Software optimization of MPEG audio layer-III for a 32 bit RISC processor

Conference Paper · February 2002

DOI: 10.1109/APCCAS.2002.1114990 · Source: IEEE Xplore

3 authors, including:

Kisun You
Apple Inc.
17 PUBLICATIONS **177** CITATIONS
SEE PROFILE

Wonyong Sung
Seoul National University
180 PUBLICATIONS **2,671** CITATIONS
SEE PROFILE

SOFTWARE OPTIMIZATION OF MPEG AUDIO LAYER-III FOR A 32BIT RISC PROCESSOR

Wonchul Lee, Kisun You and Wonyong Sung

School of Electrical Engineering
Seoul National University
San 56-1, Shinlim-dong, Kwanak-gu, Seoul 151-742 Korea
Phone: +82-2-885-7411, Fax:+82-2-882-4656
Email: chul@dsp.snu.ac.kr, ksyou@dsp.snu.ac.kr, wysung@dsp.snu.ac.kr

ABSTRACT

We have implemented the MPEG 1/2 audio Layer-III decoding algorithm using ARM7 and ARM9 based systems. Optimizations are conducted in the algorithm, C-program, and assembly program levels. The assembly program level optimization that employs the block data transfer instructions intensively is very effective for reducing the number of memory accesses. The performance effects of memory architectures, such as the number of internal buses, cache size, and cache schemes are measured both for ARM7 and ARM9 systems. The overhead of clock cycles due to external DRAM accesses is shown about 40% for a typical ARM7 based system, and 19% to 1.6% for typical ARM9 based systems.

1. INTRODUCTION

The MPEG1/2 Layer-III (MP3) algorithm [1] is the most widely adopted as a compressed audio standard, and is used for consumer portable audio players. Although there are several custom VLSI-based implementations, software-based implementation using RISC processors is very needed because of the format and application flexibility. Actually, multi-standard portable players that support not only MP3 but also Dolby-AC3, AAC and WMA (Windows Media Audio) formats are now available.

In this paper, we have optimized the MP3 decoding algorithm, starting from C-level program using floating-point arithmetic, for ARM7 and ARM9 based systems. DRAM based systems, instead of a large internal fast memory, are assumed to model the implementation using PDA's or multipurpose digital players.

Since the ARM7 and ARM9 architectures do not equip a floating-point arithmetic unit, a conversion to integer arithmetic is initially conducted and algorithms that require the smaller number of arithmetic, especially multiplication, operations are selected. The ARM7 architecture, which is shown in Fig. 1-(a), has only one internal bus and one unified cache, while the ARM9 architecture, which is shown in Fig. 1-(b), has separate buses and cache blocks for program and data. Since the access of DRAM is not only slow but also power consuming, it is necessary to reduce the off-chip memory access operations.

2. ARCHITECTURE

The ARM7 and ARM9 CPU's have a relatively simple data path, where the hardware multiplier only has an accuracy of 32*8 bits, which may mean that the CPU's are not good for executing multiplication intensive digital signal processing programs.

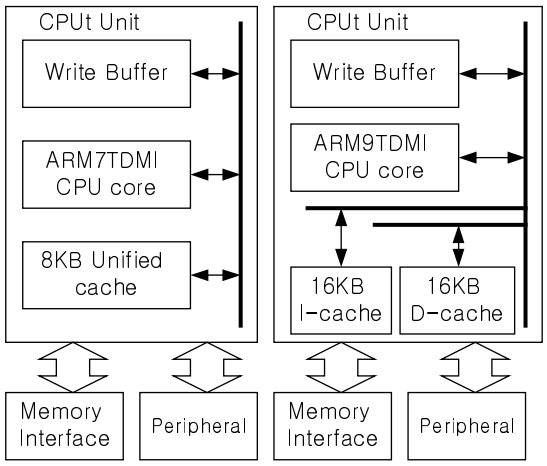


Fig.1 (a) ARM7CPU (b) ARM9CPU

However, in addition to the support of high clock frequency operation, the CPU's have a few advantageous characteristics for implementing DSP algorithms. Firstly, most of the instructions can be executed conditionally. It can help to reduce the control overhead in the MP3 decoders significantly. Secondly, it has a 32-bit barrel

shifter that can simultaneously execute shift and rotation with ALU operations. This feature is useful for scaling, multiplication by 2 constant, and so on. Thirdly, some instructions can utilize multiple registers for load and store. This can reduce total memory access time compared with transferring only one register at once [2].

