International Journal of Fuzzy Logic and Intelligent Systems, vol. 8, no. 1, March 2008 pp. 75-81

On Top-Down Design of MPEG-2 Audio Encoder

Sung-Wook Park

SAMSUNG ELECNTRONICS CO., LTD, DM R&D Center

Abstract

This paper presents a top-down approach to implement an MPEG-2 audio encoder in VLSI. As the algorithm of an MPEG-2 audio encoder is heavy-weighted and heterogeneous(to be mixture of several strategies), the encoder design process is undertaken carefully from the algorithmic level to the architectural level. Firstly, the encoding algorithm is analyzed and divided into sub-algorithms, called tasks, and the tasks are partitioned in the way of reusing the same designs. Secondly, the partitioned tasks are scheduled and synthesized to make the most efficient use of time and space. In the end, a real-time 5 channel MPEG-2 audio encoder is designed which is a heterogeneous multiprocessor system; two hardwired logic blocks and one specialized DSP processor.

Key Words : MPEG-2 audio encoder, top-down approach, partitioning,, heterogeneous multi-processor system.

1. Introduction

The MPEG audio coding algorithm has contributed on wide popularity of digital audio applications. Among them, such applications as teleconferencing, DCC, and DVCR require real-time audio encoders. Although audio encoders are necessities in these cases, work concerning real-time implementation of them has rarely been reported because of the heavy computational load and the inherent complexity of the algorithm.

The multichannel MPEG audio coding algorithm is mainly composed of transformation which is computation-intensive and quatization which is controlled by decision-intensive bit-allocation rules. When implementing such a heterogeneous and heavy-weighted algorithm using general purpose processors, the resulting system must employ multiple processors or expensive general purpose DSP. However, it is not an efficient solution in terms of hardware utilization and cost because the general purpose processors are not always the best solutions of every assigned task. To have an efficient and cost effective audio encoder, ASIC design technique should be introduced.

ASIC design techniques can be divided into three patterns: 1) pure hardware design, 2) hardware/software codesign using embedded general purpose processors, 3) hardware/software codesign using ASIP's (Application Specific Instruction-set Processors). The pure hardware approach is appropriate for designing small scale systems. The lack of programmability encountered in using only hardwired logics can be overcome by introducing embedded processor cores. Also, to reduce redundancy occurring in using general purpose processor cores while preserving programmability, the processor core introduced must be an ASIP. Since the MPEG audio coding algorithm shows heterogeneous characteristics and requires high computational power, it was decided to adopt the strategy of hardware/software codesign to implement an MPEG multichannel audio encoder. Section II describes the profile of the algorithm obtained by performing the fixed-point analysis of the MPEG-2 audio encoder. Based on the profile, partitioning and scheduling were performed as an attempt to minimize size and to maximally utilize data concurrency. A procedure to satisfy the deadline constraint is presented in Section III. The goal of satisfying the constraint was achieved by exploiting algorithmic characteristics and devising special components.

Manuscript received Sep. 13, 2007; revised Mar. 4, 2008.

II. Algorithm Analysis

A. MPEG-2 Audio Encoding Algorithm

MPEG audio encoding algorithm consists of three layers, and each layer is the result of a trade-off between performance and complexity00. Even though all three layers have the same basic structure, layer II is generally accepted to be adequate for a high quality audio coding application and to have moderate complexity. **Fig 1** shows the MPEG-2 multichannel audio layer II encoding procedure in dataflow semantics.

When time-domain input samples enter the system, analysis filterbank (AF), which is composed of 32 equi-bands, transforms them into subband samples. The matrixing module downmixes the subband samples and generates backward compatible channels (Lo and Ro). The scale factor coding (scf coding) module extracts a scale factor which has a maximum of 12 absolute sample values, and it encodes every three scale factors with scale factor select information. Scale factor select information contains the information in which one/two scale factor(s) is/are omitted in the bitstream, so the decoder should

