21

PROCESSOR ARCHITECTURES FOR MULTIMEDIA

Borko Furht
Florida Atlantic University
Department of Computer Science and Engineering
Boca Raton, Florida 33431

Abstract. In this chapter, we present contemporary VLSI processor architectures that support multimedia applications. We classified these processors into two groups: dedicated multimedia processors, which perform dedicated multimedia functions, such as MPEG encoding or decoding, and general-purpose processors that provide support for multimedia. Dedicated multimedia processors use either function-specific architectures with limited flexibility but higher speed and efficiency, or programmable architectures with increased flexibility. Both architectures are exploring parallelism inherent in video and image applications by applying single-instruction, multiple-data (SIMD) or/and very-large-instruction-word (VLIW) concepts. Advanced general-purpose processors provide the support for multimedia by incorporating new multimedia instructions and executing them in parallel by using the SIMD coprocessor approach. A survey of these processor architectures and their performance is presented in this chapter.

1. INTRODUCTION AND CLASSIFICATION

During the last few years we have witnessed the process of "poising" for multimedia: from PC and workstation manufacturers (multimedia PCs and workstations), add-in-board vendors (video and audio capture and playback cards), and silicon vendors (compression and graphics chips), to operating systems designers (OS support for multimedia) and software creators (authoring tools and a variety of multimedia applications). The last players to enter this poising game have been microprocessor designers.

In this chapter, we present a survey of processor architectures designed to support multimedia applications. Designs of these architectures range from fully custom to fully programmable dedicated architectures and to general-purpose processor architectures with an extensive support for multimedia. The classification of these architectures is shown in Figure 1.

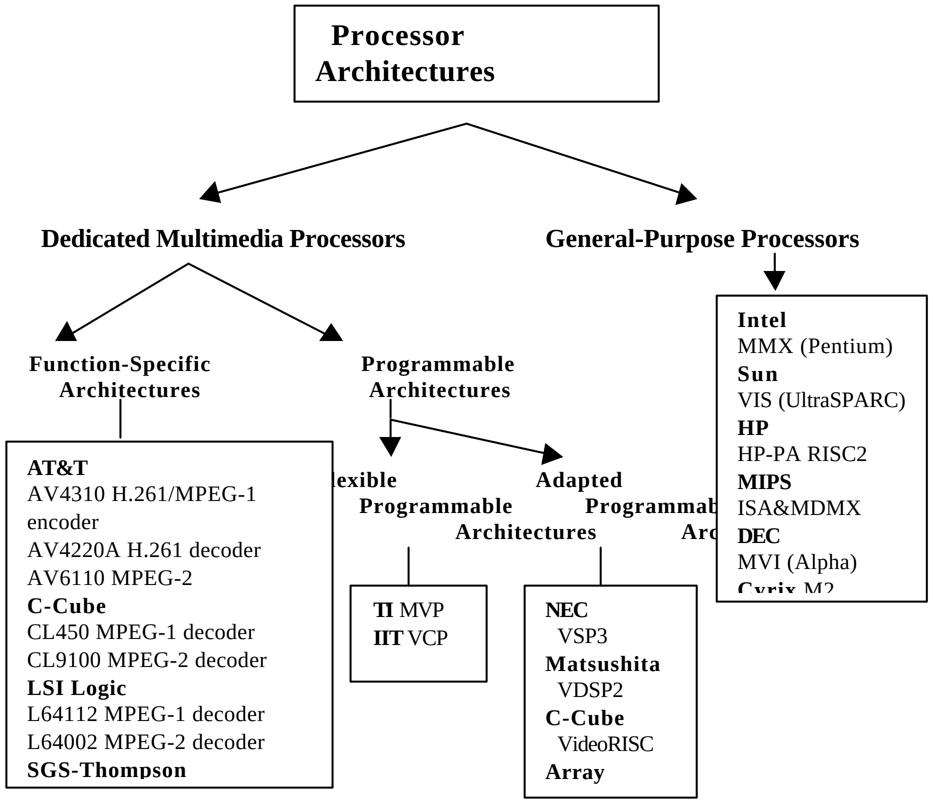

Figure 1. Classification of processor architectures that support multimedia.

Dedicated multimedia processors are typically custom designed architectures intended to perform specific multimedia functions. These functions usually include video and audio compression and decompression, and in this case these processors are referred to as video codecs. In addition to support for compression, some advanced multimedia processors provide support for 2D and 3D graphics applications. Designs of dedicated multimedia processors range from fully custom architectures, referred to as function specific architectures, with minimal programmability, to fully programmable architectures. Furthermore, programmable architectures can be classified into flexible programmable architectures, which provide moderate to high flexibility, and adapted programmable architectures, which provide an increased efficiency and less flexibility [1]. The dedicated multimedia processors use a variety of architectural schemes from multiple functional units and a RISC or DSP (digital signal processor) core processors to multiple processor schemes. Furthermore, the latest dedicated processors use single-instruction-multiple-data (SIMD) and very-long-instruction-word (VLIW) architectures, as well as some hybrid schemes. These architectures are presented in Section 3.

General-purpose (GP) processors provide support for multimedia by including multimedia instructions into the instruction set. Instead of performing specific multimedia functions (such as compression and 2D/3D graphics), GP processors provide instructions specifically created to support generic operations in video processing. For example, these instructions include support for 8-bit data types (pixels), efficient data addressing and I/O instructions, and even instructions to support motion estimation. The latest processors, such as MMX (Intel), VIS (Sun) and MAX-2 (HP), incorporate some types of SIMD architectures, which perform the same operation in parallel on multiple data elements.

2. COMPLEXITY OF MULTIMEDIA FUNCTIONS

In video and signal processing applications, a measure of algorithmic complexity is the total number of operations per second, expressed in MOPS (million operations per second), or GOPS (giga operations per second). This measure incorporates the total number of primitive operations needed to perform specific functions, and includes data load and store operations as well as arithmetic and logic operations on data elements.

For purposes, we present the calculation of the complexity (adapted from [2]) for an MPEG-2 decoder, shown in Figure 2. We assume the encoded bit rate for the input bit sequence is 4 Mbps. Assuming that the average symbol size is 4 bits, the average rate is then 1 million symbols/second.

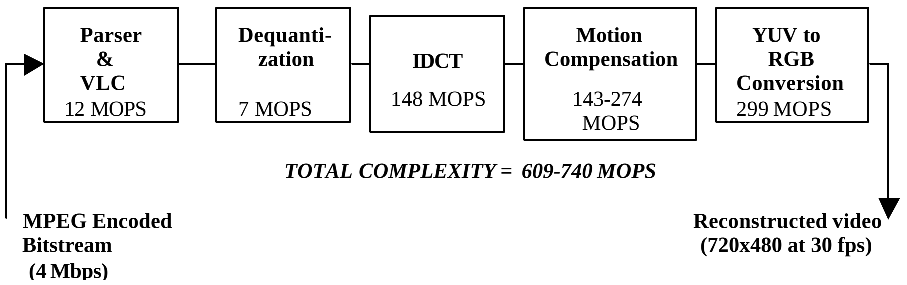

Figure 2. MPEG decoder and the complexity of its blocks.

The input video is 720x480 at 30 fps, encoded in 4:2:0 YUV format. In 4:2:0 format, the Y component contains 720x480 pixels, while U and V components have 320x240 pixels. Total number of 8x8 blocks in each frame is 90x60 = 5,400 (in Y) and 45x30 = 1,350 (in U and V). This gives 8,100 blocks in each frame, and total of a 8,100 x 30 sec = 243,000 blocks per second. The

MPEG sequence used for this calculation comprises the group of pictures (GOP) consisting of 1 I-, 4 P-, and 10 B-frames.

