Building a Full Scale VLSI-Based Volume Visualization System

Reuven Bakalash, Arie Kaufman and Zhong Xu

ABSTRACT The hardware realization of an advanced prototype of the Cube volume visualization system, Cube-3, is presented. The primary hardware component of Cube is a viewing and rendering multiprocessor with distributed 3D voxel memory. Cube-3 design is based on our experience with two earlier prototypes: Cube-1 realized in hardware using printed circuit board technology and Cube-2 our first custom-designed VLSI implementation. Both prototypes are of reduced-size resolution (16^3) and can generate only orthographic views. Cube-3 is the next generation prototype of a full-scale resolution of 256^3 voxels. It has been functionally extended to generate non-orthographic projections, 3D real-time transformations, and shading. The ability to project and manipulate volumetric images in real-time is attributed to a unique skewed memory organization, a generalized skewed mapping, a special ray projection bus, a congradient shading technique, and a new barrel-shifting mechanism. This paper specifically describes the latter mechanism.

1 Background

The Cube architecture [11] is a voxel-based architecture for 3D volumetric graphics. A volumetric object is typically represented as a large 3D grid of volume elements, or as they are called in short, *voxels*. The voxels may be derived from discrete samples of the physical phenomenon, from a scientific or engineering simulation, or they may be synthesized by the computer from a geometric model.

The voxel representation is very effective for the traditional applications employing sampled 3D voxel imagery, such as medical imaging (e.g., CT, MRI, and ultrasonography) [2, 4, 5, 7], geology (e.g., seismic data) [18], biology (e.g., confocal microscopy) [8, 9], and molecular systems (e.g., electron density maps [6]). Cube also caters to the traditional 3D graphics synthesis applications, such as computer aided design (e.g., solid modeling) [10], 3D simulation and animation (e.g., instrumentation simulation, flight simulation), and scientific volume visualization (e.g., fluid dynamics [16, 17]). The Cube approach is further effective when sampled data are intermixed with geometric data [13].

The Cube architecture is organized around a large 3D cubic frame buffer (CFB) of voxels and is comprised of three processors. These processors access the CFB to input sampled and synthetic data [14], manipulate [12], project [15], and render [3] the CFB images. In order to manage the huge quantity of voxels, the Cube architecture is equipped with several special features, such as parallel processing, incremental algorithms, a modular memory for parallel access, and a modular multiple-write bus for speeding up the viewing process. The viewing bus, called the Voxel Multiple-Write Bus (VMWB), selects

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

the opaque voxel closest to the observer [11]. The selection time is proportional to the length in bits of the depth index, that is, $\log n$, where n is the resolution. It is implemented as a multiprocessor of n processing units.

A reduced-resolution hardware prototype, Cube-1, of 16^3 resolution was realized first. It is a printed-circuit 16-board hardware assembly driven by an interface board which is hooked-up directly to an IBM-AT bus. The prototype has been operating successfully in true real-time, generating 20,000 orthographic projections per second and over 3,000 arbitrary 3D rotations per second [11].

2 Cube-2—The First VLSI Implementation

The modular nature of the Cube architecture is well suited for VLSI implementation. Such an implementation enhances the operating speeds, minimizes the physical dimensions, and lowers the hardware cost. The Cube-1 circuit design was used as the basis for the VLSI chip design of the Cube-2 VLSI prototype. Each Cube-1 board (i.e., module) was converted into a single Cube-2 chip. Each chip consists of a memory module with 16^3 bytes, an addressing system with mapping and de-mapping mechanisms, a projection processing unit, a VMWB unit, translucency control, and depth sectioning. The circuit was redesigned to meet the special requirements of the VLSI.

Two major concerns that were met in the redesign of the Cube for VLSI are pin count and testability. In order to limit the pin count to 40, the I/O lines of the module were reduced by moving several functions from the interface board to the chip. It was done at the cost of replicating some of the interface logic circuits in all the modules. Testing the chip requires access to its internal functions: the memory, the addressing logic, the control status, etc., in addition to all the I/Os of the chip. This is in conflict with the pin count reduction. Therefore, nine test points were multiplexed with some input pins, with additional pin selecting test or input mode. Unlike the printed-circuit version of Cube-1, where the ID number of each module board has been set by proper wiring, in Cube-2 a special ID register has been added on the chip, which is initialized by a boot sequence.

