Real-Time MPEG-1 Audio Coding and Decoding on a DSP Chip

Charles D. Murphy, *Student Member, IEEE*, and K. Anandakumar, *Student Member, IEEE*

Abstract— **The MPEG-1 audio standard (ISO/IEC 11172-3) establishes guidelines for the compression of high-quality digital audio signals [1]. The standard dictates the function of an encoder/decoder pair (codec), leaving form intentionally vague to allow for competing implementations. A typical approach to real-time operation is to design an application-specific integrated circuit (ASIC) dedicated to encoding, decoding [2], or both. We present an alternative codec that makes use of the general-purpose digital signal processing (DSP) chips that are now common in multimedia-capable workstations and personal computers. We discuss how selective optimization of codec structure allows robust performance using limited resources, highlight some of the problems inherent in translating the abstractions of the standard into assembly code, and point towards further investigations of real-time implementations of communications standards.**

I. Introduction

The International Standards Organization (ISO) has already approved several coding schemes on the recommendations of the Moving Pictures Experts Group (MPEG), which is charged with developing worldwide standards for compression of digital video and audio signals. Standard ISO/IEC 11172, popularly known as "MPEG-1", describes a framework for compressing video and accompanying audio signals to a combined data rate of 1.5 Mbit/sec. Part 3 of the standard deals solely with audio specifications, providing detailed procedures for decoding the compressed signals and a general overview of the encoding process.

The main purpose of the MPEG-1 audio standard (hereafter "MPEG-1") is to allow the transmission of high-quality audio signals at reduced bit-rates [3], [4]. The standard is generic in that it performs equally well for speech and non-speech inputs. Examples of candidate signals for real-time MPEG audio compression are CD music recordings, movie sound-tracks, digital radio broadcasts, and multi-media sound sequences on the Internet.

The definition of a "high-quality" audio signal has changed with the move from analog to digital storage. Today's commercial products generally have 16-bit quantization of samples obtained at 32, 44.1 (audio CD), or 48 (studio master recordings) kHz sampling rates. From a communications perspective 16-bit quantization of two stereo channels at 48 kHz presents the worst-case raw data stream of roughly 1.5 Mbit/sec, uncoded. This is comparable to what MPEG-1 allows for both audio and video data streams after compression. The standard describes how to compress down to 32 kbit/sec for each channel, a significant savings when it comes to storing or transmitting digital audio data.

In addition to the sampling and compression rates chosen by the user, the MPEG-1 standard is flexible according the quality demands of particular applications and available computing and transmission resources. Fig. 1 shows that there are three "layers" in the standard. Layer 1 is the most basic compression algorithm, with comparatively easy implementation and adequate maintenance of signal quality relative to compression. Layers 2 and 3 have increased complexity, making them more difficult to implement while at the same time allowing better quality at any given level of compression. The two "psycho-acoustic models" enable compression by attempting to mimic the processing that takes place when in the human ear. Model 1 is the less-complicated of the two, but provides poorer quality at a given level of compression.

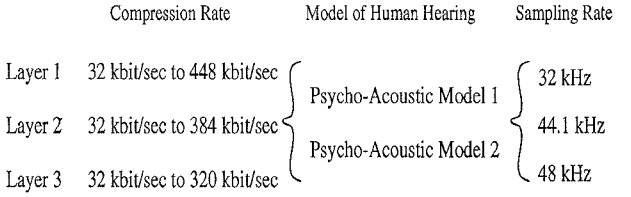


Fig. 1. A Menu of MPEG-1 Options

The pieces of the MPEG-1 encoder come together in four fundamental blocks, and while the decoder has only three, as illustrated in Fig. 2. The encoder consists of a subband analysis filter bank [7] whose outputs are quantized based on the results of a concurrent psycho-acoustic model which tries to minimize audible distortion. The compressed samples leaving the bit-allocation and quantization block are then assembled into frames of data ready for transmission on a communications channel (radio link, wire-line circuit, storage medium, and so on). At the other end, the frames of data arriving at the decoder are disassembled in order to extract the compressed subband samples, which are converted to time-domain output values by a subband synthesis filter bank.

Two competing philosophies exist when it comes to real-time encoding and decoding of an MPEG-1 audio bit stream, the twin tendencies to rely heavily either on software or on hardware. The former approach is a "quick and dirty" solution to the real-time requirement. A team of programmers translates the specifications of the MPEG-1 standard into program code, using a high-level language operating on a powerful workstation. Completion of a working prototype takes relatively little time because the soft-

The authors are with the Signal Processing Research Laboratory at the Moore School of Electrical Engineering, University of Pennsylvania, Philadelphia, PA 19104-6390, USA.