Block 1: Bit stream parser and variable length (Huffman) decoder
The decoding of 1 symbols requires the following operations:

- 1 compare, 1 subtract, and 1 shift for grabbing a variable number of bits from data buffer (total 3 operations),
- 1 load, 2 shifts, and 1 mask for searching for a code in a code table (total 4 operations), and
- 1 compare, 1 subtract,1 shift, 1 add, and 1 mask for updating the data buffer (total 5 operations).

This gives total of 12 operations per symbol, and total of 12 x 1 million symbols = 12 MOPS for this block.

Block 2: Dequantization
The dequantization, applied to each non-zero DCT coefficient, requires the following operations:

- 1 load for reading the quantizer scale matrix (total 1 operation),
- 2 integer multiplications and 1 division (by a constant of 8 or 16) for dequantization (total 3 operations), and
- 2 logical, 1 compare, 1 shift, and 1 add for the oddification (total 5 operations).

The total count for one DCT coefficient is 9 operations. Among 1 million symbols, we assume that 80% are coded DCT coefficients. This gives the total complexity of the quantization block: 9 x 800,000 = 7.2 MOPS.

Block 3: Inverse Discrete Cosine Transform (IDCT)
There are many different implementations for the inverse DCT. For illustration purposes we assume the fast inverse DCT algorithm, described in [3], that requires the following operations:

- 464 additions, 80 integer multiplications, and 64 stores for each 8x8 block (total 608 operations per block).

Then, the total complexity of the IDCT block becomes 243,000 blocks x 608 = 147.7 MOPS. Since not every block is coded in a P- or B-frame, this is the maximum complexity for using the fast IDCT.

Block 4: Motion Compensation
The worst-case scenario is that all blocks in a P-frame are motion compensated with 4-pixel interpolation and all blocks in a B-frame are motion compensated with 8-pixel interpolation from two predictor frames. For an 8x8 block in a P-frame, the following operations are required:

- 8x9x2 = 144 load operations for reading blocks with both horizontal and vertical overlapping (total 144 operations),
- 3 additions and 1 shift for interpolating one predictor pixel (total 4x64 = 256 operations), and
- 1 load, 1 addition, 2 compares, 1 assignment, and 1 store for reconstructing the pixel (total 6x64 = 384 operations).

The total number of operations for each block is 784. Because there are 8 P-frames in a one-second sequence, the total complexity for P-frames is: 8 x 8,100 blocks x 784 = 50.8 MOPS.

For each 8x8 block in a B-frame, the following operations are required:

- 2x(8x9x2) = 288 operations for reading blocks from two predictor frames (total 288 operations),
- 3 additions and 1 shift for interpolating one predictor pixel from the first frame (total 4x64 = 256 operations),
- 3 additions and 1 shift for interpolating one predictor pixel from the second frame (total 4x64 = 256 operations),
- 2 additions and 1 shift for forming the final pixel predictors (total 3x64 = 192 operations), and
- 1 load, 1 addition, 2 compares, 1 assignment, and 1 store for reconstructing the pixel with clipping (total 6x64 = 384 operations).

The total number of operations for each block is 1,376. There are total of 20 B-frames in a one-second sequence and, therefore, the total complexity for B-frames is 20 x 8,100 blocks x 1,376 = 222.9 MOPS. Adding complexities for P- and B-frames, the complexity for motion compensation becomes 273.7 MOPS.

The best case scenario for motion compensation is when all P- and B-frame blocks have no pixel interpolation. Then, the total number of operations is 143.1 MOPS.

<u>*Block 5: YUV to RGB Color Conversion*</u>

The conversion from YUV to RGB color format, based on CCIR-601 standard, can be performed using the following formula:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.1644 & 0 & 1.5966 \\ 1.1644 & -0.3920 & -0.8132 \\ 1.1644 & 2.0184 & 0 \end{bmatrix} \begin{bmatrix} Y-16 \\ U-128 \\ V-128 \end{bmatrix} \tag{1}$$

where Y and U/V are clipped prior to transform to the ranges [16,240] and [16,235], respectively.

The following operations are required for each pixel:

- 1.5 loads for reading YUV (total 1.5 operation per pixel),
- 1.5 subtractions, 2.5 compares, and 3 assignments for YUV clipping (total 7 operations per pixel),
- 2 integer multiplications, 3.25 additions, and 3 shifts for the transformation (total 8.25 operations per pixel),
- 6 compares and 3 assignments for RGB clipping (total 9 operations per pixel), and
- 3 stores for writing RGB (total 3 operations per pixel).

The total number of operations per each pixel is 28.75. In the 4:2:0 YUV format, there are 4 Ys, 1 U, and 1 V for each pixel. The total number of pixels in one second becomes 10.4 million, which gives the total complexity for YUV to RGB color conversation 28.5 x 10.4 million = 299 MOPS.

In summary, the complexity of the analyzed MPEG decoder is in the range from 609.0 to 739.6 MOPS, as indicated in Figure 2.

Table 1 shows complexities of various H.261 and MPEG encoders and decoders reported in the literature. They differ from one to another due to different implementations of DCT and IDCT algorithms, search algorithms for motion estimation, and formulas used for RGB to YUV transformations.

Table 1. MOPS Requirements for a variety of (a) H.261, and (b) MPEG Encoders and Decoders, Reported in the Literature [4,5,6]

H.261 CODECS	**Complexity of Encoders [MOPS]**	**Complexity of Decoders [MOPS]**
CIF format at 30 fps Fast implementation of DCT Logarithmic search for motion estimation [5]	**968**	**198**
CIF at 15 fps Exhaustive motion estimation algorithm [4]	**1,240-1,320**	**220-315**
CIF at 30 fps Exhaustive motion estimation algorithm [4]	**2,480-2,640**	**440-630**
CIF at 30 fps Logarithmic search for motion estimation [6]	Total Encoder/Decoder **1,193**	Total Encoder/Decoder **1,193**

	Complexity of Encoders [MOPS]			**Complexity of Decoders [MOPS]**		
MPEG CODECS [5]	**SIF 352x240**	**CCIR 601 720x486**	**HDTV 1440x1152**	**SIF 352x240**	**CCIR 601 720x486**	**HDTV 1440x1152**
No B-frames	**738**	**3,020**	**14,498**	**96**	**395**	**1,898**
20% B-frames	**847**	**3,467**	**16,645**	**101**	**415**	**1,996**
50% B-frames	**1,011**	**4,138**	**19,865**	**108**	**446**	**2,143**
70% B-frames	**1,120**	**4,585**	**22,012**	**113**	**466**	**2,241**

MOPS requirements for a variety of multimedia functions are estimated and presented in Figure 3. In the same figure, the current trends in computing power of GP processors, programmable digital signal processors, and programmable video processors are plotted [5],[7]. It can be concluded that it is feasible to implement MPEG-1 or MPEG-2 decoders using GP or DSP processors. However, the encoder requirements, which are more than 1000 MOPS, are still outside of the complexity of GP processors and, therefore, dedicated multimedia processors must be designed.

3. DEDICATED MULTIMEDIA PROCESSORS

In designing dedicated multimedia processors, the selection of architectures depends on the speed requirements of the target function and the constraints on circuit integration, performance, power requirements, and cost. In order to assess and evaluate various architectures, the well-known AT-product is used [1]. Efficiency of an architecture (E) is defined as:

$$E = \frac{1}{Asi \times Tp} \tag{2}$$

where: Asi is the required silicon area for a specific architecture under evaluation, and Tp is the effective processing time for one sample.

