

國立交通大學

電機與控制工程學系

碩士論文

MPEG-1 LAYER III 音訊編解碼演算法最
佳化及 DSP 晶片實現



MPEG-1 LAYER III AUDIO CODEC
OPTIMIZATION AND IMPLEMENTATION ON A
DSP CHIP

研究生：林煜翔

指導教授：吳炳飛 教授

中華民國九十三年七月

MPEG-1 LAYER III 音訊編解碼演算法最佳化及 DSP 晶片實現

研究生：林煜翔

Student : Yu-Shiang Lin

指導教授：吳炳飛 教授

Advisor : Prof. Bing-Fei Wu



A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control Engineering

July 2004

Hsinchu, Taiwan, Republic of China


中華民國 九十三年 七月

MPEG-1 LAYER III 音訊編解碼演算法最佳化及 DSP 晶片實現

學生：林煜翔

指導教授：吳炳飛 教授

國立交通大學 電機與控制工程學系 碩士班



摘要

這篇論文提出一套 MP3 編解碼的最佳化演算法及有效的 16 位元定點 DSP 實現。在 MP3 編碼最佳化中，我們基於移除計算量龐大的聲響心裡模型，提出一套新的速率控制迴圈演算法，並採用頻寬控制及動態位元分配等。在 MP3 解碼最佳化中，我們提出一套新的解量化方程式實現法，並可適用在定點處理器中；在實現 IMDCT 和子頻帶合成上，也採用了快速演算法。我們將 MP3 編解碼最佳化的演算法實現在一顆 16 位元定點 DSP，ADSP-2181 上，並採用動態定點格式降低定點運算時的失真。實現後的 MP3 編碼器僅需 21.05 MIPS 及 44 千位元組記憶體，而解碼器僅需 18.67 MIPS 及 44.3 千位元組記憶體，相較於其他商業化產品及學術研究，能提供最好的效能。最後，本篇論文還提出一個基於 32 位元 RISC 及 DSP 的雙核心嵌入式系統整合設計。

MPEG-1 LAYER III AUDIO CODEC OPTIMIZATION AND IMPLEMENTATION ON A DSP CHIP

Student : Yu-Shiang Lin

Advisor : Prof. Bing-Fei Wu

Department of Electrical and Control Engineering

National Chiao Tung University

ABSTRACT

This thesis presents the algorithm optimization and efficient 16-bit fixed point DSP implementation of MP3 encoding and decoding algorithms. In the MP3 encoding algorithm, we propose several approaches including the removal of psychoacoustic model, simplified iteration loop, fast rate control loop and applying of bandwidth control and dynamic bit allocation proportional to the energy of granules. In the MP3 decoding algorithm, we propose a fast dequantization method with high SNR in fixed point implementation and apply fast algorithms in IMDCT and subband synthesis. The algorithms are also completely realized on a 16-bit fixed point DSP, ADSP-2181, and the dynamic fixed point format is applied to improve audio quality. The MP3 encoder consumes 21.05 MIPS and 44k bytes memory, and the MP3 decoder consumes 18.67 MIPS and 44.3k bytes memory. Both have superior performance than other commercial products and paper works. Finally, this thesis also presents an integrated design of a dual core embedded system with a 32-bit RISC, Intel[®] StrongARM SA-1110, and ADSP-2181 DSP.

ACKNOWLEDGEMENTS

首先要感謝我的指導教授 吳炳飛教授四年來的指導，從大三的專題指導以來，吳教授給了我許多機會接觸各種研究領域及參加各種比賽，並提供豐沛的研究資源，讓我的研究得以順利進行。

另外要特別感謝已畢業的 錢昱璋、許子偉及 魏宏宇學長和 張芷燕學姐在我剛進入實驗室時，給予熱心的指導，奠定我在音效壓縮理論與實作的基礎。還要感謝 呂紹麒與 鄭光輝學長帶領我認識嵌入式系統。

顏志旭學長給予許多寶貴的意見，並指導我研究及分析的方法。還有一起做研究的 黃榮煌同學及進行音質測試的 CSSP 實驗室伙伴們，感謝你們的全力協助，我才能完成這篇論文。還有一同參加比賽的學長姐、實驗室同學及政大伙伴們，大家在比賽過程中的全力參與，讓我們得到的獎項更有意義。

另外要十分感謝我的家人，在升學的過程中提供我無憂無慮的環境，並且完全支持我，有你們的支持我才能順利地從研究所畢業。

最後要感謝我的女朋友 郭小姐，在這六年的求學生涯中，與我分享許多苦與樂，並容忍我長時間待在實驗室做研究。

謹以本論文

獻給最親愛的家人及所有支持關愛我的人

AWARDS



本研究在民國九十一年參加旺宏金矽獎第二屆半導體設計與應用大賽，並獲得應用組一獎，得獎作品為「MP3/CD-ROM Recorder System」，與賽成員尚包括許子偉、張芷燕及魏宏宇同學。



本研究在民國九十二年參加旺宏金矽獎第三屆半導體設計與應用大賽，並獲得應用組二獎，得獎作品為「Multimedia Box」，與賽成員尚包括顏志旭、王坤卿、

魏宏宇及鄭光輝同學。



本研究在民國九十二年參加由中華民國科管學會舉辦的第七屆學生創新獎競賽，並獲得第一名，得獎作品為「向下相容的 MP3 音樂安全機制」，與賽成員尚包括顏志旭、黃榮煌及林映伶同學。

CONTENTS

ABTRACT (CHINESE)	i
ABTRACT (ENGLISH)	ii
ACKNOWLEDGEMENTS	iii
AWARDS	iv
CONTENTS	vi
LIST OF FIGURES	ix
LIST OF TABLES	xi
CHAPTER 1. INTRODUCTION	1
1.1 MPEG/Audio Compression	1
1.2 Motivations	2
1.3 The Overview of The Proposed Method and Contributions	3
1.4 The Experimental Results and Potential Applications	3
1.5 Content and Organization	4
CHAPTER 2. ENCODER OPTIMIZATION	6
2.1 Encoding Overview and Complexity Analysis	6
2.1.1 Psychoacoustic model II	7
2.1.2 Time to frequency mapping transform	9
2.1.3 Iteration loop.....	12
2.1.4 Bitstream formatting	13
2.2 Simplified PAM-II	14
2.2.1 Distortion control loop analysis.....	15
2.2.2 Removal of window switching	18
2.3 Fast rate control loop	21

2.3.1	Non-uniform quantization.....	22
2.3.2	Dynamic bit allocation proportional to the energy of granules	26
2.3.3	Precise initialization of the quantization parameter.....	27
2.3.4	Fast search of the optimal quantizer parameter	31
CHAPTER 3. DECODER OPTIMIZATION		36
3.1	Decoding Overview and Complexity Analysis.....	36
3.2	Dequantization	38
3.3	IMDCT and Subband Synthesis	42
CHAPTER 4. DSP IMPLEMENTATION		43
4.1	Target DSP Architecture.....	43
4.2	Data precision optimization in the proposed MP3 encoder	45
4.3	Data precision optimization in the proposed MP3 decoder	50
CHAPTER 5. EXPERIMENTAL RESULTS		53
CHAPTER 6. DUAL CORE EMBEDDED SYSTEM.....		58
6.1	System Overview.....	58
6.2	Hardware Platform.....	59
6.2.1	Host system – AdvanTech PCM-7130 SBC	59
6.2.2	DSP system – ADI ADSP-2181 EZ-LAB.....	61
6.2.3	Design of hardware adapter	62
6.3	Firmware Design.....	66
6.3.1	Linux Character Device Driver.....	66
6.3.2	DSP BIOS	69
CHAPTER 7. CONCLUSIONS AND FUTURE WORKS		71
7.1	Conclusions.....	71
7.2	Future Works	72

REFERENCE.....74

APPENDIX.....78



LIST OF FIGURES

Figure 1.	MPEG/audio encoding process.....	7
Figure 2.	The absolute threshold of hearing.....	8
Figure 3.	Frequency masking effect combined with ATH.....	8
Figure 4.	Temporal masking effect.....	9
Figure 5.	Hybrid transform for time to frequency mapping.....	10
Figure 6.	The 32-channel analysis polyphase filterbank.....	10
Figure 7.	The coefficient of low-pass filter, $h[n]$	11
Figure 8.	The four types of MDCT window and the arrangement.....	12
Figure 9.	Distortion control in the iteration loops	15
Figure 10.	Noise analysis before distortion control	16
Figure 11.	Coefficients of bandwidth controller (sampling rate is 44100 Hz).....	17
Figure 12.	Time domain waveforms from using window switching or not.....	20
Figure 13.	Rate control in iteration loops.....	21
Figure 14.	The new rate control algorithm.....	22
Figure 15.	The error of $ x_{f,g}(i) ^{0.75}$ approximation	25
Figure 16.	The histogram to difference between initial and final value of $\Delta_{f,g}$	30
Figure 17.	The adaptive approach to iterative search optimum parameter	31
Figure 18.	Pseudo code of iteration loops (a) ISO method (b) Proposed method.....	34
Figure 19.	MPEG/Audio Layer III decoding block diagram	36
Figure 20.	Bitstream decoding	37
Figure 21.	Frequency to time mapping	37
Figure 22.	The implementation of $y_{f,g}^{\frac{1}{3}}(i)$	39

Figure 23.	The error to real output ratio of $y_{f,g}^{\frac{4}{3}}(i)$ approximation.....	41
Figure 24.	The error to real output ratio of $y_{f,g}^{\frac{4}{3}}(i)$ fixed point approximation	42
Figure 25.	The ADSP-2181 DSP core and peripheral integration	43
Figure 26.	Double precision multiplication, $R(32\text{-bit}) = X(32\text{-bit}) \times Y(16\text{-bit})$	46
Figure 27.	Data precision between each stage in proposed MP3 encoder	47
Figure 28.	Data precision between each stage in proposed MP3 decoder	50
Figure 29.	Different format between subbands and the modified IMDCT	51
Figure 30.	The dual core embedded system.....	59
Figure 31.	PCM-7130 SBC [15]	61
Figure 32.	ADI ADSP-2181 EZ-LAB evaluation board	62
Figure 33.	Functional diagram of hardware adapter	62
Figure 34.	General IDMA transfer protocol [17]	64
Figure 35.	Port access timing	65
Figure 36.	The hierarchical view of software, firmware and hardware layer	66
Figure 37.	The firmware block diagram.....	69

LIST OF TABLES

Table 1.	Predicted complexity to implement MPEG/audio encoder [4].....	14
Table 2.	The number of DSP instruction cycles in calculation of two regions.....	25
Table 3.	Symbols descriptions of Figure 17	32
Table 4.	The average number of inner iteration.....	35
Table 5.	The implementation result and comparisons with commercial products ..	54
Table 6.	The comparison of peak consumed MIPS in different MP3 encoder.....	55
Table 7.	The comparison of peak consumed MIPS in different MP3 decoder.....	55
Table 8.	Test audio samples	56
Table 9.	The subjective evaluation results (1)	56
Table 10.	The subjective evaluation results (2)	57
Table 11.	The subjective evaluation results (3)	57
Table 12.	Host port pins.....	63
Table 13.	ADSP-2181 IDMA port pins	63

CHAPTER 1. INTRODUCTION

1.1 MPEG/Audio Compression

Today the digital audio compression has been applied in various current multimedia applications, like network multimedia streaming, online music store, DAB (Digital Audio Broadcasting), digital television and portable devices (pen drive, walkman, voice recorder, cellular phone and etc.). The MPEG/audio compression is the most popular international standard for digital compression of high-fidelity audio.

The state-of-the-art algorithms for audio compression, such as MPEG and WMA, transform the audio signal for de-correlation and quantize the transformed coefficient according to the perceptual property determined by the psychoacoustic model (PAM) [1]. In this approach, the limitation of human hearing are exploited to remove the inaudible components of audio signals to achieve a high compression ratio.

MPEG/audio offers a choice of three distinct compression layers [2]. This provides a wide range of the trade-off between the codec complexity and the compressed audio quality. Layer I forms the basic algorithms and is suitable for the

bit rate above 128 Kbps per channel. Layer II targets the bit rates around 128 kbps per channel. Possible applications include the audio coding for DAB and the storage of synchronized video-and-audio sequences on CD-ROM. Layer III is the most complex but offers the best audio quality, particularly for the bit rate around 64 kbps per channel. This layer suits the audio transmission over ISDN and the multimedia application on portable devices. Which layer will be employed for an application is determined by the computational complexity and the performance requirement [3].

1.2 Motivations

MPEG/audio Layer III, also referred as MP3, is the most popular digital audio format on Internet now. And with the help of Internet, MP3 has also gained popularity as a portable solid-state audio format. Recently, various kinds of devices that support MP3 application have come out in the consumer market. However, most of all have “decoding-only” features. Few of them support MP3 encoding with high quality. This is solely because MP3 encoding algorithm often consumes too much computational resources to implement on the system powered by batteries.

A high quality MPEG/audio Layer III encoding and decoding algorithms, which are optimized for 16-bit fixed point arithmetic, and a real-time implementation on a low-cost 16-bit fixed-point DSP are proposed in this thesis. ADI ADSP-2181 is chosen as the target DSP.

The prototype design of a dual core embedded system is also presented in this thesis. The work is done by integrating the proposed MP3 codec implementation on ADSP-2181 DSP with a 32-bit RISC, Intel[®] StrongARM SA-1110 CPU, on an existing embedded system, AdvanTech PCM-7130 SBC.

1.3 The Overview of The Proposed Method and Contributions

In this thesis, we propose several fast algorithms for MP3 encoding and decoding. In the MP3 encoding algorithm, the psychoacoustic model (PAM), the most computationally complex part of the entire MP3 encoding algorithm, is removed based on several experimental results, and the PAM-based distortion control loop is also simplified. Some techniques including bandwidth control and dynamic bit allocation proportional to the energy of granules are added to improve the audio quality.

Furthermore, a fast rate control loop algorithm is proposed to reduce the complexity of non-uniform quantizer and the number of iterations. The complexity of non-uniform quantizer is reduced by moving the time-consuming operation outside the iteration and by applying piecewise linear approximation in the non-uniform quantization. Thus the quantizer is divided into two parts and consumes less than 10 and 4 DSP instructions outside and inside the iteration respectively. The number of iterations is reduced by the precise initialization and the fast iterative search of the non-uniform quantizer parameter. Thus the average number of iterations is only 1.8 while the original method takes more than 45 iterations in average.

In the MP3 decoding algorithm, the dequantization operation is implemented by applying piecewise linear approximation and Newton's method for root-finding to achieve higher SNR. And we adopt Lee's fast DCT/IDCT algorithm to realize the IMDCT and the matrixing operation in the synthesis filterbank.

1.4 The Experimental Results and Potential Applications

The results of the proposed MP3 codec optimization and implementation are also analyzed. The MP3 encoder consumes 21.05 MIPS and 44k bytes memory and

the MP3 decoder consumes 18.67 MIPS and 44.3k bytes memory. Both have superior performance than other commercial products and paper implementations.

The superior performance in MP3 codec implementation mainly brings two areas of potential application.

- The low requirements of MIPS and memory are suitable for the system powered by battery, like pen drive, walkman, voice recorder, cellular phone and etc. Thus these devices can support both MP3 encoding and decoding.
- The system integration part gives the probability of taking the low cost ADSP-2181 as an audio coprocessor in a large system. By applying the firmware loading protocol proposed in this thesis, the system can support not only MP3 but also more audio application. The innovative feature is suitable for many products nowadays like PVR/ DVR, DVD player/ recorder, IP phone, digital broadcasting system and etc.

1.5 Content and Organization

This thesis contains seven chapters:

- Chapter 1 introduces the digital audio compression algorithms and the motivation, overview and contribution of this thesis.
- Chapter 2 introduces the MPEG/audio Layer III encoding algorithm and brings the proposed optimization. The proposed methods are mainly focused on minimizing the complexity of the PAM-based bit allocation process and improving the coding efficiency. Based on a series of experiments and analysis, the PAM is simplified, and a new fast bit allocation algorithm is developed.
- Chapter 3 introduces the MPEG/audio Layer III decoding algorithm and brings the proposed optimization. The proposed methods are

focused on minimizing the complexity of the dequantization and the filterbank.

- Chapter 4 introduces the 16-bit fixed-point DSP, ADSP-2181 and brings the MP3 codec DSP implementation of proposed methods.
- Chapter 5 presents the experimental results and comparisons with other methods.
- Chapter 6 introduces an application example, a dual core embedded system architecting by Intel® StrongARM MPU and ADI ADSP-2181 DSP, and brings the firmware design.
- Chapter 7 brings the conclusions and future works.
- Appendix contains the pictures of the whole system and sub-systems.



CHAPTER 2. ENCODER OPTIMIZATION

2.1 Encoding Overview and Complexity Analysis

Figure 1 shows the block diagram of the MPEG/audio Layer III encoding process. The 1152 consecutive PCM samples are grouping together and called one audio frame. The time to frequency mapping transforms the audio input into the spectral lines frame by frame.

Then these spectral components are divided into several scalefactor bands according to the critical-band rate. The audio input simultaneously passes through the PAM-II, psychoacoustic model II, that determines the ratio of the signal energy to the masking threshold for each scalefactor band.

To achieve the bit rate constraint, the rate controller varies the quantizer in an orderly way, quantizes the spectral values and counts the number of Huffman code bits required to code the quantized values. The quantizer in MP3 is non-uniform so that the quantization noise depends on the quantized value instead of the quantization parameters like the general uniform quantizers. Huffman coding is chosen as the lossless coding tool while the Huffman tables are pre-defined and have been statistically analyzed [5]. The distortion controller adapts the scalefactors to control the quality when the quantization noise exceeds the masking threshold.

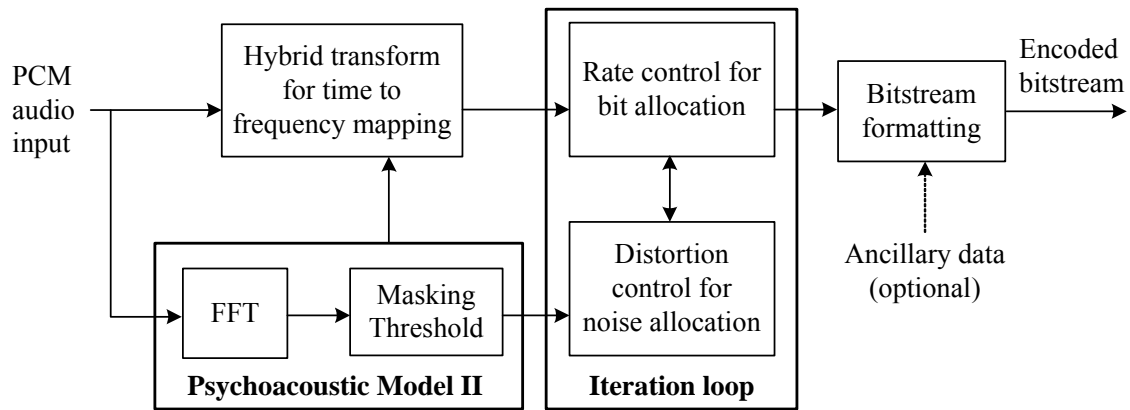


Figure 1. MPEG/audio encoding process

The functionality of each block will be described in the following subsections.

2.1.1 Psychoacoustic model II

The psychoacoustic model, a model of the human auditory perception, supplies the non-uniform quantization block with the information on how to quantized and scaled based on their perceptual relevance. The relevance is denoted as the ability to mask other signals (maskee) for a signal (masker). Usually in MPEG/audio coding, the maskee indicates the noise from the non-uniform quantization of transformed coefficients.

This masking is a perceptual property of the human auditory system that occurs when the presence of strong audio signal make a temporal or spectral neighborhood of weaker audio signal imperceptible. Three types of auditory masking effects are described below:

- The absolute threshold of hearing, ATH: It is characterized by the minimum intensity of a pure tone that the ear can hear in a noiseless environment. This threshold is frequency dependent and typically shows a minimum (indicating the maximum sensitivity of ear) at frequencies between 1kHz to 5kHz. A typical curve of ATH is shown in Figure 2.