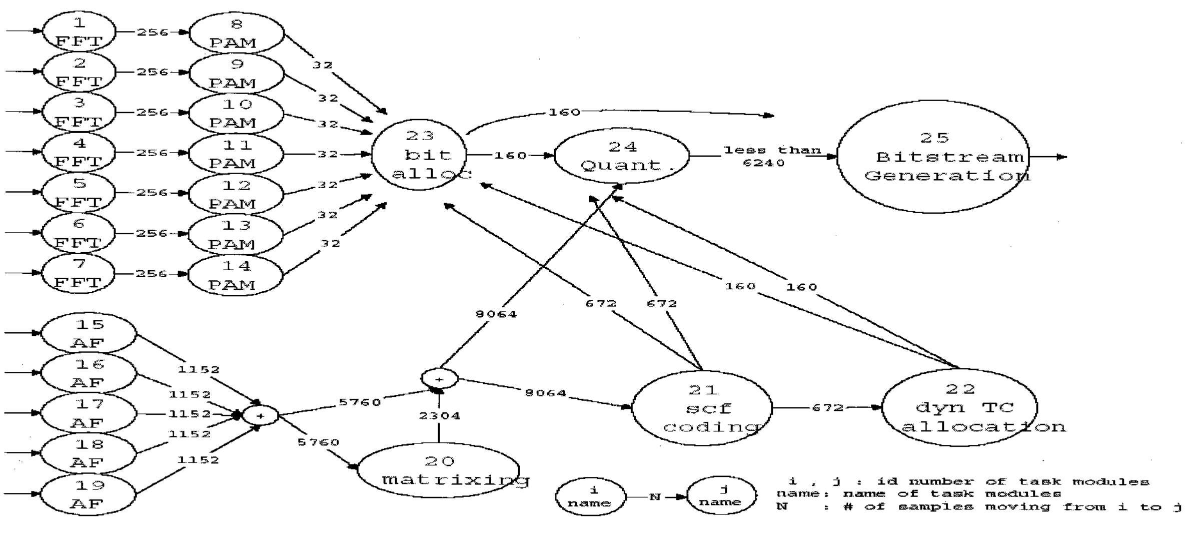

Fig. 1. MPEG-2 Multichannel Audio Layer II Encoding Procedure in Dataflow Semantics

copy them from elsewhere. The determined scale factors are also used to establish the strategy for the dynamic transmission channel switching (dyn TC allocation). Then the transmission channel samples are finally quantized and packed into a bitstream. Quantization process is controlled by the bit-allocation, which decides the step size of the quantizer based on the bitrate and the SMR (Signal-to-Mask Ratio). The SMR is calculated at the PAM (PsychoAcoustic Model) module.

B. System Level Partitioning and Scheduling of the Encoding Algorithm

The MPEG-2 multichannel audio algorithm was simulated with C language considering fixed-wordlength effect. The algorithm is decomposed into sub-algorithms (tasks) and their profiles are summarized in **Table I**. When counting the number of operations in the table, we assume that a hypothetic processor with a RISC-type instruction set runs the MPEG-2 multichannel audio algorithm. The hypothetic processor has an operational unit composed of an ALU and a hardware multiplier. It has enough number of registers for computation and memory access, that is, there are always empty registers when storing interim values after computation, and there are no contention to get empty address registers and buffer registers. And it takes one cycle for memory read or write.

When the MPEG-2 multichannel audio layer II encoder runs at the system clock of 27 MHz (which is determined to synchronize the multiplexer0 and video encoder0 of MPEG-2), and supports the sampling frequency of 48 kHz (which is the highest sampling frequency recommended by MPEG group), all the operations should be completed within 562.5 cycles per sample (= 1/48k * 27M). As the number of samples of a frame for the case of layer II is 1,152, the time-limitation is 648,000 cycles (= 562.5 * 1152). This deadline constraint implies that more than 3 concurrently operating processors are required to guarantee real-time operation of the encoder. (Assuming 4 processors can equally share the whole workloads of 2,300K cyc, each processor will have 575K cyc, then real-time operation will be achieved.)

As the first step of design, task-level partitioning is undertaken. The goals of task-level partitioning are to achieve the smallest area and the lowest communication cost. To have a small-sized audio encoder, the similarity is used as a measure for partitioning. Task modules requring similar operations are clustered into a module in which the task modules are binded to a common datapath which can cope with all the clustered tasks. This design-reuse results in resource-saving.

