Two Widely-Different Architectural Approaches to Computer Image Generation

H.W. Park, K.S. Eo*, D.L. Kim, B.K. Choi*, Y. Kim, and T. Alexander
Image Computing Systems Laboratory
Department of Electrical Engineering, FT-10
University of Washington
Seattle, WA 98195
* from the Samsung Advanced Institute of Technology, Korea

Abstract

Among several systems from the UWGSP (University of Washington Graphics System Processor) series, two recent architectures were designed for imaging and graphics. One of them, UWGSP3, has already been completed and is being used for selected applications, while the other one, UWGSP4, is being implemented now.

These two systems use parallel and pipelined architectures for high performance graphics operations. UWGSP3 uses only commercially available off-the-shelf chips, and consists of a TMS34020 graphics system processor and four TMS34082 floating point coprocessors that can be configured into pipelined or SIMD modes depending on the algorithm. UWGSP4, however, uses dedicated ASIC (Application Specific IC) chips for higher performance, and consists of two main computational parts: a parallel vector processor with 16 vector processing units, used mainly for image processing, and a graphics subsystem which utilizes a parallel pipelined architecture for image synthesis.

In this paper, the computer graphics aspects of both UWGSP3 and UWGSP4 will be described.

1 Introduction

Computer image generation, i.e., computer graphics, became very important and widespread in many applications during the last decade, and is expected to grow continuously in conjunction with image processing and multimedia in the 1990's [1]. This trend has been accelerated by technology development (such as VLSI technology, parallel and pipelined architectures, image display technology, and interactive graphical user interfaces), and proliferating application areas (particularly scientific visualization, animation, machine vision, simulation, medical and military area, and multimedia).

Among parallel and pipelined architectures, pipelining has been the predominant technique for computer graphics. Parallelism, however, can provide several advantages, such as reconfigurability according to different graphics algorithms, modularity, and so on. The problems to achieve high computational performance are load balancing between the computation stages in a pipelined structure and synchronization and data passing between processors in a parallel architecture. A combined architecture incorporating both parallelism and pipelining may provide better performance than a single architecture system if the above problems are solved. Some commercial graphics workstations use both parallelism and pipelining, for example, the AT&T Pixel Machine [2].

For high-speed graphics, the operations most frequently used in graphics should be supported by hardware. In particular, antialiasing, alpha blending, transparency, and texture mapping, as well as basic Gouraud shading and 3-D vector drawing, are common features which are implemented with hardware in high-end graphics systems.

The Image Computing Systems Laboratory (ICSL) at the University of Washington has been developing a series of imaging and graphics workstations, referred to as the UWGSP (University of Washington Graphics System Processor) series. Among the four systems (UWGSP1 to UWGSP4), UWGSP3 was completed in early 1990 [3], while UWGSP4 is currently under development [4]. Both UWGSP3 and UWGSP4 utilize parallel and pipelined architectures for image processing and computer graphics. UWGSP3 has a reconfigurable architecture for pipelined or parallel processing using only off-the-shelf programmable VLSI chips,

CH3046-0/91/0000/0042/$01.00 © 1991 IEEE

and is a low-cost and medium-performance imaging and graphics workstation, whereas UWGSP4 will be a medium cost and high performance system with a highly parallel and pipelined architecture employing dedicated ASIC chips. This paper describes the architecture of UWGSP3 and UWGSP4.

2 Configurable Parallel and Pipelined Architecture

2.1 Overview of UWGSP3

The UWGSP3 system, based on the NeXT computer platform, was completed in January 1990, and the UWGSP3-HI, a host-independent version which can work with any host computer via a simple interface card, has been subsequently developed at ICSL to accommodate various host computers without any modification of the hardware except the interface card [3]. Interfaces for the VME bus and the Micro Channel Architecture (MCA) bus have been completed.

