SiliconGraphics
Computer Systems

Onyx2™

Technical Report

Technical Report, August 1996

SiliconGraphics
Computer Systems

Onyx2 Reality,™ Onyx2 InfiniteReality and Onyx2™ InfiniteReality2™ Technical Report

Silicon Graphics, Inc.

A description of the functionality and operation of the Onyx2 Reality, Onyx2 InfiniteReality and Onyx2 InfiniteReality2 visualization systems.

August 1998

TABLE OF CONTENTS

1.0 Introduction 1

1.1 Onyx2 InfiniteReality 1
1.2 System Configurations 1
1.2.1 Onyx2 with Reality Graphics Deskside System 1
1.2.2 Onyx2 with InfiniteReality2 Graphics 2
1.2.3 Onyx2 and Multi-Pipe Rendering Mode 5
1.2.4 Onyx2 in Multi-Seat Mode 6
1.2.5 Onyx2 in Multi-Display Mode 6
1.3 About This Document 6

2.0 Onyx2 Reality and Onyx2 InfiniteReality2 Overview 7

3.0 Graphics APIs 9

3.1 OpenGL® 9
3.2 IRIS Performer 9
3.3 OpenGL Optimizer 10
3.4 “Fahrenheit” 11

4.0 Geometry Subsystem 13

4.1 Geometry Engine 13

5.0 Raster Subsystem 14

5.1 Texture Processing 14
5.2 Pixel Processing 14

6.0 Display Subsystem 16

6.1 ACCESS.bus 17

7.0 Graphics Features and Capabilities 18

7.1 Resolution 18
7.2 Graphics Primitives 20
7.3 Immediate Mode and Display List Mode 20
7.3.1 Rendering Performance and Vertex Arrays 21
7.3.2 Display List Storage Cache and Host Memory 21
7.3.3 Primitive Processing 22
7.3.4 Sprites 22
7.4 Color 22
7.4.1 Shading 23
7.4.2 Color Blending 23
7.4.3 Lighting 23
7.4.4 Advanced Lighting Model 23
7.4.5 Surface Properties 24
7.4.6 Infinite Light Sources 24
7.4.7 Local Light Sources 24
7.5 Transparency 25
7.6 Hidden Surface Removal 25
7.7 Coplanar Geometry Prioritization 25
7.8 Anti-Aliasing 26
7.8.1 Sample Memory 27

7.8.2 Onyx2 Reality Restrictions 27
7.9 Texture/Image Mapping 28
7.9.1 Texture Download 28
7.9.2 Texture Formats 29
7.9.3 Texture Filtering 30
7.9.4 Global Texturing a.k.a. CLIP mapping 31
7.9.5 Add, Replace, Blend, Decal, and Modulation 34
7.9.6 Texture/Image Memory 34
7.9.7 Texture Memory Packing 36
7.9.8 Texture Transparency and Contouring 36
7.9.9 Perspective Correction 36
7.9.10 Detail Texture 36
7.9.11 Sharp Texture 37
7.9.12 Projective Texture 37
7.9.13 Shadow Maps 37
7.9.14 Billboards 37
7.9.15 Detail Management 37
7.9.16 Video to Texture 37
7.9.17 3D Textures and Volume Rendering 38
7.9.18 Texture Color Lookup Tables 38
7.10 Imaging Operations 38
7.10.1 Convolution 38
7.10.2 Histogram and Minmax 39
7.10.3 Color Matrix and Linear Color Space Conversion 39
7.10.4 Window/Level Support 39
7.10.5 Lookup Tables (LUTs) 39
7.11 Atmospheric Effects 39
7.12 Offscreen Rendering 40
7.13 Multi-Channel Display Generator 42
7.13.1 Standard Formats and Format Combinations 42
7.13.2 2-Channel and 8-Channel Configurations 45
7.13.3 Onyx2 Reality and Onyx2 InfiniteReality2 Multi-Channel Features: Considerations 45
7.13.3.1 Swap Rates Must Be Equal 45
7.13.3.2 Frame Buffer to Video Display Subsystem Transmission Bandwidth 47
7.13.3.3 DAC Output Bandwidth 47
7.13.3.4 Frame Buffer Memory 48
7.13.3.5 Frame Buffer Read/Write Bandwidth 48
7.13.4 Video Format Combiner 48
7.13.5 Examples of Video Format Combinations 50
7.13.6 Video Format Compiler 51
7.13.7 Dynamic Resolution 51
7.13.8 Stereoscopic Video Output 52
7.13.9 Direct Support of Color Field Sequential Video 52
7.13.10 Luminance 52
7.13.11 Programmable Video Timing 52
7.14 Genlock 53
7.14.1 Swap Synchronization 53
7.15 Digital Video Multiplexer - DPLEX 54

TABLE OF CONTENTS

7.15.1 Breakthrough Scalability for Visualization 54
7.15.2 Distortion Correction 54
7.15.3 Large Model Interactivity 54
7.15.4 Application and OS Support 55
7.15.5 DPLEX and MonsterMode Rendering 55
7.16 Digital Video Option (DIVO) 56
7.17 Graphics-to-Video Option (GVO) 56
7.18 Integration with X Window System 57

8.0 Onyx2 Hardware Architecture 58

8.1 Architecture Overview 58
8.2 Node Card 59
8.2.1 Processor 59
8.2.2 Memory 60
8.2.3 Hub 60
8.2.4 Global Real-Time Clock 61
8.3 I/O Subsystem 61
8.3.1 XIO Protocol 61
8.3.2 XBOW 62
8.3.3 XIO Devices 64
8.3.4 Media Input/Output (MIO) 64
8.3.5 PCI-64 64
8.4 Interconnect Subsystem 65
8.4.1 Topology 65
8.4.2 Routers 66
8.5 Distributed Shared Address Space 66
8.6 Onyx2 Memory Hierarchy 67
8.7 Cache Coherency 68
8.8 System Latencies 69
8.9 System Bandwidths 69
8.10 R10000 Processor Architecture 71
8.10.1 R10000 Product Overview 72
8.10.2 Primary Data Cache 73
8.10.3 Secondary Data Cache 73
8.10.4 Instruction Cache 74
8.10.5 Branch Prediction 74
8.10.6 Queueing Structures 74
8.10.7 Register Renaming 74
8.10.8 Execution Units 75
8.10.9 Load/Store Units and the TLB 75

9.0 IRIX 76

9.1 Introducing IRIX for Onyx2 76
9.1.1 IRIX Means Scalability 76
9.1.2 IRIX 6.5 for Onyx2 76
9.2 Memory Management 77
9.3 Scheduling 78
9.3.1 Gang Scheduling 79
9.3.2 Processor Affinity 79
9.3.3 Frame Scheduling 79

9.3.4 Deadline Scheduling: Not Supported 80
9.3.5 Processor Sets: Not Supported 80
9.4 Symmetric Multiprocessing and Multitasking 80
9.4.1 Sprocs and Sproc Synchronization 80
9.4.2 POSIX Threads and Synchronization Primitives 81
9.4.3 Interrupt Threads and Synchronization Primitives 81
9.5 Compatibility 82
9.5.1 COFF Obsoleted 82
9.5.2 Binary Compatibility between IRIX 5.3/IRIX 6.2/IRIX 6.4 andIRIX6.5forOnyx2 82

LIST OF FIGURES

Figure 1. Onyx2 Reality System 2

Figure 2. Onyx2 InfiniteReality2 Deskside System 3

Figure 3. Onyx2 InfiniteReality2 Single-Rack System 4

Figure 4. Onyx2 InfiniteReality2 Multi-Rack System 5

Figure 5. Onyx2 Visualization Pipeline 8

Figure 6. Onyx2 InfiniteReality2 Display Resolution Examples 18

Figure 7. Choices of Allocation of 256 Bits of Frame Buffer Memory 19

Figure 8. Onyx2 Reality Display Resolution Examples 19

Figure 9. Meshed Polygon Strips 20

Figure 10. Infinite and Local Light Sources 24

Figure 11. MIPmapping 31

Figure 12. CLIP Mapping 33

Figure 13. CLIP Mapping 34

Figure 14. Frame Buffer Pixel Memory 40

Figure 15. Frame Buffer Pixel Memory Including Offscreen Buffers 41

Figure 16. Disjoint Buffers and PBuffers 41

Figure 17. 2- and 8-Channel Options 45

Figure 18. Equal and Unequal Swap Rates 46

Figure 19. Dynamic Resolution 48

Figure 20. Using the Video Format Combiner to Place Six Video Channels . 50

Figure 21. Example of 3-Pipe DPLEX Time Compositing 56

Figure 22. Onyx2 Scalability 58

Figure 23. Node Board Block Diagram 59

Figure 24. Block Diagram of a Hub ASIC 61

Figure 25. Functional Location of XBOW ASIC 63

Figure 26. Datapaths in an Interconnection Fabric 65

Figure 27. Location of a Router Board in an Onyx2 System 66

Figure 28. Memory Hierarchy, Based on Relative Latencies and Data Capacities 68

Figure 29. R10000 Processor Block Diagram 73

LIST OF TABLES

Table 1. Lighting Model Components 23

Table 2. Anti-Aliasing Sample Memory Options for Onyx2 InfiniteReality2 . 27

Table 3. Anti-Aliasing Sample Memory Options for Onyx2 Reality 28

Table 4. Texture Rate for 1024x1024 Texture (RGB10 visual for framebuffer read) 29

Table 5. Largest Available16-Bit Texture Maps 35

Table 6. 2D Texture Memory Sizes and Quantities 35

Table 7. 3D Texture Sizes for 16-Bit Texture 36

Table 8. Standard Formats Shipped with all Onyx2 Reality and Onyx2 InfiniteReality2 Systems 43

Table 9. Standard Format Combinations Shipped with All Onyx2 Reality and Onyx2 InfiniteReality2 Systems 44

Table 10. Examples of Video Format Combinations 51

Table 11. Bandwidths on Processor, Memory, and I/O Paths 70

Table 12. Bisection Bandwidths of Various Onyx2 Configurations 70

Table 13. Maximum Simultaneous System Bandwidth 71

1.0 Introduction

Silicon Graphics, the recognized leader in developing sophisticated visual computing systems, is dedicated to advancing computer visualization technology now and in the future. After more than a decade, hardware and software solutions from Silicon Graphics continue to define the state of the art in visual computing for creative professionals, scientists, and engineers. Onyx2 Reality™ and Onyx2 InfiniteReality2™ are the latest offerings in the most advanced line of binary-compatible visualization systems available. These systems establish a new high-water mark for their ability to process 2D, 3D and video data in real time, making them true visualization pipelines rather than simply graphics sub-systems.

1.1 Onyx2 InfiniteReality

For the sake of simplicity, this document refers only to Onyx2 InfiniteReality2, not Onyx2 InfiniteReality®, except that the InfiniteReality Geometry Engine board is the GE14 while the InfiniteReality2 Geometry Engine board is the GE16, and the InfiniteReality Raster Manager board is the RM7 while the InfiniteReality2 Raster Manager board is the RM9. However, everything contained herein also applies to InfiniteReality. InfiniteReality2 represents a performance enhancement over InfiniteReality, providing improved geometry, image processing and anti-aliased pixel fill performance. InfiniteReality2 is not a new architecture.

1.2 System Configurations

Onyx2 Reality and Onyx2 InfiniteReality2 systems are packaged in both compact deskside units for optimal price/performance, and larger rack units for maximum performance and expandability. All Onyx2 Reality and Onyx2 InfiniteReality2 systems take full advantage of the unrivaled I/O bandwidth afforded by the Onyx2™ host platform. Onyx2 provides a scalable system architecture, with each increment of processing power adding up to 4GB additional memory, 650MB/second of memory bandwidth, 711MB/second of interprocessor communications bandwidth, and 711MB/second of I/O bandwidth. Onyx2 Reality and Onyx2 InfiniteReality2 systems are available in the configurations highlighted below. Terms such as XBOW, XIO, XTOWN, and KTOWN are explained in Section 8.0, Onyx2 Hardware Architecture.

1.2.1 Onyx2 with Reality Graphics Deskside System

The Onyx2 Reality Deskside system includes two or four 4MB cache MIPS® R10000® processors, 128MB to 8GB of memory, and a single graphics pipeline with one or two Raster Managers providing up to 64MB of texture memory and 80MB of frame buffer capacity (Figure 1).

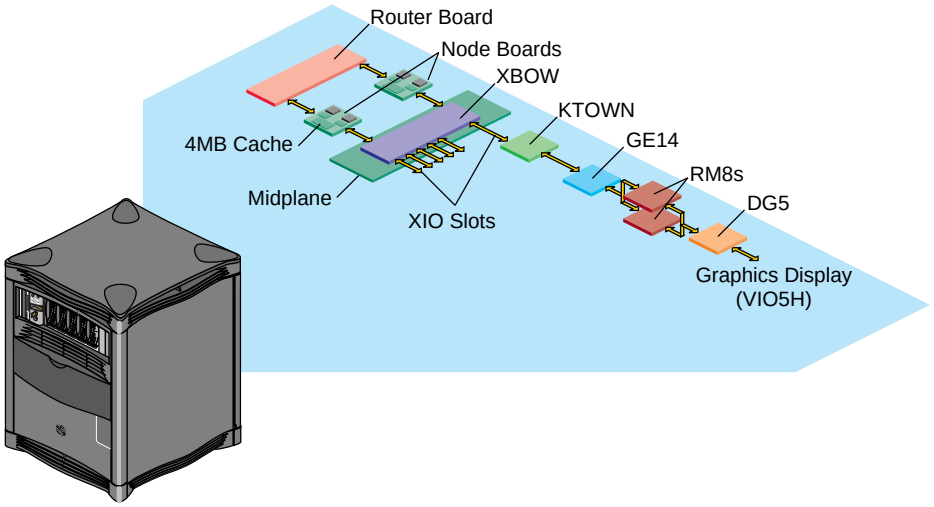


Figure 1. Onyx2 Reality System

Onyx2 Reality is the most affordable visual supercomputer available in the marketplace. It is expandable to 4 processors and 8GB of main memory and includes a graphics sub-system with many of the same performance characteristics and features as InfiniteReality2, all at a power desktop price point. Onyx2 Reality is ideal for individual power users who require greater levels of performance and/or realism than desktop systems can provide.

1.2.2 Onyx2 with InfiniteReality2 Graphics

Onyx2 with InfiniteReality2 Graphics Deskside System

The Onyx2 deskside system includes two or four 4MB cache MIPS R10000 processors, 128MB to 8GB of memory, a single graphics pipeline with one or two Raster Managers providing 64MB of texture memory and up to 160MB of frame buffer capacity, and a super-wide, ultra-high-resolution 1920x1200 monitor (Figure 2).

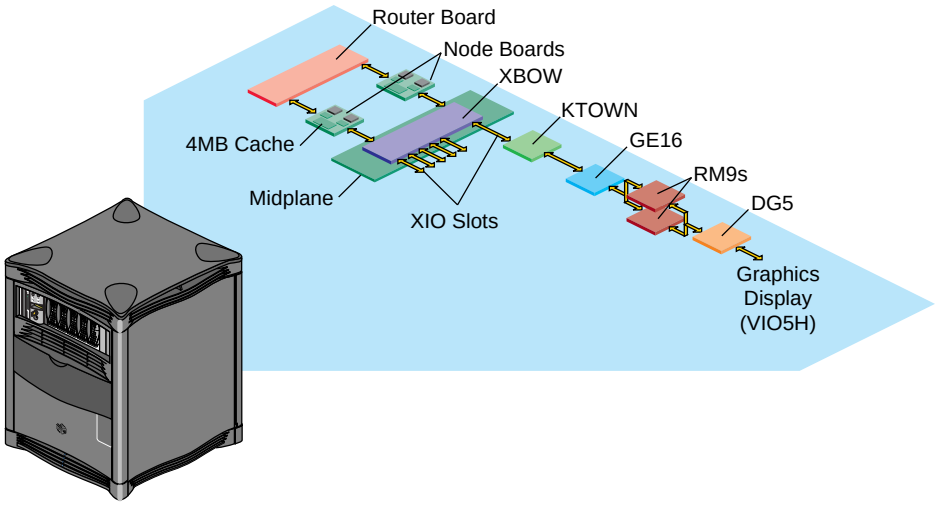


Figure 2. Onyx2 InfiniteReality2 Deskside System

Onyx2 InfiniteReality2 Deskside systems deliver all the power of the InfiniteReality2 visualization pipeline in a compact deskside enclosure. Deployed as a workgroup resource, Onyx2 InfiniteReality2 Deskside doubles as a visualization platform and high performance compute system.

Onyx2 with InfiniteReality2 Graphics Single-Rack System

The Onyx2 single-rack system includes two to eight 4MB cache MIPS R10000 processors, 128MB to 16GB of memory, one graphics pipeline with one, two, or four 80MB Raster Managers providing 64MB of texture memory and up to 320MB of frame buffer capacity, one graphics pipeline with one or two Raster Managers providing 64MB of texture memory and up to 160MB of frame buffer capacity, and a super-wide, ultra-high-resolution 1920x1200 monitor (Figure 3).

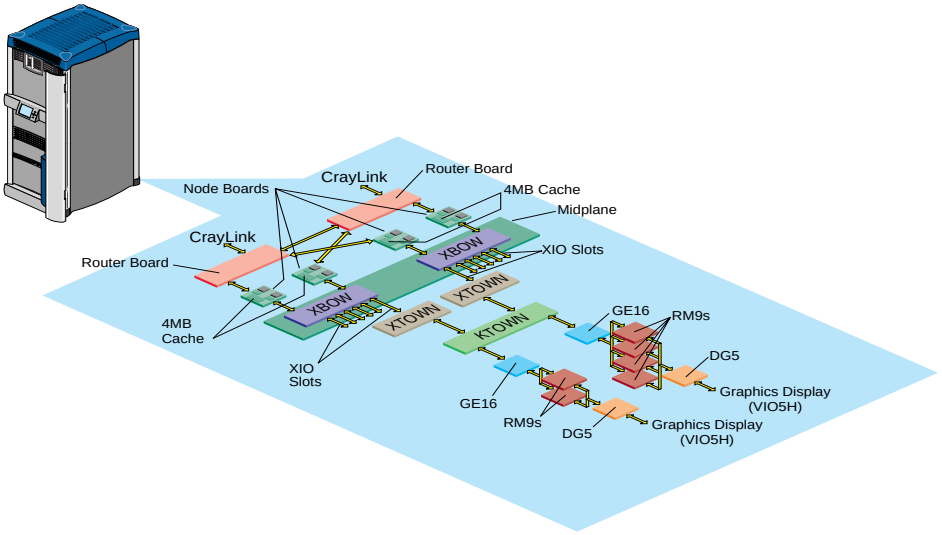


Figure 3. Onyx2 InfiniteReality2 Single-Rack System

Onyx2 InfiniteReality2 Rack systems are typically deployed as departmental resources to serve a wide range of visual and numeric computing needs. Because of their modular, highly scalable architecture, Onyx2 InfiniteReality2 Rack systems can grow to meet an organization's diverse and ever-expanding needs.

Onyx2 with InfiniteReality2 Graphics Multi-Rack System

The Onyx2 Multi-Rack system may be expanded to include up to 128 MIPS R10000 processors, 16 InfiniteReality2 visualization pipelines, and 256GB of main memory, all seamlessly interconnected to form a single machine image. (Figure 4). The first rack in the system contains a processor module (two to eight processors) in the bottom section and a graphics module (one 4RM pipe and one 2RM pipe) in the top section. The second rack may contain either two processor modules (two to 16 CPUs), two graphics modules (up to two 4RM pipes and up to two 2RM pipes), or one processor module (two to eight CPUs) and one graphics module (one 4RM pipe and one 2RM pipe).

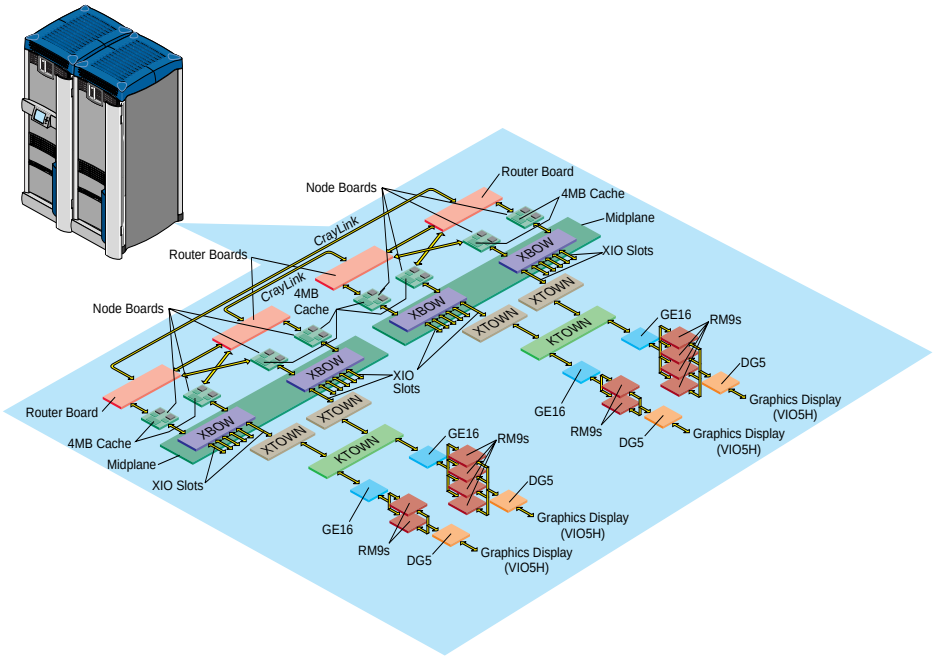


Figure 4. Onyx2 InfiniteReality2 Multi-Rack System

Onyx2 was designed specifically to deliver flexibility and extensibility, so a single Onyx2 multi-pipe system may be deployed in different operating “modes” to solve different problems. As described below, these modes include supercomputer, multi-seat, multi-display, and multi-pipe rendering. In most cases, switching from one mode to another is accomplished exclusively in software; no hardware reconfiguration is required. With their native flexibility, Onyx2 multi-pipe systems are typically deployed as enterprise-wide interactive visualization and compute seats, multimedia presentation resources, or platforms for tackling an organization’s Grand Challenge computing problems.

1.2.3 Onyx2 and Multi-Pipe Rendering Mode

Onyx2 InfiniteReality2 systems can be tasked to focus all pipelines on a single rendering window, resulting in near-to-perfect linear scalability of visualization performance (where visualization performance includes geometry rate, pixel fill rate, texture capacity and/or texture paging bandwidth). This application of multiprocessing principles to visualization can be accomplished on Onyx2 in two ways: with the DPLEX (Digital Multiplexing) hardware option or with MonsterMode software. DPLEX is an optional daughtercard for Onyx2 systems that enables digital multiplexing of two or more IR pipelines. One DPLEX option is required for each pipeline in the system. The concept is simple: multiple pipelines operate simultaneously on successive frames of an application which are then digitally multiplexed together before being converted to analog video. Key applications for DPLEX include distortion correction (required for dome simulators) and interactive

large model visualization. For suitable applications, an n-pipe system equipped with DPLEX can support n-times the frame rate of a single-pipe system running the same application. DPLEX is scheduled to ship in late 1998 and will require IRIX 6.5.2 or later.

