

國立交通大學

電機與控制工程學系

碩士論文

MPEG-2/4 低複雜度先進音訊編碼最佳化及
雙核心處理器實現

MPEG-2/4 LOW-COMPLEXITY ADVANCED
AUDIO CODING OPTIMIZATION AND
IMPLEMENTATION ON DUAL-CORE PROCESSOR

研究生：黃嘉雄

指導教授：吳炳飛 教授

中華民國 九十五年 七月

MPEG-2/4 低複雜度先進音訊編碼最佳化及 雙核心處理器實現

研究生：黃嘉雄

Student : Jia-Hsiung Huang

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 and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control Engineering

July 2006

Hsinchu, Taiwan, Republic of China

中華民國 九十五年七月

MPEG-2/4 低複雜度先進音訊編碼最佳化及 雙核心處理器實現


研究生：黃嘉雄

指導教授：吳炳飛

國立交通大學

電機與控制工程學系 碩士班

摘要



本篇論文主要針對 AAC 編碼，提出一套最佳化演算法，使得運算複雜度與記憶體需求皆能夠降低以適於在行動裝置上的實現。在 AAC 編碼最佳化中，我們移除了運算複雜度高的長短窗切換判斷，採取了簡化的聲響心理模型。此外，在 MDCT 的轉換上，套用了以 FFT 為運算核心的時頻轉換，並精簡了 FFT 運算過程中所需的記憶體；於 TNS 與立體聲編碼上，亦以較為精簡的判斷來降低運算量。最後，為了減少記憶體使用，基於統計結果，我們化簡了做 Huffman 編碼時所需要的編碼表。

基於 16 位元的定點數 DSP，我們於能量計算上採用動態格式的調整，以避免定點數運算超出所能表示的數值。我們將所提出的 AAC 編碼器實現於一個雙核心的處理器上，並於雙核心處理器上實現兩種不同的軟體架構以達到錄音的功能。實現的 DSP-based AAC 編碼器需要 86MIPS 及 107KB 的記憶體而實現的 ARM-based 錄音器可達到 1X 速的壓縮。

MPEG-2/4 LOW-COMPLEXITY ADVANCED AUDIO CODING OPTIMIZATION AND IMPLEMENTATION ON DUAL-CORE PROCESSOR

Student : Jia-Hsiung Huang

Advisor : Prof. Bing-Fei Wu

Department of Electrical and Control Engineering

National Chiao Tung University

ABSTRACT



In this thesis, several optimized techniques in the AAC encoding process are presented in order to lower down the computational complexity and required memory. The decision of block switching is removed, and adopts a simplified psychoacoustic model. For the MDCT transformation, the fast MDCT with FFT as kernel computation is applied. Moreover, the memory requirement while performing FFT processing is reduced. Other modules, such as TNS, Mid/Side Stereo coding are also simplified. In order to minimize the memory usage, the Huffman tables are reduced base on statistics.

So as to make sure the avoidance of the overflow computation and preserve the data precision, a simple dynamic scaling unit before energy calculations is applied. The proposed AAC encoder is implemented on a dual-core processor. Based-on the different software architecture, two solutions for recording system implementation are provided. The realized AAC encoder consumes 86MIPS and 107KB memory, and the recording system implemented by ARM can achieve at least 1X encoding.

ACKNOWLEDGEMENTS

感謝我的指導教授 吳炳飛老師從大三專題以來近四年的指導與教誨，在研究期間，給予我接觸各個技術領域的機會也提供我豐沛的研究資源。除了研究上的教導與指引讓我獲益良多，提升我專業上的能力之外，老師認真與嚴謹的研究態度更是我學習的典範，態度決定高度一直是老師要我們謹記於心的一句話，此句話也深深的影響我做人處事應持有的態度，吳老師不僅是指導教授，更是生活與學習上的導師。

在實驗室的生涯中，感謝已畢業的鐵男學長帶領我進入音訊編碼的領域，也感謝信元學長與俊傑學長協助我解決在嵌入式系統上的問題，在遭遇研究瓶頸時，給予我研究上協助。此外，阿誠學長、阿霖學長、重甫學長、益賓學長、助理淑閔以及其他實驗室同仁大家都給予我很多生活的幫助，亦感謝晉元、敏偉、秉宗、至明學弟們帶來生活上的娛樂與扶持。再者，與我一同奮戰熬夜及討論課業的 ppj、子萱、宗堯、岑瑋、元馨，也感謝我心情低落時，眾多朋友的鼓勵與支持，很高興有在實驗室的生涯中有大家的陪伴。

最後，更要感謝我的家人，我的爸媽、弟弟，感謝我的爸媽幫忙論文上的修正與關心，並提供我良好的生活條件，讓我於求學階段能無後顧之憂，需要感謝的太多，實在沒辦法全部點名。

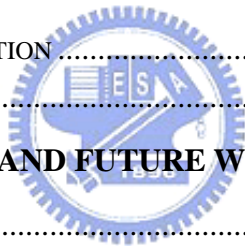
謹以本論文

獻給最親愛的家人及所有支持關愛我的人

CONTENTS

摘要	I
ABSTRACT	II
ACKNOWLEDGEMENTS	III
CONTENTS	IV
LIST OF FIGURES	VI
LIST OF TABLE.....	VIII
CHAPTER 1.INTRODUCTION	1
1.1 MPEG/AUDIO COMPRESSION.....	1
1.2 MOTIVATION	2
1.3 SCOPE OF THE THESIS.....	3
CHAPTER 2.MPEG-2 ADVANCED AUDIO CODING	4
2.1 OVERVIEW OF MPEG-2 ADVANCED AUDIO CODING	4
2.2 PSYCHOACOUSTIC MODEL (PAM).....	7
2.2.1 Threshold in quiet.....	7
2.2.2 Masking phenomena.....	8
2.2.3 Critical Band.....	9
2.2.4 Psychoacoustic model used in AAC.....	10
2.3 GAIN CONTROL	12
2.4 TIME TO FREQUENCY TRANSFORMATION.....	13
2.4.1 MDCT.....	13
2.4.2 Window Shape and Block Switching.....	14
2.5 TEMPORAL NOISE SHAPING	16
2.6 JOINT STEREO CODING	17
2.7 PREDICTION	18
2.8 ITERATION LOOPS	20
2.9 BITSTREAM MULTIPLEXING.....	22
CHAPTER 3.MPEG-2/4 AAC ENCODER OPTIMIZATION.....	24
3.1 COMPLEXITY ANALYSIS OF SOURCE CODE	24
3.2 SIMPLIFIED PSYCHOACOUSTIC MODEL.....	25
3.3 FAST MDCT	27
3.4 SIMPLIFIED TNS TOOL.....	31
3.5 MID/SIDE STEREO CODING OPTIMIZATION.....	35

3.6	MEMORY REDUCTION.....	36
CHAPTER 4. DUAL-CORE PROCESSOR IMPLEMENTATION.....		38
4.1	HARDWARE ENVIRONMENT	38
4.1.1	<i>OMAP 5912 OSK.....</i>	38
4.1.2	<i>DSP subsystem.....</i>	40
4.2	SOFTWARE ENVIRONMENT	43
4.2.1	<i>TI Software Development Tools--Code Composer Studio</i>	43
4.2.2	<i>Operating System.....</i>	44
4.2.3	<i>DSP Gateway.....</i>	46
4.3	FIXED-POINT CONSIDERATION AND DSP OPTIMIZATION	49
4.3.1	<i>Fixed-point implementation based on 16-bit arithmetic</i>	49
4.3.2	<i>Assembly and compiler optimization techniques.....</i>	54
4.4	RECORDING SYSTEM IMPLEMENTATION	56
4.4.1	<i>Recorder implementation by ARM.....</i>	58
4.4.2	<i>Recording system between two processors.....</i>	58
CHAPTER 5. EXPERIMENTAL RESULTS		62
5.1	PERFORMANCE EVALUATION	63
5.2	QUALITY EVALUATION	65
CHAPTER 6. CONCLUSIONS AND FUTURE WORK.....		68
6.1	CONCLUSIONS	68
6.2	FUTURE WORKS	69
REFERENCE		71
APPENDIX		75



LIST OF FIGURES

Figure. 1	MPEG-2 AAC encoder block diagram	6
Figure. 2	Threshold in quiet	8
Figure. 3	Frequency masking with threshold in quiet	8
Figure. 4	Temporal Masking phenomenon.....	9
Figure. 5	Block diagram of psychoacoustic model [3].....	11
Figure. 6	Block diagram of Gain Control tool in encoder.....	12
Figure. 7	Overlap region of the different blocks	13
Figure. 8	Window shape adaptation process.[3].....	15
Figure. 9	Block switching during steady-state and transient signal conditions [3].....	15
Figure. 10	(a) Source signal [4]	16
Figure. 10	(b) Transform Coded signal [4]	16
Figure. 11	Block diagram of temporal noise shaping [5]	17
Figure. 12	Block diagram of the Prediction unit for one scalefactor band [3].....	19
Figure. 13	The second-order backward-adaptive predictors [6].....	19
Figure. 14	Block diagram of outer iteration loop	21
Figure. 15	Block diagram of inner iteration loop	22
Figure. 16	Structure of ADIF format	23
Figure. 17	Structure of ADTS format	23
Figure. 18	The complexity analysis of LC AAC Encoder.....	25
Figure. 19	FAAC Implementation flow of the AAC Encoder	26
Figure. 20	Cut-off frequency V.S bit rate relationship from bandwidth control	27
Figure. 21	Implementation flow of FFT-based MDCT	28
Figure. 22	The symmetric and anti-symmetric properties of cosine and sine tables.....	30
Figure. 23	The original TNS implementation flow	32
Figure. 24	The flow chart of simplified TNS	33

Figure. 25	Simplified flow chart of M/S stereo coding.....	35
Figure. 26	Simple illustration of buffer re-usage	37
Figure. 27	The functional block diagram of the OMAP5912 [17].....	40
Figure. 28	The TMS320C55x DSP core architecture [18].....	41
Figure. 29	The C55x CCS Development Flow [19].....	44
Figure. 30	The OMAP5912 development environment with operating system.....	45
Figure. 31	Software architecture of DSP Gateway Linux APIs [24].....	46
Figure. 32	The mailbox scheme of DSP Gateway [24].....	47
Figure. 33	The DSP memory space[24]	48
Figure. 34	Double precision multiplication, R (32-bit) = X (32-bit) x Y (16-bit).....	50
Figure. 35	Double-precision multiplication, X (32-bit) x Y (32-bit).....	50
Figure. 36	The data precision of the proposed AAC encoder	51
Figure. 37	Representation of exponent detector.....	52
Figure. 38	The scaling unit applied in AAC encoder	53
Figure. 39	The development flow of OMAP 5912 with DSP gateway	57
Figure. 40	The multi-thread program flow for recording by ARM.....	58
Figure. 41	The firmware block diagram.....	59
Figure. 42	The mailbox IRQ Handler.....	59
Figure. 43	The passive way of sending/receiving for mailbox [24].....	60
Figure. 44	The ODG result at 128Kpbs.....	67
Figure. 45	The ODG result at 96Kpbs.....	67
Figure. 46	The ODG result at 64Kpbs.....	67


LIST OF TABLES

Table. 1 The sub parts loading of FFT processing.....	29
Table. 2 The inactive percentage of TNS filtering	32
Table. 3 Analysis between simplified and original LPC approach.....	34
Table. 4 C data type of TMS320C55x [25]	49
Table. 5 Description of bit-reversing addressing mode [25]	54
Table. 6 Bit-reverse Implementation result using C and ASM.....	55
Table. 7 Test audio samples	62
Table. 8 performance evaluation on TMS320C55x processor	63
Table. 9 The summary of memory section for proposed encoder	63
Table. 10 Encoding speed evaluation	64
Table. 11 Code Size of encoder and recorder.....	64
Table. 12 The Scale of ODG.....	65
Table. 13 Summary of ODG test	66



CHAPTER 1. INTRODUCTION

1.1 MPEG/Audio Compression



Multimedia technologies have intensively developed in recent years; tremendous applications covering digital audio/video have created a revolutionary change. Especially, under the popularity of internet, it even stimulates and promotes the demand for multimedia streaming. Many portable devices, such as voice recorder, MP3 player, Portable Multimedia Player (PMP), and mobile phones all require the technology of audio compression due to the limited storage. The need for interoperability, high-quality audio at lower data rates, and for a common file format led to the institution of audio standards. The state of the art audio coding algorithms, such as MPEG Audio, WMA, OGG, and Dolby AC-3 are standardized in response to these demands.

The MPEG, stands for “Moving Pictures Experts Groups” is a group that defines various standards for coding the audio-visual information. Also, the MPEG/audio is the most popular international standard for digital audio compression nowadays.

With the increasing demand of human beings on the sense of hearing, the MPEG/Audio

has already proposed many audio coding standards up to the present, such as MPEG-1 Audio(MP1,MP2,MP3), MPEG-2 BC(backward compatible), MPEG-2/4 Advanced Audio Coding (AAC, non-compatible) , and its extension, High Efficient Advanced Audio Coding (HEAAC). All MPEG/audio algorithms adhere to the basic concept that transform the source samples into frequency domain with de-correlation signals, and then quantize the transform coefficient according to the information provide by psychoacoustic model. Regarding to MPEG-1 and MPEG-2 Audio Compression algorithms, they provide different layers' implementation with different computational complexity. MPEG-2/4 AAC and HEAAC provide a high quality multi-channel standard than achievable while requiring MPEG-1 backwards compatibility. In order to define the AAC system, the audio committee selected a modular approach in which the full system is broken down into a series of self-contained modules or tools. We will introduce these coding modules in Chapter.2.



1.2 Motivation

A variety of audio formats has been proposed these days, the MPEG-2/4 audio play an important role in audio compression field. AAC provides higher coding efficiency, multi-channel support, and high-quality audio at low bit rates. Because of the AAC's superior performance, it constitutes the kernel of MPEG-4 Audio.

In addition to the audio quality, the power-consumption would be the major concern in considering the implementation on portable device. The encoder algorithm will take too much computational resources than decoder. That is why so many audio-related portable devices do not support the encoder, and is “decoder-only”. Hence, to reduce the complexity of audio algorithm is our main goal.

The OMAP5912 is a dual-core processor including one RISC (ARM926EJ-S) and one

DSP (TMS320C55x). Under such hardware architecture, the operating system can be included. And this will not only increase the flexibility of applications but also share the loading of computational power.

In this thesis, a low-complexity 16-bit fixed-point arithmetic implementation on a dual-core processor with 16-bit fixed-point DSP is proposed. And a prototype design of recording system is implemented by OMAP5912.

1.3 Scope of the thesis

This thesis contains six chapters:

- Chapter 1: the MPEG/Audio compression, the motivation, and the overview of the thesis are briefly introduced.
- Chapter 2: the algorithms of MPEG-2/4 low-complexity AAC encoder is introduced including the basic concept of psychoacoustic model.
- Chapter 3: several optimization methods are proposed in order to minimize the computational complexity and memory requirement.
- Chapter 4: the hardware and software development environment are introduced. The fixed-point optimizations based on 16-bit arithmetic are proposed; and several assembly optimization techniques are presented. At the end of this chapter, the recorder system design on a dual-core processor is realized.
- Chapter 5: the experimental results and some comparison are presented
- Chapter 6: the conclusion and possible future work are brought about.

CHAPTER 2. MPEG-2 ADVANCED AUDIO CODING

2.1 Overview of MPEG-2 Advanced Audio Coding

Started in 1994, MPEG Advanced Audio Coding, also known as MPEG-2 NBC (Non-Backward Compatible) represented the actual state of the art in natural audio coding. It provides very high audio quality at a rate of 64Kbps. Testing results carried out in 1996, showed that the MPEG-2 AAC requires 320 kbps per five full-bandwidth channels to satisfy the ITU-R quality requirements. And later it was finalized in 1997. The AAC made use of all the advanced audio coding techniques available at the time of its development to provide high quality multi-channel audio. Therefore, it also constitutes the kernel algorithm of MPEG-4 Audio standards.

In considering the tradeoff among memory cost, processing power, and audio quality, the AAC system supports three profile configurations:

- Main Profile (Main):

The main profile is intended for use when processing power and especially memory, are not a premium. With this configuration, the highest

quality for applications is provided. Except for the gain control module being applied, all the subparts of the tools are used.

- Low Complexity (LC) Profile:

The LC profile is configured without the prediction tool being employed and limited TNS order which is 7th-orders for short window and 12th-orders for long windows. The memory and processing power requirements are significantly reduced.

- Scalable Sampling Rate Profile (SSR):

The SSR profile provides the lowest complexity of the all three profiles. The gain control module is applied, including the polyphase quadrature filter, gain detectors and gain modifier. With this the gain control tool, frequency scalable signal is achievable in this configuration.

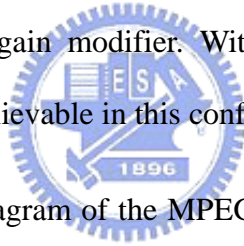


Figure.1 shows the block diagram of the MPEG-2 AAC encoder. The 2048 consecutive PCM samples are grouped together to form a “frame”. The gain control is the preprocessing tool including polyphase quadrature filter, gain detector, and gain modifier. And it will separate the time-domain signals into four sub-band signals. At this state, the signal is still in time-domain but with frequency separation property.

During the stage of time to frequency transformation, Modified Discrete Cosine Transform (MDCT) is applied in the AAC system. The audio samples are transformed into 2048 spectral lines in order to decrease the correlation of signals. Due to the property of human hearing, several scalefactor bands are defined according to the sampling rate and bitrate configured at initialization. This will divide the spectral lines into several processing groups.

Before performing time to frequency transformation, the audio samples simultaneously feed into the psychoacoustic model (PAM) [1]. The PAM will determine the masking thresholds for each scalefactor bands and provide the block-switch flags based on calculated “perceptual entropy (PE)” frame by frame.

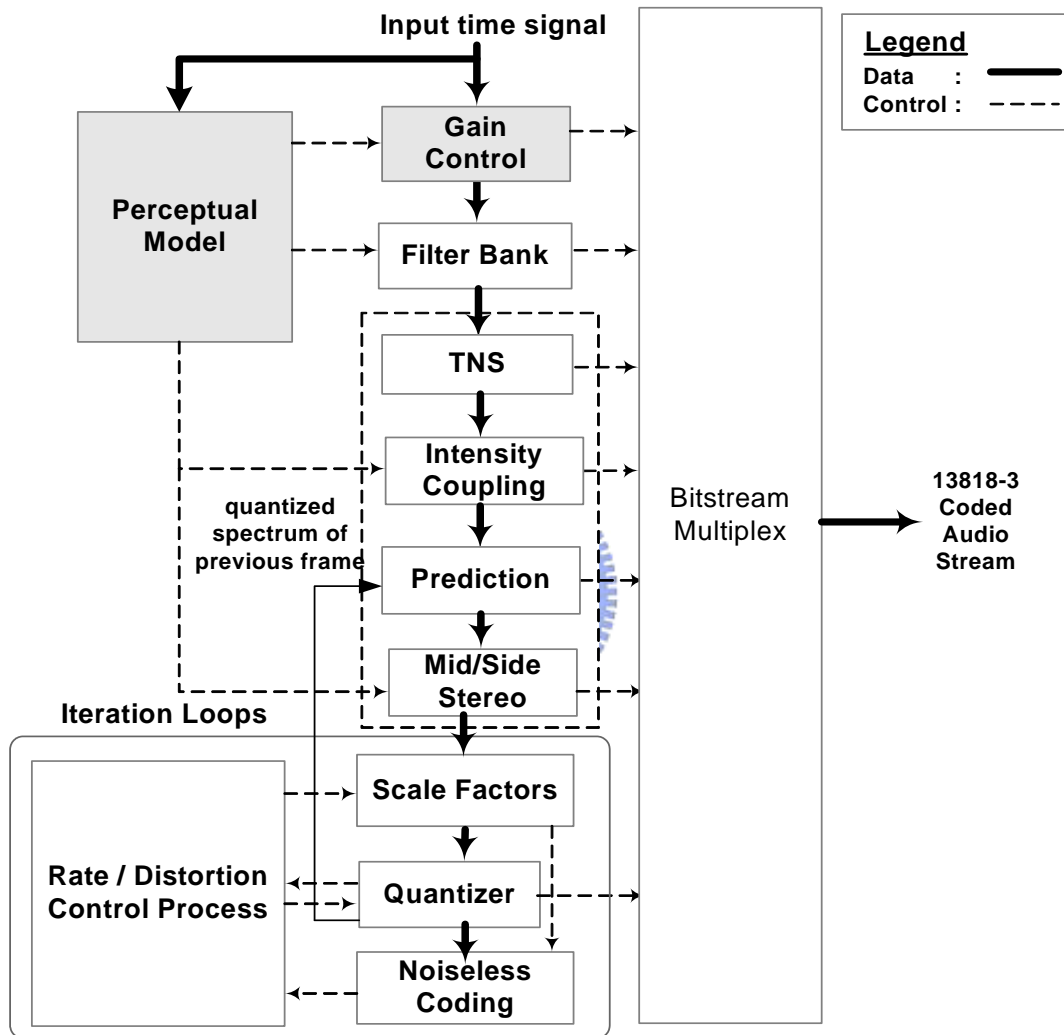


Figure. 1 MPEG-2 AAC encoder block diagram

With the block-switch information provided by PAM, time to frequency transformation has the diversity of short and long windows to obtain better time or frequency resolution. With the masking thresholds provided, the rate-distortion process will control the quantization parameters and compute the bits-cost through Huffman coding to determine suitable scalefactor for each scalefactor bands and achieve the best balance between bit usage and

audio quality.

The other blocks shown in Figure. 1, such as TNS, Intensity Coupling, Prediction, and Mid/Side Stereo Coding tools are also used in MPEG-2 AAC to obtain a better coding efficiency. We will describe the functionality of each block in the following sections.

