

An Overview of the VISUALIZE fx Graphics Accelerator Hardware

Noel D. Scott

Daniel M. Olsen

Ethan W. Gannett

Three graphics accelerator products with different levels of performance are based on varying combinations of five custom integrated circuits. In addition, these products are the first ones from Hewlett-Packard to provide native acceleration for the OpenGL[®] API.

The VISUALIZE fx family of graphics subsystems consists of three products, fx⁶, fx⁴, and fx², and an optional hardware texture mapping module. These products are built around a common architecture using the same custom integrated circuits. The primary difference between these controllers is the number of custom chips used in each product (see **Table I**).

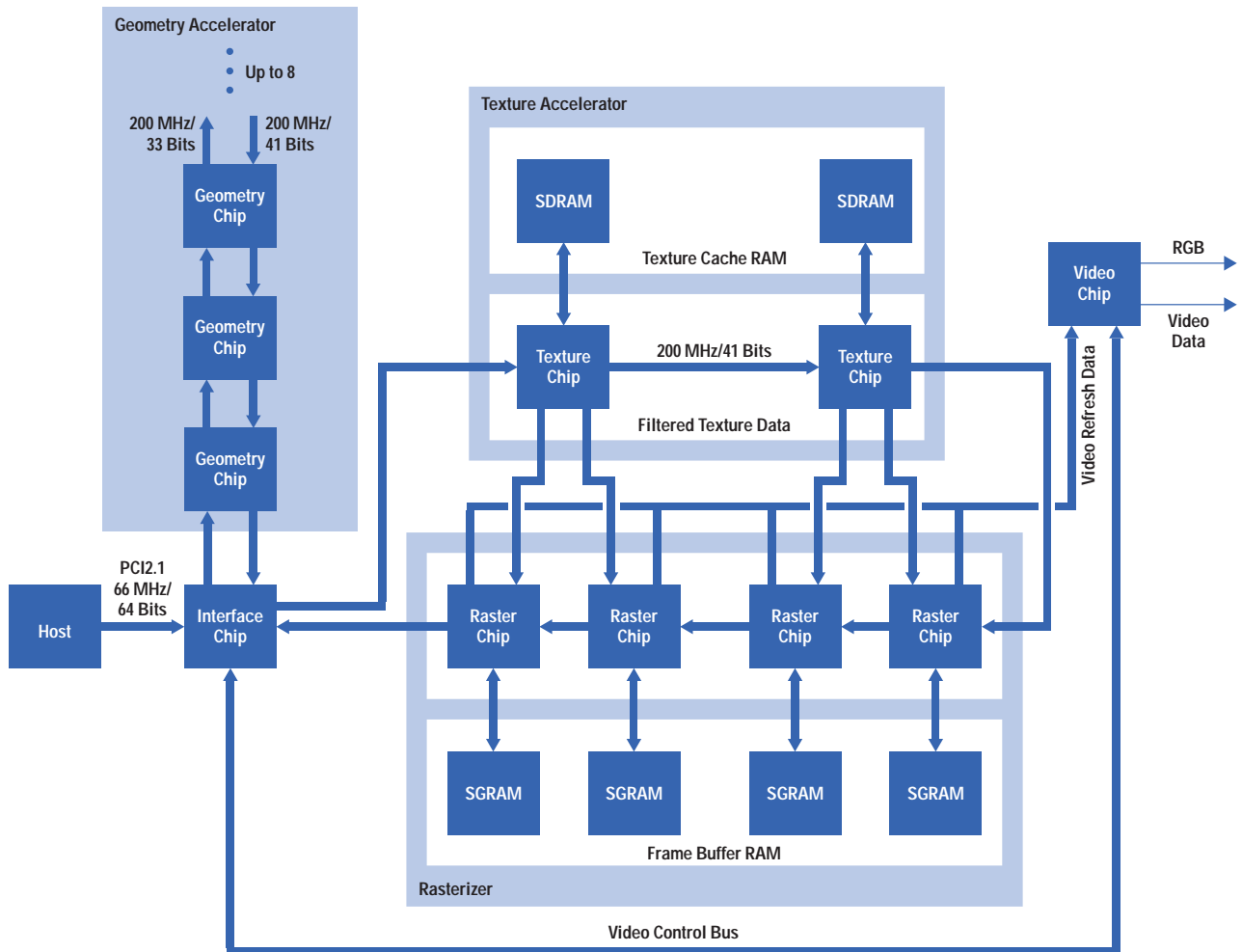
Table I
Number of custom chips in the different VISUALIZE fx products

