UWGSP4: an imaging and graphics superworkstation and its medical applications

J. M. Jong, H. W. Park, K. S. Eo, M. H. Kim, P. Zhang, and Y. Kim

Image Computing Systems Laboratory
Department of Electrical Engineering, FT-10
University of Washington
Seattle, WA 98195

ABSTRACT

UWGSP4 is configured with a parallel architecture for image processing and a pipelined architecture for computer graphics. The system's peak performance is 1,280 MFLOPS for image processing and over 200,000 Gouraud shaded 3-D polygons per second for graphics. The simulated sustained performance is about 50% of the peak performance in general image processing. Most of the 2-D image processing functions are efficiently vectorized and parallelized in UWGSP4. A performance of 770 MFLOPS in convolution and 440 MFLOPS in FFT is achieved. The real-time cine display, up to 32 frames of 1280 x 1024 pixels per second, is supported. In 3-D imaging, the update rate for the surface rendering is 10 frames of 20,000 polygons per second; the update rate for the volume rendering is 6 frames of 128 x 128 x 128 voxels per second. The system provides 1280 x 1024 x 32-bit double frame buffers and one 1280 x 1024 x 8-bit overlay buffer for supporting realistic animation, 24-bit true color, and text annotation. A 1280 x 1024-pixel, 66-Hz noninterlaced display screen with 1:1 aspect ratio can be windowed into the frame buffer for the display of any portion of the processed image or graphics.

1. INTRODUCTION

Imaging technology plays a vital role in modern medical applications, especially in radiology, radiation oncology, orthopedics, and neurosurgery, where a large number of images in digital form are generated, diagnosed, and utilized in various treatments and surgeries. With the development of a clinically effective and economically justifiable systems solution, PACS (Picture Archiving and Communications System) and digital image computing will be gradually accepted in everyday clinical use in the future. Although 2-D images have been traditionally used in the diagnosis, they do not optimally convey the 3-D nature of the involved anatomy or the full extent of the pathology. Thus, 3-D imaging is desired to assist the comprehension of complex 3-D structures such as the arterial system around the heart. To facilitate the 3-D imaging, both image processing and 3-D graphics technologies need to be supported in a single platform.

Several kinds of computer systems for either imaging or graphics have been developed during the last decade. Among them, the Titan Graphics Supercomputer [5], the Silicon Graphics Superworkstation [1] and the AT&T Pixel Machine [15] provide high polygon throughput by using dedicated graphics hardware; however, they cannot provide high imaging performance. On the other hand, the MPP [3], Connection Machine [16], and the CMU Warp [2] can achieve high image computing rates, but not high graphics performance. None of them are cost-effective alternatives in clinical environments, where integrated, high-performance 3-D graphics and imaging processing functions, e.g., computer screening of certain diseases and volume visualization of 3-D or 4-D image data sets, are required. The Pixel-Planes 5 [8, 20] of the University of North Carolina was designed with the same philosophy as UWGSP4, but the two architectures are dissimilar. The Pixel-Planes 5 consists of 16 MIMD scalar processors with local memory, 16 x 128 x 128 SIMD pixel processors for the 3-D graphics rendering, and an 8-channel ring network. The UWGSP4 has 16 MIMD vector processors closely coupled by a shared memory (8 modules each with 4-way interleaving and an 8 x 8 crossbar network), 9 scalar processors configured into two pipelines for 3-D graphics, four ASIC chips for the 3-D rendering, and four high-speed buses.

The UWGSP (University of Washington Graphics System Processor) series of imaging and graphics workstations have been developed at the University of Washington since 1986. The first two generations, UWGSP1 and UWGSP2, were low-cost and medium-performance imaging and graphics subsystems for an IBM PC/AT host. UWGSP1 consisted of a

0-8194-0805-0/92/$4.00

TMS34010 graphics system processor (GSP) closely coupled with a TMS32020 digital signal processor (DSP). UWGSP2 enhanced UWGSP1 by incorporating a floating-point processor (74ACT8837). UWGSP3 [11], developed at the beginning of 1990, consisted of a TMS34020 GSP and four TMS34082 floating-point graphics coprocessors that can be configured dynamically into a pipelined or SIMD (single instruction, multiple data streams) mode depending on the algorithm. Although the peak performance of UWGSP3 is quite respectable (160 MFLOPS), it can only sustain about 10 - 30% of this performance in general image processing operations due to the contention over the shared bus among the multiple processors.

UWGSP4 [14], which is the main topic of this paper, is designed to provide high performance in both imaging and graphics. In particular, our architecture focuses on achieving a high sustained computational rate, about 50% of the peak performance for image processing. The important features of UWGSP4 are listed as follows:

- Independent imaging and graphics subsystems: 1,280 MFLOPS peak performance for image processing plus 200,000 Gouraud shaded polygons per second for computer graphics
- Two 1280 x 1024 x 32-bit frame buffer: double frame buffer supporting realistic animation or cine display; 32 bits per pixel configured as 24 bits for true color and 8 bits for supporting transparency, antialiasing and alpha blending.
- 1280 x 1024 x 24-bit Z-buffer: accelerating the 3-D rendering
- Single 1280 x 1024 x 8-bit overlay buffer: supporting text annotation.
- 1280 x 1024, 66-Hz noninterlaced color display with 1:1 aspect ratio.
- Hardware zoom, roam, and cursor support
- RS/6000 host system
- Medium cost

