Multimedia Systems (2009) 15:101–112
DOI 10.1007/s00530-009-0152-6

REGULAR PAPER

A joint encoder–decoder framework for supporting energy efficient audio decoding

Wendong Huang · Ye Wang

Received: 14 November 2007 / Accepted: 10 February 2009 / Published online: 12 March 2009

Abstract In comparison with the relatively slow progress of battery technology, semiconductor memory has improved much more rapidly, making storage a less critical limiting factor in designing low power embedded systems such as PDAs. To exploit such technology trends, we present a novel framework, a joint encoder–decoder framework (JEDF), which allows the decoder to tradeoff energy and memory consumption without sacrificing playback quality. We employ sum-of-powers-of-two (SOPOT) technique, an approximate signal processing (ASP) technique, in an MPEG AAC decoder to reduce the computational workload. The SOPOT introduces additional ASP noise (in the decoder) on top of the quantization noise introduced in the process of lossy compression (in the encoder). The sum of these two kinds of noise may become audible when it exceeds the masking threshold. We tackle this problem from a new perspective: the proposed JEDF allows the ASP and quantization noises to be shaped jointly to match the masking threshold. In the case that the perceptual room between the masking threshold and the quantization noise is insufficient for the ASP noise, the JEDF can reduce the quantization noise level which results in an increase in bitrate. To implement the proposed scheme, we have developed two new techniques: (1) SOPOT truncation noise shaping; (2) truncation noise allocation based on a perceptual model. Experimental results show the effectiveness of our approach.

Communicated by Cormac Sreenan.

W. Huang · Y. Wang (✉)
School of Computing, National University of Singapore, Singapore, Singapore
e-mail: wangye@comp.nus.edu.sg

Keywords Energy efficiency · Workload reduction · Joint encoder-decoder framework (JEDF) · Joint ASP and quantization noise shaping · SOPOT

1 Introduction

Energy efficiency is a critical design consideration for battery-powered mobile devices, such as mobile phones, PDAs and audio/video players, due to their limited battery capacity. With the rapid growth of multimedia processing applications being executed on these platforms, energy efficiency methods optimized for these applications are becoming increasingly important.

Among the techniques to reduce energy consumption of multimedia decoding applications, a fundamental approach is to reduce their computational workload. The reduced workload can be exploited by a voltage/frequency scalable processor to save energy and to prolong the battery life. Towards this, approximate signal processing (ASP) techniques have been widely adopted [15], which exploit algorithms with flexible structures, such as tunable word length, filter order, etc., to achieve the desired tradeoff between accuracy of their results and their utilizations of the resources. A well-known technique in ASP is partial spectrum reconstruction (PSR), which only reconstructs the spectrum of a part of the coded signals, resulting in a low pass version of the original spatial samples [1,5,14]. Essentially, ASP techniques may result in a degradation of playback quality in exchange for a prolonged battery life [2].

However, if the user requires both CD-quality audio and long battery life, it is difficult to solve the dilemma with existing methods. To address this problem, we propose in this paper a new approach towards saving energy. We achieve energy efficient audio decoding by a joint encoder–decoder

framework (JEDF). This approach allows us to introduce a less critical limiting factor, the storage, into the tradeoff, so that we can significantly reduce the decoding workload, while maintaining transparent playback quality by a possible sacrifice of the compression efficiency.

In our JEDF framework, the decoder employs ASP techniques to reduce the computational workload, which results in additional ASP noise. The saved computational workload is determined by the configuration of the ASP structure, where more workload is reduced at the expense of introducing more ASP noise. In our scheme, the configurations of the ASP structures of the decoder are specified in the encoder. The encoder inserts additional side information, which describes the desired configurations of the ASP structure, into the compressed bitstream. In the process of playback, the decoder reads the related side information and adopts the specified configurations for the ASP algorithm accordingly. In other words, the encoder can accurately control the ASP computational workload at the decoder.

To guarantee the playback quality, we extend the current audio coding techniques by jointly shaping the ASP noise and the quantization noise according to the masking threshold. Masking threshold is a fundamental concept in modern perceptual audio codecs, such as MPEG-1 Audio Layer III (MP3) [7] and Advanced Audio Coding (AAC) [8]. As a property of the human auditory system, the masking threshold indicates that noise lower than this threshold is inaudible. Exploiting this principle, existing perceptual audio encoders compute maximal allowed quantization step sizes subject to masking threshold constraints, which will produce the minimal bit length of coded audio signals. We define Masking-to-Quantization-noise-Difference (MQD) as the difference between the masking threshold and the quantization noise. MQD indicates the maximum level of extra noise allowed by the masking threshold, which will not degrade the playback quality.

The key idea of our proposed work, as shown in Fig. 1, is to reduce the quantization noise level in the encoder when necessary, making the increased MQD to accommodate the ASP noise introduced by the energy efficient decoder. This approach ensures that the overall distortion is below the masking threshold. In comparison with conventional encoders, this approach may increase the size of the compressed audio files. Its effects can be analyzed from two perspectives. First, the increased file size will lead to additional energy consumption in the data reading process. However, reading operations of an audio file are only responsible for a small fraction of the overall energy consumption in an audio decoder. The storage subsystems of mobile devices are typically made of Flash ROM, which has lower energy consumption than main memory which is usually made of Dynamic RAM (DRAM) [22]. The reading process accesses the compressed audio file only once, in comparison with the computation process, where

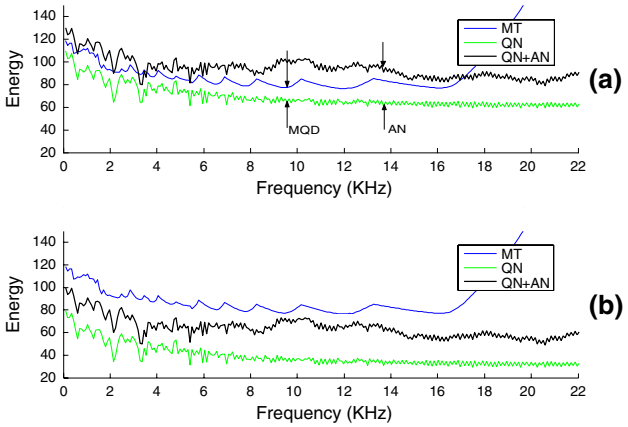

Fig. 1 Illustration of the proposed scheme, where MT, QN, AN, and MQD (the latter two are identified by arrows) stand for masking threshold, quantization noise, ASP noise, and masking-to-quantization-noise-difference, respectively: **a** for a conventional AAC encoder: the sum of the additional ASP noise and the quantization noise exceeds the masking threshold; **b** for our scheme: with reduced quantization noise, the overall noise is below the masking threshold

the decompressed data is accessed by a series of processing modules. In consideration of above factors, the small overhead in the reading operation can be easily offset by the significant savings achieved by the proposed scheme. This is also well demonstrated by [19]: on a mobile device, the file reading operation of a software MP3 decoder only consumes 1.9% of the energy of its decoding process. Our proposed scheme will lead to less than 10% increase in file sizes for 128 Kb/s AAC bitstream and above. This suggests that the additional energy consumption from file size increase is negligible. The second concern is on the storage capacity of mobile devices: if a mobile device has only limited storage capacity, then we would prefer storage efficient schemes. Fortunately, the rapid advance of the semiconductor technologies has made the storage a less critical limiting factor in mobile devices. For instance, the well-known Apple iPod nano series have already been equipped with 2, 4 and 8G bytes of storage. Large storage capacities allow us to exploit flexible design strategies in low power techniques for mobile devices. Our scheme offers an appealing tradeoff between the local storage and energy consumption. The projected applications of our scheme is download-playback services where the user downloads audio clips once and play them back multiple times from local storage. In such an application scenario, we argue that the audio file size is less critical than battery life.