Product	Texture Chip	Geometry Chip	Raster Chip
fx ²	—	1	2
fx ⁴	1	2	2
fx ⁶	2	3	4

A chip-level block diagram of the VISUALIZE fx⁶ product is shown in **Figure 1**. This is the most complex configuration and also the one with the highest performance in the product line. The VISUALIZE fx⁴ and the VISUALIZE fx² products use subsets of the chips used in the fx⁶. The fx⁶ and fx⁴ subsystems have support for the optional hardware-accelerated texture map module, which contains a local texture cache for storage of texture map images. If the texture accelerator is not present, the bus between the interface chip and the first raster chip is directly connected.

Figure 1

A chip-level diagram of the VISUALIZE fx⁶ product.



- | | | |
|---|---|--|
| <p>Geometry Chip</p> <ul style="list-style-type: none"> • 3D Geometry and Lighting Acceleration <p>Texture Chip</p> <ul style="list-style-type: none"> • Texture Rasterization • Texture Map Cache Controller • Texture Memory Control • Texture Interpolation | <p>Interface Chip</p> <ul style="list-style-type: none"> • I/O Buffering • 3D Geometry Workload Distribution and Concentration • 2D and 3D Data Path Arbitration • 2D Acceleration • YUV to RGB Conversion Support • Pixel Level Pan and Zoom • Pixel Level Image Rotations | <p>Raster Chip</p> <ul style="list-style-type: none"> • Fragment Processing • Frame Buffer Control Functions <p>Video Chip</p> <ul style="list-style-type: none"> • Color Lookup Tables • Video Timing • Digital-to-Analog Conversion • Video-Out Data |
|---|---|--|

Interface Chip

The interface chip provides a PCI 2.1 (also referred to as PCI 2X) compliant interface.* It operates at up to 66 MHz in 64-bit mode. Special efforts have been made in the

design of the buffering and the interface to the PCI. As a result, the driver is able to sustain writes of 3D geometry commands to the PCI at almost the theoretical maximum rates that could be computed for the PCI. The article on page 51 discusses PCI capability.

* PCI = Peripheral Component Interconnect.

Occlusion Culling

The HP fast-break program (page 8) enabled us to understand customer requirements by analyzing what is important in OpenGL graphics today. As a result, we developed a technology called *occlusion culling* as an extension to OpenGL and implemented it in the VISUALIZE fx graphics hardware.

We found that the data sets many graphics workstation customers are trying to visualize are very complex. These data sets have large numbers of small, complex components that are not always visible in the final images. For instance, when rendering an airplane, all of the MCAD parts are present in the data set represented by potentially millions of polygons that must be processed. However, when this airplane is viewed from the outside only the outer surfaces are visible, not the fan blades of the engine or the seats or bulkheads in the interior.

In a traditional 3D z-buffered graphics system, all polygons in a scene must be processed by the graphics pipeline because it is not known a priori which polygons will be visible and which ones will be occluded (not visible). The notion of occlusion culling, or removal of occluded objects, has been talked about in the research community for several years. However, implementations tend to be in software where the performance is not at a satisfactory level.

