The Evolution of DVI System Software

The desktop computer has been undergoing a transformation from a text and numeric processing system to a powerful tool for managing and communicating all types of digital information. Audio, photo-realistic images and motion video have joined the ranks of other digital data types available to application developers [2]. The addition of these "multimedia" data types places new requirements on the computer systems and their operating environments: Multimedia data consumes massive amounts of storage space, and playback of full-motion digital video requires real-time decompression and high I/O bandwidth [7].

DVI® multimedia products include a set of hardware and software tools that integrate the full range of these new data types in a desktop computing environment. A typical DVI system includes a personal computer or workstation with Intel's ActionMedia™ playback card, system software, and a CD-ROM drive. With the addition of the capture card and peripherals (such as a videotape player, audio cassette player or camera), the user or application developer can create multimedia elements as well as play/display them. Since all of the DVI presentation materials are in digital form, they can be stored on any random-access storage device or even transmitted across a network [16].

James L. Green

The development of DVI technology started in 1983 when a group at the David Sarnoff Research Center in Princeton, N.J., began examining the feasibility of combining the interactivity of computers with the realism of television. The results were first demonstrated publicly at the Second Annual Microsoft CD-ROM Conference in March of 1987 [5]. Two years later, the technology was purchased by Intel Corporation. Intel introduced the first board products based on this technology in July 1989.

The system software included with the first generation of DVI products only ran on IBM-compatible computers under the MS-DOS operating system. This article describes how DVI system software is evolving to take advantage of more powerful hardware and operating system environments. The article begins by examining the conceptual basis, the data flow, and the task-scheduling mechanisms employed by the current ActionMedia software system to play digital audio and video. The task-scheduling solutions employed by other multimedia systems will be discussed in this article as well. The next-generation software architecture, called the Audio Video Kernel (AVK), will be discussed in the final section of this article, illustrating the changes in these areas.

DVI is a registered trademark of Intel Corporation. ActionMedia is a trademark of Intel Corporation

The ActionMedia Software System

DVI system software is available currently on PC/AT or PS/2 platforms running MS-DOS. This system, known as the ActionMedia 750 Software Library, is comprised of three main subsystems: the Real-Time Executive (RTX), the Audio/Video Subsystem (AVSS), and a special graphics library. RTX provides real-time multitasking services to AVSS as well as to DVI applications. AVSS is used to control playback of digital audio/video files. The graphics library provides special-purpose video effects, as well as more traditional drawing primitives and fonts [11]. The conceptual model used to describe this system is a "super VCR."

The Super VCR Model

The "playback unit" in Figure 1 is the AVSS subsystem itself. Control functions for play, stop, pause and frame advance are provided as C language function calls. Applications can play an audio/video file in much the same way a VCR plays a tape. The "effects unit" is actually a collection of functions in the graphics library. Graphics can be added to video as it plays via special functions called "hook routines." A hook routine is an application-supplied C function that is registered with AVSS by passing it a pointer to the function. Hook routines encapsulate a set of DVI graphics operations that will be executed on each video frame after it is decompressed, but before it is displayed. AVSS automatically calls the hook routine at the proper time. An application can provide the user with interactive control over AVSS through keyboard or mouse input which are treated as "events" by RTX.

AVSS Data Flow

AVSS uses three parallel operations, implemented as RTX tasks, to play digital video: the *server task* reads a frame of compressed video into memory, a *decode task* requests that the frame be decompressed by the pixel processor, and a *display task* displays the decompressed frame on the computer monitor. These tasks must be performed 30 times a second in order to play a 30-frame per second motion video sequence smoothly [11].

The AVSS data-flow diagram in Figure 2 illustrates how the three tasks interact with the data buffers. The data flows from the storage device into the input buffer as compressed data. Each frame is then decompressed into one of the bitmaps in an array called the "screen array." If a hook routine has been installed by the application, it is called so that graphics can be applied before the frame is displayed. The screen array bitmaps are reused in a circular fashion. There are two bitmap resolutions that can be used in the screen array, 256 × 240 and 512 × 480. Source video is stored at 256 × 240 resolution. If the screen array is also 256 × 240, the decompressed video will fill the bitmap, and the resulting image will

Glossary of Acronyms Used in This Article

API	Application-Program Interface
AVD	Audio/Video Driver
AVK	Audio Video Kernel
AVL	Audio/Video Library
AVSS	Audio/Video Subsystem
CD-I	Compact Disk-Interactive
CD-RTOS	Compact Disk Real-Time Operating System
CMEX	Continuous Media Extensions to X-Windows
DVI®	Digital Video Interactive (registered trademark of Intel Corporation)
JPEG	Joint Photographic Experts Group
MCI	Media Control Interface
MPEG	Motion Picture Experts Group
NTSC	National Television Systems Committee of the Electronics Industries Association
PLV	Production-Level Video
RTV	Real-Time Video
RTX	Real-Time Executive

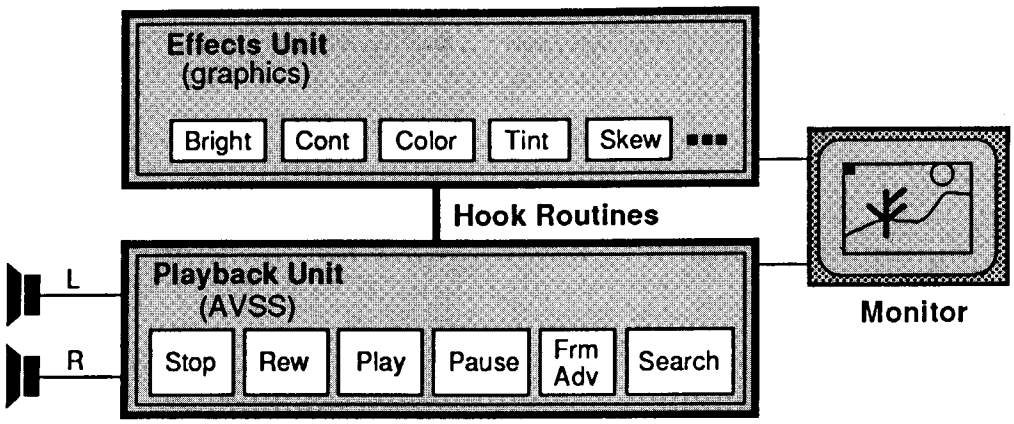


FIGURE 1.
AVSS Conceptual Model

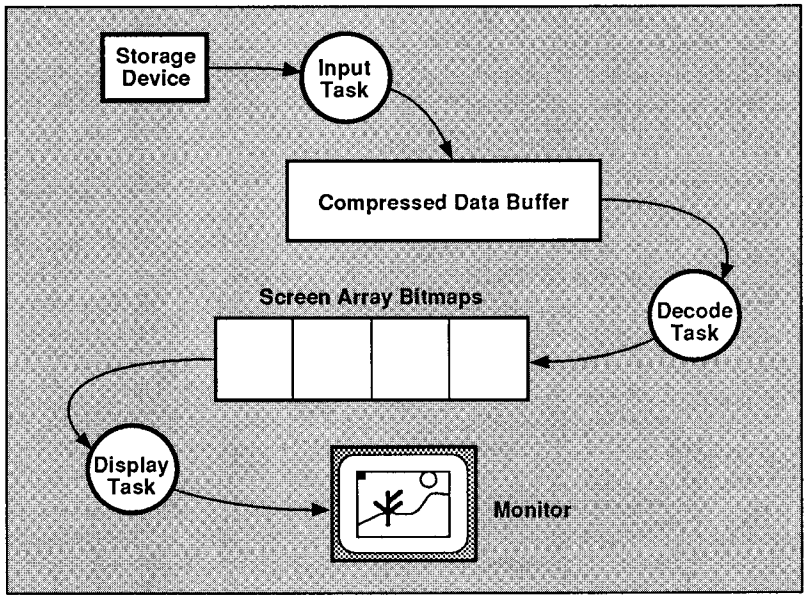


FIGURE 2.
AVSS Data Flow

fill the computer's screen. If the screen array uses 512 × 480 bitmaps, the video will only fill one-quarter of the bitmap and the screen.

