DESIGN AND IMPLEMENTATION OF AC-3 CODERS

Steve Vernon

Dolby Laboratories Inc., 100 Potrero Avenue, San Francisco, California 94103
reprinted by permission of the Institute of Electrical and Electronics Engineers
Published in IEEE Tr. Consumer Electronics, Vol. 41, No. 3, August 1995

ABSTRACT

AC-3 is the perceptual coding technology used for HDTV audio compression. This paper describes the design and implementation of AC-3 coders, focusing on issues relevant to minimum cost solutions.

1. INTRODUCTION

In the past several years, digital audio data compression has become an important technique in the audio industry. New formats have been introduced that allow high quality audio reproduction without the need for the high data bandwidth that would be required using traditional techniques [1,2,3].

AC-3 coding technology has been adopted by the Advanced Television Systems Committee (ATSC) as the audio service standard for High Definition Televison (HDTV) in the United States [4]. It has also found applications in consumer media (laserdisc, digital video disc) and direct satellite broadcast. At present, there are more than a dozen semiconductor manufacturers working on AC-3 decoder chips.

The goal of this paper is to present an overview of the AC-3 coding process, focusing on strategies for efficient implementation. The basic processing stages of AC-3 coders will be described, along with memory and complexity estimates.

2. THE AC-3 CODING SYSTEM

AC-3 is a flexible audio data compression technology capable of encoding a variety of audio channel formats into a single low-rate bitstream [5]. Eight channel configurations are supported, ranging from conventional mono or stereo to a surround format with six discrete channels (left, center, right, left surround, right surround and subwoofer). The AC-3 bitstream specification permits sample rates of either 48 kHz, 44.1 kHz, or 32 kHz, and supports data rates ranging from 32 kbps (kilobits-per-second) to 640 kbps.

The AC-3 encoder block diagram is shown in Figure 1 below. Encoding is done in the frequency domain, using a 512-point MDCT (modified discrete cosine transform) with 50% overlap [6]. In the event of transient signals, improved performance is achieved by using a block-switching technique, in which two 256-point transforms are computed in place of the 512-point transform. A floating-point conversion process breaks the transform coefficients into exponent / mantissa pairs. The mantissas are then quantized with a variable number of bits, based on a parametric bit allocation model.

