CADTM: A Database Transaction Manager for Coordinating Design Activities in CAD Environments

Haeng Rae Cho and Songchun Moon

Department of Information and Communication Engineering
Korea Advanced Institute of Science and Technology
207-43, Cheongryang, Dongdaemun, Seoul 130-012, Republic of Korea

This paper presents a new database transaction manager, CADTM, for coordinating design activities in CAD environments. Unlike the previous approaches for coordinating design activities, in which each designer faces complexity of a design activity, CADTM encapsulates the complexity of design activities from designers. This is achieved by the notion of transaction template which guides the designer not to produce incorrect design results due to misunderstanding of complex design activities, and by the notion of interleaving specification which enables the designer to cooperate with his/her group members in a consistent way. In CADTM, therefore, any designer does not need to know the details of the design activity, and also does not need to concern whether he can release intermediate results of a transaction.

Keywords: CAD databases, transaction management, intelligent user interface

1. INTRODUCTION

In computer aided design (CAD) environments, a design activity represents an abstract design function required to achieve a specific design goal, which is performed by a designer. VLSI design projects, in particular, are usually too large and too complex to be completed by a single designer. It is a common practice that a VLSI design project is accomplished by a number of design groups, and each designer cooperates with his/her group members to perform his/her design activities [1], [2]. In VLSI design projects, therefore, a flexible and efficient mechanism for coordinating the design activities is the key to the success of the projects.

2. LIMIT OF CONVENTIONAL CAD TRANSACTION MODELS

In a CAD database system, design activities are represented as transactions and coordinating the design activities is modeled by a concurrency control scheme for the transactions [2]. While the notion of a transaction provides a good starting point for modeling design activities, it has a serious limitation in coordinating the design activities. This is due to the fact that the classical notion of a transaction fails to support the functionalities required in coordinating the design activities; the functionalities required are cooperation among the design activities within a group and coordination of the design activities between multiple groups.

A number of transaction models which extend the classical notion of a transaction have been proposed to coordinate the design activities [3]-[5]. The common approach used in those models is that user interactions are incorporated as a part of the transaction management. For example, in the group-oriented CAD transaction model [3], a designer is allowed to release design data at any time, and thus the design data are available to other designers' activities in the same group. The cooperating CAD transaction model [4], under the same approach, enables a designer to define and create a short-duration transaction (SDT) which performs a part of a designer's task; this implies that a designer's task is modeled as a set of SDTs, and the design data created by a SDT is
allowed to be accessed by the other designers’ SDT in the same group. The notification-based transaction model [5], under the same approach, alerts a designer of any conflict, for instance, an attempt to lock an object that has already been locked in an exclusive mode; this kind of conflict is resolved by negotiation between the two designers under the assumption that they are able to interact to resolve the conflict.

The main problem of incorporating user interaction in the transaction management in this manner is that human beings are error-prone, which could result in inconsistent data. Furthermore, the structure of a VLSI design activity is too complex for a designer to understand the details of the design activity. This is illustrated by Example 1.

**Example 1** Suppose a transaction $TR$ of a simple design activity which consists of five CAD tools. Figure 1 shows the graphical abstraction of $TR$.

![Figure 1 Graphical view of a transaction](image)

The output of $TR$ are $obj1$, $obj3$, and $obj4$ produced by $tool1$, $tool4$, and $tool5$, respectively. When some transaction in the same group wants to access $obj1$ and $obj3$, $TR$ allows the access of $obj1$ after $tool1$ is terminated because $obj1$ is assumed not to be updated in $TR$. The access of $obj3$, however, is prohibited until $tool4$ is terminated. This requires that users must know the details of the design activity, i.e., the input and output objects of each CAD tools, and output-to-input relationships among CAD tools. When a number of CAD tools are involved in a particular design activity and their interrelationships are complex, it is difficult for a designer to understand the details of the design activity and also to decide whether intermediate results can be released or not.

While there are some approaches to hide the complexity of a design activity, for example, EDA [6], Cadweld [7], and Ulysses [8], they concentrate only on modeling single design activity using an intelligent user interface. That is, they do not address the issues associated with multiple design activities like concurrency control. Therefore, a transaction manager that incorporates the notion of a transaction into the intelligent user interface is a strong requirement for coordinating design activities.

### 3. TRANSACTION MANAGEMENT FOR COORDINATING DESIGN ACTIVITIES

We propose a new transaction manager called CADTM, which supports design transparency in coordinating the design activities. Unlike the previous approaches for coordinating design activities, in which designers directly face the complexity of design activities, CADTM hides the complexity and thus enables a designer to perform his/her design activities without acquiring detailed knowledge on them. This is achieved by the notion of **transaction template** which identifies CAD tools for the design activity and their invocation sequence, and by the notion of **interleaving specification** which represents the set of possible interleaving points where some other transaction is allowed to access the intermediate results of the transaction. Example 2 illustrates the transaction template and the interleaving specification for transaction $TR$ in Example 1.

