An Integrated Graphics Accelerator for a Low-Cost Multimedia Workstation

Designing with a system focus and extracting as much performance and functionality as possible from available technology results in a highly integrated graphics chip that consumes very little board area and power and is 50% faster and five times less expensive than its predecessor.

by Paul Martin

The graphics subsystem of the Model 712 workstation is a high-performance, low-cost solution that sits directly on the system bus of the Model 712 and consists of the graphics chip, a video RAM-based frame buffer, and a few support chips (see Fig. 1). The project goals closely reflect those of the overall HP 9000 Model 712 program. In priority order these goals were:

- Very low manufacturing cost
- Leadership graphics performance at entry cost levels
- Architectural compatibility
- Compelling new functionality.

Achieving these goals required a major step in the evolution of HP entry-level graphics workstation hardware. Two philosophies helped the team responsible for the graphics chip achieve these goals. The first guiding philosophy was to design with a system-level focus. We examined all required functionality to decide whether it was best to implement it in the graphics subsystem, the host processor, or some combination of the two.

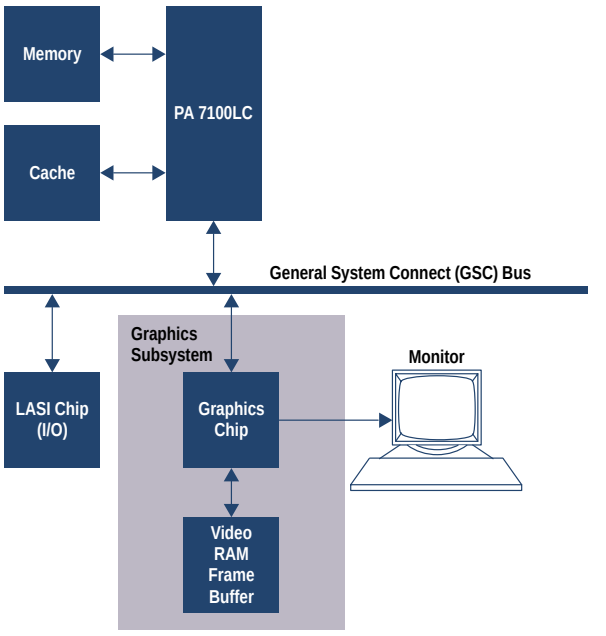


Fig. 1. A block diagram of the essential components that make up the HP 9000 Model 712 workstation.

The second philosophy was to extract as much performance and functionality as possible from readily available technology. We avoided leading-edge technology because of the cost implications. We did make an attempt to use all the features and performance available in mature technologies such as video RAMs (VRAMs) and HP's CMOS26B IC process.

This article describes the features and functionality of the HP 9000 Model 712 graphics subsystem. The considerations that went into accomplishing the goals mentioned above are also described.

Architectural Compatibility

The CRX window accelerator card† introduced by HP in 1991 marked the beginning of a standardized graphics hardware architecture for window system acceleration.[1] This architecture was chosen for its simplicity of implementation and for the clean model it presents to the software driver developers. One of our fundamental design decisions was to accelerate key primitives only—a RISC approach. Many earlier controllers chose to accelerate a large gamut of graphical operations such as ellipses, arithmetic pixel operations, and so on. Graphics subsystems designed with these controllers were typically expensive and exhibited only moderate window system performance. In the CRX and subsequent accelerators, including the Model 712's graphics chip, we decided to accelerate a carefully chosen smaller set of primitives, which are described in the following sections.

Block Transfer. Writing pixels from system memory to the frame buffer or reading from the frame buffer to system memory is a block transfer (see Fig. 2). Writes are used to transfer image data to the frame buffer. Reads are used primarily to save portions of the screen temporarily obscured by pop-up menus (see Fig. 2b).

† A window accelerator is the hardware that provides the images seen on the workstation monitor. In particular, an accelerator is geared toward speeding up environments such as the X Window System. The window accelerator enables the fast movement of windows on the screen, scrolling of text, painting of window borders and backgrounds, and so on.

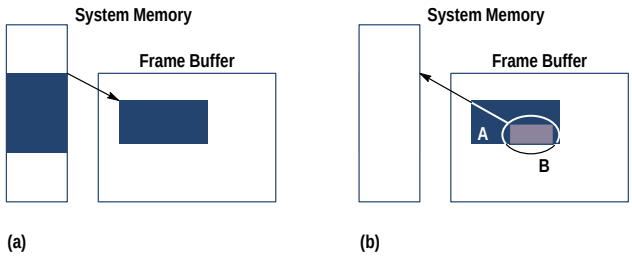


Fig. 2. (a) Block transfer write. (b) Block transfer read. Window B obscures window A. The obscured area is stored in system memory for restoration when the area of window A is exposed.

