2007 4th International Conference on Electrical and Electronics Engineering (ICEEE 2007)
Mexico City, Mexico. September 5-7, 2007

# A Real-time AAC-type Audio Codec on the 16-bit dsPIC Architecture

Ishaan L. Dalal [ishaan@cooper.edu]
Department of Electrical Engineering, The Cooper Union, New York, NY 10003, USA.

***Abstract*—This paper presents the design and implementation of an AAC-class digital audio decoder and associated playback system on Microchip's 16-bit, fixed-point *dsPIC* microcontroller. Unlike common student "MP3-player" projects that use dedicated decoder chips, one microcontroller handles everything here including I/O, decoding and the user interface. Data read from a flash card is decoded: Huffman coding, inverse quantization, the IMDCT and overlap-add for real-time playback. All code was custom-written in assembly.**

***Index Terms*—AAC, audio compression, dsPIC, PIC**

## I. Introduction

Portable digital music players such as the *iPod* have become ubiquitous. These players use audio codecs such as MP3 (MPEG-2 Layer III) or AAC (Advanced Audio Coding, sometimes called MP4) [1] to encode CD-quality audio at compression factors of 4:1 to 12:1 without perceptible loss of sound quality. To achieve such high reductions, codecs exploit *psychoacoustics* — how the human ear perceives sound across the frequency spectrum.

While less complex than encoding, decoding MP3 or AAC audio still requires significant computational effort. Hardware decoders are generally implemented as ASICs [2], [3] or on 32-bit DSPs [4], [5]. As a simple Google search demonstrates, building an "MP3-player" is an ever-popular project for electrical engineering undergraduates around the world. The embedded systems experience of students at this level is usually limited to breadboarding and 8-bit microcontrollers (MCU) such as the Intel 8051, Atmel AVR and the Microchip PIC. These are inadequate for implementing a perceptual audio decoder. Hence, these "MP3-player" projects often end up using dedicated decoder ICs such the ST Microelectronics STA013 [6] or the VLSI Solutions VS1001 [7] to do the heavy lifting, with the microcontroller only handling I/O.

Such black box decoders do not expose students to any of the underlying signal processing aspects of perceptual audio coding. The goal of this embedded systems project is to implement a decoder (along with a playback system) on an MCU that is familiar as well as easily accessible to a typical electrical engineering undergraduate. The MCU used must have an instruction set conducive to assembly programming and be available in a DIP package. The Microchip *dsPIC* microcontroller fits these requirements, and has an extended version of the regular PIC instruction set that many EE undergraduates, including the author, are familiar with.

We have implemented a simplified AAC-class decoder completely on a 16-bit dsPIC microcontroller. The dsPIC also manages the keypad/LCD user interface and handles all external (flash memory and D/A) I/O making this an integrated playback system. The decoder attempts to follow the spirit of the AAC specification, although some simplifications are inevitable due to hardware limitations.

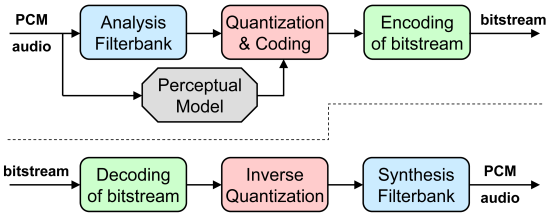


Fig. 1. Perceptual Audio Coding

This paper first discusses the theory of perceptual audio coding and the design of the encoder. The dsPIC hardware and decoding on it are described next, along with the user interface and audio output. We conclude with an analysis of the codec's performance and decoder complexity.

## II. Implementing the Audio Codec

For encoding (Fig. 1), the analysis filterbank in the encoder first decomposes a time-domain input block into the frequency domain. Based on this, the perceptual model computes a *masking threshold* using the rules of psychoacoustics. The filterbank output is then quantized such that quantization noise is below the masking threshold for that block. Quantized frequency coefficients are losslessly compressed with run-length and Huffman coding and formatted into a standards-compliant bitstream. If a specific bitrate is required, the perceptual model and the quantizer iterate in a rate-distortion (R-D) loop to achieve an acceptable compromise. Decoding reverses this process, but is less computationally intensive because no psychoacoustic calculations or an R-D loop are involved.