The AC-3 bit allocation model uses principles

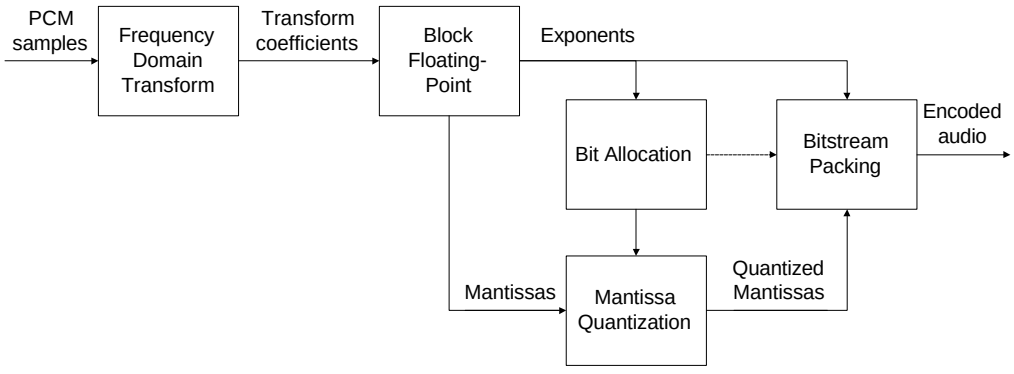


Figure 1: AC-3 encoder block diagram

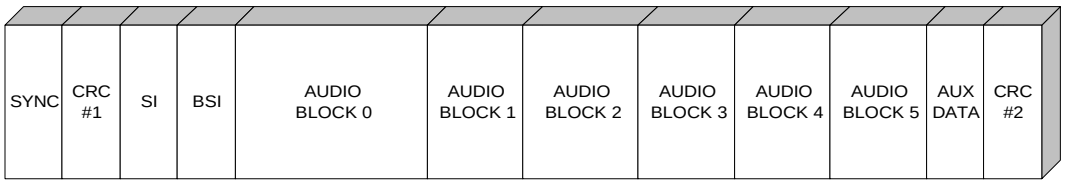


Figure 2: AC-3 frame structure

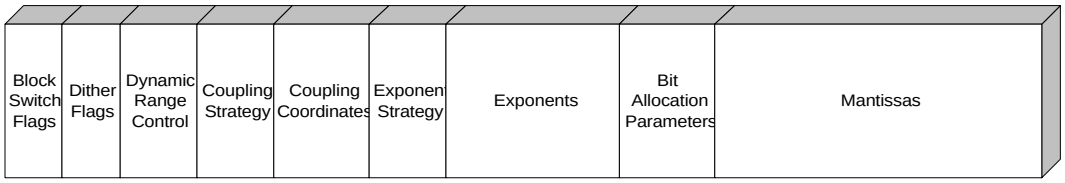


Figure 3: AC-3 audio block structure

of psychoacoustic masking to decide how many bits to provide for each mantissa in a given frequency band. Depending on the extent of masking, some mantissas may receive very few bits or even no bits at all. This reduces the number of bits needed to represent the source, at the expense of (inaudible) added noise.

Unlike some other coding systems, AC-3 does not pass the bit allocation results to the decoder in the bitstream. Rather, a parametric approach is taken, in which the encoder constructs its masking model based on the transform coefficient exponents and a few key signal-dependent parameters. These parameters are passed to the decoder in the bitstream, using far fewer bits than would be necessary to transmit the raw bit allocation values. At the decoder, the bit allocation is reconstructed based on the exponents and bit allocation parameters.

The coding efficiency of AC-3 improves as the number of source channels increases. This is due to two principle features: a global bit pool, and high-frequency coupling. The global bit pool technique allows the bit allocator to split the available bits among the audio channels on an as-needed basis. If one or more channels are inactive at a specific time instant, the remaining channels will receive more bits than they would if all channels were in high bit demand.

Coupling is used to further compress the necessary data rate for the high-frequency range of the audio spectrum. At high frequencies, the ear cannot detect individual cycles of an audio waveform, and instead responds only to its envelope. Coupling reduces the high-frequency components of correlated channels to a single coupling channel, and then generates additional side-chain data that describes the per-channel spectral envelope. Since coupling is a potential source of artifacts, its use is generally restricted to lower data rates.

2.1 AC-3 Bitstream Features

The AC-3 bitstream is composed of frames (see Figure 2), representing a constant time interval of 1536 PCM samples across all coded channels. Each frame has a fixed size, which depends only on sample rate and coded data rate. Also, each frame is an independent entity, sharing no data with previous frames other than the transform overlap inherent in the MDCT.

At the beginning of each AC-3 frame are the SI (Sync Information) and BSI (Bit Stream Information) fields. The SI and BSI fields describe the bitstream configuration, including sample rate, data rate, number of coded channels, and several other systems-level elements. There are also two CRC words per frame, one at the beginning and one at the end, that provide a means of error detection.

Within each frame are six audio blocks, each representing 256 PCM samples per coded channel (see Figure 3). The audio block contains the block switch flags, coupling coordinates, exponents, bit allocation parameters, and mantissas. Data sharing is allowed within a frame, such that information present in Block 0 may be reused in subsequent blocks.

An optional aux data field is located at the end of the frame. This field allows system designers to embed private control or status information into the AC-3 bitstream for system-wide transmission.

The AC-3 coder was designed to be a complete audio subsystem solution, incorporating many features not generally associated with low-bitrate coding. These include dynamic range compression suitable for consumer audio playback, dialog normalization, and downmixing of multichannel audio to a specified number of output channels. Dynamic range control words are embedded in the AC-3 stream and applied in the decoder, allowing different playback modes from a single source bitstream.

3. DECODER IMPLEMENTATION STRATEGY

A basic overview of the AC-3 decoding process is shown in Figure 4. In order to keep memory and decoder latency requirements as small as possible, each AC-3 frame is decoded in a series of nested loops.

The first step establishes frame alignment. This involves finding the AC-3 sync word, and then confirming that the CRC error detection words indicate no errors. Once frame sync is found, the BSI data is unpacked to determine important frame information such as the number of coded channels.

The next step is to unpack each of the six audio blocks. In order to minimize the memory requirements of the output PCM buffers, the audio blocks are unpacked one-at-a-time. At the end of each block period the PCM results are copied to the output buffers, which are generally double-buffered for direct interrupt access by the system D/A converters.