Based on the TMS34020 graphics system processor, coupled with four TMS34082 floating point coprocessors on a daughter board, UWGSP3 delivers moderate performance for image and graphics processing, and provides the following features:

- $2k \times 2k \times 32$ bit (a total of 16 Mbytes) roamable video/frame memory.
- 32 bits per pixel that can be configured either as 24-bit true-color pixels with 8 overlay bits or four 8-bit pseudo-color pixels.
- $1,280 \times 1,024$ 60 Hz noninterlaced color display with 1:1 screen aspect ratio.
- 160 MFLOPS peak performance for high-speed integer and floating-point image processing and graphics functions.
- Hardware zoom, roam and cursor support.
- Fast cine display up to 60 frames/sec.
- 3 different color lookup tables (LUT) per color plane, which can be used simultaneously for region of interest (ROI) operations.
- Supports window-oriented user interface.
- Multiple UWGSP3s can be configured in the same host computer.

2.2 Hardware Architecture

As shown in Fig. 1, UWGSP3 centers around the Texas Instruments TMS34020 graphics system processor, while four TMS34082 floating point coprocessors are used for the computation-intensive operations, producing up to 160 MFLOPS of peak performance.

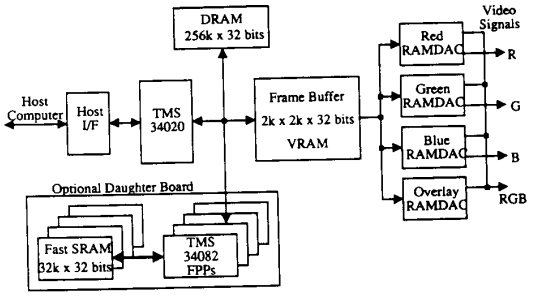

Figure 1: Block diagram of UWGSP3-HI

Each of the TMS34082 coprocessors has 128 kbytes of dedicated memory to store patches of data transferred from the video RAM (VRAM) or DRAM of the TMS34020, intermediate results, and program code. This coprocessor memory is implemented with fast static RAM so that memory access can be performed in a single cycle. After a patch of the data is transferred to the coprocessor memory, the TMS34020 signals the coprocessor to start executing its instructions. The data bus between the TMS34020 and the coprocessors can then be used to transfer another section of the data from the TMS34020 to a different coprocessor. If the same code is loaded into each of the coprocessor's memory prior to the execution, all four coprocessors perform the same processing on different data sets, after a slight delay caused by the data transfer. This is very similar to the SIMD (single instruction multiple data stream) mode of parallel processing. If different program codes are used in the coprocessors, MIMD (multiple instruction multiple data stream) processing can also be performed. In addition to the dedicated memory, a shared memory of 8 kbytes, implemented with 4-port memory chips, enables the coprocessors to communicate with each other. The coprocessor board can be configured as a 4-stage pipelined processor, in which the image data generated by the last stage are transferred back to the VRAM and displayed on the screen.

The system memory on the UWGSP3 board includes 16 Mbytes of VRAM for the frame buffer and 1 Mbyte of DRAM for program and data storage. The 16 Mbytes of VRAM are configured as $2k \times 2k \times 32$ bits, in which each 8-bit slice of the buffer constitutes four color planes: red, green, blue, and overlay. An 8-bit image can be stored in any of these four planes and displayed in gray scale or pseudo color, while a 24-bit true-color image can occupy the red, green, and

blue planes with 8 bits each. Since the display resolution ($1,280 \times 1,024$) is smaller than the frame buffer, the display window can be roamed around the frame buffer. For example, up to 64 $512 \times 512 \times 8$-bit images or 256 $256 \times 256 \times 8$-bit images can be stored in the frame buffer, and multiple frames can be displayed up to 60 frames/sec. The memory control signals for both the VRAM and DRAM are provided by the TMS34020, thereby greatly simplifying the logic circuits.

