PEARL: A KNOWLEDGE BASED EXPERT ASSISTED CAD TOOL

Edward J. DeJesus and James P. Callan

DIGITAL EQUIPMENT CORPORATION
CAD Systems Engineering
Andover, Massachusetts 01810-1098

ABSTRACT

The use of artificial intelligence (AI) expert systems technology has demonstrated its advantages with successful applications in several key areas. (e.g. R1/XCON [1] for its searching techniques, YES/MVS for control [2], and Drilling Advisor [3] for advice). This paper discusses how control, search, advice, and domain specific knowledge were integrated with a computer aided design (CAD) architecture to develop an expert system. The combination produced a tool that provides intelligent assistance to printed wiring board (PWB) layout designers.

This CAD tool focuses entirely on the layout requirements of power supply circuits for PWBs. The system is called PEARL for Power-supply Expert Assisted Rule-based Layout. PEARL acts as an advisor to the layout designer, providing expert assistance with the placement of components on PWBs. It is presently being enhanced to provide etch routing assistance as well.

INTRODUCTION

Most AI work on CAD systems (e.g. [4, 5, 6, 7]), has been concerned primarily with LSI and VLSI circuits, while most of the physical CAD layout tools for the PWB domain were developed by conventional methods and have only addressed the needs of digital circuits. Until now little has been done to resolve the PWB layout problems for analog circuits.

Power supply layout is a subset of the larger analog PWB design problem. These layouts are driven by many different rules and design constraints. Since these layouts are heavily constrained, the requirements differ greatly from those for which conventional programming techniques and algorithms are appropriate.

This paper describes an expert CAD tool for power supply layout called PEARL, the factors that influenced its design, its problem solving approach, flexible functionality, and the plans for its future.

The Problem Domain

Power supply layout is very different from the layout of digital circuits. Digital Equipment Corporation (DEC) has traditionally developed CAD tools specifically for digital circuits comprised of dual inline packages (DIPS) and gate arrays. Dips and gate arrays have consistent rectangular shapes, and are typically placed in rows and columns. Their interconnections are usually routed with consistent etch widths and spacings (e.g. 8 mil etch with 8 mil spacings). DEC and the rest of the CAD industry have developed very successful algorithms for these digital circuit layout problems.

However, power supplies are predominantly comprised of discrete components (e.g. transformers, capacitors, diodes, resistors, transistors, etc) in analog circuits. Discretes tend to be irregularly shaped and sized, making the component placement process more difficult. Component locations are further constrained by a number of factors, including:

- Certain groups of components, called functional circuit groups, must be placed near one another.
- The primary side of the power supply circuit must be kept away from the secondary side of the circuit.
- Industry safety standards (e.g. UL, VDE) and manufacturing requirements must be adhered to.

Power supply routing requires varying conductor widths and spacing for the differing voltages and currents. Etch widths and paths must also be chosen to prevent problems with inductance and signal noise from inadequate grounding separation. It is also common to see conductors run both horizontally and vertically on the same layer, something that is generally not allowed on digital logic PWBs. These constraints make power supply routing difficult for our present topological routing tools [8].

It is difficult to build and support an automated layout tool that satisfies these and other constraints using a traditional algorithmic approach. Before PEARL, power supply designs at Digital were laid out manually or on CAD systems which had no logic information, automatic tools, or analysis capabilities. Layout designers manually placed and routed prototype boards, which were then built and tested to see if they worked. Many circuit boards didn't work or failed to meet requirements, resulting in additional passes

through layout with the engineer suggesting new layout constraints. The average power supply is laid out several times before a reliable and properly functioning circuit is achieved.

A Knowledge-Based Solution

There are several reasons why a knowledge-based approach might succeed where algorithmic approaches have not. Knowledge-based systems are often applied to situations where the problem is heavily constrained and the knowledge about how to solve it is scattered or incomplete. Power supply layout fits this description nicely. It is heavily constrained. The knowledge about what constitutes a good layout is distributed between the engineer and the layout designer. The fact that it has been a multi-pass layout process indicates that there are many unknown factors affecting it.

