SRC-TR-87-55

An Expert System for Electronics Design

by

B.T. Sawyer
Michael Pecht

AN EXPERT SYSTEM FOR ELECTRONICS DESIGN

B.T. SAWYER
Institute for Defense Analyses
Alexandria, Virginia

MICHAEL PECHT
Mechanical Engineering Department
University of Maryland at College Park

After three years of development, the University of Maryland Reliability and Maintainability Computer Aided Design (RAMCAD) expert system for electronics has been implemented for on-site test and utilization. This paper discusses the operation of the RAMCAD system, the structure of the system controller, and the impact of RAMCAD on established industrial design practices.

INTRODUCTION: AN EXPERT SYSTEM FOR ELECTRONICS DESIGN

It is not uncommon to associate the term "expert system" with those used in medicine to help in diagnosis. Such systems respond to questions or data written in a natural language by following rules which have been written by or developed from experts in diagnosis. Similar systems exist for financial prediction and other types of "what if" situations.

The electronics design process is an excellent candidate for an expert system shell. However, the design process differs significantly from the medical and financial models for expert systems. A comparison with the medical model helps to illustrate the problems involved. A good diagnosis depends on the ability to ascertain the symptoms correctly. Some of the data is subjective but once the data has been determined, there are a limited number of causes for such symptoms. Once the list of probable causes has been reduced, a diagnosis can be made and suggestions for a cure can be displayed.

The thermal, mechanical and electrical variables of a printed circuit board are quantifiable. For example, it is not difficult to determine by simulation techniques, the temperature and failure rate distribution for an electronic assembly. In many cases, it is also possible to identify the mode of the failure. Thus, to this point, an expert system like those used in medical diagnosis could be utilized in the design environment. The problem for a designer begins when the "cure" must be found. Component rearrangement, better cooling, alternate components, and adding redundancy are all possible. These in turn must be balanced against cost, producability, product life, and time constraints. Creating an expert system to handle these situations is no less difficult than developing one to identify the cause and suggest cures for a

medical problem which has never before existed.

This does not mean that expert systems for electronics design can not be developed. On the contrary, RAMCAD is an attempt to do just that using both procedural methods and AI techniques. RAMCAD provides the clear and concise information necesssary to help the designer make intelligent decisions. This decision support aspect of the system is based on computational methods which have been tested against measurements of real designs. In some cases, design solutions are provided. However, the design process from a systems viewpoint is very complex and the software used to aid the designer is even more complex.

RAMCAD is most useful in giving decision support and suggestions based on previously specified and quantified goals. Decision support and data context is provided by reporting monitors and interactive graphic displays. In addition, the system presents to the designer, information on how the data relates to the desired outcome. Critical failures or design rule violations are brought to the designer's attention and suggestions for improving the design are given. In effect, the system informs the designer on how the design effort is proceeding in the context of moving toward an optimum, but not necessarily well defined design goal.

The RAMCAD project was initiated in 1983. It is presently being utilized by Westinghouse Defense Electronic in the design of the electronics for the Airborne Self Protection Jammer (ASPJ). In this paper focus is on the structure of the system controller and the use of RAMCAD in industry. Other aspects of the RAMCAD architecture are treated in papers [4,6,7]. The thermal and reliability design concepts used in RAMCAD are explained in [2,3,5], and paper [1] discusses some of the tradeoffs which occur in the electronic placement process.

SCHEDULING AND CONTROL OF THE DESIGN

Design is a creative activity. The designer must be free to investigate alternatives and should not be forced along a predetermined path. For this reason, the flow of the design process must be left in the hands of the designer while the CAD software controls specific activities, reports the results of those activities, and executes all lower level functions such as error checking, data management, and computation. However, a CAD expert system can offer facilities for notifying the designer of any rule violations and more importantly, offering decision support for design optimization.

The RAMCAD executive controller integrates the system by presenting options and executing requests. It reflects our design philosophy by permitting open-ended choices of functions using a consistent and easily learned user interface. It performs general system tasks of maintaining a list of designs and verifying the existence of data and code files. It schedules the program modules required to carry out a request. It verifies the existence of the

necessary data and program files required to carry out the request and then initiates the sequence of program modules which perform the desired function.

