

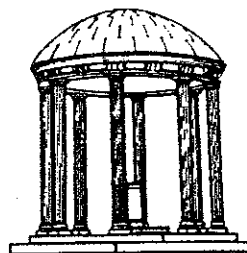
High Performance Computer
Graphics Architectures

TR89-026

July, 1989

Roger Brusq

The University of North Carolina at Chapel Hill
Department of Computer Science
CB#3175, Sitterson Hall
Chapel Hill, NC 27599-3175



UNC is an Equal Opportunity/Affirmative Action Institution.

High Performance Computer Graphics Architectures

This report is a review of several existing or annouced high performance Computer Graphics Architectures ranging from 20,000 to 1 Million triangles per second

Roger BRUSQ

Department of Computer Science
Sitterson Hall
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175

July, 15 1989

1. INTRODUCTION

This report consists of three parts:

-part A is a brief review of some recent "Graphics Supercomputers" architectures: AT&T Pixel Machine, Alliant Visualization Series, Megatek Sigma 70, Apollo DN10000, HP 835 Turbo SRX.

•part B is a more detailed presentation of the basic concepts and performances of three High Performance Computer Graphics Systems which performances range from 100 to 200 K triangles/seconde, i.e an order of magnitude higher than the previous generation (10,000 to 20,000 triangles/sec.):

- ARDENT TITAN Graphics Supercomputer,
- SILICON GRAPHICS POWER IRIS Series,
- STELLAR GS1000 Graphics Supercomputer.

These three machines are among the most powerful now available. They all implement a parallel processing approach to speed up both the geometry computations and the rendering process, combining the performance of minisupercomputers with real-time high quality graphics.

The section reviews the overall structure of each system and the different ways they implement the Graphics Pipeline.

•part C is devoted to some new designs achieving an order of magnitude increase in rendering rates over previous systems: 1 Million triangles/second:

-SAGE: The Systolic Array Graphics Engine, by Gharachorloo et al. (IBM Research Division, Watson Research Center): 1 Million polygons per second (Z-buffered, Gouraud shaded);

-The Triangle Processor, by Michael Deering et al. (Schlumberger Palo Alto Research): 1 Million triangles per second (Z-buffered, Phong shading);

-Pixel-Planes 5: 1 Million triangles per second (100 pixels, Z-buffered, Phong shaded).

2. PART A

A review of some recent Computer Graphics Architectures

2.1 Introduction

A general approach to 3D graphics systems is outlined in figure 1-1.

A Graphics Processor traverses a display list containing the geometric primitives in the scene, computes viewing transformations, lighting calculations, clipping and scaling. Computational load for these functions increases roughly linearly with the number of primitives in the scene.

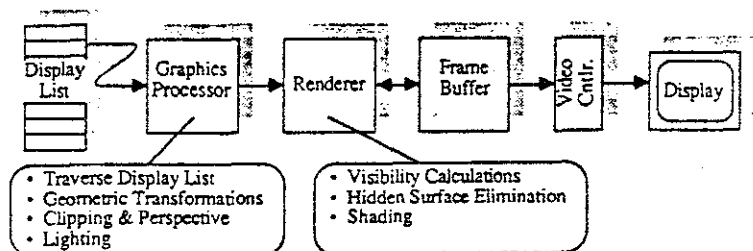
The Renderer determines which pixels are visible in each primitive and generates their final color into the frame buffer. Computation load here can explode to the order of the number of primitives times the number of pixels per primitive. In most graphics systems, computation and bandwidth limits in the renderer determine the maximum rate of image generation

The Video Controller reads and converts data from the Frame Buffer into color streams sent to the Display device.

The functional organization of a 3D graphics system such as described in figure 1-1 is basically a pipeline from the object database to the display screen., and the expression "Graphics Pipeline" will frequently appear in the following paragraphs.

Each new Graphics Supercomputer introduces some innovations into one or more of the three major components of this pipeline: geometric transformations/clipping/perspective, image rendering, and scan-out from image buffer.

In this section we review some of the recent implementations of this Graphics Pipeline: AT&T Pixel Machine, Alliant Visualization Series, Megatek Sigma 70, Apollo DN10000, HP 835 Turbo SRX.



Block diagram of a typical 3-D graphics system.

figure 1-1
from Fuchs & Poulton -An Architecture for Advanced Airborne Graphics Systems-May,1988

2.2 The AT&T Pixel Machine

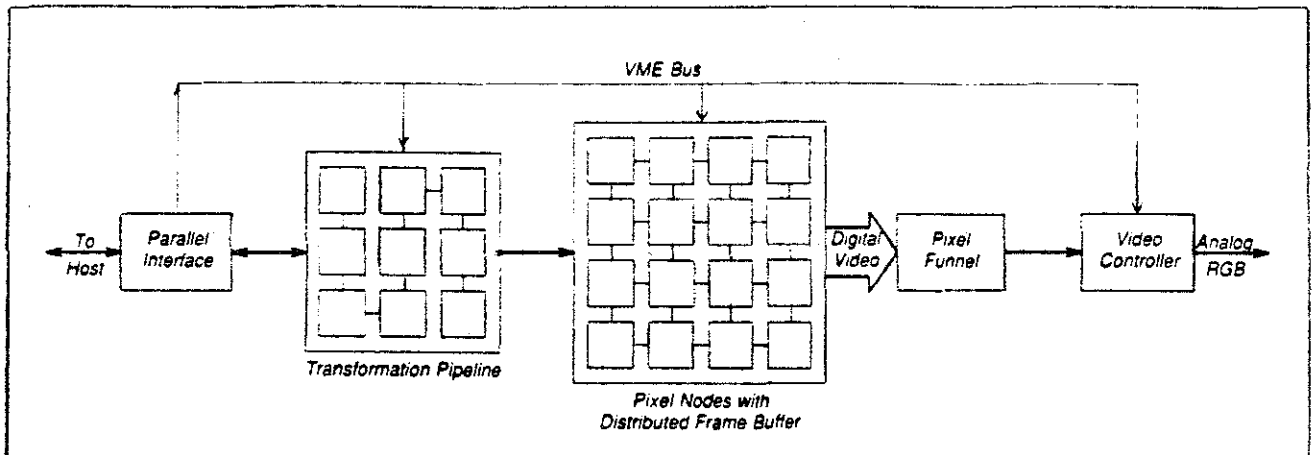


figure 2.2-1 Pixel Machine block diagram.
from [AT&T-1]

I- Overview

The figure 2.2-1 shows how the Pixel Machine combines coarse grain pipelining and MIMD computing arrays to provide the expected performance for both image synthesis and image analysis applications

The design philosophy is:

- use floating-point computation and large image memories, which are useful for image processing;
- design simple, modular processors that can be repeated a number of times to build a system;
- implement all algorithms in software.

The Pixel Machine is connected to the host via the VME bus, through the parallel interface.

The transformation pipeline comprises 9 or 18 pipe nodes configurable as one or two pipelines (see figures 2.2-1 and 2.2-4).

The rendering function is closely integrated with the frame buffer distributed memory in the MIMD Pixel Nodes array (see figures 2.2-1 and 2.2-3).

The transformation pipeline feeds primitives to parallel pixel nodes through a broadcast bus (see figure 2-2.3)

The pixel funnel rearranges pixels from the frame buffer into a properly ordered raster scan sequence presented to the display device through the video controller

The broadcast bus and the pixel funnel are physically part of the VME backplane.

Each pipe node and pixel node can be viewed as a small independent computer that executes its instructions and operates on data asynchronously with all the other nodes. Program are loaded into the nodes by the host, using unique, software defined node numbers to distinguish between them.

II- Pipe Nodes

The pipe node is built around a 5 MIPS, 10 MFLOPS DSP32 processor, with parallel DMA interface to the VME bus (see figure 2.2-2).

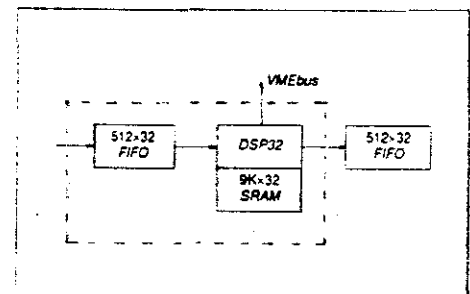


figure 2.2-2 Pipe node block diagram.
from [AT&T-1]

The host provides input to the first node via the VME bus. The output from the last node is broadcast to all of the pixel nodes and this node is also connected to the VME bus through a second output FIFO.

The Transformation Pipeline can have 9 pipe nodes or 18 pipe nodes software configurable as one 18-node pipeline or two 9-node pipelines (figure 2-2.4). In the case of two parallel pipelines, the broadcast bus is time-shared by the two ending nodes

The pipe nodes perform 3D transformations, clipping, projections, shading and image filtering. The pipeline can also be used as a hardware subroutine by processes running in the host computer, which can send data to the first node and read results from the last one.

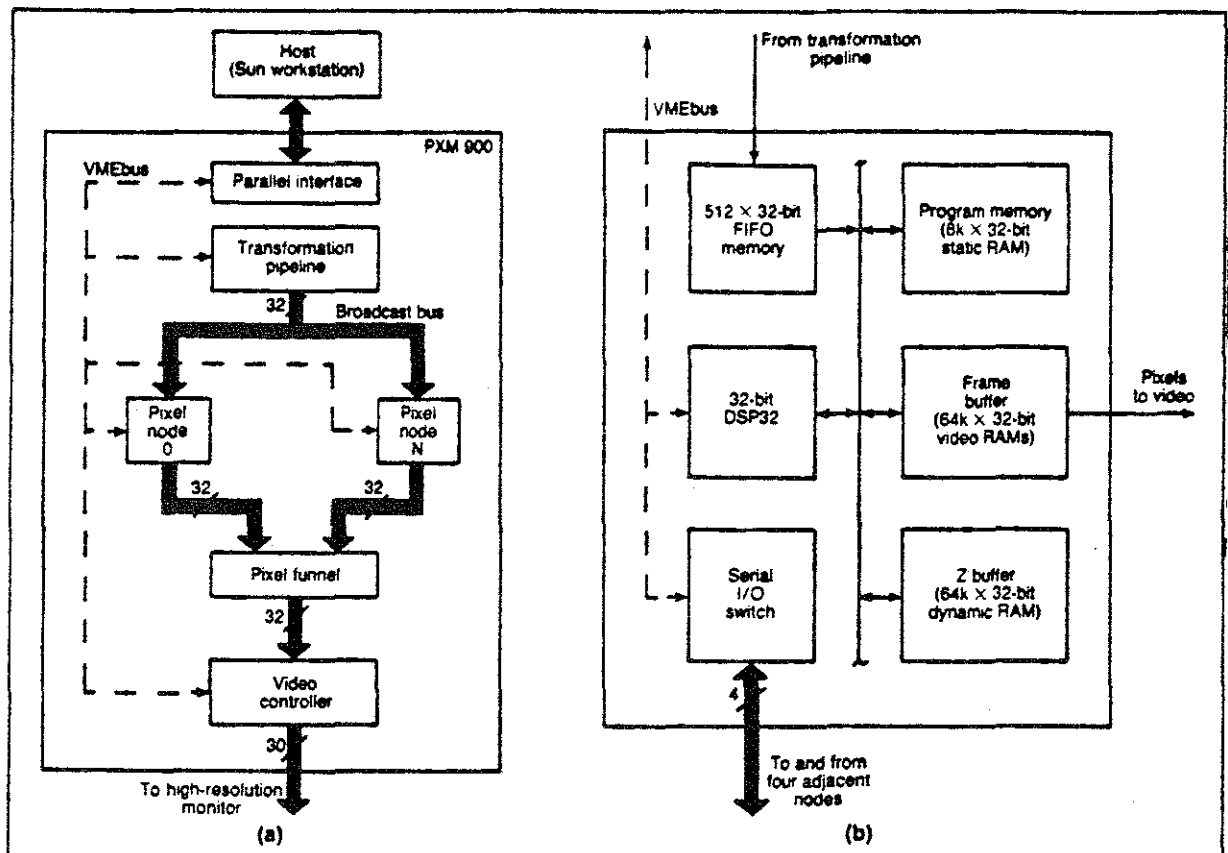


figure 2.2-3 from [Tunik87]

In AT&T's PXM 900 image-display system, the transformation pipeline feeds primitives to parallel pixel nodes through a broadcast bus (a). Inside each pixel node are double-buffered video RAMs that constitute in part the system's large frame buffer (b). Depth values reside in the Z buffer, which also could supply storage space for large images.

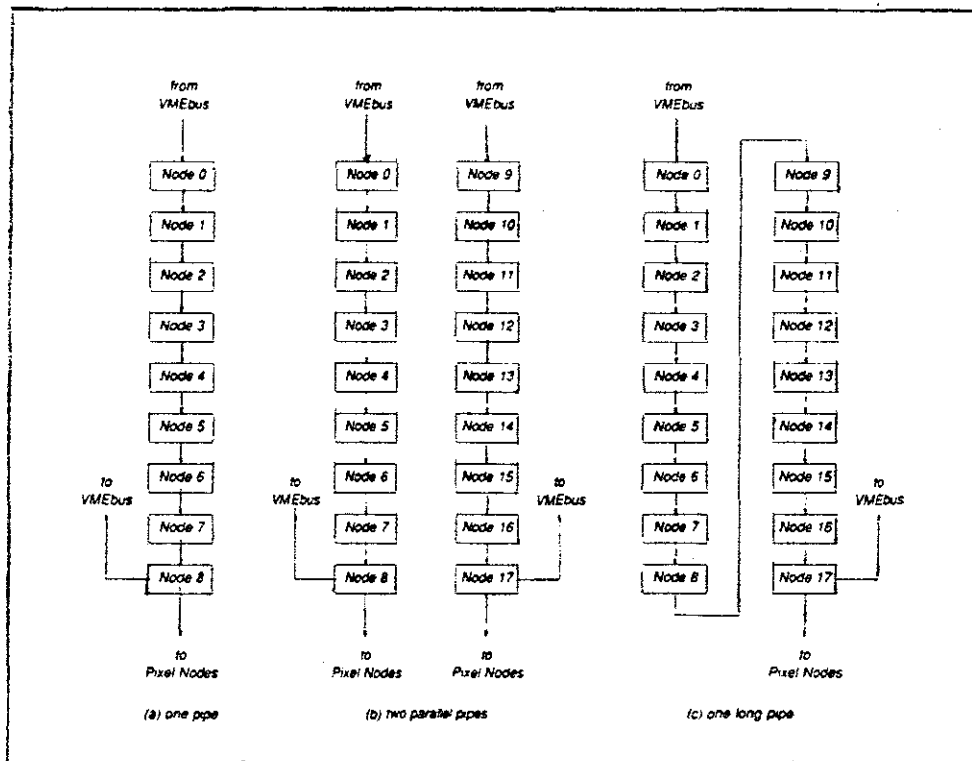
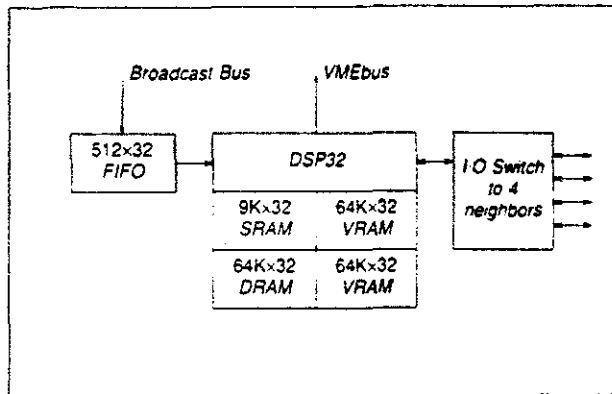


figure 2.2-4 from [AT&T-1]

Pipeline configurations

III- Pixel Nodes

The pixel node is also built around the DSP32, with an input FIFO and 9K 32-bit words of static RAM. In addition, it is provided with a 4-way multiplexed I/O switch for communication with the 4 neighbours, plus two banks of 64Kx32 bit VRAMs as part of the frame buffer, and 64Kx32 bit dynamic RAM for Z-buffering or data store. (figures 2.2-3 and 2.2-5)



Pixel node block diagram.

figure 2.2-5
from [AT&T-1]

The pixel nodes form an nxm array with a distributed frame buffer. They receive their data from the broadcast bus and store output into the frame buffer or return it to the host.

The Pixel Machine can be configured with 16,20,32,40, or 64 pixel nodes (figure 2.2-6). The VRAM and DRAM can be organized as 3 blocks of 256x256 32-bit pixels, or as smaller blocks called subscreens (e.g. 128x128). A physical pixel node can be time-shared between several processes each applying to a different single subscreen: this the concept of virtual node. A physical pixel node can contain up to 4 virtual nodes depending on the configuration of the system (figure 2.2-6).

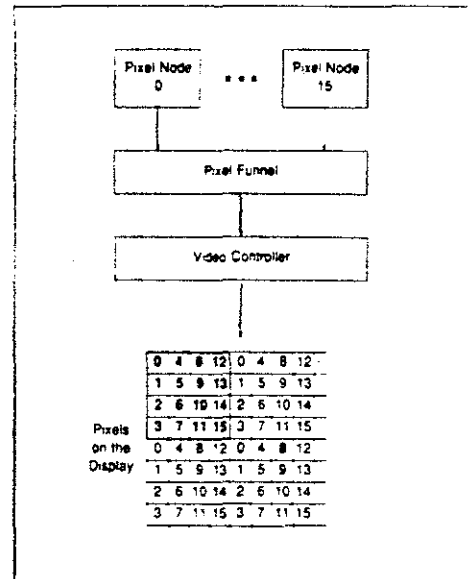
| Pixel Nodes | Pixel Node Array | | Display Resolution | Subscreens | | |
|-------------|------------------|---------|--------------------|------------|----------|----------|
| | physical | virtual | | Size | # pixels | per node |
| 16 | 4x4 | 8x8 | 1024x1024 | 128x128 | 16384 | 4 |
| 20 | 5x4 | 10x8 | 1280x1024 | 128x128 | 16384 | 4 |
| 32 | 8x4 | 8x8 | 1024x1024 | 128x128 | 16384 | 2 |
| 40 | 10x4 | 10x8 | 1280x1024 | 128x128 | 16384 | 2 |
| 64 | 8x8 | 8x8 | 1024x1024 | 128x128 | 16384 | 1 |
| 64 | 8x8 | 8x8 | 1280x1024 | 160x128 | 20480 | 1 |

Pixel array configurations.

figure 2.2-6
from[AT&T-1]

IV- Video display

The frame buffer is distributed throughout the array of pixel nodes (figure 2.2-7). The pixel funnel rearranges pixels from the frame buffer into a properly ordered raster scan sequence. Both the video processor and the pixel funnel are software configurable for the five different pixel arrays.



Pixel mapping in the distributed frame buffer.

figure 2.2-7
from [AT&T-1]

The frame buffer stores the four components of a pixel (red, green, blue, alpha) as 16-bit signed integers from which 8 bits are actually used today.

The video processor has two sets of six 256x10 lookup table for color mapping: one set of high-speed video tables used to convert video data. The other set are shadow tables which can be accessed from the VME bus. the contents of the shadow tables are copied to the video tables during a vertical retrace period.

The video resolution is either 1024 x 1024 or 1280 x 1024, 60 Hz non-interlaced, or 720x485, in NTSC mode, or 702x575 in PAL mode.

V. System configurations and performances

Physically, the Pixel Machine is a free-standing cabinet containing pipeline cards, pixel array cards, video processor cards and parallel interface cards.

The host computer accesses the Pixel Machine as a 64K byte block of memory on the VME bus. The memory is mapped into user process space, the parallel interfaces to each node, the I/O FIFOs in the pipeline, the colormaps and the video control registers are all mapped into this memory block.

The Pixel Machine performs Gouraud shading at 16 Mpixels/sec.

Figure 2.2-8 shows the different configurations of the Pixel Machine: 5 different pixel array sizes and 2 different pipeline sizes.

| Model Number | Nodes | | Peak Performance | | Memory (Mbytes) | Butters | | | Bytes per pixel |
|--------------|-------|-------|------------------|--------|-----------------|---------|-----|-------|-----------------|
| | pipe | pixel | MIPS | MFLOPS | | rgba | z | | |
| 916 | 9 | 16 | 125 | 250 | 12 | 2 | 1 | 12 | |
| 916D | 18 | 16 | 170 | 340 | 12 | 2 | 1 | 12 | |
| 920 | 9 | 20 | 145 | 290 | 15 | 2 | 1 | 12 | |
| 920D | 18 | 20 | 190 | 380 | 15 | 2 | 1 | 12 | |
| 932 | 9 | 32 | 205 | 410 | 24 | 4 | 2 | 24 | |
| 932D | 18 | 32 | 250 | 500 | 24 | 4 | 2 | 24 | |
| 940 | 9 | 40 | 245 | 490 | 30 | 4 | 2 | 24 | |
| 940D | 18 | 40 | 290 | 580 | 30 | 4 | 2 | 24 | |
| 964 | 9 | 64 | 365 | 730 | 48 | 4.8 | 2.4 | 24.48 | |
| 964D | 18 | 64 | 410 | 820 | 48 | 4.8 | 2.4 | 24.48 | |

Pixel Machine configurations.

figure 2.2-8
from [AT&T-1]

VI. References for the AT&T Pixel Machine:

-[AT&T-1]-AT&T- The Pixel Machine System Architecture- A Technical Report;

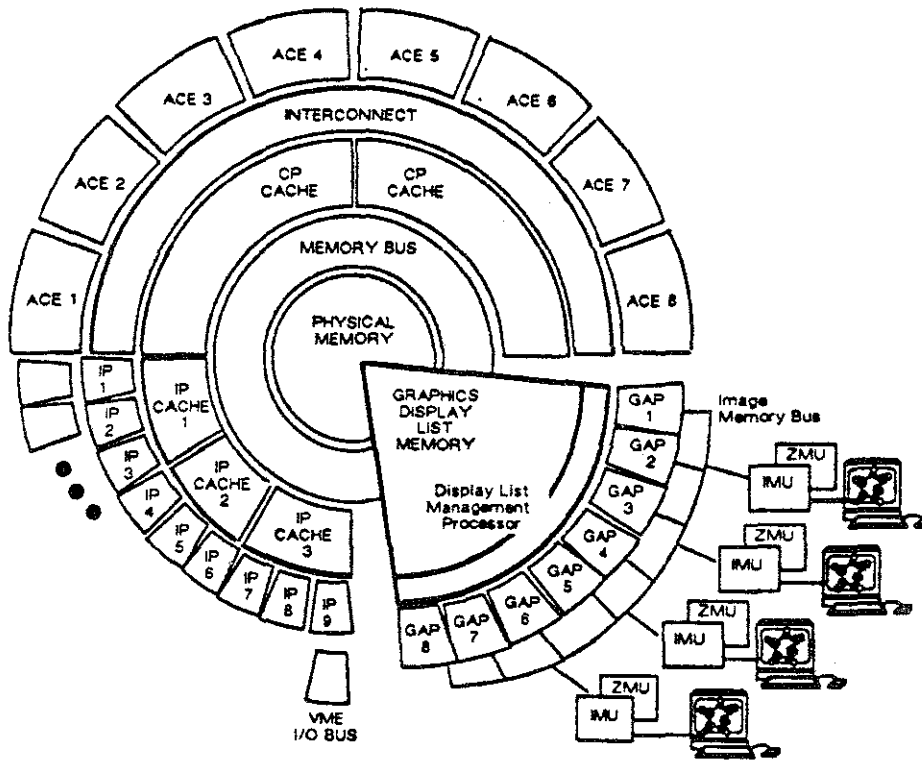
-[AT&T-2]-AT&T Pixel Machines PXM 900 Series- Product Specification, 7/87;

-[AT&T-3]-AT&T-The Pixel Machine 900 Series, 6/88;

-[Tunik87]-Diane Tunik- Powerful display system revs up image and graphics processing- Electronic Design, July, 23, 1987;

-[Runyon87]- Stan Runyon - AT&T goes to "warp speed" with its graphics engine;

2.3.The ALLIANT Visualization Series



Alliant's visualization architecture

figure 2.3-1
from [Alliant-1]

I- Overview

The Alliant "Visualization Series" is a family of "Visual Supercomputers" designed for interactive analysis and visualization. The design is based on the combination of the Alliant FX/Series supercomputers and the Raster Technologies GX4000 graphics architecture.

The Visualization Series architecture provides separated dedicated processors for applications and graphics processing. Applications processing are supported by up to eight Advanced Computational Elements (ACE), 64-bit parallel vector processors. Interactive jobs and operating system tasks are supported by Interactive Processors (IP). Typical graphics functions (transformations, tessellation, clipping, lighting) are supported by up to eight parallel graphics arithmetic processors (GAP).

The graphics processing system functions independently of the applications processors but is coupled to them through high speed global main memory.

II- The Graphics Processing System

see figure 2.3-2

The graphics processing system consists of:

- display list memory board (DLM) with display management processor;
- one or more graphics arithmetic processors (GAPs);
- one or more image memory units (IMUs);
- one or more Z-buffer memory units (ZMUs);

The DLM board contains a dedicated display list memory (eight to 32MB) and a 20-MIPs bit-slice processor. It performs display list management and traversal in hardware and software. The display list memory is multiported and connects to the main memory bus allowing for the ACEs to directly create or edit display lists.

The GAPs perform the initial processing of the graphics commands transferred from the DLM board: drawing primitives, lighting models, rendering surface properties, smooth shading (Gouraud and Phong), calculating hierarchic structures for complex 3D objects, 4x4 transformations, clipping, picking, depth cuing. Then the Gaps transfer low-level drawing primitives (points, vectors, triangles) to the IMUs over the dedicated IMU bus. The GAPs implement PHIGS+.

GAP synchronization and arbitration is accomplished over a dedicated 32-bit command bus by concurrency control logic on each GAP. Priority is given to GAPs executing commands that require the least amount of time to process.

The IMUs receive low-level drawing primitives from the GAPs and then process these primitives to draw pixels into image memory. Each IMU contains multiple custom VLSI rendering or pixel processors that directly execute vectors, triangles, rectangles and bitblt operations, and perform image reads and writes and pan and zoom functions. Each IMU contains also a 1280 x 1024 x 24-bit frame buffer and four windowing control planes.

Each IMU supports 24-bits true color and 8-bits double buffered pseudo-color. Three IMUs may be used to

provide 24-bit double-buffering.

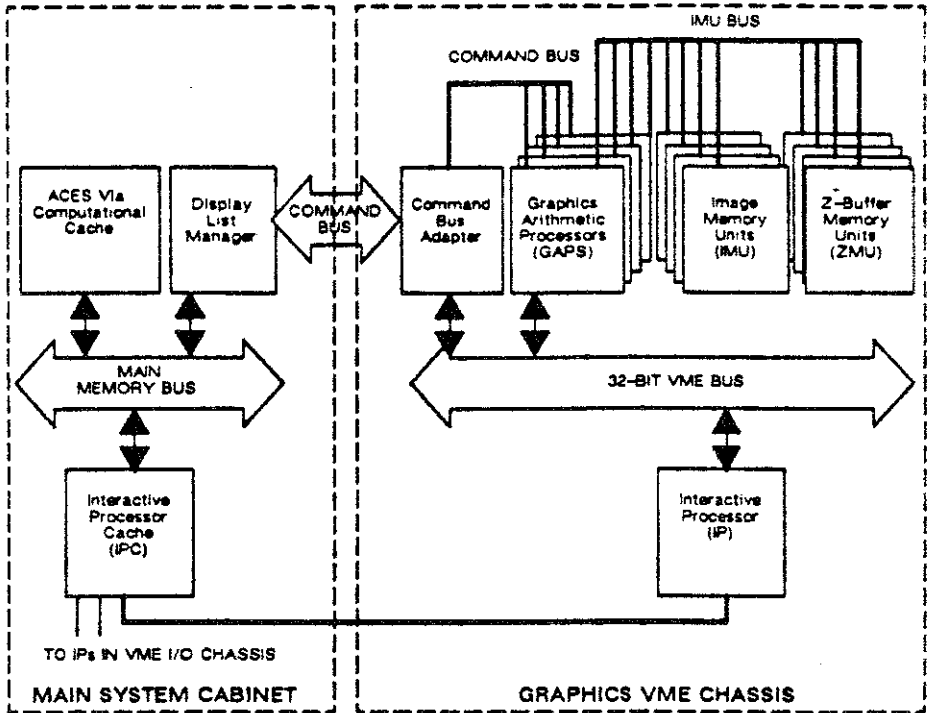
Each IMU drives an RGB, 1280 x 1024, 60 Hz non-interlaced video display. Multiple IMUs can be used to provide multiple simultaneous displays.

