# HIGH-SPEED MANAGEMENT SYSTEM FOR SIMULATING AND RETRIEVING DESIGNS OF COMPUTER COMPONENTS

Technical Report CSE-89-06

P.A.D. de Maine
D. M. Head

Department of Computer Science and Engineering
Auburn University
Auburn, AL 36849-5347

September 18, 1989

# A HIGH-SPEED MANAGEMENT SYSTEM FOR SIMULATING AND RETRIEVING DESIGNS OF COMPUTER COMPONENTS

D.M. Head and P.A.D. de Maine

Automatic Systems for the Physical Sciences
Report Number 12

Computer Science and Engineering Department
Auburn University, Auburn, AL 36849

30/June/1988

# A HIGH-SPEED MANAGEMENT SYSTEM FOR SIMULATING AND RETRIEVING DESIGNS OF COMPUTER COMPONENTS

D.M. Head and P.A.D. de Maine

Report No. 12

**ABSTRACT**

Currently available database systems have inherent limitations, which prohibit their use for many modern applications. The Self Organizing Large Information Dissemination (SOLID) system was designed to overcome these limitations.

Because the SOLID system offers considerable flexibility, it also requires control information. For example, the SOLID system allows one of three different formats for the index. A decision-maker must choose one format, and inform the SOLID system which format has been chosen.

One possible decision-maker is the user. If SOLID needs control information, the user supplies it. Unfortunately, this approach assumes that the user has a working knowledge of the SOLID system.

Another possible decision-maker is an interface between the user and the SOLID system. The primary goal of the SOLID user-interface is to allow the user to exploit the features of the SOLID system without requiring a working knowledge of it.

Keywords: Database Systems, Self Organizing,SOLID System, User Interface, Electrical Engineering Application.

## The SOLID System

SOLID is a general information management system that has been described in [1-3]. It is a simulated communications network in which the queries themselves describe information paths to the desired information. Its principal characteristics are:

1) The SOLID system is logically data independent, which means that it is information independent and the user does not need to know anything about the way the information is organized.

2) The maximum search time for processing any normal explicit or non-explicit query must be very small and virtually independent of the sizes and number of databases.

3) There is an easily used and "fool-proof" security system that can be easily invoked at both the management and user levels.

4) Memory, secondary storage, and communications channels are used economically.

5) The system is modular with respect to software and hardware so that it may be used in distributed networks or on a stand alone basis on both large and small machines [3].

6) It can be easily installed as the database management part of existing retrieval systems.

Details regarding the organization, performance, and security of the SOLID system are beyond the scope of this article but may be found in the references to [2].

## The Problem

For the SOLID system the queries are converted to control information that includes the locations of security locks and the JOBLIST Data Structure whose basic components (called JOBLIST Items) are the "telephone numbers" that are used to trace the

"information paths" to the desired information. There are three different forms of JOBLIST Items and the application determines what form should be used. For example, an inventory application uses the simple form of JOBLIST Item [3], which cannot be manipulated. On the other hand the Information Representation form must be used for graphical representations [4], like chemical formula, that must be manipulated.

The problem is to create a user-interface that will accept queries in an easily understood, natural language form and transparently construct the control information that is used by the SOLID system.

## The Solution

The solution is a two-tier expert system interface. The first part, which is the actual user interface, will be table driven and determine the keywords that are to be used. This part will be application independent and, by replacing the tables, it can be easily changed to accommodate different levels of users or different applications. This part will not be considered further in this paper.

The second part is application specific. It will use the output from the first part to construct the control information that is required by the SOLID system.

## Organization of the Application Dependent Part of the SOLID User-Interface

The SOLID user-interface requires an application module for each application of the SOLID system. Each application module is responsible for:

1) accepting queries from the user,
2) editing queries for errors, and
3) informing the user of the type and location of any errors.

The application module of the interface prototype allows users to enter descriptions of computer hardware. These descriptions are used as indexes for the database query, and are in the form of predefined FORTRAN subroutine calls. A simplified example of one description is:

```
CALL AND ( INPUT1, INPUT2, OUTPUT )
```

where the AND subroutine represents a logic gate, and the parameters represent the input/output signals. The interface then compiles the description in order to detect any syntax errors. The compiled description is then executed (or simulated) to detect any logic errors.

