Journal of Zhejiang University SCIENCE A
ISSN 1009-3095 (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

A hardware/software co-optimization approach for embedded software of MP3 decoder*

ZHANG Wei†, LIU Peng†‡, ZHAI Zhi-bo
(*Department of Information Science and Electronic Engineering, Zhejiang University, Hangzhou 310027, China*)
†E-mail: zhang1wei3@21cn.com; liupeng@isee.zju.edu.cn

Received Feb. 23, 2006; revision accepted May 24, 2006

Abstract: In order to improve the efficiency of embedded software running on processor core, this paper proposes a hardware/software co-optimization approach for embedded software from the system point of view. The proposed stepwise methods aim at exploiting the structure and the resources of the processor as much as possible for software algorithm optimization. To achieve low memory usage and low frequency need for the same performance, this co-optimization approach was used to optimize embedded software of MP3 decoder based on a 16-bit fixed-point DSP core. After the optimization, the results of decoding 128 kbps, 44.1 kHz stereo MP3 on DSP evaluation platform need 45.9 MIPS and 20.4 kbytes memory space. The optimization rate achieves 65.6% for memory and 49.6% for frequency respectively compared with the results by compiler using floating-point computation. The experimental result indicates the availability of the hardware/software co-optimization approach depending on the algorithm and architecture.

Key words: Hardware/software co-optimization, DSP, Embedded software, MP3 decoder
doi:10.1631/jzus.2007.A0042 **Document code:** A **CLC number:** TN911.7

INTRODUCTION

The rapid evolution of current consumer electronic systems has been demanding that embedded systems have extremely short time-to-market simultaneously with very low costs. As a result, it is better by implementing large parts of the system functionality in embedded software running on processor core. The solutions range from general-purpose processor cores, digital signal processor cores (DSPs) and application-specific instruction-set processor cores (ASIPs). To achieve high efficiency of software running on processors with specific architectures, the software optimization had become a main problem (Pospiech and Olsen, 2003).

During the development of our MP3 (ISO/IEC 11172-3, 1993) decoder, the same problem was encountered. MP3 decoder adopts DSP core, RISC, and dual core architecture introduced by many researchers (Lee K.S. *et al*., 2001; 2002; Lee K.H. *et al*., 2001; Yao *et al*., 2004). But most of them mainly paid attention to the algorithm optimization and did not give a general method for embedded software optimization exploiting the processor architectural features.

As the embedded software runs on DSP and ASIP, the high-level language compiler may not make full use of the processor architectural features. Especially for lost-cost, high-volume embedded applications, insufficient computing performance and insufficient memory sizes require manual optimization (Goossens *et al*., 1997). And exploiting architectural features can be as important as choosing the right algorithms in optimizing software running on such processors or IP core with high efficiency (Anguita and Martinez-Lechado, 2005). So an approach for embedded software optimization on specific processor architecture is required.

‡ Corresponding author
* Project supported by the Key-Tech Program of Zhejiang Province, China (No. 021101559), and the Fok Ying Tong Education Foundation (No. 94031), China

Hardware/software co-design techniques are important in the development of DSP applications, especially for resource constrained SoC (System on Chip) design (Nattawut and Alex, 2004). In order to deal with partitioning and scheduling problem in DSP system, a systematic approach to hardware/software co-design targeting data-intensive applications is given (Wiangtong *et al.*, 2005). In order to decrease hardware cost and chip area, a hardware/software partition method for fixed-point DSP system design is suggested (Zhou *et al.*, 2005), and MP3 decoder based on media extensions with embedded processor is proposed (Huang *et al.*, 2005). These methods can be learned for embedded software optimization.

In this paper, a hardware/software co-optimization approach for embedded software is presented. The proposed stepwise methods aim at exploiting the structure and resources of processor as much as possible to software algorithm optimization. And with the hardware and software co-optimization, we can get the most efficiency to meet the embedded system constraints such as performance, cost, hardware resources, power consumptions, and so on.