2.2 Psychoacoustic Model (PAM)

The PAM is used for modeling the human auditory perception. In the design of audio codec, the hearing threshold represents frequency-dependent levels below which the quantization noise levels will be inaudible. By exploiting the fact that these inaudible signals are irrelevant information, the frequency components are quantized and coded with a relatively small number of bits without introducing the audible distortion. There are certain factors that need to be taken into consideration while referring to the psychoacoustic model.

2.2.1 Threshold in quiet

The threshold in quiet, also called absolute threshold of hearing, stands for the lowest sound pressure level (SPL) at any given frequency that can be detected by human ear. This threshold is frequency dependent and typically relatively high at low frequencies and increase quite rapidly above 16 kHz. Figure. 2 shows the typical curve for this threshold. During the range from 2 kHz to 5 kHz, the listener would be most sensitive. The threshold curve is extremely important for audio coding, since any sound with SPL under this curve is viewed as inaudible or non-perceivable.

The threshold in quiet can be approximated [2] by the following frequency dependent function defined as

$$T_q(f) = 3.64\left(\frac{f}{1000}\right)^{-0.8} - 6.5e^{-0.6\left(\frac{f}{1000}-3.3\right)^2} + 10^{-3}\left(\frac{f}{1000}\right)^4 \quad (1)$$

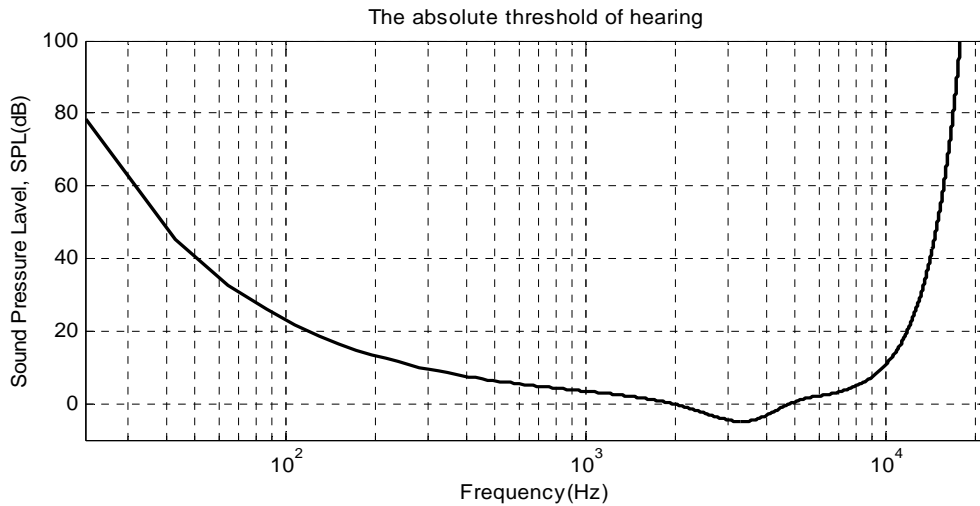


Figure. 2 Threshold in quiet

2.2.2 Masking phenomena

In addition to the threshold in quiet, the masking phenomenon refers to a soft sound becomes inaudible due to the presence of a louder sound. We can sort the masking effect into frequency domain and temporal domain for discussions.

- Frequency masking: It also called simultaneous masking happens when two adjacent frequency tones with different sound pressure level producing in the meantime. The masker (louder sound) will overwhelm the maskee (weaker sound) as shown in Figure.3

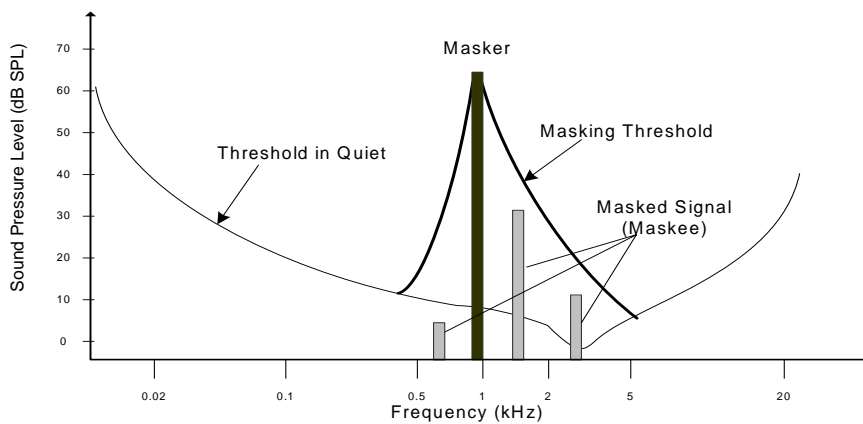


Figure. 3 Frequency masking with threshold in quiet

- Temporal masking: This masking effect takes place when the masker and maskee are not presented simultaneously. As shown in Figure. 4, the masking effect can be extended in time domain outside the period at the moment masker presented. The pre-masking and post-masking effects are both the temporal masking phenomena and the post-masking is usually last longer than pre-masking.

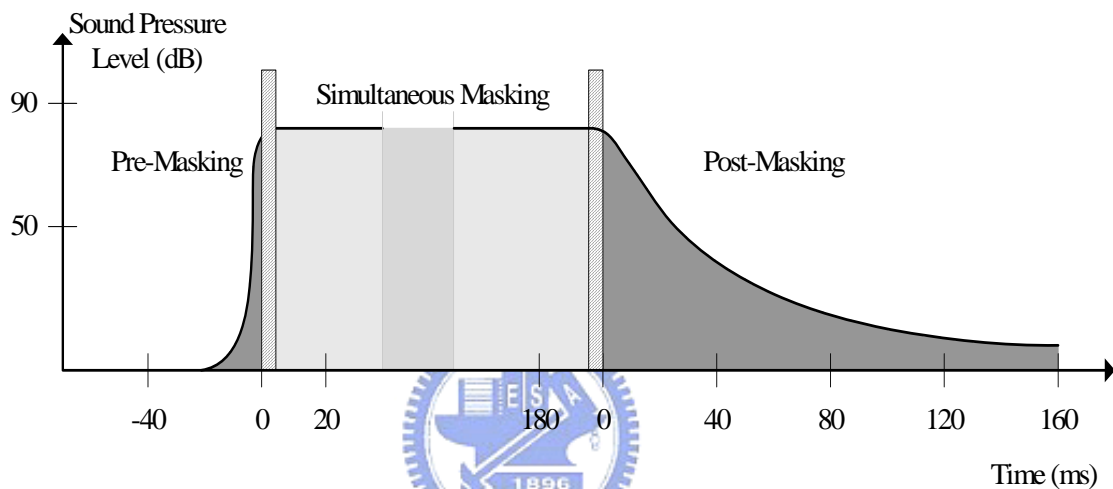


Figure. 4 Temporal Masking phenomenon

2.2.3 Critical Band

In psychoacoustic, when referring to the sound, the unit to represent the frequency of a signal is “bark”. This unit comes from the “critical band” phenomenon of human ears. They have different sensitivities to audio signals in different frequency bands. The frequency dependent limited to its frequency resolution is expressed in terms of “critical band”. Therefore, the human ears can be modeled as several overlapping bandpass filters ranging from 2Hz to 22 kHz with different bandwidths and central frequencies.

2.2.4 Psychoacoustic model used in AAC

The PAM used in MPEG-2 AAC provides the analysis of input PCM samples frame by frame and calculate the maximum allowable distortion in order to decide and remove the inaudible signal. Also, it analysis the transient properties of input frames through perceptual entropy.

The procedures are described in the following steps:

1. Performing the 256-point and 2048-point FFT on input PCM samples. The polar representation of the transform is calculated. Magnitude and phase components are obtained.
2. Using the spreading function and spectral components of input samples to calculate masking thresholds for each scalefactor band and the threshold in quiet is taken into consideration.
3. Calculate the signal-to-mask ratio (SMR) for each scalefactor band, and the SMR information will be sent to the quantize.
4. In order to decide which block type for MDCT to use, the perceptual entropy is calculated and checking whether the short window will be applied or not.

The outputs of the PAM are:

1. A set of Signal-to-Mask Ratios (SMR) and thresholds
2. The delayed time domain data (PCM samples), which are used by MDCT.
3. The block type(short or long) for the MDCT
4. An estimation of the coding bits

Figure. 5 shows the procedure for PAM. For more detail information, could refer to [3].

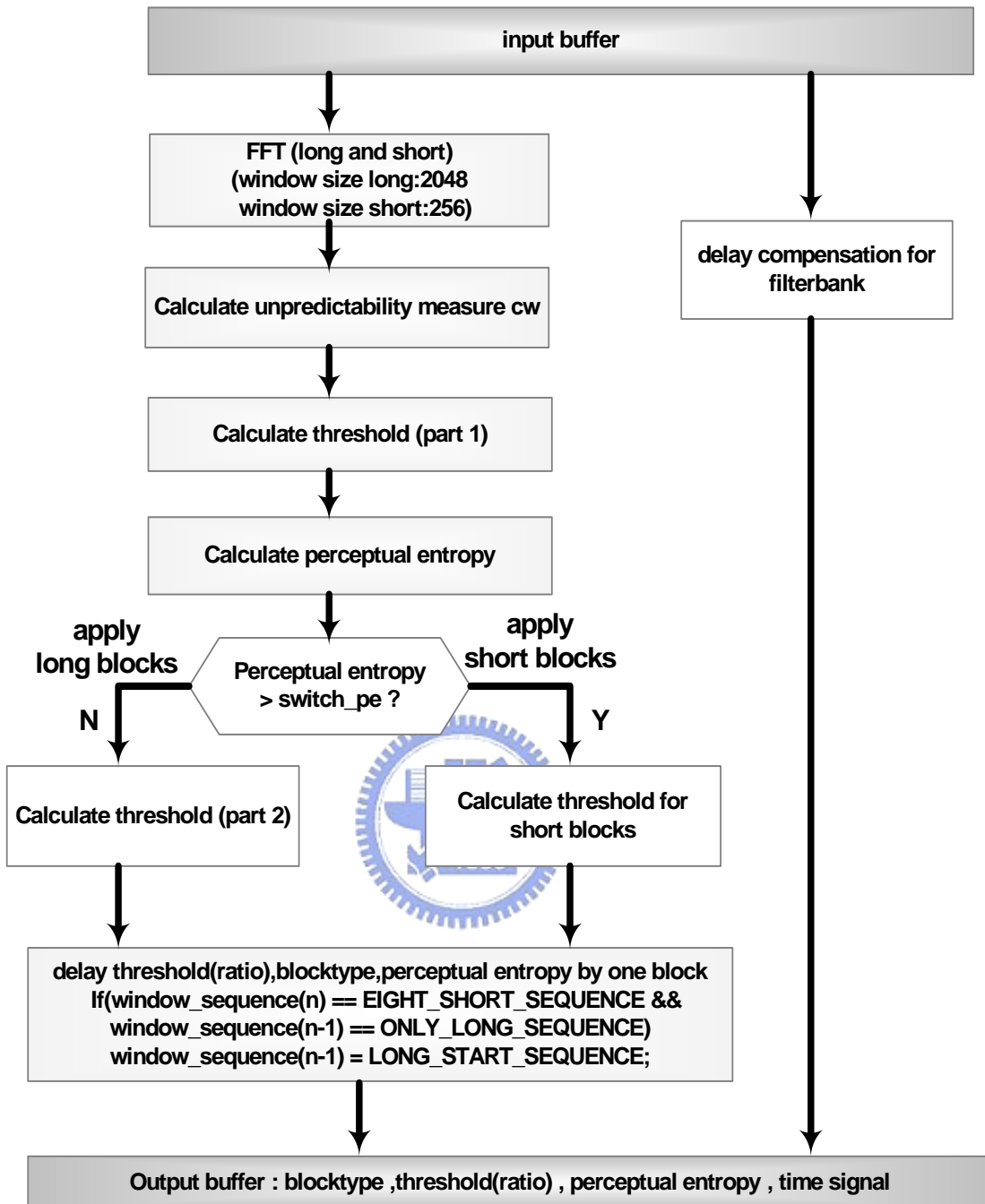


Figure. 5 Block diagram of psychoacoustic model [3]

2.3 Gain Control

The gain control tool is added into the input stage of the encoder only for the SSR profile. It consists of a PQF filter bank, gain detectors, and gain modifier, as shown in Figure. 6. Through the gain control, the 2048 input PCM samples will be split into four frequency bands with equal bandwidth. The signals from the output of the lowest band will be processed by MDCT without gain detector and gain modifier. The advantage of the scalability is that the signals from upper filter bank can be dropped in the decoder in order to reduce the complexity.

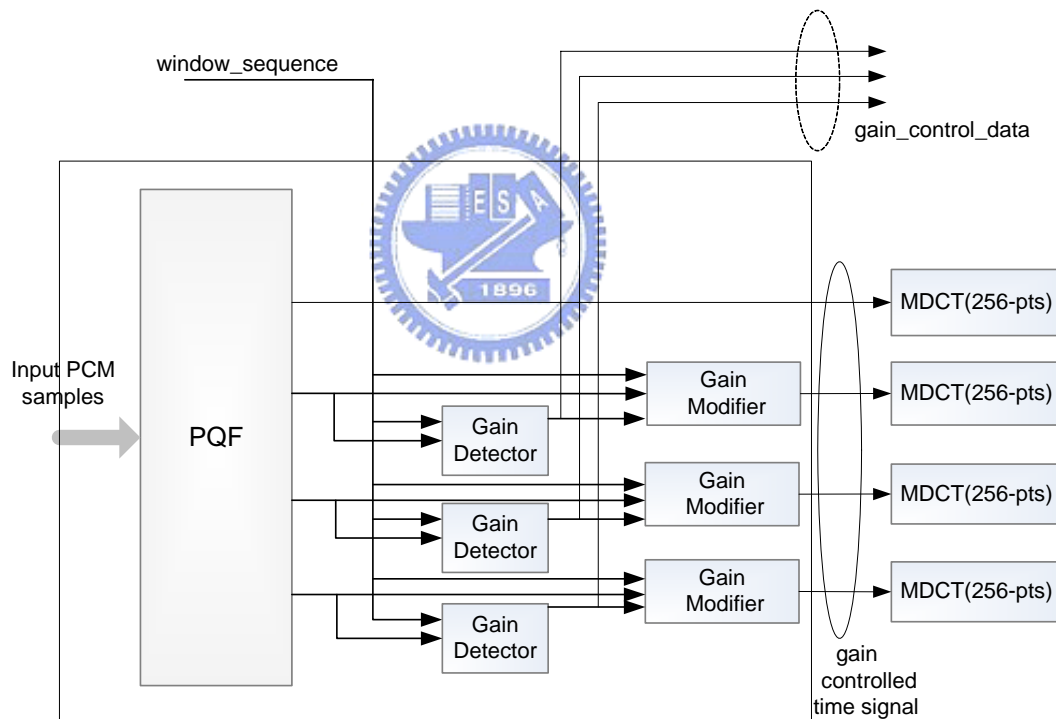


Figure. 6 Block diagram of Gain Control tool in encoder

2.4 Time to Frequency Transformation

In the block of time to frequency transform of the AAC system, filter bank is employed for time-frequency conversion. The conversion is done by a time-variant MDCT with 2048 or 256 block length. In the sub-sections, we will brief describe the MDCT and the block switching scheme in the AAC.

2.4.1 MDCT

The MDCT is actually a type-IV discrete cosine transform but overlapping by 50% with preceding block and following block as shown in Figure. 7. The overlapping of input signals will not only lead to the result of better energy-compaction, but also avoid the artifacts due to the block boundaries. The time-domain input samples are modulated by the appropriate window function first and then perform MDCT. The expression is as (1):

$$X_{i,k} = 2 \sum_{n=0}^{N-1} w(n)x_{i,n} \cos \left[\frac{2\pi}{N} (n+n_0) \left(k + \frac{1}{2} \right) \right], \quad k = 0, 1, \dots, \frac{N}{2} - 1 \quad (1)$$

where

n = sample index,

N = window length of the one transform window based on the window sequence, 2048 for long window and 256 for short window,

i = block index,

k = spectral coefficient index,

$$n_0 = \frac{1}{2} \left(\frac{N}{2} + 1 \right),$$

$w(n)$ = window function (Kaiser-Bessel Derived or Sine function).

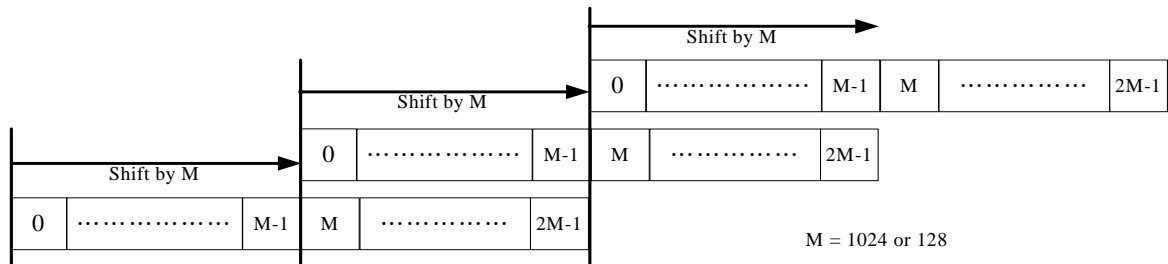


Figure. 7 Overlap region of the different blocks

2.4.2 Window Shape and Block Switching

As shown in equation (1), the window function $w(n)$ also has a significant influence on frequency response of the input sample block. Two window parameters are directly linked to these properties, selected window length and shape. The selection of the window length and the shape will determine the degree of spectral separation of the filter bank. In the AAC system, the sine window and Kaiser-Bessel-derived (KBD) window are provided as window shape selection and 1024-point or 128-point processing is provided as the window length adaptation. The short window would be used for transient signal and result in better time resolution but the long window would be used for steady state signal and result in better frequency resolution. The coefficients of sine and KBD window is as (2) and (3)

$$W_{SINE}(n) = \sin\left(\frac{\pi}{N}\left(n + \frac{1}{2}\right)\right) \quad \text{for } 0 \leq n < N \quad (2)$$

$$W_{KDB}[n] = \frac{I_0\left[\pi\alpha\sqrt{1.0 - \left(\frac{n - N/2}{N/2}\right)^2}\right]}{I_0[\pi\alpha]} \quad \text{for } 0 \leq n < N \quad (3)$$

where, $I_0(x)$ is the 0th modified Bessel function defined as $I_0(x) = \sum_{k=0}^{\infty} \left(\frac{(x/2)^k}{k!}\right)^2$

and α is the kernel window alpha factors ,

$$\alpha = \begin{cases} 4, & \text{for } N = 2048 \\ 6, & \text{for } N = 256 \end{cases}$$

Figure. 8 shows the adaptation of window shape function.

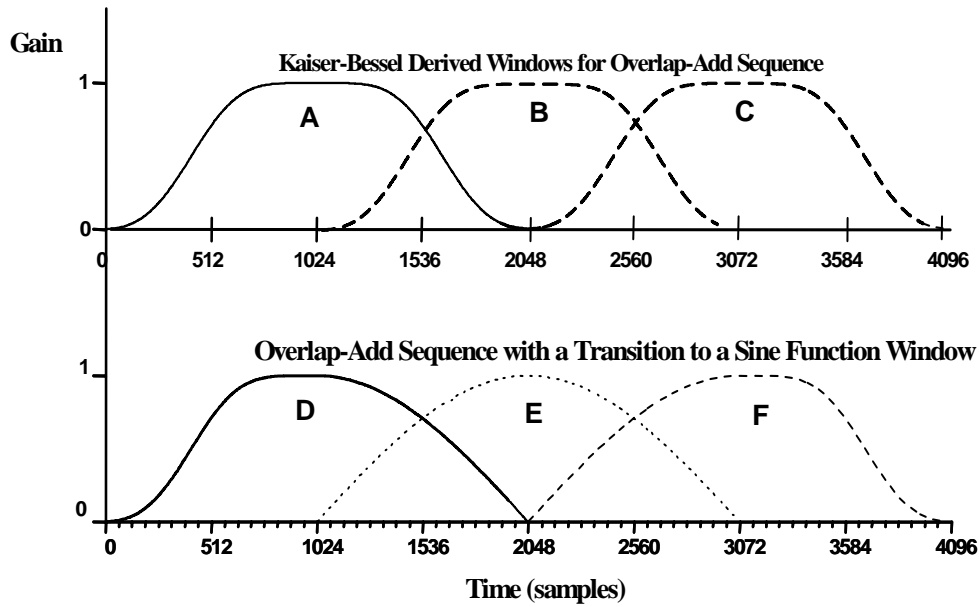


Figure. 8 Window shape adaptation process.[3]

As mentioned above, the window length selection would be the trade-off between time resolution and frequency resolution. So as to solve the problem of block alignment, the start and stop windows that are the so-called adaptive windows are used for adaptation from normal window to short window and from short window to normal window respectively. The block switching of long and short is shown in Figure.9.