MonsterMode is the collective name of several software-based methods for distributing a data set over multiple pipelines. Each uses Onyx2's high-bandwidth memory subsystem rather than special hardware (DPLEX) for the inter-pipeline transfers required to partition the data and compose the final image. MonsterMode rendering methods include 2D-composition for handling polygonal models and 3D-composition for handling volumetric models. 3D-composition provides the further benefit of additive texture-mapping resources: an n-pipe system using MonsterMode 3D-composition software has n-times the effective texture memory and texture download bandwidth of a single-pipe system (up to 1GB and 5GB/sec, respectively, for a 16-pipe configuration). Applications will require modification to support DPLEX or MonsterMode operation. Full-screen applications written to IRIS Performer™ or OpenGL Optimizer™ 1.1 will require the least work. For details, please consult your Silicon Graphics representative.

1.2.4 Onyx2 in Multi-Seat Mode

With the addition of extra monitors, keyboards and mice (and Base Graphics I/O cards if required), multipipe Onyx2 systems can be deployed in multi-seat mode. In this mode, multiple simultaneous users each have exclusive control over an InfiniteReality2 graphics pipeline as well as shared access to all other system resources, including memory, disk, and compute. Multi-seat mode provides a solution for users who require more memory capacity, disk storage, graphics performance, or processing capabilities than are afforded by desktop workstations. Aside from the need for additional monitors and keyboards, multi-seat mode requires no additional hardware reconfiguration.

1.2.5 Onyx2 in Multi-Display Mode

Each InfiniteReality2 or Reality pipeline ships standard with a 2-channel display generator and may be ordered with an optional 8-channel display generator, so every Onyx2 configuration is capable of driving multiple displays. Additionally, a suite of tools permits users to choose from a wide range of video formats and format combinations to suit any application or display technology. Onyx2 is therefore ideal for driving Virtual Reality devices such as RealityCenter™ facilities, CAVEs, visual simulators, Power Walls, head-mounted displays, and other advanced interfaces, in both monoscopic and stereoscopic formats.

1.3 About This Document

This document serves as a detailed report on Onyx2 Reality and Onyx2 InfiniteReality2 subsystems. Onyx2 implements the Cache Coherent Non Uniform Memory Access (ccNUMA) architecture, as do the Origin2000™ servers. Some sections of the Origin2000 Technical Report have been included for user convenience. For in-depth technical information on the CPU, memory, and I/O subsystems of the Onyx2 family, see the Origin2000 Technical Report. For more information about Onyx2 multipipe rendering, see the DPLEX data sheet, available through local Silicon Graphics sales offices worldwide, or the Advanced Graphics Marketing website. For more information on Virtual Reality and RealityCenters, see the Silicon Graphics® worldwide website (www.sgi.com).

2.0 Onyx2 Reality and Onyx2 InfiniteReality2 Overview

Onyx2 Reality and Onyx2 InfiniteReality2 are designed to concurrently process 3D geometry, imagery, and video data at real-time rates, making them true visualization pipelines. The architectures underlying Onyx2 Reality and Onyx2 InfiniteReality2 combine raw processing power, an advanced feature set, and record-setting throughput for unrivaled visualization performance. Differentiated capabilities include:

1. Hardware accelerated image processing
2. High multisample, full-scene anti-aliased pixel fill capacity
3. Hardware tuning for complex geometry
4. Realtime texture paging from main memory or disk
5. Scalable visualization capabilities using multipipe rendering methods for higher geometry rate, pixel fill rate, texture capacity, and texture paging bandwidth
6. Flexible display sub-system
7. Tightly-coupled host-integrated computer image generator (HI-CIG) architecture

The Onyx2 Reality and Onyx2 InfiniteReality2 visualization subsystem comprises three unique boards: Geometry Engine®, the Raster Manager (RM), and the Display Generator (DG). Geometry Engine provides the interface to the compute subsystem and is responsible for the preliminary geometric transformations (translation, rotation, scale) and lighting calculations for three-dimensional data. It also performs image processing functions (convolution, histogram equalization, orthorectification, and more) on two-dimensional images. The Raster Manager takes the results from Geometry Engine and scan-converts the data into a digital image. The RM board performs various pixel operations, including Z-buffer testing, stencil testing, color and transparency blending, texture mapping, and multisample anti-aliasing. The RM has both texture memory for storing textures (images) and frame buffer memory for performing and storing pixel operations. Finally, the DG board takes the output from the RM board and converts the digital data into an analog signal for driving a display. A DG can be programmed to display different portions of the RM frame buffer to its two display channels. One of these display channels can be converted into a composite video mode and connected directly to a video tape recorder. Optional configurations of the DG board support:

1. Up to eight analog output display channels (DG5-8),
2. Graphics to CCIR601 Serial Digital Video out (GVO), or
3. Low Voltage Differential Signal (LVDS) digital video out (DPLEX).

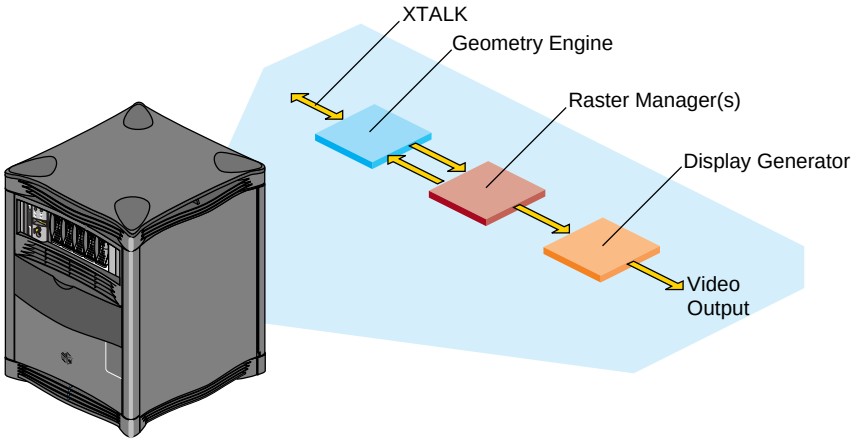


Figure 5. Onyx2 Visualization Pipeline

3.0 Graphics APIs

3.1 OpenGL®

The native graphics programming interface for Onyx2 Reality and Onyx2 InfiniteReality2 is OpenGL. OpenGL is the premier environment for developing high-performance 2D and 3D graphics, imaging, and video applications. The OpenGL application programming interface (API) is a vendor-neutral, multiplatform industry standard that provides access to a rich set of graphics capabilities, including transformations, lighting, atmospheric effects, texturing, anti-aliasing, hidden surface removal, blending, and pixel processing, among many others. OpenGL is designed to be independent of operating systems and window systems, and is supported on virtually all workstations and personal computers available in the industry today.

Among OpenGL's many desirable characteristics is its extensibility: On Onyx2 Reality and Onyx2 InfiniteReality2, OpenGL capabilities have been extended to include dozens of special functions for applications in visual simulation, imaging, volume rendering, advanced industrial design, film and video effects, and animation.

In addition, OpenGL provides the foundation for Silicon Graphics' higher level programming layers. IRIS Performer, OpenGL Optimizer, and the upcoming Fahrenheit Scene Graph, for example, are all supported by OpenGL.

3.2 IRIS Performer

IRIS Performer is a high-performance, portable 3D rendering toolkit for developers of real-time, multiprocessed, interactive graphics applications for the Silicon Graphics range of graphics systems. IRIS Performer simplifies development of complex applications such as visual simulation, simulation-based design, virtual reality, interactive entertainment, broadcast video, CAD and architectural walkthrough. IRIS Performer guarantees the highest performance and automatic, optimal use of available components and features of all Silicon Graphics machines. This includes the use of multiple CPUs, multiple graphics pipelines, filesystem and disk access, and realtime scheduling.

Key Features Include:

- Real-time fixed frame rate operation for truly immersive simulations
- Automatic scene and load management features; view frustum culling, Level-of-Detail (LOD) evaluation and Dynamic Video Resolution (DVR)
- Run-time and real-time profiling for use in debugging, tuning, and load management
- Asynchronous paging of database geometry and imagery, including fast loading formats
- Database interoperability with runtime linking with database loaders for loading data files of any or multiple file format(s)
- Automatic paging and management of huge textures, up to 8Mx8M texels in a single texture (Cliptexturing)
- Active Surface Definition (ASD) for automatic paging and evaluation of terrain with continuous Level-of-Detail (LOD) evaluation
- Asynchronous database intersection and user database processing
- Lightpoints and calligraphic points for visual simulation
- Atmospheric effects for visual simulation, including fog, haze, and support for implementing range/angle correct layered fog
- Support for special effects for realistic and dynamic environments
- Video textures

To help developers get the most out of IRIS Performer, sample applications and viewers are included with source code, and more than a Gigabyte of utility libraries, file loaders, examples and data, much contributed by third-party suppliers, are included with IRIS Performer.

3.3 OpenGL Optimizer

OpenGL Optimizer is a programmer's toolkit that provides the key enabling software technologies required for creating and interacting with extremely large and complex 3D datasets. This toolkit includes an Application Programming Interface (API) that offers access to a rich library of advanced graphics algorithms and a suite of tools that make it easier for application developers create CAD/CAM, Digital Content Creation and Digital Prototype applications.

Additionally, OpenGL Optimizer provides high-performance rendering and interaction with large models while maintaining the high model fidelity demanded by CAD and other engineering, scientific and technical applications. Providing attention to detail while maintaining high performance in a cross-platform environment is OpenGL Optimizer's primary strength. OpenGL Optimizer is a comprehensive and profoundly innovative API for large model visualization and interaction that sets new standards for computer aided design.

Features include:

- Scene Graph Culling: Object removal technologies such as view frustum, occlusion and back patch culling.
- Higher Order Representations: Coons patches, Hermite cubic spline patches, NURBS patches, spheres, tori, and many others.
- Topology Representation: topological relationships between surface patches, such as shared edges and junctions (used to provide crack-free tessellation at shared boundaries of surface).
- Integrated Multiprocessing: High performance rendering harnessing multiple CPUs.
- Integration with Scene Graph API: OpenGL Optimizer is built on the industry standard OpenGL API and a scene graph layer API. It is designed to allow the programmer to use both concurrently.
- High Quality Rendering: Unique advanced shading and reflection mapping capabilities.
- Polygonal Simplification: New advanced simplification technology known as the Successive Relaxation Algorithm allows developers control over high quality polygon mesh reduction and simplification.
- Surface Tessellation: Fast and accurate crack-free tessellation delivers both low polygon counts and high quality.

OpenGL Optimizer serves as a foundation to the forth coming “Fahrenheit Large Model API”

3.4 “Fahrenheit”

Silicon Graphics and Microsoft have formed a strategic alliance to define the future of 3D graphics. The two companies are currently collaborating on a Graphics API development project code-named “Fahrenheit” to provide a common set of low-level and high-level graphics APIs for UNIX® and Windows®. The result will be an increase in the capabilities of visualization applications for a wide variety of consumer, business and professional customers.

Two key APIs defined by the Fahrenheit initiative are the Fahrenheit Scene Graph API and the Fahrenheit Large Model API.

The Fahrenheit Scene Graph API will provide a higher level of programming abstraction for developers creating consumer and professional applications on both Silicon Graphics IRIX® and Microsoft® Windows operating systems. Derived from Silicon Graphics’ current scene graph technology, the Fahrenheit Scene Graph API provides high-level data structures, algorithms and resource management that increase overall graphics performance and assist the development of sophisticated graphics-rich applications. In addition to the performance benefit from using the Fahrenheit Scene Graph, there is an important benefit in the level of extensibility that the Fahrenheit Scene Graph allows. Software developers can build on the framework provided by Fahrenheit Scene Graph and extend its functionality to their needs. Similarly, hardware developers can extend or modify the Fahrenheit Scene Graph to allow applications to take maximum advantage of the current and future hardware capabilities.

The Fahrenheit Large Model Visualization API is an extension to the Fahrenheit Scene Graph. This extension adds functionality such as multiresolution simplification, advanced culling and the ability to deal with curved surfaces. Using this extension developers can quickly and easily write CAD/CAM, Digital Content Creation and Digital Protoype applications that allows customers to create and interact with extremely complex 3-D databases.

The Fahrenheit Scene Graph and Large Model APIs are expected to be released mid-1999.

4.0 Geometry Subsystem

The first stage in the Onyx2 Reality and Onyx2 InfiniteReality2 visualization system is the geometry subsystem. It resides on the Geometry Engine board. The primary functions of the geometry subsystem are handling data transfer to and from the host, performing OpenGL command parsing, and executing geometry and pixel processing commands. The geometry subsystem provides dedicated hardware acceleration for display list processing. The Onyx2 Reality and Onyx2 InfiniteReality2 I/O rate to and from the host is two to three times faster than Onyx2 InfiniteReality. The geometry subsystem of Onyx2 Reality and Onyx2 InfiniteReality2 was designed to support the OpenGL API. Large FIFO buffers are provided to ensure that the geometry subsystem can continue executing without stalling, regardless of the state of other stages within the pipeline.

4.1 Geometry Engine

Geometry Engine processors are custom-designed by Silicon Graphics. Two of four Geometry Engine processors are employed in multiple instruction, multiple data (MIMD) fashion within the geometry subsystem depending on the graphics option chosen. Each Geometry Engine processor contains three separate floating-point cores that execute in single instruction, multiple data (SIMD) fashion.

Geometry Engine processors do both geometry (per vertex) and pixel processing. Geometry processing includes vertex transformations, lighting, clipping, and projection to screen space. Pixel processing includes many common image processing operators, such as convolution, histograms, scale and bias, and lookup tables. Pixel operations can be applied to standard pixels, textures, or video.

5.0 Raster Subsystem

The raster subsystem, residing on the Raster Manager boards, has most of the graphics system's custom VLSI processors and memory. Triangle, point, and line data received by the Raster Manager from the geometry subsystem must be scan-converted into pixel data, then processed into the frame buffer before a finished rendering is handed to the display subsystem for generation of displayable video signals.

By using extensive parallelism in most stages, the raster subsystem performs anti-aliasing, texture mapping, and image processing functions without encountering bottlenecks. Besides basic anti-aliasing for graphics primitives, the graphics system also supports full-screen anti-aliasing with a method called multisampling. With multisampling, images are rendered into a frame buffer with a higher effective resolution than that of the displayable frame buffer; the number of subsamples computed for each pixel determines how much higher. For example, if each pixel has eight subsamples, positioned at eight of the 64 possible subpixel locations, then the effective resolution is eight times the displayable resolution.

When the image has been rendered into these subsamples, all samples that belong to each pixel are blended together to determine pixel colors. Image memory is interleaved among parallel processors so that adjacent pixels are always handled by different processors. Thus, each polygon can have multiple processors operating on it in parallel.

5.1 Texture Processing

For each pixel, scan conversion produces texture coordinate data (S, T, R, W) which are sent to a dedicated texture processing unit. This hardware performs perspective correction on texture coordinates, and determines the detail level for MIPmapping, producing addresses for the relevant area of texture memory. Texture data (texels) are stored in one of several formats, depending on the number and precision of color and transparency components desired. Texels in the area covered by a pixel are read and used in one of several user-definable texture filtering calculations. The resulting color value is sent to pixel processors along with the pre-texture color and depth values produced by the main scan converter. Texture color lookup tables are also supported, enabling hardware support for applications involving dynamic transfer functions such as sensor simulation.

5.2 Pixel Processing

At the next stage of rendering, texture color is used to modify the pre-texture primitive color according to a texture environment equation. Here is where texture can be used to decal, modulate, or blend with the underlying primitive color. The amount of fog at the current pixel is calculated from the depth value, and this fog color is blended with the textured color to produce the final color and transparency.

Pixel processors support 10-bit RGB and 12-bit RGBA components for pixels, providing a selection of more than *68 billion* colors for any given object or vertex. 16-bit luminance-alpha or color index formats are also supported. This accurate color information is critical for image processing applications and is supported through Silicon Graphics ImageVision Library®, as well as OpenGL.

The raster subsystem supports one to four Raster Manager boards, except Onyx2 Reality, which supports up to two Raster Managers. More boards increase the scale of both pixel fill performance and frame buffer resolution. With one RM board installed, Onyx2 InfiniteReality2 systems (not including Onyx2 Reality) support the new ultra-high-resolution standard of 1920x1200 pixels noninterlaced and the full specification for HDTV. The ultra-high-resolution 1920x1200 monitor is included on Onyx2 InfiniteReality2 systems. Onyx2 Reality systems include a 1280x1024 monitor.

6.0 Display Subsystem

The Onyx2 Reality and Onyx2 InfiniteReality2 video display subsystem takes rendered images from the raster subsystem digital frame buffer and processes pixels through digital-to-analog converters (DACs) to generate an analog pixel stream suitable for display on a high-resolution RGB video monitor. The display subsystem supports programmable pixel timings to allow the system to drive displays with a wide variety of resolutions, refresh rates, and interlace/noninterlace characteristics.

The standard configuration of Onyx2 Reality and Onyx2 InfiniteReality2 provides two independent video channels. Eight independent video channels are available as an option. These additional video channels can serve as standard RGB video channels. Channel number one can also serve as the composite or S-Video encoder. Flexible built-in scan conversion allows the video encoder to process any rectangular area of any video channel, up to and including the full screen. Convenient letterboxing lets the composite encoder work undistorted on video formats with aspect ratios other than the 3:4 aspect ratio of NTSC or PAL.

Advanced hardware in the video display subsystem performs real-time video resampling, giving you powerful new methods of guaranteeing scene update-rates and using frame buffer memory efficiently, as well as allowing full-screen recording to standard VCRs. Compatibility with a wide variety of video equipment such as projectors and recorders is provided by composite sync-on-green and separate horizontal and vertical sync signals.

Composite and S-Video encoder outputs are for industrial or monitoring purposes only, and are not intended for broadcast television use. For broadcast-quality composite video, a video channel may be configured as RS-170 or Euro 625 resolution and sent through an external, broadcast-quality encoder. When CCIR601 or broadcast-quality component video is desired, the Digital Video Option (DIVO, described in Section 7.16) may be used. The Graphics-to-Video Option (GVO, described in Section 7.17) card, a daughtercard that attaches to the Display Generator, also supports CCIR601.

The display system provides video support for the Japanese HDTV standard (1920x1080 pixels), both interlaced and noninterlaced, at 60Hz. HDTV trilevel sync is provided for easy recording of HDTV signals without external signal conditioning equipment. Additionally, a special 72Hz noninterlaced version of this format is provided for flicker-free viewing in ambient office lighting environments. The 1920x1080 pixel format is supported on the entry-level, 1-RM configuration of Onyx2 InfiniteReality2. This video flexibility is supported by the standard system monitor, which is a high-resolution multisync display capable of displaying any video signal from VGA (640x480) up to the 1920x1200 format.

Video features of Onyx2 Reality and Onyx2 InfiniteReality2 are easy to configure, using the Video Format Compiler and Video Format Combiner. See Section 7.13.4 Video Format Combiner and Section 7.13.11 Programmable Video Timing for more information.

6.1 ACCESS.bus

Typically, monitors have knobs that are used to adjust brightness, contrast, picture size, position, and color balance. Onyx2 Reality and Onyx2 InfiniteReality2 now use ACCESS.bus to provide this functionality via a program on the CPU or in an X Window interface. ACCESS.bus can also be used to read operational attributes from the monitor.

7.0 Graphics Features and Capabilities

7.1 Resolution

Onyx2 InfiniteReality2 supports a wide range of display resolutions. The base system includes one Raster Manager board and provides 2.62 million pixels. 2-RM and 4-RM systems offer 5.24 million and 10.48 million pixels, respectively. For example, a 1-RM system can drive a 1920x1200 display or two 1280x1024 displays. Alternately, a 2-RM system can drive six 960x720 displays or one 1280x1024 and six 800x600 displays.

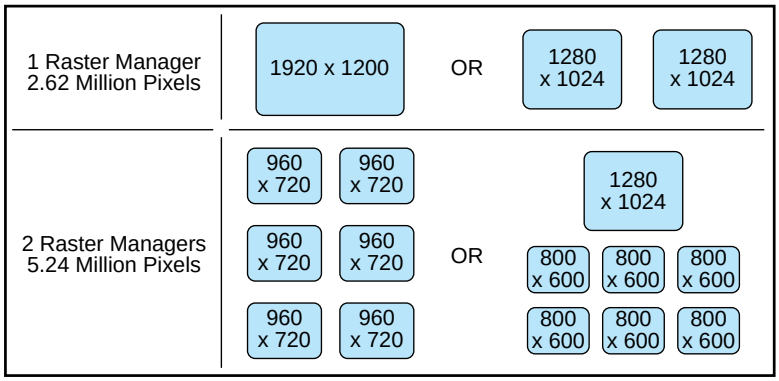


Figure 6. Onyx2 InfiniteReality2 Display Resolution Examples

The minimum amount of frame buffer memory you can allocate per pixel is 256 bits (128 bits on Onyx2 Reality), which supports quad-buffered stereo with up to 12-bit RGBA and a 23-bit depth buffer, or 4-sample antialiasing at 10-bit RGB with a 23-bit precision depth buffer. Per-pixel memory can be doubled or quadrupled, with a corresponding reduction to one-half or one-quarter of available pixels.

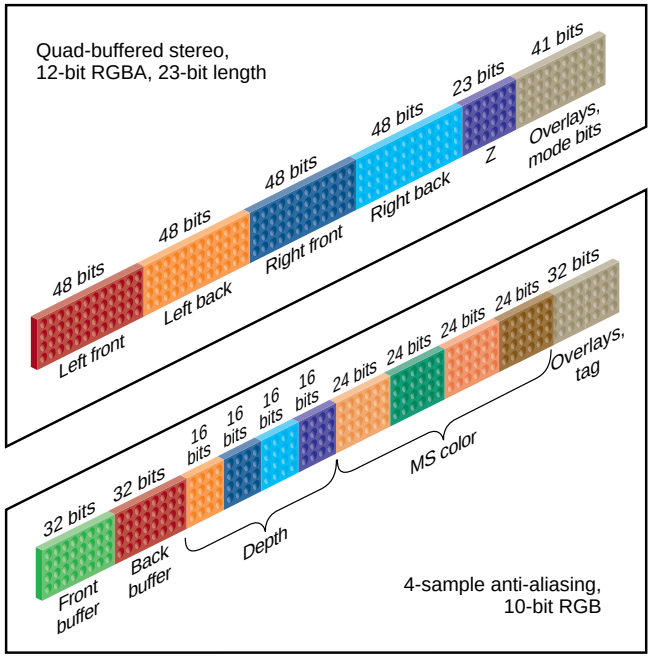


Figure 7. Choices of Allocation of 256 Bits of Frame Buffer Memory

Each Onyx2 Reality raster manager holds half the frame buffer memory as the Onyx2 InfiniteReality2 raster manager. However, the Onyx2 Reality RM can configure pixel sizes as small as 128 bits per pixel, which allows for double-buffered RGB10 plus 32-bit depth. Therefore, a maximum of 2.62 million pixels per raster manager can still be resolved. This number is reduced to half (1.31 million pixels) for 256-bit per pixel configurations, or reduced to one-quarter (640 thousand pixels) for 512-bit per pixel configurations.

