MAC_MP3: A Low Energy Implementation of an Audio Decoder

Ho Fai Ko and Nicola Nicolici, McMaster University, Ontario, Canada

1 Introduction

Among the various audio encoding formats, the MPEG-1 layer 3 (MP3) format is the most widely used, and well defined as a standard by the International Standard Organization [1]. However, most of the digital music players implement the MP3 algorithm in software or in embedded systems for real-time audio decoding during music playback. Although mobile devices nowadays already have enough computational power to decode MP3 files using a general microprocessor, existing implementations are not necessarily power-efficient. As battery-life is crucial for mobile applications, we propose a hardware architecture that enables the implementation of an MP3 decoder with low power consumption. We have tested our design using the Altera DE2 board [2] and the Xilinx multimedia board [3]. This shows the adaptability of the proposed architecture to different hardware platforms.

2 MP3 Algorithm

MP3 data is broken into frames to allow streaming of data over a communication channel. One frame of data is divided into two granules, and each granule contains 576 audio samples per channel. As a result, when a sampling rate of 48 kHz is used, one will have a real-time constraint of 576/48kHz or 12 ms for decoding data for one granule.

The algorithm for decoding MP3 files can be divided into multiple stages as shown in Figure 1. Note that readers should refer to [1] for complete descriptions on each stage of the algorithm. The Huffman decoding stage performs lossless decompression using the predefined Huffman tables from [1] to extract the main data and the side information such as scale factors and block types (long or short) used to select the appropriate coefficients in various stages of the algorithm. The requantization step performs non-uniform quantization to the main data using a power law derived from the scale factors contained in the side information. When short blocks are encountered, the reordering step sorts the requantized samples according to subbands and frequency. After that, the stereo decoding stage reconstructs the samples for the left and right channels when the stereo mode is selected. At this stage, the 576 samples are ordered in 32 subbands, with 18 samples per subband for each channel. Anti-aliasing then merges the frequency lines in each subband using a set of coefficients defined in [1]. When long block type is detected, the inverse modified discrete cosine transform (IMDCT) transforms the 18 inputs in a subband to 36 outputs, which are then multiplied with a 36-point window. However, for the short block type, windowing is performed differently to generate the 36 output samples. The first half of the block of 36 values is then overlapped with the second half of the block from the previous granule, producing 18 values in each subband for frequency inversion, which negates all the odd samples in all odd subbands. The 576 samples at the output of frequency inversion are then applied to the synthesis filter banks, which employs another set of MDCT and windowing coefficients to transform the audio samples into the time domain for audio playback.

3 MP3 Architecture

We implemented the MP3 decoder using 16-bit fixed-point arithmetic with the proposed architecture shown in Figure 1. We group the different stages of the decoding algorithm into two units: the Huffman unit and the Mac unit.

3.1 The Huffman Unit

The Huffman unit performs Huffman decoding, requantization, reordering and stereo decoding. It occupies two 1024×16 block RAMs to store the Huffman tables. For requantization, reordering and stereo decoding, two 1024×16 block RAMs are used to store the coefficients and intermediate data. To reduce resource usage, part of the RAM for buffering the intermediate data is also shared with Huffman decoding as bit reservoir, and there is only one 16-bit multiplier allocated for the requantization step. Each time a start signal is applied to the Huffman unit, it decodes data for one granule in two channels, generating 1152 samples and stores them in the dual-port RAMs.

3.2 The Mac Unit

The Mac unit consists of anti-aliasing, IMDCT, frequency inversion and synthesis polyphase filter banks. The coefficients for performing anti-aliasing, IMDCT and filter bank synthesis are stored in three 1024×16 block RAMs. Note that each component in the Mac unit is duplicated with its own memories to decode data for two channels simultaneously. However, since the two channels basically perform the same operations, the coefficient memories are shared to reduce resource usage. To further reduce area, one 16-bit multiplier is shared across all stages in one channel of the Mac unit. In our implementation, the anti-aliasing and IMDCT operations are combined such that in each iteration, a block of 18 input samples will be transformed to a block of 36 samples in a subband. In this case, by iteratively applying the same operations on the 32 subbands, control logic can be simplified and hardware can be better utilized. After IMDCT is finished, the 32 blocks of 36 samples are stored in two 1024×16 block RAMs, where frequency inversion is applied to prepare the samples for the synthesis polyphase filter bank stage. To store the intermediate samples during MDCT and windowing, one 1024×16 block RAM is allocated for this stage. In total, the Huffman and Mac units use only three multipliers and 13 1024×16 block RAMs when decoding one granule of MP3 data for two channels. This reduction in resource usage effectively achieves power reduction.

3.3 Timing Constraint for MP3

In our implementation, the Huffman unit takes 27 clock cycles to decode 2 samples. Since there are 576 samples per channel and two channels per granule, it takes a total of 15552 clock cycles to decode data for one granule of 1152 audio samples. On the other hand, the Mac unit takes 14769 clock cycles for the anti-aliasing and IMDCT operations on the 32 subbands of data, 720 clock cycles for frequency inversion, and 28314 clock cycles for the synthesis polyphase filter bank step. This gives a total latency of 43803 clock cycles for the Mac unit to decode 576 samples in each channel. With a real-time constraint of 12 ms when decoding one granule of audio samples using a sampling rate of 48