It should be noted that the proposed scheme differs fundamentally from existing energy-efficient techniques, where the decoder is solely responsible for energy savings. Its superiority to the existing approaches is twofold. First, the encoder is responsible for computing the desired configurations of the ASP structures, which alleviates the computation at the decoder. Second, the encoder can access more information to

achieve the goal than the decoder. For example, our scheme accesses the masking threshold in the encoder to shape the ASP noise, which is impossible for a decoder-only approach.

Finally it is worth pointing out that even though the proposed scheme is a joint encoder–decoder framework, the generated bitstream is fully backward compatible to standard AAC decoders with slightly improved playback quality due to the slightly higher bitrate. This is a desirable feature to facilitate its applications.

The rest of the paper is organized as follows. In Sect. 2, we briefly review the noise-shaping technique in AAC and computation-efficient techniques for transforms. Then we provide an overview of our work in Sect. 3. In Sect. 4, we present a detailed description of the technology of joint ASP and quantization noise shaping for AAC encoding. In Sect. 5, we present the experimental results. Finally, we conclude the paper in Sect. 6.

2 Related works

2.1 Noise-shaping techniques in AAC

In AAC [8], the full spectrum of a frame is partitioned into 49 scale factor bands. Different scale factor bands may have different masking thresholds for two reasons: (1) different sensitivities of the human auditory system over these scale factor bands, and (2) the characteristics of the audio signal. To fully exploit the variations of masking thresholds, different quantization step sizes are used to quantize the frequency coefficients with the goal of keeping the quantization noise below the masking threshold. This results in different quantization noises for different scale factor bands. On the other hand, the transparent playback quality is guaranteed if for each scale factor band, the following relationship holds:

$$E_{Q(i)} = \sum_{k \in \beta(i)} (F(k) - Q(F(k)))^2 \leq E_{T(i)}, \quad 0 \leq i < 49 \tag{2.1}$$

where $\beta(i)$ is the range of frequency coefficients for scale factor band i, $F(k)$ and $Q(F(k))$ are the original kth frequency coefficient and its quantized value, $E_{Q(i)}$ and $E_{T(i)}$ are the quantization noise and the masking threshold for scale factor band i, respectively.

The quantization process can be expressed as [8]:

$$Q(F(k)) = \text{int}\left(\frac{F(k)^{\frac{3}{4}}}{\Delta_Q} + 0.4054\right) \tag{2.2}$$

where Δ_Q is the quantization step size. As shown in (2.2), the quantization noise $E_{Q(k)}$ increases as the quantization step size Δ_Q becomes larger.

2.2 Computation efficient techniques for transforms

In the literature, typical computation-efficient algorithms for transforms, including (Inverse) Fast Fourier Transform ((I)FFT), (Inverse) Discrete Cosine Transform ((I)DCT) and (Inverse) Modified Discrete Cosine Transform ((I)MDCT), can be divided into two classes: data driven approaches and fixed point approximation approaches. Data driven approaches include pruning techniques [6,12,21] and forward mapping IMDCT [13]. This class of approaches eliminates the calculation for zero-valued coefficients since these coefficients make no contribution to the output of the transform. The workload reduction of data-driven approaches largely depend on the statistical properties of the input sequence and block length of the transform. An important property of this class of approach is that the efficiency of workload reduction degrades rapidly as the size of the transform block grows.

Fixed point approximation approaches include fixed point multiplication methods [4] and Sum-Of-Powers-Of-Two (SOPOT) methods [3,11]. In transforms including IMDCT, a large part of computation involves floating point multiplications. As mobile devices are rarely equipped with the floating point unit, these floating point multiplications can be simulated by software packages. As a result, the required workload increases significantly. To achieve computation-efficient implementations, floating point computations are widely replaced by fixed point approximation approaches in mobile devices. Fixed point multiplication scales and approximates those floating point coefficients by integers. The resulting transforms have much lower computational workload than the floating point version and therefore are widely employed in various applications. However, it is argued that the 32-bit integer cannot provide sufficient accuracy for long block transforms [3]. Moreover, it cannot provide tunable tradeoffs between computational workload and ASP noise. As an alternative, SOPOT methods decompose the operation of multiplication into the sum of powers of two operations. For example, we can calculate $x \cdot 0.40625$ as $x \cdot 2^{-1} - x \cdot 2^{-3} + x \cdot 2^{-5}$. For SOPOT, an effective way to save the computational workload is to reduce the number of SOPOT terms, which results in the truncation noise: more workload can be saved at the cost of introducing more truncation noise.

3 Overview of the proposed work

We have implemented our scheme with the ISO/IEC 13818-7 Advanced Audio Coding format [8] for the sake of proof of concept. In [8], three profiles are provided: main profile, low complexity profile, and scalable sample rate profile. Among them, the low-complexity profile has found the most widespread use. Therefore, we have implemented our scheme in low-complexity profile to broaden its applicability.

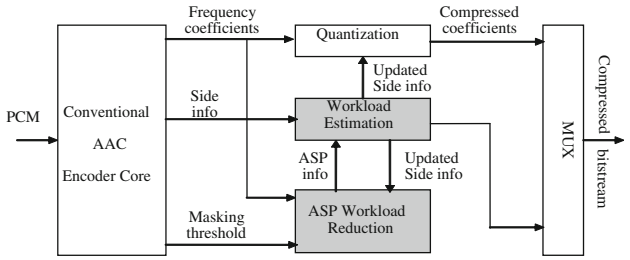

Fig. 2 Architecture of the proposed audio encoder

The block diagram of the proposed audio encoder is depicted in Fig. 2. Our scheme is essentially a two-pass encoder based on the frame structure of AAC. The first pass is implemented by a conventional AAC encoder core, which analyzes the PCM data of the current frame subject to the bit rate constraint, and provides three kinds of information to support the second pass processing: masking thresholds for all scale factor bands, frequency coefficients, and their associated side information, including the quantization step sizes.

Analogous to the bit rate constraint over a conventional AAC encoder, we introduce a computational workload level of the ASP algorithm as the constraint for the second pass. The second pass of the proposed scheme searches desired ASP configurations such that both the MQD requirement and the computational workload requirement are met. This pass involves two important modules: ASP workload reduction and workload estimation. The workload estimation module controls the processing of the second pass. Based on the ASP parameters provided by the ASP workload reduction, the workload estimation module derives the corresponding workload of decoding the current frame. If the workload is lower than the workload constraint, the workload estimation module will invoke the quantization and multiplier (MUX) modules, etc., with the updated side information. The involved modules compress the frequency coefficients of the current frame and multiplex the compressed data and side information into the coded bitstream. If the workload is higher than the workload constraint, the workload estimation module will reduce the quantization step size of some scale factor bands to yield an increased MQD level. With the updated side information, the workload estimation module invokes the ASP workload reduction module. This process repeats until the actual workload is below the workload constraint or the required quantization step sizes cannot be supported by the AAC specifications.