The paper is organized as follows. A hardware/software co-optimization approach for embedded software is presented in Section 2. Software optimization of an MP3 decoder implemented on a 16-bit DSP core is described in detail in Section 3. The evaluation platform and optimization results are presented in Section 4. Finally, the conclusions are given.

HARDWARE/SOFTWARE CO-OPTIMIZATION APPROACH

Generally, software should be optimized for system performance by matching the program code to the processor architecture. This procedure includes both hardware and software parts. For specific applications in different fields, specific methods have been introduced. But in order to find a general approach exploiting the processor architectural features to do software optimization, we should not only analyze and optimize algorithm in high-level language, but also achieve full understanding of hardware features, which include the architecture, the micro-architecture, the pipeline structure, the peripheral interface, and so on. Based on such an analysis, software optimization can be mapped on these features efficiently. The approach is an interactive procedure including both hardware and software from the system point of view. According to the strategy, hardware/software co-optimization scheme is explored in the following four steps. The proposed hardware/software co-optimization flow is shown in Fig.1.

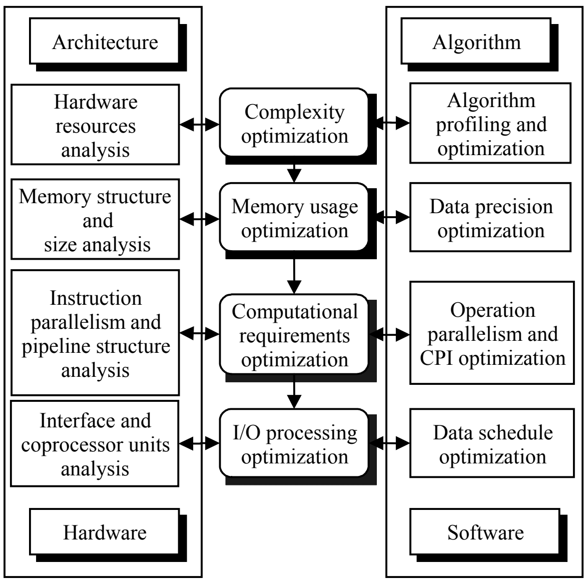
Architecture
Hardware resources analysis
Memory structure and size analysis
Instruction parallelism and pipeline structure analysis
Interface and coprocessor units analysis
Hardware
Complexity optimization
Memory usage optimization
Computational requirements optimization
I/O processing optimization
Algorithm
Algorithm profiling and optimization
Data precision optimization
Operation parallelism and CPI optimization
Data schedule optimization
Software

Fig.1 Hardware/software co-optimization flow

Step 1: Complexity optimization

The first step in complex system design is an analysis of the system under study in order to fully comprehend its basic structure, to measure its complexity, so as to discover the bottlenecks and the most critical constitutive module.

Hence, hardware limited resource should be analyzed, including memory size, arithmetic units, register, and data type. And algorithmic complexity for processor computation, storage and data-transfer ability should be analyzed. During algorithm optimization, hardware analysis results should be taken into account. There must be trade-off among computation load, the complexity of data conveying and the size of coefficients. Software modules are partitioned and performance is estimated in C language. This process can help designer to decide the program structure and find optimization goals. The hardware and software analysis results will guide the software optimization work.

Step 2: Memory usage optimization

The optimization of memory usage is one of the most critical steps in the development of efficient and low-power implementations. The memory structure, the bit width of datapath and register need to be analyzed. As for limited hardware memory size, program and data optimization are both required.

Memory usage is partitioned into several sections, such as program code, coefficient tables, temporal variable, global variable, stacks, and so on. The data precision and coefficient table length need to be considered in this step. And also the stack and buffer size need to be optimized for specific field with different system requirements.

Step 3: Computational requirements optimization

In a cost and power-sensitive embedded system, lower frequency requirements mean low power dissipation, which are better for the same performance. The instruction-level parallelism needs to be explored and the pipeline stall needs to be eliminated as much as possible to reduce computational requirements. Exploiting the software processing parallelism is also important. Generally, different modules with low dependency can be parallelized. And for a special processor, specific algorithm may be adopted to exploit the architectural features.