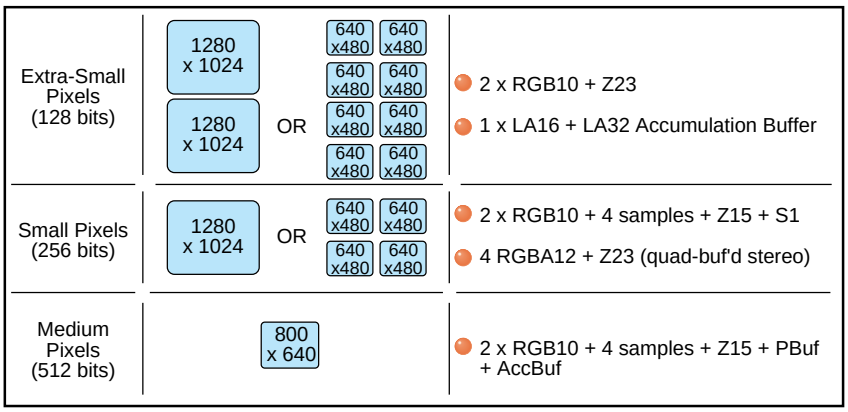


Figure 8. Onyx2 Reality Display Resolution Examples

7.2 Graphics Primitives

The general-purpose Onyx2 Reality and Onyx2 InfiniteReality2 graphics pipeline supports a wide range of graphics primitives for constructing and rendering real-time 3D objects. The basic primitives are polygons, vectors, points, and parametric polynomial surfaces (such as NURBS and Bezier patches). These primitives may be rendered with color, transparency, lighting, texture, and various surface properties.

Complex, contiguous surfaces such as terrain, triangles, or quadrilaterals may be combined into meshed polygon strips with common vertices. This reduces system transformation demands and efficiently uses the inherent parallelism in the Onyx2 Reality and Onyx2 InfiniteReality2 pipelines.

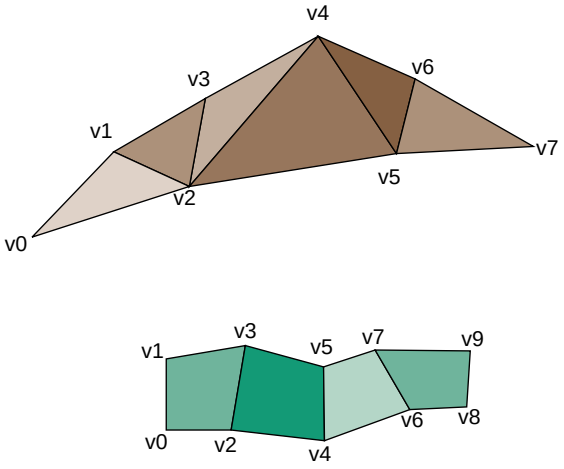


Figure 9. Meshed Polygon Strips

7.3 Immediate Mode and Display List Mode

OpenGL supports two mechanisms for delivering data to the rendering pipeline: immediate mode and display list mode. In immediate mode, transformations, geometry, and other data are delivered to the rendering pipeline directly from the application by calling a series of OpenGL routines. For example:

```
glBegin(GL_TRIANGLE_STRIP);
glVertex3fv(vertex)
...
glEnd()
```

Immediate mode has maximum flexibility because the application can recompute data when sending it to the rendering pipeline.

Additionally, the application can compile immediate mode commands into display lists, stored as part of the OpenGL rendering state. The application can execute display lists wherever it normally uses immediate mode commands. Rendering data is stored in an efficient representation and can be issued to the rendering pipeline with maximum performance. However, extra storage is required for the rendering data; and some small amount of overhead is incurred in executing the list, so lists are not suitable for small amounts of data (a single triangle, for example). Finally, display lists are usually inappropriate for dynamic data because they need to be recompiled whenever data changes. While the basic character of display modes and immediate modes are the same across the Silicon Graphics product line, some features and characteristics specific to Onyx2 Reality and Onyx2 InfiniteReality2 are worth noting. Besides those features listed below, Onyx2 immediate mode performance has doubled relative to the Onyx architecture due to the increased bandwidth to each graphics pipe.

7.3.1 Rendering Performance and Vertex Arrays

Rendering on Silicon Graphics platforms is typically split into a three-stage pipeline:

- Data delivery: host to rendering pipeline
- Geometric processing
- Rasterization

Performance bottlenecks can occur in any of the three stages. Onyx2 Reality and Onyx2 InfiniteReality2 have enhanced performance by providing two to three times the host bandwidth of Onyx InfiniteReality. Additionally, performance may be improved by tuning the algorithms through which the application traverses its data and by using suitable OpenGL commands for delivering data to the pipeline. For example, rather than using floating-point data types for colors, normals, and texture coordinates, it may be more efficient to use short or byte integer data types, reducing the bandwidth requirement to the host memory subsystem and to the rendering pipeline.

Onyx2 Reality and Onyx2 InfiniteReality2 also support vertex arrays, which allow the application to issue primitive data to the Geometry Engine processors in larger transfers and with lower overhead compared to the standard “fine-grain” OpenGL interface. Onyx2 Reality and Onyx2 InfiniteReality2 extends the original OpenGL vertex array extension with support for packed vertex arrays, helping vertex arrays to be dispatched more efficiently while permitting tight packing of vertex coordinates and associated attributes.

7.3.2 Display List Storage Cache and Host Memory

Display lists are an excellent way of ensuring that data is being efficiently delivered to a rendering pipeline. Onyx2 Reality and Onyx2 InfiniteReality2 supports two new features for substantially improved display list performance over immediate mode performance.

The first feature is a large display list storage cache built into the rendering pipeline. This cache is approximately 14MB and can deliver data to the geometry subsystem at more than 300MB/second with very low latency. The cache is automatically managed by a combination of Performer™ and the OpenGL library. Display lists are selected for

placement in the cache based on their size and their content. For example, very short lists will not be put in the cache since the overhead for executing them would outweigh the benefit of placing them in the cache. Lists containing a complex hierarchy will be excluded from the cache as the hierarchy traversal is too complex to deal with directly in the graphics pipeline.

The second display list feature allows display lists to be stored in host memory, but accessed directly by the graphics pipeline. As display lists are compiled they are stored in host memory in a representation suitable for direct interpretation by the Geometry Engine processor. When display lists are executed, the data does not need to be processed by the host CPU nor does it pass through any of the host CPU system caches. On Onyx2 Reality and Onyx2 InfiniteReality2 systems the bandwidth for these direct access display lists is approximately 320MB/second.

7.3.3 Primitive Processing

Onyx2 Reality and Onyx2 InfiniteReality2 have features for helping applications make better use of the Geometry Engine processors. One feature is the ability to perform per-vertex processing, such as viewing and lighting computations, on multiple primitives in one group. When primitives such as triangle strips are issued by the application, the vertices are accumulated and processed in batches. Geometry Engine processors and associated firmware are optimized for a particular batch length, usually around six to 12 vertices. In previous architectures, there could be only one primitive in a batch and a single primitive could span multiple batches. This meant that there was an optimal primitive size, with some performance degradation occurring for primitives of shorter or longer lengths. Now, Onyx2 Reality and Onyx2 InfiniteReality2 can process multiple primitives within the same batch, so it does not incur those performance degradations.

7.3.4 Sprites

Visual simulation applications frequently need to perform computationally intense viewing transformations for drawing sprite objects, such as trees, in which the object rotates as the view changes. Onyx2 Reality and Onyx2 InfiniteReality2 provide a sprite OpenGL extension for performing these complex calculations in the Geometry Engine processors, thus freeing up the host processor for other computations.

7.4 Color

All OpenGL color and lighting models are supported within the Onyx2 Reality and Onyx2 InfiniteReality2 architecture. Lighting operations are performed in floating point in the Geometry Engine processors. Color rasterization operations are performed by the raster subsystem with 48 bits (12 bits each for R, G, B, and A). Color index operations are supported at up to 12 bits per index. Colors may be stored in the frame buffer memory in visuals with as few as 16 bits or as many as 48 bits per color. More bits per color results in greater color fidelity and dynamic range. Special luminance formats allow the display and processing of monochrome images and textures with 16-bit components.

7.4.1 Shading

There is no penalty for smooth shading on objects rendered with Onyx2 Reality and Onyx2 InfiniteReality2. The Gouraud shading algorithm shades the surface of each polygon smoothly, using the current lighting model. This gives you clean, nonfaceted rendering of complex objects in real time and may be combined with transparency and texture for even more advanced rendering effects.

7.4.2 Color Blending

When multiple occluding surfaces have partial transparency, the raster subsystem can also perform color blending between each set of surfaces, depending upon the alpha values associated with each pixel. These alpha values commonly represent a degree of translucency from zero to one and enable simulation of windows, cockpit canopies, smoke, clouds, and other effects. To allow a wider choice of blending equations, a number of OpenGL blending extensions are supported.

7.4.3 Lighting

Hardware lighting capabilities enable the generation of more realistic-looking scenes. Lighting, which is computed per vertex, works with other rendering capabilities, including texture mapping, transparency, and Gouraud shading. All OpenGL lighting features and operations are supported.

7.4.4 Advanced Lighting Model

For creating realistic lighting scenarios, lighting effects must have many different components. Lights may be colored to allow for effects such as time-of-day lighting, where the color of the light as well as the intensity changes over time. Objects being lit may be affected both directly and indirectly by light sources. The OpenGL lighting model on Onyx2 Reality and Onyx2 InfiniteReality2 provides for these characteristics in such a way that objects being lit are affected by a combination of the components shown in Table 1.

Lighting Model Component	Use
Specular	For highlights and simulating shiny surfaces
Diffuse	For broader, directional effects and simulating rough surfaces
Ambient	For environmental, nondirectional effects
Emissive	For self-luminous surfaces, vectors, or points

Table 1. Lighting Model Components

The programmer uses OpenGL commands for setting up the current lighting model, affecting all subsequent surfaces sent down the pipeline until the model is changed. Effects of the components are combined to modify the color of each vertex sent to the graphics pipeline. Lights change the surface color based upon the four components of the current lighting model, the color of the light, and the set surface properties.

7.4.5 Surface Properties

In addition to the four lighting model components, a shininess component may be set for the current lighting model to handle different surface properties. This component may be changed from object to object to allow for a variety of different surface types in the environment. This property, in addition to light characteristics and surface color, may be combined with texture mapping and transparency to give you the maximum flexibility in defining surface reflectance characteristics.

7.4.6 Infinite Light Sources

Infinite light sources are assumed to be infinitely far from the surface they are illuminating; thus, light “rays” from these sources are assumed to arrive in parallel at all the surfaces affected.

7.4.7 Local Light Sources

Local light sources may exist close to the surfaces they are illuminating; thus, the angle between the light source and the normals of the affected surfaces can change dramatically from one surface to the next. Properties such as cutoff angle and attenuation may be specified for each light.

Since angles from surface normals to the light source must be computed every frame for each surface, local lights are far more computationally expensive than infinite lights and should be used judiciously in real-time applications.

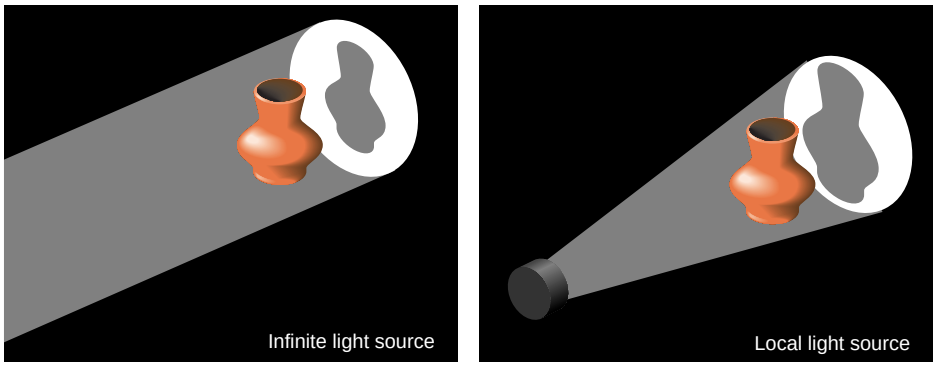


Figure 10. Infinite and Local Light Sources

7.5 Transparency

Each vertex to be rendered may have a transparency component, alpha, along with the standard color components red, green, and blue. When alpha is used to control the pixel color, objects of varying transparency or translucency may be included in a rendered scene. To ensure that transparency is computed correctly and predictably, it is best to sort transparent objects from front to back with respect to the eyepoint, then draw these objects after all the nontransparent objects have been rendered.

7.6 Hidden Surface Removal

Current Silicon Graphics machines use a Z-buffering technique to remove hidden surfaces. The Z (depth) buffer gives high-quality hidden surface removal during rendering without sorting the polygons. This process can thus be performed with few or no programming considerations in the software that traverses and renders the database.

Silicon Graphics floating-point Z-buffer technique for Onyx2 Reality and Onyx2 InfiniteReality2 is advancing the state of the art by offering more uniform resolution across the Z dimension. Unlike traditional floating point, which would concentrate resolution closer to the eyepoint, Onyx2 systems distribute useful resolution throughout the viewing volume. Onyx2 Reality and Onyx2 InfiniteReality2 offer three options for Z-buffer precision: 23-bit (not available on Onyx2 Reality when anti-aliasing [i.e. multisampling] is enabled), 16-bit compressed, and 15-bit compressed. The Z-buffer exists as dedicated memory in the frame buffer associated with each pixel. The first time a pixel is drawn, its Z-buffer value is set to the Z of the primitive being drawn. Each time a subsequent attempt is made to set the color of that pixel because a new polygon is covering it, the system checks to see if that pixel's Z-buffer value is further or closer than the Z value of the pixel being drawn. If the new pixel is closer, its color value replaces the previous value in the color buffer, and the Z-buffer is set to the new depth.

7.7 Coplanar Geometry Prioritization

There are several methods which may be used to support the rendering of multiple geometric layers of polygons which are coplanar. This situation arises when additional detail such as text or decals or other markings need to be added to a base polygon. Using a conventional Z-buffering scheme coplanar surfaces might 'Z-fight' because the hardware cannot decide which face has visual priority, for example in the case of runway markings the hardware cannot decide whether the markings are on top of the runway or vice versa.

Onyx2 Reality and InfiniteReality2 supports three methods of quickly assigning visible priority in these situations. Firstly with a combination of depth buffer offsets and depth buffer writemasking a subface depth may be offset after projection without other visible effects and therefore assume visibility priority, this offset increases with the slope of the polygon to match and correct the characteristics of the depth buffer Z-fighting.

Secondly, fast stencil operations in conjunction with framebuffer stencil bits allow base polygons to be marked in the framebuffer stencil planes where visible. Subsequent rendering of subfaces tests for the base polygon stencil value instead of the Z-buffer value, this allows perfect subface rendering without any undesirable artifacts.

The final method of setting visible priority supported on Onyx2 Reality and Onyx2 InfiniteReality2 allows subfaces to be rendered with Z-buffer information generated from the plane equation of the base polygon. The decal pixel fragments therefore exactly match the base polygon depth values and therefore avoids any zbuffer conflicts giving the subfaces visible priority over the base polygon.

7.8 Anti-Aliasing

The high-performance anti-aliasing hardware of Onyx2 Reality and Onyx2 InfiniteReality2 continues the trend started with the RealityEngine2™ graphics architecture. For the highest-available image quality, memory and processors are put in the Raster Manager subsystem to give you multisampled anti-aliasing without the usual performance penalties. This anti-aliasing technique needs no data sorting and works with the Z-buffer for superior hidden surface removal. The Onyx2 Reality system provides limited multisample anti-aliasing as described in Section 7.8.2.

Subpixel information is retained for all vertices in the pipeline as they are transformed and converted to raster primitives. Each primitive is computed with 8x8 subpixel accuracy. Thus, there are 64 possible subpixel locations within each pixel rendered. When deciding how to color a pixel, the raster system samples each primitive at one, four, or eight of the subpixel locations in each pixel it touches. It then calculates color and depth information for the subpixels covered. This information is sent to the frame buffer where Z-buffering is performed on each subpixel, and a final color for each pixel is formed from the accumulated data from the set of subpixels within it.

In multisample anti-aliasing mode, Z information is kept for every subsample, giving hidden-surface accuracy down to the subpixel level. When transparency is used, the alpha value is used to determine how much to blend the color of the closer pixel with the color of the farther pixel at that same location in the subpixel grid.

7.8.1 Sample Memory

The number of anti-aliasing samples available depends upon the total amount of frame buffer memory, the screen resolution, and the number of bits per image component desired. Table 2 shows some of the possible memory configurations for Onyx2 InfiniteReality2.

One RM provides sufficient memory for:				
Medium pixels, 512 bits/pixel	One 960x680 display	8 samples	10 bits/component for RGB	16-bit Z-buffer, 8-bit stencil
Medium pixels, 512 bits/pixel	One 1280x1024 display	8 samples	10 bits/component for RGB	23-bit Z-buffer*, 1-bit stencil
Small pixels, 256 bits/pixel	Two 1280x1024 displays	4 samples	10 bits/component for RGBA	15-bit Z-buffer*, 1-bit stencil
Small pixels, 256 bits/pixel	One 1920x1080 display	4 samples	10 bits/component for RGB	15-bit Z-buffer*, 1-bit stencil
Two RMs provide sufficient memory for:				
Medium pixels, 512 bits/pixel	One 1280x1024 display	8 samples	10 bits/component for RGB	16-bit Z-buffer, 8-bit stencil
Medium pixels, 512 bits/pixel	One 1600x1200 display or one 1920x1080 display	8 samples	10 bits/component for RGB	23-bit Z-buffer*, 1-bit stencil
		4 samples	12 bits/component for RGBA	23-bit Z-buffer*, 8-bit stencil
Small pixels, 256 bits/pixel	Two 1920x1080 displays	4 samples	10 bits/component for RGB	15-bit Z-buffer*, 1-bit stencil
Four RMs provide sufficient memory for:				
Medium pixels, 512 bits/pixel	Eight 800x600 displays	8 samples	10 bits/component for RGB	23-bit Z-buffer*, 1-bit stencil
Medium pixels, 512 bits/pixel	Three 1280x1024 displays	8 samples	10 bits/component for RGB	23-bit Z-buffer*, 1-bit stencil
Medium pixels, 512 bits/pixel	Two 1920x1080 displays	8 samples	10 bits/component for RGB	23-bit Z-buffer*, 1-bit stencil
Medium pixels, 512 bits/pixel	One 1920x1200 display	8 samples	10 bits/component for RGB	16-bit Z-buffer 8-bit stencil

* A 23-bit or 15-bit Silicon Graphics Z-buffer implementation is roughly equivalent to a classic 32-bit or 24-bit Z-buffer, respectively.

Table 2. Anti-Aliasing Sample Memory Options for Onyx2 InfiniteReality2

7.8.2 Onyx2 Reality Restrictions

The Onyx2 Reality system provides limited multisampling capacity which when enabled will reduce pixel fill performance by a factor of 1/2 or 2/3. This contrasts with the high performance anti-aliasing of Onyx2 InfiniteReality2 graphics. The only color format supported is RGB10 (no destination alpha planes), and 16-bit compressed depth (or 15-bit with one-bit stencil). Furthermore, only four samples are maintained in the frame buffer. This configuration requires 256 bits per pixel, and on a 1RM system will support up to a 1280x1024 monitor. To support monitors larger than 1.25 million pixels, two RMs are required. There is no option for 8-sample anti-aliasing on Onyx2 Reality.

Table 3 shows some of the possible memory configurations for Onyx2 Reality.

One RM provides sufficient memory for:				
Medium pixels, 512 bits/pixel	One 960x680 display	4 samples	10 bits/component for RGB, quad-buffered, accumulation buffer	15-bit Z-buffer*, double buffered, 1-bit stencil
Small pixels, 256 bits/pixel	One 1280x1024 display	4 samples	10 bits/component for RGB	15-bit Z-buffer*, single buffered, 1-bit stencil
Extra-small pixels, 128 bits/pixel	Two 1280x1024 displays	1 sample	10 bits/component for RGB, double buffered	23-bit Z-buffer*, 1-bit stencil
Small pixels, 256 bits/pixel	One 1280x1024 display	1 sample	12 bits/component for RGBA, double buffered, or quad-buffered RGB10	23-bit Z-buffer*, 8-bit stencil
Two RMs provide sufficient memory for:				
Small pixels, 256 bits/pixel	Two 1280x1024 displays	4 samples	10 bits/component for RGB	15-bit Z-buffer*, 1-bit stencil
Small pixels, 256 bits/pixel	One 1600x1200 display	4 samples	10 bits/component for RGB	15-bit Z-buffer*, 1-bit stencil

* A 23-bit or 15-bit Silicon Graphics Z-buffer implementation is roughly equivalent to a classic 32-bit or 24-bit Z-buffer, respectively.

Table 3. Anti-Aliasing Sample Memory Options for Onyx2 Reality

7.9 Texture/Image Mapping

Designed to display complex, texture-mapped scenes at real-time frame rates (60Hz), Onyx2 Reality and Onyx2 InfiniteReality2 remain compatible with earlier RealityEngine2 and POWERVision™ systems, while dramatically improving performance and texture capacity and adding new features and functionality to texture mapping.

7.9.1 Texture Download

Onyx2 Reality and Onyx2 InfiniteReality2 download and draw textures simultaneously, allowing textures to be updated on-the-fly. Synchronization, built into the hardware and software, prevents textures from being replaced until the system finishes with them and prevents a texture being drawn until it is completely loaded. The maximum texture download rate is 330MB/second from host memory or 240MB/second from frame buffer. Unlike host download operations the bottleneck for the texture readback from the frame-buffer is the pixel path from the Raster Manager boards to the Geometry Engine board and is influenced by the type of visual used. See Table 4.

