cādence®

Allegro® Design Entry HDL – Constraint Manager User Guide

Product Version 23.1
September 2023

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

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence’s trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission. All other trademarks are the property of their respective holders.

Restricted Permission: This publication is protected by copyright law and international treaties and contains trade secrets and proprietary information owned by Cadence. Unauthorized reproduction or distribution of this publication, or any portion of it, may result in civil and criminal penalties. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. Unless otherwise agreed to by Cadence in writing, this statement grants Cadence customers permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used only in accordance with a written agreement between Cadence and its customer.
2. The publication may not be modified in any way.
3. Any authorized copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement.
4. The information contained in this document cannot be used in the development of like products or software, whether for internal or external use, and shall not be used for the benefit of any other party, whether or not for consideration.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information. Cadence is committed to using respectful language in our code and communications. We are also active in the removal and/or replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

Preface

About This User Guide

The *Allegro® Design Entry HDL - Constraint Manager User Guide* explains how to use the Allegro® Design Entry HDL schematic editor with Constraint Manager for managing electrical constraints.

This user guide assumes that you are familiar with the development and design of electronic circuits at the system or board level.

Finding Information in This User Guide

This user guide covers the following topics:

See...	For Information About...
Introduction to Constraint Manager	Provides an overview of Constraint Manager and various net objects, such as XNets, buses, match groups, and so on. Also, describes the user interface of Constraint Manager.
Creating XNets and Differential Pairs	Describes how to create XNets and model-defined differential pairs using signal models.
Synchronizing Constraints	Describes how to synchronize constraints between Design Entry HDL and Constraint Manager.
Electrical Constraints	Describes how to use Constraint Manager with Design Entry HDL for managing electrical constraints

See...	For Information About...
Physical and Spacing Constraints	Describes how Constraint Manager, launched from a design capture tool can be used to capture, edit, and view physical and spacing constraints.
ECSet in SigXplorer	Describes how to view and modify net topology and constraints in SigXplorer.
Constraints in Hierarchical Designs	Describes how to work with lower-level constraints in hierarchical designs.
Restoring Constraints from Definition	Describes how to restore lower-level constraints from their original definitions.
Working with Properties	Describes how to manage and work with properties in Constraint Manager.
Migrating from Previous Releases to the Current Releases	Describes the issues you might encounter when migrating from one release to another and how to address those issues.
Frequently Asked Questions About the Constraint Manager Flow	Lists the most frequently asked questions about the Constraint Manager flow.
Glossary	Provides definitions of the common terms used in the Design Entry HDL - Constraint Manager flow.
Troubleshooting Design Entry HDL to Constraint Manager Flow	Lists the solutions to some common or intermittent problems encountered while using Allegro Design Entry HDL with Constraint Manager.
Property Mapping	Depicts the cell name, as shown in Constraint Manager, with the associated property that is stored in the schematic, board, or package database
Recommendation for Allegro Design Entry HDL — Constraint Manager Flow	Defines the Cadence-recommended flow for a constraint-enabled schematic design captured in Allegro Design Entry HDL and Allegro PCB Editor

Related Documentation

You can also refer the following documentation to know more about related tools and methodologies:

Design Entry HDL

- For information on the new features in this release, see Allegro Design Entry HDL: What's New.
- For learning Design Entry HDL, see *Allegro Design Entry HDL Tutorial*.
- For learning to use Constraint Manager with Design Entry HDL, see *Allegro Constraint Manager with Design Entry HDL Tutorial*.

Front-to-Back Flow

- For information on the front-to-back flow for PCB design, see *Allegro Front-to-Back User Guide*.
- For information on the Design Synchronization solution, see *Design Synchronization and Packaging User Guide and Design Synchronization Tutorial*.
- For information about packaging your design, see the *Packager-XL Reference*.

1

Introduction to Constraint Manager

What is Constraint Manager?

A constraint is a user-defined requirement applied to a net or pin-pair in a design. Electrical constraints (ECs) govern the electrical behavior of a net or pin-pair in a design. For example, you can capture a constraint to define the maximum voltage overshoot tolerated by a net and capture the minimum first switch delay for a driver-receiver pin-pair in your design.

To capture constraints, Cadence provides a tool named Constraint Manager. You can use Constraint Manager with Cadence logic design tools, System Connectivity Manager and Design Entry HDL, to capture and manage electrical constraints as you implement logic. Constraint Manager is well integrated with the logic design tools. As a result, the changes that you make to constraint information in Constraint Manager are displayed in the logic design tools. Similarly, the changes that you make to constraint information in the logic design tools are displayed in Constraint Manager.

Why Constraint Manager?

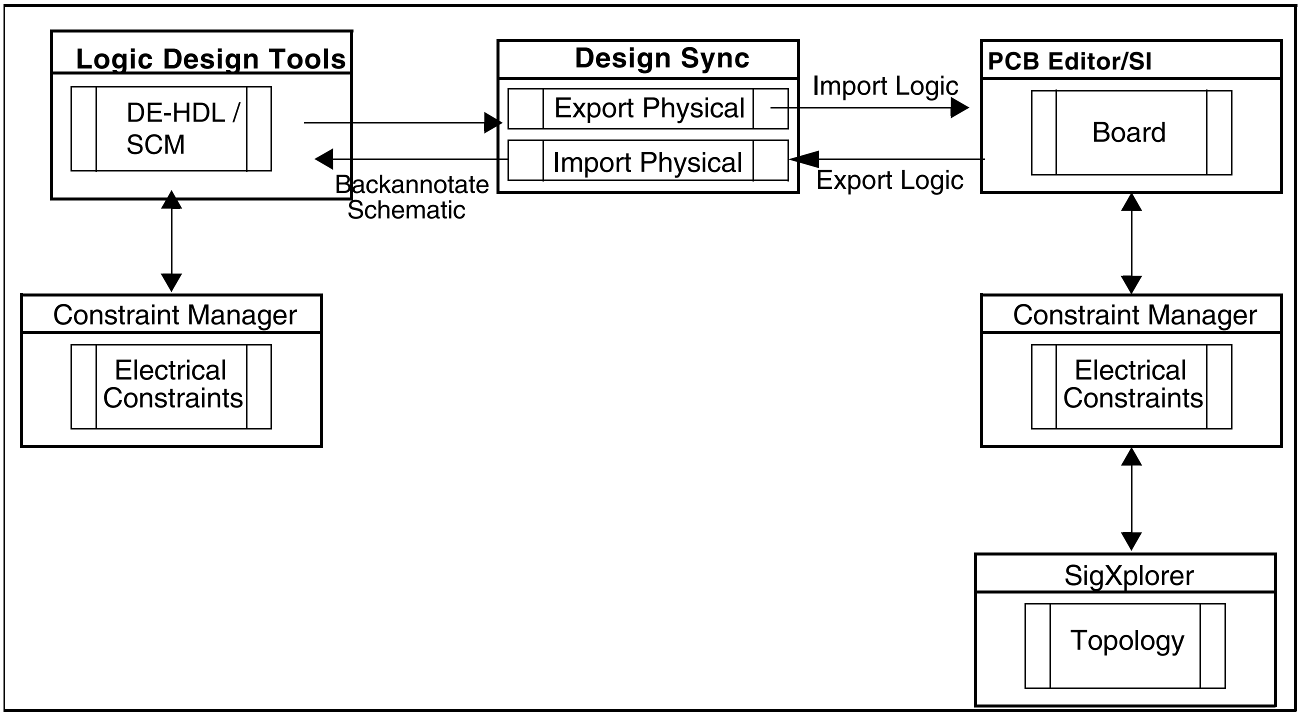
Using Constraint Manager to capture design constraints has the following advantages:

- It provides a spread-sheet based user interface that allows you to quickly capture, modify, and delete constraints.
- It supports syntax checking for all constraints.
- It supports constraint inheritance. The constraints captured on a schematic block are inherited by the design in which the block is instantiated.
- It lets you create Electrical Constraint Sets (ECSets)—a collection of electrical constraints that define a particular design requirement— and assign them to objects on which you want to capture the same set of constraints. For example, you can create an ECSet to define the default timing and noise tolerance for a net. An Electrical Constraint Set applies to an individual net although other nets may contribute to the measure of the constraint (for example, to crosstalk).

Modifying an ECSet automatically applies to all objects on which the ECSet is assigned.

- It lets you generate reports on constraints captured in the logical design and in the board.

Concurrent Capture of Constraints in Logical Design and Board



Constraint Objects

The Allegro platform has a collection of constraints that you can use to control all aspects of a design, including electrical, physical, spacing, and manufacturing parameters. The usual design practice is to capture electrical constraints during the design capture process. These constraints are then passed to the physical design process. Physical, spacing and manufacturing constraints are imposed during the physical design phase.

This section provides a brief description of the constraint objects used in Constraint Manager, when it is launched from a design capture tool. In the Constraint Manager user interface, the object type is listed in the TYPE column of all the worksheets. Acronyms are used to indicate the object type. Table 1-1 on page 17 lists the acronym used for each object type that is available when you launch Constraint Manager from a design capture tool. You can hover your mouse cursor over constraint object names in the user interface to view context-sensitive data tips, such as whether a constraint is inherited, the c-path, and so on.

Table 1-1

Constraint Object	Indicated in System Connectivity Manager as...
Bus	Bus
Design	Dsn
Design Instance	DsnI
Differential Pair	Dpr
ECSet	ECS
ECSet created Match Group	ECSM
Match Group	mgrp
Net	Net
Pin Pair	PPr
XNet	XNet

For more information on constraint objects listed in this section, see *Allegro Constraint Manager User Guide*.

Constraints

The Allegro platform organizes constraints by domain (Electrical, Physical, Spacing, and Design), depending on the nature of the constraint.

Electrical Constraints

An electrical constraint is a rule that characterizes and constrains the electrical behavior of a net. Impedance and propagation delay are examples of electrical constraints.

Physical Constraints

A physical constraint is a rule that characterizes and constrains the physical instantiation of a net. Minimum line width and minimum neck width are examples of physical constraints that apply to the connect lines (clines) of a net.

Spacing Constraints

A spacing constraint is a rule that specifies the spacing between two physical objects. These objects may be the physical instantiation of the same net or different nets. Line-to-line spacing is an example of a spacing constraint that applies between two connect lines (clines).

Note: Physical and Spacing constraints are not available when Constraint Manager is launched from design capture tools.

Design Constraints

A *Design* constraint is a rule that applies to the entire physical design. These constraints are independent of the electrical circuit. Soldermask alignment and testpoint spacing are examples of *Design* constraints.

Constraint Sets

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

Physical and 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. In all designs, the tool provides one Physical- and one Spacing-CSet, named *DEFAULT,* and each constraint within the CSet has a pre-defined value. You can set the value of any constraint within a Physical- and Spacing-CSet, but you cannot remove or add constraints to the CSet, nor can you delete the CSet.

Note: Physical and Spacing CSets are not available when Constraint Manager is launched from design capture tools.

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 an Electrical CSet.

Constraint Override

A constraint *override* occurs when an individual constraint value is applied to a member of a constraint object and it overrides the inherited constraint value. A constraint override has higher precedence than a CSet.

Important

> Throughout the Allegro platform, the terms *constraint*, *property* and *attribute* are sometimes mistakenly used interchangeably. Consider a *property* or an *attribute* as a mechanism to store supplemental information on an object that serves to augment your design. Consider a *constraint* as a rule that is validated by the constraint system.

Net Class

A Net Class constraint object lets you group net objects (clines, shapes, pins, and vias) that share common characteristics and require a similar constraint requirement. Allowable members of a Net Class include buses, differential pairs, XNets, and nets.

Net Classes apply to the *Electrical*, *Physical*, and *Spacing* domains and are constrained by one of the following:

- referencing a CSet
- a setting that overrides the *Net Class* constraint object

Differential Pairs

A *differential pair* constraint object represents a pair of XNets or nets that is routed differentially. The opposite signals of the pair, which share the same reference, cancel out any electromagnetic noise in the circuit.

Important

> The DIFFP_PRIMARY_GAP, MIN_LINE_WIDTH, DIFFP_NECK_GAP and MIN_NECK_WIDTH properties are allowed on both, ECSets, and PCSets; however, the ECSet value takes precedence over the PCSet value.

The Allegro platform 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 PCB Editor, Allegro X Advanced Package Designer (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.

Match Groups

A match group constraint object is a collection of nets, XNets, or pin pairs that must all match (in *delay* or *length*) or be relative to a specific *target* within the group.

You can create match groups manually by adding properties directly to XNets, or by using Constraint Manager to explicitly define pin pairs in an ECSet. Regardless of the method that you choose, match groups are resolved down to specific pin pairs.

Scope of Match Group

Scope controls the validation of the match group. You can create a match group with one of the following scopes:

- Local
- Global
- Bus
- Class

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. 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.

Global

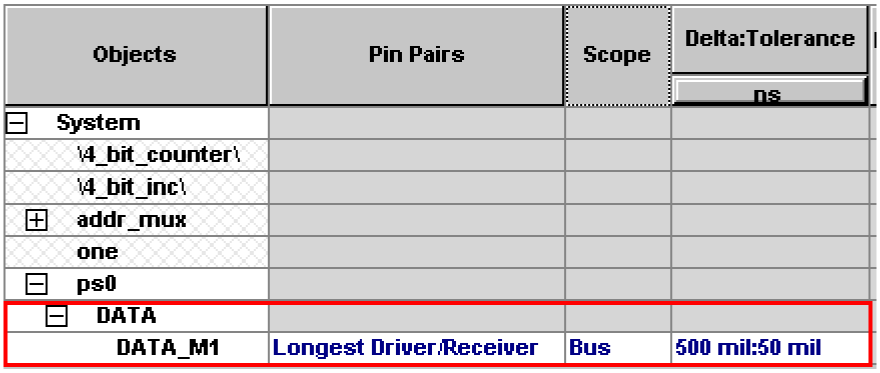
Validates all pin pairs against all other pin pairs in the match group. With a global scope, Constraint Manager validates pin pairs collectively in the (single) match group.

Bus

Generates unique match groups based on the way the target nets are grouped in the design. All pin pairs within the same bus and within the same match group are matched in such match groups. Using bus scope, a single ECSet can be used to generate multiple unique match groups, thereby optimizing the number of topologies required to constraint a design.

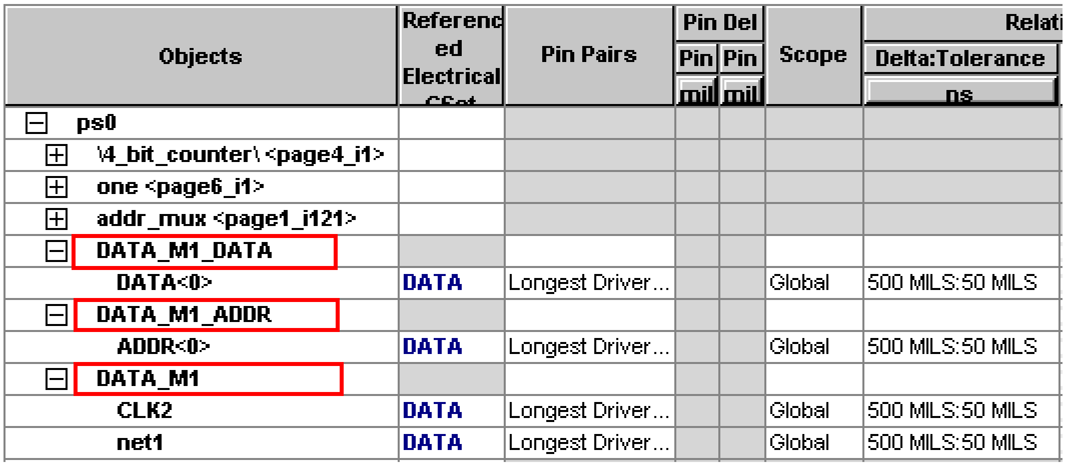
Bus scope is useful in flat designs and replicated blocks where buses (groups of signals) are replicated and the same ECSet needs to be applied to these buses. In the absence of the bus scope, to constrain these buses, you would require multiple identical ECSets that only differ by the match group name.

When you apply an ECSet to a bus, each bus inherits constraints from the ECSet. For each bus referencing an ECSet, a unique match group is created, where each bus member can be matched to the other member nets. The match group is created with a name derived from the ECSet name and the name of the bus to which the net belongs. In case of a non-bus member, Constraint Manager retains the original name of the match group. In the following example, note the ECSet definition of the ECSet `DATA_M1`.



Objects	Pin Pairs	Scope	Delta:Tolerance
			ns
⊟ System			
\4_bit_counter\			
\4_bit_inc\			
⊞ addr_mux			
one			
⊟ ps0			
⊟ DATA			
DATA_M1	Longest Driver/Receiver	Bus	500 mil:50 mil

- ❑ If the `DATA_M1` ECSet is assigned to a net in the `DATA` bus, the net will be added to the `DATA_M1_DATA` match group.
- ❑ If the net belongs to another bus, let's say `ADDR`, a new match group is created, `DATA_M1_ADDR`.
- ❑ If the ECSet is applied to a non-bus member, the net is added to the `DATA_M1` match group. All the other non-bus members referencing this ECSet will be added to this match group.



Objects	Referenced Electrical CSet	Pin Pairs	Pin Del		Scope	Relati
			Pin	Pin		Delta:Tolerance
			mil	mil		ns
⊟ ps0						
⊞ \4_bit_counter\ <page4_i1>						
⊞ one <page6_i1>						
⊞ addr_mux <page1_i121>						
⊟ DATA_M1_DATA						
DATA<0>	DATA	Longest Driver...			Global	500 MILS:50 MILS
⊟ DATA_M1_ADDR						
ADDR<0>	DATA	Longest Driver...			Global	500 MILS:50 MILS
⊟ DATA_M1						
CLK2	DATA	Longest Driver...			Global	500 MILS:50 MILS
net1	DATA	Longest Driver...			Global	500 MILS:50 MILS

Important

In the snapshot above, note that the match groups appear under the lower-level blocks to which the bus or the nets belongs.

Note: You must specify a bus scope for a match group within an ECSet in either the *ECSet folder* or in SigXplorer. You can then apply the ECSet to a bus or a net in the Relative Propagation Delay worksheet. Although you define bus scope at the *ECSet* level, when the ECSet is applied to a bus member at the *Net* level, the Scope column indicates *Global*.

Class

Generates unique match groups for each class. Similar to bus scope, class scope also optimizes the number of topologies required to constraint a design. However, no other signal from outside the net class can be added to a match group with class scope.

Why Class Scope?

Match groups created in a lower-level block of a hierarchical design are global in the context of the top-level design, which means that the nets in a match group can be matched with any other net in the entire design. However, the requirement in most of the designs is that the nets in a match group created in a hierarchical block must match the nets only within that block. For example, in a design, you might want to match the nets of the address bus with some control signals and clocks for a specific interface, but not across multiple instances of the same interface block. While the bus scope works in instances where the bus is contained in a hierarchical block, you cannot add any additional nets to the match group created by the bus scope.

To address this issue, a scope called class scope was introduced in release 16.01, which is in essence the same as the bus scope. Similar to the bus scope, the class scope also reduces the number of topologies (ECSets) required to constraint a design.

What is Class Scope?

A class scope matches all the signals within a class and within the same match group by creating a unique match group name for each class. Class scope is limited to ECSets and is only used during the ECSet mapping process.

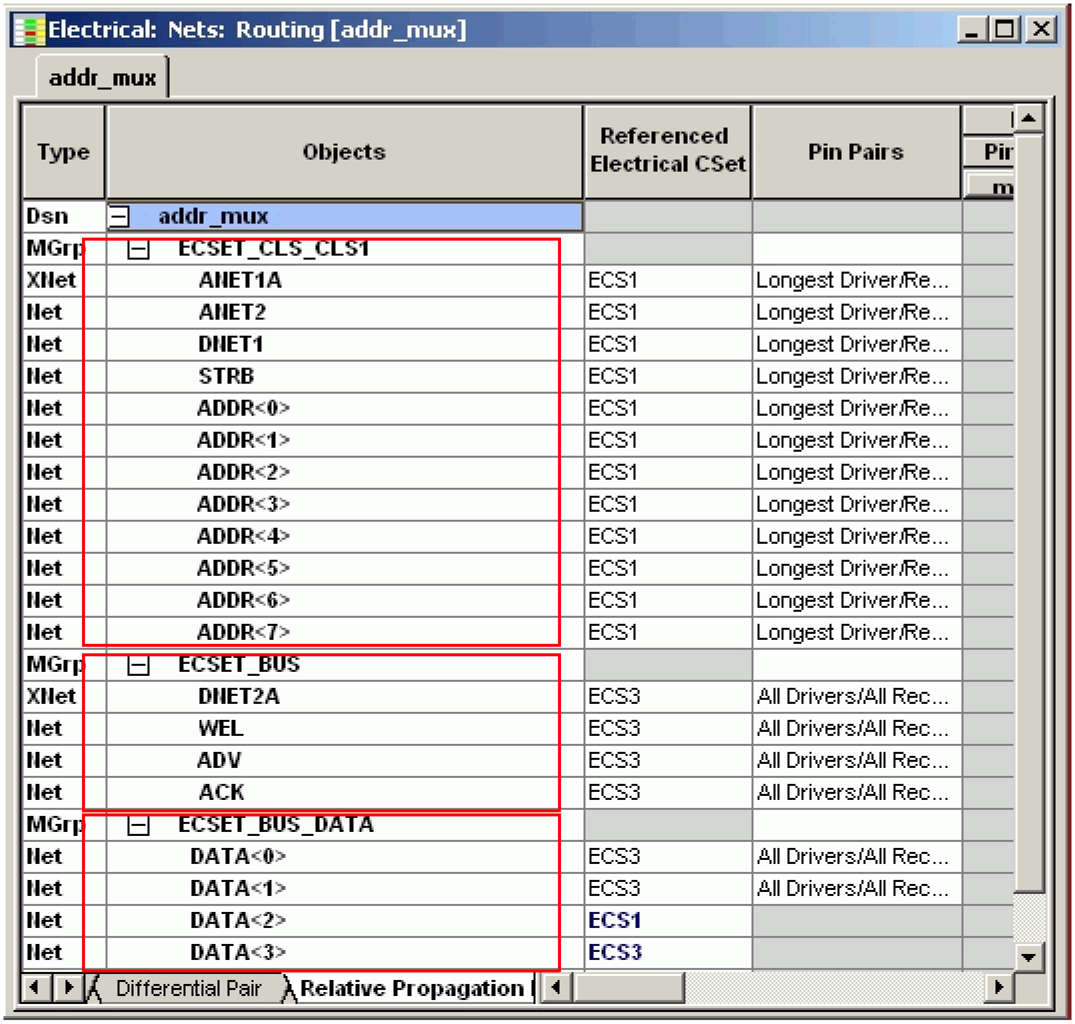
When you apply an ECSet to a class, each class inherits constraints from the ECSet. For each class referencing an ECSet, a unique match group is created, where each member of the class can be matched to the other member nets. The match group is created with a name derived from the ECSet name and the name of the class to which the net belongs.

When to Use Class Scope?

The class scope has more flexibility than the bus scope because a class can include more signals than a bus, which is typically limited to vectored nets or nets that share a common topology. Unlike the bus scope, the class scope adds all the selected members, including bus members, to the match group created by the ECSet (with class scope). When you need the functionality of a bus scope and also need additional non-bus members in a match group, use the class scope.

In the following example, an ECSet-created match group—`ECSET_BUS`–with the scope defined as BUS is assigned to the `CLS2` class containing an XNet, a net, and a bus. Two match groups are created, `ECSET_BUS_DATA` containing only the bus members, and `ECSET_BUS` with the remaining members of `CLS2`. On the other hand, when the ECSet-created match group `ECSET_CLS` with scope defined as CLASS is assigned to the `CLS1` class containing an XNet, a net, and a bus, only one match group—`ECSET_CLS_CLS1`—is created. `ECSET_CLS_CLS1` contains all the members of `CLS1` including the bus `ADDR` and its members.

Figure 1-1 Class Scope



Type	Objects	Referenced Electrical CSet	Pin Pairs	Pir m
Dsn	addr_mux			
MGrp	ECSET_CLS_CLS1			
XNet	ANET1A	ECS1	Longest Driver/Re...	
Net	ANET2	ECS1	Longest Driver/Re...	
Net	DNET1	ECS1	Longest Driver/Re...	
Net	STRB	ECS1	Longest Driver/Re...	
Net	ADDR<0>	ECS1	Longest Driver/Re...	
Net	ADDR<1>	ECS1	Longest Driver/Re...	
Net	ADDR<2>	ECS1	Longest Driver/Re...	
Net	ADDR<3>	ECS1	Longest Driver/Re...	
Net	ADDR<4>	ECS1	Longest Driver/Re...	
Net	ADDR<5>	ECS1	Longest Driver/Re...	
Net	ADDR<6>	ECS1	Longest Driver/Re...	
Net	ADDR<7>	ECS1	Longest Driver/Re...	
MGrp	ECSET_BUS			
XNet	DNET2A	ECS3	All Drivers/All Rec...	
Net	WEL	ECS3	All Drivers/All Rec...	
Net	ADV	ECS3	All Drivers/All Rec...	
Net	ACK	ECS3	All Drivers/All Rec...	
MGrp	ECSET_BUS_DATA			
Net	DATA<0>	ECS3	All Drivers/All Rec...	
Net	DATA<1>	ECS3	All Drivers/All Rec...	
Net	DATA<2>	ECS1		
Net	DATA<3>	ECS3		

How to Use Class Scope?

- Creating an ECSet with Class Scope
- ECSets with Class Scope in the Context of a Top-Level Design
- ECSets with Class Scope in Constraint Manager Connected to Allegro PCB Editor

Creating an ECSet with Class Scope

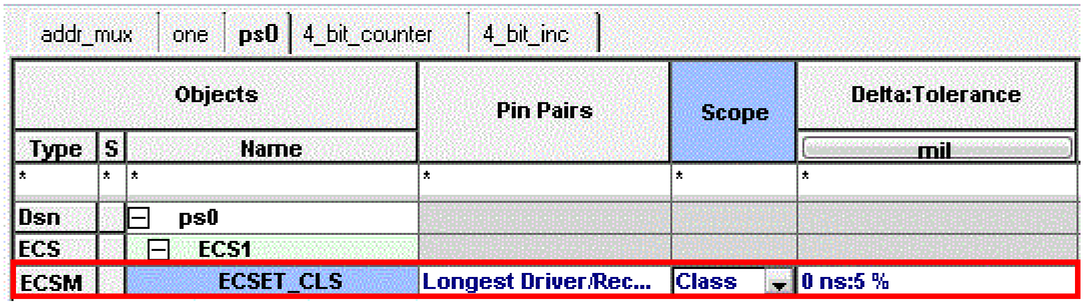
In the procedures that follow, we'll consider an example of a design `ps0`, which contains a hierarchical block `addr_mux`, which in turn contains the following objects:

- Nets - STRB, ADV, ACK, WEL, ANET1A, DNET1

- XNets - ANET1A, DNET2A
- Differential Pairs: DP1 and DP2
- Buses - ADDR, DATA
- Classes - CLS1, CLS2

To create an ECSet with the class scope, do the following:

1. Create an ECSet
   - **a.** Launch Constraint Manager on a design.
   - **b.** Select the *Electrical Constraint* folder.
   - **c.** Select the *Routing—Relative Propagation* worksheet.
   - **d.** Choose *Object – Create – Electrical Cset*.
   - **e.** Specify the name of the ECSet as *ECS1*.
   - **f.** Click *OK*.
   - **g.** Right-click *ECS1* on the worksheet and choose *Create – Match Group* from the pop-up menu.
   - **h.** Specify *ECSET_CLS* in the Create Electrical Cset Match Group dialog box.
   - **i.** Click *OK*.
   - **j.** Specify *Longest Driver/Receiver* as the pin pair value.
   - **k.** Select *CLASS* as the scope.



Objects			Pin Pairs	Scope	Delta:Tolerance
Type	S	Name			mil
*	*	*	*	*	*
Dsn		⊟ ps0			
ECS		⊟ ECS1			
ECSM		ECSET_CLS	Longest Driver/Rec...	Class	0 ns:5 %

Tip

Alternatively, you can right-click on *CLS1* under the *Net – Routing* worksheet in the *Electrical Constraints* tab and select SigXplorer from the pop-up menu then create the ECSet match group in the *Relay Prop Delay* tabbed page of the Set Topology Constraints dialog box.

