PAPER

MPEG-2/4 Low-Complexity Advanced Audio Coding Optimization and Implementation on DSP

Bing-Fei WU[†], ***Member***, **Hao-Yu HUANG**[†], ***Nonmember***, **Yen-Lin CHEN**[††a)], ***Member***, **Hsin-Yuan PENG**[†], ***and*** **Jia-Hsiung HUANG**[†], ***Nonmembers***

SUMMARY This study presents several optimization approaches for the MPEG-2/4 Audio Advanced Coding (AAC) Low Complexity (LC) encoding and decoding processes. Considering the power consumption and the peripherals required for consumer electronics, this study adopts the TI OMAP5912 platform for portable devices. An important optimization issue for implementing AAC codec on embedded and mobile devices is to reduce computational complexity and memory consumption. Due to power saving issues, most embedded and mobile systems can only provide very limited computational power and memory resources for the coding process. As a result, modifying and simplifying only one or two blocks is insufficient for optimizing the AAC encoder and enabling it to work well on embedded systems. It is therefore necessary to enhance the computational efficiency of other important modules in the encoding algorithm. This study focuses on optimizing the Temporal Noise Shaping (TNS), Mid/Side (M/S) Stereo, Modified Discrete Cosine Transform (MDCT) and Inverse Quantization (IQ) modules in the encoder and decoder. Furthermore, we also propose an efficient memory reduction approach that provides a satisfactory balance between the reduction of memory usage and the expansion of the encoded files. In the proposed design, both the AAC encoder and decoder are built with fixed-point arithmetic operations and implemented on a DSP processor combined with an ARM-core for peripheral controlling. Experimental results demonstrate that the proposed AAC codec is computationally effective, has low memory consumption, and is suitable for low-cost embedded and mobile applications.

key words: *audio coding, audio standards, MPEG-2/4 AAC, DSP, embedded systems*

1. Introduction

Due to recent advances in audio compression technology [1], [2], the convenience of multimedia communication has promoted the growth of wireless and wired network technologies. At the same time, the increasing prevalence of portable devices such as personal digital assistants (PDAs), digital audio player and smart mobile phones has also advanced audio compression technology. In the last decade, MPEG/Audio Layer3 (MP3) [3] which supports an 11:1 compression rate at a 128 Kbps bit rate with CD audio quality, has played a crucial role in the domain of audio compression. Moreover, MPEG-2/4 Advanced Audio Coding (AAC) provides CD audio quality at a 96 Kbps bit rate and has become the audio compression standard in the recent years. With the limited Internet bandwidth and portable device storage, AAC now leads audio compression technology.

Since the AAC standard first appeared, many studies have been published on optimizing the AAC algorithm. Some studies [4]–[7] developed an optimized AAC algorithm on PC-based platforms. Profiling the AAC encoding flow indicates that the Psycho-Acoustic Model (PAM) and the quantization loop module are the most complex modules in the AAC encoder [9]. Thus, many recent studies have attempted to accelerate the computational speed of the PAM and the quantization module while other studies focus on improving their efficiency [8]–[13]. Dimkoviae et al. [8] introduced an Adaptive Time-Frequency Transformation (ATFT) for coding audio signals, and developed a PAM adapted to time-frequency functions. Huang et al. [9] used a new design of Modified Discrete Cosine Transform (MDCT)-based PAM to replace the original Fast Fourier Transform (FFT)-based PAM and implemented the encoder on a 16-bit fixed-point processor. Hu et al. [10] presented an optimized AAC encoder that improved the MDCT-based PAM with a block switching scheme in the time domain.

While these publications focus on optimizing the PAM, many other papers attempt to optimize the bit-allocation mechanic in the quantization loop. Kurniawati et al. [11] proposed a new implementation technique for the PAM and the bit allocation module, which is the dominant user of computational resources in the AAC encoder. Yang et al. [12] presented a new bit allocation algorithm to improve the efficiency of MPEG-4 AAC when the inter-band dependency of coding process exists in the Breiman, Friedman, Olshen, and Stone (BFOS) algorithm. Alexandre et al. [13] also worked on optimizing the bit-allocation algorithm in the PAM with a nonuniform quantization block in the AAC encoder. Most current optimization methods concentrate on one module or block in the whole encoding flow and are implemented on a PC-based platform.

Recently, numerous multimedia use-cases on portable and mobile applications have appeared in our daily life. There are also many real-time application examples of applying the AAC audio encoder, such as mobile audio/video recorders and car surveillance systems. Due to the popularity of embedded and mobile devices, recent studies have begun to focus on optimization methodologies of the AAC codec implemented on embedded and mobile multimedia platforms [31]–[38]. Reducing computational complexity

Manuscript received August 27, 2009.

Manuscript revised December 8, 2009.

[†]The authors are with Electrical and Control Engineering, National Chiao Tung University, Hsinchu, 30050 Taiwan.

[††]The author is with Computer Science and Information Engineering, National Taipei University of Technology, Taipei 10608, Taiwan.

a) E-mail: ylchen@csie.ntut.edu.tw

DOI: 10.1587/transinf.E93.D.1225

and memory requirements are the most important optimization issues for implementing the AAC codec on embedded devices. Due to power saving issues, most embedded and mobile systems can only provide very limited computational power and memory resources for the coding process. Therefore, modifying and simplifying only one or two blocks are insufficient for optimizing the AAC encoder and enable it to work well on embedded systems. It is also necessary to enhance the computational efficiency of other important modules in the encoding algorithm, including the temporal noise shaping (TNS), mid/side (M/S) stereo, and MDCT modules. By combining these optimized blocks and modules with the proposed PAM and simplified bit-allocation approaches, this study proposes an efficient low-complexity and low-memory-cost optimization scheme for the AAC encoder on embedded platforms. This study also presents an optimized AAC decoding framework for embedded applications. Based on computational costs and performance, this study selects the TI OMAP5912 DSP-ARM dual-core platform to implement the proposed AAC codec.

The optimization of the AAC decoder is another important issue. Since AAC decompression costs less computational complexity and power consumption than encoding, it is easier to port the decoder on the portable devices than the encoder. Many studies [14]–[17] have presented optimized decoding algorithms for the DSP-based or RISC architecture. Servetti et al. [14] improved the decoding framework for the Intel Pentium series platform. Bang et al. [15] and Waston et al. [16] proposed a fast implementation of the decoding architecture for embedded applications. Yoon et al. [17] proposed a specialized DSP architecture for MPEG-2/4 AAC high-quality audio. To achieve a complete AAC codec on an embedded system, this study implements a decoder with lower computational complexity than existing AAC decoders on the OMAP5912 platform. This study also optimizes the inverse MDCT, which is the most computational complex module in the decoding flow [18]. Due to the computational limitations of the embedded system, we present a fast solution to the nonlinear computation in the inverse quantization module. As in the encoder, the memory size must be controlled and reduced in the decoder. Therefore, this study also presents a programming methodology to shrink the Huffman tables, which has the most critical memory requirements in the overall algorithm.

For portable and outdoor usage [19], the proposed AAC codec can provide higher processing speeds while maintaining audio quality. In the encoder, we optimize the whole encoding flow in the following aspects: 1). The MDCT block is simplified by mathematical techniques to reduce the computational cost and reduce memory usage. 2). This study also presents a simplified TNS and M/S stereo module based on an optimized statistical model. By performing a statistical analysis, we developed an effective early decision approach to determine the most effective orders of Levinson-Durbin Recursion (LDR) before performing the TNS computation. Moreover, the M/S decision mechanism in the M/S stereo coding module is another critical and complex computational factor. The proposed model adopts an optimized M/S coding decision scheme that analyzes the energy of a single frame instead of the scalefactor bands in the entire frame as adopted by the original AAC standard. 3). Memory consumption is also a critical issue in embedded applications, and the Huffman tables occupy a considerable size of the memory usage in the AAC codec. This paper proposes an efficient memory reduction approach which provides a satisfactory balance between reducing memory usage and expanding the encoded files. 4). Additionally, in the decoder, the most complex modules, that is the IMDCT module, is optimized for better computational efficiency due to the limited computation resources of embedded platforms. Both the proposed AAC encoding and decoding algorithms are implemented on the OMAP5912 platform with its DSP and ARM cores. This study focuses on the optimization issues of these blocks, which have not yet been examined in detail, and implements these optimized approaches in the AAC codec. In so doing, this study develops a fast and efficient AAC codec that can be applied to portable and embedded multimedia applications with well-recognized quality for low cost, flexible, portable and outdoor usages.

