Multimed Tools Appl (2008) 40:341–359
DOI 10.1007/s11042-008-0210-7

Efficient quantization algorithm for real-time MP-3 encoders

Shingchern D. You · Woei-Kae Chen

Published online: 10 June 2008

Abstract This paper reports an efficient quantization algorithm for the implementation of a real-time MP-3 encoder based on a low-cost digital signal processor. Unlike the well-known nested-loop quantization algorithm, which requires a large and unpredictable amount of iterations, the proposed algorithm uses a single loop with only three iterations to reduce the computational complexity. Since most of the existing quantization algorithms reported in the literature require peak number of iterations higher than three, our approach can effectively reduce the peak computing demand for a real-time encoder. We conduct several experiments (including the ODG rating) to validate the performance of the proposed algorithm, and the results are acceptable. We implement the proposed algorithm on a 16-bit fixed-point digital signal processor, and the encoder requires 35 MIPS of computation for encoding stereo music at 128 kbps.

Keywords MP-3 · Real-time encoder · Quantization · Digital signal processor · ODG

1 Introduction

The audio coding standard of MPEG-1 part 3 [6] layer III, also known as the MP-3, has been widely used in the past decade. Based on perceptual coding [11], it is almost the *de facto* standard for music compression. Due to market demand, manufacturers are highly interested in the implementation of MP-3 encoders and decoders. Relatively speaking, the encoder is more difficult to implement than the decoder because the standard specifies the decoding, not encoding, procedure. Since the encoding algorithm is not standardized, manufacturers need to develop their own algorithms for their products.

S. D. You (✉) · W.-K. Chen
Department of Computer Science and Information Engineering,
National Taipei University of Technology, Taipei 106, Taiwan
e-mail: scyou@ntut.edu.tw

W.-K. Chen
e-mail: wkchen@ntut.edu.tw

A major advantage of using the MP-3 encoding scheme is that it dramatically reduces the size of audio data while introducing almost imperceptible distortion. For example, a stereo CD sound track has a data rate (or bitrate) of around 1.5 Mbps. When a sound track is encoded in MP-3 at 128 kbps, causal listeners are usually satisfied with the sound quality of the compressed audio. In this case, the MP-3 encoding scheme offers more than ten-times compression ratio. Although encoding a piece of music with a bitrate higher than 128 kbps (e.g., 160 kbps) offers a better audio quality, the perceptual difference is hard to tell for a non-professional, causal listener. Therefore, most audio coding techniques focus on reducing bitrates, while maintaining satisfactory quality, for more versatile applications, such as digital radio [4] or digital TV [3]. In fact, the quality of an encoder is typically judged by its performance at lower bitrates, not higher ones. This is because encoding at a lower bitrate usually introduces a higher distortion, and therefore, it is more challenging to make the distortion imperceptible. For simplicity, we use the term *encoding quality* to mean the quality of the compressed audio produced by an encoder.

Depending on the type of applications, both offline and online MP-3 encoding are possible. Offline encoding is suitable for MP-3 bitstreams that are prepared in advance for future use, e.g., internet downloads. In this case, there is no restriction on encoding time. It is, then, reasonable to spend more time to encode audio so that the encoding quality is fully optimized. An offline encoder is typically implemented in the form of a piece of program running on a general-purpose computer.

Online encoding is suitable for an application that requires a real-time and non-stop encoding service for a continuous music source (e.g., microphone). A portable MP-3 music recorder usually contains a real-time encoder for online encoding. Such an encoder is typically implemented as an IC chip based on a DSP (Digital Signal Processor) [14] plus additional RAM, ROM, and peripheral circuits. This type of design is sometimes referred to as a System-on-Chip (SoC) design. In this case, the computing capability of the processor and the size of the memory greatly affect the cost. In addition, power consumption is a major concern when the chip is used inside a portable device.

To perform the online encoding, a real-time encoder typically has two identical input buffers, called buffer A and buffer B, as shown in Fig. 1. While buffer A is collecting incoming PCM (pulse code modulation) samples, buffer B (holding previously collected samples) supplies samples to the processor for encoding. When buffer A is filled up with input samples, the roles of buffers A and B are exchanged. For a sampling rate of 44.1 ks/s, the time interval to collect 1,152 PCM samples for one frame in buffer A is 26 ms; therefore, the samples stored in buffer B must also be encoded within 26 ms.

Given a maximal time to encode one frame of audio samples, a more complicated encoding algorithm requires a processor with relatively higher performance. Unfortunately, a high-performance processor is not preferable for a portable device due to its higher cost and/or power consumption. Therefore, the encoder of a portable device usually employs a

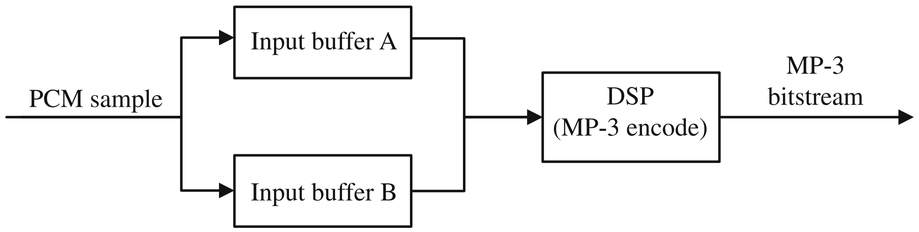


Fig. 1 A real-time MP-3 encoder

low-complexity algorithm. In terms of encoding quality, a real-time encoder is generally inferior to an offline PC-based encoder, especially at lower bit rates, e.g., 128 kbps. During the encoding process, it is usually necessary to search from a large amount of candidates the best representation for the frame to be encoded. For a real-time encoder, due to its time constraint, the search space has to be confined. Consequently, a real-time encoding algorithm must settle for sub-optimal representations, which, nevertheless, leads to inferior encoding quality. As the bitrate increases, the quality difference between these two types of encoders diminishes because, at very high bitrates, both eventually approach the quality of the original signal.

Although MP-3 encoding programs are available from ISO and other parties, these programs are mainly designed for offline applications and typically use sophisticated algorithms to optimize the encoding quality. Unfortunately, the processor embedded inside a real-time encoder chip usually has a very limited computing power. This limitation makes it almost impossible to directly adapt existing programs and algorithms to a real-time encoder chip. By carefully examining the MP-3 encoding flow, we conclude that a better quantization strategy is the key to reducing the complexity of the encoding algorithm.

In this paper, we propose an efficient quantization algorithm for real-time MP-3 encoding using low-cost DSP. The number of iterations for quantization in the proposed approach is fixed to 3. In contrast, the number of iterations in the nested-loop quantization algorithm, suggested in the standard [6], varies dramatically from less than ten to more than fifty. We also show that the encoding quality of the proposed quantization algorithm is acceptable and we present the influence of lower arithmetic accuracy on the encoding quality with some experimental results.