The VLSI was designed on SUN workstations using Magic for the layout and simulation. The chips were fabricated by MOSIS in 2.0 micron double metal CMOS/Bulk technology, using a 40 DIP package. Each chip is comprised of a logic part, which occupies 40% of its space, and a 256×8 static memory on 60% of it. In total, there are about 14,000 transistors on the chip, with an overall area of 3.4×4.6^2mm. Figure 1 is a photo of the Cube-2 chip.

A special testbench has been constructed to test the basic functionalities of the chip. After correcting the first version and refabricating, we found that nine chips out of twelve were mostly functional. The final version of the VLSI chip is currently being fabricated. The whole Cube integrated circuit assembly occupies a single board, while the interface unit occupies another board. The Cube-2 system, located entirely on these two boards, will reside in an IBM-AT computer, which will be used as its host. The interface unit intermediates between the host and the Cube logic. It interprets commands given by the user and forwards them to the Cube logic board as control signals, initializes the identification numbers of the modules, creates scan sequences, and acquires the projection data from the CFB and moves it to the host. An operating environment, including software drivers and user interface, is being written on the IBM-AT, which uses its VGA graphics system to run the VLSI assembly as a satellite of an IBM-AT computer.

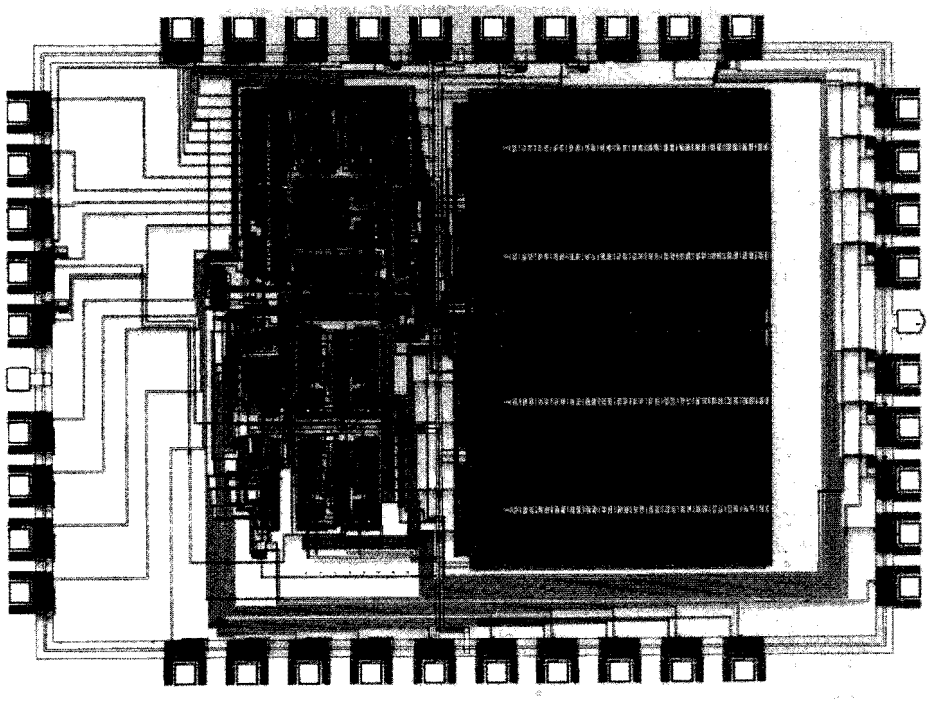

Fig. 1. Custom-designed VLSI chip of Cube-2

3 Cube-3—A Full-Scale Real-Time System

The next Cube generation is a VLSI-based prototype of a medium resolution of 256^3, called Cube-3. Each chip of this prototype will be of medium size $(6.9 \times 6.8)^2$mm with 84 pins, and will include two processing units. Consequently, the prototype will require a total of 128 identical chips that will work in parallel.

