VLSI Drawing Processor Utilizing Multiple Parallel Scan-Line Processors

Damian Denault, Eric Ryherd, John Torborg,
Robert Tosi, Ross Werner

Raster Technologies, Inc.
Two Robbins Road
Westford, Mass. USA 01886

In a typical graphics system, a single drawing processor is used to perform pixel level drawing operations, one pixel at a time. A VLSI based drawing processor and image memory controller is presented which takes advantage of scan-line partitioning of many graphics operations. A four processor implementation is described which operates on four scan-lines in parallel to achieve near real-time shading performance for complex objects.

Drawing processor commands are provided for points, vectors, triangles, rectangles, block pixel moves, and image transfers. Vectors and triangles can be drawn with shading and depth buffering. The chips also incorporate integral vector and area pattern registers, and support translucency.

The drawing processor chips directly interface to the image memory RAMs without any external buffers, registers, caches, or control logic, allowing a high performance system to be configured simply and cost effectively. These chips are implemented in the GX4000 high performance workstation graphics system which is capable of rendering close to 200,000 shaded and depth-buffered 100 pixel polygons per second and over 34,000 shaded and depth-buffered 1000 pixel polygons per second.

CR Catagories and Subject Descriptors:

B.2.1 [Arithmetic and Logic Structures]: Design Styles - Parallel;
B.2.1 [Arithmetic and Logic Structures]: Design Styles - Pipeline;
C.1.2 [Processor Architectures]: Multiprocessors - Parallel processors;
I.3.1 [Computer Graphics]: Hardware Architecture - Raster display devices;
I.3.3 [Computer Graphics]: Picture/Image Generation - Display Algorithms.

Additional Key Words & Phrases:

interpolation, scan conversion, shading, image synthesis, graphics VLSI, frame buffer control.

1. Introduction

The requirement for high performance rendering hardware has long been realized by graphics researchers. Only recently have commercially available graphics systems provided the necessary performance for interactive 3D modelling of real world objects and models. Even these systems, however, have difficulty providing adequate performance for realistic rendering of complex objects.

Shading and depth buffering [Sutherland, 1974] of polygonal approximations has become a popular technique for realistic rendering of solid objects and surfaces because of the adaptability of hardware solutions. Linear interpolation across the surface of each polygon can be used to approximate the surface position and color [Gouraud shading, Gouraud 1971]. The color information for polygon vertices can be determined from parametric information such as stress or temperature, or can be calculated from surface orientation and light sources.

Several graphics systems have been developed which incorporate these techniques in special purpose hardware. All commercially available systems employ a single drawing processor architecture which calculates a single pixel at a time. The fastest of these systems can calculate a new pixel value every 50 nanoseconds resulting in a peak performance of 20 million pixels per second. [Swanson, 1986] Although this sounds very fast, actual system performance figures when rendering small triangles (less than 100 pixels) are typically less than 5,000 triangles per second for shading only and even less for shading and depth buffering. For complex models of 5000 to 20,000 triangles, this is barely adequate performance for interactivity.

Several architectures have been proposed which utilize multiprocessing to achieve higher rendering speeds [Fuchs 1977, Fuchs 1981, Fuchs 1986, Parke 1980, Niimi 1984], most of which have employed depth buffering. One such approach proposed by Fuchs [Fuchs 1977] and discussed by Parke [Parke 1980] provides a broadcast controller which distributes polygonal patches to multiple image processors.

Fuchs proposed that each image processor render only certain pixels as determined by a preset interlace pattern. For example, a two processor system may be configured with an interlace pattern which depends on row position so that one processor renders all pixels on even scan-lines and the other renders odd scan-line pixels. In Fuchs's model, each processor receives all polygons but only renders those pixels which are assigned to it based on the interlace pattern. This forces all processors to redundantly perform polygon scan conversion and differential calculations, thus adding overhead and reducing performance.

Except for smart memory approaches [Gupta 1981, Fuchs 1986], little attention has been paid to the drawing processor/image memory interface. A superfast rendering engine is worthless if the image memory update bandwidth cannot support the drawing rate. Many high performance renderer architectures have not been designed with a specific image memory organization in mind. Hence, they have had to resort to high speed and/or low density memory components or complex memory control structures (such as pixel caches) which try to exploit some of the coherence in many graphics algorithms. These approaches often result in systems which do not meet the original performance expectations, are not cost effective, or are complex and unreliable.