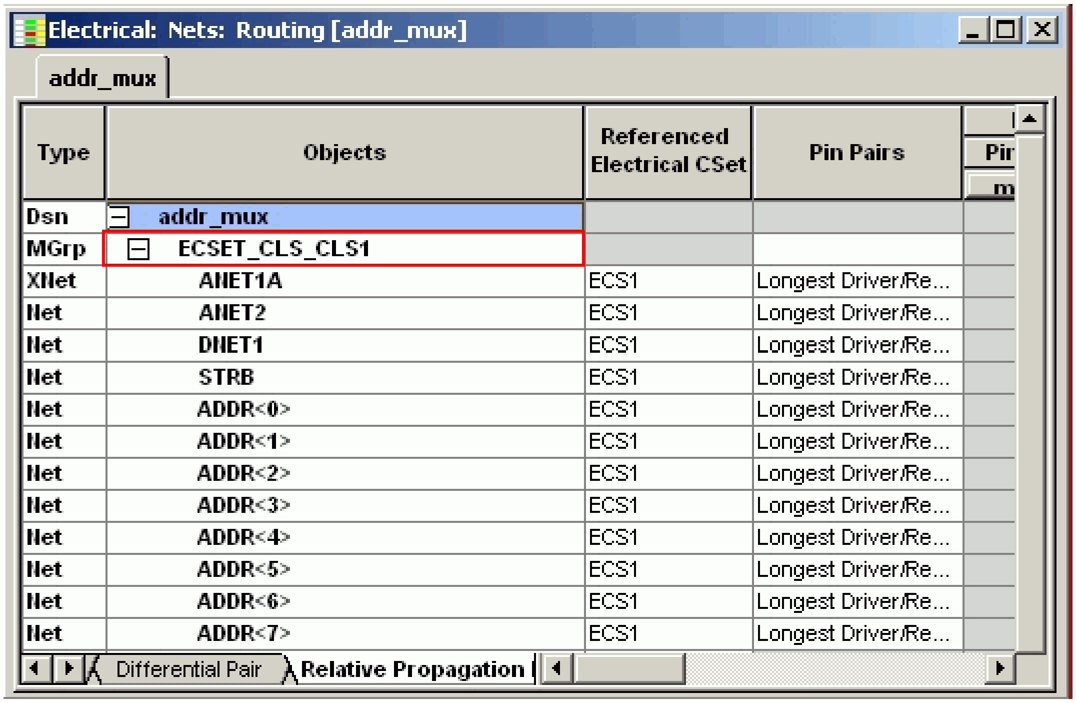
2. Assign the ECSet with class scope to a class.

Do the following to assign class scope to a class:

a. Create a class, `CLS1`, containing a net, an XNet, a differential pair, and a bus.

b. Right-click on the Referenced Electrical Cset column next to CLS1 and select ECS1.

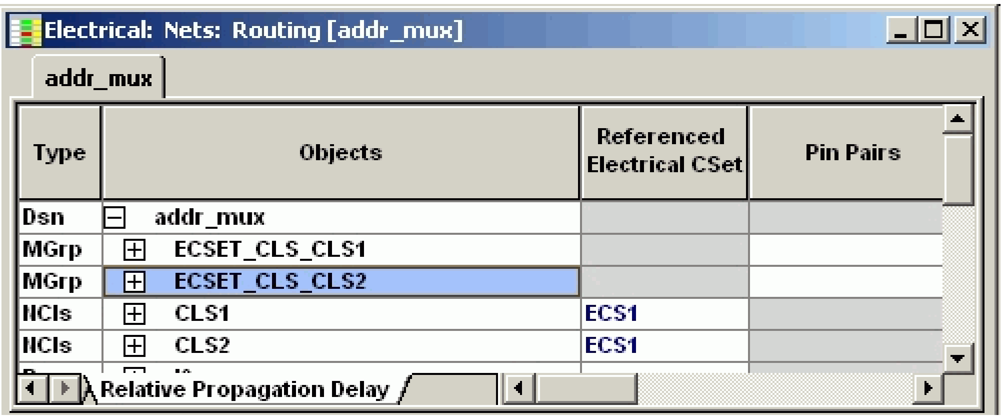
Note a new match group `ECSET_CLS_CLS1` is created. The naming convention followed is <ECMS>_<Class name>. The newly created match group contains all the objects including the bus as part of a single match group.



Type	Objects	Referenced Electrical CSet	Pin Pairs	Pir m
Dsn	addr_mux			
MGrp	ECSET_CLS_CLS1			
XNet	ANET1A	ECS1	Longest Driver/Re...	
Net	ANET2	ECS1	Longest Driver/Re...	
Net	DNET1	ECS1	Longest Driver/Re...	
Net	STRB	ECS1	Longest Driver/Re...	
Net	ADDR<0>	ECS1	Longest Driver/Re...	
Net	ADDR<1>	ECS1	Longest Driver/Re...	
Net	ADDR<2>	ECS1	Longest Driver/Re...	
Net	ADDR<3>	ECS1	Longest Driver/Re...	
Net	ADDR<4>	ECS1	Longest Driver/Re...	
Net	ADDR<5>	ECS1	Longest Driver/Re...	
Net	ADDR<6>	ECS1	Longest Driver/Re...	
Net	ADDR<7>	ECS1	Longest Driver/Re...	

c. Now assign the same ECSet to CLS2.

Note a new match group ECSET_CLS_CLS2 is created.


Electrical: Nets: Routing [addr_mux]
addr_mux
Relative Propagation Delay

Type	Objects	Referenced Electrical CSet	Pin Pairs
Dsn	⊟ addr_mux		
MGrp	⊞ ECSET_CLS_CLS1		
MGrp	⊞ ECSET_CLS_CLS2		
NCls	⊞ CLS1	ECS1	
NCls	⊞ CLS2	ECS1	

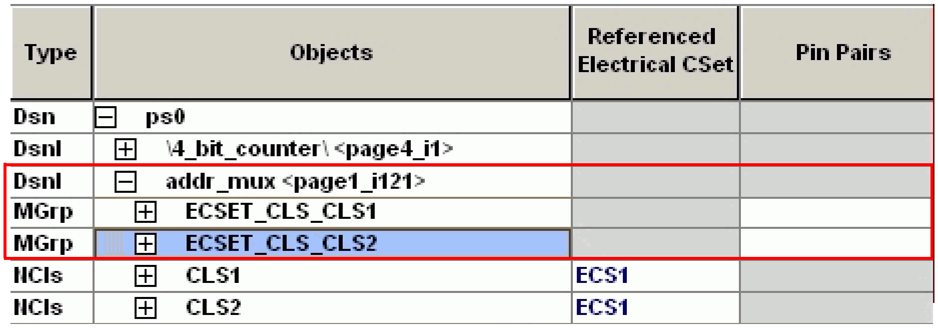
ECSets with Class Scope in the Context of a Top-Level Design

Class scope can also be used to ensure that unique match groups are created in the top-level of a hierarchical design especially when replicated blocks are used. Buses and class names are given unique names when lower-level blocks are merged into a top-level design.

Let's now see how these match groups appear in the context of a top-level design.

1. In Project Manager, click *Setup*.
2. Change the root design by specifying the name of the parent (top-level) design in the Design Name field to ps0.
3. Click *OK*.
4. Launch Design Entry HDL.
5. Choose *File – Export Physical*.
6. Click *OK*.
7. Click *No* when prompted to view the log file.
8. Launch Constraint Manager.
9. Select *Routing—Relative Propagation Delay* worksheet in the *Net* folder.

Note that the match groups `ECSET_CLS_CLS1`and `ECSET_CLS_CLS2` created in the hierarchical block `addr_mux` appear under `addr_mux`.



Type	Objects	Referenced Electrical CSet	Pin Pairs
Dsn	⊟ ps0		
Dsnl	⊞ \4_bit_counter\ <page4_i1>		
Dsnl	⊟ addr_mux <page1_i121>		
MGrp	⊞ ECSET_CLS_CLS1		
MGrp	⊞ ECSET_CLS_CLS2		
NCls	⊞ CLS1	ECS1	
NCls	⊞ CLS2	ECS1	

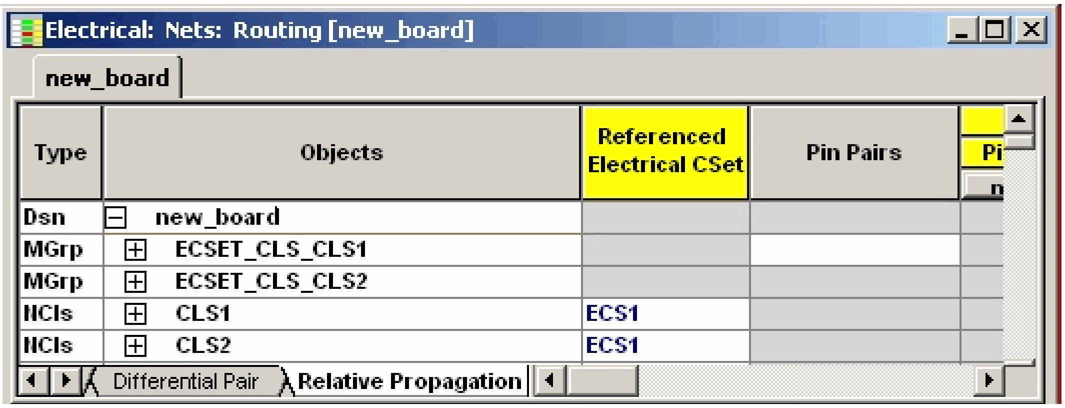
ECSets with Class Scope in Constraint Manager Connected to Allegro PCB Editor

You will now see how match groups created in a lower-level group appear in Constraint Manager connected to Allegro PCB Editor.

1. In Project Manager, click *Layout*.
2. In Allegro PCB Editor, choose *Setup – Constraints – Electrical*.

   Constraint Manager connected to Allegro PCB Editor appears.
3. Click *Routing–Relative Propagation Delay* in the *Net* folder.

   Note that the match groups created in the lower-level block appear under the parent board.



Electrical: Nets: Routing [new_board]

new_board

Type	Objects	Referenced Electrical CSet	Pin Pairs
Dsn	⊟ new_board		
MGrp	⊞ ECSET_CLS_CLS1		
MGrp	⊞ ECSET_CLS_CLS2		
NCls	⊞ CLS1	ECS1	
NCls	⊞ CLS2	ECS1	

Note: For more information on scopes, refer to the *Working with Constraint Objects* chapter of Allegro Constraint Manager User Guide.

Buses

A *bus* constraint object represents a named collection of XNets or Nets.

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.

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.

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. The constraint system, 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. XNet creation is based on the presence of the SIGNAL_MODEL property on the discrete components.

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.

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:

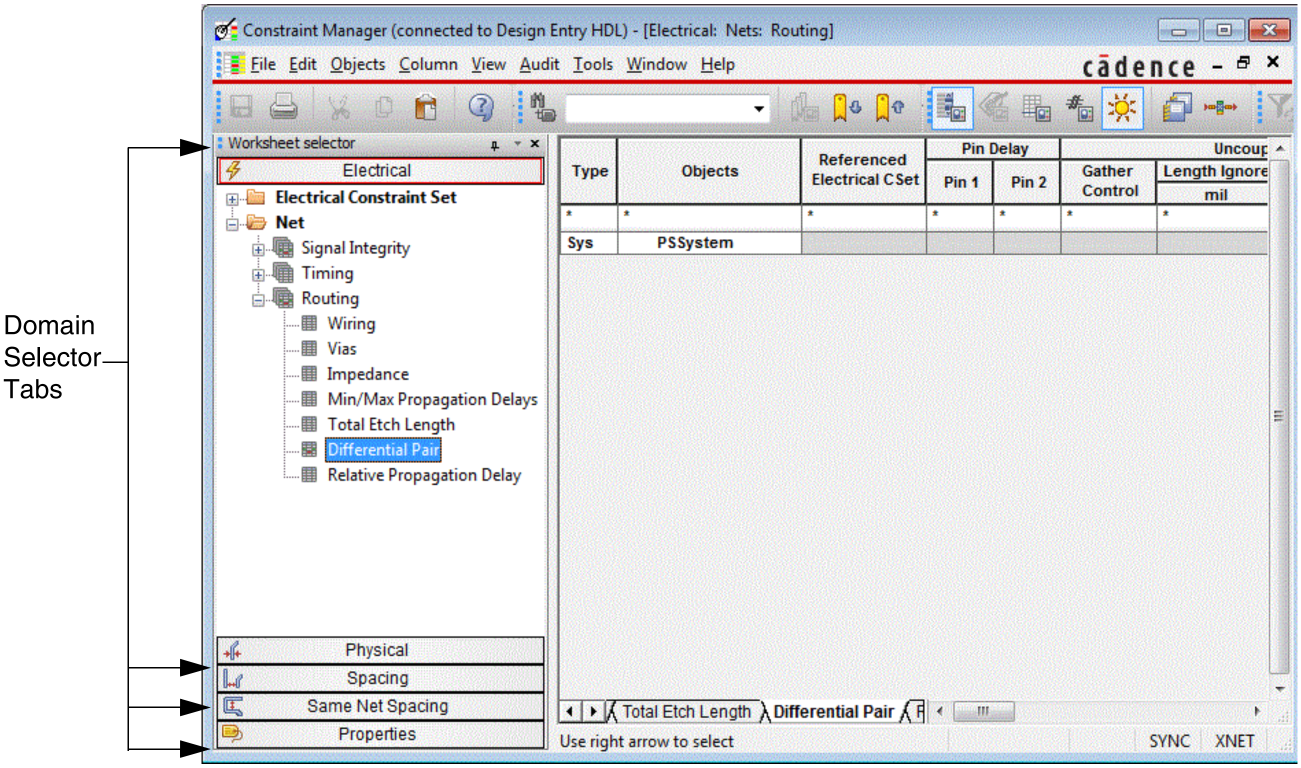
- longest pin pair
- longest driver-receiver pair
- all driver-receiver pairs

Constraint Manager User Interface

Constraint Manager user interface provides you with spreadsheet-based interface that provides a bird's eye view of constraint information in a domain at-a-glance. As shown in the figure: Constraint Manager User Interface, Constraint Manager provides separate tabs for specifying constraints in the *Electrical*, *Physical*, and *Spacing* domains. Each domain has workbooks with property sheets that lists the constraints that can be assigned to the design object. Constraint Manager provides a spreadsheet-based view of properties for all the components, nets and pins in the design. Rows in the spreadsheet view represent objects (components, nets or pins) and columns represent properties.

To know more about workbooks, worksheet, and constraint objects, see the section on Constraint Manager controls in *Allegro Constraint Manager User Guide*.

The Properties Tab provides you with the spreadsheet based interface for capturing general properties of nets in the design.



Type	Objects	Referenced Electrical CSet	Pin Delay		Uncoup	
			Pin 1	Pin 2	Gather Control	Length Ignore mil
*	*	*	*	*	*	*
Sys	PSSystem					

Figure 1-2 Constraint Manager User Interface

In any worksheet, the first column of lists the object type. For example, DPr in the Type column indicates that the object in that row is a differential pair.

In case of large designs, you can use Constraint Manager controls, to selectively display only the objects in a worksheet. For more information on viewing selective objects in Constraint Manager user interface, see the section *Viewing Worksheet Cells and Objects* in *Allegro Constraint Manager User Guide*.

Important

The worksheets displayed in a workbook depend on the tool from which Constraint Manager is invoked. For example, when invoked from Allegro PCB Editor, worksheets that have constraints and properties relevant to the manufacturing process are displayed. These are not displayed when you invoke Constraint Manger from System Connectivity Manager or Design Entry HDL.

Electrical Tab

This tab displays worksheets corresponding to the ECSets and SI/Timing/Routing constraints on nets. Use this domain to capture electrical constraints on nets.

Important

Electrical constraints can only be captured in Constraint Manager invoked from logic design tool.

Physical Tab

When Constraint Manager is invoked from a logic design tool, this tab lists all the net physical classes in the design along with the member nets of each class. You can add new net classes and can also modify the membership of these classes. This information is passed to the Allegro PCB Editor in the front-to-back flow. However, if you have invoked Constraint Manager from a logic design capture tool, you cannot view or apply the constraints to the net classes. This can only be done if Constraint Manager is invoked from Allegro PCB Editor.

Note: If you open a design created in an earlier version of the design tool, the values assigned in the Physical column of the General Properties worksheet are converted to net physical classes. The nets, on which these physical constraints were assigned, are listed as member nets of the net physical classes.

When invoked from Allegro PCB Editor, worksheets for PCSets and Region net classes will also be displayed. These are not supported in the logic design and are therefore, not listed in Constraint Manager invoked from the logic design tools.

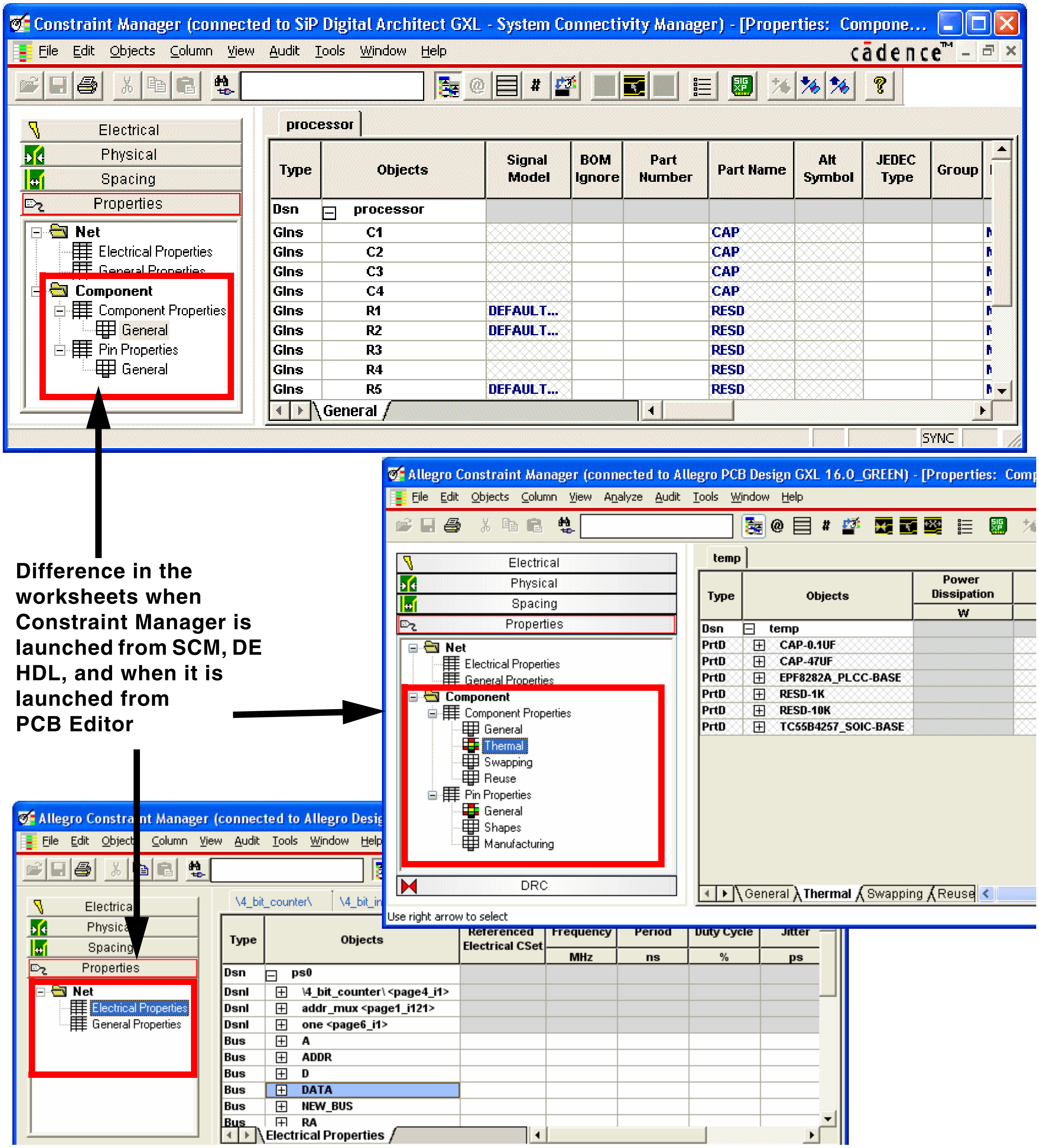
Spacing Tab

When CM is invoked from an logic design tool, this tab lists all the net spacing classes in the design along with the member nets of each class. You can add new net classes and can also modify the membership of these classes. This information is passed to the board in the front-to-back flow. However, if you have invoked Constraint Manager from a logic design capture tool, you cannot apply constraints to the net spacing classes. This can only be done if Constraint Manager is invoked from Allegro PCB Editor.

Note: If you open a design created in an earlier version of the design tool, the values assigned in the *Spacing* column of the *General Properties* worksheet are converted to net physical classes. The nets, on which these physical constraints were assigned, are listed as member nets of the net physical classes.

Properties Tab

When Constraint Manager is invoked from logic design capture tools, this tab displays the Electrical and General properties that can be assigned to the nets using Constraint Manger.



Constraint Manager (connected to SiP Digital Architect GXL - System Connectivity Manager) - [Properties: Compone...

Type	Objects	Signal Model	BOM Ignore	Part Number	Part Name	Alt Symbol	JEDEC Type	Group
Dsn	processor							
GIns	C1				CAP			
GIns	C2				CAP			
GIns	C3				CAP			
GIns	C4				CAP			
GIns	R1	DEFAULT...			RESD			
GIns	R2	DEFAULT...			RESD			
GIns	R3				RESD			
GIns	R4				RESD			
GIns	R5	DEFAULT...			RESD			

Allegro Constraint Manager (connected to Allegro PCB Design GXL 16.0_GREEN) - [Properties: Comp

Type	Objects	Power Dissipation
		W
Dsn	temp	
PrtD	CAP-0.1UF	
PrtD	CAP-47UF	
PrtD	EPF8282A_PLCC-BASE	
PrtD	RESD-1K	
PrtD	RESD-10K	
PrtD	TC55B4257_SOIC-BASE	

Use right arrow to select

Allegro Constraint Manager (connected to Allegro Desig

Type	Objects	Referenced Electrical CSet	Frequency	Period	Duty Cycle	Jitter
			MHz	ns	%	ps
Dsn	ps0					
DsnI	\4_bit_counter\<page4_i1>					
DsnI	addr_mux <page1_i121>					
DsnI	one <page6_i1>					
Bus	A					
Bus	ADDR					
Bus	D					
Bus	DATA					
Bus	NEW_BUS					
Bus	RA					

Difference in the worksheets when Constraint Manager is launched from SCM, DE HDL, and when it is launched from PCB Editor

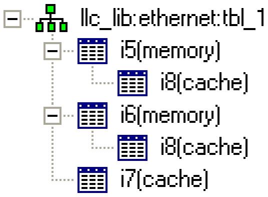
Constraint Manager Views

Constraint Manager provides a spreadsheet-based view of properties for all the components, nets and pins in the design. Rows in the spreadsheet view represent objects (components, nets or pins) and columns represent properties. There are two ways in which you can view information in Constraint Manager.

- Using Single View
- Using Tabbed View

Using Single View

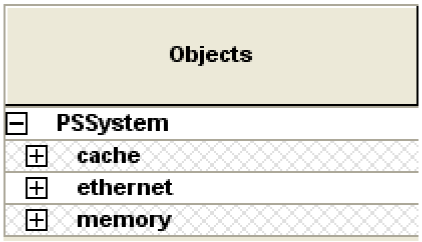
If you have a hierarchical design with the following structure in Design Entry HDL:



```
llc_lib:ethernet:tbl_1
    i5(memory)
        i8(cache)
    i6(memory)
        i8(cache)
    i7(cache)
```

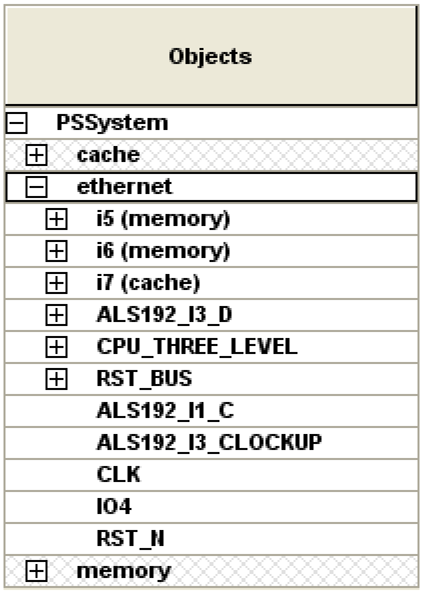
Hierarchical Design

By default, Constraint Manager displays the root design and each block in the hierarchical design as design objects under a system object named `PSSystem` as shown below:



Objects
⊟ PSSystem
⊞ cache
⊞ ethernet
⊞ memory

Click ⊞ next to the root design or a block to view the objects under the root design or a block. For example, if you click ⊞ next to the root design `ethernet`, the hierarchical structure of objects under the root design is displayed as shown below.

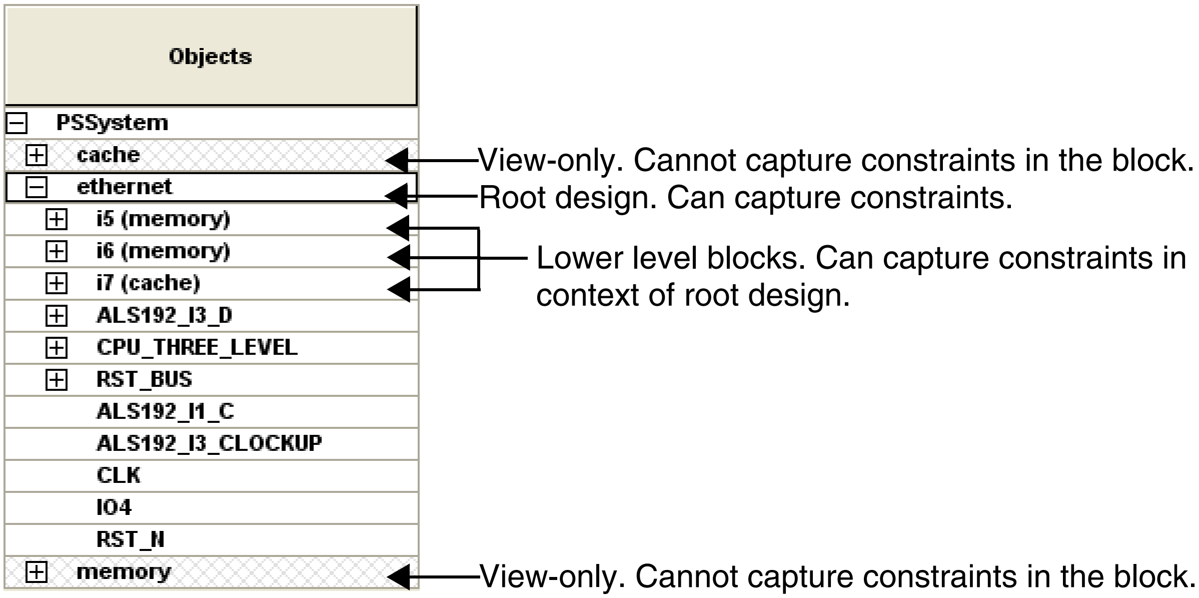


Objects

⊟ PSSystem
- ⊞ cache
- ⊟ ethernet
  - ⊞ i5 (memory)
  - ⊞ i6 (memory)
  - ⊞ i7 (cache)
  - ⊞ ALS192_I3_D
  - ⊞ CPU_THREE_LEVEL
  - ⊞ RST_BUS
  - ALS192_I1_C
  - ALS192_I3_CLOCKUP
  - CLK
  - IO4
  - RST_N
- ⊞ memory

The figure above displays the blocks and nets under the root design named `ethernet`.

Note: PSSystem is not visible if you set Constraint Manager options to display information is tabbed views



Objects

⊟ PSSystem
- ⊞ cache
- ⊟ ethernet
  - ⊞ i5 (memory)
  - ⊞ i6 (memory)
  - ⊞ i7 (cache)
  - ⊞ ALS192_I3_D
  - ⊞ CPU_THREE_LEVEL
  - ⊞ RST_BUS
  - ALS192_I1_C
  - ALS192_I3_CLOCKUP
  - CLK
  - IO4
  - RST_N
- ⊞ memory

View-only. Cannot capture constraints in the block.

Root design. Can capture constraints.

Lower level blocks. Can capture constraints in context of root design.

View-only. Cannot capture constraints in the block.

