CHAPTER 21

Hardware Implementation of Data Compression

SANJUKTA BHANJA
N. RANGANATHAN

21.1 INTRODUCTION

With the advancement of information technology, large-scale information transfer by remote computing and the development of massive information storage and retrieval systems have witnessed a tremendous growth. The growth of these systems implies the need for efficient mechanisms for storage and transfer of enormous volumes of data. Data compression is the reduction of redundancy in data representation in order to decrease data storage requirements and data transfer costs. It is the process of transforming a body of data into a smaller form from which the original or some approximation of the original can be recovered at a later time. The compression methods used depend on the type of data such as text, image, video, graphics, and audio, which collectively represent the multimedia information. In lossless data compression, the data that are compressed and subsequently decompressed must always be identical to the original data. In lossy compression, the decompressed data are an approximation of the original data. Lossless methods are referred to as entropy coding methods and are used in the compression of text and certain types of images; lossy methods are used in compressing image, video, and audio data that contain significant redundancy. Furthermore, most hardware products for image and video compression typically combine lossless and lossy compression capabilities together in the same product. For example, in the JPEG image compression implementations, the lossy compression includes hardware implementation of lossless coding, making it difficult to separate them. Thus, in this chapter, we include discussions on hardware implementation of both lossless and lossy compression.

In recent years, the demand for data compression and the need to develop efficient compression algorithms and implementations have increased considerably due to the increased use of data compression within commercial, scientific, and statistical information systems, the World Wide Web,

document delivery systems, and communications networks. Furthermore, in the field of multimedia, high-speed data compression has become critical to many communication and entertainment applications such as digital TV, video phone, video-on-demand, Internet, DVD, interactive games, and multimedia education. It is pointed out in [9] that the advances seen in multimedia and the proliferation of multimedia applications are due mainly to the progress in three areas: standards, networking and VLSI.

During the 1980s, data compression was primarily used through software implementations, and the time and space requirements of the compression and decompression routines made it less attractive. The emergence of compression standards in the early 1990s and the growth in VLSI technology made efficient hardware implementations of data compression a viable alternative. For example, as soon as the baseline JPEG standard was established, several companies such as LSI Logic, C-Cube, and Intel announced single-chip and multichip VLSI systems implementing the JPEG. Today, data compression hardware has become a critical component in virtually every household, either within the cable TV switch box or in a high-speed modem connecting the personal computer to the Internet. The main goal of compression hardware systems is to achieve high performance and throughput at low cost and with low power consumption. Although compression is often implemented using programmable processor cores such as DSP or multimedia processors, the design of dedicated architectures became essential to meet the performance requirements. The establishment of standards made it possible to design and develop special-purpose architectures and application-specific VLSI systems to implement sophisticated compression techniques that can process in real time. In fact, when the various algorithms and methods are considered for inclusion in the standards, one of the major criteria is their amenability to hardware implementation.

In developing high-speed VLSI architectures, several attributes are important: pipelining and parallel processing, high-speed memory access, scalability with technology and size, programmability, low power consumption, and low cost. The compression algorithms in general are computationally intensive in nature. Thus, it is important to exploit all inherent parallelism in the algorithms and map them onto parallel architectures. The use of array processors (for exploiting spatial parallelism) with nearest-neighbor communication and extensive pipelining of the various logic modules within the processors as well as the entire system architecture (for exploiting temporal parallelism) is critical to obtaining high throughput. Numerous systolic, SIMD, and MIMD array architectures have been proposed in the literature for data compression and decompression. The design of efficient memory systems that can store and provide image or video datastreams to the hardware processors keeping up with the high processing rate is critical to the performance of the system.

The architecture as well as the implementation must be highly scalable with technology. The scalability of an architecture determines the life span of the architecture in the market. It indicates whether an architecture can support further developments in terms of VLSI technology, data compression algorithms, and the emerging new standards. The advances in VLSI technology yielding lower gate delays, higher packing density, faster clocks, etc., directly impact the architecture. The minimum feature size (the size of the smallest transistor or the minimum wire width possible) keeps decreasing, which in turn could significantly increase the integration capability (the maximum number of devices that can be packed within a single die or chip). For example, a hardware system designed using 0.5-μm technology (the minimum feature size being 0.8 μm) must be easily portable into a 0.18-μm design when needed. Conversely, one may want to produce a low-cost system for certain applications by using an older, cheaper technology. The process of scaling up or scaling down in VLSI technology can have different implications in terms of the performance of the overall system. Thus, the architecture must be easily amenable to the changing technology. The size scalability corresponds to the architecture and the implementation being able to adapt to an increase or a decrease in the size of any system parameter or configuration. For example, in a

networked multimedia system, such changing parameters could be the video frame size, the frame rate, network traffic, performance requirements, quality of service parameters, and application behavior.

Programmability is another important criterion in multimedia hardware systems so that they can handle different standards and also variations in standards or algorithms that occur in time. However, while programmable processors are flexible, they are slower than dedicated hardware in general. Often, the computation-intensive tasks such as DCT (Discrete Cosine Transform) coding or variable-length encoding, which are common to most image and video compression standards, are implemented as dedicated special-purpose hardware to gain maximum speed. The optimization of power consumption has become important in the context of battery-operated systems as in wireless and mobile computing environments. The reduction of power consumption is targeted at almost every level: algorithmic, architectural, gate, circuit, and device levels. Thus, besides low cost and high performance, low power has become an important design goal in wireless multimedia systems.

Numerous architectures as well as commercial VLSI products for data compression have come into existence over the past decade. There are a plethora of publications in the area of hardware data compression that can be found in the literature. It is almost impractical to cite and cover every important contribution within a single chapter. Thus, our intention in this chapter is to provide a brief overview that can serve as a starting point. We include brief case studies of a few architectures merely to serve as examples for reading. The interested reader is referred to the vast literature for further exploration. The rest of the chapter is divided into sections that discuss text, image, video, and commercial products.

21.2 TEXT COMPRESSION HARDWARE

A wide variety of lossless compression methods have been proposed in the literature for text data. Examples include Huffman coding [21], the multigroup technique [5], run-length coding [77], Lempel–Ziv coding [43], arithmetic coding [27], and other dictionary-based methods. Many hardware architectures for text compression implement tree-based codes, also referred to as variable-length encoders. Other popular compression methods implemented in hardware are arithmetic coding, Lempel–Ziv coding, and some dictionary-based methods.

The implementation of variable-length codes is difficult due to the fact that the codeword boundary cannot be determined until the previous codeword has been decoded completely, indicating a bit-sequential nature of processing. This in turn puts limits on the decoding speed and throughput. A method to address this problem was reported in [5, 6] using the concept of a reverse binary tree for static Huffman encoding and decoding. Also, an adaptive and pipelined design for generating the Huffman tree codes given the statistical distribution of characters in a text file was given. A high-speed entropy decoder for VLE codes based on programmable logic arrays was proposed in [61]. A memory-based architecture called MARVLE was proposed in [4] for implementing tree-based codes in which the nodes of the tree representing the codes are mapped to specific memory locations based on a specific algorithm that aids in fast encoding and decoding of text files using tree-based codes. Area-efficient architectures with memory-mapping schemes were proposed in [29]. Hardware designs using content addressable memory for implementing dynamic Huffman coding can be found in [52].

The principle behind the Lempel–Ziv (LZ) methods is to find the longest strings that match a previous occurrence of that string within the same file or a statically predefined or a dynamically constructed dictionary of strings and then replace the current occurrence with a pointer to the previous occurrence or to the location within the dictionary. The LZ methods and their variations

have been excellent candidates for VLSI implementation. Several systolic VLSI designs for implementing the Lempel–Ziv methods can be found in the literature [10, 11, 44, 56, 71, 73, 75]. Furthermore, CAM-based designs can be found in [46] and other custom architectures can be found in [20, 90]. Since the systolic designs are ideal architectural candidates for VLSI implementation, we will describe one such approach [64] as an example later in this section.

A massively parallel architecture for text compression using textual substitution was introduced in [44]. A new algorithm for text compression based on textual substitution and its implementation as an ASIC using systolic arrays was proposed in [25]. Hardware implementations of arithmetic coding can be found in [32, 45]. The work in [45] describes a lossless compression method based on arithmetic coding that uses fuzzy inferencing and its hardware implementation. The Rice algorithm [72] for lossless compression and its implementation as a VLSI chip set are described in [42]. More architectures can be found in the literature, not covered here for want of space. In the rest of this section, we briefly discuss two architectures, one for tree-based codes [4] and another for LZ coding [64], as examples.

21.2.1 Tree-Based Encoder Example

Tree-based codes are commonly used for text compression in which the codes can be represented by trees, such as binary trees or quad trees, where the sequence of 0's and 1's on the unique path from the root of the tree to a leaf node represents the code for the symbol represented by the leaf node. Some examples of these codes are the Shannon–Fano code, the Huffman code, the Elias code, and the Fibonacci code. These codes are derived based on the statistical distribution of the various characters within a file in text compression. The tree-based codes are variable-length codes in which compression is achieved by assigning short codewords to high-probability symbols and assigning long codewords to low probability symbols. In order for these codes to be uniquely decipherable, they must also have the prefix property, which states that no code can be a prefix of any other code.

An example of a Huffman tree for the symbols a, b, c, d, e, f, and g with probabilities of 0.2, 0.15, 0.10, 0.25, 0.05, and 0.15 is given in Fig. 21.1a. In order to facilitate the implementation of the compression process in hardware, a reverse binary tree is derived for the tree representing the code. A reverse binary tree [6] is a labeled binary tree whose leaf nodes and some of the internal nodes represent the symbols to be coded. The sequence of 0's and 1's on the unique path from the symbol node to the root of the tree represents the code for that symbol. The reverse binary

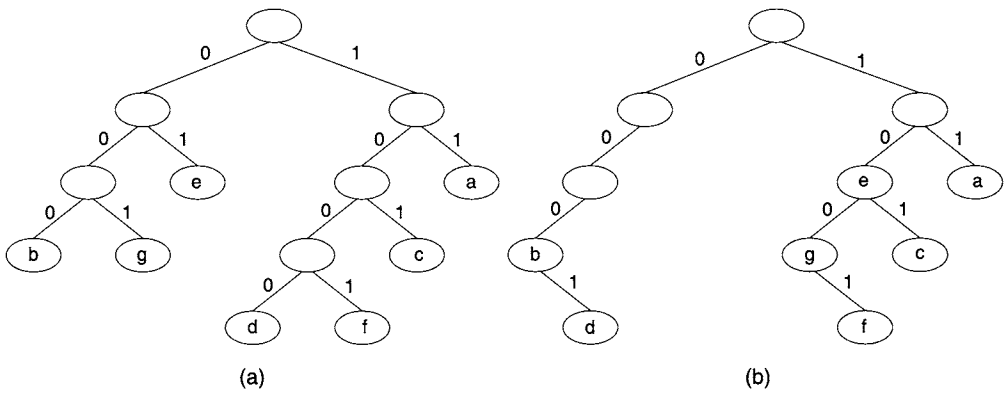

FIGURE 21.1
Huffman and reverse tree example.

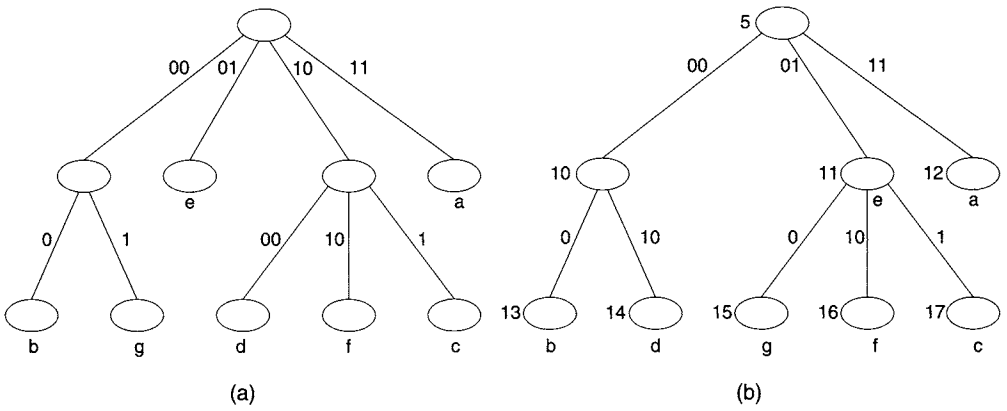

FIGURE 21.2
Two-bit trees for Huffman and reverse tree examples.

tree for a code can be constructed by using the reverse codes for each symbol. The reverse binary tree for the Huffman tree in Fig. 21.1a is given in Fig. 21.1b.

The encoding and decoding are inherently serial operations since the tree must be traversed one edge at a time. In order to speed up the process, a multibit tree may be used. In a multibit tree, each edge of the tree represents a maximum of k bits of the input code. If the length of a codeword is n bits, then it is represented by n/k edges from the root of the tree to a leaf node of which only the last edge (the one terminating at the leaf node) could possibly have a label less than k bits long. The $k-$bit reverse tree is analogous to the 1−bit reverse tree where the codes are reversed k bits at a time. Each node of the $k-$bit tree can have a maximum of $2k$ children. The 2−bit trees corresponding to the Huffman code tree and the reverse tree in the above example are shown in Fig. 21.2a and 21.2b. The architecture of MARVLE is centered around a memory device in which the code trees are mapped and stored for encoding and decoding.

21.2.1.1 Memory Mapping

A memory map of a tree is defined by associating a distinct memory location with each node of the tree [4]. The mapping of the decoding tree is relatively more complex than the mapping of the encoding tree. The following procedure can be used for the mapping of the decoding tree: Given a $k-$bit decoding tree with n nodes, of which p nodes have at least two child nodes and the remaining (np) nodes are either leaf nodes or nodes with a single child, the number of child nodes of a node N_i is C_i, where $1 \leq C_i \leq 2^k$. The memory mapping proceeds as follows:

1. Each edge that connects a node N_i and a child node N_j has a label $L_{ij} = x_1, x_2, \ldots, x_s$, where $s \leq k$ and each x_i is a single bit: 0 or 1.
2. An integer B_{ij} is assigned to each L_{ij} such that $B_{ij} = \sum_{i=1}^{S} 2^{k-i} x_i$.
3. For each node N_i a corresponding positive integer M_i will be assigned and a set of memory addresses will be defined for each node N_i as $Mem(N_i) = \{M_i + B_{ij}\}$, $j = 1, 2, \ldots, C_i$.
4. Steps 1–3 are repeated for each N_i, $1 \leq i \leq p$.
5. For each node N_i, a value for the corresponding M_i is chosen such that all $Mem(N_i)$ are distinct.
6. Map the remaining nodes $N_{p+1}, \ldots, N_n$ to distinct positive integers outside the memory map.

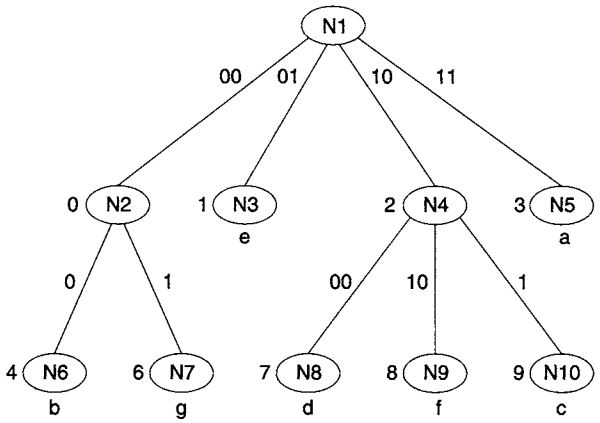

FIGURE 21.3
Memory map for Huffman example.

The mapping process will be illustrated using a 2-bit Huffman tree where each edge of the tree represents a maximum of 2 bits of the code. The 2-bit Huffman tree for the example in Fig. 21.1a with the nodes labeled N1 through N10 is shown in Fig. 21.3.

The equations for $Mem(N_i)$ are as follows:

$$Mem(N1) = \{M1 + 0, M1 + 1, M1 + 2, M1 + 3\}$$
$$Mem(N2) = \{M2 + 0, M2 + 2\}$$
$$Mem(N3) = \{M3 + 0, M3 + 1, M3 + 2\}.$$

The assignment of $M_1 = 0$, $M_2 = 5$, and $M_3 = 8$ yields a valid memory map. This maps the nodes N_2 through N_{10} to memory addresses 0, 2, 1, 3, 5, 7, 8, 9, and 10, respectively. Note that the mapping has not utilized the addresses 4 and 6. The number of memory locations not utilized in the block of contiguous memory addresses for the decoding tree will be called the gap W. For this example, $W = 2$. The encoding map is created using the reverse binary tree by mapping each node of the tree to a distinct memory location. The choice of memory locations is simplified by the fact that the algorithm starts at a symbol node and traverses up the tree until the root node is reached. Since each node has only one parent, the addresses can be arbitrarily chosen as long as they are distinct. One possible memory map for the encoding tree is shown in Fig. 21.3 with memory addresses shown adjacent to each node.

The memory required for implementing the mapping in the MARVLE architecture is composed of 12-bit data words. The 12-bit words store data and control bits related to the nodes and edges of the encoding and decoding trees. Each memory word contains a 9-bit field for the data word and three control bits, T, S1, and S2. The various fields of the memory word have different meanings for the encoding and decoding memory maps. The encoding tree contains three types of nodes that need to be mapped onto the memory locations: (i) regular nodes, (ii) special nodes, and (iii) terminal nodes. The regular node is a non-leaf node in which all the edges from the node to its children have labels of 2 bits. The special node is a non-leaf node in which at least one of the edges connecting the node to its children has a single bit label. The terminal node is a leaf node that represents an output symbol. The data field for a non-terminal node (regular or special) contains the value of M_i for the node, and the data field for a terminal node contains a binary representation of the symbol at that node (e.g., ASCII). The three control bits T, S1, and S2 are used to identify the different types of nodes.