The ASP workload reduction module is responsible for reducing the computational workload of the decoding process. Among various processing modules in an AAC decoder of low complexity profile, we focus on the IMDCT due to two factors: (1) it is challenging to design an effective ASP method for a long block of IMDCT (AAC employs a 2048-point IMDCT): as shown in Sect. 2.2, for existing techniques, either the efficiency of workload reduction is limited or the ASP noise is not acceptable; (2) IMDCT is responsible for a major part of the computational workload of the whole decoding process, especially with high accuracy computation. The latter will be shown in Sect. 5.1. In most AAC decoders, the 2048-point IMDCT is computed using a 512-point IFFT and some pre- and post-rotations. In this method, IFFT is responsible for about 50% of the workload of the IMDCT. As the first step, we concentrate on IFFT to achieve workload reduction. Based on the analysis in Sect. 2.2, we have chosen SOPOT as the implementation of IFFT at decoder, since it provides: (1) high dynamic range of computation accuracy; (2) tunable tradeoffs between computational workload and truncation noise. These two properties suggest that it is appropriate for noise shaping in our scheme. In addition, it only requires two simple arithmetic operations, shift and addition, which are found in a wide range of mobile devices. To perform workload reduction, we have developed joint ASP and quantization noise shaping for AAC encoding. This method comprises two parts. The first part includes the techniques to shape the SOPOT truncation noise to fit the noise-shaping framework used by an AAC encoder. The second part concentrates on how to allocate MQD to its associated SOPOT coefficients to effectively reduce the computational workload. These two parts are presented in Sects. 4.1 and 4.2, respectively.

4 Joint ASP and quantization noise shaping

4.1 Truncation noise shaping of SOPOT coefficients

As discussed above, truncated SOPOT coefficients cause additional noise in the reconstructed data. To guarantee the transparent playback quality, we need to keep these truncation noises below MQD. These MQD levels vary dynamically as we may change the quantization step sizes of some scale factor bands. Towards this, we shape the truncation noise of the SOPOT coefficients to match the level of MQD. In our work, the SOPOT coefficient truncation noise shaping has two implications. First, we need to truncate different coefficients at different positions. This requirement enables us to fully exploit the variations of MQDs: for larger MQDs, we can discard more SOPOT terms to save computations. Second, the truncation noise over a certain frequency coefficient should be orthogonal to other frequency coefficients. We call this property the orthogonality of the truncation noise. The orthogonality of the truncation noise can be explained as follows. When we perform IFFT on a frequency coefficient with the truncated SOPOT coefficients, time domain noises are produced over the reconstructed data. The orthogonality of the truncation noise requires that the spectrum of those time domain noises is only related to the

source frequency coefficient, no other frequency coefficients are involved. The orthogonality of the truncation noise eliminates the cross scale factor band noise, which will complicate the truncation noise shaping process.

The truncation of SOPOT coefficients appears to be equivalent to coefficient quantization problems of various transforms in the literature, such as [4,9,16], which have attracted the attention of many researchers. However, these results cannot be applied to our scheme for the following reasons. First, all of them have been developed to address the issue of finite word length of registers and these schemes have modeled the truncation errors as independent and identical distribution random variables. This implies that their results are not valid for the shaped truncation noises in our scheme. Second, in those works, the truncation noise resulting from a frequency coefficient will be spread to other frequency coefficients, which is not desirable for accurate noise control.

To shape the truncation noise, in Sect. 4.1.1, we propose a method to achieve the orthogonality of truncation noise using IFFT coefficient blocks. In Sect. 4.1.2, we propose a method to deal with the cross terms among coefficient blocks.

4.1.1 Shaping truncation errors of SOPOT coefficients via a single IFFT coefficient block

The N-point Discrete Fourier Transform and its inverse transform are defined as (4.1) and (4.2), respectively:

$$F(k) = \sum_{n=0}^{N-1} f(n) \cdot W^{k \cdot n}, \quad 0 \le k \le N-1 \tag{4.1}$$

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) \cdot W^{-n \cdot k}, \quad 0 \le n \le N-1 \tag{4.2}$$

where $W^n = \exp(-j \cdot 2\pi \cdot n/N)$

When we apply the IFFT to compute (4.2), the rotation operation W^{-nk} is decomposed into a series of sub-rotations $W_{(n,k)}^{-\alpha(\cdot)}$ [16], as illustrated in Fig. 3:

$$\prod_{i=1}^{v(k)} W_{(n,k)}^{\alpha(i)} = W^{nk} \tag{4.3}$$

where $v(k)$ is the number of sub-rotations for $F(k)$

In particular, from (4.2), we can derive $\tau_k(n), 0 \le n \le N-1$, which are the temporal components generated from frequency coefficient $F(k)$:

$$\tau_k(n) = \frac{1}{N} \cdot F(k) \cdot \prod_{i=1}^{v(k)} W_{(n,k)}^{-\alpha(i)} \tag{4.4}$$

The truncation of a SOPOT coefficient is equivalent to introducing an additive error, and the reconstructed sample can be calculated as:

$$\hat{f}(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k) \cdot \prod_{i=1}^{v(k)} \left(W_{(n,k)}^{-\alpha(i)} + \delta_{(n,k)}^{(i)} \right) \tag{4.5}$$

Next we investigate the truncation errors when we only truncate the SOPOT coefficients of a single IFFT coefficient block. We define an IFFT coefficient block as $\{W_{(n,k)}^{-\alpha(i)} | 0 \le n \le N-1\}$, which are those coefficients grouped in the same box in Fig. 3. An important property of the coefficient block is that the transform of a frequency coefficient can be decomposed into a series of calculations using the coefficient blocks. This property allows us to control the output of the coefficient block to achieve the desired truncation noise shaping.

Without loss of generality, let the jth IFFT coefficient block associated with $F(k)$ be truncated. From (4.4), we have:

$$\hat{\tau}_k(n) = \frac{1}{N} F(k) \cdot \left(\left(W_{(n,k)}^{-\alpha(j)} + \delta_{(n,k)}^{(j)} \right) \right) \cdot \prod_{i=1, i \ne j}^{v(k)} W_{(n,k)}^{-\alpha(i)} \tag{4.6}$$

To derive the spectrum of the truncation error, we conduct DFT over the time domain errors. Based on (4.1), and (4.4), we can represent the spectral error over frequency coefficient $F(m)$ as

$$\Delta \hat{F}_k(m) = \frac{1}{N} \cdot \sum_{n=0}^{N-1} \left(\hat{\tau}_k(n) - \tau_k(n) \right) \cdot W^{nm} \tag{4.7}$$

Substituting (4.4), (4.6) into (4.7), and using (4.3), we have

$$\begin{aligned} \Delta \hat{F}_k(m) &= \frac{F(k)}{N} \\ &\cdot \sum_{n=0}^{N-1} \left(\prod_{i=1}^{v(k)} W_{(n,k)}^{-\alpha(i)} + \delta_{(n,k)}^{(j)} \prod_{\substack{i=1 \\ i \ne j}}^{v(k)} W_{(n,k)}^{-\alpha(i)} - \prod_{i=1}^{v(k)} W_{(n,k)}^{-\alpha(i)} \right) \cdot W^{nm} \\ &= \frac{F(k)}{N} \cdot \sum_{n=0}^{N-1} \left(\delta_{(n,k)}^{(j)} \prod_{\substack{i=1 \\ i \ne j}}^{v(k)} W_{(n,k)}^{-\alpha(i)} \right) \cdot W^{nm} \end{aligned} \tag{4.8}$$

In (4.8), it implies that the truncation noise of the frequency coefficient $F(k)$ will be spread to other frequency coefficients when $\{\delta_{(n,k)}^{(j)} | 0 \le n \le N-1\}$ are random variables. This is the exact case for conventional analysis of coefficient quantization problems.

