Managing the Printed Circuit Board Design Process

Tomas Blain
Michael Dohler
Ralph Michaelis
Emran Qureshi

Mitel Corporation
Kanata, Ontario K2K 1X3

ABSTRACT

This paper discusses an engineering design management system that integrates printed circuit board design data into a database supported by a relational database management system In contrast to other reported systems that manage design data only, the system described facilitates design process management by enforcing procedural constraints inherent in the design process itself Design process constraints can be dynamically altered to accommodate changes in the design process By combining design data and process management, the system creates an integrated design environment

The design management system described is a menu-driven, interactive, multi-user system that integrates engineering workstations, layout devices, automatic test equipment, and numerically-controlled manufacturing tools into a common design database The system has been in use at Mitel locations in Caldicot, Wales and Kanata, Ontario since October of 1984

INTRODUCTION

Historically, design data has been transferred between design tools by dedicated file translation programs Because design data is duplicated in a multitude of files, design changes are difficult to propagate in a controlled manner The integration of new design tools requires additional translation programs that further add to the number of design data files

Clearly, design data must be integrated Merging design files reduces data redundancy but increases the complexity of the data organization [SIDL80] Because programs are data dependent, changes to the data organization must be accommodated by changes in the programs

A better solution is to integrate design data in a common design database [EAST80, BENN82, KATZ83] When design data is integrated, changes can be effectively propagated to ensure the consistency of the design Application programs access data for design tools using the standard interface of the database management system (DBMS) The DBMS is responsible for the storage and retrieval of design data, the resolution of concurrent user access, and the recovery of the database after a system crash Dedicated links between design tools are replaced by exclusive links to the DBMS Hence, new design tools can

be introduced or removed from the design process without affecting existing tools

To support engineering design, the database is maintained by a *design data management system* A design data management system controls access to the database, supports design versions and alternative representations, and enforces design integrity constraints Several such systems have been reported [ROBE81, BENN82, KATZ82]

However, design data management systems do not impose procedural constraints on users Application programs access design data on an ad hoc basis Although concurrency is guaranteed, no attempt is made to ensure that design updates occur in an expected manner Because procedural constraints are not enforced, the consistency of design data across different design tools cannot be assured For example, such a system is unaware that the design description stored at an engineering workstation will be inconsistent with a design in the database after the design has been updated by a layout device Furthermore, design progress is difficult to monitor

A *design management system* facilitates design data and process management Information about the design process is maintained by the system Procedural constraints are enforced to ensure that updates to a design occur in an expected manner To accommodate changes in the design process, the system provides a means to dynamically change procedural constraints By enforcing design data and process constraints, the system provides an integrated and controlled design environment

The following sections discuss the design management system used to manage the printed circuit board (PCB) design process at Mitel Corporation

SYSTEM ORGANIZATION

The design management system is an interactive, menu-driven, multi-user system

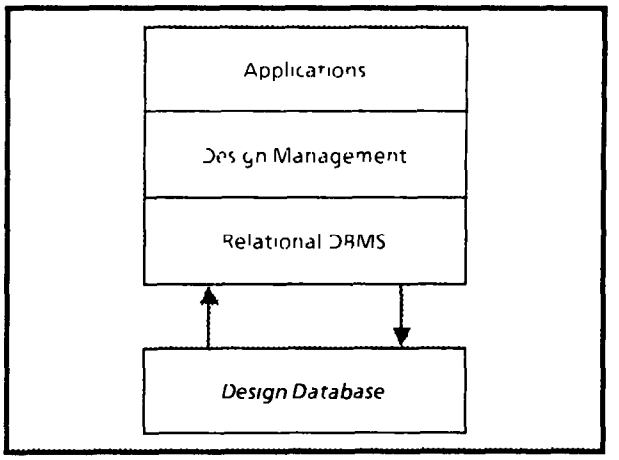


Figure 1 System Organization

partitioned into three functional layers (fig 1) The outermost layer, the *applications layer*, consists of modules that interface design and manufacturing tools to the design database The middle layer, the *design management layer*, contains all system modules, including the *system librarian*, the *component librarian*, and the *design librarian* The librarians maintain all the data in the database using permanent system relations called *catalogues* Users navigate through menu hierarchies to invoke applications and user facilities provided by the three librarians

