A Multiple Application Graphics Integrated Circuit

MAGIC II

H.R. Finch, M. Agate, A.A. Garel, P.F. Lister, R.L. Grimsdale.

School of Engineering and Applied Sciences
University of Sussex
Falmer
Brighton
East Sussex BN1 9QT
United Kingdom

This paper describes the design considerations for a polygon graphics geometry processor subsystem. The architecture for a Multiple Application Graphics Integrated Circuit (MAGIC II) is outlined, and low, medium and high performance system configurations using MAGIC II are discussed.

1. Introduction

Computer graphics research has been prominent in the research activities at the University of Sussex for some years, particularly in the area of specialised, dedicated hardware. The current major project, PRISM, is directing significant effort towards the design of custom chips for graphics tasks, using VLSI technology. The potential offered by VLSI for large, powerful processing and memory devices make it ideally suited for the computer graphics application - existing high performance graphics systems (such as flight simulator visual systems) rely heavily on large volumes of hardware in order to attain the processing speed necessary.

PRISM (Processors for Real time Image Synthesis and Manipulation) is funded under the Alvey programme, and is run collaboratively with GEC Research Ltd. at Wembley, and Singer Link Miles at Lancing.

The aim of the project is to produce a modular, expandable chipset capable of performing all the operations necessary to transform three dimensional databases in to realistic visual images. The proposed applications for the chipset range from small personal computer based systems, through high performance CAD systems, to full state-of-the-art flight simulator visual systems. The requirement can hence be seen, at the high end, for real time operation, processing a large number of polygons to a high degree of visual realism. At the low end, however, system cost is an overriding factor and a reduction in hardware, giving reduced performance, must be possible.

The overall system has been notionally divided, in the traditional way, into 3 subsystems: the database processor, which selects portions of the database for processing, the geometry processor which transforms and clips 3D world model objects into displayable 2D objects, and the display processor whose function is to perform various sorting and visual enhancement algorithms on the data before displaying it. This paper is primarily concerned with the geometry processor subsystem, and MAGIC II (Multiple Application Graphics Integrated Circuit) the chip designed to implement it.

2. Geometry Algorithms

The operations performed in the geometry subsystem are well defined and based on existing algorithms [1], [2]. Some mathematical manipulation has been necessary, however, in order to optimise the performance in hardware.

The algorithms are as follows:

(1) Back facing surface removal - this is a simple dot product plus offset calculation to determine the orientation of each surface.

(2) Parallel projection - a single matrix-vector multiplication and addition. It can be reduced to individual dot product plus offset calculations. Screen scaling factors are incorporated in the view matrix to eliminate extra calculations later on.

(3) Clipping - this can either be 3D clipping (to a number of planes or a view cone) or 2D clipping (to lines representing the screen boundary). 3D clipping is the 'generic' algorithm, with 2D forming a special case within this. The Sutherland Hodgman method of clipping all objects to each plane/line in turn has been adopted, and although this can result in the generation of extraneous edges for concave polygons, it is not considered to be a serious problem. The clipping operation has been split into two parts, namely calculation of the parameter, lamda, for each edge (vertex pair), and the intersection calculation using lamda to interpolate between the two vertices.

(4) Perspective division - the x and y coordinates of each vertex are divided by their depth value (or multiplied by their proximity value) and added to the coordinates of the screen centre. Screen coordinates are thus generated.

(5) Calculation of gradients - gradient values for x, y, z or proximity, and intensity are needed by the display processor. These are calculated in the normal way with two subtractions and a division.

3. Geometry System Architectures

The architecture of the geometry system must be viewed at two distinct levels: the system level and component level. These are related.

Firstly, due to the computationally intensive nature of the geometry tasks, some degree of parallelism must be employed at a system level in order to achieve

adequate performance. In terms of hardware, or, more specifically, VLSI chips, this parallelism can be achieved in several ways.

