New Wildcat Realizm Graphics Technology

Bigger Bandwidth, Greater Programmability, Higher Precision

Achieving New Levels of Performance, Scalability, and Visual Fidelity!

Introduction

This whitepaper introduces the unique technology underlying the next-generation Wildcat® Realizm™ graphics subsystems from 3Dlabs. Addressing high and ultra-high-end 3D graphics requirements for both AGP 8x and PCI Express-based environments, Wildcat Realizm technology establishes previously unimaginable performance and visual fidelity standards for professional visual computing applications.

Graphics technology is currently undergoing a process of extreme architectural churn. A common scenario circa 2000 was the use of fixed-function devices called Graphics Processing Units (GPUs). Engineered for extreme speed, GPUs feature highly parallel pipelines that exploit the natural parallelism inherent in vertex processing and pixel processing algorithms. However, their hardware-centric implementations largely dictate that these algorithms are frozen in silicon. This limits the algorithmic complexity that can be supported and prevents these devices from being easily adapted to accommodate evolving standards.

The latest developments in graphics processing engines are Vertex/Scalability Units (VSUs) and Visual Processing Units (VPUs). These revolutionary devices combine massive parallelism with programmable hardware architectures and a compiler-centric approach. In addition to their extreme graphics processing capability, their programmable architectures mean that VSUs and VPUs can support never-before-seen visual effects.

In addition to supporting the state-of-the-art in resolutions, fill rates, and SuperScene antialiasing (with 4, 8, or 16 samples), Wildcat Realizm technology also raises the level of realism with features such as a programmable pipeline that is fully floating-point throughout.

Wildcat Realizm technology also offers some unique features. These include the use of a Hierarchical Z-buffer depth culling and the direct display of 16-bit floating-point values, both of which provide substantial algorithmic efficiencies and corresponding increases in processing speeds and capabilities. When coupled with this level of performance, the programmable nature of Wildcat Realizm technology allows application developers to fully unleash their creativity. Since this technology is the first to be based on a 3D graphics pipeline that exhibits true floating point from the input vertices to the final displayed pixels means end users now have access to new levels of image quality and realism.

Shading Languages

To discuss Shading Languages in the appropriate context, one must understand how the 3D graphics pipeline works. A 3D graphics application running on the host platform communicates with the graphics subsystem via a bus interface (the two interfaces of interest for the purposes of this paper are AGP 8x and PCI Express).

The application views each object in its 3D world as being composed of a collection of polygons (usually triangles). The application passes data such as the XYZ (location) and color information associated with the vertices (corners) for each polygon to the graphics subsystem (Figure 1).

© 2004 3Dlabs, www.3dlabs.com

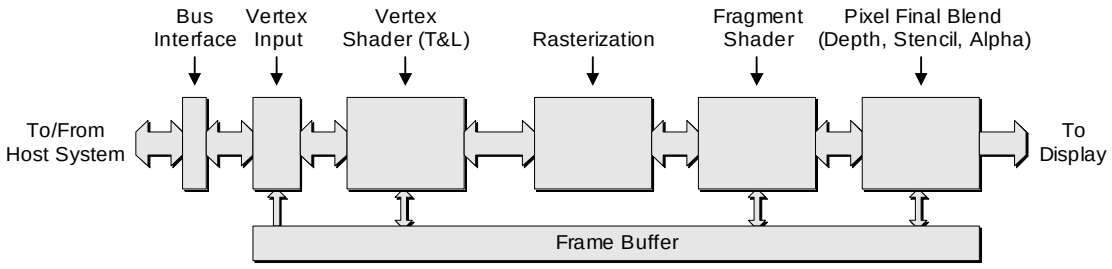


Figure 1. A highly simplified view of the 3D graphics pipeline

This vertex data is operated by a suite of Transformation and Lighting (T&L) algorithms that perform tasks such as transforming 3D world coordinates into corresponding 2D display coordinates and determining the lighting conditions at each vertex. In today's terminology, this unit is referred to as a Vertex Shader. The resulting data is passed via a rasterization engine into a Fragment Shader, which calculates values such as the red, green, blue, and alpha color components associated with each fragment (pixel). Following any final blend operations, the data is ultimately handed over to a display device.