The system librarian maintains user accounts, controls system access, initiates automatic database journalling and check-pointing, and allows a system supervisor to easily perform specialized database maintenance tasks

Component data is stored in the component library, a master parts list maintained by the component librarian The component library contains detailed electrical, physical, mechanical, and purchasing information for each component

The design librarian controls access to designs stored in the common design library To access a design, applications issue a series of standard function calls to the design librarian The design librarian thereby insulates applications from the mechanics of design data management

The innermost layer is the relational database management system The use of relational database management systems for computer-aided design and manufacturing (CAD/CAM) has been investigated extensively [HASK82a, HASK82b, HAYN83] Relational systems allow clients to access data without concern for file structures or access methods In the design environment, access paths to the data are known in advance and hence can be optimized [GUTT84] Applications and system modules access data using high-level, non-procedural query language commands embedded within the source code

The Design Object

Each design is described in a unique set of relations in the design library The *parts* relation describes each component and its placement on the PCB The *connectivity* relation describes the interconnection of components from the logical (schematic) and physical (layout) perspective An inventory of the graphics files that comprise the schematic documentation on the engineering work-station are contained in the *pages* relation Other design relations contain device-specific data required for simulators, testers, layout devices, and manufacturing tools

Users create designs using the design definition facility of the design librarian A *private* design is created by a user and resides in his personal workspace A *public* design is created by a design supervisor for a design team and exists in a collective workspace Public designs are intended for production, private designs are not Private designs allow a designer to explore the viability of alternative design implementations without affecting the production (public) design Design data can be copied and compared between private and public designs

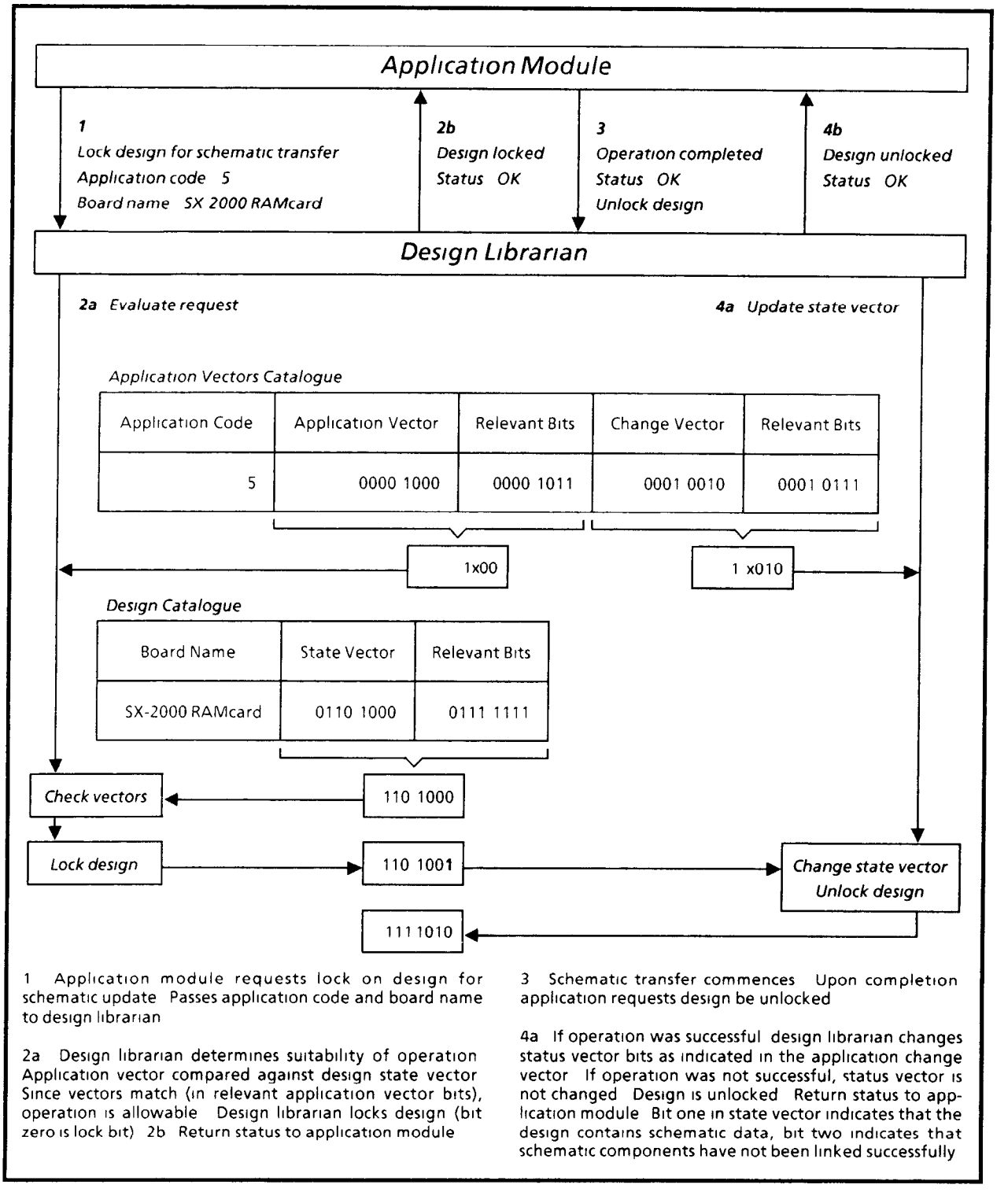
The *design profile* is created when a design is defined The design profile provides a detailed description of a design For a public design, the design profile identifies the product, product manager, design supervisor, and each designer and his tasks A project manager can use the design profile to determine the resources committed to a design effort The design profile also ensures the privacy of communications between design and product groups Most importantly, the design profile facilitates design protection by explicitly partitioning the user community into design groups and defining the design privileges and tasks granted to each user

