SmartJukebox: An Efficient and High-Quality MPEG-2 AAC Encoder

By Yuichiro TAKAMIZAWA,* Toshiyuki NOMURA* and Masao IKEKAWA*

ABSTRACT This paper describes high-quality and processor-efficient software implementation of an MPEG-2 AAC LC Profile encoder utilized in "SmartJukebox," music jukebox software. MDCT and quantization processing are accelerated by 21.3% and 19.0%, respectively, through the use of SIMD instructions. In addition, psycho-acoustic analysis in the MDCT domain makes the use of FFTs unnecessary and reduces the computational cost of the analysis by 56.0%. The results of subjective quality tests show that better sound quality is provided by greater efficiency in quantization processing and Huffman coding. All of this results in high-quality and processor-efficient software implementation of an MPEG-2 AAC encoder. Subjective test results show that the sound quality achieved at 96kbps/stereo is equivalent to that of MP3 at 128kbps/stereo. The encoder works 13 times faster than real time for stereo encoding on an 800MHz Pentium III processor.

KEYWORDS MPEG-2 AAC, Audio, Encoder, Compression, SIMD

1. INTRODUCTION

While MPEG-1/Audio Layer III (MP3)[1] has been widely used as a high-quality audio-coding algorithm for portable audio devices, PC jukebox software, and Internet music-distribution systems, MPEG-2 Advanced Audio Coding (AAC)[2] has already been standardized as a more sophisticated next-generation technology. AAC provides an audio signal that has CD quality at 96-128kbps/stereo, and a bit rate 30% lower than that of MP3. In the next few years, a wide range of AAC products, including portable audio devices and PC jukebox software, are expected to appear on the market.

Currently, PC-encoder software is usually attached to such products, and customers demand that it provides both high sound quality and fast encoding. Generally, however, these two demands require a trade-off: better sound quality usually results in slower encoding. While this problem might be overcome with sufficiently detailed information regarding encoder implementation, the AAC standard document[2] only describes decoding procedures and bit stream format and does not inform about high-quality, processor-efficient implementations. However, the performance of an encoder strongly depends on how it is implemented.

We have developed the required techniques for high-quality, processor-efficient implementation. These methods for improving both encoding speed and sound quality, which can be applied to generic DSPs and microprocessors, are described in Section 3. Furthermore, we explain in Section 4 how SIMD (Single Instruction stream-Multiple Data streams) instructions can be used to further increase encoding speed. The sound quality and encoding speed are evaluated in Section 5. A software product that utilizes the developed encoder software is introduced in Section 6, and we conclude in Section 7.

2. AAC ENCODING ALGORITHM

MPEG-2 AAC has three profiles, namely, Main, Low Complexity (LC), and Scaleable Sampling Rate (SSR) profiles. The developed encoder is designed for the LC profile because this profile has been adopted for use in Japanese digital TV and currently seems to be the most widely used in audio applications.

Figure 1 shows a block diagram of the AAC LC profile encoder. An MDCT (Modified Discrete Cosine

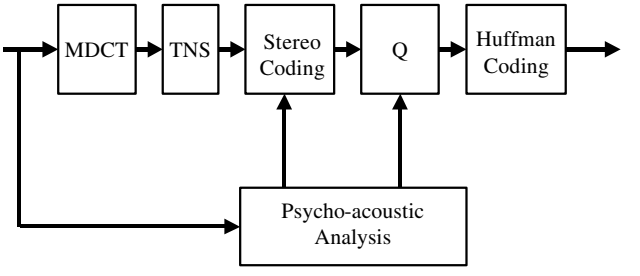

Fig. 1 AAC LC profile encoder.

*Multimedia Research Laboratories

Transform)[3] block transforms an input audio signal into MDCT coefficients, which represent a frequency spectrum. The transform is either 2048-point MDCT (long block) or 256-point MDCT (short block) depending on the characteristics of the input audio signal. The MDCT coefficients are non-uniformly quantized based on a masking threshold after redundancies have been removed in the TNS (Temporal Noise Shaping) and the Stereo coding blocks. The quantized MDCT coefficients are then Huffman coded and multiplexed into a bit stream. The psycho-acoustic analysis block calculates the masking threshold as the maximum distortion energy that is masked by the signal energy. The masking threshold is used to control the quantization step so as to minimize audible quantization error.

The ISO/IEC provides one example of software implementation of AAC encoder[2], but its performance is insufficient in terms of both sound quality and encoding speed. It encodes over eight times more slowly than real time (Pentium III 800MHz, 44.1kHz, 96kbps/stereo), and its sound quality is generally worse than that of MP3 at the same bit rate.