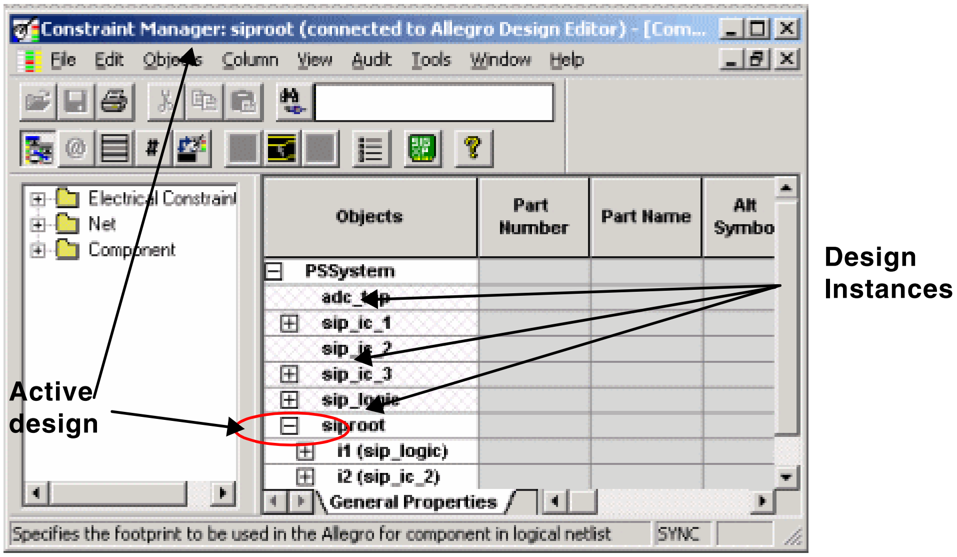
In the above figure, the blocks named `cache` and `memory` are grayed out because the blocks are not set as the root design for the project. You can view the constraints in blocks but cannot make any changes. If you want to assign constraints in the `cache` or `memory` block, you must set the block as the root design for the project.

In the above figure, the design named `ethernet` is not grayed out because it is the root design for the project. Constraint Manager lets you assign constraints only on the root design for the project. You can also assign constraints in the blocks under the root design. However, the constraints you assign in the blocks under the root design are applied in *context of the root design*. For example, if you assign constraints in the lower-level blocks named `memory` and `cache` under the `ethernet` design, the constraints are stored in the property file of the root design `ethernet`. In other words, the constraints will be visible in the `memory` and `cache` blocks only if you set the `ethernet` design as the root design in System Connectivity Manager and view the constraints on the `memory` and `cache` blocks. If you set the `memory` or `cache` block as the root design in System Connectivity Manager and open Constraint Manager, the constraints you added on the on the `memory` and `cache` blocks when the `ethernet` design was set as the root design are not displayed in the `memory` and `cache` blocks. This is because the constraints are applied in context of the root design `ethernet`.

Using Tabbed View

By default, the properties of all the design components are displayed in a single view. For large designs, this approach is not very user friendly as lots of information is displayed on the tab as shown in the figure: Information for all design blocks displayed in the same view. With all the data related to the lower blocks being displayed in the same worksheet, there is not much space available for the root design data. To overcome this problem, Constraint

Manager provides you with an option of displaying data for different design blocks in different tabs, as shown in the figure: Information displayed in tabbed view..



Objects	Part Number	Part Name	Alt Symbo
PSSystem			
adc_top			
sip_ic_1			
sip_ic_2			
sip_ic_3			
sip_logic			
siproot			
i1 (sip_logic)			
i2 (sip_ic_2)			

Specifies the footprint to be used in the Allegro for component in logical netlist

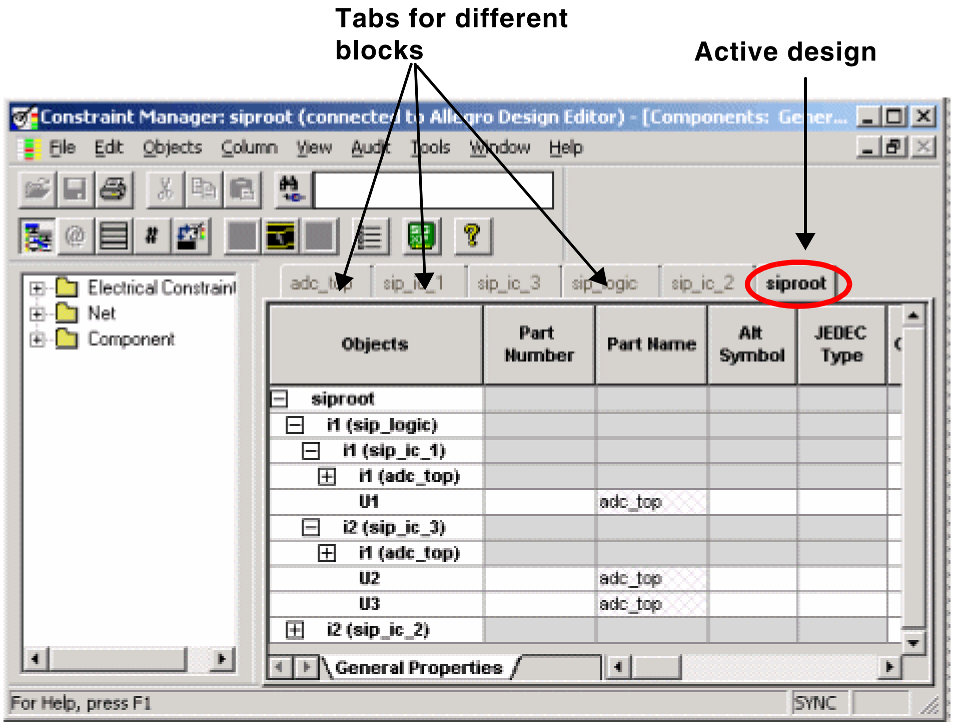
Figure 1-3 Information for all design blocks displayed in the same view.

Displaying Constraints in the Tabbed View

To display the constraint information in the tabbed view, perform the following steps in Constraint Manager:

1. Choose *View – Options*.
2. Select the *Use Tabbed View* check box.
3. Click *OK*.

Tabs are displayed in Constraint Manager as shown in the figure given below. The active tab is the tab corresponding to the root design.



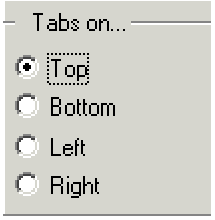
Objects	Part Number	Part Name	Alt Symbol	JEDEC Type
siproot				
i1 (sip_logic)				
i1 (sip_ic_1)				
i1 (adc_top)				
U1		adc_top		
i2 (sip_ic_3)				
i1 (adc_top)				
U2		adc_top		
U3		adc_top		
i2 (sip_ic_2)				

Figure 1-4 Information displayed in tabbed view.

Important

Number of tabs in Constraint Manager will be same as number of blocks in the logical design that can be set as root design.

Selecting the *Use Tabbed View* check box, enables the options used for specifying the position of tabs with respect to workbooks.



Tabs on...

● Top
○ Bottom
○ Left
○ Right

By default, tabs are displayed at the top of the workbook as shown in the figure: Information displayed in tabbed view. If required, select the appropriate option to display the tabs at any other location.

Synchronizing Tab Selection

Constraint Manager allows you to open multiple workbooks to view different constraints. To ensure that same design is active in all open workbooks, do the following:

1. Choose *View – Options*.
2. Select the *Synchronize Tabs* check box.
3. Click *OK*.

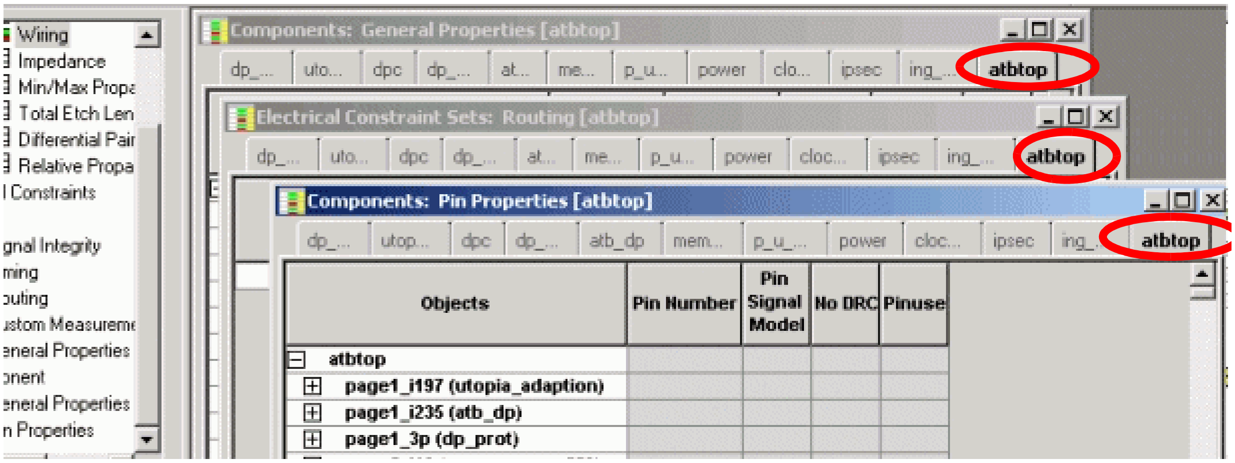

Wiring
Impedance
Min/Max Propa
Total Etch Len
Differential Pair
Relative Propa
l Constraints
gnal Integrity
ming
outing
ustom Measureme
eneral Properties
onent
eneral Properties
n Properties
Components: General Properties [atbtop]
dp_... uto... dpc dp_... at... me... p_u... power clo... ipsec ing_... atbtop
Electrical Constraint Sets: Routing [atbtop]
dp_... uto... dpc dp_... at... me... p_u... power cloc... ipsec ing_... atbtop
Components: Pin Properties [atbtop]
dp_... utop... dpc dp_... atb_dp mem... p_u... power cloc... ipsec ing_... atbtop
Objects
Pin Number
Pin Signal Model
No DRC
Pinuse
atbtop
page1_i197 (utopia_adaption)
page1_i235 (atb_dp)
page1_3p (dp_prot)

Figure 1-5 Synchronized Tabs

If the *Synchronize Tabs* check box is not selected, different workbooks show data for different blocks in the design,

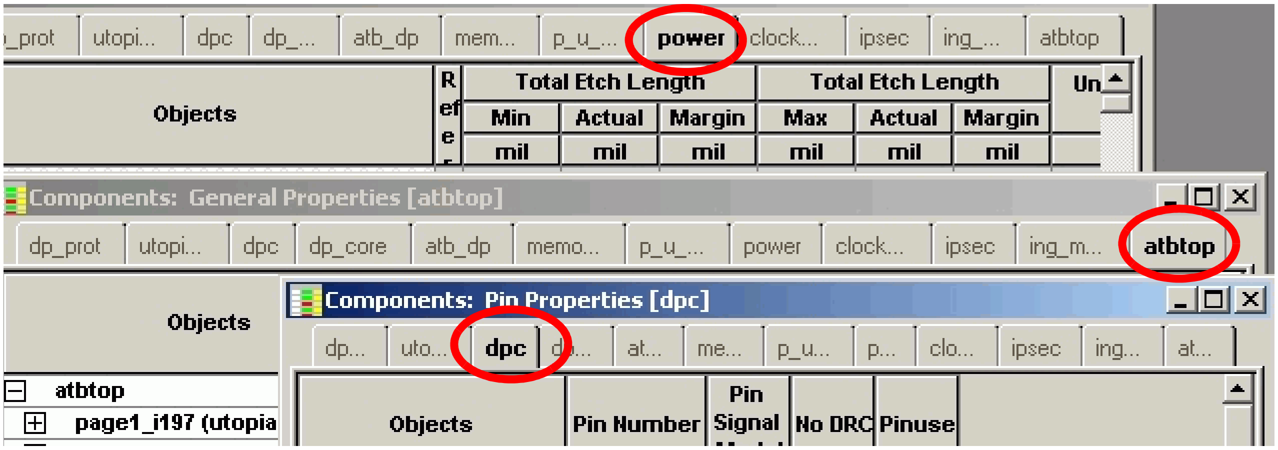

prot utopi... dpc dp... atb_dp mem... p_u_... power clock... ipsec ing_... atbtop
Objects
Ref
Total Etch Length
Min Actual Margin
mil mil mil
Total Etch Length
Max Actual Margin
mil mil mil
Un
Components: General Properties [atbtop]
dp_prot utopi... dpc dp_core atb_dp memo... p_u_... power clock... ipsec ing_m... atbtop
Objects
atbtop
page1_i197 (utopia
Components: Pin Properties [dpc]
dp... uto... dpc d... at... me... p_u... p... clo... ipsec ing... at...
Objects
Pin Number
Pin Signal
No DRC
Pinuse

Figure 1-6 Non-Synchronized Tabs

Note: If required, you can use the context-sensitive menu to modify the synchronization settings for the current session. To do this, right-click and clear the *Stay Synchronized* command.

Design Capture with Constraint Manager

The figure Constraint Manager in Design Capture Flow depicts the flow of information in the front to back flow.

Front-to-Back Flow

The part of the diagram that covers the data flow from the logic design capture tool, to physical layout tool — Allegro PCB Editor or Allegro PCB SI — is referred to as front -to-back flow. For sharing data with the physical layout tool, design capture tools use the *Export Physical* command. Using Export Physical command generates a set of package files that contain information about electrical constraints and the netlist. The files that are created when you the use the Export Physical command are:

- `pstchip.dat`
- `pstrxprt.dat`
- `pstxnet.dat`
- `pstcmdb.dat`

In this flow, Constraint Manager stores information about constraints in a file named `<`*`root_design_name`*`>.dcfx (<`*`root_design_name`*`>.dcf)`, which is created in the `sch_1` folder in a ZIP format. The `.dcf` file is a binary file, which, when extracted, creates multiple XML and text files that contain information about constraints and properties that are applied to objects in a design. For more information about viewing a `.dcfx` (`.dcf`) file, refer to the *Viewing a DCF File* section of *Allegro Design Entry HDL Reference Guide*.

To know more about the front-to-back flow, see *Allegro Front-to-Back User Guide*.

Back-to-Front Flow

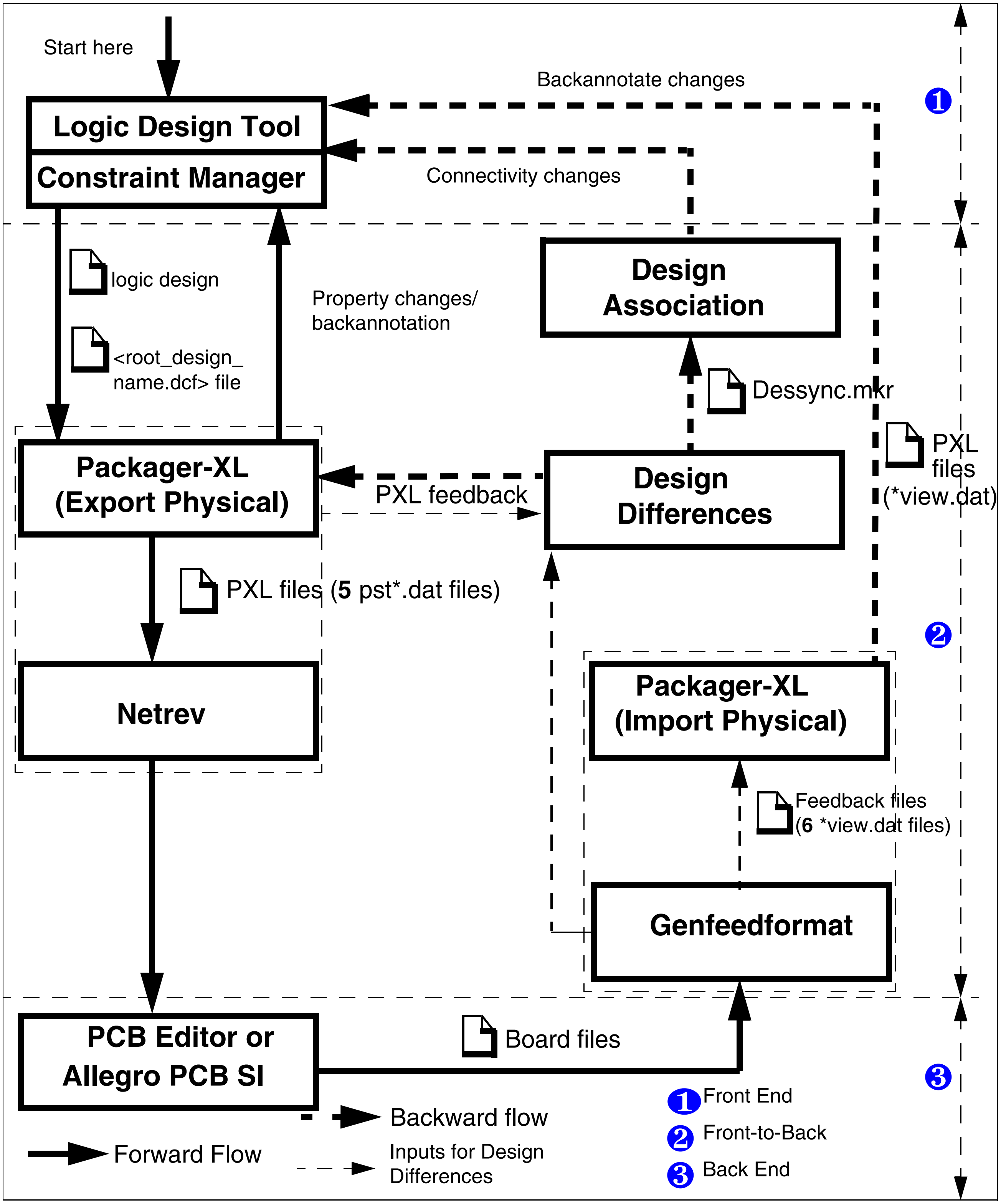
If you have made changes to the design in physical layout tool — Allegro PCB Editor or Allegro PCB SI — the logical design needs to be updated with the changes, to ensure that the designs are synchronized. This flow of constraints and other data from Allegro PCB Editor to logic design tools is referred to as back-to-front flow. To update the logical design with the modifications in the physical layout of the design, use *Import Physical* command from the logic design tools.

The files that are read by the logic design tools, while importing changes communicate component, part, function, pin, and electrical constraint information. The six files that are used to share data in the back-to-front flow are:

- `compview.dat`
- `funcview.dat`
- `netview.dat`
- `pinview.dat`

For more information on the files used in the back-to-front flow, see the section *Back to Front Constraint Flow* in *Allegro Constraint Manager User Guide*.

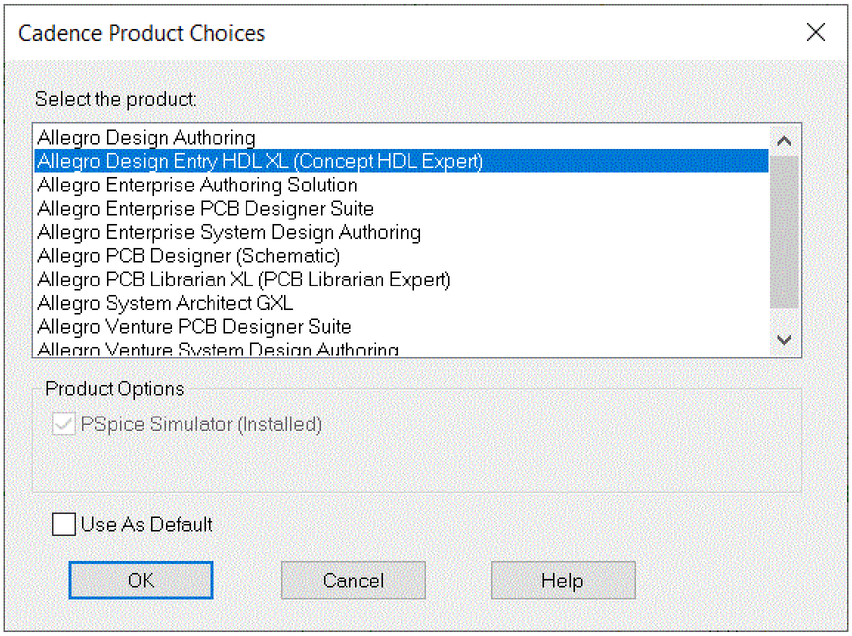
Figure 1-7 Constraint Manager in Design Capture Flow


Start here
Backannotate changes
Logic Design Tool
Constraint Manager
Connectivity changes
❶
logic design
<root_design_
name.dcf> file
Property changes/
backannotation
Design
Association
Dessync.mkr
PXL
files
(*view.dat)
Packager-XL
(Export Physical)
PXL feedback
Design
Differences
PXL files (5 pst*.dat files)
❷
Netrev
Packager-XL
(Import Physical)
Feedback files
(6 *view.dat files)
Genfeedformat
PCB Editor or
Allegro PCB SI
Board files
❸
Backward flow
Forward Flow
Inputs for Design
Differences
❶ Front End
❷ Front-to-Back
❸ Back End

Starting Constraint Manager

To start Constraint Manager from Design Entry HDL:

➡ Choose *Tools – Constraints – Edit.*



Cadence Product Choices

Select the product:

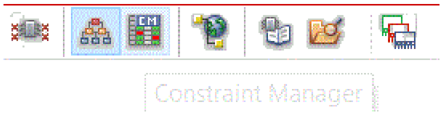
- Allegro Design Authoring
- Allegro Design Entry HDL XL (Concept HDL Expert)
- Allegro Enterprise Authoring Solution
- Allegro Enterprise PCB Designer Suite
- Allegro Enterprise System Design Authoring
- Allegro PCB Designer (Schematic)
- Allegro PCB Librarian XL (PCB Librarian Expert)
- Allegro System Architect GXL
- Allegro Venture PCB Designer Suite
- Allegro Venture System Design Authoring

Product Options

☑ PSpice Simulator (Installed)

□ Use As Default

You can also launch Constraint Manager from the toolbar.



2

Creating XNets and Differential Pairs

Design Entry HDL provides support for creating XNets and model-defined differential pairs. You can create XNets and Differential Pairs in Design Entry HDL by assigning signal models to components.

This chapter contains the following sections:

Creating XNets by Assigning Signal Models

Constraint Manager connected to Design Entry HDL reads the `SIGNAL_MODEL` property on discretes and creates XNets. The `SIGNAL_MODEL` property, attached to a component, is the name of the Packaged Device Model which defines the electrical, I/O models, and package parasitics used by SigNoise to characterize devices for simulations. The `SIGNAL_MODEL` property can be present in a schematic, chips.prt file, or the ptf. With support for XNets in Design Entry HDL, XNets are displayed for discrete parts in Constraint Manager connected to Design Entry HDL.

To help you assign the `SIGNAL_MODEL` property to components, Design Entry HDL includes a comprehensive UI support. The Model Assignment window provides a convenient way to assign models for IC devices and auto generate these models for discrete components. You can assign signal models to components and pins using the new Model Assignment window. You can assign signal models to multiple components, simultaneously.

Important

> You cannot assign a signal model to a component instance by attaching the `SIGNAL_MODEL` property using the Attributes dialog form.

For more information on signal models, see *Allegro SI Device Modeling Language User Guide*.

Important

> Before creating a new XNet in legacy designs, ensure that the Signal Integrity analysis features are enabled.

This section covers the following:

- Use Model on page 48
- Model Assignment User Interface on page 49
- Assigning Signal Models on page 52

Use Model

You use the Model Assignment window to assign signal models to components in a design. Using the Model Assignment window does the following:

- Facilitates automatic generation of signal models for 2-pin discrete components.

- Allows you to manually assign existing signal models to components and pins.
- Allows you to set the library path to the directory containing the signal models.
- Allows you to view signal models using the Model Integrity tool.
- Allows you to perform two-way cross-probing between the Model Assignment UI and schematic.

The different types of signal models you can assign include:

- IbisDevice Models on ICs
- PackageModel on connectors
- EspiceDevice Models on discrete elements

Opening the Model Assignment Window

You can open the Model Assignment window by choosing *Tools – Model Assignment* menu command on the main Design Entry HDL window.

The Model Assignment window loads the design and displays a list of components along with details, such as instance name, block name, location, model source, and the signal model (if assigned).

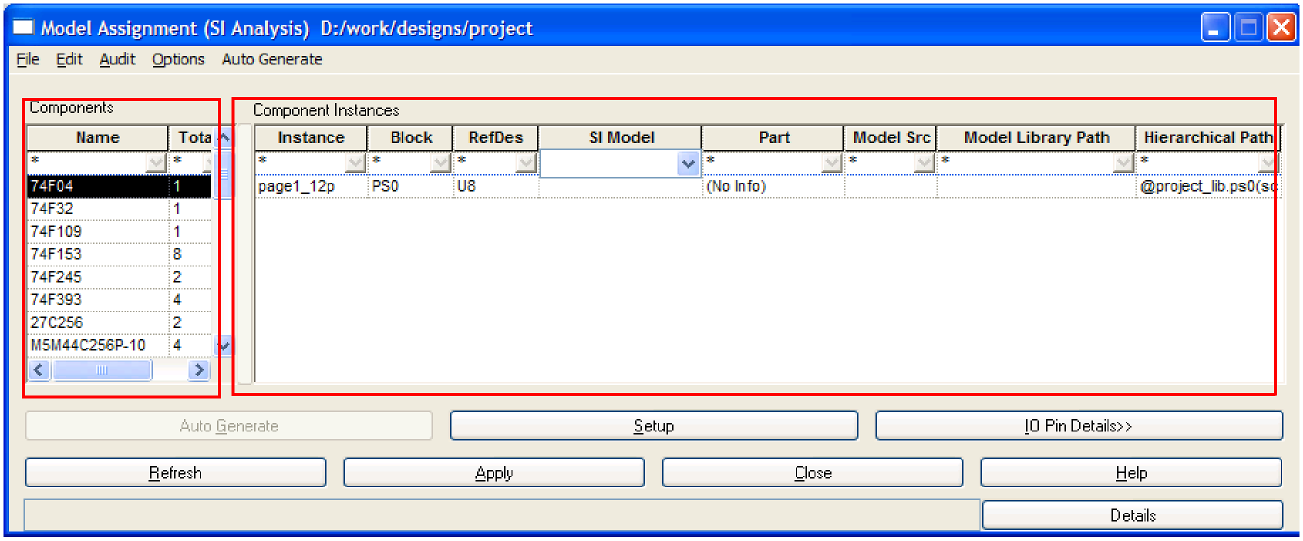
If you are not in the *retain existing XNets and diff-pairs* mode, on opening the Model Assignment window, you will be prompted to enable the Signal Integrity analysis features.

To enable the Signal Integrity analysis features, set the `retain_existing_xnets_and_diffpairs` directive to ‘OFF’.

Model Assignment User Interface

The Model Assignment user interface is a three-pane window. The first pane lists the names of the parts used in the design, the second pane lists details about the part selected in the first pane, and the third pane, which is hidden by default, lists the pin details of the part instance selected in the second pane.

Figure 2-1 Model Assignment



Components

Name	Tota
*	*
74F04	1
74F32	1
74F109	1
74F153	8
74F245	2
74F393	4
27C256	2
M5M44C256P-10	4

Component Instances

Instance	Block	RefDes	SI Model	Part	Model Src	Model Library Path	Hierarchical Path
*	*	*		*	*	*	*
page1_12p	PS0	U8		(No Info)			@project_lib.ps0(sc

The First Pane

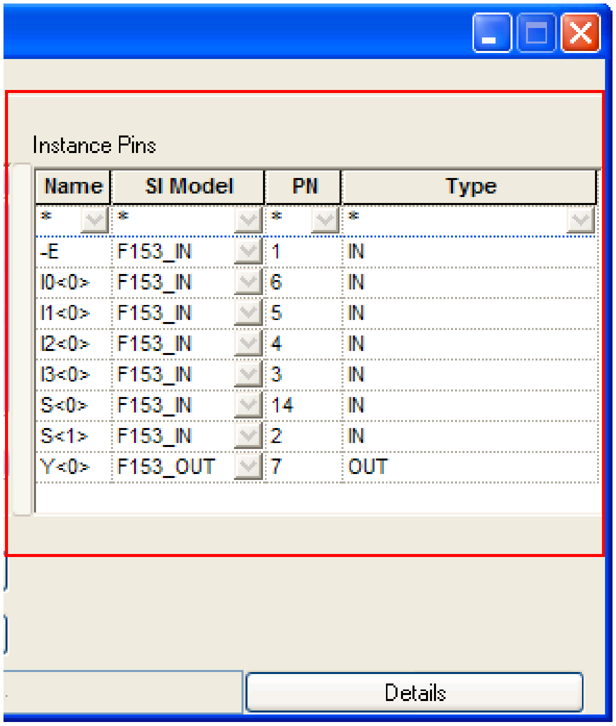
The Components list on the first pane displays the physical part names of all the electrical parts used in the design, the number of instances of each part, the total number of instances which have incorrect signal models assigned to them, and the number of instances that do not have a signal model assigned. The first component is selected and a detailed list of instances of the selected components is listed on the second pane.

The Second Pane

When you select a component in the first pane, a detailed list of information about all the instances of the selected component, is displayed on the second pane of the Model Assignment window in a grid format. This pane displays the following information about the component instance:

- Instance - Name of the component instance in the design
- Block name - Block name of the design
- Location - Reference designator of the component instance
- Signal model - Signal model if assigned
- Part - Key properties of the component instance from the physical part table
- Model Src - Source of the signal model, whether schematic or opf

Figure 2-2 Model Assignment - The Third pane



Instance Pins

Name	SI Model	PN	Type
*	*	*	*
-E	F153_IN	1	IN
I0<0>	F153_IN	6	IN
I1<0>	F153_IN	5	IN
I2<0>	F153_IN	4	IN
I3<0>	F153_IN	3	IN
S<0>	F153_IN	14	IN
S<1>	F153_IN	2	IN
Y<0>	F153_OUT	7	OUT

The Third Pane

The third pane which is hidden by default, allows you to add pin buffer models to the pins. This pane displays detailed information about the pins of the component instance selected in the second pane. The detailed information includes pin name, pin number, pin type, and signal model (if assigned). To display this pane, you need to click the *IO Pin Details* button on the window.

