PAPER *Special Issue on Development of Advanced Computer Systems*

# *ReVolver/C40* : A Scalable Parallel Computer for Volume Rendering — Design and Implementation—

**Shin-ichiro Mori**[†], *Member*, **Tomoaki Tsumura**[††], **Masahiro Goshima**[†], **Yasuhiko Nakashima**[††], **Hiroshi Nakashima**[†], *Nonmembers*, *and* **Shinji Tomita**[†], *Member*

**SUMMARY** This paper describes the architecture of *ReVolver/C40* a scalable parallel machine for volume rendering and its prototype implementation. The most important feature of *ReVolver/C40* is view-independent real time rendering of translucent 3D object using perspective projection. In order to realize this feature, the authors propose a parallel volume memory architecture based on the principal axis oriented sampling method and parallel treble volume memory. This paper also discuss the implementation issues of *ReVolver/C40* where various kinds of parallelism extracted to achieve the rendering performance are explained. The authors developed its prototype systems and their performance evaluation results are explained. According the evaluation of the prototype systems, *ReVolver/C40* with 32 parallel volume memory is estimated to achieve more than 10 frame per second for $256^3$ volume data on $256^2$ screen using perspective projection. The authors also review the development of *ReVolver/C40* from several view points.

*key words: Volume Rendering, Parallel Processing, Scalable Architecture, Visualization*

## 1. INTRODUCTION

Volume rendering is a visualization technique for a three dimensional object called *volume data* which has color and transparency on its grid points called *voxels*. Such a volume data is conventionally obtained by medical scanning equipments such as CT and MRI scanner, but recently also produced by scientific computation such as for structural analysis and fluid dynamics. Since volume rendering requires a huge amount of computational power and resources, it is natural and necessary to exploit parallel processing techniques for high speed rendering.

In order to realize real-time volume rendering for large volumedata, we had proposed a parallel hardware architecture[13], [14] for volume rendering and we have designed *ReVolver/C40* [15], [17], [18] as its prototype implementation (Fig. 1).

In this paper, we introduce the architecture of *ReVolver/C40* in section 2. Then we discuss implementation related issues of *ReVolver/C40* in section 3, followed by its performance evaluation in section 4. After that, we review the development of *ReVolver/C40* from several view points in section 5 and conclude in section 6.

Manuscript received February 7, 2003
Manuscript revised 2003

[†]Graduate School of Informatics, Kyoto University, Yoshida-Hon-Machi, Sakyou-ku, Kyoto, 606-8501 JAPAN
[††]Graduate School of Economy, Kyoto University

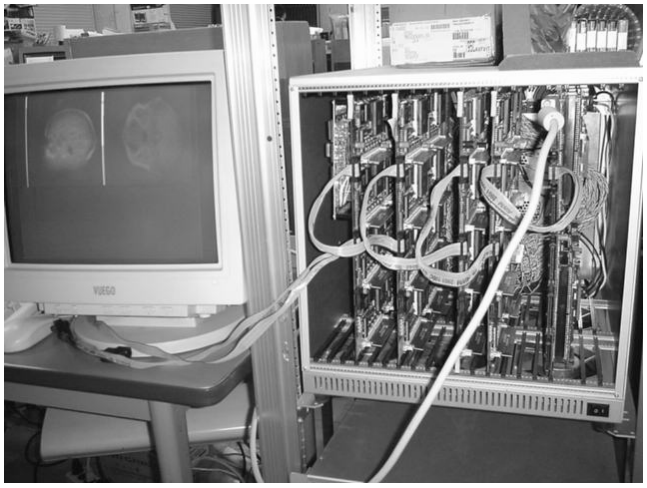

**Fig. 1** ReVolver/C40-demo

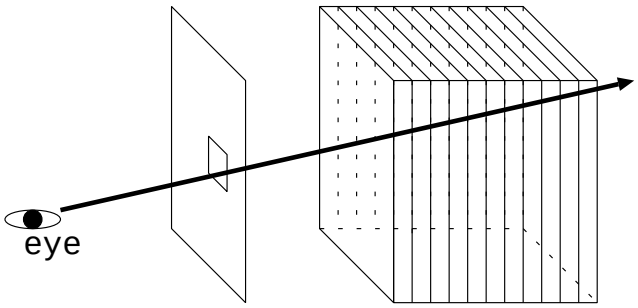


**Fig. 2** Volume Rendering (Ray Casting Algorithm)

## 2. CHARACTERISTICS OF ReVolver/C40

### 2.1 Fundamental Features in ReVolver/C40

*ReVolver/C40* adopts ray-casting style volume rendering algorithm which is widely used image-space approach. This algorithm traces rays through the volume, accumulating color along each ray(Fig. 2). Let $v_0, v_1, \ldots, v_n$ be voxels along a ray spatially ordered from view point, $c_i$ and $t_i$ be color and transparency of $v_i$ respectively. The pixel value (or color) $C$ for this ray is

$$C = \sum_{i=0}^{n-1} c_i(1-t_i)\prod_{j=0}^{i-1} t_j. \quad (1)$$

This calculation is referred to as *composition operation*[1] and it can be computed recursively using the following equation.

$$\begin{aligned} C_{out} &= C_{in} + T_{in} \times t_i \times c_i \\ T_{out} &= T_{in} \times t_i \end{aligned} \quad (2)$$

where $C_{out}$ and $T_{out}$ are the total accumulated color and transparency just after the ray hit i-th voxel, and $C_{in}$ and $T_{in}$ are the total accumulated color and transparency just before the ray hit the voxel. This calculation is referred to as *over operation*[1]. It is because $C_{out}$ and $T_{out}$ can be thought as color and opacity of the combined object in which an object that has color $C_{in}$ and transparency $T_{in}$ is placed over voxel $v_i$.