In the VISUALIZE fx series of graphics devices, HP developed a very efficient algorithm that tests objects for visibility. An application program can very quickly use the occlusion culling visibility test to determine if a simple bounding box

representation of a more complex part is visible. Since a bounding box, or more generally a bounding volume, completely encloses the more complex part, it is possible to know a priori that if the bounding volume is not visible then the complex part it encloses is not visible. Thus, the part that is not visible does not need to be processed through the graphics pipeline. The real benefit of occlusion culling comes when a very complex part consisting of many vertices can be rejected, avoiding the expenditure of valuable time to process it.

For very complex data sets, such as the airplane mentioned above or an automobile, a tremendous performance increase can be realized by using the HP occlusion culling technology. To date, several ISVs have begun using occlusion culling in their applications and are seeing a 25 to 100 percent increase in graphics performance. This magnitude of performance benefit typically costs a customer several thousand dollars for the extra computational horsepower. HP includes this technology as standard in all VISUALIZE fx series graphics accelerators, giving even better price and performance results to our customers.

The future of 3D graphics will continue toward visualizing ever more complex objects and environments. Occlusion culling together with HP's DirectModel technology (page 19) are well positioned to be industry leaders in providing the technology for 3D modeling applications.

The primary responsibility of the interface chip is to separate the streams of data that arrive from the host SPU into three paths and arbitrate access among those paths.

3D Path. Typically data from the host CPU looks very much like the OpenGL API functions themselves. Data following this first path is routed to the geometry chips. The geometry chips process the data and return the results to the interface chip. These results are then sent on to the texture chips or directly to the raster chips if the texture mapping subsystem is not installed. In either case the data is transmitted to and through all the texture and raster chips in the system.

Unbuffered Path. This path passes data directly through the interface chip to the texture and raster chips. This provides a bypass method that allows traffic to get around

other pending operations. An example would be a texture cache download that is required to complete a primitive that is currently being rasterized, a situation that would lead to deadlock without the unbuffered path.

2D Path. This path runs directly through the interface chip to the texture and raster chips. The 2D path differs from the unbuffered path in the way its priority is handled. The interface chip manages priority among the three paths as they all converge on the same set of wires between the interface chip and the first texture chip. The unbuffered path goes directly through the interface chip to those wires and has priority over the other two paths. Data targeting the 2D path is held off until all preceding 3D work in the geometry chip has been flushed through to the first texture chip.

There is also special circuitry in the interface chip that is used to accelerate many operations commonly done by X11 or other 2D APIs.

Buses

The three primary buses in the system are each run at 200 MHz, allowing sustainable transfer rates of more than 800 Mbytes per second. To control the loading on the interconnections for these buses, they are built as point-to-point connections from one chip to the next.

Each chip receives the signals and then retransmits them to the next chip in the sequence. This requires more pins on each part, but limits the number of loads on each wire to a single receiver as well as limiting the wiring length that signals must traverse. This allows for reliable communications despite the high frequency of the buses.

The first of these three buses distributes work to the geometry chips. This bus starts at the interface chip and runs through all the geometry chips in the system. Each geometry chip monitors the data stream as it flows through the bus and picks off work to operate upon based on an algorithm that selects the least busy geometry chip.

The second of these buses starts at the last geometry chip and passes through the others back to the interface chip. The results of the work done by the geometry chips is placed on this bus in the same sequence as it was moved along the first bus. This strict ordering control prevents certain artifacts from showing up in the final image.

The third bus ties the interface chip to the texture and frame buffer subsystems. It is wired in a loop that goes back to the interface chip from the last chip in the chain. 3D operations typically flow from the interface chip to the chips along this bus, and when they eventually get back to the end of the loop, they are thrown away.

For 2D operations, such as moving blocks of pixels around the frame buffer, the operation of the third bus is somewhat different. The movement of pixel data operates as a sequence of reads followed by a sequence of writes. The reads cause data to be dumped from the frame buffer locations onto the bus and the results travel back to the interface chip. This data is then associated with new addresses and sent as writes back down the bus, ending up back at the frame buffer but in different locations.