A comprehensive evaluation of dedicated multimedia processors can be found in [1]. Dedicated multimedia processors, presented in this section, are based on function specific architectures and programmable architectures.

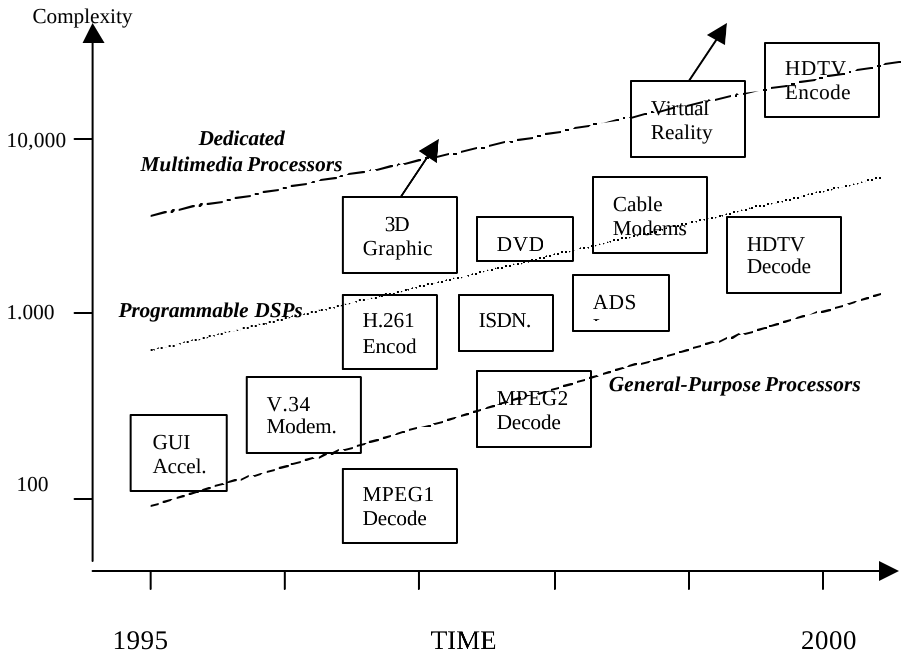

Figure 3. MOPS requirements for various multimedia functions and the current trends in computing power.

3.1 FUNCTION-SPECIFIC ARCHITECTURES

Function-specific multimedia architectures provide limited, if any, programmability, because they use dedicated architectures for a specific encoding or decoding standard. However, their efficiency and speed are typically better compared to programmable architectures. The silicon area optimization achieved by function-specific architectures allows lower production cost.

Regardless of implementation details, the general design theme for dedicated multimedia processors consists of using

- a DSP or RISC core processor for main control, and

- special hardware accelerators for the DCT, quantization, entropy encoding, and motion estimation.

A block diagram of a typical function-specific architecture for a video encoder is shown in Figure 4. In the first generation of function-specific video processors, each of these functions was implemented in one chip, and a chipset was necessary to create the system for encoding or decoding. However, the next generations of function-specific architectures integrate all these functions in a single VLSI chip.

Some popular commercially available dedicated function specific video processors are listed in Figure 1.

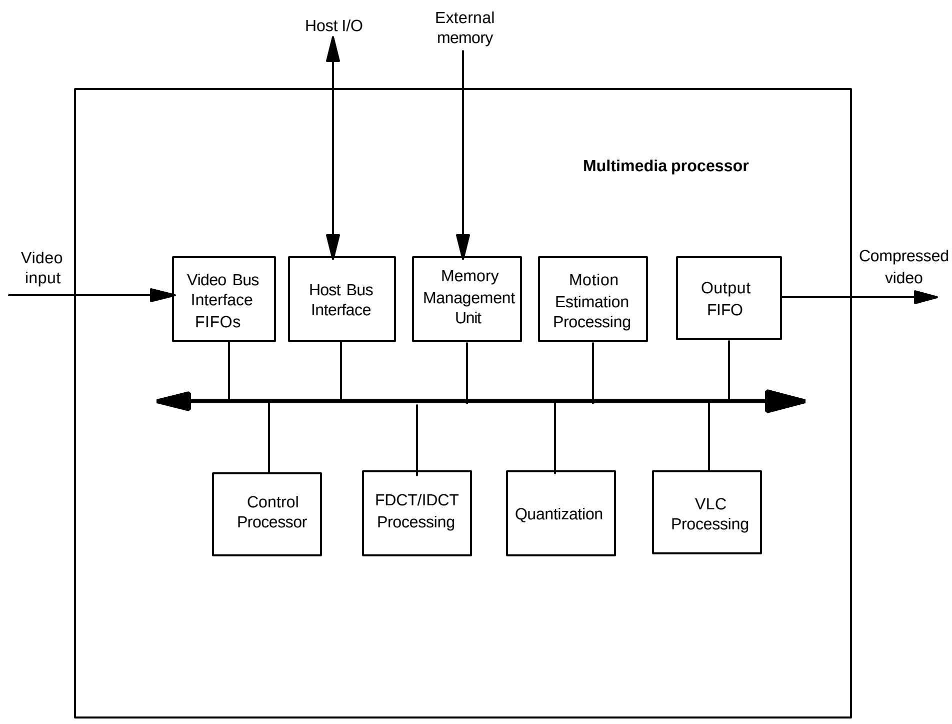

Figure 4. Block diagram of a typical function-specific architecture for a video encoder. The dedicated processors (or functional units) are used for various operations, such as DCT, quantization, variable length coding (VLC), motion estimation, etc.

3.2 PROGRAMMABLE DEDICATED ARCHITECTURES

In contrast to function oriented approach with limited flexibility, programmable architectures enable the processing of different tasks under software control. The main advantage of programmable architectures is the increased flexibility. Changes of architectural requirements, such as changes of algorithms or an extension of the application domain, can be handled by software changes.

On the other hand, programmable architectures incur a higher cost for design and manufacturing, since additional hardware for program control is required. In addition, programmable architectures require software development for the application. Video coding applications require real-time processing of the image data and, therefore, parallelization strategies have to be applied.

Two alternative programmable architectures include: (a) flexible programmable architectures and (b) adapted programmable architectures.

3.2.1 Flexible Programmable Architectures

Flexible programmable architectures, with moderate to high flexibility, are based on coprocessor concept as well as parallel datapaths and deeply pipelined designs. An example of a commercially available video processor, based on flexible programmable architecture, is TI's Multimedia Video Processor (MVP) TMS320C80 [6]. The MVP combines a RISC master processor and four DSP processors in a crossbar-based SIMD shared-memory architecture, as shown in Figure 5.

