# Modeling of an MPEG Audio Layer-3 Encoder in Ptolemy

**Patrick Brown**
EE382C – Embedded Software Systems
May 10, 2000

## Abstract

MPEG Audio Layer-3 is a standard for the compression of high-quality digital audio. It has rapidly become very popular and has found widespread application in devices such as portable, all-electronic music players and in Internet audio. Currently, many algorithms for Layer-3 audio encoding are used. Their performances vary greatly, but the average encoding rate is approximately one second of digital audio encoded per second on a typical Windows-based desktop computer. While this rate is acceptable to most personal computer users, some applications demand a much higher encoding rate. For example, a typical radio station has thousands of CDs, each containing up to 74 minutes of digital audio. If the radio station decides to convert all of its audio to MPEG Layer-3, 74 minutes to encode each CD is unacceptable. The goal of this project is a formal modeling of an MP3 encoder, which will expose the parallelism in the encoding algorithm, facilitating the scaling of the algorithm to a multi-processor implementation. Much greater throughput is achievable by scaling the algorithm to multiple processors.

## Context

MPEG (Moving Picture Experts Group) Audio Layer-3 [1], more commonly known as “MP3,” is part of the set of standards known as “MPEG-1,” which was approved by the International Organization for Standardization (ISO) in November 1992 [2]. The primary focus of this standard is the compression of high-quality, synchronized audio and video to a data rate of approximately 1.5 Mbps [3]. This standard consists of three main parts: system, video, and audio. Within the audio portion of the standard, there are three “layers.” Layer-3 provides the highest compression at a given sound quality. Table 1 [4] shows some of the common compression ratios available using MP3 compression based on the relative quality of the resulting audio.

| Sound Quality | Bandwidth | Mode | Bit-Rate | Compression Ratio |
|---|---|---|---|---|
| Telephone | 2.5 kHz | mono | 8 kbps | 96:1 |
| AM Radio | 7.5 kHz | mono | 32 kbps | 24:1 |
| FM Radio | 11 kHz | stereo | 56 - 64 kbps | 26:1 - 24:1 |
| Near CD | 15 kHz | stereo | 96 kbps | 16:1 |
| CD Quality | Over 15 kHz | stereo | 112+ kbps | Up to 12:1 |

**Table 1**: MP3 Compression ratios for various output sound qualities.

In MP3, as with other source coding standards, the decoder is rigidly defined, whereas great flexibility exists in the design of the encoder. Many “freeware,” “shareware,” and commercial encoders exist, some of which are open source. The formal model will be based on the L.A.M.E. encoder [5], because it is open source, freely distributable, efficient, and produces good sound quality. As with most signal processing applications, the encoding algorithm contains a

large amount of parallelism. A major benefit of building the formal model is the exposure of this parallelism, because this is what will make the algorithm scalable to multiple processors.

## Objectives

The formal model of the MPEG Layer-3 encoder will be a dataflow graph consisting of various blocks, or "actors," each containing a portion of the L.A.M.E. source code. The actors will exist within the SDF (Synchronous Data Flow) model of computation. In this model, or "domain," each actor has a fixed number of input and output ports. Each of these ports receives or sends a fixed number of "tokens" of data, such as a single integer or floating-point number, or a matrix of values. There is no notion of time within this domain. This domain is well-suited to the modeling of an MP3 encoder, because input audio is processed sequentially, 576 samples at a time, as quickly as possible, with no consideration of time or other factors that may be present in other domains.

In addition to exposing the inherent parallelism, the formal modeling also allows retargeting of the algorithm to different implementations. Therefore, it will be possible to apply the algorithm, which was originally a C program written to run on a single general-purpose processor, to a wide variety of platforms, such as a multiple processor workstation or custom hardware containing multiple DSPs. Converting the C code to C++ and importing it into SDF actors introduces additional processing overhead, because the domain must provide a means for communication between actors. In the original algorithm, this was simply done

by function calls. Scheduling the algorithm on multiple processors also introduces overhead. This is due to the fact that processors must spend time sending and receiving data. However, the nature of the algorithm is such that the amount of inter-actor and inter-processor communication is very small. This is the main reason that a formal model of the encoder is beneficial.

