LEVEL ✓ 12

SYNTACTIC VISUAL IMAGE GENERATION SYSTEM FEASIBILITY STUDY FINAL TECHNICAL REPORT

Jun-Nov 81 on Phase I.

December 1981

"The views, opinions, and findings contained in this report are those of the author(s) and should not be construed as an official Department of Defense position, policy, or decision, unless so designated by other official documentation".

Sponsored by

Defense Advanced Research Projects Agency (DoD)
ARPA Order No. 4278

Under Contract No. MDA903-81-C-0332, issued by
Department of Army, Defense Supply Service-Washington,
Washington, DC 20310.

Prepared by
Boeing Aerospace Company
Seattle, Washington

Effective Date of Contract 81 Jun 01
Contract Expiration Date 81 Dec 01
Reporting Period 81 Jun - Nov

059610

ABSTRACT

During

This report summarizes the research effort of phase 1 of the Syntactic Visual Image Generation program, Boeing has investigated several methods of implementing a pipelined multiprocessor/-/multimemory image generation system. Also, Boeing has assessed architectures for a fractal-object processor and has developed high level simulations to predict performance and architecture feasibility. Additionally, cost/performance analysis and possible VLSI implementation of the various architectures have been assessed.

SUMMARY

Computer graphics or computer image generation (CIG) systems employ various hidden surface elimination algorithms. Current real time CIG systems primarily use list priority schemes to eliminate hidden surfaces, while several non-real time computer graphics systems use the depth buffer (alias z buffer) approach. Current real time CIG systems process primarily planar polygons and produce simplistic, cartoonish scenes. However, recent research efforts in industry and academia have focused on processing data types other than polygons, including procedurally generated objects or textures (e.g. fractals). Complex scenes (sometimes almost realistic appearing) are being produced with non-real time systems using the depth buffer approach and using arbitrary data types (curved surfaces, textures etc.). Current real time CIG systems are used primarily for flight simulation visual systems. However, future potential applications of CIG technology include: missile retargeting, C^3I battle field management display systems, airborne navigation and display aids, and advanced CAD.

The questions are:(1) Is it feasible to develop a <u>real time</u>, depth buffer oriented CIG system capable of processing arbitrary data types (including fractals) to generate far more complex scenes than those produced by current real time systems?; and (2) if so, is there an architecture which permits a modular, scalable design and development of an upwardly compatible family of real time CIG systems for the various future applications?

The objective of phase I (feasibility study) of the Syntactic Visual Image Generation program was to investigate the technical issues related to the above questions and to provide a plan (if judged potentially feasible) for developing an architecture research system which would demonstrate the feasibility of a real time depth buffer system and also provide the technical background for developing such a system.

At the begining of the program Boeing established the following criteria for a real time, depth-buffer oriented CIG system architecture.

(1) Architecture maximizes the effective throughput of system processors.

(2) Architecture permits processing of arbitrary data types (including fractals).

(3) Architecture is modular and scalable to different requirements of eventual CIG family members.

(4) Architecture is suitable for VLSI implementation.

Boeing surveyed the literature pertaining to multiprocessing architectures in general, as well as computer graphics oriented multiprocessing systems. Crossbar switch based, common bus oriented and hybrid architectures were evaluated with respect to the above criteria. Broadcast, splitter tree and hybrid methods of distributing the computer graphics processing (Fuchs, Clark, Parke) were examined. In addition, Boeing formulated alternative architecture concepts, evaluated them with respect to the criteria, and simulated in software alternative distribution strategies. Boeing also addressed specific CIG issues including full color systems, depth precision and accuracy, and anti-aliasing.

Based on the analysis, evaluations and simulations, Boeing has concluded that the depth buffer approach appears feasible for real time CI systems. Moreover, Boeing has identified an architecture (distribution tree) which meets the above throughput and scalability criteria and will offer the potential for greater scene detail than existing real time CIG systems through the use of a variety of data type processors, including procedurally generated, fractal textures. Process distribution schemes were identified which will balance the processing load in a nearly uniform fashion and will therefore minimize queue lengths within the distribution tree. Boeing's investigation concerning depth precision led to abandoning the use of z (in an x,y,z screen coordinate system) as depth and adopting the use of a floating point format of 1/z as depth.

Boeing decided that a full color architecture research system should be developed during phase II. This research system will be capable of generating complex scenes consisting of 50,000 - 200,000 faces in near real time; will be capable of supporting a variety of data types, including fractal textures; and will demonstrate the distribution tree architecture and its scalability and throughput enhancement features.

Boeing is convinced that the development of such an architecture research system during phase II will provide the necessary background to develop a family of real time depth buffer oriented CIG systems for a variety of military applications having vastly different throughput requirements.

Phase I - Feasibility Study Objectives

The objectives of the feasibility study were governed by the long-term objective of developing a family of real time CIG systems. The performance requirements of the family could range from a 5Hz, 256 x 256 display, monochrome system on the low end to a 30Hz, 1024 x 1024 display, full color system on the high end, with each system permitting a depth complexity of 4. In addition, the family would be upwardly compatible, sc lable and modular in design, and would permit arbitrary object data types and processors (e.g. planar polygon, textures, and curved surfaces) to be interfaced into the system as required. The high end system could generate and display a 1024 x 1024 pixels scene. However, reducing the sampling effects (jagged edges etc....) would require that a 2048 x 2048 pixels scene be computed. This scene would then be anti-aliased or filtered down to 1024 x 1024 pixels. For a 30Hz frame update rate and depth complexity of 4, a total of 2048 x 2048 x 4 x 30 = 480 million tiles would be output from the tilers every second (2 nano seconds/tile). Consequently, a multiprocessor/multimemory module architecture is required where the number of tilers (processors) and memory modules is dependent upon the absolute and relative speeds of each.

Architecture Research Effort

The primary effort of the Phase I feasibility study focused on identifying a multiprocessor/multimemory module architecture which met the four major criteria mentioned above. Associated with this type of design are the inefficiencies generated by processor overhead and memory contention. Anastas and Vaughan (8) have researched the general multiprocessor/ multimemory module contention problem extensively. The following graphs (figures 1 and 2) were constructed using data from a computer program created by Vaughan and Anastas. The model they selected is the binomial approximation of Strecker (13) which was also analyzed and found to work well in all cases by Baskett and Smith (14). Figure 1 reflects the