Our codec is based on a simplified version of the AAC-LC (or MP4) codec that is gradually replacing MP3 as the standard for consumer digital audio. Since the primary goal is real-time decoding on dsPIC hardware, the encoder is software-only (MATLAB).

### A. Theory: MDCT/IMDCT and TDAC

As in AAC, the analysis filterbank is a Modified Discrete Cosine Transform (MDCT), with the synthesis filterbank an





IEEE Catalog Number: 07EX1762C
ISBN: 1-4244-1166-1
Library of Congress: 2007923398

inverse MDCT (IMDCT). Derived from the type-IV DCT, the MDCT is a *lapped* transform that operates on successive blocks of data with a 50% mutual overlap. The overlapping eliminates boundary artifacts that are common to quantized DCTs — a familiar 2-D example is blocky JPEG images on the internet.

The forward MDCT is given by

$$X_{2k} = \sum_{n=0}^{N-1} x_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2} + \frac{N}{4}\right)(4k+1)\right] \quad (1)$$

Since this is an anti-symmetric transform, i.e. $X_{k-1} = -X_{N-k}$, we keep only the odd or even omponents; thus, the 50% overlap does not lead to a size increase as $N$ real points $x_0, \ldots, x_{N-1}$ are transformed into the $N/2$ real points $X_0, \ldots, X_{N/2-1}$.

The IMDCT is

$$x_k = \frac{1}{N} \sum_{n=0}^{\frac{N}{2}-1} X_n \cos\left[\frac{\pi}{N}\left(n + \frac{1}{2} + \frac{N}{4}\right)(4k+1)\right] \quad (2)$$

The MDCT thus subsamples the frequency domain, which causes aliasing in the time-domain output after applying the IMDCT. However, adding these aliased outputs back with a 50% overlap achieves perfect reconstruction (ignoring quantization effects). This is known as time-domain alias cancellation or TDAC [8].

The FFT is often used to calculate the MDCT/IMDCT, considering the large number of optimized FFT implementations available for both hardware and software. A naive approach requires a $2N$-point complex FFT to calculate a $N$-point MDCT. However, it can be shown [9] that an $N$-point MDCT can be "packed" into an $N/4$-point complex FFT with some pre- and post-vector rotations, significantly speeding it up. We adopt this approach, which consists of the following steps ($W_N^k$ are the FFT twiddle factors; $W_{4N} = e^{j2\pi/4N}$, $n = 0 \ldots N/4-1$ and $k = 0 \ldots N/4-1$):

1) $y_n = (x_{2n} - x_{N/2-1-2n}) + j(x_{N/2+2n} - x_{N/2-1-2n})$
2) $y_n = y_n \cdot W_{4N}^{4n+1}$ (pre-rotation)
3) $Y_k = \sum_{n=0}^{N/4-1} Y_n e^{-j2\pi nk/N}$ (FFT of length N/4)
4) $Y_k = ((-1)^{k+1} W_8^{-1} W_N^k) Y_k$ (post-rotation)
5) $X_{2k} = \texttt{real}(Y_k)$ and $X_{2k+N/2} = \texttt{imag}(Y_k)$

Like the FFT, input data blocks are windowed to improve transform quality by avoiding edge discontinuities. The window function must be symmetric, i.e. ($w_n = w_{2N-1-n}$) if the window is defined as $w_n$, ($n = 0, \ldots, N-1$) and must satisfy the Princen-Bradley condition [8], i.e. $w_n^2 + w_{n+N}^2 = 1$ to preserve the TDAC property. We use the *sine* window

$$w_n = \sin\left[\frac{\pi}{N}\left(n + \frac{1}{2}\right)\right] \quad (3)$$

Also specified in AAC, the sine window is applied both before analysis and after synthesis. As it is essentially a rotation, it can be integrated within the pre-rotation of step 2 above.

