Viewing and Rendering Processor for a Volume Visualization System

A. Kaufman, R. Bakalash and D. Cohen

Department of Computer Science
State University of New York
Stony Brook, NY 11794-4400, USA

Abstract

The architecture and the hardware realization of the *3D Viewing and Rendering Processor* is presented. This processor is a component of the Cube architecture, developed primarily for volume visualization. The processor generates 2D shaded orthographic, parallel, and perspective projections of the volumetric image of n^3 voxels in $O(n^2 \log n)$ time. This performance is attributed to a unique skewed memory organization, a special ray projection bus, an extended viewing architecture, and a new *congradient* shading technique. A reduced-resolution prototype has been realized in hardware using printed circuit board technology and has been running in true real time. Currently, a VLSI version of the prototype is being tested.

1. Introduction

The *Cube Architecture* [8] is a versatile voxel-based architecture for three-dimensional (3D) volumetric graphics. The 3D objects are stored in a large 3D *Cubic Frame Buffer (CFB)* of voxels. A voxel, which is a unit volume cell, has a numerical value that characterizes its density, color, texture, opacity ratio, and the like. Cube is a multiprocessor system with several processors accessing the CFB to input, manipulate, view, and render the CFB images. The *3D Frame-Buffer Processor* acts as a channel for inputting 3D scanned voxel images, which are the primary sources of CFB data. Once the images are in the CFB they can be manipulated and transformed by the processor, which operates as a *voxblt* engine [7] (i.e., an extended 3D *bitblt* engine). It also acts as a monitor for 3D interaction. A geometric model is another source of CFB data. The *3D Geometry Processor* scan converts (voxelizes) 3D geometric models into their 3D voxel representation within the CFB, possibly intermixed with the scanned data. The *3D Viewing and Rendering Processor (VP3)* generates 2D shaded orthographic, parallel, and perspective projections of the CFB image of n^3 voxels in a conventional 2D frame buffer in $O(n^2 \log n)$ time. This paper focuses on the VP3 and its hardware realization.

In order to manage the huge quantity of voxels and still perform in real time, the VP3 is assisted by parallelism and speed-up mechanisms. Particularly, two unique features have been incorporated within the architecture: a skewed parallel memory organization, which permits the retrieval and storage of voxel rays in *arbitrary* directions, and a ray projection process, which exploits a multiple-write bus to select the voxel closest to the observer along the ray. These two key constructs of the Cube architecture reduce the 3D problem involving voxels to a 2D problem involving rays of voxels. Other key constructs of the architecture include three 2D buffers to assist in generating arbitrary projections and a shading unit that is gradient-based and table-driven.

2. Skewing Memory Schemes

A CFB of n^3 voxels is divided into n memory banks, each with n^2 voxels, in such a way as to allow, in one memory cycle, simultaneous conflict-free access to a full ray of n voxels from the CFB, accessing only one voxel from each bank. The practicality of the mapping from discrete 3D space onto the CFB memory banks depends primarily upon whether the mapping can be described by a simple formula. Consider first the case of beams, that is, rays parallel to a primary axis, as in orthographic projections. A voxel with discrete space coordinates (x, y, z) is mapped onto the k th bank $(0 \leq x, y, z, k < n)$ by the following simple skewing scheme

$$k = (x + y + z) \bmod n. \tag{1}$$

Since two coordinates are always constant along any beam parallel to a primary axis, the third coordinate guarantees that only one voxel from the beam resides in any one of the banks. The internal mapping (i, j) within the bank is simply

$$i = x, \qquad j = y. \tag{2}$$

With this mapping, conflict-free access to beams of voxels along any one of the six orthographic directions is provided. These six directions are along the positive and negative directions of rows, columns, and axles, that is, parallel to the primary axes $\pm x$, $\pm y$, and $\pm z$. A whole beam of voxels is retrieved from the skewed storage space by a process of de-mapping the voxel beam back into 3D discrete space. A voxel within bank k is de-mapped onto the voxel beam (e.g., $y = y_0$, $z = z_0$, i.e., parallel to $+x$ axis) in 3D discrete space in position

$$D_x = [k - (y + z)] \bmod n \tag{3}$$

along the beam.