Table I. Number of operations for MPEG-2 multichannel audio layer II encoding algorithm

Task	# of op. (unit: K cyc)	Ratio (%)	Frequenctly called functions & its frequency per frame
5ch AF	600	26.1	
7ch FFT PAM step1)	360	15.7	
7ch PAM step2~8	840	36.5	10log10 x : 9097 10^(x/10) : 6889
Matrixing	40	1.7	
Scf coding	65	2.8	
Dynamic channel allocation	5	0.2	
Bit allocation	180	7.8	Bit_alloc(): 278
Quantization	130	5.7	
Bitstream generation	80	3.5	Packing(): 1438
total	2,300	100.0	

Further, we also want to use all available cycles only for computation. As the communication time between processors occupies much portion of the total execution time in a multiprocessor system, the communication cost is used as the second measure. Task modules with many samples transferring between tasks are clustered to reduce the communication cost. In other words, binding tightly coupled tasks to a common datapath increases available cycles for computation since a job of binding tasks are generally expected to reduce the communication cost. To achieve these goals we partitioned the system based on the similarity between tasks and the communication cost, respectively, and we found a solution satisfying both measures.

According to the similarity measure, the audio encoder is divided into three modules. As the encoder is aimed to process 5 channels, FFT is called seven times(five input channels and two matrixed channels), and analysis filtering is called five times for encoding a frame. FFT' of seven channels (tasks 1~7) are assigned to module-1, PAM of seven channels (task 8~14) are assigned to module-2, analysis filtering of five channels (tasks 15~19) are assigned to module-3, and remaining tasks unique (tasks 20~25) are assigned to module-4. The numbers in parenthesis above represent the identification numbers of the task modules in **Fig 1**.

Next, communication cost is considered. We try to merge the modules sharing many common inputs and having many samples transferring between them to reduce the communication cost. For this we introduce a closeness function as

$$\text{closeness}(Vi; Vj) = \text{inputs}(Vi; Vj) + \text{wires}(Vi; Vj) \qquad (1)$$

where Vi is the i-th group of tasks, inputs(Vi; Vj) is the number of common inputs shared by Vi and Vj, and wires(Vi; Vj) is the number of connections between Vi and Vj . The clustering is performed by repeatedly merging modules having high closeness and the result is summarized in **TABLE II**.

TABLE II. The result of clustering

# of iterati on	Clusters (with more than 2 elementes)	# of clusters	# of samples transferring between modules
1	(21,24)	24	24,064
2	(20,21,24)	23	16,000
3	(20,21,24,25)	22	9,760
4	(15,16~,20,21,24,25), (22,23)	16	2,848
5	(15,16,~,25)	15	2,016
6	(1,2~,14),(15,16~,25)	2	224

TABLE II indicates that the previous partitioning, 4 modules based on the similarity measure, could increase the entire communication cost. The sixth iteration of **TABLE II** suggests to put tasks 1~7 (module-1) and tasks 8~14 (module-2) together, and to put tasks 15~19 (module-3) and tasks 20~25 (module-4) together, which can reduce the communication cost. This reading generates a trade-off issue between hardware reduction (by merging similar modules) and communication cost reduction (by merging modules with high closeness).

This trade-off can resolved with a closer look at the encoding procedure in **Fig 1**, it is found that the subband samples produced by tasks 15~19 should be ready together for the following tasks and the samples shall be retained till the end of the encoding process, in order to support quantization and bitstream generation. This means that a memory which can hold 8,064 samples (=1,152 * 7ch) shall be equipped in audio encoder system. Among tasks operating on subband samples, module-4 (tasks 20~25) can start only after the completion of module-3 (tasks 15~19). Thus, module-3 generates subband samples and writes them to a memory without being interrupted by other modules. After module-3 completes its tasks, module-4 reads the subband samples from the memory without interruption.

Since the algorithm requires storing the whole result of subband filtering, the amount of communcation between module-3 and module-4, i.e. storing subband samples to memory, cannot be reduced even after merging the two modules. Furhter considering the low complexity in communication between them (it happens in one-way and static way) it is not necessary to merge the modules against the partitioning result based on similarity.

Regarding the communication between module-1 and module-2, since tasks 1~7 are independent of each other and are related only to their successors, it is possible to pipeline the tasks of module-1 with those of module-2. In this pipelining, a memory bank with the capacity of 512 samples effectively realizes the communication between the modules. Thus the it also not necessary to merge the modules against the partitioning result based on similarity.

