EXAMINING ROUTING SOLUTIONS

S. Bapat, J. P. Cohoon, P. L. Heck, A. Ju, L. J. Randall

Department of Computer Science
University of Virginia
Charlottesville, VA 22903

Absttract

A visualization tool, Examine, is presented to aid router development. Examine has the ability to display routing operations in a variety of ways (e.g., selective display of layers and nets). In addition, Examine has rudimentary animation facilities that enable a router developer to view a sequence of routing frames or a routing frame composite.

Introduction

Developing VLSI physical design tools is a sophisticated process. Although a variety of development aids exist for coding, debugging, and managing this software, there are almost no tools that help computer-aided design developers to find improvements in performance and functionality. In particular, if a developer is to understand and evaluate the effectiveness of a new and complex heuristic or data structure, visualization tools are necessary that can adequately trace and animate a router's execution.

Viewing the ordinary output of a router or placer through the standard graphical display routines of a design system is ineffective during tool prototyping. Normal output and display routines are intended for a circuit designer's use in evaluating and modifying a particular circuit layout.

Standard algorithm visualization tools such as Brown's Balsa II system [1] are similarly inappropriate for router visualization as they are intended for pedagogical and documentation purposes and have more complicated user-interfaces.

Since our research focus is developing state-of-the-art routing and placement systems, we deem it worthwhile to develop a physical design visualization tool set. The focus of this paper is one such tool. The tool is named Examine and it aids router development by allowing the progress of routing algorithms to be quickly evaluated through a friendly graphical interface.

Examine is an X- and Motif-based Unix tool that makes extensive use of color and shading. It enables a router developer to examine interconnection output in a variety of ways:

- highlight just one interconnection or a collection of interconnections in isolation or in tandem with the other interconnections;
- zoom or pan a desired region of the routing solution;
- view interesting interconnections on all layers or on specific layers using any desired layering order;
- display a sequence of frames of a router's progress. The frames can be viewed either in isolation or as a *composite* picture. In addition, the sequence can be viewed in normal or in reverse order;
- maintain an Encapsulated Postscript history of the routing displays.

In Figure 1, a copy of a typical Examine raster display is given. The figure shows that Examine is divided into three major window panes. The top pane, the *control pane*, allows the router developer to configure the appearance of the router output in the middle pane, which is called the *workspace pane*. The bottom pane is the *message pane* and it is where status messages and warnings are displayed. In the figure, the workspace pane is displaying a prototype router's solution to Burstein and Pelavin's Difficult Switchbox example [2]. In particular, Examine's user is considering the prototype's performance on nets 3, 15, 18, and 20.

In the remainder of this paper, we discuss Examine in greater detail and suggest how it may contribute to a developer's understanding of a routing tool.

Input Specification

Examine expects as input a simple ASCII stream that describes the basics of the routing solution. The stream can come from either standard input, command line file name argument, or as the result of Examine initiating a routing command sequence. In Figure 1, the input was generated by the latter-most method — running prototype sroute through filter s-to-e which removes some extraneous debugging messages and converts the router's output to Examine's desired input format. This format is a sequence of workspace display requests. The requests have two forms — *normal* and *simple*.

The normal form begins typically with one of the following lower-case letters: *t*, *v*, *w*, or *o*. The letters indicate respectively terminal, via, wire, and obstacle. Following the letter is an object identification tag which is normally a net identification number. Following the tag is either a single location coordinate or a pair of location coordinates. It is a single coordinate for a terminal or via display request,