The MS-DOS BIOS and file system provide only "waited" read and write operations. The application must wait for the read or write to complete before it can continue processing. Due to the real-time requirements of playing digital video, ActionMedia software includes its own asynchronous read functions for a select number of hard drives, thereby circumventing the DOS BIOS and file system. These functions allow AVSS to continue scheduling the decompression and display tasks while the data is being transferred into memory.

Task Scheduling Using RTX

RTX assumes all of the system's resources are available for its use. It takes control over a number of the interrupt vectors, particularly the timer interrupt, and intercepts a number of calls that would normally be handled by DOS. RTX's first responsibility is to provide CPU resources to AVSS so the motion video plays smoothly. ActionMedia applications can create their own tasks under RTX, but they must be carefully designed so they do not interfere with AVSS tasks and they can actually get a chance to run under the RTX scheduling scheme. This is done by assigning the task the proper priority and making sure it issues an event wait sometime during its execution cycle. RTX priorities range from zero (the highest priority) to 15 (the lowest). The AVSS server task, decode task, and display task run at priorities 3, 4, and 5 respectively. Application tasks with lower priorities will always be preempted by the AVSS tasks. The event-wait condition is controlled via a function call in the RTX library. Common events include keyboard, mouse, and timeout events. The RTX scheduler is activated periodically based on a host-timer interrupt, and places the currently running task at the head of a "ready" list for its priority. It then hands control to the task at the head of highest priority ready list. Although the RTX scheduler and all application tasks are running on the host CPU, two of the AVSS tasks, the decode and display tasks, are actually executed by the DVI processors.

New Requirements and the AVSS Architecture

The AVSS VCR model works well for applications that treat video files like tapes (i.e., as self-contained presentations to be passively observed). Simple training applications, multimedia presentations, or on-line "video help" are

examples well-suited to the VCR paradigm. The current ActionMedia software includes interactive graphics effects and dynamic mixing of audio streams, functions that stretch the limits of the VCR model. Applications such as the Truck Driver Safety trainer developed by Applied Optical Media [16] or the Code Review Training application developed at Carnegie-Mellon University [18] demonstrate capabilities that cannot be easily described using a VCR. The next generation of DVI system software will enable even more advanced applications such as video conferencing or multimedia electronic mail [3]. A more sophisticated conceptual model is required that can be used to describe this broader class of interactive multimedia applications.

The AVSS decompression and display tasks share the bitmaps in the screen array. This prevents the application from controlling more than one stream at a time. In order to allow for the possibility of multiple streams, the decompression bitmaps and the display bitmaps will have to be separate areas of memory with the video data being copied from one to the other. The copy could also include a scale factor which would allow video to be displayed, relocated or resized in a windowing environment.

The AVSS/RTX architecture was implemented specifically on a DOS platform and is not easily ported to new environments. Thus the RTX technique for scheduling the required tasks needed to be reimplemented or replaced. The AVK team identified two requirements for the new tasking mechanism. It has to be portable across operating systems, (some of which support preemptive multitasking and some of which do not), and it should minimize the reliance on the host CPU for real-time response.

Alternate Task-Scheduling Strategies

In developing AVK, Intel considered a number of strategies for making task-scheduling and data-flow tasks as efficient as possible. A number of other companies have recently described their approaches as well. Compact Disk-Interactive (CD-I), a system developed by Philips, Sony, and Microware corporations, is a self-contained system aimed primarily at the consumer market. It consists of a "player" that is hooked up to standard home television and stereo equipment [8]. When video functionality becomes available for CD-I players, the video bitstream will be created offline using a parallel processing machine. The CD-I player hardware will include a VLSI decoder chip, a block of DRAM, a video keying subsystem, and a dedicated CPU. Coupled with the CPU is a ROM containing a real-time operating system called CD-RTOS (Compact Disk Real-Time Operating System). The CPU reads the encoded video stream from the CD-ROM and delivers it to the decoder chip. A system controller inside the decoder chip is responsible for satisfying the real-time requirements of the playback stream based on timing data extracted from the encoded bitstream [17]. Because the CD-I player hardware is fully self-contained, there is no concern for cross-platform compatibility, and its applications can assume full control over the environment. This is similar in principle to the AVSS/RTX strategy, where it is assumed that only one application is running and that it could control all of the computer's resources. The main difference between these systems is that CD-RTOS will not have to coexist with another operating system the way RTX had to cooperate with MS-DOS.

A group of researchers at the University of California at Berkeley have proposed an extension to X-Windows, called Continuous Media Extensions to X-Windows (CMEX). The CMEX server consists of a set of processes running on the workstation CPU that are scheduled preemptively with real-time deadlines. These processes handle the input/output of continuous media to a compression/decompression engine such as the 82750PB. After decompressing a frame of video, the 82750PB would issue an interrupt to the host CPU. An interrupt handler activates the I/O process and sets a deadline for it so the next frame of data will be transferred in time for it to be decompressed. It then signals the compression/decompression engine to begin executing on the data [10]. This strategy is similar to RTX because the job of scheduling the various tasks is left to the host CPU, although it differs in one important respect. RTX was designed to assume control of the host CPU from its operating system (MS-DOS) in order to achieve multiprocessing functionality, whereas CMEX uses the multiprocessing features of its host operating system (Unix). The scheduling strategy employed by CMEX could be used on any multiprocessing operating system given a sufficiently powerful CPU and I/O bandwidth. In fact, an early prototype of AVK was implemented on OS/2 using this strategy. Of course, operating systems such as DOS and, more importantly, DOS/Windows do not provide the preemptive multitasking support required for a system like CMEX.

A different strategy is employed by Fluent Machines, Inc. in their Fluency system. This system includes two add-in cards for ISA-bus machines equipped with 33MHz 80386 CPUs. This system employs a dual coprocessor strategy for processing continuous multimedia data. An Intel 80960 RISC processor running a proprietary real-time operating system called FluentStreams schedules tasks and synchronizes multiple audio and video data streams. Another processor executes the video compression/decompression tasks. This strategy offloads all video processes from the host CPU, allowing it to operate in virtually any operating-system environment [20]. This dual coprocessor strategy is an elegant technical solution, but results in significantly more expensive hardware

In developing AVK, Intel considered a number of strategies for making task-scheduling and data flow tasks as efficient as possible.

than Intel's single video processor approach.

A New Conceptual Model

For the next generation of DVI system software, a new model has been developed: the digital video production studio. A typical production studio contains mixers, tape decks, monitoring systems, effects processors and other items that connect together to record, modify and play audio or video tracks [6]. In our conceptual model, a digital production studio is composed of an analogous collection of subsystems that operate on digital streams of audio and video data. A collection of streams intended to play together forms a "group." Analog inputs and outputs can be thought of as "channels." Streams of audio and video data are routed from input channels to output channels by making "connections." Streams can be "mixed" together (e.g., assigned to the same output channel) or they can be routed through an "effects processor" to alter the data in some way. The main components of the digital production studio include the analog device interface, the display manager, the sampler, the stream manager, the effects processor, and the audio/video mixer. These components are illustrated in Figure 3.

The Nature of Compressed Digital Video

A CD-ROM drive can deliver data at a maximum rate of 150KB per second. On the other hand, broadcast video (i.e., NTSC) delivers data at a rate of 30 frames per second. These two rates, one at the front end for data delivery, and the other at the back end for data consumption, define the requirements for any software architecture that displays full-motion video from digital data.

Dividing the data rate (150KB/sec) by the frame rate (30 frames/sec), results in a budget of 5KB per frame. Current Intel off-line compression methods begin by digitizing each frame at a resolution of 512 × 480 pixels with a color depth of 24 bits per pixel, which results in roughly 720KB of data per frame. After digitizing, the compression process must reduce the data for a frame to 5KB. There are several compression techniques available, but for our purposes the most important one is "interframe coding." This technique, used by Intel and (in a modified form) by the proposed MPEG standard, involves storing information only for the pixels that change between frames. A frame for which all the pixels are defined is called an "intra" frame. Intraframes are used as random access entry points into the video data stream. Other frames, called "predicted" frames, are constructed from previous frame data. Only the data needed to change the last frame into the next frame is stored in the bitstream [12]. Interframe coding provides a dramatic reduction in the data rate required to deliver digital video data. However it introduces variability in frame size. Frames can be quite large (for an intraframe) or quite small (for a predicted frame). Given the variability due to interframe coding, the data rate constraint can now be restated as: The worst case moving average size per frame must be less than 5KB.

