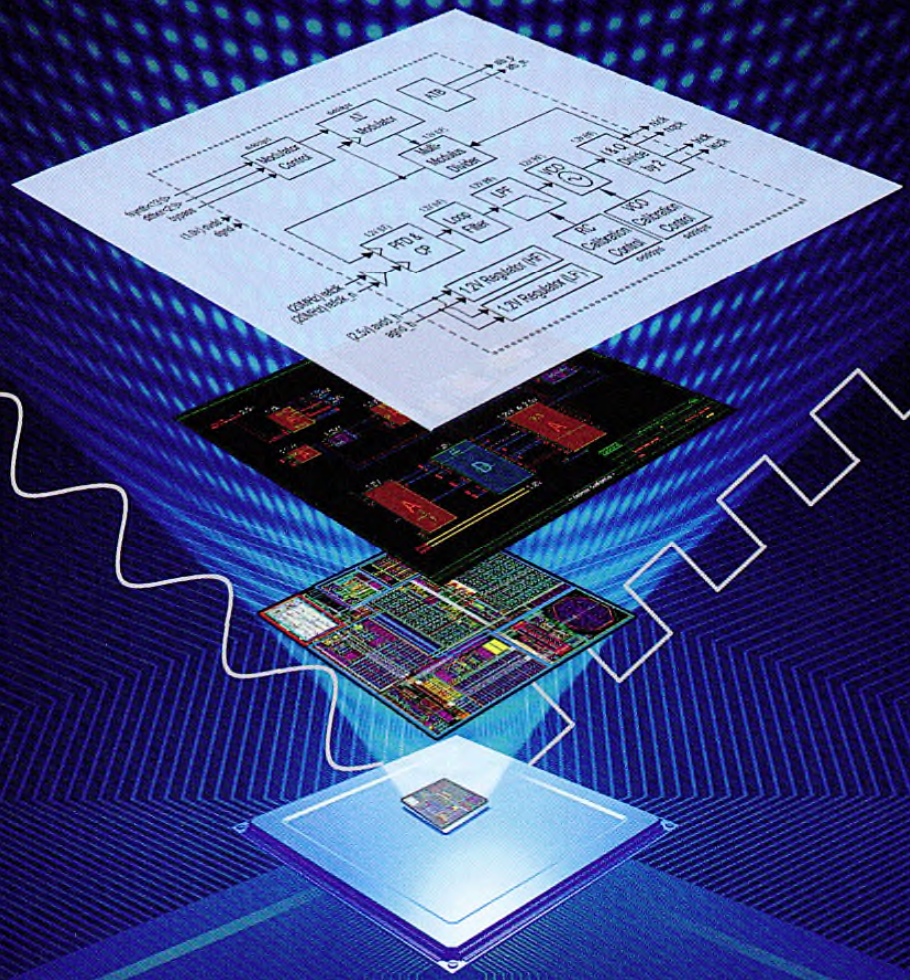
# Mixed-Signal Methodology Guide

Advanced Methodology for AMS IP and SOC Design, Verification and Implementation



Jess Chen

Michael Henrie

Mladen Nizic

CHAPTER 11

# Data Management for Mixed-Signal Designs

Michael Henrie and Srinath Anantharaman

## Introduction

Software teams have long used version control and data management systems and they have become an integral part of a software development environment. Practically, no significant software project is started without a software data management system and methodology in place. There are a variety of solutions, typically referred to as Software Configuration Management (SCM) systems, to choose from – commercial systems such as Clearcase or Perforce and public domain mainstays such as CVS and Subversion.

Hardware design teams have come late to the design data management table often referred to as Design Management or Data Management (DM). There are many reasons for this:

- Hardware teams have historically been smaller than software teams
- Hardware projects are typically partitioned into blocks. There is less overlap in responsibilities and therefore less likelihood of contention for the same data files
- Design flows involve graphical tools such as schematic or layout editors. Design data is often binary and made up of collections of interrelated program-generated files. Users cannot easily identify data files with design objects they have created
- Design flows are complex, involving several tools from multiple vendors. Version control is usually not an integrated part of this heterogeneous flow
- Hardware designers are typically less familiar than software engineers in using command line tools with arcane options.

Front end digital designers, for whom most of the design data is Verilog or VHDL text files, have by and large adopted software techniques for management. However, other flows such as analog/mixed-signal, custom layout, and board design have unique requirements that do not lend themselves to management using SCM systems and techniques. A hardware design data management system must meet the needs of digital designers and firmware/software engineers who typically work with text files and also address the requirements of other flows that employ graphical editors and create large binary files.

## Today's Mixed-Signal Design Environment

Several factors, such as an exponential increase in design complexity and shrinking market windows, are leading to larger design teams. Additionally, the increased competition due to globalization requires the use of the best available engineers irrespective of location. The design flows have multiple stages and tools - specification developed with a word processor, design and functional verification, logic synthesis, timing and power analysis, layout, etc. A broad spectrum of data is generated that needs to be shared across multiple sites and platforms including Unix, Linux and Windows. Adoption of version control and design data management solutions are on the rise as designs have become more complex, design teams have grown, and globalization has increased the need for collaboration.

