Hardware Accelerators for Bitonal Image Processing

Christopher J. Payson, Christopher J. Cianciolo,
Robert N. Crouse, and Catherine F. Winsor

1 Abstract

Electronic imaging systems transfer views of real-world scenes or objects into digital bits for storage, manipulation, and viewing. In the area of bitonal images, a large market exists in document management, which consists of scanning volumes of papers for storage and retrieval. However, high scan densities produce huge volumes of data, requiring compression and decompression techniques to preserve system memory and improve system throughput. These techniques, as well as general image processing algorithms, are compute-intensive and require high memory bandwidth. To address the memory issues, and to achieve interactive image display performance, Digital has designed a series of bitonal image hardware accelerators. The intent was to create interactive media view stations, with imaging applications alongside other applications. In addition to achieving memory, performance, and versatility goals, the hardware accelerators have significantly improved final image legibility.
[Accelerators paper starts here.]

Bitonal image technology, which can be viewed as the electronic version of today's microfilm method, is experiencing a high rate of growth. However, the electronic image data objects generated and manipulated in this technology are very large and require intensive processing. In a generic system, these requirements can result in poor image processing performance or reduced application performance. To address these needs, Digital has designed a series of imaging hardware accelerators for use in the document management market.

This paper provides a brief tutorial on electronic imaging. It begins with a general description of the imaging data type and compares this type to the standard text and graphics data types. It continues with a discussion of specific issues in bitonal imaging, such as image data size, network transport method, rendering speed, and end-user legibility. The paper then focuses on Digital's DECimage 1200 hardware accelerator for the VT1200 X window terminal developed by the Video, Image and Print Systems Group. It concludes with future image accelerator demands for the processing of multimedia applications and continuous-tone images.

2 Introduction to Imaging

Just as graphics technology blossomed in the 1980s, electronic imaging and its associated technologies should come of age in the 1990s. Digital imaging is already in use in many areas and new applications are being created for both commercial and scientific markets. The emergence of digital images as standard data types supported by the majority of systems (like text and graphics of today) seems assured. For a greater understanding of specific imaging applications, this section presents general imaging concepts and terms used throughout the paper.

Concepts and Terms

In its simplest form, imaging is the digital representation of real-world scenes or objects. Just as a camera transfers a view of the real world onto a chemical film, an electronic imaging system transfers the same view into digital bits for storage, manipulation, and viewing. In this paper, the term image refers to the digital bits and bytes that represent the real-world view.

The process of digitizing the view may be done through various methods, e.g., an image scanner or image camera. A scanner is the conceptual inverse of a normal printer. A printer accepts an electronic stream of bits that describe how to place the ink on the paper to create the desired picture. Conversely, optical sensors in the scanner transform light intensity values reflected from a sheet of paper and create a stream of electronic bits to describe the picture. Similar sensors in the focal plane of a camera produce the other common digitization method, the electronic image camera.

The format of a digitized image has many parameters. A pixel is the common name for a group of digitized image bits that all correspond to the same location in the image. This pixel contains information about the intensity and color of the image at one location, in a format that can be interpreted and transformed into a visible dot on a display device such as a printer or screen. The amount of information in the pixel classifies the image into one of three basic types.

- A bitonal image has only one bit in each pixel; the bit is either a one or a zero, representing one of two possible colors (usually black and white).

- A gray-scale image has multiple bits in each pixel, where each pixel represents an intensity value between one color (all zeros) and another color (all ones). Since the two colors are usually black and white, they produce a range of gray-scale values to represent the image.

- A color image has multiple components per pixel, where each component is a group of bits representing a value within a given range. Each component of a color image corresponds to a part of the color space in which it is represented. Color spaces may be thought of as different ways of representing the analog, visible range of colors in a digitized, numeric form. The most popular color spaces are television's YUV format (one gray-scale and two color components) and the bit-mapped computer display's RGB format (red, green, and blue components).

The resolution of an image is simply the density of pixels per unit distance; the most common densities are measured in dots per inch (dpi), where a pixel is called a dot. For example, a facsimile machine (which is nothing more than a scanner, printer, and phone modem in the same unit) typically scans and prints at 100 dpi, although newer models are capable of up to 400 dpi. As another example, most workstation display monitors are capable of 75- to 100-dpi resolution, and some high-end monitors achieve up to 300-dpi resolution.

To display an image at a density different from its scanned density, without altering the image's original size, requires the image to be scaled, so that the new image density matches the output media density. Scaling an image may be as simple as replicating and dropping pixels, or it may involve interpolation and other algorithms that take neighboring pixels into account. Generally, the more complex scaling algorithms require more processing power but yield higher-quality images, where quality refers to how well the original scene is represented in the resulting image.

Before an image can be displayed, its pixel values often require conversion to account for the characteristics of the display device. As a simple example, a color image cannot retain its color when output to a black-and-white video monitor or printer. In general, when a device can display fewer colors than an image contains, the image pixel values must be quantized. Simple quantizing, or thresholding, can be used to reduce the number of image colors to the number of display colors, but can result in loss of image quality. Dithering is a more sophisticated method of quantizing, which produces the illusion of true gray scale or color. Although dithering need use no more colors than simple quantizing, it results in displayed images of much higher quality.

Image compression is a transformation process used to reduce the amount of memory required to store the information that represents the image. Different compression methods are used for bitonal images than those used for gray-scale and color images. These methods are standardized to specify exactly how to compress and decompress each type of image. For bitonal images, the most common standards are the ones used in facsimile machines, i.e., Recommendations T.4 and T.6 of the Comite Consultatif Internationale de Telegraphique et Telephonique (CCITT).[1,2]

Commonly known as the Group 3 and Group 4 standards, the designations are often shortened to G3-1D, G3-2D, and G4-2D, referring to the particular standard group and to the coding method, which may be either one- or two-dimensional. For gray-scale and color images, the Joint Photographic Experts Group (JPEG) standard is now emerging as a joint effort of the International Standards Organization (ISO) and CCITT.[3] Whichever format or process is used, compression is a compute-intensive task that involves mathematically removing redundancy from the pixel data.

