DESIGN OPTIMIZATION OF MAIN-PROFILE MPEG-2 AAC DECODER

Kyoung-Ho Bang[*], *Nam-Hun Jeong*[*], *Young-Cheol Park*[**], *and Dae Hee Youn*[*]

[*]ASSP Lab., Dept. of Electrical and Electronic Eng., Yonsei University, Seoul, KOREA

[**]CSPR, Yonsei University, Seoul, KOREA

E-mail: euphony@cyclon.yonsei.ac.kr

ABSTRACT

In this paper, a system architecture optimized for the 2-channel main-profile MPEG-2 AAC decoder is presented. In order to enable an efficient job scheduling and allocation, the presented system comprises three hardware modules: Huffman decoder module, predictor module, and processing core module which is programmable using an assembly language of its own. Huffman decoder is designed to finish the requested job in only 1 clock cycle time and the predictor forms parallel processing with other modules, so that utilization of the system resource is maximized. The developed system has been coded in VHDL and MPEG-2 AAC decoding algorithm is programmed using the assembly of the processing core. For the verification of decoding algorithm, the 16-bit PCM output of the system was compared with the result of the floating-point simulation, and the result showed the maximum of 2-bit difference. Functional simulation verified that the developed system can decode standard MPEG-2 AAC main-profile bitstreams in real-time with high accuracy.

1. INTRODUCTION

In recent contexts of multimedia and applications, new demands for audio coding arise. As the high coding efficiency to cope with limited bandwidth has become the issue of most importance, a new MPEG standard, called advanced audio coding (MPEG-2 AAC, ISO/IEC 13818-7 [1]) has been standardized. This new standard exhibits many advantages over other MPEG audio standards. The MPEG-2 AAC standard provides very high audio quality without compatibility-based restrictions. It combines the coding efficiency of a high-resolution filter bank, prediction technique, temporal noise shaping (TNS) technique, and Huffman coding [1][2]. Due to these advantageous features, this new standard constitutes the kernel of the MPEG-4 AAC audio standard [3].

MPEG-2 AAC system offers different tradeoffs between quality and complexity depending on the application. For this purpose three profiles have been defined: main profile, low-complexity (LC) profile, and sample rate scalable (SRS) profile [1]. Among these three profiles, the main profile provides the highest quality but also demands the highest complexity since it contains all the tools the exception of the gain control tool [2].

This work was supported by Electronics and Telecommunications Research Institute (ETRI).

Thus, for the implementation of the main-profile MPEG-2 AAC decoder, heavy computational load and memory resource requirement need to be resolved. These problems are mainly concerned with Huffman decoding, prediction, and a high-resolution filter bank, among the routines constructing MEPG-2 AAC decoding algorithm. The characteristic of the tools are summarized as followings:

- Huffman decoding: The Huffman code uses variable-length codewords to further reduce the redundancy of the scalefactors and the quantized spectrum data. One scalefactor Huffman codebook and 11 spectrum Huffman codebooks are used in AAC. But storing these Huffman codebooks requires large memory. Also, codebook search demands a large amount of computational cycles.
- Prediction: Prediction is used to improve redundancy reduction and is especially effective for stationary signals. A second-order backward-adaptive predictor reduces the bit rate for coding subsequent subband samples in a given subband based on the quantized spectrum of the previous frame. But the state variables of the previous frame are associated with heavy memory resources.
- High-resolution filter bank: Rather than the hybrid filter bank of MPEG audio layer-3, AAC uses a modified discrete cosine transform (MDCT). Together with the increased window length, adapted window shape function and transform block switching, MDCT outperforms the filter banks of previous coding methods and provides better frequency selectivity for the filter bank. But this filter bank produces high computational loads.

In this paper, we propose a system architecture that is optimized for the 2-channel MPEG-2 AAC decoder supporting the main profile. To solve the problems mentioned above, we developed an efficient processing architecture. The developed system comprises three hardware modules: Huffman decoder module, predictor module, and processing core module which is programmable using an assembly language of its own. By forming the hardware modules in parallel processing structure, we maximized the system efficiency. Also, in order to ease the burden due to the high-resolution filter bank, the fast IMDCT algorithm suggested by Duhamel *et al.* [4] is implemented in Assembly software.

The organization of this paper is as follows. A design optimization of AAC algorithm is developed in Section 2, followed by detailed description of the architectural design in Section 3. The result of the simulation and the verification are presented in Section 4, and conclusions are reached in Section 5.