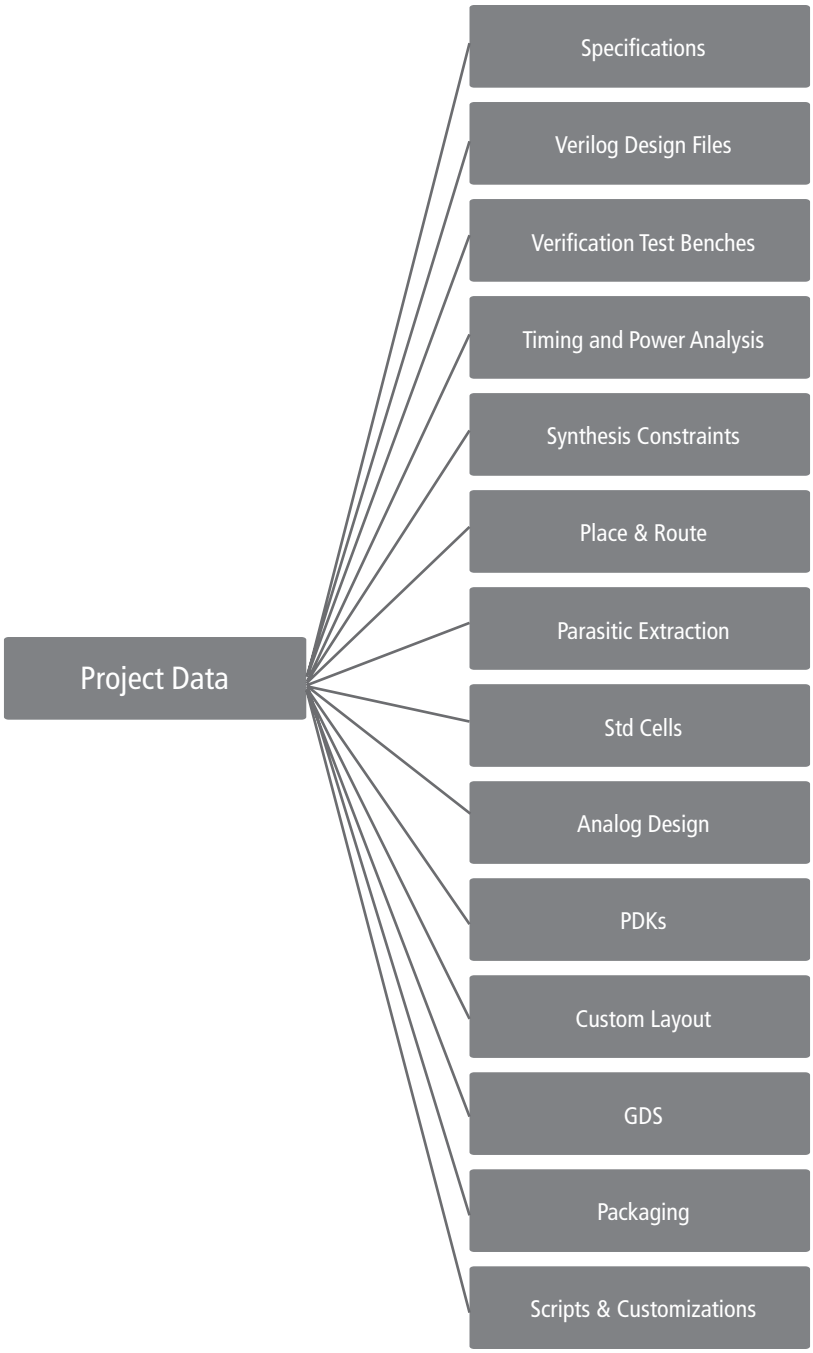



*Figure 1: Broad Spectrum of Design Data*

# Traditional Team Design Techniques and Pitfalls

Designing in a team implies sharing design data that has been completed and keeping design data that is in progress isolated. This is often achieved by using an ad-hoc process where data is managed by copying designs from the isolated development / scratch libraries to the shared master libraries. A typical workflow for managing and sharing design data is illustrated in Figure 2.

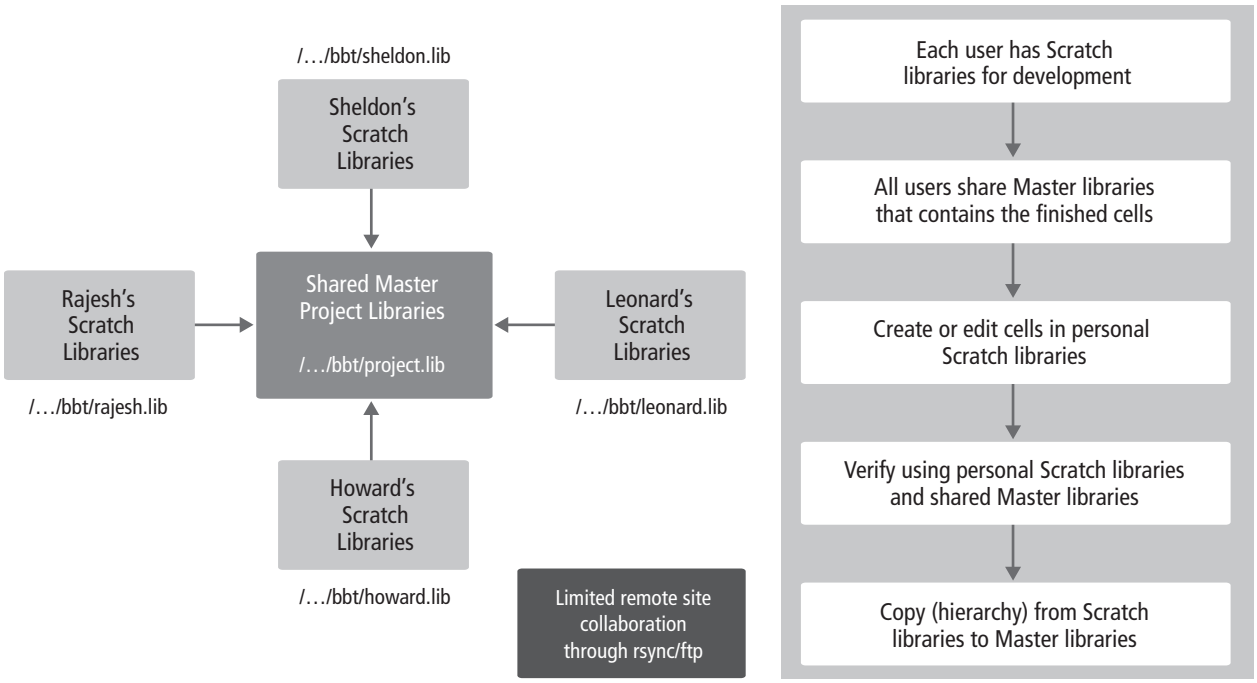



*Figure 2: Traditional Team Design Technique*

Designers often keep backups, so revision control is achieved by arbitrary naming conventions. Soon libraries are littered with cell-views containing names such as "layout_4" or "schematic_jan_14_2009". Changes are tracked manually, often by a project leader or manager, and communication is often via meetings and emails. Since everyone is sharing the design data, access controls are hard to enforce and are often left to an honor system. Finally, as project milestones are reached, the entire project is archived as a way to capture each milestone.

When development is spread across multiple sites, design management becomes even more complex and communication is impeded by distance, time zone differences, and language. A typical way to tackle this problem is to partition the project so that each part is primarily managed by one site. Completed and shared data is periodically synchronized between the sites often using a popular utility "rsync" that can synchronize data between two sites by transferring the changes.

With such traditional approaches to collaboration, project schedules and quality rely heavily on human perfection. This arrangement may work with a small team of veteran engineers but as the team size grows there will invariably be accidents. Users can inadvertently overwrite each other's data when they copy changes from their scratch libraries to the master libraries. Hierarchical copying, though very convenient, can have disastrous consequences. With no version control,

the changes may be lost forever, or at the very least be time-consuming to retrieve from backups. A much bigger problem occurs when the overwritten data is not detected and the project tapes out without this knowledge. This sort of error can necessitate an expensive re-spin.

Users have to be careful about what they modify or delete, which takes up valuable time. Libraries become cluttered with saved backup versions and shared master libraries tend to accumulate unused cells requiring periodic cleanup. Results of simulations and verifications are compromised by the fact that shared data could have changed midway through the process. Inefficiencies become even greater when multiple sites are involved and time zone differences delay turnaround of issues.

In the end, the lack of a design management system and methodology is a nightmare for management since there is no accountability or any audit trail of changes. Managers get burdened with the mundane tasks of bookkeeping and gate-keeping, which are better managed via software automation. As additional red tape is added to improve accountability and to reduce errors, it further slows down development and takes more quality time away from project leaders and managers.