There are a number of well-known graphics Application Programming Interfaces (APIs) that application developers might choose to use. In the case of home entertainment programs such as computer games, many developers choose to work with the DirectX® API from Microsoft®. By comparison, when it comes to high-end visual computing applications such as Computer-Aided Design (CAD), Digital Content Creation (DCC), engineering, medical imaging, and visual simulation applications, developers typically opt for the industry-standard open platform API, OpenGL®.

What is a Shader?
A shader is source code written in a graphics specific programming language that runs programmable graphics hardware.

Shaders are used to define the processing that occurs on each vertex, and pixel/fragment that is provided for display.

By using a shader you can implement your own specialized surface color, texture, and material properties; light source positions and emission characteristics; and transmission properties of fog and many other visual effects.

Both DirectX and OpenGL employ underlying 3D graphics pipelines architecture as illustrated in Figure 1. Over time, the level of sophistication associated with shading algorithms has increased to meet ever-increasing demands for more realism. For example, very simple "flat" shading was replaced by the slightly more sophisticated Gouraud shading algorithm. This was subsequently augmented by bilinear and later trilinear MIP-mapped textures. Similarly, only a limited selection of simple models for ambient, directional, and positional light sources were commonly available via these APIs. In addition to being relatively simplistic, these shading algorithms and lighting elements were well defined and widely accepted, which made them amenable to being implemented using fixed-function GPUs.

Problems started to arise when users wished to make use of more realistic light sources such as a square light panel combined with more sophisticated shading algorithms such as Phong Shading and more realistic illumination effects employing techniques such as Torrance Cook micro-facet and BRDF-based lighting models. As these elements had not been hard wired into the GPUs, the only way to implement them was in software running on the host system's general-purpose microprocessor. Tapping into the host system to perform rendering functions was horrendously slow, and was only used for final production rendering. The fact that rendering each frame might take minutes, or even hours, was considered acceptable by most users as the price one had to pay in order to achieve a professional level of visual fidelity.

After a number of false starts, such as the use of hardware-centric techniques based on multi-register combiners, it became obvious that the real solution was to move away from a 3D graphics pipeline based on fixed-function GPUs. Instead, the pipeline should be programmable, thereby allowing application developers to create new vertex, fragment, and pixel shaders as required (Figure 2).

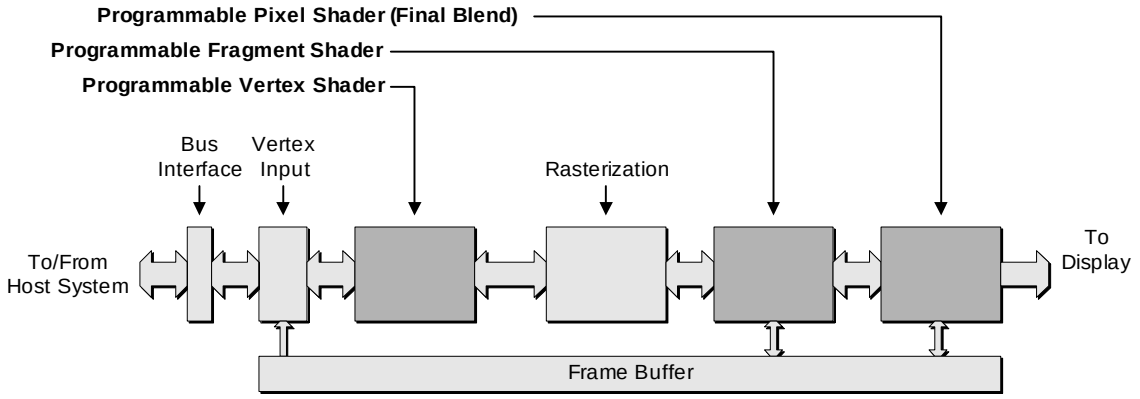
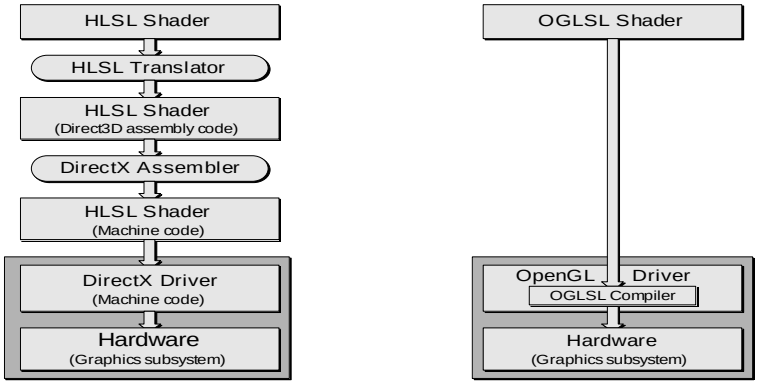