Contributed Paper

Manuscript received December 23, 1996

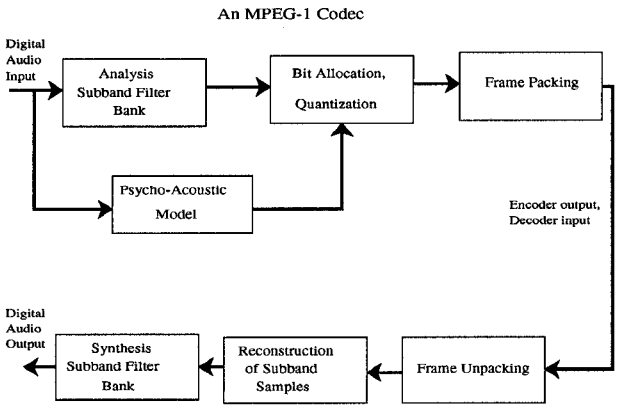


Fig. 2. Block Diagram of an MPEG-1 Codec

ware relies on a general-purpose microprocessor to provide enough speed to offset a lack of optimization. At the other extreme is the "long and painful" approach, in which an MPEG-1 codec takes the form of a pair of application-specific integrated circuits (ASICs). The encoder and decoder chips are cheap when produced in volume, but demand a large investment of resources for custom circuit design and are only useful for MPEG-1 processing.

The main drawback to both approaches is that they enforce a trade-off of equipment cost and general utility. The former limits real-time MPEG-1 coding to those who can afford expensive hardware. The latter allows cheap integrated-circuit encoders and decoders that have no flexibility whatsoever. We explore the possibility that no trade-off is necessary. The digital signal processor (DSP) is an integrated circuit designed with certain types of computationally-intensive tasks in mind, in particular those that require large amounts of arithmetic operations. DSP chips bridge the gap between the generality of a microprocessor and the specificity of an ASIC. For purposes of implementing MPEG-1 and other communications standards, DSP chips offer the promise of low-cost hardware combined with speedy design and testing of software.

For the rest of this paper, we will be discussing a real-time implementation of MPEG-1 layer 1 using psycho-acoustic model 1 and a 32 kHz sampling rate. This real-time codec was produced as the final project of a 6-week summer DSP laboratory course by two graduate students, one of whom started without any prior background in real-time systems or DSP assembly code. The preliminary results establish a baseline for further investigations of MPEG-1, MPEG-2, and other more-elaborate compression techniques.

II. Analysis Subband Filter Bank

The MPEG-1 encoder uses a cosine-modulated, critically-sampled, polyphase filter bank to convert time-domain digital input samples into subband samples ready for compression. The standard specifies one possible procedure for computing the analysis filter outputs, but careful consideration of the underlying theory leads to algorithms that are computationally more efficient and thus better-suited to real-time use.

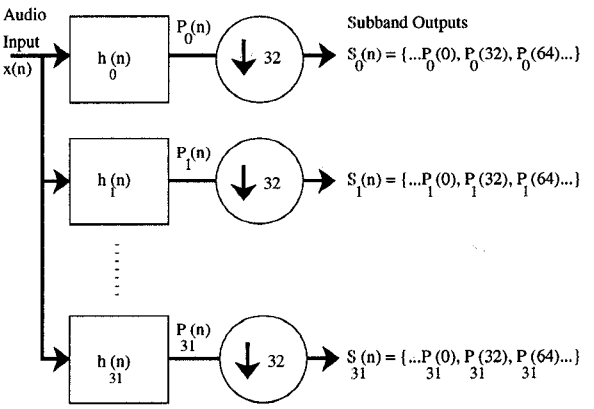


Fig. 3. Diagram of Analysis Subband Filter Bank

Fig. 3 shows the MPEG-1 analysis subband filter bank. The input signal is decomposed into 32 equal-width subbands using parallel band-pass analysis filters $h_i, i = 0, \cdots, 31$. A length-512 lowpass prototype filter h(n), shown in Fig. 4, is multiplied by a cosine term (hence the moniker "cosine-modulated") to obtain the band-pass analysis filters:

$$h_i(n) = h(n)\cos[\frac{(2i+1)(n-16)\pi}{64}], \qquad n = 0, \cdots, 511.$$

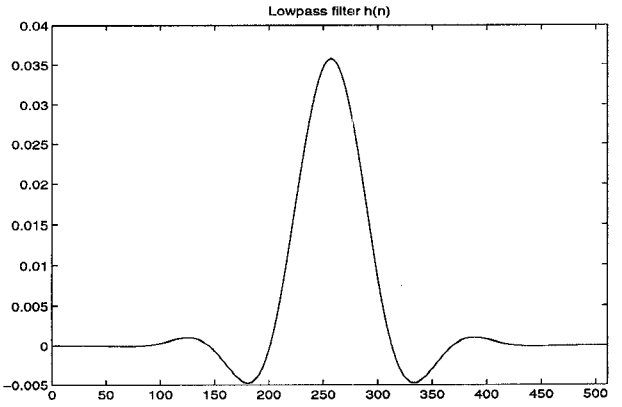


Fig. 4. Prototype Lowpass Filter h(n)

The output of the i^{th} *analysis filter* is given by the following convolution equation:

$$P_i(n) = \sum_{m=0}^{511} x(n-m)h_i(m), \qquad i = 0, \cdots, 31.$$

Fig. 3 shows that every bandpass filter produces 32 subband outputs for each set of 32 time-domain inputs. After down-sampling, 32 analysis outputs remain, meaning that the analysis filter bank is critically sampled. The outputs can be written as

$$\begin{aligned} S_i(n) &= P_i(32n) \\ &= \sum_{m=0}^{511} x(32n-m)h_i(m), \qquad i = 0, \cdots, 31. \end{aligned}$$

