

SOFTWARE OPTIMIZATION OF THE MPEG-AUDIO DECODER USING A 32-BIT MCU RISC PROCESSOR

Keun-Sup Lee, Young Cheol Park¹ and Dae Hee Youn

MCSP Lab., Dept. of Electrical and Electronic Eng., Yonsei University, Seoul, Korea

¹Dept. of Division of Information Technology, Yonsei University, Wonju, Korea

ABSTRACT

This paper proposes optimization techniques to implement MPEG audio decoding algorithms in real-time using general-purpose 32-bit MCU RISC processor. Both MP3 and AAC LC profile decoding algorithms can be partitioned into two parts: control-intensive part and computation-intensive part. Optimization techniques in this study are focused on developing methodologies that are suitable for each part. We implemented MP3/AAC decoder using ARM-based RISC MPU and its test board. The both designed decoders with proposed optimizations could achieve decoding processes in real-time within an operating frequency of 35 MHz. And ISO 13818-4 compliance test results confirmed that the proposed decoders ensured full compliance with ISO 13818-3 audio decoder. These implementation results illustrate the efficiency of the proposed design methodology in both performance and cost.

1. INTRODUCTION

Recently, portable MPEG audio devices has been actively developed and already become as popular as portable Compact Disc (CD) or Mini-Disc (MD) devices in the consumer market. Since digital audio players do not require mechanical components needed for other devices such as CD, decoding processors consume a large portion of the battery power in use. Therefore, the power consumption of the decoding processors affects the battery life of the player.

Many portable players adopt DSP because of decoding complexity. But high power consumption of the DSP architecture brings short battery life. Moreover, complete digital audio decoding system may need an additional micro-controller for the bitstream parsing and overall system control. Therefore low power architecture suitable for portable devices is required.

As processor design technology develops, the performance of micro-controller unit (MCU) reaches old DSP level. If MPEG audio decoding algorithm can be implemented using MCU without DSP, it is possible to integrate system control functionalities with audio decoding on a single MCU processor. It would be beneficial to the overall power consumption of portable MPEG player. However, since MCU has architectural disadvantages such as simple and limited hardware resource, MCU-based implementation requires a great deal of optimization compared with DSP-based case. And different optimization techniques for DSP-based case are required.

This paper proposes optimization techniques to implement MPEG audio decoding algorithms in real-time using general-purpose 32-bit MCU RISC processor. Decoding algorithms implemented in the paper are MPEG-1/2 Layer-III (MP3) and MPEG-2 AAC (Advanced Audio Coding) Low Complexity profile[1][2]. We adopted ARM7TDMI architecture as the target 32-bit MCU RISC processor. MP3/AAC decoding software was written in C codes and compiled with the ARM C-cross compiler. But the compiled codes have much redundancy. According to implementation result, the C-compiled MP3 code with no optimization required about 60~70 MIPS to decode a 128 kbps 44.1 kHz stereo bitstream. Therefore, we performed further optimization for these codes.

The organization of this paper is as follows. In Section 2, the design methodology of MPEG-audio decoders is presented. Software optimization processes and techniques based on this proposed methodology are described in Section 3. In Section 4, we will report implementation results containing complexity and output audio accuracy. Finally, we summarize our research in Section 5.

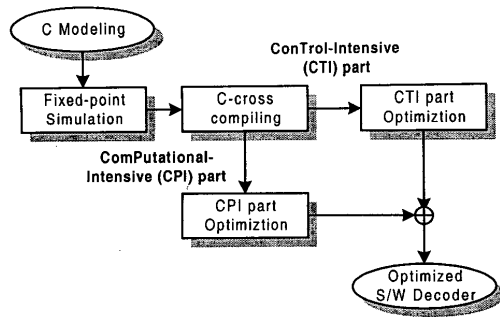


Figure 1. design methodology

2. DESIGN METHODOLOGY

The overall flow of the design methodology is illustrated in Figure 1. The flow describes implementation steps from C modeling to the optimized software decoder. In the fixed-point simulation stage, quantization errors caused by fixed-point operations are to be minimized. After the fixed-point simulation, C codes are converted into Assembly coded using a C-cross compiling tool. Since the resultant Assembly codes can contain much redundancy due to the inefficiency of the C-compiler, further optimization is required. As shown in Figure 1, there exist two distinct paths regarding algorithmic properties.