0-8186-2610-0/92 $3.00 © 1992 IEEE

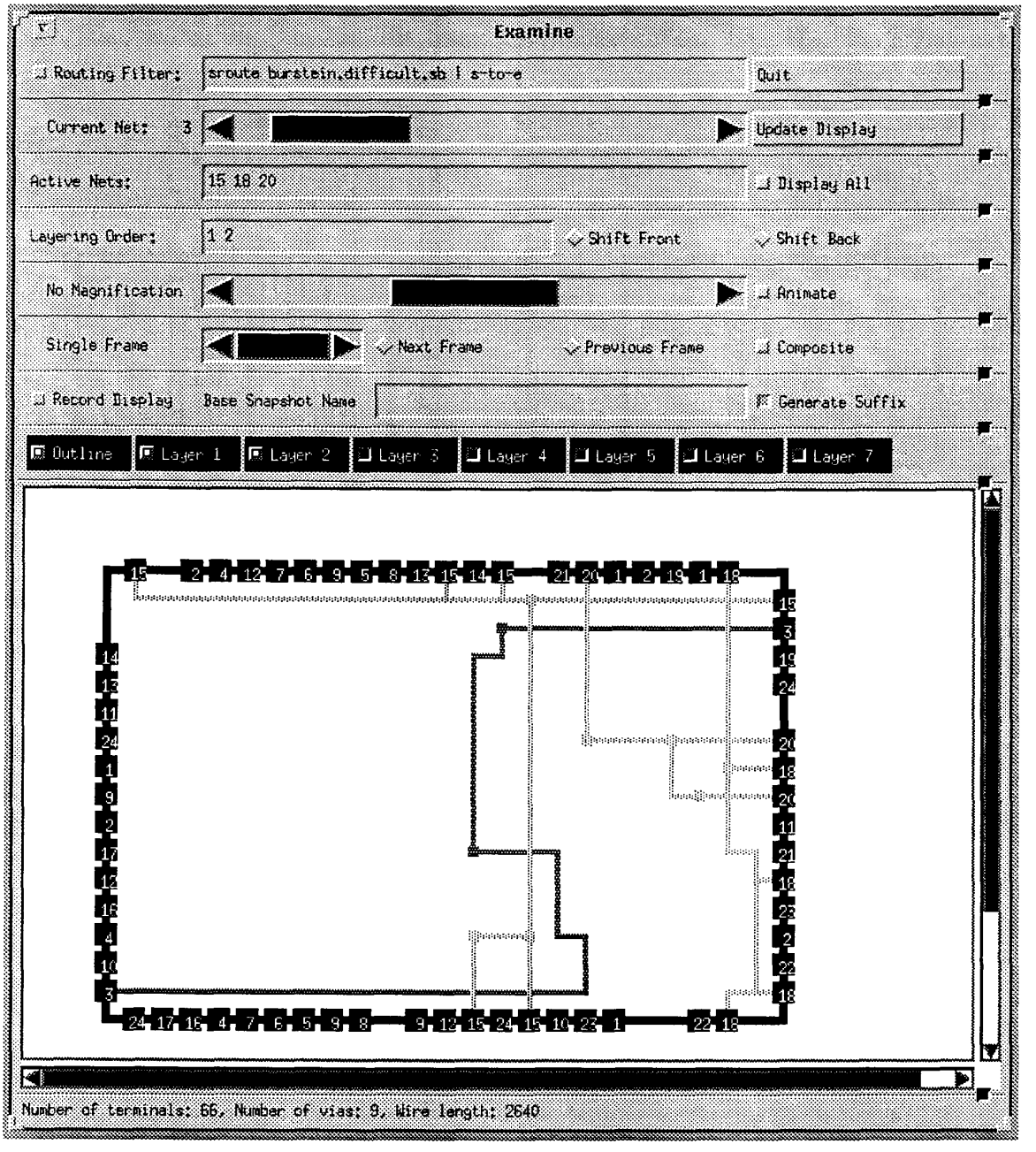

Figure 1. — Examine displaying selected interconnections of sroute for the Difficult Switchbox.

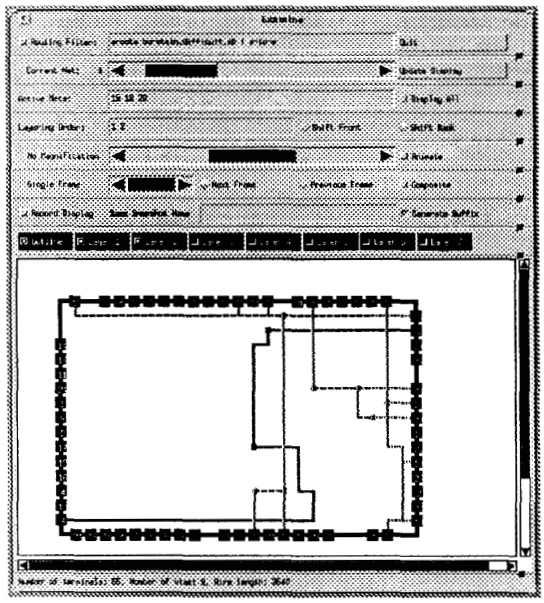

Figure 2. — Default Examine display for sroute's Difficult Switchbox solution.

