Trends in hardware acceleration for digital video on personal computers

Ken Morse and Arturo A. Rodriguez

Kaleida Labs, Inc. Mountain View, CA 94043

ABSTRACT

This article reviews the current hardware acceleration techniques employed for digital video capture/compression and playback/decompression, and discusses future implementations. We discuss current capture systems that employ either dedicated silicon or a high throughput processor to provide real-time capture and compression. The advantages offered by each approach are compared. Similarly, video decompression acceleration can take the form of dedicated silicon (e.g., MPEG, JPEG implementations) or be implemented using a dedicated high throughput processor (e.g., DVI). Programmable devices allow new algorithmic implementations to be introduced without replacing the hardware system but may not offer the performance of dedicated devices. Decompression and display hardware subsystems also provide real-time functionality like colour conversion, dithering, dynamic scaling with interpolation filters and other blitter functions. Finally, we discuss the process of mixing an independent display layer with the existing systems graphics output for display, and present new designs to combine the functionality of digital video with the system graphics controller to provide a totally integrated solution.

1. INTRODUCTION

In recent years, video has become available to the desktop PC in many forms. Video cameras, electronic still cameras, scanners, VCRs, digitisers and CD-ROM drives have become extremely affordable devices, promising the acceleration of the much-heralded multimedia revolution. At the same time, personal computers have been improving their price/performance ratio at a staggering rate. The promise has been that by combining the speed and power of desktop PCs with the affordability of consumer video, the multimedia-enabled PC users would be able to process and manipulate video images with the ease with which they now crunch numbers or display stunning graphics. Video production, a skill previously reserved for film or TV professionals with expensive studio equipment, would come to the desktop, much like publishing did with the affordability of laser printers.

But despite the enormous amount of computing power available to modern high-performance PCs, a software-only solution to desktop video is not completely satisfactory. The bandwidth, storage, and processing requirements of uncompressed digitised video are such that top-of-the-line personal computers can only manage a "postage stamp" size video window without special hardware assistance; impressive though this may be for certain applications, it is not adequate for the needs of most forseeable video applications. Although Personal Computers can provide playback of 320x240 compressed video at 15 frames-per-second (fps) without special hardware assistance, they cannot provide software-only video compression nor video playback with high image quality for certain applications.

The data rate requirements even for the humble NTSC signal, the colour television standard that was developed in the late 1940s, and for the PAL equivalent that followed it, are formidable for a computer accustomed to ISA buses and hard drive data rates. Even if the data rates could be handled, the storage requirements for even short segments of digital video are enormous and would rapidly fill a one gigabyte hard disk drive.

In an attempt to mitigate these gargantuan data rate and storage requirements, there have been great advances in video compression technology in the last few years. Their achievements have been impressive indeed since some of the better video compression algorithms can attain a compression of less than 0.5 bits-per-pixel (bpp).

Despite these tremendous strides, one problem remains: there still exists a need for uncompressed video to be transferred through a video system. Whether it be between video digitiser and compressor, or decompressor and output device, the high data rates once more emerge. Unless a video system implements the complete process in one localized area, such as within one chip or across one circuit card, that uncompressed video stream must travel between circuit cards.

The second area addressed by this paper is the acceleration of digital video by adding hardware assist for tasks that are traditionally carried out in software. Figure 1 shows the control flow for hardware-assisted capture and playback of digital video.

 0-8194-1483-2/94/$6.00

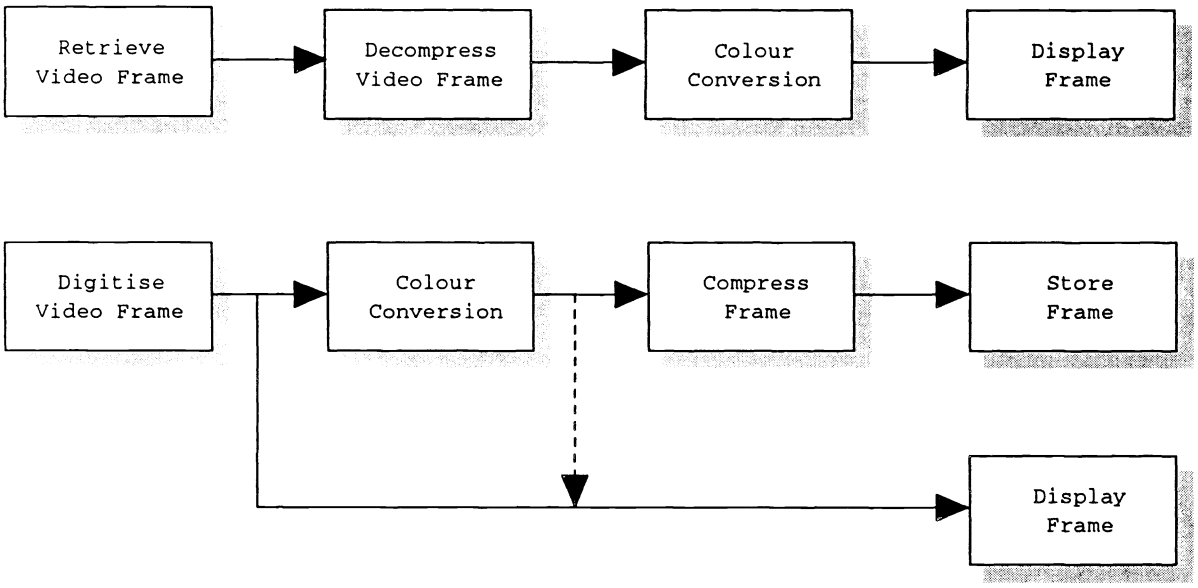

Figure 1 - Control flow of hardware-assisted digital video playback and capture

As can be seen in Figure 1, there are four potential areas for acceleration for digital video playback.

1. *Retrieve Video Frame*

   Compressed digital video data located on the storage medium (e.g. hard disk, CD-ROM, network server), must be transferred to local memory for decompression.

