cadence®

Allegro® X Constraint Manager User Guide

Product Version 23.1
September 2023

Last Updated: November 2022

Cadence Design Systems, Inc. (Cadence), 2655 Seely Ave., San Jose, CA 95134, USA.

Contents

1

Welcome to Constraint Manager

Topics in this chapter include

The Allegro Constraint Manager Information Set

The Allegro Constraint Manager information set consists of online books accessible from *Cadence Help* in both HTML and PDF formats. All documentation is accessible from Constraint Manager's help menu.

Refer to . . .	**for this level of information**
Allegro Constraint Manager User Guide (this book)	This book is for users who want to know how to use Constraint Manager in the design flow. This book complements the information in the *Allegro Constraint Manager Reference*.
Allegro Constraint Manager Reference	This book contains descriptions and procedures for all commands, organized by menu-sequence. Information about worksheet cells is also included. If you click help in a dialog box or if you highlight a menu command and press `F1`, the appropriate command description from this book appears. This book complements the information in the *Allegro Constraint Manager User Guide*.
Allegro Platform Constraints Reference	This book contains information describing the constraints architecture, and it includes reference information for each constraint.
Allegro Platform System Connectivity Manager User Guide	This book contains information describing the hierarchical, lower-level constraints used in the Constraint Manager-enabled, high-speed flow.

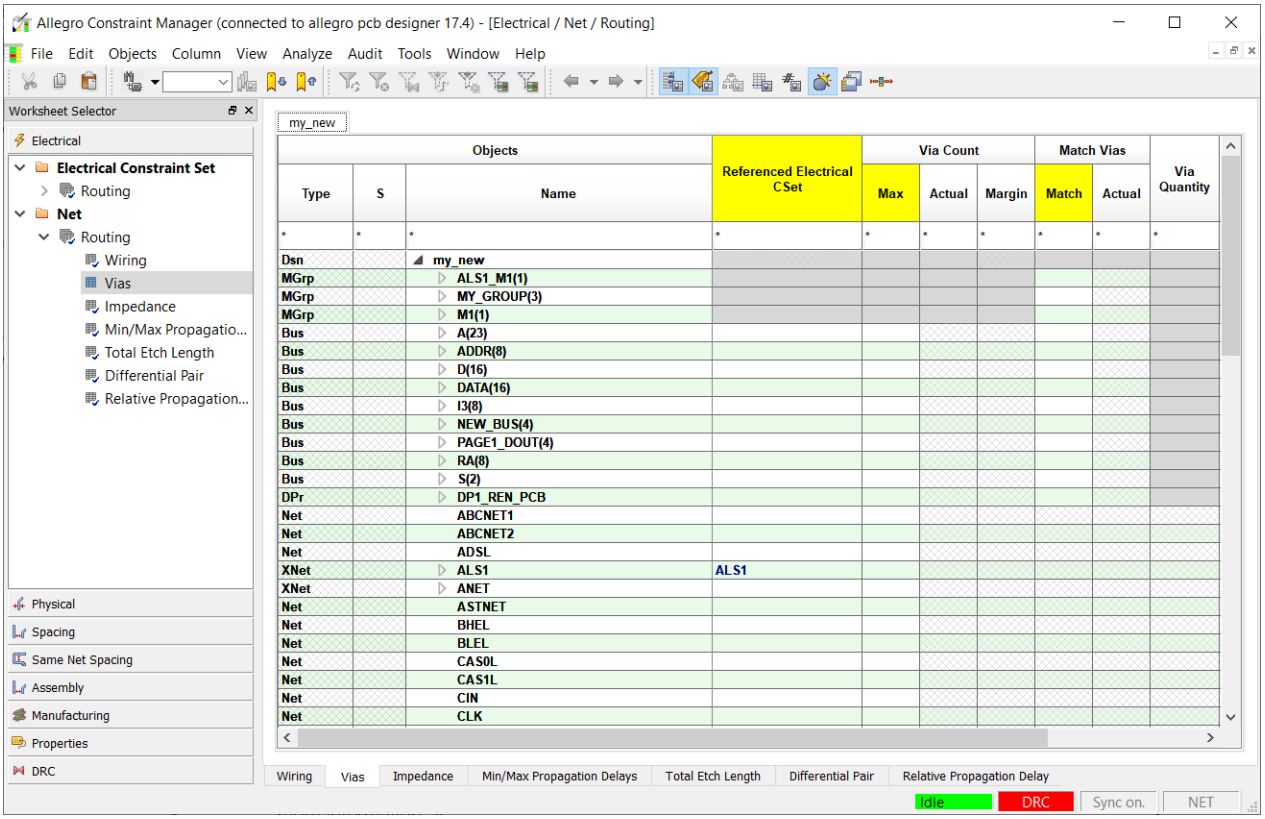
What is Allegro Constraint Manager?

Allegro Constraint Manager is a cross-platform, workbook- and worksheet-based application used to manage constraints across all tools in the Cadence PCB and IC Package design flow.

Constraint Manager lets you define, view, and validate constraints at each step in the design flow, from design capture (in Allegro Design Entry HDL or System Connectivity Manager) to floorplanning (in Allegro PCB SI) to design realization (in Allegro PCB Editor). You can also use Constraint Manager with SigXplorer to explore circuit topologies and derive electrical constraint sets which can include custom constraints, custom measurements, and custom stimulus.

Note: Figure 1-1 depicts Constraint Manager worksheets as launched from PCB Editor, OrCAD PCB Editor, or APD. The worksheet hierarchy is different for Constraint Manager when launched in exploration mode, from Allegro Design Entry HDL, or from System Connectivity Manager.

Figure 1-1 The Constraint Manager User Interface



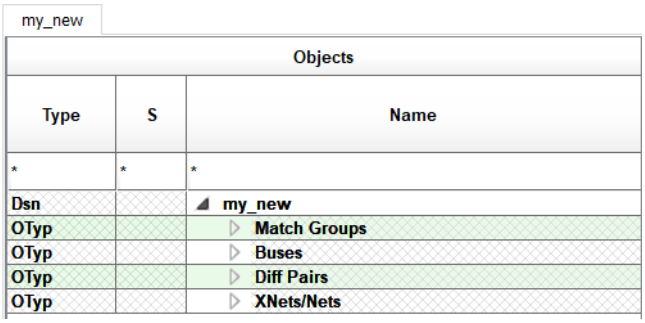
Constraint Manager provides familiar user interface controls. See Table 1-2 on page 32 for more information.

In Constraint Manager, you work with objects and constraint sets, which capture your design requirements.

Constraint Manager organizes constraints and Constraint Sets into the *Physical*, *Spacing, Same Net Spacing,* and *Electrical* domains. You then assign the appropriate constraint set to objects in your design, changing references (or re-defining the currently assigned constraint set) as your design requirements change. A constraint set can be referenced by any number of objects in your design.

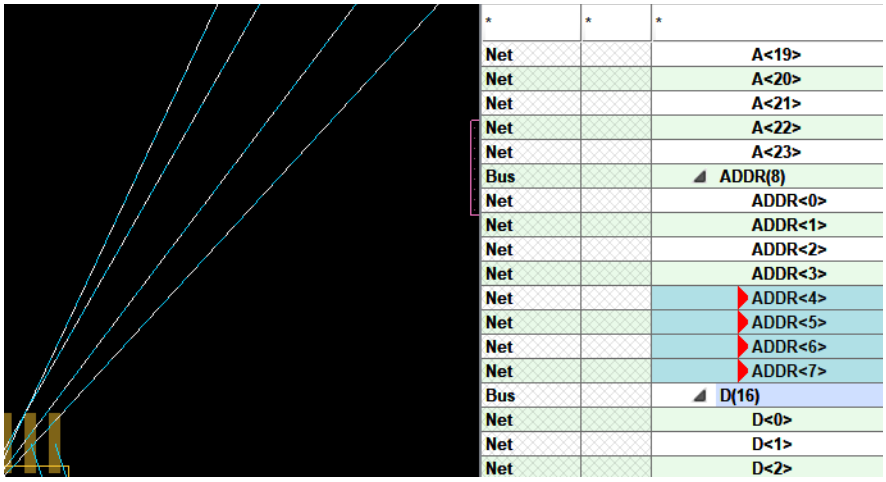
Constraint Manager affords you the following features and benefits:

Table 1-1 Constraint Manager Features

Feature	Benefit
Object Grouping	You can organize objects into easily-managed units, such as a *Net Class*, *Bus*, *Differential Pair*, or *Match Group* to make it easier to apply constraints to member objects.
Object Group Type	You can group to view all the objects of a particular type. For example, all the differential pairs can be grouped together under *Diff Pairs* object type. This grouping reduces the number of rows and provides a tidy view of a worksheet.  To work in legacy mode, set display setting for the *Object Type Divider* option in the *View – View Options* menu.
Conceptual Definition	You can define constraints in a Constraint Set and later apply those constraints to net-related objects.
Redefinable Constraints	Rather than changing individual net-related constraints one-by-one, you can redefine a constraint set and all objects that reference that constraint set get updated all at once.

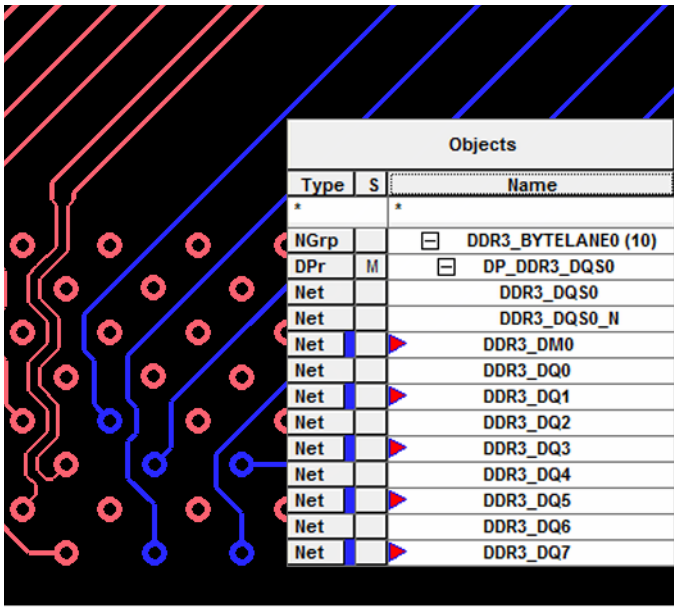
Feature	Benefit
Cross-Probing	■ You can run Constraint Manager with companion tools such as Allegro Design Entry HDL, Allegro SI, or Allegro X Advanced Package Designer and select a net in Constraint Manager and see the associated object update dynamically in the schematic, floorplanner, or layout, respectively. Conversely, Constraint Manager updates its values when they are modified in a companion tool. ■ When you cross probe cells that contain constraint violations, the respective DRC marker (bowtie) becomes highlighted in the design entry- or PCB Editor, or in APD. ■ You can cross probe the nets connected to a symbol from PCB Editor to Constraint Manager. When you select a symbol and right-click to choose *Highlight associated nets* command all the nets become highlighted in Constraint Manager. For more information see `hilight sym net` command.
Topology Exploration	You can access SigXplorer from Constraint Manager to schedule pins and derive generic or net-specific constraints, which may include custom constraints, custom measurements, and custom stimulus. The resulting topology template data can be imported into Constraint Manager as an Electrical CSet.
Design Reuse	You can group constraints that satisfy a specific design requirement into an constraint set, which can be referenced within the active design or exported for reuse in a subsequent design.
Cloning Constraints	In addition to importing constraint sets or creating them from scratch, you can copy it, modify its parameters, and save it under a new name.
Analysis	Constraint Manager performs design rule checks, and simulations as necessary, to analyze the design. Analysis results are communicated by DRC markers, results populated in worksheet cells, simulation waveforms, and reports. Analysis results (actuals) can be compared to defined constraints to derive margins.
System-level Constraints	Constraint Manager can capture board-to-board interconnect constraints.
Persistent Storage	Constraint Manager maintains constraint information in either the board or the schematic database.

Feature	Benefit
Net, Component, and Pin Properties	The *Properties* workbooks let you add and edit certain properties for nets, components, or pins.
Customizable User Interface	You can create a custom workbook and worksheets that suit your work habits.
Highlight and coloring functionality	Constraint Manager displays a red triangle to the left of the object name, when the object has been highlighted in the associated application.

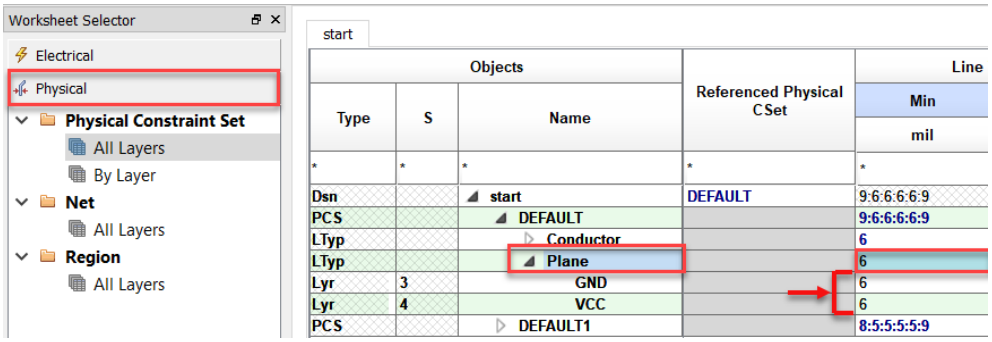


Dehighlighting the object in the application removes the triangle in Constraint Manager.

Feature	Benefit
	Constraint Manager displays a uniquely colored rectangle to the right of the object type when the object has been uniquely colored in the associated application. In addition, a red triangle is displayed to the left of the object name.



Hierarchical Layer Support	The Physical and Spacing Constraint Sets support Hierarchical Layer Types *Conductor* and *Plane*. You can apply constraints only to hierarchical layer level and the values are inherited by the child layers of each group type.



To work in legacy mode, uncheck *LayerType* option in *Objects – Filter* menu.

Accessing Constraint Manager

You access Constraint Manager in Exploration mode through the Windows *Start* menu or by entering `consmgr` in a Unix or Linux shell.

Constraint Manager can also be invoked from a host tool as follows:

From this tool	Choose this menu command
Allegro PCB Editor, Allegro X Advanced Package Designer, or Allegro SI	*Setup – Constraints –* *Electrical* *Physical* *Spacing* *Constraint Manager*
Allegro Design Entry HDL	*Tools – Constraints – Edit*
System Connectivity Manager	*Design – Edit Constraints*

You can also click the Constraint Manager icon in the host tool's toolbar.

Constraint Manager maintains constraint information in the board database when used with Allegro PCB or SI, in the package database when used with Allegro X Advanced Package Designer, or in the schematic database when used with Allegro Design Entry HDL.

Important

The appearance of the Worksheet Selector, worksheets, and commands differ depending on whether Constraint Manager is launched in Exploration mode, invoked from a front-end application, or a backend-application. For example, *By Layer* view of *Physical* and *Spacing* cells is not available in Constraint Manager, when launched from OrCAD PCB Editor or Allegro PCB Editor, Performance L option.

The name of the tool from which you launch Constraint Manager appears in the banner atop the Constraint Manager user interface. For example:

```
Constraint Manager (Connected to Allegro Design Entry HDL)
```

See Chapter 6, "Using Constraint Manager with Other Tools Across the Allegro Platform" for using Constraint Manager with other Cadence tools.

Constraint Manager launched from Allegro PCB Editor, Performance L, and OrCAD PCB Editor

This manual covers all functionality available in Constraint Manager when invoked from Allegro PCB Editor. When invoked from Allegro PCB Design L (legacy) or OrCAD PCB Designer Professional or Standard, Constraint Manager launches with these limitations:

■ Scripting	Scripting is limited to only the supported commands. **Note:** The Constraint Manager scripting system reads data from its user interface. When you open Constraint Manager in the non-graphical mode, it does not work for worksheet-related operations. Populate the worksheets before recording a script.
■ Match Groups	You can define Match Groups only in net-level worksheets; you cannot define match groups at the Constraint Set-level.
■ Pin Pairs	You can define pin pairs only in net-level worksheets; you cannot define pin pairs at the at the Constraint Set-level
■ Signal Integrity	Signal integrity analysis is not supported.
■ Timing	Timing analysis is not supported. The *Timing* workbook has been removed.
■ Custom measurements and custom stimulus	Custom measurements and custom stimulus are not supported. The *Custom Measurements* tab (*Analyze – Analysis Modes*) has been removed; only the *DRC Modes* tab remains. The *Custom Measurements* workbook is not visible.
■ Crosstalk DRC	Crosstalk analysis is not supported. The `max xtalk` and `max peak xtalk` design rule checks have been removed from the *Analysis Modes* dialog box.
■ Topology Templates	Topology import and export are not supported. As such, the *Tools* menu has also been removed prohibiting access to topology exploration tools including SigXplorer and SigWave.
■ Analysis	Simulation-based analysis is not supported. Only design rule checks can be performed.
■ Xnet Creation	You create an Xnet (extended net) through a signal model using Allegro PCB Designer or higher.

PCB Performance Series L	■ In the *Electrical* domain, custom measurements, pin delay, and Z-axis delay are not supported. You also cannot control same net crosstalk and parallelism checks. ■ *By Layer* view of *Physical* and *Spacing* cells is not supported. ■ Ratsnest Bundle worksheets are not supported. ■ Microvias are not supported.
PCB Editor	■ In the *Electrical* domain, custom measurements, pin delay, and Z-axis delay are not supported. You also cannot control same net crosstalk and parallelism checks. ■ For the *Physical*- and *Spacing*-domains, regions, pin pairs, Xnets, differential pairs, buses, and by-layer worksheets are not supported. ■ Ratsnest Bundle worksheets are not supported. ■ Microvias are not supported.
OrCAD PCB Editor	■ *By Layer* view of *Physical* and *Spacing* cells is not available. ■ Ratsnest Bundle worksheets are not supported. ■ Microvias are not supported.

When you select an object in Constraint Manager and right-click, a context pop-up menu appears. Keep in mind that this guide depicts all available options. If you launched Constraint Manager from Allegro PCB Editor (Performance), or OrCAD PCB Editor, some options will be removed or dimmed to inhibit functionality.

Constraint Manager launched from Allegro Physical Viewer

PCB collaboration tools lack constraint management access, yet companies with co-design partners may require design constraint information as specified by contract or agreement. Use Constraint Manager in conjunction with Allegro Physical Viewer as a back-end validation tool that lets design partners view electrical constraint information and analysis results and communicate it without requiring interpretation or conversion if an Allegro flow is used.

When invoked from the Allegro Physical Viewer *Setup* menu, read-only mode Constraint Manager launches with a limited functionality set. You can view the constraint information that a `.brd` file contains. All constraints appear in native delay values (for example, not a length only Performance mode). You cannot modify or export these constraints as certain menu functionality is disabled: right mouse buttons will not allow you to create, modify, or delete objects. *Print* and *View* menu options are available.

Read-only mode Constraint Manager includes all worksheets. However, SigWave or simulation actuals data are unavailable. *Actual* and *Margin* information is available for the constraints based on the design's current state.

Although Allegro Physical Viewer does not let you change DRC modes, as they are inherited from the board, you can run DRC from Allegro to display *actual* data in Constraint Manager, which changes the database; then save it in Allegro Physical Viewer.

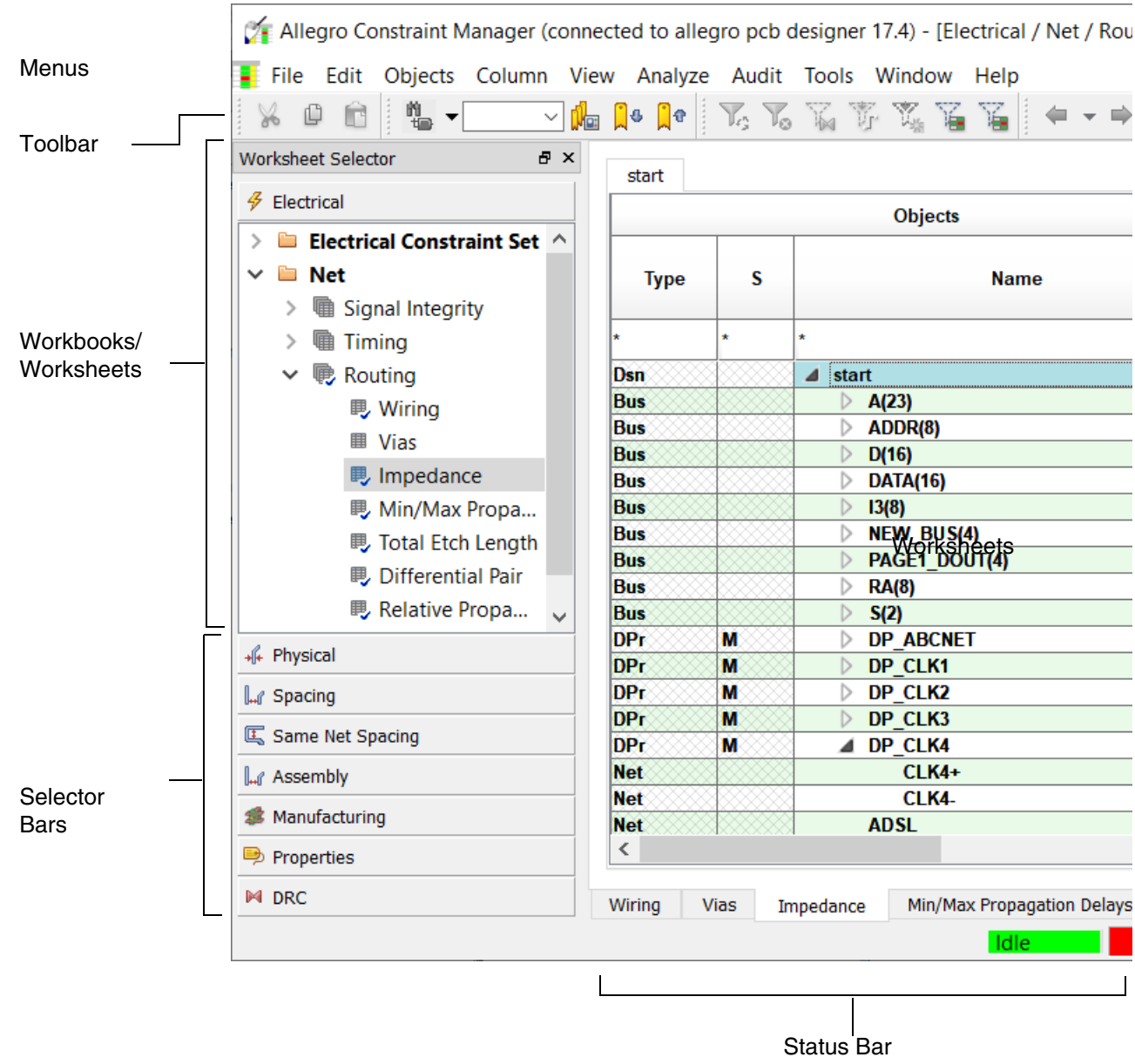
Domains, Workbooks, Worksheets, and Cells

The Constraint Manager workspace (see Figure 1-2, and Figure 1-3) contains the following components.

The:

- *Menus* for command access
- *Toolbar* for quick command access
- *Selector Bar* for switching among domains and DRC and Properties Workbooks
- *Worksheet Selector* for selecting the appropriate worksheet
- *Type* column for identifying the type of object in the *Objects* column
- *Worksheets* for capturing, editing, and validating constraints
- *Status Bar* for feedback on object selection and constraint processing
- *DRC Status* indicator for checking the state of design rule checking

Figure 1-2 The Constraint Manager workspace



Note: When you select an object in Constraint Manager and right-click, you can also access commands from a context-sensitive, pop-up menu.

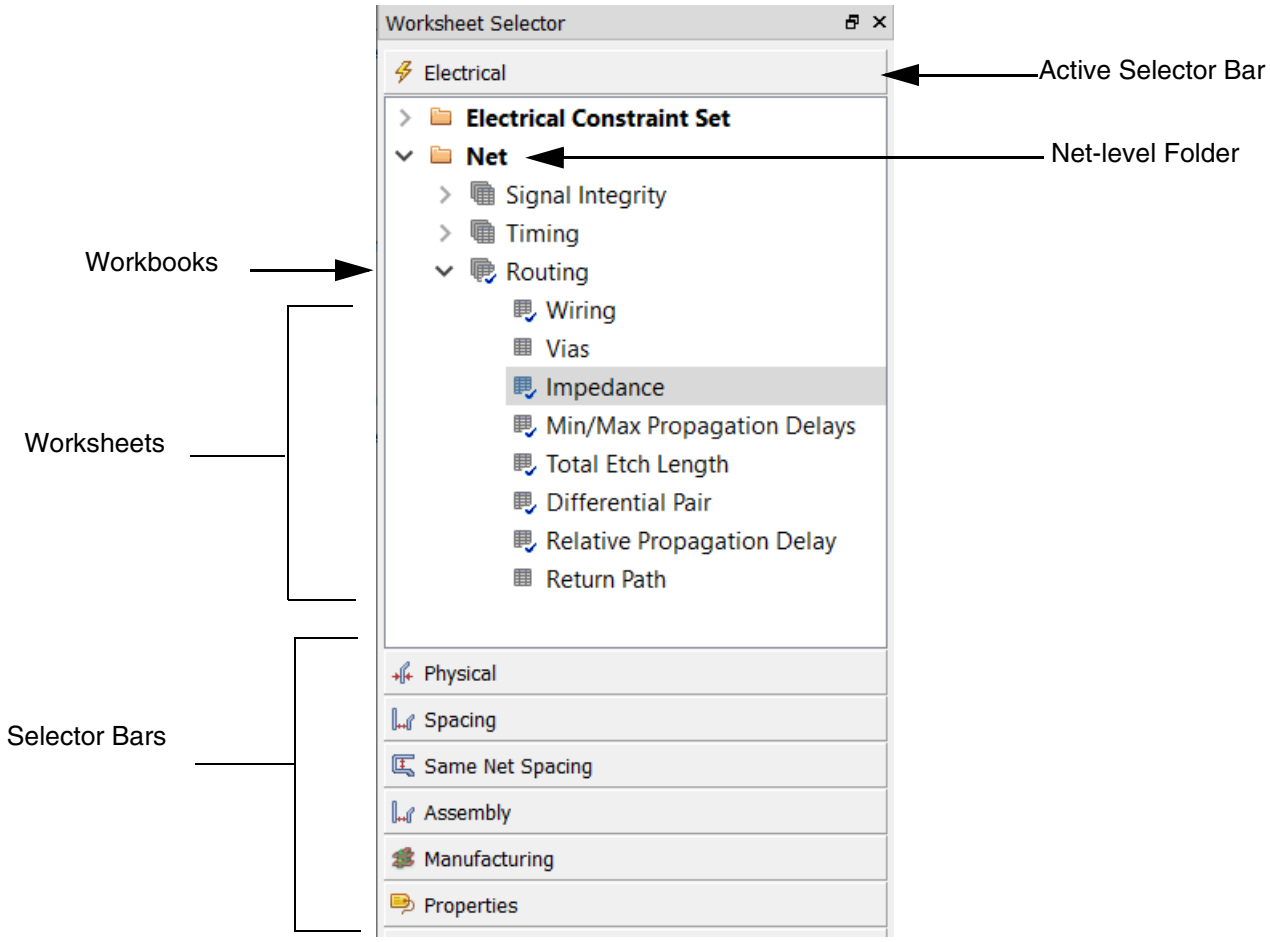
Tip

The *Status Bar* provides key information about cell contents, the state of objects, error conditions, and conditions and processes in your design. When in doubt, consult the status bar.

The Worksheet Selector

Use the *Worksheet Selector* to access the appropriate worksheet that you want to work in. Selector Bars let you access individual constraint worksheets, properties worksheets, and DRC worksheets, which you access by clicking on a *Selector Bar*. You can also undock and reposition the *Worksheet Selector*.

Figure 1-3 Worksheet Selector



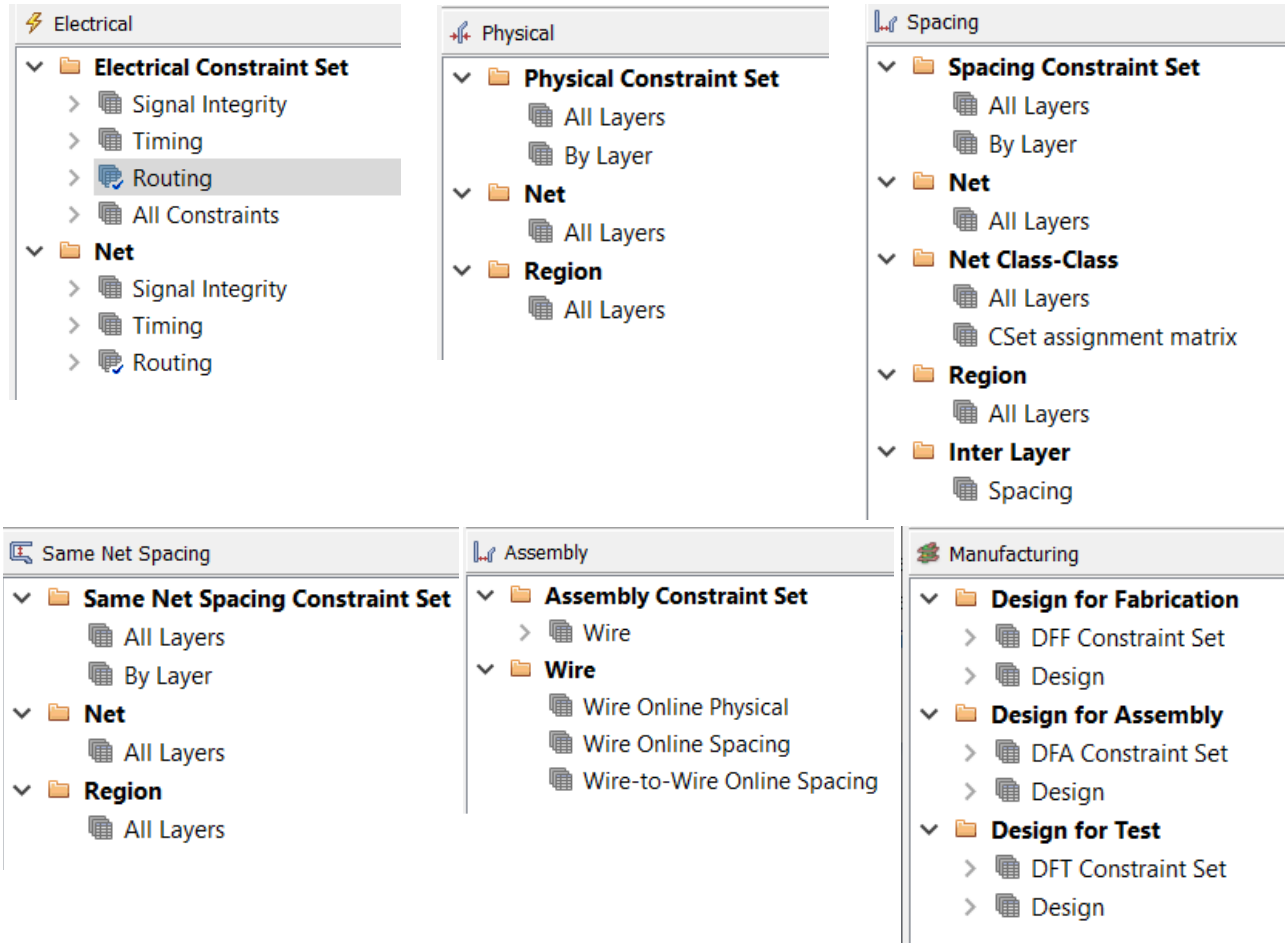
Tip

Grab the border of the *Worksheet Selector* and reposition it to get a full view of workbook and worksheet selector nodes (as shown).

Domain Selector Bars

Constraint Manager organizes constraints, and constraint sets, by domain: *Electrical*, *Physical*, *Spacing,* and *Same Net Spacing*. You access each domain by clicking on the appropriate *Selector Bar*, which is located at the bottom of the *Worksheet Selector* (see Figure 1-3).

Figure 1-4 Worksheet Hierarchy



- In the *Constraint Set Folders* for all domains, you define generic rules and you create generic object groupings. You can later assign these rules to the appropriate net-related objects in your design.
- In the *Net* folders for all domains, you can create net-specific object groupings, and you can define certain net properties. In the *Electrical* domain, you can also create a constraint set based on the characteristics of a net object.

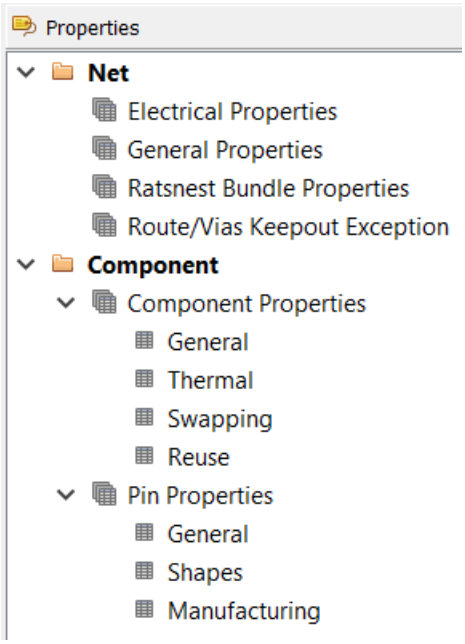
- In the *Physical*, *Spacing* and *Same Net Spacing* constraint folders, worksheets based on layer, or by all layers, contain *Nets*, *Net Classes*, and *Regions*.

Important

By Layer view of *Physical* and *Spacing* cells is not available in Constraint Manager, when launched from OrCAD PCB Editor or Allegro PCB Editor, Performance L option.

Properties Selector Bar

Use the *Properties* selector bar to manage net, component, and pin properties.



The *Net* folder provides you with a quick glance of electrical and general properties. Some cells in these worksheets cannot be edited.

The *Component* folder provides component coordinates, based on placement information, source data for third-party thermal analysis tools, and part definitions. Also included are electrical, thermal, and pin fabrication data. Some cells in these worksheets cannot be edited.

Important

System Connectivity Manager also provides component worksheets where you can define and edit these properties.

See the Allegro Platform Properties Reference for more information on component properties.

DRC Selector Bar

DRC domain in Constraint Manager has been replaced by DRC Browser. You can access the browser from within the DRC worksheet or from the layout editor *Tools* menu (*Tools – DRC Browser*).

For more information about DRC Browser, refer to the *browse drcs* command.

Workbooks

Once you expand a parent *Object Type* folder, workbooks organize objects by design discipline. For example, the *Electrical* domain contains the *Signal Integrity*, *Timing*, *Routing,* and *Custom Measurements* workbooks. Also in the *Electrical* domain, the *All Constraints* workbook in the *Electrical CSet* folder consolidates constraints from all worksheets to give you a global view. Subordinate to the *All Constraints* workbook is the *User Defined* folder, which contains constraints that you have defined in SigXplorer.

Note: The worksheet hierarchy is different if you launch Constraint Manager in exploration mode, or from Allegro Design Entry HDL or Allegro System Architect.

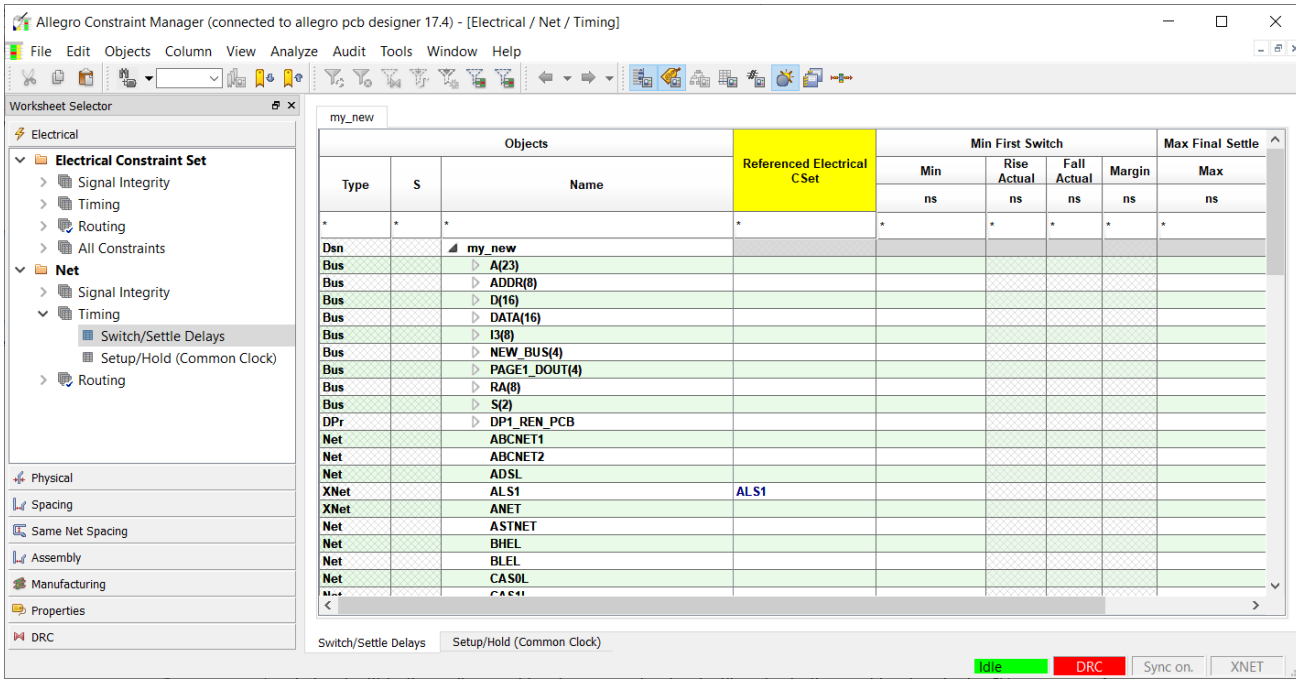
When you select a workbook, all worksheets that belong to that workbook appear in a shared worksheet window. You can use the *Worksheet Selector* to select a worksheet or you can select a worksheet by clicking on the appropriate tab in the shared workbook window. You may have to scroll horizontally to locate the desired worksheet tab.

Note: When you launch Constraint Manager from a physical layout editor, the cells that are in view are populated first. As you scroll other cells into view, the layout tool updates hidden cells as they become visible in Constraint Manager.

The Analyze – Analysis Modes command controls DRC checks. Also, refer to the *DRC State Bar*, located at the bottom of Constraint Manager, adjacent to the *Status Bar*, to

- learn the state of DRC updates
- determine if DRCs are up-to-date for all enabled checks

Figure 1-5 Workbooks and Worksheets



In Figure 1-5, notice how the *Net* object type folder is expanded to show *Timing* as the active workbook. The *Timing* workbook contains the *Switch/Settle Delays* and *Setup/Hold* worksheets. Notice how the worksheets in the *Worksheet Selector* correspond to the worksheet tabs in the active workbook. Also notice that the active workbook and the active worksheet within the active workbook are emphasized with color in the workbook selector.

Note: If any workbook has only one worksheet, Constraint Manager updates the *Worksheet Selector* to contain only the worksheet (under the Object Type folder).

Physical and Spacing Workbook Views

Unlike *Electrical* worksheets, *Physical*, *Spacing,* and *Same Net Spacing* CSet worksheets include layers, which correspond to the cross-section view of your design. Furthermore, you can view these layers collectively (*All Layers*) or individually (*By Layer*). You can also view them at the CSet-level or at the Net-level. See "All Layers / By Layer CSet Views" on page 27. and "All Layers Net View" on page 28.

Working in the *CSet* object folder lets you work in the abstract, defining CSets that will later be applied to net objects. The *All Layers* view shows CSets in collapsed form and layers associated with a CSet in expanded form. The *By Layer* view (in the CSet folder) shows layers in collapsed form and CSets associated with each layer in expanded form.

Note: The CSet view does *not* have a *Referenced CSet* column.

Figure 1-6 All Layers / By Layer CSet Views

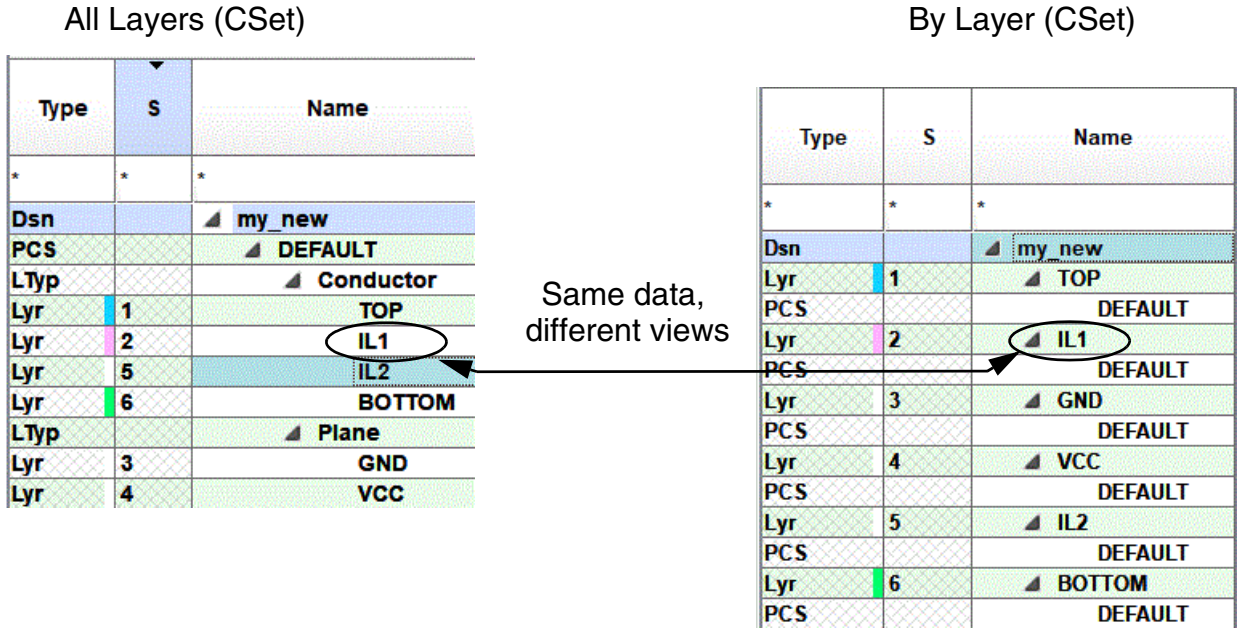


Figure 1-7 All Layers Net View

Working in the *Net* object folder lets you define CSets based on existing CSets or based on constraints already on *Net* objects. The *All Layers* view shows *Container Net* objects in collapsed form and *Net* objects in expanded form.

Note: The *Net*, *Net Class-Class* (in the *Spacing* domain) and *Region* object types have a *Referenced CSet* column but do not have a *By Layer* view

All Layers (Net)

Objects			Referenced Spacing CSet	Line To
Type	S	Name		All
				mil
*	*	*	*	*
Dsn		my_new	DEFAULT	5
NCls		NEW_PHY_CLASS1(2)	DEFAULT	5
NCls		NEW_PHY_CLASS(1)	DEFAULT	5
NCls		NEW_ONE_PHY_CLASS(3)	DEFAULT	5
Bus		S(2)	DEFAULT	5
Net		MUXS0L	DEFAULT	5
Net		HLDA	DEFAULT	5
Bus		RA(8)	DEFAULT	5
Net		RA<7>	DEFAULT	5
Net		RA<6>	DEFAULT	5
Net		RA<5>	DEFAULT	5
Net		RA<4>	DEFAULT	5
Net		RA<3>	DEFAULT	5
Net		RA<2>	DEFAULT	5

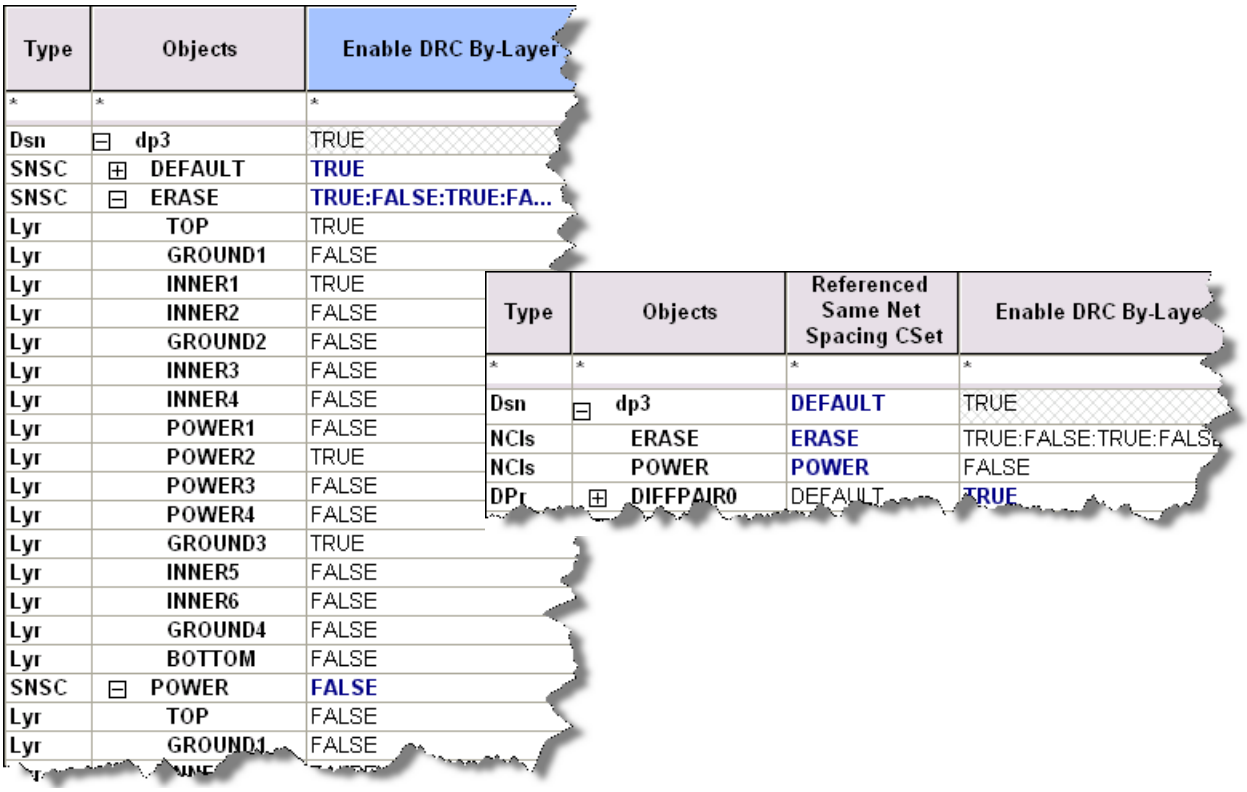
Same Net Spacing DRC Modes

As with other domains, you enable design rule checks for *all* layers through the *Analysis Modes* dialog box. However, in the *Same Net Spacing* domain, you can control design rule checks by layer.

You define a *Same Net Spacing CSet* in the CSet folder, and later assigning that CSet to a constraint object. In this way, you can enable or disable the CSet, effectively providing you with a granular level of control of setting constraint modes by layer. You do this by choosing TRUE or FALSE in the *Enable DRC By-Layer* column in the *Options* worksheet (see Figure 1-8).

Note: A by-layer constraint check is a slave to the mode for that constraint as set in the *Analysis Modes* dialog box (choose Analyze – Analysis Modes). That is, if the individual DRC is not enabled in the *Same Net Spacing Modes* tab, Constraint Manager ignores the state of the *Enable DRC By-Layer* column.

Figure 1-8 Layer-based DRC Modes



Cells

Cells hold data, results, or calculations. Different colors or shades of color are used in cells depending on the state of your design and on the scope of the data in the cell.

Indicates a value inherited from a higher-level object, such as a CSet.

Indicates a Pass state — A value indicating that all child cells of the parent object fall within the set constraint limit.

Indicates a Pass state — A value indicating that all child cells of the parent object fall within the set constraint limit.

Indicates a directly set value in a cell. Also called an override.

Indicates a value that cannot be computed. The reason appears when you hover your mouse over a yellow cell and observe the message displayed in the status line, located at the lower-left corner of Constraint Manager.

You may have not have . . .

- enabled the DRC mode for the cell
- completely placed the object
- completely routed the object
- correctly set up simulation parameters

Note: You can set certain constraints, such as differential pairs, in more than one domain. Constraint Manager indicates a constraint edit in one domain by coloring the cell of the same constraint in the opposite domain yellow.

Indicates a Fail state — A value indicating that the cell violates the set constraint limit. Constraint Manager rolls up worst-case Margins to higher-level objects.

Indicates a Fail state —– A value indicating that any child cell of the parent object violates the set constraint limit.

Indicates a cell which is not applicable to the object. These cells never contain values.

Indicates a cell that you cannot edit.

Indicates a cell containing a formula (the red bar to the right of the cell).

Indicates a cell that is bookmarked.

Tip

To guide you in entering data into a cell, right-click in the cell and choose *Change* from the pop-up menu.

Constraint Manager's User Interface Controls

Constraint Manager employs the same conventional window and worksheet controls that are used in Microsoft Windows Explorer® and Microsoft Excel®. Constraint Manager also supports the Microsoft Intellimouse® and wheel mouse.

Table 1-2 User Interface Controls

Task	Feature	Usage
Command Access	■ Pull-down Menus	Click the pull-down menu at the top of Constraint Manager to access commands.
	■ Icons	Click an icon to execute a command. If you briefly hover the cursor above an icon, a tool tip displays in the status bar (located at the lower-left corner of Constraint Manager) describing the icon's function.
	■ Keyboard Shortcuts	Press **Control** and press: **p** (to print) **z** (to undo) **c** (to cut) **v** (to paste) **f** (to find) **d** (to delete) Also, you can access many commands by pressing **Alt** along with the underlined character, and you can assign your own shortcuts.
	■ Right-Click (context sensitive)	Depending on the object selected, you can right-click to quickly access a command to act on that object.

Task	Feature	Usage
Window and Worksheet Sizing and Placement	■ Drag and Drop	You can drag to reposition the Constraint Manager window, and individual worksheets, on your desktop.
	■ Sizing Borders	You can resize the Constraint Manager window or an individual worksheet open within Constraint Manager by dragging the border.
	■ Maximize/Minimize	You can minimize an open worksheet to an icon or you can maximize it to focus only on that worksheet.
	■ Dismiss	You can click the dismiss [X] button (located at the top right-corner of the worksheet) to close a worksheet. Constraint data is not lost when you dismiss a worksheet.

Task	Feature	Usage
Worksheet Viewing	■ Window Select	You can click and drag on an open worksheet to reposition it.
	■ Object Expand/Collapse	You can use the worksheet selector to work at any object level in the hierarchy (from the system level to the pin pair level) by expanding [+] and collapsing [-] the object tree-structure. You can also choose *Objects – Expand* and *Objects – Collapse* from the pull-down menus to achieve the same effect.
	■ Cascade	You can view all open worksheets arranged one-behind-the-other by cascading (*Window – Cascade*). Constraint Manager orders Worksheets so that each is selectable with a click of the mouse. The active window is placed in the foreground and is identifiable by an active (selected) border.
	■ Tile	You can view all open worksheets simultaneously by tiling (*Window – Tile*). Each open worksheet is automatically sized to accommodate the size of the Constraint Manager window.
	■ New Window	You can duplicate the content of the active worksheet in a new window. This lets you to focus your view on different objects in the same worksheet.
	■ Worksheet Tab Select	When you expand a constraint discipline (signal integrity, timing, routing) from the worksheet selector, all objects within that discipline appear in a worksheet window. You then click a related tab to activate the desired worksheet. You may have to scroll horizontally until the desired worksheet tab is visible.

Task	Feature	Usage
Column Viewing	■ Display Priority ■ ALL Attribute ■ Object Group Type	

Accelerator Keys

Constraint Manager provides function keys and modified function keys that provide quick access to common functions.

Function	Keys
View Options	`Cntrl+F6`
Rename	`F2`
Print	`Cntrl+p`
Analysis Modes	`Cntrl+F9`
Analysis Settings	`Shift+F9`
Analyze	`F9`
Find	`Cntrl+f`
Find Next	`F3`
Find Previous	`Shift F3`
Open a new window on the active worksheet	`Cntrl+n`
Move to the next open worksheet	`Cntrl+Tab`
Move to the previous open worksheet	`Shift+Cntrl+Tab`
Next worksheet tab	`F6`
Previous worksheet tab	`Shift+F6`
Close the active worksheet	`Cntrl+F4`
Select a contiguous range of cells	`Shift+Click`
Select a non-contiguous range of cells	`Cntrl+Click`
Expand object rows	`Alt+ or Num+`

Function	Keys
Collapse object rows	`Alt- or Num-`
Cut	`Cntrl+x or Shift+Del`
Copy	`Cntrl+c or Cntrl+Ins`
Paste	`Cntrl+v or Shift+Ins`
Delete an object or cell content	`Del`
Enable object bookmarks	`Cntrl+F5`
Next object bookmark	`F5`
Previous object bookmark	`Shift+F5`

2

Working with Constraint Objects

Topics in this chapter include

About Constraint Object Hierarchy

This chapter presents information on how to use hierarchical constraint objects in Constraint Manager. See Chapter 3, "Working With Reusable Constraint Objects — CSets" on page 87 for information on reusable constraint objects — Constraint Sets.

Constraint Manager enforces a precedence on objects in your design. Constraints that you specify at the top of the object hierarchy become inherited by the next lower-level object in the hierarchy. Constraints that you define at the lower-levels of the object hierarchy take precedence over (override) the same constraints defined at the next higher-level in the object hierarchy.

This ordering of objects lets you define constraints at highest level possible, only setting overrides on lower-level objects where necessary. Refer to "Constraint Object Hierarchy for Overrides" on page 39 for information on constraint objects and their precedence.

Tip

You should work at the highest level possible in the constraint object hierarchy. For example, you should group individual address signals into a *Bus*. In this way, you only have to constrain the single *Bus* object once rather than having to repeat this process for each signal. Through inheritance, each signal in the *Bus* receives the constraint value assigned at the *Bus* level.

In certain worksheets in the *Electrical* domain, the children of an object reflect the results of an analysis and are not used for the constraint precedence hierarchy. These result-objects are not differentiated from the normal constraint hierarchy but will be maintained for reading. You cannot edit these constraints.

See Chapter 1, Constraint Architecture in the *Allegro Platform Constraints Reference* for detailed information on the Allegro System Constraint Architecture and descriptions of individual constraints.

Table 2-1 Constraint Object Hierarchy for Overrides

Electrical	**Physical**	**Spacing** (net-to-net / same-net)
----------	Design	Design
Net Class	Net Class	Net Class
Net Group	Net Group	Net Group
Bus	Bus	Bus
Differential Pair	Differential Pair	Differential Pair
Match/Relative Group	----------	----------
Xnet	Xnet	Xnet
Net	Net	Net
Pin Pair	Pin Pair	Pin Pair
----------	----------	Net Class-Class *
----------	Region	Region
----------	Region Class	Region Class
----------	----------	Region Class-Class*

OVERRIDE

* Not available in the Same Net Spacing domain

Table 2-2 Constraint Object Hierarchy for Inheritance

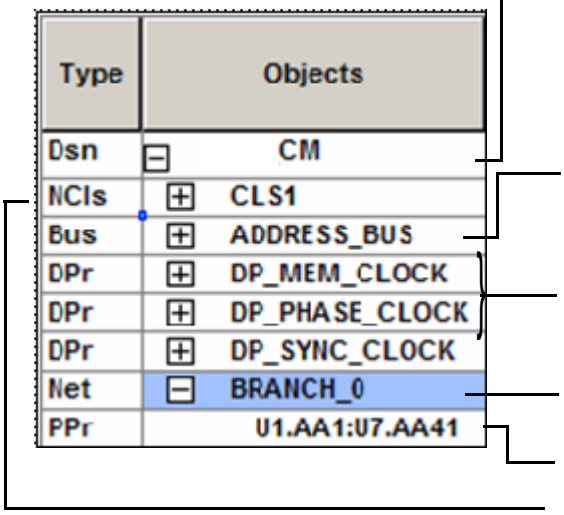
	Electrical	**Physical**	**Spacing** (net-to-net / same-net)
	----------	Design	Design
	Net Class	Net Class	Net Class
	Net Group	Net Group	Net Group
INHERITANCE	Bus	Bus	Bus
	Differential Pair	Differential Pair	Differential Pair
	Match/Relative Group	----------	----------
	Xnet	Xnet	Xnet
	Net	Net	Net
	Pin Pair	Pin Pair	Pin Pair
	----------	----------	----------
	----------	Region	Region
	----------	Region Class	Region Class
	----------	----------	----------

* Not available in the Same Net Spacing domain

Types

The *Type* column indicates the object type for the selected row, as depicted in Figure 2-1 on page 41.

Figure 2-1 Constraint Types



Dsn	Design
DsnI	Design Instance
Lyr	Layer
PrtD	Part Definition
PrtI	Part Instance
GtI	Gate Instance
Bus	Bus
MGrp	Match Group
DPr	Differential Pair
Xnet	Extended Net
Net	Net
PPr	Pin Pair
NCls	Net Class
NCC	Net Class-Class
Rgn	Region
RCls	Region Class
RCC	Region Class-Class
Rslt	Result
PCS	Physical Constraint Set
SCS	Spacing Constraint Set
SNSC	Same Net Spacing Constraint Set
ECS	Electrical Constraint Set
RBnd	Ratsnest Bundle
RPPr	Ratsnest Bundle pin pair member

About Objects

This section describes objects and object groupings in Constraint Manager. See "About Constraint Object Hierarchy" on page 38 for more information. Refer to the *Allegro Constraint Manager Reference* for detailed, step-by-step procedures.

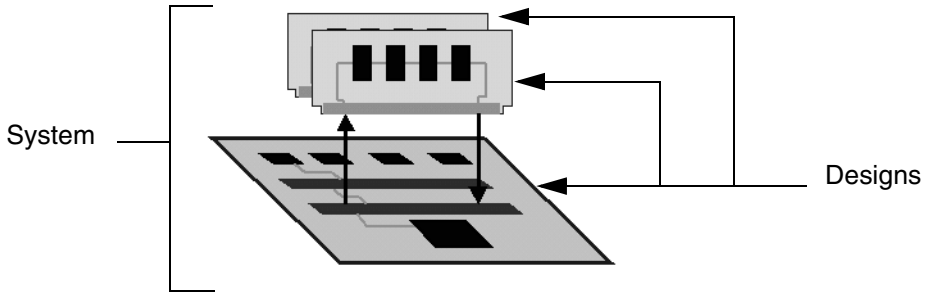
Tip

You can bookmark any design element by selecting it in the *Objects* column, then right-clicking and choosing *Bookmark – Object Bookmark* from the pop-up menu. A square appears to the left of the object to aid you in locating the object. The bookmark follows the object across worksheets. You can also cycle through defined bookmarks, and remove them as well.

Note: Do not use colon character in the object names. It is special character used by Constraint Manager.

Designs and Systems

A *Design* represents a stand-alone board or a board in a *System*, a schematic in Allegro Design Entry HDL, or behavioral logic in Allegro Design Architect. In a multi-board configuration, each board becomes a separate design in the system.



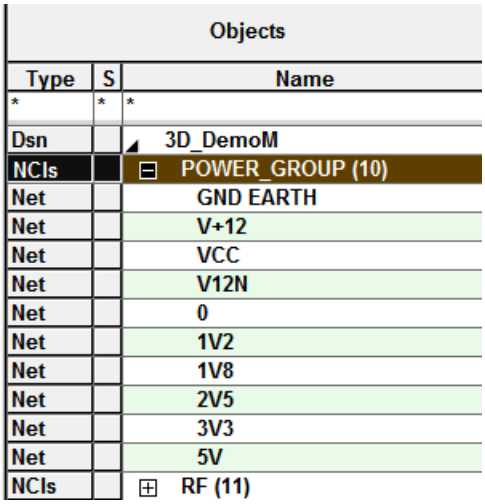
A *System* represents a configuration of designs (boards) including Xnets that traverse these designs and their interconnecting cables and connectors.

Tip

Constraint Manager's Tabbed View aids you in quickly selecting from participating designs in a system.

Net Class

A *Net Class* constraint object lets you group net objects that share common characteristics and require a similar constraint requirement.



Allowable members of a *Net Class* include buses, net groups, differential pairs, Xnets, and nets.

See Constraint Architecture in the *Allegro Platform Constraints Reference* and the Objects – Create – Net Class command.

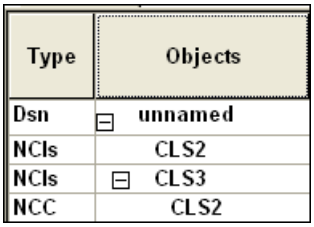
Net Class Rules

The following rules apply to creating a *Net Class*

- You can constrain a *Net Class* with a *CSet*
- You can override individual members of a *Net Class*
- You can constrain a *Net Class* directly (though we recommend using a *CSet*)
- As you create a *Net Class* in the *Physical* domain, you can specify that it also occurs in the in the *Spacing* domain. The converse is true.
- A *Net Class* in the *Electrical* domain must be unique to that domain
- A *Net Class* created in the *Spacing* domain carries over to the *Same Net Spacing* domain; the converse is also true
- A net can be a member of only one *Net Class* per domain

Net Class-Class

A *Net Class-Class is* a constraint object that you define and constrain to represent an *intra*-class spacing relationship among members within a *Net Class* or an *inter*-class spacing relationship among objects in the different *Net Classes*.



Allowable members of a *Net Class-Class* include *Net Classes*.

See Constraint Architecture in the *Allegro Platform Constraints Reference* and the Objects – Create – Net Class-Class command.

Net Class-Class Rules

The following rules apply to creating a *Net Class-Class.* You can

- constrain a *Net Class-Class* with a *CSet*
- override individual members of a *Net Class-Class*
- constrain a *Net Class-Class* directly (though we recommend using a *CSet*)
- create a *Net Class-Class* only in the *Spacing* domain

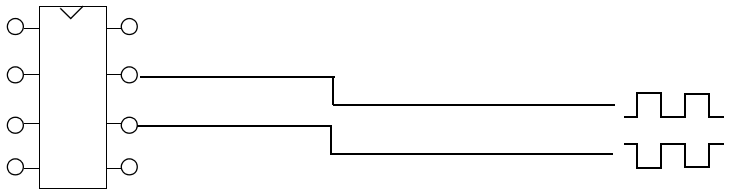
Note: *Net Class-Classes* are not supported in the *Same Net Spacing* domain, or in Design Entry HDL or Allegro System Architect.

If *Net Class-Class* does not have an explicit CSet reference or directly-set values it does not inherit constraints from the parent *Class*.

Differential Pairs

A *differential pair* represents a pair of Xnets or nets that are routed differentially. Differential signaling is a method of sending the same information over two traces.

Differential data transfer employs two traces and (at least) one driver with a positive and negative output and a two terminal receiver. For digital applications, the driver terminals are sending out signals of opposite polarity. While the non-inverted (positive) output transmits a low-to-high transmission, the inverted (negative) output transmits a high-to-low transmission.



To learn more about working with Differential Pair constraints, see

- the Objects – Create – Differential Pairs command in the *Constraint Manager Reference* for more information on the *Differential Pair* constraint object.
- Differential Pair Constraint Data Sheets in the *Allegro Platform Constraints Reference*

Constraint Manager supports two types of differential pairs:

- Model-defined Differential Pairs

  You specify model-defined differential pairs in a device signal model by designating inverting and non-inverting signals of the differential pair. You can uniquely characterize the differential pair by specifying pin parasitics, launch delays, logic thresholds, and buffer delays.

  You assign device signal models to components using the PCB Editor, APD, or SigXplorer. Constraint Manager then recognizes the model-defined differential pair through its view of the board database.

- User-defined differential pairs

  You can create user-defined differential pairs directly in Constraint Manager on a net-level object. This affords you more flexibility in renaming differential pair objects and changing differential pair membership, but you forgo the accuracy of model-defined differential pairs.

Note: Constraint Manager does not support system-level differential pairs.

Differential Pair Worksheets

In the *Electrical* domain, you specify differential pair constraints globally in the *Differential Pair* worksheet of the *Routing* workbook. In the *Physical* domain, you specify differential pair constraints in both the *Net* and *Region* worksheets. If layer variances are required, you can specify *By Layer* differential pair constraints in the *Physical Constraint Set* folder.

Important

The DIFFP_PRIMARY_GAP, MIN_LINE_WIDTH, DIFFP_NECK_GAP and MIN_NECK_WIDTH constraints are allowed on both *Electrical CSets* and *Physical CSets*; however, the *Electrical CSet* value takes precedence over the *Physical CSet* value. Furthermore, in the *Physical* domain, a *Region* takes precedence over an override. See Constraint Objects and Hierarchy in the *Analysis Modes Constraints Reference* for more information.

