A Video-Codec Chip Set for Multimedia Applications

Bryan D. Ackland
Reza Aghevli
Ismail Eldumiati
Arnold C. Englander
Eugene Scuteri, Jr.

The multimedia, video-codec chip set described in this paper represents the first generation in a family of devices that will make possible reliable, cost-effective storage and transmission of multimedia data, including video. AT&T's multimedia video-codec complies with international video standards. It supports still-image and audio compression over a wide range of quality and bit rates; multiplexing of audio, video, and data; and protocol control for a variety of telecommunications options. The codec consists of three chips—a video encoder, a video decoder, and a systems controller—plus one or two digital-signal processors and a modest amount of industry-standard, dynamic random-access memory. In this paper, we examine the potential market for such a device and the performance and capability requirements. We also describe the overall multimedia-codec architecture and review the video-signal-processing operations for standards-compliant video compression. Finally, we describe the architecture of the encoder, decoder, and systems controller.

Introduction

Until recently, telecommunications and computing were viewed as separate disciplines. Telecommunications was analog and done in real time, whereas computing was digital and done at a rate determined by the processing speed of the computer. Today, such technologies as speech processing, electronic mail, and facsimile have blurred these lines. In the next few years, computing and telecommunications will become almost indistinguishable in the race to support a broad range of new multimedia (i.e., voice, video, and data) applications. These applications are made possible by emerging digital-processing technologies, which include:

- Compressed audio (both high-fidelity audio and speech)
- High-resolution still images
- Compressed video
- Speech and handwriting recognition
- Speech synthesis.

The initial driving applications will be for collaboration at a distance, including videoconferencing, educating, consulting, negotiating, and selling.

Of these technologies, video is particularly exciting in terms of its potential applications. But video is also the most demanding in terms of processing power and the sheer quantity of data to be processed. Uncompressed digital video requires somewhere between 50 and 200 Mb/s (megabits per second) to support the real-time transmission of standard television-quality images. This makes impractical the widespread use of uncompressed digital video in telecommunications applications.

Fortunately, there is considerable redundancy in video data, both in terms of information theory and human perception. This redundancy allows us to compress digital video sequences into lower transmission rates. For some time, researchers have been aware of a variety of techniques that can be used to compress video-data sequences anywhere from 2:1 to 1000:1, depending on the quality required by the application.[1] Until recently, however, it was not practical to incorporate these techniques into low-cost, video-based applications.

The opportunity to develop cost-

Panel 1. Abbreviations, Acronyms, and Terms

ANSI — American National Standards Institute

CCITT — International Telegraph and Telephone Consultative Committee (Geneva, Switzerland)

CD-ROM — a compact disk used as a read-only memory to store digitized images or data that can be accessed interactively

CHI — concentration-highway interface, a four-wire, time-division multiplexed link

CIF — common intermediate format; the CCITT's image-format standard

CMOS — complementary, metal-oxide semiconductor

codec — coder–decoder

DCT — discrete cosine transform

DRAM — dynamic random-access memory

DSP — digital-signal processor

DSP3210 — a multimedia, digital-signal processor; comes with a library of software modules that run under the AT&T VCOS™ operating system.

FDDI-II — a variant of FDDI, the fiber-distributed data interface, and is an ANSI standard for a metropolitan-area network that supports isochronous traffic

FIFO— a first-in, first-out data queue

G.711 — CCITT standard for 3-kHz audio. The 64-kb/s algorithm uses pulse-coded modulation to digitally encode analog audio.

G.722 — CCITT standard for 7-kHz audio. The 64-kb/s algorithm is used for ISDN teleconferencing and loudspeaker telephony.

G.728 — CCITT standard for 3-kHz audio. The 16-kb/s algorithm for digitally encoding analog audio.

H.221 — CCITT standard for videoconferencing applications that specifies the frame structure

H.230 — CCITT standard for videoconferencing applications that specifies frame-synchronization control

H.242 — CCITT standard for videoconferencing applications that specifies communication between audio-visual terminals

H.261 — CCITT standard for videoconferencing applications that specifies the compressed-data syntax for encoded video

HBI — host-bus interface

HDTV — high-definition television

I/O — input/output

ISDN — integrated-services digital network, a CCITT-defined digital network

ISO — International Organization for Standardization (Geneva, Switzerland)

JPEG — Joint Photographic Experts Group, which sets ISO standards for storage and retrieval of high-quality still images

macroblock — consists of four 8-by-8 luminance blocks and two 8-by-8 chrominance blocks

MMAFC — memory management and flow control

MPEG — Motion Picture Experts Group, which sets ISO standards for video, particularly CD-ROM

NTSC — National Television Systems Committee, which sets standards for television and video playback and recording in the United States

P×64 — a set of standards established by the CCITT for video telephony and conferencing

PACER — an embedded, RISC processor on the systems-controller chip; responsible for data movement, bit manipulation, and error-code generation and detection, and for assembly and disassembly of the H.221 frame

PAL — Phase Alternating Line; the European equivalent of the NTSC standard

PBI — pixel-bus interface

RAM — random-access memory

reference macroblock — the macroblock for the current frame

RGB — red, green, and blue; a common means of interfacing to a frame buffer

RISC — reduced-instruction-set computer

ROM — read-only memory

SBI — serial-bus interface

SIMD — single instruction, multiple data

SRAM — static random-access memory

UART — universal, asynchronous receiver–transmitter

VBI — video-bus interface

VLE — variable-length encoder

VLSI — very-large-scale integration

YC_rC_b — color-separated video component used in the CIF video-format standard; the Y stands for luminance or intensity and C_r = Y − red and C_b = Y − blue are color differences

effective applications that are based on compressed video is the result of progress in three main areas:

- *Standards.* The recent development of CCITT P×64 as a standard for videoconferencing, ISO JPEG as a standard for high-quality, still-image storage and retrieval, and ISO MPEG as a standard for interactive video playback provides a platform for interoperability between vendors. This platform, in turn, leads to widespread markets that benefit from economies of scale. (CCITT is the International Telegraph and Telephone Consultative Committee, and ISO is the International Organization for Standardization. JPEG stands for Joint Photographic

Table I. Applications of Compressed Video

Market	Bandwidth	Standard	Size (pixels by lines)	Frame rate (frames per second)
Analog videophone	5 to 10 kb/s	none†	170 × 128	2 to 5
Basic-rate video telephony	56 to 128 kb/s	P×64	176 × 144 352 × 288	5 to 10
Business conferencing	≥ 384 kb/s	P×64	352 × 288	15 to 30
Interactive multimedia	1 to 2 Mb/s	MPEG	up to 352 × 288	15 to 30
Digital NTSC	3 to 10 Mb/s	MPEG II	720 × 480	30
High-definition television	> 15 Mb/s	FCC	1200 × 800	60

† Several standards initiatives are in progress.

Experts Group and MPEG stands for Motion Picture Experts Group, although both terms are also used to refer to a standard established by the group.)

- *Networking.* The availability of isochronous digital-communications networks, such as FDDI-II and ISDN, permits video data to be transported to a broad user community. (FDDI-II is a variant of FDDI, the fiber-distributed data interface, and is an ANSI standard for a metropolitan-area network that supports isochronous traffic. ISDN is Integrated-Services Digital Network, a CCITT-defined digital network. ANSI is the American National Standards Institute.)
- *Very-large-scale integration (VLSI).* High-performance VLSI technologies, such as submicrometer CMOS, have advanced to the point that the processing power and memory required to compress and decompress video in real time can be incorporated in a few silicon chips. (CMOS is complementary, metal-oxide semiconductor.)

In this paper, we describe the architecture and capabilities of three VLSI chips: a video encoder, the AT&T AVP-4310E; a video decoder, the AT&T AVP-4220D; and a systems controller, the AT&T AVP-4120C. These three chips, together with one or two DSPs (digital-signal processors) and a modest amount of industry-standard DRAM (dynamic random-access memory), provide a complete multimedia codec (coder–decoder) that supports:

- Standards-compliant video
- Still-image and audio compression over a wide range of quality and bit rates
- Multiplexing of audio, video, and data
- Protocol control for a wide range of telecommunications options.

To begin, we examine the potential market segments and derive from them a list of performance and capability requirements. Next, we describe the overall multimedia-codec architecture and the way it might be used in a videoconferencing application. We then review the video-signal-processing operations that are required to provide standards-compliant video compression. Finally, we show how these operations are implemented in a chip set by describing the architecture of each of the three chips. (Panel 1 defines acronyms and terms used in this paper.)