Because the size of each frame varies, the time required by the pixel processor to decompress it also varies. By buffering data in video memory, the effects of these variations can be overcome, so that a new frame of information is available for display every 30th of a second. Since video memory is generally a limited resource, the buffering scheme should use it as efficiently as possible.

The Analog Interface

One of these components, the analog interface, is essentially a patch bay. It allows the system to be connected to various analog input and output devices such as a laser disc player, VCR, camera, and speakers. These connections are called "physical channels" because they map directly to external physical devices. A physical channel is unidirectional, and acts as either an input or an output channel, carrying a single stream of data. These connections are also physical in the sense that they correspond to the input/output features of the DVI hardware. The ActionMedia Series II hardware allows video output to a computer monitor (via a 15-pin DIN connector), a Super-VHS recorder or monitor (via a Y-C connector), and audio output (via a stereo miniplug). Video and audio inputs are combined into a single connector plug (via an 8-pin mini-DIN connector). Digital-to-analog (D/A) conversion is performed on all output channels. Analog-to-digital (A/D) conversion is performed on all input channels.

The Display System

The display subsystem controls the display of the visual data on the computer screen. Multiple visual data types (still images and video) can be displayed simultaneously on the same screen. This collection can be thought of as the user's "view" of the data, and there may be more than one view defined in the system. The display subsystem allows for the creation of these views, and for selecting the view to be displayed (or "monitored") at any given time. This is the video equivalent of having a number of submixes, only one of which can be monitored at a time.

The Sampler

A sampling synthesizer, or sampler, is a common accessory in audio production studios. An audio sample records a waveform created by an acoustic source as a sequence of digital values in memory. The sounds can then be altered by manipulating these values. This idea can be extended to include visual data. A digital representation of a photograph can be thought of as a "visual sample" which can be manipulated in memory before being displayed.

In our model, the sample is implemented as blocks of video memory that can be used for temporary storage of audio or video data. A frame of data can be sampled from a video file and stored as a still image. An image can be routed through the effects processor and transformed (by altering the visual attributes such as the tint, brightness, etc.) or converted (from one bitmap format to another) and then placed back into the sampler and displayed or stored to disk.

The Stream Manager

The stream manager, much like the transport controls on a tape deck, is responsible for managing the flow of digital data to or from a storage device such as a CD-ROM, hard disk or local-area network. Streams "flow" through logical channels like tracks on a tape. Unlike data streams associated with physical channels, streams from logical channels do not necessarily map one to one with a physical device. A device such as a CD-ROM may provide a single file of interleaved audio/video data to the stream manager that maps to several logical channels (e.g., one for video and two for audio). Logical channels are bidirectional (except for the CD-ROM). This means that the same channels can be used for either input or output, although not at the same time.

The Effects Processor

The effects processor can be used to add image effects and graphics to stills and video, or to alter the characteristics of audio data. The effects processor serves the same function as the "effects unit" in the AVSS VCR model. It can be used to add image effects and graphics to stills or video as it plays.

The Mixer

The audio/video mixer is the heart of the production studio. It provides mechanisms for assigning incoming streams to output channels as well as the reverse. There are different types of streams and channels for audio and video data. For example, the left and right audio channels can be digitized and combined into a single stereo stream. Video channels may provide continuous motion streams, or still images.

While an audio input channel produces a single audio track, the mixer can combine multiple audio tracks in various ways to feed audio output channels. The mixer can also manipulate the streams on their way through the system. In a recording studio, audio tracks can be "panned" between the left and right outputs by turning a potentiometer from left to right. Digital audio streams can be assigned to an output channel in varying percentages ranging from 0 to 100 each for both the left and right channel.

Mixing video streams to the video output channel is simply a matter of defining the source object (a video stream or a still image) and the destination object (the application's "view" on the computer screen). Several sources can be assigned to the same destination creating a composite image, or a "visual mix." A mixer typically provides some controls for modifying the data. In the case of digital video data, this includes defining rectangular regions of the source and destination object for cropping or scaling, and altering attributes such as tint, contrast, brightness, and color saturation.

Bandwidth Saturation

In an audio production studio, if too many input channels are assigned to the same track the tape can become "saturated," which results in poor-quality sound. In the same way, there are physical limitations on the number of streams and groups that can be manipulated in real time by a computer. For exam-

ple, data rates of various devices, the host bus bandwidth, and the number of available CPU cycles, all impose limits on the ability of the computer to handle these complex data types. These limiting elements are described in the digital production studio model in terms of "saturation."

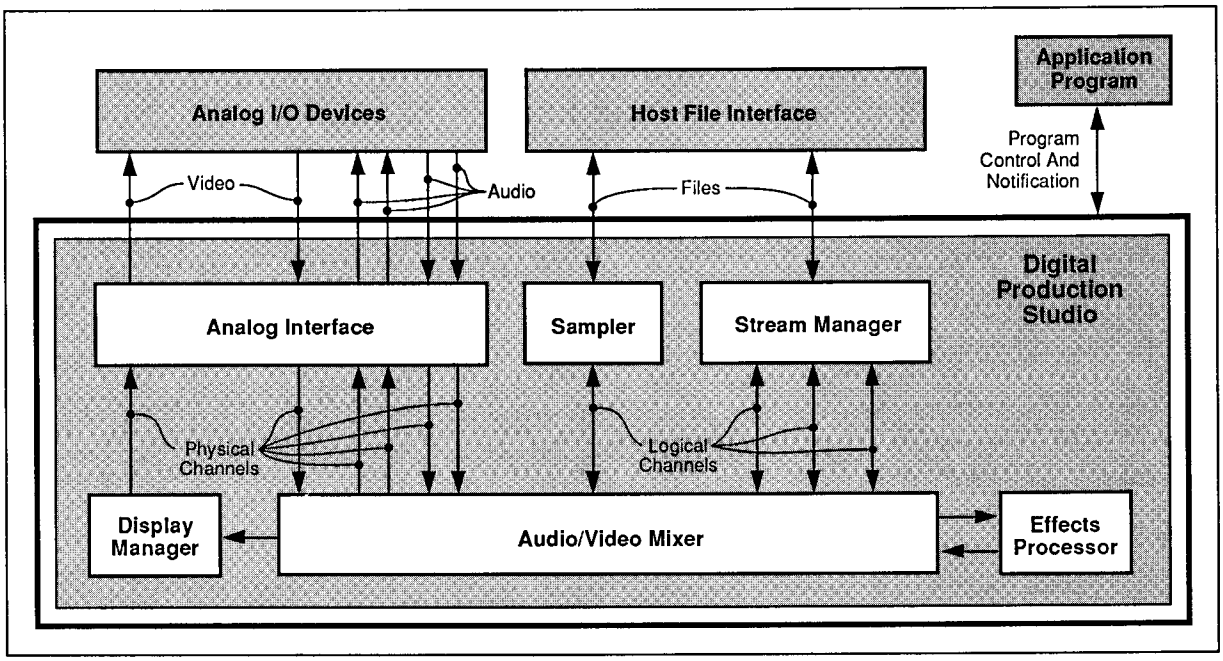
Logical tracks can be assigned so they exceed the data rate of a particular device such as a CD-ROM or hard disk. As a result, the recording data may miss frames, or the file may not play back properly. In either case, the bandwidth of the device can be said to be saturated.