A direct implementation of this equation requires 512 multiplications and 511 additions to compute one subband output sample. MPEG-1 uses a polyphase implementation of this equation in order to reduce the computational complexity. By letting the j^{th} coefficient of the k^{th} polyphase filter be

$$C(m = k + 64j) = (-1)^j h(k + 64j),$$

the output of the i^{th} subband can be expressed by

$$\begin{aligned} S_i(n) &= \sum_{m=0}^{511} x(32n - m)h_i(m) \\ &= \sum_{k=0}^{63}\sum_{j=0}^{7}\Big\{ x(32n - k - 64j)h(k + 64j) \\ &\quad \times \cos[\frac{(2i + 1)(k + 64j - 16)\pi}{64}]\Big\} \\ &= \sum_{k=0}^{63} Y(k, n)\cos[\frac{(2i + 1)(k - 16)\pi}{64}] \end{aligned}$$

where

$$Y(k, n) = \sum_{j=0}^{7} x(32n - k - 64j)C(k + 64j), \quad k = 0, \cdots, 63.$$

We need only 80 multiplications and 77 additions to compute each subband output sample using the polyphase architecture above. Further streamlining is possible because the cosine term in the polyphase representation is symmetric about 16 and anti-symmetric about 48 [4]. This allows a reduction to 48 multiplications and 46 additions per output sample. Fig. 5 shows a flow-chart for the subband analysis filter bank.

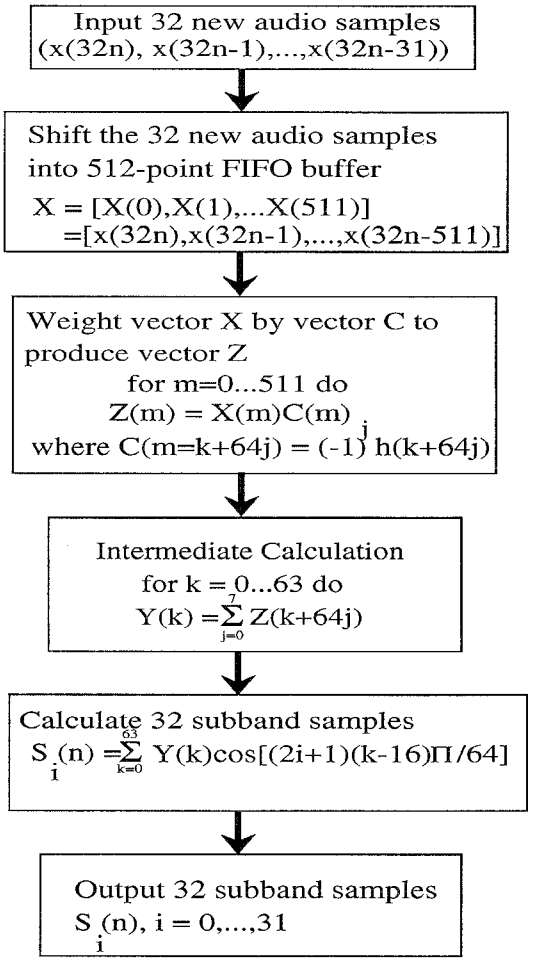


Fig. 5. Analysis subband flow chart to compute $\{S_i(n)\}_{i=0}^{31}$

III. Psycho-Acoustic Model

Arriving time-domain samples get processed by the psycho-acoustic model block in parallel with the subband analysis filter bank. The psycho-acoustic outputs control the process of quantization and bit-allocation, making the psycho-acoustic model the key to maintaining signal quality and achieving significant compression for many different audio inputs.

The MPEG-1 standard allows for the general class of audio signals by exploiting the vagaries of the human auditory system. The human ear is an excellent microphone, but nonetheless has its limitations [5]. The first is an absolute threshold of hearing that varies with frequency per Fig. 6. There is a region of maximum sensitivity between 500 Hz and 4000 Hz, which coincides with the frequencies that dominate human speech. Down to about 100 Hz and up to nearly 20 kHz, human hearing becomes increasingly less sensitive. For compression purposes, it is not important to keep information about inaudible signals (i.e. those below the threshold).

The highly non-linear nature of the absolute threshold curve should raise a warning flag in the mind of the attentive designer. Since the threshold of hearing is based

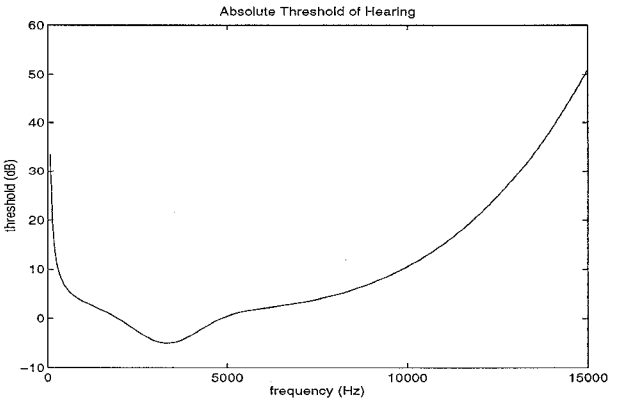


Fig. 6. Frequency-Dependent Threshold of Hearing

on empirical measurements of a biological system, it may not be amenable to convenient mathematical description. The psycho-acoustic model is an attempt to capture the nature of human hearing, but inevitably must make some compromise between biological accuracy and computability. One road-block that leads many to discount the use of DSP chips is that parts of the psycho-acoustic model are not well-tailored to DSP chip capabilities.

