KEEPING THE PROJECT ON TRACK

The ASIC design project is powering ahead with different teams working to produce the final product. How do you ensure designers stay up to date with different releases and versions of modules, and maintain information about the status of reported bugs? **Srinath Anantharaman** offers some advice.

The core of a design project is a repository of thousands of design files. Within the same collection are other files that contain test parameters and data, test results, constraints data, program initiation data, even files for reported bugs. These files change continuously as the project develops.

Even if the device is small and all the design engineers are on a single site, maintaining such a file collection, keeping track of different file versions and the different release versions, synchronising different collections of files and maintaining a management view on progress, presents severe challenges.

Large designs, such as SoCs, implemented by design teams in different locations, exacerbate the problem. Using a wrong version of a file or not knowing about changes in other areas may delay or even destroy a project.

To manage this problem, new tools have been developed. Starting from the basic role of version control, keeping track of the different versions of a design file, tools have evolved into overall project data management (DM) systems that provide detailed information on the overall progress of the project.

TODAY'S SOLUTIONS

The first set of solutions use public domain products such as the Unix RCS (Revision Control System), SCCS (Source Code Control System) or CVS. They normally require significant effort in writing scripts to meet the needs of specific projects and do not normally have a graphical user interface (GUI).

Then there are products designed primarily for software developers and adapted for use in the hardware environment, such as ClearCase. Integrated design suites may have 'free' tools, such as VersionSynch. Finally, there are tools created specifically for the hardware design environment, such as ClioSoft's SOS.

EVALUATION PROCESS

An evaluation is not undertaken in isolation; it has to be related to projects. Before beginning an evaluation, it is vital to establish parameters for project size and administrative complexity. How big are the designs and how many people will be working on a specific project? Will they all be on a single site or working on several sites? What platforms will they be using and what design tools?

Then there are administrative issues. How much training is required before a DM system can be used, not just the designers, but also the team leaders and system administrators? DM systems should be easy to install, easy to set up at the start of a project, and not require dedicated resources for day-to-day management. The user interface must be close to intuitive, otherwise users will not use the system effectively. This implies a GUI, but a GUI that can easily be tailored to a specific project's requirements, that allows a user access to a command line interface and allows command lines to be inserted into scripts or Makefile.

BASIC NEEDS

The next evaluation stage should examine the match between the tool and the detailed requirements of an ideal DM system. Clearly, the basic requirement is revision control, keeping track of the different versions of a file (both binary and text) as changes occur. Revision control should extend to directories, to cope with any reorganisation that may take place within the project. Beyond that is release management, knowing which versions of the files were used for a particular release of the design. Ideally a system should allow roll back, recreating the entire configuration at a particular time, either to a specified time or date or when a snapshot was taken.

Does the system insist on opening only the latest version or

Checklist

In evaluating a data management environment, these are key points for a checklist:

Does the system offer:

Multi-site support?
Client server architecture?
Remote site caching?
Multiple developer coordination?
Safe repository?
Easy setup with no manual scripting?
Technical support?
Measurable overall cost of ownership?
Graphical user interface?
Command line scripting option?
Hierarchical display?
Graphical display of revision history?
Communication and collaboration icons?
Integrated bug tracking?
Overall project status data?
Audit trail and reporting?
Version control for files?
Version control for directories?
Snapshot to recreate earlier versions?
Roll-back to a particular development time/status?
Comparison of changes between two versions?

In Cadence environment:
Direct access from DFII and Library manager?
Work with cells and views?

does it allow a designer to bring an earlier version into the work-area? And can a designer write rules to define the versions to be used?

For larger projects it is increasingly common to develop a design on multiple sites and this may mean multiple platforms. Any design management system used for larger projects should be able to efficiently support multiple sites and provide indications to the users as to which files are in use or have recently been changed. ClioSoft provides an indication of status, such as 'meets timing' and a change summary provides an aide-memoir of the changes made. It is only when all these requirements have been satisfied that it makes sense to begin measuring performance.

Measurement should use realistic data: checking in and out thousands of files is not realistic, but checking in and out multiple sets of tens of files is better. Measurement should not be of a single user or a single site when the system will be working with multiple users and on multiple sites. Rapid updating and synchronisation is important. Disk space is another important measure. Design datasets and their supporting files are notoriously large and can rapidly become even larger if the way in

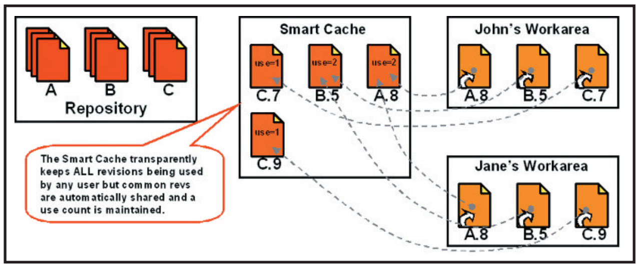
A
B
C
Repository
Smart Cache
use=1
C.7
use=2
B.5
use=2
A.8
use=1
C.9
John's Workarea
A.8
B.5
C.7
Jane's Workarea
A.8
B.5
C.9
The Smart Cache transparently keeps ALL revisions being used by any user but common revs are automatically shared and a use count is maintained.

FIGURE 1 SMART CACHE FOR DISK OPTIMISATION

which working sets of files are managed is not optimised. Giving each user their own set of files provides stable work areas but at a massive cost in disk storage.

Providing symbolic links from the designers' work areas to files in a common, read-only release area reduces disk space but causes lack of clarity. While synchronisation takes place in the common area, it is not always clear which revision is in use.

Putting the latest revision of files in the common area updates the area every time a revised file is checked in. ClioSoft has a new approach in release 4.0 of SOS, (Figure 1) Work areas are symbolically linked to files held in a smart cache, holding all revisions being used by any user. Common revisions are shared and the cache keeps a use count. This makes efficient use of disk space, and provides very efficient update and synchronisation.

The user has complete control over their work area, setting any rules on, for example, RSO (Revision Search Order) they require and can update their work area when they want and not when the system insists. The smart cache can be implemented on multiple sites.

Hardware design uses a lot of different design tools, unlike software design, and the design management software has to integrate with the existing design flow and tools from vendors such as Cadence, Mentor and Summit. The design management system has to reflect the way in which the design tools work; for example, checking out cell views in a design hierarchy. If this close integration is not immediately available, some DM vendors will help in customising the design management system.

MANAGEMENT

The final level of evaluation is project management which must have visibility of the file status and who is working on what files. Other management needs include updating the work area to synchronise the project and to write rules to specify which revisions to use, the ability to roll back the work area, and an audit trail for the whole project.

Really effective management will also benefit from project metrics, tracking for particular modules or the whole project changes in, for example, the number of changes and numbers of gates. At key points in the project life-cycle, for example when a simulation takes place, or at a tape-out, it is important to be able to snapshot the entire status of the project, including all the test files, change requests, even the directory structure to recreate that state later. Finally, there should be access control, specifying, for example, which teams and individuals can access files and which can change them. EDE

Srinath Anantharaman is president of ClioSoft