2. GX4000 Image Memory Unit Architecture

A GX4000 Image Memory Unit contains a bus interface, Drawing Processor, image memory RAM, and video output section. The GX4000 Drawing Processor uses a multiple processor architecture with a unique combination of pipelining and parallelism that exploits the scan-line partitioning of many graphics algorithms while avoiding the complexity and inefficiencies of the approaches described earlier. The Drawing Processor is composed of five VLSI processors (One Master Controller IC and four Scanline Processor ICs). These chips perform the necessary address and data calculations to decompose the commands into individual pixels and write them into the image memory. A block diagram of the Image Memory Unit and Drawing Processor is shown in Figure 1.

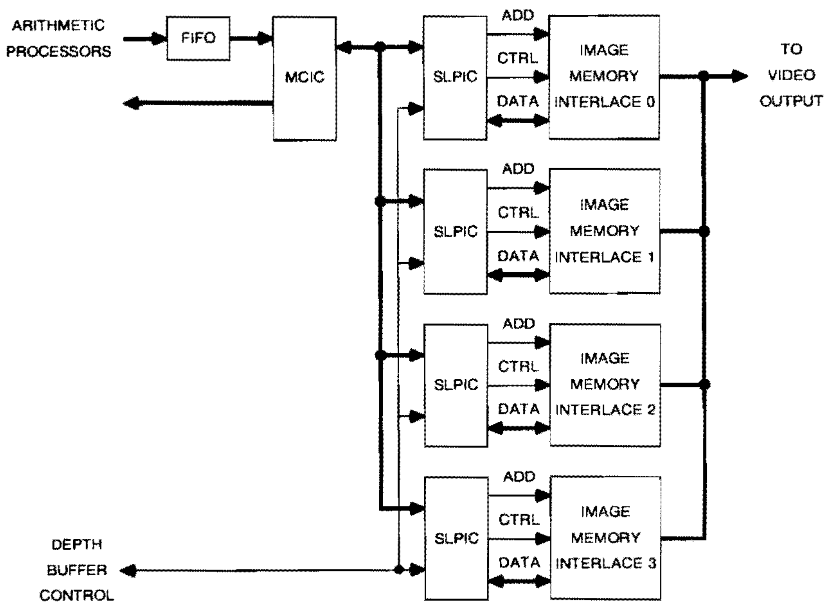

Figure 1: Image Memory Unit Architecture with Drawing Processor

The Scan-line Processor (SLPIC) and Master Controller (MCIC) are each implemented as single VLSI devices. The number of SLPICs chosen was determined by the memory configuration used and the system cost goals. Other configurations can be used for higher or lower cost configurations or for other memory organizations. The SLPICs interface directly with the video RAMs used for the frame buffer with no external logic. This allows a complete Drawing Processor to be configured with surprisingly few components.

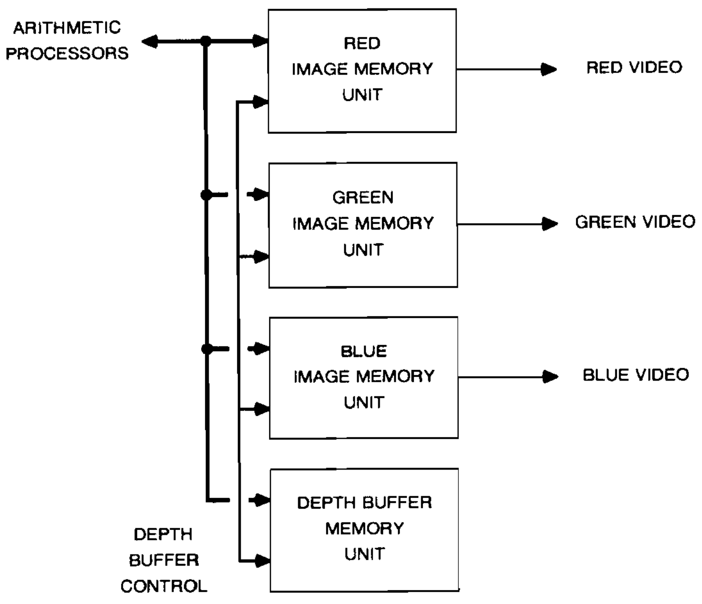

Figure 2: Full color system with depth buffer.

