pLayer-i
An internet based muzik player

[CSEE W4840 Design – March 2008]

Maninder Singh ms3770@columbia.edu

Nishant R. Shah nrs2127@columbia.edu

Ramachandran Shankar rs2857@columbia.edu

1. Abstract

The need for dedicated hardware for multimedia decoding is rising with the increase in number of handheld devices with embedded device constraints on it, like power, area and precision.

2. Introduction

To Process the multimedia data (images, audio etc...) and distribute over a network their compressed versions are used. The simple reasoning behind this approach is to raise the bandwidth capacity to process task in real time and allow the content of signals to be suitable for the band-width of processing systems. Software is the most common tool used to decompress and use the data. Several SOC solutions have been developed but they are built around a RISC processor with a suitable ISA. But the demand of handheld players and multimedia in mobile phones has raised a need for a dedicated hardware to decode these file-formats with low power consumption and faster acceleration. For the Embedded System Design class we would to propose to make an internet music player.

3. MP3 Standard

MPEG-1 Audio Layer 3, more commonly referred to as **MP3**, is a digital audio encoding format using a form of lossy data compression. It is a common audio format for consumer audio storage, as well as a de facto standard encoding for the transfer and playback of music on digital audio players. MP3 is an audio-specific format that was designed by the Moving Picture Experts Group. The MP3 standard describes a sound format with one or two sound channels sampled at 32 kHz, 44.1 kHz or 48 kHz, encoded at 32 kbit/s up to 320 kbit/s. In this format, a piece of music can be compressed down to approximately 1 Mb/minute and still sound virtually indistinguishable from the 10 Mb/minute original.

An MP3 file is made up of multiple MP3 frames, which consist of a header and a data block. This sequence of frames is called an elementary stream. Frames are not independent items ("byte reservoir") and therefore cannot be extracted on arbitrary frame boundaries. The MP3 Data blocks contain the (compressed) audio information in terms of frequencies and amplitudes. The diagram shows that the MP3 Header consists of a sync word, which is used to identify the beginning of a valid frame. This is followed by a bit indicating that this is the MPEG standard and two bits that indicate that layer 3 is used; hence MPEG-1 Audio Layer 3 or MP3. After this, the values will differ, depending on the MP3 file. *ISO/IEC 11172-3* defines the range of values for each section of the header along with the specification of the header.

The PCM input is divided into chunks of 576 samples called granules. For two-channel inputs, a sample represents two values. In this case, each granule will contain information about two channels, and the following steps will be repeated for the second channel. The samples are fed through a polyphase filter bank that splits the 576 samples into 32 subbands with 18 samples in each subband. A granule maybe initially silent, but may contain a sharp attack (a sudden loud sound) and so the masking thresholds might be improper for the silent part of the granule. This results in a brief burst of potentially audible noise.

4. Design

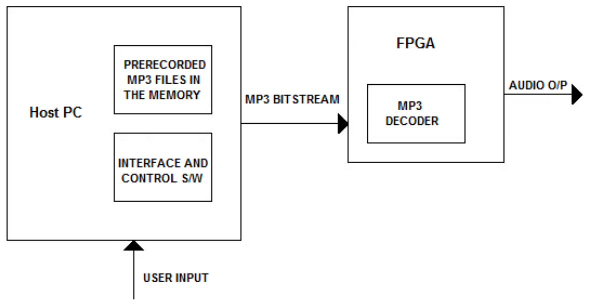

Host PC
PRERECORDED MP3 FILES IN THE MEMORY
INTERFACE AND CONTROL S/W
MP3 BITSTREAM
FPGA
MP3 DECODER
AUDIO O/P
USER INPUT

Fig 1: Block diagram of the system level design MP3 decoder

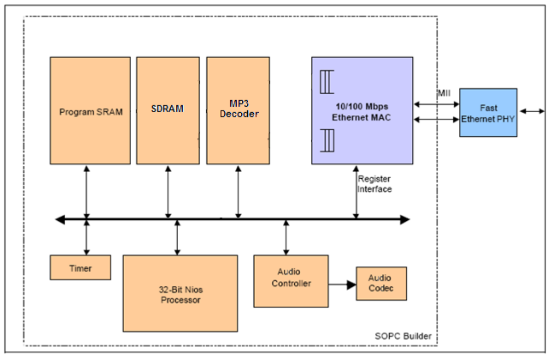
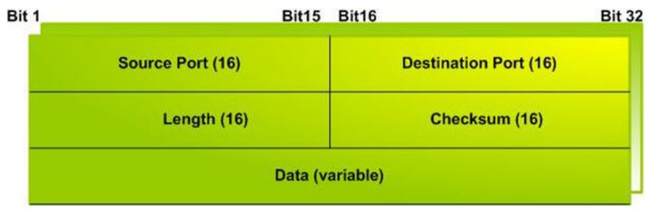