Figure 2. Augmenting the 3D graphics pipeline with programmable shaders

Depending on the actual implementation, having programmable vertex, fragment, and pixel shaders can unleash the creativity of application developers. These developers now have a mechanism to implement algorithms that are magnitudes faster and more sophisticated than the combination of Gouraud Shading with simple texture mapping, and that can provide real-time preview rendering that is much closer to production-level quality. Furthermore, developers are no longer limited to aiming only at photorealism. It is now possible to implement novel T&L algorithms and to realize watercolor, oil painting, or pen-and-ink renderings.

When it comes to creating custom shaders, two main approaches allow these shaders to be captured in a high-level, augmented C-like language. As shown in Figure 3a, the layered approach employed by DirectX involves creating a shader in what is known as High-Level Shading Language (HLSL).



(a) The DirectX "layered" approach (b) The OpenGL "internal" approach

Figure 3. High-level comparison of different techniques for implementing shading languages

This HLSL shader is then run through a translator to generate equivalent DirectX assembly language source code. In turn, the DirectX assembler processes this assembly code to generate corresponding machine code, which is subsequently loaded into the graphics hardware by the DirectX driver. Thus, an HLSL shader is simply an additional layer that conceptually resides above an existing API. Although such shaders certainly facilitate relatively easy programming,

they do not actually provide any increase in fundamental processing power or programming flexibility. Also, the fact that these shaders are translated into assembly code outside of DirectX makes for a somewhat time consuming and unwieldy implementation path.

By comparison, the technique adopted by OpenGL is to embed the compiler for the OpenGL Shader Language *inside* the OpenGL Driver itself (Figure 3b). This approach provides application developers with C-like programming capabilities for implementing shaders directly into the graphics subsystem. Of course, unlike an HLSL shader that will only work on PCs, an OpenGL shader will work on all systems that support OpenGL and the OpenGL Shading Language, including Windows® and Linux® OS environments.

AGP 8x versus PCI Express-based Bus Interfaces

Previous graphics subsystems running on the PCI bus interface, could accommodate a peak data transfer rate of only 133 MB/s at 33 MHz and over a 32-bit (4-byte) bus. This quickly became unacceptable for any kind of serious graphics-intensive application, so the Advanced Graphics Port (AGP) standard was developed.

The first incarnation of AGP was a 32-bit bus running at 66 MHz, thereby providing a peak data transfer rate of 266 MB/s. This initial release was subsequently upgraded to AGP 2x, which used a double-clocking technique to provide 533 MB/s. A later incarnation, AGP 4x, used a quad-clocked data transfer technology to achieve a peak bandwidth of a little over 1 GB/s; and the current interface, AGP 8x, octal-pumps data to realize slightly more than 2 GB/s. At the time of this writing, it is generally accepted that the AGP technology has reached its peak and there are no plans to go beyond the current AGP 8x implementation.

This poses a problem for today's high and ultra-high-end graphics subsystems, because they require significantly more bandwidth in order to handle the huge amounts of data associated with professional graphics applications. The solution is a serial data communications technology called PCI Express. A single PCI Express lane employs two differential pairs to provide a point-to-point connection between two devices: one to transmit and the other to receive data (Figure 4).

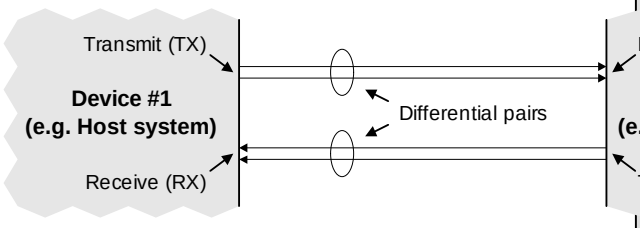


Figure 4. A single PCI Express lane

Widely respected for its industry leadership, 3Dlabs pioneered the development of the recently ratified OpenGL Shading Language.

