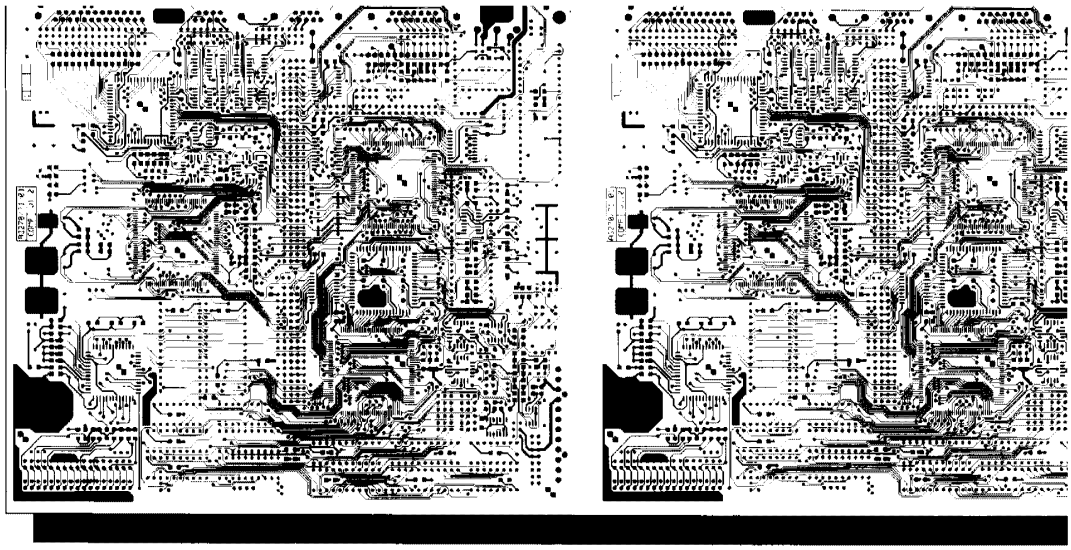


Multiprocessor Raster Plotting

Roger D. Hersch, Pierre Demartines, and Doris G. Fridman
Swiss Federal Institute of Technology, Lausanne

Jean-Marc Aberle, Peter Pisan, Andre Decurnex, and Ralf Rogner
Electronic Industrial Equipment, Geneva

Working with algorithms for fast, efficient plotting, this scalable, multiprocessor raster image processor generates printed circuit plots in alternating band buffers.

To generate images on color and black-and-white printers and plotters, we rely on raster imaging techniques. Most raster imaging systems use RISC-based rasterization processors to generate pages efficiently. Systems incorporating these processors are primarily designed to generate pages on middle resolution devices (300 to 600 dpi). Even though you can also use such systems for high-resolution photocomposers, they might slow down the imaging process considerably. Current drum-based, high-resolution laser plotters rotating at 16,000 revolutions per minute can expose a film made up of 64,000 scan lines in four minutes, but single-processor systems sometimes require more than a quarter of an hour to generate complex images.[1]

Multiprocessor systems can increase rasterization speeds.[2] But because interconnecting conventional microprocessors is expensive, most raster image processors (RIPs) contain at most two processors that contribute to rasterizing geometric shapes. Recently, however, researchers designed several microprocessors specifically for multiprocessing. The transputer was the first commercially available 32-bit processor with communication links for multiprocessing.[3] Other manufacturers are now marketing new processors with similar communication links.[4,5]

0272-17-16/92/0700-0079$03.00 © 1992 IEEE

IEEE Computer Graphics & Applications, Vol 12, No 4, July 1992, 79-87

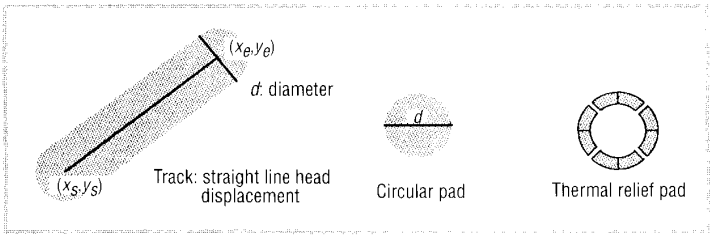


Figure 1. Description of tracks and pads.

Even if you build a cost-efficient multiprocessor RIP, multiprocessing still requires adequate algorithms to achieve high efficiency. We present new algorithms and analyze the performance of a multiprocessing system that rasterizes printed circuit plot descriptions.

Previous work

Researchers have widely studied parallel architectures for rendering 3D graphics. But since 3D rendering differs from 2D rasterizing, parallel 3D architectures are far from optimal for solving 2D or 2.5D rasterization tasks. Contributions to multiprocessor 2D rasterizers are rare. Relying on the hardware that was typically available during the early eighties, some researchers described a microprogrammed rasterizer based on dedicated very large scale integration rasterization chips.[6] They designed this rasterizer for on-the-fly generation of complex plots on electrostatic devices. Their parallel architecture was scalable to some degree, but it required sophisticated hardware built around microprogrammable controllers and special scan conversion and filling chips.

More recently, other researchers described a pipelined parallel architecture for page rasterization on fast middle resolution printers.[2] This architecture provides not only parallel rasterization but also parallel sorting of primitives according to their geographic locations. These researchers assume, however, that full-page bitmaps are provided and therefore neglect the real-time problems arising with on-the-fly rasterization.