Besides the three primary buses mentioned above, there are three secondary buses in the system. The first

bus connects the interface chip to the video chip. This provides video control, download of color maps, and cursor control. The second bus is a connection from each raster chip to the video chip. This path is used to provide video refresh data to display frame buffer contents. The final secondary bus is a connection from each texture chip to two of the raster chips. This path allows the flow of filtered texture data into the raster chips for combination with nontexture fragment data.

Geometry Chip

The geometry and lighting chips are responsible for taking in geometric primitives (points, lines, triangles, and quadrilaterals) and executing all the operations associated with the transform stage of the graphics pipeline (see the article on page 9 for more about the graphics pipeline). These operations include:

- Transformation of the coordinates from model space to eye space
- Computing a vertex color based on the lighting state, which consists of up to eight directional or positional light sources
- Texture map calculations that include:
 - Environment map calculations for texture mapping
 - Texture coordinate transformation
 - Linear texture coordinate generation
 - Texture projection
- View volume clipping and clipping against six arbitrary application-specified planes to determine whether a primitive is completely visible, rejected because it is completely outside the view area, or needs to be reduced into its visible components
- Perspective projection transformation to cause primitives to look smaller the further away from the eye they are
- Setup calculations for rasterization in the raster chip.

There were some interesting problems to solve in the design of the distribution and coalescing of work up and down the geometry chip daisy chain. For example, load balancing, maintaining strict order in the output stream, and ensuring that operations, such as binding of colors and normals to vertices, perform as required by OpenGL.

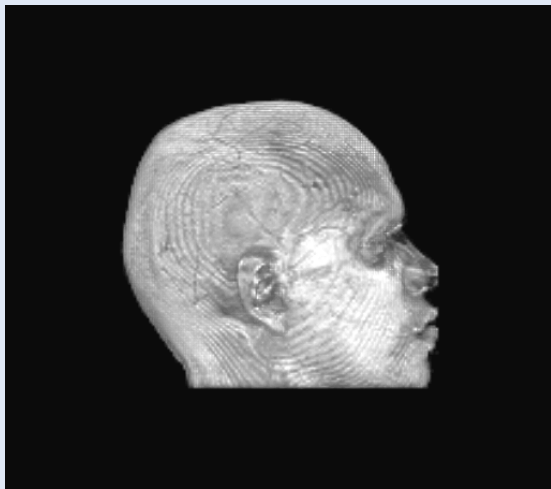
Fast Virtual Texturing

Texture mapping, which is wrapping a picture over a three dimensional object, has been used over the years as a key feature to enhance photorealism, reduce data set sizes, perform visual analysis, and aid in simulations (see **Figure 1**). Since texturing calculations are computationally expensive and memory access for large textures can be prohibitively slow, various workstation graphics vendors have provided hardware-accelerated texture mapping as a key differentiator for their product.

A primary drawback of these attempts at hardware acceleration is that dedicated local hardware texture memory is limited

Figure 1

A 3D textured skull. The VISUALIZE fx⁴ and fx⁶ subsystems support a texture map acceleration option. Pictured here is the use of 3D texture mapping OpenGL extensions with this option. This feature allows visualization of 3D data sets such as MRI images.



in size and is expensive. To take advantage of the performance boost, graphics applications were constrained to textures that fit in the local hardware texture memory. In other words, the application was responsible for managing this hardware resource.

Noticing this obvious artificial application limitation in texturing functionality, performance, and portability, Hewlett-Packard introduced, in the VISUALIZE-48, a new concept in hardware texture mapping called *virtual texture mapping*. Virtual texture mapping uses the dedicated local hardware texture memory as a true texture cache, swapping in and out of the cache the portions of textures that are needed for rendering a 3D image. Thus, for texturing applications, these limitations were eliminated. The application could define and use a texture map of any size (up to a theoretical limit of 32K texels × 32K texels*) that would be hardware accelerated, eliminating the need for the application to be responsible for managing local texture memory.