Table 2-3 Differential Pair Constraints by Domain

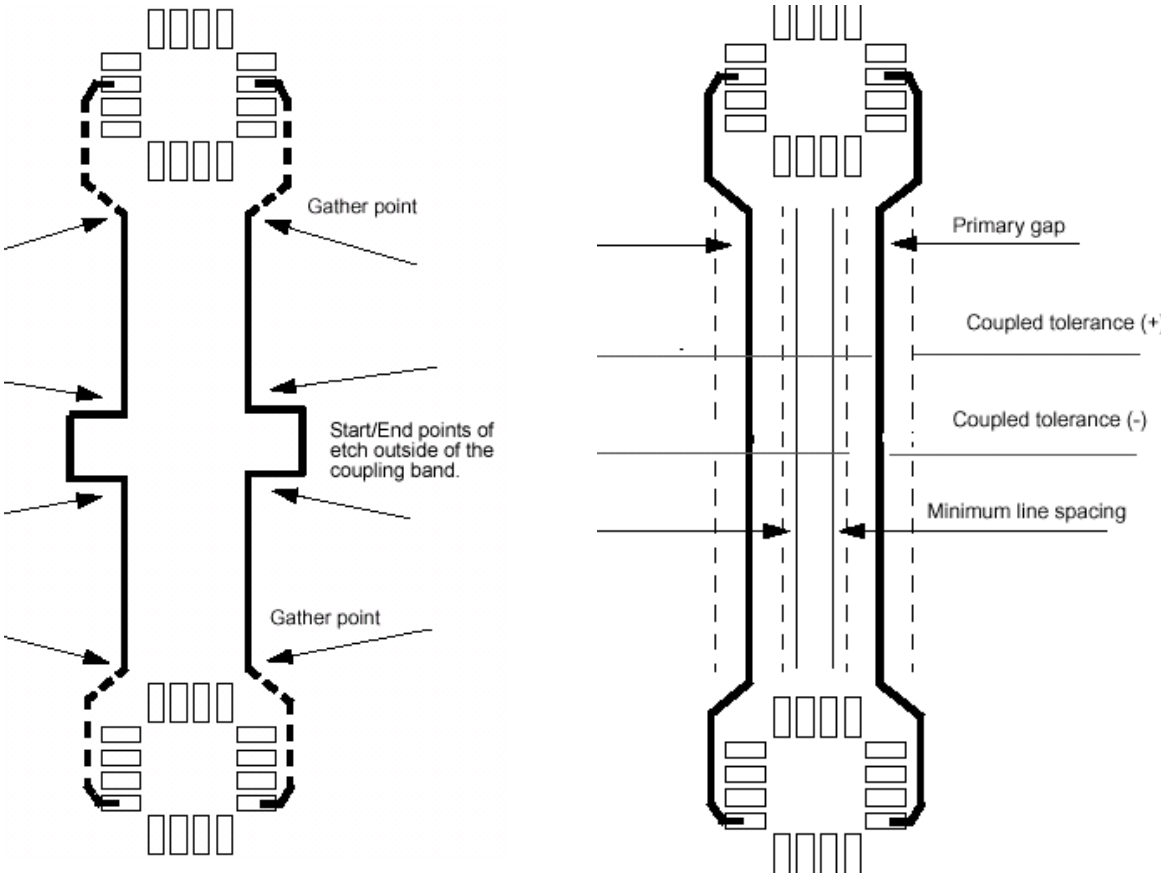
Constraint/ Property	Domain	Column Super Header	Column Label
DIFFP_MIN_SPACE	Physical	Differential Pair	Min Line Spacing
DIFFP_MIN_SPACE	Electrical	Line Spacing	Min
DIFFP_PRIMARY_GAP	Physical	Differential Pair	Primary
DIFFP_PRIMARY_GAP	Electrical	Coupling Parameters	Primary Gap
DIFFP_NECK_GAP	Physical	Differential Pair	Neck
DIFFP_NECK_GAP	Electrical	Coupling Parameters	Neck Gap
DIFFP_COUPLED_PLUS	Physical	Differential Pair	(+) Tolerance
DIFFP_COUPLED_PLUS	Electrical	Coupling Parameters	(+) Tolerance
DIFFP_COUPLED_MINUS	Physical	Differential Pair	(-) Tolerance

Table 2-3 Differential Pair Constraints by Domain

Constraint/ Property	Domain	Column Super Header	Column Label
DIFFP_COUPLED_MINUS	Electrical	Coupling Parameters	(-) Tolerance
DIFFP_UNCOUPLED_LENGTH	Physical	Uncoupled Length	Max
DIFFP_UNCOUPLED_LENGTH	Electrical	Uncoupled Length	Max
MIN_LINE_WIDTH	Physical	Line Width	Min
MIN_LINE_WIDTH	Electrical	Coupling Parameters	Primary Width
MAX_LINE_WIDTH	Physical	Line Width	Max
MIN_NECK_WIDTH	Physical	Neck	Min Width
MIN_NECK_WIDTH	Electrical	Coupling Parameters	Neck Width
MAXIMUM_NECK_LENGTH	Physical	Neck	Max Length
DIFFP_GATHER_CONTROL	Physical	Uncoupled Length	Gather Control
DIFFP_GATHER_CONTROL	Electrical	Uncoupled Length	Gather Control
GATHER_LENGTH_IGNORED	Electrical	Uncoupled Length	Length Ignored
DIFFP_PHASE_TOL	Physical	Phase Tolerance	Tolerance
DIFFP_PHASE_TOL	Electrical	Phase Tolerance	Tolerance

Figure 2-2 shows the boundaries and events that trigger differential pair rule checking and analysis.

Figure 2-2 Differential Pair Gather Points and Coupling Bands



The *Differential Pair* worksheets contain four major constraint categories:

- *Uncoupled Length*

  Uncoupled length constraints limit the amount of coupling between differential pair members. When *gather control* is set to *ignore,* the *actual* uncoupled length is the cumulative etch that lies outside of the coupling band yet lies within the boundaries of the two gather points, or from driver to receiver. A violation results when the uncoupled length exceeds the value you specify in the *max* cell.

  The *length ignored* cell contains the *actual* gather length that is ignored, which is reported on the member net or Xnet and does not bubble up to the differential pair object.

- *Phase Tolerance*

  Phase tolerance constraints ensure that differential pair members are in synchronization and in phase as they switch. You enter a *tolerance* value as a function of time (in nanoseconds) or length (in mils). The *Actual* value reflects the difference in time or length between the members of the differential pair. A violation occurs when the actual value exceeds the tolerance value.

- *Line Spacing*

  The *minimum line spacing* constraint specifies the minimum distance allowed between any two segments of each member of the differential pair. After analysis, the actual cell contains the value containing the smallest gap spacing. A violation occurs when the *Actual* spacing is less than the *min* value.

  Note: The minimum line spacing constraint value that you enter must be less than or equal to the *Primary Gap* minus the *(-) Tolerance*, and it must be equal to or less than the *Neck Gap* minus the *(-) Tolerance*.

- *Coupling*

  Coupling constraints determine the uncoupling events for a routed differential pair. These events determine the *uncoupled length* and *phase deviation*. Use the *differential calculator* to determine values to enter into the *primary gap*, *neck gap*, and *tolerance* cells. See "Using the Differential Calculator" on page 50 for more information.

  - In the *Primary Width* cell, you enter a value for the ideal width of each member of the differential pair.
  - In the *Primary Gap* cell, you enter a value for the ideal edge-to-edge spacing between the pair that should be maintained for the entire length of the pair. In the *(+/-) Tolerance* cells, you enter values to define two bands around the primary gap in which the lines of a pair can go beyond or closer than the primary gap value. When the lines of etch are within these bands, they are considered coupled.
  - In the *Neck Width* cell, you enter a value for the minimum allowable width for a line in a differential pair as it goes through confined areas among densely placed components.
  - In the *Neck Gap* cell, you enter a value for the edge-to-edge spacing between a pair as it goes through tight areas full of component pins and vias. The smallest allowable gap consists of the *Neck Gap* minus the *(-) Tolerance*.

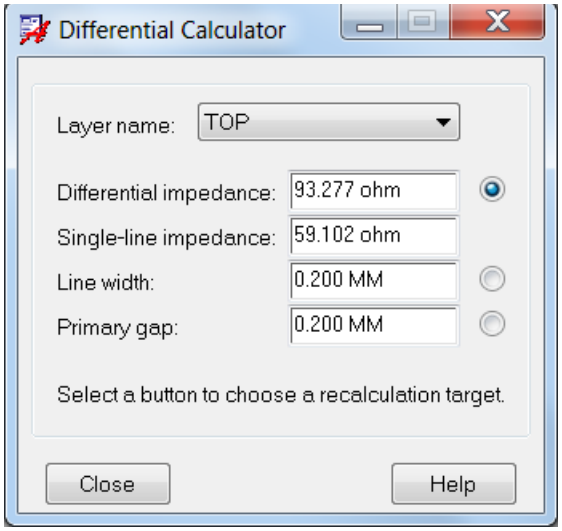
Tip

Neck Gap overrides any value in the *Primary Gap* when the differential pair's spacing collapses to or below the value of the *Min neck width* rule in an Electrical CSet assigned to the nets in the differential pair object. Therefore:

- Ensure that the neck gap does not go below any *Min* line spacing value you have set.
- You do not need to define a neck gap if you set *(-) Tolerance* with a value that accounts for the needed neck gap.

Using the Differential Calculator

Use the *Differential Calculator* to perform what-if scenarios to determine what combinations of line width and primary gap values can help you obtain a particular differential impedance. The calculator is available from the Electrical CSet- and Net worksheets.



To access the differential calculator, in the Primary Gap, Neck Gap, or +/- Tolerance cells, right-click and choose *Change* from the pop-up menu. Then click *Calculator*.

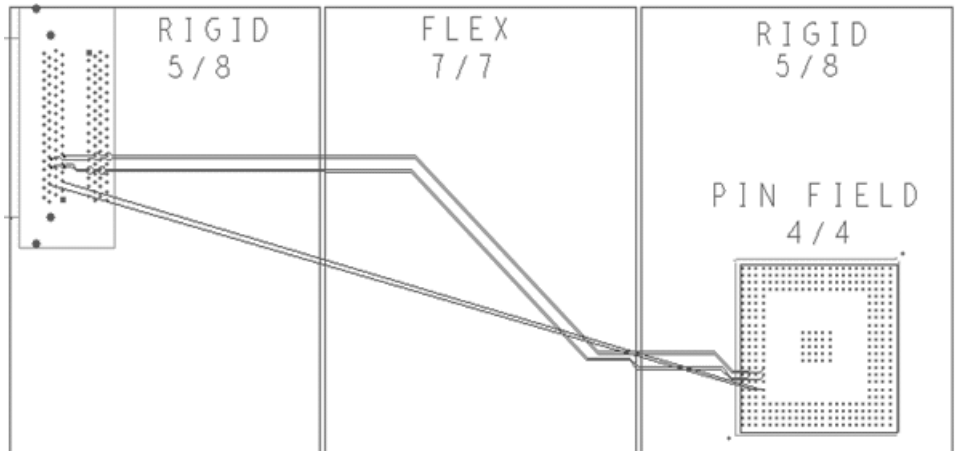
You can perform calculations only for edge-coupled differential pairs on a selected ETCH/CONDUCTOR layer and account for material, thickness, electrical conductivity, dielectric constant, loss tangent, and shield.

Layer name	Indicates the ETCH/CONDUCTOR layer for which you are running the calculation. By default, the TOP/SURFACE layer appears.
Differential impedance	Specifies the impedance of the differential pair. Calculated for a pair of lines having the specified *Line width* and *Primary gap* on the layer.
Single-line impedance	Indicates the impedance of one line of etch on the selected layer. Changing this value automatically recalculates the *Line width* field. Each time the *Line width* field in this calculator changes—either when you directly modify it or when you select it to be recalculated—this value is recalculated, too.
Line width	Specifies the minimum width of each line of the differential pair. Changing this value automatically recalculates the *Single-line impedance* field. And changing the *Single-line impedance* field automatically recalculates this value.
Primary gap	Indicates the ideal edge-to-edge spacing between the pair that should be maintained for the entire length of the pair.

Differential Pairs by Constraint Region

A signal's impedance is affected by the dielectric constant of the material through which it passes. To maintain a constant impedance, line width and gap dimensions will differ for internal and outer layers. Because the electrical constraints apply globally, you must constrain the differential pairs in the *Physical* domain, which can vary by layer.

Rigid-flex designs require a change in material. Therefore, line widths and gaps may need to be narrower or wider, depending on the two materials. Because the physical editors allow only one material per layer, the only way to specify different constraints due to a change in material, is to use a constraint region to re-define a differential pair's line width and gap.

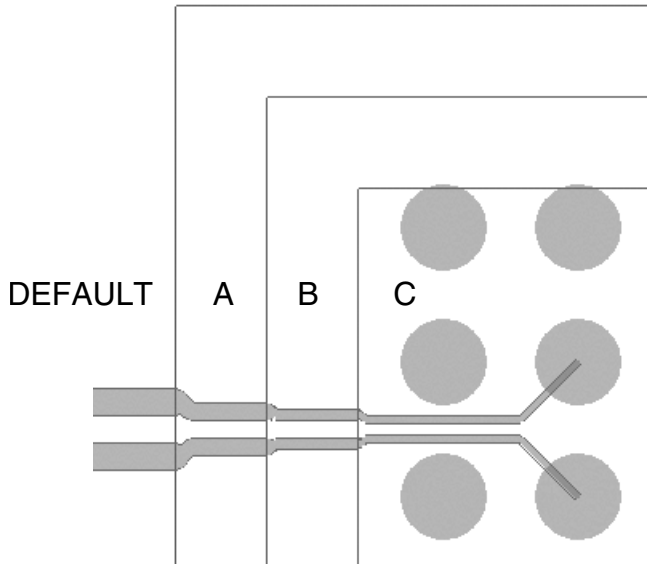


Note: To present the most flexible design options, additional differential pair physical constraints (Min Line Space and Tolerance +/-) appear in both domains (see Differential Pair Constraints by Domain on page 46). Although the *Electrical* domain lacks by-layer variance support, it does take precedence over a constraint of the same name in the *Physical* domain.

In Figure 2-3 on page 53, differential pair line and gap constraints deviate from the values specified in the DEFAULT PCSet as the signal traverses the different materials and tight spacing confines upon entering a BGA.

These are represented by unique Region rules: A, B, and C.

Figure 2-3 Differential Pair Constraints by Region



To accommodate these differing line width and gap requirements, on the same or different layers, a good strategy is to:

1. In the *Physical* domain, create three *Region* constraint objects (in the *All Layers* worksheet in the *Regions* folder) for each region: A, B, and C.

   Note: If the dimensions of the region's geometrical boundaries have not been defined, you must specify and name them in a physical editor before you can create a *Region* constraint object.

   For information on how to create a *Region*, see the Objects – Create – Region command.

2. Create and define *PCSets* to reflect the new line width and gap constraints, as well as other unique differential pair parameters.

   Note: Alternatively, use directly-set constraint values (overrides) instead of new PCSets.

   For information on how to create a *PCSet*, see the Objects – Create – Physical CSet command.

3. In the *Physical* domain, associate each *Region* constraint object with the respective *PCSet* that contains the unique rules for that region.

   For information on how to associate a *PCSet* with a *Differential Pair* constraint object, see the Objects – Constraint Set References command.

Differential Pair Rules

The following rules apply to differential pairs.

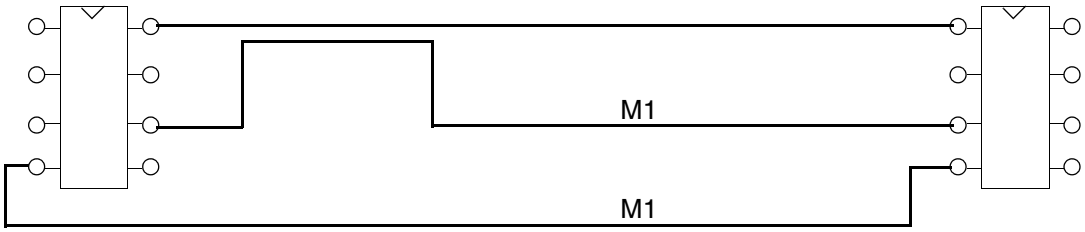
Model-defined Differential Pair	User-defined Differential Pair
■ You create model-defined differential pairs in the PCB Editor or APD using the *Analyze – SI/EMIsim – Model* command.	■ You create user-defined differential pairs in Constraint Manager using the *Objects – Create – Differential Pair* command or in the PCB Editor or APD using the *Logic – Assign Differential Pair* command.
■ Model-defined differential pairs are preferred in the high-speed flow because they uniquely characterize the differential pair members including pin parasitics, launch delays, logic thresholds, and buffer delays.	■ User-defined differential pairs are not as accurate as model-defined differential pairs because they use default IBIS device values.
■ A member in a model-defined differential pair cannot be a member of another differential pair object.	■ A member in a user-defined differential pair cannot be a member of another differential pair object.
■ You can extract both members of a model-defined differential pair object into SigXplorer with full coupling effects intact.	■ You can extract only one member of a user-defined differential pair object into SigXplorer.
■ Model-defined differential pairs take precedence. If you create a user-defined differential pair and later use the same members in a model-defined differential pair, the model-defined differential pair takes precedence.	■ User-defined differential pairs lend precedence. If you create a user-defined differential pair and later use the same members in a model-defined differential pair, the model-defined differential pair takes precedence.

Model-defined Differential Pair	User-defined Differential Pair
■ You cannot rename a model-defined differential pair object in Constraint Manager.	■ You can rename a user-defined differential pair object in Constraint Manager.
■ You cannot change the membership of a model-defined differential pair in Constraint Manager. You must use the PCB Editor, APD, or SigXplorer to do this by editing the device model.	■ You can change the membership of a user-defined differential pair in Constraint Manager.
■ You cannot create a model-defined differential pair in the design entry editor when used in the Design Entry HDL – Constraint Manager high-speed flow.	■ You can create a user-defined differential pair in the design entry editor when used in the Design Entry HDL – Constraint Manager high-speed flow. Analysis and DRC checking is not supported in this mode.
■ With a model-defined differential pair, any device that references the same device type will inherit the differential pair specified in that model. This reuse is analogous to creating an Electrical CSet and assigning it to many design objects in Constraint Manager.	■ You must create user-defined differential pairs individually in Constraint Manager. Although auto-setup simplifies the process, it is not the same as inheritance with model-defined differential pairs.

Match Groups

A *Match Group* is a collection of nets, Xnets, or pin pairs which must all match (in *delay* or *length*) or be relative to a specific *target* within the group.

In this illustration two of the three nets belong to a *Match Group*—M1. To maintain a match propagation delay, you must extend the middle net to match the delay of the bottom net.



See Constraint Architecture in the *Allegro Platform Constraints Reference* and the Objects – Create – Match Group command.

You can create *Match Groups* manually by adding properties directly to Xnets, or by using Constraint Manager to explicitly define pin pairs in an Electrical CSet. Regardless of the method that you choose, *Match Groups* are resolved down to specific pin pairs and are further defined by the following attributes: *Scope*, *Delta*, *Tolerance* and *Target*.

Generating pin pairs for the Match Group members

Match Groups can be populated directly with pin pairs manually or through explicit pin pairs defined in an Electrical CSet. If the *Match Group* was instead created by adding nets or Xnets directly, an additional attribute is needed to guide the resolution of pin pairs that will ultimately be compared in the *Match Group*. In the *Pin Pairs* column of the *Relative Propagation Delay* worksheet, you can guide the generation of pin pairs with the following options:

Choosing this option . . .	Generates pin pairs based on . . .
■ *All Drivers/All Receivers*	All combinations of drivers and receivers.
■ *Longest Driver/ Receiver*	The longest driver-receiver pin pair. Uses the longest pin pair when there are not any drivers or receivers.
■ *Longest Pin Pair*	The longest pin pair.

When you analyze the *Match Group*, Constraint Manager automatically generates pin pairs based on the pinuse of the pins in the net or Xnet and the manhattan distances between the pins. Pin pairs appear hierarchically under the net or Xnet within the *Match Group*. You can sort the pin pairs in the *Match Group.* The Constraint Manager sorts the pin pairs in the *Match Group* on the basis of net names that belong to the pin pairs*.*

Tip

You create a *Match Group* in the *Objects* column of the *Relative Propagation Delay* worksheet by adding member nets and Xnets. If you cannot get the pin pairs that you want from the available settings, you should remove the net or Xnet from the *Match Group*, create the pin pairs that you want, and then add those pin pairs back to the *Match Group*.

Defining Match Group requirements

The two main requirements for a *Match Group* are *Delta* and *Tolerance*, which occupy the *Delta:Tolerance* column in the *Relative Propagation Delay* worksheet. Based on these settings, a Match Group is further classified as *Relative* or *Match*.

Match Delay

You specify only a *Tolerance* value; all pin pairs are compared against each other and must be within the specified *Tolerance*.

Relative Delay

You specify both *Delta* and *Tolerance* values, and select a *Target*. The *Target* may be implicit or explicit; each pin pair is compared to the *Target* pin pair by the specified *Delta* and within the specified *Tolerance*.

The following attributes characterize the requirements for a Match Group:

- *Tolerance* — *Tolerance* is the allowable skew when matching member pin pairs. You specify *Tolerance* as either *length*, *delay,* or a *percentage*.

  If you define only a *Tolerance* value for a member pin pair or for the *Match Group*, the member is compared to every other pin pair within the specified *Tolerance*.

  If you define a *Delta* value for a member, the member is matched to the *Target,* plus or minus the *Delta* and within the specified *Tolerance*.

- *Delta* — *Delta* is the value added to, or subtracted from, the routed length of the *Target. Delta* determines the required length of the pin pair before applying the *Tolerance*.

  The *Delta* may be negative, in which case the value is subtracted from the routed length of the *Target*, or positive (or unsigned), in which case the value is added to the routed length of the *Target*. You can specify *Delta* as either *length* or *delay*.

  If the *Delta* is unspecified, the pin pair must match all other pin pairs in the *Match Group* (within the *Tolerance*). A design rule violation results when the difference is greater than the *Tolerance*.

  Note: If you specify zero for a *Delta* (which is different from leaving the field blank), the value is compared directly to the routed *length* or *delay* of the *Target*.

- *Target*

  Target is a pin pair that is referenced by all pin pairs in the *Match Group*. See "Determining the target pin pair" on page 61 for more information.

  Target applies only when a *Delta* value exists; only one *Target* is specified for the entire *Match Group*.

  When you explicitly set the *Target* pin pair (using the right mouse button), it remains the *Target*. When the *Target* is not user-defined, it may change from one net to another based on changes to the original *Target*, such as a change in its manhattan length.

  If a pin pair is the *Target* in one *Match Group*, it does *not* have to be the *Target* in another *Match Group*, if that pin pair is a member of multiple *Match Groups*.

Examples

The following three scenarios cover *Match Groups* (*match* and *relative delay*) under varied conditions.

Tip

You can define a generic *Match Group* in an *Electrical CSet*. You can subsequently use the generic Match Group to define the net- or Xnet-specific groups when a net references the *Electrical CSet*.

Match Group (Scenario 1)

Pin Pair	Target	Delta	Tolerance	Comments
PP1	N/A	Unspecified	10 mils	In this scenario, you do not require a *Target* pin pair because the *Delta* is unspecified. Constraint Manager compares the routed length of each pin pair to every other pin pairs' routed length. If all pin pair differences are less than 10 mils, then the *Match Group* is within constraint limits. When you do not specify a *Delta* value, it is considered a match delay, not a relative match delay. **Note:** You can verify *Unrouted Length*. Choose *Analyze – Modes* and click the *Options* tab. Constraint Manager reports the worst-case result for each pin pair.
PP2	N/A	Unspecified	10 mils	
PP3	N/A	Unspecified	10 mils	

Match Group (Scenario 2)

Pin Pair	Target	Delta	Tolerance	Comments
PP1		0 mils	10 mils	In this scenario, PP3 is the reference for the *Match Group*. You must route all pin pairs zero mils longer than PP3 with a *Tolerance* of 10 mils. If PP3 is 1000 mils, you must route PP1 and PP2 between 990 mils and 1010 mils. When you specify a *Delta* value, the Match Group is a *relative* match delay and each pin pair is compared to a single *Target* pin pair.
PP2		0 mils	10 mils	
PP3	X	0 mils	10 mils	

Match Group (Scenario 3)

Pin Pair	Target	Delta	Tolerance	Comments
PP1	X	100 mils	10 mils	In this scenario, PP1 is the reference for the *Match Group*. The routed length of all other pin pairs, plus (or minus) the *Delta*, must be within 10 mils of PP1. If PP1 is 1000 mils, PP2 must be no less than 1090 mils and no greater than 1110 mils; also, PP3 must be no less than 890 mils and no greater than 910 mils.
PP2		100 mils	10 mils	
PP3		- 100 mils	10 mils	

Determining the target pin pair

In a *Match Group*, Constraint Manager selects one of the pin pairs (or you specify one) as the *Target* and all of the other pin pairs are matched against this *Target* within the given *Delta* and *Tolerance*. When you manually specify a *Target*, there are no *Delta* and *Tolerance* values for that row.

If the TARGET keyword is displayed in the Delta/Tolerance cell for more than one member of a Match Group, the constraint information is invalid. You should *clear* or enter an explicit delta/tolerance value for each member that should not be considered the *Target*.

Constraint Manager determines the *Target* pin pair as follows. The pin pair . . .

1. that you explicitly set (using the right mouse button in the *Delta:Tolerance* column).
2. with the smallest absolute *Delta* value (if all pin pairs have a *Delta* value).
3. with the longest manhattan length (if more than one pin pair has the same—smallest—*Delta* value).

Note: If none of the pin pairs has a *Delta* value, no *Target* is chosen; therefore, all pin pairs are compared to each other and the group is considered match delay, not relative match delay.

For example, if one pin pair has a *Delta* value of -300 and two pin pairs have a *Delta* value of zero, the pin pair with a zero *Delta* and the longest manhattan distance is chosen as the default *Target*. Although zero is larger than -300, the absolute value of -300 is 300, which is larger than zero. The two pin pairs with the zero value are compared by manhattan distance, and the larger is selected as the *Target*.

Note: When the *Delta* for all pin pairs is set to Null, the delay is considered match delay, not relative match delay.

Scope

Scope controls the validation of the Match Group (*Local* and *Global*), as well as the generation of Match Groups (*Bus* and *Class*). Once you define the members of a *Match Group*, you can specify the scope in the . . .

- *Scope* column of the *Relative Propagation Delay* worksheet (in the *Net* folder).
- *Scope* column of the *Relative Propagation Delay* worksheet (in the *Electrical Cset* folder).
- *Rel Prop Delay* tab in SigXplorer (choose *Set – Constraints*).

You can specify the following scope options:

- *Local* — Validates only pin pairs within each net (or Xnet) against other pin pairs in the same net (or Xnet) for each member of the *Match Group*.
- *Global* — Validates all pin pairs against all other pin pairs in the *Match Group*.

- *Bus*

  Bus scope is useful in situations where groups of signals are replicated and constraining them would require multiple *Electrical CSets* that only differ by *Match Group* name. If the nets that reference the *Electrical CSet* are not grouped into buses, a single *Match Group* with a *Global* scope is created.

  Note: *Bus* scope can only be set in an *Electrical CSet* and only applies during the *Electrical CSet* mapping process.

  If you subsequently apply the *Electrical CSet* to . . .

  - a bus member, Constraint Manager maps the *Match Group* constraints to the bus member.
  - the parent *Bus* object, Constraint Manager maps the *Match Group* constraints to all members of the *Bus*.
  - a non-*Bus* member, Constraint Manager applies a *Global* scope and retains the original name of the *Match Group*.

  When an *Electrical CSet* is applied to nets, all generated pin pairs are added to unique *Match Groups* based upon their *Bus* membership. The unique *Match Group* name is based upon the *Electrical CSet Match Group* name suffixed with the *Bus* name.

  The generated *Match Group* is created with a scope of *Global* so that they are validated appropriately, as defined above in the *Global* setting. The *Bus* scope setting allows you to reference the same *Electrical CSet* from multiple *Buses*, resulting in unique *Match Groups* for the members/pin-pairs of each Bus (as required). This reduces the number of ECSets needed to constrain a design.

  Example:

  An *Electrical CSet* defines a *Match Group* (*mymatchgroup),* with a *Bus* scope*,* which is referenced to two buses: BusA and BusB.

  After net-level *Electrical CSet* mapping, Constraint Manager creates the *Match Groups* with a *Global* scope

  ```
  mymatchgroup_BusA

  mymatchgroup_BusB
  ```

- *Class*

  Class scope is useful in situations where groups of signals are replicated and constraining them would require multiple *Electrical CSets* that only differ by *Match Group* name. If the nets that reference the *Electrical CSet* are not grouped into *Classes*, a single *Match Group* with a *Global* scope is created.

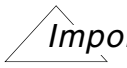
  Note: You can set *Class* scope only in an *Electrical CSet* and it only applies during the *Electrical CSet* mapping process.

  When an *Electrical CSet* is applied to nets, all generated pin pairs are added to unique *Match Groups* based upon their *Class* membership. The unique *Match Group* name is based upon the *Electrical CSet Match Group* name suffixed with the *Class* name.

  The generated *Match Group* is created with a scope of *Global* so that they are validated appropriately, as defined above in the *Global* setting. The *Class* scope setting allows you to reference the same *Electrical CSet* from multiple *Classes*, resulting in unique *Match Groups* for the members/pin-pairs of each Class (as required). This reduces the number of ECSets needed to constrain a design.

  Class scope is more flexible than *Bus* scope because a *Class* can include more disparate signals than a *Bus* can. A *Bus* is typically limited to vectored nets or nets that share a common topology; a *Class* often includes control signals, too.

  You can also use *Class* and *Bus* scope to ensure that unique *Match Groups* are created in the top level of a hierarchical design in Allegro Design Entry HDL or Allegro System Architect, where replicated blocks exist.

  The default configuration for Electrical *Classes* is *Local*, which allows for electrical constraints and *Match Groups* to be created in a hierarchical block, and remain specific to that block at a higher level, and also in PCB Editor.

- *Class* (Continued)

  Example:

  An *Electrical CSet* defines the following *Match Group* (mymatchgroup) with a *Class Scope*, which is referenced to two *Net Classes*: ClassA and ClassB.

  After net-level *Electrical CSet* mapping, Constraint Manager creates the *Match Groups* with a *Global* scope

  `mymatchgroup_ClassA`

  `mymatchgroup_ClassB`

- *Net Group*

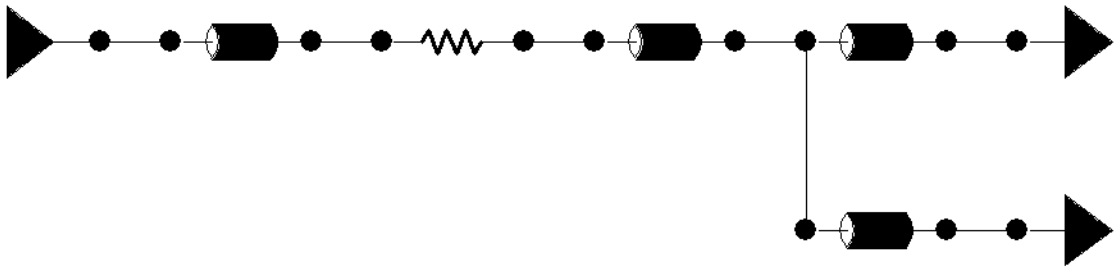
  Net Group scope is useful in situations where groups of different net objects are replicated and constraining them would require multiple *Electrical CSets* that only differ by *Match Group* name.

Scope Scenarios

The following scenarios illustrate various scope settings and how they are used to create different *Match Groups*, depending on the desired constraints.

Assume you have two buses, each with three Xnets, and you want to match the two signal paths in each Xnet by matching the driver to the top receiver and the driver to the bottom receiver (two pin pairs per Xnet, six pin pairs in each bus). Depending on the constraint requirements, you can create several different *Match Groups*, each with a relative propagation delay constraint. In this scenario, we will be creating the constraints by applying an *Electrical CSet* with an associated topology.

Figure 2-4 Match Group Topology



Scenario 1: Local Scope

Each driver to receiver path must match within each Xnet, and can match independent of the paths in any other Xnet.

You want to use a *Local* scope so that the topology in Figure 2-4 on page 65 results in a single *Match Group* with eight pin pairs, two per Xnet.

Note: With a *Local* scope, Constraint Manager validates pin pairs independently in each Xnet.

When you apply the *Electrical CSet* to a *Bus*, the *Match Group* spans both buses, adding members of each bus to the *Match Group* (as shown in Figure 2-5).

Figure 2-5 Net-level Match Group with Local Scope

Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Scope	Delta:Toler
			Pin 1	Pin 2		
			mil	mil		ns
System						
my_db						
MG1						
U17.7:U10.12 [ADDR5]					Local	:0.5 ns
U17.7:U11.12 [ADDR5]					Local	:0.5 ns
U17.9:U10.11 [ADDR4]					Local	:0.5 ns
U17.9:U11.11 [ADDR4]					Local	:0.5 ns
U18.7:U10.9 [ADDR3]					Local	:0.5 ns
U18.7:U11.9 [ADDR3]					Local	:0.5 ns
U18.9:U10.8 [ADDR2]					Local	:0.5 ns
U18.9:U11.8 [ADDR2]					Local	:0.5 ns
U19.7:U10.7 [ADDR1]					Local	:0.5 ns
U19.7:U11.7 [ADDR1]					Local	:0.5 ns
U19.9:U10.6 [ADDR0]					Local	:0.5 ns
U19.9:U11.6 [ADDR0]					Local	:0.5 ns
BUS1	EC1					
ADDR0	EC1					
ADDR1	EC1					
ADDR2	EC1					
BUS2	EC1					
ADDR3	EC1					
ADDR4	EC1					
ADDR5	EC1					
W_R						

Note: With a *Local* scope, all pin pairs occupy a single *Match Group*; Constraint Manager compares pin pairs to only pin pairs in the same Xnet. In Figure 2-5, Constraint Manager compares pin pair `U17.7:U10.12` only with pin pair `U17.7:U11.12` because they come from the same Xnet, `ADDR5`.

Scenario 2: Global Scope

Each driver to receiver path must match every other driver to receiver path, regardless of the path of the Xnet or *Bus*. This scenario is common to the majority of *Match Groups*, especially where it is only one pin pair in a particular Xnet.

Use a *Global* scope so the topology results in a single *Match Group* with eight pin pairs, two per Xnet, as shown in Figure 2-4 on page 65.

Figure 2-6 Net-level Match Group with Global Scope

Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Scope	Delta:Tol
			Pin 1	Pin 2		
			mil	mil		ns
System						
my_db						
MG1						
U17.7:U10.12 [ADDR5]					Global	:0.5 ns
U17.7:U11.12 [ADDR5]					Global	:0.5 ns
U17.9:U10.11 [ADDR4]					Global	:0.5 ns
U17.9:U11.11 [ADDR4]					Global	:0.5 ns
U18.7:U10.9 [ADDR3]					Global	:0.5 ns
U18.7:U11.9 [ADDR3]					Global	:0.5 ns
U18.9:U10.8 [ADDR2]					Global	:0.5 ns
U18.9:U11.8 [ADDR2]					Global	:0.5 ns
U19.7:U10.7 [ADDR1]					Global	:0.5 ns
U19.7:U11.7 [ADDR1]					Global	:0.5 ns
U19.9:U10.6 [ADDR0]					Global	:0.5 ns
U19.9:U11.6 [ADDR0]					Global	:0.5 ns
BUS1	EC1					
ADDR0	EC1					
ADDR1	EC1					
ADDR2	EC1					
BUS2	EC1					
ADDR3	EC1					
ADDR4	EC1					
ADDR5	EC1					
W_R						

Note: With a *Global* scope, Constraint Manager validates pin pairs collectively in the (single) *Match Group*, as shown in Figure 2-6. In this scenario, Constraint Manager compares pin pair `U17.7:U10.12` to all other pin pairs in the *Match Group*.

Scenario 3: Bus Scope

Each driver to receiver path must match every other driver to receiver path, but only within the same bus. In this example, we have the Xnets grouped into two buses; however, your interface may have the same bus structure replicated many times.

One option is to create a separate *Electrical CSet* for each bus using a *Global* scope; the resulting *Electrical CSets* would be identical except for the name of the *Match Group*. With a *Bus* scope, the topology in Figure 2-4 on page 65 results in two *Match Groups* (derived from a common *Match Group*), with six pin pairs each.

Note: As with a *Global* scope, Constraint Manager validates pin pairs collectively between both *Buses* when the **Electrical CSet** is applied to either *Bus*.

You must specify a *Bus* scope, for a *Match Group*, within an *Electrical CSet* (as shown in Figure 2-7) in either the *Electrical CSet* folder or in SigXplorer. You can then apply the *Electrical CSet* to a *Bus* member in the *Relative Propagation Delay* worksheet. Although you defined a *Bus* scope at the *Electrical CSet* level, when the *Electrical CSet* is applied to a *Bus* member at the net level, the *Scope* column indicates *Global*.

Figure 2-7 Electrical CSet-level Match Group with Bus Scope

Objects	Pin Pairs	Scope	Delta:Tolerance
			ns
⊟ System			
⊟ my_db			
⊟ EC1			
⊟ MG1			
U19.9:U10.6		Bus	:0.5 ns
U19.9:U11.6		Bus	:0.5 ns

When you apply the *Electrical CSet* to a *Bus*, each *Bus* inherits constraints from the same *Match Group*. In this way, the same *Match Group* can span many *Buses*, yet you can modify the Match Group on a per bus basis. Constraint Manager prepends the common *Match Group* name to the independent *Bus* name (as shown in Figure 2-8 on page 69), in effect, uniquely qualifying the *Match Group*. In this way, the same *Match Group* (and the same *Electrical CSet*) is common to many buses, yet you can modify each independently.

Figure 2-8 Net-level Match Group with Bus Scope

Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Scope	Delta:Tole
			Pin 1	Pin 2		
			mil	mil		ns
System						
my_db						
MG1_BUS1						
U18.9:U10.8 [ADDR2]					Global	:0.5 ns
U18.9:U11.8 [ADDR2]					Global	:0.5 ns
U19.7:U10.7 [ADDR1]					Global	:0.5 ns
U19.7:U11.7 [ADDR1]					Global	:0.5 ns
U19.9:U10.6 [ADDR0]					Global	:0.5 ns
U19.9:U11.6 [ADDR0]					Global	:0.5 ns
MG1_BUS2						
U17.7:U10.12 [ADDR5]					Global	:0.5 ns
U17.7:U11.12 [ADDR5]					Global	:0.5 ns
U17.9:U10.11 [ADDR4]					Global	:0.5 ns
U17.9:U11.11 [ADDR4]					Global	:0.5 ns
U18.7:U10.9 [ADDR3]					Global	:0.5 ns
U18.7:U11.9 [ADDR3]					Global	:0.5 ns
BUS1	EC1					
ADDR0	EC1					
ADDR1	EC1					
ADDR2	EC1					
BUS2	EC1					
ADDR3	EC1					
ADDR4	EC1					
ADDR5	EC1					
W_R						

Tip

You can also influence the mapping process by specifying *optional pins* in the topology. See Optional Pins in the *SigXplorer Command Reference* for more information.

Scenario 4: Class Scope

In this scenario, an *Electrical CSet* is defined in the *Electrical Constraint Set* folder and it contains a *Match Group* with a *Scope* of *Class*.

Type	Objects	Pin Pairs	Scope	Delta:Tolerance
				mil
Dsn	⊟ class_scope2			
ECS	⊟ DDR2_DQ0			
ECSM	MATCH_POINT	Longest Pin Pair	Class	0 ns:5 %

ECSet
Match Group

In the *Relative Propagation Delay* worksheet of the *Routing* workbook in the *Net* folder, the `DDR2_DQ` bus contains 16 data signals and is assigned the `DDR2_DQ0` *Electrical CSet*.

Type	Objects	Referenced Electrical CSet
Bus	⊟ DDR2_DQ	DDR2_DQ0
Net	DDR2_DQ0	DDR2_DQ0
Net	DDR2_DQ1	DDR2_DQ0
Net	DDR2_DQ2	DDR2_DQ0
Net	DDR2_DQ3	DDR2_DQ0
Net	DDR2_DQ4	DDR2_DQ0
Net	DDR2_DQ5	DDR2_DQ0
Net	DDR2_DQ6	DDR2_DQ0
Net	DDR2_DQ7	DDR2_DQ0
Net	DDR2_DQ8	DDR2_DQ0
Net	DDR2_DQ9	DDR2_DQ0
Net	DDR2_DQ10	DDR2_DQ0
Net	DDR2_DQ11	DDR2_DQ0
Net	DDR2_DQ12	DDR2_DQ0
Net	DDR2_DQ13	DDR2_DQ0
Net	DDR2_DQ14	DDR2_DQ0
Net	DDR2_DQ15	DDR2_DQ0

Next, the *Bus* is divided equally into separate *Net Classes* (`DDR2_0` and `DDR2_1`). Control signals are also added to each *Net Class*.

Type	Objects	Referenced Electrical CSet
NCls	⊟ DDRQ_LANE0	
Net	DDR2_DM0	CONTROL_SIGNAL
Net	DDR2_DQS0	CONTROL_SIGNAL
Net	DDR2_DQ0	DDR2_DQ0
Net	DDR2_DQ1	DDR2_DQ0
Net	DDR2_DQ2	DDR2_DQ0
Net	DDR2_DQ3	DDR2_DQ0
Net	DDR2_DQ4	DDR2_DQ0
Net	DDR2_DQ5	DDR2_DQ0
Net	DDR2_DQ6	DDR2_DQ0
Net	DDR2_DQ7	DDR2_DQ0
NCls	⊟ DDR2_LANE1	
Net	DDR2_DM1	CONTROL_SIGNAL
Net	DDR2_DQS1	DDR2_DQ0
Net	DDR2_DQ8	DDR2_DQ0
Net	DDR2_DQ9	DDR2_DQ0
Net	DDR2_DQ10	DDR2_DQ0
Net	DDR2_DQ11	DDR2_DQ0
Net	DDR2_DQ12	DDR2_DQ0
Net	DDR2_DQ13	DDR2_DQ0
Net	DDR2_DQ14	DDR2_DQ0
Net	DDR2_DQ15	DDR2_DQ0

Next, you can see the *Match Groups*. When merged to the top-level of a hierarchical block, the *Match Groups* are qualified with unique names, consisting of the concatenation of the *Electrical CSet Match Group* name and the *Class* name.

Type	Objects	Referenced Electrical CSet	Pin Pairs	Scope
Dsn	⊟ class_scope2			
MGrp	⊟ MATCH_POINT_DDRQ_LANE0			
Net	DDR2_DQ0	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ1	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ2	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ3	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ4	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ5	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ6	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ7	DDR2_DQ0	Longest Pin Pair	Global
MGrp	⊟ MATCH_POINT_DDR2_LANE1			
Net	DDR2_DQS1	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ8	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ9	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ10	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ11	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ12	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ13	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ14	DDR2_DQ0	Longest Pin Pair	Global
Net	DDR2_DQ15	DDR2_DQ0	Longest Pin Pair	Global

Match Group Rules

The following rules apply to *Match Groups*:

- You *must* specify a *Match Group* in only the *Relative Propagation Delay* worksheet of the *Routing* workbook.
- You can set *Match Group* constraints for the entire group and override individual members of the group as desired, offering differing levels of *delta* and *tolerance*.
- You specify *Match Group* delays at the *design*- or the *system*-level.
- You can include a *Match Group* member in multiple Match Groups.
- A match delay constraint from a pre-14.0 board database is upreved with a *delta* value of zero. This implies that all group members will match a specified target pin pair.

  Note: Constraint Manager, when launched from Allegro PCB Series L, supports *Match Groups* only on net-related objects, not on *Electrical CSets*.
- If you are specifying a *Bus*- or *Class*-scope, you must create the *Match Group* in a CSet in the *Electrical CSet* folder.

See "Match Groups" on page 56 for more information on *Match Group* objects.

Multi-group Membership

You can include a member of a *Match Group* in another *Match Group*. In this way, you can constrain the same member differently in each *Match Group*. In Figure 2-9, members *CAS0L* and *CAS1L* are members of two *Match Groups*. Furthermore, *CAS1L* is constrained differently in each *Match Group*.

Figure 2-9 Multi-group Membership

Objects	Scope	Rel Delta:Tolerance ns
System		
lesson5b		
MATCH_GROUP_B	Global	100 ns:5 %
BHEL	Global	100 ns:5 %
BLEL	Global	100 ns:5 %
CAS0L	Global	100 ns:5 %
CAS1L	Global	100 ns:5 %
MATCH_GROUP_C	Global	100 ns:5 %
CAS0L	Global	100 ns:5 %
CAS1L	Global	150 ns:10 %

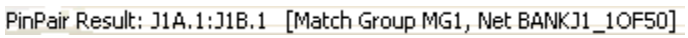
Note: Refer to the *Objects – Create – Match Group* and *Objects – Membership – Match Group* commands in the *Constraint Manager Reference* for information on including nets, Xnets, and pin pairs in multiple *Match Groups*.

Analyzed values appear in the *Actual* and *Margin* columns of a net, Xnet, or pin pair. If you include the net, Xnet, or pin pair in a *Match Group*, the analyzed values appear on that member in each *Match Group* to which it belongs, and not on the object's row outside of the *Match Groups*. To obtain the value of a net, Xnet, or pin pair that is a *Match Group* member, you must expand that *Match Group*.

With multi-group membership, use the following techniques to locate *Match Group* member objects:

- *Status Bar*

  Highlight a net, Xnet, or pin pair in the Object's column and observe the *status bar*. Constraint Manager reports all parent objects in which the member belongs.

  In the example above, the *status bar* (located at the lower-left corner of Constraint Manager) identifies the pin pair, its parent net, and the parent *Match Groups*.

- *Find*

  Use the *Find* command (choose *Edit – Find*) to locate the selected object. Use the *Find Next* command (choose *Edit – Find Next*) to find the next occurrence of the object in your design. One at a time, Constraint Manager locates the next occurrence of the selected object in all parent objects to which it belongs, including multiple *Match Groups*.

- *Select*

  Use the *Select* command to highlight all occurrences of the selected object, including instances of the object in multiple *Match Groups*.

- *Filter*

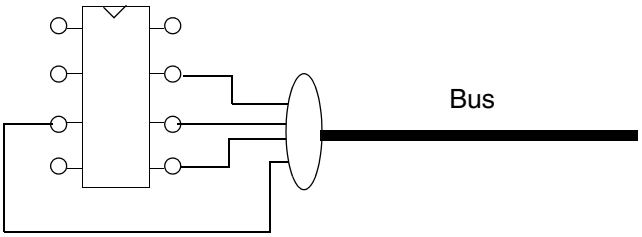
  Use the *Filter* command (choose *Objects – Filter)* to locate only the occurrences of a selected net or Xnet, including instances of the object in multiple *Match Groups*. Where *Select* locates an instance of an object in multiple groups, *Filter* lets you filter out members of the *Match Group*.

Buses

Important

Starting 16.6 release, creation of new user-defined buses is not supported in Constraint Manager. To create a collection of net objects, use of Net Groups is recommended.

A *bus* represents a named collection of differential-pairs, Xnets, or Nets.



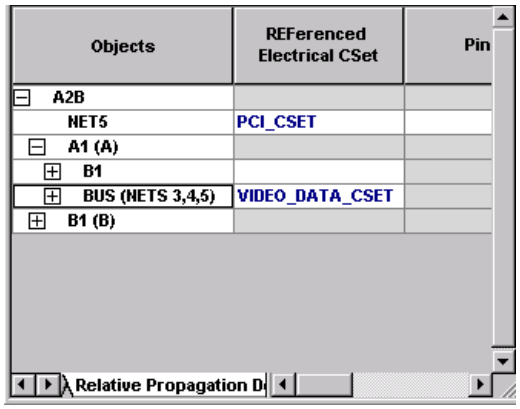
See Constraint Architecture in the *Allegro Platform Constraints Reference* for more information on the *Bus* constraint object.

You can use a bus to group functionally similar nets, Xnets, and differential pairs. Constraints captured on a bus are inherited by all members of the bus.

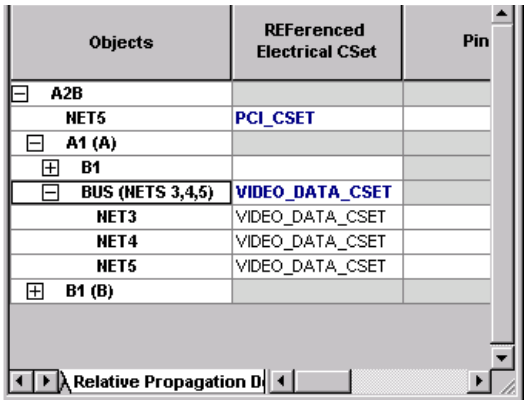
You can create an Electrical CSet based on the characteristics of a bus. You could then use SigXplorer to define pin scheduling of bus members and to augment constraint information.

When you associate the Electrical CSet with a bus, all members (bits) of the bus inherit the constraints defined in the Electrical CSet. In Figure 2-10, the `VIDEO_DATA_CSET` Electrical Cset is referenced to `BUS (NETS 3,4,5)`. Each member of the bus inherits the properties defined in this Electrical CSet.

Figure 2-10 Bus Inheritance



Collapsed Bus



Expanded Bus

See Chapter 3, “Working With Reusable Constraint Objects — CSets” for information about creating CSets and associating them with objects

See Chapter 6, “Using Constraint Manager with Other Tools Across the Allegro Platform” for information on using SigXplorer.

Bus Rules

The following rules apply to creating a bus

- You can create a bus in all net worksheets
- When used in conjunction with Allegro Design Entry HDL, Constraint Manager cannot create a bus. A bus is only realized with the appropriate property in the schematic
- A bus must be at the design-level (not the system-level).

Net Groups

A Net Group is a collection of various net (signal) objects. Different type of net objects, such as nets, buses, differential pairs, and XNets, can be the members of a Net Group. Constraints defined on a net group are applicable to all member objects.

Note: With the introduction of net groups, user-defined collection of net objects is now created as net groups instead of buses.

Net Group Rules

- A net object can only be a member of one net group.
- Net groups are design specific.

RKO Groups

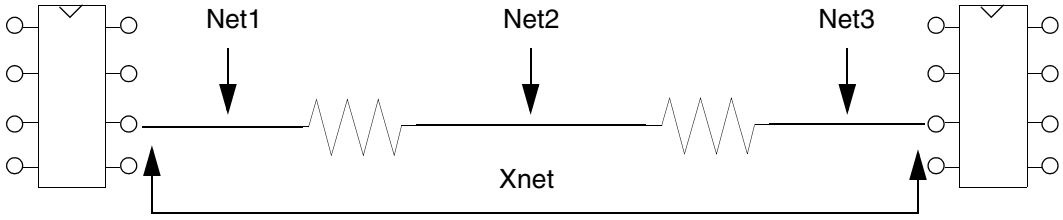
An RKO Group is a collection of various net (signal) objects. Different type of net objects, such as nets, buses, differential pairs, net groups, and XNets, can be the members of an RKO Group.

The members of the RKO group do not show route keepout DRCs.

Nets and Xnets

A *net* represents an electrical connection from one pin to another pin (or pins) on the same device or on a different device.

See Constraint Architecture in the *Allegro Platform Constraints Reference* for more information on Xnets and Nets.



If the path of a net traverses a passive, discrete device (resistor, inductor or capacitor), then each net segment is represented by an individual net entity in the board database. Constraint Manager, however, interprets these net segments as a contiguous extended net, or an *Xnet*. An Xnet can also traverse connectors and cables in a multi-board configuration.

You can associate a net or Xnet with an CSet. See Chapter 3, “Working With Reusable Constraint Objects — CSets” for information about creating CSets and associating them with objects.

How to Control Extended Net (Xnet) Naming

An Xnet name depends on a set of rules and is visible in Constraint Manager. You can, however, rename the Xnet to any of its members nets using *Rename* command. You can additionally revert to the default Xnet name utilizing the same command.

For more information, see *Objects – Rename* command.

The rules governing how an Xnet is named are as follows:

- One of the names of an Xnet's member nets is always assigned as an Xnet name.
- In case of a hierarchical schematic design, only the nets at the top-level of the hierarchy are considered during Xnet naming.
- Unnamed nets are assigned system-generated net names in a design. These nets are not considered for Xnet naming unless there are no other nets.
- If any of the nets has no driver pins and the another net has driver pins, the nets with no driver pins are considered in creating the Xnet name.

Note: Bi-directional pins are treated as driver pins.

- If all the nets have driver pins, they all are considered in creating the Xnet name.
- If none of the nets has driver pins, the nets with the lower number of receiver pins are considered in creating the Xnet name.
- If none of the nets has driver or receiver pins, all the nets are considered in creating the Xnet name.

Note: If more than one net is still under consideration, the lexicographical order is considered in creating the Xnet name.Electrical differential pairs are dependent on the names of the Xnets (or nets) that belong to the differential pair, therefore differential pairs in renamed Xnets are renamed as well.

System-level Xnet names can be chosen by selecting from which design in a system the Xnet name should derive (in the System Configuration Editor Design Link controls accessed from *Analyze – Initialize*). For example, a system Xnet contains nets in three designs:

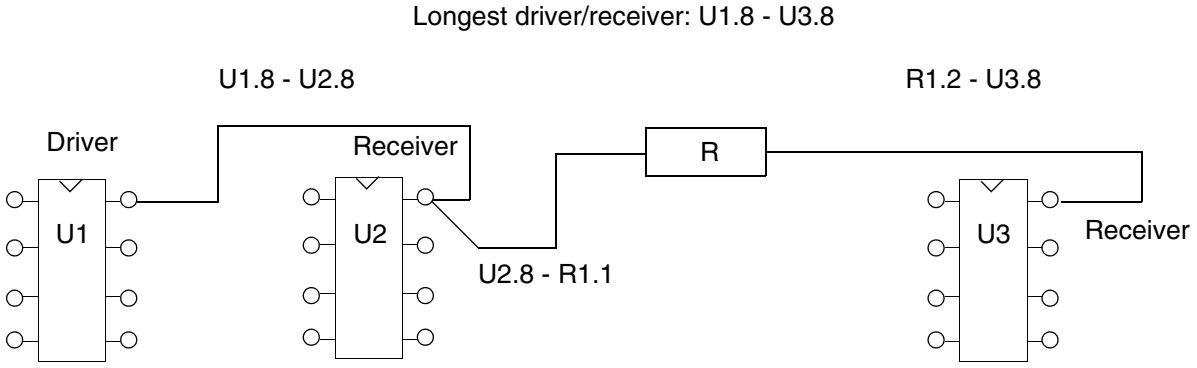
Net Name	Design
AAA	D1
BBB	D2
CCC	D3

If you specify that the net in design D2 is to govern the system Xnet name, the Xnet name becomes BBB.

Pin Pairs

A *pin pair* represents a pair of logically connected pins, often a driver-receiver connection. Pin Pairs may not be directly connected but they must exist on the same net or Xnet.

See Constraint Architecture in the *Allegro Platform Constraints Reference* and the Objects – Create – Pin Pairs command.



You use pin pairs to capture specific pin-to-pin constraints for a net or an Xnet. You can also use pin pairs to capture generic pin-to-pin constraints for CSets. Generic pin pairs are used to automatically define net- or Xnet-specific pin pairs when the CSet is referenced.

You may specify pin pairs explicitly (for example, `U1.8 U3.8`), or they can be derived based on the following criteria:

- longest pin pair
- longest driver-receiver pair
- all driver-receiver pairs

Note: Physical and Spacing pin pairs cannot be derived; you must create them explicitly.

Tip

When you import a topology file from SigXplorer and apply the *Electrical CSet* to a net object, Constraint Manager creates pin pairs on the net or Xnet based on those defined in the original topology file.

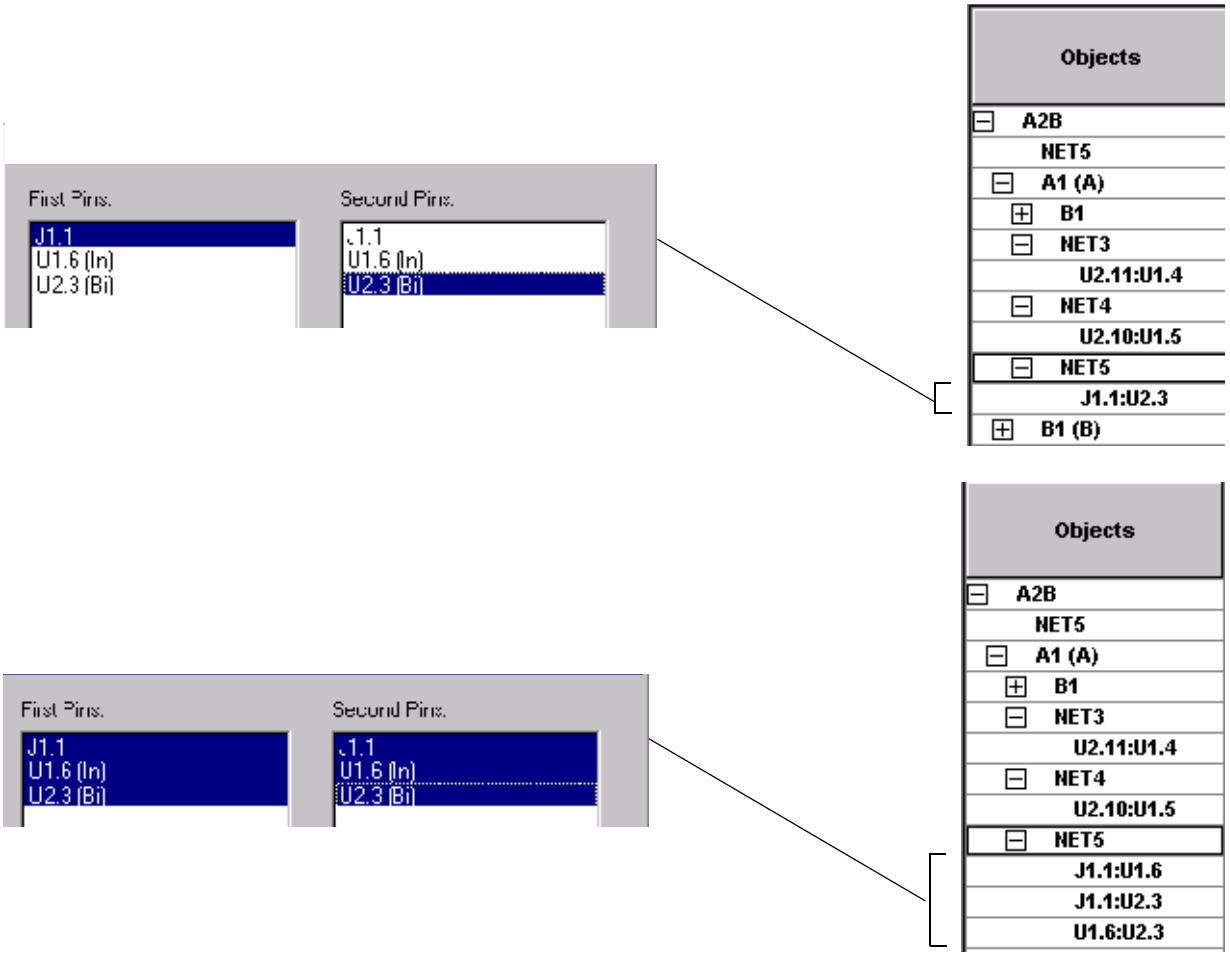
Once established, you associate a pin pair with a CSet. See Chapter 3, “Working With Reusable Constraint Objects — CSets” for information about creating CSets and associating them with objects.

Note: The Allegro Design Entry HDL database does not directly support CSet, Match Group, differential pair, pin pair, and Xnet objects. Constraint Manager does, however, update and validate constraints on these objects in the schematic.

Examples of pin pairs

Figure 2-11 shows two examples of creating pin pairs. In the top example, a single, one-to-one pin map is specified on Net 5 (`J1, Pin 1` to `U2, Pin 3`). In the bottom example, all permutations of pin mappings is realized. If all *first* pins and all *second* pins are selected, Constraint Manager maps all source pins to all target pins while excluding each pin combination that appears in both lists.

Figure 2-11 Example Pin Pairs



Important

Use `Shift-click` to select a range of pins in the pin list or `Control-click` to selectively highlight pins.

Pin Pair Rules

The following rules apply to creating pin pairs

Physical, Spacing, and Electrical domains

Pins must exist in the object from which you create the pin pair.

Electrical domain

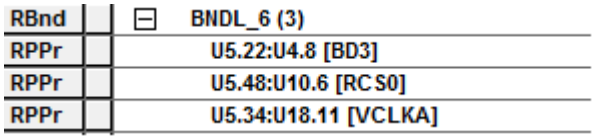
- Pin Pairs can only be defined in the following worksheets:

Electrical Workbook	Electrical Worksheet
All Constraints	■ *Signal Integrity/Timing/Routing*
Timing	■ *Switch/Settle Delays* ■ *Setup/Hold*
Routing	■ *Impedance* ■ *Min/Max Propagation Delay* ■ *Relative Propagation Delay*

- Objects in the *All Constraints* and *Timing* worksheets must have a driver and a receiver as pin pairs.
- Pin Pair length is the length of the etch path between the two pins, if the pins are routed. If not routed, the total manhattan distance of the ratsnest lines connecting the pins is used.
- Constraint Manager determines longest/shortest pin pair length based on drivers and receivers. If there are not any drivers or receivers, all pins in an Xnet are considered.
- For a relative propagation delay constraint, only the longest pin pair is determined.

Ratsnest Bundle

A *Ratsnest Bundle* constraint object lets you organize and manipulate ratsnest connections as a named group. This makes it more efficient to work with the *Interconnect Flooplanner* and the *Global Route Environment*. You edit ratsnest bundle attributes in the *Ratsnest Bundle Properties* worksheet in the *Properties* domain.



Note: Constraint Manager recognizes ratsnest bundles, and their attributes, as defined in a layout editor.

The following applies to a ratsnest bundle:

- Bundles display *RBnd* in the *Objects* column.
- Bundle pin pair members display *RPPr* in the *Objects* column.
- You can pre-populate a ratsnest bundle by selecting pin pairs before creating the bundle, or you can add members later.
- You can crossprobe to highlight a ratsnest bundle in a physical editor.
- You can sort the bundle pin pair members. The Constraint Manager sorts the bundle pin pair members on the basis of net names that are defined in the square bracket for every pin pair.

For more information about ratsnest bundles, see

- the B commands bundle operations for procedural information.
- the Objects – Create – Ratsnest Bundle command.
- the Objects – Membership – Ratsnest Bundle command.

Region

A *Region* constraint object adds or modifies constraints on all nets that cross the boundaries of the region's shape.

Objects		
Type	S	Name
*	*	*
Dsn		3D_DemoM
Rgn		BGA
Rgn		CONN_FLEX
Rgn		FLEX
Rgn		LCD_FLEX

See Constraint Architecture in the *Allegro Platform Constraints Reference* and the Objects – Create – Region command.

Region Rules

The following rules apply to creating a *Region.* You

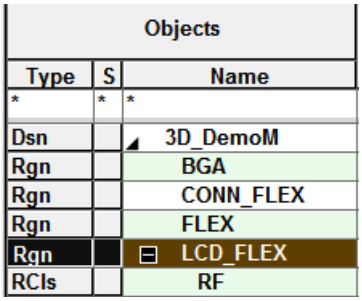
- specify a *Region* in the *Region* workbook in either the *Physical* or *Spacing* domain
- delimit a *Region* with a geometric shape, or a group of shapes, that you draw on a subclass layer in PCB Editor
- constrain a *Region* with a *Physical*- or *Spacing-CSet*
- constrain a *Region* directly with explicit constraint values (though Cadence recommends using a *CSet*)

Note: *Regions* are not supported in Design Entry HDL or Allegro System Architect.

If *Region Class-Class* does not have an explicit CSet reference or directly-set values it does not inherit constraints from the parent *Class*.

Region Class

A *Region Class* constraint object lets you constrain the members of a *Net Class*—within a *Region*—differently than the original constraints on that *Region*.



Allowable members of a *Region Class* include *Net Classes*.

See Constraint Architecture in the *Allegro Platform Constraints Reference* and the Objects – Create – Region Class command.

Region Class Rules

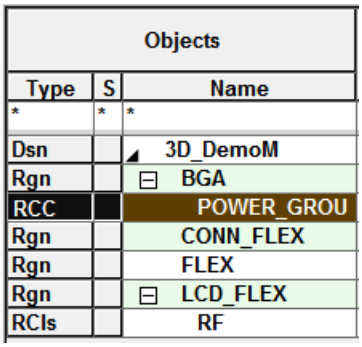
The following rules apply to creating a *Region Class.* You can

- constrain a *Region Class* with a *CSet*
- override individual members of a *Region Class*
- constrain a *Region Class* directly (though we recommend using a *CSet*)
- create a *Region Class* only in the *Physical* or *Spacing* domains

Note: *Region Class* are not supported in Design Entry HDL or Allegro System Architect.

Region Class-Class

A *Region Class-Class* constraint object lets you specify the minimum spacing between members of *Net Classes* within a *Region*.



See Constraint Architecture in the *Allegro Platform Constraints Reference* and the Objects – Create – Region Class-Class command.

Region Class-Class Rules

The following rules apply to creating a *Region Class-Class.* You can

- constrain a *Region Class-Class* with a *CSet*
- override individual members of a *Region Class-Class*
- constrain a *Region Class-Class* directly (though we recommend using a *CSet*)
- create a *Region Class-Class* only in the *Spacing* domain

Note: *Region Class-Classes* are not supported in the *Same Net Spacing* domain, or in Design Entry HDL or Allegro System Architect.

3

Working With Reusable Constraint Objects — CSets

Topics in this chapter include

Reusable Constraints

This chapter presents information on how to use reusable constraint objects in Constraint Manager.

Constraining Objects

You can constrain an object by referencing a CSet or setting a constraint value directly on the object. An object that is not constrained will inherit constraint values through the precedence rules of the constraint object hierarchy for overrides.

Constraint Sets (CSets)

The *Electrical*, *Physical, Spacing* and *Same Net Spacing* domains support Constraint Sets (CSets). A CSet is a named, reusable collection of constraint values. CSets are not supported in the *Design* domain.

Tip

You define generic rules in the *Constraint Set* object folder. These generic rules can subsequently be applied to objects in the *Net*, and *Region* folders.

As design requirements change, you can

- edit the CSet constraints. All objects that reference the CSet will automatically inherit these changes.
- assign a different CSet, one that reflects a different rule-set, to the object.
- specify override properties on individual objects. Cells with overrides are colored blue.

Tip

CSets can be referenced by any number of objects (*Bus, Differential Pair, Xnet, Net, Net Class, Net Class-Class, Region, Region Class Region Class-Class*) but an object may reference only one CSet per domain.

Physical, Spacing, and Same Net Spacing CSets

A *Physical CSet* consists of one value per layer for each physical constraint. A *Spacing CSet* consists of one value per layer for each spacing constraint. *Spacing CSets* are further classified into net-to-net and same net domains. In all designs, Constraint Manager provides one *Physical*, one *Spacing*, and one *Same Net Spacing* CSet, named *DEFAULT*, which you cannot delete or rename; however, you can modify the DEFAULT CSet constraints to suit your design requirements.

Electrical CSets

With an *Electrical CSet*, you define the constraints in the set. There is no pre-defined configuration, nor any pre-defined values. You can delete *Electrical CSets*.

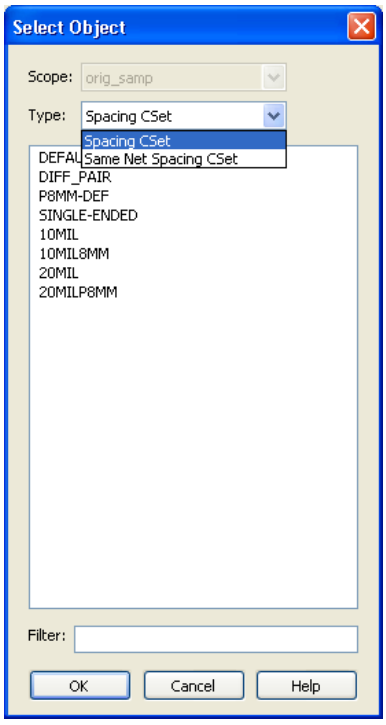
Copying Constraints from CSets

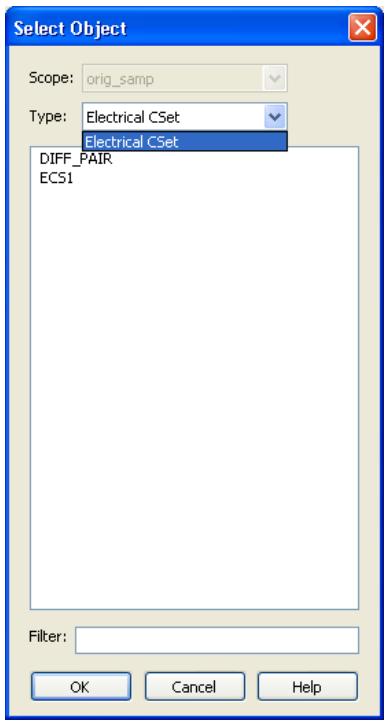
You can copy constraints in one CSets to another CSet in the same domain for PCSets and ECSets and (additionally) from SCSets and SNSCSets and vice versa in the spacing and same net spacing domains.

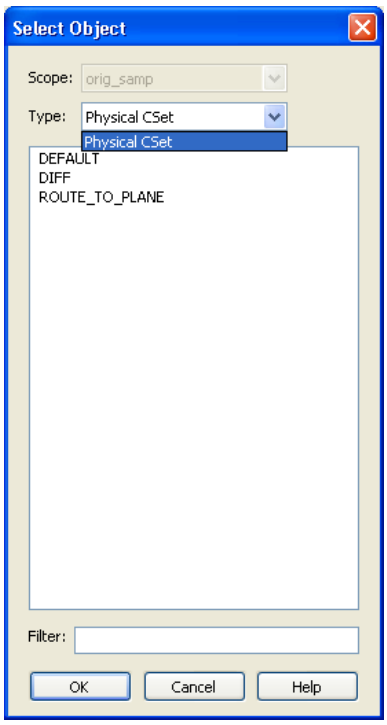
To copy constraints from one CSet to another:

1. Right-click in the *Objects* column for a PCSet, ECSet, SCSet, or SNSCSet.
2. Choose the *Copy Constraints from* menu command in the resultant pop-up menu.