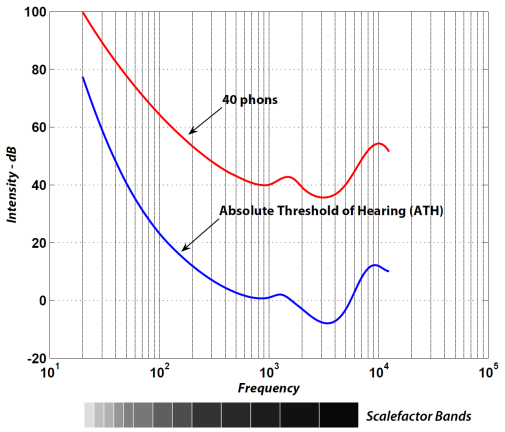


Fig. 2. Equal Loudness Contour/Absolute Threshold of Hearing

### B. Encoding

Stereo 16-bit PCM audio sampled at 44.1 KHz is first loaded into MATLAB and mapped to the dsPIC's 1.15 fractional format. The left (L) and right (R) channels are converted to *M/S stereo*, i.e. mid ($M = \frac{L+R}{2}$) and side ($S = \frac{L-R}{2}$) channels to exploit the (usually) significant correlation between channels; the mid-channel is given greater weight during the quantization stage.

AAC uses both 2048-sample *long* and 256-sample *short* blocks to achieve a trade-off between time/frequency resolution. An AAC codec uses long blocks for relatively stationary portions of the source and switches to short blocks when transients are detected. Since the dsPIC has a very limited amount of RAM (2048 bytes), we always use short blocks (i.e., requiring a 128-point IMDCT).

Since our primary focus is the hardware decoder, the encoder has a very rudimentary psychoacoustic model (although this does not affect decoder complexity. After performing a 256-point MDCT on the input, the spectral coefficients in the 128-point blocks are partitioned into 13 "scalefactor bands" with scalefactors $s_i$ that approximate the critical bands in the human auditory system. A global gain scalefactor $s_G$ is also calculated for each block as

$$s_G = \left\lceil \frac{16}{3} \log_2\left(\frac{\text{max\_mdct\_coeff}^{\frac{3}{4}}}{8191}\right)\right\rceil \quad (4)$$

Each scalefactor band in the block is then adjusted to correspond to the the equal loudness contours/absolute threshold-of-hearing (ATH) from ISO standard 226 [10] as shown in Fig. 2. Each spectral coefficient $X_{2k}$ quantized as

$$X_{2k,\text{quant}} = \text{int}\left[\frac{|X_{2k}|^{\frac{3}{4}}}{2^{\frac{3}{16}(s_G - s_i)}}\right] \quad (5)$$

with signs stored separately. Finally, the scale-factors are differentially coded and then compressed with Huffman tables adapted from AAC [11]. Similarly, the quantized MDCT coefficients are also Huffman-coded as pairs or quadruples.





IEEE Catalog Number: 07EX1762C
ISBN: 1-4244-1166-1
Library of Congress: 2007923398

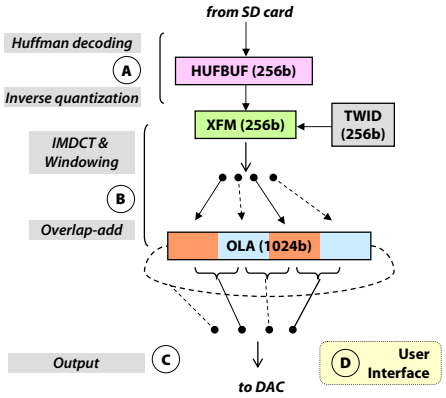


Fig. 3. Decoding Process on the dsPIC

## III. Decoding on the dsPIC architecture

The dsPIC family is a 16-bit variant of the classic 8-bit PIC microcontroller from Microchip. It includes certain DSP-like features such as single-cycle multiply-and-acculumate (MAC) and a 40-bit barrel shifter. Each instruction cycle on the device is four clock cycles, and clocking at up to 120 MHz results in 30 MIPS of performance. Although it is a fixed-point device, fractional operations are supported in the 1.15 format., i.e. bounded to $[-1, 1)$ with 1 sign-bit and 15 mantissa-bits. The dsPIC-30F4013 device used here has 2048 bytes of data (static) RAM and 48 kbytes of program (flash) memory. Programming was entirely in assembler with the free Microchip assembler (MPASM30) and the MPLAB IDE. Fig. 3 illustrates the decoding process.