All application modules are linked to a single system module which is responsible for:

1) accepting queries from the application module,
2) compressing queries,
3) normalizing queries, and
4) validating the user's security clearance.

The above AND gate description is accepted by the system module and compressed into the form:

ALBJBKCN

where AL = AND, BJ = INPUT1, BK = INPUT2, and CN = OUTPUT. This is done through the use of a symbol substitution table, which has three fields: the symbol referenced by the user, the corresponding substitution symbol, and the security classification of that symbol.

As the system module replaces each user-referenced symbol, it compares the security code for that symbol with the user's user-id. Once the description has been compressed, it is then transformed into a unique form. Since any given referenced item may have several descriptions, the system module must insure that only one unique description is submitted to the SOLID system. To insure the integrity of the database, each unique item must have only one unique description. In effect, all descriptions are sorted alphabetically prior to being submitted to the database.

## Application to Assist in the Design of Computer Systems

The application chosen for the prototype application module was a computer hardware design library. Since VHDL promises to be the first standard hardware description language [5], it was chosen as a model for the prototype application module.

In 1981, the Department of Defense (DoD) initiated the VHSIC program to reduce IC design time. In 1983, DoD initiated the VHDL

program because of the need for a standard means of communication to streamline IC design and documentation. A group of three companies: Intermetrics, IBM, and Texas Instruments was awarded a contract to design VHDL and to implement the support environment software [5].

VHDL can be considered the "FORTRAN of hardware description languages in the sense that it provides the first generally accepted standard hardware description language" [6]. Because of the almost-complete set of features, VHDL is the yardstick by which other hardware description languages (HDLs) can be measured.

VHDL supports hierarchical design, descriptive continuum, types, generic components, and tests [7].

Hierarchical design "implies that design unit description at any level will consist of an interconnection of a collection of lower level units" [7]. For example, a register can be described as a register at one level, as a collection of flip-flops at a lower level, and as a collection of gates at an even lower level. The effect of hierarchical design is the same as that of modular program design, since they both enable top-down and bottom-up design.

The descriptive continuum refers to the representation of the design unit. The representation may be a structural, behavioral, or geometric abstraction. A structural abstraction is one that can be decomposed to a collection of gates. A behavioral abstraction is one that can be decomposed to a collection of Boolean expressions. A geometric abstraction is one that can be decomposed to a collection of circuits. Figure 1, the Gajski-Kuhn "Y" diagram, illustrates the relationship between these abstractions [5].

Typing refers to the values that a signal may have. "It is assumed that strong typing does not apply when only Boolean values are permitted for signals." [7] For example, when a tri-state buffer is in a high-impedance state, VHDL allows the signal to have a value of "Z" [8].

A generic component is one in which "a particular parameter governing its structure, behavior, or environment, may differ in other instances (or uses) of the entity..." [8]. Generic components increase a design unit's reusability. Examples of generic parameters in VHDL include propagation delay, and operating temperature [6].

Testing implies the simulated operation of the design unit. Figure 2 is a block diagram of the VHDL support environment required for simulation [9].

The analyzer edits the description of the design unit for any errors that can be detected prior to simulation (static errors), such as syntax errors [9].

"The design library is a collection of the intermediate form representation of already-analyzed hardware descriptions stored in a structured library..." [9].

The simplifier takes a description with a hierarchical (multi-level) structure, and decomposes it into a single level description [9]. The effect of the simplifier is similar to that of an assembly-language macro processor, since they both produce in-line code.

The simulator compiles and executes the description for the purpose of editing it for dynamic errors, such as excessive propagation delays [9].