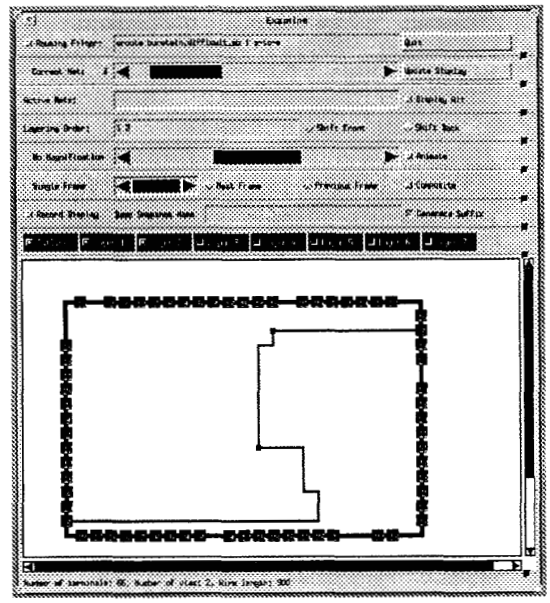

Figure 3. — Viewing Net 3 of the Difficult Switchbox solution.

where the coordinate specifies the center of the terminal or via. For the wire and obstacle display requests, it is a pair of coordinates that specifies the starting and ending locations of the object. The last part of a display request specifies the layer(s) on which the object is to appear. For all but via requests, this a single layer identification number. For a via request, the starting and ending layers are both indicated.

In the simple form, the input request does not specify exact x- and y-coordinates for a display request. Rather, a location is specified through track and column numbers. A simple request sequence is initiated with either the command *s* or *S* and is terminated with a second *s* or *S*.

For the above normal and simple commands, Examine uses default values for the width of the objects. If a developer prefers to specify these values, there are uppercase equivalents to the display requests that allow size specification. The obstacle and wire display requests take a width specification as their final argument. The via and terminal display requests take width and length specifications as their final arguments.

There are two other display request commands. One is the input break-point request which is denoted by either *b* or *B*. This request both indicates a temporary pause in the processing of the display requests and serves to group a series of display requests into a single routing *frame*.

The remaining request type is the deletion request. A deletion request is surrounded by a *d* or *D*. In between these markers, the developer may specify vias, terminals, and (partial) wire segments that should no longer be displayed. The removal is applicable to the current and successive frames. This feature is intended for routers with rip-up and reroute capabilities.

Basic Operation

The standard method of operation is to pipe the output of a router to an appropriate filter that produces the desired Examine routing display requests. This is the case for Figure 2, where prototype sroute's complete solution to the Difficult Switchbox is being displayed.

As depicted in Figure 2, the initial display of a solution shows all interconnections, terminals, vias, and obstacles. In addition, the display shows all layers in standard order.

If a router developer prefers to view a single net then the *Display All* option is un-set and the *Current Net* scrollbar is slid to the appropriate point and the *Update Display* button is selected. If the scrollbar request is set to net 3, the result would be the workspace pane depicted in Figure 3. Observe that a stipple pattern is used to highlight the current net.

If a router developer instead prefers to view the interaction of the current net with respect to some other nets, the *Active Net* widget may be accessed and the other nets of interest may be entered. If the interesting active nets were 15, 18, and 20, an *Update Display* selection would produce the display of Figure 1. Observe that the c-urrent net and active nets in Figure 1 have different stipple patterns. The

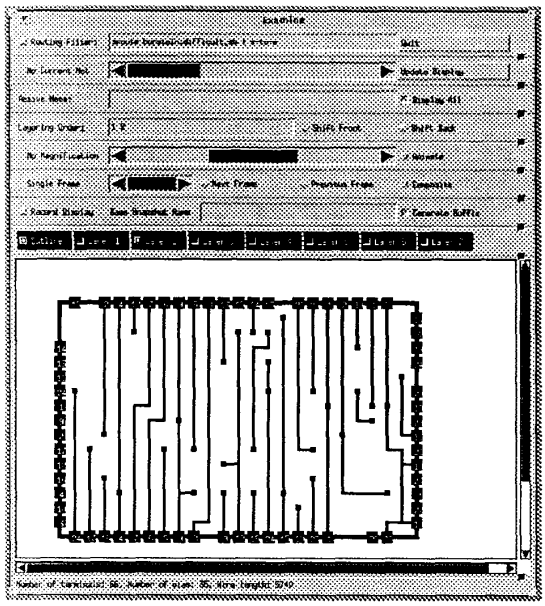

Figure 4. — Viewing layer 2 of the Difficult Switchbox solution.