Another advantage of knowledge-based systems is that the knowledge-base is expected to grow throughout the life of the system. In many cases knowledge-based systems have eventually led to a better understanding of the problem domain by acting as a repository for information, strategies and rules of thumb which might otherwise be lost or forgotten (e.g. [9]).

The Development Strategy

Although Digital has departments specializing in AI technologies that have developed several successful expert systems (e.g. [1, 10, 11]), prior to PEARL the CAD Systems Engineering department had none. The CAD department also had no experience developing power supply layout tools. This combination of factors made it a high risk project. In order to reduce the complexity of the problem, several major decisions were made about the overall design of PEARL.

First, although placement and routing appear to influence each other, particularly on power supplies, we initially chose to treat them separately. We focused on the needs of discrete mosaic placements that are highly influenced by constraints. Analog routing, and the interaction between placement and routing, were saved for later. This would provide a smooth learning curve by allowing us to develop the rule-based system in phases. PEARL is currently only a placement tool, although work on routing capabilities has begun.

Second, trying to solve the problem with one fully-automatic tool would be undesirable and limited, because most automatic approaches run for a long time, rarely solve all the problems, and do not allow the input of experienced layout designers. Therefore PEARL is designed to assist the user. It is intended to primarily operate in a highly interactive semi-automatic mode, however it can also be used in a completely manual or fully automatic mode. These modes are described in detail later.

Third, it became a goal of the project to deliver some sort of useful tool quickly in order to determine the feasibility of the approach and to elicit user feedback during further development. The advisor or semi-automatic mode became the basis of the tool, creating an excellent environment for observing the user. It also enabled us to acquire our own knowledge about layout problems, and to formulate rules and strategies for intelligent solutions to them.

Finally, adding to the above considerations was an essential need to integrate PEARL within our existing VAX Layout System (VLS). VLS is DEC's proprietary CAD system for PWB design. It is an environment and a set of application programs which share a common data model, user interface, and set of utilities. The decision to make PEARL a VLS application was significant, because it is important to produce a tool that operates in an environment that the users are familiar with. In addition, VLS offers many other functions that are vital to the layout process. One such application is EDIT ETCH, which provides manual routing capabilities that can be used to complete the layouts until PEARL's routing functionality becomes available.

ARCHITECTURE

PEARL, like VLS and many CAD layout systems, is a graphics-based interactive tool. Consequently the architecture was designed with an emphasis on speed. VLS is largely written in BLISS. Since PEARL was designed to be a VLS application program, the implementation language(s) needed to be efficient and able to communicate effectively with the VLS environment. We chose OPS5 and BLISS.

BLISS is a procedural language that is similar to the C programming language. It is extremely fast and efficient for traditional algorithmic programming. DEC's version of OPS5, which is itself written in BLISS, is a language designed for implementing forward chaining production systems. Several successful expert systems have been written in OPS5 (e.g. [1, 2, 11]). Unlike most programming languages, OPS5 has no underlying notion of sequential execution. Activation of its production rules is dependent upon the contents of OPS5 working memory [12, 13].

PEARL is structured in two layers, with the top layer providing a flexible control mechanism and the bottom layer providing efficient algorithms and primitives. The top layer is comprised of OPS5 rules that drive the user interface, the placement control strategies, and an explanation facility. The bottom layer is made up of many BLISS subroutines that implement placement algorithms, operate the user interface, and access the data structures. The interface to VLS is also contained in BLISS subroutines, providing access to VLS data structures, utilities, and graphics.

Control and Search

The OPS5 rules drive the user interface and placement control strategies, enabling the rule-based system to always know what mode and state the user and layout are in. When the user presses a button or enters a command string, an

OPS5 working memory element is created and matched against production rules. There are essentially three types of production rules:

- o user interface rules -
  to carry out specific functions chosen by the user.