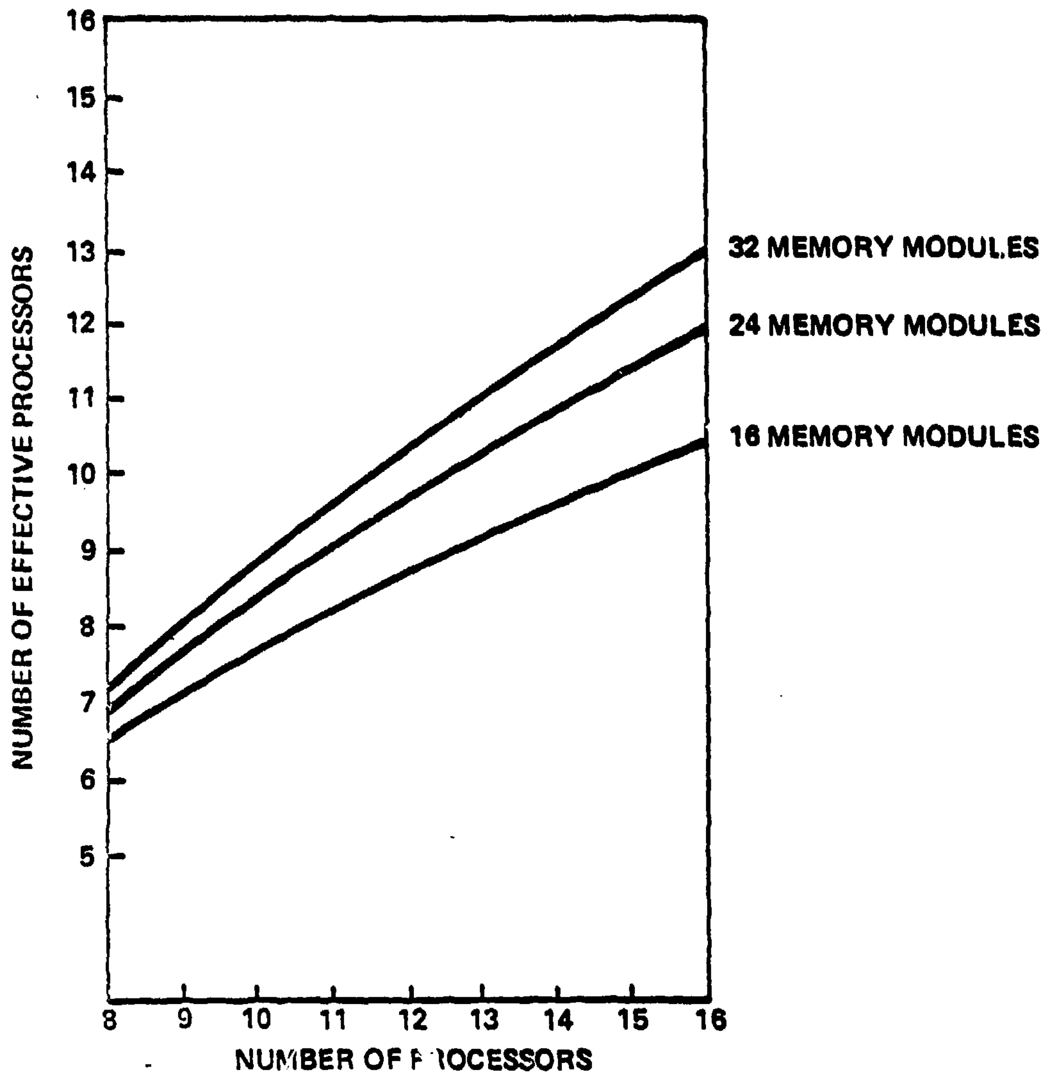

Figure 1 Memory Contention Issue
Processor Speed Equals Memory Speed

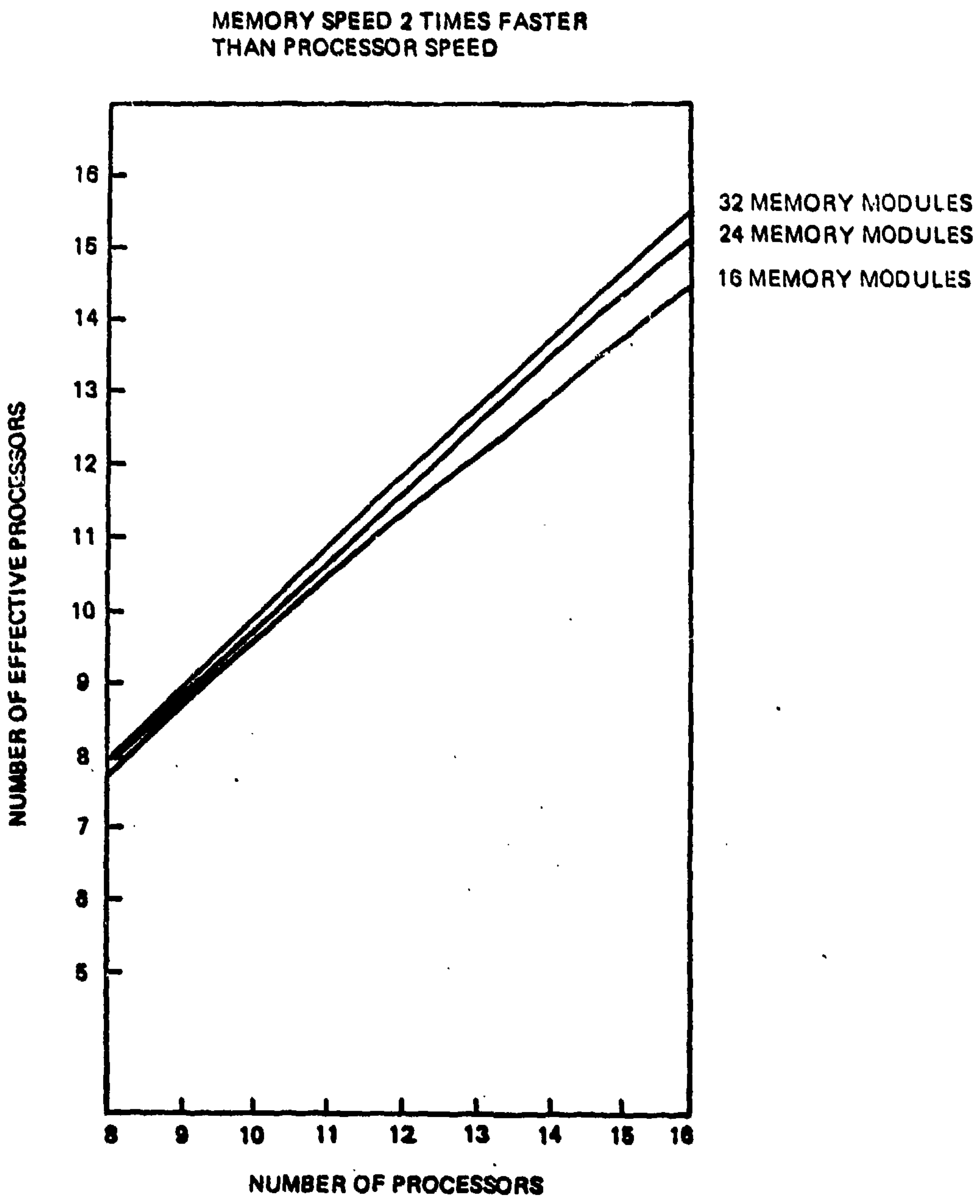
MEMORY SPEED 2 TIMES FASTER
THAN PROCESSOR SPEED
NUMBER OF EFFECTIVE PROCESSORS
NUMBER OF PROCESSORS
32 MEMORY MODULES
24 MEMORY MODULES
16 MEMORY MODULES
16
15
14
13
12
11
10
9
8
7
6
5
8
9
10
11
12
13
14
15
16

Figure 2

effective throughput of processors for different numbers of memory modules where processor speeds are equal to memory speeds. For example, if 16 processors are contending to gain access to 16 memory modules (processor speeds equal memory speeds) and each processor attempts to access each memory module with equal probability (uniform distribution), then the effective throughput is that of about only 10 processors. The effective throughput increases as more memory modules are added.