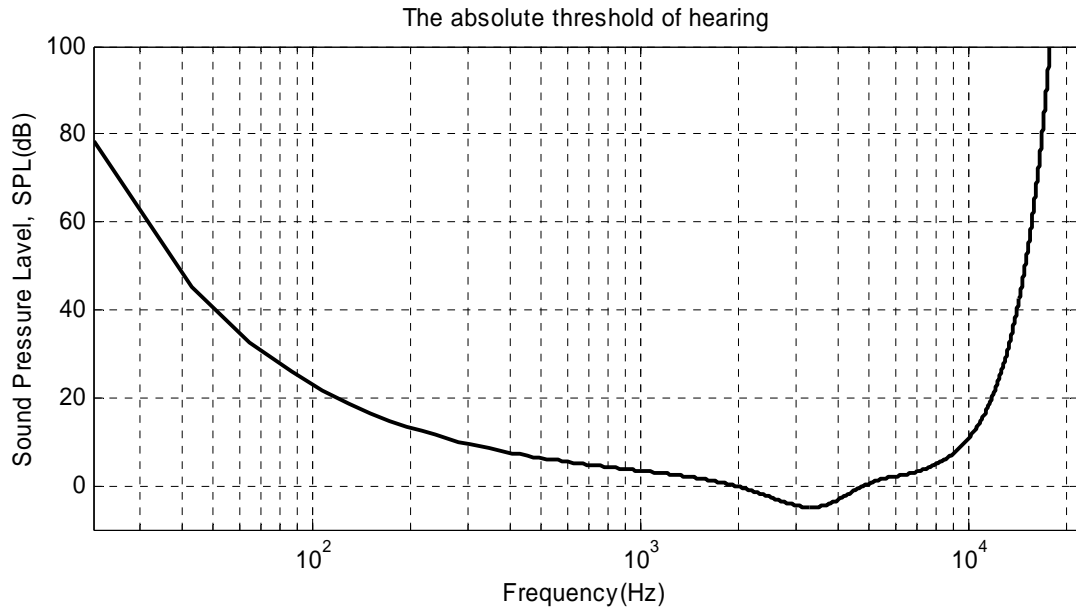


Figure 2. The absolute threshold of hearing

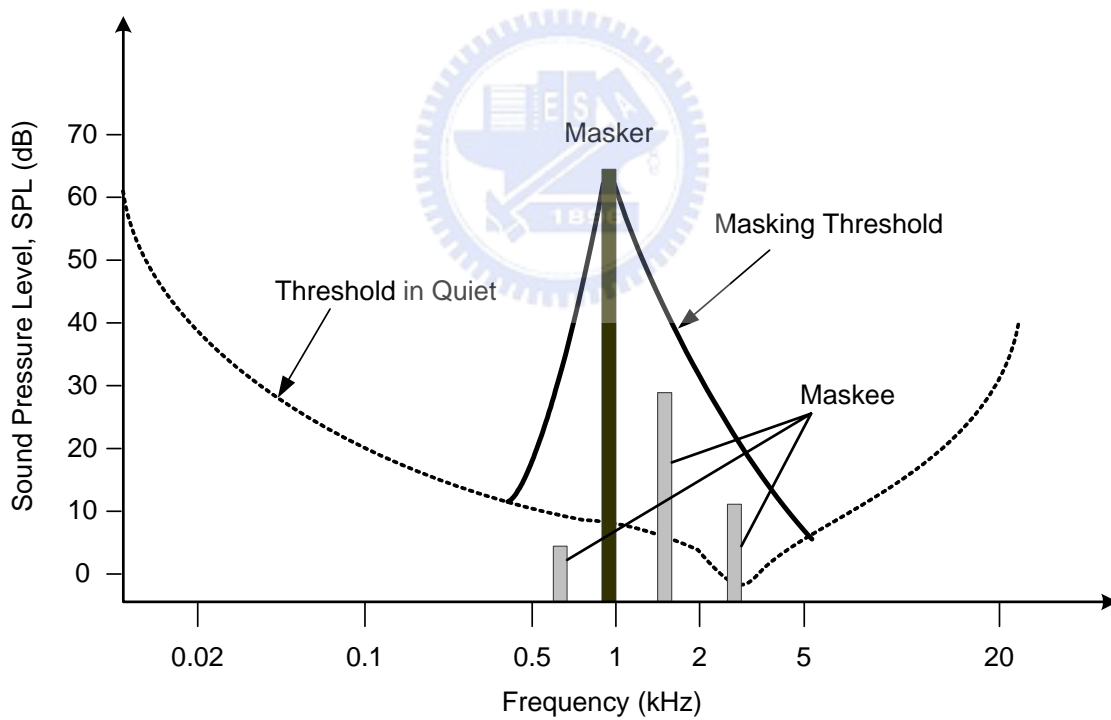


Figure 3. Frequency masking effect combined with ATH

- The frequency masking: it, also called simultaneous masking, is a frequency domain phenomenon where a weaker signal (maskee) can't be perceptible by a simultaneously occurring stronger signal (masker) as

long as they are close enough to each other in frequency. The masking threshold is measured when any signal below is imperceptible and depends on the sound pressure level and the frequency of the masker. As shown in Figure 3, the complete masking threshold is combined with the masking threshold of the masker and the absolute threshold of hearing.

- The temporal masking: It is a phenomenon that relatively loud sounds in an audio signal, such as a loud trumpet's note, will tend to overpower other sounds that occur just before and just after it as shown in Figure 4.

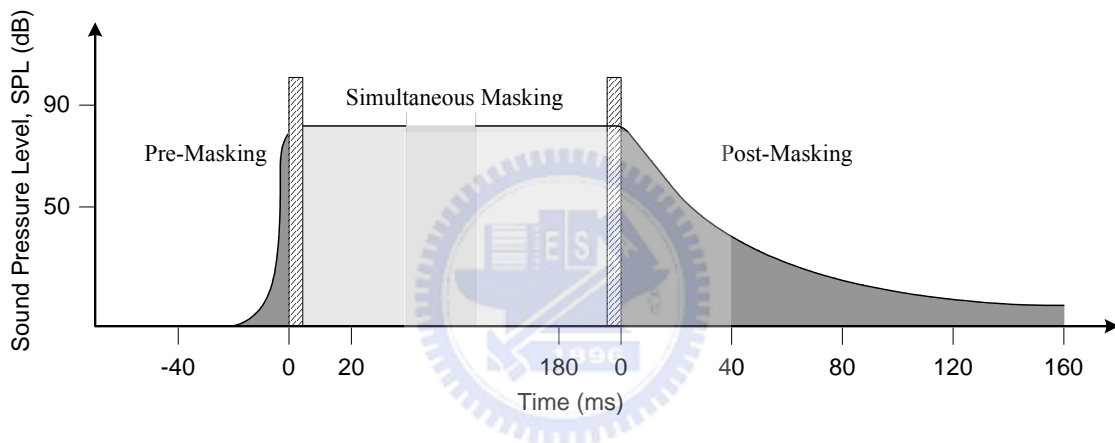


Figure 4. Temporal masking effect

2.1.2 Time to frequency mapping transform

MP3 algorithm uses a hybrid transform to perform time to frequency mapping. As shown in Figure 5, the hybrid transform includes a 32-channel analysis polyphase filterbank, also called subband analysis, and an MDCT filterbank.

Before passing the frequency lines (transformed coefficients) into next stage of the encoding process, a reduction of alias is introduced here in order to reduce amount of information for transmission.

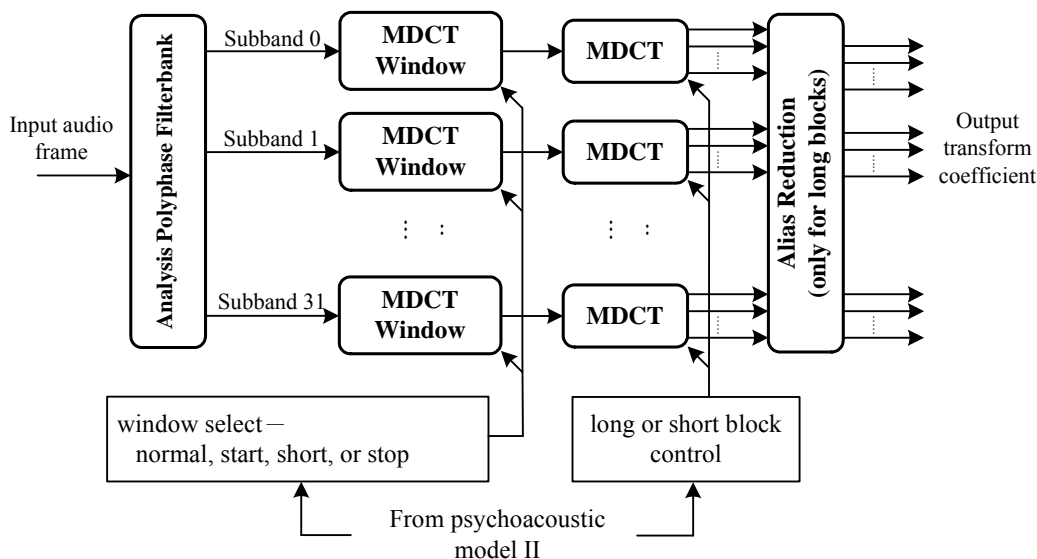


Figure 5. Hybrid transform for time to frequency mapping

Figure 6 is the function diagram of 32-channel analysis polyphase filterbank. It is composed of 32 band-pass filters. The band-pass filter, $H_i(n)$ is generated by modulating the low-pass filter, $h(n)$, to the i^{th} subband as (1). The coefficient of $h(n)$ is shown in Figure 7,

$$H_i(n) = h(n) \cdot \cos\left(\frac{\pi \cdot (2 \cdot i + 1) \cdot (n - 16)}{64}\right), \quad \text{where } n = 0 \sim 511. \quad (1)$$

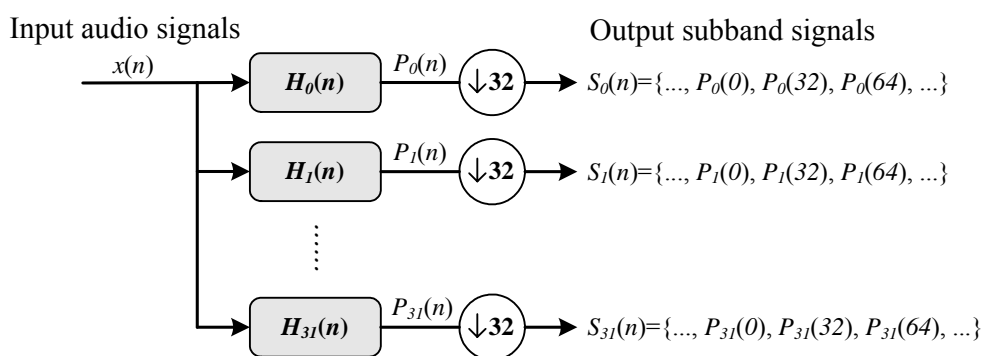


Figure 6. The 32-channel analysis polyphase filterbank

The 32 consecutive audio signals are simultaneously passed into the 32 band-pass filters. The filtering (with 480 overlapped inputs) outputs are

down-sampled and the output subband signal are then produced.

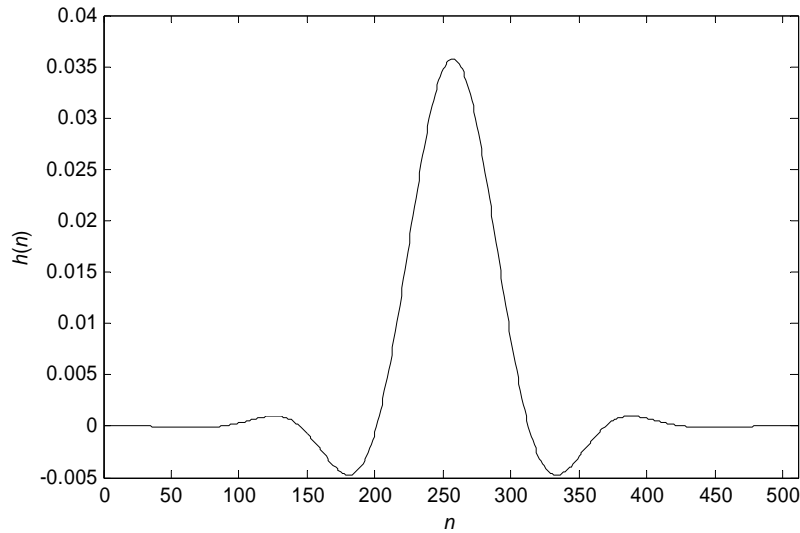


Figure 7. The coefficient of low-pass filter, $h[n]$

The MDCT (Modified Discrete Cosine Transform) performs finer resolution of the 32 subband outputs from the analysis polyphase filterbank as shown in Figure 5. First the subband output passes windowing operation. MDCT uses four types of window as shown in Figure 8 (a) to (d). MP3 specifies two different MDCT block lengths: long block of 18 samples and short block of 6 samples. The normal, start and stop windows are employed in the granule denoted as long block. And the short window is employed in the granule denoted as short block. As shown in Figure 8 (e), each window is 50% overlapped with neighborhood window. So the window size is 36 and 18 respectively.

The start and stop windows are the so-called adaptive windows. The start window provides adaptation from normal window to short window and the stop window provides adaptation from short window to normal window.

Which window is employed is determined by PAM-II. In general, the long

block length provides better frequency resolution (less block effect) with stationary characteristic, and the short block length provides better time resolution with transient.

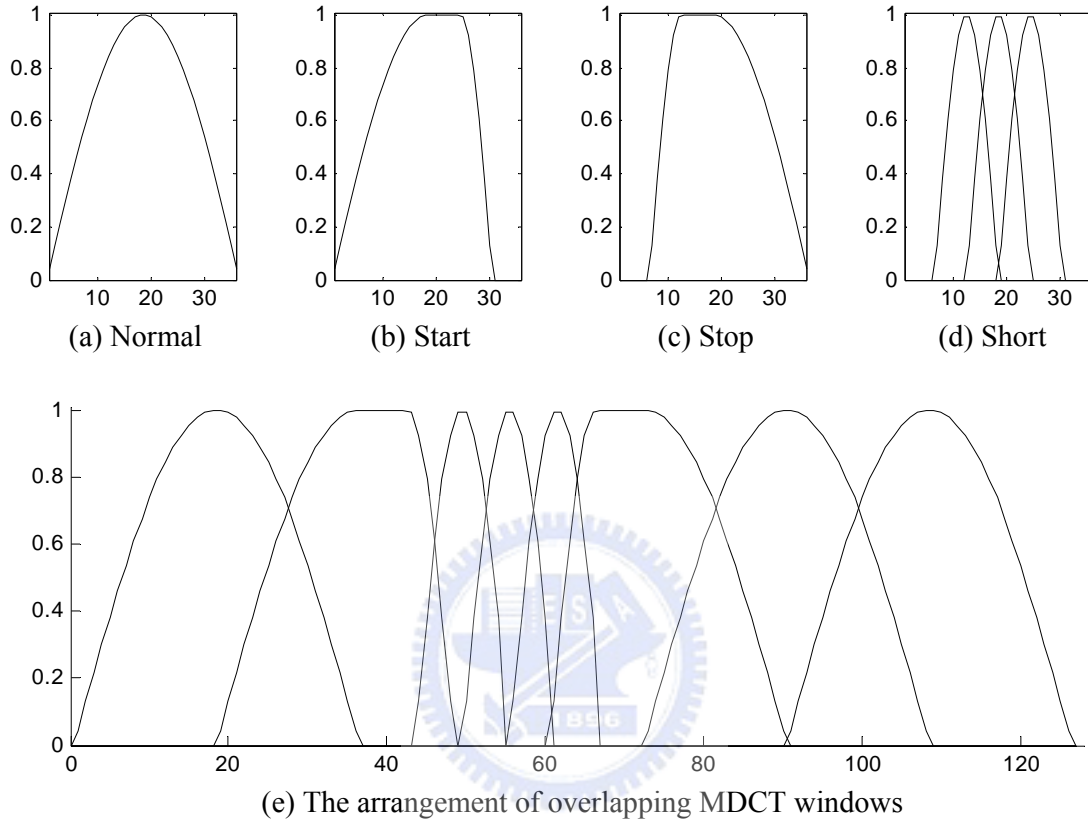


Figure 8. The four types of MDCT window and the arrangement

The formula of MDCT is shown in (2),

$$x(i) = \sum_{k=0}^{n-1} z(k) \cdot \cos\left(\frac{\pi}{2 \cdot n} \cdot \left(2 \cdot k + 1 + \frac{n}{2}\right) \cdot (2 \cdot i + 1)\right), \text{ where } i = 0 \sim \frac{n}{2} - 1. \quad (2)$$

The n is 36 for long block and 12 for short block.

2.1.3 Iteration loop

The iteration loop plays a important role of performing “quantization” and “Huffman coding” to achieve a high compression ratio. This block outputs the coded data satisfying human auditory system and the correlative side

information.

The iteration loop allocates the bits and the allowable noise to each scalefactor band from two main modules: outer and inner iteration loop. The outer iteration loop, also called distortion control loop, controls the quantization noise produced by the non-uniform quantization within the inner iteration loop. The scalefactor of the scalefactor band is adjusted to reduce the quantization noise if the quantization noise is found to exceed the masking threshold obtained from PAM-II. The outer loop is executed until the actual noise is below masking threshold in each scalefactor band.

The inner iteration loop, also called rate control loop, does the actual quantization. The quantized coefficients are then Huffman coded, and the number of coded bits is counted. If Huffman coder demands bits more than the frame can supply, the quantizer parameter needs to be adjusted. The inner iteration loop is repeated with different quantizer parameters until the demanding bits of Huffman coder is small enough.

The Huffman coding algorithm is based on 32 static Huffman tables, provides lossless compression and thereby reduces the amount data to be transmitted without loss of the quality.

2.1.4 Bitstream formatting

This block produces the MPEG/audio Layer III compliant bitstream. The Huffman coded frequency lines, side information and frame header are assembled to form the bitstream. Ancillary data not necessarily related to the audio frame can be inserted into the coded bitstream.

Table 1 summarizes the complexity of MP3 encoding algorithm in DSP MIPS. According to the analysis, PAM and iteration loops are two of the most time-critical processes. PAM-II normally requires transcendental computations

such as logarithm, exponential and power, which are often computationally demanding.

Table 1. Predicted complexity to implement MPEG/audio encoder [4]

MP3 Encoder	MIPS
Hybrid transform	25
PAM-II	90
Iteration loop	70
Etc.	5
Total	190

Another computational demanding task is the iteration loop, also called bit or noise allocation process. The process finds the optimal quantization parameters and scalefactors to obtain the best audio quality in a limited bit resource. Because the quantizer is non-uniform, MP3 adapts an iterative approach to evaluate the parameters. Thus it is based on analysis-by-synthesis scheme. The experimental result shows that the number of iterations per audio granule reached up to 50. It should be also mentioned that the number of iterations depends on the characteristic of input signal and the execution cycles are also varied in each frame.

2.2 Simplified PAM-II

As shown in Table 1, the traditional MP3 encoding algorithm consumes too much MIPS and is hard to be implemented on power-limited devices. Since the complexity analysis shows that both of the most computationally demanding processes are related to ISO PAM-II, we first consider the possibility of encoding without ISO PAM-II [4].

2.2.1 Distortion control loop analysis

Figure 9 shows the traditional iterative approach to implement distortion control. After the rate control loop quantizes the spectral lines, the distortion control loop first reconstructs the spectral by inverse quantization of the quantized value, and then we can evaluate the distortion of the quantization works. Then if the distortion exceeds the masking threshold in the scalefactor band, we can amplify the original signal, and then the masking threshold is also amplified. The pre-emphasis process turns the pre-emphasis flag on and amplifies the whole spectral by pre-defined factor if all of the upper four scalefactor bands have unmasked distortion.

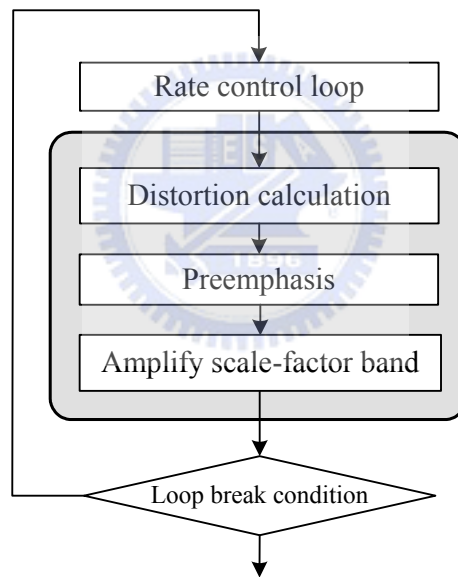
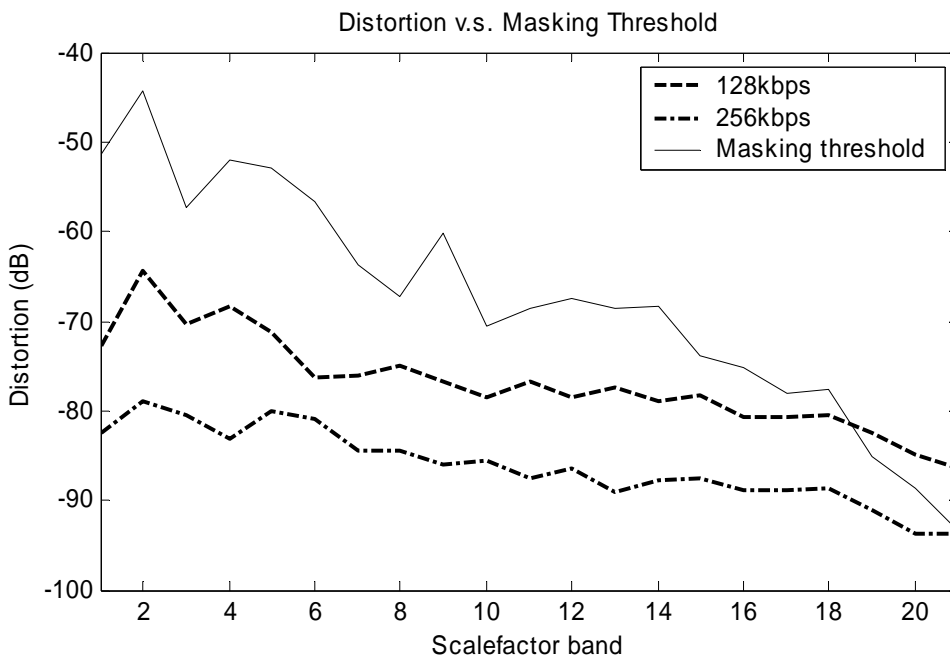
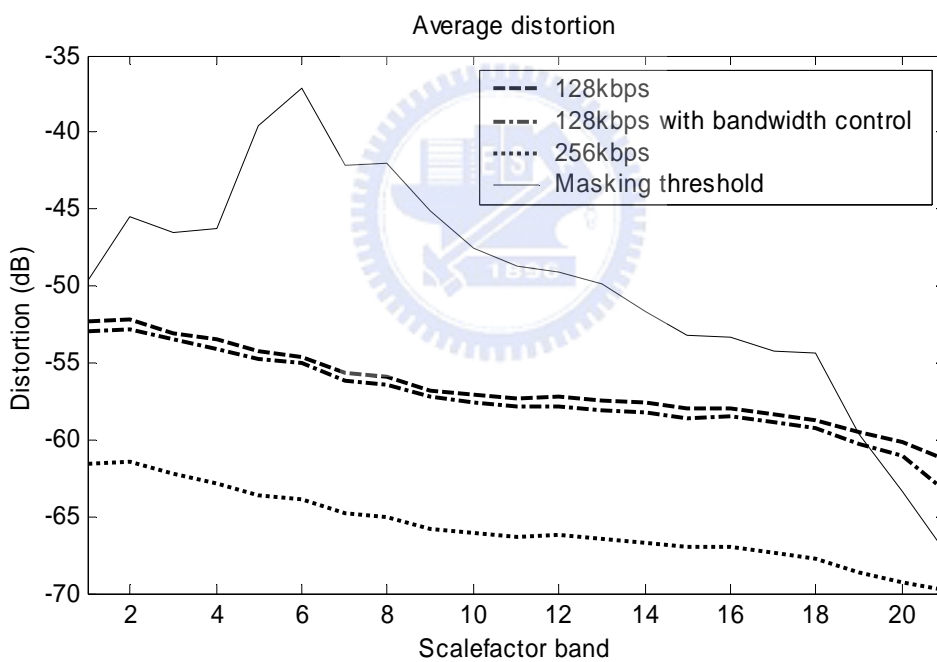


Figure 9. Distortion control in the iteration loops

The motivation of PAM-II simplification is that PAM-II is ineffective over a certain threshold of bit rate [4]. Figure 10 shows the masking threshold and the quantization noise before applying distortion control in each scalefactor band. The signal was encoded at 128kbps and 256kbps stereo with 44.1kHz sampling rate.