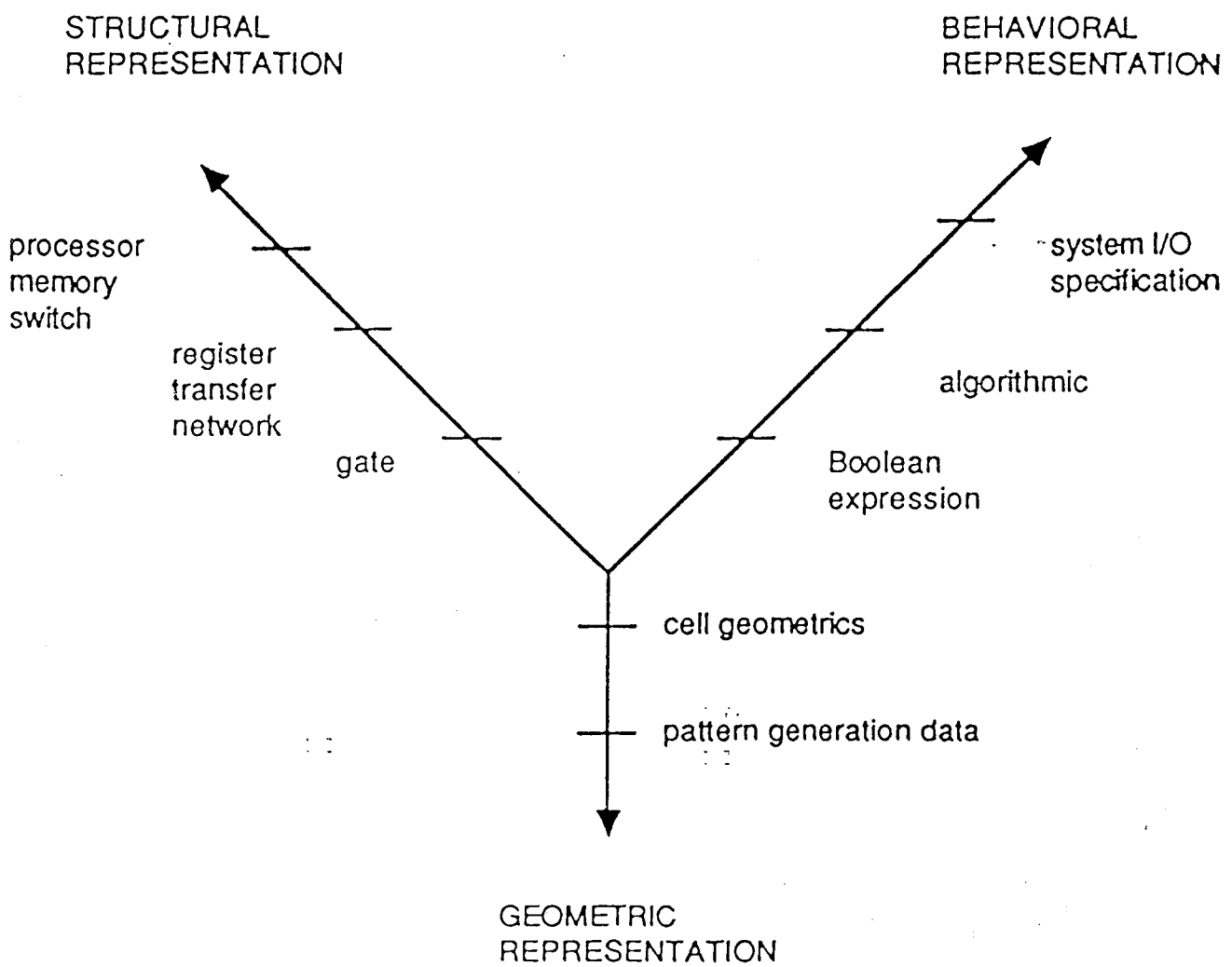


Figure 2. The Gajski-Kuhn "Y" diagram illustrating the relationship between decomposable representations.