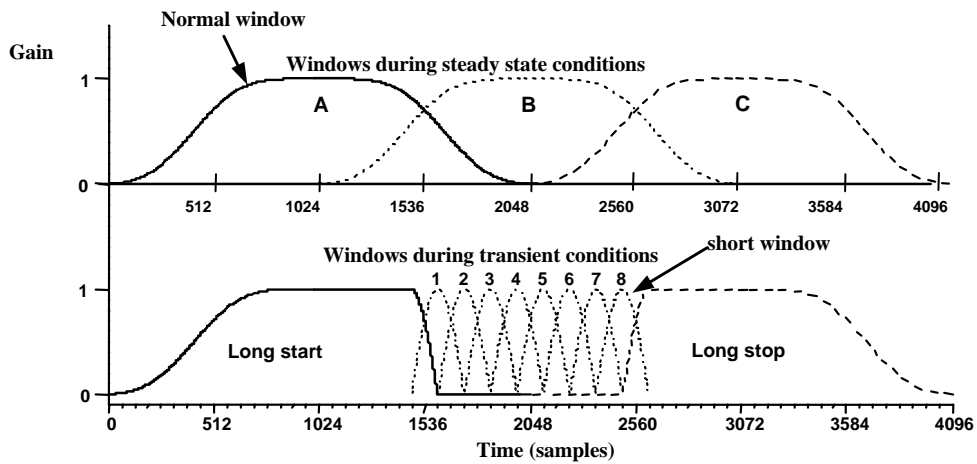


Figure. 9 Block switching during steady-state and transient signal conditions [3]

2.5 Temporal Noise Shaping

Improper encoding the transient or pitch signal will lead to pre-echo phenomenon after decoding. This phenomenon happens when the signal magnitude abruptly rises as observed in Figure.10 (a)(b). In various audio coding algorithms, dynamic window switching as subsection 2.3.2 mentioned is a good approach to conquer this problem. A novel concept in relieving the pre-echo is represented by the temporal noise shaping (TNS).

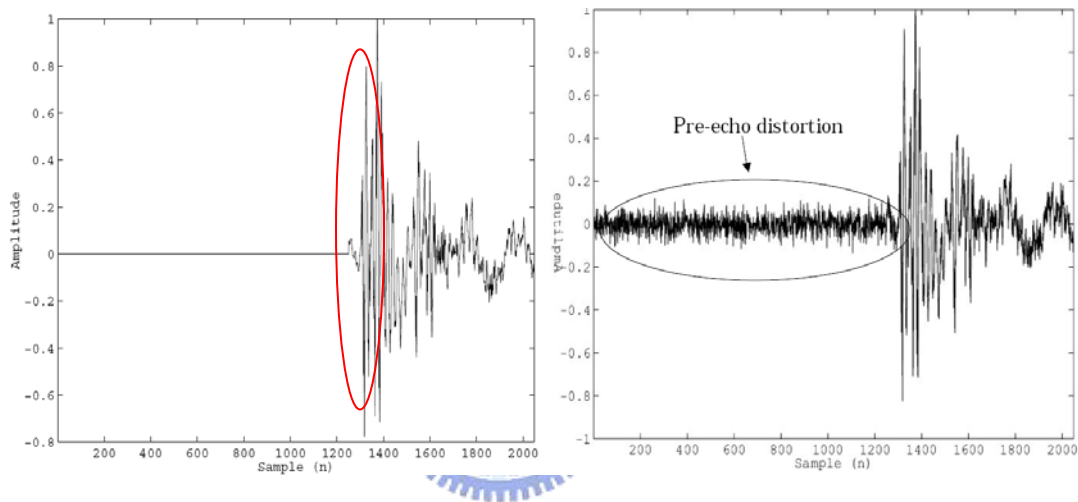


Figure. 10 (a) Source signal [4]

Figure. 10 (b) Transform Coded signal [4]

TNS technique permits the encoder to exercise control over the temporal fine structure of the quantization error. The concept of TNS uses the duality between time domain and frequency domain. Signal with an “un-flat” spectrum can be coded efficiently either by directly coding spectral lines or by applying predictive coding methods to the time signal. Here, in the TNS tool, the predictive coding is applied in frequency domain so that the quantization error will appear adapted to the temporal shape of the input signal. Figure. 11 illustrate the concept of TNS, and (4) is a simple deviation of TNS. The shaping of quantization error would depend on the linear predictor, $H(z)$.

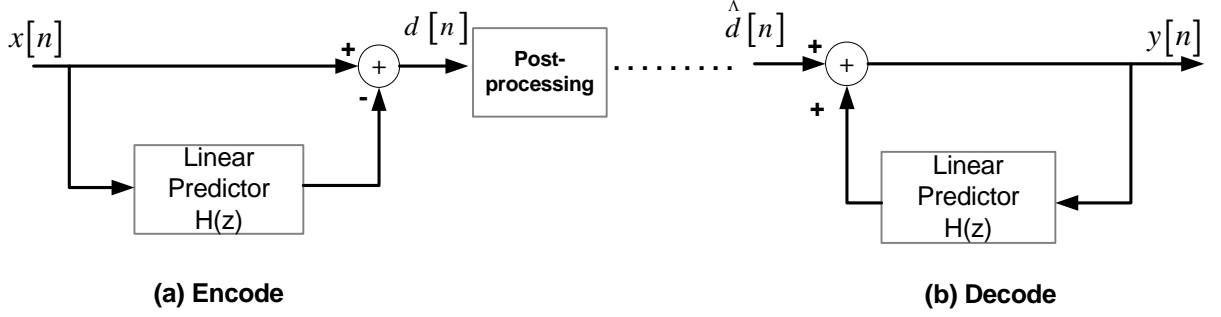


Figure. 11 Block diagram of temporal noise shaping [5]

$$E_{err}(z) = X(z) - Y(z) = \frac{D(z) - \hat{D}(z)}{1 - H(z)} = \frac{Q(z)}{1 - H(z)} \quad (4)$$

$$\implies S_{E_{err}}(z) = \frac{S_Q(z)}{1 - H^2(z)}$$

where, $E_{err}(z)$ represents the reconstruction error, and $Q(z)$ represents the quantization error producing by the encoder part.



2.6 Joint Stereo Coding

To provide a better coding efficiency, two stereo coding techniques are provided in the AAC system—Mid/Side stereo coding (M/S) and Intensity stereo coding.

For the M/S stereo coding technique, it utilizes the correlation between channel pairs. The higher the correlation left/right channels is, the required bits for the sum and difference of two channels would be less. Therefore, there will be a threshold for deciding M/S stereo applied or not. The expression of M/S stereo coding is as (5).

$$\begin{bmatrix} M \\ S \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \times \begin{bmatrix} L \\ R \end{bmatrix} \quad (5)$$

For the intensity stereo coding technique, it utilizes the properties of human ears that are sensitive to the amplitude and phase for low frequency signal but sensitive to amplitude

component only for high frequency. The intensity coupling tool is used to exploit irrelevance in the between both channels of a channel pair in the high frequency regions. Hence, the corresponding “intensity position value” will be computed scalefactor band by scalefactor band and scaling the “intensity signal spectral coefficients” are calculated to replace the left channel signal with right channel signal setting to zero. Equation (6) is the calculation of intensity position value and (7) is calculation of the intensity signal spectral coefficients.

$$is_position[sfb] = NINT \left(2 \log_2 \left(\frac{E_l[sfb]}{E_r[sfb]} \right) \right) \quad (6)$$

$$spec_i[i] = (spec_l[i] + spec_r[i]) \left(\frac{E_l[sfb]}{E_s[sfb]} \right) \quad (7)$$

where E_l, E_r, E_s represent the energy of left, right, and sum channel.



2.7 Prediction

To further improve the redundancy reduction, the AAC system applies the prediction tool in the configuration of Main profile. Figure. 12 shows the block diagram of prediction tool. The structure of the predictor adopts second-order backward-adaptive predictor, using lattice structure’s implementation, as shown in Figure. 13. The prediction tool will estimate the possible spectral coefficients in the following block by coefficients in the previous ones. Therefore, only the prediction error needs to be transmitted. It is because the required bits are based on encoding the prediction error which is the output of prediction tool. The higher the correlation between two consecutive frames is, or to say that, the more stationary the signal is, the bit-cost needed to encode prediction error would be less.

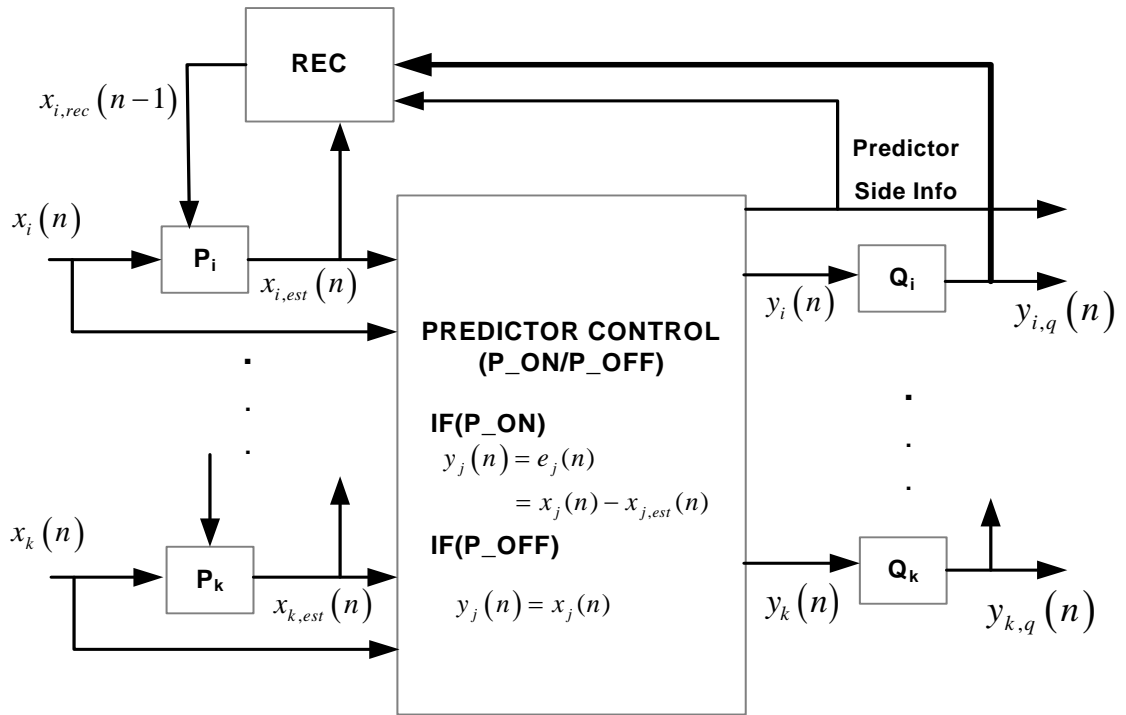


Figure. 12 Block diagram of the Prediction unit for one scalefactor band [3]

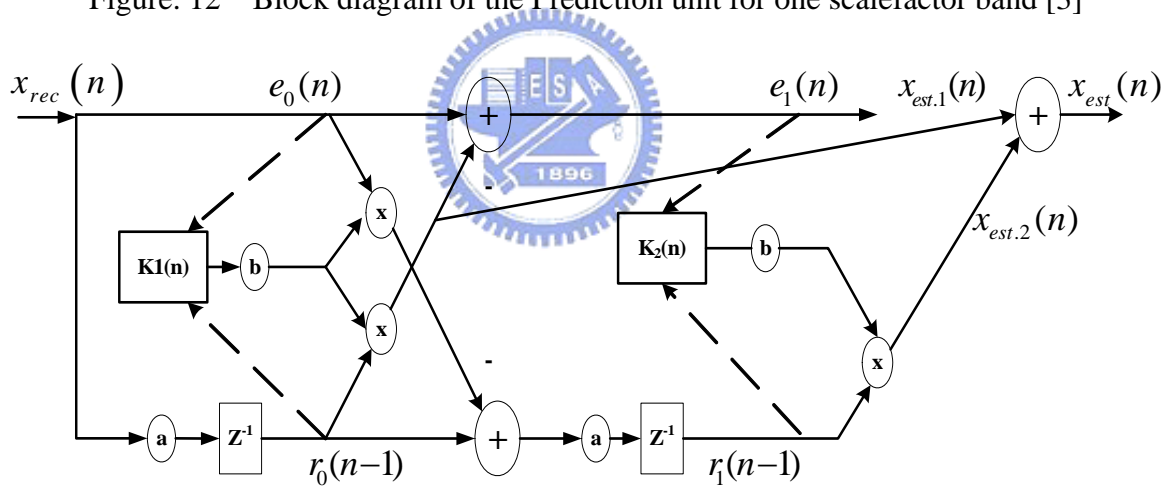


Figure. 13 The second-order backward-adaptive predictors [6]

The prediction tool is not always on the go working. In order to guarantee that the prediction is only used while the results will bring about increasing in coding gain, the appropriate predictor control is necessary and a small amount of predictor control information has to be transmitted to the decoder. For the detail control scheme and deviation of prediction coefficient can be found in [7].

2.8 Iteration Loops

In order to achieve a high compression ratio, iteration loops is applied to the spectral coefficient in the AAC system.

The quantization and Huffman coding play the important roles during the iteration loop. The thresholds called allowed distortion computed by PAM in the previous stage is used as distortion criteria in the iteration process. The primary goal of the iteration loop is to quantize the spectral coefficient passing from M/S stereo coding, then calculates the require bits and correlative information.

There is no standardized strategy for optimum quantization or fine-tune approach during the iterative process; the only requirement is that the output bit stream of the encoder must be AAC-compliant. Here, in the international standard ISO/IEC 13818-7, the two nested-loops are proposed which include the outer loop and inner loop.

The outer loop, also called distortion control loop, performs the work of shaping quantization error resulting from non-uniform quantization during the inner iteration loop. To control the distortion comes from non-uniform quantization, the scalefactors of each scalefactor bands with actual distortion exceeding the allowed distortion will be amplified. It is worth noted that the amplified scalefactor band will lead to the increasing in number of encoding bits, so that quantization stepsize might be changed once more in the inner iteration loop to decrease the bit-costs. It might be involved in the infinite loops without considering termination conditions. Therefore, several termination conditions are applied, listed as follows

- All scalefactor bands are already amplified
- The difference between two consecutive scalefactors is greater than 64

Figure. 14 shows the block diagram of the outer iteration loop

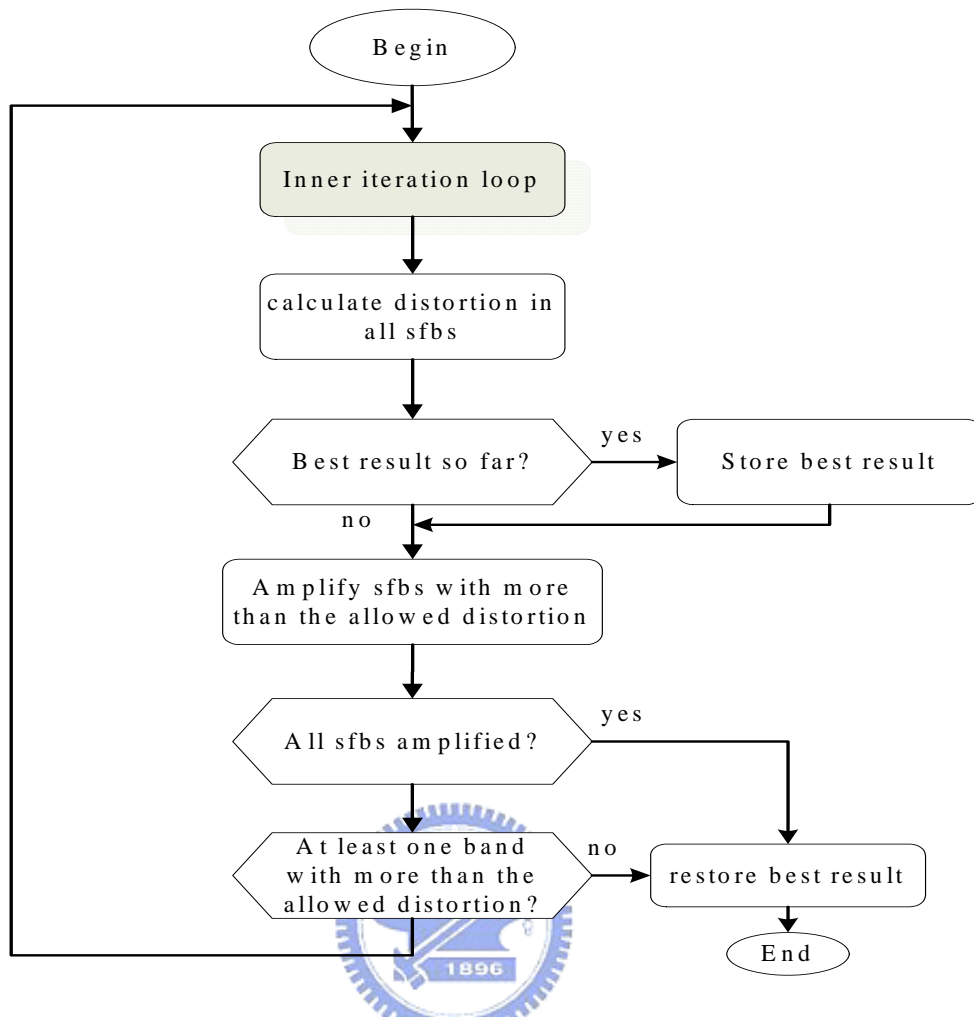


Figure. 14 Block diagram of outer iteration loop

The inner loop, also called rate control loop, performs the non-uniform quantization. After the quantization, the set of 1024 quantized spectral coefficients will be clipped and merged sections to achieve lowest bit counts. Then, Huffman coding is performed according to the 12 pre-defined Huffman tables, and the number of used bits is counted. It will be examined that whether the number of used bits are less than the number of available bits per frame. If the Huffman coded bits are more than the available bits, the quantization stepsize will be increased and perform the iterative search of proper quantization parameters (*stepsize*). The inner loop would be terminated until the Huffman coded spectral data can be encoded with the number of available bits. Figure.15 shows the flow chart of inner loop, and the non-uniform quantization used in AAC is describes as follows:

$$ix(i) = \text{sign}(xr(i)) \cdot NINT \left(\left(\frac{|xr(i)|}{\sqrt[4]{2}^{\text{quantizer_stepsize}}} \right)^{0.75} - 0.0946 \right) \quad (8)$$

where $NINT(x)$ represent the nearest integer of x .

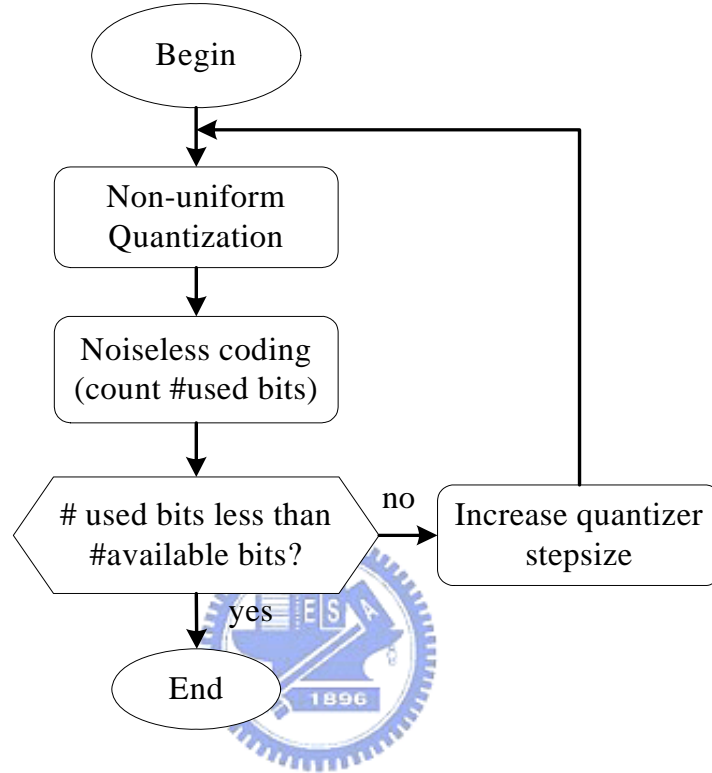


Figure. 15 Block diagram of inner iteration loop

2.9 Bitstream Multiplexing

The block, bitstream multiplexing, is the final stage of the encoder. All the information producing from the previous encoding tools, such as the short/long block information in filter bank, TNS filtering orders and other information, the `ms_used` flags of M/S tools, and quantized data are packed together as AAC-compliant format.

There are two different formats support by the AAC system. One is the Audio Data Interchange Format (ADIF); the other is the Audio Data Transport Stream Format (ADTS). The structure of these two bit streams are shown in Figure. 16 and Figure. 17



Figure. 16 Structure of ADIF format



Figure. 17 Structure of ADTS format

More detail information about bitstream formatting can be found in [3].



CHAPTER 3. MPEG-2/4 AAC ENCODER OPTIMIZATION

As mentioned in the section 2.1, the AAC system supports three profiles, the Main profile, the SSR profile, and the LC profile. Considering the memory requirement and the computational complexity, the LC profile is the most suitable profile for implementation on portable device. Here, the source code provided by FAAC [8] is adopted by us. The original source code of FAAC is a floating-point version. The complexity of each coding tools in the AAC system are first analyzed by us using Visual Studio 6.0. Then, several modules based on 16-bit fixed-point arithmetic are optimized in order to further porting on 16-bit fixed-point DSP processor. The optimizations cover the simplified PAM, Fast MDCT, simplified TNS tool, and simplified M/S stereo coding. In considering the memory issues in the encoder, some memory reduction based on statistical analysis is adopted in Huffman Tables and Huffman coding.

3.1 Complexity Analysis of Source Code

For the complexity analysis, the built-in *clock()* function are used by us in Visual Studio 6.0 to gather statistics for each coding tools in the AAC encoder. Figure.18 shows the

complexity analysis of the LC AAC encoder. According to the analysis, the quantization and PAM are the two most time-critical modules. The PAM normally requires transcendental function, such as the power, square root functions which are computationally demanding. Another computational demanding block is the Quantization. Inside the Quantization, the thresholds, so-called allowed-distortions are calculated, and the scalefactors of each scalefactor bands and Huffman coded data are produced.