Figure 2 shows the effective throughput of processors for varying numbers of memory modules when the memory speeds are twice that of processors. It is important to note that these curves represent throughputs of processors when the liklihood of any processor accessing any memory module on a given request is assumed to be equal. Boeing investigated ways to ensure that near uniform distributions of memory module accesses would be maintained for a CIG system; this research will be explained later.

In addition to identifying throughput relationships and their dependence on the relative speeds and numbers of processors vs. memory modules, Boeing considered alternative approaches for connecting the processors (tilers) to the depth/frame buffer memory modules. Alternatives explored include using very fast common busses like that in Figure 3; crossbar switch architecture like that represented in Figure 4; and tree architectures like those in Figure 7.

Common Bus Architecture

Common busses are attractive interconnective networks because of their relatively simple implementations. Common bus systems permit one transmitter at a time to send data, which can be read by any receiver on the bus. Commercially available busses are not fast enough to support the higher end product of the CIG family, which require a bus capable of transmitting tiles at a 500 mhz rate. For CIG applications, the bus

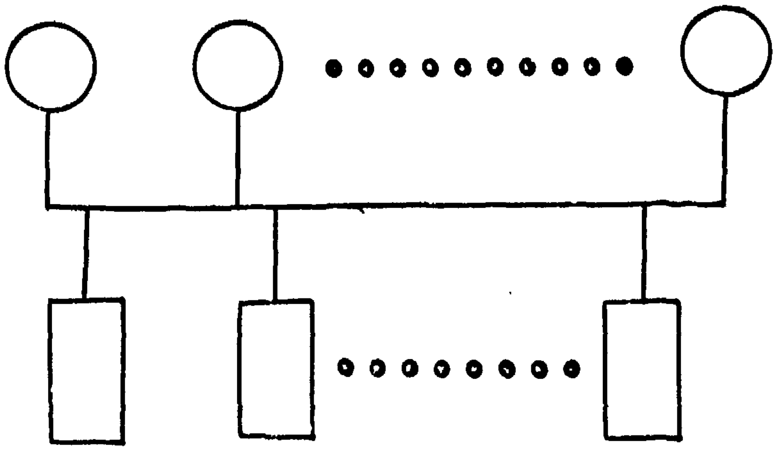

Figure 3 Time Shared BUS Architecture

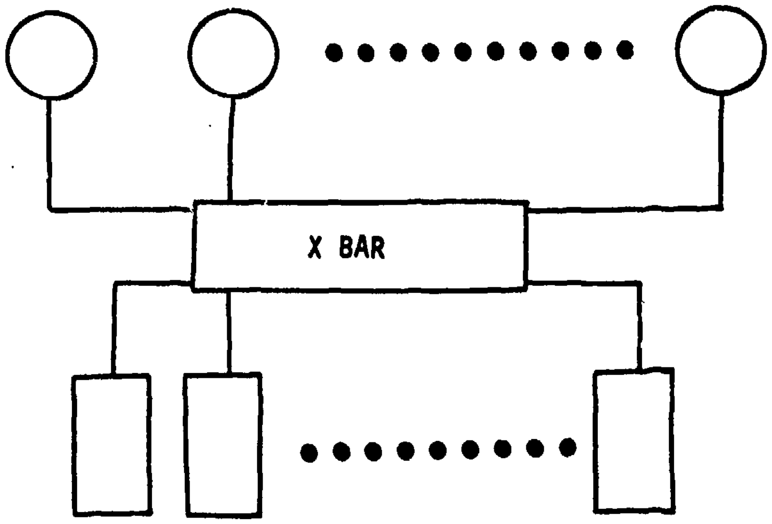

Figure 4 Crossbar Switch Architecture

receivers must be capable of receiving a string of tiles at this rate while simultaneously writing the data to memory. Both receivers and busses which can operate at these speeds are still in the experimental stage, and even if they were available they would not be scalable to arbitrary system requirements (acceptance criterion 3). Therefore, even though the common bus architecture may be quite acceptable for small scale CIG systems, it was not selected as appropriate for the CIG architecture research system.

Cross Bar Switch Architecture

Cross bar switch architectures as depicted in figure 4 consist of a switch matrix which permits any transmitter access to any receiver in the system (see Enslow (9)). Transmitters (processors) may send data to receivers (e.g. memory modules) simultaneously through the crossbar assuming that no contention for the receivers occurs. The physical size and arbitration control complexity of the crossbar switch increases quadratically as the number of processors and memory modules increases. A practical upper limit on the size and complexity of the crossbar switch exists (e.g. 32 processors X 64 memories is extremely complex), and therefore, is not a scalable architecture. Hence this architecture was concluded as inappropriate for the architecture research system. It however, may be feasible for use in a small scale CIG system.

Other Graphics Oriented Architectures

Parke (10) discusses the general broadcast approach, the splitter tree approach, and a hybrid approach to distributing the processing load for computer graphics Z-buffer systems. The architectures discussed in his paper all have the property that processors (tilers) can gain access to only part of the screen and each memory module is accessible by only 1 tiler. This eliminates the contention problem for accessing depth/frame buffer memory modules, but it introduces other disadvantages. Using the broadcast scheme, only one data entity (e.g. polygon) can be processed in a given time period, since all processors work on the same polygon simultaneously. The splitter tree approach takes large polygons in screen space and by geometric splitting and clipping at each tree level creates smaller polygons which eventually go directly to memory modules. This method artificially increases the number of polygons processed in the splitter tree.

Clark (11) and Fuchs (12) have developed different implementations of a broadcast approach to polygon processing. However, both the Distributed Processing Image Memory of Clark and the "Pixel-Planes" graphic engine of Fuchs are too specialized, since they scan only polygons. Because this is performed at the frame buffer level, the flexibility of processing a variety of data types is lost; thus they cannot satisfy criterion 2 (arbitrary data types or tiler types.)