It's also found that there are some modules which shall operate in tandem mode(Module-1,-2,-4 and Module-3,-4, respectively), and others which can work in parallel mode (Module-1 & Module-3, or Module-2 & Module-3). Considering maximum parallelism, the module-2 (PAM, task8~14) and module-4 (unique operations, task 20~25) are merged.

The resultant modules and scheduling are shown in **Fig. 2**. Thanks to the communication scheme using memories, similarity is still reflected in the partitioning, which leads to area reduction. The schedule is also intended to make each module operate concurrently as much as possible, which minimizes the execution time of the system. In other words, while FFT Module (module-1) and DSP Module (module-2 & -4) work in a pipelined manner, AF Module (module-3) also works in parallel. That is the maximum utilization of the data parallelism of the given algorithm.

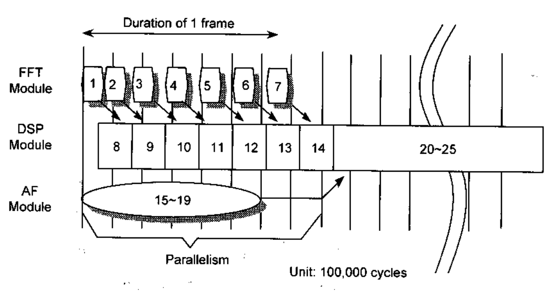

Fig. 2. Task Schedule of the Encoder

III. Cycle reduction procedures and design of accelerating components

After task-level partitioning and scheduling, the next design step is taking into account the number of operations of each task. The estimated number of operations at the system-level can be reduced by modifying the algorithm or by adopting specially-designed hardware in the course of design process..

A. Analysis FilterBank and FFT

According to **Table I**, AF Module (analysis filterbank) consumes about 600,000 cycles. Therefore, there must be a way of reducing the required number of cycles in mathematical or architectural ways. As a mathematical approach, we adopted the algorithm proposed by Konstantinos 0, who exploited the characteristics of the transform matrix, and changed the matrixing operation into a discrete cosine transform (DCT). He showed that by using Lee's fast DCT algorithm 0, the original 2,048 multiply-accumulate operations are reduced to 80 multiplications and 209 additions.

As an architectural approach, the filterbank is designed in hardware, which is an effcient solution in terms of space, because the analysis filterbank consumes and produces a fixed number of samples, and the procedure is quite regular and iterative. Controlled by FSM(Finite State Machine), the analysis filter transforms input samples of five channels in 350,000 cycles using two multiplier-and-accumulators.

For the design of FFT Module, a similar rationale is applied. The FSM controlled module-1 finishes 1,024-point FFT within 45,000 cycles using a multiplier, an adder, a subtractor, and two adder/subtractor.

B. PsychoAcoustic Model (PAM)

According to **Table I**, PAM (step 1~8) consumes about 840,000 cycles, which exceeds 640,000 cycles. Such a huge number of cycles is due to the inherently high computational complexity and the decreased regularity of PAM when compared with the tasks assigned to FFT Module and AF module.

Considering two conflicting design directions, high complexity and low regularity, a compromise was made using ASIP(Application Specific Instruction-set Processor)0. Since PAM is a recommendation changeable, the ASIP approach is more attractive.

Before discussing the optimization of the PAM calculation, computational requirements of PAM are analyzed in detail through fixed-point simulation. **TABLE III** shows that step 3~8 take more cycles than step 1. This means that in the scheduling framework of FFT, AF and DSP Module, mentioned in Section II-B, DSP Module consumes more cycles than FFT Module (for example compare task 2 with task 8 in **Fig. 2**). This imbalance in the workload leaves FFT Module idle in the pipelined process, and is one of major reasons for prolonging the overall execution time for a frame.