The video display subsystem consists of four Brooktree Bt460 RAMDACs and additional hardware circuitry. The RAMDACs receive pixel data from the VRAMs and convert them to analog signals of red, green, and blue. Each RAMDAC contains a 512×24 bit lookup table (LUT) that is used to generate 8-bit values each for red, green, and blue components. The red, green, and blue outputs of each RAMDAC are then summed together and used to drive the color monitor. To display 8-bit images, only one RAMDAC, corresponding to the color plane where the image is stored, is enabled and outputs the red, green, and blue components. When displaying 24-bit true-color images, each of the red, green, and blue RAMDACs drives one color component. In this case, the overlay RAMDAC can be enabled to superimpose text and annotations on top of the images. The RAMDACs also include the functions of hardware cursor and zoom, where the zoom factor can range from 1 to 8 in integer steps.

2.3 Software Architecture

UWGSP3 was designed for both image processing and graphics applications, and we have developed a collection of image processing and computer graphics functions; the 3-D computer graphics applications will be focused on here. There are three major categories of the UWGSP3 graphics software: a 3-D graphics rendering pipeline, graphical data management, and graphics libraries.

The 3-D graphics rendering pipeline software renders 3-D objects provided by higher-level functions (along with scene parameters, such as viewing location, light intensity, and light direction), and displays them on the screen from a perspective view. Several rendering modes are supported: wire-frame, coloring, flat shading, and Gouraud shading. The pipeline, which consists of four TMS34082 coprocessors, supports graphics operations such as coordinate transformations, back-face culling, clipping, rasterization, Z-buffer comparison, and shading. The pixel data generated by the pipeline is transferred back to the TMS34020, which stores it in the correct location of the frame buffer.

The graphical data management program runs on the host computer and utilizes many graphics functions implemented on UWGSP3 including the 3-D graphics rendering pipeline. Users can generate complex 3-D scenes by creating a hierarchical data structure of 3-D objects, and compiling the program on the host computer with the supplied graphics database management library. The library contains functions to support the opening and closing of a structure, adding and deleting an element to and from the structure, changing attributes of the structure and defining viewing and lighting parameters. When the program is executed, the structure defined by the user is transformed into a more compact form with the hierarchical information contained into individual data structures, and transferred to UWGSP3. The functions residing on UWGSP3 disassemble the received data into polygons, and each polygon is passed to the 3-D graphics pipeline until all of the polygons in the scene are drawn on the scene.

In the case of standard graphics libraries, the UNIX version of the TIGA (Texas Instruments Graphics Architecture) software has almost been completely implemented for supporting 2-D graphics capabilities, and Ithaca Software's HOOPS is also being ported to UWGSP3 to support 3-D graphics. A ray-tracing program has been implemented in UWGSP3, which generates very realistic images, including shadows, transparency through an object, and reflections from other objects. On traditional minicomputers, these programs require an exorbitantly high computing time, but, on UWGSP3, a high-resolution $1k \times 1k$ image can be generated in one minute.

2.4 Lessons Learned from UWGSP3

Data communication between the TMS34020 and the four TMS34082s became a bottleneck for the 3-D graphics pipeline execution as the graphics routines were optimized further. There is only one data bus between the TMS34020 and the four TMS34082s, as shown in Fig. 1. Therefore, the bus became saturated while the TMS34020 distributed polygon data to the four coprocessors and read back the final rendered data in order to draw the polygons on the screen, causing a bottleneck. If the last stage coprocessor could write the final computation results (synthesized image) directly into the frame buffer, instead of through the TMS34020, then better performance would be obtained for graphics pipelined operations since the input and output data flows of the pipeline could be

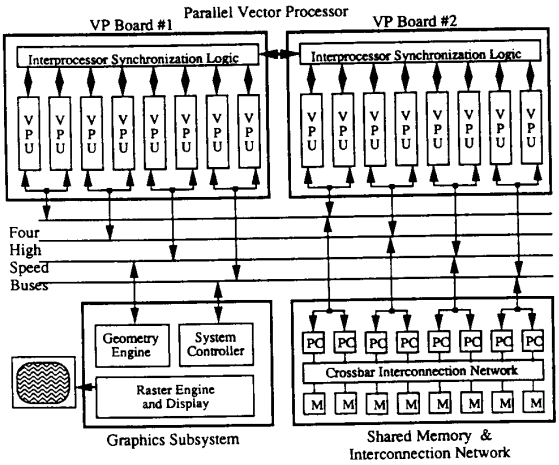

Figure 2: Block diagram of UWGSP4

separated, reducing the amount of traffic on the bus.