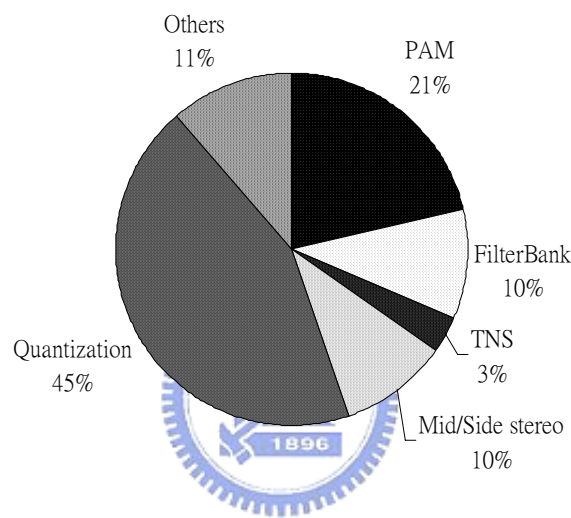


Figure. 18 The complexity analysis of LC AAC Encoder

3.2 Simplified Psychoacoustic model

The purpose to apply PAM is to analyze the transient characteristics of input PCM samples and calculate the thresholds of maximum allowed-distortion. The determination of transient signal will result in applying long or short window (2048-point or 256-point MDCT) in the filter bank in order to solve the phenomenon of pre-echo. In addition to the block switching scheme in MDCT, the AAC also adopted TNS to relieve the phenomenon of pre-echo. Based-on the consideration of existed TNS tool which is able to compensate

negative effect of pre-echo and related research [9] shows that encoding without block switching didn't cause significant effect in quality loss; we first remove the block-switching scheme in PAM. Figure.19 shows the simple implementation block diagram of the encoder.

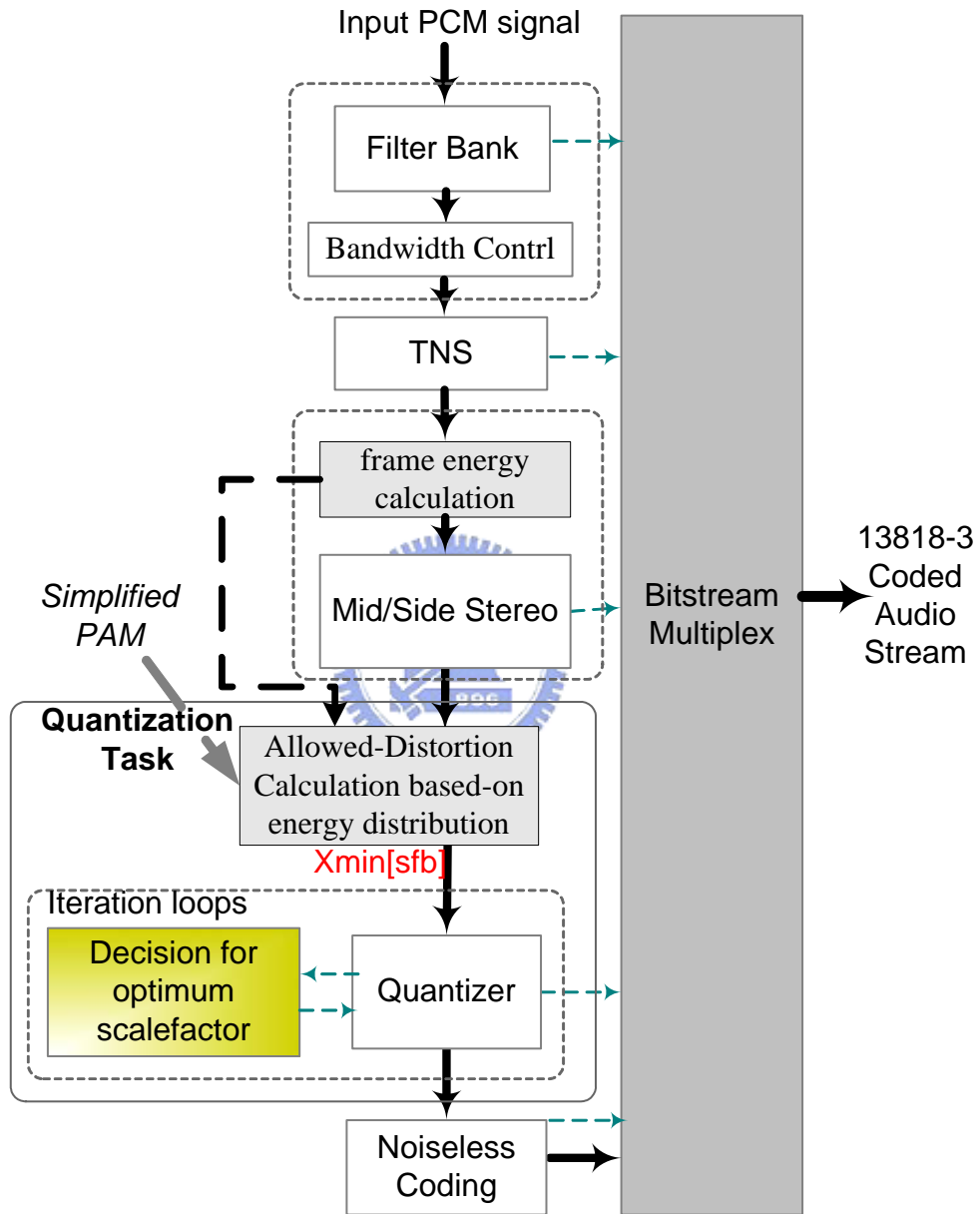


Figure. 19 FAAC Implementation flow of the AAC Encoder

Subjective tests also revealed people prefer the sound with a limited bandwidth to the sound with full bandwidth but with unmasked distortion[9][10]. In order to further recover the

audio quality, a low-pass filter is applied in the bandwidth control. In this way, more bits can be allocated for the low-frequency band. Figure. 20 shows the relationship between cut-off frequency and bit rate.

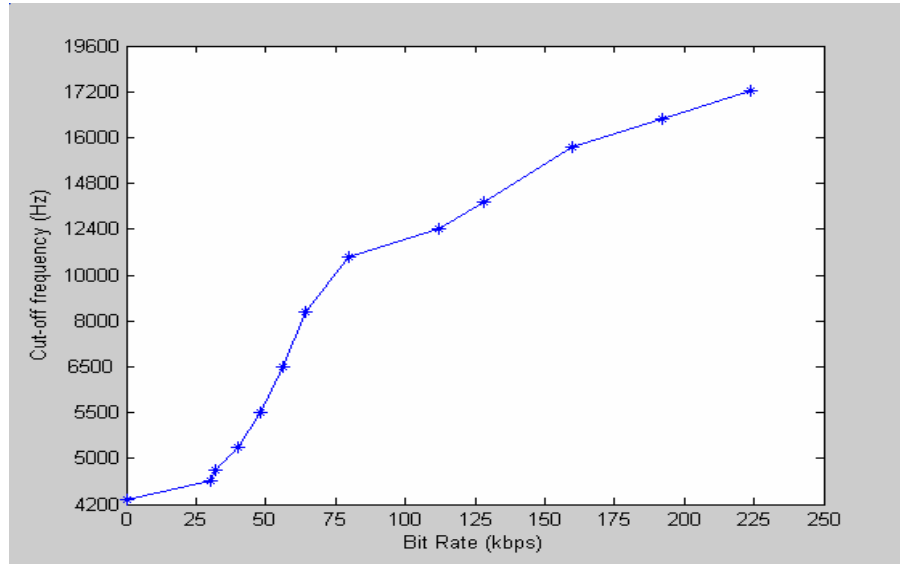


Figure. 20 Cut-off frequency V.S bit rate relationship from bandwidth control

The mathematical form of the bandwidth control is as follow

$$L(x_f(i)) = \begin{cases} x_f(i), & \text{if } i \leq NINT\left(\frac{\Omega_c}{\Omega_s} \times 1024\right) \\ 0, & \text{if } i > NINT\left(\frac{\Omega_c}{\Omega_s} \times 1024\right) \end{cases} \quad (9)$$

where Ω_c represents the cut-off frequency and Ω_s represents sampling rate

3.3 Fast MDCT

In the filter bank of the AAC System, MDCT with 50% overlap is used. The MDCT can be calculated using FFT. Because of the odd transform property, there are faster approaches

[11][12] that can be applied.

The coefficient with length N of MDCT and O^2DFT is shown in equation (10)(11)

$$MDCT_N(m) = \sum_{k=0}^{N-1} x(k) \cos\left(\frac{\pi}{2N}(2m+1)\left(2k+1+\frac{N}{2}\right)\right) \quad (10)$$

$$O^2DFT_N(m) = \sum_{k=0}^{N-1} x(k) e^{-j\frac{\pi}{2N}(2k+1)(2m+1)} \quad (11)$$

$$\text{where } 0 < m \leq \frac{N}{2} - 1$$

By using the O^2DFT coefficient to represent for coefficient of $MDCT$, MDCT can be rewritten as real part of odd-time odd-frequency discrete Fourier transforms (O^2DFT), as shown in equation (12).

$$MDCT_N(m) = F(m) = \text{Re}\{O^2DFT\{x(k-N/4)\}\} \quad (12)$$

After applying the methods proposed in [11][12], the fast MDCT can be implemented by the flow shown in Figure.21. Thus, the MDCT can be calculated using only one $N/4$ -point FFT and some pre- and post-rotation of the sample points.

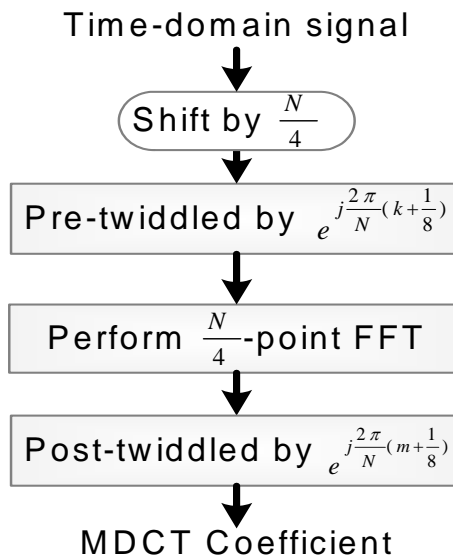


Figure. 21 Implementation flow of FFT-based MDCT

Due to the implementation of MDCT is based-on FFT, the FFT computation turns out to be the most computational demanding task in the filter bank. The analysis of filter bank shows that FFT computations occupy about 70% workload of total filter bank. Therefore, we further consider speeding up the FFT computations and reduce the memory requirement while performing FFT.

The implementation of FFT contains three sub parts. One is the *Check_Table()* task which will calculate the twiddle factors needed for performing the butterfly, as shown in equation (13), another is the *Reorder()* task which performs the in-place decimation of input samples, and the other is the *FFT_PROCESS()* which performs the butterfly construction.

$$\begin{cases} \text{costbl}[i] = \cos\left(\frac{2\pi \cdot i}{N}\right); & i = 0 \sim 255, \text{ for } N = 512 \\ \text{negsintbl}[i] = -\sin\left(\frac{2\pi \cdot i}{N}\right); & i = 0 \sim 63, \text{ for } N = 64 \end{cases} \quad (13)$$

Table.1 lists the general implementation loading and memory requirement of each sub-part in filter bank.



Table. 1 The sub parts loading of FFT processing

Task	Loading (%)	Memory requirement (N/4-pt)
<i>Check_Table()</i> (twiddle factors)	30%	long window : 256 (cosine) + 256 (negative sine) entries short window : 32 (cosine) + 32 (negative sine) entries
<i>Reorder()</i>	25%	long window : 512 entries short window: 64 entries
<i>FFT_PROCESS()</i>	45%	

It is noted that the memory requirement of *Reorder()* task is owing to the pre-calculation and storage for bit-reverse index.

To reduce the computational complexity of FFT, the look-up table method is applied to replace the computation of *Check_Table()*, therefore, 30% of loading in FFT can be eliminated. On the purpose of reducing memory requirement of twiddle factors, the symmetric and anti-symmetric relationship of cosine and sine are applied. However, a little memory addressing overhead will be increased in *FFT_PROCESS()*. Figure. 22 shows the symmetric and anti-symmetric properties of cosine and negative sine. By the way we took, the memory reduction will be reduced from $(2 \cdot 256 + 2 \cdot 32) \cdot 4 = 2304$ (bytes) to $(129 + 17) \cdot 4 = 584$ (bytes). About four times memory reduction can be achieved.

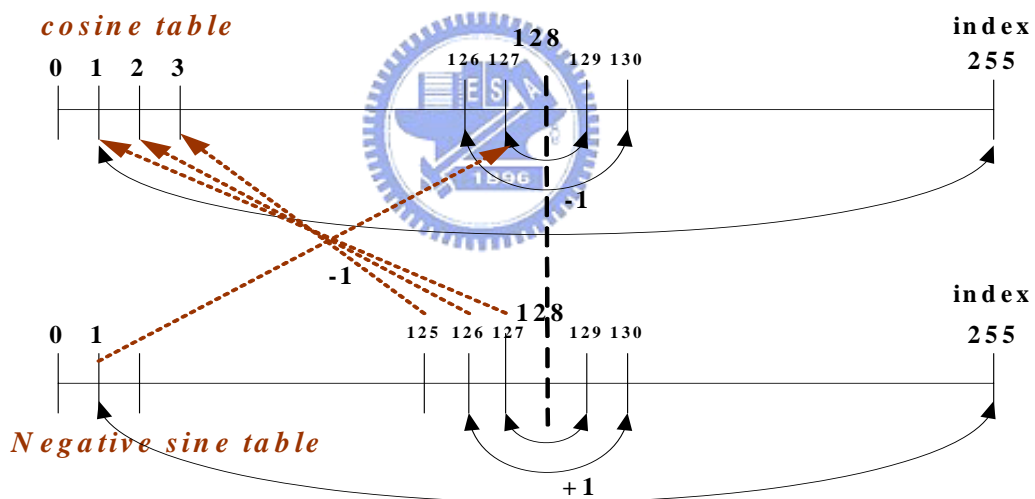


Figure. 22 The symmetric and anti-symmetric properties of cosine and sine tables

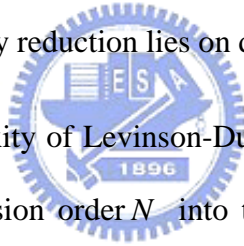
As previous mentioned, the purpose of *Reorder()* task is to perform the in-place decimation of input samples, but it will take memory to pre-store the bit-reverse index. For the bit-reversal optimization, fast bit-reversal permutation algorithm proposed in [13] is applied. The approach for computing the bit reversal is based on representations in $GF(2^b)$ (Galois Field) [13]. By applying this bit-reverse algorithm, the program loop can be reduced from $N \cdot q (N = 2^q)$ to N , and no more need for the storage of bit-reverse index.

3.4 Simplified TNS tool

The purpose of TNS tool is to perform the open-loop linear prediction on the frequency domain so as to shape the quantization noise and relieve effect caused by the pre-echo phenomenon. This is done by applying a filtering process to some parts of spectral coefficient. However, not all the spectral coefficient will apply filtering process. The filtering process will be activated only when the prediction gain of spectral data is greater than a pre-defined threshold (DEF_TNS_THRESH), as shown in Figure.23.

At first, the active percentage of the TNS filter is analyzed. Table.2 shows our statistics of the inactive percentage under the condition of applying only long window in the overall encoding process. It shows that the activation of TNS filtering has the tendency of turned-off. Therefore, the focus of complexity reduction lies on decision task.

It is known that the complexity of Levinson-Durbin Recursion is $O(N^2)$ [14]. For this reason, we try to split the recursion order N into two blocks. In the simplified TNS flow chart, 6th-order auto-correlation and Levinson-Durbin are performed at first to provide an early-decision approach. If the decision does not make properly, the following recursion order from 7th to 12th will be performed. Figure.24 shows the flow chart of simplified TNS.



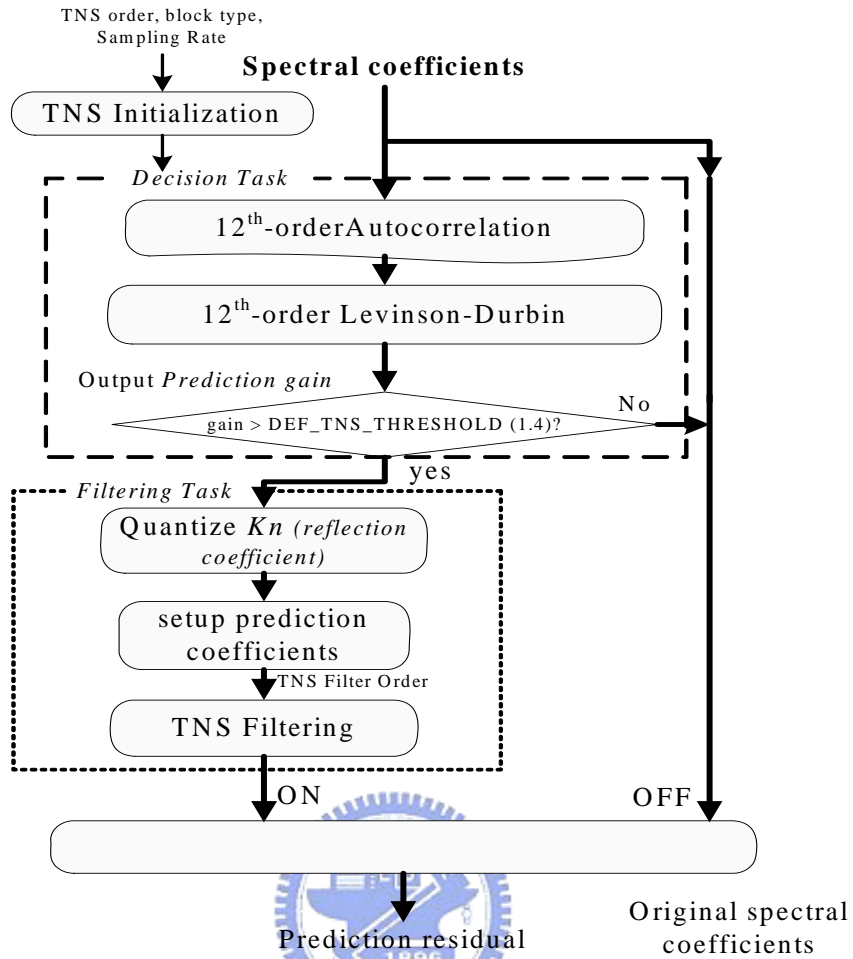


Figure. 23 The original TNS implementation flow

Table. 2 The inactive percentage of TNS filtering

Test samples	TNS Filtering inactive Percentage for Long Window
AlwaysOnYourSide[15]	94.25%
Lifetimes[15]	94.55%
sandee3[15]	89.82%
sopr44_1[16]	93.3%
quar48_1[16]	80.1%

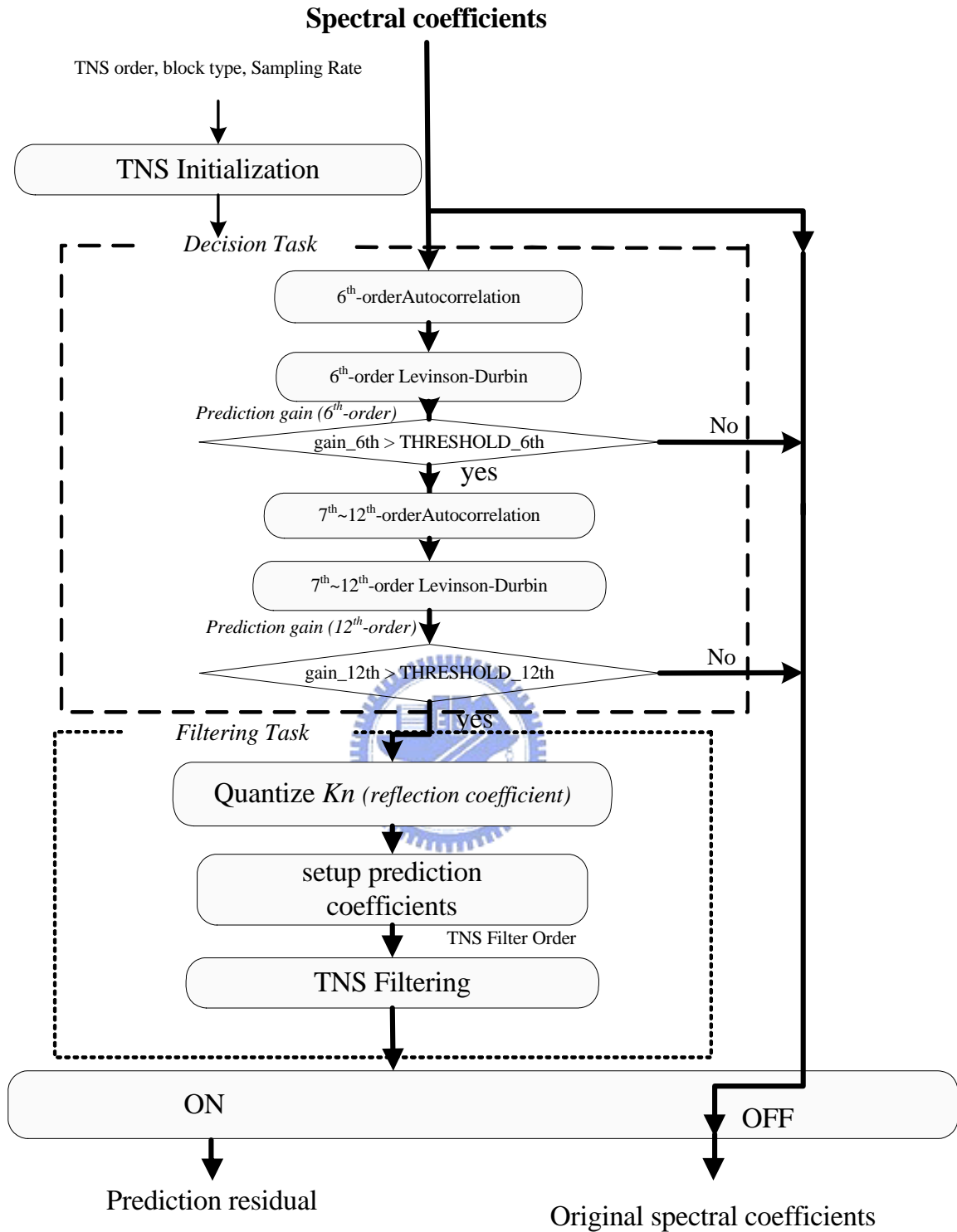


Figure. 24 The flow chart of simplified TNS

For such an early-decision approach, four cases can be taken into consideration in comparing with the original 12th-order decision approach, listed in (1)~(4).

