Comput. & Graphics Vol. 12, No. 1, pp. 91–97, 1988
Printed in Great Britain.

0097-8493/88 $3.00 + .00

DESIGN NOTES ON A SINGLE BOARD MULTIPROCESSOR FOR REAL-TIME CONTOUR SURFACE DISPLAY GENERATION*

MICHAEL J. ZYDA† AND ROBERT A. WALKER
Naval Postgraduate School, Code 52, Dept. of Computer Science, Monterey, California 93943-5100

Abstract—We present in this study a design for a VLSI multiprocessor capable of generating contour surface displays in real-time (one-thirtieth of a second). We begin by examining an application that requires real-time contour surface display generation. We sketch some outlines for an architecture based on a decomposable algorithm recently published. We then propose an architecture for a single board VLSI contour surface display generator that is pluggable into the Multibus of the Silicon Graphics, Inc. IRIS workstation.

1. INTRODUCTION

Contour surface display generation is one of the most frequently used applications graphics algorithms[1–5]. A contour surface display is a visual representation of a surface by the collection of lines formed when that surface is intersected by a set of parallel planes (Fig. 1). The lines formed on each of those planes are called contours. A contour represents the set of points that belong to both the surface and the particular intersecting plane. Contour surface displays are used in X-ray crystallography, computer-aided tomography, and other applications for which grid data is collected. Contour surface display generation is generally depicted as a computationally slow operation whose output is sent to a plotter or film recorder. One publication has described an architecture, and produced a feasibility determination for a VLSI based contour surface display generator[3]. This article is a distillation and condensation of that study.

1.1. *Contour surface display generation is slow*

Our initial premise for this study is that contour surface display generation on a single processor system, such as a graphics workstation with a floating point accelerator, is too slow to be of any use for interactive applications requiring such a display. This premise is based on [3] and is reinforced by statements found in the literature. Several papers have been written documenting "breakthroughs" that increase the speed of contour surface display generation. One author has reported that his contour surface display generation subroutine used one second of central processor time on NCAR's Control Data 7600[4]. Although a contour surface display generation program of this speed is useful for static situations, it is unacceptable for applications that generate a succession of contour surface displays in response to contour level changes read from a control dial. Such a program requires that a new contour surface display be generated, distributed, and displayed in real-time, typically one-thirtieth of a second for current display technology[6].

One application in which real-time contour surface display generation is important is the determination of molecular structures from the electron density data generated by X-ray crystallography[1]. Such an operation is executed interactively by using a computer graphics program that displays a Dreiding (stick) model of the molecule inside a contour surface display of the corresponding region of the molecule's electron density grid. In addition to the graphics function, the computer program monitors a series of signals generated by the user, while the user is turning the various knobs on a control console. The values read from these knobs are interpreted by the program as modifications to either the molecule or the surface display. Modifications to the molecule take the form of bond rotations or bond lengthenings. Modifications to the contour surface display take the form of an increase or decrease of the contour level. The goal of this process is to produce the stick model of the molecule that best fits inside the given electron density data set. The user can determine whether the model fits the density grid by modifying the contour level, shrinking the contour surface to the molecule. Similarly, the user can expand the contour surface from the stick model for better visibility. This function requires that the hardware has the capability to rapidly change the contour display as its contour level changes.

We know from [3] that the generation of a contour surface display, such as those required by the above application, cannot be completed in real-time using a conventional uniprocessor. The reason for this failure is that contour surface display generation algorithms require many more instructions executed per second than can be provided by currently available uniprocessors. In the past, this limitation of the conventional processor has relegated such applications to either the non real-time environment (waiting a few minutes for each display), or to the equally unsatisfying environment of motion picture film. Because of this, the authors of [3] looked for a VLSI multiprocessor solution to the real-time contour surface display generation problem.