2. *Decompress Video Frame*

   Video frames are decoded and reconstructed from the compressed video stream buffered in memory, potentially in conjunction with previously reconstructed frames.

3. *Colour Conversion*

   Most digital video codecs encode pixel data in the YUV colour space to obtain greater compression. The YUV colour format results in a more compact representation of colour data as a result of the Human Visual System's characteristics of colour perception. Consequently, the resulting YUV video frames must be converted to the native display format of the computer (e.g., RGB colour space) before they can be displayed.

4. *Display Frame*

   The decompressed, colour converted, video frame needs to be transferred from memory into the display frame buffer after performing cropping and scaling where necessary.

A similar flow is performed to implement capture and compression of digital video.

1. *Digitize Video Frame*

   The incoming analog video frame (from a camera, VCR, etc.) is digitised, typically into RGB data. (In digital video implementations without hardware assistance, the digitised frame would typically be then transferred to system memory for the CPU to execute the remainder of the process - colour conversion and compression).

2. *Colour Conversion*

The digitised RGB video frame must be transformed into the desired colour format (typically, YUV colour space).

3. *Compress Video Frame*

   The digitised video frame is analyzed in conjunction with one or more frames, typically, a previous frame and sometimes a successive frame (e.g., an MPEG bi-directional frame) to produce a compressed data stream.

4. *Store Video Frame*

   The compressed frame is transferred from local memory to the storage medium (e.g. hard disk, network server).

5. *Display Frame*

   The digitised video frame is displayed enabling the source material to be viewed during the compression process. The video frame is copied into the display frame buffer with appropriate cropping and scaling.

   As indicated by the dashed line in Figure 1, some form of colour conversion is required if the video subsystem does not contain a display buffer. In particular, if data is captured in RGB24 or RGB16 format, and the native display format of the PC is CLUT8, the data must be transformed and dithered in advance.

Digital video implementations can be either software-only or hardware assisted. With the recent rapid increases in processor performance, software-only digital video playback has become possible. There are many areas in which software-only digital video can be accelerated by hardware subsystems and these will be discussed.

A typical personal computer-based multimedia system, illustrating the various subsystem components, is shown in Figure 2.

The following sections describe the components of the system presented in Figure 2 and introduce new approaches that increase the playback and capture performance of digital video.

2. STORAGE MEDIUM

The storage medium plays an important part of the overall system performance. With the large amounts of data required for digital video, there are two common storage media used for holding digital video: hard disk and CD-ROM. Hard disks provide the two key characteristics required by a digital video storage system; namely, low latency and high data transfer bandwidth. Access rates can be as low as 5 ms for mass produced drives and sustained data transfer rates of 1 MB/sec are not uncommon. The hard drive can process data at a higher bandwidth and the 1 MB/sec figure is typical once system software overheads are incorporated. This sustained data rate is acceptable for digital video playback but for high-quality capture and compression higher data rates may be required. In this instance, hard disk subsystems utilizing RAID (Redundant Array of Inexpensive Disks) are used to increase the data bandwidth.

However, even though hard disk prices are falling, they are still an expensive proposition for digital video storage and are not feasible for digital video delivery to a large group of users. The CD-ROM has quickly become accepted as the medium of choice for digital video delivery. However, it does have two limitations; latency and bandwidth. Early CD-ROM drive mechanisms had latencies of 500 ms and sustainable data transfer rates of 150 KiloBytes/sec. These figures are far from those achieved by hard disk but are offset by the large storage capability of the CD-ROM. Each CD-ROM can hold up to 650 MB of data which corresponds to over an hour of digital video and audio compressed using MPEG-1. The high latency can cause problems when seeking to selected areas of the disk but can be hidden by providing sufficient data buffering in system memory. Advances are continuing in CD-ROM drive mechanisms and the most popular drives available at the time of writing provide sustainable data rates of 300 KiloBytes/sec and latencies of around 250 ms.

The introduction of intelligent I/O adapters for personal computers is increasing the overall system performance of digital video. These adapters include local processors to manage the disk interfaces thus relieving the system CPU of this time-consuming task. In addition, such adapters typically include a cache memory of between 256 K and 1 MB which can be configured as a simple data buffer thus freeing system memory that would otherwise be used for this task. Data transfer from the adapter to system memory is carried out using DMA (Direct Memory Access) or bus mastering which relieves the CPU of individual data transfers.

