Multimedia Processors

ICHIRO KURODA, MEMBER, IEEE, AND TAKAO NISHITANI, FELLOW, IEEE

Invited Paper

This paper describes recent large-scale-integration programmable processors designed for multimedia processing such as real-time compression and decompression of audio and video as well as the generation of computer graphics. As the target of these processors is to handle audio and video in real time, the processing capability must be increased tenfold compared to that of conventional microprocessors, which were designed to handle mainly texts, figures, tables, and photographs. To clarify the advantages of a high-speed multimedia processing capability, we define these chips as multimedia processors. Recent general-purpose microprocessors for workstations and personal computers (PC's) use special built-in hardware for multimedia processing, so the multimedia processors described in this paper include these modified general-purpose microprocessors. After briefly reviewing the history of programmable processors, we classify multimedia processors into five categories depending on their basic architecture. The categories are reduced instruction set computer (RISC) microprocessors for workstations, complex instruction set computer microprocessors for PC's, embedded RISC's, low-power digital signal processors (DSP's), which are mainly used for mobile communications devices, and media processors that support PC's for multimedia applications. These five classes are then grouped into two: microprocessors with a multimedia instruction set and highly parallel DSP's. An architectural comparison between these two groups on the basis of Moving Picture Experts Group decoding applications is made, and the advantages and disadvantages of each class are clarified. Future processors, including "system on a chip," and their applications are also discussed.

Keywords— *Digital signal processors, embedded processors, media processors, microprocessors, MPEG, multimedia, multimedia extensions, personal computers, software MPEG decoders, video compression/decompression.*

I. INTRODUCTION

The performance of microprocessors has steadily improved for 26 years as their clock frequencies have been increased. The pipelining used in reduced instruction set computer (RISC) chips has been a key advance in this area, and these chips are used as host central processing units (CPU's) for personal computers (PC's) and workstations with operating systems. They are also used as controllers for game machines and other consumer electronic products. On the other hand, digital signal processors (DSP's) have achieved their high performance by incorporating hardware function units such as multiply accumulators, arithmetic and logic units (ALU's), and counters, which are controlled by parallel operations with moderate clock frequencies. These processors are used for speech compression for mobile phones, voice-band modems, and facsimile machines, as well as for the acceleration of sound and still-picture image processing on PC's.

Multimedia processing is now expected to be the driving force in the evolution of both microprocessors and DSP's. The introduction of digital audio and video was the starting point of multimedia because it enabled audio and video, as well as text, figures, and tables, to be used in a digital form in a computer and be handled in the same manner. However, digital audio and video require a tremendous amount of information bandwidth unless compression technology is used. Also, the amount of audio and video data for a given application is highly dependent on the required quality and can vary over a wide range. For example, high-definition television (HDTV) (1920 $\times$ 1080 pixels with 60 fields/s) is expected to be compressed into around 20 Mbit/s, while a H.263 [1] video-phone terminal using subquarter-common intermediate format (128 $\times$ 96 pixels) with 7.5 frames/s is expected to be 10–20 kbit/s. In this example, the information throughput differs by a factor of about 1000. Compression techniques call for a large amount of processing, but this also depends on the desired quality and information throughput. The required processing rate for compression ranges from 100 megaoperations per second (MOPS) to more than one teraoperations per second. National Television Systems Committee (NTSC) resolution MPEG-2[1] [2] decoding, for example, requires more than 400 MOPS, and 30 gigaoperations per second are required for the encoding. This wide variety of demands for processing and multimedia quality has led to the software implementations of such compression techniques on micro-

Manuscript received July 15, 1997; revised January 20, 1998. The Guest Editor coordinating the review of this paper and approving it for publication was K. J. R. Liu.

I. Kuroda is with C&C Media Research Laboratories, NEC Corporation, Kawasaki 216 Japan (e-mail: kuroda@ccm.cl.nec.co.jp).

T. Nishitani is with Silicon Systems Research Laboratories, NEC Corporation, Sagamihara 229 Japan (e-mail: takao@mel.cl.nec.co.jp).

Publisher Item Identifier S 0018-9219(98)03523-3.

[1]MPEG-2 is a standard of the Moving Picture Experts Group (MPEG) of the International Standards Organization.

0018–9219/98$10.00 © 1998 IEEE

processors and DSP's to create an affordable multimedia environment.

Recently developed microprocessors and programmable DSP chips offer powerful processing capabilities that enable real-time video and audio compression/decompression. These processors were designed with the target of realizing a software-implemented MPEG-1/2 encoder/decoder in real time. To achieve this, these chips have built-in functions, in addition to their conventional architectures, that support MPEG processing. Because a wide variety of applications are available to users, careful selection of these chips is essential to ensure flexibility in the independent application areas. This is because the chips' basic architectures differ significantly, and their respective advantages are highly related to their architecture. If this is not taken into considerations, it will be very hard to design future high-performance multimedia systems.

In this paper, we begin with a brief historical overview of programmable processors. The programmable processors now available are classified in Section II, and Section III describes the features of the different classes. Multimedia-enhanced instructions for microprocessors are discussed in Section IV, and Section V describes an evaluation of the various architectural features using MPEG-2 software decoding as an example. In Section VI, we discuss our expectations as to the future path of multimedia processor development.

II. A Brief History of Multimedia Processing

A. History

Programmable DSP chips have been used since 1980. They employed a built-in multiplier in addition to the conventional ALU, and their architectures were based on a pipelined multiply-and-accumulate (MAC) function with parallel control. This made their processing capability an order of magnitude higher than that of general-purpose microprocessors and made possible single-chip modems and single-chip low-bit-rate speech codecs. The DSP processing capability has improved steadily since 1980 (Fig. 1) but has also suddenly jumped twice, when the DSP applications were shifted to more complex areas [3]. The first jump occurred in 1991, when video signals started to be processed by high-speed DSP's [4]–[6]. This performance improvement is realized by higher clocks as well as single instruction stream multiple data stream (SIMD) architectures. However, the video format mainly used at that time was a quarter NTSC format intended for video-conferencing purposes, which was called the common intermediate format (CIF) (352×288 pixels). The second jump occurred in 1993 for MPEG-2 applications [7], [8], where a full NTSC resolution format was required. This time, SIMD architectures are enhanced to incorporate very long instruction word (VLIW) controls (SIMD+ in Fig. 1). MPEG-2 applications include set-top boxes for digital cable TV and video on demand, as well as digital versatile discs (DVD's). These applications have attracted particular attention during the current multimedia boom.

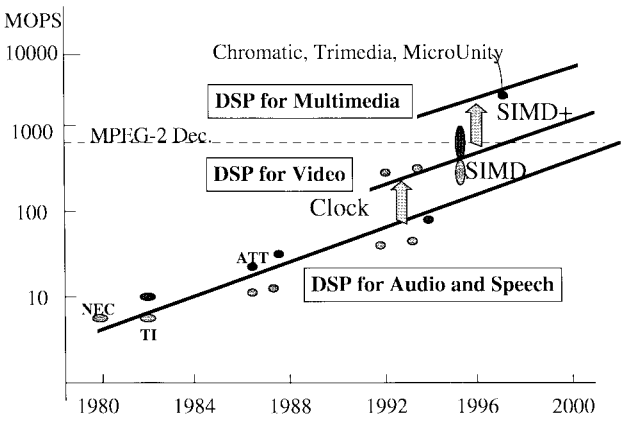


Fig. 1. Performance improvements of DSP's.

Microprocessors are also attracting interest in this area due to multimedia enhancements in their instruction sets. A long-word ALU in a microprocessor can be divided into several small-word ALU's, and several pieces of independent short-word data can be processed by a single instruction. This approach was first used for graphics instructions [9], [10]. Then RISC processors for workstations [11], [12] to process MPEG decoding followed. Recent microprocessors for PC's [13] also employ this approach. These microprocessors have an increased processing capability for video, but this multimedia enhancement makes the programming for the microprocessors much more complex. Efficient programming can only be attained if experts tune the software using assembly languages, just as in DSP approaches.

Another approach to realize multimedia functions is to enhance the graphics acceleration chips used in PC's. As today's PC's always include graphics chips to enhance the user interface, one possible way to improve the multimedia processing capability is to increase the processing capability of these graphics chips.

There are also many MPEG application-specific integrated circuit (ASIC) chips that contain a RISC core for control and housekeeping purposes. The most successful approach right now belongs to these classes [14]–[16], but in this paper, such chips are omitted for proceeding processor architectural considerations.

B. Variation of Multimedia Processors

A wide variety of processors have been derived from microprocessors and DSP's, with the goal of improving multimedia processing capabilities. These multimedia processors can be classified in terms of their structure into the five categories shown in Fig. 2 [3]. These categories are RISC microprocessors for workstations and servers, complex instruction set computer (CISC) microprocessors for PC's, embedded microprocessors, low power consumption DSP's, and DSP's for PC acceleration (which are also called media processors).

The clock frequency versus parallelism shown in Fig. 3 also illustrates the structural differences. The horizontal axis in Fig. 3 shows the number of parallel 16-bit operations or

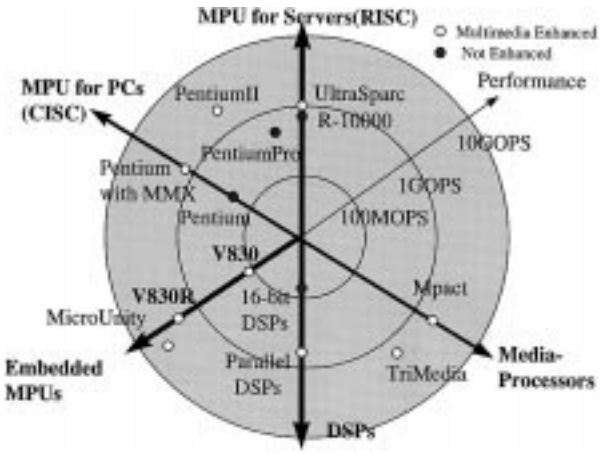


Fig. 2. Variations of multimedia processors.

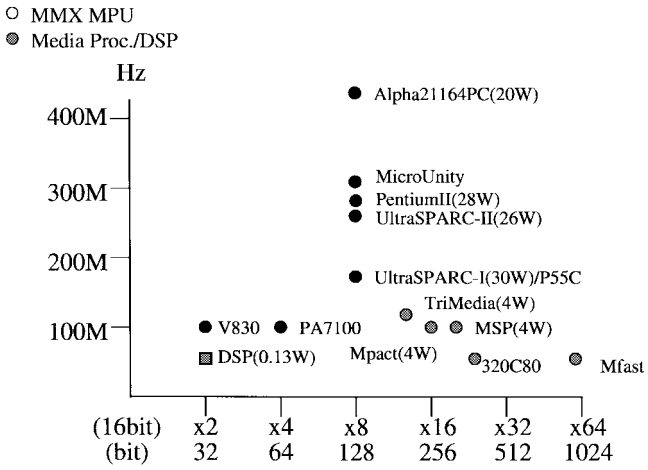


Fig. 3. Frequency versus parallelism.

the total bit length of the arithmetic units, and the vertical axis shows the clock frequency. Microprocessors can now operate at a frequency of up to 450 MHz with maximum parallelism of eight. On the other hand, media processors have greater parallelism with lower clock frequencies of 50–100 MHz. RISC and CISC microprocessors dissipate more than 20 W, while the power dissipation of media processors is about 4 W. The power dissipation of embedded RISC's [17] and mobile DSP's [18] is less than 1 W.

Although we will examine these architectural differences in more detail in the following sections, these figures reflect certain trends in multimedia processor features and development. First, microprocessors for servers and PC's have been enhanced to handle multimedia processing while maintaining the compatibility with their previous generations. To allow compatibility with the huge amounts of existing software, there have been many restrictions on their enhancement; in other words, the heavy burden of hardware increase resulted. This is why these processors use very high clock frequencies and consume a lot of power.