Depending on the domain, the Select Object dialog is populated with the appropriate CSets:







In the following example, a Spacing CSet is copied to an SNSCSet (Same Net Spacing) domain.

1. In the Same Net Spacing domain, right-click an SNSCSet in the Object's column and choose Copy Constraints from menu.
2. Choose *Spacing CSet* from the *Type* drop-down list.
3. Choose the SCSet from the list and click *OK*.

   Note: For the Spacing CSet, the default type of source object is Same Net Spacing CSet and list of the objects contains all the SNSCSets. For the SNSCSet, the default type is SCSets. You can also copy constraints information from the CSets of the same type.

   When you copy constraints from an SCSet to SNSCSet, constraint names are mapped and the `SN_` prefix is added to the constraint name.

   `LINE_TO_LINE_SPACING` is copied to `SN_LINE_TO_LINE_SPACING`

   The SNSCSet is updated with the SCSet. Constraints are copied in the merge mode, which implies that any existing constraint in the target CSet remains intact if it is not present in the source CSet.

Editing Multiple Electrical Constraints

You can edit constraints in CSets by selecting multiple cells.

However, multi-selection is not possible in following scenarios:

- Selected cells have different data types.

  For example, if two cells have integer and float as a data type, they cannot be edited as a multi selection.

- Selected cells values are represented as an array and a single value.

  For example, if two cells have values as 1.2:3.0 and 1.2, they cannot be edited as a multi selection.

- Selected cells have different units of measurement.
- Selected enumerations have different enum values.

Editing Multiple Physical, Spacing and Same Net Spacing Constraints

You can edit multiple constraints in the Spacing, Physical, and Same Net Spacing worksheets with a single click.

- Choose the object.

  The entire row is selected except the cells which have different design unit values.

Objects			Referenced Spacing CSet	Line To >>	Thru Pin To >>	SMD Pin To >>	Test Pin To >>
				All	All	All	All
Type	S	Name		mm	mm	mm	mm
*	*	*	*	*	*	*	*
Dsn		3D_DemoM	DEFAULT	0.200	0.200	0.200	0.200
Lyr	1	TOP		0.200	0.200	0.200	0.200
Lyr	2	GND		0.200	0.200	0.200	0.200
Lyr	3	INNER1		0.200	0.200	0.200	0.200
Lyr	4	INNER2		0.200	0.200	0.200	0.200
Lyr	5	POWER		0.200	0.200	0.200	0.200
Lyr	6	BOTTOM		0.200	0.200	0.200	0.200

- Choose the cell which is in edit mode.

  The value in the cell is selected automatically.

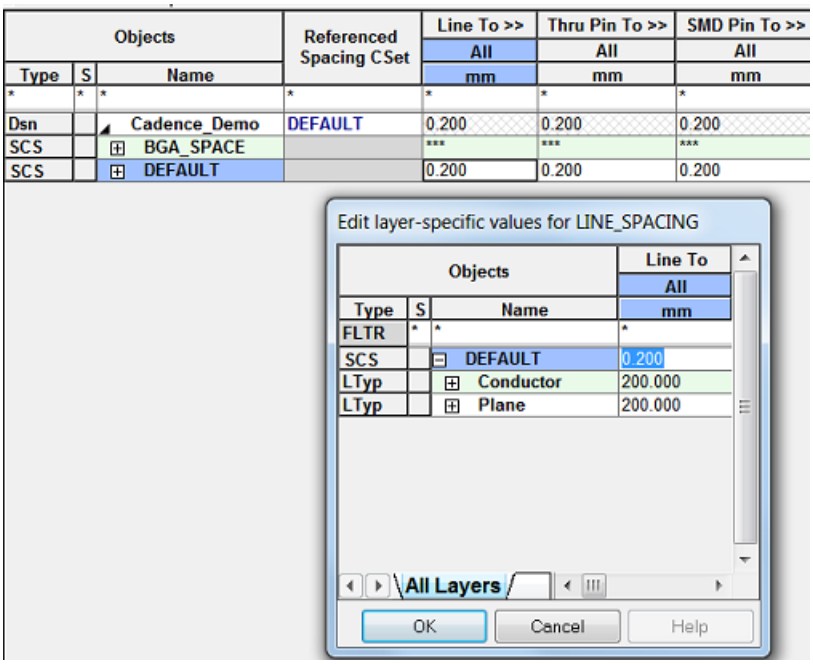
- Type the new value of design unit and press `ENTER`.

  The new value is applied to all selected cells after data validation.

You can also select multiple objects using `Ctrl` key.

Objects			Referenced Spacing CSet	Line To >>	Thru Pin To >>	SMD Pin To >>	Test Pin To >>
				All	All	All	All
Type	S	Name		mm	mm	mm	mm
*	*	*	*	*	*	*	*
Dsn		3D_DemoM	DEFAULT	0.200	0.200	0.200	0.200
SCS		⊞ BGA_SPACE		***	***	***	***
SCS		⊟ DEFAULT		0.200	0.200	0.200	0.200
LTyp		⊞ Conductor		0.200	0.200	0.200	0.200
LTyp		⊞ Plane		0.200	0.200	0.200	0.200

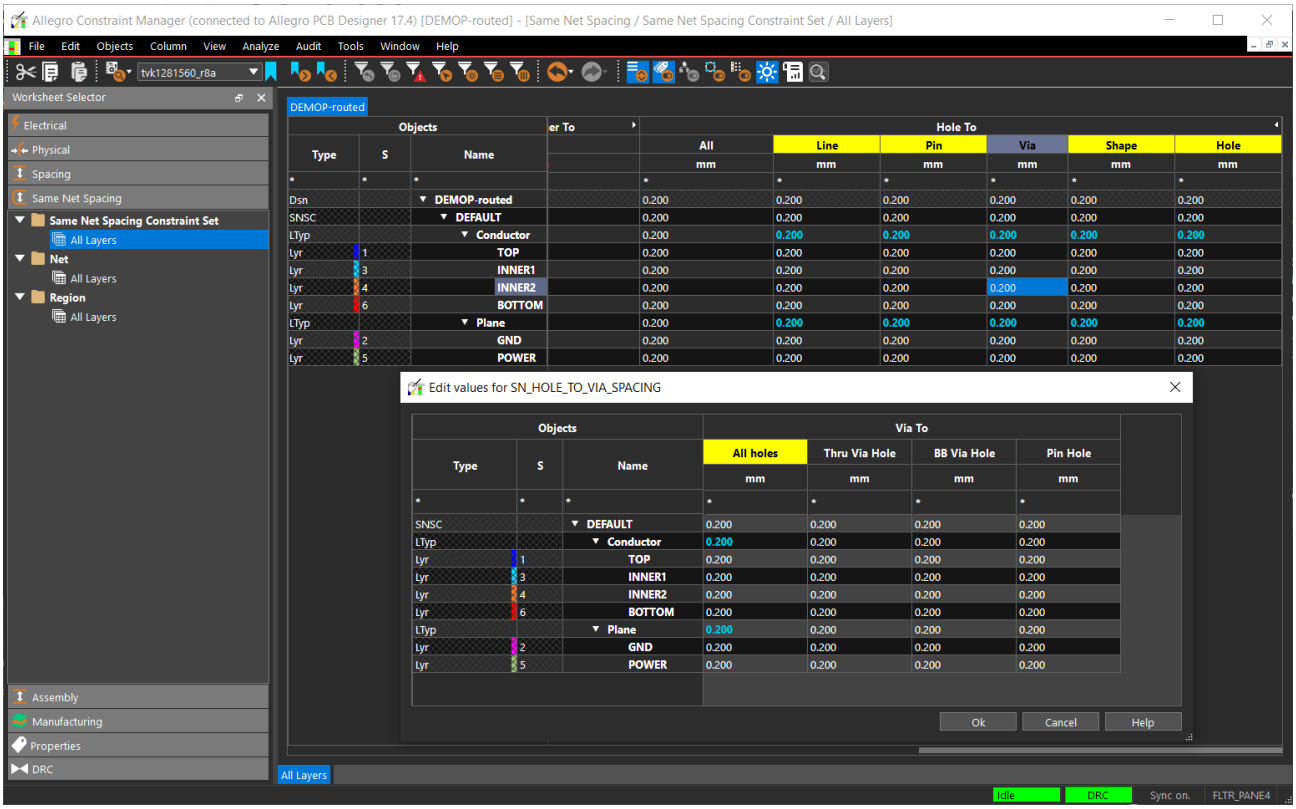
Note: If the constraint has different values per layer, clicking the editable cell opens *Edit layer-specific values for* dialog box.



Editing Hole To Hole Constraints for Spacing and Same Net Spacing Domain

You can expand the *Hole To* checks to edit separate values for Thru Via Holes, Blind/Buried Holes, Micro Via Holes and Pin Holes in both *Spacing* and Same Net Spacing domains.

Clicking any of the cells under the *Hole To* columns opens the *Edit values for* dialog box for the column where you can edit the granular *Hole To* constraint values for the all the layers.



Methods of Constraining Nets

This section covers different methods of constraining nets in your design, including:

- Inheritance
- Overrides
- Container objects
- CSet references

Inheritance

Each *Net* depicted below inherits the 5-mil *Min Line Width* from the *DEFAULT* Physical CSet.

	Type	Objects	Referenced Physical CSet	Line Width	
1				Min	Max
2				mil	mil
102	Net	USE3	DEFAULT	5.00	0.00
103	Net	USE4	DEFAULT	5.00	0.00
104	Net	USE5	DEFAULT	5.00	0.00
105	Net	USE6	DEFAULT	5.00	0.00
106	Net	VCC	DEFAULT	5.00	0.00
107	Net	VSR_RTN	DEFAULT	5.00	0.00
108	Net	24V	DEFAULT	5.00	0.00
109	Net	48V	DEFAULT	5.00	0.00

Overrides

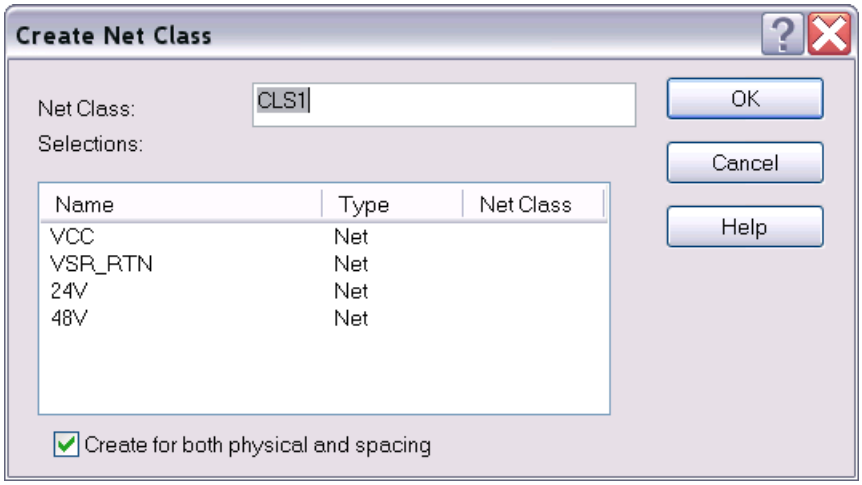
We need to increase the line width of the voltage rails (Rows 106 - 109). As these are contiguous cells, simply dragging through them and specifying 7 mils is the most-direct method of constraining these nets

	Type	Objects	Referenced Physical CSet	Line Width	
1				Min	Max
2				mil	mil
102	Net	USE3	DEFAULT	5.00	0.00
103	Net	USE4	DEFAULT	5.00	0.00
104	Net	USE5	DEFAULT	5.00	0.00
105	Net	USE6	DEFAULT	5.00	0.00
106	Net	VCC	DEFAULT	7.00	0.00
107	Net	VSR_RTN	DEFAULT	7.00	0.00
108	Net	24V	DEFAULT	7.00	0.00
109	Net	48V	DEFAULT	7.00	0.00

These directly-set constraint values are called *overrides* and appear in a blue tint. The advantage of constraining a container object is that it is quick and direct and it follows a constraint precedence; the disadvantage is that members of the container are fixed (though membership can be redefined) so constraints are not transferable to other container objects.

Container Objects

If we create a container object, such as a *Net Class*, we can constrain the container directly instead of constraining member nets individually. If we change the constraint value on the the container, members of the container automatically inherit the change in constraint value.



1	Type	Objects	Referenced Physical CSet	Line Width	
				Min	Max
2				mil	mil
3	Dsn	⊟ unnamed	DEFAULT	5.00	0.00
4	NCls	⊞ CLS1	DEFAULT	7.00	0.00

Constraint Manager presents *Net Classes* before *Nets*. The Nets from Rows 106 - 109 now appear, in collapsed form, in the *Net Class* container on Row 4.

If we expand the *Net Class*, we can see that the members inherit the new line width value of 7 mils.

1	Type	Objects	Referenced Physical CSet	Line Width	
				Min	Max
2				mil	mil
3	Dsn	⊟ unnamed	DEFAULT	5.00	0.00
4	NCls	⊟ CLS1	DEFAULT	7.00	0.00
5	Net	VCC	DEFAULT	7.00	0.00
6	Net	VSR_RTN	DEFAULT	7.00	0.00
7	Net	24V	DEFAULT	7.00	0.00
8	Net	48V	DEFAULT	7.00	0.00
9	Bus	⊞ ADDRESS_BUS	DEFAULT	5.00	0.00
10	DPr	⊞ DP_MEM_CLOC	DEFAULT	5.00	0.00

Tip

Use a net container object to group and constrain a small, focused collection of similar nets.

Referencing a CSet

Let's create a Physical CSet and define a 7-mil *Minimum Line Width* constraint.

1	Type	Objects	Line Width	
			Min	Max
2			mil	mil
3	Dsn	⊟ unnamed	5.00	0.00
4	PCS	⊞ DEFAULT	5.00	0.00
5	PCS	⊞ PCS1	7	0.00

Next, let's associate the *Physical CSet* with the *Net Class*, whose members are the voltage rails that we want to constrain.

1	Type	Objects	Referenced Physical CSet	Line
				Min
2				mil
3	Dsn	⊟ unnamed	DEFAULT	5.00
4	NCls	⊟ CLS1	DEFAULT	5.00
5	Net	VCC	DEFAULT	5.00
6	Net	VSR_RTN	PCS1	5.00
7	Net	24V	(Clear)	5.00

The advantage of referencing a CSet to impart constraints on a *Net Class* is that you can reuse the CSet to constrain similar *Net Classes,* whose members are different.

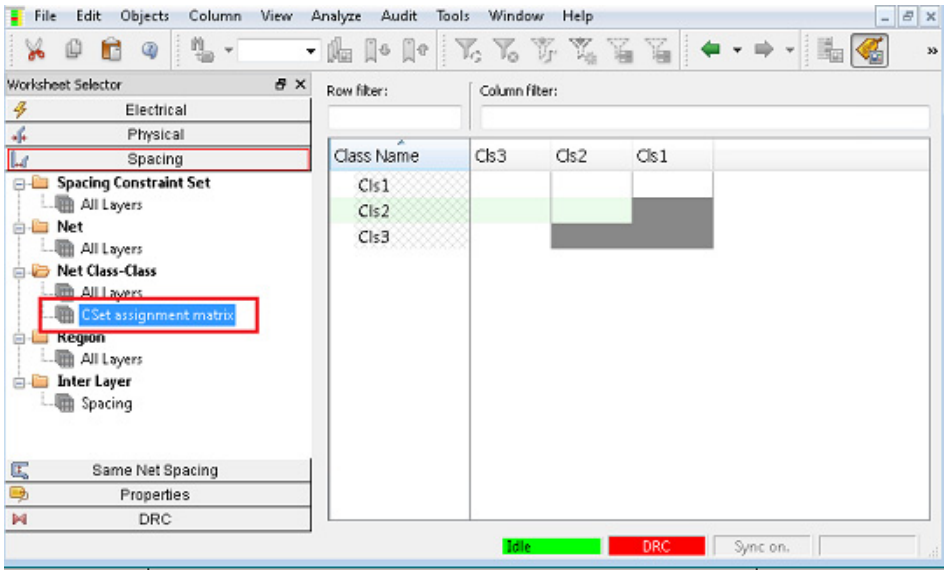
1	Type	Objects	Referenced Physical CSet	Line Width	
				Min	Max
2				mil	mil
3	Dsn	⊟ unnamed	DEFAULT	5.00	0.00
4	NCls	⊟ CLS1	PCS1	7.00	0.00
5	Net	VCC	PCS1	7.00	0.00
6	Net	VSR_RTN	PCS1	7.00	0.00
7	Net	24V	PCS1	7.00	0.00
8	Net	48V	PCS1	7.00	0.00

Tip

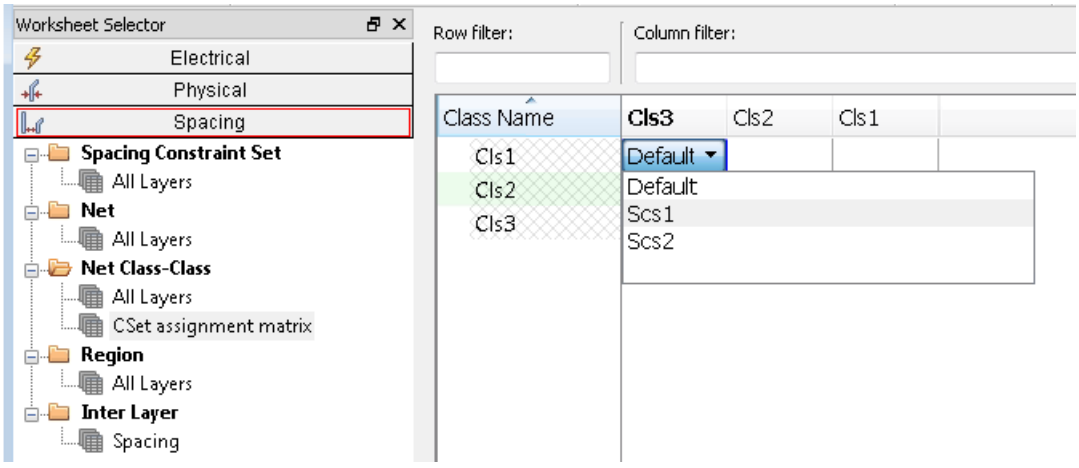
Use a net container object to group and constrain a small, focused collection of similar nets. Use a reusable CSet, referenced to one or more container objects, to apply a broad brush approach to constraining nets.

Creating Spacing Class-Class

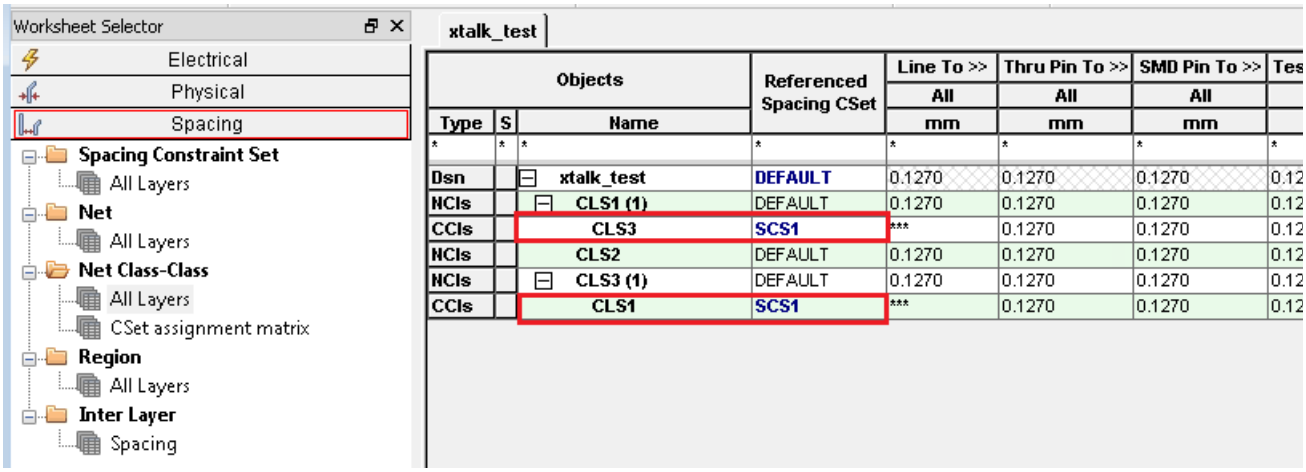
The *Cset assignment matrix* spreadsheet in the Spacing domain presents a matrix of Net Classes where Spacing Constraint Sets are available in the cells via a drop-down list. Using this spreadsheet, you can add or delete a Class-Class relationship by adding or removing the Constraint SCSets reference. Adding a Constraint SCSets reference defines a Class-Class relationship for that row and column.



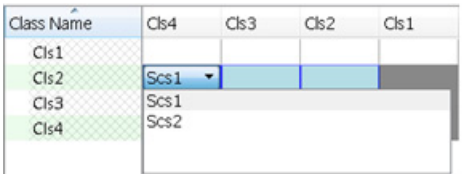
Selecting a cell for example, Cls1XCls3 and referencing a SCset *Scs1* creates a spacing class-class between Net Classes CLS1 and CLS3.

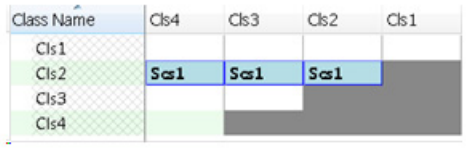


The class-class relationship is created dynamically in the Net Class-Class workbook.

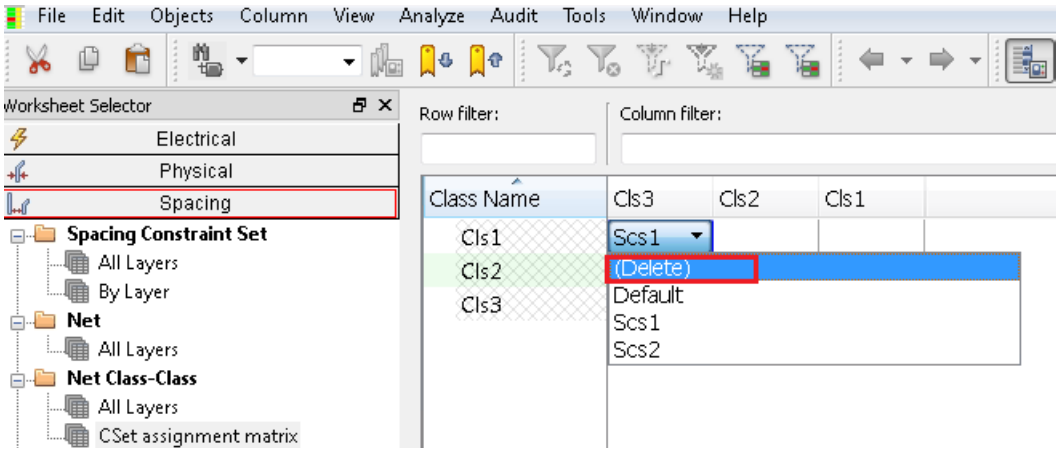


You can also create multiple class-class relationships by selecting a Net Class in the column and referencing a SCSet.

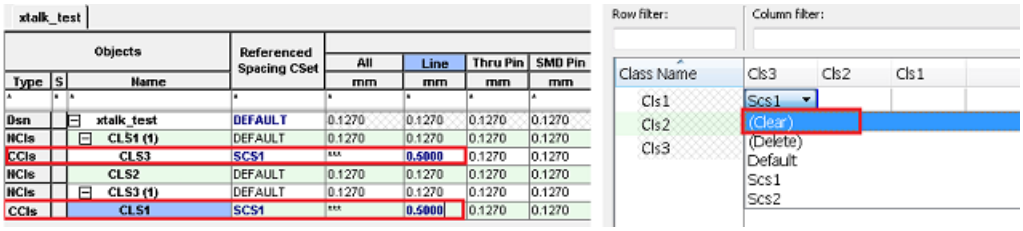




Removing a SCSet reference deletes a class-class relationship for that row and column.



If the class-class relationship has directly set values, you can either choose to clear the SCSet assignment or delete the class-class relationship.



xtalk_test

Objects			Referenced Spacing CSet	All	Line	Thru Pin	SMD Pin
Type	S	Name		mm	mm	mm	mm
Dsn		xtalk_test	DEFAULT	0.1270	0.1270	0.1270	0.1270
NCls		CLS1 (1)	DEFAULT	0.1270	0.1270	0.1270	0.1270
CCls		CLS3	SCS1	***	0.5000	0.1270	0.1270
NCls		CLS2	DEFAULT	0.1270	0.1270	0.1270	0.1270
NCls		CLS3 (1)	DEFAULT	0.1270	0.1270	0.1270	0.1270
CCls		CLS1	SCS1	***	0.5000	0.1270	0.1270

Row filter: Column filter:

Class Name	Cls3	Cls2	Cls1
Cls1	Scs1		
Cls2	(Clear)		
Cls3	(Delete)		
	Default		
	Scs1		
	Scs2		

4

ECSets and Topology Templates

Topics in this chapter include

What is a Topology Template?

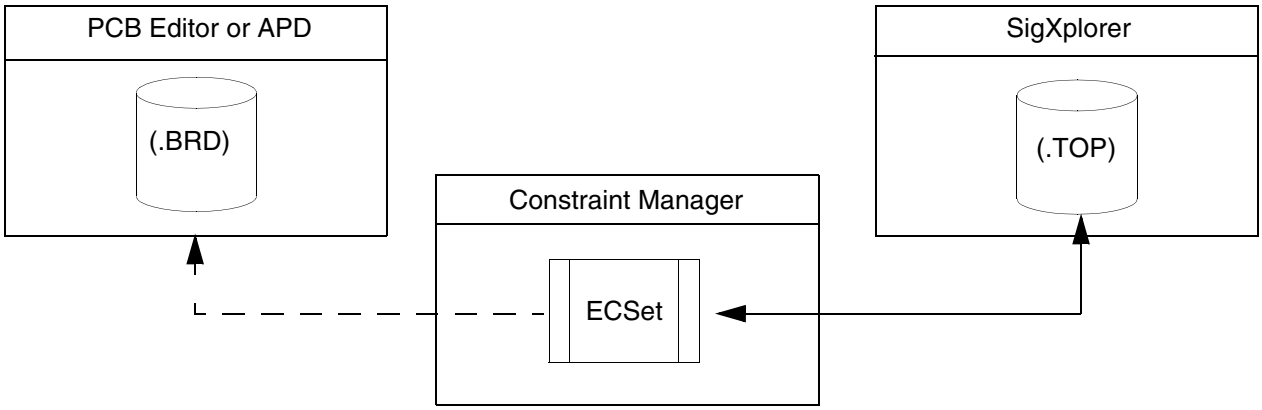
A topology template file (`.top`) is an on-disk image of the SigXplorer database. A topology template file contains the same data as an ECSet, including electrical constraints, but it also contains information to support the graphical representation of a circuit topology in SigXplorer.

> **Note:** Constraint Manager, when launched from an L Series PCB Editor, does not support topology exploration. If you are using an L Series PCB Editor, you can ignore the descriptions of the following commands in this chapter:
>
> - *File – Import Electrical CSets*
> - *File – Import Analysis Results*
> - *File – Export Electrical CSets*
> - *File – Export Analysis Results*
> - *Audit – Topology Templates*

In pre-14.0 designs, a topology template was applied to a group of target nets. In the process, constraint information was extracted from the topology file and flattened; constraints were copied to individual target nets as properties. With Constraint Manager, these properties are not flattened; they remain intact, collectively, as an ECSet. Net-related objects in the design reference the ECSet rather than a collection of individual properties.

The topology template can be imported to, and exported from, Constraint Manager. When imported, the topology template information will be instantiated within Constraint Manager as an ECSet where it can be manipulated separate from the topology template. The ECSet is saved with the database of the host application from which Constraint Manager was invoked: a board file (`.brd`) or a schematic view.

Figure 4-1 Topology Template/ECSet constraint flow



With such a close alignment between a topology template and an ECSet, you can access SigXplorer directly from Constraint Manager. In fact, you can define your constraints in SigXplorer—as a topology template—and then import this information into Constraint Manager as an ECSet. Conversely, you can define your constraints in Constraint Manager—as an ECSet—and then export this information to SigXplorer as a topology template. See the Design Exploration Phase (with SigXplorer) on page 147 for information on the Constraint Manager-SigXplorer design flow.

The only constraint that you cannot define in Constraint Manager is user-defined pin scheduling. This must be defined in SigXplorer. You can, however, select from a list or pre-defined pin schedules in Constraint Manager.

Importing ECSets

Constraint Manager promotes design reuse through the following command:

Use this command	To
File – Import – Electrical CSets	Import a selected on-disk topology template into Constraint Manager. The imported template will become an ECSet which can be referenced by net-related objects that share the same electrical characteristics. See "Mapping Templates and ECSets to Net-related Objects" on page 104. If the imported template was previously assigned as an ECset, the import will overwrite existing constraint values. **Note:** If the *Automatic Topology Update* checkbox is enabled (*Tools – Options* Figure 4-2 on page 105), the refreshed template information is immediately applied to the net-related objects; otherwise, you must choose *Tools – Update Topology* to apply the changes.

Mapping Templates and ECSets to Net-related Objects

Constraint Manager intelligently maps the constraint information, imported from a topology template or defined in an ECSet, to a candidate net that matches the topological characteristics of the referenced ECSet. If the candidate net does not match these topological characteristics, the mapping will fail and the constraints will not be applied.

In the Mapping Mode column in the *All Constraints* worksheet (in the ECSet folder), you can specify a mapping mode. Constraint Manager makes several *passes* based on the following mapping modes:

See the following commands in the Allegro *Constraint Manager Reference* for more information:

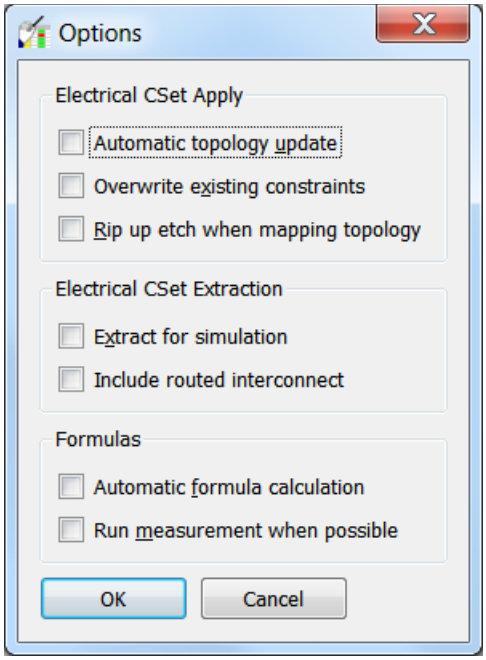
- *Objects – Electrical CSet References*
- *File – Import – Electrical CSet*

All other electrical constraints will be inherited regardless of mapping since the other constraints are not topology specific.

➤ Choose *Tools — Options* to control how objects in Constraint Manager inherit constraint information.

Figure 4-2 Options Dialog Box with Default Settings

The following describes how to use the *Electrical CSet Apply* fields of the *Options* dialog box (see Figure 4-2 on page 105).



Checkbox Option	Function
Automatic topology update (Defaults to enabled)	Controls how topology-related constraints are re-applied ■ When the design changes (component placement, signal model updates) – or – ■ When an ECSet is initially referenced When *enabled*, Constraint Manager applies changes on-the-fly as the design changes. When *disabled*, you can apply changes by choosing *Tools — Update Topology*. If you change state from *disabled* to *enabled*, Constraint Manager presents you with a confirming message stating that it will refresh stale nets and Xnets with updated topology data.  *Tip* Disabling automatic topology update may be necessary when design changes are frequent and complex ECSets are referenced.
Overwrite existing constraints (Defaults to disabled)	Controls whether constraint values in the ECSet will overwrite any existing net-related constraints when an ECSet is re-applied. See "Methods of Constraining Nets" on page 94 for information about overriding inherited constraint values.  *Important* Enabling *overwrite existing constraints* is necessary when migrating pre-14.0 designs using the *Audit – Topology Properties* command. This will ensure that all net-related overrides— created by the pre-14.0 topology template mapping software—are removed.

Checkbox Option	Function
Rip up etch when mapping topology (Defaults to disabled)	Controls whether etch (clines and vias) is removed when an ECSet is re-applied and the schedule of the net changes.

Audits

Constraint Manager provides audits to give you feedback, in report form, about the constraints and their references in the design. Audit commands are available when Constraint Manager is invoked from a PCB- or design entry-editor, or APD. The following audits (accessible from the *Audit* menu) relate to ECSets.

Run this audit	To
Constraints	List net-related overrides and constraint violations.
Obsolete Objects	List objects that should, but no longer exist in in the board or schematic database, yet they are still being referenced in Constraint Manager.
Electrical CSets	List the mapping status of all objects which reference ECSets. **Note:** Constraint Manager, when launched from a Series L editor, does not support pin pairs. Any pin pairs that exist will appear as 'not supported' in the audit report.
Topology Properties	Migrate pre-14.0 ASSIGN_TOPOLOGY and TOPOLOGY_TEMPLATE properties to an ECSet reference. **Note:** Constraint Manager, when launched from an L Series PCB Editor, does not support topology exploration.

The sections that follow describe these audits.

Constraint Audit

The Constraints audit (*Audit – Constraints*) generates a report listing constraint errors. This report aids you in troubleshooting constraint violations. The audit includes the following checks:

- Min values that exceed Max values
- Values less than zero
- Completeness violations

- Group membership violations
- Relative group violations
- Paired parallelism lengths and gap
- Setup and hold relative to clock period
- Differential pair member mismatches
- Net-related overrides

  Note: A net override lets you replace the inherited value from an ECSet with a value that you specify on a net-by-net basis. Cells with net overrides are colored blue.

Obsolete Objects Audit

The *Obsolete Objects* audit (*Audit – Obsolete Objects*) generates a report that lists objects that must be reconciled between Constraint Manager and the PCB or schematic databases. Constraint Manager displays a *No Obsolete Objects* message as appropriate.

For example, if you use Constraint Manager to constrain an object in the schematic, that object will be stored in the schematic editor's constraint view of the HDL library. If you later delete that object in the schematic, that constraint will still be in Constraint Manager until it is reconciled with the obsolete objects audit.

This command is used subsequent to importing a dictionary and constraint file (*File – Import – Constraints*) or when the connectivity is disjoint between the component or net in schematic and the corresponding Constraint Manager object.

Note: The *Audit – Obsolete Objects* command is *not* available when running Constraint Manager in stand-alone mode.

The Audit Obsolete Objects dialog box contains the following fields:

Table 4-1 Obsolete Objects Dialog Box Options

Use this field	To
Type	Filter on an object type (bus, net, Xnet)
Obsolete Objects	List all objects that no longer exist in the PCB or schematic database, yet exist in Constraint Manager

Existing Objects	List all objects that exist in the PCB or schematic database, and in Constraint Manager
Delete	Remove objects, listed in the *obsolete objects* list, from the PCB or schematic database.
Merge	Assign all properties and constraints from the object selected in the *obsolete object* list to the object selected in the *existing object* list (properties and constraints on the existing object are not overwritten).

Electrical CSets Audit

The *Electrical CSets* audit (*Audit – Electrical CSets*) generates a report listing the current ECSets in the design and the status of all net-related objects that reference them. The head of the report summarizes the number of ECSet references and any errors.

The status reports the inheritance for each constraint defined in the ECSet including:

- Any mismatch of the topological characteristics between a net-related object and the ECSet (in which case, the constraint from the ECSet is not inherited).
- The net-related object that inherits the ECSet constraint.

Tip

When the head of the *Referenced Electrical CSet* column is yellow, this indicates there is a stale ECSet reference. You run the *Electrical ECSet* audit to clear this setting and to resolve any discrepancies.

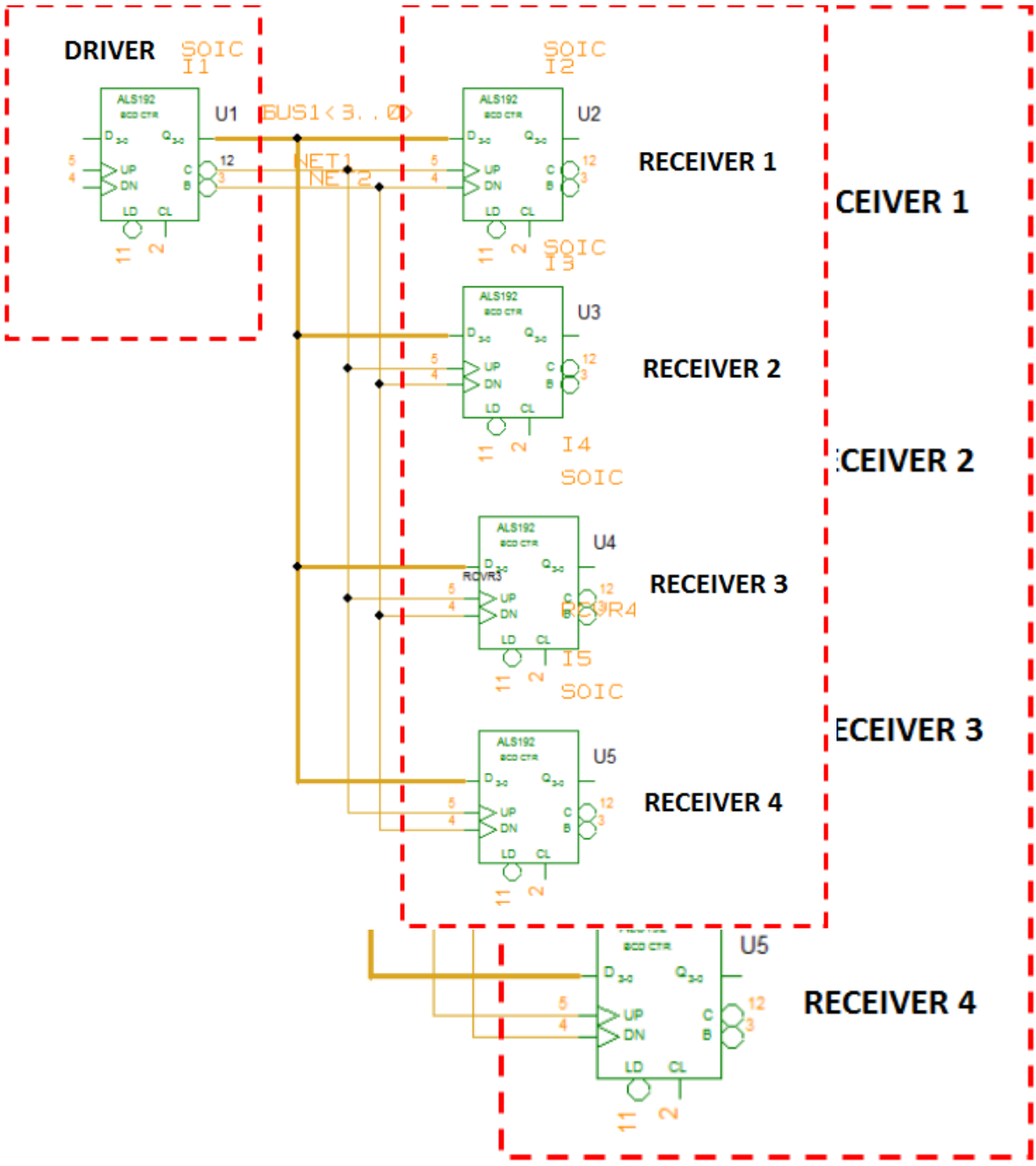
Mapping ECSets to Nets using Tags

Constraint Manager maps the constraint information defined in an ECSet to a (X)net related objects. Applying an ECSet to a (X)net involves mapping of pins in the ECSet to the component pins in the design for that (X)net. The process of ECSet mapping is based on signal model assignment, pinuse, and RefDes. If the pins of target nets use similar signal model and pinuse, they can only be differentiated by RefDes.

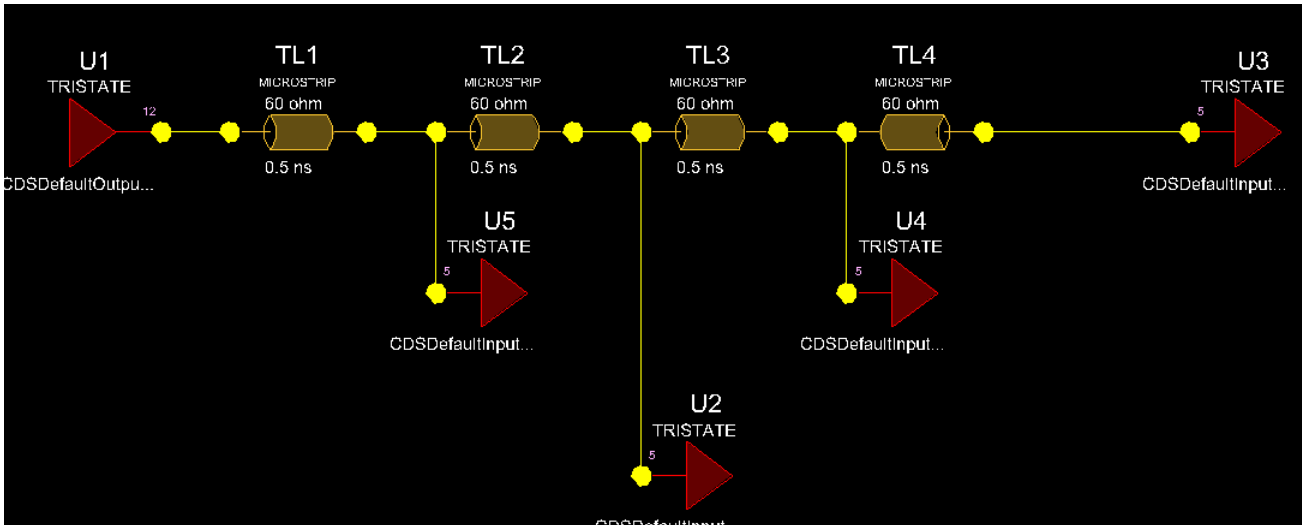
In case a (X)net topology has components of the same type (for example, all receivers with same buffer model), the mapping of pins in an ECSet to the pins of a target net is arbitrary. This type of ambiguity is avoided by including RefDes information in the ECSet to match the RefDes information in the design. However, in a hierarchical design or in a design with reused blocks where the same ECSet is applied to multiple objects, the mapping of ECSet to target

nets is unpredictable and prone to errors because there is no relation between RefDes information in the ECSet and the design.

The following illustration is an example of a circuit using driver and receiver components of the same type.



In this example, when an ECSet is applied to NET2 in Constraint Manager, the mapping process can distinguish between U2, U3, U4, and U5 only by Refdes information.



Mapping ECSets to (X)nets using Tags

To resolve ambiguities during ECSet mapping, a tag is associated with an ECSet to uniquely identify a pin. Tags should be defined for all pins that have the same signal model and pinuse. When you apply a tag-based ECSet to a (X)net, these tags are mapped to the corresponding tags on the component/pins.

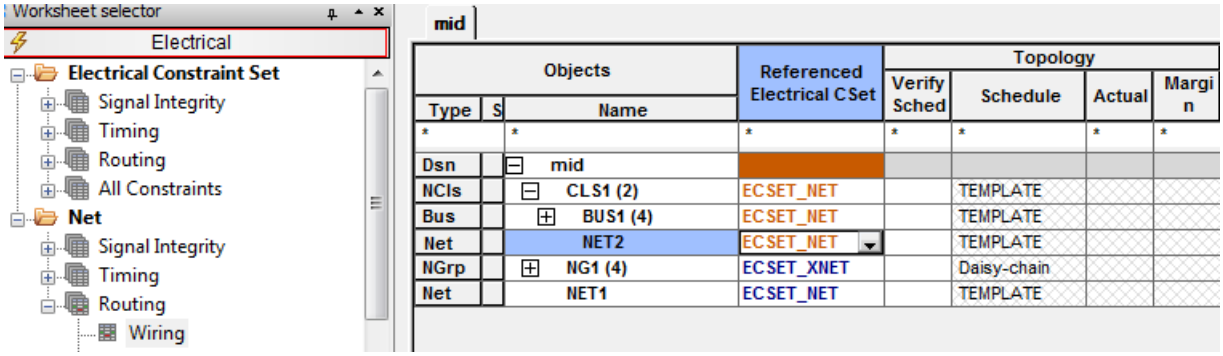
If tags on the component/pins are missing in a design and a tag-based ECSet is referenced, the mapping process indicates that the tags are present on the ECSet nodes, but not on the component/pins of the target net in the design. Auditing the referenced ECSet, assigned to a (X)net, lets you apply the ECSet tags to the desired component/pins in the design.

The tags on the component/pins are saved as ECSET_MAPPING_TAG property in the design.

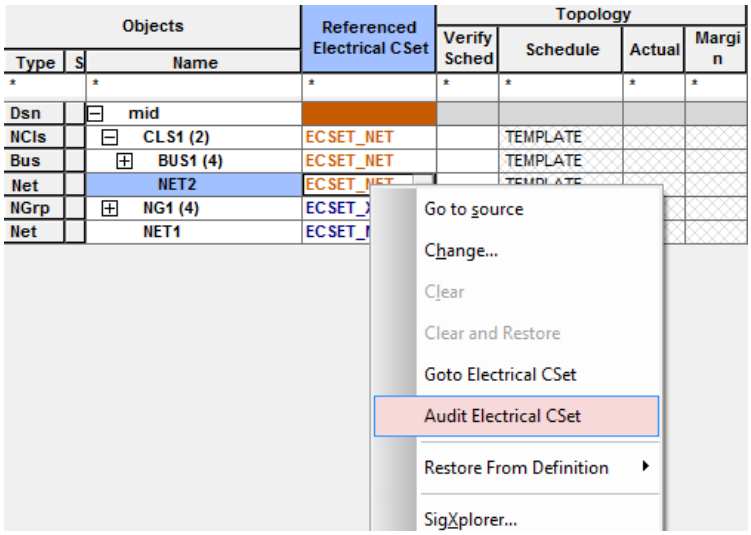
The tag-based ECSet mapping solution is available only with the High Speed option.

Mapping ECSets to (X)nets in Constraint Manager

If a tag-based ECSet is referenced, but the tags are not used to map the (X)net pins, the applied ECSet appears as an *Out of date* value (Orange color).

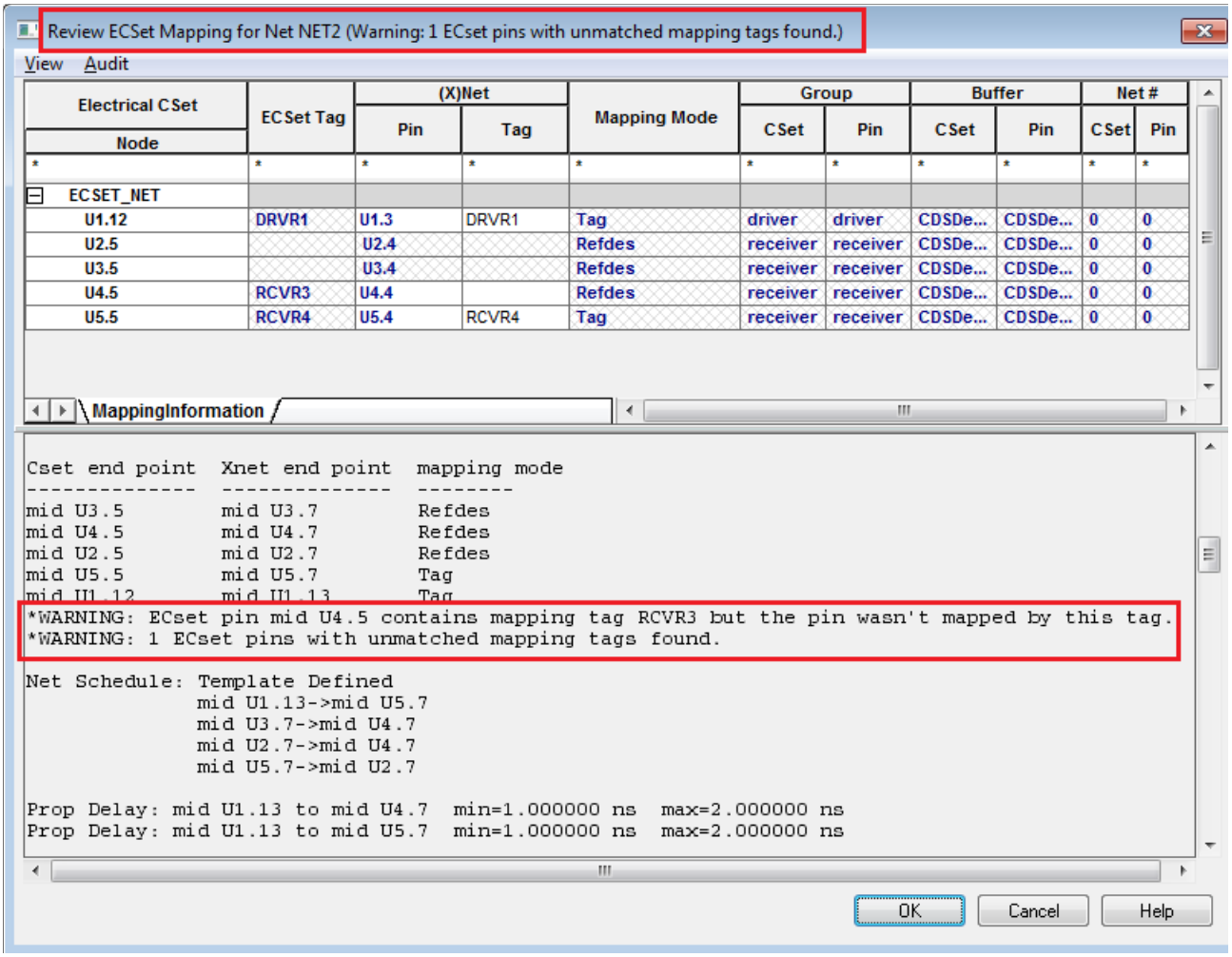


Running *Audit Electrical CSet* command on a single (X)net, launches an interactive dialog box to view the current mapping information.



In the *Review ECSet Mapping for <net_name>* dialog box, you can review the mapping between ECSet nodes and pins of a (X)net. If the tags are mismatched, warnings are

reported in the apply log. You can assign the tags in the ECSet to the appropriate pins of the (X)net.



For this example, the pins U1.3 and U5.4 of NET2 are successfully mapped to the ECSet nodes U1.12 and U5.5, respectively.

You can edit the mapped (X)net pins for those ECSet nodes which are tagged using one of the following ways:

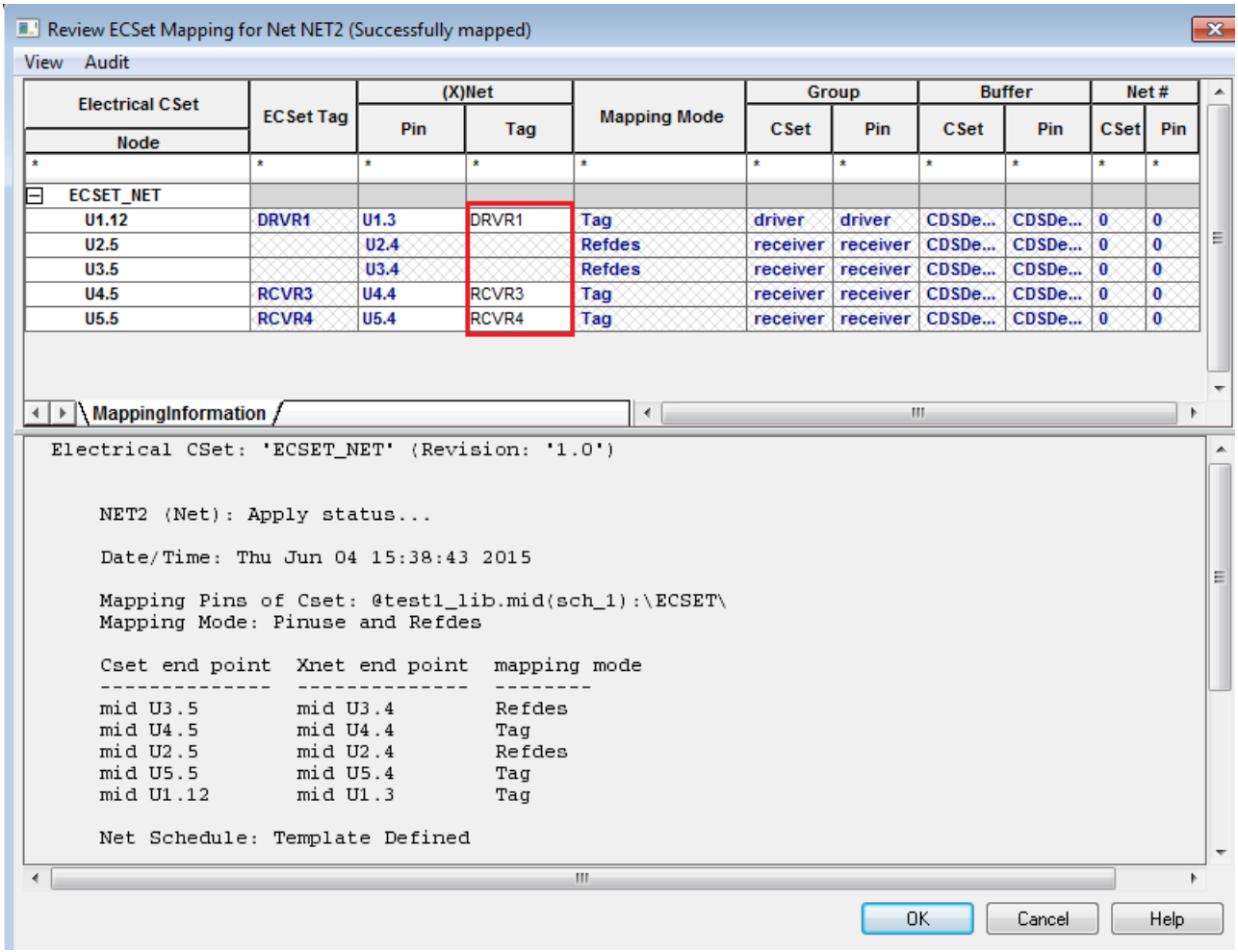
- Clear the currently mapped (X)net pins and then select the correct (X)net pin for each ECSet node.
- Edit the tag for the (X)net pin such that re-applying the ECSet maps the (X)net pin to the ECSet node with the same tag.

The tags on the pins of a (X)net are displayed in one of the three forms:

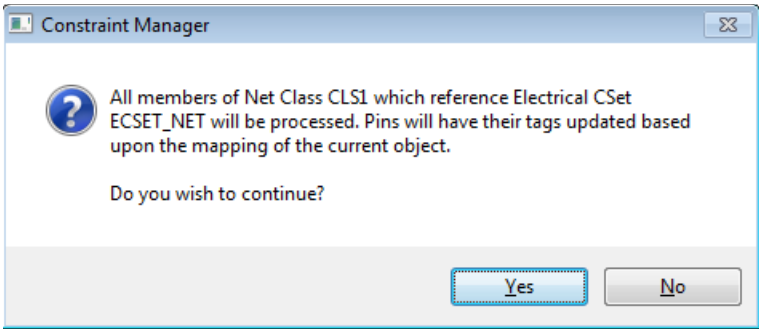
- Read-only (cross-hatched): When the tag is not defined for an ECSet node and there is no pin mapping. For example, the pins U2.4 and U3.4.
- Flattened (thin black): When the ECSet tag is applied to the (X)net pin or pin mapping is system-generated.
- Directly set (bold blue): When tag on a (X)net pin is explicitly set in the design or in the Pin tag column, or if pin mapping is user-generated.

After the mapping between ECSet nodes and (X)net pins is done, run the *Audit* command to apply the mapping. You can apply the mapping in two ways:

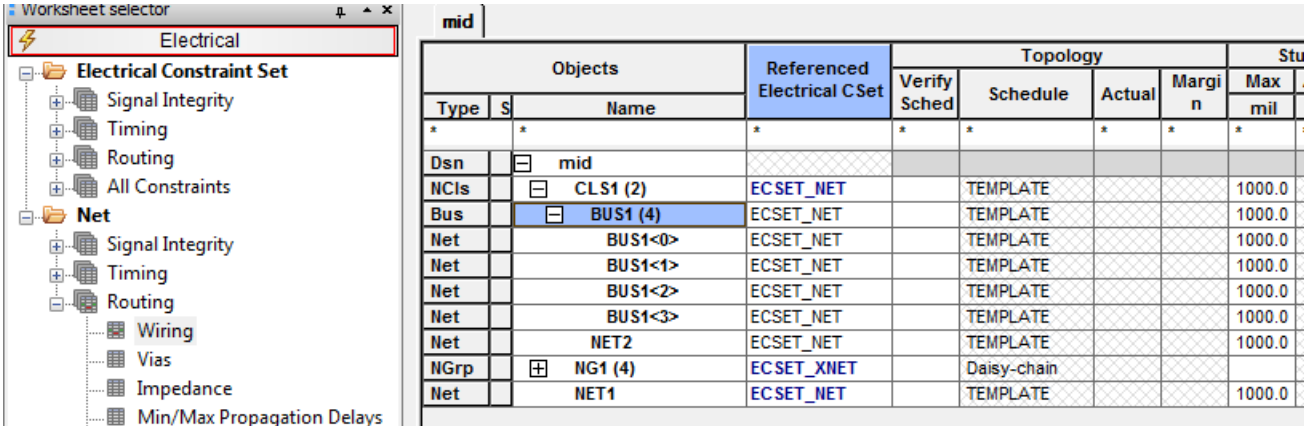
- *Apply tags to current object*: Applies the ECSet to the current (X)net. The ECSet tags are applied to those pins, which do not have tags. In the following figure, the RCVR3 tag is mapped to pin U4.4 of NET2:



- *Apply tags to all objects*: Applies the ECSet to all the net objects – within the current object's group – which are reference the same ECSet. A warning is displayed to confirm the action.



Constraint Manager applies the ECSet tags to all the (X)nets which reference this ECSet. In the following figure, the ECSet tags are applied to BUS1 that uses similar topology/ schedule/components as NET2.

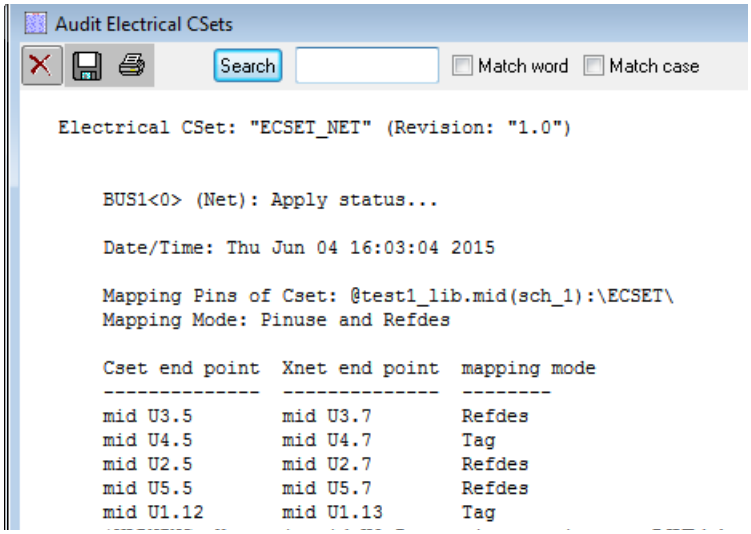


Note: It is recommended that you first *Apply tags to current object* to confirm the tag mapping for a single (X)net and then use *Apply tags to all objects.*

If an ECSet is referenced by multiple objects including individual (X)nets and group objects, the *Apply tags to all objects* command is only applied to the members of the selected object's group which reference the same ECSet.

For this example, NET2 is a member of BUS1 and CLS1. On running the *Apply tags to all objects* command, all members of CLS1 are processed as this group refrences an ECSet. Similarly, if BUS1 references an ECSet, only its members are processed.

If the mapping between ECSet tags and (X)net pins is unsuccessful, the *Audit Electrical CSets* report dialog is displayed.



Defining Tags

The tags on the components or pin instances are defined in the database(schematic or layout). The tags on an ECSet are defined in SigXplorer. Tags can be defined in a design prior to ECSet extraction or in SigXplorer when a (X)net is extracted.

Defining Tags on a Component/Pin in the design

You can add tags to a component or pins in a design prior to extracting a topology into SigXplorer. Tags can be set directly in the database(schematic or layout) by adding a property ECSET_MAPPING_TAG on the component or pins.

- In DE-HDL: You can add/edit the tag to the component or pin instance when Constraint Manager is closed.

  For more information, see Working with ECSet Tags in *Allegro Design Entry HDL - Constraint Manager User Guide*.

- In PCB Editor: You can add/edit the tag using the `property edit` command or in the *Pin Properties* worksheet of Constraint Manager.

- In System Connectivity Manager(SCM): You can add/edit the tag in the *Pin Properties* worksheet of Constraint Manager.

These tags are used when a (X)net is extracted to SigXplorer and when ECSet is applied to that (X)net.

Defining Tags on an ECSet in the SigXplorer

The package pin parameter, *mappingTags* is used in the SigXplorer topology file to uniquely identify a pin and thereby remove any ambiguity in the application of ECSets.

For more information, see *SigXplorer User Guide*.

Topology Templates Audit

The *Topology Templates* audit (*Audit – Topology Templates*) migrates deprecated properties to ECSet references used by Constraint Manager.

Note: Constraint Manager, when launched from an L Series PCB Editor, does not support topology templates.

In pre-14.0 designs, electrical constraints were captured using three entities: the topology template (specified with the TOPOLOGY_TEMPLATE property), the topology assignment (specified with the ASSIGN_TOPOLOGY property), and the constraint set (specified with the ELECTRICAL_CONSTRAINT_SET property).

In 14.0 designs, only the ECSet association is supported; the TOPOLOGY_TEMPLATE and the ASSIGN_TOPOLOGY properties are no longer required. All topology information is now contained in the ECSet (specified with the ELECTRICAL_CONSTRAINT_SET property).

The topology templates audit removes the TOPOLOGY_TEMPLATE, TOPOLOGY_TEMPLATE_REVISION, and ASSIGN_TOPOLOGY references from net-related objects.

The *Audit Topology Templates* dialog box contains the following fields:

Table 4-2 Audit Old Template Dialog Box Options

Use this field	To
Topology template values (drop-down menu)	List all TOPOLOGY_TEMPLATE and ASSIGN_TOPOLOGY property values in the design. Once a property value is selected from the drop-down menu, all nets which have the same template value are listed.
Update to use Electrical Cset	
Import (radio button)	Import a new template. The field beside the radio button will be populated with a Topology Template (.top file) name if one exists on disk. The `TOPOLOGY_TEMPLATE_PATH` environment variable is used to search for a template file with the same name as the property value
Browse	Find the appropriate template file if Constraint Manager cannot locate it.
Existing (radio button)	Use an existing ECSet. This option will be the default if the design contains an ECSet with the same name as the property value

Use this field	To
Overwrite existing constraints (check box)	Controls whether constraint values in the ECSet overwrite any existing net-related constraints when an ECSet is re-applied. *Important* Enabling *overwrite existing constraints* is necessary when migrating pre-14.0 designs. This ensures that all net-related overrides—created by the pre-14.0 topology template mapping software—are removed.
Apply	Migrate all listed nets to reference the imported or existing ECSet. Once migrated, nets will have their TOPOLOGY_TEMPLATE and ASSIGN_TOPOLOGY properties deleted. Constraint Manager displays a *properties up to date* message as appropriate.

Exporting ECSets

Constraint Manager promotes design reuse through the following commands:

Use this command	To
File – Export – Electrical CSets	Save selected ECSets, from a design, or from a system, to a topology template (`.top`) file on disk.
File – Export – Constraints	Export a dictionary and constraints file (`.dcf`) to disk. The dictionary and constraints file contains a complete snapshot of all electrical constraint information. This includes any user-defined properties, all ECSets and their constraints, and all net-related objects and their constraints (including ECSet references). The dictionary and constraints file is proprietary to Cadence Design Systems and, as such, is not available for editing. You would typically save to a dictionary and constraint file prior to making extensive constraint modifications within Constraint Manager. The dictionary and constraints file is then considered an archive from which you could revert back. Exporting a dictionary and constraints file results in overwriting constraint data saved from a previous archive.

Migrating Legacy Electrical Rule Sets

Designs created prior to release 14.0 may contain Electrical Rule Sets. As part of the process of upreving a design to release 14.0, these Electrical Rule Sets are migrated to Constraint Manager as Electrical Constraint Sets (ECSets).

5

Constraint Analysis

Topics in this chapter include

How Allegro Constraint Manager Performs Analysis

Allegro Constraint Manager analyzes the constraints in your design using two methods:

- Design Rule Checks

  Real-time design rule checks are made on objects constrained in the *Routing* worksheets. Results are returned to the worksheet cells in focus by comparing changes in the layout, such as moving a part, against the constraint limits that you specified for these objects.

  As design rule violations are encountered, Constraint Manager colors the corresponding worksheets cells in red. Additionally, bow tie markers appear on offending objects in the layout.

  See "Analyzing for DRC-based Constraints" on page 129 for information about interactive, online design rule checking.

- Simulated Analysis

  Simulated analysis is made on objects constrained in the *Signal Integrity* and *Timing* worksheets in the Electrical domain.

  Note: Constraint Manager, when launched from an L Series PCB Editor, does not support the *Signal Integrity* and *Timing* workbooks.

  Analyzed results are returned to the worksheet cells in focus by comparing computations (the actual) against the constraint limits that you specified for these objects. The actual, and the difference between the actual and the set constraint limit (the margin) are returned.

  See "Analyzing for Simulation-based Constraints" on page 131 for information about analyzed constraints.

Tip

The analysis engine computes a value (*actual*) and compares this to the value specified in the CSet. The difference between the analysis value and the specified constraint value is the *margin*. Both actuals and margins are returned to the cells in the appropriate worksheets.

Note: The same color scheme is used for both analyzed results and design rule checks.

You can click on an object in a Constraint Manager worksheet and choose *Object – Select* to highlight that object in the layout.

Viewing Worksheet Cells and Objects

As the complexity of your design increases, the number of objects in your design increases; and, correspondingly, the number of CSets associated with those objects increases. This can lead to a high-level of congestion in your worksheets. Fortunately, Constraint Manager lets you easily change your view of constraints, letting you change your focus as you work.

