APPLICATION NOTE

Keysight Engineering Data Management (SOS) Core: Integration with Synopsys Custom Compiler

Seamless Design Data Management for Custom IC Design Teams

KEYSIGHT

Table of Contents

Executive Overview

Synopsys Custom Compiler is a modern design environment for custom IC, analog, and mixed-signal design. It provides schematic capture, constraint management, layout editing, and physical verification capabilities used by engineering teams across the semiconductor industry. In collaborative, multi-designer projects, the ability to manage design data reliably within Custom Compiler is essential for maintaining design integrity, enabling predictable tapeout schedules, and sustaining engineering productivity.

Keysight Engineering Data Management (SOS) Core provides a native, deeply integrated data management layer within the Synopsys Custom Compiler environment. Rather than requiring engineers to switch between their design tool and a separate version control application, the SOS Custom Compiler integration embeds all core data management operations directly into the Custom Compiler Library Manager and editor windows. Checkout, check-in, update, diff reports, tagging, hierarchical operations, and audit trail are all accessible from within the menus that Custom Compiler designers already use.

This application note provides a comprehensive overview of SOS Core, followed by a detailed examination of its integration with Synopsys Custom Compiler. It covers the integration architecture, day-to-day designer workflows, advanced capabilities such as hierarchical operations and Revision Search Order (RSO) management, and the customization mechanisms that allow CAD teams to tailor the integration to their specific process requirements.

SOS Core: Platform Overview

SOS Core is Keysight's design data management platform built for small to mid-sized engineering teams that need performance, reliability, and simplicity. It provides high-performance version control of design data, seamless collaboration across teams and workflows, and referencing mechanisms for IP reuse-driven design.

SOS Core delivers enterprise-grade design data management in a streamlined, easy-to-deploy package. It empowers small to mid-sized design teams with fast, reliable version control, seamless EDA integration, and metadata-driven IP reuse.

Core Capabilities

Version control and data integrity. Every modification to a managed file creates a new, immutable revision in the central repository. The system maintains a complete history of every design object: who changed it, when, and what was changed. Status icons in the SOS GUI and within supported EDA environments indicate whether each object is current, out of date, checked out by you, or locked by another user.

Check-in, check-out, and preemptive locking. SOS Core implements a check-out/check-in workflow with preemptive locking, the standard concurrency model for hardware design data where concurrent modification of binary files is impractical. When an engineer checks out a design object, they receive write access while a lock prevents anyone else from modifying it until the object is checked back in.

Labels and Revision Search Order (RSO). Labels are human-readable aliases assigned to specific revisions (e.g., "design_done", "Gold"). The Revision Search Order (RSO) is a priority list of labels associated with each user workarea that controls which revision of each object is retrieved during a populate or update operation. This mechanism automates design handoffs between functional roles without manual coordination.

Snapshots. Point-in-time captures of the entire project state. If subsequent changes introduce problems, the team can revert the entire project to a snapshot state. Snapshots serve as a reliable baseline for tapeout preparation, milestone reviews, or risk mitigation.

Links-to-Cache architecture. User workareas contain symbolic links that point to the single copy of each file stored in a local Cache, rather than full physical copies. A workarea for a 100 GB project may consume only a few megabytes. This delivers dramatic storage savings and eliminates time-consuming workspace population steps.

IP reuse through references. Rather than copying IP between projects, SOS Core creates managed references that preserve provenance, traceability, and update control.

EDA tool integrations. SOS Core maintains the broadest portfolio of EDA integrations in the design data management category, including native integrations with Cadence Virtuoso, Synopsys Custom Compiler, Siemens Pyxis and Tanner, Keysight ADS, Silvaco Gateway + Expert, MathWorks MATLAB, and Empyrean Aether.

The Synopsys Custom Compiler Integration

The SOS integration with Synopsys Custom Compiler brings full design data management capabilities directly into the Custom Compiler Library Manager and editor windows [figure 1]. Engineers gain version control, collaboration, and traceability without leaving their primary design tool. The integration supports all core SOS operations, including check-in, check-out, tagging, hierarchical operations, revision comparison, workarea updates, and audit trail, all surfaced through the Design Manager menu.