3 Advanced Parallel and Pipelined Architecture

3.1 Overall Architecture of UWGSP4

Utilizing the experience gained with UWGSP3, we have designed and are now implementing UWGSP4, which provides very high performance for image processing and computer graphics. It exploits both parallel and pipelined architectures. Most of the image processing functions are performed in a parallel fashion, whereas graphics operations are mainly pipelined (but also somewhat parallelized). Figure 2 shows the block diagram of the UWGSP4 system: it consists of a parallel vector processor, a graphics subsystem, and a shared memory and interconnection network [4].

The parallel vector processor is the main computational unit in UWGSP4, through which image processing operations and general arithmetic computations are performed. In particular, most image processing operations (such as image transforms and convolution) can be accelerated by parallel and vector processing, since an image can be divided into several sub-image blocks, and multiple processors can perform operations on the sub-image blocks in parallel. In addition, most low-level image processing functions require carrying out the same operation on a large amount of data. The parallel vector processor consists of 16 vector processing units which can operate independently or in synchronism. Interprocessor synchronization logic allows the 16 vector processing units to successfully operate in a parallel fashion.

Each vector processing unit consists of dual floating point processors, scalar and vector register files, an instruction and data cache, a master control ASIC, a pixel formatter unit (PFU) for pixel data handling, and a bus interface unit (BIU). The dual floating point processors operate in an alternating fashion with a 20 MHz two-phase clock, whereas the other units operate with a 40 MHz clock. The master control ASIC fetches and interprets instructions, and controls all parts of the vector processing unit so as to execute the desired arithmetic and logical operations. The PFU handles data conversion between floating point and integer values "on-the-fly". Most image pixel data consists of 8-bit or 16-bit unsigned integer values, whereas image computations should be performed in floating point for accuracy. The PFU relieves the floating point processors from having to perform pixel formatting, so that the processors can concentrate on the main computation without continually breaking up the pipeline. The peak computational performance of the parallel vector processor is 1,280 MFLOPS.

In order for the parallel vector processor to perform image processing at high speed, the shared memory and the interconnection network should provide high-speed data transfer, since most of the performance limitations in parallel computing systems, e.g., UWGSP3, arise from memory access bottlenecks and data transfer rate. The shared memory and the interconnection network provide a 1,280 Mbytes/sec data transfer rate. This rate can supply each vector processing unit with one-word (32 bits) data in every two clock cycles (50 nsec). This is relatively slow compared with, for example, a Cray X-MP supercomputer, where each vector processing unit can be supplied with two loads and one store of a 64-bit data word every 8.5 nsec [5]. However, the parallel vector processor is optimized for image processing, which requires less data access, and not optimized for general computations which may require high memory access bandwidth.

The shared memory consists of many memory modules in a 32-way interleaved fashion, with which the required 1,280 Mbytes/sec memory access rate can be achieved. Scalar data, a column or row vector, or 2-D array data with arbitrary strides can be accessed from the shared memory with a single command. This function is supported by the port controller and the memory controller ASICs. There are eight port controllers and eight memory controllers in the shared memory, all of which are implemented in CMOS gate-array ASICs. For communication between the port

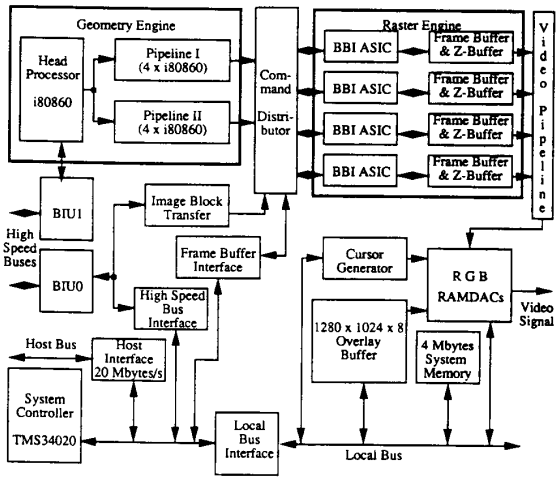

Figure 3: Block diagram of the graphics subsystem of UWGSP4

controllers and the memory controllers, an $8 \times 8 \times 40$-bit crossbar network is implemented. Each port controller is connected to two vector processing units through a high-speed bus, and two port controllers share one high-speed bus. In order to provide a 1,280 Mbytes/sec data bandwidth, four high-speed buses, operating with an 80 MHz speed, are used.