The ZMU performs the hidden-surface removal in parallel with the rendering done by the IMU. The ZMU contains the same processors found in the IMU, and a 1280 x 1024 x 16-bit Z buffer.

The ZMU and the IMU are connected through a dedicated high-speed bus.

Both of them contain five custom VLSI chips: a master controller and four scan-line processors (see section on ARDENT Titan Graphics Supercomputer figure 3.2-7). The master controller parses the drawing command stream and performs setup and control operations for the scan-line processors. Each of the scan-line processors processes one fourth of the image memory.

Together the five chips can achieve a drawing performance of up to 50 Mpixels/sec.



Parallelism in the graphics processing system is hardware-based for fast, low-overhead operation. GAP synchronization and arbitration is performed by hardware concurrency control logic on each GAP that communicates over a dedicated command (CMD) bus. The GAPs transfer low-level drawing primitives to one or more IMUs over a dedicated IMU bus.

figure 2.3-2
from [Alliant 1]

Multiple Simultaneous displays are supported by the architecture (see figure 2.3-3). One DLM card supports up to eight GAPs, which in turn support up to four IMU/ZMU pairs or, for double-buffered true-color, up to two sets of three IMUs and one ZMU. Each IMU (or set of three IMUs in the case of double-buffering) can drive an RGB video display. Therefore a complete graphics processing system can provide up to 4 simultaneous displays. The VFX/80 system can accommodate an additional graphics processing system to provide up to eight simultaneous displays.

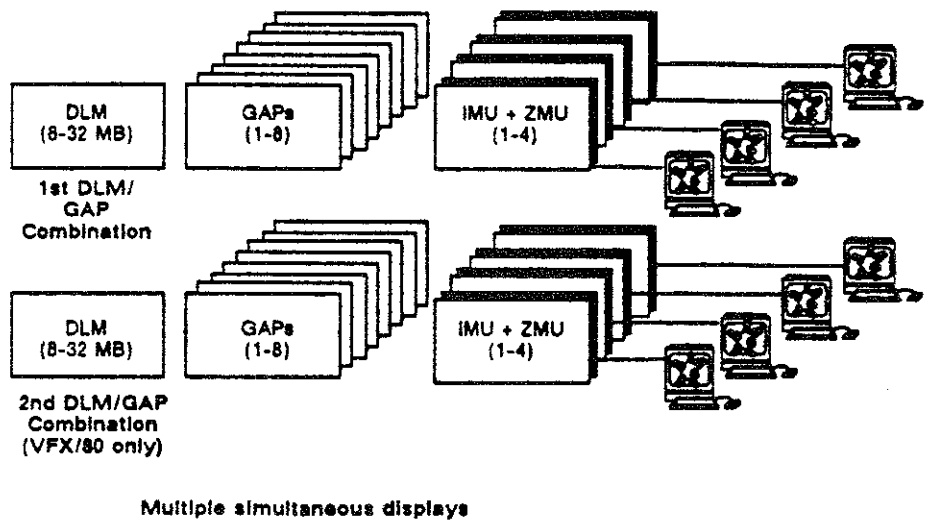


figure 2.3-3
from [Alliant-1]

III- Visualization Series configurations and performance.

Aside from the graphics processing system, the Visualization Series models are identical to the corresponding FX/Series models. See figures 2.3-4 to 2.3-6 for a summary of configuration options and performances.

| | VFX/4 | VFX/40 | VFX/80 | VFX/82 |
|------------------------------------|---------|---------|--------------------|---------------------|
| Number of Computational Processors | 1 - 4 | 1 - 4 | 1 - 8 | 16 |
| Peak Computational MFLOPS | 47.2 | 94.4 | 188.8 | 377.6 ¹ |
| Interactive Processors (IPs) | 1 - 3 | 1 - 3 | 1 - 9 ² | 1 - 18 ³ |
| Number of VME Channels for I/O | 2 | 2 | 8 ² | 16 ³ |
| Main Memory (max) | 128 MB | 128 MB | 256 MB | 512 MB ⁴ |
| Number of Graphics Processors | 1 - 8 | 1 - 8 | 1 - 16 | 1 - 32 |
| Peak Graphical MFLOPS | 160 | 160 | 320 ¹ | 640 ¹ |
| Display List Memory (max) | 32 MB | 32 MB | 64 MB ¹ | 128 MB ¹ |
| Number of visualization displays | 1 - 4 | 1 - 4 | 1 - 8 | 1 - 16 |
| Maximum Disk Storage ⁵ | 13.2 GB | 13.2 GB | 52.8 GB | 105.6 GB |

¹ Aggregate
² Max number of IPs = 6, number of VME channels = 4, if more than 4 visualization displays are used
³ Max number of of IPs = 12, number of VME channels = 8, if more than 8 visualization displays are used
⁴ Using 8", 550 MB (formatted) SMD drives

figure 2.3-4
from [Alliant-1]

| GRAPHICS PERFORMANCE | | | | |
|-------------------------------------|----------------|--------|---------|---------|
| 3D SHADED POLYGONS/SEC. | | | | |
| PATCH TYPE | NUMBER OF GAPS | | | |
| | 1 | 2 | 4 | 8 |
| 10 pixel triangle, flat shading | 30,000 | 59,000 | 117,000 | 225,000 |
| 10 pixel triangle, Gouraud shading | 18,000 | 35,500 | 70,000 | 136,000 |
| 10 pixel triangle, w/light source | 8,500 | 16,600 | 32,500 | 62,000 |
| 100 pixel triangle, flat shading | 30,000 | 59,000 | 117,000 | 157,000 |
| 100 pixel triangle, Gouraud shading | 18,000 | 35,500 | 70,000 | 136,000 |
| 100 pixel triangle, w/light source | 8,500 | 16,600 | 32,500 | 62,000 |

- All figures through PHIGS/PHIGS+
- Assumes independent polygons
- Assumes perspective and clip check against a single rectangular window with front and back Z-clipping
- All shading includes depth buffering


VFX Technical Overview 7/88-30 

figure 2.3-5
from [Alliant-3]

| GRAPHICS PERFORMANCE | | | | |
|---------------------------------|----------------|---------|---------|-----------|
| 3D VECTORS/SEC. | | | | |
| VECTOR TYPE | NUMBER OF GAPS | | | |
| | 1 | 2 | 4 | 8 |
| 5 pixel vectors, no perspective | 220,000 | 430,000 | 860,000 | 1,000,000 |
| 5 pixel vector, perspective | 170,000 | 330,000 | 640,000 | 1,000,000 |
| 10 pixel vectors, perspective | 220,000 | 430,000 | 860,000 | 900,000 |
| 10 pixel vectors, perspective | 170,000 | 330,000 | 640,000 | 900,000 |

- All figures through PHIGS/PHIGS+
- 32-bit floating-point vectors, transformed, clip-checked and rendered
- Clip check against a single rectangular window with front and back Z-clipping


VFX Technical Overview 7/88-28 

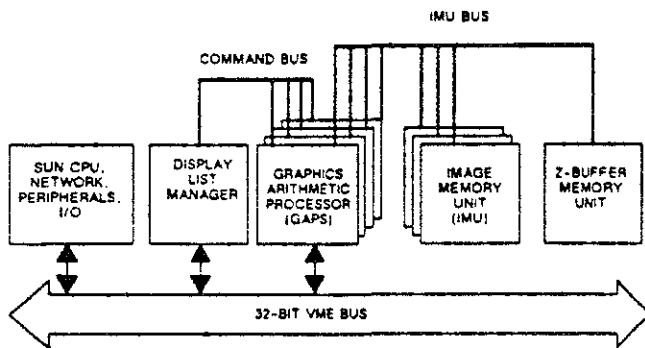
figure 2.3-6
from [Alliant-3]

The graphics processing system can also be integrated into a Personal Visualization station., based on a Sun-3/260 or Sun4/260 workstation from Sun Microsystems (see figure 2.3-7).

It uses the same graphics processing hardware as does the Visualization Series computers, but here the DLM communicates with the Sun CPU via the VMEbus.

IV- References for Alliant Visualization Series

- [Alliant-1]- ALLIANT Product Summary, Dec 1988
- [Alliant-2]- Alliant- Visualization Series
- [Alliant-3]- Alliant- VFX Technical Overview (exerpts), 7/88
- [Raster Tech]- Raster Technologies- GX4000, 3D Graphics for High-Performance Workstations.



Graphics processing architecture for the Personal Visualization Station .

figure 2.3-7
from [Alliant-1]

2.4.The MEGATEK SIGMA 70

In March 1989, Megatek announced its new Sigma 70 graphics workstation. One of the announced figures is 200,000 smooth-shaded polygons per sec.

It is a dedicated graphics processing system designed to be used with Sun 4 SPARC-based CPUs from Sun Microsystems.

I- Overview

Note: most of the "details" hereafter come from an e-mail exchange with Rusty Sanders (rgs@megatek.UUCP).

The system comprises 3 "smart" boards, each with at least one processor on it.

The first is the traversal engine, which has a single 16MIPs processor to read the display list, handling all display list jumps, refers and context switches.

The second is the transform processor, which has four floating processors operating in parallel in a SIMD fashion: one each for x,y,z and w transforms. The aggregate performance of the transform processor is 145 MFLOPs. It does 3D to screen-space transformations, and all clipping calculations.

In addition, the i,j and k axes of interpolation are also calculated, when doing lighting and shading at this level (Phong shading!). All polygon vertices are just handed down to the next board, no sorting is done on the transform board.

The third board is the vector processing board, which actually has three different SIMD bit-slice engines on it. The first does vertex sorting of polygons, the next calculates scanline endpoints, and the last does per pixel interpolations, sending x, y, z, i, j, k to the framebuffers through a color board which converts i, j, k to shading, at 25 ns per pixel. The cumulative power of these three SIMDs is 680 MIPs. A good proportion of that is in the last SIMD which does the 6 axis interpolation at 25 ns.

Graphics data are transferred over dedicated buses at speeds up to 280 MB/sec.

They also say that the frame buffer access is "heavily" interleaved, but the the pixels are not rendered one block, whatever the size, at a time. They are rendered individually at 25ns for triangles and 50 ns for lines!

Megatek® SIGMA™ Workstations

Window Management

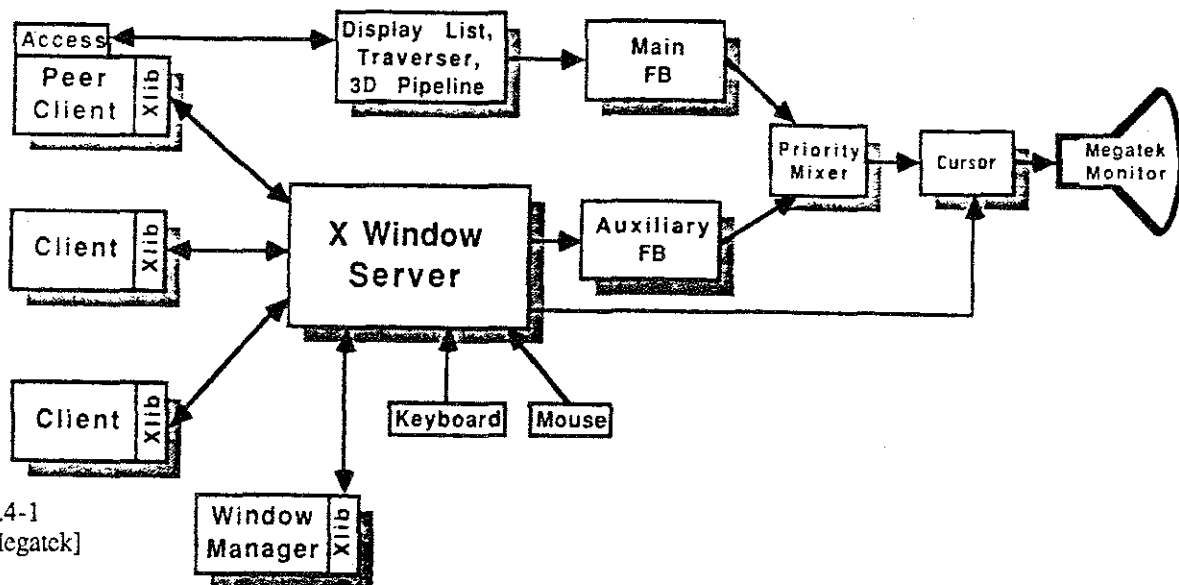


figure 2.4-1 from [Megatek]

II-Configurations and performance

The performances quoted in figure 2.4-2 are given for 10-pixel vectors and 100-pixel triangles.

| SPECIFICATIONS | | SIGMA 70 | | |
|-------------------------------|--|--|--|--|
| | <u>Model 4100A</u> | <u>Model 4100</u> | <u>Model 4200</u> | |
| System Features | | | | |
| CPU | Sun 4/100 (SPARC, 14.28 MHz) | Sun 4/100 (SPARC, 14.28 MHz) | Sun 4/200 (SPARC, 16.67 MHz) | |
| Performance | 7 MIPS | 7 MIPS | 10 MIPS | |
| Floating Point Unit | Weitek 1164/1165 | Weitek 1164/1165 | Weitek 1164/1165 | |
| System Memory | 8 - 32 MB | 8 - 32 MB | 8 - 128 MB | |
| System Bus | VME | VME | VME | |
| Ethernet™ Interface | Standard Thin Ethernet | Standard/Thin Ethernet | Standard Ethernet | |
| Graphics Features | | | | |
| Graphics Performance | | | | |
| 2D vectors/second | 2,000,000 | 2,000,000 | 2,000,000 | |
| 3D vectors/second | 1,250,000 | 1,250,000 | 1,250,000 | |
| Flat-shaded polygons/second | 240,000 | 240,000 | 240,000 | |
| Smooth-shaded polygons/second | 200,000 | 200,000 | 200,000 | |
| Pixels/second for polygons | 40 M | 40 M | 40 M | |
| Display List Memory | 4 - 16 MB | 4 - 16 MB | 4 - 16 MB | |
| Bit Planes | | | | |
| Graphics Frame Buffer | 16 bits, double-buffered (8 - 12 bit color and 8 - 4 bits for overlays, underlays or control) | 16 bits, double-buffered (8 - 12 bit color and 8 - 4 bits for overlays, underlays or control) | 32 bits, double-buffered (24 bit color and 8 bits for overlays, underlays or control) | |
| Auxiliary Frame Buffer | 1 bit, single-buffered (plus 2 bits control) | 8 bits, single-buffered (plus 1 bit monochrome window and 2 bits control) | 8 bits, single-buffered (plus 1 bit monochrome window and 2 bits control) | |
| Z-buffer | N/A | 20 bits | 20 bits | |
| Additional Control Planes | 0 | 4 bits | 4 bits | |
| Total Bits | 35 | 67 | 99 | |
| Shading | Flat, Gouraud, Depth-cueing | Flat, Gouraud, Phong, Depth-cueing | Flat, Gouraud, Phong, Depth-cueing | |

figure 2.4-2
from [Megatek]

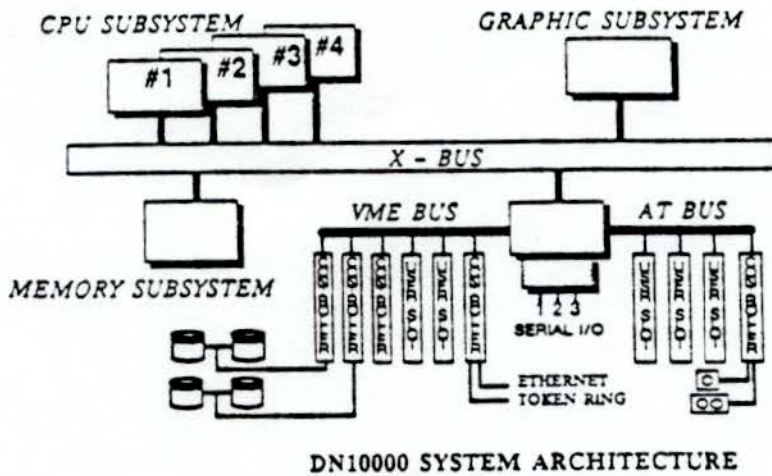
For vectors, the rate is 20 Mpixels/sec, 1,250,000 3D connected vectors per second., independently of the orientation.

The clear-screen operation is performed with block fill mode in RAM. The actual rate depends on whether we are doing a viewport erase or a full screen erase. Viewport erase proceeds at 1.5 ns/pixel, or 666 Mpixels/sec. Full screen erase takes 32 μ s.

III-References for the Megatek Sigma 70

-[Megatek]- Megatek- Sigma 70 Advanced Graphics Workstations, 3/89.

2.5. The APOLLO DN10000VS



In addition, while the operating system is ultimately responsible for the consistency of the virtual address space among the processors, the hardware assists by broadcasting translation invalidations to all CPU's and by observing software visible locking protocol during hardware updates of the mapping table entries.

The DN10000 incorporates a high speed inter-processor bus, the X bus, a synchronous 64 bit bus that operates on 55ns cycles for a peak throughput of 150 MB/sec. The bus relinquishes control every cycle. The memory modules support read queues allowing the bus request to be completed in one cycle. The main memory also supports the prioritization of requests: an instruction cache miss has the highest priority and a stalled processor will be serviced before pending reads.

figure2-5.1
from [Bemis]

I-Overview

The DN10000 is a Reduced Instruction Set, locally cache based, shared virtual memory, multi-processor design (figure 2-5.1). The architecture is known as the Parallel Reduced Instruction Set Multiprocessor (Prism). The current configurations support up to four processors but the architecture is designed to support more processors.

Each CPU has an integer processor, a floating point processor (which contains independent arithmetic logic and multiplier units), a 32x32 integer register file, a 32x64 or 64x32 floating point register file, a 128KB instruction cache, a 64KB data cache, and a Memory Management Unit (figure 2-5.2).

The Prism CPU runs at 18.2Mhz (55ns). The two floating point chips are fully custom ECL and provide an aggregate performance of 36 MFLOPS and 5 MFLOPS on Linpack applications.

The cache subsystem consists of three caches: instruction, data, and translation. The sizes of the instruction and data caches are specified above. The translation cache is two level. The first level is a 32-entry, two set associative cache implemented on an ASIC; the second level is direct mapped and holds 8192 translation entries. A required translation is searched first in the first level, then in the second level, and finally a hardware sequencer examines the memory mapping tables in main memory.

Cache coherency is maintained through a hardware bus watching mechanism that invalidates cache lines whenever a memory store collides with data currently held in either the instruction or data caches.

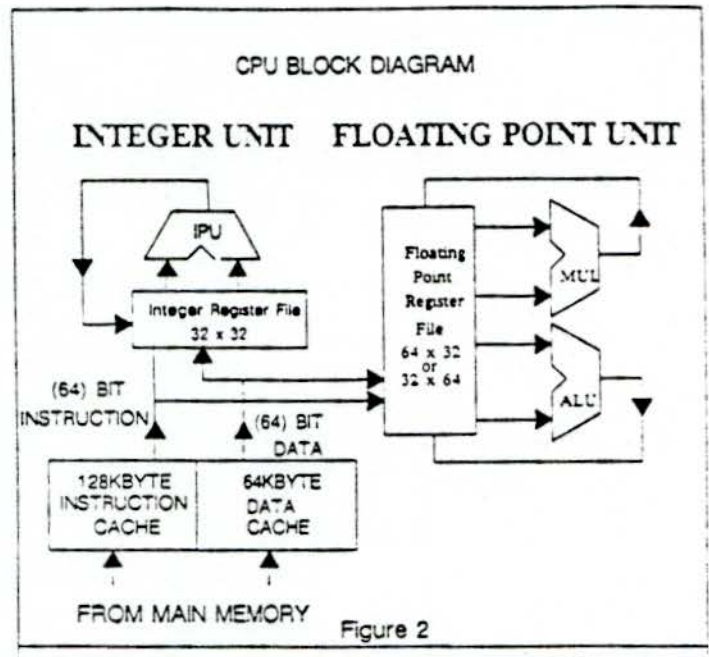


figure 2-5.2
from [Bemis]

The main memory is organized as a hierarchical memory system (figure 2-5.3). Two module controllers handle each one to four memory modules of 8MB to 16MB for a maximum capacity of 128MB in the system. Each module is organized in four independent banks that can be accessed in parallel. Up to four requests can be active simultaneously on each module, that is 32 requests on a 128MB system. The memory is also designed to support read multiples and write multiples. It can deliver 160 MB/sec for both reading and writing.

II-The Visualization System

The name for the Apollo Graphics Superworkstation is the Series 10000VS Visualization System.

The Graphics system is said to be a RISC drawing engine incorporating RISC concepts such as parallel operations, single-cycle pixel draw rates, and no microcode

Some of its main characteristics are:

- programmable 40- or 80-bit plane configurations;
- programmable Z-buffer depth up to 32 bits;
- subpixel addressing, alpha blending;
- quadratic shading (approximation to Phong shading), texture mapping, hardware dithering;
- hardware run length decoding applicable to image file storage;
- output resolution is 1280x1024, 70Hz non-interlaced;
- optional NTSC,PAL, or SECAM output

Image memory is 1536x1024 pixels. The extra 256x1024 memory is used as scratch space.

Bit plane assignment is programmable: each window can configure the planes to suit the individual application. For example a 40-plane system can be configured as 24 bits for a three-component color and 16 bits for the Z-buffer, or as 12 bits double buffered for a three-component color and still 16 bits for Z, or a single 8-bit pseudo-color component and 32 bits for Z.

The graphics system interfaces to the main CPU's via an instruction buffer [Apollo-1] and the X-bus. It contains four separate processors, each computing different portions of the 3D graphics rendering tasks: the address generator, the color processor, the depth processor and the alpha processor (figure 2-5.4).

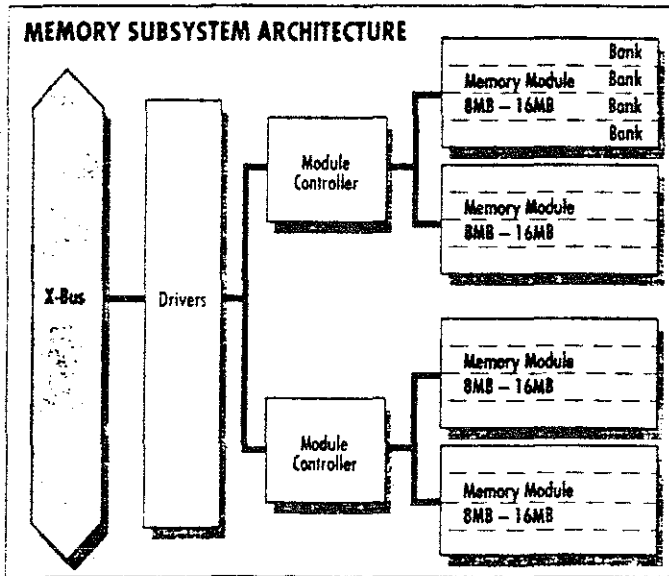


figure 2-5.3 from [Haskin]

Each CPU can execute an instruction stream independently or concurrently with all other CPU's. Only one process executes on one CPU at one time, but no process needs to complete on the same CPU. It can be blocked and return to the queue waiting to resume execution on another CPU. The Operating System maintains a list of processes awaiting execution. It schedules the process with the highest priority to the next available CPU, trying to reschedule processes to originating processors to amortize resident cache data.

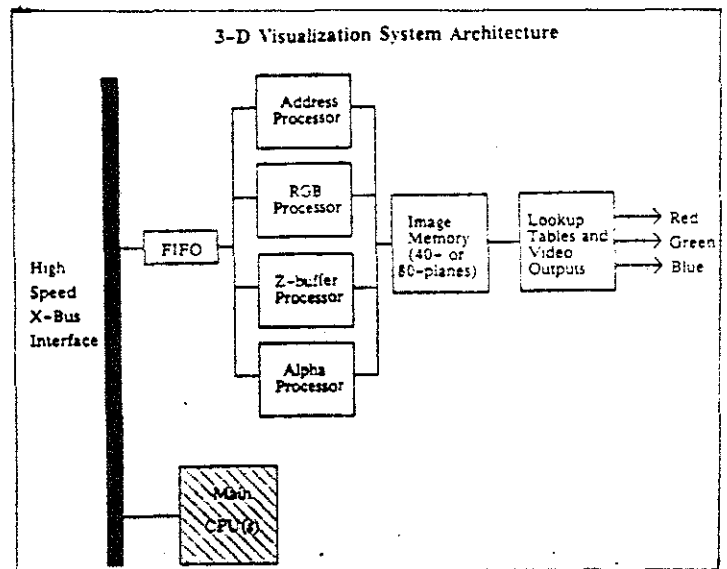


figure 2-5.4 from [Apollo-1]

III -Performances

Each CPU can achieve 1.5 million 3D transforms per second.

In their performance report [Apollo-1], three different levels of graphics performance are explored: machine-level performance, high-level 3D performance and application-level performance.

Machine-level performance numbers indicate the performance the graphics hardware can deliver regardless of how fast the 3D software running on the CPU can drive it. The graphics hardware executes low-level graphics instructions. This means that the 3-D transforms, lighting and clipping operations are not included in a measure of machine-level performance.

High-level performance is the best attainable performance through a high-level 3D graphics interface. For the polygon results shown in figure 2-5.7, they specify 100-pixel triangles, Gouraud shading, an infinite light source and Z-buffering. The performance given is for a one-CPU system which is the limiting factor in that case.

An application-level benchmark renders real images on the screen from 3D graphics metafiles containing polygon and vector data and attribute information. the time to generate the data is not taken into account, only the time to render the image is measured.

Different metafiles have been tested as shown on figure 2-5.5 and 2-5.6. Eggcarton and Head have the same type of data, but Eggcarton is twice as big therefore taking twice as much time to draw for a similar throughput. Turbine has four-sided polygons and that its main difference with Head and the reason why it takes 71% more time to draw, illustrating the impact of of the number of sides on polygon render time. Camera has roughly the same number of polygons as Turbine, but they are a mixture of different shapes, averaging to between four and five sides. Camera also contains many more attribute classes and many more attributes than other metafiles, which is the reason for the performance decrease of roughly 77% over the Turbine.

On the figure 2-5-7, the application -level performance is given for the Head metafile, which is an application that performs very close to the best case high-level rate.

This figure also illustrates that the machine-level performance (108,000) is roughly three times that of the high-level performance with one CPU (32,559). We can expect a more balanced situation with a full equipped four-CPU system.

The figure 2-5.8 gives the Vector machine-level and high-level performances. The high-level is measured for 10-pixel, 1000-segment, transformed 3D polylines.

