

DIGITAL AUDIO GETS AN AUDITION

PART TWO: LOSSY COMPRESSION

LOSSLESS COMPRESSION CAN SHRINK FILE SIZES DOWN A FAIR AMOUNT, BUT FOR SERIOUS WEIGHT LOSS, YOU NEED TO PERMANENTLY DISCARD SOME OF THE DATA. FIND OUT HOW THE LOSSY CODECS WORK AND WHETHER YOU CAN HEAR THE DIFFERENCES BETWEEN THEM.

WITH ALL THE ADVANTAGES of lossless compression that part one of this article series cites (Jan 4, 2001), what then is the role of lossy compression? Consider a lossy codec's ability to compress audio files not to half their size but to a guaranteed 1/12 (MP3) or

even 1/24 (WMA) compression ratio, with little-to-no degradation perceived in playback quality. Lossy-format conversion is necessary to cost-effectively listen to audio on a portable player that employs semiconductor storage, stream it to Internet listeners from a Web server or across a LAN, or digitally broadcast it. And if your audio device can directly read and decode MP3 or other lossy-codec-format files burned on a CD-R, you give users the ability to store as much as a few dozen hours of music on one disc.

Compressed file sizes aren't meaningful comparison points for lossy-compression algorithms, because the objec-

tive is to always encode to a specific bit rate (**Table 1**). The time it takes to encode to that bit rate for a given μ P type and speed differs from one algorithm to another, though, as does the quality at that bit rate. Quality is a user-, environment-, and application-dependent metric. Last year, my neighbor swore that he was unable to hear any differences when he listened to an original audio CD, a 128-kbps MP3 stream, a 64-kbps WMA stream, and a 16-kbps RealAudio stream through his PC's low-cost speakers. However, when I brought him to my house and played the same files on my PC's higher-end speaker set, he immediately under-

<i>At a glance</i>	88
<i>The never-ending sonic story</i>	88
<i>Acronyms</i>	91
<i>Music mysteries</i>	94
<i>For more information</i>	104

stood the need for those files “that take so long to download.” I generally resist the urge, therefore, to make quality comments on various codecs, though if one stands out as sounding particularly good or bad to my less-than-golden ears, I’ll mention it.

Even my “slow” 533-MHz CPU can rapidly encode and decode a 30-sec test-tone clip. Therefore, for the performance-analysis portions of this lossy-compression project, I employ the same 19 songs used to evaluate lossless-compression algorithms (Table 2, part one). In addition to measuring performance, I also hope to reveal the presence of various lossy-compression techniques and their artifacts. The list of things I am looking for include:

- lowpass filtering, or removal of all audio information above a certain frequency;
- stereo-to-mono conversion of the original two audio channels, com-

AT A GLANCE

▷ If double-digit compression is your goal, lossy codecs are the only way to go.

▷ Even if lossy compression artifacts exist, they might be inaudible.

▷ Song clips of similar duration from different music genres produce different compression results.

▷ Codec developers must often balance encoding speed and quality; quality depends on audio-source characteristics.

▷ Ensure that in the best case, added noise is inaudible, and in the worst case, it doesn’t obscure meaningful audio content.

▷ Head to the Web for even more analysis results, as well as the tools to let you do your own tests.

- completely or above a certain frequency;
- phase collapse, or elimination of phase differences between the two channels, completely or above a certain frequency;
- frequency masking, in which a loud tone masks lower-volume information in nearby frequencies;
- temporal masking, in which a loud tone masks lower-volume information that both precedes and follows the masking tone in time; and
- echo, or the insertion of unwanted audio information both before and after a sharp transient, such as a percussion-instrument sound.

MORE ON TEST TONES

The white- and pink-noise clips I use in the lossless-compression study are also useful in my lossy-compression work. Equal-intensity noise channels, converted to a frequency-domain display via a spectrum analyzer, enable me to identify

THE NEVER-ENDING SONIC STORY

My digital audio analysis work is by no means done, so periodically visit the Web-site addendum to this article series (www.ednmag.com/ednmag/extras/01csaddendum.asp) for any updates. Because the lossless-compression results in part one of this study are so similar, I don’t plan to evaluate any of the other lossless codecs. (If any of you would like to do so, I’d be happy to post your results.) My efforts will focus on lossy compression. Looking first at MP3, I’d like to recompress some of my test tones using VBR encoding to see whether VBR significantly reduces the presence and magnitude of artifacts.

Other versions of the Fraunhofer encoder might provide additional flexibility to enable, disable, and otherwise adjust the operation of various compression options. And although Fraunhofer is the most popular MP3 encoder, it’s not the only game in town. RealNetworks uses the Xing encoder. QDesign also sells one. And a number of independently developed encoders exist: Blade,

Gogo, Lame, and Radium, just to name a few. The choice of an MP3 decoder might even affect the results, as University of Essex doctoral candidate David Robinson’s recent study suggests (<http://privatewww.essex.ac.uk/~djmrob/mp3decoders>).

I’d like to look more closely for evidence of pre-echo in MP3 and WMA, as well as phase collapse and temporal masking in all the codecs. I’d also like to string together a series of test tones to see whether I can replicate the behavior PCABX Web-site audio consultant Army Krueger found when he evaluated WMA (see sidebar “Music mysteries”). And I’d like to perform critical listening tests of the lossy-compressed music tracks and EBU SQAM test tones; I strongly suspect that RealAudio isn’t the only codec making audible alterations.

I haven’t yet begun to evaluate a plethora of additional codecs. First on the list is AAC; Fraunhofer is creating batch-mode-capable versions of their v3 encoder and decoder for me, and Dolby Labs has already sup-

plied me with its AAC professional encoder/decoder software. Sony’s dominant position in consumer electronics is motivating me to look at 132-kbps ATRAC3. (The older 292-kbps ATRAC is of less interest.) Sony uses ATRAC3 in the Music Clip and other solid-state player/recorders; their latest MiniDisc Long Play units also support it.

A few of the other algorithms in Table 1 tweaking my intellectual curiosity include open-source perceptual coders MPEGplus and Ogg Vorbis; vector-quantization pioneer TwinVQ; and Qdesign’s codec, which employs parametric encoding techniques and is used in Apple’s QuickTime. I’ll probably skip ePAC, though; Vedalabs indicates that it’s fallen out of favor in consumer electronics, and iBiquity Digital’s compression derives from the original PAC algorithm (Reference A).

If I determine that High Criteria’s Total Recorder doesn’t alter amplitude or otherwise mangle the audio’s characteristics when intercepting and capturing the digital bit stream on

its way to a PC sound card, I’ll use it to convert lossy-compressed formats back to WAV if the formats’ players won’t natively accomplish this task. Otherwise, I’ll route the players’ signals to the digital outputs of the PC sound card, capture them with a DAT deck, then send them digitally back into the PC and capture them as a WAV in Sound Forge or Cool Edit Pro. Ego-Sys’ USB-based Waveterminal U2A is ideal for this task, because I don’t need greater than 16-bit or 44.1-kHz-sampled audio and because it doesn’t require me to open up the PC and swap out sound cards. User feedback suggests that the U2A is a more robust performer than Opcode Systems’ Sonicport Optical (Reference B).

REFERENCES

A. Dipert, Brian, “Digital-radio combatants make peace, agree on codec,” *EDN*, Nov 9, 2000, pg 32.

B. Dipert, Brian, “The high-end PC looks for a home,” *EDN*, Nov 24, 1999, pg 145.

any lowpass, bandpass, or highpass filtering that a codec performs. Channels of differing intensities provide additional details—specifically if the encoder is converting the source material from stereo to mono within certain frequency ranges. The human auditory system groups its detection of incoming audio information into a number of critical frequency bands, with most of the bands residing at less than 5 kHz (references 1 and 2). Note that in Table 2, the bands' widths increase as the corresponding center frequencies rise. A structure in the inner ear called the organ of Corti translates incoming audio

waves into nerve impulses. Its basal-membrane width, thickness, stiffness, and hair-cell clustering define the critical band-frequency ranges and endpoints.

What better way to continue my test-clip development, then, than by combining tones at the midpoints of each critical band? Syntrillium Software's Cool Edit Pro, which costs roughly the same as Sonic Foundry's Sound Forge, includes a 64-track mixer that I use extensively. Cool Edit Pro enables me to create and combine precisely defined audio tones, as well as generate white, pink, and brown noise. Its time-based (oscillo-

scope) and frequency-based (spectrum-analyzer) output displays are more informative and have more robust features than those in Sound Forge.

All of my critical-band-derived sound clips have one channel 180° out of phase from the other, to give the encoder one more challenge to surmount and to enable me to look for phase collapse in the subsequent decoding. As with the pink and white-noise clips, I created two versions of each file; one with both channels at equivalent amplitude, and the other with the left channel 20 dB "louder" than the right.

