



SiliconGraphics

Technical
Report

Power Series

280GTX
240GTX
220GTX
210GTX
120GTX
280S
240S
220S
210S
120S

Silicon Graphics designs, manufactures and markets computational systems used by engineers, scientists and animation professionals for design and analysis of 3D objects and for general-purpose technical computing. By employing RISC technology and proprietary VLSI components to provide both high performance computing and high performance graphics, Silicon Graphics continues to deliver the most powerful systems for engineering and scientific applications.

The POWER Series Systems

POWER Series™ systems are available in three major categories: 1) POWER Station™ servers which utilize advanced processor technology from Silicon Graphics and 2) POWER IRIS™ systems which couple that processing technology with Silicon Graphics' state-of-the-art GTX graphics subsystem and 3) POWER Center Servers and graphics systems provide the most expandable packaging for server and graphics applications. Combining 5 new proprietary parts with an innovative architecture allows the POWER Series to increase dramatically the compute power available in a deskside system.

Any Silicon Graphics IRIS 4D superworkstation may be upgraded to the POWER Series with the fully compatible 4D/GTX upgrade.

The CPU subsystem may be configured to include any one of the following:

- Two 16.7 MHz RISC processors and two floating point coprocessors.
- One 25 MHz RISC processor and one floating point coprocessor.
- Two 25 MHz RISC processors and two floating point coprocessors.
- Four 25 MHz RISC processors and four floating point coprocessors.
- Eight 25 MHz RISC processors and eight floating point coprocessors.

All POWER Series systems also include a fully semaphored, threaded UNIX™ kernel (IRIX™) and:

- 8-128 MB ECC memory
- 170 MB to 9.6 GB disk capacity

The GTX Graphics subsystem includes:

- 64 image bit-planes (including 24 color bit-planes and 8 optional alpha bit-planes, double buffered)
- 24-bit Z-buffer for hidden surface removal
- 4 bit-planes for overlay or underlay
- 4 window ID bit-planes (not user accessible)
- Access to 16.7 million colors in double-buffered RGB mode
- 1280x1024 screen resolution

Hardware support is included for:

- Pixel-blending capability for transparency effects
- Advanced lighting models
 - Multiple colored light sources (up to 8)
 - Ambient, diffuse, and specular lighting models
 - Phong lighting
 - Local viewer and infinite light source positioning
- Pan and zoom capabilities
- Multimode windowing environment
- Flat and Gouraud shading
- Anti-aliased lines
- High-Speed Graphics DMA

The POWER Series provides a level of processing capability never before achieved in a graphics superworkstation. Combining an advanced CPU subsystem with an enhanced version of Silicon Graphics' industry leading GTX graphics architecture, the POWER Series user has the power required by today's most demanding users. Specifically the 4D/280GTX system delivers 160 MIPS and 28 MFLOPS of sustained performance, 100,000 Z-buffered 4-sided, Gouraud shaded, Phong lighted, independent polygons per second, and a frame buffer access rate of 8 Million pixels/second. This new level of processing power is made possible by 5 new proprietary VLSI parts for the CPU as well as 7 proprietary VLSI parts in the graphics subsystem.

2.1 TECHNOLOGY ADVANCEMENTS

The sheer speed of the new GTX graphics architecture demanded high performance processing over a high bandwidth bus. A 3D graphics user requires more memory, more processing, and more I/O throughput than previously available. It was vital for Silicon Graphics to significantly enhance the computation portion of the system. The result is the POWERpath™ multiprocessing architecture, a high-power system design for the demanding user.

POWERpath Architecture

Coupling the world's fastest RISC microprocessors (the MIPS R2000/2010 and R3000/3010) with Silicon Graphics' proprietary chips and custom system architecture leads to a high efficiency, high throughput system. The goal of the architecture is to maximize the resources available to the user as needed. Using dedicated subsystems and buses for specific tasks (I/O, memory, graphics, etc.) avoids penalizing other functions of the system when using the full performance of any subsystem.

Proprietary Processors

Five new proprietary VLSI parts in the CPU subsystem and seven proprietary parts in the graphics subsystem made this new architecture possible and led to reduced cost with increased reliability and manufacturability.

SRAM

The latest RAM technology is used to provide zero wait-state access and large cache sizes for maximum cache hit rates.

ECC Memory

A new proprietary ECC chip provides "blow by" error detection for the memory subsystem. This allows the added ECC feature with no memory speed penalty or extra system cost.

PC Board

Silicon Graphics' own graphics technology was used in designing the four new CPU system cards. This allowed two processors to be placed on one card, lowering cost and size requirements.

2.2 CPU SUBSYSTEM HARDWARE

The fundamental design is a tightly coupled symmetric shared memory multiprocessing architecture. The CPUs in the 4D/210, 4D/220, 4D/240, and 4D/280 systems are the MIPS designed RISC processor, the R3000, coupled with a companion R3010 floating point chip. The CPUs in the 4D/120GTX and 4D/120S are the R2000 and its companion R2010 floating point coprocessor.