The second limitation of hearing, and the essence of the psycho-acoustic model, is the phenomenon of masking. Masking occurs when a strong frequency component is close to weaker frequency components of a signal. The strongest component tends to overwhelm perception of the weaker components. Knowledge of the masking characteristics in a particular frequency range allows us to "hide" quantization noise from the human ear, giving additional compression without any apparent signal distortion.

Fig. 7 outlines the processing flow for psycho-acoustic model 1. The model block accepts 512 time-domain samples as inputs, and produces as output the masking-to-signal ratio (MSR) for 30 of the 32 subbands of the main analysis filter.

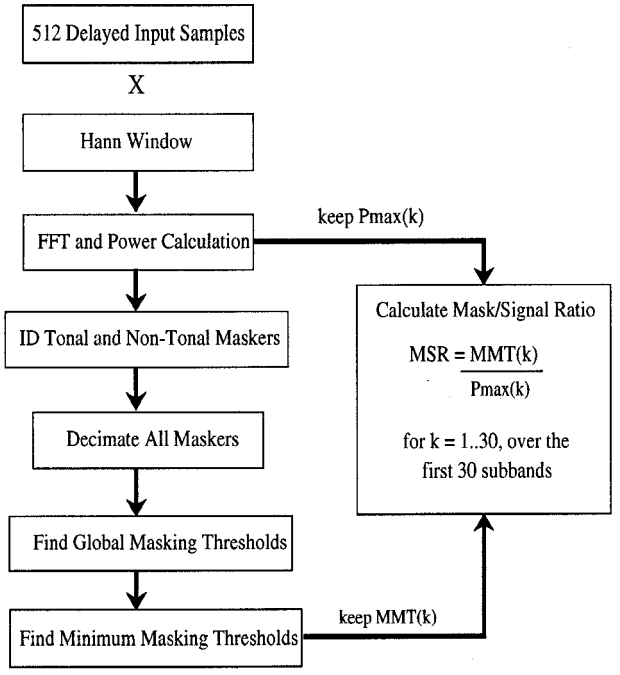


Fig. 7. Psycho-Acoustic Model Diagram

The first step of the psycho-acoustic analysis is to determine the normalized power spectrum for a frame of input samples to be coded. Per the standard, the frame size is 384 samples, with a 512-point power spectrum for adequate frequency resolution. To capture the composition of the entire frame with relatively little distortion due to edge effects, we center the frame of current inputs between the 64 preceding and 64 succeeding samples, applying a smooth taper using a Hann window.

If $x(n), n = 0, \cdots, 511$ is the set of centered input data, the windowing function produces $y(n)$:

$$y(n) = x(n)\frac{1}{2}\sqrt{\frac{8}{3}}(1 - \cos(2\pi\frac{n}{512})) \quad n = 0, \cdots, 511.$$

Next, we calculate the Fast Fourier Transform (FFT) of the windowed data, using a decimation-in-time (DIT) algorithm with radix-2 butterflies [8]. The DIT algorithm offers in-place computation, in which the two outputs of one "butterfly" computation are stored in the memory locations occupied by the inputs. The DIT FFT requires bit-reversed ordering of the inputs (e.g. the second input, 000000001 becomes the 256^{th} input, 100000000), a form of sorting to which the DSP chip caters quite well.

We compute the power spectrum as the squared magnitude of the FFT outputs and normalize it to 96 dB. To avoid the relatively-costly process of converting our inputs to log-power form, we stay in the linear-power domain, and keep the peak power value in the first 30 of the 32 subbands for use in calculation of the MSR.

Up to this point, all of the operations of filtering, windowing, and FFT computation have been well-suited to the capabilities of the DSP chip. In particular, most DSP chips are optimized to perform very fast multiplication and addition, with "number-crunching" power in lieu of speedy control and decision loops. The multiplication by the Hann window and the FFT calculation are well-known "bread and butter" applications for a DSP chip. Throughout the rest of the psycho-acoustic model, we venture into untested regions where we must translate the standard's abstractions into assembly language, with the abstractions themselves being an attempt to model the complexities of the human auditory system.

First we identify the "tonal" and "non-tonal" maskers. These are local peaks of the normalized power spectrum that will drown out or mask nearby weak maxima. Tonal maskers are very strong peaks that are sinusoidal in nature, including most signals. Non-tonal maskers are weaker peaks that are more characteristic of noise. After locating the maskers, we decimate or cull them, throwing away those that fall below the frequency-dependent threshold of hearing. We also eliminate weak tonal maskers that are too close to strong tonal maskers.

The definition of "too close" is a non-linear function of frequency. The standard overcomes this frequency-dependence by entering the critical-band domain, in which the effects of masking are constant in units of "Barks" (1 Bark = the critical band size at a given frequency). MPEG-1 uses a piece-wise linear masking function to relate the critical-band domain to the log-power domain, as shown in Fig. 8. We use an approximation of this model in the non-log-power domain that also happens to be piece-wise linear, allowing us to avoid cumbersome non-linear calculations. This simplification eases our computational burdens at the cost of introducing some degradation in the system performance. Since we are not fully familiar with the field of psycho-acoustics, we will rely on our own perception of the sound-quality to determine whether or not our model is acceptable.