## Requirements of a Design Management System

Collaborative mixed-signal design is similar to collaborative software development but with some distinct differences:

- Designers work at abstraction levels of libraries, cells, schematics & layout and are typically insulated by the design tools from the physical files used to store the design data
- Design tools create files that contain design data as well as temporary run files that need not be managed. End users may not know which files should be managed and which should not
- A design flow has several complex tools, sometimes from different vendors. Each tool may organize data in different ways
- A design object such as a schematic or layout is often stored in a collection of files created by the design tool with software-generated names
- Data is predominantly binary rather than text, and the size of the data is typically much larger
- Designers often think in terms of design hierarchy that is not apparent when looking at the file system
- Concurrent changes to a text file are easy to identify and merge. This is not the case with binary data such as schematics and layouts. Therefore, concurrent development is typically discouraged in analog and custom design
- Designers typically work within EDA tool flows and are not as comfortable using the command line as software developers.

Similar to the SCM system, a design management system must provide support for:

- Efficient collaboration between team members at the same site or across the world
- Version control

- Release management
- Variant development
- Security and access control
- Audit trail of activity and history of changes
- Integration with issue tracking.

However, it must also meet some additional requirements:

- Tight integration with the design flow so the DM commands are readily available to the user and operate on design objects
- Support for checkout with lock model to prevent concurrent editing
- Minimize use of disk space thereby reducing IT overheads of backups
- Management of composite design objects made up of collections of files
- Support for shared workspace so multiple (typically layout) engineers can modify and verify their changes together
- Identify and visualize changes made to graphical design objects such as schematics and layouts
- Support for operating on design hierarchies
- Ability to reuse Process Design Kits (PDK) and designs across multiple projects and sites.

Above all, a design management system should be both easy to administer and use for it to be effective. Designers already use multiple tools and complex flows. If design management adds roadblocks and additional overhead, then it is unlikely to be adopted.

## Managing Projects with a DM System

When a team is ready to start a new project, typically a user is chosen to be the project administrator. This project administrator then prepares the project to be used with a DM system. With the help of a System Administrator they choose the appropriate system hardware and disk storage to host the project. Once this is determined, the project administrator proceeds with configuring and starting any necessary design management software and setting up a project repository. The project repository is a central storage location on disk where the revisions of the project data along with any data attributes, often referred to as meta-data, are stored. Typically, the project repository is managed by a DM server daemon process, started and managed by an administrator, and therefore secure from any accidental damage or access. Designer's use a DM client to connect to the project repository via the DM server daemon.

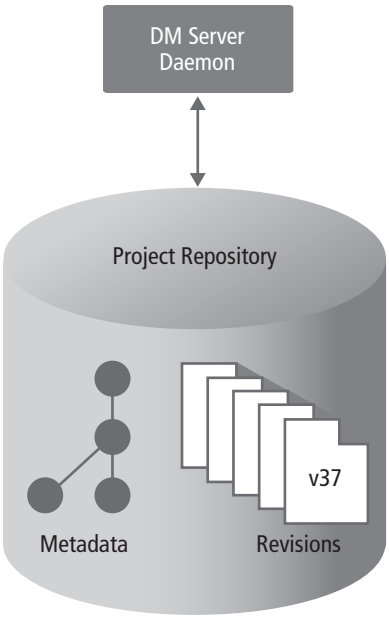



*Figure 3: Project Repository*

If the project team is made up of remote distributed users, the remote design management software will need to connect to the project's central repository, so all project contributors may access the data. The project administrator, with the help of remote project administrators if available, would set up this access to the repository.

Other project administrator tasks include assigning user roles and groups for the different users, setting up notification triggers, adding references to shared IP or PDK libraries, and importing any existing project data into the repository. Existing data can be imported via an import script or manually using the DM software.

Before the users begin their work, they will need to have a correctly configured tool environment. The project administrator can create setup scripts and tool configuration files that the team will use to setup their work environment. The project administrator could then create/check-in these configuration files into the root of the project. Using the DM system to store and distribute these files is an excellent way to ensure all team members, both local and remote, receive the same environment setup as they would be an integral part of the project's data. Managing these files as part of the project allows the setup files to be included in the final project release as well as the ability to track any changes made to them along the way.

Once the project administrator has finished these tasks, the team members can begin work on the project. The users will need a location and environment to create, save, and work on their data. This location is typically called a sandbox or workspace. The workspace is a directory located on a locally-mounted disk either in the users' home directory or on a disk setup by the project admin specifically for the project. In order to access the project, the user must first connect their workspace to the repository. This is typically done using a DM setup command.

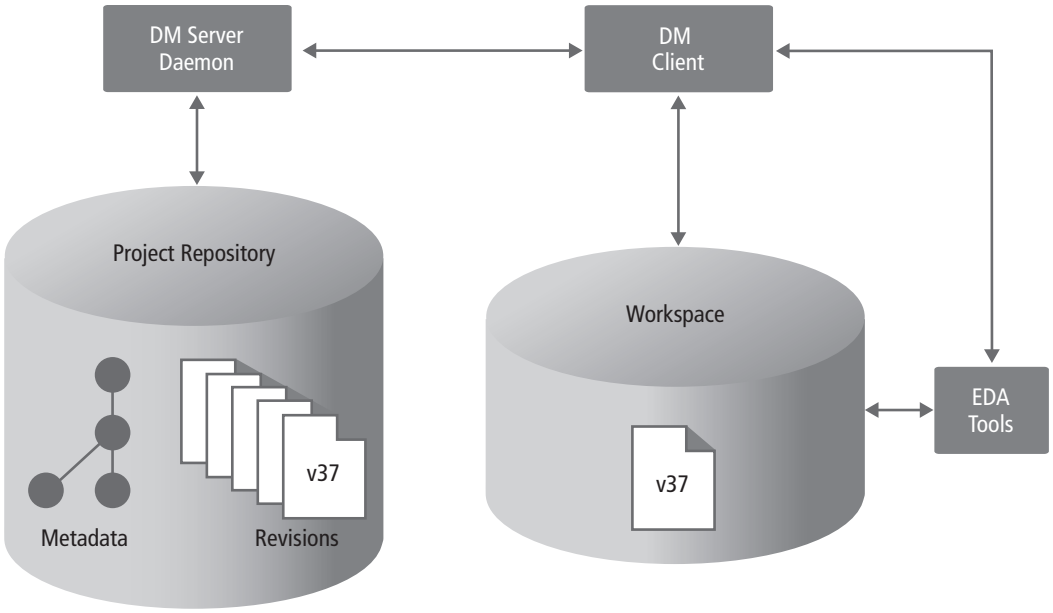



*Figure 4: User Workspace*

Once connected to the repository, the user can request the project data at which point they would receive the necessary setup and tool configuration files along with any preexisting project data. They are now ready to begin their design work by loading the setups, invoking the tools and creating their data. After they have created and saved their data using a DM integration in the design tools, they may share it with the rest of their team members by checking their files into the repository. When the files are in the repository other users can reference, check them out, change or add to them. All check-ins to the repository create a new revision of the data and get assigned a revision number, an author, a date, a comment, and any other critical attributes that are useful for tracking the state of the data. This makes it easy for all members to track the progress of each user, merge their files together as a working set, and to rollback any changes that may not be compliant. A full audit trail is created at each step making it easy for the project administrator(s) to track the progress of the project.