This paper is organized as follows. Section 2 focuses on optimizing the AAC codec using software-based and algorithmic optimization techniques. Using predictive approaches, this section simplifies and optimizes the computational critical modules of the encoding and decoding processes and proposes techniques for memory size reduction as well. Section 3 presents effective platform-based optimizations for implementing the proposed AAC codec on the OMAP platform, and introduces some DSP-based optimizations. Section 4 presents experimental results. Finally, Sect. 5 offers the conclusion to this paper.

2. MPEG-2/4 AAC Software-Based and Algorithmic Optimizations

This section presents the proposed approaches for optimizing the AAC encoder and decoder, including the fast Modified Discrete Cosine Transform (MDCT) and Inverse Modified Discrete Cosine Transform (IMDCT), simplified Temporal Noise Shaping (TNS) tool, simplified mid/side (M/S) stereo coding module, and inverse quantization block. Due to the memory consumption issues of resource-restricted embedded systems, this section also presents some memory reduction methods based on our optimized statistical model for the Huffman tables and the Huffman coding process.

2.1 Simplified TNS Tool

This section presents a simplified TNS module. The original TNS calculation applies a filtering process to some parts of spectral coefficients when the prediction gain of spectral data is greater than a pre-defined threshold, which is obtained by calculating the Signal-to-Noise Ratio (SNR) during Levinson-Durbin Recursion (LDR). Table 1 shows the statistical analysis results for the inactive percentage of fil-

Table 1 The inactive percentage of TNS filtering.

Test samples	*Signal characteristic*	*Inactive percentage for long windows*
AlwaysOnYourSide	Pop song	94.25%
AFondFarewell	Pop song	90.53%
Lifetime	Pop song	94.55 %
RedMorning	Pop song	91.13%
Sandee	Pop song	89.82%
Sopr44_1	Soprano	93.3 %
quar48_1	Quartet	80.1%
horn23_2	Horn	95.35%
trpt21_2	Trumpet	87.01%
gspi35_2	Glockenspiel	83.17%
Average	N/A	89.92%

tering process under the condition of the only-long-window blocks in the overall encoding process. Because the existing TNS tool is able to compensate for the negative effect of the pre-echo, and based on related research [20] showing that the encoding performance without block switching does not have a significant effect on the loss of the quality, the proposed Psycho-Acoustic Model (PAM) does not include a block-switching scheme, thus resulting in only-long-window blocks.

Figure 1 shows the simplified procedure, in which the TNS module is enabled only when the predicted gain of audio spectral data is greater than a pre-defined threshold. $G_{SNR,n}$ and $T_{LDR,n}$ stand for the prediction gain of nth-order auto-correlation and LDR, respectively. The complexity of LDR is $O(N^2)$ [21]. For this reason, the early decision approach splits the recursion order of the LDR into two blocks and performs the 6th-order auto-correlation and LDR. If $G_{SNR,6}$ is greater than $T_{LDR,6}$, the following recursion order from 7th to 12th will be performed.

According to the simplified TNS flow in Fig. 1, four possible cases, which describe whether or not 6th-order and 12th-order Linear Predictive Coding (LPC) are enabled in the simplified TNS flow are listed in Table 2.

Table 3 shows the active possibilities of the four cases presented in Table 2. In Table 3, the average result of the five test samples in case (1) shows that 87.94 % of the LPC predictions for TNS filtering results can be simply determined through a 6th-order LPC, while the average result of the five test samples using case (4) demonstrates that only about 1.56 % of the LPC predictions mis-predicted using 6th-order LPC. Therefore, this proposed method can provide very similar quality as compared to that of the original TNS mechanism because the percentage of mis-prediction using 6th-order LPC is only 1.56 %. Furthermore, since the average result for the inactive percentage for long windows in Table 1 is 89.92 %, the complexity of TNS can be reduced by 89.92 % · 87.94 % = 79.08 %.

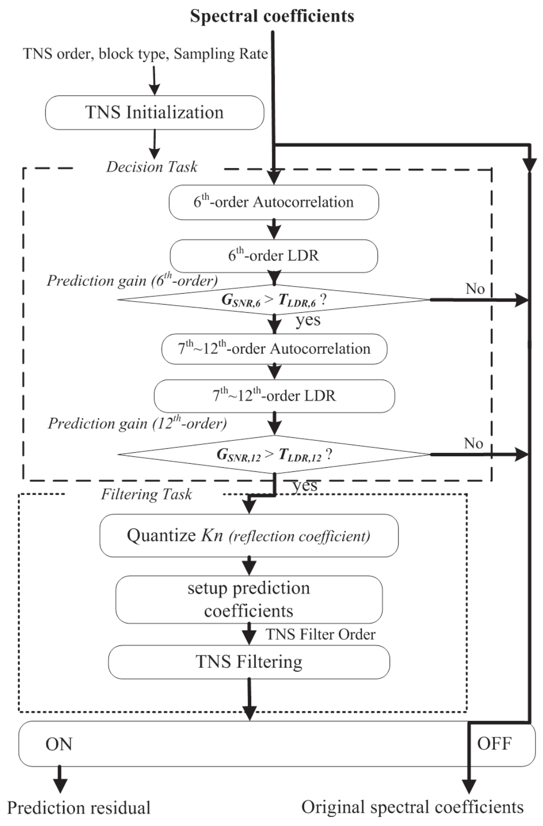

Spectral coefficients
TNS order, block type, Sampling Rate
TNS Initialization
Decision Task
6th-order Autocorrelation
6th-order LDR
Prediction gain (6th-order)
$G_{SNR,6} > T_{LDR,6}$?
No
yes
7th~12th-order Autocorrelation
7th~12th-order LDR
Prediction gain (12th-order)
$G_{SNR,12} > T_{LDR,12}$?
No
yes
Filtering Task
Quantize Kn (reflection coefficient)
setup prediction coefficients
TNS Filter Order
TNS Filtering
ON
OFF
Prediction residual
Original spectral coefficients

Fig. 1 Simplified TNS flow.

Table 2 Four possible cases in the simplified TNS.

Case (1)	12th-order LPC prediction is **disabled**. 6th-order LPC prediction is **disabled**.
Case (2)	12th-order LPC prediction is **disabled**. 6th-order LPC prediction is **enabled**.
Case (3)	12th-order LPC prediction is **enabled**. 6th-order LPC prediction is **enabled**.
Case (4)	12th-order LPC prediction is **enabled**. 6th-order LPC prediction is **disabled**.

Table 3 Analysis of the active possibilities of the four cases.

Test Samples	*Case(1)*	*Case(2)*	*Case(3)*	*Case(4)*
AlwaysOnYourSide	91.69%	2.52%	3.11%	2.68%
AFondFarewell	91.11%	2.66%	5.88%	0.35%
Lifetime	91.36%	3.17%	5.19%	0.28%
RedMorning	91.73%	2.09%	6.03%	0.15%
Sandee	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%
horn23_2	89.72%	5.04%	4.15%	1.09%
trpt21_2	85.82%	4.08%	6.98%	3.12%
gspi35_2	87.57%	1.79%	8.12%	2.52%
Average	87.94%	3.74%	6.76%	1.56%

2.2 Mid/Side Stereo Coding Optimization

The stereo coding process adopts channel correlations to determine whether or not to apply the Mid/Side (M/S) stereo coding. The coding algorithm itself is quite simple, but the process to decide whether or not this scalefactor band switches to M/S mode requires much more effort. In the method proposed by the original AAC standard, the decision process for the M/S stereo coding is thoroughly performed on each of the scalefactor bands. As a result, the number of manipulations is too high to implement the decision process of the M/S stereo coding in the original AAC standard on an embedded processor with limited computation power. Table 4 shows that at least 80 % of the scalefactor bands of a single frame will be switched to the M/S coding mode. The original decision method consumes a lot of computation time. Thus, for the simplicity, we adopt an optimized decision method that uses the energy of a single frame instead of the scalefactor bands in the entire frame.

Figures 2 and 3 illustrate the channel pair energy ratios of the scalefactor bands switched to the M/S mode or remaining in the Left/Right (L/R) mode, respectively. Figure 2 shows that the energy ratios of the channel pair of the scalefactor band switched to the M/S mode are often around 1∼2, whereas Fig. 3 shows that the energy ratios of the scalefactor band remaining in the L/R mode are usually greater than those of the M/S mode. Analysis results show that the M/S mode decision can be performed based on the comparative features of the energy ratios. Therefore, this study proposes an efficient simplified determination approach that analyzes the frame energy of the stereo channel pair. This approach only considers the average energy in each frame of the channel pairs to determine the usage of the M/S stereo coding, and thus the computation cost is significantly reduced. Additionally, since a frame with lower energy is less benefited from M/S stereo coding, such frames, which usually occur in the beginning, the end, and the relatively silent part in the middle of a song, will be determined not to perform the M/S stereo coding and thus the computational time can be significantly saved.

