A Dedicated Graphics Processor SIGHT-2

Masaharu Yoshida, Tadashi Naruse and Tokiichiro Takahashi

Autonomous Robot Systems Laboratory
NTT Human Interface Laboratories
NIPPON TELEGRAPH AND TELEPHONE CORPORATION
1-2356 Take Yokosuka-Shi
Kanagawa 238-03 Japan

Abstract

SIGHT-2 is a multiprocessor system that is intended to efficiently execute the ray tracing algorithm. To achieve high efficiency, three kinds of parallel execution mechanisms; (i) a multiprocessor configuration, (ii) a parallel execution of three dimensional vector operations, and (iii) functionally distributed parallel processing are introduced. Owing to the latter two techniques, each processing element (PE) has the ability to execute the standard ray tracing algorithm 10 times faster than a VAX11/780 with a floating point accelerator. In the present configuration, SIGHT-2 utilizes 16 PEs, which results in a peak power of 66.72 MFLOPS / 133.28 MIPS. During ray tracing, the efficiency of each PE is over 99% under static load balancing.

In this paper, SIGHT-2 system architecture, its PE configuration, and VLSIs design are discussed. The system performance is also discussed.

1. Introduction

With computer graphics, computers are able to convert large amounts of complex data into visual forms that are more readily understood by humans. There is, however, a tradeoff between visual detail and the computation cost needed to render the image. Among the various computer graphic techniques, the ray tracing algorithm offers the highest accuracy and realism; both important characteristics for image comprehension[1].

The ray tracing algorithm duplicates the action of human vision but in reverse. Instead of light rays being collected by an eye, ray-object intersections are modeled from the eye point through each pixel on the image plane to the objects forming the image. The ray tracing algorithm can yield extremely detailed images but at a very high computation cost. This cost can be reduced by increasing the processor speed or the number of processors.

The basic ray tracing algorithm is well suited to parallel processing because there is no need for inter-PE communication. Since the ray tracing algorithm is extremely time consuming, many multiprocessor systems have been developed (for example, [2]). While these multiprocessor systems do decrease image generation time, the achievable speeds are still too slow for practical commercial use.

By analyzing the ray tracing algorithm in detail, we have clarified that it can be performed in the following three levels of parallel execution[3,4].

(1) Pixel level parallel execution :
each pixel value can be calculated independently, since rays do not interact with each other.

(2) Operation level parallel execution :
three dimensional vector operations can be executed in parallel with respect to the x, y, and z axes. They occupy the largest part of the ray tracing algorithm[3].

(3) Function level parallel execution :
three function level processes can be executed in parallel: floating point calculations like intersection calculations, integer operations like address calculation, and data transfer.

Based on the above analysis, we have proposed a ray tracing computer architecture, SIGHT[3], and developed a SIGHT prototype system with one PE[3]. Its PE employs special hardware to perform three dimensional vector operations in parallel. This innovation makes it possible to execute the ray tracing algorithm very quickly.

The SIGHT prototype system verified that its PE architecture is efficiently suited to the parallel execution of the three dimensional vector operations. From SIGHT, we developed the SIGHT-2 prototype system which is a multiprocessor system with 16 PEs.

In this paper, first, the parallel execution method of the ray tracing algorithm and the architecture of SIGHT will be recapitulated. Second, the SIGHT-2 PE and VLSI designs are detailed. Finally, measured performance is compared to that of a VAX11/780.

2. Parallel Execution of Ray Tracing Algorithm

In this section, we discuss the parallel execution of the ray tracing algorithm. First, the basic ray tracing algorithm is given, second, the pixel level parallel execution method is explained, and finally, the operation level parallel execution method is detailed.

2.1 Ray Tracing Algorithm

The basic concept of the ray tracing algorithm is that the computer simulates a pin hole camera. That is, the algorithm generates the image on the screen by tracing rays from the eye to the objects as shown in Figure 1.

The basic image generation steps are as follows :

(1) The ray equation is calculated from the eye position to a pixel position.

(2) The intersections of the ray with each object are calculated.

(3) If the object closest to the screen is transparent, the reflected and refracted ray equations are calculated, and the new rays are recursively traced with the same steps as in (1) and (2). If the closest object is a mirror, the reflected ray is traced in the same way as above. if the closest object is another type, then step (4) is executed. If no object intersects with the ray, then step (4) is executed.

(4) The shading calculation is executed.

The above calculation steps are repeated until all pixel values of the screen and screen image are generated.

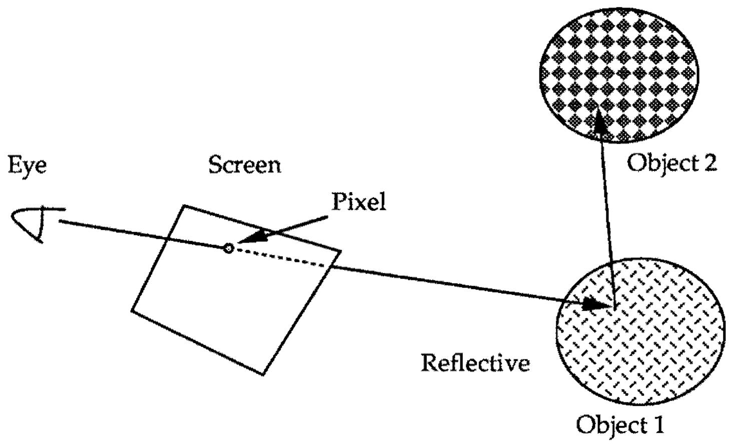


Figure 1 Ray Tracing Algorithm

The calculation cost of the ray tracing algorithm is proportional to the product of the number of pixels and objects. Therefore, the ray tracing algorithm needs enormous computing time to generate a photo realistic image, and parallel execution is necessary to realize high speed computation. In the following subsections, we discuss the parallel execution of the ray tracing algorithm.

2.2 Pixel Level Parallel Processing

In the ray tracing algorithm, since there is no interference among rays, each pixel value on the screen is calculated independently. This means that simple multiprocessor architectures are well suited to the ray tracing algorithm. If each processing element (PE) holds a copy of the entire data and program codes that are needed to generate the desired image, there is no inter-PE communication. We call this kind of parallel execution, "Pixel level parallel processing".

There have been some systems developed from this idea [for example, 2]. While these multiprocessor systems reduce image generation time considerably, the achievable speeds are still insufficient for practical use. For this reason, we can not expect sufficient performance with only pixel level parallel processing.

