


IEEE
INTERNATIONAL
CONFERENCE ON
COMPUTER-AIDED
DESIGN

ICCAD - 83

SEPTEMBER 12—SEPTEMBER 15, 1983
MARRIOTT HOTEL
SANTA CLARA, CA



DIGEST OF TECHNICAL PAPERS

Sponsored by
 IEEE COMPUTER SOCIETY
IEEE CIRCUITS AND SYSTEMS SOCIETY
In Cooperation with
IEEE ELECTRON DEVICES SOCIETY

ISBN 0-8186-0518-9
IEEE CATALOG NO. 83CH1949-7
LIBRARY CONGRESS NO. 83-81917
IEEE COMPUTER SOCIETY ORDER NO. 518

ORDER FROM: IEE SERVICE CENTER
445 HOES LANE
PISCATAWAY, N.J. 08854

IEEE COMPUTER SOCIETY
POST OFFICE BOX 80452
WORLDWAY POSTAL CENTER
LOS ANGELES, CA 90080

 THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

WIRE ROUTING EXPERIMENTS ON A RASTER PIPELINE SUBARRAY MACHINE¹

R. A. Rutenbar, T. N. Mudge and D. E. Atkins

Computing Research Laboratory
Department of Electrical and Computer Engineering
University of Michigan
Ann Arbor, Michigan 48109

Abstract

Special-purpose hardware has been proposed as a solution to several increasingly complex problems in design automation. This paper discusses the implementation of Lee-type wire routing on a class of cellular architectures called *raster pipeline subarrays* . Employed initially for cellular image processing and machine vision, machines in this class are also applicable to problems in physical DA represented on cellular grids. Detailed benchmarks are presented for routing algorithms running on an existing raster pipeline subarray machine called a *cytocomputer*[†]. The paper discusses the impact on our routing algorithms of some enhancements to the current hardware, and evaluates the merits of this class of architectures in the context of practical special-purpose hardware. We outline the goals of this work in the context of a larger project to identify the architectural and performance characteristics necessary to optimize a raster pipeline subarray machine specifically for grid-based DA applications.

1. Raster Pipeline Subarray Architectures

Several architectures have been proposed to implement maze-routing [1,2], the majority of which employ some variant of a classical cellular array: a rectangular grid of simple communicating processor/memory pairs, each manipulating some small section of the complete grid. The *raster pipeline subarray* organization represents an alternate architecture for the maze-routing task. By *subarray* we mean an array much smaller than the entire grid representing the problem; a *subarray processor* is a small processing window that can be passed across a large array. A *raster subarray processor* accepts as input and produces as output a raster-order stream of grid cells; the inclusion of buffers for a few lines of the array insures that as data passes through the processor enough of the original array is present so that each subarray eventually arrives at the subarray processor (Fig. 1). The identical format of the input and output data streams enables a series of these processors to be connected in a pipeline with each processing the output stream of its predecessor (Fig. 2). Parallelism is achieved because, after the pipeline fills, every processor is working on some subarray of the original problem. Each processor here is one stage in the pipeline, and this organization is called a *raster pipeline subarray* architecture. Benchmarks reported herein were obtained on such a machine, a Model II cytocomputer designed by the Environmental Research Institute of Michigan [3] and interfaced to a VAX 11/780 host running UNIX[‡]. This machine processes 8-bit cells through a pipeline with rate 480K-cells/sec, and is currently configured with a 256Kb buffer for intermediate array storage and two processing stages.

2. Wire Routing Experiments

Our primary motivation here is to explore the application of these architectures to DA problems, hence one natural problem to explore is maze-routing. (For application to other DA problems, such as design rule checking, see [2,4].) The wavefront expansion phase of a maze-router is a typical cellular problem where each cell in the plane of the routing grid is modified as a function of its neighbors. This phase dominates the execution time of software maze-routers, prompting considerable research into techniques to eliminate unprofitable directions of cell expansion. In software, the addition of one layer of cells to an existing wavefront (one expansion step) requires time

proportional to the number of cells already on that wavefront. This is not so in hardware. Because each cell is processed in constant time as it streams through the subarray pipeline (when streaming, cells enter and leave the pipe at the same rate), all cells can be examined, added to the wavefront, or left unchanged, with total time proportional to the size of the grid on which the problem is represented. If a few stages are required to perform a single expansion step, then a long pipeline can perform many expansion steps as the grid streams through it and incur, in comparison to a short pipeline, only the added time due to the latency of the longer pipeline.