Codec Requirements

Compressed video will make possible a broad range of new business and consumer-oriented products and services that have a wide range of requirements in terms of video quality, bit rate, delay, cost, and so on. A simple way to segment these applications is to use the bit rate required to maintain real-time video. See Table I.

At the low end of the spectrum is video telephony based on the use of regular, analog telephone lines.[2] The compressed-video bit rate is limited to somewhere around 10 kb/s (kilobits per second), in turn, limiting the picture size, quality, and frame rate. At the other end of the spectrum is high-definition television (HDTV), which requires a data rate of more than 15 Mb/s.

In designing the video-codec chip set, we have focused on a broad, intermediate range of ISDN and multimedia applications that span data rates from 56 kb/s to 4 Mb/s. These rates support a wide range of video quality, all the way from "head and shoulders" video telephony up to entertainment-quality digital television.

After discussing the requirements of the various applications with potential customers, we identified several key features and parameters that, in turn, specified the capabilities of the multimedia codec:

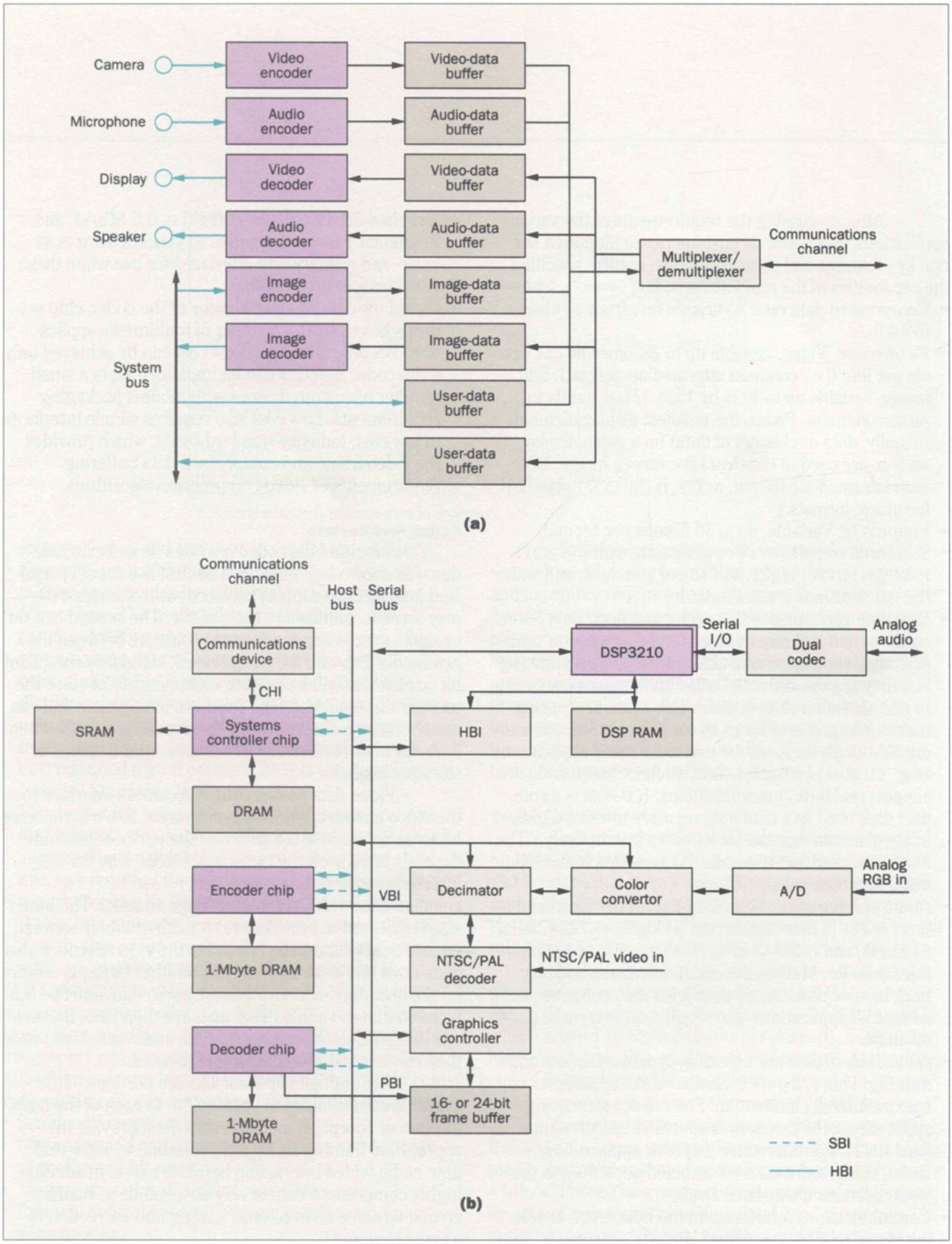
- Compressed data rate: A variable rate from 40 kb/s to 4 Mb/s.
- Picture size: Video, variable up to 288 lines by 352 pixels per line (i.e., common intermediate format). Still image, variable up to 1024 by 1024. [*Pixel* stands for picture element. Pixels, the smallest display elements (usually, dots or clusters of dots) on a video display screen, are used to construct the screen image. Common intermediate format, or CIF, is the CCITT standard for image formats.]
- Frame rate: Variable, up to 30 frames per second.
- Standards compliance: Complies fully with the CCITT P×64 (i.e., H.261, H.221, and so on) standard, and with the ISO MPEG and JPEG standards.
- Real-time performance: Encoding and decoding. Some compression techniques (e.g., Intel Corporation's proprietary DVI compression algorithm[3]) are asymmetric; i.e., they require orders of magnitude more processing to encode rather than decode. This extra processing makes it impractical for us to consider real-time encoding. Although they may be useful for some applications (e.g., CD-ROM playback, HDTV), such techniques do not support real-time communications. (CD-ROM is a compact disk used as a read-only memory to store digitized images or data that can be accessed interactively.) The P×64, JPEG, and MPEG standards can be implemented using symmetrical algorithms.
- Audio compression: According to CCITT recommendations G.711 [3 kHz (kilohertz), 64 kb/s], G.722 (7 kHz, 64 kb/s), and G.728 (3 kHz, 16 kb/s); also MPEG high-fidelity audio. Multimedia communications and playback involve much more than just video compression. Almost all applications also require some form of digital audio.
- Audio/video/data multiplexing, synchronization, and framing: This category is another nontrivial aspect of true multimedia interaction. For videoconferencing applications, the category is specified by CCITT standard H.221. For interactive playback applications, audio, video, and data are combined according to the MPEG Systems Committee Draft.
- Communications interfaces: In the near term, simple interfaces to AT&T Accunet® digital services—Switched 56 (i.e., 56 kb/s), T1 (i.e., 1.5 Mb/s), and Fractional T1—are essential. Also needed are ISDN basic- and primary-rate interfaces for use when these ISDN services are available.
- Cost: Low cost. When the price of the codec chip set drops below $500, a number of multimedia applications become attractive. Low cost can be achieved only if the codec function can be implemented in a small number of custom devices with modest packaging requirements. Low cost also requires simple interfaces to low-cost, industry-standard DRAM, which provides the video-frame and compressed-data buffering requirements of video-compression algorithms.

Codec Architecture

A multimedia codec provides three basic functions, as shown in Figure 1a. The first is a set of specialized interfaces to support connections to cameras, display screens, handsets, CD-ROMs, etc. The second is a set of signal-processing functions to translate between the raw media data and the compressed, standards-compliant bit stream that will be used to communicate or store the multimedia sequence. The third is a multiplexer that can mix, frame, synchronize, and correct the sequence when it is sent to or received from a real communications or storage channel.

Video data passes from the camera interface to the video encoder, where it is converted into a compressed-bit stream. Because the data rate the encoder generates depends heavily on the picture's content, a buffer is inserted to match the varying data rate to that of a constant-bandwidth communications channel. The buffer's size is selected to provide a reasonable tradeoff between picture quality and delay. Because the video decoder also consumes bits at an uneven rate, another buffer is required on the decoding side. In addition, audio data must be buffered so that the audio delay matches the video. Because still images do not require real-time audio synchronization, they can be treated as data transmissions.

The multiplexer must allocate portions of the available communications bandwidth to each of the transmission or reception media. It also must provide the appropriate framing and synchronization to allow real-time audio/video interaction between users. In addition, highly compressed data is very susceptible to random errors, so some form of error coding and correction is usually required.