When scanning the CFB beam after beam for viewing, the internal order of the banks along the beams of voxels changes. Actually, the bank index, which is the distance of a voxel along a beam from the viewing position, is either incremented or decremented by 1 modulo n when moving to the next beam. This change is controlled locally by the local address unit of the bank. The initial assignment of the bank indices depends only upon the viewing direction. Consequently, even the simple arithmetic involved in Equation 1 is avoided during the time-consuming viewing process.

Consider now the following linear skewing scheme of the CFB

$$k = (ax + by + cz) \bmod n' \qquad 0 \leq k, x, y, z < n' - 1 \tag{4}$$

where $n' \approx n$, and a, b, c, and n' are selected in such a way as to guarantee conflict-free access to all the desired directions. Equation 4 allows for rays in nonorthographic directions to be retrieved from the CFB conflict free. For example, the linear skewing scheme

$$k = (5x + 2y + z) \bmod 517 \qquad 0 \leq k, x, y, z < 517 \tag{5}$$

provides conflict-free access to 26 parallel directions, with projection time complexity identical to that of the orthographic projections, i.e., $O(n^2 \log n)$. The 26 projections are the six orthographic projections (along rows, columns, and axles), eight parallel projections along the major (principal) diagonals and antidiagonals, called *corner projections*, and 12 parallel projections along the minor diagonals and antidiagonals, called *edge projections*.

3. Projections

The ray projection mechanism of the VP3 exploits a special common bus, called the *Voxel Multiple-Write Bus (VMWB)* [8]. The projection logic is composed of n identical processing units. A full beam of n voxels fetched from the CFB along the viewing direction is placed into the processing logic, with each voxel in its associated processing unit, for the selection of the nontransparent voxel closest to the assumed observer. All the units holding nontransparent voxels "race" for the bus by comparing their depth index bit after bit, starting from the most significant bit, and the unit holding the voxel closest to the observer succeeds after log n steps. For every beam, the projection mechanism outputs two values associated with the closest voxel: its value (e.g., color) and its depth (e.g., distance). These two values are passed directly to the shading unit.

The memory, addressing system, and projection units have an overall modular structure comprising a sequence of identical modules indexed $k = 0,1, \cdots, n-1$. These modules are interconnected with a barrel shifter allowing fast memory-to-memory transfer of beams in $O(\log n)$ time. Each module also includes a transparency unit, which is provided with transparency control parameters broadcast to all modules. Interval and depth control parameters are also broadcast and are compared with the locally computed depth measures (Equation 3) to generate depth sections or slices of the scene.

Arbitrary parallel projections can be generated in Cube in $O(n^2 \log n)$ time, by first transforming (rotating) the scene and then viewing it through a principal orthographic direction. With this technique, however, the CFB image is distorted each time a rotation is performed. Another technique that has been developed in Cube is ray casting [1, 6, 9-11]. It is suitable for both parallel and perspective viewing and has $O(n^3)$ complexity. Cube provides 26-, 18-, or 6-connected rays to accommodate objects in the scene that may have 6-, 18-, or 26-connected tunnels. Cube explores also the reconstruction of arbitrary parallel and perspective projections of convex objects from at least three orthographic projections and their respective depth buffers. The performance complexity is comparable to that of the orthographic

projection, but may increase with the complexity of the scene. In addition, the CFB skewing storage schemes, discussed before, allow for rays in nonorthographic parallel projections to be retrieved conflict free. They can also be exploited by perspective projections, conceivably through a multiple, but limited, number of fetches per ray.

