A Rule-Based Expert System for the Diagnosis of Convergence Problems in Circuit Simulation

Christopher W. Lehman D.CS
IBM
Colorado Springs, CO 80903

Mary Jane Willshire Ph.D.
Department of Computer Science
Colorado Technical University
Colorado Springs, CO 80919

Abstract Industry standard circuit simulators use mathematical techniques that do not always guarantee a successful simulation outcome, resulting in a significant loss of productivity as the circuit designer attempts to diagnose the source of the failure. This paper addresses the problem of improving the diagnosis of circuit simulation non-convergence through computer automation of the diagnostic process. SOAR – Simulation Output Analysis and Recommendations – is the software application that implements the proposed solution to the problem.

Circuit simulation convergence failure analysis heuristics can be encapsulated into a rule-based expert system environment that emulates the reasoning of experienced circuit designers. SOAR integrates expert system technology with a modern web-based framework. Utilizing a database to store and retrieve simulation information gives SOAR the ability to diagnose simulations ranging in size from gate-level circuits to full chip architectures with virtually no increase in processing time.

Keywords: simulation, circuit convergence, expert system, diagnostic systems, DC operating point/bias point analysis

I. INTRODUCTION

Traditional circuit design was characterized by building a circuit prototype, and then providing the prototype with various stimuli (such as input signals, temperature variations, and changes in power supply voltages). Laboratory equipment was used to make measurements on the circuit prototype as the stimuli were applied.

With the advent of Very Large Scale Integrated (VLSI) circuit design, prototypes are no longer practical. The breadboard prototype of the circuit will not have the same electrical, mechanical and parasitic components as the VLSI equivalent.

Software provides a solution to the problem. Computer programs that simulate the performance of an electronic circuit can be used to confirm the performance of the circuit prior to committing fabrication resources to the actual construction of the circuit. Circuit simulation software has therefore become a vital part of the IC design cycle.

The most prevalent circuit simulators use the Newton Raphson (NR) method (or some variation) for solving nonlinear systems of equations. The NR algorithm within the circuit simulator starts with an initial guess for each node voltage in the circuit and begins its iteration process. The iterations continue until the solution is reached within the specified allowed margin of error. The drawback to the NR method is that it will not converge on a solution unless the required initial guess to start the iterations is sufficiently close to the solution [1], [3], [4]. When non-convergence occurs, the circuit designer must begin the time-consuming process of determining the cause of the failure.

The knowledge associated with the diagnosis and elimination of convergence problems in circuit simulation includes written knowledge, composed of books, papers, memos and "how to" documents, and user experience and expertise. This knowledge can potentially be analyzed and rendered into a set of inference rules. In other words, rule-based representation of the available knowledge is possible [5].

II. Related Work

A. *Avoiding Convergence Failure*

A search of the literature indicates that circuit simulation convergence issues are generally tackled by either 1) attempting to embellish existing techniques (algorithmic or numeric) for avoiding convergence failure, or 2) by providing heuristic solutions for non-convergence after the simulation fails [6].

It is possible to have a realistic circuit, but one or more models that represent the circuit elements may have difficulty in converging. Simulator engines therefore typically have built-in strategies for dealing with this type of circuit.

B. *Identifying Convergence Failure After the Fact*

When convergence failure occurs the designer must resort to trial and error in an attempt to a) identify the source of the problem, and b) make the necessary corrections prior to restarting the simulation of the circuit. As a result, vendors

Manuscript received May 05, 2006.

usually provide their customers with guidelines for troubleshooting convergence problems [1], [2].

In addition, the design community within an organization will often publish internal memos, guidelines and manuals that describe convergence problems that have previously been encountered.

A designer with many years of experience in circuit design will also have developed a certain amount of expertise in solving convergence problems. An experienced designer can be quite adept at solving convergence problems, but in many cases the expertise has not been formally captured.

C. *Expert Systems and VLSI Design*

Expert Systems have been utilized within the VLSI design cycle, especially during the 1980's when both technologies began to mature. The height of the Expert System/VLSI fusion was during the 1988 to 1991 time period [6]. However, researchers during this period were working in problem domains that were too broad [6]. Despite the rather gloomy outcomes of the early attempts at expert system solutions, expert systems continue to show promise in the areas of VLSI test, and for solving specialized VLSI and printed circuit board placement and routing problems [6].