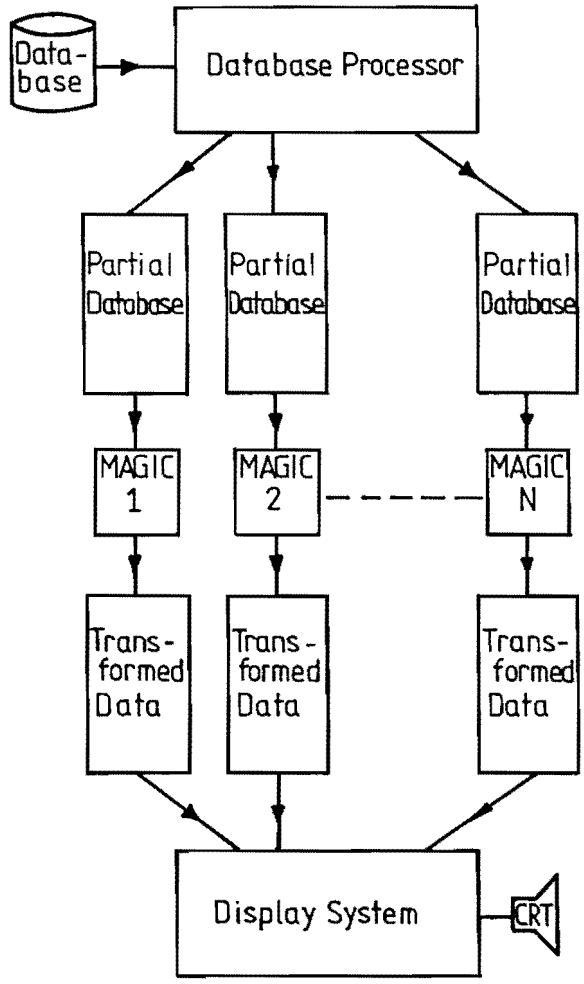

Figure 1: SIMD Style Geometry System.

The most obvious method of implementing a parallel processing scheme is to replicate functionality within the system: in the geometry system this means allocating database objects to a number of chips, each performing the entire geometric transformation, figure 1. This can be thought of as 'vertical parallelism', or a SIMD (single instruction multiple data) configuration [3], and is the technique used in an earlier design, MAGIC I [4]. Although effective for low and medium

performance systems, it is envisaged that the complex data and resource allocation schemes needed for a large number of chips, and its reliance on a microprogram and von Neumann style architecture, make the technique inappropriate for very high performance applications such as flight simulation.

A development of the SIMD idea is that of the array computer comprising a 2D array of interconnected processing elements, figure 2. While this offers still greater performance enhancements over non-parallel systems, the tasks of scheduling and synchronisation are difficult and not really necessary for the task in hand. An array configuration may be appropriate for graphics computation using only general purpose processing elements - transputers, for example, but performance is unlikely to match that of dedicated chips within the same economic bounds.

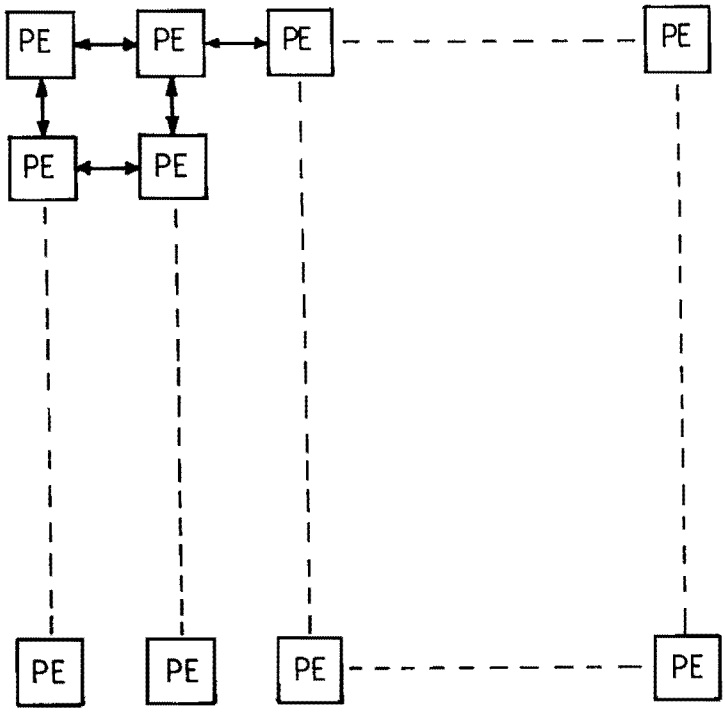

Figure 2: Array of Processors.