TABLE 1—REPRESENTATIVE LOSSY-COMPRESSION ALGORITHMS, SELECTED CODEC SOFTWARE,

Codec	URL	Algorithm	Selected software
AAC	www.aac-audio.com; www.csel.it/mpeg; www.iis.fhg.de/amm/techinf/aac; www.mpeg.org	Encoder	Fraunhofer command line encoder (v3.0)
		Decoder	Fraunhofer command line decoder (v3.0)
ATELP	www.softsound.com/ATELP.html	Encoder Decoder	Softsound ATELP audio-compression manager Softsound ATELP audio-compression manager
ATRAC	www.minidisc.org	Encoder Decoder	Kenwood MD-203 via PC sound-card digital output (v4.5) Kenwood MD-203 via PC sound-card digital input (v4.5)
ATRAC3	www.minidisc.org	Encoder Decoder	RealNetworks RealJukebox Plus v2 (beta) RealNetworks RealJukebox Plus v2 (beta)
Dolby Digital (AC-3)	www.dolby.com/digital	Encoder Decoder	Sonic Foundry Soft Encode 5.1 (v1.0 build 19) Sonic Foundry Soft Encode 5.1 (v1.0 build 19)
DTS	www.dtsonline.com	Encoder Decoder	Minnetonka Software SurCode CD Professional for DTS (v1.0.9) Minnetonka Software SurCode CD Professional for DTS (v1.0.9)
ePAC	www.lucent.com/ldr; www.vedalabs.com	Encoder Decoder	VedaLabs AudioVeda (v1a build 417) VedaLabs AudioVeda (v1a build 417)
Indeo	www.ligos.com	Encoder Decoder	Ligos Technology Indeo Media Kit Ligos Technology Indeo Media Kit
MP3	www.csel.it/mpeg; www.iis.fhg.de/amm/techinf/basics.html; www.mpeg.org	Encoder	Sonic Foundry Sound Forge 4.5h (Fraunhofer built 224)
		Decoder	Sonic Foundry Sound Forge 4.5h (Fraunhofer built 224)
MPEG(plus)	www.stud.uni-hannover.de/user/73884/audiocoder.html	Encoder	MPEG(plus) encoder (v1.7.8)
		Decoder	MPEG(plus) decoder (v1.7.6)
Ogg Vorbis	www.vorbis.com	Encoder	Ogg Vorbis Encoder (dated 8/15/00)
		Decoder	Ogg Vorbis plugin for Winamp (v0.1)
Qdesign	www.qdesign.com	Encoder	Qdesign MVP (v1.2)
		Decoder	Qdesign MVP (v1.2)
RealAudio	www.real.com	Encoder	Sonic Foundry Sound Forge 4.5h (RealAudio G2)
		Decoder	RealNetworks RealJukebox Plus v2
TAC	http://kk-research.hypermart.net	Encoder	K&K Research Music Publisher 02 (beta v22)
		Decoder	K&K Research Plugin for Winamp (v2026a)
TwinVQ	http://sound.splab.ecl.ntt.co.jp/twinvq-e; www.vqf.com; www.yamaha-xg.com/soundvq	Encoder	Yamaha TwinVQ encoder (v2.60 beta2)
		Decoder	Yamaha TwinVQ decoder (v2.52 beta1)
Windows Media Audio	www.microsoft.com/windows/ windowsmedia	Encoder	Sonic Foundry Sound Forge 4.5h (WMA v7)
		Decoder	Microsoft command line decoder (WMA v7)

Note: Some software decoders (that is, players) also require TotalRecorder or a DAT deck connected to the PC sound-card digital output to create WAV files.

To generate each file, I first created a number of 32-bit per-sample and per-channel, 44.1-kHz-sampled, single-tone sources, then mixed them together at 32-bit resolution in Cool Edit and attenuated the result to the desired maximum amplitude. I then needed to convert them to 16-bit equivalents. After discussions with both Syntrillium Software and with audio consultant Arny Krueger, I chose the following sample-type-conversion settings:

- dither on,
- 0.5-bit dither depth,
- no noise shaping.

- triangular probability-distribution function, and

Next, to test for frequency masking, I regenerated my critical-band midpoint mix, but this time mixed in 50 additional coincident tones, half of them at the one-quarter point across each critical band and the other half at the three-quarter point. One- and three-quarter-point test tones were 20 dB quieter than their midpoint neighbors. To test for temporal masking, I first determined that the pre-tone masking duration extended no further than 50 msec ahead of the masking tone, and the post-tone dura-

tion extended no more than 200 msec beyond the masking tone (**Reference 3**). Therefore, I again created my 30-sec midpoint tone combination. But this time, I preceded it by 50 msec of the same tonal mix, but 20 dB quieter, and followed it with 200 msec of the same 20-dB-quieter mix. Finally, to find pre- and post-echo noise around sharp audio transients, I turned to three tracks on the EBU SQAM disc: track 27 (castanets), track 32 (triangle) and track 35 (glockenspiel).

Although I created the noise and test tones in Cool Edit Pro, I switched to Sound Forge for the lossy-compression process because of its more comprehensive format support. Sound Forge version 4.5h can encode MP3, RealAudio G2, and Windows Media Audio 7 files. It can also decode MP3 files back to WAV, but licensing restrictions preclude it from supporting RealAudio and WMA decoding, forcing me to rely on RealNetworks Real Jukebox and Microsoft's command-line decoder, respectively. By encoding from the same WAV file to each of the three formats within an otherwise-identical software environment, I hope to be most accurately measuring the speed of the encoding algorithm, with other system overheads canceled out.

AND AVAILABLE ALTERNATIVES

Other available options

Astrid Encoder, Dolby Labs AAC professional software v1.1, FAAC (Freeware Advanced Audio Coder), Homeboy Encoder, Liquid Audio Liquifier, Psytel beta2 (Dec 29, 2000) a2b Music Player, Astrid Decoder, Dolby Labs AAC professional software v1.1, FAAC (Freeware Advanced Audio Coder), Homeboy plugin for Winamp, Liquid Audio Player, Lorentz Istvan's plugin for Winamp, Psytel beta2 (Nov 29, 2000)

None
None

Sharp MD-MT15 via PC sound-card digital output

None
None

**Liquid Audio Liquifier, Minnetonka Software SurCode for Dolby Digital
Liquid Audio Liquifier, Minnetonka Software SurCode for Dolby Digital**

None
None

**Celestial Technology AudioLib
Celestial Technology AudioLib**

None
None

Blade, Gogo, MusicMatch Jukebox (Fraunhofer), LAME, Qdesign, Radium, RealNetworks RealJukebox (Xing), Syntrillium Cool Edit Pro (Fraunhofer)

Liquid Audio Player, Microsoft Windows Media Player (Fraunhofer), MPG123, MusicMatch Jukebox (Fraunhofer), Qdesign, QuickTime, RealNetworks RealJukebox and RealPlayer (Xing), Syntrillium Cool Edit Pro (Fraunhofer), Winamp 2.22 plugin (Fraunhofer), Winamp 2.71 plugin (Nitrane)

None
MPEG plus plugin for Winamp (v1.7.8)

None
None

**Qdesign Music Codec 2 Professional Edition (in conjunction with QuickTime)
Qdesign Music Codec 2 Professional Edition (in conjunction with QuickTime)**

**RealNetworks RealJukebox and RealSlideshow, Syntrillium Cool Edit Pro
RealNetworks RealPlayer and RealSlideshow, Streambox RA2WAV/Ripper/VCR**

None
None

**NTT TwinVQ encoder
NTT TwinVQ decoder, plugin for Winamp**

**Microsoft command line encoder, Microsoft Windows Media Encoder, MusicMatch Jukebox, Nullsoft Winamp
Microsoft Windows Media Player, MusicMatch Jukebox, Nullsoft Winamp**

THE NEED FOR SPEED

I ran 19 song clips and 13 test tones through MP3 encoding 10 times, with each iteration a combination of one of five compressed target bit rates, in conjunction with either a quality- or performance-optimized encoder configuration. I ran them through WMA encoding four times and through RealAudio en-

ACRONYMS

AAC: Advanced Audio Coding

CBR: constant bit rate

CD-R: CD-recordable

codec: coder/decoder, also sometimes used to define a single-chip A/D-plus-D/A converter