2.3 Operation Level Parallel Processing

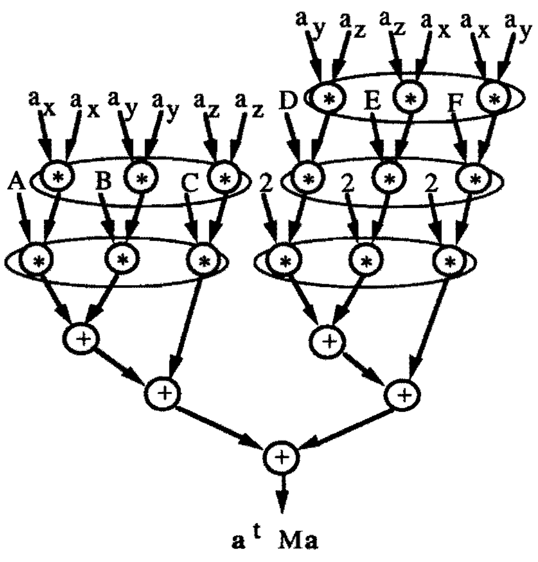
In the previous papers[3,4], we have shown another kind of parallelism for the ray tracing algorithm. This is summarized in this subsection.

As T. Whitted pointed out[1], intersection calculations occupy 60% ~ 95% of the total computation time of the ray tracing algorithm. So, let us consider the intersection calculation.

The ray equation is given by.

$$\mathbf{x} = \mathbf{a} * t + \mathbf{v} \quad (1)$$

where, **V** is the eye position vector, **a** is the direction cosine vector, and **t** is a parameter.



a_x a_x a_y a_y a_z a_z
a_y a_z a_z a_x a_x a_y
A
B
C
D
E
F
2
2
2
a^t Ma

Figure 2 The Data Flow Graph

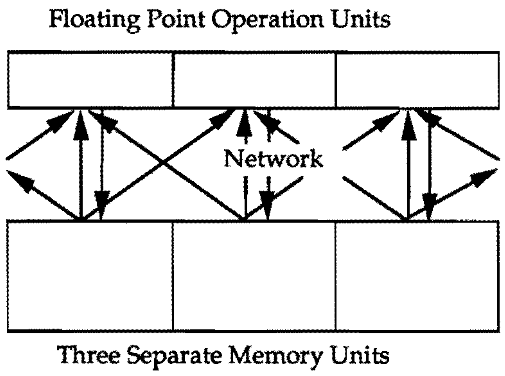


Figure 3 Three Dimensional Vector Parallel Execution Mechanism

In the case of a quadric surface, the object is given by the quadratic form.

$$x^t M x = 1 \quad (2)$$

$$M = \begin{pmatrix} A & F & E \\ F & B & D \\ E & D & C \end{pmatrix}$$

where, M is a coefficient matrix. By substituting equation (1) into equation (2),

$$a^t M a * t^2 + 2 a^t M v * t + v^t M v = 1 \quad (3)$$

is obtained.

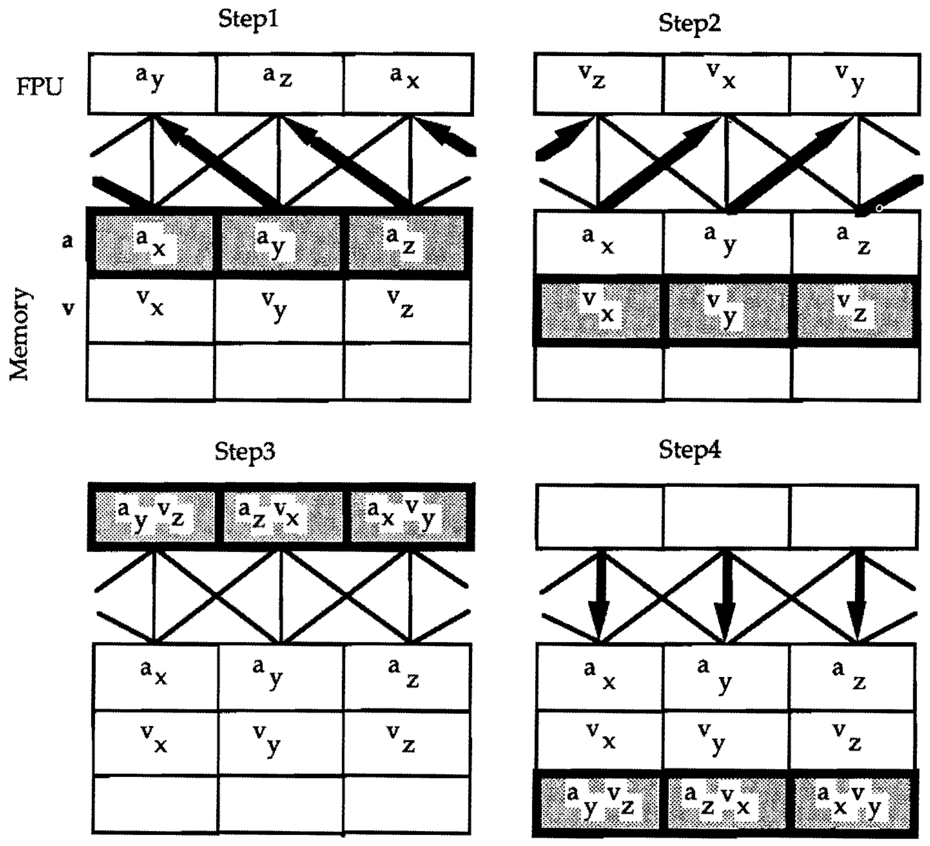


Figure 4
Example of Parallel Three Dimensional Vector Operations in the TARAI Unit

By solving equation (3), the intersection is obtained. In equation (3), it takes much time to calculate the coefficients of the quadratic equation.

Figure 2 shows a data flow graph that computes a coefficient: $\mathbf{a}^t$ M $\mathbf{a}$. From this graph, it is shown that the input data of three multiplication operations enclosed by ellipse are regular and symmetric with respect to the x, y, and z axes. This means that these multiplications can be executed efficiently in parallel by using three Floating Point operation Units (FPUs) and three separate memory units connected by a simple network. This concept was demonstrated to be correct by the tests performed on the TARAI unit in 1987[3].

One realization of the execution mechanism is shown in Figure 3. How the above operations are executed on this mechanism will be explained. Let us consider three products

$$a_y * v_z,\ a_z * v_x,\ a_x * v_y.$$

We store the data in the memories as shown in Figure 4, i.e. data for the x-axis is stored in the left memory, data for the y-axis in the center memory, and data for the z-axis in the right memory. Thus multiplication operations can be executed as follows,

(1) a_x, a_y, a_z are transferred to the upper-left operation units (step 1),
(2) v_x, v_y, v_z are transferred to the upper-right units (step 2),
(3) multiplications are executed (step 3),
(4) results are transferred to lower memories (step 4).

