A HIGH PERFORMANCE SOFTWARE IMPLEMENTATION OF MPEG AUDIO ENCODER

Manoj Kumar[1] *Mohammad Zubair*[1]

[1]IBM T.J. Watson Research Center, Yorktown Hgts, NY, USA

ABSTRACT

The MPEG/Audio is a standard for both transmitting and recording compressed audio. The MPEG algorithm achieves compression by exploiting the perceptual limitation of the human ear. The standard defines the decoding process and also the syntax of the coded bitstream. However, there is room for having different implementations to generate the compressed bitstream. In this paper we propose a high performance software implementation of the MPEG/Audio encoder. We obtained more than a factor of five improvement over a straightforward implementation on the IBM PowerPC, Model 250.

1. INTRODUCTION

The basic structure of the audio encoder as described in the ISO/IEC 11172-3 document [4] is shown in Figure 1. The input audio samples from each channel pass through a subband filter which divides the input into 32 equal-width frequency subbands. The psychoacoustic model calculates the masking threshold for each subband below which noise is imperceptible to human ear. In the bit-allocation procedure, bits are allocated to the subbands according to the masking threshold provided by the psychoacoustic model. The objective of bit allocation is to minimize the maxima of noise to mask ratio, the maximum taken over all channels and subbands. The subband samples are then scaled, quantized according to the bit allocation, and formatted into an encoded stream together with a header, bit allocation and scaling information.

In this paper we give a high performance implementation of the subband filter and bit-allocation procedures. The proposed technique for subband filter computation exploits the architectural features of the RISC machines to get better performance. In particular, we make better use of memory hierarchy, fused multiply-add instruction, and arithmetic pipelines present on a typical RISC machine. The bits are allocated to multiple subbands such that the quantization noise is masked to the greatest extent. This allocation is an iterative process and for each bit allocated, the minimum mask to noise ratio is computed over all subbands of all channels. We describe an efficient implementation of this computation using the heap data structure.

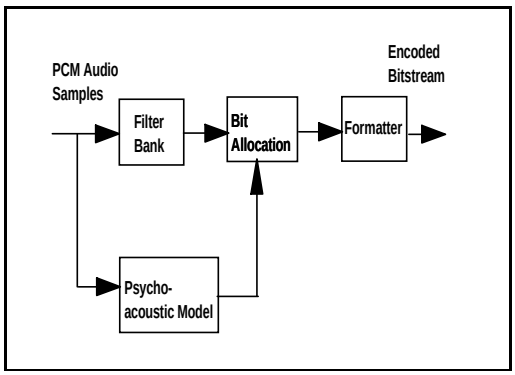

Figure 1. Basic structure of an encoder.

2. SUBBAND FILTER

The analysis subband filter divides the audio signal into 32 equal-width frequency subbands. A high level description of this computation as described in the MPEG ISO/IEC 11172-3 document is given below.

1. Shift in 32 new samples into a 512 point buffer x.

$$x_i = \begin{cases} x_{i-32} & \text{for } i = 511 \text{ down to } 32 \\ \text{next-input} & \text{for } i = 31 \text{ down to } 0 \end{cases}$$

2. Window vector x by the coefficient vector c. The coefficients are given in the Mpeg document.

$$z_i = c_i * x_i, \quad \text{for } i = 0 \text{ to } 511.$$

3. Partial calculation.

$$y_i = \sum_{j=0}^{7} z_{i+64j}, \quad \text{for } i = 0 \text{ to } 63.$$

4. Calculate the 32 output subband samples s_i by matrixing.

$$s_i = \sum_{k=0}^{63} m_{ik} * y_k, \quad \text{for } i = 0 \text{ to } 31.$$

The coefficient for the matrix m_{ik} are given by,

$$m_{ik} = \cos((2i+1)(k-16)\pi/64),$$

$$\text{for } i = 0 \text{ to } 31, \quad \text{and} \quad \text{for } k = 0 \text{ to } 63.$$

Steps 1-3 calculates the polyphase components of the prototype filter. Step 4 is the polyphase implementation of the subband filters. The vector c in Step 2 is the coefficients for the prototype filter of the polyphase filter bank.