A typical compression method creates an encoded bit stream which cannot be displayed directly; the compressed bits must be decompressed before anything recognizable may be displayed. The term compression ratio represents the size of the original image divided by the size of the compressed form. For bitonal images using the CCITT standards, the ratio is commonly 20:1 on normal paper documents, but can vary widely with the actual content of the image. The CCITT standards are also "lossless" methods, which means that the decompressed image is guaranteed to be identical to the original image (not one bit different). In contrast, many "lossy" compression methods allow the user to vary the compression ratio such that a low ratio yields a nearly perfect image reproduction and a high ratio yields a visible degradation in image quality. This trade-off between compression and image quality is very useful because of the wide range of applications in imaging. An application need pay no more in memory space and bandwidth than necessary to meet image quality requirements.

A New Data Type and Its Features

The image data type is fundamentally different from text and graphics. When a user views characters or pictures on a display device, the source of that view is usually not important. A sheet of text from a printer may have come from either a text file where the printer's own fonts were used, a graphics file where the characters were drawn with line primitives, or an image file where the original text document was scanned into the system. In any case, the same letters and words present the user with approximately the same information; the differences are mostly in character quality and format.

In spite of their large storage space requirements, images have several advantages over graphics or text. First, consider the process of getting the information into the computer. With the imaging process, documents may be scanned automatically in a few seconds or less, compared to the time required for someone to type the information correctly (absolutely no errors) into a text file. Also, even though the software exists to convert electronic raster images into graphic primitive files, the process loses detail from the original image and is relatively slow. Next, consider the variety of information possible on a sheet of paper: a user cannot easily reproduce a diagram or a signature on a document. A scanned image preserves not only the characters, but their font, size, boldness, relative position, any pictures on the page, and even smudges or tears depending on the quality of the image scan.

The major drawback in the imaging process is increased data size, which results in storage memory and network transport problems. High scan densities and color information components create large volumes of data for each image; a bitonal image scanned at 300 dpi from an 8.5-by-11-inch sheet of paper requires over 1 megabyte of memory in its original pixel form. Therefore, compression and decompression are integral parts of any imaging system. Even in compressed form, a bitonal image of a text page requires about 50 kilobytes of storage, whereas its American standard code for information interchange (ASCII) text equivalent requires only 4 to 5 kilobytes. Similarly, a graphics file to describe a simple block diagram is much smaller than its scanned image equivalent.

Based on these advantages and limitations, several applications have emerged as perfect matches for imaging technology. Bitonal images are used in the expanding market of document management, which consists of scanning volumes of papers into images. These images are stored and indexed for later searching and viewing. Basically an electronic file cabinet, this system results in large savings in physical cabinet space, extremely fast document access, and the ability for multiple users to access the same document simultaneously. Gray-scale imaging is often used in medical applications. Electronic versions of X rays can be sent instantly to any specialist in the world for diagnosis, and the ordering of sequential computer-aided testing (CAT)-scan images into a "volume" can provide valuable three-dimensional views. The applications for color imaging are relatively new and still emerging, but some are already in use commercially, e.g., license and conference registration photographs. A further extension to still imaging is digital video, which can be considered as a stream of still images. In conjunction with audio, digital video is commonly known as multimedia, applications for which range from promotional presentations to a manufacturing assembly process tutorial.

In this paper, we focus on the static bitonal imaging method of representing real-world data inside computers. Static imaging is a simpler method of representing a broader range of information than the text and graphics media types, but it carries a greater requirement for processing power and memory space. In addition, static imaging can be viewed as one part of true multimedia, as can text, graphics, audio, video, and any other media formats. Yet static imaging does not have the system speed requirements of a motion video and audio system, which must present data at real-time rates. As long as the user can deal with static images at an interactive rate, i.e., being able to view the images in the format of choice as fast as the user can select them, then static imaging is a powerful media presentation tool. The next section presents the important issues concerning bitonal imaging in a document management environment.

3 Bitonal Imaging Issues

As previously mentioned, bitonal electronic imaging as an alternative to paper documents offers many benefits, such as reduced physical storage space, instant and simultaneous access of scanned images, and in general a more accessible media. Serious issues need to be resolved before a productive imaging operation can be implemented. The chief issues are the image data size, transport method, perceived rendering speed, and final legibility. In the following sections, we examine each issue and present solutions.

Digitized Image Data Size

The most important issue concerns image data size. Images are typically documents, drawings, or pictures that have been digitized into a computer-readable form for storage and retrieval. Depending on the dot density of the scanner, a single image can be 1 to 30 megabytes or more in size. However, storing a single image in its scanned form is not the typical usage model. Instead, a company may have tens of thousands of scanned documents. Clearly, with today's storage technologies, a company cannot afford to store such a large volume of images in that format.

A typical ASCII file representing the text on an 8.5-by-11-inch sheet of paper requires approximately 3 kilobytes of memory. If the same sheet of paper is digitized by scanning at various dot densities, the resulting data files are huge, as shown by the decompressed bitonal image sizes in Table 1. Note that Table 2 includes the size of the scanned image if scanned in gray-scale and color modes, although using these modes would not make sense on a black-and-white sheet of paper. The image sizes are included for comparison and are discussed in the section Future Image Accelerator Requirements. The data presented in Tables 1 and 2 illustrates that the size of the original ASCII file is much smaller than any of the scanned versions. The data also gives evidence that scanned images, in general, require considerable memory.

Table 1: Sample Bitonal Image Sizes

Document Type (Paper Size)	Scan Density (dpi)	Pixel Form (Decompressed)	Kilobytes of Data Typical Compressed
A size	100	114	46
(8.5 x 11 inch)	200	457	47
	300	1027	50
E size	100	1826	106
(44 x 34 inch)	200	7305	114
	300	16436	127

Table 2: Sample Gray-scale and Color Image Sizes

Document Type and Size	Kilobytes of Data in Pixel Form (Decompressed)
128 by 128 pixel, 12 bits per pixel gray-scale image	24
512 by 512 pixel, 8 bits per pixel color image	256
512 by 512 pixel, 24 bits per pixel color image	768
8.5 by 11 inch, 100 dpi, 24 bits per pixel, color image	2740

Since the typical use for bitonal images is for volume document archival, an imaging application must include a compression process to reduce memory usage. This process must transform the original scanned image file to a much smaller file without losing the content of the original scanned data.