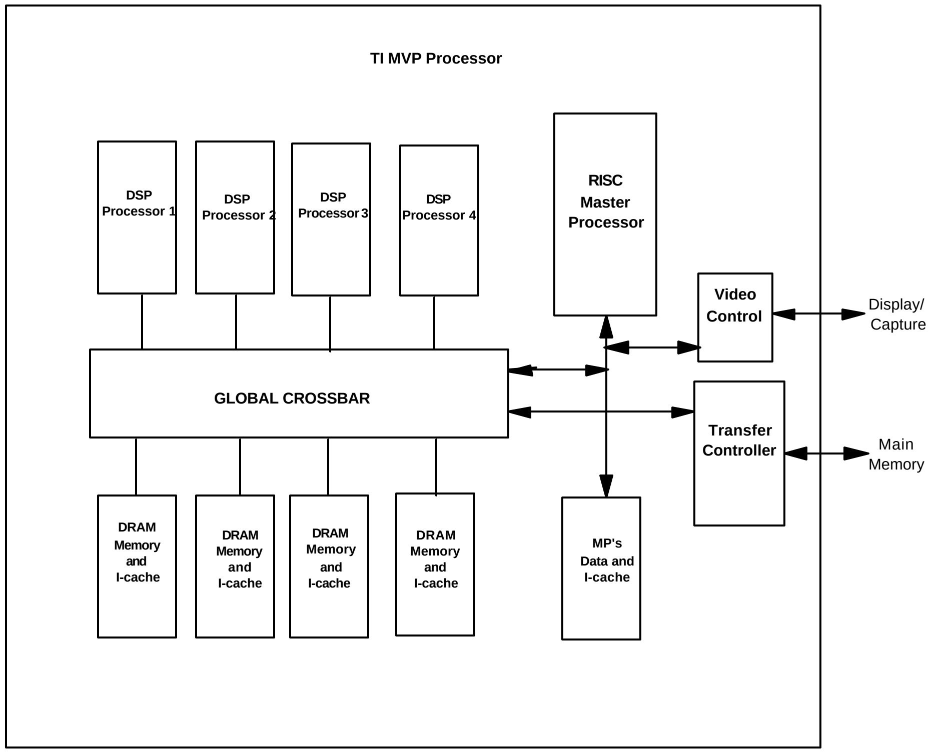

Figure 5. Block diagram of TI's Multimedia Video Processor.

The master processor can be used for control, floating-point operations, audio processing, or 3D graphics transformations. Each DSP performs all the typical operations of a general-purpose DSP and can also perform bit-field and multiple-pixel operations. Each DSP has multiple functional elements (multiplier, ALU, local registers, a barrel shifter, address generators, and a program-control flow unit), all controlled by very long 64-bit instruction words (VLIW concept). The RISC processor, DSP processors, and the memory modules are fully interconnected through the global crossbar network that can be switched at an instruction clock rate of 20 ns. A 50 MHz MVP executes more than 2 GOPS.

The MVP has been integrated into the MediaStation 5000 programmable multimedia system [8]. Its key function in this system is MPEG compression. The data flow in the system during MPEG compression is shown in Figure 6. Video data are captured into the video buffer at a resolution of 320x240. The MVP reads the data from the video buffer and stores it in the main (DRAM) memory. The MVP performs all MPEG compression functions on the data stored in the main memory. Similar operations are performed on the digitized audio samples. Once when the MVP completes the compression of a video or audio frame, the compressed bit stream is sent to the host computer, where the audio and video streams are multiplexed together, synchronized, and stored on a disk or transferred to a network.

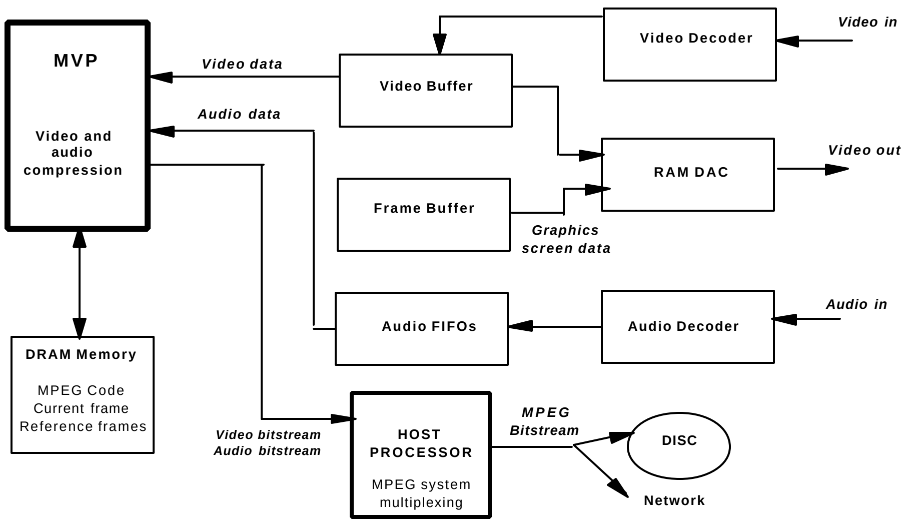

Figure 6. Data flow in the MVP during the MPEG compression [8].

Performance results, reported in [8], show that the MediaStation 5000 system can achieve a real-time compression (30 fps) of MPEG-1 video sequences with resolutions of 320x240 (SIF format). The reported compression times for I frames are 17.7 ms, for P frames 27.3 ms, and for B frames 30.5 ms; for all frame types less than the video frame period of 33 ms. Multiple MVPs are needed for real-time MPEG-2 encoding.

3.2.2 Adapted Programmable Architectures

Adapted programmable architectures provide increased efficiency by adapting the architecture to the specific requirements of video coding applications. These architectures provide dedicated modules for several tasks of the video codec algorithm, such as DCT module or variable length coding [9],[10].

Examples of a commercially available multimedia processor based on adapted programmable architecture are VideoRISC processors (VRP and VRP2) from C-Cube Microsystems. The VRP2 processor consists of a 32-bit RISC processor and two special functional units for variable-length coding and motion estimation, as shown in the block diagram in Figure 7. Specially designed instructions in the RISC processor provide an efficient implementation of the DCT and other video-related operations. The VRP can perform real-time MPEG-1 encoding and decoding; however, the real-time MPEG-2 encoding requires a design consisting of 8 to 13 VRP2 processors.

Table 2, adapted from [5], shows commercially available programmable processors and their features.

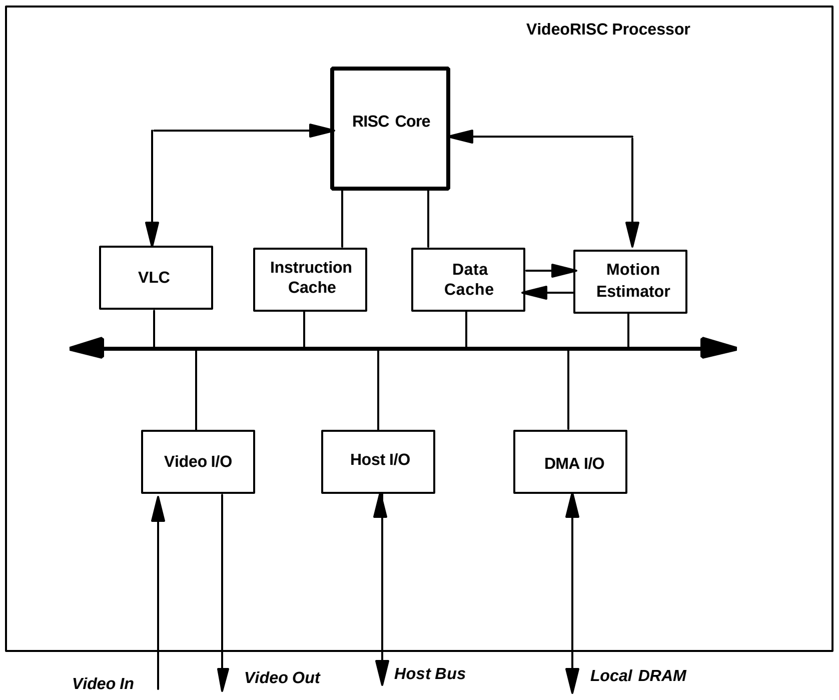

Figure 7. Block diagram of the C-Cube's VideoRISC processor, which applies an adapted programmable architecture.