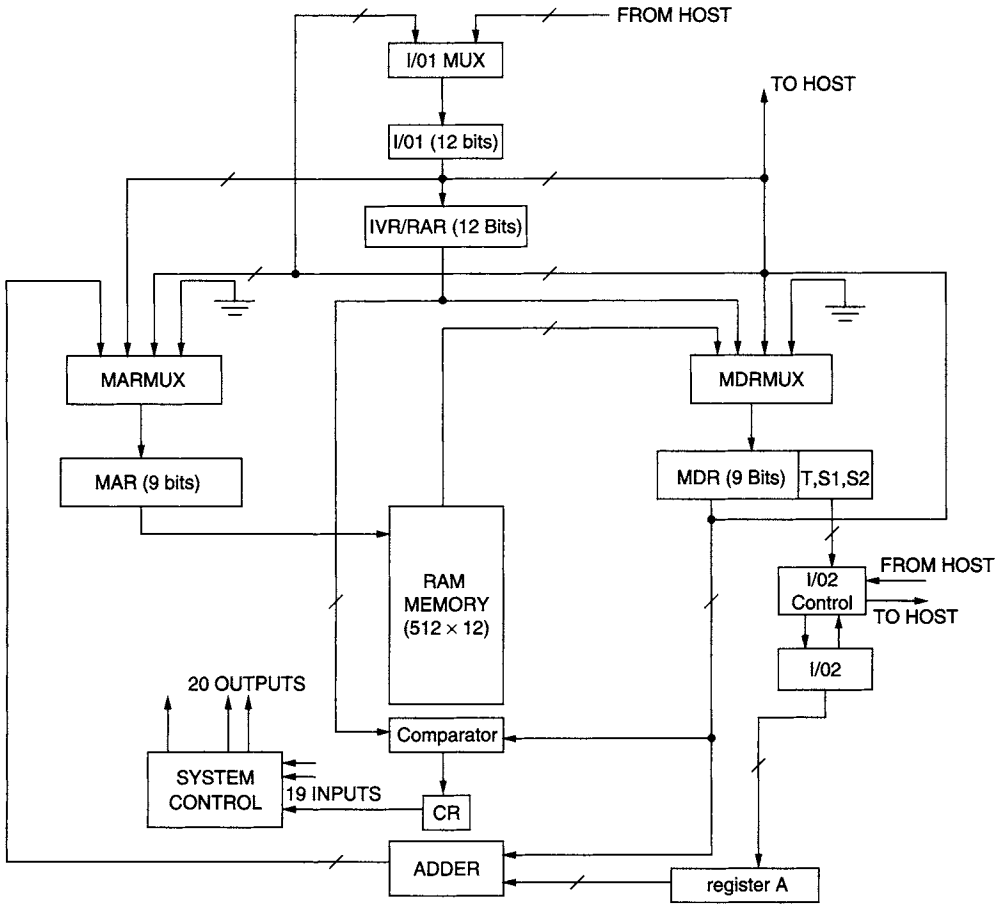

FIGURE 21.4
MARVLE architecture.

21.2.1.2 MARVLE Architecture

The main components of the MARVLE architecture are: (i) a 512 × 12 bit random access memory (MEM) consisting of 12-bit words having the format described in the previous section, (ii) a 9-bit memory address register (MAR), (iii) a 12-bit memory data register (MDR), (iv) a 12-bit parallel interface register (I/O_1), (v) a 2-bit parallel interface register (I/O_2), (vi) a 2-bit register (A) for compiling the offset value during the decoding process, and (vii) a 12-bit initial value register/root address register (IVR/RAR) that is used to hold the root address of the encoding tree and the initial T,S1, and S2 values for the root of the decoding tree. The notation MEM[ADDRESS] denotes the data stored in the memory at the location ADDRESS.

A block diagram of the architecture is shown in Fig. 21.4. The major elements of the architecture are the system memory (MEM) and a set of registers. The registers are divided into two groups internal registers and input/output (I/O) registers. The internal registers are the MAR, MDR IVR/RAR, and A and the comparator register that latches the output of an equality comparator The compression and decompression algorithms require two other functional units, an adder and a bit-wise equality comparator. This adder is used during the decoding process to add the 2-bit offset in the A register to the base address contained in the MDR. This is a special-purpose adder that adds 2 bits to 9 bits and produces a 9-bit result. No attempt is made to deal with overflow

since in the correct operation of the chip this situation never occurs. The bit-wise equality comparator is used during the encoding process to compare the 9 high-order bits of the IVR/RAR and the 9 high-order bits of the MDR to determine when the root of the tree is reached. The basic steps of the encoding algorithm are as follows:

1. Load the address of the symbol to be encoded into the MAR.
2. Check to see if the address is the root address of the tree, indicating that the encoding is done. If the address is the root address of the tree, go to step (1) (or stop if there are no more symbols).
3. Access the memory and place the contents of the location pointed to by the MAR in the MDR.
4. Output the bit(s) of the encoded symbol to the host. Output S_1 if $T = 0$ or S_1 and S_2, if $T = 1$.
5. Load the MAR with the 9 high-order bits of the MDR (address of parent of the current node). Go to Step (2).

The basic steps of the decoding algorithm are as follows:

1. Load the MDR with the initial values for the root of the decoding tree.
2. If $T = 0$, output the decoded symbol to the host and go to (1).
3. Place the next bit on the input bitstream in A[2]. This is the high-order bit of the offset to the next node.
4. Determine the low-order bit of the offset and place it in A[1]. The low-order bit of the offset is determined as follows:
   If S1 = S2 = 0, A[1] = the next bit on the input bitstream.
   If S1 = S2 = 1, A[1] = 0.
   If (S1 $\neq$ S2) and (A[2] = S2), A[1] = 0
   If (S1 $\neq$ S2) and (A[2] $\neq$ S2), A[1] = the next bit on the input bitstream.
5. Add the offset in A to MDR[12..4] and place the result in the MAR.
6. Access the memory and place the contents of location pointed to by the MAR in the MDR. Go to Step (2).

A VLSI chip prototype of the above architecture was reported to yield a compression rate of 95.2 Mbits/s and a decompression rate of 60.6 Mbits/s operating at 83.3 MHz [4]. In order to improve the performance it was pointed out that the architecture can be expanded to handle higher order trees such as 3- or 4-bit trees. This would speed the process because the limiting factor in performance is the memory cycle, which limits the clock speed of the chip. Increasing the dimension of the coding and decoding trees would mean that more bits will be decoded on each cycle of the algorithm, so that the rates would increase proportionally. However, increasing the tree size would increase the complexity of the control logic and the memory-mapping procedure. Hence, there is a trade-off between the increase in the tree dimension and the complexity of control and decoding time.

21.2.2 Lempel–Ziv Encoder Example

In this subsection, we describe a simple hardware implementation of the basic LZ77 compression technique that appeared in [64]. The LZ technique lends itself well to a systolic parallel array implementation. The systolic algorithm is explained using an example with a buffer of size 8 shown in Fig. 21.5. The first four locations of the buffer, labeled x1 through x4, contain symbols

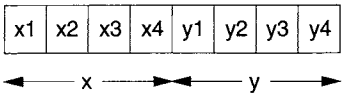

FIGURE 21.5
Buffer of size 8.

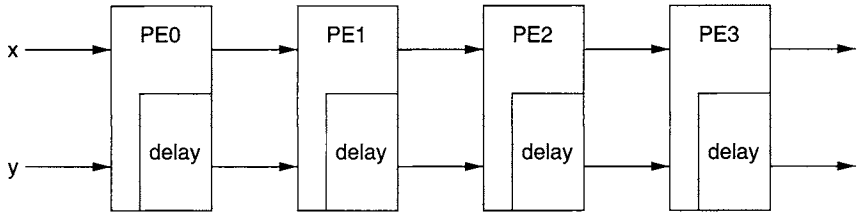

FIGURE 21.6
A systolic array of four processors (PEs).

that have already been processed and the other four locations, labeled y1 through y4, contain symbols to be processed. The symbols are input to the buffer from the right and are shifted out from the left.

The following four sets of comparisons must be done in sequence in order to find the maximum matching substring, as can be seen from the pseudo-code given above:

$$\begin{array}{lll} x1\text{–}y1 & x2\text{–}y2 & x3\text{–}y3 \\ x2\text{–}y1 & x3\text{–}y2 & x4\text{–}y3 \\ x3\text{–}y1 & x4\text{–}y2 & y1\text{–}y3 \\ x4\text{–}y1 & y1\text{–}y2 & y2\text{–}y3 \end{array}$$

Here a dash indicates an equality comparison between the symbols. The set with the maximum number of matches in sequence determines the required substring. "In sequence" indicates that if any of the comparisons in a set fails, the succeeding comparisons in that set do not count as successful ones. For example, if $x1$–$y1$ was a successful comparison and if $x2$–$y2$ was not a successful comparison, then the result of the $x3$–$y3$ comparison is ignored and the match length is 1 in that case. The number of comparisons in the sequential algorithm is $O(n^2)$. The above set of comparisons can be reorganized in the following fashion:

$$\begin{array}{llll} x1\text{–}y1 & x2\text{–}y1 & x3\text{–}y1 & x4\text{–}y1 \\ x2\text{–}y2 & x3\text{–}y2 & x4\text{–}y2 & y1\text{–}y2 \\ x3\text{–}y3 & x4\text{–}y3 & y1\text{–}y3 & y2\text{–}y3 \end{array}$$

Let us consider four processors operating in parallel, each performing one vertical set of comparisons. Each processor would require 3 time units ($n - 1$ time units in general) to complete its set of comparisons. This results in the reduction of comparison time to $O(n)$, which is achieved by performing n comparisons in parallel. Consider the systolic array of four processing elements (PEs) as shown in Fig. 21.6. The "delay" block in each PE delays the y values by one time step. A space–time diagram illustrating the sequence of comparisons as performed by each PE is given in Fig. 21.7. The data brought into PE0 are routed systolically through each processor from left to right. In Fig. 21.7, the first three comparisons shown in each column indicate the comparisons to be performed. The symbol z denotes a dummy symbol and the comparisons involving z are unused. The sequence of comparisons to be performed by the PEs is explained as follows:

cycle	PE0	PE1	PE2	PE3
1	x1-y1			
2	x2-y2			
3	x3-y3	x2-y1		
4	x4-z lgt-max	x3-y2		
5	y1-z	x4-y3	x3-y1	
6	y2-z	y1-z lgt-max	x4-y2	
7		y2-z	y1-y3	x4-y1
8			y2-z lgt-max	y1-y2
9				y2-y3
10				lgt-max

FIGURE 21.7
The space–time diagram for systolic data flow.

The data are input to PE0, which is passed systolically to the other PEs. In the first time unit, x_1 and y_1 are compared at PE0. In the second time unit, x_2 and y_2 are compared and x_1 flows to PE1 and y_1 is delayed by one cycle. In the third time unit, x_3 and y_3 are compared at PE0. At this time, y_1 gets to PE1 along with x_2 and PE1 performs its first comparison. After the third cycle, PE0 completes all its required comparisons and stores an integer specifying the number of successful comparisons in a register called length. Another register, max, holds the maximum matching length obtained from the previous PEs. In the fourth time unit, PE0 compares the values of max (which for PE0 is 0) and length and the greater of the two is sent to the max register of the next PE. Although x_4 is not used by PE0, it is used by the following PEs. Similarly, y_1 and y_2 are input to PE0 in the following time units as seen in the space–time diagram. The comparisons involving "z" are ignored. The result of the *length–max* comparison is sent to the next PE after a delay of 1 time unit for proper synchronization. Finally, the *max* value emerging from the last PE (PE3 in this case) is the length of the longest matching substring. There is another register called id, whose contents are passed along with the *max* value to the next PE. Its contents indicate the id of the processor where the *max* value occurred, which becomes the pointer to the match. The method takes $3n$ units in general to find the maximum possible match length, where n is the number of PEs.

21.2.2.1 The Processing Element

Each PE performs the following set of functions: (i) The incoming symbols are compared to check whether they are equal. If they are equal, a counter is incremented each time until an unsuccessful comparison occurs. The output of the counter gives the length of the match. (ii) Once a PE has found its match, it compares the match length with the max from the previous PE. The greater of the two is output as the new max to the next PE. (iii) The id of the PE that detected the max value

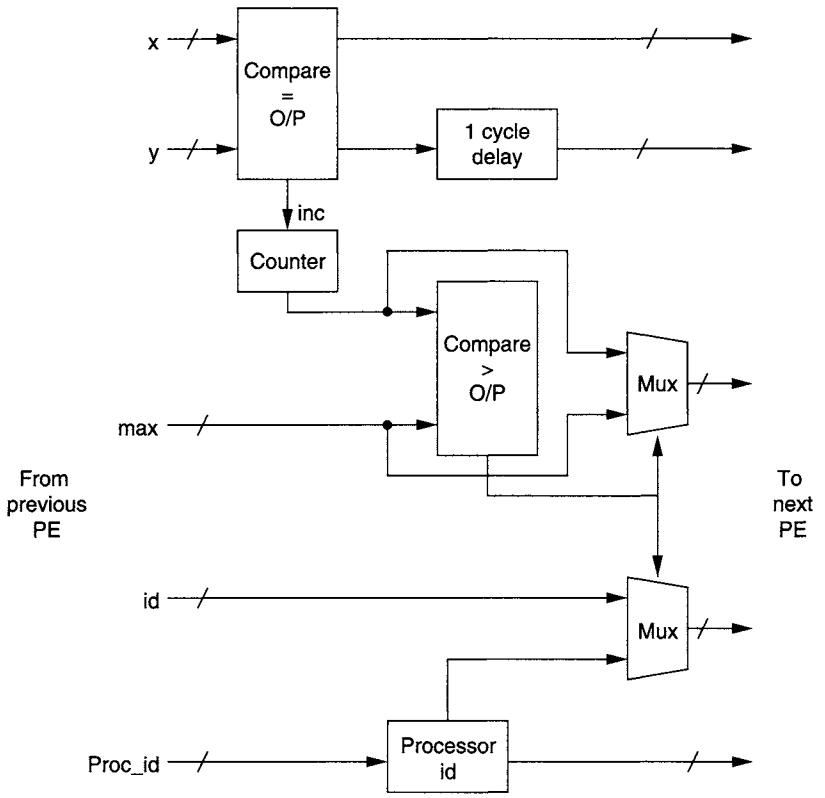

FIGURE 21.8
The processing element.

corresponding to the longest match becomes the pointer to the start of the match, which is also transmitted along with the max. The hardware design of the PE, as shown in Fig. 21.8, consists of three parts that perform the above three tasks: the upper module, the central module, and the lower module. For more details, the reader is referred to [64].

21.3 IMAGE COMPRESSION HARDWARE

Digital images require an enormous amount of space for storage. For example, a single-color image with a resolution of 1024 × 1024 picture elements (pixels) with 24 bits per pixel would require 3.15 MB in uncompressed form. Thus, image compression is critical in large image databases as well as in applications where images are being transferred over the network. Lossless image compression algorithms are especially important in medical and space applications while most other applications can afford a certain amount of loss. The lossless methods for image compression can be classified as statistical coding techniques such as Huffman and Fano and universal codes such as arithmetic coding, Lempel–Ziv codes, and other block level template matching methods. Lossy image compression often presents trade-offs between the amount of compression and the fidelity required in reconstruction and thus is interesting and challenging. Numerous types of methods and their variations can be found in the literature. Lossy methods can be typically classified based on the domains, such as spatial, frequency, and time domains, or the techniques

they are based on, such as transforms, subband coding, wavelets, quantization, and differential encoding.

Image compression methods are computationally intensive and ideal candidates for hardware implementation. A large number of works have appeared in the literature on hardware architectures and implementations for different image compression methods. The JPEG standard is used for still-picture compression and the MPEG standards are used for moving pictures requiring intraframe as well interframe coding. Since the JPEG and the MPEG standards were announced, several different VLSI architectures have been proposed and commercial chips implementing them have been available since the early 1990s. The majority of the hardware implementations for image compression have targeted the following methods: the transform methods such as the DCT, DFT (Discrete Fourier Transform), wavelets, vector quantization, and the JPEG standard itself. The first JPEG standard used DCT and DPCM (differential encoding) for the transform part and a combination of run-length, Huffman coding, and arithmetic coding for the entropy encoding part. While many hardware products implemented the baseline JPEG standard, there is no commercial hardware for the more recent JPEG 2000 standard. For detailed information on the algorithms and architectures for various image compression methods, the reader is referred to a recent book by Bhaskaran and Konstantinides [87]. Recently, the trend in the hardware market has moved more toward video compression products and image compression being packaged as part of the video standard that is implemented. Keeping that in mind, here we provide only brief pointers to a few examples of hardware approaches to image compression and move on to a more detailed treatment of video compression hardware in the next section.

21.3.1 DCT Hardware

It should be noted that two-dimensional DCT computation can be implemented as a sequence of two one-dimensional DCTs, which is commonly referred to as the separability property. This approach is simpler to implement in hardware. It was shown by Haralick [74] that the DCT of N points can be computed using two N-point FFTs by exploiting the symmetry of the inputs. Later, it was shown in [12] that the DCT can be obtained more efficiently by computing just the real part of the first N coefficients of the $2N$-point DFT. The computation of 8-point DCT needed for JPEG can be replaced by 16-point DFT computation followed by scaling. An optimum form for 16-point DFT was developed by Winograd [81] and was later adapted for 8-point DCT by reducing the computation by using the symmetry property [92]. A modification reducing the number of 2's in complement operations can be found in [57]. Recently, several new and improved VLSI designs for DCT, targeting high performance as well as low power using principles such as multirate arrays and asynchronous pipelines, have appeared in the literature [7, 22]. Due to the extensive use of DCT in various applications that demand real-time processing, numerous VLSI chips have been designed and built by both academia and industry. For a more detailed treatment of DCT and the related commercial VLSI chips, the reader is referred to Ref. [50].

21.3.2 Wavelet Architectures

Several VLSI architectures for one-dimensional and two-dimensional discrete wavelet transforms (DWTs) and continuous wavelet transforms have been proposed in the literature. A detailed survey of these can be found in [17]. In particular, the 2D DWT is used in the JPEG-2000 image compression standard and is computationally intensive; its VLSI realization is still an active area of research. Lewis and Knowles [8] used the 4-tap Daubechies filter to design 2D architectures. Two architectures that combine word-parallel and digit serial computations were proposed in [48].

Several systolic and SIMD arrays for 2D DWTs were proposed in [15–17, 34, 37, 39, 59]. Some recent works can be found in Refs. [65, 78]. Since there are too many contributions in this area, we are providing pointers to only some of the important ones.