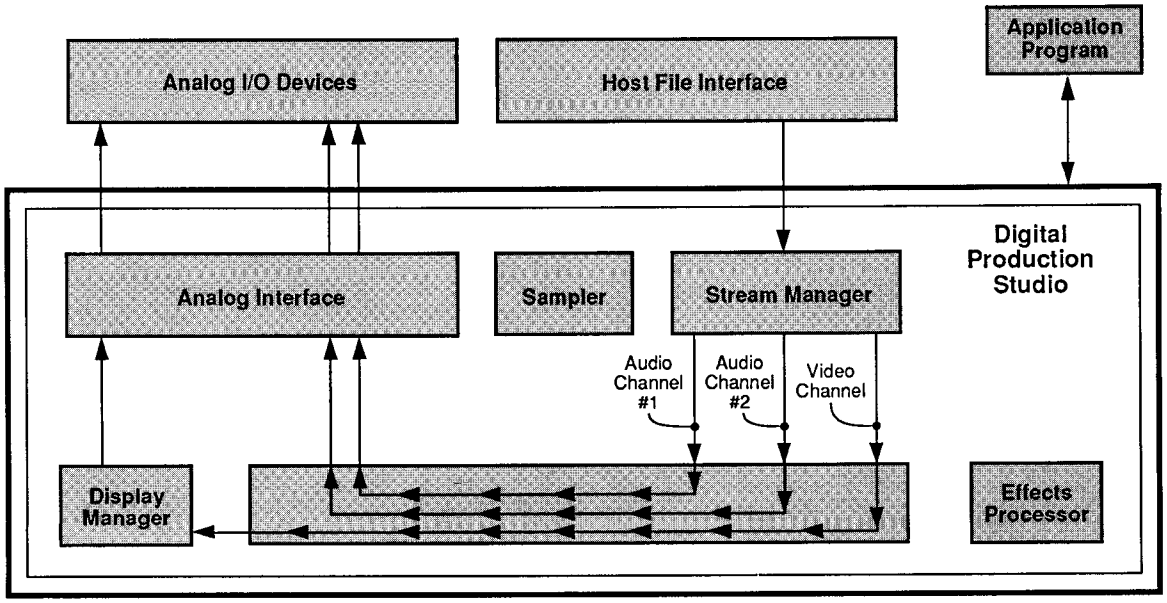
Another form of saturation occurs when too many effects are installed concurrently. The video and audio processors also have a limited bandwidth, and it is possible to overload them with more work than they can perform in real time while video is playing. This results in visible and/or audible artifacts such as dropped frames, slower playback, or audio noise.

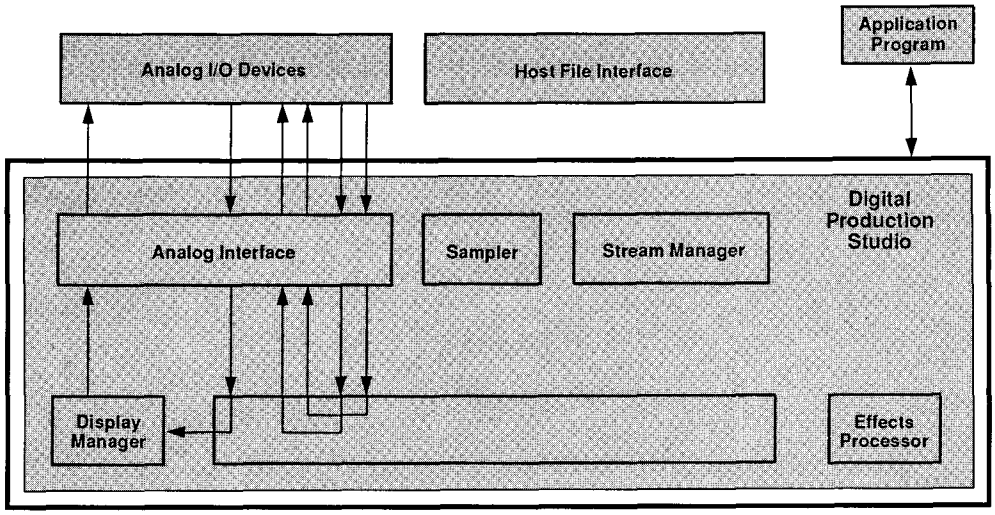
Another limiting factor is the

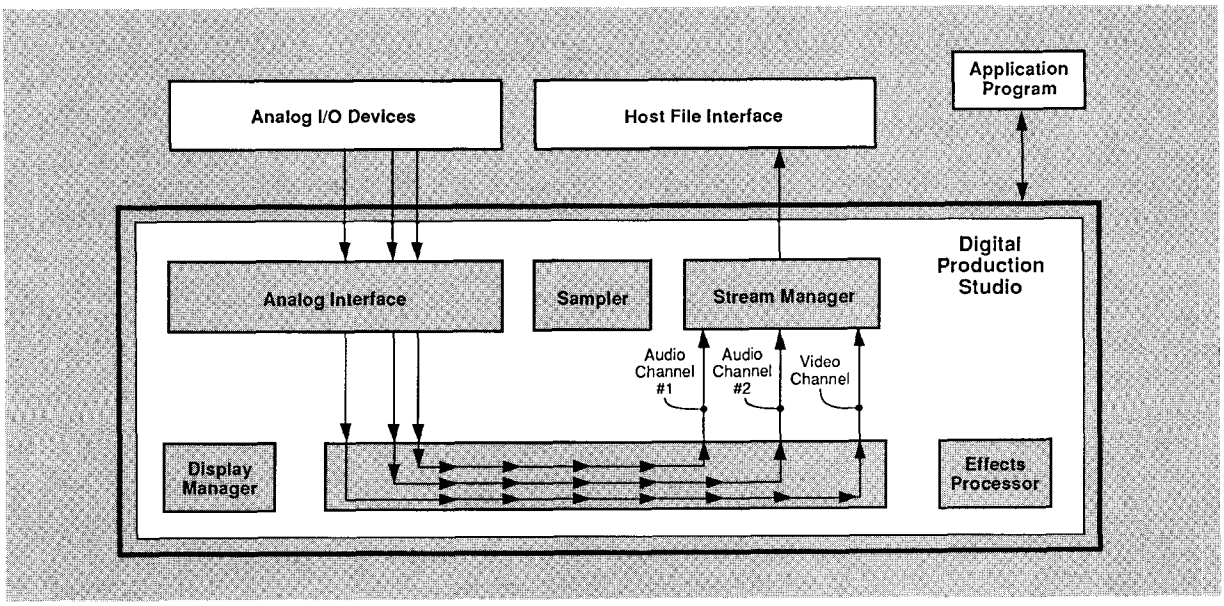
FIGURE 3.
Conceptual Model for Multimedia Computing

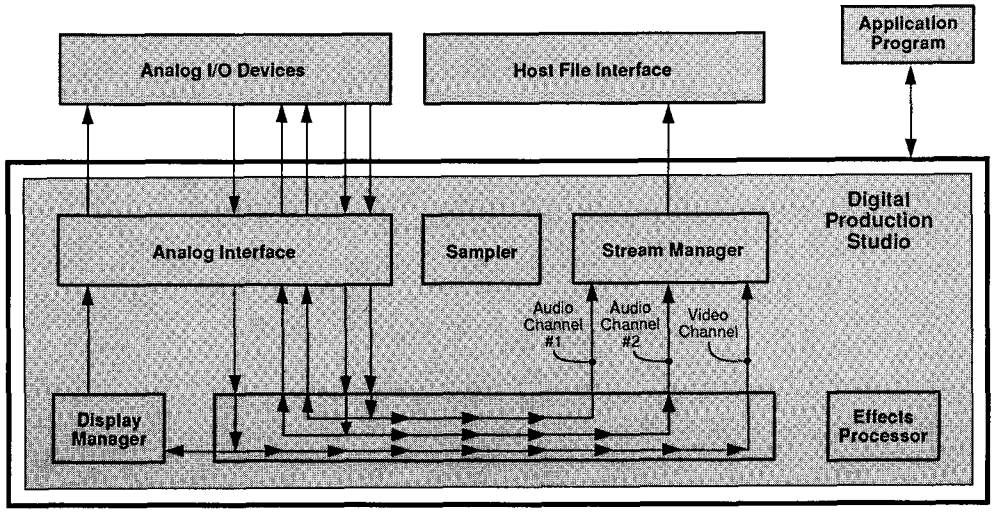
FIGURE 4.
Play Function






Application Program
Analog I/O Devices
Host File Interface
Digital Production Studio
Analog Interface
Sampler
Stream Manager
Display Manager
Effects Processor


Application Program
Analog I/O Devices
Host File Interface
Digital Production Studio
Analog Interface
Sampler
Stream Manager
Audio Channel #1
Audio Channel #2
Video Channel
Display Manager
Effects Processor


Application Program
Analog I/O Devices
Host File Interface
Digital Production Studio
Analog Interface
Sampler
Stream Manager
Audio Channel #1
Audio Channel #2
Video Channel
Display Manager
Effects Processor

bandwidth of the data bus as it performs data transfers involving memory. Digital data must be moved from the storage media or local-area network into local memory on the DVI device before it can be decompressed and displayed. If several physical devices are supplying tracks of data to the "mixer," the bandwidth of the system bus may become saturated, even if the data rates of the devices are not.