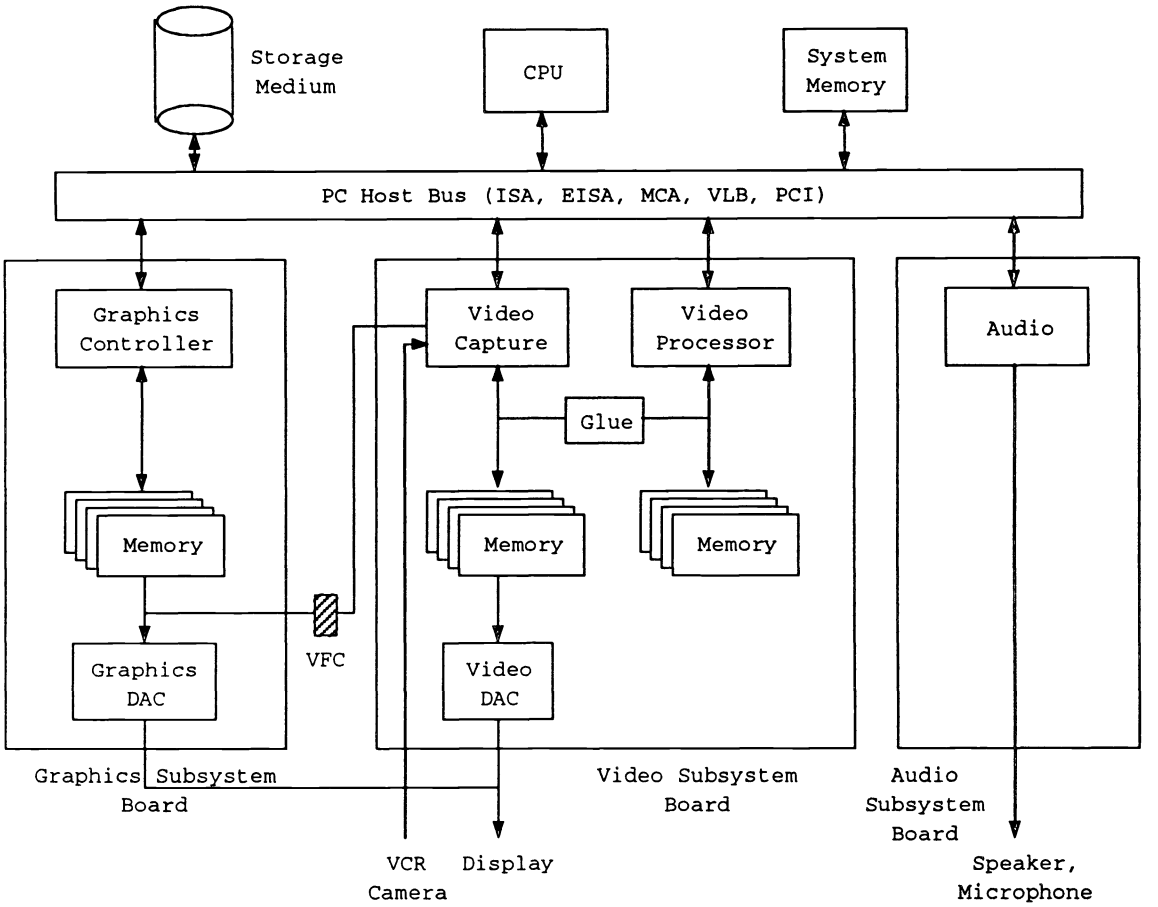

Figure 2 - Current personal computer-based multimedia system

3. HOST BUS

All data transfers from the storage medium to system memory and the graphics and video subsystems take place over the host bus. The amount of data that passes across the bus varies depending on the digital video implementation. For example, in a hardware-assisted playback environment, such as MPEG, only the compressed data is transferred across the host bus; from storage to the MPEG decoder. However, in a software-only playback environment the decompressed data must also travel over the bus from system memory to the graphics adapter for display; the exception being computers that have a local "video" bus that physically links system memory and its display buffer. Whilst the compressed data rate may be between 150 and 300 KiloBytes/sec the decompressed data rate can be around 4.5 MB/sec (320 x 240 RGB16 @ 30 fps). In such a scenario the capabilities of the host bus play a major role in overall system performance.

The range of bus interfaces on personal computers has increased over the last few years as system designers implement new architectures to address both data bandwidth and latency between devices on the bus. The most popular bus interfaces are described briefly in the following paragraphs.

ISA: The original bus introduced with the original IBM PC in 1982 has survived through today but for compatibility reasons rather than performance. It consists of a 16-bit bus design that operates at 8 MHz, independent of processor speed [1].

EISA: An extension of the ISA bus to 32-bits which offers twice the transfer rate and increased support for multiple bus masters [1].

Micro Channel: IBM's multi-master bus introduced with the PS/2 range of systems in 1987. This bus offers higher performance and supports up to 64-bit data paths [2].

VL-Bus: The VESA (Video Electronics Standards Association) Local Bus was introduced in 1992 and has quickly established itself as the standard for attaching high performance adapters within the PC. The bus is based on the Intel 486 bus (32-bit) and is typically synchronous to the processor. The majority of graphics and storage adapters are now available in VL-Bus form in addition to ISA configurations.

Two new bus architectures are being introduced which will greatly accelerate the movement of digital video data from the storage medium to system memory and system memory to the graphics and video adapters.

VL-Bus 2.0: Builds on the success of the original VL-Bus and provides support for up to 64-bit implementations running at up to 66 MHz [3]. However, the bus is still closely tied to the Intel x86 bus architecture and does not necessarily offer easy support for new processors such as the PowerPC, MIPS and Alpha.

PCI: The original work on the Peripheral Component Interconnect bus [4] was carried out by Intel but was then passed on to an independent organization. The resulting specification is processor independent enabling PCI adapters to work across a wide range of systems including IBM PCs and Apple Macintosh's. The bus specification is for 33 MHz operation and supports both 32 and 64-bit wide data buses. Multiple masters are supported enabling the storage adapter to directly transfer data to the graphics or video adapter without the intervention of the CPU. The bus is defined in 3.3 volt logic form in addition to 5 volt logic allowing a smooth migration to future low power systems.

4. CPU AND CACHE

With the migration towards intelligent subsystems for handling storage and display devices the CPU is relieved of many time consuming tasks. In a hardware-assisted digital video system this frees up the CPU to perform any other tasks associated with interactive multimedia. For a software-only digital video system the extra CPU bandwidth can be utilized to implement more elaborate algorithms or improve the frame rate of existing codecs. Video performance is also increasing with the introduction of processors such as the Pentium and PowerPC.

Software-only video playback performance can increase by making use of the cache memory available in the system. Most processors implement an on-chip cache of up to 32 KB. Off-chip cache of 256 KB is not uncommon. Algorithms that fit their inner loops and look-up tables into the cache can expect performance gains of up to 100%. However, in a multimedia system the processor is carrying out many other tasks in addition to the digital video decoding and the codec may be swapping in and out of the cache with other system software components. New processors are being introduced which implement an on-chip static RAM, as an alternative to a cache, which is under software control. Hence, algorithm authors can control the areas of code/data that are accessible at high speed and dynamically manage this 'software' cache to increase performance.

5. GRAPHICS SUBSYSTEM

The performance and capabilities of the graphics subsystem in a modern personal computer have made great leaps in the decade since the original IBM PC was introduced. The original system provided only black and white text whereas today's systems implement hardware assisted graphics in 24-bit colour at megapel resolutions.

The graphics subsystem performance has implications on the performance of digital video systems. Using the current generation of graphics adapters, the digital video may be decompressed by software-only or hardware-assisted techniques but must then be transferred to the graphics subsystem for display.