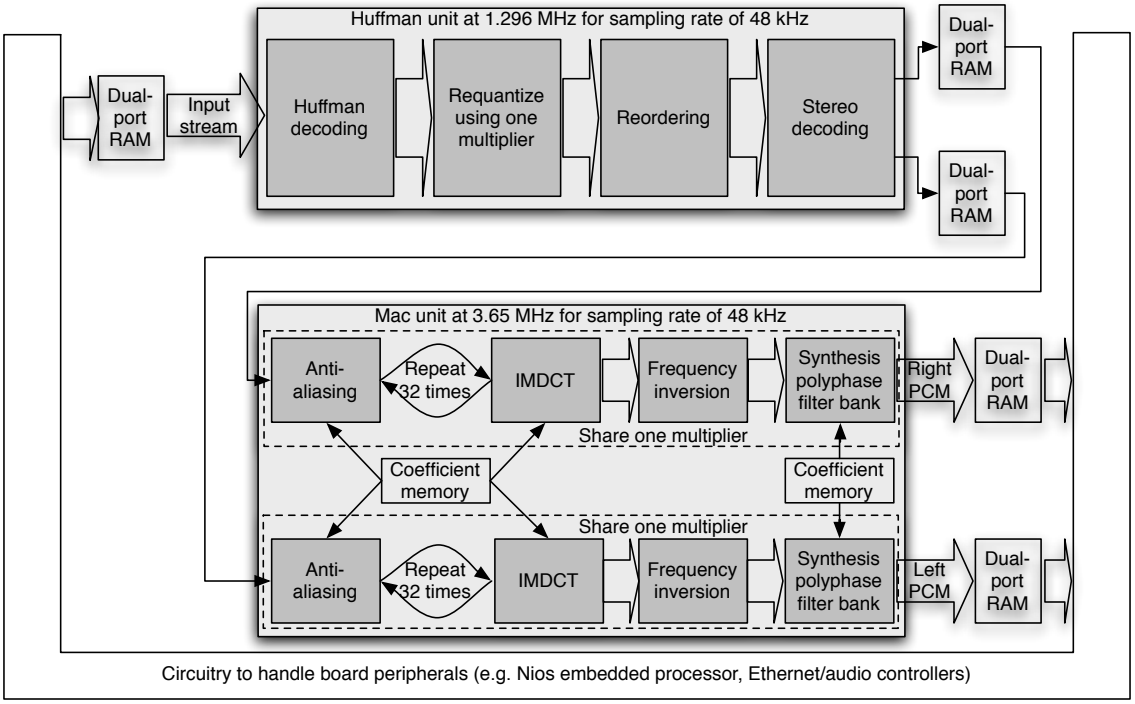

Figure 1. The MAC_MP3 architecture

kHz as discussed in Section 2, the lowest clock frequency that should be applied to the Huffman and Mac units are 1.296 MHz and 3.65 MHz respectively. In the case when lower sampling rate is required, the proposed architecture can be clocked at reduced frequencies to further decrease power consumption.

Note that the physical clock with the lowest frequency is 27 MHz on the Xilinx multimedia board and the Altera DE2 board. Using the available digital clock managers (DCMs) on the Xilinx multimedia board, we chose to operate the Huffman unit at about 1.7 MHz (27 MHz / 16), and the Mac unit at 3.8 MHz (27 MHz / 7). On the other hand, for simplicity, a counter is used as a clock division circuitry on the Altera DE2 board to generate clock frequencies of 1.7 MHz (27 MHz / 16) and 6.75 MHz (27 MHz / 4) for the Huffman and Mac unit respectively. To synchronize the two units that work at different clock frequencies, dual-port RAMs are inserted between the units for buffering the intermediate data. This also helps keep the two units busy by allowing the Mac unit to decode data for granule $i-1$, while the Huffman unit is extracting data for granule i.

3.4 Considerations for Peripherals

In order to isolate the design from the peripherals, dual-port RAMs are used to buffer the input MP3 data stream and the decoded pulse code modulation (PCM) samples at the input and output of the MP3 decoder respectively. To show the portability of the proposed architecture to various hardware platform, we implemented the design on the Altera DE2 board [2] and the Xilinx multimedia board [3]. The Altera DE2 board utilizes the Nios embedded processor to read data from an SD card to fill the input dual-port RAM, as well as extracting the PCM samples from the output dual-port RAM for the I^2C audio controller. On the other hand, the Xilinx implementation employs the Ethernet port for supplying MP3 data, and the on-board AC97 audio codec for audio playback.

4 Results

The proposed MP3 decoder takes only 555 flip-flops, 3828 logic elements, 36 KB of memory and six 9×9 multipliers on the Altera DE2 board without any peripherals. When clocking our small design at low frequency, low power consumption can be achieved when decoding MP3 files in real-time.

When the Nios embedded processor and peripherals are included, the design contains 3590 flip-flops, 9812 logic elements, 44.27 KB of memory and ten 9×9 multipliers. Please note that in this case, the Nios processor is about six times the size of our design. For the Xilinx implementation with peripherals, it utilizes 1272 flip-flops, 4920 4-input look-up tables, 36 KB of memory, three 18×18 multipliers, two DCMs for the MP3, one DCM for the Ethernet controller, and one DCM for the AC97 codec. The increase in flip-flops and look-up tables are due to the size of the Ethernet and the AC97 controllers for providing MP3 data and extracting audio samples accordingly from our design.

5 Acknowledgment

The authors want to thank David Leung and Ehab Anis for their work on the Huffman unit. We also gratefully acknowledge the Canadian Microelectronics Corporation (CMC) for the provision of FPGA prototyping platforms and CAD design software.

References

[1] ISO/IEC 11172-3, Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s - Part 3: Audio, 1993.

[2] Altera Corporation. Altera Development and Education Board. http://www.altera.com/education/univ/materials/boards/unv-de2-board.html, March 2007.

[3] Xilinx Incorporated. Xilinx Multimedia Board. http://www.xilinx.com/products/boards/multimedia/, March 2007.