Since most graphics applications require operations using very small size matrices (3×3 or 4×4) in comparison with image processing, the parallel vector processor is not an optimum computational engine for graphics operations. Therefore, a separate graphics subsystem was designed to support fast generation of realistic 3-D graphics images, to display the synthesized images into the screen, and to interface with the host computer. The TMS34020 maintains the host interface, acts as a central controller for the UWGSP4 system, and controls a $1,280 \times 1,024 \times 8$-bit overlay buffer. The data transfer rate between the host computer and UWGSP4 through the host interface is 20 Mbytes/sec.

3.2 The Graphics Subsystem of UWGSP4

The graphics subsystem of UWGSP4 consists of three major parts: the geometry engine, the raster engine and the system controller, as shown in Fig. 3.

The graphics subsystem is connected to two high-speed buses through bus interface units (BIU0 and BIU1). To deliver host commands to the parallel vector processor or the geometry engine, the system controller stores the host commands in a specific region of the shared memory via BIU0. BIU0 is also used to transfer images from the shared memory to the frame buffer of the graphics subsystem at the rate of 40 Mpixels/sec. This transfer rate enables the system to display a $1k \times 1k$ image in 25 msec. BIU1, on the other hand, is used to transfer the geometric data of polygons and other drawing parameters to the geometry engine.

The key features of the geometry engine and raster engine are: a polyhedral object model, 3-D vector drawing, Gouraud shading, antialiased vector and polygon drawing, transparency, and hardware texture mapping.

3.2.1 The Parallel Pipelined Geometry Engine

The geometry engine is responsible for the front-end processing of the global Z-buffer algorithm used for 3-D computer image generation. The processes performed in the front-end engine include geometric transformation, back-face culling, illumination, clipping, projection, triangulation, span generation, etc., while the back-end processor, called the raster engine, is implemented with four bit-blit interpolator (BBI) ASIC chips. The main functions of the raster engine are scan conversion based on the Z-buffer algorithm and digital differential analyzer (DDA) calculations for line and polygon drawing.

The geometry engine consists of nine 40 MHz Intel i80860 RISC microprocessors operating in a parallel-pipelined fashion for maximum performance. The i80860 has an on-chip pipelined floating-point processor which enhances the performance of the system, since most of the geometric computations for computer graphics are floating-point intensive. Figure 4 shows its detailed architecture. The first i80860, called the head processor, communicates with the shared memory through a high-speed bus to access the actual polygons by traversing an object hierarchy. The head processor calculates the total amount of computational effort required for rendering each polygon and distributes it such that all pipeline stages have approximately equal loads. The head processor has an $8k \times 64$ SRAM as its local memory, to store its instruction code and data as well as the structural information of the object hierarchy (the physical data is stored in the shared memory).

The graphics engine is not limited to a specific graphics algorithm. Since each pipelined stage is provided by a general floating point processor (i80860), any new graphics algorithm can be programmed and applied to the geometry engine by changing the

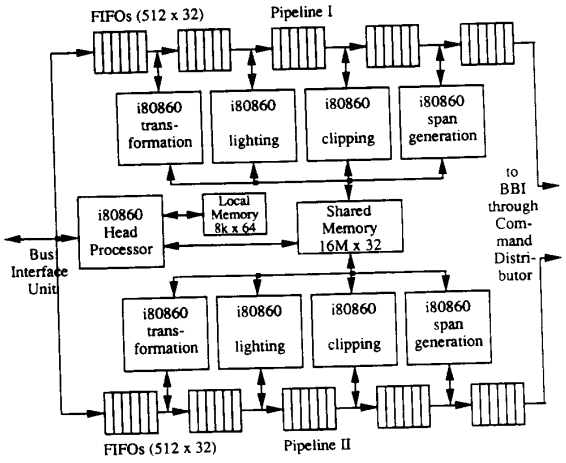

Figure 4: Block diagram of the parallel-pipelined geometry engine

firmware of the i80860s.