An extended viewing architecture, which uses three additional 2D buffers, has been designed to accommodate arbitrary parallel and perspective projections. Since there is no direct way to fetch arbitrary discrete rays from the CFB conflict free, instead, a whole projection ray plane is fetched first. The plane is fetched as n beams (parallel to a primary axis) in $O(n)$ cycles, and stored in a 2D temporary buffer. This buffer is then rearranged in 26-connected discrete ray representations that correspond to the projection rays, using a $O(\log n)$ complexity barrel shifter twice. Each projection ray is then fed into the VMWB for the selection of the first opaque voxel along that ray in $O(\log n)$ time. The total time for an arbitrary projection is $O(n^2 \log n)$, allowing real-time arbitrary projection. This is the same order of magnitude as that of the orthographic projection using the original Cube architecture.

4. Shading

The *congradient* shading technique [3] has been developed for the Cube architecture emphasizing real-time performance. It capitalizes on the advantages of both gradient shading [5] and contextual shading [2] by defining the surface orientation of a voxel as one of a finite set of local gradients and, consequently, can employ a fast table-driven mechanism. The quality of the shaded images generated by the congradient shading technique can be favorably compared to other conventional techniques. The location of the light source can be arbitrarily changed, which provides a kind of motion parallax.

Unlike other voxel-based systems [4, 6, 9, 11] that perform the shading in a separate post processor, the shading unit in Cube is an integral part of the projection process, and each projected pixel is pipelined through the memory access, the projection unit, and the shading unit. The Cube congradient shading consists of only two stages, and uses a single 2D array, e.g., a frame buffer. The projection stage generates an array of congradient values each of which holds the concatenated values of the color and the local gradient. The production time of a congradient value is interleaved with the projection time of the next pixel. Since the congradient calculation time for each pixel is much less than its projection time, no additional time is accrued for the shading. The next stage is the actual display which is done with a standard hardware where the congradient values are used as indices to the look-up table, which translates a congradient value to its shaded color based upon the light source parameters. As a result, this two-tier shading unit is faster than the alternative techniques, and is simpler for hardware realization.

Two congradient shading approaches have been implemented: unidirectional shading, in which the gradient is computed only from the depth of the horizontal neighbors, and bidirectional shading, in which both horizontal and vertical neighbors are employed. In both approaches the architecture is very simple and is suitable for hardware realization [3].

5. Implementations

A software simulation of Cube has been running successfully on color Sun-3, Sun-4, Silicon Graphics IRIS, and HP Turbo SRX workstations, using various resolutions, such as 128^3, 256^3, and 512^3. A reduced-resolution hardware prototype of 16^3 voxels, 8-bit each, has been realized in hardware and has been integrated under an IBM-AT with a VGA graphics display controller. The hardware realization consists of 16 modules, each of which is implemented as a custom-built printed-circuit board containing a CFB module with its mapping and de-mapping addressing mechanisms, its projection processing unit, VMWB unit, translucency control, depth sectioning, etc. Each printed-circuit board includes a module of CFB memory of size 256 (16^2) bytes. There are 24 ICs on each board; all of them are CMOS components. The interface board, which is built out of 40 TTL ICs, controls the whole assembly of modules using the common bus.