It should be noted that no memory access conflict occurs during the execution of these operations.

Since most of the three dimensional vector operations have regularity and symmetry, this parallel execution mechanism works well for three dimensional vector operations. We call this unit, "TARAI", and the parallel process, "Operation level parallel processing".

3. SIGHT Architecture

SIGHT employs three kinds of parallel execution mechanisms;
(1) Pixel level parallel processing,
(2) Operation level parallel processing,
(3) Functionally distributed parallel processing.

The first two were described in the previous section. The last process will be explained in Section 3.3.

3.1 SIGHT System Architecture

SIGHT is a multiprocessor system for the fast execution of the ray tracing algorithm. SIGHT-2 was designed using the SIGHT architecture.

The SIGHT-2 prototype system is shown in Figure 5. There are 16 PEs, a host interface, a data collector and a frame buffer. Each PE has six communication interfaces; one is to the host processor, one is to the frame buffer through the data collector, and four are to the neighboring PEs.

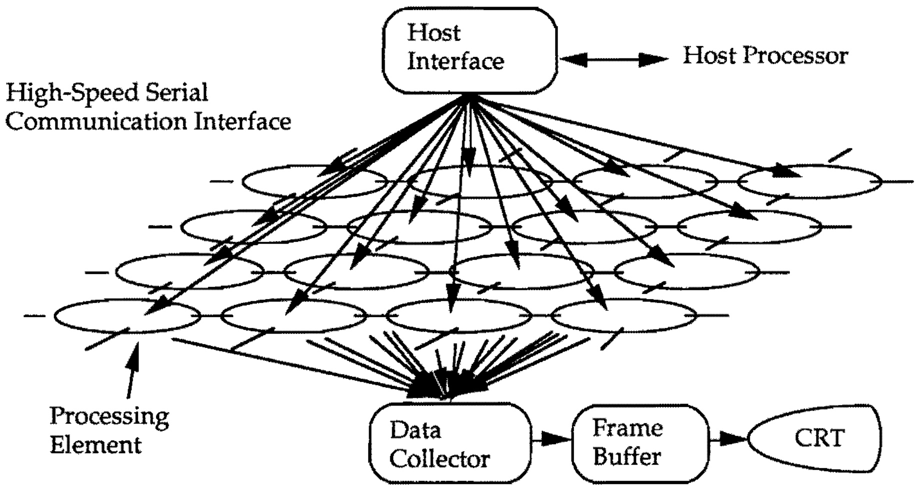


Figure 5 SIGHT-2 System Configuration

Inter-PE communication lines are prepared for the space subdivision ray tracing algorithm that will be implemented at a later date. Each communication line is a high-speed serial line, because it is necessary to reduce the connections for effective VLSI design. Data is packetized and transmitted through these communication lines.

The ray tracing program code and data for image creation are supplied by the host processor. Each PE executes the ray tracing algorithm and sends the pixel values to the frame buffer through the data collector. Finally, the images are displayed on the CRT.

3.2 The SIGHT PE Architecture

The PE architecture of SIGHT is shown in Figure 6. There are four major units; TARAI, MP, Direct Memory Access (DMA), and Data Base Memory (DBM). In the TARAI unit, there are three floating point operation units, three register files, and a TARAI network connecting FPUs to register files. The TARAI unit executes three dimensional vector operations in parallel. The MP unit contains an integer ALU, a register file and six channel high-speed serial communication interfaces. The MP unit controls the TARAI and DMA units, and communicates with the host processor, the frame buffer and four neighboring PEs. The DMA unit transfers the data between the register files and the DBM unit. The DBM unit is the main memory and holds the entire set of object and mapping data needed by each PE.

TARAI and MP units can not access the DBM unit directly. They can access only the register files. The register files and DMA units act as a cache memory controlled by the MP unit. This configuration makes it possible to construct a high-speed, high-density and large capacity memory system by using two types of memory chips: high speed, small capacity SRAMs used to construct register files; and low speed, high capacity DRAMs for the DBM unit.

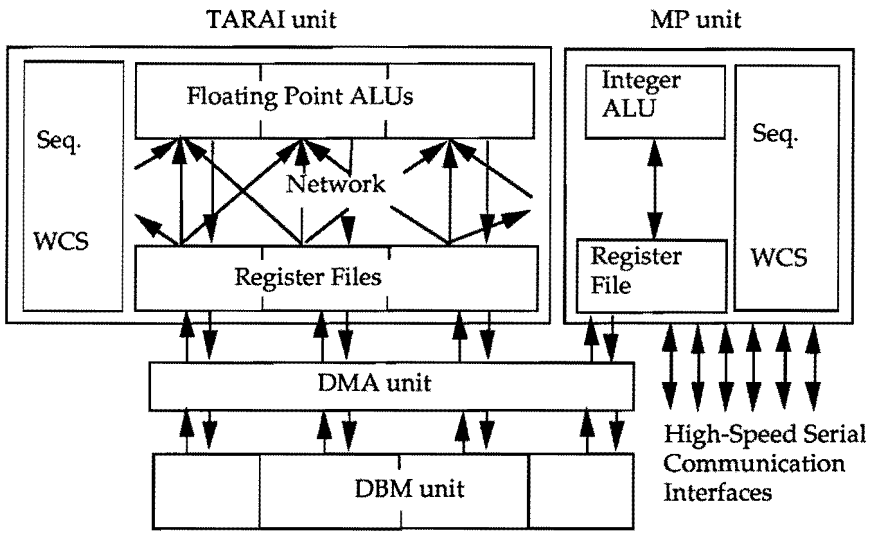


Figure 6 Processing Element (PE)

As the TARAI and MP units are controlled by separate microprogram both units can operate independently. The ray tracing program is written in micro code.

In the SIGHT PEs, there is only one dedicated hardware mechanism to execute the three dimensional vector operations in parallel. The rest of the ray tracing algorithm is executed as a microprogram. We consider this configuration is important because the ray tracing algorithm is still being optimized and there are many possibilities of improving it.

3.3 Functionally Distributed Parallel Processing

The most time consuming component of the ray tracing program is the calculation of ray-object intersections. Sequential execution of ray-object intersection is shown in Figure 7, and is summarized as follows.

(1) The MP unit calculates the stored address of one object in the DBM unit.

(2) The MP unit invokes the DMA unit.

(3) The DMA unit transfers the object definition data from the DBM unit to the register files.

