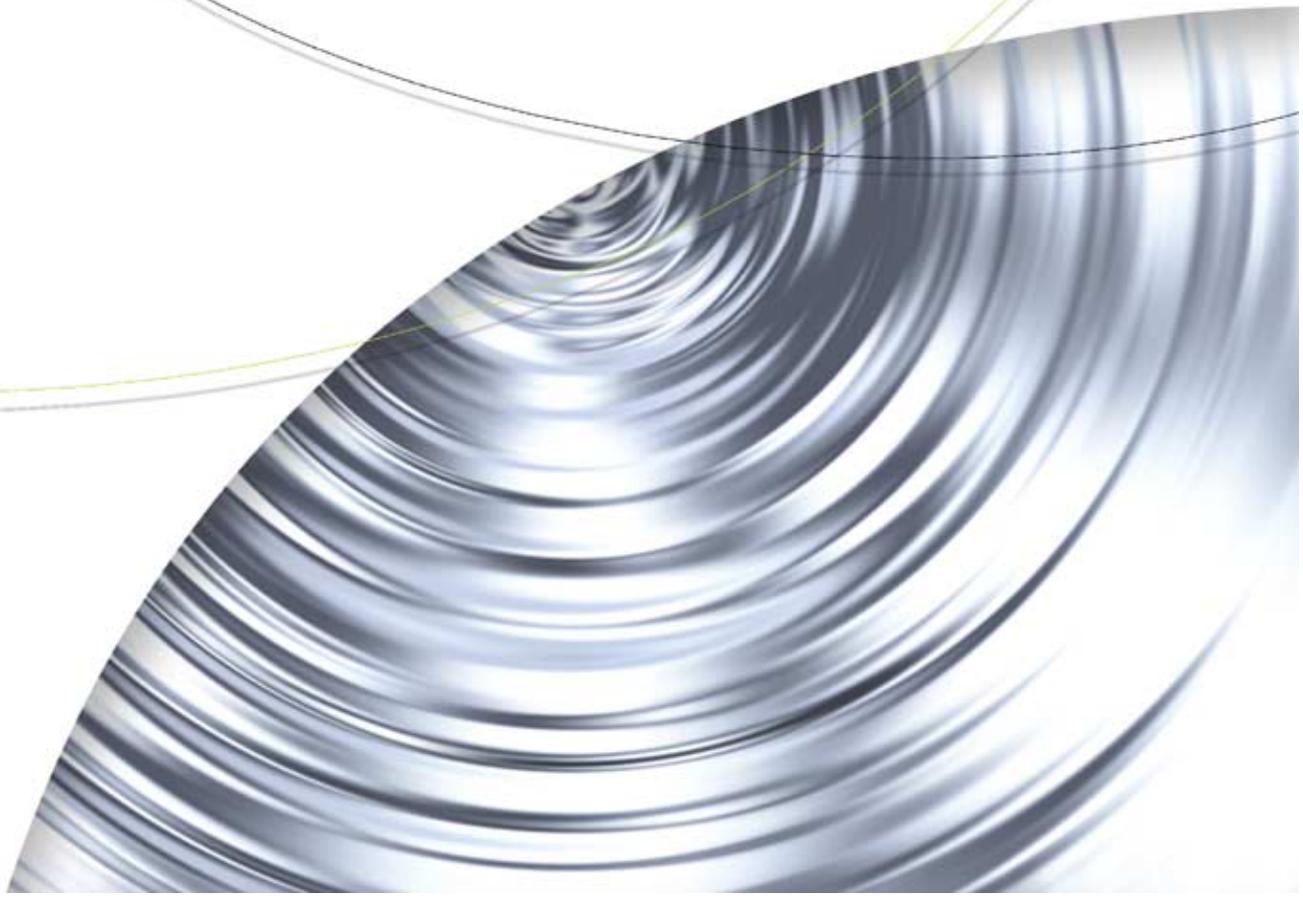




Technical Brief

NVIDIA Quadro4 Lightspeed Memory Architecture II

Breakthrough Memory Design for
Industry-Leading Workstation
Performance





Quadro4 Lightspeed Memory Architecture II (LMA II)

Delivering quality features coupled with high performance to meet the demanding needs of professional application users is a fundamental requirement for workstation graphics. In recent years, significant developments have yielded incredible benefits as well as created serious price-performance discontinuities. Relentless development of key technologies ensures this trend will continue. As workstation graphics moves to these new levels, one area in particular plays an increasingly important role in enabling quality features and high performance. That area is graphics memory bus bandwidth.