2 The MP-3 encoder

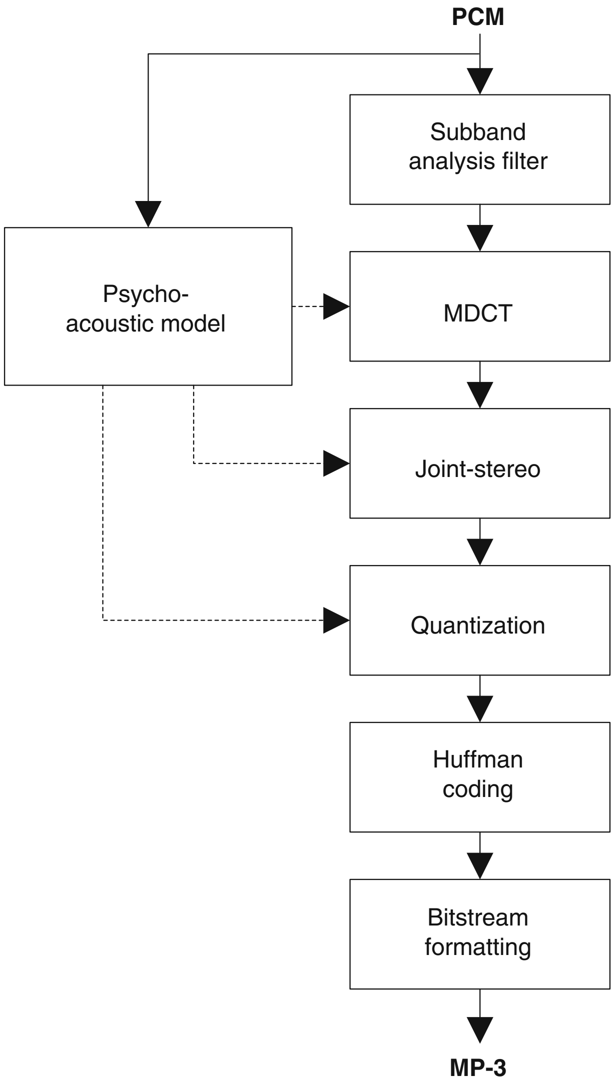
This section covers how an MP-3 encoder produces the bitstream from PCM samples. An overview of the MP-3 encoding flow is given in Section 2.1. The window operation is discussed in Section 2.2. The quantization algorithm suggested by the MPEG-1 audio standard is described in Section 2.3.

2.1 The MP-3 encoding flow

An MP-3 encoder encodes 1152 PCM samples into one frame of bitstream at a time. A frame is divided into two *granules*, each containing information for 576 PCM samples. Figure 2 shows the process of encoding one granule of samples. The first step is subband analysis. A 32-band filterbank is used to convert 576 PCM samples into subband samples. After that, subband samples are multiplied by a window function (to be explained in Section 2.2). To obtain higher frequency resolution for subband samples, the second step performs the MDCT (modified discrete cosine transformation) [12] computation on the windowed subband samples. After the MDCT computation, 576 spectral values are available. However, the spectral values contain some aliasing. Thus, an aliasing reduction operation is performed. Then, the third step, joint stereo coding is carried out if desired.

Quantization of the spectral values is the fourth step. The quantized spectrum is encoded using the Huffman codes. This step requires a loop to adjust the quantization parameters so that the total bits of the Huffman code words do not exceed the available bits for the granule. In order to efficiently use the available bits, the MP-3 standard suggests a nested-loop quantization algorithm (to be explained in Section 2.3). In addition, the MP-3 standard

Fig. 2 The encoding flow for a generic MP-3 encoder. The *solid line* indicates the signal flow and the *dashed line* indicates the control flow



employs a technique called bit reservoir to achieve a short-term bitrate variation. As the name reservoir suggests, if the quantization step does not exhaust all of the available bits in a frame, the unused bits can be reserved for a future frame. Therefore, the number of available bits varies from one frame to another. Finally, the last step, the Huffman code words along with the necessary side information are packed to form bitstream.

2.2 Window operations

There are four different types of windows given in the standard, namely *long*, *short*, *start*, and *stop*. After the MDCT computation, subband samples multiplied by a long window are converted into 18 spectral values, and into six values with a short window. Therefore, three consecutive short windows are used in a subband to produce three sets of spectral values, with each set representing one time-instance, to make a total number of 18 spectral values. Generally speaking, a long window provides a higher frequency resolution, while a short window offers a higher time resolution. In order to smoothly switch from a long window to a short window, a start window is used. For the same reason, a stop window follows a short window before switching to a long window. The psychoacoustic model [6] in the encoder determines whether long or short windows should be used.

A long window is selected to encode a stationary signal for its higher frequency resolution. On the other hand, for a signal with strong attacks (transient signal), short windows are selected for better time resolution. A stationary signal usually exhibits some frequency localization; therefore, higher frequency resolution can better localize the strong spectral components in the spectrum. Higher coding efficiency (gain) can then be achieved by encoding the strong spectral components only. On the contrary, a transient signal exhibits temporally a strong-energy portion and a weak-energy portion. To encode such a transient signal, it is better to individually perform the MDCT on the weak portion and the strong portion of the signal. In this way, both portions of the signal are independently quantized with suitable quantization parameters to better represent the signal. However, encoding with short windows has a higher overhead and lower coding efficiency. Therefore, the encoding quality degrades if short windows are used unnecessarily.

After MDCT computation, spectral values are to be quantized. The spectral values having similar sensitivities to human ears are grouped into a band, known as *scalefactor band*, to share the same quantization parameter *scalefactor*. When long windows are used for all subbands, the spectral values are divided into 21 scalefactor bands. On the other hand, if short windows are selected for all subbands, due to fewer spectral values after MDCT computation, they are divided into 12 scalefactor bands. The information in a short-window granule, therefore, contains three sets of values with each set having 12 scalefactors. The standard also supports to mix varies types of windows in a granule, known as mixed-block mode, where the two lowest-frequency subbands use long windows and the rest of the subbands use short windows.

2.3 The nested-loop quantization algorithm

The quantization step employs a non-linear quantizer with some adjustable parameters. Taking the long window as an example, a simplified version of the quantization equation is as follows:

$$ix_i = \mathrm{NINT}\left[\left(\frac{|xr_i|}{2^{0.25(\mathrm{global\_gain}-210)-\mathrm{scalefac\_l[sfb]}}}\right)^{0.75} - 0.0946\right] \tag{1}$$

where ix_i is the magnitude of the i-th quantized spectral value, global_gain and scalefac_l [sfb] are quantization parameters, and NINT() is a function to find the nearest integer value. The parameters "global_gain" and "scalefac_l[sfb]" (standing for scalefactor with long window) control the step size of the quantizer. The difference is that the global_gain is shared by all spectral values, whereas the scalefac_l[sfb] affects only the spectral values within the "sfb" scalefactor band. A larger global gain increases the value of the denominator in Eq. 1, or, equivalently, increases the quantizer step size. Therefore, for the same xr_i, a larger global gain gives a smaller ix_i, and requires fewer bits to encode. Since global gain is applied to all spectral values, it is used as a "rate control" parameter to ensure that the coded bitstream meets the bitrate constraint. The parameter "scalefac_l[sfb]" controls the quantization noise (also called distortion in the literature) of the indicated scalefactor band. The psychoacoustic model determines the maximum (allowed) distortion that is not perceivable in each scalefactor band. Based on this information, the scalefac_l [sfb] is adjusted accordingly for distortion control.