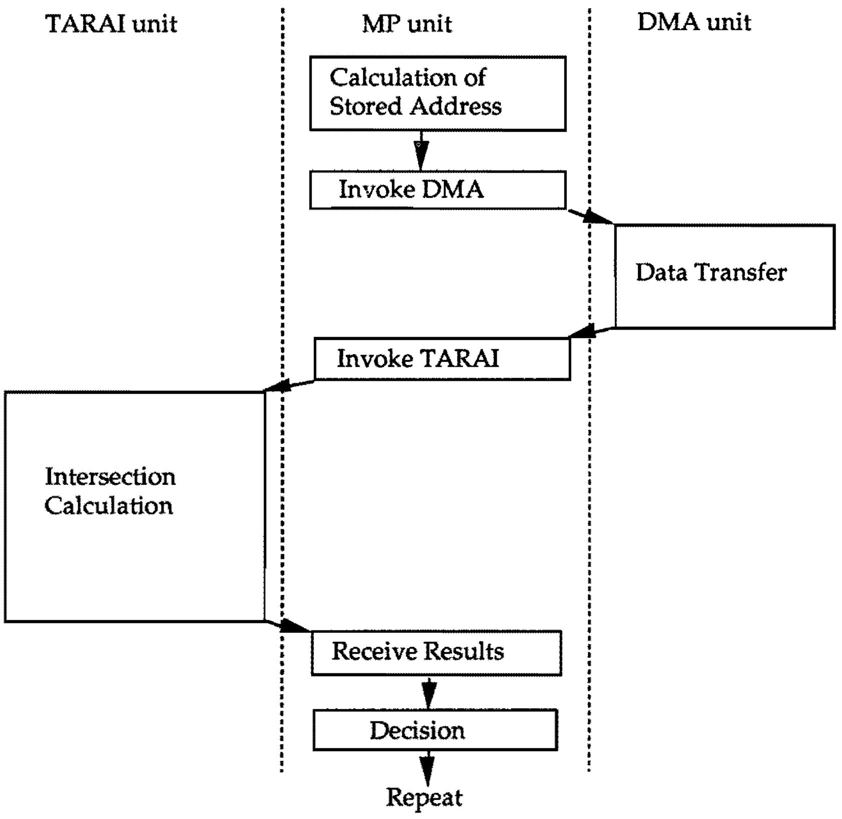


Figure 7 Execution of Ray Tracing

(4) After finishing data transfer, the MP unit invokes the TARAI unit.
(5) The TARAI unit executes the intersection calculation.
(6) The MP unit receives the intersection data from the TARAI unit, compares it with the previously calculated data, selects the object closer to the screen, and memorizes it.
(7) If there is more data to calculate, repeat above steps for a new object.

Since the TARAI unit, MP unit and DMA unit can operate concurrently, parallel pipelined operation is possible as shown in Figure 8. While the TARAI unit calculates intersections, the MP and DMA units prepare the next object's data, and the MP unit checks the previous result of the intersection calculation by the TARAI unit. We call this operation, "Functionally distributed parallel processing".

4. SIGHT-2 System Implementation

In this section, the SIGHT-2 system hardware configuration will be explained. After explaining PE configuration, we discuss VLSI design considerations for the PEs employed in SIGHT-2.

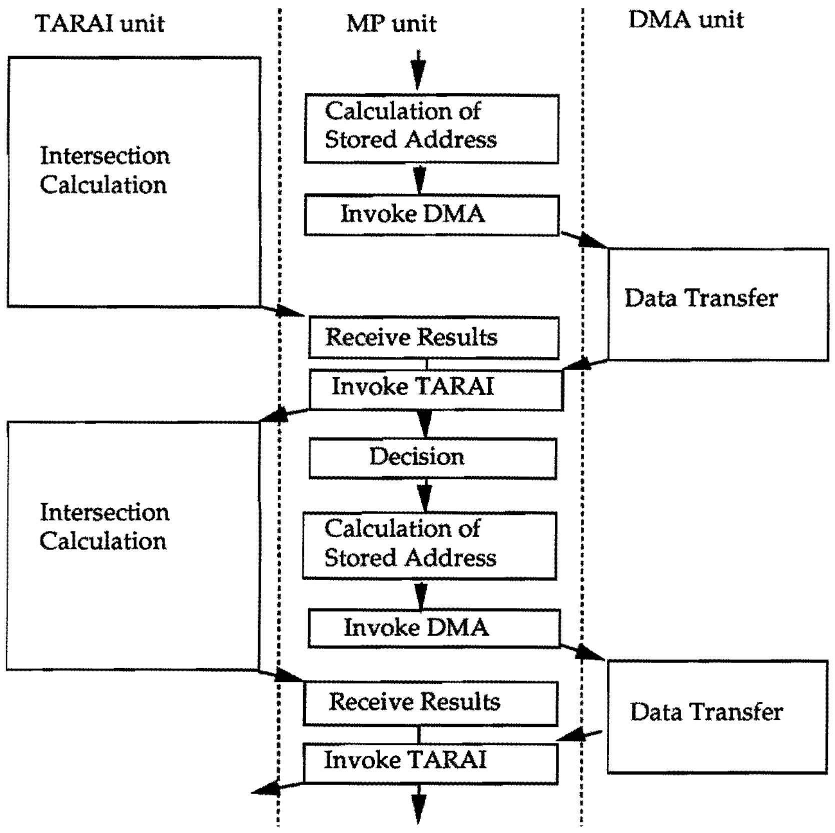


Figure 8 Pipelined Execution of Ray Tracing

4.1 PE Configuration

The hardware configuration of a PE is shown in Figure 9. The TARAI Controllers and Master Controller are custom VLSIs, which were designed for SIGHT-2. Their details are described in Section 4.3.

TARAI unit
The TARAI unit is made up of three FPUs and three register files. The TARAI network is constructed with three time-shared buses, and it will be described later. Each FPU contains a floating point ALU, a floating point multiplier, ROM function tables, and a TARAI Controller VLSI. Frequently referenced functions, such as square root, sine, cosine, and gaussian, are provided in the function tables for the high speed calculation.