TABLE III. Analysis of PsychoAcoustic Model 1

Task	Description	# of cycles required	Ratio (%)
Step 1	Calc. of FFT w/ the proposed hardware	45,000	30
Step 2	Determination of SPL (this is done in bit-allocation task)	0	0
Step 3	Determination of LTg	15,000	10
Step 4	Finding tonal & non-tonal components		
Step 5	Determination of maskers		
Step 6	Calculation of LTtm, LTnm	90,000	60
Step 7	Determination of LTg, eq (4)		
Step 8	Determination of LTmin		

B.1 Calculation of 10 log10 x, and 10^(x/10)

In step 3, 4, 6, and 7 of PAM, numerical computation is performed frequently. The equations involved are 0 :

$$X(k) = 10\log_{10} FFT(x) \tag{2}$$

$$X_{tm}(k) = 10\log_{10}(10^{x(k-1)/10} + 10^{x(k)/10} + 10^{x(k+1)/10}) \tag{3}$$

$$LT_g(i) = 10\log_{10}(10^{\frac{LTq(i)}{10}} + \sum_{j=1} 10^{\frac{LTtm(z(j),z(i))}{10}} + \sum_{j=1} 10^{\frac{LTnm(z(j),z(i))}{10}}) \tag{4}$$

where X(k) is the sound pressure level of the spectral line with index k of the FFT, Xtm(k) is the sound pressure level of tonal component, LTg(i) is the global masking threshold at the i'th frequency sample, LTq(i) is the threshold in quiet, LTtm(z(j); z(i))/LTnm(z(j); z(i)) is the individual masking threshold of tonal/non-tonal component at critical bandrate z(i) in Bark of the masking component at the critical bandrate of z(j) in Bark, z(i) is the corresponding critical bandrate of index i of the FFT. From (2)~(4), we can see that the key operations involved are :

$$y = 10\log_{10} x \quad (5)$$

$$y = 10^{x/10} \quad (6)$$

Thus, reducing the number of cycles consumed by (5) and (6) is the first job. These transcendental functions can be implemented by various forms 0. Among them, a look-up table(LUT) search approach was taken to complete the operation within one cycle.

Entering to design the architecture of an accelerator, it should be decided the numerical formats representing the power spectrum in linear scale and in logarithmic scale, respectively, which will be manipulated during PAM calculation. In MPEG-2, the maximum value of the signal power spectrum corresponds to 96 dB, which means the spectrum can be represented by unsigned numbers with 32 bits of wordlength. The corresponding values of the spectrum in logarithmic scale can also be represented by a signed number with 8 bits for each integer part, and 8 bits for each fractional part. For fractional part, 8 bits are enough to represent the given absolute threshold without the loss of information. (Hereafter expression of "x.y" format will be used to describe how many bits are used for integer part and fractional part each in fixed-point calculation. "x" means x bits for integer part are used and "y" means y bits for fractional part are used.)

The input x of (5) can be expressed like a floating-point number, using mantissa and exponent with the base of 2 as follows:

$$x = mantissa \times 2^{exponent} \quad (7)$$

where mantissa has 0.32 format and exponent has 5.0 format. Applying 10 log10 to x of (7), the final formula is as follows.

$$y = 10\log_{10} x = 10\log_{10}(mantissa) + exponent \times 10\log_{10} 2 \quad (8)$$

$$\approx manti\_table + expo\_table \quad (9)$$

The second and third terms of (8) are implemented in tables, where mantissa and exponent play a role as addresses of the tables. Thus the transformation of values from linear to logarithmic scale is realized by adding each indexed value in a single cycle.

The procedure of designing an accelerator of (6) is similar to that of (5). (6) can have another form with its base of 2, and the exponent term can be partitioned into its integer part and its fractional part as follow.

$$y = 10^{x/10} = 2^{\frac{x}{10}\log_2 10} \quad (10)$$

$$\tfrac{x}{10}\log_2 10 = I + f \quad (11)$$

The input x in (10) is represented by 8.8 format because its value is represented in logarithmic scale. Since (11) can be thought of as the multiplication of two values in logarithmic scale, x and 1/10 log2 (10), the exponent term has 16.16 format.

In terms of implementation, the accelerator needs a scaler (shifter) for 2^I and a table for 2^f (pow_table) as follows.

$$y = 10^{x/10} = 2^I \times 2^f \;=\; pow\_table << I \quad (12)$$

For the scaler, a 31-bit shifter suffices, since the value of x is less than 96 dB, and the actual value of I does not exceed 31.