To reduce CPI (cycles per instruction), main efforts are paid to reduce control hazards of the pipeline micro-architecture. Loop unrolling and instruction reordering are performed to reduce CPI and exploit the instruction-level parallelism.

Step 4: I/O processing optimization

The I/O processing is becoming a main problem as important as software processing on the embedded processor core. Sometimes the data transfer occupies a large part of total processing time.

The interface and coprocessor units for I/O processing of embedded processor are analyzed and employed. Data schedule mechanism needs to be optimized to reduce the data transferring demands and I/O bandwidth requirements. And DSP processing in core and in I/O can be parallelized by exploiting hardware units efficiently.

HARDWARE/SOFTWARE CO-OPTIMIZATION TECHNIQUES FOR MP3 DECODER

MP3 is the most popular decoding format for playback of high quality compressed audio for portable devices. According to ISO 11172-3 standard, MP3 decoding process can be divided into nine modules, as shown in Fig.2.

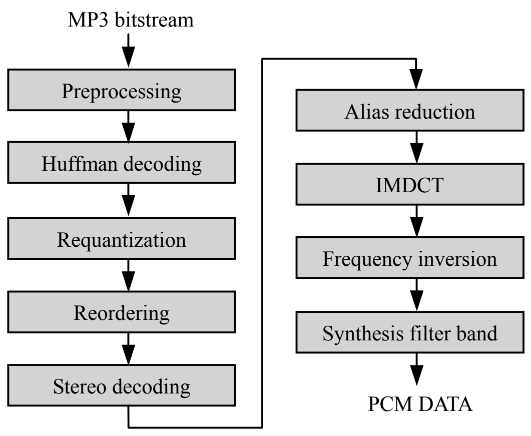

Fig.2 MP3 decoding process

MediaDSP16, a 16-bit DSP core is chosen as the target processor, which is designed by the Department of Information Science & Electronic Engineering of Zhejiang University (Chen *et al*., 2004). The architecture of MediaDSP16 shown in Fig.3 has the following features:

(1) 16-bit fixed-point DSP.

(2) Four stages pipeline (IF, ID, EX and MEM).

(3) Three computational units (ALU, SHIFTER and MAC).

(4) Two address generating units (DAG1, DAG2).

(5) Modified Harvard architecture.

(6) Single-cycle instruction execution.

(7) Dual operand fetches in one cycle.

(8) Multifunction instructions.

(9) Zero-overhead loop.

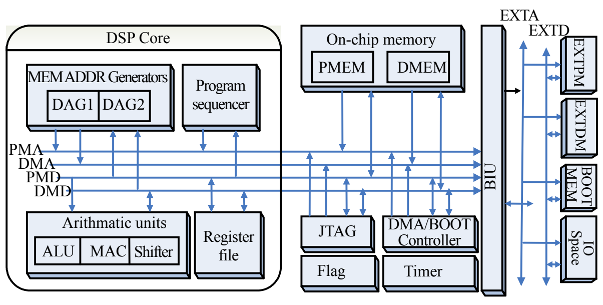

Fig.3 MediaDSP16 architecture

MediaDSP16 architecture has some RISC features in the micro-architecture design, including methods of local-homogenous register set and RISC-like pipeline to eliminate control and data hazards. DSP can provide powerful computation ability, which includes the abundant addressing mode and instruction level parallelism. MP3 decoder is implemented based on this DSP by using the proposed hardware/software co-optimization approach.

During the optimization process, the features of instruction set, micro-architecture, and pipeline of DSP are taken into account for effective running and lower CPI value. C language is used to build the program framework and assemble language is used to write some time-consuming modules in order to balance the flexibility and efficiency.

Step 1: Complexity optimization

The target DSP hardware resources have been carefully calculated to map this algorithm. The DSP provides separated 16k words on-chip data memory (16-bit width) and 16k words on-chip program memory (24-bit width). The on-chip program memory can be reconfigured as no-overlap program section and data section. The DSP supports multi-mode such as index and circular addressing and multifunction instructions to realize computation and memory access simultaneously. Abundant memory access modes, instruction-level parallelism, enhanced control ability of DSP can benefit both the control-intensive part and computation-intensive part of MP3 decoding algorithm.