Each user's workspace is usually a full copy of the latest revisions of all the data in the project even if the user is only working on a module of the design. This is needed to get all the necessary dependencies, configuration files, etc. and to be able to run simulations and verifications. As the size of the team grows, workspaces take up excessive amounts of network storage and system backups take longer often backing up the same data multiple times. DM systems can address this issue by allowing users to create workspaces with symbolic links to a mirror or cache area. A user's workspace only has physical copies of files that they have checked out for editing while all the other files are symbolic links. This helps dramatically reduce the use of network storage resources, improves backup efficiency, and also speeds up the creation and updates of workspaces because only a symbolic link has to be created instead of copying the file revisions into the user's workspace.

## Collaborating Across Globally Distributed Design Centers

Without a DM system, global teams have tried alternative methods to manage and share their data. Such methods include the use of remote file mounts, File Transfer Software (FTP), mirroring software, email attachments, or even mailing a physical disk as ways to share their data with each other. Use of these methods lead to a very ad-hoc and cumbersome flow which is sluggish and error prone. With the use of remote mounts, it can take several minutes just to open the data in the tools. Other issues such as having multiple copies of the same data at each location, leads to confusion about which is the "golden copy" and often people accidentally overwrite others' data.

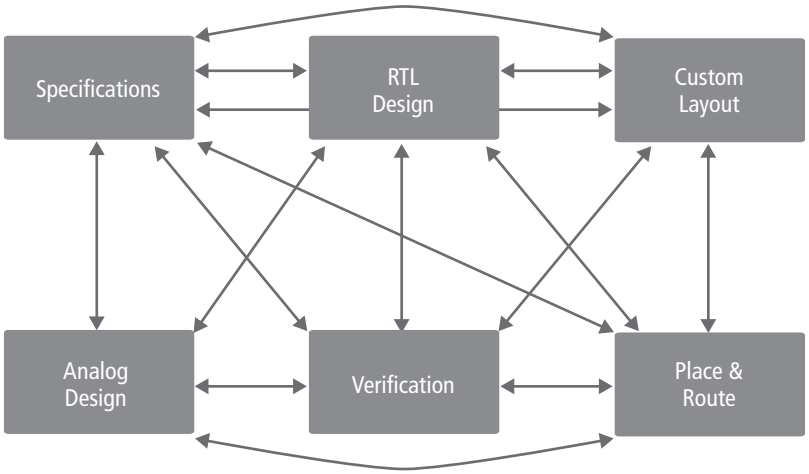



*Figure 5: Ad-Hoc Collaboration*

In such an ad-hoc system, seeing other users' changes on a regular basis is difficult. A user would need to wait for copies of the data to be sent to them so that it would be available at their site. This results in a poorly managed system leading to lost time, rework, and virtually no collaboration is accomplished, potentially pushing out the completion of the project.

With the introduction of a good DM system these problems are solved. Instead of remotely mounted file systems, data is stored on local storage, making the responsiveness much better. All team members save and check their local data in as a revision to the central repository. This repository allows all users on the team, whether locally or remote, to gain access to the same "golden copy" of the data, thus eliminating any confusion.

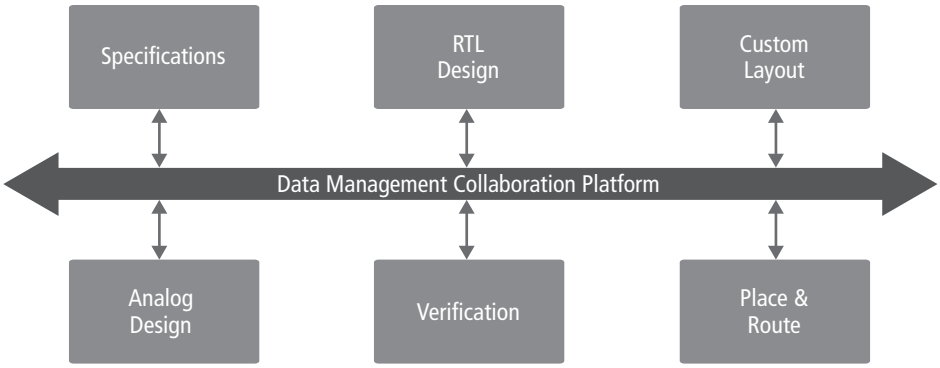



*Figure 6: Streamline Collaboration with a DM System*

For the data revisions to be available to all users, the DM system takes care of distributing the correct revisions to all local and remote locations. Users are isolated from changes made by others because they have their own sandbox/workspace. Each user can control their workspace using the DM system to 'pull', when needed, changes made by others since their last update. Alternatively, they may choose to have the changes automatically 'pushed' immediately or periodically to their workspace by the DM system. The choice of the type of update depends on whether the user values having a stable workspace or needs to be in sync with other developers. DM systems typically allow an administrator to set up a cache/proxy server at remote sites to improve performance. Clients at the remote site can connect to the cache server and get file revisions rather than go back to the primary server to get the data. The cache server provides practically the same performance at the remote site that users at the primary site see. Additionally, a new file revision is brought across from the primary site to the remote site only once to the cache server. It is then served from there to all the users at the remote site, thereby optimizing the use of bandwidth between the sites. A cache server could also be set up for automatic updates - either periodic pull or immediate push so a newly created file revision is already available in the cache when a user at the remote site needs it.

DM systems provide additional features to enhance collaboration. Notifications can be set up to send alerts to users when data is ready. These alerts are very valuable when a user is remote and they wish to know when another site has checked in data. Although email notifications are tempting to setup, they are not very effective because the volume of emails often overwhelms the users. Typically, DM systems provide alerts and icons in the GUI that are more effective and can be set up for immediate updates, so users can always get the latest project status. Users and managers can also generate a variety of reports to monitor the progress of a project. Instant messaging is another useful feature that can be provided by a DM system to better communicate recent data changes in real time between global users.

## Team Design Using a DM System

A DM system allows a team of designers to work and collaborate more effectively by addressing the pitfalls of traditional techniques. A DM system allows designers to develop in isolation and share changes effectively when needed without stepping on each other's toes. Additionally all changes are tracked and any change can be easily reversed. Figure 7 illustrates a typical workflow using a DM system.