The accelerators of (5) and (6) are merged into a log_pow unit. The log_pow unit is composed of a lead-one detector for the detection of the exponent; manti_table, expo_table, and an adder for the log calculation; and a multiplier, pow table, and scaler for the power calculation. Among them, as an adder, a multiplier, and a scaler are commonly used by signal-processing processors; the components which should be newly-introduced are only a lead-one detector and tables.

B.2 Calculation of the Individual/Global Masking Threshold

Step 6 is to calculate tonal or non-tonal masking thresholds which formulate the human auditory system. The formulas are

$$LT_{tm}[z(j);z(i)] = X_{tm}[z(j)] + av_{tm}[z(j)] + vf[z(j);z(i)] \quad (13)$$

$$LT_{nm}[z(j);z(i)] = X_{nm}[z(j)] + av_{nm}[z(j)] + vf[z(j);z(i)] \quad (14)$$

where X represents sound pressure level, av masking index, vf masking function, and z(j) a corresponding critical bandrate of index j. In these equations, when calculating a masking threshold at j, the sound pressure level and masking index are considered as constants. Thus, only the masking function affects masking threshold at i. Since the masking function has piece-wise linear characteristics, the masking threshold is generally expressed as

$$LT[z(j);z(i)] = a \times dz + b \quad (16)$$

where a and b are piece-wise constants. (16) can be evaluated in a cycle using MAC (Multiply-And-aCcumulate) operations.

As a mean of accelerating step 6 and step 7, the two steps are merged, which removes data move (memory) operations for temporal saving. In other words, the results of step 6, LTtm and LTnm are converted into linear scale values and accumulated with the threshold in quiet (LTq) with registers only.

B.3 Effect of the Modification

By using a ASIP-type DSP with an MAC unit and a single cycle log pow unit for (5) and (6) , and by applying recursive computation for the calculation of the individual/global masking threshold, the system can greatly reduce cycles for operations with an acceptable loss of accuracy. **TABLE IV** shows the reduced computational load. As the cycles required for FFT do not change, **TABLE IV** shows that the cycles for step 3 ~ step 8 are reduced to about half of the original. Furthermore, as the cycles consumed by FFT Module and DSP Module at Section II-B are similar, it promises efficient pipeline operation between them with minimized idling, waiting, and communication times.

TABLE IV. PsychoAcoustic Model 1 with new scheme

Task	Description	# of cycles required	Ratio (%)
Step 1	Calc. of FFT w/ the proposed hardware	45,000	53
Step 3	Determination of LTg	15,000	17
Step 4	Finding tonal & non-tonal components		
Step 5	Determination of maskers		
Step 6	Calculation of LTtm, LTnm	28,000	30
Step 7	Determination of LTg, eq (4)		
Step 8	Determination of LTmin		

IV. Result

The designed MPEG-2 audio encoder is composed of three modules. They are configured and scheduled to maximally utilize inherent concurrency of the algorithm. The configuration and the schedule of the encoder are shown in **Fig. 3** and **Fig. 4**, respectively.

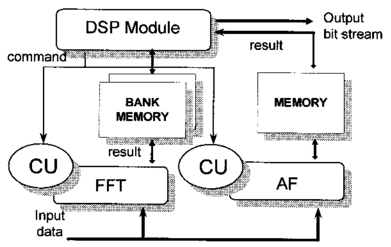

Fig. 3. Configuration of the Audio Encoder

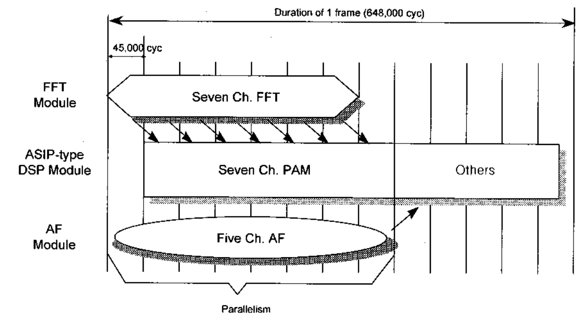

Fig. 4. Refined Schedule of the Audio Encoder

The modules in **Fig. 3** constitute the audio encoder and are tuned to accelerate the assigned tasks by introducing a new algorithm (AF Module, DSP Module), and designing special hardware (DSP Module).