III. Architecture of the Expert System and User Interface

A. *Architectural Considerations*

An important portion of the work described in this paper consisted of the task of choosing the most appropriate platform from which to build the expert system. The system needed to be low-cost, but flexible enough to work with large amounts of textual data. The C Language Integrated Production System (CLIPS) [7] was selected as the expert system environment because it is public domain software, the C source code is available for modification, it uses a simple but extensible command-line driven user interface, it contains a number of conflict resolution strategies, and the authors have previous experience with the tool.

As noted, the CLIPS user interface is constrained to command line invocation and output to either a file or a text-only environment. For academic purposes, this is perfectly acceptable. For the more general case, a more flexible solution is desirable. A web-based user interface was determined to be the most efficient method of deploying the application for the following reasons:

1 Web-based applications are accessible anywhere, at any time.

2 Software distribution can be controlled from the originating web page, and can also more easily track customer usage patterns and problems.

3 Validation of expert system knowledge bases can be a major challenge [8]. A web-based expert system's releases can be more readily controlled by the author.

Another major architectural consideration was the amount of data that the expert system would need to access. Simulation output files routinely consist of megabytes of textual information contained in structured patterns. Repeated searches over the entire file would be time-consuming and repetitive. On the other hand, programmatic extraction of pertinent data is possible because of the repeating textual structures. Once the data is extracted, it could be transformed into more regular structures such as database records. Database queries on simulation information were therefore determined to be feasible and would allow for faster processing of the available information about the simulation.

Given this analysis of the implementation features, the decision was made to utilize the Ruby on Rails (RoR) software environment for the construction of the user interface to the expert system.

Ruby on Rails [12] uses the Ruby programming language within an open-source framework for developing web-based applications. The Ruby language [9] combines the object-oriented programming power of Smalltalk with the scripting power of Perl.

B. *SOAR: Simulation Output Analysis and Recommendations*

The expert system that has been developed is called **SOAR**, that is, a program that provides for **S**imulation **O**utput **A**nalysis and **R**ecommendations. A block diagram of the overall SOAR architecture is shown in Figure 1.

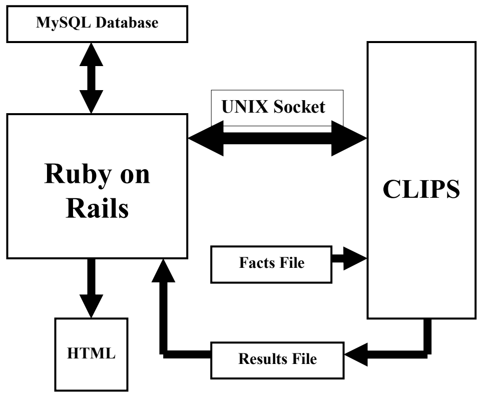


Figure 1 - SOAR Block Diagram

The primary design elements of SOAR are:

- The use of a UNIX socket as the means of communication between the Ruby on Rails application and the CLIPS expert system.
- The creation of files to transfer information into and out of the CLIPS expert system's text-based external interfaces.
- A reliance on the built-in database connection features of the Ruby on Rails application.
- A uniform "look and feel" of the Graphical User Interface through the use of Cascading Style Sheets coupled with HTML-embedded Ruby code.

1) *User Interface*

The simulation file is read and parsed, and the parameter, option and model information is extracted, reformatted and placed in a MySQL database. The user has the option of viewing the extracted parameters, options and model information by clicking on buttons placed on the web page.

Once the user has finished inspecting the various database records, the "Start The Checks" button is pressed, and program control is passed to the CLIPS expert system. Control is passed back to the Ruby on Rails interface when CLIPS has completed its analysis, and a web page is displayed that contains recommendations returned by the CLIPS program.

2) *Database*

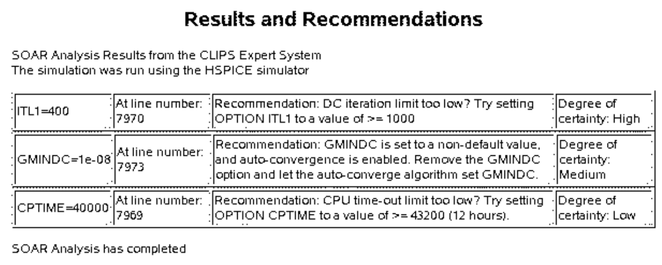
Ruby on Rails was specifically designed to interact with databases. Once a database is created, RoR can access the data almost immediately via a built-in code generation process that sets up the necessary hooks for basic operations on database tables, commonly referred to as CRUD – Create, Read, Update, and Delete. Remaining consistent with the open-source approach to the research, the MySQL database [10] was selected.