Status Bar

The status bar at the bottom of the window displays the status of the model assignment validation. If the validation routine fails, a message is displayed in the status bar stating that the signal model does not map the instance.

Example: You incorrectly assign a signal model, EightPin18V, of type IbisDevice to a part instance @diffpair_lib.new3(sch_1):page1_i1@diffpair_lib.als192(chips). When the validation routine runs, it will display the following message in the status bar:

Model EightPin18V does not map the instance
@diffpair_lib.new3(sch_1):page1_i1@diffpair_lib.als192(chips) properly.

Assigning Signal Models

The Model Assignment user interface provides you with functions to assign signal models to components and pin models to pins. You can highlight a specific instance on the schematic by selecting it in the grid in the second pane.

This section covers the following:

- Managing Device Model Libraries on page 52
- Creating an XNet by Assigning a Signal Model to a Discrete on page 57
- Defining a Model-Defined Diff-Pair by Assigning a Signal Model to an IC Device on page 59
- Other Tasks Done from the Model Assignment GUI on page 61

Managing Device Model Libraries

Model Assignment provides a user interface for managing device model libraries.You can perform the following library management functions:

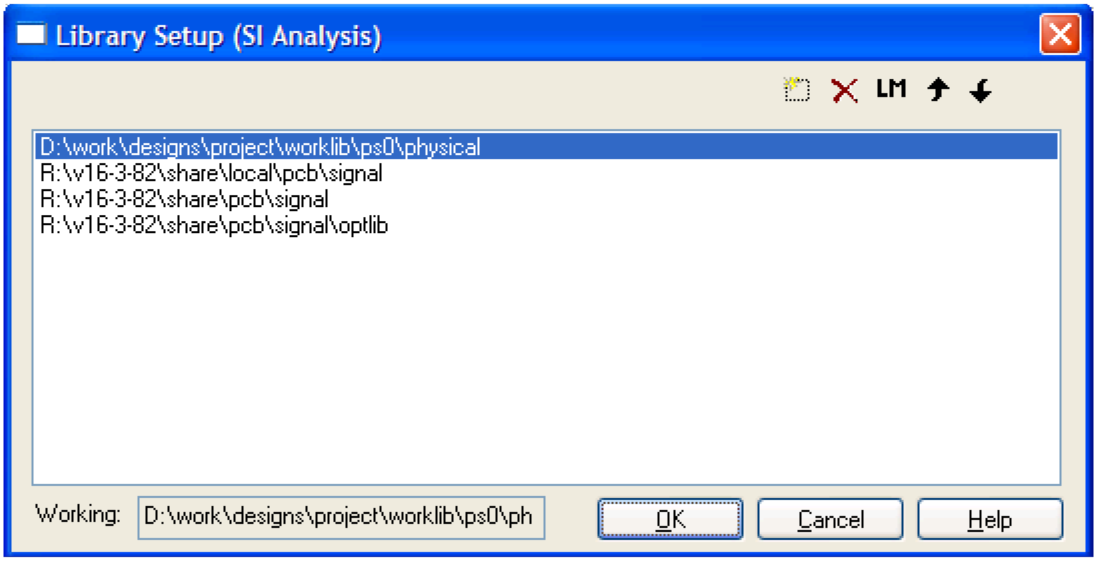
- Setting up path for the device model library on page 52
- Adding a new library on page 53
- Deleting a library on page 54
- Launching Library Management on page 54

Setting up path for the device model library

To modify or assign a different.`dml file` for a design, you need to set up the library path. To set up the library path, do the following:

1. Click the *Setup* button in the Model Assignment window.

This launches the Library Setup (SI Analysis) dialog box.



D:\work\designs\project\worklib\ps0\physical
R:\v16-3-82\share\local\pcb\signal
R:\v16-3-82\share\pcb\signal
R:\v16-3-82\share\pcb\signal\optlib

Working: D:\work\designs\project\worklib\ps0\ph

This dialog box display a list of available dml files. From this dialog box, you can:

- Add a new library
- Delete a Library
- Launch the Library Management GUI

Note: In addition to the `.dml` file there are index files (`.ndx`), which contain pointers to models in `.dml` libraries and allow the user to minimize the amount of libraries that must be added. For a complete list of models, refer to the following file in your Cadence installation directory.

```
share/pcb/signal/cds_partlib.ndx
```

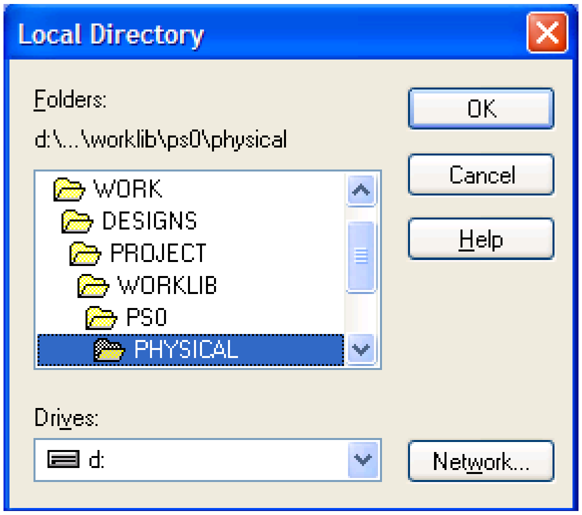
For more information on `.ndx` and `.dml` files, refer to *Allegro SI Device Modeling Language User Guide* and *Allegro PCB SI User Guide*.

Adding a new library

To add a new device model library:

1. Click the *Add a new library* icon ( ) in the Library Setup (SI Analysis) dialog box.

This launches a dialog box to add new libraries



2. Browse to the path of the folder containing the `.dml` file and click *OK*.

   The new `.dml` file is added to the list of files available for the design.

3. Select the newly added `.dml` file from the list and click *OK*.

   This option corresponds to the `SI_MODEL_PATH` directive. For more information, see CPM Directives for SI Model Setup on page 55.

For more information on device model libraries, refer to the *Model and Library Management* chapter of *Allegro PCB SI User Guide*.

Deleting a library

To delete a library:

1. Select the `.dml` file from the list of files available for the design.
2. Click the *Delete* icon (☒).

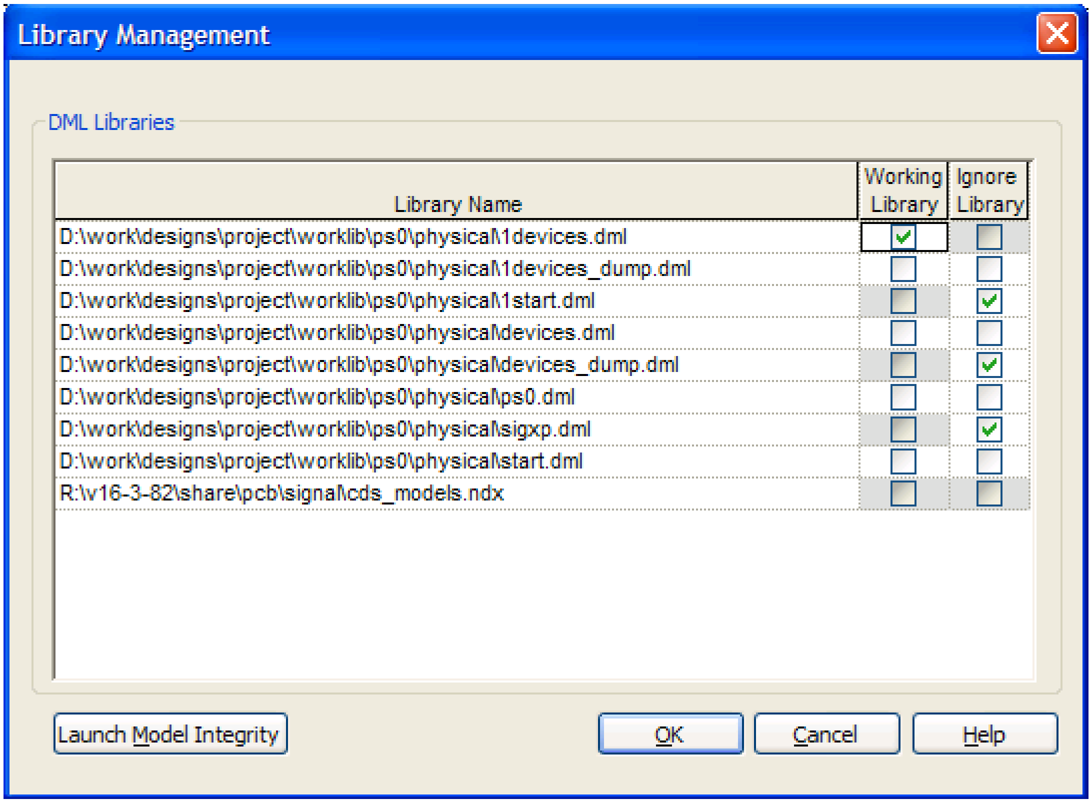
   The `.dml` file is deleted from the list.

Launching Library Management

The Library Management UI helps you in setting the working library for the design and ignoring libraries from the available libraries. You can also launch the Model Integrity UI to edit and manage signal model files.

1. Click the *Launch Library Management* icon (LM).

   The Library Management window is displayed with all the `.dml` files found in the path specified in the cpm file.



Library Name	Working Library	Ignore Library
D:\work\designs\project\worklib\ps0\physical\1devices.dml	☑	□
D:\work\designs\project\worklib\ps0\physical\1devices_dump.dml	□	□
D:\work\designs\project\worklib\ps0\physical\1start.dml	□	☑
D:\work\designs\project\worklib\ps0\physical\devices.dml	□	□
D:\work\designs\project\worklib\ps0\physical\devices_dump.dml	□	☑
D:\work\designs\project\worklib\ps0\physical\ps0.dml	□	□
D:\work\designs\project\worklib\ps0\physical\sigxp.dml	□	☑
D:\work\designs\project\worklib\ps0\physical\start.dml	□	□
R:\v16-3-82\share\pcb\signal\cds_models.ndx	□	□

The cpm file of the project contains a set of directives for SI setup. These directives are also written in $HOME/pcbenv/env file for layout designs.

Table 2-1 CPM Directives for SI Model Setup

CPM Directive	Description	Example
`SI_MODEL_PATH`	Contains the path of the directories in which `.dml` or `.ndx` files are stored. You cannot specify individual files in the model path. The '.' value directs the tool to pick all the `.dml` or `.ndx` files in the `physical` folder for the design.	`SI_MODEL_PATH '.' '$CDS_INST_DIR/share/local/pcb/signal' '$CDS_INST_DIR/share/pcb/signal' '$CDS_INST_DIR/share/pcb/signal/optlib'`

`SI_DML_WORKING_LIB`	DML library set as the working library. Autogenerated models are stored in the working library.	`SI_DML_WORKING_LIB 'start.dml'`

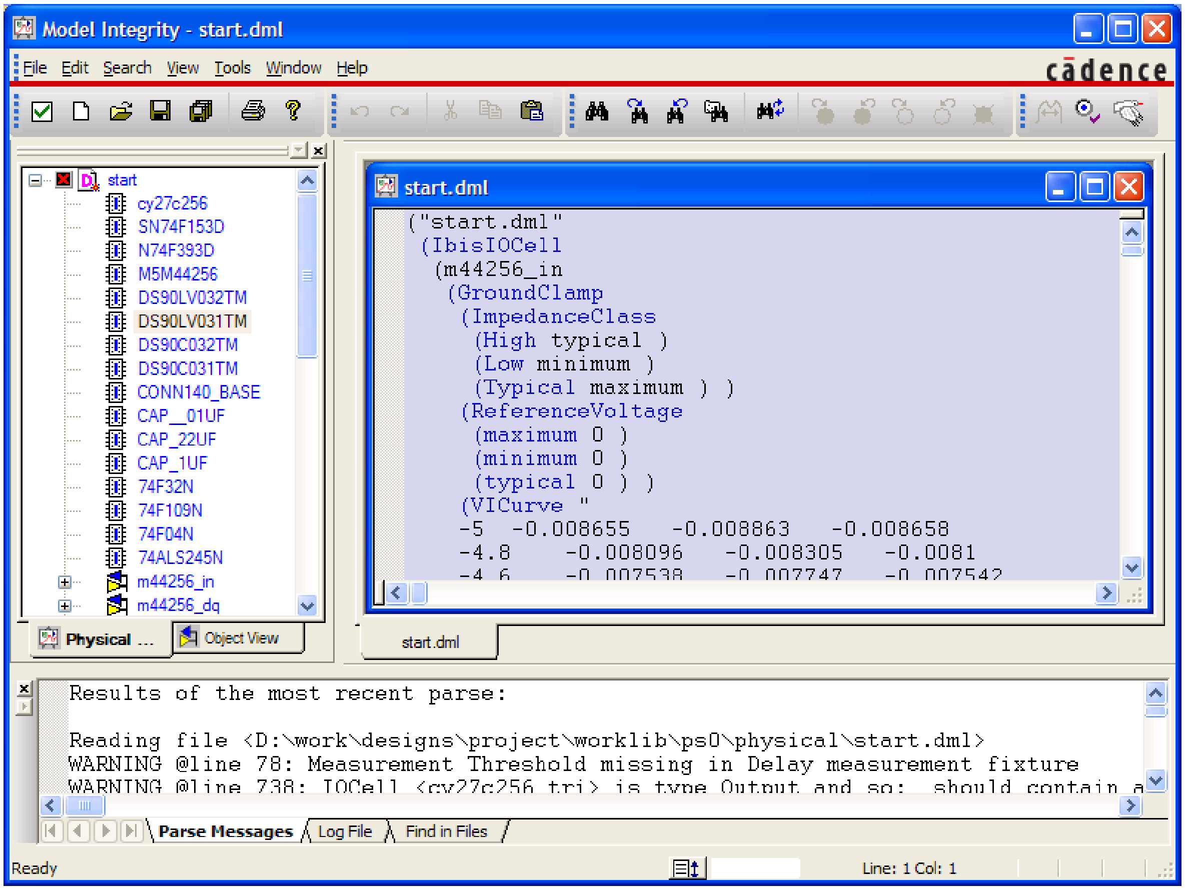
`SI_IGNORE_DML_LIBS`	DML libraries to be ignored while performing search	`'1devices.dml'`
		`'1devices_dump.dml'`
		`'1start.dml'`
		`'devices.dml'`
		`'devices_dump.dml'`
		`'sigxp.dml'`

Note: The following paths are automatically added to the `SI_MODEL_PATH` directive:

> `'$CDSROOT/share/local/pcb/signal'` — This is the Cadence-recommended location for storing standard libraries. When a new version of the software is installed, the contents of this directory are maintained.
>
> `'$CDSROOT/share/pcb/signal'` — This directory contains the default SI libraries that are shipped with any product that includes any SI tools.

1. Select a library to set it as a working library. This option corresponds to the `SI_DML_WORKING_LIB` directive as explained in CPM Directives for SI Model Setup on page 55.
2. Select the libraries to be ignored. This option corresponds to the `SI_IGNORE_DML_LIBS` directive as explained in CPM Directives for SI Model Setup on page 55.
3. Click Launch Model Integrity to display the Model Integrity UI.

Use the Model integrity editor to edit the selected model library.



```
("start.dml"
 (IbisIOCell
  (m44256_in
   (GroundClamp
    (ImpedanceClass
     (High typical )
     (Low minimum )
     (Typical maximum ) )
    (ReferenceVoltage
     (maximum 0 )
     (minimum 0 )
     (typical 0 ) )
    (VICurve "
    -5   -0.008655   -0.008863   -0.008658
    -4.8     -0.008096   -0.008305   -0.0081
    -4.6     -0.007538   -0.007747   -0.007542
```

```
Results of the most recent parse:

Reading file <D:\work\designs\project\worklib\ps0\physical\start.dml>
WARNING @line 78: Measurement Threshold missing in Delay measurement fixture
WARNING @line 738: IOCell <cy27c256 tri> is type Output and so:  should contain a
```

4. Close the Model Integrity window when you are done.

5. Click *OK* in the Library Management dialog box.

Creating an XNet by Assigning a Signal Model to a Discrete

Important

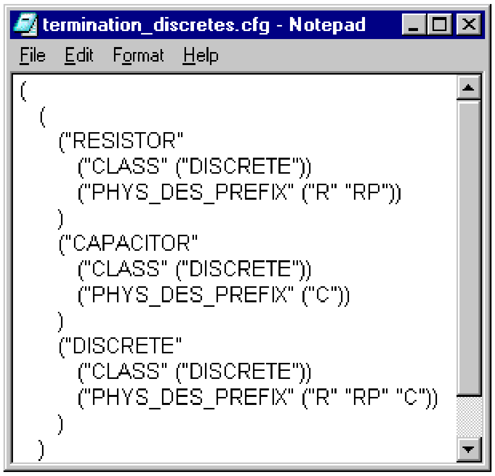
Before you proceed with XNet creation in Design Entry HDL, review Migrating a Design from pre-15.2 to a Higher Version on page 196 and Frequently Asked Questions About the Constraint Manager Flow on page 203.

You create an XNet by assigning a signal model to a discrete device (resistor, inductor or capacitor) separating segments of a net. Using the Model Assignment window, you can automatically generated signal models for discrete components in a design. To assign a signal model to a discrete component:

1. Select a discrete component from the Components list on the first pane, for example, select resistors.

   A list of all the instances of the selected discrete component is displayed on the second pane. Note that the *Auto Generate* button is activated. The activation of the *Auto Generate* button is controlled by a configuration file stored at the following location:

   ```
   <your_cadence_installation_directory>\share\cdssetup\<filename>.cfg
   ```

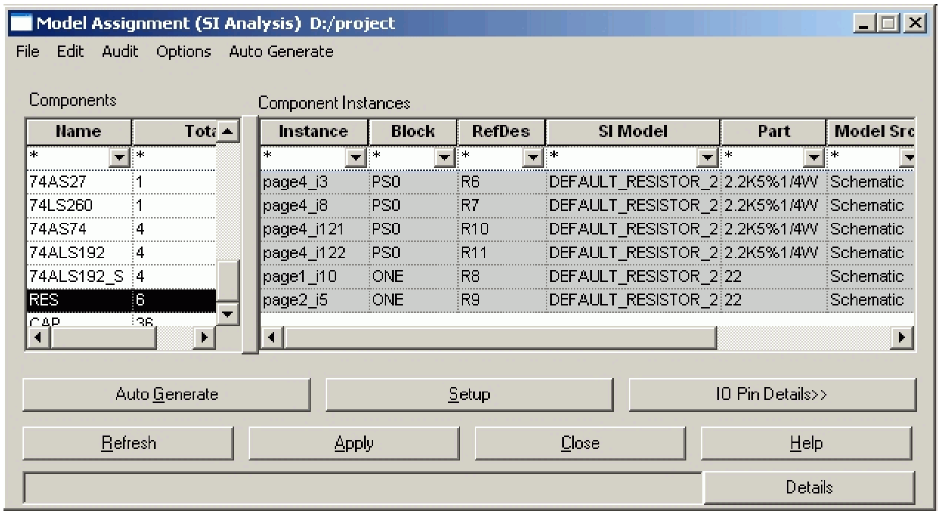


```
(
  (
    ("RESISTOR"
      ("CLASS" ("DISCRETE"))
      ("PHYS_DES_PREFIX" ("R" "RP"))
    )
    ("CAPACITOR"
      ("CLASS" ("DISCRETE"))
      ("PHYS_DES_PREFIX" ("C"))
    )
    ("DISCRETE"
      ("CLASS" ("DISCRETE"))
      ("PHYS_DES_PREFIX" ("R" "RP" "C"))
    )
  )
```

   The configuration file specifies what statements model assignment should look out for in the `chips.prt` file of a discrete component. If these qualifying statements are found in the `chips.prt` file, the *Auto Generate* button is enabled for the given component:

   ```
   CLASS = 'DISCRETE'
   PHYS_DES_PREFIX = 'R'
   ```

Note: Release 16.01 onwards, the Model Assignment feature provides support for autogeneration of signal models for discrete pack components. When models are generated, they are written to the dml file. However, Netrev still cannot autogenerate the models for discrete pack components. Therefore, to ensure that the models generated by model assignment are carried forward to the board, you must send the dml files along with the pst* files to the board designer.



Components

Name	Tota
*	*
74AS27	1
74LS260	1
74AS74	4
74ALS192	4
74ALS192_S	4
RES	6
CAP	36

Component Instances

Instance	Block	RefDes	SI Model	Part	Model Src
*	*	*	*	*	*
page4_i3	PS0	R6	DEFAULT_RESISTOR_2	2.2K5%1/4W	Schematic
page4_i8	PS0	R7	DEFAULT_RESISTOR_2	2.2K5%1/4W	Schematic
page4_i121	PS0	R10	DEFAULT_RESISTOR_2	2.2K5%1/4W	Schematic
page4_i122	PS0	R11	DEFAULT_RESISTOR_2	2.2K5%1/4W	Schematic
page1_i10	ONE	R8	DEFAULT_RESISTOR_2	22	Schematic
page2_i5	ONE	R9	DEFAULT_RESISTOR_2	22	Schematic

2. Select an instance and click the *Auto Generate* button.

   Alternatively, you can select a discrete device in the left pane and choose *Auto Generate – Selected* menu. This generates signal models for all the instances of the selected discrete device. To generate signal models for all the instances of all the two-pin discrete devices in the design, choose *Auto Generate – All*.

3. Click *Apply*.

   An appropriate signal model is assigned to the instance.

4. Click *Close*.

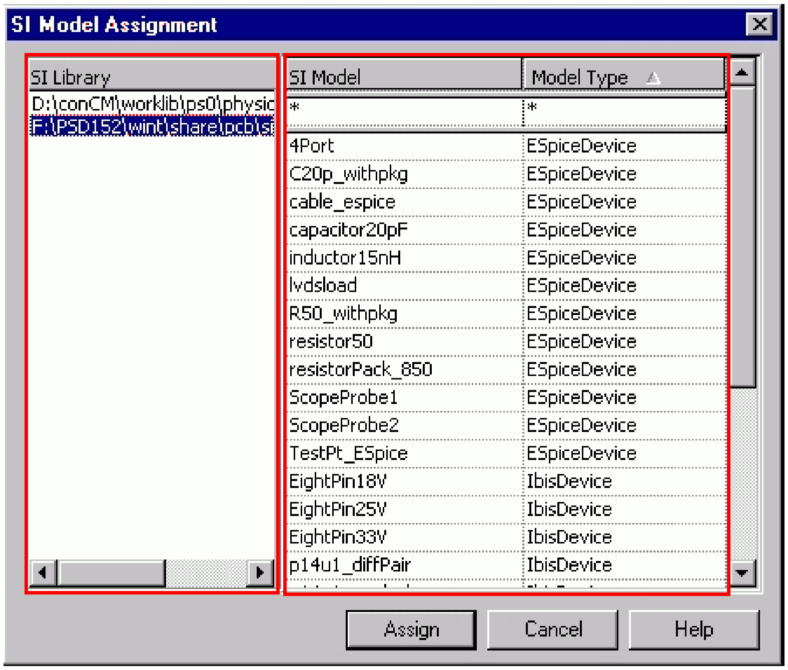
Defining a Model-Defined Diff-Pair by Assigning a Signal Model to an IC Device

You can assign an appropriate signal model to a device to assign a pair of nets or XNets you want routed as diff-pairs. Such diff-pairs are called model-defined diff-pairs. You specify model-defined differential pairs in a device model. For components other than discretes, you need to manually select signal models from the list of available models in the device model library. You use the *Assign Signal Models* menu command of the Model Assignment window to assign a signal model to a component instance or multiple instances of the same type in a schematic.

To assign a signal model to an instance:

1. Right-click the instance in the Model Assignment window.
2. On the pop-up menu, select *Assign SI Model* menu.

This launches the SI Model Assignment window.



SI Model Assignment

SI Library
D:\conCM\worklib\ps0\physic
F:\PSD152\wint\share\pcb\si

SI Model	Model Type
*	*
4Port	ESpiceDevice
C20p_withpkg	ESpiceDevice
cable_espice	ESpiceDevice
capacitor20pF	ESpiceDevice
inductor15nH	ESpiceDevice
lvdsload	ESpiceDevice
R50_withpkg	ESpiceDevice
resistor50	ESpiceDevice
resistorPack_850	ESpiceDevice
ScopeProbe1	ESpiceDevice
ScopeProbe2	ESpiceDevice
TestPt_ESpice	ESpiceDevice
EightPin18V	IbisDevice
EightPin25V	IbisDevice
EightPin33V	IbisDevice
p14u1_diffPair	IbisDevice

The SI Model Assignment window displays all the valid device model libraries in the left pane. All the available signal models along with their model types are displayed on the right pane. You can browse through the available libraries and assign a model to the device. You can also filter the model based on name and the model type applicable on the component selected, such as IbisDevice for IC, PackageModel for connectors, and EspiceModels for discrete.

Note: You can also launch the SI Model Assignment window to assign models using one of the following ways:

- ❑ Right-click a component instance in the schematic and select *Assign SIgnal Model* from the pop-up menu.
- ❑ Choose *Text – Assign Signal Model* and click a component instance in the schematic.
- ❑ Choose *Group – Assign Signal Model [A]* and select a group.

3. Select a signal model from the list and click *Assign*.

A validation routine is performed and if it succeeds, that is if the model that you assigned maps with the instance, the model is applied to the device.

However, if the validation routine fails, a message stating that the signal model does not map with the instance, is displayed. If there is a pin type, pin number, or pin model mismatch, the signal model appears in bold blue color in the SI Model column. In case the path to the `.dml` file containing the signal model is not set up correctly, the signal model appears in bold red color in the SI Model column.

Some of the errors can result because of the following reasons:

❑ Pin type, pin number, or pin model mismatch between the device and the assigned model.

❑ An inappropriate model type is assigned to a component instance. For example, an IbisDevice model could be assigned to a discrete device or an ESpiceDevice model type could be assigned to a discrete device.

Tip

To obtain a detailed list of errors that occur while assigning a signal model to a component, click the *Details* button on the bottom right corner of the Model Assignment window.

Other Tasks Done from the Model Assignment GUI

This section covers the following topics:

- Resetting Instances on page 61
- Applying a Signal Model to the Schematic on page 62
- Assigning a Signal Model to Multiple Instances on page 62
- Refreshing the Model Assignment Window on page 63
- Refreshing the Model Assignment Window on page 63
- Assigning Pin Models on page 64
- Displaying the Canonical Path on page 65
- Cross Probing between Model Assignment window and Schematic Canvas on page 65
- Manual Verification of Signal Model Assignment on page 66

Resetting Instances

If you have assigned a signal model to an instance, but not yet applied the value to the schematic, you can revert to the original schematic values. The *Reset* commands help you achieve this. This option is particularly useful when you assign a signal model incorrectly and want to undo the action.

- To reset the value of an instance, right-click the instance in the second pane of the Model Assignment window and choose the *Reset Instance* command from the pop-up menu.
- To reset the values of all the instances, right-click the instance in the second pane of the Model Assignment window and choose the *Reset All* command from the pop-up menu.

Note: These commands are only available as long as you do not apply the changes to the schematic.

Applying a Signal Model to the Schematic

Merely assigning signal model to the components does not modify the schematic information. For the changes to take effect, you also need to apply the changes to the schematic.

➤ To apply change to the schematic, select the component to which you have added the signal model and click the Apply button.

Assigning a Signal Model to Multiple Instances

You can assign a signal model to multiple or all instances of a part, simultaneously. This action updates all the selected part instances with a single signal model.

To assign a signal model to multiple instances of a component:

1. On the second pane of the Model Assignment window, right-click any instance.
2. From the pop-up menu, choose *Select All*.

   To select specific part instances on the grid, use the Ctrl + click or Shift + Ctrl + click combinations.
3. Right-click again and select *Assign SI Model*.
4. Select an appropriate signal model from the SI Signal Model Assignment dialog box.
5. Click *Assign*.
6. Click *Apply*

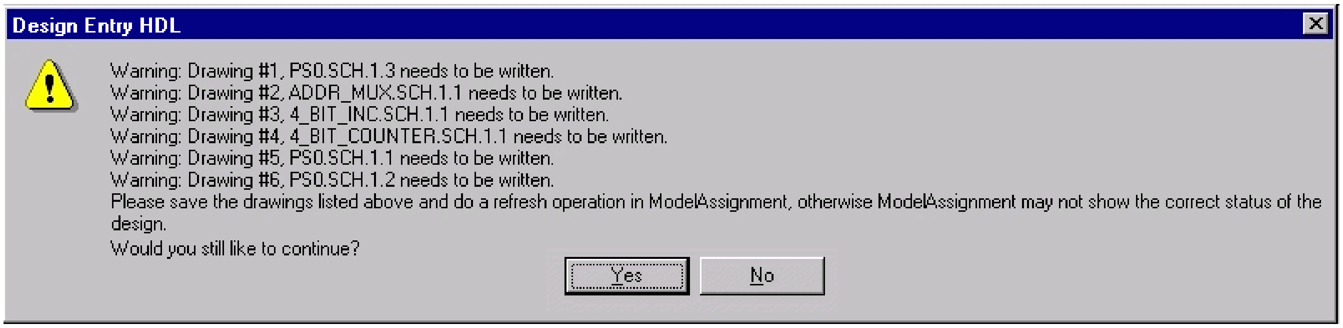
   The signal model is assigned to all the instances you selected.