Boeing Distribution Tree

Based upon our analysis Boeing has adopted an architecture approach for connecting tilers to frame/depth buffer memory modules called a tile distribution tree which meets all four criteria - throughput, scalability, arbitrary data types and VLSI implementable. The basic cell structure, a simple connection of 2 processors and 2 memory modules (queues), can be repeated as necessary. Using this scheme, each tiler outputs tiles consisting of a screen address (I,J), color (R,G,B) and a depth (D). The screen address is transformed to a memory address (M) using a tile distribution controller (scrambler). The tile is then routed through the distribution tree with binary routing decisions occuring at each level of the tree (examining a single address bit) until it reaches the appropriate depth/frame buffer memory module. The scalability of the distribution tree is depicted in figures 5, 6, and 7. Figure 6 and 7 represent a 4 tiler/8 memory module system and a 16 tiler/32 memory module system respectively.

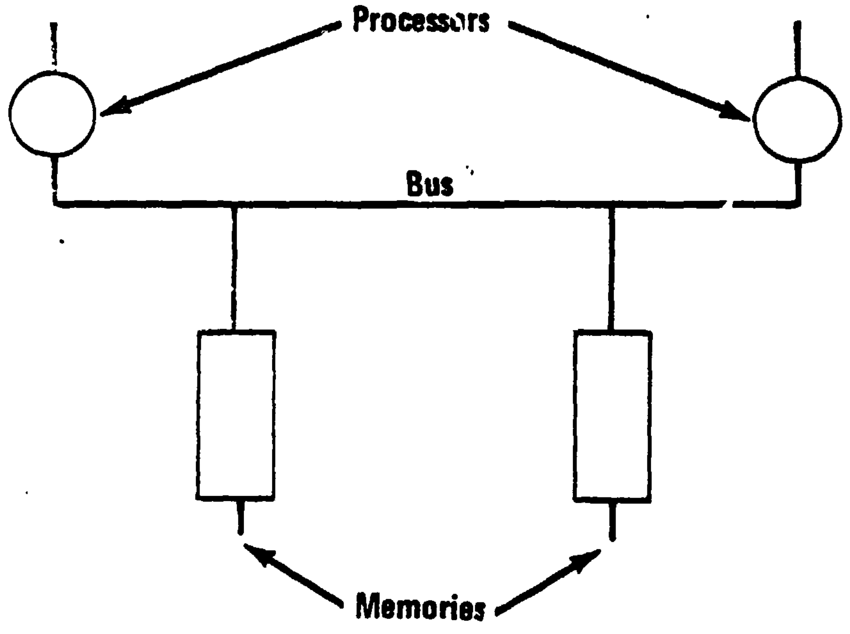

Figure 5 Tile Distribution Tree Building Block

Building Block is Repeated Throughout Distribution Tree

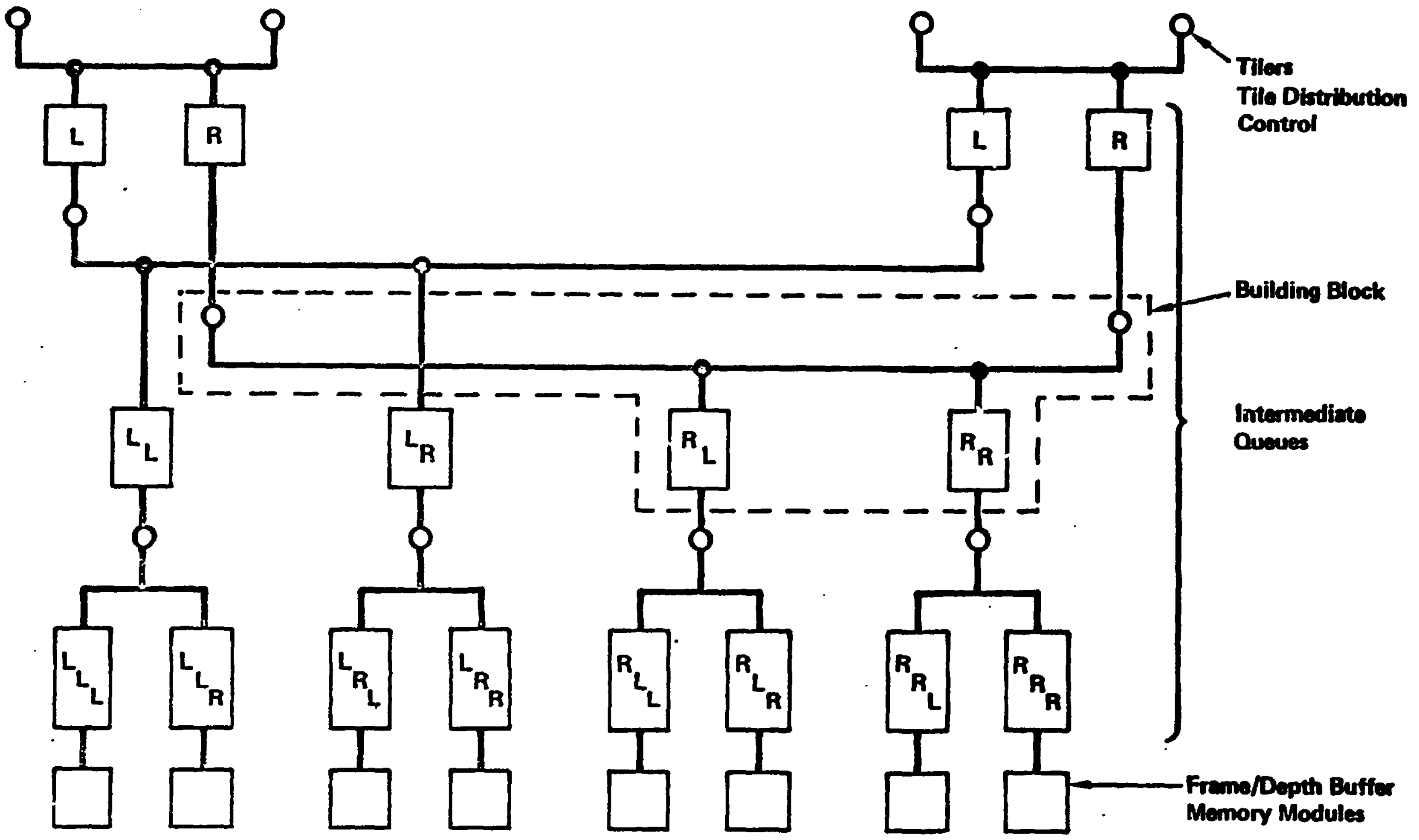

Figure 6 Tile Distribution Tree (512 X 512)
4 Tilers - 8 Memory Modules

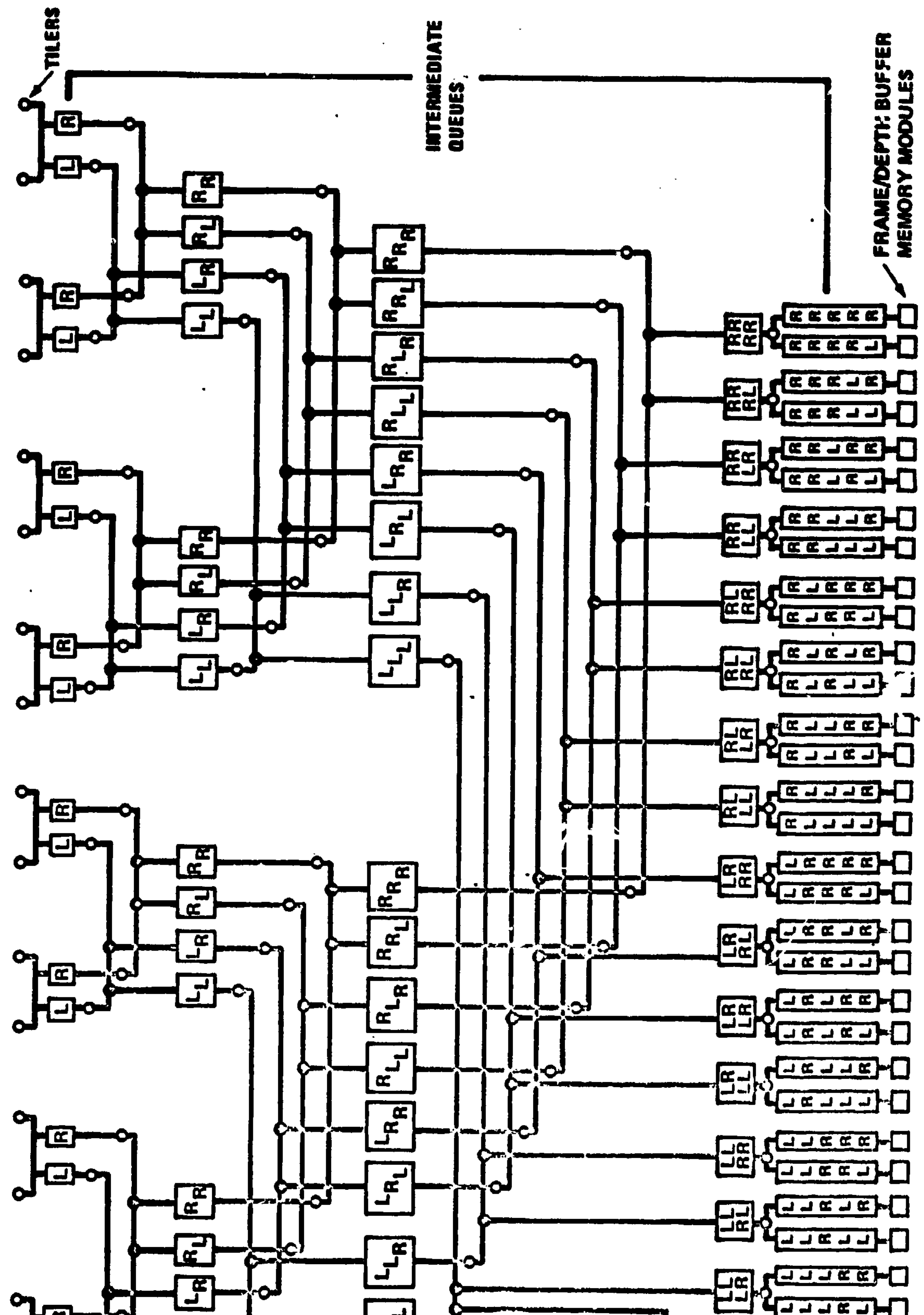

Figure 7 Tile Distribution Tree (1K X 1K) 16 Tilers - 32 Memory Moudles

As mentioned before, Vaughan and Anastas and others have assumed that processors access resources in a random fashion. In order to insure a near uniform distribution of memory module accessing, it is important to perform some form of pixel scrambling. Not only is it desirable to have a uniform distribution of total accesses to each memory module, but it is important to maintain the uniform distribution throughout the generation of each scene (at any point in time). If this is done, then queue sizes can be manageable.

Boeing has simulated the tiling operation on actual frame data to observe pixel to memory dispersions. These experiments performed with different scrambling algorithms have provided insight into a near optimal distribution scheme on depth/frame buffer memory accessing. More detailed simulations will be performed in Phase II to determine queue length requirements at each level of the distribution tree. Figure 8 represents one candidate pattern for a system with 8 memory modules where each number indicates the memory module to which that screen pixel belongs.

