The UNICAD Library Manager

Marco Carilli, Claudio Pasce
Central R&D, CAD Department – SGS-Thomson Microelectronics – Agrate Brianza, ITALY

Paolo Gubian
Dipartimento di Automazione Industriale – Università degli Studi di Brescia – 25060 Brescia – ITALY

Abstract

This paper presents a system which adds design management capabilities to an existing CAD environment composed of both internally developed and externally acquired tools. The Library Manager is composed of a set of services provided to CAD tools and to end users; tools may be either encapsulated or modified to make use of the services. Motivations for choices made in the implementation are justified by the design principles adopted. Finally, examples are provided for tool integrations of all types.

1. Introduction

Flexibility in design requires the possibility to exchange parts of design, share common developments as well as to experiment with design variations in a design space intended for private use.

The Library Manager represents a new way of interacting with the data store which completely eliminates the use of explicit pathnames in the specification of design object names. It is presently part of the Unicad design system, the official CAD system of SGS-Thomson. Basically, what the user gets is a set of logical directories, containing design objects. The user can see the contents of libraries as well as perform standard operations on libraries and their elements. However, libraries are under the control of the Library Manager which checks permissions before granting access to the user and logs each action after performing it.

In the first version, users can execute simple actions on design files through the Library Manager. These operations are creation, deletion, checkin, checkout, copy etc. on design objects. The Library Manager provides a powerful versioning mechanism for design data. It is even possible to invoke some operations on design files, such as an edit message to a schematic contained in a library or a view message on a text file.

It is not yet possible to perform complex actions on library elements, such as simulate; to do this, a user would have to fetch a copy of the file(s) in a work directory. This is not due to a design limitation in the underlying design manager, but to lack of integration between some of the subsystems in the Unicad environment and the Library Manager. The Library Manager stores away the user's design files and tracks their versions as well as access and protections. The structure with which it stores the files is hidden from the user and from CAD tools, which have to use a procedural interface to access data instead of using the standard file system calls. As of this version, the main subsystems still access true files in the file system. Complete implementation of the design manager, in future versions of Unicad, will have the subsystems talk to the Design Manager procedural interface.

2. The DEDALO Design Manager

DEDALO is the name of the Design Management project going on in SGS-Thomson. The project is aimed at providing a full set of design management capabilities to the Unicad design system. ([1], [2]) The DEDALO approach is being implemented by means of several intermediate steps before reaching a complete solution. One of these steps is the provision of logical naming capabilities as well as location transparency over the network to the Unicad system, coupled with a sophisticated versioning mechanism and support for several relationships among design data. Supported relationships include derivation of a piece of data from other pieces, composition of a block at a higher level by means of blocks at lower levels and alternate representations of the same logical object.

The present version of DEDALO makes use of DEC's PowerFrame [3] as the repository of design data and relationships among design data. However, the DEDALO library manager looks and behaves differently from off-the-shelf PowerFrame Executive. The main features that DEDALO adds to PowerFrame are:

a) Multi-user access to workspaces. Locks are placed to prevent concurrent modifications.

b) Support for design groups, with per-group access permissions and protections. Granularity extends to the level of specifying which tool(s) may be run by each user on each piece of data.

c) Off-line execution of integrated tools, with notification by mail to the user. Integrated tools are those which directly talk to the Library Manager. Once they have been started they go on unattended, the Library Manager keeping track of what happens and putting data in the right places once the tool terminates.

d) Complete RPC library of procedures callable from a C program, or by means of a simple command interpreter. Procedures represent the atomic services provided by the Library Manager. For the Unicad system, which is based on the EDG design framework by Cadence Design Systems, another version of the RPC library exists written in SKILL [5], the Lisp-like customization language of Cadence.

e) A specialized server communicating to the Cadence or to any other client requesting services by means of a network-transparent protocol.

f) Security against misuse of data through a special implementation of ownership on top of the Unix protection mechanism.

3. Library Concepts

This section explains the main concepts related to library management. It is important to recognize that the conceptual model presented by the Unicad Library Manager conforms to the DEDALO system's data schema. A reader familiar with PowerFrame should not try to map PowerFrame's specific concepts onto the Library Manager's structures.
Libraries: the main entities recognized by the Library manager, are composed of design objects. Other entities are contained in libraries or compose libraries. The next subsection presents design objects and other entities that constitute libraries.

It is possible to create and maintain relationships of various kinds among library components. Some of the relationships are automatically tracked by the system, while others have to be tracked by design tools or by the user. Subsection 2 introduces the main relationships, while subsection 3 presents an account of the user interface in terms of commands available to the user.

3.1 Objects

Libraries

A library is a collection of design objects or cells stored in the same logical location. These objects are called units in the DEDALO terminology. It is possible to store many representations of the same unit in a library. It is possible to see which representations of a unit are available.

A library has a name and a location. The library name is the same name which is used by the designer in referring to it. The name of the library bears no relationship to its location. In particular, libraries do not change name when they are moved to another location by the administrator.