Masks for printed circuits

In the last decade, mask creation for producing printed circuits has evolved considerably. Manual circuit layout by pasting strips on mylar paper has been replaced by interactive computer-aided placement of components and interconnections. The final layout of a printed circuit board is described by standard plot descriptions like the Gerber plotter format.[7] These descriptions specify an entire plot in terms of flashes that use heads with given shapes and of tracks produced by head displacements. Appropriate symbols specify soldering pads, which are given by their pad shapes (see Figure 1).

Older photoplotters were based on variable aperture head displacement. The plotters imaged tracks on film by mechanically displacing the plotter head (as with flatbed plotters) or by combining drum rotation and horizontal plotter head displacement (as with drum plotters). These mechanical photoplotters executed head displacement commands received by the host system. Since each plot strip was produced by a corresponding head displacement, total plot generation time depended on head displacement speed and plot complexity.

Today's plotter imaging devices are based on rotating laser beams or light emitting diode arrays (see Figure 2). These new scanning devices require powerful raster imaging systems to generate raster memory plot images at the required speed and resolution. The film is loaded into the interior of the drum and held stable by aspiration. It can then be imaged by the rotating laser beam. The laser beam is modulated by a "video" stream originating from the raster image memory containing the scan-converted image.

These new imaging devices provide resolutions from 2,000 to 5,000 dpi. Laser scanning devices generally work at a constant rotation speed. Mastering image generation complexity becomes the task of the raster imaging subsystem.

Synchronous raster plotting

The memory and processing requirements imposed on the raster imaging system are extremely severe. For a plot dimension of 650 × 650 mm at a resolution of 2,540 dpi, the system needs a virtual image memory buffer of approximately 500 Mbytes. It is too expensive to have the complete image

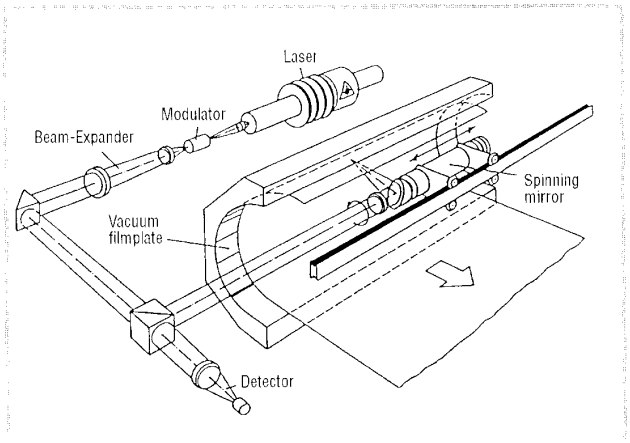


Figure 2. A drum-based high resolution laser plotter.

frame buffer in dynamic RAM. Alternate solutions include the use of one or several hard disks to store the complete frame buffer or dynamic rasterization of sorted plot segments on alternating band buffers.

For price and performance reasons, we prefer to avoid hooking disks onto the raster imaging system. For most tasks, the system should be powerful enough to generate the raster image band by band in synchronization with the scanning laser plotter. The raster imaging system will incorporate as many processors as needed to scan-convert most plots (for example, 98 percent) in the available time. The system can handle the few remaining ultra-high-density plots by a special preprocessing step that stores intermediate data on the host processor's disk or by a special plotter stop and restart mechanism.

Severe constraints limit a multiprocessor system that must rasterize horizontal bands of a high-density plot in synchronization with a scanning laser plotter. For efficiency, all contributing processors should have approximately the same load. Since graphic plot primitives are not evenly distributed on the plot surface, the size of the horizontal band image partitions (facets) to be processed by individual processors should be proportional to the inverse of their respective load. For this purpose, we propose load prediction algorithms that let the system evaluate the load quickly and determine the optimal facet sizes to be computed within one horizontal band. Since different processors synthesize image facets separately, our rasterization algorithms are required to work coherently across facet boundaries.

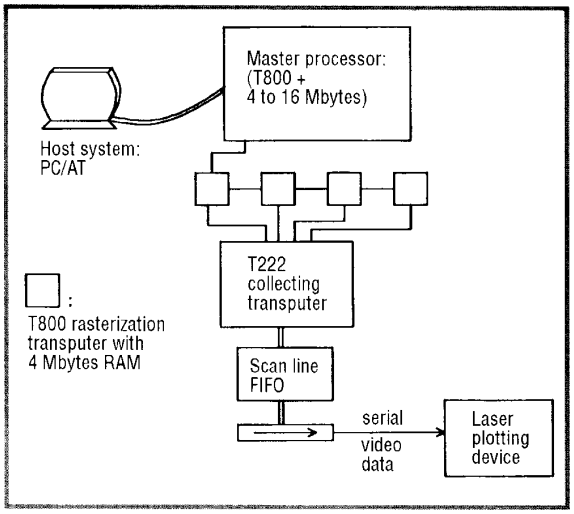


Figure 3. A six-processor raster imaging system.