Source	16 bit	32 bit	48 bit
Host Memory	144 M texels/second	80 M texels/second	42M texels/second
Frame Buffer Memory	85 M texels/second	85 M texels/second	47 M texels/second

Table 4. Texture Rate for 1024x1024 Texture (RGB10 visual for framebuffer read)

7.9.2 Texture Formats

Onyx2 Reality and Onyx2 InfiniteReality2 support three basic texture element (texel) sizes: 16-bit, 32-bit, and 48-bit, including 16-bit monochrome. Each size can represent several formats:

16-bit textures

- Dual 4-bit Luminance with 4-bit alpha
- Quad 4-bit Luminance, Intensity, or alpha
- Dual 8-bit Luminance, Intensity, or alpha
- 8-bit Luminance with 8-bit alpha
- 12-bit Luminance, Intensity, or alpha
- 12-bit Luminance with 4-bit alpha
- 5-bit (each) RGB with 1-bit alpha
- 4-bit (each) RGBA
- 16-bit Luminance, Intensity, or alpha

32-bit textures

- 16-bit Luminance with 16-bit alpha
- 12-bit Luminance with 12-bit alpha
- 8-bit (each) RGBA
- 10-bit (each) RGB with 2-bit alpha

48-bit textures

- 12-bit (each) RGB
- 12-bit (each) RGBA

The variety of formats allows you to trade off image quality against storage size and rendering speed; 32-bit textures require twice the storage of 16-bit textures, while 48-bit textures require four-times the storage of 16-bit textures. Generally, 16-bit textures render faster than 32-bit textures, and 32-bit textures render faster than 48-bit textures for a given filtering mode.

The Texture Select feature is useful when vast amounts of low-resolution texture are required. With Texture Select, four 4-bit textures or two 8-bit textures are packed into one 16-bit texture. The user may select which of the four (or two) textures are used at any given time.

7.9.3 Texture Filtering

One of the most powerful features of the Onyx2 Reality and Onyx2 InfiniteReality2 architecture is found in the variety of advanced two-dimensional and three-dimensional texture filtering techniques that are supported. This variety allows you to choose the most appropriate filtering technique for your application.

The filtering techniques comprise two different classes: MIPmapped and non-MIPmapped. The term MIPmapping is derived from the Latin phrase, *multum in parvo*, which means “many things in a small space.” The MIPmapping process takes a texture (image) to be used in a scene and generates a series of prefiltered, lower-resolution versions of that texture. The appropriate level of the MIPmapped texture is then applied to a polygon as required by the scene. This allows the texture to be minified without aliasing artifacts. Different MIPmapped filtering techniques are then available to calculate the transitions between different MIPmap levels and between different texels (texture elements) within a level. These filtering techniques include:

- Point sampled
- Linear interpolated
- Bilinear interpolated
- Trilinear interpolated
- Quadlinear interpolated (for 3D textures)

Non-MIPmapped textures require filtering only among the different texels within the texture map. These filtering techniques include:

- Point sampled (for 2D or 3D textures)
- Bilinear interpolated (for 2D textures)
- Trilinear interpolated (for 3D textures)
- Bicubic interpolated (for 2D textures)

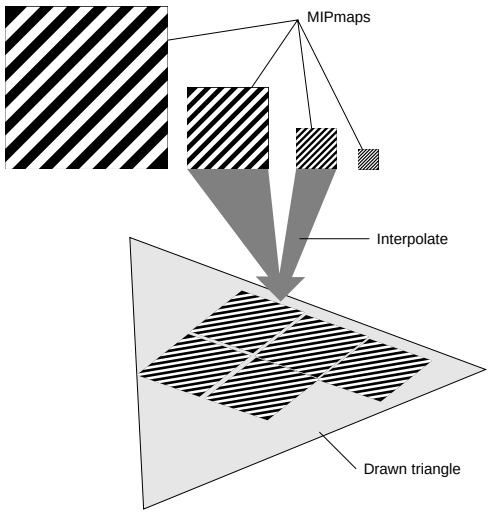


Figure 11. MIPmapping

Onyx2 Reality and Onyx2 InfiniteReality2 are designed to provide all of the filtering techniques listed above at full texture processing performance levels, except for quad-linear interpolated MIPmapping and bicubic interpolated filtering, which operate at half the rate of the others. It is important to recognize that trilinear interpolated non-MIPmapped 3D texture filtering is available at full texture processing performance levels. This represents a significant performance improvement compared to RealityEngine2, which processed this filtering technique at one-half the peak performance level of other RealityEngine2 texture filters.

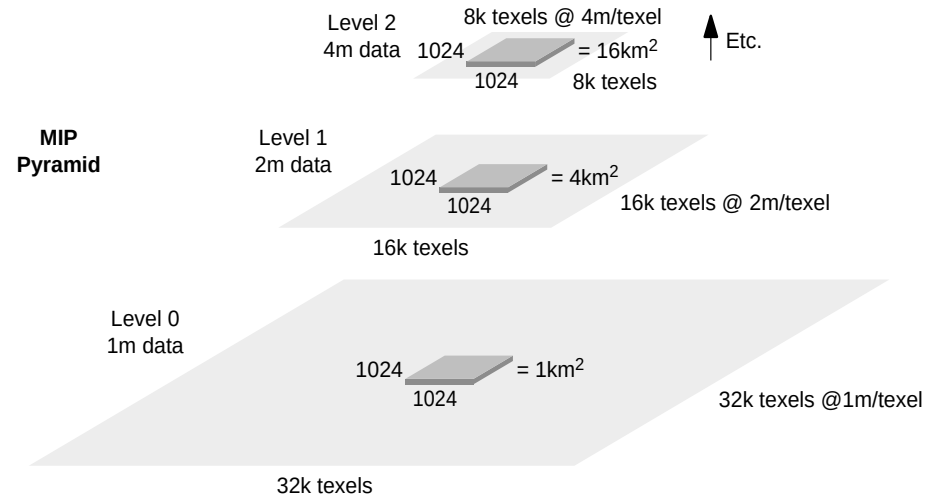
7.9.4 Global Texturing a.k.a. CLIP mapping

Global texturing is an important and innovative feature of Onyx2 Reality and Onyx2 InfiniteReality2. Global texturing accelerates processing of terrain data mapped with satellite and aerial photographs. In RealityEngine2, traditional MIPmapping techniques were used to apply aerial photographs to terrain meshes. Large geographic areas were covered by tiling a number of smaller MIPmaps together across the terrain area. This technique, though powerful, required a large number of textures and a significant amount of texture memory. Unused texture tiles were stored in main memory and downloaded into the RealityEngine2 texture buffer when required. In this case, textures were either stored in system memory or texture memory, but they could not be split between the two.

In Onyx2 Reality and Onyx2 InfiniteReality2, global texturing allows you to define single MIPmap images which are much larger than that which can be stored in texture memory. Only the parts of the MIPmap that are actually used in rendering an image are stored in texture memory; the rest is stored in main memory or on disk. As the point of view moves, the portion of the MIPmap that is stored in memory is updated to reflect the new point of view.

The implementation is effective for real-time simulation because the paging of texture is performed in anticipation of the requirements of the visual scene. These requirements are directed by the eye position before the texture is required. This avoids the problems of paging on demand which would stall the graphics system waiting for essential texture information from system memory or disk. At the same time the terrain is easily modeled with just a single texture image used to represent the entire database without tiling

The paging is performed in real-time for each level of MIP in the image pyramid which is deemed too large to fit in texture memory. So one can imagine a requisite subset of each image in the MIP map pyramid cached in texture memory, lets say we store a 1024x1024 image for each MIP texture image. If our base image held 1 meter resolution data then the texture hardware holds 1 square kilometer of information at highest resolution. When CLIP mapping the next level of MIP would also hold a 1024x1024 resolution image cache but because this MIP level is lower resolution that same amount of data covers a 2km by 2km database area with 2m texel data. This continues for each successive level of MIP until the entire texture image is held in texture memory at some ground resolution, this would be where a 1024x1024 image held the whole geographic database. Each 1024x1024 image held in memory covers a concentric region of interest around some important point in the database, normally this point is the center of the area the graphics system is currently rendering at highest detail for this frame. As the eye moves over the database the trailing border of each image cache is replaced by the information required by the leading edge, and the texture addressing is torroidally displaced to account for the newly downloaded image data.



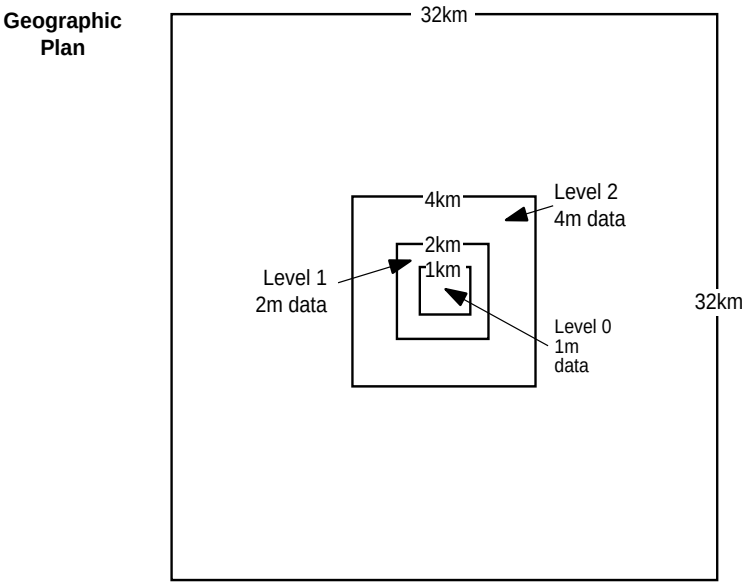


Figure 12. CLIP Mapping

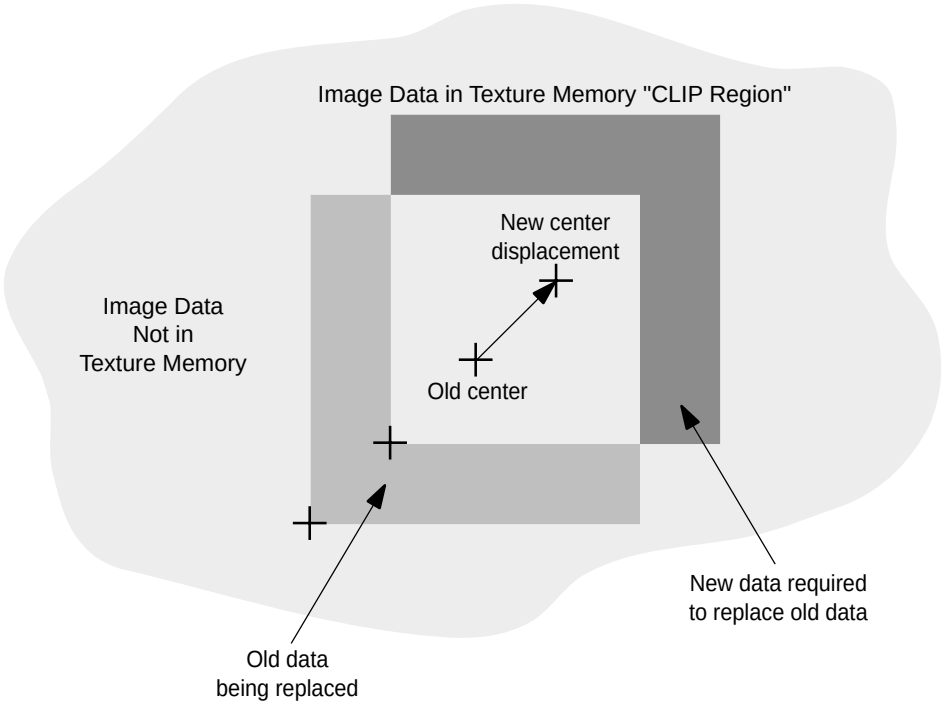


Figure 13. CLIP Mapping

7.9.5 Add, Replace, Blend, Decal, and Modulation

Each texture can be treated in several ways when applied to a polygon. When operating with an add texture environment the texture color is added to the result of the lighting calculation or polygon color with an additional color multiplier thrown in for added flexibility. With a replace or decal environment the color of the texture replaces inherent color to the polygon, alpha treatment varies between these two modes and mix either replaces the polygon alpha or is used as a blender for the degree to which the rgb replaces the polygon color. The blend uses the texture color as a blending value for the mixing of the polygon color and any specified blend color so you can use a texture image as a matte between the polygon color or lighting result and any other color. During modulation, the texture color is multiplied by the lighting result or color of the polygon which gives the classic textured lit appearance seen in most graphics images.

7.9.6 Texture/Image Memory

Each RM can have either 16MB or 64MB of texture memory. For non-MIPmapped textures this means that the texture memory can hold 8 million or 32 million 16-bit texels respectively. MIPmapped textures require extra storage for the prefiltered lower levels of detail, amounting to one-third the size of the finest level of detail for 2D and one-seventh the size of the finest level of detail for 3D.

Texture memory is divided into two banks for performance reasons. Because of this, the maximum size texture does not always fill up texture memory. With the exception of bicubic, an image, a level of a MIPmap, or slices of a volume cannot cross a texture memory bank. The largest 16-bit texture maps that fit into Onyx2 Reality and Onyx2 InfiniteReality2 texture memory are shown in Table 5 Other aspect ratios with the same number of pixels will also fit.

Texture Memory	2D Non-MIPmap	2D MIPmap	2D Bicubic	3D Non-MIPmap	3D MIPmap
16MB	2048x2048	2048x1024	4096x2048	256x256x128	256x128x128
64MB	4096x4096	4096x2048	8192x4096	512x256x256	256x256x128

Table 5. Largest Available16-Bit Texture Maps

Table 6 shows the number of 16-bit textures that can be stored in 16MB of texture memory. To convert to the number of 32-bit or 48-bit textures, divide the numbers in the last two columns by 2 or 4, respectively. Table 7 lists the same information for several 3D textures.

Texture Size	Number of 16-Bit MIPmapped Textures in 16MB of Texture Memory	Number of 16-Bit Non-MIPmapped Textures in 16MB of Texture Memory
2048x2048	0	2
1024x1024	4	8
512x512	22	32
256x256	94	128
128x128	382	512
64x64	1,522	2048
32x32	5,948	8192
16x16	21,732	32,784

Table 6. 2D Texture Memory Sizes and Quantities

Texture Memory	Resolution	Number of 16-Bit Textures
16MB	128x128x128	4 (non-MIPmapped)
16MB	128x128x128	2 (MIPmapped)
16MB	256x256x128	1 (non-MIPmapped)
16MB	256x128x128	1 (MIPmapped)
64MB	256x256x256	2 (non-MIPmapped)
64MB	256x128x128	7 (MIPmapped)
64MB	256x256x512	1 (non-MIPmapped)
64MB	256x256x256	1 (MIPmapped)

Table 7. 3D Texture Sizes for 16-Bit Texture

7.9.7 Texture Memory Packing

Although supported texel sizes pad to 16 bit sizes, the Onyx2 InfiniteReality2 and Onyx2 Reality graphics systems support the packing of multiple smaller format images into the same memory space so for example an image with 16 bit texels can pack two 8 bit images into a single 16 bit per texel image. This feature allows the efficient use of texture memory for small format images, particularly single component images.

7.9.8 Texture Transparency and Contouring

Textures may have full or partial transparency set at any texel location. If the entire outer edge of the uniquely shaped texture, such as a tree, is set to transparent, the texture may be placed upon a rectangular polygon and still appear to have the outline of a tree.

7.9.9 Perspective Correction

The raster subsystem performs per-pixel computations to correct textures and fog for perspective distortions to ensure that no artifacts are introduced to the rendering process. These computations do not affect system performance.

7.9.10 Detail Texture

Detail Texture increases the effectiveness of motion and position cues by providing additional detail on top of the existing texture as the textured surface draws nearer to the eyepoint. Detail Texture allows high-resolution textures to be represented with a small high-frequency characteristic texture and a lower resolution-base texture. The high-frequency detail is smoothly combined with the base texture to give much of the visual qualities of the original texture, while using a fraction of the texture memory.

7.9.11 Sharp Texture

Textures enable creation of highly realistic scenes, largely due to the ability to import photographic images, which can be applied to geometry. The resolution of these images is finite, however. When they are enlarged to the point where the textured polygon is larger than the original texture resolution, magnification tends to blur the original image.

The solution is the patented Sharp Texture feature in the Onyx2 Reality and Onyx2 InfiniteReality2 subsystem, which lets you specify that edges remain sharp during magnification. For example, this enables the development of a simulation whose photo-derived sign textures would continue to be readable as they get close to the viewer's eyepoint.

7.9.12 Projective Texture

With Projective Texture, textures can be projected into a scene, bending over every surface. This allows for sophisticated per pixel spotlighting effects, such as landing lights, and projection of camera data onto synthetic terrain.

7.9.13 Shadow Maps

Onyx2 Reality and Onyx2 InfiniteReality2 provide hardware support for shadowing. Applying shadows is a two-step process: the scene of interest is first rendered from the light source point of view; the depth information from the scene is then copied into texture memory. The result is a shadow map. The scene is then rendered from the eyepoint. Hardware determines which parts of the scene are in shadow and which are not, based on the shadow map, and lights them accordingly. The approach can be extended to multiple light sources by rendering in multiple passes.

7.9.14 Billboards

Billboards are textured faces that automatically track the eyepoint so that they are always facing the viewer. The billboard feature is commonly used for trees, and other complex but approximately symmetric objects.

7.9.15 Detail Management

You can restrict which MIPmap levels of detail are used. This allows use of a partially loaded MIPmap. If a MIPmap is loaded starting at the coarsest level, the texture can be used before its finest levels of detail have been loaded.

7.9.16 Video to Texture

Using the Digital Video Option, broadcast-quality live video may be brought into host memory in real time with very low latency and in formats compatible with OpenGL. This image data stream may then be sent to texture or raster memory for processing by the graphics pipeline. This provides the capability to perform true three-dimensional (3D) texture mapping with a live video source. In addition, image processing or other manipulations may be performed on the video data for image analysis or video effects.

7.9.17 3D Textures and Volume Rendering

Onyx2 Reality and Onyx2 InfiniteReality2 include the advanced 3D Texture feature, which enables real-time 3D volume rendering. For example, 3D Texture could be used to view a series of two-dimensional medical scans of a patient's head.

Onyx2 Reality and Onyx2 InfiniteReality2 have the power to reconstruct the series of two-dimensional scans into a three-dimensional volume. The volume can be manipulated (rotated, translated, filtered, and more) in real time by continuously resampling (interpolating) the data in 3D. Volume rendering can be enhanced by using the following features:

- Slicing planes
- Perspective projection
- Embedded surfaces

For more information, see Fraser, Robert, "Interactive Volume Rendering Using Advanced Graphics Architectures," Silicon Graphics Computer Systems.

7.9.18 Texture Color Lookup Tables

Onyx2 Reality and Onyx2 InfiniteReality2 support texture color lookup tables without any performance penalty. This table can change in real time to achieve selection in volume rendering or transfer functions for sensor simulation. The hardware support is for both monochrome and color functions and has 12-bit precision.

7.10 Imaging Operations

The Geometry Engine processors of Onyx2 Reality and Onyx2 InfiniteReality2 provide hardware accelerated imaging operations, including convolution, minmax, histogram, color matrix, and lookup tables. Because all pixel processing paths, including drawing, copying, and reading of pixels, texels, and video data, flow through the Geometry Engine processors, these image operators can be applied between any source and destination. For example, an application may copy pixels from a video source to texture memory and perform color space conversion using the color matrix in one pixel transfer operation.

7.10.1 Convolution

The convolution operator filters linear images in real time. The user-definable filter performs operations such as sharpening and blurring. Onyx2 Reality and Onyx2 InfiniteReality2 support separable and general convolution filters of size 3x3, 5x5 and 7x7. Separable filters are much faster than general filters. Likewise, smaller filters are faster than larger filters. Applications can trade off quality and throughput by choosing an appropriate filter size and type.

7.10.2 Histogram and Minmax

Histograms and minmax provide statistics on color component values within an image. A histogram counts occurrences of specific color component values and reports them in user-defined bins with up to 4,096 bins per component. Minmax tracks minimum and maximum color component values. These statistics can be analyzed to create a more balanced, better-quality image.

7.10.3 Color Matrix and Linear Color Space Conversion

The color matrix operation transforms the colors of each pixel by a user-defined 4x4 matrix. In addition to performing linear color space conversion, the color matrix can also be used to perform color component swapping and component replication.

7.10.4 Window/Level Support

Remote sensing, medical imaging, and film editing applications can process data with a greater dynamic range than a video monitor can display. The *window/level* operation allows a subset of the data's full dynamic range to be selected (windowed) and its contrast and brightness adjusted (leveled) so that important image features will be seen on a monitor.

The frame buffer and imaging data path of Onyx2 Reality and Onyx2 InfiniteReality2 support display and texturing using 16-bit luminance data, which is appropriate for demanding, high-precision imaging applications. The 32K-entry color lookup table on the Display Generator board enables real-time window/level operations on multiple windows over 12-bit subranges without requiring images to be redrawn.

7.10.5 Lookup Tables (LUTs)

Color tables can be inserted in various stages of the pixel path to adjust image contrast and brightness after an image operation. Each color table can have up to 4,096 entries per color component and can be loaded directly from host or frame buffer memory.

7.11 Atmospheric Effects

A programmable fog/haze function in Onyx2 Reality and Onyx2 InfiniteReality2 allows setting an arbitrary fog color that can be blended with the color of any element to be rendered. The fog may be changed for all or part of the scene on a per-channel or per-object basis. The amount of fog or haze color used depends on the depth value of each pixel of each selected object.

The user can specify the falloff function and the density of the fog. Onyx2 Reality and Onyx2 InfiniteReality2 also allow specifying a fog function using sample points.

Compared to RealityEngine2, the precision and the smoothness of fog/haze appearance in Onyx2 Reality and Onyx2 InfiniteReality2 is greatly improved. Based on the viewing projection, depth range, and the fog function itself, the system automatically optimizes generation of per-pixel fog values. Each generated fog value has 12-bits of precision per color component.

7.12 Offscreen Rendering

Onyx2 Reality and Onyx2 InfiniteReality2 systems have abundant frame buffer memory—80MB on each InfiniteReality2 Raster Manager board and 40MB on each Onyx2 Reality Raster Manager board—used to store pixels for very-high-resolution displays, or for multiple video output channels. It also may be used to store pixels for off-screen rendering.

A frame buffer pixel contains from 256 to 1,024 bits (128 to 512 bits for Onyx2 Reality), depending upon the system's Raster Manager board quantity and video output channel characteristics. Some bits are dedicated to front and back color buffers (when applications use double buffering), the depth (Z) buffer, and other buffers supported by OpenGL (see Figure 14).

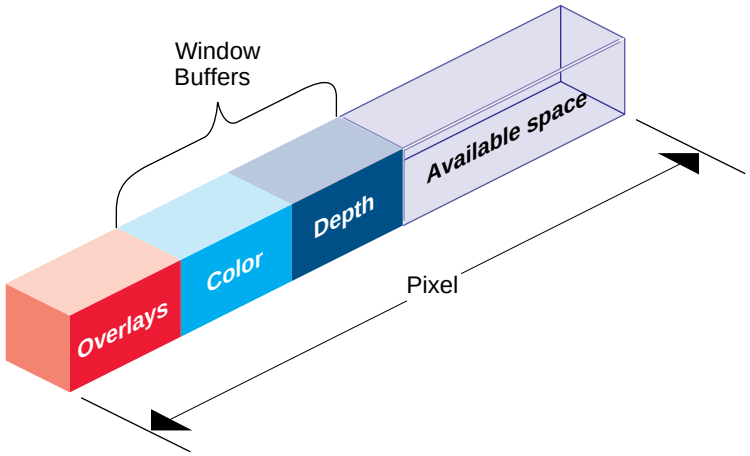


Figure 14. Frame Buffer Pixel Memory

Remaining memory is available for other uses. Applications may invoke the *SGIX_fbconfig* OpenGL extension to choose a *frame buffer configuration* for unassigned memory (see Figure 15).