Figure 1 is a block diagram of the major components of the system. Several of the important buses of the system are shown in this diagram. The Sync Bus provides high speed synchronization between the main processors of the system in support of fine grain parallelism. The processor bus allows full speed access (zero wait state) to instructions and data from the individual first level caches of the main processors. The read and write buffers allow for the efficient flow of information between the processors and the main memory of the system. The second level data cache provides the additional bandwidth needed with such high speed processors to support an automatically consistent shared memory computing model. The MLink Bus supports protocols for consistent data sharing and high speed block data transfers among the processors, main memory, the I/O system, and the graphics subsystem.

2.2.1 Processor Bus

Each processor provides both an address and a data bus (known as the Processor Bus) that can support sustained data transfers at 8 bytes every clock cycle. Thus an eight processor system has a total processor-to-cache bandwidth of 1600 megabytes per second.

The first level caches, separate instruction and data, are each 64 kilobytes. The instruction caches are fed by a custom read-buffer that allows extremely fast access to sequential instructions. The data cache drives out through a custom write-buffer that buffers off the MPlink Bus.

These buffers also provide a convenient point for an asynchronous interface between the processors and the MPlink Bus. The R3000 processors run at 25 Mhz, asynchronous from the MPlink Bus. Thus the speed of the CPUs in the system can be increased over time without changing the bus or memory timings. This ensures that POWER Series buyers can upgrade later without replacing a large portion of the system.

Each second level data cache is 256 kilobytes in size, organized as 16 lines of 16 bytes each. (Second level caches in the 4D/120GTX and 4D/120S are 64 kilobytes. The 4D/210S and the 4D/210 GTX do not require a second level cache.) This cache provides block transfer capability for the MPlink Bus and provides the additional bandwidth necessary to keep all individual caches in a consistent state.

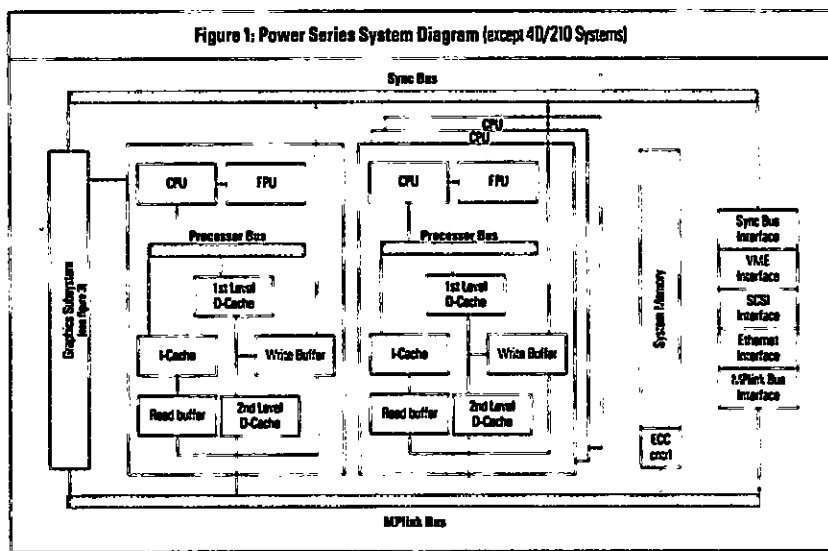
The second level data cache watches every transaction on the MPlink Bus and checks for transactions involving data in its data storage. This checking is performed by matching every address on the MPlink Bus with the addresses in the tag storage section of the second level data cache. The first level data cache is always a subset of the second level data cache so data consistency is guaranteed.

In addition, since all the caches are physical address caches rather than virtual address caches, there are no aliasing problems caused by mapping different virtual addresses to the same physical address. The difficult system level issues that arise when dealing with multiple virtual address caches are not present in this system.

2.2.2 The Sync Bus

The Sync Bus is designed for the synchronization needs of a multiprocessor supporting efficient fine-grained parallelism. It is implemented as a proprietary Silicon Graphics VLSI part. The goal is for a single application to be able to make efficient use of parallel processors even at the individual loop level, in addition to the kinds of larger-grained parallelism found in many system simulation applications and the even larger-grained parallelism found in the process structure of a UNIX™ system.

The Sync Bus provides 65,000 individual test-and-set variables. These variables are in a special part of the physical address space. They are addressed as memory and can be allocated to individual applications by the operating system. They are arranged 64 to a page and can be mapped into the virtual address space of an application. The operating system itself makes use of them to provide very fine grained locks for the control variables of the operating system.



The operating system is thus a very parallel, fully symmetric multiprocessing operating system. In other words, the Silicon Graphics version of UNIX V.3 is a well developed parallel processing application on the POWER Series—its speed demonstrates the power of this approach to high speed computing.

Because the Sync Bus can provide synchronization operations to applications with an overhead of only a few cycles, many programming and compiler techniques developed for vector processors are also suitable for this kind of parallel processor. For example, strip mining—the technique of taking a long vector and breaking it into a number of strips for use by a vector register—can be used by taking a long vector and breaking it into 8 strips, one for each processor.

The Sync Bus also provides for the distribution of interrupts from one processor to another, or from the I/O system to appropriate processors. The flexibility of the interrupt distribution system means that the operating system can provide scheduling algorithms that support the power of private caches rather than disrupt it.