General architecture

A printed circuit plot is essentially composed of tracks and soldering pads. The Gerber plot format describes each plot segment (track) by an exposition head diameter and a head displacement. We can describe each pad by an assembly of superimposed vertically convex shapes (see Figure 1).

To generate image plots in synchronization with laser plotting, the algorithm segments the page map into horizontal bands. Before plotting, the master processor distributes graphic primitives along the horizontal bands to which they contribute. At image generation time, the output subsystem reads each rasterized image band, serializes it, then sends it for laser beam modulation while the RIPs scan-convert the next band.

In preliminary tests, we found that initialization tasks (like scanning the input description, converting the primitives into a binary intermediate form, and sorting the graphic primitives according to their geographic locations) require less processing than the real page-map rasterization tasks—outline expansion, clipping, scan conversion, and filling. Therefore, image generation is the main subject of parallelization. We reject a pipelined solution consisting of separate processors handling sorting operations, outline expansion, clipping, scan conversion, and filling because its parallelization potential is poor and its speed is limited by the slowest pipeline element.

To design a scalable architecture, we decided to parallelize image generation according to geographic page-map locations. We divided a page map containing 500 Mbytes into numerous horizontal bands. Each horizontal band itself was divided into a given number of facets. All available RIPs can generate one band simultaneously during a given time interval. Alternately, several groups of processors can work on several bands simultaneously. Each RIP generates a part of a horizontal band called a *facet.*

We based the actual scalable, multiprocessor architecture used for parallel raster plotting on transputers. Each transputer incorporates four communication links. These links allow information to be transferred between the memories of two interconnected transputers at the rate of 1.5 Mbytes per second. Communication links are hooked to direct memory access controllers integrated into the transputer chip. Therefore, the system can exchange data between neighboring processors without requiring processing power. All types of data transfer—such as distributing graphic primitives and collecting raster data—are carried out via intertransputer communication links. These integrated communication features let us design a bus-less, ultracompact multiprocessor RIP.

The multiprocessor rasterization system (see Figure 3) is controlled by a master processor (T800 transputer) communicating with an IBM/PC AT host station. During job initialization, the master processor receives plotting commands from the host system, sorts them according to bands, analyzes the load distribution, and computes optimal facet sizes in each band. When generating the images, the system distributes plotting primitives to the facet processors. Facet pro-

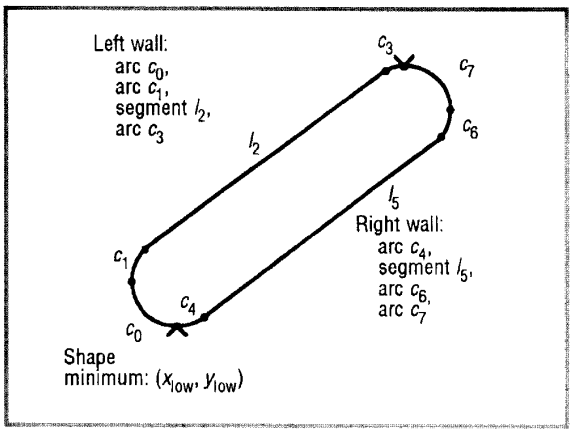


Figure 4. The ordered convex shape description.

cessors work in parallel to simultaneously rasterize the content of one horizontal band. They are responsible for outline expansion, clipping, scan conversion, and filling. The collecting processor receives raster data from each facet processor, stores this data to form complete scan lines, and sends it for laser beam modulation in synchronization with the rotation of the output device. Thanks to fast transputer communication links, the software transmits the raster data to the collecting processor. There is no need to build special hardware for scan-line data extraction and transmission. Therefore, the complete system is under software control. We can change plotter imaging size and resolution without redesigning the hardware.

The master processor must be able to communicate with each of the RIPs as well as with the collecting processor. For this purpose, we developed a communication package that uses dedicated, high-priority tasks running on each processor to provide message passing and routing.

Graphic operations

The RIP's task consists of outline expansion, clipping along facet boundaries, scan conversion, and filling. For scan conversion and filling, the system translates all graphic primitives into one graphic primitive: the *vertically convex shape.* This closed shape intersects any horizontal line at only two intersection points. Complicated shapes having holes are created by a succession of black and white vertically convex shapes.

To obtain the track plot outline, the algorithm computes an outline description from a given exposition head diameter and a head displacement. A straight line displacement produces an outline description that is a convex shape. A circular head displacement does not produce a convex shape, but segmenting it into several head displacements delimited by quadrants gives vertically convex shapes. Pads are produced by a succession of superimposed black and white convex shapes.

For fast scan conversion and filling, the system segments each convex shape outline description into a *left wall* and a *right wall.*[8] These walls describe the two contour parts between the shape's minima and maxima (see Figure 4).

To achieve separate rasterization of facets by different processors, segmenting continuous shapes into pieces and filling each one separately must have no effect on the resulting discrete shape. Filling any partition of the original shape separately generates discrete partitions that can be assembled to form exactly the same discrete shape as the one that would have been generated by filling the original continuous shape (partitioning criterion). Vertical incremental scan conversion and filling algorithms[9] ensure correct filling of segmented shapes (see Figure 5).