An alternative to vertical parallelism is 'horizontal parallelism', or pipelining. This type of system comprises a number chips connected serially; each chip performs only a part of the overall geometry calculation, but a pipeline of such chips together performs all the necessary operations. Data flows through the pipeline, 'stopping' at each stage to be processed; once the pipeline is full, all the geometry

operations are performed simultaneously but on different database elements, figure 3.

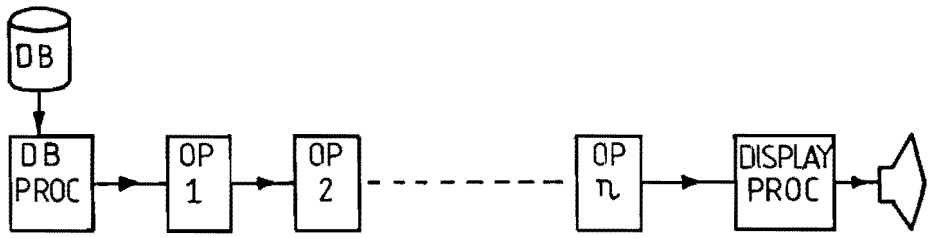

Figure 3: Pipelined Geometry System.

Although a single pipeline of chips may have a limited performance, it is possible to employ parallel pipelines (resulting in 2D parallelism) with the problem of data allocation techniques being significantly less severe than in the fully vertical case, figure 4. Very high performance systems are thus easily attainable.

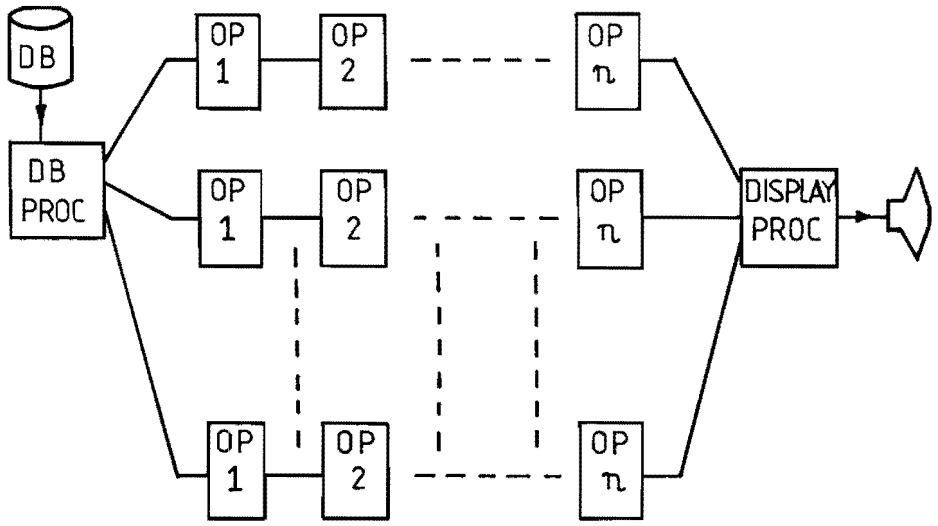

Figure 4: Parallel Pipelined System.

If each chip in the pipeline described has a fixed functionality, the smallest system possible is the single pipeline. This may give rise to a chip count that is

prohibitively high for small, PC based applications. It is therefore desirable for the chips to be reconfigurable to perform some or all of the operations that occur in the pipeline. With appropriate buffering arrangements, this means that the data can make several passes through a reduced length pipeline, resulting in lower performance. In the extreme case, the geometry system might consist of only one chip with a data pass for every algorithm to be performed, figure 5.

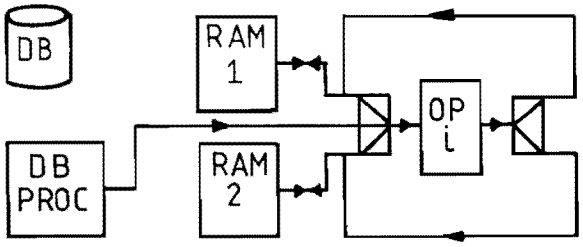

Figure 5: Reduced Length Pipeline Geometry System.