As you can see from these equations, volume rendering belongs to a class of memory intensive computation. Thus the memory performance is crucial to the rendering speed. If we could prepare enough memory performance, there exist fruitful sources of parallelism. The ReVolver/C40 was designed by compiling all these parallelisms into this machine.

Since the target of *ReVolver/C40* is real-time visualizaion of not only medical volume data but also scientific data, *ReVolver/C40* also has the following features.

1. *Discrete and continuous models*
   In *discrete* model, it is assumed that a voxel is the center of a unit cube and any points in the cube have the same voxel value. On the other hand, voxels in *continuous* model are at vertices of the cube and the value of a point in the cube is obtained by linear interpolation of the voxel values. Both models should be applicable depending on the objective of rendering.

2. *Translucent volumes*
   Volumes to be rendered should be *translucent* so that the interior of objects is seen. The translucent rendering also copes with an *opaque* object, which conventional systems can visualize only, giving zero transparency to the voxels for its surface.

3. *Perspective and parallel projection*
   For the visualization of scientific computation results, such as analysis of seismic waves, *perspective* projection is often suitable to showing wide range of space. For small objects, such as human brains, *parallel* projection is also available as in conventional systems.

4. *Real-time rendering*
   Quick response following the movement of view point is essentially important to help the analysis and recognition of complicated volume data. This *real-time* rendering requires high processing speed over 10 frames per second.

To make *ReVolver/C40* have these features, we devised a simplified voxel sampling method. We also contrived a three dimensional volume memory fit for this algorithm in order to read all volume data on a ray of any direction in parallel.

### 2.2 Architectural Features of ReVolver/C40

#### 2.2.1 Sampling Method and Parallel Volume Memory

Rendering one image(or frame) of $N \times N$ from $N^3$ volume data requires $O(N^3)$ memory throughput. Thus, to satisfy the real-time requirement, we had to develop a conflict free highly parallel multi-bank memory which can simultaneously read out all voxels along a ray of any direction. Since *ReVolver/C40* supports perspective projection method as well as parallel projection, requirements to the memory system are much more severe than the systems which doesn't support perspective projection. In order to alleviate this severity, we had proposed a simplified voxel sampling method[13], [14]. It is called object-based (or slice-aligned) sampling recently[1], [7].

In this method, instead of sampling voxels on a ray at regular intervals of the ray itself(Fig.3(a)), we sample voxels at regular intervals of the ray's principal-axis which is the x,y, or z-axis most parallel to the ray(Fig.3(b)). Therefore, if the sampling interval is equal to the unit of the volume coordinate system, the coordinates of any sampling points with respect to the principal-axis are different each other. This makes it possible to construct a bank-conflict free parallel volume memory for both perspective and parallel projections(Fig.4). Here, the principal-axis of each rays varies among x-,y-, and z-axis according to the view point and we want to avoid coordinate transformation in the memory to keep real-time follow-up to view point movement. So, we slice the volume space into planes perpendicular to x-, y-, and z-axis and store a set of i-th planes from Xplanes, Yplanes and Zplanes into i-th Parallel Treble Volume Memory node. As it is obvious from this memory structure, the effective memory throughput scales with the number of the memory nodes, irrespective of the view point and the projection method.

#### 2.2.2 Composition Network

The next to be considered in the architectural design is how to organize the composition network in which the composition operation(Eq.1) is performed. As we have mentioned before, a pixel value can be calculated by applying the composition operation recursively, so that it is relatively easy to organize a N-stages linear pipeline of over operation(Eq. 2)[9], [15].

On the other hand, since this composition operation can also be deformable into a pipeline of tree re-

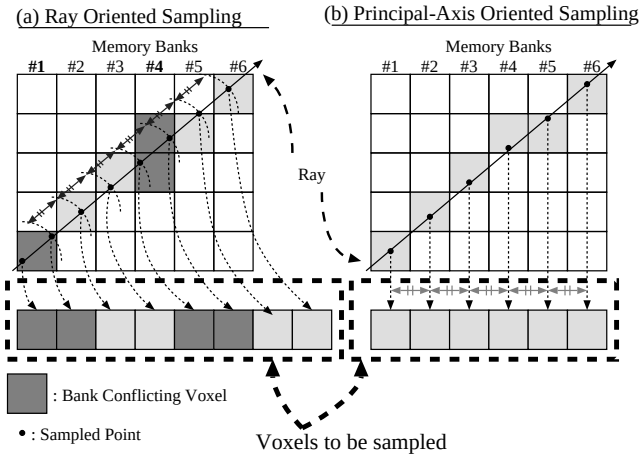


**Fig. 3** Simplified Sampling Method

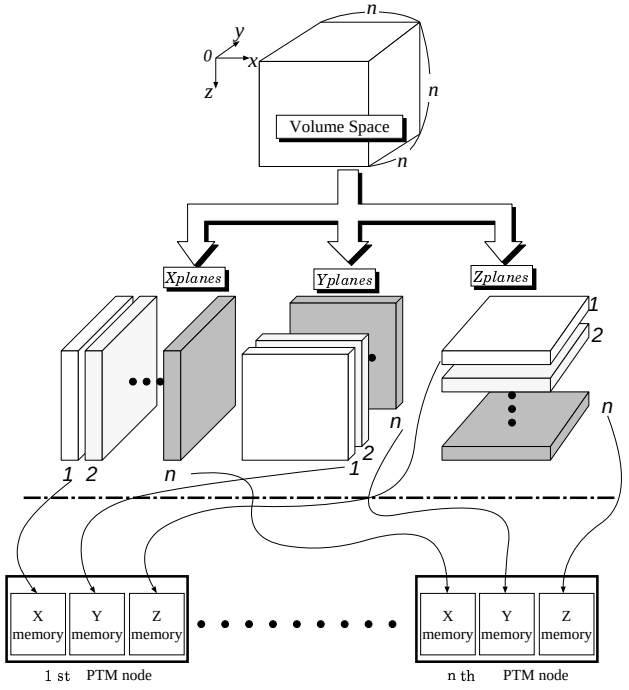


**Fig. 4** Parallel Treble Volume Memory

duction style[13], [14], there exists parallel volume rendering systems which adopt tree-based composition network.[2], [4] In our initial proposal[13], [14], we also used a tree-based composition network.