The hardware of Cube-3 will run all our software systems and applications developed within the Cube research framework in real-time. The Cube-3 prototype will allow us to complete the development cycle of the system. Its purpose is threefold:

- to examine the feasibility of implementing in VLSI a volume visualization system with a working resolution of 256^3, and to test it in real applications such as biomedical imaging, CAD, simulation, and scientific visualization;
- to measure the speeds attainable with such a resolution and to evaluate the system for interactive use;
- to test the hardware implementation of several new mechanisms developed recently for the Cube architecture.

The new mechanism includes: viewing from a non-orthographic direction using a generalized skewed mapping and parallel organization of the memory (see [15] for more details); transforming (e.g., rotating) 3D rooms (subcubes) by employing a barrel shifter; and discrete shading that employs a gradient-based table-driven technique [3].

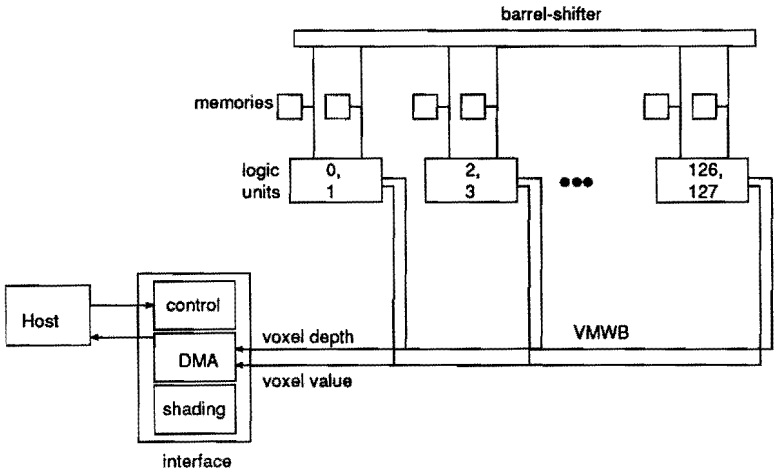

Fig. 2. The functional components of Cube-3

Cube-3 is a functionally extended version of Cube-2, which will include five basic components: interface, 256 logic modules, 256 memory modules, a barrel shifter, and a shading unit. The functional elements of the system are depicted in Figure 2. The interface, in addition to its basic intermediating function as in the previous versions, will control the barrel shifter and its data path, will manage a DMA communication with the host computer to speed up the transformation of the projected and shaded data, and will supervise the shading process.

Unlike Cube-2, the modular logic and the memory will be separated to minimize the cost, pin count, and chip area, and to increase reliability. The VLSI logic chips, two processing units on each, will include the addressing logic, the transparency and clipping controls, and the VMWB competition unit. The memory will be implemented using off-the-shelf standard 256^3 byte chips.

The barrel shifter is a network interconnection between 256 pairs of processors and memories. A simultaneous shifting of 8 bits of data in any arbitrary distance among the 256 modules would normally require 256 buses or a large crossbar matrix of 256×256 paths, 8 bit each. Since both alternatives are not realistic because of their complexity, a special interprocessor communication network was developed [1]. A significant reduction in communication area was achieved by implementing bidirectional moves. In such a case, rotation to the right by k units, when $k > 256/2$, can be performed with leftward rotation of $(256 - k)$ units. Another reduction was gained by dividing long rotations into smaller simultaneous rotational steps (but without the need for an intermediate reloading of the data). Thus, a rotation requires k/s steps (clocks), where s is the size of the step. The longest rotation is performed in $128/s$ clocks. Establishing such steps results in a desirable division of the entire communication network into small and modular building-blocks, implementable in VLSI.

The barrel shifter, shown in Figure 3, implements the divided modular mechanism based on a special flow-through network that interconnects among the modules. Its basic component is a modular barrel-switch unit (BSU) that is serving s pairs of clients (processors and memories), receiving from each a single bit of data (s bits in total) and shifting it on. A general width of w bits of data requires w duplicates of the BSUs at each group of