3. QUALITY AND SPEED ENHANCEMENT

We used three methods for our encoder software to improve encoding speed and sound quality. These methods can be applied to implementing on generic DSPs and microprocessors.

3.1 Fast Psycho-Acoustic Analysis

In conventional encoder implementations[1,2], the input PCM signal is transformed by FFT (Fast Fourier Transform), and psycho-acoustic analyses, such as masking calculations, are performed on the FFT coefficients[1,2,4]. We found, however, that the 2048-point MDCT (256-point MDCT for short blocks) used in AAC has sufficient frequency resolution for psycho-acoustic analysis and can replace the 2048-point FFT (256-point FFT for short blocks) without causing sound-quality degradation.

Consequently, we have omitted the FFT calculation and utilized existing MDCT coefficients, which are generated in the MDCT block, for the psycho-acoustic analysis. Substituting the existing MDCT calculation for the FFT one accelerates the encoding speed while maintaining the sound quality.

In the MDCT coefficients, the phase information of the input PCM signal that is utilized in conventional FFT based psycho-acoustic analysis[1,2] is lost. We modified the conventional psycho-acoustic analysis algorithm[4], which is performed on FFT coefficients, to fit the analysis on the MDCT coefficients[5].

3.2 Smoothing of Scalefactor Values

In AAC, 1024-MDCT coefficients are grouped into 49 bands called scalefactor bands. For each scalefactor band, the quantization block searches for the quantization step that achieves the best sound quality below a given bit rate. The derived quantization steps for each scalefactor band sfb are expressed as scalefactor values scalefactor[sfb]. These values are integer values and are differential-coded along the frequency (scalefactor band) on the basis of the values shown in Table I[2].

Since this table is designed so that the smaller differential values can be coded with shorter codes, sudden major changes or successive minor changes in scalefactor values might make the code longer and degrade the coding efficiency. To prevent this, smoothing (low-pass filtering) is applied to the scalefactor values to suppress the changes.

In our implementation, scalefactor values scalefactor[sfb] for each scalefactor band sfb are smoothed in the scalefactor band order (1, 2, ..., 49) through the following procedures.

1) Calculate differential value diff by
   diff = scalefactor[sfb] – scalefactor[sfb-1]
2) Suppress major changes by making diff = 0.8 diff
3) If diff is +1 or –1, set diff to zero to suppress successive minor changes.
4) Set scalefactor[sfb] to scalefactor[sfb-1]+diff

This method decreases the code bits for the scalefactor and increases those that can be used for

Table I Code table for differential scalefactor values.

Differential value	Code
60	111111111111110011
:	:
3	11011
2	1100
1	1010
0	0
–1	100
–2	1011
–3	11010
:	:
–60	111111111111101000

the quantized values. Although low-pass filtered scalefactor values may not be the best ones from the psycho-acoustic point of view, we have confirmed that bit-reduction by using this method improves the sound quality.

3.3 Selection of Huffman Tables

The quantized MDCT coefficients are Huffman coded, in which one out of 12 Huffman tables is selected for each scalefactor band. In general, the table that outputs the shortest code is selected. The numbers (0, ..., 11), which indicate the selected table in each scalefactor band, are run-length coded and multiplexed into the bit stream as Huffman table information.

In run-length coding, a longer run improves the coding efficiency. The bit count for the Huffman table information should be considered when a table is selected, and selecting the one that minimizes the Huffman code is not necessarily the best approach. A Huffman table should be selected so that the total bit count for the Huffman code and Huffman table information is minimized[2]. To this end, we have employed the following procedures in our encoder software to select Huffman tables.

1) For each scalefactor band sfb, select a Huffman table that enables the shortest Huffman code, and set the selected Huffman table number to cb[sfb]
2) Set sfb to 1
3) If cb[sfb] = cb[sfb−1], go to 10) because cb[sfb] is already suitable for run-length coding and should not be changed
4) Calculate the total bit count normal_bits for the Huffman code and Huffman table information for scalefactor band 0, ..., sfb+1
5) Calculate the total bit count bits1 for Huffman code and Huffman table information for scalefactor band 0, ..., sfb+1 with cb[sfb] = cb[sfb−1]
6) Calculate the total bit count bits2 for Huffman code and Huffman table information for scalefactor band 0, ..., sfb+1 by replacing the run of cb[sfb−1], which starts from (sfb−1) to lower frequency, with that of cb[sfb]
7) If normal_bits is the minimum among normal_bits, bits1, and bits2, do nothing and go to 10)
8) If bits1 is the minimum among normal_bits, bits1, and bits2, let cb[sfb] = cb[sfb−1]
9) If bits2 is the minimum among normal_bits, bits1, and bits2, replace the run of cb[sfb−1] with that of cb[sfb]
10) If sfb < 49, increment sfb and go to 3)

In these procedures, each neighboring run is examined for merging {at steps 5), 6)}. Generally, merging the runs reduces the bit count for Huffman table information and increases that for the Huffman code. If the total bit count for the Huffman code and Huffman table information is reduced, the runs are merged { at steps 8), 9)}.