DAT: digital audio tape

EBU: European Broadcast Union

(e)PAC: (enhanced) Perceptual Audio Coder

MMX: multimedia extension

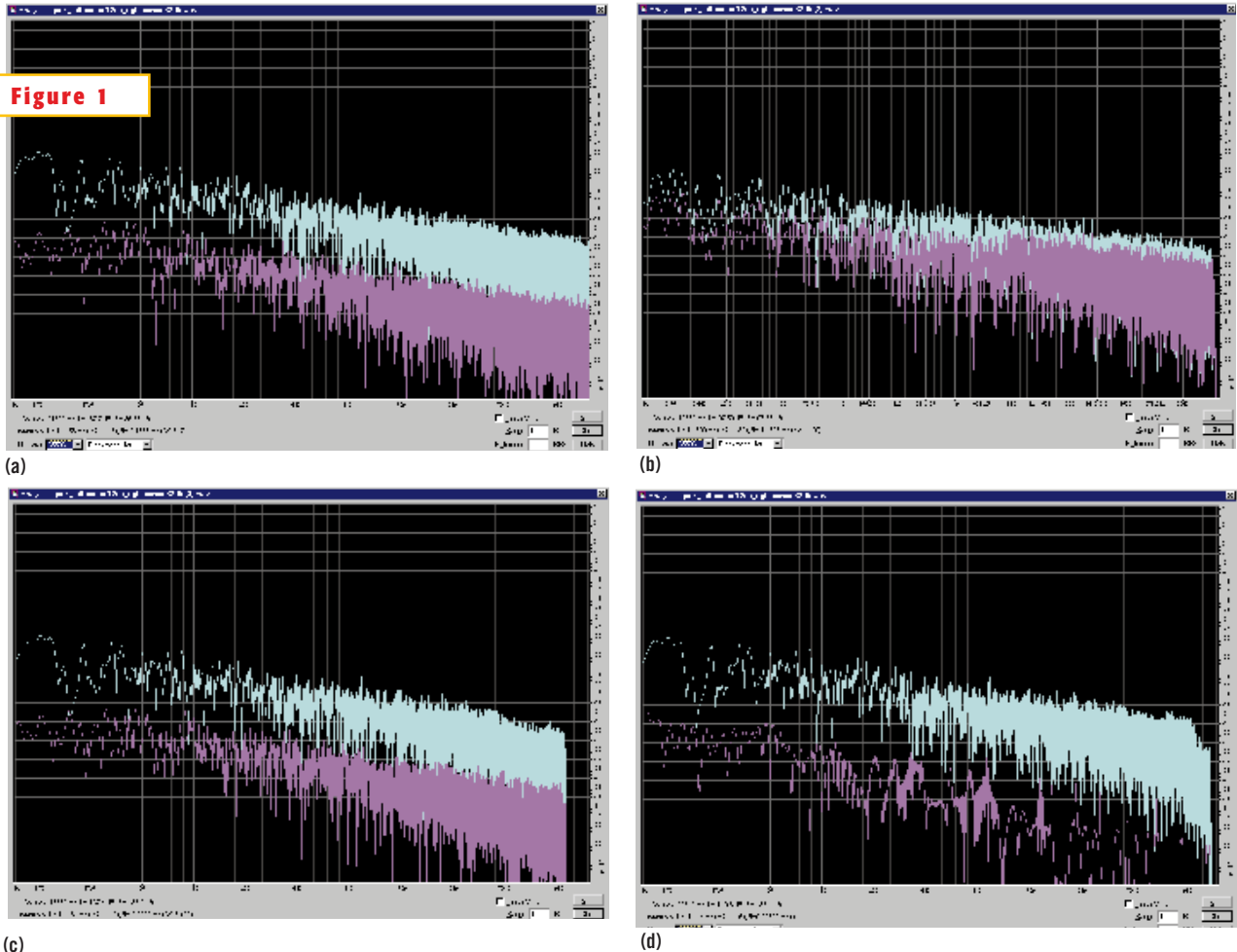
SQAM: Sound Quality Assessment Material

TwinVQ: Transform-Domain Weighted Interleaved Vector Quantization

VBR: variable bit rate

WMA: Windows Media Audio

Figure 1



(a) (b) (c) (d) Spectrum-analyzer (frequency-based) displays show test clip 2 in original WAV (a) and lossy-compressed MP3 (b), RealAudio (c), and Windows Media Audio (d) formats.

TABLE 3—LOSSY-COMPRESSION RESULTS

Codec	Bit rate (kbps)	Optimization	Classical (instrumental)		Hard rock (with vocals)	
			Compression time (hours:minutes:seconds)	Decompression time (hours:minutes:seconds)	Compression time (hours:minutes:seconds)	Decompression time (hours:minutes:seconds)
MP3 (Fraunhofer)	64 (CBR)	Highest Quality	0:01:02	0:00:14	0:00:30	0:00:07
	64 (CBR)	Fastest Encode	0:00:52	0:00:13	0:00:25	0:00:08
	96 (CBR)	Highest Quality	0:19:24	0:00:24	0:08:40	0:00:11
	96 (CBR)	Fastest Encode	0:00:59	0:00:23	0:00:29	0:00:10
	128 (CBR)	Highest Quality	0:20:06	0:00:28	0:10:19	0:00:13
	128 (CBR)	Fastest Encode	0:01:02	0:00:27	0:00:30	0:00:13
	160 (CBR)	Highest Quality	0:23:18	0:00:31	0:10:34	0:00:14
	160 (CBR)	Fastest Encode	0:01:03	0:00:30	0:00:30	0:00:15
	192 (CBR)	Highest Quality	0:15:13	0:00:32	0:05:30	0:00:14
	192 (CBR)	Fastest Encode	0:01:04	0:00:31	0:00:32	0:00:14
RealAudio G2	64	N/A	0:00:48	0:01:00	0:00:24	0:00:27
	96	N/A	0:00:52	0:01:09	0:00:26	0:00:32
Windows Media Audio v7	64	N/A	0:01:02	0:00:28	0:00:30	0:00:13
	96	N/A	0:01:06	0:00:31	0:00:32	0:00:14
	128	N/A	0:01:05	0:00:34	0:00:32	0:00:15
	160	N/A	0:01:05	0:00:36	0:00:32	0:00:17

Note: Table 3 in the version of this article on EDN's Web site contains results for all 19 music genres.

coding twice. I also ran each resultant MP3 file through the decoder built into Sound Forge. That's 512 total encoder runs, 320 decoder runs, a whole lot of mouse clicks, and a whole lot of time spent staring at a computer monitor. Fortunately, Sound Forge supports batch-mode capability and gives you the option to create a log file that captures time to encode and decode.

For Windows Media Audio 7 decoding, I used a DOS-command-line utility that Microsoft supplied me. I was unable to figure out how to capture to a file the time-to-decode message displayed on-screen, so I manually logged each displayed value as the batch file ran. RealAudio G2 decoding uses RealJukebox's convert-to-WAV capability, and I referenced the "created" and "modified" time/date stamps, which are viewable through Windows Explorer, to determine decode time.

In analyzing the encode- and decode-performance-testing results, several trends are evident (Table 3). (This article on EDN's Web site contains results for all 19 music genres.) Look at the disparity in encoding times between MP3's "fastest encode" and "highest quality" settings, even at the same bit rate; 64 kbps deviates from the general trend, but a good reason for this anomaly exists. The MP3 encoder, when set to 64 kbps, down-samples the original 44.1-kHz material to 22.05 kHz and severely lowpass filters out

TABLE 2—CRITICAL BANDS AND FREQUENCY POINTS WITHIN THOSE BANDS

Band	Low (Hz)	High (Hz)	Width (Hz)	quarter point (Hz)	midpoint (Hz)	three-quarter point (Hz)
0	0	100	100	25	50	75
1	100	200	100	125	150	175
2	200	300	100	225	250	275
3	300	400	100	325	350	375
4	400	510	110	427.5	455	482.5
5	510	630	120	540	570	600
6	630	770	140	665	700	735
7	770	920	150	807.5	845	882.5
8	920	1080	160	960	1000	1040
9	1080	1270	190	1127.5	1175	1222.5
10	1270	1480	210	1322.5	1375	1427.5
11	1480	1720	240	1540	1600	1660
12	1720	2000	280	1790	1860	1930
13	2000	2320	320	2080	2160	2240
14	2320	2700	380	2415	2510	2605
15	2700	3150	450	2812.5	2925	3037.5
16	3150	3700	550	3287.5	3425	3562.5
17	3700	4400	700	3875	4050	4225
18	4400	5300	900	4625	4850	5075
19	5300	6400	1100	5575	5850	6125
20	6400	7700	1300	6725	7050	7375
21	7700	9500	1800	8150	8600	9050
22	9500	12000	2500	10125	10750	11375
23	12000	15500	3500	12875	13750	14625
24	15500	22050	6550	17137.5	18775	20412.5

the upper portion of the frequency spectrum. These alterations ensure that the encoder has less source data to work with and at least partially explain why the "highest quality" and "fastest encode" results are more alike at this bit rate.

Also, notice that MP3 "highest quality" encoding to 192 kbps is actually faster than encoding to 160 kbps. Although the encoder is generating more compressed data at the higher bit rate, this trade-off gives an overall performance benefit: The encoder needs not work so hard at 192 kbps to squeeze the data down while maintaining quality. This result also suggests that, thanks to a fast hard drive and DRAM, the additional system overhead that my PC needs to store the larger compressed bit stream is an insignificant factor in the results. I was actually measuring the encode speed.