Because it is inefficient to process a large, mostly empty grid during the initial expansion steps of any wire, but unavoidable during the final expansion steps for a long, spatially distributed wire, it is essential to control the size of the bounding rectangle of the cells being expanded. We employ a strategy of expanding in a sequence of successively larger bounding frames starting from the original source point(s). At the conclusion of the k-th expansion the k-th frame is approximately a bounding rectangle for the active wavefront; the wavefront is never allowed to hit a frame because it will be distorted, destroying any guarantee that a minimum length path will be found. These frames increase in size until we reach either the net target point, or an artificial boundary placed to constrain the extent of the net [5], or the edges of the routing plane.

The time for this expansion phase is optimized by considering the following three factors. First, we must account for the performance of the pipeline because the number of available stages, their bandwidth, and the time to reprogram them determines the basic pipeline processing time for each expansion step. Second, we must account for the expected spatial distribution of the wire because the wire's final length and bounding rectangle determine the final number of expansion steps and the largest frame that must be passed through the pipe. Finally, we must account for the communication overhead introduced by the interface from the host operating system to the hardware because each pass through the pipeline, each adjustment of a frame size, each test to see if a final target cell has been expanded, etc., is accomplished by a host command that must pass through several layers of host system software before it gets to the hardware. If, in a sequence of frames, each frame is too large, the expansion phase time is spent expanding mostly empty grids. If each frame is too close in size to the previous one, most of the time is spent in host-to-hardware I/O overhead, making each single expansion step costly and inefficient. An optimum framing strategy exists between these two extremes.

A complete router for n-point nets on a single conductor layer in a unit-cost grid is currently functional on our hardware. It accepts a wire-list and initial grid, and produces a final grid with embedded wires. Parallel operations, such as the expansion phase and grid clean-up phase, are performed in the pipeline; sequential operations, such as the back-trace of each wire segment, are performed by the host. As benchmarks we considered the test problems shown in Fig. 3. The terminals of one 2-point or 4-point net are placed on the diagonal of a 512 x 512 grid so that the Manhattan length of the final routed connection will be 2^N cells. Several iterations of the router for each of several values of N were performed, and statistics collected; a 1-stage pipe was employed. Although these are simple problems for many routers, e.g., a line-probe router, recall that it is the length of the resulting path and the size of its bounding frame that determine the execution time, not the presence or absence of clutter surrounding the path. Therefore, since an identical connection routed on a densely occupied grid will run in exactly the same time, these are appropriate benchmarks. Fig. 4 shows the results for elapsed and host CPU times. Note that both elapsed and host CPU times increase with net length: pipeline processing

¹This work was supported in part by NSF grants MCS-8009315 and MCS-8007298.

[†]Cytocomputer is a trademark of the Environmental Research Institute of Michigan.

[‡]UNIX is a trademark of Bell Laboratories.

time increases as more expansion steps in increasingly large frames are processed through the pipe, and host CPU time increases because some I/O overhead is associated with each iteration of the expansion step.

As another test we attempted to route 250 2-point nets on two separate layers of an 8" x 12" printed circuit board again using a 1-stage pipe. A simple greedy strategy tried to connect all 250 on the first layer, recorded the failures, then attempted these unroutable connections on the second layer. In all 210 nets--not predominately "right-way" connections on each layer--were placed in 680 seconds: 296 seconds to route 210 nets with total length 4401 cells, and the remaining time to determine the unroutable nets on the first and second layers. It is the time spent actually completing connections here that is of consequence: sophisticated ordering and via schemes will greatly improve the overall performance but are unrelated to the hardware's performance on individual wires; lengthy time spent on unroutable connections is characteristic of maze-routers generally.

3. Context of the Application

This work is being conducted in the context of a larger project to identify the architectural and performance characteristics necessary to optimize a raster pipeline subarray machine specifically for grid-based DA applications [2]. The primary strengths of the raster pipeline subarray architecture are its loose coupling of system components, such as processing stages and buffers, and the relative ease with which processing power is increased by adding additional stages. As a practical matter, this permits explicit tradeoffs to be made between cost and performance. (Compare, for example, with the initial cost and expandability of a conventional large, fixed array of tightly-coupled processor-memory pairs.) The above benchmarks are essentially worst-case times because they are running on a machine with a short pipe, a slow clock, and a datapath architecture not designed specifically for DA applications.