Evaluation of the New Model by Cases

To ensure the new conceptual model is a superset of the AVSS "Super VCR" model, a series of cases were examined that demonstrate the VCR functions in terms of channel assignments between the components of the digital production studio.

Figure 4 shows the channel assignments for playing a file from a digital storage device. In this example, the file contains a single track of video data, plus left and right audio tracks. The play function assigns the video track to the physical video output channel where the data is converted to an analog signal for display or to record on a VCR. The decompression of the digital video is handled automatically by the system software. The audio tracks are assigned to each of the audio output channels that may be connected to an amplifier or a tape recorder.

There are times when the application must digitize from an analog source (like a videocassette recorder), and display the result on the computer screen without compressing and storing the data. This scenario is depicted in Figure 5. The analog input channels are assigned directly to their corresponding output channels.

FIGURE 5.
Monitor Function

FIGURE 6.
Record Function

FIGURE 7.
Monitor/Record Function

Figure 6 illustrates the channel assignment for recording. The physical input channels are assigned to the logical output channels, where they are either combined for storage as an audio/video file on a hard disk or sent out onto a network. As with the playback scenario, the digital compression is handled automatically by the system software.

Figure 7 shows the monitor case combined with the record case. In this scenario, the physical input channels are assigned simultaneously to the physical output channels as well as to the logical channels for storage or transmission.

The digital production studio model sufficiently describes the VCR functions provided by the current generation of DVI software and hardware. Through the use of the sampler and effects processor, image manipulation and the addition of graphics effects can easily be described. Video conferencing applications can be described as a combination of the play (Figure 4) and record (Figure 6) cases. The streams in the record group originate from a video camera and microphone with the output channels transmitted across a network. Simultaneously, the playback group originates from a network connection and its output channels are connected to the speakers and the screen display. Other combinations are also possible. For instance, two playback groups or groups of streams that originate from separate storage devices can be described.

The digital production studio model was used to drive the design of Intel's second generation of DVI system software, referred to as the Audio Video Kernel (AVK). The model proved useful in describing the features of AVK and for deriving the semantics of the application-program interface (API).

The AVK System Architecture

The increased power of the 82750PB pixel processor [9], the migration to multiple operating environments, and a desire to support windowing environments have all contributed to the design of the Audio Video Kernel. The AVK project was started with a number of additional design goals in mind. First, the design would be portable to a number of host platforms and operating environments. Second, the system needed to be able to expand as the power of the hardware increased. For example, the software system design should not limit the number of video streams that could play simultaneously. Finally, it was decided that the reliance on the host CPU should be kept to a minimum.

The AVK system architecture is illustrated in Figure 8. It is built in layers to allow for maximum portability between platforms. The bottom layer includes a collection of 82750PB microcode routines referred to collectively as the "microcode engine." One of these functions, called DoMotion, manages decompression tasks and buffers, while another, called CopyScale, copies and optionally scales the video images in real time into the display buffer. Microcode is also used to schedule the display task. DoMotion allows other microcode routines and their associated parameters to be loaded and executed dynamically to add video effects.

The next layer is called the Audio/Video Driver (AVD). This layer encapsulates knowledge of the ActionMedia hardware (Figure 9), separating it from the rest of the system. The AVD interface provides functions for accessing the local video memory (VRAM) on the device, setting display formats for the 82750DB, and loading microcode functions from VRAM into the 82750PB's on-chip instruction memory. AVD also provides an interface into the audio subsystem and the optional capture subsystem.

The third layer is called the Audio/Video Library (AVL). AVL provides most of the functionality

described in the digital production studio model by encapsulating the new digital data types needed for manipulating motion video and audio. These data types are generalized into "streams" which are collected together into "groups." A stream group is simply a collection of one or more streams that need to be controlled synchronously. Control functions such as play, pause, stop, and frame advance, operate on groups. AVL implements these control functions as well as read and write data functions for capture and display data buffers. AVL also provides control over the attributes of these data types. For example, functions for adjusting the volume of an audio stream or for adjusting the tint on a video stream are provided. This layer is largely platform-independent and therefore easily portable to other hardware environments and operating systems.

AVL also includes a set of C functions that are used internally for VRAM memory management, bitmap formatting, and for generating and managing command lists. A command list is a collection of microcode functions and their parameters. These command lists can be built in memory and then scheduled for execution by the 82750PB as a group. This is how the effects processor in our AVK design model is implemented. Each microcode function can be thought of as an effect. Placing the "effect" on a command list is analogous to "installing" the effect to the effects processor. Command lists are either associated with a bitmap (in the case of a still image) or with a video stream. If the command list is associated with a video stream, then it will be executed on all frames until the list is replaced or cancelled.

The final layer shown in Figure 8, is an environment-specific layer that has two main functions: It reads and writes data to the host file system and integrates AVK into the environment's windowing system. These functions can be efficiently implemented in an environment-specific manner, and were placed outside of the AVK architecture for portability reasons. The Media Control Interface (MCI), defined for Microsoft's Multimedia Extensions to Windows [15], and the Quicktime Movie Toolbox interface [14] are examples of environment-specific interfaces that could be implemented on top of AVK.

Objects in the Audio/Video Library

Using the conceptual model described previously, a number of objects were identified and abstracted to form the basis for the AVK interface. The interface consists of a collection of objects with their associated behavior and attributes. The AVK objects and their relationships are illustrated in Figure 10.

The analog interface subsystem (patchbay) contains two objects, the AVK *Session,* and the AVK *Device.* The calls associated with these objects start up AVK, define the communication mechanism between AVK and the application, allow query of device capabilities, and open and close the DVI device within an AVK application session.

The Stream Manager (tapedeck) is implemented as a collection of objects that control digital data streams. These objects were treated by AVSS as a tightly coupled collection (i.e., a file). AVK treats these objects as independent of the file (or files) storing the data. A *Group* is the unit of control synchronization and communication (i.e., the tape transport controls). *Group* calls include functions such as play, pause, and record. A *Group Buffer* represents the tape. A group buffer can contain multiple *Streams,* all of which play at the same rate. A *Stream* is a single track of audio or video data. Several streams may be stored in the same file.

The function of the mixer in AVK is provided by an object called a *Connector.* A connector can be thought of a pipe which accepts data flow in, optionally transforms that data, and pumps data out. The connector is a higher-level abstraction of a copy operation. Connectors allow rectangular regions, called "boxes," to be defined for the source and destination bitmaps. The size of the boxes can be modified in real time to allow resizing and relocating of images to support windowing.

The display subsystem is embodied in an object called a *View.* A view is a displayable image, and a collection of visual regions (boxes) which are mapped into windows by the application. A view is most often the destination of a connector, where the source is another displayable object such as an image or a video stream. An application can switch between multiple views.

The sampler in our current design consists only of the *Image* and *Image Buffer* objects. An image is a portion of VRAM which may be used to store still images. An image buffer is a compressed image. Images can be "compressed" into an image buffer, and image buffers can be "decompressed" into images. Image effects can be performed in-place on an image, or as part of a copy operation using a connector.

AVK Data Flow

The technique of using the screen-array bitmaps as display buffers was selected for use by AVSS for performance reasons. The old A-series pixel processor had enough processing power to decompress 256 × 240 images in real time at 30 frames per second, but not enough power to copy or scale the images as well. The new B-series 82750PB pixel processor has more than twice the processing power of the earlier A-series version. With the additional power, a more flexible buffering scheme can be used, as illustrated in the AVK data-flow diagram in Figure 11. Separating the display bitmap from the array of decompression bitmaps allows for the insertion of a copy/scale operation. This allows "windowing" effects such as relocating and resizing the video image. In order for the video to appear as a quarter-screen "window" using AVSS, all

the bitmaps need to be 512 × 480 pixels. Separating this into two areas allows the decompression bitmaps to be 256 × 240 pixels, with only one 512 × 480 pixel bitmap acting as the display buffer. Since there are a minimum of four bitmaps required in the decompression array, this scheme actually uses less memory.

