MPEG1 Audio Decoder With System Decoder Running in MSP Under MMOSA

Kicheon Hong*

1. Introduction

Standardization of recording media, device and various aspects of data handling, such as audio recording, storage and playback, is highly desirable for continued growth of this technology and its applications. One compression standard which has attained wide spread use for compressing and decompressing video information is the moving pictures expert group (MPEG) standard for audio and video encoding and decoding. The MPEG standard[1] is defined in International Standard ISO/IEC 11172-1, "Information Technology-Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s", Parts 1, 2 and 3.

The MPEG International Standard addresses the problem of combining one or more data streams from video and audio sources with timing information to form a single information stream. Typically massive volumes of information stream data are handled by an MPEG processing system. Although the computing power of processors has advanced steadily and rapidly, the demand for higher processing power and computing efficiency remains unabated due to the development of aggressive new applications in the multimedia field that call for the display and performance of over larger data quantities.

* Dept. of Information and Telecommunications, The University of Suwon

2. Multimedia Processor System Structure

Operations of multimedia applications including MPEG1, MPEG2, video conference, fax support, modem support and the like are efficiently performed using a multiple processor system. One example of a multiprocessor system includes a scalar processor and a vector processor. Audio decoding is an operation that is accelerated through the usage of a multiprocessor system and design of operating procedures to efficiently allocate steps in the decoding process between the scalar processor and the vector processor.

Referring to Fig. 1 a high-level schematic block diagram illustrates a multimedia multiprocessor system including a host processor and a multimedia signal processor. A typical host

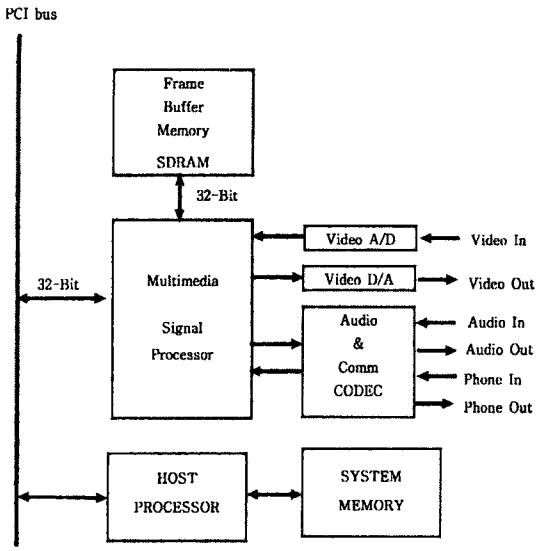

Fig. 1 Multimedia Processor System

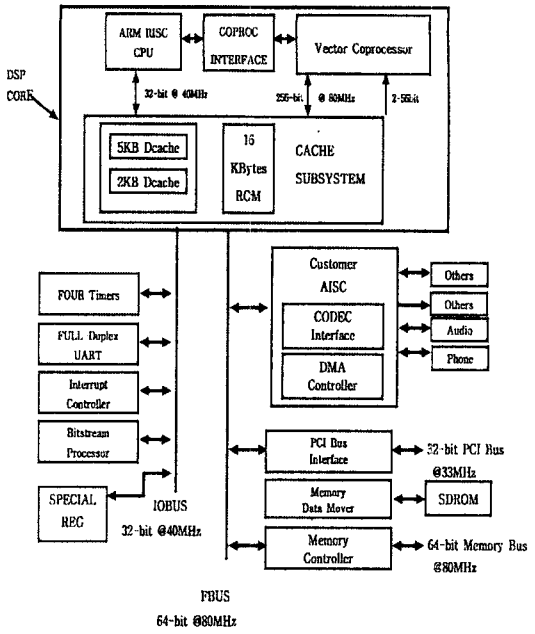

Fig. 2 Multimedia Signal Processor

processor is an Pentium processor. The host processor executes programs based on instructions and data held in a system memory. The host processor communicates with the multimedia signal processor via s system bus such as a PCI bus[3]. The multimedia signal processor(MSP)[4] interfaces to various functional blocks such as an audio and communication CODEC, a video A/D converter, a video D/A converter, and a frame buffer SDRAM memory. Referring to Fig. 2, a schematic block diagram shows the multimedia signal processor within the multimedia multiprocessor system. The multimedia signal processor includes signal processor(DSP) core which is connected to a plurarity of multimedia interfaces. The DSP core is the computation of the multimedia signal processor and includes a scalar processor, a vector processor, cache subsystem, a fast bus(FBUS), and an I/O bus. The scalar processor is a processor based on a 32-bit ARM7[5] control processor which is designed and manufactured by ARM Limited, Great Britain, and performs general processing function such as real-time operating system operations, interrupt and execution handling, input/output device management, communication with the host processor and the like. The multimedia signal processor performs multiple various multimedia operations. One multimedia operation is audio decoding so that the scalar processor and the vector processor, in combination with program codes which operate on the processor, form an audio processor.