Camera
Video encoder
Video-data buffer
Microphone
Audio encoder
Audio-data buffer
Display
Video decoder
Video-data buffer
Speaker
Audio decoder
Audio-data buffer
Image encoder
Image-data buffer
Image decoder
Image-data buffer
System bus
User-data buffer
User-data buffer
Multiplexer/ demultiplexer
Communications channel
(a)
Communications channel
Host bus
Serial bus
Communications device
CHI
SRAM
Systems controller chip
DRAM
HBI
DSP3210
Serial I/O
Dual codec
Analog audio
DSP RAM
Encoder chip
VBI
Decimator
Color convertor
A/D
Analog RGB in
1-Mbyte DRAM
NTSC/PAL decoder
NTSC/PAL video in
Graphics controller
Decoder chip
PBI
16- or 24-bit frame buffer
1-Mbyte DRAM
SBI
HBI
(b)

As Figure 1b shows, three custom chips and one or two DSPs provide these functions in our multimedia codec. The video encoder and decoder do real-time coding and decoding of full-motion video, according to the P×64 and MPEG standards. The AT&T DSP3210 provides real-time audio coding and decoding, still-picture coding and decoding (according to the JPEG standard), and echo suppression and cancellation for speakerphone applications. Certain combinations of audio and still-picture encoding require two DSP3210s to provide adequate processing power. All the DSP multimedia software is implemented under AT&T's VCOS™ computer software for real-time operation of DSPs[4]—a multitasking, DSP operating system that allows for easy distribution of multiple tasks across one or more DSPs.

As Figure 1b shows, all the devices are connected via two busses:

- The system or *host bus* is used to set up and monitor the operation of the codec chips. It can also be used to move compressed or uncompressed video data to and from the codec chips.
- The *serial bus* is really a 10-channel serial pathway between the encoder, decoder, and DSP chips and the systems-controller chip. This bus is designed to carry all the compressed-data traffic between various devices. Thus, a multimedia conference call can be conducted without any interaction with the host (or the host bus). The host (i.e., the main controller in the applications platform) is only required for call setup and error monitoring.

In addition to its host-bus and serial-bus interfaces (i.e., the HBI and SBI), the encoder chip also has a video-bus interface (i.e., the VBI) through which it receives uncompressed, digital, video data. The encoder requires up to 1 Mbyte (megabyte) of DRAM to provide the frame buffering required by the encoder algorithms.

Figure 1. AT&T's multimedia codec provides (a) three basic functions: specialized interfaces to various media; signal processing to translate raw media data into the compressed-bit stream; and multiplexing for the sequence that is sent to or received from a communications or storage channel. (b) The multimedia codec consists of three custom VLSI chips that have an assortment of bus interfaces; external and internal industry-standard DRAM and buffers support the three custom chips and a DSP. For adequate computational power, some applications require two DSPs.

The decoder chip also requires up to 1 Mbyte of DRAM, and is designed to interface directly to an RGB (i.e., red, green, and blue) frame buffer.

The systems-controller chip uses the concentration-highway interface (CHI)—a four-wire, time-division multiplexed link—to interface directly with a range of existing AT&T communications devices. These devices provide direct connection to ISDN and its digital precursors (e.g., T1, Switched 56). The systems-controller chip requires 1 Mbyte of DRAM, which it uses to configure the various data buffers in Figure 1a. The chip also requires 64 kwords by 24 bits of external SRAM (static random-access memory), to which it downloads the microcode for the P×64 or MPEG protocols. The use of SRAM to store the protocols enables the chip set to keep pace as the protocols evolve. Stored protocols can be updated easily, so the chip set remains compatible with the protocols used in multimedia devices from other manufacturers.

The DSP3210 is a multimedia, digital-signal processor that comes with a library of software modules that run under the VCOS operating system. These modules provide:

- μ-law, 64-kb/s audio, according to the G.711 standard
- 7-kHz wideband audio at 64 kb/s, according to the G.722 standard
- 16-kb/s, low-delay, audio coding, according to the G.728 standard
- MPEG audio for high-fidelity multimedia applications
- JPEG still-image coding and decoding
- Acoustic echo suppression and cancellation.

The DSP can operate from its own dedicated program memory, or it can share the memory address space of the host. Connection to a regular audio codec is done via the DSP's serial I/O (input/output) port.

In a typical video-teleconferencing application, video would enter the codec as analog NTSC composite video directly from a camera. (NTSC is the National Television Systems Committee, although the term is also used for the video standard the committee established.) The NTSC decoder converts this to digital format (i.e., 480 lines by 720 pixels per line, 30 frames per second, interlaced). A decimator circuit selects one field of this digital video and then does subsampling (i.e., reduces the field's resolution) and filtering. To be compatible with the CIF standard (i.e., the digital input format required by the encoder), the luminance information is

subsampled and filtered to 352 pixels horizontally, and interpolated and filtered to 288 lines vertically. The chrominance information is subsampled in the horizontal and vertical directions to yield a resolution of 144 lines by 176 pixels per line. This progressively scanned CIF data feeds directly into the encoder at 30 frames per second. (*Luminance* represents information about the black-and-white aspects of the image, while *chrominance* provides the color information.)

The encoder chip converts this video into a compressed-bit stream, using several parameters that are downloaded from the host. These parameters determine the target bit rate, picture quality, coded-frame rate, coding delay, and so on. The compressed-bit stream is then passed along the serial bus to the systems-controller chip. The systems controller uses its external DRAM to provide the buffering needed for the compressed video data to meet the constant-bit requirements of the communications channel.

Analog audio is digitized and passed to the DSP, which converts it into a compressed-data stream. This compressed-audio stream is also passed along the serial bus to the systems-controller chip. There, it is buffered and delayed to match the video-encoding delay. The systems controller takes the compressed audio and video, along with any host data that may have been presented on the host bus, and frames these into one or more H.221 streams. These streams are sent directly to the communications channel via the concentration-highway interface.

At the receiving end, data enters the systems-controller chip via the concentration-highway interface. Audio, video, and host data are demultiplexed, error corrected, buffered, and resynchronized. Compressed-audio data is passed along the serial bus to the DSP, which decodes it into real-time, analog audio. Compressed-video data is passed to the decoder chip.

The decoder converts the compressed data into 4:2:2 YC_rC_b or 24-bit RGB CIF frames at the coded frame rate. (The Y in YC_rC_b stands for luminance; the C_r stands for Y – red; and the C_b stands for Y – blue. The notation 4:2:2 for coding means that each 2-by-2 group of pixels is represented by four Y values, two C_r values, and two C_b values.) These frames are read from the pixel-bus interface (PBI) directly into the frame buffer. The frame buffer's refresh logic generates the necessary 60-Hz, display-refresh rate.

Video Compression

Both MPEG and P×64 use a variety of techniques to isolate and remove redundancies in the image sequence and then code the remaining data efficiently, based on the information content. Before we examine the detailed architecture of the codec chip set, it is useful to review these techniques:

- *Chrominance subsampling.* Because the human eye is less sensitive to high-frequency color information, it is possible to code the chrominance channels at reduced resolution with little perceived loss in quality. Hence, the input to the encoder is assumed to be in a luminance–chrominance format, known as YC_rC_b. In addition, the chrominance data is subsampled by a factor of two in each direction. This subsampling is known as 4:1:1 coding where each 2-by-2 group of pixels is represented by four Y values, one C_r value, and one C_b value.
- *Transform coding.* If we transform an image from the pixel domain into the frequency domain, then we usually find large amounts of low-spatial-frequency information and relatively small amounts of high-frequency information. This allows us to code the image more efficiently by allocating more transmitted bits to the low-frequency portions of the image. Each frame is divided into 8-by-8 pixel blocks. To each of these blocks, we apply a two-dimensional, discrete cosine transform (DCT). This produces an 8-by-8 block of two-dimensional, spatial-frequency components. Typically, many of these components will be zero and need not be sent. Others will be near zero and can be coded efficiently, using standard entropy-coding techniques.
- *Frame differencing.* A normal video sequence has little variation from one frame to the next. If, instead of coding each frame, we code only the difference between a frame and the previous frame, then the amount of information needed to describe the new frame will be dramatically reduced.
- *Motion compensation.* Much of the difference that does occur between successive frames can be characterized as a simple translational motion, caused either by the moving objects in the scene or by a pan of the field of view. Rather than form a simple difference between a macroblock in the current frame and the same macroblock in the previous frame, we can search the area near the chosen macroblock in the previous frame to find an offset block that more