Table 2 in part one of this article lists the songs I used for each music genre, their duration, and their uncompressed WAV sizes. Match this information with that of Table 3 in this article, and you'll find that, as with lossless compression, songs of similar duration but different genres sometimes have significantly different encoding delays. This trend indicates that some types of music are "harder" to compress to a given bit rate and quality than others, and it validates the

hunch that prompted me to do all this work in the first place! The results make sense: Compare a techno track to spoken word, for example, and you'll find that the techno track has a broader meaningful frequency spectrum, increased high-frequency content, greater channel-to-channel variation in both amplitude and phase, and more abrupt transients.

Evaluate WMA against MP3, particularly in the context of the quality results that follow, and WMA will probably impress you. As a general rule (with a few exceptions), the WMA encoder performance approximates that of the MP3 encoder set to "fastest encode," while its quality at least matches (and, at lower bit rates, exceeds) that of MP3 files created using the "highest quality" setting. RealAudio's encoder speed is approximately the same as that of WMA and MP3 set to "fastest encode," but the quality news isn't so good. On both test tones and music tracks, RealAudio files consistently sounded the worst and contained the largest number of lossy compression artifacts.

And what about decoders? In all three cases, their speed scaled with the bit rate of the file they were decoding. (More bits to decode means a slower decoding speed, all other factors being equal.) At 64 kbps, MP3 decoding runs much faster

Notes

Compression and decompression times reported by Sound Forge's batch-mode log file

Compression times reported by Sound Forge's batch-mode log file; decompression times determined via Windows Explorer "time/date created" and "time/date modified" file information

Compression times reported by Sound Forge's batch-mode log file; decompression times reported by Microsoft's decoder utility

than the other two decoders, but remember that the encoder had previously halved the sample rate, halving the size of the resulting decoded WAV file and giving the MP3 decoder a significant

built-in speed advantage. At greater than 64 kbps, MP3 and WMA decoder speeds were comparable. Poor RealAudio, though, was consistently slower than its peers, roughly twice as slow on average.

Be careful when drawing definitive conclusions here. I used three decoding-software packages, so some of these differences may be the result of factors other than the decoding algorithms themselves.

MUSIC MYSTERIES

Microsoft steadfastly refuses to reveal the implementation details behind its WMA compression algorithm. Reviewers generally evaluate this codec as having quite good quality, particularly notable at low bit rates, and it uncharacteristically also delivers very fast encoding performance. With no official word from the company, audio aficionados are doing their best to figure out how WMA works its compression magic.

Reference 2 suggests that Microsoft might be employing a variant of the vector-quantization technique that a lossy codec called TwinVQ uses. The TwinVQ encoder and decoder both contain a prefabricated set of data coefficients called a codebook that represents what the algorithm's developers believe are the most common sets of per-frame frequency combinations in audio. The TwinVQ encoder, after completing a time-to-frequency transformation and subsequent compression of each frame of audio data, finds the closest match in its codebook, and instead of sending the actual coefficient data, it sends the codebook index. The decoder uses this index to pull the matching data approximation from its codebook, which it then outputs.

Because each codebook index (analogous to a book page, paragraph, and line-number combination) is much smaller than the actual data, vector quantization can produce impressive reductions in file sizes. But such reduction comes with a trade-off. If the codec developer poorly constructs the codebook or if the encoder doesn't do a good job of matching the actual data to a code-

book entry, the compressed audio can sound terrible. Therefore, several variations on the basic vector-quantization technique are possible. Instead of using a preassembled codebook, the encoder might create it on the fly, based on the characteristics of the audio it's compressing.

The advantage of on-the-fly custom-codebook creation is that you're more likely to get good matches between data samples and codebook entries. But now you have to transmit the codebook along with the compressed audio, because the decoder won't already have it. The bigger (and theoretically the better the quality of) the codebook, the larger the compressed bit stream that the encoder creates and the less efficient the compression results. Also, on-the-fly codebook creation is a computing-intensive operation, which leads to slow encoding speeds. As an interim step, therefore, the vector-quantization algorithm may rely mostly on a prefabricated codebook, supplementing it with a smaller, unique codebook appendix that the encoder creates on the fly.

In response to **Reference 2**, Sean Alexander, product manager for Microsoft's digital media division, responded that Microsoft doesn't "do vector quantization." However, some reported characteristics of WMA lead me to suspect that although Microsoft might not be employing a strictly defined vector-quantization approach, their algorithms might be analogous to or derived from TwinVQ-like techniques. Several sources say, for example, that snippets of WMA-encoded audio sound better at a given bit rate than entire

encoded songs. This feedback wouldn't make sense if, like MP3 and many other perceptual coders, WMA simply transformed and compressed multi-sample frames of several milliseconds one at a time.

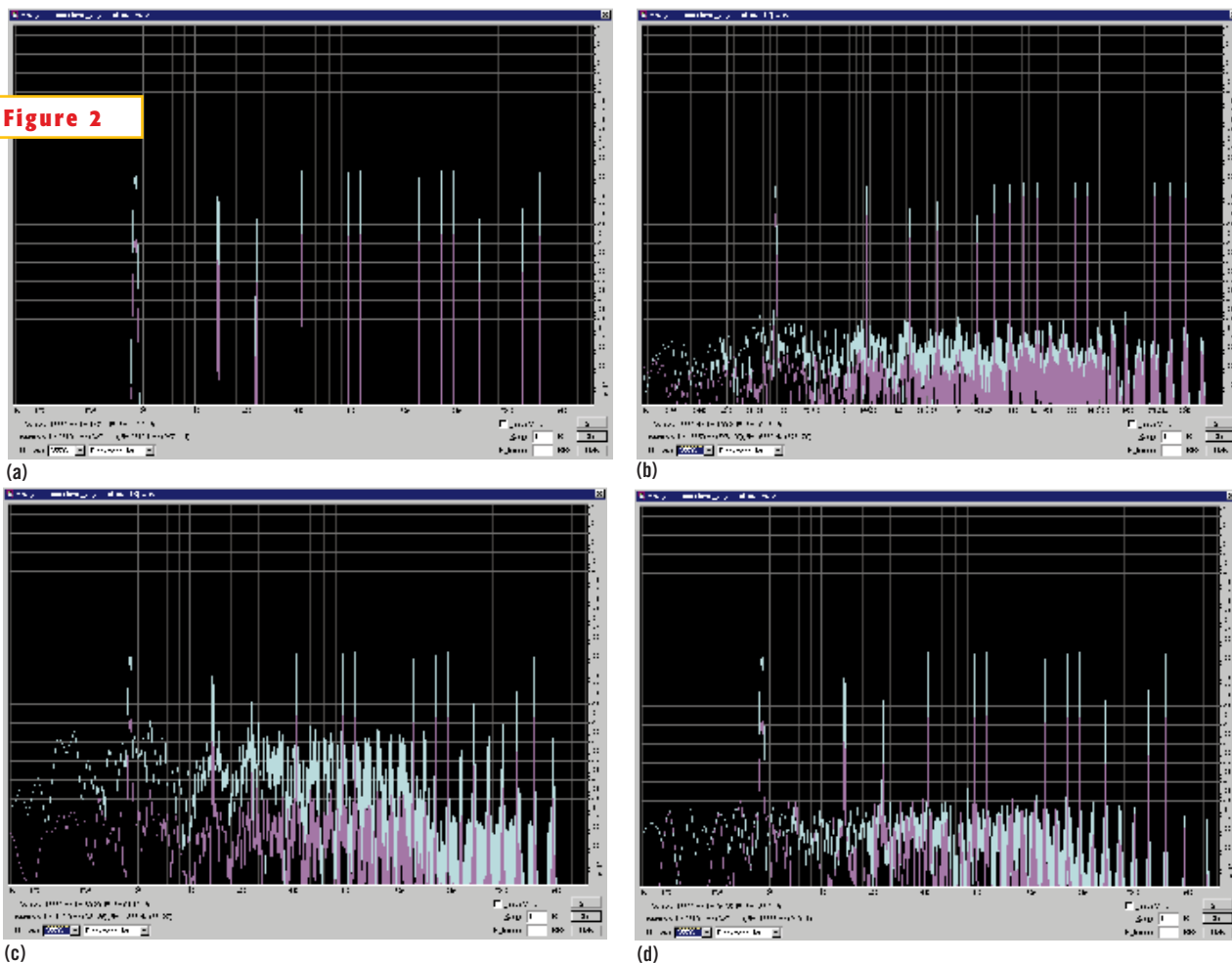
Quality degradation with increasing audio duration could, however, occur if the encoder creates a codebook on the fly. Longer audio sequences tend to contain more randomness than shorter clips—randomness that a fixed-size codebook less accurately approximates. A discussion on Internet newsgroup [rec.audio.pro](#) (search on topic "Sound & Vision's Download Showdown: MP3 versus AAC & Windows Media") between PCBX Web-site audio consultant Army Krueger and Microsoft's general manager for digital media, Amir Majidimehr, also hints at this phenomenon. Majidimehr was unable to replicate Krueger's results, in which WMA-encoded versions of test clips exhibited pre-echo, quantization noise, missing information, and other artifacts. Majidimehr and Krueger determined that whereas Majidimehr separately encoded each test clip, Krueger combined all of his test clips into one big file before running them through the WMA encoder.