Optimizing the selection of Huffman tables while considering both the Huffman code and the table information improves coding efficiency. Figure 2 shows the reduced bit count for each frame when a typical pop music song (261 sec, 11,255 frames) is encoded at 96kbps/stereo (2,229 bits/frame). The average reduced bit count is 267 bits, which is equivalent to 11.5kbps reduction (gaining) in the bit rate. This means that the sound quality at 96kbps can be derived at 84.5kbps (96 – 11.5kbps) through this method. Looking at it another way, this method enables better sound quality at the same bit rate compared to the conventional.

4. ACCELERATION BY SIMD INSTRUCTIONS

Most recent PC microprocessors have a SIMD instruction set which is designed to accelerate the execution of multi-media applications. This section describes how to utilize the SIMD instruction set for AAC encoder software.

4.1 Parallel MDCT

MDCT can be efficiently implemented by utilizing FFT[6]. SIMD instructions are known to be effective for accelerating such transform operations. Implementation examples of a sub-band synthesis filter in

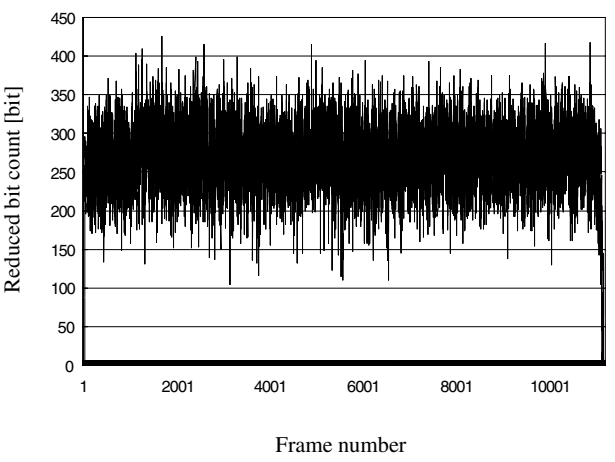

Fig. 2 Reduced bit count in each frame.

MPEG-1 Audio and FFT with SIMD instructions have been reported in [7,8]. These methods find the parallelism in one transform operation and use SIMD instructions to execute multiple operations in one transform by one instruction. To use the SIMD instructions with the greatest efficiency, values to be stored on a SIMD register should be located on consecutive memory addresses.

However, this is not possible with sub-band synthesis filters or FFTs because of their complex signal-flow. They need re-ordering or packing instructions to store multiple values located at non-consecutive memory addresses on a SIMD register. The overhead resulting from the re-ordering or packing instructions degrades execution speed.

To reduce the overhead, we took a different approach to the use of SIMD instructions for MDCTs. Although conventional methods use SIMD instructions to perform multiple operations in one MDCT, our method uses them to perform a single operation in multiple MDCTs. We have implemented this method on an Intel Pentium III processor that has a SIMD instruction set (SSE: Streaming SIMD Extension)[9] that performs four floating-point operations in parallel. In a stereo AAC encoder, four MDCTs (left and right channels of current and next frames) are performed at the same time by using SIMD instructions.

This method reduces the overhead caused by the re-ordering or packing operations described above. By interleaving and storing the four input signals to the MDCT, complex signal-flow operations can be easily and efficiently implemented by the SIMD instructions.

By using this method, MDCT processing is accelerated by 27% without any degradation in sound quality. There are two reasons why the processing is not accelerated by 75% by 4-parallel processing. One is that the SIMD instructions are not four times faster than conventional floating-point instructions[9]. The other is that we simply rewrote the C-code using the intrinsic functions[10]. Performance might be improved further by rewriting with assembly code. Our method is also applicable to other coding/decoding systems, such as MP3.

4.2 Parallel Quantization

In the quantization block, the calculation of (M^0.75) is executed repeatedly. In general, this calculation should be performed as follows:

$$\mathrm{sqrt}(\mathrm{sqrt}(M) \times M).$$

By using SIMD instructions, this calculation is done in parallel. The simplest way is to replace “sqrt” with a SIMD sqrt instruction. However, this is not the best approach for a Pentium III processor. In an SSE instruction set, “sqrt” is slower than “rsqrt” which calculates (M^–0.5). By using “rsqrt” with “rcp,” which calculates (1/M), a processor-efficient implementation of (M^0.75) can be achieved as follows:

$$\mathrm{rsqrt}(\mathrm{rcp}(M) \times \mathrm{rsqrt}(M)).$$

These instructions execute four (M^0.75) calculations in parallel. By using SIMD instructions, the quantization operation is accelerated by 20%.