router developer also has the option by setting *Display All* to view the remaining nets. These remaining nets would be displayed using the standard stipple pattern.

If the router developer prefers to see a selected layer(s), the appropriate *Layer* button(s) can be de-selected. This is demonstrated in Figure 4 where the two-layer routing solution of Figure 2 is displayed with layer 1 turned off. Although the control panel displayed in Figure 2 implies a seven layer limit, this is just the default.

ill see its greatest use in the prototyping of multi-layer routers, particular attention was paid during its development to ensure that color and shading layering cues were present, as well as other layering display mechanisms. For example, in addition to its layer selection capabilities, Examine has the capability to display the layers in an arbitrary order. This is performed by entering the desired sequence in the *Layering Order* widget. Another layering option allows the current layer ordering to be permuted either forward or backward. This is done respectively through *Shift Front* or *Shift Back* button selection.

If the router developer instead prefers to examine a different solution or a different router's performance, it is not necessary to exit Examine. Rather, the developer may issue a Unix command pipe to generate the new input to Examine. For example, to examine a different input file, the Unix command cat can be used. A different example is depicted in Figure 4, there the *Routing Filter* widget specifies a routing command pipe in which sroute solves the switchbox instance burstein.difficult and pipes its output through the filter s-to-e.

Cinema Veritae

As stated above, we expect that Examine will receive its greatest use in visualizing the output of multi-layer routers. In particular, we believe developers will be interested in examining a series of routing actions either individually or in composite form. To aid this interest, Examine allows the router developer to group the routing requests — each such request group is called a *frame*.

Since color cues are vital for understanding a multi-layer router's solution, such solutions are inappropriate for depiction in this paper. Therefore, we use a simple Lee-Moore maze router [5] as our example router for discussing animation. The routing displays produced in the next four figures, show the progress of a maze router running a variant of Soukup's heuristic maze routing improvements [6] on a single layer problem instance. For the problem instance in question, the developer wishes to determine which routing locations are considered by the router during its routing exploration.

To allow an Examine user to specify a routing "background" that is common to all frames, Examine follows the convention that all routing display requests that occur before the first break-point are displayed in every routing frame or composite. These requests are considered Frame 0. A sample Frame 0 for the maze routing solution under consideration is given in Figure 5.

A developer may walk through the frames by first selecting the *Next Frame* button and then pushing the *Update Display* button as desired. Similarly, if a developer prefers to walk backward through the frames, the *Previous Frame* may be selected. If a developer prefers to jump forward or backward to a particular frame, the *Frame* scrollbar can be slid to the appropriate point.

In Figure 6, a composite frame is depicted of Frames 0-6 of our maze routing example. In this and in subsequent figures, the small squares represent grid cells the maze router has explored. The squares are drawn using via requests.

Figure 7 shows individual Frame 10 of our routing trace. In the figure, the maze router is starting its two-way expansion around the obstacle that is in front of the target terminal. Finally, Figure 8 shows composite Frame 16 which corresponds to the maze router discovering simultaneously two different shortest paths to the target terminal.

If a router developer prefers to view a "film" version of a frame sequence, the *Animate* button is selected. The next update of the display will then iteratively display each frame up through the current frame. Examine makes uses of visible polygon determination algorithms for a smooth

