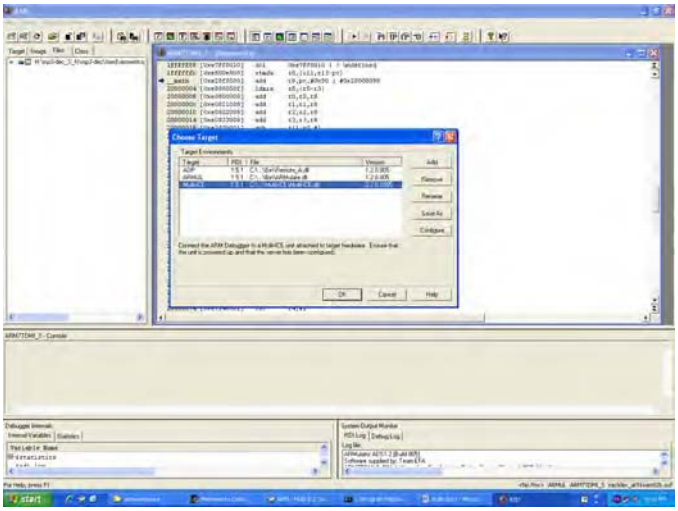

Figure4.4 Enabling the target board

- Here Target board is enabled to download the MP3 decoder software,MP3 audio on to the target board from PC workstation.

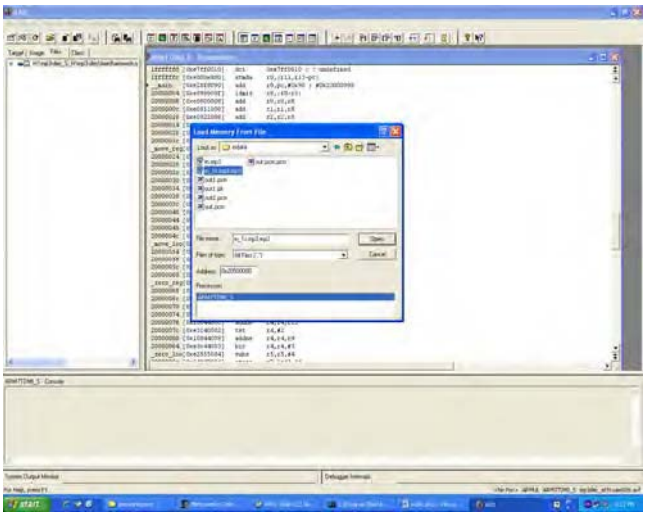

Figure4.5 Loading the **.mp3** input file on the board

- MP3 audio (i.e. **.mp3** file) is downloaded on to the target board memory from PC workstation

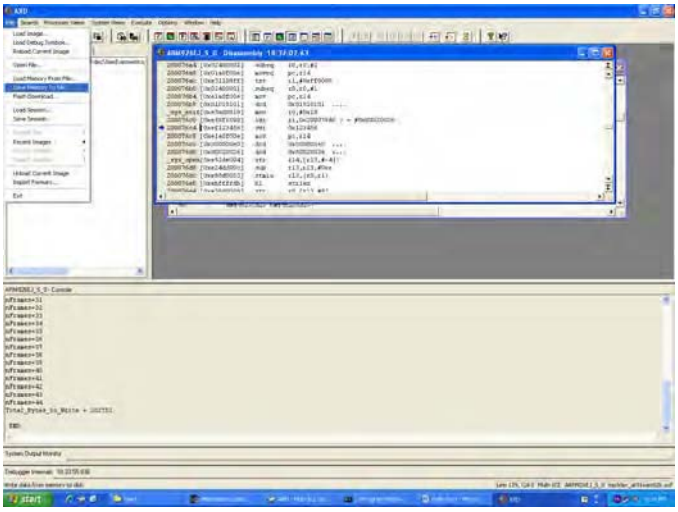

Figure4.6 Saving the .pcm file on to the board

- Here .pcm file(decompressed file) is stored on to the target board memory ,it is generated after decoding the mp3 audio (i.e. .mp3 file)

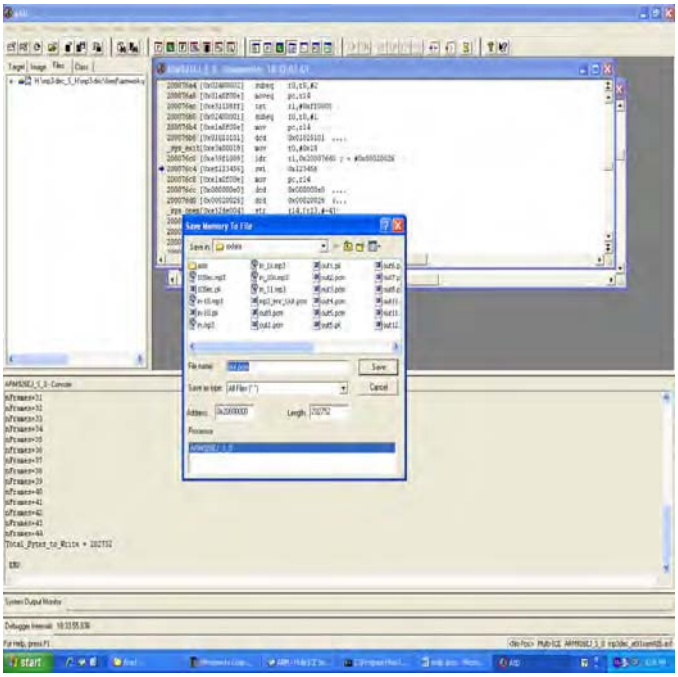

Figure4.7 Output of number of frames in .mp3 file(44 frames in one second audio)