The use of this distribution tree architecture allows multiple tilers to operate on different data types in parallel. It also permits each tiler access to the entire screen, so no intra-screen geometric clipping is necessary as is the case with the splitter tree approach discussed by Parke (10). Moreover, a relatively high effective throughput can be attained if appropriate scrambling is performed and an appropriate relationship is maintained with the relative speeds and numbers of the tilers and depth/frame buffer memory modules. Consequently, this interconnection architecture satisfies the 4 criteria: flexibility of arbitrary tiler types, scalability to different requirements, high effective throughput of tilers, and VLSI implementable.

1	4	7	2	5	8	3	6	1	
2	5	8	3	6	1	4	7	2	
3	6	1	4	7	2	5	8	3	
4	7	2	5	8	3	6	1	4	
5	8	3	6	1	4	7	2	5	• • •
6	1	4	7	2	5	8	3	6	
7	2	5	8	3	6	1	4	7	
8	3	6	1	4	7	2	5	8	
1	4	7	2	5	8	3	6	1	
				•					
				•					
				•					

Figure 8 Candidate Pixel/Memory Module Distribution -8 Memory Modules

Depth Precision Research

Since Phase I of the Syntactic Visual Image Generation Program involved determining the feasibility of using the depth buffer approach for real time CIG systems, Boeing investigated the technical problems associated with calculating the depth of surfaces and comparing the depths of surfaces to eliminate occluded surfaces from the scene. Most of the computer graphics literature which deals with the depth buffer approach (alias z buffer) discusses the use of the z component (using x, y, z screen coordinate system (7) of the distance from the viewpoint location to a point (vertex) as the depth value of the point. However, substantial error is introduced if one elects to use a linear interpolation scheme to compute the depths of tiles in the interior of a polygon (e.g. triangle) given the depth (z values) of the vertices of the polygon.

Since linear interpolation of z produces too much error and introduces hidden surface elimination anomalies, Boeing has elected to abandon the use of z as depth. Instead, Boeing has adopted the use of 1/z as depth, since linear interpolation is exact for 1/z. In this case, the closer of two surfaces will have the larger depth value. However, complications arise with the use of 1/z as depth, since either large fixed point words are required to represent a reasonable range of depth for a CIG system or a floating point format must be adopted. For example, 24 bits of z (in meters) would permit a range of 512k meters at a precision of 1/32 meter, or in other words, 16 million units where 1/32 meter is a unit. To gain the same effective precision and range using 1/z as the depth requires about twice as many bits (48) using a fixed point data format. For this reason, Boeing has elected to use a 24 bit floating point format for representing the depth. Additional software simulation and testing of tiler algorithms and depth comparison algorithms using this floating point format will be performed early in Phase II to determine a final design specification for the tiler and memory components of the architecture research system.