(a) Sample granule



(b) Average distortion

Figure 10. Noise analysis before distortion control

As observed in Figure 10 (a), the distortion is much lower than masking threshold at 256kbps mode, and then no distortion control is needed. However at 128kbps mode, the distortion exceeds the threshold at the 19th and higher scalefactor bands, and then the distortion control is needed to shape the noise.

Figure 10 (b) shows the average. Similarly, the distortion only exceeds the threshold at higher scalefactor bands.

Related research has been made to investigate the contribution of PAM-II to the distortion control [4]. By analyzing the number of distortion control iteration and the result of subjective quality preference tests, Oh et al. [4] showed that PAM-II is unnecessary when the bit rate is over 256kbps. To recover the audio quality at lower bit rate, they proposed a bandwidth control scheme. Subjective test revealed people prefer the sound with a limited bandwidth to the sound with full bandwidth but with unmasked distortion. In this thesis, we also employ bandwidth control of input signal. Figure 11 shows the bandwidth coefficient versus demand bit rate.

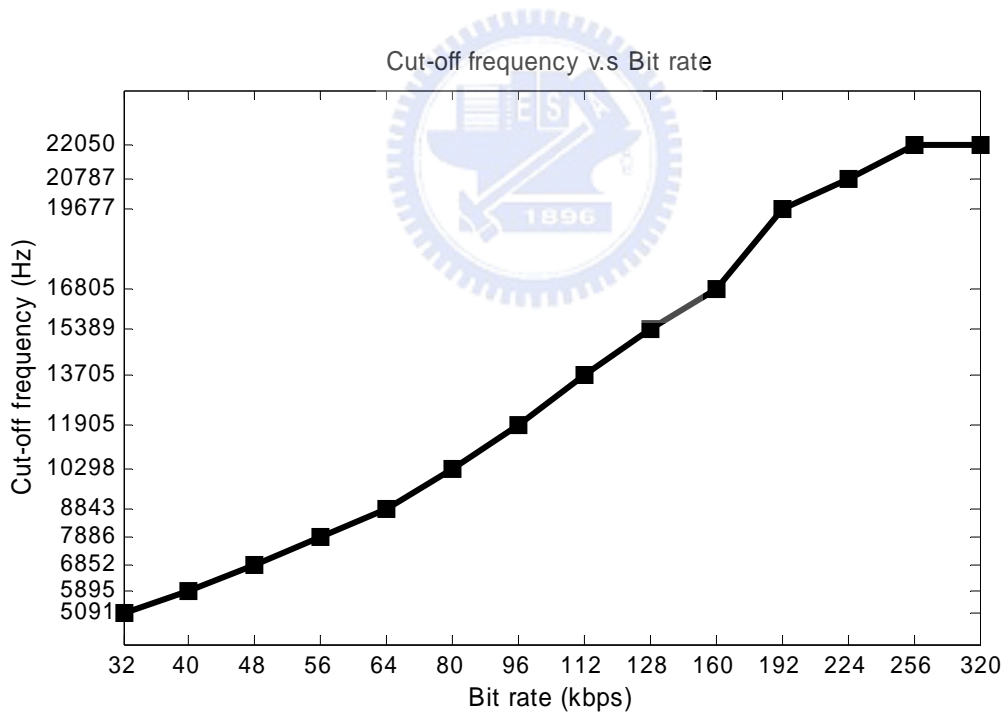


Figure 11. Coefficients of bandwidth controller (sampling rate is 44100 Hz): the corresponding cut-off frequency of each bit rate is obtained from LAME [14]

A low pass filter is applied in the bandwidth control. The i^{th} frequency line

$x_{f,g}(i)$ in the g^{th} granule of the f^{th} frame is filtered by (3),

$$L(x_{f,g}(i)) = \begin{cases} x_{f,g}(i) & , \text{if } i \leq \text{nint}\left(\frac{\Omega_c}{\frac{f_s}{2}} \times 576\right) \\ 0 & , \text{if } i > \text{nint}\left(\frac{\Omega_c}{\frac{f_s}{2}} \times 576\right) \end{cases}, \quad (3)$$

where the cut-off frequency, Ω_c is defined as the bandwidth coefficient.

In other words, the bandwidth control can allocate more bits for low frequency band, and then the quality can be improved. Figure 10 (b) shows the decrease of average distortion when bandwidth control is employed. Experiments shows that the bandwidth control scheme is effective when the MP3 is encoded at lower bit rate.

In this thesis, we propose removal of ISO PAM-II and related processes like distortion control and window switching and employ efficient allocation of limited bit resource to recover the audio quality. Later we will address the proposed method to allocate bit resource more efficiently.

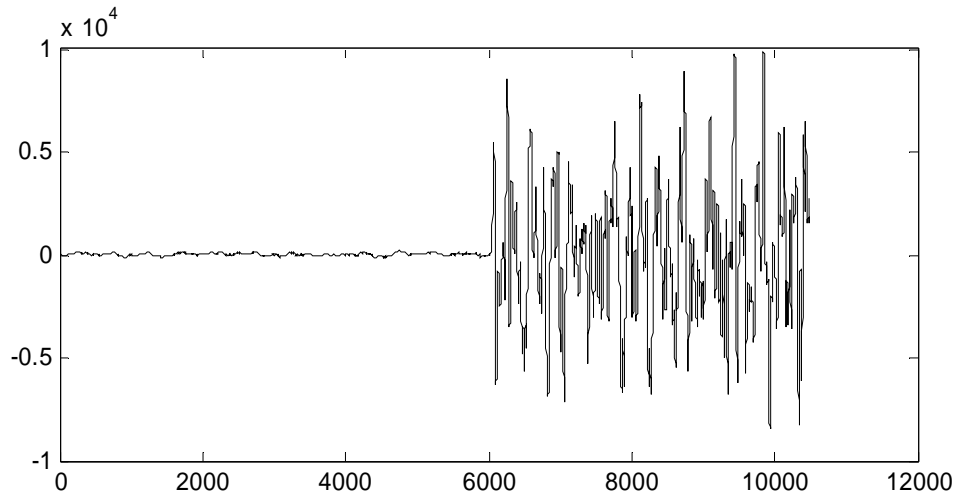
2.2.2 Removal of window switching

Modern audio compression algorithm often use dynamic window switching to avoid preechoes. Preecho happens when we encode audio signals that the amplitude raises violently in an instant as observed in Figure 12 (a). If the algorithm can't individually encode the signals of different characteristic, the signals grouped by algorithm will be encoded by using the same quantization parameter, i.e. the quantization noise are spreading to the whole block, and it is hard to get better coding gain. In transform coding based algorithm, signal of the same time-slice are always grouped first and then encoded at a time so the preechoes are unavoidable. But the psychoacoustics reveals that the preecho less than 18ms can be masked by a loud voice behind it.

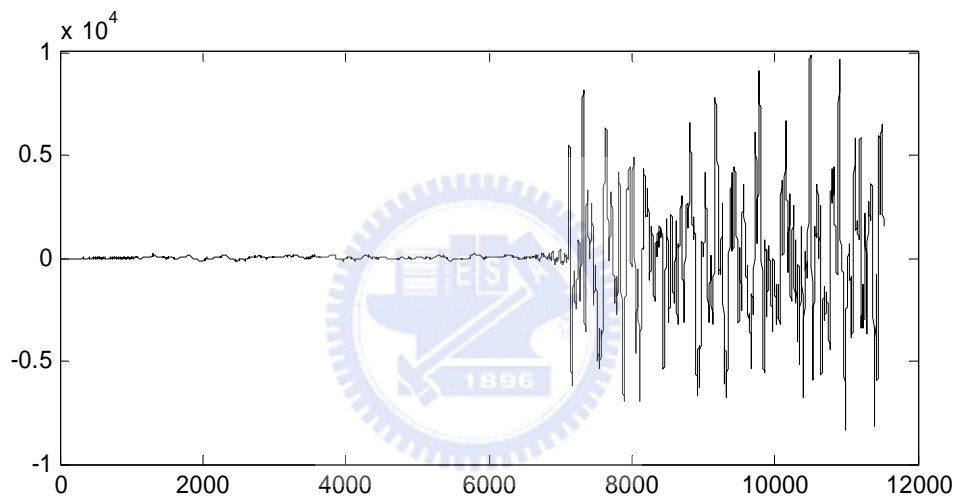
Thus the algorithm can group less amount of signals and encode them together even if the preecho is produced. That is why we need dynamic window switching in MP3 algorithm. In general PAM-II detects the appearance of preecho by calculating the perceptual entropy (PE), i.e. the predicted amount of bits needed to encode the granule. But PAM-II is not implemented in proposed design for power-limited device. It is also not easy to have another metric to detect the appearance of preecho.

Related research shows that encoding without window switching didn't cause significant negative effect to the audio quality [4]. Figure 12 (b) and (c) show the time domain waveforms encoded with and without window switching in 128Kbps. Preecho appears as a notable difference around the 7000th sample (the 6000th sample of source signal) whether the window switching is used or not.

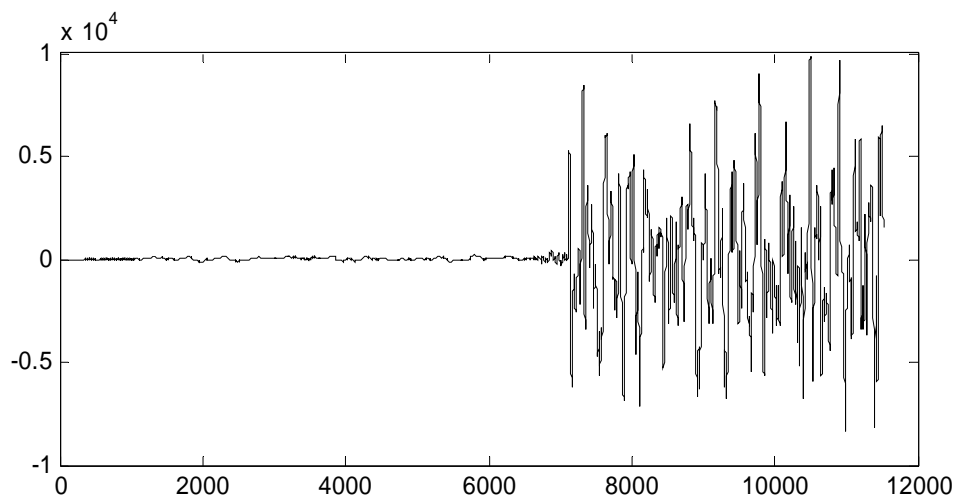




(a) Source signal



(b) Encoded with window switching



(c) Encoded without window switching

Figure 12. Time domain waveforms from using window switching or not

2.3 Fast rate control loop

As observed in Figure 13, the rate control loop also called inner iteration loops allocates the bit resources to each frequency line by quantization and Huffman coding. The difficulty is to find an optimal quantizer parameter also called global gain and select a suitable Huffman table. The ISO standard adopts a step-by-step approach to obtain the optimal parameter from an initial value determined by spectral flatness measure. Considering the input range of Huffman coding, more iteration taken in quantization process will be tested to guarantee the quantization output in the range.

In this thesis, we propose a new rate control algorithm. Figure 14 illustrates the flowchart of the new algorithm. With the removal of PAM-II and related distortion control loop, the iteration loops is also simplified as Figure 14.

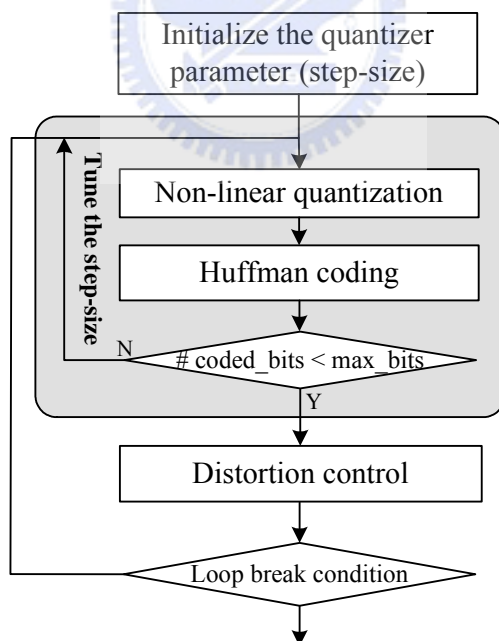


Figure 13. Rate control in iteration loops

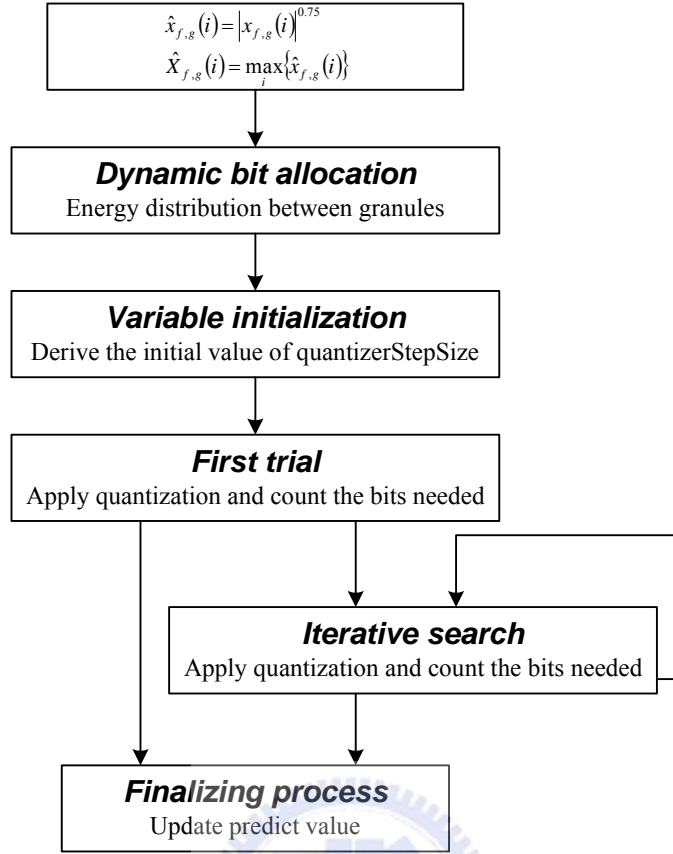


Figure 14. The new rate control algorithm

2.3.1 Non-uniform quantization

In ISO MP3 algorithm, the non-uniform quantizer was defined as (4),

$$y_{f,g}(i) = \text{nint} \left(\left(\frac{|x''_{f,g}(i)|}{2^{\frac{\delta+q}{4}}} \right)^{0.75} - 0.0946 \right), \quad (4)$$

where nint is a rounding function, q is the lower bound of quantization parameter, i.e. the initial value, δ is the increasing variable, and $x''_{f,g}(i)$ is the i^{th} frequency line pre-emphasized (5) and amplified (6) in the distortion control loop.

$$x'_{f,g}(i) = x_{f,g}(i) \times \sqrt{2}^{z_2 \times (1+z_1) \times P(b_i)}, \text{ and} \quad (5)$$

$$\hat{x}_{f,g}''(i) = x'_{f,g}(i) \times \sqrt{2}^{(1+z_1) \times C(b_i)}, \quad (6)$$

where $x_{f,g}(i)$ represents the original frequency line, i is the index of spectral line, $z_2 \in \{0,1\}$ switches on or off preemphasis, $z_1 \in \{0,1\}$ determines whether the scalefactors are logarithmically quantized with a step size of 2 or $\sqrt{2}$. b_i is the scalefactor band index of the i^{th} spectral line. $P(\cdot)$ is the preemphasis table defined in [5]. $C(b_i)$ is the scalefactor of the scalefactor band, b_i .

Since the distortion control is not used in this implementation, (5) and (6) no longer exist. Then (4) can be simplified to (7),

$$y_{f,g}(i) = \text{nint} \left(\left(\frac{|x_{f,g}(i)|}{2^{\frac{\Delta_{f,g}}{4}}} \right)^{0.75} - 0.0946 \right), \quad (7)$$

where the quantization parameter $\Delta_{f,g} = \delta + q$.

(7) is executed iteratively in the finding of an optimal $\Delta_{f,g}$. The rounding function nint is unnecessary in fixed point implementation. We can further rearrange (7) to (8),

$$y_{f,g}(i) = |x_{f,g}(i)|^{0.75} \times 2^{-\frac{3 \times \Delta_{f,g}}{16}} - 0.0946. \quad (8)$$

In the rate control iteration, $\Delta_{f,g}$ is the only running variable. So we can take $|x_{f,g}(i)|^{0.75}$ out from the iteration. Therefore the quantizer can be decomposed into two equation where one is calculated outside the iteration (9),

$$\hat{x}_{f,g}(i) = |x_{f,g}(i)|^{0.75}, \quad (9)$$

and another is calculated in the iteration (10),

$$y_{f,g}(i) = \hat{x}_{f,g}(i) \times 2^{-\frac{3 \times \Delta_{f,g}}{16}} - 0.0946. \quad (10)$$

The decomposition benefits the complexity reduction of the non-uniform quantizer. The most computationally demanding process, the $|x_{f,g}(i)|^{0.75}$ function, is only calculated once in each granule. And the iterative equation (10) in the fixed point implementation can be simplified to one multiplication, one shift operation and one subtraction.

The implementation of (9) is optimized for the target DSP, ADSP-2181. The unsigned 16-bit fixed point inputs $x_{f,g}(i)$ ranged from 0 to 65535 are divided into two regions. The first region covering range from 0 to 31 is implemented using a 32-word lookup table. From the probability model of $|x_{f,g}(i)|^{0.75}$, the first region covered over 60 percentage of inputs. A small lookup table is applied here to speedup the calculation. The second region from 32 to 65535 is implemented using piecewise linear interpolations. There are 11 sub-regions between 32 to 65535. The segmentation is also optimized for the target DSP. Since ADSP-2181 supports hardware detector of leading ones/zeros, we can derive biased $\log_2(x)$ in one instruction cycle. Thus the boundaries of sub-regions are proposed to be set to power of 2, i.e. 32, 64, 128, ..., 65536. The approximation error has been analyzed in (11),

$$\varepsilon(x_{f,g}(i)) = \frac{|x_{f,g}(i)|^{0.75} - \text{pow075}(x_{f,g}(i))}{|x_{f,g}(i)|^{0.75}}, \quad (11)$$

where pow075 is the implementation of proposed approximation method, also represented by $Q_1(\cdot)$.

Figure 15 shows the error to real output ratio, ε . The ratio is around $\pm 1\%$.

Table 2 summaries the number of DSP instruction cycles in calculation of two regions.

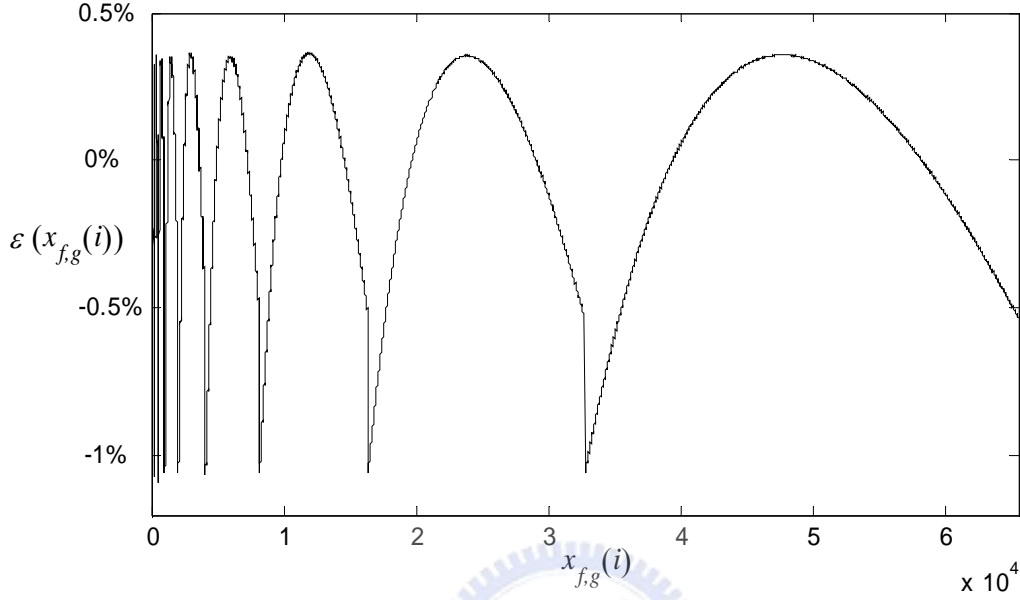


Figure 15. The error of $|x_{f,g}(i)|^{0.75}$ approximation

Table 2. The number of DSP instruction cycles in calculation of two regions

Input range	DSP instruction cycles	Probability	Table size
0 ~ 31	4	> 60%	32 words
32 ~ 65535	9	< 40%	22 words

We can rewrite the (10) as (12),

$$y_{f,g}(i) = \hat{x}_{f,g}(i) \times 2^{\Delta_Q} \times 2^{\Delta_N} - 0.0946, \quad (12)$$

where Δ_N is the integer part, and Δ_Q is the fractional part of $\frac{-3 \times \Delta_{f,g}}{16}$. In fixed point implementation, the multiplication of 2^{Δ_N} can be easily implemented by the hardware barrel shifter. And the 2^{Δ_Q} is derived from a

16-word lookup table that contains fixed point value $2^{\frac{0}{16}}$, $2^{\frac{1}{16}}$, ..., $2^{\frac{15}{16}}$. The implementation is denoted as $Q_2(\cdot)$.