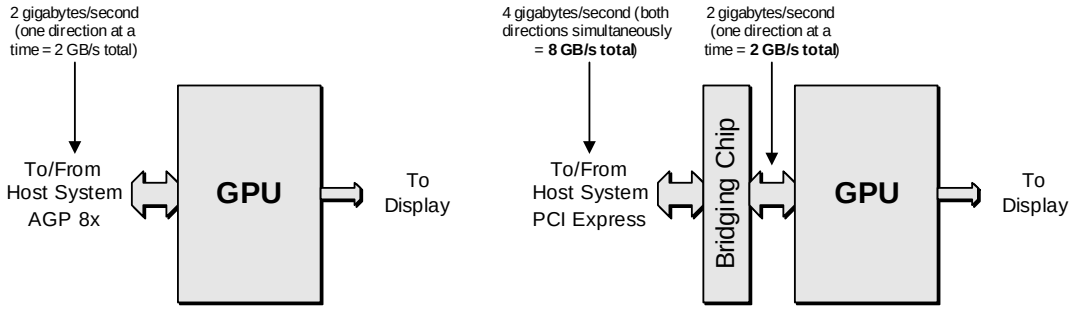
As another industry first, 3Dlabs has also released an open source version of the OpenGL Shading Language compiler front-end to stimulate creativity

Developer
Fort Collins, CO,
m, Randi Rost, is
Shading
Language book, published by Addison Wesley, ISBN: 0-321-19789-5.

Depending on their requirements, some devices will be able to make do with a single lane. One lane can support up to 250 MB/s of real data communications in both directions simultaneously, which equates to a total bandwidth of 500 MB/s. (By comparison, an AGP interface can pass data in only one direction at any particular time.) Some devices may require two or four PCI Express lanes to provide additional bandwidth, while today's high-end graphics subsystems employ 16 lanes. Known as 16x PCI Express, such an implementation provides a data bandwidth of 4 GB/s in both directions simultaneously, which equates to a total bandwidth of 8 GB/s.

There are several reasons why graphics subsystems may wish to pass data back to the host system. As a simple example, consider the case where the graphics subsystem is being used to merge a real-world video signal with computer-generated imagery (CGI) for a DCC application. In this case, in addition to presenting the results on a display device, it may be required to bring those results back to the host system to be stored for future processing.

Unfortunately, some vendors of graphics subsystems are jumping on the PCI Express bandwagon by using a front-ending technique with an existing AGP 8x device and bridging chip (Figure 5).



(a) GPU with internal AGP 8x interface

(b) GPU with bridging chip to PCI Express

Figure 5. Using a PCI Express bridging chip does not increase the performance of an AGP 8x-based GPU

This can lead to specmanship at its worst, because it allows these vendors to claim that their subsystems have the capability of communicating with the host system at the PCI Express bandwidth of 4 GB/s in both directions simultaneously. While theoretically true, there is no practical use for this bandwidth, because the downstream (GPU) end is limited to the AGP 8x interface, which can support only 2 GB/s in a single direction.

In those cases where one requires the bandwidth of a PCI Express interface, the graphics subsystem should be capable of taking full advantage of this interface.

Wildcat Realizm VPU-based Solutions

With over 150 million transistors, the Visual Processing Unit (VPU) featured in 3Dlabs' new Wildcat Realizm technology is quite simply the most advanced and highest-performance graphics device in the world at this time.

3Dlabs pioneered the development of OpenGL Shading Language and designed their VPU with OpenGL Shading Language in mind. The Wildcat Realizm VPU fully addresses OpenGL Shading Language requirements by combining massive parallelism with the most sophisticated programmable vertex and fragment shaders available (Figure 6).