2. DESIGN OPTIMIZATION

2.1 Huffman decoding tool

In the MPEG-2 AAC decoder, one of the 12 Huffman codebooks is selected by the side information of bitstream. Then, Huffman codeword is unpacked from bitstream referring selected Huffman codebook. The quantized spectral coefficients are obtained using Huffman code index of unpacked Huffman codewords. Since Huffman decoding tool contains 12 Huffman codebooks, this tool needs large memory space. Unpacking Huffman codeword is associated with various operations demanding relatively low computational power, which makes the control of DSP core laborious.

In this study, we designed a Huffman decoder using fast hardwired logics. The designed Huffman decoder can decode one Huffman code index in 1 clock cycle time. Table 1. compares the hardware implementation of Huffman decoder with software implementation. Based on these, it is clear to see that the hardware implementation provides incomparable efficiency to the software implementation.

Table 1. Huffman decoder according to implementation method

	Implemented by H/W	Implemented by S/W
Method	Combinational logic (logic-gate & MUX)	Searching codebook (worst case 289 times)
Speed	1 clock cycle	Variable according to cases
Memory requirement	Unnecessary	Fairly large-sized ROM
Efficiency	Efficient	Inefficient

2.2 Prediction tool

MPEG-2 AAC uses the prediction tool to improve redundancy reduction and the tool is especially effective for stationary signals. Predictor exploits the autocorrelation between the spectral component values of consecutive frames, and uses a second-order backward-adaptive lattice filter to predict each spectral coefficient. For each spectral component up to 16 kHz there is one predictor and an LMS algorithm is used for each predictor [2].

So, prediction tool requires a great number of floating-point multiplications and additions, and a large number of storage elements for the status variables of each predictor. In this study, we implemented the predictor using hardwired logic containing floating-point arithmetic unit and memory module. Since the prediction and other processes are independent, they can be operating in parallel to increase the processing speed of the decoding system.

2.3 Fast T/F transform

Filter bank tool performs the conversion of the time-frequency representation of the encoded signal into the time-domain signal. This conversion is done by the inverse modified discrete cosine transform (IMDCT), which employs a technique called time-domain aliasing cancellation (TDAC) [2].

The analytical expression for the IMDCT is

$$x_{i,n} = \frac{2}{N}\sum_{k=0}^{\frac{N}{2}-1} X_{i,k} \cos\left(\frac{2\pi}{N}(n+n_0)\left(k+\frac{1}{2}\right)\right), \quad n = 0,\ldots, N-1, \qquad (1)$$

where n=sample index

N=transform block length

i = block index

$n_0 = (N/2+1)/2$.

Since the method of computing $x_{i,n}$ using Eq.(1) contains many computational redundancies, a fast algorithm should be used. In this study, we implemented the filter bank tool using the fast IMDCT algorithm proposed by Duhamel *et al.* [4]. This algorithm employs an N/4-point complex FFT for the N-point IMDCT, so that it can reduce the computational loads of overall system by a factor of 10. Fig. 1. shows a flow diagram of the algorithm

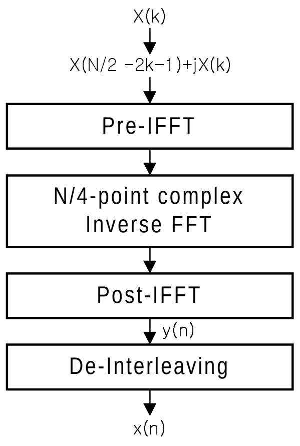

Fig. 1. Flow diagram of the fast IMDCT algorithm by Duhamel *et al.* [4].

2.4 Proposed system architecture

The architecture of the developed system is shown in Fig. 2. The overall system consists of a dedicated processing core and two hardwired logic modules.

The hardwired logic modules are the Huffman decoding module and the prediction module. Due to the high computational loads of Huffman decoding tool, its architecture design must be focused on the fast operation. Since prediction tool needs high-accuracy operations, high computational power, and a large amount of memory requirement, it must be designed to contain floating-point arithmetic unit and external memory modules.

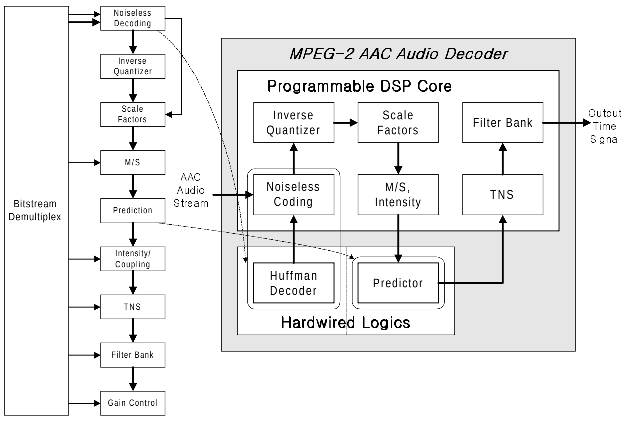

Fig. 2. The overall architecture of the developed MPEG-2 AAC decoding system .