Figure 2 shows MP3 and AAC decoding flows. Both MP3 and AAC decoding algorithms consist of bit parsing, bit allocation decoding and unique synthesis filtering of their own. As shown in Figure 2, each decoding algorithm can be partitioned into two parts, which are control-intensive part and computation-intensive part. Control-intensive part consists of various operations related to control program flow. They have relatively low computational loads, but are difficult to control. On the contrary, Computation-intensive part has regular operations of high computational loads. Considering these algorithmic properties, distinct optimization technique must be performed for each part of the algorithmic. The optimized software decoder is obtained from these two distinct optimizations.

3. DESIGN OPTIMIZATION

Before the main optimization, we performed C-code level optimization to improve the performance of the code. We modified the code to be more suitable for the ARM processor and to help the compiler generate better code.

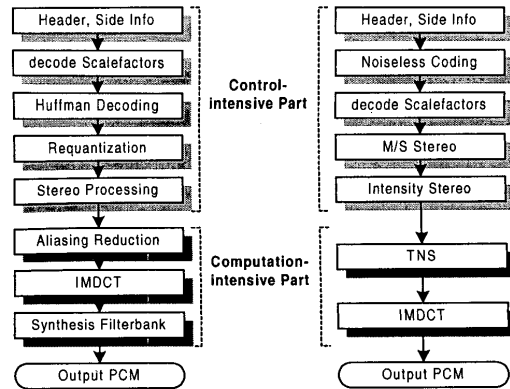


Figure 2. Partitioning MP3 and AAC decoding algorithm

3.1. C-compiler based Implementation

In general, assembly code from C-compiler has much redundancy. So, in order to get more efficient assembly code, hand assembly coding technique is required. But hand assembly coding requires so much time, and therefore, development cost increases.

Although C-compiler produces much redundancy, the redundancy from compiler varies according to processor architecture. Figure 3 shows the result of performance comparison between hand-optimized code and C-cross compiled assembly code of ARM7 Core. Target C code is Huffman decoding core routine, which belongs to control-intensive part.

According to Figure 3, two assembly codes are the same and the efficiency of C-cross compiler is very high. It is because of simple architecture and minimum resources of MCU processor that these results are obtained. Few instructions and resources due to simple architecture reduce the flexibility of assembly coding. So the optimizer of MCU compiler has more efficiency than that of DSP compiler with versatility.

Therefore, in this paper we compiled C code using C-cross compiler firstly, and then performed further optimization.

Original C-code	<pre> while((x = *val++) < 0){ if(Mask >= 0) val -= x; Mask <=<= 1; num++; } </pre>
Hand Assembly Code	<pre> B ReadX Loop: CMP Rd, #0 SUBGE pHuff, pHuff, Re, LSL #2 MOV Rd, Rd, LSL #1 ADD Rb, Rb, #1 ReadX: LDR Re, [pHuff], #4 CMP Re, #0 BLT Loop </pre>
Cross Compiled Code	<pre> 0xf894:b 0xf8a8 0xf898:cmp r4, #0 0xf89c:subgr r12, r12, r0, lsl r3 0xf8a0:mov r4, r4, lsl #1 0xf8a4:add r14, r14, #1 0xf8a8:ldr r0, [r12], #4 0xf8ac:cmp r0, #0 0xf8b0:blt 0xf898 </pre>

Figure 3. Comparison between two assembly coding methods

3.2. Control-intensive Part Optimization

The control-intensive part of MPEG audio algorithm accompanies instructions related to the program flow control such as loop, conditional branch, and so on. Of all control-intensive routines Huffman decoding and scalefactor unpacking routines are very complex routines.