IV-References for the Apollo DN10000VS

-[Apollo-1]- Apollo Graphics Performance Report, Version 1.0 - Technical Marketing Group, Apollo Computer Inc, March, 1989;

-[Apollo-2]- The Series 10000 Personal Supercomputer - Apollo Computer Inc., 12/88.;

-[Apollo-3]- Series 10000VS Graphics Superworkstation - Apollo Computer Inc., 01/89;

-[Bemis] - Bemis,Paul - High End Systems - The implementation of an industrial strength cache based locally parallel Reduced Instruction Set Computer: The Apollo DN10000 - Apollo documentation, 11/11/88;

-[Haskin] - Haskin, Denis W. - At the speed of light through a prism - Digital Review, December,19,1988.

figure 2-5-5
from [Apollo-1]

| Metafile Characteristics | | | | |
|---------------------------------------|--------|-----------|---------|--------|
| Characteristic | Camera | Eggcarton | Turbine | Head |
| Total structures | 570 | 2 | 1 | 2 |
| Total polygons | 10,150 | 20,000 | 11,310 | 10,170 |
| Total vectors | 45,028 | 60,000 | 45,240 | 30,510 |
| Total attributes | 289 | 2 | 4 | 5 |
| Avg. sides/polygon | 4.43 | 3 | 4 | 3 |
| DN10000VS Refresh Time (milliseconds) | 1,023 | 655 | 554 | 315 |

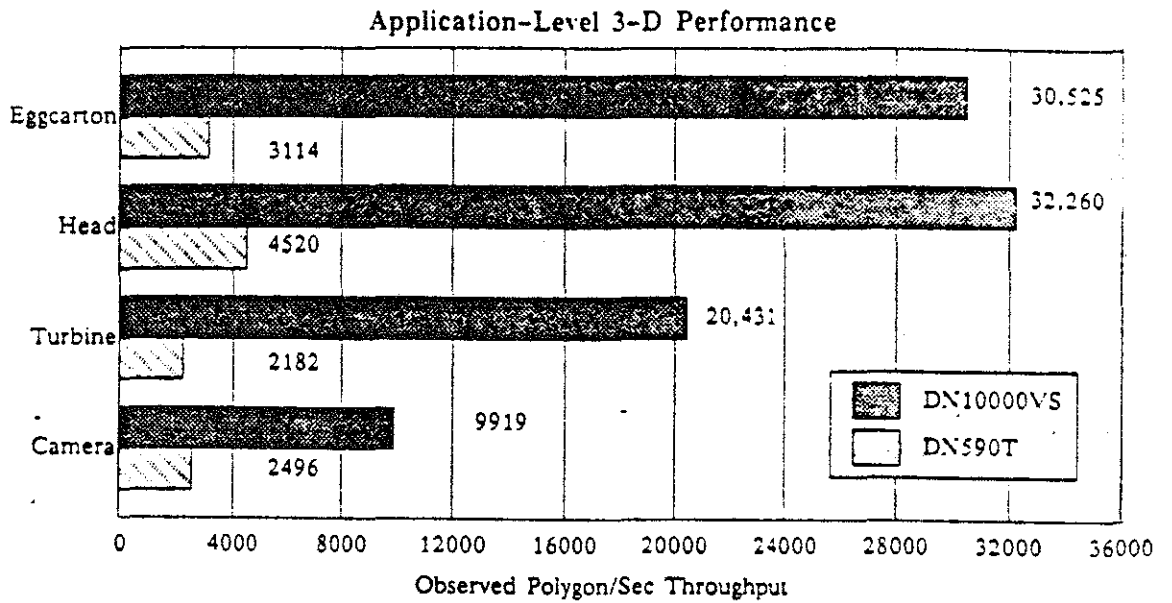


figure 2-5-6 from [Apollo-1]

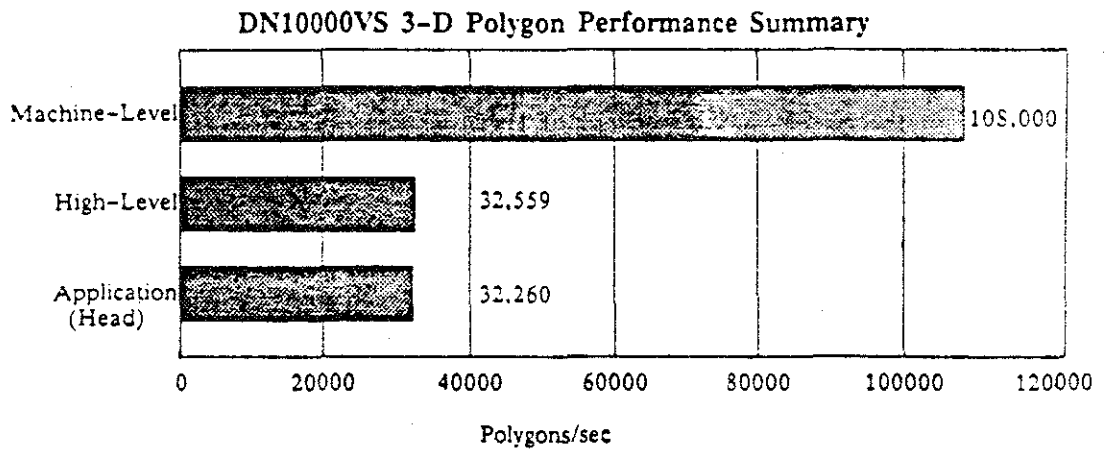


figure 2-5.7 from [Apollo-1]

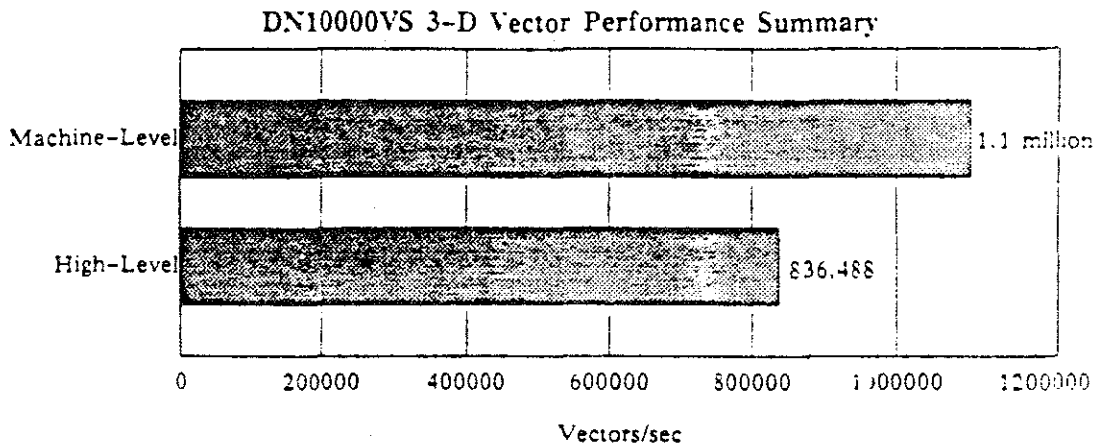


figure 2-5.8 from [Apollo-1]

2.6. The HP 835 TurboSRX

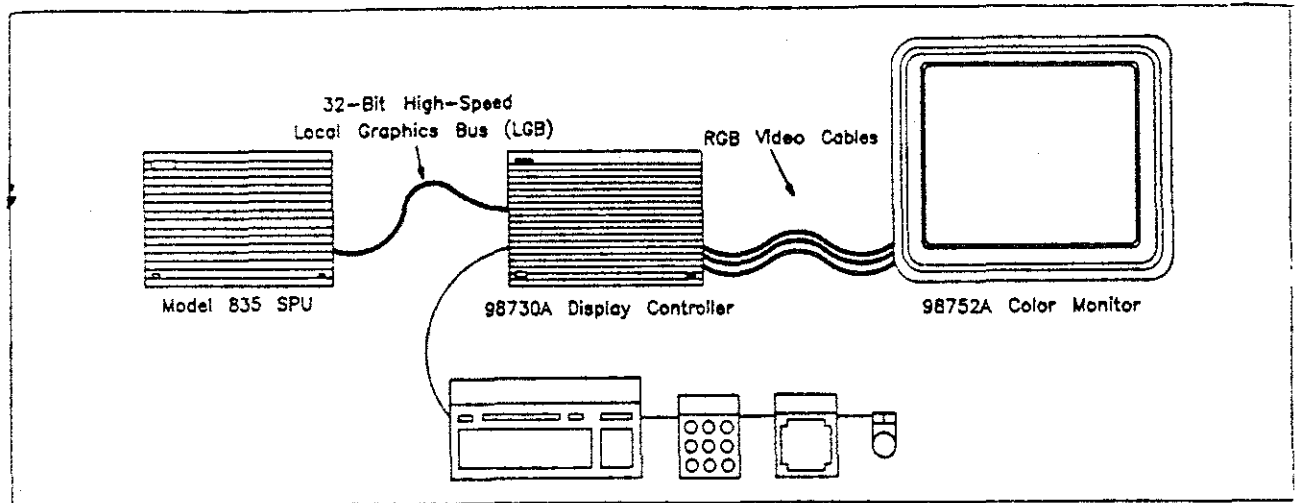


figure 2-6.1 TurboSRX Graphics Subsystem from [HP-2]

I-Overview

The HP 835 TurboSRX Graphics Superworkstation is the high end system in a series of three machines: 825SRX, 835SRX, and 835 TurboSRX. They all run the HP-UX operating system, a superset of AT&T's UNIX System V Interface Definition. They all use the RISC-based HP Precision Architecture for the computing system. SRX is the graphics architecture and it stands for Solids Rendering Accelerator.

The HP Precision Architecture:

- pipelined single-chip CPU, executes 3 instructions simultaneously at 66.7ns instruction cycle,
- 14 VAX MIPS,
- 32 general purpose 32-bit registers in the CPU,
- 48 bits of virtual address space,
- load/store access memory,
- IEEE floating-point coprocessor with 12 data registers (single or double precision) and 4 status/exception registers, running at 2.02MFLOPS.

The System Processing Unit (SPU) (figure 2-6.2) manages the processor, memory and I/O for the entire system. The processor accesses memory and I/O via the Central Bus. The Central Bus runs at 10 MHz, provides a 32-bit data path and supports sustained data transfer rates of 22.3 MB/sec. The Central Bus interfaces with the Channel I/O Buses via the Channel I/O adapters.

The Channel I/O adapters manage DMA transfers between system memory and Channel I/O interfaces with their associated peripherals. The transfer rate is 6.4Mbytes per second per channel.

The Channel I/O Bus on the Model 835 has a 16-bit, 6.4-Mbyte bandwidth. Seven I/O slots are available, two are used by the HP-IB and LAN interfaces.

The graphics subsystem is connected to the SPU through either the Graphics Interface card HP A1017A or the Graphics Animation Interface HP A1047A. The Graphics Interface allows image playback at about 1.74 MB/sec., whereas the Graphics Animation Interface playback speed is about 12 Mb/sec. The Graphics Animation Interface houses, on two different boards, the Midbus interface which plugs into the memory bus, and the Local Graphics Bus which is the link to the TurboSRX system. Two separate graphics subsystems can be supported, using two HP A1047A.

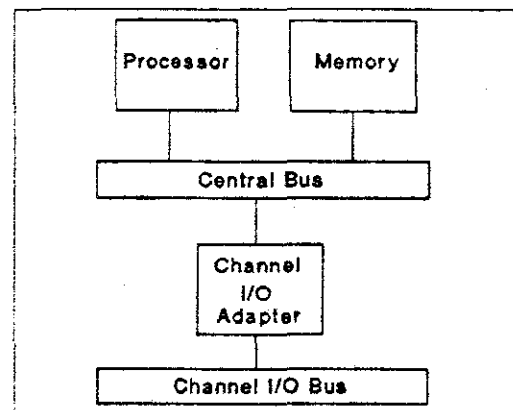


figure 2-6.2 SPU Organization from [HP-2]

The processor module itself contains 8 VLSI chips. Two of these chips are commercially-available VLSI floating point math chips (ALU and MULTIPLIER), and six of them are custom VLSI chips: the CPU, the control unit for the TLB, two control units for the cache, a floating point controller (FPC), and a System Interface Unit (SIU) which monitors communication between the components of the processor and the central bus.

The Model 835 uses 128 Kbytes of high-speed, unified, write-back, two-set associative, parity-checked cache.

The FPC allows floating point operations to overlap with CPU operations, and within the floating point subsystem itself.

Virtual addresses are 48 bits in length. Virtual memory is divided into 65,535 spaces of 4 Gbytes, leading to virtual memory space of about 280,000 Gbytes.

The TLB performs parallel translation of instructions and data addresses.

The Model 835 includes 8 Mbytes of ECC memory expandable in 8 or 16 Mbyte increments to 96 Mbytes. The internal memory word size is 72 bits: 64 data bits and 8 error detection and correction bits.

III- The basic SRX pipelined Graphics Processor

Of the basic image-rendering processes (figure 2-6.3 top), the SRX engine incorporates them all except object modeling and the display-list processing (figure 2-6.3 bottom).

In order to exploit the maximum amount of concurrency inside the SRX architecture, intrablock parallelism and interblock pipelining have been adopted.

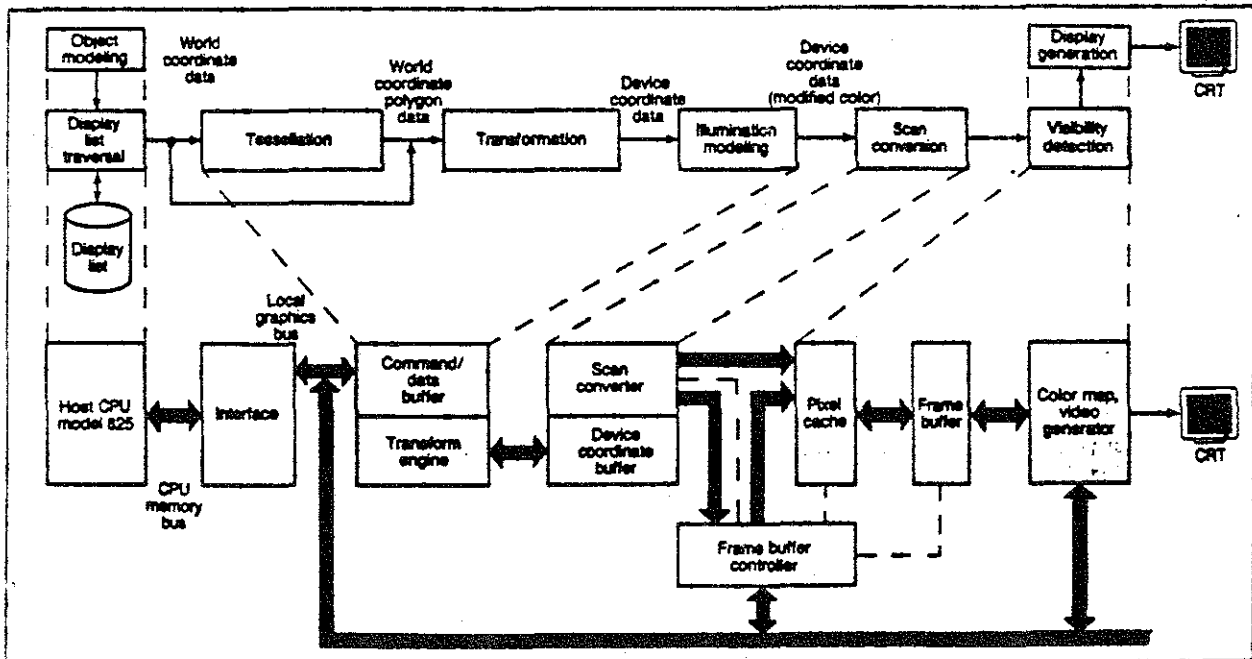


figure 2-6.3
from [Burgoon]

Building the pipeline requires adding dual-ported RAM buffers between blocks: the command/data buffer, the device coordinate buffer, and the pixel cache.

The command/data buffer is halved into two parts that are swapped to ease parallel access both from the host and from the transform engine. Each half is filled in sequence by the host with commands and world coordinate data.

The output of the transform engine is polygon vertex data in screen coordinates, plus color values and low-level graphics commands such as "draw vector" or "fill polygon".

The device coordinate buffer is a dual-ported static RAM bank that holds these data, values and commands for the scan converter.

The scan converter performs a linear, six-axis interpolation of the position and color data for each vertex (x,y,z and R,G,B) to determine the raster location, depth, and color of each pixel belonging to the polygon. The resulting pixel is stored in the pixel cache.

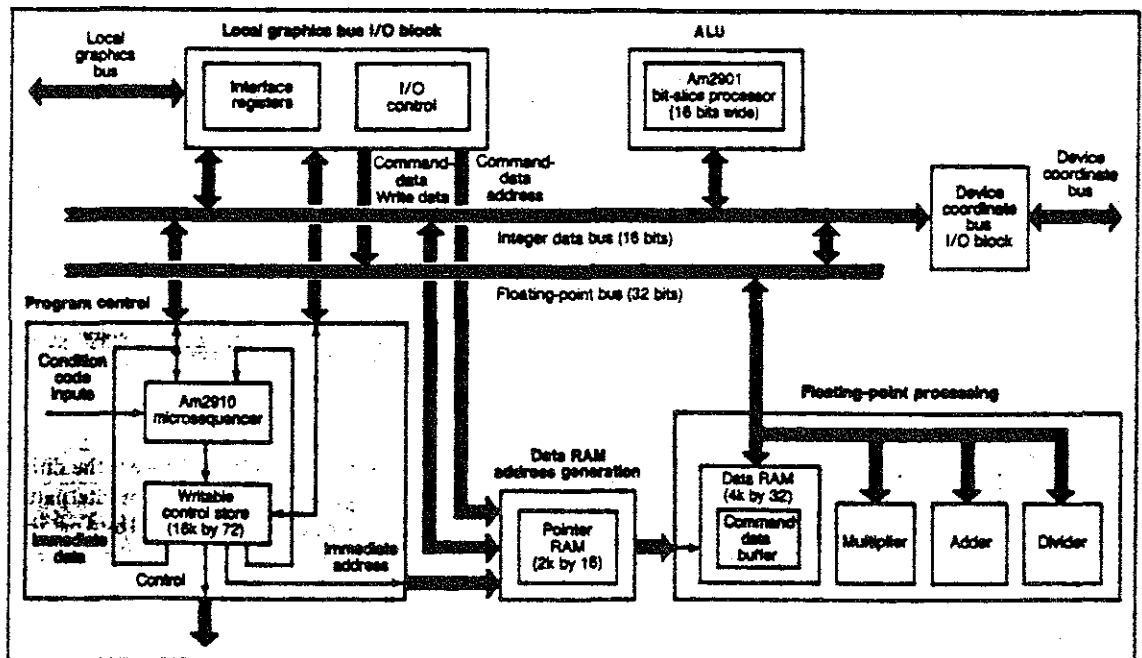
The pixel cache accepts a serial stream of pixels from the scan converter at the rate of 60ns/pixel. The cache stores 16 pixels at a time, in a rectangular array named "tile". When the scan converter attempts to write a pixel outside the boundary of the current tile, the cache writes the 16 pixels in the tile into the frame buffer in a 360ns-write-cycle. The flush operation can overlap the filling of the next tile.

The pixel cache is also used in the process of hidden surface removal, this will be discussed later.

Each block of the pipeline (transform engine, scan converter, and pixel cache) also exploits concurrency and pipelining as much as possible.

The transform engine (figure 2-6.4) is a microprogrammable processor based on an Am2910 microsequencer, with a 72-bit wide control word. The engine includes a 16-bit ALU (four Am2901) and a 32-bit floating point math unit (three HP proprietary custom chips). This floating point unit operates in vector mode.

figure 2-6.4 from [Burgoon]



The SRX's transform engine features a 72-bit-wide control store word that exploits the benefits of parallelism, or concurrency. With the wide word, users control many parts of the processor during one state clock of the microsequencer.

The next block in the rendering pipeline is the scan converter, which includes the device coordinate buffer, a polygon-rendering chip, a dither circuit, and a transparency circuit (figure 2-6.5 a).

The device coordinate buffer specifies data (vertices of lines and polygons), and instructions for the polygon rendering chip.

The polygon-rendering chip (figure 2-6.5 b) interpolates between the vertex data to produce fully shaded lines and polygons. It performs two major functions. First it interpolates between a pair of polygon vertices to determine the screen location, depth, and color of each pixel on the edge represented by the vertex pair. Then it fills in each polygon interior by interpolating along the six axis (x,y,z,R,G,B) between each pair of edge pixels.

Basically the first loop in figure 2-6.5 b (edge-setup-interpolator) performs the first function and the second loop (fill-setup-interpolator) performs the second scan conversion function. A complete description of the process is given in [Burgoon] and [Swanson].

With the two pipelined feedback loops carefully interleaved, the interpolator keeps busy alternating between drawing edge pixels and fill-vectors.

The dither circuit applies a 4x4 filtering pattern, increasing the color resolution at the expense of spatial resolution.

The transparency circuit simulates a transparency by imposing a "screen door" pattern on the image.

The scan converter writes pixels in the pixel cache specifying a pixel's screen location as a tile address and a location within the tile (bit address).

The address calculator, part of the frame-buffer controller, maps the tile addresses to produce addresses suitable for the frame buffer (figure 2-6.6). The frame-buffer controller is able to change the tile organization dynamically to match the type of primitives the system is rendering. They say in [Burgoon] that 16-by-1 tiles are best for area-fill drawing (e.g. fully shaded polygons), whereas 4-by-4 tiles offer the best performance for randomly oriented vectors.

The pixel-color data and bit address produced by the scan converter enter the cache through the pixel port (figure 2-6.6). The cache's Z port handles the depth data, including a comparison with previously stored Z values, using the Zout port.

Figure 2-6.7 shows the data paths inside the pixel cache. The pixel-color data is loaded into the data cache. The contents of the data cache are transferred to the source

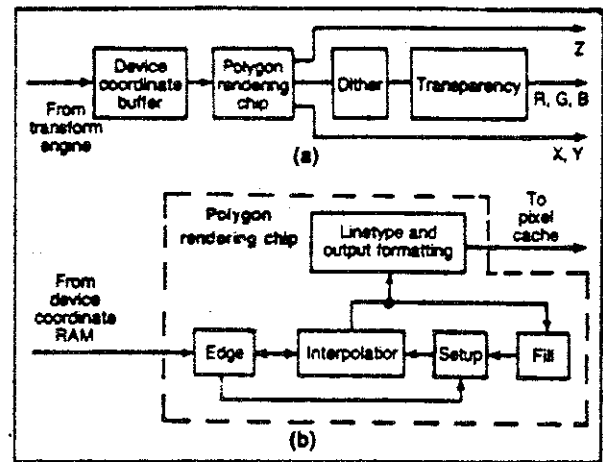


figure 2-6.5 from [Burgoon]

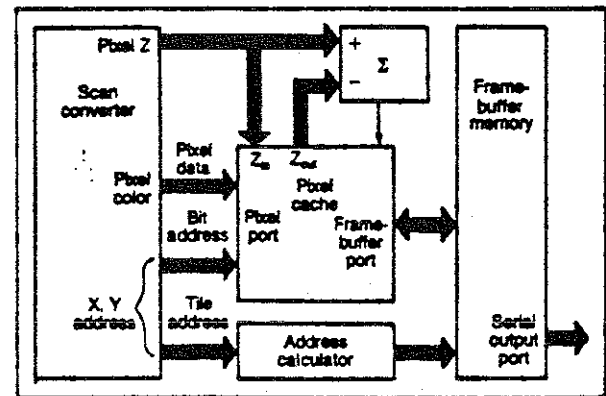


figure 2-6.6 from [Burgoon]

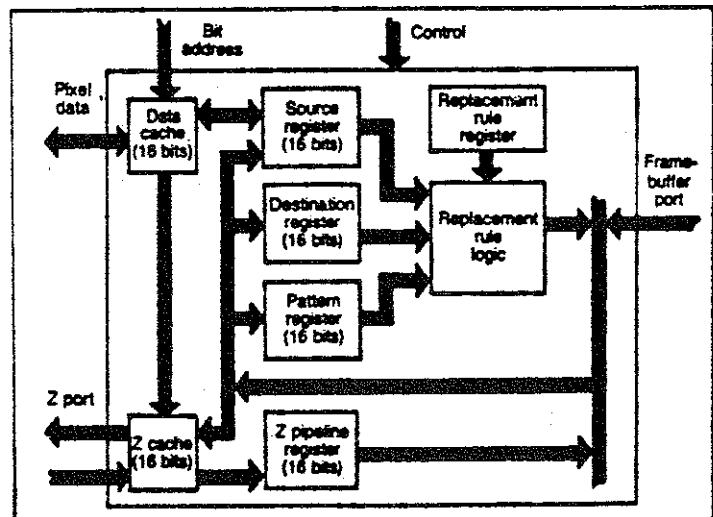


figure 2-6.7 from [Burgoon]

register when the scan converter signals the frame-buffer controller that it wants to write a pixel outside the current tile. While new data is loaded into the data cache, data in the source register mixes with the data in the destination and pattern registers according to a Boolean equation specified by a code in the replacement-rule register. This modified pixel data is then written to the frame buffer under the control of the frame-buffer controller. The destination register contains the previously stored pixel data in the current tile. These data have been fetched from the frame buffer while the data cache was being loaded by the scan converter.

Operation on the Z path is similar (figure 2-6.7). The Z cache contains copies of the depth values in the frame buffer locations corresponding to the current tile, which is compared with the depth data for the current tile. Upon reception of a pixel outside the current tile, the contents of the Z cache transfer to the Z-pipeline register, and then to the Z buffer, part of the frame buffer.

III The TurboSRX Graphics Subsystem

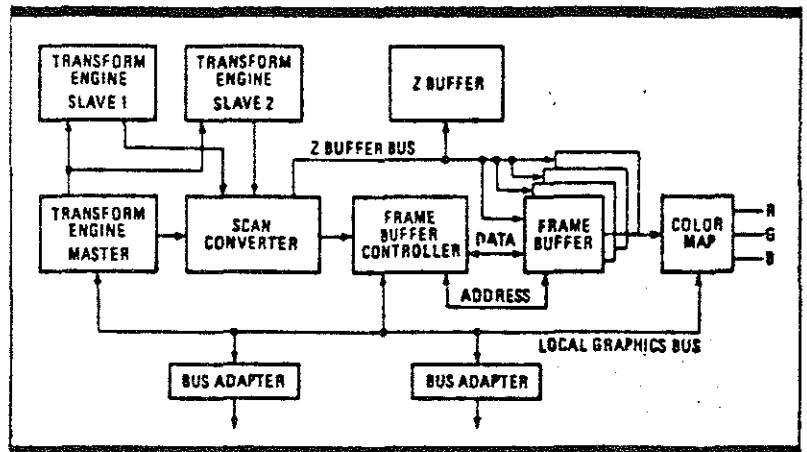
The design of the TurboSRX Graphics Accelerator is based upon the SRX architecture. To achieve three to ten times the SRX performance, the SRX Graphics Accelerator has been integrated into a custom VLSI microprogrammable CPU. This integration allows the graphics subsystem to be configured with up to three identical accelerators running in parallel. Additional enhancements include high-speed Z-buffer and improvements in the communication links.

Figure 2-6.8 shows an overview of the TurboSRX architecture. It contains three transform engines. The master transform engine receives data and commands from the host. If busy, it passes them to the slaves both of which operate in parallel. These engines also perform all the computation for B-splines, light source, and color. They are 177,000-transistor HP custom chips called Treis (transform engine in silicon) implemented in HP's n-MOS III process.