21.3.3 JPEG Hardware

Several special-purpose VLSI chips implementing the JPEG baseline compression standard have been built and successfully commercialized. The majority of earlier image products were from C-Cube, LSI Logic, Zoran, Fujitsu, Intel, and Atmel [87]. Intel's i750 video processor [35, 36] consisted of two chips, the 82 750-PB pixel processor and the 82 750-DB display processor. The pixel processor can be programmed to implement the JPEG compression standard. C-Cube developed two processors, CL560 and CL550 [18]. CL550 operates at 35 MHz; data sustenance was at the rate of 2 MB/s and the compression rate was 30 fields/s. CL560 was installed with enhanced video applications. The sustenance was 60 MB/s. The chip could handle 601 frames in real time. LSI Logic announced a chip-set for JPEG compression that consisted of an L64735 DCT processor, an L64745 JPEG coder, and an L74765 color and raster-block converter and could process still-image data at up to 30 million bytes per second (Mbps). LSI Logic's JPEG chip-set is described in [53, 54]. In July 1993, LSI Logic announced the L64702 single-chip JPEG coprocessor designed for graphics and video applications in personal computers, engineering workstations, and laser printers [1]. The chip was capable of compressing and decompressing data at rates up to 8.25 million bytes per second with an operating frequency of 33 MHz. The architecture of JAGUAR, which implemented the various logic blocks of the entire JPEG compression standard as a linear superpipeline to achieve 100 million bytes per second throughput, was described in [57]. Zoran offered ZR36040 and ZR36050. ZR36040 works with an external DCT processor and controller. Data processing could vary from 15 to 21 MB/s and could process 601 CCIR frames in real time. The ZR36050 JPEG chip was designed as a superset of ZR36040, which had a video encoder and decoder in a single chip. Fujitsu developed MB86357A, which supported color components with either 8 or 12 bits/pixel with a 20-MHz clock. Atmel targeted a low-power, low-cost chip suitable for camera or video applications.

In the present scenario, image compression products are seldom developed. Still-image compression is performed by the video compression chip-sets. We found one JPEG image processor by Oak Technology Inc. The relevant features are discussed below. The product PM-36 is a fixed-function iCODEC for high-speed color and grayscale image data compression. The data sustenance is 110 Mbps. This chip can operate at 80 MHz (for raster/block mode) or 110 MHz (for block mode only). It uses four loadable Q-tables and two loadable Huffman table pairs. The chip has internal PLL and operates at 3.3 V with 5-V tolerant I/O. The JPEG standards are ISO IS10918-1/10918-2 for baseline and color images. The chip has internal buffers of 256 bytes on input and output ports. I/O's can transfer up to 280 Mbps. Figure 21.9 depicts the block diagram for the PM-36 image processor.

21.4 VIDEO COMPRESSION HARDWARE

In recent years, video communication has become an essential part of a wide range of applications. Cheap digital storage media and an abundance of digital transmission channels have triggered a huge growth in video communication. In response to the wide range of applications, various standards for coding and transmission have been proposed by the ISO and ITU groups. MPEG standards such as MPEG-1, MPEG-2, and MPEG-4 are commonly used in video communication, whereas H.261 and H.263 are standards specific to teleconferencing. Video compression

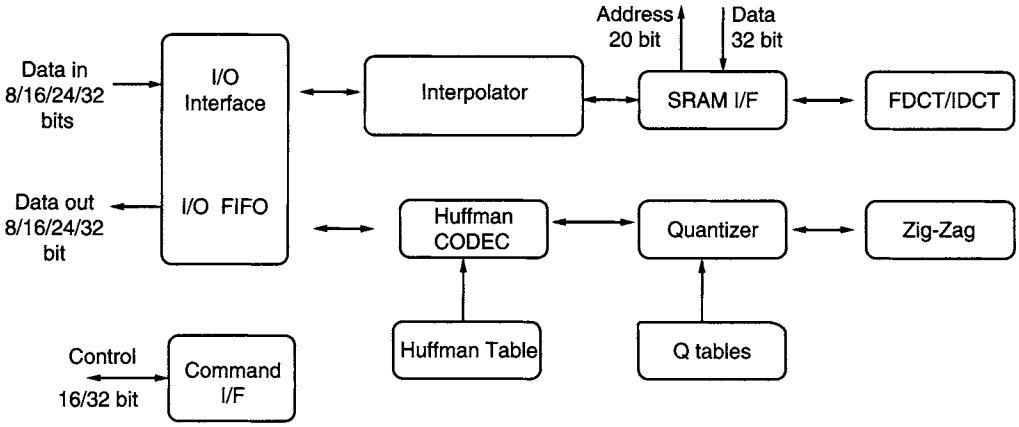

FIGURE 21.9
The JPEG image processor by Oak Technology Inc.

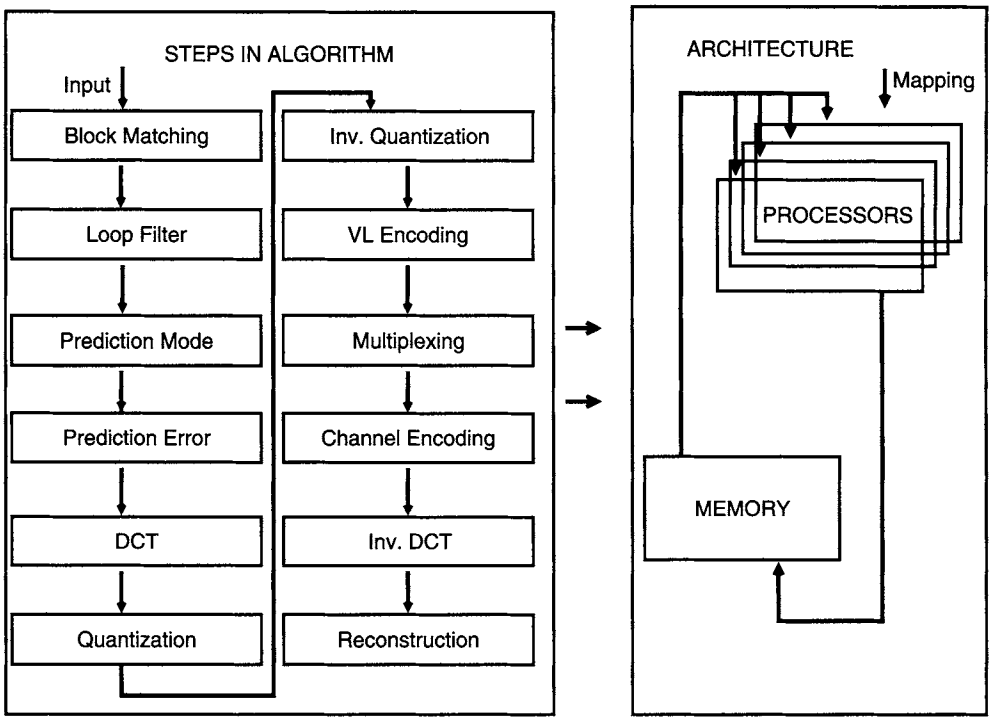

FIGURE 21.10
Functional specification in a video compression technique [67].

standards are based on a unified underlying coding scheme in which motion-compensated inter-frame prediction is used to reduce redundancy in consecutive frames and is combined with block-based transform coding to allow selective removal of irrelevant information through adaptive quantization and entropy coding [66]. Specifically, the tasks involved are DCT, motion estimation, quantization, and variable-length coding. In MPEG-4, object-based coding is additionally required for encoding video objects. The various tasks are shown in Fig. 21.10. Since these tasks are common to the various standards, the architectural implementation of these tasks is of

much interest. Two basic architectural choices for hardware implementation of video compression are (i) dedicated architectures that try to maximize performance and (ii) programmable architectures that maximize flexibility in terms of adapting to different tasks as required by different standards leading to a more cost-effective system. Dedicated algorithms use the greatest amount of parallelism and exploit completely the regularity and features of the algorithm and hence they can meet the performance criterion with a lower silicon area. DCT and motion estimation as well as entropy coding are often implemented this way. Programmable architectures are preferred when cost and flexibility are of primary concern.

2D-DCT (two-dimensional DCT) can be used as a frequency transform with L^2 computations on an $L \times L$ image block. By decoupling the 2D-DCT into two 1D-DCTs (one-dimensional DCTs) the computations can be reduced to order $2L$. 1D-DCTs are fairly common, and well-known architectures exist in the literature [63, 68, 80, 86]. Fast DCT algorithms have also been proposed by many researchers [3, 13, 70]. Alternatively, the multiplication is completely omitted by using precalculated lookup tables stored in ROM [62]. For motion estimation, block-matching algorithms [14, 38, 41, 85] are used to estimate the motion between consecutive frames. The Mean Absolute Distance (MAD) is used to estimate the motion for each $N \times N$ block in the frame. Several alternative approaches are proposed based on signal flow graphs [67]. 2D exhaustive block matching is implemented on a 2D array architecture. Hierarchical block matching has also been proposed [47, 49, 51]. Apart from the function-specific architectures, dedicated implementation of video compression has been widely reported [1, 68, 76, 83]. For example, Ruetz and Tong [68] used a dedicated seven-chip architecture for full-motion video compression. The chip-set has a separate motion estimator, DCT block, quantizer, etc., performing the full compression tasks.

Programmable architectures are more flexible than dedicated architectures and can perform various algorithms on the same hardware. Algorithms are implemented in software. General-purpose hardware does not provide adequate performance for real-time video. Generally, parallelism in the task, instruction, and data levels is exploited to enhance performance [33, 84]. Some strategies typically used for enhancement of performance are (i) increasing clock frequency through pipelining, (ii) parallelizing the datapath to exploit instruction level parallelism as in superscalar and VLIW architectures, (iii) using concurrently working coprocessors, (iv) using array processors, and (v) using SIMD and MIMD parallel models of architecture. Video coprocessors are the current trend in the commercial products for enhancement of performance retaining the programmability [82]. Fig. 21.11 depicts a coprocessor-based architecture implementation that is targeted to provide a cost-effective solution for a low-bit-rate application [89]. The design consists of a control module and a block-level coprocessor along with other I/O and memory modules.

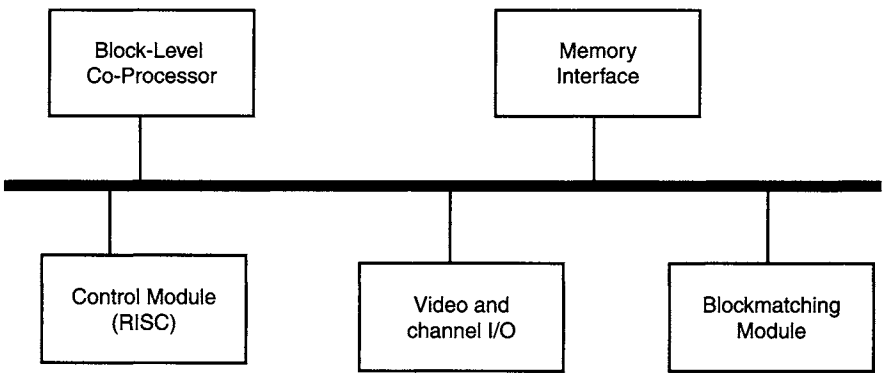

FIGURE 21.11
Co-processor-based architecture proposed by Gehrke *et al.* [89].

This architecture can implement the variable-quantization step. The block-level coprocessors perform DCT, IDCT (inverse DCT), and quantization with two parallel datapaths.

Since numerous architectures exist, we first briefly outline different architectures to point to the current trend in video compression research and then discuss a few examples in more detail.

As an example of programmable architectures, we intend to highlight features from an MPEG-2 4:2:2@HL encoder (developed by Mitsubishi R&D) architecture, which is a hybrid architecture [30]. The single chip performs video and audio encoding and performs system multiplexing of MPEG-2 main profile at main level (MP@ML). Moreover, this chip can be extended easily by parallel processing for MP@HL or 4:2:2@HL video encoding. The system consists of a VLIW-type two-way SIMD processor [30] with communication ports. The performance is 300 MIPS at 162 MHz. This encoder can handle SDTV (standard-definition television) video formats with a single chip. However, for HDTV (high-definition television) format more than six times as many computations are required. Hence the video is segmented and six parallel units work together with the help of two 8-bit communication ports. The motion compensation blocks for MPEG-2 use high computational power and this cost is eliminated by using good motion estimation in the encoder chip.

Berekovic *et al.* [55] proposed the TANGRAM coprocessor aimed at system-on-chip applications for MPEG-4 video. The processor can composite the scenes at the display. This work involves real-time warping and alpha-bending of multiple full-screen video textures. The RISC processor works independently of the decoder. The major computations performed are blending, warping, address computation, and interpolation. VHDL implementation of TANGRAM shows that a 100-MHz clock is achievable with 160K gates, with 350 Kbits of memory, and with consumption of 1 W of power with 0.35-μm technology.

Badawy and Bayoumi [88] proposed a hybrid object-based video motion estimator by using function-specific architecture. A 2-D mesh created by the architecture captures the dynamics of the video objects, and the architecture relies on a block-matching core to generate motion vectors. The video object is represented by a 2D triangular mesh. The mesh-based motion estimation algorithm works as a 10-state FSM. Using 0.6-μm CMOS, the power consumption is less than 0.5 W.

Chang *et al.* [28] proposed a bitstream parser suitable for MPEG-4 applications. Based on a bitstream analysis, only seven types of parsing instructions are needed. The core of the architecture consists of three major units (implemented by dedicated hardware implementation): (i) the functional unit, (ii) the memory management unit, and (iii) the instruction decoder unit. The functional unit performs codeword decoding, arithmetic, and logic computation. The memory management unit works to store decoded data and the parsing instructions. The instruction decoder decodes the instructions and generates data for the functional unit and address generator.

Sriram *et al.* [69] proposed an MPEG-2 decoder microprocessor (MAJC) that uses VLIW instructions suitable for multimedia applications. A high-speed VLSI architecture for discrete wavelet transforms was proposed by Chang *et al.* [79] for MPEG-4. Liu [60] proposed an MPEG decoder for embedded systems applications where performance may be relaxed to achieve a simpler and more compact hardware implementation. Features from bus-based and pipeline-based architectures are used to implement simple hardware. Lin *et al.* [19] proposed a low-power design for an MPEG-2 decoder in which Gray code encoding is proposed for increasing the correlation of the bus data transfer to reduce switching activity and hence achieve power reduction.

21.4.1 Some Detailed Examples

Due to the vastness of the literature in this area, we elaborate on only a few recent works as examples, to provide more details that will aid in the general understanding of video architectures.

21.4.1.1 Architecture for Compressed Bitstream Scaling

The idea of scaling the bitstream is generally used to reduce the rate of the constant-bit-rate-encoded bitstream. The applications that use scaling are video-on-demand, trick-play track on digital video tape recorder (VTRs), and extended play recording on digital VTR. The various scaling techniques have different hardware complexities and performance trade-offs. We concentrate on the various bitstream scaling architectures discussed by Sun *et al.* [31]. They perform scaling irrespective of the way in which the stream was encoded. The various available architectures are (i) scaling higher frequencies, (ii) scaling by requantization, (iii) scaling by reencoding the reconstructed pictures with motion vectors and coding decision modes extracted from the original high-quality bitstream, and (iv) scaling as in architecture (iii) above, but with new coding decisions based on reconstructed pictures. The first two architectures, (i) and (ii), are important for VTR applications and the last two architectures, (iii) and (iv), are important for VOD applications.

In architecture (i), bitstream scaling is performed by scaling higher frequencies. This is achieved by using two parallel paths in the architecture. In one path, a Variable-Length Decoder (VLD) parser is used to obtain the codeword length and a bit-allocation analyzer collects all the AC bit-counts for every macroblock in that frame. The scaling factor is determined based on profiling these AC bits, and the linearly scaled profile is sent to the final rate-controller block, as shown in Fig. 21.12. The other path consists of a delay proportional to the first path and another VLD parser that delivers all the codeword blocks to the rate controller. The rate controller accepts only when the running sum of DCT codeword bits exceed the scaled profile value less the EOB bits. The rest of the DCT codewords are excised.

Architecture (ii) has variable-length coding and additional quantizer and dequantizer blocks. In this architecture, shown in Fig. 21.12, the rate-controller functions are based on a requantization scheme. The DCT coefficients are first dequantified to a finer grain quantization and then requantified back with a coarser quantizer scale using the profiling information from the first path, adaptively determining the quantization steps.

In architecture (iii), the motion vector and macroblock coding decision blocks are first extracted, and reconstructed pictures are obtained by normal decoding procedures. Scaling is then performed by reencoding and reconstructing, using the motion vector and macroblock decision extracted earlier. Applying reencoding after a full decoding eliminates the need for separate motion estimation and decision computations. Figure 21.13 describes architecture (iii).

Architecture (iv) is a modification of architecture (iii), where new macroblock decision modes are computed during the reencoding of the reconstructed picture. It is theoretically important to obtain decision modes from the scaled bitstream rather than from the initial bitstreams.

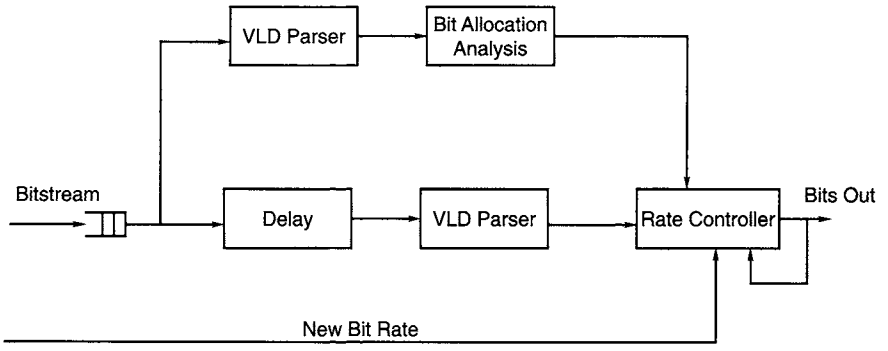

FIGURE 21.12
Bitstream scaling by cutting high frequency as proposed by Sun *et al.* [31].

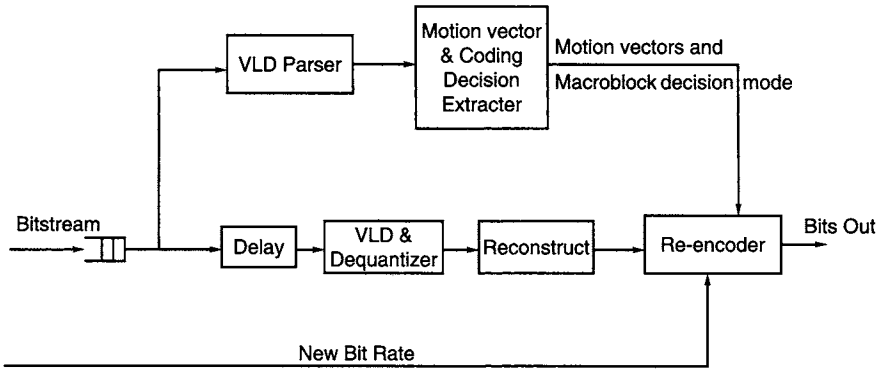

FIGURE 21.13
Bitstream scaling by reencoding using original motion vector and decisions as proposed by Sun *et al.* [31].