Like public designs, private designs can be profiled in a similiar way However, private designs cannot be used in operations reserved for production designs (eg layout) The design librarian checks the design profile before allowing a design operation to proceed For example, a designer may be permitted to enter schematic data into the design but prohibited from attempting design layout

DESIGN PROCESS MANAGEMENT

To further effect design management, the design librarian enforces procedural constraints inherent in the design process The design *state vector*, which describes the current state of design development, must match the design tool *application vector* before an operation is allowed to proceed (fig 2) The design state vector is a catalogue attribute in which each bit corresponds to a particular design event or condition For example, the state vector may indicate that a design is locked, contains incomplete component specifications, or that a local copy of the design on an engineering work-station may be inconsistent with the design in the database To allow flexibility in the design process and the quick integration of new tools, design tool application vectors can be interactively modified by privileged users Changes in the design process do not require changes to the design librarian since the procedural constraint mechanism is data-driven

An update to a design is allowed to proceed if the user is permitted to perform the operation (as specified in the design profile) and the operation is allowable at the current state of



1 Application module requests lock on design for schematic update Passes application code and board name to design librarian

2a Design librarian determines suitability of operation Application vector compared against design state vector Since vectors match (in relevant application vector bits), operation is allowable Design librarian locks design (bit zero is lock bit) 2b Return status to application module

3 Schematic transfer commences Upon completion application requests design be unlocked

4a If operation was successful design librarian changes status vector bits as indicated in the application change vector If operation was not successful, status vector is not changed Design is unlocked Return status to application module Bit one in state vector indicates that the design contains schematic data, bit two indicates that schematic components have not been linked successfully

Figure 2 Example of data-driven enforcement of procedural constraints by the design librarian

design development (as specified in the design state and application vectors) If the design is locked, the locked-out user is supplied with information about the locking user and operation Users may optionally queue design lock requests if the design is locked at the time of the request or send a message to the locking user's terminal If the design is unlocked, the design librarian locks the design on the user's behalf before allowing the operation to proceed When the operation completes, the design is unlocked, the design state vector is updated, and a design audit trail is updated to allow designers and management to monitor the implementation effort