Using the local hardware texture memory as a cache also means that this memory uses only the portions of the texture maps needed to render the image. This efficiency translates to more and larger texture maps being hardware accelerated at the same time. Applications that previously could not run because of texture size limits can now run because of the unlimited virtual texture size. Also, with only the used portions of the texture map being downloaded to the cache, far less graphics bus traffic occurs.

The system design of virtual texture mapping involved changes in the HP-UX operating system to support graphics interrupts, onboard firmware support for these interrupts, the introduction of an asynchronous texture interrupt managing daemon process, and the associated texturing hardware described in this

*A texel is one element of a texture.

The output of the geometry chip's daisy chain is passed back through the interface chip. Generally, for triangle based primitives, the output takes the form of plane equations. As these floating-point plane equations are returned from the geometry chip to the interface chip and passed on to the texture chips, certain addressed locations in the interface chip will result in the floating-point values being

converted to fixed-point values as they pass through. These fixed-point values are in a form the raster chips need to rasterize the primitive.

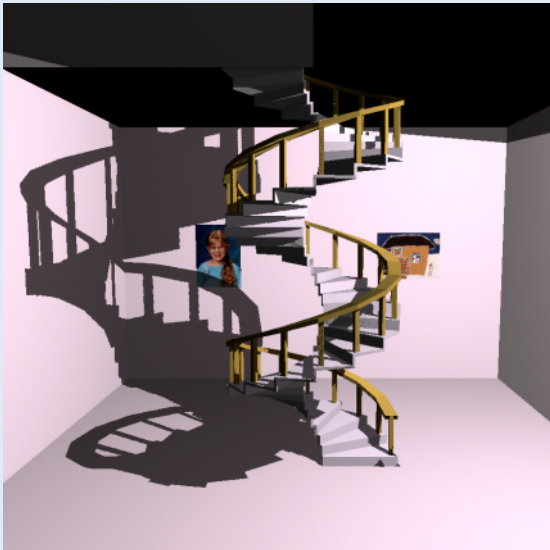
The daisy-chain design allows up to eight of the geometry chips to be used although only three are applied in the case of the VISUALIZE fx⁶ product at this time.

article. Having a centralized daemon process manage the cache allows for cache efficiency, parallel handling of texture downloads while 3D graphics rendering is occurring, and sharing textures among graphics contexts.

The VISUALIZE fx⁴ and VISUALIZE fx⁶ texture mapping options incorporate the second generation advances in virtual texture mapping. Full OpenGL 1.1 texture map hardware support has brought about dramatic improvements in texture map download performance and switching between texture maps and new extended features such as 3D texture mapping, shadows (**Figure 2**), and proper specular lighting on textures

Figure 2

A shadow texture image.

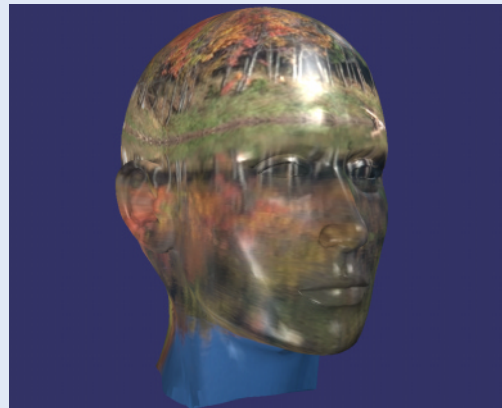


(**Figure 3**). These features have made these products very appealing systems for texturing applications on workstation graphics.

The texture mapping performance on these systems is very competitive. The VISUALIZE fx⁶ texture fill rate is about twice that of the VISUALIZE fx⁴ texture option. However, fill rates alone do not describe how these systems perform in a true application environment. Aggressive texture mapping application performance comparisons show two to three times performance superiority over similarly priced graphics workstation products.