To apply the correlation and the mean energy of two stereo channels and determine the usage of the M/S stereo coding, it is first necessary to define two thresholds. The first is the threshold of the average frame energy, T_{AVG}, and the second is the threshold of the ratio of the frame energy, T_{RATIO}. Figure 4 depicts the flow chart of simplified determination approach of M/S coding, where e_L, e_R and e_{RATIO} denote the average energy of the right channel, left channel and the energy ratio, respectively, which is defined as (1), and $AVG(e_L, e_R)$ represents the average of e_L and e_R. This optimized M/S mode decision method can achieve a significant 73.60 % reduction of computational costs compared to the original M/S decision module.

$$e_{RATIO} = \begin{cases} e_L/e_R & ,if \quad e_L \geq e_R \\ e_R/e_L & ,if \quad e_R > e_L \end{cases} \tag{1}$$

Table 4 The percentage of encoder switching to M/S mode.

Test Samples	***Percentage***
AlwaysOnYourSide	89.26%
AFondFarewell	88.16%
Lifetime	87.35%
RedMorning	95.03%
Sandee	88.34%
Sopr44_1	89.46%
quar48_1	88.84%
horn23_2	83.18%
trpt21_2	80.66%
gspi35_2	85.13%
Average	87.54%

2.3 Modified Fixed-Point Inverse Quantization

The inverse quantization defined in the AAC decoder standard can be computed using Eq. (2), where x_q denotes the quantized spectral coefficients, and x'_q represents the inverse quantized results of x_q. This equation shows that the computation of $\left|x_q\right|^{\frac{4}{3}}$ costs the most computational time.

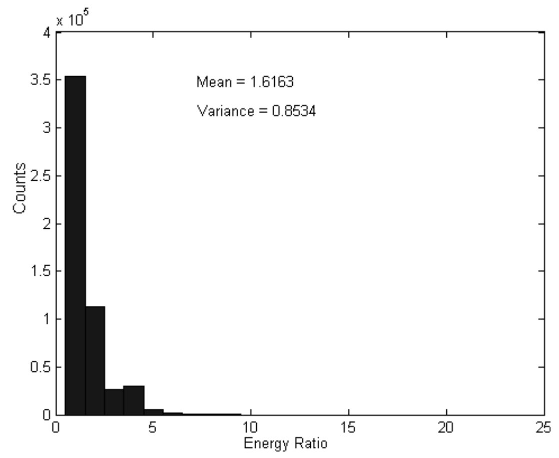


Fig. 2 Energy ratio of the scalefactor bands switched to M/S mode.

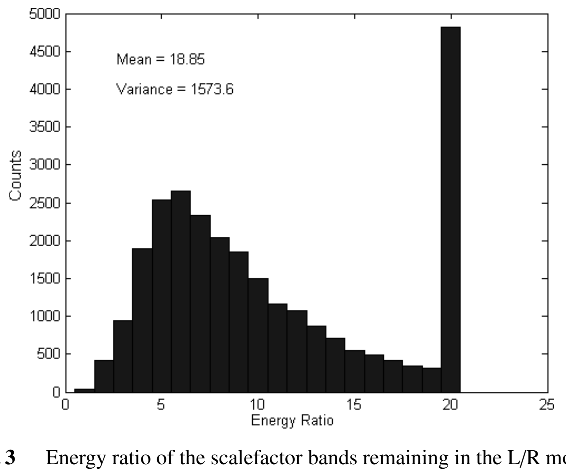


Fig. 3 Energy ratio of the scalefactor bands remaining in the L/R mode.

$$x'_q = Sign\left(x_q\right) \cdot \left|x_q\right|^{\frac{4}{3}} = Sign\left(x_q\right) \cdot \left|x_q\right| \cdot \left|x_q\right|^{\frac{1}{3}} \quad (2)$$

To apply the inverse quantization in the original AAC decoder standard on a fixed-point DSP platform, it is necessary to optimize the inverse quantization algorithm using a fixed-point scheme. From [22], we can find that the inputs $\left|x_q\right|$, from 0 to 8191, can be segmented into 3 sections. The first section, $0 \leq x_q < 32$, utilizes a small Lookup Table (LUT) to directly obtain the noiseless values. The following two sections, that is, $32 \leq x_q \leq 255$ and $256 \leq x_q \leq 8191$, adopt a piecewise linear approximation method to compute the values. Thus, the computation of $\left|x_q\right|^{\frac{1}{3}}$ in the three sections above can be obtained using respective schemes given in Eq. (3).

$$\begin{cases} LUT\left(x_q^{\frac{1}{3}}\right),\ 0 \leq x_q < 32 \\ \alpha_2\left(S_2\left(x_q\right)\right) \cdot x_q + \beta_2\left(S_2\left(x_q\right)\right) \\ ,\ 32 \leq x_q < 256 \\ \alpha_3\left(S_3\left(x_q\right)\right) \cdot x_q + \beta_3\left(S_3\left(x_q\right)\right) \\ ,\ 256 \leq x_q \leq 8191 \end{cases} \quad (3)$$

LUT(.) means that the LUT scheme is applied in the 1st section, α_2 and β_2 are the linear approximation coefficients for the 2nd section, α_3 and β_3 are the linear approximation coefficients for the 3rd region, $S_2(.)$ is the segment index derived from (4), and $S_3(.)$ is the segment index from (5). This optimal quantization method can effectively reduce computational costs, especially for fixed-point processors.

$$\begin{aligned} & B_2(j) = n\mathrm{int}\left(\left((32)^3 \cdot 2^j\right)^{\frac{1}{3}}\right),\ j = 0 \sim 9 \\ & \Rightarrow B_2(j) \in \{32, 41, 51, 64, 81, 102, 128, 162, 204, 256\} \quad (4) \\ & S_2\left(y_{f,g}\right) = \left\{j \middle| B_2(j) \leq y_{f,g} < B_2(j+1)\right\} \\ & B_3(j) = \begin{cases} n\mathrm{int}\left(256 \cdot 2^j\right), & j = 0 \sim 4, \\ 8192, & j = 5 \end{cases} \\ & \Rightarrow B_3(j) \in \{256, 512, 1024, 2048, 4096, 8192\} \end{aligned}$$

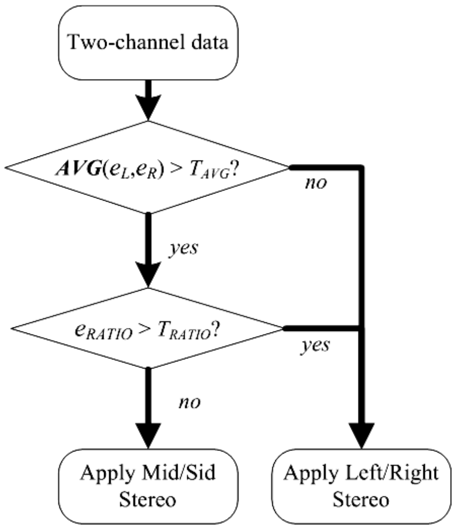


Fig. 4 Simplified flow chart of M/S stereo coding.

$$S_3\left(y_{f,g}\right) = \left\{j \middle| B_3(j) \leq y_{f,g} < B_3(j+1)\right\} \quad (5)$$

Table 5 illustrates analytical results of the inputs of the inverse quantization, and demonstrates that most of the inputs are located in the first section which adopts the LUT scheme. Thus, the proposed simplified inverse quantization method works nearly same as the original non-linear computation of the AAC standard.

Table 6 shows the results of the complexity reduction efficiency of the proposed method as compared to the ones of the original method. By applying the proposed inverse quantization method, the average computational complexity is reduced by 41.51 % as compared to the original one. Therefore, the proposed simplified method cost about only 124 bytes space but can achieve about 40 % reduction of the computational complexity.

2.4 Memory Reduction

Because memory reduction is also an important issue in embedded application, this study presents three schemes to address this problem, as follows:

- A Buffer re-usage scheme, which considers the life span of data memory
- A Huffman Tables reduction scheme based on the statistical analysis
- The use of various sizes of data types for storing the

Table 5 The percentage of the inputs located in the first section.