The scan converter operate in the same way as described earlier.

While the scan converter routes depth data to the Z buffer, the frame buffer controller routes the pixel-color data from the scan converter to the correct frame buffer bank, using data from the Z buffer to conditionally rewrite the color values in the pixel locations.

The pixel cache seems to be integrated into the frame buffer controller.



HP's new TurboSRX graphics processor contains the graphics transform engine, scan converter, Z buffer, frame buffer controller, frame buffer, and color map for fast solids modeling.

figure 2-6.8
from [McLeod]

IV- Some TurboSRX Graphics Features

(see [HP-2] for more details)

- Graphics interactivity:
 - Sixth-order Non-Uniform Rational B Splines (NURBS) with trimming,
 - polygon mesh primitives,
 - hardware cursors,
 - image blending,
 - pixel pan and zoom,
- Photorealism:
 - Radiosity,
 - Ray-Tracing,
 - advanced hardware lighting models,
 - Gouraud shading and Phong lighting,
- Graphics animation (using the optional interface), with DMA transfers from main memory to the frame buffer at 12 MB/sec. on the Model 825 and 13.1 MB/sec. on the Model 835,
- Starbase Graphics Library, and Starbase PHIGS display list structure, both integrating Radiosity and Ray-Tracing

V. Performances

The performances are summarized in the table figure2-6.9

| | (3D) 825SRX | (3D) 835SRX | (3D) 835 TurboSRX |
|-----------------------------------|--|--|--|
| CPU | | | |
| Type | HP Precision Architecture | HP Precision Architecture | HP Precision Architecture |
| Word Size | 32 bits | 32 bits | 32 bits |
| Virtual Memory Address Space | 48 bits | 48 bits | 48 bits |
| Physical Address Space | 29 bits (512 Mbytes) | 29 bits (512 Mbytes) | 29 bits (512 Mbytes) |
| Cache Memory | 16 Kbytes | 128 Kbytes | 128 Kbytes |
| Instruction Cycle Time | 80 nsec | 66.7 nsec | 66.7 nsec |
| Average CPU Throughput | 16,672 Dhrystones | 23,430 Dhrystones | 23,430 Dhrystones |
| Floating Point Coprocessor | | | |
| Floating Point Performance | .65 MFLOPS (double precision Linpacks) | 2 MFLOPS (double precision Linpacks) | 2 MFLOPS (double precision Linpacks) |
| Floating Point Format | 754 ANS/IEEE 1985 Standard | 754-ANS/IEEE 1985 Standard | 754-ANS/IEEE 1985 Standard |
| Capacity | | | |
| Memory | 8-96 MB ECC RAM | 8-96 MB ECC RAM | 8-96 MB ECC RAM |
| I/O Slots | Seven | Seven | Seven |
| Power | | | |
| Line | 48-66 Hz | 48-66 Hz | 48-66 Hz |
| Current | 9.5A @ 100 Vac 8.0A @ 120 Vac 5.3A @ 240 Vac | 9.5A @ 100 Vac 8.0A @ 120 Vac 5.3A @ 240 Vac | 9.5A @ 100 Vac 8.0A @ 120 Vac 5.3A @ 240 Vac |
| Power Consumption | 600 Watts max. | 600 Watts max. | 600 Watts max. |
| Size And Weight Of SPU | | | |
| Size | 234 mm x 325 mm x 500 mm 9.2" x 12.8" x 19.7" | 234 mm x 325 mm x 500 mm 9.2" x 12.8" x 19.7" | 234 mm x 325 mm x 500 mm 9.2" x 12.8" x 19.7" |
| Weight | 23 kg (~50 lb.) | 23 kg (~50 lb.) | 23 kg (~50 lb.) |
| GRAPHICS SUBSYSTEM: | | | |
| Monitor | | | |
| Type | Color | Color | Color |
| Resolution | 1280 x 1024 | 1280 x 1024 | 1280 x 1024 |
| Size | 19 inch diagonal | 19 inch diagonal | 19 inch diagonal |
| Frame Buffer | | | |
| Size | 2048 x 1024/plane | 2048 x 1024/plane | 2048 x 1024/plane |
| Planes | | | |
| Standard | 8 | 8 | 8 |
| Optional | 24 | 24 | 24 |
| Overlay Planes | 4 | 4 | 4 |
| Z-Buffer | Hardware, 16-bit full or strip | Hardware, 16-bit full or strip | Hardware, 16-bit full |
| Displayable Colors | | | |
| Standard | 256 | 256 | 256 |
| Optional | 16.7 million | 16.7 million | 16.7 million |
| Color Palette | 16.7 million | 16.7 million | 16.7 million |
| Performance For 3D Brief | | | |
| 3D Vectors/sec. | 67,000‡ | 67,000‡ | 240,000‡ |
| 3D Triangles/sec. | 7,500§ | 7,500§ | 38,000§ |
| Transforms | 180,000 coord./sec. | 180,000 coord./sec. | 900,000 coord./sec. |
| Scan Conversion | 16 million pixels/sec. | 16 million pixels/sec. | 16 million pixels/sec. |

‡ 3-D vector performance is measured for transformed and clipped vectors 10 pixels in length. The lines are constant color with no hidden-surface removal.

§ 3-D polygon performance is measured with one light source and shading. Polygons are 30 pixels on a side.

VI-References

for HP 9000 Series 800
Graphics Superworkstations

-[Burgoon] - Burgoon, Dave - Pipelined graphics engine speeds 3-D image control - Electronic Design, July 23, 1987;

-[McLeod] - McLeod, Jonah - HP delivers photo realism on an interactive system - Electronics, March, 17, 1988;

-[Swanson] - Swanson, Roger W., and Thayer, Larry J. - A fast shaded-polygon renderer - in Proceedings SIGGRAPH'86, Computer Graphics, Vol. 20, number 4, August 18-22, 1986;

-[HP-1] - HP 9000 Series 800 Models 825SRX, 835SRX & 835 TurboSRX Graphics Superworkstations - Hewlett-Packard, 3/88;

-[HP-2] - HP 9000 Series 800 Model 835 TurboSRX Hardware Technical Data, 8/88;

-[HP-3] - HP 9000 Series 8000 Models 825CHX and 835CHX Hardware Technical Data - Hewlett-Packard, 4/88;

3. PART B:

Current available 100K tr.sec. graphics machines

3.1 Introduction: General overview of the three machines

Graphics Performances: (commercial figures)

(true color, Gouraud shaded, Z-buffered,100-pixel triangles)

-*Ardent Titan*: (4 CPU) 100K tr./sec.

-*SGI GTX POWER Series*: 115K tr./sec.

-*Stellar GS1000*: 100K tr./sec.

Linpack 100x100 Floating point performances: (respectively): 6-24 Mflops, 4-16Mflops, 8.6Mflops

Parallel Processing approach:

-Shared Memory Multiprocessor systems,

-High Memory bandwidth: 256MB/sec.,64MB/sec,320MB/sec.

-Homogeneous Concurrency applied to pixel rendering: basically a divide-by-pixel approach.

Operating System and Graphics Interface:

Ardent Titan:

-Titan UNIX, compatible with AT&T UNIX System V.3 and Berkeley 4.3 UNIX.

-Doré Graphics Interface, PHIGS+, CGI and X+ (extension of X Window System,v11).

SGI GTX or POWER Series:

-IRIX Operating System based on AT&T UNIX System V, release 3.1 and supporting many features of BSD 4.3.

-IRIS Graphics Library, 4Sight Windowing System, X-Window, NeWS.

Stellar GS1000:

-STELLIX Operating System based on AT&T UNIX System V, release 3.0

-PHIGS+ and XFDI,Stellar extension of X Window System v11.

3.2 ARDENT TITAN Graphics Supercomputer

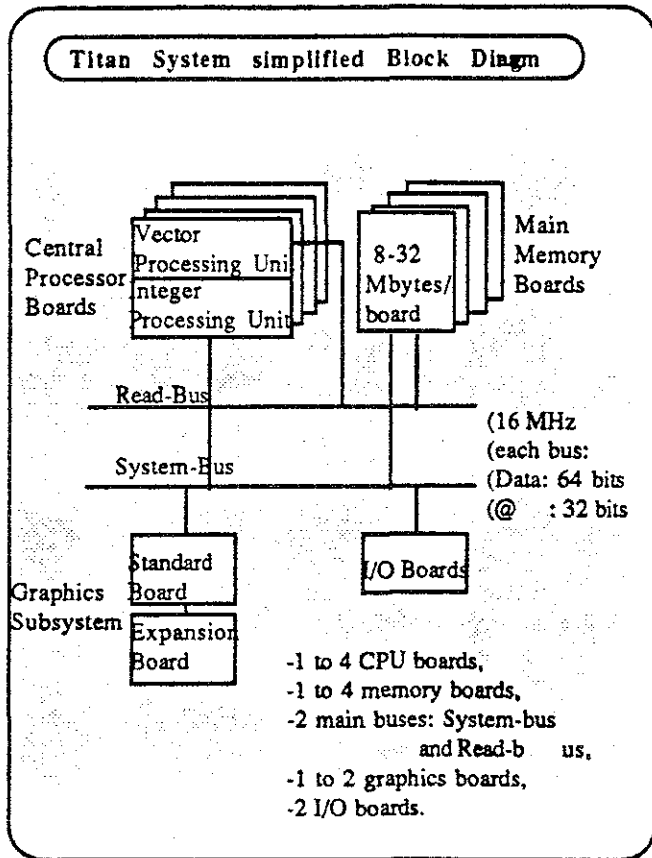


figure 3.2-1

I-Computer Graphics Performances

(see Titan Technical Overview p. 90)

3D transform peak rate per processor:

- 275K vertices/sec.;
- 21K triangles/sec. (25K in a mesh): independent Gouraud shaded ;

Rendering peak rates:

- 100K triangles/sec.:100-pixels, Gouraud shading, true color expansion mode:
- 50M pixels/sec: large full screen triangles, 8-bit mode base graphics, and 8-bit or 24-bit mode expansion graphics, Z-buffered, Gouraud shading;
- 11.6M pixels/sec: 100-pixels triangles, (same conditions as above);
- 3.9M pixels/sec.: 100-pixels triangles, 24-bit mode

base graphics, Z-buffered, Gouraud shading;

Interpolated Phong shading is supported by the processors at the rate of 20K Triangles/sec. for 3D transform, clip,perspective divide, and computation of triangle color value.

Some of the figures seem to be confirmed by Matt Fitzgibbon's experiences. For example, the church (15K triangles) is drawn at about 3 to 4 frames per second (about 50K triangles per second), on a 2-processor configuration, matching the advertized 25K triangles/sec. per processor (in a mesh). on the other hand, when a lot of clipping against vertical and horizontal planes are required, then the drawing rate falls by a factor of up to 2 or 3, emphasizing the transformation rate limitation of the configuration.

The true color 24-bit mode expansion graphics is based on a parallel rendering configuration for the R G and B leading to the same peak rendering rate as for the basic pseudo color 8-bit mode.

II-Computational Subsystem

A tightly coupled symmetrical shared memory multiprocessor (up to four processors): see figures 2.2-1, 2.2-2 & 2.2-3.

The CPU board:

- 16 MIPS 32-bit RISC R2000 Integer Unit,
- proprietary 16Mflops Vector Floating Point Unit (VPU),
- 16KB instruction cache,
- 16KB write-through data cache,
- a bus watcher to maintain cache coherence,
- a read buffer, input interface from system bus and memory,
- a write buffer, four write deep, output interface bus and memory,
- an External Translation Lookaside Buffer for virtual memory access from VPU and for semaphore access.

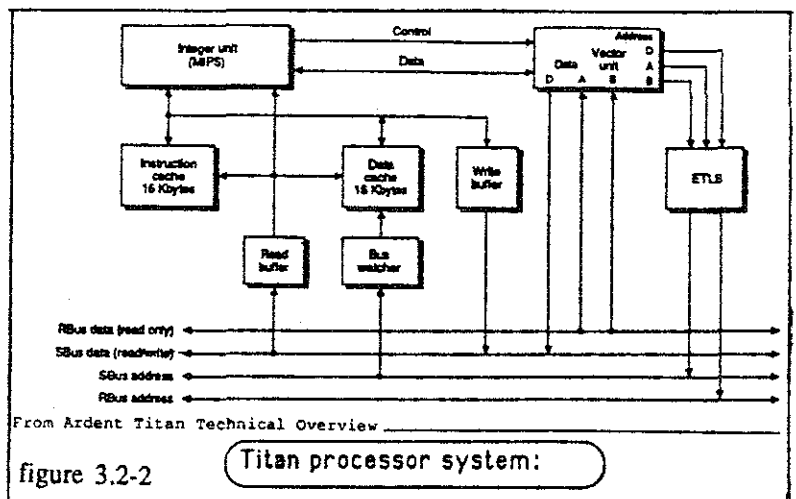
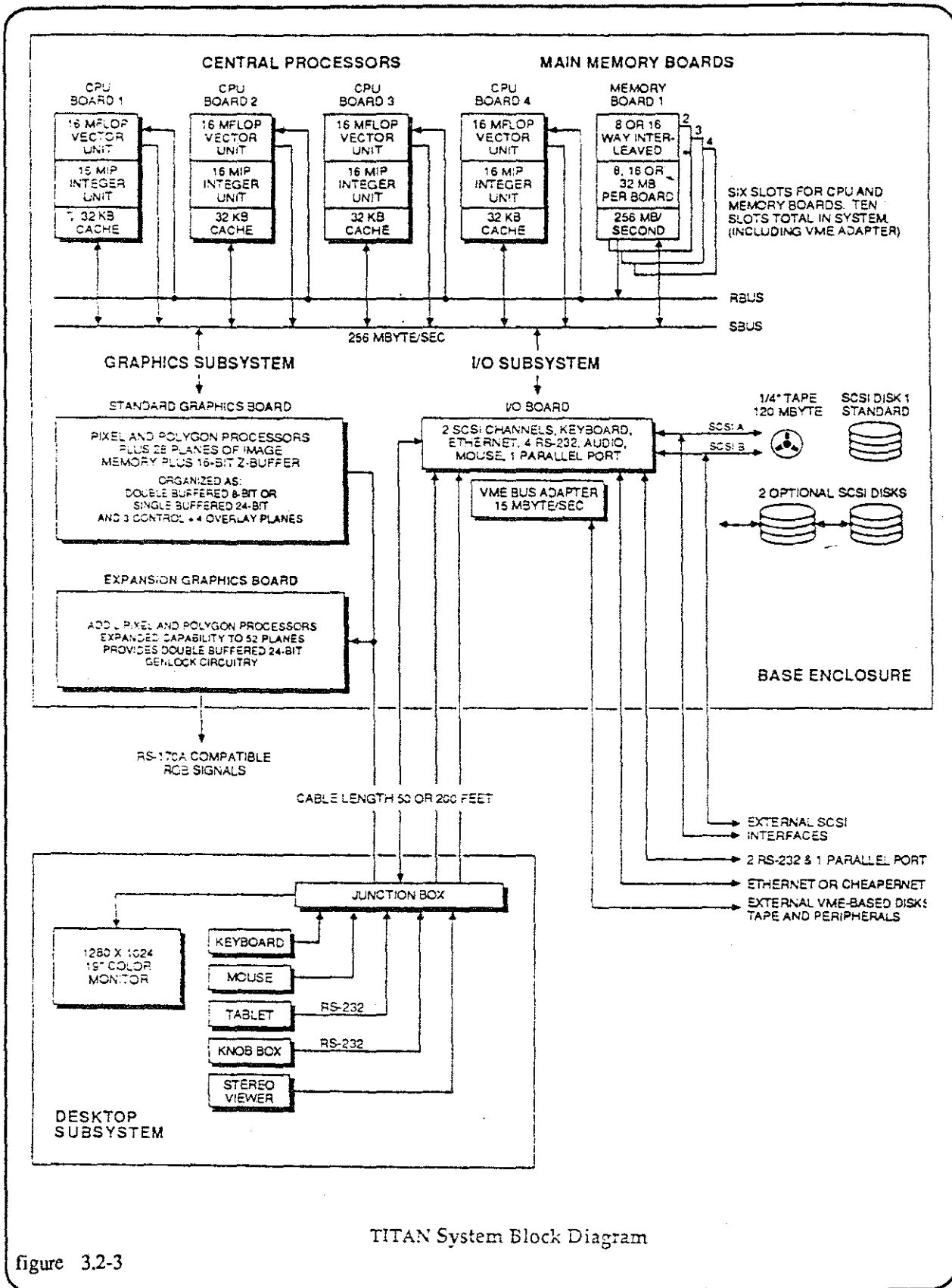


figure 3.2-2



TITAN System Block Diagram

figure 3.2-3

From Ardent Titan Technical Overview

Proprietary Bus-Design:

The Titan bus has four logical pieces:

- an arbitration bus,
- a control bus (clocks, interrupts, reset lines...),
- two data transfer buses: the R-bus (Read Bus)

dedicated to read operations from the Vector Unit, and the S-bus (System Bus) for all read and write operations between the Integer Processors, I/O subsystem and Graphics Subsystem.

The R-bus and S-bus both handle 32-bit addresses and 64-bit data bits. They run at 16 Mhz and obey a synchronous, split-transaction bus protocol. There aggregate bandwidth is 256 MB/sec.

All transactions are acknowledged via handshake signals. Write operations are performed on the S-bus: the address is output first and then the data are presented after a fixed delay. Read operations take place on both R-bus and S-bus: read requests are tagged with a unique requester ID so that the data can be routed back to the requester. The response time may be variable; it is bounded by a time-out mechanism.

All devices that use a bus must arbitrate for a free slot on that bus. A request may not be placed on a bus unless the target resource is free. Resource status is denoted by a backplane signal. So no bus cycle is wasted on a request that cannot be honored immediately. The bus arbitration mechanism arbitrates local requests (for example write-buffer and Vector Unit to S-bus) and global requests (CPUs to Memory).

Multiprocessing synchronization is supported by simple/counting semaphores and doorbells.

Memory Subsystem:

The Memory Subsystem delivers data to both Rbus and Sbus at a sustained throughput of 256M bytes/sec.: 64 bits per bus per cycle at 16 Mhz.

Memory space ranges from 8 to 512 M bytes over a max. of 4 boards. Each board is 8-way interleaved, each of the interleaves can be accessed independently

Memory boards operate in pairs, providing 16-way interleaving.

Any request on the bus is serviced by one interleave. Each interleave is physically 32 bits wide.

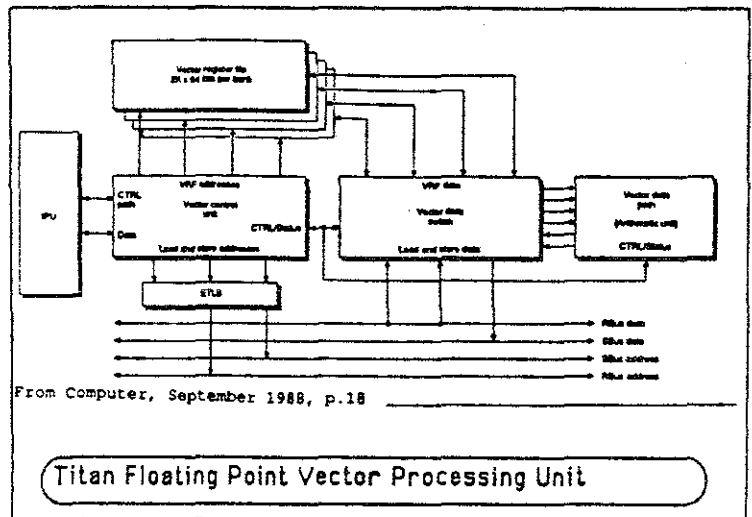
64-bit words are accessed in two clock cycles and the interleave must be addressed on double-word boundaries. Data returned on the bus in each cycle of a sustained sequence, consists of the high-order 32-bit word of read

request i, together with the low-order 32-bit word of read request i+1.

Memory is protected by an ECC that detects and corrects single-errors, and detects double-errors. The corrected results are rewritten in memory either on a scrub transaction or immediately in case of a read-modify-write cycle.

The memory subsystem can have multiple requests pending simultaneously from one or more requesters. Data will be returned in the same order the requests were received. Read responses are tagged with the ID of the initiator of the read request.

Floating Point Vector Processing Unit



...figure 3.2-4

The Vector Processing Unit (VPU), under instructions from the IPU, executes all floating point vector and scalar operations, and vector integer arithmetic. The VPU also acts as the geometry engine for graphics processing.

The major components of the VPU are: the Vector Register File, the Vector Data Path (arithmetic units), the Vector Control Unit, the Vector Data Switch and the Control Path to IPU.

The Vector Register File can store up to 8K double-precision words organized in 4 banks.

The Vector Data Path is based on the WEITEK 2264/5 chip set. It operates on single and double-precision floating point vectors, integer vectors and floating point

scalars and contains three independent arithmetic functional units, ALU, multiplier, divider with a peak aggregate performance of 16MFLOPS;

The Vector Control Unit contains three vector address generators that control three independent memory pipes (two load pipes each running at 8 M accesses/sec., and one store pipe, 16 M accesses per sec.); a scoreboard keeping track of the operations to ensure the proper sequencing of the address generators and the arithmetic unit; one address path to each of the four vector register file banks; control paths to the vector data switch, the vector data path and the IPU.

The Vector Data Switch is a cross-bar system handling the three memory pipes, the four pipes to the vector register file, the data paths to/from the arithmetic units and the control paths.

The VPU is a slave to the Integer Processing Unit, but the IPU, however does not control the cycle by cycle operation of the VPU. Instead, the IPU controls how the VPU operates by setting internal VPU registers and sending instructions to the VPU. The VPU maintains an instruction write buffer

The Vector Register File:

the Vector Register File must be at least six-ported:

- one for each of the two loads,
- two ports for each of the two operands,
- one port for the operation result,
- one port for the store.

It is a small high-speed static RAM organized as four independent banks.

Each bank is 2048 x 8-byte cells, 16 M accesses per sec.

The Operating System divides the Vector Register File into 8 frames of 256 cells yielding to 8 user sets of vector registers, each set counting 32 registers of 32 cells.

III-Data Flow in Graphics applications

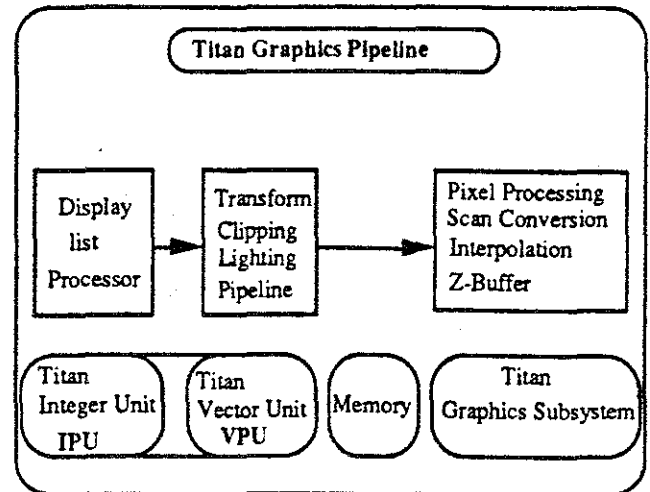


figure 3.2-5

-The Integer Processor Unit (IPU) executes the application program: scene database construction, modification and traversal.

-The Vector Processing Unit, under the control of the IPU, does coordinate transformations, clipping, shading calculations, projection and perspective divide, device coordinate scaling.

-The IPU generates commands for the rasterizers (Graphics Subsystem) and stores them in memory. Then it instructs the Graphics Subsystem to read the command stream using DMA access.

-The rasterizers generate the graphics images, operating independently of the IPU and the FPU.

IV-Dedicated Hardware for low-level graphics rendering

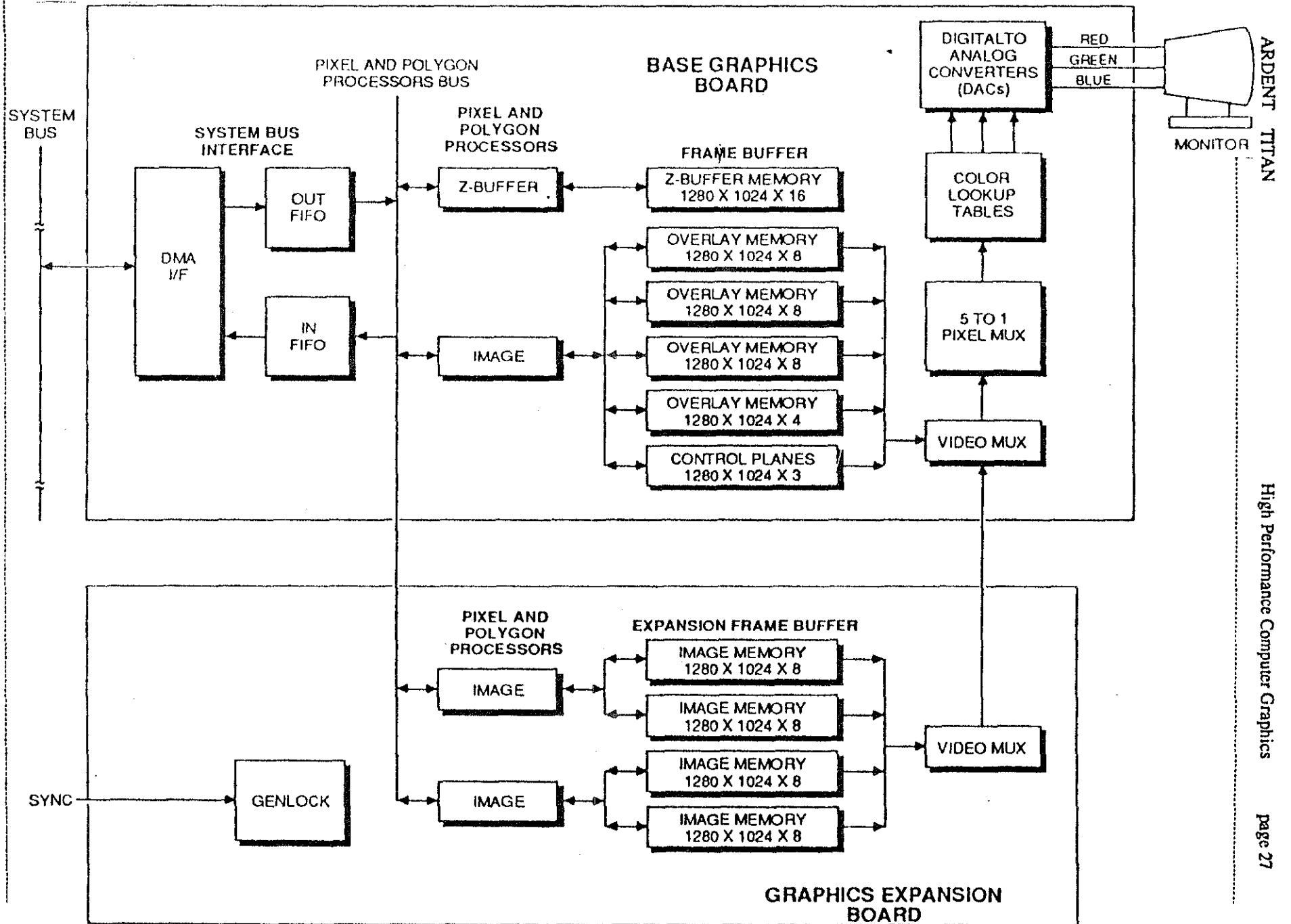
see figure 3.2-6 & 3.2-7

Connected to the System Bus, the Graphics Hardware reads the Data Commands from main memory using DMA access at 20 Mbytes/sec and performs the rendering into the frame buffer memory,

The DMA interface can also transfer data from the frame buffer to main memory at 8.5Mbytes /sec.