NVIDIA's Lightspeed Memory Architecture™ II (LMA II) comprises several revolutionary architectural advances that dramatically improve the memory bandwidth efficiency. This allows the Quadro4 Graphics Processing Units (GPUs) to deliver an unprecedented set of features and set new standards in performance. This technical brief describes the key features of LMAII, explaining how they relate to professional applications and provide significant benefits to the workstation user.

The Importance of Memory Bandwidth

In the same way a workstation's CPU is co-dependent with main memory, GPUs rely heavily on graphics memory. A typical workflow of a professional application will result in a continuous flow of high-bandwidth transactions through the memory interface as both graphics data and commands are transferred between elements of the GPU and graphics memory. Examples of such transactions include command buffer transfers, pixel reads and writes, as well as texture fetches.

During a typical workflow, the professional user probably has little concern for neither the nature nor specifics of the transactions through the GPU memory interface. It's clearly more important that they spend their time and creative energies on their specific task at hand. Unfortunately, low memory bandwidth and inefficient design result in the memory interface becoming a significant bottleneck during graphics activities, which in turn yields poor graphics performance and low visual quality. Clearly these issues are of major concern to the professional workstation user.

The Relationship Between Memory Bandwidth and Professional Application Performance

To create the high visual quality expected by workstation users, professional applications typically take advantage of the OpenGL pipeline. In their typical workflows, workstation users rarely need to know the specifics of how the pipeline is implemented. However, they clearly do see the resulting pixels. In generating these pixels, modern graphics architectures read and write to various buffers in graphics memory. These buffers hold many data including graphics commands for the GPU; color data representing the red, green, blue and alpha (RGBA) components of the pixel color; as well as the Z- values for depth. After primitives are transformed, lit and rasterized, the color and Z-data stored in the color, and Z- buffers for that pixel location are read and used to generate pixel color values. A new color and Z-value is calculated by comparing Z-values to determine whether the new pixel is seen, as well as taking into account any blending, defined by alpha values, with any previous color values. Once the new color and Z-values are calculated, the new data must be written to the memory. Using this description of the OpenGL pipeline we can start to illustrate the importance of memory bandwidth through the following example:

The minimum color and depth precision for professional workstation graphics is 32-bit color and 24-bit Z. Since memory accesses are partitioned on 4 byte boundaries, the Z-buffer is usually combined with an 8-bit stencil buffer thus rounding the total bytes allocated per-pixel to 8 bytes. Each per-pixel memory transaction, therefore, represents 16 bytes of data per pixel:

$$\begin{aligned} \text{Read: } & 32\text{-bit color} + 32\text{-bit Z/Stencil} = 8 \text{ bytes to Read, } + \\ \text{Write: } & 32\text{-bit color} + 32\text{-bit Z/Stencil} = 8 \text{ bytes to Write,} \\ & = 16 \text{ bytes total} \end{aligned}$$

For professional applications, interactivity is usually considered to be a minimum of 12 frames per second. However, the desirable goal is 60 frames per second, which corresponds to the limit above which human eyes cannot discern changes in position between frames. This is particularly important in Digital Content Creation applications when computer graphics are combined with video playback. In a similar way to frame rate requirements, professional applications usually work at resolutions of at least 1280x1024 pixels.

Given these requirements the graphics memory bandwidth consumed to display a smooth shaded model that fills the entire screen is:

$$1280 \times 1024 \times 16 \text{ bytes} \times 60 \text{fps} = 1.26 \text{MB/sec.}$$

Clearly this assumes that every pixel is changed or written to every frame. For the professional Computer Aided Designer or Digital Content Creators this may not be true some of the time, however, during a typical workflow, models or scenes

usually do to fill the entire graphics window for a significant portion of the time. In such situations the bandwidth assumptions are valid.

Current workstation graphics performance now allows product designers to visualize entire assemblies and gain greater understanding of how the entire product will look and function. In a similar way, professional artists typically utilize many high quality visual effects combined with complex underlying geometry in order to be able to create realistic and convincing scenes. At first sight, it may appear that both of these capabilities are somewhat distanced to memory bandwidth, however, in reality the connection is much closer than would be expected.

For the professional MCAD application user interactively viewing a large assembly, at any particular viewing position, it is very likely that many components will be obscured by other components. In graphics terminology, depth complexity refers to when parts of a model or scene obstruct other parts of the same model or scene. The more parts of a model or scene that overlap, the higher the depth complexity. As depth complexity increases, it linearly increases the demands on graphics memory bandwidth. This is because when one component or face lies in front or another, pixel color and Z values must be calculated and compared for both faces before the graphics hardware can determine which particular pixels are obscured. Obviously, this forces a complete read and write of any affected pixels that correspond to overlapping surfaces. Apart from performing unnecessary work on the discarded pixel, this clearly also consumes valuable graphics memory bandwidth.