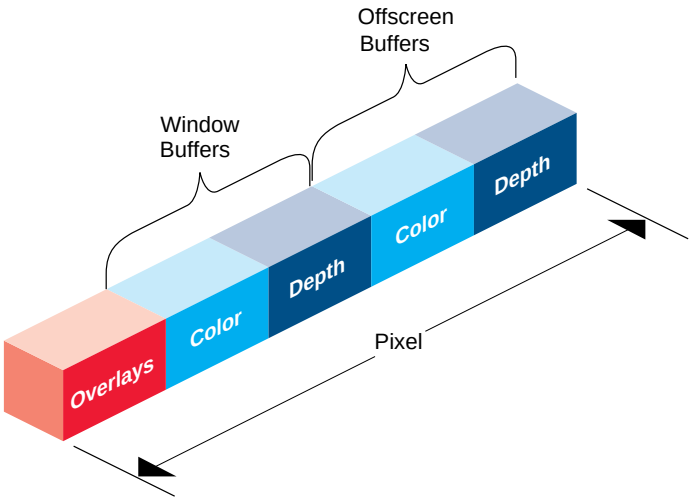


Figure 15. Frame Buffer Pixel Memory Including Offscreen Buffers

After selecting a frame buffer configuration, applications may invoke the *SGIX_PBuffer* OpenGL extension to create an offscreen rendering area called a *PBuffer* (Pixel Buffer). PBuffers occupy part of the frame buffer not otherwise occupied by ordinary windows (see Figure 16).

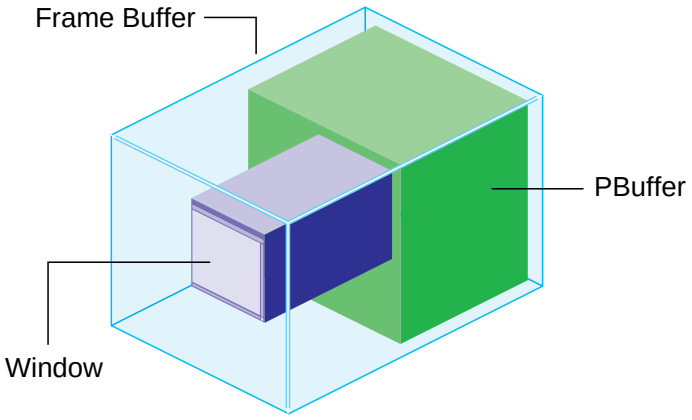


Figure 16. Disjoint Buffers and PBuffers

PBuffers are much like windows; OpenGL applications may render into PBuffers just as they render into windows. However, PBuffers never obscure color buffers of windows and are never displayed on any video output channel.

PBuffers may serve as a cache for frequently accessed images, improving performance dramatically by removing the need to transfer pixels from host memory to the graphics pipeline. They may be used for storing results of special-purpose renderings that are not intended for display, such as shadow maps. PBuffers are also convenient repositories for the intermediate results of multipass rendering algorithms, which produce effects such as reflections, Phong shading, and bump mapping. Finally, PBuffers permit Onyx2 Reality and Onyx2 InfiniteReality2 to accelerate background batch processing of long-running imaging operations without disabling interactive use of the main display.

7.13 Multi-Channel Display Generator

The Display Generator takes digital, ordered pixel output from the frame buffer and lets you specify from two to eight separate rectangular areas to be sent from the rectangular frame buffer area managed by the x-server to independent component RGB video outputs. Each video channel may have its own video timing (subject to constraints detailed in Section 7.13.3), increasing flexibility over previous products.

7.13.1 Standard Formats and Format Combinations

To provide as much built-in flexibility as possible, Onyx2 Reality and Onyx2 InfiniteReality2 systems are delivered with a broad range of software-selectable video formats and format combinations, from NTSC to 1920x1080 progressive-scan HDTV. Table 8 and Table 9 show Onyx2 Reality and Onyx2 InfiniteReality2 formats shipped standard with the system.

Video Format	Video Format File	Notes
640x480 120Hz stereo	640x480_120s.vfo	
640x480 180Hz field sequential	640x480_180q.vfo	Color field sequential 180Hz format
640x480 60Hz	640x480_60.vfo	VGA
646x486 30Hz interlaced	646x486_30if.vfo	Frame locking format
646x486 30Hz interlaced	646x486_30i.vfo	NTSC
768x576 25Hz interlaced	768x576_25i.vfo	PAL
768x576 25Hz interlaced	768x576_25if.vfo	Frame locking format
800x600 60Hz	800x600_60.vfo	SVGA
960x620 60Hz	960x620_60.vfo	
960x680 60Hz	960x680_60.vfo	
1024x768 120Hz stereo	1024x768_120s.vfo	
1024x768 60Hz	1024x768_60.vfo	
1024x768 96Hz stereo	1024x768_96s.vfo	

Continued on page 43

Continued from page 42

Video Format	Video Format File	Notes
1080x809 30Hz interlaced	1080x809_30i.vfo	RS343-A “875-line” format
1120x840 96Hz stereo	1120x840_96s.vfo	
1200x900 72Hz	1200x900_72.vfo	High-resolution 4:3 format
1280x1024 114Hz stereo	1280x1024_114s.vfo	
1280x1024 120Hz stereo	1280x1024_120s.vfo	Ultra-high-resolution stereo
1280x1024 25Hz PAL	1280x1024_25r2.vfo	Special format
1280x1024 25Hz PAL	1280x1024_25r3.vfo	Special format
1280x1024 30Hz NTSC	1280x1024_30r2.vfo	Special format
1280x1024 50Hz	1280x1024_50.vfo	
1280x1024 60Hz	1280x1024_60.vfo	High-resolution
1280x1024 72Hz	1280x1024_72.vfo	
1280x959 30Hz interlaced	1280x959_30i.vfo	
1500x1200 60Hz	1500x1200_60.vfo	
1600x1200 60Hz	1600x1200_60.vfo	Extra-high-resolution
1760x1100 60Hz	1760x1100_60.vfo	
1920x1035 30Hz interlaced	1920x1035_30i.vfo	Japan HDTV
1920x1080 72Hz	1920x1080_72.vfo	US HDTV
1920x1200 66Hz	1920x1200_66.vfo	New ultra-high-resolution
CCIR601_525	CCIR601_525.vfo	
CCIR601_625	CCIR601_625.vfo	

Table 8. Standard Formats Shipped with all Onyx2 Reality and Onyx2 InfiniteReality2 Systems

Standard Combinations	Combiner File
1 1280x1024 60Hz	1280x1024_60.cmb
1 1280x1024 72Hz	1280x1024_72.cmb
1 960x680 60Hz	960x680_60.cmb
1 1024x768 60Hz	1024x768_60.cmb
1 1200x900 72Hz	1200x900_72.cmb
1 1600x1200 60Hz	1600x1200_60.cmb
1 1920x1035 30Hz interlaced	1920x1035_30i.cmb
1 1920x1080 72Hz	1920x1080_72.cmb
1 1920x1200 66Hz	1920x1200_66.cmb
1 1280x1024 60Hz and 1 NTSC	1280x1024 + NTSC_60.cmb
1 1280x1024 72Hz and 1 1920x1080 72Hz	1280x1024 + 1920x1080_72.cmb
1 1280x1024 60Hz and 4 640x480 60Hz	1280x1024 + 4@640x480_60.cmb
1 1280x907 60Hz and 5 640x453 60Hz	1280x907 + 5@640x453_60.cmb
2 1280x1024 60Hz	2@1280x1024_60.cmb
2 1280x1024 72Hz	2@1280x1024_72.cmb
3 1024x768 60Hz and 1 1280x959 30Hz interlaced	3@1024x768_60 + 1280x959_30i.cmb
3 1280x1024 60Hz	3@1280x1024_60.cmb
1 640x480 120Hz stereo	640x480_120s.cmb
7 800x600 60Hz and 1 NTSC	7@800x600_60 + NTSC.cmb
8 640x480 60Hz	8@640x480_60.cmb
8 800x600 60Hz	8@800x600_60.cmb

Table 9. Standard Format Combinations Shipped with All Onyx2 Reality and Onyx2 InfiniteReality2 Systems

7.13.2 2-Channel and 8-Channel Configurations

Both versions of the Onyx2 Reality and Onyx2 InfiniteReality2 video display subsystem are multi-channel capable. The standard version has two independent video channels for driving two separate RGB displays or virtual reality (VR) stereoscopic displays, or for a single RGB video channel and an independent video encoder. The high-performance version provides eight independent channels, each with its own 12-bit gamma table. On either version, channel 1 can be a normal RGB channel or the video encoder channel.

Because multi-channel capability is included in the standard system, users do not need to sacrifice a slot in the backplane to get this feature, even if the system is configured for eight outputs. Use of the Onyx2 Reality and Onyx2 InfiniteReality2 multi-channel feature does not disturb or disable operation of the “main” video output channels. All additional channels are as easy to program as the main channel; the same video format compiler works for all channels, including the CCIR601 Serial Digital Video Channel enabled by the Graphics to Video Out option (GVO).

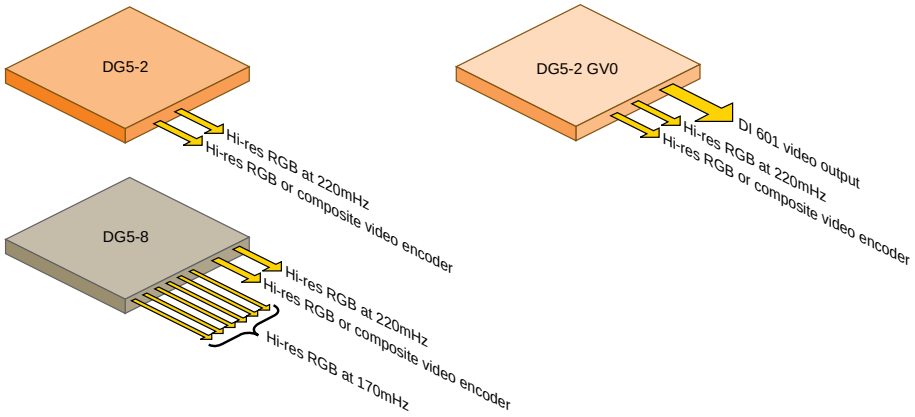


Figure 17. 2- and 8-Channel Options

7.13.3 Onyx2 Reality and Onyx2 InfiniteReality2 Multi-Channel Features: Considerations

Several system resources must be considered in multi-channel applications: frame buffer memory, frame buffer read/write bandwidth, DAC bandwidth, and frame buffer-to-video subsystem transmission bandwidth.

7.13.3.1 Swap Rates Must Be Equal

Video formats that run together in a video combination must provide time for “housekeeping” operations such as updating cursor position and glyph, setting parameters for dynamic pixel resampling, and more. This may be done at field boundaries, frame

boundaries, or in certain cases, across multiple frames. This interval is defined by the “maximum swap rate.” This rate must be the same for all video formats running together, so that the Onyx2 Reality and Onyx2 InfiniteReality2 video display subsystem can simultaneously perform these housekeeping services for all running video formats.

For example, a legal combination of formats would be the following:

- 30Hz interlaced NTSC (swap rate is the 60Hz field rate for this format)
- 60Hz non-interlaced 1280x1024
- 120Hz stereo (swap rate is the 60Hz frame rate)
- 180Hz color field sequential (again, the swap rate is the 60Hz frame rate)

Examples of illegal combinations would be NTSC (60Hz swap rate) and PAL (50Hz swap rate), or 60Hz and 72Hz versions of 1280x1024. In these cases, the swap rates are not the same.

“Equal” is a relative term. For example, NTSC, which really runs at 59.94Hz field rates, is close enough to be run with 60Hz 1280x1024. In general, if the swap rates are close enough to allow the video formats to be reliably synchronized to one another, they are considered “equal.” This tolerance is usually less than 0.2 percent.

Figure 18. Equal and Unequal Swap Rates

7.13.3.2 Frame Buffer to Video Display Subsystem Transmission Bandwidth

Digital video data is sent serially from the Onyx2 Reality and Onyx2 InfiniteReality2 frame buffer to the video subsystem. Each video channel uses a portion of the system's aggregate video transmission bandwidth. The required bandwidth per channel depends on the resolution of the video output format running on that channel and on the type of video it receives from the frame buffer. The requirements of all video channels must be less than the aggregate video bandwidth of the system.

Depending on the number and precision of color components requested from the frame buffer by each video channel, the available video transmission bandwidth ranges from approximately 290M pixels/second (for 10-bit RGB or 12-bit color field-sequential video) to 210M pixels/second (for 10-bit RGBA or 12-bit RGB video). The digital data stream from the frame buffer to a particular video channel must be of a single type and precision, for example, 10-bit RGB. However, different video channels (and associated digital video streams from the frame buffer) may be mixed: one channel may request 10-bit RGB, while another channel may request 12-bit RGB, and yet another channel may request 10-bit RGBA. Depending on the mix of currently running video formats, the total required video transmission bandwidth will fall between approximately 210M and 290M pixels/second.

Distinguishing the transmission format for video from the depth of pixels in the frame buffer is important. Pixel depth (small, medium, large, extra-large) must be uniform across the entire managed area of the frame buffer. On Onyx2 Reality, valid pixel depths are extra small (128 bits per pixel), small (256 bits per pixel), and medium (512 bits per pixel). As stated above, the format of the colors transmitted as video from a frame buffer to the video display subsystem may vary on a per-channel basis to conserve aggregate video transmission bandwidth. Frame buffer memory stores non-video information about each pixel (such as multisample and Z information); the color fields are a subset of the frame buffer pixel storage. Distinguishing between frame buffer representation of pixels and the format chosen for video transmission to the video display subsystem is important if you want to understand and optimize the performance of both the frame buffer memory and video display subsystem.

The total frame buffer to video subsystem bandwidth is independent of the number of Raster Manager boards in the system.

7.13.3.3 DAC Output Bandwidth

When configuring a video format combination, you must decide which format to assign to which video channel. The first two video channels have DACs with a 220M pixel/second bandwidth limit. The remaining six channels have DACs with a 190M pixel/second bandwidth limit. Because each channel can do dynamic resolution (see Section 7.13.7), actual video resolution as output by the DAC (and seen by the display monitor

connected to the channel) may be higher than the resolution of the rectangular region of the frame buffer assigned to that channel. The highest-bandwidth video formats of a video format combination should always be assigned to channels 0 and 1.

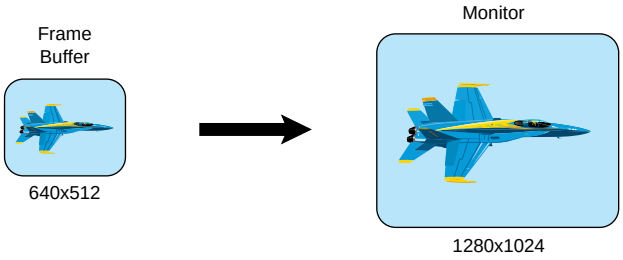


Figure 19. Dynamic Resolution

7.13.3.4 Frame Buffer Memory

Another requirement for video format combinations is that they must all be able to tile rectangular subregions of a rectangular frame buffer. Depending upon the depth of pixels selected and the number of Raster Manager boards installed, some combinations of formats may require more than one Raster Manager board. When selecting frame buffer pixel depth you must consider the quality of multisampling, the precision of the Z-buffer and RGB colors stored in the frame buffer, and total system fill rate capability.

Adding RM boards does not increase total frame buffer to video subsystem bandwidth, but it increases the total drawing fill rate available, and amortizes the video refresh overhead over more RM boards, leaving more fill capacity available for drawing.

7.13.3.5 Frame Buffer Read/Write Bandwidth

A final consideration for video format combinations is the overhead that refreshing video channels presents to the frame buffer. The greater the number and resolution of video channels, the more the pixel fill rate of the system is reduced. A particular combination of formats may fit into a 1-RM or 2-RM configuration, but it may unacceptably reduce the fill rate.

7.13.4 Video Format Combiner

To make the necessary trade-offs between these requirements, the system includes the Video Format Combiner software tool. This tool has a command-line interface and a graphical user interface (GUI). Using the GUI form, it is easy to see which regions of the frame buffer are mapped to which channels. Pixel bandwidth and fill rate overhead due to video are dynamically displayed. When system limits are exceeded, the tools attempt to provide hints about what remedial action to take.

If system resources of a target hardware configuration are sufficient enough to support a group of video formats, the combiner tool creates a *combination* file with filename references to the video formats and information about configuring the video hardware to run that channel combination.

Only video format combinations “approved” by the video format combiner are allowed to be loaded into the video subsystem (even if only a single channel is enabled). This is done with the video format combiner itself or with the familiar setmon program. In addition, there is a programming interface provided by an X-Windows™ server extension, XSGIvc, for application developers who want to integrate video control into their applications.

Figure 20 shows an example of using the video format combiner tool interactively to place six video channels on a 1-RM Onyx2 InfiniteReality2 system in a managed area of 1920 x 1360 pixels. Small (256-bit) pixels are used. Channel 0 is a 1280x1024 video format mapped to a 1280x907 region of the frame buffer using static resizing. Channels 1 through 4 are in a 640x480 video format. Using static resizing, each is mapped to a 640x453 region of the frame buffer. The Digital Video Option (DIVO) and the graphics-to-video option (GVO) are discussed in Section 7.16 and Section 7.17, respectively.

This example illustrates the use of video resizing to efficiently use the frame buffer. If static resizing were not used, only four 640x480 channels would fit in addition to the 1280x1024 channel. Note indicators at the bottom of the screen that show this video format combination is using 58.90 percent of the available pixel transmission bandwidth, and that the fill capability of this combination is reduced by 18.45 percent. For comparison, a single 1280x1024 screen would reduce the fill rate of a 1-RM system by about 10 percent. In other words, when running this combination, 81.55 percent (100 percent minus 18.45 percent) of the fill rate is available for drawing into the frame buffer.

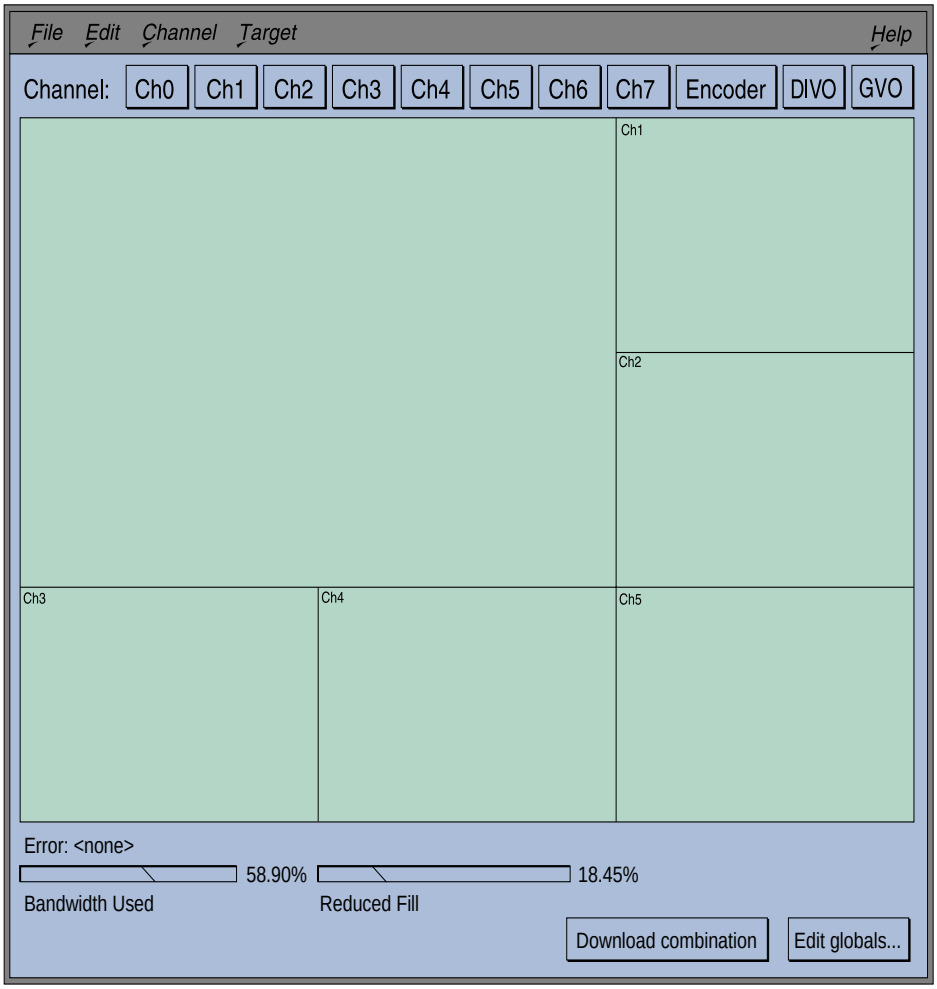


Figure 20. Using the Video Format Combiner to Place Six Video Channels

7.13.5 Examples of Video Format Combinations

As an example, a 1-RM system supports two 1280x1024, 60Hz noninterlaced channels using small (256-bit) pixels. Adding another RM board allows the use of two of these channels with medium (512-bit) pixels, or three channels using small pixels. A 2-RM configuration supports eight channels at 800x600 pixels at 60Hz. The video combiner tool allows these kinds of “what if” scenarios to be explored, enabling users to easily configure the Onyx2 Reality and Onyx2 InfiniteReality2 graphics systems for their particular applications. Table 10 shows several example configurations. Many others are possible.

Video Format Combinations	Notes
2 @ 1280x1024, 60Hz	Uses all memory on 1-RM (small frame buffer pixels)
3 @ 1280x1024, 72Hz	Uses nearly the entire available pixel bandwidth; requires 2-RMs for sufficient memory (small pixels)
8 @ 800x600, 60Hz	Uses all channels; needs 2-RMs for sufficient memory (small pixels)
7 @ 800x600, 60Hz + NTSC Encoder	Uses all channels; Channel 1 is used by Composite Video Encoder when enabled
3 @ 1024x768, 60Hz + RS343-A 1260x945, 30Hz interlaced	Not all formats need the same resolution; field rates of all video formats must match; requires 2-RMs for sufficient memory (small pixels)
1 @ 1920x1080 + 1 @ 1280x1024 both at 72Hz	1920x1080 channel must be assigned to high-speed DAC (channel 0 or channel 1); requires 2-RMs for sufficient memory (small pixels)

Table 10. Examples of Video Format Combinations

7.13.6 Video Format Compiler

A video format is the set of electrical and timing characteristics that drive a monitor (or any video output device). In the context of this report, a video format usually refers to the video format source language you use to describe the video format itself.