3. MPEG1 Audio Decoder Implementation on MSP

The MPEG International Standard defines a

suitable audio decoding process. The audio decoding process includes the steps of inputting an encoded bit stream, decoding bit allocation bits, decoding scale factor selection information, decoding scale factor, requantization of sample data, synthesis of a subband filter and outputting of PCM[2] samples.

3-1. Requantization Operation Analysis

The requantizing sample data operation is one of the most computationally intensive operations performed by the MPEG audio decoder. One technique for performing partial requantization of sample data is illustrated by the following "C" language program code. In the illustrative program code, sb designates a subband, bound is the lowest sub-band which uses intensity stereo coding, sblimit is the lowest sub-band which uses stereo coding (see MPEG International Standard, PartⅢ, Table B.2a-Table B.2d),nb is the number of bits assigned to each sub-band, and scalefactor[ch][sb][z]is a scalefactor coefficients index(see MPEG International Standard, PartⅢ, Table B.1). The 36 samples in one subband within a frame are divided into three equal parts of 12 subband samples. Each part may have a separate scalefactor. Therefore, three scalefactor exist having an index(Z)is either 0, 1,or 2, depending on the iteration number of a group, gr. The number Z is defined so that Z is zero if gr is less than 4. Z is one if gr ranges from 4 to less than 8. Otherwise Z is two. C2[nb] and[nb] designate layerⅡ classes of quantization coefficients C and D(see MPEG International Standard, PartⅢ, Table B.4), sample[s][ch][sb] is an array of degrouped audio sample data, and desample[s][ch][sb] is an array of requantized audio sample data. The following program code includes a **[NOTE]** marking which is indicative of program codes that are modified in a subsequently discussed embodiment of a requantization operation. The program code for a partial requantization operation of an audio decoding operation of is set forth in TABLE I , as follows:

The audio decoder program code illustrated in TABLE I performs a partial requantization calculation for data in groups of three samples. The three samples in a group are included in the same sub-band of 32 sub-bands. The audio sample requantization operation of the audio

TABLE I

```
for(sb=0;sb<bound;sb++) {
  for(ch=0;ch<stereo;ch++) {
    if(bit_alloc[ch][sb]) {
      nb=bit_alloc[ch][sb];
      factor=sf[scalefactor[Z][ch][sb];
      C=C2[nb];                          [NOTE]
      D=D2[nb];
      for(s=0;s<3;s++) {
      if(((sample[s][ch][sb]>>nb-1)&1)==1)
          fraction=0.0;
          else
          fraction=1.0;
      }
```

decoder program code depicted in TABLE I is performed in the scalar processor since the requantization operation includes index operations are conditional that are efficiently performed by a scalar-type processor. Furthermore, the layer II classes of quantization coefficients C and D are determined dependent on the number of bits assigned to each sub-band(the parameter nb) and are therefore not available as a sequential data.

3-2. Vectorization of Requantization Operation

Most requantization operations may be advantageously performed on the vector processor upon appropriate modification of various instructions of audio decoder program code to gain additional improvements in audio decoding. These modifications generally transform data in a non-sequential form into a sequential format data in a vectorization process.

One modification is entered at the position of **[NOTE]** in the program code illustrated in TABLE I . The expression, factor=sf[scalefactor [Z][ch][sb]], is indicative of a scalefactor index in which Z is the number of a group, ch is the channel number, and so indicates the sub-band number. In a vectorized embodiment of the audio decoding operation, the scalefactor index is extracted from the bitstream in the step of extracting scalefactor selection information and the scalefactor 618. The plurality of scalefactor coefficients is directly calculated based on the scalefactor index and placed into the array scalefactor [Z][ch][sb] during the step of extracting scalefactor selection information and the scalefactor. TABLE II illustrates a program code for inclusion in the step of extracting scalefactor selection information and the scalefactor, which is performed by the scalar processor prior to the requantization step. where map is the scalefactor index from the bitstream, BITMAP is 6-bit designation of a bit stream file, and sf[] is a 63-element scalefactor coefficients array. Performance of the instructions in TABLE II during the step of extracting scalefactor selection information and the scalefactor transforms the data in the array scalefactor [Z][ch][sb] into sequential data which is dependent upon sub-band number rather than on bit index operation.