## Implementation and Modeling

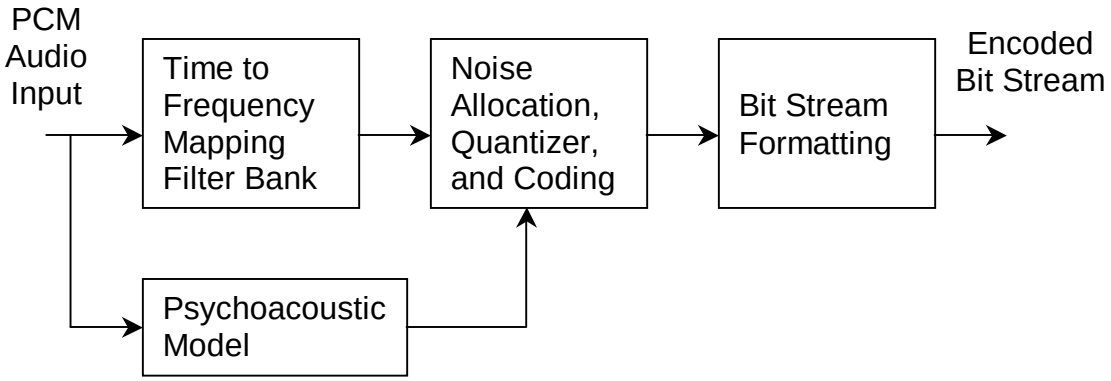


**Figure 1**: Block diagram representation of an MP3 encoder.

The formal model was constructed using Ptolemy [6]. Layer-3 is a very complex encoding scheme; the actual ISO standard [2] is nearly 200 pages long, and the L.A.M.E. encoder source is approximately 20,000 lines in length. Unfortunately, it was not possible to implement a fully functional encoder due to this complexity and the timing constraints of the project. Figure 2 shows a screen shot of the implemented model.