2.3.2 Dynamic bit allocation proportional to the energy of granules

In the MP3 bitstream, each frame has fixed amount of bit resources on the constant bit rate. With a help of bit reservoir control, we can save the unused bits in the reservoir if the distortion of quantization is imperceptible, and it will benefit the encoding of succeeding frames. But in the proposed algorithm without PAM-II and distortion control, the quality constraint is no longer exist. Then the rate control loop will exploit all the bit resource as possible as it can to encode one audio granule.

Normally an audio frame contains two (mono) or four (stereo) granules. The traditional MP3 algorithm portioned out the total bits equally for granules in each frame. An asymmetric allocation of the bit resources which is proportional to the energy of granules is proposed to equalize the quality, i.e. allowed distortion, between granules in the same frame.

It is general that the transformed coefficient with higher amplitude will be quantized to higher integer values. And from the property of the Huffman code words, the integer input with higher value is usually coded with more bits. We can also extend the idea to the group of coefficients, i.e. granule. If the granule has more energy or more number of coefficients with higher amplitude, it need more bits to maintain the same quality as others.

Considering the non-uniform property of quantizer, the power function is taken into account. From experimental results, the frequency lines below 4,000Hz dominates the full bandwidth (22050Hz) energy. In the proposed approach, for sampling rate of 44.1kHz the score of the granule energy defined as (13) takes only $\frac{4000}{44100/2} \times 576 \approx 105$ spectral lines of $|x_{f,g}(i)|^{0.75}$ for

calculation.

$$E_{f,g} = \sum_{i=1}^{105} \hat{x}_{f,g}(i), \quad (13)$$

where $\hat{x}_{f,g}(i)$ are determined from (9), and $E_{f,g}$ is the energy score of the granule.

The resource allocation is not exactly proportional to the granule score. The modification as shown in (14) takes the minimum resource into account.

$$B_{f,g} = b_{f,g} + \frac{E_{f,g}}{\sum_g E_{f,g}} \times B_p, \quad (14)$$

where $b_{f,g}$ is the minimum encoding bits given from (15), and B_p is the number of bits used to distribute to each granule given from (16). B_f is the total available number of bits in the frame.

$$b_{f,g} = \begin{cases} \frac{B_f}{6}, & \text{Mono} \\ \frac{B_f}{12}, & \text{Left/Right Channel} \\ \frac{B_f}{9}, & \text{Mid channel} \\ \frac{B_f}{18}, & \text{Side channel} \end{cases} \quad (15)$$

$$B_p = B_f - \sum_g b_{f,g} \quad (16)$$

In this thesis, we also propose the fast search approach which has two following parts. One is the precise initialization of the quantization parameter, i.e. $\Delta_{f,g}$. Another is the fast search of the optimal quantizer parameter.

2.3.3 Precise initialization of the quantization parameter

In the ISO MP3 algorithm, the initial value of quantization parameter is

derived as (17),

$$q = 8.0 \times \ln(\mu_{f,g}). \quad (17)$$

The spectral flatness measure, $\mu_{f,g}$, is defined as (18). The derivation contains complex non-linear mathematic and is inefficient on fixed point implementation.

$$\mu_{f,g} = \frac{e^{\frac{1}{576} \left(\sum_{i=0}^{575} \ln(x_{f,g}(i)^2) \right)}}{\frac{1}{576} \cdot \left(\sum_{i=0}^{575} x_{f,g}(i)^2 \right)} \quad (18)$$

We propose that the initialization of $\Delta_{f,g}$ is predicted by the one of previous granule and a lower bound. To derive the lower bound we consider (10) again. From the property of Huffman table, the quantized value, $y_{f,g}(i)$, has a upper bound, 8207. So the lower bound of $\Delta_{f,g}$ comes out from the direct derivation from (19),

$$\begin{aligned} 8207 &> \hat{x}_{f,g}(i) \times 2^{\frac{3 \times \Delta_{f,g}}{16}} - 0.0946 \\ \Rightarrow 8207 &> \max_i \{ \hat{x}_{f,g}(i) \} \times 2^{\frac{3 \times \Delta_{f,g}}{16}} - 0.0946 \\ \Rightarrow \Delta_{f,g} &> -\frac{16}{3} \log_2 \left(\frac{8207 + 0.0946}{\max_i \{ \hat{x}_{f,g}(i) \}} \right) \\ \Rightarrow \Delta_l &= \left\lceil \frac{16}{3} \log_2 \left(\max_i \{ \hat{x}_{f,g}(i) \} \right) - 69.35 \right\rceil \end{aligned} \quad (19)$$

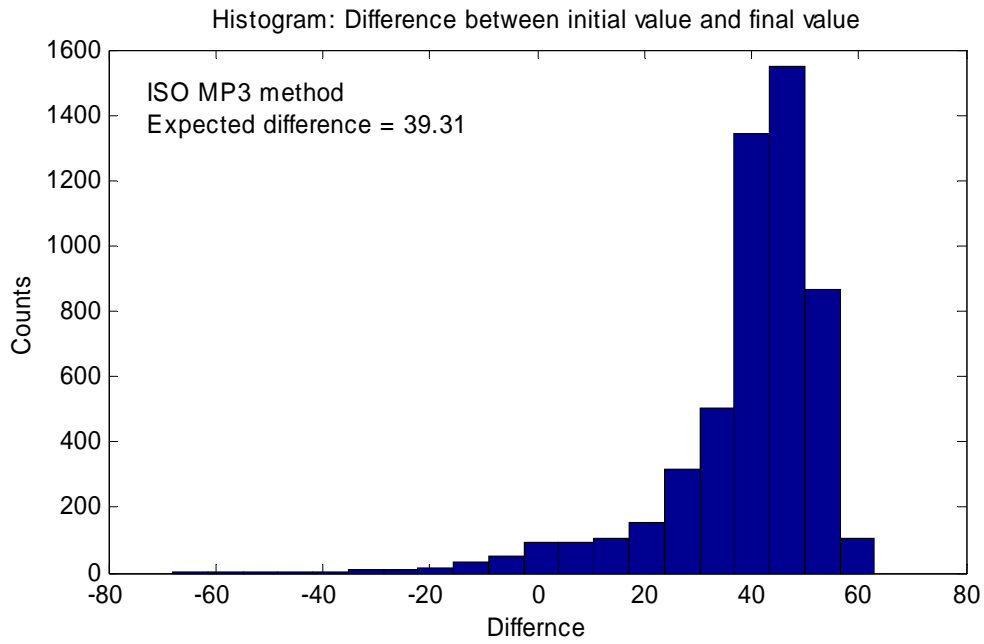
The lower bound, Δ_l , guarantees the quantized value in range of Huffman table. So the initialization with prediction is derived as (20),

$$\Delta_{f,g}(n) = \max \{ \Delta_l, \Delta_{f,g}(n-1) + \sigma \}, \quad (20)$$

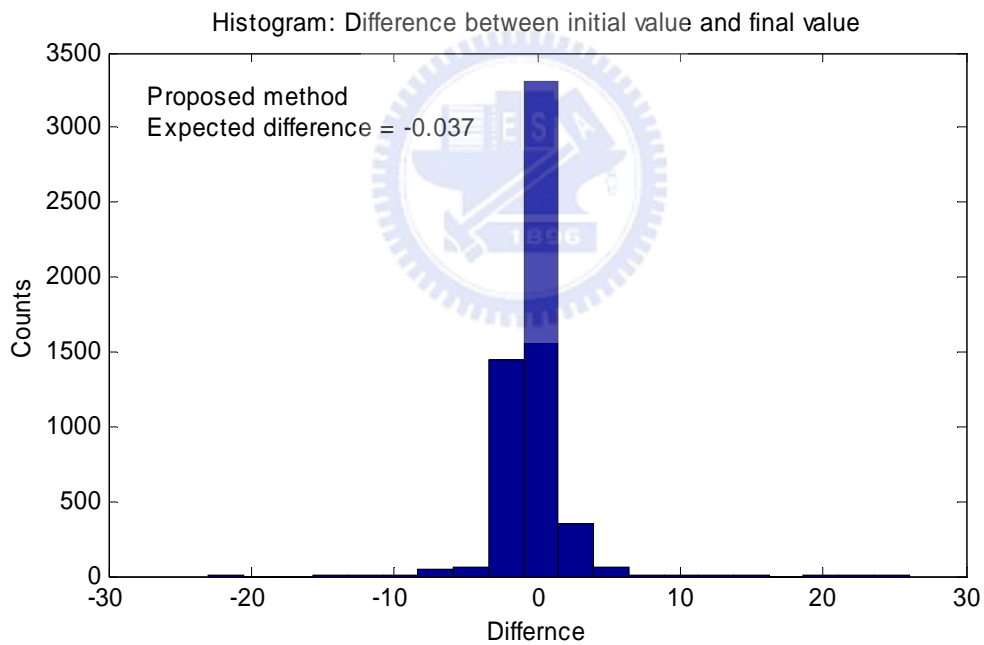
where n is the iteration index of iterative search in Figure 14 and starts from zero. $\Delta_{f,g}(-1) = \Delta_{f-1,g}$, $\Delta_{0,g}(-1) = -150$, σ is the addend of the step size and equal to zero during initialization.

The great achievement of proposed method is proved by comparing the difference between initial value and final value. Figure 16 shows the difference histogram of ISO MP3 method and proposed method. From the statistic, over 60 percent of predicted initial values are very close to the final values, i.e. the difference $|\varepsilon| \leq 1$. The precise decision of initial value benefits to reduce the number of iteration of the following iterative search.





(a) ISO MP3 method in (17) and (18)



(b) Proposed method in (19) and (20)

Figure 16. The histogram to difference between initial value and final value of $\Delta_{f,g}$. The accuracy is determined by the expected difference of initial value and final value. (a) ISO method initializes it by the measure of spectral flatness. (b) Proposed method initializes it by the one of previous granule and a lower bound.

2.3.4 Fast search of the optimal quantizer parameter

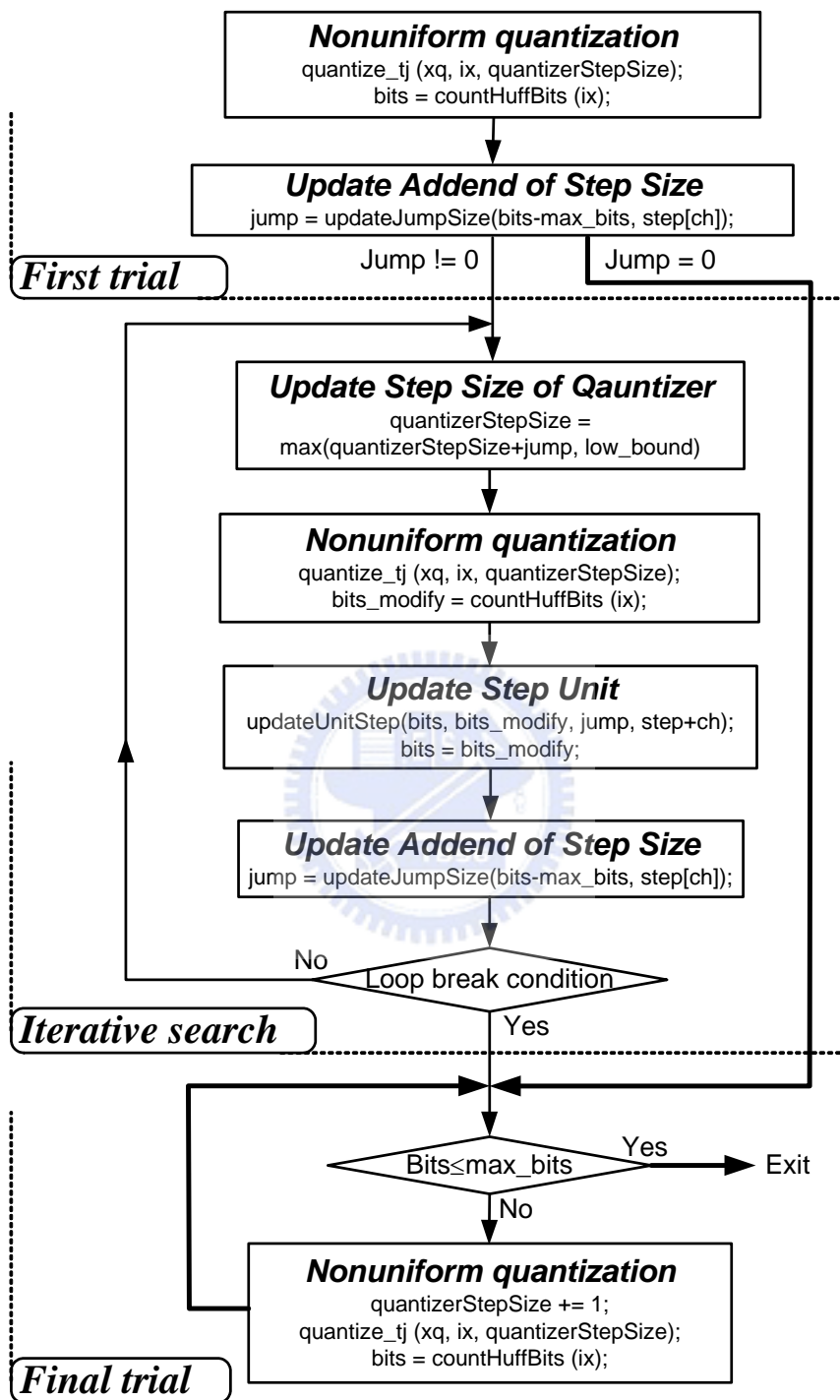


Figure 17. The adaptive approach to iterative search optimum parameter

Figure 17 illustrates the our approach to iterative search. Table 3 describes the symbols used in Figure 17.

Table 3. Symbols descriptions of Figure 17

Symbol name	Description	Abbreviation
xq	The frequency lines powered by 0.75 in (9)	$\hat{x}_{f,g}$
ix	The quantized integer value in (10)	$y_{f,g}$
$quantizerStepSize$	Quantizer parameter also called global gain	$\Delta_{f,g}$
low_bound	The lower bound of $\Delta_{f,g}$ guaranteeing that the quantized value can be coded within Huffman table	Δ_l
$jump$	Addend of $\Delta_{f,g}$	σ
$step$	Predicted value of the difference number of bits used in Huffman coding when $\Delta_{f,g}$ is increased by one.	ρ_c
max_bits	The bits budget of this granule determined from (14)	$B_{f,g}$
$bits, bits_modify$	Number of bits used in Huffman coding of the quantized values	b^h, \hat{b}^h
$quantize_tj$	Implementation of (10)	$Q_2(\cdot)$
$countHuffBits$	Counting the number of bits used in Huffman coding of $y_{f,g}$	$C^h(\cdot)$
$updateUnitStep$	Updating ρ_c by results of the latest two iterations	$U^s(\cdot)$
$updateJumpSize$	Updating σ by ρ_c	$U^j(\cdot)$

The proposed approach can be divided into three parts. The first part, the first trial, performs quantization with the initial value of $\Delta_{f,g}$ derived from (20). Then $\hat{x}_{f,g}$ are quantized to $y_{f,g}$ by $Q_2(\cdot)$. The following $C^h(\cdot)$ will choose appropriate Huffman tables for $y_{f,g}$ and count the number of coded bits, $b^h(0)$. Based on ρ_c and the difference of $b^h(0)$ and $B_{f,g}$, a new σ is derived by $U^j(\cdot)$. The σ equal to zero implies that $b^h(0)$ is very close to $B_{f,g}$ then we omit the iterative search part and apply final trial directly.

The iterative search is applied when the σ is not equal to zero. n is

representing the iteration index. In the n^{th} iteration, (20) is used to derive the $\Delta_{f,g}(n)$, and n starts from one where $\Delta_{f,g}(0)$ is used in the first trial. After the update of $\Delta_{f,g}(n)$, $Q_2(\cdot)$ and $C^h(\cdot)$ is used to obtain $y_{f,g}$ and \hat{b}^h . The difference number of bits with previous iteration, $\hat{b}^h - b^h(n-1)$, σ and $\rho_c(n-1)$ are sent to $U^s(\cdot)$, and a new $\rho_c(n)$ is updated. Similar with the first trial, $U^j(\cdot)$ determines a new σ used in the $(n+1)^{\text{th}}$ iteration. The iterative search block is terminated while one of the following loop break conditions exists.

- n is greater than 5,
- $\hat{b}^h - b^h(n-1)$ is less than 32,
- σ is zero.

The final trial is applied to guarantee that $b^h \leq B_{f,g}$. Different from the iterative search, the fine tune of $\Delta_{f,g}$ is applied here to prevent the deadlock loop condition.

For example, let $B_{f,g} = 1000$, $\Delta_{f,g}(0) = -80$, $\Delta_l = -100$, and $\rho_c(0) = 100$ (obtained from previous granule), $\hat{x}_{f,g}$ are passed to $Q_2(\cdot)$ and $C^h(\cdot)$ then we obtain $b^h(0) = 500$. $U^j(\cdot)$ updates by

$$\sigma = \text{nint}\left(\frac{b^h(0) - B_{f,g}}{\rho_c(0)}\right) = -5$$

after the first trial. Since σ is not equal to zero the iterative search is applied. (20) will update $\Delta_{f,g}(1)$ as -85. $Q_2(\cdot)$ and $C^h(\cdot)$ are then executed again to

obtain $\hat{b}^h = 900$. $U^s(\cdot)$ updates by

$$\rho_c(1) = \text{nint}\left(\frac{3}{4} \cdot \rho_c(0) + \frac{1}{4} \cdot \frac{\hat{b}^h - b^h(0)}{\sigma}\right) = 95$$

in 1st iteration. Then $\Delta_{f,g}(2)$ is updated again to -86. And we will repeat the process iteratively until any one of loop break condition exists.

In this thesis, we propose a new iteration loops algorithm. Figure 18 compares the pseudo code of ISO and the proposed method. With the removal of PAM-II, the distortion control is also removed, and the rate control is optimized for speedup. The solid lines in Figure 18 link the blocks with the same functionality but optimized in proposed method. The dotted lines link the blocks with different measurement in the proposed method. And the boldface represent the added blocks of the proposed method.

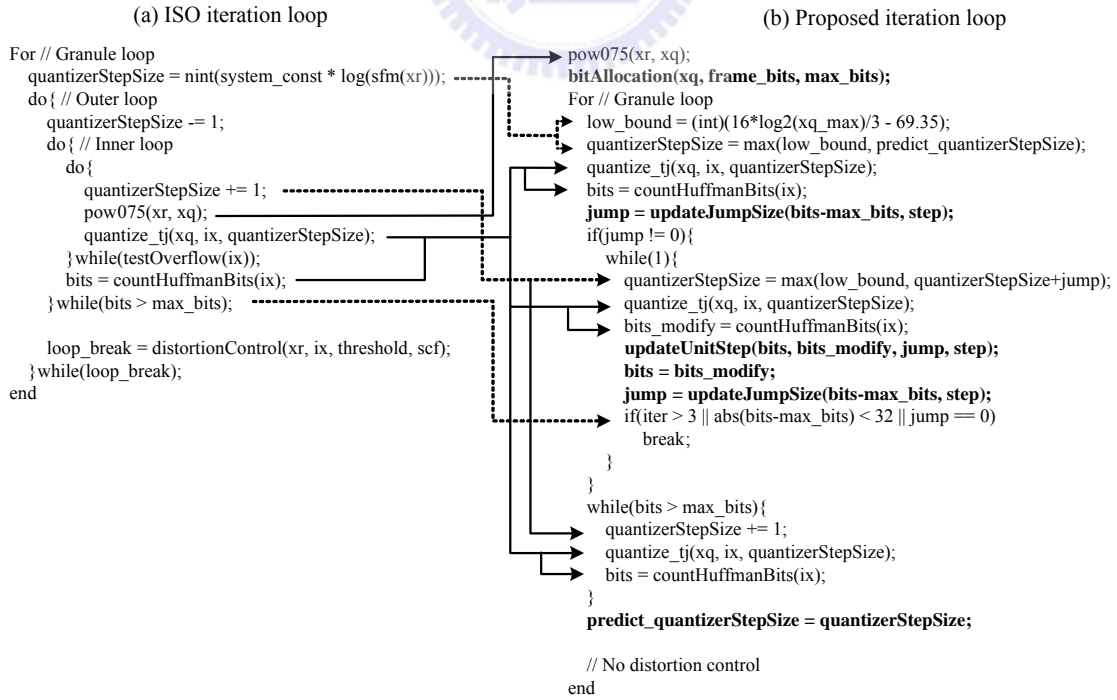


Figure 18. Pseudo code of iteration loops (a) ISO method (b) Proposed method

To evaluate the performance of optimized rate control process, the number of iterations has been analyzed by calculating the execution times of $Q_1(\cdot)$, $Q_2(\cdot)$, and $C^h(\cdot)$ in each granule. The computational complexities are denoted as p , q , c individually. In encoding stereo MP3 with 128Kbps, the experiments show that the ISO method takes $45p+45q+47c$ in average while the proposed method takes $1p+2q+2c$ only. Table 4 lists the number of inner iteration in each method. The proposed method takes less iteration numbers than other methods. And due to decomposition of non-uniform quantizer, $Q_1(\cdot)$ and $Q_2(\cdot)$, the computational complexity of inner iteration is also much less than other methods.