2.2.3 The MPlink Bus

The MPlink Bus is a pipelined, block transfer bus that supports the cache consistency protocol as well as providing 64 megabytes of sustained data bandwidth between the processors, memory, I/O system, and the graphics subsystem. Because the Sync Bus provides for efficient synchronization between processors, the cache consistency protocol is designed to support efficient data sharing between processors. If a cache consistency protocol had to support synchronization as well as sharing, a compromise in the efficiency of the data sharing protocol would be necessary to improve the efficiency of the synchronization operations. With separate buses for each of these separate functions, each bus can be designed for optimal performance without compromise.

The cache consistency protocol used is sometimes called the Illinois protocol. Each second level data cache maintains the state values for each cache line. A line can be in one of four states: invalid, private read, shared read, or private write. If a processor writes into a shared read line, the processor must first invalidate other copies of that cache line before the write can be completed. Simultaneous writes into a shared read line by several processors will result in write misses in all the processors except the one which successfully acquired the MPlink Bus and issued an invalidate operation on it. In addition, any synchronization operation performed by a processor on the Sync Bus must not complete until all pending write activity by that processor is complete.

With these simple rules enforced by the hardware protocols of the Sync Bus and the MPlink Bus, efficient synchronization and data sharing are achieved with a simple shared memory model of parallel processing. The only data not visible to any processor are the data in another processor's registers. These sorts of invisible data are handled by the usual safeguards in modern optimizing compilers.

The physical structure of the MPlink Bus is 32 address lines and 64 data lines. An MPlink Bus transaction is six cycles in length although the last two data transfer cycles can overlap with the first two cycles of the next transaction, resulting in a sustainable data transfer rate of 64 megabytes per second in a very economical configuration. New data arrives in the last two cycles. Old data from a swapped cache line is carried in the middle two cycles. Addresses are transferred in the first two cycles. Bus arbitration is pipelined and doesn't add to the cycle cost of transactions.

2.2.4 I/O Buses

The POWERpath™ architecture also incorporates three separate I/O buses; VME, Ethernet, and SCSI (Small Computer System Interface). These buses operate independently to maximize the I/O throughput of the total system. The VME bus is supported via a proprietary SGI chip (DMAP) as a full functioning, full performance VME interface. VME support includes double and triple high Eurocard formats as bus masters and block mode transfer. The SCSI bus provides a low cost, standard interface for disk and tape. The built-in Ethernet provides access to TCP/IP and DECnet™ networks.

The IRIX™ kernel is designed to operate a symmetric shared memory multiprocessor efficiently and reliably. This is accomplished through a software architecture which allows each processor to independently execute kernel code. Independent execution allows full use of each processor by the process which is executing on it. Kernel data structures are protected via locks and semaphores, which allows data structures and I/O paths to be shared and used in parallel.

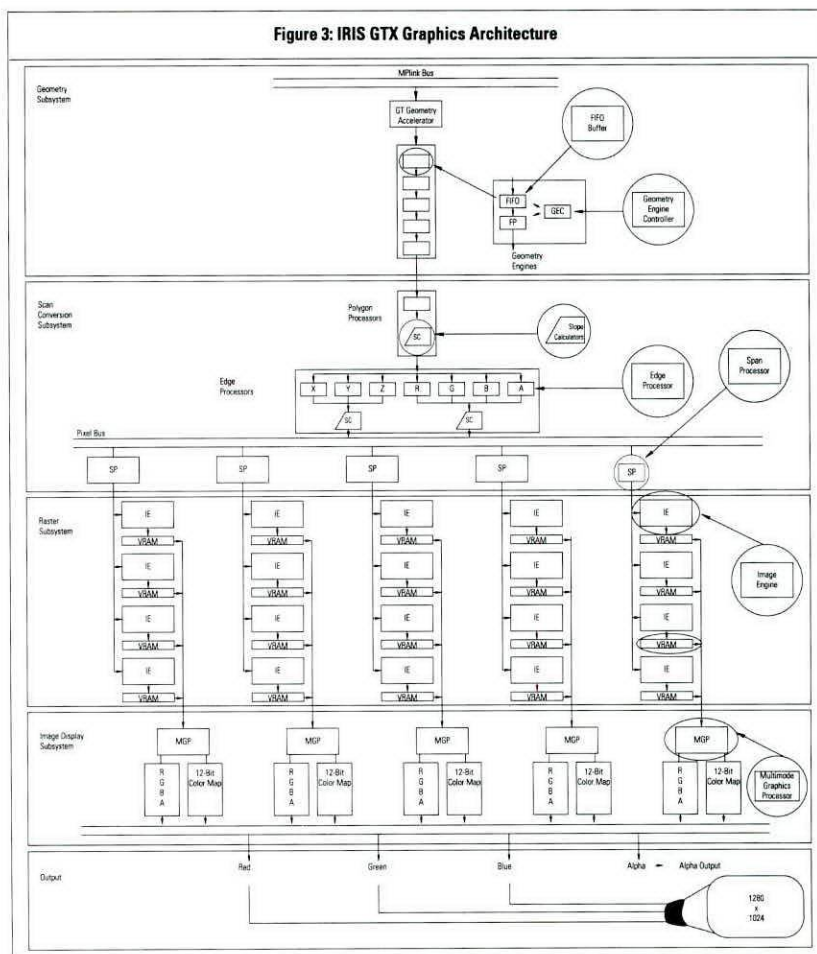
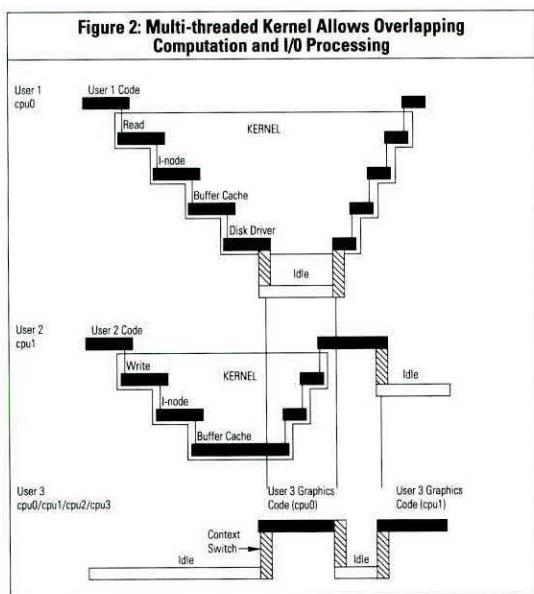
3.1 LOW LEVEL SYNCHRONIZATION