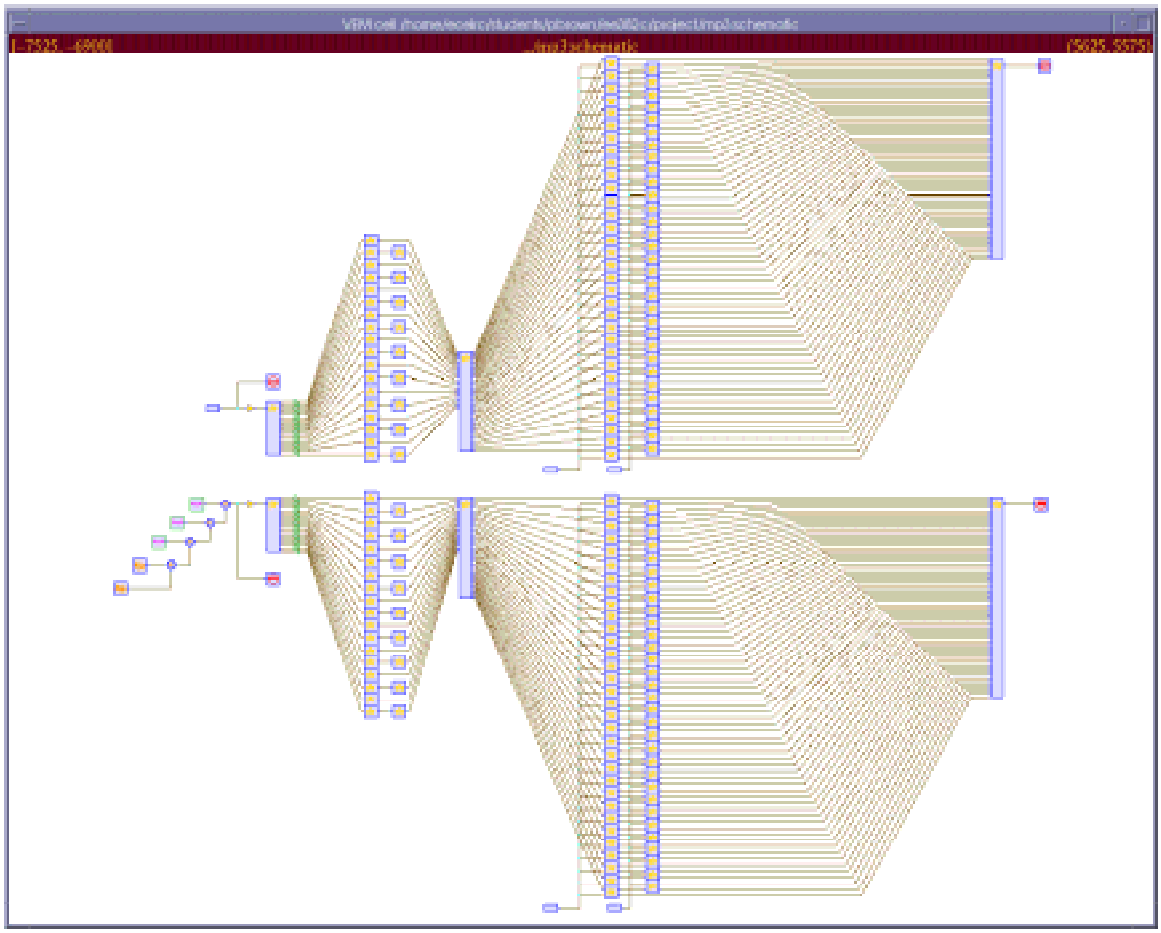

**Figure 2**: The model

The model includes the most important filtering actors. The model does not include the final stages of the algorithm, which involve noise allocation and bit stream formatting to produce the actual MP3 output file. These portions of the algorithm do not involve a large amount of parallel data and, therefore, would benefit least from formal modeling. A truly complete model of an encoder would include these blocks at the far right of the graph, in place of the two red blocks in Figure 2. These red blocks are currently actors that plot the output of the filters in the frequency domain.

The top and bottom halves of the graph represent the two channels (left and right) of audio. The actors on the far left are source actors, included for the purpose of simulation. They generate an input signal composed of three sine waves at different frequencies with two Gaussian random noise sources added. This input provides a wide range of frequency content, similar to what CD-quality music might contain.

Immediately to the right of the source actors is the polyphase filterbank. Barely visible in Figure 2 are the delays (shown in green) on each arc before each of the 18 filters in the filterbank. Each of these delays is a different amount, so that the input to each filter is the same, but phase shifted by 32 samples. As described mathematically in [8], each of the 18 filters produces one sample in each of the 32 frequency bands. The large block following the polyphase filter rearranges the samples into 32 separate frequency bands, each consisting of 18 samples.

Each band is then operated on by a MDCT (Modified Discrete Cosine Transform) actor. This actor requires additional input data from the psychoacoustic model. Both the polyphase filterbank and the MDCT operations are lossy, but the quality lost due to the MDCT is insignificant when compared to the polyphase filterbank [9]. However, the MDCT introduces “aliasing,” which is the result of the overlapping of the frequency subbands. The next column of actors performs the aliasing reduction butterfly, as described in [7]. The butterfly actor also requires data from the psychoacoustic model.

In looking at Figure 2, it is apparent that the model does, in fact, expose a large amount of parallelism. All 18 of the polyphase filters can operate on the input data simultaneously. The rest of the actors can operate on the frequency subbands independently of each other. Therefore, there is a tremendous potential for scheduling onto multiple processors. One of the fundamental laws in parallel computing is known as Amdahl's Law [10], and it helps to illustrate why this algorithm is a good candidate for parallel execution.