Table 5-1 Common worksheet commands

Task	Related Commands	Actions
Locating an object, a result, or a CSet	*Edit – Find*	Finds the specified object. You can filter on the following: ■ Match whole word only ■ Expand hierarchy You can click *Find Next* (or `F3`) to locate the next occurrence.
	Edit – Go to source	Locates the parent object that owns the inherited ECSet of the selected child object. For example, if you select a child object, such as a bit in a bus, its constraints are most-likely inherited from a CSet that is associated with the parent object, in this case the bus.
	View – Options – Row Numbers	Enable row numbering in the worksheets.
	Objects – Filter	Selectively display (or hide) the following objects in the worksheets: ■ net ■ Xnet ■ pin pair ■ results ■ differential pair ■ bus ■ match group ■ net class ■ net class-class ■ region ■ region class ■ region class-class

Table 5-1 Common worksheet commands

Task	Related Commands	Actions
Controlling the worksheet or object hierarchy	*Objects – Expand/ Collapse* (or use the [+] and [-] controls)	Expand or collapse the worksheet hierarchy in the worksheet selector or the object hierarchy in the worksheets. Worst-case analysis results on collapsed (hidden) objects are rolled up to the expanded object. This notification lets you work at any level in the object hierarchy.
	View – Show All Rows	Expand or collapse all rows in all worksheets.
Working in columns	*Column – Sort* (or double-click the column head)	Reverse the ordering of objects or constraint values in a column.
	View – Hide/Show Column	Hide columns in a worksheet so you can focus on a single, or a few, columns. Otherwise, you may have to scroll horizontally to access an out-of-view column.
	Resize	Resize column width. Grab a column border and drag.
Comparing cells	*Window – Tile*	Compare the cells of two or more different worksheets. You may have to scroll to view the desired cells.
	Window – New Window	Compare cells in the same worksheet. Constraint Manager opens the same worksheet in a different window. This lets you scroll to different cell views while allowing you to make concurrent edits in the same worksheet.

Analyzing for DRC-based Constraints

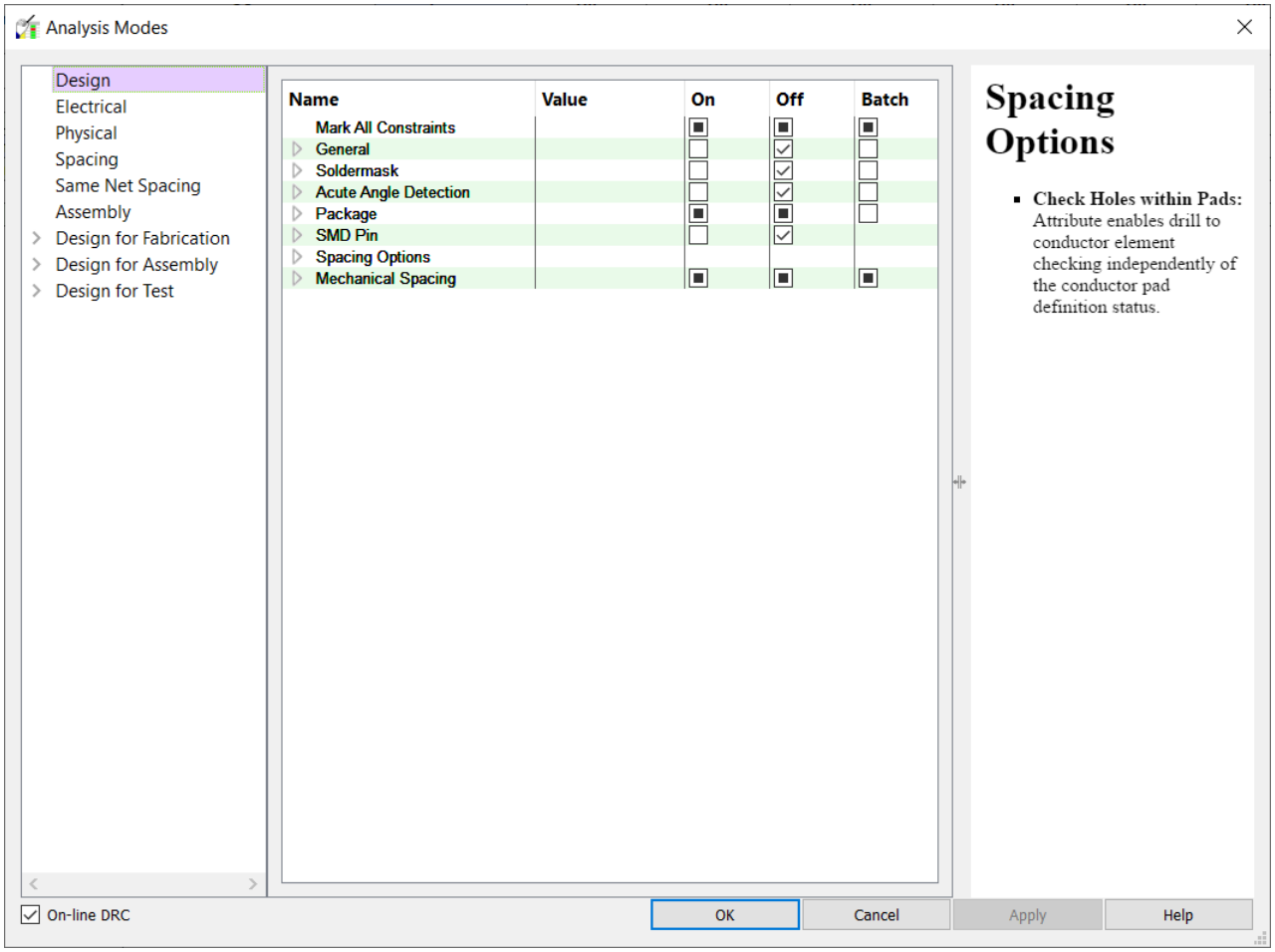
You control design rule checks with the domain tabs of the *Analysis Modes* dialog box (choose Analyze – Analysis Modes). Alternatively, you can specify analysis settings, DRC modes, and desired reports from a single dialog box (choose Tools– Report).

Note: Constraint Manager, when launched from an L Series PCB Editor, does not support custom measurements or custom stimulus; therefore, these tabs are not visible in the *Analysis Modes* dialog box. Furthermore, the *Max xtalk* and *Max peak xtalk* DRC fields in the Electrical Mode are hidden.

DRC Constraint Modes

Use the *Analyze – Analysis Mode* dialog box to control which design rule checks (DRC) to run. When the layout changes, an enabled design rule check is triggered.

Figure 5-1 Analysis Modes dialog box



Refer to the Analyze – Analysis Modes in the Allegro Constraint Manager Reference guide for information about how to use DRC constraint modes.

Analyzing for Simulation-based Constraints

Certain constraints in the Electrical Domain (*Signal Integrity* and *Timing*) require simulation to compute *actual* values. When the actual value is analyzed and returned to a worksheet cell, it is compared with the specified constraint value that is associated with the object being analyzed. The difference is calculated and displayed in the *Margin* column.

Important

To analyze for simulation-based constraints, Constraint Manager must be run with a PCB Editor or APD.

Before you initiate an analysis (*Analyze – Analyze*) on an object, you should configure the analysis engine (*Analyze – Settings*). Alternatively, you can specify analysis settings, DRC modes, and desired reports from a single dialog box (*Tools – Report*).

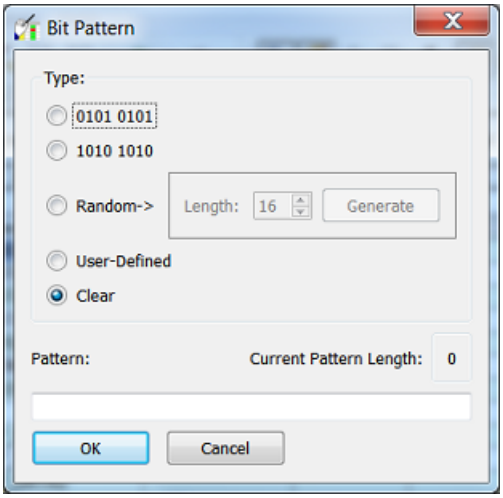
In the *Analysis Settings* dialog box (see "Analysis Settings" on page 133, you specify the type of simulation, whether to use crosstalk timing windows, and the type of stimulus.

You can click *Preferences* to specify buffer information (in the *Analysis Preferences* dialog box). You can also choose to save a waveform for each analysis; waveforms can subsequently be viewed in SigWave (choose *Tools – SigWave*).

Simulation-based Custom Stimulus

In addition to capturing custom stimulus in a SigXplorer topology file and importing it into Constraint Manager, you can create your own stimulus patterns directly in the *Electrical Properties* worksheet in the *Signal Integrity* workbook (Electrical Domain). Custom Stimulus is associated with Custom Measurements.

Type	Objects	Duty Cycle	Jitter	Cycle to Measure	Offset	Bit Pattern
		%	ps		ns	
Dsn	⊟ unnamed					
Bus	⊟ ADDRESS_BUS					0001
Net	AD0					0001
Net	AD1					0001



Double-click in the Bit Pattern cell to invoke the editor.

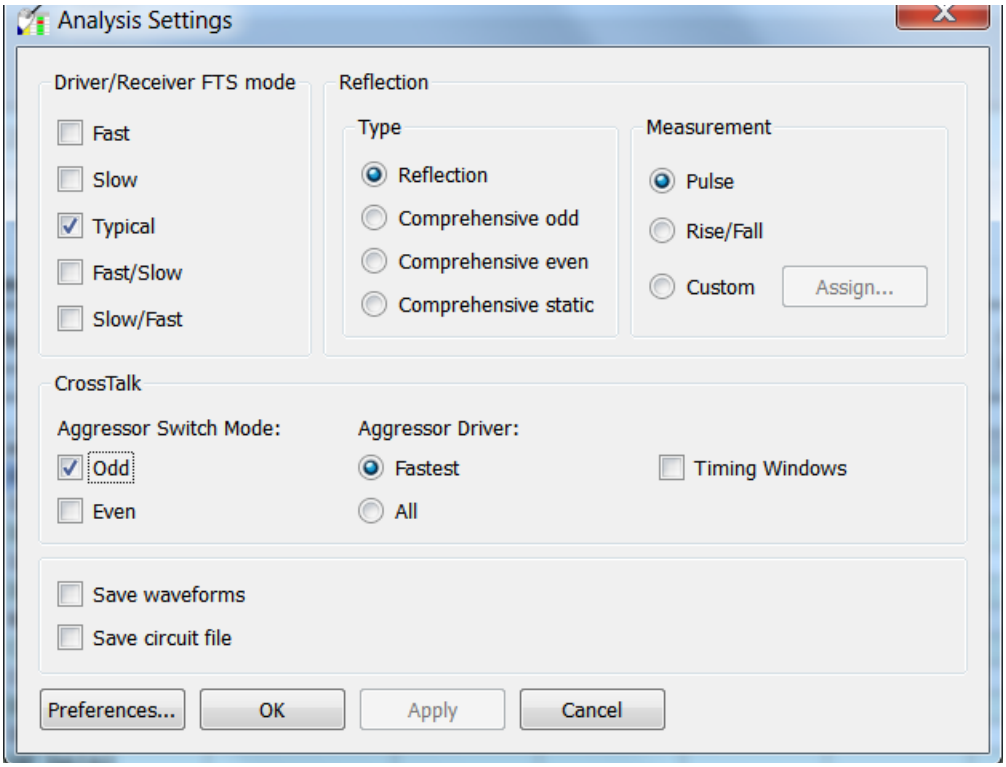
You can choose from 8-bit repeating patterns that begin with either high or low. You ca also specify a randomly-generated pattern or a pattern of your choice, up to 512 bits.

Note: Constraint Manager captures clock measurements in the PULSE_PARAM property. The Offset column converts all cell entries to nanoseconds.

To enable custom stimulus

Custom Stimulus exercises Custom Measurements. You enable Custom Stimulus through the (Analyze – Settings) dialog box. Enable the *Reflection*, *Custom*, and *Use custom stimulus* for custom measurements radio buttons, as shown in Figure 5-2 on page 133.

Figure 5-2 Analysis Settings



The Analysis Process

The following steps serve as a guideline (a checklist) of the steps involved in performing analysis in Constraint Manager. You may not need to perform all the steps all the time; it depends on where you use Constraint Manager in the design flow. For example, once you set DRC modes and analysis settings, you may decide to retain these settings for subsequent analysis.

Step 1
Creating Design Objects

You want to combine objects, where appropriate, into easily-managed object groupings. In this way, constraints can be set at different levels in the object hierarchy.

- Where appropriate, combine designs into systems.

  A system configuration database is advisable for maintaining system-level constraints and design objects.

- Where appropriate, combine nets and Xnets into buses and net classes.

- Where appropriate, combine nets or Xnets into differential pairs.

  For modeled-defined differential pairs, each member of the differential pair must have the appropriate signal model assignment.

- Where appropriate, combine nets, Xnets, and pin pairs into match groups when specifying relative propagation delays.

- Where appropriate, specify pin pair connections.

Step 2
Setting Constraints

Next, you create CSets based on your design requirements.

➤ Create an CSet in the appropriate worksheet.

This can be done (1) from scratch in the CSets object folder; (2), based on an existing net-related object in the Nets object folder; (3), by cloning an existing CSet; or (4), by importing a CSet.

Note: You can also select a net, extract it to SigXplorer, set constraints, update the topology in Constraint Manager (as an Electrical CSet), and apply the it to other objects.

Important

When specifying constraint parameters in a worksheet cell, it may be helpful to right-click and choose *Change* from the pop-up menu. This will guide you through the appropriate parameters and syntax that applies for the specific constraint type that the cell represents.

See "Working With Reusable Constraint Objects — CSets" on page 87 for information about creating and assigning CSets.

Step 3
Assigning Constraints

Next, you assign CSets to appropriate objects in your design. Child objects inherit the constraints from an CSet assigned to a parent object.

➤ Assign the CSet to a net-related object

-or-

Set a constraint directly on a net-related object.

If a CSet is already assigned to that object, the constraint change that you make will override the constraint value inherited from the CSet.

Assignments can be from the CSet or from a net-level object.

See "Working With Reusable Constraint Objects — CSets" on page 87 for information about creating and assigning CSets.

Step 4
Setting DRC Modes

Next, you specify how Constraint Manager performs design rule checks. You may want to make a trade-off between completeness and performance.

➤ Set the appropriate mode for design rule checking as described in Analyzing for DRC-based Constraints on page 129.

Step 5
Setting View Options

Next, you may want to change the way Constraint Manager presents data.

➤ Ensure that the *use color* checkbox is enabled (choose *View – Options*).

➤ Set the desired colors to use for results returned from analysis as described in "Viewing Worksheet Cells and Objects" on page 125

Step 6 Setting Analysis Parameters (Electrical)	Next, you set up simulation parameters for reflection and crosstalk analysis. ➤ Specify parameters (*Analyze – Settings*) that govern the analysis engine as described in "Analyzing for Simulation-based Constraints" on page 131. For an in-depth discussion of analysis parameters, see the Allegro® SI Simulation and Analysis Reference.
Step 7 Setting Report Parameters (Optional)	Next, you specify report types and what objects to include in the report. You typically will want a report when you want to analyze *many* objects; otherwise, it is more practical to interpret results returned to worksheet cells when you are concerned with only a *handful* of objects. ➤ Specify the reports that you want generated from simulation-based analysis (choose *Tools – Report*). In the *Report* dialog box, you identify the CSets, and net-related worksheets, to be included in the analysis results. You can limit the report to specific object types (bus, differential pair, Xnet, net, net class, Cset), and to a specific condition (any condition, only violations, only failures, only objects that are constrained). **Note:** Incidentally, from the *Report* dialog box, you can also specify DRC modes and analysis settings, and you can initiate simulation-based analysis. For an in-depth discussion of reports, measurements, and computations, see the Allegro® SI Simulation and Analysis Reference.

Step 8 Selecting an object	Next, you choose which objects to analyze. At this stage, some analysis is complete based on DRC settings. Worst-case results of child objects roll up to the respective parent object. ➤ Specify a net, Xnet, or object grouping to be analyzed. ➤ Once in view, click in a cell in the object column. You can select a range of cells using `Shift-Click` and non-contiguous cells by using `Cntrl-click`.
Step 9 Analyzing	Finally, you initiate the simulation(s). ➤ Choose *Analyze – Analyze* (or right-click and choose *Analyze* from the pop-up menu). *Important* As the analysis progresses, you can receive feedback by monitoring the status bar (located at the lower-left corner of Constraint Manager).

Analysis Results

Results returned from Analysis take four forms:

- Generated DRCs in the layout
- Waveforms
- Reports
- Calculated *actuals* and *margins* populated in the worksheets

Each is discussed in the sections that follow.

Generated DRC Output

Updated constraint information is communicated to the PCB Editor or APD. If a violation exists, a DRC bow tie marker is attached to the offending object in the layout.

Waveforms

Analysis results returned for certain constraints in the *Signal Integrity* and *Timing* worksheets yield waveform files. In Constraint Manager, choose *Tools – SigWave* to view these waveforms.

Reports

For each *enabled* net-related worksheet or CSet, a report is produced, `consmgr.rpt`, that lists constraint parameters, object assignments, and analysis results.

Worksheet cells

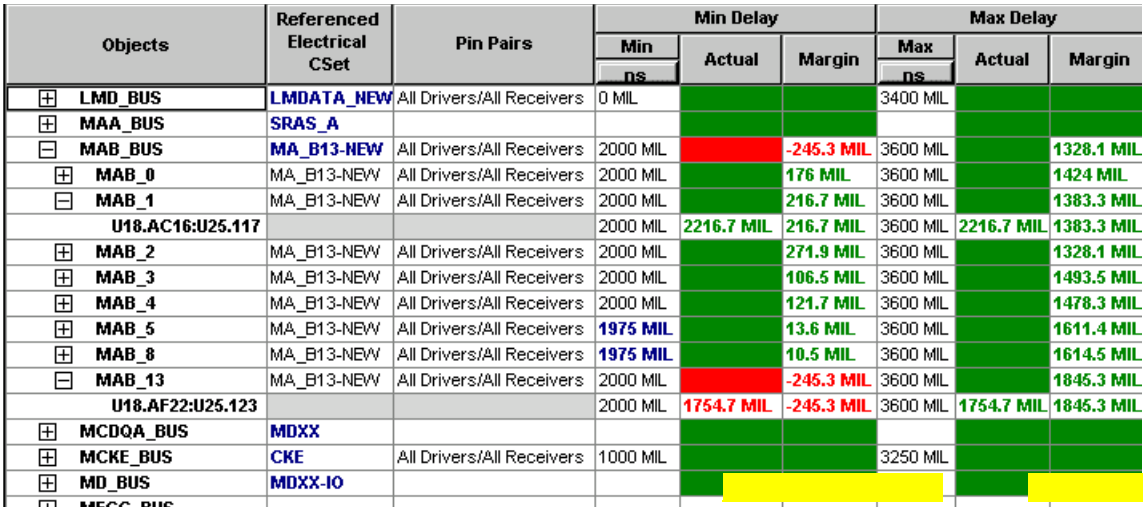
Analysis results returned to worksheet cells exhibit the following behavior.

- Cells give graphical feedback to reflect their status. See Viewing Worksheet Cells and Objects on page 125 for more information on default and user-defined colors. By default, the following color scheme is used for analysis:
  - Pass = green
  - Fail = red
  - Analysis error = yellow
  - Directly set = blue
- Cells that are grayed-out reflect that the cell is not applicable for the selected object.
- Cells will be colored blue if the cell contains a value which has been explicitly entered. This could happen when you override, for example, one bit of a bus object or when you specify a constraint directly on a net-related object rather than having that object inherit its constraint value from a referenced CSet.
- Cells which are populated and colored black reflect that the value is inherited from a higher-level cell or an CSet reference on the object. When you select an inherited cell, the status bar will indicate the source of the value. The source (the owner of the object) is reported as the object type and its name.

Interpreting Analysis Results Returned to a Worksheet

Figure 5-3 and Table 5-2 take you through a typical scenario of analyzing for propagation delay. Together, they explain how to interpret the analyzed results fed back to the worksheet.

Figure 5-3 Analyzing for Propagation Delay



Objects	Referenced Electrical CSet	Pin Pairs	Min Delay			Max Delay		
			Min ns	Actual	Margin	Max ns	Actual	Margin
⊞ LMD_BUS	LMDATA_NEW	All Drivers/All Receivers	0 MIL			3400 MIL		
⊞ MAA_BUS	SRAS_A							
⊟ MAB_BUS	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		-245.3 MIL	3600 MIL		1328.1 MIL
⊞ MAB_0	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		176 MIL	3600 MIL		1424 MIL
⊟ MAB_1	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		216.7 MIL	3600 MIL		1383.3 MIL
U18.AC16:U25.117			2000 MIL	2216.7 MIL	216.7 MIL	3600 MIL	2216.7 MIL	1383.3 MIL
⊞ MAB_2	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		271.9 MIL	3600 MIL		1328.1 MIL
⊞ MAB_3	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		106.5 MIL	3600 MIL		1493.5 MIL
⊞ MAB_4	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		121.7 MIL	3600 MIL		1478.3 MIL
⊞ MAB_5	MA_B13-NEW	All Drivers/All Receivers	1975 MIL		13.6 MIL	3600 MIL		1611.4 MIL
⊞ MAB_8	MA_B13-NEW	All Drivers/All Receivers	1975 MIL		10.5 MIL	3600 MIL		1614.5 MIL
⊟ MAB_13	MA_B13-NEW	All Drivers/All Receivers	2000 MIL		-245.3 MIL	3600 MIL		1845.3 MIL
U18.AF22:U25.123			2000 MIL	1754.7 MIL	-245.3 MIL	3600 MIL	1754.7 MIL	1845.3 MIL
⊞ MCDQA_BUS	MDXX							
⊞ MCKE_BUS	CKE	All Drivers/All Receivers	1000 MIL			3250 MIL		
⊞ MD_BUS	MDXX-IO							

Table 5-2 Dissecting the Analyzing for Propagation Delay figure on page 140

Object	Cell Column	Comments
LMD_BUS MAA_BUS MAB_BUS MCDQA_BUS MCKE_BUS MD_BUS	Referenced CSet	CSets are set directly on the bus-level object. The cell is rendered blue. Members of the bus inherit the constraint values set on the bus. This is evident in the individual nets under the expanded MAB_BUS. Inherited constraint values are rendered black.

Table 5-2 Dissecting the Analyzing for Propagation Delay figure on page 140

Object	Cell Column	Comments
LMD_BUS	Min Delay (Actual/Margins) Max Delay (Actual/Margins)	Cells are rendered green and do not contain values. This indicates that the last time the object (LMD_BUS) was analyzed, it was within the specified constraint values. For example, if the board was analyzed in an earlier design session, the analyzed values would not be saved with the board database. However, the last analyzed state of the object (*pass*, in this case) is communicated back to the cell in the form of a solid color. To populate the cells with integral values, you must re-run analysis (choose *Analyze – Analyze*). You could also import saved analysis results (choose *File – Import – Analysis results*).
MAB_5 MAB_8	Min Delay (Min)	Nets 5 and 8 of MAB_BUS have overrides. Because the overrides were specified explicitly in each cell, the cell is rendered blue. Notice that all other members of MAB_BUS inherit their values from the constraint specified at the bus level. Therefore, these cells are rendered black.
MCKE_BUS	Min Delay/ Max Delay	Analysis failed, rendering the cells yellow. This was caused by an unplaced component attached to a net member of this bus.
MAB_4	Min Delay (Actual/Margin) Max Delay (Actual/Margin)	Analysis passes, rendering the cells green. Notice that only the *Margin* column contains an integral value; the *Actual* is solid. This is because the net has several hidden pin pairs. Since the cell can contain only one value, the cell is rendered a solid color to represent a pass/fail condition.
MAB_1	Min Delay (Actual/Margin) Max Delay (Actual/Margin)	Analysis passes, rendering the cells green. Notice that both the *Margin* and the *Actual* columns contain integral values. This is because the net has been completely expanded (bus, to net, to pin pair).

Table 5-2 Dissecting the Analyzing for Propagation Delay figure on page 140

Object	Cell Column	Comments
MAB_BUS MAB_13	Min Delay (Actual/Margin)	Analysis is in violation, rendering the cells red. Notice that both the *Margin* and the *Actual* columns contain integral values at the pin pair level of the MAB_13 net. This is because the net has been completely expanded (bus, to net, to pin pair). Also, notice that the net object that owns the pin pair displays a solid red in the actual column and an integral value in the margin column. This is because the worst-case violation is rolled up to the object that owns it. In this example, there is only one pin pair so it was rolled up. Again, if there were more than one pin pair in violation, since the *Actual* cell can contain only one value, the cell would be rendered a solid color to represent a pass/fail condition. *Important* Worst-case constraint violations on child objects are rolled up the object hierarchy to the parent object. That is, pin pairs roll up to the parent net or Xnet, nets or Xnets roll up to the parent bus, and buses roll up to the parent design. In this way, you can work at any level in the object hierarchy and still be informed of a constraint violation on a lower-level object that is hidden. Finally, the same worst-case pin pair violation on the MAB_13 net is rolled up to the parent bus, MAB_BUS.

Constraints Across the System

A system configuration represents the electrical characterization of a system including all the participating designs, including interconnecting cables and connectors, as well as Xnets and pin pairs and their assigned CSets

Note: Constraint Manager, when launched from an L Series PCB Editor, does not support board-to-board constraints.

You can set a constraint directly on a system-Xnet in your layout or you can define the constraint in Constraint Manager as a CSet and then reference it to a system-Xnet.

See the Allegro SI Simulation and Analysis Reference for a thorough discussion of system-level designs.

6

Using Constraint Manager with Other Tools Across the Allegro Platform

Topics in this chapter include

Phases in the Design Flow

A typical PCB design flow contains the following phases:

- Exploration

  SigXplorer

- Capture

  Allegro Design Entry HDL, Allegro System Architect, System Connectivity Manager

- Floorplanning

  Allegro PCB SI

- Implementation

  PCB Editor, APD, SiP

Each phase in the design flow requires different tools. Constraint Manager provides a *common* environment for managing constraints across all tools in the design flow.

Not all phases in the design flow are mandatory. For example, a new design may be a derivative of a prior design. In this case, the *exploration* and *floorplanning* phases may not be needed.

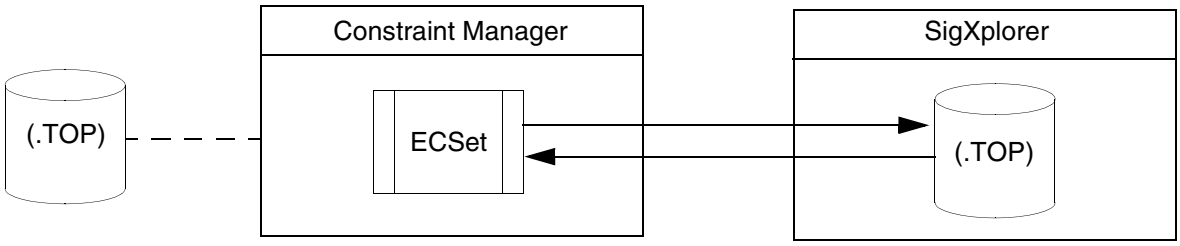
Constraint information in the board and in the schematic databases are synchronized using Design Sync. With Design Sync, you can specify that all constraints be synchronized or only those that have changed.

The sections that follow describe how to use Constraint Manager with other tools at each phase in the design flow.

Design Exploration Phase (with SigXplorer)

In the exploration phase, you focus on up-front exploration before the board is placed and routed. Board cross-section and material type are usually *not* known, although you can make assumptions from past designs. A netlist is *not* available in this phase of the design flow.

Use SigXplorer to perform simulations based upon the Electrical CSets characteristics (pins, scheduling, models). A unique topology template can be saved for each point explored in the solution space. The end result of the exploration phase is to create a library of Electrical CSets (`.top` files on disk) which would then be imported back into Constraint Manager where they could be swapped with other Electrical CSets, or where individual constraints could be moved between Electrical CSets.



In the exploration phase, you have a choice. Constraints can be proven in SigXplorer and saved as topology templates or they can be defined in Constraint Manager. The primary difference is presentation: SigXplorer is form-based; Constraint Manager is worksheet-based and perhaps it is easier to use for viewing and manipulating multiple constraint definitions.

Note: You will notice that only the *Electrical Constraint Set* folder is shown in Constraint Manager's worksheet selector. Absent of a board database, and a netlist, Constraint Manager does not show the *Nets* folder. For the same reason, *Physical*, *Spacing*, and *Same Net Spacing* worksheets do not appear. See the Workbooks and Worksheets figure on page 26 for information about the Nets folder.

Also, without a database with which to save constraint data, before exiting SigXplorer, or Constraint Manager, you must ensure that you save to a topology template to preserve your work.

Tip

You also use SigXplorer in the Constraint Manager flow to define custom measurements and custom stimulus. See Analyzing for DRC-based Constraints on page 129 for more information.

In SigXplorer, you simulate and analyze the topology. The following can be captured in a topology template:

- pin ordering (topology scheduling)
- termination strategy (and location on net)
- electrical constraints
- custom measurements and constrained custom measurements
- custom stimulus

Once exploration is complete, you save this information as a topology template (a `.top` file). This file represents the SigXplorer database. Constraint Manager is later used to import this information as an Electrical CSet.

Pin Scheduling

In Constraint Manager, you can select from the following pre-defined pin scheduling topologies:

- *minimum spanning tree*
- *star*
- *daisy chain*
- *source load daisy chain*
- *far-end cluster*

You select these from the *Wiring* worksheet of the *Routing* workbook. If you want to define your own pin schedules, you must manually wire the connections in SigXplorer and then export this information back to Constraint Manager (choose *File – Update Constraint Manager*).

Design Capture Phase

For information on using Constraint Manager in the Design Capture Phase, refer to the

- Allegro Design Entry HDL – Constraint Manager User Guide
- System Connectivity Manager – Constraint Manager User Guide
- Capturing Design Constraints chapter in the *Allegro Front-to-Back User Guide.*

Design Capture Phase (with System Connectivity Manager)

In the design capture phase, System Connectivity Manager provides a spreadsheet-based design environment. The spreadsheet-based interface is very effective while creating connectivity for designs with large pin count devices. While working with System Connectivity Manager, you use Constraint Manager to capture design constraints.

Tip

Enabling the *Transfer to/from Physical* button in the *Create Attribute Definition* dialog box ensures that user-defined properties flow between the logical and physical tools.

For more information, refer to . . .

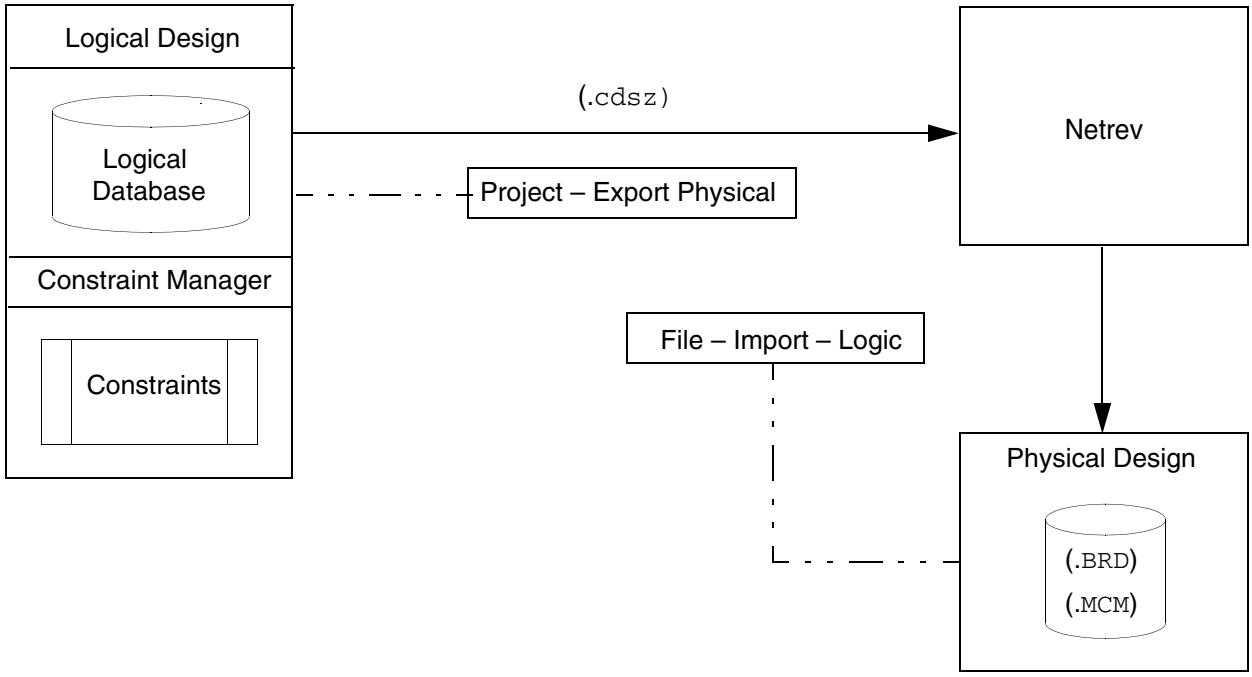
- Front to Back Constraint Flow on page 150.
- Back to Front Constraint Flow on page 152.
- Working with Properties and Electrical Constraints in the *System Connectivity Manager User Guide* for detailed information about how to capture design constraints while creating a design in System Connectivity Manager.
- Working with Properties in the *System Connectivity Manager User Guide* if you are using Constraint Manager as the property editor for System Connectivity Manager.

Front to Back Constraint Flow

To create a physical layout for the design in System Connectivity Manager, the design data and constraint information is exported to the physical database of PCB Editor, using *Project – Export Physical*.

Export Physical extracts five package files — which communicate logic, part, pin, reference designator, and constraint information — and writes to the (`.cdsz`) file. This file is then used by Netrev for back-end processing, as illustrated in Figure 6-1. Refer to Table 6-1 on page 151 for an explanation of package files (`pst*.dat`) used in the front to back flow.

Figure 6-1 Constraint Manager in Feed Forward Mode



The logical tools pass electrical, physical, and spacing constraint modifications (and net classes) to the physical design tools (based on mode) for layers that exist in the physical database, or a new, empty physical database seeded from information passed from a logical database.

Tip

See the Allegro Design Entry HDL - Constraint Manager User Guide for more information on *Overwrite Mode* and *Change Only* mode.

In the front-to-back flow, Design Editor generates the composite file, `pstdedb.cdsz`, which contains the following package files:

Table 6-1 Package Files

This file . . .	Contains . . .
`pstchip.dat`	Physical information for each type of symbol read from the `chips.prt` files and the physical parts tables, including electrical characteristics, such as pin direction and loading, logical to physical pin mapping, and voltage requirements. It defines the number of gates in each device, including gate and pin swapping information. This file also contains the name of the package and part symbol used to represent the device type in the physical layout (`JEDEC_TYPE`). **Note:** Device files are the third party equivalent of this file.
`pstxnet.dat`	A netlist that uses keywords (`net_name, node_name`) to specify the reference designators and pin numbers associated with each net. Constraints added to nets using Constraint Manager are written to the `pstcmdb.dat` file.
`pstxprt.dat`	Contains each physical package or part in the logic design along with its reference designator and device type. For packages or parts composed of multiple logic gates, the file identifies which gate was placed in which section of the package or part. Also contains attributes for parts and functions, and pin attributes specifically used for packaging. All other Pin attributes (constraints) are written to the `pstcmdb.dat` file.
`pstcmdb.dat`	Constraint and property information for the design.

Back to Front Constraint Flow

Import Physical is a prescribed set of processes that reconciles design data and constraints between physical- and logical databases.

The physical tools pass constraints to the logical database (based on mode). Physical and Spacing constraints, objects, and layers also make the transition to reconcile both databases.

Tip

See the Allegro Design Entry HDL - Constraint Manager User Guide for more information on *Overwrite Mode* and *Change Only* mode.

Import Physical calls Genfeed to extract six view files — which communicate component, part, function, pin, and constraint information and passes this file for front-end processing, as illustrated in Figure 6-2. Refer to Table 6-2 for an explanation of view files (`*view.dat`) used in the back-to-front flow.

Figure 6-2 Constraint Manager in Feedback Mode

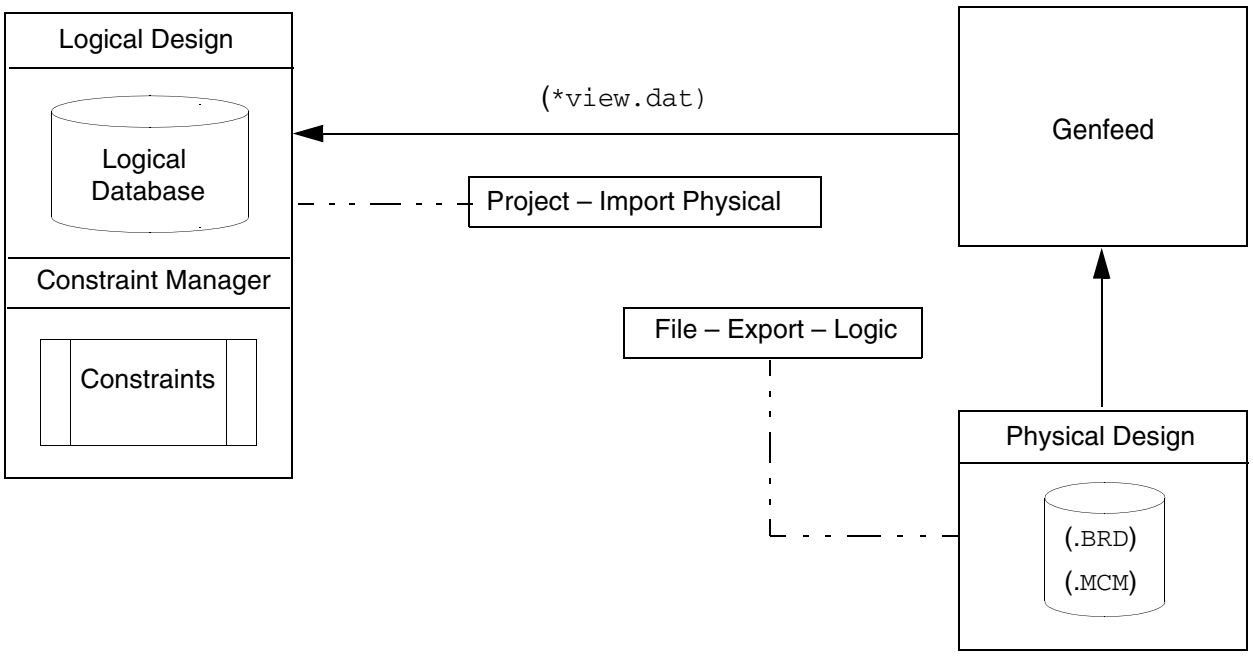


Table 6-2 View Files

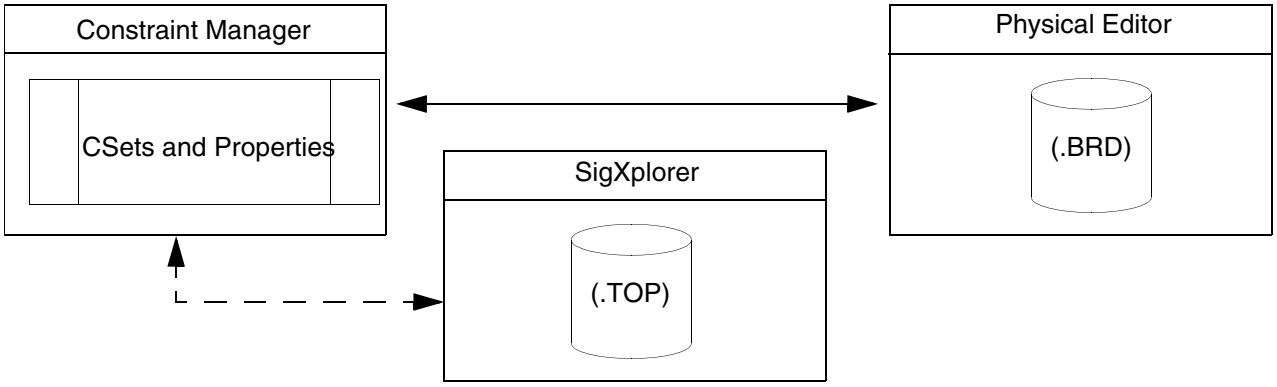
This file . . .	Contains . . .
`compview.dat`	Component information and properties, as defined in the physical tools.
`funcview.dat`	Function information and properties, as defined in the physical tools.
`netview.dat`	Connectivity information, as defined in the physical tools.
`pinview.dat`	Pin information and package-related properties, as defined in the physical tools.

Design Floorplanning and Implementation Phases

In the floorplanning and implementation phases, you focus on placement, routing, and manufacturing output. This section focuses on using Constraint Manager with back-end tools. For information on the front-to-back flow, you should also refer to Design Capture Phase on page 149.

Note: Constraint Manager, when launched from a Series L PCB Editor, does not support topology exploration with SigXplorer.

Constraint Manager is used along with your physical editor to manage constraints. Constraint creation or modifications in Constraint Manager will automatically be synchronized with the board (`.brd`) database.



You can also use SigXplorer to perform simulations based upon the Electrical CSet's characteristics (pins, scheduling, models) of the net-related objects in your design. See Using SigXplorer in the capture, floorplanning and implementation phases on page 156 for information on using SigXplorer.

- Launch Constraint Manager from your physical editor (choose *Setup – Constraints – Constraint Manager*).
- Launch SigXplorer from Constraint Manager by selecting a net-related object (or an Electrical CSet) and choosing *Tools – SigXplorer*. You can also right-click and choose *SigXplorer* from the pop-up menu.

In the floorplanning and implementation phases, you use Constraint Manager to:

- Import reusable topology templates from SigXplorer. These map to Electrical CSets in Constraint Manager.

  See Importing ECSets for information on using reusing topology template files from SigXplorer.

- Consolidate individual Nets and Xnets into more easily-managed units such as buses and match groups.

  See Working with Constraint Objects for information on buses and match groups.

- Define bus, differential pair, net, Xnet, or pin pair constraints.

  See Working with Constraint Objects for information on objects in Constraint Manager.

- Define differential pairs.

  See Differential Pairs.

- Define net-related constraint overrides, as appropriate.

  See Methods of Constraining Nets for information on overriding a constraint.

- Create CSets based on net-related objects such as buses, differential pairs, nets, and Xnets.

- Explore net topologies and schedule pins.

  See Pin Scheduling for information on pre-defined pin schedules.

- Audit CSets to resolve inconsistencies.

  See Audits for information on constraints and their assignments.

- Validate the design through design rule checks and analysis.

  See Using Constraint Manager with Other Tools Across the Allegro Platform for more information on validating constraints.

- Communicate layout changes to logical tools.

  See Design Capture Phase for more information on the front-to-back constraint flow.

- Open a partitioned design. Sections of the design that are partitioned are not editable, and open in Constraint Manager in read-only mode as indicated by cross-hatch shaded cells. You can analyze a partitioned design in Constraint Manager, but you cannot import constraints.

  See Objects Filter in the *Constraint Manager Reference* for information on filtering partitioned designs.

  See Design Partitioning in the *Allegro X PCB and Package User Guide* for information on setting design partitions.

- Migrate constraint sets, from older, pre-14.0, databases into Electrical CSets used in Constraint Manager.

  See Topology Templates Audit on page 118 for instructions.

Using SigXplorer in the capture, floorplanning and implementation phases

Note: Constraint Manager, when launched from a Series L PCB Editor, or a logical editor, does not support topology exploration with SigXplorer and database synchronization.

Unlike in the exploration phase, where there is no board or netlist available, SigXplorer employs a different use model in the floorplanning and implementation phases.

Use SigXplorer to extract a net (or a net-related object such as a bus, differential pair, or match group) for topology exploration and constraint modification. The extraction can be routed (a trace in the PCB) or unrouted (a ratsnest). Used in this way, SigXplorer is aware of the electrical and physical characteristics of the net.

You also use SigXplorer in the Constraint Manager flow to define custom measurements and custom stimulus. See Analyzing for DRC-based Constraints on page 129 for more information.

➤ To extract a net-related object, select a net in the worksheet, then right-click and choose *SigXplorer* from the pop-up menu

SigXplorer will:

- Extract all electrical constraint and topology information from the selected object.

  If the

  - *Use Include Routed Interconnect* box is checked (choose *Tools – Options* in Constraint Manager), interconnect details (clines and vias) will be included.

Important

You cannot update Constraint Manager with a topology that contains traces or vias. You must condition the topology for Constraint Manager by choosing *Edit – Transform – for Constraint Manager* in SigXplorer.

  - *Schedule Based on Routed Interconnect* box is checked (*Tools – Options* in Constraint Manager), the extraction derives connections from the user-defined net schedule (or from the default net schedule if none is specified). SigXplorer derives propagation delay and impedance from traces. Any unrouted segment derives its impedance and propagation velocity from the default settings.
  - Selected object is a Bus or Differential Pair, the template will include information from the *first* Xnet or Net.

Refer to the Tools – Options command in the *Constraint Manager Reference* for more information on topology extraction.

- Display the appropriate ratsnest based upon the chosen topology schedule:
  - for pre-defined scheduling, this choice is the value of the RATSNEST_SCHEDULE property. See Pin Scheduling on page 148 for information on pre-defined pin schedules.
  - for user-defined scheduling, this choice is the value of the TEMPLATE property. See the online help for instructions on scheduling topologies.

7

Customizing Constraint Manager Your Way

Topics in this chapter include:

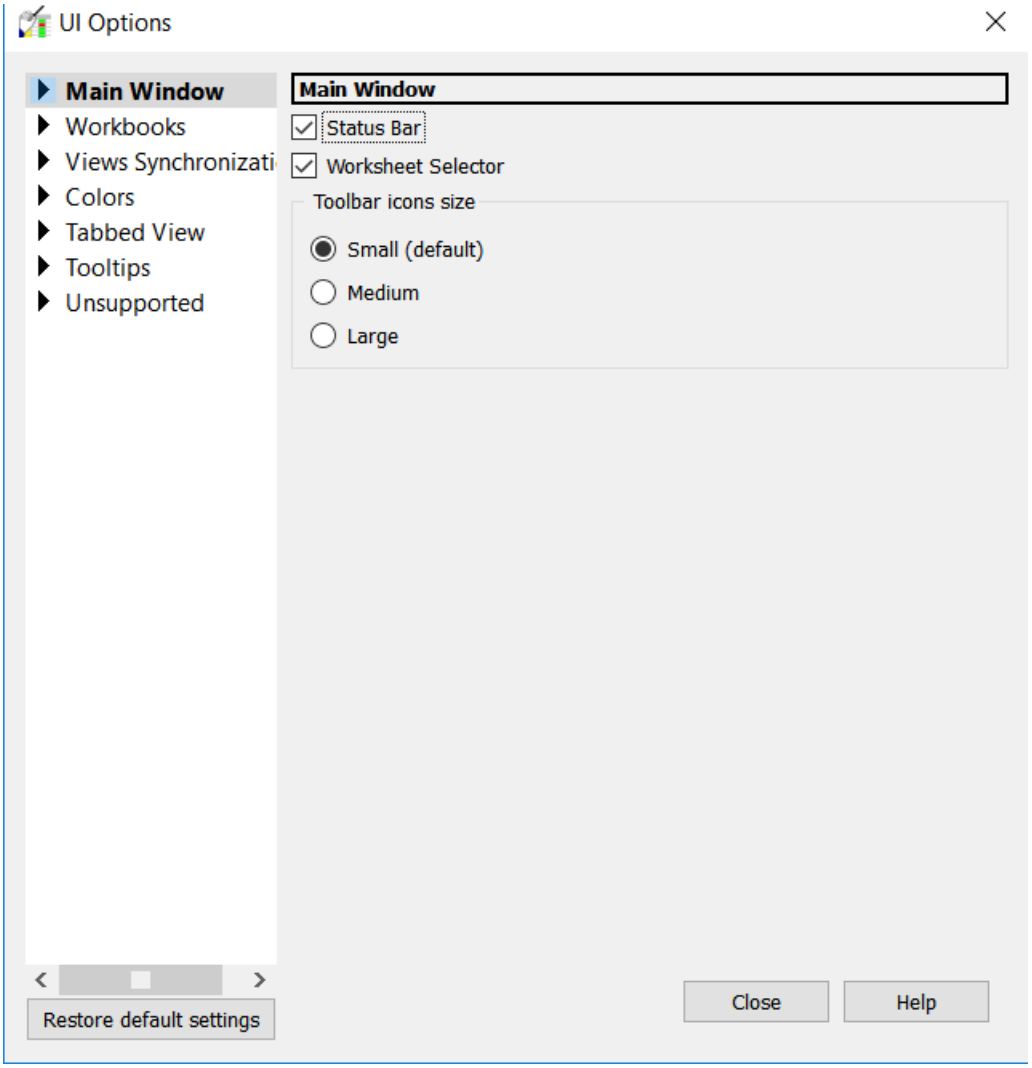
Customizing the User Interface

Constraint Manager gives you control of its user interface, including visibility options, user-definable keyboard shortcuts and user-definable toolbars.

Customizing Visibility

Constraint Manager affords you many options (choose View – Options) to personalize your view of the user interface.

Figure 7-1 Setting View Options



For information on field descriptions, see the View – Options command in the *Allegro Constraint Manager Reference*.

Customizing Toolbars

You can undock the toolbar (or a section of the toolbar) and reposition it in your workspace. You can also rearrange command icons within the toolbar strip and you can create your own toolbar, drawing from existing *File*, *Edit*, *View*, and *Filter* commands.

Customizing Documentation for User-Defined Attributes

The *Information* command available from the pop-up menu on a column header, including a *user-defined attribute (constraint)*, in Constraint Manager is used to access the associated help information for the selected column. This information describes the purpose of the property or constraint.

Note: A grayed out *information* menu command indicates that no documentation exists for the user-defined attribute.

You can create documentation for a user-defined attribute (column) and make it accessible using the following steps:

1. Create documentation on the user-defined attribute.

   You can capture user-defined documentation in a file with the same name as the attribute. The following file formats are supported:

   - `.html`
   - `.bmp`
   - `.txt`

2. Save the file as `<attribute_name>.<html>/<bmp>/<txt>`.

   For example, for an attribute named `NEW_COL`, the help file name will be `NEW_COL.html`.

3. Save the file in one of the following locations:

   - current directory - along with the `.brd` file
   - `<install_directory>`*/share/pcb/consmgr*
   - *CDS_SITE/cdssetup/consmgr*, provided the `CDS_SITE` environment variable is configured

4. In Constraint Manager, right-click the column header of the user-defined attribute and choose the *Help* command from the pop-up menu.

   The associated help file is displayed on top of the Constraint Manager application.

Customizing Worksheets

You can add user-defined or pre-defined attributes to Constraint Manager's default worksheets, and you can create your own customized workbooks and worksheets.

Important

Historically, PCB Editor uses the term *Property*, Constraint Manager uses the term *Constraint*, and Design Entry HDL uses the term *Attribute*. This chapter uses the term Attribute throughout to describe a Constraint, Property, or Attribute, which is validated or not.

Each net-level customized worksheet that you add has an *Objects* column and a *Referenced CSet* column (you can hide the latter). Customized worksheets do not contain *Actual* and *Margin* columns.

Each attribute that you add to a worksheet requires a new column. New columns appear to the right of the active worksheet. As with overrides, Constraint Manager renders customized workbooks, worksheets, and columns with a blue tint in the *Worksheet Selector*.

You can add, rename, and delete workbooks, worksheets, column headers and columns, and you can control their visibility. You can also export worksheets that you define, and you can import customized worksheets from a different design. Although you can hide columns of default worksheets, you cannot delete them. Also, you cannot delete predefined worksheets or predefined attributes.

Customize Mode

In addition to a *Browse* mode, Constraint Manager has a *Customize* mode, which uses special icons in the *Worksheet Selector* to assist you in differentiating among predefined workbooks, worksheets, and columns from those that you add in *Customize* mode.

Tip

See the Tools – Customize Worksheet command in the *Constraint Manager Reference* for detailed information about *Customize* mode, including adding columns, adding workbooks, adding worksheets, and drag-and-drop support for column positioning.

Note: You can also access *Customize* mode by right-clicking in the *Worksheet Selector* and choosing *Customize Worksheet*.

Figure 7-2 on page 165 depicts the expanded *Impedance* worksheet of the *Routing* workbook. Notice that default columns have a neutral, grayish tint except for the hidden *Referenced CSet* column, which is represented by the silhouette of a circle. Also note the column's superheader label (Single-line Impedance) depicted with a rectangle over a circle.

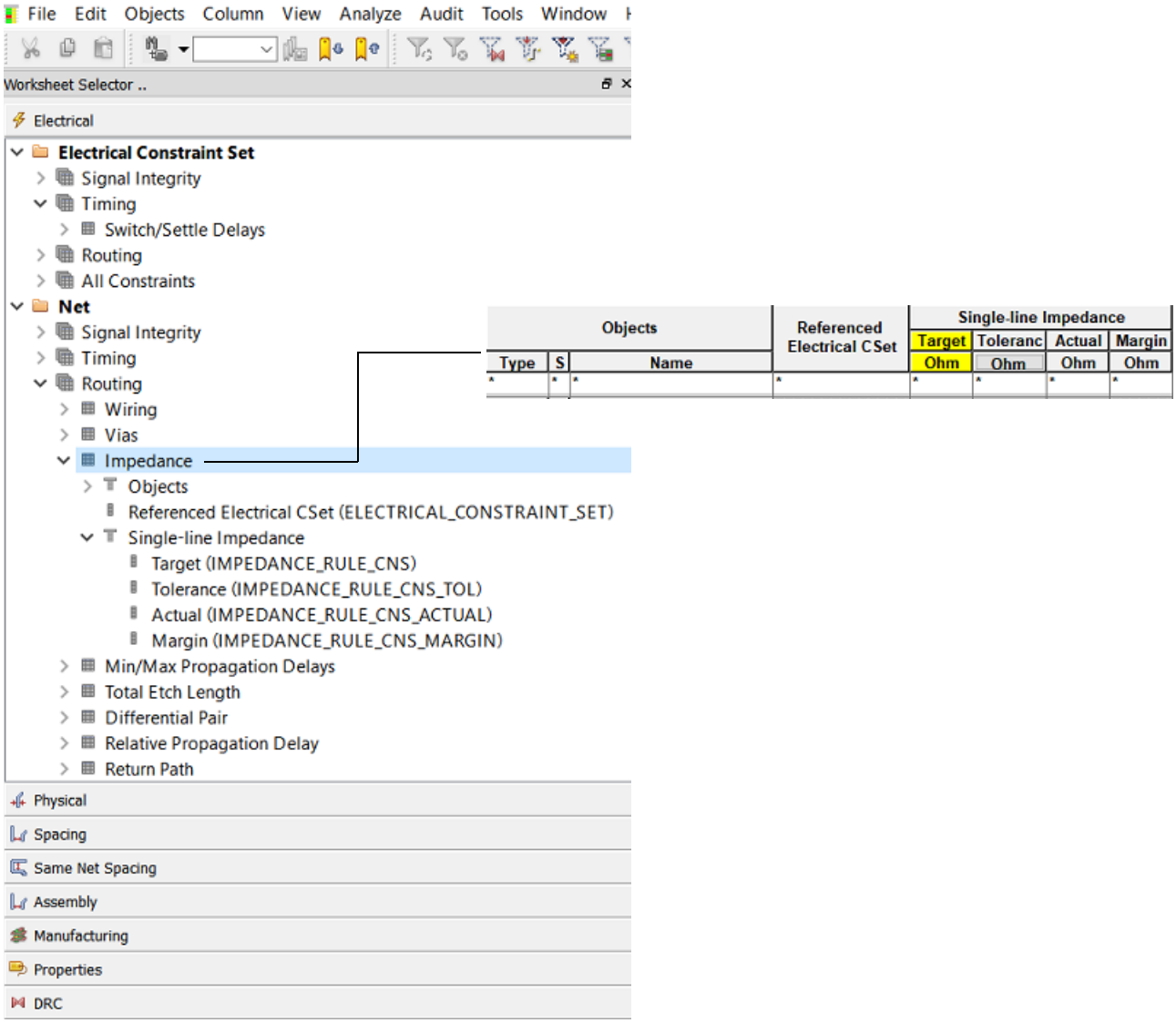
In the same figure, notice the *My Impedance* worksheet of the *My Routing* workbook. The workbook, worksheet, and all columns have a bluish tint. Also note the column's superheader (*My Single Line Impedance*).

The columns in the *My Single Line Impedance* worksheet are the same as those in the *Single Line Impedance* worksheet except for the difference in column labels: *Which Net* substitutes for *Target* and *Allowable Deviation* substitutes for *Tolerance*. Also, the *Electromotive Force* column label identifies the added column for the *Voltage* attribute.

Tip

A column superheader icon appears as a rectangle over a circle; a column icon appears as a circle. A predefined column icon is gray; a customized column icon is blue; a hidden column icon is a silhouette of the column icon.

Figure 7-2 Worksheet Selector icons in Customize mode



User-defined Properties

You use a user-defined property to capture a characteristic of an object. Constraint Manager does not perform design rule checks or analysis on this property; it facilitates communication of the design intent to down-stream tools with which you may want to use to manipulate the objects associated with these properties.

You create user-defined properties in the PCB- or Package-Editor using the *Setup – Property Definitions* command, or in SigXplorer using the *Set – Constraints – User-Defined* command.

Note: You can also create a validated user-defined property directly in Constraint Manager. See the Constrained Custom Measurements on page 169.

For Constraint Manager to report or display a user-defined property, you must add a column in any net-related worksheet and choose the desired property from a dialog box. Constraint Manager then adds a column to the far right of that worksheet with the property name as its column label. There are no *Actual* and *Margin* cells associated with user-defined properties. Additionally, the property appears under the *Electrical CSet* folder, in the *User-Defined* worksheet (and in the *Signal Integrity/Timing/Routing* worksheet) of the *All Constraints* workbook (see Figure 7-3 on page 171).

To add a column, enable customization with the Tools – Customize Worksheet command, then Right-click and choose *Add Column* from the pop-up menu.

Customizing Simulations

Constraint Manager supports custom constraints, custom measurements, and custom stimulus when used with the SigXplorer legacy flow.

Note: Constraint Manager, when launched from a Series L PCB Editor or OrCAD PCB Editor, does not support custom constraints, custom measurements, or custom stimulus. The *Custom Measurements* tab in the *Analysis Modes* dialog box (choose *Analyze – Analysis Modes*) as well as the *Custom Measurements* workbook is hidden.

Without Constraint Manager, you would have to extract a net from a board layout, define any custom constraints, custom measurements, or custom stimulus in SigXplorer, then re-apply the changes back to the net in the board layout. Because you have to do this one net at a time, this can be error prone and tedious. Because Constraint Manager has a global view of all nets in a board layout, application of custom constraints, custom measurements and custom stimulus is simplified.

You do not define custom constraints, custom measurements or custom stimulus in Constraint Manager, you only assign, manage, and analyze them. You define them in SigXplorer, save them as a topology file, and import them into Constraint Manager as an *Electrical CSet* (Choose *File – Import Electrical CSet)* or refresh the current *Electrical CSet* reference *(Choose Tools – Update Topology)*. Any net-related object that references that *Electrical CSet* will inherit any custom constraint, custom measurement or custom stimulus data that was captured in that *Electrical CSet*.

Tip

By default, custom measurements are not included with an imported *Electrical CSet*. To override this behavior, you must enable the *Update existing or create new Custom Measurement worksheet* option. See the File – Import – Electrical CSet command in the *Constraint Manager Reference*.

In this way, an *Electrical CSet* performs triple-duty. Not only does an *Electrical CSet* contain pre-defined and custom constraints, it can also contain custom measurements and custom stimulus.

Tip

Because you can assign only a single *Electrical CSet* to a net-level object, you must define any constraint data along with custom measurement and custom stimulus in that same *Electrical CSet*.

Working with Custom Constraints

In addition to its broad set of pre-defined constraints, Constraint Manager supports constrained custom measurements.

Constrained Custom Measurements

You use constrained custom measurements (and custom stimulus) to specify your own constraints. These constraints differ from user-defined properties in that Constraint Manager can validate them through design rule checks and analysis.

You create constrained custom measurements in SigXplorer in the same way as you define unconstrained custom measurements, using the measurements expression editor. When you choose *None* as the constraint type, SigXplorer creates an unconstrained custom measurement. When you choose *minimum*, *maximum*, *min-max*, or *target: tolerance* as the constraint type, SigXplorer creates a constrained custom measurement, which is, in effect, a user-defined constraint.

Tip

To invoke SigXplorer's *Measurement Expression Editor*: (1) click the *Measurements* tab; (2) right-click on the *Custom* tab; and (3) select *Add*.

Constrained custom measurements from SigXplorer appear under the *Electrical CSet* folder, in the *User-Defined* worksheet of the *All Constraints* workbook, as well as under the *Net* folder in the *Custom Measurements* workbook (see Figure 7-3 on page 171). *Actual* and *Margin* cells associated with constraint also appear.

Use Model

Typically, you will select a net object in Constraint Manager, right-click, then choose *SigXplorer* from the pop-up menu. The net's topology will then appear in SigXplorer where you can define the necessary custom constraints, or custom measurements and custom stimulus.

Tip

Consult the *Allegro SI SigXplorer User Guide* for information about how to define custom constraints, custom measurements, and custom stimulus.

Next, you choose *File – Update Constraint Manager* to export the topology file from SigXplorer. The corresponding *Electrical CSet* is refreshed in Constraint Manager and all net-related objects that reference the *Electrical CSet* will inherit the custom constraints or custom measurements and custom stimulus that you just defined, along with any electrical constraints that were captured in the *Electrical CSet* before they were exported from Constraint Manager to SigXplorer.

Note: If the object that you extracted from Constraint Manager into SigXplorer does not reference an *Electrical CSet*, when you choose *File – Update* Constraint Manager from SigXplorer, the topology file is imported into Constraint Manager as an *Electrical CSet* with the same name as the extracted object, unless you chose *File – Save As*. You can then associate that *Electrical CSet* with other net objects (choose *Objects – Electrical CSet Reference*), and rename the *Electrical CSet* if you want to (choose *Objects – Rename*).

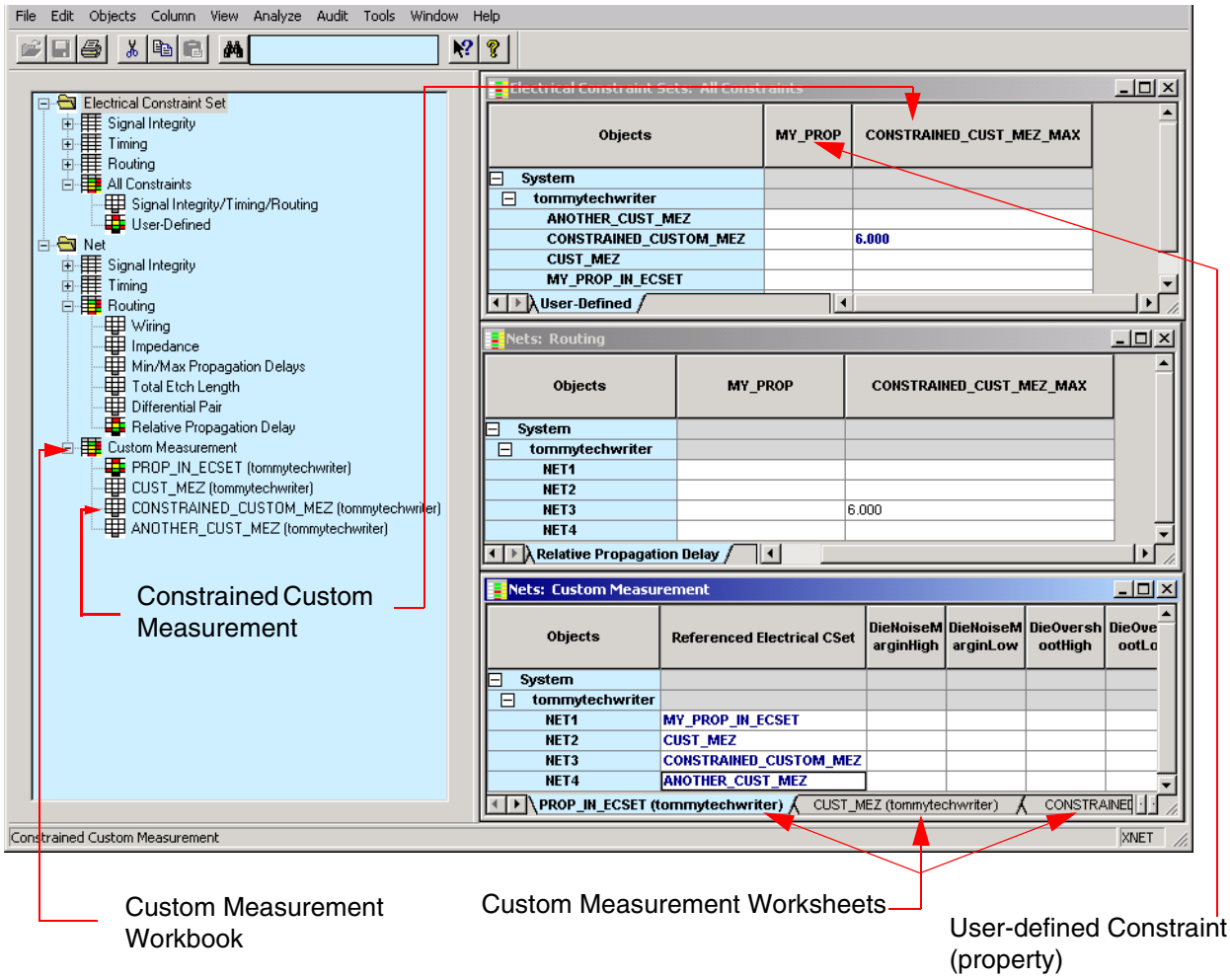
Tip

Constraint Manager retains the analysis modes settings that you define in the *Electrical CSet* when you export the topology to SigXplorer. See Analyzing with Custom Measurements and Custom Stimulus on page 172 for information about analysis modes.

Managing Custom Measurements

Once defined and imported from SigXplorer, custom measurements and constrained custom measurements populate an *Electrical CSet* under the *Custom Measurements* workbook in the *Net* folder. Each set of custom measurements or constrained custom measurements (an *Electrical CSet*) appears as an individual worksheet. Each custom measurement appears as a column in the worksheet (see Figure 7-3 on page 171).

Figure 7-3 User-defined Constraints and Custom Measurements



Constraint Manager maintains custom measurement and custom stimulus associations—*Electrical CSet* to object—as well as analyzed results from one session to the next.

Tip

When you select an *Electrical CSet* in the *Referenced Electrical CSet* column of the *Custom Measurements* worksheet, and then right-click and choose *SigXplorer* from the pop-up menu, that *Electrical CSet* is exported from Constraint Manager to SigXplorer as a topology with custom measurements and custom stimulus intact. Choosing *File – Update Constraint Manager* from SigXplorer then refreshes the custom measurements in Constraint Manager with any changes.

Analyzing with Custom Measurements and Custom Stimulus

Analyzing a net-related object with a custom constraint or custom measurement involves the same steps as described in The Analysis Process on page 133. Additionally, you must follow these steps:

➤ Specify Analysis Settings

Custom measurements apply only to *reflection* simulations. You specify the simulation type in the *Analysis Settings* dialog box. If you have custom stimulus defined in the *Electrical CSet*, it too must be enabled for analysis. See the Analysis Settings on page 133 for more information on how to specify the simulation type and how to enable custom stimulus.

➤ Enable Custom Measurements

You enable custom measurements through the *custom measurements* tab of the *Analysis Modes* dialog box (choose *Analyze – Analysis Modes*). Measurements appear as children in a tree structure with the parent object representing the *Electrical CSet* that contains the set of custom measurements.

Note: Constraint Manager, when launched from Allegro PCB Board L Series or OrCAD PCB Editor, do not support custom measurements or custom stimulus.