Classical graphics adapters provided access to the display frame buffer via a memory window. Hence, the entire frame buffer was not available directly and precious time was spent manipulating the memory window to the correct partition in the display frame buffer, reducing overall performance. The first attempt to increase performance was to provide graphics accelerators on the adapter. These devices can generate graphics primitives such as arcs, lines, and polygons which greatly accelerate standard graphics operations but are not useful for digital video. However, these accelerators have direct access to the frame buffer and can thus manipulate the display buffer much faster than the CPU going over the host bus. This improvement is visible in screen-to-screen bitblt (Bit Blitter) operations where a fifteen fold performance increase over classical memory window designs is observed. However, this does not help address the performance of system memory-to-screen transfers.

The only way to increase the system memory-to-screen transfer rate was to map the entire frame buffer into the CPU memory address space. However, this can cause problems since the architecture dictated by the original IBM PC limited the memory address space to 16 MB and graphics adapters that support RGB24 data at megapixel resolutions require 4 MB of memory, thus potentially limiting the system memory of some existing PC's to 12 MB. The performance of system memory-to-screen bitblts increases fivefold with the addition of the described *linear frame buffer*. The screen-to-screen and system memory-to-screen bitblts for graphics subsystems with different combinations of features: ISA or VESA Local Bus (LB), VGA with or without accelerator (ACC) or with linear frame buffer (LFB), are compared in Figure 3.

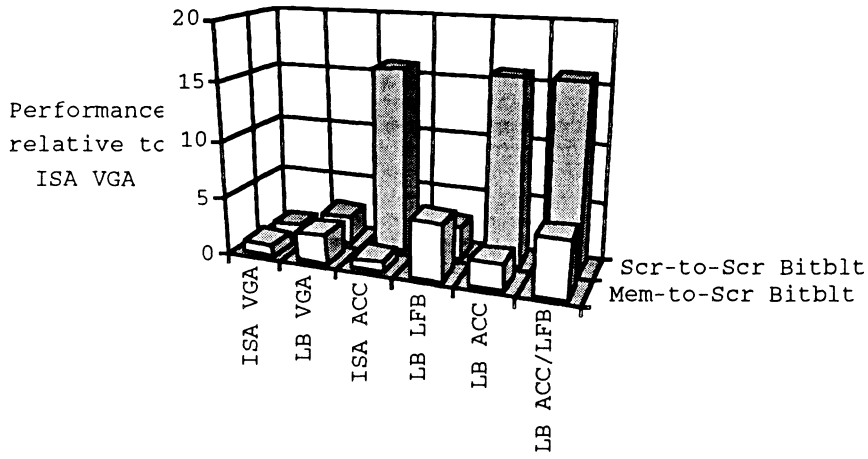

Figure 3 - BitBlt performance for various adapter configurations

The graphics subsystem can also increase performance of software-video codecs by providing the following features:

Colour conversion:

As explained previously, most digital video is compressed in the YUV colour space. On playback the codecs output YUV video frames (typically YUV 4:2:0 or YUV 4:2:2) which must be converted to RGB colour space before placement in the display frame buffer. The conversion from YUV to RGB colour space is a matrix transformation which, for performance reasons, can be implemented using a series of look-up tables in software. For instance, see Rodriguez, et al. [5]. However, even with look-up-tables, some software-only video codecs [6] can spend approximately one third of the playback execution time performing the colour conversion since it must be applied to every pixel.

Graphics adapters are now coming on the market that support real-time conversion of YUV colour space data into the RGB domain by extending the bitblt operator. Such adapters can convert to RGB24, RGB16 and CLUT-8 display formats and can increase the playback frame rate of software-only video codecs considerably.

In addition, when the display format is CLUT-8, new graphics adapters support hardware-implemented dithering to improve the visual quality of the video frames at the reduced colour resolution.

Image scaling:

In today's windowed GUI environments it is necessary to provide support for the dynamic scaling of window contents. For windows containing text and graphics objects this is relatively easy but this is not the case for digital video frames. Simple scaling can be achieved in software by replicating pixels and lines to double the image size, both in the horizontal and the vertical directions, but this results in noticeable image artifacts. Additionally, doubling the image size in both directions increases the data transfer bandwidth by a factor of four (e.g., the previously mentioned 4.5 MB/sec transfer rate into the frame buffer rises to 18 MB/sec). Artifacts also occur when scaling down by decimating (or discarding) pixels.

The solution is to implement the scaling on the graphics adapter. This limits the data bandwidth requirement over the host bus to the original frame size and enables the adapter to perform more elaborate hardware scaling. Typically, the scaling is implemented using line buffers and interpolation

filters to somewhat preserve the quality of the video frame. Using multi-tap filters allows source frames to be scaled up to full-screen whilst reducing the introduction of visible artifacts.

Image cropping: Another feature of windowed GUIs is the need to perform cropping on the contents of windows since they may be overlapped by other windows. This can be an arduous task in software but is now implemented in hardware by the latest generation of graphics adapters.

6. VIDEO SUBSYSTEM

Video is quickly making inroads into the PC and is becoming a standard data type in major operating systems. Video capture, overlay, compression and playback can now be dealt with through a standardized software interface without concern for the underlying hardware. This is a significant enhancement to the PC, and insures rapid growth in applications.

A typical video subsystem is shown in Figure 2. Such subsystems include a video processor to provide decompression and/or compression services. The video processor may take the form of a dedicated device or a programmable device depending on the application.

Dedicated devices offer high-performance due to their optimized design and are usually low-cost since they target large volume products. Their drawback is inflexibility. They invariably support a single video codec algorithm, such as MPEG, but some devices can support slight variants (e.g., devices that support both MPEG and H.261). An example of a dedicated device is the C-Cube CL450 MPEG Video Decoder [7] shown in Figure 4.