There is another advantage to the AVK data flow. As the DVI hardware becomes more powerful, multiple simultaneous video windows can be implemented by allocating a compressed data buffer and decompression array for each video stream while performing the copy/scale operation to the same display bitmap. The AVSS architecture used a single array of bitmaps for both decompression and display.

Task Scheduling Using the 82750PB Pixel Processor

Playing and recording digital motion video and audio requires real-time processing of several interdependent tasks. Most of the operating systems in use in desktop computers do not provide real-time support. The solution to this dilemma for AVK takes advantage of the programmability of the 82750PB by performing the real-time scheduling of tasks using a collection of microcode routines called the "microcode engine." The control-flow diagram in Figure 12

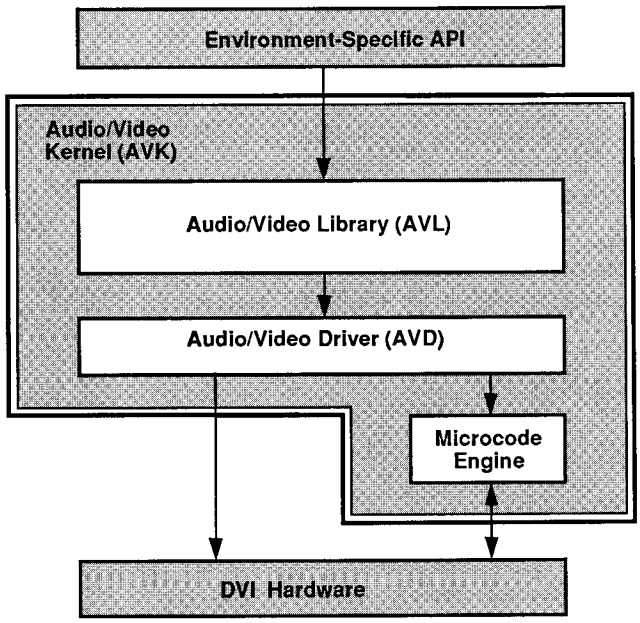


FIGURE 8.
AVK Software Architecture

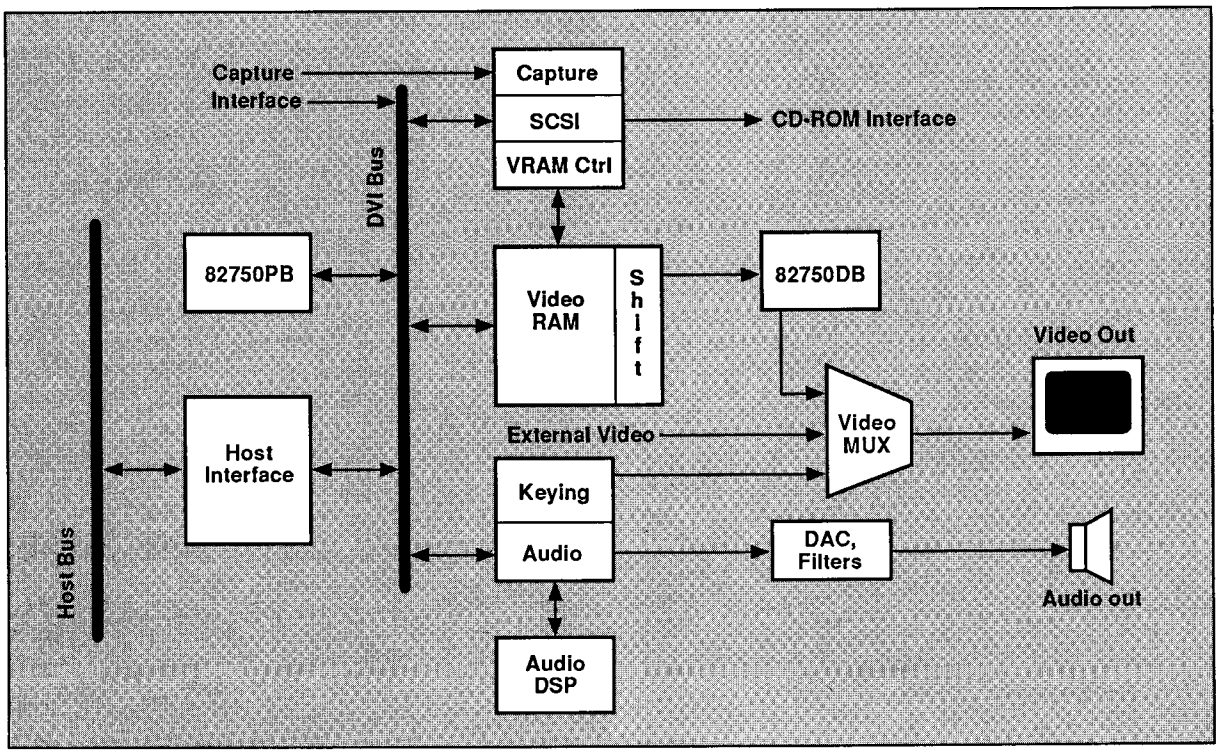
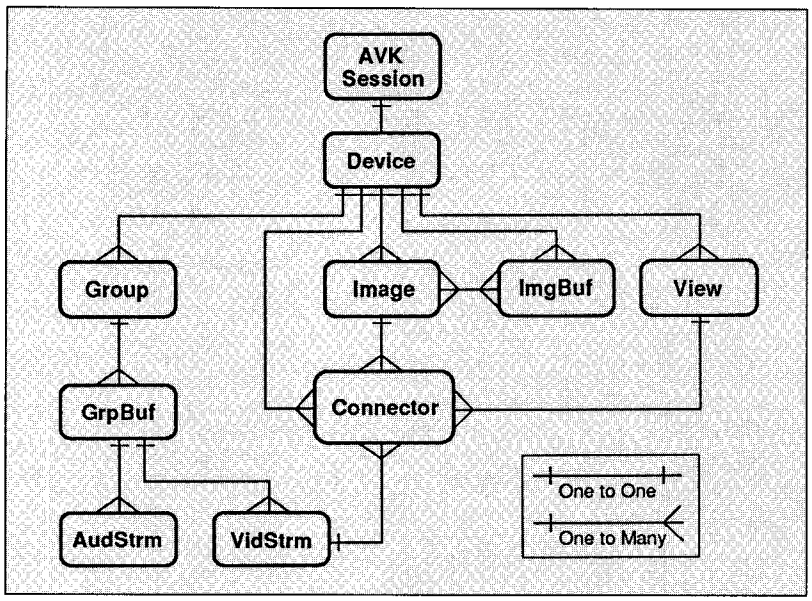
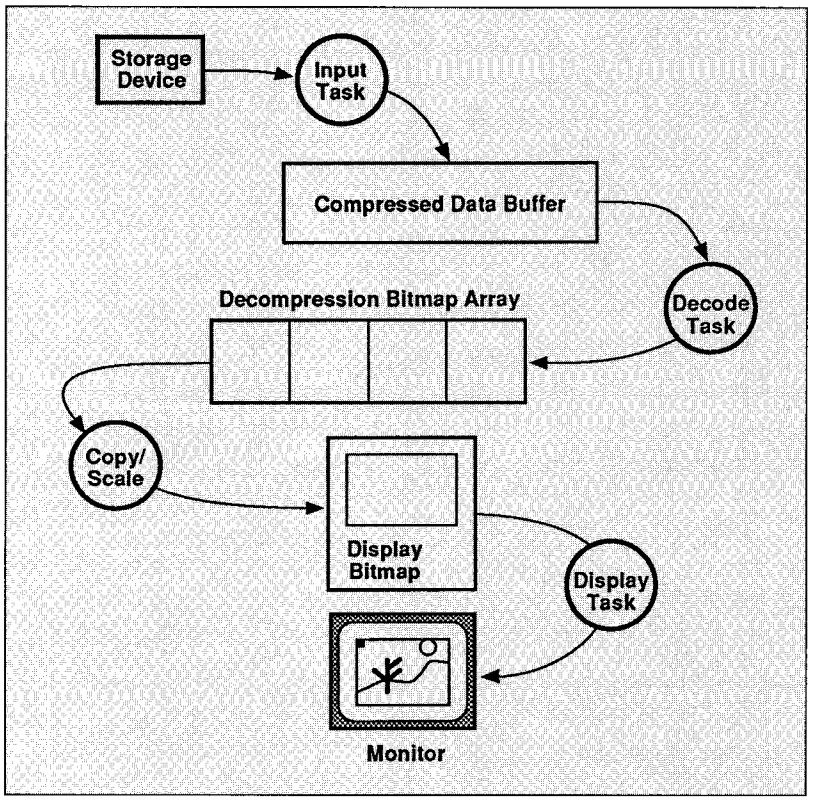


FIGURE 9.
ActionMedia Series II Hardware

shows the components of the microcode engine, and their relationships. The main components are the scheduler (DoMotion), the buffer/stream processing task, the command list processing task, and a periodic processing task.

