

ECE253: Final Project

Embedded MP3 Player

By:

Arup De (Perm #8730400)

Achutam Murarka (Perm #8730319)

{arup_de, [achutammurarka](mailto:achutammurarka@umail.ucsb.edu)}@umail.ucsb.edu

1. Introduction

MP3 (MPEG 1 layer 3) is a standard for compressing digital audio. It was developed by the Moving Picture Experts Group (MPEG). It is one of the dominant and widely used digital formats today. The use in MP3 of a lossy compression algorithm is designed to greatly reduce the amount of data required to represent the audio recording and still sound like a faithful reproduction of the original uncompressed audio for most listeners.

Here in this project we have implemented an MP3 decoder using Spartan 3E Starter kit and an amplifier from Digilent Inc. on Microblaze processor. Our main implementation on the FPGA processor is based on the MAD (MPEG Audio Decoder) Player freely available under the GNU Public License.

2. Design Overview

The block level view of an MP3 decoder is as given below.

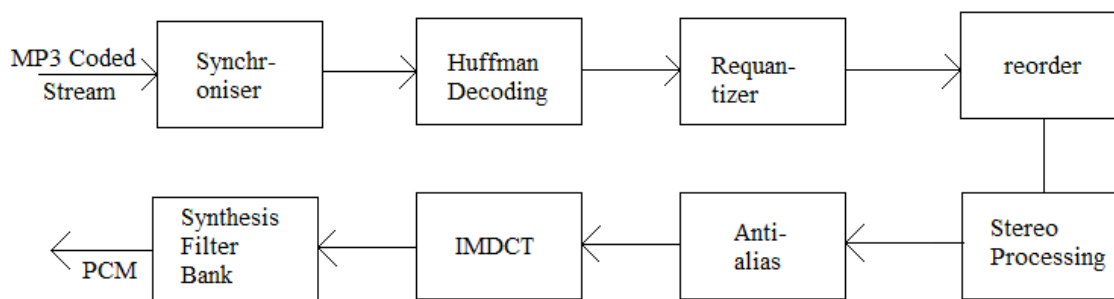


Fig 1. Block Level View of an MP3 Decoder

The main stages in the decoding of an MP3 file format is as given below:

1. Synchronizer
2. Huffman Decoding
3. Requantizer
4. Reordering
5. Antialias
6. IMDCT
7. Synthesis Filterbank

2.1 Synchroniser

Frame information is required to know when the given frame ends and when the next frame starts. It is basically used to derive the information related to the frame correctly. Structure of typical a frame is as given below.

Header	CRC	Side Info	Main Data	Misc. Data
--------	-----	-----------	-----------	------------

2.2 Huffman Decoding

Huffman Decoding is an entropy encoding algorithm used for lossless data compression. The decoding is based on Huffman tables that are used to map the Huffman codes to symbols. Symbols that occur less frequently are coded with longer Huffman codes and those that occur more frequently are shorter in length. The output of the Huffman decoder is 576 frequency lines for each granule.

2.3 Requantizer

The decoded symbols obtained from the previous decoding step is then reconstructed into original frequency lines using the scalefactors in the side information preset in the frame header. These frequency lines are further sub-divided into 21 divisions called scalefactor bands, each of which has its own scalefactor.

2.4 Reordering

During the encoding process, the MDCT can arrange the output in two different ways. Normally the output from the MDCT is sorted by subbands in increasing frequency. When a short block is decoded, a short window will be used. The output will in this case be sorted on subbands, then on windows and then on increasing frequency. Reordering is only applied on short blocks, sorting the frequency lines first by subbands and then by frequency.

2.5 Stereo Processing

Dual channel decoding was used by in the MP3 decoder out the three possible options of single, dual, stereo options.

2.6 Antialias

Antialias is an attempt for reducing the inevitable alias effects because of using a non-ideal bandpass filtering in the subband synthesis block of encoder. The alias reconstruction calculation consists of butterfly calculations for each subband.

2.7 IMDCT

The Inverse Modified Discrete Cosine Transform used in MP3 is an 18-point DCT that produces 36 output values from 18 input values. The equation for IMDCT is as given below.

$$xi = \sum_{k=0}^{\frac{n-1}{2}} X_k \cos\left[\frac{\pi}{2n}\left[2i+1+\frac{n}{2}\right](2k+1)\right]$$
$$0 \leq i \leq n-1$$

It does not produce more information than it is provided with, but it disperses the result from the calculation on a larger number of values. This is why the term modified is being used. The DCT is inverse because it produces time samples from frequency lines. The transformation from the frequency domain to the time domain is done in conjunction with the synthesis polyphase filter bank. Instead of directly producing time samples, the IMDCT generates polyphase filter subband samples

from the input frequency lines. These will later be used for creating time samples. The 36 calculated values from the IMDCT must be multiplied with a 36-point window before it can be used by the next step in the decoding process. There are four different window types that can be applied to the output samples. The window to use is based on the block type, and the block type can be found in the side information. The reason for generating 36 output values is that the IMDCT uses a 50% overlap. The lower 18 values are added with the higher 18 values from the previous frame, and used as output. The higher 18 values are then stored and used the same way when the next frame is being decoded.

2.8 Synthesis Filterbank

It is the last step in the decoding process where 32 PCM samples are produced from 32 subbands. In the synthesis process, the 32 subband values are transformed to the 64- value V vector using matrixing. The V vector is pushed into a FIFO which stores the last 16 V vectors. A U vector is created from the 32 component blocks in the FIFO and a window function D is applied to U to produce the W vector. The reconstructed PCM samples are obtained from the W vector by decomposing it into 16 vectors each 32 values in size and summing these vectors.