The head processor is followed by two identical pipelines. The head processor distributes polygons to the first stages of the two pipelines according to their requests. The number of pipelines required is determined by the processing speed of the head processor, the bandwidth of the high-speed bus, and the performance of the raster engine consisting of BBI ASIC chips. Our baseline system design has high flexibility and can be reconfigured to meet a wide variety of performance requirements.

In general, the loads of the four stages in the pipeline of the geometric engine are assigned as follows:

- Stage 1: Transformation of coordinates and vectors
- Stage 2: Light modeling (illumination)
- Stage 3: Clipping polygons against the six view facets
- Stage 4: Triangulation and span generation

The boundaries of load assignments, however, can be moved dynamically for load balancing. Each i80860 reads input data from the input FIFO (which is the output FIFO of its previous stage), processes it, and writes the results to its output FIFO. The geometry engine can sustain a performance of over 200,000 Gouraud-shaded 100-pixel polygons per second.

The hierarchical object definition method requires object flattening; in this process, the object hierarchy is traversed and each polygon is extracted. The architecture of the graphics subsystem supports this feature. Manipulation of the object hierarchy will be done by the head processor of the geometry engine. Calculation of the polygon surface normal and vertex surface normal (obtained by averaging the surface normals of the polygons containing the vertex) can be also performed in the head processor. These will make it easy to port standard 3-D graphics packages, e.g., PHIGS+ or HOOPS, into UWGSP4.

The polygon data are received from the host computer through the host interface, and are stored in the shared memory. In order to facilitate the hidden-surface removal process, which is one of the most time-consuming tasks in image synthesis, they need to be transformed into a new coordinate system, i.e., the "eye coordinate system." Such a coordinate transformation requires vector-matrix multiplications, which are well suited to the i80860s.

If all objects (polygons) in the scene are blocking (opaque), we can eliminate those objects which are back-facing with respect to the eye from the illumination and hidden surface removal processes, because they cannot contribute to the final synthesized image. This is called back-face culling. Since, on the average, about half of each object in the scene is back-facing with respect to the eye, the number of polygons will be reduced to half the original number after back-face culling. Back-face culling requires only a simple vector inner product between the polygon surface normal and the eye vector (whose direction is from the eye to the polygon), and it saves many unnecessary computations, resulting in an effective performance increase by nearly a factor of two.

It is very difficult to model the illumination of a natural object based on the real properties of light. A popular model used in computer graphics today is to compute the color of an object surface from the light source, surface characteristics, and its orientation with respect to the light source. The calculations are classified into three components such as the ambient, diffuse reflection, and specular reflection terms. In the case of a transparent object, an additional term, the transmittance, is added. This popular illumination model (including transparency) is used in the graphics subsystem.

Sophisticated smooth-shading techniques are necessary to render the smooth surface in the polyhedral object model. Gouraud shading has been most popular while Phong shading is more computationally expensive than Gouraud shading, but can generate more realistic images [6] [7]. However, since the quality of the images generated by Gouraud shading is acceptable in most engineering applications, and

Phong shading requires more complicated hardware, the Gouraud shading scheme is used in the graphics subsystem. And many objects in the scene may lie on the screen boundary or out of it. As these objects cannot contribute to the output image in the local illumination model, 3-D clipping is required to reduce the number of polygons to be rendered and to facilitate scan conversion.

The input to the BBI is in the form of spans of pixels which can be generated by slicing the polygon horizontally. These are often called scan line commands, or simply spans. A span consists of the X and Y coordinates of the starting point, a run length, the function values at the starting point, and their X derivatives. The BBI interpolates the scan line command by iteratively adding the X derivatives to the corresponding function values.