THE DESIGN ENVIRONMENT

The PCB design environment supported by the current version of this system is shown in figure 3 Application modules link the various design and manufacturing tools to the design database In the following discussion, design and manufacturing tools will be referred to in generic terms rather than by the product names shown in figure 3

Schematic Capture

At the conceptual stage of design development, the *design browser* can be used to enter a basic parts list into a private design to determine board power dissipation, current source/drain, and projected cost to help in selecting from among alternative design implementations

When a particular implementation is decided upon, the designer enters the schematic description on an engineering work-station Functionality may be verified with timing verification and logic simulation checks at the work-station The schematic description file (called the *netlist*) may then be copied from local storage to shared storage on a multi-user based computer system

The designer may then invoke the design management system and request that the schematic data be transferred into a design in the database If the operation is permitted by the design librarian, a schematic lock is placed on the design on behalf of the designer The contents of the public or private design relations are erased prior to being loaded with the schematic description Although the design librarian does not explicitly allow versions, a designer can easily create private designs to contain design versions if need be

The transfer of design files into the database involves thousands of individual database operations However, each design-level transaction commits as a single, atomic database transaction A user can abort a database operation at any time without compromising the integrity of the data Partial transaction commitment cannot occur For example, if a designer aborts the entry of schematic data into the database, no changes are made to the design relations and any outstanding locks are released

The next step in the design process requires the designer to *link* (join) the logical components on the schematic (stored in the design parts relation) with the parts master contained in the component library A component either finds a unique match, a non-unique (multiple) match, or no matches at all For layout, each component on the schematic must have a unique *shape* (physical characteristics) name in the component library For simulation, each component must have a unique *model* name in the component library

If a component has multiple matches, the designer may select the desired component from a list of acceptable matches using the design browser The design browser maintains the relationship throughout subsequent schematic updates The schematic designer can also use the design browser to edit existing design information rather than repeating the schematic transfer procedure In this case, the design must be re-linked to insure unique component specifications

Design Layout

The designer can release the schematic lock after each component on the schematic finds a

unique shape name in the component library The layout designer may then place a layout lock on the design, prohibiting further updates by the design engineer The design engineer may continue revising the schematic but only in private designs in his personal workspace Only updating by the layout designer is permitted to the public design

The layout designer, at the request of the schematic designer, may change the components used in the design using the design browser Updating via the design browser is a cost-effective alternative to repeating the schematic capture to layout procedure If such changes are made, the design must be re-linked prior to extracting schematic information for layout

To transfer schematic information to the layout device, a schematic description file is generated In the file, logical components on the schematic are related to device-specific shape names extracted from the component library If a shape is used that does not have a corresponding part number, the component engineer and the layout designer are warned about the new component However, the layout of the design can begin before the new component is formally approved and ordered

When the layout is complete, the layout device generates a file describing the physical implementation of the logical design captured on the schematic for transfer to the design database If new components were added during layout, the layout designer must enter the specifications for each new component using the design browser If necessary, the designer can also edit the specifications of existing components in the design In both cases, the design must be re-linked with the component library to insure that each new component is uniquely described in the library and that the editted specifications have not altered the layout characteristics of the component If the layout characteristics have changed, the system warns the designer that an inconsistency exists between the layout device and the design in the database The designer must lay out the board using the new component specifications and repeat the transfer procedure

After the design in the database is updated with the physical design description, the layout lock on the design is released However, the design librarian will not allow schematic updates to the design until the schematic designer verifies that the design description stored locally on the work-station agrees with the description contained in the database

The designer must extract the updated design information from the database and apply the changes to the current schematic description file via the *back-annotation* module Differences in the connectivity and parts list must be manually annotated on the schematic When the connectivity and parts list are equivalent, physical characteristics acquired during layout are automatically added to the schematic description file The design librarian then unlocks the design Hence, even though a user can *release* a design lock, only the design librarian can *unlock* a design