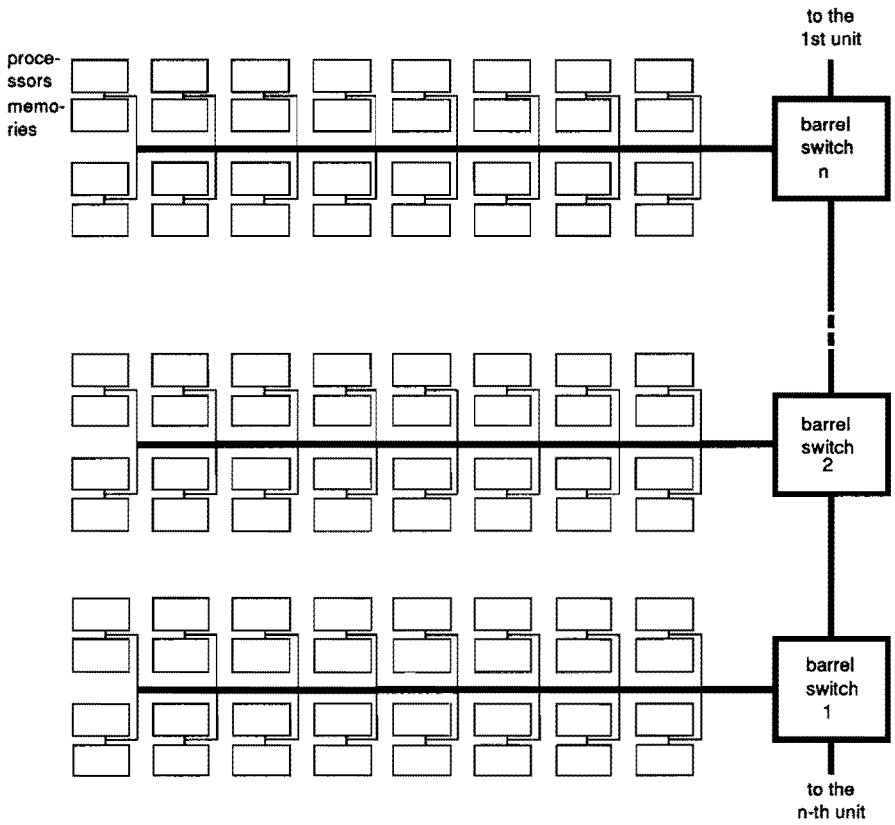

Fig. 3. The barrel-shift interconnection network

clients. This layout provides modularity and flexibility of the entire network. The network is extendible in its overall length, by serially connecting arbitrary numbers of BSUs, and in data width, by stacking up any number of BS units, one layer per data bit. The BSUs are organized in a simple and repetitive pattern of junction elements and connection paths, which is well suited for VLSI implementation. Each BSU will be implemented as a custom-designed VLSI chip. The barrel shift array counts about 2500 transistors. Adding the decoder, control circuits, and I/O pads to it totals 4000 transistors and 64 pins per chip.

Arbitrary data moves along the network have a unique meaning of real-time 3D graphics transformations. Typically, geometrical transformations in 3D systems are accomplished by a time-consuming matrix multiplication. In contrast, Cube employs the barrel shifter, a beam-based transformation in which intermodular data shifts along the beam result in a spatial transformation. For example, a 3D rotation about a primary axis is performed as a 2D rotation of beams parallel to the axis, where a beam is retrieved from the CFB, barrel shifted to reflect the rotation modulo 256 pertaining to the new location of the beam, and then pushed back into the CFB modules.

The real-time congradient shading [3], developed as part of the Cube project, will be implemented on the interface board. The hardware structure of the congradient shading

includes a triple shift register, which holds the depth and color values of the voxel being processed and a whole scan-line before and a whole scan-line after it, so that the voxel's two horizontal and vertical neighbors are readily available for computing the gradient. The shift registers are implemented by memory and additional addressing logic. Congradient shading is a table-driven, depth-gradient, volumetric shading technique, which uses a predefined set of surface gradients stored in a look-up table. Congradient shading is suitable for real-time hardware implementation, operating as a pipeline with the viewing process.

4 Summary

We described the complete sequence of the Cube hardware realizations. The first implementation—Cube-1—16^3 resolution using printed circuit technology, verified the basic concepts of the unique skewed memory, the parallel access, and the parallel processing of an orthographic beam. Since the modular nature of the Cube architecture is well suited for VLSI implementation, the next natural step was a 16^3 resolution custom-designed VLSI implementation. The experience accumulated with the two first prototypes has been applied to the design of the full-scale 256^3 system, Cube-3, which is currently under development. Cube-3 will be able to run in real-time the interactive Cube software environment, the 3D user interface, and the applications (e.g., biological [9] and medical [2]), developed as part of the Cube project.