Refreshing the Model Assignment Window

You can reload the Model Assignment window with the updated details of the components in the schematic. This helps you verify if the design has actually been updated with the modifications that you made from the Model Assignment window. Refreshing also helps in synchronizing the Model Assignment window with Design Entry HDL, in case you delete any signal models from a component in Design Entry HDL.

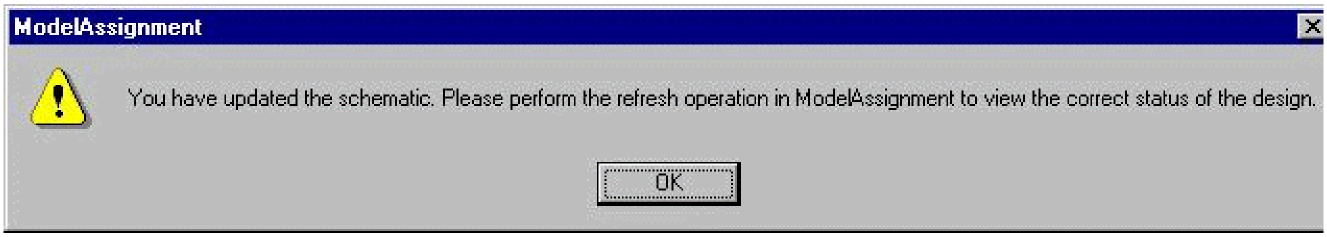
To refresh the contents of the Model Assignment window, click the *Refresh* button. The Model Assignment window reloads the updated signal model information from the schematic.

Note: If you make any changes to the schematic in Design Entry HDL and refresh the Model Assignment window, a warning message appears, prompting you to save the schematic pages where you made changes and then perform the refresh operation in the Model Assignment window.



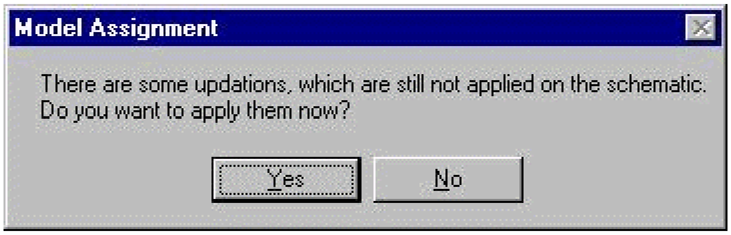
Warning: Drawing #1, PS0.SCH.1.3 needs to be written.
Warning: Drawing #2, ADDR_MUX.SCH.1.1 needs to be written.
Warning: Drawing #3, 4_BIT_INC.SCH.1.1 needs to be written.
Warning: Drawing #4, 4_BIT_COUNTER.SCH.1.1 needs to be written.
Warning: Drawing #5, PS0.SCH.1.1 needs to be written.
Warning: Drawing #6, PS0.SCH.1.2 needs to be written.
Please save the drawings listed above and do a refresh operation in ModelAssignment, otherwise ModelAssignment may not show the correct status of the design.
Would you still like to continue?

Note: If you save the schematic and then move the focus to the Model Assignment window, it prompts you to refresh the window.



You have updated the schematic. Please perform the refresh operation in ModelAssignment to view the correct status of the design.

Note: If you try to close the Model Assignment window without applying the changes on the schematic, a message prompting you to apply the changes to the schematic is displayed.



There are some updations, which are still not applied on the schematic. Do you want to apply them now?

Assigning Pin Models

In addition to assigning models to component instances, you can assign pin models from the Model Assignment user interface. You can display the pin details of each of the pins on various component instances on the Model Assignment window and then assign appropriate pin models to the pins.

To assign a pin model to a pin:

1. Select the pin in the second pane of the Model Assignment window.
2. Click the *I/O Pin Details* button.

   A detailed grid of pin information of the pins on the selected component instance is displayed in a third pane. The detailed information includes pin name, pin model (if assigned), pin number, and type.

   Note: For programmable pins, all the valid pin model types are available as a drop-down list in the SI Model column in the third pane.
3. Right-click the pin in the third pane of the Model Assignment window
4. Select *Browse Pin Models* from the pop up menu.

   The SI Model Assignment window is displayed with a list of all the available pin models in the available libraries.
5. Click *Assign*.

Note: Model assignment operation on pins works on a single pin at a time. Unlike devices, simultaneous model assignment for multiple pins is not allowed.

Displaying the Canonical Path

Model Assignment provides an easy way of locating the various part instance on a schematic using the complete canonical path. You can display the full canonical or hierarchical path of all instances in the Model Assignment window.

➤ To show the hierarchical path of all part instances, move the mouse pointer over any instance name in the second pane of the Model Assignment window.

The complete canonical path of the instance will be displayed in the form of a tool tip.

You can also display the canonical path in an additional column in the Model Assignment window.

➤ To display the column, choose *Options – Show Hierarchical Path*.

Changing the Visibility of the SIGNAL_MODEL property

You can also control the visibility of the `SIGNAL_MODEL` property on the schematic at the time of applying the signal model. The *Make visible on Schematic* menu option helps you control the visibility of the property. At the time of applying a signal model, if the menu option is selected, the `SIGNAL_MODEL` property is annotated on the schematic. You can later change the visibility of the property using the Attributes dialog box.

Cross Probing between Model Assignment window and Schematic Canvas

Another way of easily locating a specific instance on a cluttered schematic is the Highlight Instance feature. The Model Assignment window provides support for cross-probing between instances in the window and the schematic. You can highlight a specific instance on a schematic from within the Model Assignment window.

To highlight an instance on a schematic from within the Model Assignment window:

1. Select the appropriate component instance in the second pane and right-click.
2. Select *Highlight Instance* from the pop-up menu.

   Alternatively, you can select *Highlight Instance* from the *File* menu. The part instance is highlighted on the schematic. This option is available for single rows only.

Similarly, you can locate a component instances in the Model Assignment window from schematic canvas. Choose the *Highlight Instance* command on the pop-up menu that appears when you right-click a component instance in the schematic. The corresponding entry for the instance will be highlighted in the second pane of the Model Assignment window.

Manual Verification of Signal Model Assignment

The *Highlight Instance* command is also useful in ascertaining if the model has actually been assigned to an instance. After assigning a model to an instance, you can highlight the instance from within the Model Assignment window. You can then confirm the assignment by right-clicking on the component (device) in the schematic and selecting Attribute.

You will see the newly assigned signal model listed as one of the attributes on the device. You cannot add or edit any existing `SIGNAL_MODEL` property here. However, you can delete a user-defined signal model.

Creating XNets in DML-Independent Mode

The DML-independent mode in Design Entry HDL enables you to create XNets for discrete devices without DML models.

For more information about creating XNets in DML-independent mode, refer to the *DML-Independent Flow in Allegro Design Authoring (DE-HDL)* application note.

XNets Visualization

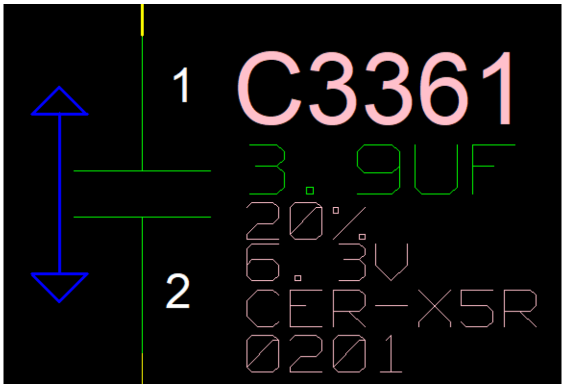
In DE-HDL, you can show or hide a visual indication on components with and without XNets on their pins. When this option is on, all the components that have XNets on their pins are marked with a blue arrow and all the components that do not have XNets on their pins are marked with a blue cross sign. This indication is displayed on components that are 2-pin discrete devices and are electrical parts.

By default, this feature is disabled and visual indication is not shown. To enable the visual indication by default when you open a design in DE-HDL, set the value of *SHOW_XNET_STATUS* directive to `ON` in the *cds.cpm* file. You can also enable the visual indication using the *Show XNet Status* menu option.

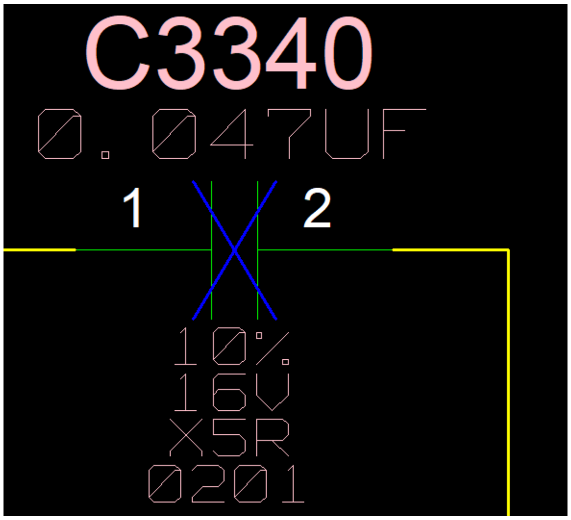
To enable the visual indication on components using the menu:

- Do one of the following:
  - ❑ If Windows mode is enabled, choose *Edit* - *Component* - *Show XNet Status*.
  - ❑ If Windows mode is disabled, choose *Component* - *Show XNet Status*.

A blue arrow is displayed on the components that have XNet in their pins:



A blue cross is displayed on the components do not have XNets on their pins:



You can also set the default blue color of the arrow and cross signs by setting the *XNET_ABSENT_COLOR* and *XNET_EXISTS_COLOR* directives in the `.cpm` file.

Important

This functionality is available for DML-Independent designs only.

Working with XNets in Constraint Manager

This section discusses the following topics:

- Showing XNet in Constraint Manager on page 68

Showing XNet in Constraint Manager

By default, an XNet is displayed without its members. To view a list of the members of the XNet:

1. In Constraint Manager connected to Design Entry HDL, choose *Objects – Filter*.

   The Filter dialog box displays.

2. Select *XNet* under the Select Object Types list.
3. Click *OK*.

   All the XNets are displayed in the Constraint Manager spreadsheet.

4. Click the plus sign (+) next to an XNet name.

   All the members of the XNet are displayed.

The Type column to the left of net objects indicates the presence of the newly created XNet.You can also move your mouse pointer over the net/XNet names in the Constraint Manager spreadsheet, a tool tip indicating the name and type displays.

Renaming XNets

The support for renaming XNets in a design aims at providing you greater control over managing XNets in the Design Entry HDL-Constraint Manager flow. In Constraint Manager, by default, the XNet name is taken from the lowest alphabetical name of the member nets. From this release onwards, you can name the XNet to any of its members in Constraint Manager and the name is honored in the front-to-back flow.

For example, if an XNet is created from nets *A_NET*, *B_NET,* and *C_NET*, the XNet is named *A_NET*, by default. However, with the support for XNet renaming, you can define from which net name the XNet takes its name

Renaming an XNet

To rename an XNet, perform the following steps:

1. With the XNet selected in the *Objects* column, right click and choose *Rename* from the pop-up menu.

The Rename XNet dialog box appears.

2. In the Rename XNet dialog box, choose from the available net names (that comprise the XNet) in the drop-down menu.
3. Click *OK*.

   The XNet is renamed accordingly.

Other Operations on XNets

This section contains the following topics:

- Handling Constraints on XNet Segments on page 69
- Showing Pin-Pair Constraints in a Schematic on page 75
- Cross-Probing between Constraint Manager and Design Entry HDL for XNets on page 76
- Controlling the number of nets in an XNet on page 76
- Defining Number of Pins in an XNet on page 76

Handling Constraints on XNet Segments

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.

Now, 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:

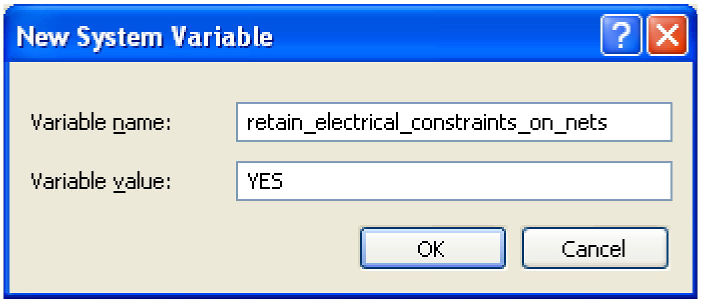
```
RETAIN_ELECTRICAL_CONSTRAINTS_ON_NETS  'YES'
```

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:

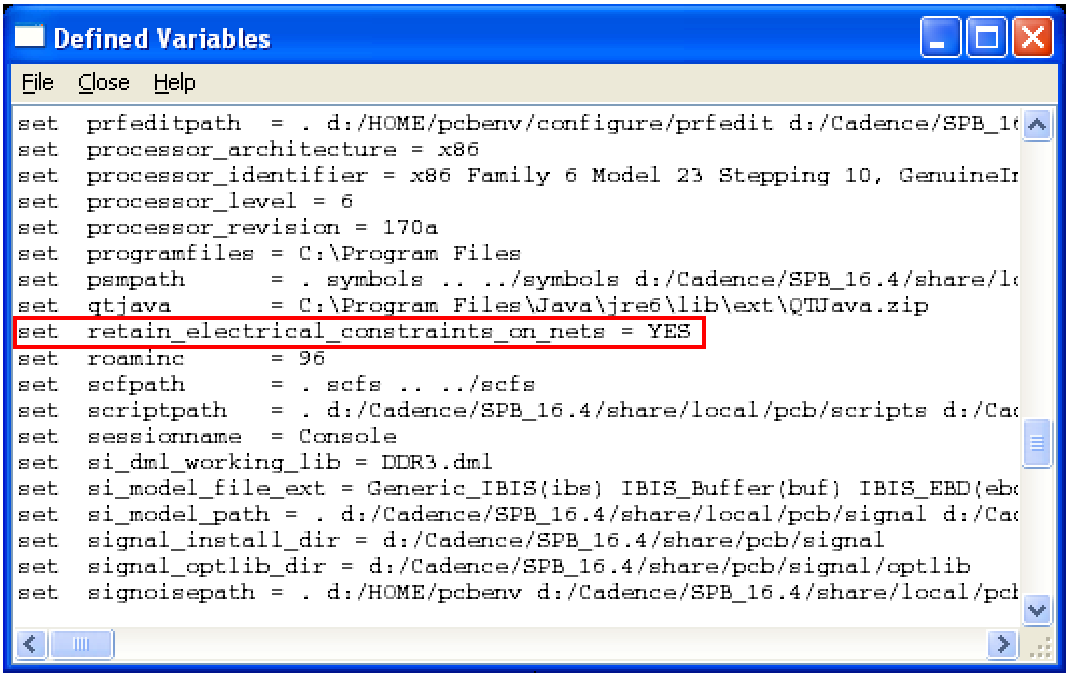

New System Variable

Variable name: retain_electrical_constraints_on_nets

Variable value: YES

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.



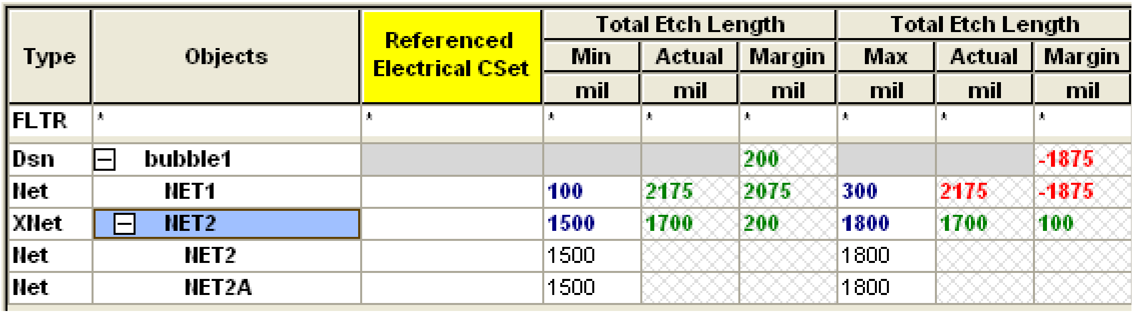
```
set  prfeditpath   = . d:/HOME/pcbenv/configure/prfedit d:/Cadence/SPB_16
set  processor_architecture = x86
set  processor_identifier = x86 Family 6 Model 23 Stepping 10, GenuineIn
set  processor_level = 6
set  processor_revision = 170a
set  programfiles = C:\Program Files
set  psmpath       = . symbols .. ../symbols d:/Cadence/SPB_16.4/share/lo
set  qtjava        = C:\Program Files\Java\jre6\lib\ext\QTJava.zip
set  retain_electrical_constraints_on_nets = YES
set  roaminc       = 96
set  scfpath       = . scfs .. ../scfs
set  scriptpath    = . d:/Cadence/SPB_16.4/share/local/pcb/scripts d:/Cad
set  sessionname   = Console
set  si_dml_working_lib = DDR3.dml
set  si_model_file_ext = Generic_IBIS(ibs) IBIS_Buffer(buf) IBIS_EBD(ebd
set  si_model_path = . d:/Cadence/SPB_16.4/share/local/pcb/signal d:/Cad
set  signal_install_dir = d:/Cadence/SPB_16.4/share/pcb/signal
set  signal_optlib_dir = d:/Cadence/SPB_16.4/share/pcb/signal/optlib
set  signoisepath = . d:/HOME/pcbenv d:/Cadence/SPB_16.4/share/local/pcb
```

Note: 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 Allegro environment variable is checked. If none of these options is found, the option is assumed to be off.

Caution

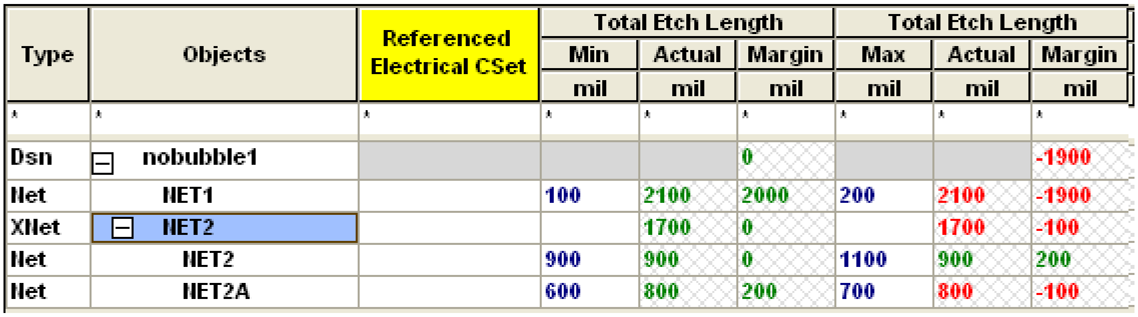
The option is set when the design is created. You cannot change the option after that.

Table 2-2 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 2-3 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

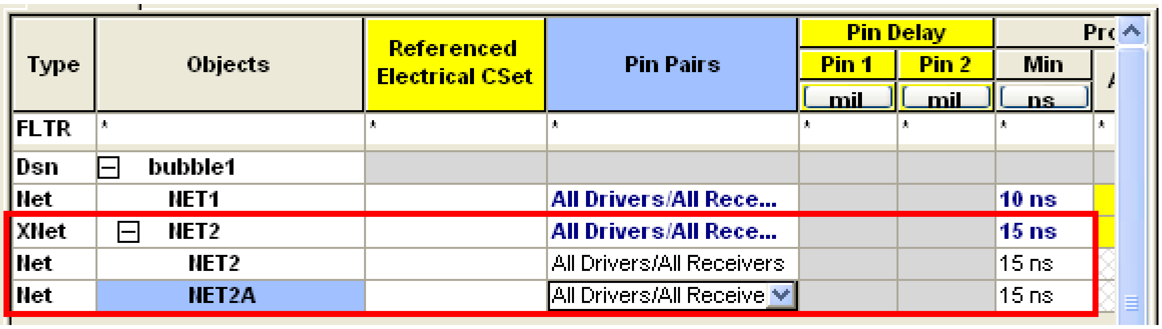
Similar to electrical constraints, ECSet assignment is also moved between member nets and the owner XNet. If the retain electrical constraints at net level option is on, these assignments are not moved. You can assign separate ECSets for a net and its owner XNet. If both ECSets contain topology data, 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.

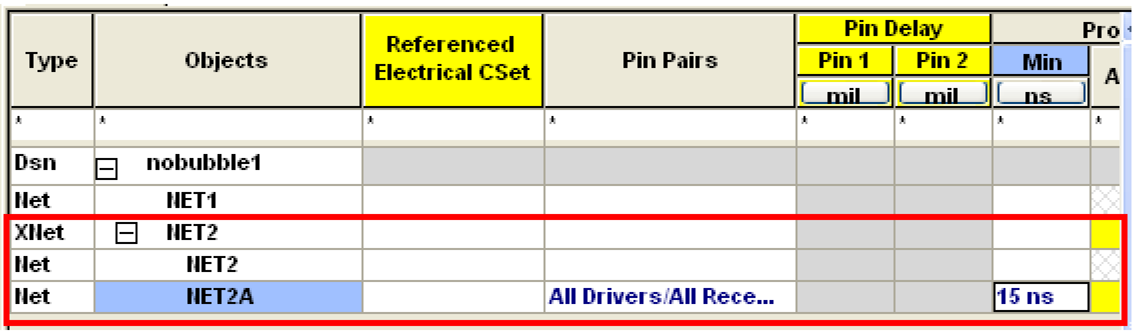
Constraint Manager Behavior

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.



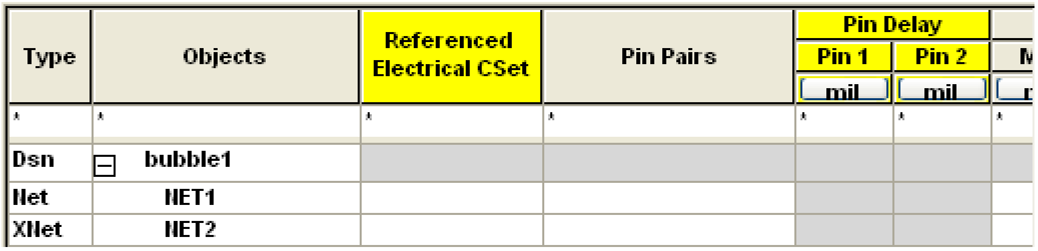
Type	Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Pro
				Pin 1	Pin 2	Min
				mil	mil	ns
FLTR	*	*	*	*	*	*
Dsn	⊟ bubble1					
Net	NET1		All Drivers/All Rece...			10 ns
XNet	⊟ NET2		All Drivers/All Rece...			15 ns
Net	NET2		All Drivers/All Receivers			15 ns
Net	NET2A		All Drivers/All Receive			15 ns

If the retain electrical constraints at net level option is turned on, the constraint remains on the member net.



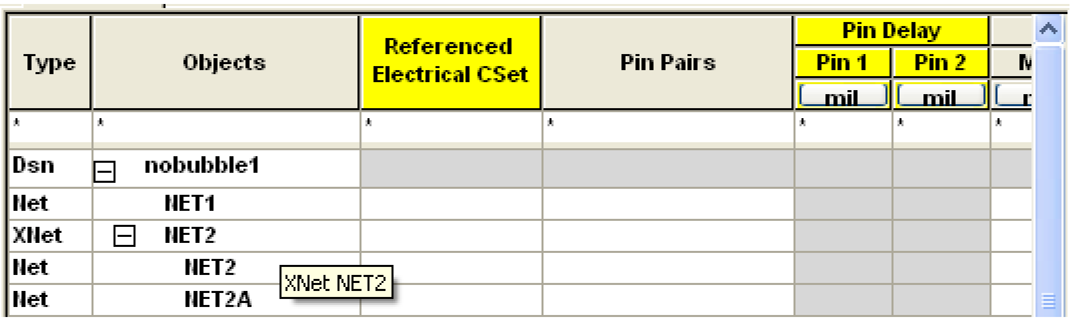
Type	Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Pro
				Pin 1	Pin 2	Min
				mil	mil	ns
*	*	*	*	*	*	*
Dsn	⊟ nobubble1					
Net	NET1					
XNet	⊟ NET2					
Net	NET2					
Net	NET2A		All Drivers/All Rece...			15 ns

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	
*	*	*	*	*	*	*
Dsn	⊟ bubble1					
Net	NET1					
XNet	NET2					

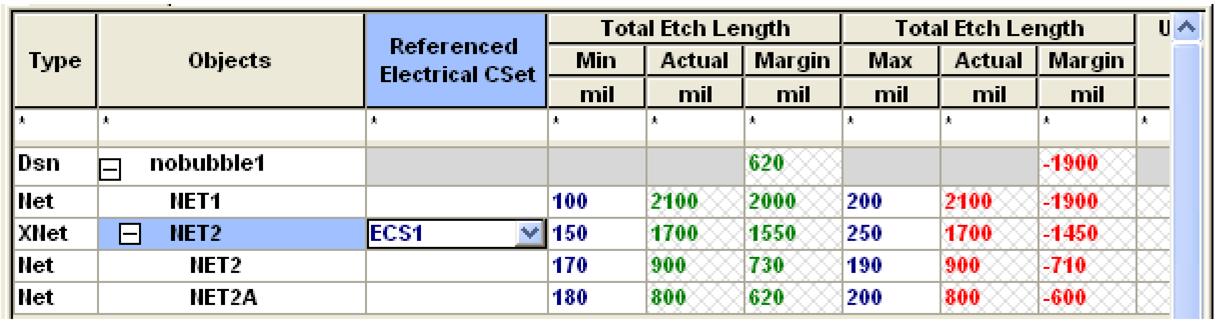
If the retain electrical constraints at net level option is enabled, nets are displayed as children of their XNet, by default. You can change this behavior from the Constraint Manager Filter dialog.



Type	Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		M
				Pin 1	Pin 2	
				mil	mil	
*	*	*	*	*	*	*
Dsn	⊟ nobubble1					
Net	NET1					
XNet	⊟ NET2					
Net	NET2					
Net	NET2A					

XNet NET2

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.

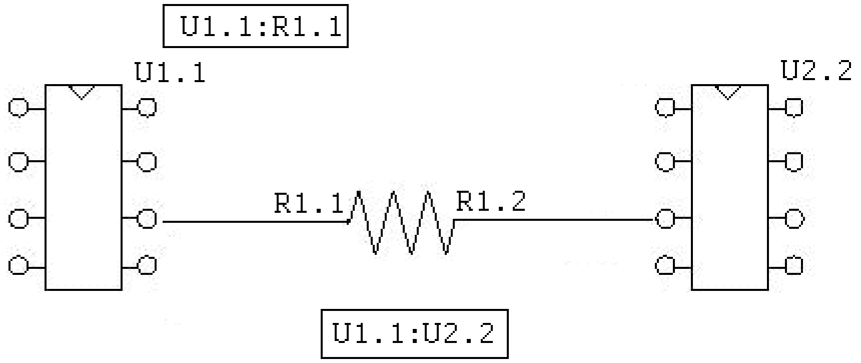
Similarly, if the corresponding environment variable is specified, it is processed for the new design. If the option differs between front end and back end for an existing design, the F2B flow fails. You need to update either the FE or BE before you re-run the F2B flow

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.

Showing Pin-Pair Constraints in a Schematic

If you add pin-pair constraints on driver-discrete and driver-receiver in Constraint Manager, only the constraints on driver-discrete are annotated to Design Entry HDL. XNet level pin-pair constraints (between driver-receiver) stay in Constraint Manager and are not backannotated to Design Entry HDL.



For example, in the figure shown above only the constraints on the pin-pair, U1.1:R1.1 are written back to the schematic. The constraints on the XNet-level pin pair, U1.1:U2.2, stay in Constraint Manager.

Note: Pin-pair constraints cannot be specified in Design Entry HDL.

Cross-Probing between Constraint Manager and Design Entry HDL for XNets

When an XNet is selected in a Constraint Manager spreadsheet, the member nets are displayed in the Global Navigation window in Design Entry HDL. Similarly, when cross probing is initiated from Design Entry HDL to Constraint Manager by selecting the member nets on the schematic, the XNet is selected in the Constraint Manager.