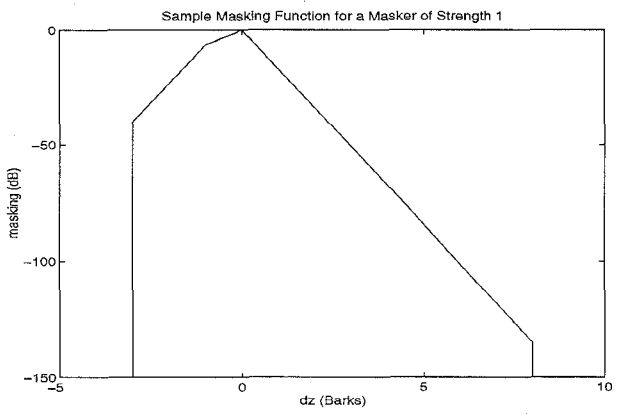


Fig. 8. Model-1 Masking Function

Details of the criteria for identification of tonal and non-tonal maskers and decimation may be found in the standard itself [1]. From an implementation perspective, instead of arithmetic operations, we suddenly must look at comparisons of several elements in a list. High-level programming languages facilitate this with built-in functions for manipulating sets. Our DSP chip's assembly language did not. This forced us to develop our own search and identification techniques, writing explicitly the program steps that are implicit in higher-level languages.

The base masking threshold at each point in the frequency spectrum is identical to the absolute threshold of hearing. In addition, maskers remaining after decimation each contribute to the masking threshold at nearby frequencies, per the masking function. The global masking threshold at each point consists of the absolute threshold and the contributions from all maskers within the proper range. For each subband, we find the minimum masking threshold (MMT). Division by the stored peak power values for the first 30 subbands gives values of the MSR.

To recapitulate, the psycho-acoustic model performs a frequency analysis on each frame of time-domain samples to find the strength and location of various audible peaks, some of which are tone-like and some of which are noise-like. Then a masking function is applied to a subsampled set of spectral lines to determine how much distortion will be imperceptible to the human ear at each line in the set. Lastly, we choose the minimum masking level for the subsampled spectral lines in each subband as the best guaranteed masking threshold, and find the MSR.

IV. Dynamic Bit-Allocation and Quantization

After subband analysis each frame contains 12 output samples per subband. The standard specifies that for each subband we choose the maximum absolute value of the 12 samples as the scale factor and normalize the samples to this value. The psycho-acoustic model and the compression rate are then used to compute the number of bits per sample allocated for quantization in each subband.

A. Bit-Allocation

The bit allocation algorithm begins with computation of the mask-to-noise ratio (MNR) for each subband:

$$MNR = MSR \times SNR$$

where SNR is the signal-to-noise ratio that would result from quantizing a subband output sample to a given number of bits using a uniform, zero-symmetric, rounding quantizer. To calculate the SNR values, the standard assumes that the subband output resembles a sine wave and sets the maximum absolute signal value/rms signal value to $\sqrt{2}$. In our implementation, we use a uniform truncation quantizer for which the SNR is given by the following expression for a sine wave input, with values stored in a table:

$$\begin{aligned} \text{SNR} &= \frac{\text{Signal power}}{\text{Quantization noise power}} \\ &= 1.5 \times 2^{2(B-1)} \end{aligned}$$

where $B \in \{2, \cdots, 15\}$ is the number of bits allocated.

The number of bits available to encode a frame is determined from the desired bit rate and the sampling rate:

$$total\ bits/frame = \frac{384}{Sampling\ rate} \times Bit\ rate$$

The number of bits that are available for encoding the samples and the scale factors is determined by subtracting the number of bits needed for the header (32 bits, plus a possible 16-bit cyclic-redundancy check code) and the number of bits needed for bit allocation representation (4×32 bits) from the total number of bits available.

The bit allocation algorithm now attempts to maximize the minimum MNR of all the subbands by assigning the remaining bits to scale factors and subband samples. Basically, there is a bit-allocation pool, and the process loops. If there are any bits remaining in the allocation pool, the subband with the most un-masked distortion gets an additional bit for representation of its samples. First, the bit allocation to subband samples and scale factors are initialized to zero. Next the following iterative procedure is performed:

1. Determine the subband with the minimum MNR
2. Increase the number of bits for this subband by using the next available higher number of bits
3. For this subband, calculate the new MNR and update the number of scale factor bits. If a non-zero number of bits is assigned to a subband for the first time, then the number of scale factor bits is set to 5 bits.
4. Update the number of bits available for encoding the samples and the scale factors
5. If any possible increase of the required bits within one loop is less than the available bits, go back to 1; else return from the procedure.

B. Quantization

The scale factors are quantized using a non-uniform truncation quantizer. Each scale factor has an allocation of 0

or 5 bits, depending on whether it and its subband samples will be transmitted. The actual scale factor is determined from the base-2 exponent field of the stored floating-point scale factor value. In turn, the subband samples are "fixed", or converted from floating-point to integer (fixed-point) form, with the allocated number of bits less one read off based on the scale factor's exponent. Figure 9 clarifies this process, which depends on the 32-bit fixed-point representation used by the DSP chip.