Unfortunately the aforementioned bandwidth calculation doesn't factor in depth complexity. In reality, most models and scenes have a depth complexity of at least two, since there is always a front and a back regardless of what angle they are viewed at. Still, a depth complexity of two is probably a little low for typical products and scenes, and is very low for large complex CAD assemblies and very detailed scenes. Trying to define a single depth complexity that covers all situations, however, is a bit like trying to decide how long a piece of string is. A value of three (3) is considered to be reasonably representative, so taking this into consideration the bandwidth requirements become:

$$1280 \times 1024 \times 16 \text{ bytes} \times 60 \text{ fps} \times 3 = 3.78 \text{ GB/sec.}$$

To illustrate where these bandwidth requirements may occur in typical workflows, figure 1 shows a screenshot from Dassault Systems' CATIA V5 application where the engine assembly used in the CATIA Solutions 2001 benchmark occupies the entire CATIA V5 graphics window:

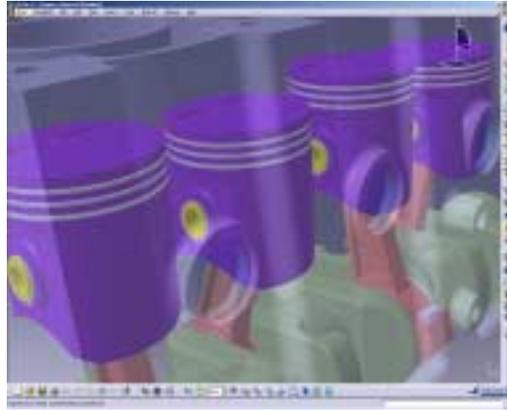


Figure 1. Screenshot of Dassault Systems' CATIA V5 showing the engine assembly used in CATIA Solutions 2001 benchmark. (For more details see www.catiasolutions.com)

In both the professional CAD and DCC markets, texturing is frequently used to increase realism of products and scenes. To provide rich image detail and increase visual quality, texturing typically uses trilinear filtering. Instead of approximating a pixel's color value by selecting the nearest value in the texture image, tri-linear filtering determines a pixel color value by interpolating between surrounding pixels in X and Y directions, as well as between different copies of the texture image scaled to different resolutions (mip-map levels). This dramatically improves overall visual quality and also noticeably reduces harsh changes in color values as textured objects are rotated in three dimensions. To further improve realism, DCC applications will often apply multiple textures to the same object to simulate real world effects such as spotlights, as well as complex materials and decals.

It's reasonably intuitive to expect that using textures will impact graphics memory bandwidth. To generate a pixel's color value when an object is textured, the appropriate pixel in the texture image (texel) must be retrieved from the same memory that is used to also store the Z- and color buffers. When using trilinear filtering, eight texture samples per pixel are used to determine the final color value. Assuming that every pixel is also textured, using the above example increases the data transferred for each pixel by 32 bytes. Remember: each pixel is derived from eight texels, which in turn are each defined by four bytes (32 bits). Multiple textures clearly scale this further by the number of textures applied. Since all GPUs offer some level of texture caching, it would be reasonable to assume that for professional applications using multiple textures, some degree of texture caching balances the instances where pixels have multiple textures. As a consequence, the bandwidth requirement becomes:

$$1280 \times 1024 \times (16 \text{ bytes} + 32 \text{ bytes}) \times 60 \text{ fps} \times 3 = 11.32 \text{ GB/sec.}$$

To show an example in a typical workflow where these bandwidth requirements may occur, figure 2 shows a screenshot from Discreet's *3D StudioMax* application where a trilinear textured scene occupies the entire graphics window:



Figure 2. Screenshot from Discreet's 3D Studio Max application showing textured scene.

Some professional workstation users use Full-Scene Antialiasing (FSAA) to further enhance image quality. When using FSAA, each pixel is derived from a combination of sub-samples corresponding to the degree of sub-sampling. It's reasonable that rendering a scene with 4x FSAA would increase the memory bandwidth by a factor of four. Applying this to the above calculation yields:

$$1280 \times 1024 \times (16 \text{ bytes} + 32 \text{ bytes}) \times 60 \text{ fps} \times 3 \times 4 = 45.3 \text{ GB/sec.}$$

The above calculation clearly shows that FSAA has a dramatic impact on memory bandwidth demands. In a similar way, all the calculations have used a typical screen resolution of 1280x1024 pixels. However, increasingly larger resolutions such as 1600x1200 are being used with professional applications, and in those cases the memory bandwidth requirement would increase to a mindnumbing 66.3GB/sec.!