Block Move. A block move involves transferring pixels from one rectangular area in the frame buffer to another (possibly overlapping) area in the frame buffer (Fig. 3). This is very useful for moving windows on the screen and scrolling lines of text within a window. The block move in the graphics chip supports Boolean operations on the data being moved, such as highlighting text by complementing colors.

Vectors. The ability to draw vectors (line segments) very quickly is a requirement of design applications such as schematic capture and mechanical design (Fig. 4). Thus, the graphics chip has a high-performance vector generator that creates X Window System-compliant line segments.

Fast Text. Characters are accelerated by the graphics chip because of their pervasive use in window systems and the large potential for performance improvement over software-only solutions. A character is defined as a rectangular array of pixels that contains only two colors called foreground and background colors. Because there are only two choices, a single bit is sufficient to specify the color of each pixel in a character. This improves performance by reducing the amount of data that is transmitted from the processor to the graphics chip. For example, the h_p character in Fig. 5 requires only 8 bytes of data versus 48 bytes if this optimization had not been made.

Rectangular Area Fill. This primitive is widely used by window systems to generate window borders, menu buttons, and so on (Fig. 6). It is also important for applications such as printed circuit board layout and IC physical design. Rectangular areas can be patterned using two colors or contain only a single color. Hardware acceleration again gives a large speedup over software-only solutions.

Cursor. Until the late 1980s when hardware cursors started appearing in video ICs, screen cursors were typically generated using software routines. Hardware support is a good trade-off because the circuitry is relatively simple, and a system without hardware acceleration can spend a significant portion of its time updating the cursor. A 64-by-64-pixel, two-color cursor is supported directly in the graphics chip.

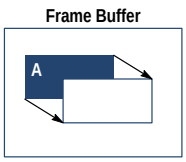


Fig. 3. Block move. Rectangular area A is moved to a new, possibly overlapping location.

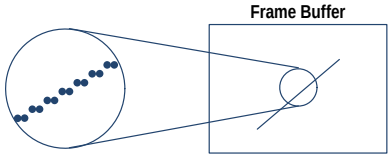


Fig. 4. Vector primitive. A vector is drawn by turning on successive pixels using the Bresenham algorithm.

More complex functionality such as wide lines, circles and ellipses, and 3D primitives are not accelerated directly by the graphics chip because the application performance improvement was determined to be too low for the cost of implementation. These functions can be efficiently implemented in software. This is an example of the system-level design trade-offs mentioned above.

An important aspect of this standardized architecture is software leverage. It is estimated that several software engineering years were saved on the graphics chip because the architecture is virtually identical to that of the CRX graphics subsystem. The savings in software engineering time was applied to tuning and adding new functionality instead of rewriting drivers.

Graphics Chip Operation

To get a better understanding of the operation of the graphics chip let's follow a graphics primitive through the block diagram shown in Fig. 7. A vector is a good example because it involves all of the blocks in the chip. Assume we have a vector that starts at x,y coordinates 0,0, is 8 pixels long, and has a slope of 1/2.

First, several parameters are calculated to set up the vector in the graphics chip. This is done by graphics software (e.g., the X Window System) running on the PA 7100LC CPU. The high-level specification of a vector is:

- Starting x,y coordinate
- Ending x,y coordinate.

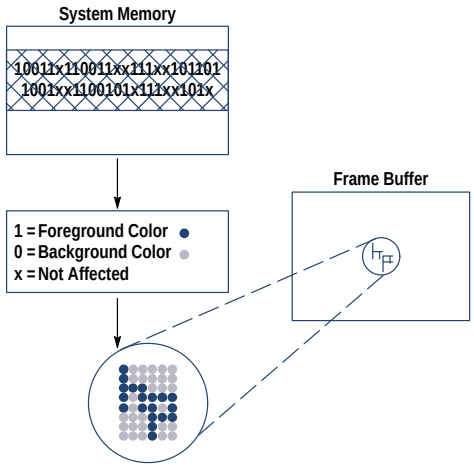


Fig. 5. Fast text primitive. A character is a rectangular array containing two colors, foreground and background colors. Only a single bit is needed to specify each color.

This data is transferred across the GSC bus, through the GSC interface, and into a set of registers in the macro function unit. If these registers are already in use by the macro function unit the data is placed in a 32-word-deep FIFO buffer that the unit can access when it becomes free. This increases efficiency by allowing overlap between the software and hardware processes. The macro function unit's basic job is to break down the high-level descriptions of graphics primitives such as vectors, text, and rectangles into a series of individual requests to draw pixels.