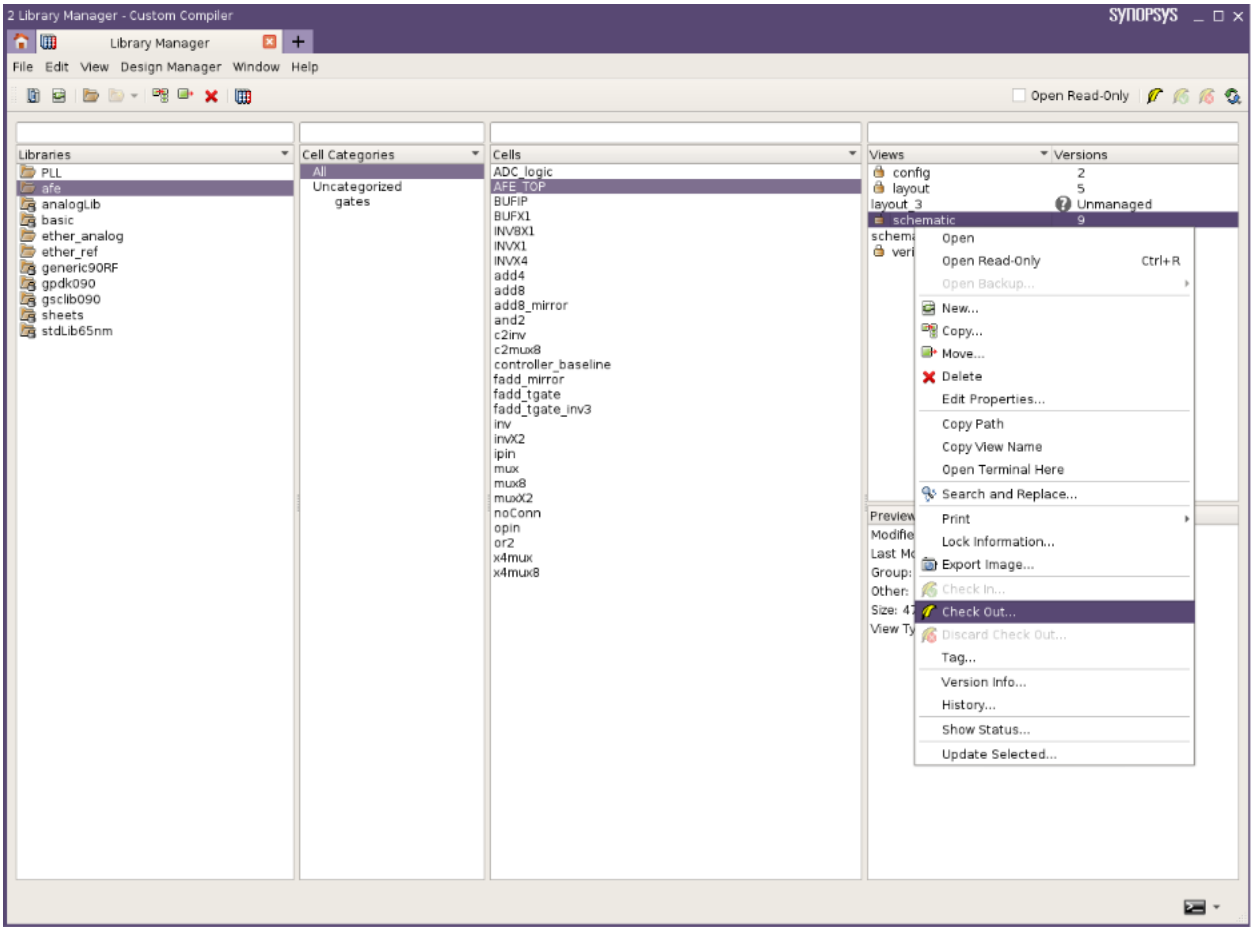

Figure 1. SOS integrates directly into Custom Compiler Library Manager

Integration Architecture

The SOS Custom Compiler integration is loaded at Custom Compiler startup through the $SYNOPSYS_CUSTOM_SITE environment variable, which specifies the location of third-party extensions. SOS installs its integration components into the $SYNOPSYS_CUSTOM_SITE/images/sos directory. Once configured, the integration injects the Design Manager menu into the Library Manager window and editor windows, registers status icon callbacks, and connects the Custom Compiler session to the SOS server.

The integration communicates with the SOS server through the SOS client libraries, which connect either to a local Cache Repository or directly to the Primary Repository depending on the deployment topology. For multi-site teams, the Cache architecture ensures that Custom Compiler operations resolve against a local data store, delivering local-speed performance regardless of where the Primary Repository is hosted.

To verify that SOS is properly integrated with Custom Compiler, designers can run the following command to check for the integration directory:

```
ls $SYNOPSYS_CUSTOM_SITE/images/sos
```

If this directory is found and populated, and the Design Manager menu appears in the Custom Compiler Library Manager window, the integration is active and operational.

User Interface Overview

The Design Manager Menu

The Design Manager menu appears in both the Library Manager window and in Custom Compiler editor windows. It provides access to all SOS operations a designer needs during daily work. The menu is organized into logical groups covering basic operations, status and history queries, hierarchical operations, and workspace management [figure 2].

Command	Description
Check In	Check into the repository for any checked-out and unmanaged cellview in your work area.
Check Out	Check out and place a lock on a cellview, giving you write access and ensuring no one else can modify it.
Discard Check Out	Cancel a locked checkout of a cellview or file and discard any changes.
Tag	Attach a symbolic name (label) to a specific revision of a cellview or file for milestone tracking.
Show Status	Print the version control status of the selected libraries, cells, cellviews, or files.
Version Info	Display the complete revision history of a cellview, with options to revert, export, or use earlier revisions.
History	Display the design management history of the selected file or cellview, including all operations performed.
Merge	Merge a cellview from one branch to the main branch, or to another selected branch.
Miscellaneous	Run site-customizable SOS commands on the currently selected objects.
List Revision Diffs	View or save the revision diff summary of selected labels in HTML, Text, or CSV format.
Show Labeled	View or save the label report summary of selected labels in HTML, Text, or CSV format.
Manage Hierarchy	Perform operations on an entire design hierarchy starting from a selected top-level cellview.

Update	Update your work area from the project repository, with options for selective or full-workarea updates.
Audit Trail	List all or selected actions performed in the project for traceability and review.
Preferences	Customize library management options, audit trail options, RSO options, snapshot options, and view type settings.
Open SOS	Open the SOS GUI window for the work area connected to the current Custom Compiler session.