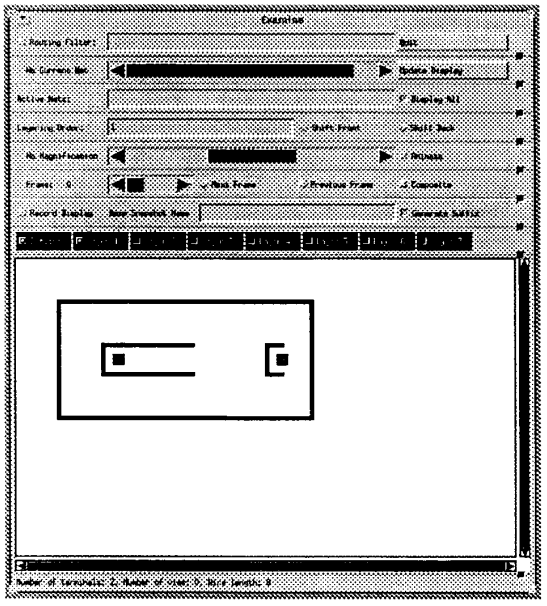

Figure 5. — Viewing the routing background (Frame 0) of a maze routing solution.

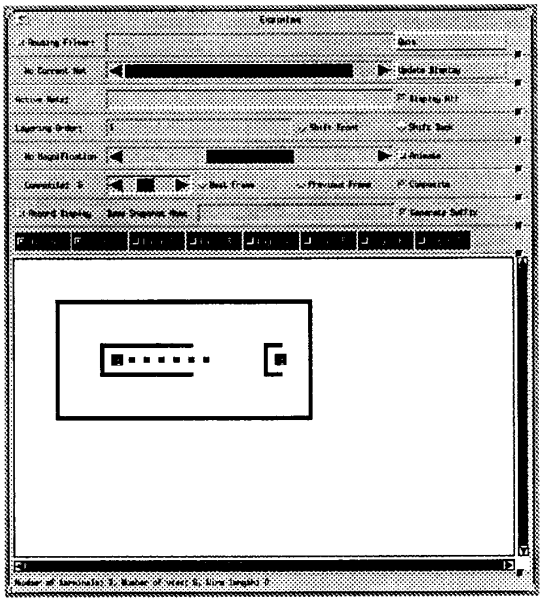

Figure 6. — Viewing composite frame 6 of the maze routing solution.

updating of the display [4].

Other Features

While the workspace window is sufficient for the individual routing of a relatively simple channel or switchbox, and while the workspace scrollbars enable a developer to examine any portion of a solution, a developer also often needs the ability either to view the "big picture" or and an extreme "close-up". Thus, Examine also has the ability to pan or zoom the routing solution. Figure 9 depicts a channel routing solution for a variation of Deutsch's Difficult Channel example [3], where the magnification factor is $1/6^{th}$ of normal magnification.

enting a router's solution is also important, Examine provides a developer with the ability to maintain a history of the routing displays with Encapsulated Postscript. This is performed by setting the *Record Display* button and by entering a file name in *Base Snapshot Name* widget. The *Generate Suffix* widget is useful if the developer wishes to save a series of routing displays. If it is set, a copy of each routing display is individually stored. All of the copies will have the same base name and the suffices will be numbered in the order that they are generated. Figure 10 shows the displays the output associated with the capture of the routing in Figure 2.

Future Enhancements

We are also considering different composition features. For

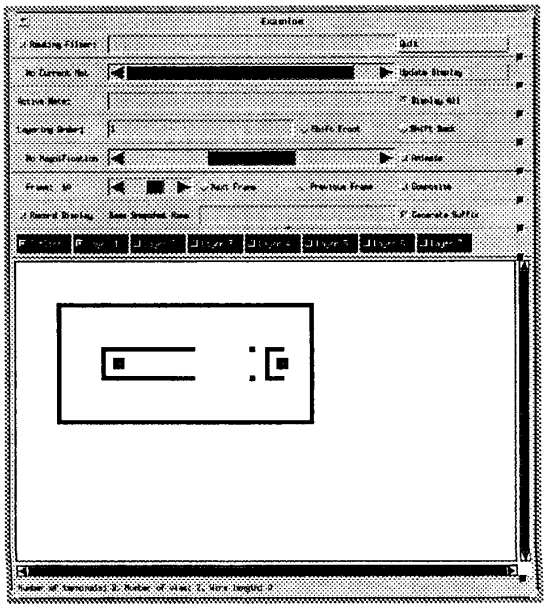

Figure 7. — Viewing individual frame 10 of the maze routing solution.