- (1) 12th-order LPC predicted TNS filtering as “turned-off”, and 6th-order LPC predicted TNS as “turn-off”.
- (2) 12th-order LPC predicted TNS filtering as “turned-off”, but 6th-order LPC predicted as “turn-on”.
- (3) 12th-order LPC predicted TNS filtering as “turned-on”, and 6th-order LPC predicted TNS as “turn-on”.
- (4) 12th-order LPC predicted TNS filtering as “turned-on”, but 6th-order LPC predicted TNS as “turn-off”.

Table. 3 Analysis between simplified and original LPC approach

Test samples	case (1)	case (2)	case (3)	case (4)
AlwaysOnYourSide	91.69%	2.52%	3.11%	2.68%
Lifetimes	91.36%	3.17%	5.19%	0.28%%
sandee3	85.59%	4.22%	7.59%	2.30%
sopr44_1	90.65%	4.15%	5.15%	0.05%
quar48_1	74.16%	7.52%	15.26%	3.06%

From case (1) in Table.3, it can be said that 86.69% of LPC predicted TNS filtering can be decided through a 6th-order LPC. From case (4) in Table. 3, only about 1.7% is mis-prediction using 6th-order LPC. Therefore, from Table.2 and Table.3, we can conclude that there has about $90.4\% \cdot 86.69\% = 78.37\%$ in complexity reduction.

3.5 Mid/Side stereo coding optimization

Whether to apply M/S stereo coding or not is based-on channel correlations. It is known that the higher the channel correlation, the required bits would be less for sum/difference of two channels. In the original implementation method, the decision of applying M/S stereo coding is performed scalefactor band by scalefactor band. It is considered to be tedious for deciding M/S stereo scalefactor band by scalefactor band; hence, we proposed that the decision should be made frame by frame based on the frame energy of channel pair. In addition to the channel correlation, the average energy of the channel pairs is taken into consideration since the lower the average energy of two channels has, the lower benefit of M/S stereo coding will have.

Based on the concepts of channel correlation and level of average energy of two channels, two thresholds are defined in order to decide the switching of M/S stereo. One is *THRESHOLD_AVG* used for comparison of the average energy of channel pair, the other is the *THRESHOLD_RATIO* used for comparison of energy ratio of channel pair. Figure. 25 shows the flow chart of simplified Mid/Stereo coding

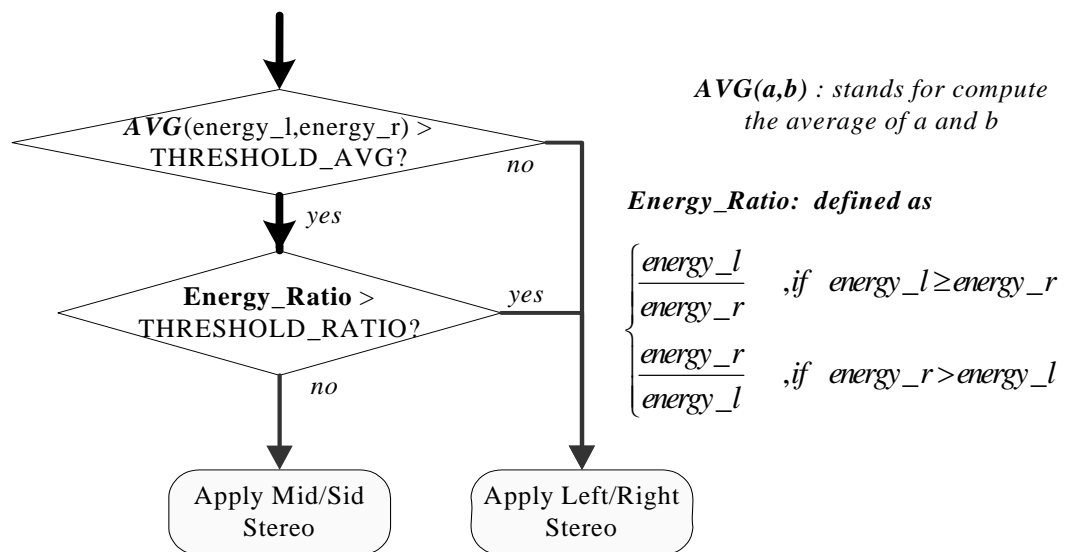


Figure. 25 Simplified flow chart of M/S stereo coding

3.6 Memory Reduction

Except for the computational complexity reduction, the memory requirement is another concern. The reference C code is not optimized for the usage of RAM and ROM. Hence, we use some techniques to reduce the usage of RAM and ROM in the overall encoding process.

Three methods are adopted in memory reduction, list as follows:

- Buffer re-usage by considering the life span of data memory.
- Huffman Tables reduction based on statistics.
- Use different size of data types to store the length and codeword information of Huffman Tables, and separate the processing of length and codeword.

To reduce the usage of buffer, the declared memory blocks are first analyzed and examined when they will be used. In general, the time duration when data is under processing is called the “life cycle (life span)” of data. It can help us applying the buffer re-usage if understanding the life cycle of processing data. Figure.26 illustrate the concept of buffer re-usage for different data with different life span. For the AAC encoder, the input buffer and output buffer is the largest declared memory blocks. Therefore, other temporary data, such as temporary real/imaginary-parts data of FFT process, quantized spectral lines, and coded data...etc; all of them can utilize the same block of memory space at the different time periods.

Huffman tables are the other important part for memory reduction. There are 12 Huffman tables used for the AAC encoder. It occupies a large block of memory space. It usually coded the quantized spectrum with 2 or 4 coefficients as a coding unit in AAC encoder. Generally speaking, two or three tables will be provided to the encoder as choices of quantized spectral coefficients and compute required bits respectively. The table with less required bits will be

chosen finally. This method can indeed save the bits and apply the saving bits to the distortion signal. However, we perform some experimental tests using different Huffman tables to encoded spectral data. Based on statistics result, it is found that there is no need to provide so many tables as choices; with max absolute value of coefficients ranges from 8~12 only need to use the codebook.10, and the codebook.9 can be removed without listening distortion.

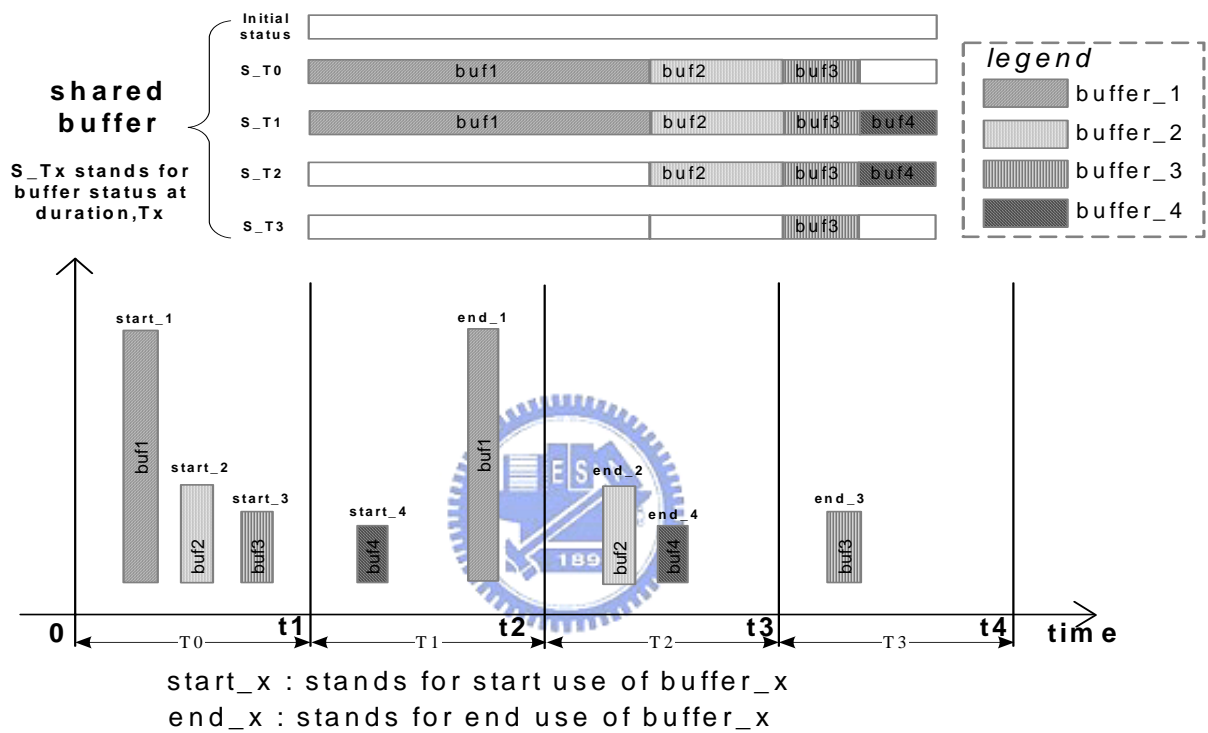


Figure. 26 Simple illustration of buffer re-usage

Another way to reduce memory usage is to modify the declaration of data type width for storing each Huffman table entries. We split two elements (*length* and *codeword*) in Huffman coding into separated arrays for split processing. Due to the dynamic range of *length* elements are generally 0 ~ 255 , only the register with 8-bit wide is need.

After removing the codebook.9 and separate processing of *length* and *codeword* in Huffman table, the size of Huffman table can be reduced from 5932 Bytes to 4327 Bytes. About 1.6 KB memories are saved.

CHAPTER 4. DUAL-CORE PROCESSOR IMPLEMENTATION

In this chapter, the hardware and software development environment are briefly introduced. The target board is a dual-core processor architecture with ARM Core and DSP Core built-in. With the support of dual-core architecture, our implementation of MPEG-2/4 LC AAC will not only focus on ARM core, but the DSP core also. Therefore, the fixed-point optimization based on 16-bit arithmetic will be introduced. In considering of increasing the flexibility of the applications, the operating system environment is built up. Based on the software architecture, two solutions of AAC recorder system are implemented on the target board. The recording system is firstly implemented by ARM core and then using the inter-processor communication scheme to implement the recorder system between ARM core and DSP core.

4.1 Hardware Environment

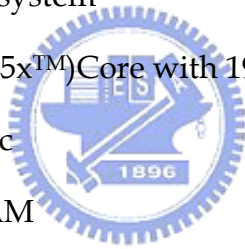
4.1.1 OMAP 5912 OSK

The OMAP5912 is a development platform highly integrating with hardware and

software environment. It is designed for high-efficiency signal processing and high-level portable device application. With the dual-core hardware architecture, it provides better real-time applications. In order to improve the overall system performance, different workload can be allotted to the different computational unit.

The chip of OMAP5912 adopts the OMAP gigacell revision 3.2 which integrate the RISC ARM926EJ-S with DSP TMS320C55x Processors. The functional block diagram of the OMAP5912 is shown in Figure.27. The OMAP5912 dual-core architecture utilizes a Low-Power, and High-performance CMOS Technology. Some essential features of the OMAP5912 are listed as follows:

- ARM926EJS Core with 192MHz maximum frequency , supporting multiple operating system
- TMS320C55x™ (C55x™)Core with 192MHz maximum frequency
- TLV320AIC23 codec
- 32 MBytes DDR RAM
- 32 MBytes on board Flash ROM
- 4 Expansion connectors (bottom side)
- RS-232 serial port
- 10 MBPS Ethernet port
- USB port
- On board IEEE 1149.1 JTAG connector for optional emulation



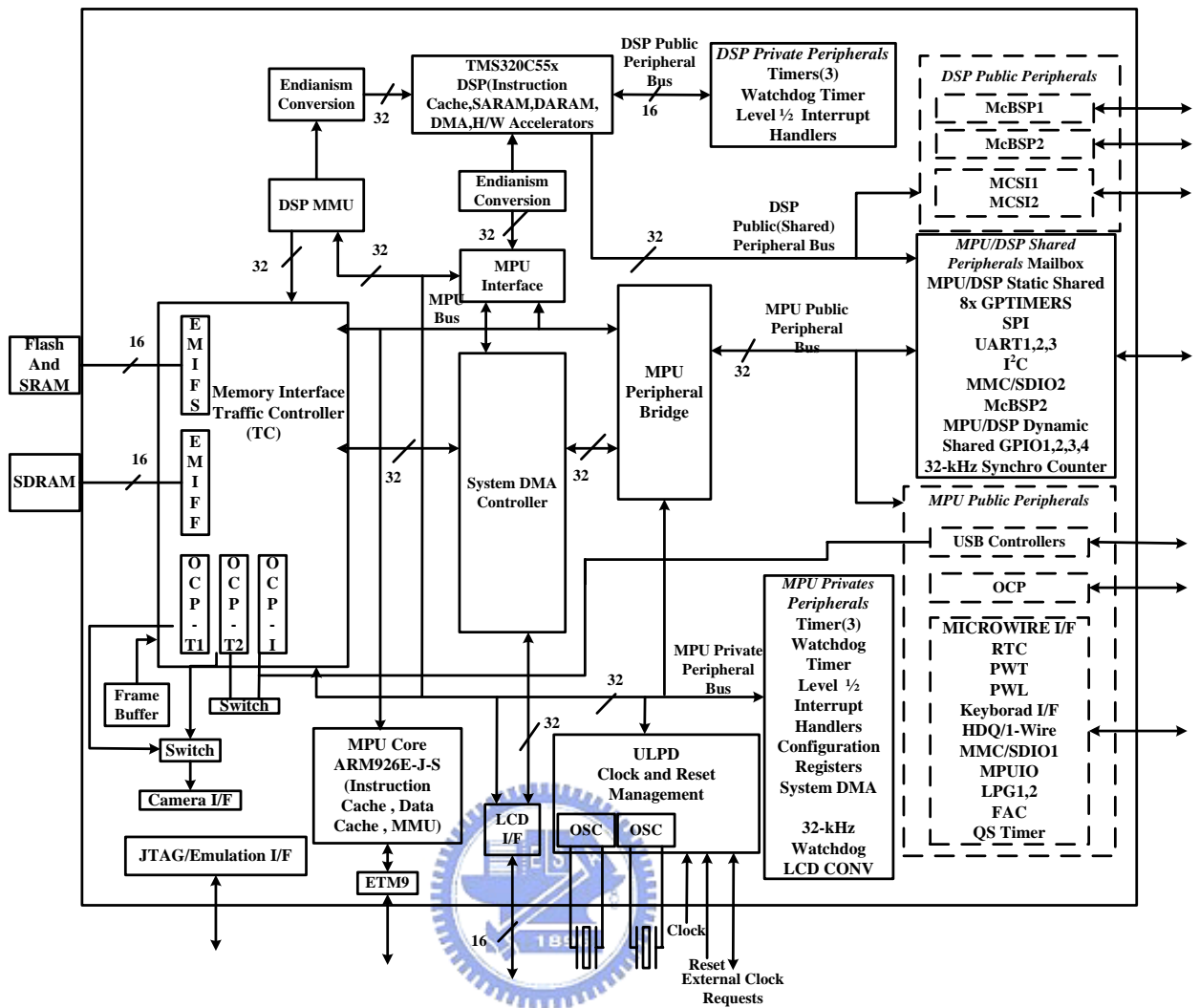


Figure. 27 The functional block diagram of the OMAP5912 [17]

4.1.2 DSP subsystem

With regarding to the dual-core processor OMAP5912, the RISC ARM926EJ-S provides the operating frequency up to 192MHz. It is used for host control processor that handles the operating system and the most peripheral interface access. The DSP subsystem in OMAP5912 adopts the Texas Instruments' TMS320C55x DSP chip. The C55x CPU consists of four processing units, an instruction buffer unit (IU), a program flow unit (PU), an address-flow unit (AU), and a data computation unit (DU). These units are connected to 12 different address and data buses as shown in Figure.28.

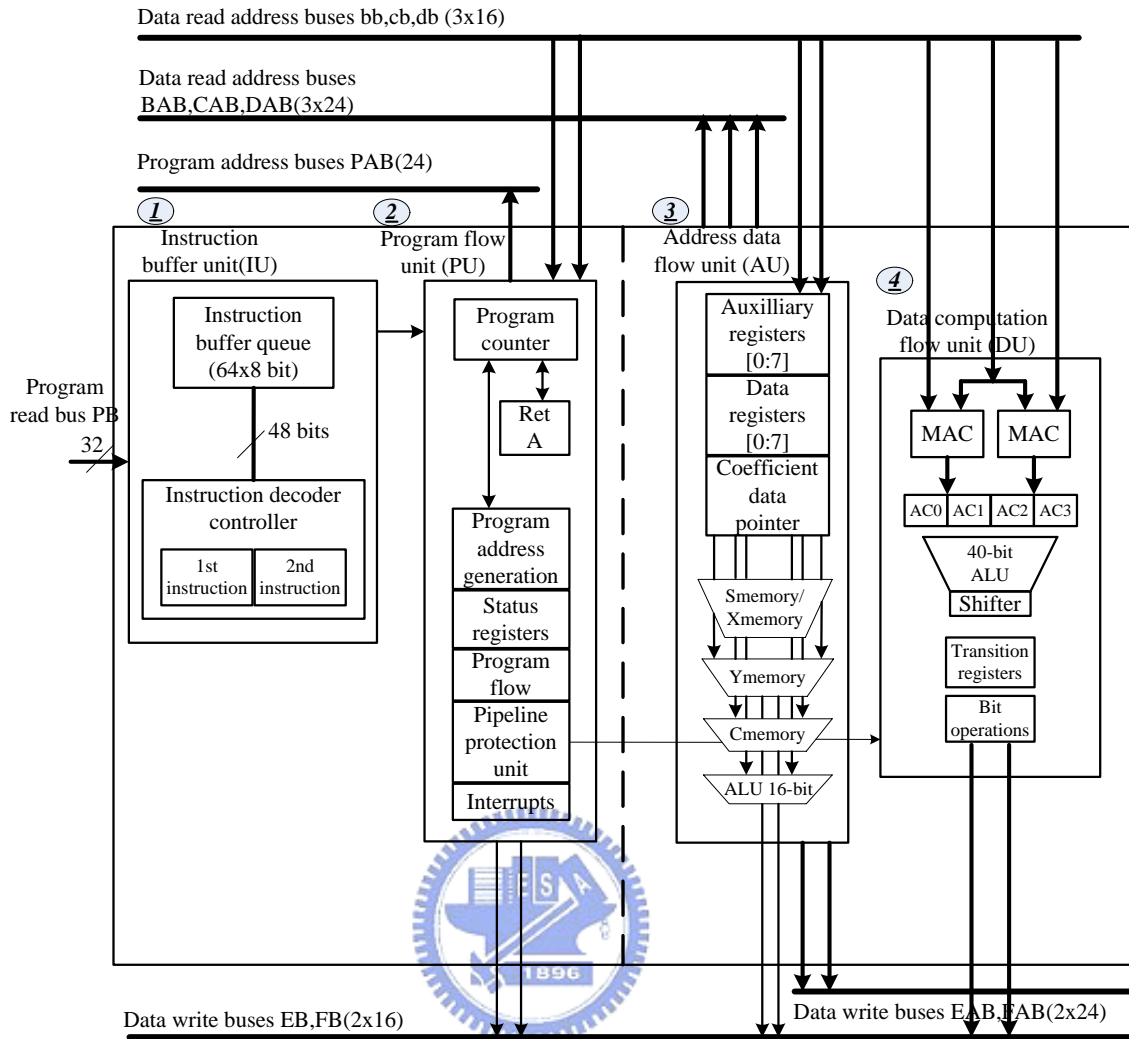


Figure. 28 The TMS320C55x DSP core architecture [18]

Some essential features of the C55x device are listed below [18]:

- A unified program/data memory map. In program space, the map contains 16M bytes that are accessible at 24-bit addresses. In data space, the map contains 8M words that are accessible at 23-bit addresses.
- An input/output (I/O) space of 64K words for communication with peripherals.
- Software stacks that support 16-bit and 32-bit push and pop operations. You can use these stack for data storage and retrieval. The CPU uses these

stacks for automatic context saving (in response to a call or interrupt).and restoring (when returning to the calling or interrupted code sequence).

- A large number of data and address buses, to provide a high level of parallelism. One 32-bit data bus and one 24-bit address bus support instruction fetching. Three 16-bit data buses and three 24-bit address buses are used to transport data to the CPU. Two 16-bit data buses and two 24-bit address buses are used to transport data from the CPU.
- An instruction buffer and a separate fetch mechanism, so that instruction fetching is decoupled from other CPU activities.
- The following computation blocks: one 40-bit arithmetic logic unit (ALU), one 16-bit ALU, one 40-bit shifter, and two multiply-and-accumulate units (MACs). In a single cycle, each MAC can perform a 17-bit by 17-bit multiplication (fractional or integer) and a 40-bit addition or subtraction with optional 32-/40-bit saturation.
- An instruction pipeline that is protected. The pipeline protection mechanism inserts delay cycles as necessary to prevent read operations and write operations from happening out of the intended order.
- Data address generation units that support linear, circular, and bit-reverse addressing.
- Interrupt-control logic that can block (or mask) certain interrupts known as the maskable interrupts.

4.2 Software Environment

The implementation of the recording system includes the user-applications handling the peripheral, memory access through ARM core, and kernel encoder program execution by DSP core. Besides, the communication and synchronous scheme between ARM and DSP are needed.