### A. Huffman Decoding and Inverse Quantization

The dsPIC reads encoded blocks from a Secure Digital (SD) flash card over the 3-wire SPI (Serial Peripheral Interface) Bus. Huffman decoding is performed via table lookup. Although the size of these tables requires them to be stored in the dsPIC's program memory, *Program Space Visibility* is used to "map" to data memory. This allows transparent access with some additional latency (one-two cycles).

Inverse quantization (of eq. 5) is defined as

$$\tilde{X}_{2k} = \text{int}[(X_{2k,\text{quant}})^{4/3} \cdot 2^{\frac{1}{4}(s_G - s_i)}] \tag{6}$$

Power scaling in fixed-point is performed using piecewise linear interpolation along with a 256-entry lookup table as described in [12]. $x^{4/3}$ is calculated as $x^{1/3} \cdot x$, with lookup table $L(x) = x^{1/3}$ for $x = \{1, \ldots, 256\}$;. Therefore,

$$x^{\frac{1}{3}} = \begin{cases} L(x) & \text{if } 0 \le x < 256, \\ 2L(\lfloor \frac{x}{8} \rfloor) + \frac{1}{4}\left[L(\lfloor \frac{x}{8} + 1 \rfloor) - L(\lfloor \frac{x}{8} \rfloor)\right] \cdot (x \bmod 8) & \\ & \text{if } 256 \le x < 2048, \\ 4L(\lfloor \frac{x}{64} \rfloor) + \frac{1}{16}\left[L(\lfloor \frac{x}{64} + 1 \rfloor) - L(\lfloor \frac{x}{64} \rfloor)\right] \cdot (x \bmod 64) & \\ & \text{if } 2048 \le x < 8192. \end{cases} \tag{7}$$

Fig. 4 shows the interpolation error for all possible quantized values. The maximum error is 0.69832 ($\approx 0.00263\%$), while the mean error is 0.20990 ($\approx 0.000297\%$). All $2^{\frac{1}{4}}(s_G - s_i)$ are also stored in a lookup table.

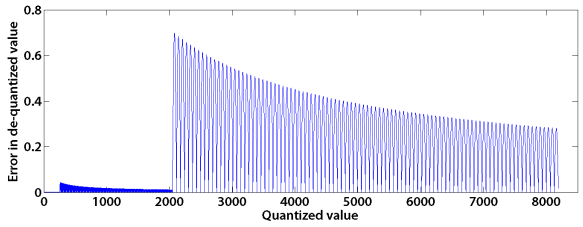


Fig. 4. Interpolation Error in Inverse Quantization Routine

### B. Performing the IMDCT and Overlap-Add

An optimized radix-2 64-point, decimation-in-time, in-place complex FFT was coded for the dsPIC. The transform exploits the device's bit-reversed addressing capability as well as concurrent prefetching for pipelining. The twiddle factors are pre-loaded from program memory into `TWID` upon boot-up. The 128-point block that has been pre-rotated into `XFM` is first transformed and then post-rotated (including windowing, as discussed in section II-A) into the circular overlap-add buffer `OLA`.

`OLA` is two output-frames (1024 bytes) wide and is divided into four subsections. It is filled with decoded data such that at any given time $t$, two subsections are being overlap-added to form the active audio output; one subsection contains future data for $t+1$ while the remaining one contains past data from $t-1$ that is being overwritten with data needed for $t+2$. `OLA` sub-sections are aligned in memory such that the two samples needed for each overlap-add operation can be concurrently fetched.