Given the fact that the fastest graphics memory interfaces available today only provide about 10.3GB/sec. of peak bandwidth, and accepting that the above example makes some assumptions, it's very clear that memory bandwidth is a critical factor in workstation graphics performance for professional applications. Simply rendering a textured scene that fills the entire graphics window can result in memory bandwidth requirements close to the limits of current memory technology. Using higher screen resolutions combined with visual quality enhancements such as 4x FSAA pushes bandwidth requirements well beyond the capability of current memory technology, and as a consequence, results in reduced performance. Professional applications frequently use other techniques such as multipass rendering, as well as features such as off-screen buffers, all of which place further demands on the memory subsystem. Memory bandwidth, therefore, can be a

significant limitation for professional applications to deliver high quality, full-featured models and scenes with high performance.

NVIDIA Lightspeed Memory Architecture II: Breakthrough Memory Design for Industry-Leading Workstation Performance

The NVIDIA Lightspeed Memory Architecture II (LMA II) implements many patent-pending technologies to improve the efficiency with which GPUs render pixels. These technologies include a crossbar-based memory controller; Quad Cache memory caching subsystem; lossless Z-buffer compression; a visibility subsystem; fast Z-buffers clearing, and auto pre-charge.

Crossbar Memory Controller

As we have previously demonstrated, 3D graphics depends heavily upon memory bandwidth. It's no surprise, therefore, that the crux to improving performance lies with the memory controller architecture, which is incorporated into the GPU. The next most significant component is the local graphics memory itself. As previously mentioned, the local graphics memory holds various data and buffers: the frame buffer, Z/stencil buffer, texture data, geometry data and more. Because it's so heavily used and is subject to many demands, local graphics memory is typically the highest bandwidth memory system in a PC. It's critical, therefore, to make the most efficient use of the resources available.

Under most conditions, traditional memory controllers are reasonably efficient, and deliver more than 50% of the peak memory bandwidth from the frame buffer. The memory used on current workstation graphics cards is typically Double Data Rate Dynamic Random Access Memory (DDR DRAM). Since they also typically use a 128-bit memory controller, DDR memory designs transfer twice the bus width of data in a single access, with transfers occurring in 256-bit "chunks" of data.

While transferring large amounts of data, large data blocks may appear optimal. However, the data access dynamics that typically arise when rendering complex scenes with hundreds of thousands of polygons per frame make this assumption incorrect. For example, professional workstation applications define 3D data in terms of triangles. Since the models or scenes that yield these triangles are fairly complex, when they are viewed in their entirety the average size of a triangle is typically very small and sometimes only one or two pixels. A one-pixel triangle is

defined by 64 bits, comprised of 32 bits of color and 32 bits of Z/Stencil. In reality, however, the color and buffer and the Z/Stencil buffer are stored in separate areas of memory, and access to the color and Z information occurs as two separate 32-bit transfers. If memory controllers access information only in 256-bit “chunks” then in the case of a one-pixel triangle, most of the data retrieved in one access is unnecessary. As a result, the memory interface is effectively “wasting” 75% of the memory bandwidth for that single transaction.

The Quadro4 XGL family of GPUs implement a radical crossbar memory controller that is optimized for a fine granularity access pattern. The advantage of the crossbar memory interface is the flexibility to access information in 64-bit chunks, 128-bit chunks or 256-bit chunks, essentially whichever access pattern would be most efficient at that instant in time. By dynamically adjusting data transfers, the crossbar memory controller is able to load balance memory requests. This in turn ensures optimal access efficiency for frame buffer and Z/Stencil data, and keeps the GPU more fully utilized.

As shown in Figure 1, the Quadro4 700 XGL and 900 XGL implements four independent crossbar memory controllers. These crossbars memory controllers improve bandwidth utilization and transfer efficiency by a factor of four when compared with traditional architectures.

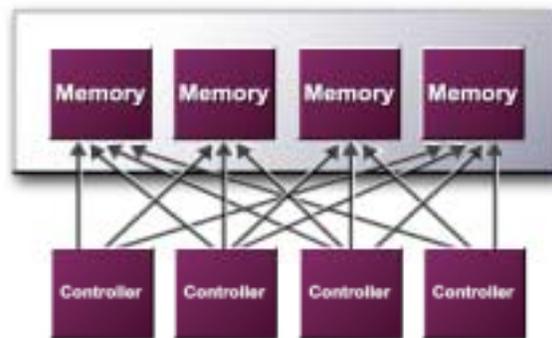


Figure 1: The Quadro4 XGL 900 memory crossbar interface with four independent controllers delivers 4X the memory bandwidth efficiency of standard architectures.

The Quadro4 XGL 500 GPUs implement two independent controllers, as shown in Figure 2, to provide a more economical alternative. This architecture still delivers significant performance benefits compared with non-crossbar implementations, improving data transfer efficiency by a factor of two.