To guarantee the orthogonality of the truncation noise, the truncation errors of a coefficient block should satisfy the orthogonal condition of the truncation noise, which is described in (4.9):

$$\delta_{(m,k)}^{(j)} \cdot W_{(m,k)}^{\alpha(j)} = \delta_{(n,k)}^{(j)} \cdot W_{(n,k)}^{\alpha(j)}, \quad 0 \le m, n < N \tag{4.9}$$

In other words, $\{\delta_{(n,k)}^{(j)} \cdot W_{(n,k)}^{\alpha(j)} | 0 \le n \le N-1\}$ have equal values. This is shown as follows.

$$\begin{aligned}
&\Delta \hat{F}_k(m) \\
&= \frac{F(k)}{N} \cdot \sum_{n=0}^{N-1} \left(\delta_{(n,k)}^{(j)} \prod_{\substack{i=1 \\ i \neq j}}^{v(k)} W_{(n,k)}^{-\alpha(i)} \right) \cdot W^{nm} \\
&= \frac{F(k)}{N} \cdot \sum_{n=0}^{N-1} \left(\delta_{(n,k)}^{(j)} \prod_{\substack{i=1 \\ i \neq j}}^{v(k)} W_{(n,k)}^{-\alpha(i)} \right) \cdot W^{nk} \cdot W^{-nk} \cdot W^{nm} \\
&= \frac{F(k)}{N} \cdot \left(\delta_{(n,k)}^{(j)} W_{(n,k)}^{\alpha(j)} \right) \sum_{n=0}^{N-1} W^{-nk} \cdot W^{nm}
\end{aligned}$$

Clearly, $\sum_{n=1}^{N-1} W^{nk} \cdot W^{-nm} = 0$ for any $m \neq k$. This justifies the orthogonality of the truncation noise.

It should be noted that when we let various $\{\delta_{(n,k)}^{(j)} \cdot W_{(n,k)}^{\alpha(j)} | 0 \le n \le N-1\}$ have equal values, $\{\delta_{(n,k)}^{(j)} | 0 \le n \le N-1\}$ are no longer truncation errors. But they support the truncation operation in the way that these errors dominate the distortion and make the actual truncation errors of smaller levels negligible. Thus the magnitude of $\delta_{(\cdot,k)}^{(i)}$ can serve as an indicator for the truncation position of the corresponding SOPOT coefficient, which will be shown in Sect. 4.2. In this sense, we still call them "truncation errors".

4.1.2 Modeling the noise from a series of IFFT coefficient blocks

We have proposed a method to shape the truncation errors by a single IFFT coefficient block in Sect. 4.1.1. As illustrated in Fig. 3, most frequency coefficients are associated with a series of blocks. To build a model to represent the sum of truncation noises, which results from a series of blocks, for a frequency coefficient, we need to extend the results presented in Sect. 4.1.1.

The sum of truncation errors for frequency coefficient $F(k)$ can be represented as

$$\begin{aligned}
e(k) &= \hat{F}(k) - F(k) \\
&= \frac{F(k)}{N} \sum_{n=0}^{N-1} \left(\prod_{i=1}^{v(k)} \left(W_{(n,k)}^{-\alpha(i)} + \delta_{(n,k)}^{(i)} \right) - \prod_{i=1}^{v(k)} W_{(n,k)}^{-\alpha(i)} \right) \cdot W^{n \cdot k}
\end{aligned} \tag{4.10}$$

As the errors are of small values, we can neglect the higher order error terms. Substituting (4.3), we have

$$\begin{aligned}
&\prod_{i=1}^{v(k)} \left(W_{(n,k)}^{-\alpha(i)} + \delta_{(n,k)}^{(i)} \right) - \prod_{i=1}^{v(k)} W_{(n,k)}^{-\alpha(i)} \\
&= \sum_{i=1}^{v(k)} \delta_{(n,k)}^{(i)} \cdot \prod_{j=1, j \neq i}^{v(k)} W_{(n,k)}^{-\alpha(j)} \\
&= W^{-n \cdot k} \sum_{i=1}^{v(k)} \delta_{(n,k)}^{(i)} \cdot W_{(n,k)}^{\alpha(i)}
\end{aligned} \tag{4.11}$$

From (4.11), for any $i \in [1, v(k)]$, if we let $\{\delta_{(n,k)}^{(i)} \cdot W_{(n,k)}^{\alpha(i)} | 0 \le n \le N-1\}$ have equal values, the orthogonality of the truncation noise holds for a series of coefficient blocks. For convenience, when the orthogonality of the truncation noise holds, we denote $\delta_{(n,k)}^{(i)} \cdot W_{(n,k)}^{\alpha(i)} = |\delta_{(k)}^{(i)}| \cdot e^{j\varphi(k,i)}$.

Thus the energy of the accumulated truncation noises for frequency coefficient $F(k)$, which results from a series of

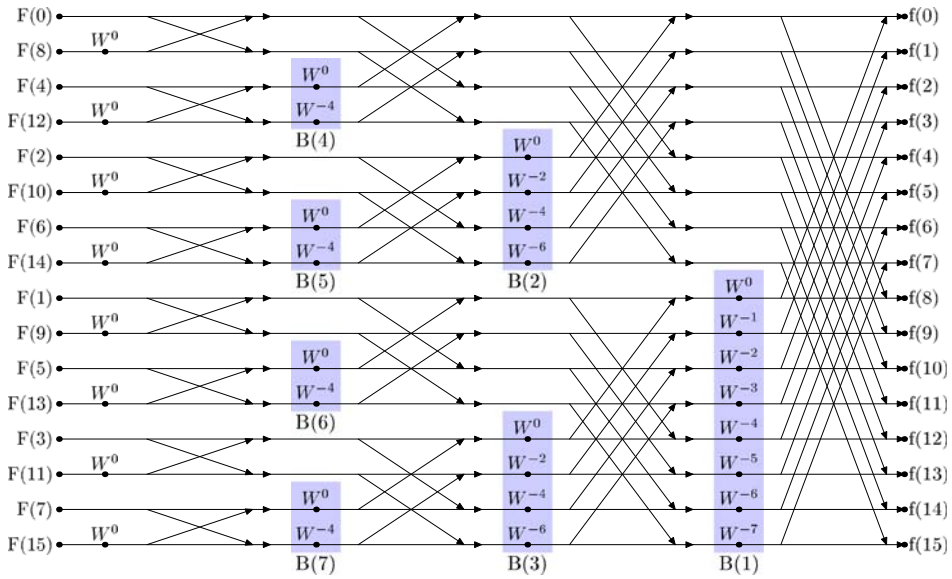

Fig. 3 The flow graph of a 16-point inverse FFT with marked coefficient blocks

blocks, is represented as

$$|e_T(k)|^2 = F^2(k) \cdot \left| \sum_{i=1}^{v(k)} \left| \delta_{(k)}^{(i)} \right| \cdot e^{j\varphi(k,i)} \right|^2 \tag{4.12}$$

As we further develop (4.12), a critical issue is how to effectively deal with the cross terms between truncation errors. In statistical model based approaches [9,16], these truncation errors are assumed as independent random variables with mean value of zero and their cross terms will vanish for the expectation of sum of truncation noises. In our work, however, these errors are subject to the orthogonal conditions, and they cannot be assumed as independent random variables with mean value of zero. These non-zeroed cross terms lead to two undesired consequences. First, these cross terms will complicate the analysis of noise allocation for IFFT coefficient blocks. Second, and more importantly, these cross terms potentially lead to additional noises. These additional noises will degrade the efficiency of workload reduction. To address the issue of the cross terms between truncation errors, we suppress the sum of truncation noises by shaping the angles of the truncation errors. Furthermore, the angle-shaping technique enables us to develop a cross term deleted representation of the sum of truncation noises.

