

SMC MANAGES HARDWARE DESIGN

Software configuration management (SCM) techniques can be used to manage the chip design process resulting in greater accuracy, control and productivity

Although the development model for hardware may seem very different to that for software, the extensive use of hardware descriptions languages means that the processes are actually very similar. Essentially hardware designs are a digital asset management process up to the point of fabrication. As such, SCM techniques can manage the entire hardware design process. With the right SCM software, the rewards of adopting such a flow for hardware design are greater accuracy, control and productivity with significantly reduced administration overhead. Traditionally, chip design managers have used some combination of TCM (tar, copy, move) to manage their design trees. The process usually requires a database 'czar' to oversee it, ensuring the consistency of the design tree in its various stages of development. A TCM methodology usually follows a fairly well defined pattern, a 'freeze', followed by a period of intense activity, where different groups work to synchronise changes into a common, working database that accurately represents the current state.

TCM management

Typically, TCM systems are expensive to maintain and run. The difficulty of managing them is compounded by the requirement to manage multiple variants of the same database to meet specific schedules and/or features. As they are not incremental, 'freeze'-based methods also slow down the pace of development. A completely new set of copies is typically required to create a new database, a process that is often error prone due to poor management of the underlying data by the users themselves.

As a design progresses towards tape-

out, there is an exponential growth in the size of data files. These typically take the form of change dump files, analogue simulation output and parasitic data. The generation time for the data is long. In addition, the design tools that generate the data may change versions and have slight incompatibilities.

For any given design, there will probably be a large number of variants due to process issues, system bugs, timing and

Propagating the right set of changes to the appropriate variant is a very difficult problem without a powerful SCM system

crosstalk problems, and changed customer requirements. Propagating the right set of changes to the appropriate variant is a very difficult problem without a powerful SCM system. In many cases, the work is repeated across multiple variants, since there is no easy way to apply sets of changes to snapshot or freeze data. This is often the cause of wasted mask sets in the fab.

Managing shared components

By implementing a development model that enables automatic change propagation, design sharing, reuse and collaboration are optimised.

SCM is the ability to manage sets of files and their associated dependencies automatically. Many traditional tools provide configuration management explicitly through the use of symbolic names or tags. However, the hardware design process needs more implicit configuration management abilities.

With the Perforce SCM, for example,

every transaction in the system is assigned a unique change number, making it very easy to track changes throughout the entire design tree. Since state is stored centrally in the server and not in the workspaces, data recovery, mining and reporting tasks are fast and easy to perform. Its client-server architecture means metadata is stored on the server so all state can be queried directly through the database, rather than having to collect file status information from distributed workspaces across the design network.

The ability to track change packages and the ease of propagation is an essential part of the SCM process. Atomic transactions allow sets of files to be tracked

together. Even though each file in a codeline has its revision history, each revision is only useful in the context of a set of related files. The question 'What other source files were changed along with this particular one?' cannot be answered unless you track change packages, or sets of files related by a logical change.

Inter file branching

Inter file branching (IFB) provides the cornerstone for working with configurations. It begins with the natural model of copying software files and renaming them and finishes with a collection of techniques that make its model usable. In practice, if branching a file means copying it in the repository then storage space can be a concern. A simple antidote is to perform a virtual copy, where the newly branched file makes use of the contents of the original file. This requires a level of indirection between the repository name space and the actual underlying object store.

When the branch is extended by adding

a new revision, the branched file can acquire its own separate entity in the object store.

Supporting variants with virtual copies reduces the space requirement of a variant to be merely a record that points to the original file. This virtual copy can also be used when one variant is explicitly synchronised with another. If the designer wishes to fold a branch back into a trunk and make the two identical, both can reference the same content at that point. The change from virtual copy to entity is tracked internally and provides a complete audit trail. Through transitive closure, it is possible to compute for any revision of any file whatever deltas it incorporates from other files through first, second, third, etc. generation merges or replaces.

This powerful feature allows changes between branches to be propagated automatically. There is no need for diffs (often an unusable technique for determining changes on binary objects), or other manual techniques to determine the difference between trees that shared a common fork point. It works seamlessly across design hierarchies too, because Perforce is able to compute and apply the incremental updates between data sets and also report them.

The traditional way to create branches is through the use of tags. However, they are generally very inefficient in comparison to IFB techniques. They are not incremental, data must be continually re-tagged and it is difficult to visualise or get reporting information on the difference between two or more sets of tags. Tags also proliferate making tree management more complex than it needs to be and tagging is a generally slow operation in most databases, both for the creation and recovery of file sets. As an explicit operation - if something isn't tagged, it is next to impossible to return to that state. Using IFB techniques, all state is recoverable at all times. In CAE databases traversing the hierarchy using Skill, Scheme or other languages builds a tag set, yet this is an inefficient way to map data sets and requires the database to be active in order to make a configuration.

Implementation strategies

Perforce implementations will vary according to the specific nature of the chips being designed. For 100 per cent

synthesis-based ASICs, the majority of configuration management activity will take place at the RTL level. In general, for RTL and other hardware databases, branching and merging is not the normal activity as it is in the software world. If place and route is performed in-house, the Avant! MilkyWay database is amenable to Perforce revisioning. For other chips, where there is a component of custom layout and schematic databases, the Open Source Cadence-Perforce integration can manage Cadence layout and/or schematic databases seamlessly. This integration is coded in less than 5,000 lines of Skill code and provides Cadence users with the power of Perforce in an integrated GUI. It

has the richest feature set and best performance of any known Cadence integration including TDM, DesignSync or ClioSoft for zero additional capital cost, once Perforce has been licensed.

The server hardware requirements can be met with low cost PC hardware running Linux or FreeBSD, unlike other systems that require expensive proprietary machines.

Shiv Sikand is Senior CAD engineer, Matrix Semiconductor; Dave Robertson is director of European operations, Perforce Software

● www.perforce.com/understanding.html