Table 4. The average number of inner iteration

	ISO	Oh et al. [4]	Proposed
Average	45	2.1	1.8
Max	>100	3	8

CHAPTER 3. DECODER OPTIMIZATION

3.1 Decoding Overview and Complexity Analysis



Figure 19. MPEG/Audio Layer III decoding block diagram

The MPEG/Audio layer III decoding process has three main parts [5]: bitstream decoding, inverse quantization and frequency-to-time mapping as shown in Figure 19. The first part synchronizes the encoded bitstream input and extracts the quantized frequency coefficients and other information of each frame. Figure 20 illustrates the detail function blocks.

The second part, inverse quantization also called dequantization, reconstructs a perceptually identical data of the frequency coefficients generated by the MDCT block during encoding. Based on the output of Huffman decoding and scalefactor information, the dequantization equation is represented in (21) [5].

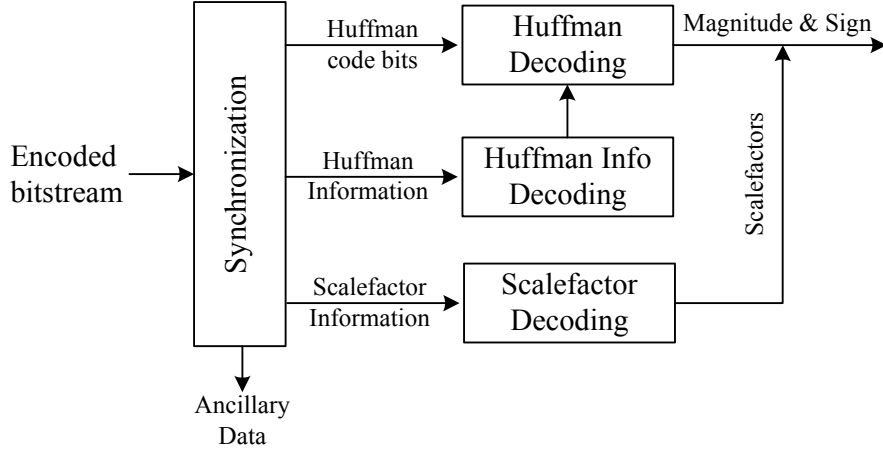


Figure 20. Bitstream decoding

$$x_{f,g}(i) = (-1)^{s(i)} \cdot y_{f,g}^{\frac{4}{3}}(i) \cdot \frac{2^{\frac{1}{4}(\Delta_{f,g} - 8 \cdot \Delta_s(w_i))}}{2^{\frac{1}{4}(1+z_1)(C(b_i)+P(b_i))}}, \quad (21)$$

where $y_{f,g}(i)$ is the output of Huffman decoding, and $\Delta_{f,g}$, z_1 , $\Delta_s(w_i)$ (subblock gain only used in short block), and $C(b_i)$ are part of side information.



Figure 21. Frequency to time mapping

The last part, frequency to time mapping, produces the audio PCM output from the dequantized frequency lines. The part is a set of reversed operations of the MDCT and analysis polyphase filterbank in the encoder. The alias reduction block adds alias artifacts to dequantized outputs in order to obtain a correct reconstruction of subband signals. Then the inverse MDCT reconstructs time domain subband signals from frequency lines. The frequency inversion is then applied in order to compensate the decimation used in the analysis polyphase filterbank. After that, the synthesis polyphase filterbank, also called subband synthesis, is applied to the subband signals to yield the audio PCM output.

Among them, dequantization, IMDCT, and synthesis polyphase filterbank in particular require a large number of arithmetic operations and produce quantization noise in fixed point implementation. In this thesis, we propose a fast realization of dequantization and adopt fast algorithms on IMDCT and synthesis polyphase filterbank.

3.2 Dequantization

The dequantization equation is represented in (21). The complexity is the calculation of $y_{f,g}^{\frac{4}{3}}$ where $y_{f,g}(i)$ is an integer ranging 0 to 8207. The direct derivation using mathematic libraries is too time-consuming and not suitable for real-time implementation.

First the calculation of $y_{f,g}^{\frac{4}{3}}$ is decomposed into (22) in order to minimize the quantization noise of fixed point implementation. Comparing the dynamic range of $y_{f,g}^{\frac{4}{3}}$ (0 to 165543.67) and $y_{f,g}^{\frac{1}{3}}$ (0 to 20.171), it is obvious that the implementation of $y_{f,g}^{\frac{1}{3}}$ produces lower quantization noise because of the smaller dynamic range.

$$y_{f,g}^{\frac{4}{3}}(i) = y_{f,g}^{\frac{1}{3}}(i) \cdot y_{f,g}(i) \quad (22)$$

Similarly with encoder case, the power function is implemented with hybrid scheme. First the input range is split into three section as shown in Figure 22. The first section, $0 \leq y_{f,g}(i) < 32$, utilizes a small lookup table to obtain the noiseless value directly. Another two sections adopt the piecewise linear approximation method. The segmentation is also optimized for the target DSP. In order to minimize the approximation error, the segmentation of the second section has been

made according to the leading-zeros of $y_{f,g}^3(i)$.

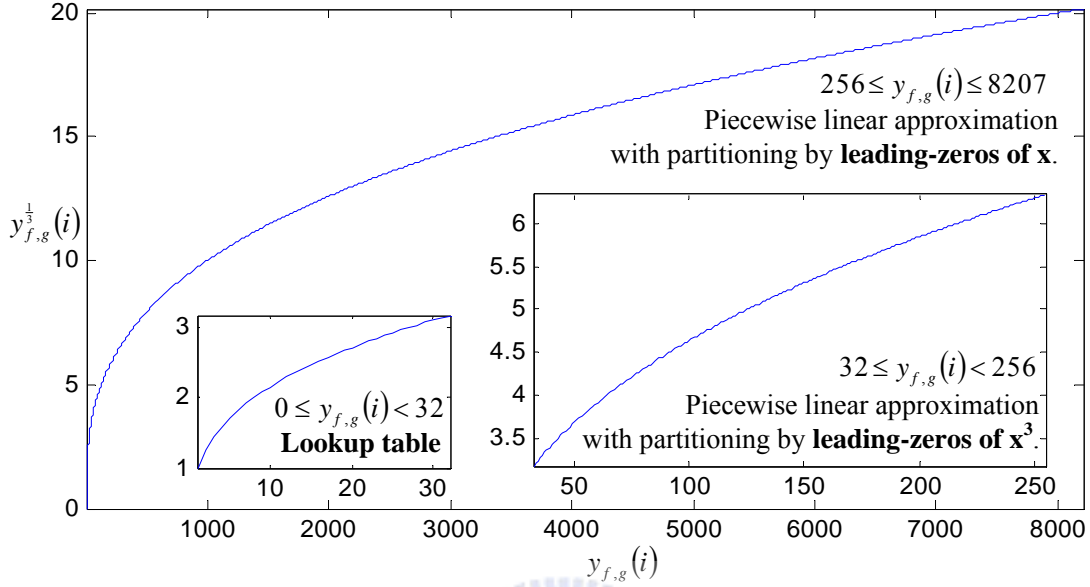


Figure 22. The implementation of $y_{f,g}^{\frac{1}{3}}(i)$

Ignoring the frequency index i for general, (23) represents the approximation of $u = y_{f,g}^{\frac{1}{3}}$.

$$u = \begin{cases} \text{LUT}(y_{f,g}^{\frac{1}{3}}) & , 0 \leq y_{f,g} < 32 \\ \alpha_2(S_2(y_{f,g})) \cdot y_{f,g} + \beta_2(S_2(y_{f,g})) & , 32 \leq y_{f,g} < 256 \\ \alpha_3(S_3(y_{f,g})) \cdot y_{f,g} + \beta_3(S_3(y_{f,g})) & , 256 \leq y_{f,g} \leq 8207 \end{cases} \quad (23)$$

where $\text{LUT}(\cdot)$ represents the lookup table method applying in 1st section, α_2 and β_2 are the linear approximation coefficients of the 2nd section, α_3 and β_3 are the linear approximation coefficients of the 3rd section, $S_2(\cdot)$ is the segment index of the 2nd section derived from (24), and $S_3(\cdot)$ is the segment index of the 3rd section derived from (25).

$$\begin{aligned}
 B_2(j) &= \text{nint}\left(\left((32)^3 \cdot 2^j\right)^{\frac{1}{3}}\right), \quad j = 0 \sim 9 \\
 \Rightarrow B_2(j) &\in \{32, 41, 51, 64, 81, 102, 128, 162, 204, 256\} \\
 S_2(y_{f,g}) &= \{j \mid B_2(j) \leq y_{f,g} < B_2(j+1)\}
 \end{aligned} \tag{24}$$

$$\begin{aligned}
 B_3(j) &= \begin{cases} \text{nint}(256 \cdot 2^j), & j = 0 \sim 5 \\ 8208, & j = 6 \end{cases} \\
 \Rightarrow B_3 &\in \{256, 512, 1024, 2048, 4096, 8192, 8208\} \\
 S_3(y_{f,g}(i)) &= \{j \mid B_3(j) \leq y_{f,g} < B_3(j+1)\}
 \end{aligned} \tag{25}$$

The approximation error has been analyzed that the error to real output ratio is around $\pm 1\%$, and the SNR is around 46dB. The error is still too large and will probably lead the following processes like IMDCT and subband synthesis to produce more error, especially in fixed point implementation. Nevertheless in the encoding case, the following process, Huffman coding, is noiseless.

In order to obtain the further approximation, we propose to apply the Newton's method in the section of $32 \leq y_{f,g}(i) \leq 8207$. Let $u = y_{f,g}^{\frac{1}{3}}(i)$, where (26) is another representation which is suitable for the Newton's method of root-finding. The method will yield a value of u that approximates $y_{f,g}^{\frac{1}{3}}(i)$.

$$u^3 - y_{f,g}(i) = 0 \tag{26}$$

The function result is calculated through the repeated iterations that can successively reduce the residual error $|u^3 - y_{f,g}(i)|$. The iteration formula is shown in (27),

$$\tilde{u}_1 = \tilde{u}_0 - \frac{\tilde{u}_0^3 - y_{f,g}(i)}{3 \cdot \tilde{u}_0^2} = \frac{2\tilde{u}_0^3 + y_{f,g}(i)}{3 \cdot \tilde{u}_0^2} = \frac{1}{3} \cdot \left(2 \cdot \tilde{u}_0 + \frac{y_{f,g}(i)}{\tilde{u}_0^2} \right), \tag{27}$$

where the starting value \tilde{u}_0 is obtained from (23).

The desired accuracy can be achieved in only one iteration. Figure 23 shows

the error to real output ratio. The ratio is around $\pm 0.01\%$ and the SNR is increased to 86dB.

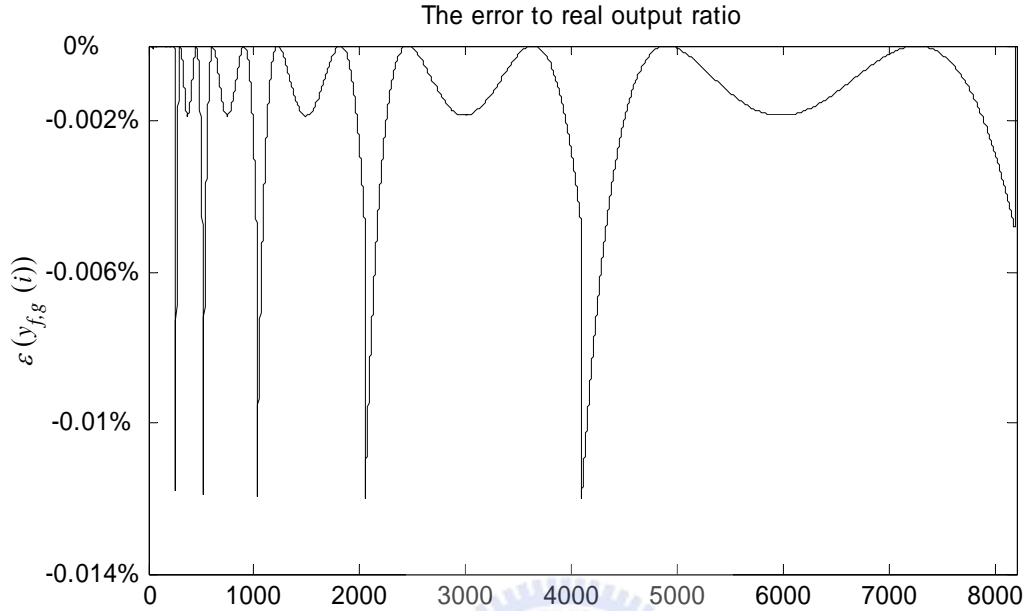


Figure 23. The error to real output ratio of $y_{f,g}^{\frac{4}{3}}(i)$ approximation.

$$\varepsilon(y_{f,g}(i)) = (y_{f,g}^{\frac{4}{3}}(i) - \text{pow3}(y_{f,g}(i)) \cdot y_{f,g}(i)) / (y_{f,g}^{\frac{4}{3}}(i)) \quad \text{where pow3}$$

is the proposed implementation of $y_{f,g}^{\frac{1}{3}}(i)$.

The effect of fixed point implementation have been analyzed. Figure 24 shows the error to real output ratio. The ratio is around $\pm 0.08\%$, and the SNR is around 82dB.

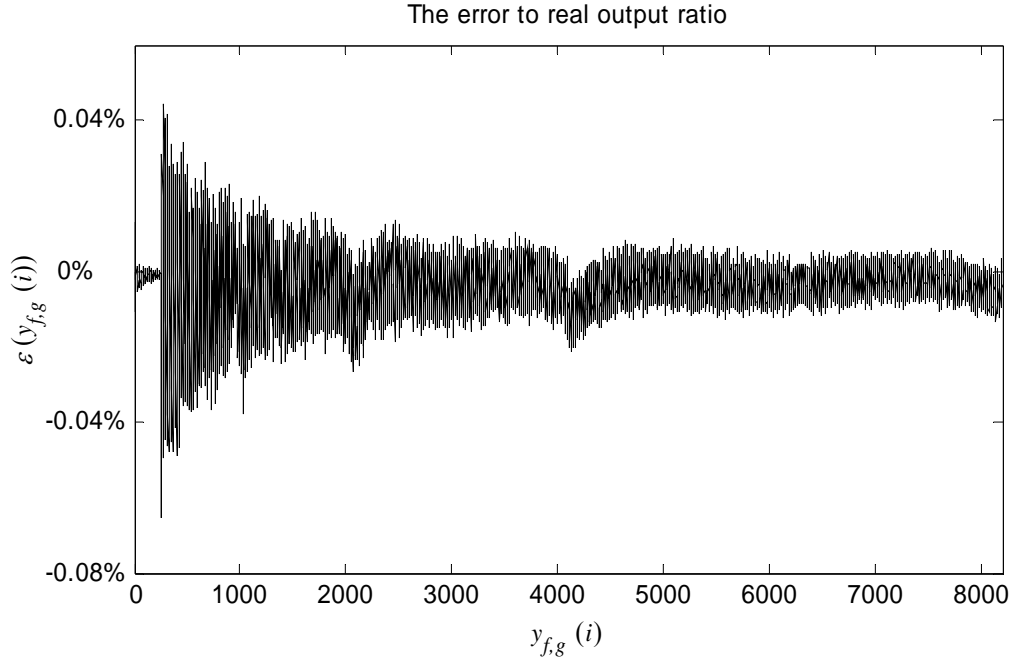


Figure 24. The error to real output ratio of $y_{f,g}^{\frac{4}{3}}(i)$ fixed point approximation.

$$\varepsilon(y_{f,g}(i)) = \left(y_{f,g}^{\frac{4}{3}}(i) - \text{pow3fx}(y_{f,g}(i)) \cdot y_{f,g}(i) \right) / \left(y_{f,g}^{\frac{4}{3}}(i) \right) \quad \text{where}$$

pow3fx is the proposed fixed point implementation of $y_{f,g}^{\frac{1}{3}}(i)$.

3.3 IMDCT and Subband Synthesis

The frequency to time mapping tool is another computationally demanding process. Especially in IMDCT and subband synthesis blocks there are a lot of multiply-accumulation operations with cosine coefficients. It is necessary to perform optimization such as fast algorithm. But, in general, a fast algorithm brings more quantization errors due to fixed point operations.

From the analysis result of Lee etc. [6], prevailing Lee's Fast DCT algorithm [7] is adopted for the fast algorithms of IMDCT and subband synthesis block. For IMDCT block 9-point and 3-point Lee's Fast IDCT is applied, and for matrixing routine in subband synthesis block 64-point Lee's Fast DCT is used.

CHAPTER 4. DSP IMPLEMENTATION

4.1 Target DSP Architecture

Using the proposed architecture, we implement the MP3 encoder and decoder by a 16-bit fixed point DSP, ADSP-2181. Figure 25 shows the block diagram of ADSP-2181 [17].

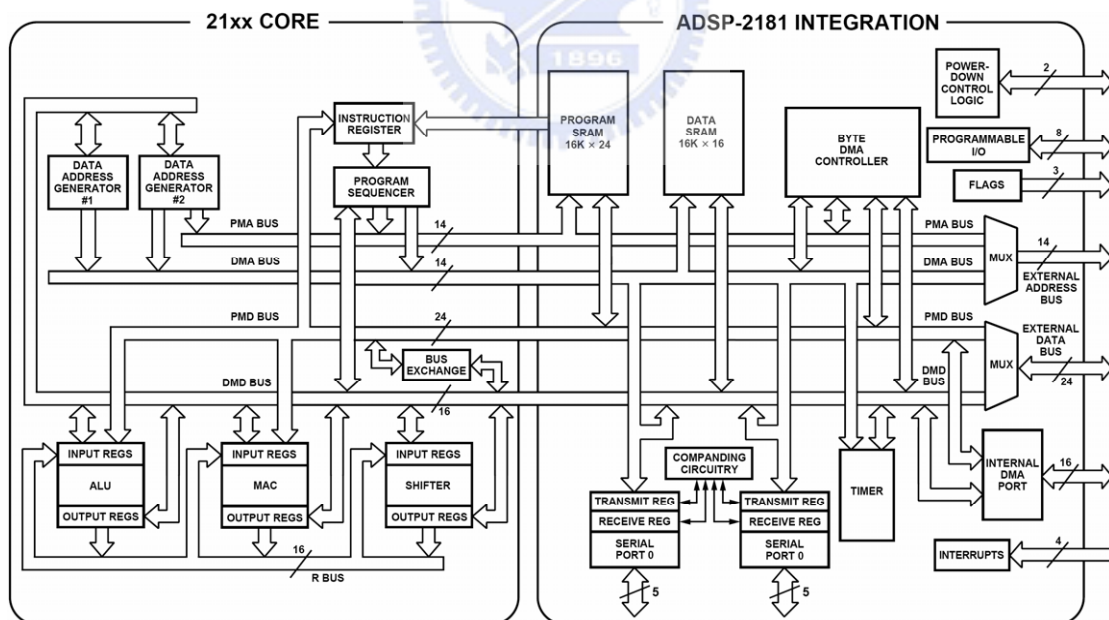


Figure 25. The ADSP-2181 DSP core and peripheral integration

The ADSP-2181 is a single-ship microcomputer optimized for digital signal processing (DSP) and other high speed numeric processing applications [17]. It

combines the ADSP-2100 family base architecture (three computational units, data address generators and a program sequencer) with two serial ports, a 16-bit internal DMA port, a byte DMA port, a programmable timer, Flag I/O, extensive interrupt capabilities and on-chip program and data memory.

The features of the ADSP-2100 family DSP core are as following [17]:

- **Computational units:** There are three independent, full-functional computational units including an 16-bit arithmetic/ logic unit (ALU), a 40-bit multiplier/ accumulator unit (MAC) and a 32-bit barrel shifter. The ALU performs a standard set of arithmetic and logic operations; division primitives are also supported. The MAC performs single-cycle multiply, multiply/ add and multiply/ subtract operations with 40 bits for accumulation. The SHIFTER performs logic and arithmetic shifts, normalization, denormalization and derive exponent operations. The SHIFTER can be used to efficiently implement numeric format control including multiword and block floating point representations.
- **Data address generators (DAGs):** Dual DAGs allow the processor to generate simultaneous address for dual operand fetches and support circular, post-modify and bit-reversed addressing modes. In sum-of-product calculation, DAGs allow the processor to fetch two operands and execute one ALU/ MAC/ SHIFTER instruction in single cycle.
- **Program sequencer:** provides single-cycle conditional branching and executes program loop with zero loop overhead.

ADSP-2181 also integrates on-chip RAM and peripherals. The DSP core can access the on-chip peripherals by memory-mapped control register. The integration are as following:

- **80K bytes on-chip RAM:** They are configured as 16K words program

memory RAM (24 bits per word) and 16K words data memory RAM (16 bits per word). ADSP-21xx uses a modified Harvard architecture in which data memory stores data, and program memory stores both program and data. This allows the processor core to fetch two operands (one from data memory and one from program memory) and an instruction (from program memory) in a single instruction cycle.

- **Serial ports (SPORTs):** There are two bi-directional, double-buffered serial ports for serial communication. Each SPORT can use an external serial clock or generates its own in a wide range of frequency down to 0 Hz. The SPORTs also support framing, hardware companding (A-law and μ -law), autobuffering, interrupt generation and multichannel capability (time-division multiplexed into 24 or 32 channels).
- **Timer:** The programmable interval timer provides periodic interrupt generation.
- **DMA ports:** There are two DMA ports, Internal DMA (IDMA) port and Byte DMA (BDMA) port. The IDMA port is a parallel I/O port that lets the processor's internal memory (except for the processor's memory-mapped control registers) be read or written by a host system. The read/ write access is completely asynchronous, and a host can access the DSP's internal memory with an overhead of one DSP processor cycle per word while the DSP is operating at full speed. The BDMA port allows processor load program and data from/ to external byte memory with very low processor overhead and supports interrupt generation while the DMA transfer is completed.