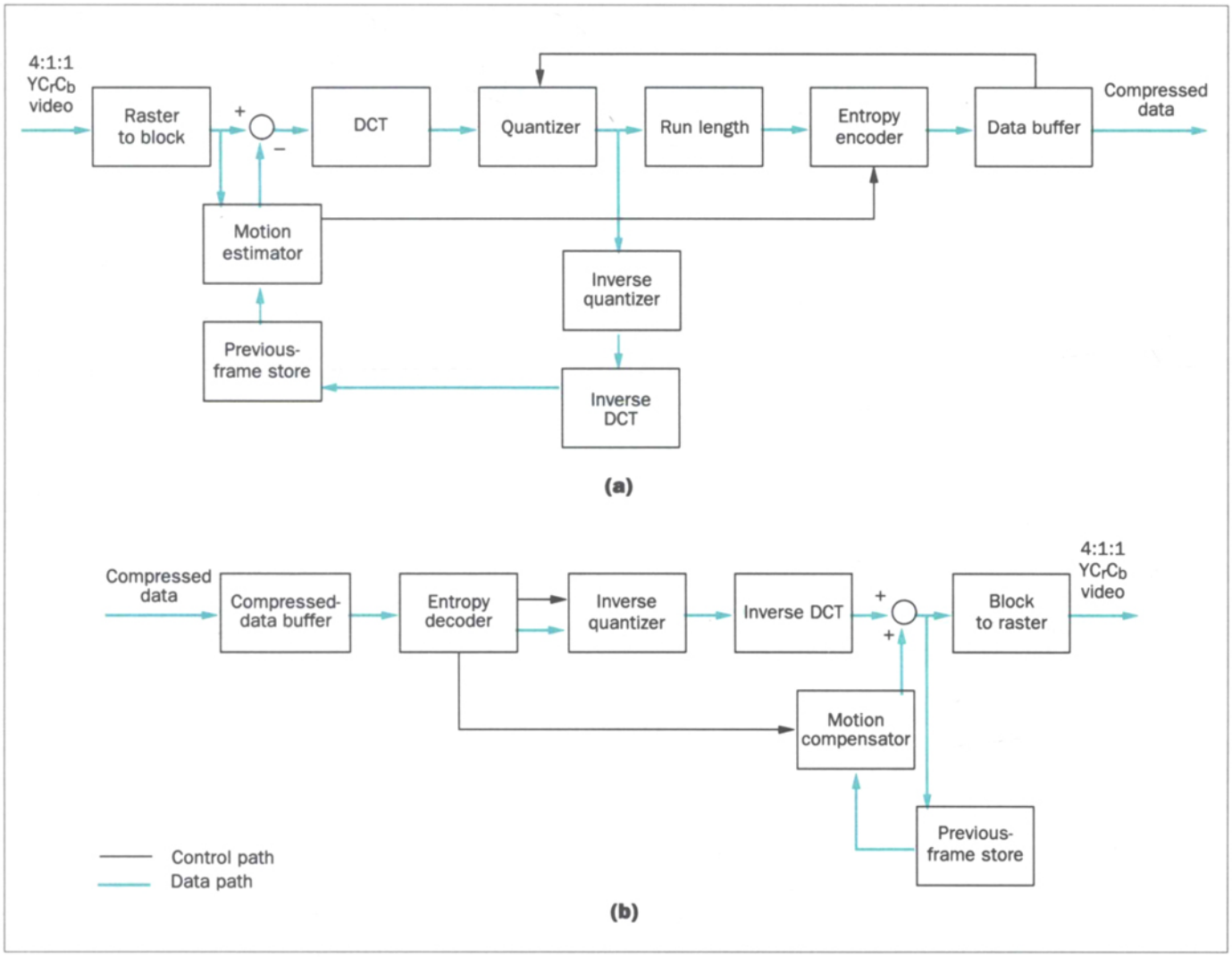


Figure 2. The (a) video encoding and (b) decoding algorithms. The encoder transmits differential information that is referenced to the previous frame; hence, the encoder and decoder must have identical copies of that frame. Because the coding or decoding operation is lossy, the only way to minimize error is to include a portion of the decoder in the encoding feedback loop.

closely matches the macroblock in the current frame. (A *macroblock* is a group of 16-by-16 pixels.) Once we have identified the best match, we code the difference between the reference macroblock in the current frame and the best match in the previous frame, along with the vector that describes the offset of the best match.

- *Interpolation.* In MPEG, picture-coding efficiency is further improved by predicting the value of an intermediate frame as the weighted average of a previous frame and a subsequent frame. Motion compensation must be applied to avoid artifacts caused by objects that have moved from one reference frame to another.
- *Entropy coding.* After all these steps have been performed, the remaining data will be unevenly distributed across the range of possible numerical values. Small values of DCT data will occur more frequently than large values. The same is true for the values of the motion vectors. Entropy coders take advantage of

this uneven distribution by using fewer bits to represent values that occur frequently. *Hierarchical Huffman coding* is the particular technique used in both P×64 and MPEG.

For a more comprehensive description of video-compression algorithms, refer to the paper by Aravind et al. in this issue.[5]

Encoder Techniques. Figure 2a illustrates how the various techniques are combined to produce a P×64 encoder. Assume that the video data is presented to the encoder in 4:1:1 YC_rC_b format.

The first step is to buffer the data to produce 16-by-16 macroblocks, each consisting of four 8-by-8 luminance blocks and two 8-by-8 chrominance blocks.

For each reference macroblock (i.e., the macroblock for the current frame), the motion estimator searches the previous-frame buffer for an offset 16-by-16 block of pixels that most closely matches the reference. This best-match block is retrieved from the previous-frame buffer and subtracted from the reference to create the difference block. (The *previous-frame buffer* holds the previous frame.) Next, the difference block is converted to the frequency domain, using the DCT. The frequency components generated by the DCT are now quantized according to the number of bits available for coding. A run-length encoder removes zero-valued coefficients. (*Run-length encoding* is a technique for compressing data sequences that have large numbers of zeros.)

Finally, the entropy encoder converts DCT values and motion vectors to variable-length codes, and then outputs the coded sequence into the compressed-data buffer. This buffer is used to maintain a constant bit rate for the output. The buffer fullness is used to regulate the degree of quantization that is applied to the DCT values and, hence, the degree of compression.

Decoder Techniques. Figure 2b shows the decoding operation. The compressed-data stream is first applied to the entropy decoder, which detects and converts the variable-length codes into DCT values, quantization levels, and motion vectors. The DCT values are passed through an inverse quantizer that restores the correct level (but not the truncated bits—these are lost in the process). The inverse-DCT processor converts the quantized values back into real pixel-difference values.

The motion compensator takes the motion vector and uses it to calculate the address of the best-match, 16-by-16 pixel block in the previous frame. Then, this block is added to the output of the DCT processor to produce the new, decoded macroblock. Next, the decoded macroblock is converted to raster form for output to a display screen. The macroblock is also written into the previous-frame buffer to be used as a reference for subsequent frames.

Note that a significant portion of the decoder is also included in the encoder architecture. The encoder transmits differential information that is referenced to the previous frame. Therefore, it is critical that the encoder and decoder have identical copies of the previous frame. Because the coding or decoding operation is lossy, the only way to ensure this is to include a portion of the decoder as part of the encoding-feedback loop.

Decoder

The video-decoder chip (AVP-4220D)[6] is designed to decode full-motion video, according to the P×64 and MPEG standards. The decoder accepts compressed-video data at rates up to 4 Mb/s from either the serial bus or the parallel host bus via the host/serial interface. The decoder supports variable frame rates up to 30 frames per second, and variable spatial resolutions up to 288 lines by 360 pixels per line. In MPEG video mode, the decoder can accommodate an arbitrary number of bidirectional interpolated frames. In MPEG still-picture mode, it supports variable resolutions up to 1024 lines by 1024 pixels per line.

The decoder outputs the decoded video (in raster-scanned order) on either its host bus or the dedicated pixel bus. The output format is either 24-bit RGB or 4:2:2 YC_rC_b. (The choice of output bus and format is programmed through the host bus.) The decoder requires 1 Mbyte of 70-ns (nanosecond) DRAM to provide the previous-frame buffers for predictive decoding in P×64 and MPEG and for interpolative decoding in MPEG.