* This work has been supported by the VHSIC Program Office, the U.S. Army Combat Developments Experimentation Center, Fort Ord, California and the Naval Ocean Systems Center, San Diego. It is a condensation of [8], which is available by request to the Department of Computer Science at the Naval Postgraduate School.

† Contact author.

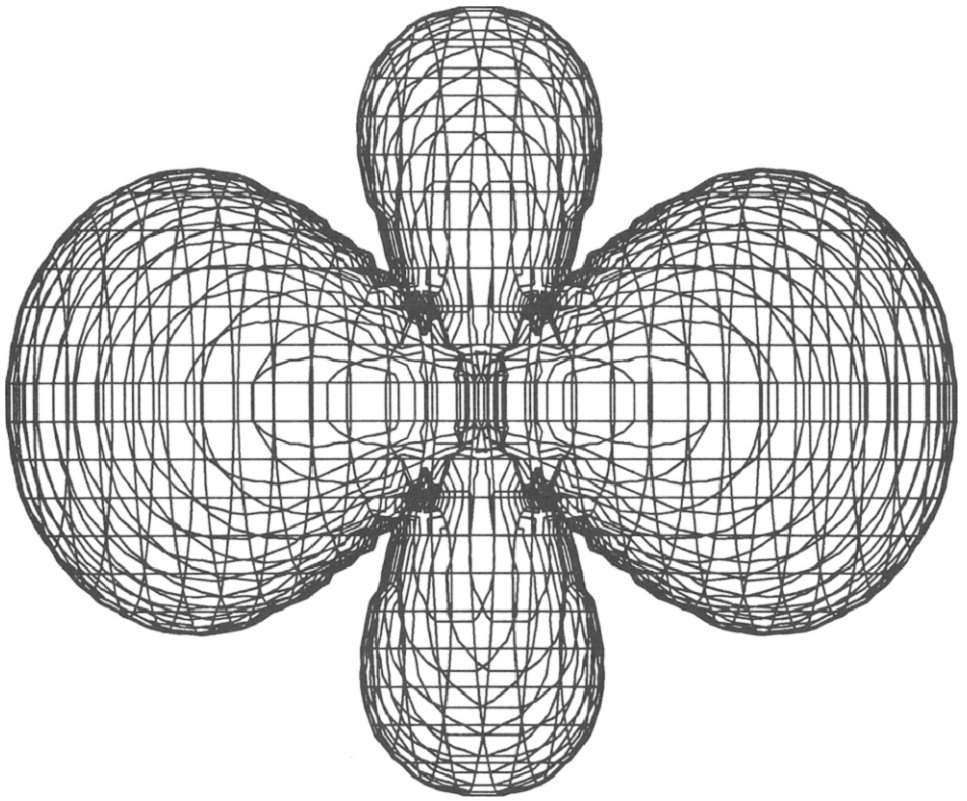


Fig. 1. Contour surface display generated from a hydrogen atom wavefunction squared (3dz2 orbital).

1.2. *Contouring definitions*

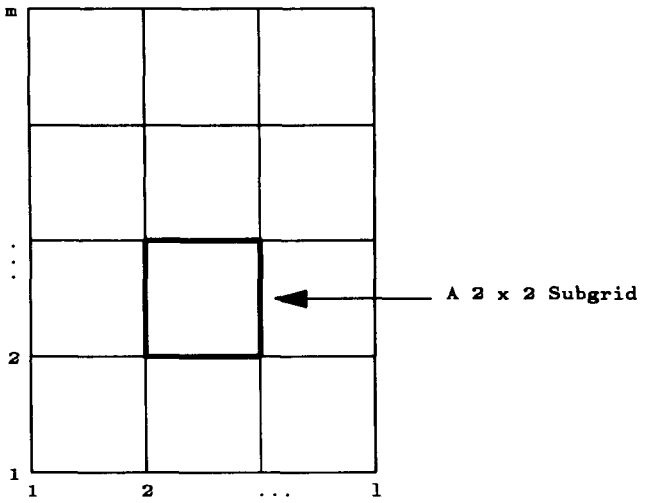
A contour surface is a visual display that represents all points in a particular region of three-space $\langle x, y, z\rangle$ which satisfy the relation $f(\langle x, y, z\rangle) = k$, where k is a constant known as the contour level (Fig. 1). The function f represents a physical quantity defined over the three-dimensional volume of interest. The visual display created by this algorithm is the collection of lines that belong to the intersection of both the set of points that satisfy the relation $f(\langle x, y, z\rangle) = k$, and a set of regularly spaced parallel planes that pass through the region of three-space for which the relation is defined.