4.2 Data precision optimization in the proposed MP3 encoder

Basically ADSP-2181 performs 16-bit arithmetic. However, the double precision, i.e. 32-bit, arithmetic provides more accuracy of processing data but also

increases the computational complexity. As shown in Figure 26, five instructions are needed to perform the double precision multiplication, and the complexity is five times of the complexity of the single precision multiplication.

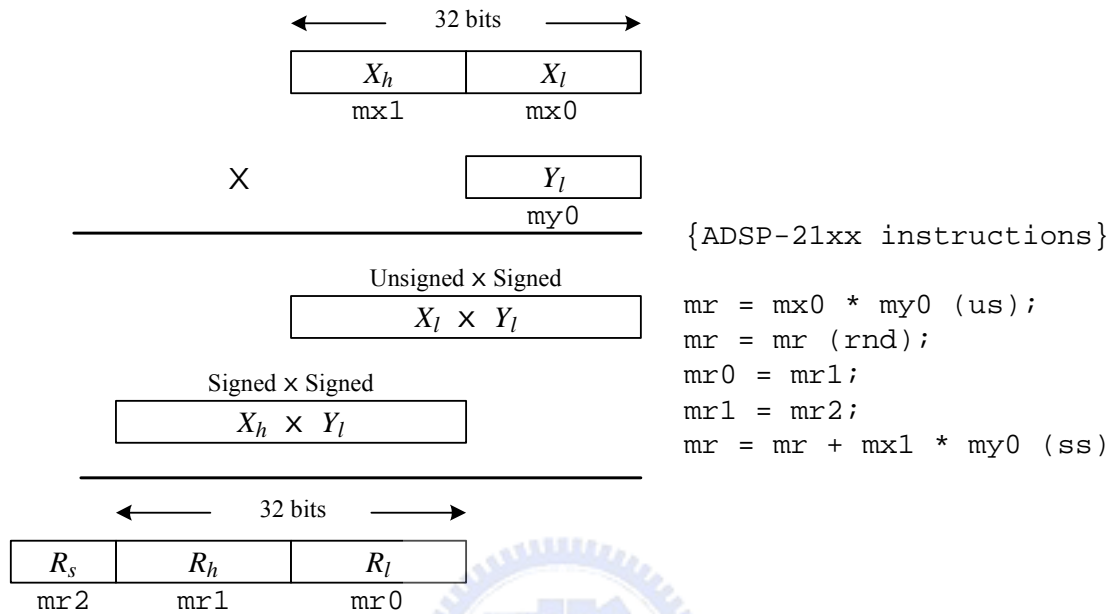


Figure 26. Double precision multiplication, $R(32\text{-bit}) = X(32\text{-bit}) \times Y(16\text{-bit})$.

To determine the data precision, we first divide the encoding processes into six stages as shown in Figure 27. The PCM samples are always 16-bit, and the format is denoted as $(1.15)_{16}$, i.e. the format $(\alpha.\beta)_{\gamma}$ means that a fixed point number of γ bits is represented by lying the binary point just after the α^{th} most significant bit. It is obvious that $\alpha + \beta = \gamma$. The subband analysis has divided into two stages, the windowing with partial calculation [5] and the matrixing [5]. The windowing with partial calculation performs 16-bit multiply-then-accumulate operations and produces 32-bit results vector Y [5]. The matrixing performs double precision multiply-then-accumulate like Figure 26 and produces subband signals S , only the 16-bit rounding result of R_h . According to the static analysis, the dynamic range of subband signals is $-2.0 < S < 2.0$, therefore the format is derived as $(2.14)_{16}$.

Then the subband signals are passed the MDCT and antialias stage. A faster

MDCT algorithm is applied here to decrease computational complexity but also maintain the quantization error due to fixed point arithmetic. After performing 16-bit multiply-then-accumulate operations on subband signals, the 32-bit transformed coefficients are produced and then pass antialias block. Again the double precision arithmetic as Figure 26 is performed, and a antialiased 32-bit transformed coefficients $x_{f,g}(i)$ are produced from R_h and R_l .

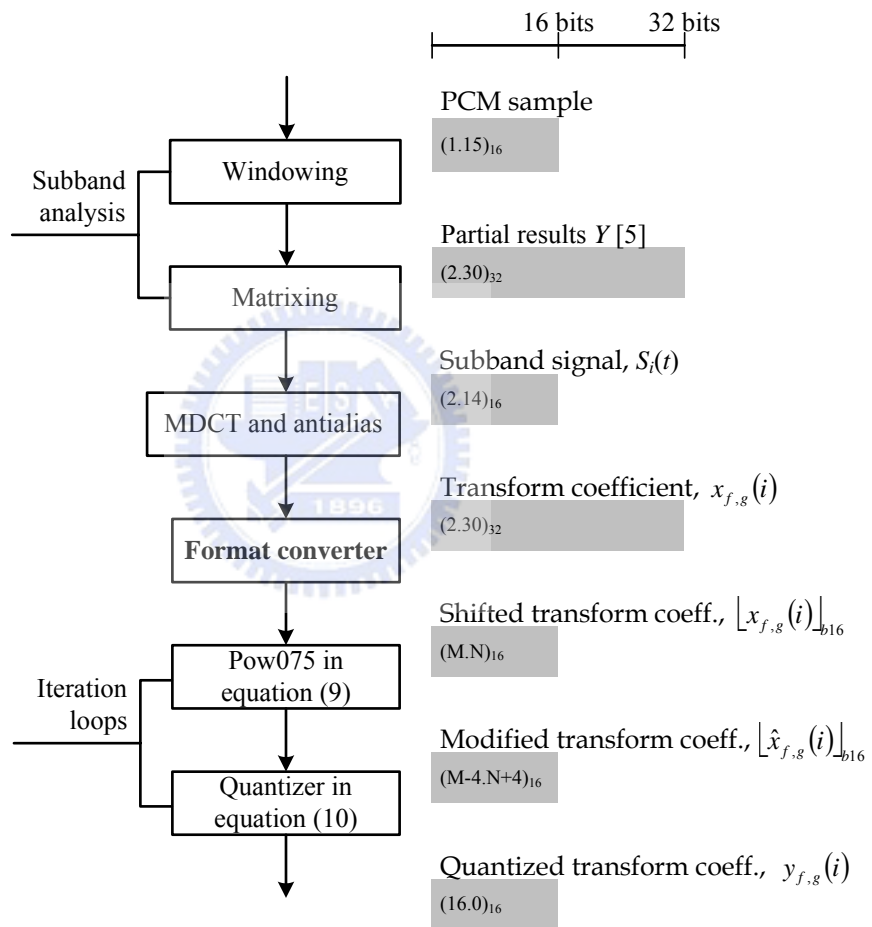


Figure 27. Data precision between each stage in proposed MP3 encoder. $(M.N)_{16}$, determined from the format converter, is the fixed point format of transformed coefficients.

A special format converter added after antialias block is used to convert 32-bit data with format $(2.30)_{32}$ to 16-bit data while the fixed point format is determined

by dynamic range of $x_{f,g}(i)$ at run-time. It first finds the maximum of the transformed coefficients in the granule, $X_{f,g}$. As shown in (28), a right shift amount, k , is derived from a special function, exp_{32} .

$$k = 16 + \text{exp}_{32}(X_{f,g}) \quad (28)$$

The SHIFTER unit of ADSP-21xx core supports hardware exponent detector which can count the number of leading zeros or ones of single precision data in one instruction cycle and double precision data in two instruction cycles. The exponent detector is functionality equal to (29),

$$\begin{aligned} \text{exp}_{16}(x) &\equiv \lfloor \log_2 |x| \rfloor - 14, \quad x \text{ is single precision} \\ \text{exp}_{32}(x) &\equiv \lfloor \log_2 |x| \rfloor - 30, \quad x \text{ is double precision} \end{aligned} \quad (29)$$

For an example, a double precision data,

$$X_{f,g} = (11111111 0abcdefg hijklmno pqrstuvw)_2,$$

the exponent detector produces result of -7. By adding of 16 as (28), k equals to 9 then the format converter shifts the 32-bit transformed coefficients, $x_{f,g}(i)$, right by 9 and produces the 16-bit shifted transformed coefficients as (30),

$$\lfloor x_{f,g}(i) \rfloor_{b16} = x_{f,g}(i) \times 2^{-k}. \quad (30)$$

Meanwhile, the maximum, $X_{f,g}$, is also converted into 16-bit data,

$$\lfloor X_{f,g} \rfloor_{b16} = (10abcdef ghijklmn)_2.$$

The format converter compacts the 32-bit data with $(2.30)_{32}$ into 16-bit data with format $(M.N)_{16}$ optimized for decreasing the quantization error due to fixed point arithmetic. The format $(M.N)_{16}$, different between each encoding granule is

decided at run-time, and M is usually negative. And of course, the shifting operations as (30) will be inverted later.

The shifted transformed coefficients, $\lfloor x_{f,g}(i) \rfloor_{b16}$, are then passed into iteration loops. The proposed algorithm in Figure 14 performs $|x_{f,g}(i)|^{0.75}$ operation before iterative quantization. (31) is rewritten from (9). The multiplying term, 2^4 , converts the format to $(M+4.N-4)_{16}$ because the dynamic range of $|x_{f,g}(i)|^{0.75}$ is one sixteenth of the one of $\lfloor x_{f,g}(i) \rfloor_{b16}$.

$$\begin{aligned} \lfloor \hat{x}_{f,g}(i) \rfloor_{b16} &= \left| \lfloor x_{f,g}(i) \rfloor_{b16} \right|^{0.75} \times 2^4 \\ &= \left| x_{f,g}(i) \times 2^{-k} \right|^{0.75} \times 2^4 \\ &= \left| x_{f,g}(i) \right|^{0.75} \times 2^{-0.75 \cdot k + 4} \end{aligned} \quad (31)$$

The relationship between $\hat{x}_{f,g}(i)$ and $\lfloor \hat{x}_{f,g}(i) \rfloor_{b16}$ can be rewritten as (32),

$$\hat{x}_{f,g}(i) = \lfloor \hat{x}_{f,g}(i) \rfloor_{b16} \times 2^{0.75 \cdot k - 4}. \quad (32)$$

To obtain the correct quantized value, the quantizer is modified from (10). As shown in (33), the modification is done by adding additional offset to exponent term.

$$\begin{aligned} \lfloor y_{f,g}(i) \rfloor_{b16} &= \hat{x}_{f,g}(i) \times 2^{\frac{-3 \times \Delta_{f,g}}{16}} - 0.0946 \\ &= \lfloor \hat{x}_{f,g}(i) \rfloor_{b16} \times 2^{-0.75 \cdot k + 4} \times 2^{\frac{-3 \times \Delta_{f,g}}{16}} - 0.0946 \\ &= \lfloor \hat{x}_{f,g}(i) \rfloor_{b16} \times 2^{\frac{-(3 \times \Delta_{f,g} - 12 \cdot k + 64)}{16}} - 0.0946 \end{aligned} \quad (33)$$

4.3 Data precision optimization in the proposed MP3 decoder

Jeong et al.[11] reveals that there is no audible noise due to fixed point implementation when the MAC based MPEG/audio decoder has at least 21-bit multiplier and 25-bits adder. Lee et al. [6] implements MPEG audio decoding by performing double precision arithmetic during all decoding processes in a 16-bit fixed point DSP.

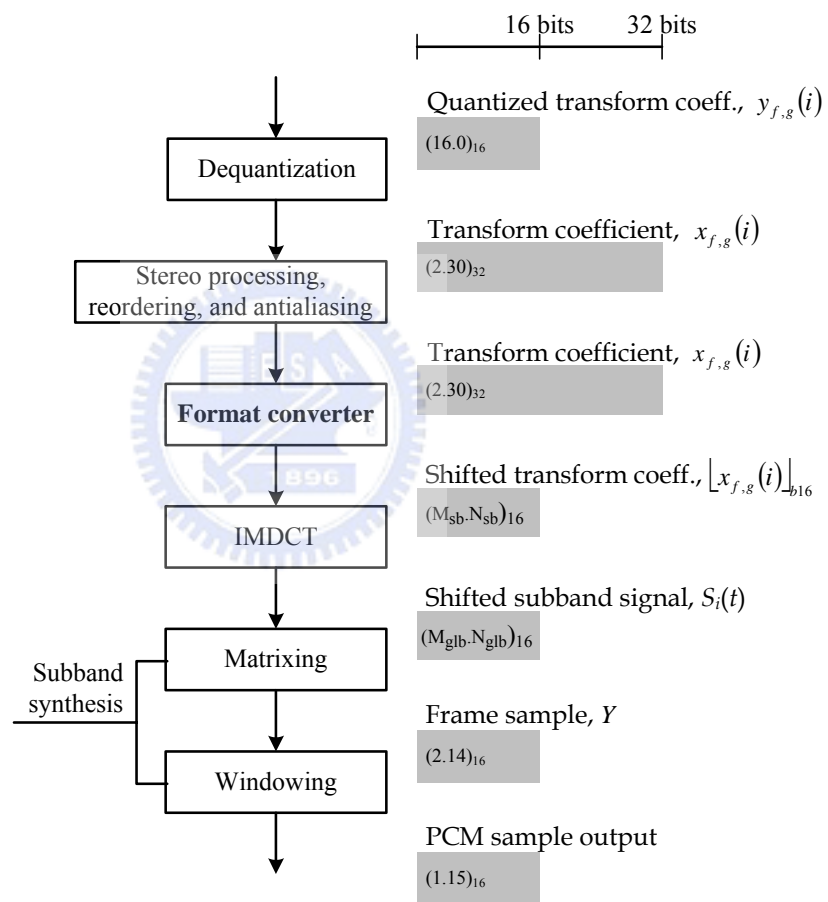


Figure 28. Data precision between each stage in proposed MP3 decoder. $(M_{sb} \cdot N_{sb})_{16}$, determined from the format converter, is the fixed point format of each subband. $(M_{glb} \cdot N_{glb})_{16}$, equal to one of $(M_{sb} \cdot N_{sb})_{16}$ that the subband has the coefficient of highest amplitude in the granule, is the fixed point format of subband signal.

As shown in Figure 28, the quantized transformed coefficients are decoded from Huffman decoder and then dequantized. The optimized dequantizer produces 32-bit data with format $(2.30)_{32}$ and high accuracy as mentioned in Figure 24. Then the succeeding stage including stereo processing, reordering and antialiasing performs double precision arithmetic and produces 32-bit data with the same format $(2.30)_{32}$. The format converter used in encoding is also applied here in decoding.

Different from the encoding case, the format converter converts data format in each subband. By finding the maximum of transformed coefficients in each subband, the individual right shift amount, i.e. format converting parameter, is derived from (28), and the fixed point format of each subband, $(M_{sb}.N_{sb})_{16}$, is determined. As shown in Figure 29, 32 formats denoted as $(M_0.N_0)_{16}$, $(M_1.N_1)_{16}$, ... and $(M_{31}.N_{31})_{16}$ are corresponding to 32 subbands.

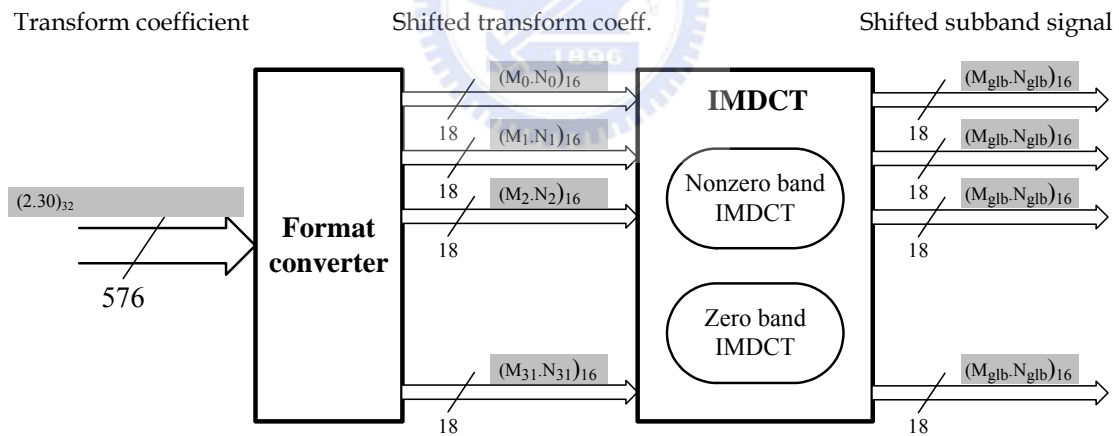


Figure 29. Different format between subbands and the modified IMDCT

A modified scheme of IMDCT is also shown in Figure 29. According to the right shift amount of the subband, the algorithm can divide 32 bands into two groups, nonzero band and zero band. The subband that one of its 18 coefficients is not zero is denoted as nonzero band otherwise zero band. The nonzero band IMDCT performs the 9-point and 3-point Lee's Fast IDCT. But the zero band

performs the simplified IMDCT that the results are produced from overlapped block in the previous granule only.

Each IMDCT performed in each subband will produce 16-bit subband signals with the same format $(M_{\text{glb}}.N_{\text{glb}})_{16}$, derived from (34).

$$\begin{aligned} M_{\text{glb}} &= \max(M_{sb}) \quad , sb = 0 \text{ to } 31 \\ N_{\text{glb}} &= \min(N_{sb}) \quad , sb = 0 \text{ to } 31 \end{aligned} \quad (34)$$

After IMDCT, the subband signals are synthesized to time-domain PCM sample through two operations, matrixing and windowing. The matrixing operation is implemented as the 64-point Lee's Fast DCT. The 16-bit arithmetic is performed, and a 16-bit result vector is produced with format $(2.14)_{16}$. After the windowing operation, the PCM samples with format $(1.15)_{16}$ are produced.



CHAPTER 5. EXPERIMENTAL RESULTS

Using the proposed architecture, the MP3 encoder and decoder are implemented by a 16-bit fixed point DSP, ADSP-2181. Table 5 figures out the superior performance of the proposed MP3 encoder over other commercial products. All MIPS are estimated for 44.1KHz sampled and stereophonic audio input and 128Kbps output MP3 bitstream. Totally we need only about 37.5k bytes program RAM to store both encoder and decoder program code and 27.2k bytes data RAM at most during encoding or decoding. The on-chip RAM of ADSP-2181, 48k and 32k bytes RAM for program and data, is sufficient for the proposed MP3 codec.

The consuming MIPS of each part in proposed MP3 encoder is listed in Table 6 and compared with other works by Oh et al. [4] and Wang et al. [8]. For signal-dependent blocks such as iteration loops and Huffman encoding, the worst-case results are listed. Because of the removal of PAM-II, the proposed new rate control algorithm and non-uniform quantizer, the computational loads of iteration loops in the proposed encoder is much less than that in other two encoders. However, due to the applying of dynamic bit allocation proportional to the energy of granules and the implementation of dynamic data precision, the proposed can also get the similar performance of other two encoders.

Table 5. The implementation result and comparisons with commercial products

Implementation	Processor	MIPS	PM (bytes)	DM (bytes)
Algorithm: MP3 Encoder				
Proposed	ADSP-2181	21.05	16.8k	27.2k
Tensilica [28]	Xtensa HiFi Engine	65	90k	46.6k
ADI, Melody™ chipset [29]	ADSP-218x	40	< 48k	< 32k
CuTe Solutions [31]	ADSP-218x	40	32k	16k
SpiritDSP [30]	MIPS-based TX49xx	80	Not mentioned	Not mentioned
CuTe Solutions [31]	TI C54x	36	22k	21.8k
CuTe Solutions [31]	TI C55x	72	62k	30.3k
CuTe Solutions [31]	TI C64x	33	121k	46.7k
Algorithm: MP3 Decoder				
Proposed	ADSP-2181	17.67	20.7k	23.6k
CuTe Solutions [31]	ADSP-218x	20	33k	17.5k
Nuntius Systems [32]	ADSP-2185	36	25k	23k
Nuntius Systems [32]	Proprietary SIMD DSP core	22	24k	22k
Tensilica [33]	Xtensa HiFi Engine	18	37k	27.3k
SpiritDSP [30]	TI C55x	12.5	20k	12k
CuTe Solutions [31]	TI C54x	31	29.7k	14.2k
CuTe Solutions [31]	TI C64x	20	82k	33.2k
SpiritDSP [30]	ARM7	25	31k	24k

Table 6. The comparison of peak consumed MIPS in different MP3 encoder

Peak MIPS	Proposed encoder	Oh et al. encoder [4]	Wang et al. encoder [8]
Subband analysis	7.09	10.4	5.64
MDCT	3.99		3.74
PAM-II	Not presented	Not presented	8.96
Iteration loops	4.50 (peak)	18.43 (peak)	11.87 (peak)
Huffman encoding and bitstream formatting	5.47 (peak)	2.07 (peak)	5.86 (peak)
Total	21.05	30.9	36.07

Table 7 lists the consuming MIPS of each part in proposed MP3 decoder and comparison with other works by Lee et al. [6] and Bang et al. [10]. Lee et al. implemented the MP3 decoder on Motorola DSP56654, a dual-core processor with a 32-bit RISC MCU and a 16-bit fixed point DSP. Bang et al. realized it on a self-design VLSI of 20-bit fixed point DSP core with hardware Huffman decoder.

Table 7. The comparison of peak consumed MIPS in different MP3 decoder

Peak MIPS	Proposed decoder	Lee et al. decoder [6]	Bang et al. decoder [10]
Synchronization and bitstream unpacking	0.44	6.2 (peak)	NA
Scalefactor and Huffman decoding	5.95 (peak)		
Dequantization	2.38 (peak)	5.4 (peak)	4.51 (peak)
IMDCT	4.45 (peak)	2.8	2.85
Subband synthesis	4.45	6.3	5.97
Total	17.67	20.7	13.33

To evaluate the audio quality of the proposed MP3 encoder and decoder, the subjective evaluation is applied via “Double blind triple stimulus with hidden reference” listening tests [12]. Three different audio samples as summarized in Table 8 are used in this experiment. All samples are stereophonic and sampled with 44.1KHz. Eleven listeners are involved in the experiments.