Each Image Memory Unit supports a double buffered 1280 x 1024 x 8 bit frame buffer. A similar module, the Z-buffer Memory Unit, contains a 1280 x 1024 x 16 bit depth buffer. Up to three Image Memory Units and one Z-buffer Memory Unit may be combined in the GX4000 Graphics System to provide red, green, and blue image banks with depth (Z) buffering. This configuration is shown in Figure 2. In a fully configured system, 16 SLPICs are operating concurrently performing 320 million interpolation operations per second. Since the depth buffer interpolation and comparisons and color interpolations are all performed in parallel, the GX4000 Image and Z-Buffer Memory Units can render close to 200,000 shaded and depth buffered 100 pixel area triangles per second. Vector drawing performance is over 350,000 one centimeter vectors per second.

2.1. Command Processing

Input to the Drawing Processor is a series of high level graphics commands. The MCIC accepts drawing commands from the GX4000 graphics arithmetic processors (which perform transformations, lighting models, tesselation, and other high level graphics functions). The MCIC handles command setup and controls the multiple SLPICs. It also performs Y address calculations for all commands other than vector draw.

The SLPICs perform the inner loop calculations for vector draw, scan-line shading and depth buffering, rectangle fill, and pixel block move. The SLPICs also control the image memory RAMs including all cycle timing, refresh control, and video display refresh.

When the Image and Z-Buffer Memory units are performing depth buffered rendering functions, the MCICs on the Z-Buffer and Image Memory modules are all performing the same setup operations at the same time, and loading the SLPICs with the same X and Y scan-line information. Each MCIC also sends the proper color or depth information to the SLPICs it controls. The SLPICs on the Z-Buffer module interpolate the depth value and perform the depth comparisons while the SLPICs on the Image Memory modules are interpolating the color data for the same pixels. The SLPICs on the Z-Buffer module inhibit the write to image memory by the corresponding SLPICs on the image memory modules depending on the result of the depth comparisons.

3. Drawing Processor Command Set

The Drawing Processor receives and replies to commands over a synchronous message oriented bus. Each message contains a complete Drawing Processor command. Most commands include a control word which controls patterning, shading, and depth buffering. The command set is described below.

3.1. Register Read/Write

This command allows the Drawing Processor internal registers to be loaded or read back. These registers control drawing and display functions. The register set includes:

Bank Select	Foreground Value 0 - 5	Background Value
Writemask	Vector Pattern 0 and 1	Image Screen Origin
Pixel Function	Pattern Prescale 0 and 1	Overlay Screen Origin
Pixclp	Area Pattern (8 x 8)	Depth Buffer Function
Zoom Factor	Read Mask	Look-up Tables

3.2. Vector

The GX4000 Drawing Processor renders arbitrary vectors expressed as two endpoints. The X and Y endpoint addresses are specified in a 12.4 format allowing subpixel addressing accuracy. Vectors can also be optionally depth buffered and the drawing color can be optionally interpolated for depth queueing. The execution of this command will be discussed in detail later in this paper.

3.3. Triangle

Shaded and depth buffered triangles are handled directly by the drawing processor. Edge and scan-line differentials are supplied with the command information because these can more easily be calculated by the graphics arithmetic processors. This also reduces the complexity of the drawing processor VLSI chips. Subpixel addressing is used for all addressing calculations for this command to insure that the triangles are rendered with no holes and no overlapping pixels. The execution of this command will also be discussed in detail later in this paper.

3.4. Scan-Line

This command is provided as a subset of the triangle command to handle the cases where the triangular representation would have only one or two lines in it. This results in lower processing and data transfer overhead. All other triangle shading and depth buffer functions are provided.

3.5. Rectangle

The rectangle command fills the rectangle specified by the upper left and lower right corner points.

3.6. Point Read/Write

This command reads or writes a single pixel to or from the specified location.

3.7. Image Read/Write

This command allows a pixel array to be written to or read from a rectangular region specified by the upper left and lower right corner addresses.

3.8. Block Move

The block move command is used to move a rectangular region of pixels from one location to another. The source region is specified by two diagonally adjacent corner addresses. The first address specifies the start point of the move and the second address specifies the end. The destination location is specified with a single address which must correspond to the same corner as the start point of the source region. Because data can not be rapidly moved between scan-line processors, the difference between the source and destination region Y addresses must be an integer multiple of four pixels. Other cases are handled by using the image read and write commands.

4. Master Controller IC Implementation

The master controller IC is implemented with a 1.5 micron gate array using 8700 gates. Figure 3 shows a block diagram of the master controller IC.