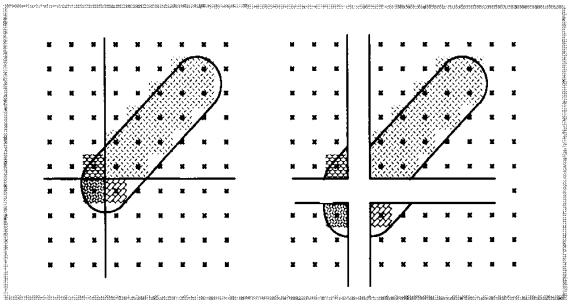


Figure 5. Clipping, filling, and reassembling a shape generated by several facet processors.

For a one-pass interior fill, we need to cross all scan lines, from the lowest to the highest. At each scan line we must compute incrementally[10] the intersection with all shape contour parts (complexity is linear to the shape perimeter projected on the y axis).

We already segmented each vertically convex shape outline into one left and one right wall. Since the system knows the number of intersections with horizontal scan lines, the algorithm allocates memory to each wall for storing all intersection points.

Figure 6 describes the relative processing times for facet generation of normal and high-complexity plots. For these performance figures, facet size is set at 2,048 × 2,048 pixels, which at the given resolution of 100 dots per mm corresponds to a plot facet dimension of 20.48 × 20.48 mm. The raster image memory space required by this facet size is 500 Kbytes. We measured processing times on one T800 20-MHz transputer.

Thanks to the transputer's internal floating-point unit, shape outline expansion and clipping take only a small frac-

tion (less than 17 percent) of overall facet generation time. Therefore, we can adapt facet size to available memory, number of transputers, and plot size. Decreasing facet size will not produce large overheads, even if the number of clipping operations increases linearly with decreasing facet areas.

Load prediction

Because of the real-time constraints induced by on-the-fly rasterization of image bands in synchronization with the plotting engine, we must segment each band into several facets, each handled by a separate processor. The distribution of graphic primitives along one band will not be uniform. Image borders have a lower plot density than the center part of the image. Therefore, if we assigned a uniform facet size to each contributing processor, we'd get an unfavorable load balance (see Figure 7).

Procedures	Normal plot	High-density plot
Number of primitives	46	2,991
Plot outline expansion and clipping	18 ms	1,502 ms
Scan conversion of plot outlines	201 ms	4,342 ms
Filling	250 ms	2,556 ms
Facet generation time	529 ms	9,348 ms

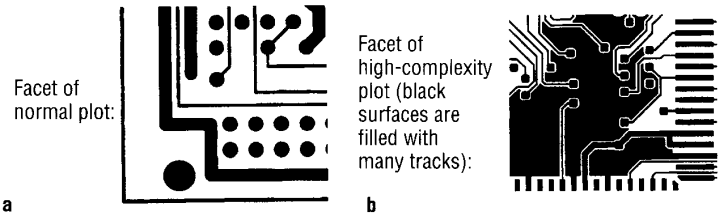


Figure 6. Processing time for outline expansion, clipping, scan conversion, and filling of a normal and of a high-density plot.

Since plotting systems generally have several speed settings, we should choose the laser imaging speed according to the worst-case plotting load facet. To maintain high efficiency despite uneven load distribution and still be able to choose the correct laser imaging speed, our accurate load prediction algorithm estimates the rasterization speed of all supported graphic primitives.

Load prediction for rasterizing operations is based on the complexity analysis of shape clipping, scan conversion, and filling. The number of clipping steps is proportional to the size of the considered graphic primitive's bounding box. Scan conversion is proportional to the projection of the shape perimeter onto the vertical axis. Filling is proportional to the vertically projected shape perimeter and square to the filling surface. Every clipping, scan conversion, and filling operation requires some constant time for initialization and access to the data structure. Based on this general complexity analysis, we can set up formulas for estimating plot generation time for circular flashes and for vertical, horizontal, diagonal, and circular head displacements. To compute the different parameters used for performance estimation, we measured primitive generation time on a 20-MHz T800 transputer in terms of their free variables (diameter, height, length, angle). These measurements let us compute the unknown estimation parameters using the least-mean-square-error method (shown in Figure 8). For estimation purposes, we can consider the shape of a diagonal track to be made up of two half flashes and one rectangle (again see Figure 8).

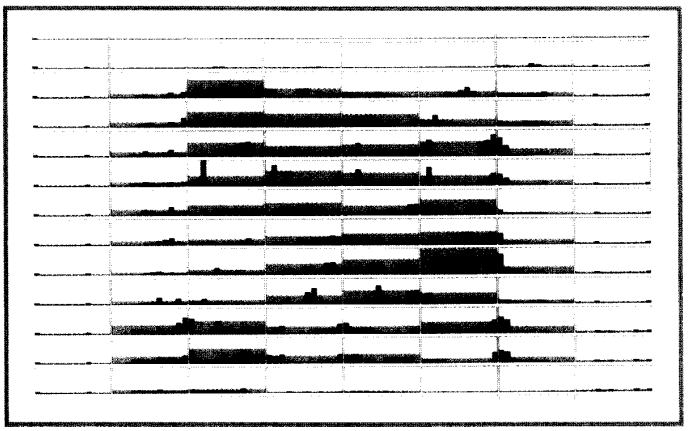


Figure 7. Unbalanced processing load resulting from uniform facet size.