On the other hand, new architectures with aggressive performance enhancements are being introduced for embedded microprocessors and DSP's, even though some of them have the functionality of a stand-alone CPU with high-level language support. The architecture of mobile DSP's is not being aggressively modified at present, due to the need for low power dissipation for battery operation and the limited bandwidth of wireless channels. Accelerator DSP's or media processors have an interface for the host CPU. Their internal structures are designed for more complex multimedia processing such as MPEG-1 encoding. These embedded microprocessors and DSP-based chips enable the use of lower system clock frequencies and reduce power dissipation.

Table 1 MPEG Parameters

	MPEG-1	MPEG-2 MP@ML
Horizontal size (pixels)	352	720
Vertical size (lines)	240	480
Frames/s	30	30
Display byte/s	3.8M	15.55M
Compressed bits/s	~1.5M	4-15M
Number of macroblocks/s	9,900	40,500
Number of blocks/s	59,400	243,000

C. Architecture Evaluation Points for MPEG Decoding

As the multimedia processors currently being developed target MPEG-2 decoding in software, the performance and functionality for MPEG-2 video decoding are key issues in the design of the multimedia processor architectures. Decoding and playback of the compressed bitstream start with variable-length decoding, followed by inverse quantization to retrieve discrete cosine transform (DCT) coefficients from the compressed bitstream. Then, an inverse (I)DCT operation produces the prediction-error signal. This signal retrieves the decoded frame with the addition of a motion-prediction signal. The motion-prediction signal is calculated by pixel interpolation using one or two previously decoded frames. The decoded frame is transformed into a display format and transferred to the video random-access memory (RAM) and a video output buffer. These transforms to display formats include a YUV to red-green-blue transform as well as dithering. This decompression process is carried out by a square image block called the macroblock (16×16 pixels with color components) or the block (8×8 pixels).

Table 1 shows the major parameters for MPEG-1 [19] and MPEG-2 MP@ML [2].

MPEG decoding for multimedia processors requires that the following five important functions be included in the system architectures.

1) *Bit manipulation function*, which parses and selects bit strings in serial bit streams. Variable-length encoding and decoding belong to this category.

2) *Arithmetic operations*, which consist of multiplication, add/subtract, and other specific arithmetic operations, such as the sum of the absolute difference for motion estimation. Different word lengths are also desirable to improve hardware efficiency in handling many different media, such as 8-bit video and 20-bit audio data. Parallel processing units are also important for efficient IDCT processing, which requires a lot of multiplication due to the nature of the two-dimensional IDCT algorithms.
3) *Memory access to a large memory space* to provide a video frame buffer that usually cannot reside in a processor on chip memory. The frequent access to the frame buffer for motion compensation requires a high-bandwidth memory interface.
4) *Stream data input/output* (I/O) for media streams such as video and audio as well as compressed bitstreams. For video signals, for example, this may consist of the capture and display of the signals, as well as video-format conversion (e.g., RGB to YUV). This kind of I/O functionality is also needed for compressed bitstreams for storage media, such as hard disks, compact discs and DVD's, and for the communication networks.
5) *Real-time task switching* that supports hard real-time deadlines. This requires a sample-by-sample or frame-by-frame time constraint. Switching between different types of simultaneous media processing to synchronize video and audio decoding is one example.

In the following sections, we discuss how the different processors are being enhanced in terms of those five key functions.

III. Architectures for Media Processing

A. General-Purpose Microprocessors (RISC and CISC)

As today's CISC processors share the advanced architectural technologies used in RISC processors, the processors in both categories are considered in this section. Today's high-end general-purpose microprocessors can issue two to four instructions per cycle by using superscalar control [21], which enables more than one floating-point instruction or several multimedia instructions to be issued at one time. This control mechanism has two types of issuing mechanisms. One is the in-order-issue control, which issues instructions in the order they are stored in the program memory. The other is the out-of-order-issue control, where the issue order depends on the data priority rather than the storage order. This is effective for microprocessors that operate above 200 MHz and have long pipeline latency instructions, where out-of-order control can maximize the high-speed pipelined ALU performance.

Fig. 4 shows an example of an out-of-order superscalar microprocessor. Implementation of out-of-order control requires several additional hardware functional units such as a reorder buffer that controls the instruction issue and completion and a reservation station that reorders the actual instruction issues for the execution units and renamed register files, as well as control circuits for these units. These components take up a large part of the silicon area and contribute to the power dissipation.

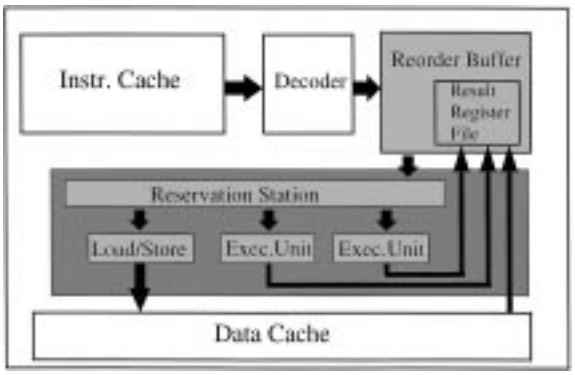


Fig. 4. Out-of-order superscalar microprocessor.

In out-of-order control, the number of registers can actually be increased by register renaming. This improves the processing performance by reducing the number of load and store operations of intermediate calculation results to the memory as well as by reducing the processor stall cycles due to data dependencies. On the other hand, image processing has inherent parallelism in pixels and macroblocks. Although a large amount of hardware must be implemented for superscalar control, the issue of two to four parallel instructions does not fully take advantage of the parallelism in image processing. Other aspects of general-purpose microprocessor architectures related to media processing in terms of the five functions given in Section II are illustrated in Fig. 5.

The arithmetic operations of microprocessors have a word-length problem. Microprocessor word lengths have been increased to 32 or 64 bits. On the other hand, the word lengths needed for multimedia processing are 8, 16, or 24 bits, which are much shorter than the word lengths of today's microprocessors. This provides extravagant margins if we handle multimedia data with arithmetic instructions on microprocessors. Also, the data type available in high-level languages such as C is not suitable for multimedia data. This complicates the problem because general-purpose microprocessors are commonly programmed using high-level languages.

The memory access to a large memory space in microprocessors is also a problem. In image processing, there is frequent access to large video frames (4.15 Mbit/frame or 518.4 Kbyte/frame for MPEG-2) that cannot reside in the first- or second-level cache. Therefore, we cannot expect the high cache hit rate usually assumed in general-purpose applications. Moreover, there is a difference in the data locality, which affects the memory access performance. The cache mechanism is designed to use the one-dimensional locality of consecutive addresses, but image processing has the two-dimensional locality of access.

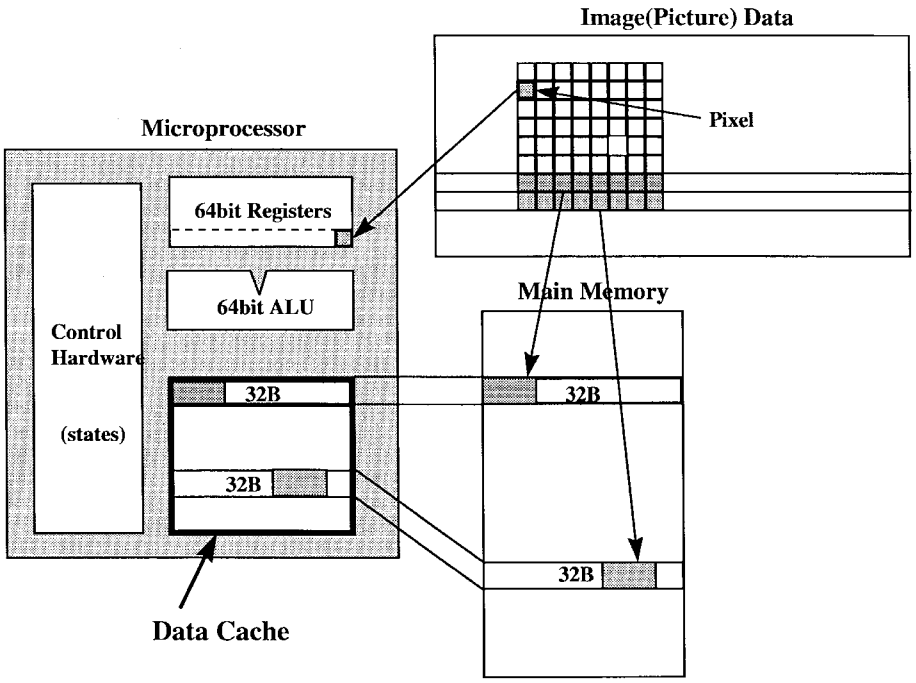


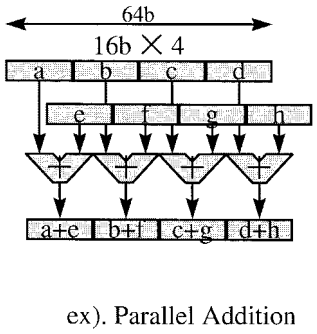
Fig. 5. Microprocessor architecture.

The demands of hard real-time processing for multimedia applications are also a problem for microprocessors. In media processing for audio and video, processing of each audio sample or video frame should be completed within a predetermined sample or frame interval time. This requires the predictability of the execution time, which is not easy to achieve in microprocessors. Data-dependent operations and memory accesses using cache mechanisms make it difficult to predict the processing cycles in the microprocessors. Task switching with less overhead is also required; however, internal state introduced by out-of-order superscalar processors makes it difficult to achieve fast task switching. Moreover, operating systems are not designed for hard real-time processing.

Other functions, such as bit manipulation for multimedia processing, are not very efficient in microprocessors because they are designed to treat continuous text streams with a fixed word length. Stream I/O functions are usually not included in the microprocessor. They are performed by other chips in the system.

The multimedia processing capability of recent microprocessors has been improved by multimedia extensions. These microprocessors enhance the arithmetic performance by dividing the long-word arithmetic unit (e.g., 64 bits) to execute two to eight operations in parallel by using SIMD-type multimedia instructions, as shown in Fig. 6. These instructions are implemented in either an integer data path [22], [23] or a floating-point (or a coprocessor) data path [12], [13] in microprocessors.

The cache-miss penalties by off-chip main memory access are going to be a serious problem in microprocessor-based multimedia processing because the memory access latencies of microprocessors tend to be longer due to their higher clock frequencies. For most multimedia applications, however, the address for the memory access can be calculated beforehand. It is possible to avoid cache-miss penalties by getting the data into the cache line using software prefetch instructions, which are implemented by some microprocessors. Moreover, one microprocessor [12] has block-transfer instructions that transfer a block of data (64 bytes) with one instruction outside of the cache mechanism.



ex). Parallel Addition

Fig. 6. Multimedia instruction.

B. Embedded Microprocessors

Microprocessors for embedded applications (not for PC's or workstations) also enhance the multimedia-processing capability. Target applications of this class of microprocessors include applications that originally used DSP's. In addition to these applications, multimedia applications such as Internet terminals, set-top boxes, car navigators, and personal digital assistants will be targets for these microprocessors.