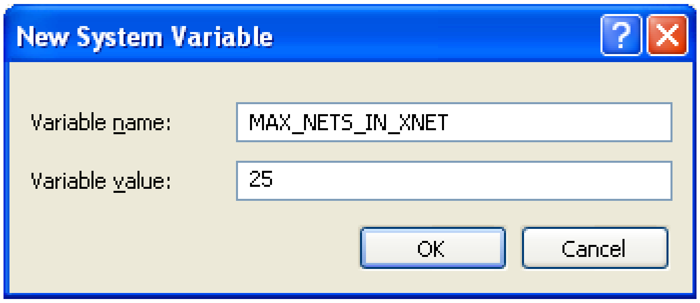
Controlling the number of nets in an XNet

You can control the maximum number of nets that will be allowed in an XNet using the following environment variable:

```
MAX_NETS_IN_XNET
```

You can set any number for this variable. The default value of this environment variable is 25.

Example



Variable name: MAX_NETS_IN_XNET

Variable value: 25

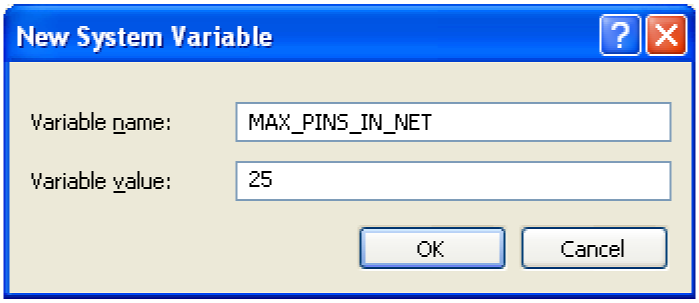
Defining Number of Pins in an XNet

You can also define the maximum number of pins in a net before it will be considered a VOLTAGE net. For this, use the following environment available:

```
MAX_PINS_IN_NET
```

You can set any number for this variable. The default value of this environment variable is 25.

Example:



New System Variable

Variable name: MAX_PINS_IN_NET

Variable value: 25

Working with Differential Pairs

This section discusses the following topics:

- Overview of User-Defined Differential Pairs on page 77
- Creating a User-Defined Differential Pair in Design Entry HDL on page 78
- Creating a User-Defined Differential Pair in Constraint Manager on page 78
- Deleting a Differential Pair on page 79
- Renaming a Differential Pair on page 80
- Creating Synonym Nets in Differential Pairs on page 82

Overview of User-Defined Differential Pairs

In addition to model-defined differential pairs, which are created when you assign a signal model to a component, you can also create user-defined differential pairs in Design Entry HDL as well as in Constraint Manager. However, it is recommended that you create differential pairs only in Constraint Manager and not in Design Entry HDL because of the following reasons:

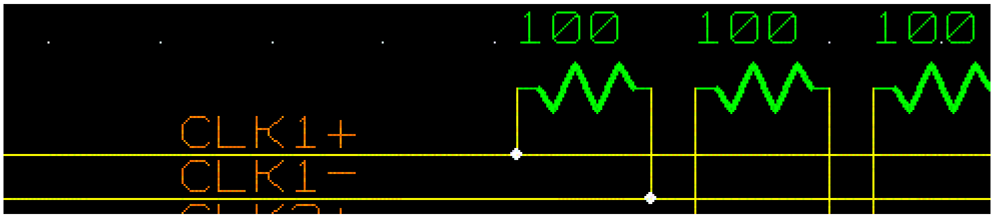
- Constraint Manager performs semantic checks when you create differential pairs.
- You can capture constraints on differential pairs only in Constraint Manager.
- Constraint Manager lets you create differential pairs automatically for nets in your design based on their names.

Note: Model-defined (Electrical) Differential Pairs take precedence over user-defined Differential Pairs and are more persistent because they can only be changed by modifying the model.

Creating a User-Defined Differential Pair in Design Entry HDL

You can create a differential pair in Design Entry HDL by assigning the `DIFFERENTIAL_PAIR` property to nets. The `DIFFERENTIAL_PAIR` property is included in the synch_props.cfg file, by default. Assign the same value to the `DIFFERENTIAL_PAIR` property on the nets constituting a differential pair.

The procedure for creating a differential pair is explained below using the following example circuit:



To create a differential pair constituting nets `CLK1+` and `CLK1-`, do the following:

1. From the Design Entry HDL menu, choose *Text – Attributes* and click on the net `CLK1+`.

   The Attributes dialog box appears.

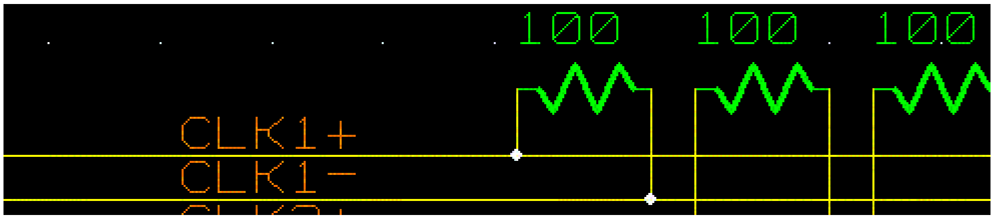
2. Add the `DIFFERENTIAL_PAIR` property with value `DP1`.
3. Repeat steps 1 and 2 for the net `CLK1-`.

When you invoke Constraint Manager, the differential pair DP1 will appear along with the other nets in the design.

Note: If you assign the `DIFFERENTIAL_PAIR` property to only one net or to more than two nets, a netlisting error is displayed when you save your design in Design Entry HDL.

Creating a User-Defined Differential Pair in Constraint Manager

You can create a differential pair in any worksheet of Constraint Manager. The procedure for creating a differential pair is explained below using the following example circuit:



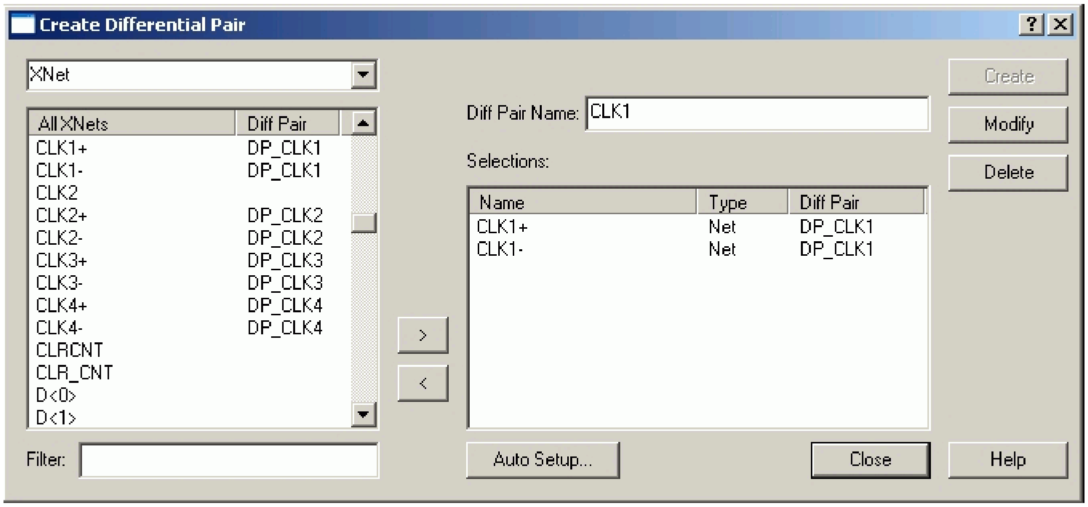
To create a differential pair constituting nets `CLK1+` and `CLK1-`, do the following:

1. In the Constraint Manager spreadsheet, select the nets `CLK1+` and `CLK1-` and right-click.

   The pop-up menu appears.

2. Choose *Create – Differential Pair*.

   The *Create Differential Pair* dialog box appears. You can see that nets `CLK1+` and `CLK1-` are members of differential pair `CLK1`.



All XNets	Diff Pair
CLK1+	DP_CLK1
CLK1-	DP_CLK1
CLK2	
CLK2+	DP_CLK2
CLK2-	DP_CLK2
CLK3+	DP_CLK3
CLK3-	DP_CLK3
CLK4+	DP_CLK4
CLK4-	DP_CLK4
CLRCNT	
CLR_CNT	
D<0>	
D<1>	

Diff Pair Name: CLK1

Selections:

Name	Type	Diff Pair
CLK1+	Net	DP_CLK1
CLK1-	Net	DP_CLK1

3. Click *Create*.

4. Click *Close*.

   The *Create Diff Pair* dialog box closes. You can view the newly created differential pair `CLK1`.

Note: You can also use the *Auto Setup* option in the *Create Diff Pair* dialog box to automatically create differential pairs for your design based on signal names. For more details on using the *Auto Setup* option, refer to the Allegro Constraint Manager User Guide.

Deleting a Differential Pair

Deleting a Differential Pair in Design Entry HDL on page 80

Deleting a Differential Pair in Constraint Manager on page 80

Deleting a Differential Pair in Design Entry HDL

You can delete the `DIFFERENTIAL_PAIR` property from nets using the *Attributes* dialog box in Design Entry HDL. The corresponding differential pair is deleted from Constraint Manager.

Note: You can delete a differential pair in Design Entry HDL, only if the `$DIFFERENTIAL_PAIR` placeholder exists on the net before the differential pair is created in Constraint Manager.

Deleting a Differential Pair in Constraint Manager

Similarly, when you delete a differential pair in Constraint Manager, it is deleted in Design Entry HDL. To delete a differential pair in Constraint Manager, perform the following steps:

1. Right-click the differential pair in the Differential Pair worksheet and choose *Delete* from the pop-up menu.

   The differential pair from deleted Constraint Manager.

2. Choose *File – Save*.
3. Choose *File – Exit*.
4. Check the Attributes dialog box for the relevant nets in Design Entry HDL.

   Note that the `DIFFERENTIAL_PAIR` property is removed from the nets.

Renaming a Differential Pair

A Library- or Model-defined differential pair is automatically named based upon the member nets comprising the differential pair. However, you might want to rename a differential pair based on specific naming conventions. Until now, you could rename only user-defined differential pairs in Constraint Manager. Now, Constraint Manager extends the support for renaming all types of differential pairs - User-defined, Library-defined, and Model-defined.

To rename a differential pair in Constraint Manager, do one of the following steps.

1. In the *Objects* column of the Differential Pair worksheet, select a differential pair

You can identify the type of differential pair by the identifiers displayed in the Type column. The table shown below lists the characters identifying different types of differential pairs:

Type Column Information	**Differential Pair Type**
DPr	User-Defined
DPr (M)	Model-Defined
DPr (L)	Library-Defined

2. Select one of the following ways to rename the differential pair:

   ❑ Choose *Object – Rename*.

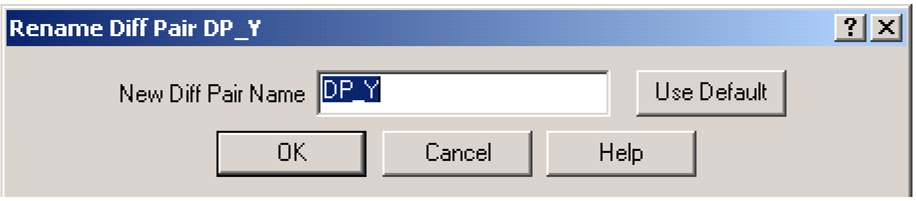
   OR

   ❑ Right-click and choose *Rename* from the pop-up menu.

   OR

   ❑ Press the *F2* key.

   The Rename dialog box appears:



Rename Diff Pair DP_Y

New Diff Pair Name: DP_Y

When you rename a library- or a model-defined differential pair, the Rename Diff Pair dialog box includes the *Use Default* button, which lets you revert to the default tool-assigned name for the differential pair.

Note: The *Use Default* button does not appear for user-defined differential pairs.

3. Specify the new name in the *New Diff Pair Name* edit box and click *OK*.

   Note: You can also rename a differential pair object from the Diff Pair Membership dialog box (*Objects – Membership – Differential Pair*).

Creating Synonym Nets in Differential Pairs

Let us suppose that two nets `N1` and `N2` are members of a differential pair, say DP1, and a third net `N3` is a member of `DP2`. Now, `N3` is made synonym to `N1`. Of the two synonym nets `N1` and `N3`, if `N1` is the base net, differential pair `DP1` is pushed to Constraint Manager and `DP2` is lost. This is logged as an error in the `concept2cm.log` file.

3

Synchronizing Constraints

Overview

All the constraints are stored in a single location, which is the Constraint Manager database. To store a constraint or a property in the schematic, you need to add the constraint to a configuration file, `synch_props.cfg.` This configuration file controls the synchronization of constraints between Design Entry HDL and Constraint Manager. All constraints are captured in the schematic canvas and then synchronized with the design.

This section contains the following topics:

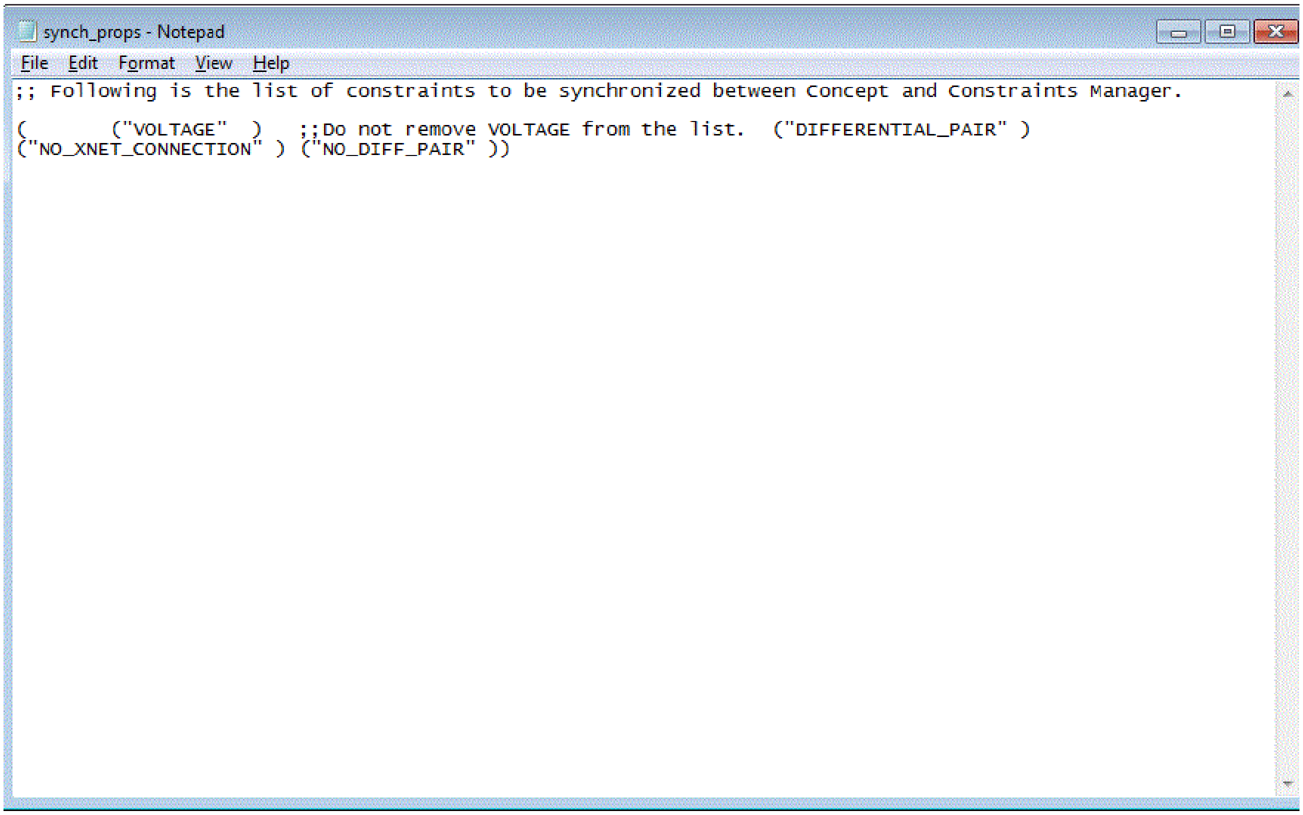
- Controlling Constraints on Schematic through a Configuration File
- Synchronizing Constraints between Schematic and Constraint Manager
- Converting Non-Synch Constraints to Synch Constraints
- Converting Synch Constraints to Non-Synch Constraints
- Plotting Constraints on a PDF Document
- Plotting Constraints on a PDF Document
- Updating Schematic Placeholders

Controlling Constraints on Schematic through a Configuration File

The properties or constraints that can reside on the schematic are stored in a configuration file, `synch_props.cfg`. This file contains a list of properties, which are synchronized between Design Entry HDL and Constraint Manager. By default, the configuration file contains two constraints, `DIFFERENTIAL_PAIR` and `VOLTAGE`. Although it is recommended that you add constraints only in Constraint Manager, these four constraints are stored in the schematic, by default as they are part of the design connectivity. Therefore, these four constraints need to be added to the schematic canvas rather than in the Constraint Manager spreadsheet.

The `synch_props.cfg` file is located at the following location:

- *<your_install_dir>\share\cdssetup*.



```
;; Following is the list of constraints to be synchronized between Concept and Constraints Manager.

(       ("VOLTAGE"  )   ;;Do not remove VOLTAGE from the list.  ("DIFFERENTIAL_PAIR" )
("NO_XNET_CONNECTION" ) ("NO_DIFF_PAIR" ))
```

Note: You can add more constraints to the configuration file and even delete the default constraints. However, it is recommended that you do not remove the `VOLTAGE` property from the configuration file.

Note: It is recommended that you capture all the constraints in Constraint Manager.

Synch Constraints - All The constraints included in the `synch_props.cfg` file are considered as *synch* constraints and can be written on to the schematic. You can add constraints to the configuration file and then write them on to the schematic. These constraints are synchronized with the Constraint Manager database, which means that the value of the constraints can be edited on the schematic canvas or in the Constraint Manager spreadsheet and is always in synch. Synch constraints are editable only if the placeholder exists on the canvas and the CM window is closed. The `DIFFERENTIAL_PAIR` constraint cannot be edited on a schematic sheet when Constraint Manager is open. The `DIFFERENTIAL_PAIR` and `VOLTAGE` properties are the default synch constraints.

Non-Synch Constraints - All the constraints, which are not listed in the configuration file, are considered as non-synch and are stored in the Constraint Manager database.

If there are non-synch constraints present on the schematic canvas, introduced by means of a copied or imported sheet, the `Synchronize` utility moves those constraints to the Constraint Manager database.

Pin-Pair and Other Constraints - There are certain non-synch constraints, including pin-pair constraints, which always remain non-synch. These are electrical constraints with more than one value or instance. These constraints should not be added to the configuration file as they will not be written to the schematic:

- `IMPEDANCE_RULE`
- `MAX_FINAL_SETTLE`
- `MAX_PARALLEL`
- `MIN_FIRST_SWITCH`
- `PROPAGATION_DELAY`
- `PULSE_PARAM`
- `RELATIVE_PROPAGATION_DELAY`
- `RELATIVE_SKEW`
- `NET_PHYSICAL_TYPE`
- `NET_SPACING_TYPE`

Important

The above mentioned constraints cannot be stored on the schematic. Therefore, you should not include entries for such constraints in the configuration file. Even if you add these constraints to the `sych_props.cfg` file, these constraints cannot become synch constraints. If you try to add any of these constraints to the schematic using the Attributes form, you see a message similar to the following message in the Design Entry HDL console window.
The `PROPAGATION_DELAY` constraint can only be edited using Constraint Manager.

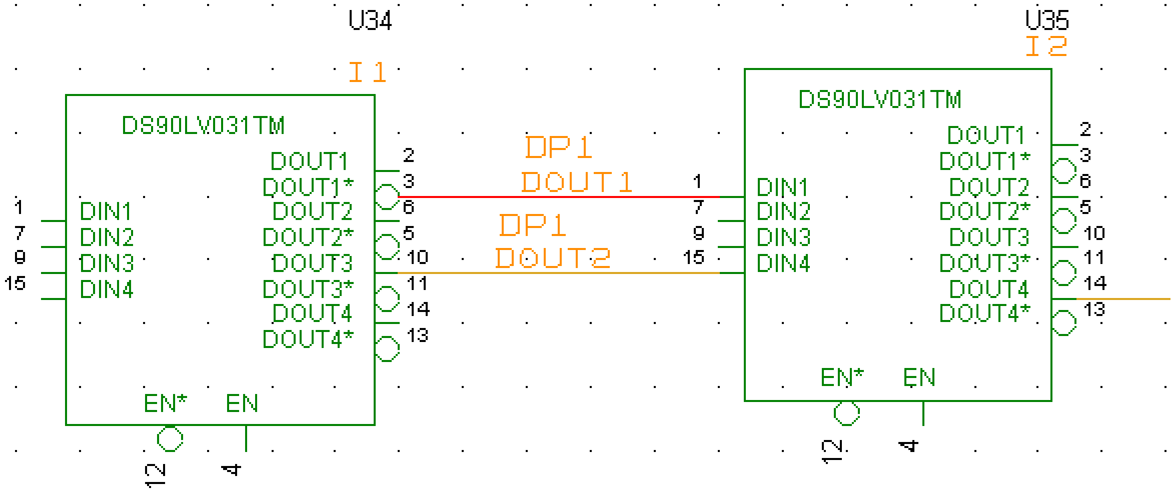
Synchronizing Constraints between Schematic and Constraint Manager

This section covers the following topics:

- Synch Constraints Added in Schematic and Edited in Constraint Manager
- Non-Synch Constraints Added in Constraint Manager

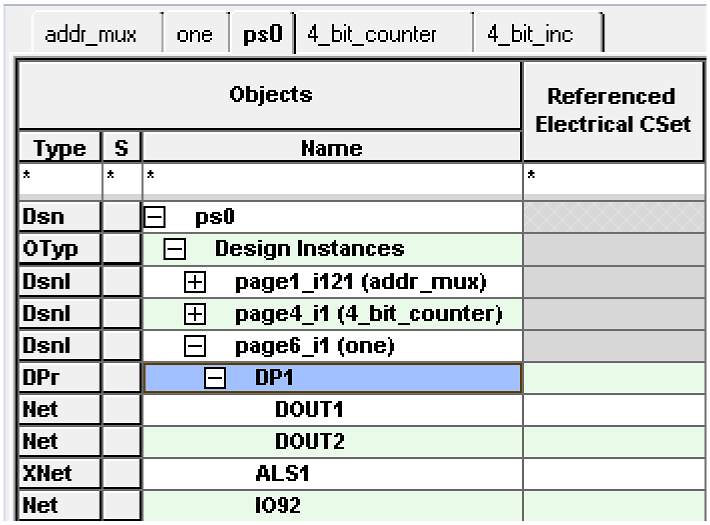
Synch Constraints Added in Schematic and Edited in Constraint Manager

1. Create a `DIFFERENTIAL_PAIR` constraint named *DP1* for the `DOUT1` and `DOUT2` nets on the schematic.



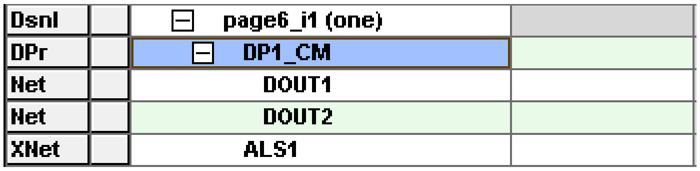
2. Launch Constraint Manager.

Note that the `DIFFERENTIAL_PAIR` constraint is visible in Constraint Manager.



Objects			Referenced Electrical CSet
Type	S	Name	
*	*	*	*
Dsn		⊟ ps0	
OTyp		⊟ Design Instances	
DsnI		⊞ page1_i121 (addr_mux)	
DsnI		⊞ page4_i1 (4_bit_counter)	
DsnI		⊟ page6_i1 (one)	
DPr		⊟ DP1	
Net		DOUT1	
Net		DOUT2	
XNet		ALS1	
Net		IO92	

3. Rename the differential pair to *DP1_CM*.

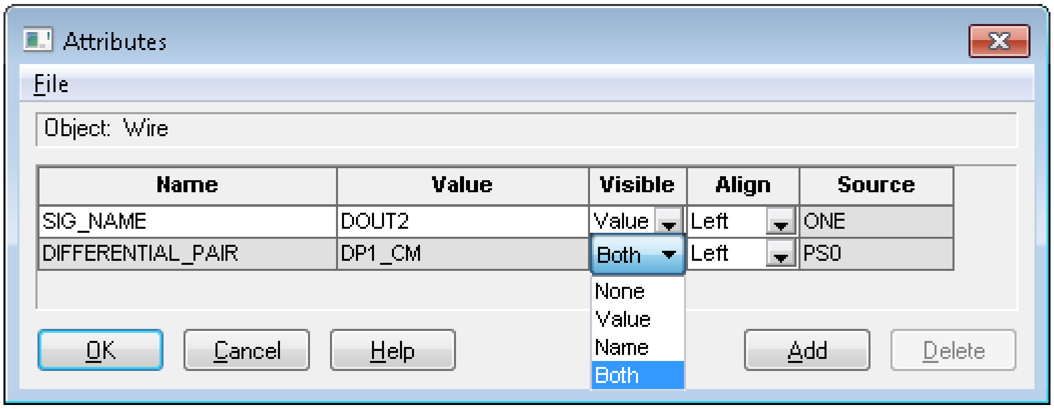


DsnI		⊟ page6_i1 (one)	
DPr		⊟ DP1_CM	
Net		DOUT1	
Net		DOUT2	
XNet		ALS1	

4. Save and exit from Constraint Manager.

   The Constraint Manager changes are automatically reflected in the schematic when you refresh the schematic.

The constraints that you add in Constraint Manager, including electrical constraints on a differential pair, are visible in the Attributes dialog box of Design Entry HDL. To annotate these constraints on the schematic canvas, add placeholders by changing the visibility of the constraints to *Value* or *Both*.



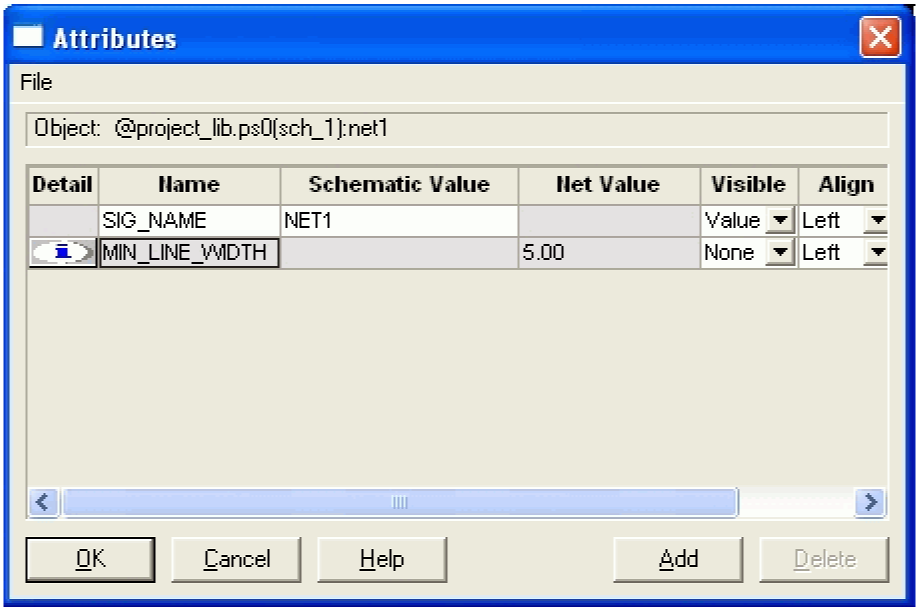
Object: Wire

Name	Value	Visible	Align	Source
SIG_NAME	DOUT2	Value	Left	ONE
DIFFERENTIAL_PAIR	DP1_CM	Both	Left	PS0

Non-Synch Constraints Added in Constraint Manager

1. Launch Constraint Manager.
2. Capture `MIN_LINE_WIDTH` constraint on a net in Constraint Manager.
3. Save and exit from Constraint Manager.
4. In Design Entry HDL, right-click the net on which you captured the constraint, and choose *Attributes* from the pop-up menu.