While video streams are playing, DoMotion loops continuously between the master command list processing task, and the buffer/stream processing task. For playback, the buffer/stream processing task looks for a frame of compressed data to be available, along with a free bitmap into which the frame can be decompressed. When both these conditions are met, it calls the appropriate decompression algorithm. During video capture, the buffer/stream processing task looks for a completed bitmap to arrive from the digitizer, and for space in the compressed data buffer, and then calls the compression algorithm.





Video algorithms are executed on the 82750PB pixel processor. There are currently two types of algorithms for video, PLV and RTV. PLV stands for production-level video. It refers to video compressed off-line, on a parallel processing computer, and provides the best possible video image quality. PLV can be decompressed in real time using the 82750PB [17]. RTV, or real-time video, refers to motion video compression/decompression performed in real time using the 82750PB.

There are also algorithms for working with still images. These include the DVI algorithms for compressing/decompressing 16-bit and 9-bit (subsampled) images, as well as the Joint Photographic Experts Group (JPEG) [21] algorithm for compressing/decompressing 9-bit (subsampled) images.

Audio algorithms are executed on the audio digital signal processor. The algorithms supplied with AVK include ADPCM4E (a 4-bit adaptive differential pulse code modulation algorithm) used by the ActionMedia system, and 8-bit PCM (also used by Microsoft's Multimedia Extensions) [1].

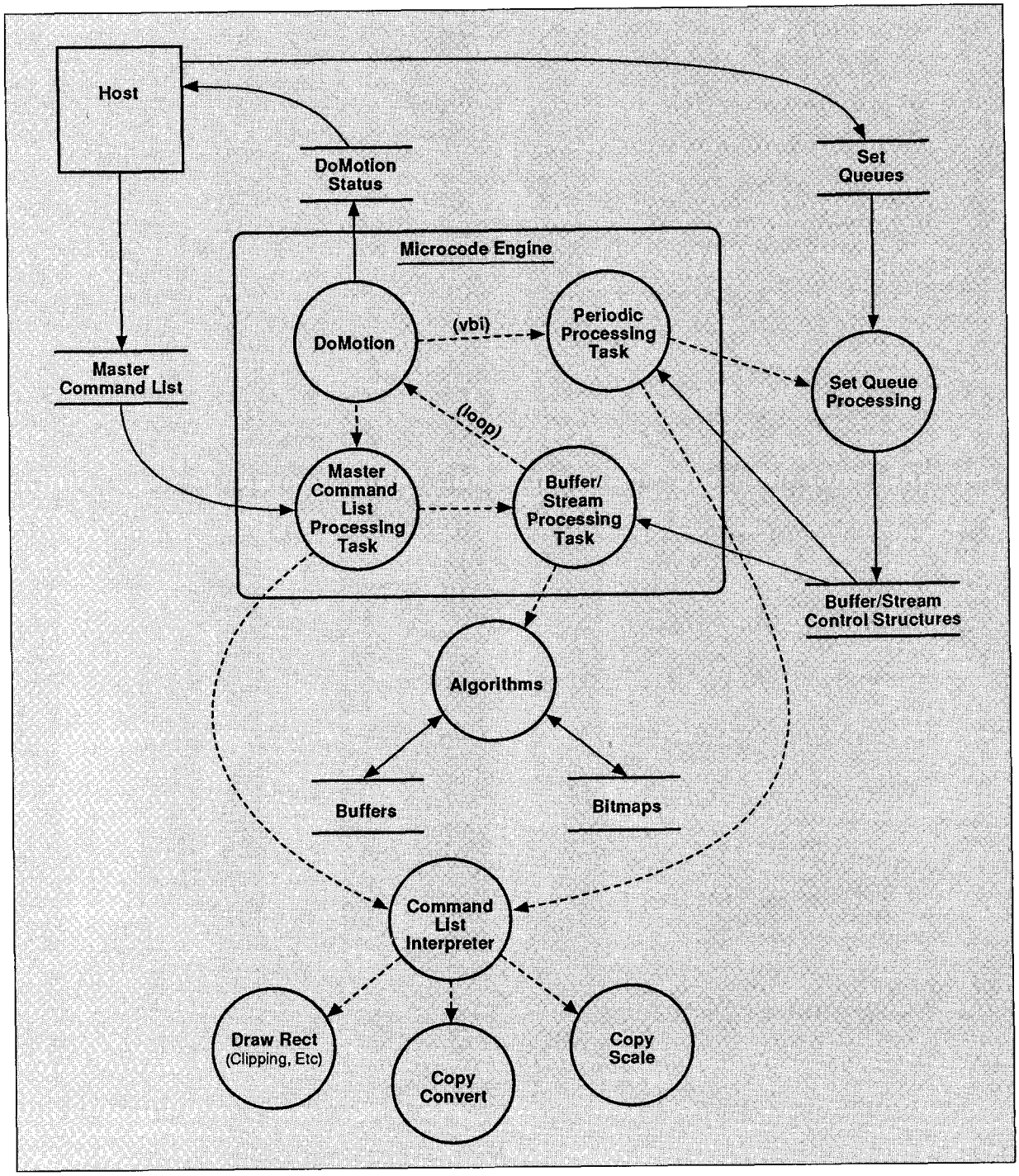
The master command list-processing task executes microcode functions requested by the host via an area of memory called the set queue. The command list interpreter uses the master command

FIGURE 10.
AVK Object Relationships

FIGURE 11.
AVK Data Flow

FIGURE 12.
AVK Microcode Engine

list as an instruction stack. The host can dynamically "program" the 82750PB by placing commands on the set queue. During periodic processing, these commands are relocated to the command list. Command lists provide the address of each microcode function and the arguments to be used during its execution. The microcode function must exist in device memory, so it can be loaded in real time into the 82750PB on-chip instruction RAM. When the command list interpreter is called from the periodic processing task, it gets its instructions from the stream command list; when it is called from the master command list-processing task, it gets the instructions from the master command list. Each command is executed in order, returning to the interpreter when complete. When the entire command list has been



executed, the command list interpreter itself returns to the calling task.

A vertical blanking interrupt triggers execution of the periodic processing task. If it is time to copy a frame, it calls the copy/scale microcode via the command list interpreter. In this case, a stream command list is used instead of the master command list.

The copy/scale microcode copies rectangular regions from one bitmap to another. It can optionally scale the bitmap, or alter its appearance by changing the tint, contrast, saturation, or brightness. If the bitmap is one of a series of video frames, the copy is performed continuously as the frames are decompressed. If the bitmap is a still image, the copy is only performed once per request. The connector object in the AVK interface is used by the application to control the copy/scale operations.

The set queue function is called from the periodic processing task, and is used to transfer commands (from the host) to the master command list or the stream command lists. This allows the host processor to communicate instructions to the 82750PB processor.

AVK uses the 82750PB as a coprocessor rather than a slave processor (as in the AVSS/RTX approach). By allowing DoMotion to perform the real-time task scheduling, the host CPU is free to perform the data delivery and other application tasks.

Conclusion

In this article the evolution of DVI system software from its original MS-DOS implementation (AVSS) to the next generation currently under development (AVK) has been described. Both the increasing power of 82750 hardware and the desire to move DVI to other platforms and operating environments has affected the AVK design goals.