4.2.1 TI Software Development Tools--Code Composer Studio

The Code Composer Studio (CCS) is a software integrated development environment (IDE) for building and debugging programs. CCS extends DSP code development tools by integrating editor, debugger, simulator, and emulation analysis into one entity. We can use CCS to develop and debug the projects. Some features of its functionality are listed below.

The details can be found in [19].



- Real time analysis and Debugging
- Provide debug options such as step over, step in, step out, run free.
- Compile codes and generate Common Object File Format (COFF) output file.
- Support optimized DSP functions such as FFT, filtering, convolution and some mathematical operations
- Count the instruction cycles between successive profile-points
- Arrange code/data sections into different memory space by linker command file.

We use CCS tool to simulate the encoder on desktop and emulate the result on target platform. By using the profiler provided by CCS, the accurate execution clock cycle count can be evaluated. Figure. 29 shows the software development flow of using CCS.

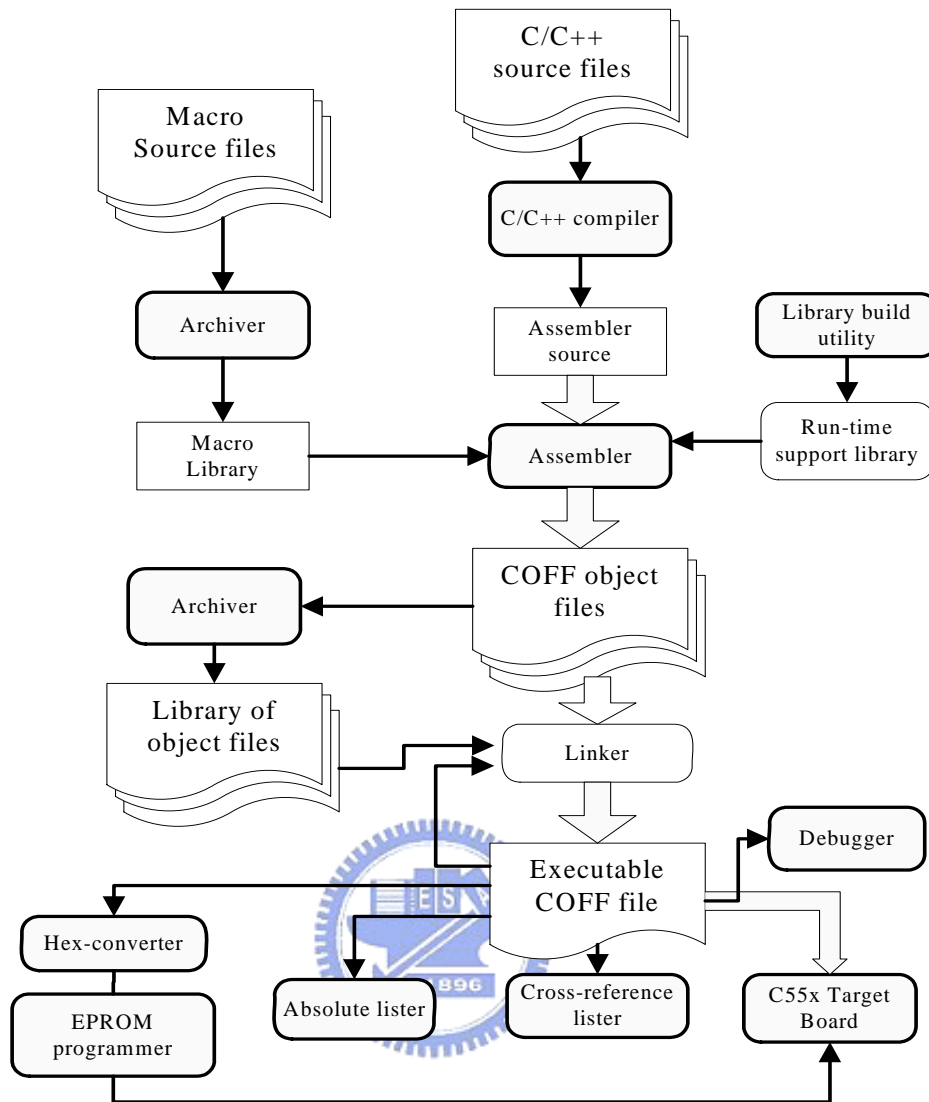


Figure. 29 The C55x CCS Development Flow [19]

4.2.2 Operating System

It will increase the functionality and portability of system by integrating the operating system with hardware platform. With the help of operating system, we could easily add any specific function without considering the low-level memory or devices driver issues, because all of the bottom layer issues are properly handled by operating system and device driver.

Due to the MPU of the OMAP5912 is ARM core, the “Linux operating system” and the Cross-compiler “arm-linux-gcc [20]” are adopted. However, the CCS tool is installed on

Windows. To avoid using two desktops, the software, VMWARE [21] which is a virtual machine simulating another desktop is adopted. By setting up the SAMBA [22] which represented for “network neighbor” under Linux environment, the access of files or programs can be easily transferred between Windows and Linux. The OMAP5912 development environment with operating system is shown in Figure. 30.

During the development stage, the setup of operating system makes the target board able to mount the network file system which Motavista 2.4 built-in under Linux. To make the mounting NFS works, the following suites are necessary:

- Montavista 2.4 – provide the Linux 2.4 kernel source and network file system used for OMAP.
- Boot loader -- the boot code of OMAP 5912 , the original version of boot loader is 1.1.1, 「osk5912-u-boot-1.1.1.out」 .
- DSP Gateway Patch file -- patch-2.4.20_mvl31-dspgw2.0.1.bz2
- TFTP server -- a.tftp-0.32-4.i386.rpm b.tftp-server-0.32-4.i386.rpm

The TFTP Server can be used to transfer the boot loader image or kernel image to OMAP5912. The details for operating system setup procedure can refer to [23].

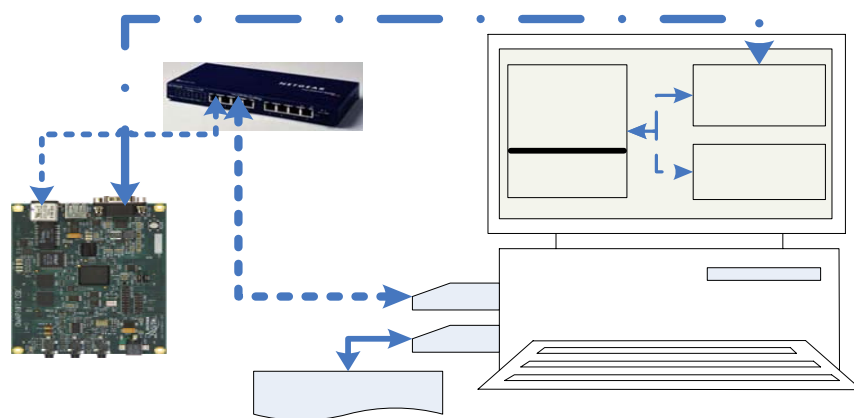


Figure. 30 The OMAP5912 development environment with operating system

4.2.3 DSP Gateway

The OMAP5912 is a dual-core processor with ARM926EJ-S and TMS320C55x inside. In order to make the inter-processor communication possible, the tool “DSP gateway” is applied. DSP Gateway is a program which makes us able to utilize DSP power on OMAP platform from standard Linux kernel. It consists of Linux device driver, DSP side library and utility tools on Linux.

The DSP Gateway can be classified into three portion for discussion, one is the 「DSP Driver and Linux API」, another is the 「DSP Gateway BIOS (tokiliBIOS) and DSP APIs」, and the other is the 「Mailbox transmission command scheme」. The 「DSP Driver and Linux API」 defines the operating methodology and interface for intercommunication. The 「DSP Gateway BIOS (tokiliBIOS) and DSP APIs」 defines the DSP intra-communication and DSP subsystem management. And the 「Mailbox transmission command scheme」 are used for transmission of “DSP Task ID”, “interpreting mailbox command”, and “Task Type”. The details can be found in [24]. Figure. 31 shows the software architecture of DSP gateway.

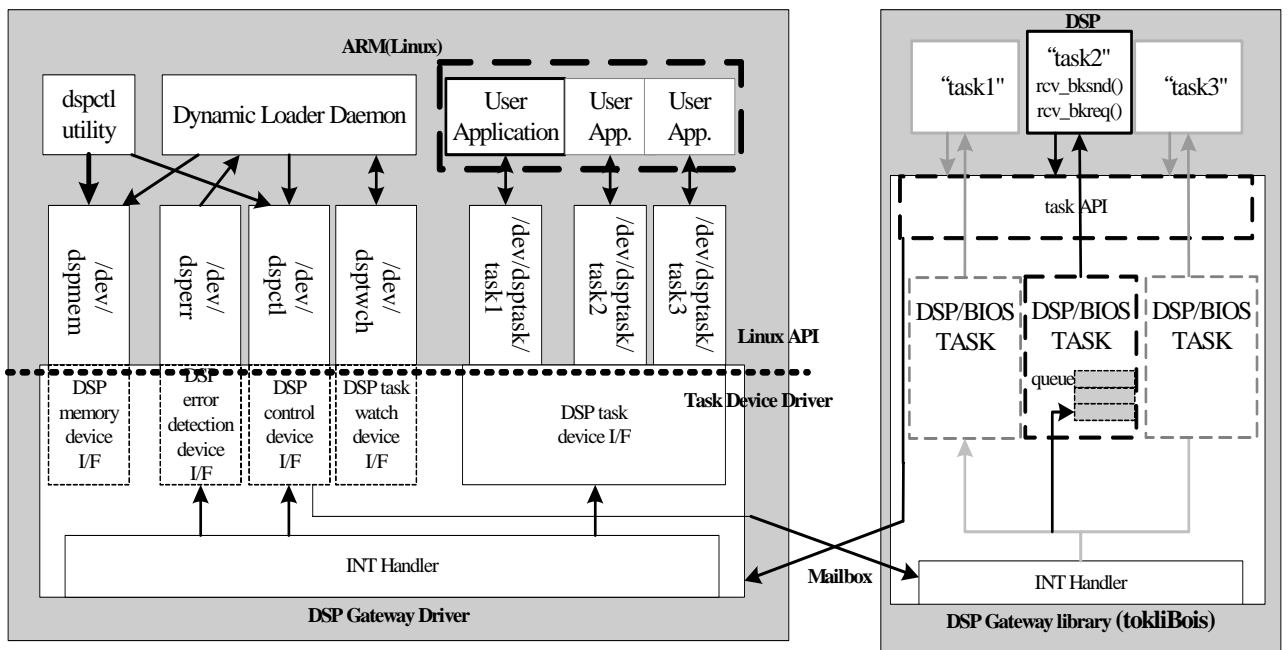


Figure. 31 Software architecture of DSP Gateway Linux APIs [24]

Through the Linux APIs, all the peripheral devices can be viewed as general files, we can use the calls of `open()` , `close()` , `read ()` ,and `write ()` to utilize the DSP core or DSP memory space. Through the mailbox transmission scheme, the control command or user-define protocol can be achieved. In the OMAP5912, four sets of shared mailbox registers are available for communication between the DSP and MPU including:

- Two reads/writes accessible by the MPU, read-only by the DSP.
- Two reads/writes accessible by the DSP, read-only by the MPU.

Each mailbox is implemented with 2x16-bit registers. When a processor writes into a register, it will generate an interrupt to the other processor. The DSP Gateway architecture uses one mailbox register for each direction---One is for ARM to DSP, the other is for DSP to ARM, as shown in Figure. 32. When ARM core writes the data into mailbox register, the ARM core will generate an interrupt (INT5) to DSP.

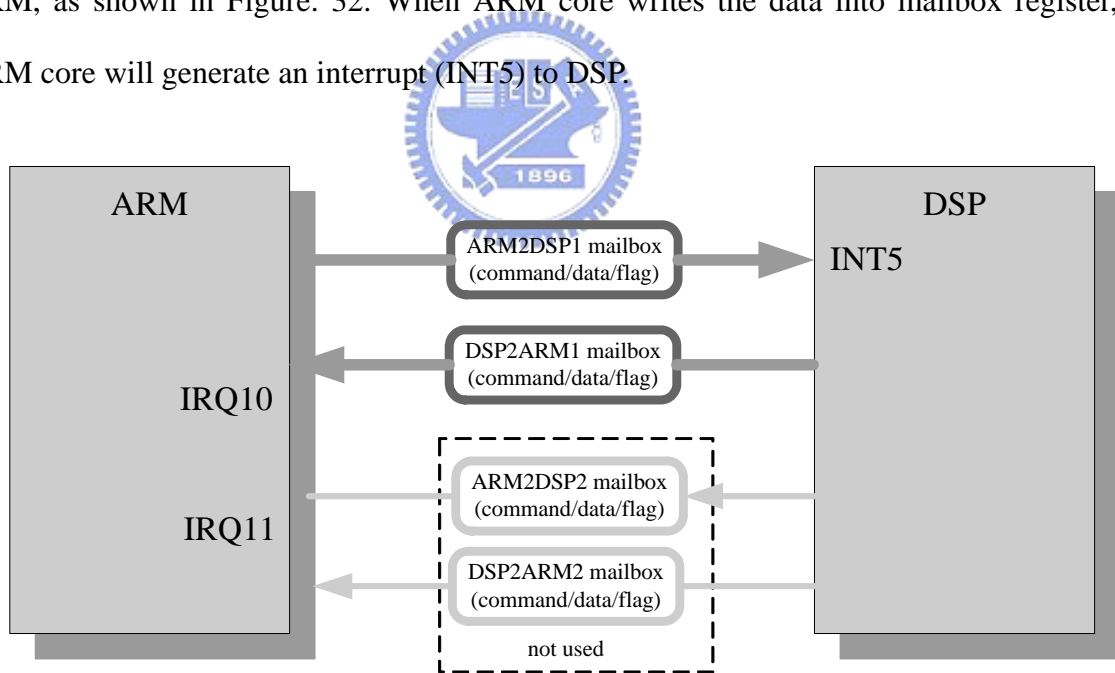


Figure. 32 The mailbox scheme of DSP Gateway [24]

By patching the DSP gateway into Montavista 2.4, there are five device interfaces regarding the DSP gateway can be seen under the path `/dev/`

- *DSP Task Device*

- *DSP Task Watch Device*
- *DSP Control Device*
- *DSP Error Detection Device*
- *DSP Memory Device*

Here, the *DSP Task Device* and *DSP Memory Device* are the two important device interfaces. Any application-specific code that we design will be treated as a task device under the interface of *DSP Task Device* (`/dev/dsptask/taskname`). The *DSP Memory Device* (`/dev/dspmem/`) is another important device interface. For the transmission of few data or commands, only one mailbox queue is quit enough. Nevertheless, it is impossible to use one mailbox register for transferring large amount of data between two processors. Considering large memory movement and data transmission, the *DSP memory device* will be responsible for it. The *DSP memory device* provides the access to the DSP memory space for the DSP program loader in Linux side. When executing a DSP program, the first step is to load DSP binary code to the pre-defined internal or external memory sections (i.e. DARAM, SARAM, EXMEM). These operations are accomplished through this device. For loading binary code to external SDARM, the memory mapping must be done once by using *ioctl* to *DSP memory Device*. Figure. 33 Shows the concept of DSP internal memory space and external memory mapping.

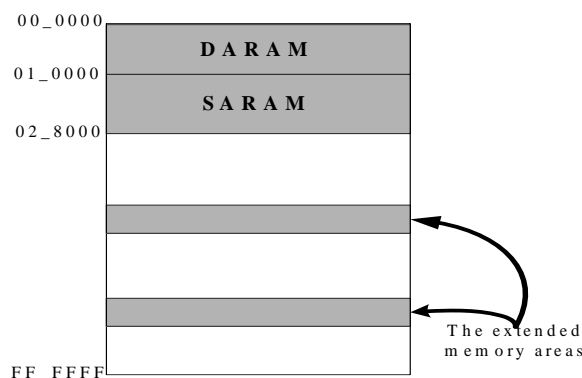


Figure. 33 The DSP memory space [24]

4.3 Fixed-point consideration and DSP Optimization

Due to the DSP processor built in OMAP5912 is a 16-bit fixed-point processor; it only supports the integer mathematical operations. Using the integer operations to emulate the floating point operations is time-consuming and high-cost in ROM size. Therefore, converting the floating-point programs to fixed-point one is necessary in order to make MPEG-2/4 LC AAC execute normally and efficiently. In this section, some fixed-point implementation based on 16-bit arithmetic consideration is discussed and the DSP assembly optimization techniques are presented in order to speed up the encoder.

4.3.1 Fixed-point implementation based on 16-bit arithmetic

Table. 4 shows the data type and the corresponding width of TMS320C55x for supporting C code. It is noted that the data type size of *char* and *int* are both 16-bit width. These might result in precision loss during the block of audio processing or data shuffle if the data type is not properly used.

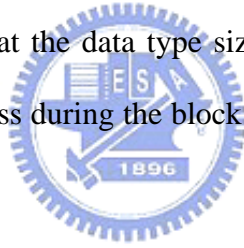


Table. 4 C data type of TMS320C55x [25]

data type	Char	short	int	long	long long	float	Double
width	16bits	16bits	16bits	32bits	40bits	32bits	64bits

Basically, the TMS320C55x performs 16-bit arithmetic operation. However, the double-precision, i.e: 32-bit arithmetic operations provide more accuracy for processing data but increase the computational complexity. In order to perform the 32-bit arithmetic computations such as 16-bit x 32-bit multiplication or 32-bit x 32-bit multiplication, it must

use the 16-bit x 16-bit multiplication as the basic operation, as shown in Figure. 34 and Figure. 35. The complexity of double-precision multiplication will take much more times of the complexity using single precision multiplication.

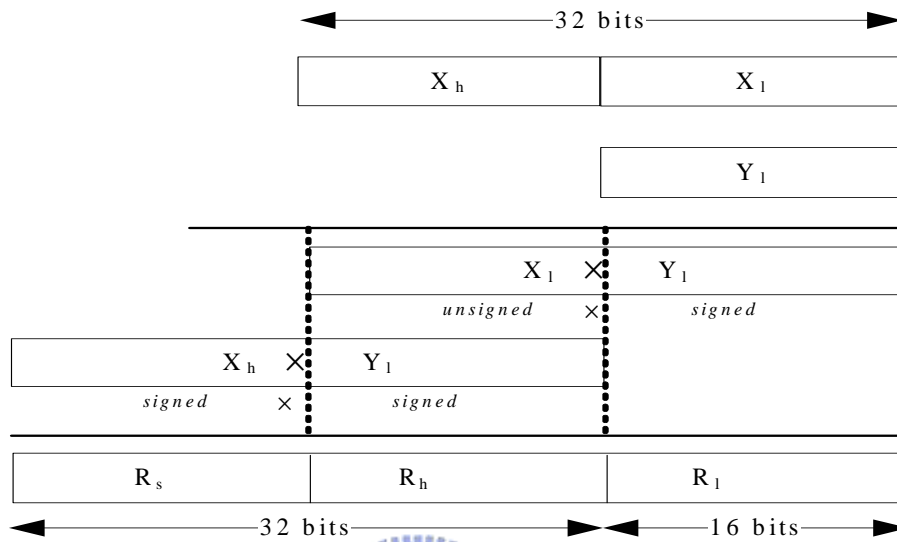


Figure. 34 Double precision multiplication, R (32-bit) = X (32-bit) x Y (16-bit)

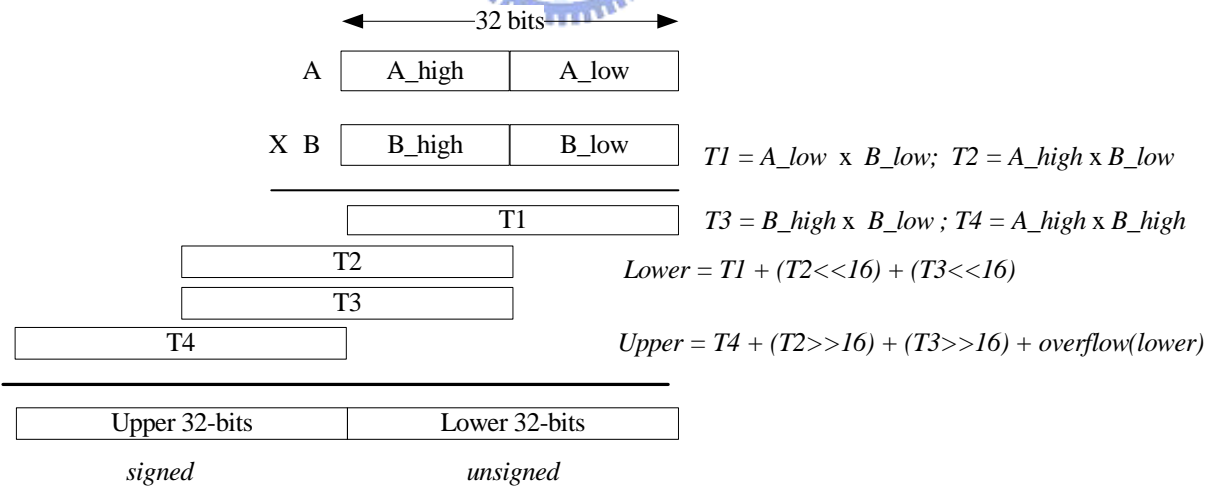


Figure. 35 Double-precision multiplication, X (32-bit) x Y (32-bit)

To determine the data precision of the AAC encoder, it is divided into five stages. The PCM sample is a 16-bit data, therefore, the input PCM samples are viewed as $(16.0f)_{16}$

(format $(M.N)_s$ stands for that using M bits to represent the integer part, and N bits to represent the fraction part, and the suffix S is the total width of data type). At the first stage of AAC encoder, the PCM samples are windowed by cosine table which is viewed as $(1.15f)_{16}$ owing to the data range of cosine table is $-1 \leq \cos(x) \leq 1$. The format of windowed data would be $(16+1 \cdot 0+15)_{32} = (17.15)_{32}$. The format decision of input and output signals at each stage is based on the calculation and statistics of their dynamic range of values. Figure.36 shows the data precision of the proposed AAC at each stage.

