A MEMORY AND COMPUTATION EFFICIENT STRUCTURE FOR MPEG POLYPHASE SYNTHESIS

Mohamed F. Mansour

DSP Solutions R&D Center,
Texas Instruments Inc.,
Dallas, TX 75243, {mfmansour@ti.com}

ABSTRACT

We propose a new structure for polyphase synthesis in MPEG-1 audio standard. The algorithm is based on factorizing the DCT matrix in a way similar to FFT factorization. The proposed algorithm reduces the memory requirement significantly even with direct implementation. With optimized implementation, the computational requirement can be reduced considerably as well.

1. INTRODUCTION

MPEG-1 audio standard [4] is the most popular audio standard today. In particular, layer III audio format (mp3) has gained much popularity and become a standard compression technique for audio files on the internet. The polyphase synthesis part of the MPEG audio decoder consumes a significant amount of computational and memory resources primarily because of the matrixing operation with the Discrete Cosine Transform (DCT) matrix. It was reported in [3] that the DCT matrixing operation represents 40% of the overall decoding time. This part is common between all layers in the MPEG-1 standard. It is frequently implemented on a special hardware (usually with small memory size) to accelerate the overall performance. Many efficient structures of the MPEG polyphase synthesis have been proposed in the literature (e.g., [2] and [3]). However, the primary goal of these algorithms was to reduce the computational requirements of the decoding. In this paper, we propose an algorithm that is designed to significantly reduce the memory requirement of the polyphase synthesis. Moreover, with optimized implementation the complexity is significantly reduced as well. The algorithm is based on factorizing the DCT matrix to much smaller matrices using basic trigonometric relations in a way very similar to FFT factorization.

The DCT matrixing described in this paper is common between the three layers of the MPEG-1 standard. A similar approach can be used to simplify the calculation of the IMDCT in layer III decoding. However, the improvement is not significant in this case because of the relatively small IMDCT matrix size.

This paper is organized as follows. In section 2, we give a brief description of the MPEG standard recommendation of the polyphase synthesis filter. The proposed algorithm is described in section 3 and the efficient implementation of the algorithm is desribed in section 4.

2. MPEG RECOMMENDATION

The MPEG standard recommendation adopted the polyphase implementation of the cosine-modulated filter banks ([1], chapter 8). The overall decoder structure is illustrated in figure 1.

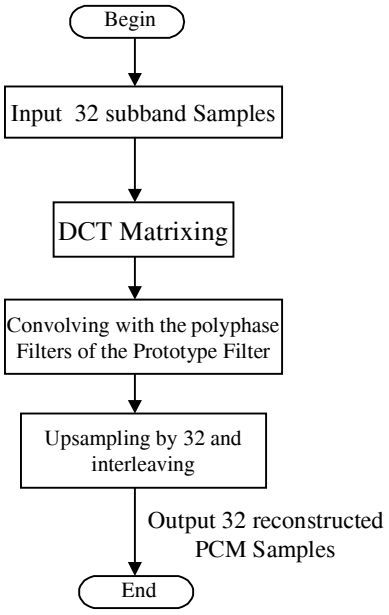

Fig. 1. MPEG Standard Polyphase Synthesis

The DCT matrixing in the polyphase synthesis is performed using a direct matrix multiplication :

$$for\ i = 0\ to\ 63\ \ V_i = \sum_{k=0}^{31} N_{ik}.S_k \qquad (1)$$

where S_k are the input 32 subband samples, and

$$N_{ik} = \cos((i+16)(2k+1)\frac{\pi}{64}) \qquad (2)$$

This above matrix multiplication requires 2048 multiply and accumulate operations (MACs). If the symmetry of N_{ik} is exploited then the total calculation may be reduced to 1088 MACs. The size of the DCT matrix is 2048 words. If the symmetry is exploited then only 1088 words need to be stored. It should be mentioned that the common fast algorithms of DCT using FFT calculation are generally inefficient in the above implementation because of the relatively small size of the DCT matrixing.

Note that, the following part of the polyphase synthesis that includes convolving with the polyphase representation of the prototype filter followed by interpolation cannot be optimized further with the approach described in this paper.