Low level synchronization is achieved through hardware-supported spinlocks. Each spinlock provides atomic access to a single-bit flag which indicates ownership of a resource. If the lock is held by one processor, other processors will busy-wait, looping on the lock until it is freed by the owning processor. This form of synchronization is very useful when the waiting time will be short, typically less than the time it would take to switch contexts.

If a resource is to be locked for a longer period of time, a semaphore is used instead. The semaphore queues a process for access to a resource and provides for the process to be unblocked when the owning process releases the resource.

3.2 DATA STRUCTURE ACCESS

All shared kernel data structures, such as inodes, buffers, or process table entries, are protected via some form of lock. This allows multiple processors to access the data structure in parallel without interference. This locking has been fine-tuned throughout the system. The use and format of shared data structures has been optimized to eliminate or minimize the amount of spinlocking required.



The key limiter of performance in a multiprocessor kernel is the contention that may occur for data structures. A fully configured IRIX™ kernel uses over 4000 spinlocks. Lock and semaphore metering are used to insure that these synchronization utilities are effectively utilized.

3.3 SELF-SCHEDULING PROCESSORS

The kernel maintains a single run queue for all processes. Each processor is self-scheduling. When a process blocks, that processor can immediately search the run queue for new work and dispatch it without interaction with other processors. Each processor is capable of executing any process which might be runnable, providing fast response and automatic workload balancing.

3.4 I/O HANDLING

The I/O subsystem of the kernel is constructed to allow each processor to independently access the drivers or support code. For instance, the buffer cache is multi-threaded. High performance drivers, such as the disk drivers, are fully parallelized, resulting in higher I/O throughput. A transparent driver locking scheme is incorporated to enhance portability of existing drivers.

3.5 REAL-TIME PERFORMANCE FEATURES

The trap and system call handling paths in the kernel have been highly optimized. Exception condition handling is optimized for the no-exception case, unlike most other UNIX based systems. Self-scheduling processors provide an additional level of response, since a processor can be dispatched to handle an event more quickly.

The system run queue has been reorganized as a sorted list of processes based on process priority. This provides $O(1)$ algorithmic performance for dispatching processes, which is extremely important in a self-scheduling system where multiple processors may be accessing the run queue simultaneously. Normal UNIX™ scheduling is $O(n)$, where n is the number of runnable processes.

3.6 PARALLEL PROGRAMMING SUPPORT

The kernel supports parallel programming by providing a low-level resource sharing model. Parallel programming is aided by getting the operating system out of the way and allowing processes to synchronize independently of the kernel. System calls are expensive in any system, and should be avoided if the programmer desires the best performance possible from his application.

The IRIX™ kernel introduces a new programming paradigm similar to threads or lightweight processes, with little of the additional programming burden caused by these schemes. This paradigm, called process share groups ("Beyond Threads: Resource Sharing in UNIX," Winter 1987 USENIX Conference), allows a process to share many of its resources, such as the virtual memory address space, file descriptors, user and group IDs, working and root directory and others. Within the kernel, virtual address sharing is accomplished by sharing the kernel memory management structures between the processes, and providing simple locking for those structures.

Since other resources may be shared as well, it is possible to construct many sophisticated applications, such as network servers or asynchronous I/O handlers, with a minimum of effort.

3.7 TRANSLATION LOOKASIDE BUFFER SYNCHRONIZATION

The translation lookaside buffer (TLB) is a cache of virtual to physical address translations that is used by the processor to speed virtual memory handling. This cache is managed entirely by software. Hardware synchronization is costly in terms of both dollars and performance. IRIX™ manages all TLBs in all the processors as a distributed resource. ("Translation Lookaside Buffer Management in a Multiprocessor System," Winter 1987 USENIX Conference Proceedings). TLB flushing is minimized through a sophisticated algorithm. The usual, low performance solution of TLB flushing on context switch is totally avoided.