The Graphics Subsystem comprises a Base Graphics Board and an optional Graphics Expansion Board, both equipped with two rasterizers that generates scan lines into 4x5 interleaved video memory banks.

Figure 2.2-6 a block diagram of the graphics subsystem.



Roger Brusq UNC-CH July, 15 1989

From Ardent Titan Technical Overview

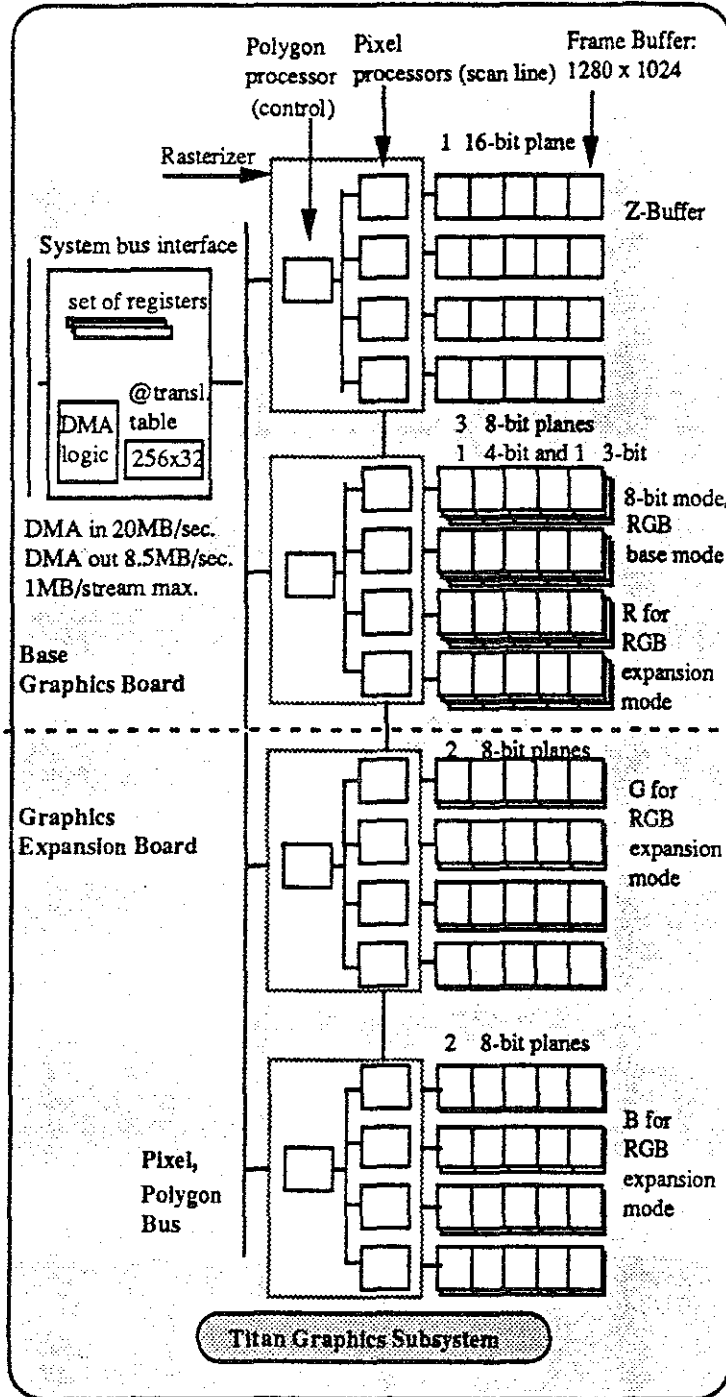


figure 3.2-7

On the Base Graphics Board, one of the rasterizers handles the 16-bit Z-buffer memory planes, the second rasterizer handles three 8-bit color planes, a 4-bit overlay plane and a 3-bit control plane.

This rasterizer generates the 8-bit mode base graphics at the peak rate of 50M pixels/sec. and the 24-bit mode base graphics at the peak rate of 16.7M pixels/sec. (that is the 1/3 of the 8-bit peak rate).

On the Expansion Graphics Board, each rasterizer handles two 8-bit color planes.

The 24-bit mode expansion graphics is generated as follows:

- the R color is generated on the base board, by the color rasterizer,
- the G color is generated on the expansion board, by the first rasterizer,
- the B color is generated on the expansion board, by the second rasterizer.

Each of these rasterizers support double-buffering mode for its dedicated color.

Each rasterizer comprises a polygon-processor (or control processor) that handles four pixel-processors (or scan-line processors). The polygon-processors are connected through the polygon bus and the pixel-processors seem to be directly connected to each other to perform the Z-buffer algorithm. A write-enable signal comes from the pixel-processors in the Z rasterizer and goes down to all the other pixel-processors.

The Rasterizers perform various pixel-level algorithms:

- full screen clearing @ 106Mpixels/sec.;
- rectangular area fill @ 60Mpixels/sec.(20x20 area)
- pixel move @ 12Mpixels/sec.(20x20, 24-bit, expansion)

- drawing control:
16-bit mask to display dotted and dashed lines,
8x8 mask to control rectangle and triangle drawings,

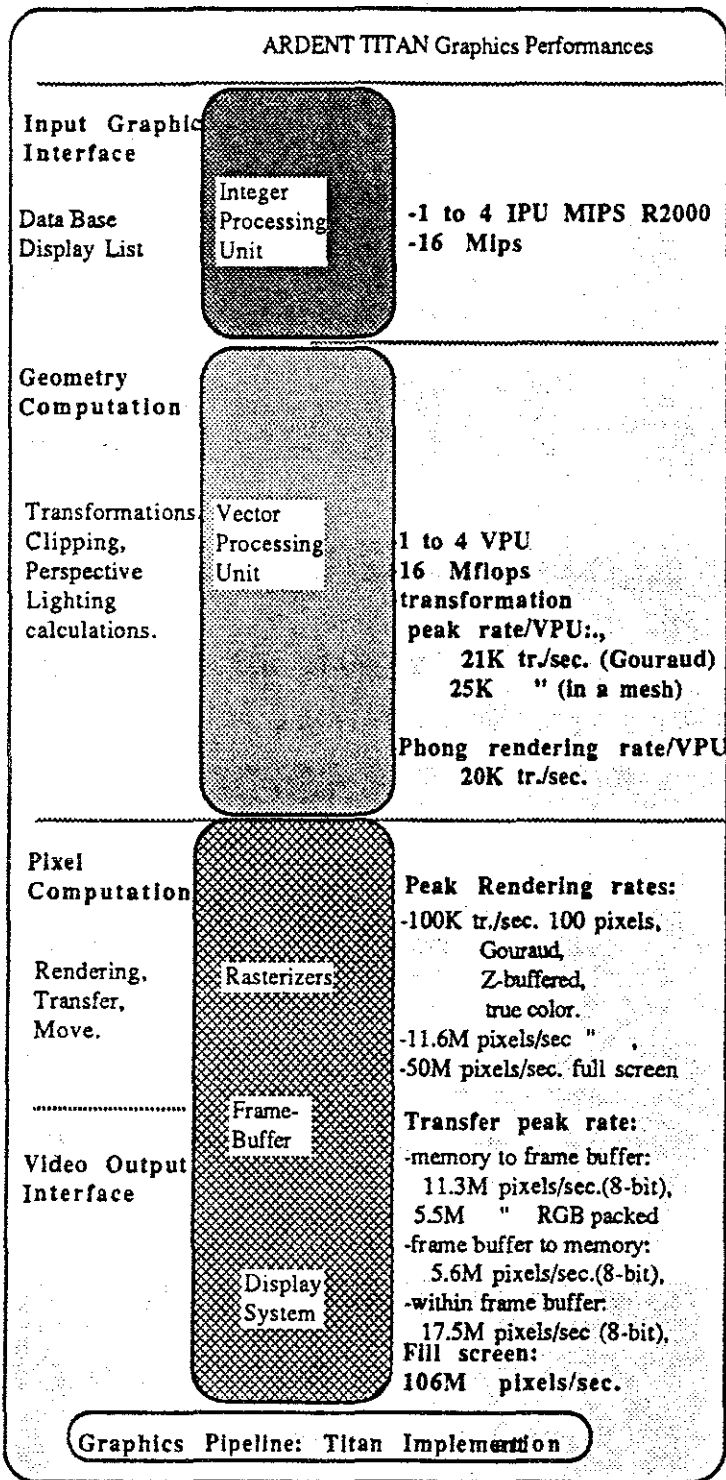
- pixel operations: or, xor, nor, and, nand, not, add, subtract, equate, clear.

- pixel move within frame buffer at up to 17 Mpixels/sec.

- pixel transfer:
memory to frame buffer at up to 11.2 Mbytes/sec.
frame buffer to memory at up to 5.6 Mbytes/sec.

- rendering:
Gouraud shaded, Z-buffered triangles at up to 50 Mpixels/sec.
shaded Z-buffered vectors at up to 10 Mpixels/sec.

V- Structure and performances overview



VI- References for Ardent Titan

- Diede, T. et al. - The Titan Graphics Supercomputer Architecture - Computer, September 1988, pp 13-30.
- McLeod, J. - Ardent launches first "Supercomputer on a desk" - Electronics, March 3, 1988.
- Sanguinetti, J. - Micro-analysis of the Titan's Operation Pipe - in Proc. 1988 Int'l Conf. Supercomputing, November 1988, pp 190-196.
- Till, J. - Single User Supercomputer is Interactive Graphics Powerhouse - Electronics Design, March 31, 1988.
- Ardent Computer:
 - The Titan Graphics Supercomputer, a Systeem Overview.
 - Doré Graphics: an Interactive Visualization Toolkit.

figure 3.2-8

3.3 SILICON GRAPHICS IRIS POWER Series

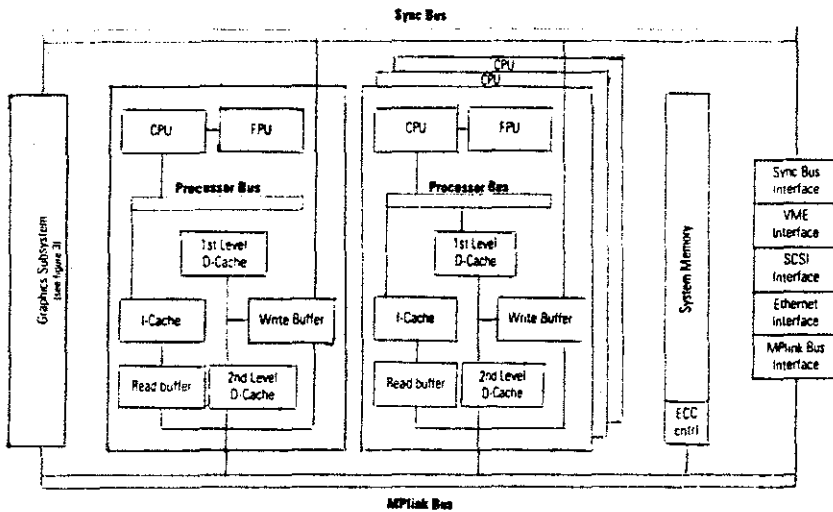


figure 3.3-1 : GTX System Diagram from IRIS Power Series Technical Report

I-Computer Graphics Performances:
(from [Akeley88] p.246)

- 101K quadrilaterals /sec. (100 pixels, arbitrary rotated, lighted, Z-buffered.
- 137K Triangles /sec. (50 pixels, arbitrary strip direction, lighted, Z-buffered.
- 394K lines /sec. (10 pixels, arbitrary directed, depthcued, Z-buffered.
- 210K antialiased lines /sec. (10 pixels, arbitrary directed, Z-buffered.
- 8.3 milliseconds full screen clear.

II-Computational Subsystem

(5 new proprietary VLSI parts around MIPS CPU and FPU)

A tightly coupled symmetric shared memory multiprocessor.

Configurations (one of the following):

- Two 16.7Mhz RISC processors and two floating point coprocessors : R2000 and R2010,
- Two 25Mhz RISC processors and two floating point coprocessors: R3000 and R3010,
- Four 25Mhz RISC processors and four floating point coprocessors: R3000 and R3010.

Separate Buses for synchronization and data sharing:

-The Sync Bus for multiprocessing synchronization (spinlocks, semaphores) and interrupts distribution,

-The MPIink Bus for Data Block transfer (64Mbytes/sec.) and Data sharing with cache consistency protocol.

Two-level Data Cache:

-first level: 64Kbytes, write-through;

-second level: 256K bytes, write-back, bus watching, data coherence control.

The second level Data Cache interrupts the CPU to invalidate an entry in the first level Data Cache when a write-match occurs on the MPIink Bus.

The Write buffer, as in the Titan machine, maintains a write queue to accelerate the first level Data Cache write-through mechanism.

Self-Scheduling:

-A single run queue,

-Each processor is self-scheduling: when a process blocks, that processor search the run queue for new work which provides automatic load balancing.

III-GTX Graphics Subsystem:

The GTX architecture is an enhanced implementation of the GT architecture (based on seven new proprietary VLSI designs).

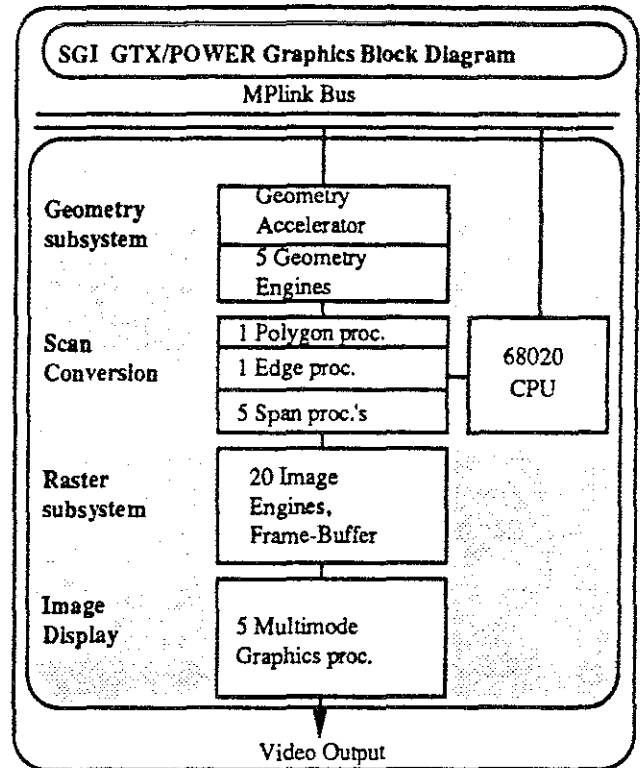


figure 3.3-2

The geometry computations, the pixel computations, and the display interface are implemented in this dedicated Graphics Subsystem.

The pixels are generated into the frame buffer along vertical span lines, from Span Processors which are given, for each span, the bottom coordinates and initial values (x,y,z,r,g,b,alpha), the length of the span, and the slopes for color and z, related to delta y.

There are fifo-buffers at each inter-stage junction, allowing Data Flow Regulation.

The first stage (Geometry subsystem) comprises two parts: the Geometry Accelerator and the Geometry Pipeline.

The Geometry Accelerator performs bus-interface flow control and data-format conversion into 32-bit floating point format.

The Geometry Pipeline performs all the geometry computations and delivers transformed and lighted vertices at the rate of 400K vertices per second. It is a five-stage pipeline made up of 5 Geometry Engines. According to Kurt Akeley, the GE uses WEITEK 3332 floating point processors and proprietary memory and controller-sequencer: it is capable of 20 Mflops peak performance. The Geometry Pipeline sustains an aggregate rate of 40 to 50 Mflops.

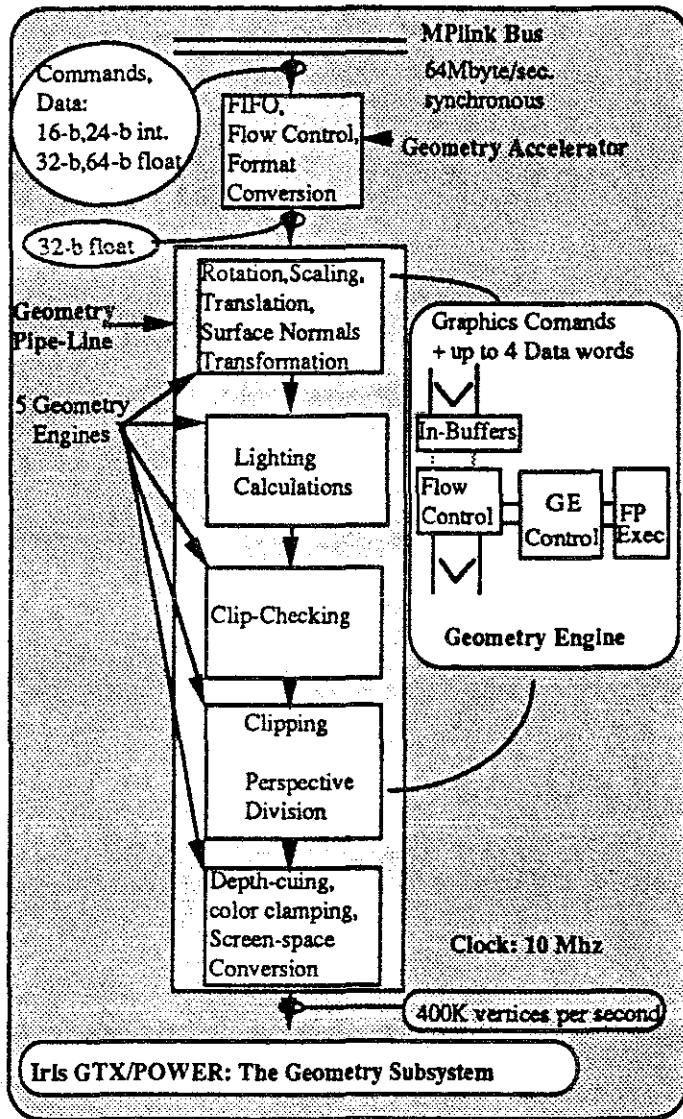


figure 3.3-3

The whole Graphics system comprises four pipelined stages: Geometry subsystem, Scan Conversion Subsystem, Raster Subsystem and Image Display Subsystem.

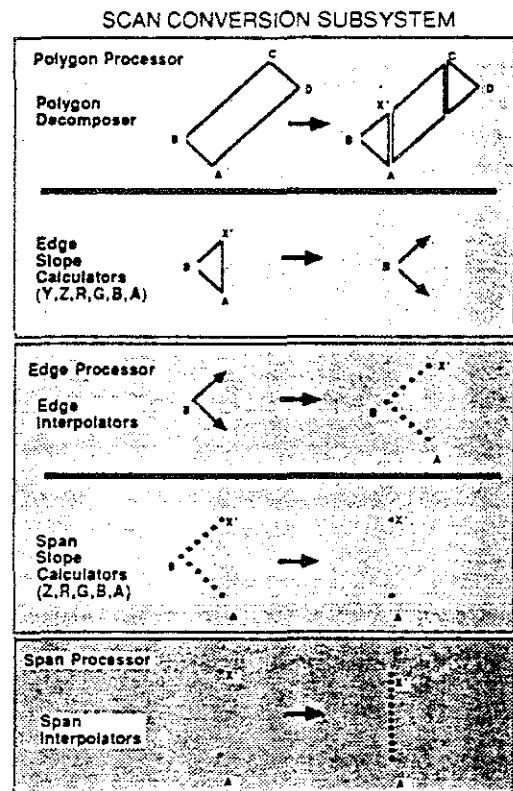


figure 3.3-4 from [Akeley88], p 243.

The second stage (Scan Conversion Subsystem) transforms polygon vertices into spans specifications for the Span Processors which transfer pixels descriptions to the third stage (Raster Subsystem).

The polygon vertices are sorted and decomposed into trapezoid edges by the Polygon Processor, and for each edge, the slopes of y,z,r,g,b,α are computed relative to Δx .

From each pair of top and bottom edges of the trapezoids, the Edge Processor interpolates the top and bottom coordinates of a series of spans (vertical segments) and computes the slopes of z,r,g,b,α relative to Δy for each span, thus producing the span specifications for the Span Processors.

The Span Processors compute the pixels values for each vertical segment and send them to the Image Engines

The Polygon Processor delivers 1M trapezoid edges per second, that is a 4-sided polygon in 6 microseconds, if the 4-sided polygon is decomposed in 6 edges (worse case, depending on the orientation of the polygon relative to the axes), which is a bit faster than the required 10 microseconds to achieve the goal of 100K 4-vertex polygons per second.

The Edge Processor generates two spans per microsecond, a bit faster than required to generate the 14 spans of a worse case 10x10 polygon with a diagonal x-oriented.

The Edge Processor also generates pixels during line fill. It fills lines as though each was a single trapezoid edge, generating pixels at the rate of 8 (Z-buffered) or 16 Mpixels per second. For long vectors, this will be the limitation of the drawing rate.

The Edge Processor also acts as an address generator and a flow controller. As so, it monitors the Span Processors and the Image Engines when performing the pixel transfer operations to/from/within the frame buffer.

The five Span Processors generate pixels at an aggregate rate of 40M pixels (Z-buffered) per second, 4 times the required rate for the 10x10 polygon.

Connecting the Edge Processor to the Span Processors the Pixel Bus is designed to support Span definition transmissions during polygon fill, pixel transfers during line fill, and also pixel moves within the frame buffer (span and zoom) and external video input to the frame buffer.

According to Kurt Akeley a Pixel Cache is connected

to the Pixel Bus to accelerate these transfer operations. It is 4K pixels wide. It is readable and writable by the Span Processors under the control of the Edge Processor, and it is also connected to the 68020 CPU for external pixel transfer.

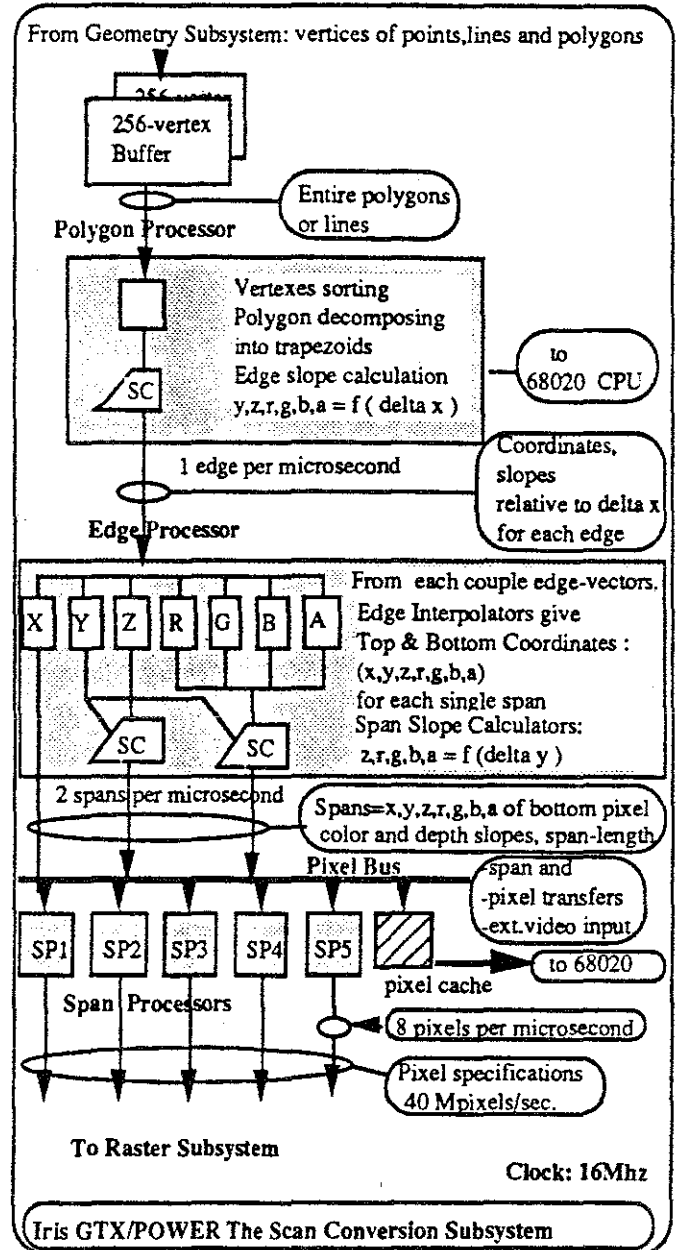


figure 3.3-5

The peak Pixel Bus rates are :

- read operation @ 5.3Mpixels/sec.;
- write operation @ 16Mpixels/sec..

The Video Input rate is 16Mpixels/sec.

The fill rates for various zoom factors are:

| | | |
|--------|---|-------------------|
| zoom 1 | @ | 4Mpixels/sec.; |
| zoom 2 | @ | 9.1Mpixels/sec.; |
| zoom 3 | @ | 12.1Mpixels/sec.; |
| zoom 4 | @ | 13.5Mpixels/sec.; |
| zoom 8 | @ | 15.3Mpixels/sec.; |

The Pixel Cache and the Polygon Processor are also connected to a 68020 CPU, connected itself to the MPlink Bus. According to Kurt Akeley this CPU is used to perform context switching in the Polygon Processor and Pixel transfer to/from the frame buffer.

The measured pixel transfer rates for 1/4 screen areas (640x512) are:

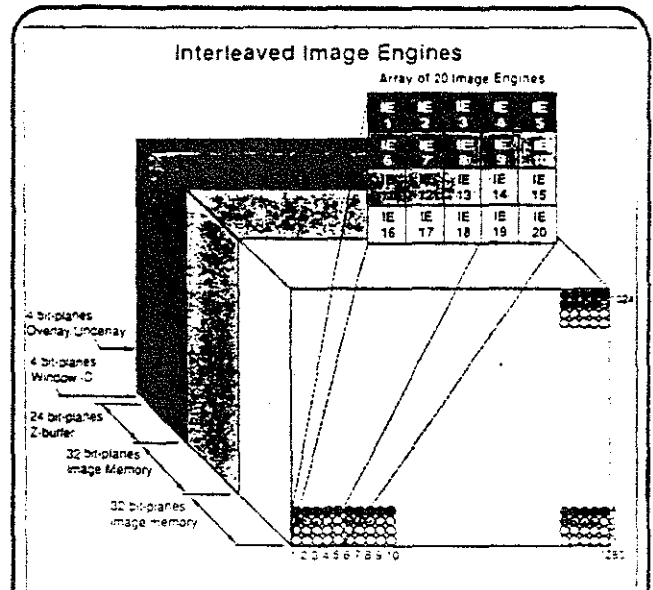
| | | |
|--------------------|---|--------------------|
| true color write | @ | 5.7 M pixels/sec.; |
| "" read | @ | 2 M pixels/sec.; |
| pseudo color write | @ | 8 M pixels/sec.; |
| "" read | @ | 3 M pixels/sec.; |

In the third stage(Raster Subsystem), the Image Engines, organized in an MIMD architecture, perform pixel access algorithms such as:

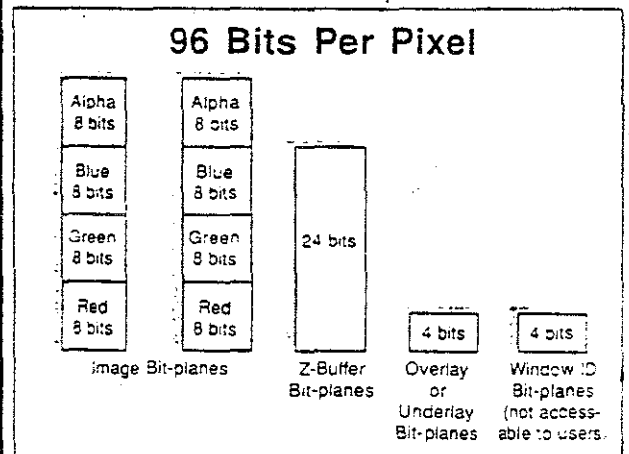
| | | |
|------------------------|---|------------------|
| Replace | @ | 80Mpixels/sec.; |
| Z-buffer | @ | 40Mpixels/sec.; |
| Z-buffer with blending | @ | 10Mpixels/sec.; |
| High-speed clear | @ | 160Mpixels/sec.. |

They also perform Window ID masking, testing the ID of each pixel against the ID of the current drawing process, before changing the frame buffer contents. And they support an alpha blending algorithm.