2. HARDWARE ARCHITECTURE

As shown in Fig. 1, UWGSP4 consists of three subsystems: the vector processor subsystem, the graphics subsystem, and the shared memory subsystem. The four high-speed buses are the arteries among these subsystems, and all communications are performed through the shared memory. The whole system works as a back-end computer which connects to the host computer (RS/6000) through the VME bus. The graphics subsystem supports this host interface, the system control, the graphics processing, and the image display. The vector processor subsystem consists of up to 16 vector processor units (VPUs) which perform the imaging or general mathematical computation in synchrony or independently. Usually, every VPU in the subsystem handles the same size of sub-image. The 32-way interleaved shared memory is organized in eight modules, and connects to the four high-speed buses through a crossbar network. UWGSP4 has the same bandwidth, 1,280 Mbytes/sec, for the crossbar network, the shared memory, and the four high-speed buses; 1,280 MFLOPS peak performance for the vector processor subsystem, and 200,000 Gouraud shaded polygons per second for the graphics subsystem. The following sections describe each subsystem in detail.

2.1 Vector Processor Subsystem

The vector processor subsystem is the primary computation engine in the system, and is used mainly for imaging and general mathematical computations. As shown in Fig. 1, the complete vector processor subsystem consists of 16 vector processor units (VPUs) over two boards. Fig. 2 shows the block diagram of a single VPU. Each VPU consists of two floating-point processing units (FPU), a set of scalar and vector register files, a master control unit (MCU) ASIC chip for control and instruction issue, a pixel formatter unit (PFU) for pixel format conversion, a unified instruction and data cache, and a bus interface unit (BIU) for interface to the high-speed buses.

The BIU provides the signal conversion between the standard 40 MHz TTL-level interface to the VPUs and the 80 MHz BTL (Backplane Transceiver Logic)-level interface to the high-speed buses. Four VPUs are connected to a single high-speed bus in two pairs, as shown in Fig. 1. Two pairs of VPUs share the one high-speed bus in an alternating fashion. Each BIU has a bus arbiter that controls the arbitration between the VPUs which communicate on the same bus phase.

Two FPUs, implemented using 74ACT8847 CMOS floating-point processor chips from Texas Instruments (TI), are used for each VPU. The FPUs operate in an alternating fashion with a 20 MHz two-phase clock. Each floating-point processor possesses a full set of arithmetic and logic instructions, and can handle IEEE 754 standard single and double-precision floating-point operands as well as 32-bit integer data values. The arithmetic and logical unit (ALU) and the multiplier within the floating-point processor can operate independently or be used simultaneously for pipelined multiply-accumulates. The peak performance of a single floating-point processor is 40 MFLOPS, so that one VPU provides 80 MFLOPS peak performance.

The control of the VPU, including instruction fetch/issue and data cache handling, is the domain of the master control unit (MCU) ASIC. It fetches and interprets instructions, and controls the FPUs so as to execute the desired arithmetic and logical operations. The MCU is implemented using the 1 μm CMOS standard-cell technology, and has the complexity of more than 50,000 gates. It also performs all the control and sequencing necessary for proper operation of the VPU; both scalar and vector operations are supported.

A set of vector and scalar register files facilitate the fast execution of vector instructions and scalar (bookkeeping) calculations. Sixty-four four-ported 32-bit scalar registers are provided. During the scalar operation, the MCU manipulates the four ports to move data to and from the two FPUs and the data cache. For the vector operation, a set of three 2048-word vector register files (VRX, VRY, and VRZ) are provided to each VPU. The control ASIC is equipped with three vector address generators which can generate various addressing patterns for accessing VRX, VRY, and VRZ, respectively. According to these addressing patterns, the vector operands are fetched from VRX and VRY, fed to the FPUs, and the computational result is stored to VRZ from the FPUs. Before entering the vector operations, these three vector address generators need to be set up according to the addressing patterns of the algorithm.

In general, image pixel data consists of 8-bit unsigned integer values, whereas image processing is performed in floating point for computational accuracy. In UWGSP4, data conversion between integer and floating-point value is carried out by the pixel formatter unit (PFU) which is implemented with a field programmable gate array (FPGA). Due to the PFU, each VPU can provide a high computing rate without the burden of data conversion, thus breaking the smooth flow in the pipeline. The other function of the PFU is to transfer data from the vector register file VRZ to VRX/VRY, from VRZ to shared memory, or from shared memory to VRX/VRY.

2.2 Shared Memory Subsystem

In parallel computers with central shared memory, the sustained performance of the system is usually limited by the interconnection network and the memory bandwidth of the shared memory. Therefore, the configuration of the shared memory and interconnection network should be designed to minimize data access conflicts. In order to increase memory bandwidth and to match the relatively slow memory to the processor speed (the processor is generally an order of magnitude faster than the memory), a memory interleaving is used to achieve a 1,280 Mbytes/sec bandwidth. Fig. 3 shows the block diagram of the shared memory subsystem which consists of four bus interface units (BIUs), eight port controllers (PCs), an 8 x 8 x 40-bit crossbar network, eight memory controllers (MCs), and eight four-way (total 32-way) interleaved memory modules.

For the interconnection network between the parallel vector processor and the shared memory, a combination of four high-speed buses and a crossbar network is adopted. Each BIU interfaces one high-speed bus to two PCs. All shared memory components operate with a 40 MHz clock cycle, except the high-speed buses and BIUs using an 80 MHz clock cycle. Therefore, two PCs can communicate with one high-speed bus in an alternating fashion without any conflict. The BIUs interface the TTL-level logic in the shared memory to the BTL bus.