The check box adjacent to the parent object also serves as a toggle switch for all measurements in the *Electrical CSet*: *all on* (when checked) or *all off* (when unchecked). Only checked measurements appear in analysis results (see Analysis Results on page 138 for more information).

➤ Analyze Custom Measurements

You do not analyze a custom constraint or custom measurement; rather, you analyze the object that references an *Electrical CSet* which contains custom constraints or custom measurements.

The scope of a net object that contains a custom measurement can range from a pin-pair to a bit of a bus to an entire bus. Once you have (1) imported an *Electrical CSet* that contains custom measurements, (2) assigned the *Electrical CSet* to a net object, (3) specified analysis settings, and (4) enabled custom measurements, you then select the net object and choose *Analyze – Analyze* (or right-click and choose *Analyze* from the pop-up menu).

As many net objects can have the same custom measurement, you must click the appropriate tab to access the object that you want to analyze.

Tip

You can specify analysis settings, DRC modes, custom measurements, and desired reports from a single dialog box (choose Tools – Report).

Customizing Design Rule Checks

This section presents different methods used to customize DRCs, including

- Formulas
- User-defined Constraints
- User-defined Predicates
- User-defined Measurements

You can choose from pre-defined predicates and pre-defined measurements or you can create your own to address your unique design requirements. This section complements the procedures and reference information of the Edit – Formula command in the *Constraint Manager Reference*.

Formulas

To meet your unique constraint requirements, you can extend constraints and properties (pre-defined and user-defined) with formulas. A formula is required when a static value is not sufficient and a relatively simple calculation is required to compute the desired constraint or property value.

You can build a formula using the following methods, alone or in combination:

- Cell selection, mixed with operands
- Pre-defined and user-defined predicates
- Programatically, with the Cadence SKILL scripting language

Each method of defining a formula is progressively more complex, yet yields more power and flexibility in using them to constrain your design.

Inserting a formula in a cell

Click in any writable cell, and then right-click and choose *Formula* from the pop-up menu (or choose *Edit – Formula*). This invokes the *Single-line Editor*, as shown in Figure 7-4.

If you defined a formula using the *Multi-line Editor,* it will be called if you need to subsequently edit a formula. See Working with the Multi-line Editor in the *Constraint Manager Reference*. Constraint Manager remembers which editor that you last used in

building a formula. For easy identification, Constraint Manager renders the far end of a cell that contains a formula with a thin, red, vertical stripe.

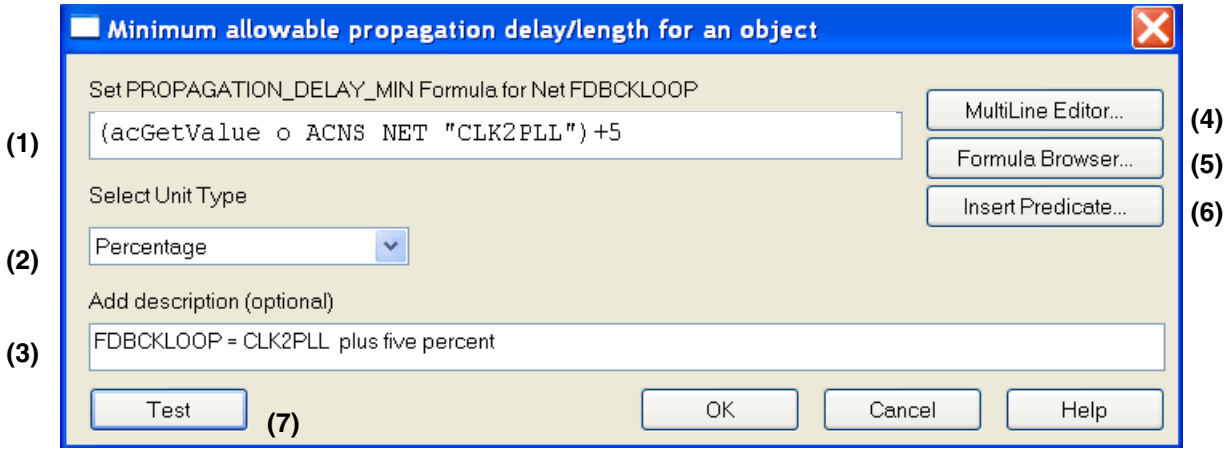
The Single-line Editor

You use the *Single-line Editor* to build simple formulas, which may contain user-defined constraints, properties, and predicates.

In Figure 7-4, Item (1) is where you type in your formula. You can use point-and-click to add cell addresses to the formula. Item (2) is the *Unit Type*, which varies depending on the cell being modified by the formula. You enter a description of the formula in Item (3). Item (7) exercises the formula. You should always test your expressions for proper syntax and function. Item (4) launches the *Multi-line Editor* used to create formulas programmatically (see Programmatic Formulas on page 179).

Once you create formulas, Item (5), they are available in the formula browser for use elsewhere in the design. Item (6) is for adding predicates to the formula (see Pre-defined Predicates on page 178).

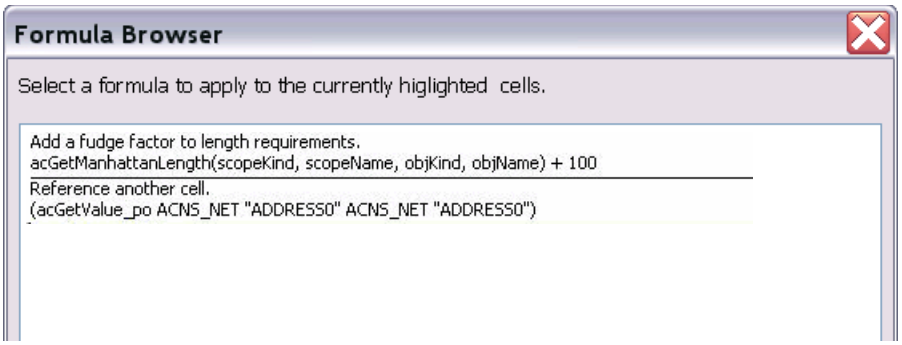
Figure 7-4 Single-line Editor



A Library of Formulas

The *Formula Browser* lets you view all the formulas in the design and assign them to specific cells.

Figure 7-5 The Formula Browser



Cell selection and operands

The cell selection and operand method provides for point-and-click cell selection, mixed with operands to build a formula. Clicking within a cell adds the code required to access that cell in the formula. For example, `(acGetValue_o ACNS_NET "CLK2PLL")+5` results from selecting a cell for an another object in the worksheet, in the same column as the one selected for Formula editing, and then typing `+5`.

Cell selection automatically inserts the appropriate predefined predicate to get the value for the selected cell. The inserted predicates will differ based upon what information is needed to retrieve the value. For example, the same object in a different cell only needs the name of the cell's attribute because the object being edited is considered the default object.

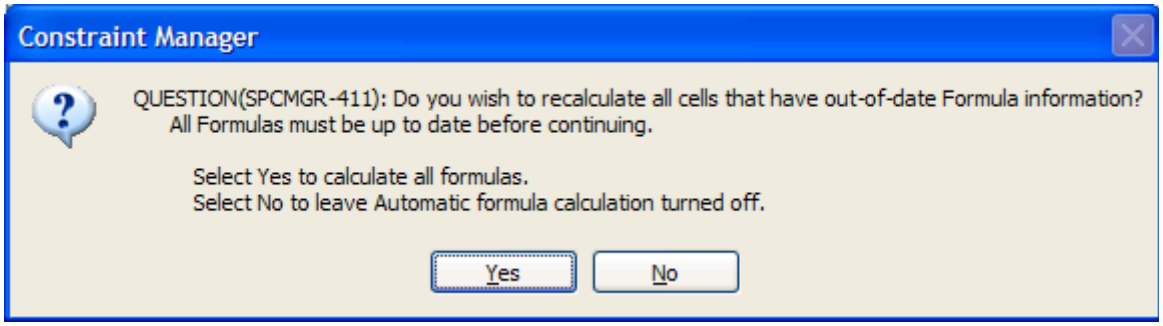
Calculating a Formula

Constraint Manager keeps track of a formula's dependencies on other cells or database objects. For example, the value in a cell or the etch length of a net. When one of those dependencies changes, the formula is known to be out of date or stale. If automatic formula calculation is on, out of date formulas recalculate their new values as needed. If automatic formula calculation is off, out of date formulas are indicated by the background of the cell turning yellow.

You can use the *Edit – Calculate* or *Edit – Calculate All* menu commands to bring these formulas back up to date.

To turn the automatic formula calculation on or off, select or deselect the *Automatic formula calculation checkbox* in the Options dialog box (choose *Tools – Options*).

When the automatic formula calculation function is set from Off to On, the following message is displayed



You can also review or modify the dependencies using the Dependencies dialog box (choose *Edit – Dependencies*).

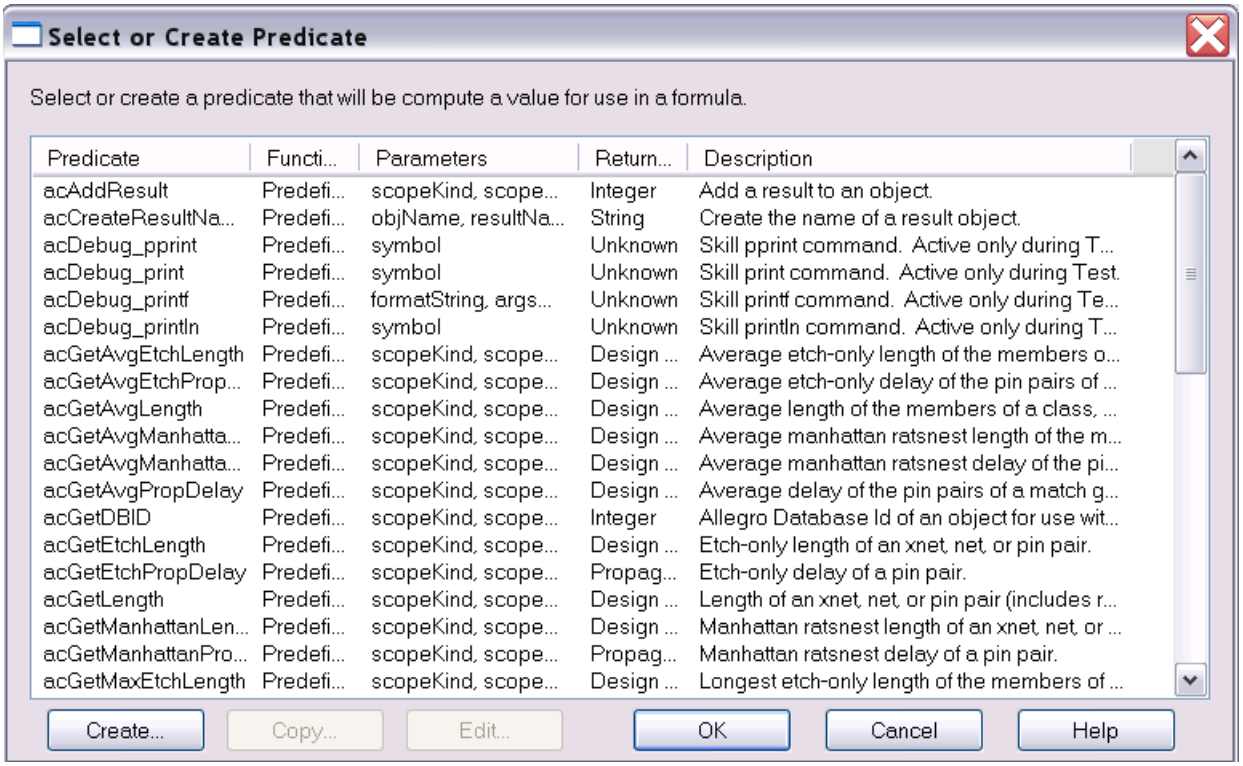
See *Automatic Formula Calculation* in Allegro Constraint Manager Reference for more information.

Pre-defined Predicates

Predicates are functions that return data (usually a single value). They work to access design information to enhance the functionality of formulas, other predicates, and measurements (see Creating User-defined Predicates and Measurements on page 186). You can choose from pre-defined predicates and you can create your own. See Edit – Formula in the *Constraint Manager Reference* for a list of pre-defined predicates, their supported parameters, and their function.

You access the *Select or Create Predicate* dialog box (see Figure 7-6) by clicking *Insert Predicate* (see Item 6 in Figure 7-4 on page 175).

Figure 7-6 Select or Create Predicate dialog box



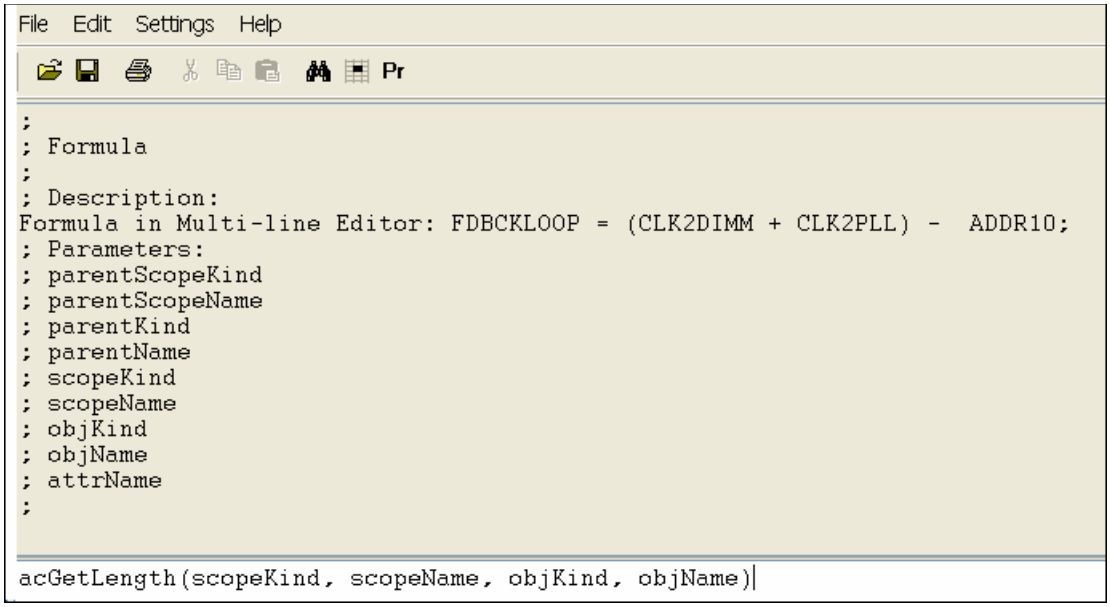
Note: See Predicate Parameters on page 189 for column definitions.

In addition to a broad collection of pre-defined predicates, you can create your own (see Creating User-defined Predicates and Measurements on page 186).

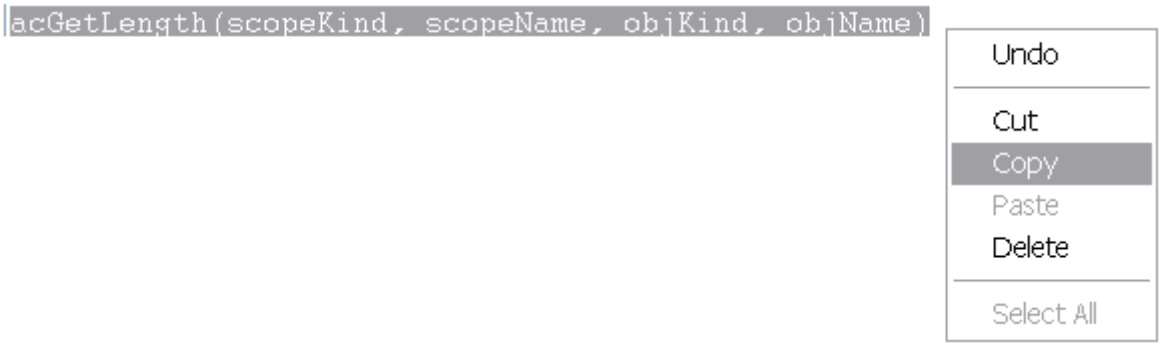
Programmatic Formulas

The *Multi-line Editor* (see Item 6 in Figure 7-4 on page 175) allows for the most flexibility when adding programming constructs to formulas. For usage information, see Using the Multi-line Editor in the *Constraint Manager Reference*.

The top of the *Multi-line Editor* shows the formula's description and the list of parameters; the bottom shows the formula that you are building.



You can type any text in the *Multi-line Editor* to build your formula. Common editor functions, such as *Copy* and *Paste*, are selectable from the context menu (right-click).



You can improve the formula by assigning each predicate to a variable and wrapping the formula in a `let` statement. The `let` statement also protects the variables from being influenced by global access.

```
let( (a b c)
   ;
   ; Set up lengths to be used in the calculation
   ;
   a = acGetLength(scopeKind, scopeName, objKind, "CLK2PLL")
   b = acGetLength(scopeKind, scopeName, objKind, "CLK2DIMM")
   c = acGetLength(scopeKind, scopeName, objKind, "ADDR<10>")
   ;
   ; Caluculate the value to be returned by the formula
   ;
   (a + b) - c
)
```

Support for Online Formula

Constraint Manager determines and stores dependency information for all the Advanced Constraints formulas as they are created or loaded for the first time. The dependency information stored with the formulas such that a formula is not recreated each time the design is closed and reopened.

Dependency information is used to update the status of calculated values and indicates when they have become stale.

Some cases where calculated values can become stale (out of date) are:

- A formula on a cell that references the value in another cell.
- A formula that calls a predicate that returns some information about a database object.

When values are changed in Constraint Manager, or if database objects are updated in PCB Editor, Constraint Manager determines which values are out of date, and recalculates them when necessary.

Note: Launching Constraint Manager marks all formula's stale. This is indicated by cell with formula's colored yellow. When you run *Calculate* on a stale cell, the cell value is updated and cell color is cleared. If you run *Calculate All,* all the cells in all worksheets are updated and the yellow color is cleared.

See *Options Dialog Box* in Allegro Constraint Manager Reference for more information.

Testing and Debugging Formulas

You can use Constraint Manager's test environment to debug formulas. In a *debug* session, Constraint Manager temporarily makes changes to the database; however, these changes will not be saved once you exit the test environment.

You enter the test environment from the

- *Multi-line Editor (*choose *File – Test)*
- Single-line Editor (click *Test*). See Item 7 in Figure 7-4 on page 175.

Formula testing produces a log file similar to the log file used for testing user-defined predicates and user-defined measurements (see Figure 7-12 on page 193). See the Multi-line Editor commands of the *Edit – Formula* command in the *Constraint Manager Reference* for information on how to interpret these log files.

Tip

Constant Manager supplies many pre-defined predicates (look for `acDebug_print*`), which you can use to debug your formulas. See Pre-defined Predicates in the *Constraint Manager Reference*.

Creating User-defined Constraints

In addition to its broad set of pre-defined constraints, Constraint Manager lets you define constraints to meet your unique design requirements. A user-defined constraint is required to capture a unique design requirement, which is not represented in the default constraint system. User-defined Constraints also specify how to validate a new column.

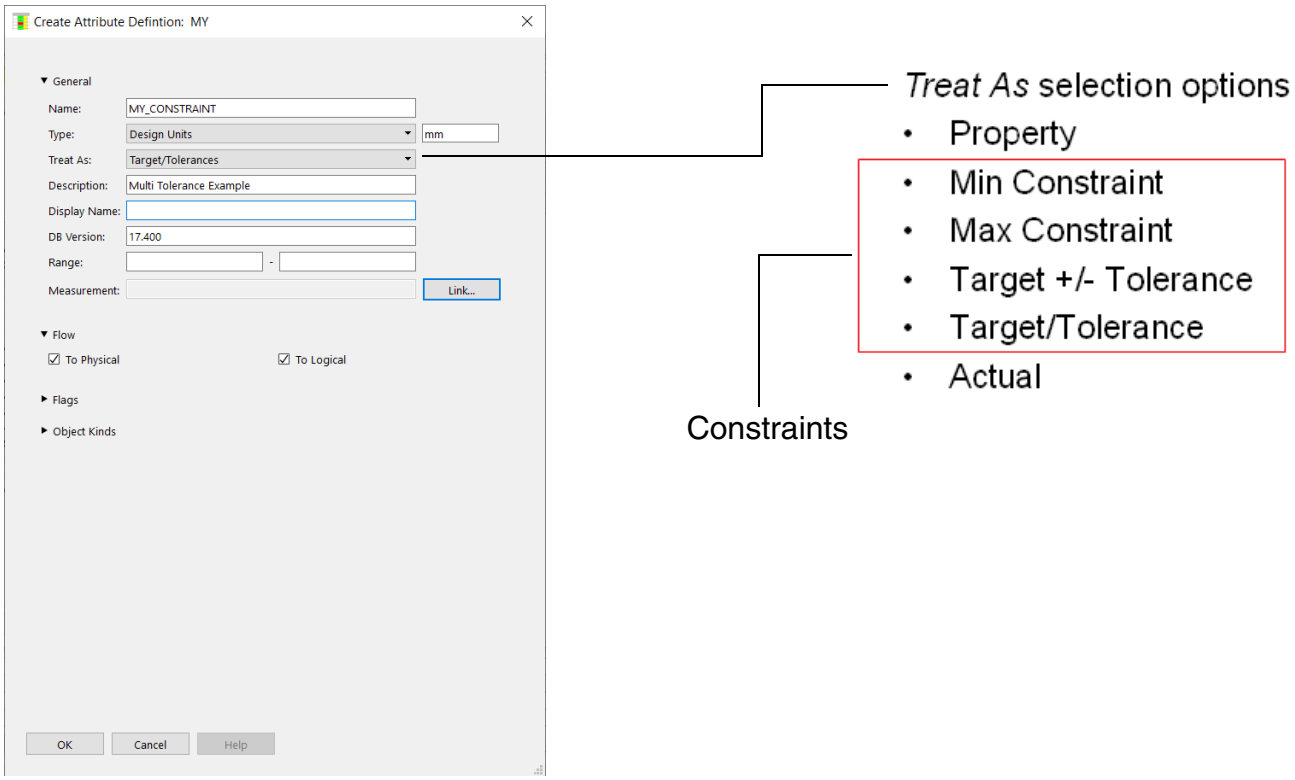
Important

Unlike pre-defined constraints, user-defined constraints analyze only within Constraint Manager and not in a PCB or package editor.

A user-defined constraint requires its own column. For information on adding custom workbooks, worksheets, columns, and attributes, see

- Customizing Worksheets on page 163.
- The Tools – Customize Worksheet command in the *Constraint Manager Reference.*

Figure 7-7 Adding a column and a constraint



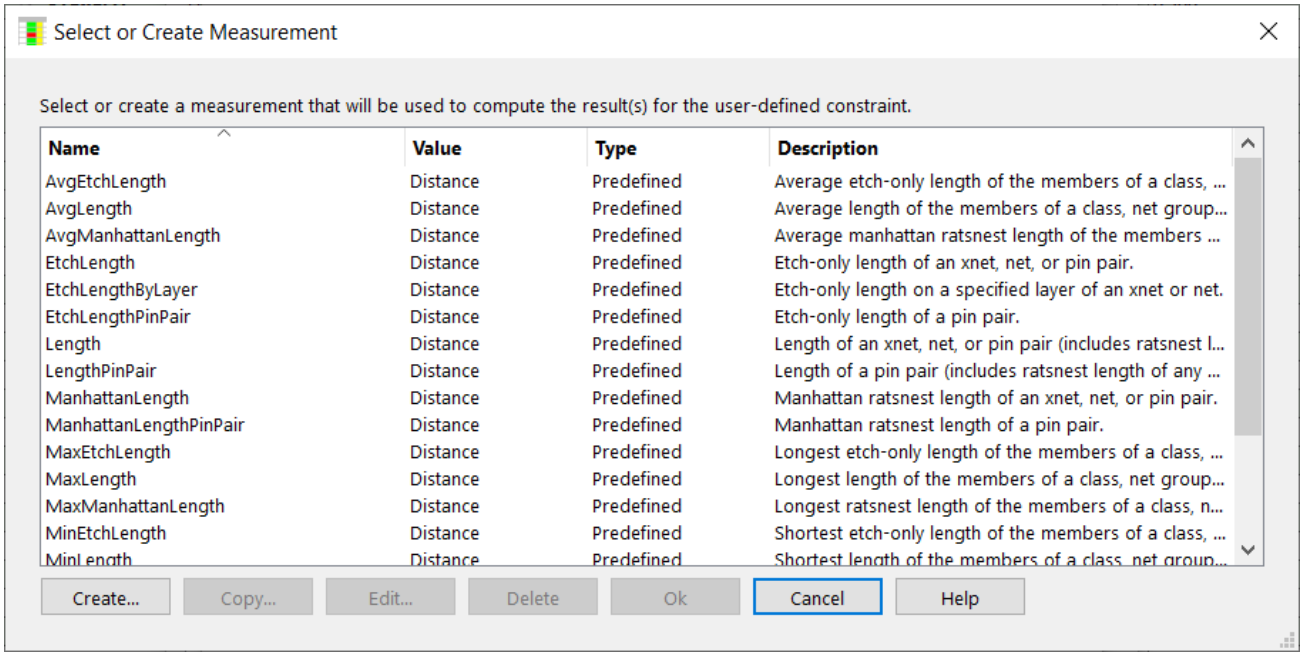
Tip

The *Treat As* drop-down menu in the *Create Attribute Definition* dialog box (see Figure 7-7 on page 183) is where you specify the new user-defined attribute's type. This can be a property, a constraint, or an actual.

- *Property* specifies a user-defined property
- *Min Constraint, Max Constraint, Target +/- Tolerance, Target/Tolerance* specifies a user-defined constraint, and a corresponding *Actual* and *Margin*

  Target/Tolerance allows for separate plus and minus tolerance values; *Target +/- Tolerance* allows for a single tolerance value
- *Actual* specifies a single column that is linked to a measurement and used to display the measurement results. An *Actual* can be part of a user-defined constraint.

When you click *OK* to accept one of the above constraint types, the system automatically creates a related *Actual* and *Margin* for the constraint. The related *Actual* also needs an associated measurement to provide results when the constraint is analyzed. As such, the system presents the *Select or Create Measurement* dialog box.



Note: The *Copy*, *Edit*, and *Delete* buttons are enabled only for user-defined measurements. If you select a user-defined measurement in the list, and click the *Delete* button, a warning message appears about some Actuals which may no longer analyze correctly. If you select *OK*, the measurement is deleted from the list.

Note: The list of measurements is based on the *Data Type* field in the *Create Attribute Definition* dialog box (see Figure 7-7 on page 183). If the pre-defined measurements are not what you need, you can create your own (see Creating User-defined Actuals on page 195).

Creating User-defined Predicates and Measurements

Constraint Manager has many pre-defined, readily-usable, predicates and measurements and it provides the capability to augment them with predicates and measurements that you build to meet your unique design requirements. Predicates function as the building blocks of Formulas, Measurements, and other Predicates.

User-defined Predicates

Predicates are functions that return data (usually a single value). They work to access design information to enhance the functionality of formulas, other predicates, and measurements.

Predicates are more flexible than formulas or measurements. You can define any parameters in any order and have any return type that you want. When you use a predicate you have to match its use to its definition—all parameters have to match the definition and its return value must be appropriate for the object type.

You can write a simple predicates to

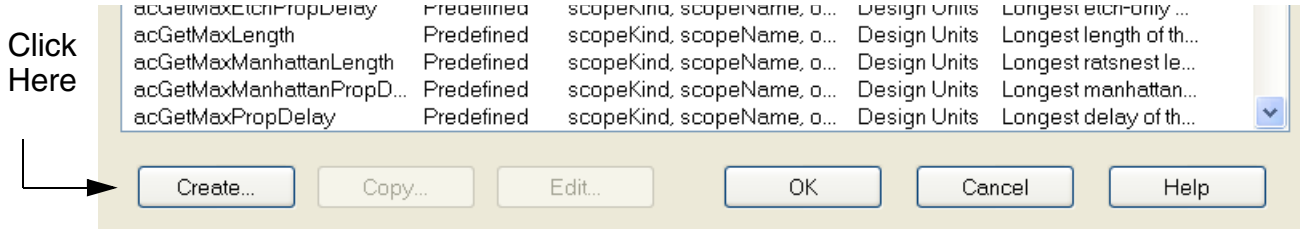
- sort a list of numbers
- find the smallest number
- open, close, or write to a log file
- perform an algebraic calculation

You can write a complex predicate to

- determine the overlap of two physical objects
- return the distance of an object to a plane
- return a list of database objects that match a specific criteria
- find the closest via to a cline

The road to a predicate is through both formulas and user-defined measurements. Once you add a formula (see Inserting a formula in a cell on page 174), choose *Insert Predicate* (see Item 6 in Figure 7-4 on page 175).

➤ In the *Select or Create Predicate* dialog box, choose *Create*.

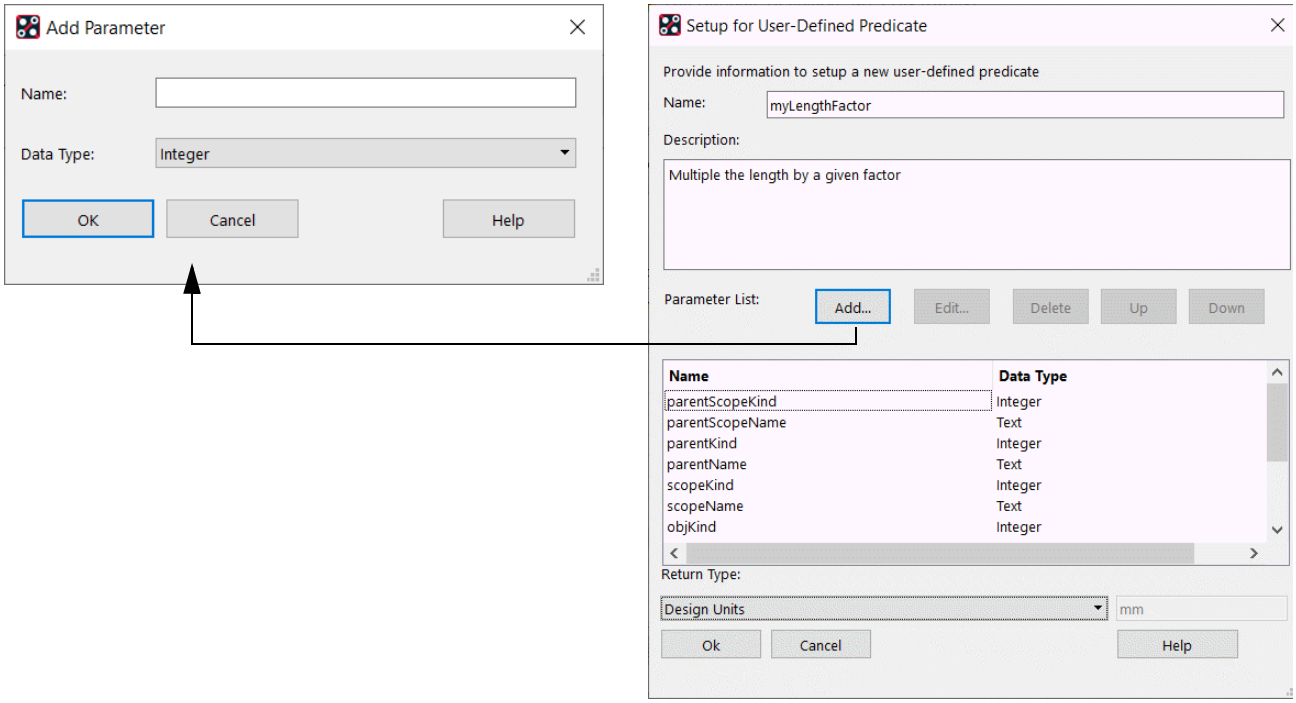


This invokes the *Setup for User-Defined Predicate* dialog box, as shown in Figure 7-8.

Defining a predicate

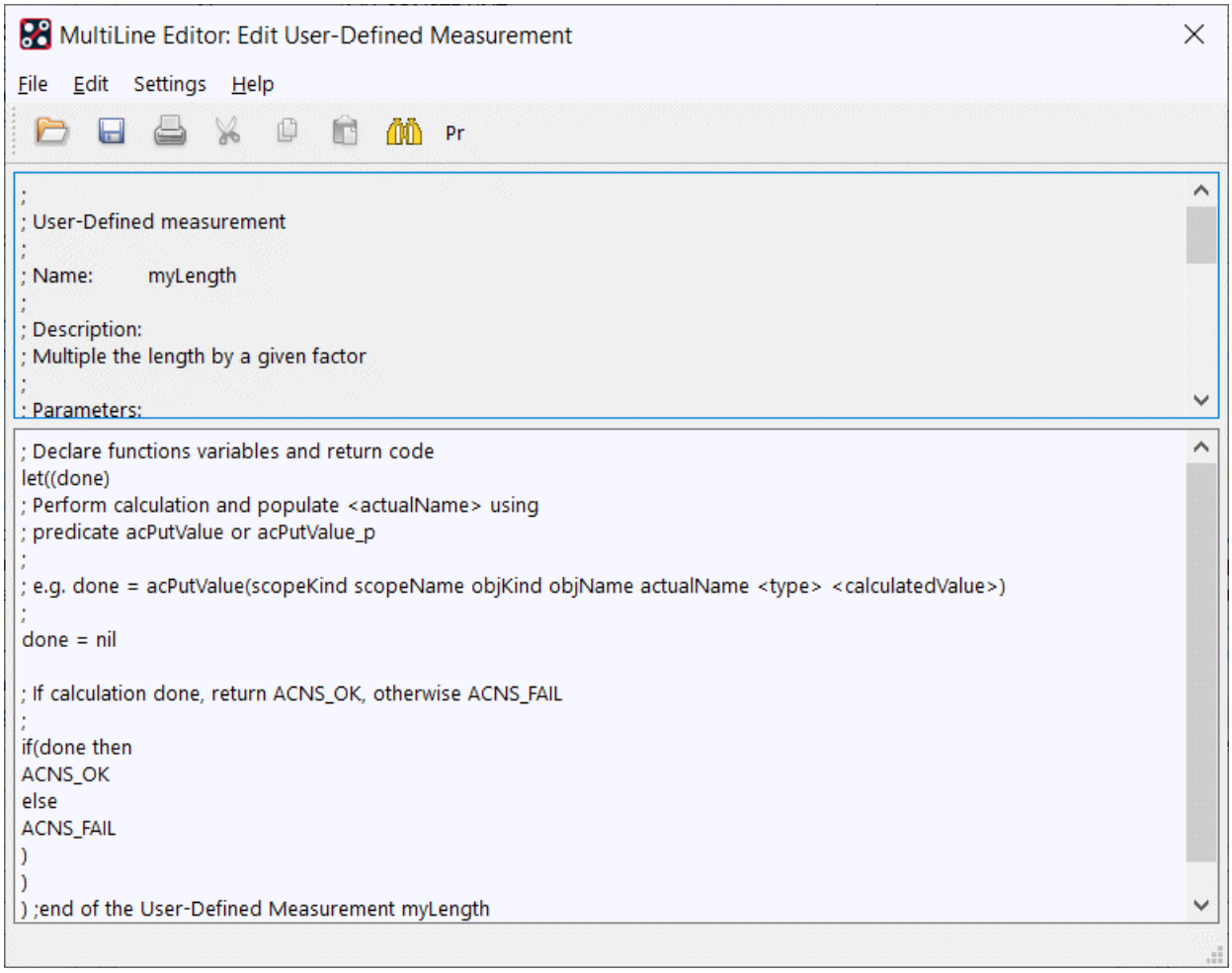
You supply a name, a description, a list of parameters, and a return type (boolean, integer, real, or text). Parameters are the conduit between a predicate and a formula (see Predicate Parameters on page 189).

Figure 7-8 Defining a predicate

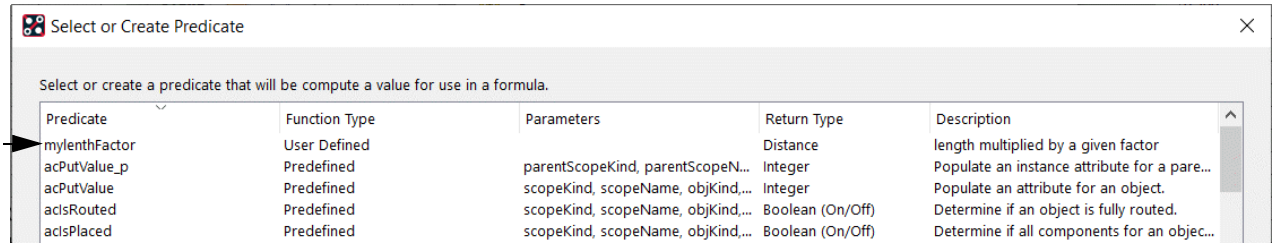


➤ Once you complete all fields and click *OK*, the *Multi-line Editor* appears (see Figure 7-9 on page 188).

Figure 7-9 Editing a predicate



Once you code and save your predicate, the *Select or Create Predicate* dialog box appears with your new predicate, ready for future use.



Predicate Parameters

A formula contains parameters that are automatically filled in with information about the cell that contains the formula (see Edit – Formula in the *Constraint Manager Reference* for a list of formula parameters). A predicate also has parameters, often of the same name and function as those in formulas.

If you want to use a predicate to get information about the current cell, you can just use these parameters as is in the predicate call:

```
acGetLength(scopeKind, scopeName, objKind, objName)
```

If you want to use a predicate to get information about other objects, you can substitute the predicate parameters with other values.

- `scopeKind` parameters include `ACNS_SYSTEM` and `ACNS_DESIGN`
- `scopeName` parameters are strings containing the name of your system or one of the designs in it

Substitute `scopeKind` and `scopeName` together to indicate which *design* or *system* you want to reference to find your object. If you are not using Design Links, you do not need to define these objects.

Substitute the `objKind` parameters to specify the type of the object you are referencing. These include:

- `ACNS_NULL`
- `ACNS_SYSTEM`
- `ACNS_DESIGN`
- `ACNS_XNET`
- `ACNS_NET`
- `ACNS_REGION`
- `ACNS_ECSET`
- `ACNS_CLASS`
- `ACNS_DIFFPAIR`
- `ACNS_BUS`
- `ACNS_PINPAIR`

- ACNS_MATCHGROUP
- ACNS_RESULT
- ACNS_PCSET
- ACNS_SCSET
- ACNS_CLASS_CLASS
- ACNS_REGION_CLASS
- ACNS_REGION_CLASS_CLASS

Note: Use ACNS_NULL when you do not want any objects, such as a substitute for the parent object.

Substitute the objName parameter to specify a string containing the name of the object. Substitute the attrName parameter to specify a string containing the name of an attribute.

Examples of predicate parameters

Requirement	Predicate Call
To find the length of a system-level Xnet in your formula	`acGetLength(ACNS_SYSTEM, "MY_SYSTEM", ACNS_XNET, "SOME_XNET")`
To find the length of the current object **Note:** You do not have to substitute the parameters, as they are substituted automatically based on the current object row	`acGetLength(scopeKind, scopeName, objKind, objName)`
If you have a formula on *NET1* and you want the length of *NET2*	`acGetLength(scopeKind, scopeName, objKind, "NET2")`
If you have a formula on *NET1* in *DESIGN1* and you want the length of *NET1* in *DESIGN2* of the active System	`acGetLength(scopeKind, "DESIGN2", objKind, objName)`

Requirement	Predicate Call
If you have a formula on *NET1* in *DESIGN1* and you want the length of *XNET2* in *MYSYSTEM*	`acGetLength(ACNS_SYSTEM, "MYSYSTEM", ACNS_XNET, "XNET2")`

Testing and Debugging User-defined Predicates

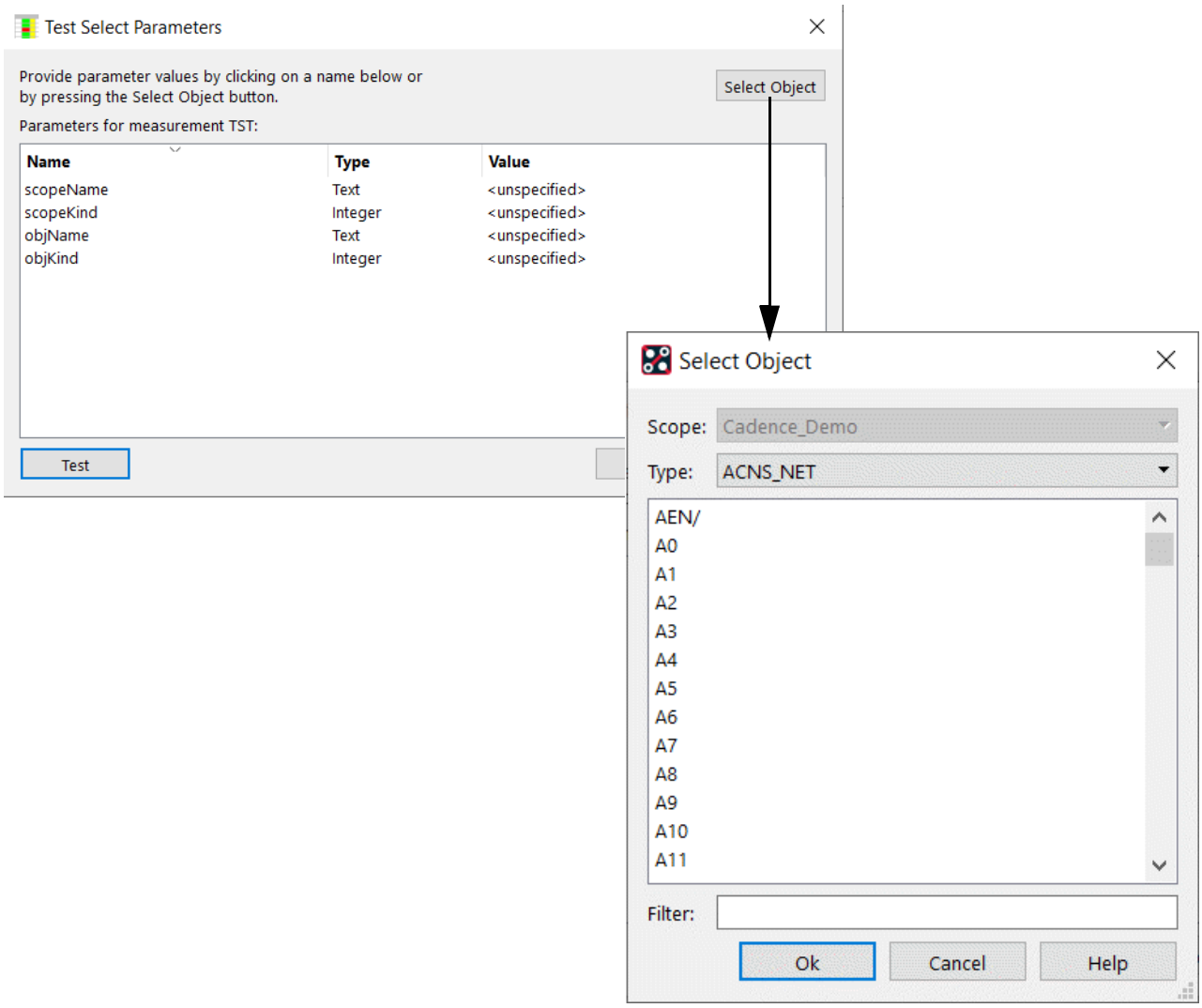
You can use Constraint Manager's test environment to debug user-defined predicates (and user-defined measurements, as described in Testing and Debugging User-defined Measurements on page 199). In a *debug* session, Constraint Manager temporarily makes changes to the database, based on the parameters that you supply. These changes will not be saved once you exit the test environment.

You enter the *Test Environment* from the *Multi-line Editor (Choose File – Test).* Before you can run a test, you must supply parameter `VALUES`. The *Test Select Parameters* dialog box (see Figure 7-10 on page 192) lists all the parameters that you defined when you set up the predicate. When you click on a parameter `NAME`, a pop-up window appears, and is based on the parameter `TYPE` (Boolean, Integer, Real, Text).

Tip

Constant Manager supplies many pre-defined predicates (look for `acDebug_print*`), which you can use to debug your user-defined predicates. See Pre-defined Predicates in the *Constraint Manager Reference*.

Figure 7-10 Test Select Parameters dialog box

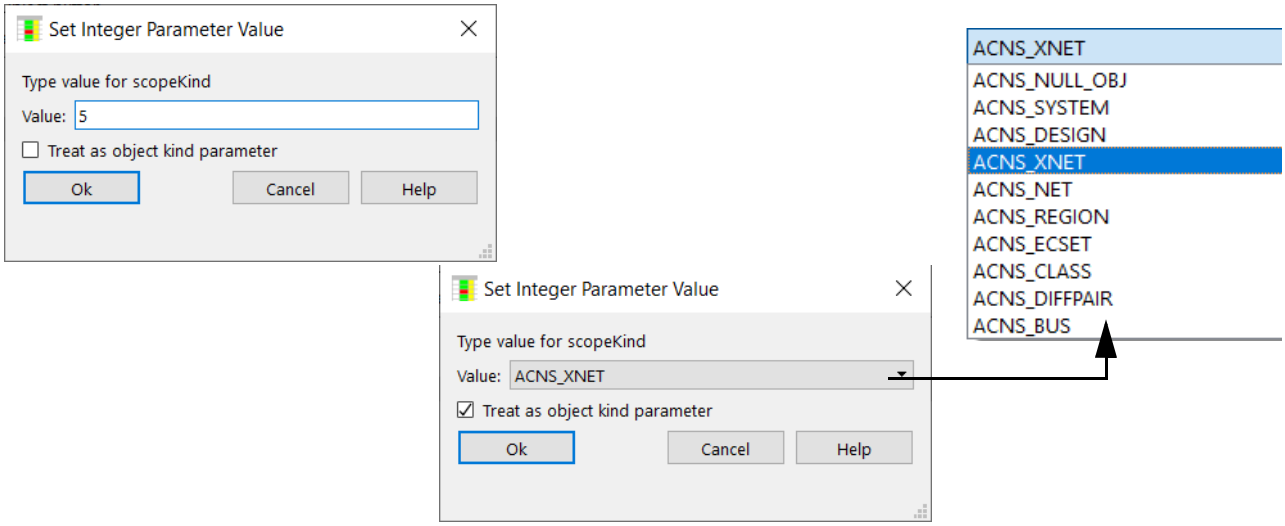


In Figure 7-10, the *Select Object* window lets you select any object, and information about this object is used to automatically populate the corresponding parameters.

Note: For measurements, this will populate all visible parameters. For predicates, any parameters with the same name as the measurement parameters are populated. When you press *OK*, the *Test Select Parameters* dialog box returns with as many values as possible, updated automatically.

Boolean-, Real-, and Text-parameter `TYPEs` accept a `VALUE` that you enter directly; Integer parameter `TYPEs` accept a direct `VALUE` when the *Treat as object kind parameter* is unchecked. If checked, you are presented with a list of all parameter `TYPEs` in the design.

Figure 7-11 Parameter values by integer and by object kind



Once all parameters are entered in the *Test Select Parameters* dialog box, click *Test*. A debug window appears with metrics of the test, including the function's name, calculated results, return value, and LINT parsing.

Figure 7-12 Functional test output

```
**** DEBUG OUTPUT ****
Begin function
Length 12425.8, Calculated Length 15532.3


*** RETURN VALUE ***
15532.29


**** SKILL LINT ****
INFO (REP008): Program SKILL Lint started on Oct 15 09:42:35 2007.
INFO (PREFIXES): Using prefixes: "none"
INFO (STRICT): Using strict checking of global variables.
INFO (IQ): IQ score is 100 (best is 100).
INFO (IQ1): IQ score is based on 0 error messages, 0 general warning messages.
INFO (REP110): Total enhancement      : 0.
INFO (REP110): Total external global  : 0.
INFO (REP110): Total package global   : 0.
INFO (REP110): Total warning global   : 0.
INFO (REP110): Total error global     : 0.
INFO (REP110): Total unused vars      : 0.
INFO (REP110): Total next release     : 0.
INFO (REP110): Total alert            : 0.
```

You can continue to test the predicate by supplying new or different parameters, and values, in the *Test Select Parameters* dialog box (see Figure 7-10 on page 192).

When you click *Close* in the *Function Test* dialog box (see Figure 7-12), then the *Test Select Parameters* dialog box, you exit the *Test Environment* and return to the *Multi-line Editor* (see Figure 7-13).

Figure 7-13 A user-defined Predicate with debugging statements

```
; Declare functions variables
let((tempValue returnValue)
    ; Calculate returnValue
    ;
    acDebug_printf("Begin function\n")
    tempValue = acGetEtchLength(scopeKind scopeName objKind objName)
    returnValue = tempValue * factor
    acDebug_printf("Length %g, Calculated Length %g\n" tempValue returnValue)

    ; Return computed value
    returnValue
)
) ;end of the User-Defined Predicate myLengthFactor
```

Creating User-defined Actuals

A user-defined *Actual* is an attribute that is associated with a measurement. When Constraint Manager analyzes the *Actual*, the measurement is triggered and the cell that contains the *Actual* is populated with results.

A user-defined *Actual*

- is a term used interchangeably with the term *Measurement*
- displays the results of a measurement
- can occupy a user-defined column to facilitate custom reports
- can be used in defining a user-defined constraint

You must add a column to a worksheet to accommodate an *Actual* cell. For information on adding custom workbooks, worksheets, columns, and attributes, see

- Customizing Worksheets on page 163.
- The Tools – Customize Worksheet command in the *Constraint Manager Reference.*

The measurement is coupled with the *Actual* and executes when the *Actual's* associated constraint is analyzed.

Note: The measurement executes for the object being analyzed and typically returns results (*Actual* values) for the same.

Once the new column is in place, the *Create Attribute Definition* dialog box appears (see Figure 7-14 on page 196). Note that *Actual* is selected from the *Treat As* drop-down menu.

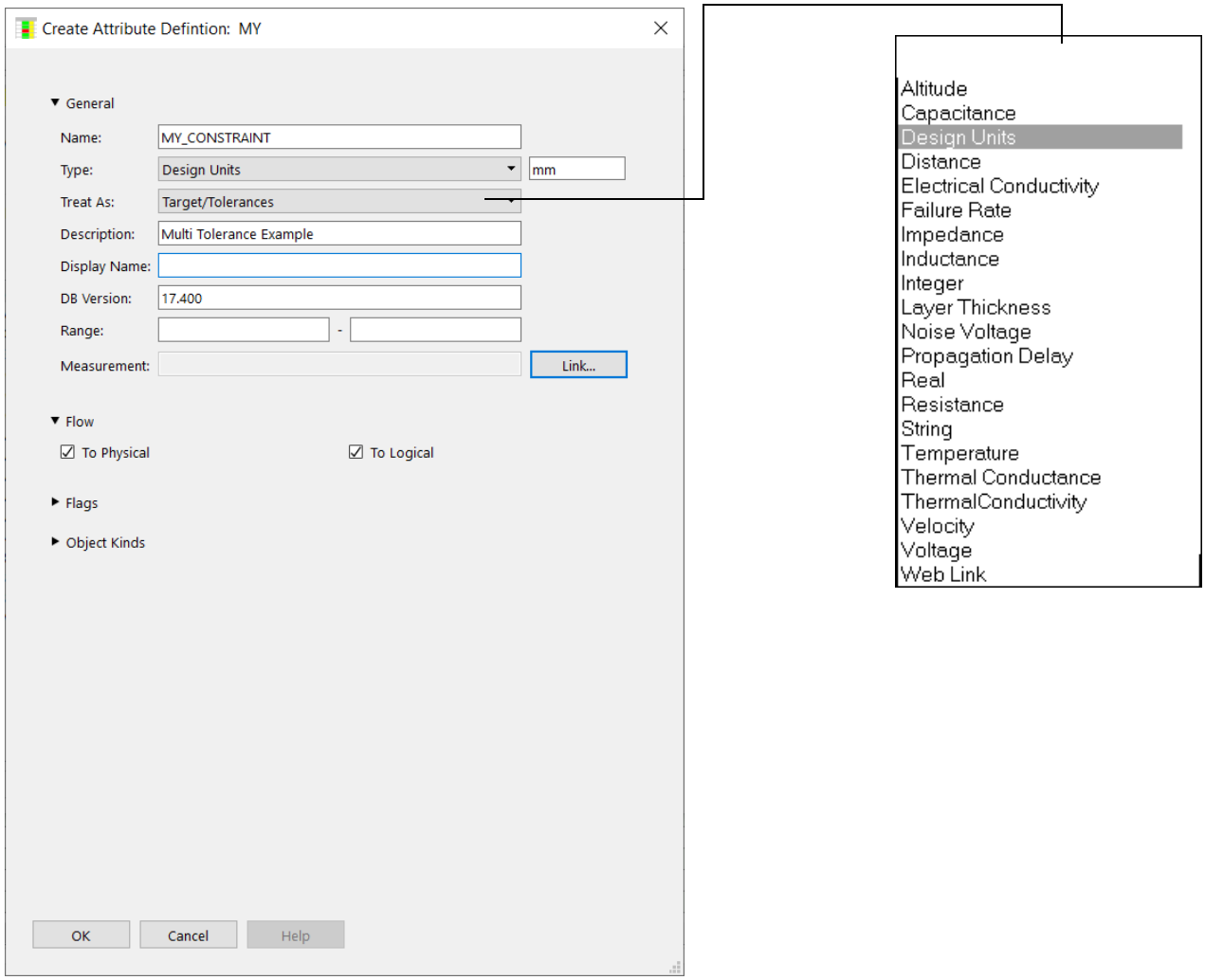
➤ Complete the *Setup for User-defined Measurement* dialog box as shown in Figure 7-15 on page 197. Specify a name, a description, and supported objects.

Note: The *Parameter List* is fixed and the buttons are inactive for user-defined measurements. This dialog box is also used to set up user-defined predicates (see User-defined Predicates on page 186).

Creating User-defined Measurements

A user-defined measurement is required to compute a unique result, which is needed to validate the design intent. It requires a new column and it governs how to calculate the results of the new column.

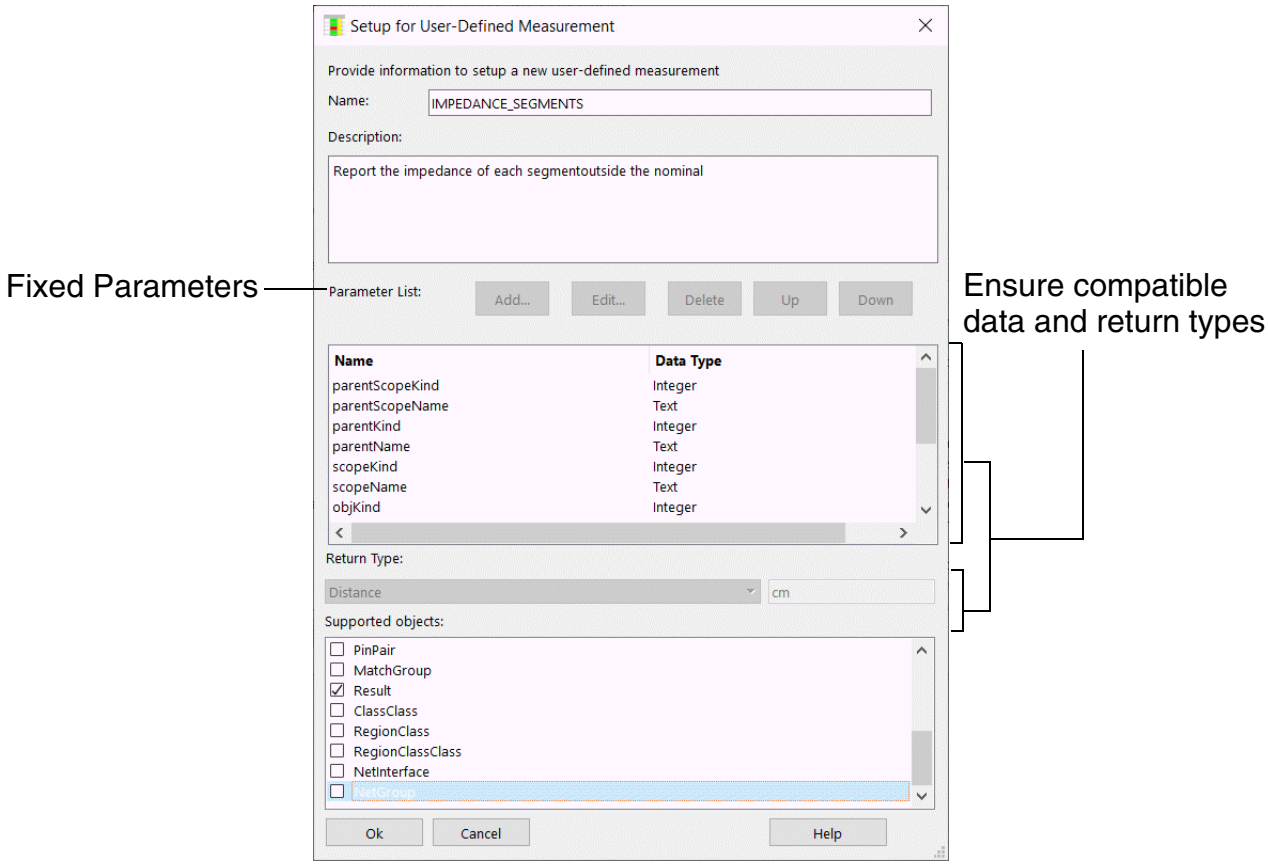
Figure 7-14 Measurement attribute definition (Treat As equals Actual)



User-defined Measurements . . .

- have a fixed set of parameters, which are populated by Constraint Manager upon analysis.
- are based on data types, such as design units, as defined in the *Data Type* drop-down menu
- are based on supported object. *Actuals* calculate for rows that contain supported objects.
- return `ACNS_OK` to indicate successful completion; `ACNS_FAIL` to indicate errors.

Figure 7-15 Setting up your measurement



Once completed, click *OK*. The *Multi-line Editor* appears. For information on using the *Multi-line Editor* (see the Edit – Formula command in the *Constraint Manager Reference)*.

Figure 7-16 A User-defined measurement

```
;
; User-Defined measurement
;
; Name: newMeas1
;
; Description:
; ;
; Parameters:
; parentScopeKind       Integer
; parentScopeName       Text
; parentKind    Integer
; parentName    Text
; scopeKind     Integer
; scopeName     Text
; objKind       Integer
; objName       Text
; constraintName        Text
; actualName    Text
; vstatusName   Text
; verifyMode    Integer
;
; Return Value:
; Design Units
;
procedure( newMeas1 (parentScopeKind parentScopeName parentKind parentName scopeKind scopeName objKi
```

```
   ; Declare functions variables and return code
   let((done tempValue)
      ; Perform calculation and populate <actualName> using
      ; predicate acPutValue or acPutValue_p
      ;
      ; e.g. done = acPutValue(scopeKind scopeName objKind objName actualName <type> <calculatedValu
      ;
      acDebug_printf("Start Measurement\n")
      tempValue = myLengthFactor(scopeKind scopeName objKind objName 1.25)
      done = acPutValue(scopeKind scopeName objKind objName actualName ACNS_DOUBLE_TYPE tempValue)

      ; If calculation done, return ACNS_OK, otherwise ACNS_FAIL
      ;
      if(done then
          ACNS_OK
      else
          ACNS_FAIL
      )
   )
) ;end of the User-Defined Measurement newMeas1
```

When defining a measurement:

- Use one of the `acPutValue()` predicates to populate the *Actual*. The *Actual* is passed into the measurement through the `actualName` parameter.
- If the measurement cannot compute a result, populate the verification status with the reason using `acPutValue()` predicate, with an `ACNS_STRING_TYPE`. The verification status is passed into the measurement through the `vstatusName` parameter.

- If the measurement needs to create a new result object to display under the object being measured, the following predicates are required:
  - `acCreateResultName` to create the name of the new object
  - `acAddResult` to associate the new object with the object being analyzed
  - `acPutValue` to populate the actualName on the new object

➤ When you are finished with defining the measurement, *Save* and *Close* the *Multi-line Editor*. The new measurement is available for selection.

Tip

A measurement returns a status to the calling function, not a calculated value: `ACNS_OK` upon success; `ACNS_FAIL` upon encountering an error.

Calculated values populate an *Actual* cell using the `acPutValue` predicate in the body of the measurement.

Actual cells are typically populated for the current object or for result objects created by the measurement using the `acCreateResultName` and `acAddResult` predicates.

➤ Once you dismiss the remaining dialog boxes, right-click on the new column head and choose *Analyze* from the pop-up menu. This yields results for the new *Actual* cell with the measurement that you just defined.

Testing and Debugging User-defined Measurements

Most parameters for user-defined measurements are determined from the object row that is being measured; however, because a measurement can be applied to more than one row, this information is not known when the measurement is being created or edited. A set of values for the parameters must be provided before the measurement can be tested.

Tip

Constant Manager supplies many pre-defined predicates (look for `acDebug_print*`), which you can use to debug your user-defined measurements. See Pre-defined Predicates in the *Constraint Manager Reference*.

All measurements require the following, fixed set of parameters:

- `parentScopeKind`
- `parentScopeName`
- `parentKind`
- `parentName`
- `scopeKind`
- `scopeName`
- `objKind`
- `objName`
- `constraintName`
- `actualName`
- `vstatusName`
- `verifyMode`

When running the measurement, the last four parameters are always the same; therefore, you do not need to specify them.

Some of these parameters can be determined by Constraint Manager, or they are always the same when running the measurement. These parameters do not need to be specified and will not appear in the parameter list in the *Test Select Parameters* dialog box. Usually, only the `scopeKind`, `scopeName`, `objKind`, and `objName` parameters need to be specified

All measurements include one of the following `TYPE` parameters:

- `ACNS_DOUBLE_TYPE`
- `ACNS_STRING_TYPE`
- `ACNS_ENUM_TYPE`
- `ACNS_INTEGER_TYPE`
- `ACNS_LONG_TYPE`
- `ACNS_DOUBLE_ARRAY_TYPE`
- `ACNS_STRING_ARRAY_TYPE`
- `ACNS_ENUM_ARRAY_TYPE`

- ACNS_INTEGER_ARRAY_TYPE
- ACNS_LONG_ARRAY_TYPE

The TYPE parameter indicates how the calculatedValue parameter is stored in the attribute named by the actualName parameter. Enum types are treated as strings and must be a legal value for the actualName. Array types are lists of values used to populate attributes that contain multiple values (colon separated strings).

Important

The *Test Environment* for debugging user-defined measurements is identical to that of user-defined predicates.

See

- Creating User-defined Measurements on page 196
- Testing and Debugging User-defined Predicates on page 191

8

Generating and Viewing Constraints Differences

Topics in this chapter include

Generating Constraints Differences Report

Managing design differences is critical to deliver a quality product with today's complex designs. Design differences include connectivity (logical), implementation (physical), and constraint changes.

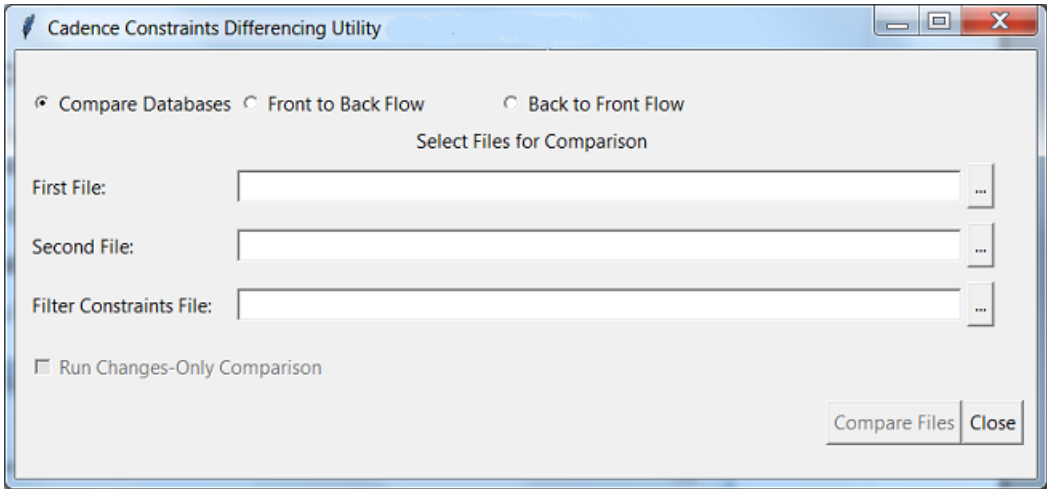
Comparing Constraints using Differencing Utility

The *Constraints Differencing Utility* compares two databases and reports the differences in constraints values. Following types of databases can be compared:

- Schematic database (`*.cpm`)
- Layout database (`*.brd`, `*.sip`, `*.mcm`)
- Constraints Manager database (`*.dcf`, `*.tcf`)

You can run this utility anytime during the design flow and see the differences in constraints between two databases before proceeding further.

To invoke the utility, run `cmDiffUtility` command from the command prompt. This command opens the *Cadence Constraints Differencing Utility* dialog box.



On selecting schematic database created in Allegro System Architect, the utility always run in *Compare Databases* mode. The schematic databases are only compared and constraints difference report is generated.

Comparing The Same Databases

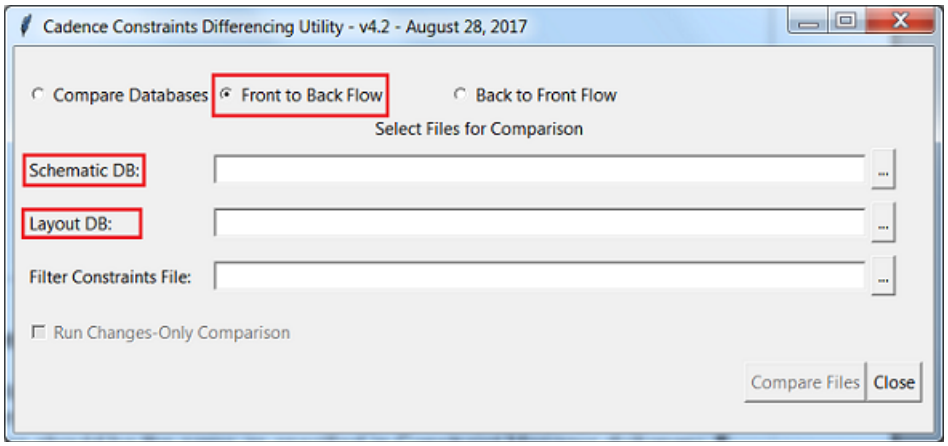
- Schematic: Compares two different versions of the same schematic database.
- Layout: Compares two different versions of the same layout database.

Valid Files

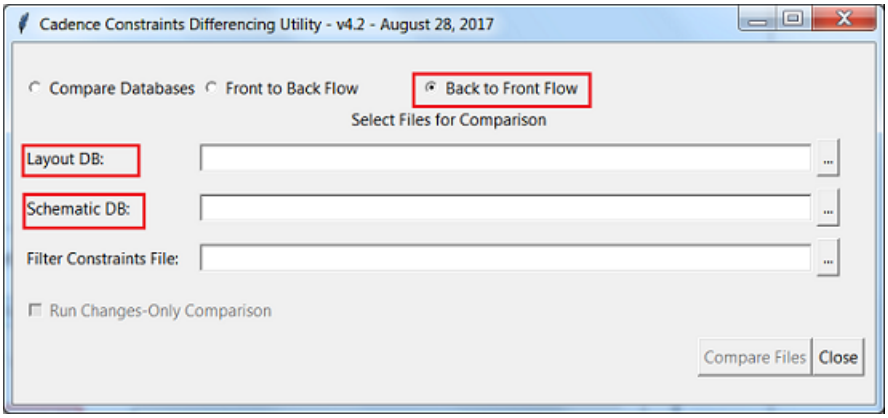
- Constraints Files: `*.dcfx, *.dcf, *.tcfx, *.tcf`
- Design Files: `*.brd, *.sip, *.mcm`
- Schematic Projects: `*.cpm`

Comparing Schematic Versus Layout Databases

- Front to back flow: Compares schematic database with the layout database. Valid files are: `*.cpm, *.brd, *.sip,` and `*.mcm.`



- Back to front flow: Compares layout database with the schematic database. Valid files are: `*.cpm, *.brd, *.sip,` and `*.mcm`.