- o control rules -
  heuristics to handle special layout sequences based upon power supply constraints and layout strategies.

- o component searching rules -
  to identify one or more components that satisfy a set of constraints.

PEARL uses a number of different placement strategies to partition the search space. Heuristic rules determine the order in which placement strategies are considered. They insure that the placement algorithms give priority to components within functional circuits and critical groups, and to components that only qualify for placement under very specific conditions (see figure 1).

```
;+
;   This rule processes the PLACEMENT key on the MAIN keypad.  Basically
;   it sets up goals to initialize Working Memory with placement data,
;   and to control the placement cycle.
;-
(P  keypad_main:1:placement_key
    {<button>
         (user_input     ^user_keypad     keypad_main
                         ^user_event      button
                         ^user_data       placement_key)}
        -->
         (REMOVE <button>)
         (MAKE   goal    ^goal_name                          placement_cycle
                         ^goal_control_flag                  first
                         ^goal_subgoal_list
                                 find_unplaced_comps         not_done
                                 make_cand_counters          not_done
                                 get_candidates              not_done
                                 call_peaplif_show_list      not_done
                                 wait_for_input              not_done
                                 call_peaplif_place_cand     not_done
                                 do_bookkeeping              not_done)
         (MAKE   goal    ^goal_name                          placement_init
                         ^goal_control_flag                  first
                         ^goal_subgoal_list
                                 extract                     not_done
                                 constraints                 not_done
                                 preplaced                   not_done
                                 initialize_peaplif          not_done )
         (CALL   peacmd_keypad_placement)
)

;+
;   The goal created establishes what kind of candidates to look for, and
;   in what order.  Notice that the subgoals all correspond to placement
;   reason codes used later by the explanation facility.  When the
;   candidates have been generated, the next rule in the placement cycle
;   fires. It is important that the control_flag be modified BEFORE the
;   new goal is created, otherwise the new goal will never gain control.
;-
(P  placement_cycle:5.1:get_candidates_no_goal
     {<cand_goal>
         (goal           ^goal_curr_subgoal_name             get_candidates
                         ^goal_curr_subgoal_status           not_done
                         ^goal_control_flag                  active)
         -(goal          ^goal_name                          get_candidates)}
         -->
         (MODIFY <cand_goal> ^goal_control_flag              next)
         (MAKE   goal    ^goal_name                          get_candidates
                         ^goal_control_flag                  first
                         ^goal_subgoal_list
                                 place_preplaced             not_done
                                 place_input_connectors      not_done
                                 place_edge_connectors       not_done
                                 place_keep_near             not_done
                                 place_emi_filter            not_done
                                 place_prim_rectifier        not_done
                                 place_input_filter          not_done
                                 place_inverter              not_done
                                 place_other_primary         not_done
                                 place_main_transformer      not_done
                                 place_other_crit_nets       not_done
                                 place_other_secondary       not_done
                                 place_constructive          not_done )
)
```

Figure 1. Sampling of PEARL layout control rules.

Within a particular placement strategy, component selection rules search for components that satisfy the appropriate constraints and placement criteria (see figure 2). The component selection rules for a given placement strategy can only be considered when that placement strategy is active. These rules are effective because they focus on a restricted search space.

The advantage of the rule-based system is apparent in the way constraints from the knowledge base are used to control and prune the search for placement candidates. The control rules are maintained by meta-rules that monitor the status fields of each placement strategy. Control is transferred from the OPS5 inference engine to groups of rules with goals and subgoals that guide the processing through a series of placement strategies. The placement strategies and the sequencing rules could easily be altered to handle different technologies for other PWB layout problems and the BLISS primitives and algorithms would still be effective.