The video format compiler takes the video format source language file and produces a video format object file. You can use the video format object file to load video generation hardware (some hardware architectures require an intermediate step that combines multiple video format object files).

7.13.7 Dynamic Resolution

Onyx2 Reality and Onyx2 InfiniteReality2 incorporate on-the-fly video resampling on all video channels. This feature is known as dynamic resolution, which works on all video formats with pixel rates of 120MHz or less. This restriction eliminates only the highest-resolution formats.

This feature provides several benefits:

- Dynamic resolution, available through IRIS Performer as well as OpenGL, provides a powerful new method of guaranteeing scene update rates. Dynamic resolution works in addition to geometric level-of-detail methods.
- The video encoder channel can now encompass an entire video channel, allowing whole-screen recording without an external scan-converter. For video formats with pixel clock rates above 120MHz, the encoder works in “pass-through” mode similar to RealityEngine2, allowing recording of a pannable NTSC or PAL-sized region of the display.
- Dynamic resolution of a frame buffer region allows a smaller frame buffer region to be “zoomed” or enlarged to fit the video format resolution of the video channel assigned to that frame buffer region. This aids efficient use of frame buffer memory without requiring nonstandard video formats.

7.13.8 Stereoscopic Video Output

Onyx2 Reality and Onyx2 InfiniteReality2 continue the RealityEngine2 style of support for stereoscopic viewing in a window. Individual windows may be run in stereoscopic viewing mode, displaying a separate left-eye view and right-eye view at the appropriate frame rate. All other windows and pop-ups appear normal.

7.13.9 Direct Support of Color Field Sequential Video

Onyx2 Reality and Onyx2 InfiniteReality2 provide direct, efficient support of time-multiplexed, sequential color field video, commonly known as color field-sequential video. Cursors and color index pop-up regions appear in their true colors, making it easy to integrate applications using color field-sequential video with the X-Windows environment.

7.13.10 Luminance

Luminance (monochrome rendering) is supported by the video display subsystem. Applications may select luminance to be displayed as color mapped pseudo-color or as static-gray X-visual. Ultra-high color resolution (16 bits) is supported for rendering operations such as texture (but not lighting) using luminance.

7.13.11 Programmable Video Timing

The advanced circuitry of Onyx2 Reality and Onyx2 InfiniteReality2 generates a programmable pixel clock to enable the broadest range of applications, from video production to visual simulation. The clock controls the rate at which the frame buffer is scanned out to a video signal. The standard system supports 1920x1200 pixels at 66Hz noninterlaced.

Some applications need different video timing, such as military displays requiring the RS-343 standard, or stereoscopy for molecular modeling, which requires 120Hz video field rate to output a left-eye view and a right-eye view at 60Hz each. Onyx2 Reality and

Onyx2 InfiniteReality2 programmable video timing supports these and other video formats. By introducing the Video Format Compiler, Onyx2 Reality and Onyx2 InfiniteReality2 put video format programming capability in your hands.

Video formats produced by the Video Format Compiler must be put into a video format combination using the Video Format Combiner tool (Section 7.13.4) to validate that they will operate correctly on the Onyx2 Reality and Onyx2 InfiniteReality2 video display subsystem hardware. In the case of a single format, the word “combination” is a misnomer; the key concept is that all video formats, whether running alone or in combination with other video formats, must be validated by the video combiner to ensure that the video format is compatible with the display hardware.

7.14 Genlock

To ensure that Onyx2 Reality and Onyx2 InfiniteReality2 graphics subsystems video output is synchronized with other video sources or pipelines, the system includes a standard external genlock capability. Genlock assures that multiple video sources all start scanning out their frames at the same time for frame-synchronized multi-channel output.

Video channels in video subsystems of Onyx2 Reality and Onyx2 InfiniteReality2 are automatically frame-synchronized, whether or not they are of the same resolution, and regardless of whether the external genlock feature is enabled. External genlock capability lets the master video format (the format running on the lowest-numbered video channel) genlock to an external video source.

Unlike previous systems, the external video source only needs the same vertical sync rate as the master format; it does not need the identical format running on the master channel. This provides maximum flexibility in synchronizing to multiple Onyx2 Reality and Onyx2 InfiniteReality2 graphics pipelines or to video equipment of other vendors.

There are two methods of frame synchronization: genlock and frame-reset.

The highest-quality method of frame synchronization is genlock. It is used when two video formats are the same resolution and frame rate. The internal circuitry can make adjustments every horizontal period to ensure that the system is locked.

In the frame-reset method, the system merely issues a reset signal to the timing circuitry of the system to be synchronized. This method may allow frames to be “dropped” occasionally. Genlock guarantees that frames will not be dropped or missed.

The video combiner tool will select the best-quality frame synchronizing method, based on the video formats in the combination file, and the video format selected for external genlocking.

7.14.1 Swap Synchronization

Swap buffer timing can be synchronized between any number of Onyx2 Reality and Onyx2 InfiniteReality2 systems by means of external cabling to the swap_ready connector. The systems should be genlocked together when using swap synchronization.

7.15 Digital Video Multiplexer - DPLEX

7.15.1 Breakthrough Scalability for Visualization

The Digital Video Multiplexer Option (DPLEX) takes visualization scalability to a new level. DPLEX permits multiple InfiniteReality or InfiniteReality2 pipelines in an Onyx2 system to work simultaneously on a single suitable visual application. Moreover, DPLEX provides this capability in hardware, resulting in perfect (100 percent) scaling of both geometry rate and fill rate on some applications. An n-pipe system equipped with DPLEX may support up to n-times the frame rate of a single-pipe system running the same application, or n-times the scene complexity for a given frame rate, provided the rest of the system and application scale.

DPLEX is an optional daughtercard for Onyx2 systems that enables digital multiplexing of two or more pipelines. One DPLEX option is required for each pipeline in the system. The concept is simple: multiple pipelines operate simultaneously on successive frames of an application, which are then digitally multiplexed together before being converted to analog video. Any user seeking greater rendering speed may benefit from this technology. Key applications include distortion correction (required for dome simulators) and interactive large model visualization (required in many manufacturing settings, research institutions, etc.).

In addition to its multiplexing capabilities, DPLEX also provides high-resolution Low Voltage Differential Signal (LVDS) digital video output to drive next- generation digital displays, hardware-in-the-loop devices, and other custom peripherals.

7.15.2 Distortion Correction

With Onyx2, distortion correction is achieved using an approach in which a flat image is rendered to the frame buffer, copied back into texture memory, and then projected on a grid of polygons that represents an inverse mapping of the required distortion. A single InfiniteReality2 or InfiniteReality pipeline is capable of rendering 1280x1024 distortion-corrected imagery at 30Hz. With DPLEX, two such pipes can be multiplexed together to deliver a 60Hz update rate. And since the distortion correction occurs in the IG rather than in the projection system, the Onyx2 system-based solution can support dynamic distortion correction, in which the relative location of the projector or the shape of the screen changes over time.

7.15.3 Large Model Interactivity

DPLEX may allow an application to achieve higher performance levels. This can be very useful in digital prototyping, where large models are required to maintain visual fidelity. With DPLEX, even the largest, most complex visualizations become manageable because performance can be scaled to meet virtually any challenge.

7.15.4 Application and OS Support

Full-screen applications written to IRIS Performer or OpenGL Optimizer 1.1 are most easily adaptable to DPLEX operation. Other OpenGL applications will require additional modification to enable multipipe support in general and DPLEX support in particular. Not all applications can be made to work with DPLEX. DPLEX requires IRIX 6.5.2 or later.

7.15.5 DPLEX and MonsterMode Rendering

Underlying DPLEX is a data partitioning scheme known as time-composition: multiple pipelines operate simultaneously on different time-slices of a data set to generate successive frames, which are then composited together by the DPLEX network to form a continuous video stream. Because the composition takes place in dedicated hardware, DPLEX is said to perform hardware-based time-composition.

Onyx2 InfiniteReality2 also supports several software-based methods for distributing a data set over multiple pipelines. These methods are collectively known as MonsterMode rendering. Each uses the Onyx2 high-bandwidth memory subsystem rather than special hardware (DPLEX) for the interpipeline transfers required to partition the data and compose the final image. MonsterMode rendering methods include 2D composition for handling polygonal models and 3D composition for handling volumetric models. The latter provides the further benefit of additive texture capacity: an n-pipe system using MonsterMode 3D-composition software has n-times the effective texture memory of a single-pipe system (up to 1GB for a 16-pipe configuration). The choice of which method to use, DPLEX or MonsterMode, will depend on the nature of the application. Volume visualization should benefit more from MonsterMode, whereas polygonal data may favor DPLEX, MonsterMode, or a combination of the two. Please consult your Silicon Graphics representative for more information.

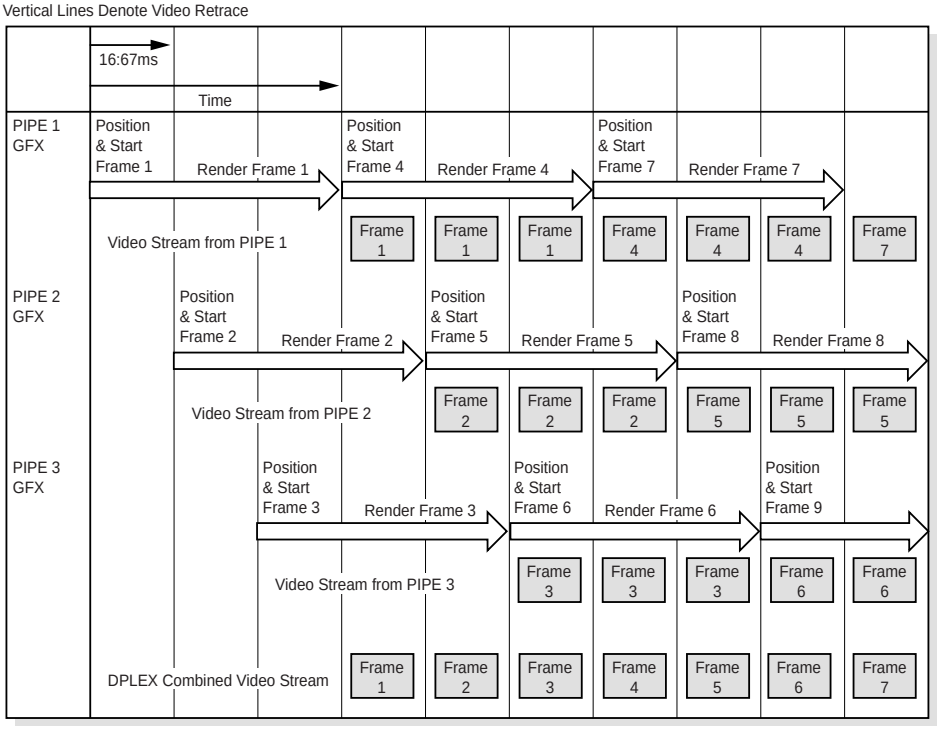


Figure 21. Example of 3-Pipe DPLEX Time Compositing

7.16 Digital Video Option (DIVO)

The Digital Video Option (DIVO) board is a half-size XIO board that is capable of streaming lossless noncompressed real-time video and audio data to and from system memory. Each DIVO board includes two fully independent channels, one for input and one for output.

DIVO supports noncompressed serial digital video formats ranging from single-link 4:2:2 8-bit to dual-link 4:4:4:4 10-bit, specifically SMPTE 259M (10-bit 4:2:2 Component Digital only), SMPTE 272M (video with embedded audio), and SDDI. DIVO also supports Rice encode/decode of these noncompressed formats, providing lossless reduction of approximately 2:1, reducing disk storage and network bandwidth requirements. DIVO supports up to 16 channels of embedded audio on both input and output.

7.17 Graphics-to-Video Option (GVO)

The Graphics-to-Video Option (GVO) is a daughtercard that connects directly to the Onyx2 Reality or Onyx2 InfiniteReality2 graphics pipeline and allows the user to output CCIR 601 serial digital video from the graphics frame buffer. GVO by itself is perfect

for output of graphics to digital video in real time. GVO and DIVO together are ideal for virtual sets where video is input to graphics and output in real time, and for content creation applications for simultaneous content input and output.

When the Graphics-to-Video Option card is installed on a Display Generator with two channels, both channels remain available. When the Graphics-to-Video Option card is installed on a Display Generator with eight channels, three channels remain available.

7.18 Integration with X Window System

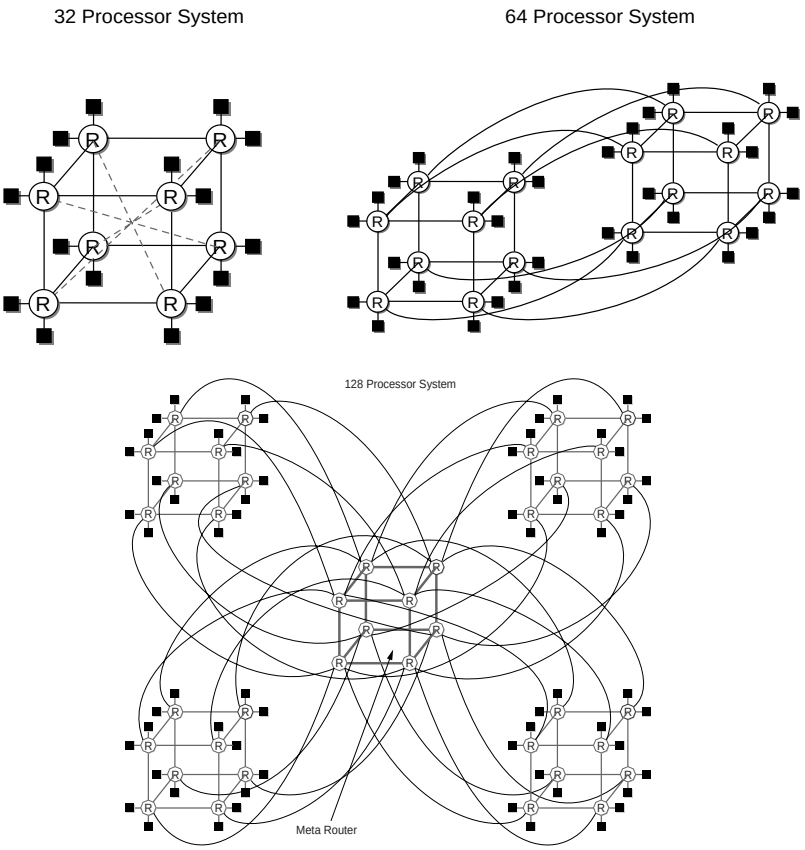
Onyx2 Reality and Onyx2 InfiniteReality2 graphics are integrated with the X Window System, giving multiple, independent graphics displays from a single frame buffer. The X11 standard is fully supported and native OpenGL is integrated without degradation of graphics pipeline performance when drawing into windows (versus a dedicated full screen).

Onyx2 Reality and Onyx2 InfiniteReality2 also unite a number of unique capabilities with respect to X Window System support. Up to 16 windows can have their own 12-bit color lookup tables (LUTs) to provide a stable, independent data space that does not require reloading with each current window switch.

8.0 Onyx2 Hardware Architecture

8.1 Architecture Overview

Onyx2 is a scalable system. Onyx2 is also a modular system because it can be increased by adding standard nodes to the interconnection fabric. This modular architecture gives the system a few important characteristics. When purchasing an entry-level system, customers need not pay for complete expandability. Rather, they pay for infrastructure to expand the system as they grow the system. Also, the system is inherently reliable, as the hardware in one module can survive failures in other modules.



Interconnection fabric: multidimensional. Many connections can be made at a time.

R=Router Black squares = Node with one or two CPUs

Figure 22. Onyx2 Scalability

8.2 Node Card

The node card (deskside or rack) is the basic building block of the Onyx2 systems. The node card contains two R10000 processors, a second level cache, main memory, directory memory for cache coherence, an ASIC called the hub for interconnection, an I/O interface, and an interface to the interconnect fabric. For a block diagram of the node card see Figure 23

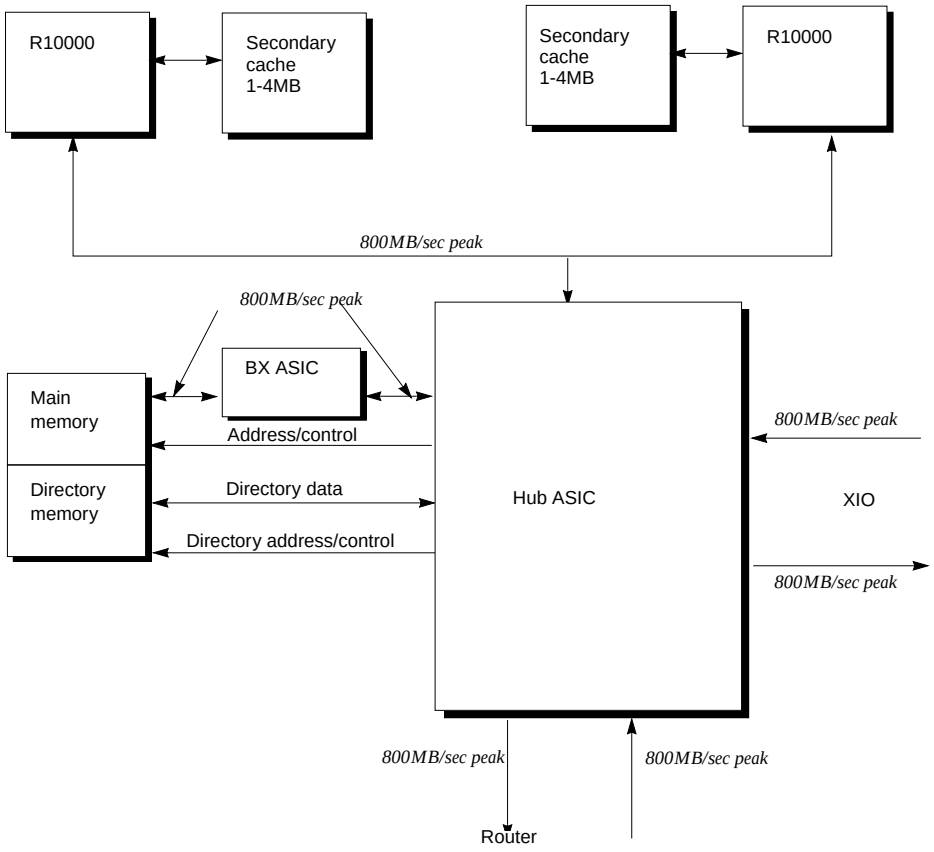


Figure 23. Node Board Block Diagram

8.2.1 Processor

Each deskside Onyx2 node board (also referred to in this text as a “node”) is capable of supporting two R10000 processors. In a deskside or rackmounted configuration, each processor is mounted on an HIMM (horizontal in-line memory module) together with its primary cache, and either 1 or 4MB or secondary cache.

8.2.2 Memory

Onyx2 systems use distributed shared-memory (DSM). With DSM, main memory is partitioned among processors but accessible to and shared by all of the processors. Onyx2 divides main memory into two classes: near and remote. Memory on the same node as the processor is labelled near, with all other memory in the system labelled remote.

To a processor, main memory appears as a single addressable space containing many blocks, or 4KB pages. Each node is allotted a static portion of the physical address space — which means there is a gap if a node is not present. The gap is not visible to user processes since they use virtual addresses.

8.2.3 Hub

The node board has a hub ASIC, which connects the processors, main memory and its associated directory memory, the system interconnection fabric, through a dedicated router port, and the I/O subsystem. The hub is a four-port crossbar switch, with four bidirectional ports, each running at 800MB/second (peak bandwidth) each way, full-duplex, for a total bandwidth of 1.6GB/second at each port (processor, memory, system, I/O). It is responsible for connecting all four interfaces of the node together:

- processor
- memory
- XIO
- CrayLink™ interconnection fabric through a router board

These four interfaces are interconnected by an internal crossbar, as shown in Figure 24. The interfaces on the hub communicate by sending messages through the crossbar.

The hub controls *intranode* communications between the node's subsystems, and also controls *internode* communications with other hub ASICs in other nodes. The hub converts internal messages, using a request/reply format, to and from the external message format used by the XIO or CrayLink port. All internal messages are initiated by processors and I/O devices.

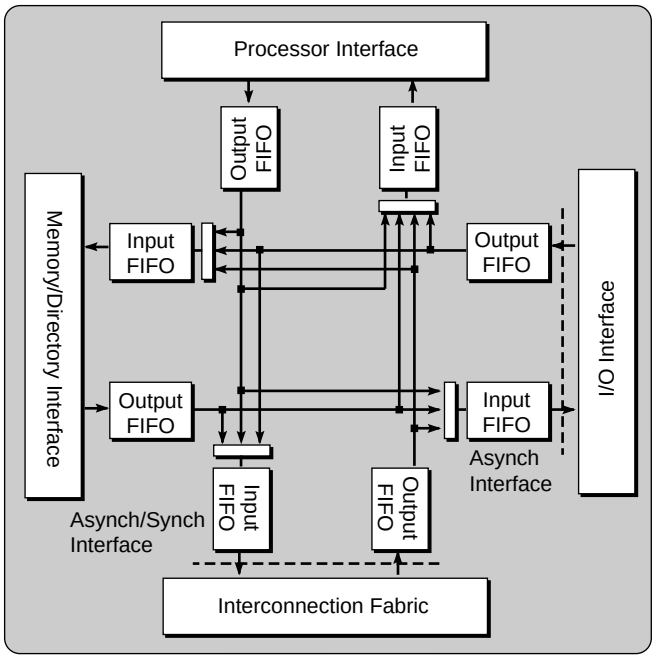


Figure 24. Block Diagram of a Hub ASIC

As shown in Figure 24, each hub interface has two first-in/first-out (**FIFO**) buffers: one for incoming messages and one for outgoing messages. The FIFOs provide buffering between the hub chip and the devices to which it connects. When empty, the FIFOs provide bypassing for lower latency.

8.2.4 Global Real-Time Clock

One hub can be designated a clock "master," sending a global clock out through the interconnect fabric to the rest of the hubs, which then slave off this clock. A BNC connector on the edge of the node board can feed an external 1MHz TTL real-time clock to the system if greater accuracy is needed.

8.3 I/O Subsystem

8.3.1 XIO Protocol

Onyx2 systems use an advanced input-output (I/O) subsystem, consisting of a number of high-speed **XIO** links. XIO supports a wide range of Silicon Graphics and third-party I/O devices.

XIO is distributed, with an I/O port on each node board. As with Onyx2 distributed memory, each I/O port is accessible by every CPU. I/O is controlled either through the single-port XIO-protocol link on the node board, or through an intelligent crossbar interconnect on the **XBOW** ASIC.

8.3.2 XBOW

The **XBOW** ASIC has a dynamic crossbar switch which expands the dual-host XIO port to eight 16-bit ports. Six ports are used for I/O and two connect to node boards. Each I/O port can run in either 8- or 16-bit mode, with rate-matching buffers to decouple 16-bit to 8-bit ports. At least one and a maximum of two XBOW ports must connect to a host. The electrical interface for XIO is the same as that used by CrayLink.