Test Samples	*Percentage*
AlwaysOnYourSide	100%
AFondFarewell	100%
Lifetime	100%
RedMorning	99.99%
Sandee	99.99%
Sopr44_1	100%
quar48_1	100%
horn23_2	99.99%
trpt21_2	100%
gspi35_2	99.99%
Average	99.99%

Table 6 Results of the complexity reduction efficiency of the proposed method as compared to the ones of the original method.

Test Samples	*Percentage*
AlwaysOnYourSide	41.82%
AFondFarewell	41.23%
Lifetime	41.38%
RedMorning	40.85%
Sandee	40.60%
Sopr44_1	40.59%
quar48_1	41.87%
horn23_2	41.92%
trpt21_2	43.69%
gspi35_2	41.14%
Average	41.51%

length and the codeword information of Huffman Tables

To efficiently implement the AAC codec on an embedded platform with limited resources, the declared memory blocks are first thoroughly analyzed and examined as they are used. In general, the time duration for data processing is called the "life cycle" or "life span." For a DSP-based platform, dynamic memory allocation must be removable. Therefore, this subsection presents an efficient framework for static memory management in a DSP-based platform. Figure 5 illustrates the concept of buffer re-usage for various life spans, where buf_n denotes the given buffers; s_n and e_n denote the start and the end usage of buf_n, respectively; and S_{Tn} represents the status of buf_n at duration T_n. Using this framework, a set of the buffers which exist in different life spans can share and be declared in the same memory space. Once a given buffer buf_{n1} ends its usage in the corresponding life time T_n, its occupied space can be immediately used by the other buffer buf_{n2}.

The 12 Huffman tables used by the AAC encoder consume a large memory space. The 1st – 11th codebooks are used for encoding the quantized spectral values, while the 12th codebook is specified for coding the scalefactors. Table 7 shows the maximum absolute value of the quantized coefficients that can be coded according to each Huffman codebook and the number of coefficients in each n-tuple associated with each codebook. Generally speaking, two tables will be provided as candidate choices for quantized spectral coefficients. The encoder will compute the required bit cost of each table, and the table with minimum required bits will be chosen.

Therefore, regardless of the codebook selected, audio quality is not affected within the same range of the maximum absolute value. However, the total statistical results of test samples, which are listed in Table 1, in Fig. 6 indicated that it is unnecessary to provide whole tables as candidate choices. For instance, the coefficients with the maximum absolute value ranging from 5∼12 only need the 8th and 10th codebooks, and thus the 7th and 9th codebooks can be removed. By removing the 7th and 9th codebooks and separating the elements in Huffman tables, the Huffman tables can be reduced from 4,278 Bytes to 3,579 Bytes. Though this scheme optimizes memory consumption, it may expand the encoded AAC bitstream. To achieve the best tradeoff, Table 8 illustrates the results of the memory reduction and the size increase of the encoded files when economizing different sets of codebooks. The tradeoff analysis presented in Table 8 suggests that the 7th and 9th codebooks should

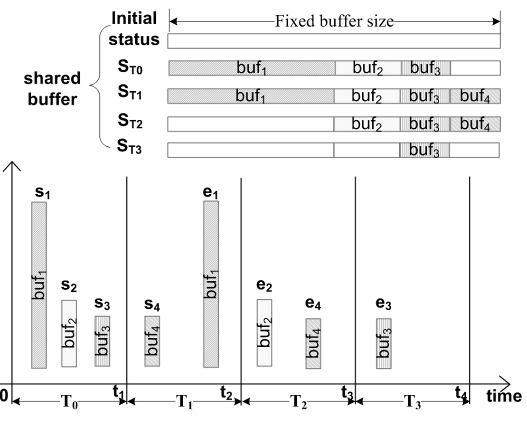


Fig. 5 Illustration of buffer re-usage.

Table 7 Huffman codebooks.

Codebook Index	*n-Tuple Size*	*Maximum Absolute Value*	*Signed*
0		0	
1	4	1	Yes
2	4	1	Yes
3	4	2	No
4	4	2	No
5	2	4	Yes
6	2	4	Yes
7	2	7	No
8	2	7	No
9	2	12	No
10	2	12	No
11	2	16 (ESC)	No

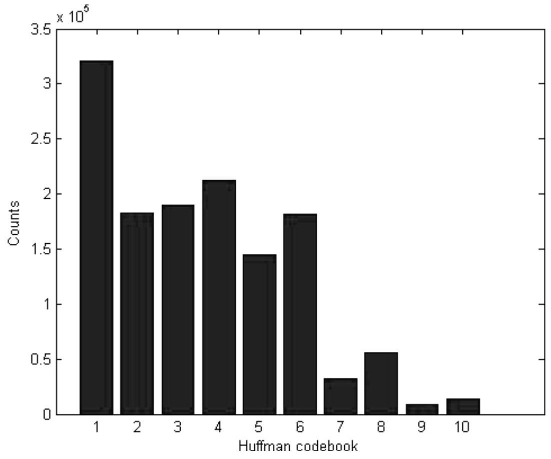


Fig. 6 The counts of the Huffman codebooks used by the tested audio samples.

Table 8 Tradeoff between reducing the codebooks and inflating the encoded files.

Codebooks being removed	*The average percentage of memory reduction in Huffman tables*	*The average percentage of inflation in encoded AAC files*
Remove half of the candidate codebooks	33.18%	14.55%
Remove codebooks 7 and 9	16.34%	1.10%
Remove codebook 9	11.85%	0.08%

be removed for the best tradeoff between memory reduction and coded file inflation.

2.5 Fast MDCT and IMDCT

This section introduces the optimizations of MDCT/IMDCT transforms, which are the core architecture of the filter bank module in the AAC codec. Previous studies [23], [24] provided faster approaches called the Odd-time Odd-frequency Discrete Fourier Transform (O2DFT) for odd transforms such as MDCT/IMDCT. The MDCT/IMDCT can be computed using only one N/4-point FFT/IFFT and some pre- and post-rotation operations of the sample points. According to the O2DFT computation process, the FFT/IFFT is the most computationally demanding task in the filter bank. This study will adopt the O2DFT in the filter bank module and also propose the optimization of the FFT/IFFT for embedded applications.

The modified FFT is performed by adopting the symmetric characteristics of FFT and IFFT. Figure 7 shows that the implementation of FFT involves three sub parts. Part A will compute the twiddle factors required to perform the butterfly manipulation and the general FFT as depicted in (6). Equation (6) also shows that the twiddle factors in the FFT can be partitioned into cosine and negative sine computations. In (6), x_n and X_k represent the nth input and the kth output of the FFT, respectively. In this application, the cosine and negative sine coefficients are calculated in advance and the look-up tables of these coefficients are obtained by (7), where costbl[i] and negsinbl[i] represent the cosine and negative sine tables, respectively. Part B performs the in-place decimation of input samples, and Part C is the butterfly construction.

$$X_k = \sum_{n=0}^{N-1} x_n e^{-2\pi k/N} = \sum_{n=0}^{N-1} x_n \left(\cos 2\pi k/N - j \sin 2\pi k/N\right) \quad (6)$$

$$\begin{cases} \text{costbl[i]} = \cos(2\pi \cdot i/N); & i = 0\sim255, \text{ for } N=512 \\ \text{negsintbl[i]} = -\sin(2\pi \cdot i/N); & i = 0\sim63, \text{ for } N=64 \end{cases} \quad (7)$$

To reduce the computational complexity of FFT, this study presents a look-up table based approach to replace the computation of part A. This design achieves a 30 % reduction in computational costs. To reduce the memory requirements of the twiddle factors, the symmetric and anti-symmetric relationship of the cosine and sine computations are applied with only a little addressing overhead in part C. Equations (8) and (9) describe the long and short window case, respectively, and (10) expresses the relationship between them. To more clearly describe the computation process of the proposed approach, Fig. 8 illustrates the long window case. For both cosine and negative sine tables, Equation (10) shows that the values are symmetric with $i = 128$. Moreover, their values are equal in certain indices. Equations (11), (12), (13), and (14) show the corresponding values, which are drawn with red, green, blue and violet lines in Fig. 8, respectively.