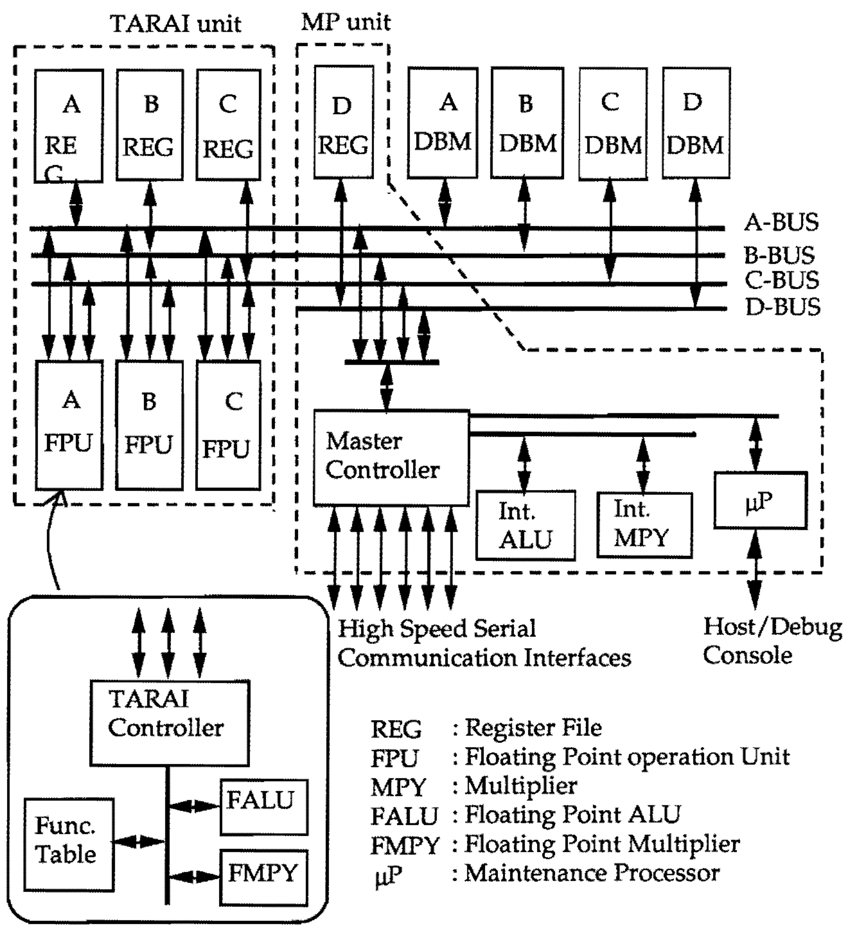


Figure 9 The hardware configuration of a PE

The TARAI Controller VLSI, controlled by the microprogram of the TARAI unit, controls the bus selection of each FPU. The bus architecture reduces the number of pins in the TARAI Controller VLSI chip.

MP unit

The MP unit contains a 32 bit integer ALU (Int. ALU) and multiplier (Int. MPY), a maintenance processor (μP) and a Master Controller VLSI. The Master Controller VLSI includes six channel high-speed serial communication interfaces and a DMA unit. The capacity of each serial communication interface is 2 mega bytes per second.

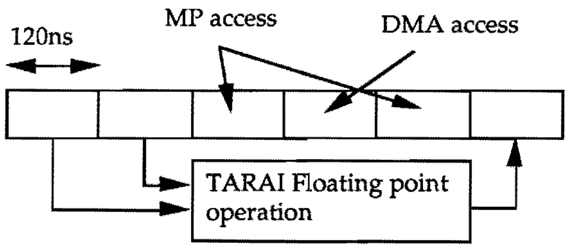
The basic clock rate of the MP unit is 120 nano seconds, which is synchronized to the clock in the TARAI unit. The MP unit can invoke the TARAI unit microprogram routine at any address of the TARAI unit's WCS.

The μP controls the microprogram execution of both the TARAI unit and MP unit. It also supports a microprogramming developing environment by offering facilities such as microprogram loading, inline assembler, and debugging.

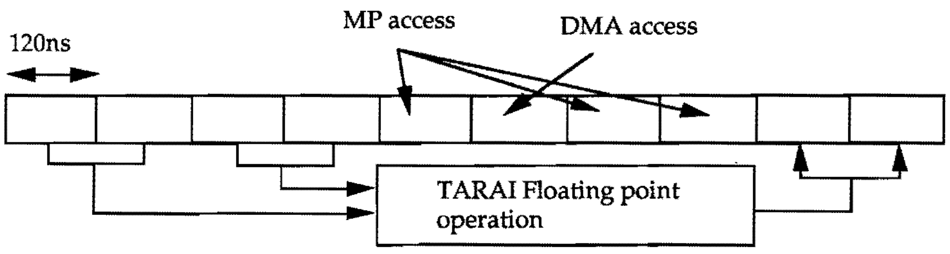
Bus

The TARAI network is constructed with three 32 bit time-shared buses; A-BUS, B-BUS, and C-BUS. This architecture realizes the concurrent execution of floating point operations and data transfer between the register files and the MP unit or DBM unit without decreasing performance of the TARAI unit. The three FPUs operate in a three address mode.

The bus timing of the TARAI unit is shown in Figure 10. The bus clock period is 120 nano seconds. For the single precision floating point data format, six time slots are needed. As a result, the TARAI unit cycle time is 720 nano seconds, and the TARAI unit can perform at up to 4.17 MFLOPS in single precision.



(a) Single Precision



(b) Double Precision

Figure 10 Bus Timing

The first 2 time slots are assigned to transfer two operands from the register files, and the final time slot is to store operation results in the register files. While the FPUs execute floating point operations, they do not access the buses. The remaining time slots are used for the MP and DMA units to access TARAI unit's register files.

The time slots for the TARAI and DMA units are assigned statically, while the time slots for the MP unit are assigned dynamically. The MP unit can use time slots only when other units are not using the buses. The bus is arbitrated by the Master Controller VLSI.

For the double precision floating point data format, ten time slots are needed as shown in Figure 10. Two time slots are needed to transfer one word between the FPUs and the register files, and one extra time slot is needed to execute the double precision floating point operations.

4.2 Design Consideration of VLSI Chips

We developed two types of VLSI chips. Since a fair amount of miscellaneous control gates are included on the VLSI chips, each PE can be implemented on just two printed circuit boards. The key design factors of the VLSIs are as follows :

TARAI Controller

(1) Bus interfaces:
As shown in Figure 9, the TARAI Controller must interface four 32 bit buses; three constitute the TARAI network, and one connects the floating point ALU, floating point multiplier, and function table. About one half of the TARAI Controller chip pins are tied to these buses. These buses are also used for chip testing. Through them, test data can be efficiently read and/or written.

(2) Nanoprogram control:
For the single precision floating point data format, each TARAI instruction is executed in 6 bus time slots (Figure 10). Each FPU is controlled by a nanoprogram. As the architecture of the TARAI unit is not yet optimum, it is desirable to be able to modify the instruction set; moreover, it is necessary to create application specific instruction sets. Therefore, we added RAM storage to the TARAI Controller chip to contain the nanoprogram.