Fig 2 : The block diagram of the Hardware on the FPGA

4.1 UDP

4.1.1 UDP Packet Structure

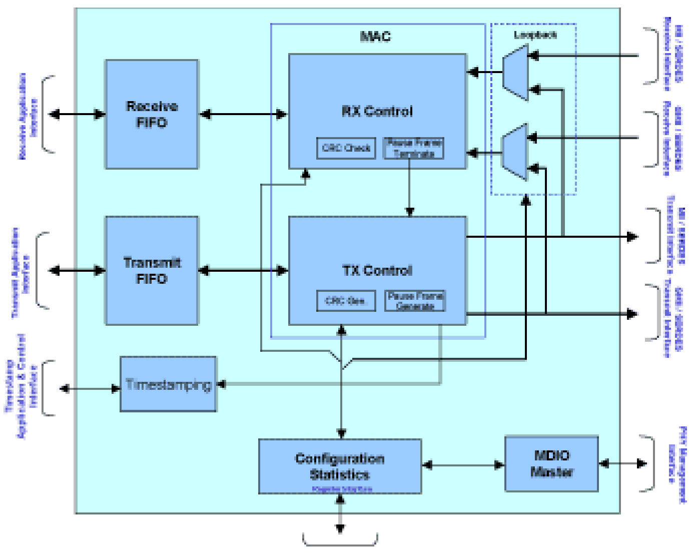
UDP is a minimal message-oriented Transport Layer protocol. In the Internet Protocol Suite, UDP provides a very simple interface between the Internet Layer below (e.g., IPv4) and the Application Layer above. UDP provides no guarantees to the upper layer protocol for message delivery and a UDP sender retains no state on UDP messages once sent.



The MP3 bitstream is broken down and shoved into the Data field of each UDP packet and sent from the host PC to the FPGA using Ethernet LAN cable.

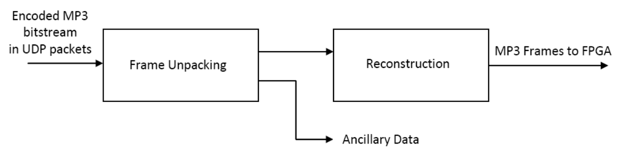
4.1.2 Ethernet MAC:

The Encoded MP3 bitstream that are wrapped in UDP packets will be input to the decoder, where it will first be unpacked. For the FPGA to receive the incoming packets from the Ethernet cable [Fast Ethernet PHY] the Ethernet MAC block is used to interface the Ethernet cable with the FPGA. The block diagram of the Ethernet MAC is shown below.


MAC
Loopback
Receive FIFO
RX Control
CRC Check
Pause Frame Terminate
Transmit FIFO
TX Control
CRC Gen
Pause Frame Generate
Timestamping
Configuration Statistics
MDIO Master
Receive Application Interface
Transmit Application Interface
Timestamp Application & Control Interface
MII / SERDES Receive Interface
GMII / SERDES Receive Interface
MII / SERDES Transmit Interface
GMII / SERDES Transmit Interface
PHY Management Interface

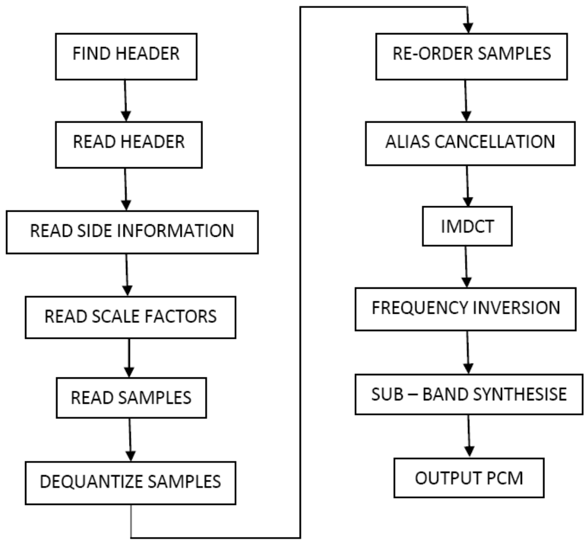
4.1.3 UDP Receiver

The output from the Ethernet MAC is fed to UDP decoder that we plan to implement in software. The UDP decoder unwraps the MP3 frames from the payload of the UDP packets. Error checking will also occur in the extraction phase. Any ancillary data will most likely be discarded at this point in the system and the output from the UDP decoder are MP3 frames, which are fed as input to the MP3 decoder block of the FPGA.