$$\begin{cases} \cos(2\pi \cdot i/N) = -\cos(2\pi \cdot (256 - i)/N), \\ \quad i = 1 \sim 127, N = 512 \\ \sin(2\pi \cdot i/N) = \sin(2\pi \cdot (256 - i)/N), \\ \quad i = 1 \sim 127, N = 512 \end{cases} \quad (8)$$

$$\begin{cases} \cos(2\pi \cdot i/N) = -\cos(2\pi \cdot (32 - i)/N), \\ \quad i = 1 \sim 15, N = 64 \\ \sin(2\pi \cdot i/N) = \sin(2\pi \cdot (32 - i)/N), \\ \quad i = 1 \sim 15, N = 64 \end{cases} \quad (9)$$

$$\begin{cases} \sin(2\pi \cdot (128 - i)/N) = \cos(2\pi \cdot i/N) \cdot (-1), \\ \quad i = 1 \sim 127, N = 64 \\ \sin(2\pi \cdot (16 - i)/N) = \cos(2\pi \cdot i/N) \cdot (-1), \\ \quad i = 1 \sim 16, N = 64 \end{cases} \quad (10)$$

$$(-1) \cdot \sin(2\pi \cdot 1/N) = \cos(2\pi \cdot 127/N) \quad (11)$$

$$(-1) \cdot \sin(2\pi \cdot 125/N) = \cos(2\pi \cdot 3/N) \quad (12)$$

$$(-1) \cdot \sin(2\pi \cdot 126/N) = \cos(2\pi \cdot 2/N) \quad (13)$$

$$(-1) \cdot \sin(2\pi \cdot 127/N) = \cos(2\pi \cdot 1/N) \quad (14)$$

Therefore, the proposed optimization effectively reduces the required memory usage from (2 × 256 (index) + 2 × 32 (index)) × 4 (bytes for short) = 2,304 (bytes) to (129 (index) + 17 (index)) × 4 (bytes for short) = 584 (bytes).

The purpose of Part B in Fig. 7 is to perform the in-place decimation of input samples. However, pre-storing the bit-reverse index requires a lot of memory. This study

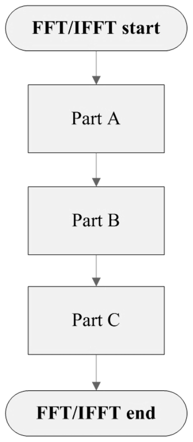


Fig. 7 The sub parts in the implementation of the FFT/IFFT.

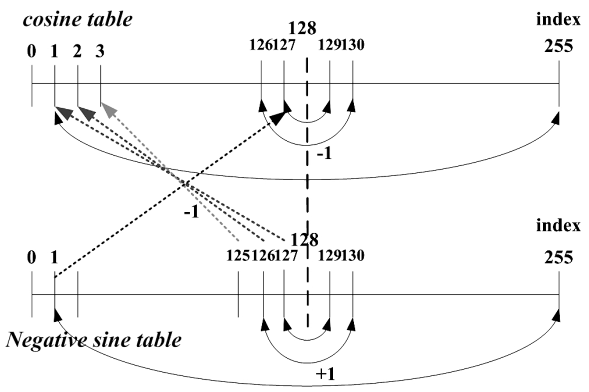


Fig. 8 The symmetric and anti-symmetric properties.

applies a fast bit-reversal permutation algorithm proposed in [25] to optimize a bit-reversal. As a result, the program loop can be reduced from $N \cdot q(N = 2^q)$ to N, and the storage space for bit-reverse indexing can also be omitted.

3. Experimental Results

By applying the proposed techniques, the AAC codec were implemented on the OMAP platform, which includes a dual core processor, ARM9 and C55x DSP. This section conducts a performance evaluation of the computational complexity and audio quality of the proposed AAC codec techniques, demonstrating the efficiency and feasibility of the proposed techniques.

3.1 Comparative Results of Encoding and Decoding Timings

The experimental results in this subsection are simulated on a PC platform with a Pentium IV 2.8 GHz platform and 768 MB RAM. One of the audio files is a pop-song performed by a female singer, the others, including artificial signals, single instruments, vocal, and speech sounds, are obtained from Sound Quality Assessment Materials (SQAM) [26]. Table 9 lists the length and characteristics of these test audio files. The proposed AAC with Low Complexity (LC) Profile encoder is simulated at a bitrate of 96 kb/s and compared with an open source MPEG-2/4 AAC encoder called FAAC [27], which is the only open source AAC encoder. Additionally, FAAC has the feature of portable and reasonably fast while providing the equivalent quality as the AAC standard. Table 10 compares the FAAC and the proposed AAC codec in terms of computational timings for critical modules, including the filterbank, Temporal Noise Shaping (TNS), Mid/Side (M/S) stereo coding and bitstream encoding modules. Because the proposed design omits the block-switching scheme from the Psycho-Acoustic Model (PAM), Table 10 does not include the computational timing results for the PAM. Results show that the proposed software-based and algorithmic optimization schemes and the fixed-point modification perform 3.5 times better than those of the FAAC. This means that the proposed one works on the same quality as AAC standard with superior coding speed.

Furthermore, this study implemented the proposed AAC decoder with Odd-time Odd-frequency Discrete Fourier Transform based (O2DFT-based) Inverse Modified Discrete Cosine Transform (IMDCT) and optimized inverse quantization modules. As above, this section compares the decoding performance of the proposed AAC decoder with that of the FAAD2 [27], which can provide faster and better quality than the ISO reference coder [28]. In fact, the FAAD2 with floating point manipulation is fast enough for PC-based platforms. However, it is still not suitable for applications on low-power handheld and portable embedded systems, such as DSP-based platforms. Although the FAAD2 somewhat supports fixed-point decoding, its computational costs are still too high for embedded applications. Table 11 compares the computational timing results of the heavy computational modules, including the inverse filterbank and dequantization modules, for the fixed-point FAAD2 and the proposed AAC decoder. The test audio samples are 96 kbps AAC files encoded by FAAC of LC Profile. Table 11 shows that the decoding speed of the proposed AAC decomposer is more than twice that of the FAAD2.

Table 9 Tested audio samples with their song length and characteristics.

Audio samples	*Characteristics*	*Length*
frer07_01	Electronic tune	0:34
gspi35_2	Glockenspiel	0:19
horn23_2	Horn	0:25
quar48_1	Quartet	0:23
spfe49_1	Female speech in English	0:19
trpt21_2	Trumpet	0:17
sandee	Female voice pop-song	3:42

Table 10 Comparative results on simulated encoding timings with critical modules.

Audio samples	*Method*	*Filter bank (ms)*	*TNS (ms)*	*M/S (ms)*	*Bitstream (ms)*	*Total time (seconds)*	*Gain*
frer07_1	FAAC	1073	329	862	356	10.27	4.01
	Proposed	903	124	16	220	2.56	
gspi35_2	FAAC	559	204	500	410	5.92	3.23
	Proposed	594	32	15	156	1.83	
horn23_2	FAAC	763	218	620	316	8.59	3.33
	Proposed	607	77	16	265	2.58	
quar48_1	FAAC	649	233	561	248	7.26	3.08
	Proposed	749	139	31	235	2.36	
spfe49_1	FAAC	390	159	391	342	5.36	2.76
	Proposed	468	125	16	126	1.94	
trpt21_2	FAAC	422	216	424	264	5.55	3.34
	Proposed	419	96	15	63	1.66	
sandee	FAAC	5473	1544	4038	3080	61.3	2.39
	Proposed	7085	1414	79	2027	25.7	

Table 11 Comparison of simulated decoding times with critical modules.

Audio samples	*Method*	*Inverse filterbank (ms)*	*Inverse quantization (ms)*	*Total time (seconds)*	*Gain*
frer07_1	FAAD2	4999	202	8.39	2.67
	Proposed	1610	156	3.14	
gspi35_2	FAAD2	2612	127	3.94	2.45
	Proposed	936	91	1.61	
horn23_2	FAAD2	3690	171	5.59	2.65
	Proposed	953	78	2.11	
quar48_1	FAAD2	3385	189	5.38	2.06
	Proposed	1158	92	2.61	
spfe49_1	FAAD2	2366	169	3.64	2.07
	Proposed	813	110	1.76	
trpt21_2	FAAD2	2211	126	3.42	2.09
	Proposed	904	79	1.64	
sandee	FAAD2	24406	1550	40.16	2.02
	Proposed	10000	1466	19.92	

3.2 Encoding Quality Evaluation

To evaluate the audio quality of the proposed coder, this study adopts ITU-R Recommendation BS 1378 [29] as the Objective Difference Grade (ODG) measurement. The reference codecs are the floating-point FAAC and FAAD2 [27], respectively. The test audio samples listed in Table 1, Table 4 and Table 9 are encoded by FAAC and the proposed encoder, and then both the encoded files are decoded by FAAD2. Two reconstructed audio sequences are compared with the source sequences to obtain the ODG grades using Evaluation of Audio Quality (EAQUAL). The ODG grades range from [−4; 0], where −4 is "very annoying," and 0 reflects "imperceptible" differences between the source and reconstructed signals. Table 12 shows the ITU-R 5-grade impairment scale.