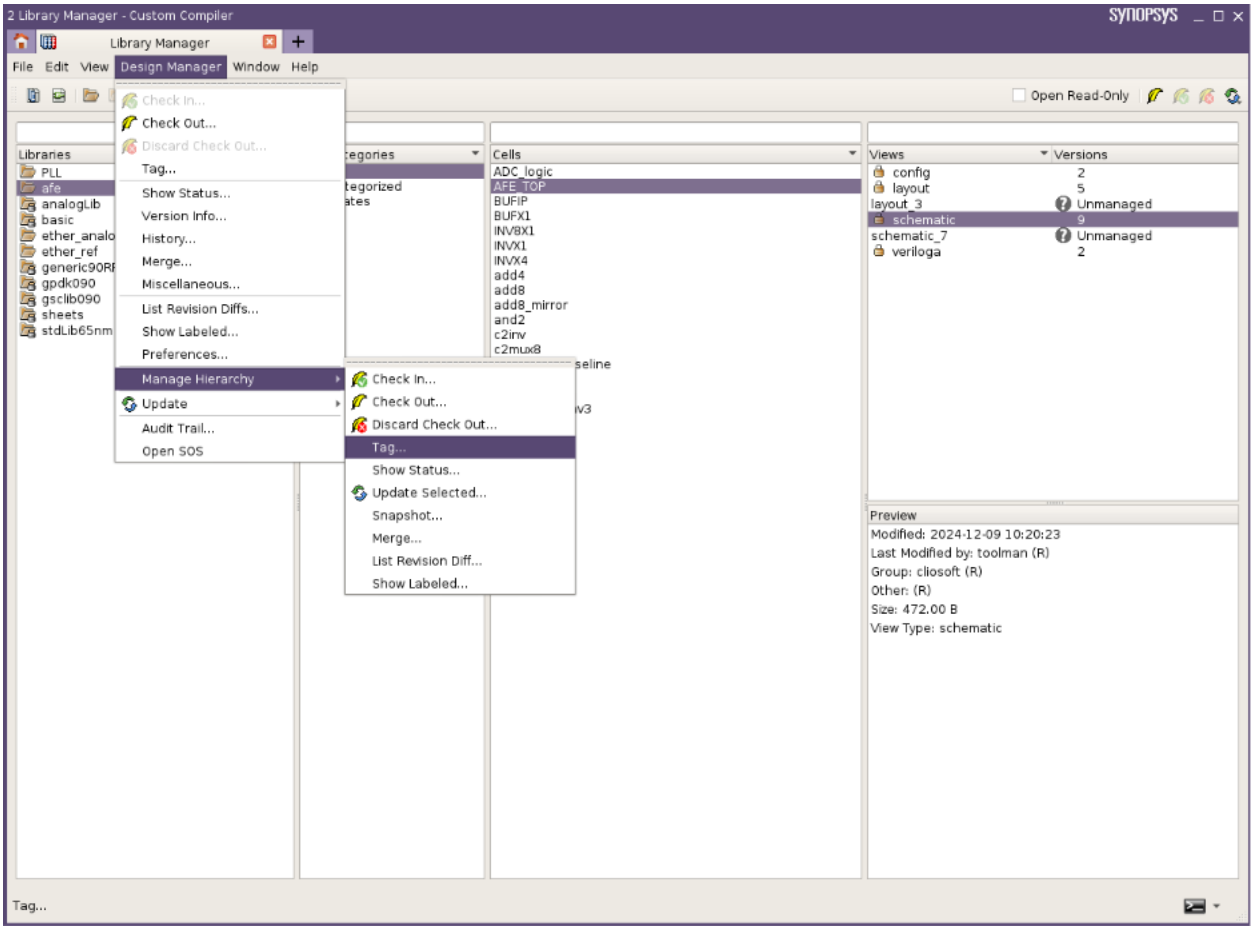

Figure 2. The Designer Manager menu options are logically grouped according to options

Status Icons

SOS communicates the version control status of every managed cellview through icons that appear directly in the Custom Compiler Library Manager. These icons provide at-a-glance awareness of the project state without requiring any additional tool interaction.

Icon	Status	Meaning
Yellow Feather	**Checked out by you**	You have the cellview locked for editing. Other users cannot modify it.
Feather (no lock)	**Checked out (no lock)**	You have checked out the object without acquiring a lock, allowing concurrent editing.
Lock icon	**Checked out by another user**	Another engineer has the cellview locked. You can open it read-only.
Out-of-date icon	**Out-of-date**	A newer revision exists in the repository. You should update your workarea.
RSO mismatch icon	**RSO mismatch**	The revision in your work area does not match your current Revision Search Order. A newer revision matching your RSO is available.
Question mark	**Not managed**	The object is not under SOS revision control. It can be added via Check In .

The toolbar in the Library Manager window also provides quick-access buttons for the most frequently used design management commands, enabling one-click access to check-in, check-out, and discard checkout operations.

Right-Click Context Menu

The most frequently used SOS operations are also available through the right-click context menu on libraries, cells, and views in the Library Manager. The context menu includes Check In, Check Out, Discard Check Out, Tag, Show Status, Update Selected, Snapshot, and Merge. The Manage Hierarchy submenu is also accessible from the right-click context menu, enabling hierarchical operations directly from the Library Manager.

Core Designer Workflows

The SOS Custom Compiler integration is designed to feel natural within the standard Custom Compiler design flow. Engineers work with libraries, cells, and cellviews using the same Custom Compiler conventions they are accustomed to, with SOS providing the underlying data management infrastructure.

Setting Up the Work Area

The first step in using SOS within Custom Compiler is configuring the work area. Work areas are isolated copies of the project libraries that give each designer a private sandbox for edits. After the initial setup through the SOS user interface, designers connect their Custom Compiler session to the work area. All SOS-managed libraries in the work area appear in the Custom Compiler Library Manager with status icons indicating the revision control state of each object.

Typical Daily Task Flow

A typical daily workflow for a Custom Compiler designer using SOS follows a five-step cycle.

Step 1: Check out a cellview. The designer selects the cellview in the Library Manager and invokes Design Manager > Check Out. The Check Out dialog box displays all available cellviews [figure 3]. The designer selects the objects to check out, optionally adds a description, and clicks OK. The lock is applied in the repository, and the designer receives write access to the cellview in their workarea. While checked out, no other user at any site can modify the same cellview.