Table 8. Test audio samples

Signal characteristic	Time	Abbreviation
Violin solo in arpeggio [13]	0:37	VL
Melodious quartet [13]	0:28	QT
German female speech [13]	0:21	GF

The reference codec is the traditional MP3 encoder and decoder with ISO method implemented in floating-point. The “Diffgrade” and “number of misidentification items” are presented in three tests. Diffgrade is the subjective rating given to coded test item minus the rating given to the hidden reference.

Table 9. The subjective evaluation results (1), DG: Diffgrade. MI: Number of misidentification over 11 listeners. The diffgrade scale is partitioned into five ranges: “imperceptible (>0.00)”, “perceptible but not annoying ($0.00 \sim -1.00$)”, “slight annoying ($-1.00 \sim -2.00$)”, “annoying ($-2.00 \sim -3.00$)” and “very annoying ($-3.00 \sim -4.00$)”. The “number of misidentification” represents the number of subjects that incorrectly identified test item and hidden reference.

Proposed encoder / ISO decoder				
Bit rate		VL	QT	GF
192Kbps	DG	-0.04	0.02	0.2
	MI	7	7	10
128Kbps	DG	-0.2	-0.3	0.04
	MI	6	6	9
96Kbps	DG	-0.7	-0.55	-0.46
	MI	3	4	6

Table 9 shows the results of MP3 encoded in proposed encoder and decoded in ISO decoder. Table 10 shows the results of MP3 encoded in ISO encoder and decoded in proposed decoder. Table 11 shows the results of MP3 encoded in proposed encoder and decoded in proposed decoder.

Table 10. The subjective evaluation results (2)

ISO encoder / Proposed decoder				
Bit rate		VL	QT	GF
192Kbps	DG	-0.02	-0.02	0.01
	MI	9	8	9
128Kbps	DG	-0.1	-0.04	-0.04
	MI	8	8	9
96Kbps	DG	-0.1	-0.04	-0.06
	MI	9	9	9

Table 11. The subjective evaluation results (3)

Proposed encoder / Proposed decoder				
Bit rate		VL	QT	GF
192Kbps	DG	-0.25	-0.4	0.0
	MI	6	7	7
128Kbps	DG	-0.7	-0.5	-0.5
	MI	3	3	6
96Kbps	DG	-1.02	-0.8	-0.6
	MI	2	3	5

CHAPTER 6. DUAL CORE EMBEDDED SYSTEM

6.1 System Overview

In this thesis, the DSP implementation of MP3 codec is integrated into a host system. The host system, AdvanTech PCM-7130 SBC (Single board computer) [16], based on a 32-bit RISC, Intel® StrongARM SA-1110, supports various kind of peripherals such as USB, CF, Ethernet and etc. The DSP system is on the development board of ADSP-2181 DSP, ADI ADSP-2181 EZ-LAB. Figure 30 shows the architecture of the dual core embedded system. Section 6.2 will introduce the two subsystems and the design of hardware adapter.

The interprocessor communication is done through a set of memory-mapped mailbox register in the DSP's internal memory. As shown in Figure 30, the ADSP-2181 IDMA port is adopted as the communication channel. There are several advantages of adopting IDMA port:

- The read/ write access of IDMA is completely asynchronous. It simplifies the interprocessor design since we don't need to build a synchronous channel between two different processors.
- The host can access the DSP's internal memory with an overhead of only one DSP processor cycle per word while the DSP is operating at full

speed. Thus DSP's internal memory becomes the shared memory between two processors.

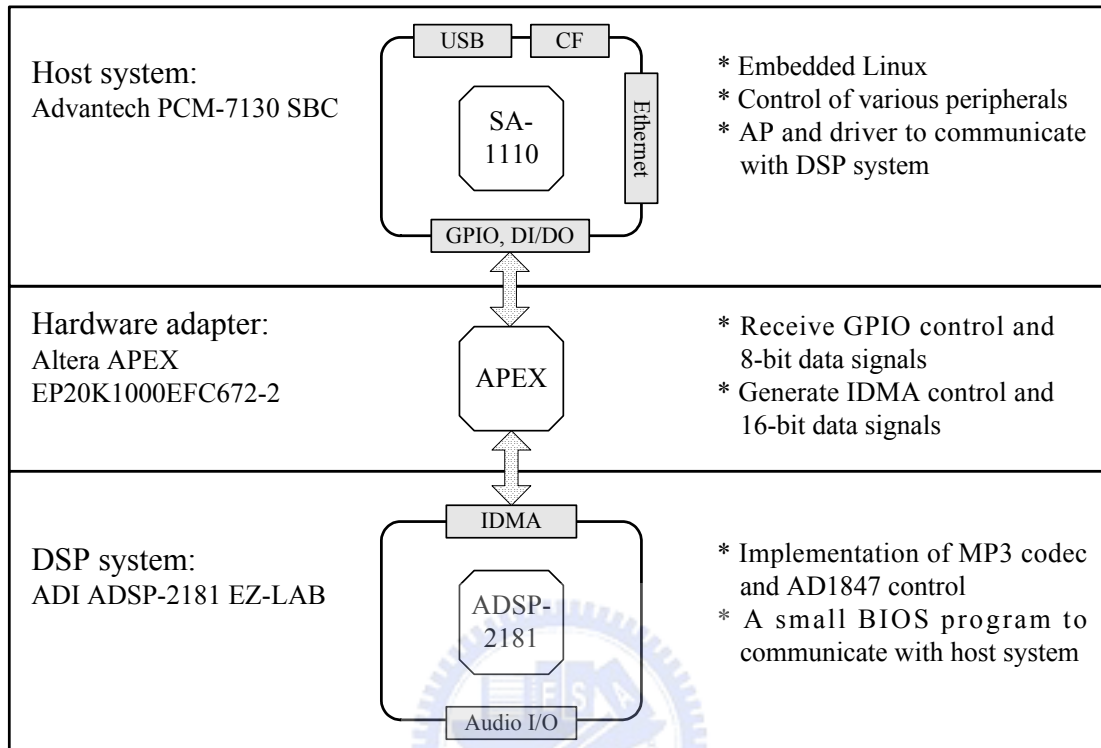


Figure 30. The dual core embedded system

With the help of firmware protocol, the host can download program at run-time, instruct DSP to execute program-dependent operations and fill (or take out) the input (or output) of the operations to (or from) the shared memory area based on pre-defined rules.

The firmware protocol is designed to best suite for real-time processing of audio codec and will be described in section 6.3.

6.2 Hardware Platform

6.2.1 Host system – AdvanTech PCM-7130 SBC

The PCM-7130 is an Intel® StrongARM low-power RISC processor single

board computer that is designed to serve power/ environment critical applications. The features are as following [16]:

- Ultra-compact size SBC as small as a 3.5" hard disk drive (145 mm x 102 mm)
- On-board Intel StrongARM SA-1110 CPU operating at 206MHz
- 64MB system memory on board (SDRAM)
- 32MB flash memory on board
- One 10Base-T Ethernet port
- Two RS-232 ports and one RS-485 port
- One USB host and one USB client ports
- One mini-DIN PS/2 port for keyboard and mouse
- AC'97 audio interface and a buzzer
- One VGA output for CRT monitor
- 18-bit TFT active color LCD interface
- One CompactFlash slot
- One PCMCIA slot
- One IrDA interface
- 8 GPIO, 8 digital input and 8 digital output interfaces (3.3V)
- 4-wire resistive touchscreen interface
- Smart battery interface
- One TV-out supporting both NTSC and PAL signals

Figure 31 is the top view of PCM-7130 SBC, and the peripheral interfaces are also shown. The expansion bus is directly connected to the system bus of SA-1110 and is the best choice to connect memory mapped peripherals, like ADSP-2181 IDMA port. But the B2B connector is proprietary and not available. Another proper choice is the combination of GPIOs, DIs and DOs. The 8 of 26 GPIOs are not used in the design of SBC and available for connection to other peripherals. We take the 8 GPIOs as duplex and bidirectional address/ data bus

and 1 DI and 5 DOs as the control bus as shown in Figure 33. Since the width of data bus are not compatible to IDMA port an additional hardware adapter is also needed.

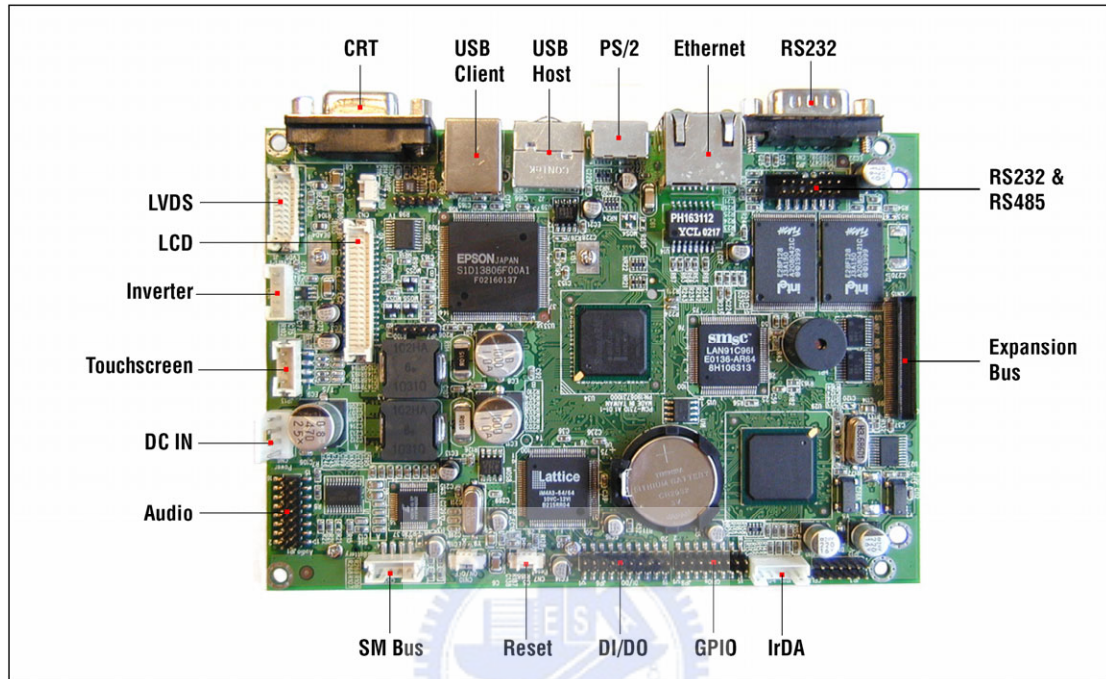


Figure 31. PCM-7130 SBC [15]

6.2.2 DSP system – ADI ADSP-2181 EZ-LAB

The ADSP-2181 EZ-LAB evaluation board is an example of minimum implementation of an ADSP-2181 processor [18]. The specifications are as following [18]:

- ADSP-2181KS-133 DSP operating at an instruction rate of 33M Hz (16M Hz external clock)
- AD1847 SoundPort® stereo codec including a stereo pair of $\Sigma\Delta$ ADCs and a stereo pair of $\Sigma\Delta$ DACs.
- One stereo pair of 2V RMS AC coupled line level inputs and one stereo pair of 20mV RMS AC coupled microphone inputs

- One stereo pair of 1V RMS AC coupled line level outputs
- A EPROM socket to accept EPROMs from 256K bits up to 8M bits used in booting DSP when reset is deasserted
- One RS-232 port



Figure 32. ADI ADSP-2181 EZ-LAB evaluation board

6.2.3 Design of hardware adapter



Figure 33. Functional diagram of hardware adapter

The hardware adapter here is a bridge to connect host (GPIO and DI/DO ports) and DSP (IDMA port). Figure 33 shows the functional diagram. In this thesis, a programmable logic device is used to complete the design, and the Altera APEX™ II FPGA (EP20K1000EFC672-2) is adopted as the bus master of ADSP-2181 IDMA port.

Table 12 and 13 explains the pin functions of host and DSP port.

Table 12. Host port pins

<u>Pin Name(s)</u>	<u>Input/ Output</u>	<u>Function</u>
SA_nRST	I	Reset signal of FPGA
SA_nAW	I	Address write strobe
SA_nDW	I	Data write strobe
SA_nDR	I	Data read strobe
SA_nWrite	I	Tri-state enable signal of SA_AD
SA_nWait	O	Acknowledge signal
SA_AD[0:7]	I/O	Bidirectional address/ data bus

Table 13. ADSP-2181 IDMA port pins

<u>Pin Name(s)</u>	<u>Input/ Output</u>	<u>Function</u>
DSP_nIS	O	Port select signal
DSP_IAL	O	Address latch enable
DSP_IACK	I	Access ready acknowledge
DSP_nIWR	O	Data write strobe
DSP_nIRD	O	Data read strobe
DSP_IAD[0:15]	I/O	Bidirectional address/ data bus

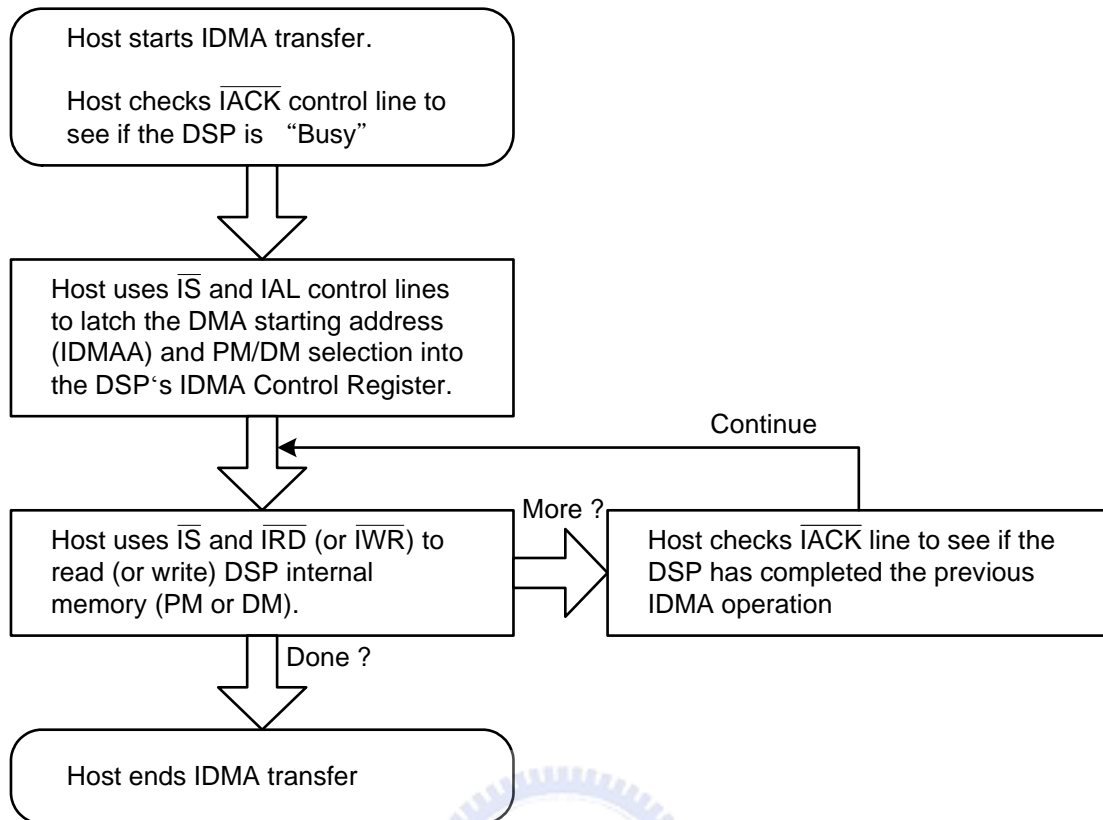


Figure 34. General IDMA transfer protocol [17]

Figure 34 shows the general IDMA transfer protocol. Bus master generates three types of signaling to complete the IDMA write (or read) operations. As shown as Figure 35, the signaling are as following:

- **Set Address:** the FPGA receives two assertions of SA_nAW and latches low byte and high byte of starting address internally. In the second assertion, the FPGA will generate the access timing, putting the starting address on the DSP_IAD port and asserting DSP_nIS and DSP_IAL that the DSP will latch the data on the DSP_IAD port into the IDMA, DMA starting address register.
- **Write Memory:** the FPGA receives two assertions of SA_nDW and latches low byte and high byte of writing data internally. In the second assertion, the FPGA will generate the access timing and handshake with

DSP, putting the data on the DSP_IAD port and asserting DSP_nIS and DSP_nIWR that the DSP will write the data on the DSP_IAD port into the internal memory located by the IDMAA. The DSP will automatically increment the value in IDMAA after each memory access, writing or reading, that the host do not need to update the IDMAA again in memory access of consecutive location.

- Read Memory:** the FPGA receives two assertions of SA_nDR and will generate the access timing, asserting DSP_nIS and DSP_nIRD and then latching the data on the DSP_IAD port internally after the deassertion of DSP_IACK, in the first assertion of SA_nDR. Host then deasserts SA_nDR twice to latch low byte and high byte of data individually.

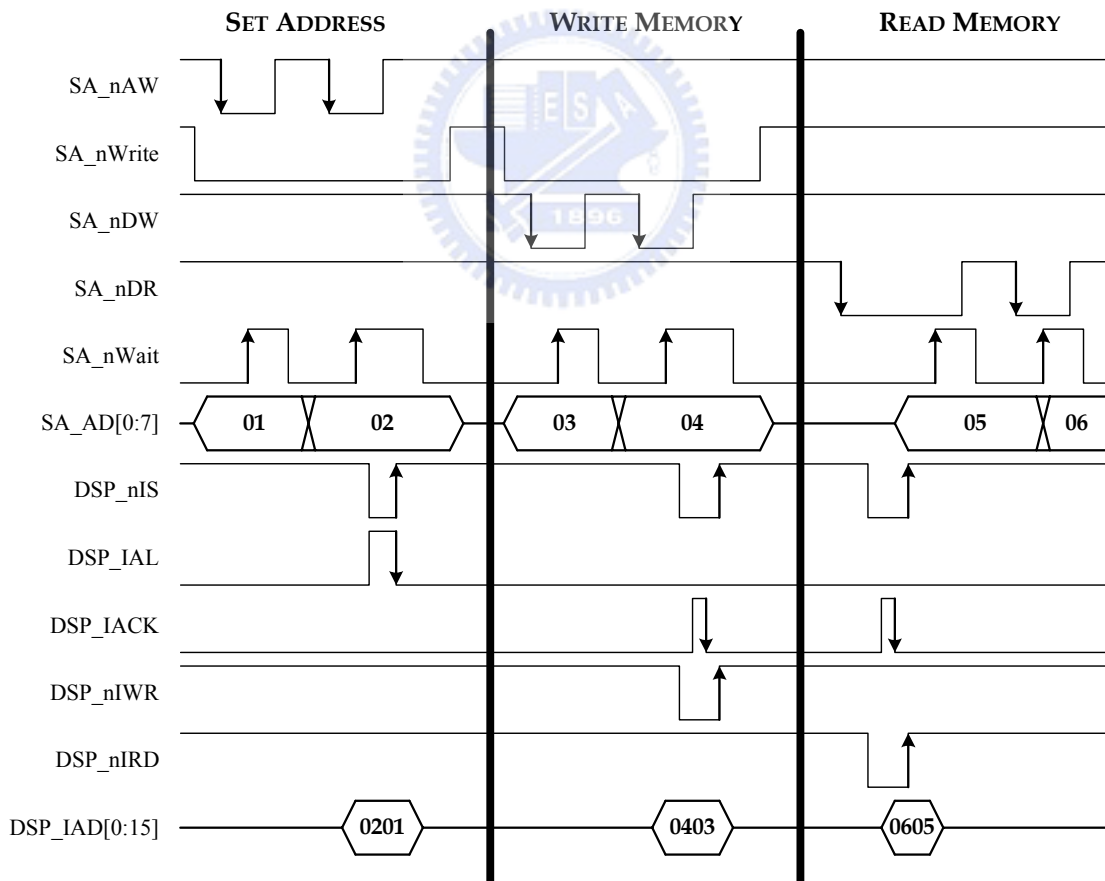


Figure 35. Port access timing

6.3 Firmware Design

Figure 36 shows the hierarchical view of software, firmware and hardware layer in this system. The implementation of MP3 programs, software part in DSP system, has been described in Chapter 4. The hardware functionality, communication ports, and the self-design adapter are also introduced in section 6.2.

In this thesis, the host system has an interactive GUI on the external display device and can be controlled through touchpanel by human user. So the computing power of the host system will be dominated in the handling of GUI. We design the GUI by writing QT application on Linux. QT [27], product of Trolltech, is a complete C++ application development framework and includes a class library and tools for cross-platform development and internationalization. Beside the GUI handling, we also integrate the host programs used in communication with DSP system into QT application. The host programs talk to DSP in some firmware protocol described later in section 6.3.2. In the DSP side, a self-design firmware called DSP BIOS is the housekeeper of DSP used to implement the protocol and manage the resource.

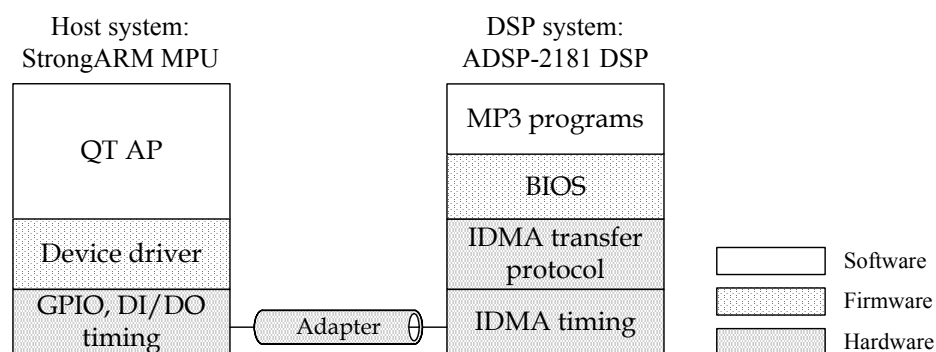


Figure 36. The hierarchical view of software, firmware and hardware layer

6.3.1 Linux Character Device Driver

In the memory management of Linux, the kernel and the user process can

only access its own memory space. And for safety issue, the user process can't directly access neither memory mapped nor I/O mapped hardware devices.