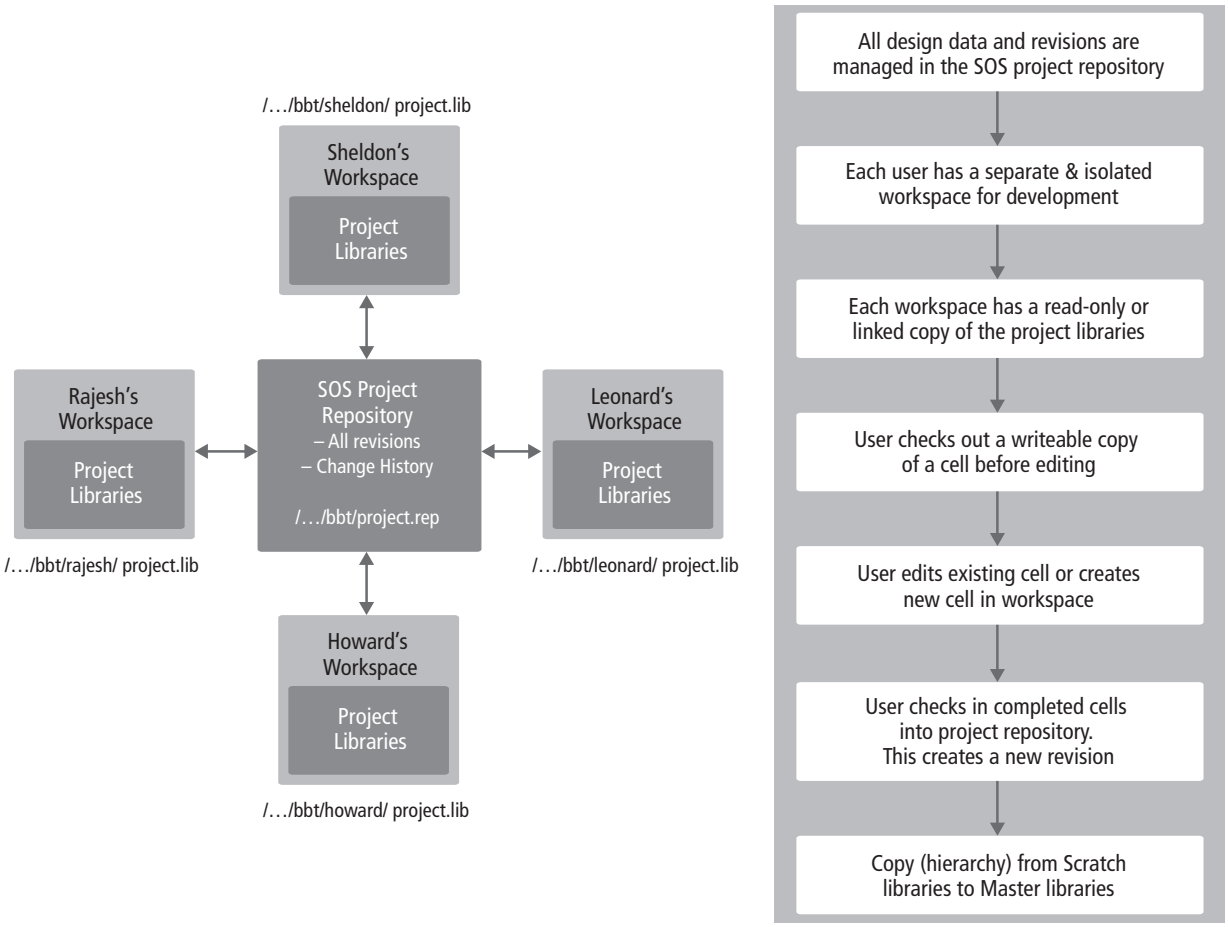



*Figure 7: Team Design with DM*

Each designer has an isolated workspace created by the DM system that essentially has a copy of the project with the latest versions of all the files. To optimize disk space, the workspace could be created with symbolic links to a cache. When a designer wants to modify a design object, they use the DM system to check-out the object which makes it writeable in the workspace and places a checkout lock on the object in the project. This prevents anyone else from accidentally changing the same object. The designer can make changes and verify the design in the isolation of their workspace, thus not affecting anyone else or being affected by anyone else's changes. Once the changes are complete and verified to the best of the designer's ability, the design object can be checked in. This creates a new revision of the object and releases the lock in the project. Other designers can now get the checked-in change into their workspace when needed. Project leaders can query the DM system to get a list of all changes to the entire project and revert back any change if needed. This flow using a DM system provides a controlled and stable environment for designers to work, prevents data loss, provides full accountability, and allows easy rollback if necessary.

# Workflow for Analog Design with DM

Digital designers typically work directly with the RTL source files and are aware of what files need to be managed. With a DM system in place, the only change a designer will need to do in a daily workflow is to perform a check-in after editing their files, and to do an update to synchronize with other designers' changes.

Managing design data for analog and custom design is complicated by the fact that the designers are using graphical tools such as schematic or layout editors which produce design files stored in libraries. Designers work at the higher abstraction levels of libraries, cells and views and are not necessarily aware of the physical files created by these tools.

Project planners begin by developing a specification as to what a chip should accomplish. Specifications are usually done in standard office documents on a PC or Mac and need to be revised and shared as the project develops. It is best to manage these documents in the DM system so they can be easily shared and everyone has the latest revisions. Once the planning stage is complete, the designers begin to engineer the actual circuit for the project.

The analog circuit designers begin by creating circuits using schematic capture software to complete this task. During the creation of the schematics, they can save and check-in the schematics to the DM repository when they are satisfied with the quality of the schematic. The DM commands should be available directly within the design flow to make the process seamless thereby improving compliance.

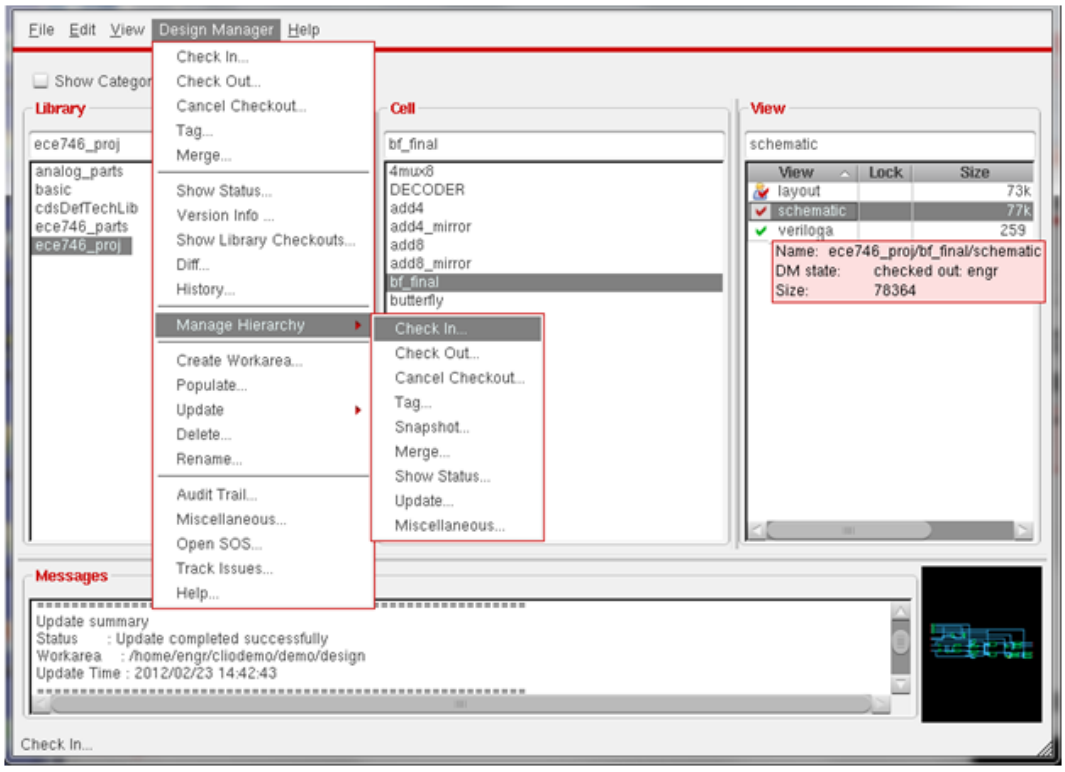



*Figure 8: DM Commands Integrated into Design Browser Window*

The designers will continue to change the schematics thus creating new revisions. At any point they will always have the opportunity to revert to a previous revision if new changes they have made are not successful. The DM system gives all the team members the ability to see what other team designers have accomplished, allowing for full and updated collaboration on a consistent basis.