Given the load estimations for each graphic primitive, we can establish a load distribution profile for each band. To do this, we divide each horizontal band into 100 vertical slices. We distribute the load of each graphic primitive among the slices it covers. Each slice therefore accumulates its share of the total load. This produces a load profile for the whole plot. Load distribution among contributing bands is mostly based on bounding-box computations (see Figure 9).

Within one horizontal band, we use rough approximations to distribute the load of a graphic primitive among slices. For the load distribution of round flashes and vertical and horizontal

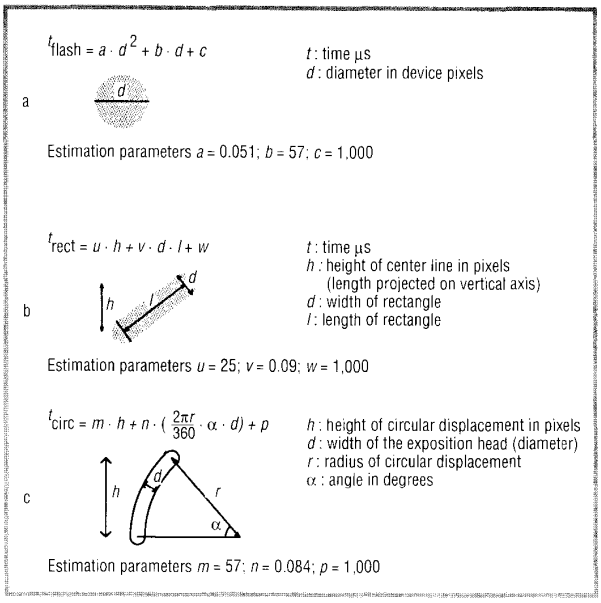


Figure 8. The least-mean-square-error method of computing unknown estimation parameters. (a) Estimated processing time for a circular flash; (b) estimated processing time for a diagonally oriented rectangle; and (c) estimated processing time for a circular plot displacement.

tracks, we divide the dynamic load proportionally to the horizontal bounding-box coverage of individual slices. The algorithm distributes the dynamic load of a diagonal track among its closest intersecting slices (see Figure 10). The static load given by a constant mainly represents data structure initialization and access time. We must add this constant to the load of each potential facet. Therefore, for a primitive likely to intersect two facets, the algorithm multiplies the constant by two and distributes it among intersecting load slices.

The exact computation of each load slice need not be precise, although the sum of the loads of neighboring slices should reflect the real load with a maximum deviation of 10 percent. Figure 7 shows the load distribution among a complete plot. Its deviation from measured performance figures is less than 10 percent.

By integrating the load profile of each band, you can compute the band having the worst-case load. You can then use this worst-case load to set up the plotter engine's speed or, if possible, to program the plotter to stop and restart the plotting engine. The load profile also provides the information needed to compute the facet sizes in each band.

Computing facet size

Let's assume that we assigned k processors to work one band simultaneously. We should then subdivide this band

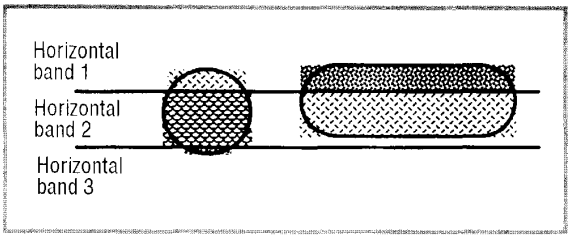


Figure 9. Load distribution among bands in proportion to bounding box coverage.

into k facets in such a way that all facets have approximately the same load. Facet width can vary within a range given by the available memory space, the band height, and the number of facets per band. Because memory size on each processor is fixed, we can only expand facet width at the cost of a reduction in band height. Reducing the number of scan lines per horizontal band entails more clipping and initialization overhead for the rasterization of graphic primitives. Therefore, we must optimize the maximal facet width given by the *facet extension factor*. The graph in Figure 11 shows how increasing band height speeds up rasterization by allocating more memory to the rasterization processors.

We can deduce that by increasing available memory size from 350 Kbytes per facet (that is, 1 Mbyte per transputer) to 700 Kbytes per facet (2 Mbytes per transputer), we improve rasterization speed by 30 percent. Further, doubling memory size to 1.4 Mbyte per facet means just an additional 10 percent improvement in rasterization time. This graph also shows that if available memory permits a maximal facet size of 700 Kbytes, you should define the size of the smallest facet as greater than 500 Kbytes. This gives you a facet extension factor of 1.4, which limits performance degradation by 10 percent because of reduced band height. If we compensate for uneven horizontal load distribution with facet sizes that can vary up to 140 percent, we can achieve performance improvements of close to 100 percent (see Figure 12).

To compute optimal facet size restricted by a given *facet extension factor*, we can't simply integrate the load distribution profile. For a k processor system, with k being a power of two, the following recursive algorithm produces optimal facet sizes that will not be larger than the standard facet size

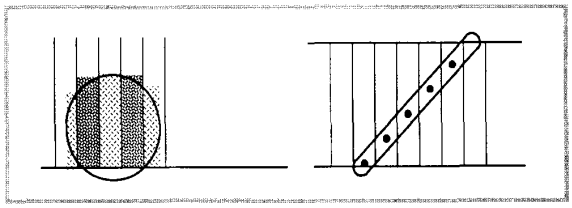


Figure 10. Load distribution among slices.