Table 2. Programmable Multimedia Processors

Multimedia Processor	**Clock [MHz]**	**GOPS**	**Key Characteristics**
TI MVP (TMS320C80)	50	2	Flexible programmable MPEG-1 encoder/decoder MPEG-2 decoder H.261 codec
IIT VCP	80	2	Flexible programmable MPEG-1 encoder/decoder MPEG-2 decoder

			H.261 codec
NEC VSP3	300	1.5	Adapted programmable H.261 codec
C-Cube VideoRISC2	60	2.5	Adapted programmable MPEG-1 encoder/decoder
Matsushita VDSP2	100	2	Adapted programmable MPEG-2 encoder (requires external motion estimation)
Array Microsystems VideoFlow	50	1	Adapted programmable MPEG-1 encoder (requires External motion estimation and Huffman encoder) MPEG-2 decoder

Comparison of these two programmable architectures in terms of silicon area and frame rate, for a variety of codec implementations reported in the literature, is performed in [1]. Adapted processor design can achieve an efficiency gain in terms of the AT criterion by a factor 6-7 compared to flexible architectures. According to this study, 100mm^2/GOPS for flexible architectures and 15 mm^2/GOPS for adapted programmable architectures are needed for a typical video codec.

3.3 NEW ARCHITECTURAL TRENDS

The advanced dedicated multimedia processors use SIMD and VLIW architectural schemes and their variations to achieve very high parallelism. Figure 8 shows the architectural models applied in contemporary multimedia processors and several promising approaches.

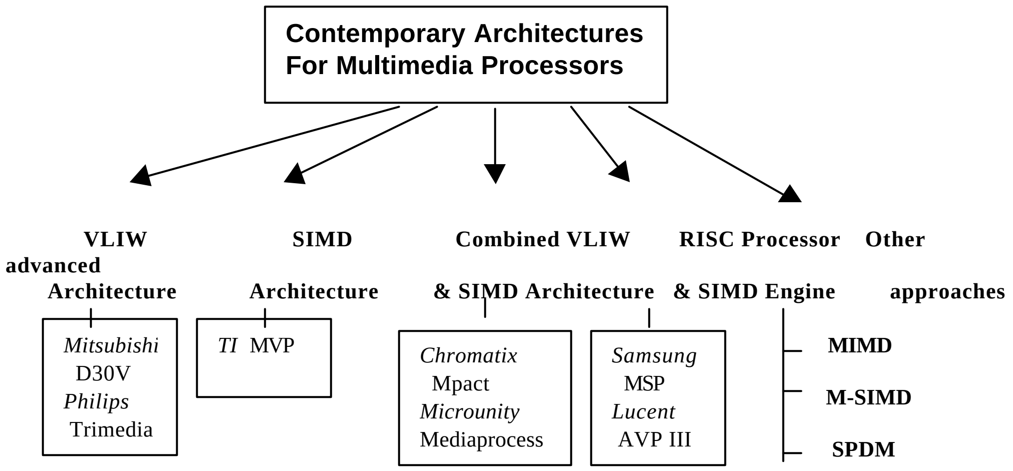

Figure 8. The architectural models applied in advanced multimedia processors.

Two commonly used parallel schemes, the SIMD and the VLIW, are described next. The SIMD parallel computer organization, applied in multimedia processors, typically uses a single control unit (or master processor), a number of processing elements (PEs), and shared memory among the PEs, as shown in Figure 9 [11]. An interconnection network, such as crossbar switch, is used to interconnect the control processor, all PEs, and shared memory. The control processor evaluates every instruction. If it is a scalar or program control operation, a master processor will directly execute it. If the instruction is a vector operation, it will be broadcast to all the PEs for parallel execution. Partitioned data sets are distributed to the shared memory modules before starting the program execution. Then, the same instruction is executed by all the PEs in the same cycle, but on different data elements.

The VLIW architectural model is used in the latest dedicated multimedia processors. A typical VLIW architecture uses long instruction words with more than hundreds of bits in length. The idea behind VLIW concept is to reduce the number of cycles per instruction required for execution of highly complex and parallel algorithms by the use of multiple independent functional units that are directly controlled by long instruction words. This concept is illustrated in Figure 10, where multiple functional units operate in parallel under control of a long instruction. All functional units share a common large register file [11]. Different fields of the long instruction word contain opcodes to activate different functional units. Programs written for conventional 32-bit instruction word computers must be compacted to fit the VLIW instructions. This code compaction is typically done by a special compiler, which can predict branch outcomes by applying an algorithm known as trace scheduling.

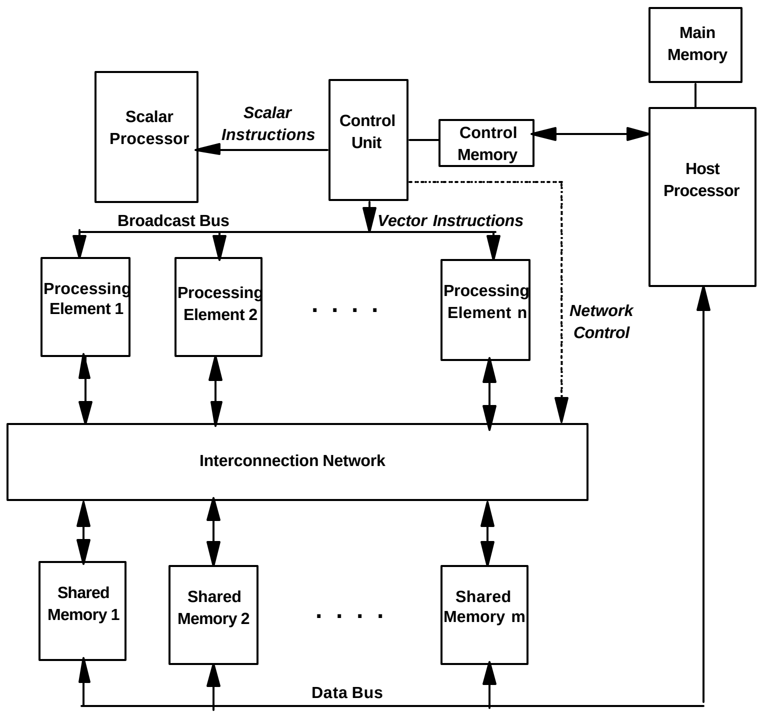

Figure 9. A general SIMD architectural model applied in multimedia processors.

SIMD and VLIW approaches require a giant routing network of buses and crossbar switches. VLIW machines fetch one large instruction per clock cycle and execute all operations on different ALUs. The advantage of this approach is that the ALUs can be specialized to fit the requirements of a given set of applications. However, these architectures’ use of silicon is inefficient, because of the area required by the huge interconnection network and hundreds of pipeline data paths.

Another approach consists of combining a RISC processor with a SIMD machine that operates as a vector processor. A SIMD engine simply executes a conventional 32-bit instruction per clock cycle. This instruction processes a single vector of data points, which execute on a set of identical ALUs of a single pipeline. The data vector is treated as a single number for operand access and memory references. The advantage of this approach includes an efficient use of silicon, because only one pipeline has to be implemented, rather than hundreds.

The next generation of programmable multimedia processors incorporates increased parallelism by combining SIMD, VLIW, and some other hybrid architectural schemes. For example, the Mitsubishi D30V and Philips Semiconductor's Trimedia use the VLIW architecture to boost the performance of their video processors. The Chromatix Mpact multimedia processor combines both VLIW and SIMD concepts. The Lucent's AVP III uses a RISC processor and an SIMD engine for highly parallel operations, and has dedicated functional units for motion estimation and variable-length encoding and decoding.