A number of interface ASICs are available to link the XBOW ports to PCI, VME, SCSI, Ethernet, ATM, Fibre Channel, and other I/O devices. The interface ASICs include the IOC3, LINC, and Bridge ASICs.

The XBOW ASIC uses the XIO protocol at all of its ports. There are two XBOW ASICs on each deskside midplane. On the Onyx2 deskside graphics system, a single XBOW ASIC is provided. The electrical interface for XIO is the same as that used by CrayLink.

A functional view of a XBOW ASIC, with dual hosts and six half-size XIO boards, is shown in Figure 25. (CrayLink cabling is not shown in this illustration.)

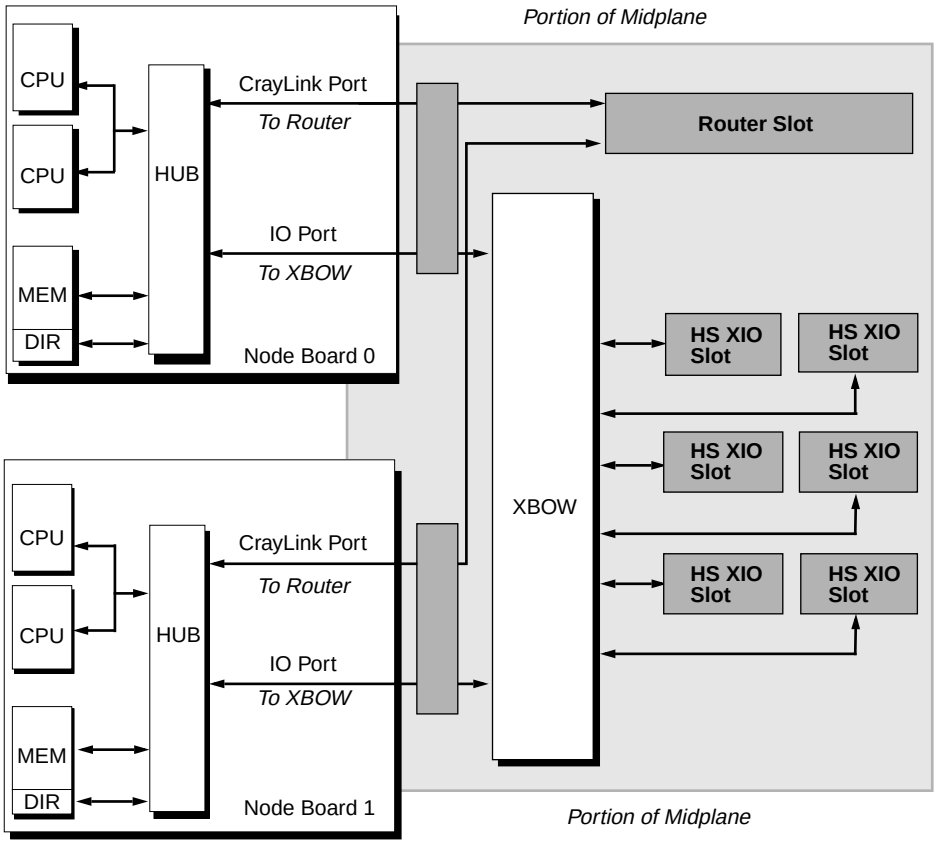


Figure 25. Functional Location of XBOW ASIC

In this dual-host configuration, the six remaining XIO ports are statically partitioned between the two node boards; if one of the host nodes is inoperable, the second host node can be programmed to take control of all the XIO ports. Note that the node boards connect to ports 1 and 3 of the XBOW, and the remaining six ports connect to the XIO widgets.

The XBOW ASIC contains a crossbar switch that dynamically connects individual ports to particular I/O devices (host, graphics board, serial I/O). The XBOW decodes fields in XIO messages to determine control and destination information.

8.3.3 XIO Devices

The form-factor for XIO boards may vary. Typically a device is a single board, either half-size (10-inch x 6.5-inch x 1-inch) or full-size (10-inch x 13-inch x 1-inch); however it is possible for a device to include a daughterboard.

One specific case of XIO devices are the graphics pipelines (Onyx2 Reality and Onyx2 InfiniteReality2). Onyx2 desksides use the XIO protocol, while Onyx2 racks use the Crosstown protocol.

XIO can also run outside an enclosure, using the Crosstown protocol, which is described in the Scalable Shared-memory Multiprocessing Server Technical Report.

On all systems a basic IO device is included for the standard support for two UltraSCSI buses (40Mb/second each), serial ports, and fast Ethernet (100Base-T, 100Mbit/seconds). On all graphics systems an MIO audio serial option is also included in the standard configuration. The MIO device is optional in nongraphics configurations.

8.3.4 Media Input/Output (MIO)

Standard with all graphics configurations, the Media Input/Output (MIO) device includes analog audio (stereo in/out lines), professional digital audio (AES/EBU in/out and ADAT 8+8 channel optical in/out), loop through video blackburst sync input, system-wide UST in even the largest system configurations, two serial ports RS-232 and RS-422 (up to 115200 baud), two sets of keyboard/mouse ports, and one parallel port.

8.3.5 PCI-64

Onyx2 also supports PCI-64 boards. In the deskside and rack systems, PCI is supported through an optional PCI expansion board.

PCI support is optionally added to the deskside and rack systems using an internal PCI expansion box. This box plugs into one of the XIO slots in the module. This box requires access to a wide XIO slot. Only a single PCI expansion box is supported per deskside or rack module. Each PCI expansion box provides three 64-bit PCI full size card slots.

8.4 Interconnect Subsystem

8.4.1 Topology

Onyx2 nodes are connected by an interconnection fabric. The **interconnection fabric** is a set of switches, called *routers*, that are linked by cables in various configurations, or *topologies*. The interconnection fabric differs from a standard bus in the following important ways:

- The interconnection fabric is a mesh of multiple point-to-point links connected by the routing switches. These links and switches allow multiple transactions to occur simultaneously.
- The links permit extremely fast switching. Each bidirectional link sustains as much bandwidth as the entire bus in a CHALLENGE® class system.
- The interconnection fabric does not require arbitration nor is it as limited by contention, while a bus must be contested through arbitration.
- More routers and links are added as nodes are added, increasing the interconnection fabric's bandwidth. A shared bus has a fixed bandwidth that is not scalable.
- The topology of the CrayLink is such that the bisection bandwidth grows linearly with the number of nodes in the system.

The interconnection fabric provides a minimum of two separate paths to every pair of Onyx2 nodes. This redundancy allows the system to bypass failing routers or broken fabric links. Each fabric link is additionally protected by a CRC code and a link-level protocol, which retry any corrupted transmissions and provide fault tolerance for transient errors.

Figure 26 illustrates an 8-node hypercube with its multiple datapaths. Simultaneously, R1 can communicate with R0, R2 to R3, R4 to R6, and R5 to R7, all without having to interface with any other node.

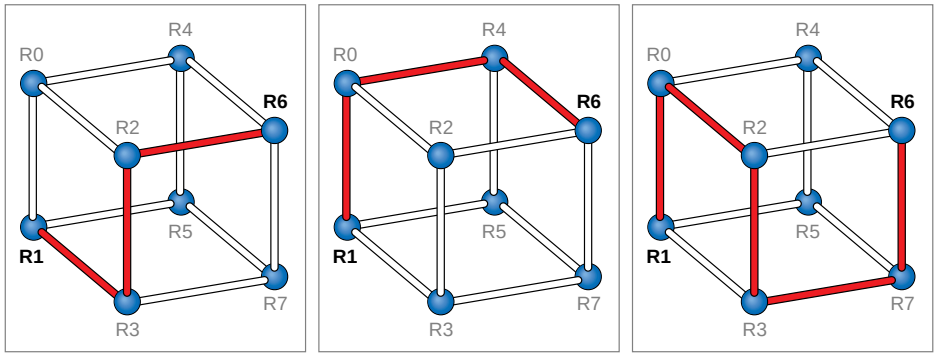


Figure 26. Datapaths in an Interconnection Fabric

8.4.2 Routers

Router boards physically link the hub ASIC on the node board to CrayLink. CrayLink provides a high-bandwidth, low-latency connection between all the node boards. Central on a router board is the router ASIC, which implements a full six-way non-blocking crossbar switch.

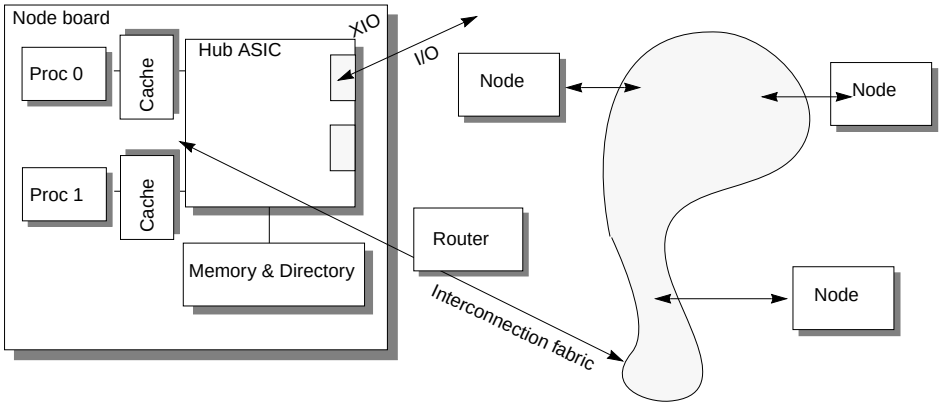


Figure 27. Location of a Router Board in an Onyx2 System

The router crossbar allows all six of the router ports to operate simultaneously at full-duplex; each port consists of two unidirectional data paths. The router board also includes a set of protocols, which provides a reliable exchange of data even in the face of transient errors on links, manages flow-control, and prioritizes data so that older data is given precedence over newer data.

8.5 Distributed Shared Address Space

Onyx2 memory is located in a single shared address space. Memory within this space is distributed among all the processors, and is accessible over the interconnection fabric. This differs from a Challenge-class system, in which memory is centrally located on and only accessible over a single shared bus. By distributing memory of Onyx2 among processors memory latency is reduced: accessing memory near to a processor takes less time than accessing remote memory. Although physically distributed, main memory is available to all processors.

I/O devices are also distributed within a shared address space; every I/O device is universally accessible throughout the system.

8.6 Onyx2 Memory Hierarchy

Onyx2 memory is organized into the following hierarchy:

At the top, and closest to the processor making the memory request, are the **processor registers**. Because they are physically on the chip they have the lowest latency—that is, they have the fastest access times. In Figure 28, these are on the processor labelled P0.

The next level of memory hierarchy is labelled **cache**. In Figure 28., these are the primary and secondary caches located on P0. Aside from the registers, caches have the lowest latency in Onyx2, since they are also on the R10000 chip (primary cache) or tightly coupled to its processor on a daughterboard (secondary cache).

The next level of memory hierarchy is called **local memory**. This is the memory and the directory on the same node as the processor. In Figure 28, local memory is the block of main memory on Node 0.

The next level of memory hierarchy consists of the **remote caches** that may be holding copies of a given memory block. If the requesting processor is writing, these copies must be invalidated. If the processor is reading, this level exists if another processor has the most up-to-date copy of the requested location. In Figure 28, remote cache is represented by the blocks labelled “cache” on Nodes 1 and 2.

Caches are used to reduce the amount of time it takes to access memory—also known as a memory’s **latency**—by moving faster memory physically close to, or even onto, the processor. This faster memory is generally some version of static RAM, or SRAM.

The DSM structure of Onyx2 also creates the notion of local memory. This memory is close to the processor and has reduced latency compared to bus-based systems, where all memory must be accessed through a shared bus.

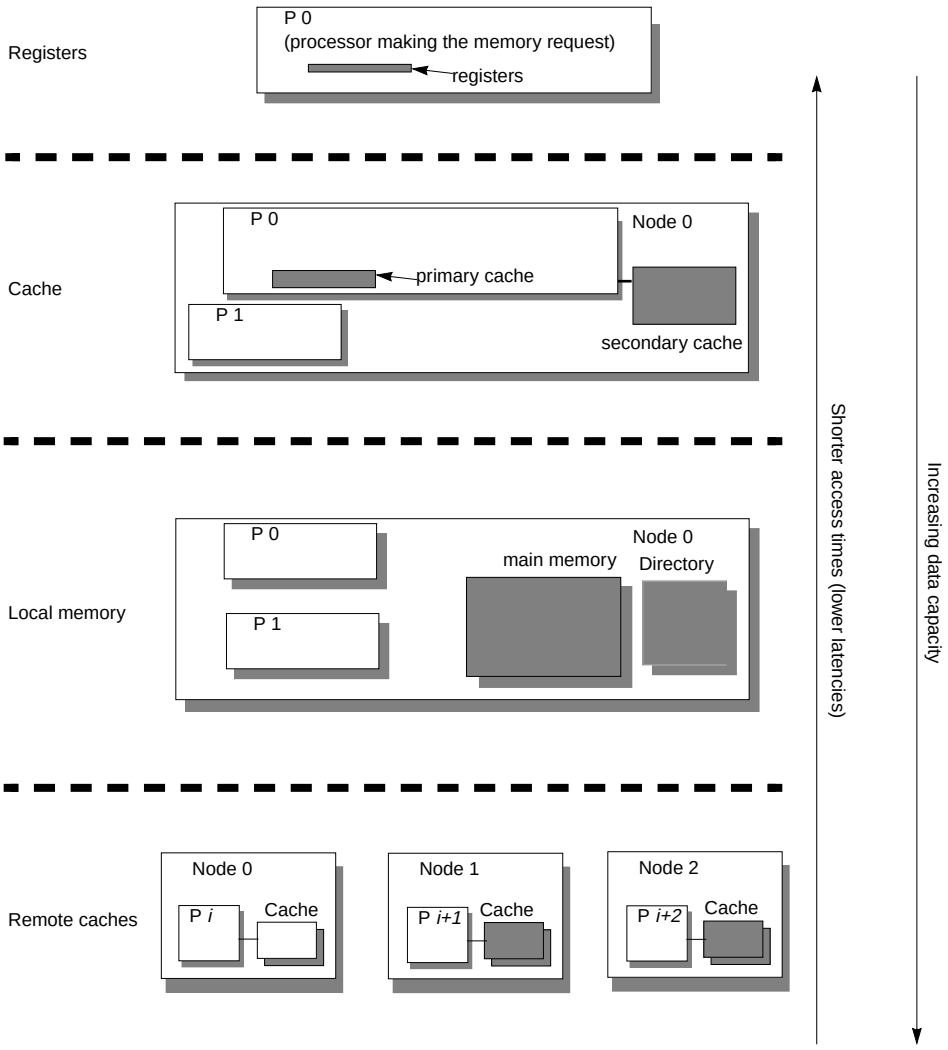


Figure 28. Memory Hierarchy, Based on Relative Latencies and Data Capacities

While data only exists in either local or remote memory, copies of the data can exist in various processor caches. Keeping these copies consistent is the responsibility of the logic of the various hubs. This logic is collectively referred to as a *cache-coherence protocol*, described in Section 8.7.

8.7 Cache Coherency

Simply put, coherence is the ability to keep data consistent throughout a system. Data coherence in a uniprocessor system is usually managed directly by the processor, so no separate coherence protocol is needed.

A multiprocessor configuration is different. In a system such as Onyx2, data can be copied and shared among all the processors and their caches. Moving data into a cache reduces memory latency, but it can also complicate coherence because the cached copy may become inconsistent with the same data stored elsewhere. A **cache coherence protocol** is designed to keep data consistent and disperse the most-recent version of data to wherever it is being used.

Onyx2 uses a directory-based coherence protocol, described further in the S2MP Server Technical Report.

8.8 System Latencies

Memory latency is the amount of time it takes a system to complete a read or write to memory. It is quoted using two metrics: access time and cycle time.

- **Access time** is the time between the issuance of a data request and the time when the data is returned.
- **Cycle time** is the time between repeated requests.

Memory latency is reduced by:

- The inherent spatial and temporal locality in caches
- Distributing memory, and moving some of it close to each processor
- Page migration, in which frequently accessed data is moved from remote memory to local memory
- The integrated node design and CrayLink topology, which reduces the number of chip crossbars and the contention to reach remote memory

8.9 System Bandwidths

Three types of bandwidth are cited in this manual:

- Peak bandwidth, which is a theoretical number derived by multiplying the clock rate at the interface by the data width of the interface.
- Sustained bandwidth, which is derived by subtracting the packet header and any other immediate overhead from the peak bandwidth. This best-case figure, sometimes called Peak Payload bandwidth, does not take into account contention and other variable effects.
- Bisection bandwidth, which is derived by dividing the interconnection fabric in half, and measuring the data rate across this divide. This figure is useful for measuring data rates when the data is not optimally placed.

A comparison between peak and sustained data bandwidths at the XIO and interconnection fabric interfaces of the hub is given in Table 11. System bisection bandwidth is given in Table 12.

Interface at Hub	Half/Full Duplex	Peak Bandwidth per Second	Sustained Bandwidth per Second	Maximum Simultaneous System Bandwidth per Second
Processor	Unidirectional	800MB		800MB
Memory	Unidirectional	800MB		
Interconnection Fabric	Full duplex	1.6GB	1.28GB	1.6GB
	Half duplex	800MB	711MB	
XIO	8-bit full duplex	800MB	640MB	
	8-bit half duplex	400MB	355MB	
	16-bit full duplex	1.6GB	1.28GB	
	16-bit half duplex	800MB	711MB	
Maximum Bandwidth per Hub (2 CPUs)				2.4GB

Table 11. Bandwidths on Processor, Memory, and I/O Paths

System Size (Number of CPUs)	Bisection Bandwidth (without Xpress Links)	Bisection Bandwidth (with Xpress Links)
8	1.6GB/second	3.2GB/second (star router)
16	3.2GB/second	6.4GB/second
32	6.4GB/second	9.6GB/second
64	12.8GB/second	n/a
128	25.6GB/second	n/a

Table 12. Bisection Bandwidths of Various Onyx2 Configurations

The total system bandwidth is shown in Table 13. This is the bandwidth while all CPUs are reading local memory and all routers are reading and writing across the XBOWs at the same time.

System Size (Number of CPUs)	Maximum Simultaneous System Bandwidth
8	9.6GB/second
16	19.2GB/second
32	38.4GB/second
64	76.8GB/second
128	153.6GB/second

Table 13. Maximum Simultaneous System Bandwidth

8.10 R10000 Processor Architecture

The R10000 microprocessor from MIPS Technologies is a four-way superscalar architecture that fetches and decodes four instructions per cycle. Each decoded instruction is appended to one of three instruction queues, and each queue can perform dynamic scheduling of instructions. The queues determine the execution order based on the availability of the required execution units. Instructions are initially fetched and decoded in order, but can be executed and completed out of order, allowing the processor to have up to 32 instructions in various stages of execution. The impressive integer and floating-point performance of R10000 make it ideal for applications such as scientific computing, engineering workstations, 3D graphics workstations, database servers, and multiuser systems. The high throughput is achieved through the use of wide, dedicated data paths and large on-and-off chip caches.

The R10000 microprocessor implements the MIPS IV instruction set architecture. MIPS IV is a superset of the MIPS III instruction set architecture and is backward compatible. The R10000 microprocessor delivers peak performance of 800 MIPS (400 MFLOPS) with a peak data transfer rate of 3.2GB per second to secondary cache. The R10000 microprocessor is available in a 599 CLGA package and is fabricated using a CMOS sub-.35-micron silicon technology.

Key features of the R10000 include:

- ANDES Advanced Superscalar Architecture
  - Supports four instructions per cycle
  - Two integer and two floating-point execute instructions plus one load/store per cycle
- High-Performance Design
  - 3.3 volt technology
  - Out-of-order instruction execution
  - 128-bit dedicated secondary cache data bus
  - On-chip integer, FP, and address queues
  - Five separate execution units
  - MIPS IV instruction set architecture

- High Integration Chip-Set
  - 32KB two-way set associative, two-way interleaved data cache with LRU replacement algorithm
  - 32KB two-way set associative instruction cache
  - 64 entry translation look-aside buffer
  - Dedicated second level cache support
- Second Level Cache Support
  - Dedicated 128-bit data bus
  - Generation of all necessary SSRAM signals
  - 3.2GB/second peak data transfer rate
  - Programmable clock rate to SSRAM
- Compatibility with Industry Standards
  - ANSI/IEEE Standard 754–1985 for binary floating-point arithmetic
  - MIPS III instruction set compatible
  - Conforms to MESI cache consistency protocol
  - IEEE Standard 1149.1/D6 boundary scan architecture
- Avalanche Bus System Interface
  - Direct connect to SSRAM
  - Split transaction support
  - Programmable interface

8.10.1 R10000 Product Overview

The R10000 microprocessor implements many techniques to address common computing challenges. This section discusses some of these features, such as caches, branch prediction, queueing structures, register renaming, execution units, and load/store units. We begin with a block diagram of the R10000 microprocessor.

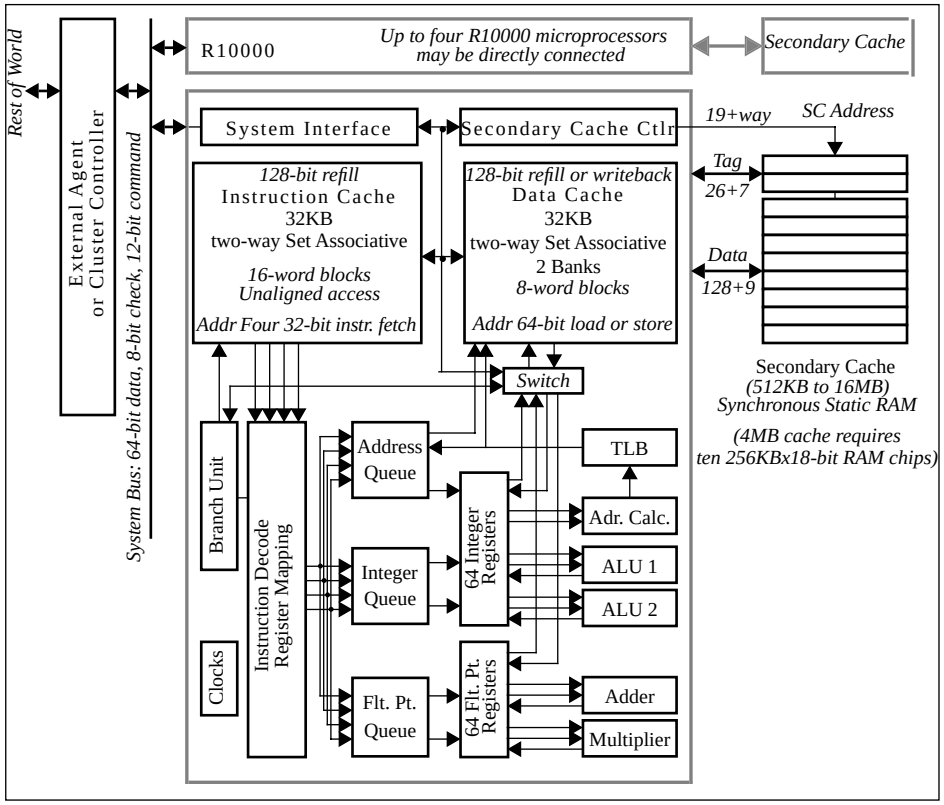


Figure 29. R10000 Processor Block Diagram

8.10.2 Primary Data Cache

The 32KB primary data cache of the R10000 microprocessor is arranged as two identical 16KB banks. The cache is *two-way interleaved*. Each of the two banks is *two-way set associative*. Cache line size is 32 bytes.