The RAMCAD executive controller is primarily written in a procedural language. This is consistent with our idea that the activities that make up the process can be expressed as actions or procedures, even though the design process is open-ended. The user interface consists of a series of menus which are defined as program constants. There is a single menu selection procedure, but each menu has its own procedure which verifies and executes the possible choices. These procedures are linked to a central controller which passes system status data. Each menu displays a set of tools that are available for a specified task. Once a tool is selected, the system takes care of the details. The intent is to have a "seamless" system where tasks proceed in such a way that they appear to comprise a single action rather than a series of independent program modules. Furthermore, the data structures used in the RAMCAD controller are flexible enough to allow us to add or delete modules, change execution sequences, and rearrange options without major code revision.

Prior to the execution of a request, the system is checked for the presence of all required program and data files. If all files are present, the appropriate modules are scheduled. Otherwise the names of the missing modules are displayed. The flexibility of the system comes from the method we use to verify files and schedule program modules. These functions are carried out using identifiers whose values are defined in the program declaration section. Each program and data file identifier is mapped to an actual file name and then grouped into sets which may be enlarged, changed or rearranged without affecting the code. The structure and operation of the controller is best explained by an illustration. The coplanar thermal analysis function will serve as our example.

On the main system menu, there is an "Analysis" option, which presents the available analysis routines. The program definition of that menu is shown below. The number of choices and the first letter of each one is part of the declaration. These are followed by the choices as they are shown to the user. The identifier shows the position of the menu in the system hierarchy. "123" means the third level of choice number 2 of the main menu.

```
menu123 : menu_rec =
    ( entries : 4;
      choicestr : 'CISE';
      title : 'THERMAL/RELIABILITY ANALYSES';
      labels : ('Coplane',
                'Inline',
                'Standard Electronic Module',
                'EXIT to the Previous Menu',
                '', '', '', ''));
```

The statement, ch := Menu_Choice(menu123) performs the operation of menu display and selection, and returns the selected value to the character variable, ch. The resulting value of ch is passed through a CASE statement which determines the action to be carried out. The 'C' and 'E' options of this statement are shown below:

```
CASE ch OF
  'C':BEGIN                              {Coplanar thermal analysis}
        pnode := Check_pgm_list(coplane_pgm_s);
        enode := Check_ext_list(bd_name,coplane_ext_s);
        dnode := Check_datfi_list(coplane_dat_s);
        Mark(node);
        node^.next  := Link_nodes(pnode,dnode,enode);
        IF node^.next = nil THEN BEGIN
          Schedule(coplane_pgm_s);
          complete := true;
    END
        ELSE BEGIN
          Show_missing_list(node^.next,choice_name);
          complete := false;
        END;
        Release(node)
      END; {'C'}

  'E': BEGIN                                        {Exit menu123}
        act_menu := 'menu12';
        complete := false
      END; {'E'}
END; {case}
```

These two choices illustrate the way that all menu selections are handled in the controller. If a choice requires action, the presence of the appropriate files is checked. If they exist, a sequence of modules is scheduled. Otherwise the missing file names are shown and the menu returns a value of 'false' for the variable 'complete', which informs the menu control procedure that the choice can not be carried out and the active menu (act_menu) should be displayed.

The method of changing menus is illustrated by the exit ('E') option. If the user chooses to exit this menu (menu123) the value of the next menu is set (menu12) and 'complete' is set to false. This causes the menu controller to display the next menu (menu12) and await further action. The system control program only exits when the value of 'complete' is 'true.'

Verification and scheduling are accomplished by passing pre-defined values to Check and Schedule. The data structures for these values are defined in the program's declaration section which is illustrated below. Only identifiers relevant to this example are listed.

```
{identifiers for executable programs}
pgm_id_type   = (pgmF,
                ldb2ana, flowrow, coplane, tem2ldb, reliable,
                pgmL);

pgm_id_array  = Array [pgm_id_type] of str20;
pgm_set       = Set of pgm_id_type;

{identifiers for datafile extensions}
ext_type      = (extF,ldb,uky,boa,pwb,lay,extL);
ext_array     = Array [ext_type] of str4;
ext_set = Set of ext_type;

datfi_type    = {similar to the above declarations}
datfi_array   = Array [datfi_type] of str20;
datfi_set     = Set of datfi_type;
```