Although the RISC CPU's have a very different architecture when compared to programmable digital signal processors (DSP's), many software optimization methods applied for them, such as loop fusion, loop unrolling, loop termination, circular addressing, and arranging data can also be employed. The performance can be further increased by employing ARM-CPU specific architectural features, such as barrel-shifter, post/pre increment/ decrement addressing, multiple register transfer [3].

3. PROGRAM OPTIMIZATION

MP3 decoding algorithm consists of several functional steps, which are side information parsing, scale factors, Huffman decoding, requantization, stereo processing, alias reduction, IMDCT, and synthesis via polyphase filterbank. A profiling result of the C-based MP3 decoding program is shown in Fig. 1, which shows that subband synthesis and IMDCT occupy the major part (about 84%) of the processing time. Obviously, these routines should carefully be optimized. The other part of the algorithm is the control intensive part such as requantization, Huffman decoding, unpacking scale-factor, and so on. These routines can be fairly well optimized by the conditional execution instructions of ARM processors.

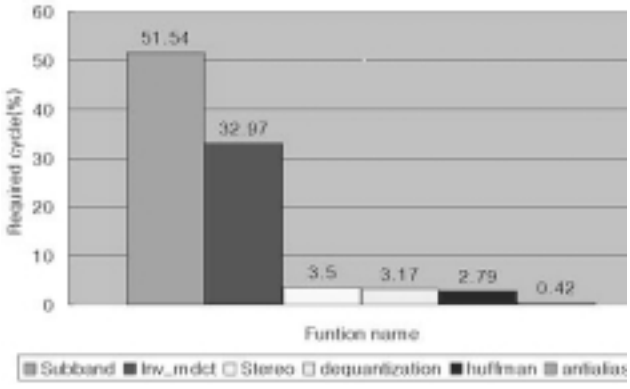


Fig.1. Floating-point profiling of MPEG audio Layer-III

In order to implement a floating-point version of DSP programs on the ARM based systems, we need to convert the floating-point arithmetic based algorithm to the integer one. An automatic scaling method that converts a floating-point arithmetic to an integer counterpart with an optimal scaling is employed [4].

3.1 IMDCT and Subband Synthesis Optimization

Although there are many efficient fast MDCT/IMDCT algorithms, most of them have been derived for data sequences with lengths of $N=2^n$. Since MDCT/IMDCT computation in Layer-III of MPEG1/2 standards employs the lengths of N=36 for long blocks and N=12 for short blocks, the Britanak and Rao's algorithm[5] is employed for this implementation. Table 1 compares the number of multiplications and additions based on the ISO reference and the chosen algorithm.

Table 1. Multiplication/Addition in MDCT of MP3

	N=36		N=12	
	Mul	Add	Mul	Add
ISO reference	648	630	72	66
Britanak & Rao [5]	47	165	13	39

Note that the number of multiplications is one of the most important concern for the algorithm selection since the ARM CPU's do not have a full precision multiplier.

For optimization of subband filtering, we applied the fast subband algorithm suggested in [6][7]. The characteristics of the routine are similar to those of the IMDCT, which contains many multiplication, load, and store instructions.

3.2 Assembly Language-level Optimization

Based on this C-program based implementation, we further optimized the IMDCT and subband synthesis code by employing the block load and store instructions, which are rarely found at the compiler generated code. Note that ARM CPU supports block data transfer, which is used for load (LDM) or store (STM) of any subset of currently visible registers to/from sequential memory. The block data transfer of 15 32-bit registers, from registers to sequential memory (STM) takes 14 sequential (S), 2 non-sequential (N), and 1 internal (I) cycles. The LDM requires a total of 15S + 1N + 1I cycles. On the other hand, the transfer of 15 32-bit registers from registers to memory using the store instruction (STR) requires (2N)*15 cycles. Note that when accessing the internal memory or cache, the number of clock cycles for N, S, and I can be all assumed to be 1, but the number of clock cycles for the non-sequential access becomes very large for external DRAM access.

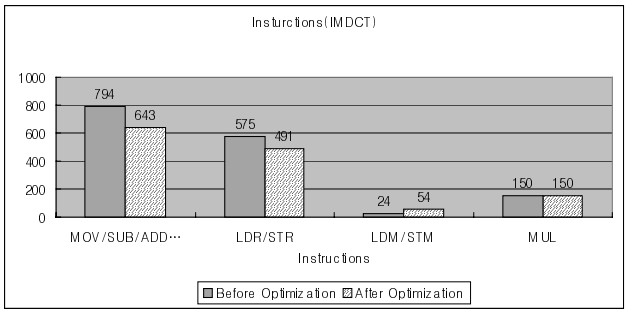
For IMDCT, the required number of instructions and clock cycles for one frame after employing LDM/STM instructions are shown in Table 2, where the number of instructions and clock cycles are 28% and 21%, respectively, decreased by the optimization. Also, for subband synthesis, the number of instructions and clock

cycles are 34% and 35% decreased, respectively.