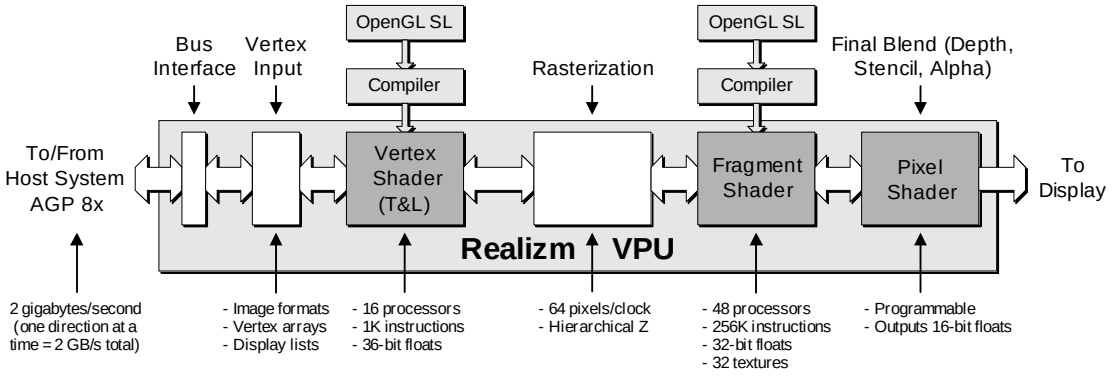


Figure 6. The 3D graphics pipeline in the context of a VPU-based architecture

The VPU's Vertex Shader combines a vector processor capable of operating on 16 vertices simultaneously with a shader containing up to 1 K instructions. The resulting data is handed over to the VPU's rasterization engine, which can process 64 pixels-per-clock.

This rasterization engine also offers the unique capability of Hierarchical Z-buffer depth culling. Objects in the scene are evaluated to determine if they are hidden by (located behind) other objects. If the objects are hidden, they are discarded from the graphics pipeline much earlier than is typically the case. This culling saves time by not processing pixels that will not be displayed. The resulting efficiency is fantastic and significantly boosts the overall performance of the system.

Following rasterization, the data is handed over to the fragment shader. This comprises 48 independent multi-threading processors that can operate with 32-bit floating-point accuracy on 48 pixels simultaneously. Due to the fact that fragment shading algorithms are invariably much more complex than their vertex shading counterparts the VPU's fragment shader supports 256 K instructions. It is important to note that the 1 K and 256 K values quoted for the VPU's vertex and fragment shading units are *real* machine code instructions. In another example of "specmanship", some vendors multiply the maximum number of times a program might go around a loop by the number of instructions contained within that loop and use this value to report inflated instruction-count capabilities.

In addition to incredible processing power, a Wildcat Realizm VPU-based graphics subsystem is capable of addressing up to 512 MB of physical GDDR3 memory. This is typically used as the working set for the very much larger 16 GB of unified virtual memory (used for the frame buffer, storing textures, etc.). This is extremely useful for applications that employ large numbers of large textures, because only visible elements need to be loaded into the physical memory while the page-fault mechanism allows the VPU's memory to behave as an L2 cache. (In many cases, the host system's memory may not be much larger than the 512 MB on the graphics card.)

When it comes to outputting data, a VPU can be used to drive two single-link DVI outputs, each of which can accommodate a digital or analog display device (Figure 7a). Alternatively, a VPU can drive two, dual-link DVI outputs, thereby providing the capability to handle much higher resolutions and refresh rates (Figure 7b).

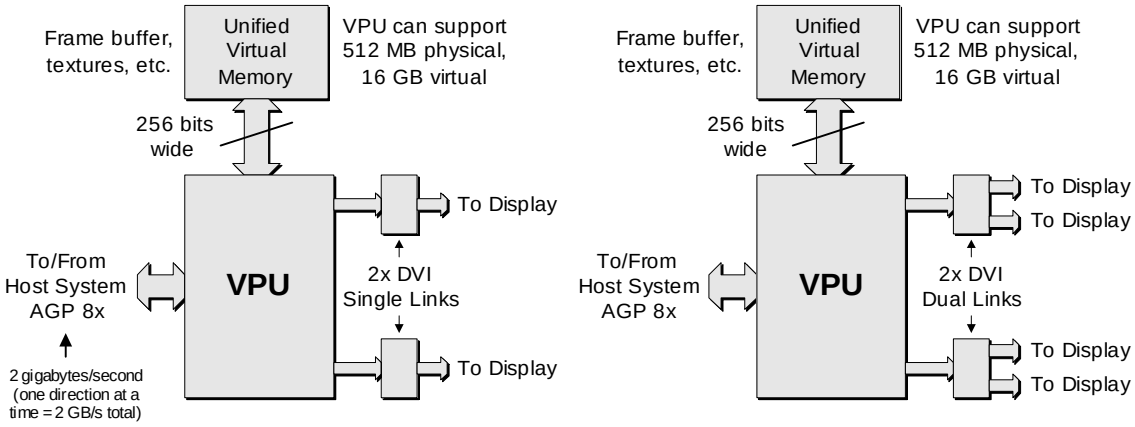


Figure 7. VPU-based systems address high and ultra-high-end AGP 8x implementations

Consider the 9.2 Megapixel (3,480 x 2,400) flat-panel displays that are used for extremely high-end visualization applications such as DCC, engineering, medical imaging, and geophysical research. Such a display can accept one single-link DVI input, two single-link DVI inputs, one dual-link DVI input, or two dual-link DVI inputs. Even with a graphics subsystem as advanced as a Wildcat Realizm VPU-based unit, the awesome resolution of this display means that one single-link DVI connector would provide a refresh rate of only 12.5 Hz, while two single-link DVIs could achieve a maximum of only 24.5 Hz. However, a VPU equipped with two dual-link DVI outputs can drive such a display at a very respectable 50 Hz.