This style of system architecture, then, provides a modular, expandable approach with systems ranging from a single chip with controller, to highly parallel pipelined systems. For this reason, the architecture of MAGIC II has been designed with such a scheme in mind.

4. Architectural Considerations for MAGIC II

The fundamental principle of pipelined systems is that every stage in a system should have the same time delay. This ensures that all stages produce and consume data at the same rate, resulting in a continuous, even flow of data from one end to the other. If any stage in a pipeline introduces extra delay, this delay will propogate to the output and impair the performance of the entire pipeline.

The algorithms in the geometry system involve varying degrees of complexity and hence varying time delays. An instance of MAGIC II configured to perform any of the algorithms cannot, therefore, be regarded as the fundamental pipeline stage, and internal pipelining is necessary. Such pipelining is extremely well suited to the graphics geometry problem due to the repetitious nature of the task and the large quantities of data involved; broadly speaking, every vertex (or vertex pair) undergoes exactly the same processing. Pipelining is also a favoured technique when designing VLSI devices since local and regular connections can be made - the large cost in terms of time, power and silicon area of long, complex connections is eliminated [5]. Other factors to consider when designing systems using VLSI include the need for repetition, simplicity and regularity: it is very much easier to use a small number of simple cells which can be replicated many times, than to use

a large, complex design to solve the same problem. Pipelining meets this constraint, since only a few basic elements, such as arithmetic units and registers, are required.

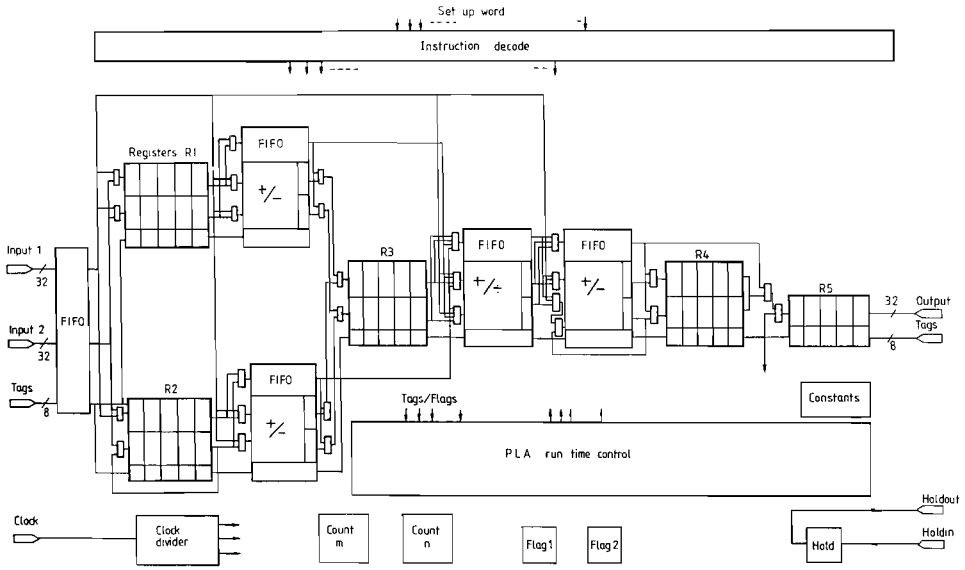

Figure 6: Architecture of MAGIC II.

In order to ensure optimal performance of MAGIC II for the five geometry algorithms, each algorithm has been considered in turn, and a pipeline of arithmetic and register units designed to implement it. The architectures for each algorithm were verified using the hardware description language and simulation tool, ELLA. ELLA provides a high level language which can be used to describe circuits at various levels of abstraction, from top-level functional representations down to gate level descriptions. The architectural elements of each of the algorithms have been combined and 'generalised' into an architecture capable of performing all the geometry tasks efficiently, figure 6. Although MAGIC II has been designed to support five specific algorithms directly in hardware, additions to the basic 'skeleton' architecture provide an element of flexibility. More general purpose functionality is available, catering for possible future CGI requirements, display processor functions and certain signal/image processing tasks. Modifications to the original algorithms may also be possible. It is noteworthy that the provision of additional functionality does not incur any run time performance degradation, assuming the extra algorithms can be implemented directly using the hardware resources available. This can be likened to the RISC (Reduced Instruction Set Computer) philosophy

[6] where most algorithms can be implemented with very high performance using only a small, basic instruction set. If the algorithm is not suitable, however, performance may be very low - such cases are assumed to be infrequent for RISC, and a similar assumption can be made for MAGIC II within the class of applications described.