Compression algorithms may take different paths to achieve the same result, but they share one basic process, the removal of redundant information to reduce the object size. A common compression routine searches the pixel data for groupings, or "run lengths," of black or white pixels. Each run length is assigned a code significantly shorter than the run length itself. The codes are assigned by statistics, where the most frequent run lengths are assigned the shortest codes; statistics have been amassed on a variety of document types for different scan densities and document sizes. A compression process parses through the original image file, generating another file that contains the codes representing the original image. Figure 1, a sample bitonal image compression, illustrates these compressed codes in a serial bit stream.

Several algorithms for bitonal compression are widely used today. As mentioned in the previous section, the most common for bitonal images are the CCITT standards G3-1D, G3-2D, and G4-2D, which all use the approach just described. For the one-dimensional method, the algorithm creates run lengths from all pixels on the same scan line. In the two-dimensional methods, the algorithm sometimes creates run lengths the same way, but the previous scan line is also examined. Some codes represent run lengths and even whole scan lines as "the same as the one in the previous scan line, except offset by N pixels," where N is a small integer. The two-dimensional method takes advantage of most of the redundancy in an image and returns the smallest compressed file. In addition to preserving system memory, these compression methods significantly improve network transport performance.

Network Transport Constraints

The network transport performance for an image is important, because images are most often stored on a remote system and viewed on a widespread group of display stations. For example, one group in an insurance company receives and scans claim papers to create a centralized image database, while users in another group access the documents simultaneously to process claims. For the imaging system to be productive, this image data needs to be transported quickly from one group to the other: telephone attendants answering calls must have immediate access to the data.

Scanned image documents take a long time to transport between systems, simply because they are so large. When compression techniques are used, a typical uncompressed image stored in 1 megabyte can be reduced to approximately 50 kilobytes. Since transport time is proportional to the number of packets that must be sent across the network, reducing the data size to 5 percent of its original size also reduces the transport time to

5 percent of the original time. Therefore, you can now send twenty compressed images in the same time previously spent sending one uncompressed image.

Even with compression techniques, the image files are still larger than their text file equivalents. Moreover, most network protocols limit their packet size to a maximum number of bytes, i.e., an image file larger than the maximum packet size gets divided over multiple packets. If the protocol requires an acknowledgment between packets, then the transport of a large file over a busy network becomes a lengthy operation.

The platform for our most recent accelerator is the VT1200 X window terminal, which uses the local area transport (LAT) network protocol. We soon realized that the X server packet size was limited to 16 kilobytes and the typical A-size compressed document was approximately 50 kilobytes. With this arrangement, each image transport would have required four large data packets and four acknowledgment packets. Working with the X Window Terminal Base System Software Group, we were able to raise the packet size limit to 64 kilobytes. The base system group also implemented a delayed acknowledgment scheme, which eliminates the need for the client to wait for an acknowledgment packet before sending the next data packet. Table 3 shows compressed image data taken during the DECimage 1200 development cycle. Notice that the network transport times for Digital document interchange format (DDIF) decrease sharply after the packet changes.

Table 3: DDIF Image File Read Time and File Transport Performance

Image Size (kilobytes)	Disk Read Time (Milliseconds) MicroVAX	VAX8800	VAX6440	Network Transport Time (milliseconds) After Packet Change	Before Packet Change
19	1223	480	281	325	960
41	1534	655	332	614	1792
99	2351	1035	598	1351	3928
157	3288	1380	716	2283	6430

Perceived Rendering Speed

Because the image scanning and compression operations occur only once, they are not as performance-critical as the decompression and rendering for display operations, which are done many times. Decompression and rendering are part of the system's display response time, which is a critical factor in a system designed for high-volume applications that access thousands of images daily. This time is measured from the instant the user presses the key to select an image to view, to the moment the image is displayed completely on the screen. The display response time is a function of the disk read time, network transport time, and display station render time.

Although network transport time and disk file read time have a direct effect on the response time, accelerator developers rarely have any control over them. The disk access time data from the DECimage project analysis shown in Table 3 demonstrates that the disk file read time is a significant portion of the overall response time. Thus, the display station render time is the only area of the display response time which can be clearly influenced and is, therefore, the main focus of our image accelerators. The local processing that must occur at the display station is not a trivial task; an image must be decompressed, scaled, and clipped to fit the user's current window size, and optionally rotated.

The decompression procedure inverts the compression process; both are computationally complex. Input to the procedure is compressed data, and output is the original scan line pixel data, which can be written to a display device. Scaling the data to fit the current window or fill a region of interest is not trivial either: a huge input data stream must be processed (the decompressed, original file), and a moderate output data stream must be created (the viewable image to be displayed). While simple pixel replicate and drop algorithms may be used to scale the data, a more sophisticated scaling algorithm has been shown to greatly enhance the output image quality.

In addition to scaling and clipping, the orthogonal rotation of images (in 90-degree increments) is a useful function on a display station. Some documents may have words running in one direction while pictures are oriented another way, or the user may wish to view a portrait-mode image in landscape mode. In either case, orthogonal rotation can help the user understand the information; i.e., the increased time to rotate the view is warranted.

When an image is scanned, particularly with a hand-held scanner, the paper is never perfectly aligned. Thus, the image often requires a rotation of 1 to 10 degrees to make the view appear straight in the image file. However, multiple users want the information from the document as quickly as possible, and should not have to rotate the image by a few degrees to make it perfectly straight on the screen. Therefore, this minimal rotation should be done after the initial scanning process; i.e., only once, prior to indexing the material into the database, and not by every user in a distributed environment. Because any form of rotation is compute-intensive, allowing the user to perform minimal rotations at a high-volume view station would reduce the application's perceived rendering speed and add little value to the station's function.