The PCs were designed using 22,000 gate 1 μm CMOS gate-array ASICs. The PCs translate memory access commands from the VPUs into simple commands which can be executed by the MCs, and control the crosspoint switches in conjunction with the MCs. The VPUs can access a scalar data item, a column or row vector, or 2-D array data with a single command.

The crossbar network provides a separate path between any PC and any MC at all times, so that eight PCs can communicate with eight MCs simultaneously. The crossbar network is implemented with discrete Advanced Schottky TTL transceiver chips.

The MC generates the physical address of each data word from the command received from the PC, and was designed as an 12,000 gate 1 µm CMOS gate-array ASIC. Each MC controls four interleaved DRAM memory modules, and accesses vector data from the memory modules at an access rate of 160 Mbytes/sec. Also, DRAM refresh is carried out by the MCs. In order to accommodate multiple memory chips of different capacities, the memory controller ASIC was designed to be able to support 1-Mbit, 4-Mbit, or 16-Mbit DRAMs.

The four memory modules connected to each MC provide a 25 ns access time in the case of row vector data, which can utilize the four-way interleaving and page-mode access capability of the DRAM modules. Memory depth is 32 bits. All memory accesses are word-based, but any VPU can selectively write to any specific byte within a word by using byte mask. Since the shared memory has eight MCs, each of which controls four interleaved memory modules, a total of 32 interleaved memory modules are supported, which provides a maximum memory space of 1 Gbyte (256 Mwords).

2.3 Graphics Subsystem

The graphics subsystem is the primary agent for maintaining and drawing the image, and also for generating realistically-shaded 3-D images from scene descriptions. As shown in Fig. 4, the graphics subsystem consists of two independent polygon processing pipelines, four bit-blit interpolator (BBI) ASICs, a Z-buffer and a double-buffered frame buffer, a TI TMS34020-based graphics processor and system controller, an overlay buffer, a cursor generator, RAMDACs and a host interface.

The two polygon processing pipelines are responsible for the front-end processing for three-dimensional computer image synthesis. The operations provided by the pipelines include geometric transformation, back-face culling, illumination, clipping, projection and span generation. There are nine Intel 80860 CPUs operating in a parallel-pipelined fashion. One of the nine CPUs, called the head processor, accesses the shared memory to extract the actual polygons by traversing an object hierarchy, and distributes polygon processing jobs to the two pipelines. Each pipeline is composed of four CPUs (corresponding to four pipeline stages). The head processor dynamically assigns polygon rendering jobs to each pipeline stage in such a way as to balance the load between the eight CPUs of two pipelines. The computation results of these pipelines are fed to four BBI ASICs, where scan conversion is performed in conjunction with hidden surface removal.

The BBI ASICs carry out image drawing to the screen (frame buffer), using the span data computed by the polygon processing pipelines. The four ASICs, each of which is a 51,000 gate 1 µm CMOS standard-cell IC, are capable of drawing and filling image pixels at the rate of 40 million pixels per second. They also control the memory in the frame buffer and Z-buffer, and carry out screen refresh functions under the supervision of the system controller. Another feature incorporated into the BBI chips is the ability to support a multi-windowed display with minimal overhead based on an extended Z-buffer concept. The combination of the polygon processing pipelines and four BBIs is capable of delivering over 200,000 100-pixel polygons per second with Gouraud shading. Other special functions which are provided by the BBIs are transparency, antialiasing, texture mapping, and Z-buffer handling.

The double frame buffer consists of two 1280 x 1024 pixel buffers by using video RAMs (VRAMs) with each pixel being 32-bit deep to support 24-bit true color and realistic animation. The other eight bits in the 32 bits of the frame buffer are used for supporting transparency, antialiasing and alpha blending. The Z-buffer, whose size is 1280 x 1024 x 24-bit, is implemented with VRAMs as well to significantly speed up 3-D rendering algorithms. The image data stored in the frame buffer is serially transferred to the RAMDACs to be displayed on the monitor.

The image block transfer unit (BLITTER) transfers image data from the shared memory to the frame buffer at the speed of 160 Mbytes/sec which corresponds to 32 frames of 1280 x 1024 color pixels per second. This serves as the path for the images processed by the vector processor subsystem going to the display in real time.

The system controller of the TMS34020 maintains the host interface, acts as a central controller for the UWGSP4 system, and controls the 1280 x 1024 x 8-bit overlay buffer. 4 Mbytes of local memory is provided for storing programs and data associated with the system controller.

All communications between UWGSP4 and the host computer are performed through the VME host interface in the graphics subsystem; the data transfer rate is 20 Mbytes/sec. The graphics subsystem occupies two high-speed buses to store and load data to and from the shared memory. One interface with a high-speed bus is used to load graphics data from the shared memory to the polygon processing pipelines. The other interface is used to transfer the image data, program and any control information between the shared memory and the host or the graphics subsystem.

3. SOFTWARE STRUCTURE

The overall software structure of the UWGSP4 is shown in Fig. 5. There are five categories of software: the host software, the executive-based programs on the TMS34020, the image processing library on the vector processor subsystem, the graphics library on the polygon processing pipelines, and system-support software.