3) *CLIPS Implementation*

In an attempt to maintain consistent Object Oriented Programming (OOP) coding standards, the CLIPS rule set was written in COOL, the CLIPS Object Oriented Language, as defined in [11]. Circuit simulators, and their associated rule sets, are defined using the *defclass* declaration. Additional simulators and rule sets can be defined by declaring additional *defclasses*. When a rule set changes, only the associated classes are affected – the verification and validation of the CLIPS output is therefore confined only to the classes associated with the change, thus reducing the scope of the verification and validation effort.

The function, SOAR_sock_client.c, establishes a socket connection with the Ruby on Rails GUI. The user function was compiled and linked with the standard issue CLIPS modules, creating a custom executable. The CLIPS version used was 6.22.

When the CLIPS program is invoked, it begins to request information about the simulation. This is the "fact-gathering" stage of the operation. The facts to be extracted from the simulation are those associated with user-configurable options that direct the processing details of the simulation. These details include such features as error tolerance values, voltage and current constraints, algorithmic options during runtime, and options to regulate the simulation speed and accuracy. The options typically consist of a keyword and a value, although there are some options that when specified, act simply as flags to modify the behavior of the simulation. The Rails interface is used to capture a CLIPS request for a particular fact to be extracted. Rails sends the request to the database server, and then passes the result back to the CLIPS program. This process continues until all database queries are sent and processed, i.e. until all required facts are assembled. The appropriate rule sets are then applied to the fact set. When rule processing is complete, control is passed back to the RoR interface. The RoR interface displays the recommendations to the user in HTML table format. A partial rendering of a sample web page presented to the user is shown in Figure 2.



Results and Recommendations

SOAR Analysis Results from the CLIPS Expert System
The simulation was run using the HSPICE simulator

ITL1=400	At line number: 7970	Recommendation: DC iteration limit too low? Try setting OPTION ITL1 to a value of >= 1000	Degree of certainty: High
GMINDC=1e-08	At line number: 7973	Recommendation: GMINDC is set to a non-default value, and auto-convergence is enabled. Remove the GMINDC option and let the auto-converge algorithm set GMINDC.	Degree of certainty: Medium
CPTIME=40000	At line number: 7969	Recommendation: CPU time-out limit too low? Try setting OPTION CPTIME to a value of >= 43200 (12 hours).	Degree of certainty: Low

SOAR Analysis has completed

Figure 2 - Results Displayed to the User

IV. TEST CASES

The test cases were derived from actual circuits created for use in CMOS Static RAM (SRAM) memory chips, using many full-custom analog circuits in order to meet stringent design requirements; as a result, convergence problems are commonly encountered when simulating the various analog components of an SRAM design. Specific test case selection was based upon three general requirements:

1 The test cases needed to be based upon real problems encountered by real designers. Each test case represented a circuit that, at one point during the design cycle, failed to converge. Each test case was analyzed by SOAR, i.e. the simulation information was read into the SOAR application, and a solution was returned. When the recommended solution was implemented and the circuit was re-simulated, each test case then attained convergence.

2 The sizes of the circuit test cases were selected to be broad enough to represent the entire design cycle in order to demonstrate that SOAR would work with state of the practice designs. The test cases represent circuit sizes spanning three orders of magnitude, from small 16 node circuit simulations to full chip simulations of over 11,000 nodes.

3 The scope of the work was limited to DC operating point simulations of analog circuits, and the test cases were selected accordingly.

V. RESULTS

Table 1 presents the results of the individual test cases.

Table 1 - SOAR Test Case Results

Name	Test	Number of Circuit Nodes	Circuit Description	CLIPS Run Time (seconds)
Case1	No Errors	1161	tAA	25
Case2	All Errors	1161	tAA	25
Case3	ACCT	1161	tAA	27
Case4	ITL1	1161	tAA	25
Case5	CPTIME	1161	tAA	26
Case6	RELV	16	Trip Point	25
Case7	ABSV	16	Trip Point	27
Case8	ABSI	16	Trip Point	25

Case9	Error Tol.	16	Trip Point	25
Case10	KCLTEST	510	LVDetect	25
Case11	DV	25	Senseamp	30
Case12	DCON	11131	Read 100C	32
Case13	GMINDC	11131	Read 25C	36
Case14	CONVERGE	11258	tHA	32

Case 1 was a custom simulation file that deliberately tests for CLIPS “false positives”; all CLIPS rules are exercised, but no rules should generate a recommendation. Case 2 was also contrived in order to force all rules to detect and display a recommendation. The remaining test cases were used without modification from the original simulations.