Algorithm profiling and optimization are adopted in the software. The statistical results of profiling of the main steps of ISO reference MP3 decoder employing floating-point computation is shown in Table 1. Subband synthesis, IMDCT, requantization, Huffman decoding, are most time-consuming modules in MP3 decoder.

Table 1 Profiling of ISO reference decoder

Module	CPU time (%)
Huffman decoding	4.1
Requantization	16.9
Stereo processing	1.2
IMDCT	17.9
Subband synthesis	58.2
Other	1.7

Fast algorithms (Lee, 1984; Konstantinides, 1994; Britanak and Rao, 2001) are adopted in the time-consuming modules to do high-level optimization, which can reduce the computation load and memory requirement for DSP.

Konstantinides (1994)'s method reduces the number of operations in synthesis polyphase filter bank by transforming the matrix operation in a 32 discrete cosine transform (DCT) and some reorder operations. We implement DCT using Lee (1984)'s fast DCT algorithm, which divides DCT recursively into two smaller DCTs. This method eliminates 96% of the multiplications and 90% of the additions.

Following Britanak and Rao (2001)'s method, we reduce IMDCT to a fast DCT computation and some data copying operations. This method eliminates 93% of the multiplications and 74% of the additions for long blocks. For short blocks, it eliminates 82% of the multiplications and 41% of the additions. The results of operation reduction are shown in Table 2.

Table 2 Complexity optimization results

Algorithm		IMDCT (operations)		IDCT (operations)
		Long block	Short block	
ISO reference	Mul	648	72	2048
	Add	630	66	1984
Fast algorithm	Mul	47	13	80
	Add	165	39	209

Step 2: Memory usage optimization

For the modified Harvard architecture of this 16-bit fixed-point DSP, part of data can be relocated in program memory. And for the 16-bit memory and register width, 16-bit data precision is used in processing. For limited memory, proper coefficient table precision needs to be analyzed and calculated.

Since DSP has 16-bit data path, multiply operation using data in excess of 16 bits will be calculated by more than one instruction. Therefore, not only the accuracy for output data but also operational complexity need to be considered. Data precision was optimized to decrease the computation complexity while not causing too much sound quality loss. Because of the 16-bit data width, we should make a trade-off between dynamic range and data precision in the floating-point to fixed-point conversion.

In MP3 decoding process, fixed-point program conversion was needed for the following modules: requantization, joint stereo decoding, IMDCT, anti-aliasing and synthesis. In each module, the dynamic range of each of the operation data was analyzed to do proper scaling.

For example, first precision loss arises in the requantization stage where the Huffman data are requantized to spectral values. The equation is

$$xr_i=\mathrm{sgn}(is_i)\times|is_i|^{4/3}\times 2^a,$$

where

$$a=(global\_gain[gr]-210)/4-scalefac\_multiplier\times \{scalefac\_1[sfb][ch][gr]+preflag[gr]\times pretab[sfb]\}.$$

For each output value '*is*' from the Huffman decoding and $is^{4/3}$ can be done either by table lookup or by explicit calculation. For efficiency of operation, table lookup method is preferred in calculation of $is^{4/3}$. But because of the wide dynamic range of is_i, this lookup table requires a large amount of memory usage. To reduce the memory usage without significant loss of accuracy caused by truncation errors, a lookup table size reduction algorithm was adopted: $|is|^{4/3}=|is'|^{4/3}\times 16$, where $is'=is/8$.

This algorithm reduces lookup table size by a factor of 8. The value of *is* varies from 0 to 8206 (The standard value is 8191, but actually it should be 8206 with 15 must be add to the given value). However, the probability of *is*, which is greater than 256, is very low according to the results of statistics taking from many MP3 test streams. Therefore, 256-sized lookup table is adopted and for the higher is_i value linear interpolation method is applied to approximate it. By using this method in requantization, accuracy loss is kept in a small range, when the memory usage and computation complexity reduced obviously. The data memory and program memory usage optimization results are shown in Table 3 and Table 4.