21.4.1.2 Transportation of MPEG-4 on the Internet

Let us now focus on a recent contribution by Wu *et al.* [24] on transporting MPEG-4 video over the Internet. Any transporting mechanism must include source-rate adaption, packetization, a feedback system between the sending and the receiving ends, and an error control scheme. This work focuses on (i) an end-to-end feedback controller that helps estimate the network bandwidth by the packet-loss information, (ii) an MPEG-4 video encoding rate controller that adaptively changes the output rate according to the network bandwidth, and (iii) a packetization scheme utilizing MPEG-4 video-object plane features. We highlight some important ideas regarding the above three aspects here.

The feedback controller is designed according to the RTP/RTCP standard. The network bandwidth is estimated indirectly through a feedback from the receiver end. The feedback algorithm modifies the bandwidth according to the packet-loss ratio sent back from the receiver. The output rate of the MPEG-4 is adjusted through the feedback scheme to keep the the packet-loss ratio below a required threshold value.

The role of the Adaptive Encoding Rate Controller (ARC) is to maintain the output rate of the encoder based on the feedback controller estimates. Figure 21.14 depicts the structure of this ARC unit. The ARC uses an accurate second-order R-D model that introduces two important components, namely, M, which is MAD, and H, which denotes bits for header, motion vector, and shape information in the quadratic model. By introducing M (MAD), this method ensures scalability with coding complexity. The first stage, initialization, as shown in Fig. 21.14, sets the initial buffer size and output rate. In the second, preencoding stage, the target bit-count is first estimated at three different levels, namely, the frame, object, and macroblock levels, to improve estimation accuracy. The bit-count is further adjusted based on the buffer information for each video object, and then quantization parameters are computed. The third stage, encoding (Fig. 21.14), uses the measurement of actual bit-rates. Macroblock-layer rate control may also be activated at this stage. The final, postencoding stage is responsible for updating the R-D model parameters. At this stage, a decision is made to balance the usage between shape information and texture information. Moreover, buffer overflow is controlled by skipping some frames.

Wu *et al.* also proposed a packetization scheme appropriate for MPEG-4 transportation [24]. The proposed packetization algorithm uses a packet size that is the minimum of the maximum transit unit and the current VOP (Video Object Plane) size. When the VOP is too large to fit into one packet, then it is broken up into multiple packets. The proposed algorithm is designed to minimize the number of packets and also the dependency among packets. If a single VOP does not fit into a packet, a maximum number of Motion Blocks (MBs) are collected in one packet.

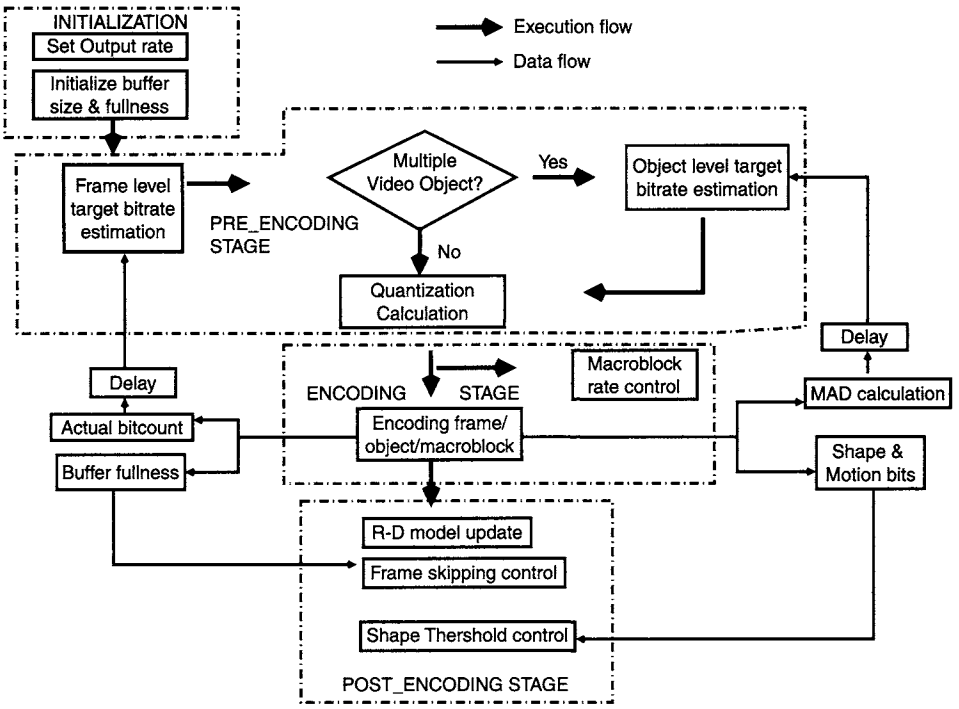

FIGURE 21.14
Adaptive rate controller for transportation of MPEG-4 video over the Internet as proposed by Wu *et al.* [24].

The VOP header information is copied into all the packets that are part of the same VOP. MBs from two VOPs are never put into a single packet, even if space is available, to reduce conflicting header information.

21.4.1.3 MPEG-4 Video Codec by Toshiba

Recently many works have focused on MPEG-4 video and audiovisual codecs [40, 91]. We will focus on hardware implementation of a complete MPEG-4 video codec by Toshiba and discuss various components in the codec.

The MPEG-4 architecture model developed by Toshiba [58] consists primarily of a 16-bit multimedia-extended RISC processor, MPEG-4 audiovisual LSI containing three 16-bit RISC processors and 16-bit DRAM, and a dedicated hardware accelerator suitable for IMT-2000 applications. The objective of the architecture is to achieve higher programmability with low power consumption. The RISC processors can perform complicated tasks on the hardware accelerator such as bitstream syntax processing, rate control, and parallel operation control [58]. Moreover, the hardware accelerator itself performs critical tasks such as motion estimation, motion compensation, and DCT. The accelerator has its own memory and can perform independently of the RISC processor, making parallelization possible. Low power consumption is feasible due to the parallelization [58].

In Fig. 21.15, we explain the functionality of the video codec architecture. The RISC processor has a 4-KB direct-mapped instruction cache (I-cache) and 8 KB of local memory. Variable-length coding and decoding are performed for the DCT coefficients by the processor. The processor also analyzes and synthesizes the bitstream, monitors an intra-AC/DC prediction, and monitors the control of the hardware accelerators. The instruction cache is responsible for the reduced area of

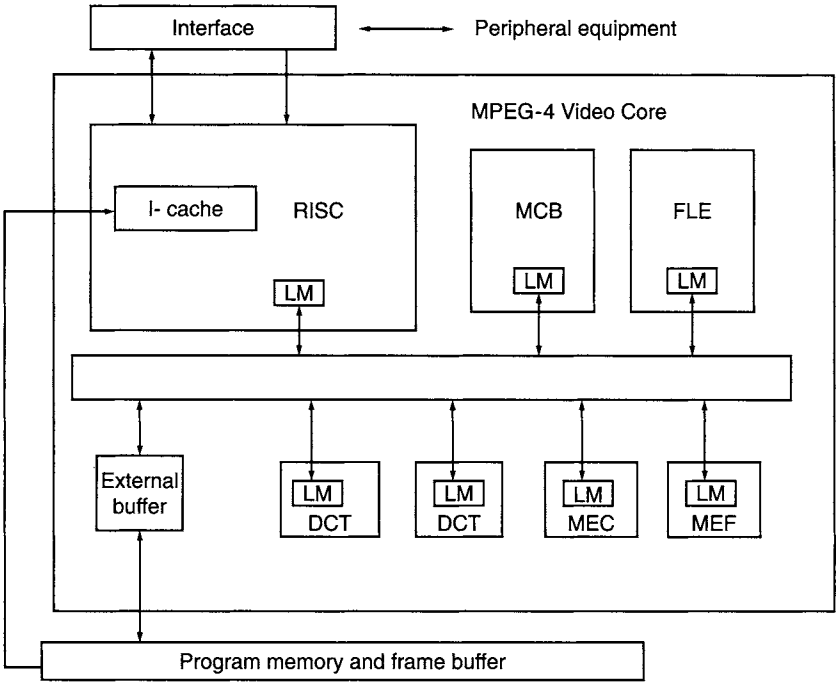

FIGURE 21.15
MPEG-4 architecture for video core LSI [58].

the chip (since on-chip SRAM is not required) and reduced memory access. The DCT block either performs DCT and quantization (Q) or performs IDCT with IQ (Inverse Q). Both the DCT blocks can operate simultaneously for speed-up. The motion compensation block MCB, MEF, and MEC handles the motion estimation with a variety of search windows using a three-step hierarchical search.

The FLE block is used for postfiltering the hardware accelerator for better image quality. The frame buffer and firmware are stored in the external DRAM. The hardware accelerator uses local memory for parallel operations. As a consequence of the reduced external memory access, effective memory access time is also reduced. The total local memory is 5.3 KB, which occupies a reasonably small area. DMA controls all the memory accesses, and power-efficient multiplexors are used for data transfer, rather than buses with large capacitance.

The processing of the video core is performed by an external firmware on the central processing unit (CPU). A few threads run concurrently to generate the desired amount of parallelism. The encoding thread handles six blocks of data with 6-stage pipelining for Y0, Y1,Y2, Y3, Cb, and Cr. By this pipelining, 15 frames of encoding and decoding can be processed with QCIF (quarter common intermediate format) at a 60 MHz/s clock rate.

Effort has been made in this architecture to reduce power via various means. The architecture also uses clock-gating to reduce clock power. When the hardware accelerators are idle, the clock circuit is stopped. The RISC processor has its own instructions to go into sleep mode by stopping its own clock. Embedded DRAM cuts down the power consumption in I/O circuits. Moreover, techniques to optimize the page and word size are used to obtain better power efficiency. A low-power motion estimator has been adopted using techniques similar to those outlined in Ref. [2]. The motion estimator achieved almost 50% of the total power savings.

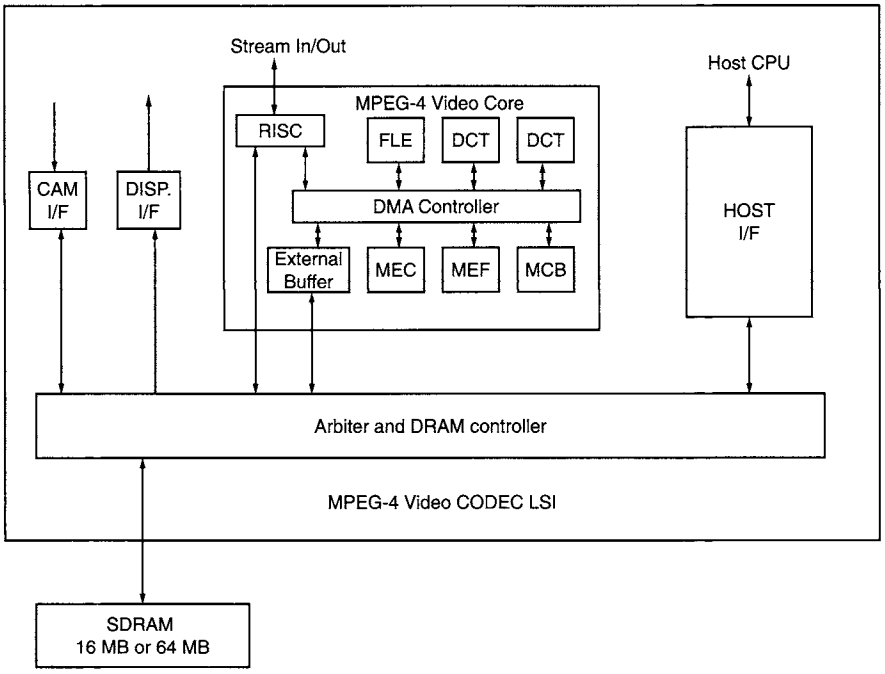

FIGURE 21.16
MPEG-4 architecture for video codec LSI [58].

The MPEG-4 video codec, as depicted in Fig. 21.16, consists of the RISC core described earlier and some peripherals. A CIF CMOS image sensor and an LCD controller are directly connected to the inputs and the outputs. The codec uses an ordinary bus interface along with a bitstream serial interface for multimedia applications. Additional memory (16- or 64-Mbit SDRAM) can be connected for program storage and frame storage.

21.4.1.4 The MOVIE Project

Another interesting architecture involves hardware support for software-only real-time video processing as proposed by Charot *et al.* [26]. This approach, termed MOVIE, utilizes a high level of parallelism. The computation and data access in algorithms such as DCT, quantization, and motion estimation are regular [26] and hence SIMD with a linear array processor is used. The MOVIE chip consists of the following components, as shown in Fig. 21.17: (i) four 16-bit data ports, (ii) three 16-bit video ports, (iii) two instruction ports to send to the PE_c processor array for computation and to the PE_{io} I/O processor for I/O operations, and (iv) a memory interface for external memory. Each PE_c is a 16-bit processor and the PE_{io} is a 32-bit I/O processor with the same internal architecture for low-overhead software development. The instruction is controlled by 40-bit instructions, and decoder units are used internally for decoding. Since SIMD allows all processors to use a single instruction, one I/O processor and one controller are sufficient. Instructions can also be pipelined to generate greater throughput. One communication operation between the neighboring processors in the systolic array of processors can be done simultaneously with the computation. The instruction formats use two independent fields: one for computation and one for communication. The MOVIE approach relies on C-Stolic software support for real-time video processing.

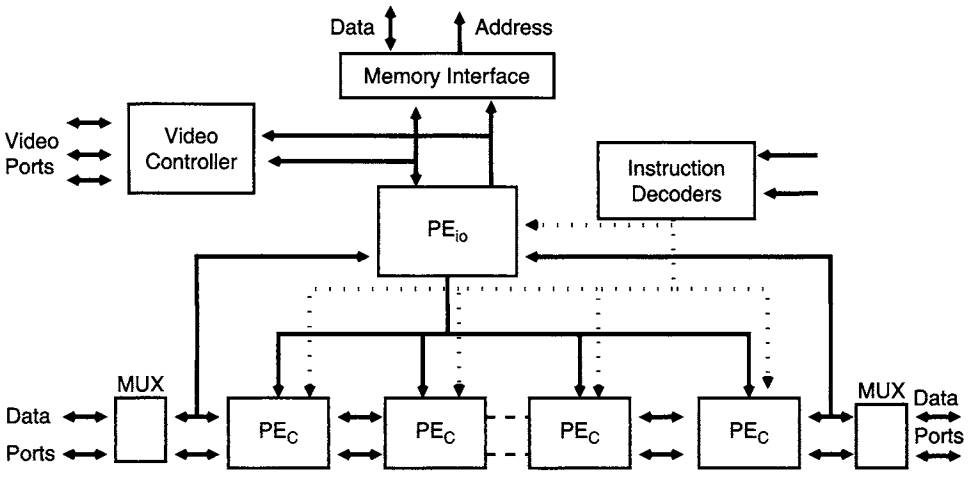

FIGURE 21.17
Architecture for real-time video processing [26].

21.4.2 Commercial Video and Audio Products

We focus on the industry-released compression products available for video and audio compression, most of which come as single-chip solutions. We combine the descriptions of audio and video products since they are supported together in most multimedia systems. Products differ in terms of the standards they support, input and output formats, and compliance with various application standards. Products support different standards, such as MPEG-1, MP@ML MPEG-2, and 5.1 Dolby Digital. The inputs can be bitstreams (packetized elementary or elementary streams) in serial or parallel formats. The output format can be of various types, such as NTSC or PAL. The products comply with DVD, Digital Video Broadcasting (DVB), Digital TV, HDTV, or SDTV applications. Recent trends also indicate that most codecs are for combined audio and visual applications, with synchronized audio and video units. An important reference in this area is the book by Bhaskaran and Konstantinides [87]. Here, we briefly outline various products, examining a few in further detail.

Earlier audio products were either dedicated or programmable DSPs with glueless interface to memory and a host controller. Crystal semiconductors produced the CS4920 programmable DSP chip for MPEG-1 or AC-2 audio decoding. CS4922 from the same manufacturer could support both MPEG-1 and MPEG-2 layers 1 and 2. L64111, developed by LSI Logic, is a two-channel MPEG audio decoder supporting MPEG-1 and MPEG-2. SGS-Thomson developed the STi4510 MPEG-1 audio decoder, which supports all MPEG-1 sampling rates. Most processors from Texas Instruments (TI) can be run as audio decoders. Moreover, TI produced a dedicated audio decoder, TMS320AV110. Zoran produced programmable audio, DSP, which could potentially handle six-channel AC-3 decoding and two-channel MPEG audio decoding for future generation audio products.

Some of the video products mentioned here are the predecessors of the current products described below. These chips have extensive applications in set-top boxes for interactive TV, HDTV, etc. C-Cube developed the CL480VCD and CL480PC, AViA-550, and AViA-502 audio/video decoders. IBM produced chip-sets MPEGSE10, MPEGSE20, and MPEGSE30 for MPEG-2 video encoding. IBM also had an MPEGCD20 MPEG-2 audio/video decoder. TI developed AV7000, which is a complete set-top decoder. It has a 32-bit RISC controller, A/V decoder, graphics accelerator, and traffic interface manager. Some of the other important products were L64002,

L64005, and L64020 A/V decoders from LSI Logic; STi3400, STi3430, and STi3520 MPEG-2 A/V decoders from SGS-Thomson; HDM8211M from Hyundai Electronics; and TC81201F and TC81211F from Toshiba.

We describe a number of current audio and video products in the rest of this section. Most of the information was obtained from the datasheets provided by the vendors. We found that most recent video products come with the audio component as well.

21.4.2.1 C-Cube Video Products

The C-Cube MPEG-2 Video Codec (DVXCEL) is a single-chip MPEG-2 video and system encoder and decoder. This chip can simultaneously encode and decode MPEG-2 video with audio synchronization to make it usable for Digital Video Recorders (DVRs). The device has a 16-/32-bit host interface with a DMA target of primary, secondary, and tertiary hardware I/O ports (8 bits each) for bitstream data transfer. The device has 8- or 10-bit 656-video input as well as an 8-bit video output port. The 110-MHz micro-SPARC RISC core consists of hardware for video compression preprocessing, motion estimation and compensation, DCTs, IDCTs, variable-length encoding and decoding, and high-quality video scaling and compositing. This device, developed by C-Cube, uses 8 MB of external SDRAM and has an on-chip controller with a 64-bit-wide interface. The chip consumes 1.8 W of power at a voltage of 2 V. The block diagram is shown in Fig. 21.18.