3.1 Decoder synchronization timing

Figure 5 shows the timing stages used for AC-3 decoder synchronization. By design, all AC-3 encoders will ensure that the first two audio blocks are completely contained within the first 5/8 of the frame. Also, the first CRC word covers only the first 5/8 of the frame. This helps reduce decoder latency, as the decoder can begin to unpack and process the first two blocks before the frame has been completely received.

Once the first 2/3 of the frame has been received, the decoder begins by checking the first CRC word for errors. If there are no errors, the decoder will decode Block 0 and copy the reconstructed PCM samples to the output buffer. At the end of the block period (5.33 msec for a 48 kHz sample rate system), the output interrupt handler starts outputting the Block 0 samples to the DAC, and Block 1 decoding may commence.

By the end of the Block 1 decoding period, the entire frame will have been received by the decoder and the second CRC word can be checked. As can be seen, this approach yields a total decoder latency equal to one block period plus the time to receive 2/3 of the input frame (27 msec assuming continuous input in a 48 kHz sample rate system).

```
AC-3 frame alignment / CRC check
Unpack BSI data
For block = 1 to 6
{
    Unpack fixed data
    For chan = 1 to # coded channels
    {
        Unpack exponents
        For band = 1 to # bands
        {
            Compute bit allocation
            Unpack mantissas
            Scale mantissas / undo coupling
            Denormalize mantissas by exponents
        }
        Compute partial inverse transform
        Downmix to appropriate output channel(s)
    }
    For chan = 1 to # output channels
    {
        Window / overlap-add with delay buffer
        Store samples in PCM output buffer
        Copy downmix buffer values to delay buffer
    }
}
```

Figure 4: AC-3 decoder pseudocode

3.2 Decoder input processing

The audio block processing may be divided into two distinct stages, referred to here as input and output processing. Input processing includes all bitstream unpacking and coded channel manipulation. Output processing refers primarily to the window and overlap-add stages of the inverse MDCT transform.

This distinction is made because the number of output channels generated by an AC-3 decoder does not necessarily match the number of channels encoded in the bitstream. By using a technique called downmixing, a decoder can accept a bitstream with any number of coded channels and produce an arbitrary number of output channels. Input processing is performed on a per-coded-channel basis, while output processing is performed on a per-output-channel basis.

Input processing begins when the decoder unpacks the fixed audio block data, which is a