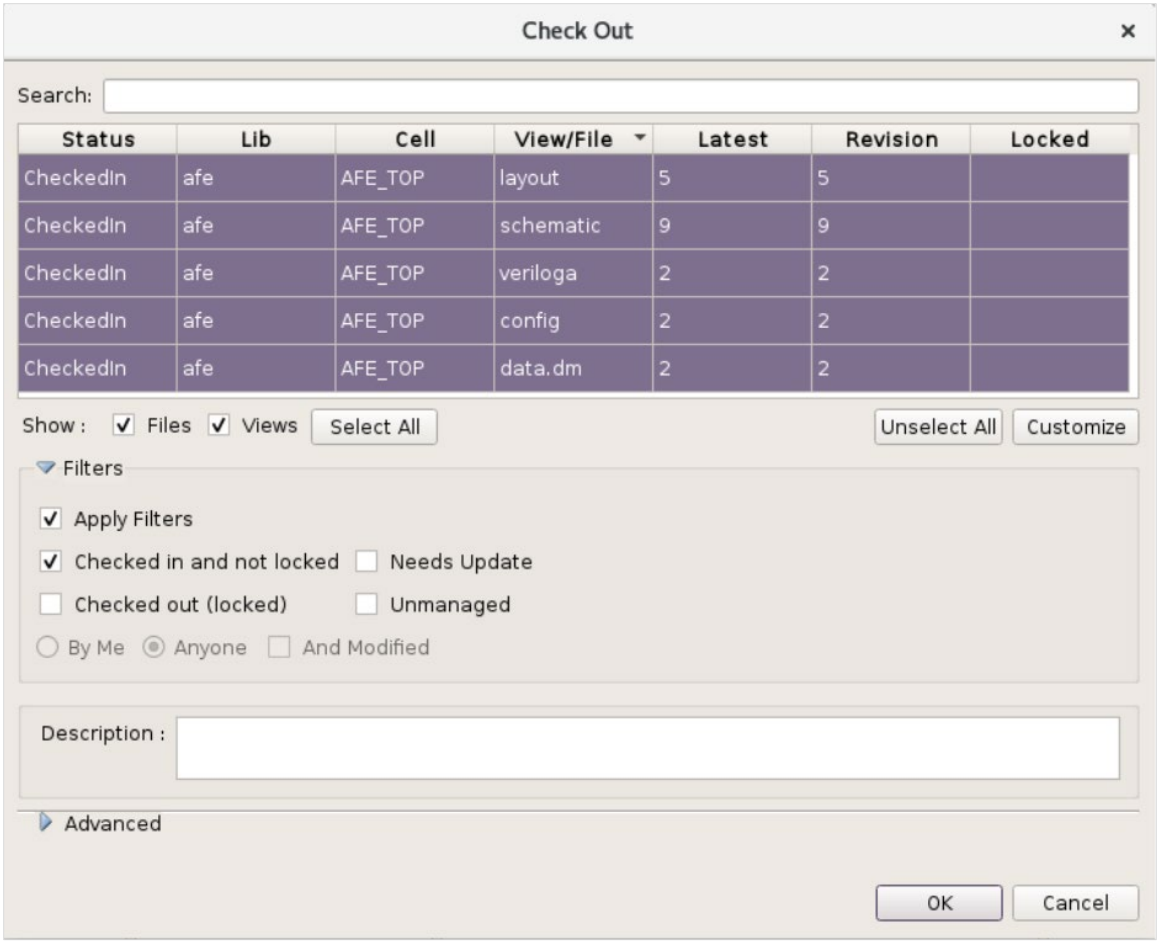

Figure 3. Checking out files with precise controls and filters

Step 2: Edit the cellview and save changes. The designer works normally in the Custom Compiler schematic or layout editor, saving changes to disk periodically [figure 4]. All saves are local to the designer's workarea and do not affect other users.

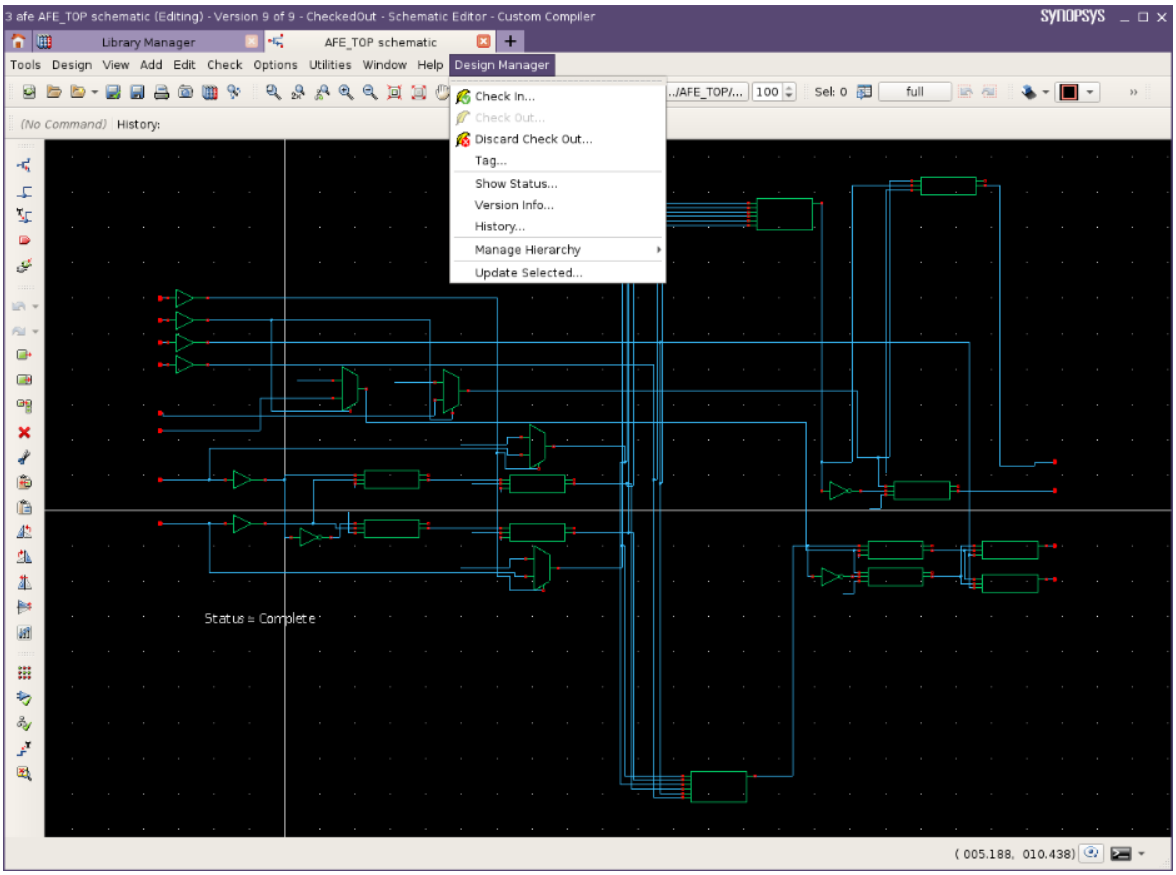

Figure 4. Quick access to common data management operations directly within the editor

Step 3: Verify the changes. The designer runs simulations, DRC, LVS, or other verification steps on the modified cellview before committing changes to the repository.

Step 4: Check in the cellview. From the Library Manager, the designer selects Design Manager > Check In. The Check In dialog displays the objects being committed, allows the designer to enter a description of the changes, choose whether to discard unmodified checkouts or force check-in, and optionally apply a tag or link to an issue tracking ID. Clicking OK creates a new revision in the repository and releases the lock, making the updated cellview available to the rest of the team.