In his book, *Object-Oriented Software Construction,* Meyer introduces the concept of an "operation model" as a starting point for software design, stating that "any useful system must be based on a certain interpretation of some work phenomenon [13]." The digital production studio model has proved useful in describing the features of the DVI systems, and multimedia systems in general. The author has made numerous presentations to both technical and nontechnical audiences, and found that borrowing terms and concepts from a "real-world" model facilitates understanding. It also provides a useful language for discussing features and requirements with potential users of the new system. The model selected for AVK is considered rich enough to support the addition of new features and extensions in a conceptually consistent way.

The data flow and object decomposition is sufficiently "granular" to allow the software to expand to take advantage of more powerful hardware as it becomes available. Separating the decompression bitmaps from the display bitmap is one example of this. It has been demonstrated that new (simultaneous) data streams can be added as I/O bandwidth and processor power expand. Also by allowing the host to "program" the microcode engine, the addition of custom microcode for video effects and new and improved image compression/decompression algorithms can be made without altering the underlying structure of the software.

Finally, using 82750PB microcode to perform the real-time task scheduling leaves the host CPU free to perform the file I/O and other application tasks. This method of task scheduling has a number of advantages over the other strategies examined in this article. First, it is independent of host CPU architectures and operating environments allowing the remainder of AVK (which is implemented almost entirely in C) to be more portable. Any future DVI hardware architecture can use the microcode engine to schedule real-time video tasks. Another advantage of this technique is that the 82750PB will rarely be idle because of a bottleneck on the host CPU. In the AVSS/RTX implementation, the A-series video processor must wait for instructions from the host processor and can potentially be idle, waiting for the host CPU to schedule its next task. By allowing the pixel processor to schedule its own work, the 82750PB is more efficiently utilized. In addition, the host CPU only needs to communicate with the 82750PB when it wants to change the operation of the microcode, and not on each frame (30 times per second) as in some of the other approaches discussed in this article. And finally, this approach does not require the use of another processor (as in the Fluency system). The microcode engine uses less than 7% of the cycles of the 82750PB at 25MHz.

Future directions for AVK development will include implementations by Intel and others on operating systems such as Unix/X-Windows and Apple's System 7. In addition, alternate hardware architectures which use the 82750PB/DB processors are being examined and their impact on AVK's design assessed. ◻

References

1. Amy, M., Ed. *Digital Signal Processing Applications.* Prentice Hall, N.J., 1990, 373–418.
2. Arnett, N. Computing faces the dawn of a new age. *Comput. Graph. World* (Aug. 1989), 35–37.
3. Borenstein, N.S. Multimedia electronic mail: Will the dream become a reality? *Commun. ACM 34,* 4 (Apr. 1989), 117–119.
4. Curran, L. Here's an integrated approach to multimedia. *Electronics* (Feb. 1991).
5. Dixon, D.F. Life before the chips: Simulating digital video interactive. *Commun. ACM 32,* 7 (July 1989), 824–831.
6. Everest, A. *Handbook of Multichannel Recording.* TAB Books, 1975.
7. Fox, E.A. The coming revolution in interactive digital video. *Commun. ACM 32,* 7 (July 1989), 794–801.
8. Frenkel, K.A. The next generation of interactive technologies. *Com-*

mun. ACM 32, 7 (July 1989), 872–881.
9. Harney, K., et al. The i750[R] video processor: A total multimedia solution. *Commun. ACM 34,* 4 (Apr. 1991), 65–78.
10. Homsy, G., Anderson, D.P., and Umemura, K., CMEX on a DVI platform: An implementation design. University of California Berkeley, Apr. 23, 1990.
11. Luther, A.C. *Digital Video in the PC Environment,* 2d ed. McGraw-Hill, N.Y., 1991.
12. Le Gall, D. MPEG: A video compression standard for multimedia applications. *Commun. ACM 34,* 4 (Apr. 1991), 47–58.
13. Meyer, B. *Object-Oriented Software Construction.* Prentice Hall, N.Y., 1988.
14. Petzold, C. The multimedia extensions for windows—Enhanced sound and video for the PC. *Microsoft Syst. J. 6,* 2 (Mar. 1991), 19–26.
15. Poole, L. Quicktime in motion. *MacWorld* (Sept. 1991), 154–159.
16. Ripley, G.D. Digital video interactive—A digital multimedia technology. *Commun. ACM 32,* 7 (July 1989), 811–822.
17. Sijstermans, F., and van der Meer, J. CD-I full-motion video encoding on a parallel computer. *Commun. ACM 34,* 4 (Apr. 1989), 81–91.
18. Stevens, S.M. Intelligent interactive video simulation of a code inspection. *Commun. ACM 32,* 7 (July 1989), 832–843.
19. Tinker, M. DVI parallel image compression. *Commun. ACM 32,* 7 (July 1989), 844–851.
20. Uppaluru, P. Network Computer Video. Tech. Rep. Fluent Machines, Inc. 1991.
21. Wallace, G.K. The JPEG still picture compression standard. *Commun. ACM 34,* 4 (Apr. 1991), 30–44.

CR Categories and Subject Descriptors: B.4.1 [**Input/Output and Data Communications**]: Data Communications Devices—*Processors, receivers (e.g., voice, data, image);* B.4.2 [**Input/Output and Data Communications**]: Input/Output Devices—*Image display;* C.1.3 [**Computer Systems Organization**]: Processor Architectures—*Other architecture styles;* C.3 [**Computer Systems Organization**]: Special-purpose and application-based systems—*Microprocessor/microcomputer applications; real-time systems;* I.3.1 [**Computing Methodologies**]: Computer Graphics—*Hardware architecture;* I.4.0 [**Computing Methodologies**]: Image Processing—*General;* I.4.1 [**Computing Methodologies**]: Image Processing—*Digitization*

General Terms: Design

Additional Key Words and Phrases: Digital multimedia systems, DVI

About the Author:

JAMES L. GREEN is a senior software engineer at Intel's Multimedia Products Operation in Princeton, N.J., and is one of the principal architects of the Audio Video Kernal. His current research interests include object-oriented design and programming. **Author's Present Address:** Intel Corporation, 313 Enterprise Drive, Plainsboro, NJ 08536; email: jlg@provax.intel.com

Nominees for Elections and Report of the ACM Nominating Committee

In accordance with the Constitution and Bylaws of the Association for Computing Machinery, the Nominating Committee hereby submits the following slate of nominees for the 1992 general election. The terms of office for those elected are indicated below.

The names are listed in random order. The nominees are:

For President (7/1/92–6/30/94)
- Herb Maisel, Georgetown University
- Gwen Bell, The Computer Museum

For Vice President (7/1/92–6/30/94)
- Stuart H. Zweben, Ohio State University
- John Werth, University of Texas

For Secretary (7/1/92–6/30/94)
- Frank L. Friedman, Temple University
- John H. Esbin, City of Toledo

For Members-at-Large
(Two Terms: 7/1/92–6/30/96, and One Term: 7/1/92–6/30/94)
- Ronald J. Norman, San Diego State University
- A. Joe Turner, Clemson University
- Michael Garey, AT&T Bell Laboratories
- John Hopcroft, Cornell University
- Jan Wilson, Syracuse University

The Constitution and Bylaws provide that candidates for elected offices of the Association may also be nominated by petition of one percent of the Members, who as of November 1 are eligible to vote for the nominee. Such petitions must be accompanied by a written declaration that the nominee is willing to stand for election. The number of Member signatures required for each office is as follows:

President, Vice-President, Secretary and Member-at-Large 597

The Bylaws provide that such petitions must reach the Elections Committee before January 31. Original petitions for ACM offices are to be submitted to the ACM Elections Committee, c/o Pat Ryan, ACM Headquarters, by January 31, 1992. Duplicate copies of the petitions should also be sent to the Chairman of the Elections Committee, Robert Bigelow, 10 Converse Place, Winchester, MA 01890. All candidates nominated by petition are reminded of the requirements stated in the Policy and Procedures on Nominations and Elections that a candidate for high office must meet in order to serve with distinction (copies of this document are available from the Director of Policy and Administration, ACM Headquarters).

The Nominating Committee would like to thank all those who made suggestions, gave advice, and wrote letters of support for potential candidates.

Roy Cottrell, *Chairman,* Frances Allen, Robert Haas, S. Ron Oliver, David C. Wood