The naming scheme is enforced over a network domain, in a way similar to the Yellow Pages service in the Unix environment. Therefore, even if the library is moved to another network node, access remains the same through its name.

The location of the library is related to the directory on which data files and administration files, managed by PowerFrame reside. However, no user has direct access to the directories and every operation has to be carried out through the library manager. When a user creates the library, he/she may also specify a location option. It is thus possible to exercise some control over the location of the library, for example in cases in which the library could fit into several disk partitions. The location may be specified by means of a Unix path name or by choosing options on a set of dialog boxes. Exercising control over the location of a library does not mean that the user will be able to access the actual files that compose the library during normal use. The creator of the library is considered its administrator and is said to have administration privileges on the library.

The user specifies unit names when he wants to refer to them. These names are resolved by means of a mechanism which is very similar to the Unix path mechanism. Specifically, there is the notion of current library on which the user is working at each instant of time. A name specified at a certain point is searched for in the current library in force at that time. If the name is not found in the current library, a list of libraries is searched in order, thus providing a mechanism called hierarchy of libraries. The only difference between this and the directory path of Unix is that names that appear in the library path are library logical names, which are independent of their physical location. Special care has to be taken when the other CAD tools and framework are not aware of the existence of the Library Manager and have to find their data in the correct places. Usually, the library path is composed of a private library, a group library and of one or more public libraries which represent primitives and ASIC libraries with which the design is being built. Figure 1 shows an example of this mechanism.

Each library has a type: a library can be of type private, group or public. A private library is a repository of data for a single user/designer. Designers work as they are used to without the Library Manager. When they reach a more or a less stable version of a piece of design, they can store it in their private library. Each designer may have as many private libraries as he/she wishes. Private libraries provide a set of functionalities but do not enforce a particular style of usage. A designer may store data for different projects into different private libraries, or in the same library.

Private libraries provide versioning of the objects they store as well as checkin/checkout mechanisms. A designer may have the library manager track modifications that are taking place on the library data. Furthermore, a library provides support for hierarchical data in the sense of providing ways to traverse hierarchies and to give access to entire hierarchies to design tools.

A group library provides more sophisticated mechanisms to support the work of teams. A group library is created by any user who will become the group administrator. This user retains special privileges during all the library's lifetime. A list of group members may be specified and maintained by the administrator. Group members have access to the library according to the user privilege masks established by the administrator.

Basically, users may have read/write or read-only access to a group library. Read/write access implies the possibility of doing a checkin and a subsequent checkout of a piece of design. Locks are maintained by the library manager to prevent concurrent modification of an object.

Therefore, a group library provides a controlled way of sharing common developments among a group of designers. Typically, when a designer wants some of his/her data used by other designers, he/she will "migrate" those data to the proper group library.

A library of class public is a special library representing a checkpoint of a development library at a stable state or a company- or division-wide library such as libraries of ASIC primitives. A public library is read-only, and only administrators have the privilege of creating one. An administrator is a special username who has full access to the system data and functions.

A public library is created by the administrator. It can be populated with design objects by the same administrator and is considered read-only by all the other users. It is particularly well suited to contain stable versions which are ready to be distributed to the design community. For efficiency reasons, it is possible to have multiple copies of the same library stored on different nodes of the network. The library manager duplicates administration operations on all the copies.

Units

The basic data object stored in a library is the unit. A unit supports a full structure to store representations and tool run outputs. In the present version, tools are not integrated with the library manager, and thus the main purpose of a unit is as a repository for representations of design objects.

Each unit has a name, which has to be unique in a library. Unit names may be the same across different libraries. In case of multiple units with the same name, the library path mechanism chooses the one in the first library found on the search path.

Representations

Each unit may contain different representations of a design object. Representations should belong to a list of supported representations which are part of the system. The reason of having system-defined representations is that integrated tools are aware of the structure of supported representations.

Figure 1 – Library Path Mechanism

All representations known by the Library Manager are supported in terms of basic services like versioning, access control, concurrency etc.
In addition to these services, selected representations are said to be fully supported when more sophisticated services are provided for them. However, to obtain full support for a representation, negotiation with the design manager development team is needed. For example, if the representation is hierarchical, the Library Manager includes a routine which is able to traverse the hierarchy in the representation's format and is able to provide this service to any interested client (tool or user).

**Versions**

A versioning mechanism is provided for representations inside units. The versioning model is the standard version tree model supporting evolution and alternatives [4]. The version numbering scheme for the same unit/representation is independent inside different libraries. At any given instant, there is a current version which represents, in many cases, the most up to date version. It is possible to have the library manager move the currency pointer automatically any time the user checks in a version, or to move the currency pointer manually by means of a command.