The DSP processor controls FFT and AF modules as a master. This configuration permits users to control the whole behavior of the audio encoder by changing the program of the DSP Module.

For efficient communication between modules, the audio encoder system has shared memory architecture, with two separated memory blocks. With the separate memory blocks, the DSP processor can receive the spectrum from the FFT module by bank-switching with negligible waiting time, since the two modules work in a pipelined fashion and their running times are almost the same as shown in **Fig. 4**. While the AF module transforms input data into subband samples, the DSP Module is scheduled not to use any subband samples.

As the processing time of the AF module is shorter than that of the DSP Module, the output of the AF module stored at a shared memory can be directly used by the DSP Module without waiting.

As a result, the audio encoder encodes five-channel audio input data within the deadline constraint. The timing information of the resulting system is shown in **TABLE V**, from which we can see that if the audio encoder had not exploited pipelining, it would need 875,000 cycles to complete all the tasks, which violates the deadline constraint. In this case, another effort should be devoted to satisfy the constraint such as introducing more operational modules. However, the designed audio encoder makes use of parallelism which is inherent from specification, and needs only 568,000 cycles. This leaves room for higher bitrates.

TABLE V. Timing analysis of finally designed audio encoder

Task	# of op. (unit: 1000 cyc)	Ratio (%)	Frequenctly called functions & its frequency per frame
1ch FFT	1	45	358
1ch PAM	3	43	=315+43
7ch FFT	2	315	
5ch AF	1	350	
Matrixing	3	30	210
Scf coding	3	40	
Bit allocation	3	50	
Quantization	3	60	
Bitstream generation	3	30	
total	2300	100.0	568

V. Conclusion

This paper presents the design of the MPEG-2 5 multichannel audio encoder based on hardware/software codesign strategy. We have partitioned the MPEG-2 audio encoding algorithm into three modules in the way of maximizing design reuse. Each module has been mapped into a proper hardware block that implements the assigned function efficiently. To determine proper architectures, the inherent characteristics of each module are considered. The FFT module and the analysis filter module which show high regularity are implemented as hardwired logic blocks, and the other tasks which show irregularity in their control-flow and require high computational power are merged and implemented in the form of an ASIP. The three modules are scheduled to maximally utilize the inherent data parallelism in the form of pipelining and parallel processing. To minimize both area and communication cost which seems to be contradictory, shared memory architecture is incorporated.

To reduce the cycles consumed by each module, the analysis filterbank takes an algorithm different from the recommended one. For further acceleration, frequently called functions are realized as application-specific hardware units. They are designed to complete their functions in a single cycle, and are merged as an application-specific datapath of a DSP Module.

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(Part3. AUDIO)", ISO/IEC JTC1/SC29/WG11.

[2] ISO/IEC 13818-3, "Generic Coding of Moving Pictures and Associated Audio -AUDIO," ISO/IEC JTC1/SC29/WG11, Nov. 1994.

[3] ISO/IEC 13818-1, "Generic Coding of Moving Pictures and Associated Audio - SYSTEMS," ISO/IEC JTC1/SC29 /WG11, Nov. 1994.

[4] ISO/IEC 13818-2, "Generic Coding of Moving Pictures and Associated Audio Information - VIDEO," ISO/IEC JTC1 /SC29/WG11, Nov. 1994.

[5] Konstantinos Konstantinides, "Fast Subband Filtering in MPEG Audio Coding," *IEEE Signal Processing Letters*, vol. 1, no. 2, pp. 26-28, Feb. 1994.

[6] P. Paulin, G. Goossens, C. Liem, M. Cornero, and F. Nacabal, "Embeded software in real-time signal processing systems : Applications and architecture trends," *Proc. of the IEEE*, Vol. 85, No. 3, pp. 419-435, March 1997.

[7] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, no. 6, pp. 1243-1245, Dec. 1984.

[8] Israel Koren, *Computer Arithmetic Algorithms*, Prentice-Hall, pp. 163-189, 1993.

Sung-Wook Park received the B.S., M.S., and Ph.D. degree in Electronic Engineering from Yonsei University in 1993, 1995, and 1998, respectively. He is now working for Samsung Electronics. His research interest includes VLSI signal processing and Multimedia Signal Processing.