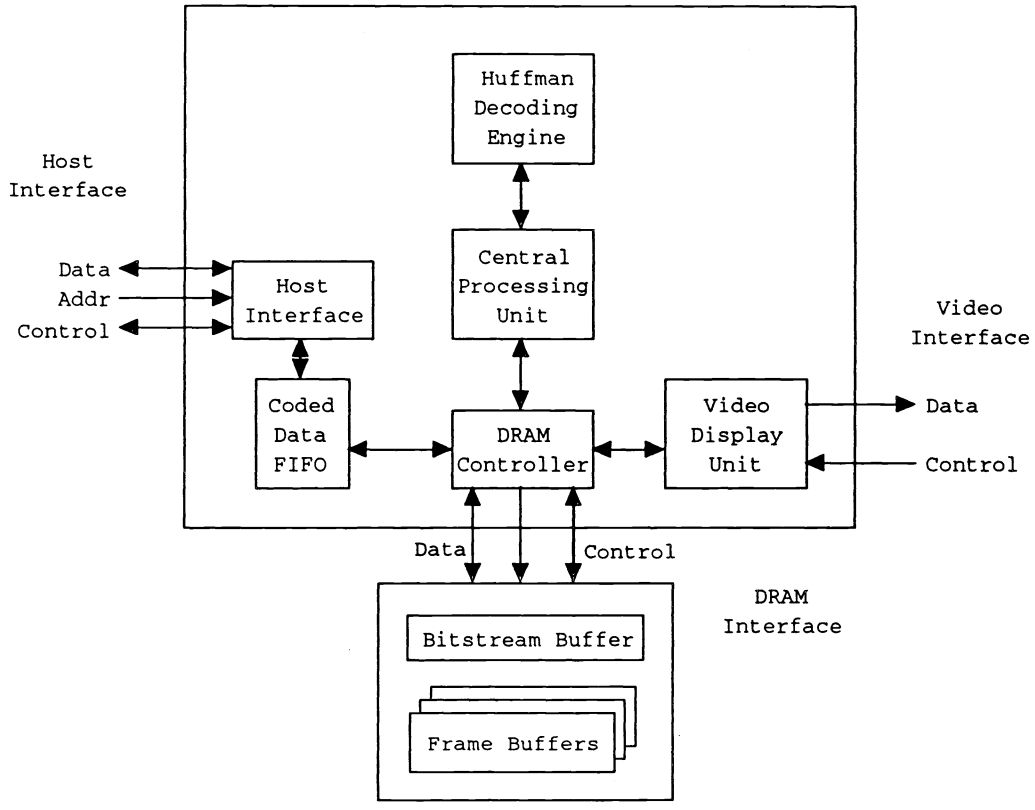

Figure 4 - C-Cube CL450 internal block diagram [7]

The CL450 decompresses SIF-resolution MPEG bit streams in real time with typical compressed data rates of 1.2 to 3 MBit/sec. It can perform real-time horizontal pixel interpolation and frame duplication to produce output formats of 704 x 240 pixels at 60 Hz or 704 x 288 at 50 Hz to support NTSC and PAL video timings, respectively [8]. The CL450 supports downloadable microcode and can be used to implement variants of the MPEG algorithm.

Programmable video processors provide flexibility by enabling downloadable algorithms. This allows algorithms to be refined over time or totally replaced as new techniques are discovered and implemented. Typically, such devices incur a performance penalty for this flexibility. It is almost impossible to achieve the performance attained with dedicated silicon by using a general purpose microcode processor.

The most prolific of the programmable video processors in the marketplace is the Intel i750 family which has been used as the backbone of the DVI (Digital Video Interactive) product line since 1988. The latest member of the family is the 82750PD [9] which is a variant of the 82750PB used on the ActionMedia II DVI adapter. Figure 5 is a block diagram of the 82750PD.

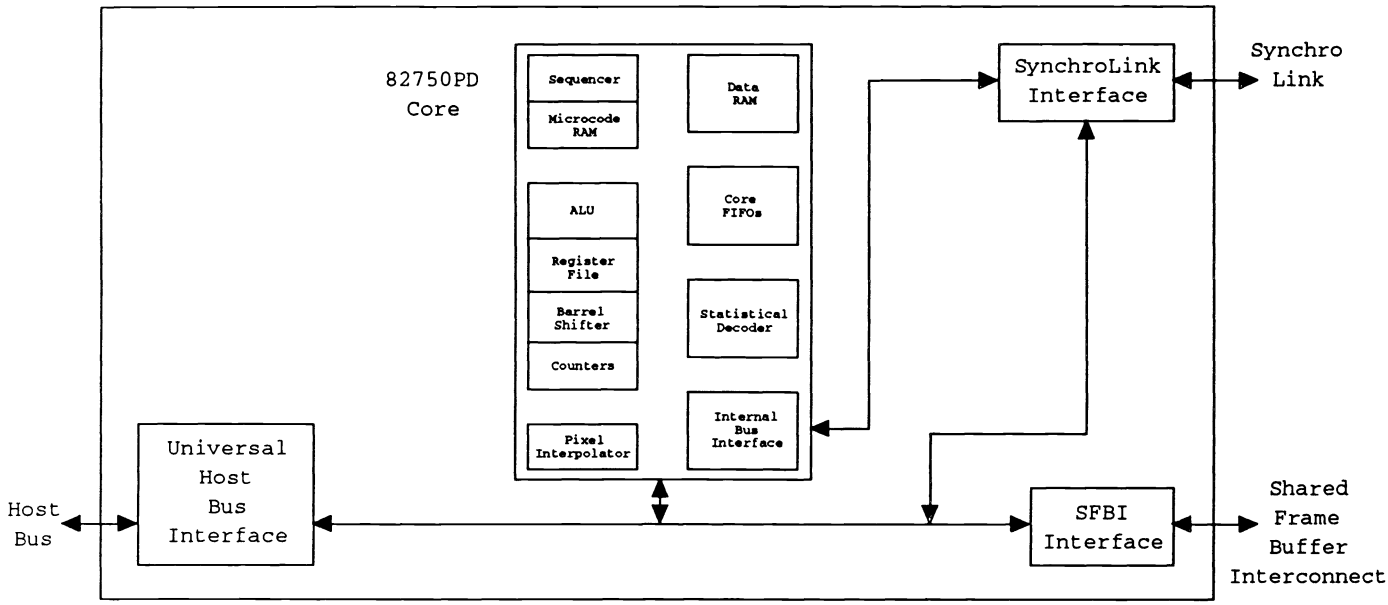

Figure 5 - 82750PD block diagram [9]

The 82750PD is an i750 Video Processor that operates in conjunction with a graphics processor and a video capture processor to bring real-time video compression and decompression to the graphics subsystem. The 82750PD has been designed to operate in a shared frame buffer architecture where it provides video compression/decompression.