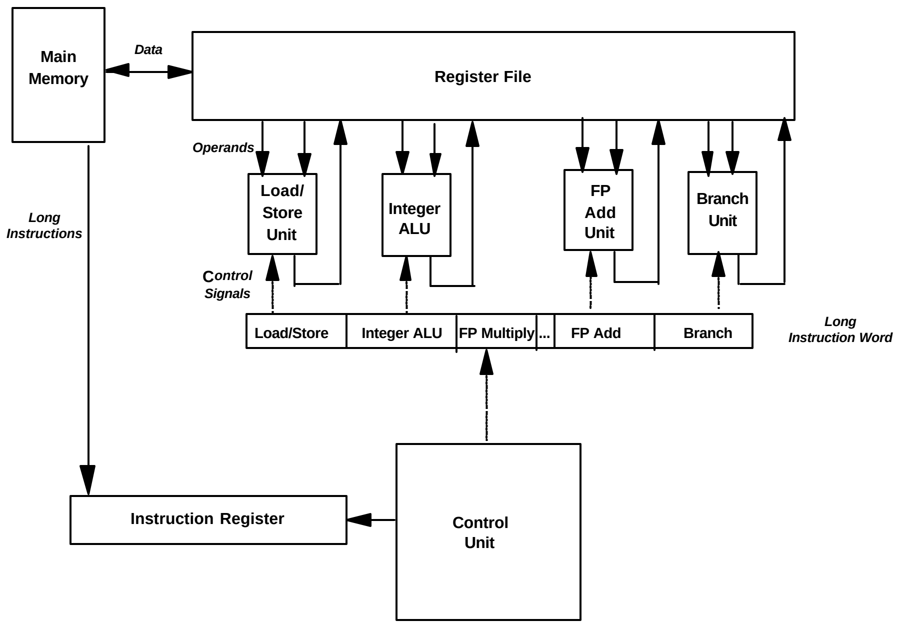

Figure 10. The very-long-instruction-word (VLIW) architectural model applied in dedicated multimedia processors.

Samsung's Multimedia Signal Processor (MSP) combines a traditional RISC controller with SIMD vector processor (VP) and special-purpose hardware engines. The RISC processor runs the RTOS, performs overall system management and control, and some scalar media processing. The VP processor performs high-performance signal processing. Special-purpose hardware units handle some other functions that cannot be performed efficiently in the other two units.

The Mediaprocessor from Microunity Systems Engineering (Sunnyvale, CA) combines a 128-bit load-and-store RISC engine (VLIW concept) with an SIMD-like variation, called single-instruction-group-data (SIGD) parallelism. The architecture also includes a large register file allowing tens of instructions to be executed in parallel. In addition, it also has an execution pipeline that can be either deep (superpipelined), wide (superscalar), or both [7].

Promising Approaches

For the performance and functionality that will be required in next five years, several new approaches are evolving (see Figure 8).

Multiple-instruction, multiple-data (MIMD) architectures offer 10 to 100 times more throughput than existing VLIW and SIMD architectures. In the MIMD approach, multiple instructions are executed in parallel on multiple data, requiring a control unit for each data path. This requires a significant increase in silicon area to implement a control unit for each data path. In addition, a major practical limitation of the MIMD approach is that its implicit asynchronous nature increases the complexity of software development.

Due to these limitations, other hybrid solutions are being studied. One approach is referred to as *multiple single-instruction, multiple-data (M-SIMD*) or SIMD clustering. In this approach, several SIMD clusters are used, each of which consists of a specific number of data paths and an associated control unit. The data paths within each cluster operate as a SIMD array, while the clusters operate in the MIMD mode.

Another promising approach, referred to as *single-program, multiple-data (SPDM)*, combines SIMD and MIMD architectural features. The SIMD nature of this architecture is that it executes a single program or a task at a time, while the MIMD feature is that the data paths operate asynchronously.

4. GENERAL-PURPOSE PROCESSORS AND THEIR SUPPORT FOR MULTIMEDIA

The real-time multimedia processing on PCs and workstations is still handled by dedicated multimedia processors. However, the advanced GP processors provide an efficient support for certain multimedia applications. These processors can provide software-only solutions for many multimedia functions, which may significantly reduce the cost of the system.

GP processors apply the SIMD approach, described in previous section, by sharing their existing integer or floating-point data paths with a SIMD coprocessor. All leading processor vendors have recently designed GP processors that support multimedia, as shown in Figure 1. The main differences among these processors are in the way they reconfigure the internal register file structure to accommodate SIMD operations, and the multimedia instructions they choose to add.

4.1 GENERIC OPERATIONS IN MULTIMEDIA PROCESSING

The instruction mix of multimedia extensions of GP processors varies depending on their application focus. Table 3, adapted from [5], shows typical arithmetic operations required for the main functional blocks of the image and video compression standards, their complexity, inherent parallelism, and the speed-up achieved by current GP processors based on the SIMD approach.

The following conclusions drawn from Table 3 can be used as the main guidelines when specifying multimedia extensions for GP processors [5]:

- Input data and coefficients are typically 8-bit and 16-bit data elements.

- There is no need for floating-point operations.
- The multiply-accumulate operation is very common, but most of multiplications are with constants.
- Saturation arithmetic, where the result is clipped to the maximum or minimum value of a predefined range, is common in many operations.

With the exception of the Huffman (variable length) encoder and decoder, all other operations can be parallelized. Therefore, contemporary GP processors take advantage of this fact by applying the SIMD approach to these operations. A SIMD coprocessor typically performs up to four identical arithmetic or logic operations on different integer-type data. This approach can significantly boost the performance of the GP processors in handling multimedia applications with inherent parallelism (video compression and decompression, image filtering, etc.).

In addition to the arithmetic operations, video processing requires efficient data addressing and I/O processing, which is implemented in some GP processors.

Several contemporary GP processors with multimedia extensions are described in the following sections.

Table 3. Generic Operations Needed for Multimedia Compression

Function	Operations	Complexity	Parallelism
Color transformation (RGB-YUV) Preprocessing and Postprocessing	ΣCiXi, *clip*() (Xi+Xj)/2 (1/4)ΣXi	Constant for every pixel	Highly parallel
FDCT and IDCT	ax+b ΣCiXi	Either constant or a function of the average number of non-zero DCT coefficients	Depends on the implementation of FDCT or IDCT
Quantization	Xi/Ci	Constant for every pixel	Highly parallel
Dequantization	XiCi	Function of the average number of non-zero DCT coefficients	Highly parallel
Motion estimation (Encoder)	Σ|Xi-Yi| or Σ(Xi-Yi)^2 min(a,b)	Depends on the selected motion estimation algorithm	Highly parallel both data-intensive and instruction-intensive processing
Motion compensation (Decoder)	Xi+cXj Block copies Pixel interpolations	Block copies with pixel interpolations	Highly parallel
VLC (Huffman) Encoding/Decoding	Data shifts Comparisons	Function of the average number of symbols in the bitstream	Fully sequential

4.2 INTEL MMX TECHNOLOGY

Intel MMX technology for Intel Pentium processors is targeted to accelerate multimedia and communications applications, especially on the Internet. The fundamental architectural concept in the MMX system consists of the parallel, SIMD-like operation on small data elements (8 and 16 bits). The MMX system extends the basic integer instructions: *add, subtract, multiply, compare, and shift* into SIMD versions. These instructions perform parallel operations on multiple data elements packed into new 64-bit data types (8x8 bit, 4x16 bit, or 2x32 bit fixed-point elements). The MMX instructions also support saturation arithmetic, described in Section 4.1, which is important for multimedia applications.