Since both the global gain and the scalefactors control the quantization step size, the quantization algorithm needs to simultaneously adjust both to meet the bitrate requirement and to achieve the best audio quality. The MPEG-1 audio standard suggests a nested-loop algorithm. The algorithm, shown in Fig. 3, has an outer loop for distortion control and an inner loop for rate control. The outer loop calculates the distortion of each scalefactor band and adjusts the corresponding scalefactor if necessary. The inner loop counts the required bits to encode the quantized spectral values and then uses the global gain to control the encoding bits. The tricky part is that once the global gain is changed, the distortion of each scalefactor band is also changed. Therefore, the outer loop must be re-executed for distortion control. Similarly, adjusting the scalefactor of a scalefactor band affects the overall encoding bits. Thus, the inner loop must be re-executed. Therefore, unlike the other steps in the encoding flow, the overall iterations (outer and inner loops) and subsequently the execution time of the quantization step is not a constant. Note that both of the two steps

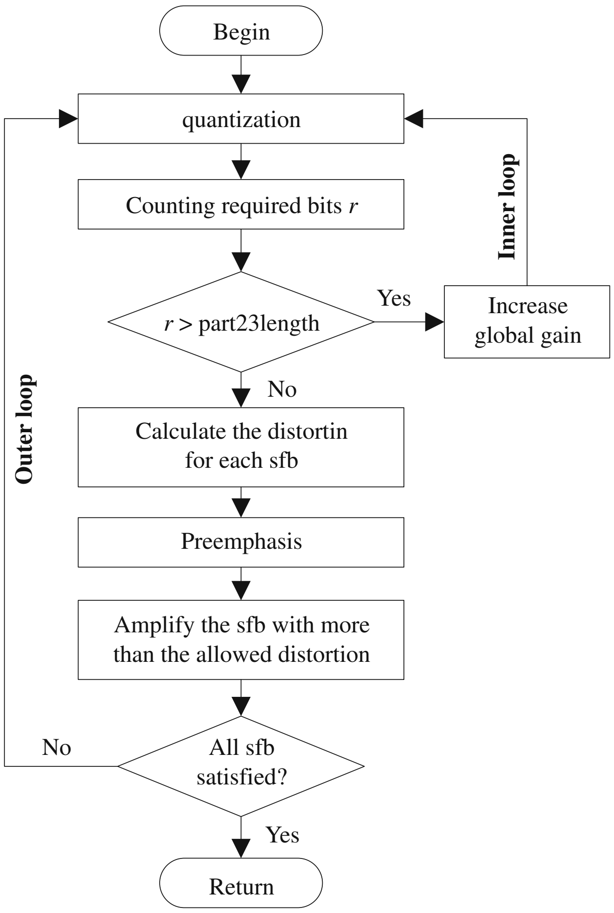


Fig. 3 The nested-loop quantization algorithm [6]. The sfb denotes scalefactor band

in the inner loop (quantization and counting) are computationally expensive, because there are many spectral values to be processed. It is possible to reduce the complexity of the inner loop by using advanced searching algorithms [16]. However, the total number of iterations remains high and unpredictable.

3 The proposed quantization algorithm

The computational demand of the nested-loop quantization algorithm varies dramatically from one piece of music to another. Therefore, even if the average computational demand is low, the peak demand may still be extremely heavy. Since a real-time encoder must encode one frame of music in a constant time, a high performance processor is required to satisfy the peak computational demand. Unfortunately, for cost considerations, the processor embedded in an encoder chip is usually a low-power, low-speed one with little room for the fluctuation of computational demand. Therefore, instead of using the nested-loop algorithm, we need to develop an alternative quantization algorithm for the real-time encoder chip. We will discuss how the proposed algorithm handles window switching, distortion control, and rate control in the following subsections.

3.1 Window switching

The psychoacoustic model suggested in the standard is very complicated, and requires lots of computation. It is estimated to consume 90 MIPS (million instructions per second) [10]. Since a real-time encoder is usually operated with a computing demand of around 40 MIPS, we decide not to implement the psychoacoustic model in our encoder (see also Section 3.2). Without the psychoacoustic model, it is very difficult to determine whether long or short windows should be used for the incoming signal. Therefore, we use only long windows in encoding.

The main disadvantage of using only long windows is that a transient signal (strong attack), which is present in any kinds of music, may not be encoded with enough time resolution. Thus, a strong attack, when encoded with long windows, becomes relatively softer. Fortunately a causal listener cannot easily perceive the difference between a strong and a softer attack. In fact, the idea of using only long windows has been studied for both MP-3 [10, 17] and AAC [18, 1] coding schemes, and acceptable encoding quality, not in favor of any particular kind of music, were reported. Our experimental results, reported in Section 4, also indicate that the quality degredation is acceptable in comparison to a highly optimized encoder that employs the window-switching mechanism.

3.2 Distortion control

Due to the use of the outer loop for distortion control, the total number of iterations required for the nested loop quantization algorithm is unpredictable. In order to reduce and regularize the amount of computation to a manageable degree, we omit the distortion loop and use fixed scalefactors for all scalefactor bands. Recall that the psychoacoustic model is used to estimate the allowed distortion for each scalefactor band. In our case, therefore, the psychoacoustic model is no longer necessary. Without distortion control, the encoding quality of our real-time quantization algorithm is, of course, inferior to that of a regular one. Through experiments described in Section 4, we have quantitative assessment of the quality difference between these two.

3.3 Rate control

Without using the outer loop, the quantization step contains only the inner loop to determine the value of the global gain. For each granule, there exists a *best global gain* (BGG) such that the quantized spectral values, after Huffman coding, consume as many available bits as possible. The number of available bits in a granule, called *part23length* in the standard [6], is mainly determined by the bitrate. If a 16-bit mathematical precision is desired, it is possible to determine the BGG of a granule in six iterations by using a binary search. To further reduce the computational complexity of the inner loop, we present a heuristic algorithm that can find a good approximation for the BGG in three iterations.

Our search algorithm starts with the determination of an *initial global gain* (IGG). The IGG has to be close to the BGG to localize the search to a small range. Usually the BGG is strongly related to the spectral values to be quantized, because a signal with larger spectral values should be quantized with a larger global gain (i.e., larger quantization step size) to keep the number of encoded spectral bits less than part23length. Therefore, the IGG may be obtained through a lookup table based on the spectral values. Specifically, we develop the following equation to find the *energy* (effective bits) eb of all spectral values for this purpose:

$$\mathrm{eb} = \sum_{i=0}^{575} \mathrm{Bits}(|xr_i|), \tag{2}$$

where xr_i is the un-quantized spectral value with spectral index i and Bits(x) is the number of effective bits of the binary number x. For example, 10_2 is a two-bit number; therefore, Bits(10_2)=2. By the same argument, 10111_2 is a five-bit number, so Bits(10111_2)=5.