Master Controller

(1) Communications:
Since each PE communicates with the host processor, four neighboring PEs, and the frame buffer, six communication lines are needed. Due to the restriction in pin number, a serial communication interface was adopted. The capacity of each serial communication line is 2 mega bytes per second. This performance is sufficient for host-PE and PE-frame buffer communications. This is because the interface between the host interface and host processor is constructed by using the DR11W[1] interface, the capacity of which is 2 mega bytes per second. In addition, the pixel value execution task takes much longer than communication tasks.

[1] Digital Equipment Corporation

Though inter PE communication is not necessary for standard ray tracing, it is provided for implementing advanced ray tracing techniques, such as space sub-division ray tracing. In such cases, the inter PE communication speed is fast enough, because the intersection computation takes much longer than communication tasks.

(2) Bus interfaces:
Due to the restriction in the number of pins, the Master Controller chip interfaces four multiplexed buses (A,B,C,D-Bus). This does not decrease performance, because only one register file is accessed by the MP at a time.

4.3 VLSI Chips

We developed two custom VLSIs: TARAI Controller and Master Controller. Both chips are 1.5 micron CMOS channel-less gate arrays.

TARAI Controller Chip
The block diagram of the TARAI Controller VLSI chip is shown in Figure 11. The TARAI Controller was designed to yield a function sliced architecture for the TARAI network, and three chips are used to construct the network. Each Controller includes a bus coupler, floating point operation unit support circuits, and writable control store (WCS) for the nanoprogram. In addition, a 32 bit integer ALU, different from that in the MP unit, is also included for register file address calculations.

The nanoprogram controls the operation of the floating point ALU, floating point multiplier, and function table. Since the nanoprogram is stored in the WCS, the microprogram architecture of the TARAI unit can be changed to fit various applications. The size of the WCS is 128 words each 48 bits wide. The nanoprogram is upward loaded or downward loaded by the μP through the A-BUS, B-BUS, and C-BUS.

The Controller interfaces four 32 bit bi-directional buses. In general, a VLSI chip with many output pins may generate electric noise when many output signals change simultaneously. This is no problem for the TARAI Controller because only one bus is driven by the TARAI Controller at a time.

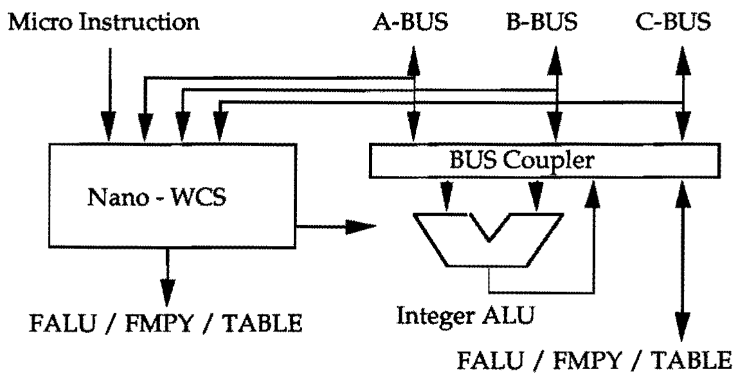


Figure 11 TARAI Controller

The TARAI Controller VLSI chip has 28 kilo gates, housed in a 233 pin Pin Grid Array (PGA) package.

Master Controller Chip
The block diagram of the Master controller VLSI chip is shown in Figure 12. It includes four MP bus interfaces, a 32 bit barrel shifter, six channel high-speed serial communication interfaces, the DMA unit, a μP interface, and a system timing generator. Each MP bus interface contains a 32 bit bi-directional data register and a 16 bit address register/ incrementer. The address increment value is micro-programmable in the range of 0 to 15.

High speed serial lines are provided for the transmission of data packets to neighboring PEs, the host processor, or the frame buffer through the data collector. The length of each packet is 64 bits. Because 11 bits are provided for packet address, it is logically possible to connect over 2,000 PEs. The barrel shifter is included to manipulate the packets.

The system timing generator supplies all timing clocks used in the PE and also generates the bus time slots clock.

The system timing generator controls the execution timing of the TARAI unit , MP unit and DMA unit, arbitrates four buses; A-BUS, B-BUS, C-BUS and D-BUS, and refreshes DRAMs in the DBM unit.

This VLSI has 27 kilo gates in a 299 pin PGA package.

4.4 System Overview

SIGHT-2 Prototype System
The appearance of the SIGHT-2 prototype system is shown in Figure 13. The front door has been removed to take this picture. The cabinet is about 175 cm high, 130 cm wide, 110 cm deep. 16 PEs are vertically mounted in the upper half of the cabinet. The host interface, the data collector and the frame buffer are mounted in the lower left of the cabinet.

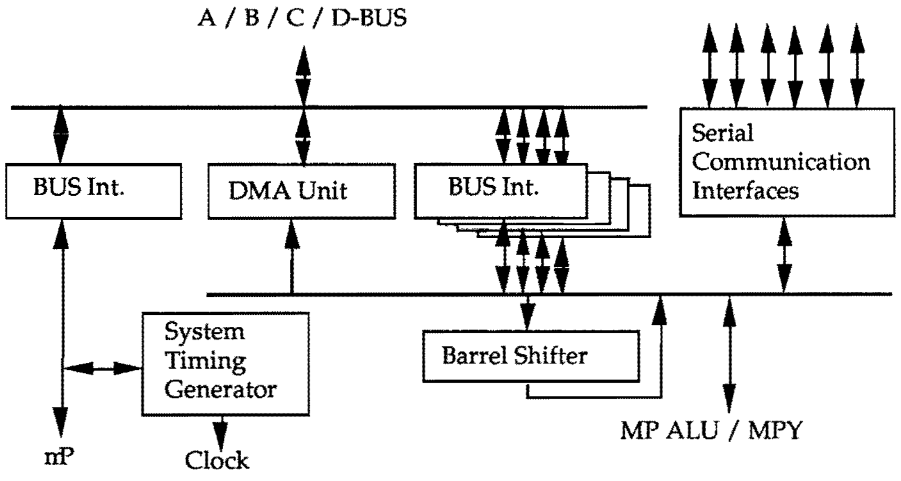


Figure 12 Master Controller

The ribbon cables are the high-speed serial communication lines. Each PE communicates with its neighbors, the host processor and the data collector through the serial lines. There is no mother board (back plane) to make it easy to change connections between PEs.

PE of SIGHT-2

A PE of SIGHT-2 is shown in Figure 14. Each PE is consists of two piggybacked printed circuit boards. The board size is 53 cm wide, 47 cm deep.

A SIGHT prototype with a single PE was developed and tested in 1987[3]. The single PE was implemented on 15 wire wrapped circuit boards. In SIGHT-2 the number of boards is reduced to only two. This compact implementation is made possible with the two specialized VLSIs.