Huffman decoding consists mainly of bit unpacking and Huffman table searching routine, and these routines are iterated. Since bit unpacking and table search routines refer to bitstream buffer and ROM tables respectively, Huffman decoding routine requires to access memory frequently. These frequent memory accesses degrade the performance of overall system based on MCU, which has a weak point in memory access cycle. To minimize the number of bitstream buffer access, we use in-line hand-assembly to read maximum bits of Huffman codebook at once from bitstream buffer. This implementation reduces unpacking function call and bitstream buffer access overheads in iteration loop.

It is bit unpack routine that extracts Huffman code and scalefactor from bitstream buffer. Since bitstream buffer is finite size, to extract information from it circular addressing mode is required. But most of MCU's don't

have complex addressing modes such as circular and bit-reverse addressing mode. So, to implement circular addressing with MCU additional computational loads due to if-clause are required. Whenever bit unpack routine is called, it must be checked whether bitstream read point reaches the end of bitstream buffer.

In order to remove this additional routine, we modified bitstream buffer control method as shown in Figure 4. By shifting unused bitstream to the first position at the start of each frame decoding, read point can be prevented from reaching the end of buffer. Considering the number of bit unpacking routine call during decoding process, complexity for shifting unused bitstream is very low. Furthermore, since bit unpacking routine called very often during control-intensive routines, the loads of subroutine call are somewhat heavy. So, it is advantageous to apply inline function to bit unpacking routine.

3.3. Computation-intensive Part Optimization

In MPEG audio decoding algorithm, frequency to time mapping synthesis filterbank routines come under computation-intensive part. For heavy computational loads of these routines fast algorithm commonly applies to them.

From the algorithmic point of view, most of operations are memory access, addition, and multiplication rather than branch. For example, as shown in Equation 1 one MAC operation needs one addition, one multiplication and three memory accesses that are two loads and one store.

$$y(i) = \sum_n x(n) \times c(n, i) \quad (1)$$

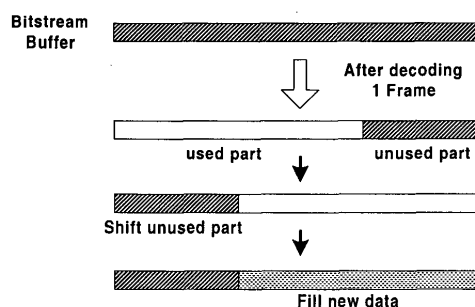


Figure 4. Modified bitstream buffer control method

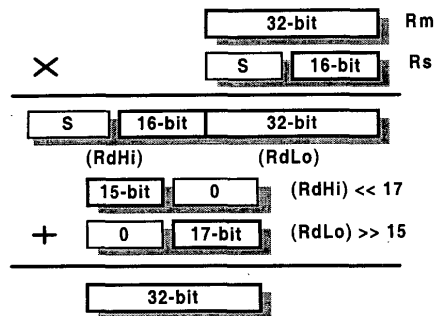


Figure 5. 32-bit x 16-bit fixed-point multiplication

Considering that MAC is the major operation of MPEG audio synthesis filterbank, computational loads for memory access are dominant. And due to low performance of multiplier in MCU multiplication also requires many clock cycles. Therefore we concentrated optimization on reducing the computational loads of memory access and multiplication.

Many MCU processors support multiple memory access operation. Multiple memory access makes it possible to load from and store to memory within less clock cycles than sequential memory access. For example, although ARM7 processor is capable of implementing a memory load in 3 cycles and store 2 cycles, by organizing data so that memory access is via consecutive addresses it is possible to achieve effective load and store operations in about single-cycle[3]. So with this multiple access the performance of synthesis filterbank can be significantly enhanced.