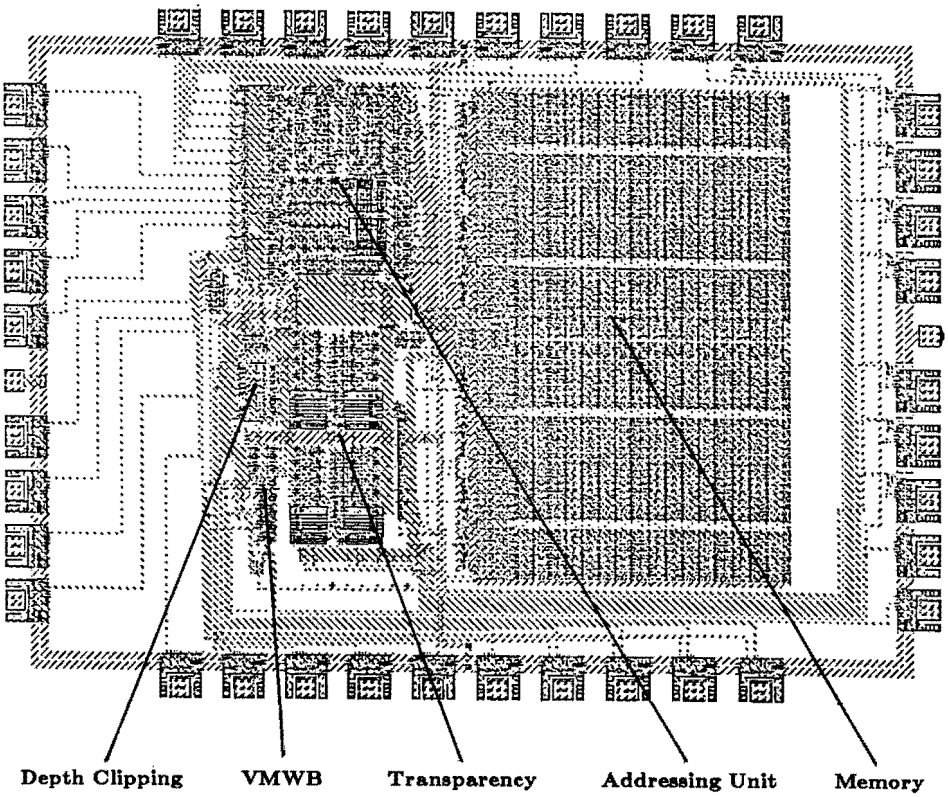

Figure 1: Layout of the VLSI design of the Cube chip

The prototype has been operating successfully in true real time. Measured performance figures on the prototype are 290 μsec for an arbitrary 3D rotation, 50 μsec for an orthographic projection, 330 μsec for an arbitrary parallel projection using rotation followed by orthographic viewing, and 160 μsec for arbitrary parallel or perspective projection using the extended viewing architecture. Estimated performance times for a 512^3 CFB resolution, using the same printed-circuit technology, are 511 msec for rotation, 62 msec for orthographic projection, and up to 573 msec for arbitrary projection. Using the VLSI technology will enhance the Cube speeds even further.

The modular structure of the Cube architecture is well suited for VLSI implementation. Currently a VLSI implementation is being pursued in which each board is fabricated as a single chip. Figure 1 depicts the layout of the VLSI chip design. The memory occupies the majority of the silicon area. The local addressing unit is the mapping and de-mapping circuitry of the skewed memory. The clipping unit is responsible for hither and yon clipping and slicing along the viewing direction. The transparency unit allows only those voxel values defined as opaque to be considered for the current operation. The VMWB is the unit that competes over the output bus for the selection of the nontransparent voxel closest to the observer. The various units of the Cube chip can be programmed by means of parameters which are broadcast by the central control unit.

The VLSI Cube chip has been designed on a Magic System, using SUN workstations, and fabricated with the MOSIS prototyping service. The Cube chip employs 2.0μ double metal CMOS/Bulk technology in a 40 DIP package. Figure 2 is a picture of the fabricated Cube chip. The whole IC assembly of 16 Cube chips occupies only part of a single board, which is shared with the interface unit.

The next prototype, which is under development, is a large real-time prototype for 256^3 resolution. The memory module will be separated from the logic and will be implemented using off-the-shelf memory chips. In order to reduce the fabrication cost and foster compaction of the hardware, two or more of the logic modules will be grouped together on one custom-built VLSI chip. This prototype will consist of 256 units of memory and logic chip each, assembled on several boards. The overall size will be about the same as the first 16^3 printed circuit prototype.

Acknowledgments:

We are grateful to S. Sherman and Z. Xu for their contributions to the design and building of the Cube prototypes. This work was supported by the National Science Foundation under grant MIP 88-05130.