Placement Algorithms

When the rule-based system finds several components that it considers equally good placement candidates, it relies on a component selection algorithm implemented in BLISS to choose among them. The component selection criteria is one which chooses components that are most heavily connected to previously placed components, with a negative weight included for interconnections attached to unplaced components. This empirical selection method is similar to many "least commitment" approaches, and is intended to avoid the need for backtracking. PEARL has no backtracking designed into its fully automatic mode, but a later discussion about it's flexible semi-automatic mode does address the problem.

```
;+
;   This rule fires if a placed component  is designated as the "A"
;   component in a component pair, and the "B" component is unplaced.
;   If it fires, the "B" component will be made a placement candidate.
;   It will also attempt to find a location for the "B" component that
;   keeps it near the "A" component.
;-
(P  keep_near_groups:2.1:comp_a_placed_find_b
    {<kng_goal>
         (goal           ^goal_curr_subgoal_name             place_keep_near
                         ^goal_curr_subgoal_status           not_done
                         ^goal_control_flag                  active)}
         (component      ^wds_id                             <placed_id>
                         ^comp_pstat                         << placed fixed >>
                         ^keep_near_grp_num                  {<> nil})
         (comp_pair      ^comp_a_id                          <placed_id>
                         ^comp_b_id                          <unplaced_id>
                         ^comp_a_refdes                      <placed_ref_des>
                         ^comp_b_refdes                      <unplaced_ref_des>)
         (component      ^wds_id                             <unplaced_id>
                         ^comp_pstat                         unplaced )
    {<cnt>
         (counter        ^counter_name                       cand_counters
                         ^counter_value                      <value>
                         ^counter_max                        {<max> > <value>})}
        -->
         (REMOVE <kng_goal>)
         (MAKE   cand            ^cand_assa_id               <unplaced_id>
                                 ^cand_choice                (COMPUTE <value> + 1)
                                 ^cand_reason                place_keep_near
                                 ^cand_reason_arg            <placed_ref_des>)
         (MODIFY <cnt>           ^counter_value              (COMPUTE <value> + 1))
         (CALL   peaplif_build_list (COMPUTE <value> + 1) <unplaced_id>)
         (CALL   pealocand_pair <placed_id> <unplaced_id>)
         (CALL   pearlsubs_vls_info_message
                         |Engineering Constraint:  Keep| <unplaced_ref_des>
                         |near| <placed_ref_des>)
)
```

Figure 2. A sample component selection rule.

Once a component has been chosen for placement, geometric primitives determine where to locate it. The rule-based system may specify that the component must be kept near another component or group of components, but it is up to a placement algorithm to figure out the topology. These algorithms, many of which carry their own constraints in the form of weighting factors, are intended to be generic for solving many classes of PWB layout problems. Locations are based upon constraints affecting the component, orientation and alignment for polarization and machine insertion, and minimizing connection length. Typical algorithms, such as nearest neighbor using manhattan and euclidean metrics, are common throughout PEARL and VLS.

The Knowledge Base

Knowing that OPS5 would run much faster if the number of working memory elements was kept to a minimum [13], we limited that area to only relevant or temporary knowledge. The global knowledge base was designed into the existing VLS data base, so that the constraints information is always carried with the design.

PEARL's knowledge base consists of constraints and critical layout information specific to power-supply designs. It is captured interactively in a forms mode, which consists of predefined screens with fill-in-the blank fields (see figure 3). This mode provides a method of recording the engineer's knowledge of these details and constraints, that today are generally given to designers verbally or on a marked up schematic. Now the knowledge can be entered by the layout designers or directly by the power supply engineers. This bridges an important knowledge gap between the power supply engineer and the layout designer. PEARL does not require this data to do a placement, but it will produce much better results if the information is available.

```
PEARL 4.1(1.53)          CONSTRAINT MODE
Page 2 of 9
                          KEEP NEAR GROUPS

Group 1:   Q1     T3
Group 2:   Q1     R4
Group 3:   T3     C7      R5      R12
Group 4:   D5     C13     R10
Group 5:   D6     C14     R9
Group 6:   D7     C15     R8
Group 7:   J1     VR1
Group 8:   J2     C23     C24
Group 9:   J2     C18     C20     L1
Group 10:  C8     R2
Group 11:  C9     D2
Group 12:  C10    D3
Group 13:  R6     R7      D8      D9      C12
Group 14:

 Enter string name:
```

```
PEARL 4.1(1.53)          CONSTRAINT MODE
Page 3 of 9
                      POWER TRAIN (PRIMARY SIDE)

Input Connectors:  J1

Emi Filter:        C8      C9

Primary Rectifer:  D2

Input Filter:      VR1     R1      D1      C1      C3
                   C4      C2

Inverter:          T2      C5      C6      C7

Remaining Primary
Side Components:   D8      D9      C10     C11     D3
                   D4      R2

Main Transformer:  T3

 Enter string name:
```