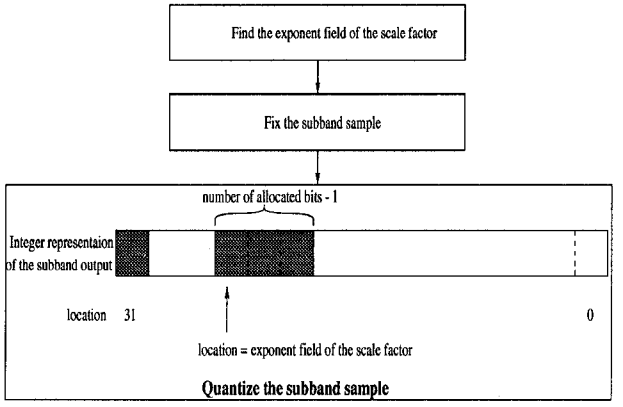


Fig. 9. Chip-Specific Quantization of Subband Samples

V. Frame Packing and Unpacking

After compressing the signal in the dynamic bit-allocation and quantization block, the MPEG-1 standard requires a certain syntax for the encoder output [1]. One frame of data in the MPEG-1 output bitstream contains a 32-bit header, an optional 16-bit cyclic redundancy check (CRC) code, a list of the bit-allocations of the subbands, the quantized scale-factors, and the quantized subband samples. Any MPEG-1 decoder must be able to interpret the bitstream from an MPEG-1 encoder using the same or lower layers, so there is no room for compromise in frame packing.

Likewise, there is little room for optimization on the decoding end with respect to unpacking arriving frames. Extraction of the parameters proceeds in a straightforward manner, with simultaneous rescaling of the subband samples.

During our investigation, we skipped the frame packing and unpacking process in favor of having both the encoder and decoder portions of the codec on a single DSP chip. This deviation was based on our own resource limitations and the desire to explore areas of MPEG-1 that are not "cut and dry".

VI. Synthesis Subband Filter Bank

In the synthesis operation, the 32 dequantized subband samples $\hat{S}_i(n), i = 0, \cdots, 31$ are applied to the synthesis subband filter bank to produce 32 time-domain audio samples. The synthesis subband filter bank is shown in Fig. 10, where $\{g_i(n), i = 0, \cdots, 31\}$ are individual synthesis filters. The standard describes the synthesis filter bank implementation shown in Fig. 11. In the figure, V, U and W are intermediate variables and the coefficients $\{D(i), i = 0, \cdots, 511\}$ are given by $D(i) = 32C(i)$, where $C(i)$ are the analysis polyphase filter coefficients. We include the following modifications to optimize synthesis [2]:

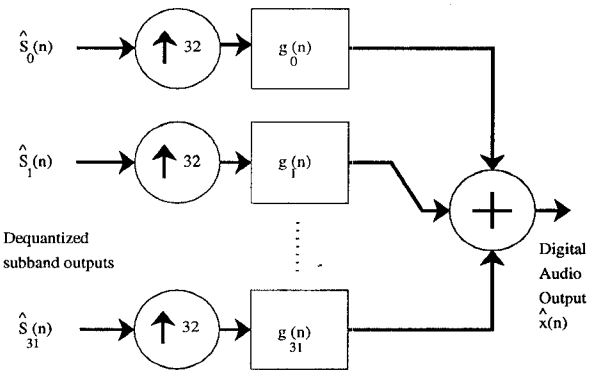


Fig. 10. Synthesis Subband Filter Bank

$$
\begin{aligned}
V(16) &= 0 \\
V(16-i) &= V(16+i) \text{ for } i = 1, \cdots, 16 \\
V(48+i) &= V(48-i) \text{ for } i = 1, \cdots, 15 \\
V(i) &= \sum_{k=0}^{15} \Big\{ \cos\Big[\frac{(16+i)(2k+1)\pi}{64}\Big] \\
&\quad \times [S_k + (-1)^i S_{31-k}] \Big\} \\
&\quad \text{for } i = 17, \cdots, 48.
\end{aligned}
$$

VII. DSP Chip Strengths and Weaknesses

Our hardware consisted of a plug-in board containing a floating-point DSP chip[1] with 2000 bytes of on-chip random access memory (RAM), 16000 bytes of additional static random-access memory (SRAM), an input port with an analog-to-digital convertor, an output port with a digital to analog convertor, and a DOS-based software interface for controlling the chip operation. Nominal performance of the chip is 15 Mflops, a speed typical of "mature" DSP chip technology.

The assembly language of the DSP chip consists of 85 single-operation instructions and 28 parallel-operation instructions. A set of 17 single-operation instructions govern program control. The most useful of these let us force an immediate branch to another portion of the assembly code, initiate a delayed branch, and repeat a block of commands several times. Twelve instructions deal with moving data to and from on-chip registers and other memory locations, while 5 instructions serve to facilitate interaction among multiple DSP chips. The remaining 51 single-operation instructions encompass a variety of arithmetic and logical operations, from addition and multiplication to shifting and comparison. The parallel-operation instructions cover simultaneous loading and storage with arithmetic and logical computation.