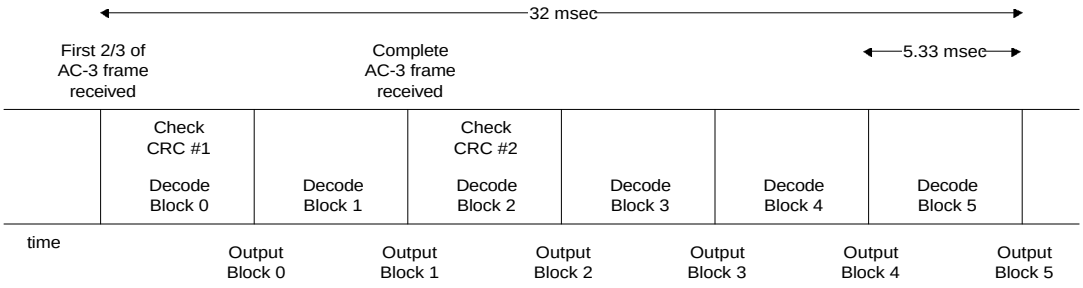


Figure 5: AC-3 decoder synchronization timing

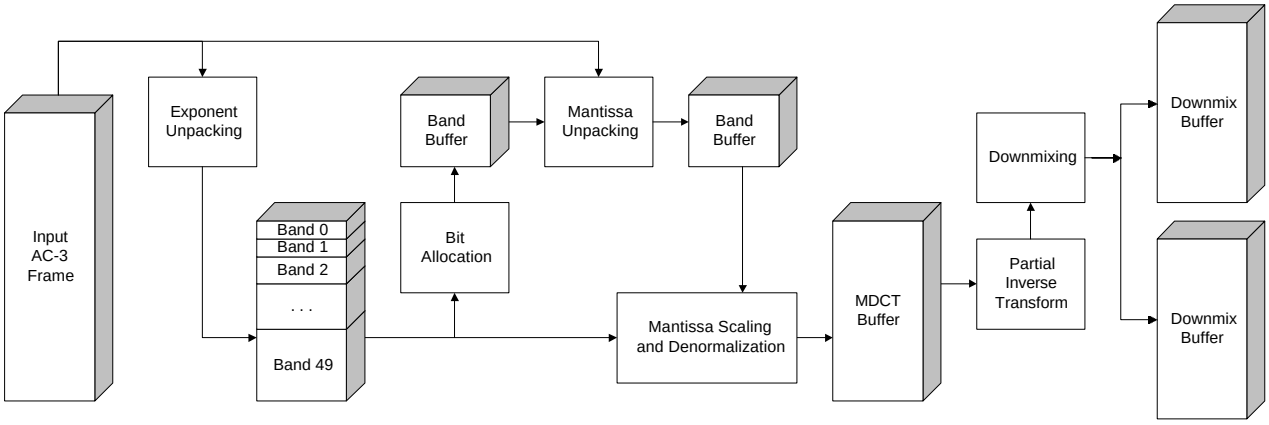


Figure 6: Decoder input processing

collection of parameters and flags located at the beginning of the audio block. This fixed data includes such items as block switch flags, coupling information, exponents, and bit allocation parameters. The term "fixed data" refers to the fact that the word sizes for these bitstream elements is known a priori, and does not require a bit allocation to be performed.

The exponents make up the single largest field in the fixed data region, as they include all exponents from each coded channel. Depending on the coding mode, there may be as many as one exponent per mantissa, up to 253 mantissas per channel. Rather than unpack all of these exponents to local memory, it is generally preferable to save pointers to the exponent fields, and unpack them as they are needed, one channel at a time.

Once the fixed data is unpacked, the decoder begins processing each coded channel (see Figure 6). First, the exponents for the given channel are unpacked from the input frame. A bit allocation calculation is then performed, which takes the exponents and bit allocation parameters and computes the word sizes for each packed mantissa. The mantissas are then unpacked from the input frame. The mantissas are scaled to provide appropriate dynamic range control (and to undo the coupling operation if needed), and then denormalized by the exponents. Finally, a partial inverse transform is computed, and the results are downmixed into the appropriate downmix buffers for subsequent output processing.

In the first of these steps, the exponents for the individual channel are unpacked into a 256-long buffer, called the "MDCT buffer". These exponents are then grouped into as many as 50 bands for bit allocation purposes. The number of exponents in each band increases toward higher audio frequencies, roughly following a logarithmic division that models psychoacoustic critical bands.

For each of these bit allocation bands, the exponents and bit allocation parameters are combined to generate a mantissa word size for each mantissa in that band. These word sizes are stored in a 24-long band buffer (the widest bit allocation band is made up of 24 frequency bins). Once the word sizes have been computed, the corresponding mantissas are unpacked from the input frame and stored in-place back into the band buffer. Finally, these mantissas are scaled and denormalized by the corresponding exponent, and written in-place back into the MDCT buffer. After all bands have been processed, and all mantissas unpacked, any remaining locations in the MDCT buffer are written with zeros.

At this point, a partial inverse transform is performed in-place in the MDCT buffer. The partial inverse transform is realized as a 128-point complex pre-multiplication stage, a 128-point complex inverse FFT, and a 128-point complex post-multiplication stage. The output of this processing is then downmixed into the appropriate downmix buffers.

Note that the downmixing could be done entirely in the frequency domain, prior to the partial inverse transform, if not for the use of block switching. In the case of block-switched transforms, a different partial inverse transform process is used, and the results of the two different transforms cannot be directly combined until just prior to the window stage.

As will be shown later, the input processing stage uses a relatively small amount of memory, as it is able to process the input stream one channel at a time, and even one band at a time. On the other hand, this is the most MIPS intensive part of the decoding process, as the bit allocation and unpacking routines alone account for half of the total decoding complexity.