The assignment of a file name to each of the constant identifiers is done by means of the following program constant declarations.

```
program_map : pgm_id_array =
  ('', 'LDB2ANA.COM', 'FLOWROW.COM', 'COPLANE.COM',
       'TEM2LDB.COM', 'RELIABLE.COM',     ''  );

extension_map : ext_array =
  ('', '.LDB', '.UKY', '.BOA', '.PWB', '.LAY', ''  );

datafile_map :  datfi_array =  {as above}
```

The program, extension and datafile identifiers are used to build sets of values for each of the sequences in the system. Each set is defined as a program constant and given a name that identifes the function it carries out. The constant sets for the thermal analysis coplane program are

```
coplane_pgm_s : pgm_set = [ldb2ana,flowrow,coplane,tem2ldb,reliable];

coplane_ext_s : ext_set = [ldb,uky,boa,pwb,lay];

coplane_dat_s : datfi_set = [];
```

The main program's Check_pgm_list function scans the set of values (in our example, coplane_pgm_set) to see if the corresponding file names exist. If not, a pointer to the missing name(s) is returned. Otherwise the 'nil' value is returned. Check_ext_list appends the file extensions to the board name since this is the method we use to identify data files belonging to a particular design. The Check_dat_list function checks for data files which are required by the program modules. If all files exist, coplane_pgm_s is sent to the Schedule procedure which writes the names of the files to a text file. As soon as the control program ends, a background program executes the

programs in the text file and finally runs the controller to complete the cycle.

In our example, the following programs are executed for a board named "TEST":

Ldb2ana : extract data from the local database (TEST.LDB) and put it in an ASCII file (TEST.ANA) for the next program to pick up.

Flowrow : show a graphic display of the PWB (using TEST.BOA and TEST.LAY) and preprocess geometric parameters. Place the results in TEST.ANA.

Coplane : read data and get thermal parameters from the user. Execute thermal analysis and display the results. Put the results in an ASCII file (TEST.TEM).

Tem2ldb : pick up the data produced by Coplane and put it into the local database.

Reliable : Perform a failure rate calculation for each component based on temperatures generated by the coplanar analysis. Calculate the total failure rate for the board and display the results.

These programs execute in sequence. The user interface is the same in all of them so one is not really aware that the controller has stopped and that five programs are executed before it resumes.

If a module fails to operate, the subsequent modules check this by means of a "state" data structure that all programs access before and after operating. If a program fails, the next one in line determines this from the "state" record and does not execute. Thus "nothing" happens until the controller becomes active. At this time error messages are presented.

The flexibility of this programming method is apparent, in that any series of programs can be executed by grouping the appropriate identifiers into a set. A new module is added by creating an identifier and a file name, and by appending them in the appropriate declaration sections. For example, a report module for reliability analysis can be introduced into the coplane sequence with the declaration

```
coplane_pgm_s :
      pgm_set = [ldb2ana,flowrow,coplane,tem2ldb,reliable,report];
```

The identifier 'report' and its associated name 'REPORT.EXE' would be added to the respective lists. No other program modification is necessary. A module is removed by taking its name out of the list.

Introducing a new menu requires the addition of a procedure (e.g.

Domenu123). This is not difficult since all menus use the same Check and Schedule routines. A new menu is easly inserted into the main controller, a portion of which is given below:

```
Procedure Menu_controller(VAR bd_name:keystr;
                  VAR active_menu:str20; VAR initok : boolean);
                                  {Central controller for menus}
VAR
  completed : boolean;
BEGIN
  IF NOT (initok) THEN  {no active board, select or initialize one}
    active_menu := 'menu11';        {first time through}
  REPEAT
    completed := false;
    IF active_menu = 'menu1' THEN              {Outermost Menu}
      Domenu1(active_menu,completed,bd_name,initok)
    ELSE IF active_menu = 'menu11' THEN        {Board Manager}
      Domenu11(active_menu,completed,bd_name,initok)
    :
    :
    ELSE IF active_menu = 'menu123' THEN       {Thermal Menu}
      Domenu123(active_menu,completed,bd_name)
    ELSE
      active_menu := 'menu1'    {default to main menu}
  UNTIL completed;
END; {Menu_controller}
```

Each time this is run, the system controller checks the state of the active board, updates the status of each initialized board in the system and picks up the most recently active menu (if any). This value is passed to the Menu Controller which displays the menu which was last selected. A closing procedure does system housekeeping and stores the active menu's name so it can be accessed when the system controller becomes active again.