To further explore the relationship between the BGG and eb in Eq. 2, we develop a program to encode audio with the best global gain. The program, called the BGG encoder, records the eb and its associated BGG for each granule during encoding. The BGG encoder is used to encode a training set of audio, which contains over 1,000 pieces of stereo music. The training audio is carefully selected to include all sorts of music genres so that the derived lookup table is genre independent. The results are shown in Fig. 4 for 128k bitrate. Apparently, the BGG of a granule is almost proportional to its corresponding eb. Thus, by using eb as an index, a simple table lookup offers a good IGG that is very close to the BGG.

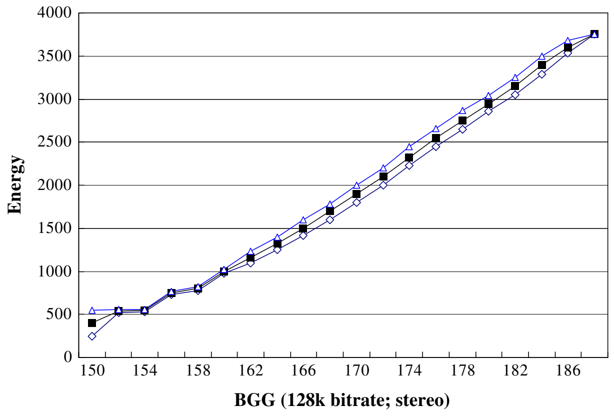


Fig. 4 The energy eb and its associated BGG. The average energy is marked by the *filled squares* notation, and the standard deviation is marked by *open triangles* and *open diamonds* notations

The energy-to-global-gain lookup table is a two-dimensional table with 14×38 entries, one dimension for bitrate (14 rows) and the other for global gain (38 columns). Each entry of the table stores the average energy associated with a particular bitrate and global gain. Therefore, given an energy value and the desired bitrate, a simple linear search suffices to find out the initial global gain efficiently. Note that building a lookup table from the training set is very time-consuming: it takes several days for each bitrate. Fortunately, this is done only once. At run time, the real-time quantization algorithm simply uses the lookup table to obtain the IGG.

To further explore the correlation between the IGG and the BGG, the relative difference between them is calculated for each granule. The results, given in Fig. 5, reveal that the IGG has a very high possibility of coinciding with the BGG. Furthermore, the IGG falls within ±4 of the BGG for over 99.98% of the granules.

The flowchart of the proposed three-iteration algorithm is shown in Fig. 6. The first iteration of the algorithm assigns global gain gg as IGG, obtained from the lookup table. Then, quantization and counting (counts the number of Huffman code bits for quantized spectral values) are performed. Depending on whether the space is enough (part23length ≥ the number of Huffman code bits), the second iteration either increases or decreases gg by 4. After that, the quantization and counting are re-executed to see if the bits are enough for the new gg. The algorithm once again adjusts gg based on the results of the second iteration. Then, the quantization and counting are performed the third time to determine the final global gain. Once the search algorithm is completed, the bitstream formatting procedure proceeds to complete the encoding of the current granule.

In the proposed algorithm, the search results are categorized into eight different *placements*, namely P1, P2,…, and P8 (see Fig. 6). Table 1 summarizes the maximum possible deviation between the obtained global gain and the BGG for each placement in our algorithm. The placements P3, P4, P5, and P6 guarantee a deviation of no more than one due to the arrangement of the algorithm, whereas the deviation in placements P2 and P7 is no more than three. The placements P1 and P8 have unknown deviations. By applying our algorithm with the training set, we have a plot for the histogram of the number of granules fall in each placement (see Fig. 7). The figure indicates that most of the granules fall within placements P3 to P6. Therefore, the vast majority of granules (99.98%) are encoded either with the BGG or the one next (larger) to the best.

Although virtually no granules in any sound tracks we have ever encountered fall in the placement P1 and P8, we still include them in the algorithm to handle the exceptional cases.

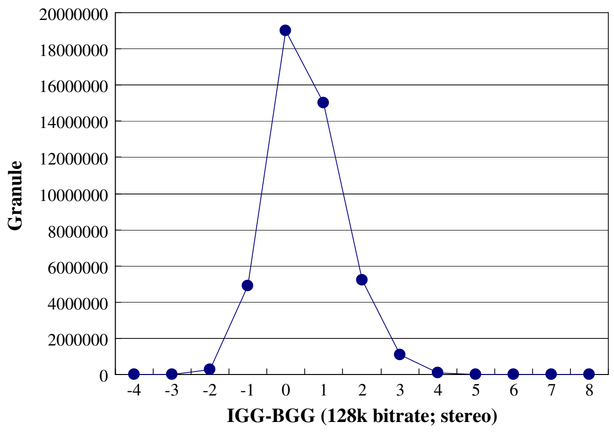


Fig. 5 The histogram of the differences between the IGGs obtained from a lookup table and the BGGs

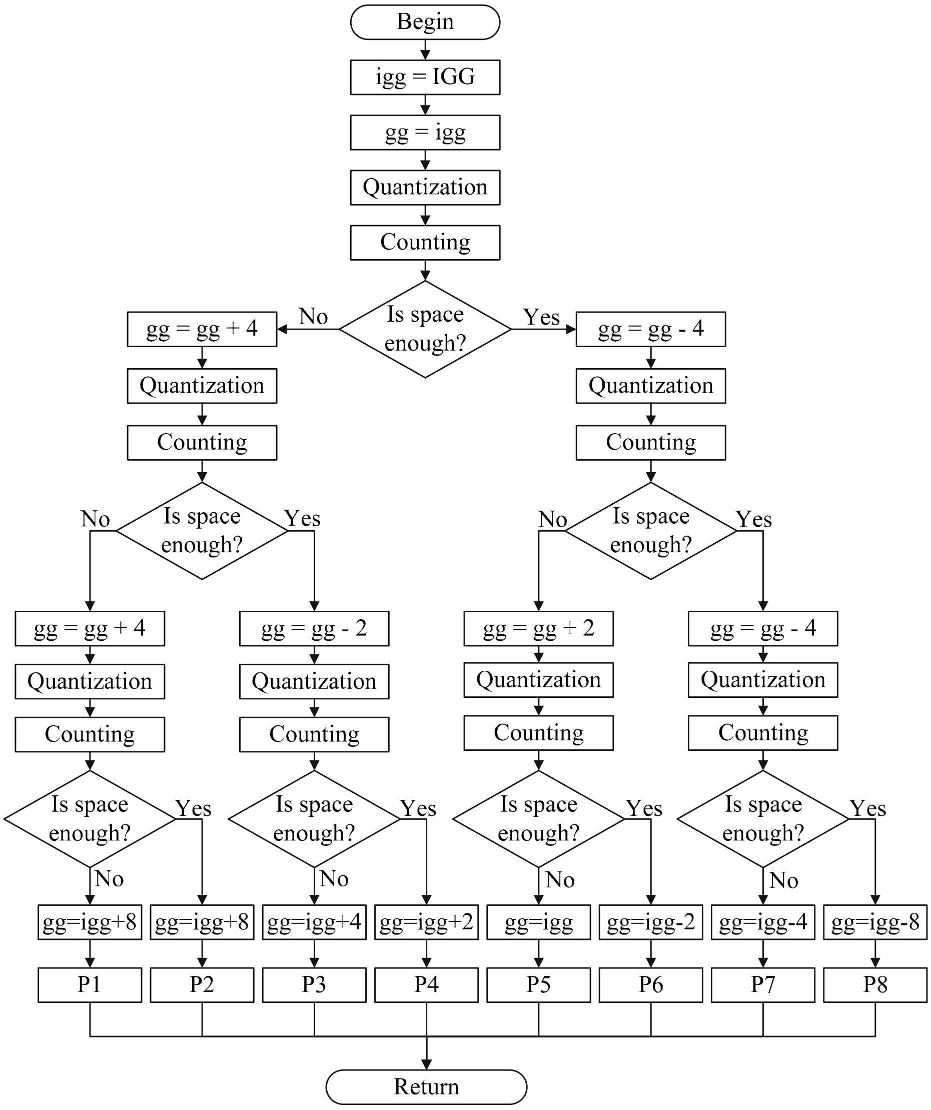
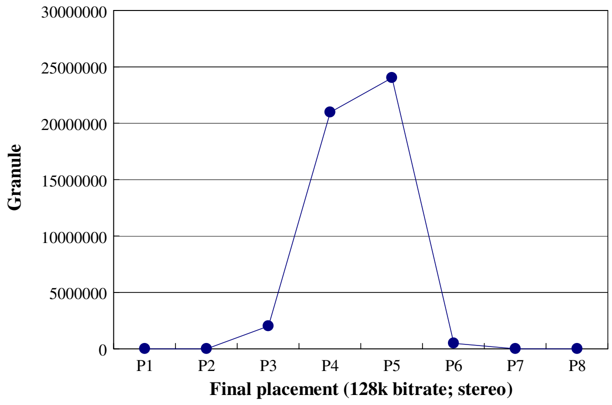