Majidimehr wrote in one of the newsgroup postings that indeed, a combined input file of all the samples does generate different results than encoding each clip. He warns that if you want to gang encode a lot of samples, then you must leave sufficient space between them, so that the samples are truly independent. (Later in that posting, he recommends leaving 5 sec between samples.)

Otherwise, he warns, your results will represent only a composite, as the previous clip may change the outcome for some codecs, such as those that Microsoft produces. I hesitate to conclude that WMA is using vector-quantization-like techniques, for two reasons. The first of these reasons is Sean Alexander's aforementioned "no-vector-quantization" comment, although something peculiar is definitely going on.

WMA also exhibits a curious discrepancy from TwinVQ and other vector-quantization compression approaches targeting multimedia. (Similar techniques are possible with still- and video-image compression.) Vector quantization has a reputation for extreme slowness, particularly when the algorithm incorporates on-the-fly codebook creation. WMA encoding, however, is reputed to be faster than other comparable-quality lossy-compression routines, and my results bear out this reputation. Perhaps Microsoft's engineers have just written tight code that takes advantage of processor-acceleration hardware, such as MMX instructions. Or maybe WMA doesn't use vector quantization at all; both Krueger and Ken Gundry from Dolby Labs have hypothesized that perhaps some kind of moving, large-window Huffman coding might explain the reported time-dependent effects. I'll continue to dig into the algorithm and report on the *EDN* Web site any interesting results that I encounter. Microsoft also recently announced version 8 WMA tempting me to rerun my tests on this new codec version.

Figure 2



Spectrum-analyzer (frequency-based) displays show test clip 6 in original WAV (a) and lossy-compressed MP3 (b), RealAudio (c), and Windows Media Audio (d) formats.

Now let's see how well the test tones unveil the secrets behind the lossy codecs' magic. First, look at a spectrum-analyzer (frequency-sweep) plot of the original sound clip 2 (Figure 1a), along with its 64-kbit MP3 (Figure 1b), RealAudio (Figure 1c), and WMA (Figure 1d) counterparts. This diagram, and all subsequent MP3 diagrams, show the output of the encoder set to its "fastest encode" setting. As you examine the data that follows, as well as the additional information in this article's Web site addendum (www.ednmag.com/ednmag/extras/01cs/addendum.asp), compare the trade-offs that codec developments made at each compressed bite rate, such as encode and decode speed, noise floor versus frequency, overall frequency range, and type and amount of various artifacts.

As expected, the original file shows content extending to 22.05 kHz; the summation of the left channel's frequency components is 20 dB "louder" than the right. (The left channel appears in aqua,

and the right channel appears in violet). Also, notice the negative slope of both channels' amplitude-versus-frequency plots. This negative slope occurs because pink noise, which contains equivalent audio energy in each octave frequency, proportionally places a greater amount of content in low frequencies than it does in high frequencies. A white-noise graph, in contrast, would show a flat amplitude slope versus frequency.

Now, compare the original plot to the MP3 graph. Two things are immediately evident. First, the upper end of the MP3-encoded frequency range terminates at just greater than 10 kHz, meaning that the encoder has lowpass filtered and discarded all information above this point. One reason the encoder does this filtering and discarding is because it makes the highly dubious assumption (at this chosen cut-off frequency) that many of us would be unable to hear high-frequency content above this point even if it existed. Secondly, compression algorithms work best

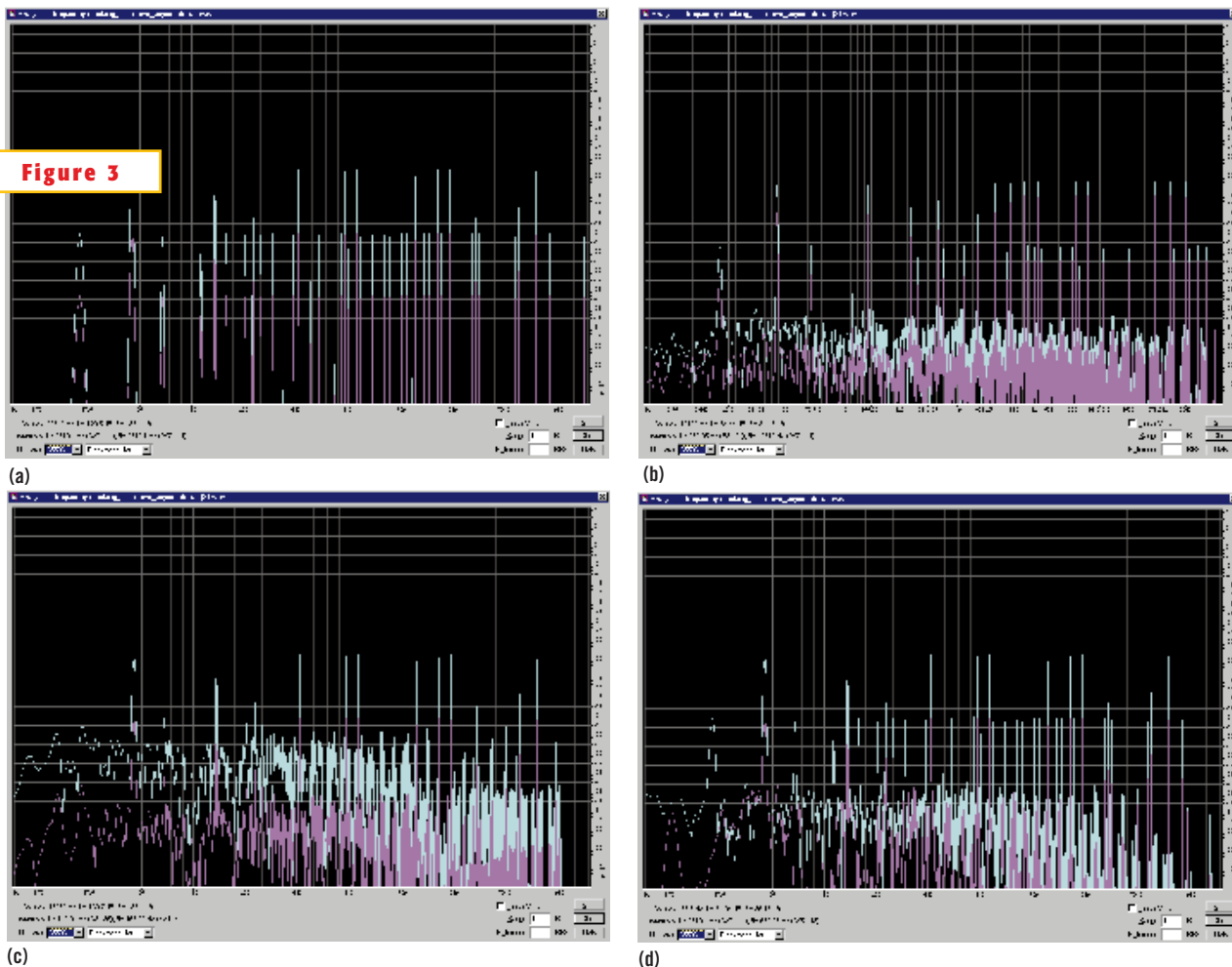
if they can reduce sample-to-sample variation. For audio, this variation is most significant at high frequencies.

Next, notice that the amplitude deviation between the two channels is much less pronounced in MP3 than in the original, particularly at high frequencies. This trend indicates that the encoder is doing a frequency-dependent stereo-to-mono partial conversion to reduce channel-to-channel differences and consequently simplify its job.

Compared with MP3, RealAudio looks pretty good. The frequency response extends quite a bit higher, past 16 kHz, and the channel-to-channel amplitude difference is better preserved across the entire frequency range. Finally, take a look at WMA. Of the three lossy codecs, WMA delivers the widest frequency response, and the left channel looks pretty good. But what about that right channel? Much of the frequency detail has been altered and discarded.

Noise files provide useful data on how

Figure 3



Spectrum-analyzer (frequency-based) displays show test clip 8 in original WAV (a) and lossy-compressed MP3 (b), RealAudio (c), and Windows Media Audio (d) formats.

the compression algorithm works, but their results don't necessarily correlate with how real-life compressed audio sounds. So don't reject WMA quite yet. Also because the files contain random noise, they tend to obscure the subtle alterations that the codecs make. So, next analyze sound-clip 6.

First look at the original file (**Figure 2a**). As expected, it contains 25 distinct tones; both the left and right channels are at uniform amplitudes across frequency, and the right channel is 20 dB below the left. No tone information exists between the 25 critical band midpoints; the noise floor is -120 dB.

The MP3 file looks ugly (**Figure 2b**). Notice again the lowpass filtering: The last three tones in the original file (10,750; 13,750; and 18,775 Hz) are now missing. Also notice the suppressed amplitude difference between the left and right channels even at low frequencies and how this difference further

diminishes as frequency increases. Finally, and perhaps most obviously, look at all the added noise clustered around each of the original tones. Pragmatically, it looks worse than it is; at -80 dB it's not very audible, particularly outside the human auditory system's 2 to 5 kHz "sweet spot."