In *ReVolver/C40* , we organize the composition network as one directional link structure(Fig.5)†. The primary advantages of the link network are the good scalability and the ease of implementation; the cost of wiring becomes half while the logical throughputs of both networks are the same at the output-end. Though the latency increase from $O(logN)$ to $O(N)$, it is negligible if each created images has the order of $N^2$ pixels. This argument is not always applicable to most parallel volume rendering systems[4], [5]. It is because they use frame-based composition to hide their large communication overhead, whereas in *ReVolver/C40* we could use pixel-based composition since *ReVolver/C40* has the low latency composition network embedded into its N-stages linear pipeline of over operation.

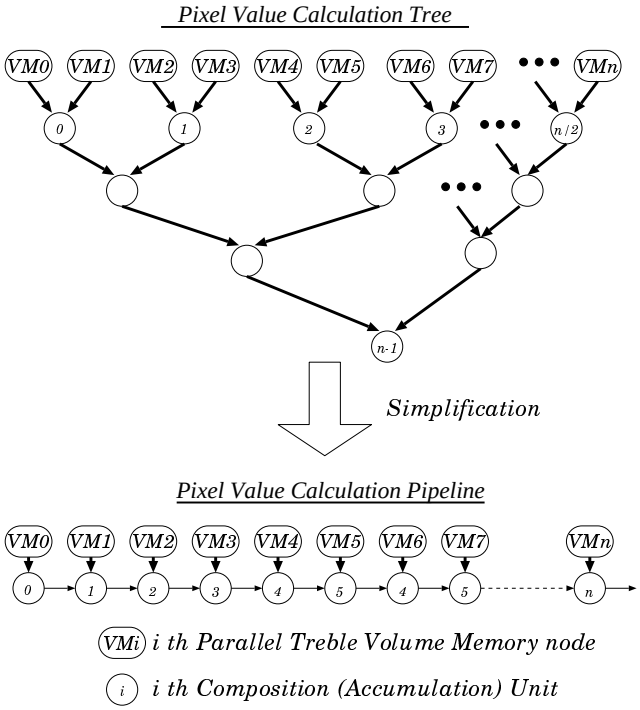


**Fig. 5** Composition Network

## 3. PROTOTYPE IMPLEMENTATION

### 3.1 Preliminary Specification

As a prototype implementation of our parallel volume rendering architecture, we had defined the specification of *ReVolver/C40* as follows based on the technology available in early '90s.

**Table 1** Preliminary Specification

| Parallel Volume Memory | |
|---|---|
| Number of Nodes (N) | 128 |
| Memory Size(total) | 512MB |
| Memory Size(node) | 4MB |
| Rendering Speed(perspective projection,discrete model) | |
| $128^3$ Volume on $128^2$ Screen | 30>FPS |
| $256^3$ Volume on $256^2$ Screen | 30>FPS |
| $512^3$ Volume on $512^2$ Screen | 10FPS |

### 3.2 Hardware Configuration of *ReVolver/C40*

Figure 6 shows the overview of the hardware config-

†The latest version of PCB boards of *ReVolver/C40* has bi-directional ring structure for another reasons[25] (see section 5) but they are still waiting for parts assembly.

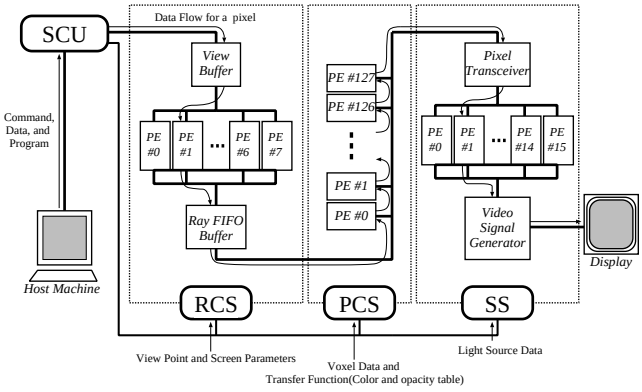


**Fig. 6** Overview of the Prototype System: ReVolver/C40

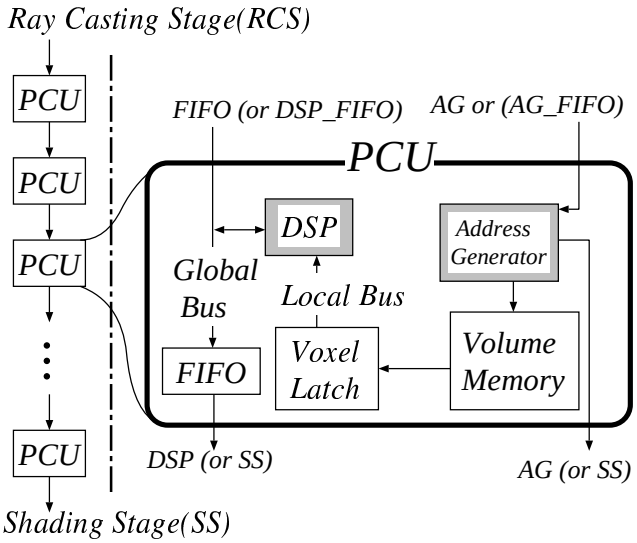


**Fig. 7** Pixel Value Calculation Unit(PCU)

uration of *ReVolver/C40* . The system consists of a control unit and a three-stage macro pipeline as follows. In *ReVolver/C40* , we adopts TI's Digital Signal Processor(DSP) TMS320C40[28] operating at 50MHz as processing element(PE) since it has useful internal configuration for volume rendering calculation.