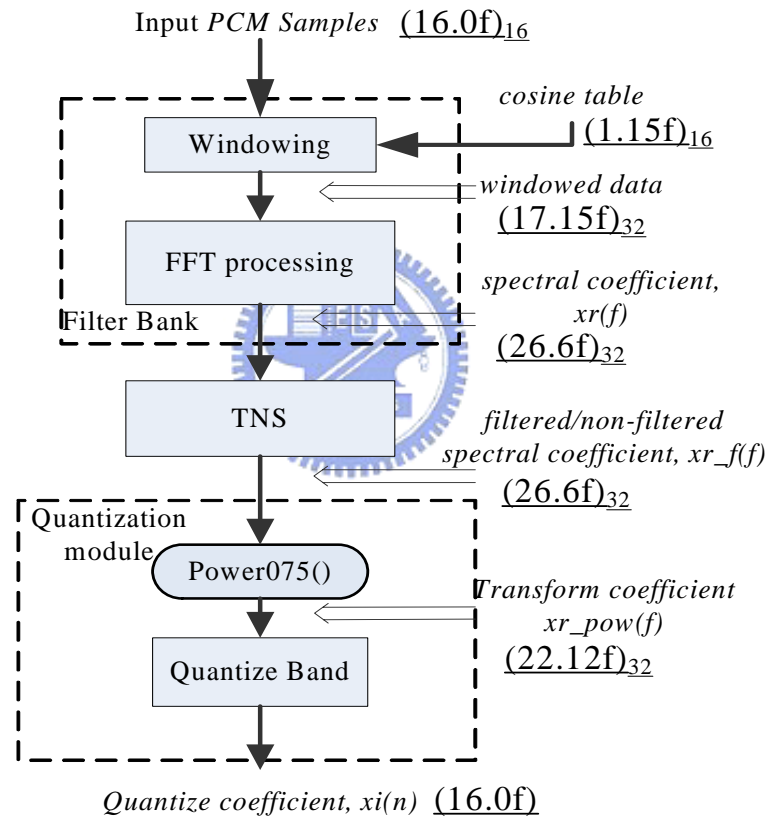


Figure. 36 The data precision of the proposed AAC encoder

For the target processor, TMS320C55, the signed data representation is ranged from -2^{31} to $2^{31} - 1$. Due to the limited data range, the mathematical processing must be cautious, especially while performing the multiplication and accumulation.

In the overall AAC encoding process, in considering of data precision, the fixed-point format of the spectral coefficient $xr(f)$ are defined as $(26.6f)_{32}$ as shown in Figure.36. With the format of spectral coefficient $xr(f)$ defined as $(26.6f)_{32}$ using 32-bit data type, the use of 32-bit data type will be not wide enough for calculating the autocorrelation, energy of total frame or energy of each scalefactor band. To solve the problems arose from fixed-point optimization; a simple scaling unit is applied on spectral coefficient before performing the computations. There are three parts that we applied the scaling unit as shown in Figure.36.

The first one is the frame energy calculation $\sum_{all\ freq.\ lines} (xf(i))^2$. It will calculate the total energy of each channel, and such information will be used for decision of switching M/S stereo coding and for computing allowed-distortion. The second is the autocorrelation computation $\sum_{order} \sum_i xf(i) \cdot xf(i + order)$ used for Levinson-Durbin recursion. The third is the energy calculation of each scalefactor band computed as $\sum_{sfb} (xf(sfb))^2$.

The scaling unit is implemented by detecting the dynamic range of the processing group of data at run-time. It will determine the maximum value $xf_{i,max}$ of the processing group of data, $xf(i)$ at first. And the processing group of data will right shift by a run-time determined value $SHIFTVAL$ according to the detected exponent value K to avoid the overflow during multiplication and accumulation. Figure. 37 shows the determination of value K .

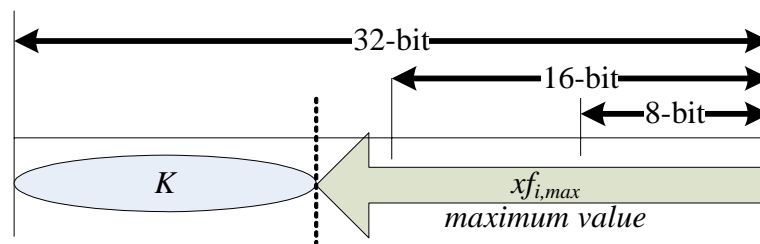


Figure. 37 Representation of exponent detector

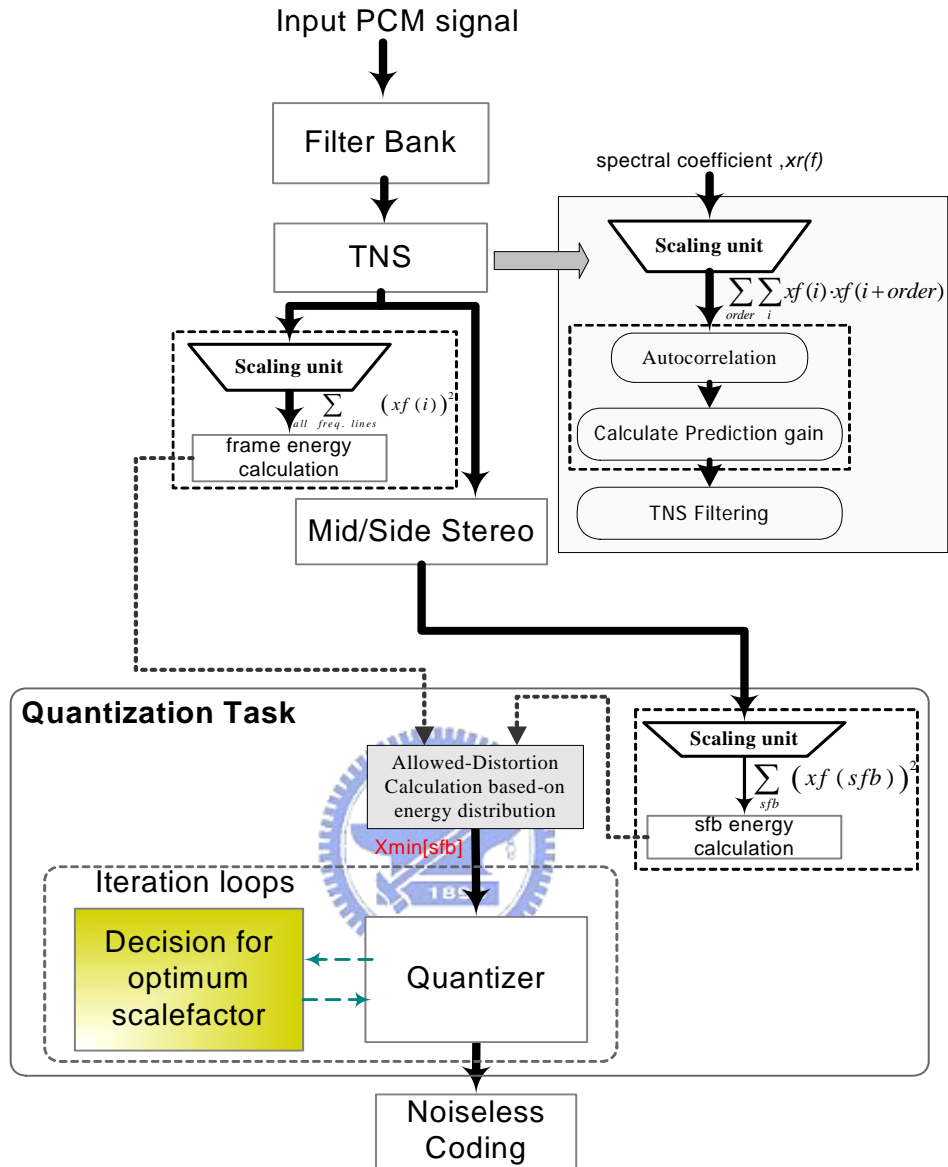


Figure. 38 The scaling unit applied in AAC encoder

For the TMS320C55x hardware architecture, it supports computing the leading zeros of the data in a single cycle.

$$\text{Syntax: } EXP \ AC_x, T_x \quad (15)$$

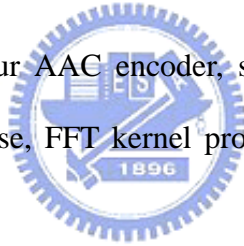
The mnemonic instruction EXP can compute the exponent of the source accumulator AC_x in the D-unit shifter, and the result of the operation will store in the temporary register, T_x . The applying of scaling unit before the processing data will dynamically scale the data or

to say, changing the fixed-point format dynamically. In this way, the overflow of the multiplication and accumulation can be avoided, and the unsigned range of data could be limited to $2^{32} - 1$.

4.3.2 Assembly and compiler optimization techniques

The CCS tool offers high language support by transforming C code into more efficient assembly code. The compiler options provide the choices to reduce code size or execution time. Four optimization levels are provided in CCS compiler: register (-o0), local (-o1), function (-o2), file level (-o3). File level (-o3) is the highest one of the available optimization. With file level optimization, various loop optimizations are performed, such as software pipelining, loop unrolling, and various file-level characteristics are also used to improve performance.

In the implementation of our AAC encoder, several blocks are optimized by writing assembly code, such as bit-reverse, FFT kernel processing, M/S stereo Coding, and Frame Energy Calculation...etc.



Regarding the assembly optimization, for the bit-reverse optimization, a special address mode called bit-reversing addressing mode is applied, as shown in Table. 5

Table. 5 Description of bit-reversing addressing mode [25]

Operand syntax	Function	Description
* $(ARx-T0B)$	address = ARx $ARx = (ARx-T0B)$	After access, $T0$ is subtracted from ARx with reverse carry(rc) propagation
* $(ARx+T0B)$	address = ARx $ARx = (ARx+T0B)$	After access, $T0$ is added from ARx with reverse carry(rc) propagation

The off-place and in-place bit-reverse are both written in assembly code. (The off-place represents that the input and the output buffer are the same and in-place stands for the same memory block of input and output buffer.). Table.6 shows the implementation result written by C code and assembly code. The speedup is about 4.5 times comparing off-place bit-reverse with pure-C code.

Table. 6 Bit-reverse Implementation result using C and ASM

Method	pure-C code	In-place ASM	Off-place ASM
cycle counts	114123	5901	2583

In order to utilize the features of DSP hardware architecture, the features of parallelism and zero-overhead looping are adopted in the written assembly code. Considering the parallelism optimization, several types of parallelism are provided by C55x.



- Using the build-in parallelism within a single instruction (Because of Dual-MACs architecture).

```
MPY *AR0, *CDP, AC0
:: MPY *AR1, *CDP, AC1
```

(16)

- The two lines shown in (16) are viewed as a single instruction; the data referenced by AR0 is multiplied by coefficient referenced by CDP. At the mean time, the data referenced by AR1 is also multiplied by the same coefficient CDP.

- Using user-defined parallelism between two instructions. (Because of different functional unit and proper data bus connection)

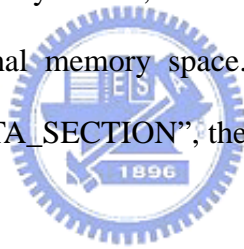
```
MPYM *AR1-, *CDP, AC1
|| XOR AR2, T1
```

(17)

- In the assembly optimization, C55x ASM provides the notation “||” to instruct the compiler that the instruction behind can execute in parallel. In the (17), the first instruction performs a multiplication in the D-unit and the second instruction performs a logical operation in the A-unit ALU.

Another assembly optimization technique lies on zero-overhead looping, the C55x provides repeat block of instructions such as *RPT* or *RPTBLOCAL*. The instruction will repeat a block of instructions number of times specified some specific registers.

In addition to the assembly coding techniques, the implementation of DSP code includes the design of memory allocation. It is known that the accessing data located at external memory will take much more clock cycles than accessing them located in internal memory. In order to reduce the external memory access, the linker command file are modified to place most of the codes in the internal memory space. With the compiler directives such as “PROGRAM_SECTION”, “DATA_SECTION”, the memory allocation can be easily done in C code.



Although there still have other optimization techniques, only the optimization techniques that applied in the written assembly code are introduced in this section. For more details information can be found in [25][26].

4.4 Recording System implementation

AAC recording system is implemented and verified on OMAP5912 by two solutions. One is to implement the recorder by ARM core, and the other one is to implement the recorder between ARM and DSP. With the help of Linux operating system and the DSP

gateway for bridging the inter-communication between two processors, the host terminal can load the application-specific codes into DSP memory space at run-time and to instruct DSP for initialization, start, stop, or IDLE. The development of the system includes the user-application on ARM side (Linux) and the specific-application on DSP side. The development flow is shown in Figure.39. In the following section, the software architectures for recording system are described.

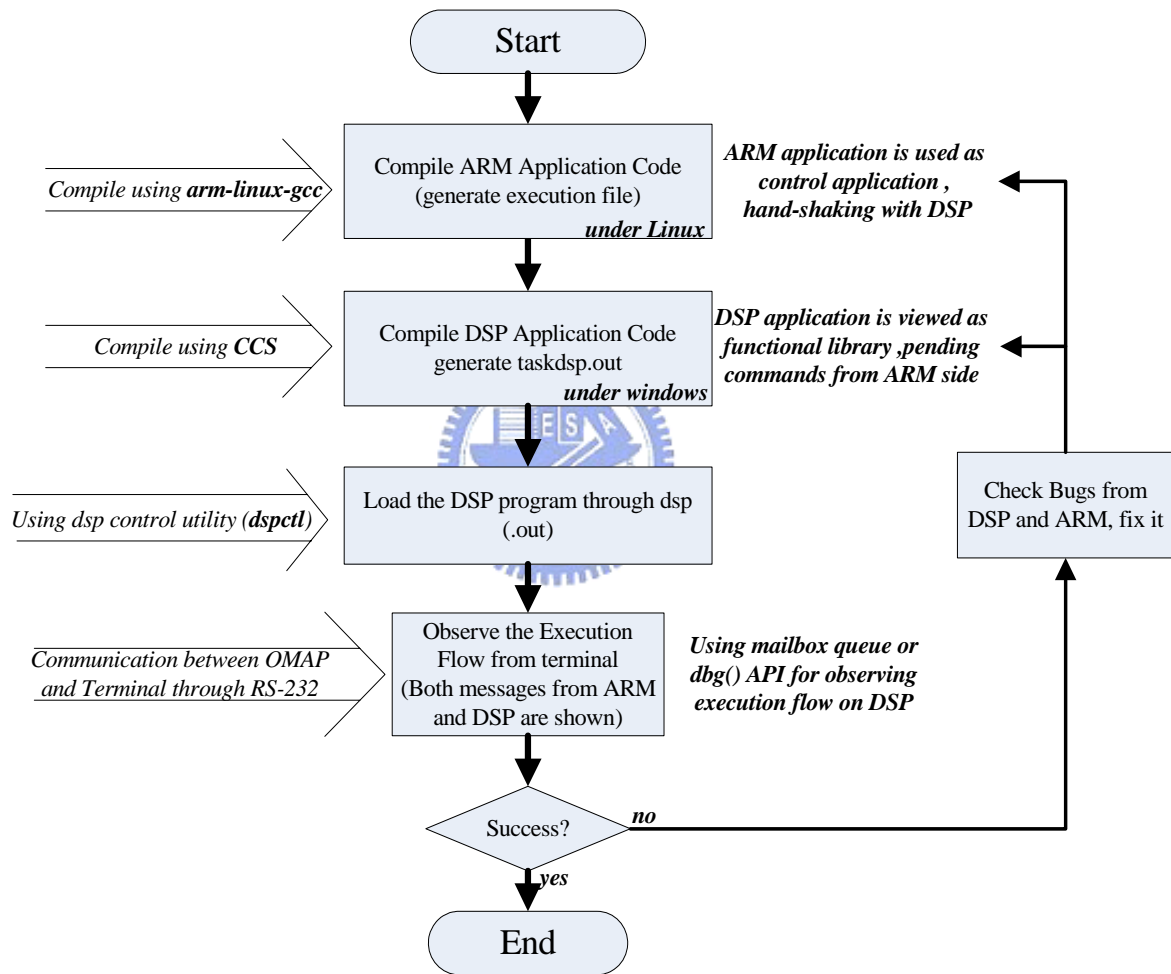


Figure. 39 The development flow of OMAP 5912 with DSP gateway

4.4.1 Recorder implementation by ARM

The difference between file-based encoder and recorder is the input source. The source of file-based encoder is from the PCM audio samples already exist on memory device. Hence, the input PCM data is buffered. However, the source of the recorder is to read PCM samples from line-in and then encodes these data into AAC file. Owing to the access of /dev/sound/dsp device is non-buffering, two thread programming is necessary. One thread is designed for reading the PCM samples from line-in, the other thread encode the PCM data into AAC file. For the sake of bridging data hand-over between two thread, a buffer queue is applied. Figure. 40 show the program flow of the recording system.

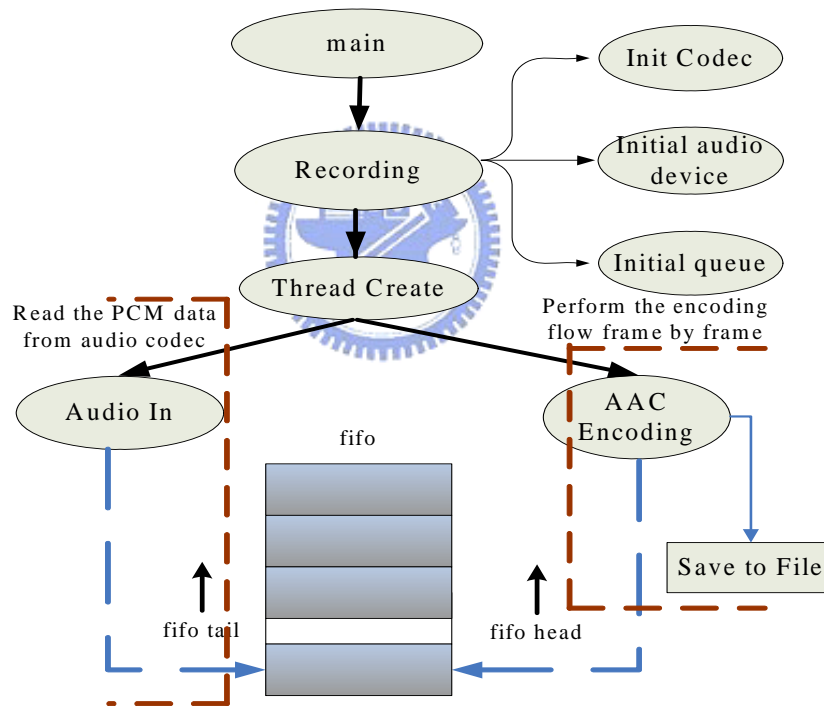


Figure. 40 The multi-thread program flow for recording by ARM.

4.4.2 Recording system between two processors

Another solution for recording system implementation is to allocate the encoder workload on DSP core. Under such software architecture, Linux side is responsible for

accessing the memory device, audio codec, and other peripherals. Figure. 41 is the block diagram of the total recorder system.

For the user program executing by ARM core, one thread is created for pending the message or information from DSP through Inter-processor buffer (IPBUF) defined by DSP gateway and mailbox buffer in user-program, as shown in Figure. 41 and Figure. 42.