To facilitate the angle shaping, we limit the value set of $\{\varphi(k, j)|0 \le k \le N-1, 1 \le j \le v(k)\}$ to be $\{\pi/4, 5\pi/4\}$. These two angles are opposite to each other. When we assign those block truncation errors with different angles, these truncation errors will subtract from each other, to make the noises as small as possible. Further, it reduces the angle-shaping process into "sign" assignment operation: "+" denotes $\pi/4$, and "−" denotes $5\pi/4$. We have designed the following algorithm to accomplish the assignment for all the blocks.

We organize IFFT coefficient blocks as the following structure: a block and all of its left-covering blocks form a block tree and the largest block acts as the root of the tree. On the other hand, for block k, we define $P(k)$ as a block set, all of its elements left-cover block k. For example, in Fig. 3, {B(1), B(3), B(6), B(7)}, {B(3), B(7)} and {B(7)} are three of such block trees; P(6) = {B(1)} and P(7) = {B(3), B(1)}.

We start with the largest tree and perform the assignment for the root. Deleting the root which has been processed, the current tree is decomposed into several smaller trees. We then iteratively move to these block trees and conduct the same computation for their roots until all blocks are processed.

We denote δ_j and $S(j)$ as the noise level and sign of block j. For current block k, given the truncation noise levels of all coefficient blocks which left-cover block k, we have:

$$S(k) = -SIGN\left(\sum_{j \in P(k)} S(j) \cdot \left| \delta_j \right|^2 \right) \tag{4.13}$$

According to this algorithm, we can derive an upper bound on the overall truncation noise.

In (4.13), the "sign" relationship among truncation errors is described. Next we need to determine the angle of the concerned truncation error according to its given "sign".

It should be noted that according to our angle-shaping specification, $\pi/4 + \pi \to 5\pi/4$, and $5\pi/4 + \pi \to \pi/4$. In other words, sign switching can be achieved by rotating an angle of π. From (4.13), we have

$$\varphi(k, v(k)) = \pi + \angle \left(\sum_{i=1}^{v(k)-1} \left| \delta_{(k)}^{(i)} \right| \cdot e^{j\varphi(k,i)} \right) \tag{4.14}$$

Thus

$$\left| \sum_{i=1}^{v(k)} \left| \delta_{(k)}^{(i)} \right| \cdot e^{j\varphi(k,i)} \right|^2 \le \left| \delta_{(k)}^{(v(k))} \right|^2 + \left| \sum_{i=1}^{v(k)-1} \left| \delta_{(k)}^{(i)} \right| \cdot e^{j\varphi(k,i)} \right|^2 \tag{4.15}$$

In this way, we can iteratively decompose the last term at the right side of (4.15) and we have a cross-term deleted representation of the sum of truncation noise:

$$\begin{aligned} |e(k)|^2 &= F^2(k) \cdot \left| \sum_{i=1}^{v(k)} \left| \delta_{(k)}^{(i)} \right| \cdot e^{j\varphi(k,i)} \right|^2 \\ &\le F^2(k) \cdot \sum_{i=1}^{v(k)} \left| \delta_{(k)}^{(i)} \right|^2 \end{aligned} \tag{4.16}$$

4.2 Noise allocation over SOPOT coefficient blocks

Given the MQDs of all the scale factor bands, we need to allocate them to the IFFT coefficient blocks. Based on its allocated noises, we can perform truncation over an IFFT coefficient block to achieve workload reduction.

First we need to establish the relationship between the number of reserved SOPOT terms and their corresponding truncation noise. We denote b as the truncation position of coefficient x, and $I_i(x)$ as its indicator of SOPOT terms. Then we have

$$\sum_{i=0}^{b-1} I_i(x) \le b, \quad \begin{cases} I_i(x) = 1, & x \text{ has a SOPOT term at position i} \\ I_i(x) = 0, & x \text{ has no SOPOT term at position i} \end{cases} \tag{4.17}$$

Its implication is straightforward: we can effectively reduce the number of SOPOT terms by left-shifting the truncation position. Due to this, we use the truncation position as the estimation of the computational workload of a SOPOT coefficient. On the other hand, the relationship between the truncation noise e_t^2 and the truncation position of b has been well-established in the literature.

For a fixed point coefficient, we have [9]:

$$e_t^2(b) = 2^{-2b}/6 \tag{4.18}$$

Next we need to associate truncation noise with the allocated noise $|\delta_i|^2$ of block i. Towards this, we choose the truncation position for block i as follows, where $c > 0$, being a constant for all coefficients:

$$b = \arg\min_j \left(I_j\left(|\delta_i|\right) = 1\right) + c \tag{4.19}$$

This indicates that we only reserve c bits of $|\delta_i|$ from its first non-zero most significant bit and the rest of the bits are discarded to save computation. Thus we can derive the relationship between the allocated noise $|\delta_i|^2$ of block i and its associated truncation noise. We first calculate the expectation of the ratio between the value of those reserved bits of $|\delta_i|$, which we denote as $|\tilde{\delta}_i|$, and the unit at the truncation position, whose value is 2^{-b} according to (4.19). Such ratio can be employed to develop the concerned relationship. Considering $|\tilde{\delta}_i|$, (a). the bit offsetting the truncation position c bits must be 1, following (4.19); (b). the value of the rest $(c-1)$ bits can be assumed to be evenly distributed, ranging from 0 to $2^c - 1$. Then we have

$$\frac{|\tilde{\delta}_i|}{2^{-b}} = 2^c + 2^{-c} \cdot \sum_{k=0}^{2^c-1} k = 2^c + 2^{c-1} - 0.5 \tag{4.20}$$

Making use of the relationship between the actual truncation error $e_t(b)$ and 2^{-b} described in (4.18), and $|\delta_i| = |\tilde{\delta}_i| + e_t(b)$, we have

$$\begin{aligned} |\delta_i|^2 &= \left(\frac{\left|\tilde{\delta}_i\right| + e_t(b)}{e_t(b)}\right)^2 \cdot e_t(b) \\ &= \left(\sqrt{6} \cdot \left(2^c + 2^{c-1} - 0.5\right) + 1\right)^2 \cdot e_t^2(b) \end{aligned} \tag{4.21}$$

In (4.21), we can see that the constant c determines the scaling operation by right-shifting the truncation position. Due to this, we call c the scaling factor of truncation noise. The relationship described in (4.21) implies that the truncation noise is reduced rapidly as c grows. For example, when c equals 3, the truncation noise is only a factor of 0.00117 of the allocated noise. Therefore these actual truncation errors can be safely neglected when using appropriate values of the scaling factor of truncation noise.

From (4.18) and (4.21), we have $b = -0.5 \cdot \log_2(\sqrt{6} \cdot (2^c + 2^{c-1} - 0.5) + 1)^2 - 0.5 \log_2 |\delta_i|^2$. Neglecting those constants, we can estimate the workload of a SOPOT coefficient of the ith block as $-\log_2 |\delta_i|^2$. Based on (2.1), we can formulate the noise allocation problem as follows:

$$\begin{aligned} &\text{Min} \sum_{j=1}^{N/2-1} -l(j) \cdot \log_2 \left|\delta_j\right|^2 \\ &\text{Subject to} \sum_{k\in\beta(i)} |e_T(k)|^2 + E_{Q(i)} \le E_{T(i)}, \quad 0 \le i < 49 \end{aligned} \tag{4.22}$$

where N is the block size, $l(j)$ is the number of coefficients of the jth coefficient block.