The ZiVA-5 DVD processor was also developed by C-Cube. ZiVA-5 is a single-chip compatible with DVD-video, DVD-audio, DVD-VR, Chaoji-VCD (CVD), SuperVCD, VCD, CD-DA, and

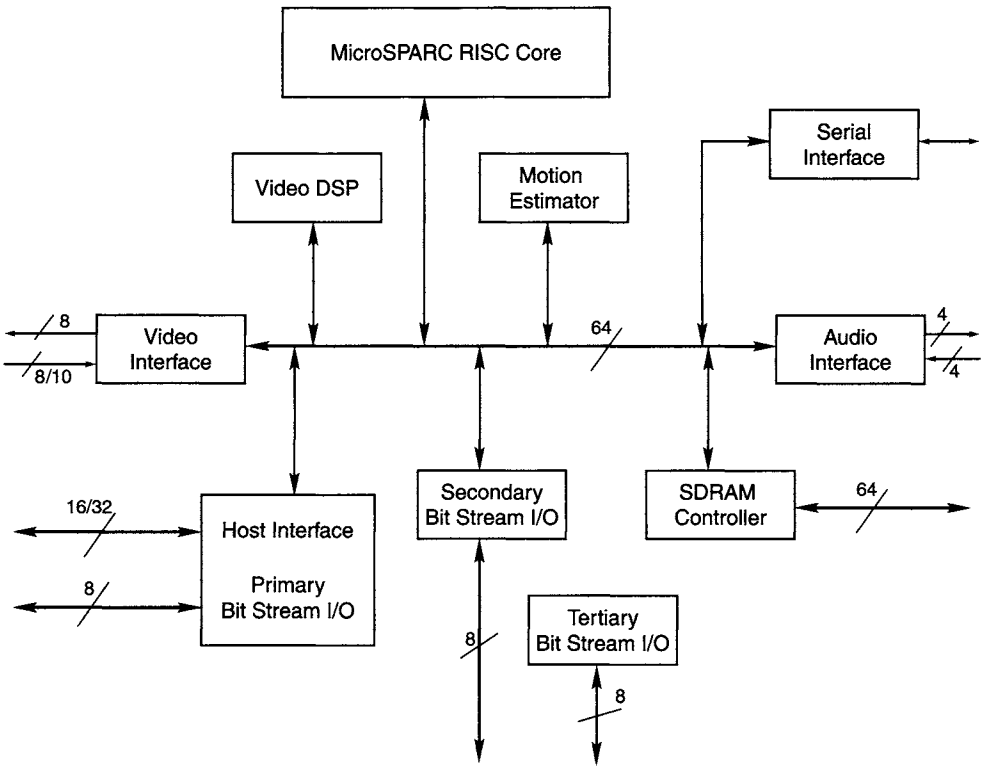

FIGURE 21.18
The DVxcel MPEG-2 video codec by C-Cube.

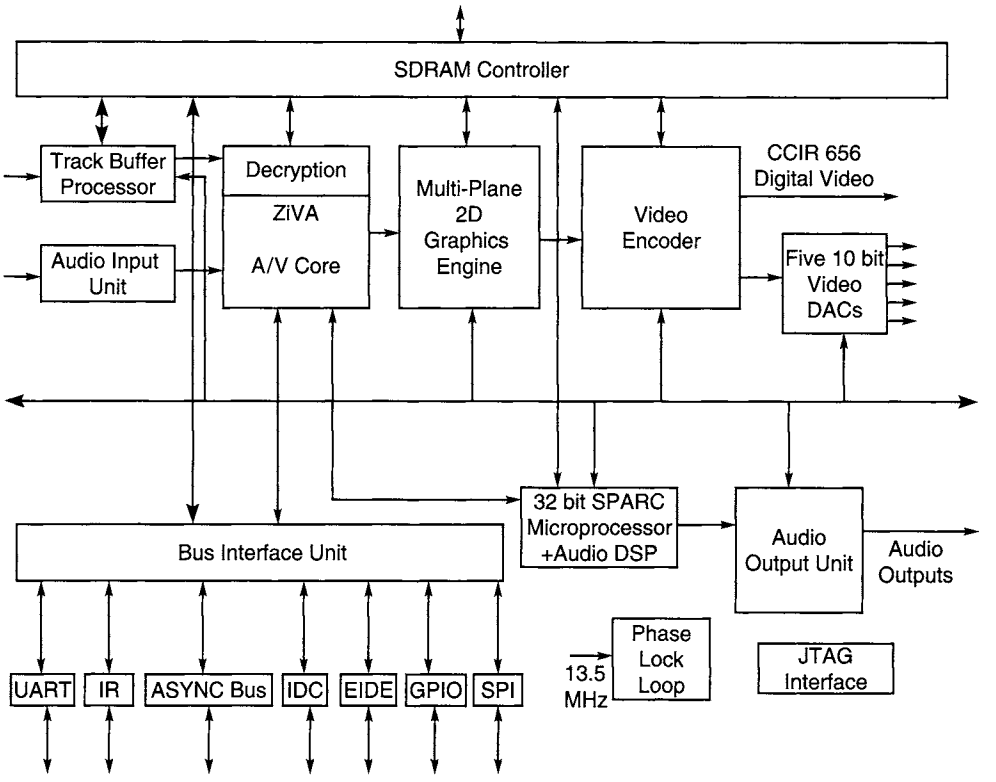

FIGURE 21.19
The ZiVA-5 DVD system processor by C-Cube.

CD-ROM formats. The chip has a dual-audio DSP architecture with a ZiVA audio/video decoder. The chip also includes a high-performance 2D graphics engine and a video encoder with five 10-bit, 54-MHz DACs for NTSC, PAL, and 480P formats. The DVD-audio support includes MLP decode, content protection for prerecorded media, and audio watermark detection. Moreover, ZiVA-5 includes an MP3 audio codec. The block diagram is shown in Fig. 21.19.

21.4.2.2 Lucent System-Layer Decoder

Lucent Technologies developed an AV6220A MPEG-2 system-layer decoder in 1996 that works with multiple audio/video decoders. The device is compatible with system-level DVD requirements and MPEG-2 specifications. This device can handle serial or byte-wise parallel input streams (transport stream format) and may also receive data in transport stream format for CD-ROM applications. The product also provides a clock recovery at 27 MHz and helps synchronize video and audio data. The device uses 0.55-μm CMOS technology, requires 5.0 V, and consumes less than 1 W of power. The block diagram is shown in Fig. 21.20.

21.4.2.3 LSI Logic Products

LSI Logic developed the L64105 MPEG-2 audio/video decoder in 1997. This decoder chip works for MPEG-2 decoding standards. This chip can be used inside a decoder system chip, which includes other units such as L64108 transport Demux, audio DAC, PAL, and the NTSC format converter. This chip can also be used in set-top box applications or in PC-based MPEG-2 applications. This chip is an improved version of L64005 with enhanced video quality and with

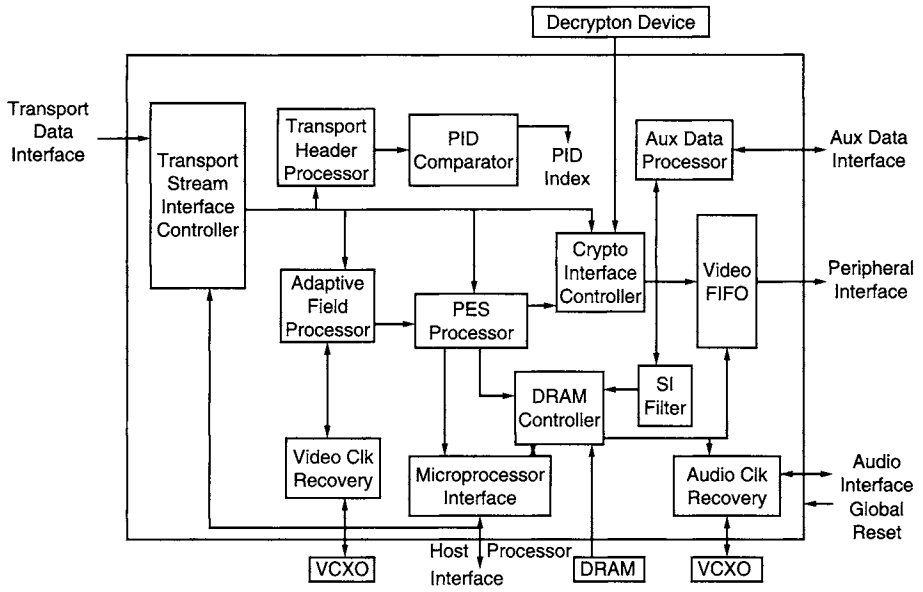

FIGURE 21.20
The AV6220A MPEG-2 system-level Demux by Lucent Technologies.

low host-processor overhead. The chip is built using 0.35-μm CMOS technology with a 3.3-V supply voltage.

The video decoding unit is fully compliant with the system-level MPEG-2 standard ISO 13818-2. The decoding unit processes the MPEG-2 bitstream including the MPEG-2 program streams PS and PES. MPEG-1 bitstreams are also decoded including an MPEG-1 system layer. The decoder can handle image sizes up to 720 x 480 pixels at 30 frames/s for NTSC and 720 x 576 at 25 frames/s for PAL.

The audio decoder follows MPEG audio decoding standards with support for linear PCM data. It also decodes dual-channel MPEG audio layer I and layer II ISO 11172-3. The bit-rates can range from 8 to 448 Kbps. The sampling frequencies can be 16, 22.05, 24, 32, 44.1, or 48 KHz.

The L64105 chip supports on-screen display (OSD) and display management. The graphics support unit can handle a programmable channel buffer and display buffer size. The chip also has a data- and error-handling unit. The data can be maintained at a rate of 20 Mbits/s for the PS mode and 40 Mbits/s for the PES mode. Input data are in 8-bit parallel mode and output is 8 bits (Y, Cb, Cr) in slave mode. The error-handling unit maintains the video during channel error, whereas the audio unit is muted. The chip also has a cost-effective system implementation with a preparser accepting PES, ES, and PS. The system is directly connected to one or two 16-bit SDRAMs. The chip operates at a 27-MHz clock and can directly interface with available NTSC/PAL encoders and audio DACs. The circuit diagram is shown in Fig. 21.21.

LSI Logic also developed an L64118 MPEG-2 transport controller with an embedded MIPS CPU in 1998. This device has combined set-top box control along with the logic for a communication system and digital broadcast system. The TinyRISC 32-bit CPU provides the transport and system support as well as general-purpose system control. This system integrates with other satellite and cable single-chip channel decoders as well as the L64105 MPEG-2 A/V (audio/video) decoder, which was discussed earlier. THE MPEG-2 transport and system Demux can handle 32 packets simultaneously, including video, audio, and general-purpose data services. The system also supports a DVB-compliant descrambler block with increased security. The external system

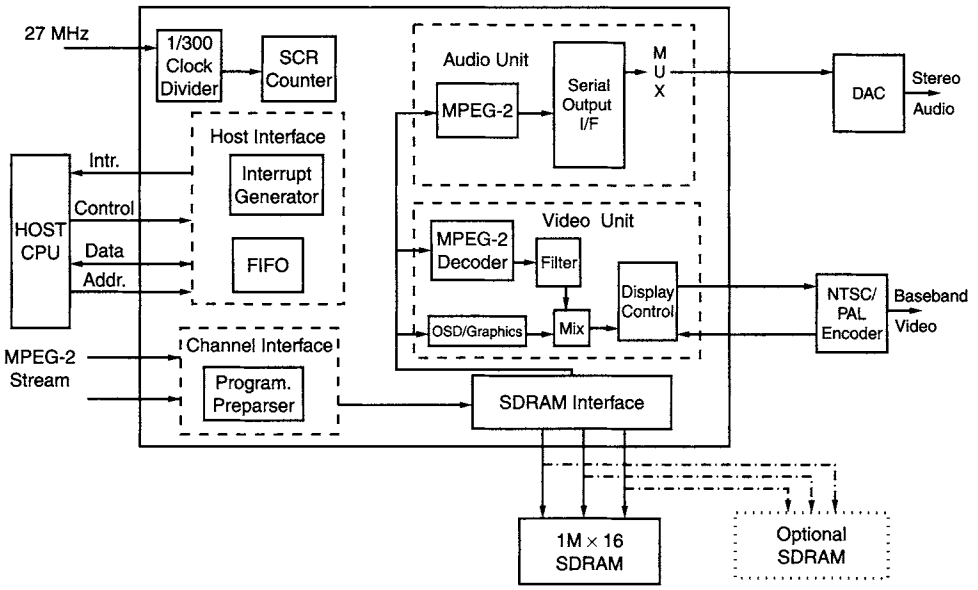

FIGURE 21.21
The L64105 MPEG-2 audio video decoder by LSI Logic.

bus communicates through serial, parallel, SmartCard, and interface ports. The memory support can be up to 16 MB using 16- or 64-Mbit SDRAMs. This system is built via 3.3-V G10-p-cell-based technology developed by LSI Logic.

The LSI Logic chip SC2000 provides a multimedia chip (released in 1999) that contains DVB transport and MPEG-2 decoding, a 2D 5-plane graphics engine, a multistandard video encoder, audio DACs, and a RISC MIPS CPU. The chip operates at 108 MHz and includes a unified OSG, CPU, transport memory architecture, and a dedicated A/V memory. This chip complies with ISO/IEC 13818 MPEG-2 MP@ML standards and provides DVD support. The SC2000 chip has a SmartCard interface, a UARTs/ modem codec interface, and an I^2C compatible interface. This chip provides Dolby AC3 support and external CPU support. It also has general-purpose I/Os, IEEE1248 ports, and a general-purpose DMA controller.

SC2000 uses a dual-memory architecture that helps the performance. The dedicated A/V memory bus provides the high bandwidth without hampering CPU and graphics performance. CPU and OSG memories are unified, delivering higher graphic performance. The block diagram is shown in Fig. 21.22. The SC2005 single-chip source decoder designed by LSI Logic in 2000 has almost the same functional features as SC2000. However, the latter is integrated with NDS ICAM technology accommodating NDS Videoguard conditional access. The SC2005 chip also integrates an IDE/ATA interface.

21.4.2.4 NEC Audio Video Codec

NEC Electron Devices launched two audio-video codecs, μ PD61051 and μ PD61052. Video encoding can be performed at a constant bit-rate or a variable bit-rate. The products support MPEG-2 video, MPEG-1 video, MP@ML, and SP@ML (simple profile at main level). Video decoding is performed using a constant bit-rate or a variable bit-rate. The formats supported are MPEG-2 and MP@ML. The audio formats are MPEG audio layer 2 and Dolby Digital. The products use a program stream/transport stream, DVD, D-VHS (for MPEG-2), and Video CD

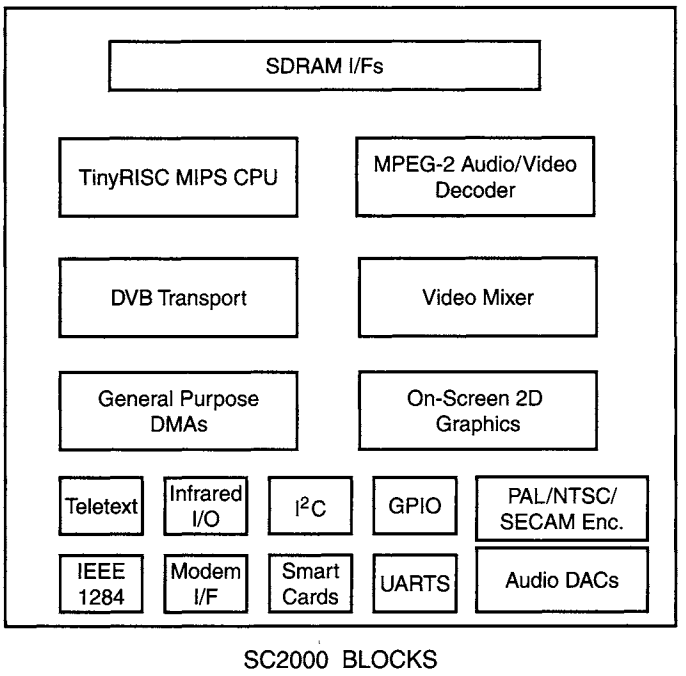

FIGURE 21.22
SC2000: a single-chip source decoder by LSI Logic.

for MPEG-1. μ PD61051 and μ PD61052 have a horizontal resolution conversion (720 pixels to 544, 480, or 352 pixels). The products operate at 27 MHz.

21.4.2.5 Audio Video Products by Fujitsu

Fujitsu developed the MB87L2250 MPEG-2 transport video and audio decoder with 32-bit RISC CPU. This product was released in 1998. This single chip is an audio/video decoder coupled with the transport Demux and a 32-bit RISC CPU. The MPEG decoding and transport Demux is implemented in the hardware. The chip operates at 54 MHz and uses 2.5-frame architecture. The memory requirement is only one 16-bit SDRAM. Along with the RISC CPU, the chip also has a 1-KB I-cache. MB87L2250 also features a DVD descrambler, IR receiver, FR30/68 K/SPARClite processor interfaces, 2-, 4-, 6-, or 8-bit OSD, and a 16-bit programmable general-purpose I/O.

The transport Demux allows parallel/serial input format for the transport stream. The maximum input data rate is 60 Mbps. The MPEG-2 transport stream is ISO/IEC compliant. The MPEG-2 video decoder uses MP@ML, ISO/IEC 13818-2, MP@LL, and SP@ML formats. The MPEG-1 stream is decoded as DVB compliant. The audio decoder features ISO/IEC 11172-3 (MPEG-1 audio) and ISO/IEC 13818-3 (MPEG-2 audio in two channels). The audio decoder works with various sampling rates such as 32, 44.1, 48, 16, 22.05, and 24 KHz. The audio decoder also supports all bit-rates. Both the video and audio decoders have error detection and concealment schemes. Both video and audio decoders support different DACs. The block diagram is shown in Fig. 21.23.