Table 3 Data memory usage optimization results

Algorithm	Data space		
	Before optimization (byte)	After optimization (byte)	Optimization ratio (%)
Huffman	5640	3400	39.7
IMDCT	5160	2588	49.8
Synthesis	10996	5904	46.3

Table 4 Program memory spaces optimization results

Algorithm	Program space		
	Before optimization (byte)	After optimization (byte)	Optimization ratio (%)
Huffman	4503	3107	31.0
IMDCT	6900	2649	61.6
IDCT	2325	1308	43.7
Total	40533	20307	49.9

Step 3: Computational requirement optimization

The DSP provides instruction-level parallelism with parallel execution of computation and memory access, which means the full use of system resources in each cycle. To realize effective memory access, the DSP provides two dedicated address-generating units (DAG1, DAG2) to support multi-mode such as index and circular addressing together with high memory access bandwidth. One computation combined with move operation (include load/store) are supported. In one cycle, five operations can be completed with the multifunction instructions at most, including one computation, two memory access, two data pointers update.

In order to use multifunction instructions to reduce the number of instruction for computation, the operation parallelism was carefully analyzed and properly arranged to exploit hardware features. To improve pipeline running efficiency, instruction sequence was reorganized manually and loop instruction was used to optimize the CPI by delay slot techniques.

For example, in the IMDCT module, the main time-consuming computation formula is

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

The main computation is multiplication and accumulation. The code is relatively short size with plenty of loop operation. It is suitable for multifunction instruction to improve data parallelism. In the computation, one operand is fetched from the coefficient table, and the other operand is in data memory from the results of the last module processed. As DSP can fetch dual operands in one cycle, the coefficient table can be put in the program memory. Moreover, multifunction instructions support the computation and two memory accesses in one cycle. So one MAC computation and two operands load can be finished in one cycle. Besides this, zero-overhead loop operation supported by DSP was used to cut down pipeline stall. Also IDCT computation can utilize these mechanisms, data and instructions need to be arranged properly, and parallel operation and CPI reduction can be realized by hardware features.

The optimization reduces the program instructions and execution cycles. The results are shown in Table 4 and Table 5.

Table 5 Computation requirements optimization results

Algorithm	Execution cycles			
	Before optimization (byte)	After optimization (byte)	Optimization ratio (%)	CPI
Huffman	201704	90576	55.1	1.166
IMDCT	311165	105808	65.0	1.286
IDCT	11398	3219	72.8	1.477
Total	3511322	1371052	60.9	1.255

Step 4: I/O processing optimization

Data transfer and scheduling were optimized by utilizing the DMA unit in hardware. According to limited on-chip memory resources, the raw MP3 stream and the decoded stream are stored in off-chip memory, while the frequently used coefficients are stored in on-chip RAM. So data scheduling becomes a bottleneck in the decoding process. In order to reduce the memory access times, buffer mechanism was used in the on-chip DM, and DMA unit was used to transfer the raw MP3 stream data to buffer and the decoded PCM data to audio D/A.

Compared with shift bitstream buffer, circular bitstream buffer was used to reduce data transfer from buffer end to head. As DMA transfers data without CPU interference, the I/O processing and decoding processing can be parallelized. The mechanism of DMA transfer with circular buffer is shown in Fig.4.

While I/O processing occupies nearly 10% of total processing time, with parallel operation, much I/O processing time is saved. MP3 decoder needs 45.9 MIPS after optimization compared with 52.1 MIPS before optimization, which gets 11.9% optimization ratio.