Part of the design process is to simulate the schematics to make sure they meet the specifications. Circuit designers begin testing by creating various testbenches and stimulus files, all of which can be versioned using the DM system. During the testing/simulation process, the designers switch between different cases to simulate the schematics by using different revisions of the files. This greatly simplifies the process of testing, because the testers now have a unified way to make the process happen. Using scripting and particular DM commands to choose the testbenches and stimulus files, they can batch the corner case runs together, eliminating the need for the designer to be present to begin each new simulation.

After the simulations are completed, the designers will review the simulation results. Simulation data output files can be fairly large, so it's up to the discretion of the designer to place this data under DM control. Usually, most designers avoid managing simulation data because they are large and can be regenerated. Instead, they will view the results in the design tools, at which point they can take snapshots of the data/waveforms, add any annotations, save it out as an image/file, and check them into the DM system to share with the team.

Once the circuit designers are satisfied that their schematics meet the requirements, they can then pass the schematics off to the physical layout team. Using the DM system, the circuit designers can tag the appropriate revision of the schematics that the physical designers should use. The physical designers will receive notification that the schematics are ready. The tagged schematics can then be brought into their design environment, allowing for another seamless transition using the DM system.

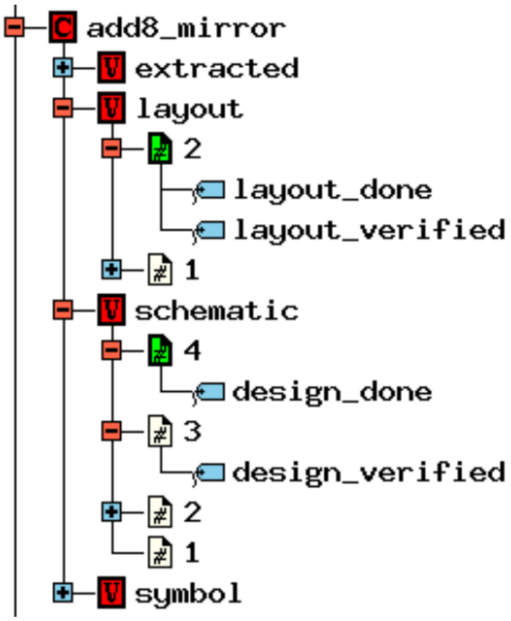



*Figure 9: Using Tags to Communicate Design Status*

The layout process is now ready to begin. As with the circuit designers, the physical designers will utilize the DM system to revision the progress of their layout development. The use of the DM tool, allows the circuit designers to view and track the progress of the layout. This makes the progress most effective because it again allows full collaboration of the team. The physical designers run Design Rule Checks (DRC) and Layout Verses Schematic (LVS) checks and upon success of these tests, the layout can be tagged as DRC and LVS clean. When the layout is ready, the layout designer can tag it, for example, 'layout_done', to notify the circuit designers to review the layout for any flaws. States of the design, reviews, and other critical clearances along the way can be easily attained through the use of the DM tool.

If requested from the design engineers, physical designers can also run parasitic extraction on the design. The parasitic extractions can be used by the circuit designers to re-simulate the physical layout characteristics to make sure that specifications are met. Again, through the DM system, the parasitic extraction output can be managed and tagged to notify the designers that the data is ready for re-simulation and review. Any further changes to the schematics that result from the reviews will be done and new revisions of the schematics will be made. The process repeats itself through notifications and tagging until the section has been reviewed and deemed ready by the design engineers.

When all cells of the design are complete and have been tagged, the team can hold a design review with the help of the DM system's audit information. Upon completion of the review, the project is deemed ready to go to mask for production. Any output generated to produce the mask can be managed in the DM system as well.

## Managing Engineering Change Orders

During the design and testing phase of the project, problems or improvements are discovered along the way resulting in the need to change the current design and develop a process to communicate the required changes to the rest of the team. This is typically referred to as an Engineering Change Order, or ECO.

An ECO contains specific information such as what changes are being made and by whom, a description of what the problem or improvement is, what files are involved, the date and time of when the changes were made, and usually for whom the changes are intended. Through the use of the ECO the changes can be created, implemented, tested and reviewed giving the team a good method to track the changes from start to finish.

With the use of design management software, a team can create and track their ECOs. The DM software provides the basic features required for creating an ECO such as:

- The file revisions to store the actual changes
- A history, containing author, date, change comments, and important notes
- File tagging for marking changed files
- Notifications to provide updates for the team.

In addition, third party bug tracking software can be integrated into the DM system, allowing engineers to further enhance an ECO by the use of tickets. These tickets will track the changes from start to finish.

An ECO is first initiated by a circuit designer with updates to the schematics or model files. Using the DM software, the designer checks in their changes, adds comments about the change, and then tags them marking them ready for use. An ECO ticket can be created and a notification is sent to the physical designer to notify them about the new ECO.

The DM software provides several features allowing the physical designer to review the ECO and determine what they need to change. Using the revision history, which includes important notes about the changes, the designer can begin making updates. However, relying on notes made by the circuit designer is both inefficient and error prone. All DM systems provide tools to quickly and accurately highlight the differences between two revisions of a text file. This is invaluable when trying to identify the ECOs to a text based model.

Changes to a schematic or layout are harder to identify manually and some changes to properties may not be visible. It is imperative that all ECOs are carefully reviewed before tape-out but scheduling can make it difficult. Some DM systems provide features to compare revisions of graphical design objects such as schematics and layouts. A hierarchical comparison can identify changes at all levels of the design hierarchy. This ability allows designers to ensure that all ECOs for the project are identified and implemented in the layout. Changes to the layout can be quickly reviewed without going through the time consuming process of creating the GDS II files and running XOR comparisons.