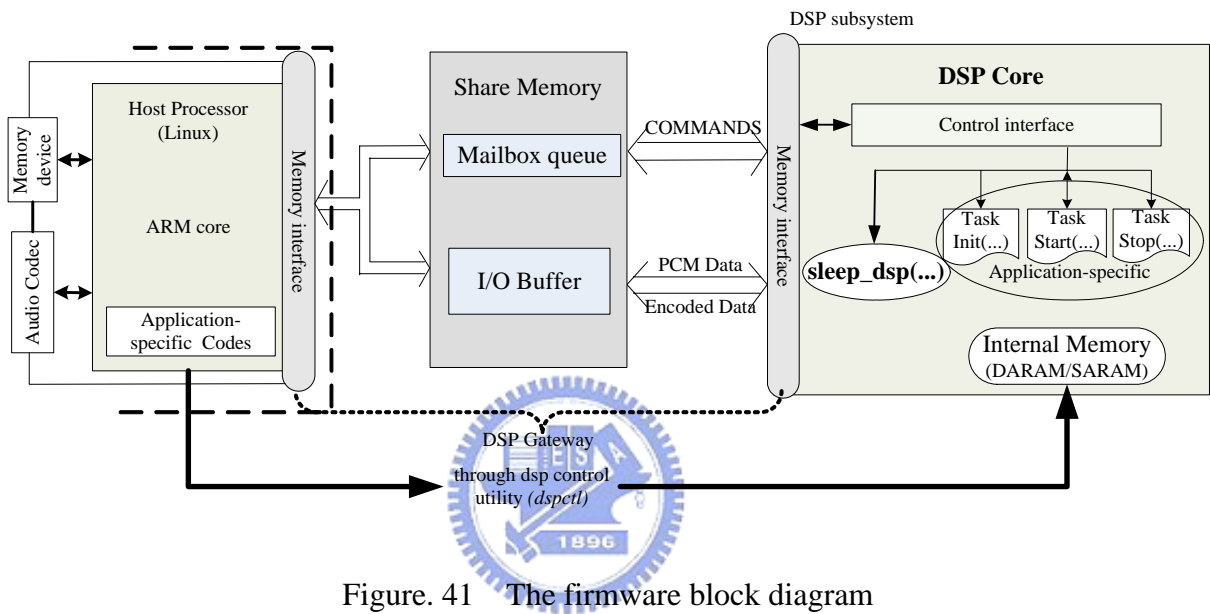


Figure. 41 The firmware block diagram

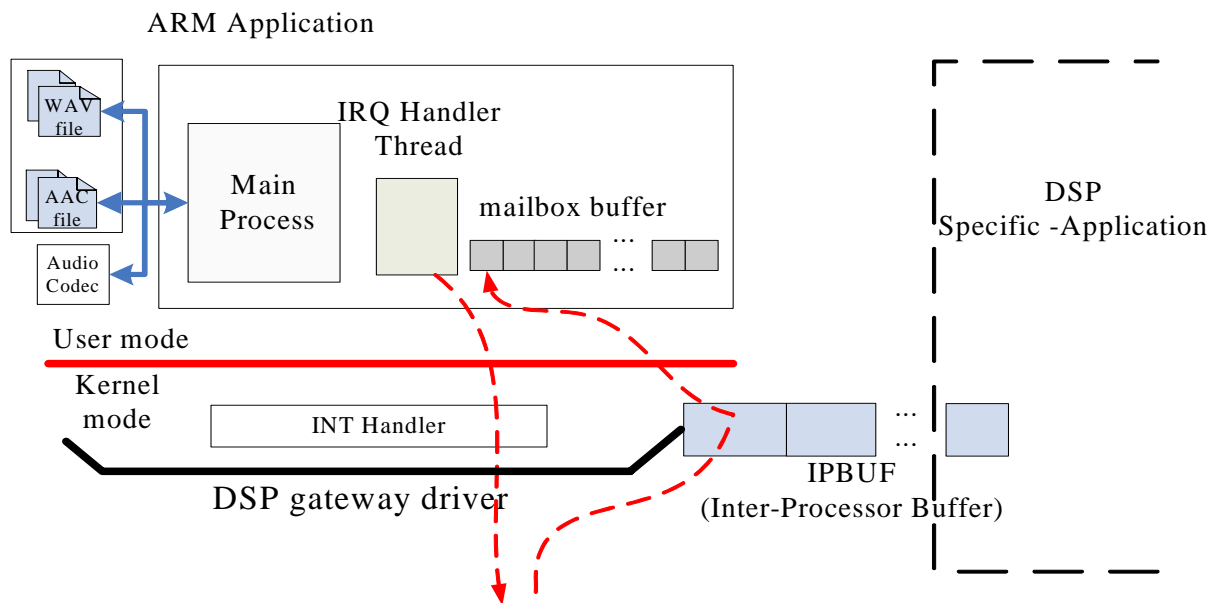


Figure. 42 The mailbox IRQ Handler

It is the mailbox queue and arranged I/O buffer that take care of the transmission of DSP control command and large amount of data bit streams, respectively.

For receiving commands from the DSP task or sending commands to the DSP task, both the passive way protocol in the mailbox of DSP gateway are taken. For this reason, the DSP task sends BKSND commands only in response to the BKREQ command from ARM. ARM can send BKSND command at any time it wants to send data. Figure. 43 show the mechanism we adopted in mailbox. The detail definitions of mailbox commands are listed in Appendix .

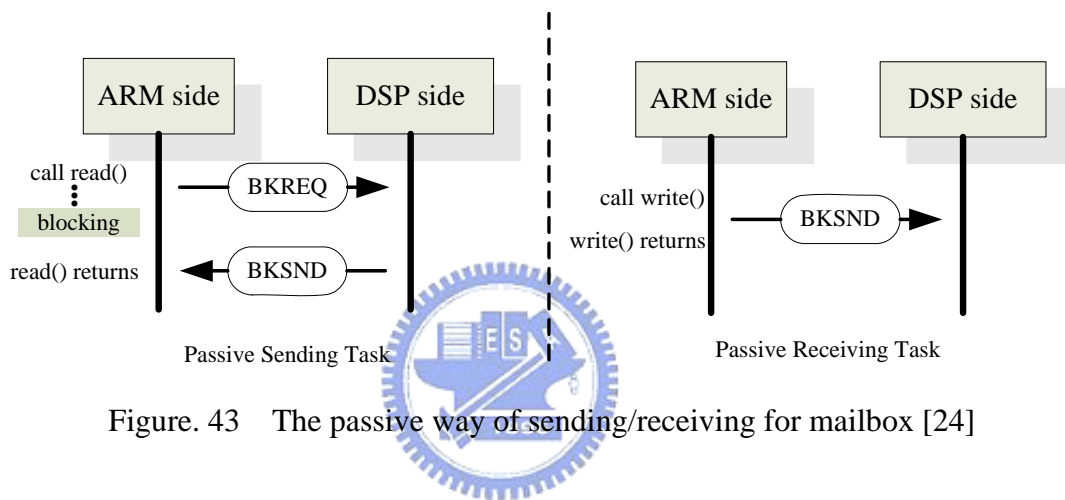


Figure. 43 The passive way of sending/receiving for mailbox [24]

There are four self-defined commands at the mailbox of DSP side, TASK_INIT, TASK_START, TASK_STOP, and IDLE. The previous three commands are application-specific ones that we can modify according to the application codes.

The TASK_INIT is the command used for initialization, including initializing the internal configuration such as sampling rate, bit rate, and channels that post from ARM. And it also performs a simple hand-shaking with user-application at ARM side to return the status information on DSP. The TASK_START command instructs the DSP performing the frame by frame audio encoding. Once the frame encoding is done, the DSP will post a frame finish message to inform the user-application at ARM side. The TASK_STOP instruct the DSP to stop encoding loop and will release the resource usage of DSP.

The IDLE command is used for power-saving consideration. It will notify the DSP to

execute a `sleep_dsp()` routine so that the processor will enter the sleeping mode with less power-consumption.



CHAPTER 5. EXPERIMENTAL RESULTS

Applying the proposed algorithms and techniques to AAC encoder, it is implemented on a 16-bit fixed-point processor. The evaluation of the performance includes two parts. One is the pure encoder performance that the profiler is used to get the accurate clock cycles and the other is the system performance.



The test audio samples are downloaded from EBU website [16], or converted from CD tracks [15]. All samples are stereophonic and sampled with 44.1 KHz, listed in Table. 7.

Table. 7 Test audio samples

Audio Samples	Time	Signal characteristics	Abbreviation
sopr44_1	0:23	Soprano. vocal	SO
quar48_1	0:28	Quartet vocal	QT
bass47_1	0:24	Bass. Vocal	BS
AlwaysOnYourSide	4:15	Pop music with piano background	ALW
Lifetimes	4:12	pop music andantino	Life
sandee3	3:42	Soft rock with female voice	SAN

5.1 Performance Evaluation

➤ Performance for DSP:

In order to get a specific complexity of the encoder, the profiler provided by CCS tool is used to get accurate clock cycle counts. (i.e.: C55x cycle accurate simulator). So as to get better compiler optimization, the file-level optimization (-o3) is adopted. The cycle counts and corresponding MIPS are shown in Table. 8.

Table. 8 performance evaluation on TMS320C55x processor

Samples	SO	QT	BS	ALW	Life	SAN
Cycle counts	4,026,640	4,072,911	4,063,272	4,004,696	4,016,694	3,748,210
MIPS	86.71	87.70	87.50	86.23	86.50	80.71

The cycle counts are the average clock cycles per frame and the MIPS calculation is based on the following computation.

$$\left(\frac{\text{cycle counts per frame}}{\text{samples per frame}} \right) \cdot \frac{\text{SamplingRate}}{10^6} \quad (18)$$

In addition to the evaluation of encoding speed, the memory requirement is the other important evaluation.

From section allocation map file generated by CCS compiler, the *.text* (Code) section of our occupies 32.8KB, the *.bss* (Global & static variables) section occupies 59.4KB, and the *.cinit*(Auto-initialization tables)section occupies 8.1KB,list in Table 9.

Table. 9 The summary of memory section size for proposed encoder

Memory Section	<i>.text</i>	<i>.bss</i>	<i>.cinit</i>
Size	32.8KB	59.4KB	8.1KB

➤ **Performance for ARM:**

Table. 10 list the encoding speed of each test audio samples, all the samples are running at the configuration of bit rate 96Kbps, and Table. 11 shows the code size of the encoder and the recorder using the cross-compiler arm-linux-gcc with different optimization flags.

Table. 10 Encoding speed evaluation

audio samples	Length(sec)	Speed
SO	0:23	1.21X
QT	0:28	1.21X
BS	0:24	1.20X
ALW	4:15	1.17X
Life	4:12	1.18X
SAN	3:42	1.12X

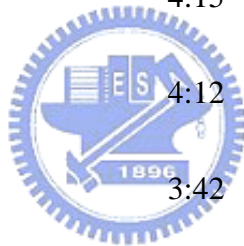


Table. 11 Code Size of encoder and recorder

	-o1	-o2	-o3
File-Based Encoder	65,341	60,425	72,699
Recorder	71,049	66,117	78,391

Note: use cross-compiler arm-linux-gcc, unit: **bytes**

5.2 Quality Evaluation

To evaluate the audio quality of the proposed AAC encoder, ITU-R Recommendation BS. 1378 [28] is adopted as the objective audio quality (ODG) measurement. We use the software objective measurement tool for audio quality tool called EAQUAL [29], which stands for Evaluation of Audio Quality. There are several scales for defining the ODG, Table. 12 list the ranges and meanings.

Table. 12 The Scale of ODG

ODG scale	Quality
0.0	Imperceptible
-0.1 ~ -1.0	Perceptible but not annoying
-1.1 ~ -2.0	Slightly annoying
-2.0 ~ -3.0	Annoying
-3.0 ~ -4.0	Very annoying

The reference codec is the traditional AAC encoder and decoder implemented by FAAC[8] and FAAD[8] in floating point, respectively. The test audio samples will be encoded by reference encoder and the proposed encoder, and then both the encoded file will be decoded by the reference decoder. Two reconstructed PCM audio samples will be compared with the source samples to get the ODG grades using EAQUAL.

Table. 13 list the ODG testing result under the bit rate of 128Kbps, 96Kbps, and 64Kbps, and Figure. 44, Figure. 45, Figure. 46 illustrate the results.

Table. 13 Summary of ODG test

BitRate		Test Audio Samples					
		SO	QT	BS	ALW	Life	SAN
128Kbps	Original	-2.86	-1.17	-2.66	-1.0	-0.80	-0.91
	proposed	-2.23	-2.01	-2.52	-1.26	-1.20	-1.22
96Kbps	Original	-3.46	-2.22	-3.28	-1.96	-0.84	-0.95
	proposed	-3.48	-2.30	-3.37	-2.05	-1.95	-1.38
64Kbps	Original	-3.52	-2.31	-3.38	-2.01	-1.05	-1.13
	proposed	-3.69	-2.65	-3.44	-2.37	-2.60	-2.41



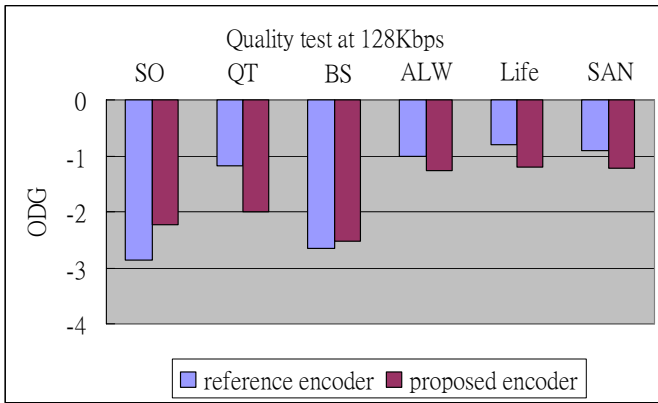


Figure. 44 The ODG result at 128Kbps

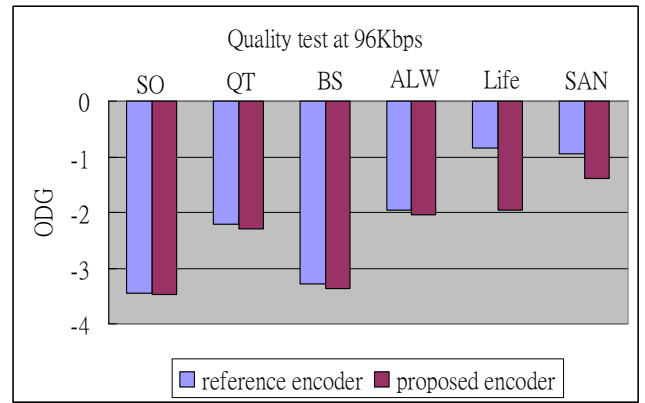


Figure. 45 The ODG result at 96Kbps

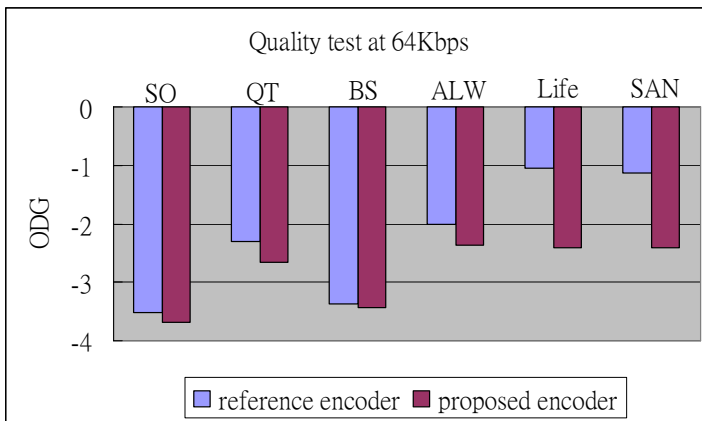


Figure. 46 The ODG result at 64Kbps

For the quality evaluation, it shows that the proposed encoder would not have much quality loss at the bitrate of 128Kbps. However, for the case of 96Kbps and 64Kbps, some audio test samples, i.e.: “Life” and “SAN” might have worse ODG grade. It is because that the data dynamic ranges of such two samples are larger. Using data precision of 16-bit width might not be enough for covering most of the audio samples.

CHAPTER 6. CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

In this thesis, we give:



- Briefly introduce the algorithm of MPEG-2 Advanced audio coding.
- Several algorithms optimization are proposed including, simplified psychoacoustic mode, fast MDCT, simplified TNS tool, simplified Mid/Side coding, and memory reduction techniques in Huffman Tables.
- The fixed-point optimizations based on 16-bit arithmetic are proposed. In order to solve the problem of data precision in encoding process during fixed-point conversion, a scaling unit is applied to adjust the level of spectral signal energy.
- Based on the target DSP processor, a 16-bit fixed-point processor, TMS320C55x, we optimize and ported the AAC encoder using CCS development tools.
- Based on a dual-core hardware architecture, OMAP5912 OSK, the operating system on target platform is build up and provide two solutions to accomplish the recording

system. One is to use multi-thread programming to implement recording by ARM core, the other is to use the inter-processor communication with ARM as the host control processor and DSP as slave processing unit to fulfill the recording.

- Base on 16-bit fixed-point implementation with the proposed optimized algorithm, our encoder require about 86MIPS and 107KB memory. And the recording system by ARM core can achieve at 1X encoding.

6.2 Future works

Our optimization work concentrate on MPEG-2/4 Low-complexity AAC encoder, and implement as a recorder on a dual-core platform, OMAP5192 OSK. The state of the art algorithm, such as Dolby AC-3, HEAAC v1 (aacplusv1), HEAAC v2 (aacplusv2), Microsoft® WMA [30], might apply the same coding techniques as MPEG-2/4 AAC. Besides, the MPEG-2 LC AAC also constitutes the kernel algorithm of MPEG-4 Audio standards. Therefore, the concept of the proposed algorithm in this thesis can be applied to other audio compression algorithm to decrease the computational complexity in order to realize on fixed-point processor.

Regarding to the codec, it mixes up with the C code and assembly code in our encoder implementation. Hence, there still have several coding modules that are not fully assembly optimized. The CCS compiler has its limit for C code to assembly code conversion. Using the assembly language provides us a fully control of the program flow. This is an essential work in the future. In addition to the encoder itself, the software architecture of a recording system is the other issue that reduces the system performance. Our recording system use ARM to access the audio codec and move block of data to DSP as input source. The other way to let

the input source samples feed into DSP directly and move the encoded data with DMA. This will save the time of moving block data and highly increase the performance of recording system.



REFERENCE

- [1] E. Zwicker and H. Fastl, “*Psychoacoustics: facts and models*,” Springer-Verlag, Berlin, Heidelberg, Spring, 1999.
- [2] E. Terhardt, “Calculating Virtual Pitch,” *Hearing Research*, pp. 155-182, 1979.
- [3] ISO/IEC 13818 – 7, “Information Technology – Generic Coding of Moving Pictures and Associated Audio, Part 7: Advanced Audio Coding,” 1997.
- [4] T. Painter and A. Spanias, “Perceptual Coding of Digital Audio”, *Proc. the IEEE*, Vol.88, No.4, pp. 451-515, Apr. 2000.
- [5] N. Jayant and P. Noll, “*Digital Coding of waveforms*”, Prentice-Hall, Englewood Cliffs, NJ, pp.362-370, 1984.
- [6] K. Brandenburg, M. Bosi, S. Quackenbush, L. Fielder, K. Akagiri, H. Fuchs, M. Dietz, J. Herre, G. Davidson and Y. Oikawa, “ISO/IEC MPEG - 2 Advanced Audio Coding”, *J. Audio Eng. Soc.* Vol.45, No.10, pp. 789 – 811. , October 1997.
- [7] H. Fuchs, “Improving MPEG Audio Coding by Backward Adaptive Linear Stereo Prediction,” *99st AES convention*, Preprint 4086, Oct. 1995
- [8] FAAC – Freeware Advanced Audio Coder [online]
URL: <http://www.audiocoding.com>
- [9] 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 Transactions on Consumer Electronics*, Vol.47, No.3, August 2001.

- [10] Y. S. Lin, “MPEG-1 Layer III Audio Codec Optimization and Implementation on a DSP Chip,” Master thesis submitted to department of Electrical and Control engineering, National Chiao Tung University, July 2004.
- [11] R. Gluth, “Regular FFT-Related Transform Kernels for DCT/DST-based polyphase filter banks,” in *Proc. of IEEE ICASSP’91*, Toronto, Canada, pp. 2205 – 2208, May 1991.
- [12] G. Bonnerot and M. Bellanger, “Odd-Time Odd-Frequency Discrete Fourier Transform for Symmetric Real-Valued Series,” *Proceedings of IEEE*, Vol.64, No.3 pp. 392 – 393, March 1976.
- [13] Orchard, M., “Fast bit-reversal algorithms based on index representations in GF (2b),” *IEEE Transactions on Signal Processing*, Vol. 40, No.4, pp.1004 – 1008, April 1992.
- [14] K. Sayood, *Introduction to data compression*. Morgan Kaufmann Publisher, 2000.
- [15] Test Audio Samples:
- Sandee*: Artist/Sandee Chan, Album/When We All Wept in Silence, Title/Track03, Label/Music 543.
- Lifetimes*: Artist /Sheryl Crow_Wildflower, Album /Track06
- Always On Your Side*: Artist /Sheryl Crow_Wildflower, Album /Track10
- [16] SQAM - Sound Quality Assessment Material: EBU SQAM disc tracks. [Online]. Available: <http://www.tnt.uni-hannover.de/project/mpeg/audio/sqam/>
- [17] Texas Instruments, “OMAP5912 Applications Processor” Data manual , Literature number:SPRS231E December 2003 – Revised December 2005

- [18] Texas Instruments, “TMS320C55x Technical Overview” Literature number SPRU393B, Feb. 2000
- [19] Texas Instruments, “Code Composer User’s Guide” Literature number SPRU328B, Feb. 2000
- [20] <http://www.arm.linux.org.uk/>
- [21] <http://www.vmware.com/>
- [22] <http://www.samba.org/>
- [23] <http://oskfordummies.hp.infoseek.co.jp/>
- Download:
Building Linux for the OMAP5912 OSK™ Development Platform
OSK5912 Newbie Guide Matthew Percival 27th September 2005
- [24] Linux DSP Gateway Specification Rev 2.0.1 Available: <http://dspgateway.sourceforge.net/>
- [25] Texas Instruments, TMS320C55x DSP Programmer’s Guide number SPRU376A July 2001
- [26] Texas Instruments, TMS320C55x Mnemonic Instruction Set Reference Guide , number SPRU374G July 2002
- [27] European Broadcasting Union (EBU), Sound Quality Assessment Material: Recordings for Subjective Tests, Brussels, Belgium, Apr. 1988.
- [28] ITU-R Recommendation BS.1387, “Method for objective measurements of perceived audio quality”, 2001.
- [29] EAQUAL.[Online].Available: <http://www.mp3-tech.org/programmer/sources/eaqual.tgz>

[30] Microsoft: Windows Media Player Multimedia File Format. [Online]. Available:

<http://support.microsoft.com/default.aspx?scid=kb;zh-tw;316992>



APPENDIX

The Table below defines the mailbox command used for inter-processor communication.

Command	CMD_H	CMD_L	Data register	Description
WDSND	0x10	TID	Word data to send	Word send
WDREQ	0x11	TID	--	Word request
BKSND	0x20	TID	BID	Block send
BKREQ	0x21	TID	requesting count	Block request
BKYLD	0x23	--	BID	Block yield
BKSNDP	0x24	TID	--	Block Send Private
BKREQP	0x25	TID	--	Block Request Private
TCTL	0x30	TID	ctcmd / data	Task Control
WDT	0xd0	--	notification value	Watchdog Timer Notification
TCFG	0xe0	TID	configuration data	Task Configuration
TADD	0xe2	TID	BID	Task Add
TDEL	0xe3	TID	--	Task Delete
TSTOP	0xe5	-	--	Task Stop
DSPCFG	0xf0	CFGTYP	configuration data	System configuration
REGRW	0xf2	TYPE	address / data	Register Read/Write
GETVAR	0xf4	VARID	data	Get Variable
SETVAR	0xf5	VARID	data	Get Variable
ERR	0xf8	EID	Command caused the error	Error Information