Another modification is entered at the position of **[NOTE]** in the program code illustrated in TABLE I . The expressions C=C2 [nb] and D=D2[nb], designate quantization coefficient that are calculated at every iteration but are common to each sub-band. TABLE II B illustrates a program code for inclusion in the step of extracting scalefactor selection information and the scalefactor 618, which is performed by the scalar processor prior to the requantization step. The program code shown in TABLE II B is performed only one time during each one frame reconstruction. The data

TABLE II

```
map=BITMAP;
scalefactor[Z][ch][sb]=sf[map];
```

in each array becomes exclusively sequential data which corresponds to the index of the channel and the sub-band number. The data is no longer related to the index of the bit allocation after the program code shown in TABLE II B has completed execution. where Q[ch][sb] is a mask function and C[ch][sb] and D[ch][sb] are arrays holding requantization coefficients data.

3-3. Job Allocation between ARM and VP for MPEG1 Audio Decoder

An efficient operation of MPEG decoding

TABLE II B

```
for(ch=0;ch<stereo;ch++)
  for(sb=0;sb<min(bound,sblimit);sb++)
  if(nb=bit_alloc[ch][sb]) {
    Q[ch][sb]=1L<<nb-1; C[ch][sb]=C2[nb];
    D[ch][sb]=D2[nb];
  } else {
    Q[ch][sb]=0; C[ch][sb]=0; D[ch][sb]=0;
    }
  for(sb=bound;sb<sblimit;sb++) {
    if(nb=bit_alloc[0][sb]) {
    Q[0][sb]=1L<<nb-1;Q[1][sb]=Q[0][sb];C[0][sb]=C2[nb];
    C[1][sb]=C[0][sb]; D[0][sb]=D2[nb]; D[1][sb]=D[0][sb];
  } else {
    Q[0][sb]=0; Q[1][sb]=0; [0][sb]=0; C[1][sb]=0;
    D[0][sb]=0; D[1][sb]=0;
  }
}
for(sb=sblimit;sb<32;sb++) {
 Q[0][sb]=0; Q[1][sb]=0; [0][sb]=0; C[1][sb]=0;
 D[0][sb]=0; D[1][sb]=0;
}
```

operations is advantageously performed by sharing audio decoding operations between the scalar processor and the vector processor to exploit the advantages of each processor type. In MPEG1 audio decoding, the scalar processor is assigned to perform operations involving bit manipulation, indexing and conditional operations of MPEG audio decoder including operations of frame synchronization, and extraction of information from the MPEG signal stream, including extraction of header information, extraction of bit allocation bits, extraction of scalefactor selection information and a scalefactor, and extraction of sample data.

In accordance with one aspect of the present MSP chip, the MPEG1 audio decoder will be performed through multiple parallel processors including a scalar processor(ARM7) and a vector processor(VP). The scalar processor most efficiently performs tasks including bit manipulation, indexing and conditional operations. The vector processor most efficiently performs operations involving multiple data calculation, operational on unconditional operations, sequential data. Based on each performance of processor, the job allocation between ARM7 and VP can be done for the operation of MPEG1 audio decoder on MSP. From the job allocation, ARM7 performs the audio pre-processing such as Frame synchronization, bitstream parsing, extracting of frame header information, extraction of frame bit allocation, scalefactor selection information, scalefactor and audio sample data information. The Vector Processor performs the computationally intensive post-processing such

as requantization, matrixing, filtering, windowing and merging of two channels into one channel for stereo.

3-4. MPEG1 Audio Decoder Data Path on MSP

Referring to Fig 3, the schematic diagram illustrates a data path of the MPEG1 audio decoder on MSP chip. Each paths description is as follows:

Path #1: As of request of system decoder to host, the MPEG1 multiplexed data passed to system decoder through PCI bus will be demultiplexed into audio packet and video packet data. The audio packet will be stored into SDRAM.

Path #2: The ARM7 will process the audio packet data to extract all header information.