Step 5: Update the workarea. The designer periodically synchronizes their workarea to pull in changes from other engineers. The Design Manager > Update menu provides multiple options: Update Selected (for specific libraries, cells, or views), Update WorkArea (for the entire workspace with RSO configuration), and Change Workarea RSO (for modifying the revision search order). A Preview button allows the designer to review pending changes before applying them.

Designers may repeat this cycle once or several times per day, depending on the scope of changes in each iteration.

Canceling a Checkout

If a designer decides to discard their edits, they can cancel the checkout using Design Manager > Discard Check Out. The Discard Check Out dialog offers multiple scope options: discard checkouts by the current user in the current workarea, discard by the current user in any workarea, or (for administrators) discard checkouts by any user. The “Do NOT discard if modified” safety option is available and can be selected to prevent accidental loss of work.

Adding New Cellviews to a Project

Cellviews that do not yet belong to an SOS project are labeled “Unmanaged” in the Check In dialog. Designers can add new cellviews to the project from the Library Manager by selecting the parent library or cell and using the Check In command. The Check In dialog lists both checked-out and unmanaged objects, and the designer can select which items to commit. The Show filters at the bottom of the dialog allow toggling between Files (library-level files such as data.dm) and Views (cellviews under libraries).

Version History, Revert, and Recovery

SOS maintains the complete revision history of every managed cellview. The Version Info dialog, accessible from Design Manager > Version Info, displays every revision with its author, timestamp, change description, and associated tags or snapshots. The designer can filter the revision list by tags to quickly locate milestone revisions.

From the Version Info dialog, a designer can perform several recovery operations:

- **Use Rev:** Temporarily switches the designer’s workarea to use an earlier revision of the cellview without affecting other users or creating a new repository revision.
- **Revert:** Creates a new revision in the repository that is identical to a selected earlier revision, effectively rolling back the object for all users.
- **Export:** Extracts a specific revision as a new, unmanaged cellview named CELLVIEW_REVISION_NUM, providing an editable copy outside of SOS control.
- **Delete Rev:** Permanently deletes the selected revision from the repository. This operation is irreversible from within SOS.
- **Check Out:** Check out the latest revision directly from the Version Info dialog for immediate editing.

The History command (Design Manager > History) provides a complementary view, displaying the complete design management history of a cellview including all operations (check-in, check-out, tag, untag, discard) performed on the object, with timestamps and user information.

Hierarchical Operations

In complex analog and mixed-signal designs, individual cellviews do not exist in isolation. A top-level schematic references sub-blocks, which in turn reference leaf cells, forming a deep design hierarchy. SOS provides hierarchical operations that traverse this hierarchy and apply a data management operation to the entire design tree in a single step.

The Design Manager > Manage Hierarchy submenu exposes hierarchical variants of all core SOS operations: Check In, Check Out, Discard Check Out, Tag, Show Status, Update Selected, Snapshot, Merge, List Revision Diffs, and Show Labeled [figure 5].

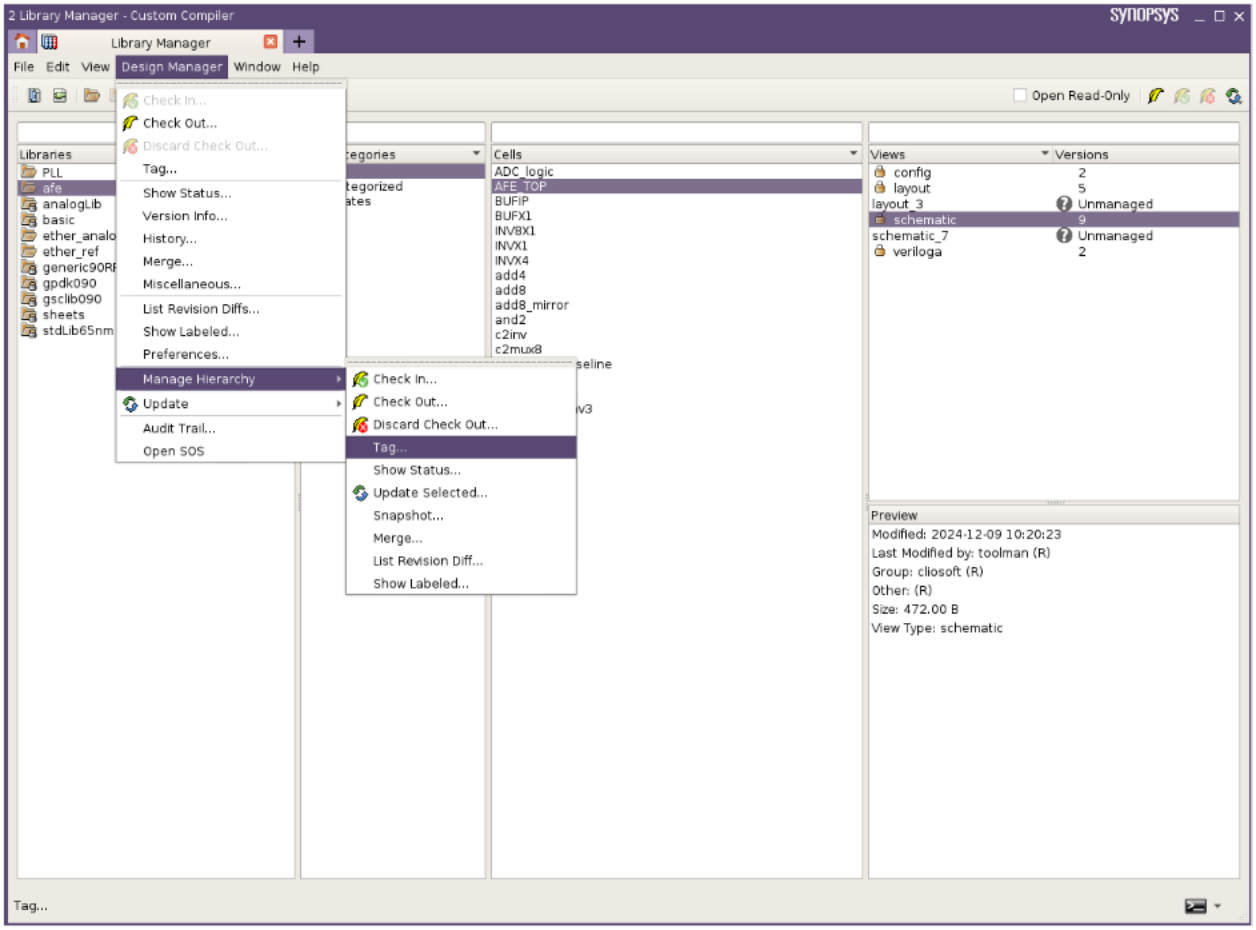

Figure 5. Manage Hierarchy submenu exposes hierarchical variants of all core SOS operations