5. Operation of MAGIC II

The architecture of MAGIC II is essentially a pipeline of arithmetic and register file elements, with each element connected via several multiplexers to the next. The arithmetic elements are an array multiplier/divider and several adder/subtractors, and the register files are specially designed units suitable for general vector manipulation, figure 6.

The chip has two input ports and one output port; if a second output port is required (this is rarely necessary), this can be provided by an additional instance of MAGIC II in parallel. A bus has been included which effectively bypasses all the functional units and register files, the data from which can be 'tapped in' to the pipeline at any point. This is typically used to carry the offset, from the second input port, in dot-product-plus-offset type calculations.

Control of MAGIC II falls into two categories: setup type control, which configures the pipeline to perform a given task, and run-time control which coordinates 'conditional', data dependent operations. Setup control signals are multiplexed onto the data port lines and these are then decoded into a very long instruction word (about 128 bits) to configure the register structures, arithmetic units and interconnections within the pipeline. The pipeline is then switched to run time control.

Run time control is effected by a PLA structure which takes flags (from the arithmetic units) and data tags as input, and produces control signals (usually for data routing) for all elements in the pipeline. Each piece of data in the pipeline has an encoded 'tag' associated with it; the tags 'follow' the data through the pipeline in parallel tag registers so that the internal controller 'knows' the context of the data. For example, the Last vertex in a polygon always has a particular tag so that the controller can generate a closing edge using saved first vertex data.

6. MAGIC II System Configurations

The MAGIC II processor provides direct support for three different levels of pipeline complexity, as typified in the following:

- A minimum geometry system, based on two instances of the MAGIC processor.
- A fully pipelined system with a single numeric data stream, offering performance sufficient for use in flight simulators.
- A pipelined system with three parallel streams of vector elements, giving a very high performance for use in future simulators.

The first of these systems utilises pipelining only at a numeric processing level (for example, a single chip may execute 32 simultaneous divide operations). The second pipeline operates at and the third runs at 'vector rate', processing entire vectors at each clock. Other systems of intermediate performance are also possible, using buffering around partial pipelines. A further option, as described in section 3, is to operate a number of parallel pipelines at element rate, offering greater potential for increased performance than the vector rate system.

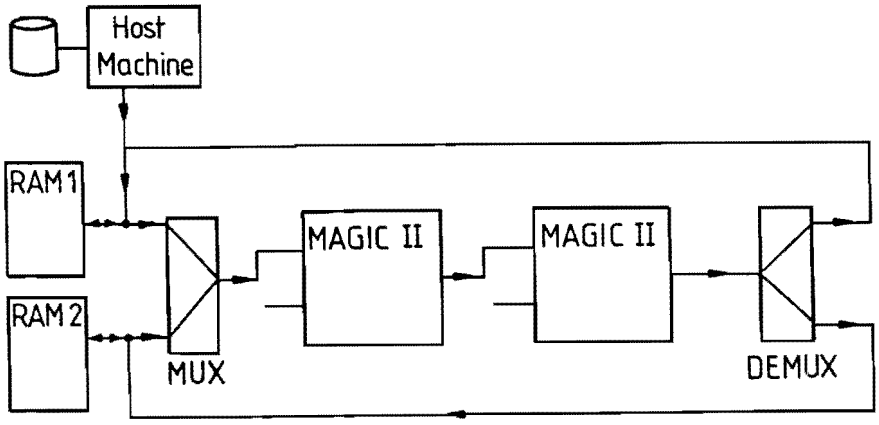

Figure 7: Minimum MAGIC II geometry system.

The minimum number of chips required to implement all of the geometry operations is two, since each clipping operation requires a parameter calculation and an intersection calculation working in parallel. The two processors must each be configured for a single geometric operation prior to the data being passed through, and buffering is required so that intermediate results are stored after each pass. This gives rise to the system shown in figure 7, with two blocks of RAM and appropriate multiplexing to allow alternate read and write access to each block.