The host software is based on the Unix operating system and X Window. We have developed several medical application programs and user interfaces, such as RadGSP [18], prototype PACS workstation [19], and VIVA [4]. The main UWGSP3 user interface [11] developed using NextStep can also be ported to any host workstation which supports NextStep. The only new development in host software is the communication driver between the host computer and TMS34020 executive.

The executive which is written in C with a few lines of TMS34020 assembly code is a compact multiprocess operating system. The window server process and some system server processes, such as host communication process, vector processor service process, and graphics service process, always reside in this multiprocess environment. The window server process supports 2-D graphics, so it manages the RAMDAC, BLITTER, overlay buffer, cursor generator, and BBI. The host communication process handles the host interface; the vector processor service process and graphics service process coordinate general image processing and 3-D graphics tasks, so they deal with the vector processor subsystem and polygon processing pipelines, respectively.

The coordination among TMS34020 and vector processor units (VPUs) is achieved through a task queue in the shared memory. The TMS34020 stores tasks, e.g., FFT on image A, to the task queue. All 16 VPUs are synchronized in accessing the task queue by circulating a token. Only the VPU which gets the token is allowed to access the task queue. The token will be released after accessing the task queue, thus it continues to circle. The VPU which gets the FFT task will split image A into sub-images (child-tasks), store these child-tasks to the task queue, and put the original FFT task (parent-task) to sleep. Then every VPU can get a child-task, and the FFT is executed in parallel. The VPU which finishes the last child-task need to wake up the parent-task. Because the VPUs need to dynamically allocate shared memory more often than the TMS34020 does, the shared memory is managed by VPUs. The image processing library in VPUs is implemented mainly in C, with some speed-critical parts written in vector microcode and scalar assembly code.

The typical scanline-based 3-D graphics algorithm can be divided into four pipeline stages: geometric transformation, illumination, clipping, and span generation. The execution times of these four stages are different. We have mapped these four stages of the "software" pipeline to the four stages of the "hardware" pipeline (the polygon processing pipelines of Intel 80860's) and achieve load-balance among the four stages of the hardware pipeline. For a given number of polygons, the illumination stage usually takes much more time than the geometric transformation and clipping stages, respectively. The execution time of the span generation stage depends on the number of polygons as well as the shape and size of polygons. The shape and size of polygons vary with applications, thus we did not consider the span generation in the optimization of load balancing. It is mapped to the last stage of the hardware pipeline. The geometric transformation and clipping are mapped to the first and third stages, respectively. The illumination operation is mapped to the first, second, and third stages by distributing the light sources and vertices among them.

Since the VPUs are custom-designed processors, a VPU simulator with X Window user interface was developed on the SUN workstation. This simulator serves as a tool for program testing and performance estimation. We adapted the GNU C compiler for the VPUs; an assembler was also developed.

4. APPLICATIONS AND PERFORMANCE

The integration of balanced image processing and 3-D graphics in a single platform enables the UWGSP4 to be applied in a wide range of medical imaging applications. Among them, 2-D imaging and cine display have been developed. Our initial investigations also show that UWGSP4 is a good candidate for 3-D imaging (volume rendering and surface rendering).

4.1 2-D imaging

The 2-D images are usually generated from traditional X-ray projections or from the cross sections of 3-D data of various acquisition modalities, such as computed tomography (CT), magnetic resonance imaging (MRI), or positron emission tomography (PET). It has the advantage of presenting all information pertaining to the structure without obscuration. Currently, most of 2-D images are still printed onto films and viewed on the light boxes. Although this conventional approach is much less expensive in hardware, the management of a vast amount of films involves substantial manpower. Also, the films and diagnostic reports needed for rapid and effective emergency intervention may not be available in time. Thus, PACS which provides electronic image database and transmission is a potential solution to improving patient care and reducing treatment costs.

The 2-D imaging performance of UWGS4 is listed in Table 1 according to the simulations on 1024 x 1024 images. Because most 2-D image processing operations are easily parallelized by dividing the image and need to heavily access the shared memory, the operations are performed on the vector processor subsystem. Any algorithm that accesses data in a regular address pattern can be vectorized. Irregular address patterns may result from either comparison/branch operations or indirect address references, such as lookup table. Except for the 2nd-order warp, median filter, and adaptive histogram equalization, all the operations listed in Table 1 are vectorized.

The 2nd-order warp consists of two phases: geometrical transformation and bilinear interpolation. The geometrical transformation maps each output pixel location back to its corresponding input pixel location and only involves straightforward quadratic polynomial evaluation, thus it can be easily vectorized. However, the bilinear interpolation cannot be vectorized, because the input pixels for the interpolation are determined by the geometrical transformation and indirect address references are required to access the input pixels. The adaptive histogram equalization is not easily vectorizable either in updating the histogram table due to indirect address references. The large amount of comparison/branch operations in the median filter prohibit the vectorization.

The convolution operation achieves 770 MFLOPS by taking advantage of the dual-operation mode (simultaneous multiplication and addition) of the TI 74ACT8847 in performing the sum of products. The unsharp masking is also executed very efficiently, because of the efficient implementation of the convolution. The FFT is implemented using the row and column method. For a 1024 x 1024 image, there are 10 stages of "butterfly" operations for each row/column. The regular patterns of butterfly operations enable the FFT being vectorized to sustain 440 MFLOPS. The window/level operation only involves a linear mapping and two thresholdings, so it is fully vectorizable. The 3-pass shearing algorithm [13] is used to implement the rotation operation. The 90-degree rotation only involves the movement of vector data; the image is read in by the vector processors from the shared memory in row and then written back in column.