Final Legibility

While the primary issue facing imaging applications is data size, image viewing issues must also be addressed. In short, an effective bitonal imaging display system must be responsive to overall image display performance and the resulting quality of the image displayed. To enhance our products, we optimized the display performance parameters as best we could, given that some parameters are not under our control. Improvements to monitor resolution and scanner densities continue to increase the legibility of images. An affordable image system should increase the image legibility by rendering a bitonal image into a gray-scale image using standard image processing techniques. We discuss the method used in our accelerators, i.e., an intelligent scale operation in the hardware pipeline, in the next section.

4 Hardware Accelerator Design

As explained in the previous section, transforming documents into a stream of electronic bits is not the demanding part of a bitonal imaging process for document management. Also, scanners and dedicated image data-entry stations abound in the marketplace already. Instead, the challenge lies in: (1) managing the image data size to control memory costs and reduce network slowdown; (2) increasing the image rendering speed, i.e., decompress the image, scale it, and clip it to fit the window size with optional rotation; and (3) increasing the quality of the displayed images. This section describes the way our strategy influenced the design of DECimage products. We also discuss the chips used for decompression and scaling, and how Digital's existing client-server protocols support these imaging hardware accelerators.

General Design Strategy

The number of applications using bitonal image data continues to increase. In general, these applications attempt to offer low cost while achieving an interactive level of performance, defined as no more than 1 second from point of request to complete image display. Ultimately, software may provide this functionality without hardware acceleration, but today's software cannot. Moreover, the parameters of image systems are not static; scan densities, overall image size, and the number of images per database will all increase. These increases will provide the most incentive for hardware assist at the low end of the X window terminals market, because software alone cannot perform the amount of processing that users will expect for their investment.

The User Model Although a single model cannot suit every application, imaging is centered on certain functions. Therefore, a user model built on these functions would be very useful in mapping individual steps to the hardware: hardware versus software performance, the function's frequency of use, and the cost of implementation.

The general user model for bitonal imaging systems is relatively simple. A small market exists for image entry stations, in which documents are scanned, edited, and indexed into a database. While a high throughput rate is important at these stations, a general-purpose image accelerator is not the solution- dedicated entry stations already exist in the market. Instead, we designed a general-purpose platform, or versatile media view station, to be used for imaging applications alongside other applications. The user model for this larger market is a set of operations for viewing and manipulating images already entered into a database. The most common operations in this model are decompression, scaling, clipping, orthogonal rotation, and region-of-interest zooming.

Display Performance and Quality Optimization The main thrust of the DECimage accelerator is to achieve interactive performance for the operations defined in the user model. A secondary goal is to bring added value to the system by increasing the quality of the displayed image compared to the quality of the scanned image. A side effect of maximizing performance in hardware is that the main system processor has work off-loaded from it, freeing it for other tasks.

The general design of the accelerator uses a pipelined approach. Since maximum performance is desired and a large amount of data must be processed by the accelerator board, multiple passes through the board are not feasible. Similarly, the targeted low cost does not allow a whole image buffer on the board. With one exception (rotation), all board processing should be done in one pipeline, with the system processor simply feeding the input end of the pipe and draining the output end. Because of the large amount of data to be read from the board and displayed on the screen, the processor should only have to move that data, not do any further operations on it. To this end, any logic required to format the pixels for the display bitmap should be included in the pipeline.

Cost Reduction through Less Expensive System Components The net cost of a bitonal imaging system is influenced by the capability of the assist hardware. The capability of the hardware implies flexibility in the choice of other system hardware. In this regard, the most significant impact on cost occurs in the memory and the display. A system that makes use of fast decompression and scaling hardware can quickly display compressed images from memory. This means either more images can be maintained in the same memory, or the system can operate with less memory than it would without the assist hardware; less memory means lower cost.

A more dramatic effect on system cost is in the display. Imaging systems generally need higher-density displays than nonimaging systems, but the cost of a 150-dpi display is approximately twice the cost of a 100-dpi display of the same dimensions. However, we found that we could increase legibility, i.e., expand a bitonal image to a gray-scale representation, by using an intelligent scale operation in the hardware pipeline. For example, a bitonal image rendered to a 100-dpi display using the intelligent scale process gives the perceived legibility of the same image rendered to a 150-dpi display with a simple scaling method. That is, by adding the intelligent scale, a 100-dpi display can be used where previously only a 150-dpi display would be adequate.

Cost Reduction through Integration Presently, as in the DEC-image 1200, hardware-assisted image manipulation exists as a board-level option. Higher levels of integration with the base platform will provide lower overall cost for an imaging system. The most straightforward method of integration is to relocate the hardware from the present option to the main system processor board; successive steps of integration would consolidate mapped hardware to fewer total devices. The most cost-effective integration will be the inclusion of the mapped hardware in the processor in a way similar to a floating-point unit (FPU). Just as graphics acceleration is now being included in system processor design, images will eventually achieve the status of a required data type and thus be supported in the base system processor.

Product Definition-What Does the User Want?

The previously described strategy was used in the design of the image accelerator board for the DECimage 1200 system. The product requirements called for a low-cost, high-performance document image view station. These requirements evolved from the belief that most users currently investigating imaging systems are interested in applications and hardware that will enable them to quickly and simultaneously view document images and run their existing nonimaging applications. These users are involved with commercial and business applications, rather than scientific applications. The DECimage 1200 system was planned for the management of insurance claims processing, hospital patient medical records, bank records, and manufacturing documents. As previously stated, the imaging functions required for these view-oriented applications are high-speed decompression, scaling, rotation, zooming, and clipping.