**Parcels**

Parcels are special objects which exist outside of libraries and provide a means of transmitting the contents of libraries between far away sites running the Unicad system. An administrator may create a parcel of a library or parts thereof. The result of parceling is a shell script and a file which contains the administration information necessary to recreate the library. By running the procedure the administration file together with the necessary data files are put on a tape. Specifically, the parcel command creates the parcel file, and a parceling shell script. The user would run the script specifying the output unit on which a full archive should be created. At the receiving site, the local administrator would extract the first file from the tape, and run it inside the command line interface to the library manager. In this way, the entire library is recreated with versions and relationships in place. It is possible to produce parcels which pack entire libraries or to explicitly indicate units and/or configurations to be parcelled.

**Users**

A user has a name and a profile stored in the library manager. There is a true login procedure at the time of first connection to the server. A user profile specifies privileges attributed to the user. A special privilege allows a user an administrator level, providing full access to workspaces, as well as privilege to create and perform operations on public libraries.

**Groups**

Groups are the entities which access group libraries. A group is a list of user names maintained by the group library administrator. For each group, the administrator has the added responsibility of maintaining the list of authorized users for that group library. There are two kinds of users: read/write and read only users. Read/write users participate in the design done by the group, and can make modifications, after having reserved them, to the units stored in the group library. Read only users instead can make use of the developments done by the group but cannot modify any piece of design.

**3.2 Relationships.**

There are several relationships that may exist among pieces of design inside a library. Some of them are understood and maintained by the library manager. Others are defined but, due to lack of complete tool integration, have to be modified and maintained by users. Administrative users have also the option of defining new relationships giving them the meaning they prefer, and have a limited support from the system in their maintenance.

**Version derivation.**

Successive versions of the same representation of the same unit are related by the version derivation relationship. It is possible, given a version, to retrieve the parent version from which it was generated and the version(s) that have been derived from it.

**Equivalent representation.**

For representations supported by the system, there is a limited support to the notion of equivalence of different representations of the same unit. For example, a particular version of the layout representation may be said to be equivalent to the corresponding version of the schematic representation. Actually, equivalence is a very delicate concept and in many cases equivalence is checked through verification and by running. In many cases, a better relationship to represent, especially in the case of synthesis tools, is derivation. If a version of a representation has been produced from a version of another representation by running a tool, this can be explicitly stored in the database.

**3.3 Operations on Objects.**

There are a number of operations that can be performed on objects from the library manager. These operations are invoked by pop-up menus in the standard Cadence style. Communication to the server over the network is transparent to the user.

**Checkin / Checkout.**

Checkin is the operation of inserting a new version of a representation in a library. Before checking in the user should have checked the object out, which prevents concurrent modifications; the user should also have the privilege of doing the checkin in the library. When a user checks out a representation, a local copy is produced and a lock is placed on the representation in the library so that other users are not allowed to make modifications to it, unless they decide to start a branching path in the derivation tree. A new version slot is reserved for when the representation will be checked back in. Checkin and checkout are complex operations that can take place from/to private or group libraries. When a checkout form a group library is done, it actually takes place between the group library and the current private library. The library manager creates the unit in the private library and inserts the representation checked out from the group library. It then checks out this representation in the working directory of the user. The user can check the representation in the private library or in the group library. When a checkin is done in the private library the lock in the group library is not removed. This allows more development versions than stable versions. When finally the user checks the unit back in the group library, many checks are performed before accepting the action. For example, it is not possible to check in a cell to a group library if all the master of cells instantiated in it are not already existing in the group library. Finally, it is possible to cancel a checkout previously done and remove the lock on the representation so that other users can check it out.

**Hierarchy Management.**

For a limited set of representations (Cadence schematic and layout among them), the library manager supports representation and retrieval of hierarchical relationships. It is also possible to traverse a hierarchy from inside a library. For the special case of netlisting, it is possible to obtain netlists directly from data inside libraries. Two important benefits from this integration of the netlister are:
The current mark specifies the "most interesting" version from the development and usage point of view. The administrator has the possibility of moving these marks to whatever version is desired. It is also possible to specify that, for example, currency has to be moved forward each time a new version is checked in.

Parcelling a group of objects for transmission.

The user can create parcels composed of sets of objects for transmission to a remote design site running Unicad. The system keeps only a log of the fact that a parcel has been created in the log file. There is no scripting facility, for example, to rebuild a parcel with data previously specified for another parcel.

Change location of a library.

With the proper permission, an administrator has the option of moving the location of a library to another network node/disk partition. By definition of library manager, such an operation is transparent to the users of the library.


In this paper, the main concepts underlying the Unicad Library Manager, together with a sketch of the data schema implemented, have been presented. Nothing has been said about the user interface, apart from the list of available commands. At least three user interfaces have been implemented on top of the Library Manager, one for each of the operating environments in which it is needed: a textual interface, a Cadence-based menu-driven interface, and an X/Motif-based interface. Commands coming from all the user interfaces may be internized. Also, the use of DEC's PowerFrame is particularly original and has overcome some of the most important limitations of this system.

The work continues, both in the direction of making the Unicad tools aware of the Library Manager, thus directly p and depositing data from/to the repository, and in the direction of improving the data schema and/or exploring new technologies for the representation/management of the relevant relationships.

5. References