The 82750PD core includes a wide instruction processor that is optimized for implementing algorithms such as compression /decompression of video frames. The core is comprised of a number of processing, storage, and input/output elements as shown in Figure 5. The various elements are connected via two 16-bit buses, called the A bus and the B bus. During each instruction execution cycle, data can be transferred from a bus source to a bus destination on both buses. The processor includes a universal host bus interface that supports ISA, PCI, VL-Bus, EISA and Micro Channel system buses. The Shared Frame Buffer Interconnect (SFBI) [10, 11] is a multi-master interface similar to the VESA Media Channel described in Section 8. The SynchroLink interface provides a local method of synchronizing graphic, video and audio events without relying on the use of host interrupts. The SynchroLink interface connects components to a time multiplexed serial bus where each device on the bus has an opportunity to transmit messages to other devices.

Many new hardware and software products are starting to appear featuring video capture, overlay, compression, decompression, genlock, etc. Unfortunately, each product has a proprietary hardware and software interface, limiting market growth. A particularly severe problem is that most video overlay products require the graphics system to be in a standard VGA mode, while the market has clearly embraced accelerators that support higher resolutions and colour depths.

Since many users purchase video and graphics hardware from different vendors, there is a need to fully decouple the video and graphics systems through independent drivers. In addition, the current 8-bit VESA Standard VGA Passthrough Connector (VSVPC) is not useful beyond 256 colours at 640 x 480 pixel screen resolution. A new feature connector has been proposed by VESA [12], which allows high bandwidth data exchange between the video and graphics systems, while allowing the two systems to be designed and built independently. This new connector allows transfer of video pixel data, either synchronously (in baseline configuration) or asynchronously (through extended modes) between a graphics and video subsystem with a throughput of up to

150 megabytes/sec. This new connector is termed the VESA Advanced Feature Connector [13] and supports both RGB and YUV data formats in colour depths from 8-bit through 32-bit. A typical system configuration utilizing the VAFC is shown in Figure 6.

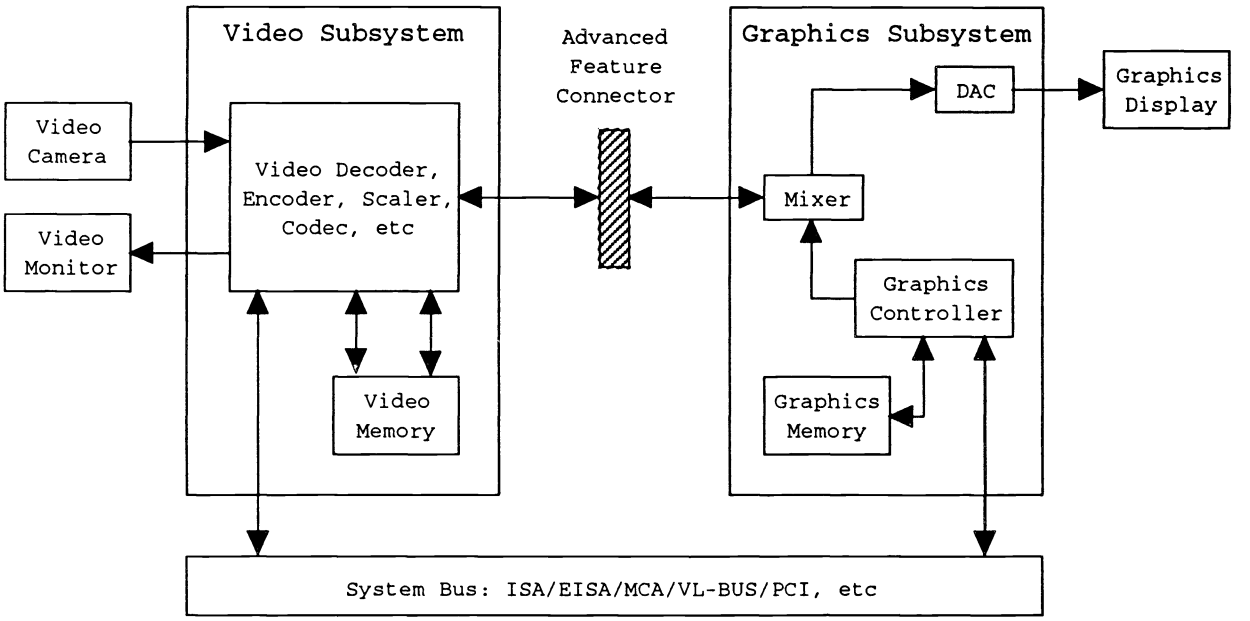

Figure 6 - VAFC implementation

In such a system the decompressed digital video data is transmitted over the VAFC and a mixer on the graphics subsystem combines the digital video with the computer generated graphics. This is typically achieved by using chroma keying to specify which regions of the graphics display should be replaced by data from the digital video source.

7. INTEGRATED GRAPHICS AND VIDEO

Until now, video systems have been implemented locally - packaging a video digitiser, frame buffer, video processor or compressor together on one card. But the most successful advances in personal computers have been due to its modular nature, where a computer user can "mix and match" accessories using standard interfaces such as the ISA bus, or the IDE or SCSI connections. What the industry is lacking is a standardized digital video interface between devices, one that can transfer many uncompressed video streams as well as compressed video and audio streams.