5. PERFORMANCE EVALUATION

5.1 Encoding Speed

We used a PC with 800MHz Pentium III processor to evaluate the performance of our encoder software. Table II shows the consumed CPU cycles in each AAC block when a typical pop music song (44.1kHz, stereo) was encoded at 96kbps/stereo in real time. The encoder software was accelerated by the fast psycho-acoustic analysis described in Section 3.1 and by utilizing SIMD instructions as described in Sections 4.1 and 4.2.

By employing the new methods described above, our encoder software achieves real time encoding with a 48.6MHz CPU (excluding file access), and works 13 times faster than real time (including file access).

5.2 Sound Quality

We subjected 11 trained listeners to a subjective quality test using CMOS (Comparison Mean Opinion Score) test methodology[11]. The sequence played to the listeners for each trial was Ref/A/B, Ref/A/B, where Ref was the original (not coded) sound, and A

Table II Consumed CPU cycles (Mcycles/s/stereo).

Processing Block	Optimized (reduction [%])	Not Optimized
MDCT	5.9 (21.3)	7.5
TNS	5.8	5.8
Quantization	20.6 (19.0)	24.7
Psycho-acoustic ana.	8.8 (56.0)	20.0
Others	7.5	7.5
Total	48.6 (25.8)	65.5

and B were both coded signals. The assignment of encoders (MP3/AAC) to positions A and B was randomized and unknown to the listener. The listeners were asked to judge whether “A” or “B” had better sound quality by using a seven-grade comparison scale (Table III). The playback was done using STAX Lambda Nova headphones in a controlled (acoustically isolated) room. The MP3 encoder used in the test was a commercial software product well known for its high sound quality. Figure 3 shows the test results (average scores / 95% confidence interval) obtained. The figure shows a comparison of subjective sound quality for “castanets,” “pop music,” “glockenspiel,” and “Suzanne Vega,” all of which are known as critical materials. As the figure indicates, the sound quality of our AAC encoder was significantly better than that of MP3 at the same bit rate (96kbps/stereo) and was equivalent to or better than that of MP3 at 128kbps.

6. SOFTWARE PRODUCT

The developed encoder software for PCs is utilized in our music jukebox software product “SmartJukebox” shown in Fig. 4. Technologies described in Sections 3 and 4 provide both high sound quality and fast encoding speed that customers demand for the music jukebox software. In addition to high performance, “SmartJukebox” provides easy operation by introducing voice recognition technology. Users can operate the software with their voice, such as “Play,” “Stop,” “Next,” etc.

Table III Seven-grade comparison scale.

B is much better than A	+3
B is better than A	+2
B is slightly better than A	+1
B is the same as A	0
B is slightly worse than A	–1
B is worse than A	–2
B is much worse than A	–3

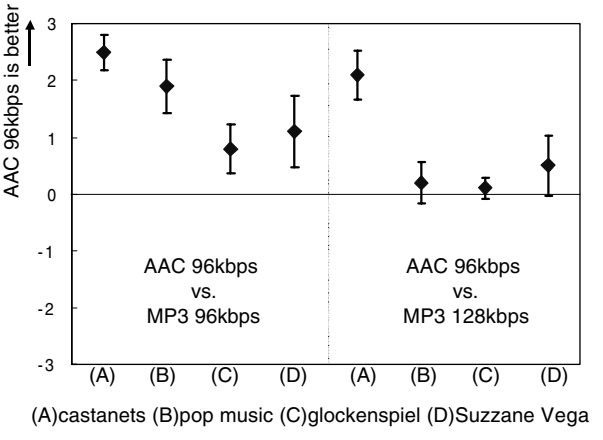

Fig. 3 Subjective test results.