- Let $\alpha$ represent the fraction of the algorithm which can be executed in parallel.
- Let P be the total number of processors.
- $S_P$ is then the gain in performance, or "speed up," due to using P processors, and is given by the following formula:

$$S_P = \frac{1}{\alpha/P + (1 - \alpha)}$$

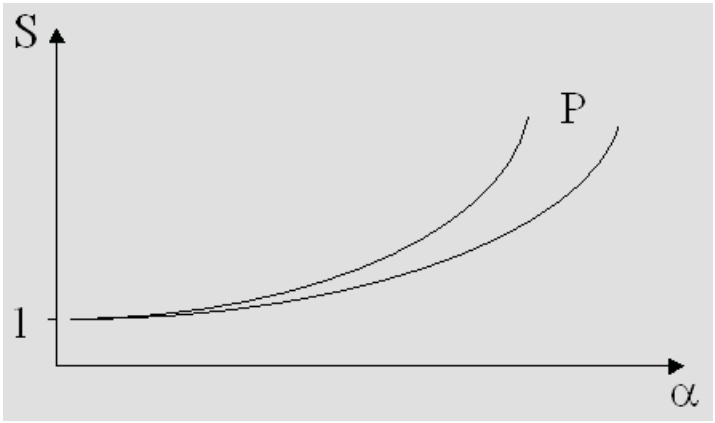


**Figure 3**: Amdahl's Law

Figure 3 shows the speed-up, S, plotted versus $\alpha$. The curve shifts up and to the left as more processors are added. The important aspect of the graph is that $\alpha$ must be large to obtain a significant improvement by increasing P. This is because applications with little parallelism must waste large amounts of time communicating between processors, effectively canceling the benefits of using the additional processors. As demonstrated by this project, the MP3 encoding algorithm has a very high amount of parallelism, $\alpha$.

## Further Work

The model created for this project comprises a large portion of a fully functional MP3 encoder. Future work on this model should include the addition of the final stages of the encoder: noise allocation and bit stream formatting. This would allow the simulation to produce an actual MPEG Layer-3 output file. Also, multiprocessor scheduling techniques should be investigated – Ptolemy has the capability to schedule SDF graphs onto multiple processors. To do this effectively, the user must allocate enough delay on the appropriate arcs in order to maximize the utilization of each processor. Further research in this area could determine the optimum combinations of buffer sizes and number of processors, and possibly even further division of the blocks into smaller actors. Finally, the L.A.M.E. source code includes several NSP (native signal processing) kernels for performing some of the most processor-intensive math routines. The encoder could be benchmarked on various architectures, using these kernels, to determine the merit of employing NSP in MP3 encoding.

## References


1. "The Moving Picture Experts Group Home Page," http://www.mpeg.org/MPEG.
2. "Coding of Moving Pictures and Associated Audio for Digital Storage Media at Up to About 1.5Mbps – Part 3: Audio," *ISO/IEC 11172-3*, Nov. 1991.
3. Davis Pan, "A Tutorial on MPEG/Audio Compression," *IEEE Multimedia Journal*, vol. 2, no. 2, Summer 1995, pp. 60-74.
4. "MPEG Audio Layer-3," Fraunhofer-Gesellschaft Institute, http://www.iis.fhg.de.
5. "The L.A.M.E. Project," http://www.sulaco.org/mp3.
6. "The Ptolemy Project," University of California-Berkeley, http://ptolemy.eecs.berkeley.edu.
7. T Sporer, K Brandenburg, B Elder, "The Use of Multirate Filterbanks for Coding of High-Quality Digital Audio," *6th European Signal Processing Conference,* Amsterdam, June 1992, vol. 1, pp. 211-214.
8. M Kumar, M Zubair, "High Performance Software Implementation of MPEG Audio Encoder," *1996 IEEE International Conference On Acoustics, Speech, and Signal Processing*, paper no. 1628, pp. 1049-1050.
9. J Enerstam, J Peman, "Hardware Implementation of MPEG/Audio Real-Time Encoder," *Master's Thesis*, Lulea University of Technology, Sep. 1998.
10. G M Amdahl, "Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities," *American Federation of Information Processing Societies Conference Proceedings*, vol. 30, Atlantic City, N.J., Apr. 1967, pp. 483-485.