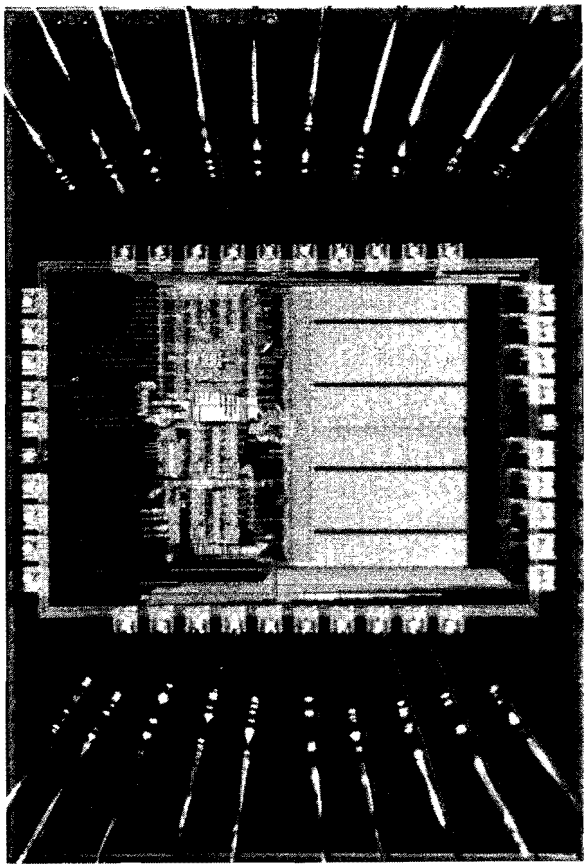

Figure 2: The fabricated Cube chip

6. References

1. Atherton, P. R., "A Method of Interactive Visualization of CAD Surface Models on a Color Video Display", *Computer Graphics*, **15**, 3 (August 1981), 279-287.
2. Chen, L. S., Herman, G. T., Reynolds, R. A. and Udupa, J. K., "Surface Shading in the Cuberille Environment", *IEEE Computer Graphics & Applications*, **5**, 12 (December 1985), 33-43.
3. Cohen, D., Kaufman, A., Bakalash, R. and Bergman, S., "Real-Time Discrete Shading", *The Visual Computer*, **6**, 1 (February 1990), 16-27.
4. Goldwasser, S. M., "A Generalized Object Display Processor Architecture", *IEEE Computer Graphics & Applications*, **4**, 10 (October 1984), 43-55.
5. Gordon, D. and Reynolds, R. A., "Image Space Shading of 3-Dimensional Objects", *Computer Vision, Graphics and Image Processing*, **29**, (1985), 361-376.

6. Jackel, D. and Strasser, W., "Reconstructing Solids from Tomographic Scans - The PARCUM II System", in *Advances in Computer Graphics Hardware II*, A. A. M. Kuijk and W. Strasser, (eds.), Springer-Verlag, Berlin, 1988, 209-227.

7. Kaufman, A., "The *voxblt* Engine: A Voxel Frame Buffer Processor", in *Advances in Graphics Hardware III*, A. A. M. Kuijk and W. Strasser, (eds.), Springer-Verlag, Berlin, 1989.

8. Kaufman, A. and Bakalash, R., "Memory and Processing Architecture for 3-D Voxel-Based Imagery", *IEEE Computer Graphics & Applications*, **8**, 6 (November 1988), 10-23.

9. Ohashi, T., Uchiki, T. and Tokoro, M., "A Three-Dimensional Shaded Display Method for Voxel-Based Representation", *Proceedings EUROGRAPHICS '85*, Nice, France, September 1985, 221-232.

10. Tuy, H. K. and Tuy, L. T., "Direct 2-D Display of 3-D Objects", *IEEE Computer Graphics & Applications*, **4**, 10 (November 1984), 29-33.

11. Uchiki, T. and Tokoro, M., "SCOPE: Solid and Colored Object Projection Environment", *Transaction of the Institute of Electronics and Communication Engineers of Japan*, **68-D**, 4 (April 1985), 741-748.