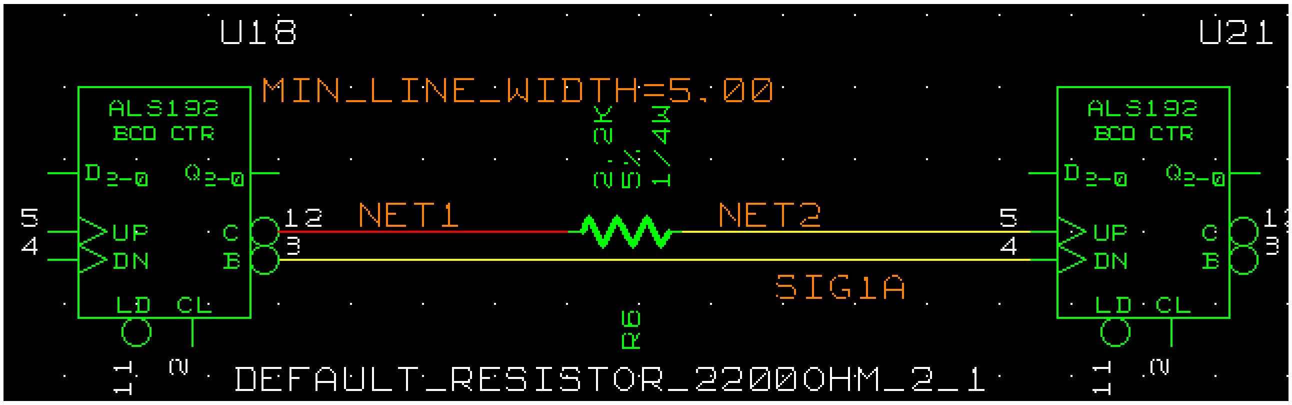
The Attributes dialog box shows the `MIN_LINE_WIDTH` property and the *Net Value* column lists the value of the constraint as `5.00`.



Object: @project_lib.ps0(sch_1):net1

Detail	Name	Schematic Value	Net Value	Visible	Align
	SIG_NAME	NET1		Value	Left
	MIN_LINE_WIDTH		5.00	None	Left

5. Change the visibility of the `MIN_LINE_WIDTH` property to *Value* or *Both*.



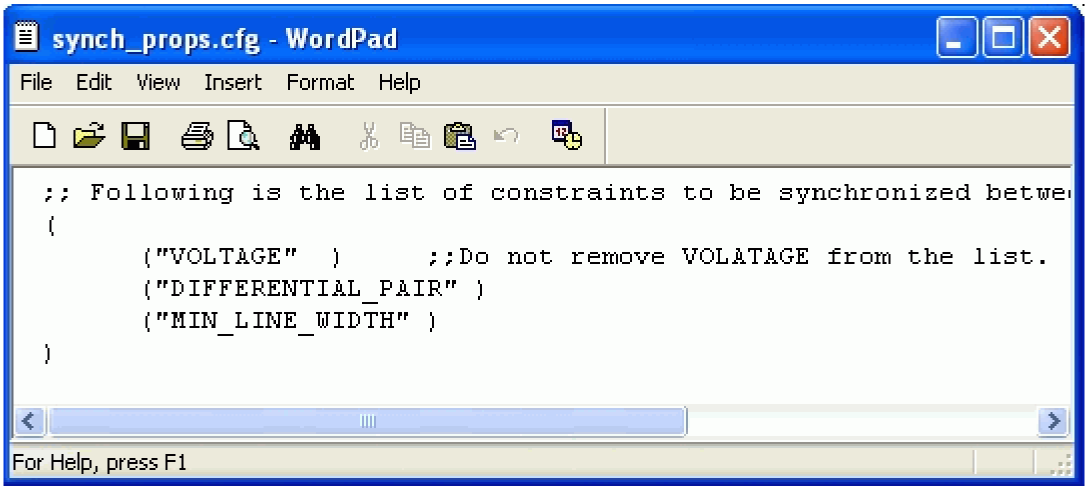
Note: If you change the value of a non-synch constraint in Constraint Manager, on exiting from the Constraint Manager the schematic placeholders are automatically refreshed and updated with the value.

Converting Non-Synch Constraints to Synch Constraints

You can convert non-synch constraints to synch constraints by adding them to the `synch_props.cfg` file.

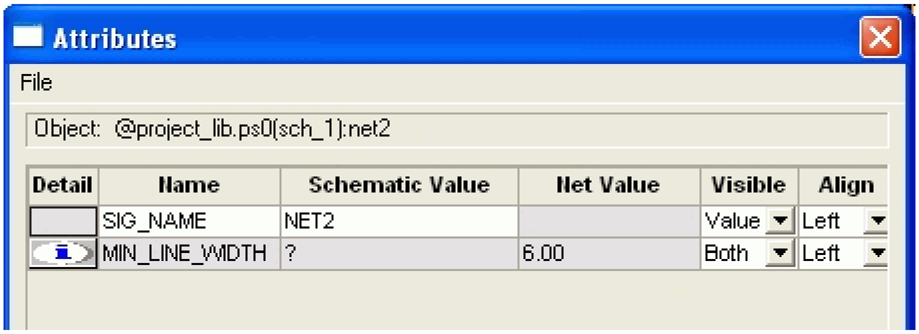
To convert non-synch constraints to synch constraints, perform the following steps.

1. Launch Constraint Manager.
2. Add the `MIN_LINE_WIDTH` constraint to a net.
3. Save and Exit from Constraint Manager.
4. Open the `synch_props.cfg` file, and add the `MIN_LINE_WIDTH` constraint.



```
;; Following is the list of constraints to be synchronized betwe
(
        ("VOLTAGE"  )      ;;Do not remove VOLATAGE from the list.
        ("DIFFERENTIAL_PAIR" )
        ("MIN_LINE_WIDTH" )
)
```

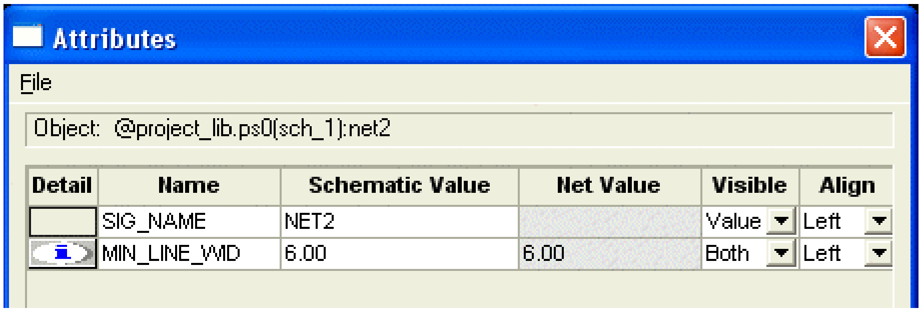
5. Create placeholder for the constraint on the net in the schematic.



Object: @project_lib.ps0(sch_1):net2

Detail	Name	Schematic Value	Net Value	Visible	Align
	SIG_NAME	NET2		Value	Left
	MIN_LINE_WIDTH	?	6.00	Both	Left

6. Save the design.

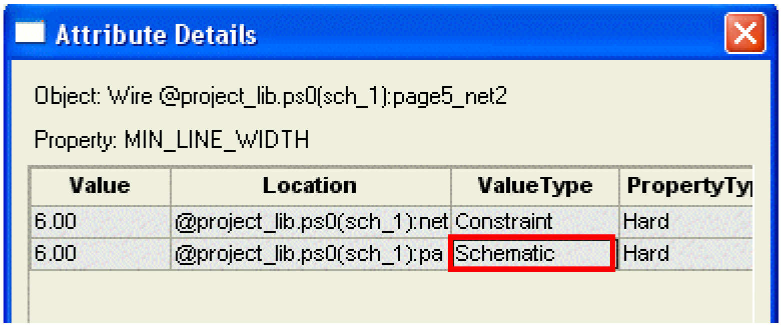


Object: @project_lib.ps0(sch_1):net2

Detail	Name	Schematic Value	Net Value	Visible	Align
	SIG_NAME	NET2		Value	Left
ℹ	MIN_LINE_WID	6.00	6.00	Both	Left

The value of the MIN_LINE_WIDTH constraint is backannotated to the schematic and the schematic placeholder for the constraint is refreshed.

7. Click the ℹ icon to open the Attributes details window.



Object: Wire @project_lib.ps0(sch_1):page5_net2

Property: MIN_LINE_WIDTH

Value	Location	ValueType	PropertyTy
6.00	@project_lib.ps0(sch_1):net	Constraint	Hard
6.00	@project_lib.ps0(sch_1):pa	Schematic	Hard

Note that the *Value Type* for MIN_LINE_WIDTH is *Schematic*, which means that it is a synch constraint.

Converting Synch Constraints to Non-Synch Constraints

You can also convert synch constraints to non-synch constraints by removing them from the synch_props.cfg file. If you delete a constraint from the synch_props.cfg file, launch Constraint Manager. The constraint is automatically deleted from the Constraint Manager database.

To convert a synch constraint to a non-synch constraint, perform the following steps:

1. Remove the DIFFERENTIAL_PAIR constraint from the synch_props.cfg file.
2. Launch Design Entry HDL.
3. Delete the DIFFERENTIAL_PAIR constraint from the appropriate nets in the Attributes dialog box.

4. Choose *Tools – Constraints – Edit* to launch Constraint Manager.

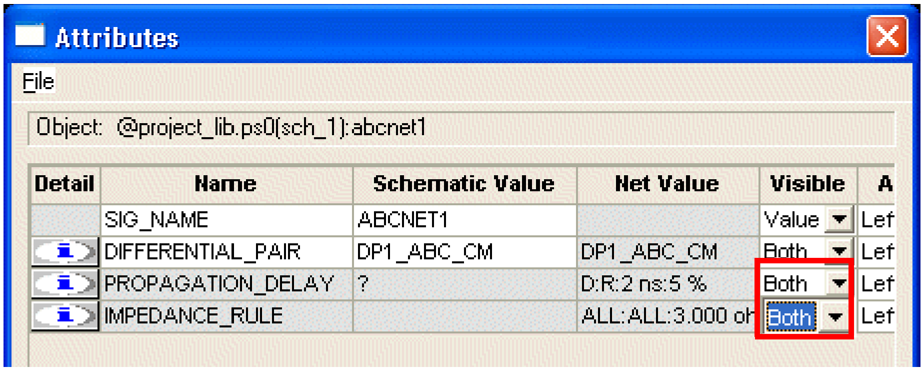
   The `DIFFERENTIAL_PAIR` constraint is also deleted from Constraint Manager.

Plotting Constraints on a PDF Document

To plot constraints in a schematic on a PDF document, you need to create placeholders in the schematic. You can plot synch constraints as well as non-synch constraints. However, schematic placeholders for non-synch constraints are not automatically updated on exiting from Constraint Manager.

You plot non-synch constraints to a PDF document by performing the following steps:

1. Create placeholders on the schematic.

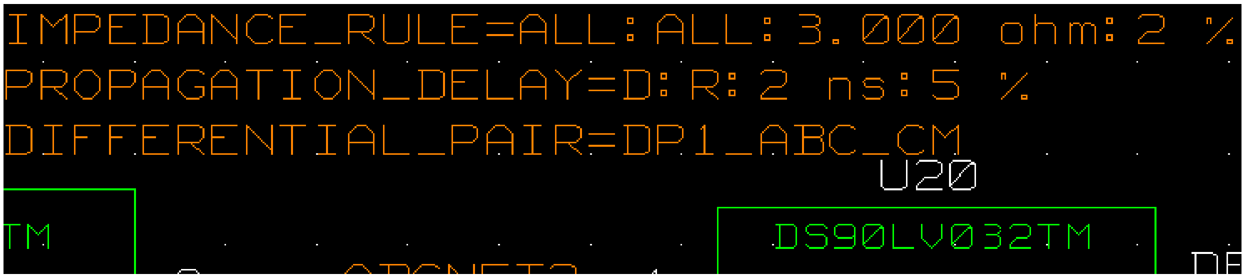


Object: @project_lib.ps0(sch_1):abcnet1

Detail	Name	Schematic Value	Net Value	Visible	A
	SIG_NAME	ABCNET1		Value	Lef
i	DIFFERENTIAL_PAIR	DP1_ABC_CM	DP1_ABC_CM	Both	Lef
i	PROPAGATION_DELAY	?	D:R:2 ns:5 %	Both	Lef
i	IMPEDANCE_RULE		ALL:ALL:3.000 oh	Both	Lef

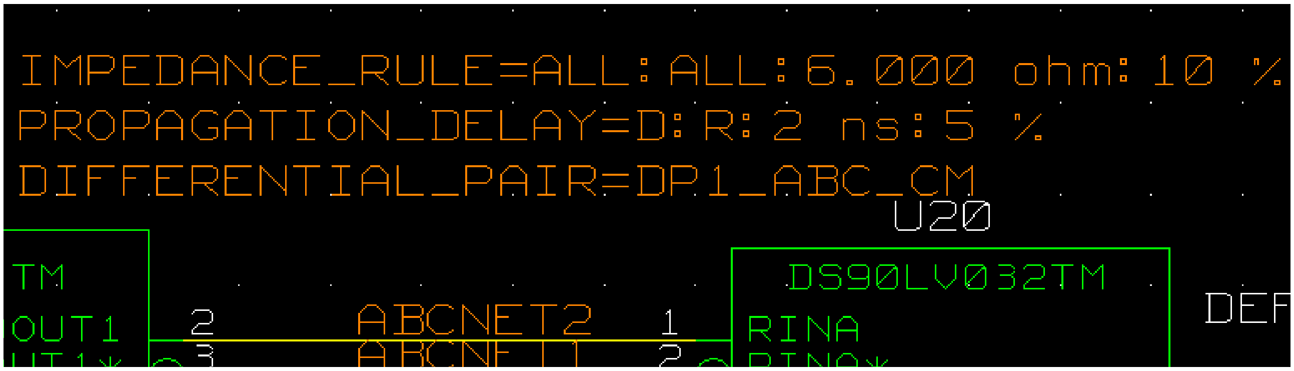
2. Save the design.

   Note that placeholders are not updated.



IMPEDANCE_RULE=ALL:ALL:3.000 ohm:2 %
PROPAGATION_DELAY=D:R:2 ns:5 %
DIFFERENTIAL_PAIR=DP1_ABC_CM

Placeholders are updated. Constraints are now visible from the `.dcf` file



```
IMPEDANCE_RULE=ALL:ALL:6.000 ohm:10 %
PROPAGATION_DELAY=D:R:2 ns:5 %
DIFFERENTIAL_PAIR=DP1_ABC_CM
```

Note: For more information on plotting constraints on a PDF document, refer to the *Allegro Design Publisher User Guide*.

Updating Schematic Placeholders

You need to update the schematic with the changes done on synch constraints in Constraint Manager. When you exit from Constraint Manager, all the changes done on synch constraints are backannotated to the schematic. Changes are backannotated for only those nets which belong to the top-level design. Any overrides or changes done on the lower-level blocks are not backannotated to the schematic. Each time you exit from Constraint Manager, schematic is backannotated with the changes in synch constraints. Also, backannotation for a net takes place only if placeholder exists for that net or its synonym in the schematic.

4

Electrical Constraints

Electrical constraints (ECs) govern the electrical behavior of a net or pin-pair in a design. For example, you can capture a constraint to define the maximum voltage overshoot tolerated by a net and capture the minimum first switch delay for a driver-receiver pin-pair in your design.

You can use Constraint Manager with design capture tools, Design Entry HDL and System Connectivity Manager, to capture and manage electrical constraints as you implement logic. The changes that you make to constraint information in Constraint Manager are displayed in these design capture tools. Similarly, the changes that you make to constraint information in the design capture tools are displayed in Constraint Manager.

Constraint Manager’s user-friendly interface allows you to quickly capture and manage electrical constraints. Constraint Manager validates the constraint information that you enter and passes the information in the correct syntax to the design capture tools.

Note the following when working with constraints in Constraint Manager:

- If you capture a constraint on a bus (vectored net) or vectored pin, the constraint is applicable to all the bits of the bus or vectored pin. You can also capture a constraint on a bit of a bus or vectored pin. A constraint captured on a bit of a bus or vectored pin overrides a constraint captured on the bus or vectored pin.
- If a net is aliased to another net or nets, only the base net is displayed in Constraint Manager. The base net inherits all the constraints that exist on the nets aliased to it. A constraint you add on a base net also applies to the nets aliased to it.
- If two nets having the same constraint are aliased, the constraint value on the base net is applied on both the nets.
- Undo and redo of constraints is not supported in Constraint Manager.

This chapter covers the following sections:

Capturing Electrical Constraints with ECSets

In Constraint Manager, you can capture an electrical constraint on design objects in the following two ways:

- Create an electrical constraint set (ECSet) in Constraint Manager and assign the ECSet to a net in the *Net* worksheets.
- Specify the constraints directly on an object (net or pin-pair) in the *Net* worksheets.

Important

Before you capture electrical constraints in your design, ensure that the design units in the board are the same as the design units (precision) in which you want to capture electrical constraints in Constraint Manager (that you launch from System Connectivity Manager or Design Entry HDL).

- ❑ To set the design units in Allegro, choose *Setup – Design Parameters* and select a unit from the *User units* drop-down list of the *Design* tab, in the *Drawing Parameter Editor* dialog box. The available units in the *User units* drop-down list are *Mils*, *Inch*, *Microns*, *Millimeter*, and *Centimeter*.
- ❑ To set the design units (precision) in Constraint Manager, choose *Tools – Precision* and select a unit from the *Units* drop-down list in the *Design Units and Precision* dialog box. The available units in the *Units* drop-down list are *Centimeters*, *Inches*, *Microns*, *Millimeters*, and *Mils*.

This section covers the following topics:

- Overview of ECSets on page 96
- Creating an ECSet on page 97
- Assigning ECSets on page 98
- Creating a Match Group based on an ECSet in Constraint Manager on page 103
- Auditing Electrical Constraint Set on page 105
- Working with Electrical Constraints on page 106

Overview of ECSets

An Electrical Constraint Set (ECSet) is a collection of electrical constraints that define a particular design requirement— and assign them to objects on which you want to capture the same set of constraints. For example, you can create an ECSet to define the default timing and noise tolerance for a net.

This way an ECSet can be used to define a generic set of rules applicable to a number of nets. If your design requirement changes at a later point in time, you can edit your constraint and all the objects referencing the ECSet will inherit the changed ECSet automatically. Therefore, using ECSets is a very efficient way of capturing constraints in Constraint Manager.

As design requirements change, you can:

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

Note: ECSets can be referenced by any number of objects, such as Buses, Differential Pairs, XNets, Nets, and Net Classes.

Creating an ECSet

To create an ECSet, do the following:

1. In the Constraint Manager window, click the *Electrical Constraint Set* folder.
2. Click *All Constraints*.
3. Choose *Objects – Create – Electrical CSet*.

   The Create Electrical CSet dialog box appears.
4. Specify a name for the Electrical CSet.
5. Click *OK*.
6. Click *Signal Integrity/Timing/Routing* workbook under All Constraints.

   This creates an empty ECSet. You can now specify constraint parameters in a worksheet cell of the ECSet.
7. Click the entry for the newly created ECSet.
8. Specify the required values in various sections for defining the ECSet. For example, set the values for the Reflection, Switch/Settle Delays, and Single-line Impedance etc.
9. Choose *File – Save*.

This ECSet is created.

Tip

Alternatively, in the Electrical Constraint Set object folder, click on a workbook, or a worksheet within a workbook, then click on an existing ECSet, and choose *Objects – Create – Electrical CSet.* Make sure you deselect the *copy constraints from* check box.

Note: You can also create an ECSet using the SigXplorer tool. For more information, refer to the chapter, Topology Extraction in SigXplorer.

For detailed descriptions of electrical constraints, see the Electrical Constraints Data Sheets chapter of the *Allegro Constraint Architecture* guide.

Note: An ECset in Constraint Manager maps to the `ELECTRICAL_CONSTRAINT_SET` property in Design Entry HDL. Like other properties, this is a read-only property.

Assigning ECSets

You can assign constraints across nets and objects. When you apply electrical constraints defined in an ECSet to another net, the ECSet is first validated against the net and then applied. A success or failure notification is displayed depending on whether the ECSet was assigned or not.

This topics covers the following tasks:

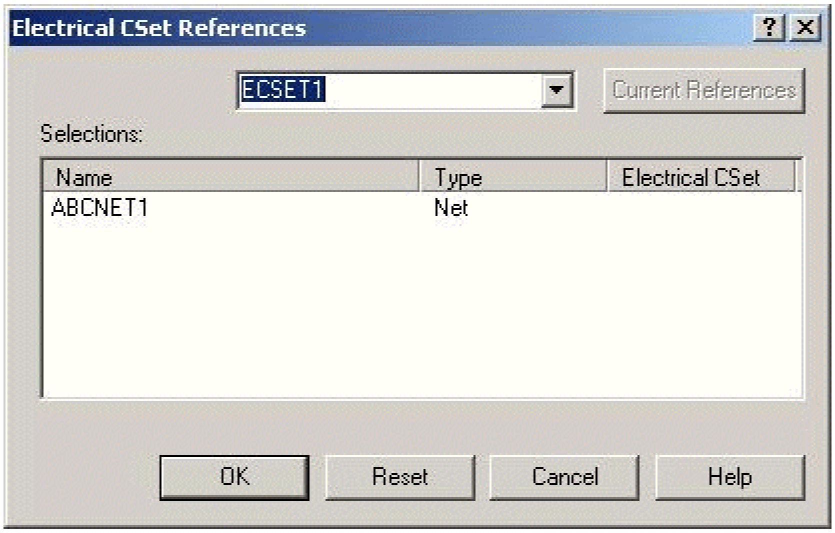
- Applying an ECSets on a Net or XNet on page 98
- Applying an ECSets on a Bus on page 100
- Applying an ECSet on a Differential Pair on page 101
- Applying an ECSet to Members of a Match group on page 102

Applying an ECSets on a Net or XNet

To apply an ECSet on a net or XNet, do the following:

1. Right-click the net under the *Net* folder and choose the *Constraint Set References* menu from the pop-up menu.
2. Select the ECSet to assign from the drop-down list in the Electrical CSet References dialog box.

3. Click *OK*.



Electrical CSet References

ECSET1

Selections:

Name	Type	Electrical CSet
ABCNET1	Net	

Figure 4-1

Tip

You can also click the Referenced Electrical CSet column next to the net name and select the ECSet from the drop-down list.

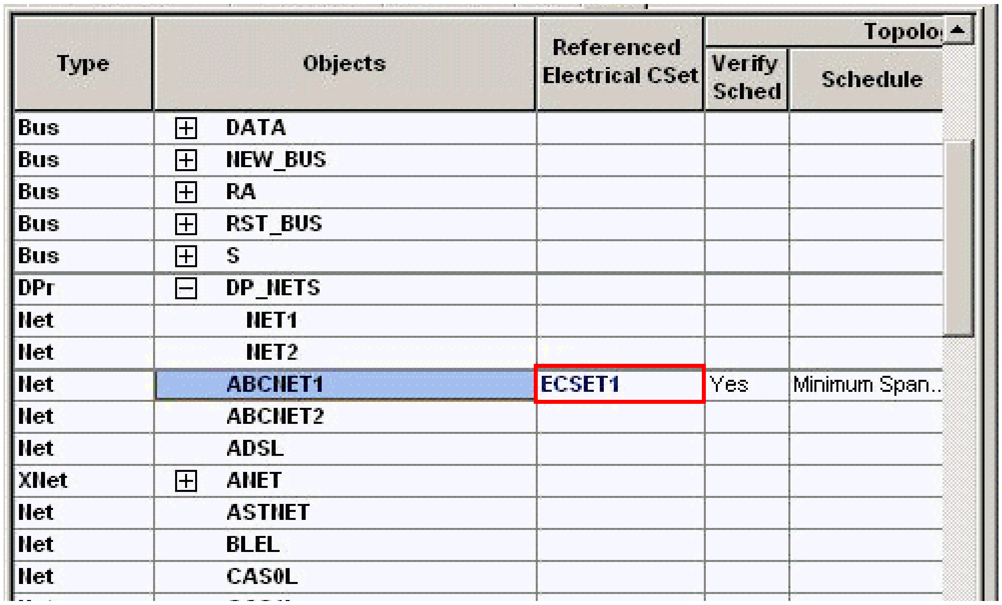
Tip

Alternatively, you can right-click on the ECSet that you created in the Electrical Constraint Set workbook, and choose the *Constraint Set References* command from the pop-up menu. Select the object type from the drop-down list, such as Net, Bus, Differential Pair, and so on. Finally, select the object from the list, click the right arrow and click *OK*.

The Electrical CSet Apply Information message box confirms that the ECSet is attached to the net.

4. Click *Close*.

The associated ECSet name appears in the Referenced Electrical CSet column next to the net name.



Type	Objects	Referenced Electrical CSet	Topolo	
			Verify Sched	Schedule
Bus	⊞ DATA			
Bus	⊞ NEW_BUS			
Bus	⊞ RA			
Bus	⊞ RST_BUS			
Bus	⊞ S			
DPr	⊟ DP_NETS			
Net	NET1			
Net	NET2			
Net	ABCNET1	ECSET1	Yes	Minimum Span..
Net	ABCNET2			
Net	ADSL			
XNet	⊞ ANET			
Net	ASTNET			
Net	BLEL			
Net	CAS0L			

Figure 4-2

Note: You can also create an ECSet in SigXplorer and apply it on the Constraint Manager objects. For more information, refer to the chapter, Topology Extraction in SigXplorer.

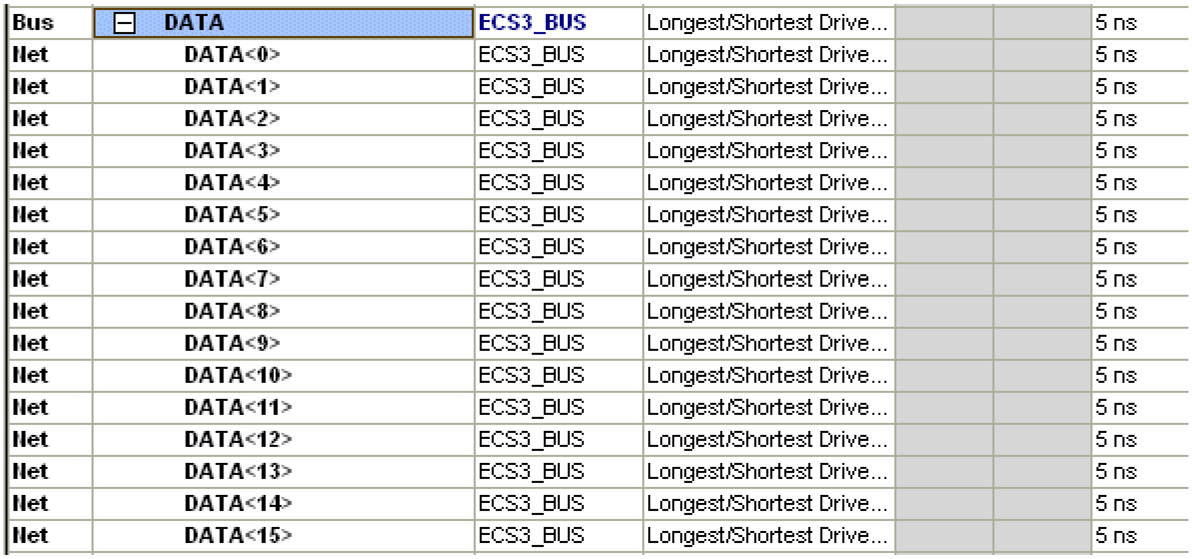
Applying an ECSets on a Bus

Constraints captured or applied on a bus (vectored net) are inherited by all members of the bus. When you associate the ECSet with a bus, all the members (bits) of the bus inherit the constraints defined in the ECSet. For example, when you associate the ECSet with a bus, all members (bits) of the bus inherit the pin-pair constraints defined in the ECSet. You need not explicitly assign constraint to individual bits of a bus. A constraint captured on a bit of a bus overrides a constraint captured on the bus.

To apply an ECSet on a bus, do the following:

1. Right-click the bus name and choose the *Constraint Set References* menu command from the pop-up menu.
2. Select the appropriate ECSet name in the Electrical CSet References dialog box.

The ECSet is assigned to the bus and all its members.



Bus	⊟ DATA	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<0>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<1>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<2>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<3>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<4>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<5>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<6>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<7>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<8>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<9>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<10>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<11>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<12>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<13>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<14>	ECS3_BUS	Longest/Shortest Drive...			5 ns
Net	DATA<15>	ECS3_BUS	Longest/Shortest Drive...			5 ns

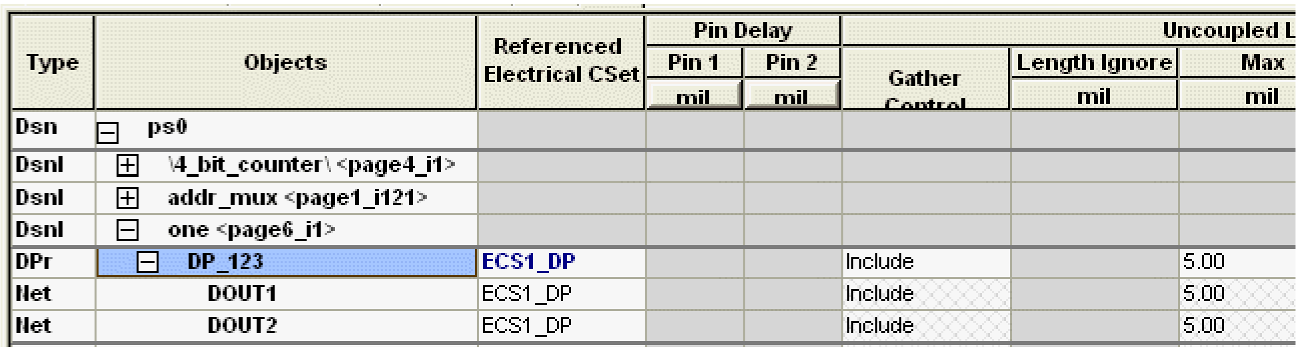
Figure 4-3

For more information on capturing constraints on a bus or a bit of a bus in Constraint Manager, see the *Allegro Constraint Manager User Guide*.