3.8 GRAPHICS SUBSYSTEM SUPPORT

Support of the GTX graphics subsystem is integrated into the IRIX multiprocessor kernel. The graphics subsystem can be driven from any processor on demand. Access to the graphics pipe is achieved through direct mapping of the pipe to the user's address space. Management of graphics context switching is handled fully by the kernel.

The graphics subsystem is also capable of direct DMA to the frame buffer. This is supported in the kernel through direct mapping of the user's buffer for DMA operations, which avoids all data copying.

4.1 GRAPHICS HARDWARE ARCHITECTURE

The GTX graphics architecture is an enhanced implementation of the industry leading GT architecture from Silicon Graphics. Through its use of parallel processing technology and 50 proprietary graphics processors, the GTX architecture allows users to interact with complex 3-dimensional images in real-time, under direct input device control.

Graphics data are processed by four pipelined graphics subsystems before being displayed on the screen:

- Geometry Subsystem
- Scan Conversion Subsystem
- Raster Subsystem
- Display Subsystem

The Geometry Subsystem converts the 32-bit or 64-bit graphical data to screen space data. The Scan Conversion Subsystem then breaks down points, lines and polygons into pixel data which are then fed to the Raster Subsystem, where the Z-buffer removes the hidden surfaces. The Raster Subsystem also performs various blending functions on a pixel-by-pixel basis as the pixels are written to the image bit-planes, or frame buffer. The Display Subsystem then displays the contents of the frame buffer on the color monitor.

4.1.1 Geometry Subsystem

The geometry subsystem is connected to the CPU subsystem by the 64-bit MPlink Bus. This subsystem performs the specified transformations and lighting computations on the world coordinate geometric data before sending the screen-space data to the Scan Conversion Subsystem. The graphics processing in the Geometry Subsystem is performed in the Geometry Pipeline™, a serial configuration of proprietary processors capable of delivering 100 million floating point operations per second (MFLOPS) of graphics processing power.

Geometry Pipeline

The Geometry Pipeline™ processes a stream of high level graphics commands with five Geometry Engines (GEs), each of which is a proprietary processor capable of 20 million floating point operations per second (MFLOPS). Every Geometry Engine in the pipeline includes local memory for loadable microcode and data.

The first GE uses

- a 4x4 coordinate matrix stack to rotate, translate and scale 2D and 3D homogeneous coordinates,
- a 3x3 normal matrix stack to transform surface normals

The second GE performs lighting calculations. It supports up to eight point-source lights; material specifications including ambient, diffuse and specular reflectance; and lighting model information. The result of the lighting calculation is either a quad set of four 8-bit red, green, blue and alpha values (in RGB mode) or a single 12-bit color index (in color index mode).

The third GE clips coordinates to a 6-plane bounding box. A fast accept/reject clip-checking algorithm eliminates the need for complex clipping calculations in most cases. When complex clipping is required, the affected vertices are forwarded to the fourth GE, which performs the actual clipping calculation. The fourth GE also performs perspective division.

The fifth GE calculates depth-cued color values and clips color components to a maximum value before converting all coordinates to screen space integers.

4.1.2 Scan Conversion Subsystem

Screen coordinates sent from the Geometry Pipeline™ to the Scan Conversion Subsystem specify the vertices of points, lines and polygons. The Scan Conversion Subsystem performs the calculations required to reduce vertex data to individual pixels. Each pixel is assigned an X, Y and Z coordinate and an R, G, B and Alpha value. Color data are always linearly interpolated between vertices and between the edges of polygons.

The task of scan-converting polygons is performed in three stages, each performed by one of three specialized processors:

- Polygon Processor
- Edge Processor
- Span Processors

The Polygon Processor sorts the incoming vertices of polygons from left to right. The sorted vertices are then decomposed into the vertices of trapezoids before being used to calculate slopes and Z, R, G, B and alpha values.

The Edge Processor uses the vertex and slope information to compute X, Y, and Z coordinates and color values for each pixel that lies between the two vertices. The results of these calculations are used by the Edge Processor to calculate Z coordinates and color values for every pixel along each vertical span of 1024 pixels.

The Edge Processor passes span coordinates and slopes to one of five parallel span processors. Span Processor 0 manages spans 0, 5, 10, etc. Span Processor 1 manages spans 1, 6, 11, etc. Since spans generated from a single polygon are always adjacent, the span processing load is evenly distributed across the five processors.

4.1.3 Raster Subsystem

The Raster Subsystem contains 20 Image Engines, each of which controls 1/20th of the pixels on the screen in an interleaved pattern of 2-dimensional sections. A single Image Engine owns 65,536 of the screen pixels; each pixel is associated with 96 bits of data. Each pixel's 96 bits are organized as follows:

- Image bit-planes: 32 bits of R, G, B and optional Alpha information per buffer (2 buffers); supports either double-buffered 32-bit color (R, G, B and optional Alpha) or double-buffered 12-bit color index modes of operation.
- Depth planes: 24 bit Z-buffer
- Overlay/Underlay planes: 4 bits
- Window planes: 4 bits

An additional feature of the Raster System is the fast pixel read and write rate. The pixel write rate is 16 million pixels per second in image mode. This allows the hardware to implement a real-time pan and zoom capability by reading pixels from the frame buffer, replicating them, and writing them back to the frame buffer.