3. PROPOSED ALGORITHM

3.1. Algorithm

The idea of the algorithm is to factorize the DCT matrix using basic trigonometric relations in a way very similar to FFT factorization. The 32-element sum is reduced to two summations with order four and eight.

The first step is to put the data in matrix form using row reordering. The 32-elements input data vector is reordered as an 4×8 matrix, and the 64-elements output vector is reordered as an 8×8 matrix. Let $i = 8p+q$, where $p, q = 0$ to 7, and let $k = 8l + m$, where $l = 0$ to 3, and $m = 0$ to 7, then

$V(p,q) = \sum_{m=0}^{7}\sum_{l=0}^{3} S(l,m)\cos((8p+q+16)(16l+2m+1).\pi/64)$

after straightforward trigonometric manipulation and removing the terms that are multiple of 2π, we have,

$\cos(8p+q+16)(16l+2m+1)\pi/64) =$

$\cos(lq\pi/4)\cos((2m+1)(q+16).\pi/64 + p(2m+1)\pi/8) -$

$\sin(lq\pi/4)\sin((2m+1)(q+16).\pi/64+p(2m+1)\pi/8)$

Define,

$$G_c(q,m) = \sum_{l=0}^{3} S(l,m)\cos(\frac{lq\pi}{4}) \qquad (3)$$

$$G_s(q,m) = \sum_{l=0}^{3} S(l,m)\sin(\frac{lq\pi}{4})$$

then ,

$V(8p+q) = \sum_{m=0}^{7} G_c(q,m)\cos((2m+1)(q+16).\pi/64 + p(2m+1)\pi/8) - \sum_{m=0}^{7} G_s(q,m)\sin((2m+1)(q+16).\pi/64 + p(2m+1)\pi/8)$

Note that, the transform in (3) is a nonstandard discrete cosine and sine transforms. They are similar to type-I 4-point DCT and DST [5], but the numerator in this type is "$N-1$" (i.e., 3) rather than N (i.e., 4) as in (3).

Similarly the two cosine and sine terms in the above relation can be simplified further using basic tringonometric relations. Then define,

$$\begin{aligned} G_{cc}(q,m) &= G_c(q,m)\cos((2m+1)(q+16)\pi/64) \\ G_{cs}(q,m) &= G_c(q,m)\sin((2m+1)(q+16)\pi/64) \\ G_{sc}(q,m) &= G_s(q,m)\cos((2m+1)(q+16)\pi/64) \\ G_{ss}(q,m) &= G_s(q,m)\sin((2m+1)(q+16)\pi/64) \end{aligned} \qquad (4)$$

with q and $m = 0$ to 7.

Finally, we have,

$$\mathrm{V(p,q)} = \sum_{m=0}^{7}(G_{cc}(q,m) - G_{ss}(q,m))\cos(p(2m+1)\pi/8) - \sum_{m=0}^{7}(G_{cs}(q,m) + G_{sc}(q,m))\sin(p(2m+1)\pi/8) \qquad (5)$$

Note, these are again nonstandard DCT and DST transforms although they are similar to Type-II DCT and DST [5]. However, in type-II DCT and DST the denominator is $2N$ (i.e., 16) rather than N.

The proposed algorithm is summarized in (3), (4), and (5) and it is illustrated in figure 2.

3.2. Memory Requirements

For the proposed algorithm we will need the following constant memory storage:

1. $\{cos(lq\pi/4), sin(lq\pi/4)\}_{l=0:3,q=0:7}$, i.e., it requires 64 words for direct implementation. However, if symmetry is exploited then the total requirement becomes only 32 words.