**System Control Unit(SCU)** This unit gives other units various data/commands, such as volume data, viewing parameters such as positions of view point and screen, and rendering mode.

**Ray Casting Stage(RCS)** This stage casts rays which start from the view point and penetrate pixels on the screen, and send them to Pixel Calculation Stage. In order to generates a ray data in every machine cycle, this stage consists of 8 PEs. The unit of load distribution is one scanline and the scanlines are statically assigned to each PEs in cyclic fashion. Generated ray data is stored into a FIFO memory common to the 8PEs. The output of the FIFO memory is directly connected to the address generator mentioned below to feed ray data directly into Pixel Value Calculation Stage.

**Pixel Value Calculation Stage(PCS)**

This stage organizes a 128-stages linear pipeline of over operation. We call the unit correspond to one stage of this pipeline Pixel value Calculation Unit(PCUs). Each PCU is configured with Volume Memory(VM), Address Generator(AG), *C&T* Look-Up-Table(LUT), and one PE.

**Volume Memory(VM)**: The VM corresponds to the PTM node in Fig.4 and may contains upto 3sets†† of 4 $512^2$ planes out of $512^3$ volume data.

**Address Generator(AG)**: The Address Generator computes Volume Memory addresses of voxels to be sampled using viewing parameters, current pixel coordinate, and PCU's relative position in PCS. Since this kind of computation includes a lot of bit-based operations, we have implemented AG with FPGA(Xilinx XC4010D, 10,000 gates). All the information necessary to compute the address is transferred through the direct link between neighboring AGs, without any intervention by the processing element.

The AG also has an important function of the voxel preloading to hide the latency of the VM accesses, as follows: Voxels to be sampled are read out from the VM using the addresses generated by AG and are stored into the Voxel Latch (see Fig. 7) during the PE is doing the over operation for the previous pixel.

*C&T* **Look-Up-Table(LUT)**: Before the over operation, the sampled pixel value has to be translated into color and transparency. *C&T* Look-Up-Table(LUT) is provided with each PCU to do this translation. In our *ReVolver/C40* implementation, this table is allocated into the embedded memory located at local bus inside the DSP so that it could be accessed in one cycle.

**Processing Element(PE)**: As we have metioned before, we have adopted TI's DSP TMS320C40, we call it C40 in the rest of this paper, as our processing unit. It was because, 1) C40 has two physically separated 32-bit buses, called *Global Bus* and *Local Bus*, which support simultaneous access to the two different external/internal devices if they are located at different busses each other; 2)C40 has two embedded SRAMs which can be used as *C&T*

††These 3sets correspond to X-memory, Y-memory, and Z-memory in Fig.4, respectively

LUT and the storage are for program memory and runtime environments; 3)C40 has six embedded bi-directional communication ports with 20MB/s peak transfer speed for each which can be used at initial program loading and debugging in *ReVolver/C40* .

As it is shown in Fig.7, a FIFO buffer(32bits x 64entries) is placed on Global Bus. The output of this FIFO buffer is directly connected to the Global Bus of the C40 in the next PCU. The maximum throughput of this FIFO is 100MB/s. During volume rendering operation, the C40 picks up intermediate color and transparency data from the FIFO of its previous PCU, then it performs the over operations using preloaded voxels and stores the result into its local FIFO.

**Shading Stage(SS)** This stage consists of a $4 \times 4$ two-dimensional array of Shading Units(SUs). The SU consists of one C40 and two kinds of double-buffered memories: Receive Buffer(RB) and frame memory. The pixel value from the pixel calculation stage is distributed among 16RBs according to the pixel's coordinate. When all pixels belongs to an image get ready in RBs, this image is shaded using depth gradient shading, by default, if the user want to have better visual effect. The final image is stored into the double-buffered video memories distributed among 16 C40s. A video signal generation circuit collects sub-images generated by each C40s, converts them into analog video signal and sends it to a Disply.

### 3.3 PCB-Level System Configuration

We have designed four kinds of printed circuit board: SCU boards, RCS boards, PCS board and SS board. Each boards except for SCU boards are VME standard triple-height board and they contains 8 C40s each. So, 16 PCS boards for the PCS and 2 SS boards for the SS have to be mounted onto *ReVolver/C40* prototype. Fig. 8 shows a floor-plan of the prototype system.

### 3.4 Overview of Rendering Pipeline in *ReVolver/C40*

Fig.9 shows how the rendering pipeline of *ReVolver/C40* works. From the highest level, there is the frame-based pipeline where image generation, shading, and image output organize a 3-stages pipeline. The last two stages are implemented by the well-known double buffering of the frame memory in the SS. The second level is pixel-based pipeline that exists in the image generation stage. This pipeline consists of the ray casting stage and the N-stages linear pipeline of over operation. Though the RCS itself utilizes scanline-based parallelism, it is not visible to the image generation