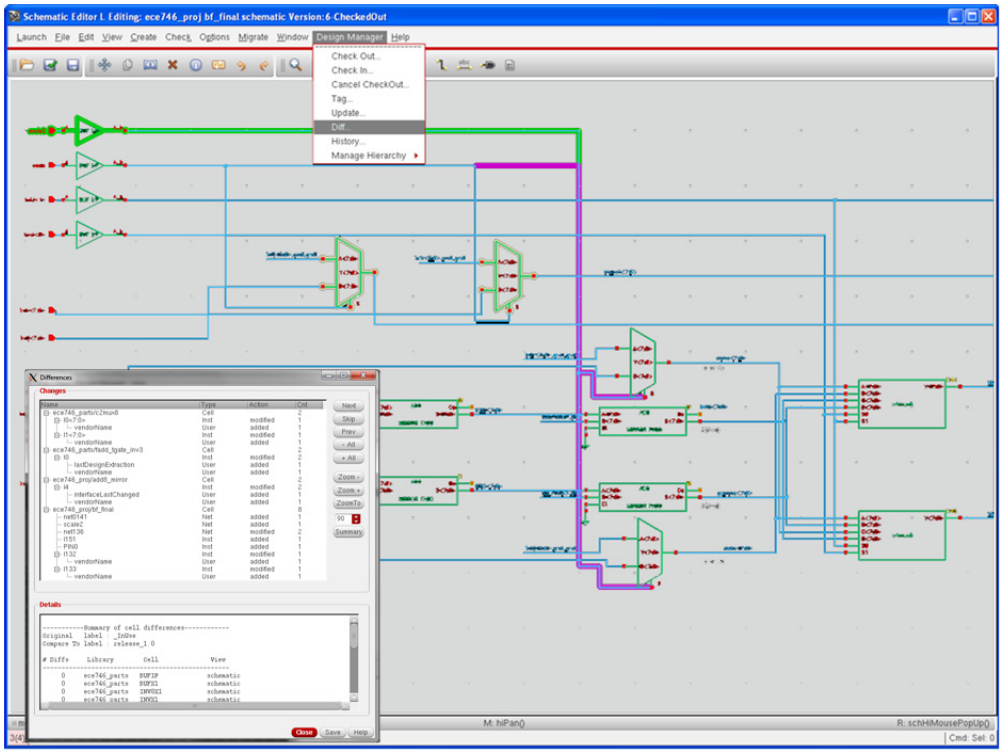



*Figure 10: ECO changes Highlighted using Schematic Comparison Feature*

Using these DM features, the designers can easily find, resolve, and implement the changes. Through the use of DM and ECOs, the overall state of a project is consistently updated and tracked. When a project is nearing completion, all ECOs must be finalized. A review could be held to ensure that all ECO tickets have been fixed and closed. The project would then be ready for tape-out.

## Tracking Releases and Variants

When a project is completed, the team should capture all revisions of the data that have been used to create the final product. Using the DM system, they can record a snapshot of the current configuration by tagging all the files and their current revision with a meaningful name such as "tapeout_1.0". The snapshot captures the state of the project at that particular time and records that state in the project repository without making an entire copy of the project.

Using a snapshot, the data can be recreated using the DM system at any time. This allows team members to clean up their workspace, freeing up disk space for future projects. Finally, through the use of the DM system and the snapshot, the data can be easily archived.

While testing a new product, problems may be discovered and variants of the release may be needed. DM software can easily track the needed changes. The team can continue to work in the same repository and make changes for the new variant and create a new snapshot when completed.

However, companies are always trying to create a new generation of the product, while still maintaining the original product in a maintenance mode. It can be a challenge to make minor fixes to the existing product while at the same time developing the next generation of the product on the same set of data. Through the use of DM this can be easily accomplished through one of two ways, either copying or branching.

The copy method is accomplished by having the project administrator create a new repository as a whole new project. The administrator would take the snapshot of the current product and import/copy it into the new repository. Once the data is imported into the new repository, the team creates new workspaces and begins work on the product. With this setup, any maintenance changes to the existing products continues in the original repository while the new development is isolated as a new project. Although this does provide the separation needed between the two, it does have its downsides. First, the data is duplicated between the two repositories requiring additional disk space. Second, all the history of the past development is not carried into the new repository. And third, any common fixes needed in both the maintenance and new development repositories would need to be made twice, leading to the possibility it could get missed in one.

The branching method, unlike the copy method, continues to use a single repository. With this method the new development for the next generation of the product continues forward on what is called the "trunk" or "main" branch, after taking a snapshot of the current product. Any maintenance changes required for the current product would be done on a "branch" originating from the snapshot. Additionally, any changes made on a branch that also needs to be part of the new development can be merged back to the main development trunk. This method provides several advantages; first, it uses only one repository so it does not require the additional setup. Second, any common data between the maintenance and production release remains the same

and shared, thus avoiding duplication. Using the branching method requires all team members to become familiar with how to create/merge data on a branch. With training, the users can master this and utilize the branching method to achieve a very efficient development model.

Design management gives project managers full control of all project releases and variants. Using the DM audit features, they can easily reference when a project was completed and released as well as track any problems and change requests along the way.

## Rules, Roles, Access and Permissions

A laissez-faire development methodology may work for small design groups of experienced engineers working together closely. However, as the development team becomes larger, more heterogeneous and spread across different time zones the likelihood of accidental changes and errors increase. An administrator needs to set rules about who gets to do what and have the DM system automate and enforce the rules. The rules should help prevent accidents without becoming onerous.

Some typical examples are:

- Preventing layout engineers from modifying schematics and vice-versa while still having read access to everything
- Making sure that contractors get to view and modify only those parts of the design that are necessary for them to get their jobs done
- Controlling what DM operations a user is allowed to do based on their expertise and role in the project.

A user's need for access to a resource is typically determined by two factors:

- What does the user do? For example, analog design, digital design, layout, functional verification, layout verification, etc.
- What is the user's role, expertise or level of responsibility? Is the user an administrator, manager, project leader, expert, novice, contractor, etc.?

The interpretation of read/write permissions is a little different for a DM system than for the operating system. Read-only permission in a DM system implies that a user can get a copy of the design object into their workspace and open it for reading, but cannot check-in a new revision of the object. It does not prevent the user from changing permissions on the files in the workspace and modifying the design object in their own workspace. However, without write permissions the user cannot check-in the changes thereby preventing the user from affecting the design.

Conceptually the Unix/Linux model of user, group, and read/write permissions works well even for a DM system. YGroups can be defined and permissions set to prevent one group of users from modifying design objects owned by another group. However, it is too simplistic in two ways:

- It relies on the user to assign permissions. For example, a user with an incorrect umask can create files that can be read or even written by everyone
- Many different operations may 'modify' a file. You can modify the contents, change attributes such as permissions or even delete the file entirely. With a DM system you have even more

operations such as tagging, branching, removing locks, or deleting revisions of a file. Therefore you need finer grain control over what write privilege means.

Using the DM system, the project administrator should create roles such as administrator, manager, engineer, novice etc. and define exactly what commands each user is allowed to perform. Additionally, the administrator must make sure that the right permissions are automatically assigned as new objects are created without relying on each user establishing the correct settings.

Access rules and roles must be agreed upon and understood by the team members and they should provide the right level of access for each member. Since the DM system provides version control and errors can be easily reversed it is probably better to err on the side of flexibility as too much control may hamper development. The administrator can then set up the DM system to automate and enforce the permissions.

## Reusing IP and PDKs Across Projects

When developing a new product it makes sense to take advantage of the vast resource of previously developed and tested Intellectual Property (IP) that can be used to decrease the development cycle. This IP could either be internally developed or taken from an external source. Part of the challenge of reusing IP is the ability to properly advertise, package and distribute it in a method where it can be easily found and used by teams across the enterprise.