For this study, the function f is approximated by a discrete, three-dimensional grid created by sampling that function over the volume of interest. The three-dimensional grid contains a value at each of its defined points that corresponds to the physical quantity obtained from the function, i.e., the value associated with point (x_0, y_0, z_0) is v_0, where $f(x_0, y_0, z_0) = v_0$.

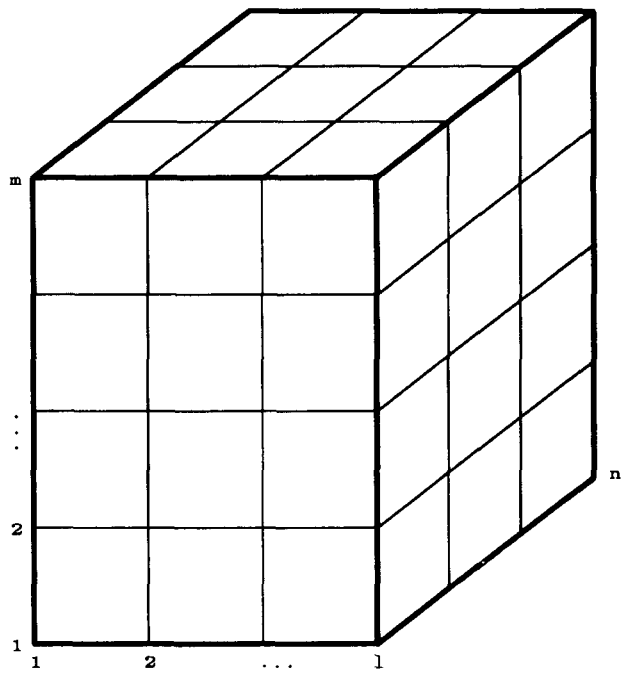
A decomposable algorithm for contour surface display generation is described in [5]. That algorithm is constructed from a two-dimensional contouring algorithm used to contour all the possible planar, orthogonal, two-dimensional grids of a larger three-dimensional grid. The contouring algorithm that works on the two-dimensional grid is comprised of components, called algorithm components. The algorithm components operate on individual 2×2 subgrids of the larger two-dimensional grid. (Note: a 2×2 subgrid is defined to be that portion of the two-dimensional grid bounded by four adjacent grid points.) In the algorithm, the computations necessary for generating the contour lines for a single 2×2 subgrid are independent from those required for any other 2×2 subgrid. If we compute the contours corresponding to contour level k for all 2×2 subgrids of a two-dimensional grid, then we will have determined the complete set of contours for that grid. If we compute the contours corresponding to contour level k for all possible 2×2 subgrids of the larger three-dimensional grid, then we will have the complete contour surface display for that grid. The assemblage of the contours created by this process, i.e., the simultaneous display of all the contours created for all 2×2 subgrids of the larger three-dimensional grid, produces a "chicken-wire-like" contour surface display (Fig. 1). The full development of this algorithm can be found in [5]. We refer to the results of those studies and, consequently, do not cover the algorithm here in great detail. We only note that for the largest three-dimensional grid of interest for the above application, a $30 \times 30 \times 30$ grid, this means the potential for 75,690 parallel operations (see Fig. 2 and [1]).