General Product Design

In defining the image capable system, the key points in the product requirements list were

- High-performance image display
- Low cost
- Bitonal images only (not gray-scale or color)
- View-only functions

The need for high-performance display influenced the project team to design the hardware accelerator board to handle image decompression, scaling, and rotation. Previous performance testing on a 3-VUP (VAX-11/780 units of performance) CPU had yielded image software display times from 5 to 19 seconds. These images were compressed according to the CCITT Group 4 standard (300 dpi, 8.5-by-11 inches), and ranged from 20 to 100 kilobytes in size. In addition, the software display times were highly dependent on the image data content. The more complex image files, which had lower compression ratios, took significantly longer to decompress, scale, and display than the simpler image files. For example, an A-size, 300-dpi, CCITT Group 4 compressed image with a compression ratio of 10:1 took approximately 18 seconds to display, while another with a ratio of 33:1 took approximately 7 seconds.

The other three requirements led to decisions about the specific design of the image accelerator board. The need for low cost meant designing an option for an existing low-cost platform, which led us to Digital's VT1200 X window terminal. This requirement also led to our support of the proposed X Image Extension (XIE) protocol.[4] The XIE protocol extends the X11 core protocol to enable the transfer of compressed images across the wire and to enable interactive image rendition and display at the server. In the X windowing client-server environment, image applications and compressed image files exist on the client host machine, as depicted in Figure 2. In addition, the XIE protocol standardizes the interface-to-image functions in the X windowing environment and enables the development of a common application that can be used on any XIE-capable station. The client application issues commands to the X server display subsystem and the XIE specialized image subsystem. When a user selects an image to view, the compressed image file is transported from the client-side storage device to the X server memory.

Because the proposed accelerator would handle only bitonal images, we could specialize our board to decompress only the standard CCITT Group 3 and Group 4 bitonal compression algorithms. This specialization allowed the use of a Digital application-specific integrated circuit (ASIC) decompression chip. Finally, the view-only requirement limited the scope and complexity of the design by eliminating the need for extra hardware to handle the compression of images after they have been scanned and edited.

Specific Product Design

The decisions described in the previous section led to our design of an image accelerator board that supports: CCITT Group 3 and Group 4 image decompression using an ASIC decompression chip; integer scaling using an ASIC scaling chip; orthogonal rotation; and image display. Figure 3 shows a general block diagram of the board and how it fits into Digital's VT1200 system architecture. The accelerator board is attached to the system address/data bus, and its registers, data input port, and data output port are mapped into the CPU's I/O space. The accelerator board is accessed by reading and writing specific addresses like any other system memory space. Note that the image accelerator logic is separate from the video terminal logic. Decompressed images are read from the image board and written to the base system video memory for display.

The main operation consists of the following steps: compressed image data is read from system memory and written to the ASIC decompression buffer by the processor; the data is then decompressed, scaled by the ASIC scaling chip, packed into words, and written to the output buffer. Figure 4 shows a detailed block diagram of the image accelerator board logic. The scaling chip outputs pixels of data (1 bit per pixel in this case) which are packed into words using shift registers. As soon as a word of data is available, the scaling chip output halts. Control signals generated in programmable array logic (PAL) write the packed word into the output buffer and tell the scaling chip to begin outputting pixels again. When the output buffer is full, the processor reads the rendered image data from the buffer. If rotation is required, the processor writes the data to the rotation matrix; otherwise, the data is clipped and written to the bit map. The image driver software, after setting up the board, alternates between checking whether the input buffer is empty and whether the output buffer is full.

The rotation circuit handles 90- and 270-degree rotation, whereas 180-degree rotation is handled in the data packing shift registers by changing the shift direction. The circuit rotates an 8-by-8-bit block of data at a time. The first byte of eight consecutive scan lines is written into eight individual byte-wide registers. The most significant bit (MSB) of each of these registers is connected to the byte-wide rotation output port latch. A processor read of this port triggers a simultaneous shift in all of the rotation data registers so that the next bit of each register is now latched at the rotation output port for the next read. Figure 5 diagrams the rotation circuitry just described.

To achieve the best performance, we pipelined the functional blocks in the hardware. The scaling engine does not need to wait for the entire image to be decompressed before it can begin scaling; instead, scaling begins as soon as the first byte of data is output from the decompressor. Thus different pieces of the image file are being decompressed, scaled, and rotated simultaneously. The hardware pipeline also eliminates the need to store the fully uncompressed image (approximately 1 megabyte of data for A-size 300-dpi images) in memory. The compressed image is written from system memory to the accelerator board and a decompressed, scaled, and clipped image is read from the board. Because of the speed of the hardware, the software can redisplay an image with different scaling, clipping, or rotation parameters; it merely changes the hardware setup for the different parameters and sends the compressed image file back through the accelerator board pipeline.

ASIC Design Description

The ASIC design consists of a decompressor chip, which decodes the compressed image data to pixel image data, and a scaling chip, which converts the image from the input size to the desired display size.

Decompressor Chip The decompressor chip acts as a CCITT binary image decoder. The chip contains three distinct stages, which are pipelined for the most efficient data processing. Double buffering of compressed input data is implemented to enable simultaneous input data loading and image decoding to occur. Compressed data is loaded into the input buffer by the processor through a 16- or 32-bit port. Handshaking controls the transfer of decompressed data from the decompressor's 8-bit-wide output bus to the scaling chip.

The first stage of the decompressor chip converts CCITT-standard Huffman codes, which are of variable-length, to 8-bit, fixed-length codes (FLCs).[5] A sequential tree follower circuit is implemented to handle this conversion. Every Huffman code corresponds to a unique path through the tree, which ends at a leaf indicating the FLC. The 8-bit FLC is sent to a first-in, first-out (FIFO) buffer, which holds the data for the second stage.

The second stage of the chip generates a 16-bit, run-length value from the FLC. The lower 15 bits of the word contain the number of consecutive white or black pixels (called the run length). The upper bit of the word contains the run-length color code (0 for a white run and 1 for a black run). An FLC is read from the FIFO buffer and decoded into one of eight routine types. Each routine is made up of several states that control the color code toggling, run-length adder, and accumulator circuits. At the end of each routine, a new word containing the run-length and color information is written into a FIFO buffer for the final stage.

The final stage of the decompressor chip converts the run-length and color information to black or white pixels. This stage outputs these pixels in 16-bit chunks when the scaling chip sends a signal indicating a readiness to accept more data.

