Implementation of IMDCT Block of an MP3 Decoder through Optimization on the DCT Matrix

Miroslav GALABOV

Dept. of Computer Systems and Technologies, University of Veliko Turnovo, T. Tarnovski 2, 5003 V. Turnovo, Bulgaria

plam@vali.bg

Abstract. *The paper describes an attempt to create an efficient dedicated MP3-decoder, according to the MPEG-1 Layer III standard. A new method of Inverse Modified Discrete Cosine Transform by optimization on the Discrete Cosine Transform (DCT) matrix is proposed and an assembler program for Digital Signal Processor is developed. In addition, a program to calculate DCT using Lee's algorithm for any matrix of the size 2^M is created. The experimental results have proven that the decoder is able to stream and decode MP3 in real time.*

Keywords

IMDCT, MP3 decoder, bitstream, DSP.

1. Introduction

MPEG-1 Layer III is a highly compressed digital audio format. The Moving Pictures Expert Group standardized it in 1991. MPEG was formed in 1988 by ISO/IEC organizations to develop generic coding standards of moving pictures, audio and systems containing both [1]–[2].

2. Functionality

The input to the MP3 decoder is a bitstream and the decoding process turns this into samples which are sent to a digital to analog converter (DAC) add-on board. The MP3 stream is divided into frames where each frame contains 27 ms of music data. The decoding process can be divided into six blocks and these are implemented independent each other (Fig. 1).

The first three blocks – Huffman decoding, descaling and reordering – convert the original MP3 bitstream into 576 frequency lines divided into 32 subbands with 18 frequency lines in each. This is the format which the last three blocks – antialias, Inverse Modified Discrete Cosine Transform (IMDCT) and filter bank – use. These blocks transform the MP3 from the frequency domain into the time domain. A more in-depth description of the decoding process as well as MP3 encoding can be found in [1-2].

The task of the Huffman decoder is to transform incoming compressed data into scalefactors and symbols representing the 576 original frequency lines. The scalefactors are then used in the next block (the descaling block) to rescale the symbols into non-scaled frequency lines.

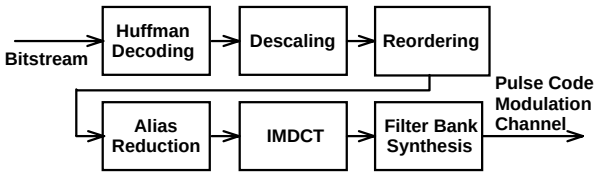


Fig. 1. The MP3 decoding process.

2.1 Huffman Decoder

The decoder compares the input sequence with information in the Huffman table and produces a symbol when a match is found. Information on what table to use for any given frame is found in the frames side information. Output from the Huffman decoder is 576 scaled frequency lines (symbols) which are divided into three partitions:

- *Big-values* contain the lowest frequency lines and are coded with the highest precision. Normally the scaled value is between -15 and 15, but higher precision can be obtained by using an escape sequence. When the decoder finds the value 15 it assumes that higher precision is needed and reads additional bits from the input stream. This value is then added to the original value of 15. The number of bits is specified in the Huffman table.
- *Countl* represents the higher frequency lines and does not need the high precision, they are simply coded with the values 1, 0 and –l.
- *Rzero* represents the highest frequency lines. They have simply been removed by the encoder. These values are filled with zeros by the decoder.

2.2 Descaling

The descaling block rescales Huffman decoded scaled and quantized frequency lines. The result from the Descalizer is the original frequency lines. What equations to use

depend on the windowing function used in the encoding procces? The descaling step must be performed once for each frequency line in the bitstream.

2.3 Reordering

The reorder block has one task; it reorders the frequency lines within a granule. The way that the frequency lines are reordered depends on flags in the side information header. The block works in three different ways depending on the side information header:

- All frequency lines are reordered.
- Only frequency lines after the line number 36 are reordered.
- No frequency lines are reordered.

2.4 Antialias

The antialias block attempts to reduce the inevitable alias effects introduced by the use of a non-ideal bandpass filter. This reduction is done by merging frequencies using butterfly calculations.

2.5 IMDCT

IMDCT reproduces, in cooperation with the synthesis polyphase filter bank, time samples from frequency lines. Given the frequency lines X_k, time samples x_i can be obtained by using the following equation:

$$x_i = \sum_{k=0}^{n/2-1} X_k \cos\left[\frac{\pi}{2n}\left(2i+1+\frac{n}{2}\right)(2k+1)\right],\ 0 \le i \le n \quad (1)$$

In our case, $n = 36$, which means that the IMDCT takes 18 frequency lines as an input and generates 36 polyphase filter sub-band samples. These samples are multiplied with a 36-point window before they can be passed on to the next step in the decoding process. Windowing contains four different types of windows; the types are normal, short, start and stop. Information on what type to use is found in the side information part of each frame. Depending on window type two different implementations are used.