Based on (4.16) and the result in [10], which calculates the $E_{Q(i)}$ in terms of the quantization step size and the coefficient values of scale factor band i, we can develop (4.22) into the following, where $\Delta_{Q(i)}$ denotes the quantization step size for scale factor band i:

$$\begin{aligned} &\sum_{k\in\beta(i)} F^2(k) \sum_{j=1}^{v(k)} \left|\delta_{(k)}^{(j)}\right|^2 = E_{T(i)} - \frac{4}{27} \cdot \Delta_{Q(i)}^2 \\ &\sum_{k\in\beta(i)} |F(k)|^{0.5}, \quad 0 \le i < 49 \end{aligned} \tag{4.23}$$

Although this is a problem of finding extrema under constraints which can be solved by the Lagrange multiplier method in principle, the actual computation is prohibitively complex: it needs to solve a non-linear equation group including 49 equations, having 49 variables with maximal power of 255.

To address this issue, we resort to a heuristic to accelerate the computation and find a satisfactory solution. We make use of the same structure of IFFT coefficient blocks used in algorithm 4.1. Instead of finding the global optimal solution, we search a value of the truncation noise for the root of a block tree which will make the current block tree have the smallest sum of reserved bits. By deleting the root which has been assigned a truncation noise value, the current tree is decomposed into several smaller trees. We iteratively move to these block trees and conduct the same computation for their roots until each block is assigned a truncation noise.

In this method, we are frequently required to allocate the allowed noise of a scale factor band or a frequency coefficient over its associated IFFT coefficient blocks to achieve the minimal workload, without involving other scale factor bands or frequency coefficients. We call this problem "independent noise allocation". In this case, the allocation algorithm should allocate larger noise to blocks including more coefficients. For IFFT, although different stages have different numbers and sizes of blocks, the total number of coefficients remains the same for all stages. Based on this observation, identical noise should be allocated to all of the associated coefficient blocks.

The proposed allocation method consists of two steps. The first step is to allocate the MQD of a scale factor band to its frequency components. Following the independent noise allocation principle, we should allocate identical noise to all

the blocks associated with the scale factor band. In this way, for scale factor L, let its associated blocks be $B(L)$, and we have $|\delta_{(m)}^{(i)}| = |\delta_{(n)}^{(j)}|$ for any $m, n \in B(L)$, and $i \in v(m)$, $j \in v(n)$. Then we can derive the initial allowed noise for each coefficient of the scale factor band from (4.23). For a frequency coefficient k, let its initial allowed noise be $\varepsilon(k)$, its associated block number be $\|B(k)\|$, and the average noise of each block for the jth scale factor band be $\zeta(j)$, then we have

$$\zeta(j) = \frac{E_{T(j)} - \frac{4}{27} \cdot \Delta_{Q(i)}^2 \sum_{k \in \beta(i)} |F(k)|^{0.5}}{\sum_{k \in \beta(j)} F^2(k) \cdot \|B(k)\|} \tag{4.24}$$

$$\varepsilon(k) = \|B(k)\| \cdot \zeta(j), \quad k \in \beta(j) \tag{4.25}$$

After the first step, a coefficient block usually has multiple noise values assigned by various frequency coefficients. In the second step, we will choose an appropriate value for each coefficient block. As discussed earlier, we reduce this problem to find a desired noise value for the root of a block tree. Let the block index of the root be i, its associated frequency coefficients be $T(i)$, block numbers between $F(j)$ and block i be $R(i, j)$, the residual allowed noise for $F(j)$ be $\breve{\varepsilon}(j)$, the desired noise value of the root be $\bar{\varepsilon}_i$. To estimate the workload for $F(j)$, we perform the independent noise allocation over its residual allowed noise. Thus $\bar{\varepsilon}_i$ will minimize the following:

$$-l(i) \cdot \log_2 \bar{\varepsilon}_i - \sum_{j \in T(i)} R(i, j) \cdot \log_2 \left(\frac{\breve{\varepsilon}(j) - \bar{\varepsilon}_i}{R(i, j)} \right) \tag{4.26}$$

Numerical techniques can be employed to find the desired value of $\bar{\varepsilon}_i$ in (4.26).

4.3 The workload estimation module

As discussed in Sect. 3, the workload estimation module provides three functionalities: (1) to estimate the workload for the set of truncated SOPOT coefficients; (2) to choose a scale factor band to decrease its quantization step; (3) to control the second pass processing. In this section, we will discuss these three aspects in detail.

In SOPOT operations, the basic calculation units are the shift and add operations. This enables us to measure the computational workload of SOPOT IFFT in terms of number of shift and add operations. This measure is similar to the widespread workload estimation for IFFT of the floating point version using the number of multiplications. On the other hand, we can derive the exact number of shift-and-add operations from the sum of reserved SOPOT terms of the IFFT.

When the estimated workload is greater than the workload constraint, the workload estimation module needs to choose a scale factor band to reduce its quantization step size. From (4.22), we can see that MQD of the jth scale factor band allocates truncation noise $\zeta(j)$ for each of its associated coefficient blocks. On the other hand, these coefficient blocks are also shared by other scale factor bands. Then these blocks have different allocated truncation noises from different scale factor bands. To provide transparent playback quality, the performance of workload reduction is limited by the minimal level of the various allocated truncation noises. Due to this fact, we should choose scale factor band j to increase its MQD, where $\zeta(j) \leq \zeta(i), 0 \leq i < 49$.

The workload estimation module employs an iteration procedure to control the second processing pass. As the procedure of control is presented in Sect. 3, we only present the termination conditions of the processing loop. Normally the processing loop terminates when the estimated workload is below the specified workload level. However, this is not always possible to achieve. In this case we exploit two other termination conditions, which are in accordance with those used in a conventional AAC encoder [8]: (1) The next iteration would require all of the scale factor bands to be amplified; (2) The next iteration would cause the difference between two consecutive scale factors to exceed 60.

5 Experimental results

To evaluate the performance of the proposed scheme, we implemented a prototype. We employed Free Advanced Audio Coder (FAAC) version 0.60 [17] as the conventional AAC encoder core in our work, as FAAC is a well-known open source AAC encoder. We note that FAAC 0.60 is not the latest version. (In August 2006, FAAC version 1.25 was released.) We have chosen FAAC 0.60 as our implementation platform because that the important algorithms used in the FAAC 0.60 are in accordance with those described in the informative parts of AAC standards [8]. On the other hand, FAAC 1.25 employs new techniques, which are not well-documented, for psychoacoustic modeling and noise allocation. Both of them produce output for the second processing pass. In this case, well-documented techniques are better for serving as a proof of concept.

By analyzing the output of the conventional encoder core, we have found that the masking threshold constraints of some scale factor bands had been violated sometimes. For these scale factor bands, we define the initial level of their MQDs as zero, rather than a negative value. This method ensures that the generated audio file will not be of lower quality than the version by the conventional encoder core.

An important parameter, which is missing in the conventional AAC encoder core, is the scaling factor of the truncation noise, which has been defined in (4.20). The value of the scaling factor has a close relationship with the computational workload and the playback quality. Due to its importance, we perform experiments for various values of the scaling factor, ranging from 3 to 5.

We carried out experiments on six selected audio clips, including five popular songs, and one instrumental music. All of them were extracted from CDs, coded in WAV format, at a sampling rate of 44,100 samples/s, 16 bits per sample, stereo mode.