To perform a comprehensive comparison, this study uses a set of different bitrate settings ranging from 32 K bps to 320 K bps, and divides the audio samples into two groups, namely the high-level group and low-level group. The high-level group contains half of the test items with higher signal levels, whereas the low-level group contains the remaining samples [30]. This subsection conducts two experiments to assess the encoding quality of the proposed encoder.

The first experiment compares the encoding quality of the FAAC with that of our proposed encoder. Figure 9 presents the results, where each curve indicates the average ODG grades obtained by the two encoders under different bitrates, and Hi and Lo denote the ODG results of the high-level and low-level group, respectively. These results show that the proposed encoder achieves similar quality in the results of low-level group compared with the results encoded by the FAAC with floating point computation. In the high-level group, the proposed encoder also produces acceptable quality for human hearing. For example, the average ODG rating of the proposed encoder in the high-level group at 160 K bitrate is around −0.7, which means that on average the encoding quality of the proposed encoder at 160 k bitrate is better than the quality of "Perceptible but not annoying," as Table 12 indicates. The quality grades of the proposed encoder in high-level group are somewhat lower than those of the original FAAC. This is because the FAAC is performed using the floating-point architecture, while the proposed encoder is implemented on a fixed-point architecture to increase computational effectiveness on the OMAP platform.

To perform a fair comparison, we implement a fixed-point FAAC, where the filterbank and quantization modules are modified to adapt for the fixed-point architecture. Accordingly, the second comparative experiment compares the encoding quality of this fixed-point FAAC with that of our proposed encoder. Figure 10 illustrates the results, which show that the proposed encoder can achieve similar quality grades compared to those obtained by the fixed-point FAAC in both high-level and low-level groups. These results demonstrate that the encoding quality of the proposed encoder is comparable with that of the FAAC in the fixed-point architecture. Moreover, the results in Sect. 3.1 showed that the proposed encoder can provide significantly better computational efficiency than that of the FAAC, and is highly effective in fixed-point embedded DSP platforms.

3.3 Performance for DSP

It is very important for technical feasibility and market acceptance that the AAC coding algorithm can be implemented on embedded systems such as DSP-based platforms. DSPs have very low power consumption, which enables small, handheld, and even battery-driven devices without cooling fans. On the other hand, developers of multimedia applications will experience several unexpected difficulties when using a DSP as the target platform instead of a desktop personal computer. These difficulties arise from the architectural limitations of DSPs, including limited mem-

Table 12 The ITU-R 5-grade impairment scale.

ODG rating	*Meaning*
0	Imperceptible
-1	Perceptible but not annoying
-2	Slightly annoying
-3	Annoying
-4	Very Annoying

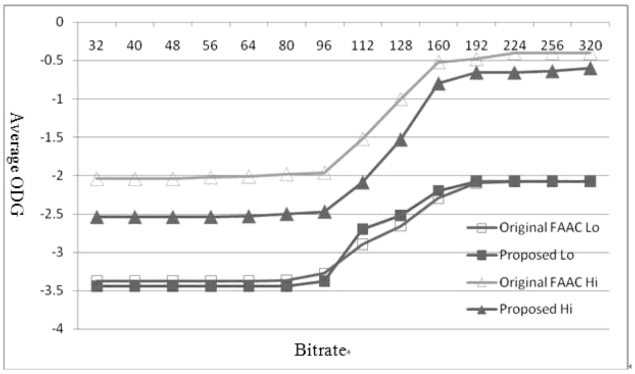


Fig. 9 Comparative results of the sound quality between our proposed encoder and the floating-point FAAC.

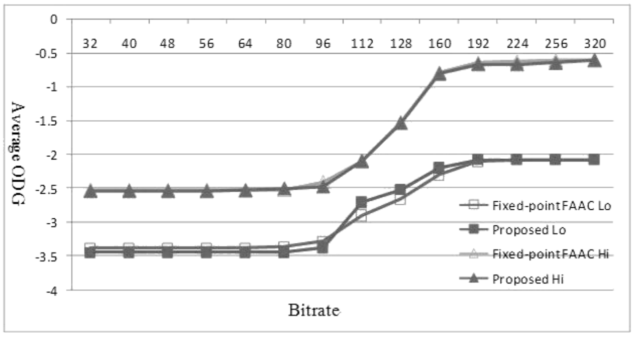


Fig. 10 Comparative results of the sound quality between our proposed encoder and the fixed-point FAAC.

ory resources, lower CPU clock frequencies, and fixed point architectures. Thus, some platform-based optimizations associated with the DSP-based systems are necessary for efficiently implementing our proposed AAC codec on the system. In this subsection, we demonstrate the proposed AAC codec on the DSP-based platform. Before porting the proposed codec to the target platform, it is necessary to do the fixed-point modifications to the proposed codec. In the classical fixed-point modification, the range and shifted value of the data are assumed to be known a priori and determined beforehand. However, the data in the AAC coding process are mostly random and vary from frame to frame, and applying a fixed shifted value applied to all data will waste resources unnecessarily. Therefore, this study implements a dynamic fixed-point modification method for the AAC codec. The dynamic fixed-point modification focuses on three blocks in the AAC coder, including the total energy computation of each channel, the auotocorrelation for the LDR in the TNS block, and the energy of each scalefactor band.

In accordance with the hardware of the target platform, TMS320C55x DSP, numerous platform-based optimization techniques using C and various assembly programs can be used to improve the performance. Platform-based intrinsic functions are applied to replace the original functions and thus can be directly mapped onto inline C55x instructions using the compiler, Code Composer Studio (CCS). Besides, although TI's C compiler has sophisticated optimization capabilities, programs generated from ordinary C code may still suffer insufficient performance. Therefore, several blocks should be optimized by substituting them with inline assembly codes, such as the bit-reverse operations, the FFT kernel processing, and the M/S stereo coding. With regard to this issue, this paper adopts a useful library, known as the DSP Library (DSPLIB) [31] provided by TI, to help increase computational efficiency.

Table 13 lists the average coding Million Instructions per Second (MIPS) and the memory usage needed for the proposed AAC codec, where *.text*, *.bss* and *.cinit* represent the size of the global and local variables, and the initial value of the variables, respectively. Table 13 shows that the proposed AAC encoder and decoder require about only 101 KB and 46 KB, respectively.

Many studies have examined the implementation of the AAC codec on different embedded platforms [31]–[38]. These studies performed their proposed AAC codec on various kinds of embedded platforms. Therefore, to perform a fair comparative experiments, the comparative results of cycle counts and memory shown in this subsection will be normalized in Million Instructions/Operations per Second (MIPS/MOPS) and bytes respectively. Tables 14 and 15 present the comparisons of cycle counts and memory usages of different AAC encoders and decoders [31]–[38] implemented on different embedded platforms.

First, we perform the comparative experiments on AAC encoding of the proposed method and the other existent techniques, as Table 14 shows. Huang et al. [31] implemented an encoder with platform-based optimizations on a DSP-based platform with a computing speed of 350.39 ms/frame, which equals 328 MIPS by (15)(16), where the audio sampling rate is 48 KHz and DSP runs at 40 MIPS speed. Kim et al. [33] proposed a non-iterative quantization method based on the direct noise shaping and level matching and achieved their codec on ARM9 with 85.6 MIPS, which is slightly slower than that of the proposed method. In [34], Huang et al. simplified the PAM and ported it on a 16-bit fixed-point processor. However, the computational efficiency and memory cost of the AAC encoder could still be further enhanced by adopting the optimizations of the aforementioned critical modules. Thus, the proposed encoder promotes more comprehensive algorithmic and software-based optimizations for the AAC encoding flow. For encoding quality performance issues, only Huang et al. [34] presented the quality difference between their proposed codec and the AAC standard in ODG and NMR evaluations. The encoding quality evaluations of the encoders proposed by Huang et al. [31] and Kim et al. [33] are unavailable. However, the experimental results of this study show that although the encoding quality and computational efficiency is a tradeoff, the proposed encoder can still

Table 13 The summary of memory section size and MIPS.

	MIPS	***.text***	***.bss***	***.cinit***	***others***	***total***
Decoder	12MIPS	9.4KB	23.8KB	12.4KB	0.4KB	46KB
Encoder	84MIPS	32.8KB	59.4KB	8.1KB	0.7KB	101KB

Table 14 Comparison of cycle counts and memory in different AAC encoders.