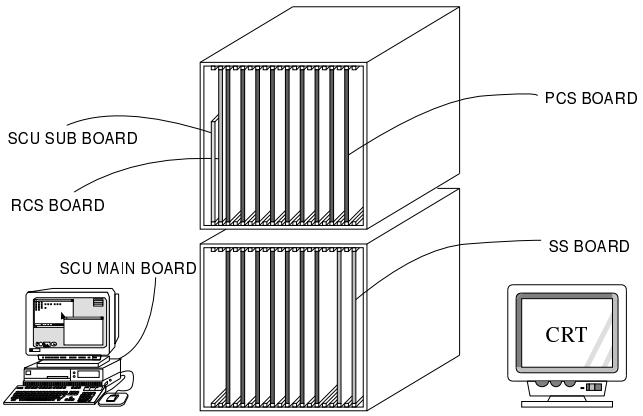


**Fig. 8** A Floor-plan of the Prototype System: ReVolver/C40

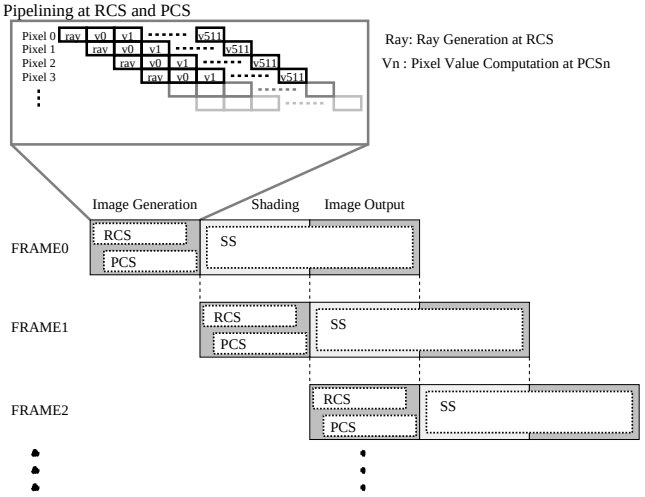


**Fig. 9** Rendering Pipeline

pipeline as long as there exists ray information in RCS's output FIFO.

### 3.5 Development of *ReVolver/C40*

After 5years from the initial proposal of the ReVolver architecture[13], [14], a part of RCS board with 3PEs, one PCS board with 8PEs, and one SS board with 8PEs became operational in late 1998. Using these parts, we had constructed a evaluation subsystem which we called *ReVolver/C40-mini*. And then, in early 1999, we have reorganized it into a little bit larger system which we call *ReVolver/C40-demo*. *ReVolver/C40-demo* was configured with four PCS boards (32PEs) and one SS board(8PEs) but it does not have voxel preload function of PCS boards neither RCS boards. In *Revolver/C40-demo*, all the communication between neighboring PEs is performed through the embedded communication ports of TMS320C40 and the host computer(PentiumIII 800MHz) also functions as RCS. Due to the memory size restriction, *ReVolver/C40-demo* can

render upto $256^3$ volume at the price of declining frame rate.

## 4. PERFORMANCE EVALUATION

We had made detailed performance evaluation based-on *ReVolver/C40-mini* and the results ware summarized in [17], [18]. In this section, we have focused on the evaluation of PCS since the performance of PCS is the most dominant part in the overall performance. In the following discussion, we assume the image size of $256 \times 256$ unless noticed especially. And also we assume that each PCUs are responsible for 3-sets of $P$-$256 \times 256$ planes out of $256^3$ volume data where $\{P|1, 2, 4, 8\}$.

### 4.1 Rendering Performance

The voxel preloading is one of the most important issue in the implementation, we first examine the performance of the address generator(AG). Table2 shows the time spent for preloading P voxels along a ray. Since we have designed two kinds of AG, this table contains both of these performance. The type1 AG is the non-pipelined implementation of AG, while the type2 AG is the pipelined version where 1)ray-data read-out from previous FIFO, 2)Address Calculation, 3)Volume Memory Access, and 4)ray-data store into its FIFO are performed in pipelined fashion. From this table, we could figure out that type1 and type2 AGs have performance of 6.1FPS(Frame Per Second) and 29.3FPS, respectively, if P = 1. However, we could not achieve the goal of 30FPS for P=2 even with type2 AG. It is because we could not extract the peak performance of DRAM device whose ras-access time is 160ns.

**Table 2** Performance of AG in ReVolver/C40-mini

| # of planes per PCU(P) | Voxel Preload Time($\mu$s)/Frame Rate(FPS) | |
|---|---|---|
| | Type 1 | Type 2 |
| 1 | 2.52/6.05 | 0.52/29.34 |
| 2 | 2.80/5.45 | 0.80/19.1 |
| 4 | 3.36/4.54 | 1.36/11.2 |

The next thing to examine is the time spent for over operation which determines the pipeline pitch of the rendering pipeline. Table 3 shows per-pixel rendering time and frame rate for $256 \times 256$ image. In order to examine the effects of AG, this table contains the result of two different configurations of PCU: with and without AG. In case of PCU with AG, we used the Type-I AG since Type-II AG was a little bit unstable. In case of PCU without AG, not only the results of over operation but also ray-data have to be transferred between C40s and C40 also has to substitute for AG. From this table, we could confirm the great impact of AG on the rendering performance. Indeed, the PCU with Type-I AG performs roughly 2.5 times faster than the PCU without AG. However, comparing with table2, the rendering performance has declined. It wad due to the communication overhead of C40. Because of the implementation dependent critical timing of FIFO access and the restriction of C40 on the consecutive write accesses, roughly $2\mu$s was spent for communication between C40 and FIFO memories. As a result, effective throughput decreased to only 20MB/s which is comparable to the embedded communication port of C40.