MONITORING AND DECISION SUPPORT

Progress towards a desired design state can be monitored and supported by examining both individual and system parameters and comparing them to allocated values, optimum values, and values obtained by conducting tradeoff studies. The creative and exploratory nature of the design process requires that competing and sometimes conflicting subgoals be balanced in order to achieve the best design. This means that more than simple optimizing routines must be employed.

The monitor and decision support modules in RAMCAD are based on representing the state of the data in relation to desired goal states and allowing the designer to examine alternatives. Movement towards a desired goal is not limited to a linear sequence of processes. Rather it is represented by data states which result from the flow of data through various program modules. The designer is allowed to experiment with alternatives

which may produce a less desirable state. Such a condition, once reported, helps strengthen confidence in decisions which produce positive results.

Feedback on the design progress is provided by modules which check the data against design parameters. Some of the decision support modules are executed as part of a sequence, usually after an analysis has been completed. Others are themselves sequences which perform iterations of analyses for different parameters. The decision support system is enhanced through the use of graphic representations which allow the designer to examine the data in the context of the total design and the conditions that have been specified. This not only promotes better decisions but reduces the chance of forgetting or overlooking details. For example, one interactive module displays a printed circuit board subdivided into user defined rows and columns to show how the power is distributed. This dynamic data display can be used to identify potential hot spots on the board. The system has color coded displays of the power, temperatures and failure rates of components on the board. These displays can be used to examine regions of high power density, board hot spots and components with high failure rates. This is important, since in many instances, the highest failure rates do not correspond to the hottest regions.

The MTBF versus temperature trade-off module is another example of monitoring, dynamic data display and decision support. The board's mean time between failures (MTBF) is plotted as a function of temperature. The designer can use the arrow keys to move along either axis. The movement is reproduced on the graph and at the same time the values for the resulting coordinates are displayed. At all times the calculated MTBF, the allocated MTBF and the environmental temperature are compared. This makes it easy to see what would happen if the assembly were cooled or whether or not the allocated value is in a "safe" region of the curve. The same display can be accessed after the board has been analyzed for actual component temperatures. This not only gives immediate feedback, but it facilitates decision making by putting the design parameters in a wide context.

The operation of the RAMCAD system is based heavily on exact analysis of the data. Although military standards allow for approximations in the early stages of design we have taken pains to make the analytical results as close to the actual situation as possible. In many cases exact analyses is conducted even though they may take slightly more time than the approximations. The reasoning stems from potential inaccuracies which can arise is multi-goal tradeoffs between nonlinear (often exponential) objective functions and constraints.

As an example, the coplanar analysis module treats components as distributed rather than point-source heat producers. Thus, the thermal effects of one component on its neighbors is taken into account and the junction temperatures of the components more closely simulate the real situation. Row locations, air-flow rate,

fin conductance, inlet temperature, and location of the inlet are all specified by the user. Our goal in this analysis is to produce a temperature pattern that will resemble those generated by infrared thermographs taken in our test lab.

Once temperatures have been determined for each component, the failure rates are calculated from the formulas given in the MIL-217E Handbook. Each component's data record contains all of the MIL-217E constants needed to calculate reliability. All of the acceleration constants, shaping parameters, and stress constants are used in the calculation. This is in contrast to the method of using look-up tables for failure rates. The coplanar and reliability analyses can be completed in less than three minutes for a board with 300 components, using an IBM-AT.

The benefit of such analyses to the designer is considerable. Potential flaws in a design appear early enough to allow for correction without undue loss of time or effort. Of equal importance is the level of confidence produced. Since decisions are based on the available input and simulation data, we feel that the best data helps one make the best decisions.

RAMCAD IN INDUSTRY

There are problems with leaving decisions up to the designer and they relate to the way in which a particular institution deals with the design tasks. The development of independent design tools has led to the partitioning of design tasks and decision making within an institution. For example, it is common practice to divide the electrical, thermal and reliability aspects of a design among different groups, with each group functioning separately and sometimes in conflict. This reality has serious implications for those who are creating integrated expert design systems.