4.2 Cine display

To provide a smooth and continuous process for examining how structures change over time, from slice to slice, or to create the animation of a moving organ such as the heart, the cine display needs to be at least at the speed of 10 frames/second. The cine display is easily realized by using the double frame buffer. Two frame buffers work alternately; when one supplies the data to the RAMDAC for the display, the other is updated by either the image block transfer unit (BLITTER) or the polygon processing pipelines and BBIs. The BLITTER can transfer 32 frames of 1280 x 1024 pixels per second from the shared memory, and the polygon processing pipelines and four BBIs can render over 200,000 Z-buffered, Gouraud shaded 100-pixel polygons or 250,000 3-D shaded 100-pixel lines per second. In comparison, the Silicon Graphics 4D/240GTX system provides about 100,000 Z-buffered, lighted quadrilaterials per second and the Titan Supercomputer can render up to 200,000 Gouraud shaded triangles per second.

4.3 3-D imaging

The 3-D imaging deals with techniques for extracting 3-D information from the given image data and projecting such information on a 2-D display. Most techniques fall into two broad categories of surface rendering and volume rendering. Surface rendering first forms the surface of the structure and renders the surface on a 2-D display. In volume rendering, a color and partial opacity is assigned to each voxel, and the images are formed by blending together voxels projecting to the same pixel on the picture plane.

A common approach in surface rendering applies edge tracking or thresholding to each 2-D slice to get a set of contours which define features of interest. Then, a mesh of polygons can be constructed connecting the contours on adjacent slices [7, 9]. Another surface technique is to produce a surface of structure as a set of voxels by a surface tracking method [17]. Then, the voxel faces are handled as polygons. Both methods are not fully automatic; human intervention is required during the segmentation or polygon generation. All of these operations can be performed on the vector processor subsystem in scalar mode. Our major goal is the interactive display of the 3-D image after the database containing the polygon information has been established. Utilizing the scanline-based 3-D graphics method, the graphics subsystem can render 10 frames of 20,000 polygons per second.

The volume rendering of CT data can be roughly divided into five steps: volume formation, classification, surface extraction, shading, and compositing [6, 10, 12]. The detailed implementation of each step varies with applications and individual preference. In the volume formation, cubic voxels are usually formed by performing linear or trilinear interpolation. Higher-order interpolations can be used at the expense of longer execution time; the bicubic interpolation was used in [12]. In the implementation of Drebin et al. [6], the classification is the process that converts every voxel from a CT number to a color and opacity, i.e., an R, G, B, and alpha four-tuple, by a piecewise linear mapping (lookup table). In the surface extraction, each material, i.e., air, fat, tissue, or bone, is assigned a refractive index number. Then, the gradient of every voxel is computed by the first-order central difference on the index numbers. The surface normal and strength are derived from the gradient. Levoy [10] directly used the CT number to compute the surface gradient, then both the CT number and gradient are used as the indices to a lookup table to find the opacity. To provide a satisfactory perception of smooth surfaces, the Phong-based shading models are applied to every voxel. Finally, parallel rays are cast into the volume from the observer's eyepoint, and all the 2-D slices perpendicular to the rays are composited by merge-over operations to form the final image [10]. Since the rays usually do not hit the voxel centers, the trilinear interpolation is applied to reduce the artifacts. Another approach [6, 12] is the geometric transformation and resampling of the volume to the viewing orientation, which performs the compositing without the interpolation.

A fast image update is essential to the interactive visualization and rotation sequences required in the clinical diagnosis. Basically, this involves the change of the observer eyepoint, thus only the shading and compositing need to be recomputed. A Levoy's implementation [10] on 113 slices of 256 x 256 samples of a cadaver on a Sun 4/280 with 32 Mbytes of main memory took 20 minutes for the shading and classification, and another 20 minutes for compositing each image. The compositing was optimized in [20], which includes the octtree algorithm to skip the transparent regions, the alpha cutoff to terminate the rays when they attenuate to opacity, and the adaptive ray casting. Under this optimized algorithm, the computation time of the compositing is data-dependent. It takes more time to render a human torso as a semi-transparent gel than to render a skull surface, because the rays are cast deeper in the former case. The shading is also expedited by reducing each component of the surface normal to 6-bit precision, and each voxel was shaded by referencing a shading table. By taking advantage of the variations above, a recent implementation on the Pixel-Planes 5 achieved an update rate of 15 frames/sec on a 128 x 128 x 128 data set.

The algorithm simulated on the UWGSP4 is based on [6] with the shading and compositing models adapted from [12] and [10], respectively. Since most of the operations are natural extensions of 2-D image processing, it is more efficient to execute all these steps on the vector processor subsystem. The surface extraction and shading are easily vectorized, and the volume formation, classification, and compositing are computed in scalar mode because of the interpolation and lookup table operations. Each component of the surface normal and the four-tuple: R, G, B, and alpha, is 8-bit wide in the shared memory. All the computations are 32-bit floating-point operations. The Pixel Format Unit (PFU) performs the conversion between the 8-bit integer and 32-bit floating-point format. A 128 x 128 x 128 data set projected onto a 512 x 512 image was used to estimate the performance. The shading and compositing take about 350 msec and 150 msec, respectively. Thus, the image update rate is about two frames/sec. If we skip the shading and only perform the compositing for each frame, the image update rate is about 6 frames/sec. This is only our preliminary attempt at volume rendering. To increase the speed and maintain reasonable image quality, we are fine-tuning the above algorithms and researching other algorithms, such as Fourier projection slice theorem.