Figure 3 shows a photograph and a block diagram of the decoder. Compressed data is passed through the host-bus or serial-bus interface to the compressed-data FIFO—a small (i.e., 512 bits by 8 bits), first-in, first-out buffer that is designed to hold only bursty input. (The systems-controller chip provides the larger, compressed-data FIFO needed to smooth frame-to-frame variations.) The FIFO includes a user-programmable flag that is set during initialization and can be used to generate a host interrupt at any predetermined FIFO level.

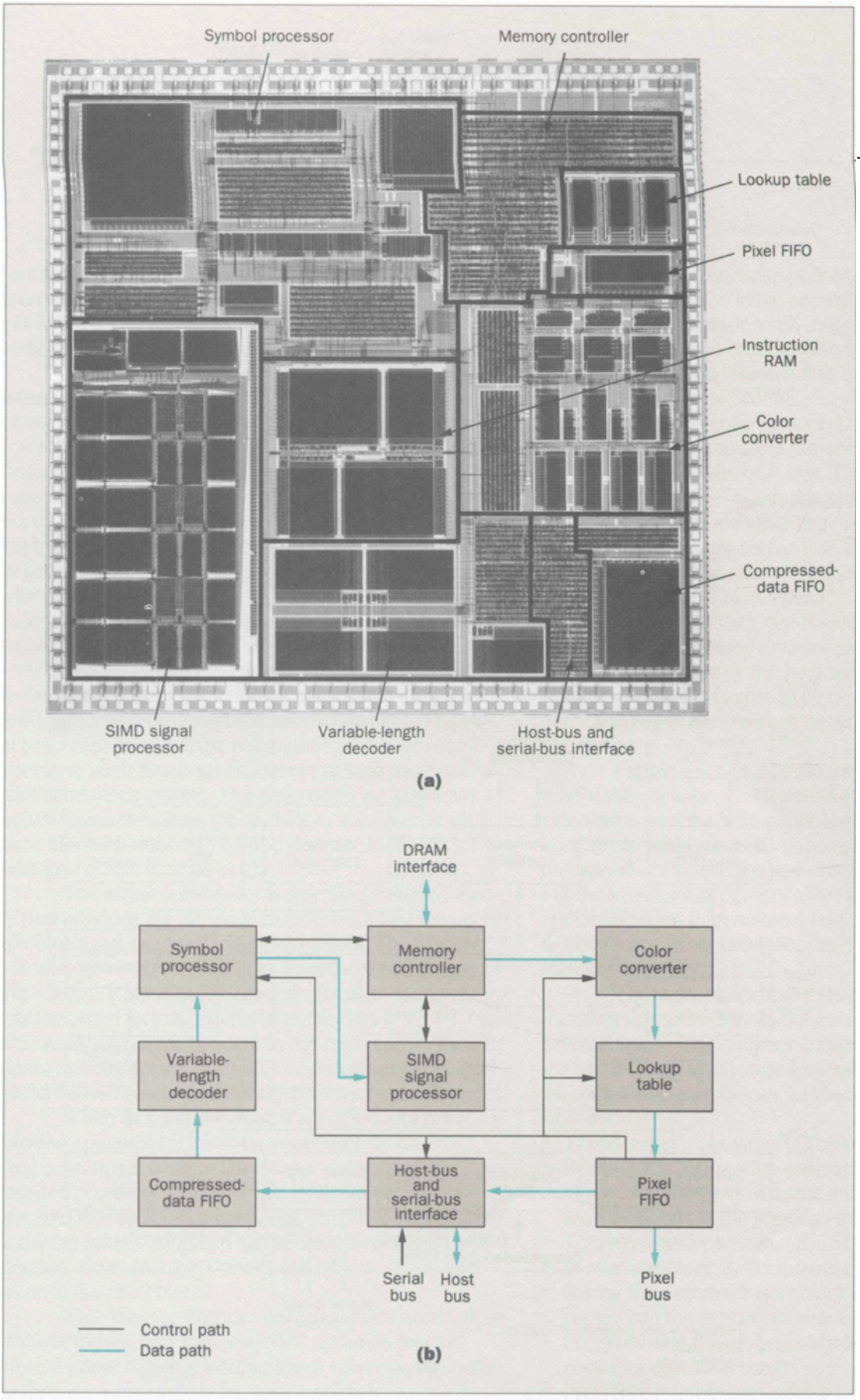


Figure 3. The video-decoder chip (AVP-4220D) decodes full-motion video, according to the P×64 and MPEG standards. (a) The chip was designed using full-custom and standard-cell techniques. It was built using a 0.9-μm CMOS process, contains 1.2 million transistors, and occupies an area of 145 mm^2. (b) The decoder accepts compressed-video data on either its serial-bus interface or its parallel host-bus interface at rates up to 4 Mb/s. It outputs the decoded video (in raster-scanned order) on either its host bus or the dedicated pixel bus.

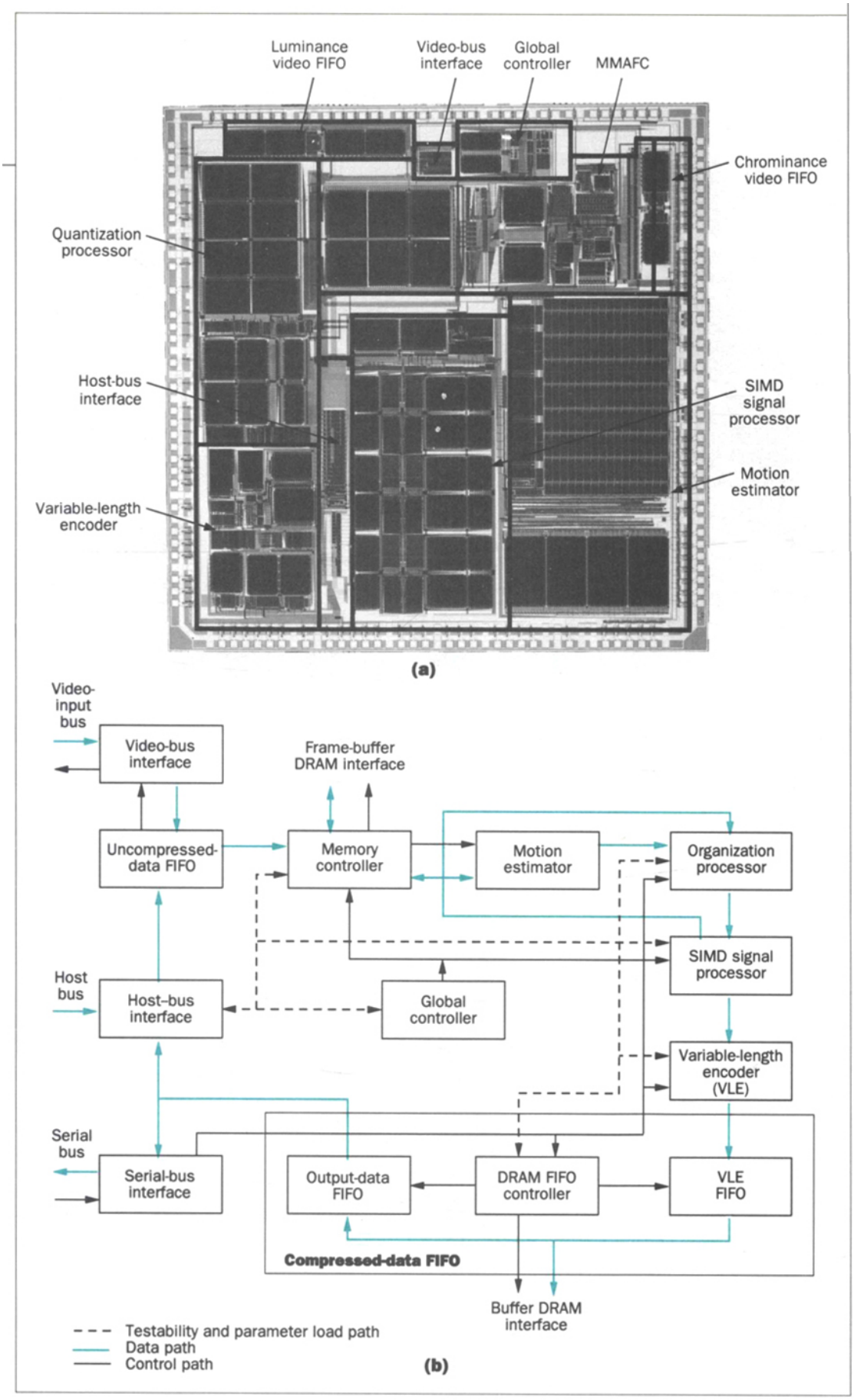


Figure 4. The video encoder (AVP-4310E) compresses full-motion video according to the P×64 standard and high-resolution still images according to the MPEG standard. It also supports MPEG intracoded-frame video encoding for digital-editing (i.e., authoring) applications. (a) The chip was designed as a full-custom device using symbolic layout and compaction, and was built using a 0.9-μm CMOS process. The 120-mm^2 die contains about 1.6 million transistors. (b) The encoder accepts video input in 4:1:1 YC_rC_b format at 30 frames per second on either its video input bus or the host bus. The output data rate is selectable, from 40 kb/s to 4 Mb/s.