Software which provides electrical-thermal tradeoffs is of little value if there is no one with the authority to make a tradeoff decision. For example, a reliability engineer does not rearrange or substitute electrical components to meet reliability criteria. Rather, a report is given to the electrical engineers who are responsible for rearranging or changing components.

To address this problem, the RAMCAD expert system uses a modular approach which allows for data access and sharing by means of a network or workstation cluster. This does not adversely affect the institution's decision making hierarchy. Furthermore, it promotes communication among various design groups and reduces the lag time that is caused by separation of electrical, thermal and reliability tasks in the design cycle.

In some instances management may impose new methodologies for design but introducing integrated design software in a modular fashion permits a more gradual transition to new ways of treating the design problem in the institution. Just as separate design

tools gave rise to specialists, integrated design tools will produce generalists. Without the latter, integrated design software will not find a welcome place in the design community.

The use of integrated CAD software for electronic assemblies not only requires changes in the institutions that use it but also in the methods used to write it. Our experience has shown that computer aided design is more than a programming problem. Good programmers are necessary but not sufficient for the task. Engineers must be involved in all stages of program development. Ideally, each group should have some knowledge of the others' discipline in order to produce software that accurately describes the state of a design.

For CAD software to be successful, it must be used in real-world situations as early as possible. This provides feedback on the the engineering problems and on the human interface problems that the software must address. Our association with Westinghouse has proven very beneficial in this regard. The data structures which we used early on were sound from a computer science point of view but they were not as adaptable to industry requirements as our present structures. The extensive use of text files which the programmers initially resisted has proven to be a source of flexibility especially with regard to interfacing with other design software. Were it not for this industry feedback, one of the tasks initially set forth by IDA, the ability to interface with other systems, might not have been accomplished to the extent it has been.

In the best of all possible worlds, the companies that design electronic equipment for the government would share resources and technologies. DOD's initiative to encourage the formation of consortia for initial proposal development shows that such cooperation is possible and that there are alternatives to the traditional view of "winner take all" that has characterized the defense industry. In this regard, industry can be challenged to alter its proprietary attitude toward the software it develops and uses. It is possible for separate groups to develop integrated systems but it is more efficient to integrate the many excellent tools that already exist. This requires some sharing of technology but is the only way that a comprehensive system can be developed in a reasonable length of time.

ACKNOWLEDGEMENTS

This effort has been supported in part by the Institute for Defense Analyses, Westinghouse Defense Electronics, and the University of Maryland Systems Research Center under grant No. CDR8500108 from the National Science Foundation.

REFERENCES

1. "An Investigation Into PWB Component Placement Tradeoffs", M. Pecht, M. Palmer, W. Schenke and R. Porter, accepted for publication to IEEE Transactions on Reliability, Aug., (1986).

2. "Thermal Reliability Management in PCB Design", M. Pecht, M. Palmer, and J. Naft, Proceedings: 1987 Annual Reliability and Maintainability Symposium, Jan. 27-29 (1987).

3. "Computer-Aided Heatsink Design For Printed Wiring Boards", M. Pecht, M. Palmer and J. Horan, Proceedings: IEEE International Conference on Computer Aided Design, ICCAD-85, Vol.1, pp.253, Nov 18, (1985).

4. "Intelligent Design of Printed Wiring Boards", J. Horan, M. Palmer, M. Pecht and Y. Wong, Proceedings: Association For Computing Machinery, Vol.1, pp.123, (1985).

5. "PCB Design for Thermal Reliability", M. Pecht and B.T. Sawyer, accepted for publication, Printed Circuit Design Journal, Feb. (1987).

6. "A RAMCAD\ULCE Workstation Shell Structure", J. Naft and M. Pecht, Proceedings : 1987 Annual Reliability and Maintainability Symposium, Jan 27-29, (1987).

7. "Workstation Requirements for Printed Wiring Board Design", M. Pecht, J. Naft, and M. Palmer, Proceedings: IEEE Workstation Technology and Systems Conference, Vol. 1, pp. 63, Mar. 18-20, (1986).