As was noted earlier, the VPU's vertex shader works with 36-bit floating-point values, while its fragment shader works with 32-bit floating-point values. The VPU directly outputs 16-bit floating-point values, making it the first device to be based on a 3D graphics pipeline that is truly floating-point all the way from the input vertices to the final displayed pixels. The result is that VPU-based graphics subsystems provide truly stunning image quality and realism that can more than satisfy the requirements of high and ultra-high-end AGP 8x-based host systems.

Wildcat Realizm VSU/VPU-based Solutions

For those applications that demand the ultimate in graphics processing, 3Dlabs' new Wildcat Realizm technology can augment the power of one or two VPUs with an additional processing engine called a Vertex/Scalability Unit (VSU). Comprising more than 130 million transistors, a VSU supports a full 16x PCI Express interface and it doubles the number of vertex processors available in a VPU-only configuration. A VSU can be used to drive one or two VPUs, where *each* VSU-to-VPU interface has a bandwidth of 4.2 GB/s (twice that of an AGP 8x interface).

First, consider a configuration comprising a VSU driving a single VPU (Figure 8). In this case, the VPU's vertex shader is automatically disabled and the two vertex shaders in the VSU take over this portion of the processing, thereby providing a much greater vertex processing capability than a VPU in isolation. The 4 GB/s incoming data bandwidth of the 16x PCI Express interface means that the 2 x 16 = 32 processors in the VSU's dual vertex shaders are never starved of data. As well, the fact that the 16x PCI Express interface is a fully integrated element in the VSU, as opposed to using a bridging chip, means that an additional 4 GB/s outgoing data bandwidth is simultaneously available to return information to the host system (for example, the results of merging a live video feed with CGI.).

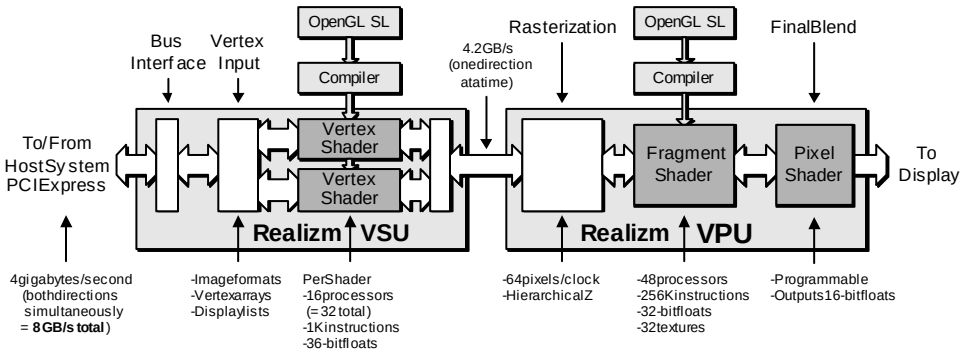


Figure 8. The 3D graphics pipeline in the context of a VSU-to-1xVPU-based architecture

Of course there is little point in having the VSU's phenomenal vertex processing capabilities if the results have to pass through a bottleneck, such as an AGP 8x port, to reach the rasterization and fragment shading engines in the downstream VPU. Thus, the VSU-to-VPU interface supports a bandwidth of 4.2 GB/s, which is *twice* that of an AGP 8x port.

The VSU also has an interface to 128 MB of GDDR3 DirectBurst™ memory. This is used to implement 3Dlabs DirectBurst technology, which utilizes the internal architecture of the host processor to its fullest extent (Figure 9).