Fujitsu also developed the MPEG2 single-chip audio/video encoder in 2001. This chip integrates a video encoder, an audio encoder, and system encoders. The video encoder works for MPEG-1- and MP@ML-compliant MPEG-2 standards. The video format can be NTSC or PAL. The maximum bit-rate is 15 Mbps. The audio encoder unit follows the MPEG-1 layer 1/2 standard. The maximum bit-rate is 448 Kbps. The system mux operates at a constant or variable

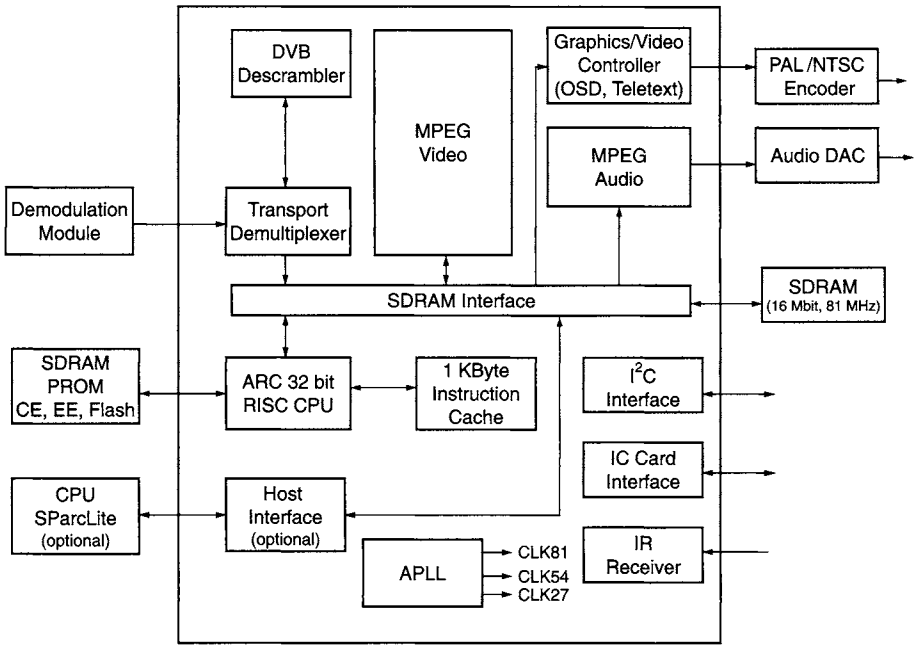

FIGURE 21.23
ML87L2250: MPEG2 decoder with integrated 32-bit RISC CPU by Fujitsu.

bit-rate with a maximum of 20 Mbps. The output format can be PS, TS, PES, or ES. This unit has an external memory SDRAM (four 16-Mbit memory chips or two 64-Mbit memory chips). The unit has an internal 32-bit RISC processor with a time base corrector. This chip can be useful for video transmission systems, telephony systems, digital video recorders, and other applications. The block diagram is shown in Fig. 21.24.

21.4.2.6 ATMEL Digital Audio Decoder

ATMEL has developed the AT76C2XX Dolby Digital AC-3 decoder. The decoding is performed on a 5.1-channel Dolby Digital (AC-3) audio stream. The chip supports all Dolby Digital configurations. The chip supports 8 audio channels and also decodes MPEG-2 and AAC. The audio input and output provide 24-bit precision at sample rates of 32, 44.1, 48, and 96 KHz. The chip also receives data from an IEC 958-compatible interface (AES/EBU or S/PDIF) or directly from an A/D converter. All 8 audio channels that the chip supports are compliant with the 5.1-channel Dolby Digital standard. The chip is based on the ATMEL 24-bit DSP. This solution does not require any external SRAM. The audio data output is compatible with the available audio DACs. The chip has 64 pins and requires 3.3 V. The AT76C2XX chip can handle a Dolby Digital bitstream encoded at a rate of 640 Kbps at 48 KHz. Moreover, it can generate audio samples at 96 KHz. The architectural features are a 45-MHz instruction clock with two 24-bit datapaths and one 40-bit datapath with a 16-bit addressing mode. A three-stage pipeline is used. A high degree of parallelism is due to the 40-bit instruction wording. The DSP has 4K $\times$ 40 program memory, 8K $\times$ 24 X-data memory, and 4K $\times$ 24 Y-data memory. The DSP also has two 56-bit accumulator registers and performs multiply, accumulate, and convergent rounding in one clock cycle. DSP has two maskable interrupts one unmaskable interrupt, and two PLLs: one for internal logic clocks and one for D/A converter clocks. This implementation is awaiting the evaluation process by Dolby Digital Laboratories.

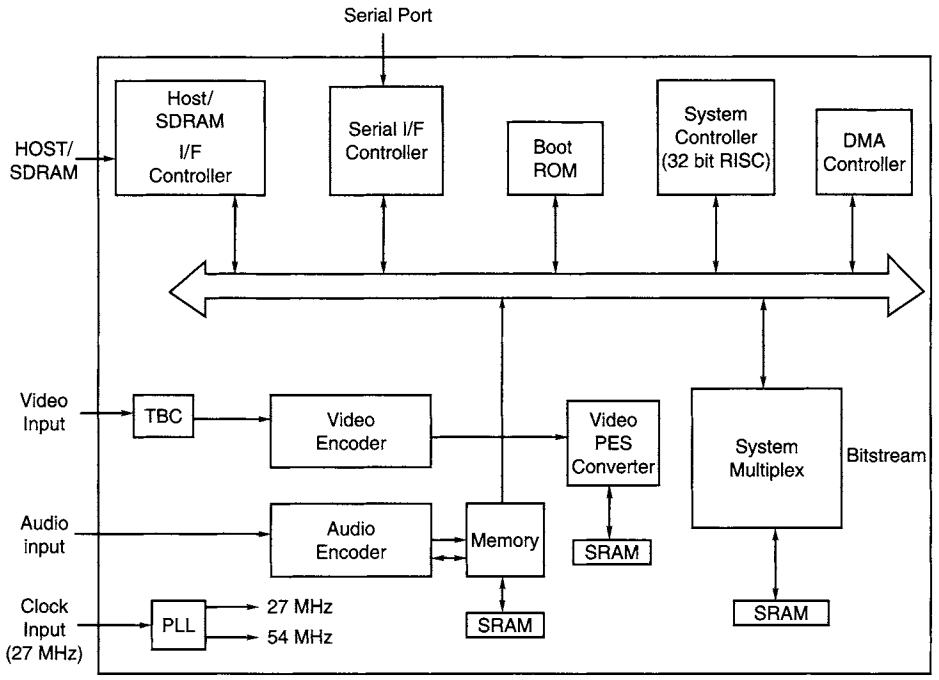

FIGURE 21.24
MPEG-2 system encoder LSI by Fujitsu.

Other ATMEL products include an accelerator for video conferencing and demodulators for digital reception and transmission.

21.4.2.7 Products by STMicroelectronics

STMicroelectronics developed the STi5518 single-chip set-top box decoder in 2001. This chip is designed to be used in pay-TV set-top boxes. It has a 32-bit CPU, a dedicated block for transport Demux and descrambling, video and audio decoders, and low-cost satellite receivers. This chip uses a sector processor and a CSS decryption block in the transport interface and a subpicture decoder in the video decoder. Moreover, the audio decoder handles Dolby Digital audio. These features allow this chip to be used as a DVD-compatible set-top box. The ATAPI interface promotes connection of DVD-ROM and hard disk drives. Thus, live-TV pausing and timeshifting (trick modes) can be handled. This product is compatible with other STMicroelectronics products such as STi5500.

The video decoder supports MPEG-2 MP@ML with zoom-in/zoom-out facilities, PAL to NTSC and vice versa conversion, and DVD and SVCD subpicture decoders. The video decoder also supports 2 to 8 bits/pixel OSD options. The PAL/NTSC/SECAM encoder is interfaced. The output formats supported are RGB, CVBS, Y/C, and YUV with 10-bits. The RISC processor operates at 60 MHz and has a 2-KB instruction cache, 2-KB data cache, and 4 KB SDRAM as an additional data cache.

The audio decoder uses 5.1-channel Dolby Digital MPEG-2 multichannel decoding and 3 × 2-channel PCM outputs. The decoder produces IEC60958–IEC61937 digital outputs. The DACs are macroversion 7.0/6.1 compatible with a shared-memory interface. The DACs can support one or two 16-Mbit or one 64-Mbit 125-MHz SDRAM. This chip is integrated with UARTs, SmartCards, I^2C controller, and 3 PWM outputs, modem support, and a 44-bit programmable I/O.

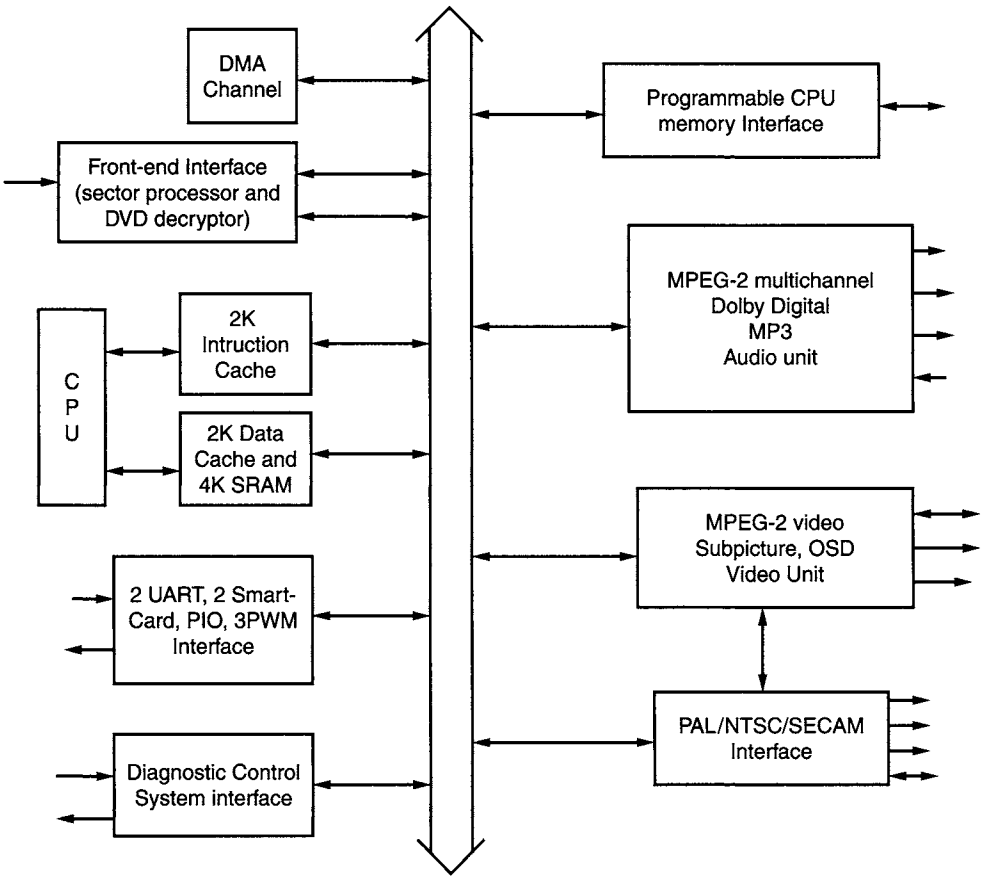

FIGURE 21.25
Single-chip decoder by STMicroelectronics.

The transport Demux includes a DES and DVB descrambler and 32 PID support, and it can handle serial and parallel input. The block diagram is shown in Fig. 21.25.

STMicroelectronics produced the STi7000 HD MPEG-2 video decoder and display in 2000. This chip is a real-time high-definition MPEG-1 and -2 video decoder. The video rates are 1920 × 1088 × 30 Hz interlaced or 1280 × 720 × 60 Hz progressive. Memory reduction schemes are used to restrict the memory requirements to 64 Mbits for hard drive applications. Video format converters are integrated and an MPEG-2 MP@HL-compliant video decoder is used. OSD with 1, 2, 4, or 16 bits/pixel and full, one-half, or one-third resolution is supported. The output can be either 8-bit, 27-MHz CCIR 601 compatible or 24-bit, 81-MHz 4:4:4 output. D1 video input and a standard 8-bit interface for microcontrollers and compressed data inputs are used. A clock speed of 27 MHz is required for the entire chip. The chip uses 0.35-μm CMOS technology with 3.3 V.

STMicroelectronics released the STi4600 Dolby Ac-3 Audio decoder in 2000. This chip decodes 5.1 and 2 channels with DVD standards. This device also decodes MPEG layers I and II with a precision of 24 bits. The input can be a serial or parallel MPEG-2 PES stream. The controller interface can be I^2C or 8-bit parallel. External memory is not required if the delay requirement is higher than 35 ms. The output can be up to six channels. The PCM mode is transparent down-sampling 96 to 48 KHz. Prologic decoders are used. This chip uses a 3.3-V supply voltage.

The other products related to digital video chips are STi5500, STi5505, STi5512, and ST20TP4. STi55XX is a set-top/DVD backend decoder with host processors. STi5512 is a tuned broadcast application and STiT20TP4 is a programmable transport IC for broadcast and DirecTV applications.

21.4.2.8 Single-Chip Decoder by Zoran

Zoran Inc. released the ZR36215 single-chip audio/video decoder. This chip supports SuperVCD standards along with Video CD1.1/2.0. The product also supports CD-DA. This chip has a video decoder that decodes MPEG-2 and MPEG-1 with a variable bit-rate. The MPEG-2 audio decoder uses a sampling rate of 44.1 or 48 KHz. An audio decoder with MPEG-2 provides multiple channels and is backward compatible with MPEG-1. MPEG-1 standards are used for VideoCD. The outputs can be mono or stereo. This chip supports multiple features of Karaoke. The programmable 40-MIPS DSP can process various audio algorithms to enhance audio effects in a two-speaker environment (Fig. 21.26).

Zoran produced the ZR38601 Programmable Digital Audio Processor. This product is the fourth-generation decoder made by Zoran that handles real-time Dolby Digital 5.1-channel and MPEG-2 digital surrounding algorithms. The floating-point hardware used in this product is optimal for audio applications. With a single DAC, an optical interface for the S/PDIF input, and an oscillator crystal, the chip produces standard decoding functions. It may have 8 channels of output, analog inputs, and long delay memories. The audio decoder handles a maximum bit-rate of 640 Kbits/s and can also use Dolby Pro-logic encoding and decoding. The decoder parses PES streams, decodes PTS formats, and handles SCR. The silicon software allows a magnitude of audio enhancements. The inputs can be serial or parallel streams. Serial SPI, serial Z2C, or 8-bit host interface is supported. Formatted input in the S/PDIF receiver can have a maximum 96-KHz sample rate. The output is formatted as S/PDIF Dolby Digital and MPEG transmitter. The chip has separate PLLs for DSP core and audio I/O. No external memory is needed for basic decoding. The chip operates at 3.3 V. The block diagram of ZR38601 is shown in Fig. 21.27.

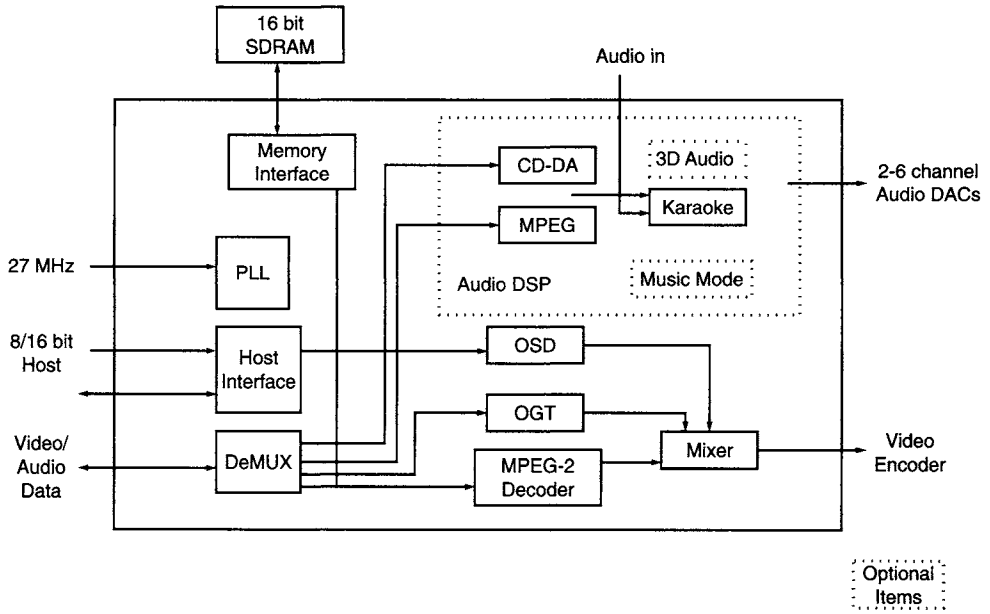

FIGURE 21.26
Single-chip decoder solution by Zoran for SuperVCD applications.

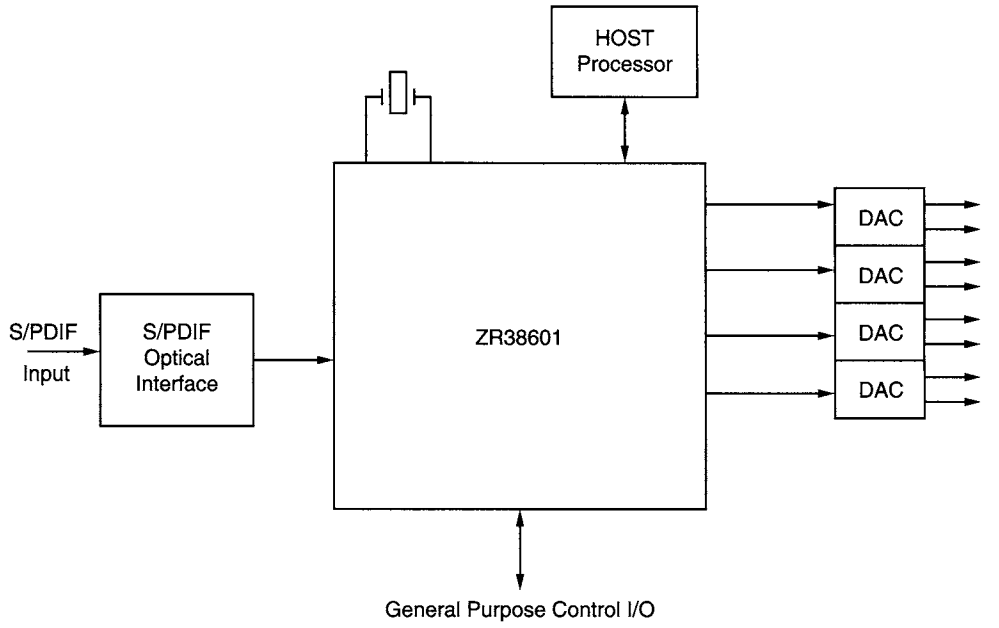

FIGURE 21.27
Single-chip programmable digital audio processor by Zoran.