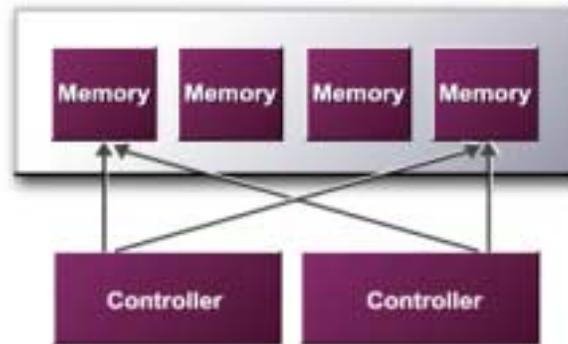


Figure 2: **The Quadro4 XGL 750 memory crossbar interface with two independent controllers delivers 2x the memory bandwidth efficiency of standard architectures.**

As the above bandwidth calculations illustrate, memory bandwidth is a major factor in professional application performance during typical workflows. The patented crossbar memory interface controller in the Quadro4 XGL family of GPUs ensures optimal memory bandwidth utilization and ensures maximum performance.

Quad Cache

Coupled with the crossbar memory controllers, LMA II also includes cache subsystems collectively known as Quad Cache. These caches provide special buffers to ensure data is queued and ready to be written to memory, as well as provide a place to buffer data being read from the memory.

The Quad Cache has four independent caches for primitive, vertex, texture and pixel data. Each cache is individually optimized for its respective data and each dramatically improve data transfer rates to and from graphics memory. Quad Cache helps the overall graphics pipeline performance because the data stored in these caches can be retrieved almost instantaneously instead of having to retrieve them from memory or even worse, recalculate the values. In the case of vertices, the cache provides a double benefit since they are stored after the transform calculations. Hence, when a vertex is retrieved from a cache, it doesn't need to be transformed again, ultimately providing a very significant performance gain. The combined result of Quad Cache is high-speed retrieval of key data that enables maximum performance through the graphics pipeline.

For the example above, the approximate bandwidth requirement for a professional application running full-screen at a resolution of 1280x1024, and drawing a complex scene trilinear-textured scene filling the graphics window is:

$$1280 \times 1024 \times (16 \text{ bytes} + 32 \text{ bytes}) \times 60 \text{ fps} \times 3 = 11.32 \text{ GB/sec.}$$

Assuming three out of every four texel fetches can be satisfied from within the texture component of the Quad Cache, the bytes transferred from memory to the GPU arising from texture fetches would reduce by approximately 75%. This may appear somewhat aggressive, however, consider that neighboring pixels can easily share a significant number of the same eight texels and a textured surface also typically covers a reasonable screen area in terms of pixels. If the texture cache is large enough, texel re-use will be significantly increased. The impact of this on the bandwidth requirement is significant. Using the above illustration, the bytes transfer from memory become:

$$1280 \times 1024 \times (16 \text{ bytes} + 8 \text{ bytes}) \times 60 \text{ fps} \times 3 = 5.66 \text{ GB/sec.}$$

As the calculation clearly shows, the bandwidth requirement is reduced to nearly a third and goes from being beyond the limit of traditional memory controller architectures to something actually achievable.

The above example here focused on texel data, however, the other elements of Quadro4's Quad Cache provide similar benefits for primitive, vertex and pixel data. The combined effect delivers significant performance benefits for professional applications, resulting in professional workstation users being able to create more realistic products and scenes with more complexity while maintaining interactivity.

Lossless Z Compression

The Z-buffer represents depth information and is used to determine whether pixels are visible given the current viewing orientation of the object or scene. When using professional applications, even simple models or scenes can result in most of the visible screen area having a depth complexity of greater than two. As the above example illustrates, this has a significant impact on memory bandwidth because the Z-value of every pixel has to be compared with the existing Z-value to determine the correct color. As a consequence, Z-buffer traffic is typically one of the largest "consumers" of memory bandwidth in a graphics system.

The Quadro4 XGL family of GPUs use an advanced proprietary form of lossless data compression, that compresses Z-values by a factor of 4:1. This reduces memory bandwidth arising from Z-buffer traffic by a factor of four. As data is read from, and written to, memory, the LMA-II's Z-compression/decompression engines operate in real time. And, because the compression is completely lossless, there is no reduction in image quality or precision.