The following example of *image composition* illustrates how the SIMD concept has been implemented in the Intel MMX system [12]. In this example, fade-in-fade-out effect in video production is performed between two images, A and B, to produce the final image, R, as a weighted average of A and B:

$$R = A * fade + B*(1 - fade) = fade*(A - B) + B \qquad (3)$$

where *fade* is gradually changing from 1 to 0 across a few video frames, thus producing a fade-in-fade-out effect. Let's assume that the frames are in RGB format, where R, G, and B components are not interleaved. In that case, the MMX processor can access four elements of both A and B frames in a single memory access, subtract them in parallel, and then multiply the result with the fade factor in parallel, as illustrated in Figure 11. The MMX code performing this operation is shown in Figure 12.

Performance results for the Pentium processor with MMX technology, reported in [12], show the improvement between 65% to 370% over the same Pentium processor without MMX technology. For example, MPEG-1 video decompression speed up with MMX is about 80%, while some other applications, such as image filtering speed up to 370%.

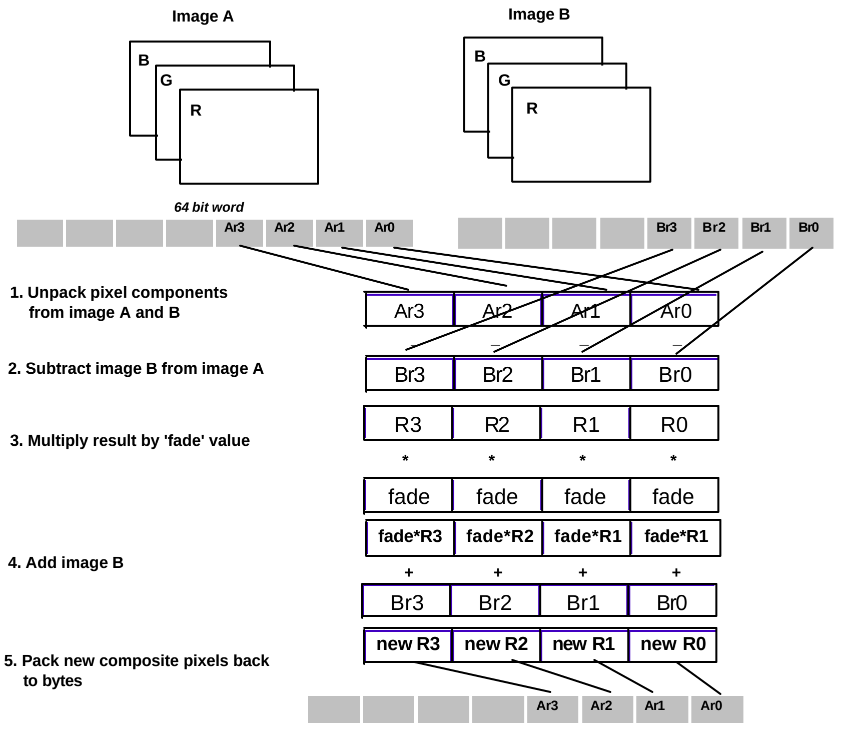

Figure 11. Image composition – fade-in-fade-out effect performed by the MMX system [12].

4.3 SUN'S VISUAL INSTRUCTION SET

Sun has developed Visual Instruction Set (VIS) for its UltraSPARC processors, which provides graphics and image processing capabilities needed for multimedia applications [13],[14]. The VIS supports new data types used for video processing pixels and fixed data. Pixels consist of four 8-bit unsigned integers contained in a 32-bit word, while fixed data consist of either four 16-bit fixed point components or two 32-bit fixed point components both contained in a 64-bit word. The SIMD concept has also been applied for some arithmetic and logic instructions (multiplication, addition, subtraction, and logical evaluations), providing parallel operation on four pixels. An innovative concept, applied in the UltraSPARC, is the *motion estimation instruction* (PDIST), implemented in hardware rather than software. The hardware-implemented instruction PDIST computes the sum of the absolute differences between two 8 pixel vectors, which would require about 48 operations on most processors. Accumulating the error for a 16x16 block requires only 32 PDIST instructions; this operation typically requires 1500 conventional instructions. The hardware implementation of the PDIST (pixel distance) instruction is shown in Figure 13. The circuitry consists of three 4:2 adders, two 11-bit adders, and a 53-bit incrementer. It operates on 8-bit pixels, stored in a pair of double-precision registers, and produces the result in a single-cycle operation.

As reported in [2],[14], the VIS provides four times or greater speedup for most of the time-critical computations for video decompression, including IDCT, motion estimation, and color conversion. According to an analysis, reported in [2], a 200-MHz UltraSPARC processor is capable of decoding MPEG-2 video of 720x480 pixels resolution at 30 fps entirely in software.

```
pxor          mm7,mm7          ; zero out mm7
movq          mm3,fade_val     ; load 4 times replicated fade value
movd          mm0,image A      ; load 4 red pixel components from
image A
movd          mm1,image B      ; load 4 red pixel components from
image B
punpcklbw     mm0,mm7          ; unpack 4 pixels to 16 bits
punpcklbw     mm1,mm7          ; unpack 4 pixels to 16 bits
psubw         mm0,mm1          ; subtract image B from A
pmulhw        mm0,mm3          ; multiply result by fade values
paddw         mm0,mm1          ; add result to image B
```

Figure 12. MMX code performing fade-in-fade-out effect [12].

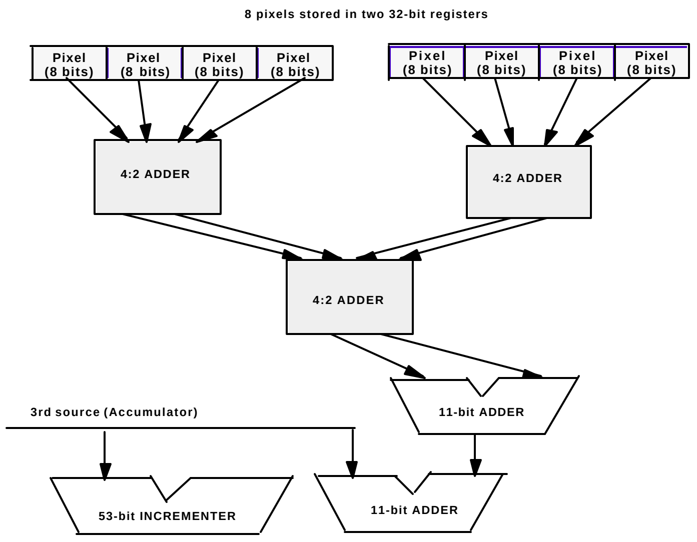

Figure 13. The implementation of the motion estimation instruction in UltraSPARC processor.

4.4 OTHER GENERAL-PURPOSE PROCESSORS

The majority of contemporary general-purpose processors also support multimedia applications (Figure 1). They all apply similar concepts, such as support for new data types and SIMD-like parallel execution of arithmetic and logic operations.

4.4.1 HP PA7100LC Processor

Hewlett-Packard has introduced the PA7100LC processor and its successor HP-PA RISC2 [15]. The processor has two ALU units, which allows a parallelism of four 16-bit operations per cycle. Implementing the color conversation step together with the color recovery step has enhanced the graphics subsystem. Color conversation converts between YCbCr and the RGB color formats, while color recovery allows 24-bit RGB color that had been "color compressed" into 8 bits to be converted back to 24-bit color before being displayed (dithering operation). This enhancement reduces the display operation in MPEG decoding. The memory controller and the I/O controller have been integrated, which significantly reduces the overhead in the memory to frame buffer bandwidth.