Anti-Aliasing

Boeing has investigated implementation alternatives of an anti-aliasing algorithm which involves down sampling and filtering the computer (oversampled) scene (15). Emphasis was placed on identifying methods where successful unscrambling of the red, green and blue frame buffers of the high end real time CIG system (2048 x 2048 pixels) along with filtering could be performed. A conceptual design has been identified which will perform the anti-aliasing process for a full scale, full color, real time CIG system. More detailed hardware design will occur in Phase II.

Full Color System

Boeing investigated the impacts of developing a full color architecture research system (24 bit frame buffer) instead of the proposed 8 bit frame buffer system which would use a color lookup table and allow up to 256 colors to be displayed. Boeing concluded that the full color system would be developed, since the eventual high end CIG system would require full color. Database structures and fading, lighting effects, and shading algorithms developed for the architecture research system would then apply to the eventual high end CIG system.

The most cost effective approach to developing a full color architecture research system would be to purchase a 24 bit frame buffer (512 x 512) raster graphics system and to develop the custom tilers and frame buffer to handle one primary color (red, green, blue) at a time (serially). Three separate scenes (8 bits wide) would be generated and shipped to the 24 bit frame buffer where the primaries would be combined to generate the full color scene. Development of tilers and frame buffer memories to handle the three primaries in parallel (24 bits) would be more expensive and would not add much to the technical knowledge gained by developing the single-primary research system. For the same reason (cost effective research), Boeing has decided to develop a research system consisting of a frame/depth buffer memory of 512 x 512 size divided into 8 memory modules. Four tilers will be developed and the tiler speeds and memory speeds will provide tiling for the 512 x 512 screen at real time rates. Since anti-aliased scenes displayed at 512 x 512 pixel resolution require that a 1024 x 1024 pixel scene be generated, 4 passes through the scene generation process for each primary color will be performed. Each pass of 4 will represent one quadrant of the displayed scene; scenes being anti-aliased in full color will require 12 passes.

Fractal Object Processor

Alternative conceptual designs for a fast fractal object processor were assessed. Methods for implementing the recursive subdivision process efficiently and for tiling the small polygons thereby produced were evaluated to determine approaches to be further researched in Phase II. Partially overlapping polygons are produced during the generation of a fractal tree, which artificially increases the depth complexity. Since the depth buffer comparisons of tiles (memory accessing) is the CIG system bottleneck, it is very important that these redundant tiles are eliminated before they are output to the tile distribution system and depth buffer. Therefore, design efforts focused on reducing the number of redundant tiles being output by the fractal object processor and a small face tiler. Boeing concluded that development of a fractal object processor is feasible for a real time CIG system and more research will be performed in this area during Phase II.

VLSI Technology

Boeing assessed the VLSI design technology to determine if this technology was mature enough to support part or all of the Phase II architecture research system development. The VLSI design facilities and capabilities at Boeing were assessed and found to be unsatisfactory for use at present in developing any major components of the architecture research system.

Adequate simulation and testing tools are practically nonexistent at this time, not only at Boeing but throughout the country. Consequently, Boeing has decided to implement only a small part of the system (e.g. backface elimination) using the VLSI tools during phase II. Boeing will monitor the progress made in VLSI design tool development and will identify components or subsystems of the CIG system which make feasible candidates for VLSI embedding during a later phase (III).

Phase II Plan

Phase II involves development of a near real time, depth buffer oriented CIG system. The objective of the Phase II program is to expand the technology base to the point where design and development of a real time CIG system capable of supporting a variety of object types and producing far more detailed scenes than current real time systems is feasible.

Research will be concentrated in three areas: (1) multiple tiler/memory module architecture research; (2) alternative object processor algorithm development; and (3) database retrieval/level of detail and regional memory management research. Special graphics oriented processors to perform the common graphics functions such as perspective projection or clipping are being developed elsewhere. Boeing will not develop these kinds of processors, but may purchase such processors and integrate them with the architecture research system. Jim Clark's Geometry Engine is an example of a processor which will be considered.

Figure 9 outlines the 2 year Phase II program, which consists of development and demonstration of a basic system in the first year and development and demonstration of the complete architecture research system in the second year. Figure 10 is the schedule of general tasks to be performed where Tilers, Depth & Frame Buffer Memories, Back Face Eliminater, Distribution Tree, and Anti-Aliasing Processor represent custom hardware development, and the other tasks are software development tasks.

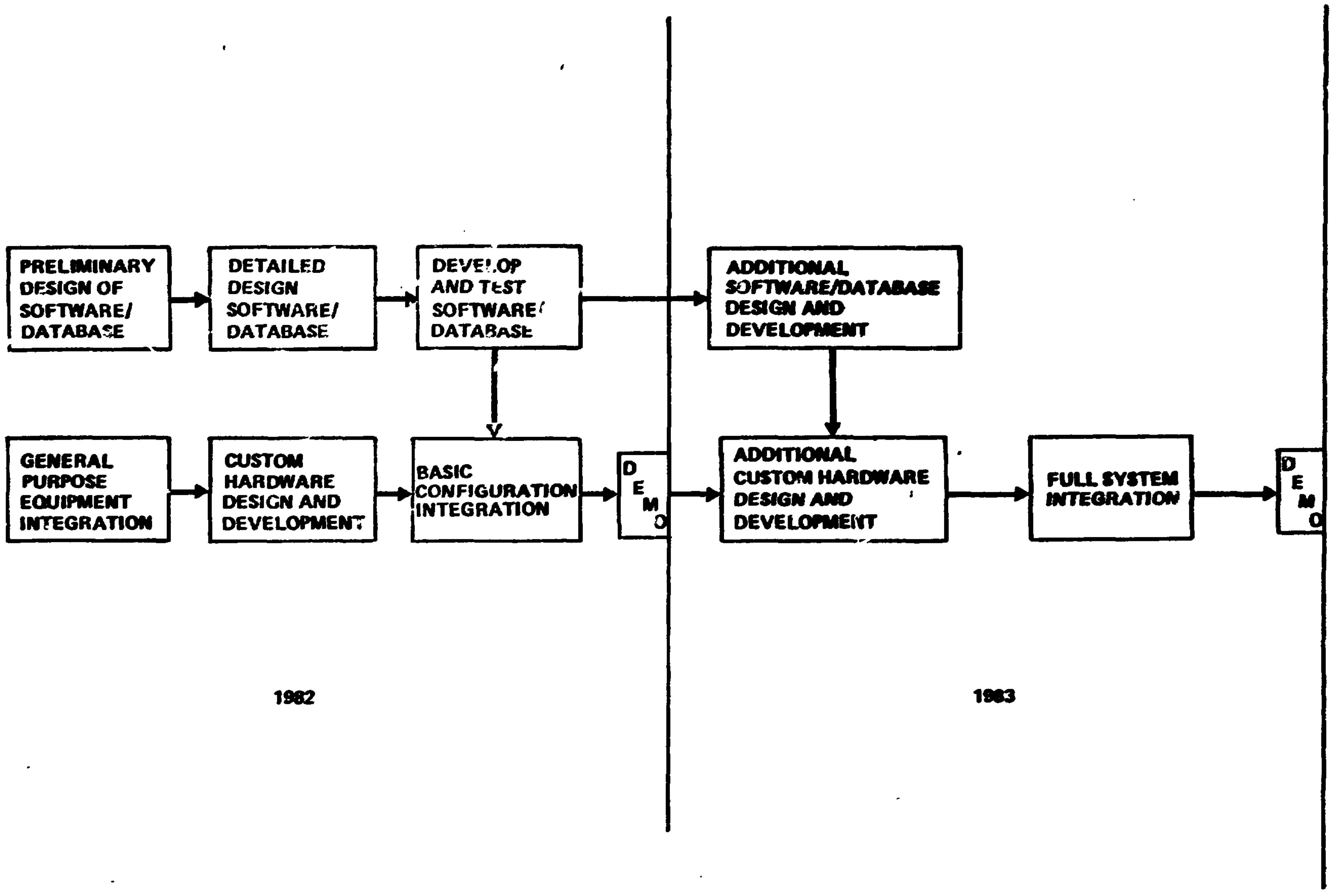
PRELIMINARY DESIGN OF SOFTWARE/ DATABASE
DETAILED DESIGN SOFTWARE/ DATABASE
DEVELOP AND TEST SOFTWARE/ DATABASE
ADDITIONAL SOFTWARE/DATABASE DESIGN AND DEVELOPMENT
GENERAL PURPOSE EQUIPMENT INTEGRATION
CUSTOM HARDWARE DESIGN AND DEVELOPMENT
BASIC CONFIGURATION INTEGRATION
DEMO
ADDITIONAL CUSTOM HARDWARE DESIGN AND DEVELOPMENT
FULL SYSTEM INTEGRATION
DEMO
1982
1983