The VESA Media Channel (VM-Channel) [14] is a recently introduced standard for passing uncompressed video streams around a system at their native rate, regardless of the resolution or refresh rate of the system on which they are displayed. This method of attachment of video devices obviates the need for frame buffers in those devices, keeping the system cost to an absolute minimum. It has been designed with low cost in mind, allowing a range of implementations form low to high end. VM-Channel devices may be connected with a low-cost ribbon cable, or may be joined at the chip level. Finally, it has been designed for maximum flexibility, allowing manufacturers to implement any level of sophistication desired, while still maintaining compatibility.

One of the key features of the VM-Channel is its scalability in terms of channel width and complexity. The VM-Channel is designed to provide automatic dynamic bus-sizing. This allows devices to use one of three possible (and compatible) pin-outs. This is particularly relevant for attaching low-end video digitisers and codecs. Within a VM-Channel environment, real-time data such as video is transported as streams. Streams are packets of data with a header (identifier) that specifies a particular stream. The context of a stream, e.g. the pixel format and associated scaling, etc., are simply encapsulated in the stream ID which is used by a source or destination to associate the relevant context to the stream. This approach avoids the need for transferring vast amounts of attributes with each stream while at the same time offering a flexible and open way of handling or extending the system capabilities. In particular, other real-time streams such as audio and/or even timing tokens can be transported within the VM-Channel framework.

The VM-Channel is a multidrop highway, thus allowing multiple devices to be combined in a modular fashion as shown in Figure 7. For example, this allows a base system such as a graphics system supporting VM-Channel to be configured as a capture, decode only, encode only, or full encode/decode video system. Furthermore, a system based on VM-Channel can cater for multiple video stream transactions between video subsystems, or between video subsystems and the graphics subsystem.

This is important in applications such as video-conferencing as a delivery system supporting multiple video windows.

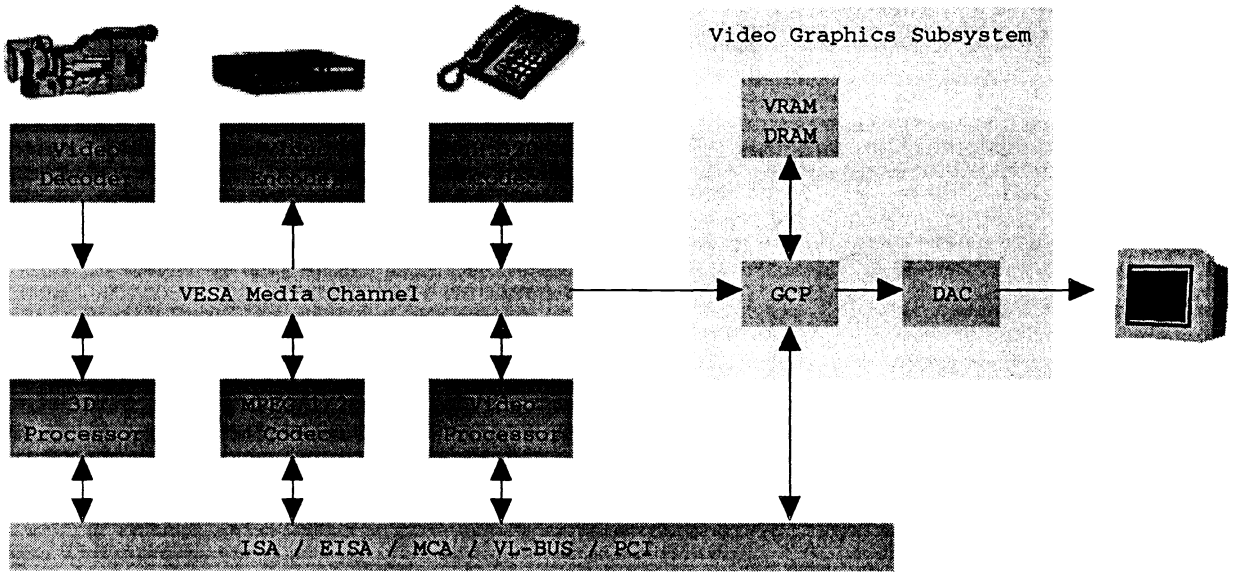

Figure 7 - VM-Channel example configuration [14]

The basic scheduling of activities of stream transactions on the VM-Channel is based on ensuring guaranteed access for a specific period of time by each agent on the channel. Each device is programmed via a register to use the channel for no more than a specific time duration before it has to hand over the channel to the next device. Furthermore, the handover mechanism allows flexibility in the selection of the next device to take mastership of the channel. This "constrained" token passing scheduling scheme provides a simple, powerful, and extendible mechanism for channel sharing while not requiring dedicated arbitration signals.

As illustrated by the example in Figure 7, the VM-Channel allows the interconnection of a wide range of video systems. The addition of a 3-D processor to the digital video subsystem opens up possibilities far beyond the "video in a window" applications available today. With a real-time 3-D processor in the system, digital video could be texture-mapped in real-time onto the surface of illuminated surfaces bringing realistic virtual reality systems within the grasp of personal computer users.

8. SYSTEM SOFTWARE SUPPORT

Finally, the software issues must be addressed for making all the hardware-acceleration techniques previously mentioned available to application writers. Otherwise, the performance-enhancing features described will not be enabled. We consider the Video for Windows architecture released by Microsoft in 1992. The initial release did not support the full range of hardware-assist capabilities available in hardware today. Codecs could be written that took advantage of specific video subsystems but there was no mechanism for codecs to access the colour conversion and dynamic scaling support provided by graphics subsystems. In addition, the performance of software video codecs was constrained by forcing the codecs to communicate via the graphic-device interface (GDI). This removed the possibility gaining performance by writing frames directly to the frame buffer as described in Section 6.