The benefits of this can be illustrated using the above example. The bandwidth requirements of a professional application rendering a complex model, comprising lit and shaded triangles, that occupies the full 1280x1024 screen resolution were:

$$1280 \times 1024 \times 16 \text{ bytes} \times 60 \text{ fps} \times 3 = 3.77 \text{ GB/sec.}$$

Reducing the number of bytes used to define Z-values from 4 bytes to 1 byte, yields:

$$1280 \times 1024 \times 10 \text{ bytes} \times 60 \text{ fps} \times 3 = 2.36 \text{ GB/sec}$$

which represents nearly a 40% reduction memory bandwidth. This translates to either improved interactivity or increases in model and scene complexity, which directly translate to quantifiable performance benefits. Since Quadro4's lossless Z-compression is implemented in the GPU's hardware, the benefits are completely transparent and allow professional workstation users to enjoy immediate benefits in their everyday workflows without compromising any aspect of image quality.

Visibility Subsystem: Z-Occlusion Culling

When rendering triangles, traditional GPU architectures compare Z-values with the corresponding location in the frame buffer to determine the appropriate color and Z values for a specific pixel. While producing correct results, this method does require all pixels to be rendered, regardless of whether they are visible or not. Apart from the wasted GPU cycles performing the transforming calculations, this also leads to an additional color and Z-value read and write for every rendered pixel. Nearly all models and scenes rendered by professional application yield an average depth complexity of two. This means that for every visible pixel, the graphics processor is forced to do twice the work and access the frame buffer twice. Clearly this consumes valuable GPU cycles and graphics memory bandwidth rendering pixels that ultimately won't be seen.

To address this problem, Quadro4's LMA II implements two approaches to minimize the impact of high depth complexities, resulting in significantly higher performance for professional applications:

1. Z-occlusion Culling technology
2. Support for API Occlusion queries

At a low level, and transparent to all applications, Quadro4's LMA II Z-occlusion Culling technology determines whether a pixel will be seen or not. If the pixel is not going to be seen, then the pixel is not rendered, the frame buffer is not accessed, and valuable frame buffer bandwidth is saved. Even for models and scenes with relatively low depth complexities, this can yield significant improvements in memory bandwidth utilization and significant performance improvements for professional applications. As models or scenes become more complex and depth complexity increases, the performance benefits of Quadro4's LMA II Z-occlusion Culling similarly scale.

These benefits can be easily demonstrated by considering the SPECviewperf MedMCAD-01 test, which is specifically designed to closely represent graphics used in typical MCAD applications. Figure 3 shows a screenshot from this test and the model is based upon a real world product. As illustrated, there are a high number of occluded components, which is very typical for such models created with professional CAD applications.

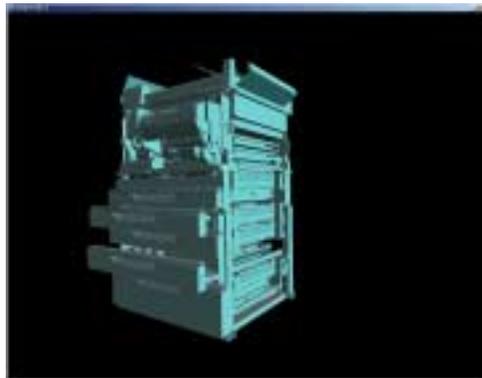


Figure 3. Screenshot from SPECviewperf MedMCAD-01 test12

The SPECviewperf MedMCAD-01 comprises several sub tests that evaluate performance using various graphics attributes. To provide a baseline for comparison of GPU performance, Figure 4 shows smooth shaded triangle performance, with two infinite lights for both the Quadro2 Pro and Quadro4

550XGL products. Figure 5 shows a comparison between the same two products for the SPECviewperf MedMCAD-01 tests 11 and 12:

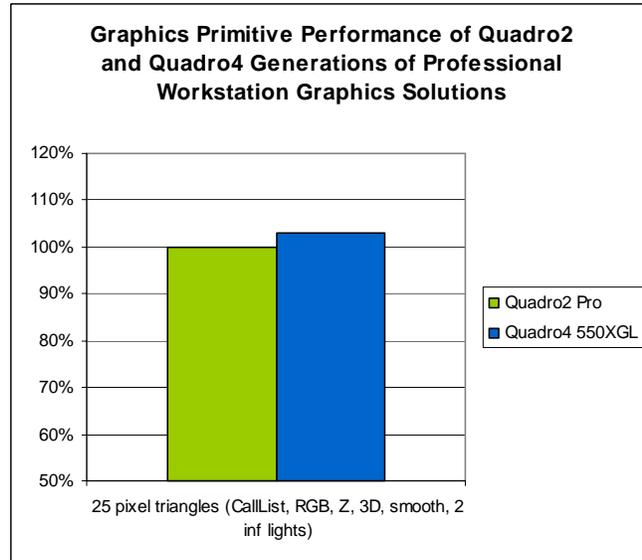


Figure 4. A comparison of line and triangle primitive performance between the Quadro2 and Quadro4 generations of professional workstation solutions.