- Here number of frames in mp3 audio are shown these are displayed while MP3 decoder is decoding the mp3 audio(one second audio will takes around 44 frames).

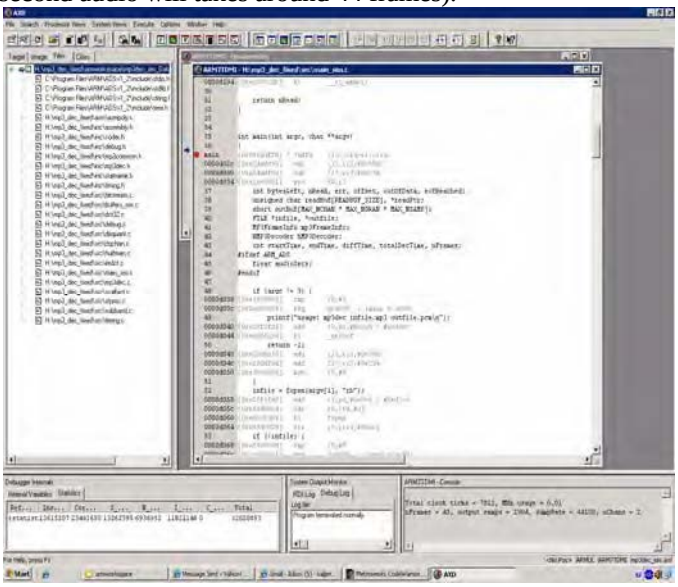

Figure4.9 Mips (32)of fixed point MP3 DECODER

- Debugger Internals is one of the window of the ARM tools it provides lnformation about the no.of cycles taken by the mp3(MIPS are the measure of power consumption)

Figure4.10 Output of MP3 decoder i.e. **.pcm** file on cool edit_pro tool

- Cool edit pro is the Audio tool used to play the audio after decoding the mp3 audio,the decoded audio is played via cool edit pro tool,it also displays strenth(magnitude) of the audio and duration of the audio.

Profiling:

The following information is captured by doing flat profiling using "armprof" tool on fixed point MP3 decoder. The top 10 time consuming functions are considered here.

Name	time%
xmp3_PolyphaseStereo	41.21%
xmp3_FDCT32	15.10%
HybridTransform	13.41%
xmp3_IMDCT	4.67%
DequantBlock	4.52%
idct9	4.42%
DecodeHuffmanPairs	4.19%
xmp3_MidSideProc	2.48%
xmp3_DecodeHuffman	2.40%
FreqInvertRescale	2.10%