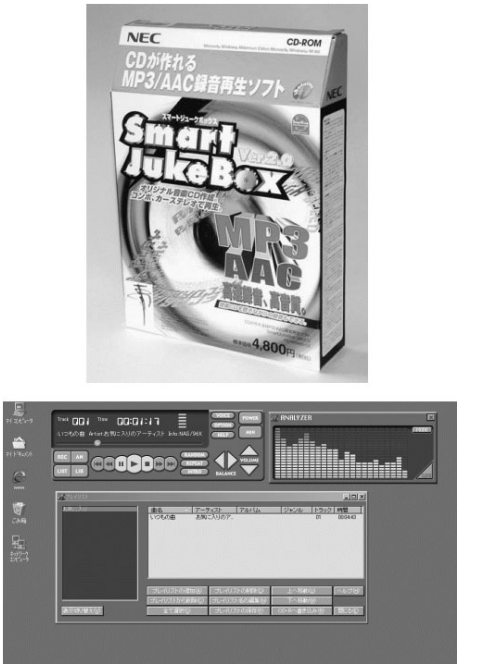

Fig. 4 Software product “SmartJukebox.”

7. CONCLUSION

We have developed MPEG-2 AAC LC profile encoder software. We introduced several new methods to enhance sound quality and encoding speed to provide a high-quality, processor-efficient implementation of this software. The psycho-acoustic analysis on MDCT coefficients and the introduction of SIMD instructions into the MDCT and quantization processing accelerated the encoder software by 25.8% while the sound quality was maintained. The smoothing of scalefactor values and optimized selection of Huffman tables helped improve sound quality. The encoder achieved a significantly better sound quality than MP3, and works 13 times faster than real time for stereo encoding on an 800MHz Pentium III processor.

ACKNOWLEDGMENTS

The authors thank Dr. Ichiro Kuroda, Dr. Akihiko Sugiyama, and Dr. Masahiro Serizawa for their

invaluable encouragement and support.

REFERENCES

[1] ISO/IEC 11172-3, "Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbit/s, Part 3: Audio," Aug. 1993.

[2] ISO/IEC 13818-7, "Generic coding of moving pictures and associated audio, Part 7: Advanced Audio Coding (AAC)," Mar. 1995.

[3] J. Princen and A. Bradley, "Analysis/Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation," IEEE Trans. on ASSP, **34**, pp.1153-1161, Oct. 1986.

[4] J. D. Johnston, "Transform Coding of Audio Signals Using Perceptual Noise Criteria," IEEE J. on Selected Areas in Communications, **6**, 2, Feb. 1988.

[5] T. Nomura and Y. Takamizawa, "Processor-Efficient Implementation of a High Quality MPEG-2 AAC Encoder," 110th Audio Engineering Society Convention Paper, May 2001.

[6] D. Sevic and M. Popovic, "A New Efficient Implementation of the Oddly Stacked Princen-Bradley Filter Bank," IEEE Signal Processing Letters, **1**, 11, pp.166-168, Nov. 1994.

[7] Intel, "Using MMX(tm) Instructions to Implement a Synthesis Sub-Band Filter for MPEG Audio Decoding," http://developer.intel.com/software/idap/resources/technical_collateral/mmx/ap533.htm.

[8] Intel, "Using MMX(tm) Instructions to Perform Complex 16-Bit FFT," http://developer.intel.com/software/idap/resources/technical_collateral/mmx/AP555.HTM.

[9] K. Diefendorf, "Pentium III = Pentium II + SSE," MICROPROCESSOR REPORT, **13**, 3, Mar. 1999.

[10] Intel, "Software Development Strategies for Streaming SIMD Extensions," http://developer.intel.com/vtune/cbts/strmsimd/appnotes/ap814/swdevel.pdf

[11] B. Edler, J. Herre, and K. Brandenburg, "Core experiment methodology for MPEG-4 audio," ISO/IEC JTC1/SC29/WG11, N1748, Jul. 1997.

*Names of companies and products in this paper are trademarks or registered trademarks of each company.

Received January 17, 2003

* * * * * * * * * * * * * * * *

Yuichiro TAKAMIZAWA received the B.E. and M.E. degrees from Waseda University in 1993 and 1995, respectively. He is currently a researcher in Multimedia Research Laboratories at NEC Corporation. He has been working in the area of algorithm and implementation for audio coding.

Mr. Takamizawa is a member of IEEE.

Toshiyuki NOMURA received the B.E. and M.E. degrees from Nagoya University in 1990 and 1992, respectively. He is currently a researcher in Multimedia Research Laboratories at NEC Corporation. He has been working in the area of algorithm and implementation for speech and audio coding.

Mr. Nomura is a member of IEEE, IEICE Japan, and the Acoustical Society of Japan.

Masao IKEKAWA received the B.E. and M.E. degrees in computer science from the University of Electro-Communications in 1985 and 1987, respectively. From 1987 to 1991, he was a graduate student at the Department of Information Sciences at the Tokyo Institute of Technology. He is now working in Multimedia Research Laboratories at NEC Corporation. His current research interests are digital signal processor architecture and multimedia software implementation.

* * * * * * * * * * * * * * * *