Scaling Chip The primary purpose of the scaling chip is to input high-resolution document images (300 dpi) and scale them for display on a medium-density monitor (100 dpi). The chip offers independent scaling in the horizontal and vertical directions. The scaling design implemented in the chip is a patented algorithm that maps the input image space to the output image space. General M-to-N pixel scaling is provided where M and N are integers between 1 and 127, with the delta between them less than 65. M represents the number of pixels in and N represents the number of pixels out (in the approximated scale factor).

Given an image input size and a desired display size, we must find the M and N scale factors that best approximate the desired scale factor, within the range limits of M and N as previously stated. Thus an input width of 3300 and a desired output width of 550 are represented by an M of 6 and an N of 1. The approximated M and N values are loaded into the chip scale registers for downscaling or upscaling.

The chip scaling logic uses the scale register values to increment the input pointer position and generate output pixels. A latched increment decision term is updated every clock cycle, based on the previous term and the scale register values. When scaling down (where fewer pixels are output than are input), the logic increments the input pointer position every clock cycle, but only outputs a pixel when the increment decision term is greater than or equal to zero. Figure 6a illustrates how this algorithm maps input pixels to output pixels for a sample reduction. When scaling up (where every input pixel represents at least one output pixel), the logic outputs a pixel every clock cycle, but only increments the input pointer position when the increment decision term is greater than or equal to zero. Figure 6b illustrates how this algorithm maps input pixels to output pixels for a sample magnification. For both cases, the value of the pixel (black or white) being output is the value of the input pixel pointed at during that clock cycle. In this description, simply substitute rows for pixels to represent the vertical scaling process.

Software Support for the Hardware

Software support is needed to enhance the functions of the hardware accelerator in our image view station. As mentioned in the section General Product Design, the XIE protocol extends the X11 core protocol to enable the transfer of compressed images across the wire and to enable image rendition and display at the server using the hardware accelerator board. Like the X11 protocol, the XIE protocol consists of a client-side library called XIElib, which provides client applications access to image routines, and a server-side piece, which executes the client requests. The XIE server implements support at two levels: device-independent and device-dependent. The device-dependent level supports the functions that benefit from optimization for a particular platform, or functions that are implemented in hardware accelerators. The device-independent level enables quick porting of functionality from platform to platform. Figure 7 illustrates the X/XIE client-server architecture.

The client-side XIElib offers the minimum functions necessary for image rendition and display. The toolkit level offers higher-level routines that assist with windows application development. An example of a routine at this level might be ImageDisplay, which displays an image in a previously created window. ImageDisplay parameters might include x and y scaling values, the rotation angle, and region-of-interest coordinates. Whether programming with the XIE protocol at the library or toolkit level, applications developers benefit from the platform interoperability of the standard interface. Image accelerator hardware and optimized device-dependent XIE code changes the application's image display performance, but an application developed using the XIE protocol can run on any XIE-capable server.

Accelerator Performance Results

With the DECimage 1200 X terminal, we have achieved interactive performance rates, reduced memory usage, and increased final image legibility. We achieved these rates by transporting compressed files instead of huge pixel files and by implementing specialized image processing hardware. The DECimage 1200 can read, transport, decompress, scale, and display an 8.5-by-11-inch bitonal document in 1 to 2 seconds. Successive displays, i.e., rotating, region-of-interest zooming, panning around the image, all occur in less than 1 second, which is essentially as fast as the user can ask for the displays. This speed is possible because the image already resides in compressed form in the server memory. Thus, the image does not have to be read from the disk or transported across the network.

5 Future Image Accelerator Requirements

Hardware accelerators will continue to be required for bitonal imaging until software can provide the same functionality at the same performance level. This section discusses the more complex image schemes that are used for gray-scale imaging and multimedia applications. In contrast to bitonal imaging, these applications will require the use of hardware accelerators well into the future.

Other applications will require richer user interfaces utilizing continuous-tone images, video, and audio. All of these new data types are generally data-intensive, and compression or decompression of any one of them is a significant processing burden. Handling them in combination indicates that the need for specialized hardware assistance will persist for the foreseeable future.

Continuous-tone Images

Bitonal images are either black or white at each point, but some applications require smoothly shaded or colored images. These images are typically referred to as continuous-tone images, a term that denotes either color or gray-scale, e.g., photographs, X rays, and still video. The representation and required processing of this image format is significantly different from that of bitonal images.

Continuous-tone images are represented by multiple bits per pixel. This format allows a greater range of values for each pixel, which yields greater accuracy in the representation of the original object. Additionally, each pixel can consist of multiple components, as in the case of color. The number of bits used to represent a continuous-tone image is chosen according to the nature of the image.

For example, medical X rays require a high degree of accuracy. Consequently, 12 bits are generally regarded as the minimum acceptable for the rendering of this class of image. Color images typically require 8 bits per pixel for each component (YUV or RGB format) for a total of 24 bits per pixel. Table 2 shows the relative size of samples of each image. The need to express these images in a compressed format is obvious from the storage space requirements and the current storage media limits.

The compression of continuous-tone images can be accomplished in several ways. However, most imaging applications are not closed systems; inevitably, each system needs to manipulate images that are not of its own making. For this reason we adopted the JPEG standard, which specifies an algorithm for the compression of gray-scale and color images. Specifically, the JPEG compression method is based on the two-dimensional (2D) discrete cosine transform (DCT). The DCT decomposes an 8-by-8 rectangle of pixels into its 64 2D spatial-frequency components. The sum of these 64 2D sinusoids exactly reconstructs the 8-by-8 rectangle. However, the rectangle is approximated-and compression is achieved-by discarding most of the 64 components. Typically adjacent pixel values vary slowly, thus there is little energy in most of the discarded high-frequency components.