4.4.2 DEC Alpha 21164 Processor and MVI Extensions

The Alpha architecture has been enhanced with Motion Video Instruction (MVI) extensions [16]. The MVI consists of three classes of instructions: pixel-error instruction (PERR), Max/Min, and Pack/Unpack instructions. The PERR motion estimation instruction computes the sum of absolute value of differences of eight pairs of bytes. It replaces nine traditional operations. The Max and Min instructions allow efficient clamping of pixel values to the maximum or minimum values, which is convenient for image and video processing applications. The Pack and Unpack instructions expand and contract the data width on bytes and words. Initial results indicate that MVI improves MPEG-2 compression performance more than 400%, compared to the Alpha 21164 processor without MVI [16].

4.4.3 MIPS V Processor and SGI MDMX Extensions

MIPS V processor supports multimedia applications through its instruction-set architecture (ISA) extensions and MIPS Digital Media Extensions (MDMX). Similar to other processors, ISA extensions use new data types (8 bits and 16 bits) and a SIMD approach to perform some arithmetic and logical operations in parallel. MIPS ISA extensions, unlike other processors, also provide support for single-precision floating point operations, which are useful for front end image and video synthesis. MDMX extensions, nicknamed Mad Max, use the innovation from the DSP world, by providing an accumulator with extra precision to support the width required by intermediate calculations. Unlike DSPs, MDMX implements a vector accumulator to take advantage of the SIMD aspect of the instruction set.

4.5 PERFORMANCE ANALYSIS

In summary, GP processors that support multimedia apply new integer data types, well suited for multimedia, new multimedia instructions, and an SIMD architecture for parallel execution of identical instructions on different data.

An example, adapted from [2], shows how a GP processor with an SIMD coprocessor can speedup the MPEG-2 decoding. In the example in Section 2, we calculated the complexity of a typical MPEG-2 decoder, and the obtained results were in the range 609-740 MOPS. However, in most GP processors integer multiplication has a higher complexity than other instructions and, therefore, we will assume that one multiplication is equivalent to 4 generic operations. In this case, the complexity of the MPEG-2 decoder becomes 735 to 876 MOPS for a bit rate of 4 Mbps.

As indicated earlier, the block 1 bit stream parsing and Huffman decoder, cannot be paralellized, and its complexity remains the same – 12 MOPS. However, operations in all the other blocks can be parallelized by a factor 4, assuming a SIMD coprocessor that executes four identical parallel operations on different data. The complexity of the MPEG-2 decoder, implemented on such GP processor, becomes in the 194 to 227 MOPS range, as illustrated in Table 4.

Similarly, for an increased bit rate of 8 Mbps, the total MPEG-2 complexity on a GP processor with an SIMD coprocessor becomes 220-250 MOPS, and for 16 Mbps, is in the 320-352 MOPS range [2].

Table 4. The Complexity of the MPEG-2 Decoder and Its Implementation on a GP Processor with an SIMD Coprocessor

MPEG-2 Functions	**Complexity [MOPS]**	**Parallel Implementation on a GP Processor Using a SIMD coprocessor**
Parser & VLC	**12**	**12**
Dequantization	**14**	**4**
IDCT	**206**	**52**
Motion Compensation	**143-274**	**36-69**
YUV to RGB Conversion	**360**	**90**
TOTAL COMPLEXITY	**735-876**	**194-227**

Contemporary GP processors use superscalar RISC architectures, in which the number of executed instructions is at least 2 x Clock Frequency. Therefore, a 200 MHz processor can execute about 400 MOPS, which suggests that a software-only MPEG-2 decoder can be easily implemented. However, according to Table 1, MPEG encoders require 3,000 to 22,000 MOPS, and even their parallel implementation using a SIMD approach, will require around 1,000 MOPS. This still cannot be achieved with GP processors and dedicated multimedia processors must be used.

5. CONCLUSIONS

In summary, general-purpose designers have recently realized that they should begin investing some of the available chip real estate to support multimedia. We should expect that new generations of GP processors would devote more and more transistors to multimedia. How far and how fast this process will go will be determined by the market demand. By the end of this decade we may see a complete MPEG decoder, large frame buffers, a variety of functional units for video, image, and audio processing, and much more, all packed within a single processor chip.

On the other hand, computationally intensive multimedia functions, such as MPEG encoding, HDTV codecs, 3D processing, and virtual reality, will still require dedicated processors for a long time to come. Therefore, we can expect that general-purpose processors that support multimedia and dedicated multimedia processors will coexist for some time.

REFERENCES

1. Pirsch, P., Demassieux, N., and Gehrke, W., "VLSI Architectures for Video Compression – A Survey," *Proceedings of the IEEE*, 83(2), 220-246, February 1995.
2. Zhou, C-G., Kohn, L., Rice, D., Kabir, I., Jabbi, A., and Hu, X-P., "MPEG Video Decoding with the UltraSPARC Visual Instruction Set," *Proceedings of the IEEE Compcon,* San Francisco, CA, March 1995, pp. 470-475.
3. Pennenbaker, W.B. and Mitchell, J.L., "JPEG Still Image Data Compression Standard," Van Nostrand Reinhold, New York, 1993.
4. Fujiwara, H., Liou, M.L., Sun, M-T., Yang, K-M., Maruyama, M., Shomura, K., and Ohyama, K., "An All-ASCI Implementation of a Low Bit-rate Video Codec," *IEEE Trans. On Circuits and Systems for Video Technology*, 2(2), 123-134, June 1992.
5. Bhaskaran, V. and Konstantinides, K., "Image and Video Compression Standards – Algorithms and Architectures," Kluwer Academic Publishers, Boston, MA, 1995.
6. Guttag, K., Gove, R.J., and Van Aken, J.R., "A Single-Chip Multiprocessor For Multimedia: The MVP," *IEEE Computer Graphics & Applications*, 53-64, November 1992.
7. Cole, B., "New Processors Up Multimedia's Punch," *Electronic Engineering Times*, 71, February 3, 1997.
8. Lee, W., Kim, Y., Gove, R.J., and Reed, C.J., "MediaStation 5000: Integrating Video and Audio," *IEEE MultiMedia*, 1(2), 50-61, Summer 1994.
9. Ackland, B., "The Role of VLSI in Multimedia," *IEEE J. Solid-State Circuits*, 29, 1886-1893, December 1992.
10. Akari, T. et al., "Video DSP Architecture for MPEG2 Codec," *Proceedings of ICASSP*, Vol. 2, IEEE Press, 417-420, 1994.

11. Hwang, K., “Advanced Computer Architecture with Parallel Programming,” McGraw-Hill, 1993.
12. Peleg, A., Wilkie, S., and Weiser, U., “Intel MMX for Multimedia PCs,” *Communications of the ACM*, 40(1), 25-38, January 1997.
13. Tremblay, M., O’Connor, J.M., Narayanan, V., and Liang, H., “VIS Speeds New Media Processing,” *IEEE Micro*, 16(4), 10-20, August 1996.
14. Kohn, L., Maturana, G., Tremblay, M., Prabhu, A., and Zyner, G., “The Visual Instruction Set (VIS) in UltraSPARC,” *Proceedings of the IEEE Compcon*, San Francisco, CA, 462-469, March 1995.
15. Lee, R.B. “Realtime MPEG Video via Software Decompression on a PA-RISC Processor,” *Proceedings of the IEEE Compcon*, San Francisco, CA, 186-192, March 1995.
16. Bannon, P. and Jain, A., “MVI Instructions Boost Alpha Processor,” *Electronic Engineering Times*, 74, February 3, 1997.