Figure 9 Phase II Technical Plan Overview

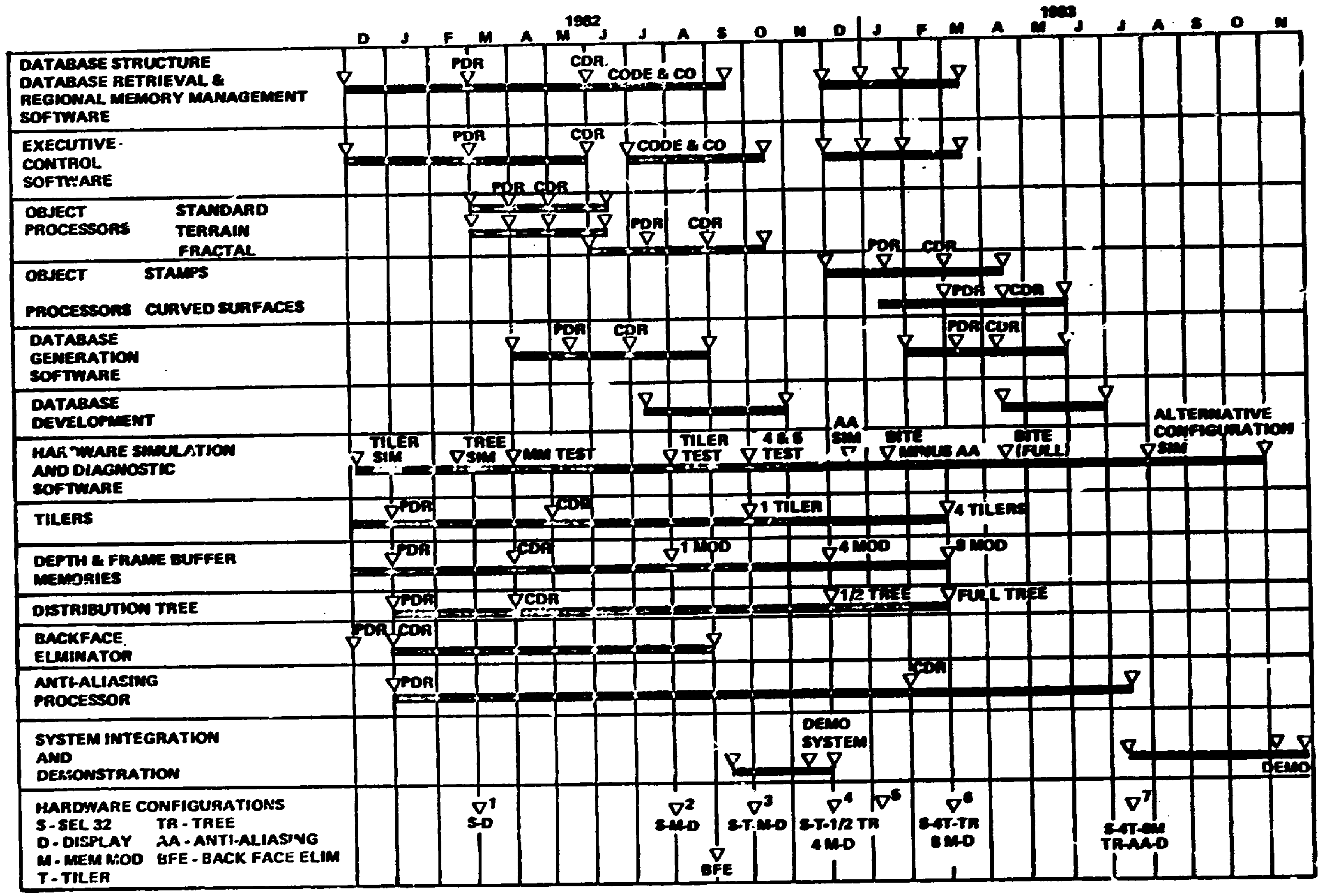

Figure 10 Phase II Schedule

Boeing intends to integrate the architecture research system in a systematic fashion. Configuration building blocks are numbered on the schedule; for example, configuration 3 will consist of the general purpose computer (SEL 32/87), one tiler, one memory module and the raster graphics display unit. The image generation software and the configuration 3 hardware will be demonstrated at the end of the first year. Gradually more tilers, more memory modules, the tile distribution tree and the anti-aliasing processor will be integrated into the system.

The architecture research system to be developed in Phase II is depicted in Figure 11. The SEL 32/87 computer system will be interfaced to the raster graphics unit and will be capable of generating scenes without the help of the custom hardware. The hardware shown in figure 11 represents equipment to be purchased or developed for the Phase II project and does not include the Boeing owned video disk which will also be interfaced to the system for real-time playback and evaluation of scene sequences generated and recorded in non-real time.

The architecture research system will be a very useful tool for expanding the technical knowledge necessary for developing a family of real time depth buffer oriented CIG systems for military applications. Boeing believes that the 2-year Phase II program will demonstrate the merits of the depth buffer approach for many applications and will motivate the development of real time depth buffer oriented CIG systems.