4.1.4 Display Subsystem

Five Multimode Graphics Processors (MGPs) concurrently read the contents of the Image and Window Planes. These MGPs allow simultaneous display of up to 64 different windows in various color modes. This allows users to display both static and dynamic images without having to consider the underlying color representation scheme.

The GTX graphics subsystem displays multiple images simultaneously, in an overlapping window environment, in single buffered RGB, double buffered RGB, single buffered color-index and double buffered color-index color modes.

(For more information on the graphics architecture, see "IRIS GT Graphics Architecture: A Technical Report.")

4.2 GRAPHICS SYSTEM SOFTWARE ARCHITECTURE

The IRIX™ graphics system software manages the graphics subsystem, as well as providing a high-speed path for delivery of graphics data to the subsystem.

4.2.1 Graphics DMA

To achieve the graphics performance possible with the GTX, it is necessary that an order of magnitude more data be delivered to the graphics system in the same amount of time. One key element in delivering this data is optimization of the graphics hardware to accept vertices as the basic drawing primitive. Taken together, a set of vertices describes a polygon that the hardware draws and fills as appropriate. Each vertex may have several pieces of information, such as location, color and normals for lighting.

Each piece of information can be delivered to the graphics subsystem through a DMA transaction, which allows overlap between moving the data and setting up the next vertex. Each transaction typically moves four 32-bit words of data. These four words are enough to describe the location ($\langle x, y, z, w \rangle$), or the color ($\langle r, g, b, a \rangle$).

This DMA transaction is initiated by the processor through a single load instruction to prime the DMA engine, followed by a store instruction to the address of the vertex in user virtual memory. This triggers the DMA transaction immediately. This allows the user total freedom in placement of graphics data in his address space. All this occurs within the Graphics Library, transparent to the user application, requiring no special coding techniques.

4.2.2 Pipe Access and Feedback Areas

The graphics pipe is mapped into the user address space, along with a separate portion of memory called the feedback area. The pipe can thus be accessed either directly (for short commands) or indirectly through graphics DMA (for high speed drawing). In addition, each process using graphics has a private communication area with the graphics mapped into its address space. This feedback area is used to return results of queries or other interaction with the graphics subsystem.

4.2.3 Graphics Context Switching

The graphics subsystem supports up to 16 contexts in parallel within the graphics subsystem. Additional contexts may be managed in a "virtual context" manner by swapping an entire graphics hardware context as needed.

As the kernel context switches among processes using the graphics hardware it informs the subsystem of each host context switch, which results in a parallel context switch within the subsystem.

Graphics processes without a current context in the subsystem have the pipe and feedback areas unmapped from their virtual address space. Thus they may continue to run and process data as long as no graphics access is made. If graphics access is attempted, a page fault occurs, the process is blocked, and the window manager informed of the event. The window manager can then schedule the process to a hardware context as desired.

4.2.4 Kernel Graphics Management

Many implementations of graphics under UNIX™ treat the graphics subsystem as just another device, and force all management to be done at user level. This approach has severe drawbacks. For instance, heavy context switching between the user level processes and the graphics manager is often seen. This approach makes consistent and integrated graphics support difficult or impossible.

The IRIX™ kernel integrates graphics management into the basic functionality of the kernel. This is not the simple "text output" mode of many primitive graphics subsystems—the GTX graphics subsystem knows how to draw a simple character set on its own. Instead, a sophisticated protocol for controlling graphics operation is built into the kernel. Swapping buffers on vertical retrace, exceptions in the pipe, and other conditions result in interrupts to the host, which are then handled with high level protocols.

As mentioned above, graphics context management is integrated into the kernel as well. This provides the high performance and smooth operation of multiple independent graphics windows that is the hallmark of the GTX graphics subsystem.

5.1 SIMULTANEOUS ANALYSIS AND GRAPHICS

By combining tightly coupled processors sharing a common memory with fast access to the graphics pipeline, it is possible to support multiple processes using the CPU and graphics subsystems simultaneously. Ownership of the graphics hardware can be allocated between competing processes on a per context switch basis. While one process is accessing the graphics pipe, others can be taking full advantage of the other CPUs and FPUs.

Take, for example, a two process application, one drawing while the other renders the next frame. The analysis process has the full power of up to seven CPU/FPU pairs, while another is drawing the previous frame at full speed to the graphics pipe.

5.2 LEVELS OF PARALLELISM

In order to provide linear throughput increases with multiple processors, it is important to exploit the hardware parallelism in the system. In the IRIX™ system, this is accomplished by providing three levels of software parallelism: a multi-threaded kernel, explicit parallelism constructs for user applications, and a data-directed parallelizing FORTRAN compiler.

5.2.1 Multi-Threaded Kernel

At the heart of all POWER Series systems is the multi-threaded kernel, which permits multiple processes to be running simultaneously in both User and System space.

As processes become eligible to run, the first available CPU picks up that process and begins to execute it. Which CPU that process was running on previously is immaterial except in so far as the kernel utilizes a cache affinity scheduling algorithm to match processes to their previous CPUs to maximize cache and TLB hit rates.