4.2 MP3 DECODER

The decoder basically applies the inverse transformations on the incoming MP3 frames to restore the PCM audio stream for playback. The flowchart of the MP3 decoder is shown below:



Read Header
The first task is to locate the Synchronization Word that marks the beginning of valid MPEG Audio frame. The Synchronization Word is part of the header that contains information about the layer number, sample rate and channel configuration. These do not change for the duration of the entire bit stream.

Read Information
The information that is required by the decoder to decode the MP3 data is known as side information and each channel is allocated a side information in each granule. This information contains various decoding parameters used to decode and dequantize the MP3 file.

Read Scale Factors
The frequency spectrum is divided into scale factor bands. These bands are determined by the sample rate and correspond to the critical frequency bands of the human ear. For each scale factor band, there is a scale factor that is used to control gain during sample dequantization.

Huffman Decompressor
The 576 Huffman coded sample values are now read and decoded using Huffman tables indicated by the side information. The encoder may use several different Huffman tables on different sample regions.

Sample Dequantization
Here the samples from the bitstream are dequantized and scaled to the proper values using the scale factors and granule gain value. Sample values are raised to the power of the 4/3 during Dequantization process.

Reorder Sampling
Samples in the blocks that use the short time window setting must be re – ordered to be processed by the following steps.

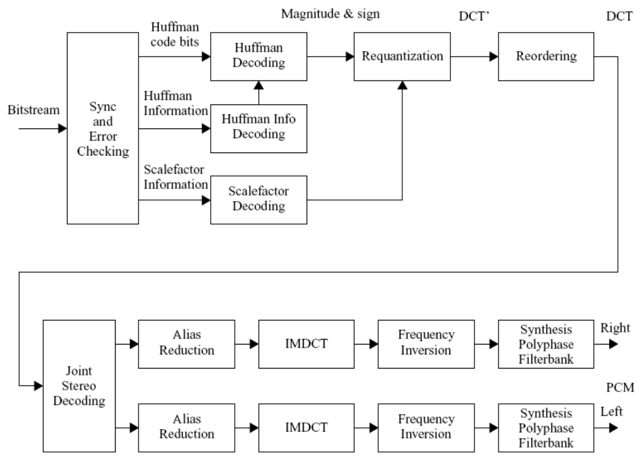
Alias Cancellation
The decoder applies alias cancellation to blocks which use long time window setting to compensate for the frequency overlap of the sub band filter bank.

IMDCT
Each sub band is transformed back to time domain. For long blocks the 36 point IMDCT calculates the 36 output samples directly. For the short samples, the outputs from the 12 point IMDCT are combined into 36 output samples. The first 18 samples are added to the stored overlap values from the granule. These values are the new output values. The last 18 output are stored for overlap with the next granule.

Frequency Inversion
Every second sample in every second sub band is multiplied by -1 for frequency inversion of the sub band filter bank.

Sub-band Synthesize
The 32 sub bands are combined into time domain samples that cover the whole frequency spectrum. One sample is taken from each sub band and transformed using DCT. The result is written to low end of a large array after room has been made by shifting its previous contents towards higher indices to obtain the PCM output

4.2.1 IMPLEMENTATION OF MP3 DECODER



MP3 FRAME FORMAT

An MP3 frame consists of 1152 frequency domain samples, which is divided into two granules of 576 samples each. Each granule is further divided into 32 sub band blocks of 18 frequency lines. The MP3 frame consists of 4 parts: Header, Side Information, Main Data and Ancillary data.

Header	Side info	Main data	Ancillary data

Huffman Decoding

Huffman Decoding can be implemented by two approaches: Binary Tree Search and Direct Table Lookup. The Huffman decoder tables can be translated into binary trees. Each tree then represents a certain table. The trees are traversed according to the bits in the bitstream; where a ‘0’ might mean go ‘left’ and a ‘1’ go ‘right’. An entire code-word is fully decoded when a leaf is encountered. The leaves contain the values for the spectral lines. For the direct table lookup method the decoder uses large tables. The length of each table is 2^b, where b is the maximum number of bits in the longest code-word for that table. To decode a code-word, the decoder reads ‘b’ bits. The bits are used as a direct index into the table, where each entry contains the spectral line values and information about the real length of the code-word. The surplus bits must then be re-used for the next code-word.