Frequency Analysis

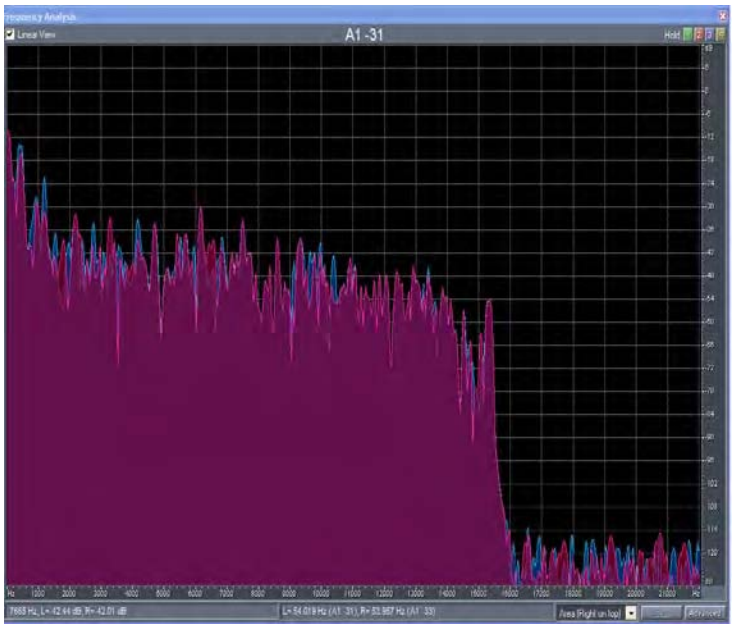

Figure4.11 Frequency Analysis on MP3 decoded file - Left Channel

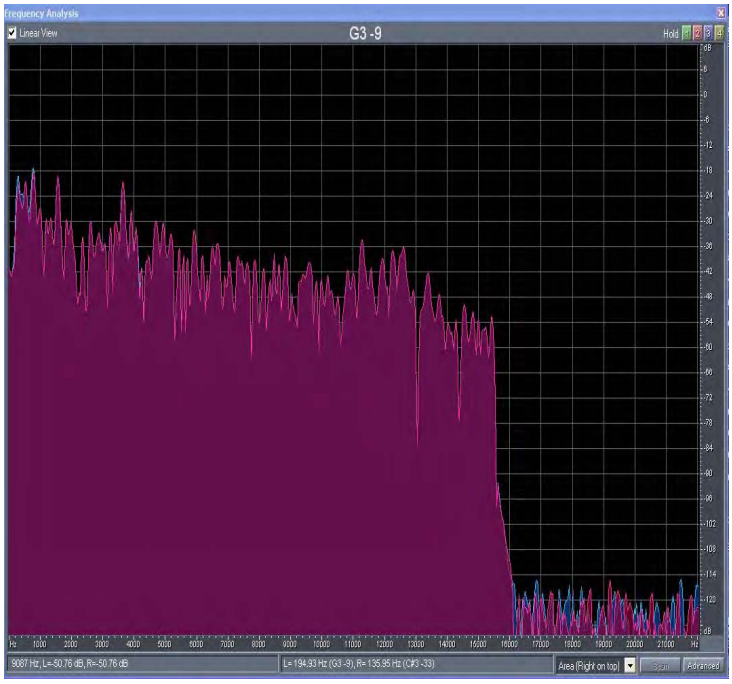

Figure4.12 Frequency Analysis on MP3 decoded file - Right Channel

9. CONCLUSIONS:

This Thesis has provided the reader with an insight to the MPEG-1 Layer III standard and thus given a good preparation for a future encoder/decoder implementation. A decoder would be simpler to implement bearing in mind that it only has to reconstruct the bit stream and does not need to be concerned about psychoacoustics or the quality of the encoded data. Although this paper has given enough information on the frame header and side information to be able to locate the frames, read the parameters and browse through them, more detailed information regarding the sub procedures has to be examined to put this theory into practice. The MPEG-1specification is a good place to start but additional papers will probably be useful since is ambiguous in some parts and lacks details in other parts. The newest version of is recommended where previous type errors have been corrected. This implemented MP3 decoder on ARM7TDMI can play MP3 stereo in real time with low power consumption if we operate any portable device ideally for MP3 playing the battery will work upto 26 hours

10. REFERENCES:

[1] Ted Painter, Andreas Spaniels, "A Review of Algorithms for Perceptual Coding of Digital Audio Signals"
[2] K. Salmons, "Design and Implementation of an MPEG/Audio Layer III Bitstream Processor", Master´s thesis, Aalborg University, Denmark, 1997
[3] International Organization for Standardization webpage, http://www.iso.ch
[4] Scot Hacker, Mp3: The Definitive Guide, O´REILLY 2000
[5] K. Brandenburg, G. Stoll, "ISO-MPEG-1 Audio: A Generic Standard for Coding of High Quality Digital Audio"
[6] MP3´ Tech, http://www.mp3-tech.org/
[7] ID3.org, http://www.id3.org
[8] ISO/IEC 11172-3 "Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s – Part 3"
[9] M. Sieler, R Sperschneider, " MPEG-Layer3 Bitstream Syntax and Decoding"

Authors Biography:

[1] E.KALPANA working as an Assistant Professor in ECE department at Vidya Jyothi Institute of Technology, Hyderabad from 2007.she is having more than 4years experience as an assistant professor.

Her areas of research interests include Waveguides, Transmission lines, Electromagneticwaves, Antennas, Microwave engineering, and Embedded systems.

[2]VARADALA SRIDHAR is from HYDERABAD,ANDHRAPRADESH.Completed M.TECH in ECE with specialization (WIRELESS AND MOBILE COMMUNICATION SYSTEMS) from vardhaman college of engineering affiliated by JNTUH in 2011.he has completed M.Sc (IT)from Nagarjuna University, guntur, AndhraPradesh.and B.TECH in ECE from vidya jyothi institute of technology affiliated by JNTUH in 2007. Currently he is working as an Assistant professor in ECE department at Vidya Jyothi Institute of Technology, Hyderabad from 2010. His areas of research interests include

WirelessandMobilecommunicationssystems,Digitalsignalprocessing,Imageprocessing,Telecommunications,communication systems, Signal processing,Embedded systems.He is Lifetime Membership of ISTE, IETE.

[3] M.REJENDRA PRASAD obtained his B.E and M.E Electronics and communication engineering and digital systems from OSMANIA UNIVERSITY, hyderabad.HE has 11 years of experience in embedded and telecom research and development. He is currently working as anassociateprofessor,ECE, DEPARTMENT, VJIT, HYD.his main interests are embedded systems, wireless protocol, and RTOS.