Several researchers in the past have reformulated the above computation to reduced the number of operations. This is basically achieved by using a DCT or DFT algorithms for some part of the computation [1, 3, 2]. For example, the formulation in [1, 2] exploits the symmetry of cosine function to replace the 32×64 matrix, m_{ik}, by a 32×32 DCT matrix. (This requires some pre-computation to reduce the size of the y vector to 32.) The operation count can be further reduced by using a fast DCT formulation (FDCT). However, we discovered that the FDCT formulation is of limited advantage because of short DCT lengths (32 point DCT). In fact, as we discuss later, one level of FFT like decomposition of the DCT matrix is enough to give FDCT like performance. We also found that the rest of the computation (Steps 1-3) is a significant part of the overall computation, and hence we focused on those steps. The key reasons for the poor performance of Steps 1-3 are:
(i) The number of loads and store per operation is quite high. Once the data item is brought into the register, there is much not reuse.
(ii) There is no use of fused multiply add store instruction available on most of the RISC machines.

We addressed these problems by restructuring the computation and doing it for a group of 12 blocks of 32 samples together. A high level description of the new technique is outlined below.

1. Shift in 32*12 new samples into a 864 point buffer x.

$$x_i = \begin{cases} x_{i-384} & \text{for } i = 863 \text{ down to } 384 \\ \text{next-input} & \text{for } i = 383 \text{ down to } 0 \end{cases}$$

2. Compute y's for a group of 12 samples as follows.

for $j = 0$ *to* 63 **do**
  $t_k = 0,\ 0 \le k < 12$
  for $i = 0$ *to* 7 **do**
    $\mathrm{m} = i * 64 + j$
    $c0 = c_m$
    $t_0 = t_0 + c0 * x_{m+352};\ t_1 = t_1 + c0 * x_{m+320}$
    $t_2 = t_2 + c0 * x_{m+288};\ t_3 = t_3 + c0 * x_{m+256}$
    $t_4 = t_4 + c0 * x_{m+224};\ t_5 = t_5 + c0 * x_{m+192}$
    $t_6 = t_6 + c0 * x_{m+160};\ t_7 = t_7 + c0 * x_{m+128}$
    $t_8 = t_8 + c0 * x_{m+96};\ t_9 = t_9 + c0 * x_{m+64}$
    $t_{10} = t_{10} + c0 * x_{m+32};\ t_{11} = t_{11} + c0 * x_m$
  end for
  $y_{0,j} = t_0;\ y_{1,j} = t_1;\ y_{2,j} = t_2;\ y_{3,j} = t_3$
  $y_{4,j} = t_4;\ y_{5,j} = t_5;\ y_{6,j} = t_6;\ y_{7,j} = t_7$
  $y_{8,j} = t_8;\ y_{9,j} = t_9;\ y_{10,j} = t_{10};\ y_{11,j} = t_{11}$
end for

3. Calculate the 32*12 output subband samples s_i by matrixing. As outlined in [1, 2], we first exploits the symmetry of cosine function to replace the 32×64 matrix, m_{ik}, by a 32×32 DCT matrix r_{ik}. The coefficient for the DCT matrix r_{ik} are given by,

$$r_{ik} = \cos\left((2i+1)k\pi/64\right),$$

for $i = 0$ to 31, and for $k = 0$ to 31.

The necessary pre-computation to reduce the size of the y vector to an f vector of size 32 is given by,

for $k = 0$ *to* 11 **do**
  $f_{k,0} = y_{k,16}$
  for $j = 0$ *to* 16 **do**
    $f_{k,j} = y_{k,j+16} + y_{k,16-j}$
  end for
  for $j = 17$ *to* 31 **do**
    $f_{k,j} = y_{k,j+16} - y_{k,80-j}$
  end for
end for

As mentioned before, we can reduce the number of operations further by one level of FFT like decomposition of the DCT matrix. This decomposition is clear from the structure of the DCT matrix. The even columns of r are symmetric about their center, and the odd columns are symmetric with a change of sign. The code which uses this property along with the inner-loop unrolling to minimize loads and stores is illustrated below.

for $k = 0$ *to* 11 **do**
  for $i = 0$ *to* 15 **do**
    $s0 = 0$
    $s1 = 0$
    for $j = 0$ *to* 31 in steps of 2 **do**
      $s0 = s0 + r_{i,j} * f_{k,j}$
      $s0 = s0 + r_{i,j+1} * f_{k,j+1}$
    end for
$s_{k,i} = s0 + s1$
$s_{k,31-i} = s0 - s1$
  end for
end for

Note that in the above restructured code we have merged the widowing and partial calculation step (Steps 2 and 3). This allow us to use the fused multiply and add instruction of the PowerPC architecture. Also, by performing computation for a group of 12 samples together, the inner loop of the restructured code consists of (i) twelve independent multiply-add operations, and (ii) a significant reuse of elements of c vector. As a result the arithmetic pipelines of the PowerPC architecture are fully utilized, and the number of loads/store per arithmetic operation is minimized.