21.4.2.9 MPEG-2 Video Decoder by Motorola

Motorola developed MCT4000 as a transport and video processor for digital television applications following the Advanced Television System Committee (ATSC) standard. This device provides the MPEG-2 transport parsing function. The decoder accepts MPEG-2 MP@HL PES. The ATSC standards are followed as a guideline on whether to use 480 lines × 720 pixels as interlaced or progressive outputs. This device is useful for set-top applications or television applications. This decoder requires only 4 Mb external SDRAM for the compression algorithm and postfiltering without compromising on good picture quality. Memory access is also optimized to have improved performance. The output end, OSD, is implemented with 256 colors, 64 levels of transparency, and 30-bit resolution. The chip integrates a high-performance video encoder that creates standard definition video output, NTSC, composite video, S-VHS, and RGB formats. The chip also has a bidirectional ITU-R 656 interface with other video processors and for reception of externally digitized analog video data. The chip can operate at 1.8 and 3.3 V for I/O. The chip provides an interface for audio processors. The block diagram is shown in Fig. 21.28.

21.4.2.10 MPEG-2 Digital Audio/Video Decoder by IBM

The MPEGGC22 audio/video decoder is a single chip that can decode MPEG-2 MP@ML video and MPEG-2 stereo layers I and II (CD quality) audio, released by IBM in 1997.

The video decoder handles MPEG-2 MP@ML for a 4:2:0 chroma application. The chip is compatible with European DVB standards. The inputs can be PES layers, Horizontal and vertical filters are used to deliver high-quality video. The chip delivers 11-bit DC precision for enhanced picture quality. Error concealment is performed at all video layers. The chip supports teletext or vertical blanking interval data support. The system provides 2 MB of SDRAM with a 16-bit interface for MPEG-2, NTSC, and PAL up to 15 Mbps.

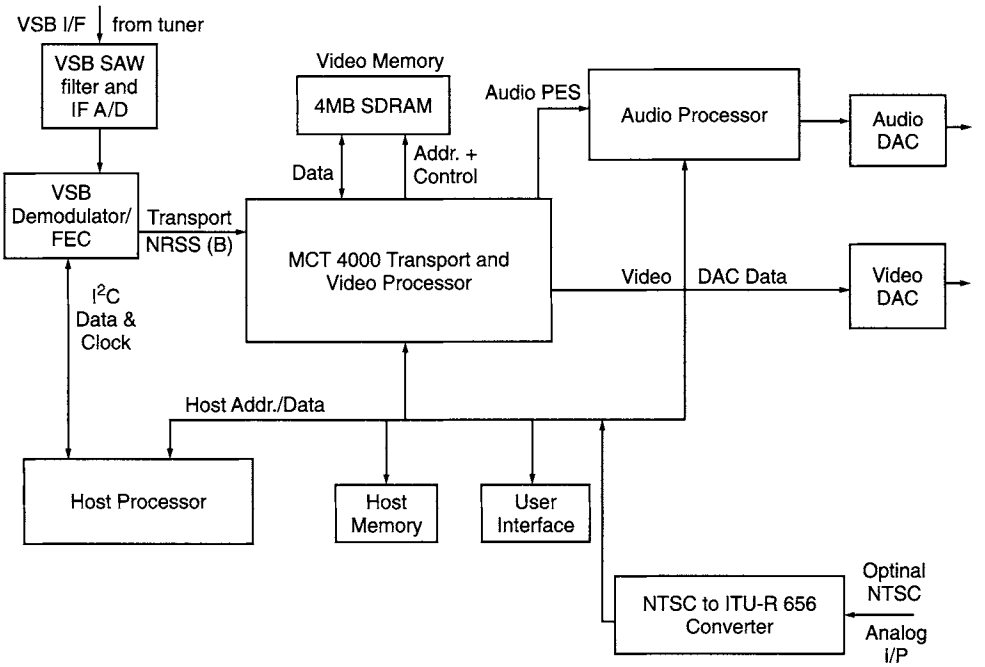

FIGURE 21.28
Transport demultiplexer and video decoder by Motorola.

The audio decoder supports MPEG-2 layer I and layer II stereo. The decoder can accept PCM audio data. Sampling rates are 16, 22.05, 24, 32, 44.1, and 48 KHz. Audio is attenuated at 63 increments per channel. The chip operates at 27 MHz. The OSD is programmable, and video shading and blending can be performed. The input of 8- or 16-bit audio/video compression data and host interface is used. Serial input data can also be provided for the transport and host interface. The chip is built on 0.4-μm CMOS technology with 3.3 V and it consumes 1.4 W of power.

21.4.2.11 MPEG-4 Products by Toshiba

Toshiba developed the LSI TC35274 MPEG-4 video decoder in 2000. The decoder chip performs at 15 frames/s. The MPEG-4 video is decoded with QCIF (Quarter Common Intermediate Format) (176 $\times$ 144 pixels) and the clock is 30 MHz. The chip includes a 4-bit embedded DRAM for low power consumption. The core of the decoder is a 16-bit RISC processor and dedicated hardware accelerator. The firmware program for the RISC to run the chip as an LSI decoder is downloaded into the DRAM before starting. The MPEG standard is ISO MPEG-4 SP@L1 (simple profile at level 1). The output is 4:2:2 Y:Cb:Cr 8-bit digital image data for LCD. The block diagram is shown in Fig. 21.29.

Toshiba also released the MPEG-4 audiovisual codec LSI in 2001. The TC35273 chip supports 3GPP 3G-342M video telephony systems with QCIF format and 15 frames/s. Adaptive multirate speech codec and ITU-T h.223 are executed concurrently at 70 MHz. The unit contains video, audio, and a multiplexer unit. A DRAM of 12 Mbits is shared by all three units. The video codec, audio codec, and multiplexer are each 16-bit RISC processors. All the units are integrated and synchronized by PLLs.

The video codec supporting the MPEG standard is ISO MPEG-4 SP@L1 with QCIF. The input is 4:2:2 Y:Cb:Cr 8-bit digital image data. A CMOS camera or an NTSC decoder can be connected. The output can be 4:2:2 Y:Cb:Cr 8-bit digital image data for LCD, or an NTSC

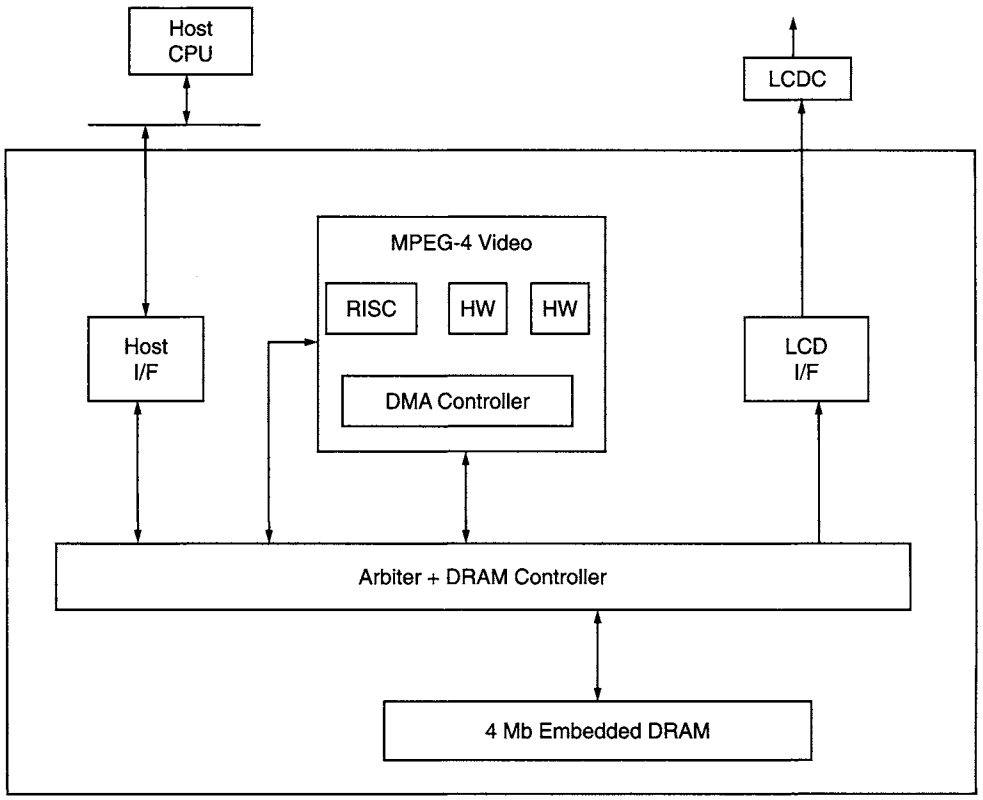

FIGURE 21.29
MPEG-4 video decoder LSI by Toshiba.

encoder is connected. The audio codec uses the AMR speech codec at 8 Kbps with CS-ACELP or the ITU-T G.729 speech codec at 8 Kbps with Cs-ACELP, and other formats. The stereo audio decoder runs at 96 Kbps with a maximum frequency of 44.1 KHz. The decoder can also handle the ISO/IEC 13818-7 AAC LC audio decoder at 144 Kbps with a maximum of 48 KHz of sampling frequency. The input and output can be in PCM format. The multiplexer and demultiplexing units operate at 32 and 384 Kbps, respectively. The unit can handle bitstream input/output via a serial interface. The block diagram is seen in Fig. 21.30.

21.4.2.12 MPEG-4 Decoder by Sigma Designs Inc.

Sigma Design Inc. developed the EM8470 MPEG-1, MPEG-2 MP@ML, and MPEG-4 SP@L1 decoders for set-top DVD. This chip is based on the REALmagic video streaming technology. The formats that the product handles are DVD-video, DVD-audio, SVCD,VCD, CD-DA, and CD-ROM. The audio decoding is performed according to Dolby Digital AC-3 (two channels), MPEG-1 (layers I and II, down-mixed to two channels), and DVD-audio linear PCM. The output can be interlaced or progressive. The block diagram is shown in Fig. 21.31.

21.4.2.13 Other Chip-sets

We describe some other industry products in this section, just highlighting the dominant features of each of them. We mention products released by Sigma Design Inc. Cirrus Logic, Jacob Pineda Inc., MICRONAS, Philips, Conexant, Analog Devices, Amlogic, and Acer Laboratories. In Tables 21.1, 21.2, 21.3, and 21.4, the features are presented in tabular form. We find that most of the applications are DVD compliant and some are compliant with the broadcast standard.

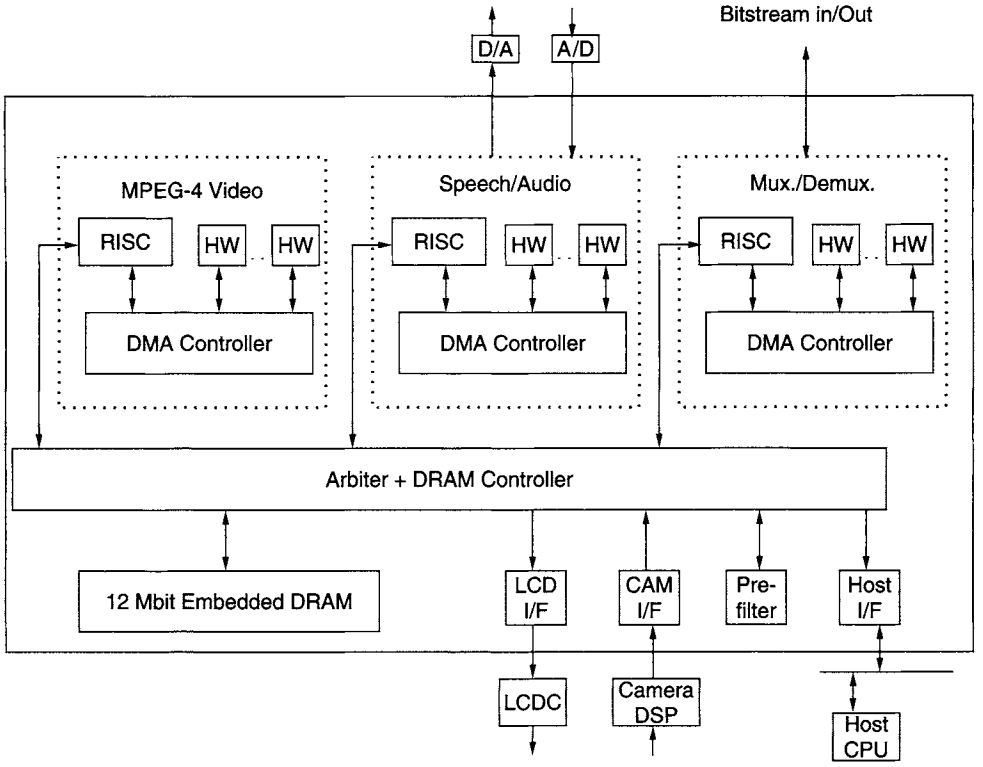

FIGURE 21.30
MPEG-4 audiovisual codec LSI by Toshiba.

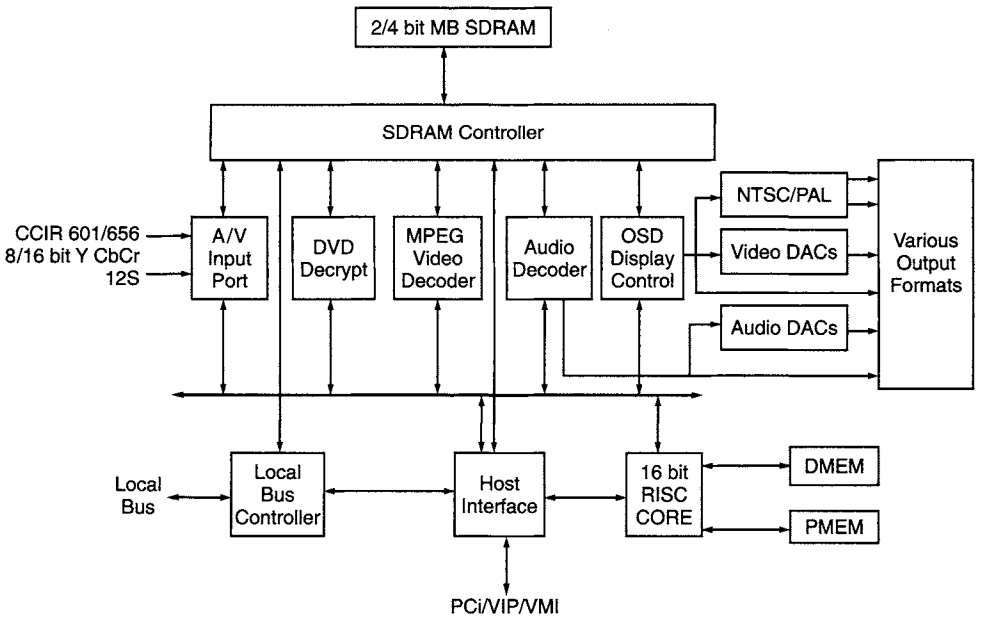

FIGURE 21.31
MPEG-4 video decoder by Sigma Designs Inc.

Table 21.1 Available MPEG Audio or Video Compression Products from Other Vendors

Product	Company	Year	Main Purpose	Video or Audio Standard
Audio products				
CS492301	Cirrus Logic	1999	Audio decoder	Dolby Digital, MPEG-2, PCM
J1	Jacob Pineda Inc.	1997	Audio decoder	5.1-Channel Dolby, AC-3, MPEG layers I and II
MAS3507D	MICRONAS	1998	Audio decoder (low power)	MPEG-1 and -2 layers 2 and 3 (for compression up to 12:1)
MAS3509F	MICRONAS	2000	Audio stereo decoder (low power)	MPEG2-AAC layers 2 and 3
SAA2502	Phillips	1997	Audio source decoder	MPEG-1 layers 1 and 2 and MPEG-2
Video products				
EM8220	Sigma Designs Inc.	1999	MPEG-2/DVD decoder	MP@ML MPEG-2, MPEG-1
VideoFLOW	Array Microsystems	1997	Real-time video encoder	IBBP or I-frame MPEG-1 encoding

Table 21.2 Available MPEG Audio or Video Compression Products from Other Vendors

Product	Frequency (MHz)	Throughput	O/P Format	Host CPU
Audio products				
CS492301	12.288/27	500 Kbps/1.5 Mbps	I2S/left,right justified	24-bit DSP
J1	54	640 Kbps	16-bit PCM, S/PDIF	Commercial
MAS3507D	14.725	<64 Kbps	Multiple formats	RISC CPU
MAS3509F	13–28	<64 Kbps	PCM, S/PDIF	RISC CPU
SAA2502	24.576	<417.96 Kbps	I2S, S/PDIF, 256 or more over sampled analog audio	
Video products				
EM8220	27	64 Kbps	YUV format: CCIR supported resolution for NTSC and PAL	66 MIPS RISC
VideoFLOW	Not available	64 Kbps/3.07 Mbps	MPEG-1	Not available

Table 21.3 Available MPEG Combined Video and Audio Compression Products from Other Vendors

Product	Company	Year	Main Purpose	Video Decoding Standard	Audio Decoding Standard
REALmagic DVR	Sigma Designs Inc.	2000	Digital video recorder	MPEG-1, MPEG-2	MPEG-1 layers 1 and 2
CS98000	Cirrus Logic	2000	Internet DVD	MPEG for DVD, VCD, SVCD	5.1-channel AC-3, MPEG stereo
CX22490/CX22491	Conexant	2000	MPEG video processor	MP@ML, MPEG-2	Dolby Digital, MPEG-1 and MPEG-2
AML3250	Amlogic Inc.	2000	DVD application	MPEG-2, MP@ML	Dolby Ac-3 5.1 channel
M3321	Acer Laboratories	1997	DVD and STB application	MPEG-1, MPEG-2, MP@ML	MPEG-1, MP3, Dolby Digital, DVD LPCM, MPEG-2

Table 21.4 Available MPEG Combined Video Audio Compression Products from Other Vendors

Product	Frequency (MHz)	Throughput	O/P Format	Host CPU
REALmagic DVR	266	500 Kbps to 15 Mbps	NTSC, PAL, MPEG-1, MPEG-2, PCM/WAV	Pentium
CS98000	27	Not available	8-channel PCM output,	Dual 32-bit RISC, 32-bit DSP (A)
CX22490/CX22491	160	Not available	NTSC, PAL with CCIR-supported resolution	32-bit RISC
AM3250	Not available	Not available	NTSC, PAL, CCIR, S/PDIF(A), PCM(A)	RMRISC 32-bit (A)
M3321	27	Not available	I2S(A), CCIR, PAL	Not available

21.4.2.14 Commercial Web Pages for the Vendors

Acer Laboratories at http://www.ali.com.tw/
Amlogic Inc. at http://www.amlogic.com
Analog Devices at http://www.analog.com/
Array Microsystems at http://www.array.com/
ATMEL at http://www.atmel.com/
C-Cube at http://www.c_cube.com/
Cirrus Logic at http://www.cirrus.com/
Conexant Inc. at http://www.conexant.com/
Fujitsu at http://www.fujitsu.org/
IBM Inc. at http://www.ibm.com/
Intel at http://www.intel.com/
Jacob Pineda Inc. at http://www.jacobspineda.com/
LSI Logic at http://www.lsilogic.com/index4.html
Lucent Technologies at http://www.lucent.com/micro
MICRONAS at http://www.micronas.com/
Motorola Inc. at http://www.motorola.com/
NEC Electron Devices at http://www.nec.com
Philips Semiconductors at http://www-eu2.semiconductors.philips.com/cms/
Sigma Designs Inc. at http://www.sigmadesigns.com/
STMicroelectronics at http://us.st.com/
Texas Instruments at http://www.ti.com/
Toshiba America Inc. at http://www.toshiba.com/
Zoran at http://www.zoran.com/

21.5 REFERENCES

1. LSI Logic, 64700 Series Data-Sheet.
2. Kuroda, T., 1997. A 0.9 V, 150 MHz, 10 mW, 4 mm 2,2-D discrete cosine transform core processor with variable-threshold-voltage scheme. In *Digest of IEEE International Solid-State Circuits Conference*, pp. 166–167, February 1997.
3. Artieri, A., E. Macoviak, F. Jutand, and N. Demassieux, 1988. A VLSI one chip for real time two-dimensional discrete cosine transform. In *Proceedings of International Symposium on Circuits and Systems*.
4. Mukherjee, A., N. Ranganathan, J. Fleider, and T. Acharya, 1993. MARVLE: A VLSI chip for data compression using tree-based codes. *IEEE Transactions on VLSI Systems*, Vol. 1, No. 2, pp. 203–214, June 1993.
5. Mukherjee, A., N. Ranganathan, and M. Bassiouni, 1991. Efficient VLSI designs for data transformation of tree-based codes. *IEEE Transactions on Circuits and Systems*, Vol. 38, No. 3, pp. 306–314, March 1991.
6. Mukherjee, A., N. Ranganathan, and M.A. Bassiouni, 1989. Adaptive and pipelined VLSI designs for tree-based codes. In *Proceedings of the 1989 IEEE International Conference on Computer Design*, pp. 369–372.
7. Wu, A., and K.J.R. Liu, 1998. Algorithm-based low-power transform coding architectures: The multirate approach. *IEEE Transactions on VLSI Systems*, Vol. 6, No. 4, pp. 707–717, December 1998.
8. Lewis, A.S., and G. Knowels, VLSI Architecture for 2-D Daubechics Wavelet Transform without Multipliers.
9. Ackland, B., 1993. Video compression and VLSI. In *Proceedings of the IEEE Custom Integrated Circuits Conference*, pp. 11.1.1–11.1.6.