As the prior pink-noise results predicted, RealAudio has a better-preserved channel separation and frequency response. (The 13,750-Hz tone survived compression; the 18,775-Hz tone did not.) (**Figure 2c**). But at what trade-off? Here, the noise floor extends at times above -60 dB, just a few decibels below the "real" right-channel information. WMA compression (**Figure 2d**), in contrast, delivers clean stereo separation and wide frequency response. (Even the 18,775 Hz tone made it through.) Its noise floor, at no greater than -80 dB, is comfortably below the levels of even the right-channel tones. WMA seems to like

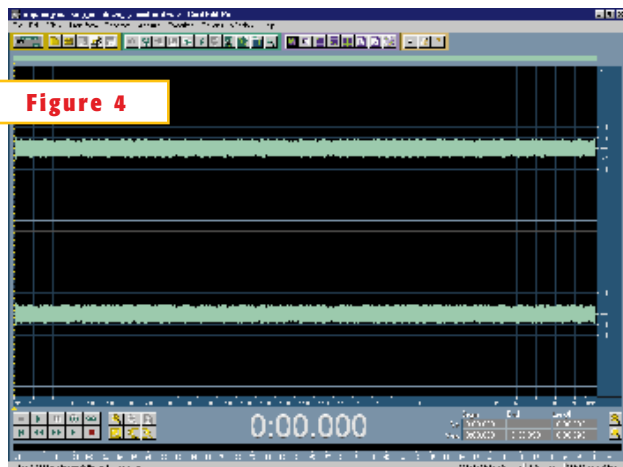
critical band midpoints much more than pink noise.

THE MASK

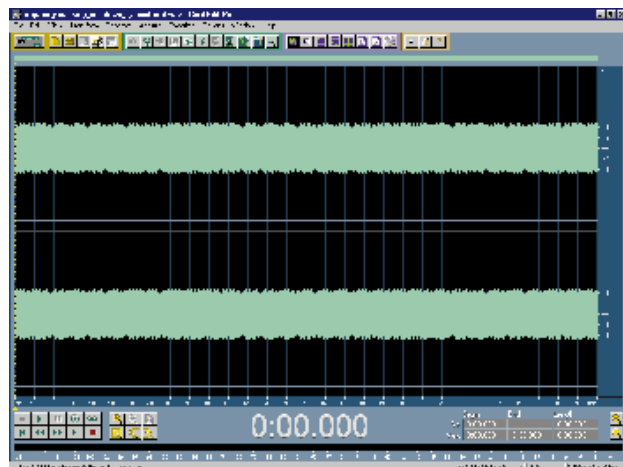
Next, look at test tone 8. First, a spectrum-analyzer display of the original file clearly shows the quarter-, mid- and three-quarter-band tones, with the quarter- and three-quarter-band info 20 dB down (in both channels) from the mid-point tones (**Figure 3a**). Now look at the MP3 version, and you'll see little evidence that the algorithm has done any frequency masking (**Figure 3b**). The encoder algorithm did not eliminate any of the quarter- and three-quarter tones, at least the ones that survived the lowpass filter. Note that the 10,125-Hz quarter tone made it through the lowpass filter, but the corresponding 10,750-Hz midpoint tone and 11,375-Hz three-quarter-band tone did not.

The RealAudio graph is a mess (**Figure 3c**). From the frequency plot, you can't

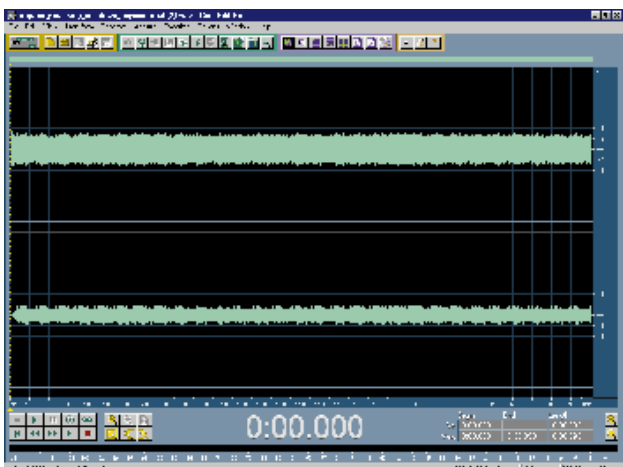
Figure 4



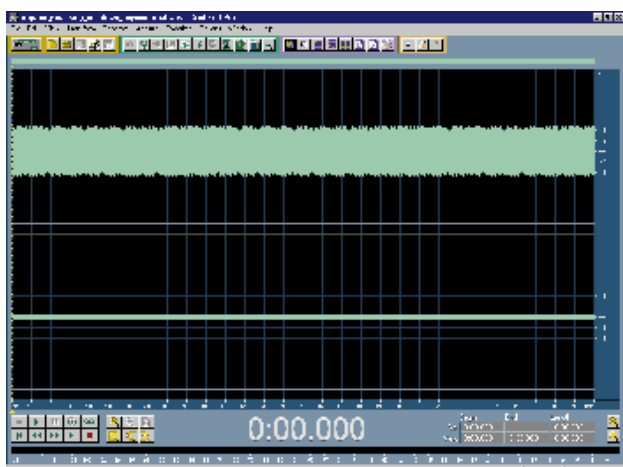
(a)



(b)



(c)



(d)

Oscilloscope (time-based) displays show test clips 7 (in lossy-compressed MP3 (a) and in original WAV (b) formats) and 8 (in lossy-compressed MP3 (c) and in original WAV (d) formats).

distinguish a distorted quarter- or three-quarter-tone from unwanted noise. In test tone 8, I intentionally set the amplitude of the original file’s left channel quarter- and three-quarter-tones identical to the amplitude of the right channel’s mid-tones. I suspect this amplitude and tone combination didn’t simplify the encoder’s job, although it appears that as with test tone 6, the midtones in both channels survived the encoding process pretty well. The additional and altered stuff in between the midtones causes the problems.

What about WMA (**Figure 3d**)? Remember that with the pink-noise file, the left channel survived pretty much unscathed, but the right channel came out looking very different from its original state. A similar phenomenon happened here. The quarter- and three-quarter tones of the left (louder) channel remain intact. But right-channel quarter- and three-quarter-tones, particularly below

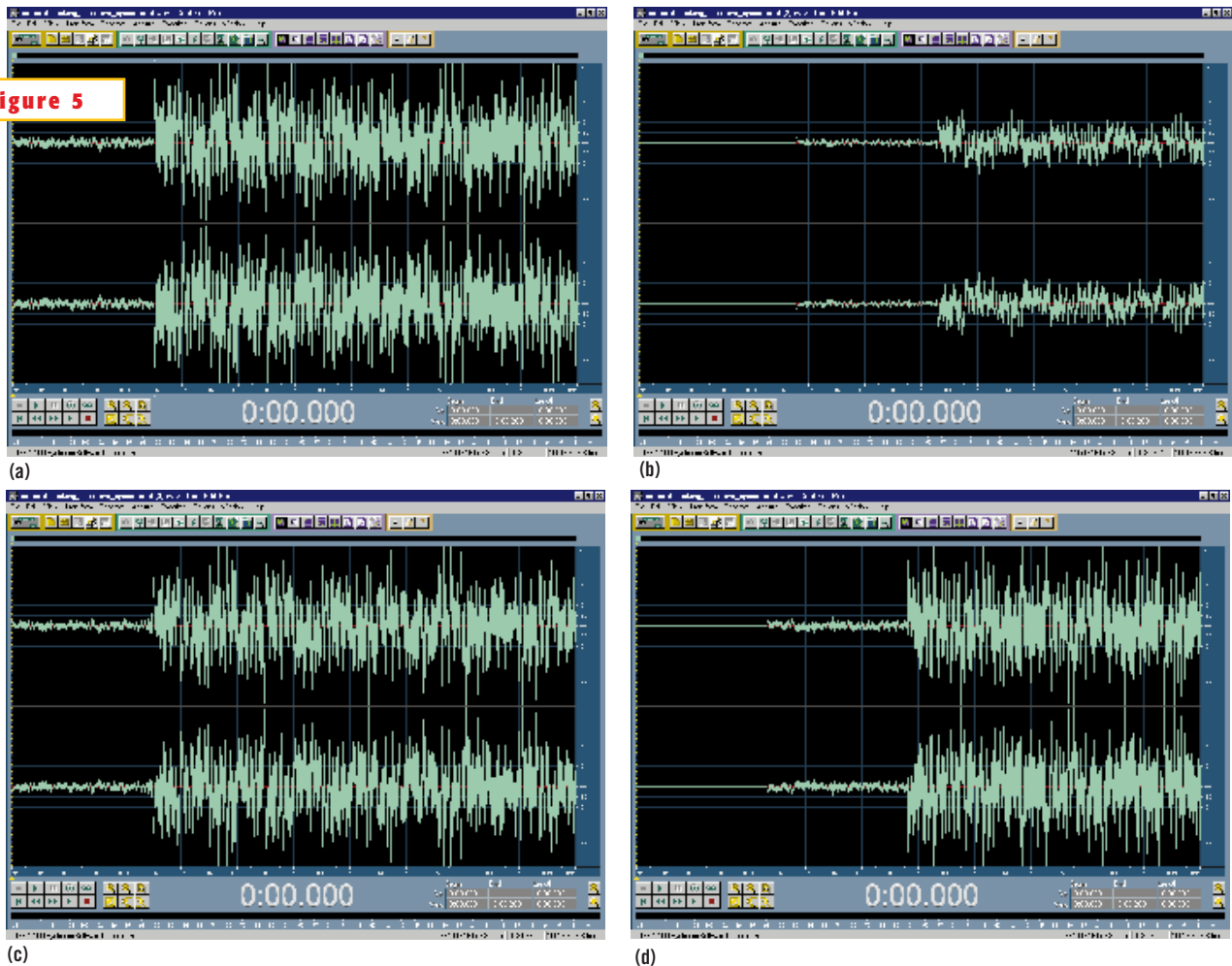
critical band 18, are nonexistent; the disappearing act is most obvious with critical band 0 data. Keep in mind that this artifact, as is the case with many artifacts I find in this study, isn’t necessarily “bad”; frequency-masking theory dictates that even if the quarter- and three-quarter-tone data remains, you might be unable to hear it.