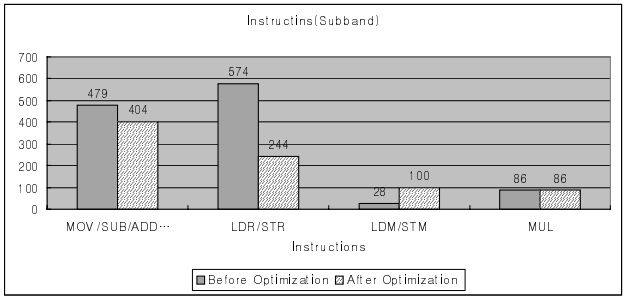
Figure 2 shows the types of instructions for IMDCT and subband synthesis implementations, where both un-optimized and optimized results are presented.

Table 2. The number of instructions and cycles after assembly language level optimization for one frame

	IMDCT		Subband	
	Ins.	Cycles	Ins.	Cycles
Before Opt.	134144	275840	345580	790986
After Opt.	97024	217472	227328	515484



(a) IMDCT



(b) Subband Synthesis

Fig 2. Instruction types in (a) IMDCT, (b) Subband synthesis

3.3 ARM7 Architecture-based Implementation

Since the ARM7 CPU only has an 8KB size of unified cache memory, the performance degradation due to cache miss can be significant. The test MP3 bit-stream is intentionally selected at the middle position to exclude the effects of silence. The simulation results that show the number of memory accesses for instruction (I) and data (D) are shown in Table 3. The results show that the cache miss ratio for instruction is about 1.76% and that for data is about 13.5%.

Table 3. Instruction vs. Data demand and miss

	I demand	I miss	D demand	D miss
Before Opt	6021044	145704	2135815	289573
After Opt	4766283	84082	1972899	266959

Optimizing in assembly language with block load and store instructions improves the spatial locality of data, which reduces the cache miss ratio and, as a result, decreases the number of accesses to/from the external DRAM. Table 4 shows the required number of external memory read and write operations for one second to decode MP3, before and after optimization.

Table 4. Required off-chip memory access for a second

	Demand Fetches (From Mem)	Demand Writes (To Mem)	Total R/W Memory
Before Opt	8.9MB	3.50MB	12.48MB
After Opt	6.5MB	3.41MB	9.92MB

3.4 ARM9 Architecture-based Implementation

The ARM9 CPU has a much improved memory architecture as shown in Fig. 1. The architecture allows instruction fetches in parallel with data accesses. While the ARM7 cache model doesn't support write allocation, ARM9 cache model supports it. Figure 3 shows data cache miss ratio in ARM9 according to the cache size. The data cache misses are greatly reduced compared with the results of Table 3, mainly due to the cache allocation scheme. Note that the instruction and data cache miss ratios with an 8KB of unified cache are 1.7% and 13.5%, respectively, but the miss ratios with two 4KB of caches, one for instruction and the other for data with write allocation support, are 0.55% and 6.08%.

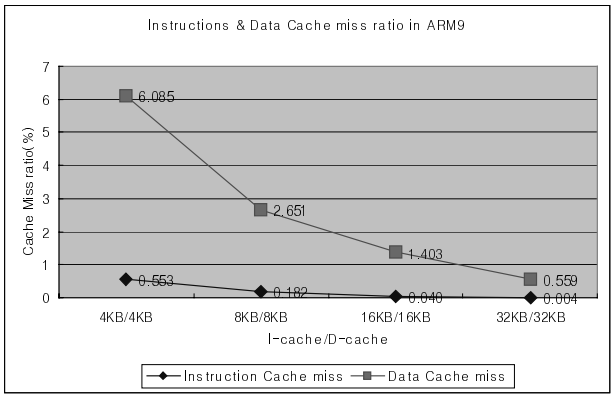


Fig. 3 instruction and data cache miss varying the cache size in ARM9

3.5 Implementation Performance

Finally, we have simulated the miss penalty between the main processor and the external SDRAM. Two different off-chip bus sizes, one is 16bit and the other is 32bit, are assumed. The DRAM is modeled having 8 clock cycles of latency for the first word read, 5 clock cycles for the first word write, and one clock cycle for successive memory read or write. This is a little pessimistic model

for SDRAM considering that the SDRAM needs a smaller latency when the same row is accessed again. It is also assumed that the SDRAM clock frequency is the same to that of the CPU.