Because of the need to reconfigure the MAGIC II processors before each pass, control of the minimum system is more complex than for the fully pipelined configurations. In addition to performing initialisations, the controller must maintain frame synchrony with the host, control the multiplexer and demultiplexer, and generate RAM addresses. A standard sequencer such as an AMD 2910 is suitable for this, though it is omitted from figure 7 for clarity.

Figure 8 shows a fully pipelined geometry system comprising element rate modules for parallel projection, perspective projection and clipping to a near plane and the edges of the screen. A single back facing surface remover interacts with the controller to determine which surfaces are subsequently passed through the pipeline. A total of sixteen instances of MAGIC II are required.

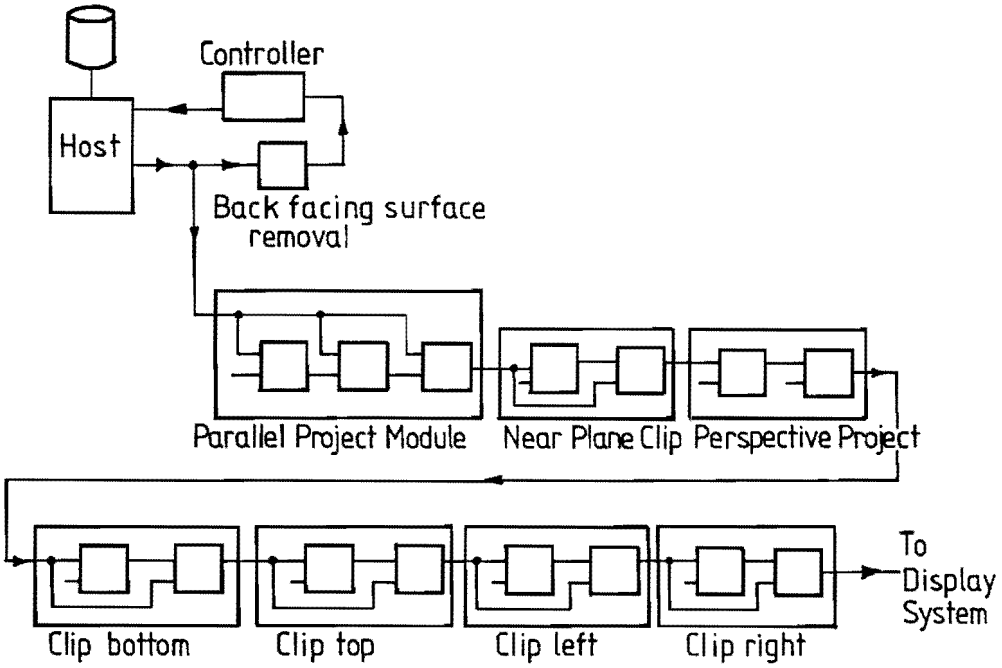

Figure 8: Element rate pipelined geometry system.

The vector rate pipeline is shown in figure 9, although separate instances of MAGIC are not indicated within each module. A bus width of 96 bits is required from the database (128 if homogeneous co-ordinates are used, or if intensity values are associated with each vertex). The system shown comprises 36 MAGIC II processors.

An estimate of the performance of the minimum system is most easily derived from consideration of the fully pipelined systems, so this is deferred to last. An average of 4 vertices per polygon are assumed, with a real time frame rate of 60Hz. The pipeline is assumed to be clocked at 100ns. Since it is difficult to quantify the number of surfaces which are back facing or off-screen in any one frame, the number of polygons processed is discussed in preference to the number displayed. Polygons which are off-screen will give rise to empty data slots at the clipping stages, while those detected as back facing will yield only a few empty data slots between potentially visible surfaces.