As mentioned above, computation-intensive routines consist of many multiplications. Since multiplication in MCU is a heavy burden, before the optimization for this we adopted fast algorithms that have few multiplications. Contrary to DSP, many MCU processors cannot operate full-length multiplication in 1 clock cycle due to simple multiplier. For that reason, many MCU processors support partial multiplication such as 32-bit x 8-bit. But, since multiplication with short data length can result in degrading data precision, it is important to determine the length of multiplicand and multiplier. Previous work showed that 32-bit x 16-bit multiplication is enough to obtain good audio output quality in MPEG audio

decoder[4]. Therefore, considering not only data precision but also complexity, we adopted 32-bit x 16-bit. 16-bit data is mainly used for coefficient table. Figure 5 shows fixed-point multiplication operation we used. These operations require 5 clock cycles. But, since MAC operation don't need to execute last two instructions for extracting high product before the accumulation is completed, the computational loads for multiplication is smaller than 5 clock cycle in many cases.

4. IMPLEMENTATION

4.1. Target RISC Processor Architecture

In order to determine target MCU RISC processor for implementing MPEG audio decoder, next two requirements must be considered; the first requirement is high operational performance of processor suitable for real-time decoding process, and the second one is high accuracy of decoder's PCM output.

For real-time implementation of audio decoder, the performance of MCU RISC processor must be able to reach at least 30 DSP MIPS[4]. Therefore, considering the architecture of general MCU, target processor is required to operate in very high clock frequency. And, since MPEG audio decoding algorithms consists of many MAC (Multiply-ACcumulation) operations, target processor must have a multiplier to reduce the computational loads.

The sound quality of decoder's PCM output is affected by the accuracy of decoding operation. Especially, since fixed-point ALU produces a lot of quantization noise, it is required to reduce these errors for the cost of increasing operational complexity. To minimize the both these efforts and additional complexity, 32-bit data path was chosen for fixed-point ALU architecture. Since long data path produces high resolution and large dynamic range, 32-bit data path has higher accuracy than 16-bit or 24-bit data path.

After considering above requirements, we adopted ARM7TDMI as target MCU RISC processor. ARM7TDMI satisfies above all requirements, and besides, now it is used for digital audio codec applications. The components of this processor are as follows[3]:

- 32-bit RISC Architecture
- 65 ~ 100 MHz clock frequency
- 31 x 32-bit registers and 6 status registers
- 32-bit x 8-bit multiplier and 32-bit ALU
- 32-bit Barrel shifter

4.2. Implementation Result

In order to measure the sound quality for the output PCM of designed decoders objectively, ISO 13818-4 compliance test was performed[5]. According to Table 1 both MP3 and AAC decoder don't exceed the criteria of compliance test, and therefore, both satisfy ISO 13818-4 full compliance.

Table 2 summarizes the computational complexity of designed software decoders. For MP3 bitstream popular music signal and for AAC bitstream white noise signal are used for performance test. All two bitstream have the specification of 2 channels, 128 kbps and sampling frequency of 44.1 kHz.

In order to verify these results, we implemented MP3/AAC decoder using Samsung S3C4510B01 ARM-based RISC MPU and its test board. According to this implementation results, the both designed decoders with proposed optimizations could achieve decoding processes in real-time within a maximum operating frequency of 35 MHz. According to software libraries fully optimized by ARM Ltd., the operating frequency of MP3 decoder is 29 MHz and that of AAC decoder is 22 MHz (without TNS) and 28 MHz (with TNS)[6]. Considering that the proposed methodology is the optimization in C-cross compiling level with low development cost, the performance of designed software decoder is comparable with that of ARM's software libraries. These implementation results illustrated the efficiency of the proposed optimization techniques in both performance and cost.