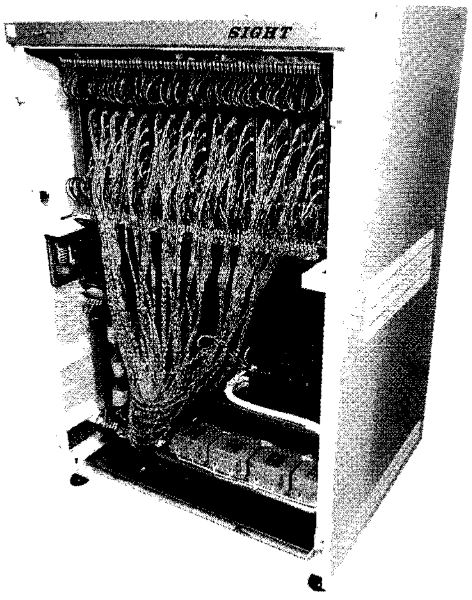


Figure 13
The SIGHT-2 Prototype System

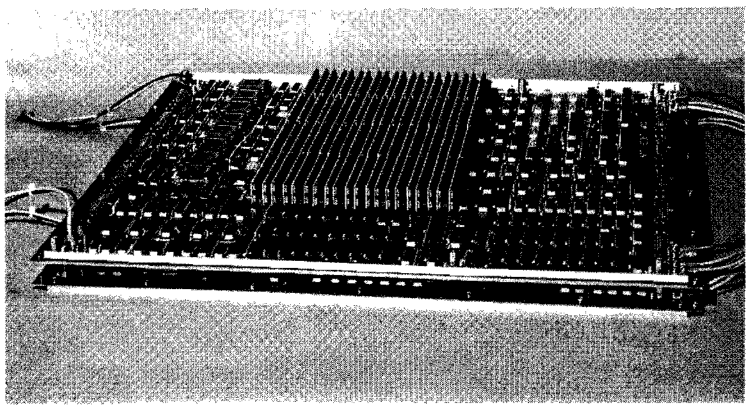


Figure 14 PE

The upper board is shown in Figure 15. The majority of this board is occupied by memory chips. The upper center area is the DBM unit, which consists of 64 mega bytes of DRAM. Another 16 mega bytes of memory are added for error correction. Eight mega bit DRAM modules are used to construct the DBM unit. The upper left area contains the error correcting units for the DBM unit. The error correction units are needed because the total capacity of the DBM unit of the SIGHT-2 prototype system (16 PEs) is one giga bytes. The lower middle area is the WCS of the TARAI unit, and its size is 64 kilo words, each is 104 bits wide. The lower right area is the WCS of the MP unit. Its size is 64 kilo words each 96 bits wide. In the upper right area are the MP sequencer and pipeline registers.

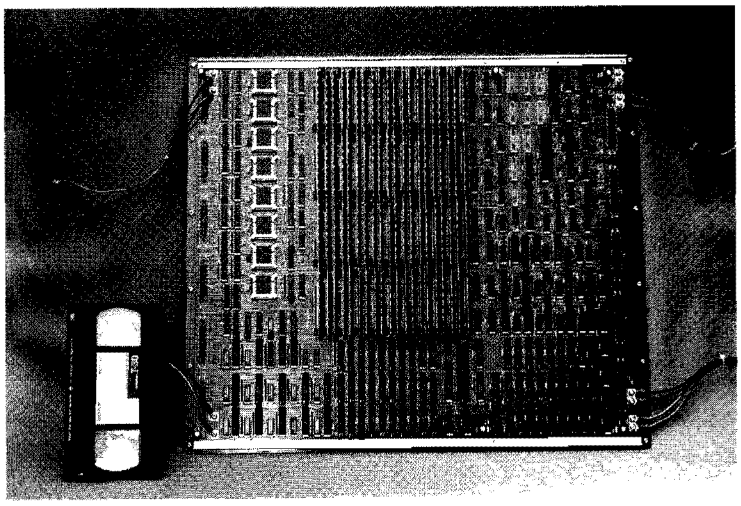


Figure 15
Upper Board

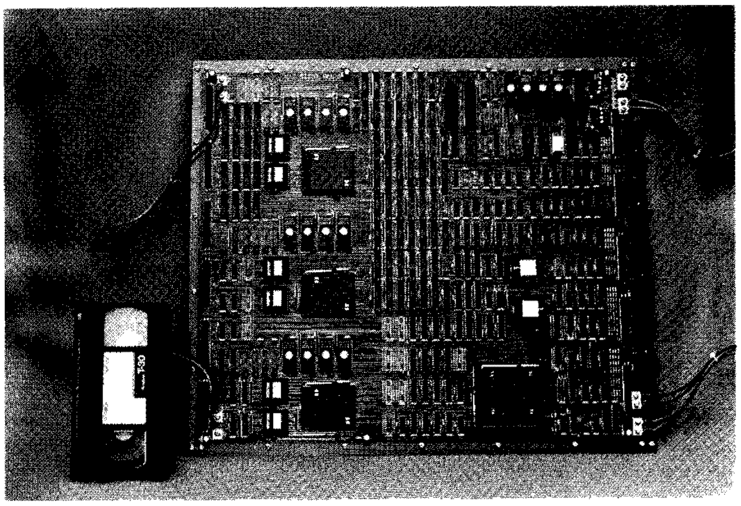


Figure 16
Lower Board

The lower board is shown in Figure 16. The major part of this board is occupied by the ALUs. The chip groups on the left of the board are made up of the floating point multiplier; WTL1164GCB[2], the floating point ALU; WTL1165GCB[2], EPROMs for the function table, and the TARAI controller VLSI. These units construct the TARAI unit and execute three dimensional vector operations in parallel.

The lower right area holds the MP unit. The connectors for the ports of the high-speed serial communication interfaces are located across the right edge of the board. The upper center area is the register files. Each register file is 256 kilo bytes.

The μP is located in the upper right. The monitor, the debugger and the initial microprogram for the MP unit are stored in EPROMs, the total capacity of which is 256 kilo bytes.

5. System Performance

The image shown in Figure 17 was used for comparing the performance of SIGHT-2 and a VAX11/780 with a floating point accelerator. The size of the image is 512 x 480 pixels. The surfaces of all spheres are reflective.

The assignment of image generation areas to each PE is shown in Figure 18, that is, vertical stripes the full height of the screen are assigned to each PE. The load balance of each PE is statically controlled. Static load balancing works well with increasing image complexity.