### C. D/A Conversion, Audio Output and Voice Memos

The dsPIC's Data Converter Interface (DCI) is used to connect to a D/A converter for audio output. Two kinds of DACs were used: a dual-channel channel 12-bit DAC (Texas Instruments TLC5618 - breadboarded) as well as a delta-sigma 16-bit stereo DAC (Burr Brown PCM1725 - soldered to DIP adapter).

With a microphone, users also can record short voice memos on to the SD card in 2-bit Adaptive Differential Pulse Code Modulation (ADPCM) format. Through adaptive quantization, ADPCM provides speech-quality audio with 4:1 compression (here, 8-bit to 2-bit). The microphone input is pre-amplified by +20 dB with an op-amp and sampled by the dsPIC's on-board A/D at 8 KHz/8-bit. The reference IMA-ADPCM code was ported from C to dsPIC assembly. Regular audio playback cannot occur while voice recording is in progress.

### D. User Interface: Keypad and LCD

The system is controlled via a 16-button keypad, whose matrixed 4-bit output is connected to the four "Interrupt-on-toggle" pins on the dsPIC. A 4-line, 20-character LCD based on the Hitachi HD44780 controller serves as a display and communicates with the dsPIC via a 4-bit data/2-bit control


1-4244-1166-1/07/$25.00 ©2007 IEEE.
IEEE Catalog Number: 07EX1762C
ISBN: 1-4244-1166-1
Library of Congress: 2007923398

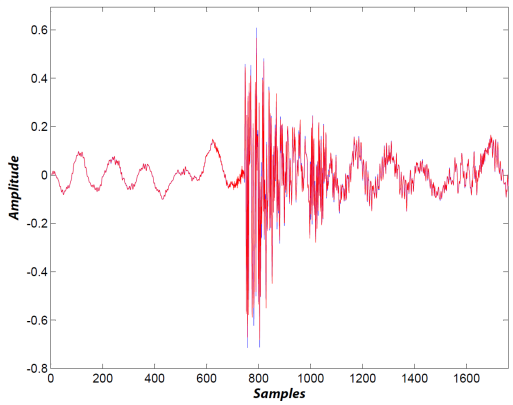


Fig. 5. Transient Encoding/Pre-echo

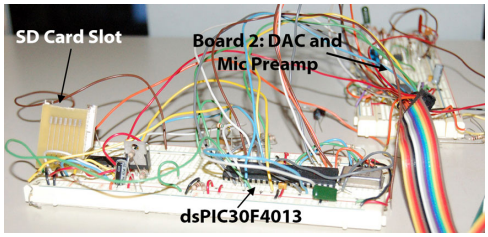


Fig. 6. The dsPIC Decoder/Playback System

signals.The keypad provides standard play/pause, stop, fast-forward/rewind, forward/backward-skip along with volume controls. The LCD displays song information (stored as 256-bytes of ASCII at the end of each encoded piece) and a time counter.

## IV. Analysis and Conclusion

The codec was tested with various kinds of audio samples, such as classical, pop, rock and female vocal. Compression ratios of between 4.64:1 and 8.54:1 were obtained. These are not as good as AAC's, primarily due to the simplistic psychoacoustic model in the encoder as well as the inability to use longer blocks (> 256-samples) because of memory limitations. Using short 256-sample blocks does lead to excellent transient performance; Fig. 5 shows a sharp transient from the notoriously difficult ISO-MPEG sample *castanets*; the codec (red) performs comparably to the original (blue) without any sign of pre-echo. The down-side of short blocks is that they cannot fully exploit the correlation inherent to more stationary portions of the source.

Table I breaks down the complexity for various processes in the decoder for stereo playback at 44.1 KHz. Only $\approx 12\%$ of the available 30 MIPS is being used. Most of the algorithms necessary for a standards-compliant AAC decoder are already present; future work to reach full compliance includes adding an AAC format parser as well as using a dsPIC with sufficient RAM for AAC long blocks (1024-point IMDCTs).