All CPUs can access the graphics pipe or I/O subsystems simultaneously providing a symmetric rather than a master/slave arrangement. This makes it unnecessary to switch a process from one CPU to another when it accesses an I/O device.

The multi-threaded kernel provides high bandwidth system access. Processes enter the kernel on the CPU they are currently running on, perform the system service, and return. The individual data structures (such as process blocks, inode table, buffer cache, etc.) are semaphored so that multiple CPUs can be executing the particular code simultaneously. Contention for operating system resources is regulated via the use of spinlocks and semaphores.

5.2.2 User Directives

The multi-threaded IRIX kernel provides a seamless way to increase throughput of a multi-user and/or multiple process environment. But it is also desirable to increase the throughput of a single process application.

This is possible using the multi-processing library “libmp” which includes support for application level spin locks, semaphores, shared memory, and dynamic allocation. It also includes a full “libc” with the most often used library routines semaphored.

Synchronization is possible between both share group processes and unrelated processes. This provides a quick way for existing multi-process applications (for instance, an analysis process talking to a display process) to exploit the multiple processors—using the primitives in the “libmp”; a shared region can be created between them. If any synchronization is required (over and above a simple producer-consumer queue) spinlocks and semaphores are available. For simple synchronization, shared memory may also be used.

With the new *arenamalloc()* library call, dynamic space may be allocated inside this shared space. Thus existing communication mechanisms such as pipes, sockets, or files can be replaced with a high bandwidth memory messaging scheme.

If the application is a single large process, then the explicit multi-threading afforded via share groups can be used. If a process executes an *sproc(2)* system call then a share group is formed. Any number of processes/share group members may be created, and each will run on a separate processor (up to the number of processors).

A share group, by virtue of a common address space and file descriptors, can execute parts of an application in parallel. For instance, one part can perform data generation while the other is displaying the data. Synchronization is again performed using either memory, or the libmp spin locks and semaphores. Using libmp, share group members may also request system services, since the library routines such as malloc, perror, and printf have all been semaphored.

5.2.3 Parallelizing Compiler

The IRIX™ kernel supports a sophisticated program development environment, based on the MIPS compiler suite. Multiple language front ends convert each language to a common intermediate form which is then handled by common tools for optimization and code generation.

The MIPS optimizer provides several levels of optimization. Like other systems, local optimization and pipeline scheduling is performed easily. In addition global optimization and procedure inlining are also supported, allowing the programmer to achieve extremely high performance levels.

Layered on top of the FORTRAN front end is the POWER FORTRAN accelerator. This data flow analyzer is one of the most sophisticated analysis tools available for producing parallelized FORTRAN. Its output is parallelization directives interspersed in the original output file. These are recognized by the FORTRAN front end, which generates appropriate common intermediate code containing parallelization directions. The common code generator then produces parallelized code which takes direct advantage of operating system thread support and hardware locks.

In cases where the automatic analyzer is unable to safely parallelize a piece of code, the user can explicitly add parallelization directives to his code if he knows that the section of code is indeed safe to parallelize. The directives used are the same as the analyzer uses and thus are treated identically in the common processing steps.

A parallel processing library has support for shared address processes, semaphores, spinlocks and barriers. A fully semaphore standard C library completes the software support for the procedure level model of parallelism.

As a result of this strategy, the optimizer can be used directly to further enhance the performance of parallelized code. As technology becomes available to parallelize other languages, they can be added to the compiler suite easily, leveraging off the existing parallel programming support.

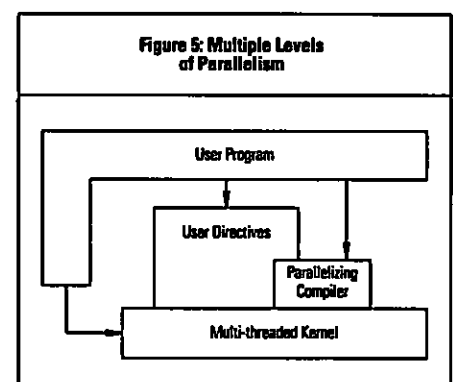
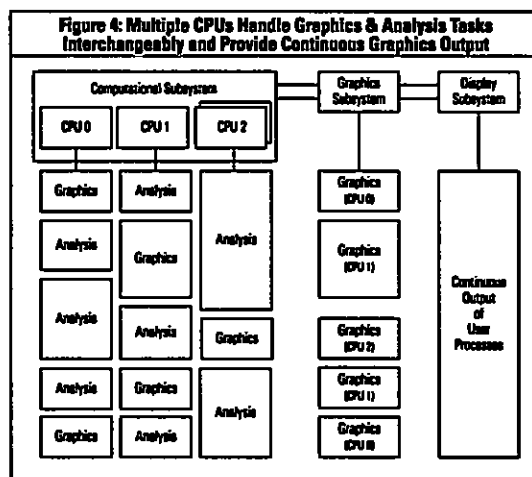
5.3 I/O THROUGHPUT

The hardware interface for the primary I/O—SCSI, Ethernet, and VME—provides a parallel interface to the kernel. Multiple drivers can be talking to the different I/O subsystems at the same time. Interrupts from different devices (including each device on the VME) are routed to different processors so that interrupt servicing is also distributed.