Drawing the vector is automatically triggered when the last of the parameters described in the specification is written into the macro function unit. The macro function then steps its way along the vector using the Bresenham algorithm[2] and issues requests to draw pixels. Since the slope of our vector is 1/2, the y-coordinate is incremented after every two steps along the x-axis as indicated in Fig. 8.

One might expect that a separate x- and y-address would be specified for each pixel to be written. However, with vectors there is excellent coherence between successive x- and y-addresses as pixels are drawn sequentially along the vector. Thus, there are special bus cycles between the macro function unit and the data formatter that specify that the previous x- or y coordinate should be incremented or decremented to generate the new coordinate. This saves sending a full x,y coordinate pair for each pixel drawn and significantly improves bandwidth use on the bus. This optimization is also useful for other primitives such as text and rectangles.

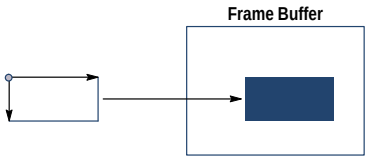


Fig. 6. Rectangular area fill primitive. A rectangle is defined by corner, width, and height. Color or pattern may be applied.

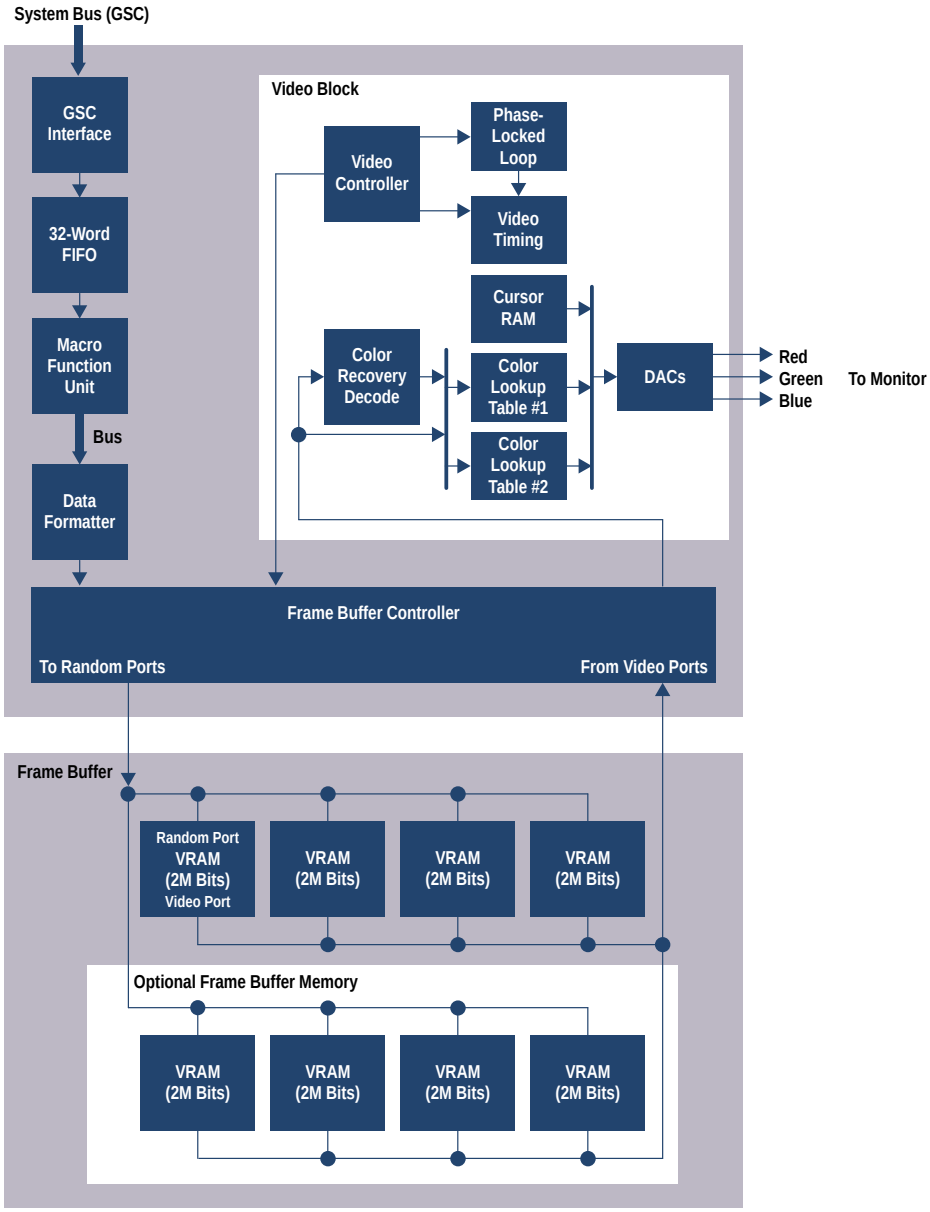


Fig. 7. A block diagram of the components inside the graphics chip.