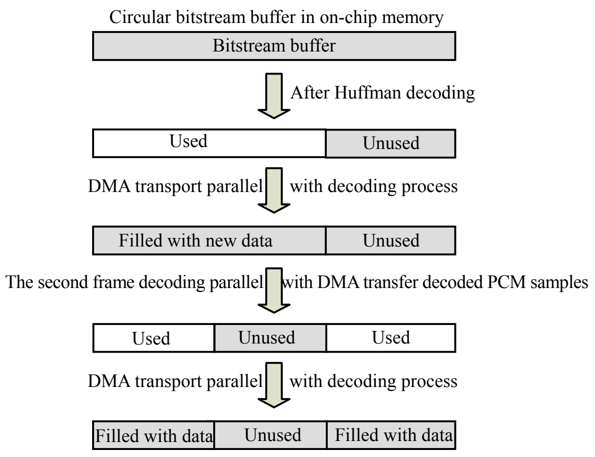
Circular bitstream buffer in on-chip memory
Bitstream buffer
After Huffman decoding
Used
Unused
DMA transport parallel with decoding process
Filled with new data
Unused
The second frame decoding parallel with DMA transfer decoded PCM samples
Used
Unused
Used
DMA transport parallel with decoding process
Filled with data
Unused
Filled with data

Fig.4 Circular bitstream buffer with DMA transport

EVALUATION PLATFORM AND OPTIMIZATION RESULTS

In order to evaluate the optimization results of MP3 decoder, an evaluation platform including hardware and software is designed.

The software platform includes compiler, assembler, simulator, and communication module, which uses USB interface to communicate with hardware platform. The connection diagram of the evaluation board is shown in Fig.5. As FPGA has programmable ability, many interface modules have been implemented, including communication cores based on USB protocol, memory controller, audio interface controller. A photograph of a hardware evaluation platform equipped with the FPGA is shown in Fig.6.

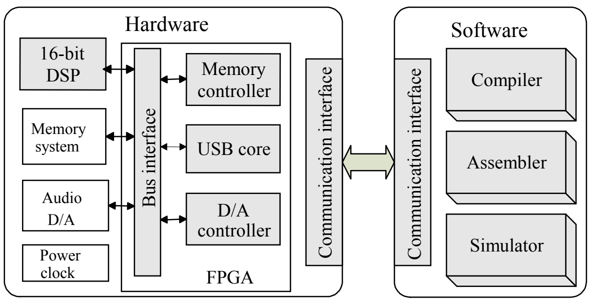

Fig.5 Diagram of the evaluation platform

This platform was used to evaluate the optimization performance of MP3 decoder. Before optimization, because compiler cannot utilize the architectural

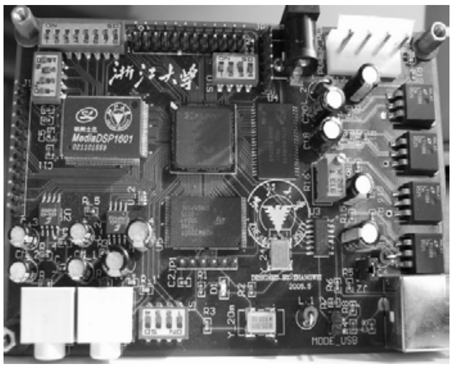

Fig.6 Evaluation platform photograph

features and fast algorithms are not implemented, the result is not satisfactory. After the optimization, especially the hardware/software co-optimization, the architectural features can sufficiently be exploited, and nearly 65.6% computation load and 49.6% memory usage can be optimized. The total optimization results are shown in Table 6.

Table 6 Total optimization results

	Before optimization	After optimization	Optimization ratio (%)
Program space (kbytes)	40.5	20.4	49.6
Execution cycles (MIPS)	133.4	45.9	65.6

In order to measure the sound quality of output PCM for proposed optimization approach, a comparison between the decoded floating-point data and fixed-point data is provided in Table 7. Although coefficient precision is reduced, the accuracy in decoded audio quality does not decrease much.

Table 7 Data precision optimization results

Total samples	Different samples	RMS of difference (dB)	Max difference
276524	103220	55.2	102

CONCLUSION

In this paper, a hardware/software co-optimization approach for embedded software is proposed from the system point of view. The proposed stepwise methods aim at exploiting the structure and resources of the processor as much as possible for software algorithm optimization. According to these optimization techniques, software optimization of an MP3 decoder based on 16-bit fixed-point DSP has been implemented. The experimental result indicates the availability of this approach depending on the algorithm and processor architecture. In future work, this approach can be generalized for other multimedia algorithm optimizations on embedded processor.