2. ARCHITECTURAL GOALS FOR THE CONTOUR SURFACE DISPLAY GENERATOR

The first goal in the design of the contour surface display generator is to build a system that meets the performance requirements, i.e., a new contour surface display computed from a $30 \times 30 \times 30$ grid, and delivered to a display device in one-thirtieth of a second. This is an ambitious goal but it must be noted that one-thirtieth of a second is the maximum amount of time allowable for the operation. Any longer amount



A 2D Grid of Size 1 x m Has
(1 - 1) x (m - 1) 2 x 2 Subgrids.



A 3D Grid of Size 1 x m x n Has
1 x (m - 1) x (n - 1) + m x (1 - 1) x (n - 1) + n x (1 - 1) x (m - 1)
2 x 2 Subgrids

Fig. 2. 2 × 2 subgrid count for 2D and 3D grids.

of time does not provide the viewer smooth transitions between successive contour surface displays. This goal says nothing about the load time of the 30 × 30 × 30 grid to the special piece of hardware that computes the contour surface display. Consequently, we allow solutions that pre-load the grid.

The second goal for the construction of the contour surface display generator is that we be able to plug it into an existing graphics system with minimal hardware and software changes. For the purposes of this study, the target graphics system is chosen to be the Silicon Graphics, Inc. IRIS workstation (see Fig. 3 and [7]). The Silicon Graphics, Inc., IRIS is currently the highest performance graphics system that best matches the selected application's goals.

3. ARCHITECTURE OF THE CONTOUR SURFACE DISPLAY GENERATOR

There is not enough space in this paper to present the complete architecture of the contour surface display generator. The reader is instead referred to [3]. An overview architectural description is that the contour

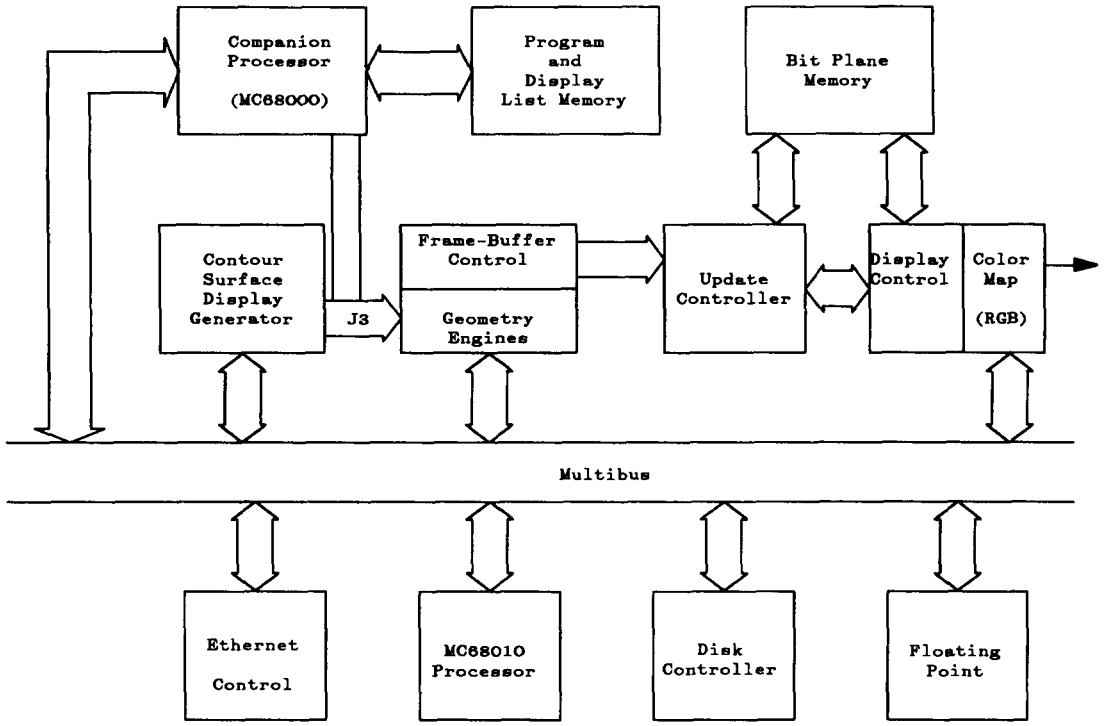


Fig. 3. Block diagram of the Silicon Graphics, Inc. IRIS (with additional, non-SGI contour surface display generator).

