A Real-Time Implementation of MPEG Audio Layer I Decoding on a Fixed-Point DSP Platform

Kwang Rip Hyun, Raja Banerjea, Manchurt Kim, and Haniph Latchman
Department of Electrical Engineering, University of Florida

Subramania I. Sutharsanan
Advanced Development System Design, IBM Corporation

Abstract
As a vital element of multimedia applications, high quality digital audio is of special interest. However, the associated data storage and bandwidth requirements often become prohibitively large. The ISO/IEC standardization body has established the Moving Pictures Expert Group (MPEG) in order to develop an international standard for the coded representation of moving pictures and associated audio. The audio coding algorithm, developed by the ISO/MPEG, has been gaining wide acceptance in industry as well as with audio specialists.

Since high quality music decoding is initially marketed in playback environments, the real-time implementation of the MPEG decoding algorithm is of great interest. In this paper we address several practical issues in the implementation of the algorithm on the widely available TI TMS320C25. These issues include the conversion of floating-point computations to fixed-point without losing much resolution, memory optimization, and others. Since the power of the CPU is limited, an intensive optimization is a must. Due to the same reason, we discuss only the mono decoding.

1. Introduction
1.1. Encoder
A block of the audio encoding process is shown in Figure 1 [ISO, 91]. The transform block maps the incoming signal from time domain to frequency domain. Using a appropriate human hearing model, the psychoacoustic model determines the number of bits required to encode each subband samples and, effectively, controls the total number of bits. It helps remove redundancy in the audio samples and thereby compresses the audio samples.

Figure 1. Block diagram of Encoder

For high bit rate transmission the role of the psychoacoustic model becomes less significant since there is relatively large bandwidth available. The quantizer block uses a linear quantization with symmetric zero representation. The bits are coded using a bit allocation table for layer I and II, and the Huffman codes for layer III. The output of the encoder should be in the form of frames. 8.707 msec worth of audio samples (equiva-

lently, 384 samples for mono and 768 samples for stereo) are encoded into a frame. The structure of a typical frame is found in [Kraehein and Gerhard, 91].

1.2. Decoder
1.2.1. Bit Decoding
The synchronization of the decoder with the incoming bitstream is achieved by first searching in the bitstream for the 12 bit syncword (0x0FFF). The incoming bit-

stream is assumed to be layer I frames since this specific implementation is only for the layer I decoding. In the layer I, the bit allocation information for each subband is represented by a 4 bit coded word (i.e. 0 to 15), which specifies the number of bits that need to be read in for each subband. The decoder performs scale index and sample unpacking only when the bit allocation of that subband is non-zero.

1.2.2. Subband Synthesis Filter
The decoded samples are in the transformed domain. To obtain the matching PCM samples the decoder needs to perform inverse transform of the sample domain and then window. If a subband has no bits allocated to it, the samples in that subband are set to zero. The subband samples for all the 32 subbands have to be calculated.

TP.03  412
ICMC Proceedings 1993
and then fed to the synthesis subband filter to generate 32 consecutive audio samples.

1.1.3. IDCT Computation

The inverse discrete cosine transform of the decoded samples is computed. The inverse filtering may lead to imaginary values we obtain 64 values at this stage. These samples are placed in a 512 long buffer discarding the past 64 samples. This stage of the subband synthesis filter is described in greater detail in Section 2.

1.1.4. Windowing of Data

To generate the fixed PCM samples the IDCT samples have to be filtered by the subband filter. The subband filter consists of a collection of baseband filters shifted in the frequency domain. The baseband filters are designed such that they give an overall smooth frequency response. Instead of computing the filter coefficients every time, all of them are stored as a look up table. The buffer samples are multiplied by the window coefficients and the result gives the PCM samples.

2. Implementation Procedure

2.1. Overview

The first stage of the implementation was to develop a C version of the MPEG audio Layer I algorithm. This C code was compiled using the GCC compiler with an optimizing option. The 'C' compiler has powerful number crunching capabilities, but is not good at logical operations such as comparison and shifting. Therefore, we separated the program into the bit unpacking part which requires logical operations, and the subband synthesis part which is computationally intensive. The bit unpacking is done on the low PCM while the dequantization, denormalization, subband synthesis and window computation are done on the V25. Since the decoding of a frame must be done within 8.707 msec we measured the actual time taken to perform the bit unpacking on three different platforms.

Table 1 shows the results. Allowing a safe margin we decided to move to a faster platform, a 486 PC 33MHz.