Filter Constraints File

To control the display of constraint information in the report, choose a filter file. A filter file is a text file that contains the names of properties that you define either to display or to ignore from the difference report. The name of properties should be the same as specified in Constraint Manager dictionary.

You can specify the filter files in two ways:

- **Blacklist file**: This file contains a list of properties that are not included in the difference report. The file is considered as blacklist if the first line of a file is set as *!blacklist*.
- **Whitelist file**: This file contains a list of properties that should be included in the difference report. The file is considered as whitelist if the first line of a file is set as *!whitelist*.

Important

If not specified, the filter file is set as a whitelist filter file. Additionally, if more than one values are set the last one is considered.

Sample Filter File

The following is an example of a blacklist file:

```
!blacklist
STUB_LENGTH
```

For adding comments in the filter file using # symbol.

Important

Using filter file may hide an important information. Use the filter file only when two databases are compared. Avoid using filter file during the front-to-back or back-to-front flow.

You can either navigate to the filter file location, or it will automatically appear if any of the following (options) are set:

- an environment variable *CM_DIFF_UTILITY_PROP_FILTER_FILE* with name and path of filter file.
- an environment variable *CM_DIFF_UTILITY_PROP_FILTER_FILE* with name of filter file. The file will be searched at following locations:
  - $HOME/cdssetup
  - $CDS_SITE/cdssetup
- a *propFilter.txt* exists at following locations:
  - $HOME/cdssetup
  - $CDS_SITE/cdssetup
- a .cpm directive CM_DIFF_UTILITY_PROP_FILTER_FILE is set in the CONSTRAINT_MANAGER section of the .cpm file.

  You can define this directive either at site level or project level.

Changes-only comparison

When a schematic database is compared with a layout database and the *Run Changes-Only Comparison* option is enabled, the comparison is performed in changes-only mode to verify front-to-back or back-to-front flow and a report is generated.

In changes-only mode, if the constraint value has been changed by the user in the destination database after the last flow, it will be preserved.

Procedure

To compare the constraints, complete the following steps:

1. Run `cmDiffUtility` command from the command prompt.

The *Cadence Constraints Differencing Utility* dialog box is displayed.

2. Choose *First File.* Browse to select the first database.
3. Choose *Second File.* Browse to select the second database.
4. Optionally, choose a *Filter Constraint File*.
5. Optionally, enable *Run Changes-Only Comparison* if a logical database is compared with a physical database.
6. Click *Compare Files.*

   A report is generated as a result of comparison.

Comparing Constraints using PCB Editor

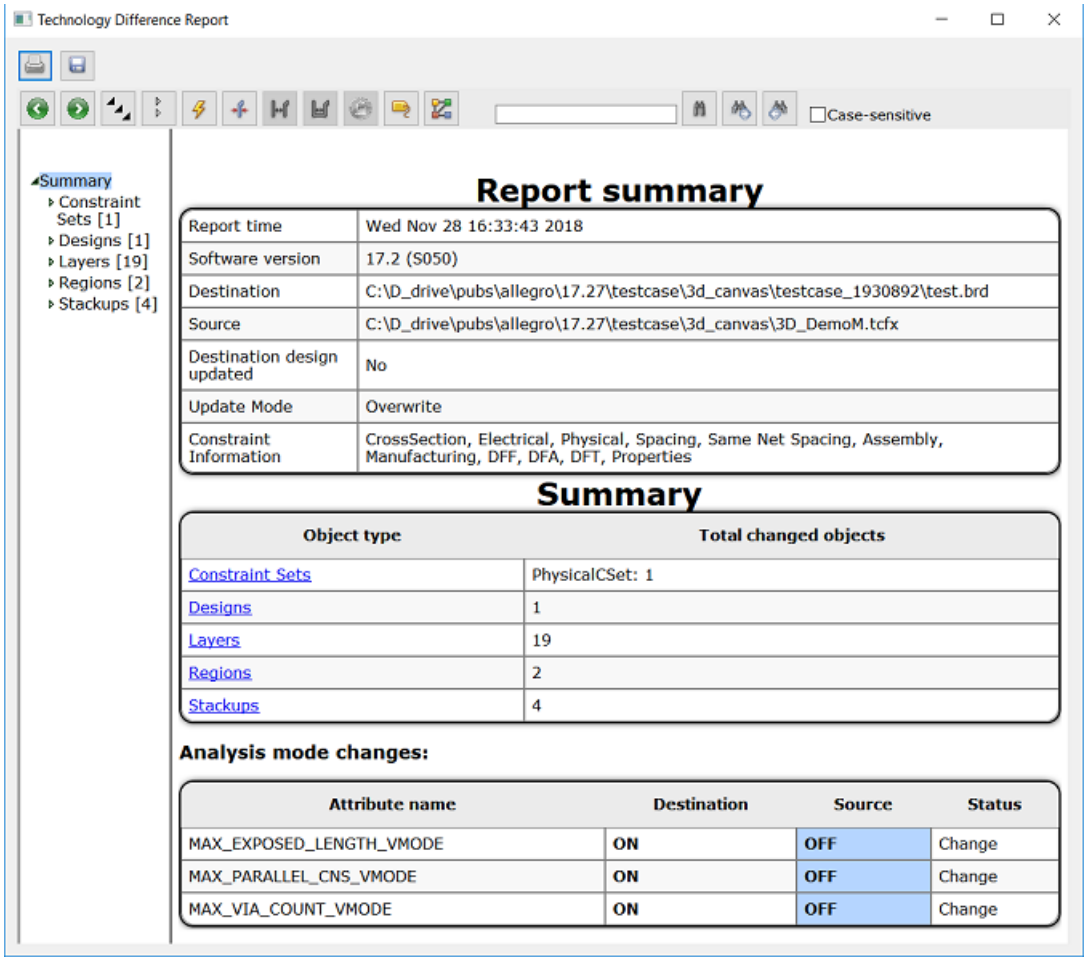
You can compare constraint values from technology files in PCB Editor.

To generate the constraint differences, complete the following steps:

1. Choose *File – Import – Constraints* or *File – Import – Technology file.*

   The *Import Constraints* or *Import a technology file* dialog box is displayed.
2. Select *Report Only* check-box for generating the constraint difference report.
3. Choose a `.dcf/.tcf` file to compare.

A progress bar is displayed during report generation. Once the process is over a Constraints Difference Report is displayed.



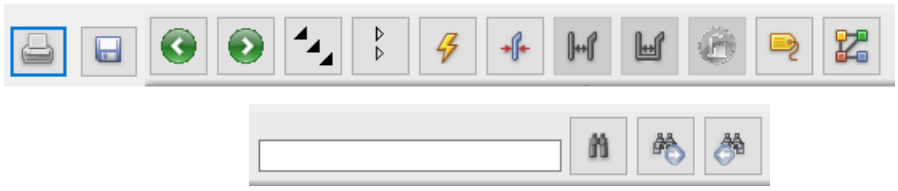
Viewing Constraints Differences Report

The report is displayed in a built-in report viewer.

The report viewer supports a simple, intuitive graphical user interface for displaying constraint differences between the two databases.This viewer consists of a toolbar, and view window.

Report Viewer Toolbar

The following figure displays the report viewer toolbars.



The following table describes the function of each button:

Table 8-1 Constraint Difference Report: Toolbar Description

Name	Icon	Function
Previous Selection		shows previous view of the report
Next Selection		shows next view of the report
Export report to HTML		export report as HTML page
Print Preview		shows printer-friendly view of the report
Expand all		Expands all rows and columns
Collapse all		Collapse all rows and columns

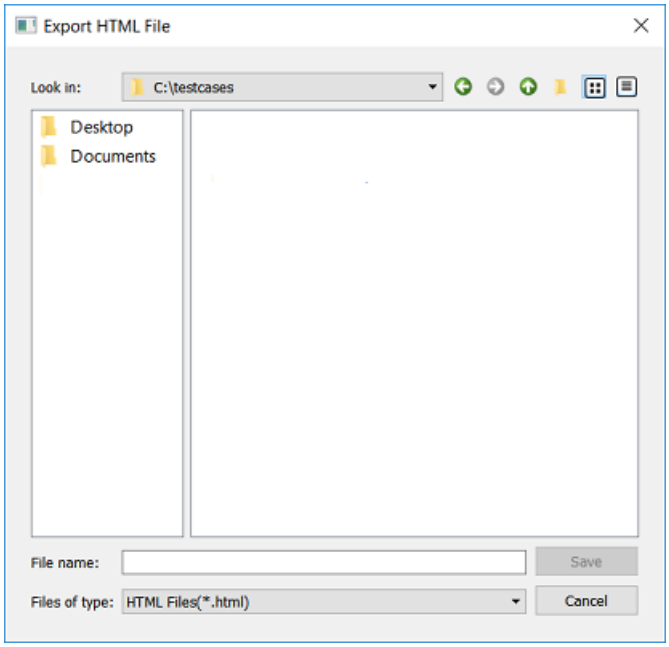
Name	Icon	Function
Toggle Electrical constraint changes		Displays only those Electrical Constraints Sets for which there is change in constraint values.
Toggle Physical constraint changes		Displays only those Physical Constraints Sets for which there is change in constraint values.
Toggle Spacing constraint changes		Displays only those Spacing Constraints Sets for which there is change in constraint values.
Toggle Same Net Spacing constraint changes		Displays only those Same Net Spacing constraints for which there is change in constraint values.
Toggle Design For Fabrication changes		Displays only those Design for Fabrication Constraints Sets for which there is change in constraint values.
Toggle Property changes		Displays only those properties for which there is change in constraint value.
Toggle Structural changes		Displays only those grouping objects (differential pairs, match groups, net groups, and so on) for which there are change in constraint value.
Find		Display search result by highlighting it.
Find Next Object		Display next object from the search results.
Find Previous Object		Display previous object from the search results.

Exporting Report as an HTML Page

You can export the constraint difference report as an HTML page for sharing.

1. Choose *Export report to HTML* from toolbar.

The *Export HTML File* dialog is displayed.



2. Browse the directory to save the report.
3. Specify a name for the HTML file.
4. Click the *Save* button.

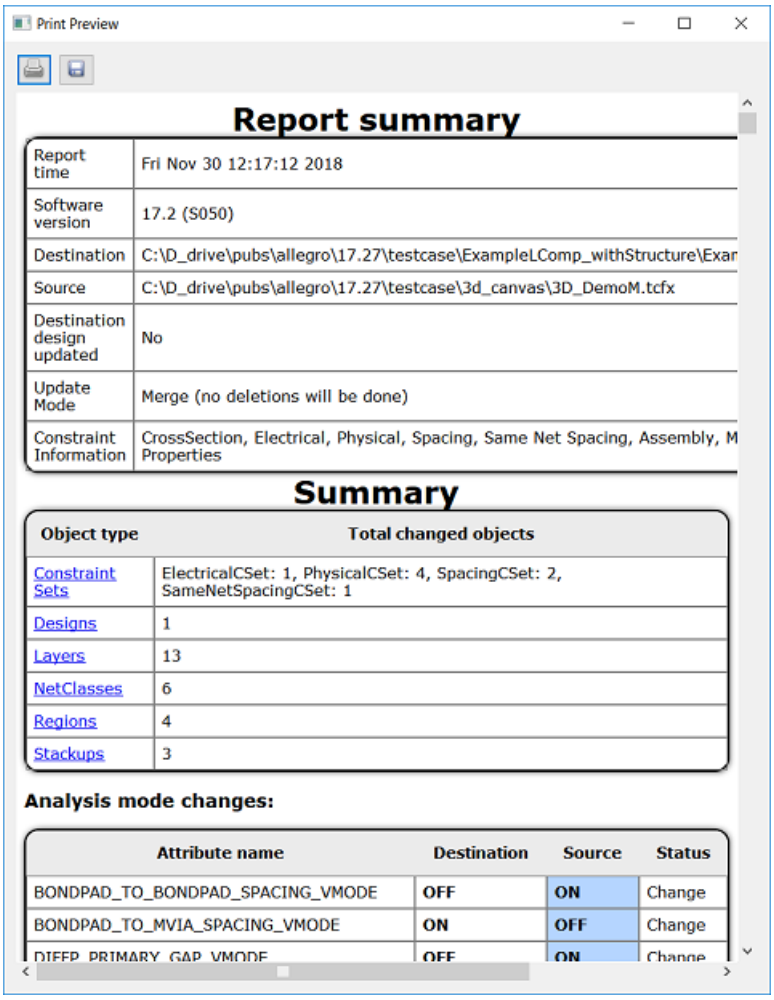
   The HTML page of the report is generated and can be opened in the HTML report page in any browser.

Printing Report

To print the report, complete the following steps:

1. Choose *Print Preview* icon.

A single-page view is displayed for reviewing.



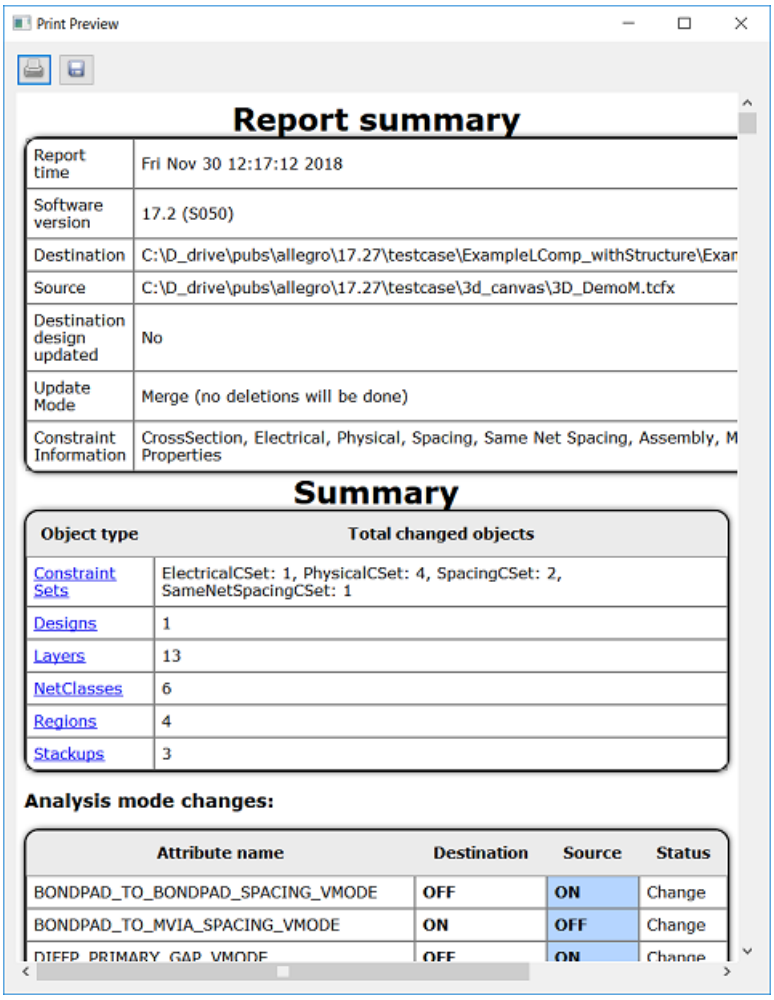
2. Choose *Print* and specify the printer details for printing the report.

Saving Report as a Single File

To save the complete report, complete the following steps:

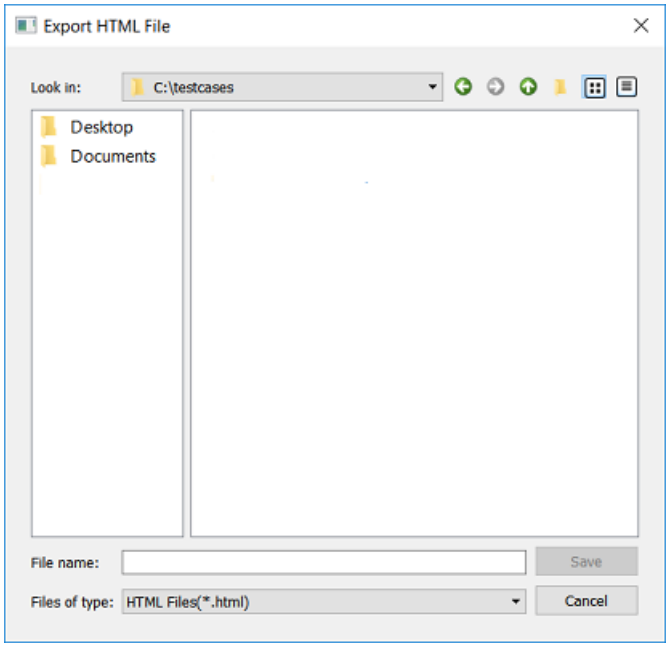
1. Choose *Print Preview* icon.

A single-page view is displayed for reviewing.



2. In the *Print Preview* window, click *Export report to HTML* icon.

The *Export HTML File* dialog is displayed.



3. Browse the directory to save the report.
4. Specify a name for the HTML file.
5. Click the *Save* button.

   A single-page report is generated that can be opened in any browser.

Viewing Reports on Web Server

You can generate multiple reports and save them on a server that can be accessed anytime using a web browser.

To generate constraints difference reports stored on a web server, set an environment variable `CM_VDD_REPORT_URI_OVERRIDE`. This variable provides a Uniform Resource Identifier (URI) to specify the path of the report directory. For example, *http://<server_name>/VDD/report*, where you can copy the VDD/report directory.

By default, the value of the variable is mapped to the path of report directory that exist in the installation directory, `<installation_directory>\share\pcb\consmgr\VDD\report`.

To view an XML version of the report in Firefox, disable URI and enter the following command in the system command window:

```
firefox -app
     <installation_directory>\share\pcb\consmgr\VDD\diff3rptViewer\diff3rptViewer
     .ini -file file://<path to rpt>.xml
```

When using Chrome, use *--allow-file-access-from-files* option to view the XML file.

```
chrome --allow-file-access-from-files file://<path to rpt>.xml
```

In Internet Explorer version 9, XML file can be directly opened for viewing.

Note: Reports already saved on the disk can also be viewed using XULRunner utility.

Interpreting Constraint Differences

The report viewer is a two-frame, tree-view based window.

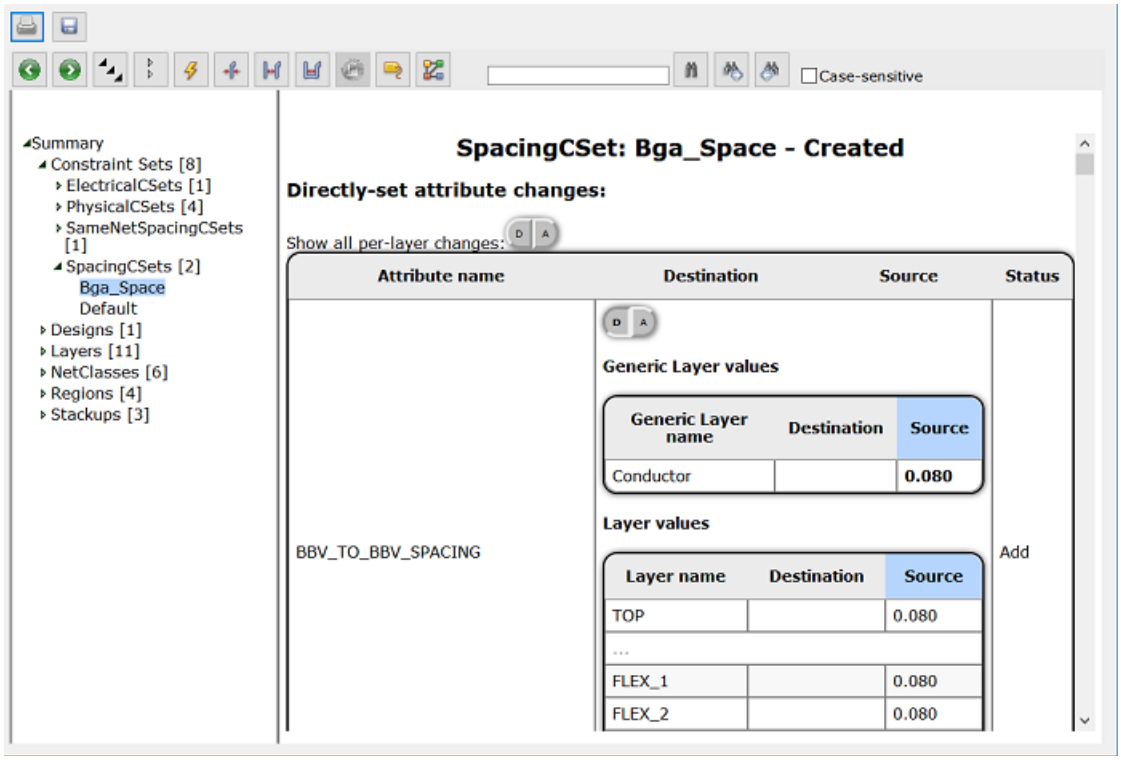
Navigating Report

You can navigate the report using following modes:

Navigation Mode	Function
By Tree-view	shows summary of the selected node in the tree-view
By following Links	shows summary of an object when selected the object name link
By previous and next selections	show selections made forward and backward in the report viewer

Viewing Differences

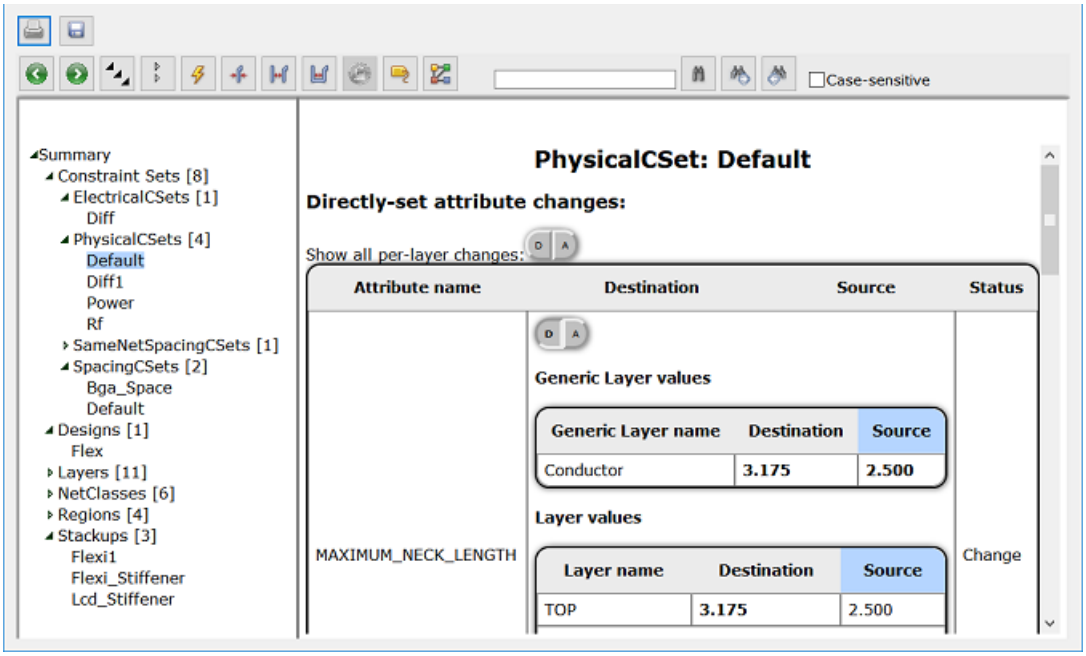
The Constraint Difference Report for a single layer is displayed in the following figure:



Viewing Modes

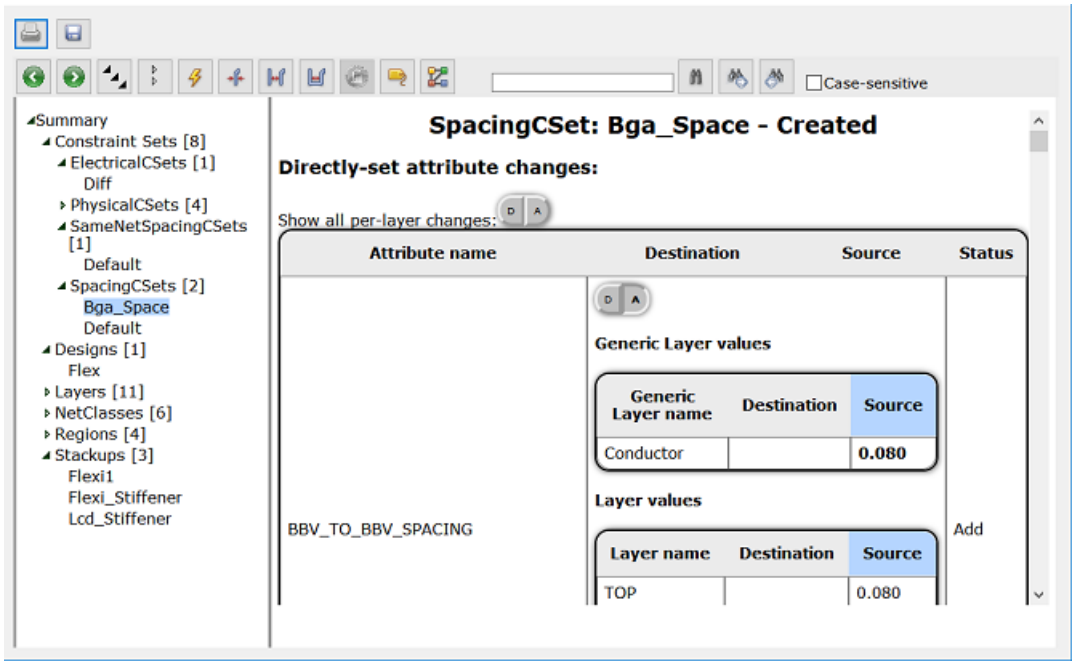
The following viewing modes are available in the viewer:

- Differences only(D): Shows differences only between destination and source constraints values.



Note: Unchanged layers are not reported.

- All values(A): Shows constraint values at destination and source for all layers.



9

Inter Layer Spacing Checks

Topics in this chapter include:

What are Inter Layer Checks?

In standard PCB designs various masks and surface finishes require verification of proper clearances and coverage. Rigid Flex designs not only have the same mask and surface finish requirements, but also have additional geometries such as bend areas and stiffeners. These geometries, present on different layers, require verification of special clearances or overlaps of materials, and spacing between these layers.

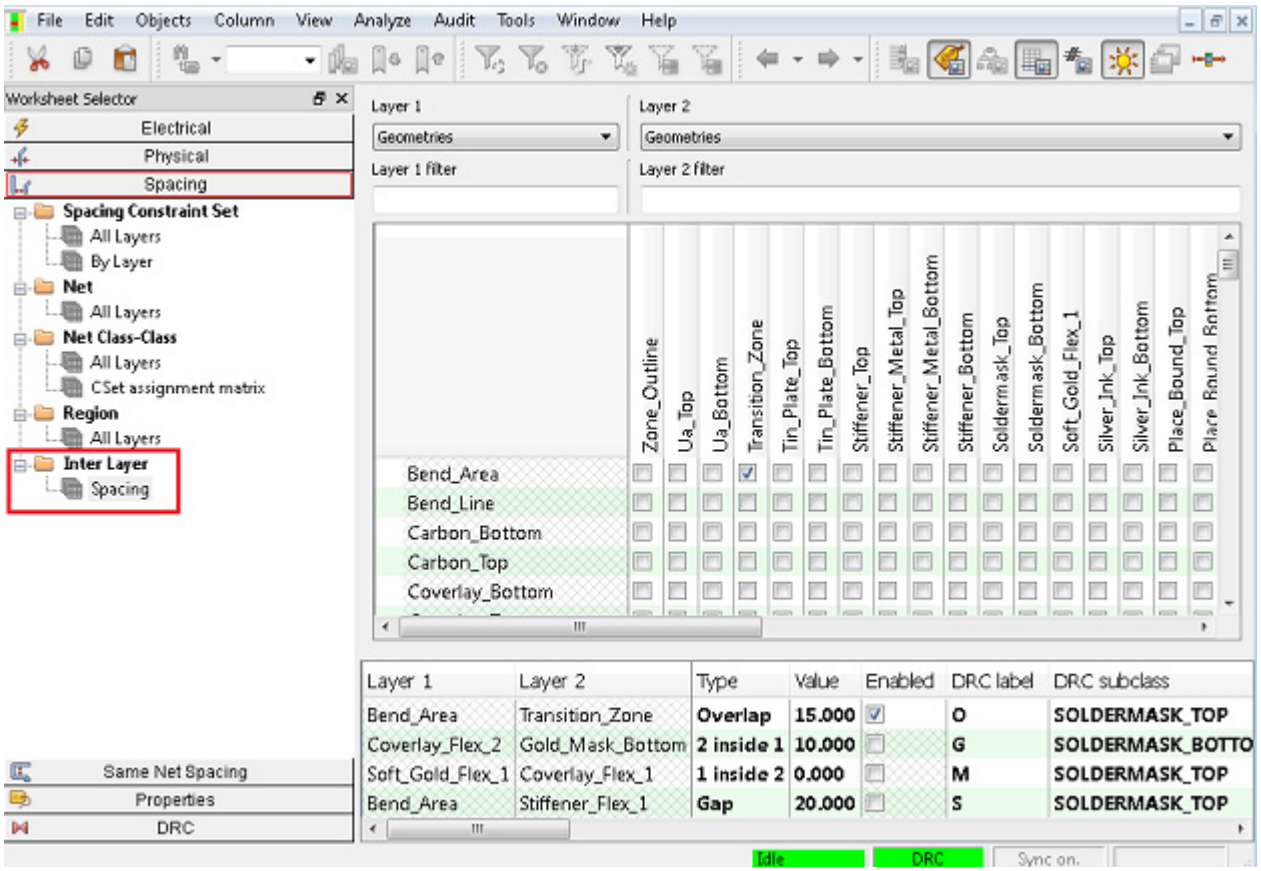
The Inter Layer checks provides spacing checks between objects of one layer to those on another layer. These checks are usually defined for Rigid Flex designs can also be used in standard single or multi-layer designs.

To create Inter Layer checks you need to select two subclasses, set rule type, value, DRC label and DRC layer display. You can view the effects of these rules in the Allegro PCB Editor.

Inter Layer Spacing Workbook

In Spacing domain, an *Inter Layer Spacing* workbook provides a matrix to select subclasses, types of constraints and their values for defining Inter Layer spacing checks. For example, you can create a check to verify spacing between conductor etch to (non-etch/conductor) shapes on any of the supported subclasses.

You can also import and export inter layer spacing constraints defined for different subclasses in the Inter Layer Spacing workbook.



The Inter Layer Spacing worksheet has two resizable panes:

- Layer pair management pane: located at the top for adding/deleting the layer pairs.
- Constraints pane: located at the bottom for editing the constraints on the existing layer pairs.

Layer Pair Management Pane

Consists of two columns in the top, and a matrix with checkboxes for creating or deleting layer pairs in the bottom. The left column lists eligible subclasses labeled as *Layer 1*. The right column is labeled as *Layer 2*.

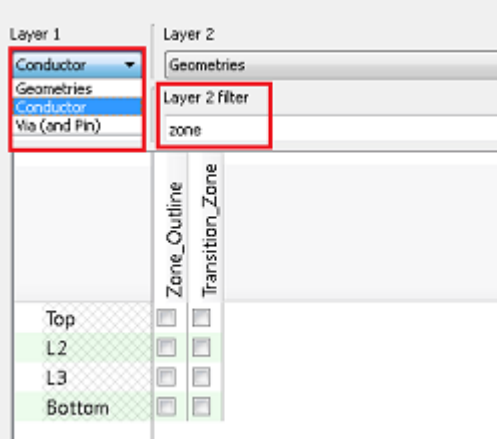
You can choose following types of subclasses for Inter Layer checking:

Subclass Types	Example
Conductor Layers	Place Bound (Top/Embedded/Bottom)
Pin/Via Layers	Filmmask (Top/Bottom)
All Mask Layers	Soldermask(Top/Bottom)
Rigid Flex Subclasses	Pastemask(Top/Embedded/Bottom)
Surface Finishes Subclasses	User-defined Subclasses

Other classes, subclasses, and objects that are not included in the inter layer checks are:

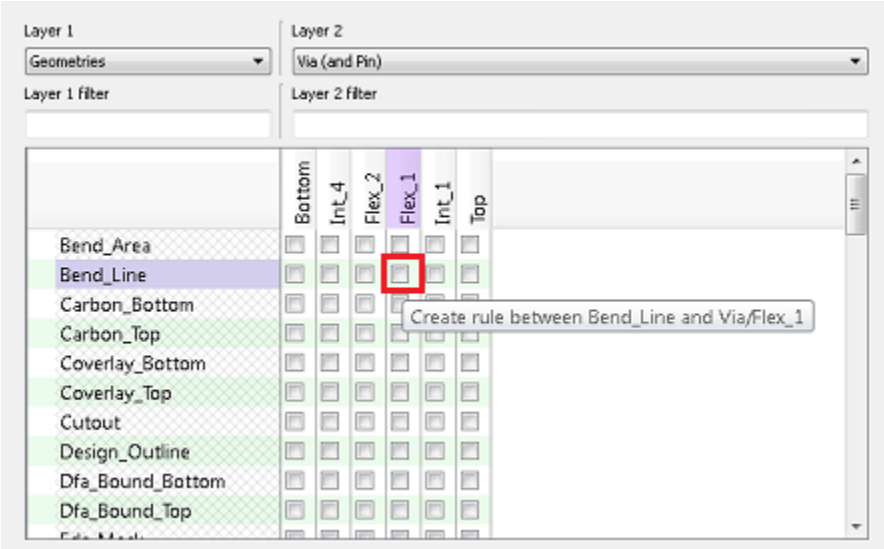
- drawing format
- analysis
- DRC
- text (on any subclass)
- board geometry outline
- silkscreen layers
- stackup named dielectric layers
- manufacturing
- constraint regions
- all keepin and keep outs
- component value
- device type
- Ref Des
- tolerance
- user part number

Row and column filters are available with the columns to search subclasses and subclass types.

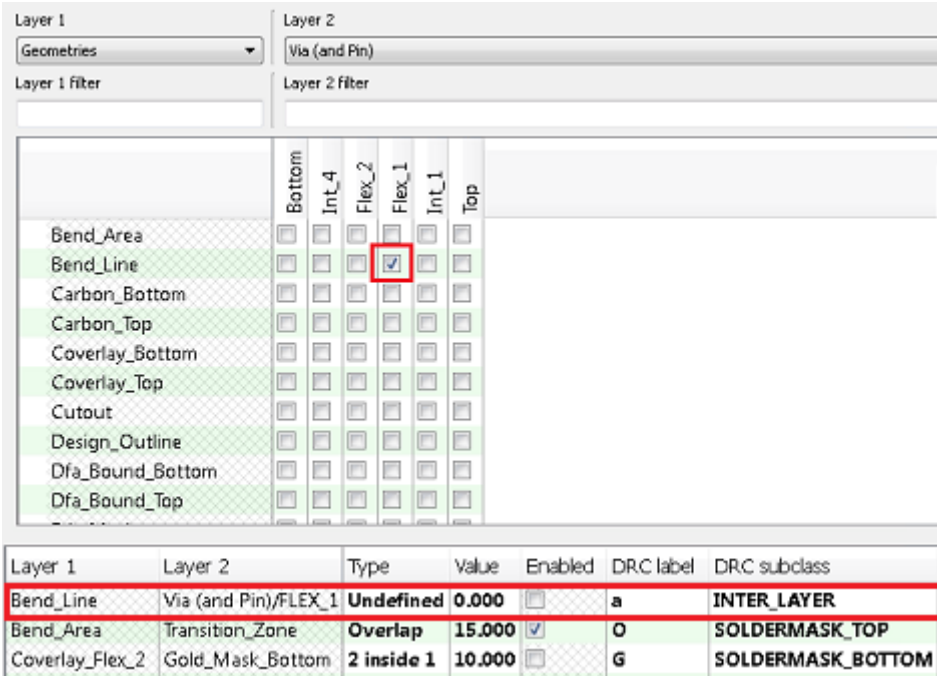


Defining Constraints for Inter Layer Checks

To create a constraint between two subclasses, enable the checkbox where the two subclasses intersects in the matrix. Hovering over the checkbox highlights the row and column headers and a tooltip displays the layer pair name.

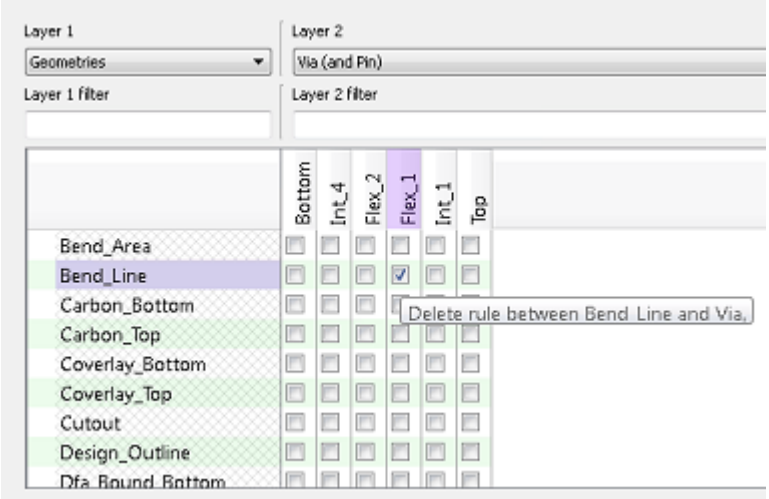


On enabling the checkbox, a new row is added at the top of the constraint table.



Set the values for type, value, DRC label, and then enable the check.

To delete a rule deselect the checkbox in the selection matrix or select “X” in the *Delete* column of the constraint table for that row.



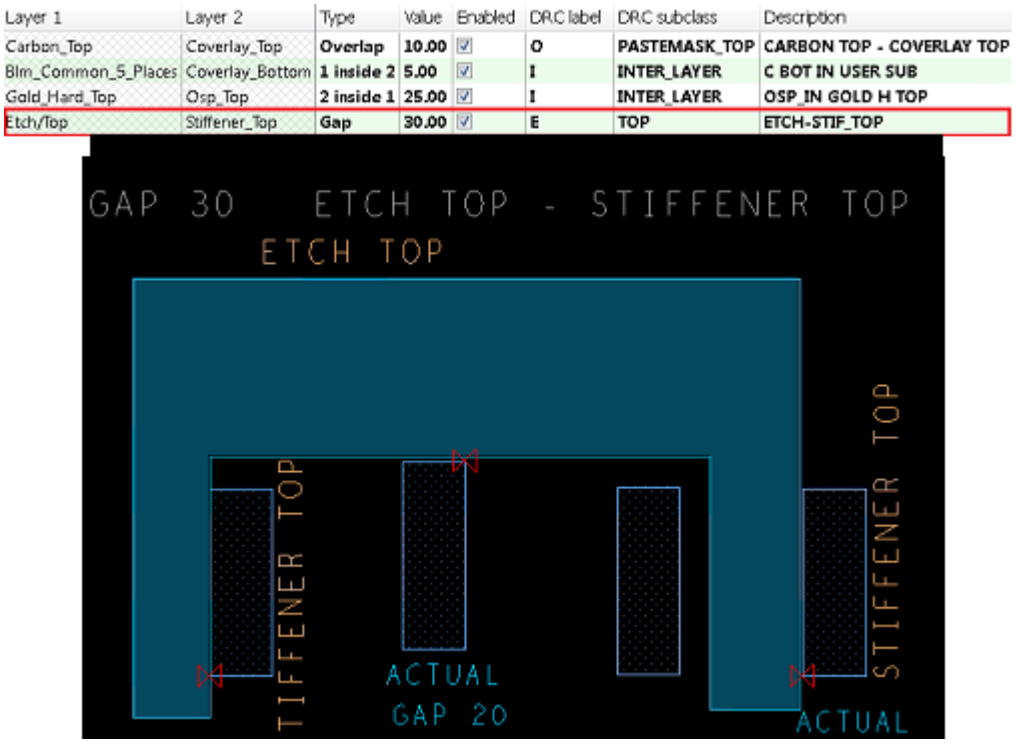
Constraints Pane

The constraint pane displays a table at the bottom of the Inter Layer Spacing worksheet.

Layer 1	Layer 2	Type	Value	Enabled	DRC label	DRC subclass	Description	Delete
Carbon_Top	Coverlay_Top	**Overlap**	**10.00**	☑	**O**	**PASTEMASK_TOP**	**CARBON TOP - COVERLAY TOP**	✖
Blm_Common_5_Places	Coverlay_Bottom	**1 inside 2**	**5.00**	☑	**I**	**INTER_LAYER**	**C BOT IN USER SUB**	✖
Gold_Hard_Top	Osp_Top	**2 inside 1**	**25.00**	☑	**I**	**INTER_LAYER**	**OSP_IN GOLD H TOP**	✖
Etch/Top	Stiffener_Top	**Gap**	**30.00**	☑	**E**	**TOP**	**ETCH-STIF_TOP**	✖

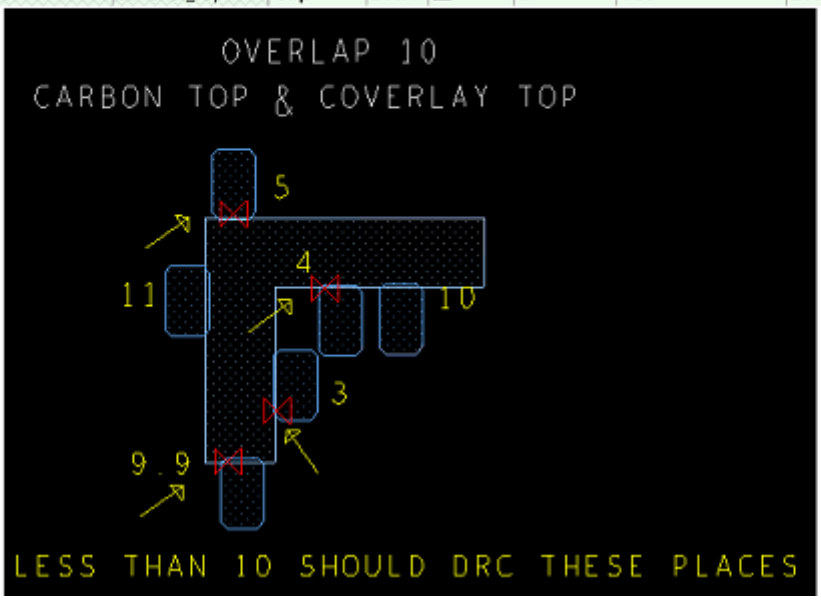
The table includes following fields:

- Layer 1: is a read-only field that displays the name of the subclass selected as Layer 1.
- Layer 2: is a read-only field that displays the name of the subclass selected as Layer 2.
- Type: defines type of spacing checks between subclasses. Four types of inter layer spacing checks are supported.
  - Gap: specifies a minimum spacing value between two objects on the selected subclasses.



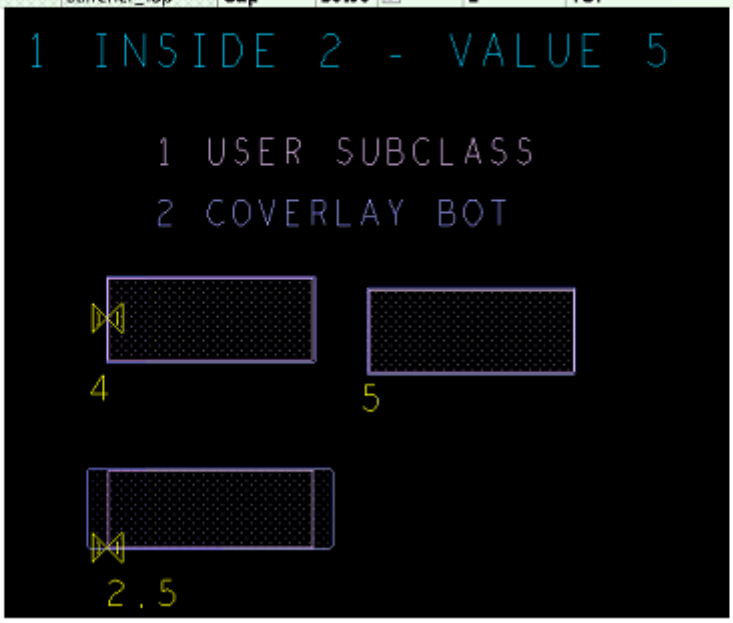
- Overlap: specifies a minimum overlap value between two objects on the selected subclasses.

Layer 1	Layer 2	Type	Value	Enabled	DRC label	DRC subclass	Description
Carbon_Top	Coverlay_Top	**Overlap**	**10.00**	☑	**O**	**PASTEMASK_TOP**	**CARBON TOP - COVERLAY TOP**
Blm_Common_5_Places	Coverlay_Bottom	**1 inside 2**	**5.00**	☑	**I**	**INTER_LAYER**	**C BOT IN USER SUB**
Gold_Hard_Top	Osp_Top	**2 inside 1**	**25.00**	☑	**I**	**INTER_LAYER**	**OSP_IN GOLD H TOP**
Etch/Top	Stiffener_Top	**Gap**	**30.00**	☑	**E**	**TOP**	**ETCH-STIF_TOP**



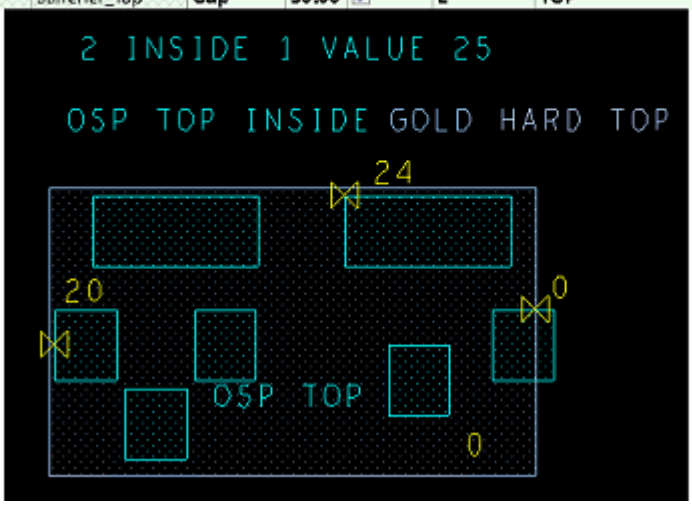
- 1 inside 2: The geometry on the subclass defined as Layer 1 must be contained within a geometry on the subclass defined as Layer 2.

Layer 1	Layer 2	Type	Value	Enabled	DRC label	DRC subclass	Description
Carbon_Top	Coverlay_Top	**Overlap**	**10.00**	☑	**O**	**PASTEMASK_TOP**	**CARBON TOP - COVERLAY TOP**
Blm_Common_5_Places	Coverlay_Bottom	**1 inside 2**	**5.00**	☑	**I**	**INTER_LAYER**	**C BOT IN USER SUB**
Gold_Hard_Top	Osp_Top	**2 inside 1**	**25.00**	☑	**I**	**INTER_LAYER**	**OSP_IN GOLD H TOP**
Etch/Top	Stiffener_Top	**Gap**	**30.00**	☑	**E**	**TOP**	**ETCH-STIF_TOP**



- 2 inside 1: The geometry on the subclass defined as Layer 2 must be contained within a geometry on the subclass defined as Layer 1.

Layer 1	Layer 2	Type	Value	Enabled	DRC label	DRC subclass	Description
Carbon_Top	Coverlay_Top	Overlap	10.00	☑	O	PASTEMASK_TOP	CARBON TOP - COVERLAY TOP
Blm_Common_5_Places	Coverlay_Bottom	1 inside 2	5.00	☑	I	INTER_LAYER	C BOT IN USER SUB
Gold_Hard_Top	Osp_Top	2 inside 1	25.00	☑	I	INTER_LAYER	OSP_IN GOLD H TOP
Etch/Top	Stiffener_Top	Gap	30.00	☑	E	TOP	ETCH-STIF_TOP

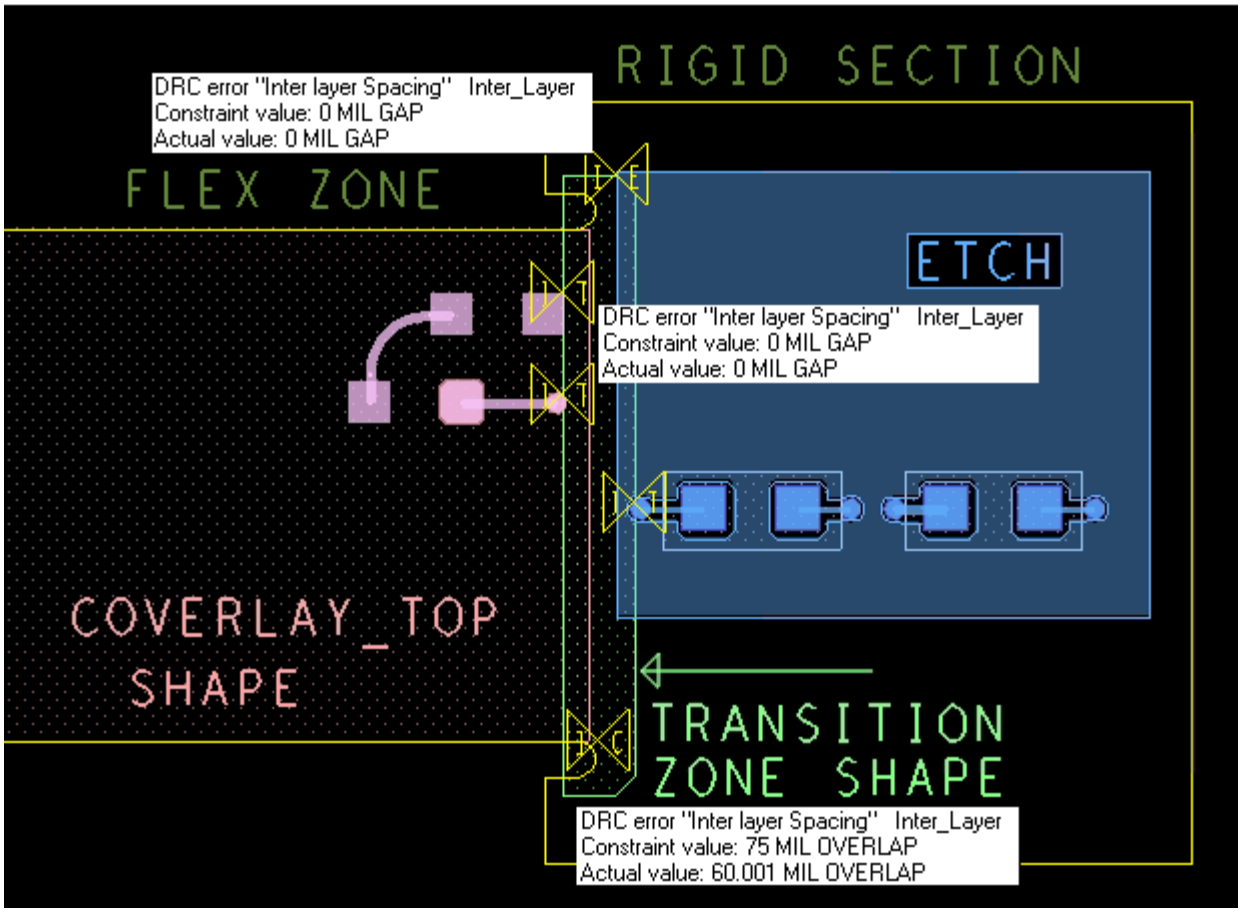


- Value: specifies the spacing dimension in design units.
- Enabled: allows spacing check for selected layer pair. Enabling any of the check sets the inter layer checks to *On* globally.
- DRC Label: specifies a user-defined DRC marker label for the second character of the DRC. The first character “I” is reserved for Inter Layer check. The second character may be any single character a-z, A-z, 0-9. You can use any special characters. For example, I-[a-z], I-[A-Z], and so on. You can also use duplicate letters.
- DRC Subclass: defines the display subclass for the DRC marker. A pull-down menu lists the available subclasses.
- Description: allows you to add comment or description for reference.
- Delete: removes the selected entry from the table.

Example in Rigid-flex Design

The following image illustrates a part of rigid flex design in which a transition zone shape is created to verify the following design requirements:

- vias and pins are not too close to the edge of the stackup
- two etch subclasses on the rigid side of a stackup change to the transition zone
- coverlay extends beyond the zone boundary



To verify these checks, following inter layer spacing checks are defined in the constraints table:

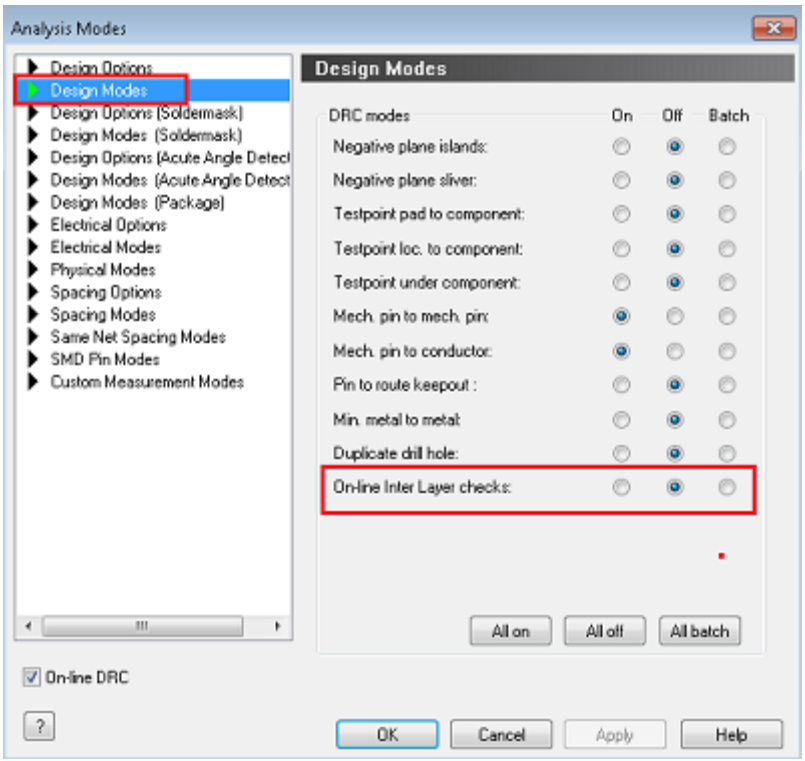
- gap between transition zone to via/pin
- gap between transition zone to etch

- overlap between transition zone to coverlay

Layer 1	Layer 2	Type	Value	Enabled	DRC label	DRC subclass	Description	Delete
Transition_Zone	Via (and Pin)/TOP	**Gap**	**0.000**	☑	**T**	**INTER_LAYER**		✖
Transition_Zone	Via (and Pin)/2-RIGID	**Gap**	**0.000**	☑	**T**	**INTER_LAYER**		✖
Transition_Zone	Via (and Pin)/3-FLEX	**Gap**	**0.000**	☑	**T**	**INTER_LAYER**		✖
Transition_Zone	Via (and Pin)/4-FLEX	**Gap**	**0.000**	☑	**T**	**INTER_LAYER**		✖
Transition_Zone	Via (and Pin)/5-RIGID	**Gap**	**0.000**	☑	**T**	**INTER_LAYER**		✖
Transition_Zone	Etch/Top	**Gap**	**0.000**	☑	**E**	**INTER_LAYER**		✖
Transition_Zone	Coverlay_Top	**Overlap**	**75.000**	☑	**C**	**INTER_LAYER**		✖
Transition_Zone	Via (and Pin)/BOTTOM	**Gap**	**0.000**	☑	**T**	**INTER_LAYER**		✖

Enabling On-line Inter Layer Checking

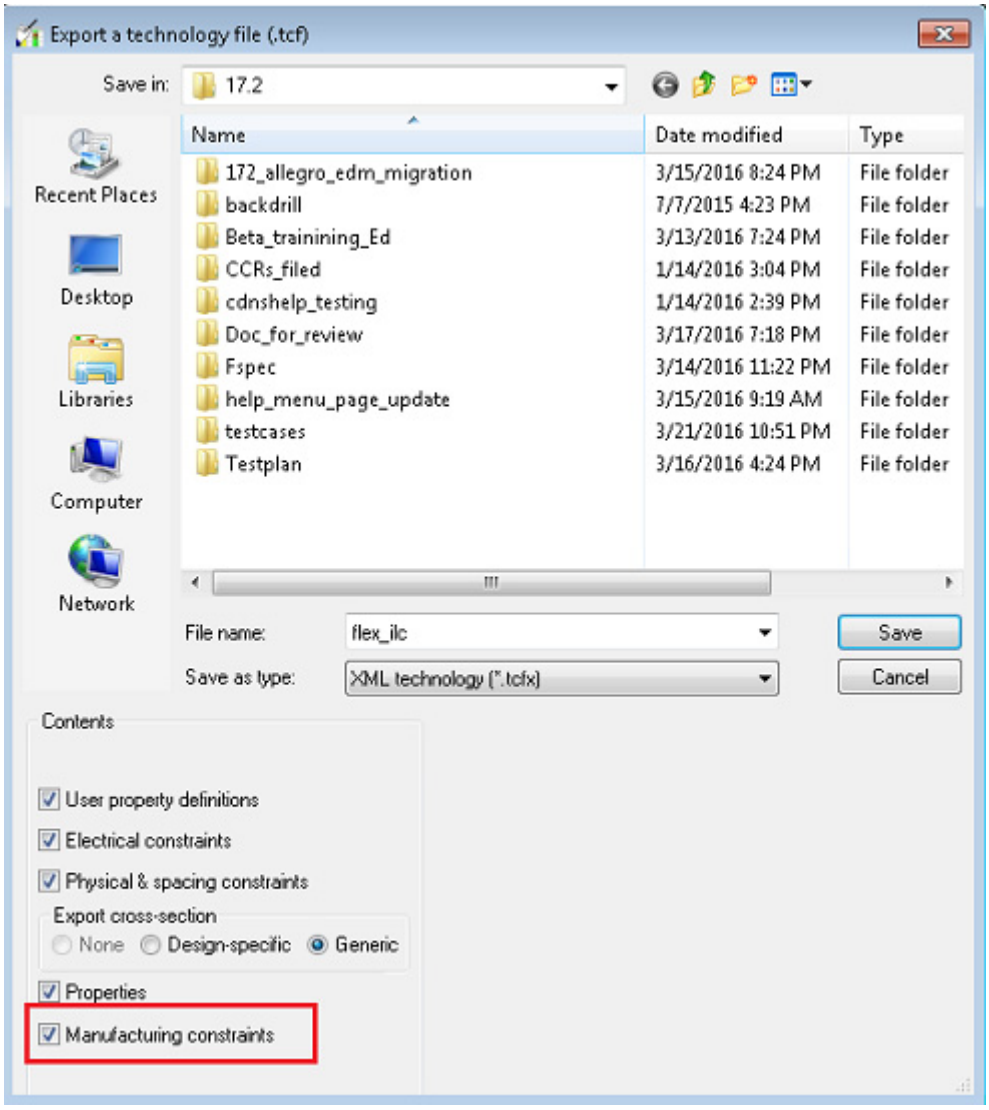
By default, the on-line Inter Layer checks are set to off in the PCB Editor. To enable, set the constraint mode for Inter Layer Checks in the *Design Modes* tab in the *Analysis Modes* dialog box.



Exporting/Importing Inter Layer Checks

Exporting Checks

On exporting techfile(`.tcfx`) or constraints (`.dcfx`) file, inter layer checks are also included if *Manufacturing constraints* option is enabled.



The exported technology file contains only inter layer spacing constraint information required for manufacturing and does not include any stackup information.

Importing Checks

Importing a techfile (.tcfx) or constraints (.dcfx) file with inter layer spacing checks in a design does not create any new subclasses. However, the inter layer checks for non-existing subclasses are imported.

The DRC subclasses not present in the destination design are mapped to default subclass (Inter_layer) after import. You cannot enable constraints that are referenced to subclasses that are not present in the design.

Limitations

You cannot create Inter layer checks between:

- etch layers, for example Top etch to Bottom etch.
- same layers, for example Coverlay_top to Coverlay_top.

10

Constraint Compiler

Constraint Compiler is a mechanism for translating design constraints from an external source directly into Constraint Manager. Constraint Compiler introduces constraints per manufacturer guidelines at the interface level in a design. The compiler can insert initial constraint or update the existing constraint information in a design. To create specific rules for various interfaces in a design, Constraint Compiler uses connectivity information (buses, differential pairs, nets, and so on) of a design in conjunction with data agnostic constraint information provided by the manufacturer.

Constraint Compiler is available with the following product licenses:

- Allegro PCB Designer with High Speed Option
- Allegro Venture PCB Designer Suite
- Allegro Enterprise PCB Designer Suite
- Allegro X Advanced Package Designer (Packaging)
- Allegro Design Authoring (Schematic)

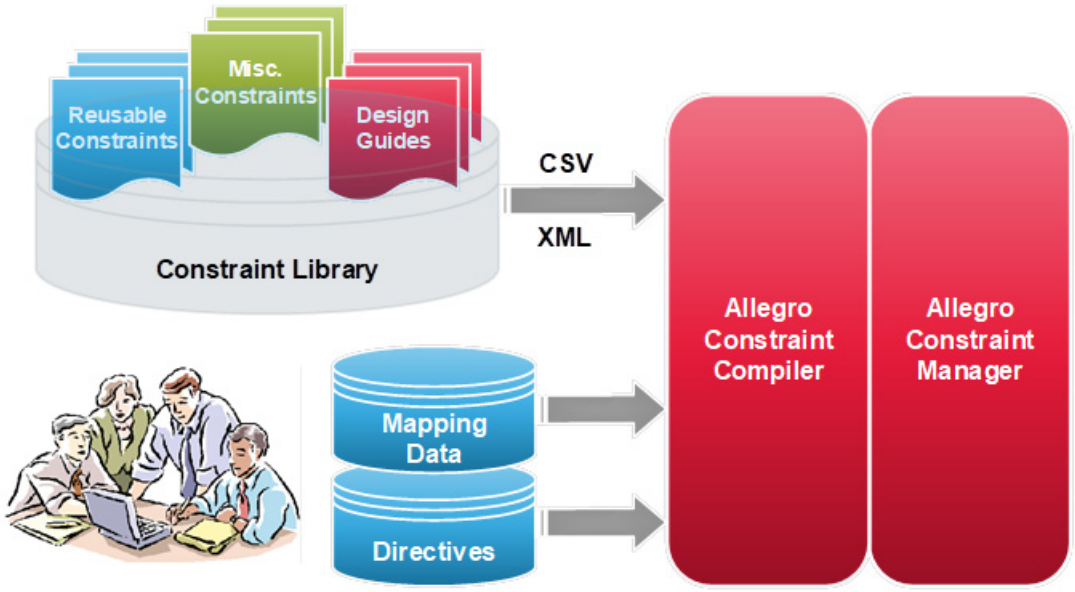
Need for Constraint Compiler

Various manufactures provide EDA design guides to ensure that the customers leverage the technology to its fullest extent and be able to develop products that perform as expected. These design guides are normally specified at interface level. The PCB designer interprets these guidelines and apply appropriate rules in the layout tool. Incorrect understanding of these rules leads to under or over constraining of the design. To understand correct requirements for constraints, consulting reference designs and taking design review services helps, but it increases the design schedule considerably, which could impact the overall product schedule. Constraint Compiler is devised to facilitate designers by translating the manufacturer’s design guides automatically into Constraint Manager.

Overview

The power of Constraint Compiler is the ability to leverage data-agnostic constraints that can be developed once, validated and placed in a central library for future use in other designs. All designs are not exactly alike and may have slightly different net names or component reference designators. Constraint Compiler acknowledge this fact and provides a mapping table to correlate constraint and design-specific information and generates a standard Constraint Manager difference report to review changes that will be made to the design. You can run the compiler in either validation mode (report-only mode) or apply mode to incorporate the changes to the design with compiler options to merge or replace existing constraints.

The following illustration shows an overview of process flow for Constraint Compiler.



Design guides once transposed into the agnostic format can be stored in the constraint library for use in future designs. The data tables in the constraint library should be structured to be generic as possible to allow it to be leveraged with any design. Any design-specific objects should be stored as alias variables that can be updated through a mapping data table on a per design basis. Special table keys can be entered in the constraint library data table for filtering tables using the query functions of the compiler.

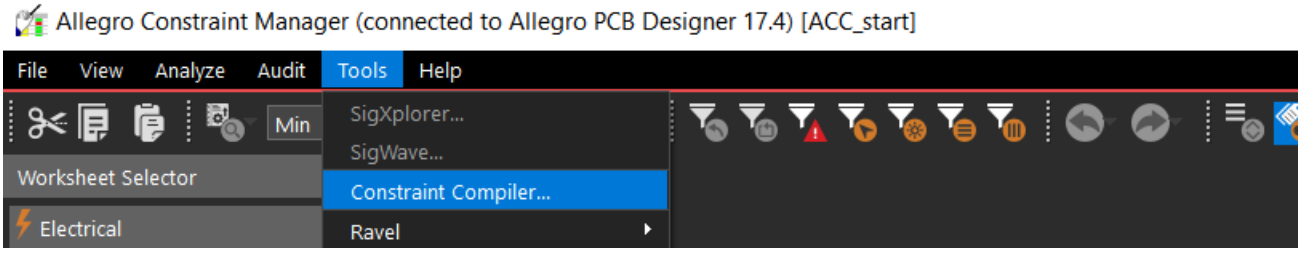
Advantages of Constraining Designs Using Constraint Compiler

The Constraint Compiler is capable of redirecting constraint information with absolutely no manual intervention and without any errors into a design.