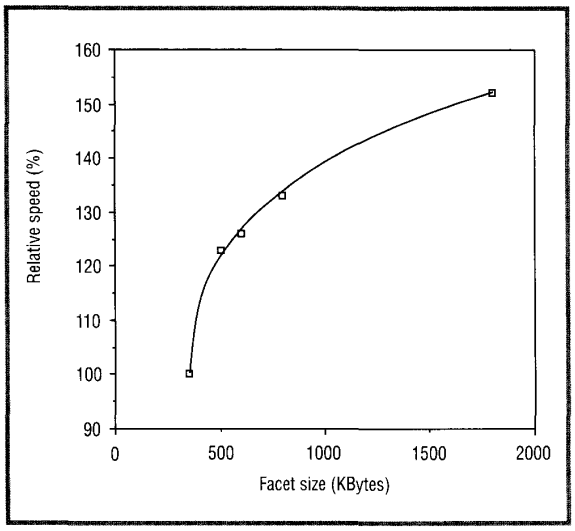


Figure 11. Performance improvements for different memory sizes.

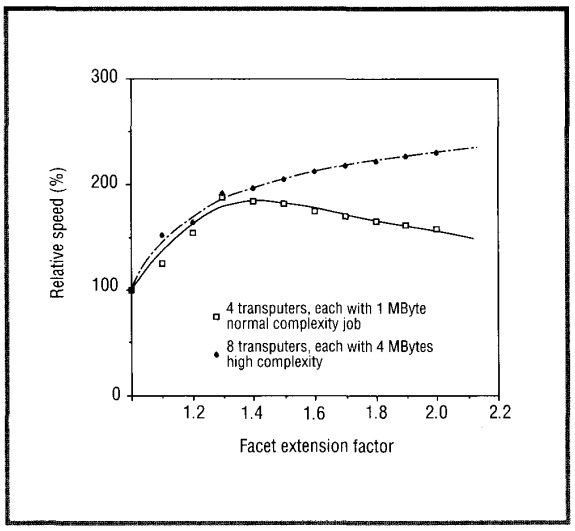


Figure 12. Performance improvements for different facet extension factors.

multiplied by the facet extension factor. The algorithm subdivides the current load interval recursively into two equally loaded new load intervals by integrating the corresponding load distribution profiles. The algorithm then distributes each new interval among half the number of processors that were available for the previous load interval. In keeping with the given facet extension factor, the load distribution program must ensure that the new load intervals are smaller than the number of available processors multiplied by the facet extension factor. To fulfill this condition, you might need to displace the border between the two new load intervals as required.

We must slightly modify the algorithm to accommodate k processors where k is not a power of two. To do this, we subdivide the current load interval into two new load intervals; we distribute the loads in proportion to their associated number of processors. Again, we might have to displace the border between the two new load intervals to ensure that the new load intervals are smaller than the number of available processors multiplied by the facet extension factor.

For the high-density plot partially shown at the top of page 79, we computed load distribution and optimal facet sizes for eight RIPs, each equipped with 4 Mbytes of RAM. The plot size is 345 × 490 mm, and resolution is 2,540 dpi. This plot includes 90,932 graphic primitives. We divided the plot area into 25 bands, each having 1,985 scan lines. We divided the available 4 Mbytes of RAM into a program and data area of 400 Kbytes and into two facet buffers, each having a maximum size of 1.8 Mbytes. Because of the large amount of available memory, we chose a facet extension factor of 2.15. Figure 13 shows the load distribution superimposed with estimated facet sizes and loads (the upper 12 bands). The maximum facet load (vertical axis within one band) corresponds to a facet generation time of 4.5 seconds.

Load estimation and optimal facet size computation ensures an even horizontal load distribution between rasterizing processors. But because of different load profiles in different bands, there is still room for load balancing between vertical bands.

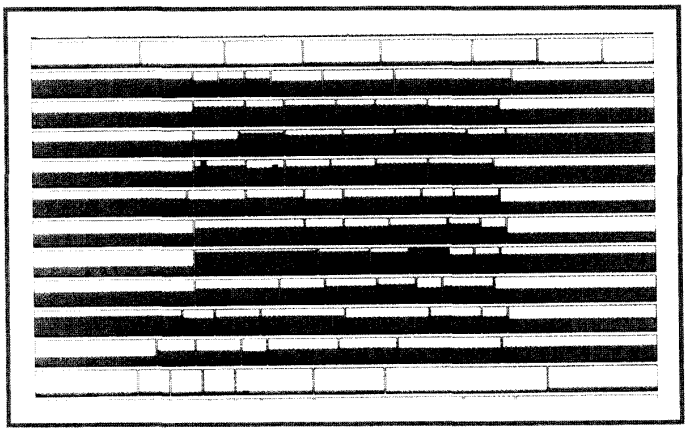


Figure 13. Load distributions, facet sizes, and facet loads.