Figure 3

A specular lit texture image. Correct specular lighting of textured images can be achieved with VISUALIZE fx⁴ and fx⁶ texture mapping options.



Texture Chip

The texture chip is responsible for accelerating texture mapping operations. Towards this end, it performs three basic functions:

- Maintains a cache of texture map data, requesting cache updates for texture values required by current rendering operations as needed (see “Fast Virtual Texturing” on page 32)

- Generates perspective corrected texture coordinates from plane equations representing triangles, points, or lines
- Fetches and filters the texture data as specified by the application based on whether the texture needs to be magnified or minimized to fit the geometry it is being mapped to and passes the result on to the raster chips.

Raster Chip

The raster chip rasterizes the geometry into the frame buffer. This means it determines which pixels are to be potentially modified and, if so, whether they should be modified based on various current state values (including the contents of the z buffer). The raster chip also controls access to the various buffers that make up the frame buffer. This includes the image buffer for storing the image displayed on the screen (potentially two buffers if double buffering is in effect), an overlay buffer that contains images that overlay the image buffer, the depth or z buffer for hidden surface removal, the stencil buffer,* and an alpha buffer** on the VISUALIZE fx⁶. To accomplish its work the raster chip performs four basic functions:

- Rasterize primitives described as points, lines, or triangles
- Apply fragment operations as defined by OpenGL (such as blending and raster operations)
- Control of and access to buffer memory, including all the buffers described earlier
- Refresh the data stream for the video chip, including handling windows and overlays.

Video Chip

The video chip provides video functions for controlling the data flow from the frame buffer to the display and

* A stencil buffer is per pixel data that can be updated when pixel data is written and used to restrict the modification of the pixel.

** An alpha buffer contains per pixel data that describes coverage information about the pixel and can be used when blending new pixel values with the current pixel value.



Noel D. Scott

Noel Scott is a senior engineer at the HP Workstation Systems Division.

He is responsible for product definition, performance projections, and modeling. He designed the I/O bus for the geometry chip described in the article. He came to HP in 1981 after receiving a BS degree in computer engineering from the University of Kansas.



Daniel M. Olsen

A software engineer in the graphics products laboratory at the HP

Workstation Systems Division, Daniel Olsen is responsible for the development of new 3D products for HP workstations. He has been with HP since 1994. He has a BSEE degree (1991) from North Dakota State University and an MS degree in computer engineering (1997) from Iowa State University. Daniel was born in Des Moines, Iowa, is married and has two daughters. His leisure time activities include skiing, home projects, scuba diving, and aviation.



Ethan W. Gannett

Ethan Gannett is a lead engineer for graphics software development

at the HP Workstation Systems Division. He came to HP in 1988 after receiving an MS degree in computer science from Iowa State University. He also holds a BS degree in physics (1983) and a BS degree in astronomy (1983) from the University of Iowa. Born in Davenport, Iowa, he is married and has one daughter. He enjoys kayaking, backcountry camping, telemarking, and hiking.

mapping data from values to color. The features of the video chip include:

- Data mapping to colors:
 - Two independent 4096-by-24-bit lookup tables
 - Four independent 256-by-3-by-8-bit lookup tables for image planes
 - A bypass path for 24-bit true color data
 - Two independent 256-by-8-bit lookup tables for overlay planes
- Digital-to-analog conversion
- Video timing
- Video output.

Conclusion

The VISUALIZE fx family of products currently has a substantial lead in not only price/performance measurements, but it also leads in performance independent of cost.

For information regarding how these systems compare against the competition, visit the SPEC (an industry standard body of benchmarks) web page at:

<http://www.spec.org/gpc>

Acknowledgments

We would like to thank Paul Martz for the shadow texture image (Figure 2 on page 33).

▶ [Go to Next Article](#)

▶ [Go to Journal Home Page](#)