**Table 3** Effect of AG in ReVolver/C40-mini

| # of planes per PCU (P) | Rendering Time ($\mu$s/pixel)/Frame Rate(FPS) | |
|---|---|---|
| | with AG(Type I) | without AG |
| 1 | 3.28/4.65 | 8.31/1.84 |
| 2 | 4.80/3.18 | 11.96/1.28 |
| 4 | 7.24/2.11 | 19.11/0.80 |

In the above discussion we assumed *ReVolver/C40-mini*. From, now we continue our discussion based on the performance of *ReVolver/C40-demo*. Table 4 shows the rendering performance of *ReVolver/C40-demo*. In this table, the performance of simulated configuration that assumes a perfect AG (which always supplies voxel data in one cycle) is shown as well as the normal configuration without AG. The normal configuration without AG has better performance in *ReVolver/C40-demo* than in *ReVolver/C40-mini*. It is due to the very high internal I/O throughput(100MB/s) of communication ports in C40. When we assume the perfect AG, the performance reaches to 11.6FPS and seems to be saturated when P ¡ 4. From this results, we could assume the effective throughput of communication port between two C40s in *ReVolver/C40-demo* is 12.2MB/s. According to the hand-coded assembly program which assume the perfect AG, the peak performance of *ReVolver/C40-demo* could achieve about 20FPS for P=1 if there is sufficient bandwidth between two C40s.

**Table 4** Rendering Speed in ReVolver/C40-demo

| # of planes per PCU | Rendering Time(s/frame)/Frame Rate(FPS) | |
|---|---|---|
| | normal(without AG) | with perfect AG |
| 1 | 0.31/3.2 | 0.086/11.6 |
| 2 | 0.56/1.8 | 0.086/11.6 |
| 4 | 1.0 /1.0 | 0.094/10.7 |
| 8 | 1.8 /0.55 | 0.26/3.8 |

In the above discussions, we always assume the image size of $256^2$. Since the image size does not affect the rendering time for a pixel, the overall rendering performance is inversely proportional to the number of rays in the image. On the other hand, if there is enough capacity to store volume data in each PCUs, the overall rendering performance decrease with the number of planes (P) per PCU but it is much better than 1/P because the increase of the computational time for over operation reduces the communication overhead relative to the computational time.

### 4.2 Volume Data Initialization

This section discuss about an issue of volume data initialization. In advance to any rendering operation, volume data should be stored into volume memories of PCUs. In *ReVolver/C40-demo*, volume data stored in the hard disk of host computer is download sequentially as a set of four consecutive voxels. This is done through the communication link between SCU and PCU. Every time a set of four voxels is arrived at a PCU, the PCU performs the following operations: 1)It examines if some of them should be stored inside its volume memory according to the voxel's (x,y,z)-coordinate and relative location of the PCU. 2) It performs memory accesses if necessary. 3)It forwards them to the next PCU. The parallel trebled volume memory is initialized in this way. During the initialization, $9/4N^3$ memory accesses are performed in PCS. It takes about 500ns for each of this memory accesses including miscellaneous overhead in PCU. Indeed, for $256^3$ volume data, we have observed that it takes 19.8sec on the host machine including disk access time. Recent graphics cards for volume rendering[3], [7] also takes similar latency, though it is not negligible if the user want to change the volume data frequently. One solution to this problem is to have multiple pathes to download the data[23], [31].

## 5. CONSIDERATIONS

### 5.1 Consideration on Advances in Technology

As we mentioned before, the design of *ReVolver/C40* was based on the technology roughly 10 years ago. So we are going to review *ReVolver/C40* from the view point of technology advances.

1. Processor Technology

We adopts TI's DSP TMS320C40 as our processing element. The C40's features benefit *ReVolver/C40* are 1) hardware support for inverse and inverse square root which are very useful in the RCS, 2)parallel instructions which realize multiply-and-add like operations useful in all stages, 3)repeat block operation which uses single cycle branch very useful in PCS, for P=1 in particular, 4)embedded communication ports for parallel processing, 5)embedded small local sram block, and so forth.

Indeed, we didn't have any plan to use the embedded communication during rendering operation at a time of initial design, *ReVolver/C40-demo* exactly benefits from the low latency read/write operations to the communication ports. If they have much more throughput, like Alpha 21364[29], it would be very useful for fine grain communication like we need in pixel-based rendering pipeline.

Embedded small capacity first memory is quite useful in the applications which does not necessarily require high precision calculation. Since some function calls can be replaced by a single cycle memory lookup, like $C\&T$ LUT in PCU.

But thing for C40 is that it does not have multimedia extension instruction like quadruplet instructions which can be used to compute color value. If C40 would have such an instruction, performance of over operation in PCS would tripled. Since the clock frequency of the latest processor is more than 50 times faster than C40, we could achieve 150 times speedup totally if we could use all there latest technologies.

2. Memory Technology

The volume memory in PCU is 4MB DRAM of 32bit width. It's ras access latency is 160ns and has throughput of 66.6MB/s. Now, we could purchase 512MB DDR-SDRAM of 64-bits width, 2666MB/s, at the similar price. This memory is large enough for volume data of $512^3$ and has acceptable bandwidth for $512^2$ screen. However, ras access latency for random accesses is still far beneath the required speed. So, still there is need for parallel volume memory organization even with the latest memory technology in order to realize view-independent real-time volume rendering using perspective projection. However, if we allow view-restricted or view-dependent rendering, there are some possibilities to reduce the randomness in memory access patterns: cache blocking, or tiling, technique is one of such possibility.

3. FPGA and Dynamic Reconfigurable Tech.

The most powerful FPGA we used in *ReVolver/C40* is Xilinx XC4010D which has 10,000 equivalent gates, operating at 25MHz in our system. It was a biggest FPGA reasonably available at that time. Since it was impossible to implement both AG and circuits for over operation in one chip, we had decided to employ DSP for over operation. Now, we have various choices of versatile FPGAs. Some FPGAs have embedded processor(or DSP)-cores running 300MHz and up, some have embedded synchronous memory block accessible at 150MHz and up, and some have more than 10 million gates. If we can use such a FPGA device, we could implement a few tens of PCU including AG in one FPGA with a few channels of DDR-SDRAM Memories. Indeed, we are currently designing such a system in our laboratory[27]. In addition to that, if we could utilize the latest technique on dynamic reconfigurability[30], it is possible to consider a graphics processor which can dynamically adapt itself to meet the requirement of rendering conditions, like projection modes, volume data structures[19], lighting and shading parameters, and so on.