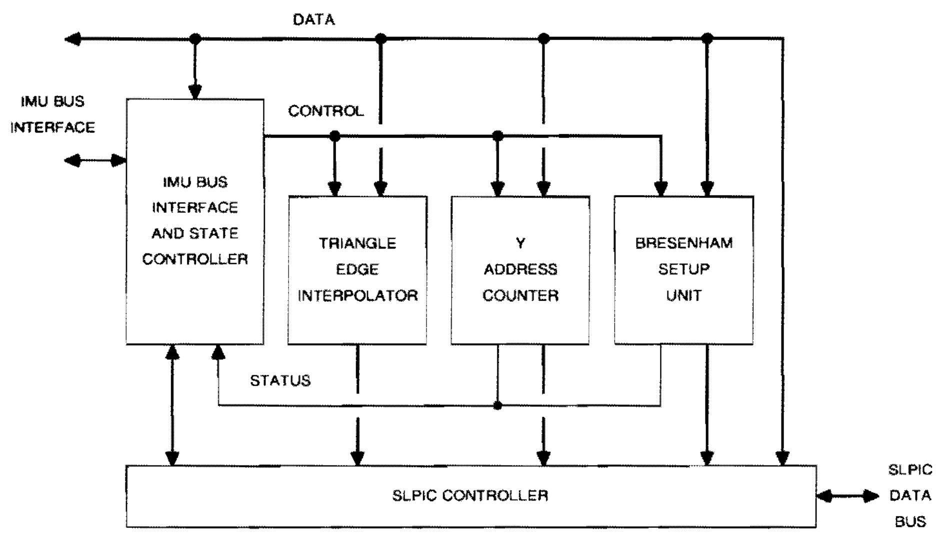

Figure 3: Memory Controller IC block diagram.

4.1. State Controller

The state controller is a synchronous state machine which controls reading and writing of IMU BUS commands, loading of all registers, and processing of all commands. The state controller directly controls command transfers to and from the IMU BUS interface. Handshaking signals are provided by the SLPIC interface controller to control transfers to and from the scan-line processors.

The state controller performs conditional operations based on status signals from the Y address counter and the Bresenham setup unit. This allows the state controller to execute a different state flow when the Y address reaches a triangle vertex or the end of the triangle, or when the Bresenham U and V parameters must be swapped.

4.2. SLPIC Interface Controller

The SLPIC interface controller incorporates a two word fifo to allow maximum rate synchronous transfers of command and pixel data to the SLPICs. This logic also controls data transfers to and from the look-up tables and other external control hardware.

4.3. Y Address Counter

The Y address counter consists of a counter, a comparitor, and two holding registers. The registers store the intermediate and bottom vertices of the triangle being processed. This logic informs the state controller when the Y counter is equal to the appropriate vertex so that a different state flow can be executed.

4.4. Triangle Edge Interpolator

This logic includes two interpolators for calculating the new X addresses for each scan line for each step in Y. One of the two interpolators has an additional set of holding registers to store the third triangle edge parameters.

Each interpolator is wide enough to maintain 1/16th pixel accuracy across the entire triangle. Special rounding logic is provided to round the left edge X address up to the nearest whole pixel. The error term (up to 15/16ths of a pixel) is transferred to the SLPIC to allow more accurate color and depth calculations.

4.5. Bresenham Setup Unit

The Bresenham setup unit is used primarily for calculating the parameters for the vector drawing algorithm performed by the SLPICs. The basic algorithm performed is the Bresenham line drawing algorithm [Bresenham, 1965].

This logic is also used as a counter to control pixel data read and write transfers and rectangle fill commands.

5. Scan-Line Processor Implementation

The scan-line processor IC is implemented using a 1.5 micron channel-less array with 15,000 gates. Figure 4 shows a block diagram of the SLPIC.

5.1. State Controller

The state controller contains the state sequencers for all SLPIC commands and generates the control signals to the rest of the chip to route the data and perform the desired operations. It also controls register read/write operations and video display cycles.