Measured system performance is shown in Figure 19. One PE is 10 times faster than the VAX, and 16 PEs are 158.8 times faster.

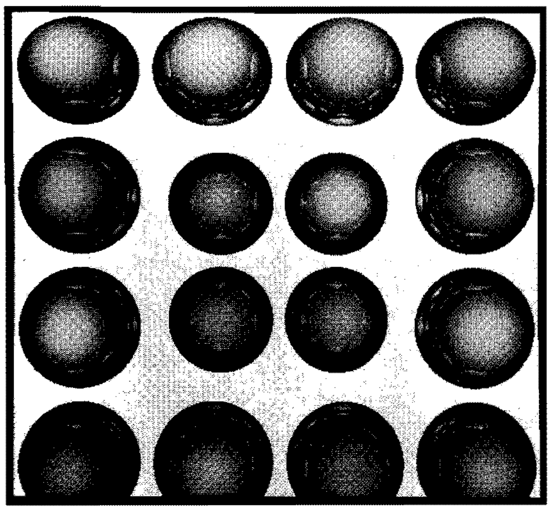


Figure 17 A Synthesized Image for Performance Comparison

[2] Weitek Corporation

We define PE efficiency by the mean ratio of one PE execution time to the system execution time. Thus, PE efficiency is 99.2% in the ray tracing experiment. The overhead to wake up PEs and the load unbalance among PEs do lower the efficiency; however, from Figure 19, it is shown that this effect is small.

These results show clearly that the SIGHT architecture is well suited to efficiently execute the ray tracing algorithm in parallel.

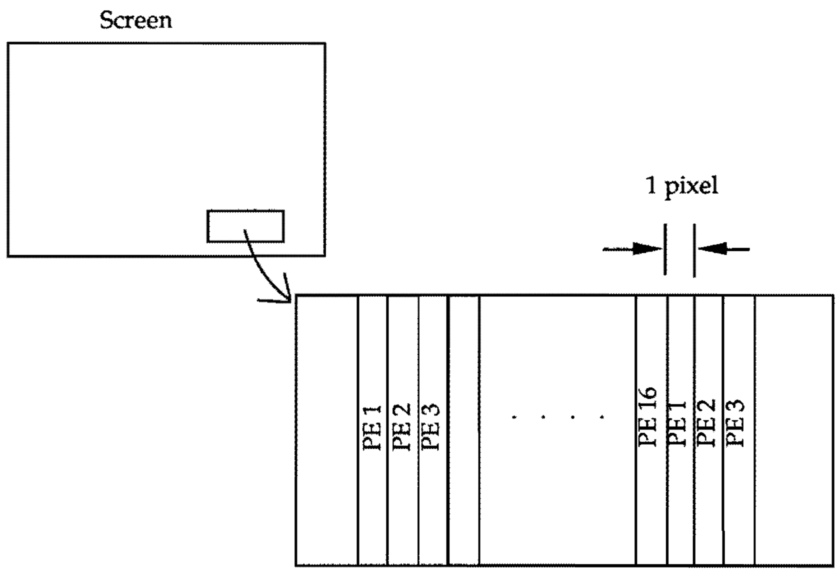


Figure 18 PE assignment

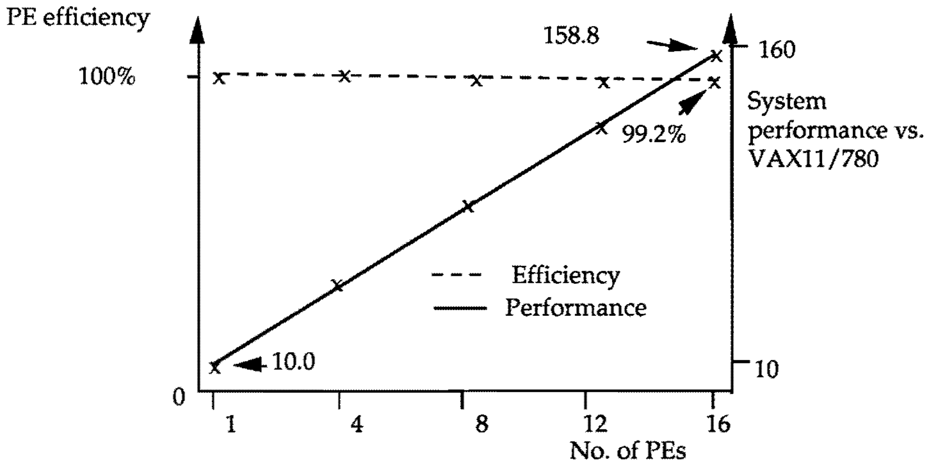


Figure 19 System Performance

6. Conclusion

The SIGHT is a multiprocessor system for the fast execution of ray tracing. To achieve high efficiency, three kinds of parallel architecture are introduced.

(1) Pixel level parallel processing,
(2) Operation level parallel processing,
(3) Functionally distributed parallel processing.

We designed two kinds of custom VLSIs; the TARAI Controller and Master Controller. With them, one PE can be constructed on just two printed circuit boards.

Each PE has a peak performance of 4.17 MFLOPS / 8.33 MIPS, and executes the standard ray tracing algorithm 10 times faster than a VAX11/780 with a floating point accelerator. The SIGHT-2 has 16 PEs and its overall performance is about 160 times faster than the VAX. Its peak performance is 66.72 MFLOPS / 133.28 MIPS. PE efficiency is 99.2% when executing the ray tracing algorithm.

Though SIGHT-2 has achieved significantly faster ray tracing, even higher performance is possible through hardware optimization with the same architecture.

Acknowledgements

We are grateful to Mr Hiroshi Kaneko, Mr Kei Takikawa and other colleagues in our section for their invaluable advice and encouragement.

Reference

1. Whitted, T.: An improved illumination model for shaded display. Comm. ACM 23(6), pp.343-349 (June 1980)
2. Nishimura, H., Ohno, H., Kawata, T., Shirakawa, I., and Ohmura, K.: LINKS-1 : a parallel pipelined multimicrocomputer system for image creation, Proc. 10th Annu. Int'l Symp. on Computer Arch., pp. 387-394 (1984)
3. Naruse, T., Yoshida, M., Takahashi, T., and Naito, S.: SIGHT - A Dedicated Computer Graphics Machine, Computer Graphics Forum (Journal of the European Association for Computer Graphics), Vol. 6, No. 4, pp. 327-334 (1987-12)
4. Takahashi, T., Yoshida, M., and Naruse, T.: Architecture and Performance Evaluation of the Dedicated Graphics Computer : SIGHT, Proc. of IEEE MONTECH '87 (Compint '87), pp. 153-160 (1987-11)