3.3 Decoder output processing

Once the input processing is completed and the downmix buffers have been fully generated, the decoder can perform the output processing (see Figure 7). For each output channel, a downmix buffer and its corresponding 128-long half-block delay buffer are windowed and combined to produce 256 PCM output samples. These samples are then rounded to the DAC word width and copied to the output buffer. Once this is done, half of the downmix buffer is then copied to its

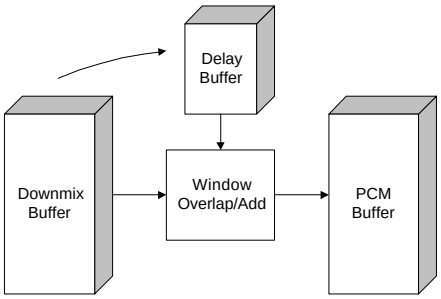


Figure 7: Decoder output processing

corresponding delay buffer, providing the 50% overlap information necessary for proper reconstruction of the next audio block.

In constrast with the input processing stage, decoder output processing uses very few MIPS but a relatively large amount of memory. For each output channel, the decoder requires a 256-long downmix buffer, a 128-long delay buffer, and a 512-long output buffer (assuming double-buffered output). Further, these buffers require reasonably large word widths in order to keep the wideband signal-to-noise ratio within professional audio standards. For 16-bit PCM output precision, the delay buffer should be at least 18 bits wide, and the downmix and MDCT buffers should be at least 20 bits wide. For more accurate PCM output, these numbers need to grow accordingly.

3.4 Decoder memory and MIPS summary

The total memory and MIPS requirements of an AC-3 decoder are summarized in the tables in Figure 8. The memory figures are grouped separately as RAM and ROM. In these tables, the parameter K is the number of coded channels in the input bitstream, while the parameter N is the number of output channels. These tables also assume that the input frame is single-buffered while the output PCM buffers are double-buffered, and that the AC-3 bitstream is a six-channel program coded at 384 kbps with a 48 kHz sample rate (the HDTV standard configuration).

Looking at the memory figures, it is clear that a two-channel decoder (i.e. a decoder that can downmix any number of input channels to two output channels) requires significantly less RAM than a six-channel decoder. Since RAM accounts for a considerable part of total chip cost, this indicates that two-channel decoders can be produced at substantially lower cost than full-featured decoders. Two-channel decoders can provide a low-cost entry point to multichannel audio, as users can downmix six-channel program material at first, and upgrade to discrete reproduction in the future.

Regarding MIPS, the computational complexity is roughly equal for either a two-channel or six-channel decoder. This indicates that two-channel decoder chip designs may be extended to full six-channel devices without significant changes to the processor core.

Both the MIPS figures and the program code ROM size were determined assuming the Zoran ZR38001 DSP processor instruction set. This chip is a general purpose processor, which provides a single-cycle barrel shifter and a four-cycle FFT butterfly in addition to the usual set of DSP features. By way of comparison, a Motorola DSP56002 requires roughly 45 MIPS to perform six-channel AC-3 decoding.

4. ENCODER DESIGN CONSIDERATIONS

Unlike decoding, AC-3 encoding is not a fixed procedure, but rather requires sophisticated strategy analysis for optimal performance. Nearly all of the parameters passed to the decoder in the bitstream must be adaptively determined by the encoder based on signal characteristics.

For example, a transient detection routine is used to determine whether or not to block switch the MDCT transform size. Further processing examines the frequency domain exponents and other fixed data in order to decide whether to retransmit different values in each audio block or to share common values between blocks. Finally, a sophisticated bit allocation model is used to determine the bit allocation parameters for best subjective sound quality.

For this reason it is impractical to specify an upper bound on the complexity of an AC-3 encoder.

RAM Data Structure	Words	# Bits
Input AC-3 Frame	768	16
Audio block fixed data	≈ 384	16
MDCT Buffer	256	≥ 20
Downmix Buffers	N * 256	≥ 20
Delay Buffers	N * 128	≥ 18
PCM Buffers	N * 512	≥ 16

ROM Data Structure	Words	# Bits
Transform lookup tables	768	≥ 20
Other lookup tables	≈ 1250	16
Program code	≈ 3500	32

Processing Task	MIPS	% of Total
Frame alignment / CRC check	1.3	5 %
Unpack BSI	0.2	1 %
Unpack fixed data	0.9	3 %
Unpack exponents	K * 0.5	9 %
Compute bit allocation	K * 1.6	30 %
Unpack mantissas	K * 1.1	20 %
Scale mantissas & denormalize	K * 0.3	6 %
Partial inverse transform	K * 0.9	20 %
Downmixing	K * 0.1	2 %
Window / overlap-add	N * 0.2	4 %

6-channel decoder: 6.6K RAM, 5.4K ROM, 27.3 MIPS
2-channel decoder: 3.1K RAM, 5.4K ROM, 26.5 MIPS

Figure 8: Decoder memory / MIPS summary

Nonetheless, there are a few points that can be made that apply to almost any AC-3 encoder design.