The Raster Subsystem contains 20 Image Engines each of which controls 1/20th of the frame buffer memory in an interleaved scheme. Groups of 4 Image Engines are driven by each Span Processor. The array of Image Engines tiles the frame buffer in a 5-wide, 4-high pattern.



From SIGGRAPH'88 Proceedings, p.243



Interleaved pixel access

The figure shows how Image Engines and pixel-memory planes are interleaved:

- each Span Processor manages each fifth column of 1024 pixels in the Frame Buffer,
- each Image Engine manages 256 x 256 pixels,
- 20 adjacent pixels on the screen are managed one by each of the 20 Image Engines.

figure 3.3-6

The bitplane memory comprises a total of 96 bits per pixel arranged as follows:

- two image banks of 32 bits, 8 bits for each of R,G,B,Alpha;
- one depth bank of 24 bits;
- one overlay bank of 4 bits;
- one window ID bank of 4 bits;

The fourth stage (Display Subsystem) comprises 5 Multimode Graphics Processors (MGP) that operate in parallel, one assigned to the pixels controlled by each Span Processor. They interpret pixel data received from the frame buffer and route the resulting R,G,B and alpha data to the Digital-to-Analog converters for display.

The five MGP's :

- concurrently read content of Image and Window planes,
- receive 64 image bits , 4 overlay bits and 4 window ID bits for each pixel; (The image and auxilliary bits are interpreted as a function of of the Window ID bits)
- determine color mode : RGB single or double-buffered, or color-index (pseudo-color 12 bits/pixel) single or double-buffered,
- display 16 arbitrary shaped different windows simultaneously,
- display cursor and exploit a 16x24-bit auxilliary table for overlay/underlay.
- exploit the alpha channel for video compositing.

The display format is 1280 x 1024, 60 Hz non-interlaced, or 30 Hz interlaced

IV-Some Graphics Applications implementation notes:

Access path between the processors and the Graphics Subsystem (IRIX graphics system software)

The Graphics pipe is mapped into the user address space. The pipe can thus be accessed either directly (for short commands) or indirectly through graphics "DMA" (atomic data transfer for high speed drawing). In addition , a feedback area, mapped into the address space of each graphic process, is used to return results of interactions with the graphics subsystem.

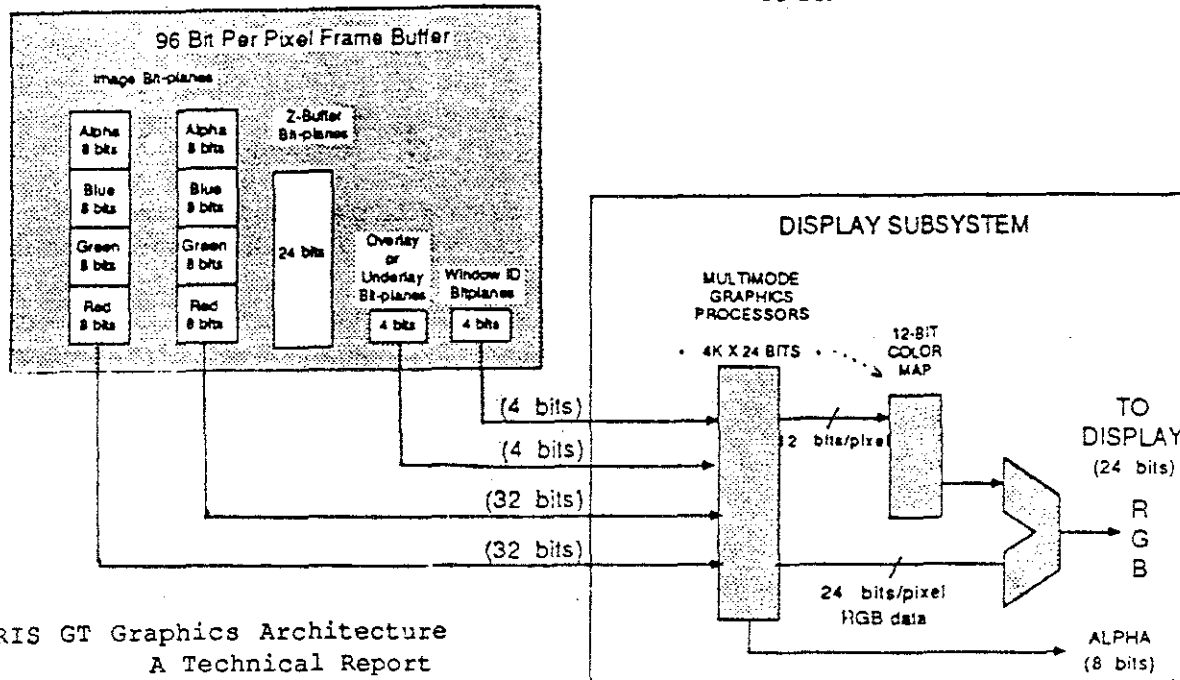
Context switching

A working set of up to 16 contexts is supported within the graphics hardware, and the Geometry Engines are able to dump and restore a context to and from the host, allowing additional contexts to share the graphics hardware.

The context of the Polygon Processor and of the other processors deeper in the structure is maintained through the 68020 CPU.

Multiple processes can share CPU's and Graphics hardware

Ownership of the graphics hardware can be allocated between competing processes on a per context switch basis. While a process is accessing the Graphics Pipe, others can be taking full advantage of the other CPUs and FPUs.



From IRIS GT Graphics Architecture
A Technical Report
figure 3.3-7

figure 3.3-8 : IRIS GTX Graphics Architecture

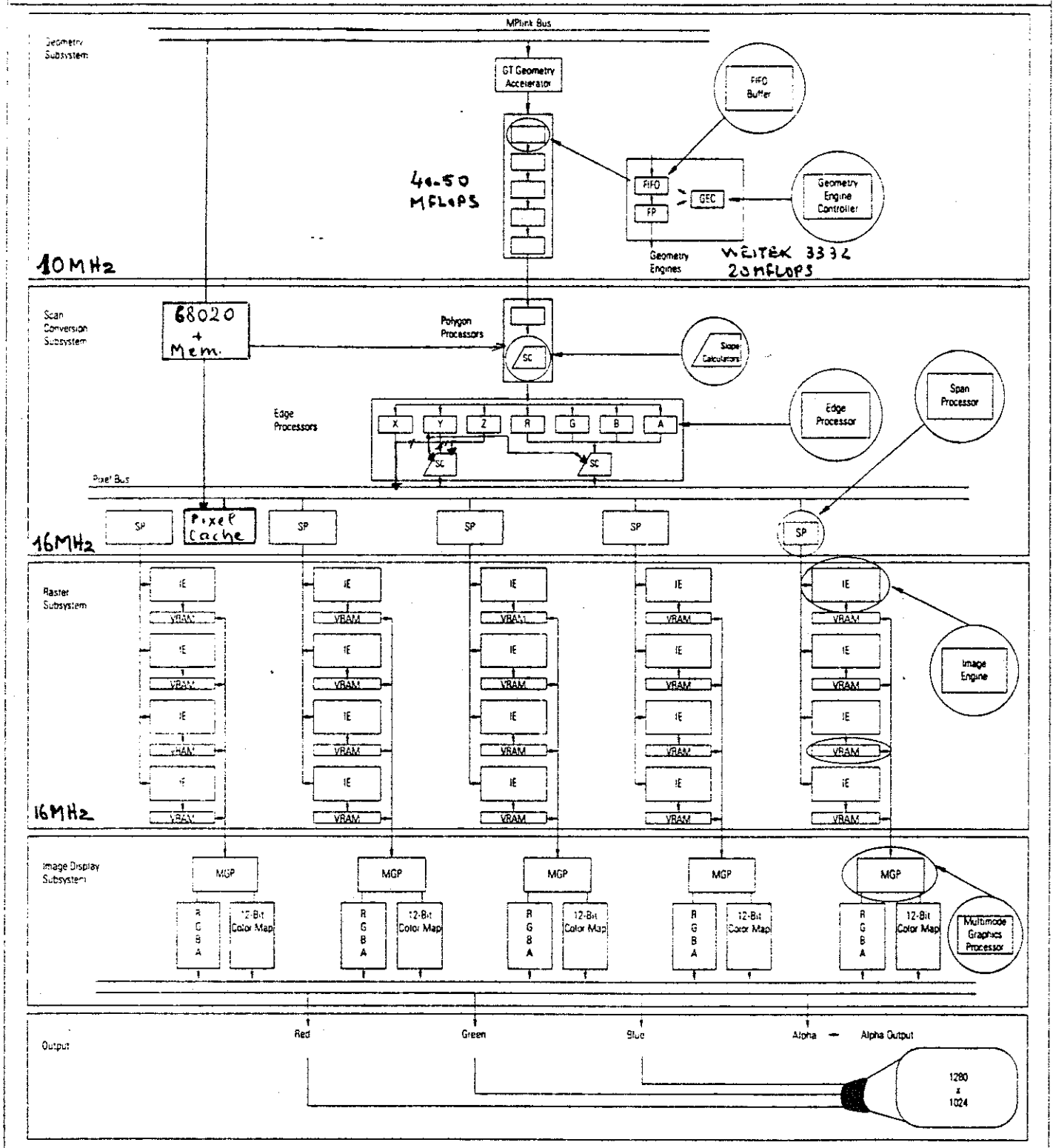


figure 3.3-8
 From POWER series Technical Report
 Modified after a phone call with Kurt Akeley

V- Structure and performances overview

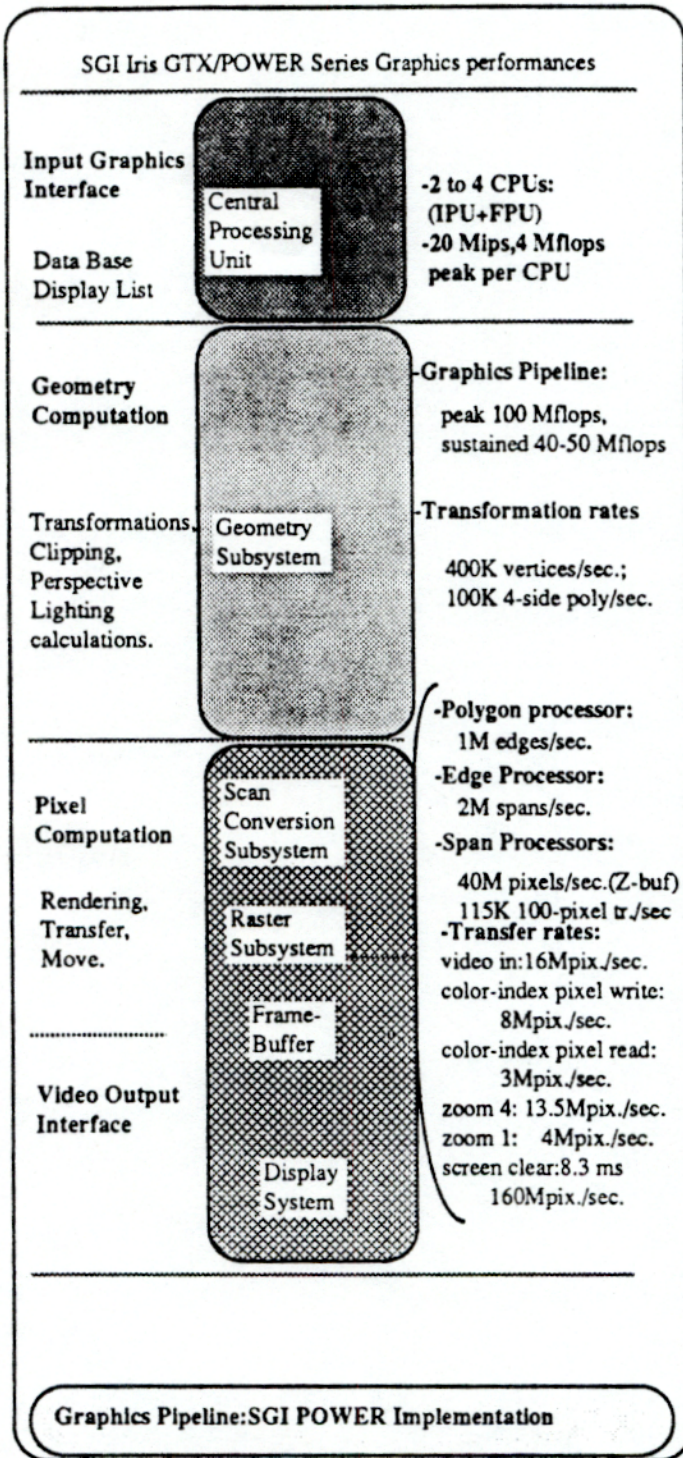


figure 3.3-9

VI References for Silicon Graphics

[Akeley88]-Akeley, K. and Jermoluk, T. - High Performance Polygon Rendering - SIGGRAPH'88, Computer Graphics, Vol.22, No. 4, August 1988, pp 239-246.

- Silicon Graphics:
 - IRIS GT Graphics Architecture: a Technical Report,
 - IRIS GTX : a Technical Report,
 - IRIS POWER Series Technical Report,
 - The Personal IRIS, a Technical Report.

3.4 STELLAR GS1000 Graphics Supercomputer

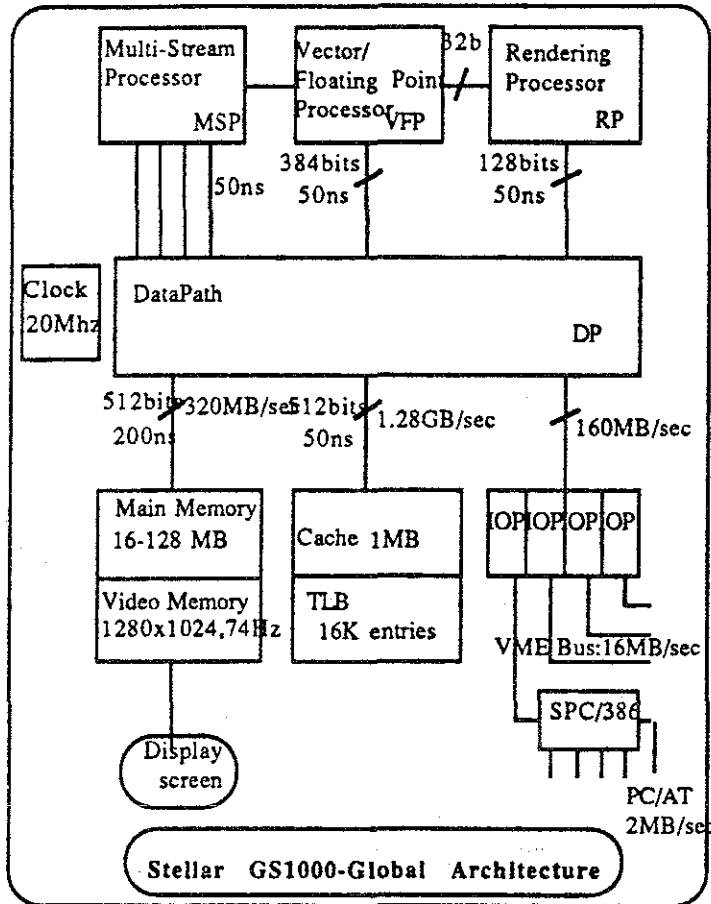


figure 3.4-1

I-Computer Graphics performances
(from [Apgar88], SIGGRAPH'88, p.260)

Geometry Computation rates:

- 551.2K lines/sec.: true or pseudo color 10-pixel lines,
- 163.7K triangles/sec.: pseudo color gouraud shaded Z-buffered, 100 pixels,
- 25.1K tr./sec. : pseudo color Phong shaded, Z-buffered, 100 pixels, 1 directional light.

Rendering rates:

- 600K lines/sec.: pseudo color, 10 pixels,
- 155K triangles/sec.: pseudo color, Gouraud shaded, Z-buffered, 100 pixels.
- 100K triangles/sec. : true color.....",
- 40.7 triangles/sec. : pseudo color, Phong shaded, Z-buffered, 100 pixels, 1 directional light.

- The pixel filling rate for large triangles:
- 80 M pixels/sec. : true and pseudo-color, without Z-buffering;
- 40 M pixels/sec. : pseudo-color, Z-buffering;
- 25 M pixels/sec. : true-color, Z-buffering;

These figures show that "additional vector processing throughput would be useful" ([Apgar88] p.260) to take full advantage of the performance of the Rendering Processor.

Some additional statistics are available in STELLAR System Overview p.63 in Animation.

II-Computational System

- highly integrated shared memory multiprocessor system,
- CPU and vector processor tightly coupled with the high performance display system ,
- highly integrated, large bandwidth Datapath between CPU, Vector Processor, Rendering Processor and Memory,

The DataPath

The Datapath is the core of GS1000 architecture.

It acts as an interconnect path between all of the functional units, and supplies the register storage for the MSP and VFP.

The Datapath hold an independent set of registers for each stream: 32 integer registers, 8 scalar floating point registers, 6 vector registers of 32 elements each, 3 vector control registers, Operating System control registers and an instruction buffer.

The Datapath also contains a pixel buffer (8 pixel registers each of 16 32-bit pixels) and 4 64-byte I/O-DMA registers.

The pixel buffer is used by the Rendering Processor in a "load-store like" architecture to perform the pixel rendering process.

The I/O DMA registers are used to buffer 64 bytes of data between the I/O Processor cache and main memory.

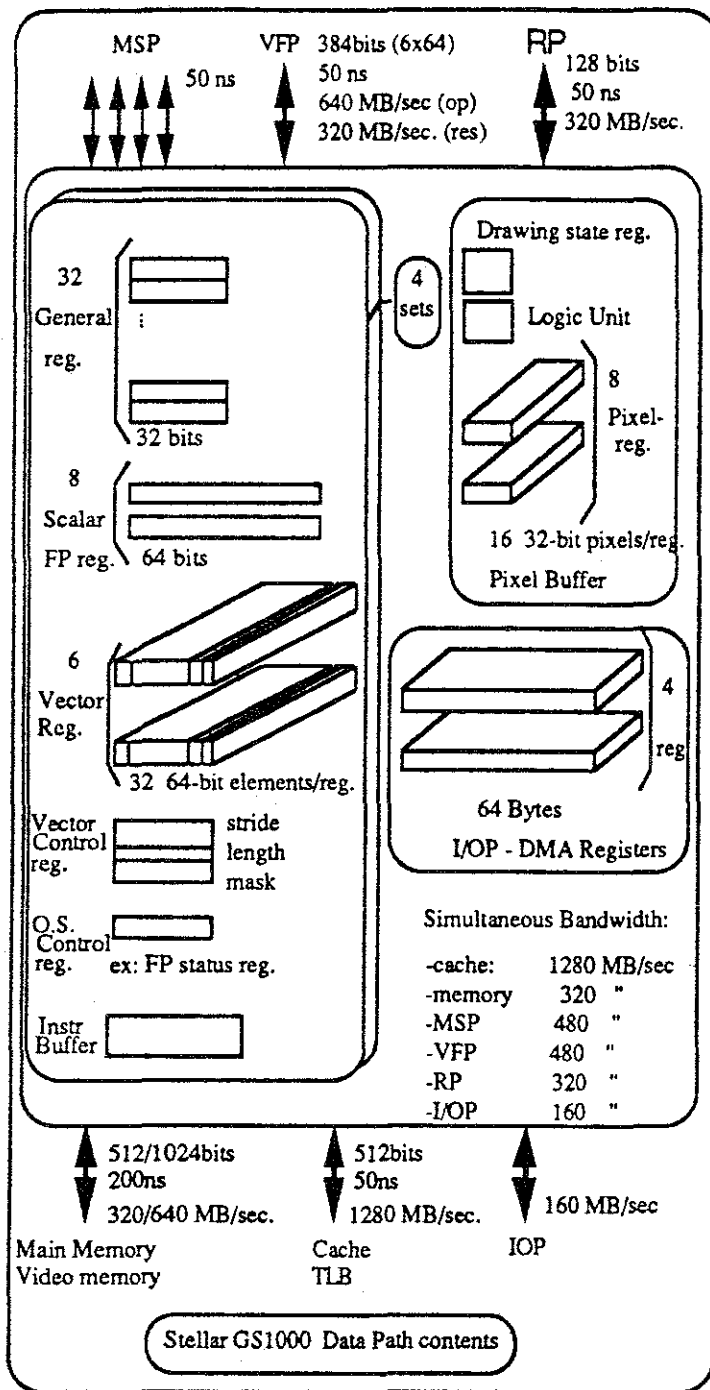


figure 3.4-2

The Multi-Stream Processor

The CPU is a Multi-Stream Processor (MSP), a single unit that operates as four logical processors. Each processor (or stream) can execute any set of machine instructions. The streams run concurrently, each completing an instruction every 200ns.

The MSP is organized as a 12-stage synchronous pipeline. The four instruction streams are interleaved onto the single pipeline. The instructions from the four streams are dispatched in a synchronous, round-robin manner. A new instruction from each stream enters the pipeline every 50 nanosecond tick, yielding, in the steady state, an instruction throughput of 20 MIPS.

A 25 MIPS throughput is made possible by the packetization technique, allowing the execution of two consecutive instructions in the same instruction cycle. For instance two register-to-register adds can be executed at the same time.

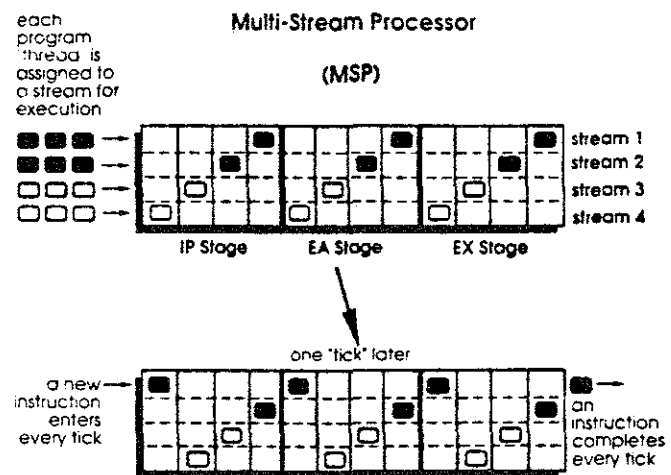
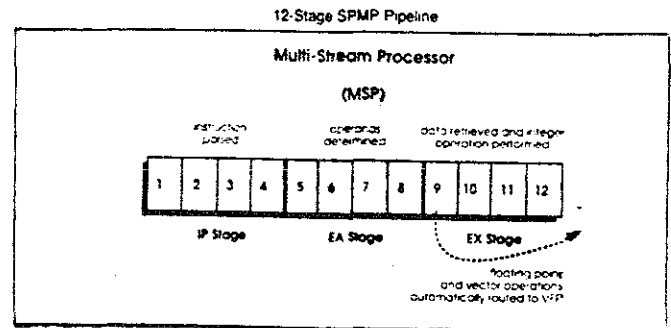


figure 3.4-3

From Stellar Graphics Supercomputer System Overview.

The Vector Floating Point Processor

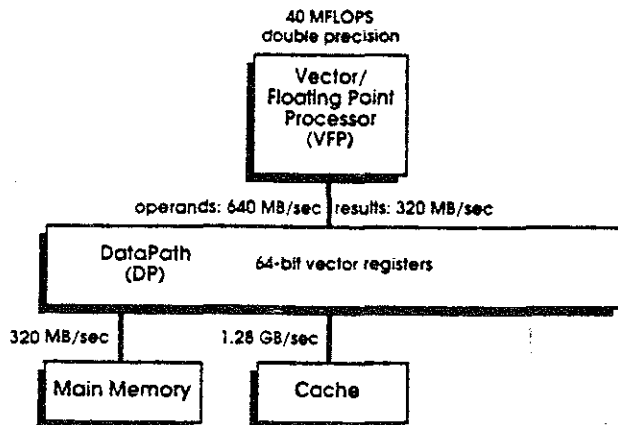


figure 3.4-3
From Stellar Graphics Supercomputer
System Overview.

The Vector Floating point Processor (VFP) is separate from the MSP, but is tightly integrated with it. The MSP fetches and decodes a floating point or vector arithmetic instruction entering a stream, and automatically activates the VFP to perform the execution phase.

The VFP is shared by all four streams of the MSP. It contains four high speed floating point compute engines pipelined to produce one floating add, multiply or multiply-add result every clock cycle (50ns).

The VFP can accept two new single-precision or double-precision floating point scalar arithmetic instructions every 50ns, taking advantage of the packetization technique to reach the peak throughput of 40MFLOPS.

see Stellar GS1000 Technical Overview p.16.

Memory subsystem

The Stellar GS1000 can be configured with 16 to 128 M bytes of main memory, 1 M bytes direct-mapped write-back cache and a 16K-entry direct-mapped Translation Lookaside Buffer (TLB).

Main Memory can be accessed at the rate of 64 bytes every 200ns (320 MB/sec.) over a 512-bit wide path.

The Datapath architecture guarantees that each of the four streams of the MSP can access the cache and TLB with no inter-stream contention. The MSP can access an entire 64-byte cache line every 50 ns (1.28 GB/sec.) over a 512-bit wide path.

A single cache is shared among all streams, so there is no need for explicit cache synchronization.

The TLB is connected to the Datapath through the same 512-bit wide path as the cache. The page size is 8KB so that as much as 128 MB of virtual memory can be mapped at one time in the TLB.

III. THE RENDERING PROCESSOR

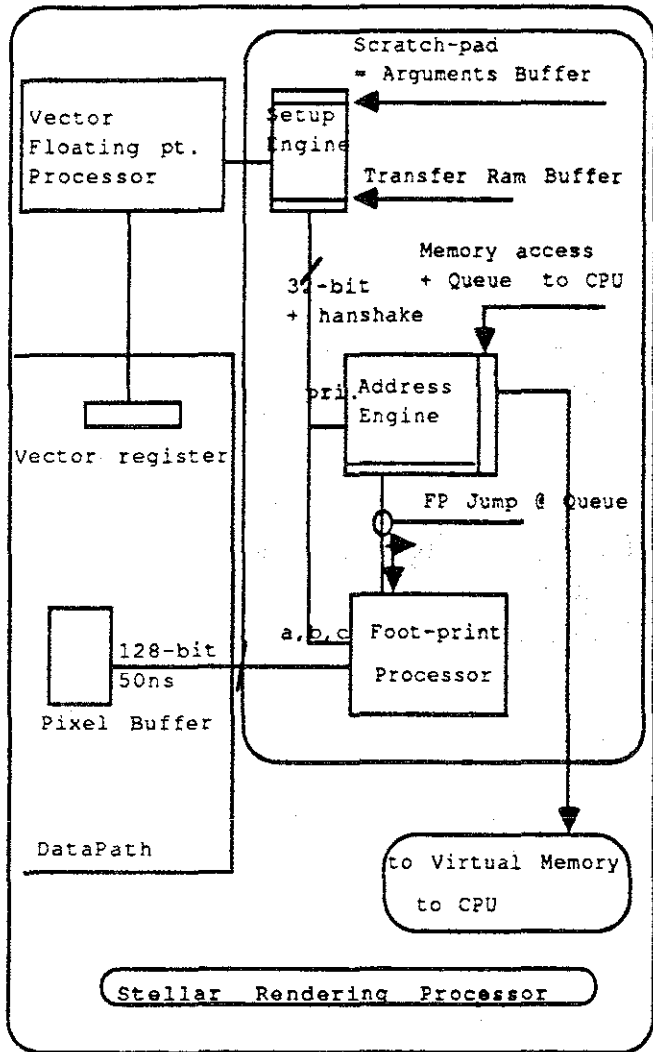


figure 3.4-4

The Rendering Processor (RP) performs all the per-pixel rendering computations in the system. It is micro-programmed for fast graphics rendering functions. It receives rendering commands from the vector unit, calculates individual pixel values, and writes the new values into pixel maps stored in virtual memory.