In response to these limitations, Intel proposed the Video Device Interface (VDI) which is a software interface between motion video compression/decompression drivers (codecs) and display drivers under Microsoft Windows. The VDI is designed to be an extension of the interface between the Windows GDI and the display driver, which is used by motion video codecs running under Microsoft Video for Windows. The purpose of the VDI is to improve performance in motion video codecs by cooperating with the display subsystem. This improved performance is gained primarily by adding two new features to the display subsystem:

Display support for YUV colour formats, and support for direct hardware access to the display. The interface also enables the motion video codec to use clipping and image stretching support which may be provided by the display subsystem.

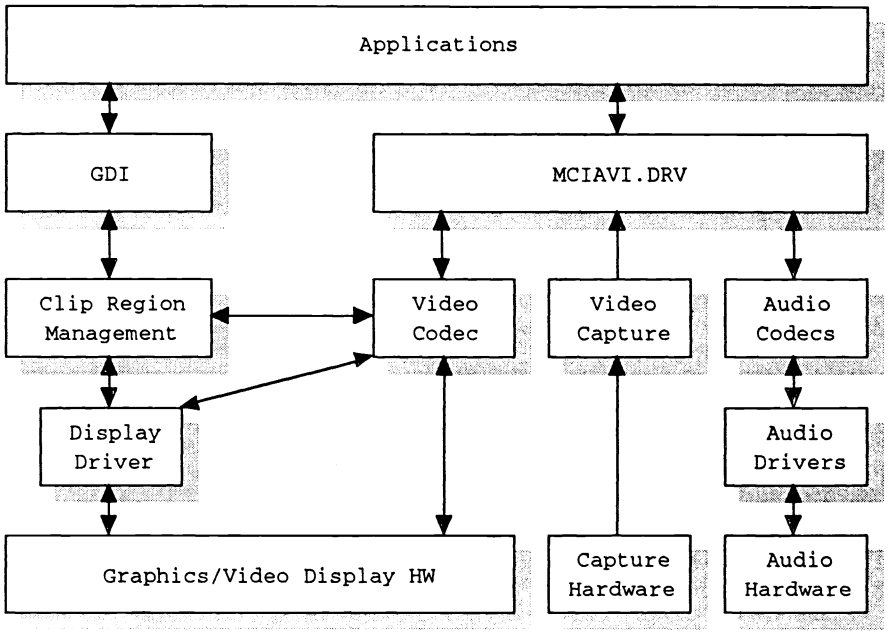

Figure 8 - Video for Windows architecture and extensions

The existing Windows architecture, shown in Figure 8 (with extensions) works well when the overhead required to display data is small compared with the overhead required to generate it. This is usually the case with computer graphics, where the same displays are used repeatedly. But in motion video, displays can change 30 times a second. The overhead associated with such display updates becomes significant, and the display architecture that was satisfactory for computer graphics is now a motion-video bottleneck.

At the frame rates required for motion video, format conversion overhead also becomes significant. The Windows GDI accepts only RGB_8 and RGB_24 colour display formats and the Media Control Interface extends this to also accept RGB_16 colour output format. In contrast, motion-video drivers work best with YUV colour-format data because it can be compressed more efficiently. Motion video decompressors must convert the YUV data to RGB format (via the CPU or via hardware devices) before returning the data for drawing.

A software motion-video driver spends its time decompressing frames, converting pixels to the display format, and copying the pixels to the display. Improvements to motion video acceleration can be made in the areas of colour space conversion and writing to the display.

Most software-only video codecs exploit frame to frame redundancy by compressing only the pixels that change, rather than the entire image. Some video codecs compress only the pixels that change [6], but they write the entire frame. This requires writing on a pixel-by-pixel basis, but doing so is impractical through the GDI because of the multiple interface layers. Efficient pixel-by-pixel writing capability requires bypassing the GDI to allow direct writes to the display.

Passing YUV-format data to a display driver cannot be done via the GDI since it does not currently accept YUV as a colour input format. Obtaining YUV colour support requires bypassing the GDI and extending the display driver to accept YUV as a supported data format.

Performance gains from the extensions described (direct writes to the display and YUV colour support) means that the window management must be performed by the motion video codec. This requires bypassing the GDI and extending the display driver to provide clipping and chroma-keying support. In initial tests indicate that YUV colour space and direct memory access support can accelerate software-only playback by as much as 100%.

Intel and Microsoft are currently designing an improved version of VDI which will become standard with future releases of Microsoft Windows thus enabling a wide range of applications to take advantage of the hardware acceleration capabilities of new digital video subsystems.

9. CONCLUSION

We have shown that there are major advances underway in combining graphics and video subsystems in personal computers. The contributory factors towards overall system performance have been presented and explained. New areas for hardware acceleration of video codecs have been described along with an explanation of the software layers that enable this acceleration to be available to a wide range of applications.

REFERENCES

[1] ISA and EISA, Theory Of Operation, Edward Solari, Annabooks, 1992

[2] IBM Publication "Personal System/2 Hardware Interface Technical Reference", first edition, May 1988

[3] VESA VL-Bus 2.0 Proposal, September 1993

[4] PCI Local Bus Specification, Revision 2.0, 30 April 1993

[5] A. A. Rodriguez et al., "Method of Converting Luminance-Color Difference Video Signal to a Three Color Component Video Signal, " United States Patent No. 5,262,847, Nov. 16, 1993.

[6] A. A. Rodriguez and K. Morse, "Feasibility of video codec algorithms for software-only playback," in *Digital Video Compression on Personal Computers: Algorithms and Techniques*, A. A. Rodriguez, Editor, Proc. SPIE 2187, (1994)

[7] CL450 MPEG Video Decoder User's Manual, C-Cube Microsystems, 1992

[8] CCIR Recommendation 601, Document 11/1041-E, 11 December 1985

[9] 82750PD Video Processor, Document 272341-2, February 1993

[10] ATI-68890 Video Capture Processor, Data sheet, 1993

[11] ATI-68800DX Advanced Graphics and Video Accelerator, Data sheet, 1993

[12] VESA Standard VS890803 "Vesa Standard VGA Pass-Through Connector"

[13] VESA Advanced Feature Connector (VAFC) Proposal, September 1993

[14] VESA Media Channel Proposal, September 1993