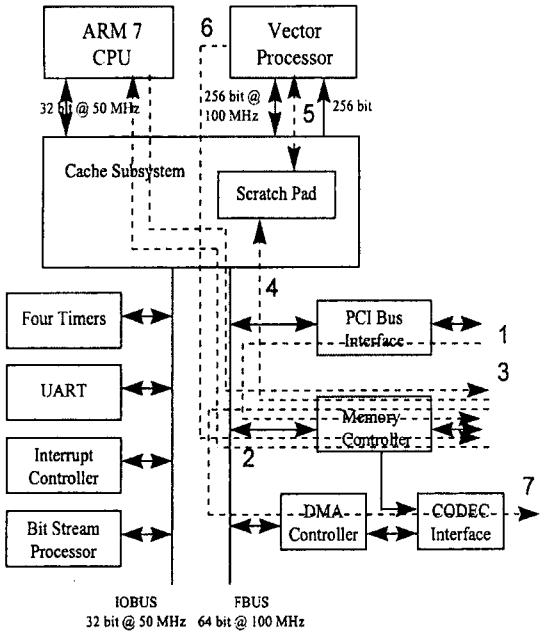

Fig. 3 MPEG1 Audio Decoder Data Path on MSP

Path #3: All extracted audio sample information will be moved into SDRAM by ARM7.

Path #4: Requantization coefficients and filter coefficients used for synthesis filter operation will be fetched into Scratch Pad.

Path #5: VP will requantize the extracted audio sample data and do synthesis filter operation.

Path #6: The fully reconstructed audio PCM data will be put into SDRAM.

Path #7: The PCM data in SDRAM will be moved through DMA controller to codec interface.

4. MSP Multi-Media Operating System Architecture(MMOSA)

The MSP is a processor device from Samsung that contains both a 32-bit RISC processor and a DSP vector processor. The RISC processor is an ARM7 processor core. The DSP is a vector coprocessor(VP). The MMOSA[6,7] is a new embedded operating system with a combination of speed, power, and simplicity that will enable software developers and hardware engineers to create a wide array of consumer products.

MMOSA has a compact high-speed design that is small enough to run tiny hand-held computers and yet it's also fast and powerful enough to run huge interactive television systems. One of MMOSA's most unique features is that is can run on many different processors, platforms, and networks using the advanced distributed object-oriented style of programming.

5. Integrated MPEG1 Audio Decoder with System Decoder Under MMOSA

The MPEG1 System decoder demultiplexes the video and audio data and feeds them into the video and audio decoder respectively. The multiplexed data is assumed to be sent by the host processor and this functionality is mimicked by the system decoder implementation. In this implementation the host and the system decoder running on ARM7 are two different threads under MMOSA. The control signals, and data are transferred between these two threads through sockets shown in Fig. 4. Sockets S1 and S1′ are meant for transferring control signals, and sockets S2-S2′ and S3-S3′ are for transferring video and audio data respectively. The control data is just 4 bytes long, and specifies information to the host about the type of data to be sent back, and the length of the data sent. The actual data however is transferred via sockets S2-S2′ and S3-S3′. The audio decoder and video decoder run as two different application threads within the ARM7 thread.

The data transferred from the host to the system decorder resides on the SDRAM, and