	Proposed	**Huang *et al.*[31]**	**Kim *et al.*[33]**	**Huang *et al.*[34]**
Platforms	OMAP5912 TMS320C55X	ADSP 21060 SHARC	ARM 9	A 16-bit fixed-point processor
MIPS/ MOPS	84 MIPS	328 MIPS (350.39 ms per frame)	85.6 MIPS	6.3 MOPS
Code size	Program size:32.8KB Data size:59.4KB	Program size:78KB Data size:256KB	N/A	Intra:3506 Bytes Inter:4612 Bytes

Table 15 Comparison of cycle counts and memory in different AAC decoders.

	Proposed	**Chen *et al.*[35]**	**Bang *et al.*[37]**	**Choi *et al.*[38]**	**Waston *et al.*[36]**
Platforms	OMAP5912 TMS320-C55X	TMS320-C6201	Hybrid-architecture	TMS320-C5510 with ARM9 RISC core	Motorola fixed-point DSP
MIPS/ MOPS	12 MIPS	29.65 MIPS	16.9 MIPS	86 MIPS (80fps)	24 MIPS
Code size	Program size:9.4KB Data size:23.8KB	Program size:14.7KB Data size:2.84KB	Program size:8.2KB Data size:25.2KB	N/A	Program size:10KB Data size:24KB

achieve well-optimized performance in both computational complexity and memory costs, and also provide satisfactory encoding quality, as Sect. 3.2 illustrates. Table 14 shows that the proposed encoder can provide the best encoding speed and the least memory cost.

$$40 \times 10^6 \text{instructions/sec} \times 350.39\text{ms/frame} = 14 \times 10^6 \text{instructions/frame} \quad (15)$$

$$\text{MIPS} = \frac{\text{cycle counts per frame}}{\text{samples per frame}} \times \frac{\text{sample rate}}{10^6} = \frac{14 \times 10^6}{2048} \times \frac{48000}{10^6} = 328 \quad (16)$$

Table 15 shows the comparisons between the proposed and other techniques of the AAC decoders. Previous studies [35], [36] and this study design the AAC decoders with the LC Profile, while other studies [37], [38] develop the Main Profile AAC decoders. As depicted in Table 15, the proposed decoder provides significantly better computational efficiency compared with those in [35] and [38], which ports their algorithms on TI DSP embedded platforms. Considering the decoding speed of presented work, sets of look-up tables are adopted to replace and accelerate numerous complex computations in the decoding flow, such as the IFFT and the inverse quantization module. Therefore, although the required memory storage of the proposed decoder is somewhat greater than those of the method in [35], the memory cost is still acceptable for embedded applications. Bang et al. [37] implemented an optimized Main Profile AAC decoder on their hybrid architecture, which contained a fixed-point DSP core and hardwired logic modules. Although hardware implementation is a good solution for reducing the computational complexity and memory usage requirements, its implementation costs may be significantly higher. Choi et al. [38] presented an implementation of AAC and MPEG4 decoders on a combined DSP and RISC platform. Equations (17) and (18) show that the decoding speed of [38] is approximately 86 MIPS where the working speed of the TI C5510 DSP is about 320 MIPS. The comparative results in Table 15 show that the proposed AAC decoder can effectively perform with the least computational loading and sufficiently low memory costs.

$$\frac{320\times10^6\text{instructions/sec}}{80\text{frames/sec}} = 4\times10^6\text{instructions/frame} \quad (17)$$

$$\text{MIPS} = \frac{\text{cycle counts per frame}}{\text{samples per frame}} \times \frac{\text{sample rate}}{10^6} = \frac{4 \times 10^6}{2048} \times \frac{44100}{10^6} = 86 \quad (18)$$

4. Conclusions

This study has been presented several software-based and algorithmic optimization approaches on computational efficiency and memory reduction for the AAC codec, including the optimization schemes of MDCT/IMDCT, TNS, M/S stereo coding, inverse quantization and Huffman coding modules. Based on the optimization to FFT computation above, the memory usage of this module has been reduced from 2,304 bytes to 584 bytes. To optimize the TNS module, this study has presented a fast scheme, which significantly reduces the computational complexity of TNS by 78.37 %, for the TNS flow in the encoder. To promote the computational efficiency of the M/S stereo coding module, this study has adopted an optimized M/S coding decision scheme, which achieves a significant reduction (73.6 %) in computational costs compared to the original one, by analyzing the energy of a single frame. Additionally, to improve the inverse quantization module, this study has also performed a mixed-up optimization of the look-up table and the piecewise linear approximation method to substitute the computation of $\left|x_q\right|^{\frac{1}{3}}$. In accordance with reducing the memory consumption of the Huffman coding module, this study has proposed a reduction architecture, which yields a 16.34 % reduction in memory usage and only costs 1.10 % expansion of encoded files, for the Huffman tables based on statistical analysis results. Besides, We adopt 10 typical audio samples for statistical analysis, where 5 audio samples are pop songs with long length and the others are signals of single instruments and vocals, such as soprano, quartet, horn, trumpet and glockenspiel, to ensure the proposed optimizations for each module is effective and robust. The variety of tested audio samples covers various frequencies of signals and certifies the proposed method for confidence. Accordingly, the computational speeds of the proposed fixed-point encoder and decoder are about 3.5 times and 2.3 times faster than those of the FAAC and FAAD2, respectively. Audio quality evaluation demonstrates that the performance of the presented AAC encoder is also comparable to the fixed-point FAAC for embedded applications. To more effectively perform the proposed AAC codec on embedded systems, this study has also adopted platform-based optimizing techniques, such as the dynamic fixed-point and DSP-based modification, for DSP-based embedded systems. With the dynamic fixed-point modification, the proposed encoding architecture only requires about 84 MIPS computational costs and 101 KB memory usage, while the decoding process requires only 12 MIPS computational costs and approximately 46 KB memory storage. Experimental results demonstrate that the proposed AAC codec is computationally efficient, and is suitable for various embedded and handheld multimedia applications, such as video codecs, automotive entertainment and surveillance applications.

Acknowledgments

This research was supported by National Science Council under Grant no. NSC 98-2752-E-009-003-PAE and NSC-98-2220-E-027-010.

References

[1] T. Mochizuki, “Perfect reconstruction conditions for adaptive block-

size MDCT," IEICE Trans. Fundamentals, vol.E77-A, no.5, pp.894–899, May 1994.

[2] T. Moria, A. Jin, T. Mori, K. Ikeda, and T. Kaneko, "Lossless scalable audio coding and quality enhancement," IEICE Trans. Inf. & Syst., vol.E86-D, no.3, pp.425–429, March 2003.

[3] E. Zwicker and H. Fastl, Psychoacoustics: Facts and Models, SpringerVerlag, New York, 1990.

[4] MPEG-4 Version 1 Reference Software (ISO/IEC 14496-5:2000) http://sound.media.mit.edu/mpeg4/audio/software/

[5] Y. Takamizawa, T. Nomura, and M. Ikekawa, "High-quality and processor-efficient implementation of an MPEG-2 AAC encoder," IEICE Trans. Inf. & Syst., vol.E86-D, no.3, pp.418–424, March 2003.

[6] Y. Takamizawa, T. Nomura, and M. Ikekawa, "High-quality and processor efficient implementation of an MPEG-2 AAC encoder," Proc. IEEE 2001 Int'l Conf. on Acoustics, Speech, and Signal Processing, vol.2, pp.985–988, May 2001.

[7] D.H. Kim, D.H. Kim, and J.H. Chung, "Optimization of MPEG-4 GA AAC on general PC," Proc. 44th IEEE 2001 Midwest Symposium on Circuits and Systems, vol.2, pp.923–925, Aug. 2001.

[8] I. Dimkoviae, D. Milovanoviae, and Z. Bojkoviae, "Fast software implementation of MPEG advanced audio encoder," Proc. 14th IEEE 2002 Int'l Conf. on Digital Signal Processing, vol.2, pp.839–843, Santorini, Greece, 2002.

[9] S.W. Huang, T.H. Tsai, and L.G. Chen, "A low complexity design of psycho-acoustic model for MPEG-2/4 advanced audio coding," IEEE Trans. Consum. Electron., vol.50, no.4, pp.1209–1217, Nov. 2004.

[10] X. Hu, G. He, and X. Zhou, "An efficient low complexity encoder for MPEG advanced audio coding," Proc. 8th IEEE Int'l Conf. on Advanced Communication Technology, vol.3, pp.1501–1505, Korea, Feb. 2006.

[11] E. Kurniawati, C.T. Lau, B. Premkumar, J. Absar, and S. George, "New implementation techniques of an efficient MPEG advanced audio coder," IEEE Trans. Consum. Electron., vol.50, no.2, pp.655–665, May 2004.