[1] A TMS320C30 DSP chip on an EVM board.

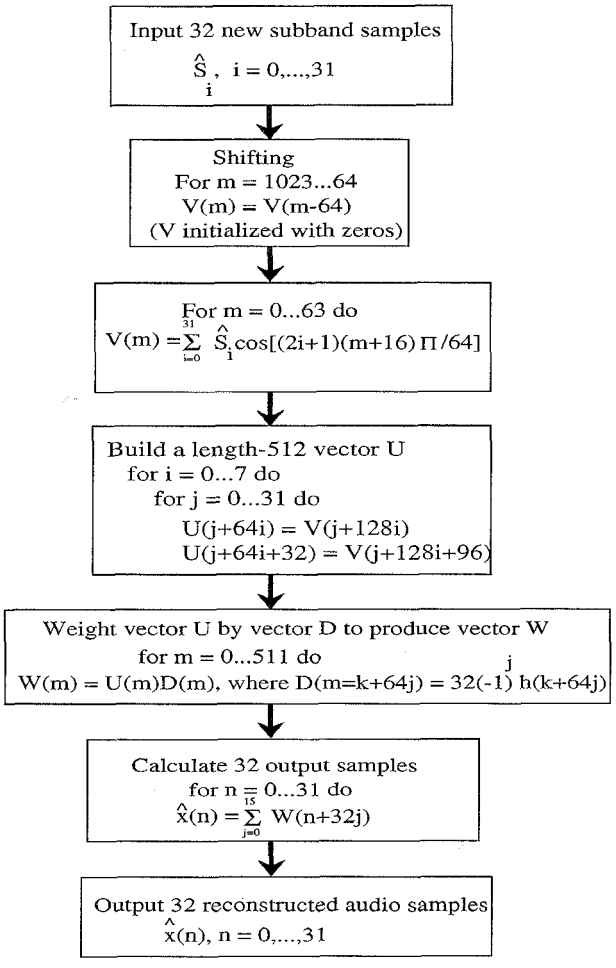


Fig. 11. Flow Chart for Computing $\{\hat{x}(n)\}$ from the Synthesis Subband Filter Bank

Other architectural quirks that add to the power of the DSP chip are circular and bit-reversed addressing. Circular addressing enables the user to set up a group of memory locations that may be accessed one after the other without any extra test to determine when the last memory location has been reached. Pointers to the memory locations automatically wrap around to the beginning of the set once they reach the end. Bit-reversed addressing caters to the needs of certain signal-processing techniques, notably the decimation-in-time FFT used in the psycho-acoustic model block, effectively streamlining a computationally-intensive algorithm [8].

The assembly language and the built-in addressing schemes provide a powerful set of tools for performing traditional digital signal-processing tasks. In many DSP algorithms, there is a heavy reliance on repetitive multiplication and addition. Efficient computation requires a fast, convenient framework for getting at large sequences of data, manipulating them, and storing them. To this end, the parallel-operation instructions were extremely important, permitting access to one memory location in the SRAM and two memory locations in the on-chip RAM during each clock cycle. Combined with other features, this makes the DSP chip ideal for our many signal-processing tasks, from subband analysis and synthesis to FFT and power calculation.

The DSP-specific strengths of the chip are a drawback when we incorporate unusual algorithms. In particular, the non-linear biological model used in the psycho-acoustic analysis presents a major hurdle, not for the chip, but for the programmer.

The main problem is that high-level programming languages have linguistic parallels to human thought processes. A few lines of text in the standard that describe a highly-abstract concept can be swiftly converted into C (or Fortran or Pascal) code by the average programmer. As an example, finding a local maximum in a sequence of data points involves statements such as, "If point n is greater than point n-1, and if point n is greater than point n+1...", a simple enough set of code in C. With only a bare-bones assembly language to work with, intended mostly for DSP algorithms, the translation is not so obvious. For more complex statements, the programming problem becomes extremely difficult. A cursory examination of speed and memory limits leads many prospective designers to head for the relative comfort of the work-station or ASIC-based systems discussed earlier.

To accomodate both the DSP-oriented and non-DSP portions of the MPEG-1 standard, we had to develop "custom" code on a local level. Especially with time and memory constraints, it is important to be ruthless in producing efficient pieces of the bigger puzzle. When the smaller blocks are working well, the whole system typically will not. Because there is little or no margin for inefficiency, several iterations are necessary to designate an appropriate storage and retrieval structure for the data to be processed and the program instructions.

In summary, the DSP chip occupies particular niche somewhere between the general-purpose microprocessor and the single-purpose ASIC. It tends to excel at certain tasks for which it was designed, and present impediments to tasks outside its sphere of expertise. Combining the strengths and weaknesses to push the chip's limits is possible, but requires an attention to detail and a thorough knowledge of nuts-and-bolts chip operation and algorithmic theory. Many product designers tend to overlook the possibilities of the DSP chip as a tool due to its unique pitfalls.

VIII. Performance Results

Figure 12 presents a summary of the operation of the final program. Having skipped the full assembly of transmission-ready frames, we find that the encoding and decoding processes take slightly less than the number of machine cycles available during real-time input and output. There is some leftover memory, but we used all of the on-chip RAM and a great deal of the static RAM available on our test board. The slim margin of safety indicates that the final product pushes the DSP chip almost to its limits.