Multiple processor I/O access is combined with state-of-the-art disk controllers to provide control of up to four disks per controller, each with full track caches, overlapped seek optimization, and disk sorting. This combined with the IRIX EFS file system provides up to 2000 Kbytes/sec. per controller through the file system (using SMD disk drives). Even using only a single SCSI disk drive, file system throughput surpasses 1000 Kbytes/Sec.

Networking is supported via an integral Ethernet adapter, access to which is completely independent of access to the VME adapter. The SCSI adapter, useful for cartridge tape and lower speed disks is also independent of other I/O activity.

Devices on the VME bus that don't support page scatter-gather DMA can still be utilized as high speed controllers by using the integral hardware page maps. To further speed reads from an I/O device the hardware detects data in the caches that is going to be made 'stale' by the read request and invalidates it. This eliminates the need for the operating system to do a costly cache flush.



The IRIX™ operating system is based on AT&T System V UNIX™ Release 3.1. This version of UNIX™ has a substantial set of features to easily support both the programmer and the user. In addition to these features, the following facilities have been added to further enhance the value of the system.

6.1 PARALLEL PROGRAMMING SUPPORT

To take full advantage of a multiprocessor, a parallel programming model which is easy to use and powerful is necessary. This is accomplished through several levels of new functionality.

Process share groups provide a lightweight processing facility with all of the advantages of thread-like schemes and little of the burden. A parallelized version of the C library is available for using in a threaded process to insure that commonly used facilities like Standard I/O and *malloc()* are available and work as expected. Arena based memory allocation is supported.

Included in the parallel C library are locking and semaphore primitives which operate without kernel interaction. This avoids performance penalties for invoking kernel facilities. The primitives include debugging support to ease tracking down problems in complex multi-thread applications. In addition, metering is available to determine the usage and contention of semaphores and locks for fine-tuning an application.

6.2 REAL-TIME PROGRAMMING SUPPORT

Several features of IRIX™ provide support for high-speed interactive and control applications.

Non-degrading priorities provide a method for the programmer to precisely control the priority of his application in relation to other processes on the system. Normal problems associated with the standard UNIX™ scheduling paradigm can be avoided.

High-precision timing is supported through one millisecond accuracy of the time reporting mechanism, as well as being available through a signal-based timer facility based on Berkeley timers.

Fine-grained memory locking allows the programmer to insure that critical code or data is always memory resident.

Mapped files and devices provide the programmer access to files or physical devices as if they were part of the process address space. Complex file manipulation is simplified, and devices can be accessed and managed directly from user level.

6.3 BERKELEY FEATURES

Many features of BSD 4.3 are supported in IRIX™. Job control, 4.3 networking facilities, and many Berkeley commands are supported. For the programmer, BSD 4.3 system calls are supported where they do not conflict with System V semantics.

6.4 LANGUAGE SUPPORT

IRIX™ supports C, FORTRAN 77 with VMS extensions, Pascal, ADA, and PL/1. This support is integrated with the multi-process graphical debugger Edge, which is based on a multi-process version of dbx. This debugger also allows attaching to an arbitrary running process, which is convenient for debugging demons. Many processes may be debugged in a single session, even though they are unrelated.

6.5 FILE SYSTEM ENHANCEMENTS

The EFS extent-based file system provides high levels of throughput for both computational and graphics programs. EFS supported symbolic links and long filenames. While extremely large block sizes are used to enhance throughput, EFS can deal with a fragmented disk easily. The user can use the entire disk with only a moderate loss in performance.

7.1 PRODUCT LINE COMPATIBILITY

Compatibility between all 4D products is provided via shared libraries, which serve both to reduce the size and disk space requirements of a program, as well as hiding system differences from programmers. This provides binary compatibility across the entire 4D product line, including Personal IRIS models, all 4D Superworkstations and all POWER IRIS and POWER Station systems.

7.2 UPGRADABILITY

The POWER Series systems are 100% binary compatible with all IRIS-4D systems. No code changes are required when moving from one machine to another.

Corporate Office
2011 N. Shoreline Boulevard
Mountain View, CA 94043
(415) 960-1980

Federal Sales Division
Bethesda, MD
(301) 564-1980

Regional Offices
Mountain View, CA
(415) 960-1940

Farmington, MI
(313) 478-5446

Dallas, TX
(214) 788-4122
Waltham, MA
(617) 891-8100

*For additional U.S. sales office
locations call (800) 338-6272*

Canadian Office
Toronto, Ontario
(416) 674-5300

International Headquarters
Geneva, Switzerland
(41-22) 7987525

Sydney, Australia
(61-2) 959-3349
Beijing, China
(86-1) 8312211/ 456

Paris, France
(33-1) 34659685

Hong Kong
(852-5) 257237
Tel Aviv, Israel
(972-3) 492191/ 93
Milano, Italy
(39-2) 95300268

Tokyo, Japan
(81-3) 473-8444
Gouda, Netherlands
(31) 1820-34144

Singapore
(65) 777-3088

Stockholm, Sweden
(46-8) 330-705
Abingdon, United Kingdom
(44-235) 554-444
Cologne, West Germany
(49-221) 443011



SiliconGraphics
The 3D Computer Company