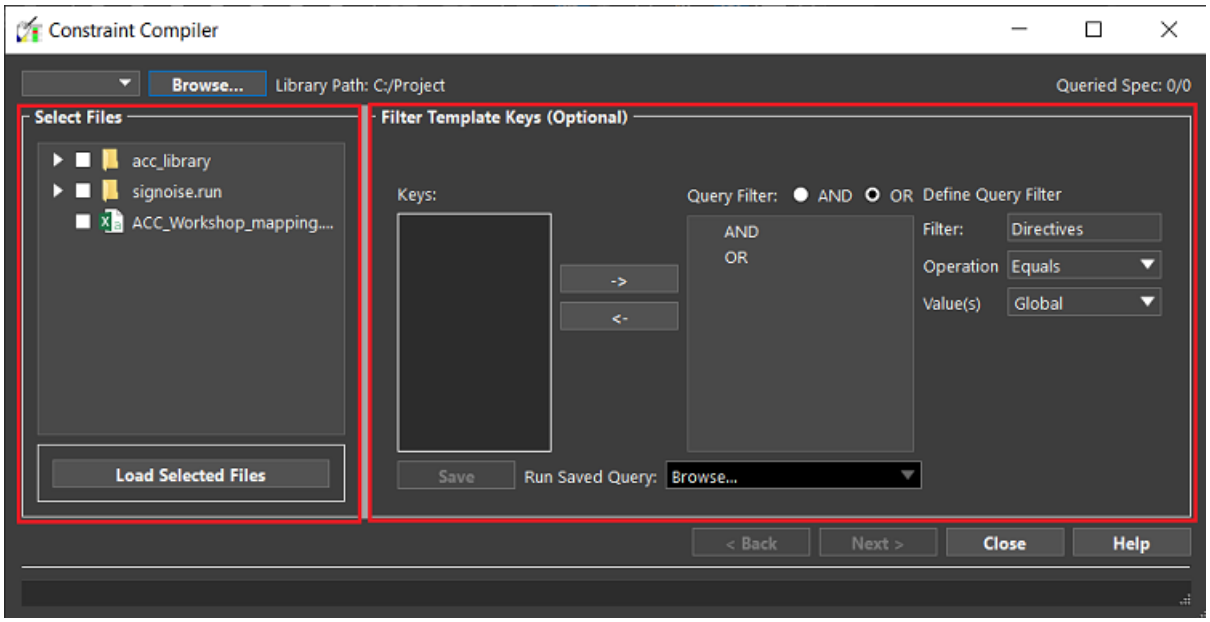
- Provides quick entry of constraints without having advanced knowledge of Constraint Manager
- Provides a way for manufacturers (silicon vendors) to develop specification tables to supplement their design guides
- Constraint management can be done at the abstract level instead of the design level
- Constraint information is easily transferred using a single source into multiple designs. This process ensures consistent constraint data in multiple designs and does not require export or import of technology files (`.dcfx`, `.tcfx`)
- Facilitates automatic generation of constraint objects (Differential pairs, Net Classes, Net Groups, Match Groups and expanded Constraint Sets)
- Constraints specified at top of the object hierarchy are propagated down to the lower-level objects without any effort
- Version controls of golden library constraints

Constraint Compiler User Interface

In Constraint Manager, choose *Tools – Constraint Compiler* to open Allegro Constraint Compiler.



The Constraint Compiler has two main sections:



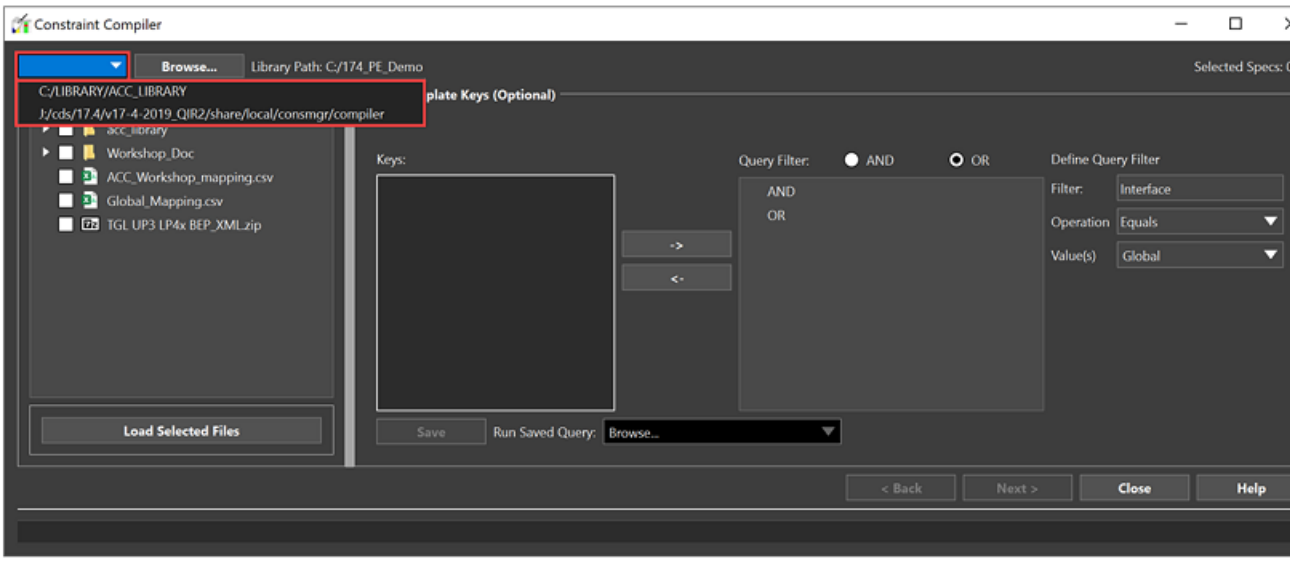
Library Browser

The left section represents constraint libraries. Each library consists of constraints and mapping data in form of tables, which are saved in `.csv` format. The data tables captures constraint information in form of key/value pairs and is used when searching a library to find data for compiling. The data tables are generic in structure and can be leveraged with any design.

For information on how to create and use data tables, see Data Tables

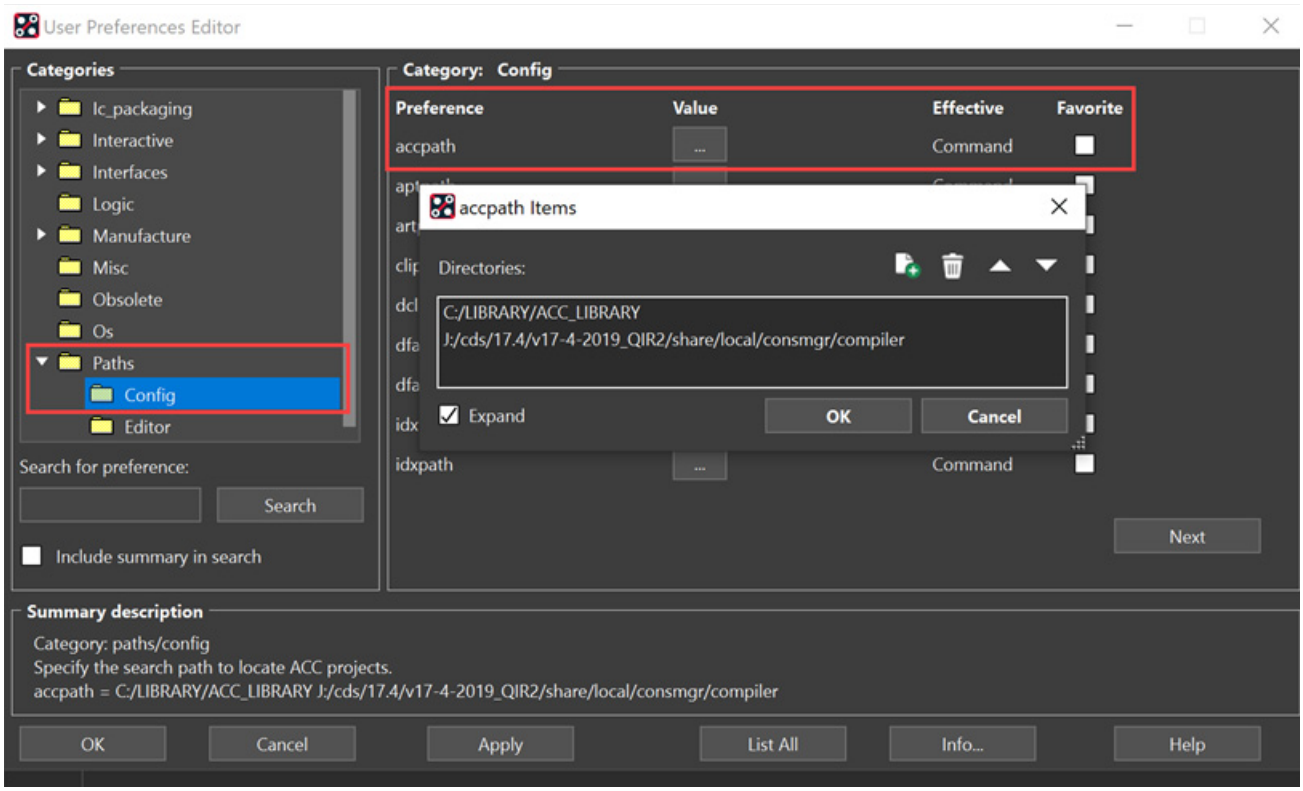
By default, library browser shows all the `.csv`, `.xml`, and `.zip` files in the current working directory and its sub-directories, which is specified by the *Library Path*. To access other libraries, click the pull-down combo box before *Library Path.* It displays default Cadence

compiler library from the install hierarchy and user-defined libraries that are defined using the `accpath` environmental variable.

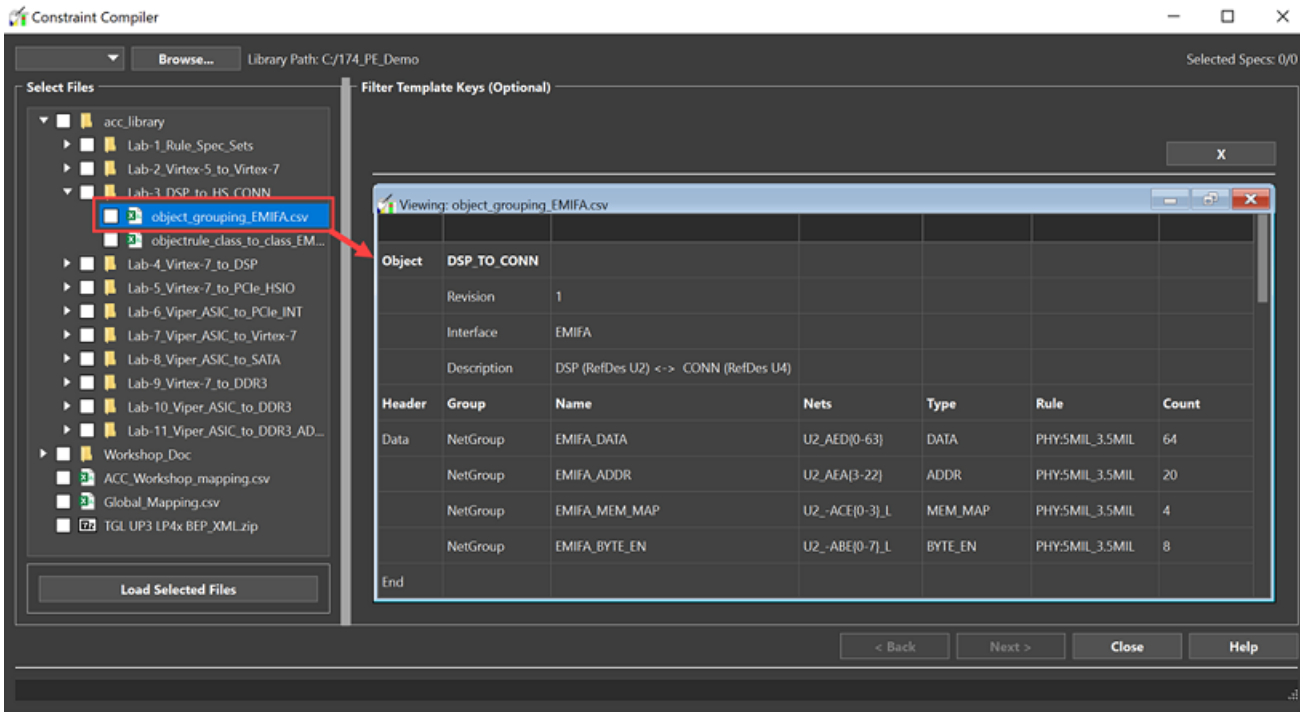


Clicking the *Browse* button opens a file browser to select library folder for viewing in the *Select Files* section of the compiler.

The library pull-down combo box uses the value of `accpath` environment variable to display the libraries. You can set the `accpath` environment variable in *Config* folder in the *Paths* category in the User Preferences Editor.



Double-clicking any `.csv` file opens a preview in the right side of the library browser.

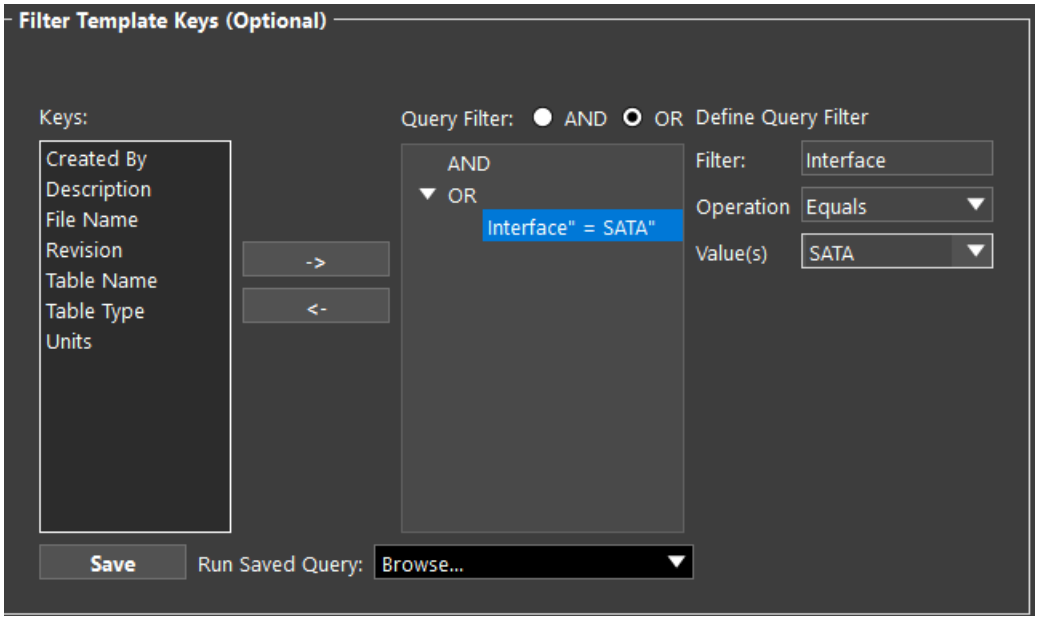


To process the selected files, click the *Load Selected Files* button. This option extracts all the key/value pairs from the data tables and loads them in the filter section for query processing.

Filter Section

The right section represents a filter. You can create a query based on the keys defined in the data tables to filter relevant tables from the library to compile. You can set filter operation to

either AND and OR state and choose values for the selected keys in the drop-down list of the *Define Query Filter*.

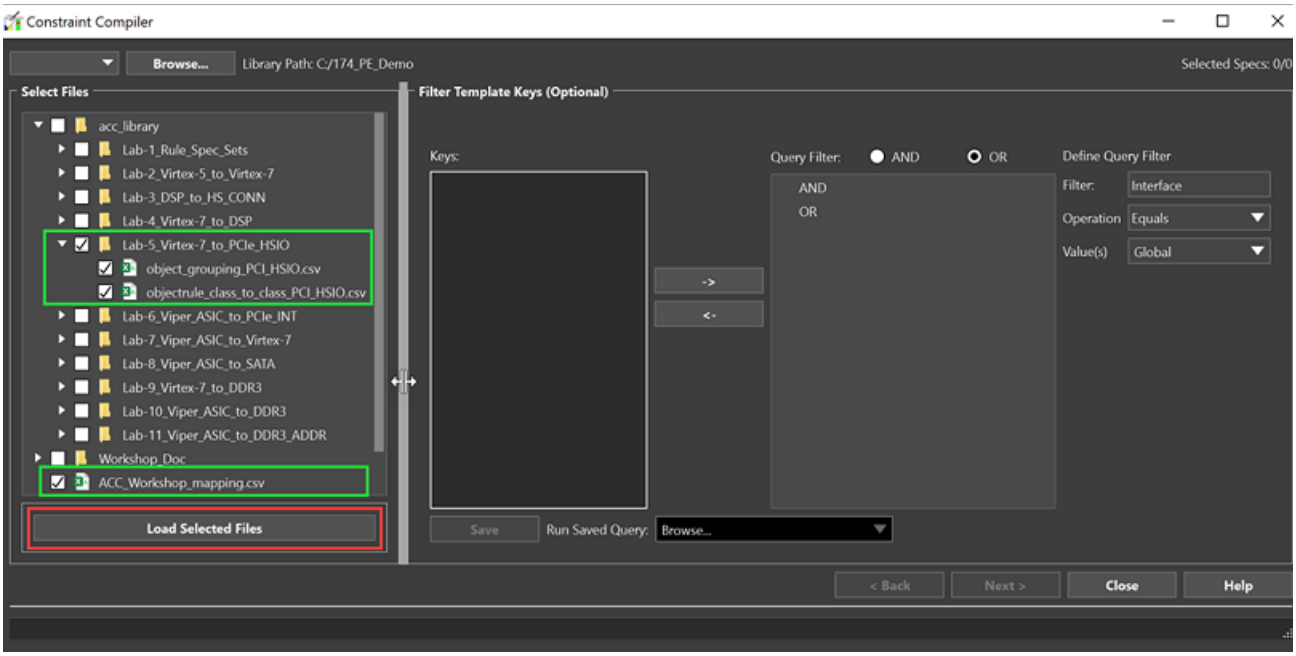


The *Save* option saves the query to a file (`.qry`). You can run any saved query using *Run Saved Query*.

Running constraint Compiler

To choose the required data tables for compilation, first select the library files. You can select the files in two ways:

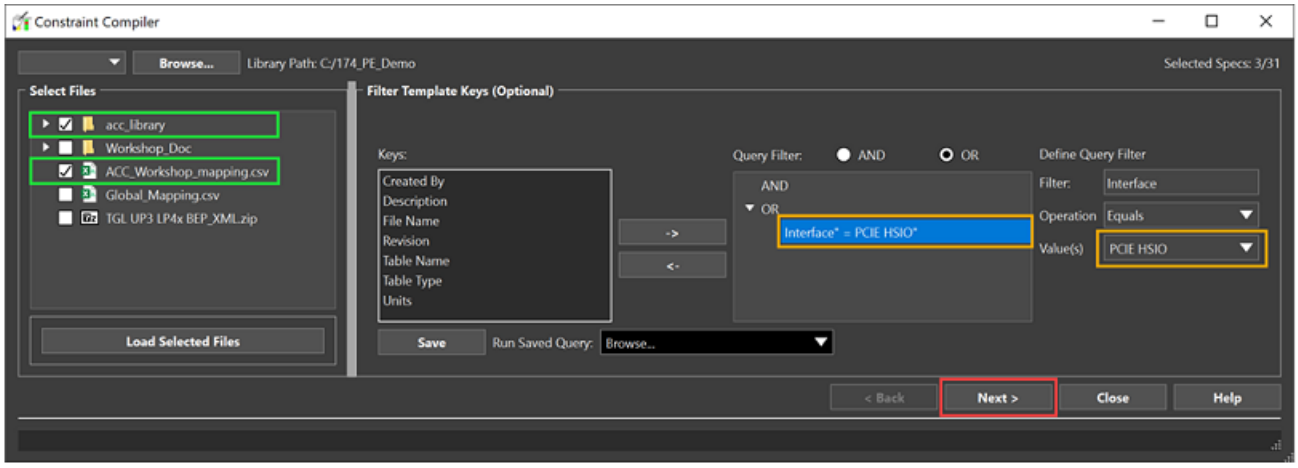
- Select explicit `.csv` files in the *Select Files* section by their names and dependencies and load them for processing

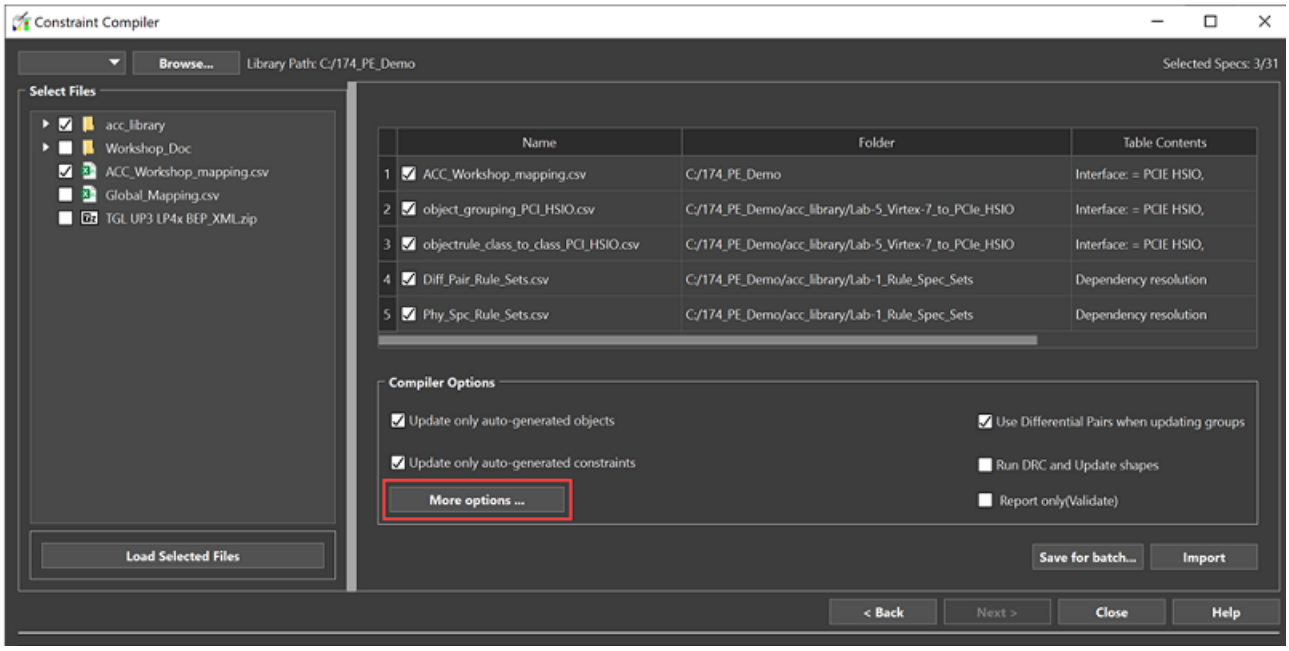


- Select all the `.csv` files and load them for filtering using queries.

  Note: If a table name is found in multiple files, only one file is checked with all other automatically unchecked

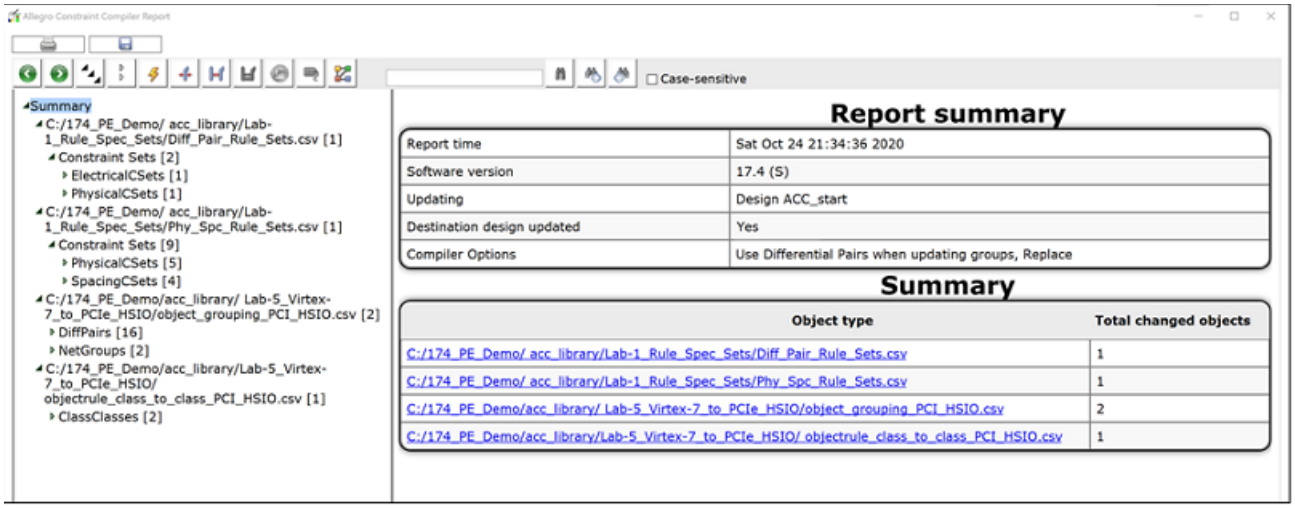
Click the *Next* button to preview the selected files before running compiler.

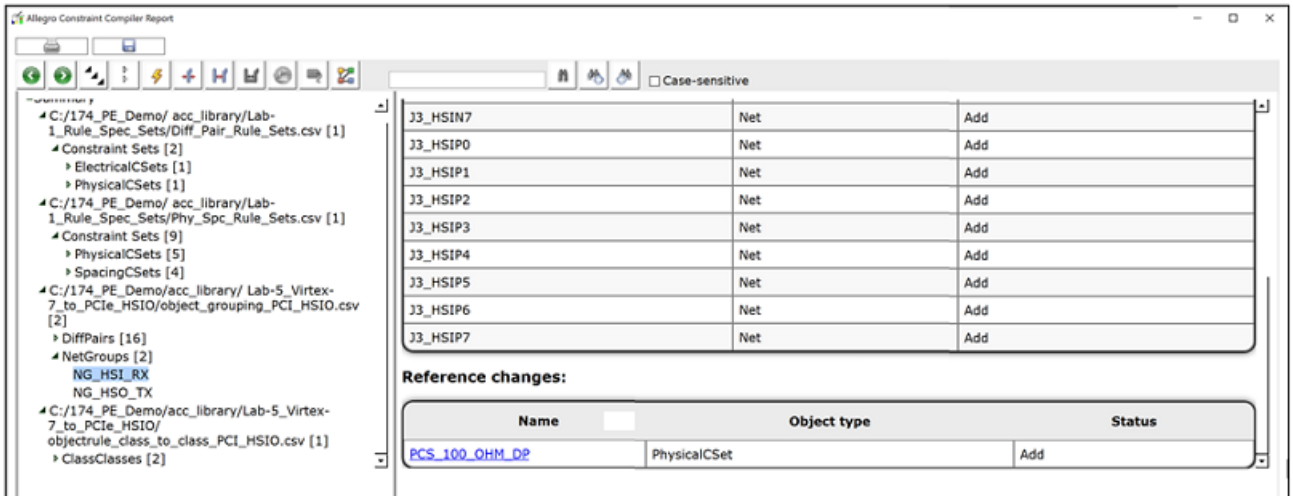




Before applying constraints to design, you can run the compiler in read-only mode to check for errors or compatibility with the design. Enable *Report only (Validate)* compiler option

and click *Import*. Compiler validates constraints and generates Allegro constraint complier report that shows summary of constraint changes, errors and warnings.





Troubleshooting Compiler Errors

Compiler reports error during import in some cases. To identify and fix the issues do the following:

- Inconsistent topology: Applying a Net topology to a XNet or a XNet topology to a single Net in the design.
  - Review mapping results in report, identify problem and corrective action
  - Generate XNets in the design to resolve the issue

Following is an example of the mapping results table to troubleshoot mapping issues:

Topology "HBR3 CTLE NET Link Topology - Tx" mapping results

Net	Net Pin/T-Point	Topology Endpoint	Status
NET4_N			Error: Unmapped Pins/Unmapped Topology endpoint(s)
	U401.11	$IO_SOC (Part)	Mapped by alias
		V0 (Via)	T-Point will be created
		V1 (Via)	T-Point will be created
		CONNECTOR (Connector)	Pin not found
	C5006.1		Endpoint not found
NET4A_N			Error: Net not processed
	R5003.3		Endpoint not found
	C5006.2		Endpoint not found
NET4B_N			Error: Net not processed
	R5003.4		Endpoint not found
	J5101.6		Endpoint not found

Error: Pins cannot be mapped to endpoints in the topology.

Topology "HBR3 CTLE XNET Link Topology - Tx" mapping results

Net	Net Pin/T-Point	Topology Endpoint	Status
NET8_N			Error: Unmapped Pins/Unmapped Topology endpoint(s)
	U401.15	$IO_SOC (Part)	Mapped by alias
		V0 (Via)	T-Point will be created
		V1 (Via)	T-Point will be created
		C1.A (Discrete)	Pin not found
	J5101.2		Endpoint not found
Unknown			Error: No valid net found.
		C1.B (Discrete)	Endpoint not found
		CMC.A (Discrete)	Endpoint not found
Unknown			Error: No valid net found.
		CMC.B (Discrete)	Endpoint not found
		CONNECTOR (Connector)	Endpoint not found

Error: Number of Nets in the XNet does not match the design, generate XNets and rerun.

- Constraint table alias to design name association is not in name mapping table
  - Compiler automatically generates a log file with separate name tables by Interface identifying the required alias to design specific name as follows:

- Name mapping specification for the Interface exists but some aliases are missing. Following is a Name and Object Table scenario:

Name	HSIO		
	Interface	HSIO	
Header	Type	Alias	Design
Data	*	HSIO_TX	J3_HSO
End			

Object	HSIO_BUS			
	Interface	HSIO		
Header	Group	Name	Member Kind	Member
Data	NetGroup	HSO_TX	Net	${HSIO_TX}${LANE}

cds_missing_alias.csv: Copy the Type and Alias cells to Name Mapping table for the design and enter the design specific name.

Name	HSIO		
	Interface	HSIO	
Header	Type	Alias	Design
Data	*	LANE	
End			

- Name mapping specification for the Interface does not exist. Object table scenario which contains aliases but no mapping:

Object	HSIO_BUS			
	Interface	HSIO		
Header	Group	Name	Member Kind	Member
Data	NetGroup	HSO_TX	Net	${HSIO_TX}${LANE}

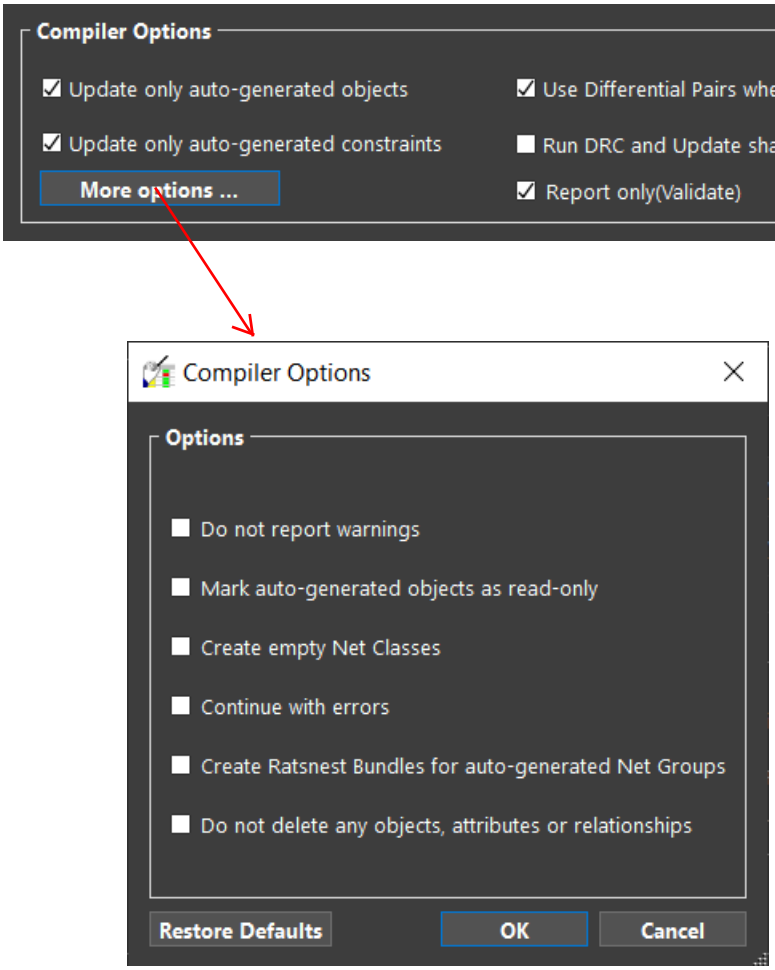
cds_missing_name.csv: Copy entire name table to name mapping table for the design and specify the design specific name

Name	HSIO		
	Interface	HSIO	
Header	Type	Alias	Design
Data	*	HSIO_TX	
	*	LANE	
End			

- Copy log files information into the Mapping Name table for the design prior to running the compiler.

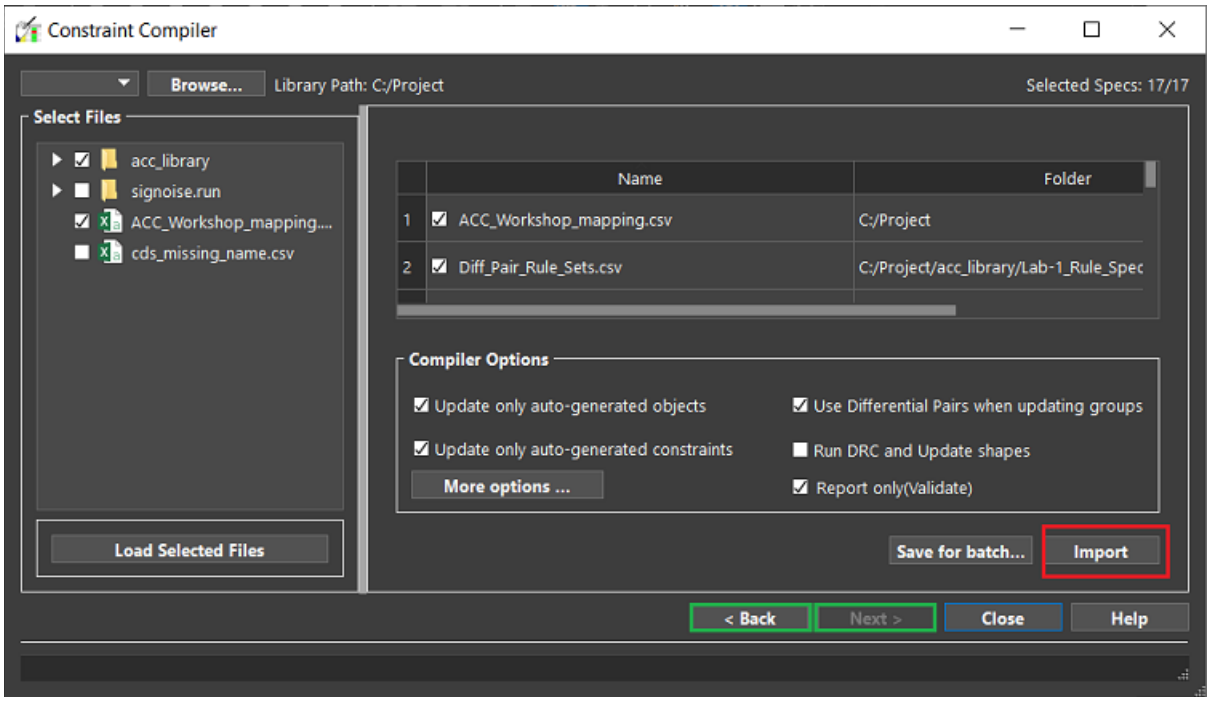
Complier does not delete any existing constraint information.It only adds new constraints or modify the existing constraint values as specified in the data tables. You can change the

default behavior in the *Complier Options* section to and import the constraints into the design.



For more information, see the Tools – Constraint Compiler command in the *Constraint Manager Reference*.

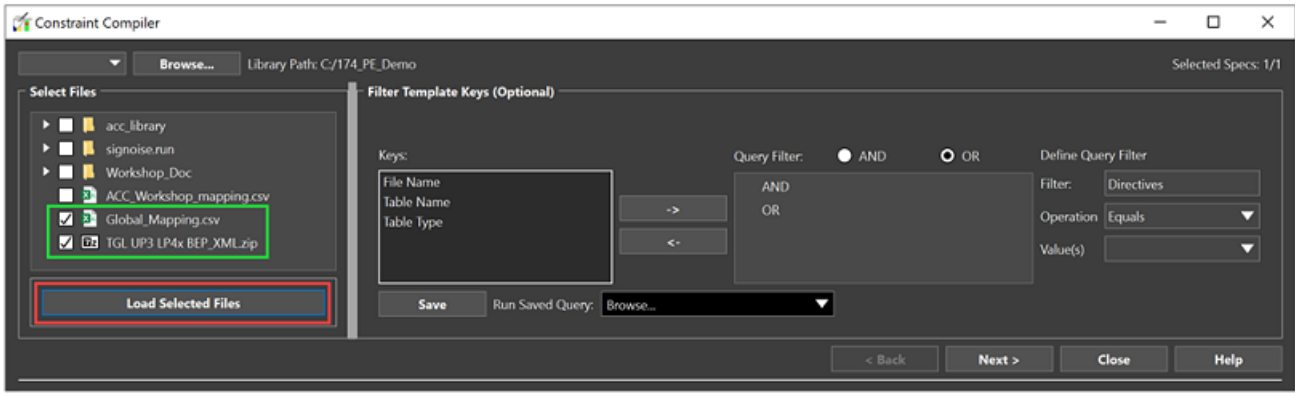
To modify the selection of files you can navigate using *Back* and *Next* buttons and verify the compiler report generated each time you run validation. To apply constraints, click *Import*.

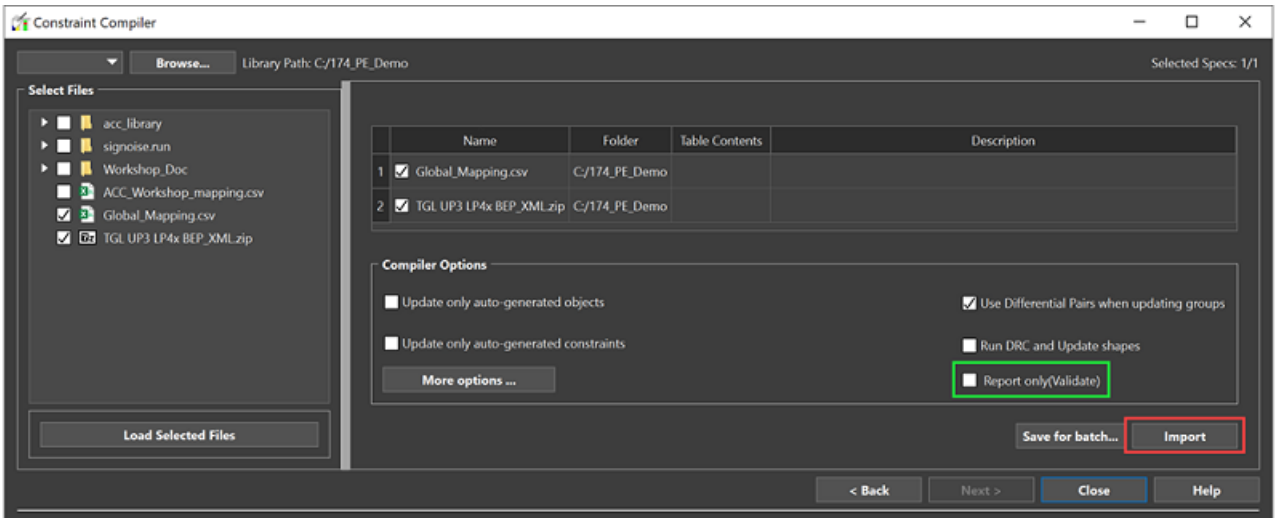


As an alternate to `.csv` files you can select `XML Constraint Data Set ZIP`, or if already extracted, choose `Design.XML` file and Global Mapping Table to apply the constraints in a design.

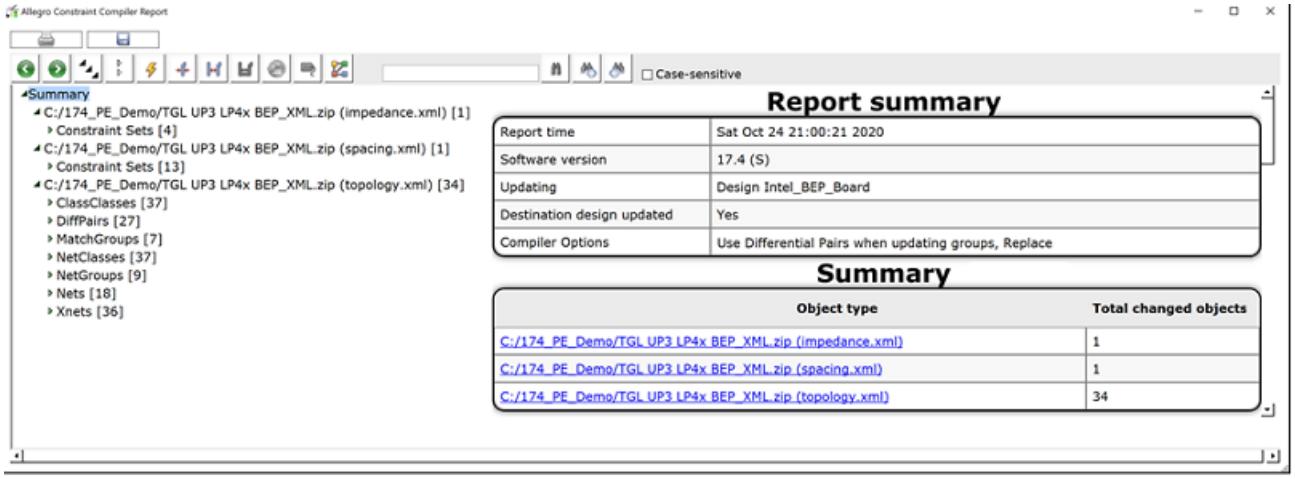
The following mapping table example shows a single table with the name `Global` and is used across all Interfaces included with the Constraint Data Set.

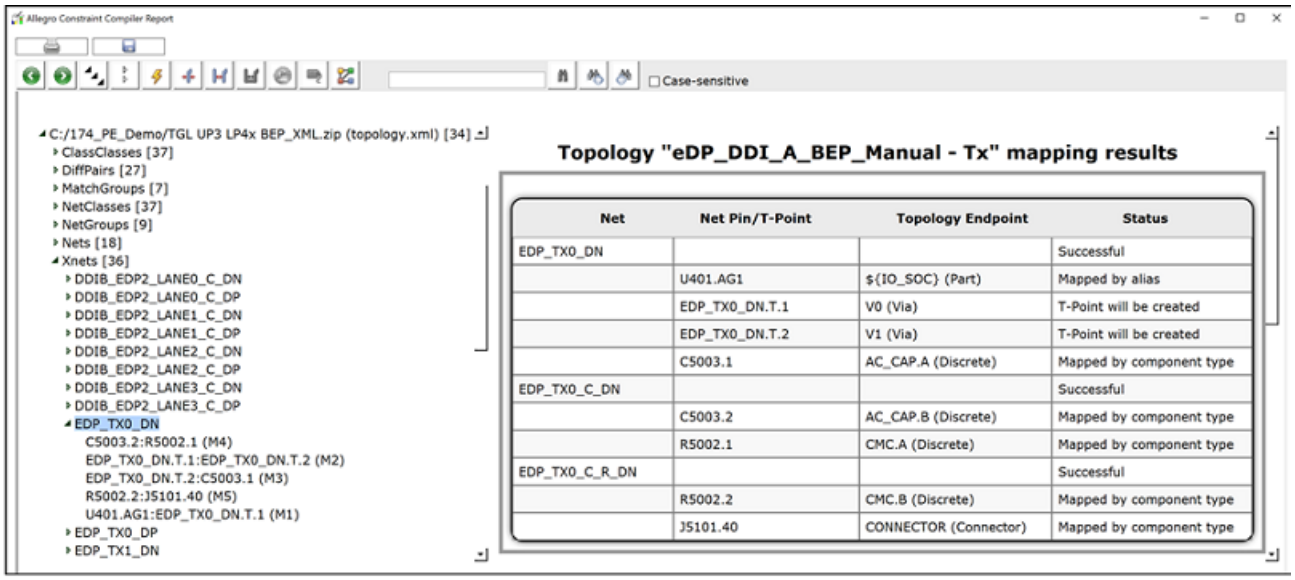
Name	Global		
Header	**Type**	**Alias**	**Design**
Data	Part	IO_SOC	U401
End			



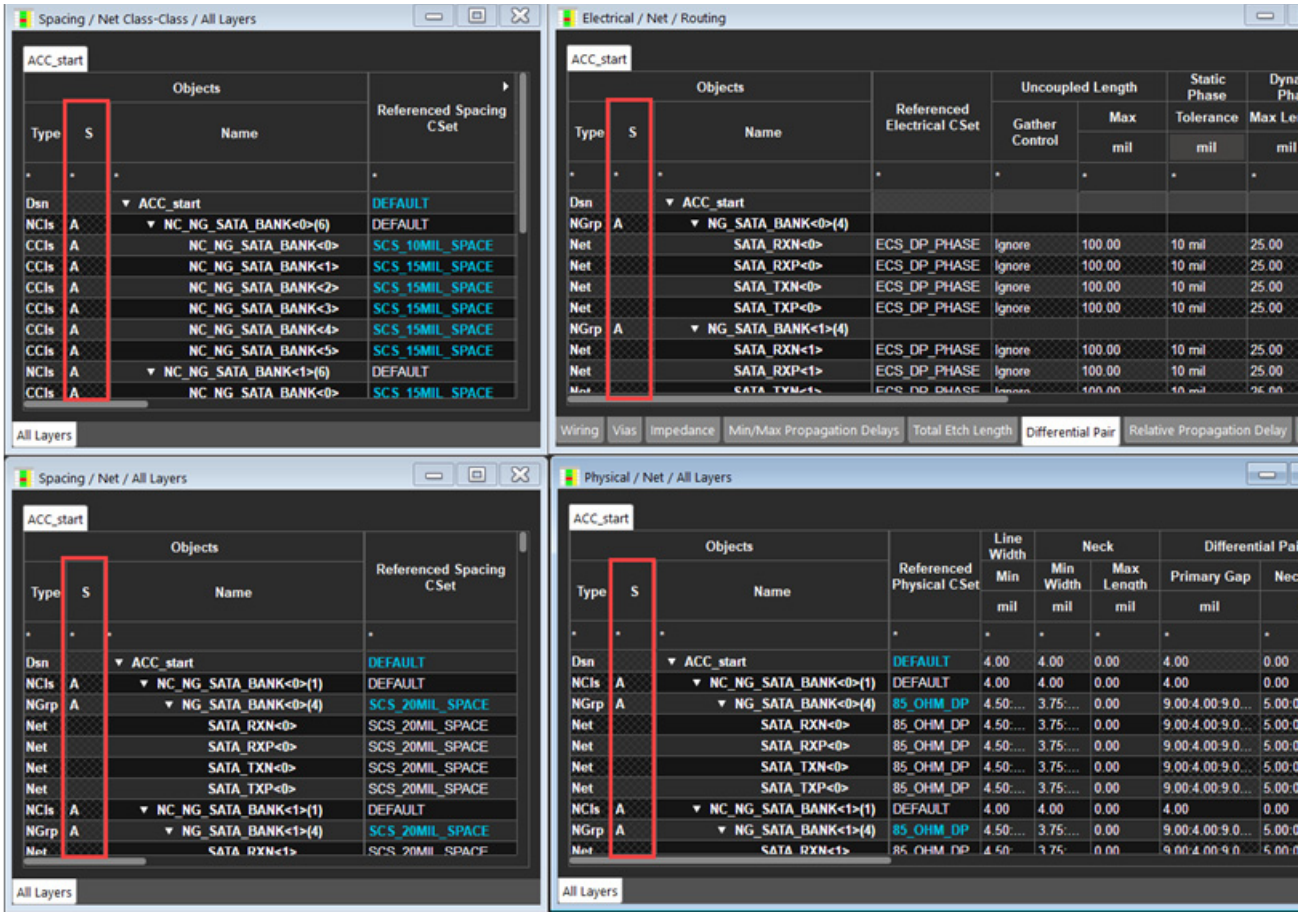


The complier process all the files and display the summary in constraint difference report. The report contains all warnings, errors and changes made to the design. Expanding summary items displays the respective updates in the right pane of the report viewer.





In Constraint Manager, constraints objects created by the compiler are labeled with an “A” in the sub-filter field and indicates an Auto-generated object. These objects are read-only and can be updated only by compiler.



Running Constraint Compiler in Batch Mode

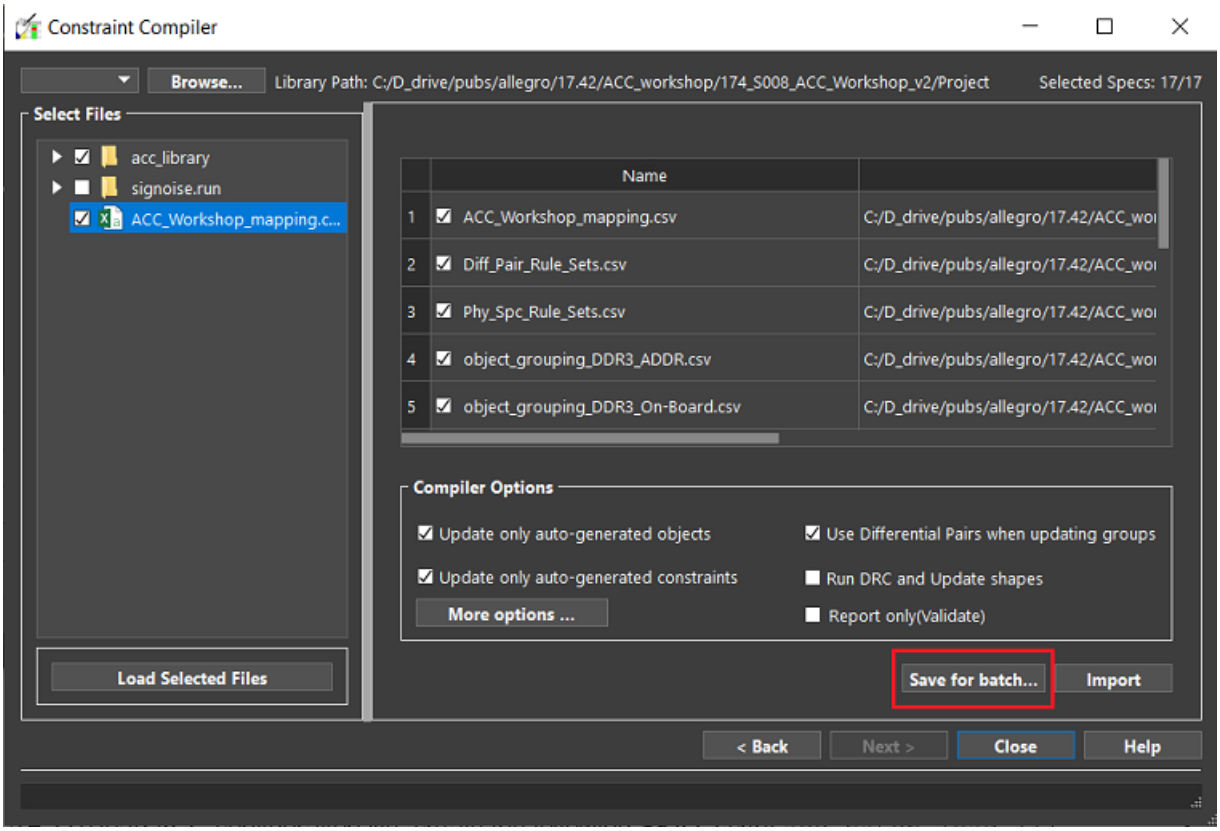
Constraint compiler can be run in batch mode using a SKILL API `cmxlCompile()`. This SKILL function requires an ACC configuration file (`.accx`) which is generated from the Constraint Compiler UI.

Follow the steps to run compiler in the batch mode:

1. Choose *Tools – Constraint Compiler*.
2. To import tables, select them from the *Select Files* section.
3. Click *Load Selected Files* followed by the *Next* button.

4. If needed, set *Compiler options* and use *More Options* to adjust the other compiler settings.

5. Click *Save for batch* and save the file as `acc_batch.accx`.



6. To load ACC configuration file, create the following SKILL script `acc_batch_load.il`:

```
procedure(acc_batch_load()
    axlCMDBInit()
    design = cmxlFindObject(REDS_DESIGN_OBJ)
    when(design
        cmxlDBSkillInit(design)
        cmxlCompile(design, "./acc_batch.accx")
    )
    axlCMDBExit()
)
```

7. To run the SKILL script in layout editor, enter the following commands in the command window:

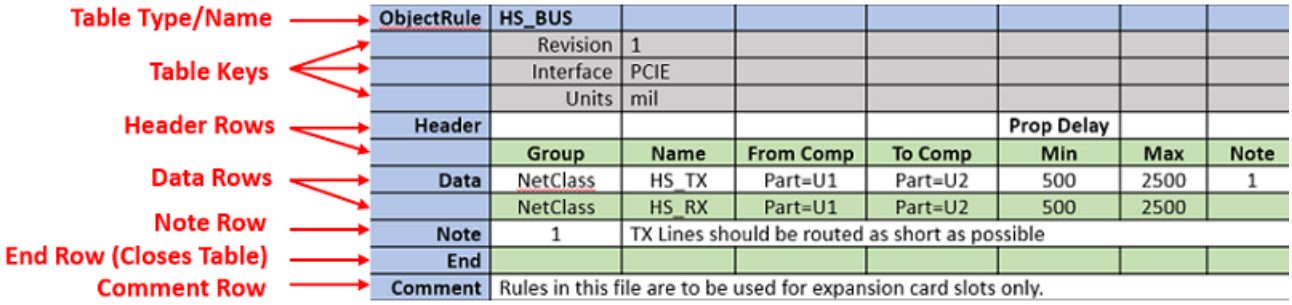
```
skill load("acc_batch_load.il")
skill acc_batch_load
```

Data Tables

For complier to read and apply the constraints information in a design, two types of tables are needed: object tables and mapping tables. The object tables contain agnostic constraint information and the mapping tables associate the agnostic constraints to the design-specific objects. The tables are in `.csv` and `.xml schema` format and have the same structure.

Table Structure

All tables follow a general format with the only differences being additional columns. The following image shows a sample data table.



ObjectRule	HS_BUS						
	Revision	1					
	Interface	PCIE					
	Units	mil					
Header					Prop Delay		
	Group	Name	From Comp	To Comp	Min	Max	Note
Data	NetClass	HS_TX	Part=U1	Part=U2	500	2500	1
	NetClass	HS_RX	Part=U1	Part=U2	500	2500	
Note	1	TX Lines should be routed as short as possible					
End							
Comment	Rules in this file are to be used for expansion card slots only.						

Row Name	Row Description
Table Type and Name	Identify the table usage. It begins a table and concludes with an *End* statement in first column. A file can contain multiple tables and/or table types.

Table Keys	Provide specific information that can be queried by the compiler to narrow down the table selection for execution.Table keys are specified between Table Type/Name and Header rows. Two keys are reserved keys and are: ■ Interface: Name that links associated tables together ■ Units: Units of values entered in tables ■ DefPhysicalCSet: Physical CSet name used as baseline for new Rule Set ■ DefSpacingCSet: Spacing CSet name used as baseline for new Rule Set ■ All other Keys defined in the header are user definable All other Keys defined in the header are user-definable
Header Rows	Specify data type or constraint type information. Header rows are specified between Table Keys and Data Rows. The name of column header with semicolon separators could span two rows. For example, *Prop Delay: Min* shown as *Prop Delay* in first row and *Min* in second row
Data Rows	Specify constraint values and requirements. Data rows are specified between Header Rows and the End statement.
Comment Rows (;)	Communicates design intent.
Note rows (#)	Adds notification to compiler report.

Tables Types

The Object and Specification tables contain agnostic constraint information for common design circuity and the Mapping table associates the agnostic constraints to design specific objects. The compiler reads and combines all this constraint data to fully define the constraint requirements for a design.

- Mapping Name Table
- Global Mapping Table
- Topology Table
- Object Table

- Rule Specification Table
- Object Rule Specification Table

Mapping Name Table

Mapping table associates the design specific name to a string alias that is defined in other tables. For example, Part (Component), Net or "*" (all object types). The compiler reads this table to map alias place holders referenced in the data tables to be replaced with design specific names. The mapping name table provides a central location to remap all the aliases to the design specific names.

Column Header Description

Type	Design object type to be processed (Part, Net, "*****" means all object types)
Alias	Name for design objects referenced in other tables
Design	Design object to be mapped using Explicit Name, Regular Expression or Number Range {<range start> - <range end>}

Design Object Selection

Selection of design object names can be done using explicit name, partial name with number range or regular expression or a combination of both. For example,

- Number range for nets `DDR_DQ<7>` through `DDR_DQ<16>` would be `DDR_DQ<{7-16]>`
- Regular expression for nets `DDR_CK0_N` and `DDR_CK0_P` would be `DDR_CK0_[N,P]`
- Regular expression and number range for nets `HSI_N0, HSI_P0, HSI_N1, HSI_P1` would be `HSI_[N,P]{0-1}`

Sample Mapping Table

The following image shows a sample mapping table entries for a table *MY_MAP*.

Name	MY_MAP		
	Revision	1	
	Created by	Cadence	
Header	**Type**	**Alias**	**Design**
Data	Part	PROC	U25
	Part	MEM	U{1-5}
	Part	FLASH	U10:U14:U28:U35
	Net	DDR_DQ	DDR_DQ<{0-3}>
	Net	PCIE_HS	HS[O,I]_{0-3}
	Net	PCIE_DP	HS[O,I]_[N,P](0)
	*	BANK	M{0-3}
	*	LANE	BYTELANE{0-8}
End			

Alias referenced and expanded in tables

- **${PROC}** -> U25
- **${MEM}** -> U1, U2, U3, U4, U5
- **${FLASH}** -> U10, U14, U28, U35
- **${DDR_DQ}** -> DDR_DQ<0>, DDR_DQ<1>, DDR_DQ<2>, DDR_DQ<3>
- **${PCIE_HS}** -> HSO_0, HSO_1, HSO_2, HSO_3, HSI_0, HSI_1, HSI_2, HSI_3
- **${PCIE_DP}** -> HSO_N0, HSO_P0, HSI_N0, HSI_P0
- **DDR_${BANK}** -> DDR_M0, DDR_M1, DDR_M2, DDR_M3
- **DDR_${LANE}** -> DDR_BYTELANE0, DDR_BYTELANE1, DDR_BYTELANE2, DDR_BYTELANE3, DDR_BYTELANE4, DDR_BYTELANE5, DDR_BYTELANE6, DDR_BYTELANE7, DDR_BYTELANE8

Compiler Results

If the compiler detects an alias place holder without appropriate mapping to a design object, it reports an error and automatically generates a log file. The log file contains separate Name tables by Interface identifying the required alias to design specific name.

Global Mapping Table

The Global Mapping table is used to provide common mapping across all data tables. You can create this table with name `ACC_Directives.csv` and save in the `compiler` directory inside your current working directory.

The default global mapping table exist in the installation directory at *`<installation_directory>\share\pcb\consmgr\CDS_ACC_Directives.csv`*.

Sample Global Mapping Table

;	==========		
;	Global Name mapping. Can be leveraged by all Object specifications.		
Name	Global		
	LayerCount	Generic	
	Description	Generic layer type mapping information.	
Header	Type	Agnostic	Design
Data	GenericLayer	Microstrip	Conductor/Outer
	GenericLayer	Stripline	Conductor/Inner
End			
Directive	Global		
	Description	Allegro Constraint Compiler default directives	
Header	Type	Keyword	Value
Data	Calculation	Height	useHeightFromRefNum(x)
	Calculation	CALCBITS	bitsForLane(x)
	ObjectKind	Layer Assignment:Layer Type	GenericLayer
Comment	Physical Domain		
	ConstraintName	Width:Min Line	MIN_LINE_WIDTH
	ConstraintName	Width:Max Line	MAX_LINE_WIDTH
	ConstraintName	Neck:Min Line	MIN_NECK_WIDTH
	ConstraintName	Neck:Length	MAXIMUM_NECK_LENGTH
	ConstraintName	DP Min Space	DIFFP_MIN_SPACE
	ConstraintName	DP Gap	DIFFP_PRIMARY_GAP
	ConstraintName	DP Neck Gap	DIFFP_NECK_GAP

Topology Table

The Topology table is used to define routing sections for a Net or XNet which can be pin to pin or different transition points along the routed interconnect. These routing sections can then be referenced in an Object Rule table to drive Physical, Spacing and Electrical constraints for the specific section.

Column Header Description

Routing Sections	Top row header name with **Name**, **Begin** and **End** on the second-row of header
Name	Unique Routing Section Name used to identify a specific section of a routed connection Topology and Routing Section Name can be referenced in an ObjectRule Table to constrain a portion of the Net or XNet
Begin	Defines the beginning of the Routing Section. **NOTE:** First Data row "**Begin**" definition identifies RefDes starting point for the topology and must be **Part=***"RefDes"* or **Part=***${String}* Alias associated to RefDes in Name Mapping Table
End	Defines the end of the Routing Section.

Begin and End Supported Fields

Field Name	Description
Part=*<"RefDes">* **Discrete=***<"RefDes">* **Connector=***<"RefDes">*	Explicit "**RefDes**" present in design, must be in double quotes **NOTE: Discrete=** and **Connector=** values could be a **Unique Name** that will be auto-mapped to the design RefDes
Part=*${String}* **Discrete=***${String}* **Connector=***${String}*	**${String}** Alias name associated to a design RefDes in a Name Mapping Table **NOTE: Discrete=** and **Connector=** values could be a **Unique Name** if they are not the first **Begin** definition in a Topology
Part=*<Name>*	**Unique Name** or **RefDes Prefix** used for auto-mapping design components prioritized using Component Class **(CLASS=IC)** then **RefDes Prefix U***
Discrete=*<Name>*	**Unique Name** or **RefDes Prefix** used for auto-mapping components prioritized using Component Class **(CLASS=DISCRETE)** then **RefDes Prefix R***, **C***, **L*** **RefDes Prefix.A** for Route Section End and **RefDes Prefix.B** for Route Section Begin indicates pass-through components (XNets). **NOTE: XNets must exist in the design**
Connector=*<Name>*	**Unique Name** or **RefDes Prefix** for auto-mapping design components prioritized using Component Class **(CLASS=IO)** then **RefDes Prefix J***
Width=*<Name>*	**Unique Name** to identify a Width transition
Via=*<Name>*	**Unique Name** to identify a Via transition
RegionExit=*<Name>*	Name of **Constraint Region** in design to identify a transition out of a defined Region
RegionEntry=*<Name>*	Name of **Constraint Region** in design to identify a transition into a defined Region
Tee=*<Name>*	**Unique Name** to identify a routing tee for branched topologies (Symmetrical, Star, etc.)

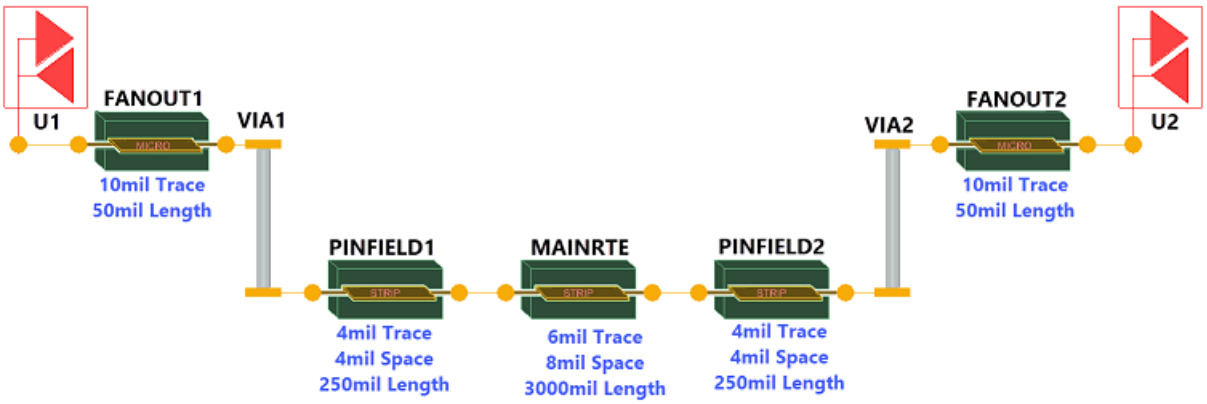
Mapping Examples (Begin/End)

Field Example	Description
Part="U1"	Design RefDes **U1**
Part=${PROC}	Alias name **PROC** mapped to Design **RefDes** in a Name Mapping Table
Part=MEM1	Auto-mapped to RefDes with Component Class **(CLASS=IC)**
Discrete=CAP1	Auto-mapped to RefDes with Component Class **(CLASS=DISCRETE)**
Connector=CONN1	Auto-mapped to RefDes with Component Class **(CLASS=IO)**
RegionExit=1MM_BGA	Exit Constraint Region shape name "**1MM_BGA**"
RegionEntry=P8MM_BGA	Entry into Constraint Region shape name "**P8MM_BGA**"

What is a Topology and how can it be used?

Physical and spacing constraints are defined across the entire Net or XNet whereas electrical constraints can be defined at the lower pin pair level. In complex designs, it is important to define constraints on specific sections of the routed connections beyond basic pin pair or region-based rules, such as fanout, pin field, breakout, main route, and so on. To facilitate this, begin and end points of routing sections can be defined based on physical representation in the design (via, width and region entry/exit transitions).

The following illustration shows net sectioning to drive different constraints per routing section.



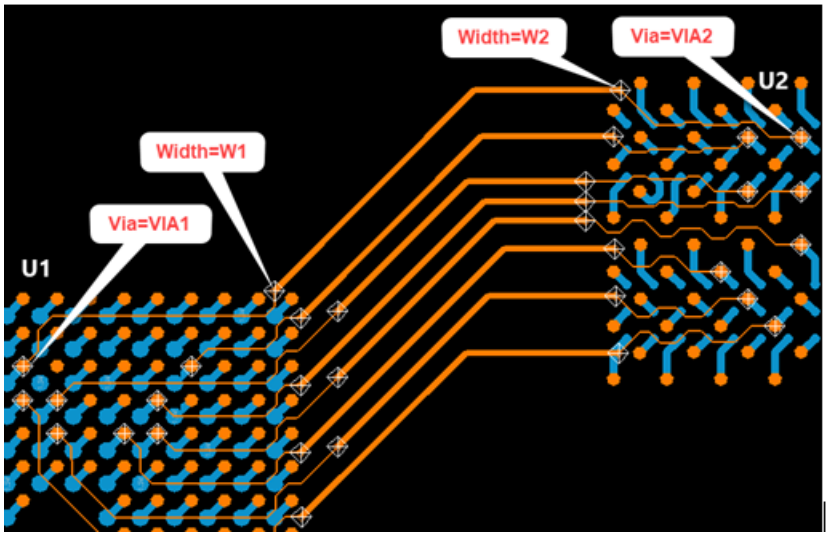
Note: Topology graphical representation is not available in the layout editor.