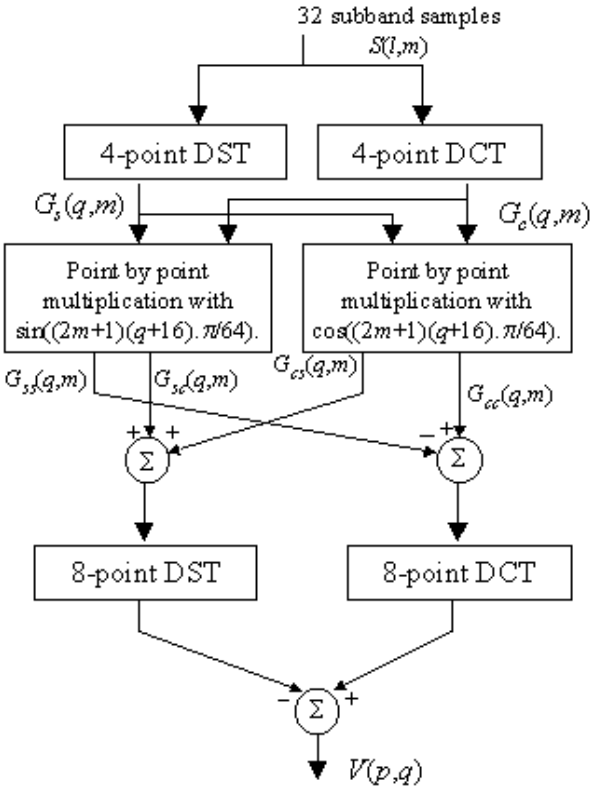

Fig. 2. DCT Matrixing Scheme

2. $\{cos((2m+1)(q+16)\pi/64), sin((2m+1)(q+16)\pi/64)\}$, with m and $q = 0$ to 7, i.e., it requires 128 words.

3. $\{cos(p(2m+1)\pi/8), sin(p(2m+1)\pi/8)\}$, with m and $p = 0$ to 7, i.e., it requires 128 words. However, if redundancy is exploited, then the total memory requirement becomes 64 words.

Therefore, the constant memory requirement of the proposed algorithm is 224 words. Also, we need an extra 64 words for storing $G_c(q,m)$ and $G_s(q,m)$ simultenously. Hence, the total requirement is 296 words. The memory requirement in the MPEG recommendation is 1088 words, i.e., we have a saving factor of 3.68. Further reduction in the requirements is described in section 4.

3.3. Computation Requirements

In this subsection, we calculate the computational load of the direct implementation that does not exploit the structure of the transforms in (3) and (5). In the next section we will exploit the structure of the transforms to significantly minimize the computational requirements.

The computational requirements for each step of the algorithm are as follow:

1. $G_c(q,m)$ and $G_s(q,m)$ computations require 4 MACs per element in a total of 512 MACs. However, both transforms are symmetric. Therefore, it can be reduced to only 256 MACs.

2. The computation of $(G_{cc} - G_{ss})$ and $(G_{cs} + G_{sc})$ require in general 256 MACs.

3. The final step (5) will require 1024 MACs by direct calculation. However, only 512 MACs are needed if the symmetry is exploited.

Hence we have a total of 1024 MACs for the direct implementation of the new algorithm which is the same as the computational requirement of the MPEG standard recommendation [4].

3.4. Comments

The proposed algorithm has another advantage for fixed-point implementation which is usually used in real time signal processing. As discussed in [6], the variance of the quantization error is linear with the summation order. This order equals 32 in the standard MPEG representation, whereas it is only 13 for the proposed algorithm, i.e., the quantization error is reduced by 60%, which is significant in low-precision implementation.

Note that, the 8-point DCT can be reduced further to two stages of 4-point and 2-point DCT and DST. However, the saving in the computation does not justify the increase in the complexity and code size.

It should be mentioned that the proposed algorithm does not change the polyphase reperesentation of the baseband prototype filter. It follows the proposed implementation in the standard.

4. EFFICIENT IMPLEMENTATION

A closer look at the nonstandard transforms in (3) and (5) reveals some useful features that considerably simplify the computational and memory requirements. We will discuss in detail the first transform, then the others are very similar.

1. $\{cos(lq\pi/4)\}_{l=0:3,q=0:7}$: all the nonzero components in the base vectors are either ± 1, or $\pm\frac{1}{\sqrt{2}}$. The transform matrix has the structure:

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \\ 1 & 0 & -1 & 0 \\ 1 & \frac{-1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 1 & -1 & 1 & -1 \\ 1 & \frac{-1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \\ 1 & 0 & -1 & 0 \\ 1 & \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \end{pmatrix}$$

if multiplication by $\frac{1}{\sqrt{2}}$ is delayed after adding/subtracting the corresponding components, it is straightforward to verify that the total computational requirements for each column in $G_c(q,m)$ is 11 additions and 2 multiplications. Hence, the total computational requirement for $G_c(q,m)$ with q and $m = 0$ to 7, is 88 additions and 16 multiplications. The optimized implementation is illustrated in figure 3.