In the case of triangles, the X and Y derivatives of the function values are constant. Thus it is not necessary to update those derivatives at every scan line and every pixel, which results in a significant reduction of the computational cost. The output image of this scheme is not identical to that of real Gouraud shading, but the results are acceptable and indistinguishable from those of real Gouraud shading.

3.2.2 Raster Engine

The raster engine in the graphics subsystem of UWGSP4 consists of four CMOS standard cell-based ASICs, called bit-blit interpolators (BBI), and VRAMs comprising double frame buffers and a Z buffer.

As mentioned in the previous subsection, geometry transformation, clipping, lighting, and projection for computer image generation are performed by the geometry engine. The BBI's primary role is the rendering of the span data delivered by the geometry engine. The BBI can perform the following functions: polygon span drawing, 3-D line drawing, Gouraud shading, antialiasing, transparency, alpha blending, plane masking, bit plane and block image transfer, texture mapping, and multi-window control. Four BBIs operate in parallel to deliver a peak polygon rendering throughput of 200,000 Gouraud-shaded 100-pixel polygons per second.

There are seven instructions for the BBI: scan conversion of general spans, scan conversion of texture-mapped spans, general line drawing, color and Z-value fill, bit plane and block image transfer, mask register load, and screen refresh counter load. The BBI receives instructions from the geometry engine through the command distributor, interprets the instructions, and performs proper operations.

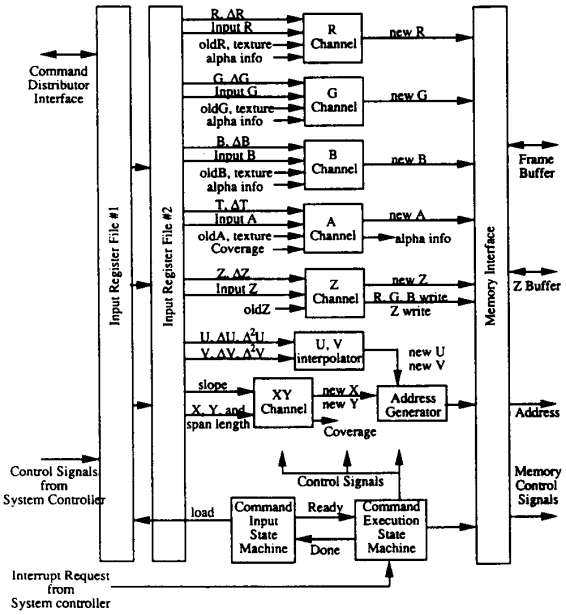

Figure 5: Block diagram of the BBI ASIC

Figure 5 shows the block diagram of the BBI. Each BBI has two separate input register files. While the BBI executes an instruction from one input register file, it also fetches the next instruction from the command distributor and stores it to the second input register file, effectively eliminating the instruction fetch time. Each BBI has eight channels, consisting of an interpolator and alpha blending unit to calculate new pixel values for red, green, and blue colors, an alpha value for transparency/coverage, a Z value for image depth, U and V values for texture mapping, and a Y value for 3-D vector drawing. A quadratic interpolation scheme is used for U and V, whereas the other values are linearly interpolated. The new pixel value is selected from four different values, i.e., the pixel data transferred from the command distributor, the interpolator output, the alpha blending unit output, and the masked values in each channel, according to the instruction. After the BBI computes a new pixel value, the Z channel compares the new Z value with the old Z value (stored in the Z buffer) to determine whether the new color and Z values should be written into the frame buffer and Z buffer. The BBI also contains address generators to generate row and column addresses for the frame buffer and Z buffer, and to select the drawing buffer or the refresh buffer in

the double buffer structure. The image data transfer speed from the shared memory to the frame buffer is 40 Mpixels/sec. Therefore, 40 consecutive $1k \times 1k$ images (e.g., cine angiogram) can be displayed every second. The BBI also generates memory control signals to control the VRAMs in the frame buffer and Z buffer. If any memory refresh or display refresh commands are received from the system controller, the BBI generates appropriate control signals for VRAMs.