Fig. 6 The proposed quantization algorithm

For the placement P1, the available bits are not enough to encode all quantized spectral values. Thus, P1 requires a special treatment to discard excessive bits so that the iteration count of the algorithm remains the same.

Note that, theoretically, it requires at least two iterations for any algorithm to find the BGG. The reason is simple: even if the IGG is the BGG, performing a second iteration is necessary to make sure that encoding with a global gain of IGG-1 requires more bits than available. This is the only way for an algorithm to confirm that the IGG is indeed the BGG. Since the proposed algorithm uses only three iterations, its iteration number is already close to the lower bound.

Fig. 7 The histogram of the final placement



4 Performance evaluation

We implement a few reference encoders using the proposed quantization algorithm, and evaluate their encoding quality with several experiments. Based on the results of the experiments, we then port the final real-time encoder into a DSP platform. The reference encoders are described in Section 4.1, the test bed for quality assessment is discussed in Section 4.2, and the experimental results are reported in Section 4.3. In Section 4.4, we compare the proposed quantization algorithm with the approaches given in the literature.

4.1 Reference encoders

In order to assess the encoding quality of the proposed quantization algorithm, we need to compare it with a standard nested-loop quantization algorithm. After carefully studying existing encoder programs, we decide to use the LAME [8] encoder as the reference. The LAME encoder is publicly known as a high quality MP-3 encoder optimized for both encoding speed and quality. Thus, the encoding quality of the LAME encoder provides a good measure on the quality achievable by an MP-3 encoder.

We implement two encoders, BGG and GGG, using C language for quality assessment. Both encoders discard window switching and distortion loop. The BGG encoder always encodes each granule with its best global gain. The GGG (guess global gain) encoder, on the other hand, uses the proposed quantization algorithm to encode each granule. The energy-to-global-gain lookup table for the GGG encoder is obtained from the training set

Table 1 Maximum possible global gain error for each placement

Placement	Final global gain	Global gain error
P1	IGG +8	Unknown
P2	IGG +8	0…3
P3	IGG +4	0…1
P4	IGG +2	0…1
P5	IGG	0…1
P6	IGG −2	0…1
P7	IGG −4	0…3
P8	IGG −8	Unknown

mentioned in Section 3.2. We have carefully maintained the consistency between the BGG and the GGG encoder so that they are exactly the same, except for the quantization algorithm.

To improve encoding quality, we also realize the bit-reservoir and the M/S (middle/side) stereo coding for both the BGG and GGG encoder. For the bit reservoir, the standard provides the flexibility that several future frames can use the residual bits of the current one. However, without distortion control, we adopt a simpler algorithm: a frame can use only the bits left from the preceding one. With this restriction, the bit reservoir can be realized easily.

The proposed quantization algorithm also implements the M/S stereo coding. In the M/S coding, the middle and side channels are encoded, instead of the left and right channels. The middle and side channels are obtained by adding and subtracting the left and right channels. Correspondingly, the IGG should be obtained from the energy of the middle and side channels. The energy values in Eq. 2 can be used to determine whether to switch to the M/S mode at run-time. Specifically, our program computes the energy levels of eb_i, eb_r, eb_s, and eb_d for left, right, middle, and side channels, respectively, and then selects M/S mode if $(eb_s + eb_d) \leq (eb_r + eb_l)$.

We implement each encoder with three different arithmetic accuracies: (1) a 32-bit version, (2) a 16-bit version, and (3) a mixed 32- and 16-bit version. The mixed version has a 32-bit accuracy for subband analysis and MDCT operations, and the resultant spectral values are cast into 16-bit representations before quantization. For simplicity, we call the BGG encoder with 32-bit, 16-bit, and mixed accuracies as BGG32, BGG16, and BGG3216 encoders, respectively. The same naming convention also applies to the GGG encoders. In order to study the benefit of using M/S coding, we can turn off the M/S coding during experiments. We use "-MS" to denote an encoder without M/S coding. For example, the BGG3216-MS encoder is the BGG encoder with mixed accuracy and without M/S coding. Similarly, we can also turn off bit-reservoir. We use "-MSR" to denote an encoder without both M/S coding and bit reservoir.

The proposed three-iteration algorithm can be extended to four iterations with the price of higher computational complexity. The extra iteration can be used to make sure that almost all granules (placements P3–P6) are coded with their BGGs. However, our experimental results show that the encoding quality of the three-iteration algorithm is practically as good as that of the BGG encoder. Thus, no extension is necessary. The four-iteration algorithm in our implementation is used only for the MPEG-2 LSF (low sampling frequency) format [7]. With lower sampling rates, the LSF format has longer frame durations. Thus, the processor has sufficient time to execute the fourth iteration.

4.2 Test bed

Since the MP-3 is one of the perceptual audio coding schemes, it is inappropriate to use the S/N ratio for quality assessment. Instead, subjective (listening) experiments are usually conducted. However, subjective experiments rely heavily on experienced audiences, and are both time-consuming and difficult to implement. As a substitute, we use EAQUAL [2] (evaluation of audio quality), a PEAQ (perceptual evaluation of audio quality) [5] compliant program, for quality evaluation. The EAQUAL program returns the ODG (objective difference grade) after evaluating the difference between the reference and the test audio signals. The ODG has a high correlation with SDG (subjective difference grade), the ratings given by expert listeners. Therefore, the ODG may serve as an indication to the perceptible impairment by using the ITU-R five-grade impairment scale, as shown in Table 2.