Hierarchy Traversal

When a hierarchical operation is invoked, SOS presents the Descend Hierarchy dialog. The designer selects the traversal mechanism, which determines how SOS discovers the objects in the hierarchy. Two mechanisms are available:

- **Select view by type:** Choose from available view types (maskLayout, schematic, schematicSymbol, netlist, wafer, etc.) to include all cellviews of those types in the hierarchy.
- **Select view by name:** Specify a list of cellview names separated by spaces for precise control over which views are included in the traversal.

The designer also selects which libraries to descend into while traversing the hierarchy. The “Warn about missing cells” option can be enabled to identify cellviews that are referenced in the hierarchy but missing from the workarea.

After traversal, SOS presents the standard operation dialog (e.g., Tag, Check Out) pre-populated with all discovered objects. The designer can refine the selection using filters before confirming the operation.

Practical Example: Hierarchical Tagging

Consider a design engineer who has completed schematic design for an AFE (Analog Front End) block with 15 sub-cells. Rather than manually tagging each sub-cell individually, the engineer selects the top-level AFE cellview, invokes Design Manager > Manage Hierarchy > Tag, selects “schematic” as the view type, selects the relevant libraries to descend into, and clicks OK. SOS traverses the hierarchy, discovers all 15 schematic views, and presents them in the Tag dialog. The engineer selects the “design_done” tag, clicks OK, and the entire schematic hierarchy is tagged in a single operation. A layout engineer in a different timezone can then configure their RSO to pull the “design_done” versions automatically.

Labels, RSO, and Automated Design Handoffs

The combination of Labels and Revision Search Order (RSO) is the mechanism by which SOS automates design handoffs within the Custom Compiler environment. This capability eliminates the need for email-based notifications, manual file transfers, or ad hoc coordination between design, layout, and verification engineers.

How RSO Works in Practice

Each user workarea has an associated RSO, which is an ordered list of labels. When a workarea is updated (via Design Manager > Update), SOS resolves which revision of each object to retrieve by walking the RSO from highest to lowest priority. For a given object, SOS checks each label in priority order: if a revision tagged with that label exists, it is used; otherwise, SOS moves to the next label. The built-in label “main” always resolves to the latest revision.

For example, with an RSO of [design_done, gold, main], SOS first checks whether a revision tagged “design_done” exists for each object. If so, that specific revision is placed in the workarea. If no “design_done” revision exists for a given object, SOS falls back to “gold,” and if that is also absent, it retrieves the latest revision. By manipulating the RSO, designers can precisely control which version of every object appears in their workspace.

Workarea Configuration for RSO

The SOS Custom Compiler integration supports advanced workarea configuration that allows different RSOs for different view types within the same workarea. For example, a verification engineer might configure their workarea to use “design_done” for schematic views while using “layout_done” for layout views.

RSO management is provided through multiple dialogs in the Update submenu:

- **Update WorkArea:** Provides a comprehensive interface for selecting the workarea, project, and constructing an RSO from available view types, tags, branches, and snapshots. Supports update options including “Changed CellViews And Files Existing In Workarea Only,” “Get New Cells And Views Into Workarea,” “RSO Only,” and “Force All.” A Preview button allows inspection before committing.

- **Change Workarea RSO:** Allows modification of the RSO without requiring a separate update step. The designer can re-order labels, add new labels, and optionally update the workarea with the new RSO.

- **Change Library RSO:** Provides a three-panel interface for per-library RSO configuration: library selection, view-class-to-RSO mapping (update rules), and label/RSO assignment. This enables fine-grained control where different libraries within the same workarea can follow different RSOs.

- **Show Library RSO:** Displays the effective RSO for a specific library or for all managed libraries in the workarea, including whether the RSO was applied from the library, project, parent, or ancestor level.

Workarea configurations can be saved to external files or to the project itself (Design Manager > Update > Save Workarea Config / Load Workarea Config). This enables CAD managers to define standard configurations and distribute them across the team, ensuring consistent workspace setup.

Library-Level Diff Reports

Beyond cellview-level comparison, SOS provides library-level diff reports that summarize all changes across an entire library between two RSO states. These reports are accessible from Design Manager > List Revision Diffs (at the library level) or from Manage Hierarchy > List Revision Diffs (across a full design hierarchy).

The List Revision Diffs dialog allows selection of “From” and “To” RSO pairs, with each side supporting independent object type filtering (Any, schematic, simulation, layout) and label selection. Additional configuration options include:

- **Label scope:** “Show library labels only” restricts the label list to labels applied to the selected library, while “Show all project labels” displays all user-defined labels across the project.
- **Time filtering:** The “Specified Time” option allows running the diff at a specific historical point in time.
- **Content filtering:** The Filter dropdown controls which objects appear in the report: show added, deleted, and updated objects; show only newly added objects; show only deleted objects; show only updated objects; show objects that have not changed; or show all objects.
- **Output formats:** Results can be displayed as a table view, HTML, text, or CSV. Reports can optionally be saved to a file for archival or external analysis.

The resulting report enumerates every object that differs between the two states, with columns for the object path, “From RSO” revision, “To RSO” revision, check-in author, check-in time, and change description. This capability is particularly valuable for release engineering, where a team needs to understand exactly what changed between two milestones.

Show Labeled Reports

The Show Labeled command (Design Manager > Show Labeled) generates a report of a library and its content revisions matching a given set of labels. This provides a snapshot view of which revision of each cellview corresponds to specified labels, helping project leads verify label coverage and consistency. The report supports the same output formats and save-to-file capabilities as the List Revision Diffs report.

Branching and Merging

SOS supports branching to enable parallel development paths for cellviews. Branching is initiated from the Check Out dialog: the designer selects an existing branch or defines a new branch, and can optionally set it as the default for future checkouts using the “Make Default” option.