Thus, an AAC-class hardware audio decoder/playback system that uses just one dsPIC microcontroller has been designed and presented. It diverges from common undergraduate-EE "MP3-player" projects by implementing the decoding entirely on the MCU in assembly and not using dedicated decoder chips. Since it uses commonly available components that can be breadboarded and a microcontroller whose instruction set is a superset of the popular 8-bit PIC MCUs, such a system can provide great educational value to students who are interested in experimenting with perceptual audio signal processing algorithms but may not have access to or familiarity with advanced DSPs, compilers and other equipment.

TABLE I
dsPIC Decoding Complexity

| Operation | Instr./Block | MIPS/Sec | Percentage |
|---|---|---|---|
| Huffman Decoding | $\lesssim 2560$ | 0.443 | 1.47% |
| Inverse Quantization | $\lesssim 1152$ | 0.199 | 0.67% |
| Pre-rotation | 2355 | 0.408 | 1.36% |
| 64-pt Complex FFT | 9680 | 1.675 | 5.58% |
| Post-rotation/Windowing | 3017 | 0.522 | 1.74% |
| Flash I/O, UI and Display | $\approx 1400$ | 0.242 | 0.81% |
| DAC/Audio Playback | $\approx 300$ | 0.052 | 0.17% |

## Acknowledgment

The author would like to thank Prof. Stuart Kirtman, instructor for the embedded systems course where this project began, for his continued patience and support as it kept evolving. Ashwin Kirpalani's help with debugging the FFT on the dsPIC is also appreciated.

## References

[1] ISO/IEC 14496-3:1999, *Information Technology: Coding of audio-visual objects, Part 3: Audio*.
[2] S. Hong *et al.*, "A low power full accuracy MPEG1 audio layer III (MP3) decoder with on-chip data converters," *IEEE Trans. Consum. Electron.*, vol. 46, no. 3, pp. 903–906, 2000.
[3] K. H. Bang, N. H. Jeong, J. S. Kim, Y. C. Park, and D. H. Youn, "Design and VLSI implementation of a digital audio-specific DSP core for MP3/AAC," *IEEE Trans. Consum. Electron.*, vol. 48, no. 3, pp. 790–795, 2002.
[4] D. Lai, Q. Lin, S. Chen, and M. Margala, "A low power DSP core for an embedded MP3 decoder," *Proc. 27th Conf. IEEE Industrial Electronics Society*, vol. 3, pp. 1892–1897, 2001.
[5] K. H. Lee, K.-S. Lee, T.-H. Hwang, Y.-C. Park, and D. H. Youn, "An architecture and implementation of MPEG audio layer iii decoder using dual-core DSP," *IEEE Trans. Consum. Electron.*, vol. 47, no. 4, pp. 928–933, 2001.
[6] *MPEG 2.5 LAYER III AUDIO DECODER*, ST Microelectronics, 2004.
[7] *VS1001k - MPEG Audio Codec*, VLSI Solutions, 2004.
[8] J. P. Princen and A. B. Bradley, "Analysis/synthesis filter bank design based on time domain aliasing cancellation," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 34, no. 5, pp. 1153–1161, 1986.
[9] P. Duhamel, Y. Mahieux, and J. P. Petit, "A fast algorithm for the implementation of filter banks based on time domain aliasing cancellation," in *ICASSP-91*, vol. 3, Toronto, Canada, 1991, pp. 2209–2212.
[10] *Acoustics – Normal equal-loudness-level contours*, ISO Standard 226:2003, 2003.
[11] ISO/IEC 13818-7:2004, *Information technology - Generic coding of moving pictures and associated audio information - Part 7: Advanced Audio Coding (AAC)*.
[12] T.-H. Tsai and C.-C. Yen, "A high quality re-quantization/quantization method for MP3 and MPEG-4 AAC audio coding," *Proc. IEEE International Symposium on Circuits and Systems*, vol. 3, pp. 851–854, 2002.





IEEE Catalog Number: 07EX1762C
ISBN: 1-4244-1166-1
Library of Congress: 2007923398