I mentioned earlier that the original quarter- and three-quarter-tone data seemed to survive MP3 encoding. But the MP3 compression algorithm wasn’t immune from sound-altering behavior with test clips 7 and 8. Look at the additional, slowly decaying amplitude in both channels in the first few hundred milliseconds of the MP3-compressed version of test-tone clip 7 (**Figure 4a**), representing increased volume absent from the original WAV file (the left channels on top with the right channels below it) (**Figure 4b**). For an even stranger oscilloscope plot,

look at the results from test-tone clip 8 (**figures 4c and 4d**), in which the increased amplitude in the left channel corresponds with decreased amplitude in the right channel. Similar MP3 behavior occurred with some of the other test tones, although not to this extreme. Neither RealAudio nor WMA exhibited similar behavior.

My initial attempts to uncover temporal masking were unsuccessful, but they did reveal other strange encoder and decoder behavior. Take a look at the first 200 msec of test tone 9 (**Figure 5a**). If temporal masking had occurred as part of lossy compression, you would see an interval with a reduced amplitude or a completely silent interval in the lossy-compression clips just prior to the onset of the “normal” audio material (at the 50-msec point in the original WAV file). Neither the MP3 (**Figure 5b**), RealAudio (**Figure 5c**), nor WMA (**Figure**

Figure 5



Oscilloscope (time-based) displays show the first 200 msec of test clip 9 in original WAV (a), lossy-compressed MP3 (b), RealAudio (c), and Windows Media Audio (d) formats.

5d) versions of the test tone exhibit such masking evidence. Also note that all three lossy codecs appear to have preserved at least some of the channel-to-channel phase differences present in the original; one channel is a mirror image of the other.

You should, however, notice a couple of odd occurrences in Figure 5. First, see how the MP3 algorithm significantly attenuates the original signal, whereas the RealAudio and WMA clips are as “loud” as the original version. Also, notice that MP3 inserts in its compressed version of the test tone a filter-bank-delay-created 55-msec initial silent gap, and WMA inserts a 45-msec gap. These gaps are neither present in the original nor does RealAudio insert them.

RealAudio’s gap addition occurs at the tail end of the sound clip. Comparing Figure 6a with Figure 6c, RealAudio inserts at the end of the test tone 1.385 sec of silence. The 50-msec gap added at the MP3

version’s back-end (Figure 6b) is smaller than RealAudio’s but still present, and WMA (Figure 6d) sticks an even smaller 30-msec gap at the end of the test tone.

Why didn’t I find temporal masking? Keep in mind that with all of these test tones, I chose specific frequencies, as well as specific masked- and masking-tone amplitudes. Changes in any of these source variables can trigger temporal masking or any other lossy-compression technique, as can compressing to a different encoder setting combination. For example, versions of the Fraunhofer MP3 “engine” in some software packages enable you to select whether to allow the encoder to use channel-combining joint stereo techniques; Fraunhofer MP3 encoder versions in other products don’t give you this customization option.

Finally, let’s look for echo artifacts. First, a quick review about what causes echo in the first place might be helpful.

One of the first steps that nearly all lossy-compression audio algorithms (as well as lossy codecs for still images, such as JPEG, and video, such as MPEG) take involves converting a group of contiguous samples (called a frame) from their time-domain representation to the frequency domain. This process is analogous to the algorithm my computer uses to create the spectrum-analyzer plots in this article.

Once in the frequency domain, the encoder decides which portions of the frame’s data are inaudible and, therefore, appropriate to diminish in importance or even discard. This culling process can inject into the frame quantization noise and other undesirable data. The corresponding frequency-to-time retransformation within the decoder spreads this noise throughout all of the frame’s samples.

Ordinarily, the noise isn’t a big deal; the “real” audio data covers it up. Similarly, temporal masking can hide noise injected after a sharp audio transient (a

hands-onproject *Digital audio compression (Part two)*

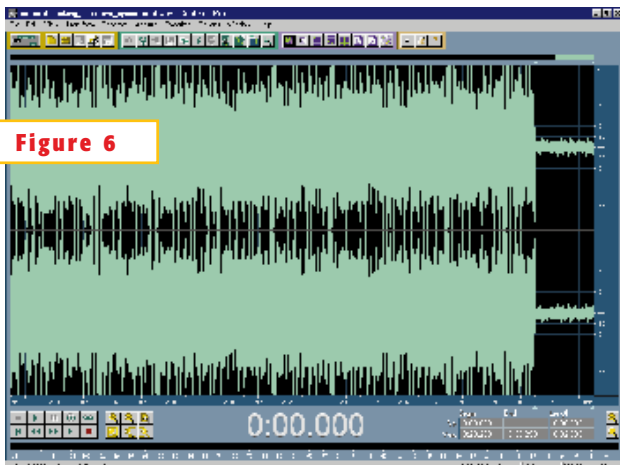
tap on a cymbal or a handclap, for example). But prior to a transient, the “real” audio information is subdued, or

worst-case, silent. Pre-echo not only smears transients, it also injects annoying hiss into the previously quiet gaps

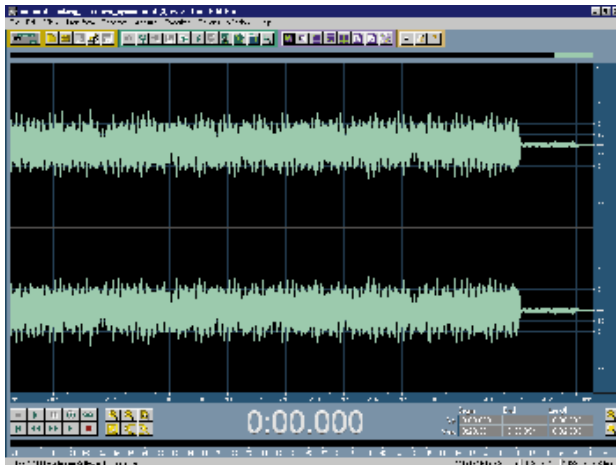
ahead of transients, hiss which temporal masking only partially hides.

My wife, a Cuban music aficionado,

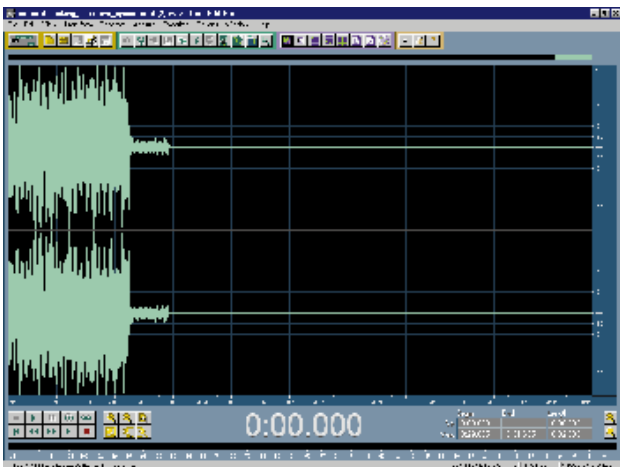
Figure 6



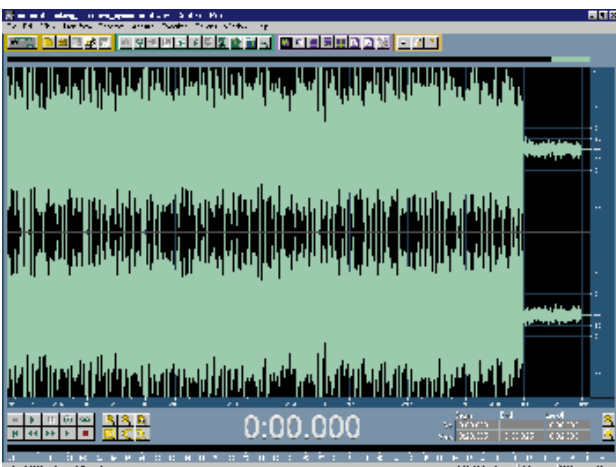
(a)



(b)



(c)



(d)

Oscilloscope (time-based) displays show the last 2 sec of test clip 9 in original WAV (a) and lossy-compressed MP3 (b), RealAudio (c), and Windows Media Audio (d) formats.