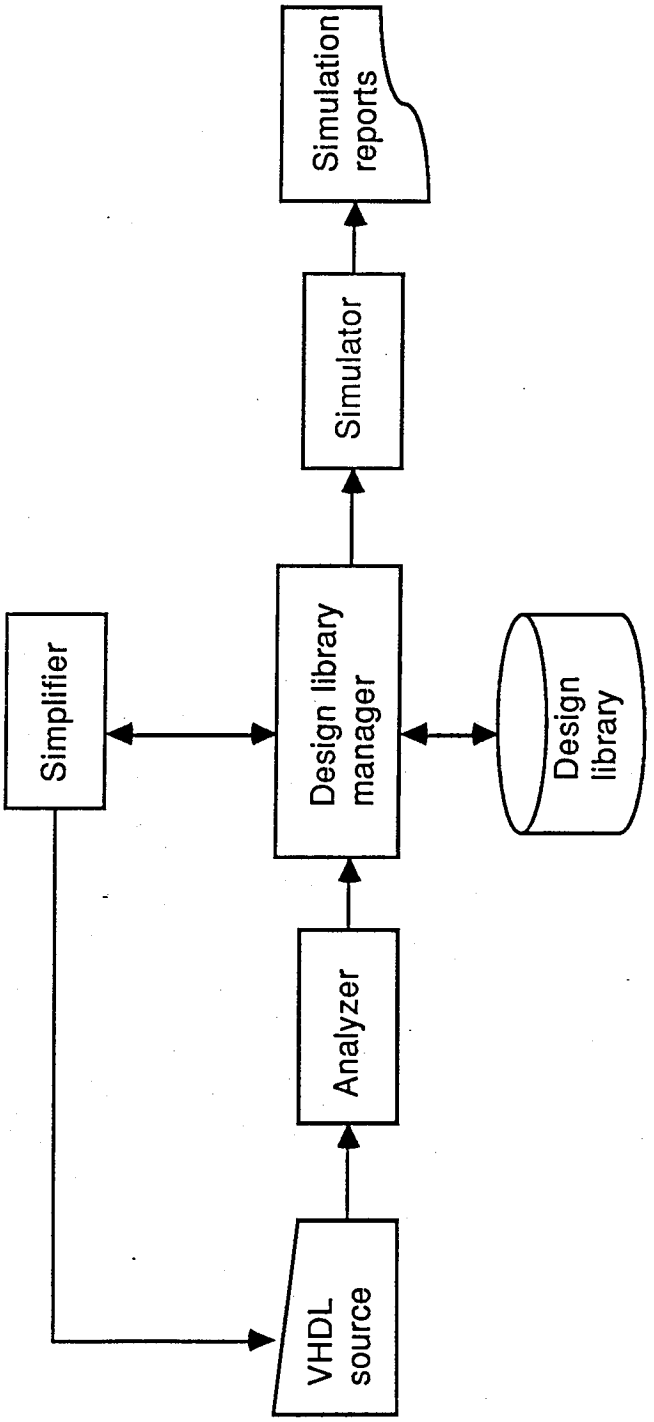


Figure 2. Block Diagram of VHDL Support Environment

## REFERENCES

[1] "Expert Systems for Science and Engineering", P.A.D. de Maine, M.M. de Maine and M.S. Wojtyna, in Tagungsband Analytiktreffen 1988, Editor: Tuebnerverlag, Chemische Gesellschaft der Deutschen Demokratischen Republik - 21 Pages (In Press)

[2] "SOLID: A High-Speed Data- and Question-Independent Information Management System", M.S. Wojtyna and P.A.D. de Maine, Proc. 1988 ACM 16th. Annual Computer Science Conference, 652-657 (1988).

[3] P.A.D. de Maine and D. E. Whitten, "Design for a Distributed Information Network", Advances in Information Systems Science 7, 1978, pp. 89-155.

[4] "Problems of Hardware and Software in Chemical Information Systems", P.A.D. de Maine and D.M. Head, Anal. Chim. Acta (In Press).

[5] Allen Dewey and Anthony Gadient, "VHDL Motivation", IEEE Design and Test of Computers, 3, No. 2, April 1986, IEEE Computer Society, New York, NY, pp. 12-16.

[6] J.D. Nash and L.F. Saunders, "VHDL Critique", IEEE Design and Test of Computers, 3, No. 2, April 1986, IEEE Computer Society, New York, NY, pp. 54-65.

[7] J.H. Taylor, R. Waxman, and C. Scarratt, "VHDL-Feature Description and Analysis", IEEE Design and Test of Computers, 3, No. 2, April 1986, IEEE Computer Society, New York, NY, pp. 17-27.

[8] Roger Lipsett, Erich Marschner, and Moe Shahdad, "VHDL-The Language", IEEE Design and Test of Computers, 3, No. 2, April 1986, IEEE Computer Society, New York, NY, pp. 28-41.

[9] Alfred S. Gilman, "VHDL-The Designer Environment", IEEE Design and Test of Computers, 3, No. 2, April 1986, IEEE Computer Society, New York, NY, pp. 42-47.