In general, the encoder must consider all channels in all blocks in a frame in order to determine the optimal bit allocation parameters and fixed data reuse strategy. Whereas the decoder was able to perform inverse transforms one channel at a time (and one block at a time), the encoder will normally have to perform all transforms for all channels in all six blocks prior to doing any coding. As a result, the memory requirements of an encoder will be several times that of an AC-3 decoder.

Further, the encoder bit allocation routine is generally an iterative process, as the coder needs to allocate as many bits as possible without exceeding the available data rate. Finally, the encoder is tasked with generating several control values that are simply used as is by the decoder, such as coupling parameters and dynamic range compression words.

Given that the strategy analysis for encoding is a complex process, the MIPS requirements of encoders will be significantly higher than decoders. Current encoder designs require between two and six times the MIPS of their associated decoders, depending on their degree of sophistication. It is worth noting that for 2-channel data rates above 192 kbps, or 6-channel data rates above 384 kbps, only moderate encoder complexity is necessary for near-transparent audio performance.

Finally, since the encoder word width affects the dynamic range of the decoder, extra care must be taken in computing the forward transform. Current encoders use either floating-point or 24-bit fixed-point processors to ensure accurate PCM to at least 18 bits.

5. CONCLUSION

The AC-3 audio data compression system was described, and an overview of encoding and decoding strategies was presented. Several techniques for creating efficient decoder implementations were discussed in some detail. It was shown that an AC-3 decoder may be realized using a series of tightly-nested loops.

The decoder implementation described here had the useful property that the amount of required RAM was roughly proportional to the number of output channels. Since decoders have the ability to downmix to an arbitrary number of output channels, this means that lower cost decoders can be built that are compatible with multichannel bitstreams.

6. ACKNOWLEDGEMENTS

The author would like to acknowledge the contributions of several members of the engineering team at Dolby Laboratories for their contributions to the AC-3 specification, including Craig Todd, Louis Fielder, Grant Davidson, Mark Davis, Matt Fellers, and Marina Bosi.

6. REFERENCES

[1] G. Davidson, L. Fielder, and M. Antill, "Low-Complexity Transform Coder for Satellite Link Applications," AES 89th Convention, Los Angeles, Sept. 1990, Preprint 2966.

[2] L. Fielder and G. Davidson, "AC-2: A Family of Low-Complexity Transform-Based Music Coders," 1991 AES Workshop on Digital Audio, London, Oct. 1991.

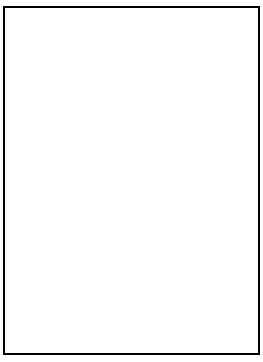
[3] G. Stoll and Y. F. Dehery, "High-Quality Audio Bit-Rate Reduction System Family for Different Applications," Proc. of IEEE Intl. Conf. on Comm., Atlanta, pp. 937-941, April 1990.

[4] ATSC A/52, "Digital Audio Compression (AC-3) Standard," United States Advanced Television Systems Committee.

[5] C. Todd, G. Davidson, M. Davis, L. Fielder, B. Link, and S.Vernon, "AC-3: Flexible Perceptual Coding for Audio Transmission and Storage," AES 96th Convention, Amsterdam, Feb. 1994, Preprint 3796.

[6] J. Princen and A. Bradley, "Analysis / Synthesis Filter Bank Design Based on Time Domain Aliasing Cancellation," IEEE Trans. ASSP, vol. ASSP-34, no. 5, pp. 1153-1161, Oct. 1986.

BIOGRAPHY



Steve Vernon received a B.S.E.E. degree from the University of Southern California in 1986 and an M.S.E.E degree from Stanford University in 1987. His education was in electrical engineering and mathematics, with an emphasis in statistical signal processing. He is currently employed as a senior project engineer at Dolby Laboratories, where he has been working on low-bitrate audio coding algorithms and implementations since 1991.