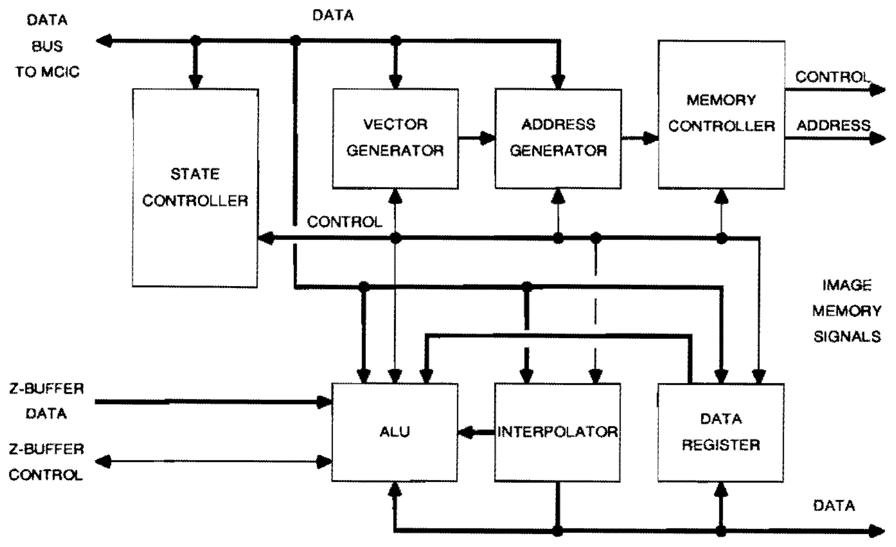

Figure 4: Scan-Line Processor IC block diagram.

5.2. Address Generator

The address generator has two pairs of X and Y up/down counters which are used for source and destination addresses. The X counters can count by five pixels each clock cycle for horizontal vectors or rectangle fills to allow five pixels to be written at once. The least significant two bits of the Y counter are used to determine if the pixel address should be handled by the particular SLPIC.

The SLPIC incorporates an 8 x 8 area pattern register which is indexed into by the LSBs of the X and Y address. This results in the area pattern being replicated, or tiled, across the screen starting from the upper left corner. The pattern register can be used to select between the foreground/background register or select between writing or inhibiting the pixel.

The address generator also generates addresses for display cycles to allow panning and zooming. Two pairs of counters are provided to allow independent control of the image memory bank and the overlay bank.

5.3. Vector Generator

The vector generator is used to control the clocking of the address counters in the address generator block when performing vector draw operations. This block contains the error accumulator used to implement the Bresenham vector generation algorithm. A sixteen bit data path is implemented to allow 1/16th pixel accuracy to be maintained in vector calculations. The address generator also contains an iteration counter to keep track of the number of pixels remaining to be drawn.

Two 16 bit vector patterns are provided to allow stipple patterns. The pattern can select between the foreground/background register, or inhibit pixel writing. Each register has a prescaler which can be set to rotate the vector pattern one bit position for every 1, 2, or 4 pixels in a vector. When vector patterns are enabled, an option is provided to force writing of the first and last 3 pixels in a vector, overriding the vector pattern.

5.4. Arithmetic Logic Unit

The ALU contains the logic for performing pixel function and depth buffer function calculations. The 8-bit ALU performs arithmetic (Src+Dest, Src-Dest, Dest-Src) operations, or per-bit logical operations using a 4 bit mask. The arithmetic operations may be clipped to 8 bits on overflow conditions by setting a flag. Conditional write operation (write only if the source is non-zero) may also be selected. A 16-bit magnitude comparitor is provided to perform depth buffer comparisons.

One of the five foreground value registers can be selected for vector and horizontal vector commands. The select register can be used either as the value to write to memory or as the Pixel ALU input value.

5.5. Interpolator

The interpolator block calculates a new value for each pixel in an interpolated scan line or vector by adding a slope to the previous value. The calculations are 32 bits wide in a 17.15 two's complement format, where the msb is the sign bit, the next 16 bits are the integer portion, and the low 15 bits are fractional information for increased precision. There are three data paths in the interpolator: one for interpolating a new value down the left edge of a triangle; a shift-and-add multiplier path to calculate the value difference between the edge and the first pixel to be written (using fractional addressing information from the MCIC); and an add-accumulate path for calculating pixel values across a scan line or along a vector.

5.6. Data Registers

This block contains five 16-bit latches which are used for block move operations. These latches allow an entire five pixel block to be read into the scan-line processors, and then written out into a new memory location. This allows substantially higher performance than if single pixel reads and writes were performed. A data path from the data registers to the Pixel ALU is provided so that a pixel function can be applied to block moves.

5.7. Memory Control

The memory control block handles all memory timing and control signals. Memory cycles are included for reads, writes, read-modify-writes, dynamic RAM refresh cycles, and video DRAM shift register loads. The memory timing generator will always try to perform page mode cycles to minimize the memory cycle times.