Acknowledgements

This project has been supported by the National Science Foundation under grants MIP-8805130 and MIP-9049094 and grants from Hewlett Packard. We would like to thank David R. Smith for his invaluable advice concerning the VLSI design and testing.

References

[1] Bakalash, R. and Xu, Z.: Barrel Shift Microsystem for Parallel Processing, *Proc. Micro 23, 23rd Symposium and Workshop on Microprogramming and Microarchitecture*, Orlando, Florida, November 1990.

[2] Bakalash, R. and Kaufman, A.: MediCube: a 3D Medical Imaging Architecture, *Computers & Graphics*, 13, 2 (1989), 151-157.

[3] Cohen, D., Kaufman, A., Bakalash, R. and Bergman, S.: Real-Time Discrete Shading, *The Visual Computer*, 6, 1 (February 1990), 16-27.

[4] Farrell, E. J., Yang, W. C. and Zappulla, R. A.: Animated 3D CT Imaging, *IEEE Computer Graphics and Applications*, 5, 12 (December 1985), 26-32.

[5] Goldwasser, S. M., Reynolds, R. A., Bapty, T., Baraff, D., Summers, J., Talton, D. A. and Walsh, E.: Physician's Workstation with Real-Time Performance, *IEEE Computer Graphics & Applications*, 5, 12 (December 1985), 44-57.

[6] Goodsell, D. S., Mian, S. and Olson, A. J.: Rendering of Volumetric Data in Molecular Systems, *Journal of Molecular Graphics*, 7, (March 1989), 41-47.

[7] Hoehne, K. H., Bomans, M., Tiede, U. and Riemer, M.: Display of Multiple 3D-Objects using the Generalized Voxel-Model, *Proceedings of SPIE, Medical Imaging II*, 914, (1988), 850-854.

[8] Jense, G. J. and Huijsmans, D. P.: Interactive Voxel-Based Graphics for 3D Reconstruction of Biological Structures, *Computers & Graphics*, 13, 2 (1989), 145-150.

[9] Kaufman, A., Yagel, R., Bakalash, R. and Spector, I.: Volume Visualization in Cell Biology, *Proceedings Visualization '90*, San Francisco, CA, October 1990, 160-167.

[10] Kaufman, A.: Voxel-Based Solid Modeling, *Proc. International Conference on CAD/CAM and AMT*, Jerusalem, Israel, December 1989, 1.1.3-1-3.

[11] 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.

[12] 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.

[13] Kaufman, A., Yagel, R. and Cohen, D.: Intermixing Surface and Volume Rendering, in *3D Imaging in Medicine: Algorithms, Systems, Applications*, K. H. Hoehne, H. Fuchs and S. M. Pizer, (eds.), June 1990, 217-227.

[14] Kaufman, A. and Shimony, E.: 3D Scan-Conversion Algorithms for Voxel-Based Graphics, *Proceedings ACM Workshop on Interactive 3D Graphics*, Chapel Hill, NC, October 1986, 45-76.

[15] Kaufman, A., Bakalash, R. and Cohen, D.: Viewing and Rendering Processor for a Volume Visualization System, in *Advances in Graphics Hardware IV*, R. L. Grimsdale and W. Strasser, (eds.), Springer-Verlag, 1991, 171-178.

[16] Shirley, P. and Neeman, H.: Volume Visualization at the Center for Supercomputing Research and Development, *Proceedings of the Chapel Hill Workshop on Volume Visualization*, Chapel Hill, NC, May 1989, 17-20.

[17] Upson, C. and Keeler, M.: V-BUFFER: Visible Volume Rendering, *Computer Graphics*, 22, 4 (August 1988), 59-64.

[18] Wolfe, R. H. and Liu, C. N.: Interactive Visualization of 3D Seismic Data: A Volumetric Method, *IEEE Computer Graphics & Applications*, 8, 7 (July 1988), 24-30.