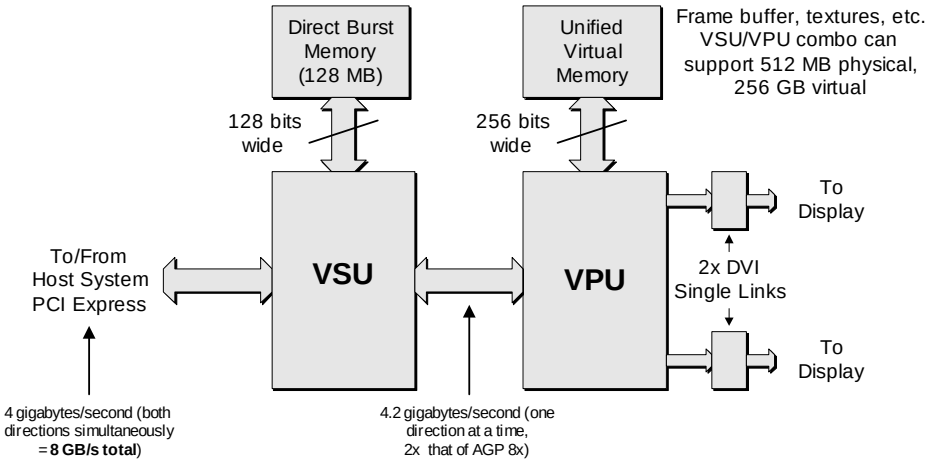


Figure 9. When combined with a VSU, a VPU can support up to 256 GB of virtual memory

In this case, instead of constantly updating a vertex buffer in the host system's main memory, vertex data from the 3D graphics application is stored in a *burst buffer* located in the processor itself. As soon as the processor's burst buffer is full, this data is streamed directly to the VSU's DirectBurst memory. Unlike conventional graphics subsystem architectures, which require multiple accesses to the main memory (a processor read, a processor write, and a DMA read), Wildcat Realizm DirectBurst technology employs only a single processor-to-main-memory read. This optimizes the system's memory bandwidth requirements and balances the 3D graphics pipeline to achieve optimal system performance.

Finally, for those applications that demand the ultimate in graphics processing power and performance, a Wildcat Realizm VSU can be used to drive two VPU devices (Figure 10).

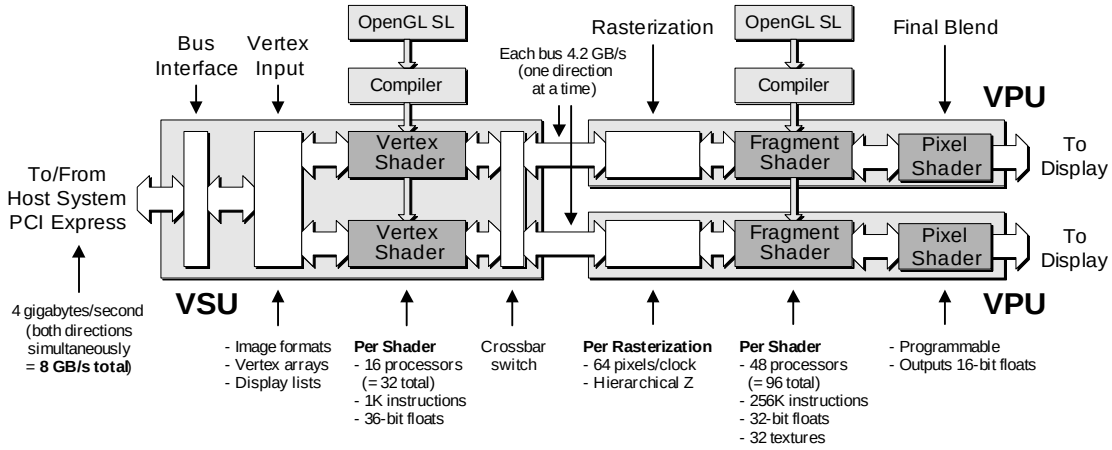


Figure 10. The 3D graphics pipeline in the context of a VSU-to-2xVPU-based architecture

In this case, the (2 x 16 =) 32 processors in the VSU's dual vertex shaders are now complemented by (2 x 48 =) 96 processors in the dual VPU fragment shaders. Once again, the combination of a VSU with two VPUs means that the VPUs are now capable of addressing up to 256 GB of virtual memory. B*oth* VSU-to-VPU interfaces support a bandwidth of 4.2 GB/s, thereby providing an aggregate data bandwidth of 8.4 GB/s, which is *four times* that of an AGP 8x port (Figure 11).