<table>
<thead>
<tr>
<th>Type of Computer</th>
<th>total time</th>
<th># of frames</th>
<th>time taken per frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>PS/2 model 286</td>
<td>63.516 sec</td>
<td>2504</td>
<td>0.03176 sec</td>
</tr>
<tr>
<td>PS/2 model 75%</td>
<td>19.176 sec</td>
<td>2504</td>
<td>0.0366 sec</td>
</tr>
<tr>
<td>486 at 33MHz</td>
<td>8.187 sec</td>
<td>2504</td>
<td>0.03276 sec</td>
</tr>
</tbody>
</table>

2.2. Optimization of Dequantization

The input samples are quantized in variable word lengths determined by the psychoacoustic model. The bit allocation index contains how many bits are used to quantize that sample. The bit dequantization takes the information from the bit allocation and dequantizes the subband samples.

SHIPT and ANAC are the only steps involved in the implementation. The efficient use of those logical operations greatly reduced the program execution time. A comparison of the computation time requirements for one frame of samples is shown in Table 2.

2.3. Optimization of Denormalizing Samples

The denormalized samples are multiplied by the scale-tanks, and stored as a 64 element array. The ratio of one element of the next is fixed, at a constant. Therefore, a table lookup method is used to eliminate redundant computing time. Since the relationship of the array elements are exactly known, we can further reduce the size of the table by half without wasting any additional processing power. Refer to Table 2.

2.3.1. Optimization of Subband Synthesis

A DFT is required after the denormalization. We had two options; one was DCT and the other was FFT. Generally, the FFT requires fewer computations than the DCT and obviously smaller memory space.
3.3.3. Comparison of DCT and FFT

The MAC instruction multiplies the operand and the contents of R-register and stores the result in the accumulator in one instruction cycle when it is used along with a FPX instruction. The number of instruction cycles required for the DCT, therefore, is 1024 for 32 samples. In case of the FFT computation, the multiply-and-accumulate instructions cannot be pipelined. This restriction imposes a considerable overhead on the CPU. The processing time for the FFT will be $320 + 415 \times 2 = 1224$ instruction cycles for 32 samples after optimization [Sudarshan, 91].

The TMS320C25's architecture allows us to do DCT computation in a time comparable to or less than the FFT computation. Though the data memory required by the FFT is far less than that by the DCT we decided to go for a DCT implementation since the CPU time requirement was of prime concern. In the practical implementation of the FFT and DCT we found that the time required were

- Original Program = 3168 msec
- DCT time required for one Frame = 6 msec
- FFT time required for one Frame = 64 msec

2.4. Window Computation

A subband synthesis window filter is used to produce the final PCM samples. This filter is related to the filter bank used for the encoding process. The buffer containing the past IDFT of the samples is 512 words long, and the data needs to be accessed in a circular fashion. The buffer values are duplicated in our computation to produce a 1024 word long buffer instead of the 512 word long buffer. This way multiplications can be achieved with the window values without shifting the buffer values or using complicated pointer computations. The use of a long buffer also enables pipelining of the computations, thereby enabling MAC to be used with RPTX. Refer to Table 2 for the time requirements.

2.5. Time Requirements

The total time required on the TMS320C25 is shown in Table 2. This computation was done for 32 subbands. As the time required is more than 8.707 msec the number of subbands was decreased to 18.

2.6. Fixed point implementation

Round-off errors occurring during fixed point computations is always a major concern. We were successful in preventing round off errors from corrupting the MPEG audio decoding on a fixed point platform. This was achieved through proper choice of scale factors in the various stages.

<table>
<thead>
<tr>
<th>Function</th>
<th>Cross Compiled</th>
<th>Optimized Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Denormalize sample</td>
<td>19.3</td>
<td>1.02</td>
</tr>
<tr>
<td>Quantize sample</td>
<td>63.36</td>
<td>0.8</td>
</tr>
<tr>
<td>Subband Synthesis</td>
<td>3168</td>
<td>6</td>
</tr>
<tr>
<td>Window Computation</td>
<td>116</td>
<td>2.568</td>
</tr>
<tr>
<td>Total</td>
<td>3369</td>
<td>10.46</td>
</tr>
</tbody>
</table>

3. Conclusion and Future Work

A single TMS320C25 can barely handle the mono decoding of the MPEG algorithm. In order to add new features to the current implementation an upgrade to a new CPU is necessary. The new CPU should provide a larger internal memory space and faster instruction cycle for more computing power.

Now, the Texas Instrument has TMS320C50, which has larger internal memory, faster clock cycle, new instructions, and etc. [TI, 90], [TI, 91]. At the time of writing this paper we are finalizing a full stereo implementation of the MPEG Layer I decoding on a single 'C50. As the popularity of the MPEG algorithm increases many work should be done including a low-cost MPEG encoder implementation, and etc.

The wavelet transform has proposed interesting new properties, thus collecting a lot of attention from industry and academic world. We are currently investigating the possibility of replacing the Fourier transform with the wavelet transform in the MPEG audio.

4. References