6. Vector Drawing

The GX4000 drawing processor can evaluate and draw vectors expressed as a pair of endpoints. The vector setup and the Bresenham vector generation algorithm are all performed in hardware for maximum performance. Interpolated and depth buffered vectors can also be handled.

6.1. Vector Setup

The vector setup is performed by the Master Controller IC. The MCIC accepts vectors from the graphics arithmetic processors as a pair of endpoints and calculates the terms necessary to perform the Bresenham algorithm. The setup parameters are passed to the scan-line processor chips, which contain the vector generators (see below).

6.2. Vector Generation

Each scan-line processor chip contains a Bresenham vector generator. All chips are passed the same setup vector, and write only the portions of the vector that are on the set of every fourth horizontal line handled by each chip. The vector generators operate at the full clock rate (20 Mhz) for pixels which are not handled by the particular scan-line processor, and only slow down to memory rates when a pixel is to be drawn. This allows the four scan-line processor chips to process each vector in parallel for higher performance. Horizontal vectors are special cased to allow multiple pixels to be written each clock cycle. The Drawing Processor can render over 350,000 one centimeter vectors per second.

7. Triangle Drawing

The triangle rendering algorithm requires that some preliminary calculations be performed by the GX4000 graphics arithmetic processor(s) to calculate edge and scan-line differentials. This is done to reduce the complexity and processing time of the drawing processor. Since these calculations involve several divide operations, significant time would be spent in performing these operations in the VLSI chips.

The following parameters are necessary to describe a shaded, depth buffered triangular patch to the Drawing Processor:

X1	dX1/dY			
X2	dX2/dY			
X3	dX3/dY			
Ytop	Yvertex	Ybottom		
Z1	dZ1/dEdge	Z2	dZ2/dEdge	dZ/dX
R1	dR1/dEdge	R2	dR2/dEdge	dR/dX
G1	dG1/dEdge	G2	dG2/dEdge	dG/dX
B1	dB1/dEdge	B2	dB2/dEdge	dB/dX

Figure 5 shows the relationships of these parameters for the two general cases of triangles. The term "F" has been used to represent the R, G, B, and Z values and differentials.

This information is used by the MCIC and SLPICs to calculate and draw each scan-line. The X1 and X2 terms are the left and right X positions on the scan-line given by Ytop. The dX1/dY and dX2/dY terms are the slopes of the first two sides of the triangle. The X positions on subsequent scan-lines are calculated by adding the slopes to the current X values. This is done so that it takes only one cycle to find the next X values, instead of possibly several cycles if a Bresenham walk was used.

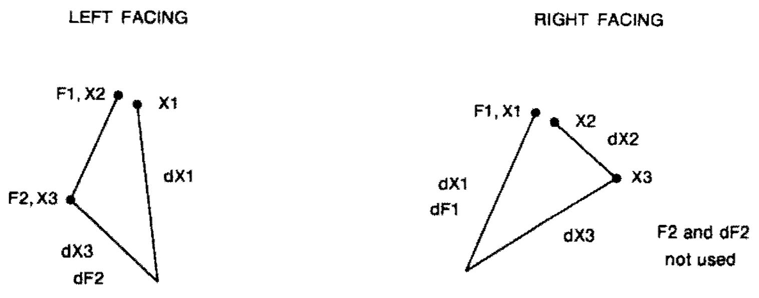

Figure 5: Triangle parameter definitions.

X1 and dX1/dY are defined to be the edge that has the longest span in Y (Edge1). Similarly, X2 and dX2/dY define Edge2. Two X terms are necessary to account for the special case of a triangle with a horizontal top edge, and the case where the true vertex of the triangle falls on a sub-pixel boundary so that the first scan-line that is actually drawn contains more than one pixel.

There are two triangle modes: left facing triangles, and right facing triangles. A left facing triangle will have Edge1 on the right side of the triangle, and a right facing triangle will have Edge1 on the left side. This is specified by the opcode passed to the master controller. If dX2/dY is more positive than dX1/dY, then Edge 1 is the left edge, and vice versa. This information is needed since the scan-lines are always drawn with the X address increasing.

The MCIC is only concerned with the address information. However, it must properly initialize the SLPICs since each processor handles a different scan-line. (For this discussion, the symbol F will be used to represent R, G, B, or Z, as the processing is the same.) Since the scan-lines that make up a patch are always drawn with the X address increasing, F1 and dF1/dEdge define the starting values on the left edge of the triangle. For a left facing triangle, F2 and dF2/dEdge are used along with X3 and dX3/dY when Y becomes equal to Yvertex. For right facing triangles, the F2 and dF2/dY terms are ignored, only X3 and dX3/dY are needed. Also, since triangles are by definition planar, the dF/dX terms are constant.