The edges of objects generally contribute to the high-frequency components of an image, whereas the low-frequency components are made up of intensities that vary more gradually. The more frequency components included in the approximation, the more accurate the approximation becomes. Table 4 shows some sample JPEG image compression ratios.[6]

Table 4: Typical Compression Parameters for JPEG

Compression Ratio	Compression Method	Rendered Image Integrity
2:1	Lossless	Highest quality - no data loss
12:1	Lossy	Excellent quality - indistinguishable from the original
32:1	Lossy	Good quality - satisfactory for most applications
100:1	Lossy	Low quality - recognizable

The most popular part of the JPEG standard, the "baseline" method, was defined to be easily mapped into software, firmware, or hardware. Straightforward DCT algorithms can be efficiently implemented in firmware for programmable DSP chips, due to their pipelined architecture. The first systems to embody the standard did so using DSPs, because any change to either the evolving standard or a standard extension could be easily introduced to the firmware. The fastest implementations, of course, are achieved by special-purpose hardware accelerators.

The JPEG implementation does not require hardware, i.e., the algorithm can be performed completely in software. The case for hardware assist is made in performance. Table 5 describes the reduced instruction set computer (RISC) processor performance, in millions of operations per second (mops), needed to provide the specified operation at a motion video rate of 30 frames per second.[7] However, generic RISC processors of those speeds are not available today. Therefore, dedicated, custom very large-scale integration (VLSI) devices (such as the CL550-10 from C-Cube Microsystems) must be used to perform the operations.[8]

Even if the motion video rate is not required, the ASIC devices offer the simplest hardware solution.

Table 5: Processing Requirements for Imaging Functions

Imaging Functions	Processor Operations per Pixel*					Processor Op-era-tions at 30 fps (mops)
	Read	Write	ALU+	Multiply	Total	
Pixel move	.25	.25	0	0	.5	15
Point operation	2	1	1	0	4	120
3 by 3 convolve	9	1	8	9	27	810
8 by 8 DCT	24	1	14	16	65	1950
8 by 8 block matching	128	1	191	0	320	9600

*RISC processor, 1M pixels, 30 frames per second (fps), 8 bits.
+ALU = arithmetic logic unit

Live Video and Video Compression

Video captures the natural progression of events in an environment, and is therefore a natural and efficient way to communicate. Consider, for example, the assembly of a set of components. One way to express the assembly process is to show a series of photographs of the assembly at successive steps of completion. As an alternative, video can show the actual assembly process from start to finish. Subtle details of the process such as part rotations and movements can be clearly conveyed, with the added dimension of time.

Obviously, information expressed in video form can be valuable; however, significant problems arise in adapting video for use in computer systems. First, the huge data size of video applications can strain the system's storage capability. Video can be characterized as a stream of continuous-tone images. Each of these images consists of pixel values with individual components making up each pixel. For video to have full effectiveness, the still images must be presented at video rates. In many cases the rate to faithfully reproduce motion is 30 frames per second, which means that one minute of uncompressed video (512-by-480 pixels at 24 bits per pixel) would consume over 1 gigabyte of storage. In addition to storage demands, large volumes of data cause bandwidth problems. Presenting 30 frames per second to the video output with the above parameters would require a transfer rate of more than 22 megabytes per second from the storage device to the video output. Thus, reducing the amount of data used to represent the video stream would alleviate both storage and bandwidth concerns.

The starting point for the compression of video is with still images and, as previously mentioned, the JPEG algorithm can be used to compress still continuous-tone images. Because video can be represented as a sequence of still images, the algorithm could be applied to each still. This procedure would produce a sequence of compressed video frames, each frame independent of the other frames in the sequence.

The evolving Motion Picture Experts Group (MPEG) standard takes advantage of frame-to-frame similarities in a video sequence, thereby enabling more efficient compression than the application of the JPEG algorithm alone.[9] In most situations, video sequences contain high degrees of similarity between adjacent frames. The compression of video can be increased by encoding a frame using only the differences from the previous frame. The majority of scenes can be greatly compressed; however, scene transitions, lighting changes, or conditions of extreme motion need to be compressed as independent frames.

The need for hardware assist in this area is compelling. Table 5 shows that to sustain a JPEG decompression at 30 frames per second would require a 1950-mops processor. The same result can be obtained using the CL550-10 JPEG Image Compression Processor.[9] Although this device does not make use of interframe similarities to increase compression efficiency, a device implementing the MPEG standard would exploit these similarities. Table 5 shows that motion compensation, to be supported at 30 frames per second, requires a 9600-mops processor.

Audio and Audio Compression

Video is usually accompanied by audio. The audio can be reproduced as it was recorded (with the video), or it can be mixed with the video from a separate source (such as a compact disc (CD) player). The audio data is defined by application requirements. If the application allows lower quality, the audio can be sampled at lower rates with fewer bits per sample, such as telephony rates, which are sampled at 8 kilohertz and 8 bits per sample. For applications requiring high-quality (CD) audio, samples are usually taken at 44 kilohertz and 16 bits per sample.

Integrating audio data into an application creates special problems. The major characteristic that differentiates audio from the other data formats presented here is its continuous nature. Audio must flow uninterrupted for it to convey any meaning. In video systems, the flow of frames may slow down under heavy system loading. The user may never notice it, or may not be annoyed by it. Audio, however, cannot slow or stop. For this reason, large buffers are used to allow for load variations that may affect audio reproduction.

A more subtle problem in creating applications using audio is in synchronization. Audio data is usually included to add another dimension of information to the application (such as speech). Without a method of synchronizing the video and audio, one data stream will drift out of phase with the other. One way to include synchronization is to use time stamps on the audio and video. This is particularly useful because standard time codes are used in most production machines.

The compression of audio data is not as efficient as that of the other data formats. Since a statistical approach to coding audio is highly dependent on the type of input (i.e., voice, musical instrument), another method is required for generalized inputs. Differential pulse code modulation (DPCM) is often used to encode audio data. DPCM codes only the difference between adjacent sample values. Since the difference in value between samples is usually less than the magnitude of the sample, modest compression can be achieved (4:1). The limitation using this technique is in the coding of high-frequency data.

Hardware assist for the audio data format will probably come in the form of hardware to perform functions other than compression. For instance, DSP algorithms can perform equalization, noise reduction, and special effects.