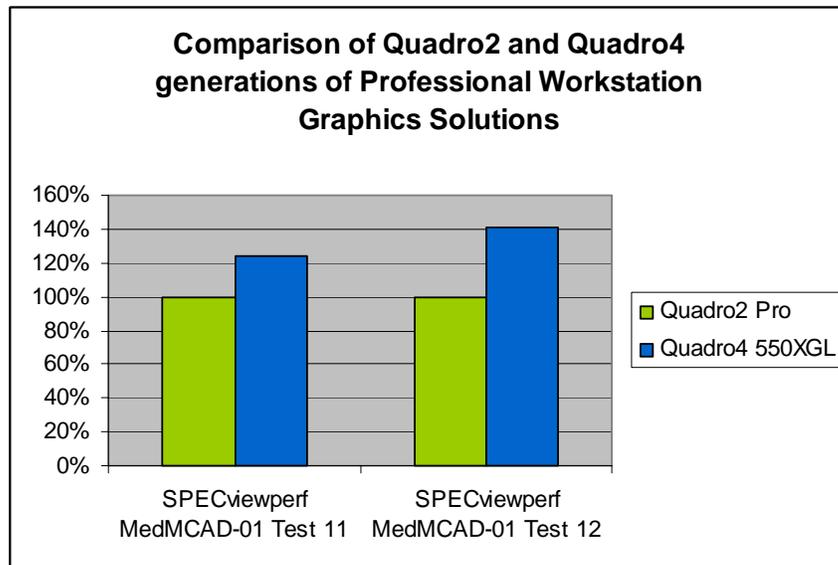


Figure 5. A comparison of SPECviewperf MedMCAD-01 tests 1, 2, 11 and 12 between Quadro2 and Quadro4 generations of workstation solutions.

Even though Figure 4 shows Quadro4 550XGL has comparable triangle primitive performance compared with Quadro2 Pro, its retail price is approximately half. Comparing results on the SPECviewperf MedMCAD-01 tests clearly show the benefits of Z-Culling since the Quadro4 550XGL demonstrates a 25-40% advantage over Quadro2 Pro. Like Quadro4's LMA II lossless Z-compression, Z-occlusion is integral to the Quadro4 GPU and completely transparent to professional applications. For workstation users, therefore, the performance benefits are immediate.

In addition to the low-level Z-occlusion included in Quadro4 LMA II, the Quadro4 900XGL product employs two higher level occlusion queries that professional applications can employ during their render loops for further performance benefits. These queries are accessed through OpenGL extensions.

The first of these queries is the HP_OCCLUSION_TEST extension. In essence, it provides a mechanism by which applications—within their main drawing loop—can first render the bounding box of a sub-component or scene element to test for visibility. The GPU then determines if any of the resulting geometry is visible, i.e. modifies the depth buffer, and if not, then the associated geometry can be skipped over. Any associated vertices for the skipped geometries do not need to be transformed, nor do the primitives need to be rasterized, conserving GPU cycles in the process. Skipping the geometry also conserves precious memory bandwidth by avoiding unnecessary frame and Z-buffer access. If the time taken to render the skipped geometry is larger compared with the time it takes to render the bounding box and read the occlusion result back, the technique can yield significant performance improvements. For large CAD assemblies, where components are entirely contained within other components, and for complex scenes where viewing orientations mean that many elements are completely hidden much of the time, occlusion culling can yield major performance gains. This in turn allows professional application users to work with larger assemblies and create more complex and realistic products.

In some situations the HP_OCCLUSION_TEST query may show some limitations, particularly when the time spent rendering the bounding box and reading the result query is longer compared than time saved by not drawing the geometry. In such situations the gain may be minimal. In fact, if little geometry ends up being occluded then the test clearly results in extra work for both the GPU and CPU negatively impacting performance. To overcome these limitations, Quadro4 900XGL provides the NV_occlusion_query extension which provides professional applications the ability to make more informed decisions on when to occlude objects, as help improve the efficiency of the render loop.

The advantage of the NV_occlusion_query extension is that the query can be issued asynchronously so that the application is then free to perform other tasks while waiting for the result. Unfortunately, the HP_OCCLUSION_TEST can often result in a “stop and go” approach during the render loop. The application renders the bounding box, waits for the result from the GPU, and only if the object is

visible is it drawn. This process then continues for the next object and so on. In some situations however, this can lead to an imbalance between CPU activity and GPU activity, and is largely inefficient. NV_occlusion_query allows the application to interleave the GPU and CPU work, take advantage of both and improve overall application performance. The second advantage of the NV_occlusion_query is that instead of returning a true/false value, it actually returns the number of pixels drawn in the bounding box. In situations where only a few pixels are visible this allows applications to make more informed decisions about whether to draw the object or not, and reduce the work on the both GPU and CPU.