The Merge command (Design Manager > Merge) allows merging cellviews from one branch to the main branch, or to another selected branch. The designer selects the source branch, source revision, and target branch, then chooses the merge type:

- **Overwrite:** Replaces the target branch’s revision with the source branch’s content. Because EDA cellview files are binary, this is the standard merge strategy. After the merge, the cellviews are left in a checked-out state so the designer can review and verify the result before checking in.

- **Record Merge Only:** Records the merge operation in the version history without modifying any files. This is useful for tracking that a merge decision was made, even if the actual file content was handled through other means.

Hierarchical merge is also supported through the Manage Hierarchy > Merge command, which applies the merge across an entire design hierarchy in a single operation.

Audit Trail

The Audit Trail (Design Manager > Audit Trail) generates a chronologically sorted report of all data management operations executed on project data [figure 6]. It provides a comprehensive, filterable history that answers questions like: who checked in this cellview, when was this tag applied, what operations were performed on this library last week?

The Audit Trail dialog supports filtering by:

- **Time range:** Options include “last update time,” “last one year,” “last one month,” “last one week,” “last one day,” or “specific dates.”
- **Operation type:** Selectable categories include create + checkin, checkout, snapshot + tag + branch, delete + move + rename, and addreference + editreference.
- **User:** Filter by specific users using a comma-separated list.
- **Scope:** Audit the entire project or only the selected paths.
- **Branch:** Filter by a specific branch for branch-aware reporting.

The report can be displayed in a grouped transaction view or as individual operations.
Two Tcl variables control the default behavior: sosAuditTrailRunAuditTrailSince controls the default time range, and sosAuditTrailGroupTransactions controls whether operations are grouped.
The Audit Trail report includes buttons to “Update To Time” (roll back the workarea to a specific audit trail timestamp), “Use Prev Revision” (revert a cellview to its state before a selected operation), and “Save” (export the report for external analysis).

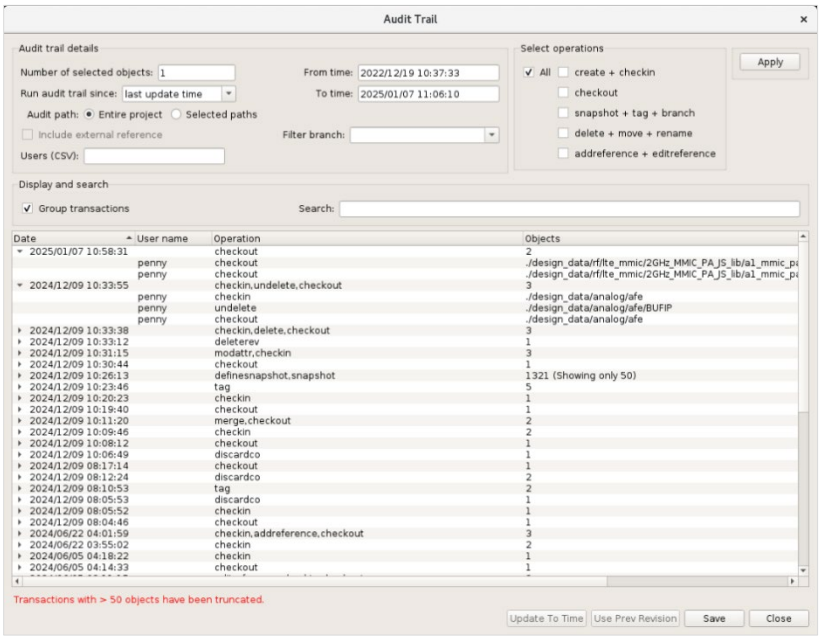

Figure 6. Build and manage a comprehensive audit trail

Customization and Tcl APIs

The SOS Custom Compiler integration provides extensive customization capabilities through the SOS Preferences dialog (Design Manager > Preferences) and through Tcl programming interfaces.

SOS Preferences

The SOS Preferences dialog contains two primary tabs for configuring integration behavior:

General tab: Controls SOS library management enablement, the SOS library management filename, audit trail default options (time range, grouping, transaction count), Change Workarea RSO options (update timing), Change Library RSO display options (console output, viewer display, file export, workarea RSO printing), and snapshot options (ancestor directory inclusion).

View types tab: Configures which view types are displayed in the Change Library RSO dialog. The default set includes dir, file, schematic, simulation, layout, emulation, test, control, HDL, RTL, PNR, STA, IP, and ISO. Teams can add or remove view types to match their specific design methodology.

Preferences can be saved to and loaded from the .sosInit-<username>.tcl file, enabling consistent configuration across sessions and across team members. The “Reset Tab,” “Reset All Tabs,” and “Load Init File” buttons provide quick restoration of defaults or saved settings.

Custom Miscellaneous Commands

The Miscellaneous dialog (Design Manager > Miscellaneous) provides a site-customizable command framework. CAD administrators can add custom commands to the Command dropdown by defining a global Tcl array in the $SYNOPSYS_CUSTOM_SITE/.cdesigner.tcl file. Two keywords are available for command definitions: @SELECTION (substitutes the selected objects) and @POPUP (displays results in a pop-up text editor window).

Additionally, SOS provides a registered callback function (sos::scdMiscellaneousProc) that accepts a list of paths of selected items and returns a result. CAD teams can override this function to implement custom Tcl workflows that integrate with their specific design methodology and process requirements.

Tcl Variables for Behavior Control

The integration exposes several Tcl variables for controlling advanced behaviors, including Show Library RSO output formatting (sosShowLibRsoSaveToFileG, sosShowLibRsoPrintInConsole, sosShowLibRsoPrintWARsoG, sosShowLibRsoDisplayInViewerG), audit trail defaults, and other runtime configuration. These variables can be set in the site-wide or per-user initialization files for consistent behavior across the team.

Multi-Site Collaboration

SOS Core's architecture is designed for teams distributed across multiple geographic locations. The system uses a Primary Repository at the main development site and Cache Repositories at each remote site. This topology ensures that Custom Compiler users at every location experience local-speed performance for all data management operations.