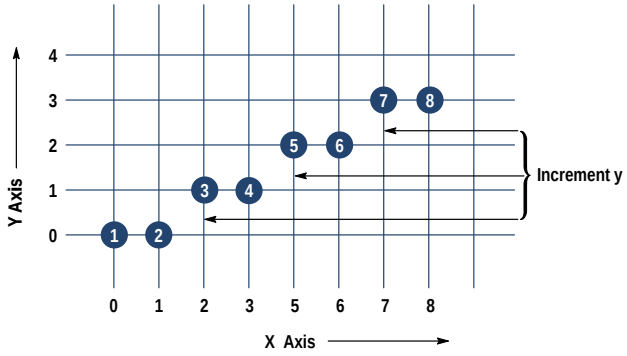


Fig. 8. Pixel representation of a vector that starts at coordinate 0,0, is 8 pixels long, and has a slope of 1/2.

The data formatter's job is to take requests and data from the macro function unit and format them in a way that is best for the frame buffer. In the case of our vector, the pixel addresses received by the data formatter are coalesced into rectangular tiles that are optimized for the frame buffer. The data formatter also recognizes when special VRAM modes may be enabled to improve performance, based on the sequence of data it receives from the macro function unit. For example, page mode (which is described in more detail later in this article) would be enabled during a vector draw. The data formatter also stores the current pixel address, vector color, and a host of other parameters for other primitives.

The frame buffer controller generates signals for the VRAMs based on the requests from the data formatter. The controller looks at the sequence of writes and reads requested and adjusts the timing on the VRAM signals to maximize performance. For our vector, we only need to do simple writes into the frame buffer, and cycles can be as fast as 37.5 ns per pixel. More complex primitives might require data to be read, modified, and written back, possibly to a different frame buffer location.

The graphics chip supports an 8-bit-per-pixel frame buffer. This means that, using normal techniques, only 256 colors can be displayed simultaneously. This is not always adequate for today's graphics-oriented systems. Two methods can be employed to increase the perceived number of colors. The first is dithering, in which an interleaved pattern of two available colors is used to visually approximate a requested color that is not directly available. The second approach is color recovery. Color recovery is visually superior to dithering and is described later.

The Model 712's entry-level configuration frame buffer uses four 2M-bit VRAM parts which allows screen resolutions of up to 1024 by 768 pixels. Adding four more VRAM chips on a daughter card enables screen resolutions up to 1280 by 1024 pixels.

In addition to the screen image data, data for the cursor, color lookup table, and attributes are stored in offscreen frame buffer memory. This is an area in the video RAM frame buffer that is never directly displayed on the CRT. Data in this region is accessed in exactly the same fashion as the screen image data, presenting a consistent interface to software driver writers.

At this point our vector exists in the frame buffer, but cannot be seen by the user. The video block is responsible for getting the screen image data from the frame buffer and converting it for display on the monitor. This display process is asynchronous to the rendering process which placed our vector in the frame buffer.

To get the data in the frame buffer to the monitor, the video controller first sends a request to the frame buffer controller to access the frame buffer data. This data is requested in sequential or scan-line order to match the path of the beam on the monitor. Next, the data from the frame buffer is run through a color lookup table to translate the 8-bit values into 8 bits each of red, green, and blue. The graphics chip supports two independent color lookup tables which are selected on a per-displayed-pixel basis by the attribute data. This feature helps eliminate color contention between applications sharing the frame buffer. Finally, cursor data is merged in by the video block and the digital video stream is converted to analog signals for the monitor.

This completes an overview of the life of a vector primitive, from a high-level description in the software driver to display on the monitor. This basic data flow is the same for other primitives such as rectangles and text.

Low Manufacturing Cost

Low cost was the primary objective for the graphics chip design. As a measure of our success, the manufacturing cost for the Model 712 graphics subsystem is 1/3 the cost of the original CRX graphics subsystem. In addition, the entry-level 1024-by-768-pixel version of the graphics chip costs five times less than the CRX subsystem.

These cost reductions were achieved primarily through an aggressive amount of integration, which is summarized in Fig. 9. The graphics chip represents the culmination of a series of optimizations of the CRX family, combining almost the entire GUI (graphical user interface) accelerator onto a single chip. The only major function not currently integrated is the frame buffer. Frame buffer integration is not feasible today because RAM and logic densities are not quite high enough and there is currently a cost advantage to using commodity VRAM parts.