For this topology the topology table and object rule table are as follows:

Topology	DDR_DQ		
	Interface	DDR	
Header	Routing Sections		
	Name	Begin	End
Data	FANOUT1	Part="U1"	Via=VIA1
	PINFIELD1	Via=VIA1	Width=W1
	MAINRTE	Width=W1	Width=W2
	PINFIELD2	Width=W2	Via=VIA2
	FANOUT2	Via=VIA2	Part=MEM1
End			

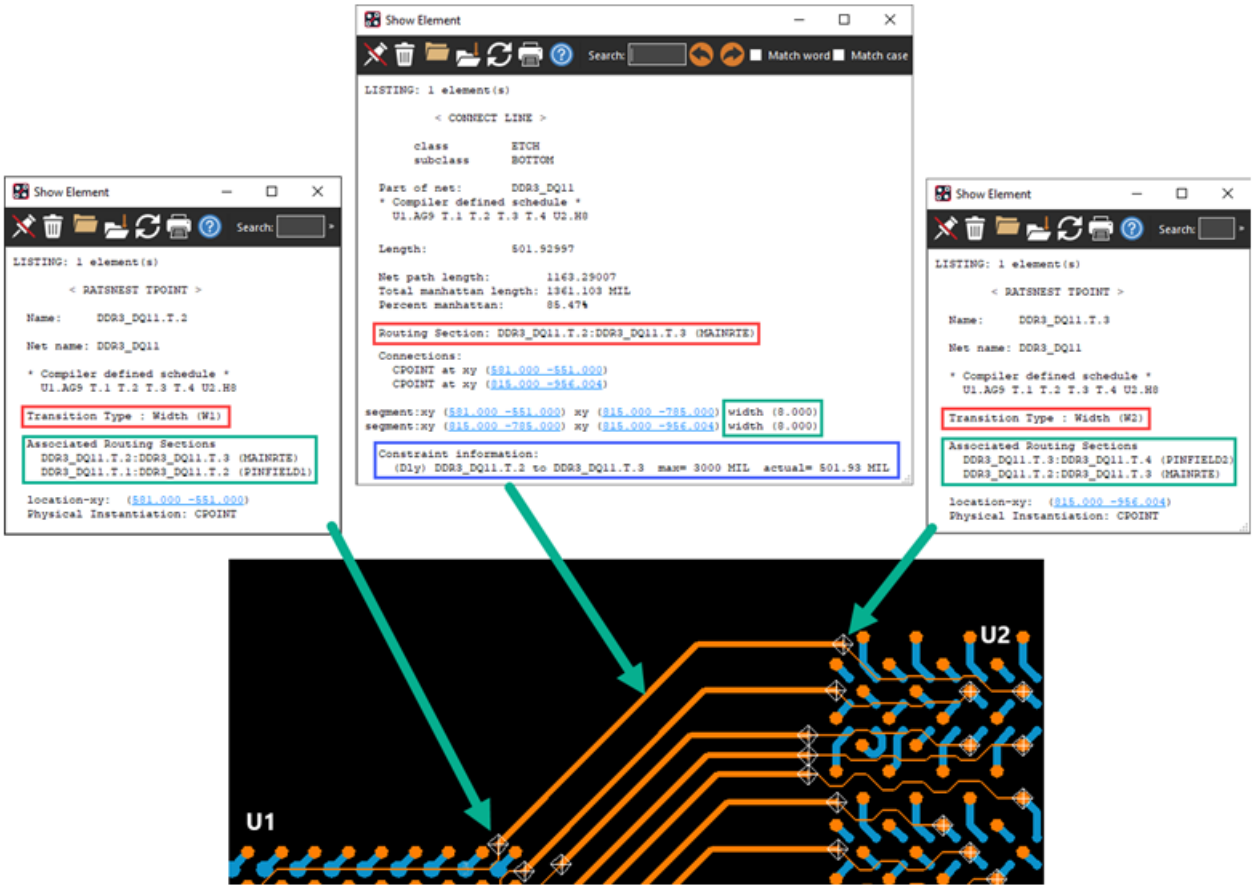
ObjectRule	DDR_DATA					
	Units	mil				
	Interface	DDR				
Header	Type	Topology Rule	Routing Sections	Physical Rule	Spacing Rule	Prop Delay
						Max
Data	DQ	DDR_DQ	FANOUT1	10MIL_TRACE		50
	DQ	DDR_DQ	PINFIELD1	4MIL_TRACE	4MIL_SPACE	250
	DQ	DDR_DQ	MAINRTE	8MIL_TRACE	8MIL_SPACE	3000
	DQ	DDR_DQ	PINFIELD2	4MIL_TRACE	4MIL_SPACE	250
	DQ	DDR_DQ	FANOUT2	10MIL_TRACE		50
End						

Topology table drives constraints on a section of a net. After constraints are loaded, special transition Rat-Ts becomes visible on the canvas and dynamically snap to the appropriate location as defined in the topology. The following image show the canvas view of Rat-Ts that are displayed as diamond shape with cross.



These transition Rat-Ts define begin/end points of the route representing the routing section with constraints being checked accordingly. This approach provides more flexibility beyond traditional region-based constraints as well as electrical constraints specific to a pin pair.

The show element command on these Rat-Ts displays information of constraints, routing sections and transition type as shown in the following image.



In Constraint Manager, you can view the topology schedule and electrical routing section constraints.

Objects			Topology		
Type	S	Name	Schedule	Actual	Margin
*	*	*	*	*	*
Net		DDR3_DQ8	User Defined (Compiler)	PASS	
Net		DDR3_DQ9	User Defined (Compiler)	PASS	
Net		DDR3_DQ10	User Defined (Compiler)	PASS	
Net		DDR3_DQ11	User Defined (Compiler)	PASS	
Net		DDR3_DQ12	User Defined (Compiler)	PASS	
Net		DDR3_DQ13	User Defined (Compiler)	PASS	

Objects			Prop Delay		
Type	S	Name	Max mil	Actual	Margin
*	*	*	*	*	*
Net		▼ DDR3_DQ11			-107.837...
PPr	A	U1.AG9:DDR3_DQ11.T.1 (FANOUT1)	50 mil	27.84 mil	22.16 mil
PPr	A	DDR3_DQ11.T.1:DDR3_DQ11.T.2 (PINFIELD1)	250 mil	357.837 ...	-107.837...
PPr	A	DDR3_DQ11.T.2:DDR3_DQ11.T.3 (MAINRTE)	3000 mil	501.93 mil	2498.07 ...
PPr	A	DDR3_DQ11.T.3:DDR3_DQ11.T.4 (PINFIELD2)	250 mil	253.416 ...	-3.416 mil
PPr	A	DDR3_DQ11.T.4:U2.H8 (FANOUT2)	50 mil	22.268 mil	27.732 mil

Similarly, spacing and physical routing section constraints can be verified in Constraint Manager.

Objects			Referenced Physical CSet	Line Width
Type	S	Name		Min
*	*	*	*	*
Net		▼ DDR3_DQ11	DEFAULT	7.500:4.5
PPr	A	U1.AG9:DDR3_DQ11.T.1 (FANOUT1)	PCS_10MIL_TRACE	10.000
PPr	A	DDR3_DQ11.T.2:DDR3_DQ11.T.3 (MAINRTE)	PCS_8MIL_TRACE	8.000
PPr	A	DDR3_DQ11.T.3:DDR3_DQ11.T.4 (PINFIELD2)	PCS_4MIL_TRACE	4.000
PPr	A	DDR3_DQ11.T.1:DDR3_DQ11.T.2 (PINFIELD1)	PCS_4MIL_TRACE	4.000
PPr	A	DDR3_DQ11.T.4:U2.H8 (FANOUT2)	PCS_10MIL_TRACE	10.000

Objects			Referenced Spacing CSet	Line To
Type	S	Name		Line
				mil
*	*	*	*	*
Net		▼ DDR3_DQ11	DEFAULT	6.500:6.000:
PPr	A	DDR3_DQ11.T.2:DDR3_DQ11.T.3 (MAINRTE)	SCS_8MIL_SPACE	8.000
PPr	A	DDR3_DQ11.T.1:DDR3_DQ11.T.2 (PINFIELD1)	SCS_4MIL_SPACE	4.000
PPr	A	DDR3_DQ11.T.3:DDR3_DQ11.T.4 (PINFIELD2)	SCS_4MIL_SPACE	4.000

Notes

- It is recommended to apply topology-based constraints on routed nets to verify the routed interconnect has the required transitions to start checking constraints accurately.
- If a routed interconnect does not have the required transition to identify the routing sections, then transition Rat-Ts will remain floating until the routing is corrected. Use show element for transition Rat-Ts to resolve the issue. Once updated, routing automatically snaps the Rat-T to its appropriate location.
- It is possible to apply topology-based constraints on unrouted nets. Some of the interactive routing commands automatically identify a transition and start checking constraints accordingly.
- Topologies with XNets expects that the appropriate XNets already exist in the design. This information can be checked from the compiler report and can be used to update a design.
- In some cases, a routed interconnect may or may not contain all the required transitions as outlined in the topology and a scheduled DRC error is generated. For example, a pin

field trace neck down may not be required on connections on the edges of the BGA pin field. As a result the transition will be left floating.

- Topology using TEE= transitions does not snap to the appropriate branching Tee location and reports a schedule DRC. TEE= transition are displayed as a solid-fill diamond shape. Show Element on DRC error helps to determine the transition location and the Rat-T can be moved manually to resolve the schedule error.

Object Table

The Object table creates various constraint objects, their members and type classifications for reference in other tables. Rule Set and existing Constraint Set references can be made during object group creation as well a *Count* check to ensure that all the expected members are selected in the design.

Column Header Description

Group or Kind	Type of Group to create: **DiffPair**, **NetGroup**, **MatchGroup**, **NetClass**, **ElectricalNetClass**, **PhysicalNetClass**, **SpacingNetClass**, **SameNetSpacingNetClass** or **Region**
	Using **NetClass** will create both Physical and Spacing Net Classes. Leaving field blank will create a **NetGroup** of the objects. Creating **SpacingNetClass** or **MatchGroup** groups in this table are **not** required as a **NetGroup** can be referenced in an ObjectRule using a **Type** classification which automatically create the required groups to drive constraints
Name	Name of Group that can be leveraged by Rule and Object Rule Tables to drive rule assignment
	Group Name will have following prefix added to name when created into Constraint Manager
	Physical CSet = "**PCS_**", Spacing CSet = "**SCS_**", Electrical CSet = "**ECS_**"
	Net Class ="**NC_**", Net Group = "**NG_**", Match Group = "**MG_**", Diff Pair = "**DP_**"
Member Kind	Indicates the type of data entered in the **Member** column: Legal values **Pin, Net, XNet, DiffPair, Bus** and **NetGroup.** Leaving field blank defaults to Net objects in **Member** column
Member	**Member Kind** determines what is entered in this column
	Member Kind **Pin**: **${***Alias***}** with Pin # (**${*Alias*}.Pin#**) OR RefDes with Pin # (**RefDes.Pin#**)
	Member Kind **Net**: • Net Names using Explicit Name or with Number Ranges **{**<*range start*> - <*range end*>**}** • Combination of **${***Alias***}**, Number Ranges **{**<*range start*> - <*range end*>**}**, or Regular Expressions **[…]** **NOTE:** Specifying Net Name which is part of XNet will automatically add XNet to group
	Member Kind **XNet**: Same criteria as **Net** except looks for XNets, fails if does not exist
	Member Kind **Diff Pair**, **Bus** and **NetGroup**: Hierarchal object name must already exist in design prior to loading table, fails if does not exist
	Alias strings must be associated to design objects using a Mapping Name Table
Nets or Signals	Field value is same as **Member** column when **Member Kind** indicates **Net** **NOTE:** Use of **Member Kind** and **Member** columns are recommended over this column

Optional Column Header Description

Type	Type Classification of objects to be referenced in Rule Specifications for rule assignment.
	Can be referenced in an ObjectRule Specification table to automatically generate required Match Groups and Classes to drive Class to Class spacing constraints
Rule	**Rule Specification** or **Rule Specification:Rule Set** containing constraints for multiple domains
	Multiple Rule Specifications can be listed separated by a semicolon ";"
Physical Rule	Reference existing Physical Constraint Sets name in the design (First Priority)
	Rule Specification or **Rule Specification:Rule Set** reference containing Physical Constraints
Spacing Rule	Reference existing Spacing Constraint Sets name in the design (First Priority)
	Rule Specification or **Rule Specification:Rule Set** reference containing Spacing Constraints
Electrical Rule	Reference existing Electrical Constraint Sets name in the design (First Priority)
	Rule Specification or **Rule Specification:Rule Set** reference containing Electrical Constraints
Count	Target membership count for a Group of objects
	Compiler uses this value as a check to ensure the correct number of members are found
Optional	Enter **YES** on data row to indicate that some members may not be found in design
	Flexibility to include optional nets in table allowing compiler to continue import of remaining
	NOTE: **Count** column is used to ensure the minimum number of design members are found

Byte Lane Bit Calculator: $ByteLane(x, y, z)

- x = Byte Lane number, starting from 0
- y = Number of BITS in each Byte Lane
- z = Number of digits in each BIT number

Results: Generates series of data bit numbers based on current Byte Lane being processed.

Examples

- Enter $ByteLane(0,8,1) expands to number pattern: 0,1,2,3,4,5,6,7
- Enter $ByteLane(1,8,2) expands to number pattern: 08,09,10,11,12,13,14,15

Byte Lane 2nd Diff Pair Calculator (x4 Mode Data Strobe): $x4HighDP(x, y, z)

- x = Byte Lane number, starting from 0
- y = Total Number of Byte Lanes
- z = Number of digits in each BIT number

Results: Generates 2nd Diff Pair number in sequence based on Total number of Byte Lanes and current Byte Lane being processed.

Examples

- Enter $x4HighDP(0,8,1) in column expands to number pattern: 8
- Enter $x4HighDP(1,8,2) in column expands to number pattern: 09

Byte Lane Calculators requires NetGroup and Member entries in row to share a common Byte Lane number to determine which Byte Lane is being processed. Using an Alias in both entries to cycle through each Byte Lane number provides the linkage.

Net Selection

The Nets or Signals column entries are limited to XNet/Net objects, which can be selected in the design using explicit net names or with a partial net name and a number range specified within curly brackets {n-n}. For more complex net name selection regular expressions are also supported and if the net name contains a square bracket, then a forward slash "\" can be used to treat it like a regular character. If selecting all XNet/net members for group assignment, which are already members of a different type of group (Net Class, Net Groups, Differential Pairs), then the existing group gets added to the newly created group. For example,

- Selecting all members of a Net Group to create a Net Class, the Net Group object gets added to the Net Class
- Select both members of a Differential Pair for addition to a Net Class, the Differential Pair object gets added to the Net Class

Objects created by the compiler are labeled with an ‘A’ indicated as auto-generated objects in Constraint Manager.

Sample Object Table

Object	MY_OBJECT						
	Revision	1					
	Interface	PCI					
	Created by	Cadence					
Header	**Group**	**Name**	**Member Kind**	**Member**	**Type**	**Rule**	**Count**
Data	NetClass	HS-PORT_DATA	Net	HD{0-31}	HS-DATA	SPC:HS_SPACE	32
	DiffPair	HS-PORT_DP	Pin	U1.U5	HS-DIFFP	PHY:HS_DIFFP	2
			Pin	U1.U6			
	Region	HS-PORT_BGA			HS-BGA_REGION	BGA_AREA	
End							

Compiler Results

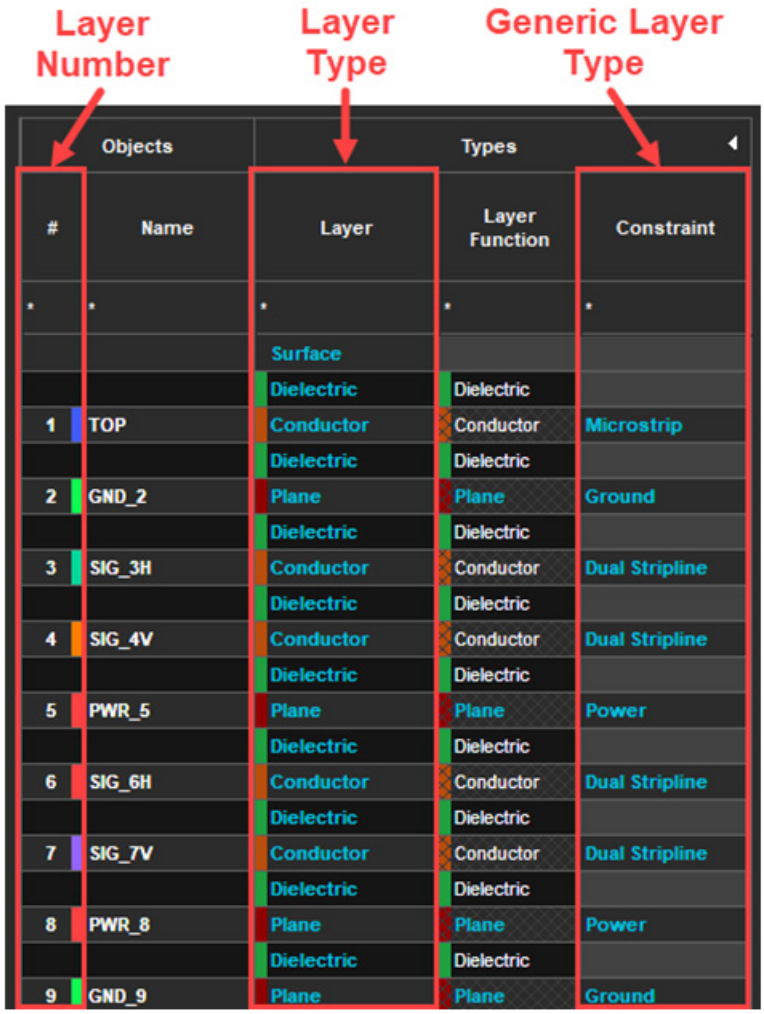
- Net Name string ending with multiple numbers within a number range specified within curly brackets "{ }"
- Spacing Net Class `HS-PORT_DATA` that contains nets `HD0, HD1, HD2, HD3, HD4, … HD31` assigned to rule set `HS_SPACE` from the `SPC` rule specification. When imported into Constraint Manager,
  - Net Class name starts with prefix `NC_`
  - Spacing CSet name starts with prefix `SCS_`
- Differential pair `HS-PORT_DP` member net names `HSON(0)` and `HSOP(0)` are extracted from the device pins `U1.U5` and `U1.U6` and then assigned to rule set `HS_DIFFP` from the `PHY` rule specification. When imported into Constraint Manager,
  - Differential pair name starts with the prefix `DP_`
  - Physical CSet name starts with the prefix `PCS_`
- Region `HS-PORT_BGA` creates physical and spacing region objects in Constraint Manager assigned to `BGA_AREA` rule specification
  - Basic physical and spacing constraints can be assigned to a region in an Object Table to control constraints for all nets entering the region
  - To create Region Class or Region Class to Class constraints specify them in an Object Rule Table using *Type* classification passed from the Object Table

  Note: Region object in Constraint Manager needs to be manually assigned to Constraint Region shape on the canvas.
  - When imported into Constraint Manager, physical CSet name starts with the prefix `PCS_` and spacing CSet name starts with prefix `SCS_`

Rule Specification Table

The Rule Specification table is used to capture Electrical, Physical and Spacing requirements to create an associated Constraint Set in Constraint Manager. Physical and Spacing rules

can be defined using existing Layer type or Generic layer designations in the stackup or by layer in the stackup using the Layer number(s).



The DEFAULT Physical and Spacing Constraint Sets in Constraint Manager can be used as a starting point with only the constraints entered in the table updated.

The compiler determines which Constraint Set type (domain) to be created based on entered rules in the Rule Specification. For example, specifying a “Width: Min Line”, “Line To Line”, and “Max Length” in one Rule Specification generates Physical, Spacing, and Electrical CSets for the three different constraint types.

In support of rules which can be present in multiple domains the Rule Specification type can be updated to include the target Domain (prefix to Rule):

- PhysicalRule,<table name> (Physical domain)

- SpacingRule,<table name> (Spacing domain)
- ElectricalRule,<table name> (Electrical domain)

Note: For example, Differential Pair rules can be defined in the Electrical Domain (all layer's rules) and Physical Domain (layer-specific rules)

A Rule Specification can be defined to create only one CSet for each domain or multiple CSets per domain by specifying different Rule Set names in the header using the Rule column (for example, Rule=<rule set name>).

Derived Constraint Set Support

By default, the Physical and Spacing DEFAULT CSet from the design is used as a starting point with the values in the table updated to form a new Constraint Set. A special key can be used in a rule specification to specify which CSet is used as a starting point.

- Table keys are the rows above the header row and below the table name row.

DefPhysicalCSet	Physical Constraint Set name in the design used as baseline for new Rule Set
DefSpacingCSet	Spacing Constraint Set name in the design used as baseline for new Rule Set

Column Header Description

Rule Specification Type	**Rule**, **PhysicalRule**, **SpacingRule** or **ElectricalRule** (Cell A1)
Layer Type or **Layer Index**	Layer Type = Conductor or Plane or Generic Layer Name "Conductor/External" Layer Index = **Layer Number**
Rule=<name>	Rule Set Name referenced in other tables **Rule Specification Name:Rule Set Name** or **Rule Specification Name**.
	Remove Rule Name (blank cell) will place all constraints in one Rule Set Name based on the Rule Specification Name
	Multiple Rule Specifications or Constraint Sets from different domains can be listed separated by a semicolon **";"**
Constraint Name	Constraint Attribute Name (**MIN_LINE_WIDTH**, **LINE_TO_LINE_SPACING**, **MAX_VIA_COUNT**)
	NOTE: Hover over header in Constraint Manager will report the Constraint Attribute Name in the Datatip
	NOTE: Custom Keyword mapping defined in Directive Global section of **CDS_ACC_DIRECTIVES.CSV** (Cadence Default) or **ACC_DIRECTIVES.CSV** (User Defined)

Sample Rule Specification

PhysicalRule	MY_PHY_RULES				
	Revision	1			
	Units	mil			
	Created by	Cadence			
Header	Layer Type	Rule=CSET1		Rule=CSET2	
		Width:Min Line	Width:Neck	Width:Min Line	Width:Neck
Data	Conductor	6	4	10	8
	Plane	12	6	15	10
End					

Compiler Results

- The default Physical CSet gets copied to `PCS_CSET1` with the following values:
  - For Conductor layers: `Min Line Width=6 mil` and `Neck Width=4 mil`
  - For Plane layers: `Min Line Width=12 mil` and `Neck Width=6 mil`
- The default Physical CSet gets copied to `PCS_CSET2` with the following values:
  - For Conductor layers: `Min Line Width=10 mil` and `Neck Width=8 mil`
  - For Plane layers: `Min Line Width=15 mil` and `Neck Width=10 mil`
- Rule Set name with prefix `PCS_` added to the new Physical CSet name

Note: Multi-Row Header `Rule=` designation indicates the beginning of a Rule Set with all constraint columns included in it until another `Rule=` designation starts.

Object Rule Specification Table

The Object Rule Specification table is used to assign rules to objects in a design. Rule assignment can be accomplished by referencing group or Kind name and type classification defined in an Object table or an existing Net or XNet in the design. The primary use for the Object Rule table is to define explicit pin to pin rules of a group of objects by specifying a from-to component designation. Once these pin-pairs are defined a rule set or an existing CSet can be applied to the appropriate object level (Net Class, Net Group, and so on).

You can specify one-to-many constraint expansion that propagate rules down to the lowest level while generating all supporting objects. For example, if relative propagation delay rules are defined, the compiler automatically creates the required match groups. In addition, part assignment can be driven by a mapping table to create many pin-pair relationships with associated match groups.

Note: Relative Propagation Delay rules are only supported when an explicit pin-pair is specified in the table using from component to component. Specifying global or local Scope or longest pin-pair, longest driver/receiver and all drivers/all receivers are ignored.

Column Header Description

Group or Kind	Type of Group defined in an Object Table or Net / XNet (leave blank if not required)
Name	Name of Group defined in an Object Table (required if Group or Kind is specified) or existing Net / XNet name existing in the design
	NOTE: Assigning rules to existing Net / XNet design objects are supported
	NOTE: Assigning a rule to a Net which is a member of a XNet will apply the rule to the XNet
Type	Type Classification assigned to objects in an Object Table
Topology Rule	Name of Topology table to be applied to Group or Type Classification for rule assignment
Routing Sections:Name	Routing Sections Name defined in the Topology Table for rule assignment (leave blank if it is for the entire Net)
From Comp	Used to generate Pin Pair assignments - Reference Designator or Reference Designator and Pin Number (Refdes.Pin#)
To Comp	Used to generate Pin Pair assignments - Reference Designator or Reference Designator and Pin Number (Refdes.Pin#)
Rule	**Rule Specification** or **Rule Specification:Rule Set** containing constraints for multiple domains
	Multiple Rule Specifications can be listed separated by a semicolon **";"**
Physical Rule	Reference existing Physical Constraint Sets name in the design (First Priority)
	Rule Specification or **Rule Specification:Rule Set** reference containing Physical Constraints
Spacing Rule	Reference existing Spacing Constraint Sets name in the design (First Priority)
	Rule Specification or **Rule Specification:Rule Set** reference containing Spacing Constraints
Electrical Rule	Reference existing Electrical Constraint Sets name in the design (First Priority)
	Rule Specification or **Rule Specification:Rule Set** reference containing Electrical Constraints
Constraint Names	Constraint Attribute Name (MIN_LINE_WIDTH, LINE_TO_LINE_SPACING, LINE_TO_THRUVIA_SPACING) or Keyword mapped alias to Constraint Attribute Name (Width:Min Line, Line To Line, Line To Thru Via)
	NOTE: Hover over header in Constraint Manager reports the Attribute Name in the Datatip
	NOTE: Custom Keyword mapping in Directive Global section of **CDS_ACC_DIRECTIVES.CSV** (Cadence Default) or **ACC_DIRECTIVES.CSV** (User Defined)
	NOTE: Match Groups will be automatically created If Relative Prop Delay rules are specified.

Sample Object Rule Table

ObjectRule	MY_OBJ_RULE					
	Revision	1				
	Units	mil				
	Interface	PCI				
	Manufacturer					
	Created by	Cadence				
Header					Relative Prop Delay	Prop Delay
	Group	Name	From Comp	To Comp	Delta:Tolerance	Max
Data	NetClass	HS-PORT_DATA	Part=U20	Part=U21	:50	2500
	NetClass	HS-PORT_ADDR	Part=U20	Part=U21	:100	4000
End						

Compiler Results

- Generates Match Group `HS-PORT_DATA` with the pin-pairs between `U20` and `U21` for Net Class `HS-PORT_DATA` with the following rules:
  - Relative Propagation Delay Tolerance = `50`
  - Max Propagation Delay = `2500`
- Generates match group `HS-PORT_ADDR` with the pin-pairs between `U20` and `U21` for Net Class `HS-PORT_ADDR` with the following rules:
  - Relative Propagation Delay Tolerance = `100`
  - Max Propagation Delay = `4000`
- Relative propagation delay with an empty `Delta` field matches all match group members within the `Tolerance`
  - Setting the `Delta` field to `0` to match all match group members to the longest pin to pin manhattan length within the defined `Tolerance` (+/- Tolerance)
  - Setting the `Delta` to anything other than `0`, on a specific net, gets treated as a +/- offset to the member which yields the longest pin to pin manhattan length within the `Tolerance`
- Propagation delay keeps the pin-pair connection below the Max length
- When imported into Constraint Manager, match group name have prefix `MG_` and based on the Net Class name

Object Rule Table (Match Group Support)

In many cases, Match Group membership is based on a group of nets that already exist in other groups (Net Group, Net Class or Bus). Referencing these groups in an Object Rule table, with Relative Propagation Constraints, automatically generates the required Match Group without the need of generating multiple groups with the same membership.

Naming a Match Group

- By default, the Match Group name use `MG_` as a prefix with the suffix being the group name containing the group of XNets (Group and Name columns).
  - Value specified in the Routing Section name column (Routing Section:Name) is leveraged as a suffix.
- Match Group name column (Relative Prop Delay:Group) can be specified as either a suffix or an explicit name for the Match Group.
  - If the name is referenced more than once and any occurrence is explicit, then they are all explicit.

Adding Members to a Match Group

- Reference a Group and Name or Type Classification in an Object Table representing a Group of XNet.
- By default, Pin Pairs are set to All Drivers/Receivers unless one of the following is specified:
  - Component From/To columns (From Comp/To Comp) populates Match Group with the Pin Pairs of the referenced objects.
  - Routing Section name column (Routing Section:Name) populates Match Group with Routing Sections (Pin Pairs for ratsnest connections) of the referenced objects.

  Note: Component From/To columns (From Comp/To Comp) and Routing Section name column (Routing Section:Name) should not be used at the same time in the same Data row.
- Specific XNets or Differential Pair that was not part of the initial Group, can be added to Match Group using the Match Group name column (Relative Prop Delay:Group). The value/name must be captured for both the Group and the XNets/Differential Pairs.

Constraining a Match Group

- Delta and tolerance can be specified using the Relative Prop Delay:Delta:Tolerance column.

- By default, the scope is set to Global (G) but can be changed to Local (L) using the Relative Prop Delay:Scope column.
- Match Group Target net can be specified using:
  - TARGET in the Relative Prop Delay:Delta:Tolerance column on a specific object.
  - Reference an object name in the Relative Prop Delay:Target column.
- For Match Groups which have XNets as members, the Pin Pairs column (Relative Prop Delay:Pin Pairs) can be used to validate auto-generated Pin Pairs. Valid values are as follows:
    - Longest Pin Pair = L:S
    - Longest Driver/Receiver = D:R
    - All Drivers/Receivers = AD:AR (Default when not specified in table)

Defining Match Group Membership in an Object Table

Match Groups as well as their membership can be created in an Object table but it is not recommended and should be the exception instead of standard practice. The following are some of the effects of creating Match Groups in an Object Table:

- Match Groups are created only with XNets as their members while the Relative Propagation Delay rules need to be defined in an ObjectRule Table.
  - Net Group membership could easily be leveraged in an ObjectRule table to generate Match Groups when Relative Propagation Delay rules are assigned
- Match Groups should only be constrained using the appropriate Driver/Receiver (Relative Prop Delay:Pin Pairs) and Delta/Tolerance (Relative Prop Delay:Delta:Tolerance) constraints in an ObjectRule Table.
  - Pin Pair based constraints driven by Routing Sections or From Comp and To Comp columns should not be used for Object table Match Groups

Object Rule Group and Name or Type Classification reference is a Group of XNets

Column in ObjectRule table				Results
Routing Sections	From Comp	To Comp	Relative Prop Delay	Match Group Name
Name			Group	
Not Set	Not Set	Not Set	Not Set	Group Name as suffix: **"MG_<Group Name>"**
N/A	**U1.1**	**U2.1**	Not Set	Group Name as suffix: **"MG_<Group Name>"**
BREAKOUT	N/A	N/A	Not Set	Group Name with **Routing Sections:Name** value as suffix: **"MG_<Group Name>_*BREAKOUT*"**
Not Set	Not Set	Not Set	***MG1***	**Relative Prop Delay:Group** value as suffix: **"MG_*MG1*"**
N/A	**U1.1**	**U2.1**	***MG1***	Group Name with **Relative Prop Delay:Group** value as suffix: **"MG_<Group Name>_*MG1*"**
BREAKOUT	N/A	N/A	***MG1***	Group Name with **Relative Prop Delay:Group** value as suffix: **"MG_<Group Name>_*MG1*"**

Object Rule Group and Name or Type Classification reference is a XNets or Diff Pairs

Column in ObjectRule table				Results
Routing Sections	From Comp	To Comp	Relative Prop Delay	Match Group Name
Name			Group	
Not Set	Not Set	Not Set	Not Set	N/A
N/A	**U1.1**	**U2.1**	Not Set	(X)Net Name or Diff Pair Name as suffix: **"MG_<(X)Net Name>"** or **"MG_<Diff Pair Name>"**
BREAKOUT	N/A	N/A	Not Set	**Routing Sections:Name** value as suffix: **"MG_*BREAKOUT*"**
Not Set	Not Set	Not Set	***MG1***	**Relative Prop Delay:Group** value as suffix **"MG_*MG1*"**
N/A	**U1.1**	**U2.1**	***MG1***	**Relative Prop Delay:Group** value as suffix: **"MG_*MG1*"**
BREAKOUT	N/A	N/A	***MG1***	**Relative Prop Delay:Group** value as suffix: **"MG_*MG1*"**

Object Rule Table Example (Match Group – From Comp/To Comp Common Target)

Header					Relative Prop Delay		
	Group	Name	From Comp	To Comp	Group	Delta:Tolerance	Target
Data	NetGroup	DDR_ADDR	Part=U34	Part=U26		0:5	DDR_CLK_N
	NetGroup	DDR_BYTE0	Part=U34	Part=U26	DQ_MEM1	0:5	
	Net	DDR_CLK_N	Part=U34	Part=U26	DQ_MEM1	TARGET	
End							

Compiler Results

- Match Group Name `MG_DDR_ADDR` with Pin Pair members (`U34` to `U26`) of NetGroup `DDR_ADDR` members and Net `DDR_CLK_N` in Target column
- Match Group Name `MG_DQ_MEM1` with Pin Pair members (`U34` to `U26`) of NetGroup `DDR_BYTE0` members and Net `DDR_CLK_N` with TARGET entered in Delta:Tolerance column.

Object Rule Table Example (Match Group – Topology/Routing Sections Common Target)

Header				Routing Sections	Relative Prop Delay		
	Group	Name	Topology Rule	Name	Group	Delta:Tolerance	Target
Data	NetGroup	DDR_ADDR	DDR_ ADDR _T	ASIC-MEM2		0:5	DDR_CLK_N
	NetGroup	DDR_BYTE1	DDR_DATA _T	BYTE1	DQ_MEM2	0:5	
	Net	DDR_CLK_N	DDR_ ADDR _T	ASIC-MEM2	DQ_MEM2	TARGET	
End							

Compiler Results

- Match Group Name `MG_DDR_ADDR_ASIC-MEM2` with Routing Sections Pin Pairs of NetGroup `DDR_ADDR` members and Net `DDR_CLK_N` in Target column.
- Match Group Name `MG_DQ_MEM2` with Routing Sections Pin Pairs of NetGroup `DDR_BYTE1` members and Net `DDR_CLK_N` with TARGET entered in Delta:Tolerance column.

Match Group members with different Delta/Tolerance values

To specify a unique Delta/Tolerance for member(s) inside of a Match Group (for example, Clock Diff Pair, Dual Diff Pair Btye lanes) can be done in an ObjectRule table by adding a separate row, after the initial Match Group row, to specify the net which requires a different Delta/Tolerance value. This row must include the Match Group name to ensure the updates are made appropriately.

Object Rule table example (Match Group – Clock Net with different Delta/Tolerance):

Header					Relative Prop Delay		
	Group	Name	From Comp	To Comp	Group	Delta:Tolerance	Target
Data	NetGroup	DDR_ADDR				0:5	
	Net	DDR_CLK_N			DDR_ADDR	0:10	
End							

Compiler Results

- Match Group Name `MG_DDR_ADDR` with NetGroup `DDR_ADDR` members will be matched to Delta:Tolerance value
- Net `DDR_CLK_N` will be added to Match Group `MG_DDR_ADDR` with the value entered in Delta:Tolerance column, if it is already part of the Match Group it will be updated.

Note: Net `DDR_CLK_N` must be added after the initial Match Group row to ensure that appropriate Delta:Tolerance is applied.

Object Rule table example (Match Group – 2nd Diff Pair with different Delta/Tolerance)

Header					Relative Prop Delay		
	Group	Name	From Comp	To Comp	Group	Delta:Tolerance	Target
Data	NetGroup	DDR_BYTE0				0:5	
	Net	DDR_DQS1_P			DDR_BYTE0	0:15	
	Net	DDR_DQS1_N			DDR_BYTE0	0:15	
End							

Compiler Results

- Match Group Name `MG_DDR_BYTE0` with NetGroup `DDR_BYTE0` members will be matched to Delta:Tolerance value
- Nets `DDR_DQS1_P` and `DDR_DQS1_N` are already members of Match Group `MG_DDR_ADDR` but will have the value entered in Delta:Tolerance column

Note: Nets `DDR_DQS1_P` and `DDR_DQS1_N` must be added after the initial Match Group row to ensure that appropriate Delta:Tolerance is applied.

Object Rule Table (Class to Class Rules)

The Object Rule Specification table can also be used to create Class to Class relationships between type classifications defined in an Object table. Objects grouped under a type classification are processed and generate the required spacing Net Classes to support the Class to Class spacing requirements. Each type-to-type relationship reference to either a rule specification, rule set or an existing spacing CSet (Spacing Set:<Set Name>) to apply an appropriate rule.

Column Header Description

Type	Type Classification assigned to objects in an Object Table (1st Class to Class Relationship)
	Update name to match Type Classification name in Object Table
Topology Rule	Name of Topology table to be applied to Group or Type Classification for rule assignment
Routing Sections:Name	Routing Sections Name defined in the Topology Table for rule assignment (leave blank if it is for the entire Net)
Spacing Rule	Specify an existing Constraint Set Name in Layout (First priority)
	Rule Specification or **Rule Specification:Rule Set** reference containing Spacing Constraints
	NOTE: Companion column header to "**Type=Intra-Group**" and "**Type=<Type>**" headers
Type=Intra-Group	Control spacing between members within **Type** (**Type** Column)
Type=<Type>	Controls spacing between **Type** (**Type** column) and **Type=<Type>** (2nd Class Relationship)
	Update "**<Type>**" in **Type=<Type>** to match Type Classification name in Object Table
	Add more **Spacing Rule** columns for addition Class to Class Relationships

Sample Object Rule Table (Class to Class Rules)

ObjectRule	**MY_CLASS**				
	Revision	1			
	Created by	Cadence			
Header			**Type=Intra-Group**	**Type=BYTE**	**Type=ADDR**
	Type	**Rule**	**Spacing Rule**	**Spacing Rule**	**Spacing Rule**
Data	BYTE	HS_SPACE:30MIL	HS_SPACE:6MIL	HS_SPACE:HS_15MIL	HS_SPACE:25MIL
End					

Compiler Results

- Spacing for the group of nets in Type `BYTE` gets checked to all other nets using rule set `30 mil` from the `HS_SPACE` rule specification
- Spacing for nets contained in Type `BYTE (Intra-Group)` gets checked to each other using rule set `6 mil` from the `HS_SPACE` rule specification
- Spacing for a group in Type `BYTE` gets checked to other Type `BYTE` groups using rule set `15 mil` from the `HS_SPACE` rule specification
- Spacing for each group in Type `BYTE` gets checked to Type `ADDR` groups using rule Set `25 mil` from the `HS_SPACE` rule specification
- Rule columns can also reference existing CSets SpacingCSet:<CSet Name>
- A Spacing Net Class with prefix `NC_` gets created using the group name defined in Object Table

Object Rule Table (Region Class/Class-Class Rules)

The Object Rule Specification table can also be used to create Region Class and Region Class to Class relationships between *Type* classifications defined in an Object table. Objects grouped under a *Type* classification are processed and generate the required Net Classes to support the Region Class/Class-Class spacing requirements. Each Type-to-Type relationship must reference either a Rule Specification, Rule Specification: Rule Set or an existing CSet so that the appropriate rule can be applied.

Column Header Description

Group or Kind	Enter **Region** when specifying existing Region name in the design in **Name** column (leave blank if not required)
Name	Name of Region in the design (required if Group or Kind is specified)
Type	Type Classification assigned to Region (**Group** and **Name**) in an Object Table
Physical Rule	Physical Region rules using Physical Constraint Set name in the design (First Priority)
	Rule Specification or **Rule Specification:Rule Set** containing Physical Constraints
Spacing Rule	Spacing Region rules using Spacing Constraint Sets name in the design (First Priority)
	Rule Specification or **Rule Specification:Rule Set** containing Spacing Constraints
Type=<Type>	Controls Class based Region Constraints (Net Class entering Region)
	Update "**<Type>**" to match Type Classification name in Object Table
	Specify **Spacing Rule** or **Physical Rule** in 2nd Row to align with entered Rule / CSet
	Add more columns for additional Region Class Physical or Spacing Constraints
Type=<Type1>:<Type2>	Controls Class to Class based Region Constraints (Class to Class interaction in Region)
	Update "**<Type1>**" and "**<Type2>**" to match Type Classification name in Object Table
	Specify **Spacing Rule** in 2nd Row to align with entered Rule / CSet
	Add more columns for additional Region Class to Class Spacing Constraints

Sample Object Rule Table (Region Class to Class Rules) of Region Created in Object Table

ObjectRule	**MY_REGION**					
	Revision	1				
	Created by	Cadence				
Header				**Type=BYTE**	**Type=ADDR**	**Type=BYTE:ADDR**
	Type	**Physical Rule**	**Spacing Rule**	**Physical Rule**	**Physical Rule**	**Spacing Rule**
Data	CONN_AREA	HS_PHY:4MIL	HS_SPC:4MIL	HS_PHY:6MIL	HS_PHY:7MIL	HS_SPC:15MIL
End						

Compiler Results

- Region `CONN_AREA` created in Object Table generates a region object in Constraint Manager

Note: Region object in Constraint Manager needs to be manually assigned to Constraint Region Shape on the canvas

- Physical constraints checked to all nets entering the region CONN_AREA using rule set 4MIL from the HS_PHY rule specification
- Spacing constraints checked to all nets in the region CONN_AREA using rule set 4MIL from the HS_SPC rule specification
- Physical constraints checked to group of nets in *Type* BYTE entering the region CONN_AREA using rule set 6MIL from the HS_PHY rule specification
- Physical constraints checked to group of nets in *Type* ADDR entering the region CONN_AREA using rule set 7MIL from the HS_PHY rule specification
- Spacing constraints for a group in *Type* BYTE are checked to other *Type* ADDR groups in the region CONN_AREA using rule set 15MIL from the HS_SPC rule specification
- Type designations result in a spacing net class with a NC_ prefix in layout using the group name defined in Object Table

Sample Object Rule Table (Region Class to Class Rules) of Existing Region in the Design

ObjectRule	MY_REGION					
	Revision	1				
	Created by	Cadence				
Header					Type=BYTE	Type=BYTE:ADDR
	Group	Name	Physical Rule	Spacing Rule	Physical Rule	Spacing Rule
Data	Region	BGA_AREA	HS_PHY:4MIL	HS_SPC:4MIL	HS_PHY:6MIL	HS_SPC:15MIL
End						

Compiler Results

- Existing region BGA_AREA in Constraint Manager generates the required class and class to class relationships so that the appropriate Constraint Sets can be assigned
- Physical constraints checked to all nets entering the region BGA_AREA using rule set 4MIL from the HS_PHY rule specification
- Spacing constraints checked to all nets in the region BGA_AREA using rule set 4MIL from the HS_SPC rule specification
- Physical constraints checked to group of nets in *Type* BYTE entering the region BGA_AREA using rule set 6MIL from the HS_PHY rule specification

- Spacing constraints for a group in *Type* `BYTE` are checked to other *Type* `ADDR` groups in the region `BGA_AREA` using rule set `15MIL` from the `HS_SPC` rule specification
- Type designations result in a spacing net class with a `NC_` prefix in layout using the group name defined in Object Table

Object Rule Table (Properties)

The Object Rule Specification table can also be used to assign Component, Pin and Net properties in design. Using an ACC table provides flexibility to assign related properties to drive interface guidelines alongside constraint definitions such as ROOM, PIN_DELAY, BACKDRILL_MAX_PTH_STUB, and so on.

Column Header Descriptions

Kind	Type of object for property assignment: **Part**, **Pin** and **Net**
Name	**Part**: Reference Designator using **${Alias}** mapped to **RefDes** OR Explicit **RefDes**
	Pin: Reference Designator and Pin Number using **${Alias}.Pin#** OR Explicit **Refdes.Pin#**
	Net: Explicit Net / XNet name
Property Names	Property Names (**ROOM**, **PIN_DELAY**, **BACKDRILL_MAX_PTH_STUB**, etc.)
	NOTE: Hover over header in Constraint Manager reports the Property Name in the Datatip

Object Rule Table (Component Property assignment)

ObjectRule	COMP_PROPS			
	Revision	1.0		
	Created by	Cadence		
Header	**Kind**	**Name**	**ROOM**	**ECSET_MAPPING_TAG**
Data	Part	U27	DDR0	MEM1
	Part	U29	DDR0	MEM2
	Part	U30	DDR0	MEM3
	Part	U12	DDR0	MEM4
End				

Compiler Results

- `ROOM` property assignment to drive placement in a particular region in design.
  - `U27, U29, U30` and `U12` assigned a `ROOM` Property with a value of `DDR0`.

- Used to drive placement in a particular room defined in design.

- ECSET_MAPPING_TAG property assignment to assist with topology mapping.
  - U27 assigned an ECSET_MAPPING_TAG property with a value of MEM1.
  - U29 assigned an ECSET_MAPPING_TAG property with a value of MEM2.
  - U30 assigned an ECSET_MAPPING_TAG property with a value of MEM3.
  - U12 assigned an ECSET_MAPPING_TAG property with a value of MEM4.
  - U27 assigned an ECSET_MAPPING_TAG property with a value of VIPER.

Object Rule Table (Pin Property assignment)

ObjectRule	PIN_PROPS			
	Revision	1.0		
	Units	mil		
	Created by	Cadence		
Header	**Kind**	**Name**	**PINUSE**	**PIN_DELAY**
Data	Pin	U27.E3	BI	42.907 ps
	Pin	U27.F7	BI	62.149 ps
	Pin	U27.F2	BI	58.395 ps
	Pin	U27.F8	BI	54.231 ps
	Pin	U27.H3	BI	58.697 ps
	Pin	U27.H8	BI	40.773 ps
	Pin	U27.G2	BI	61.183 ps
	Pin	U27.H7	BI	59.67 ps
End				

Compiler Results

- PINUSE Bi-Directional (BI) assigned to U27 Pins E3, F7, F2, F8, H3, H8, and G2.
- PIN_DELAY (Pin delays) applied to U27 Pins E3, F7, F2, F8, H3, H8, and G2.

Note: Units keyword in header used to indicate default length units for table. When pin delay value are in time (ns/ps) then time measurement should proceed value. (42.907 ps)

Object Rule Table (Net Property assignment)

ObjectRule	NET_PROPS				
	Revision	1.0			
	Units	mils			
	Created by	Cadence			
Header	Kind	Name	VOLTAGE	TESTPOINT_QUANTITY	BACKDRILL_MAX_PTH_STUB
Data	Net	DDR_VDDO	1.8	15	
	Net	DDR0_A4			10
	Net	DDR0_A5			10
	Net	DDR0_A6			10
End					

Compiler Results

- `VOLTAGE` Property with a value of `1.8V` assigned to power net `DDR_VDDO`.
- `TESTPOINT_QUANTITY` of `15` indicates number of test points for net `DDR_VDDO`.
- `BACKDRILL_MAX_PTH_STUB` of `10` indicates maximum stub requirements before backdrilling is required.

XML Constraint Data Schema

A complete data set in an XML format provides executable constraints for common interfaces and integrated chips, such as CPU, SoC, and PCH.The XML data sets are developed in collaboration with IC manufacturers that publish PCB Design guidelines for their chip sets.

Constraint data set can be used directly into Constraint Manager using the Constraint Compiler avoids the need of interpreting and extracting relevant information from PCB guidelines. Using data set saves the time spent in mapping the design data and manual effort of entering into Constraint Manager.

XML Constraint Data Set Format

To create specifications in new format, a corresponding schema (XSD) file is available to validate the structure of the created XML files.

The schema validation files exist in the installation directory at
<installation_directory>/share/pcb/consmgr/schemas/ACC_schemas

The following is a list of constraint data set:

Constraint Data Set	Used for ...
`Design/Design.xml` (High level design details)	Constraint units and Interface references
`Topology/Topology.xml` (Routing topologies	Topology by Interface, Route Section rules, general constraints
`Interface/Interface.xml` (Interface structure)	Pin and Net Groups, Match Groups and Diff Pairs
`Impedance/Impedance.xml` (Physical requirement)	Differential and Single-ended Impedance physical requirements
`Spacing/Spacing.xml` (Spacing requirement)	General spacing and Class to Class spacing requirements
`PCB-Stackup/PCB-Stackup.xml` (PCB Cross-Section structure)	Stack-up for impedance calculations driving physical/ spacing requirements Stack-up used as reference by the compiler to verify layer count match

Constraint Compiler Starter Templates

To assist in table creation, Microsoft Excel spreadsheet (`.xlsx`) template files are provided as a part of installation. You can copy, modify, and save these tables with new names in your constraint library.

Note: The tables should be saved in `.csv` format only using *File – Save As* command to be read constraint compiler.

The *Starter Table* worksheet tab contains the default column headers as well as a description of each at the bottom of the worksheet. This worksheet tab is where the table information should be entered.

Each template file has *Data Row Examples* worksheet tab that contains sample templates with specific examples based on the type of template file. Rule template files have additional worksheet tabs that provides the different constraint header names for each of the domains (Physical, Spacing, and Electrical).

The starter templates exist in the installation directory at *<installation_directory>\share\pcb\examples\acc\Starter_Templates.*

Table 10-1 Data Table Templates

Name of Data Table	Name of Template File
Mapping Name Table	`mapping_starter.xlsx`
Topology Table	`topology_starter.xlsx`
Object Table	`object_grouping_starter.xlsx`
Rule Specification Table (Rule Sets)	For all domains ■ `rule_spec_starter.xlsx` For Physical domain ■ `rule_spec_Phy_starter.xlsx` For Spacing domain ■ `rule_spec_Spc_starter.xlsx` For Electrical domain ■ Wiring, Vias and Total Etch Length Worksheets ❑ `rule_spec_Elec_WireViaEtch_starter.xlsx` ■ Differential Pair Worksheet ❑ `rule_spec_Elec_DiffPair_starter.xlsx` ■ Propagation Delay Worksheet ❑ `rule_spec_Elec_PropDelay_starter.xlsx` ■ Return Path Worksheet ❑ `rule_spec_Elec_Return_Path_starter.xlsx`
Object Rule Table (Object Rules)	`objectrule_spec_starter.xlsx`
Object Rule Table (Class to Class Rules)	`objectrule_class_to_class_starter.xlsx`

Object Rule Table (Region Class/Class-Class Rules)	`objectrule_class_to_class_starter.xlsx`
Object Rule Table (Component Properties)	`objectrule_Comp_Props_starter.xlsx`
Object Rule Table (Pin Properties)	`objectrule_Pin_Props_starter.xlsx`
Object Rule Table (Net Properties)	`objectrule_Net_Props_starter.xlsx`

The starter templates generated as separate files per table type for simplicity and instructional purposes but could be combined into one file.The Mapping Name Table is the only exception, recommended to remain separate for alias to design specific mapping.

Design Specific Mapping Name Table

Name	PCIE				
	Revision	2.0			
	Interface	PCIE			
Header	**Type**	**Alias**	**Design**		
Data	*	HSIO_X	J3_HSI:J3_HSO		
	*	HSIO_RX	J3_HSI		
	*	HSIO_TX	J3_HSO		
End					

Object and ObjectRule Table combined in one spreadsheet

Object	PCIE_HSIO				
	Revision	2.0			
	Interface	PCIE			
Header	**Group**	**Name**	**Member Kind**	**Member**	**Type**
Data	NetGroup	HSI_RX	Net	${HSIO_RX}[N,P]${LANE}	HSI_RX
	NetGroup	HSO_TX	Net	${HSIO_TX}[N,P]${LANE}	HSO_TX
End					
ObjectRule	PCIE_HSIO				
	Revision	2.0			
	Units	mil			
	Interface	PCIE			
	Group	**Name**	**Type**	**Type=Intra-Group**	**Type=HSI_RX**
Header				**Spacing Rule**	**Spacing Rule**
Data			HSO_TX	SPC:5MIL_SPACE	SPC:10MIL_SPACE
			HSI_RX	SPC:5MIL_SPACE	
End					

A

Retaining Electrical Constraints at Net Level

When an Xnet is created, all the electrical constraints on the nets that form the Xnet are moved from the nets to the Xnet. If the same electrical constraints exist on more than one of the nets comprising the Xnet, pre-defined rules determine how these constraints are combined to form a single constraint which is then added to the Xnet.

The electrical constraints are checked at the Xnet level rather than the net level. For example, on a net with a `TOTAL_ETCH_LENGTH` of `500 mils`, the constraint is checked by totalling the length of all the clines in the net. When the net becomes a member of an Xnet, the constraint is moved from the net to the owner Xnet and the constraint is checked by totalling the length of all the clines in each of the nets in the Xnet. Let's assume that you change or delete a signal model that is assigned to a component and it results in the destruction of the existing Xnet. In such a case, the electrical constraints assigned to the Xnet being destroyed are moved to each of the nets in the Xnet.

In 16.5, you can retain electrical constraints at the net level. This feature lets you optionally disable the process of moving electrical constraints from member nets to the owner Xnets when an Xnet is created and destroyed. You can control when to check a constraint at the net level or at the Xnet level. In essence, this feature helps you decide whether an electrical constraint continues to reside on the net or be moved to the Xnet it is assigned.

How to Retain Electrical Constraints at Net Level

The option to retain electrical constraints at the net level is disabled, by default, which means that the constraints are moved to the nets comprising the Xnet. You can opt to retain electrical constraints at net level using one of the following two methods:

- Setting CPM Directive
- Defining Allegro Environment Variable

Setting CPM Directive

1. Open the `.cpm` file.
2. In the `GLOBAL` section, add the following directive:

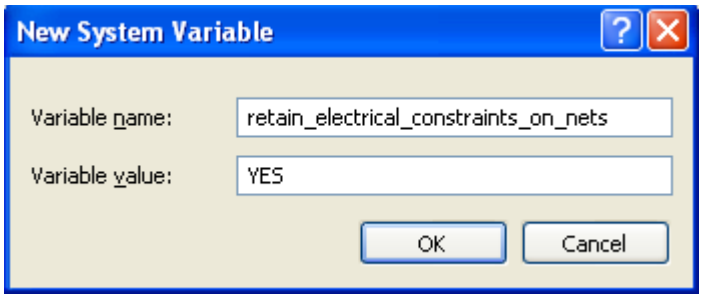
```
RETAIN_ELECTRICAL_CONSTRAINTS_ON_NETS  'YES'
```

This indicates that the option is turned on. A value of `NO` or the absence of this directive in the `.cpm` file indicates that the option is turned off.

Note: This value will be applicable to any new logic design created using Allegro Design Entry HDL or System Connectivity Manager. It will also be applicable to any new board design where the editor has been started with the `-proj` command line option that defines a `.cpm` file.

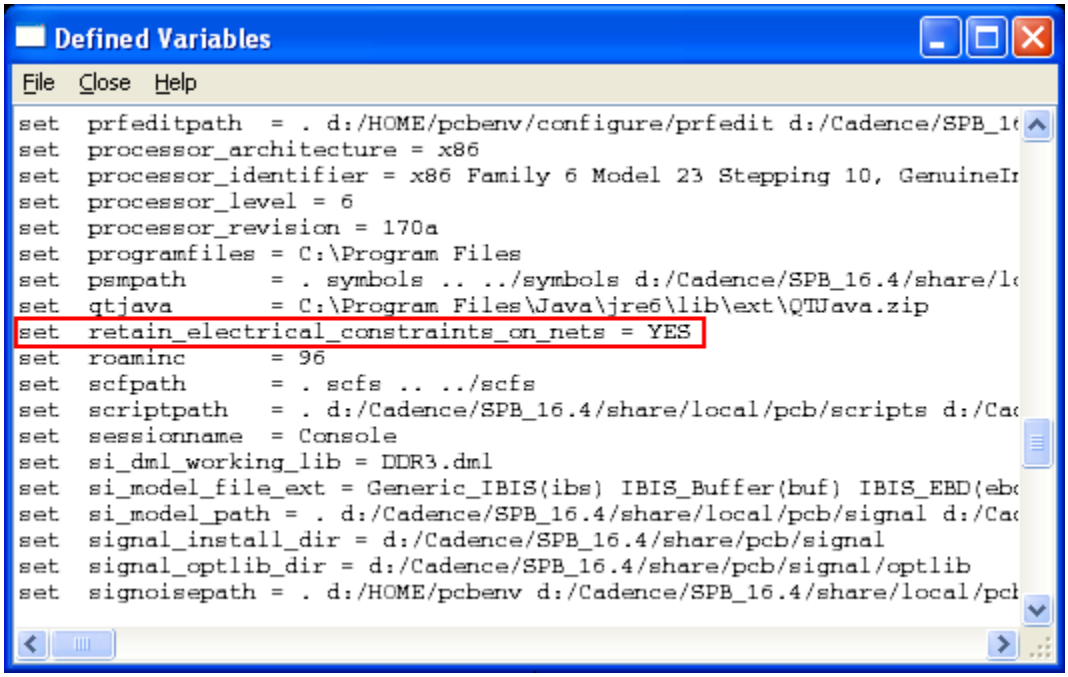
Defining Allegro Environment Variable

Set the `retain_electrical_constraints_on_nets` environment variable:



Note: This will only affect new designs created by a back-end tool.

You can confirm if the variable has been set, by choosing *Tools – Utilities – Variables* in PCB Editor.



Important

When starting a new logic design, only the CPM directive is checked. When starting a new board design, first the CPM directive is checked. If it is not found, the environment variable is checked. If none of these options is found, the directive is assumed to be off and there is no entry in the `.dcf` file.

Caution

Use this directive with extreme caution, because once you set the directive either in the `.cpm` file or in the `env` file, it is written in the database and is locked. Now the design will always manage the constraints at the net level. You cannot revert the design to its original state by removing the directive. To revert the design, the databases would need to be edited and all Xnet constraints would need to be checked.

Table A-1 Without the retain electrical constraints at net level option

Type	Objects	Referenced Electrical CSet	Total Etch Length			Total Etch Length		
			Min	Actual	Margin	Max	Actual	Margin
			mil	mil	mil	mil	mil	mil
FLTR	*	*	*	*	*	*	*	*
Dsn	⊟ bubble1				200			-1875
Net	NET1		100	2175	2075	300	2175	-1875
XNet	⊟ NET2		1500	1700	200	1800	1700	100
Net	NET2		1500			1800		
Net	NET2A		1500			1800		

Table A-2 With the retain electrical constraints at net level option

Type	Objects	Referenced Electrical CSet	Total Etch Length			Total Etch Length		
			Min	Actual	Margin	Max	Actual	Margin
			mil	mil	mil	mil	mil	mil
*	*	*	*	*	*	*	*	*
Dsn	⊟ nobubble1				0			-1900
Net	NET1		100	2100	2000	200	2100	-1900
XNet	⊟ NET2			1700	0		1700	-100
Net	NET2		900	900	0	1100	900	200
Net	NET2A		600	800	200	700	800	-100

Electrical Constraints on Nets and Xnets

In general, electrical constraints can have different values for the constraint on both a net and its owner Xnet. Each constraint is checked separately and DRC errors can be generated for each. The following section describes the exceptions to this rule:

Pin-Pair Constraints

The user-defined pin-pair constraints, which define specific pins of the net or Xnet are not affected by the new option. The constraint continues to be owned by the object to which the pin-pair belongs.

However, the auto-generated pin pair constraints, such as AD:AR and L:S are impacted by the option. These constraints are expanded on the fly when the constraint is checked. The pin-pairs that are selected depend on the object to which the constraint is applied. For example, an L:S constraint on a net selects the longest and the shortest pin-pair in that net. If the same constraint is on the Xnet that owns this net, the longest and shortest pin pair across the entire Xnet is selected.

Schedule/Stub Length

The Xnet constraints are ignored if the member nets of the Xnet are constrained.

Impedance

Explicit impedance constraints on pin-pairs follow the rules described above. Constraints captured on Xnets are ignored if the member nets of the Xnet are constrained.

ECSet Assignment

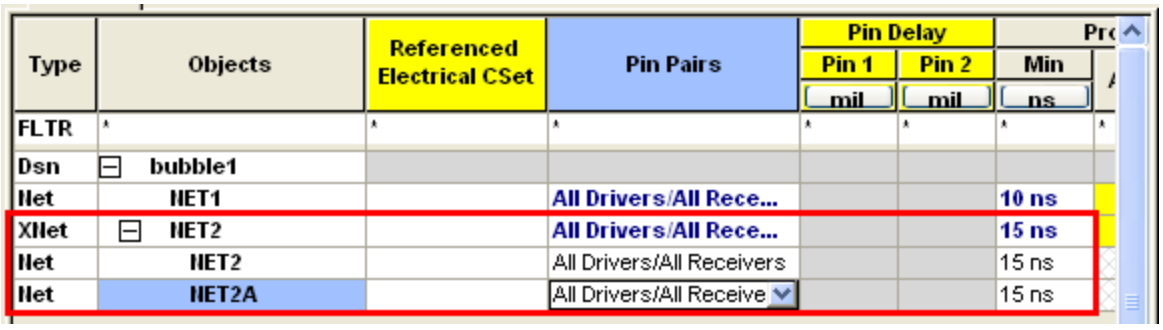
Similar to electrical constraints, ECSet assignment is also moved between member nets and the owner Xnet. If the retain electrical constraints at net level option is on, these assignments are not moved. You can assign separate ECSets for a net and its owner Xnet. If both ECSets contain topology data, the net is scheduled based on the topology data in the ECSet that is assigned to the net. Any net in the Xnet that does not have an ECSet assignment is scheduled based on the topology in the ECSet assigned to the owner Xnet. The rules for the pin scheduling based on an ECSet topology do not change.

Note: Any constraints in an ECSet assigned to a net only apply to the net. The constraints in the ECSet assigned to the owner Xnet only apply to the Xnet.

Note: This new option has no impact on the members of differential pairs and buses. If a net that is a member of a differential pair or bus becomes part of an Xnet, the Xnet always becomes a member of the differential pair or the bus.

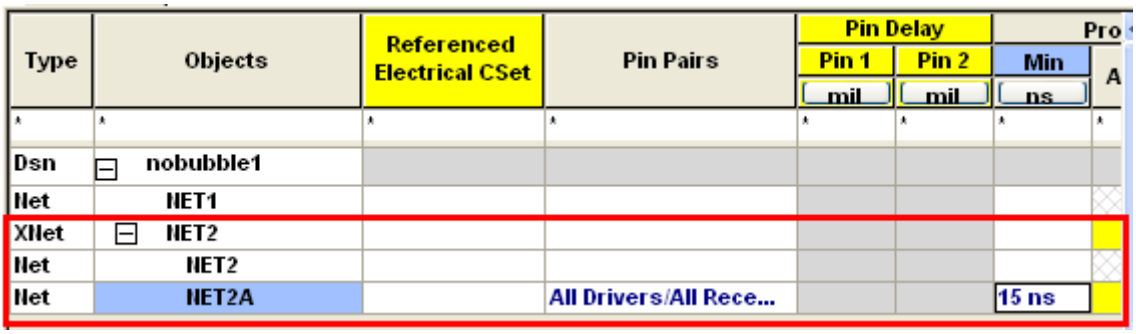
Constraint Manager Behavior

In Constraint Manager, if an electrical constraint is added to a net that is a member of an Xnet, the constraint is automatically moved to the Xnet.



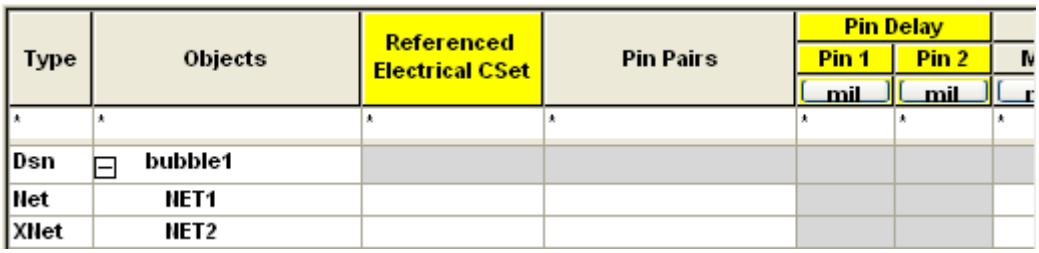
Type	Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Pr
				Pin 1	Pin 2	Min
				mil	mil	ns
FLTR	*	*	*	*	*	*
Dsn	⊟ bubble1					
Net	NET1		All Drivers/All Rece...			10 ns
XNet	⊟ NET2		All Drivers/All Rece...			15 ns
Net	NET2		All Drivers/All Receivers			15 ns
Net	NET2A		All Drivers/All Receive			15 ns

If the retain electrical constraints at net level option is turned on, the constraint remains on the member net.



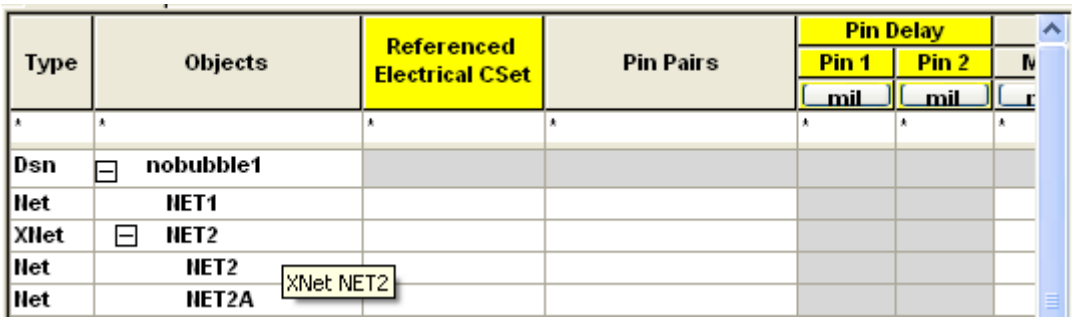
Type	Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay			Pro
				Pin 1	Pin 2	Min	A
				mil	mil	ns	
*	*	*	*	*	*	*	*
Dsn	⊟ nobubble1						
Net	NET1						
XNet	⊟ NET2						
Net	NET2						
Net	NET2A		All Drivers/All Rece...			15 ns	

By default, nets are not displayed as children of their Xnets in the electrical worksheets.



Type	Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		M
				Pin 1	Pin 2	
				mil	mil	r
*	*	*	*	*	*	*
Dsn	⊟ bubble1					
Net	NET1					
XNet	NET2					

If the retain electrical constraints at net level option is enabled, nets are displayed as children of their Xnet, by default. You can change this behavior from the Constraint Manager Filter dialog.



Type	Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		M
				Pin 1	Pin 2	
				mil	mil	r
*	*	*	*	*	*	*
Dsn	⊟ nobubble1					
Net	NET1					
XNet	⊟ NET2					
Net	NET2					
Net	NET2A					

If the retain electrical constraints at net level option is enabled, ECSet at the Xnet is not inherited by its member nets.

Type	Objects	Referenced Electrical CSet	Total Etch Length			Total Etch Length			U
			Min	Actual	Margin	Max	Actual	Margin	
			mil	mil	mil	mil	mil	mil	
*	*	*	*	*	*	*	*	*	*
Dsn	⊟ nobubble1				620			-1900	
Net	NET1		100	2100	2000	200	2100	-1900	
XNet	⊟ NET2	ECS1	150	1700	1550	250	1700	-1450	
Net	NET2		170	900	730	190	900	-710	
Net	NET2A		180	800	620	200	800	-600	

Important

When exporting a DCF, the retain electrical constraints at net level option is written to the DCF file. The option is not written to any other file including technology, actuals, and worksheet. When importing a DCF, the option is compared to the setting in the current design. If the option in the DCF does not match the setting in the design, the Import will generate an error and not continue.

Front-to-Back Flow

The new option is only processed in the front-to-back (F2B) flow for new designs. When a board is created with the -proj command line option, the new board is created with the retain electrical constraints at net level option as defined in the CPM file.

Similarly, if the corresponding environment variable is specified, it is processed for the new design. If the option differs between front end and back end for an existing design, the F2B flow fails. You need to update either the FE or BE before you re-run the F2B flow

Back-to-Front Flow

The new option will not be processed in the back-to-front (B2F) flow. If the option differs between front end and back end for an existing design, the B2F flow fails. You need to update either the FE or BE before you re-run the B2F flow.

Show Element Command in PCB Editor

The Show Element command that is available in all of the Allegro-based editors lists the electrical constraints that are assigned to a net being displayed. With the retain electrical

constraints at net level option on, If the net is a member of an Xnet both the net level constraints and the Xnet level constraints are listed.