Table 1 contains circuits with node counts that span three orders of magnitude. It is remarkable to note that the run times are virtually the same regardless of circuit node count. Transforming the circuit information into database records is a key element in the ability of the SOAR application to scale up to VLSI-sized circuits.

In all test cases, the simulation originally failed to converge. When the simulation was re-run using the solution recommended by SOAR, the circuit attained convergence. A 100 per cent accuracy was attained for the test case set.

VI. Future Direction of Research

Based upon the work thus far completed, a number of enhancements and new research directions can be contemplated.

Not all of the possible convergence problem classes are represented in the current version of SOAR. Additional queries and associated rule sets could be added for more comprehensive coverage of circuit simulation convergence scenarios.

Due to the OOP modular construction of the GUI and the CLIPS code, additional circuit simulators can be added without affecting the existing components of the system.

Database queries pertaining to model information and additional rule sets designed specifically to diagnose potential problems associated with model definitions could be added to the application.

Now that a method for transforming circuit simulation information into a database has been developed, analysis of a circuit is not confined to convergence problems alone. SOAR could be altered to act as a “best practices” guide for use by new college graduates who are just beginning to learn to be circuit designers.

VII. Conclusion

The research activity in this paper has addressed the problem of non-convergence associated with analog VLSI circuits, an ongoing problem for the popular simulation environments that use the Newton-Raphson root solving algorithm.

This paper has demonstrated that rule-based expert systems can successfully diagnose analog circuit simulation convergence problems for circuit complexities ranging from simple gate-level functions and all the way up to full chip design hierarchies. The research described in this paper:

1 Converted heuristics for diagnosing DC operating point convergence problems into a software application.

2 Implemented a process that transforms text-based circuit simulation information into database records.

3 Designed and implemented processes by which the simulation database records were converted into sets of facts suitable for use by a rule-based expert system.

4 For every non-convergent test case circuit, the recommendations made by SOAR allowed the circuit to converge.

5 The expert system successfully processed the massive amounts of information associated with state of the practice VLSI designs, thus demonstrating the ability of the application to scale up to the data-intensive rigors of modern circuit design.

6 The test set contained circuits with node counts that spanned three orders of magnitude, but the run times were virtually the same regardless of circuit node count. Transforming the circuit information into database records was a key element in the ability of the SOAR application to scale up to VLSI-sized circuits.

References

[1] Ron Kielkowski, *Inside SPICE* Second Edition. New York, New York, McGraw-Hill, Incorporated, 1998.

[2] Synopsys, Incorporated, *Convergence*. unpublished.

[3] Francky Leyn and Georges Gielen and Willy Sansen, “An Efficient Root Solving Algorithm with Guaranteed Convergence for Analog Integrated CMOS Circuits,” ACM ICCAD98, pp. 304–307, 1998.

[4] William H. Press and Brian P. Flannery and Saul A. Teukolsky and William T. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*. New York, New York, Cambridge University Press, 1993.

[5] Frederick Hayes-Roth, “Rule-Based Systems,” Communications of the ACM, Volume 28, Number 9, pp. 921–932, 1985.

[6] C. W. Lehman, “A Rule Based System for the Diagnosis of Convergence Problems in Circuit Simulation,” Doctoral dissertation, Dept. Comp. Sci., Colorado Technical University, Colorado Springs, CO, 2006.

[7] Joseph Giarratano and Gary Riley, *Expert Systems Principles and Programming*. Boston, Massachusetts, PWS Publishing Company, 1998.

[8] Avelino J. Gonzalez and Douglas D. Dankel, *The Engineering of Knowledge-Based Systems Theory and Practice.* Prentice-Hall, Incorporated, Englewood Cliffs, New Jersey, 1993.

[9] RubyCentral Home Page, Available: http://www.rubycentral.com, 2005.

[10] MySQL AB Home Page, Available: http://www.mysql.com, 2005.

[11] Chris Culbert and Gary Riley, *CLIPS Reference Manual Volume I Basic Programming Guide*. unpublished.

[12] Dave Thomas and David Heinemeier Hansson, *Agile Web Development with Rails*. Raleigh, North Carolina, The Pragmatic Bookshelf, 2005.