5. CONCLUDING REMARKS

UWGSP4 was designed for high performance of image processing and 3-D graphics applications. Our goal is to achieve a peak imaging performance of 1,280 MFLOPS and a graphics rate of over 200,000 polygons/sec with Gouraud shading, and also to support a high sustained performance of about 50% of the peak performance on imaging.

High performance results from the innovative vector address generation, long vector registers, powerful floating-point processors, efficient pixel-format conversion, high-speed buses, and high-bandwidth shared memory. Using these innovative features, UWGSP4 can achieve 770 MFLOPS in 2-D convolution and 440 MFLOPS in 2-D FFT. The double frame buffer supports the smooth real-time cine display up to 32 frames of 1280 x 1024 pixels per second. In 3-D surface rendering, the polygon processing pipelines and rendering ASIC chips deliver an update rate of 10 frames of 20,000 polygons per second. Our first attempt at 3-D volume rendering also achieves 6 frames/sec on a 128 x 128 x 128 3-D data set.

6. REFERENCES

[1] K. Akeley, "The Silicon Graphics 4D/240GTX Superworkstation," *IEEE Computer Graphics & Applications*, Vol. 9, pp. 71-83, July 1989.

[2] M. Annaratone, E. Arnould, T. Gross, H. T. Kung, M. Lam, O. Menzilcioglu, and J. A. Webb, "The Warp Computer: Architecture, Implementation, and Performance," *IEEE Trans. on Computer*, Vol. C-36, pp. 1523-1538, 1987.

[3] K. E. Batcher, "Design of a Massively Parallel Processor," *IEEE Trans. on Computer*, Vol. C-29, pp. 836-840, 1980.

[4] J. J. Birchman, S. L. Tanimoto, A. H. Rowberg, H. S. Choi, and Y. Kim, "Applying a Visual Language for Image Processing as a Graphical Teaching Tool in Medical Imaging," *SPIE Medical Imaging VI*, Vol. 1653, in press, 1992.

[5] B. S. Borden, "Graphics Processing on a Graphics Supercomputer," *IEEE Computer Graphics & Applications*, Vol. 9, pp. 56-62, July 1989.

[6] R. Drebin, L. Carpenter, and P. Hanrahan, "Volume Rendering," Proceedings of SIGGRAPH '88, *Computer Graphics*, Vol. 22, pp. 65-74, August 1988.

[7] H. Fuchs. Z. M. Kedem, and S. P. Uselton, "Optimal Surface Reconstruction from Planar Contours," *Comm. ACM*, Vol. 20, pp. 693-702, 1977.

[8] H. Fuchs, J. Poulton, J Eyles, T. Greer, J Goldfeather, D. Ellsworth, S. Molnar, G. Turk, B. Tebbs, and L. Israel, "Pixel-Planes 5: A Heterogenous Multiprocessor Graphics System Using Processor-Enhanced Memories," Proceedings of SIGGRAPH '89, *Computer Graphics*, Vol. 23, pp. 79-88, July 1989.

[9] S. Ganapathy and T. G. Dennehy, "A New General Triangulation Method for Planar Contours," Proceedings of SIGGRAPH '82, *Computer Graphics*, Vol. 16, pp. 69-75, 1982.

[10] M. Levoy, "Display of Surfaces from Volume Data," *IEEE Computer Graphics & Applications*, Vol. 10, pp. 29-37, May 1988.

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

[12] D. R. Ney, E. K. Fishman, D. Magid, and R. A. Drebin, "Volumetric Rendering of Computed Tomography Data: Principles and Techniques", *IEEE Computer Graphics & Applications*, Vol. 10, pp. 24-32, March 1990.

[13] A. W. Paeth, "A Fast Algorithm for General Raster Rotation," *Graphics Interface '86*, pp. 77-81, May 1986.

[14] H. W. Park, T. Alexander, K. S. Eo, and Y. Kim, "UWGSP4: Merging Parallel and Pipelined Architecture for Imaging and Graphics," *International Conference on Parallel Processing*, pp. 399-403, August 1991.

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

[16] L. W. Tucker and G. G. Robertson, "Architecture and Applications of the Connection Machine," *IEEE Computer*, Vol. 21(8), pp. 26-38, 1988.

[17] J. K. Udupa and G. T. Herman, *3D Imaging in Medicine*, CRC Press, 1991

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

[19] D. K. Yee, D. R. Haynor, H. S. Choi, S. W. Milton, and Y. Kim, "Development of a Prototypical PACS Workstation Based on the IBM RS6000 and the X Window System," *SPIE Medical Imaging VI*, Vol. 1653, in press, 1992.

[20] T. S. Yoo, U. Neumann, H. Fuchs, S. M. Pizer, T. Cullip, J. Rhoades, and R. Whitaker, "Achieving Direct Volume Visualization with Interactive Semantic Region Selection," *Proceedings of Visualization '91*, pp. 58-65, October 1991.