Multimedia

As the term implies, multimedia may integrate all of the previously mentioned image formats. The word "may" is important in this context. This area has been mainly technology-driven, due to such factors as lack of standards, developing I/O devices, insufficient system bandwidth, differing data formats, and a vast amount of software integration.

It is currently a topic of debate whether typical users will require the ability to create, as opposed to only access, multimedia source material. However, for discussion purposes, multimedia platforms can be classified into two categories: authoring and user. Authoring refers to creation of multimedia source material and requires different capabilities than user platforms. In the creation of a multimedia application, data from many different devices may need to be digitized and cross-referenced. As the data is incorporated, it is compressed and stored. Authors require the capability to edit and mix video and audio passages to get the desired result. Moreover, the video and audio may originate from different devices and may even be in different formats.

As defined above, "user systems" do not require all of the functions that authoring systems need: only decompression is required in a typical user system. Most existing user systems require an analog video source (videodisk), which is purchased as part of the application. The device control is performed by the application, i.e., when a user selects a passage to be replayed, the application sends commands to the videodisk. Figure 8 depicts an authoring system and a user system, along with suggested I/O capability.

Next-generation multimedia platforms will make full use of digital video and audio. This implies that systems will be able to receive and transmit multimedia applications and data over networks. This interactive capability will improve the efficiency of many mundane applications and devices. For example, electronic mail can be extended with video and audio annotations, or meetings can be transformed into video teleconferencing. The adoption of completely digital data for multimedia also implies that the platform I/O will change. Some user systems will not require analog device interfaces or control: the user will load the application over the network or from an optical disk.

Each of the image formats described in this section has different characteristics, and each will be presented in the embodiment of multimedia. Given the size, processing requirements (compression and decompression), and real-time demands of applications, hardware assist will be a necessity.

6 Summary

Imaging is a unique data type with special system requirements. To achieve interactive rates of bitonal image display performance today, hardware accelerators are needed; that has been the primary focus of this paper. In the future, a general-purpose processor should be able to handle the imaging process at the necessary speed, and beyond that, the processor should be affordable in a low-cost bitonal imaging system. However, the bitonal document processing market will not wait; it is in a high state of growth and requires that products like accelerators be developed for at least a few years.

Continuous-tone documents and multimedia applications will place an even heavier processing load on an imaging system. These areas will require accelerators for several years. As imaging applications, including bitonal, expand to cover more markets, the quality enhancements and performance benchmarks met by accelerators today will set customer expectations. Consequently, our future imaging products must be designed to meet these expectations.

7 Acknowledgements

The authors wish to express thanks to the X Window Terminal Hardware and Software Design Groups for their support in developing the DECimage 1200 option. The two major ASICs used in the design were developed for previous projects, and those two design teams are also offered our thanks. Special thanks to Frank Glazer and Tim Hellman for their insightful research on the image rendering process.

8 References and Note

1. Standardization of Group 3 Facsimile Apparatus for Document Transmission, CCITT Recommendations, Volume VI-Fascicle VII.3, Recommendation T.4 (1980).

2. Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus, CCITT Recommendations, Volume VII-Fascicle VII.3, Recommendation T.6 (1984).

3. Digital Compression and Coding of Continuous-Tone Still Images, Part I, Requirements and Guidelines, ISO/IEC JTC1 Draft International Standard 10918-1 (November 1991).

4. J. Mauro, X Image Extension Concepts, Version 2.4 (Cambridge: MIT X Consortium, June 1988).

5. D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," Proceedings IRE, vol. 40 (1962):1098-1101.

6. G. K. Wallace, "The JPEG Still Picture Compression Algorithm," Communications of the ACM, vol. 34, no. 4 (April 1991):30-44.

7. Table 5 is adapted from Y. Kim, "Image Computing Requirements for the 1990's: From Multimedia to Medicine," Proceedings of Electronic Imaging West (April 1991).

8. CL550 JPEG Image Compression Processor, Preliminary Data Book (San Jose, CA: C-Cube Microsystems Inc., November 1990).

9. Coding of Moving Pictures and Associated Audio, Committee Draft of Standard ISO 11172: ISO/MPEG 90/176 (December 1990).

Christopher J. Payson Chris Payson joined Digital as a hardware design engineer in 1989 after five co-op terms. He is currently working on XIE software and image hardware accelerators. Chris previously worked on performance testing, diagnostics, logic design, and demonstration software, all associated with imaging. He is coapplicant for a patent related to an image clipping algorithm and hardware logic. Chris received a B.S.C.E. from

Rochester Institute of Technology with highest honors and is currently pursuing an M.S.C.E. from Northeastern University.

Christopher J. Cianciolo As a hardware design engineer in the Video, Image and Print Systems Group, Chris Cianciolo is currently working on the design for their latest imaging product. Chris joined Digital in 1985 after participating in a co-op session in the Power Supply Engineering Group. He also participated in co-op sessions for Charles Stark Draper Laboratory, Inc. on a fiber-optic missile guidance system project. He received his B.S.E.E. from Northeastern University in 1988 and is currently pursuing an M.S.E.E., also from Northeastern.

Robert N. Crouse Senior engineer Bob Crouse is a member of the Video, Image and Print Systems Group. He is currently working on the advanced development of new imaging technology for X window terminals. Bob was project engineer for the development of a bitonal imaging accelerator for a low-end VAXstation workstation. As a member of the Electronic Storage Development Group, he designed a double-bit error detection and correction circuit for a VAX mainframe. Bob received his B.S.E.E. from Northeastern University and holds one patent.

Catherine F. Winsor As a senior engineer in the Video, Image and Print Systems Group, Cathy Winsor has worked on image accelerators. As the project leader for the DECimage 1200 hardware and the image utility library software, Cathy was involved in the planning and development of an image-capable VT1200. She is currently leading the project to support imaging on the next generation of Digital's X terminals. The project includes an image accelerator board and XIE software. Cathy received an A.B. in engineering sciences from Dartmouth College and a B.S.E.E. from the Thayer School of Engineering.

DECimage, Digital, VAX, and VT1200 are trademarks of Digital Equipment Corporation.

===

==