The *variable-length decoder* is a ROM-programmable, finite-state machine that converts the hierarchical, variable-length codes specified by MPEG and P×64 into a series of symbols and data that are passed to the symbol processor.

The *symbol processor*—a programmable, reduced-instruction-set computer (RISC)—performs most of the control functions in the decoder. The symbol processor receives symbols from the variable-length decoder and issues commands to the memory controller and the signal processor. It uses the sparse data contained in the symbols to reconstruct the complete DCT-difference blocks, which are then passed to the signal processor. The symbol processor includes hardware to execute time-critical tasks (such as run-length decoding of DCT data) in a single instruction. The program is stored in on-chip RAM and can be downloaded from the host-bus interface.

The *signal processor* is a single instruction, multiple data (SIMD) machine made up of six processing elements and a ROM-programmed microcontroller. The signal processor provides the computational power required to perform such functions as inverse DCT, inverse quantization, and block update. The six processing elements operate concurrently on the six blocks (i.e., four luminance blocks and the two color-difference blocks that make up a single macroblock).

The *memory controller* is a preprogrammed module that provides the interface to external DRAM to support both block and raster access of video data. The memory controller moves decoded blocks from the signal processor to the DRAM. It also reads decoded data from the DRAM in raster-scanned format, interpolates the data from 4:1:1 to 4:2:2, and then moves the data to the color converter. The memory controller provides a direct interface to industry-standard DRAMs and automatically generates refresh cycles.

When 24-bit RGB output is specified, the *color converter* converts 4:2:2 YC_rC_b pixels into 24-bit RGB, using a multiplier-accumulator and a lookup table. Decoded pixels are stored in a 32-by-24-bit pixel FIFO. This FIFO also includes a user-programmable flag that is set during initialization and can be used to generate a host interrupt. From this FIFO, pixels can be sent to the host bus or to the pixel bus.

Note that the decoder only generates pixel values for those frames that were coded. It assumes that an external frame buffer is loading these values and generating the necessary signals for screen refresh.

Encoder

The video encoder (AVP-4310E)[7] is designed to compress full-motion video according to the P×64 standard and high-resolution still images according to the MPEG standard. The encoder also supports MPEG intracoded-frame video encoding for digital-editing (i.e., authoring) applications.

The video encoder accepts video input in 4:1:1 YC_rC_b format at 30 frames per second on either its video input bus or the host bus. The encoder supports variable video-spatial resolutions up to 288 lines by 360 pixels per line and still-image resolutions up to 1024 lines by 1024 pixels per line. Predictive-frame coding is based on exhaustive search-motion estimation, with a search range of ±15 pixels.

The output data rate is selectable, from 40 kb/s to 4 Mb/s. Compressed data can be made available on either the serial bus or the host bus. The encoder requires 1 Mbyte of external 60-ns DRAM to hold both the current input frame and the reference frame for predictive coding.

Figure 4 shows a photograph and the architecture of the encoder. The input—i.e., 4:1:1 raster-scanned video data—passes from either the video-input bus or the host bus to a 3-kb (kilobit) uncompressed-data FIFO, i.e., the luminance video FIFO and chrominance video FIFO in Figure 4a. A user-programmable FIFO flag can be set during initialization to generate a host interrupt at any user-defined level. From this FIFO, the data passes to the memory controller.

The *memory controller* (MMAFC in Figure 4a) transfers this video data to the current-frame buffer in external DRAM. The memory controller is a RISC processor whose program is stored in on-chip RAM. It is the main controlling processor on the encoder and is responsible for:

- Transferring input video to the DRAM.
- Reading into the motion estimator the next macroblock from the current frame and the appropriate search area from the previous frame.
- Transferring the best-match 16-by-16 block and reference macroblock to the quantization processor.
- Saving the decoded macroblock from the signal processor into the previous-frame buffer in DRAM.
- Programming the global controller (GC) before the encoding operation starts.
- Providing hundreds of software registers that the

user can set via the host-bus interface when the encoder is not actively encoding. Once coding begins, these registers control the operation of the other functional units.

In addition, the memory controller provides a direct interface to industry-standard DRAMs and automatically generates refresh cycles.

As input, the *motion estimator* takes a reference macroblock from the current frame, together with several macroblocks from the previous frame that define a search area around the position of the reference macroblock. It finds the pixel offset of a 16-by-16 block in the previous frame that provides a best match (in terms of summed, absolute difference) with the reference macroblock. It outputs a best-match motion vector to the quantization processor and provides the memory controller with address information to fetch the best-match 16-by-16 block. If the best match is poor, the motion estimator may decide to "turn off" predictive coding for that block and instruct the quantization processor to code it as an intra block (i.e., a block that is not coded with respect to the previous frame).

The motion estimator is implemented as a two-dimensional systolic array that consists of 256 processing elements. To conduct an exhaustive search over a ±15 pixel search range, the estimator needs to perform 6.5 billion operations per second.

The *quantization processor* is a RISC engine that takes the reference macroblock, subtracts it from the best-match 16-by-16 block, and passes the result (i.e., the difference macroblock) to the signal processor. However, this processor's primary function is to determine the correct level of quantization to apply to each macroblock to maintain a constant compressed-bit rate and good image quality. A program stored in on-chip RAM is used to execute an adaptive-quantizer, buffer-control algorithm. This algorithm adjusts quantization according to the fullness of a virtual output buffer that the variable-length encoder (VLE) maintains. The algorithm employs a large number of programmable registers that enable users to control the picture quality, bit rate, encoding delay, coded-frame rate, and so on.

The *signal processor* is a ROM-programmed, six-processing-element, SIMD array similar to the one used in the decoder. It performs the arithmetic-intensive tasks of the encoding algorithm. It accepts the difference macroblock from the quantization processor and performs a DCT on each of the component blocks. It then performs quantization, according to the value supplied by the quantization processor. Next, run-length encoding of the quantized values is applied, and the result is sent to the variable-length encoder.

The signal processor also performs inverse quantization and inverse DCT to construct a copy of the decoded macroblock to be stored in the previous-frame buffer.

The *variable-length encoder* is a ROM-programmed, RISC engine that takes run-length-coded DCT data from the signal processor and motion vectors from the quantization processor. From these, it produces a hierarchically coded bit stream, according to the syntax requirements of the P×64 and MPEG standards. Special hardware is included to generate and pack variable-length symbols at peak rates up 80 Mb/s. The variable-length encoder also keeps count of how many bits have been generated by the current frame. The quantization processor uses this count to maintain a virtual buffer and ensure correct operation over a constant-bandwidth channel.

The *compressed-data FIFO* is a 1-Mb (megabit) buffer that is used to smooth the bursty output of the variable-length encoder to support a reasonable rate at the serial output port. This FIFO is not intended to be the system-level video-data buffer (a function that is supplied by the systems-controller chip). The compressed-data FIFO consists of three parts:

- A 4-kb, *variable-length-encoder FIFO* (the VLE FIFO) that absorbs the direct output of the variable-length encoder.
- A *DRAM FIFO controller* that directly interfaces to an external 1-Mb DRAM chip. This controller operates the DRAM in page mode to give maximum bandwidth. Most of the cycles are preallocated to the writing function to absorb the large peak-data rates that may come from the variable-length-encoder FIFO. The remaining cycles are used to maintain a lower steady-transfer rate to the output-data FIFO.
- A 2-kb, *output-data FIFO* that buffers data between the buffer DRAM and either the serial-bus or host-bus interface. The peak-data rate at this point is 15 Mb/s. During initialization, a user-programmable flag can be set to generate an interrupt at any predefined level.