Operation	Time
Window/Level	10 msec
90-degree Rotation	8 msec
Rotation	27 msec
5 x 5 Convolution	68 msec (770 MFLOPS)
Unsharp Masking	76 msec
2nd-order Warp (bilinear interpolation)	160 msec
Fast Fourier Transform	170 msec (440 MFLOPS)
Median Filter	1 sec
Adaptive Histogram Equalization (contextual region size = 128 x128) (12 bits/pixel)	30 sec

Table 1. 2-D image operations on a 1024 x 1024 image (8-bit per pixel except otherwise specified)

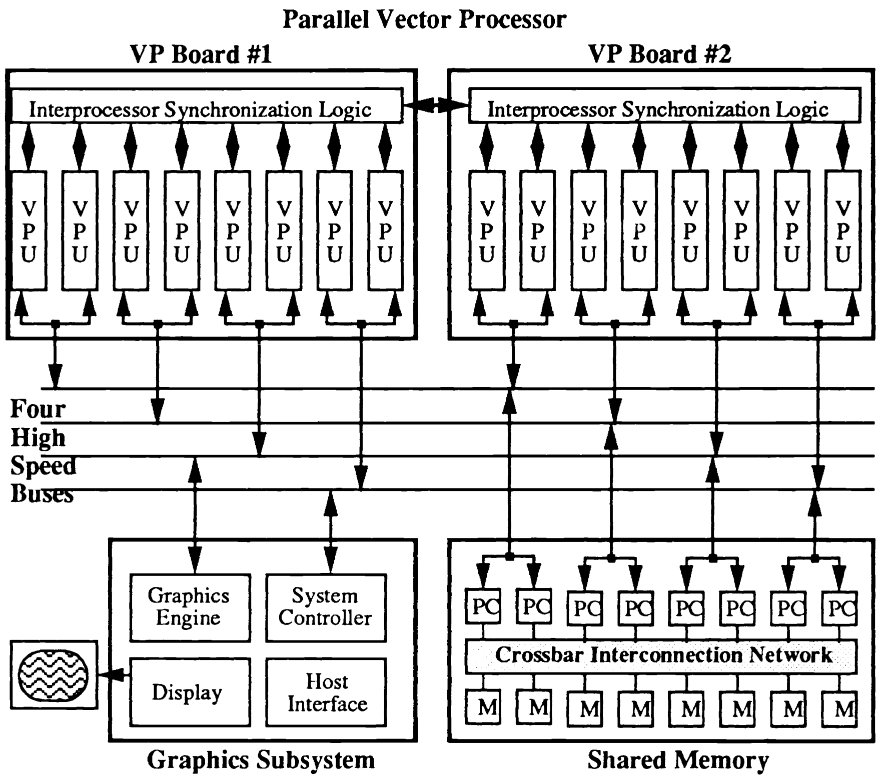

Fig. 1 Overall block diagram of UWGSP4

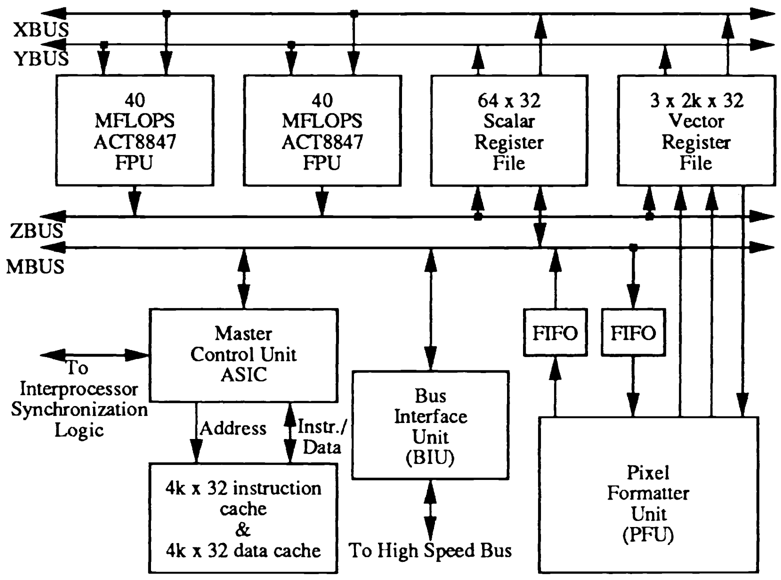
XBUS
YBUS
40 MFLOPS ACT8847 FPU
40 MFLOPS ACT8847 FPU
64 x 32 Scalar Register File
3 x 2k x 32 Vector Register File
ZBUS
MBUS
Master Control Unit ASIC
To Interprocessor Synchronization Logic
Address
Instr./ Data
4k x 32 instruction cache & 4k x 32 data cache
Bus Interface Unit (BIU)
To High Speed Bus
FIFO
FIFO
Pixel Formatter Unit (PFU)