surface display generator is comprised of four subsystems: (1) the array of algorithm component processors, (2) the controller for that array of processors, (3) the algorithm component processor itself, and (4) the interface to the graphics system. Figure 4 shows how the four subsystems relate to the target graphics system.

Figure 4 depicts the array of algorithm component processors as a single box, with three connections to the outside environment: an input bus for contour levels and subgrids, an output bus for coordinates and drawing instructions, and a bus for controlling the array of processors. A dual bus configuration is chosen to maximize concurrency in the system. This is possible because of the autonomous nature of the input and output operations of the algorithm.

The input bus is the medium responsible for delivering subgrid definitions and contour levels to the array of algorithm component processors. Because this is the

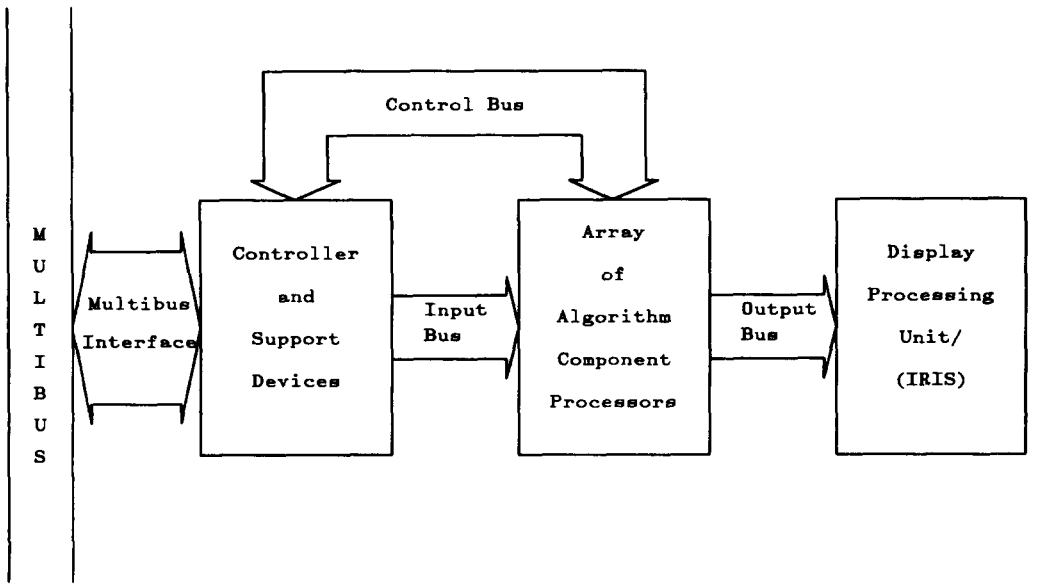


Fig. 4. Contour surface display generator: buses and data flow.