Table 2 The ITU-R 5-grade impairment scale

SDG	Meaning
0	Imperceptible
−1	Perceptible but not annoying
−2	Slightly annoying
−3	Annoying
−4	Very annoying

Thirty sound tracks taken from thirty different stereo CD titles (not from the training set mentioned in Section 3.2) are used as the test items for the experiments. These test items are carefully chosen to contain diverse types of music, including classical music, pop music, soft music, solo, and chorus. Each item is at least 100 s in length so that the EAQUAL program can provide reliable ODG. For the GGG encoder, the energy-to-global-gain lookup table used to obtain the IGGs is pre-determined; it is not derived from the test items used for quality evaluation.

To represent the encoding quality of an encoder, we use the average ODG of the test items. We use an encoder to encode one test item t with a specified bitrate, and then decode it into t' (we use the LAME decoder, a derivative of mpg123 [9], as the reference decoder). The inputs to EAQUAL are the reference item t and the test item t'. After evaluation, EAQUAL returns an ODG rating for t'. Then, the average ODG over the thirty test items is used as an indication of the performance of the encoder. Since there are 14 different bitrates in MPEG-1, we use a curve to represent the encoding quality of the encoder over all bitrates.

We use a PC (personal computer) to conduct the experiments. The purpose of the experiments is to assess the encoding quality of different quantization algorithms, not the speed. Therefore, a PC is suitable and convenient for the job. We will report the complexity and the speed of the proposed algorithm in Section 5, and show that the encoding quality of the DSP version is exactly the same as the PC version. The most important reason that a PC is used is that it is computationally much more powerful than the DSP core that we choose. Therefore, the experiments can be conducted much more efficiently.

4.3 Experiments and results

We conduct four experiments to assess the encoding quality of the proposed quantization algorithm. The first experiment evaluates the encoding quality of the BGG encoder to study the influence due to the lack of the psychoacoustic model. The second experiment compares the encoding quality between the proposed quantization algorithm (GGG encoder) and the best global gain algorithm (BGG encoder). The third experiment studies whether the bit reservoir actually improves the encoding quality, and, if so, to what extent. The last experiment investigates the influence of signal level on the encoding quality. During the experiments, we use only MPEG-1 format to simplify the comparison although our encoders also support MPEG-2 LSF format.

The first experiment compares the encoding quality of the LAME encoder with that of the BGG encoder possessing different arithmetic accuracies. To obtain the highest possible encoding quality, both the M/S mode and bit reservoir are in use for the BGG encoder. The results are given in Fig. 8, where each curve indicates the average ODG received by each encoder over different bitrates. For example, the average ODG of the BGG32 encoder at 160k bitrate is around −0.7, which means that on average the encoding quality of the BGG32 encoder at 160k bitrate is a little better than “Perceptible but not annoying.” From Fig. 8,

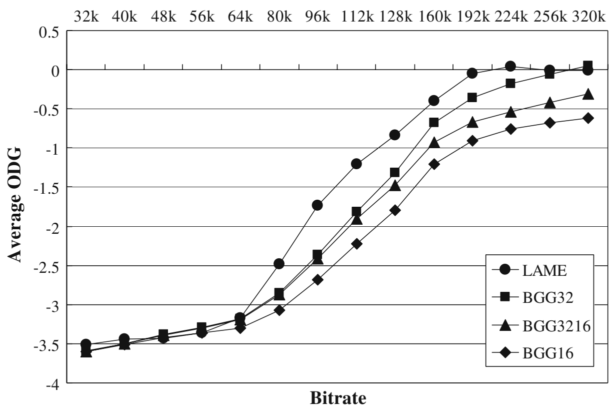


Fig. 8 The average ODG of the LAME and the BGG encoders with different bitrates. The BGG32, BGG3216, and BGG16 are the BGG encoders with 32-bit, mixed, and 16-bit arithmetic accuracies, respectively

the ODG degradation due to the lack of psychoacoustic model is apparent, but acceptable. The BGG32 encoder with a bitrate of 128 kbps has the encoding quality comparable to that of the LAME encoder with 112 kbps. The results also confirm that the arithmetic accuracy greatly affects the ODG, because the BGG32 encoder has a better encoding quality than the BGG16 encoder. The reason, justified in the fourth experiment, will be given later.

In the second experiment, we compare the encoding quality between the GGG and BGG encoder. The results, shown in Fig. 9, show that the encoding quality of GGG3216 is slightly inferior to that of BGG3216, either with or without M/S coding. However, the ODG difference is less than 0.1 for all bitrates. Thus, the encoding quality of the GGG encoder is practically as good as that of the BGG encoder. As expected, when M/S coding is used, the encoding quality is better than that of normal stereo coding. This is because the two channels of all test items are highly correlated. Note that, though we use the mixed version of the encoders to present the results, the same observations also hold for the other arithmetic accuracies.

In the third experiment, we investigate the quality improvement due to the use of bit reservoir (see Fig. 10). Without bit reservoir, the ODG difference between the BGG3216-MSR and GGG3216-MSR encoder becomes quite obvious. This phenomenon is not difficult to explain. Recall that the global gain obtained by the proposed algorithm may not be exactly the best. Therefore, a granule encoded by the GGG encoder may have more unused bits in comparison to the BGG encoder. Let x and y be the number of unused bits of

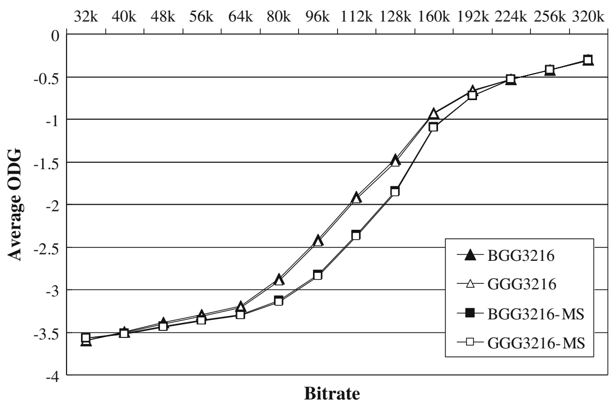
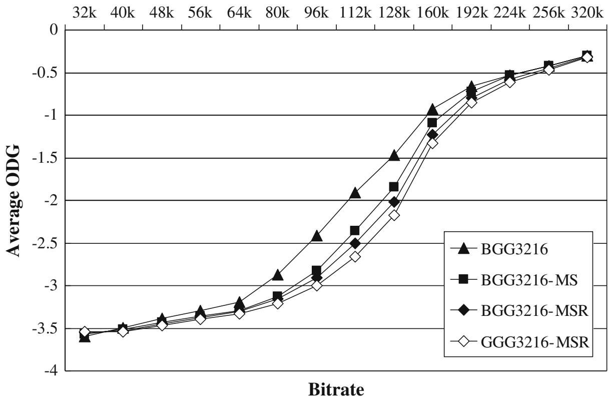