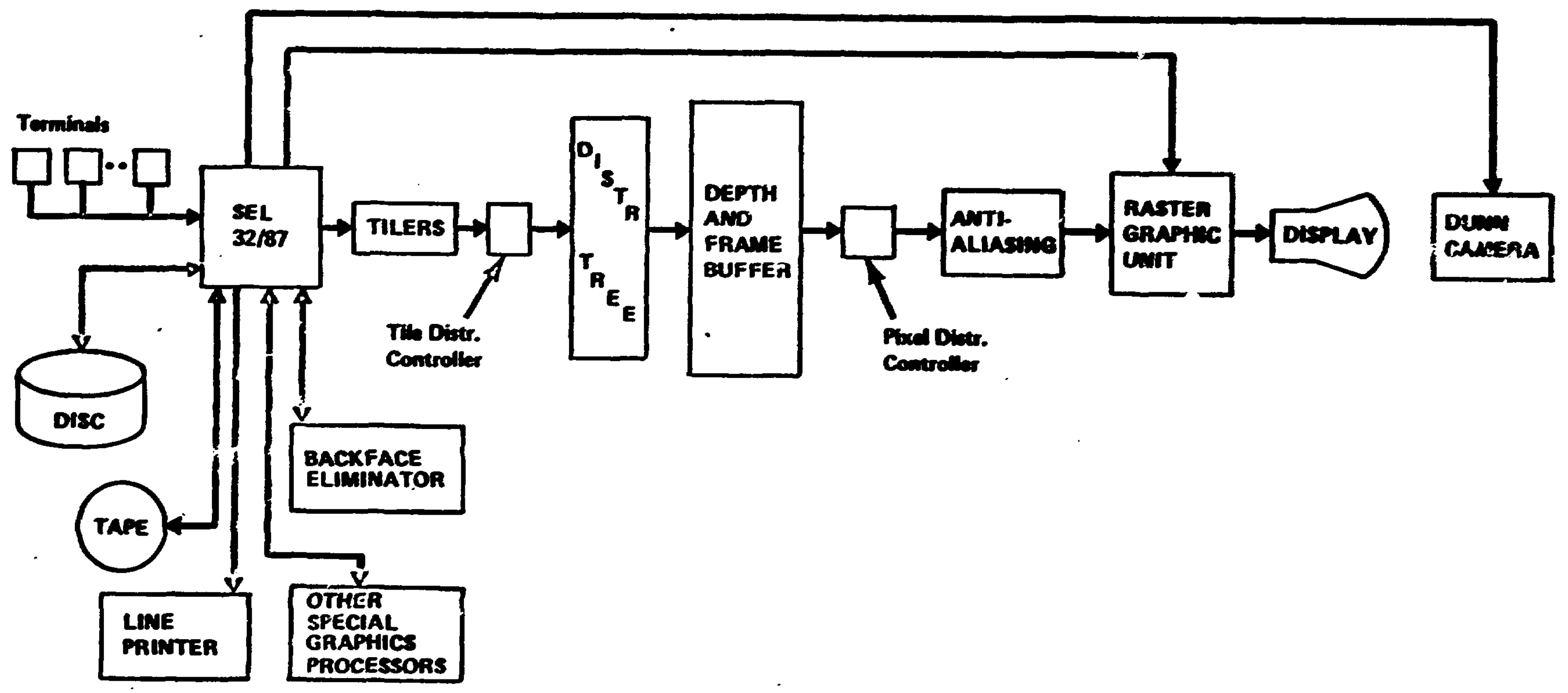

Figure 11 Architecture Research System

GLOSSARY OF TERMS

Depth Complexity:	The number of times, on the average, that a computed screen subpixel is accessed for a depth compare.
Pixel:	Smallest resolution element of a display screen.
Subpixel:	Smallest resolution element for a computed screen where the computed screen contains an oversampled version of the scene to be eventually displayed.
Anti-aliasing:	The process of down-sampling an oversampled scene and filtering the scene to minimize the undesireable sampling effects inherent in raster graphics systems (e.g. jagged edges).
Tile:	A subpixel sized segment of an object entity (e.g. triangle) after that entity has been perspectively projected onto a portion of the oversampled screen. The tile data consists of a screen address (I,J), a color (Red, Green, Blue), and a depth value (D).
Tiler:	A processor which computes and outputs tiles for any arbitrary data type (e.g. triangle, curved surface, texture).

References

(1) Schacter. B., Computer Image Generation For Flight Simulation. IEEE Computer Graphics and Applications, Oct. 1981, pp.29-64.

(2) Schumacker, Robert A., A New Visual System Architecture. Proceedings : Interservice/Industry Training Equipment Conference (Nov. 1980), 94-101.

(3) Dichter. W., Doris, K., Conkling, C., A New Approach To CGI Systems. Proceedings: Interservice/Industry Training Equipment Conference (Nov. 1980), 102-109.

(4) Gunwaldsen, Roger, Charge Image Generator: Theory of Operation and Author Language Support (Draft Tecchnical Report). Human Resources Research Organization, Feb 1975.

(5) Ranjbaran. D.E., Swallow, R. J., "Computrol in Flight Simulation", Pro. Eurographics '80 Geneva, Sept.. 1980, North-Holland Pub. Co. Amsterdam,
pp. 321-329.

(6) Rife, R., Allen, C., Advanced Tactical Visual System (ATVS) Development: Phase I - Technical Assessment. Boeing Tech. Report, Dec. 1979.

(7) Newman, W., Sproull, R., Principles of Interactive Computer Graphics, 2 Edition, NY. McGraw-Hill, 1979.

(8) Vaughan, R., Anastas, M., An Analysis of Multiprocessor Throughput Performance in the Limit. Journal of Digital Systems, Vol IV, Issue 2, pp. 153-175 (1979).

(9) Enslow, P., Multiprocessor Organization - A Survey. Computing Surveys, Vol. 9, Nov. - March 1977.

References (cont.)

(10) Parke, F., Simulation and Expected Performance Analysis of Multiple Processor Z-Buffer Systems. Proceedings: Siggraph "80, pp. 48-56, July 1980.

(11) Clark, J., Hannah, M., Distributed Processing in a High-Performance Smart Image Memory. LAMBDA, fourth quarter, 1980, pp. 40-45.

(12) Fuchs, H., Poulton, J., Pixel-Planes: A VLSI - Oriented Design for a Raster Graphics Engine. VLSI Design, third quarter, 1981, pp. 20-28.

(13) Strecker, W. D., Analysis of the Instruction Execution Rate in Certain Computer Structures. PhD thesis: Carnegie Mellon Univ. Pittsburg P.A. 1970.

(14) Baskett, F. and Smith, A.J., Interference in Multiprocessor Computer Systems With Interleaved Memory. Communications of ACM. Vol. 19 No. 6, June 1976 pp 327-334.

(15) Allen, C., Anti-Aliasing Considerations in Computer-Generated Imagery. Proceedings of the 1980 Summer Computer Simulation Conference pp 282-285