A class of embedded RISC processors are inexpensive and consume little power. They do not employ a com-

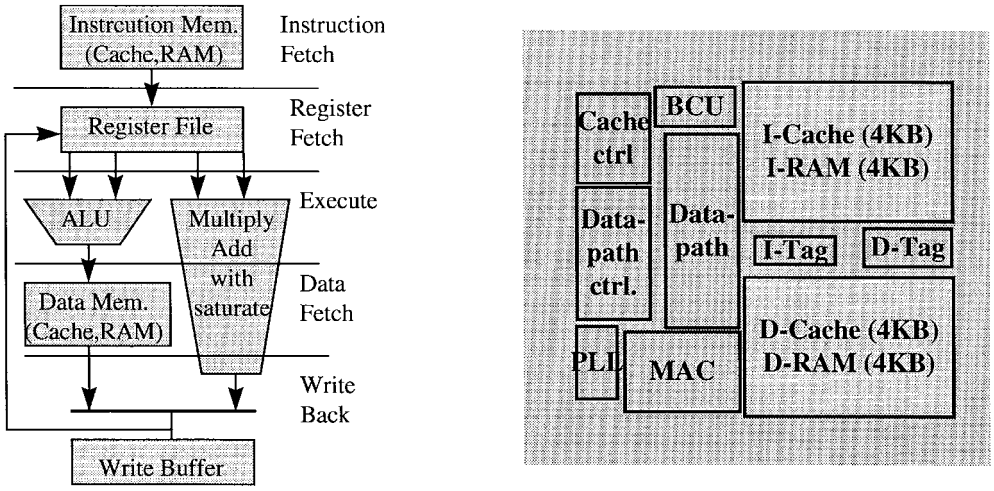


Fig. 7. Embedded RISC (V830). (a) Data path. (b) Memory configuration.

plicated control mechanism such as out-of-order controls. Therefore, they can be used in low-cost systems such as game machines and consumer electronics. The multimedia performance of these processors has been enhanced to meet the requirements for these applications.

The arithmetic performance of embedded microprocessors can be enhanced by using a hardware multiply accumulator such as that shown in Fig. 7(a) [17]. The requirements for real-time processing and the large memory space are met by having both caches and internal memories (buffers). An example of a V830 [17] cache/RAM memory structure is shown in Fig. 7(b). The V830 has 8-kB direct-mapped/write-through caches (4 kB for instructions and 4 kB for data) and 8-kB RAM's (4 kB for instructions and 4 kB for data). As embedded microprocessors are normally used in low-cost systems that do not have a second-level cache, the cache-miss penalty is likely to be heavy. To prevent this, an internal RAM that is guaranteed not to cause a cache miss has been developed. This has been instrumental in realizing high-performance MPEG-1 [24] program development.

Another class of embedded microprocessors includes more sophisticated chips, such as the MicroUnity Mediaprocessor [25], which is also classified as a media processor. The processor has special memory interfaces as well as a stream I/O interface with accompanying chips (Fig. 8). High-bandwidth memory access is realized by using special memory interfaces such as synchronous dynamic (SD)RAM and rambus (R)DRAM [26]. The processor also has the capability for general-purpose microprocessors, such as virtual memory and memory management for stand-alone use. The arithmetic performance of the processor has been enhanced by using SIMD-type multimedia instructions with long word size [27]. A large register file (128×32 bits) also helps to improve the arithmetic performance of the processor. A memory mapped I/O avoids coherency and latency problems in a division multiple access (DMA)-based I/O system. Analog interfaces for audio and video are implemented on a separate chip [28]. With the advanced features described here and a higher clock frequency (1 GHz), this processor should be able to handle broad-band media, though it will result in higher power consumption.

C. DSP

DSP's have been developed mainly for speech processing and communications processing, such as in modems [29], [30]. They are also used in sound processing and fax modems for PC accelerator boards.

DSP architectures (Fig. 9) are designed for high-speed multiply-accumulate operations and are capable of two operand data transfers with two internal memory address modifications and one multiply-accumulate operation for each cycle. Recently, they have been used extensively in speech compression/decompression for mobile phones. Lower power consumption while maintaining high processing performance are the main requirements for such applications, which makes DSP's have a higher performance/power ratio than other processors.

Referring back to the key functions we listed in Section II, the bit manipulation performance has been enhanced by employing a special function unit for some mobile DSP's, but the performance of other functions such as arithmetic operation, external memory access, and stream I/O is relatively low compared with today's high-speed microprocessors because current mobile applications do not require high performance in these areas. A mobile video phone is now being developed using the DSP because of its low power consumption [31]. The limited communication bandwidth makes the required performance for this application lower than in other video applications such as MPEG-2 applications.

RISC microprocessors that enhance DSP capability are also candidates for mobile applications. However, a separate DSP and microprocessor configuration in a chip looks more promising because it is easier to control the power, depending on the state of the communication. Currently, the DSP is the most suitable processor for mobile-phone

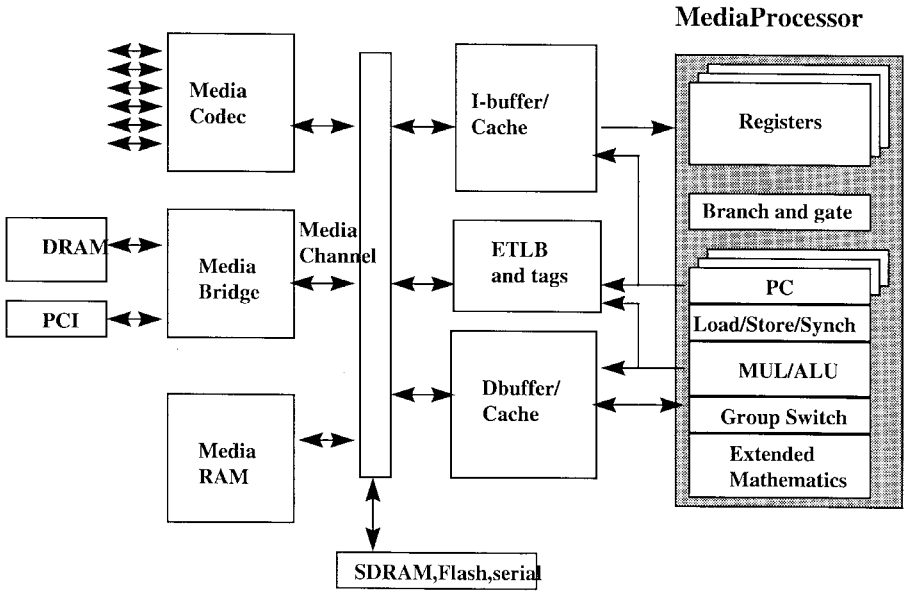


Fig. 8. MicroUnity Mediaprocessor.

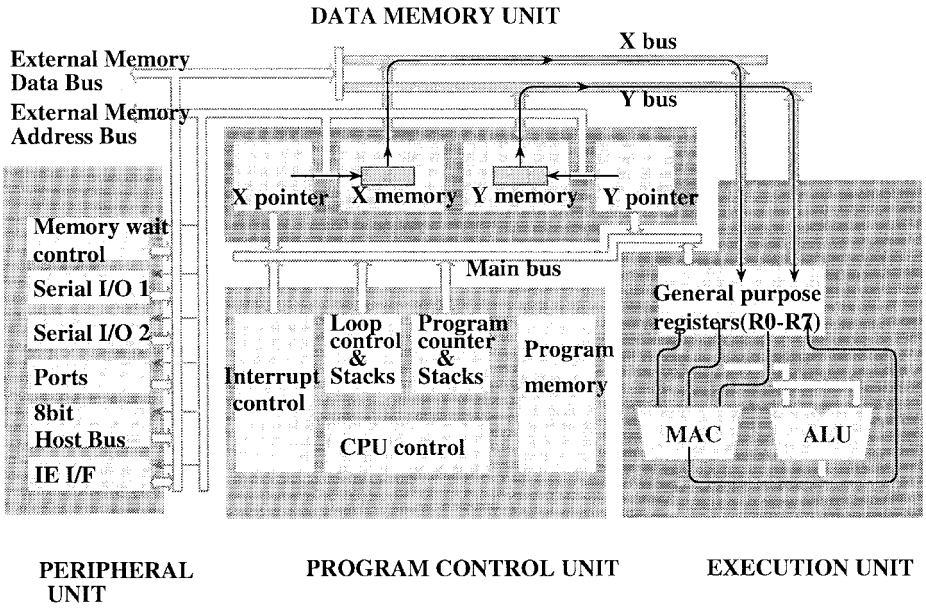


Fig. 9. DSP architecture (7701×).

applications (compression and decompression of speech and video). When the communication bandwidth is increased in the next-generation communication systems, however, such as wide-band code (C)DMA or high-speed wireless local-area networks, higher DSP performance without sacrificing low power consumption will be required.

1) Parallel DSP's: One way to extend the DSP architecture is to get higher performance by integrating parallel DSP's on a chip [32] to handle multimedia applications. As an example of this, a multiple-instruction-stream/multiple data-stream-type parallel processor that integrates several DSP's as well as a microprocessor is shown in Fig. 10. The DSP's and memory blocks are connected by a cross-bar switch and form a shared memory multiprocessor system. For the concurrent operation of several tasks, a multitask kernel on the microprocessor controls each task on the parallel DSP's. Each DSP has functions for media processing such as the multimedia instructions that enable parallel operations by dividing the 32-bit data path into two 16-bit or four 8-bit units, as well as an instruction for motion estimation that processes one pixel in 0.5 cycles.

The H.324 [33] video-conference system, which simultaneously runs multiple tasks such as an audio codec, a video codec, and a system control, is realized using the multitask capability of the parallel processors [34].

Another parallel DSP, an array processor of 20 DSP's for the PC accelerator application, is shown in Fig. 11 [35]. Each DSP is connected in a two-dimensional mesh and is

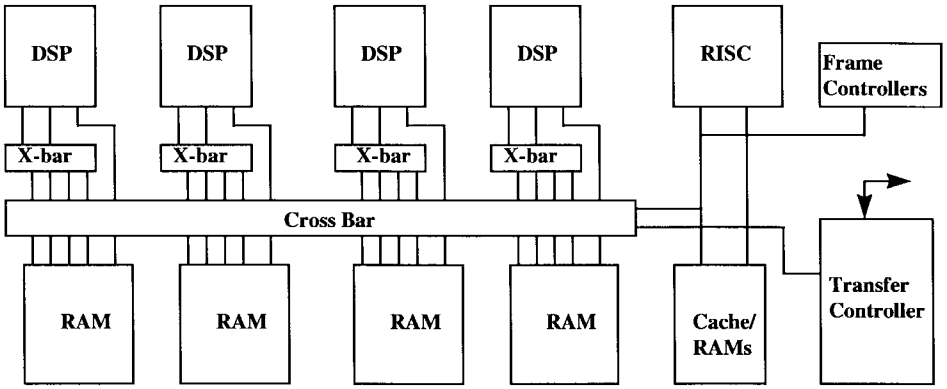


Fig. 10. Parallel DSP (320C80).

controlled in an SIMD scheme. Some DSP's have a control capability over other sets of DSP's, and a memory interface supports block transfer using DMA.

D. Media Processors

Another extension of DSP's that has enhanced VLIW control is media processors [7], [8], which can issue two to five instructions in parallel. VLIW achieves high performance with less control circuits than with superscalar control in microprocessors. In addition to VLIW, some features used in DSP's [18]—such as zero-overhead loop control, which allows a specified number of iterations without introducing overheads—are employed. In respect to the key functions for multimedia applications, media processors have more function units than issue slots to enable higher arithmetic performance. One instruction controls several function units, and some instructions enable SIMD-type parallel operations like the multimedia instructions for microprocessors.

Media processors inherit many features from traditional DSP's in their data-path architectures, for example, the direct connection of function units (Fig. 12), which is used in the multiply-accumulate architectures of DSP's. However, media processors also have new features that are not usually supported in DSP's. One new feature is operations for parallel-packed data, which can be efficiently utilized in image processing like multimedia instructions. Another new feature is a large register file that is useful for storing intermediate data for video compression/decompression.

In addition to these programmable function units, special-purpose function blocks are used to achieve high multimedia performance at lower clock frequencies. In particular, the bit manipulation performance is enhanced by a special circuit block for variable-length decoding; such a block is shown as the variable-length decoder (VLD) coprocessor in Fig. 13.