5.1 IFFT workload reduction

A primary motivation of our work is to reduce the computational workload at the decoder side. To achieve this, we implemented IFFT in SOPOT, which is an important step in IMDCT module.

We first estimated the workload portion of IMDCT module in the low complexity profile. Towards this, we executed an AAC decoder, Free Advanced Audio Decoder 2 (FAAD2) 2.0 [17] on an ARM simulation tool: Simplescalar/arm [18]. We carried out the simulation in two settings: (1) floating point version: we made use of a software package to implement the floating point operations for all the processing modules; (2) fixed point version: we employed un-truncated SOPOT coefficients for IMDCT, which does not introduce any coefficient quantization noise, and fixed point multiplication for the other modules. This was because fixed point multiplication cannot provide required accuracy for 2,048-point IMDCT. Simulation results showed that IMDCT is responsible for 86 and 92% workload of the entire decoding process, in the floating point version, and the fixed point version, respectively. These results provide strong motivations for workload reduction in IMDCT. In addition, IFFT is responsible for around 55% workload of IMDCT in both settings.

Next we estimated the workload reduction of IFFT. We measured the computational workload of IFFT of SOPOT version in terms of the number of shift-and-add operations, as the method used in Sect. 4.3. We have counted the exact number of shift-and-add operations when performing the IFFT. As mentioned above, the workload is related to the scaling factor of truncation noise. We varied the values of the scaling factor of truncation noise from 3 to 5, to change the workload. The baseline of the workload was calculated using the un-truncated SOPOT coefficients for the same input data. We present the results in Fig. 4, where the baseline workload is assumed as unit.

As shown in Fig. 4, we have achieved significant workload reduction for IFFT computation in an AAC decoder. When the proposed work encodes the audio data with the scaling factors of truncation noise of 3, 4 and 5, on average, we save 77.8, 75.0 and 72.8% computations, respectively. To our knowledge, the presented results are better than all the results reported in the literature for a 512-point of transform. In general, although various methods have been proposed to save the computations for transform, it is hard to find an effective way to reduce the workload of a long block of transforms. We illustrate this fact by the following examples.

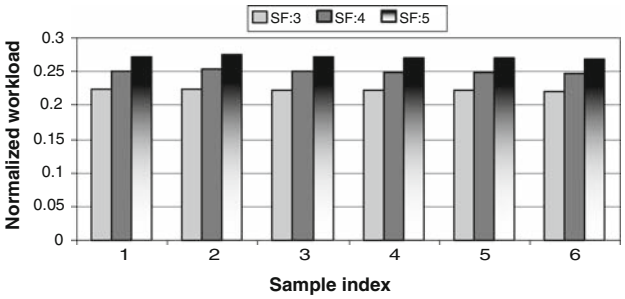

Fig. 4 Normalized workload for the test audio clips, where SF denotes the scaling factor of truncation noise

In terms of the pruning techniques, for IDCT, the relationship between workload reduction ratio G, the block size B, and the number of non-zero frequency coefficient b, can be described as follows [21]:

$$G \approx 1 - \log b / \log B \tag{5.1}$$

We can then estimate the workload reduction for a 512-point of IDCT using the pruning technique. According to (5.1), even though we discard three quarters of the frequency coefficients, where audible quality degradation occurs, we can only achieve 22.2% computation savings for the best case.

On the other hand, for approximate techniques, the basic assumption is that the noise introduced by approximation is negligible in comparison with the energy of the signal. This assumption is valid only for transforms with short block [3]. However, the approximation noise accumulates as the block size increases. As a consequence, the approximation noise can no longer be neglected for a long block transform.

We solved the problem in an alternative way. In a similar manner to the "lossy compression" used in audio encoding to achieve high-compression ratios, for the long block size of transform, we performed "lossy transform", where we allowed the noises with larger levels to achieve significant workload reduction. We then addressed these noises by exploiting MQD and the possibility to control the quantization noise.

5.2 Subjective evaluation

To evaluate our scheme, we carried out subjective tests on a group of 30 subjects (male and female undergraduate students with normal hearing). All subjects were asked to evaluate audio quality using the mean opinion score (MOS), which is a five-point scale (5-excellent, 4-good, 3-fair, 2-poor, and 1-bad).

We encoded the selected six audio clips into AAC bitstreams using the prototype encoder. We set the encoding parameters for the conventional AAC encoder core as follows: bitrate 128 Kb/s, switching on low complexity profile, temporal noise shaping, Middle/Side coding, and switching

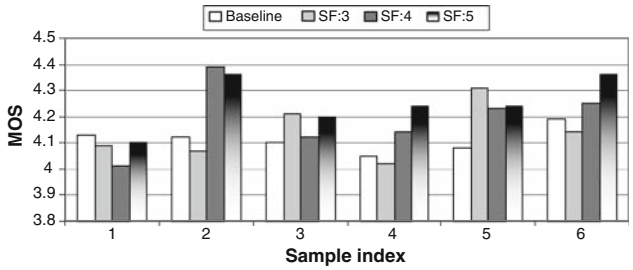

Fig. 5 Averaged MOS values for the test audio clips, where SF denotes the scaling factor of truncation noise

off long term prediction, perceptual noise substitution, and intensity coding, which are typical options for an AAC encoder. We prepared four copies for each program. Three copies were generated by our scheme with scaling factor of truncation noises as 3, 4, and 5, and the corresponding workload reduction ratios were chosen as 77.8, 75.0, and 72.8%, respectively. The fourth copy was generated by a FAAC 0.60 encoder with the same encoding parameters, which served as the baseline sample. In addition, each program was also given the uncompressed clip as reference of MOS 5. For fairness, all test samples were arranged in random order. We presented the averaged MOS values for each copy in Fig. 5.

From Fig. 5, we can see that the subjects cannot tell perceptual differences among most test samples. This fact shows that our scheme can usually provide transparent playback quality in comparison with the baseline. We can achieve comparable playback quality using a scaling factor of truncation noise no less than 3. Another interesting observation is that the proposed scheme provides better quality than the baseline for sample 2 when SF equals 4 or 5. This example reveals an important property of our proposed scheme: it yields improved playback quality compared to the baseline. We analyze it as follows. In our scheme, the actually introduced truncation noise is less than the noise estimation, according to (4.16). The noise estimation is further employed to allocate additional MQD by the joint noise shaping process. Thus the increased MQD will not be filled fully, and consequently, the proposed scheme has higher residual MQD than the baseline. This is especially important for those scale factor bands where the quantization noise exceeds the masking threshold (initial MQD being zero, as discussed in introduction part of Sect. 5), since it represents less audible noise.

5.3 Increase of file sizes

High compression ratios are an important goal for audio encoders. As mentioned, our scheme will sacrifice some compression efficiency in exchange for the reduction of the decoder's computational workload. In this section, we will investigate the compression efficiency of our scheme by comparing the file sizes generated by our work with that by the baseline AAC encoder.

For the prototype encoder, we used the same encoding parameters as those presented in 5.2. In particular, scaling factor of truncation noises was chosen to be 3 and the workload reduction ratio was chosen to be 77%. FAAC 0.60 is chosen to be the baseline encoder with the identical bit rates and encoding parameters as those used by the conventional encoder core in the prototype encoder. The file size generated by our scheme depends on the specified bit rate for the conventional encoder core. In AAC standard, ten bit rates are provided, ranging from 64 to 320 Kb/s. As our scheme targets scenarios of high quality audio entertainment, the requirement for the bit rate of the conventional encoder core should be at least 128 Kb/s. But we added two lower bit rates to demonstrate the characteristics of the proposed work. Thus we varied the bit rate of the encoders as 96, 112, 128, 160, 192, 224, 256 and 320 Kb/s. For each level of the bit rates, we measured the generated file sizes and computed the increase ratios compared to the baseline, as shown in Fig. 6.