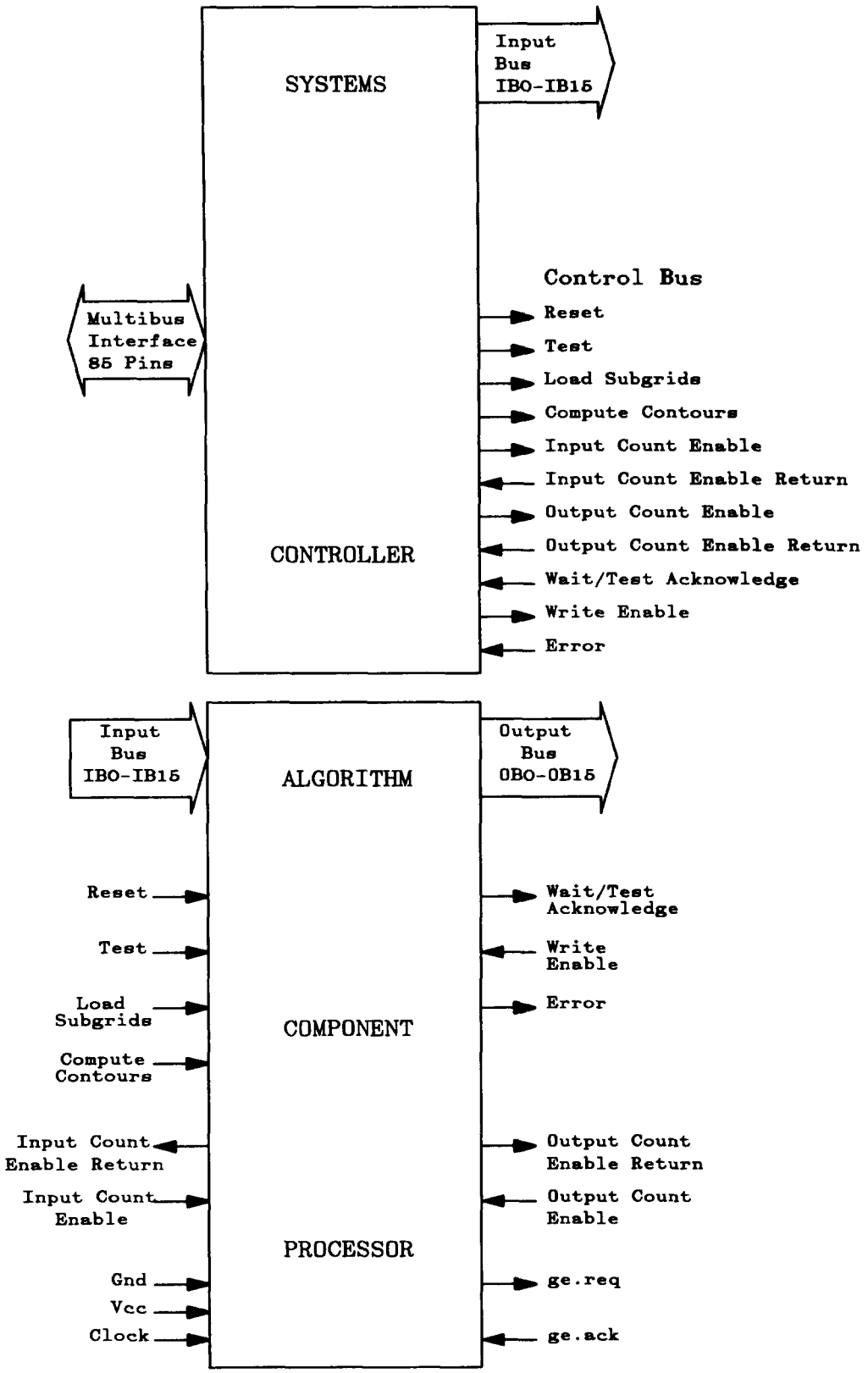


Fig. 5. Functional pin diagrams of the systems controller and the algorithm component processor.

only data required to be transmitted on the bus, the bandwidth of the input bus does not need to be high. The rate at which subgrid definitions are loaded into the algorithm component processors does not directly affect the real-time capabilities of the system. The real-time capabilities of the contour surface display gen-

erator are determined by the rate at which data can be produced in each algorithm component processor. This, in turn, directly affects the rate of output to the display processing unit. The output bus is responsible for delivering the coordinates and drawing instructions to the display processing unit.

The control bus for the contour surface display generator contains all the control lines necessary to manage the data flow on the input side of the system. Two additional control lines are required on the output side of the system to coordinate the two wire handshake between the algorithm component processors and the display processing unit (Geometry Engines).