The device driver provides a standard way for user process to access the hardware devices without knowing how they work. All version of UNIX have an abstract way to make devices logically work as well as regular files. Therefore, the same calls (read(), write() and etc.) can be used for devices and files [19]. There are two main types of devices under all UNIX systems, character and block devices. Character devices are those for which no buffering is performed, and block devices are those which are accessed through a cache.

In this thesis, a character device driver is designed to generate access timing of the host side as shown in Figure 35. Three basic operations (set address, write memory and read memory) are performed through calls of read() and write() from user process. The essential parts of writing driver are registering file operation structure (actually the entry points of routines) as below and implementing the routines that user process will invoke. The device driver is also part of Linux kernel so the kernel needs to be re-compiled after adding a new device driver.

```
/* File operation structure used in Linux kernel */
struct file_operations mw_fops = {
    owner:        THIS_MODULE,
    read:         read_mw,
    write:        write_mw,
    open:         open_mw,
    release:      release_mw,
};
```


The C code below represents the device handling in user programs.

- Open the device

```

/* Open device */
int OpenDevice(){
    int fd;
    fd = open(DEVICE_NAMES, O_RDWR);
    return(fd);
}

```

- Operation of "Set Address"

```

/* Set address */
void SetAddress(int fd, unsigned short address){
    unsigned short buf[2];
    buf[0] = address;
    buf[1] = 1; // OP code of write address
    write(fd, buf, 4);
}

```

- Operation of "Write Memory"

```

/* Write memory */
void WriteMemory(int fd, unsigned short data){
    unsigned short buf[2];
    buf[0] = data;
    buf[1] = 0; // OP code of write data
    write(fd, buf, 4);
}

```

- Operation of "Read Memory"

```

/* Read memory */
unsigned short ReadMemory(int fd){
    unsigned short buf;
    read(fd, &buf, 2);
    return(buf);
}

```

- Close the device

```

/* Close device */
void CloseDevice(int fd){
    close(fd);
}

```

6.3.2 DSP BIOS

In this thesis, the DSP BIOS is burned into flash EPROM, and the DSP will automatically load it after hardware reset is deasserted. The BIOS program is divided into two part, booting and housekeeping. The booting code is a series of instructions to initial the DSP to ready-to-run state and will run only once after BIOS is re-loaded.

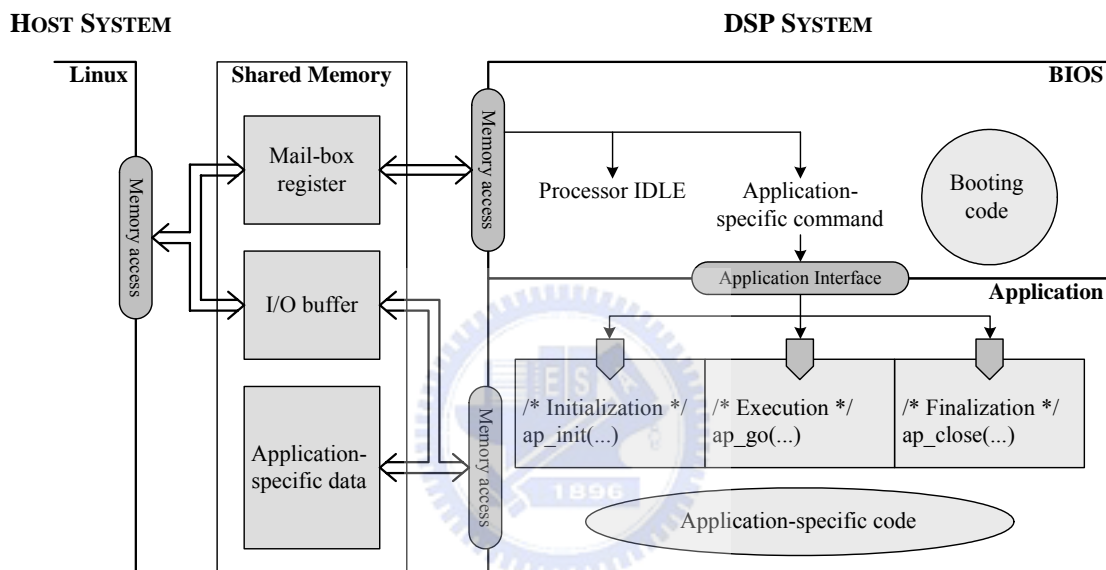


Figure 37. The firmware block diagram

After booting, the housekeeping code starts to run. The housekeeper function is implemented as a never-ending, `while(1)`, loop. The host sends command to mail-box, and the housekeeper receives it. After the commands is completed, the housekeeper will respond the status and then continue to receive next commands.

There are two types of commands, IDLE and application-specific. The IDLE command instructs the DSP entering the power-saving mode. That keeps the processor fully functional, but operating at the slower clock rate. The host will send IDLE command for two purpose. One is when no task is needed to be

performed on DSP. Another is when the host want to load a new application program the IDLE command let the DSP lock in the housekeeping loop that it guarantees the safety program loading into application section.

The application-specific command instructs the housekeeper to call one of the application-specific routines. The host needs to write `ap_id` (application identifier) in mail-box and register the entry point of the application-specific routines after the program is successfully loading into application section. At least three application-specific routines, `ap_init()`, `ap_go()`, and `ap_close()` are necessary to perform tasks.

The application developer can use `ap_init()` routine to run the initialization, like clearing the application-specific data, initializing the stereo codec (AD1847) and etc. In contrast, `ap_close()` routine is used to finalize the task if necessary.

The `ap_go()` routine suits for application of processing frame-by-frame, like audio encoding or decoding. If the host needs faster responding time the application developer may need to implement the algorithm in pipeline way.

CHAPTER 7. CONCLUSIONS AND FUTURE WORKS

7.1 Conclusions

In this thesis, we give:

- A brief introduction to MPEG/ Audio Layer III coding algorithm,
- A proposed fast algorithm of MP3 encoding including the removal of PAM-II, the simplification of PAM-II related process, a fast quantization method by applying polynomial approximation and a new fast bit allocation algorithm,
- A proposed fast algorithm of MP3 decoding including a fast dequantization method by applying polynomial approximation and the Newton's method for root-finding and applying fast DCT/IDCT algorithm in IMDCT and subband synthesis process.

and also present:

- The performance analysis and subjective test of sound quality of the proposed algorithm,
- A Real-time implementation of proposed MP3 encoding and decoding algorithm in a 16-bit fixed point DSP, ADSP-2181,
- Applying dynamic fixed point format to optimize the data precision of each process in the target DSP of 16-bit word-length only,

- Comparison of DSP MIPS and memory usage with other implementation.

The 16-bit fixed point implementation with the proposed optimization algorithm only needs 21.05 DSP MIPS for MP3 encoding and 17.67 for decoding. Compared to other pure software DSP implementation, both of the proposed encoding and decoding algorithm are the fastest. The memory usage (16.8KB PM/ 27.2KB DM for encoder and 20.7KB PM/ 23.6KB DM for decoder) can also meet the requirement of the target DSP, 48KB PM/ 32KB DM.

And in Chapter 7, the DSP implementation of an MP3 codec is also applied in an host system based on a 32-bit RISC processor, Intel® StrongARM SA-1110. We also present a complete design of the dual core embedded system including:

- A hardware adapter realized by VHDL on an Altera APEX™ II FPGA for translating the bus timing of of two subsystems, host and DSP,
- A Linux character device driver generating host bus timing,
- A firmware protocol designed for interprocessor communication, the corresponding implementation on DSP, also called DSP BIOS, and host software which is integrated to a QT GUI application.

7.2 Future Works

The population of MP3 brings not only the low cost and convenience digital audio to the world but also the rapid growth of advanced audio compression knowledge in recent ten years. Many other audio compression formats are developed and realized like Dolby AC-3 [20], MPEG-2/4 AAC [21], Microsoft® WMA [22], Coding Technologies mp3PRO and aacPlus [23], and Ogg Vorbis [24]. Compared to MP3, they afford better audio quality and lower bit rate. But their computational complexities are also higher. We may apply the concepts of the proposed algorithm in this thesis on these modern audio compression algorithm and

CHAPTER 7. CONCLUSIONS AND FUTURE WORKS

decrease the computational complexity that the algorithm can be realized on low cost fixed point DSP. Applying to the dual core embedded system in this thesis, we can also develop multi-format codec by loading different implementation file at run-time.

The industrial trend of SOC (System-On-a-Chip) recently brings the system with the smaller board size, the lower manufacturing cost, the lower power consumption and the best performance. The dual core design in this thesis can be realized in SOC way, too. We may integrate MPU IP and DSP IP with some peripheral controller, memory and associated hardware accelerators in a chip. Thus the DSP unit becomes a coprocessor of MPU used to execute numeric processing with high complexity.



REFERENCE

- [1]. E. Zwicker and H. Fastl, “*Psychoacoustics: facts and models*,” Springer-Verlag, Berlin, Heidelberg, Spring, 1999.
- [2]. D. Pan, “A tutorial on MPEG/audio compression,” *IEEE Multimedia*, vol.2, no.2, pp.60-74, 1995.
- [3]. Peter Noll, “MPEG digital audio processing,” *IEEE Signal Processing Magazine*, pp.59-81, September 1997.
- [4]. H. Oh, J. Kim, C. Song, Y. Park and D. Youn, “Low power MPEG/audio encoders using simplified psychoacoustics model and fast bit allocation,” *IEEE Transaction on Consumer Electronics*, vol.47, no.3, August 2001.
- [5]. ISO/IEC JTC1/SC29/WG11 MPEG, International Standard IS 11172-3, “Coding of moving pictures and associated audio for digital storage media at up to about 1.5M bit/s, part 3: audio,” 1993.
- [6]. Keun-Sup Lee, Hyen-O Oh, Young-Cheol Park, and Dae Hee Youn, “High quality MPEG-audio Layer III algorithm for a 16-bit DSP,” in *Proceeding of IEEE International Symposium on Circuit and Systems*, vol. II, pp.205-208, Sydney, Australia, May 6-9, 2001.
- [7]. Byeong Gi Lee, “A new algorithm to compute the discrete cosine transform,” *IEEE Trans. On Acoustic, Speech and Signal Processing*, vol. ASSP-32, no.6, pp.1243-1245, 1984.
- [8]. Xin Wang, Weibei Dou and Zhaorong Hou, “An improved audio encoding architecture based on 16-Bit fixed-point DSP,” *IEEE International Conference*

REFERENCE

- of Communications, Circuits and Systems 2002 (ICCCAS'02)*, vol.2, pp.918 - 921, June 29 - July 1, 2002.
- [9]. Analog Devices: OEM Solutions: Market Solutions: MPEG-1 Layer III. [Online]. Available: http://www.analog.com/Analog_Root/static/marketSolutions/oem/audio/mpeg1_3decoder.html
- [10]. Kyoung Ho Bang, Nam Hun Jeong, Joon Seok Lim, Young Cheol Park, and Dae He Youn, "Design and VLSI implementation of a digital audio-specific DSP core for MP3/ AAC," in *Proceeding of International Conference on Consumer Electronics*, Los Angles, pp. 790-795, June 18-20, 2002.
- [11]. Min-seep Jeong, Seehyun Kim, Jongseo Sohn, and Ji-Yang Kang, "Finite Wordlength Effects Evaluation of the MPEG-2 Audio Decoder," *International Conference on Signal Processing Applications & Technology*, pp.351-355, January. 1996.
- [12]. ITU-R Rec. BS.1116, "Methods for the subjective assessment of small impairment in audio systems including multichannel sound systems," October, 1997.
- [13]. SQAM - Sound Quality Assessment Material: EBU SQAM disc tracks. [Online]. Available: <http://www.tnt.uni-hannover.de/project/mpeg/audio/sqam/>
- [14]. Sourceforge project: LAME Aint an MP3 Encoder (LAME). [Online]. Available: <http://sourceforge.net/projects/lame/>
- [15]. AdvanTech, Inc., "AdvanTech PCM-7130 datasheet."
- [16]. AdvanTech, Inc., "AdvanTech PCM-7130 user manual."
- [17]. Analog Devices, Inc., "ADSP-2100 family user's manual."

REFERENCE

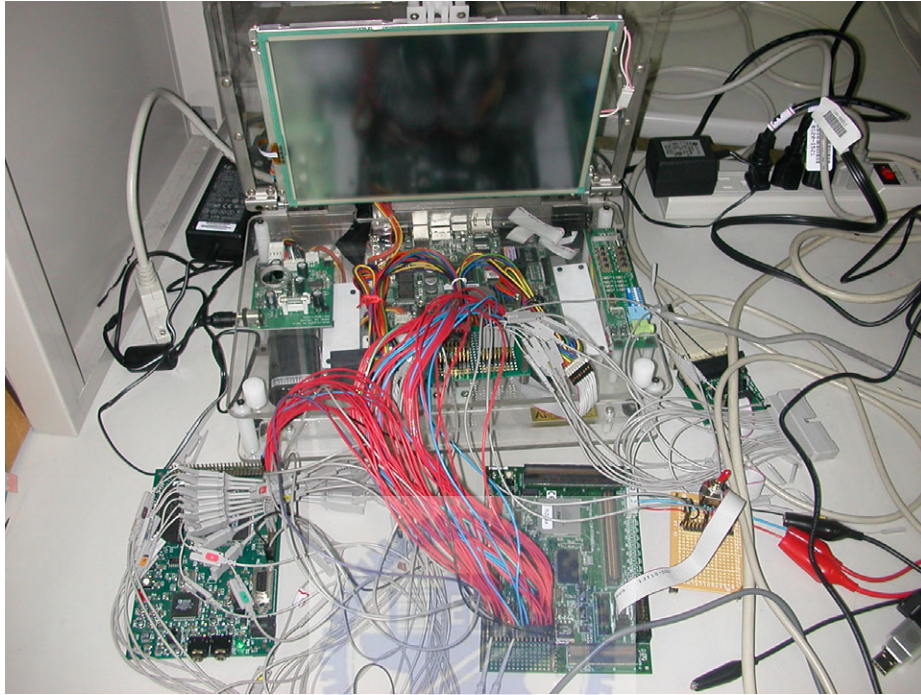
- [18]. Analog Devices, Inc., “*ADSP-2100 family EZ-KIT Lite reference manual.*”
- [19]. Linux Kernel Hackers' Guide: Device Drivers. [Online]. Available:
<http://en.tldp.org/LDP/khg/HyperNews/get/devices/devices.html>
- [20]. C. Todd, et. Al., “AC-3: Flexible Perceptual Coding for Audio Transmission and Storage,” *AES 96th Convention*, Preprint 3796, Audio Engineering Society, New York, N.Y., February 1994.
- [21]. ISO/IEC 13818-7, “Information technology – generic coding of moving pictures and associated audio Information, part 7: Advanced Audio Coding,” 1997.
- [22]. Microsoft: Windows Media Player Multimedia File Format. [Online]. Available:
<http://support.microsoft.com/default.aspx?scid=kb:zh-tw:316992>
- [23]. Coding Technologies: products and Technologies. [Online]. Available:
<http://www.codingtechnologies.com/products/index.htm>
- [24]. Vorbis.com – Open, Free Audio. [Online] . Available: <http://www.vorbis.com/>
- [25]. Fact Index: Introduction to psychoacoustics . [Online]. Available:
<http://www.fact-index.com/p/ps/psychoacoustics.html>
- [26]. Hung-Chih Lai, “Real-time implementation of MPEG-1 Layer 3 audio decoder on a DSP chip,” Master thesis submitted to department of Electrical and Control Engineering, National Chiao Tung University, June 2001.
- [27]. TrollTech Inc.: product: QT overview. [Online]. Available:
<http://www.trolltech.com/products/qt/index.html>
- [28]. Tensilica’s MP3 Encoder Application Package. [Online]. Available:

REFERENCE

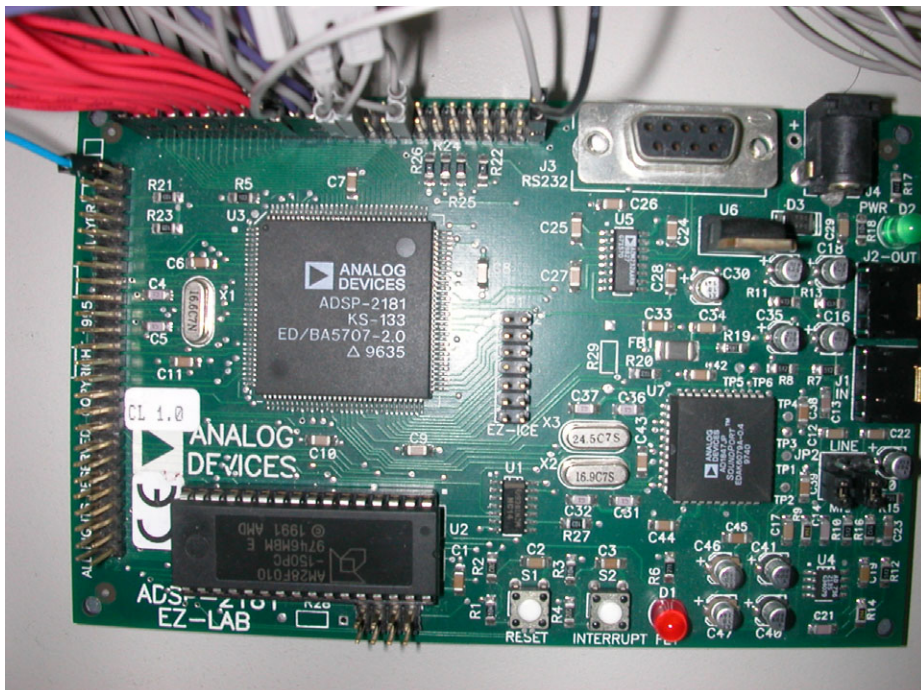
- http://www.tensilica.com/html/mp3_encoder.html
- [29]. Futurtec News: Analog Devices Releases New MP3 Chip. [Online]. Available: <http://www.futurlec.com/News/Analog/MP3.html>
- [30]. Spirit Corp.: Products: Audio/Video Processing Overview. [Online]. Available: http://www.spiritdsp.com/audio_processing.html
- [31]. CuTe Solutions Inc., Audio solutions on Analog Device Inc. ADSP-218x DSP Devices and Texas Instruments Inc. TI C54x, C55x and C64x Processors. [Online]. Available: <http://www.cutesolinc.com>
- [32]. Nuntius Systems Inc., Multimedia – Streaming audio CODECs. [Online]. Available: <http://www.nuntius.com/solutions31.html#mp3>
- [33]. Tensilica's MP3 Decoder Application Package. [Online]. Available: http://www.tensilica.com/html/mp3_decoder.html

APPENDIX

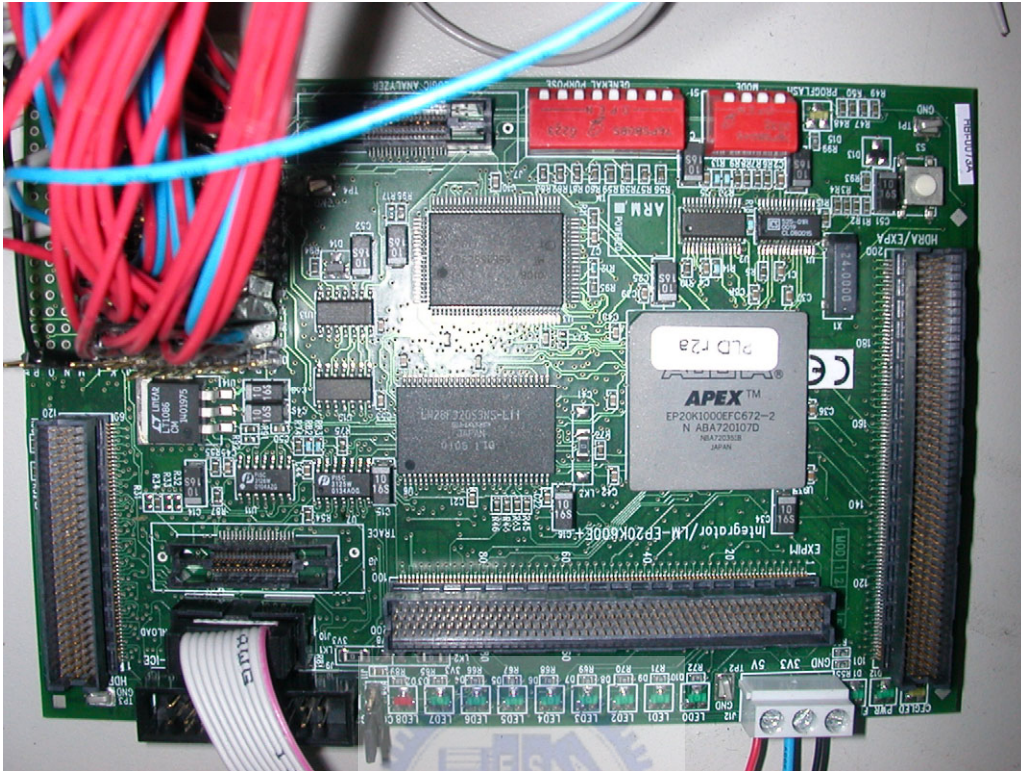
系統實照



ADSP-2181 發展板，ADI ADSP-2181 EZ-LAB



匯流排轉換電路板，Altera APEX™ II FPGA (EP20K1000EFC672-2)



Intel® StrongARM SA-1110 發展板，Advantech PCM-7130 SBC