The value of F across a scan-line is calculated in the SLPICs by adding dF/dX to F for each pixel. The initial value of F on subsequent scan-lines is calculated by adding dF/dEdge to the current initial value of F. Since the calculations for F are done in the SLPICs and each chip handles every fourth scan-line, they have to be initialized with different values of F. The MCIC will load all four SLPICs simultaneously with the initial values of F, dF/dEdge, dF/dX, and Y. The SLPICs will look at the two least significant bits of the Y address and run their interpolators zero, one, two, or three times as appropriate so that they each have the proper initial values of F for the first scan-line they will draw. The SLPICs will then shift the value of dF/dEdge left two bits so that they will interpolate to every fourth line.

For each scanline in the triangle, the MCIC will generate a pair of X coordinates by running the two edge interpolators. The interpolators are in 12.15 format, using a 12.4 starting value and a 12.4.11 differential value. The .4 part of the starting and differential value is a subpixel address, and the extra 11 bits in the differential are a fractional component that allows subpixel accuracy to be maintained even over edges the full width of the screen. Once calculated, the two X coordinates need to be rounded so that only pixels inside the patch will be drawn: the left X coordinate is rounded up to the nearest whole pixel, and the right X coordinate is rounded down to the nearest whole pixel. The MCIC passes the rounded left and right X addresses, and the subpixel (4 bit fractional) part of the difference between the rounded and unrounded left X coordinate (referred to as Xerr) and the Y coordinate to the SLPIC whose line number matches the two least significant bits of the Y address. Once the MCIC has completed passing the scan-line data to the appropriate SLPIC, it calculates the data for the next scanline and passes it to the next SLPIC. This allows all four SLPICs to be writing pixels into image memory in parallel.

The Xerr term is used by the SLPICs to adjust the starting value of F by calculating Xerr * dF/dX and adding it to F. This is to ensure that the value of F that is written at the starting pixel is the true value and not the value calculated at the edge, since the rounded X coordinate may be as much as 15/16s of a pixel away from the edge. This will prevent the holes and jagged edges present at adjacent triangles and intersections in some depth buffered images on previous systems.

Each SLPIC draws the scan-line by interpolating F for each pixel by adding dF/dX to a temporary copy of F. When the scan-line is complete, each SLPIC calculates the next starting value of F by adding dF/dEdge, which represents the slope of F along the left edge, to F.

8. Performance

At the time that this paper was written, the VLSI drawing processor was fabricated and working in prototype systems. Complete benchmarking has not been done although preliminary testing indicates that the simulated performance goals have been met. Detailed simulation software has been developed to allow accurate system analysis. A wireframe and shaded patch representation of an automobile (shown in Figure 6 and 7) have been run through this simulator to show the performance of the chip set. The simulated performance is shown below:

8.1. Wireframe Performance

Wire Frame Database of Chrysler Laser (courtesy of Chrysler Corporation).

Number of Vectors:	99,236
Average Vector Length:	4.77
Total Pixels Drawn:	473,217
Execution Time:	104.2 milliseconds

8.2. Shading Performance

Shaded Patch Database of Chrysler Laser (courtesy of Chrysler Corporation).

Number of Triangles:	64,696
Left Facing Triangles:	22,970
Right Facing Triangles:	41,726
Total Scan-lines:	296,388
Total Pixels Rendered:	802,185
Average Triangle Size:	12.4 pixels
Rendering Time:	237.7 milliseconds