Figure 3. Two pages from Constraints mode.

PLACEMENT USER INTERFACE

The placement tool can be used interchangeably amongst three modes: manual, semi-automatic, and fully-automatic. All three modes operate with the same user keypad (see figure 4).

The manual mode is equivalent to what layout designers have today. This is provided largely to give the designer the ability to override PEARL's decisions at any time. Keys are available for selecting, moving, rotating and placing components (see figure 4).

The semi-automatic mode is PEARL's default mode and operates as an assistant to the designer. Once the Placement mode is entered PEARL offers the user up to four candidate components with up to four possible locations per candidate (see figure 5). Designers can examine the components in their various locations using the Next Candidate and Next Location keys. When a

SELECT	BY AREA	UNSELECT ALL	UNSELECT
PLACE	ROTATE	NEXT LOCATION	NEXT CANDI-DATA
MOVE HORIZ ON/OFF	MOVE VERT ON/OFF	BEST CAND & LOCATION	EXPLAIN
START/ STOP	UNPLACE	OPTIONS/ CONS-TRAINTS	HELP
WINDOW		EXIT	

Figure 4. Functions assigned to placement keypad.

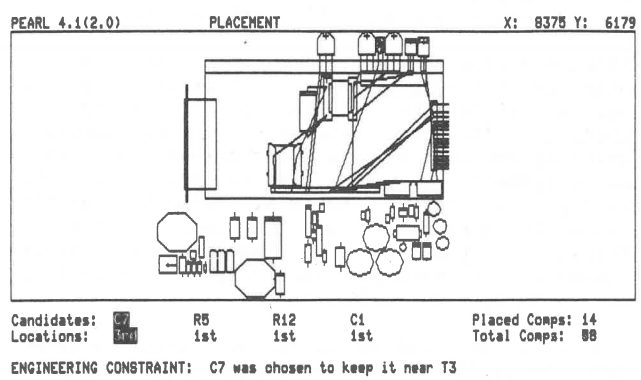

Figure 5. Graphics display in Placement mode showing four candidates, with C7 as the 1st candidate in its 3rd location. Note the explanation of why C7 was chosen.

candidate is offered in a location, its interconnections are dynamically reconnected to the nearest placed components in the network. This dynamic reconnecting simplifies the picture and illustrates the reason for choosing the different locations.

As each candidate is offered, PEARL displays a reason for its choice. Messages such as "Engineering Constraint: Keep C5 near R4", or "C19 was chosen as part of the secondary side of the power train" are generated by PEARL's explanation facility (see figure 5). The explanation facility will be described later.

The user has the choice of accepting PEARL's recommendation, by marking the candidate as placed via the place key, or creating a choice of their own in manual mode.

Keys are provided to place or unplace any candidate (PEARL's or their own). Once place or unplace steps are taken, the semi-automatic placement immediately continues by offering new candidates and locations.

A Start/Stop key allows the user to change back and forth from the semi-automatic mode to a fully-automatic mode at any time. Fully-automatic mode places the best candidate in its best location, without requesting approval from the user after each choice. As each component is placed, the layout is constantly updated and progress is displayed on the screen until all parts are placed or the user stops it. Stopping it returns PEARL to the semi-automatic mode, where the designer may examine a new list of choices or unplace some of PEARL's previously placed components.