In the context of a product available to the public, a number of shortcuts that we took should be avoided, or at least carefully considered for validity. At the outset, we

Ultimate Limits

Space: "fast" Random-Access Memory = 2000 bytes
"slow" Static RAM = 16000 bytes

Time: Machine cycles/384-sample frame = 184248 cycles/frame

Final Specifications

Space: "fast" RAM = 2000 bytes
"slow" SRAM = 10385 bytes

Time: Analysis 4757 cycles x 12/frame
Psycho-Acoustic Model = 44242 cycles
Bit-Allocation/Quantization = 13779 cycles
Synthesis = 4765 cycles x 12/frame

= 172285 cycles/frame

Fig. 12. Real-Time Performance at Rate 96 kbit/sec

chose to develop a codec for a fixed sampling rate, layer, and psycho-acoustic model, with bit-allocation of at most between 96 and 128 kbit/sec. A full codec would have to meet a wider variety of the requirements laid out in the standard, including provisions for switching among the available options. Frame packing and unpacking, of course, is also a must. A third consideration is our simplified masking function - how much does it affect the quality-maintenance capabilities of the standard?

From our perspective, the investigation was a success in demonstrating real-time MPEG-1 audio compression. Our final gauge was what we could (or could not) hear. We tested the codec by applying various MIDI sequences to the input jack of our test board, performing encoding and decoding in real time, and piping the resulting signal to a speaker. At a bit rate of 96 kbit/sec, the output was of good quality in our estimation.

IX. Conclusion

We developed a real-time MPEG-1 audio codec using a single DSP chip, including full analysis, compression, decompression, and synthesis. Speed limitations of the chip prevented us from undertaking frame assembly in real time as well, but by splitting the codec into two separate pieces - an encoder and a decoder - we could easily incorporate this and other features of the MPEG-1 standard.

The main obstacle to overcome in this or any similar work is how to allocate limited amounts of memory and processing cycles. The DSP chip can be a workhorse in many ways, but is not all-powerful, especially when it comes to advanced communications standards. On the algorithmic side, one must look for shortcuts in performing the repetitive multiplications and additions that populate typical signal-processing tasks such as filtering and Fourier Transforms. On the modeling front, simplified representations of complex physical phenomena must be translated from high-level abstractions to efficient assembly language programs. Designers often look at these disparate needs and decide that DSP chips do not warrant further consideration as a viable real-time platform.

Our investigation demonstrates that while DSP chips do have their shortcomings, they are also highly versatile. Their successful use in communications standards requires a judicious blend of bottom-up assembly-language design and top-down adaptation of specifications. Many future avenues of exploration exist, including full implementation of the MPEG-1 standard, and a thorough examination of techniques for broadening the scope of real-time DSP chip applications.

References

[1] ISO/IEC Int'l Standard IS 11172-3 "Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s - Part 3: Audio"

[2] T. Tsai, T. Chen, and L. Chen, "An MPEG Audio Decoder Chip", *IEEE Trans. on Consumer Electronics*, Vol. 41, No. 1, Feb. 1995.

[3] S. Shlien, "Guide to MPEG-1 Audio Standard", *IEEE Trans. on Broadcasting*, Vol. 40, No. 4, pp 206-218, Dec. 1994.

[4] D. Pan, "A Tutorial on MPEG/Audio Compression", *IEEE MultiMedia*, Summer 1995.

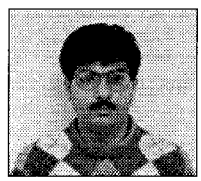
[5] N.R. Carlson, *Physiology of Behavior*. Boston: Allyn and Bacon, 1991.

[6] D. Pan, "Working Draft of MPEG/Audio Technical Report", Release 3.9, Sept. 1993.

[7] A.N.Akansu and R.A.Haddad, *Multiresolution Signal Decomposition: Transforms, Subbands and Wavelets*, Academic Press Inc., 1992.

[8] A.V. Oppenheim and R.W. Shafer, *Discrete-Time Signal Processing*, Englewood Cliffs: Prentice Hall, 1989.

Charles D. Murphy was born on September 25, 1973 in Philadelphia, PA. After receiving the degree of BSEE from Yale University in New Haven, CT, he returned to Philadelphia to pursue a Ph.D. in Electrical Engineering at the Moore School of Electrical Engineering of the University of Pennsylvania. His academic and professional interests include signal processing arrays, telecommunications, remote sensing and exploration, engineering education, science policy, and aspects of international trade, diplomacy, culture, and technology exchange.



K. Anandakumar received the B.A. Hons. degree in Electrical and Information Sciences Tripos (EIST) from the University of Cambridge, Cambridge, United Kingdom in 1993 and the M.S.E.E. degree from the University of Pennsylvania, Philadelphia, U.S.A. in 1995. While at Cambridge University, he was a recipient of the Cambridge Commonwealth Trust Scholarship. Currently, he is working toward the Ph.D. degree in Electrical Engineering at the University of Pennsylvania, where he is a Research Assistant.

His research interests include multiresolution signal representation (including wavelet/subband techniques), signal compression, signal processing algorithms and communication systems.