Media processors use noncache memories to store programs and data in the same way as internal memories are used in DSP's. Although media processors do not always have the functionalities for general-purpose processors, such as a virtual memory, one processor [8] has CPU functions such as an instruction and a data cache, as shown in Fig. 13.

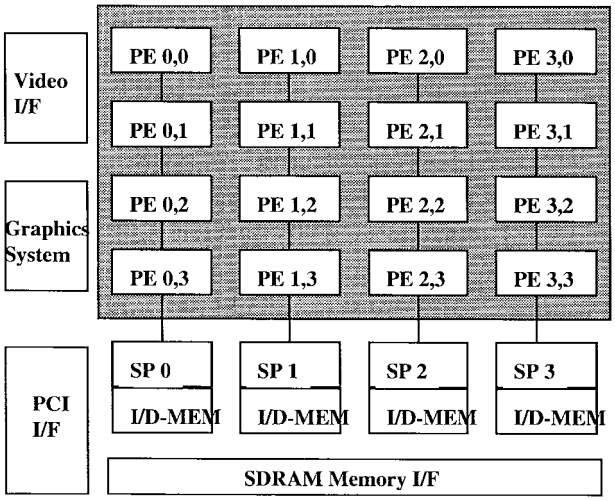


Fig. 11. Array DSP (Mfast).

They also employ a high-speed memory interface for the RDRAM or SDRAM [26], which are used for image processing. Frame memory access and data transfer for video output, which cause cache-miss penalties for microprocessors, can be realized without introducing overheads by using DMA transfers.

Media processors have bitstream data I/O interfaces for audio, video, and other media with a special function block for I/O processing such as a YUV-to-RGB format conversion for display processing (Fig. 13). These video I/O interfaces and a peripheral component interconnect interface that allows use of a PC accelerator enable the media processor to replace conventional graphics accelerators.

Media processors have advantages in real-time processing. The deadline scheduler in their own real-time kernel can realize hard real-time scheduling between audio and video processing with accurate timing. No virtual-memory support and minimal cache support make hard real-time multimedia processing possible.

Unfortunately, use of the high-level language compilers is very limited because applications require highly optimized VLIW code. Therefore, assembler-based extensive manual optimization is needed to realize a code that realizes enough performance for media processing.

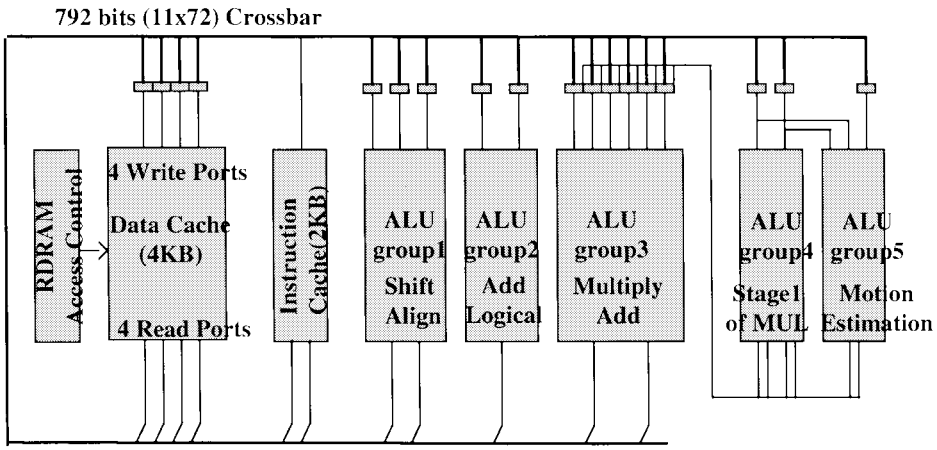


Fig. 12. Media processor data path (Mpact).

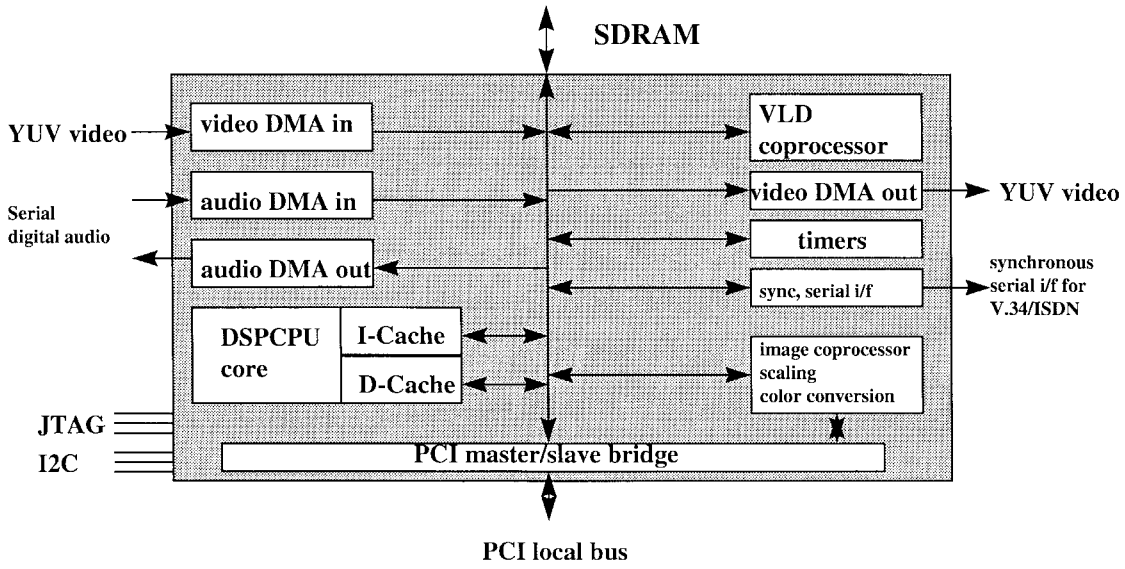


Fig. 13. Media processor system (Trimedia).

E. Comparison of Architectures

In terms of multimedia processing capability, enhancements in RISC, CISC, and embedded RISC are all based on multimedia-enhanced instructions. Low-power DSP's are now used only for mobile communication and have limited capabilities for multimedia processing. Therefore, we focus on multimedia-enhanced microprocessors and media processors in the following comparison of architectures.

The advantages of enhancing the multimedia instructions of microprocessors are that the performance improvement is scalable to the clock frequency and more rapid frequency improvement is feasible compared with the ASIC approach. However, there are performance bottlenecks related to the external bus and the memory access. Another constraint in this approach is that full compatibility with existing operating systems and application software is needed. For example, a new register set, control registers, or other state variables like condition codes cannot be added to the enhanced instruction set. One solution for this is to share the existing floating-point registers between floating-point instructions and multimedia instructions, but this leads to several cycles of overhead in switching between these two classes of instructions in one implementation [13].

For the acceleration of PC multimedia, one approach is a software solution using a high-end CPU with multimedia instructions. Another approach is acceleration by DSP's or media processors with a lower end host CPU. Currently, media processors seem to have advantages in realizing higher quality media processing. With the increased media performance of media-enhanced microprocessors, however, the competition between microprocessors and media processors will be severe.

Fig. 14 shows the typical data flow for each processing block in MPEG decoding with a media-enhanced microprocessor and with a media processor. The multimedia performance for the media-enhanced microprocessor system depends on the system configurations of the memory system and the bus system as follows. The performance of a bit-manipulation-intensive VLD depends on the integer performance of the microprocessors and L1/L2 cache system whose data flow is shown as ① in Fig. 14. The per-

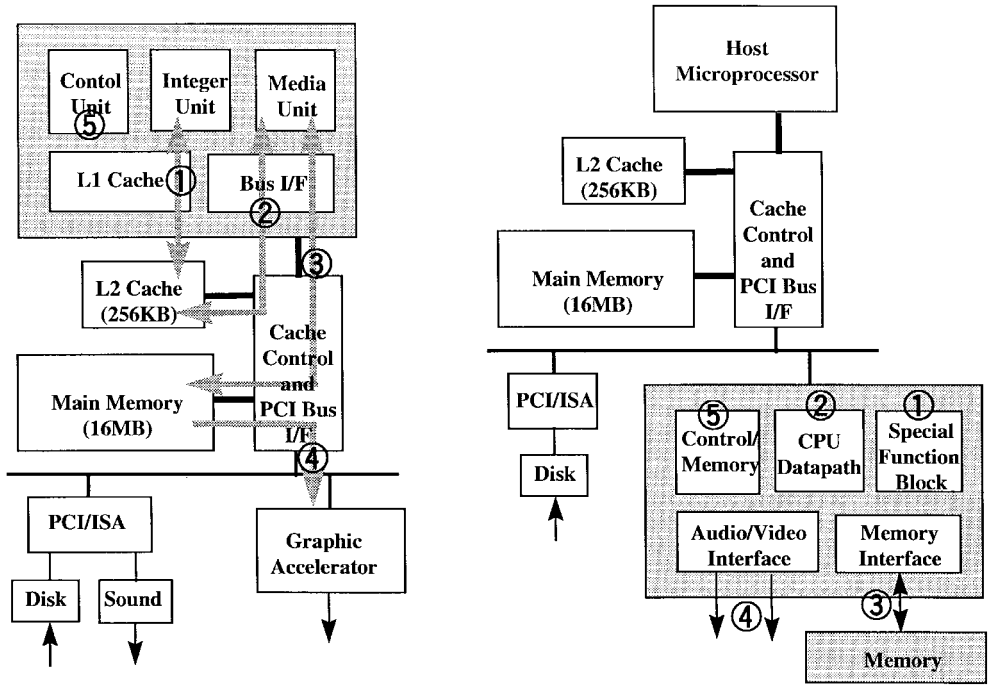


Fig. 14. Data flow in multimedia systems. (a) Media-enhanced microprocessor. (b) Media processor.

formance of an arithmetic-intensive IDCT mainly depends on the media unit and L1/L2 cache [②]. The performance of memory-access-intensive motion compensation depends on the media unit, bus system, and memory system [③]. The performance of a stream-data-I/O-intensive display operation depends on the memory system, the bus, and I/O peripherals, such as a graphics accelerator [④]. The performance of real-time task-switching-intensive audio/visual (A/V) synchronized playback depends on the microprocessor-control mechanism and the operating system [⑤].

On the other hand, for media processors, the performance of all of these components depends on the processor architecture itself and the dedicated memories. Since the five key multimedia functions (Section II) are enhanced in media processors mainly through the use of special hardware, we concentrate in the following sections on the software-based solution using media-enhanced microprocessors, which should be carefully studied for overcoming media processor capability in the future.

IV. Data Path of Media-Enhanced Microprocessors

A. Data-Path Overview

The most enhanced feature in media-enhanced microprocessors is the use of SIMD-type multimedia instructions that use pixel parallelism in video processing. The basic idea is that parallel operation of short words is enabled by dividing the long word for built-in arithmetic units. For example, split addition/subtraction can perform parallel 8-, 16-, or 32-bit word saturated or wrapped operations.

Each sample of a video signal can be expressed as an unsigned byte. Arithmetic operations for a video signal are usually performed in 16-bit units. Each sample of an audio signal is expressed in 16 bits, and arithmetic operations for audio signals are usually performed in 16- or 32-bit units. If we use 16-bit units for audio, though, the signal quality is degraded due to roundoff noise in the arithmetic operations.

The multimedia instruction set includes the split addition/subtraction operations, multiplication/multiply-accumulate operations, special operations such as those for motion estimation, split conditional operations, and shuffle operations. The complexity of the multimedia instruction set depends on the processors. Table 2 summarizes the number of instructions for various multimedia processors.