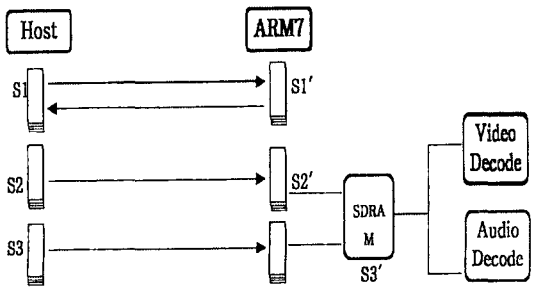

Fig. 4 System decoder & Sockets

can be used by the audio decoder thread to process. The sockets are basically code structures and under these are defined the buffer lists. When the data transfer between the host and the system decoder occurs the data is stored in these buffers. There are functions defined within MMOSA for the management, allocation, and deallocation of these buffers. Once the data is transferred to SDRAM, the audio decoder thread makes use of the buffer list pointers to access the audio data. When the ARM7 thread reads a chunk of data from the host it posts a "Conditional_Signal" to the audio decoder thread that it has read that chunk. When the audio thread is ready to process the data it checks for the above signal, and if it is set it reads the data, otherwise it will wait on the "Conditional_Wait" signal. Once the audio decoder processes the data it notifies the buffer structure that it has read that amount of data, so that if a chunk of data size has been read that chunk of memory is deallocated. If no constraints are imposed on the threads, MMOSA schedules them as time sharing threads. A real time constraint is imposed on the audio decoder thread, as would occur in the hardware solution. the audio decoder thread itself is partitioned into the ARM7 part which is coded in "C", and the VP part which is coded in assembly language. The communication between these two threads is through a set of MMOSA functions shown in Fig. 5. These functions lock the VP process for this ARM7 thread, sets the program counter to the first instruction of the VP code, sets register values, resumes execution of VP, and gets the

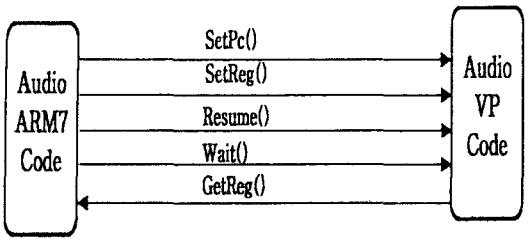

Fig. 5 ARM7/VP Code Interaction

contents of VP registers. The ARM7 thread may context switches at any given point in the ARM7 code if the kernel needs to, but the VP context switches only to at points referred to as Synch points(VpMaybeSync) or at exit points. These points are chosen so that the least amount of information has to stored on exiting, and restored when the code is reentered. On reaching these points context switching routine will be called which will store/restore the necessary state. The VP code is assumed using "asmp.exe" which creates a *.o file. Next using the utility elf2c.exe the *.o is converted to an array of opcode which is defined as a global variable in the system decoder. To simulate the above described system MMOSA is loaded into the location 0x3f80, and the image file is loaded in next, and the input to the system decoder which is a multiplexed video and audio data is read in through the "Get File" option of MSP-Sim and is read into the array defined in the system decoder.

6. Conclusion

A system decoder in MPEG1 demultiplxes the multiplezed stream, and the elementary streams so produced serve as inputs to video and audio decoders, whose outputs are decoded video and audio signals. An audio decoder in a multimedia processor improves decoding efficiency and performance through the usage of multiple parallel processor including a scalar processor and a vector processors(VP). The scalar processor(ARM7) most efficiently performs tasks including bit manipulation, indexing and conditional operations. The vector processor efficiently performs operations involving multiple data calculations, operating on unconditional, sequential data. Improved performance is achieved by executing as many operations as possible on the vector processor, rather than the scalar processor, so long as the data is sequential data. In this paper, we show the description of the implementation of efficiently integrated MPEG1 Audio decoder with MPEG1 System decoder under the MMOSA that has a compact high-speed design and is also fast and powerful enough to run huge interactive systems on the MSP, with emphasis on the scheduling synchronization, and communication between threads running on ARM7 and VP processors.

Reference

[1] ISO/IEC CD11172-3 *Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1.5 Mbit/s*,p41, Apr. 1992.

[2] N.S.Jayant and P.Noll, *"Digital Coding of Waveform"*, Englewood Cliffs, NJ: Prentice Hall, pp.272, 1984

[3] Tom Shanley, *"PCI System Architecture"*, PC System Architecture Series.

[4] *"Samsung MSP Architecture Specification,"*, Feb. 1997.
[5] *"ARM Software Development Toolkit Reference Manual"*, Advanced RISC Machine Ltd(ARM), England.
[6] *"MSP Escalante Real-Time Kernel"*, Microsoft-MSP Escalante Real-Time Kernel(1996)
[7] *"MMOSA Adaptation Kit"*, Microsoft(1995)

홍 기 천

- 1985년 성균관대학교 전자공학과 졸업(공학사)
- 1987년 Stevens Inst. of Tech. 전자공학 석사
- 1994년 Stevens Inst. of Tech. 전자공학 박사
- 1989~1994년 Advanced Telecommunications Inst 연구조교
- 1992~1993년 미국 벨통신연구소(Bellcore) 연구원
- 1994~1998년 삼성 반도체 기흥연구소 및 산호세 연구소 선임연구원
- 1998년 현재 수원대학교 정보통신공학과 전임교수
- 관심분야 : 멀티미디어 시스템, 프로세서 개발, 통신 및 신호처리 압축, 입체 음향효과, 음향 및 영상 스트리밍 기술