From Fig. 6, we can see that the increase ratios of the compressed file sizes decrease as the bit rate increases. This can be explained as follows. The initial levels of MQDs increase as the specified bit rates of the baseline encoder become higher. The required changes of the quantization step sizes for each scale factor band become smaller. Consequently, this results in smaller increase of the bit length of the coded frequency coefficients. For the bit rate of 128 Kb/s, the average file size increase ratio is 9.52%. This implies that our scheme only incurs a modest increase in file sizes. We believe that this is acceptable for the targeted application scenarios. On the other hand, the results shown in Fig. 6 also demonstrate the necessity of our work. Though we have encoded the audio data in 320 Kb/s, which is the highest bit rate supported by the AAC standard, there was 4.13% increase in file size. This

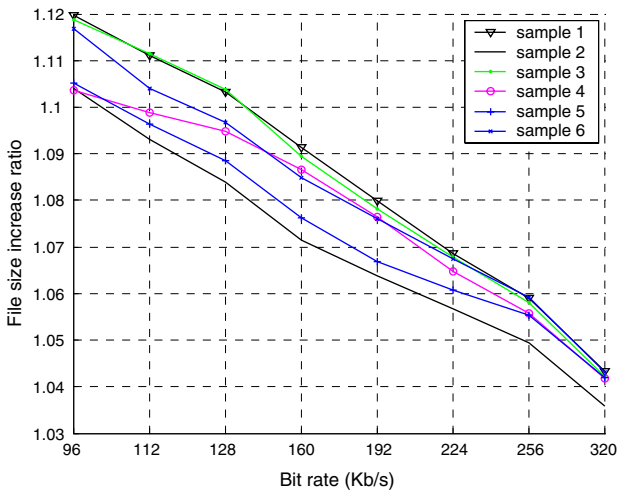

Fig. 6 Increase ratios of file sizes for various encoding bit rates

suggests ASP noise still violates the constraints of MQDs. This implies that the initial levels of the MQDs provided by the conventional AAC encoder cannot mask all the ASP noise and special techniques, like our scheme, are required to address those ASP noises.

6 Conclusion and future work

We have proposed a novel framework, a joint encoder–decoder framework (JEDF), which allows the decoder to reduce workload at the expense of storage without sacrificing playback quality. This is achieved by a joint noise shaping process. In comparison with existing methods, we believe that the JEDF concept has identified a new direction in designing low power multimedia-capable embedded systems. We have implemented the proposed scheme with the MPEG AAC codec for proof of concept.

The reduced workload resulting from JEDF should be exploited by dynamic voltage scaling (DVS) techniques to save energy and to prolong the battery life. The appropriate implementation of DVS underlying JEDF is out of the scope of this paper. However, we can roughly estimate the saved energy in terms of workload reduction. With DVS, as a first order of approximation, an application has a cubic relationship between its workload and its energy consumption [20]. We can then estimate the energy saving of JEDF to be more than 70%, which demonstrates its potential in terms of energy efficiency.

In our pilot implementation, we have concentrated on the truncation noise shaping of IFFT in the IMDCT to achieve the specified workload reduction. As an immediate next step, we plan to extend the proposed scheme to other parts of the IMDCT. Furthermore, how to represent the new side information such as the truncation positions of SOPOT coefficients in an efficient way remains to be addressed by the future work.

Acknowledgements We thank the anonymous reviewers for their critical comments, which have helped to improve the quality of this paper. The work carried out was funded by the Singaporean Ministry of Education under the research grant R-252-000-236-112.

References

1. Argenti, F., Del Taglia, F., Del Re, E.: Audio decoding with frequency and complexity scalability. IEE Proc. Vis. Image Signal Process. **149**(3), 152–158 (2002)
2. Benini, L., Micheli, G.D.: System-level power optimazation: techniques and tools. ACM Trans. Des. Autom. Electron. Syst. **5**(2), 115–192 (2000)
3. Chan, S.C., Yiu, P.M.: An efficient multiplierless approximation of the fast fourier transform using Sum-of-Powers-of-Two (SOPOT) coefficients. IEEE Signal Process. Lett. **9**(PART 10), 322–325 (2002)
4. Ghurumuruhan, G., Prabhu, K.M.M.: Fixed-point fast hartley transform error analysis. Signal Process. **84**(8), 1307–1321 (2004)
5. Huang, W., Wang, Y., Chakraborty, S.: Power-Aware bandwidth and stereo-image scalable audio decoding. In: ACM Multimedia Conference. Singapore, pp. 291–294 (2005)
6. Hu, Z., Wan, H.: A novel generic fast fourier transform pruning technique and complexity analysis. IEEE Trans. Signal Process. **53**(1), 274–282 (2005)
7. ISO/IEC (2000) MPEG1 11172-3: Audio Coding
8. ISO/IEC (2006) MPEG2 13818-7: Advanced Audio Coding
9. James, D.V.: Quantization errors in the fast fourier transform. IEEE Trans. Acoust. Speech Signal Process. **ASSP-23**(3), 277–283 (1975)
10. Liu, C.M., Lee, W.C., Chien, C.T.: Bit allocation for advanced audio coding using bandwidth-proportional noise-shaping criterion. In: International Conference on Digital Audio Effects (DAFX). London, pp. 233–237 (2003)
11. Liang, J., Tran, T.D.: Fast multiplierless approximations of the DCT with the lifting scheme. IEEE Trans. Signal Process. **49**(12), 3032–3044 (2001)
12. Markel, J.D.: FFT pruning. IEEE Trans. Audio Electroacoustics **AU-19**, 305–311 (1971)
13. McMillan, L., Lee, W.: A forward-mapping realization of the inverse discrete cosine transform. In: IEEE Data Compression Conference. Snowbird, Utah, USA, pp. 219–228 (1992)
14. Mesarina, M., Turner, Y.: Reduced energy decoding of mpeg streams. Multimed. Syst. **9**(2), 202–213 (2003)
15. Oppenheim, A., Nawab, H., Verghese, G., Womell, G.: Algorithms for signal processing. In: Rapid Prototyping of Application Specific Signal Processors Conference (RASSP). Arlington, Virginia, USA, pp. 146–153 (1994)
16. Oppenheim, A.V., Weinstein, C.J.: Effects of finite register length in digital filtering and the fast fourier transform. Proc. IEEE **60**(8), 957–976 (1972)
17. http://www.audiocoding.com
18. Simplescalar/Arm: http://www.simplescalar.com/v4test.html
19. Tajana, S., Micheli, G.D., Benini, L., Mat, H.: Source code optimization and profiling of energy consumption in embedded systems. In: International Symposium on Systems Synthesis. Madrid, Spain, pp. 193–199 (2000)
20. Trevor, M.: Power: a first-class architectural design constraint. IEEE Comput. **34**(4), 52–58 (2001)
21. Wang, Z.: Pruning the fast discrete cosine transform. IEEE Trans. Commun. **39**(5), 640–643 (1991)
22. Zheng, F., Garg, N., Sobti, S., Zhang, C., Joseph, R., Krishnamurty, A., Wang, R.: Considering the energy consumption of mobile storage alternatives. In: IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. Orlando, USA, pp. 36–45 (2003)