Since the introduction of the CRX subsystem, industry trends such as denser and cheaper memory and inexpensive IC gates have contributed to cost reductions in graphics hardware. However, the graphics chip's high level of integration also contributes cost reductions in the following areas:

- Elimination of value-priced parts. The color lookup table and the digital-to-analog converter (DAC) have traditionally been an expensive component of the graphics subsystem. This is especially true for systems capable of high resolution (1280 by 1024 pixels, 135 MHz) and having multiple color lookup tables, such as the one built into the graphics chip. The digital phase-locked loop in the graphics chip replaces another expensive external part.
- The density of FETs achieved with the graphics chip, over 4500/mm^2, is significantly higher than with previous generations. This is important because silicon area is a major contributor to overall design cost.
- IC packaging and testing contribute significantly to the cost of each chip in a system. Reducing the number of chips eliminates this overhead. The graphics chip has a full internal scan path and many internal signature registers to reduce test time and chip cost significantly.

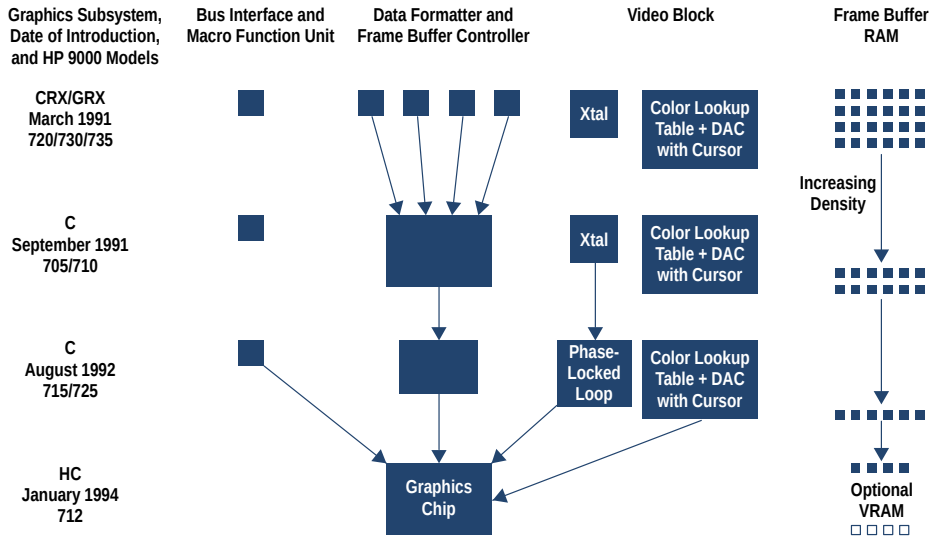


Fig. 9. The evolution of HP's graphical user interface (GUI) accelerator.

- Printed circuit board area is a significant system cost. The elimination of a large number of chips not only reduced the printed circuit board area from about 60 in^2 for the CRX to 14 in^2 for the graphics subsystem in the Model 712, but allowed the graphics to be integrated directly onto the motherboard, eliminating connectors, a bulkhead, and other mechanical components.
- Power consumption for the graphics subsystem in the Model 712 is only six watts. This low power consumption reduces power supply capacity and cooling requirements and therefore cost.
- Manufacturing costs associated with parts placement, test, and rework are proportional to the number of discrete components in a system. The graphics chip and and other chips in the Model 712 include JTAG (IEEE 1149.1) capability and signature generators to reduce the cost of printed circuit board test.

Several factors made this high level of integration practical. First, improved VLSI capabilities such as increased FET density, decreasing wafer costs and the availability within HP of video DAC technology. Secondly, the desktop availability of design and simulation tools capable of handling a model of over 300,000 gates and 500,000 transistors. VLSI design and verification were accomplished on HP 9000 Series 700 workstations using Verilog, Synopsys, and many in-house IC development tools. The performance of the workstations allowed the gate-level simulation of entire video frames (1/60 s of operation) of over 1.2 million pixels, which was the first time this was accomplished within HP.

Performance

The integration described above has also resulted in significant performance benefits. The two major reasons for the performance benefits are wider buses and increased clock rates.

Wider buses are possible between blocks when they are on the same piece of silicon. Wider buses allow better communication bandwidth at a given clock rate, with very little cost impact. A good example on the graphics chip is the much improved communication between the macro function unit and the data formatter which once existed as separate chips.

Increased clock rates are possible because of the elimination of chip-to-chip synchronization delays, pad delays, and printed circuit board trace delays. This compounds the bandwidth benefit of wider buses. HP's CMOS26B technology allows the bus interface, macro function unit, and frame buffer controller blocks of the graphics chip to operate at 80 MHz while the three DACs and two color lookup tables of the video block operate at 135 Mhz.