When a designer at one site checks in a new revision, the data is propagated to the Cache Repository, and designers at other sites can begin using the updated cellview immediately upon their next workarea update. The integration is transparent to the Custom Compiler user: the Design Manager menu, status icons, and all operations behave identically regardless of whether the designer is co-located with the Primary Repository or connected through a Cache.

This architecture is particularly valuable for semiconductor design teams that span multiple countries and time zones. A schematic designer in San Jose can tag a hierarchy with "design_done" at the end of their day, and a layout engineer in Shanghai can configure their RSO, update their workarea, and begin physical implementation the next morning with the exact revisions their colleague designated.

Real-World Workflow: A Multi-Engineer Design Scenario

The following scenario illustrates how the SOS Custom Compiler integration supports a realistic multi-engineer design flow from schematic design through layout completion and release.

Project Setup

A project lead, Penny, initializes the SOS project from the SOS GUI. The project repository stores all design data centrally, with Cache Repositories deployed at each engineering site. Penny creates project views to give analog engineers visibility into analog libraries and layout engineers visibility into physical implementation data.

Schematic Design (Engineer: Sheldon)

Sheldon, an analog design engineer, launches Custom Compiler and opens the Library Manager. He selects the "afe" library and invokes Design Manager > Update > Update Selected to synchronize his workarea with the latest project state. The Update dialog identifies two out-of-date cellviews, which Sheldon updates before beginning work.

Sheldon selects the AFE_TOP schematic in the Library Manager and invokes Design Manager > Check Out. The Check Out dialog presents the available objects; he selects the schematic view, confirms, and begins editing. After completing his schematic changes, he saves the cellview and runs simulations to verify functionality.

Satisfied with the simulation results, Sheldon selects the library and invokes Design Manager > Check In. He enters a change description and optionally links the check-in to a Jira issue ID for traceability. SOS creates a new revision and releases the lock.

Sheldon then applies a milestone tag to his entire schematic hierarchy. He selects the AFE_TOP cellview, invokes Design Manager > Manage Hierarchy > Tag, chooses the “schematic” view type in the Descend Hierarchy dialog, selects the relevant libraries to traverse, and applies the “design_done” tag to all schematic views in the hierarchy in a single operation.

Layout Design (Engineer: Amy)

Amy, a layout engineer at a remote site, opens the Audit Trail (Design Manager > Audit Trail) to review recent project activity. She sees Sheldon’s check-in and tagging operations in the report.

Amy opens Design Manager > Update > Change Library RSO and moves “design_done” to the top of her RSO priority list for schematic views. She then updates her workarea. SOS resolves the RSO for each object: schematic views tagged “design_done” are retrieved at the specific revision Sheldon tagged, while other view types fall back to the next label in the RSO.

Amy completes the physical layout, runs DRC and LVS, and checks in her work. She applies the “layout_done” tag to her layout hierarchy using the same Manage Hierarchy > Tag workflow.

Release (Project Lead: Penny)

Penny creates a project snapshot from the SOS GUI, entering “fp2” as the snapshot name. The snapshot freezes the exact combination of revisions across all objects at that moment, providing a reliable baseline that can be recreated at any point in the future for tapeout preparation or post-silicon debugging.

Why SOS for Custom Compiler

The SOS Custom Compiler integration delivers value across multiple dimensions that compound over time as teams grow and projects multiply.

Capability	Value Delivered
Native Custom Compiler integration	Engineers never leave their design environment. All data management operations are accessible from the Library Manager menus, editor menus, and right-click context menus. The learning curve is minimal.
Hierarchical operations	Apply check-in, check-out, tag, update, and other operations across an entire design hierarchy in a single step. Critical for complex analog and mixed-signal designs with deep hierarchies
Labels and RSO	Automate design handoffs between functional roles (design, layout, verification) without email, file transfers, or manual coordination. Each role configures their RSO to pull exactly the revisions they need.
Per-library RSO control	Configure different RSOs for different view types and libraries within the same workarea, enabling fine-grained revision control tailored to each engineer's role.
Links-to-Cache	Lightweight symbolic-link workspaces that consume megabytes instead of duplicating terabytes of project data per engineer. Dramatically reduces storage costs and workspace setup time.
Comprehensive diff reporting	Library-level and hierarchy-level diff reports in multiple formats (table view, HTML, text, CSV) with flexible filtering options for release engineering and milestone tracking.
Audit trail and traceability	Complete, chronologically sorted history of all project operations. Integrated issue tracker linking. Essential for tapeout readiness, post-silicon debugging, and regulatory compliance.
Customizability	Tcl APIs, custom Miscellaneous commands, configurable preferences, workarea configurations, and initialization file hooks allow CAD teams to tailor the integration to their specific design methodology and process requirements.

Scaling Your Approach

SOS Core is designed as the entry point to the broader Keysight Engineering Data Management platform. As organizations grow in team size, number of sites, regulatory requirements, and data maturity, SOS Core provides a seamless upgrade path to SOS Enterprise, which extends the platform with:

- Global, multi-site collaboration with real-time synchronization.
- Enterprise-grade governance including role-based access control, encryption, and automated approval workflows.
- Full lifecycle traceability from schematic through verification and test.
- AI-ready data infrastructure with structured metadata and APIs for ML pipelines and agentic AI workflows.
- IP lifecycle management with catalogs, bills of materials, and standardized reuse processes.

The upgrade preserves all existing project data, configurations, user workflows, and Custom Compiler integration settings. There is no migration penalty.

Conclusion

The Keysight SOS Core integration with Synopsys Custom Compiler transforms Custom Compiler from a standalone design environment into a fully data-managed platform. Engineers gain version control, collaboration, and traceability without leaving their primary tool. Hierarchical operations and the Labels/RSO system automate the cross-functional handoffs that are critical to analog and mixed-signal design flows. The Links-to-Cache architecture ensures that all of this is delivered with minimal storage overhead and local-speed performance, regardless of team size or geographic distribution.

For semiconductor design teams that rely on Synopsys Custom Compiler, SOS Core is a deeply integrated, production-proven design data management solution. Built on the same platform trusted by the world's leading semiconductor companies, it provides a reliable, scalable foundation for design data management in the AI era.

To learn more about Keysight SOS Core and the Custom Compiler integration, visit: EDA Integrations | Keysight