Fig. 9 The comparison of BGG3216 and GGG3216. The "*-MS*" denotes an encoder without using M/S coding. The curves of the GGG3216 and BGG3216-MS encoders almost coincide with those of the BGG3216 and BGG3216-MS encoders, respectively

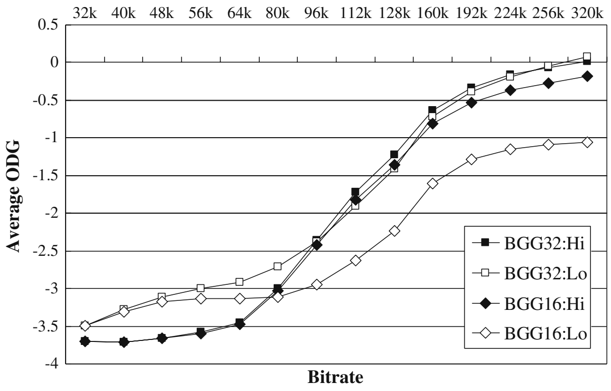
Fig. 10 The effect of using bit-reservoir. The "*-MSR*" denotes an encoder without using M/S coding nor bit reservoir



the GGG and BGG encoder, respectively. Then, $x \geq y$. When bit-reservoir is disabled, the unused bits are wasted and the GGG encoder wastes $x–y$ bits more than the BGG encoder. Thus, the quality difference becomes obvious. On the other hand, when bit-reservoir is enabled, the $x–y$ bits will be retained in the next granule for the GGG encoder. The additional $x–y$ bits provide the GGG encoder the possibility of encoding the next granule with less distortion. So, on average, the performance difference between these two encoders becomes very minor. Therefore, the bit reservoir must be used for the proposed algorithm to work best.

As we noted in the first experiment, arithmetic accuracy greatly affects the encoding quality. To investigate the reasons, we divide the thirty test items into two groups, namely *high-level group* and *low-level group*. The high-level group contains 15 test items with higher signal levels among the thirty test items, whereas the low-level group contains the rest of the items. The average ODG of the BGG32 and BGG16 encoders for both the high-level and low-level groups are shown in Fig. 11. The results clearly show that the signal level strongly affects the encoding quality of the 16-bit encoder. The high-level group has a much higher average ODG than the low-level group. On the other hand, the signal level only slightly affects the encoding quality of the BGG32 encoder. This is because, with extremely low input signal levels, mathematical errors of one or two bits lead to a significant degradation of S/N ratio. Thus, the weakness of 16-bit fixed-point processors becomes evident when the signal level is low. Since our 16-bit encoder already has an average accuracy of around 14.5 bits, there is little room for any further improvement. This

Fig. 11 The encoding quality of high-level group versus that of low-level group. The "*:Hi*" and "*:Lo*" denote the average ODG of the high-level group and low-level group performed by an encoder, respectively



experiment suggests that using automatic gain control (AGC) is beneficial for a 16-bit encoder to encode music with low signal-level.

From the previous experiments, we obtain the following conclusions. Firstly, the encoding quality of the BGG encoder is acceptable. Secondly, the encoding quality of the GGG encoder is practically as good as that of the BGG encoder. And finally, when the signal level is high, the encoding quality of GGG16 is as good as that of GGG32. Therefore, it is feasible to implement GGG16 in a DSP.

4.4 Comparison with existing methods

Many efficient algorithms for quantization (also referred to as bit allocation in the literature) have been reported [16, 10, 17, 15]. It is computationally expensive to perform a single iteration of quantization and counting. Therefore, all of these algorithms attempt to reduce the overall number of iterations. Most algorithms use the global gain obtained in the previous granule as the IGG of the current granule. Furthermore, adaptive step sizes are used to speedup the search of global gain.

Table 3 summarizes the number of iterations required for different approaches. Note that the average number of iterations is not suitable to indicate the efficiency of an algorithm when it is implemented inside a real-time encoder with the architecture similar to Fig. 1. The efficiency of an algorithm should be evaluated by the maximum number of iterations. That is because the maximum number of iterations affects the peak computational demand, which in turn determines the clock rate and power consumption of the processor. One might imagine that the average number of iterations may be close to the maximum. However, this is not true. For example, the LAME encoder has a large fluctuation on the number of iterations: the average is 8.7 iterations and the peak is 66 iterations in the inner loop for encoding the test items (sound tracks) discussed in Section 4.2.

Yen et al. [17] report an algorithm that performs quantization with an average of 1.8 iterations and a maximum of eight iterations. In addition, advanced acceleration techniques such as bandwidth control are also reported. The algorithm is implemented in a DSP with an impressive speed. However, it is possible to further improve the speed of quantization based on the three-iteration algorithm proposed in this paper. Wang et al. [15] also report the implementation of a real-time encoder based on a DSP. With the use of a modified psychoacoustic model, they describe an algorithm to adjust the scalefactors for each scalefactor band. The average and maximum iterations for the outer loop are two and six, respectively. However, the number of iterations in the inner loop is not reported. In contrast, the proposed algorithm offers an obvious speed advantage.

Table 3 The number of iterations (per granule) for various quantization algorithms

Method	Proposed algorithm	LAME [8]	Oh et al. [10]	Yen et al. [17]	Wang et al. [15]	Yang and Chen [16]
Inner loop	3 (fixed)	8.7 (avg) 66 (max)	2.1 (avg) 3 (max)	1.8 (avg) 8 (max)	Not reported	2.52~4.18 (avg)
Outer loop	1	1.98 (avg) 2 (max)	1	1	<2 (avg) 5×6 (max, per scalefactor band)	Not reported[a]
Overall	3	17.4 (avg) 66 (max)	2.1 (avg) 3 (max)	1.8 (avg) 8 (max)	Unknown	Unknown[a]

[a] Yang and Chen use the standard nested-loop algorithm

Yang and Chen [16] report a dynamic search algorithm for the inner loop. The algorithm always obtains BGGs. It outperforms binary search and reduces the average number of iterations to 2.52~4.18. However, the algorithm is not designed for real-time encoders and the maximum number of iterations is unreported (at least $\lceil 4.18 \rceil = 5$). Therefore, the proposed algorithm with a maximum of three-iterations is preferred when real-time encoding is desired. Oh et al. [10] report an algorithm with a maximum of three iterations. The algorithm is also implemented in a DSP. However, the algorithm is not clearly explained and the encoding quality is vaguely described. It is unknown whether the obtained global gain is close to the BGG or not. In this regard, the proposed algorithm, with the same number of iterations, gives an encoding quality that is practically as good as BGG.

Since all real-time quantization algorithms attempt to discover or approximate the BGG efficiently, the encoding quality of the BGG encoder is a standard benchmark for real-time encoders. The strength of the proposed algorithm is that it uses only three-iterations to achieve the encoding quality of the BGG encoder. Furthermore, the proposed algorithm works well in conjunction with the M/S coding and bit reservoir mechanisms. Our experimental results show that these two mechanisms have great influences on the overall encoding quality. In contrast, the encoding quality of the algorithms proposed by Yen et al. [17] and Oh et al. [10] are unknown in comparison to that of the BGG encoder.