The *global controller* is a programmable-state machine that synchronously starts each of the processors in the encoder, based on the frame and macroblock boundaries. This controller ensures that all the data pipelines operate correctly at the prescribed frame rate.

The memory controller programs the global controller immediately before the start of coding. The serial-

bus interface also provides a serial input that the global controller uses to transfer information about real hardware-buffer fullness from the systems-controller chip to the encoder. This information allows corrective action to be taken if the encoder (which is locked to the camera-frame rate) and the systems controller (which is locked to the communications interface) drift too far out of synchronization.

Systems Controller

The systems controller (AVP-4120C), a multiple-protocol communications controller for interactive video and video teleconferencing, is designed to relieve the host system of much of the burden of handling and synchronizing compressed-video and -audio streams and the associated signaling. These functions include:

- Multiplexing, demultiplexing, and framing of P×64 audio, video, and user data based on the H.221 specification.
- Multiplexing and demultiplexing of compressed MPEG audio, video, and user data according to the ISO MPEG system-layer specification for digital storage media. Functions, here, include the MPEG-reference clock, presentation-time stamps, buffering, and bit-stream management.
- Forward-error coding and correction of the P×64 video stream based on the H.261 specification.
- Channel management, call setup, and call tear down based on the H.221, H.242, and H.230 specifications.
- Video buffering of transmitted and received data to maintain a constant transmitted-bit rate according to the H.261 specification.
- Audio buffering and delay adjustment, relative to the video data, to maintain lip synchronization.
- Combination and synchronization of two communications channels—including two Switched 56-kb/s channels, two Switched 384-kb/s channels, or two ISDN bearer or B channels—that may experience different delays as they travel through the network. (*Switched* means they are digital channels that use the switched, public telephone network rather than private lines.)

Communications-protocol activities are divided among multiple tasks that run on the host and the systems controller, which consists of an embedded processor (the *pacer*) that is assisted by several on-chip engines (i.e., hardware modules) dedicated to specific tasks. Time-critical activities that require extensive bit manipulations are delegated to the pacer and its attached hardware engines. Activities that are not time-critical (but are byte-aligned) are delegated to the host. The pacer and its attached engines are responsible for data movement, bit manipulation, and error-code generation and detection; and for assembly and disassembly of the H.221 frame.

Figure 5 shows a photograph and a simplified block diagram of the systems controller. The pacer is an embedded RISC processor whose instruction set was judiciously chosen for optimum performance of those operations that are normally needed in communications-protocol controllers. In this application, the pacer runs at 22.5 MHz (megahertz) and performs 24-bit integer instructions on any of its 256 internal registers in a single clock cycle. Each of the 24-bit registers can be addressed as any combination of 8-, 16-, or 24-bit data types or as a 24-bit pointer.

The pacer uses a Harvard architecture in which instructions and data reside in separate, distinct memories. Its dedicated instruction-memory port supports up to 64 kwords of 24-bit instructions (where k = 1024). Through its data-memory port, the pacer can address other sections of the systems controller, as well as the external DRAM. The pacer has a two-stage, I/O pipeline that allows independent execution of I/O instructions in data memory. Data-memory accesses are used to communicate with other sections of the device and to access the external DRAM through the built-in DRAM controller.

The *bit server* is a dedicated engine that is designed to efficiently handle the bit-level operations called for by the H.221 protocol. The bit server's functions include:

- Flow control of information bits for assembly into the H.221 frame.
- The assembly of bytes from various small groups of bits.
- Calculation of independent cyclic-redundancy checks for each channel.
- UART emulation. (UART stands for universal, asynchronous receiver–transmitter.)
- Fast-framing (i.e., find a frame—a structured bit stream—quickly).
- Generation and detection of forward error-correction-code, based on the H.261 protocol.

To do these functions with real-time software would have required an execution rate of 50 million instructions per second.

The *FIFO server* is a controller whose main function is to create large FIFOs in DRAM to facilitate system

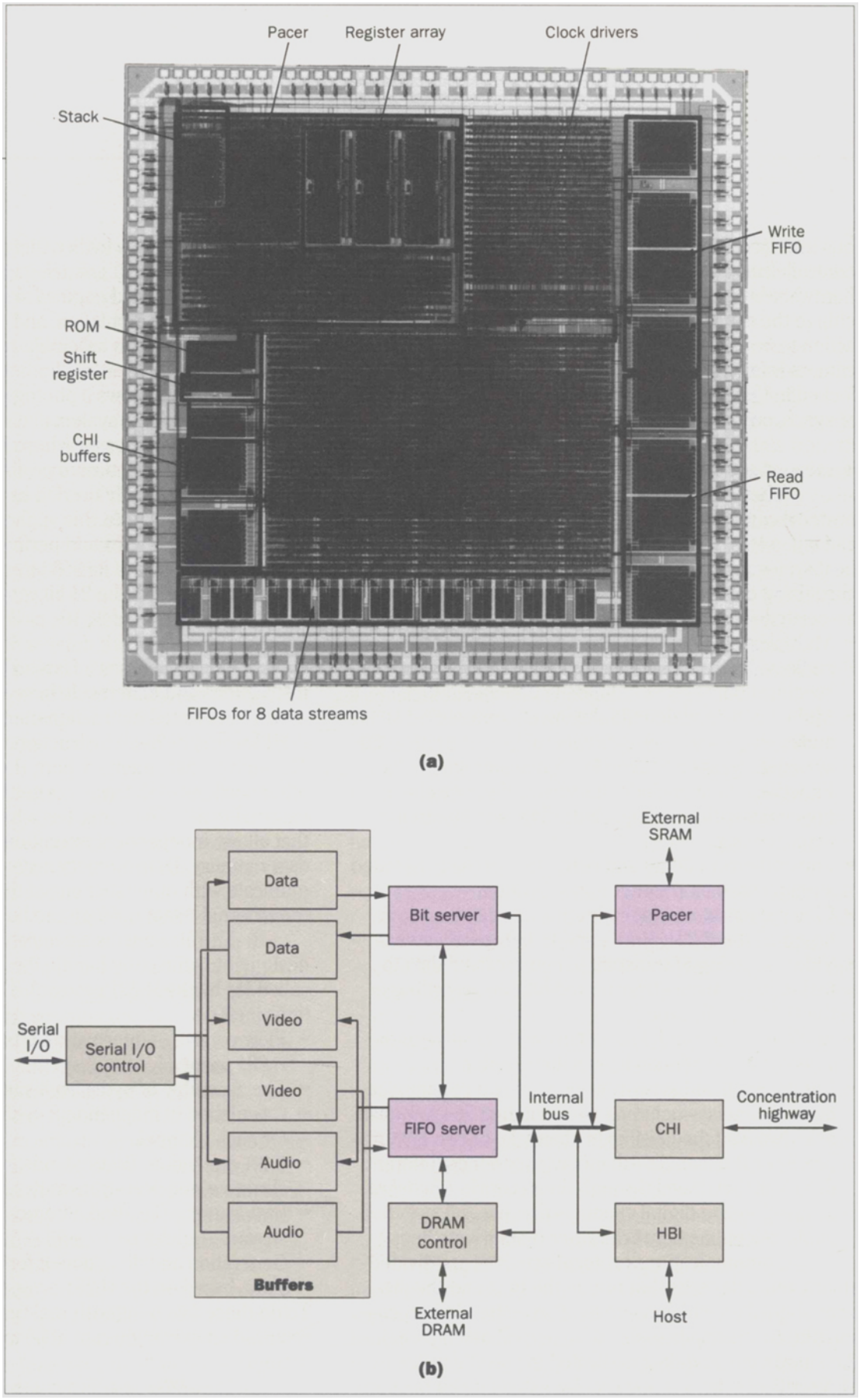


Figure 5. The systems controller (AVP-4120C), a multiple-protocol communications controller for interactive video and video teleconferencing, relieves the host system of the burden of handling and synchronizing compressed-video and -audio streams and the associated signaling. (a) The chip employs a hybrid mix of standard-cell and custom-layout techniques. Built using a 0.6-μm CMOS process, it contains 425,000 transistors and occupies an area of 115 mm^2. (b) The *pacer* (an embedded RISC processor) and its attached engines are responsible for data movement, bit manipulation, error-code generation and detection, and for assembly and disassembly of the H.221 frame. The *bit server*, a hardware dedicated engine, handles bit-level operations called for by the H.221 protocol. The *FIFO server*, a controller, creates large FIFOs in DRAM to synchronize the audio and video data, prevent information overruns and underruns, and recover from error conditions.

synchronization of the audio and video data, prevent information overruns and underruns, and provide recovery from error conditions. While the audio and video data pipes are served by the FIFO server, the primary and secondary data pipes are directly connected to the bit server. Each FIFO is selectively accessible through the host-bus interface in big-endian or little-endian format or via a serial bus with either the most-significant bit or the least-significant bit first.