3. BIT ALLOCATION

Allocation of bits to a subband reduces the quantization noise in that subband. For each audio frame (384 samples/channel for layer 1 and 1152 samples/channel for layer 2), bits must be distributed across the subbands from a pool of predetermined number of bits. Before this bit distribution is attempted, a psychoacoustic model is used to determine the quantization noise for each subband that will be masked in the human auditory system. This masking level in each subband gives us the masking curve shown in Figure 2, which is a function of short term Fourier transform of the audio frame being coded and the psychoacoustic model used.

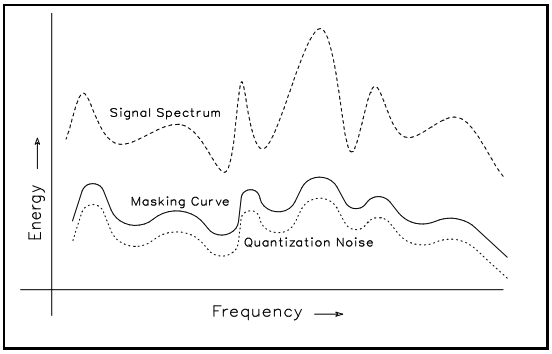

Figure 2. Masking curve.

The objective of the bit allocation process is to minimize the noise to mask ratio (NMR) over all subbands. (Here, noise is the quantization noise introduced by the bit allocation process.) As a result the quantization noise curve after bit allocation is an offset of the mask curve.

The high level description of the bit allocation computation in the MPEG ISO/IEC is an iterative procedure. Bits are allocated one at a time to the subband with the maximum NMR, and that subband's NMR is reduced accordingly. Furthermore, for each bit allocated, the NMR values of all subbands in all channels are scanned to select the subband-channel pair with maximum NMR value. This is a very time consuming process.

We optimize the bit allocation process by using the heap data structure. A heap is a complete binary tree with the property that the value (key) of each node is at least as large as the value of its children nodes (if they exist). For our case, NMR is used as the value. There is a heap element for every channel-subband pair which has yet not received a pre-determined maximum number of bits.

The commonly used operations on a heap are delete and insert. A heap is built from a collection of elements by starting with a one element heap, and iteratively inserting all elements in it. The insert operation is performed by initially placing the element to be inserted at the left most vacant location of the bottom level. Then the element being inserted is exchanged with its parent if its value or key (NMR in our case) is less than that of its parent. We keep exchanging the element being inserted with its parent until it reaches a position where its value is less than that of its parent. The insert process is shown in Figure 3.

An element with maximum value is at the top of the heap. If we delete it, the heap must be rebalanced as shown in Figure 4. The right most element in the bottom level of the heap is moved to the vacant spot at the top of the heap. Then this element is compared with both of its child nodes, and if its key is not greater than that of both child nodes, it is exchanged with the child node having the larger key. This exchange process is repeated until the exchanges take place.

In the bit allocation process, we repeatedly delete the element at the top of the heap, modify its NMR value (key) by allocating a bit to it, and then reinsert this element back into the heap. This delete-insert operation pair can be combined to eliminate the overhead of insert operation. Essentially after deleting the element from top of the heap,

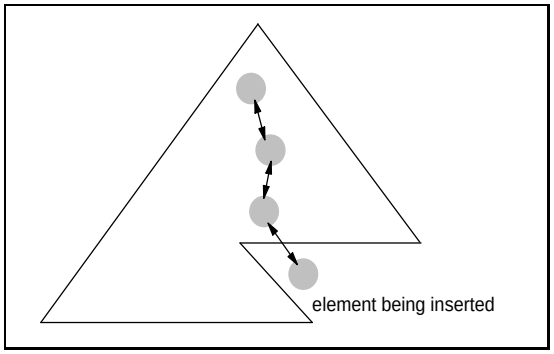

Figure 3. Adding an element to a heap.

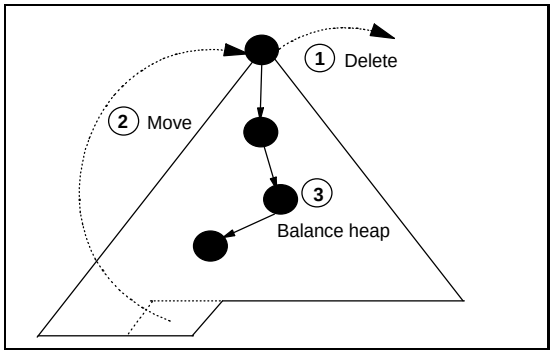

Figure 4. Deleting an element from a heap.

Step 1 in Figure 4, we do not carry out Steps 2 and 3 to balance the heap immediately. When the deleted item has to be reinserted in the heap with a modified NMR value, we bring it back to the top of the heap instead of using Step 2 of the delete process, and then carry out Step 3 of the heap deletion procedure shown in Figure 4.