**Example 2** In the transaction template, five CAD tools are identified by declaring their input and output. For example, the input of $tool2$ is $obj2$ and the output of $tool2$ are $obj3$ and $obj4$, while $obj2$ is produced by $tool1$. The invocation sequence of the transaction template is specified by language constructs: sequential (separated by ;), conditional (if-then-else), iterative (while-do), and parallel (parbegin-parend) statements. With the language constructs, the invocation sequence for Example 1 is specified as follows:

```
parbegin {{tool3; tool4}, (tool5)} parend
```
An interleaving point of \( TR \) is a pair of \(<TR, obj>\) where \( obj \) is the output object of \( TR \). The interleaving point \(<TR, obj>\) is created just after \( obj \) is updated at last in \( TR \). For example, \(<TR, obj1>\) is created after \( tool1 \) terminates its execution. But \(<TR, obj3>\) is not created until \( tool4 \) is terminated. When \(<TR, obj1>\) is created, \( obj1 \) is allowed to be accessed by other transactions in the same group of \( TR \).

In CADTM, a transaction template is predefined by an expert designer and stored in the database. A transaction template is retrieved by a designer according to its functionality, the required input data format, and its output data format, each of which is stored altogether in a database. When a transaction template is retrieved, a designer can instantiate it just by assigning the actual arguments and options. The details of the transaction template are, therefore, hidden from the designer.

An interleaving specification is determined automatically according to the invocation sequence of a transaction template and actual arguments of the transaction template. With the notion of interleaving specification, CADTM supports various levels of cooperation among transactions. When two transactions, \( TR_1 \) and \( TR_2 \), are in the same group, \( TR_1 \) can access all design data of \( TR_2 \) which appear in the interleaving specification of \( TR_2 \), and vice versa. Transactions in the other groups, however, may not access the design data \( obj \) of \( TR_1 \) (or \( TR_2 \)) even though an interleaving point for \( obj \) has been created. The reason is that \( obj \) may be updated later by other transactions which are in the same group of \( TR_1 \) (or \( TR_2 \)). In order to support coordination of design activities in other groups, CADTM uses the notion of group interleaving specification. When a transaction group \( TR_G \) has \( n \) member transactions \( TR_i \), \( 1 \leq i \leq n \), the group interleaving specification of \( TR_G \) is set of \(<TR_i, obj>\) where \( TR_i \) updates \( obj \) at last among all member transactions. Therefore, a transaction can access the design data of some transaction in the other group \( TR_G \) when the group interleaving specification of \( TR_G \) includes that data. This procedure can be applied recursively to a transaction group hierarchy, in which a transaction group becomes a member transaction of a parent transaction group. When a transaction, \( TR_i \), wants to access the intermediate results, \( obj \), of the other transaction, \( TR_j \), which is in the different group, \( TR_j \) can access \( obj \) according to the following access protocol.

**Access Protocol for Different Groups**

**Proposition:**
1. \( TR_G \) is a least common ancestor transaction of \( TR_1 \) and \( TR_2 \) in the transaction group hierarchy.
2. \( TR_{G1} (TR_{G2}) \) is a member transaction of \( TR_G \) and an ancestor transaction of \( TR_1 (TR_2) \).

**Protocol:**

- \( TR_j \) can access \( obj \) when \(<TR_k, obj>\) is included in group interleaving specification of \( TR_{G2} \). \( TR_k \) is a member transaction of \( TR_{G2} \) which updates \( obj \) at last.

For example, suppose a design project, \( TR_{G1} \), which consists of two transaction groups, \( TR_{G1} \) and \( TR_{G2} \). Figure 2 shows a transaction group hierarchy of the design project.

According to the access protocol, \( TR_1 \) can access the intermediate results, \( obj \), of \( TR_3 \) whenever \(<TR_k, obj>\) is included in the group interleaving specification of \( TR_{G2} \). \( TR_k \) may be \( TR_3 \) or \( TR_4 \).

With the notion of the transaction template and the interleaving specification, CADTM can resolve the complexity problems associated with the previous CAD transaction models. The notion of the transaction template guides the designer not to produce incorrect design results.
due to misunderstanding of complex design activities. Furthermore, the notion of the interleaving specification enables the designer to cooperate with his/her colleagues in a consistent way. This is guaranteed by releasing data only if the interleaving point of that data have been created. Releasing data at the interleaving point is correct in the context of the transaction template and the transaction group hierarchy.

4. SYSTEM ARCHITECTURE OF CADTM

Implementation of CADTM includes three components: a transaction template manager (TTM), a transaction scheduler (TSCH), and a version manager (VM). The TTM has a role to store, retrieve, and instantiate a transaction template. We have chosen to use a relational database to store and retrieve transaction templates due to the convenience of using SQL. The TTM, therefore, provides an SQL interface. When a transaction template is retrieved by an SQL statement, the TTM executes the transaction template, which in turn results in executing component CAD tools according to the invocation sequence of the transaction template. The TSCH checks whether an input object of a CAD tool is available. If some transaction has updated the same object and the interleaving point for the object is not created yet, the designer of the requesting transaction is notified. When a designer is notified, he/she should decide whether to suspend the transaction and resume it later, to abort the transaction, or to find another version of the design object and make that version as the input object of the CAD tool. In order to support the consistency of versions, the VM has a role to maintain version history, configuration, and alternative versions [10]. A transaction releases a version of a design object at the interleaving point of the object and the version can be accessed by other transactions in a group.

5. CONCLUSIONS AND FURTHER STUDY

In this paper, we proposed a new database transaction manager, CADTM, for coordinating design activities in CAD environments. The major contribution of CADTM is to provide a facility for encapsulating the complexity of design activities from designers. Therefore, any designer does not need to know the details of the design activity, and does not need to concern whether he can release intermediate results of a transaction or not. Currently, we implement CADTM on top of the first-ever relational DBMS developed in Korea by KAIST, called Information Management (IM).

REFERENCES