Revision Control

When the layout lock is released, the design librarian copies the public design into a read-only *revision* design If a previous revision exits, the design librarian determines the differences between revisions and appends the changes to the current revision If the previous revision contains revision change information tuples, these too are appended to the current revision design The design librarian then destroys the previous revision The revision design is used to generate design files for automatic test equipment and numerically-controlled manufacturing tools

In a manufactured printed circuit board, the final component selection is typically determined by preference, packaging, availability, cost, or manufacturing requirements The manufacturing engineer may substitute part numbers in the revision design using the

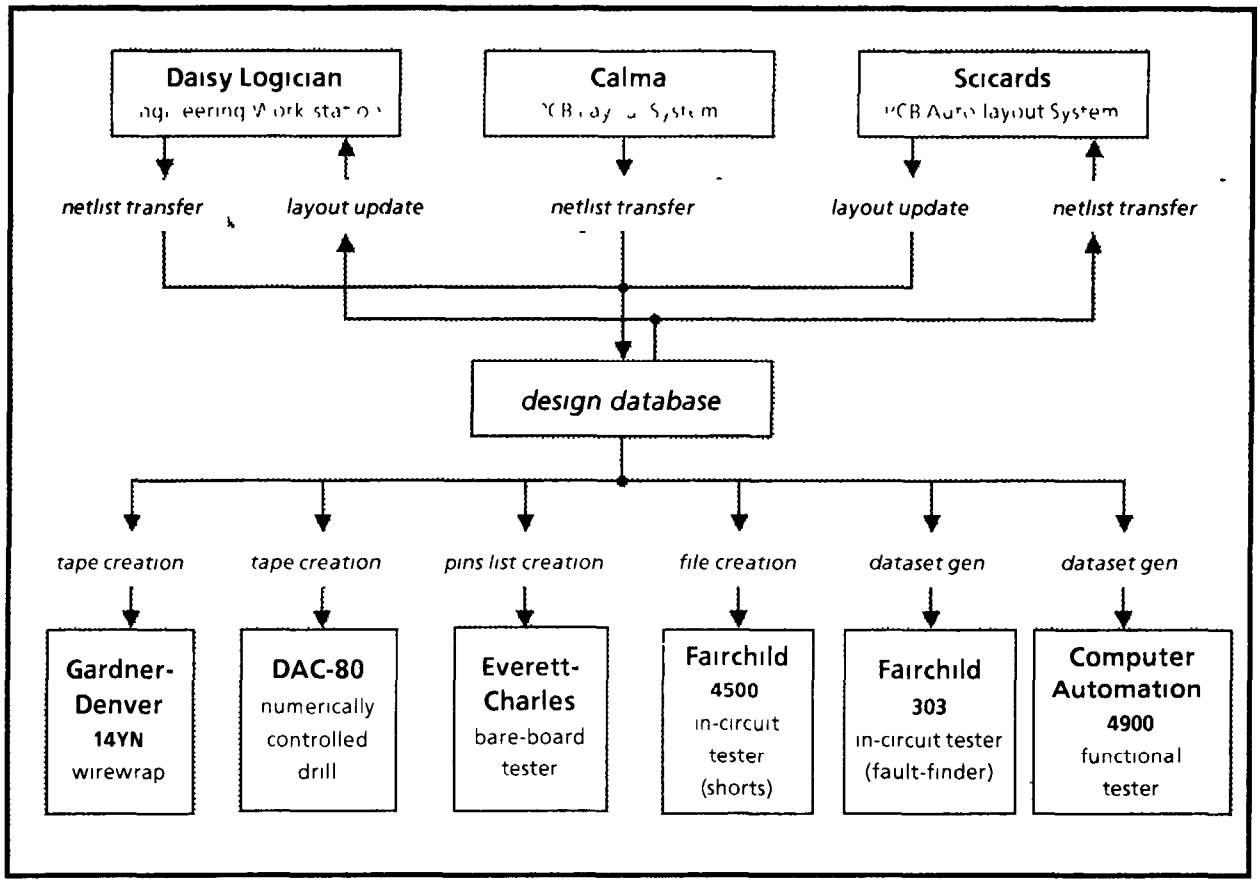


Figure 3 Mitel Printed Circuit Board Design Environment

revision design browser Substitutions are recorded in a separate relation to ensure a record of the changes made in manufacturing If component substitutions are made, the revision design must be re-linked to the component library before being released to manufacturing