The processing core is a 20-bit fixed-point programmable processor suitable for audio signal processing [5]. It performs noiseless coding, inverse quantizer, scalefactors, M/S intensity stereo, TNS, and filter bank tool.

3. DESIGN OF HARDWARE MODULES

3.1 Huffman decoder

The Huffman decoding module is designed using combinational hardwired logic to reduce the computational loads. This module consists of 12 Huffman table modules, and Huffman table select logic selects one of 12 Huffman table modules. Each Huffman table (codebook) ROM and a Huffman index select logic, which compares bitstream data with Huffman table ROM data and returns the adequate Huffman code index. With this architecture, the decoding process takes only 1 clock cycle time. The architecture of Huffman decoding module is shown in Fig. 3.

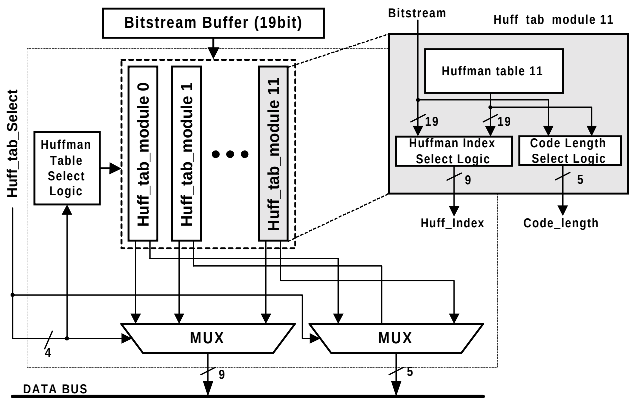

Fig. 3. Huffman decoder architecture

3.2 Predictor

Prediction tool uses a second-order backward lattice filter to predict each spectrum coefficient. The resolution of the prediction tool should support high-accuracy operations. In this design, we employed a 16-bit floating-point arithmetic unit. By using a floating-point unit, it is also possible to reduce the power consumption of the system. DAG module generates the address of various status variables of predictor, and predictor controller performs control between arithmetic unit and DAG module. The architecture of predictor is shown in Fig. 4.

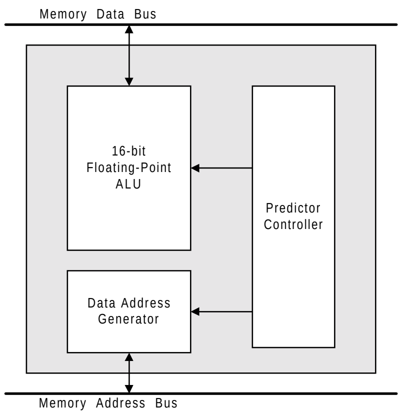

Fig. 4. Predictor architecture

3.3 Processing core

Processing core module is an application-specific dedicated processor, whose architecture is suitable for audio signal processing. It adopts 3-stage pipeline for enhancing the performance and the Harvard architecture for efficient pipelining. This processor can execute minimal clock speed of 40 MHz.

The processing core is programmable using its own assembly language and it supports special instructions like UNPACK, HUFFMAN as well as general arithmetic and logical instructions including hardware MAC. Especially, UNPACK is a useful instruction for bit-parsing. All instructions are completed within a single cycle.

4. SYSTEM EFFICIENCY

To verify the implemented system, two verifications were applied to the designed system. First verification was to examine the validation of the decoding algorithm with the designed system, and second one was to verify the feasibility of the real-time operation for the decoding process. For these two verifications, we used the C-language based simulator.

Algorithmic verification was carried out by comparing the results of floating-point and fixed-point simulations. These two simulations were performed using C-language in algorithm design step. The floating-point simulation was for the verification of the MPEG-2 AAC decoding algorithm and the fixed-point one concerned is with reducing the error due to finite word-length restriction. The requirements of ISO/IEC 13818-4 compliance test [6] are that NL (noise level) be less than –101dB FS and MER (maximum error ratio) be less than 1. The performance of the implemented system is shown in Table 2. Np is the number of processing-bit and No is the number of output PCM bit.