Intelligent system-level design also made major contributions to performance. A simple example is the block transfer commands which are responsible for transferring data from system memory to the graphics chip and its frame buffer. A special mode was introduced to the memory and I/O controller in the PA 7100LC which allows fast sequential double-word transfers without incurring the overhead of two single-word transfers. This simple change boosted block transfer performance by 50%.

Besides designing with a system-level focus, the other driving philosophy was to extract as much performance and utility as possible from available technology. A good example of this is the use of the advanced features available in the latest 2M-bit and 4M-bit VRAMs. HP has been instrumental in proposing and driving many of these enhancements within the JEDEC committee over the last few years. The more important features include:

- Page mode. This feature eliminates the need to send redundant portions of the pixel address when writing into the frame buffer. The result is that many operations can write a pixel in as little as 37.5 ns versus the more typical 70 ns (see Fig. 10). The key here is that these operations must occur within a page of VRAM or a significant penalty is incurred. By default this page is long and narrow, which is good for block move and block transfer operations but bad for randomly oriented vectors and rectangles. To achieve a better performance balance, we made use of the next feature.
- Stop register/split transfer. This feature allows the frame buffer to be organized in pages that are more square than long and narrow. Moving to this organization improves random vector and small rectangle performance significantly while only slightly reducing large horizontal primitive performance (see Fig. 11).

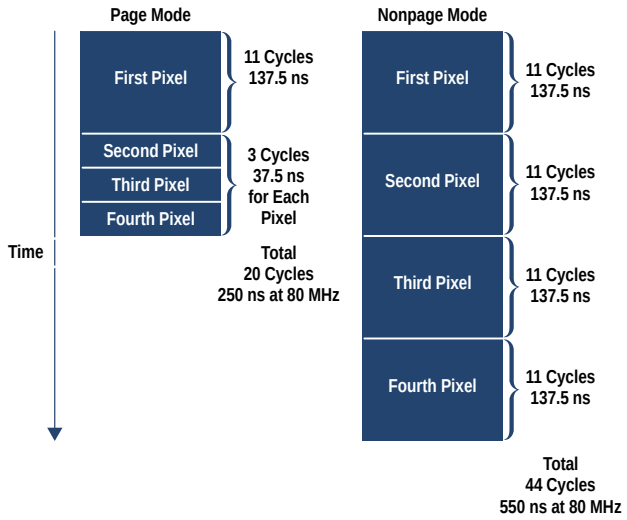


Fig. 10. An illustration of the performance improvement possible using the page mode to write pixels into the frame buffer. This example compares the performance of each mode when just four pixels are transferred to the frame buffer.

- Block write. As mentioned earlier, operations such as text and rectangular fill frequently require only one or two colors to be selected on a per-pixel basis. For this reason VRAMs provide a mode (via a single bit) in which a pixel's color can be selected from an 8-bit foreground or background color stored in the VRAMs. This translates into an 8× performance improvement for these types of operations.

The graphics chip's performance is summarized in Table I. The table compares the performance of the graphics chip at its theoretical hardware limit to its performance in 80-MHz and 60-MHz Model 712 workstations and the Model 720 CRX. The final row in Table I, Xmark, is an industry-standard metric that is an average of several hundred X Window System tests.

Note that the graphics chip's hardware limit is significantly higher than the Model 712 system performance limits. This headroom means that future systems with higher levels of CPU performance or even more highly tuned software drivers will be capable of even better window system performance.

Table I
Summary of the Graphics Chip's Performance

Benchmark	Hardware Limit	Model 712/80	Model 712/60	CRX 720
Block transfer 8-bit pixels/s (frame buffer to system memory)	96 M	60 M	52 M	42 M
Block transfer 8-bit pixels/s (system memory to frame buffer)	20 M	9M	8 M	2 M
Block move pixels/s (frame buffer to frame buffer, 500 by 500 pixels)	47 M	40 M	31 M	40 M
Vectors/s (10-pixel, X compliant)	2.1 M	1.4 M	1.1 M	1.1 M
Text characters/s (6 by 13 pixels/character)	1.0 M	681 k	385 k	295 k
Rectangles/s (10 by 10 pixels/rectangle)	1.7 M	790 k	588 k	270 k
Xmark	—	7.9	6.0	5.6

Compelling Functionality

Beyond improving performance and dropping cost substantially it was an important goal to include useful new functionality in the graphics chip. Below are some of the more important additions.

Software Video Support. One of the design goals for the Model 712 was to be able to play MPEG and H.261 video sequences without expensive hardware acceleration. Through careful analysis of the decoding process it became clear that this was possible at full frame rates and high visual quality using a combination of the following algorithmic, PA 7100LC, and graphics enhancements:

- Rewriting the standard decode algorithms to make them as efficient as possible
- Adding key instructions to the PA 7100LC
- Implementing YUV-to-RGB color space conversion in the graphics chip.