The data cache is *virtually indexed* and *physically tagged*. The virtual indexing allows the cache to be indexed in the same clock in which the virtual address is generated. However, the cache is physically tagged in order to maintain coherency with the secondary cache.

8.10.3 Secondary Data Cache

The secondary cache interface of the R10000 microprocessor provides a 128-bit data bus. All of the standard Synchronous Static RAM interface signals are generated by the processor. No external interface circuitry is required. The minimum cache size is 512KB. Maximum cache size is 16MB. Secondary cache line size is programmable at either 64 bytes or 128 bytes.

8.10.4 Instruction Cache

The instruction cache is 32KB and is two-way set associative. Instructions are partially decoded before being placed in the instruction cache. Four extra bits are appended to each instruction to identify the execution unit to which the instruction will be dispatched. The instruction cache line size is 64 bytes.

8.10.5 Branch Prediction

The branch unit of the R10000 microprocessor can decode and execute one branch instruction per cycle. Since each branch is followed by a delay slot, a maximum of two branch instructions can be fetched simultaneously, but only the earlier one will be decoded in a given cycle.

A *branch bit* is appended to each instruction during instruction decode. These bits are used to locate branch instructions in the instruction fetch pipeline.

The path a branch will take is predicted using a *branch history RAM*. This two-bit RAM keeps track of how often each particular branch was taken in the past. The two-bit code is updated whenever a final branch decision is made.

Any instruction fetched after a branch instruction is speculative; it is not known at the time these instructions are fetched whether or not they will be completed. The R10000 microprocessor allows up to four outstanding branch predictions that can be resolved in any order.

Special on-chip *branch stack* circuitry contains an entry for each branch instruction being speculatively executed. Each entry contains the information needed to restore the processor's state if the speculative branch is predicted incorrectly. The branch stack allows the processor to restore the pipeline quickly and efficiently when a branch misprediction occurs.

8.10.6 Queueing Structures

The R10000 microprocessor contains three instruction queues. These queues dynamically issue instructions to the various execution units. Each queue uses instruction tags to track instructions in each execution pipeline stage. Each queue performs dynamic scheduling and can determine when the operands that each instruction needs are available. In addition, the queues determine the execution order based on the availability of the corresponding execution units. When the resources become available, the queue releases the instruction to the appropriate execution unit.

8.10.7 Register Renaming

Dependencies between instructions can degrade the overall performance of the processor. Register renaming is a technique used to determine these dependencies between instructions and provide for precise exception handling. When a register is *renamed* the logical registers which are referenced in an instruction are mapped to physical registers

using a mapping table. A logical register is mapped to a new physical register whenever it is the destination of an instruction. Hence when an instruction puts a new value in a logical register, that logical register is renamed to use the new physical register. However, the previous value remains in the old physical register. Saving the old register value allows for precise exception handling.

While each instruction is renamed, its logical register numbers are compared to determine the dependencies between the four instructions being decoded during the same cycle.

8.10.8 Execution Units

The R10000 microprocessor contains six execution units that operate independently of one another. There are two integer arithmetic logic units (ALU), two primary floating-point units, and two secondary floating-point units, which handle long-latency instructions such as *divide* and *square root*.

8.10.9 Load/Store Units and the TLB

Load/Store units consist of the address queue, address calculation unit, translation lookaside buffer (TLB), address stack, store buffer, and primary data cache. Load/Store units perform load, store, prefetch, and cache instructions.

All load or store instructions begin with a three-cycle sequence that issues the instruction, calculates its virtual address, and translates the virtual address to the physical address. The address is translated only once during the operation. The data cache is accessed and the required data transfer is completed, provided there was a primary data cache hit.

If there is a cache miss, or if the necessary shared register ports are busy, the data cache and data cache tag access must be repeated after the data is obtained from either the secondary cache or main memory.

The TLB contains 64 entries and translates virtual addresses to physical addresses. The virtual address can originate from either the address calculation unit or the program counter (PC).

9.0 IRIX

9.1 Introducing IRIX for Onyx2

Since the advent of the microprocessor, computer system users and designers have dreamed about being able to configure large powerful systems by connecting many smaller, less powerful systems together. Ideally, a modular system would be immensely scalable and resilient, yet have the manageability, efficiency, and throughput associated with mainframes and supercomputers. Many vendors have tried to deliver on this vision, but the systems that emerged have behaved as anything but a single system whose resources work together.

With the first shipments of Onyx2 hardware and IRIX 6.4 for Onyx2, Silicon Graphics began to deliver the dream. IRIX 6.4 for Onyx2 was the initial release of an operating system with major scalability, performance, throughput, and availability enhancements to the industry's richest UNIX operating environment. IRIX 6.4 for Onyx2 also delivered the first products of a united Cray Research and Silicon Graphics: industry-leading supercomputing/datacenter software functionality on scalable cost-effective and high-performance Silicon Graphics hardware.

9.1.1 IRIX Means Scalability

IRIX for Onyx2 is the distributed software architecture for scalability and availability from Silicon Graphics. Silicon Graphics has been a leader in scalable UNIX from the day the company shipped its first symmetric multiprocessing systems in 1988, demonstrating continuing leadership with exceptional (NFSv2) LADDIS, TPC-C, and HIPPI throughput results on 64-bit CHALLENGE and POWER CHALLENGE™ servers running IRIX 5.3, 6.1, and 6.2.

IRIX for Onyx2 supports modular, pay-as-you-go "computing by the yard," providing availability and throughput on small, one-to-four processor systems and scalability, performance, resilience, and throughput on large systems with tens or hundreds of processors and up to tens of gigabytes of memory. IRIX for Onyx2 takes traditional symmetric multiprocessing systems to the next level of scalability, and marks a new era in computing: modular, scalable systems that behave like a highly resilient mainframe or supercomputer because the operating system is both distributed and unified.

9.1.2 IRIX 6.5 for Onyx2

IRIX 6.5 for Onyx2 fully delivers on the promises of the software architecture. IRIX 6.5 for Onyx2 is scalable: it is expandable immediately from one to 128 processors. IRIX 6.5 for Onyx2 is compatible: it builds on the features, functions, and interfaces of 64-bit IRIX 6.2. IRIX 6.5 for Onyx2 provides high-performance through innovative dynamic, explicit, and implicit data placement functionality. IRIX 6.5 for Onyx2 provides high throughput via an advanced scheduler that ensures fairness for interactive users and via supercomputing software from Silicon Graphics and Cray Research.

IRIX 6.5 builds on the exceptional heritage of IRIX. IRIX 6.5 for Onyx2 builds on one of the industry's first fully functional 64-bit UNIX implementations, provides exceptional real-time capabilities, presents virtually unmatched NFSv2 and NFSv3 network performance and data capacities, and offers an exceptional suite of supercomputing software. Building on earlier IRIX releases, IRIX 6.5 for Onyx2 is a leader in open systems with X/Open XPG4 Base 95 Profile Branding and broad POSIX standards compliance. Government and other security-conscious customers can take advantage of standard C2 or optional B1 security, or get a level of security in-between with the Silicon Graphics Commercial Security Pack (which adds kerberos, ACLs, and extended capabilities to C2 security without requiring a full upgrade to a B1 release).

IRIX 6.5 for Onyx2 is supported on all current Silicon Graphics systems including Onyx2, Origin200™, Origin2000, Octane™, O2™ and earlier systems such as Onyx®, CHALLENGE, Indigo2™, and Indy®.

9.2 Memory Management

Memory management in IRIX for Onyx2 is a sophisticated demand paging virtual memory system designed to meet the challenges of the broadest possible spectrum of applications and the broadest range of machines from uniprocessor desktop graphics systems to the largest multiprocessor supercomputers. The IRIX VM system has been extensively tuned and optimized to provide the functionality and performance required to support everything from the world's busiest Web servers to real-time graphics to desktop applications to video-on-demand systems to the largest database servers to the most demanding supercomputing applications.

The IRIX VM system is the result of 10 years of evolution and technology investment by Silicon Graphics to produce a memory management system that has the internal infrastructure and the external interfaces required to meet the demands of evolving multiprocessor computer systems and leading-edge applications. The IRIX VM system has been designed to scale to very large physical memories, very large address spaces, very large number of processors, and very large number of processes, without sacrificing performance in small desktop systems. The release of IRIX on Onyx2 marks the third full generation of industry-leading MP hardware supported by the IRIX VM system, starting with the release of the original POWER Series™ MP systems in 1988.

The IRIX for Onyx2 VM system has the following features:

- Full support for the SVR4 API, including memory mapped files and devices (mmap) and per page protection attributes (mprotect)
- Full support for the SVR4 MIPS ABI
- 64-bit user virtual addresses, allowing individual processes with address spaces spanning terabytes
- Dynamic loading of libraries. At run-time, a process can dynamically bind and unbind libraries using DSOs (Dynamic Shared Objects)
- Efficient process creation (fork) by using lazy replication of data pages (copy-on-write)

- Efficient support for parallel programming by sharing address space between multiple processes or threads
- Multiple swap devices, including swapping to regular files in local or remote (NFS™) filesystems. Swap devices and swap files can be dynamically added and removed without interrupting system operation or rebooting
- Multiple page sizes within a single process address space. On machines with TLB support for multiple page sizes (R4000® and R10000), the VM system allows a program to request specific ranges of its address space to be represented using pages that are multiples of the base page size of the system. This allows programs with large data sets and poor spatial reference locality to minimize address translation (TLB) overhead
- Topologically aware memory placement, including both user APIs and automatic tools to allow programs to be mapped onto Onyx2 hardware as efficiently as possible
- Automatic replication of shared text pages on Onyx2
- Automatic migration of shared data pages on Onyx2. Migration of pages can also be performed explicitly under user program control

9.3 Scheduling

An IRIX process is an executable instance of a program. The context of the process when it is running includes the process virtual address space, process attributes (working directory, signal handling status, process, user, and group ID numbers), and the contents of machine registers.

To support the sizable number of processes typically running on a Silicon Graphics workstation or server, IRIX provides a time-sharing priority band with a process-scheduling algorithm that assures an equitable division of processor time among all processes. This time-sharing algorithm is nonpreemptive, that is, the running process cannot be preempted by another process (but can be preempted by the kernel).

In normal operation, the kernel pauses briefly every few milliseconds to make scheduling decisions for the processors under its control. Processes are each given a guaranteed time slice, a period when each is allowed to execute without being preempted. Often, a process yields to another process “voluntarily,” by making a system call (such as an I/O request) that causes it to sleep, in which case another process is selected to run. If not, at the end of a time slice, the kernel will choose which process to run next based on process priority: if two runnable processes have the same priority, the kernel runs them in turn.

There are three major types of jobs that can run on IRIX for Onyx2: time-sharing/interactive, real-time, and background (batch). Priorities in IRIX for Onyx2 range from 0 (low) to 255 (high): real-time processes can be assigned a priority anywhere in the entire range, background processes (with priority 0) run when free cycles are available, while time-sharing processes can have priorities anywhere in the range from 0 to 40. The kernel chooses which interactive/time-sharing process to run based on the currency-

based scheduling mechanism—the chosen process then runs on the processor with a non-degrading default priority of 20 until it sleeps on a system call or reaches the end of its time slice.

Scheduling features in IRIX for Onyx2 include:

- Gang scheduling
- Affinity scheduling
- Frame scheduling

Beginning with IRIX 6.4 for Onyx2, deadline scheduling and processor sets are no longer supported.

9.3.1 Gang Scheduling

The user can schedule related processes or threads to run as a group. Threads that communicate with each other using locks or semaphores can be scheduled to start in parallel. IRIX then attempts to schedule the related treads concurrently, provided the parent process has earned enough CPUs. Gang scheduling helps ensure that a thread holding a lock is scheduled in the same time interval as another thread that is waiting on the lock, to avoid having the second thread spin while the first thread is not running.

9.3.2 Processor Affinity

As a process executes on a processor, it causes more and more of its data and instruction text to be loaded into processor cache and contiguous main memory. This creates an "affinity" between the running process and that CPU: no other process can use that CPU as effectively, and the process cannot execute as fast on any other CPU.

The scheduler on a multiprocessor based on IRIX automatically notes the CPU on which a process last ran, and attempts to run a process on that same CPU, based on a calculation that some of the process data remains in cache and possibly local memory on that CPU.

9.3.3 Frame Scheduling

Real-time applications by their nature require a precise rate of forward progress, hence real-time users need complete control over a fixed set of system resources. The REACT™/Pro Frame Scheduler is a process execution manager that completely takes over scheduling and dispatching processes on one or more CPUs. The Frame scheduler makes it easier to organize a real-time program as a set of independent processes, cooperating to ensure that activities happen on time, as scheduled in a predefined sequence for data sampling, visual simulation, and related applications.

9.3.4 Deadline Scheduling: Not Supported

Deadline scheduling was a feature in earlier versions of IRIX, which allowed the real-time user to “guarantee” a certain proportion of processing time to a process in successive time intervals, without being able to “guarantee” where in each time interval the cycles would be available. Deadline scheduling has been superseded by frame scheduling; consequently support for deadline scheduling has been deleted since IRIX 6.4 and subsequent IRIX releases for Onyx2.

9.3.5 Processor Sets: Not Supported

Processor sets were a feature in earlier versions of IRIX, which allowed the system administrator to reserve sets of processors to run specific tasks, in effect partitioning the system into groups of CPUs that could run batch, real time, and time-sharing jobs separately. IRIX for Onyx2 distributes scheduling and provides much finer-grained control over process priorities, making processor sets obsolete.

9.4 Symmetric Multiprocessing and Multitasking

IRIX 6.5 for Onyx2 supports up to 128 MIPS RISC processors, with the option of support for even larger numbers should such configurations become available. Multiprocessor versions of IRIX have been available since 1988. Since that time, access to kernel resources has become more and more optimized. Thousands of locks ensure that access to individual data structures is synchronized rather than allowing access to entire subsystems. This reduces the likelihood that any given processor must wait for another processor to release a locked resource. As a result, IRIX offers exceptional scalability and performance as processors are added.

In addition to a re-entrant multiprocessing kernel, IRIX provides interrupt thread support and two types of user threads, with the appropriate synchronization mechanisms in each environment. User threads allow developers to create high-performance applications for single-job throughput across multiple CPUs and more modular and efficient applications on uniprocessors. Interrupt threads allow device driver writers and real-time developers to develop higher performance code more easily.

9.4.1 Sprocs and Sproc Synchronization

Sproc() is a system call that permits a developer to create an additional user process that shares the virtual address space (and potentially other attributes) of the parent process. The parent and child both have their own program counter, register contents, and stack pointer, but all the text and data in the shared address space is visible to both processes. There are three basic synchronization mechanisms to serialize access to shared data using sprocs: semaphores, locks, and barriers.

Sprocs() have long provided the basic mechanism upon which many parallel programs have been built on Silicon Graphics platforms. Beginning with IRIX 6.4 for Onyx2, POSIX threads become the preferred threading mechanism for new applications.

9.4.2 POSIX Threads and Synchronization Primitives

A POSIX (1003.1c-1995) thread is an industry-standard lightweight user execution entity that can share text and data with other threads within a user process context. POSIX threads are lightweight (in general and compared to sprocs, not much more than a program counter, stack, and register contents), enabling quick context switches and low-cost thread creation/destruction.

POSIX threads provide a mechanism for developers to enable and exploit parallelism in programs that run on a variety of vendor-compliant multiprocessing platforms. POSIX threads are also useful across the vendor spectrum of uniprocessors for managing asynchronous behavior or structuring applications composed of many logically distinct paths (e.g. simulations, windowing systems).

There are three basic mechanisms to serialize access to shared data using POSIX threads:

- Mutexes
- Condition variables
- Semaphores (as per POSIX 1003.1b)

POSIX threads are available on Silicon Graphics systems running IRIX 6.2 or subsequent releases. POSIX threads do not replace sprocs, both threading models are supported (though not in the same application) to ensure compatibility with the full range of new and existing applications.

9.4.3 Interrupt Threads and Synchronization Primitives

Traditional IRIX (prior to 6.2) allowed multiple traditional heavyweight processes to run within the kernel, where critical regions were protected with mutexes and spinlocks. Since the kernel did not support lightweight threads independent of a user process, device driver writers and real-time programmers worked around the absence of interrupt threads with device interrupt routines. Programming these workarounds was difficult; the driver code that emerged was complex and inefficient.

Interrupt threads provide a lightweight kernel execution entity that shares common text and data space with other threads independent of a user process context. Interrupt threads are easy to create and destroy, are schedulable, and use common, familiar kernel synchronization primitives:

Non-sleeping locks

- Spinlocks

Sleep-wakeup locks

- Mutexes
- Semaphores
- Multi-reader locks

Interrupt threads offer new capabilities for device driver writers and real-time programmers:

- Device driver code is simpler (common primitives) and more efficient (sleep-wakeup locks).
- Driver performance is better (low cost thread create and destroy)
- Real-time performance is more predictable because of better preemption

9.5 Compatibility

Every effort has been made to maintain or expand compatibility among IRIX versions.

9.5.1 COFF Obsoleted

IRIX 5 and IRIX 6 binaries are produced using the SRV4 Executable and Linking Format (ELF), which replaces the ECOFF object format in IRIX 4 and enables Dynamic Shared Objects. Development and execution of COFF (IRIX 4 and earlier) binaries is not supported on IRIX 6.5 for Onyx2, and has not been supported since IRIX 6.2. A utility to find and identify COFF-dependent binaries is provided with the release and installed on the system. It may be executed before IRIX 6.5 for Onyx2 is installed.

9.5.2 Binary Compatibility between IRIX 5.3/IRIX 6.2/IRIX 6.4 and IRIX 6.5 for Onyx2

Nearly all binaries built on IRIX 5 or IRIX 6 can be run under IRIX 6.5 for Onyx2. It may be necessary to recode or recompile to take advantage of IRIX 6 enhancements. There may be rare cases—such as applications that examine internal operating system data structures—in which application code must be recompiled.

Symbols
"housekeeping" operations 45
"pass-through" mode 52

Numerics
10-bit RGB 15
12-bit RGBA 15
16 bit textures 29
32 bit textures 29
48 bit textures 29

A
Alpha values 23
Ambient 23
anti-aliasing 26
Atmospheric Effects 39

B
batch processing 42
Billboards 37
blending equations 23
broadcast-quality composite video 16

C
CCIR601 16
Channel Configurations 45
color blending 23
color component swapping 39
Color index operations 22
color matrix 39
color matrix operation 39
Color rasterization operations 22
color space conversion 38
component replication 39
composite sync-on-green 16
convolution filters 38
convolution operator 38

D
DAC Output Bandwidth 47
DACs 16
decal 34
depth (Z) buffer 40
Detail Management Levels 37
Detail Texture 36
DG4 board 39
Diffuse 23
Disjoint Buffers and pbuffers 41
display list storage cache 21
Display lists 21
display lists 21
dynamic data 21
Dynamic resampling 52

E
Emission 23
Euro 625 resolution 16

F
floating point data types 21
fog 14
fog color 14
fog/haze function 39
framebuffer 39
Framebuffer Memory 29, 48
framebuffer memory 40
Framebuffer Read/Write Bandwidth 48
framebuffer-to-video-subsystem bandwidth 47
frame-synchronized multichannel output 53

G
genlock 53
geometric processing 21
Geometry Engine (GE) board 13
Geometry Engine (GE) processors 13
Geometry processing 13
geometry subsystem 13
graphics pipeline 20
graphics primitives 20

H
Hardware lighting 23
hidden surface removal 25
Histograms 39
Host Memory 29

I
illegal combinations 46
ImageVision Library® (IL) application 15
imaging datapath 39
Imaging Operations 38
InfiniteReality Video Format Compiler 53
IRIS Performer™ software 52

L
legal combination of formats 46
letterboxing 16
linear color space conversion 39
Local light sources 24
Look up Tables 39
Luminance 52
LUTs 39

M
magnification 37
master video format 53
MIMD 13
minimum memory per pixel 18
minmax 39
modulation 34
monochrome renderin 52
Multi-Channel Display Generator 42
multipass rendering algorithms 42
multiple primitives 22
multisample anti-aliasing mode 26
multisampled anti-aliasing 26

N
NTSC 16

O
occulting surfaces 23
offscreen rendering 40
OpenGL color and lighting models 22
OpenGL routines 20

P
packed vertex arrays 21
PAL 16
PBuffer (Pixel Buffer) 41
Performance bottlenecks 21
Perspective Correction 36
Pixel depth 47
Pixel processing support 13
primitive data 21
Programmable Video Timing 52
projection 37

R
raster primitives 26
raster subsystem 14
rasterization 21
rectangular frame-buffer 48
Rendering 21
rendering pipeline 21
resolutions 18
RS-170 16
RS-343 standard 52

S
Sample Memory 27
Scan conversion 14
SGIX_pbuffer OpenGL extension 41
shadow map 37
shadowing 37
sharp texture feature 37
shininess 24
SIMD 13
Specular 23
spot-lighting effects 37
stereoscopic viewing 52
stereoscopy 52
surface types 24
S-video encoder outputs 16
Swap Rates 45
Synchronization 28

T
texels 14
texture coordinate data 14
texture element (texel) sizes 29
texture environment equation 14
texture memory 34
texture processing unit 14
Texture Transparency and Contouring 36
tile 48
time-of-day lighting 23
transformations 20

transparency 25
transparency/translucency 25
tuning algorithms 21

U
Ultra-high color resolutio 52
US HDTV standard 16

V
VCRs 16
vertex arrays 21
vertical sync rate 53
video display subsystem 16
video encoder channel 52
Video Format Combinations 50, 51
video formats 42
video refresh 48
video resizing 49
video source 53
video timing 42
VLSI processors 14

W
window/level operation 39

X
X11 standard 57
X-Windows 49

Z
Z-buffer value 25
Z-buffering technique 25

SiliconGraphics
Computer Systems

Corporate Office
2011 N. Shoreline Boulevard
Mountain View, CA 94043
(650) 960-1980
URL: http://www.sgi.com

U.S. 1(800) 800-7441
Europe (44) 118-925.75.00
Asia Pacific (81) 3-54.88.18.11
Latin America 1(650) 933.46.37

Canada 1(905) 625-4747
Australia/New Zealand (61) 2.9879.95.00
SAARC/India (91) 11.621.13.55
Sub-Saharan Africa (27) 11.884.41.47