A Flexible Tool

PEARL's interactive nature enables the user to unplace or undo any portion of the layout that may be going astray. This capability eliminates the need for the fully-automatic tool to have an elaborate backtracking scheme, or some iterative algorithm to detect and resolve problems. It also enables designers to include their own creative and experienced contributions in the layout, making them more comfortable with the system.

PEARL provides the flexibility for the user to override its suggestions, yet it still safeguards against violations of constraints. A dynamic checking facility produces warning messages whenever the user moves a constrained component. If component C5 were Selected and Moved, the message "Engineering Constraint: Keep C5 near R7" would be displayed. If the user Selected and Rotated a polarized or machine insertable component, a message like "D5 pin one direction should be up (^)" would appear.

The manual and semi-automatic modes were designed into the system to also serve the developers as an observation lab. It enables us to understand why the users reject and override the choices offered by PEARL, and to try and formulate more rules from the designers' actions.

The entire placement process is always restartable. In other words, users can stop, exit PEARL or VLS at any time, and return later to pick up where they left off. This feature is an inherent result of the interactive approach, requiring constant updating of data structures, and the fact that the knowledge base is saved with other design data. Restartability exemplifies the flexibility of the placement tool.

EXPLANATION FACILITY

One aspect of PEARL that is unique among CAD tools is that it can explain its reasoning. It has an explanation facility loosely modeled after the one developed for XSEL [14]. The explanation facility provides information in two ways. The user can ask for an explanation by pressing the Explain key and typing a component reference designator after the prompt. PEARL will also generate explanations automatically whenever it suggests a component for placement (note message in figure 5). In either case, an explanation consists of a one line reason why the component was placed or suggested for placement. At this time PEARL generates explanations for the following layout situations: functional groups, keep near groups, interconnections to other components, manually placed, preplaced, and unplaced.

PEARL also provides a hierarchical level of explanation about placed components or candidates for placement. For example, if the user inquires about a component that is a member of the input filter functional group, on asking for MORE (by typing the word MORE to the prompt) a list of other components in the input filter would be provided. The user could pursue the chain of information by asking for an explanation about other components in the input filter, or keep asking for more information about the original component until told that no more information is available.

The explanation facility increases the users confidence in PEARL's suggestions, and also serves as a very useful debugging tool for developers. It verifies that the knowledge base is accurate and being interpreted correctly. It can also be used to find out what influenced decisions, after the fact. The ability to go back and find out why something was done has always been desired in our CAD systems, but until now it remained unfulfilled.

FUTURE DIRECTION

PEARL was released as a placement tool to DEC's design community during the summer of 1985. We expect that the placement knowledge-base will be enhanced significantly during its first year in the field. We hope that it ultimately leads to a better understanding of the factors affecting power supply placement so that fewer prototypes are required.

Work has also begun to add routing facilities to PEARL. We envision that the style of interaction in the Routing Mode will be quite similar to the interactive approach in the Placement Mode.

PEARL is intended to be a specialized tool for power supply placement. However, we believe it may ultimately be used to place other types of PWBs. Many of the rules and placement primitives are very generic, and could easily accommodate placement constraints for digital circuits (e.g. keep near groups, clustering by network, etc).

Eventually we would like to expand PEARL with more dynamic checking and signal integrity knowledge. Currently a designer uses physical layout tools, and then uses analysis tools to determine whether the board will function properly. We would like to see this type of analysis moved into PEARL, so that problems are identified, and perhaps resolved, as soon as they are detected. This would involve adding more technology-specific rules to the system.

CONCLUDING REMARKS

Our original goal was to provide a CAD tool for power supply layout within the existing CAD system, VLS. PEARL satisfies that goal by providing excellent power supply placement capabilities now and the promise of routing capabilities with a similar approach in the future.