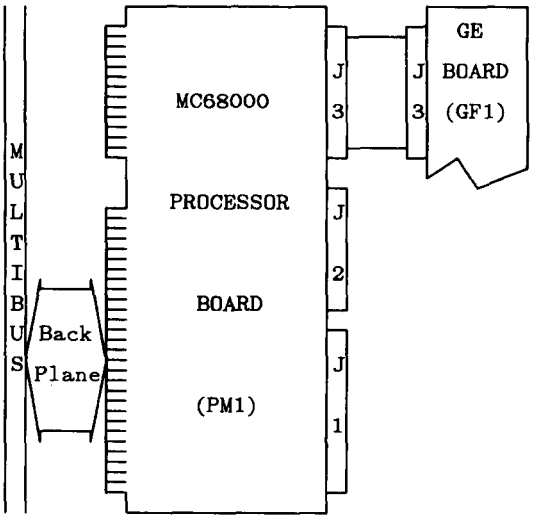
Control of the array of algorithm component processors involves the integration of several different components. The systems controller coordinates the operation of all other components (Fig. 5). The systems controller converts incoming signals from the Multibus bus master of the Silicon Graphics, Inc. IRIS workstation into signals that make sense to the algorithm component processors. The Multibus bus master is the board in the Multibus Backplane that places the commands on the Multibus. The systems controller is a slave in that it reacts to commands placed on the Multibus.

The component that is responsible for the production of the coordinates and drawing instructions for the contour surface display generator is the algorithm component processor (Fig. 5). Each of these processors is identical and functions independently. We do not go into great detail about that processor other than to state that the processor is a full microprocessor of the Motorola MC68000 class.

The contour surface display generator is connected to the Silicon Graphics, Inc. IRIS graphics system by the IEEE standard Multibus Backplane Bus[8]. This Multibus connection provides all inputs to the contour surface display generator. The Multibus interfaces to two different classifications of bus modules: (1) Masters—those modules that generate commands, and (2) Slaves—those that respond to commands. The parent processor (MC68000) is the Master module for the graphics system. The contour surface display generator is a slave module in that system.

The output of the contour surface display generator is to the Private Bus of the IRIS system (Figs. 3 and 6). The Private Bus is a unidirectional, 16-bit bus dedicated to the provision of coordinate and drawing instructions to the high speed Geometry Engines. Coordination of the transfer of data between the algorithm component processors and the Geometry Engines is done via a two line handshake protocol.

When the contour surface display generator is added to the system, a physical connection to the Geometry Engine pipeline must be shared by both itself and the system processor (the J3 connection of the Geometry Engine board). To enable the user to alternatively route processor and generator data to the Geometry Engines, a hardware switch is added to the system. This hardware provides the system with a way to multiplex the direct path of the Private Bus. A software switch then provides the control of the Private Bus' origin and configuration. This software switch establishes either a path from the contour surface display generator to the Geometry Pipeline, or a path directly from the MC68000 processor (Fig. 3).



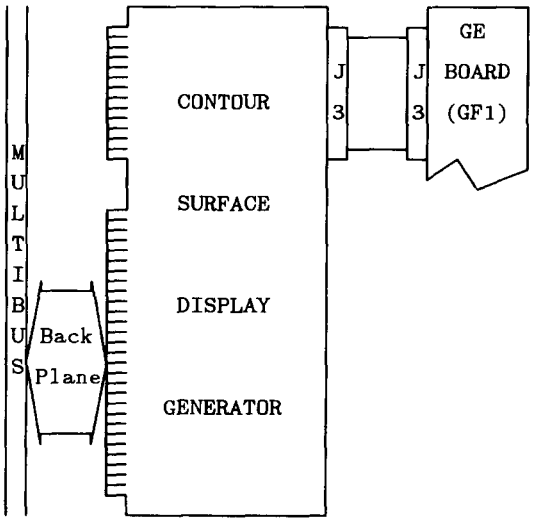


Fig. 6(a). Silicon Graphics, Inc. IRIS pipeline connection for the private bus (courtesy of Silicon Graphics, Inc.).
Fig. 6(b). The J3 pipeline connection of the Silicon Graphics, Inc. private bus for the contour surface display generator.