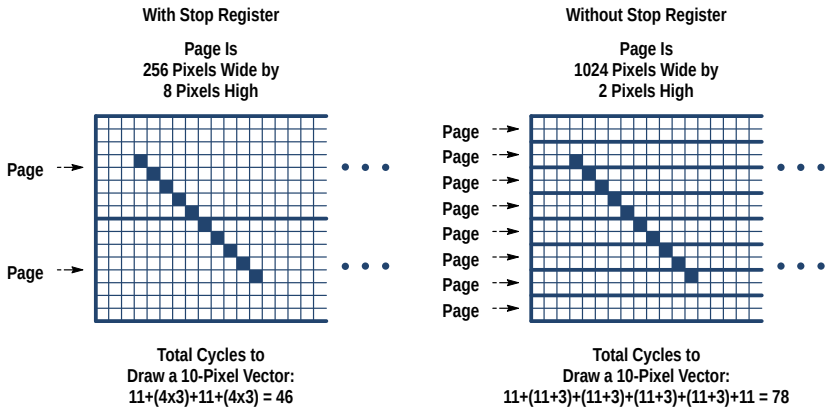


Fig. 11. Improving performance with frame buffer pages that are more square than long and narrow.

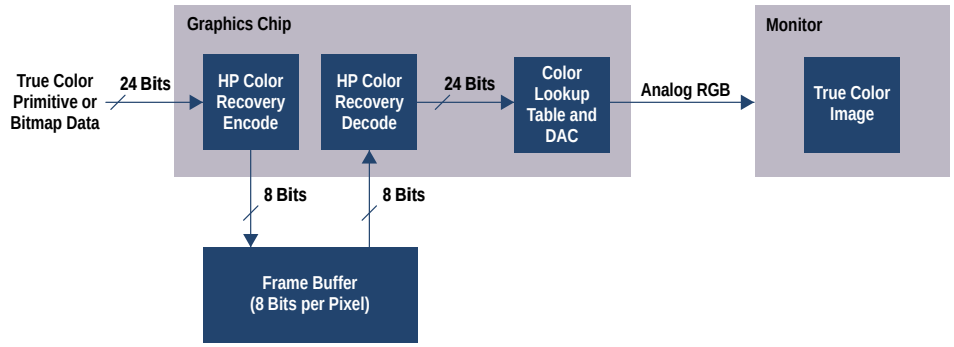


Fig. 12. The HP Color Recovery pipeline.

YUV encoding is used in many video formats. It allocates proportionately more bits to encode the brightness or luminance (Y) of the image, and fewer bits to represent the color (UV) in the image. Since the human eye is more sensitive to brightness than color, this is an efficient scheme. However, since the graphics chip's frame buffer is stored in RGB format, a conversion from YUV to RGB is necessary.

This conversion is a good example of an operation that was relatively expensive in software (a 3-by-3 16-bit matrix multiply) but simple to do in the the graphics chip hardware. This simple addition alone improves video playback performance by as much as 30% and helps enable full 30-frame/s 320-by-288-pixel resolution MPEG playback on a Model 712/80.

HP Color Recovery. The graphics chip incorporates a new display technology called HP Color Recovery. Using a low-cost 8-bit frame buffer and HP Color Recovery, the graphics chip can display images that are in many cases visually indistinguishable from those of a 24-bit frame buffer costing three times more. This feature is useful for the following application areas:

- Visual multimedia (JPEG, MPEG, etc.)
- Shaded mechanical CAD models
- Geographical imaging system
- Document image management
- Visualization
- High-quality business graphics.

A block diagram of the HP Color Recovery pipeline is shown in Fig. 12.

The HP Color Recovery encoding scheme causes no loss of performance for rendering operations and is related to traditional ordered dithering. Dithering is widely used to approximate a large number of colors with an 8-bit frame buffer and is also available in the graphics chip.

The HP Color Recovery decode is much more sophisticated and based on advanced signal processing techniques. This circuitry cycles at 135 MHz and achieves over 9 billion operations per second. HP Color Recovery is described in more detail in the article on page 51.

Multiple Color Lookup Tables. Typically, entry-level workstation and personal computer graphics subsystems have had only a single color lookup table with a limited number of entries, usually 256. In the X Window System this results in the annoying flashing of backgrounds or window contents when a new application is started that takes colors from existing applications. The graphics chip solves this problem in a majority of cases by providing two 256-entry color maps. For most interactions in which the user is focused on a single application and the window manager, this completely eliminates the resource contention and results in a visually stable screen (see Fig. 13).