The Rendering Processor is a microcoded engine that operates on a 4x4 block of pixels (a "footprint") simultaneously. To render an entire graphics primitive, the RP calculates different pixel blocks until all points in the primitive have been processed.

The RP is implemented with three independent engines acting as a rendering pipeline: the Setup Engine, the Address Engine and the Footprint Engine. These engines use two specially designed ASICs: the Setup-Address (SA) ASIC and the Toe ASIC. The SA ASIC is used once in the Setup Processor and once in the Address Processor. The Toe ASIC is used 16 times in the Footprint Processor.

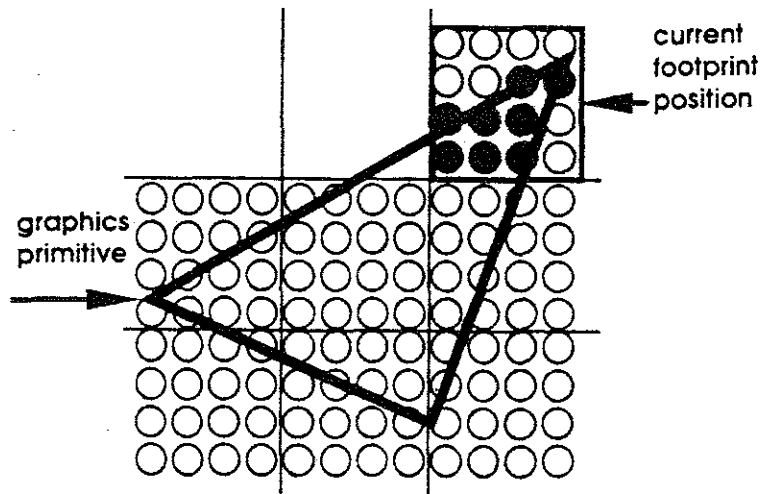
The Setup Engine converts vertex coordinates and color values into coefficients of linear equations evaluated by the next two stages.

The Address Engine uses the linear equations to determine which 4x4 blocks of pixels are covered by a primitive, and sequentially pilots the "Foot Print" Engine over those blocks while coordinating the virtual memory accesses. As necessary, it steals one of the MSP streams in order to transfer the blocks of pixels between memory and the pixel buffer, where they can be read by the Foot Print Engine.

The Foot Print Engine consists of 16 identical "toe" processors that evaluate the same linear equation in parallel. The FP Engine evaluates the bounding functions for the primitive, computes the pixel values and Z coordinates, and performs the depth buffering comparison in parallel for the 16 pixels of the block.

Each of the 16 toe processors works on a different pixel, performing an addition or multiplication every 50ns, for an aggregate rate of 320 MIPS.

Rendering a Graphics Primitive



16-pixel footprints

figure 3.4-5
From Stellar Graphics Supercomputer System Overview

IV- VIRTUAL PIXEL MAPS

The Rendering Processor performs all of its operations on Virtual Pixel Maps (VPMs). It writes the computed new values into pixel maps stored in virtual memory.

The Rendering Processor supports rendering with 12-plane pseudo-color and 24-plane true color. A single VPM is used for rendering in pseudo-color: the Z-buffer occupies 16 bits, the color 12 bits and 4 bits are unused. Two VPM are used for true color: 24 color bits and 8 unused bits occupies the first VPM, a 32-bit Z-buffer is held in the second VPM. Several VPM are needed to support anti-aliasing of solids, texture mapping, and transparency.

The Rendering Processor can read, modify and write images at the rate 40 M Z-buffered pixels/sec. (80 M non-Z-buffered pixels/sec.). It supports flat, Gouraud and Phong shading, transparency and anti-aliasing both during primitive generation and for post-processing filtering.

V- THE VIDEO MEMORY

The Video Memory is a subsystem separate from the Rendering Processor and connected to the Datapath through a 512-bit wide path common to Video Memory and Main Memory.

During the Rendering Process, pixels are stored as rectangular arrays in virtual memory called Virtual Pixel Maps (VPM). Then the images (or visual portions of images) are transferred from main memory to Video Memory to be displayed.

Each Main Memory location holds 16 32-bit words, thus it can hold 16 pixels organized either as 4x4 blocks or 16x1 strokes. The Video Memory is a 1280x1024 array of 32-bit pixels, each location holding a 16x1 stroke.

Both Main Memory and Video Memory support two types of memory cycles: the standard cycle (a single 512-bit memory location every 200ns) and the overlapped memory cycle (1024 bits in 200ns).

The Memory access rate can be either 320MB/sec or 640 MB/sec., yielding a peak read or write rate of 160 M pixels/sec (in configurations of 32MB or more memory).

The transfer rate from main memory to video memory is 60 M pixels/sec. because the pixels need to be reorganized from 4x4 blocks to 16x1 strokes, which is done in the Pixel Buffer.

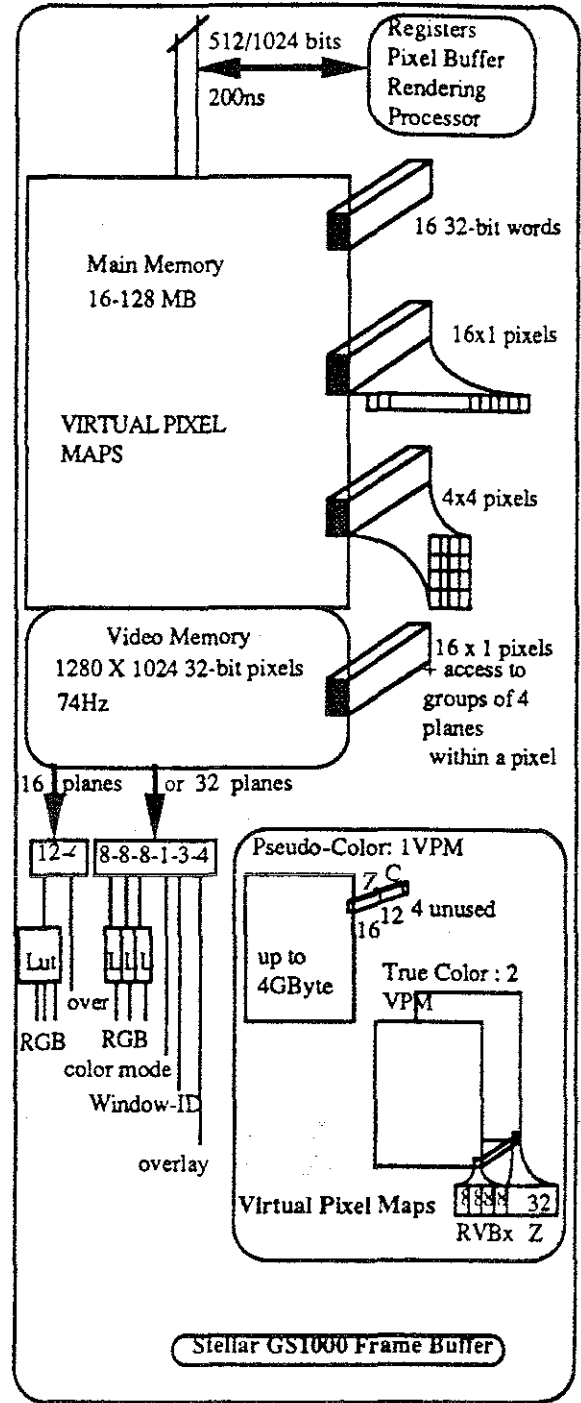


figure 3.4-6

VI- The GRAPHICS PIPELINE

All the geometry computations are implemented by a software package written in C and assembly code running on the MSP and the VFP. This software package provides the graphics device interface for the machine. It is called X Floating Point Device Interface (XFDI) and it supports a large number of graphics primitives: points, lines, line meshes, line polyhedral meshes, triangles, triangle meshes, triangle polyhedral meshes, and spheres.

The XFDI performs coordinate transformations, clipping, lighting calculations and device coordinate conversion, in single precision floating point using the VFP. The transformation of a 3D point takes about 800ns with the VFP operating at over 35 MFLOPS.

The XFDI also supports flat, Gouraud and Phong shading with diffuse and specular lighting, and up to 16 colored light sources, with point or directional lights. For specular lighting, the viewer may be at infinity or local to the object. Clipping can be done to the standard six clipping planes and to 16 arbitrary clipping planes.

The VFP acts as a front-end for the Rendering Processor, supporting three dedicated vector instruction (send-to-renderer, receive-from-renderer and execute-renderer). The VFP sends rendering commands and parameters to the Rendering Processor.

The Rendering Processor (RP) performs all of the per-pixel rendering computation in the system. It calculates pixel values, operating on the pixel buffer in the Datapath, and stores them into pixel maps in virtual memory.

The MSP transfers the images from main memory to video memory, through the pixel buffer.

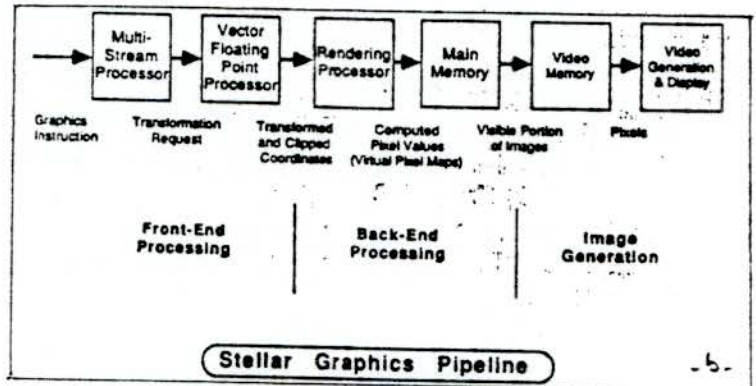
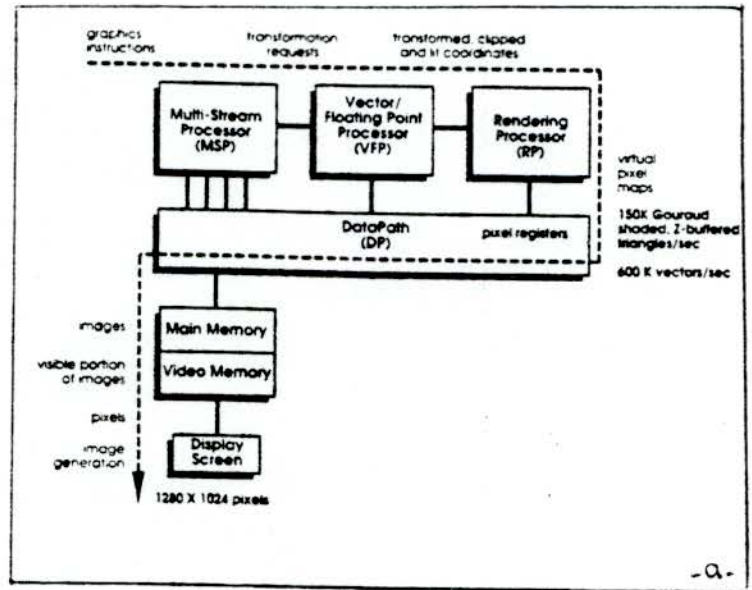


figure 3.4-7

- a) from Stellar Graphics Supercomputer System Overview.
- b) from [Sporer88], p.246

VII- Structure and performances overview

VII - References for Stellar GS 1000

[Apgar88]-Apgar, Brian et al. - A Display for the Stellar Graphics Supercomputer Model GS1000 - SIGGRAPH'88, Computer Graphics, Vol. 22, No. 4, August 1988.
 [Curran88]-Curran, Lawrence - Stellar's Graphics Machine Stresses Interactivity - Electronics, March 17, 1988.
 [Sporer88]-Sporer, Michael et al. - An Introduction to the Architecture of the Stellar Graphics Supercomputer - COMPCOM'88, February 29, 1988.
 -Stellar Computer: Stellar Graphics Supercomputer Model GS1000, System Overview.

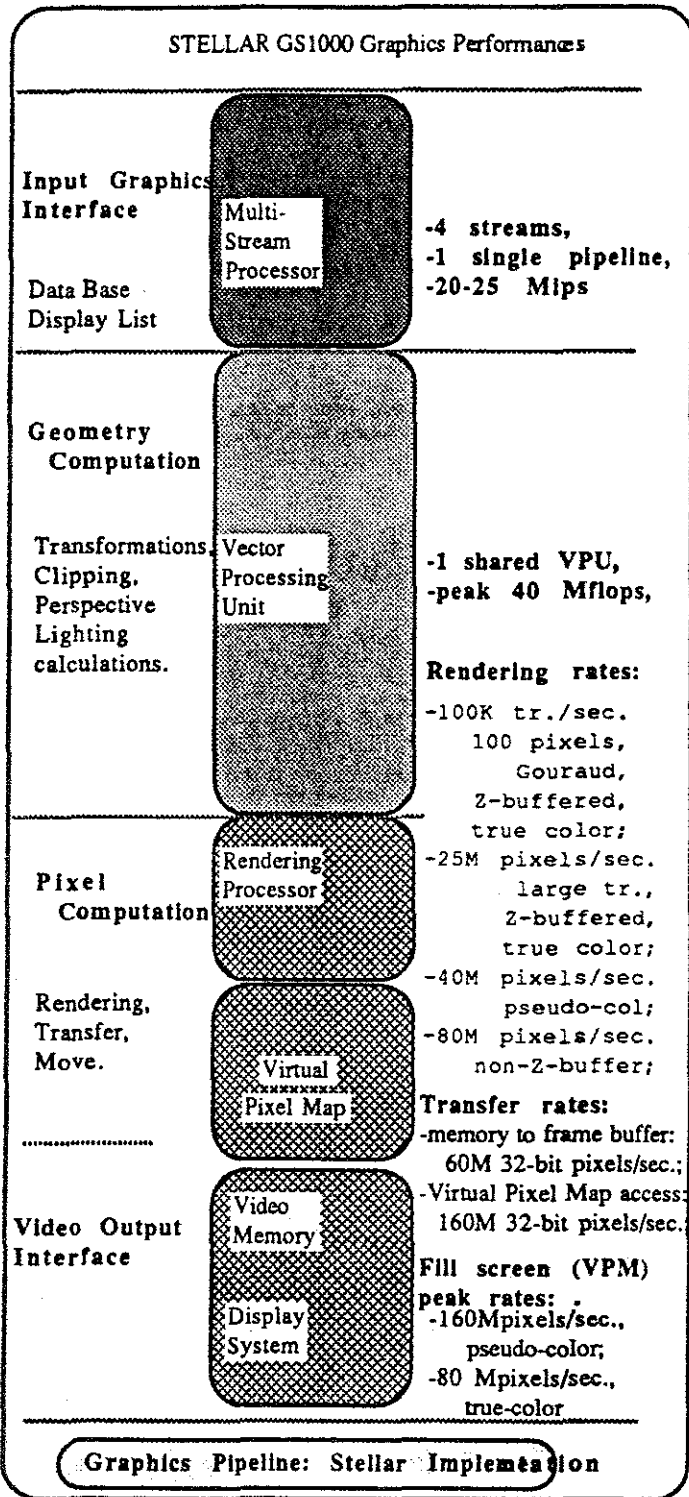


figure 2.4-8

4 PART C:

Announced 1Million tr./sec. graphics designs

4.1 INTRODUCTION

1 Million triangles/second graphics designs

The current commercial Graphics Supercomputers have just began their live and yet, some new designs are already claiming an order of magnitude increase in rendering rates.

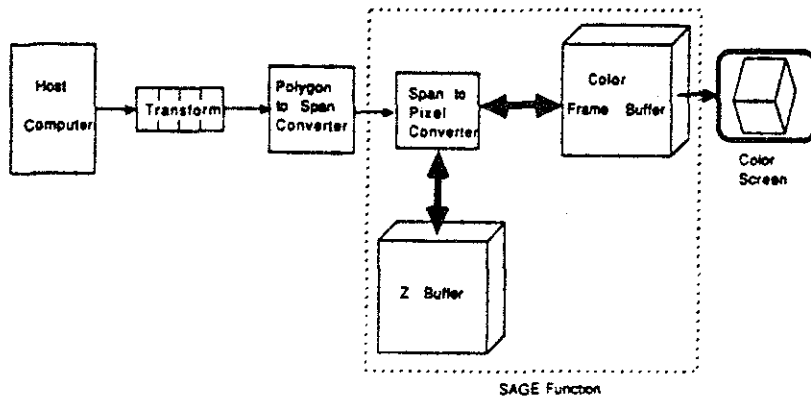
How do they do? where were the bottlenecks ? what is THE solution?

We look at three different approaches, all based on parallel implementations at different levels of the graphics pipeline, and also on the design of custom VLSI:

- SAGE: Systolic Array Graphics Engine;
- the Triangle Processor.
- Pixel planes 5;

The informations we have on these three designs are not of equal volume and validity, but it is interesting to compare the basic principles.

4.2 SAGE: Systolic Array Graphics Engine



Three Dimensional Graphics System

figure 4.2-1
from[Gharachorloo 5-88]

I-Introduction

The system designed at IBM Research Division, Watson Research Center, is built around a VLSI chip (SAGE) and a graphic system (VIP's). It can sustain sub-nanosecond pixel rendering rates for 3D polygons, and can be used to render about a million Z-buffered Gouraud shaded polygons per second.

In their analysis of the current graphics designs, they point out that the major barriers to real time raster color image generation are the computation rates and the large image memory bandwidths.

Computation rates can be reached using parallel processing, but the frame buffer remains the main bottleneck, especially while we keep rendering pixels as the basic primitive. Their suggestion is to change the basic hardware rendering primitive from low level pixels to higher level primitives, such as horizontal spans.

This has two advantages:

- communication bandwidth reduction because of the smaller number of primitives transmitted;
- parallel rendering of the primitives by exploiting wider and faster on-chip bussing structures.

Furthermore, "assuming that the computation for generating a new frame can be performed in real time, one may then attempt to reduce the size of the frame buffer memory since it is no longer necessary to store a complete frame buffer in order to refresh future frames." [1]

II-Basic Principles

The basic idea of the implementation was presented in [Gharachorloo 85]. The Super Buffer approach is an attempt to break the barrier of limited image memory bandwidths using on-the-fly real time image generation in synchronization with the raster beam.

The architecture is based on a virtual scan line buffer instead of a whole frame buffer. The system is composed of two main parts: the Scanline Processing Unit which transforms high level polygons into intermediate scan line instructions; and a VLSI Raster Graphics Engine which simultaneously updates, buffers and refreshes one line of the raster image.

In such a system the polygons have to be presorted according to the topmost line on which they become active.

The basic hardware rendering primitive they have chosen is the horizontal span because [2]:

- it is simple enough to be executed in hardware;
- it is powerful enough to achieve the nanosecond pixel rendering goal;
- it is general enough to allow for future algorithmic enhancements.

A span is described by:

- the position of its two end points,
- its depth at the left end point,
- the slope of depth relative to x along the scan-line,
- a color triplet value at the left end point,
- and the slope for each color value along the scan-line.

Using the horizontal span as primitive, the polygon rendering problem splits into two loops:

- a horizontal filling inner loop which transforms spans into pixels;
- a vertical interpolation outer loop which breaks individual polygons into horizontal spans.

This is the basis of the design:

- the horizontal span-pixel transformation algorithm is mapped into a custom CMOS VLSI called SAGE (Systolic Array Graphics Engine);
- the vertical polygon-spans interpolation process is performed by the VIPs (Vertical Interpolation Processor).

III-SAGE: the Systolic Array Graphics Engine

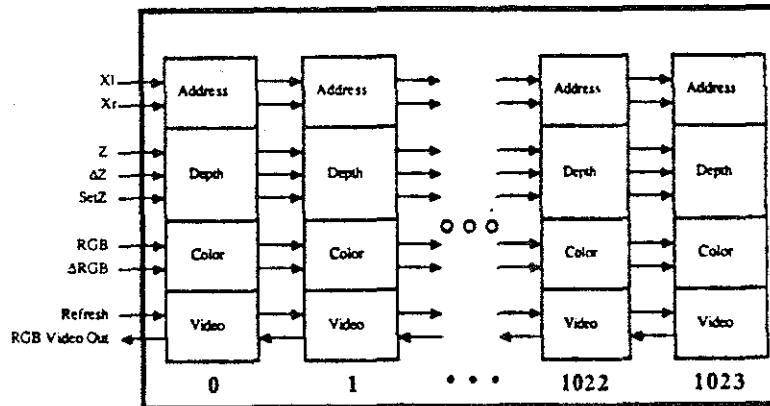


figure 4.2-2 Systolic array of pixel processors in SAGE.
from[Gharachorloo 8-88]

The Systolic Graphics Engine is the heart of the Super Buffer system. It maps the horizontal span rendering loop into a systolic array of pixel processors: one per pixel on a scanline. Each SAGE chip contains 256 pixel processors and 4 chips are needed to build an 8-bit 1024x1024 system, (12 chips for a 24 bits/pixel system) [3].

The pixel processor itself is designed to efficiently execute the body of the loop in the Fillspan procedure (figure 3.3-3). The incremental interpolation involves adding ΔZ and ΔRGB to the current Z and RGB values to get the depth and color to the next pixel

```
PROCEDURE FILLSPAN (Xl,Xr,Z,ΔZ,RGB, ΔRGB)
```

```
static int ZBuf [0..1023], RGBBuf [0..1023]
```

```
FOR x := 0 to 1023 DO BEGIN
```

```
  IF x1 ≤ x ≤ xr THEN BEGIN
    IF Z ≤ ZBuf[x] THEN BEGIN
      ZBuf[x] := Z;
      RGBBuf[x] := RGBBuf;
    END;
    Z := Z + ΔZ;
    RGB := RGB + ΔRGB;
  END;
END;
```

figure 4.2-3
from [Gharachorloo 8-88]

The pixel processor is composed of 4 blocks:

- the address block compares the processor ID (or x position in the pipeline) against the left and right ends of the current span, and generate a write-enable signal to the depth and color blocks when the comparison is successful;

- the depth block, holds in a register Zbuf the previous Z value at this pixel address, compares the input Z against Zbuf, updates Zbuf if necessary, and adds ΔZ to Z according to the depth slope of the span;

- the color block stores input RGB values into 3 registers RGBbuf if depth comparison was successful in the depth block, and adds RGB with ΔRGB according to each color component slope along the span;

- the video block or refresh block allows the shifting out of the pixel color from RGBbuf.

Each SAGE chip contains a pipeline of 256 pixel processors, and actually pipelining is used both in the horizontal direction (a 256-stage pipeline) and the vertical direction (a three-stage pipeline at the pixel processor level). The address block, the depth block, and the color and refresh blocks operate independently. All inputs and outputs to the array are through the first pixel processor: the depth parameters are delayed by one clock cycle, the color parameters by two clock cycles to maintain proper operation of the three-stage pixel processor.

After all the spans for a given scanline have been clocked into the pipeline, a refresh token is sent which causes the content of all the color buffers to be left-shifted out on the video port of the first pixel processor at the rate of one pixel every clock cycle.

The chip operates at a 40 ns clock cycle allowing for the processing of up to 25 Million spans per second (one span per cycle enters the array).

To build a 1024x1024 system, 4 256-pixel processor SAGE chips are needed to handle the 1024 pixels of one scanline and they can be connected either in series or in parallel.

When connected in series, the 4 chips act as a single 1024-pixel processor chip: the spans are passed in at a rate of 40 ns/span and video is shifted out in the reverse direction at the same rate. This rate is not fast enough to refresh a 1024x1024 CRT directly for which the need is 11 ns/pixel and a frame buffer is hence necessary.

When connected in parallel, each chip contains pixel processors for every fourth pixel in the scanline. The four chips receive the same span command in parallel at a rate of 40 ns/span. Now 4 pixels are available every clock cycle, yielding to a rate of 10 ns/pixel when serialized, but only 256 span command can be accepted before refresh. This configuration allows for direct driving of a 1024x1024 CRT.

The configurations can be tuned to the particular application, with or without an additional frame buffer.

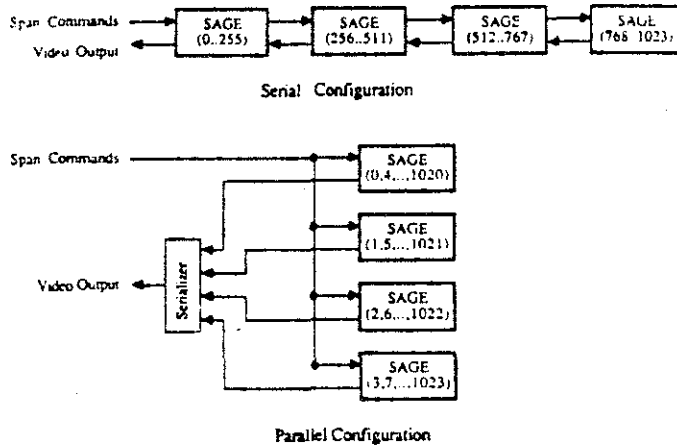


figure 4.2-4 SAGE Configurations for 1024 processors. from [Gharachorloo 8-88]

IV-The Graphics System

As previously mentioned, the rendering problem splits into two steps: one is carried out by the SAGE chips, the other deals with converting polygons into spans and managing the active polygon list for each scanline.

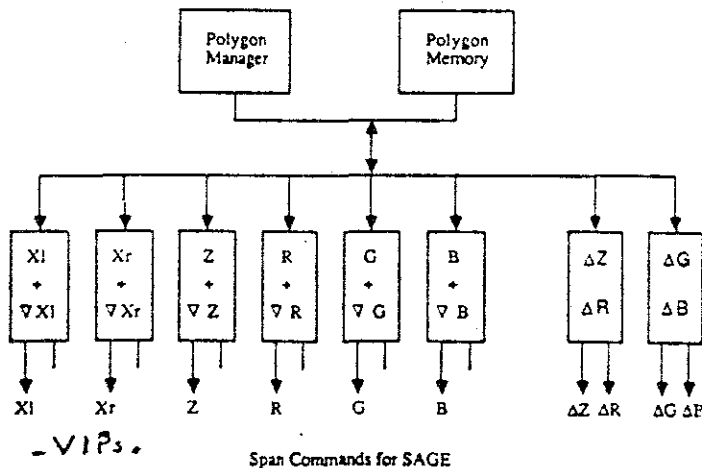


figure 4.2-5 Polygon to Span Interpolation. from [Gharachorloo 8-88]

This is supported by an array of 8 identical Vertical Interpolation Processors which execute the conversion

procedure (six axis interpolation)[2].