Table 3 shows the sizes and the number of access ports of the register files for multimedia instructions of multimedia-enhanced microprocessors as well as the types of the register files [integer registers or floating-point (or coprocessor) registers]. With a multimedia instruction set that performs parallel operations, a large register file with many access ports increases the number of operations executed in one processor cycle. Moreover, having enough registers makes it possible to avoid storing intermediate values into the memory.

As shown in Table 3, multimedia instructions are implemented on either an integer data path or a floating-point data path. There are several advantages of implementing multimedia instructions on a floating-point data path [12]. One is that floating-point registers can be used to store the data for multimedia instructions, while integer registers are

Table 2 List of Media Instruction Sets

	UltraSPARC[36]	Pentium[13]	V830R[37]	MIPS[38]	PA-RISC[22]	Alpha[23]
Add/Subt	8	14	10	12	6	-
Mul/Mac	7	3	11	10	-	-
Special	3	-	3	5	3	1
Compare	12	6	4	6	-	8
pack/shuffle	5	9	8	2	2	4
Logical	18	4	4	8	-	-
Shift	0	18	8	5	3	-
Transfer	19(2)	2	8	-	-	-
etc	11(3)	1	-	16(5)	-	-
Total	83	57	56	64	14	13

Table 3 Register Files

	register file type	# of regs.	word width	# of multimedia units
UltraSPARC[36]	FPU	32	64	2
Pentium[13]	FPU	8	64	2
V830R[37]	FPU	32	64	1
MIPS[38]	FPU	32+Acc	64(acc192)	NA
PA-RISC[22]	INT	31	64	2
Alpha[23]	INT	31	64	1

used to store address and control variables. The second is that multimedia and floating-point instructions are usually not used simultaneously, which enables the parallel issue of integer and multimedia instructions.

On the other hand, the advantage of implementing multimedia instructions on an integer data path is that integer functions such as shift or logical function need not be replicated for the floating-point data path [22]. This results in a relatively small number of new instructions and fewer circuits than with the former approach. As shown in Table 2, processors using this approach, such as PA-RISC [22] and Alpha [23], do not have split multiplication/MAC instructions. Alpha [23] especially has a very simple multimedia instruction set that does not even have a split addition operation. However, modifications of the integer data path to accommodate multimedia instructions might have a significant effect on the critical path of the integer data-path pipeline [12].

B. Examples of Multimedia Instructions

1) Multiply/MAC Instructions: The main design issue in multiply-accumulate instructions arises from the fact that the result register requires a word length double that of the input registers. There are several solutions to this problem for 64-bit architectures, as follows.

Type 1) Four parallel 8 $\times$ 16-bit multiplications with scaling to 16-bit outputs, as shown in Fig. 15(a). This approach is designed for picture-format transformations such as RGB to YUV [36].

Type 2) Four parallel 16 $\times$ 16-bit multiplications with two 32-bit results, where each result is the sum of the two multiplications as shown in Fig. 15(b). Four 16 $\times$ 16 multipliers work in parallel with no truncations [13]. However, this approach leads to overhead due to the shuffle operations for the four parallel SIMD algorithms, such as the parallel 1D-IDCT.

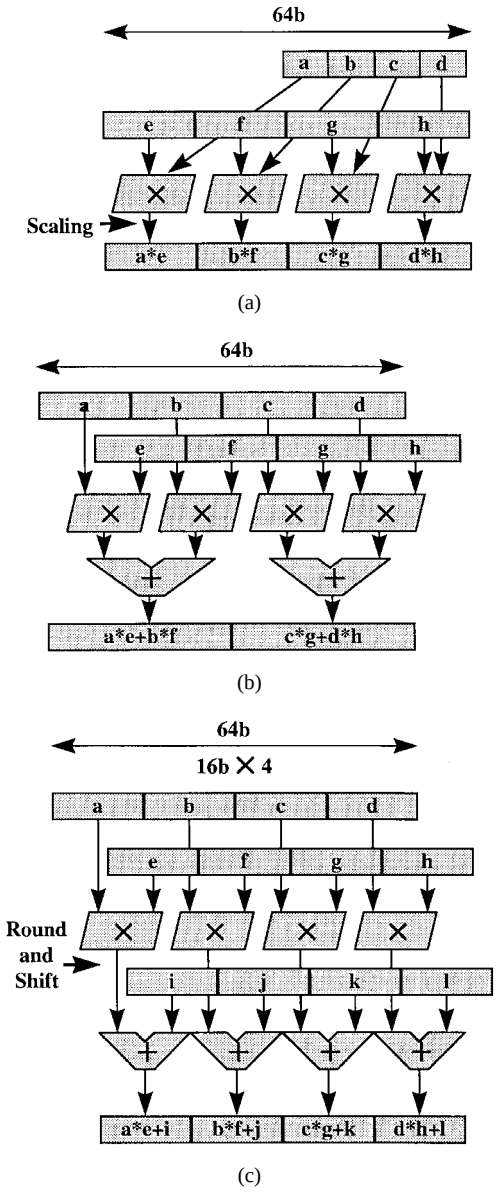


Fig. 15. Split MAC instructions. (a) Type 1. (b) Type 2. (c) Type 3.

Type 3) Four parallel 16 × 16 multiplications with 16-bit accumulations, as shown in Fig. 15(c) [37], [39]. To ensure sufficient accuracy, a rounding and a shift are included in the accumulation. Applications are limited, though, due to the low accuracy of the result.

2) Shuffle Instructions: Shuffle instructions, such as interleave instructions and precision-conversion instructions, as shown in Fig. 16, have an important role in the multimedia instruction set that divides the long word registers. These shuffle instructions exchange data between SIMD parallel arithmetic units and are used for data transfers between split words in combination with full/split word shift instructions and logical instructions. Shuffle instructions convert between different types of data, for example, 8 to 16 bit or vice versa in parallel. These instructions can be used for the matrix transposition, which will be discussed below. They are also used for precision conversion where short word data is converted into long word data by having all zero data as one of the operands.

On the other hand, precision-conversion instructions, which convert long word data into short word data, need to realize scaling operations and saturate operations. Therefore, pack instructions, which are a type of precision-conversion instructions, are used for this purpose [Fig. 16(c)].

V. Processor Solution for MPEG Decoding

This section describes the processor performance required for each function block, including those for bit manipulation, arithmetic operations, memory access, and real-time switching. The effects of multimedia instructions on arithmetic performance are also discussed. The display part is omitted because this is realized through graphics accelerators in today's PC's. The actual performance of MPEG software decoders is also examined using various examples.

A. Bit-Manipulation Performance and Variable-Length Decoding

Variable-length decoding (i.e., Huffman decoding) extracts code words from the compressed bitstream. This bit-manipulation operation consists of the parsing of a variable-length code obtained from a bitstream buffer and the search for a symbol from code tables. A table lookup algorithm [40] can be used for this purpose. As shown in Fig. 17, the lookup table consists of a zero-run (a number of zero coefficients), a level, and a code length. The table is accessed by the first few bits of the bitstream buffer. Then, the buffer is shifted by the code length for the next table lookup. This buffer operation is simulated in microprocessors that have word boundaries by using the comparison, shift, masking, and branch operations. The operation is difficult to parallelize, though, because of the sequential nature of the bitstream.

The required performance is approximately proportional to the average number of symbols in a bitstream processed per second. Parallel symbol decoding is possible depending on the design of the code tables [41], [42]. Application of the SIMD-type parallel instructions, however, is limited. There are instructions, such as double-word shift, which can implement the parsing operation [24] efficiently.

The average number of cycles needed to decode one symbol (DCT coefficient) is around 33, including inverse quantization for a 4-Mbit/s stream in an implementation of parallel decoding [42]. If we assume 5 bits/symbol and that the cycles per instruction is 1.0, a 4-Mbit/s stream requires 26.4 million instructions/s (MIPS). This is a 27% reduction of cycles from nonparallel implementation.

B. Arithmetic Performance and IDCT

An 8 × 8-point two-dimensional IDCT can be realized by using the 8-point one-dimensional IDCT for eight rows

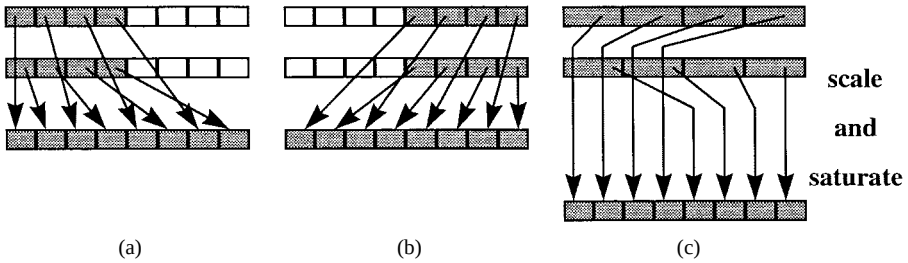


Fig. 16. Shuffle instructions. (a) Interleave(1). (b) Interleave(2). (c) Pack.

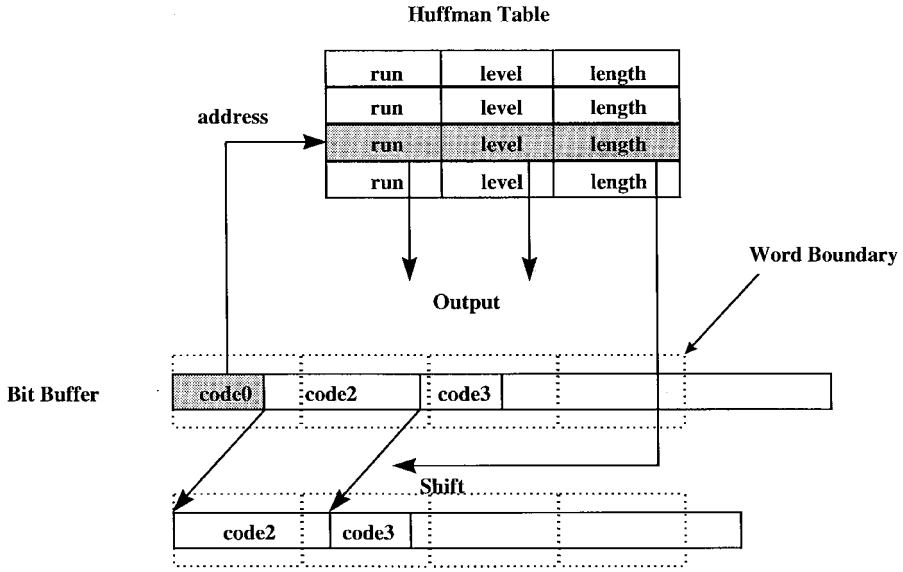


Fig. 17. Software implementation of variable-length decoding.

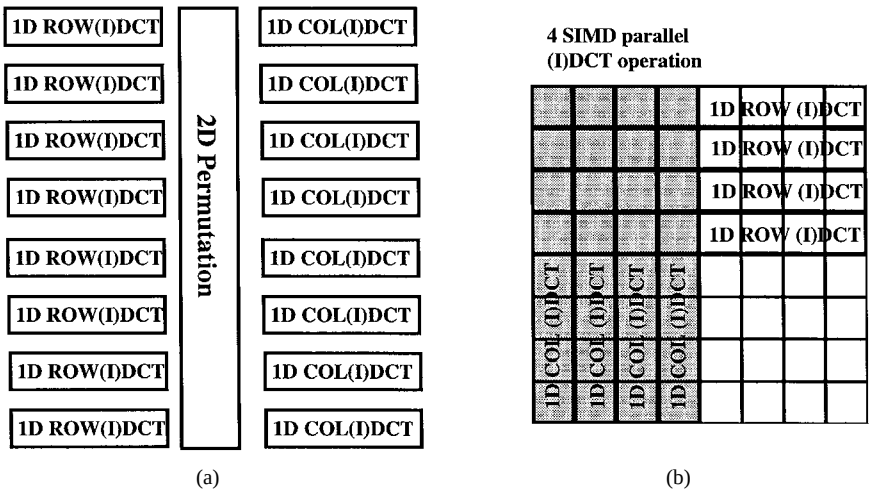


Fig. 18. 2D-(I)DCT implementation using SIMD instructions. (a) Row-column algorithm. (b) Implementation using SIMD instructions.

and eight columns, as shown in Fig. 18(a). An 8-point one-dimensional IDCT is realized through a single 8×8 matrix-vector multiplication that requires 64 multiplications and 64 additions. If we assume that one multiplication requires ten cycles for a nonmedia-enhanced microprocessor, an $8\times$ 8 two-dimensional IDCT that consists of 16 iterations of 8-point one-dimensional IDCT's will require 11 264 cycles.