The concentration-highway interface is a time-division-multiplexed serial bus that can be configured to connect to a wide range of telecommunications components. It can also be configured to be compatible with the timing of most existing systems. As a result, it gives the systems controller a simple interface to ISDN, pre-ISDN, and other synchronous-communications systems.

Other blocks provide additional support functions that enable the pacer to do all the needed protocol functions in real time. The host-bus interface provides a command-oriented interface for the portion of the P×64 protocol stack that runs on the host and on the application software. This interface also provides an alternate path for data transport. The systems-controller chip comes with protocol code to support P×64 video teleconferencing and MPEG authoring and playback, and suites of test code to facilitate system debugging and in-system DRAM and SRAM tests.

In addition to providing formatting and framing functions, the systems controller synchronizes the data to timing that is imposed by the communications network or by fixed-rate storage devices. The protocol code identifies synchronization patterns in multiple channels where the delay between channels may be different and builds code that recovers a coherent channel in real time.

The systems controller may also be used with the host-bus interface that handles the network traffic, instead of with the concentration-highway interface. Network traffic must be redirected to the host-bus interface when an existing bus-based subsystem, such as a local-area network, is being used to move synchronous data. In these applications, the device does not rely on the concentration-highway interface to control the rate of data transport. Instead, an internal timer controls the transmit and receive rates. The rate of the timer is adjusted dynamically to match the aggregate rate of data transport of the network. In this way, the entire system still synchronizes its timing to the network, but the host system can transfer blocks of data in bursts to and from the systems controller.

Regardless of the source of networking timing, a feedback path dynamically adjusts the serially connected encoder's rate of video generation to match the network and control the quantization steps the encoder uses. The FIFO server delays the audio to achieve lip-synchronization with the video, and controls the audio sampling rate to match the network-transfer rate.

Silicon Devices

Preproduction models of the three multimedia-codec chips have been designed and built using AT&T Microelectronics' 0.9-μm (micrometer) and 0.6-μm CMOS processes. Figures 3a, 4a, and 5a are photographs of the three chips.

The video encoder (Figure 4a) has been designed entirely as a full-custom device using symbolic layout and compaction. (*Full custom* means that every circuit was custom made, and was not from a components or circuits library.) The 120-mm^2 (millimeters square) die contains about 1.6 million transistors and runs at 45 MHz under worst-case power, voltage, and process parameters. It dissipates about 5 watts at 45 MHz and is packaged in a 159-pin, plastic-pin grid array with a heat-spreading metal insert. Production devices are expected to dissipate less than 5 watts.

The video decoder (Figure 3a) has been designed using a mixture of full-custom and standard-cell techniques. It contains 1.2 million transistors and occupies an area of 145 mm^2. The production device will occupy less than 120 mm^2. At 45 MHz, the device dissipates about 2.5 watts. The decoder is packaged in a 159-pin, plastic-pin grid array.

The systems controller (Figure 5a) employs a hybrid mix of standard-cell and custom-layout techniques. Wherever possible, standard-cell techniques were used to maximize the design's portability to other technologies. The pacer block was laid out separately to optimize performance of both the arithmetic-logic unit and other critical data paths.

The systems controller contains 425,000 transistors and occupies an area of 115 mm^2. Its power dissipation at 45 MHz is measured at 2 watts under worst-case processing, electrical, and environmental conditions. This chip will be manufactured using the AT&T two-level-metal, 0.6-μm CMOS process, and is packaged in a 156-pin, plastic-pin grid array.

Summary

The three multimedia-codec chips described in this paper represent the first generation in a family of devices that will make possible reliable, cost-effective storage and transmission of multimedia data, including video. The goal in designing these chips has been to provide a complete multimedia platform that addresses a wide spectrum of capability, quality, and bandwidth and complies with emerging standards. The combination of readily available, digital-communications services; low-cost, VLSI codec chips that comply with standards; and powerful, multimedia computing platforms will make possible a range of products and services that we can only begin to imagine. As these applications evolve, new versions of these chips will be required that are tailored to specific markets and optimized for performance, power, or cost.

References

1. A. N. Netravali and B. G. Haskell, *Digital Pictures: Representation and Compression*, Plenum Press, New York, New York, 1988.
2. S. H. Early, A. J. Kuzma, and E. Dorsey, "The VideoPhone 2500—Video Telephony on the Public Switched Telephone Network," *AT&T Technical Journal*, Vol. 72, No. 1, January/February 1993, pp. 22–32.
3. A. C. Luther, *Digital Video in the PC Environment*, Second edition, Intertext Publications, McGraw-Hill, New York, New York, 1991.
4. AT&T, *VCOS Multimedia Development Kit, Technical Reference Manual*, AT&T Microelectronics, 555 Union Boulevard, Allentown, PA 18103, 1992.
5. R. Aravind, G. L. Cash, D. L. Duttweiler, H–M. Hang, B. G. Haskell, and A. Puri, "Image and Video Coding Standards," *AT&T Technical Journal*, Vol. 72, No. 1, January/February 1993, pp. 67–89.
6. D. M. Brinthaupt, D. L. Letham, V. Maheshwari, J. H. Othmer, R. R. Spivak, B. Edwards, C. Terman, and N. Weste, "A Video Decoder for H.261 Video Teleconferencing and MPEG Stored Interactive Video Applications," *Proceedings of the 1993 International Solid-State Circuits Conference*, San Francisco, California, February 24–26, 1993, IEEE, San Francisco, to be published February 1993.
7. S. Rao, M. Hatamian, M. Uyttendaele, S. Narayan, J. O'Neill, and G. Uvieghara, "A Real-Time P*64/MPEG Video Encoder Chip," *Proceedings of the 1993 International Solid-State Circuits Conference*, San Francisco, California, February 24–26, 1993, IEEE, San Francisco, to be published February 1993.

(Manuscript received June 17, 1992)

Bryan D. Ackland *is head of the VLSI Systems Research Department with AT&T Bell Laboratories in Holmdel, New Jersey. He is responsible for research into architectures, circuits, and tools for full-custom VLSI design in high-performance signal-processing and communications applications. Mr. Ackland joined the company in 1978. He has a B.Sc. in physics from Flinders University, Australia, and both a B.E. and Ph.D. in electrical engineering from the University of Adelaide, Australia.*

Reza Aghevli *is a technical manager in the Video Subsystems Department with AT&T Bell Laboratories in Allentown, Pennsylvania. For the past several years, he has been involved in the design and development of communications-protocol controllers to support AT&T's digital-telecommunications network for voice and data applications, and to support local-area networks. His most recent activity has involved development of the systems-controller VLSI for audio–video teleconferencing and multimedia applications. Mr. Aghevli joined the company in 1982. He has a B.S. in electrical engineering from The University of Tehran, Iran, and both an M.S. and Ph.D. in electrical engineering from The University of Michigan in Ann Arbor.*

Ismail Eldumiati *is head of research and development for AT&T's Visual Communication Solutions business unit in Holmdel, New Jersey. He is responsible for R&D activities in the areas of video conferencing, video telephony, and desktop applications. His past work included integrated-circuit development for digital-signal processing, telecommunications, and data communications. Mr. Eldumiati joined Bell Laboratories in 1972. He received a B.S.E.E. from Alexandria University, Egypt; and an M.S.E.E., an M.S. in physics, and a Ph.D. in electrical engineering from The University of Michigan in Ann Arbor.*

Arnold C. Englander *manages the video codec VLSI program and is responsible for multimedia and video-telephony market development for AT&T Microelectronics in Berkeley Heights, New Jersey. Mr. Englander joined the company in 1990. He has M.S. degrees in electrical engineering and architecture from Yale University in New Haven, Connecticut.*

Eugene Scuteri, Jr. *is a supervisor in the Video Subsystems Department with AT&T Bell Laboratories in Holmdel, New Jersey. He manages the encoder and decoder device-design effort and is part of the system-definition team. Mr. Scuteri joined the company in 1979. He has a B.S.E.E. from Fairleigh Dickinson University in Teaneck, New Jersey, and an M.S.E.E. from the Polytechnic Institute of New York in Brooklyn.*