FOR MORE INFORMATION...

For more information on products such as those discussed in this article, go to our information-request page at www.rscanners.ims.ca/ednmag/. When you contact any of the following manufacturers directly, please let them know you read about their products in *EDN*.

Dolby Labs

1-415-558-0200
www.dolby.com
Enter No. 400

Fraunhofer Institute

+49 (0) 9131 / 776 0
www.iis.fhg.de
Enter No. 401

Lucent Technologies

1-908-582-8500
www.lucent.com
Enter No. 402

Microsoft

1-425-882-8080
www.microsoft.com
Enter No. 403

Nippon Telephone and Telegraph

03-5205-5550
www.ntt.co.jp
Enter No. 404

QDesign

1-604-688-1525
www.qdesign.com
Enter No. 405

RealNetworks

1-206-674-2700
www.realnetworks.com
Enter No. 406

Sony

1-201-930-1000
www.sony.com
Enter No. 407

Yamaha

1-714-522-9011
www.yamaha.com
Enter No. 408

Other companies mentioned in this article

AT&T

www.att.com

Apple

www.apple.com

Kenwood

www.kenwood.com

Ligos Technology

www.ligos.com

Liquid Audio

www.liquidaudio.com

Minnetonka Audio Software

www.minnetonkaaudio.com

MusicMatch

www.musicmatch.com

Sharp

www.sharp.com

Streambox

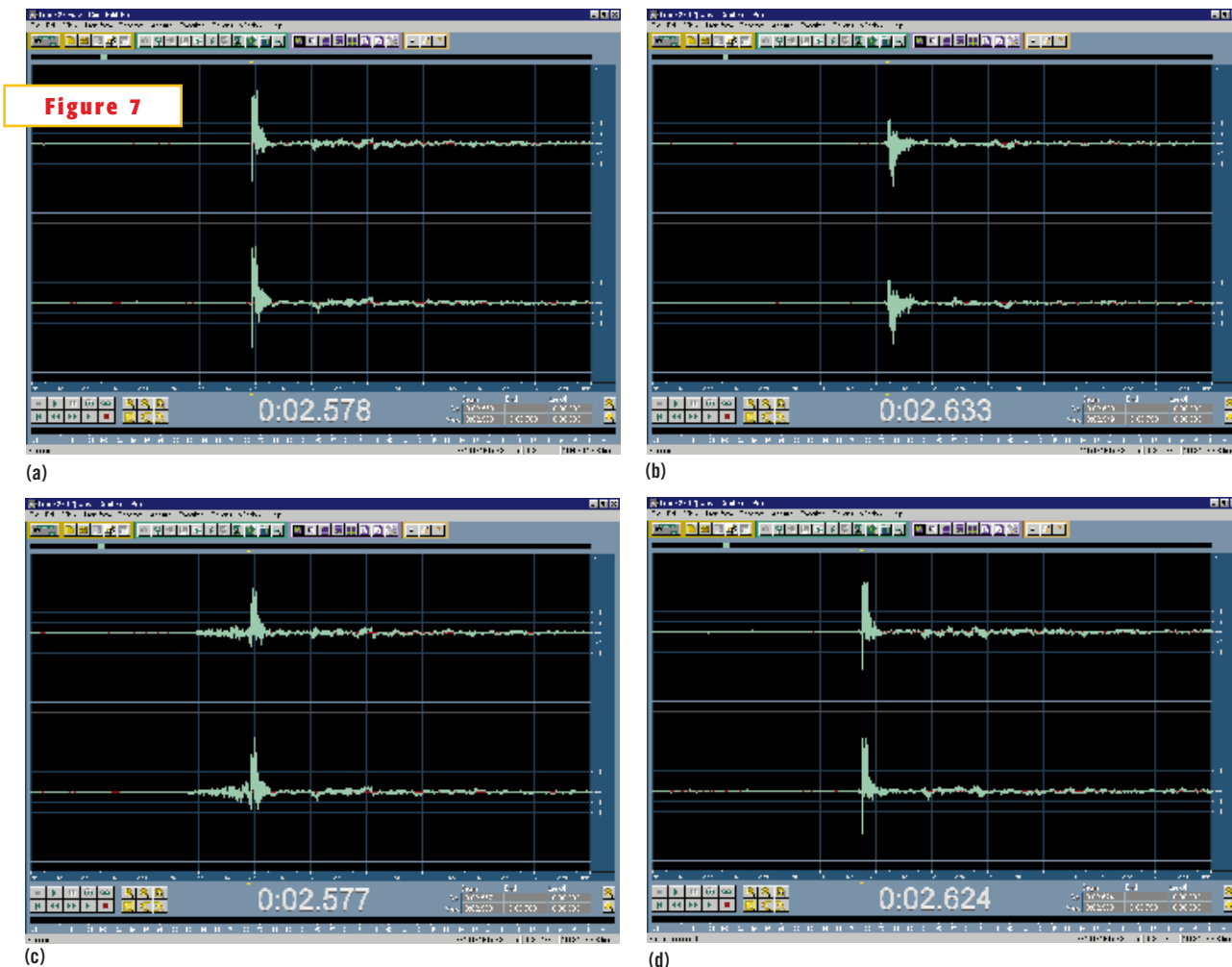
www.streambox.com

Vedalabs

www.vedalabs.com

SUPER INFO NUMBER

For more information on the products available from all of the vendors listed in this box, enter No. 409 at www.rscanners.ims.ca/ednmag/.



Oscilloscope (time-based) displays show the 15th transient in EBU SQAM file 27 (castanets), in original WAV (a) and lossy-compressed MP3 (b), RealAudio (c), and Windows Media Audio (d) formats.

listened to the uncompressed and lossy-compressed versions of the castanets in EBU SQAM test tone 27. Even with my PC's low-quality speakers and without my prompting, she immediately pointed out the pre-echo noise in the 64-kbit

RealAudio file (Figure 7c). This echo doesn't exist in the original WAV (Figure 7a), lossy-compressed MP3 (Figure 7b), and WMA (Figure 7d) files.

The added noise prior to the onset of the transient in the RealAudio file should be obvious to your eyes. And it'll be obvious to most ears, too. In fairness to RealAudio, different codecs use different frame sizes for their time-to-frequency transforms, so other types of transients, or those occurring at other points in time, might cause

the other codecs problems, too. More advanced codecs minimize pre-echo effects by supporting multiple frame sizes. They use less efficient, smaller frames when the encoder detects a transient and longer frames for more conventional material. □

REFERENCES

1. Dipert, Brian, "Now hear this," *EDN*, Feb 3, 2000, pg 50.
2. Dipert, Brian, "Digital audio breaks the sound barrier," *EDN*, July 20, 2000, pg 71.
3. Dipert, Brian, "Technical reference enhances your audio perception," *EDN*, Sept 1, 2000, pg 20.
4. Bier, Jeff and Jennifer Eyre, "DSPs court the consumer," *IEEE Spectrum*, March 1999, pg 47.
5. Ranada, David, "Download show-down II," *Stereo Review's Sound & Vision*, September 2000, pg 113.
6. Ranada, David, "Download show-

down," *Stereo Review's Sound & Vision*, September 1999, pg 98.

7. Ranada, David, "MP3 sound quality?" *Stereo Review's Sound & Vision*, April 1999, pg 40.

8. Dipert, Brian, "Digital audio gets an audition Part 1," *EDN*, Jan 4, 2001, pg 48.

ACKNOWLEDGMENTS

Audio consultant Army Krueger, who maintains the PXABX (www.pcabx.com) and PC AV Tech (www.pcavtech.com) Web sites, provided me with extensive and much-appreciated assistance throughout my entire project. I'm also grateful for the guidance and suggestions of Jim Johnston of AT&T; Eric Benjamin, Andrew Fischer, Ken Gundry, and David Robinson of Dolby Labs; Sean Alexander and Amir Majidmehr of Microsoft; Rebecca Grow and others at Sonic Foundry; Nariman Sodeifi at Syntrillium Software; and numerous employees at Fraunhofer.

You can reach Technical Editor Brian Dipert at 1-916-454-5242, fax 1-916-454-5101, e-mail bdipert@pcbell.net.