### 5.2 Consideration on Volume Rendering Algorithm

The fundamental algorithm we used in *ReVolver/C40* is the ray casting algorithm. Due to its simplicity and its image quality, almost all of the hardware accelerated volume rendering systems[2]–[4], [9]–[11] adopt this al-

gorithm. But none of these system support parallel volume rendering using perspective projection, except *ReVolver/C40* .

Recent advances in PC-based commodity graphics cards which support $\alpha$-*blending* for rendering non-opaque polygon realize the texture-based approach to the volume rendering[7]. If a whole volume data can be stored inside video memory of such a graphics card, it can perform volume rendering with acceptable frame rate. However, such a commodity graphics card does not have enough video memory to store large volume data, so it significantly slows down for large volume data due to the slicing effect. Some systems have proposed to alleviate this effect by parallel processing[8], [26].

There are some other algorithms[12] for parallel volume rendering, like splatting and cell projection[6]. These algorithms have a capability of volume rendering with unstructured grid volume data, though, they are difficult to implement in hardware due to some kind of complicated list processing to keep the order of over operation to intermediate sub-images.

### 5.3 Consideration on Human Visual Perception

So far, we have intended not to consider the approximation techniques in image generation because we thought it is not acceptable in medical applications which we thought the biggest market of volume rendering. However, there are a lot of possibilities to increase rendering speed if we take the human visual perception into account. Here we consider three types of approximation.

1. Early Ray Termination[20], [21]

During the composition computaion, if the accumulated transparency becomes very low, say $\epsilon$, the further accumulations do not affect the image too much. The early ray termination is a technique which cuts off the accumulation for the rest of sampling points on a ray if the accumulated transparency $< \epsilon$. This technique is know to be very useful. But it has no effect in current *ReVolver/C40* because even if ERT applies intermediate results have to be propagated to the end of rendering pipeline. If *ReVolver/C40* is configured with a feedback loop from the tail of rendering pipeline to the top, say from the last PCU to the first PCU, and we apply cyclic data distribution, we could benefit from the ERT[25].

2. Level of Detail

If the view point moves very quickly, it is impossible to recognize a detail of each images. So it is better to generate less detailed image to keep track of the view point movement in real time, rather than to generate full detailed image but less frame rate. Since this technique is easy to implement in *ReVolver/C40* , we have already examined in some cases and obtained very nice results.

3. Pseudo-Perspective Projection

If the screen size $S$ gets wider, parallel projection become no more acceptable. However, real-time rendering using perspective projection is fairly expensive to support, in general. So, we have been investigating a technique which we call *Pseudo-Perspective Projection*. Let's consider that the screen $S$ has been divided into sub-screens $s_{ij}(i, j : [0 : N-1])$ of equal size. For a sub-screen $s_{ij}$, calculate a ray $R_{ij}$ which goes through the center of $s_{ij}$. In *Pseudo-Perspective Projection*, for the pixels inside $s_{i,j}$, we cast rays parallel to $R_{i,j}$. If N=1 it is equivalent to the parallel projection. As N gets bigger, generated image gets closer to the image using perspective projection. Because of the increased ray coherency inside $s_{ij}$ in *Pseudo-Perspective Projection* mode, it can increase the regularity in memory access patterns. Though there exists trade off between rendering time and image quality, we believe it is a feasible approach to the real-time rendering using perspective projection.

## 6. CONCLUSION

We have described the architecture of *ReVolver/C40* a scalable parallel machine for volume rendering and its prototype implementation. The most important feature of *ReVolver/C40* is view-independent real time rendering of translucent 3D object using perspective projection. In order to realize this feature, we have proposed a parallel volume memory architecture based on the principal axis oriented sampling method and parallel treble volume memory. According the evaluation of the prototype systems, *ReVolver/C40* with 32 parallel volume memory is estimated to achieve more than 10 frames per second for $256^3$ volume data on $256^2$ screen. We have also confirmed the scalability of this system though we could not describe its detail in this paper.

Currently, we are continuing the work on the parallel volume rendering for large-data visualization[22] and interactive visualization of real-time simulation data[23], [24], [26], [27]. As a part of these works, we have been developing a new graphics card for parallel volume rendering[27]. In order to deal with the time-varying volume data, we are also developing a new rendering algorithm based-on the work in [16] which does not require any replication of volume data.

### Acknowledgment

Development of *ReVolver/C40* can not be accomplished without contribution of the following students who graduated Tomita Laboratory; H. Akashi, T. Kitasuga, T. Kuroda, M. Susukita, Y. Tsushima, T. Ogino, X. Jin, A. Nakayama, Y. Kiwata, T. Honda, I. Kinaka, M.Fujiwara, N. Yoshitani, D. Shigeta, F. Harase adn S. Yamauchi. We wish to express our appreciation for thier great work. And we also thank to M. Ikumo, M. Takayama, Y. Maruyama, S. Nakata, and the other members of Tomita Laboratory.

A part of this research was supported by the Grant-in-Aid for Scientific Research (A)(1)#06508001, (B)(2)#10558045, (B)(2)#13480083 from Japan Society for the Promotion of Science and also by the Grant-in-Aid for Scientific Research on Priority Areas (C)#13224050 from the Ministry of Education, Culture, Sports, Science, and Technology of Japan.

**References**

[1] Barthold Lichtenbelt, Randy Crane, and Shaz Naqvi, "Introduction to Volume Rendering," **Prentice-Hall, Inc.**, 1998.