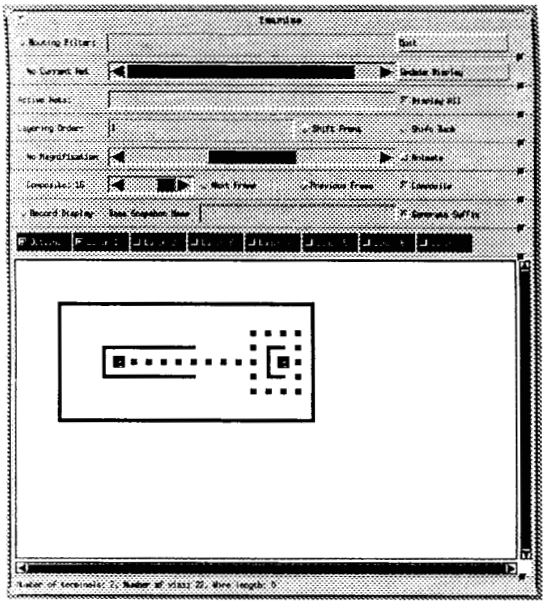

Figure 8. — Viewing composite frame 16 of the maze routing solution (target cell is reached).

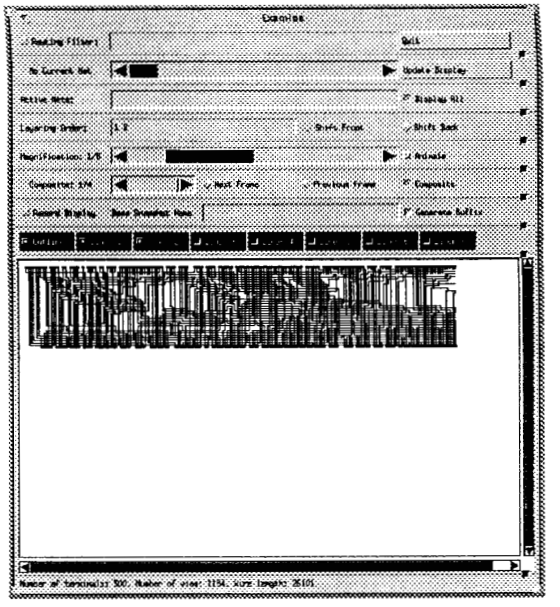

Figure 9. — Panning a channel routing solution.

example, we are adding the ability to view a composite image that is composed of an arbitrary, user-specified frame subset. We are also investigating whether placement visualization aids are appropriate in Examine or whether a different visualization tool is needed.

Summary

Examine is a visualization tool that enables router developers to trace and animate a router's execution in a variety of ways. For example, Examine has the ability to selectively display layers and nets. It also has rudimentary animation facilities that enable a developer to view a sequence of routing frames or a routing frame composite.

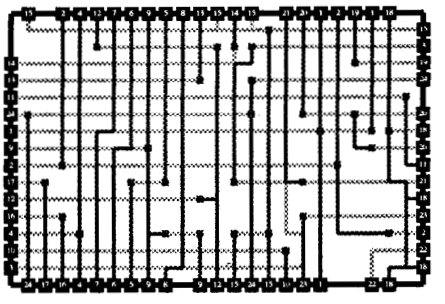

Figure 10. Encapsulated postscript version of a outing display.

Acknowledgements

The authors' work have been supported in part through National Science Foundation grants MIP-9107717 and CDA-8922545, Virginia CIT award 5-30971, and the SIGDA Design Automation fellowship. Their support is greatly appreciated.

References

[1] Brown, M. H., *Algorithm Animation*, MIT Press, Cambridge, MA, 1988.

[2] Burstein, M., and R. Pelavin, Hierarchical Wire Routing, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems cad-2(4)*, October 1983, pp. 223-234.

[3] Deutsch, D. N., A 'Dogleg' Channel Router, *13th ACM/IEEE Design Automation Conference Proceedings*, San Francisco, CA, 1976, pp. 425-433.

[4] Foley, J.D., A. van Dam, S. K. Feiner, J. F. Hughes, *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading, MA,1990.

[5] Lee, C. Y., An Algorithm for Path Connections and Its Application, *IRE Transactions on Electronic Computers ec-10(3)*, September 1961, pp. 346-365.

[6] Soukup, J., Fast Maze Router, *15th ACM/IEEE Design Automation Conference*, Las Vegas, NV, 1978, pp. 100-102