There is a tradeoff between encoding quality and speed. To compare real-time quantization algorithms impartially, in addition to justifying the maximum number of iterations, the encoding quality must also be considered. Unfortunately, we found that it is difficult to compare the encoding quality of real-time encoders based on the experimental results reported in the literature. For example, Yen et al. [17] and Oh et al. [10] evaluate the quality of their encoders by human listeners. It is then difficult to reproduce exactly the same results. Moreover, they do not offer a complete picture of how the algorithm performs over different bitrates. In addressing these issues, we use EAQUAL [2] instead of human listeners. We hope that the ODG reported hereinbefore gives a clear indication of the experimental results so that other researchers can easily compare our results with theirs in the future.

5 Implementation for DSP

There are two classes of DSPs available in the market: fixed-point processors and floating-point processors [14]. A fixed-point processor contains hardware to perform only integer multiplication operations, whereas a floating-point counterpart has hardware for floating-point arithmetic operations. In terms of cost, a floating-point processor is usually more expensive because it requires much more complicated hardware. Thus, low-cost MP-3 encoders and decoders are mostly implemented using fixed-point processors.

To verify our algorithm, we implement the real-time MP-3 encoder with the TI's C5x 16-bit DSP [13]. The GGG16 encoder is ported to the DSP platform. To speedup the encoder, time-critical procedures are rewritten in assembly language. Overall, about 20% of code are hand-written assembly code, including subband analysis, MDCT, energy calculation, and inner quantization loop. In terms of performance, when stereo CD music is encoded at 128 kbps, the real-time encoding is achieved with 35 MIPS of computation with M/S coding and bit reservoir. No more than 42 MIPS of computation is required when the bitrate is at 192 kbps. Since 35 MIPS has reached the commercial standard for the C5x DSP (see [17] for a list of MIPS required by various encoders), we did not push the speed of the encoder further, as we are mainly trying to show that the proposed quantization algorithm is suitable for real-time encoding with DSPs.

During the porting process, whenever a procedure in C code is re-written as assembly code, we have been very careful to make sure that the assembly code produces exactly the same results as its original C code. To fully explore the hardware capability offered by the DSP (e.g., multiply and add in one instruction), there are also cases that we have to rewrite the C code so that it correctly emulates the hardware behavior. The final result is that the DSP encoder produces exactly the same MP-3 output stream as the GGG16 encoder. Therefore, the encoding quality of the DSP encoder is exactly the same as that of the GGG16 encoder reported in Section 4.3.

The last issue remains to be addressed is that “is it possible to implement the LAME encoder in a low-cost DSP?” After all, the LAME encoder is significantly better than the BGG encoder for bitrates ranging from 80 to 128 kbps (Fig. 8). The LAME encoder is equipped with a sophisticated psychoacoustic model, which requires 90 MIPS [10] of computation. In addition, up to 66 quantization iterations (Table 3) are required, which may cost another 330 MIPS of computation. The estimation is based on the measurement that our DSP implementation of the quantization routine consumes 5 MIPS per iteration (per second). Therefore, if the LAME encoder is ported to a 16-bit DSP platform, at least 420 MIPS of peak computation will be required to perform real-time encoding. This is far beyond the current commercial standard, which is around 40 MIPS [17]. Thus, most real-time encoders employ dramatically simplified quantization algorithms.

6 Conclusion

In this paper, we give an overview of the MP-3 encoding flow, and explain that the nested-loop quantization algorithm suggested by the ISO’s standard is not suitable for a real-time encoder based on a low-cost DSP. We then present an efficient (three-iteration) quantization algorithm for such an application. The experimental results show that our encoder with the proposed quantization algorithm has acceptable quality degradation in comparison with a highly optimized encoder. The advantage of our quantization algorithm, when compared with other algorithms in the literature, is that its iteration number is fixed to three without variation. Based on the experimental results and cost concerns, we implement our algorithm on a 16-bit DSP, and it is able to encode stereo music at 128 kbps with 35 MIPS.

References

1. Chang F-M, You SD (2004) Using only long windows in MPEG-2/4 AAC encoding. Lecture Notes in Computer Science LNCS 3333, pp. 151–158
2. EAQUAL (1999) http://www.mp3-tech.org/programmer/sources/eaqual.tgz
3. EBU (2005) Digital Video Broadcasting (DVB); Implementation guidelines for the use of MPEG-2 systems, video and audio in satellite, cable and terrestrial broadcasting applications, ETSI TR 101–154
4. EBU (2006) Radio broadcasting systems; digital audio broadcasting (DAB) to mobile, portable, and fixed receivers, ETSI EN 300 401 v1.4.1
5. International Telecommunication Union, ITU-R Rec. BS-1387, Method for objective measurements of perceived audio quality
6. ISO/IEC (1993) Information technology—coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s—part 3: audio, IS 11172–3
7. ISO/IEC (1998) Information technology—generic coding of moving pictures and associated audio information—part 3: audio, IS 13818-3, 2nd Ed
8. LAME, Version 3.97b2, http://lame.sourceforge.net/
9. MPG123, http://www.mpg123.org/

10. Oh HO, Kim JS, Song CJ, Park YC, Youn DH (2001) Low power MPEG/Audio encoders using simplified psychoacoustics model and fast bit allocation. IEEE Trans Consum Electron 47(3):613–621
11. Painter T, Spanias A (2000) Perceptual coding of digital audio. Proc IEEE 88(4):451–513
12. Princen JP, Johnson AW, Bradley AB (1987) Subband transform coding using filter bank designs based on time domain aliasing cancellation. Proc IEEE ICASSP, Dallas, TX, USA, 12:2161–2164
13. Texas Instruments (1993) TMS320C5x user's guide, 2547301-9721 revision D
14. Venkataramani B, Bhaskar M (2002) Digital signal processors: architectures, programming and applications. McGraw-Hill, Englewood Cliff, NJ
15. Wang X, Dou W, Hou Z (2002) An improved audio encoding architecture based on 16-bit fixed-point DSP. IEEE 2002 International Conference on Communications, Circuits and Systems and West Sino Expositions 2:918–921
16. Yang CK, Chen SG (2003) New static and dynamic search algorithms for fast MP3 bit allocations. Proc Int Conf Multimedia Expo 1:I-77–I-80
17. Yen C-H, Lin Y-S, Wu B-F (2007) An efficient implementation of a low-complexity MP3 algorithm with stream cipher. Multimedia Tools and Applications, online published, 25(3):335–355 (June)
18. Yu C-H, You SD (2002) On the possibility of only using long windows in MPEG-2 AAC Coding. Lecture Notes in Computer Science LNCS 2532:663–670

Shingchern D. You received the Ph.D. degree in Electrical Engineering from the University of California, Davis, CA, USA in 1993. Dr. You's research interests include audio signal processing and recognition, applied digital signal processing to communication systems, and circuit design.

Woei-Kae Chen received the diploma in Electronic Engineering from National Taipei Institute of Technology, Republic of China, in 1984, the M.S. and Ph.D. degrees in Computer Engineering from North Carolina State University, NC, USA in 1988 and 1991. Dr. Chen's research interests include software engineering, distributed computing, and graph algorithms.