Several fast algorithms to minimize the number of multiplications have been proposed [43]–[48]. One fast algorithm reduces the number of operations to 80 multiplications and 464 additions [45], which requires 1264 cycles. There is also a fast algorithm that reduces the number of operations to 46 multiplications and 253 additions in average—a total of 713 cycles—by eliminating the operations for zero-value DCT coefficients [48], which requires 173.3 MIPS for

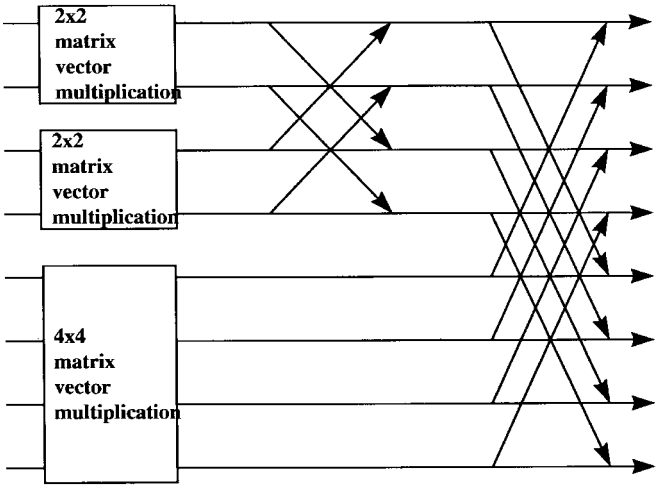


Fig. 19. An 8-point IDCT algorithm.

IDCT in MPEG-2 MP@ML. On the other hand, a simple algorithm can be efficiently used for processors that have fast multiply-accumulate instructions [49].

As shown in Fig. 18(b), parallel SIMD multimedia instructions can provide four parallel 8-point one-dimensional IDCT's for four rows. One-dimensional IDCT's for eight rows can be realized by repeating this twice. A matrix transposition done through shuffle instructions [22] is needed to start one-dimensional DCT's for eight columns.

If we consider the multimedia instructions described in the previous section, Type 1) has a problem in terms of accuracy without the double-precision arithmetic. Type 2) requires shuffle operations to realize four parallel one-dimensional IDCT's. Type 3) can realize four parallel IDCT's easily. An example of an 8-point IDCT algorithm using multiply-accumulate operations is shown in Fig. 19. This straightforward method requires around 200 cycles to realize an 8×8 two-dimensional inverse DCT using Type 3) instructions, that is, 48.6 MIPS for MPEG2 MP@ML. This is more than 50 times faster than the original and 3.5 times faster than the fast algorithms for nonmedia-enhanced processors.

The accuracy of the arithmetic operation is important in IDCT calculations. IEEE 1180 [50], [51] defines the accuracy required for IDCT to avoid degrading the quality of the decoded image. Truncation of the 32-bit multiplication result into 16 bits does not provide sufficient accuracy for IDCT. It has been reported, however, that sufficient accuracy can be obtained by shifting and rounding before truncating to 16 bits [37], [39].

C. Memory-Access Performance and Motion Compensation

MPEG video frames consist of I-frames, which are compressed without motion prediction; P-frames, which are compressed by unidirectional prediction using one previous I- or P-frame, and B-frames, which are compressed by bidirectional prediction using two frames (one past frame and one future frame). These three types of frames are laid out as shown in Fig. 20. Typically, for each I-frame, there are four P-frames and ten B-frames.

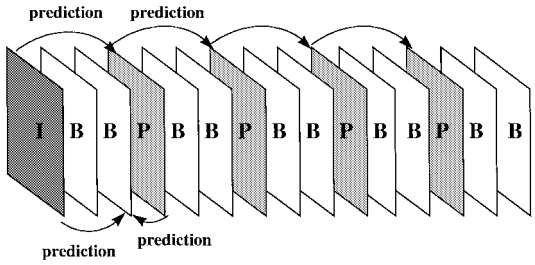


Fig. 20. MPEG IPB frame structure.

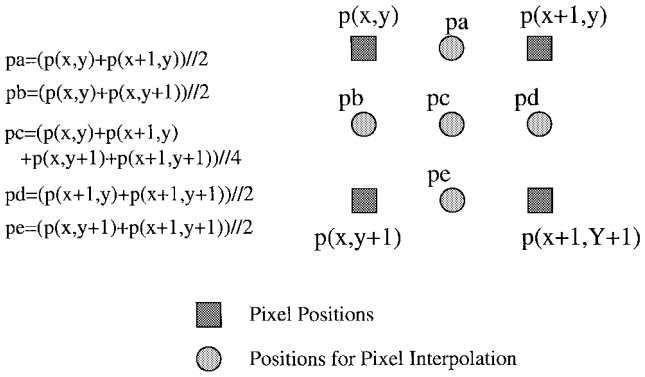


Fig. 21. Pixel interpolation on motion compensation.

In the reconstruction of P-frames or B-frames, motion-compensation operations reconstruct each macroblock in a frame by using the IDCT output and the past or future reference macroblocks. These reference macroblocks are decoded in advance and are pointed by the motion vectors.

Unidirectional motion compensation produces a prediction signal by using one reference (past) frame with pixel interpolation if necessary. When the motion vector is located between each pixel, pixel interpolation is needed to produce a prediction signal by interpolating between neighboring pixels. Fig. 21 shows possible locations of interpolation samples, denoted by pa, pb, pc, pd, and pe.

In the case of bidirectional motion compensation, an average of two prediction signals are necessary. Unidirectional motion compensation requires at most a total of four addition operations, one shift operation, and one 8-bit clipping. One addition is to add a DCT coefficient and a prediction signal. That means 384 operations for one 8×8 block. Bidirectional motion compensation requires at most a total of eight addition operations, three shift operations, and one 8-bit clipping, which is 768 operations for one block. For the IPB frame structure shown in Fig. 20, it requires 149.3 MOPS for MPEG-2 MP@ML.

In the addition of two-pixel data for interpolation using SIMD-type addition operations, more than 8-bit arithmetic is required. Usually, four parallel 16-bit operations are used for this purpose. To do this, packed 8-bit data has to be converted into 16-bit data. Unidirectional motion-compensation operations for 4-pixel data require two 8-to-16 conversions, four split additions, one split shift, and one packing operation. That means 128 operations for an

8×8 block, which is three times faster than the motion compensation without SIMD-type operations. Bidirectional motion compensation operations for 4-pixel data require four 8-to-16 conversions, eight split additions, three split shifts, and one 8-bit clipping. That means 256 operations for an 8×8 block, or 50 MIPS for MPEG-2 MP@ML, which is also three times faster than the implementation without SIMD-type instructions.

Some multimedia processors [8], [22] have SIMD-type multimedia instructions for this pixel averaging operation. On the other hand, it has been reported [24], [52] that this split-word-type operation can be simulated using nonsplit-word instructions with a special treatment of the carry propagation between each byte in a word.

Another consideration in motion compensation is unaligned data access. Unaligned memory access is needed depending on the motion vector. Instruction cycles for alignment are required for processors that do not have an unaligned-data-access function.

1) Requirements on Memory Access: Motion compensation is an extremely memory-bandwidth-intensive operation. Unidirectional motion compensation for one macroblock requires access to one macroblock of reference frame data, while bidirectional motion compensation requires access to two macroblocks. The address of these reference macroblocks can be random depending on the motion vectors.

Decoding of one P- (B)-frame requires reading one (two) frame(s) and writing one frame of data at most. The size of the frame memory is 126.7 kB ($352 \times 240 \times 1.5$ B) for MPEG-1 and 518 kB ($720\times 480 \times 1.5$ B) for MPEG-2 MP@ML. Motion compensation for one P- (B)-frame requires reading one (two) frame(s) and writing one frame, which is in total 253.4 kB (380 kB) access for MPEG-1 and 1.03 MB (1.55 MB) for MPEG-2 MP@ML. For MPEG-2 MP@ML motion compensation, frame-memory access bandwidth of 24.9 MB/s for reading and 15.6 MB/s for writing are needed at least. In the worst case without cache operation, half-pixel interpolation in motion compensation is required to read four times the data, that is, 99.6 MB/s. Given the second-level (L2) cache size of a typical PC, which is 256–512 kB, it is possible for the cache to contain the input and output frames for MPEG-1 but not for MPEG-2 MP@ML. Therefore, enough access bandwidth to the main memory is required for motion compensation with MPEG-2.

Table 4 shows the cycle time needed for cache memory access with a 200-MHz clock CPU that has a 66-MHz CPU bus and L2 bus (a typical PC at present). In the case of a 64-bit bus with a 32-byte cache line, the number of cycles needed to refill each type of memory is 45 for an FP-DRAM, 36 for an EDO-DRAM, and 27 for an SDRAM.

Software implementation of MPEG-2 motion compensation on microprocessors results in at least 32 cache misses per macroblock for unidirectional prediction and 64 cache misses per macroblock for bidirectional prediction, which is 2.07-M cache misses per second. As each cache miss requires cache-miss penalty cycles for the main memory, we have to suffer, for example, 100-M cache-miss penalty cycles per second for the FP-DRAM main memory system.

Table 4 Cache-Miss Penalty

	Access	Bus Clocks	MPU Clocks
L1 Cache	1-1-1-1	-	4
L2 Cache	3-1-1-1	6	18
FP-DRAM	6-3-3-3	15	45
EDO-DRAM	6-2-2-2	12	36
SDRAM	6-1-1-1	9	27

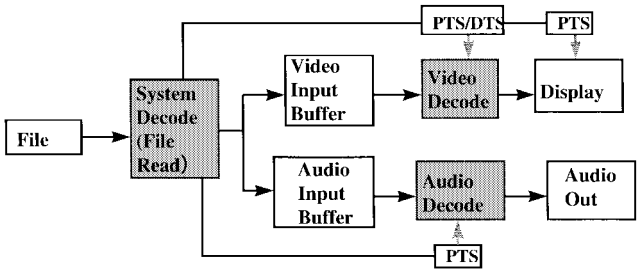


Fig. 22. MPEG A/V decoder system.

D. Real-Time Task Switching and MPEG A/V Decoding

As shown in Fig. 22, decoding of an MPEG-2 bitstream requires video decoding, audio decoding, bitstream demultiplexing, and control of the synchronized playback of audio and video. The playback timing can be controlled by using time stamps in the bitstream. Synchronized playback of audio and video in software decoding requires coordination of these multiple tasks.

It is difficult to control these tasks accurately by time stamps in PC-based software decoding, however, because the precision of the timer used for real-time scheduling in a PC is not high enough. Moreover, the media performance cannot always be guaranteed because of background tasks. Therefore, approximation techniques are used in this software decoding. For example, the audio buffer state is used instead of the timer for scheduling. When the video-processing performance is insufficient, "graceful degradation" by dropping video frames can be used.

E. Summary of MPEG Complexity

The total processing performance needed to realize MPEG-2 MP@ML decoding with and without multimedia instructions is shown in Table 5. The table shows the worst case estimate for several core operations that we have calculated in this section that do not include data transfers, such as memory reading or writing.