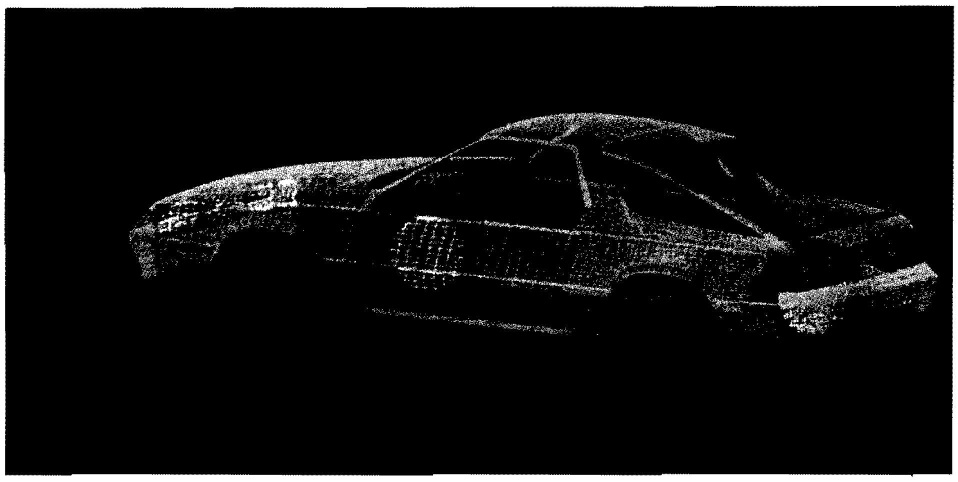

Figure 6: Wireframe image (database courtesy Chrysler Corporation)

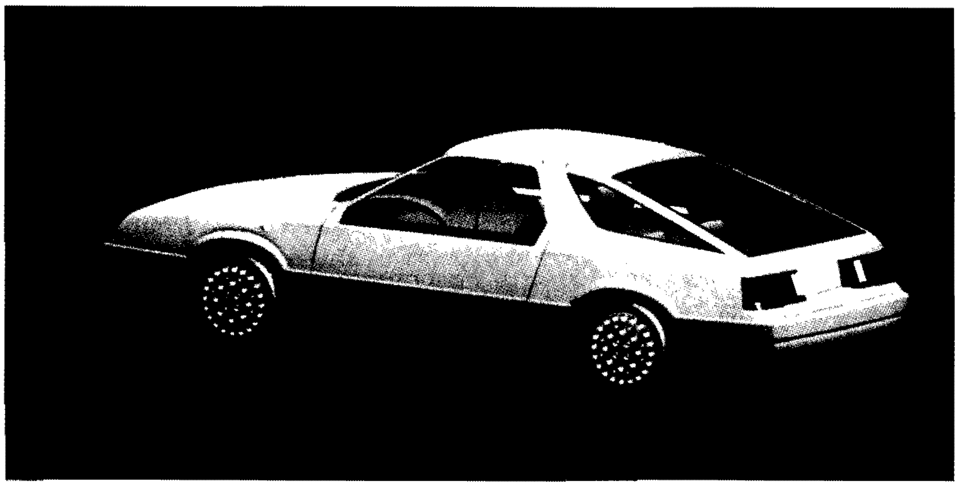

Figure 7: Shaded image (database courtesy Chrysler Corporation)

9. Acknowledgements

Thanks to the entire GX4000 design team for their participation in the development of the graphics system which employs these chips. Special thanks to Fred Oliver for his assistance in the algorithm development, to Chan Verbeck for the development of the simulator used to test the algorithms before implementing in VLSI, and to Dave Youatt for generating the color images, and to Greg Bartlett for creating the figures.

10. References

Bresenham, J.; "Algorithm for Computer Control of a Digital Plotter," *IBM System Journal* **4**,1 (1965), 25.

Fuchs, H.; "Distributing a Visible Surface Algorithm Over Multiple Processors," *Proceedings (ACM)* (1977)

Fuchs, H. and Poulton, J.; "Pixel-Planes: A VLSI-Oriented Design for a Raster Graphics Engine," *VLSI Design* **3** (1981), 20.

Fuchs, H., Goldfeather, J., Hultquist, J.; "Fast Constructive Solid Geometry Display in the Pixel-Powers Graphics System," *Computer Graphics (ACM)* **20**,4 (1986), 107.

Gouraud, H.; "Continuous Shading of Curved Surfaces," *IEEE Transactions on Computers* **C-20**,6 (1971), 623

Niimi, H., Imai, Y., Murakami, M., Tomita, S., and Hagiwara, H.; "A Parallel Processor System for Three-Dimensional Color Graphics," *Computer Graphics (ACM)* **18**,3 (1984), 67.

Parke, F.I.; "Simulations and Expected Performance Analysis of Multi Processor Z-Buffer System," *ACM Computer Graphics* **14**,3 (1980), 48

Sutherland, I.E., Sproull, R.F., Schumaker, R.A.; "A Characterization of Ten Hidden-Surface Algorithms," *Computing Surveys* **6**,1, (1974), 293

Swanson, R.W., Thayer, L.J.; "A Fast Shaded-Polygon Renderer," *Computer Graphics (ACM)* **20**,4 (1986), 95