10. Jung, B., and W. Burleson, 1994. A VLSI systolic array architecture for Lempel–Ziv based data compression. In *Proceedings of 1994 IEEE International Symposium on Circuits and Systems*, Vol. 3, pp. 65–68.
11. Jung, B., and W.P. Burleson, 1998. Efficient VLSI for Lempel–Ziv compression in wireless data communication networks. *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, Vol. 6, No. 3, pp. 475–483, September 1998.
12. Tseng, B.D., and W.C. Miller, 1978. On computing discrete cosine transform. *IEEE Transactions on Computers*, Vol. C-27, No. 10, pp. 966–968.
13. Lee, B.G., 1984. A new algorithm to compute the discrete cosine transform. In *Proceedings of International Conference on Acoustics, Speech and Signal Process*, Vol. ASSP, pp. 1243–1245, December 1984.
14. Cafforio, C., and F. Rocca, 1976. Methods for measuring small displacements of television images. *IEEE Transactions on Information Theory*, Vol. IT-22, pp. 573–579, September 1976.
15. Chakrabarti, C., and M. Vishwanath, 1995. Architectures for wavelet transforms: A survey. *Journal of VLSI Signal Processing*, Vol. 10, pp. 225–236.
16. Chakrabarti, C., and M. Vishwanath, 1995. Efficient realization of the discrete and continuous wavelet transforms: From single chip implementations to mappings on SIMD array computers. *IEEE Transactions on Signal Processing*, Vol. 43, No. 3, pp. 759–771, March 1995.
17. Chakrabarti, C., and M. Vishwanath, 1996. Architectures for wavelet transforms: A survey. *Journal of VLSI Signal Processing*, Vol. 14, pp. 171–192.
18. C-Cube Microsystems, 1992. *CL550 Users Manual.*
19. Lin, C.-H., C.-M. Chen, and C.-W. Jen, 1996. Low power design for MPEG-2 video decoder. *IEEE Transactions on Consumer Electronics*, Vol. 42, No. 3, pp. 513–521, August 1996.
20. Chen, C.-T., and L.-G. Cheni, 1997. High-speed VLSI design of the LZ-based data compression. In *Proceedings of 1997 IEEE International Symposium on Circuits and Systems*, Vol. 3, pp. 2064–2067.
21. Huffman, D., 1952. A method for the construction of minimum redundancy codes. *Proceedings of IRE*, Vol. 40, pp. 1098–1101.
22. Johnson, D., V. Akella, and B. Scott, Micropipelined Asynchronous Discrete Cosine Transfer (DCT/IDCT) Processor.
23. Milovanovic, D., and Z. Bojkovic, 1999. MPEG-4 video transmission over the Internet. In *Proceedings of IEEE Conference on Telecommunications in Modern Satellite, Cable and Broadcasting Services*, pp. 309–312, June 1999.
24. Wu, D., Y.T. Hou, W. Zhu, H.-J. Lee, T. Chiang, Y.-Q. Zhang, and H.J. Chao, 2000. On end-to-end architecture for transporting MPEG-4 video over the Internet. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 6, pp. 923–941, September 2000.
25. Royals, D.M., T. Markas, N. Kanopoulos, J.H. Reif, and J.A. Storer, 1993. On the design and implementation of a lossless data compression and decompression chip. *IEEE Journal of Solid-State Circuits*, Vol. 28, No. 9, pp. 948–953, September 1993.
26. Charot, F., G. L. Fol, P. Lemonnier, C. Wagner, and C. Bouville, 1999. Towards hardware building blocks for software-only real-time video processing: The MOVIE approach. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, No. 6, pp. 882–894, September 1999.
27. Langdon, G., 1984. An introduction to arithmetic coding. *IBM Journal of Research and Development*, Vol. 28, No. 2, pp. 135–149, March 1984.
28. Chang, H-C., Y-C. Chang, Y-B. Tsai, C-P. Fan, and L-G. Chen, 2000. MPEG-4 video bitstream structure analysis and its parsing architecture design. In *Proceedings of International Symposium on Circuits and Systems*, pp. 184–187.
29. Park, H., and V.K. Prasanna, 1993. Area efficient VLSI architectures for Huffman coding. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 40, No. 9, pp. 568–575, September 1993.
30. Sato, H., *et al.*, 2000. MPEG-2 4:2:2@HL encoder chip set. In *Proceedings of International Symposium on Circuits and Systems*, Vol. 4, pp. 41–44.
31. Sun, H., W. Kwok, and J.W. Zdepski, 1996. Architectures for MPEG compressed bitstream scaling. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 2, pp. 191–199, April 1996.

32. Lee, H.-Y., L.-S. Lan, M.-H. Sheu, and C.-H. Wu, 1996. A parallel architecture for arithmetic coding and its VLSI implementation. In *Proceedings of the IEEE 39th Midwest Symposium on Circuits and Systems*, Vol. 3, pp. 1309–1312.
33. Yamauchi, H., 1992. Architecture and implementation of a highly parallel single chip video DSP. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 2, pp. 207–220, June 1992.
34. Chuang, H.Y.H., and L. Chen, 1995. VLSI architecture for fast 2-D discrete orthonormal wavelet transform. *Journal of VLSI Signal Processing*, Vol. 10, pp. 225–236.
35. Intel, 1993. *82750DB Display Processor Databook*, Santa Clara, CA, September 1993.
36. Intel, 1993. *82750PB Pixel Processor Databook*, Santa Clara, CA, October 1993.
37. Bae, J., and V.K. Prasanna, Synthesis of VLSI architecture for two-dimensional discrete wavelet transorms. In *Proceedings of IEEE International Conference on Application Specific Array Processors*, pp. 174–181, July 1995.
38. Biemond, J., L. Looijenga, and D. E. Boekee, 1987. A Pel-recursive Wiener-based displacement estimation algorithm for interframe image coding applications. In *Proceedings of SPIE Visual Communication and Image*, Vol. II-845, pp. 424–431.
39. Chen, J., and M.A. Bayoumi, 1995. A scalable systolic array architecture for 2-D disrete wavelet transforms. In *Proceedings of IEEE VLSI Signal Processing Workshop*, pp. 303–312.
40. Kneip, J., B. Schmale, and H. Moller, 1999. Applying and implementing the MPEG-4 multimedia standard. *IEEE Micro*, Vol. 19, No. 6, pp. 64–74, November/December 1999.
41. Limb, J. O., and J. A. Murphy, 1975. Measuring the speed of moving objects from television images. *IEEE Transactions on Communication*, Vol. COM-23, pp. 474–478, April 1975.
42. Venbrux, J., P.-S. Yeh, and M.N. Liu, 1992. A VLSI chip set for high speed lossless data compression. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 2, No. 4, pp. 381–391, December 1992.
43. Ziv, J., and A. Lempel, A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, Vol. IT-23, No. 5, pp. 337–343.
44. Storer, J.A., and J.H. Reif, 1990. A Parallel architecture for high speed data compression. In *Proceedings of the 3rd Symposium on the Frontiers of Massively Parallel Computation*, pp. 238–243.
45. Jou, J.M., and P.-Y. Chen, 1999. A Fast and efficient lossless data-compression method. *IEEE Transactions on Communications*, Vol. 47, No. 9, pp. 1278–1283, September 1999.
46. Lin, K.-J., and C.-W. Wu, 2000. A low-power CAM design for LZ data compression. *IEEE Transactions on Computers*, Vol. 49, No. 10, pp. 1139–1145, October 2000.
47. Kim, A pipelined systolic array architecture for the hierarchical block-matching algorithm. In *Proceedings of International Symposium on Circuits and Systems*, Vol. 3, pp. 221–224.
48. Parhi, K.K., and T. Nishitani, VLSI architectures for discrete wavelet transforms. *IEEE Transactions on VLSI Systems*, Vol. 1, No. 2, pp. 191–202, June 1993.
49. Komarek, 1989. Array architectures for block matching algorithms. *IEEE Transactions on Circuits and Systems*, Vol. 36, October 1989.
50. Rao, K.R., and P. Yip, 1990. *Discrete Cosine Transform*. Academic Press, San Diego, CA.
51. De Vos, L., 1990. VLSI architectures for the hierarchical block matching algorithm for HDTV applications. In *Proceedings of SPIE Visual Communication and Image*, Vol. 1360, pp. 398–409.
52. Liu, L.-Y., J.-F. Wang, R.-J. Wang, and J.-Y. Lee, 1994. CAM-based VLSI architectures for dynamic huffman coding. *IEEE Transactions on Consumer Electronics*, Vol. 40, No. 3, pp. 282–289, August 1994.
53. LSI Logic, 1993. JPEG Chipset Technical Manual, Milpitas, CA.
54. LSI Logic, 1993. L64702 JPEG Coprocessor Technical Manual, Milpitas, CA.
55. Berekovic, M., P. Pirsch, T. Selinger, K.-I. Wels, A. Lafage, C. Heer, and G. Chigo, 2000. Architecture of an image rendering co-processor for MPEG-4 systems. In *Proceedings of ICASSAP*, pp. 15–24. IEEE Press, San Mateo, CA.
56. Gonzalez-Smith, M., and J. Storer, 1985. Parallel algorithms for data compression. *Journal of ACM*, Vol. 32, No. 2, pp. 344–373, April 1985.
57. Kovac, M., and N. Ranganathan, 1995. JAGUAR: A fully pipelined VLSI architecture for JPEG image compression method. *Proceedings of the IEEE*, Vol. 83, No. 5, pp. 247–258, February 1995.
58. Takahashi, M., 2000. A scalable MPEG-4 Video CODEC architecture for IMT-2000 multimedia applications. In *Proceedings of IEEE Symposium on Circuits and Systems*, pp. 188–191, May 2000.

59. Vishwanath, M., R.M. Owens, and M.J. Irwin, VLSI architectures for the discrete wavelet transform.
60. Liu, M.N., 1996. MPEG decoder architecture for embedded applications. *IEEE Transactions on Consumer Electronics*, Vol. 42, No. 4, pp. 1021–1028, November 1996.
61. Sun, M.T., 1991. VLSI architecture and implementation of a high speed entropy decoder. In *Proceedings of International Symposium on Circuits and Systems*, pp. 200–202.
62. Sun, M.T., L. Wu, and M.L. Liou, 1987. A concurrent architecture for VLSI implementation of discrete cosine transform. *IEEE Transactions on Circuits and Systems*, Vol. 34, No. 8, pp. 992–994, August 1987.
63. Demassieux, N., and F. Jutland, 1993. *Orthogonal Transform*, pp. 217–250. Elsevier Science, Amsterdam.
64. Ranganathan, N., and S. Henriques, 1993. High-speed VLSI designs for Lempel–Ziv based data compression. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, Vol. 40, No. 2, pp. 96–106, February 1993.
65. Wu, P-C., and L-G. Chen, 2001. An efficient VLSI architecture for two-dimensional discrete wavelet transform. *IEEE Transactions on CAS for Video Technology*, Vol. 11, No. 4, pp. 536–545, April 2001.
66. Pirsch, P., and H.J. Stolberg, 1997. Architectural approaches for video compression. In *Proceedings of IEEE International Conference on Application-Specific Systems, Architectures and Processors*, pp. 176–185.
67. Pirsch, P., N. Demassieux, and W. Gehrke, 1995. VLSI architectures for video compression— A survey. *Proceedings of the IEEE*, Vol. 83, No. 2, pp. 220–246, February 1995.
68. Ruetz, P., and P. Tong, 1992. A 160-Mpixels/s IDCT processor for HDTV. *IEEE Micro*, Vol. 12, No. 5, pp. 28–32, October 1992.
69. Sriram, P., S. Sudharsanan, and A. Gulati, 2000. MPEG-2 video decompression on a multi-processing VLIW microprocessor. In *Proceedings of IEEE Conference on Consumer Electronics*, June 2000.
70. Jain, P.C., W. Schlenk, and M. Riegel, 1992. VLSI implementation of two-dimensional DCT processor in real time for video codec. *IEEE Transactions on Consumer Electronics*, Vol. 38, No. 2, pp. 537–545, August 1992.
71. Zito-Wolf, R., 1990. A systolic architecture for sliding-window data compression. In *Proceedings of IEEE Workshop on VLSI Signal Processing*, pp. 339–351, November 1990.
72. Rice, R.F., P.-S. Yeh, and W.H. Miller, 1991. Algorithms for a very high speed universal noiseless coding module. In JPL Publication 91-1, JPL Propulsion Laboratory, Pasadena, CA.
73. Zito-Wolf, R.J., 1990. A broadcast/reduce architecture for high-speed data compression. In *Proceedings of the Second IEEE Symposium on Parallel and Distributed Processing*, pp. 174–181.
74. Haralick, R.M., 1976. A storage efficent way to implement the discrete cosine transform. *IEEE Transactions on Computers*, Vol. C-25, 764–765, July 1976.
75. Hwang, S.-A., and C.-W. Wu, 2001. A unified VLSI systolic array design for LZ data compression. *IEEE Transactions on Very Large Scale Integration(VLSI) Systems*, Vol. 9, No. 4, pp. 489–499, August 2001.
76. Purcell, S.C., and D. Galbi, 1992. C-Cube MPEG video processor. In *Proceedings of SPIE Image Processing and Interchange*, Vol. 1659.
77. Golomb, S., 1996. Run-length encodings. *IEEE Transactions on Information Theory*, Vol. IT-12, pp. 399– 401, July 1966.
78. Paek, S.-K., and L.-S Kim, 2000. A real-time wavelet vector quantization algorithm and its VLSI architecture. *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 3, pp. 475–489, April 2000.
79. Chang, S.J., M.H. Lee, and J.Y. Park, 1997. A high speed VLSI architecture of discrete wavelet transform for MPEG-4. *IEEE Transactions on Consumer Electronics*, Vol. 43, No. 3, pp. 623–627, August 1997.
80. Kim, S.P., and D.K. Pan, 1992. Highly modular and concurrent 2-D DCT chip. In *Proceedings of International Symposium on Circuits and Systems*.
81. Winograd, S., 1978. On Computing the Discrete Fourier Transform.
82. Akari, T., 1994. Video DSP architecture for MPEG2 CODEC. In *Proceedings of ICASSP*, Vol. 2, pp. 417–420. IEEE Press, San Mateo, CA.

83. Fautier, T., 1994. VLSI implementation of MPEG decoders. In *Proceedings of International Symposium on Circuits and Systems.*
84. Inoue, T., 1993. A 300-MHz BiCMOS video signal processor. *IEEE Journal of Solid-State Circuits*, Vol. 28, December 1993.
85. Huang, T. S., 1983. *The Differential Method for Image Scene Analysis.* Springer-Verlag, Berlin/ New York.
86. Totzek, U., F. Matthiesen, S. Wohllenben, and T.G. Noll, 1990. CMOS VLSI implementation of the 2D-DCT with linear processor arrays. In *Proceedings of International Conference on Acoustics, Speech and Signal Process*, Vol. 3.
87. Bhaskaran, V., and K. Konstantinides, 1997. *Image and Video Compression Standards: Algorithms and Architectures*, 2nd ed. Kluwer Academic, Dordrecht/Norwell, MA.
88. Badawy, W., and M. Bayoumi, 2000. VLSI architecture for hybrid object-based video motion estimation. In *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering*, May 2000.
89. Gehrke, W., R. Hoffer, and P. Pirsch, 1994. A hierarchical multi-processor architecture based on heterogeneous processors for video coding applications. In *Proceedings of ICASSP*, Vol. 2. IEEE Press, San Mateo, CA.
90. Lai, Y.-K., and K.-C. Chen, 2000. A novel VLSI architecture for Lempel–Ziv based data compression. In *Proceedings on IEEE International Symposium on Circuits and Systems*, Vol. 5, pp. 617–620.
91. Tung, Y.-S., J.-L. Wu, and C.-C. Ho, 2000. Architecture design of an MPEG-4 system. In *Proceedings of International Conference on Consumer Electronics*, pp. 122–123.
92. Arai, Y., T. Agui, and M. Nakajima, 1998. A fast DCT-SQ scheme for images. *Transactions on IEICE*, Vol. E71, No. 11, pp. 1095–1097.