Producing 36 samples from 18 frequency lines means that only 18 of the samples are unique. Therefore IMDCT is said to use a 50 % overlap. The 36 values from the windowing operation are divided into two groups, a low group and a high group, containing 18 values each. Overlapping is carried out by interleaving (adding) values from the lower group with corresponding values from the higher group from the previous frame.

The general view of the "operation flow" is shown in Fig. 2.

The entire cosine term for each output can be treated as a known constant factor based on the combination of i and k in (1). Some investigation of these terms shows the symmetry between the different x_i. Only half of the values are uniquely determined. The rest can be obtained as a function of the previously calculated terms. The following symmetry holds:

$$x_i = -x_{n/2-i-1},\ i = 0 \ldots \frac{n}{4} - 1 \quad (2a)$$

and

$$x_i = x_{3n/2-i-1},\ i = \frac{n}{2} \ldots \frac{3n}{4} - 1\ . \quad (2b)$$

Therefore, calculating the first quarter and the third quarter of all values will be enough to determine the entire set.

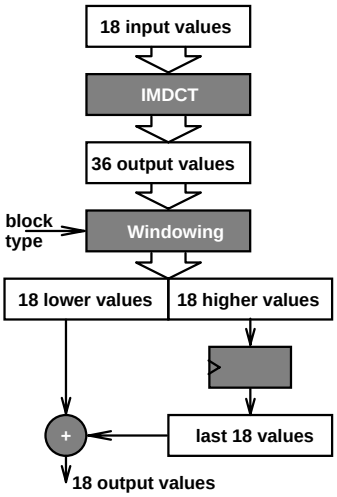


Fig. 2. IMDCT with windowing and overlap add.

2.6 Filter Bank

The synthesis of a polyphase filter bank is the final step in the decoding process [5]. It exploits aliasing and windowing to move the subbands back into their frequency domain origins. The process is naturally divided in two parts, an MDCT part for translating the aliased subband signals, and a windowing part to filter out the undesired aliasing in the translated signal.

- Modified discrete cosine transform: The sub-samples from the transpose block are ordered is such a way that the 32 first values are the first sub-sample from each subband, the next 32 are the second sub-sample and so forth. The MDCT processes 32 values at a time using the following equations:

$$Y_i = \sum_{k=0}^{31} N_{ik} S_k \quad (3a)$$

and

$$N_{ik} = \cos\left[\frac{\pi}{64}(16+i)(2k+1)\right], \quad (3b)$$

where S_k are input PCM samples.

The output values Y are stored in a barrel shifter.

- Windowing: Multiplying the values from the barrel shifter with the window function, specified in the ISO standard. 32 Pulse Code Modulation (PCM) samples are computed each iteration. The MDCT and windowing are performed 18 times for each granule, resulting in 576 PCM samples, i.e. 27 ms at 44.1 kHz.

3. Software Implementation of the IMDCT Decoder Block

The IMDCT used in MP3 is an 18-point DCT that produces 36 output values from 18 input values. The IMDCT can be implemented many ways [2]. The optimization on the DCT matrix is the technique currently used by the decoder.

3.1 Optimizations on the DCT Matrix

A program was developed with the purpose of examining the DCT matrix and generating the source code directly. The program tries to minimize the number of multiplications needed by keeping track of previous calculations. If it finds two identical multiplications, the result from the first calculation will be fetched and used. It can also find summations of two identical multiplications, where it can remove both multiplications and use the previously calculated summation instead. This significantly increases the computation speed and the required number of multiplications will drop from 648 to 256. Thanks to the architecture of the signal processors, the summations can be done in parallel using the multiply and accumulate (MLA) instruction, reducing the number of additions from 612 down to zero. The synthesis polyphase filter bank uses the output from the IMDCT calculation. According to the ISO/IEC 11172-3 standard [1], the synthesis polyphase filter bank must transpose the input before it is being used. To avoid this, the output from the DCT is rearranged so it can be fed directly to the synthesis polyphase filter bank instead.

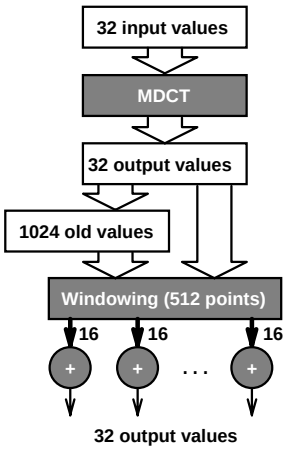


Fig. 3. The subband synthesis filter bank divided into two parts.

The frequency inversion has been integrated in the IMDCT calculation. This improves performance, but it is not crucial to the implementation.