An element rate transformation pipeline is capable of reading a single cartesian co-ordinate vertex with associated intensity value every 4 clock periods. This corresponds to a rate of 2.5 million vertices per second, or approximately 10,000

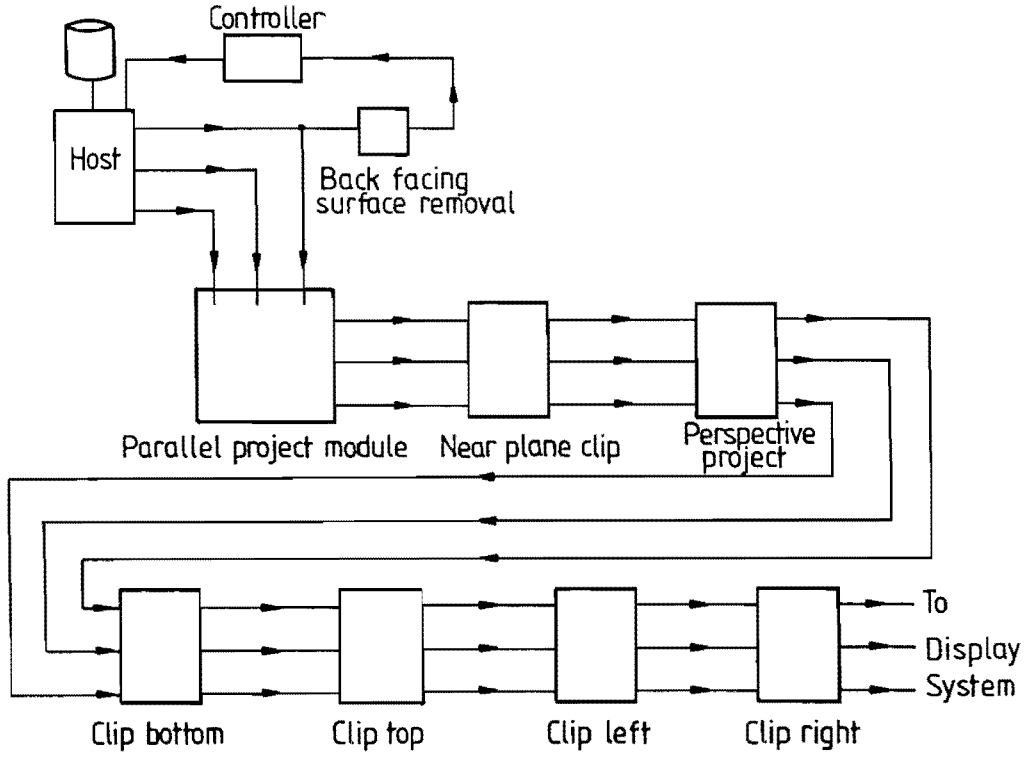

Figure 9: Vector rate pipelined geometry system.

polygons processed per frame. The vector rate equivalent runs at four times this speed, giving a processing rate of 40,000 polygons per frame.

The 2-chip equivalent of the above pipelined system has only a single module which is reconfigured for each operation. Referring to the element rate pipeline it can be seen that the clipping phase will take 5 passes, perspective projection and back facing surface removal will each take one pass, as will back facing surface removal, and the parallel projection phase will run 3 times slower. The whole projection therefore runs at about one tenth of the speed of the element rate pipeline, corresponding to approximately 1000 polygons per frame.

7. Conclusion

An architecture for MAGIC II has been presented, showing how it can be used for any of the geometry algorithms typically required in a polygon-based graphics system. A general purpose nature within the chip itself has been

cultivated, resulting in a very flexible device suitable for many vector/matrix type calculations.

Various system configurations have been discussed, each using multiple instances of MAGIC II; performance estimates for the different configurations have been included, demonstrating the suitability of MAGIC II for a wide range of applications, from personal computers to state-of-the-art flight simulator visual systems.

8. References

[1] Newman, W.M. and R.F. Sproull "Principles of Interactive Computer Graphics", McGraw Hill (1979)

[2] Foley, J.D. and A. Van Dam "Fundamentals of Interactive Computer Graphics", Addison Wesley (1982)

[3] Levin, D. "Theory and Design of Digital Computer Systems", Nelson (1980)

[4] Agate, M., H.R. Finch, A.A. Garel, P.F. Lister, R.L. Grimsdale "A Multiple Application Graphics Integrated Circuit - MAGIC", *Eurographics '86*, Elsevier Science Publishers B.V., The Netherlands (1986)

[5] Foster, M.J. and H.T. Kung "The Design of Special Purpose VLSI Chips", *IEEE Computer*, Jan. (1980)

[6] Katevenis, M.G.H. "Reduced Instruction Set Computer Architectures for VLSI", MIT Press (1985)