PEARL, through low level control, uses AI "expert system" technology to develop an intelligent layout assistant. A similar low level control scheme is also described by DeJong [15]. PEARL's knowledge base and flexible interactive approach offers expert advice and explains its reasoning, allows users to explore alternatives, warns of potential violations, and remembers forgotten details. This approach is not necessarily a typical expert system, but it is a knowledge-based CAD system that offers expert placement choices and produces intelligent results.

PEARL has also proved to be a very valuable learning experience for Digital's CAD Systems Engineering department. It has demonstrated that a knowledge-based system can be successfully applied to a previously intractable CAD problem. It has also demonstrated that a knowledge-based system can be built within the VLS environment. These are particularly significant results because they open up another programming methodology to CAD software developers.

ACKNOWLEDGEMENTS

We want to extend our thanks to several people who helped us with PEARL. Professor Jon Leitch of Medicine Hat University spent most of his sabbatical working on the placement primitives and manual override functions. Dave Concordia and Dave Wall, undergraduates at Worcester Polytechnic Institute, built the explanation facility. Design engineers Anil Ohri and Jim Gregorich, and layout designer Jim McDonough, were the experts in the power supply domain. Finally, Bill Barabash provided technical support on OPS5.

REFERENCES

1. J. McDermott, "R1: A Rule-Based Configurer of Computer Systems," Artificial Intelligence, Vol. 19, No 1, 1982, pp. 39-88.

2. J.H. Griesmer, S.J. Hong, M. Karnaugh, J.K. Kastner, M.I. Schor, "YES/MVS: A Continuous Real Time Expert System." Proc. of the Nat'l Conf. on AI, 1984.

3. C.R. Hollander, Y. Iwasaki, J.M. Courteille, M.Fabre, "The Drilling Advisor," Proc. Trends and Applications, 1983.

4. J.H. Kim, J. McDermott, D.P. Siewiorek, "Exploiting Domain knowledge in IC Layout," IEEE Design & TEST, August, 1984, pp 52-64.

5. H. Mori, K. Mitsumoto, T. Fujita, S. Goto, "Knowledge-Based VLSI Routing System - WIREX -," Proc. of the Int'l Conf. on Fifth Generation Computer Systems, 1984

6. W.P. Birmingham, D.P. Siewiorek, "MICON: A Knowledge Based Single Board Computer Designer," Proc. of 21st Design Automation Conf., 1984.

7. H. Brown, C. Tong, G. Foyster, "PALLADIO: An Exploratory Environment For Circuit Design," IEEE Trans. Comput., 1983.

8. R.A. Armstrong, M.T. Doreau, "A Topological Representation for Single Layer Routing Problems with Variable Sized Objects," Proc. of Int'l Symposium on Circuits and Systems, 1982.

9. J. Bachant, J. McDermott, "R1 Revisited: Four Years in the Trenches," AI Magazine, Vol. 5, No. 3, 1984.

10. H. Shubin, J. W. Ulrich, "IDT: An Intelligent Diagnostic Tool," Proc. of the Nat'l Conf. on AI, 1982.

11. E. Orchiuch, J Frost, "ISA: Intelligent Scheduling Assistant," The First Conference on Artificial Intelligence Applications, 1984.

12. C. Forgy, "The OPS5 User's Manual," Technical Report CMU-CS-81-135, Computer Science Dept., Carnegie-Mellon Univ., 1981.

13. L. Brownstone, R. Farrell, E. Kant, N. Martin, "Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming," Addison-Wesley Publishing Co., Reading, Mass., 1985.

14. K. Kukich, "Knowledge-Based Explanation Generation," Second Annual Language Generation Workshop, Stanford Univ., July 8-10, 1984.

15. K. DeJong, "Intelligent Control: Integrating AI and Control Theory," Proc. Trends and Aplications 1983. National Bureau of Standards.