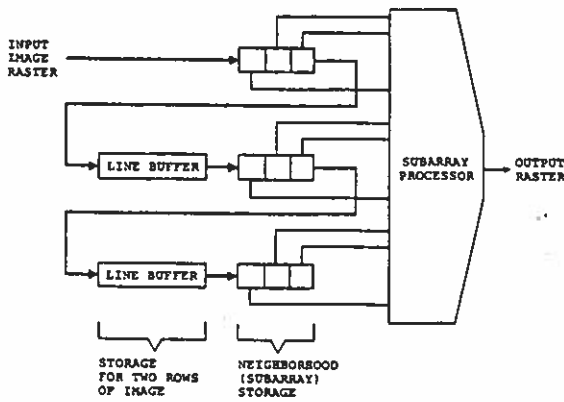


Figure 1. Single Raster Subarray Processor



Figure 2. Raster Pipeline Subarray Architecture

Although comparable hard data for proprietary systems is sparse in the literature, these worst-case benchmark times for our current low-performance hardware configuration are as far as we can tell within an order of magnitude of those for a production quality maze-router. Longer pipelines and faster stages will both decrease routing time. Although the exact time is a function of the framing strategy as well as the hardware speed, the time is roughly inversely proportional to pipeline length and rate. For example, an 8-stage pipe of stages with bandwidth 2M-cells/sec will run at least an order of magnitude faster. We are currently rerunning our benchmarks with a 2-stage pipeline, and intend soon to expand to more stages.

Maze-routers have been supplanted in many applications by faster routers, and are often relegated to those last few connections uncuttable by any other means. We are exploring an implementation that makes the power and flexibility of general maze-routers available with competitive execution times. This work provides concrete data for the analysis of the hardware tradeoffs required to complete an architectural specification for such an optimal raster pipeline subarray DA machine.

References

- [1] H. G. Adshead, "Towards VLSI complexity: The DA algorithm scaling problem: Can special hardware help?" *Proc 19th Design Automation Conf.*, pp. 339-344, June 1982.
- [2] R. A. Rutenbar, T. N. Mudge, and D. E. Atkins, *A Class of Cellular Architectures to Support Physical Design Automation*, CRL-TR-10-83, Computing Research Laboratory, University of Michigan, Feb. 1983.
- [3] R. M. Loughheed and D. L. McCubbrey, "The cytocomputer: A practical pipelined image processor," *Proc. 7th International Symp. Computer Architecture*, pp. 271-277, May 1980.
- [4] T. N. Mudge, R. A. Rutenbar, R. M. Loughheed, and D. E. Atkins, "Cellular image processing techniques for VLSI circuit layout validation and routing," *Proc. 19th Design Automation Conf.*, pp. 537-543, June 1982.
- [5] S. Akers, "Routing," in *Design Automation of Digital Systems*, vol. 1, M. Breuer, Ed., Englewood Cliffs, NJ: Prentice Hall, Chapter 6, 1972.

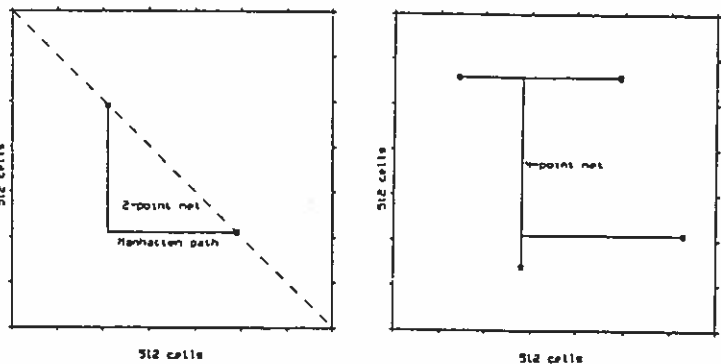


Figure 3. Benchmarks for 2-Point and 4-point Nets

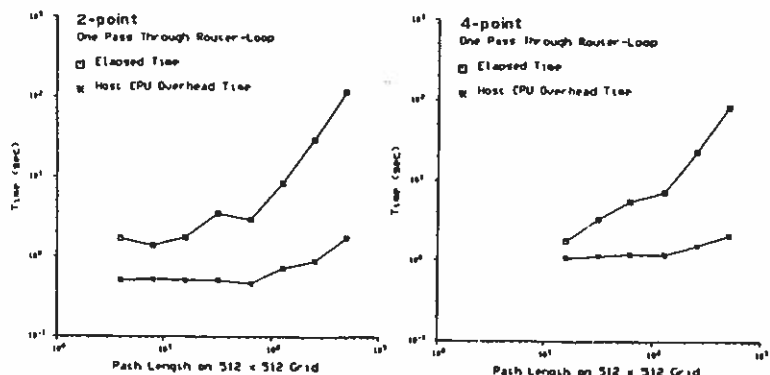


Figure 4. Timing Results for 2-Point and 4-point Net Benchmarks