[12] C.H. Yang and H.M. Hang, "Efficient bit allocation algorithm for MPEG-4 advanced audio coding," Proc Fortieth Asilomar Conf. on Signals, Systems and Computers, pp.2119–2124, Monterey, California, USA, Oct.-Nov. 2006.

[13] E. Alexandre, A. Pena, and M. Sobreira, "Low-complexity bit-allocation algorithm for MPEG AAC audio coders," IEEE Signal Process. Lett., vol.12, no.12, pp.824–826, Dec. 2005.

[14] A. Servetti, A. Rinotti, and J.C. De Martin, "Fast implementation of the MPEG-4 AAC main and low complexity decoder," Proc. IEEE 2004 Int'l Conf. Acoustics, Speech, and Signal Processing, vol.5, pp.249–252, Montreal, Canada, May 2004.

[15] K.H. Bang, J.S. Kim, N.H. Jeong, Y.C. Park, and D.H. Youn, "Design optimization of MPEG-2 AAC decoder," IEEE Trans. Consum. Electron., vol.47, no.4, pp.895–903, Nov. 2001.

[16] M.A. Watson and P. Buettner, "Design and implementation of AAC decoders," IEEE Trans. Consum. Electron., vol.46, no.3, pp.819–824, Aug. 2000.

[17] S.H. Yoon, M.H. Sunwoo, and J.H. Moon, "Design of a high-quality audio-specific DSP core," Proc. IEEE 2005 Workshop on Signal Processing Systems Design and Implementation, pp.509–513, Athens, Greece, Nov. 2005.

[18] Y. Takamizawa, K. Nadehara, M. Boegli, M. Ikekawa, and I. Kuroda, "MPEG-2 AAC 5.1-channel decoder software for a low-power embedded RISC microprocessor," J. VLSI Signal Process., vol.29, no.3, pp.247–257, Nov. 2001.

[19] C.W. Yoon and H.J. Yoo, "Low power motion estimation and motion compensation block IPs in MPEG-4 video codec hardware for portable applications," IEICE Trans. Electron., vol.E86-C, no.4, pp.553–560, April 2003.

[20] 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 Trans. Consum. Electron., vol.47, no.3, pp.613–621, Aug. 2001.

[21] K. Sayood, Introduction to data compression, Morgan Kaufmann, New York, 2000.

[22] C.H. Yen, Y.S. Lin, and B.F. Wu, "A low-complexity MP3 algorithm that uses a new rate control and a fast dequantization," IEEE Trans. Consum. Electron., vol.51, no.2, pp.571–579, May 2005.

[23] R. Gluth, "Regular FFT-Related Transform kernels for DCT/DST-based polyphase filter banks," Proc. IEEE 1991 Int'l Conf. on Acoustics, Speech, and Signal Processing, vol.3, pp.2205–2208, Toronto, Canada, May 1991.

[24] G. Bonnerot and M. Bellanger, "Odd-time odd-frequency discrete Fourier transform for symmetric real-valued series," Proc. IEEE, vol.64, pp.392–393, March 1976.

[25] M. Orchard, "Fast bit-reversal algorithms based on index representations in GF (2b)," IEEE Trans. Signal Process., vol.40, no.4, pp.1004–1008, April 1992.

[26] SQAM—Sound Quality Assessment Material: EBU SQAM disc tracks. [Online]. URL: http://www.ebu.ch/en/technical/publications/tech3000_series/tech3253/index.php

[27] FAAC, FAAD2—Freeware Advanced Audio Coder [online] URL: http://www.audiocoding.com

[28] MPEG. MPEG-2 Advanced Audio Coding, AAC, International Standard IS 13818-7, ISO/IEC JTC1/SC29 WG11, 1997.

[29] ITU-R Recommendation BS.1387, "Method for objective measurements of perceived audio quality," 2001.

[30] S.D. You and W.K. Chen, "Efficient quantization algorithm for real-time MP-3 encoders," Multimedia Tools and Applications, vol.40, no.3, pp.341–359, Dec. 2008.

[31] TMS320C55x DSP Library Programmer's Reference.

[32] D.Y. Huang, X. Gong, D. Zhou, T. Miki, and S. Hotani, "Implementation of the MPEG-4 advanced audio coding encoder on ADSP-21060 SHARC," Proc. 1999 IEEE Int'l Symp. on Circuits and Systems, vol.3, pp.544–547, Orlando, USA, June 1999.

[33] B.I. Kim, E.S. Lee, and T.G. Chang, "A quantization method of direct noise shaping and level matching for the low complexity implementation of MPEG-4 AAC encoder," Proc. IEEE 2005 Digest of Technical Papers Int'l Conf. on Consumer Electronics, pp.261–262, Las Vegas, USA, Jan. 2005.

[34] S.W. Huang, L.G. Chen, and T.H. Tsai, "Memory and computationally efficient psychoacoustic model for MPEG AAC on 16-bit fixed-point processors," Proc. IEEE 2005 Int'l Symp. on Circuits and Systems, vol.4, pp.3155–3158, Kobe, Japan, May 2005.

[35] J. Chen and H.M. Tai, "MPEG-2 AAC decoder on a fixed-point DSP," IEEE Trans. Consum. Electron., vol.45, no.4, pp.1200–1205, Nov. 1999.

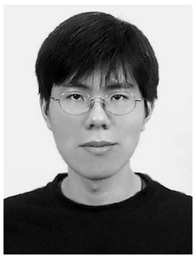
[36] M.A. Watson and P. Buettner, "Design and implementation of AAC decoders," IEEE Trans. Consum. Electron., vol.46, no.3, pp.819–824, Aug. 2000.

[37] K.H. Bang, J.S. Kim, N.H. Jeong, Y.C. Park, and D.H. Youn, "Design optimization of MPEG-2 AAC decoder," IEEE Trans. Consum. Electron., vol.47, no.4, pp.895–903, Nov. 2001.

[38] B.D. Choi, K.S. Choi, S.J. Ko, and A.W. Morales, "Efficient real-time implementation of MPEG-4 audiovisual decoder using DSP and RISC chips," Proc. IEEE 2003 Int'l Conf. on Consumer Eletronics, pp.246–247, Hong Kong, China, June 2003.

Bing-Fei Wu was born in Taipei, Taiwan in 1959. He received the B.S. and M.S. degrees in control engineering from National Chiao Tung University (NCTU), Hsinchu, Taiwan, in 1981 and 1983, respectively, and the Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, in 1992. Since 1992, he has been with the Department of Electrical Engineering and Control Engineering, where he is currently a Professor. He has been involved in the research of Intelligent Transportation Systems for many years and leads a research team to develop the first Taiwan smart car, TAIWAN *i*TS-1, with autonomous driving and active safety system. His current research interests include data compression, vision-based vehicle driving safety, intelligent vehicle control, multimedia signal analysis, embedded systems and chip design.

Hao-Yu Huang was born in Kaoshiung, Taiwan, R.O.C., in 1983. He received the B.S. degree in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2004. In present, he is studying for the Ph.D. degree electrical and control engineering from the National Chiao-Tung University, Hsinshu, Taiwan. His research interests include audio and video coding, and embedded system design.



Yen-Lin Chen was born in Kaohsiung, Taiwan in 1978. He received the B.S. and Ph.D. degrees in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2000 and 2006, respectively. From Feb. 2007 to Jul. 2009, he was an Assistant Professor at the Dept. of Computer Science and Information Engineering, Asia University, Taichung, Taiwan. Since Aug. 2009, he is now an Assistant Professor at the Dept. of Computer Science and Information Engineering, National Taipei University of Technology, Taipei, Taiwan. Dr. Chen is a member of the IEEE and IAPR. In 2003, he received Dragon Golden Paper Award sponsored by the Acer Foundation and the Silver Award of Technology Innovation Competition sponsored by the AdvanTech. His research interests include image and video processing, pattern recognition, document image analysis, and intelligent transportation system.

Hsin-Yuan Peng was born in Taitung, Taiwan, R.O.C., in 1980. He received the B.S. and Ph.D. degrees in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2002 and 2008, respectively. He is currently working toward the Ph.D. degree at the same institute. His research interests include video processing, and embedded system design. Mr. Peng received the First Prize Award of TI China-Taiwan DSP Design Contest in 2006, and the Silver Award of Technology Innovation Competition sponsored by the Advantech Foundation in 2003.

Jia-Hsiung Huang received the B.S. degree in electrical and control engineering from National Chiao Tung University, Hsinchu, Taiwan, in 2006. His research interests include video and audio coding, and embedded system design.