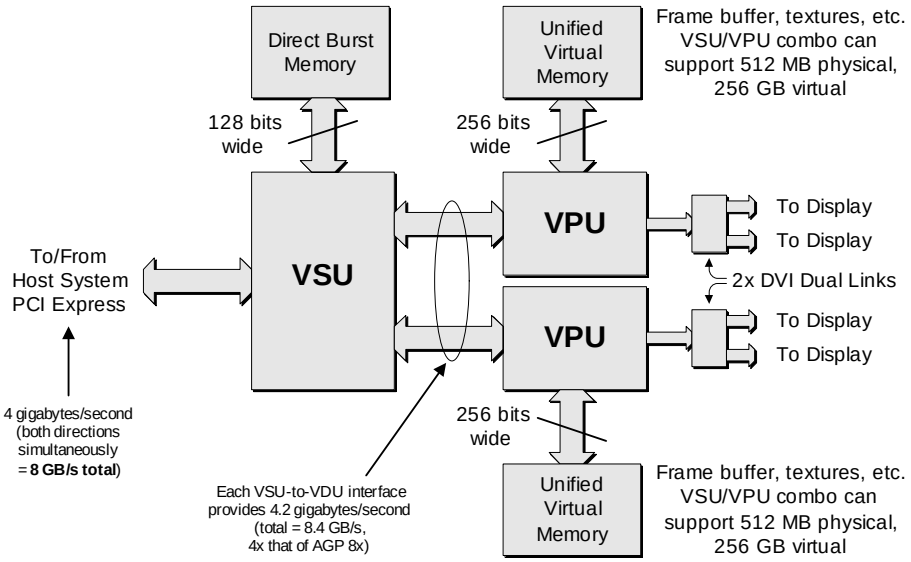


Figure 11. The VSU-to-2xVPU data bandwidth is an incredible 8.4 GB/s

One very important point concerns the way in which the VSU uses its crossbar switch (shown in Figure 10) to partition data and hand it over to the two VPUs. The idea here is that a 3D scene may contain some areas that encompass an extreme amount of detail, while other areas may include relatively little data that requires processing. As an extreme case, consider a scene in which the left-hand side contains the bulk of the objects in the scene, while the right-hand side contains only a few scene elements.

In such an eventuality, if one VPU were to be presented with the data relating to the left-hand side of the screen and the other VPU was placed in charge of the right-hand side, the result would be for the first VPU to become overloaded and to slow things down. Thus, the VSU actually partitions the display area into a checkerboard pattern of relatively small blocks. If we visualize

this checkerboard as comprising white and black squares, then the VSU passes all of the display areas corresponding white squares to the first VPU and all of the areas corresponding to the black squares to the second VPU. The resulting load balancing allows the entire system to achieve optimum performance.

Genlock, Framelock, and Ratelock Capability

Visual computing applications often need to synchronize their displays to an external source, where this feature is called Genlock. Also, visual computing applications often need to use multiple displays, in which case it is extremely important that the system treats all displays as a single virtual canvas. Referred to as Framelock or Framesync, this includes locking both the buffer swaps and the display refreshes.

In the case of the new Wildcat Realizm technology Genlock and Framelock capabilities can be achieved based on both the VPU and the VSU/VPU combination-based graphics subsystems. With regards to Genlock, the technology supports both the traditional bi-level synchronization pulses featured in conventional NTSC, PAL, and SECAM systems along with the latest tri-level synchronization scheme favored by today's high-definition (HD) display devices.

The Wildcat Realizm technology also supports the Ratelock feature, in which the software graphics application instructs the hardware graphics subsystems as to the minimum swap period it is prepared to tolerate. If one of the graphics subsystems detects that it cannot complete the rendering of a frame in the required time, it simply discards that frame and moves on to the next one. This allows the system as a whole to maintain the required frame rate across the single virtual canvas.

Summary

The VSU and VPU devices featured in the new Wildcat Realizm technology maintain 3Dlabs' position as an innovator and industry leader in professional graphics. VSU and VPU-based graphics subsystems can provide previously unattainable levels of quality and performance.

For example, at the time of this writing, the highest reported performance score for the industry-standard ViewPerf UGS benchmark is 45 frames-per-second (FPS). By comparison, Wildcat Realizm technology will more than double this performance and set a new industry standard.

Finally, as incredible as Wildcat Realizm technology is, it is important to note that 3Dlabs sees this technology as just a step along the path to the future. Thus, graphics accelerators based on Wildcat Realizm technology are not the final word in graphics subsystems, they will simply be the best that are available.