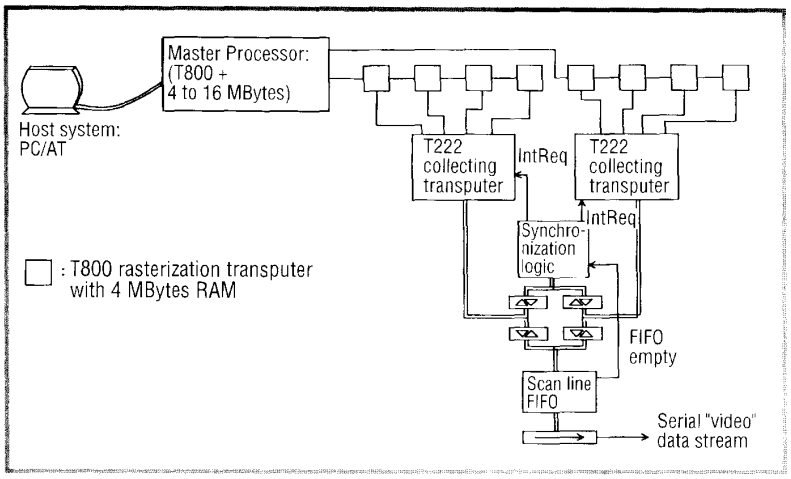


Figure 14. A high bandwidth parallel rasterization system.

Performance analysis

We implemented this multiprocessor architecture for raster imaging of printed circuit layouts in two versions: one with four rasterization transputers and the other with eight. To obtain a high bandwidth between the rasterization processors and the plotting engine, we placed all eight rasterization processors in one horizontal band. We expanded the collecting device to incorporate two collecting transputers that jointly provide eight communication links and have a common output FIFO for assembling scan-line parts and generating the full video data stream. The eight communication links offer a maximum throughput of 12 Mbytes per second, enough to expand the effective plotting size and double the plotter's resolution (see Figure 14).

We can further increase the number of processors by having multiple transputer rows work on different bands simultaneously. To collect rasterized scan-line data in a multiple-row transputer system, the system must either store and forward data through intermediate transputers or multiplex communication links with a programmable crossbar switch.[11]

If a full page map or ultrafast mass storage device were available, there would be no synchronization constraints on the rasterization processors, and plot generation speed would increase linearly with the increase of the number of rasterization processors. In that case, the only limiting factor would be the data throughput from the parallel rasterization processor's main memory to the raster data collecting device.

Since today's technology does not provide enough memory density or ultrafast disks for storing the full page map, several rasterization processors must work on the same band simultaneously. Therefore, we test the system's performance assuming that all rasterizing processors contribute toward generating one horizontal page band at a time. We also assume that we can set the laser plotter rotation speed before starting the job and that it will remain constant during job execution. We analyze the rasterization performance that can be obtained by synchronous, parallel, horizontal page-map band generation. We carried out representative tests on representative plots of various complexity. We based our performance tests on configurations with four or eight rasterization processors, each one having 1 or 4 Mbytes of main memory.

We derive the maximum rotation speed directly from the most loaded facet size. The graph on Figure 15 shows maximal rotation speed versus plot complexity given by the Gerber file size.

These tests show that memory size is almost as important as the number of available processing units. Increasing the number of RIPs from four to eight has little or no effect if total available memory is reduced by half. But, if you double the number of RIPs and increase total memory space in the same proportion, speed increases twofold.

Current laser plotting devices rotate at speeds between 3,000 and 18,000 rpm. A high-performance laser plotter rotating at 12,000 rpm can produce a 64,000 pixel by 64,000 scan-line plot in about five minutes. In Figure 15, the mea-

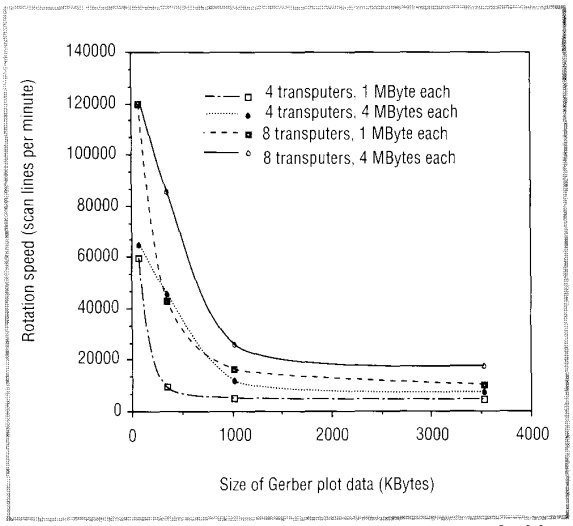


Figure 15. Rasterization speed for different plot complexities.

sured performance figures show that four T800 rasterization processors are sufficient for simple to normally complex jobs. We need an extension to eight T800 rasterization processors for plotting high-density jobs at 18,000 rpm.

Conclusions

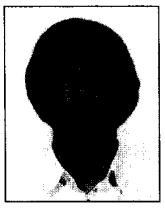
Efficient real-time rasterization of graphic primitives represents a considerable challenge for a scalable multiprocessor system. We accomplish horizontal load balancing by dynamic load prediction and facet size computation. For the most heavily loaded horizontal bands, performance figures show that, thanks to dynamic load prediction and balancing, we can attain a linear speed-up. But overall performance tends to remain lower than the maximal performance we would get if a full page map were available. This restriction results from the load's being unequally distributed in the vertical direction. We can solve this problem in several ways. One solution is purely mechanical: within certain bounds, the system could continuously adapt laser plotting speed to the predicted band load. In an expensive solution commonly found in industry, we can use disks to store intermediate raster data. While one plot is being rasterized, the system simultaneously extracts previously generated raster data from the disk and sends it to the laser plotting engine.