Because all revision changes are logged, the design librarian can reconstruct any previous design revision by recursively applying the change tuples to the current revision design On-line revision recall is the preferred alternative to complete design archival on secondary storage media Recalled design revisions can only be written into private designs to ensure that the public design remains consistent with the schematic data stored locally on the work-station

The design librarian archives the design *configuration* - the complete description of the design for a specified release [BENN82] The design configuration consists of the revision design relations and any non-tabular representations of the design, like schematic graphics and documentation text files, not stored in the design database but specified in the design profile Facilities are provided to archive and recall data files as integral design representations

PERFORMANCE CONSIDERATIONS

In designing the system, one of the major design goals was to achieve optimal performance for interactive users [BLAI85] To this end, three DBMS bottlenecks were identified design data

input/output, query evaluation, and run-time integrity and security mechanisms

Rather than appending a tuple at a time, design data is *bulk copied* directly from pre-formatted temporary files built by each design tool application module Queries are compiled to reduce the overhead of parsing at run-time [KATZ79, STON82] Finally, the integrity and security mechanisms of database systems, which degrade system performance and are too simple to be of use, are replaced with an efficient security mechanism aptly suited to the design environment

The performance statistics for a typical design are shown in Table 1 The printed circuit board design in the example contained 1390 connections between 506 schematic components A designer interactively resolved 86 multiply-matched components into 36 components uniquely recognized by the component librarian The operating environment was a VAX-11/780 under VMS (a migration to a UNIX-based operating system is planned) The system software contains approximately 19 000 lines of VAX-11 C source code including embedded query language commands to the INGRES relational database management system [RTI84]

Operation	CPU Time (sec)
Design Creation	0 09
Logician to Database	0 39
Component Link	1 57
Design Browser Update	1 20
Database to Scicards	0 43
Scicards to Databases	1 29
Database to Logician	0 40
Design Consistency Check	0 31
Database to Mfg Tool	0 15

Table 1 CPU time for selected operations for example design

THE RELATIONAL DBMS

Because they lack transitive closure operators [GUTT84], relational database management systems are unable to easily model the hierarchical approach used to manage the complexity of a VLSI design The partitioning of a VLSI design into interdependent *cells* requires that the dependencies among the cells be explicitly defined [KATZ83] A VLSI design data management system must use these cell dependencies to reconstruct the *flat* representation of the circuit before effecting cell design changes

Because printed circuit board design is typically less complex than VLSI design, an ability to represent design hierarchies is less of a concern in PCB design A system can instantiate the flat representation of the PCB design directly in database relations Said another way, the *object* manipulated in a VLSI design system is the design cell, in a PCB design system it is the design itself [LORI83, PLOU84] Hence, design-level locking mechanisms are acceptable in the PCB design environment, in the VLSI design environment they are not The reduced complexity of a PCB design also implies that the database I/O bottleneck that occurs while accessing design data is not as critical a problem for a PCB design system as for a VLSI design system

The benefits of using a commercial relational database management system in the design environment are apparent [ROBE81] The cost and expertise involved in designing and implementing such a system would be restrictive to most design support groups Commercial systems are powerful, portable, proven, and supported Further, many of the earlier criticisms of commercial database management systems [SIDL80] are no longer issues

However, relational database management systems could be improved for use in the design

automation environment by providing user-defined *abstract data types* [STON83] and non-tabular data storage

User-defined abstract data types (ADTs) would allow design environment semantics to be managed by the database management system rather than by an application program which is currently the case ADTs would provide the ability to represent design objects and establish design constraints from within a database management system and would further simplify the implementation of object-level check-in/check-out facilities