[2] Hanspeter Pfister, et al., "Cube-3: A Real-tme Architecture for High-resolution Volume Visualization," **Proc. of 1994 Symp. on Volume Visualization**, pp. 75–82, 1994.

[3] Hanspeter Pfister, et al., "The VolumePro Real-Time Ray-Casting System", **SIGGRAPH 99 Conference Proceedings**, pp.251–260, 1999.

[4] Shigeru Muraki, et al., "VG Cluster: Large Scale Visual Computing System for Volumetric Simulations," SC2000 Research Gems (Poster), November 2000

[5] K.-L. Ma, et al., "Parallel Volume Rendering using Binary-Swap Image Composition," *IEEE Trans. on Computer Graphics and Applications*, Vol.14, No.4, pp.59–68, 1994.

[6] K.-L. Ma, et al., "A Scalable Parallel Cell-Projection Volume Rendering Algorithm for Three-Dimensional Unstructured Data," *Proc. of Parallel Rendering Symposium*(PRS'97), pp.95–104, 1997.

[7] C. Rezk-Salama, et al., "Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization", *Proc. of SIGGRAPH/EUROGRAPHICS Graphics Hardware Workshop 2000*, pp.109–118, 2000.

[8] Marcelo Magallon, et al., "Parallel Volume Rendering Using PC Graphics Hardware," *Proc. of Pacific Graphics*, 2001.

[9] Laurent Moll, et al., "Sepia:scalabel 3D compositing using PCI Pammete," *Proc. of the 7th IEEE Symp. on Field-Programmable Custom Computing Machines*, April 1999.

[10] G. Knittel and W.Strasser, "VIZARD – visualization accelerator for real-time display," *Proc. SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pp.139–146, 1997.

[11] M. Meissner, et al., "VIZARD II: A Reconfigurable Interactive Volume Rendering Systems," *Proc. of Graphics Hardware*, pp.137–146, Sept. 2002.

[12] Ken Brodlie and Jason Wood, "Recent Advances in Volume Visualization," *EUROGRAPHICS Computer Graphics Forum*, Vol.20, No.2, pp.125–148, June 2001.

[13] Hideya Akashi, "A Parallel Computer Architecture for Volume Rendering," Master Thesis, Dep. of Computer Science, Kyoto University, 1993(in Japanese).

[14] Yuji Tsushima, et al. , "A Parallel Computer Architecture for Volume Rendering:ReVolver," Trans. of IPS Japan, Vol.36, No.7, pp.1709-1718, 1995(in Japanese).

[15] Yuji Tsushima, et al. , "The Parallel Computer for Volume Rendering –ReVolver/C40–," Proc. of Joint Symp. on Parallel Processing 1995, pp.11-18, 1995(in Japanese).

[16] Jin Xidu, et al., "Super-high Speed Volume Rendering Architecture Based on Pixel Parallelism with the Restrained Visual Angle," *Journal of IPSJ*, Vol.38, No.9, pp.1668–1680, 1997.

[17] Naoki Yoshitani et al, "Performance Evaluation of Parallel Volume Rendering Machine ReVolver/C40", IPSJ SIG Notes, 99-ARC-132, pp.79-84, 1999(in Japanese).

[18] Naoki Yoshitani, "Implementation and Performance Evaluation of Parallel Volume Rendering Machine ReVolver/C40 ", Master Thesis, Dep of Computer Science, Kyoto University, Feb. 1999 (in Japanese).

[19] Masahiro Fujihara, et al., "Realtime Visualization Method for Hierarchical Grid Volume Data," *IPSJ Sig. Note.*, 98-ARC-128, pp. 7–12, Mar. 1998.

[20] Marc Levoy, "Efficient ray tracing of volume data" ACM Trans. on Graphics, Vol.9, Issue 3, pp.245–261, July 1990.

[21] Philippe Lacroute and Marc Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transformation," Proc. of SIGGRAPH'94, pp.451-458, July 1994.

[22] Satoshi Yamauchi, et al., "Active Volume Rendering with Simulation Steering," *IEICE Tech. Rep.*, Vol.101, No.216, CPSY2001-35, pp.1–8, July 2001.

[23] Fumiyasu Harase, et al., "Real-Time Volume Visualization on ReVolver/C40," *IPSJ Sig. Note.*, 2002-ARC-148, pp. 7–12, May 2002.

[24] Fumiyasu Harase et al, "A Hardware Assist for Scientific Visualization", Proc. of the 30th Symp. on Visualization, pp.119–122, 2002.7(in Japanese).

[25] Masataka Ikumo, et. al, "The Effects of Early Ray Termination in Parallel Volume Rendering with Cyclic Data Distribution," *Proc. of Forum on Information Technology*, Vol.3, J-96, pp.233–234, Sep.2002(in Japanese).

[26] Masataka Ikumo, et. al, "Parallel Volume Rendering Environment for Interactive Real-Time Simulation," *Proc. of IPSJ Kansai-branch Forum*, pp.121–124, Nov.2002(in Japanese).

[27] Masataka Ikumo, "Development of a Graphics Card VisA for Real-Time Visualization of Time-varying Volume DATA," Master Thesis, Grad. School of Informatics, Kyoto University, Feb. 2003.

[28] Texas Instruments: TMS320C4x User's Guide, 1993.

[29] Shubhendu S. et al., "The Alpha 21364 Network Architecture," *IEEE Micro*, pp. 26-35, January 2002

[30] Hideharu Amano, "A dynamically adaptive hardware using a multi-context reconfigurable processor," *IPSJ Sig. Note.(2002-ARC-150)*, Vol.2002, No.112, pp.59–64, Nov. 2002.

[31] Ken'ichi Itakura, et al., "Parallel visualization for parallel data stream," *Proc. of JSPP2001*, pp.189–196, June 2001.