In joint stereo coding, once we select the channel-subband pair with the maximum NMR value, bits are allocated and NMR values are modified for that particular subband in all channels. Furthermore, any of these channel-subband pairs may get the predetermined maximum number of bits as a result of this allocation, and therefore must be deleted from the heap. We note that the subtree under any element of the heap has all the properties of the heap. Thus, any element of the heap can be deleted by replacing it by the rightmost element in the last level of the heap, and then balancing it as outlined in Step 3 of Figure 4. Similarly, the delete-insert operation pair can be combined for interior heap elements just as is done for the top element of the heap.

The advantages of this approach over the traditional one using a linear search are: (i) for every bit allocated we need to work on $logn$ elements as opposed to n elements (here n is the list size) , and (ii) the heap size becomes smaller as the subband becomes ineligible for further bit allocation resulting in the reduced search time.

4. PERFORMANCE RESULTS

We evaluated the performance of the MPEG audio encoder on a Model 250, 66MHz, PowerPC 601 machine. The performance results are summarized in Figure 4. For comparison, we have also included the performance for the code

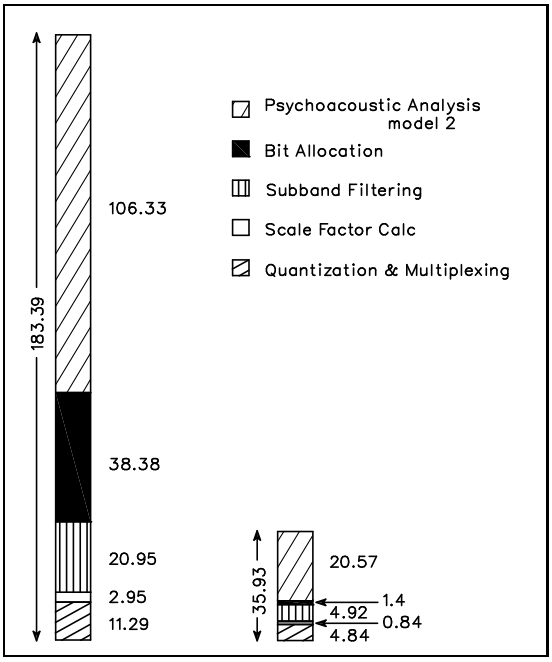

Figure 5. Execution time for encoding a 30 sec. stereo audio sampled at 44.1KHz. The audio is compressed to 384 Kbps using Layer 2 Psychoacoustic Model 2.

based on the description in the MPEG ISO/IEC 11172-3 document. For all our experiments the output of both the implementations were bit compatible. Note that we were able to get a factor of five overall performance improvement over the standard code. A twenty fold improvement was observed for bit allocation procedure. The major factor for this was the new proposed algorithm for bit allocation based on heap data structure. We saw an improvement of a factor of five for subband filter computation. Again, this was due to the proposed new algorithm as outlined in Section 2. The improvement in the Psychoacoustic model routine was due to some major modifications which will be discussed elsewhere. The rest of the modules were optimized using standard optimization techniques.

REFERENCES

[1] D. Pan, "Digital Audio Compression," *Digital Technical Journal*, Vol. 5, No.2, 1993, pp.28-40.

[2] K. Konstantinides , "Fast subband filtering in MPEG Audio Coding," *IEEE Signal Processing Letters*, 1(2), Feb. 1994, pp.26-28.

[3] H.J. Nussbaumer and M. Vetterli, "Computationally Efficient QMF Filter Banks," *Proceeding International Conference IEEE ASSP*, IEEE Press, N.J., 1984, pp. 11.3.1-11.3.4.

[4] International Standard. *Information Technology – Coding of Moving Pictures and Associated Audio Information: Audio*, 1993. ISO/IEC 11172-3.

A HIGH PERFORMANCE SOFTWARE IMPLEMENTATION OF MPEG AUDIO ENCODER

Manoj Kumar[1] *and Mohammad Zubair*[1]

[1]IBM T.J. Watson Research Center, Yorktown Hgts, NY, USA

The MPEG/Audio is a standard for both transmitting and recording compressed audio. The MPEG algorithm achieves compression by exploiting the perceptual limitation of the human ear. The standard defines the decoding process and also the syntax of the coded bitstream. However, there is room for having different implementations to generate the compressed bitstream. In this paper we propose a high performance software implementation of the MPEG/Audio encoder. We obtained more than a factor of five improvement over a straightforward implementation on the IBM PowerPC, Model 250.