We propose to combine the two approaches by using the Cluster Decoder Method and implement it in the software. A fixed number of bits is read from the bitstream and used as a lookup index into a table. Each table element contains a hit/miss bit that indicates whether the code-word has been fully decoded yet. If a hit is detected the symbol is read from the table element as well as the number of bits that is used for the code-word. If it is a miss the decoding continues by using the information from the table element to determine how many more bits to read from the bitstream for the next index, as well as the starting address of the next table to use.

Experiments have shown that individual tables should be at most 16 elements long when decoding MP3. It is further shown that the processing requirements are approximately 1 MIPS and the memory requirements are 56 kbits for the lookup tables.

Requantization

The requantization step must be performed once for each sample in the bitstream. The profiling done on MP3 Decoder [FPA] has shown that more than 50% of the execution time is spent in the windowing. Hence to optimize this step, we plan to implement this using the Newton’s method on the FPGA.

The $y = x^{4/3}$ function can be rewritten as $y^3 - x^4 = 0$. This form is suitable for Newton’s method of root-finding which will yield a value of y that approximates $x^{4/3}$. The function result is calculated through repeated iterations that successively reduces the residual error $| y - x^{4/3} |$:

$$y_{n+1} = y_n - \frac{y_n^3 - x^4}{3y_n^2} = \frac{2y_n^3 + x^4}{3y_n^2}$$

Iteration formula for $y=x^{4/3}$

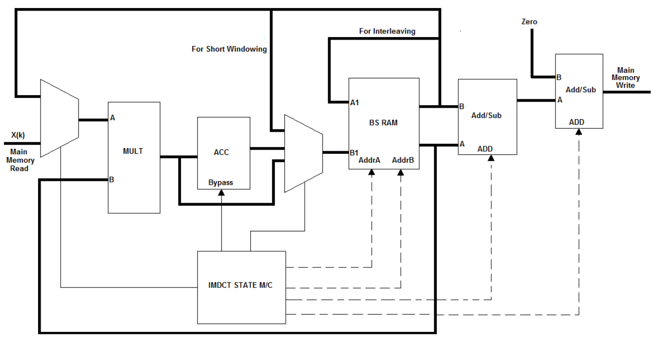
The formula is rewritten as the second form to avoid floating-point cancellation. The starting value $y0$ for the iteration formula affects the number for iterations needed to achieve the desired accuracy. For this application accuracy larger than 16 bits is sufficient. A good starting value for y_0 is calculated by the polynomial fit function $y_0=a_0+a_1*x+a_2*x^2$. This function is designed to resemble $y = x^{4/3}$ as closely as possible for $0<x<8207$. The starting value will yield the desired accuracy in 3 iterations.

Reordering and Alias Reduction

Since the reordering and alias reduction are not time critical parts of the decoder, we plan to implement this block in software.

IMDCT

The block diagram of the Inverse Modified Discrete Cosine Transform is shown below:



The equation below gives the Inverse Modified Discrete Cosine Transform. This process is a calculation intensive operation. Hence we plan to implement it in the Hardware where all the cosine values used for IMDCT computations and signed values for windowing will be stored in the Block RAM. Approximately 16Kbits are required to store the lookup table.

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

IMDCT Equation

The following table shows the number of clock cycles needed to compute all time samples in a frame. The time
required, when using a clock frequency of 24 MHz, is shown too.

IMDCT type	Clock Cycles	time/μs
Short Windowing	15,615	651
"Normal" Windowing	15,196	633

Polyphase Filterbank

The polyphase filterbank converts the time domain samples from the IMDCT to PCM samples. The steps involved are:

1. Matrixing of 32 sub band samples to produce 64 V Vectors.
2. Windowing of selected samples from the V vector FIFO with a constant window function D to produce W vector
3. Summing the W vector with itself to produce 32 output PCM samples.

A 32 point Fast DCT implementation has been proposed to substantially improve on the step 1 of polyphase filterbank. To keep the filterbank design efficient and simple, DCT part can be implemented straight forward, using matrix multiplications instead of a butterfly scheme. But due to lack of proper data we are not sure whether to implement the filterbank using this method.

4. References

[1] FPGA based Architecture of MP3 Decoding Core for Multimedia Systems, Thuong Le- Tien, Vu Cao-Tuan, Chien Hoang-Dinh

[2] A hardware implementation of an MP3 decoder , Irina F¨altman, Marcus Hast, Andreas Lundgren, Suleyman Malki, Erik Montnemery, Anders Rangevall, Johannes Sandvall, Milan Stamenkovic

[3] A hardware MP3 decoder with low precision floating point intermediate storage , Andreas Ehliar, Johan Eilert