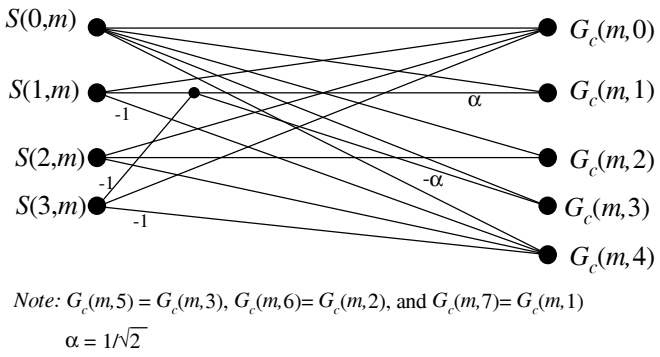

Fig. 3. Recommended Implementation for $G_c(q,m)$

2. $\{\sin(lq\pi/4)\}_{l=0:3,q=0:7}$: This matrix can be optimized significantly be removing the zero rows. The nonzero components in the matrix are either ± 1, or $\pm\frac{1}{\sqrt{2}}$. It is straightforward to show that, the total computational requirement for $G_s(q,m)$ is 56 additions, and 8 multiplications (where we assumed that sign inversion is equivalent to one addition).

3. The calculations of G_{cc}, G_{cs}, G_{ss},and G_{sc} are not easy to optimize. It requires, in general, 256 additions and 256 multiplications.

4. $\{cos(p(2m+1)\pi/8)\}_{p=0:7,m=0:7}$: The only nonzero values in the matrix are: $\pm 1, \pm\frac{1}{\sqrt{2}}, \pm\cos(\frac{\pi}{8})$, and $\pm\cos(\frac{3\pi}{8})$, and it is anti-symmetric around the middle row. Therefore, the total computational requirement for the transform is 248 additions and 40 multiplications.

5. $\{\sin(p(2m+1)\pi/8)\}_{p=0:7,m=0:7}$: The structure is very similar to the above case, however, it is symmetric. Therefore, the total computational requirement is 224 additions and 40 multiplications.

In the following table, we compare the performance of the proposed implementation with the standard MPEG recommendation (when the redundancy in the standard is exploited).

	Standard MPEG	Proposed Algorithm
Multiplications	1088	360
Additions	1088	872
Memory	1088	296

Table 1. Comparison with MPEG Standard Recommendation

5. CONCLUSION

In this paper, we described a new structure for DCT matrixing in the MPEG-1 audio decoder. The structure is based on factorizing the DCT matrix to smaller size matrices in a way similar to FFT factorization. This factorization results in a memory saving factor of about 3.7. Moreover, with the efficient implementation described in the paper, the total computation was reduced considerably. Also, the proposed algorithm significantly reduces the quantization error in fixed-point implementation which is a critical issue in limited precision implementations.

6. REFERENCES

[1] P. Vaidyanthan, " Multirate Systems and Filter Banks", Prentice Hall, 1993.

[2] D. Chan, J. Yang, and C. Fang, " Fast Implementation of MPEG Audio Coder using Recursive Formula with Fast Discrete Cosine Transforms", IEEE Trans. on Speech and Audio, Vol. 4, No. 2, pp. 144-148, March 1996.

[3] K. Konstantinides, " Fast Subband Filtering in MPEG Audio Coding", IEEE Signal Processing Letters, Vol. 1, No. 2, pp. 26-28, February 1994.

[4] ISO/IEC International Standard 11172-3, " Information Technology- Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbits/s- Part 3: Audio"

[5] G. Strang, " The Discrete Cosine Transform", SIAM Review, No. 41, pp. 135-147, 1999.

[6] J. Proakis, and D. Manokalis, "Digital Signal Processin: Principles, Algorithms, and Applications", chapter 7, 3rd edition, Prentice Hall, 1996.