Figure 4-(a) shows the number of clock cycles for the ARM7 based implementation for ideal memory case, 16bit SDRAM, and 32bit SDRAM buses. The results show that the 16-bit SDRAM based implementation with an 8KB unified cache requires 40% more clock cycles. Figure 4-(b) shows the results for ARM9. An ARM9 based system, which has a 16KB instruction and a 16KB data cache with 32-bit SDRAM, requires about 4% more clock cycles when compared to the ideal memory based implementation.

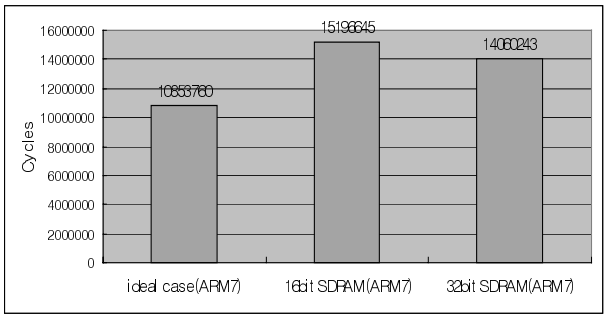


Fig.4 (a) The number of clock cycles according to different SDRAM bandwidths.

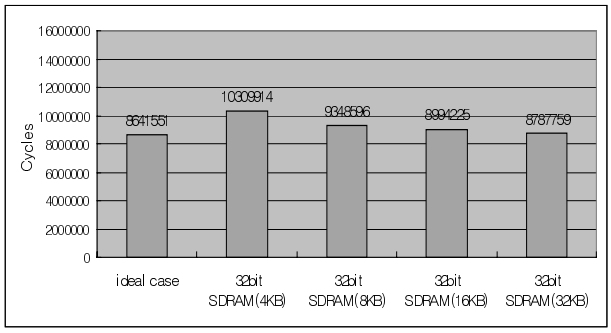


Fig.4 (b) The number of clock cycles according to different Cache size.

The quality of the implemented program is tested using the ISO reference standard, and also proved by using 10 popular pop-songs, mono, stereo, joint stereo with the sampling frequency of 44.1kHz, 22.05kHz, and bit rate varying 32kbps to 192kbps. An Average of 94.24dB SNR (Signal to Noise Ratio) is obtained, and the required MIPS (Million Instruction Per Second) for real-time operation for an ARM7 based system is about 16.5 MIPS.

4. CONCLUDING REMARKS

The MPEG1/2 Layer-III decoding algorithm is implemented in software using ARM7 and ARM9 based systems. The implementation results show that the overhead of data-transfer between the registers and memory should be considered very seriously for real-time and low-power implementation.

5. ACKNOWLEDGMENTS

This study was supported by the Brain Korea 21 Project (0019-19990027) and the National Research Laboratory program (2000–X–7155).

REFERENCES

[1] ISO/IEC 11172-3 *Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s (Part 3: Audio)*.

[2] *ARM Architecture Reference Manual*, ARM, 1996.

[3] Steve Furber, *ARM System-On-Chip Architecture*, Addison-Wesley, 2000.

[4] Ki-Il Kum, Jiyang Kang and Wonyong Sung, "AUTOSCALER for C: An Optimizing Floating-point to Integer C Program Converter for Fixed-point Digital Signal Processors," *IEEE Tr. Circuits and Systems II*, vol. 47, No. 9, Sep. 2000, pp. 840-848.

[5] Vladimir Britanak, K.R.Rao, "An Efficient Implementation of the Forward and Inverse MDCT in MPEG Audio Coding," *IEEE Signal Processing Letters*, vol.8, no.2, Feb. 2001.

[6] Konstantinos Konstantinides, "Fast Subband Filtering in MPEG Audio Coding," *IEEE Signal Processing Letters*, vol. 1, no.2, Feb. 1994.

[7] Maiko Taruki, Tadashi Sakamoto, Tomohiro Hase, "Fast MPEG-Audio Layer III Algorithm for a 32-Bit MCU," *IEEE Transactions on Consumer Electronics*, vol. 45, no. 3, Aug. 1999.