References

Anguita, M., Martinez-Lechado, J.M., 2005. MP3 optimization exploiting processor architecture and using better algorithms. *IEEE Micro.*, **25**(3):81-92. [doi:10.1109/MM.2005.57]

Britanak, V., Rao, K.R., 2001. An efficient implementation of the forward and inverse MDCT in MPEG audio coding. *IEEE Signal Processing Letters*, **8**(2):48-51. [doi:10.1109/97.895372]

Chen, J.C., Yao, Q.D., Liu, P., Shi, C., 2004. MD16, DSP with Some RISC Features for Embedded System. IEEE Int. Conf. Signal Processing Proceedings, p.144-147.

Goossens, G., van Praet, J., Lanneer, D., Geurts, W., Kifli, A., Liem, C., Paulin, P.G., 1997. Embedded software in real-time signal processing systems: design technologies. *Proc. IEEE*, **85**(3):436-454. [doi:10.1109/5.558718]

Huang, W.K., Lin, I.T., Chen, S.W., Huang, I.J., 2005. A Cost-effective Media Processor for Embedded Applications. ISCAS 2005, p.6122-6125.

ISO/IEC 11172-3, 1993. Informational Technology—Coding of Moving Pictures and Associated Audio for Digital: Storage Media at up to about 1.5 Mbit/s. Part 3: Audio. 1st Ed.

Konstantinides, K., 1994. Fast subband filtering in MPEG audio coding. *IEEE Signal Processing Letters*, **1**(2):26-28. [doi:10.1109/97.300309]

Lee, B.G., 1984. A new algorithm to compute the discrete cosine transform. *IEEE Trans. Acoustic Speech Signal Processing*, **32**(6):1243-1245. [doi:10.1109/TASSP.1984.1164443]

Lee, K.H., Lee, K.S., Hwang, T.H., Park, Y.C., Youn, D.H., 2001. An architecture and implementation of MPEG audio layer3 decoder using dual-core DSP. *IEEE Trans. Consumer Electronics*, **47**(4):928-933. [doi:10.1109/30.982810]

Lee, K.S., Oh, H.O., Park, Y.C., Youn, D.H., 2001. High Quality MPEG-Audio Layer3 Algorithm for a 16-bit DSP. ISCAS 2001, p.205-208.

Lee, K.S., Park, Y.C., Youn, D.H., 2002. Software optimization of the MPEG-audio decoder using a 32-bit MCU RISC processor. *IEEE Trans. Consumer Electronics*, **48**(3):671-676. [doi:10.1109/TCE.2002.1037059]

Nattawut, T., Alex, D., 2004. Hardware-software Co-design of Resource Constrained Systems on a Chip. Distributed Computing Systems Workshops, Proceedings 24th International Conference, p.818-823.

Pospiech, F., Olsen, S., 2003. Embedded Software in the SoC World. How HdS Helps to Face the HW and SW Design Challenge. Proc. IEEE, Custom Integrated Circuits Conference, p.653-658.

Wiangtong, T., Cheung, P.Y.K., Luk, W., 2005. Hardware/software codesign: a systematic approach targeting data-intensive applications. *IEEE Signal Processing Magazine*, **22**(3):14-22. [doi:10.1109/MSP.2005.1425894]

Yao, Y.B., Yao, Q.D., Liu, P., Xiao, Z.B., 2004. Embedded software optimization for MP3 decoder implemented on RISC core. *IEEE Transactions on Consumer Electronics*, **50**(4):1244-1249. [doi:10.1109/TCE.2004.1362526]

Zhou, F., Yang, J., Shi, L.X., Zhang, Y., 2005. Hardware-software Partition of Fixed-point Hardware Accelerator from Statistical Perspective. International Conference on ASIC, p.148-151. [doi:10.1109/ICASIC.2005.1611272]