5. CONCLUSIONS

In this paper, we proposed optimization techniques for implementing MPEG audio decoding algorithms in real-time using a general-purpose 32-bit MCU RISC processor. Target MPEG audio algorithms were MPEG-2 Layer-III (MP3) and MPEG-2 AAC Low Complexity profile. Considering algorithmic properties, both MP3 and AAC decoding algorithms could be partitioned into two parts, which are control-intensive part and computation-intensive part. Therefore, we proposed optimization techniques suitable for each part.

Test result confirmed that the proposed decoding algorithm ensured full compliance with ISO 13818-3 audio decoder. According to the implementation results, the both designed decoders with proposed methodology could achieve decoding processes in real-time within an operating frequency of 35 MHz. These implementation results

Table 1. Results of compliance test

	RMS of Error	Maximum Error
Full Compliance	8.8097×10^{-6}	6.1035×10^{-5}
MP3	4.7483×10^{-6}	2.8729×10^{-5}
AAC	5.6012×10^{-6}	3.0518×10^{-5}

Table 2. Maximum complexity of designed decoder

MP3	Complexity	AAC	Complexity
Bitstream	17 %	Bitstream	18 %
Decoding	5 %	Decoding	6 %
IMDCT	16 %	TNS	22 %
Filterbank	62 %	IMDCT	54 %
Total	33.4 MHz	Total	32.6 MHz

illustrated the efficiency of the proposed optimization techniques in both performance and cost.

REFERENCES

- [1] ISO/IEC 13818-3 "Generic Coding of Moving Pictures and Associated Audio (Part 3: MPEG-Audio)", 2nd Edition, Feb., 1997
- [2] ISO/IEC JTC1/SC29/WG11 No.1650 "IS 13818-7 (MPEG-2 Advanced Audio Coding)", Apr., 1997.
- [3] ARM, *ARM7TDMI Data Sheet*, 1995
- [4] Keun-Sup Lee, Hyen-O Oh, Young-Cheol Park, and Dae Hee Youn, "High Quality MPEG-Audio Layer III Algorithm for a 16-BIT DSP", in Proc. IEEE ISCAS, vol. II, pp.205-208, May 6-9, 2001
- [5] ISO/IEC 13818-4 "Generic coding of moving pictures and associated audio information (Part 4: Compliance testing)"
- [6] ARM Ltd. Homepage (<http://www.arm.com>)

BIOGRAPHY



Keun-Sup Lee was born in Seoul, Korea, on 1975. He received his B.S. and M.S. degrees in Electronic Engineering from Yonsei University, Seoul, Korea in 1997 and 1999 respectively. He is currently a Ph.D. candidate in Department of Electrical and Electronic Engineering at Yonsei University, Seoul, Korea. His research interests include audio signal processing, DSP & ASIC implementation, high quality audio coding, HDL chip design, and algorithm optimization.

filter and its application, speech and audio signal processing, and real-time implementation of DSP algorithms.



Young Cheol Park received his B.S., M.S., and Ph.D. degrees in Electronic Engineering from Yonsei University, Seoul, Korea in 1986, 1988, and 1993 respectively. From 1993 to 1995, he was a Post Doctoral Research Scholar of the graduate program in acoustics in Pennsylvania State University, PA, USA. From 1996 to 1998, he was with Samsung Electronics Co., Ltd., Kiheung Korea, working on the development of digital hearing aids. In 1999 he joined InTime Corp. in Seoul, Korea. His research interests include digital audio, speech coding, and adaptive filters. He is currently an assistant professor in Yonsei University, Wonju, Korea.



Dae Hee Youn received his B.S. degree in Electronic Engineering from Yonsei University, Seoul, Korea, in 1977, and the M.S. and Ph.D. degrees in Electrical Engineering from Kansas State University, Manhattan, Kansas, in 1979 and 1982, respectively. From 1982 to 1985, he was an assistant professor at the University of Iowa, Iowa City, Iowa. Since 1985, he has been with the Department of Electrical and Electronic Engineering at Yonsei University, Seoul, Korea, where he is currently a professor. His research interests include adaptive digital