4. HARDWARE COMPLEXITY ESTIMATE

The above is a quick overview of the architecture of the contour surface display generator. One of the key components in this system is obviously the algorithm component processor. In [3], it is determined that 50 algorithm component processors are all that

are needed in the system to generate and deliver the average sized picture for a $30 \times 30 \times 30$ grid. To determine the feasibility of the complete system, we need a circuit complexity estimate for the size of the algorithm component processor. In [3], we find a summary of the quantity of transistor equivalent devices necessary. We note only that the total devices required for one algorithm component processor is about 660K devices. This number is well below the two million devices per chip level that is currently being produced in research laboratories[9]. For this level of chip complexity, the array of algorithm component processors can be built in less than 25 VLSI chips. At the ten million devices per chip level promised in [10], this is less than 5 VLSI chips. At either chip complexity level, this is a smaller quantity of chips than normally found on a single Multibus board. Consequently, the design of this system is within our goal of putting all elements on a single board pluggable into our workstation.

5. CONCLUSIONS

This study is intended to give an overview of the design for the real-time contour surface display generator described in [3]. We can be assured that once we produce such a system that the applications user will return to us with further hardware demands for either other algorithms, or additional real-time graphics display generation capabilities. There are several ways to respond to such demands. One way is to use the knowledge gained in the design of the contour surface display generator as a base for the design of the display generator for the later graphics algorithm. Such design efforts are expensive and must be justified by demand. An alternative is to step back from the contour surface display generator and attempt to generalize the architecture required such that other graphics algorithms with similar characteristics can use the same special hardware. This later approach is our current research direction. We hope to provide from this effort a series of single board (one or two board) multiprocessors useful for different *classes* of graphics applications algorithms. We are at an early phase of this effort and are working on defining those algorithm classes.

REFERENCES

1. C. D. Barry and J. H. Sucher, Interactive real-time contouring of density maps. American Crystallographic Association Winter Meeting, Honolulu, Poster Session (March 1979).
2. D. H. Faber, E. W. M. Rutten-Keulemans, and C. Altona, Computer plotting of contour maps: an improved method. *Computers & Chemistry* **3**, 51–55 Pergamon Press Ltd., Oxford (1979).
3. Robert A. Walker and Michael J. Zyda, An integrated systems architecture for real-time contour surface display generation. Technical Report NPS42-85-010, Department of Computer Science, Naval Postgraduate School, Monterey, California (1985).
4. T. Wright and J. Humbrecht, ISOSRF—an algorithm for plotting iso-valued surfaces of a function of three variables. *Computer Graphics: A Quarterly Report of SIGGRAPH-ACM* **13**(2), 182–189 (1979).
5. Michael J. Zyda, A decomposable algorithm for contour surface display generation. Technical Report NPS52-84-011, Department of Computer Science, Naval Postgraduate School, Monterey, California (1984).
6. William H. Newman and Robert F. Sproull, *Principles of Interactive Graphics,* Second Edition. McGraw-Hill, New York (1979).
7. J. H. Clark and T. Davis, Workstation unites real-time graphics with unix, ethernet. *Electronics* (October 20, 1983).
8. Intel Corporation Technical Publication, *Intel Multibus Specification,* Intel Corporation, Santa Clara, California (1982).
9. Micronews, IBM experimental million bit memory chip, *IEEE Micro* **4**(4), 191 (1984).
10. Leonard Uhr, *Algorithm-Structured Computer Arrays and Networks.* Academic Press, Orlando, Florida (1984).