As well, graphical descriptions and documentation could be stored within the database if the relational system supported non-tabular data storage Currently, non-tabular data is stored in data files maintained by the design librarian Such enhancements would extend the functionality of relational database management systems, and provide greater support for applications within the realm of engineering design data management

CONCLUSION

A design management system creates an integrated design environment by combining design data and process management The enforcement of design process constraints facilitates process management by fostering communication and cooperation among design team members, ensuring that illegal operations cannot proceed, warning designers of possible error conditions, and allowing management to monitor the implementation effort A relational database management system can provide efficient and reliable storage for PCB design data

ACKNOWLEDGEMENTS

The authors wish to acknowledge the invaluable contribution of the other members of the design team, namely Stephen Grant, Pamela Harrison, Carl Robinson, and Don Wells We are especially thankful to Larry Rowe and François Vernadat for their comments on an early draft of this paper

REFERENCES

[BENN82] Bennett J *A Database Management System for Design Engineers*, Proc of the 19th Design Automation Conference, 1982

[BLAI85] Blain T , Qureshi E *Maxims for Database Design and Performance Optimization*, Internal Memorandum Mitel Corporation March 1985

[EAST80] Eastman C *System Facilities for CAD Databases*, Proc of the 17th Design Automation Conference, June 1980

[GUTT84] Guttman A *New Features for a Relational Database System to Support Computer Aided Design* Memorandum No UCB/ERL M84/52, Electronics Research Laboratory University of California Berkeley CA June 1984

[HASK82a] Haskin R , Lorie R , *On Extending the Functions of a Relational Database System*, Proc ACM Sigmod Conf , 1982

[HASK82b] Haskin R , *Using a Relational Database System for Circuit Design* IEEE Database Engineering, Vol 5, June 1982

[HAYN83] Haynie M , *Tutorial The Relational Data Model for Design Automation* Proc of the 20th Design Automation Conf 1983

[HAYN84] Haynie M , Gohl K , *Revision Relations Maintaining Revision History Information*, IEEE Database Engineering, Vol 7, June 1984

[KATZ79] Katz R *Performance Enhancements for Relational Systems through Query Compilation* Proc AFIPS National Computer Conf , 1979

[KATZ83] Katz R , *Managing the Chip Design Database* IEEE Computer Dec 1983

[KATZ84] Katz R , Lehman T *Storage Structures for Versions and Alternatives* IEEE Transactions on Software Engineering March 1984

[LORI83] Lorie R , Plouffe W *Complex Objects and Their Use in Design Transactions* Proc ACM Sigmod Conf Databases for Engineering Design 1983

[PLOU84] Plouffe W Kim W Lorie R McNabb D *A Database System for Engineering Design* IEEE Database Engineering Vol 7 June 1984

[ROBE81] Roberts, K , Baker T Jerome D , *A Vertically Organized Computer-Aided Design Data Base* Proc of the 18th Design Automation Conf 1981

[RTI84] *INGRES Reference Manual,* VAX/VMS Version 3 0, Relational Technology Inc , Berkeley CA October 1984

[SIDL80] Sidle T *Weaknesses of Commercial Data Base Management Systems in Engineering Applications* Proc of the 17th Design Automation Conf 1980

[STON82] Stonebraker M , Woodfill J Ranstrom J Murphy M Meyer M Allman E *Performance Enhancements to a Relational Data Base System* Memorandum No UCB/ERL M81/62 Electronics Research Laboratory, University of California, Berkeley CA Sept 1982

[STON83] Stonebraker M Rubenstein B , Guttman A , *Applications of Abstract Data Types and Abstract Indices to CAD Databases* Memorandum No UCB/ERL M83/3 Electronics Research Laboratory University of California Berkeley, CA Jan 1983

TRADEMARKS

SX 2000 is a registered trademark of Mitel Corporation

LOGICIAN is a registered trademark of Daisy Systems Inc

SCICARDS is a registered trademark of Scientific Calculations, Inc

VAX and *VMS* are registered trademarks of Digital Equipment Corporation

UNIX is a registered trademark of Bell Laboratories

INGRES is a registered trademark of Relational Technology, Inc