Fig. 2 Block diagram of a single vector processing unit

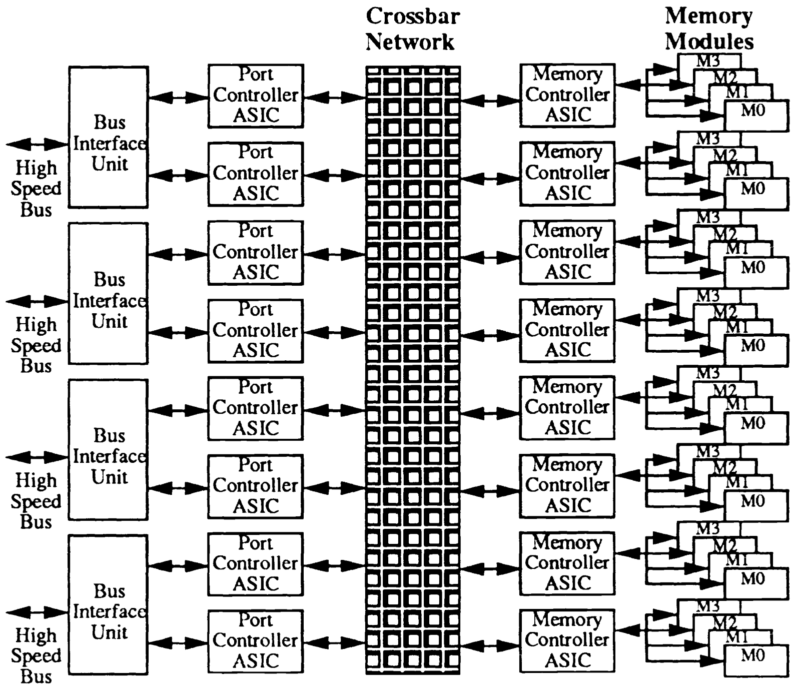

Fig. 3 Block diagram of the shared memory and crossbar network

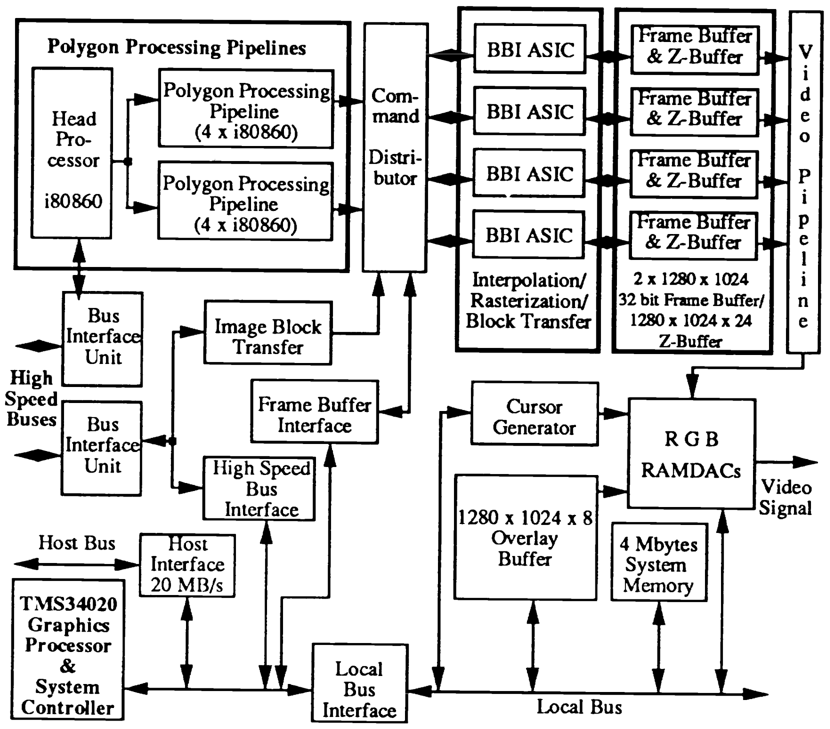

Fig. 4 Block diagram of the graphics subsystem

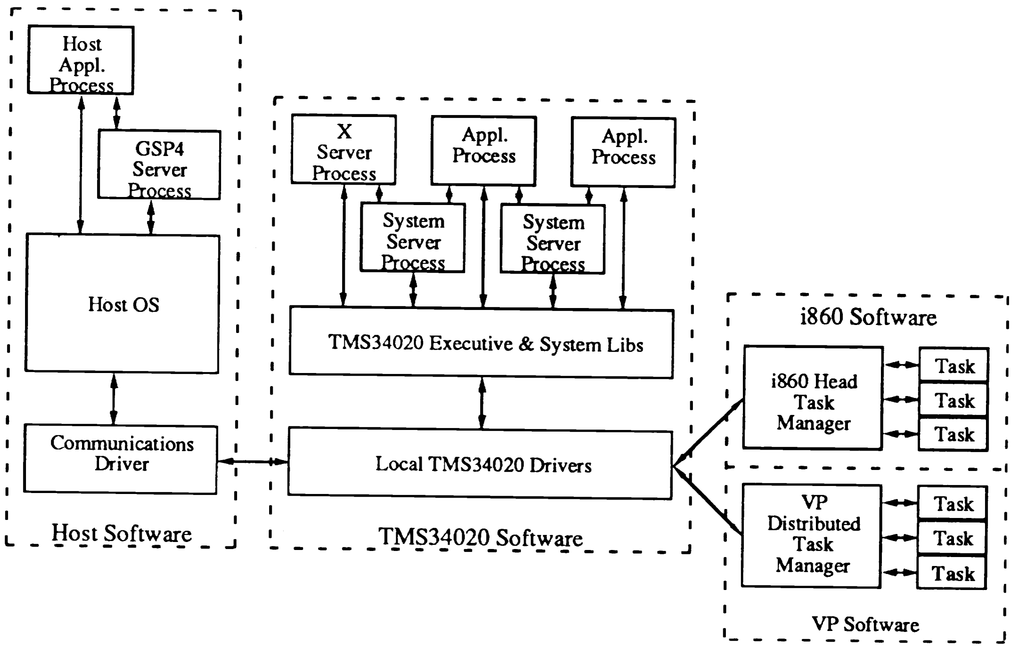

Fig 5. Overall software structure