The implementation of the synthesis polyphase filter bank can be divided into two parts. The first part is a 32-point modified DCT calculation and the second part is the windowing and summation of 512 values to produce 32 output samples. A flowchart of the operations can be found in Fig. 3.

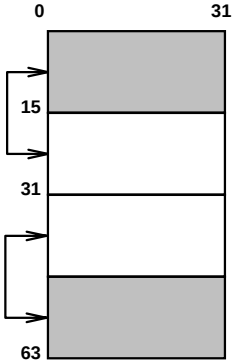


Fig. 4. Modified DCT matrix redundancy.

A 32-point modified DCT requires 32 × 32 multiplications using a non-optimized calculation method. Algorithms performing fast DCT computations are available, and based on the symmetry of the DCT matrix. The problem is that the modified DCT used by the subband synthesis has to be translated into a normal DCT before optimizations can be applied.

The difference between normal DCT and the modified DCT is the transform matrix being used. For a normal DCT, the transform matrix N_{ik} for a DCT with size M can be calculated from the following equation:

$$N_{ik} = \cos\left[\frac{\pi}{2M}(2i+1)k\right], 0 \le i \le 31 \;\; 0 \le k \le 31 \cdot \tag{4}$$

The matrix used in the original subband synthesis has the following properties:

$$N_{ik} = \cos\left[\frac{\pi}{2M}(16+i)(2k+1)\right], 0 \le i \le 63 \;\; 0 \le k \le 31 \cdot \tag{5}$$

It can easily be converted to a 32×32 matrix by the observation that it is uniform, leading to 50 % redundancy. The first and last 16 tines in the matrix are identical with the 32 center-most lines, as shown in Fig. 4.

The reason for using a 64×32 matrix is that the modified DCT produces 64 output values, also with 50 % redundancy, which will be fed to the windowing block. When the size of the DCT matrix is reduces the 32 output values still need to be doubled in a way shown by Fig. 5.

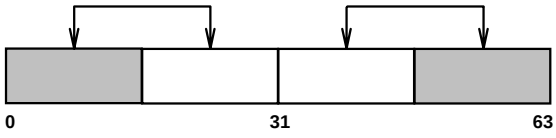


Fig. 5. Converting 32 output values from the DCT into 64 values.

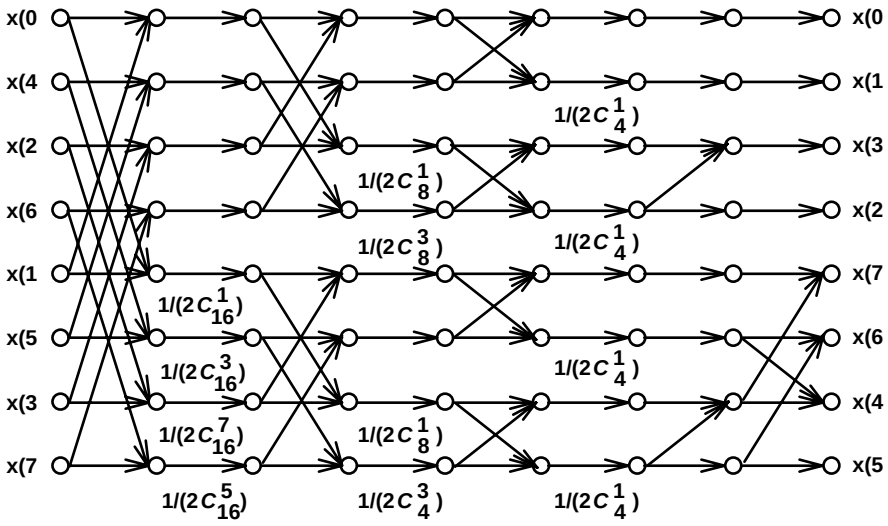
Before any optimization techniques can be used, the modified DCT mast is mapped into a standard DCT. By

plotting N_{ik} for both transforms, identical lines and columns can be found.

The information in the modified DCT can also be found in the normal DCT. Row 1, 2, 3 etc in the modified matrix can be mapped to row 31, 30, 29 etc in the normal DCT matrix if they are negated. Row 0 in the normal DCT matrix is all zeros and can therefore be neglected.

Next, we have to find a suitable optimization method.

One of these methods was proposed by B.G Lee in 1984 [3], decreasing the required number of multiplication needed from 1024 down to 80 for a 32-point IDCT calculation. This leads to a speed gain approximately 13 times. Lee's algorithm is based on the divide-an-concur pattern, splitting large DCT into a series of smaller DCTs (Fig. 6).



$$C_k^i = \cos \pi \left(\frac{i}{k} \right)$$

Fig. 6. Eight point DCT using Lee's algorithm.