In the actual implementation of microprocessor-based software MPEG decoders, there are other performance factors that are not taken into account in the MIPS figures given in Table 5. The following factors may cause the cycle

Table 5 Summary of MPEG Complexity

	MPEG-1 (MIPS)	MPEG-1 (with media inst.)	MPEG-2 (MIPS)	MPEG-2 (with media inst.)
VLD+IQ	13.6	9.9	36.2	26.4
IDCT	42.4	11.9	173.3	48.6
MC	36.5	12.1	149.3	49.8
Total	92.5	33.9	358.8	124.8

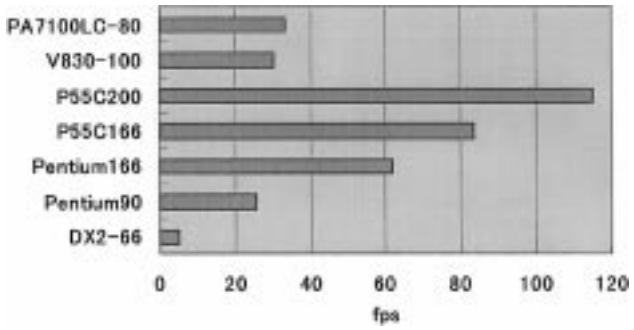


Fig. 23. Performance of MPEG software decoders.

times for actual implementations to differ depending on the systems and bitstreams.

1) The number of operations can be reduced by the distribution of zero coefficients in a DCT block and skipped blocks or macroblocks. It depends on the bitstream, especially the bit rate.
2) The type of motion-compensation prediction will change depending on the macroblock. For example, it is possible to have unidirectional prediction macroblocks in a B-frame, and this may reduce the number of operations compared to the value given in Table 5, depending on the bitstream.
3) As described in this section, the performance of the memory (access time) and the bus (bandwidth) has a great impact on the total decoding performance. Sometimes the stall time due to these factors will exceed the core processing time given in Table 5 [53].
4) The performance and functionality of graphics accelerators also affect the performance in combination with the bus bandwidth.

The performance of actual software MPEG decoders implemented on microprocessors as well as that of media-enhanced microprocessors is shown in Fig. 23. As there are only a few MPEG-2 software decoder implementations [41] reported so far, the values in Fig. 23 are for MPEG-1 software decoders [24], [41], [52]–[56].

As shown in Table 5, the required performance for the core operations of MPEG-1 decoding is 92.5 MIPS on nonmedia-enhanced processors and 33.9 MIPS on media-enhanced processors. In other words, the media instructions reduce the number of instructions by about 63%.

On the other hand, Fig. 23 shows that conventional Pentium microprocessors for a PC system require 105.4 or 80.7 MHz operation to decode 30 frames per second in real time. Comparable media-enhanced processors (Pentium with MMX) for a PC system require 59.9- or 52.1-MHz operation, meaning that the media instructions enhance the decoding performance by only about 35–43%, respectively. One of the reasons is that other factors such as the bit-manipulation performance, the bus/memory performance, and the display performance are less enhanced than the arithmetic performance on media-enhanced microprocessors. Therefore, in addition to arithmetic enhancements such as a multimedia instruction set, the other key functions given in Section II should also be enhanced for media-enhanced microprocessors as well as for the multimedia PC systems.

VI. Future Directions

The performance of microprocessors has increased steadily with the advances in device technology and architectures. However, it is likely to be difficult to continue this trend over the next five years [57]. On the other hand, there is still plenty of room to improve the multimedia performance of microprocessors through progress in device technology, because the multimedia enhancements in current microprocessors have yet fully to utilize the inherent parallelism in multimedia processing algorithms.

Media processors have been introduced as accelerators for host processors and are likely to be threatened by the increased multimedia performance of host processors. The accelerator board approach suffers from a cost disadvantage and tends to be less compatible. Moreover, the disadvantages on host processor software processing seem to be surmountable and are likely to be solved if PC systems employ architectures that enhance the memory/bus performance for multimedia processing and an advanced operating system with real-time support. In this case, the only way for media processors to survive will be to deliver better cost performance at a lower cost and consume less power.

Multimedia processing seems likely to progress tremendously for high-quality applications as well as portable

applications. However, there are still problems to overcome. This section describes these problems, especially the memory-access bottleneck and algorithms that cannot be easily parallelized, and discusses a possible solution that uses both an on-chip frame buffer and reconfigurable logic. The "system on a chip" designed for use in a portable system is described as an example of such a solution.

A. Trends Toward Higher Quality Multimedia

Software solutions for MPEG-1 [24], [52], [54], [55] became available for use on PC's within five years after MPEG-1 decoder ASIC large-scale integration (LSI) appeared as the first generation of multimedia LSI. Microprocessor-based solutions for the HDTV decoding and MPEG-2 encoding currently realized by ASIC LSI's seem likely to be realized within the next five years. This seems possible if the media-processing performance can be increased by a factor of ten. However, two problems—the memory-access bottleneck and the acceleration of algorithms not easily parallelized—must first be solved.

MPEG decoding algorithms require a huge amount of memory access through a frame buffer in both the decoding process and the display process. Memory access is already a bottleneck in MPEG-2 software decoding with today's typical PC systems, and this bottleneck will probably remain a big problem when the performance of the arithmetic operation is increased by a factor of ten. An on-chip frame buffer is one way to improve the memory-access performance, but this will require a 12-MB frame buffer memory for HDTV. Use of a DRAM instead of SRAM makes sense in this situation if efficient use of the chip area is a priority.

Variable-length encoding/decoding using Huffman code is hard to explore parallelism in the implementation, compared with other parts of the MPEG algorithm. In a software solution of MPEG-2 decoding on a PC with a media-enhanced microprocessor, variable-length decoding with inverse quantization accounts for about 25% of the total video-decoding time for a 4-Mbit/s stream [41]. Therefore, variable-length decoding by software is unlikely to be feasible when the bit rate reaches more than 20 Mbit/s (HDTV). To solve this problem, special decoding circuits will be required. How those decoding circuits can be incorporated into general-purpose microprocessors while maintaining good balance will be the next challenge.

B. Trends Toward Portable Multimedia

Another area where we can expect fast growth is mobile multimedia communication. Video communication using wireless phone systems is made possible through the use of video codecs using DSP's [31]. Standardization of mobile video-compression algorithms such as H.324 for mobile use and MPEG-4 is likely to accelerate this trend. Moreover, next-generation mobile-phone systems, such as wide-band code division multiple access, will provide 384–2-Mbit/s lines, which will be able to deliver higher quality video communications.

The multimedia performance required for this application will be within the range of today's media processors. However, low power consumption is required. The voice-codec LSI's currently used in mobile phones consume around 0.1 W. On the other hand, media processors (which consume less power than microprocessors) require more than 1 W.

Low power consumption will also be important in input/output devices such as charge-coupled-device cameras and liquid-crystal displays. The power consumption of media processors can probably be reduced to one-tenth of its current level by reducing the supply voltage from 3.3 to 1 V. To reduce the power consumption of the total system, however, frame-buffer memory access to the off-chip memory will have to be reduced. An on-chip frame buffer can be used if a low-cost, low-power on-chip memory is enabled. Whether an SRAM or a DRAM is used for this purpose will depend on a tradeoff between lower power dissipation and higher integration.

Another way to reduce power consumption is to use ASIC circuits for part of the function. ASIC solutions for motion estimation, which consume 50–90% of the computation in video encoding, and display processing functions in video decoders enable use of a lower clock frequency, which helps reduce power consumption. The implementation of special circuits as a coprocessor is also a possible solution to this problem. Incorporating a reconfigurable logic in general-purpose processors may also be possible. However, efficient use of the chip area and low power consumption in the reconfigurable logic will be required for this purpose.

C. Toward a Multimedia System on Silicon

To make portable multimedia systems such as a portable video-communication terminal a reality, multimedia processing functions such as video and audio codec must be integrated. Moreover, microprocessor functions for the user interface and ASIC functions, such as a wireless modem, should also be integrated in one chip. With the progress currently being made in integration density, it may soon be possible to integrate these functions in a single system.

There are two approaches to this goal. One is to integrate different kinds of cores, such as microprocessor and DSP, and the other is a software solution on an integrated very-large-scale-integration processor. In portable systems especially, it is desirable to divide the functions into several cores whose power consumption can be controlled depending on the state of the communication. On the other hand, it is also advantageous to have the flexibility of software solutions because of the different standards used in communication and multimedia processing. Therefore, balancing a core-based solution and the flexibility of a software solution will be a challenge in the development of future systems on silicon.

VII. Conclusion

In this paper, we have examined multimedia processors that realize multimedia processing through the use of software. The performance and functional requirements of multimedia processing, such as MPEG video decoding, was described. The functional requirements include those for bit manipulation, arithmetic operations, memory access, stream data I/O, and real-time switching. Then, programmable processors for multimedia processing were classified into media-enhanced microprocessors (CISC and RISC), embedded microprocessors, DSP's, and media processors. The architectures of these multimedia processors were introduced, and the performance for the five functional requirements was examined. Especially, the media-enhanced microprocessors and media processors were compared with respect to the acceleration of PC multimedia. As the most enhanced part of media-enhanced microprocessors, the data path for multimedia processing and the design of multimedia instruction sets were also described. The effect of these media instructions on media-enhanced microprocessors was discussed using an MPEG software decoder implementation as an example. The performance improvements in media operations and those of software decoder implementations were compared, which showed the difference due to other performance factors such as memory access. These examinations indicate the requirements for future multimedia processors, especially for high-quality multimedia such as HDTV as well as portable multimedia applications, which include integration of the DRAM and reconfigurable logic.

References

[1] "Video coding for low bitrate communication," International Telecommunications Union, Geneva, Switzerland, ITU-T Recommendation H.263, 1995.

[2] "Information technology—Generic coding of moving pictures and associate audio information: Video," International Standards Organization, ISO/IEC JTC1/SC29/WG11, Recommendation H.262, ISO/IEC 13818-2, 1994.

[3] T. Nishitani, "Trend and perspective on domain specific programmable chips," in *Proc. IEEE Workshop Signal Processing Systems (SiPS'97)*, Nov. 1997, pp. 1–8.

[4] J. Goto, K. Ando, T. Inoue, M. Yamashima, H. Yamada, and T. Enomoto, "250 MHz BiCMOS super-high-speed video signal processor (S-VSP) ULSI," *IEEE J. Solid-State Circuits*, vol. 26, no. 12, pp. 1876–1884, 1991.

[5] K. Aono, M. Toyokura, A. Ohtani, H. Kodama, and K. Okamoto, "A video digital signal processor with a vector-pipeline architecture," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1886–1893, 1992.

[6] P. Pirsh, N. Demassieux, and W. Gehrke, "VLSI architecture for video compression-a survey," *Proc. IEEE*, vol. 83, no. 2, pp. 220–246, Feb. 1995.

[7] P. Foley, "The Mpact media processor redefines the multimedia PC," in *Proceedings of Compcon*. New York: IEEE Computer Science Press, 1996, pp. 311–318.

[8] S. Rathnam and G. Slavenburg, "An architectural overview of the programmable multimedia processor, TM-1," in *Proceedings of Compcon*. New York: IEEE Computer Science Press, 1996, pp. 319–326.

[9] "i860™ microprocessor family programmers reference manual," Intel corporate literature, 1991.

[10] K. Deifendorff and M. Allen, "Organization of the Motorola 88110 superscalar RISC microprocessor," *IEEE Micro Mag.*, vol. 12, pp. 40–63, Apr. 1992.

[11] R. B. Lee, "Accelerating multimedia with enhanced microprocessors," *IEEE Micro Mag.*, vol. 15, pp. 22–32, Apr. 1995.