3. Implementation Details

Our implementation is based on MAD library from [3]. MAD is a high-quality MPEG audio decoder. It currently supports MPEG-1 and the MPEG-2 extension to lower sampling frequencies, as well as the de facto MPEG 2.5 format. All three audio layers - Layer I, Layer II, and Layer III (i.e. MP3) are fully implemented.

MAD has the following special features:

- 24-bit PCM output
- 100% fixed-point (integer) computation
- completely new implementation based on the ISO/IEC standards
- available under the terms of the GNU General Public License (GPL)

The original MAD library has a lot of implementation details relating to title & timing information(ID3 Decoding) for the MP3 song. Also it contains implementation details for different layers viz. layer 1, layer 2, layer3. In our embedded version(highly stripped down so as to run it on the primitive Microblaze processor) all these implementation details relating to layer 1 & 2 have been removed. Also the ID3 decoding information present in the MP3 file are all neglected. Also the all the sine and cosine and mathematically computationally intensive have been ported to look-up tables. We also modified the MAD player code to work on the big-endian architecture of the Microblaze processor. Many of the computationally intensive parts of the code were also removed from the software and implemented as a hardware peripheral on the Spartan 3E FPGA. In spite of the fact our code was a highly optimized version to run on an embedded platform the code was extremely large and could not be booted of the on-chip BRAM's. So we in our implementation here we run our source off the on-board 64Mbytes SDRAM.

The other things that were also incorporated as peripherals were the SPI bus interface to interface the on-board 12-bit DAC to the microblaze processor. As an alternative to the inferior quality of sound that the 12-bit DAC produces since MAD produces 24-bit PCM which results in 50% of the PCM information being lost while playing of the 12-bit DAC, we implemented a digital PWM signal

based on the PCM information that was used to drive the speakers. But, to our dismay even this latter implementation did not improve the sound quality drastically because of noise amplification by the amplifiers of the speaker. A possible solution to this problem could be another band-pass filter to remove the high frequency noise.

As a third option we are only decoding the MP3 file on the Spartan board and then the raw PCM data is transferred to a PC via the serial interface. This raw PCM data can then be played on the PC rather easily. The sound quality in this option was satisfactorily good because of the superior noise filters in the drivers.

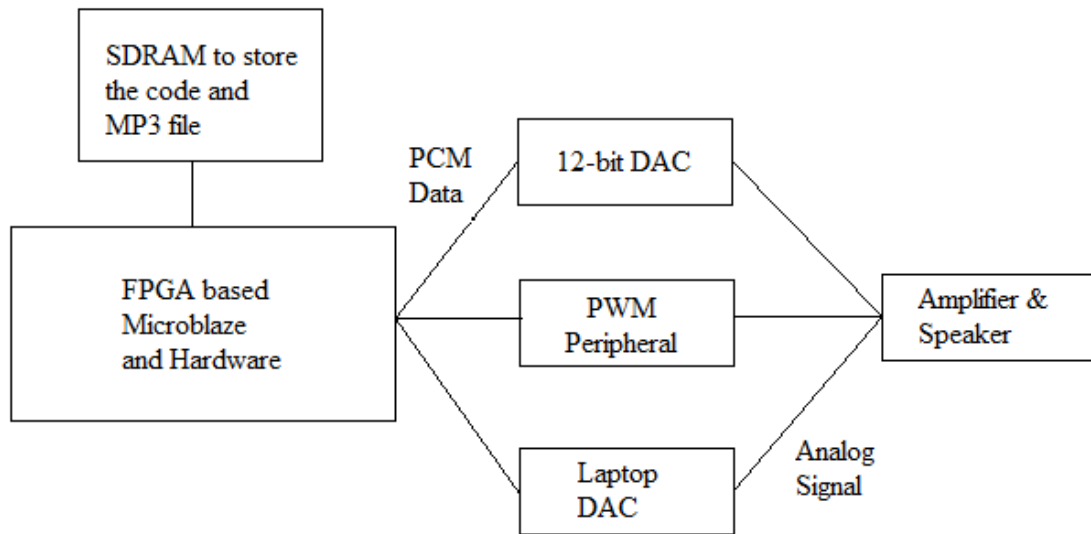


Fig 2. Block Level Diagram of the MP3 Decoder Implementation on Spartan 3E Kit

4. Conclusions

Although the Spartan 3e Starter Kit was able to play music after the entire decoding process the sound produced was not very clear due to the low resolution of the DAC(12-bits). To produce high quality stereo sound a 16-bit DAC would be much more preferable. The PWM option also did not produce the desired clarity in sound due to the same problem. However, when the decoded PCM data was ported back to a laptop and the analog to digital conversion was done using the hardware available on any standard laptop the the output had the desired quality as the DAC's used were of higher resolution.

5. References

- [1] FPGA based Architecture of MP3 Decoding Core for Multimedia Systems, Thuong Le-Tien, Vu Cao-Tuan, Chien Hoang-Dinh
- [2] A Full Hardware Implementation for an MP3 Decoder Chip using VHDL, Ahmed H. Abdel-Gawad, Saad F. Abdel-Aziz, et al.
- Implementation of an 18-point IMDCT on FPGA, Huibert J. Lincklaen.
- [3] <http://www.underbit.com/products/mad/>
- [4] <http://www.xilinx.com/support/documentation/index.htm>
- [5] <http://en.wikipedia.org/>