IP management and reuse features typically provided with DM systems makes IP sharing successful and efficient. The IP librarian features provide the IP developer with the tools needed to update, bundle, document, publish, and distribute the IP to end users.

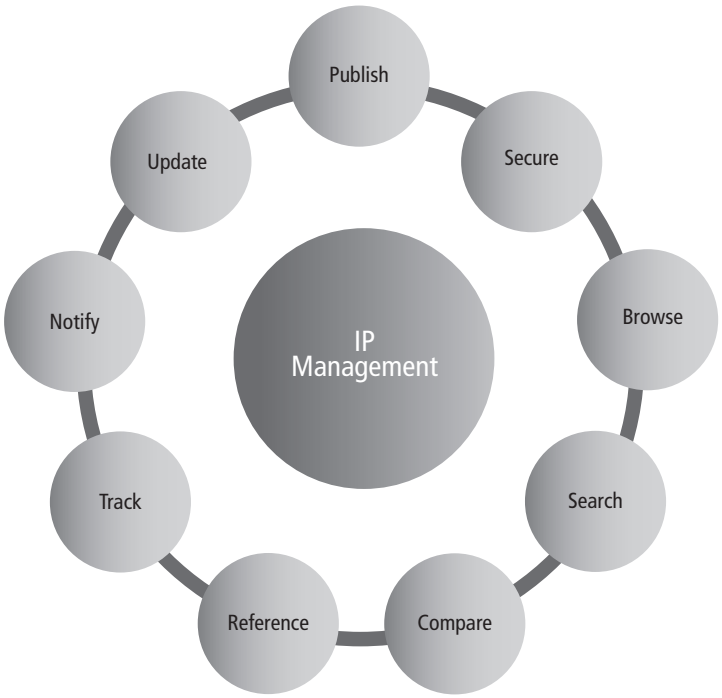



*Figure 11: Managing and Reusing Design IP*

The IP developers and librarians, through the use of DM software, develop, document and test the IP data. Datasheets for the IP describing its specifications are then created and managed. Using the IP management software, the IP can be bundled and published to make it available to internal or external customers.

The customers using the IP management portal, typically a web page, can search for, browse and view the IPs that are available. When an IP is found to match the requirements, end users can subscribe to it and pull it into their design environment. The DM software takes care of seamlessly pulling in the IP data for the project allowing for instant use. The IP management software further allows the IP publisher the ability to track who is using the IP and provides an easy avenue for notifications to be sent to the end users if improvements or bugs are fixed. New releases of the IP can be pulled as needed when they become available.

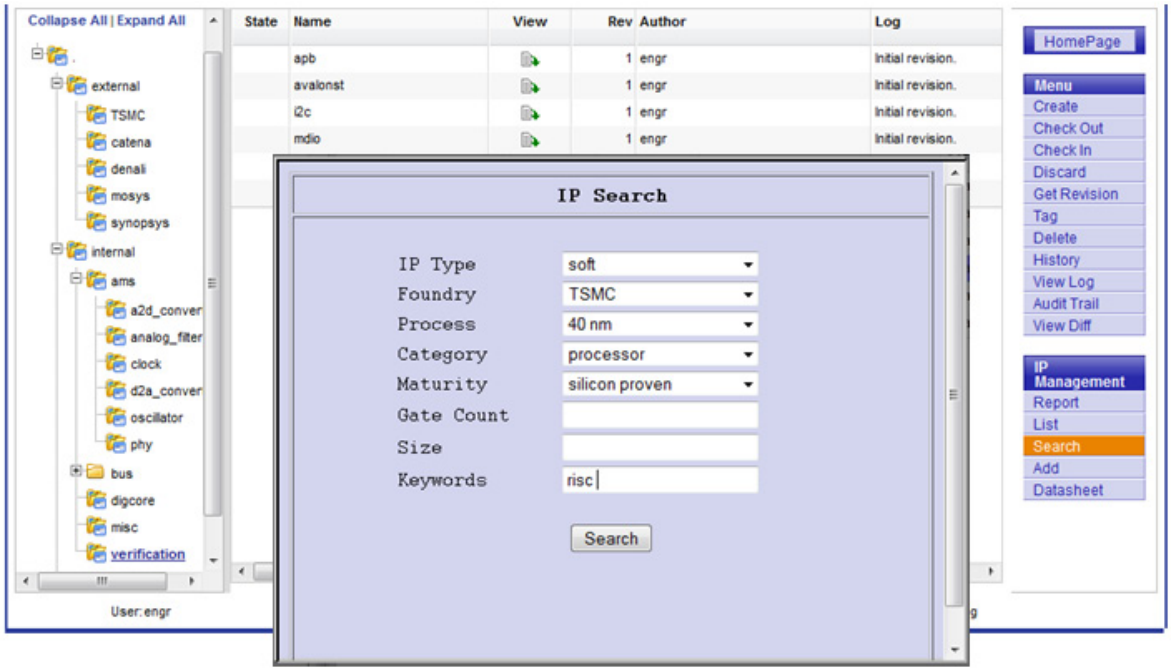



*Figure 12: Searching For Suitable IP*

In addition to IP libraries, projects need access to Standard Cell Libraries (SCL) and PDKs which contains the needed process components to develop the product. Similar to IP, these libraries need to be published, distributed, and shared among the projects that need them. The IP management software could be used to distribute these libraries or they could be stored in a DM repository and managed by a CAD team. Using a DM reference, these libraries can be used by any project that requires them. This provides a good method for the CAD team and the project teams to stay in sync and track the correct versions of the libraries that should be used and easily update to a new release when it is available.

# Conclusion

Software teams have long realized that they cannot function without a SCM system. As AMS teams have grown and become widely dispersed the use of design data management systems is increasing.

Design teams have found that by deploying an DM system, the productivity of the team and the efficiency of management are improved, and the probabilities of expensive re-spins are markedly reduced. The ROI of an DM system is realized in several ways:

- Real-time communication and collaboration between engineers at the same site or around the world, reducing the need for expensive, time-consuming meetings and error-prone email communication
- Prevention of data loss and easy recovery from mistakes
- Automated and error proof bookkeeping and gate keeping capabilities that relieve managers and engineers from mundane tasks
- Quick visual review of changes
- Efficient reuse and distribution of IPs and PDKs across the enterprise
- Reduced need for EDA support services since processes are streamlined and data sharing is automated.

# References

Evaluating a Design Data Management System, Scott Woods, IC Journal http://eejournal.com/archives/articles/20090224_ddm/

Multisite, collaborative hardware design calls for HCM, Ravi Poddar and Srinath Anantharaman, EE Times Design http://eetimes.com/design/industrial-control/4006392/Multisite-collaborative-hardware-design-calls-for-HCM

Design Data Management Improves Productivity – Even for Small Design Teams, Grego Sanguinetti, Chip Design Magazine http://chipdesignmag.com/display.php?articleId=2275

Implementing a Team Design Environment, Tom Dewey, Chip Design Magazine http://chipdesignmag.com/display.php?articleId=2353