[12] M. Tremblay, D. Greenley, and K. Normoyle, "The design of the microarchitecture of UltraSparc™-I," *Proc. IEEE*, vol. 83, pp. 1653–1663, Dec. 1995.

[13] A. Peleg and U. Weiser, "MMX technology extension to the Intel architecture," *IEEE Micro Mag.*, vol. 16, pp. 42–50, Aug. 1996.

[14] D. Brinthaupt, D. L. Letham, V. Maheshwari, J. H. Othmer, R. R. Spivak, B. Edwards, C. Terman, and N. Weste, "A video decoder for H.261 video teleconferencing and MPEG stored interactive video applications," in *Dig. Tech. Papers ISSCC'93*, Feb. 1993, pp. 34–35.

[15] D. Galbi, E. Bird, S. Bose, E. Chai, Y.-N. Chang, P. Dermy, N. Fernando, J.-G. Fritsch, E. Hamilton, B. Hu, E. Hua, F. Liao, M. Lin, M. Ma, E. Paluch, S. Purcell, H. Yanagi, S. Yang, M. Chow, T. Fujii, A. Fujiwara, H. Goto, K. Ibara, S. Isozaki, J. Jao, I. Kaneda, M. Koyama, T. Mineo, I. Miyashita, G. Ono, S. Otake, A. Sato, A. Sugiyama, K. Tagami, K. Tsuge, T. Udagawa, K. Yamasaki, S. Yasura, and T. Yoshimura, "An MPEG-1 audio/video decoder with run-length compressed antialiased video overlays," in *Dig. Tech. Papers ISSCC'95*, Feb. 1994, pp. 286–287.

[16] M. Harrand, M. Henry, P. Chaisemartin, P. Mougeat, Y. Durand, A. Tournier, R. Wilson, J.-C. Heriuison, J.-C. Longchambon, J.-L. Bauer, M. Runts, and J. Bulone, "A single chip videophone video encoder/decoder," in *Dig. Tech. Papers ISSCC'95*, Feb. 1994, pp. 292–293.

[17] K. Nadehara, I. Kuroda, M. Daito, and T. Nakayama, "Low-power multimedia RISC," *IEEE Micro Mag.*, vol. 15, pp. 20–29, Dec. 1995.

[18] NEC Corporation, "μPD77016 user's manual," 1992.

[19] "Coding of moving pictures and associated audio for digital storage media up to 1.5 Mbits/s," International Standards Organization, ISO/IEC JTC1 CD 11172, 1992.

[20] M. R. Choudhury and J. S. Miller, "A 300 MHz CMOS microprocessor with multi-media technology," in *Dig. Tech. Papers ISSCC'97*, Feb. 1997, pp. 170–171.

[21] M. Johnson, *Superscalar Microprocessor Design*. Englewood Cliffs, NJ: Prentice-Hall, 1991.

[22] R. B. Lee, "Subword parallelism with MAX-2," *IEEE Micro Mag.*, vol. 16, pp. 51–59, Aug. 1996.

[23] P. Bannon and Y. Saito, "The alpha 21164PC microprocessor," in *Proceedings of Compcon*. New York: IEEE Computer Science Press, 1997, pp. 20–27.

[24] K. Nadehara, H. J. Stolberg, M. Ikekawa, E. Murata, and I. Kuroda, "Real-time software MPEG-1 video decoder design for low-cost, low-power applications," in *Proc. IEEE VLSI Signal Processing IX*, Oct. 1996, pp. 438–447.

[25] C. Hansen, "Architecture of a broadband mediaprocessor," in *Proceedings of Compcon*. New York: IEEE Computer Science Press, 1996, pp. 334–340.

[26] Y. Oshima, B. J. Sheu, and S. H. Jen, "High-speed memory architectures for multimedia applications," *IEEE Circ. Devices Mag.*, vol. 13, pp. 8–13, Jan. 1997.

[27] C. Abbott, H. Massalin, K. Peterson, T. Karzes, L. Yamano, and G. Kellogg, "Broadband algorithms with the MicroUnity mediaprocessor," in *Proceedings of Compcon*. New York: IEEE Computer Science Press, 1996, pp. 349–354.

[28] T. Robinson, C. Hansen, B. Herndon, and G. Rosseel, "Multi-gigabyte/sec DRAM's with the MicroUnity mediachannel interface," in *Proceedings of Compcon*. New York: IEEE Computer Science Press, 1996, pp. 378–381.

[29] E. A. Lee, "Programmable DSP architectures: Part I," *IEEE Acoust., Speech, Signal Processing Mag.*, pp. 4–19, Oct. 1988.

[30] E. A. Lee, "Programmable DSP architectures: Part II," *IEEE Acoust., Speech, Signal Processing Mag.*, pp. 4–14, Jan. 1989.

[31] Y. Naito and I. Kuroda, "H.263 mobile video codec based on a low power consumption digital signal processor," in *Proc. ICASSP 1998*, to be published.

[32] K. Guttag, R. J. Gove, and J. R. Van Aken, "A single-chip multiprocessor for multimedia: The MVP," *IEEE Comput. Graph. Applicat. Mag.*, vol. 12, pp. 53–64, Nov. 1992.

[33] "Terminal for low bitrate multimedia communication," International Telecommunications Union, Geneva, Switzerland, ITU-T Recommendation H.324: 1995.

[34] J. Golston, "Single-chip H.324 videoconferencing," *IEEE Micro Mag.*, vol. 16, pp. 21–33, Aug. 1996.

[35] D. Epstein, "IBM extends DSP performance with Mfast," *Microprocessor Rep.*, vol. 9, no. 16, pp. 1, 6–8, Dec. 1995.

[36] M. Tremblay, J. M. O'Connor, V. Narayanan, and L. He, "VIS speeds new media processing," *IEEE Micro Mag.*, vol. 16, pp. 10–20, Aug. 1996.
[37] T. Arai, K. Suzuki, and I. Kuroda, "V830R/AV: Embedded multimedia superscalar RISC processor with rambus interface," in *Proc. HOTCHIPS IX*, Aug. 1997, pp. 177–189.
[38] L. Gwennap, "Digital, MIPS add multimedia extensions," *Microprocessor Rep.*, vol. 10, no. 15, pp. 24–28, Nov. 1996.
[39] E. Murata, T. Motizuki, and I. Kuroda, "Fast 2D IDCT implementation via split ALU operations," (in Japanese) *IEICE Jpn.*, NC D-11-95, Mar. 1997.
[40] K. Patel, B. C. Smith, and L. A. Rowe, "Performance of a software MPEG video decoder," in *Proc. ACM Multimedia Conf.*, 1993.
[41] M. Ikekawa, D. Ishii, E. Murata, K. Numata, Y. Takamizawa, and M. Tanaka, "A real-time software MPEG-2 decoder for multimedia PC's," in *ICCE Dig. Tech. Papers*, June 1997, pp. 2–3.
[42] D. Ishii, M. Ikekawa, and I. Kuroda, "Parallel variable length decoding with inverse quantization for software MPEG-2 decoders," in *Proc. IEEE Workshop Signal Processing Systems (SiPS97)*, Nov. 1997, pp. 500–509.
[43] B. G. Lee, "A new algorithm to compute the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-32, pp. 1243–1245, Dec. 1984.
[44] Y. Arai, T. Agui, and M. Nakajima, "A fast DCT-SQ scheme for images," *Trans. IEICE*, vol. E71, no. 11, p. 1095, Nov. 1988.
[45] W. B. Pennbecker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. New York: Van Nostrand, 1993.
[46] W. A. Chen, C. H. Smith, and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform," *IEEE Trans. Commun.*, vol. COM-25, no. 9, pp. 1004–1011, 1977.
[47] C. Loeffler, A. Ligtenberg, and G. S. Moschytz, "Practical fast 1-D DCT algorithms with 11 multiplications," in *Proc. Int. Conf. Acoustics, Speech, and Signal Processing 1989 (ICASSP'89)*, pp. 988–991.
[48] A. C. Hung and T. H.-Y. Meng, "Statistical inverse discrete cosine transforms for image compression," in *Proc. SPIE Digital Video Compression on Personal Computers: Algorithms and Technologies*, A. A. Rodriguez, Ed., vol. 2187, 1994.
[49] M. Yoshida, H. Ohtomo, and I. Kuroda, "A new generation 16-bit general purpose programmable DSP and its video rate application," in *Proc. IEEE VLSI Signal Processing VI*, Oct. 1993, pp. 93–101.
[50] *IEEE Standard Specification for the Implementation of 8 by 8 Inverse Discrete Cosine Transform*, IEEE Standard 1180–1990, 1990.
[51] "Inverse discrete cosine transform," International Standards Organization, ISO/IEC 13818-2:1996/Cor.2 1996(E) in Annex A, 1996
[52] S. Eckart, "High performance software MPEG video player for PC's," in *Proc. SPIE*, San Jose, CA, 1995, vol. 2419, pp. 446–454.
[53] Intel Corp. (Apr. 1996). Choosing a platform architecture for cost effective MPEG-2 video playback (white paper). [Online]. Available WWW: www.intel.com.
[54] V. Bhaskaran, K. Konstantinides, R. B. Lee, and J. P. Beck, "Algorithmic and architectural enhancements for real-time MPEG-1 decoding on a general purpose RISC workstation," *IEEE Trans. Circuits, Syst., Video Technol.*, vol. 5, pp. 380–386, Oct. 1995.
[55] Intel Corp. (Dec. 1995). Designing a low cost, high performance platform for MPEG-1 video playback (white paper). [Online]. Available WWW: www.intel.com.
[56] H. Bheda and P. Srinivasan, "A high-performance cross-platform MPEG decoder," in *Proc. SPIE*, vol. 2187, 1994, pp. 241–248.
[57] H. Sasaki, "Multimedia complex on a chip," in *Dig. Tech. Papers ISSCC96*, Feb. 1996, pp. 16–19.

Ichiro Kuroda (Member, IEEE) received the B.E. degree in mathematical engineering and instrumentation physics from the University of Tokyo, Tokyo, Japan.

He currently is a Principal Researcher with the Signal Processing Technology Group, C&C Media Research Laboratories, NEC Corporation, Kawasaki, Japan. He has been working in the area of very-large-scale-integration signal processing, especially on programmable digital signal-processing chips, tools, and application design.

Mr. Kuroda is member of the IEEE Signal Processing Society's Technical Committee on Design and Implementation of Signal Processing Systems.

Takao Nishitani (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees from Osaka University, Osaka, Japan, in 1971, 1973, and 1992, respectively.

He joined NEC Corporation, Kawasaki, Japan, in 1973, where he currently is a Deputy General Manager of NEC Silicon Systems Research Laboratories. He developed the world's first commercially available programmable digital signal-processing (DSP) chip (NEC-7720) in 1980. He contributed to the world standard ITU-T 32-kbit/s adaptive differential pulse code modulation (ADPCM) speech-compression algorithm on tandem transcodings without distortion accumulation. He also contributed to the MPEG-1 audio-compression algorithm on preecho elimination in transform coding approaches. Since he developed programmable DSP chips in 1980, he has been engaged in the architecture design of many NEC signal-processing chips, such as a 32-bit floating-point programmable DSP chip, 32-kbit/s ADPCM LSI codec, parallel programmable DSP system for video signal processing (VSP and VISP), and so forth. He has managed the recent chip development of the MPEG-1 audio/video decoder chip and MPEG-2 single chip video encoder, as well as the MPEG-2 decoder chip and multimedia reduced instruction set computer chips. He is a member of the editorial board of the Institute of Electronics, Information, and Communications Engineers of Japan.

Dr. Nishitani received the 1989 Research Award from the Japanese government for his research of very-large-scale-integration signal processors. He currently is a Distinguished Lecturer of the IEEE Circuits and Systems Society.