Table 2. The test of the implemented decoding system

	No = Np		No = 16		No = 20	
Np	NL	MER	NL	MER	NL	MER
16	-85.6	5.05	-85.6	5.05	-85.6	5.05
17	-90.5	3.15	-90.3	3.19	-90.5	3.15
18	-94.6	2.27	-94.0	2.33	-94.6	2.27
19	-97.7	1.39	-96.5	1.62	-97.7	1.39
20	-100.2	1.11	-97.8	1.25	-100.2	1.11
21	-104.6	0.65	-100.0	0.75	-104.5	0.65
22	-109.8	0.64	-100.9	0.64	-109.7	0.64
23	-114.8	0.64	-101.2	0.64	-114.5	0.64
24	-118.1	0.64	-101.3	0.64	-117.3	0.64

The verification of real-time decoding was carried by comparing the number of the clock cycles in the worst simulation case with that of the required clock cycles for the real-time decoding. When 48kHz sampling frequency is considered and the processing core operates at 40MHz, the number of clock cycles for decoding one time sample in real-time is

$$f_{clk} \times \frac{1}{F_s} = 40 \times 10^6 \times \frac{1}{48 \times 10^3} = 833.3(cycles)$$

Therefore, the decoding system must finish the entire processing in less than 853,333 clock cycles a frame. Table 3 summarizes the required clock cycles in each decoding step. Table 2 indicates that designed MPEG-2 AAC decoding system consumes less than 361,828 cycles to decode one frame of audio. For the results of Table 3, worst simulation case was considered. The result of real-time verification shows that the designed system can decode a 2-channel MPEG-2 AAC main profile bitstream in real time with high efficiency. The complexity of the implemented system is summarized in Table 4. The proposed design is implemented in VLSI using SAMSUNG 0.35μm 3.3V CMOS technology and the gate count of the system is summarized in Table 5. This decoding system uses less resource and complexity than other case can be found in [7].

Table 3. Clock cycles to decode MPEG-2 AAC bit-stream

Decoding process	**No. of cycles**	**MIPS**
Noiseless coding	112,140	5.26
Stereo	13,560	0.64
Prediction	15,430	0.65
TNS	105,370	4.94
Filter bank	115,328	5.41
Total sum	**361,828**	**16.9**

Table 4. Complexity of the implemented system

Program Memory	4.1k word
Data ROM	5.5k word
Data RAM	7.1k word

Table 5. The gate count of MPEG-2 AAC decoder

DSP Core	23145.0 gates
Huffman Decoder	3987.5 gates
Predictor	6673.0 gates

5. CONCLUSIONS

We have designed a real-time MPEG-2 AAC decoding system supporting the main-profile. The designed system has a hybrid-architecture of a fixed-point processing core for the software implementation and two hardwired logic modules: Huffman decoder module and predictor module. The designed system can support all decoding tools except for coupling channel tool, and sampling rates of 32, 44.1, 48kHz.

To verify the designed system, simulator model has been used in C-language. The 16-bit PCM output of the system was compared with the result of the floating-point and fixed-point simulation to evaluate the accuracy of the system. The result of simulation showed that both the implemented system and fixed-point C model provide the same accuracy. The results of floating-point and fixed-point simulations showed the maximum of 2-bit difference when the 16-bit PCM outputs were compared.

Required clocks for the real-time decoding were calculated as 853,333 cycles a frame which when 40 MHz DSP core was considered. The designed system needed only 361,828 cycles (16.9 MIPS) a frame for the decoding of 2-channel MPEG-2 AAC main profile bitstreams with high efficiency.

6. REFERENCES

[1] ISO/IEC JTC1/SC29/WG11 No. 1650 "IS 13818-7 (MPEG-2 Advanced Audio Coding, AAC)", Apr., 1997.

[2] M. Bosi and *et al*., "ISO/IEC MPEG-2 Advanced Audio Coding, *J. Audio Eng. Soc.*, Vol. 45, No. 10, pp. 789-814, Oct., 1997.

[3] ISO/IEC JTC1/SC29/WG11 No. 2203TF "IS 14496-3 (Information Technology - Coding of Audiovisual Objects – Part 3: Audio – Subpart 4: Time/Frequency Coding)", May, 1998.

[4] P. Duhamel, Y. Mahieux, and J. P. Petit, "A fast algorithm for the implementation of filter banks based on 'time domain aliasing cancellation'" *Proc. ICASSP*, May 1991, pp. 2209-2212.

[5] K.H. Bang, J.S. Kim, N.H. Jeong, K.S. Lee, Y.C. Park, and D.H. Youn, "VLSI Design of MPEG-2 AAC Audio Decoder", *Proc. KSPC*, pp. 247-250, Sep., 2000.

[6] ISO/IEC 13818-4 (Information Technology – Generic Coding of Moving Pictures and Associated Audio: Conformance)", Mar., 1996

[7] http://www.arm.com/sitearchitek/armtech.ns4/163f0eba66e77d004125665000520db4/fe1c0898732538f98025691e0054f482!OpenDocument