"The VIPs are responsible for supplying SAGE with all the horizontal span primitives on a given scanline by incrementally computing the intersection of a horizontal cutting plane with the 3D polygons in the scene"[2]. The VIPs have internal memory to store up to 512 active polygons. Six VIPs are used to perform interpolations and two store the constant slopes relative to x ($\Delta xZ, \Delta xR, \Delta xG, \Delta xB$) for the planar polygons.

A polygon manager computes slopes for the polygons and updates the list of currently active polygons in the VIPs.

The VIPs are synchronized with the SAGE chip, converting and rendering a span every 40 ns

The Graphics System is intended to be connected to a floating point multiprocessor workstation (ACE).

V-Performances

The system is still in a debugging state and is not planned to be part of IBM products but the average expected performances are quite high:

SAGE can render a horizontal 3D span in every 40ns-clock-cycle (25 million spans/sec) independently of the pixel length of the span as long as this length does not exceed the actual configuration of the system (for example 4 SAGE chips to render 1024 pixels long spans).

For an average span width of 32 pixels, SAGE updates pixels at an average rate of 1.25 ns/pixels (800 million pixels/sec).

For an average polygon height of 32 lines (1000 pixels polygons) the system is capable of rendering close to one million polygons per second (780K).

In the best case (10-span high polygons), the performance can reach 2.5 million polygons/sec [3].

VI-References for SAGE:

[1][Gharachorloo 85]- Gharachorloo, Nader and Potle, Christopher: SUPER BUFFER : a Systolic VLSI Graphics Engine for Real Time Raster Image Generation. 1985 Chapel Hill Conference on VLSI, pp 285-305.

[2][Gharachorloo 8-88]- Gharachorloo et al.: Sunanosecond Pixel Rendering with Million Transistor Chips. in Proceedings SIGGRAPH'88, Computer Graphics, vol. 22, no. 4, August 1988.

[3][Gharachorloo 5-88]- Gharachorloo et al. - A million transistor Systolic Array Graphics Engine - in Proceedings of 1988 IEEE International Conference on systolic arrays, San Diego, May 88, pp 193-202

4.3 The Triangle Processor

I-Introduction

The designers of the Triangle Processor, at Schlumberger Palo Alto Research, have also pointed out the memory access bottleneck as the most significant factor to be accelerated when designing a new high performance graphics architecture. They have re-examined the whole process of image generation, looking for an improvement of at least an order of magnitude over conventional state-of-the-art systems [Deering 88].

As an alternative to the common association processors-pixels, they suggest to associate processors with polygons.

A pipeline of Triangle Processors rasterizes the geometry, then a further pipeline of Shading Processors applies Phong shading with multiple light sources. The triangle processor pipeline performs 100 billion additions per second, and the shading pipeline performs two billion multiplies per second.

The 3D graphics System built with these two pipelines should be capable of displaying more than one million triangles per second.

II-Basic concepts

The Triangle Processor

The Triangle Processor is a processor dedicated to the rasterization of a single triangle. Connected in series, they form a triangle pipe into which a raster ordered stream of uncolored pixels is fed. Each triangle processor is responsible for one triangle, but it is not allocated to this particular triangle for the whole image. When the last pixel in a triangle has been rendered, its processor becomes available for re-use on another triangle. Thus a Triangle Processor may be used to render a number of triangles which do not overlap in the y direction.

The triangle pipeline delivers surface normals and color or material indexes for each pixel.

The Normal Vector Shader

The second pipe applies a full multiple light source Phong illumination model independently to each pixel coming from the triangle pipe.

III-System overview

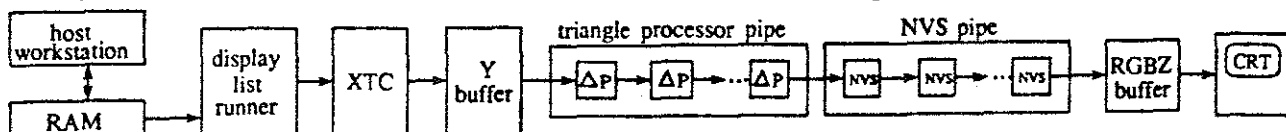


figure 4.3-1
from [Deering 88]

Block diagram of the GSP-NVS system.

The complete graphics system is called "graphical signal processing with normal vector shading" (GSP-NVS).

The block diagram in figure 3.4-1 shows a fairly conventional processing up to the Y buffer.

The DLR (display list runner) handles viewing matrix manipulations and performs simple subroutines (jump, pick, tests,...) but passes triangle and vector commands to the XTC.

The XTC applies the usual transformation-clip-and-set up operation to triangles and vectors.

The Y buffer is organized as linked lists of active triangles, one for each scanline. The Y-buffer input process sorts the triangles into bins indexed by their first active scanline number and stores them until all the triangles in a given frame have been buffered. The output process feeds these triangles into the triangle pipe for rasterization, in scan line order, and sequences the rasterizing, first sending the triangles data and then the stream of 1280 "blank" pixels.

Unlike the processor-per-pixel architectures, geometry overflow is possible if there are more active triangles on the same scan line than Triangle Processors available. The Y-buffer then takes care of the overflow, watching the number of triangle processors actually busy, and keeping the overflowed triangles for a second rendering pass.

The RGB pixels are finally stored in a frame buffer.

IV-Internal architecture and operation

The Triangle Processor

A complete triangle pipe can be built integrating eight or ten Triangle Processors per chip and directly wiring together any number of chips in a linear pipe.

In order to reduce the chip pin count, data are transferred from chip to chip on both rising and falling phases of the clock, and the x pixel location is locally generated on each chip by the G-unit. The G-Unit is reset on the reception of the SOL (Start Of Line) command and generates pixel addresses until reception of the EOL (End Of Line) command. The pixel address flows through the Triangle Processors of one chip following this SOL command. Upon reception of a new pixel address, the TP performs various computations to determine if this pixel lies in the area of its current triangle. If no, the current pixel values are passed unchanged to the following TP. If the input pixel address lies in the current triangle area, the values of the pixel are updated before being passed to the following TP.

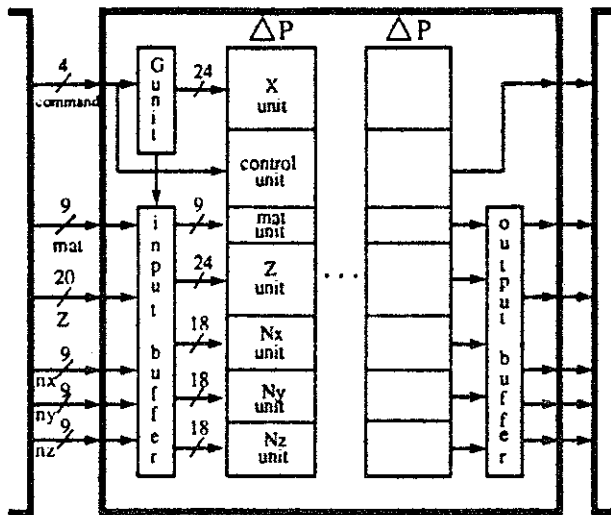


figure 4.3-2 External data flow and internal organization of the Triangle Processors chip. (Note that only 30 data input pins are needed for the 60 signals shown, as the pins are double clocked.)
from [Deering 88]

Each Triangle Processor is composed of 7 different function units (figure 3.4-2):

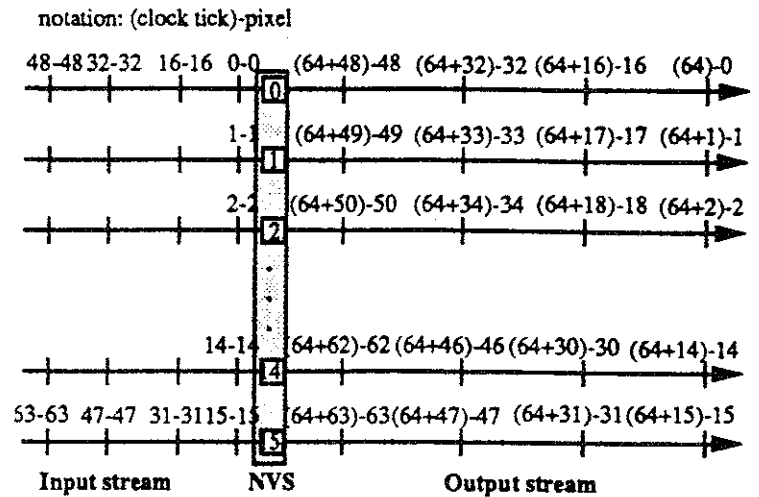
- the Control Unit receives commands via the 4-bit command data bus and sequences the other units;
- The X-Unit tracks left and right edges of triangles relative to the current scan line, compares the pixel x coordinate with the x position of these left and right edges and signals the other units whenever the current pixel falls within the local triangles;
- the Z-Unit interpolates the z depth in both x and y directions across the triangle, compares this locally computed value to the z value input from the preceding TP, and attach the new z value to the current pixel if it falls within the local triangle and if the locally computed z value is less than the input z value; the resulting z value is propagated to the next TP;
- the three N-Units (Nx, Ny, Nz) interpolate the surface normal vector N over the triangle; according to the result of the comparison in the X-Unit and the Z-Unit, either the locally interpolated N normal vector or the input N normal vector is forwarded to the next TP;
- the M-Unit latches a 9-bit material index for the local triangle; at each pixel, depending on the x and z comparisons, either the input index is propagated or the stored index is substituted;

The lighting chip:NVS pipe

The illumination pipe of the GSP-NVS consists of 16 identical NVS chips.

The NVS chip is pipelined: the shading process takes 64 cycles, but 4 pixels can be at various stages of the process at the same time, providing a 16 clock cycle throughput per chip.

One guess of the pipe structure could be the following, using the mentioned round-robin technique to distribute the input stream:



pixel stream through NVS pipe

figure 4.3-3

V-Performances

The cycle time is 50 ns.

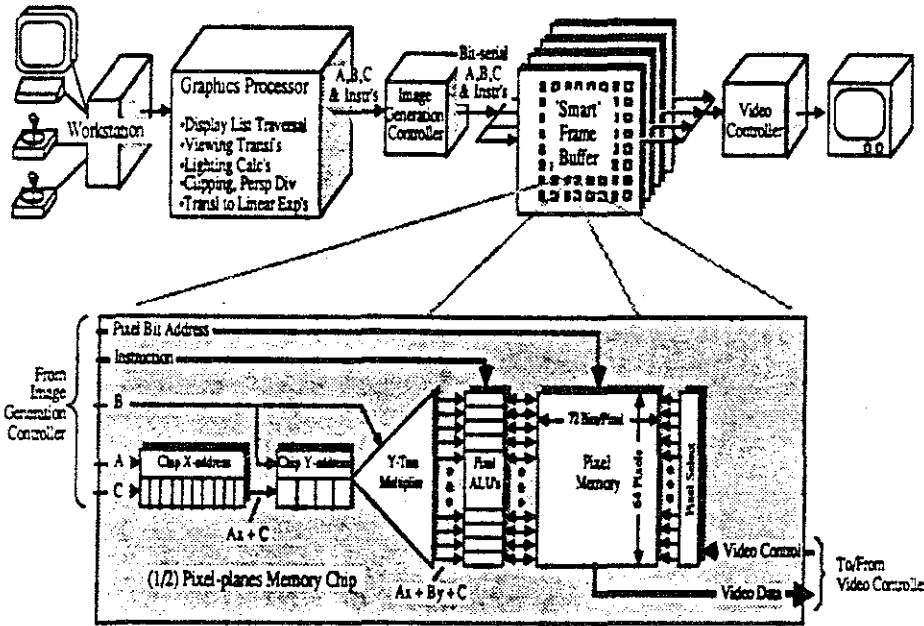
The rasterization time for an image is approximately the addition of the time needed to load all the triangle data and the time needed to send a screen-full of pixels through the pipeline. "In many normal cases, the rasterization time is independant of the size of the triangles". Each triangle is entered only once into the pipe and it takes 8 50ns-cycle per triangle. The pipe input rate for "blank" pixels is 50 ns/pixel.

The XTC uses 20 40-MFLOFS chips to be able to process 1.6 million triangles per second. The triangle and NVS pipes perform at 20 MHz and two of these pipes share the display region (left and right halves). Triangles are loaded at 200ns each (400ns for each half) and the whole region is rendered at 25 ns/pixel.

VI-Reference for the Triangle Processor:

[Deering 88] Michael Deering et al.- The Triangle Processor and Normal Vector Shader: a VLSI System for High Performance Graphics- in Proceedings of SIGGRAPH'88, Computer Graphics, vol.22, no. 4, August 88, pp 21-30.

4.4 Pixel-Planes 5



II From Pixel-planes 4 to Pixel-planes 5

In order to reach the rates required by a truly interactive graphics system, the designers first focused on the back-end bottleneck of the graphics pipeline: the image rendering process which is performed in the "smart frame-buffer".

The frame buffer is composed of custom logic-enhanced memory chips that can be programmed to perform most of the time-consuming pixel-oriented tasks at every pixel in parallel. Each pixel is provided with a minimal, though general processor, together with local memory to store pixel color, z-depth, and other pixel information.

figure 4.4-1 from [Fuchs 88b]

Pixel-planes 4 block diagram: the exploded view shows details of the custom processor-enhanced memory chip.

I- Overview

We will not give here another complete description of the Pixel-planes systems. Rather, we will briefly describe the basic principles and the performances of Pixel-planes 4 and Pixel planes 5, its successor.

The Pixel-planes systems use a conventional graphics pipeline organization (figure 4.4-1): display list traversal, geometric transformations, clipping and perspective, lighting, visibility calculations, hidden surface removal using a depth buffer algorithm, and shading. The novelty is in the "smart frame-buffer", which is also the core of the systems.

Pixel-planes 4 renders 39,000 Gouraud-shaded, z-buffered polygons per second on a 512x512 pixel full-color display.

Pixel-planes 5 is expected to render over 1 Million Phong-shaded, z-buffered polygons per second.

The front part of the system performs geometric transformations, clipping and lighting calculations, and then specifies the objects on the screen to the frame buffer in geometric, pixel-independent terms: a sequence of sets (A,B,C,instruction). Image primitives such as lines, polygons and spheres are each described by expressions and operations that are linear in screen space, that is, by coefficients A,B, and C such that the value desired at each pixel is $Ax+By+C$, where x,y is the pixel's location on the screen. The frame buffer memory chips generate the image from this description.

Pixel-planes 4 was completed in 1986. It was the first full-scale prototype Pixel-planes system. It consists of a host and a separate cabinet which contains the prototype custom hardware. The host is connected to the prototype through a DMA link. The prototype is composed of two parts: a Graphics Processor and a Frame Buffer.

The Graphics Processor (GP) is a floating-point uniprocessor implemented using the Weitek 8032 XL chip set. It runs at 8 Mhz and attains a peak performance of 16 Mflops (1 multiply-accumulate per cycle). The GP's interface to the Frame Buffer is a 1024 36-bit words FIFO memory. The GP performs the front-end operations mentioned above.

The Frame Buffer is made from custom VLSI processor-enhanced memories. It handles 512 x 512 pixels

and 72 bits per pixel. It consists of three parts: the array of custom VLSI processor-enhanced memory chips, the Image Generation Controller (IGC), and the Video Controller (VC).

Each custom chip in the array have three main parts (see figure 4.4-1): a conventional memory array that stores all pixel data for a 128-pixel column (72 bits per pixel), an array of 128 one-bit ALU's that carry out pixel local arithmetic and logical operations, and a multiplier tree, also called linear expression evaluator (LEE), that generates bilinear expressions $Ax+By+C$ in parallel for all pixels.

An array of 2048 of these chips forms the core of the massively parallel (512 x 512), bit-serial, SIMD machine.

The IGC provides the SIMD instruction for the pixel ALU's, the address into local pixel memory, and serializes the A,B,C coefficients into the suitable bit streams for the multiplier tree.

The video controller provides the control signals for the video output port of the enhanced memories and the video train. The video data come from 32 bits of pixel memory.

The Pixel-planes 4 system is a successful research vehicle, but it has several limitations:

- large amount of hardware often poorly utilized (e.g. when rendering small primitives);
- limited memory-space at each pixel (72 bits);
- no access to pixel data from the Graphics Processor or the host;
- insufficient front-end computation power.

Pixel-planes 5 addresses these limitations.

Pixel-planes 5 (figures 4.4-2 to 4.4-4) uses screen subdivision and multiple small rendering units in a modular, expandable architecture to address the problem of processor utilization. A full-size system can render more than 1 Million Phong-shaded polygons per second. Sufficient front-end power to sustain this performance is provided by an MIMD array of Graphics Processors. Pixel-planes 5's rendering units each are 5 times faster than Pixel-planes 4 and contain more memory per pixel, distributed in a memory hierarchy: 208 bits of fast local storage on the processor-enhanced memory chips, 4K bits of memory per pixel processor in a VRAM backing store. The frame buffer is separate from the rendering units; it refreshes normal and stereo images on a 1280 x 1024 72 Hz display. The machine's multiple processors communicate over a high-speed Ring Network.

Screen subdivision

In the new architecture, instead of one large computing surface of SIMD parallel processors operating on the entire screen space, there are one or more small SIMD engines, called Renderers, that operate on small, separate 128x128-pixel patches in a virtual pixel space. Virtual patches can be assigned on the fly to any actual patch of the display screen. The system can process in parallel graphics primitives that fall entirely within different patches on the screen.

The primitives handled in the transformation engines must be sorted into "bins" corresponding to each patch-sized region of the screen space. Primitives that fall into more than one region are placed into the bins corresponding to those regions. When a Renderer completes a patch, it transfers pixel color values into its backing store and discards all other pixel values. It is then assigned to the next patch to be processed.

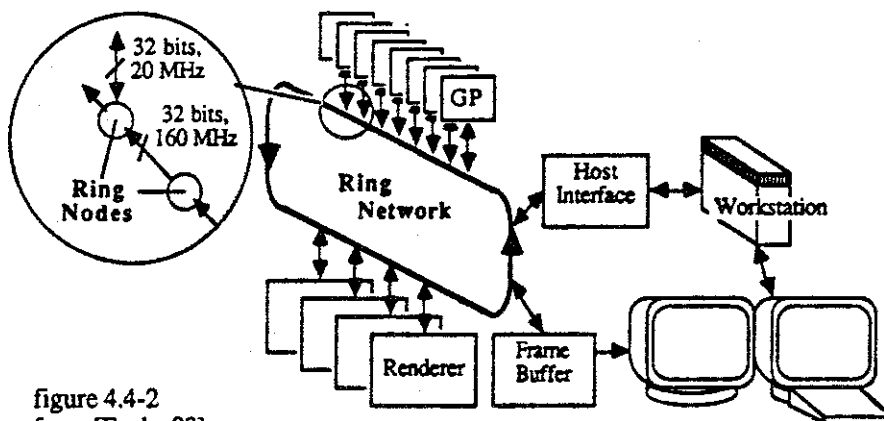


figure 4.4-2
from [Fuchs 89]
Overview of Pixel-planes 5 system: the Ring Network, Graphics Processors, Renderers, Host Interface, Frame buffer.

The figure 4.4-3 shows this process for a system configured with four Renderers. The 1280 x 1024 screen is divided into 80 128x128 patches which are processed in raster order. Renderers 1 to 4 are assigned initially to the first four patches. Then the next available Renderer is assigned to the next available patch.

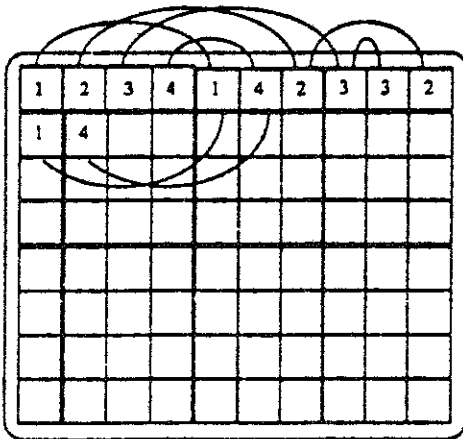


figure 4.4-3
from [Fuchs 89]
Rendering process for a Pixel-planes 5 system with 4 Renderers.

When the entire image has been rendered, each of the regions that have been stored in a Renderer's backing stores are transferred in a block to the double-buffered display memory in the Frame Buffer, from which the display is refreshed.

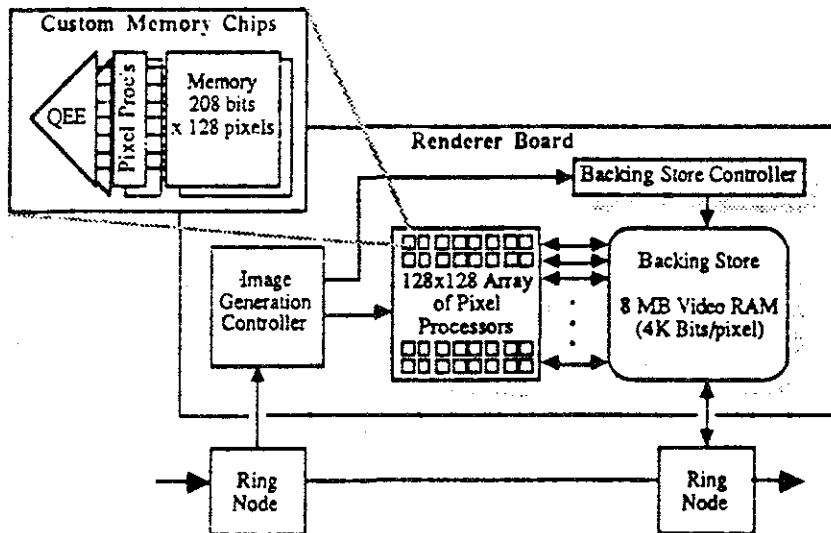


figure 4.4-4
from [Fuchs 89]
Block diagram of the Renderer.

Graphics Processors

The Graphics Processor is a high-performance math-oriented engine built using the Intel i860 microprocessor. The several hundred megaflops of front-end performance required in Pixel-planes 5 are achieved using a large number of MIMD-parallel GP's. A full-equipped system will consist of about 32 GP's.

Renderers (figure 4.4-4)

The design is based on a logic-enhanced memory chip using 1.6μ CMOS technology and operating at 40 Mhz bit-serial instruction rates. Each chip contains 256 pixel processing elements, each with 208 bits of static memory, and a quadratic expression evaluator (QEE) that produces the function $Ax+By+C+Dx^2+Exy+Fy^2$ simultaneously at each pixel. Quadratic expressions are very useful for rendering curved surfaces and for computing a spherical radiosity lighting model.

The pixel processor array is implemented in 64 custom chips, each with 2 columns of 128 pixel processors-with-memory and a quadratic expression evaluator.

In addition to the array of pixel processors, the Renderer also contains the backing store, built from VRAMs, tightly linked to the custom logic-enhanced memory chips.

It consists of an array of VRAMs each connected via its video port to one of the custom memory chips. The backing store memory is also available through the VRAM random I/O port to the rest of the system via the Ring Network. The Renderer uses this memory to save and retrieve pixel values, allowing context switching. A typical context switch takes about 0.4 msec.

A full equipped system will consist of 16 Renderers.

Ring Network

A single, flexible, very high performance network connects all parts of the Pixel-planes 5's multi-processor architecture: the Ring Network which has a total bandwidth of 16 million 32-bit words per second (160 MW/sec). Communications take place from Node to Node. Each Ring Node is composed of commercial MSI bus-oriented data parts and field-programmable controllers. Each client in the system has one or more of these Nodes, and each Node provides to the client a 20MW/sec. bi-directional port onto the Ring Network.

The network can support eight simultaneous messages, each at 20MW/sec.

At the rendering rates of 1M primitives per second, the GP's send object descriptions to the Renderers at up to 40 MW/sec. Simultaneously, pixel values must be moved from the Renderers to the Frame Buffer at to 40 MW/sec. in order to allow updating of a 1280 x 1024 frame bufer at more than 24 frames/sec.

These typical requirements are covered by the specifications of the Ring Network.

A low-level message-passing operating system, called Ring Operating System (ROS), has been developed for the ring devices. It provides device control routines and hardware-independent communication, and it controls the loading and initialization of programs and data.

III - Performances

Each GP in Pixel-planes 5 can process of order 30,000 Phong-shaded triangles per second. 32 GPs will then be necessary to meet the goal of 1 Million Phong-shaded triangles per second.

A single Renderer has a raw performance of about 150,000 Phong-shaded triangles per second. Nevertheless some primitives, lying over more than one patch have to be processed twice or more times, reducing somewhat the peak performance. Simulations predict an actual performance of around 100,000 triangles per second per Renderer. The configuration required to meet the goal will thus be around 10 Renderers.

IV-References for Pixel-planes systems

-[Eyles 87] - Eyles,J., J. Austin, H. Fuchs, T. Greer, J. Poulton, " Pixel-planes 4: A Summary", in Avances in Computer Graphics Hardware II, Proceedings of the Eurographics '87 Second Wokshop on Graphics Hardware.

-[Fuchs 89] - Fuchs, H., J. Poulton, J. Eyles, T. Greer, J. Goldfeather, D. Ellsworth, S. Molnar, G. Turk, L. Israel, " A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories" to appear in proceedings of SIGGRAPH'89, August 1989.

-[Fuchs 88a] - Fuchs, H., J. Poulton, " An Architecture for Advanced Airborne Graphics Systems", May, 1988.

-[Fuchs 88b] - Fuchs, H., J. Poulton, J. Eyles, T. Greer, "Coarse-grain and Fine-grain Parallelism in the Next Generation Pixel-planes Graphics System", in proc. of the Int'l Conf. and Exhibition on Prallel Processing for Computer Vision and Display, University of Leeds, UK, 12-15 January 1988.

-[Poulton 87] - Poulton, J, H. Fuchs, J. Austin, J. Eyles, T. Greer, " Building a 512x512 Pixel-planes System", in proc. of the 1987 Stanford Conference on Advanced Research in VLSI, MIT press, pp 57-71.

-[Poulton 85] - Poulton, J., H. Fuchs, J.Austin, J. Eyles, J. Heinecke, C-H Hsieh, J. Goldfeather, J. Hultquist, S. Spach, " Pixel-planes : Building a VLSI-based Graphics System", in proc. of the 1985 Chapel Hill Conference on VLSI, Rockville, MD, Computer Science Press, pp 35-60

5. CONCLUSION

This report has reviewed a number of different architectures used in High Performance Graphics designs.

Parallel processing is one of the main key used to increased performance. Both the geometry computation and the rendering process take advantage of this approach.

It is however noticeable that the number of processing elements devoted to geometry computation remain fairly limited in the real commercial machines.

The AT&T Pixel Machine has the largest number, with 18 elements in the transformation pipe. Most of the superworkstations has 1 to 4 general CPU's or equivalent (Stellar streams).

The design of the Rendering Processors is also very crucial for the effectiveness of overall system. Fast read and write accesses to the frame buffer, video memory distribution, parallel rendering of the primitives, performing image processing on the pixel data in the frame buffer, those are some of the issues that the Rendering Processors have to deal with.

Designers don't disclose very easily their tricks for this part of the machine. Parallel processing is also one of the key, but, as the computations are more specific, the design can be more dedicated and closely related to the actual algorithm. Hence the number of processing elements here can be much higher: 16 "toes" in the Stellar GS1000's "foot-print" engine, 20 Image Engines in the Silicon Graphics GTX architecture, 64 nodes in the AT&T Pixel Node array, 256 pixel processors in the IBM SAGE chip, an array of 128x128 pixel processors in Pixel-planes 5 Renderer.

The successful design of a real graphics system is a subtle balance of performance and generality in all the stages of the graphics pipeline.