The double frame buffers consist of two $2,048 \times 1,024 \times 32$-bit buffers built using VRAMs, with each pixel being 32 bits deep to support realistic animation with 24-bit true color. The remaining eight bits are used for storing alpha values which are used for transparency and antialiasing. Although each frame buffer is built with $2,048 \times 1,024$ pixels, the actual screen resolution is $1,280 \times 1,024$. Therefore, we have two remaining memory areas each of which has a dimension of $768 \times 1,024 \times 32$ bits. These memory areas are used for storing the image data for texture mapping. Since they are directly accessible from the BBIs, the texture mapping operation can be performed very fast. The Z buffer, whose size is $1,280 \times 1,024 \times 24$ bits, is implemented with VRAMs as well, which significantly speeds up the global Z buffer algorithm. Each pixel has a 16-bit offset Z value and an 8-bit base Z value. The image data stored in the frame buffer is transferred to the RAMDACs to be displayed on the screen.

4 Conclusions

Two different architectures have been designed for high performance imaging and computer graphics. The first architecture, UWGSP3, which consists of exclusively commercially-available processors (one graphics system processor and four floating point coprocessors), has structure that is reconfigurable according to the imaging and graphics algorithms being implemented. UWGSP3, which has moderate performance, is applicable to an image display and processing system for an electronic filmless PACS (Picture Archiving and Communications System) for radiology which requires a high display speed with moderate computing power [8].

The second architecture, UWGSP4, was designed with highly parallel and pipelined structures, and consists of two parts, performing imaging and graphics separately. The graphics part uses a combination of parallel and pipelined architectures to achieve a graphics performance of 200,000 Gouraud shaded 100-pixel polygons/sec and 250,000 3-D shaded 100-pixel lines/sec. Since the imaging and graphics performance of UWGSP4 is very high (e.g., simulation results show that less than 0.2 sec is required for a 2-D FFT of a $1k \times 1k$ image), it has wide application areas including scientific visualization, medical imaging, various military applications, animation and simulation. In particular, by integrating high performance imaging and graphics into one system, UWGSP4 meets the requirements of those application areas where both imaging and graphics must be supported in a single platform: for instance, volume rendering and ray tracing as applied to medical imaging (e.g., image reconstruction and bone detection from X-ray CT) to aid in planning radiation therapy and surgery by allowing the physician to visualize a 3-D model of the extracted target on the screen.

References

[1] J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics: Principles and Practice (second edition),* Addison-Wesley Publishing Co., 1990.

[2] M. Potmesil and E.M. Hoffert, "The Pixel Machine: A Parallel Image Computer," ACM Computer Graphics, Vol. 23(3), pp. 69-78, 1989.

[3] K.S. Mills, G.K. Wong, and Y. Kim, "A High Performance Floating-Point Image Computing Workstation for Medical Application," Proceedings of SPIE Medical Imaging IV, Vol. 1232, pp. 246-256, 1990.

[4] H.W. Park, T. Alexander, S.H. Moon, and Y. Kim, "A High Performance Parallel Computing System for Imaging and Graphics," Proceedings of IEEE Pacific Rim Conference, Victoria, Canada, May 9th-10th, pp. 223-226, 1991.

[5] T. Cheung and J.E. Smith, "A Simulation Study of the CRAY X-MP Memory System," IEEE Trans. on Computer, Vol. C-35(7), pp. 613-622, 1986.

[6] H. Gouraud, "Continuous Shading of Curved Surfaces," IEEE Trans. on Computers, Vol. C-20(6), pp. 623-629, 1971.

[7] B.T. Phong, "Illumination for Computer Generated Pictures," Communications ACM, Vol. 18(6), pp. 311-317, 1975.

[8] D.K. Yee, W. Lee, D.L. Kim, C.D. Haass, A.H. Rowberg, and Y. Kim, "RadGSP: A Medical Image Display and User Interface for UWGSP3," Proceedings of SPIE Medical Imaging V, Vol. 1444, pp. 292-305, 1991.