Both the transparent hardware Z-occlusion Culling and the occlusion query extensions are key technologies that effectively amplify the bandwidth of the GPU while also dramatically increasing memory bandwidth efficiency. In many typical workflows using professional applications, each of these benefits can result in significant performance improvements—up to four times the performance of previous architectures in some specific situations. The example of the SPECviewperf MedMCAD-01 test shown previously produces a more realistic 40% performance gain, however, this is still a very significant benefit from one single aspect of the GPU's architecture. Applied to typical workflows in the MCAD and DCC markets, this equates to a very significant productivity increase. The occlusion query extensions provide by Quadro4 900XGL augments this benefit even further.

Auto Pre-Charge

A hidden 'tax' on memory bandwidth that is often overlooked are the various delays that arise due to page management inside the memory chip itself. To address this, Quadro4's LMA II has a special feature called Auto Pre-charge which reclaims much of the associated lost bandwidth. To understand the benefits of auto pre-charge, it's important to know that Dynamic Random Access Memories (DRAM) are organized in rows, columns and "banks", as a way to simplify the structure of the memory device and the memory interface. This organization allows access to data on very large memories with relatively few control wires. For example, a 4Mx32 memory equals 16MB of memory on a single memory chip. The benefit of this to the professional application user is that it significantly reduces overall costs for both memory and the GPU.

Unfortunately, there is a drawback: only the current row and column of an active bank can be accessed immediately. If the GPU wants to read or write from a different area of the memory chip, it must tell the memory to close the current bank and then activate the bank with new row and column settings. This process can take as much as ten DRAM clock ticks since a bank must be "pre-charged" after it is closed and once again before it can be reactivated. As a result, there may be situations where no data moves across the bus while the GPU waits for the memory

to get ready. This ten-clock penalty must be paid every time a row and column change is necessary.

To avoid the potential significant performance penalties that may arise, the Quadro4 GPUs have the ability to proactively tell the memory device to 'pre-charge' areas of the memory that are not currently in use, but are likely to be used in the very near future. In current memory terms, "near future" is measured in microseconds. When it subsequently needs to access those areas, the GPU only has to wait for the activation step, which varies by memory vendor and memory device, but is generally two or three clocks. Clearly, reducing access times from the order of 12-13 cycles down to two to three cycles is a very large, and in situations when accessing many separate locations is necessary, such as reading and writing frame and Z-buffer values during rendering, the benefits become particularly significant. Again, these benefits are completely transparent to professional applications, allowing workstation users to create and work with larger, more complex models and scenes with noticeably higher visual quality.

Fast Z-Clear

In order to provide a meaningful comparison, the Z-buffer must be cleared at the start of each frame when the model or scene is drawn. If the Z-buffer isn't cleared, then depth comparisons become inaccurate, artifacts would start to appear, and the display would quickly become meaningless. Unfortunately, clearing the Z-buffer means writing zero to all of the locations which takes time as well as memory bandwidth. To address this, the LMA II in the Quadro4 generation of GPUs includes Fast Z-Clear technology to minimize the time taken to clear the "old" data in the Z buffer. The bandwidth saving per-frame arising from fast Z-clears typically equates to approximately 10%. As part of Quadro4's LMA II, this again is transparent to professional applications although clearly beneficial to workstation users.

Conclusion

As the professional workstation market place continues to demand higher levels of performance and quality, the Quadro4 generation of workstation GPUs incorporates several industry leading technologies to meet these requirements. Quadro4's LMA II comprises several technologies aimed at removing inefficiencies in maximizing the utilization of memory bandwidth seen with more traditional graphics architectures. The sophisticated crossbar-based memory interface, Quad Cache, advanced lossless-Z compression, auto pre-charge, fast Z-clears and Z-occlusion culling, all maximize the utilization of precious memory bandwidth. The higher level application occlusion culling queries provide easy ways for applications to efficiently manage what geometry is drawn geometry, allowing model and scene complexity to be significantly increase without impacting performance.

The combined effect of Quadro4's LMA II technologies is to provide significant performance advantages for professional application users in all market segments including MCAD and DCC. In typical workflows, these performance advantages translate to both increased productivity and quality of product, which in turn yields competitive advantage. In this way, the Quadro4 generation of professional workstation GPUs set a new standard for workstation graphics.



Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication or otherwise under any patent or patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. NVIDIA Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

Trademarks

NVIDIA and the NVIDIA logo are trademarks of NVIDIA Corporation.

Microsoft is a registered trademark of Microsoft Corporation. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

Copyright NVIDIA Corporation 2002.



NVIDIA Corporation
2701 San Tomas Expressway
Santa Clara, CA 95050