Software Programmable Resolutions. One of the problems of past workstation graphics subsystems is that they operate at a fixed video resolution and refresh rate. This has posed problems in configuring systems at the factory and during customer upgrades. The graphics chip incorporates an advanced digital frequency synthesizer that generates the clocks necessary for the video subsystem. This synthesizer, based on HP proprietary digital phase-locked loop technology, allows software configurability of the resolution and frequency of the video signal. Thus, alternate monitors can be connected without changing any video hardware. Currently supported configurations include:

- 640 by 480 pixels 60 Hz, standard VESA timing
- 800 by 600 pixels 60 Hz
- 1024 by 1024 pixels 75 Hz and flat panel
- 1280 by 1024 pixels 72 Hz.

As new monitor timings appear, the graphics chip can simply be reprogrammed with the parameters associated with the new monitor.

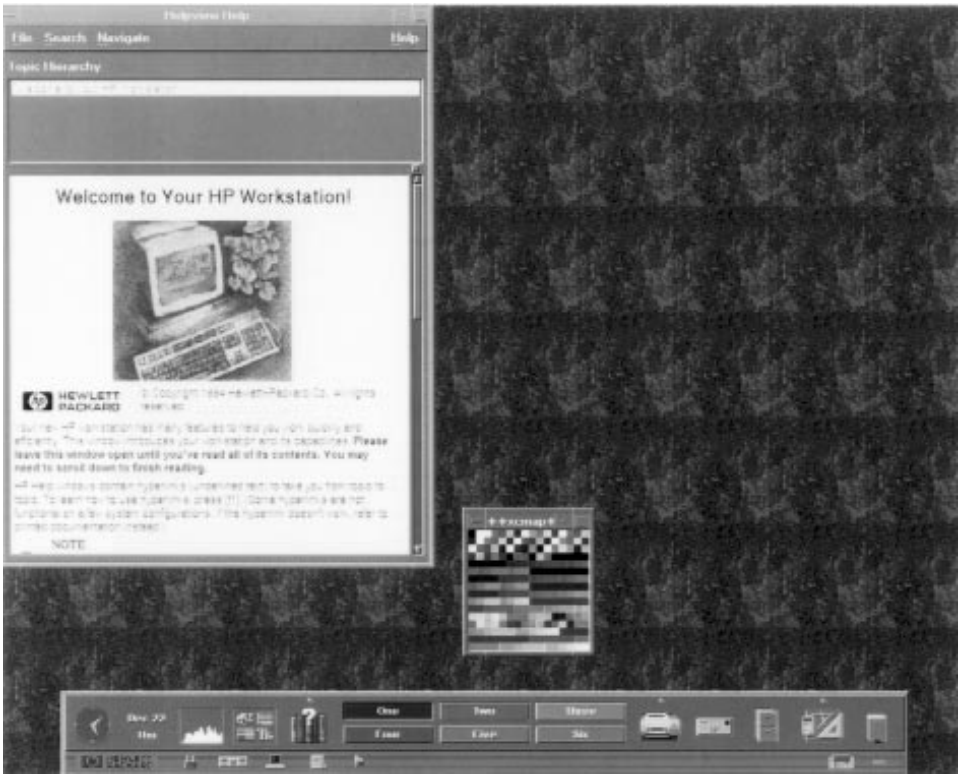
Summary

We created the graphics chip with the philosophies of system-level-optimized design and optimal use of technology. This enabled us to meet our goals of very low manufacturing cost, leadership performance at our cost point, architectural compatibility, and introduction of some important new functionality.

Acknowledgments

This paper briefly summarizes the work of many dedicated and creative members of the graphics chip development team in the graphics hardware and software laboratories in Fort Collins and the Integrated Circuits Business Division. Many thanks to Harry Baeverstad, Tony Barkans, Raj Basudev, Dale Beucler, Rand Briggs, Joel Buck-Gengler, Mike Diehl, Ales Fiala, Randy Fiscus, Dave Maitland, Bob Manley, Dave McAllister, Peter Meier, John Metzner, Brian Miller, Gordon Motley, Donovan Nickel, Cathy Pfister, Larry Thayer, Brad Reak, Cal Selig, James Stewart, and Gayvin Stong for their exceptional efforts.

(a)



(b)

Fig. 13. Comparison between single and multiple color lookup tables. (a) One color lookup table. (b) Two color lookup tables.

References

1. D. Rhoden and C. Wilcox, "Hardware Acceleration for Window Systems," *Proceedings of SIGGRAPH '89, in Computer Graphics,* Vol. 23, no. 9, July 1989, p. 67.

2. J. Bresenham, "Algorithm for Computer Control of a Digital Plotter," *IBM System Journal,* Vol. 4, no. 1, 1965, pp. 25-30.