Both solutions are costly, since they require either more complex speed control of the output device or a disk subsystem with rather limited throughput. We are now studying a cheaper solution: Rasterization processors will carry out advanced facet rasterization and compression. Thus, these processors will generate the current band at the same time they are generating the subsequent heavily loaded bands. Such a system should provide vertical load balancing and make maximal use of available parallel processing power. ❑

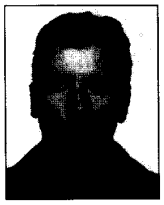
References

1. L. Brunner,"Imagesetter Speed Comparison," Seybold Report on Publishing Systems, Vol. 20, No. 10, Feb. 1991.
2. I. Birk and J.M McCrossin, "A Pipelined-Parallel Architecture for 2.5-D Batch Rasterizers," *Proc. Eurographics,* North-Holland, Amsterdam, 1990, pp. 31-40.
3. M. Homewood et al., "The IMS T800 Transputer," *IEEE Micro,* Vol. 7, No. 5, Oct. 1987, pp. 10-26.
4. R. Weiss, "32-bit Floating-Point DSP Processors," *EDN,* Vol. 36, No. 23, Nov. 1991, pp. 127-146.
5. C. Peterson, J. Sutton, and P. Wiley, "iWarp: A 100-MOPS LIW Microprocessor for Multicomputers," *IEEE Micro,* Vol. 11, No. 3, June 1991, pp. 26-29, 81-87.
6. A. Ben-Dor and B. Jones, "New Graphics Controller for Electrostatic Plotting," *IEEE CG&A,* Vol. 6, No. 1, Jan. 1986, pp. 16-25.
7. N. Andreiev, "Taking the Mystery out of Data Formats," *Printed Circuit Design,* Vol. 6, No. 8, Aug. 1989, pp. 23-27.
8. R.D. Hersch, "Descriptive Contour Fill of Partly Degenerated Shapes," *IEEE CG&A,* Vol. 6, No. 7, July 1986, pp. 61-70.
9. R.D. Hersch, "Vertical Scan-conversion for Filling Purposes," in *New Trends in Computer Graphics,* N. Magnenat-Thalmann and D. Thalmann, eds., Springer Verlag, New York, 1988, pp. 318-327.
10. R.D. Hersch, "Efficient Rendering of Outline Characters," *Proc. Society for Information Display,* Vol. 32, No. 1, Jan. 1991, Palisades Institute for Research Services, New York, pp. 55-58.
11. U. deCarlini and U. Villano, *Transputeres and Parallel Architectures: Message-Passing Distributed Systems,* Ellis Horwood, London, 1991.

Roger D. Hersch is a professor of computer science and head of the Peripheral Systems Laboratory at the Swiss Federal Institute of Technology, Lausanne (EPFL). The lab's main research topics are digital typography, 2D graphic algorithms, high-quality color rendering, and multiprocessor systems for high-speed image generation. Hersch received his engineering degree from Eidgenossische Technische Hochschule, Zurich and his PhD from EPFL.

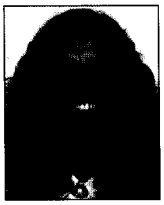


Pierre Demartines is a PhD student at the Image Processing and Shapes Recognition Laboratory of the National Institute of Technology of Grenoble (France). Since leaving the Peripheral Systems Lab of EPFL in 1990, he has focused his research on the artificial neural networks theory. Demartines received an engineering diploma from EPFL in Switzerland.



Jean-Marc Aberle is a programmer at Electronic Industrial Equipment in Geneva. He is now working on the development of parallel raster photoplotter software. Aberle holds a technician's degree from the Ecole Technique et Des Metiers of Geneva, Switzerland.

Andre Decurnex is a research and development engineer at EIE Geneva. Since joining the company in 1982, he has worked on vector photoplotters and CAD system hardware. He graduated in electrical engineering from EPFL in Switzerland.



Doris G. Fridman is an associate researcher at the Informatics Institute of the Federal University of Rio Grande do Sul, Brazil. Her current research interests include fundamental graphics algorithms and interface design. Fridman got her BS and MS degrees in computer science from the Federal University of Rio Grande do Sul.

Peter Pisan is a software engineer at EIE in Geneva. He developed the transputer-based RIP. Before joining EIE in 1990, he was engaged in research on nuclear magnetic resonance spectrometers. He graduated from the Brno Technical University in Czechoslovakia in 1984.

Ralf Rogner is a software research and development engineer at EIE Geneva. Since 1987, he has been developing CAD system postprocessors and raster photoplotter software. Rogner received a diploma in electrical engineering from EPFL in Switzerland.

Roger D. Hersch can be reached at Ecole Polytechnique Federale de Lausanne, Laboratoire de Systemes Peripheriques, INF— Ecublens, CH-1015 Lausanne, Switzerland.