A program was developed to auto-generate the source code using Lee's algorithm for any matrix of size 2^M. The program can generate DCT transforms as well as inverse DCT transforms. Built-in memory optimization keeps the temporary variables needed to a minimum.

4. Experimental Results

In order to test the decoder, MPEG supplies a number of test sequences. Most test sequences are sine sweeps with various amplitudes. During the experiment, we use the developed assembler program of decoder for DSP TMS 320C50, Texas Instruments DSP Starter Kit and test sequences according to the ISO/IEC11172-3 standard [1].

A supplied signal is sine sweep from 20 Hz to 10 kHz with the amplitude of –20dB (full scale is normalized to the range between –1 and +1). Two tests only verify computational accuracy of implementation, and not functionality.

The first criterion states that the maximum allowed difference between two 16-bit waveforms is ±3 [1]. The difference between the signals does not exceed ±2 according to the plot.

The second criterion states that the root mean square value may not exceed 0.00000881 [1].

In our case, we are 8192 sampling points on the waveforms containing 5 errors, which give 0.00000095367.

Both criterions passed the test. The decoder producing output above is valid ISO/IEC 11172-3 MP3 decoder.

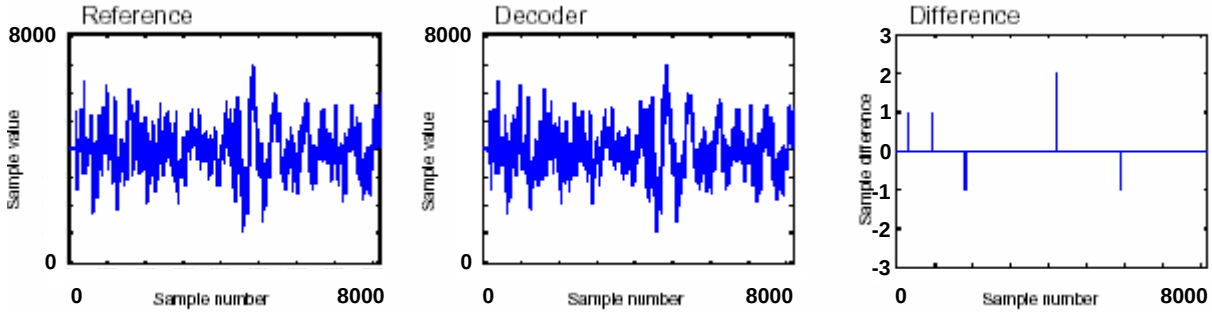


Fig. 7. Two waveforms produced with the reference source code and our decoder and the difference between.

5. Conclusions

Already at the beginning of the article, critical decisions concerning what algorithms of IMDCT to use had been made. Many fast algorithms were found, but choice was done to use the most straightforward one, due its low complexity compared to another more sophisticated algorithms. Loss of efficiency could be accepted to as quickly as possible achieve working architecture in the first hand; especially that loss of efficiency was not critical for the entire design.

The result is an efficient audio decoder, capable of streaming and decoding MP3 in real time. The software decoder complies with the quality requirements in the ISO/IEC 11172-3 standard.

References

[1] ISO/IEC 11172-3:1993 Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s 1993.

[2] GADD, S., LENART, T. A hardware accelerated mp3 decoder with Bluetooth streaming capabilities. *Master's thesis*, Lund Institute of Technology, Sweden, 2001.

[3] LEE, B.G. A new algorithm to compute the discrete cosine transforms. *IEEE Transactions on Acoustics, Speech and Signal Processing*. 1984, vol. 32, no. 6, p. 1243–1245.

[4] BRITANAK, V., RAO, K. R. A new fast algorithm for the unified forward and inverse mdct/mdst computation. *Signal Processing*. 2002, vol. 82,no. 3, p. 433–459.

[5] PAIK, W., HWANG, S. Design of a novel synthesis filter for real-time MPEG-2 audio decoder implementation on a DSP chip. *IEEE Transaction on CE*. 1999, vol. 45, no. 4, p. 1119–1129.

Acknowledgements

Research described in the paper was financially supported by the University of Veliko Turnovo.

About Authors...

Miroslav Galabov was born in Veliko Turnovo. He received his M.S.E degree in radio television engineering from the Higher Naval School "N. Vapcarov",Varna, Bulgaria, in 1989. After that he worked as design engineer for Institute of Radio Electronics, Veliko Turnovo. From 1992 to 2001 he was assistant professor at Higher Military University, Veliko Turnovo. He received his Ph.D. degree in automation systems for processing of information and control from the Higher Military University, in 1999. Since 2002 he has been assistant professor in Computer Systems and technologies department at "St. Cyril and St.Methodius" University of Veliko Turnovo. He is the author of three textbooks, and over 20 papers. His current interests are in signal processing, DSP and multimedia.