Applying an ECSet on a Differential Pair

You specify differential pair constraints in the Differential Pair worksheet of the Routing workbook. Differential pair constraints present in the *Differential Pair* worksheet can be applied to only differential pairs and not to individual member nets of a differential pair. For details on the various constraints that can be captured on differential pairs, refer to the *Allegro Constraint Architecture Guide*.

Note: All other constraints present in the Constraint Manager spreadsheet can be applied to differential pairs as well as to members of a differential pair.



Type	Objects	Referenced Electrical CSet	Pin Delay		Uncoupled L		
			Pin 1	Pin 2	Gather Control	Length Ignore	Max
			mil	mil		mil	mil
Dsn	⊟ ps0						
Dsnl	⊞ \4_bit_counter\ <page4_i1>						
Dsnl	⊞ addr_mux <page1_i121>						
Dsnl	⊟ one <page6_i1>						
DPr	⊟ DP_123	ECS1_DP			Include		5.00
Net	DOUT1	ECS1_DP			Include		5.00
Net	DOUT2	ECS1_DP			Include		5.00

Figure 4-4

The constraints added on differential pairs in Constraint Manager are saved only in the Constraint Manager database.

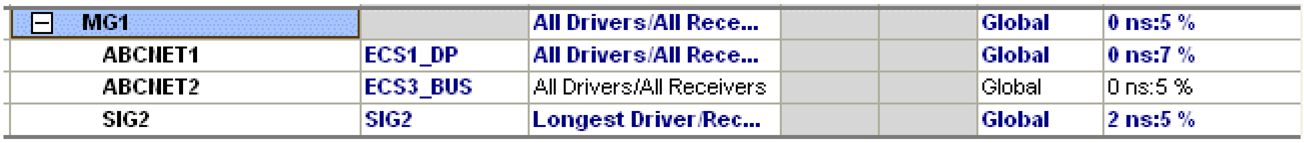
Applying an ECSet to Members of a Match group

You can assign ECSets to members of a match group, but not to a match group.The constraints you assign on a net or XNet, which is part of a match group, will be reflected on the net or XNet across match groups.

To assign an ECSet to a net or XNet:

1. Right-click the net or XNet and choose *Electrical CSet References* from the pop-up menu.
2. In the Electrical CSet References dialog box, choose the ECSet you want to assign.
3. Click *OK*.

The ECSet is assigned to the selected net/XNet and it is reflected in the match group to which the XNet belongs. Each member of the match group can have a different ECSet assigned to it.



⊟ MG1		All Drivers/All Rece...			Global	0 ns:5 %
ABCNET1	ECS1_DP	All Drivers/All Rece...			Global	0 ns:7 %
ABCNET2	ECS3_BUS	All Drivers/All Receivers			Global	0 ns:5 %
SIG2	SIG2	Longest Driver/Rec...			Global	2 ns:5 %

Figure 4-5

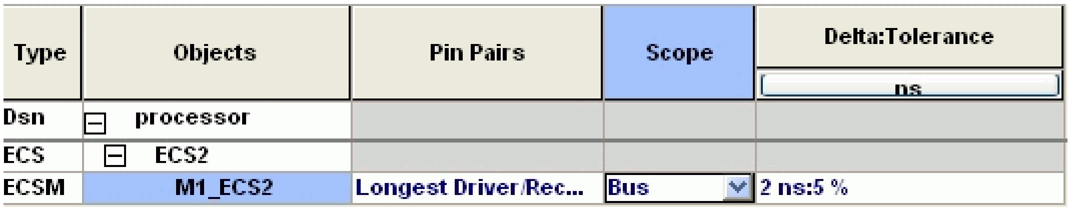
For more information on match groups, refer to the chapter Working with Objects of *Allegro Constraint Manager User Guide*.

Creating a Match Group based on an ECSet in Constraint Manager

When you create an ECSet of the Relative Propagation Delay constraints, you cannot specify constraints value at the time of creating the ECSet rule. To assign specific constraint values, such as pin pairs, scope, and Delta:Tolerance to the ECSet rule, you need to create a match group based on the ECSet. Such match groups are called ECSet match groups.

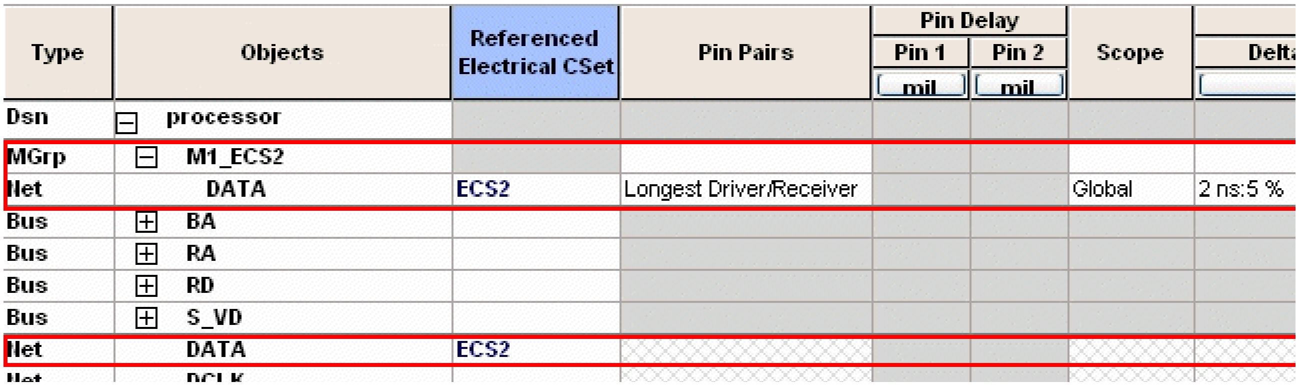
1. In the Electrical Constraint Set workbook, select the *Relative Propagation Delay* worksheet.
2. Choose *Objects – Create – Electrical CSet*.
3. Specify a name for the ECSet. For example, *ECS2*.
4. Deselect the *Copy Constraints From* check box.
5. In the Relative Propagation Delay worksheet, right-click the newly created ECSet, *ECS2*.
6. Choose *Create – Match Group* from the pop-up menu.
7. Specify a name for the match group in the Create Electrical CSet Match Group dialog box. For example, *M1_ECS2*.

   You can now specify specific values for the Relative Propagation Delay constraints in the match group, *M1_ECS2*.



Type	Objects	Pin Pairs	Scope	Delta:Tolerance
				ns
Dsn	⊟ processor			
ECS	⊟ ECS2			
ECSM	M1_ECS2	Longest Driver/Rec...	Bus	2 ns:5 %

When you assign the ECSet, *ECS2*, to a net, for example *DATA*, it will be automatically added to the ECSet match group *M1_ECS2* and inherit the constraint values from the match group.



Type	Objects	Referenced Electrical CSet	Pin Pairs	Pin Delay		Scope	Delt
				Pin 1	Pin 2		
				mil	mil		
Dsn	⊟ processor						
MGrp	⊟ M1_ECS2						
Net	DATA	ECS2	Longest Driver/Receiver			Global	2 ns:5 %
Bus	⊞ BA						
Bus	⊞ RA						
Bus	⊞ RD						
Bus	⊞ S_VD						
Net	DATA	ECS2					
Net	DCLK						

Note: You can also create an ECSet generated match group in SigXplorer. For more information, see *Creating ECSet Generated Match Groups in SigXplorer*.

Working with ECSet Tags

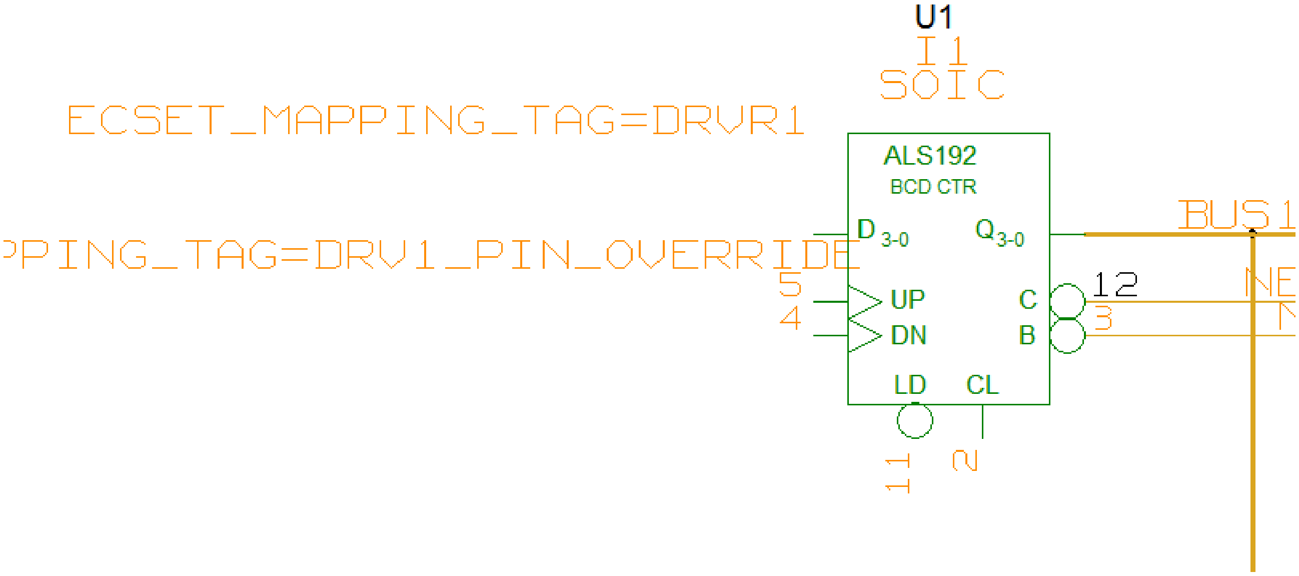
When applying an ECSet to target nets, the pins in the ECSet are mapped to the design component pins of those nets. This is done based on various factors such as signal_model assignments, pinuse, and reference designators (RefDes) to make a match.

When the mapping between ECSet pins and design component pins is forced to rely solely on RefDes information, and there is no one-to-one correlation between the RefDes information in the ECSet and the design, the mapping may be ambiguous.

To address this, you can create ECSet tags. These are user-defined properties that ECSet nodes support, and which can be used to uniquely identify a pin, thus resolving any mapping ambiguity. These tags can be used to lock the mapping between the ECSet and associated nets; the mapping will not be impacted by placement or RefDes changes.

To define an ECSet tag in the schematic, use the ECSET_MAPPING_TAG property. You can apply the ECSET_MAPPING_TAG property to components or to individual pins in a design.

When you apply the tag to a component, all the pins in the component inherit the tag. You can choose to specify different ECSET_MAPPING_TAG property values for individual pins in a component. In this case, the property values applied to the pin take precedence over the value defined for the component. For example, in the following image, ECSET_MAPPING_TAG value DRVR1 is applied to U1. The tag value DRV1_PIN_OVERRIDE is applied to pin 5 of U1 and takes precedence over the component tag value.

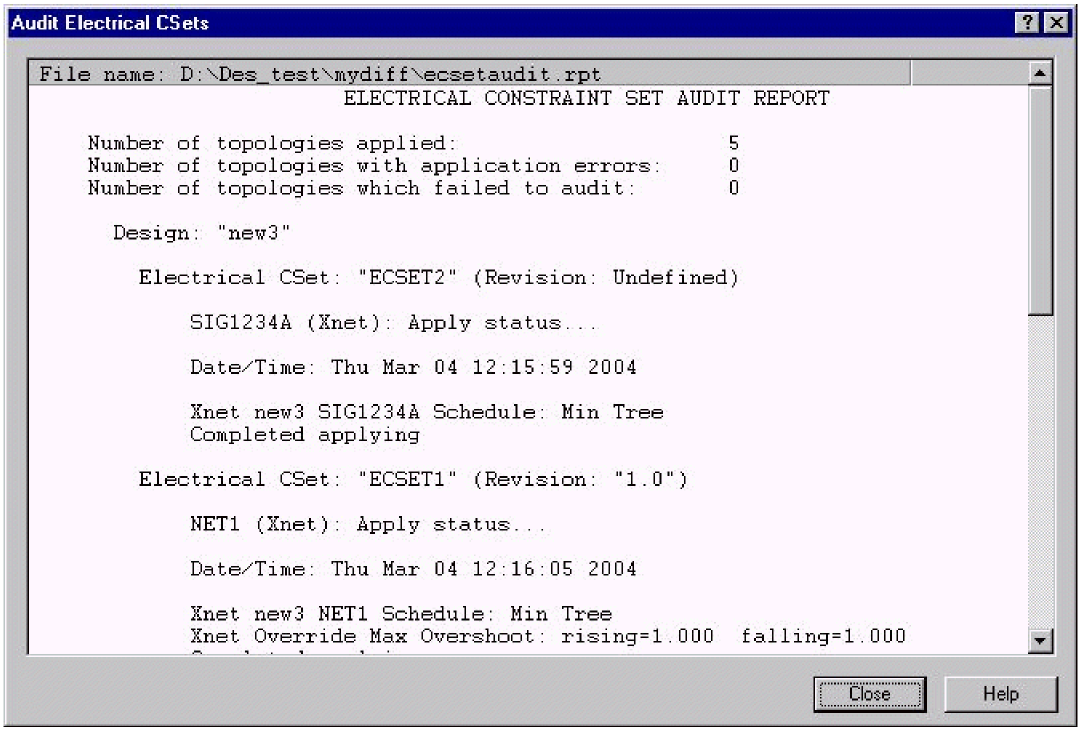


For more information, see *Allegro Constraint Manager User Guide*.

Auditing Electrical Constraint Set

When a topology mismatch occurs between a net/XNet and its ECSet, the Referenced Electrical CSet cell for the net/XNet is colored red. In addition, the status bar provides an indication of why the mismatch occurred. However, if you close Constraint Manager, the color coding is lost on subsequent invocation of Constraint Manager. Constraint Manager provides you with the option to audit all the ECSets and regenerate the color coding along with a report with detailed information on the ECSet application or violations.

➤ To generate the report, choose *Audit – Electrical CSet*.



```
File name: D:\Des_test\mydiff\ecsetaudit.rpt
                      ELECTRICAL CONSTRAINT SET AUDIT REPORT

    Number of topologies applied:                   5
    Number of topologies with application errors:   0
    Number of topologies which failed to audit:     0

      Design: "new3"

        Electrical CSet: "ECSET2" (Revision: Undefined)

            SIG1234A (Xnet): Apply status...

            Date/Time: Thu Mar 04 12:15:59 2004

            Xnet new3 SIG1234A Schedule: Min Tree
            Completed applying

        Electrical CSet: "ECSET1" (Revision: "1.0")

            NET1 (Xnet): Apply status...

            Date/Time: Thu Mar 04 12:16:05 2004

            Xnet new3 NET1 Schedule: Min Tree
            Xnet Override Max Overshoot: rising=1.000  falling=1.000
```

You can view the latest validation of the ECSet. As ECSets are stored in the `.dcf` file, topology can be reapplied to all the nets to which the ECSet is assigned.

Working with Electrical Constraints

This section covers the following topics:

- Adding Electrical Constraints on page 106
- Modifying Electrical Constraints on page 108
- Deleting Electrical Constraints on page 108
- Renaming Nets on page 109

Adding Electrical Constraints

In addition to creating an ECSet and then assigning it to objects, you can add specific electrical constraints to individual objects in the Constraint Manager spreadsheet.

To add constraints to an object in Constraint Manager, do the following:

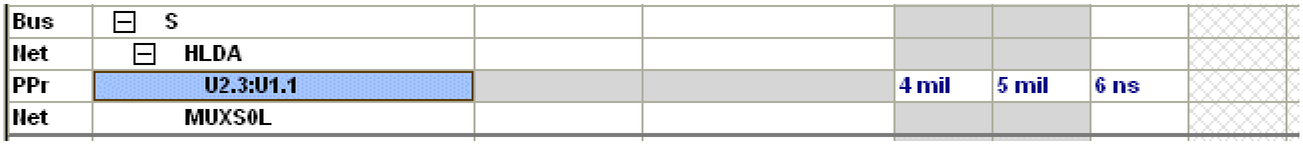
1. Start Constraint Manager from the design capture tool.
2. Click an appropriate cell next to the object to which you want to add the constraint. Add the requisite value for the cell. For example, to specify the Min/Max Propagation Delay constraints under the Routing worksheet, enter the requisite values in the *Min Prop Delay* and *Max Prop Delay* columns for a specific net or XNet.
3. Choose *File – Save*.

Capturing Constraints on Pin-Pairs

A pin-pair represents a pair of logically connected pins that form a driver-receiver connection. Pin pairs may not be directly connected but they must exist on the same net. You use pin pairs to capture specific pin-to-pin constraints for a net.

1. Open the *Min/Max Propagation Delays* worksheet in Constraint Manager.
2. Right-click on a net and choose the *Create – Pin Pair* menu command from the pop-up menu.
3. Select the driver and receiver pins form the *First Pins* and *Second Pins* columns of the *Create Pin Pair of* <net_name> dialog box.
4. Click *OK*.

   The specified pin-pair is created.
5. Specify appropriate values in the *Pin Delay* and *Prop Delay* cells to specify the `PIN_DELAY and PROPAGATION_DELAY` constraints, respectively.



Bus	⊟ S						
Net	⊟ HLDA						
PPr	U2.3:U1.1			4 mil	5 mil	6 ns	
Net	MUXS0L						

Figure 4-6

Note: For more information on capturing constraints on pin-pairs in Constraint Manager, see the Working with Constraint Objects chapter of *Allegro Constraint Manager User Guide*.

Handling Changes in Pin-Pairs

After you have created pin-pairs and have captured certain constraints on them, packaging information such as the section, pin number, or the reference designator of the component(s) forming the pin-pair might change.

The reference designator of a component can change in the following cases:

- You change the `LOCATION` property assigned to the component during packaging or that you had set previously.
- You change packaging properties like `GROUP` or `ROOM` as a result of which the reference designator changes during packaging.

The pin number of a pin of a component can change in the following cases:

- Change the section of a component
- Swaps the pins of a component

In any of the above cases, the pin-pair information in Constraint Manager becomes outdated. Constraint Manager updates the database with the changed pin-pair when you save the constraints in the Constraint Manager window.

Modifying Electrical Constraints

If you have captured an electrical constraint in Constraint Manager or added an electrical constraint in Design Entry HDL or System Connectivity Manager, you can modify the constraint in Constraint Manager.

To modify constraints in Constraint Manager, do the following:

1. Launch Constraint Manager from your design capture tool.
2. Modify the constraint in Constraint Manager.
3. Choose *File – Save*.

For more information on modifying constraints in Constraint Manager, see the *Allegro Constraint Manager User Guide*.

Deleting Electrical Constraints

If you have captured an electrical constraint in Constraint Manager, you can delete the constraint in Constraint Manager.

To delete constraints in Constraint Manager, do the following:

1. Start Constraint Manager from the design capture tool.
2. Delete the constraint in Constraint Manager.

3. Choose *File – Save*.

For more information on deleting constraints in Constraint Manager, see the *Allegro Constraint Manager User Guide*.

Renaming Nets

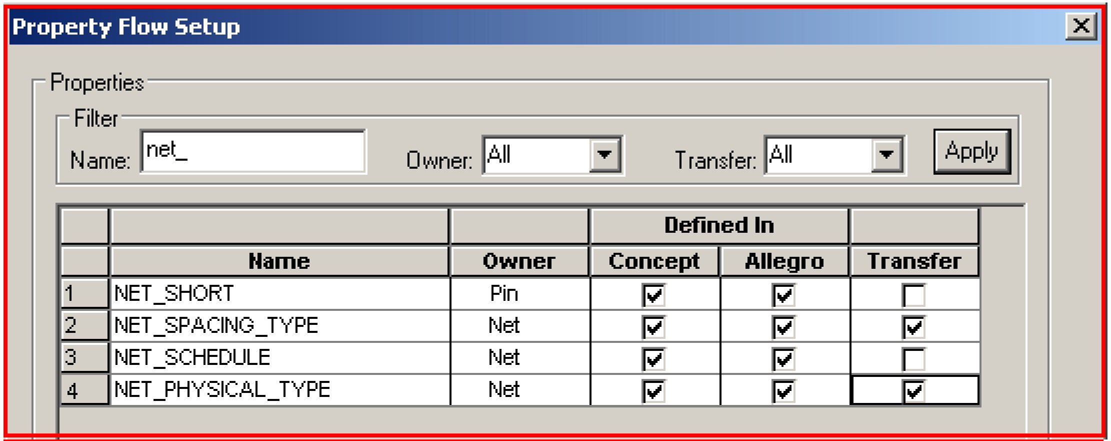
Preserving constraints on renaming nets is currently not supported in the Design Entry HDL – Constraint Manager Flow. Therefore, If you rename net names after synchronizing the schematic and board (maintain the PCB Editor-driven Constraint Manager traditional flow), you run the risk of losing ECSet assignments, differential pair assignments, and `NET_PHYSICAL` or `NET_SPACING` properties on new net names.

To ensure that there is no loss of connectivity, you need to take one of the following measures:

Option 1

1. Backannotate ECSets referenced to nets before changing net names (maintain the PCB Editor-driven Constraint Manager traditional flow)
2. In the schematic, add `$ELECTRICAL_CONSTRAINT_SET, $NET_SPACING_TYPE, and $NET_PHYSICAL_TYPE` placeholders to the nets that you want to rename.
3. Launch Import physical from *File – Import Physical*
4. Click the *Options* button.
5. Click the *Property Flow Setup* button.

6. Select the *Transfer* check box for `ELECTRICAL_CONSTRAINT_SET`, `NET_SPACING_TYPE`, `NET_PHYSICAL_TYPE` in the Property Flow Setup dialog box.



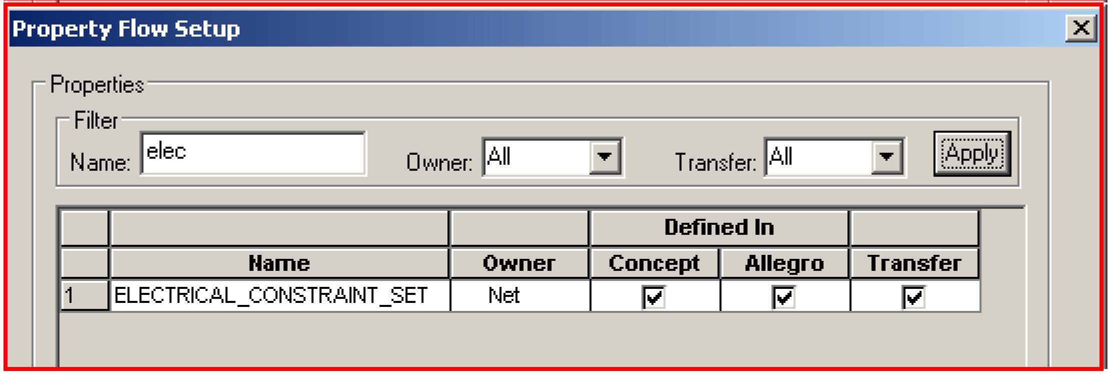
Property Flow Setup

Properties

Filter

Name: net_ Owner: All Transfer: All

			Defined In		
	Name	Owner	Concept	Allegro	Transfer
1	NET_SHORT	Pin	☑	☑	□
2	NET_SPACING_TYPE	Net	☑	☑	☑
3	NET_SCHEDULE	Net	☑	☑	□
4	NET_PHYSICAL_TYPE	Net	☑	☑	☑



Property Flow Setup

Properties

Filter

Name: elec Owner: All Transfer: All

			Defined In		
	Name	Owner	Concept	Allegro	Transfer
1	ELECTRICAL_CONSTRAINT_SET	Net	☑	☑	☑

7. Click *OK*.
8. Click the *From Layout* tab.
9. Select the *Net* option in the Annotate section and click *OK*.
10. Click *OK* to run Import Physical to annotate the above property values to the schematic.

    Note: There will be warnings about all the other nets that do not have placeholders-which can be ignored.

11. Review the schematic to ensure that property values have been assigned to the above property placeholders on nets to be renamed.

    You can now consider it safe to change the net names, and run Export Physical to create *.dat files.

12. Launch PCB Editor
13. Choose *File – Import – Logic*.

14. Click the *Import Cadence* button.

Note: If the property placeholders exist in the schematic, net names can be subsequently renamed and associated constraints will be retained.

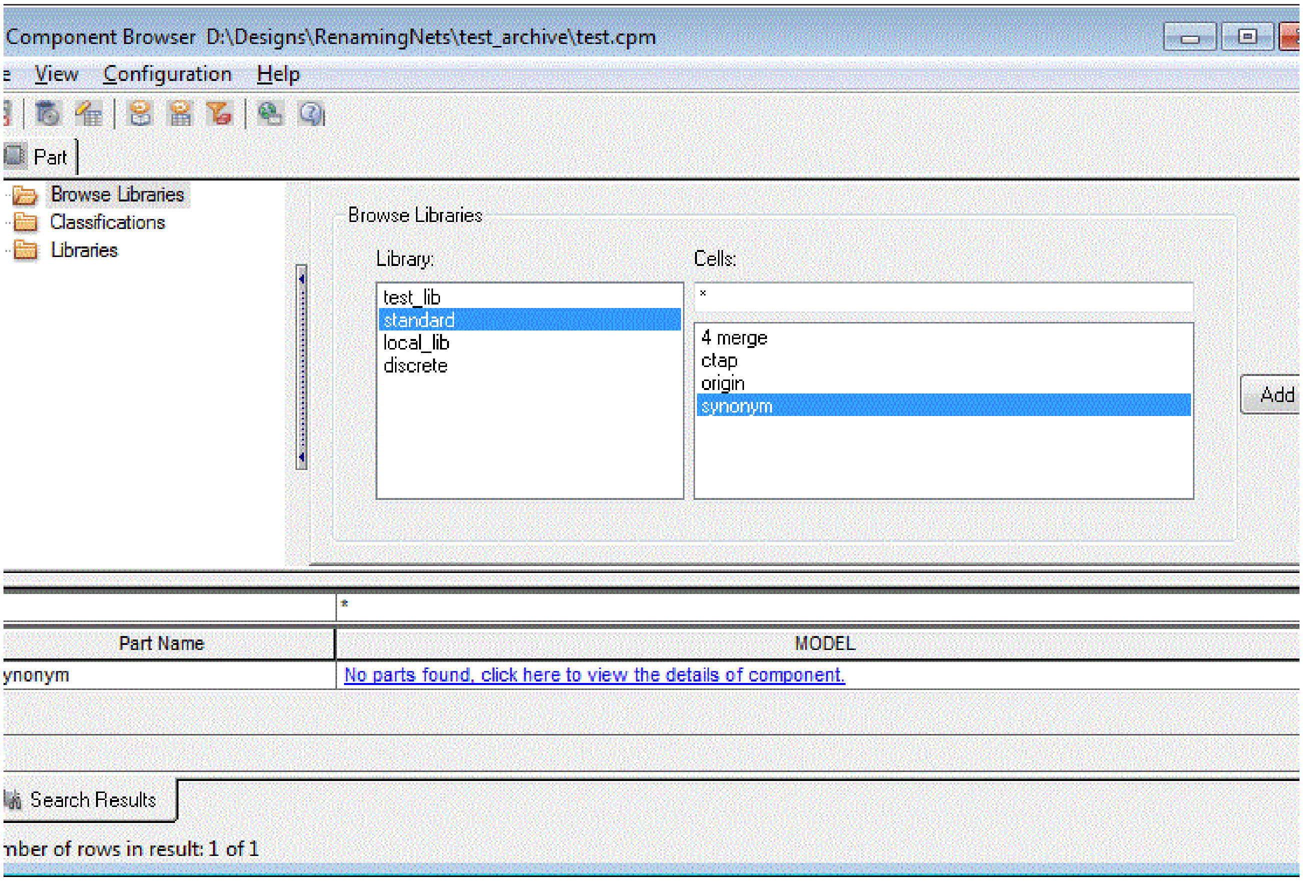
Option 2

If you want to rename scalar nets and preserve the constraints on the net, then it is recommended that you create a new scalar net and assign a name that needs to be assigned to the renaming net. Alias this scalar net using a synonym body, with the existing net that needs to be renamed. You can control the winning net name by using \BASE in the signal name, or by assigning the MAKE_BASE property on the aliased scalar net.

Follow the steps listed to rename the net.

1. Rename nets in Design Entry HDL.
2. Choose *Component – Add.*

The Component Browser appears.



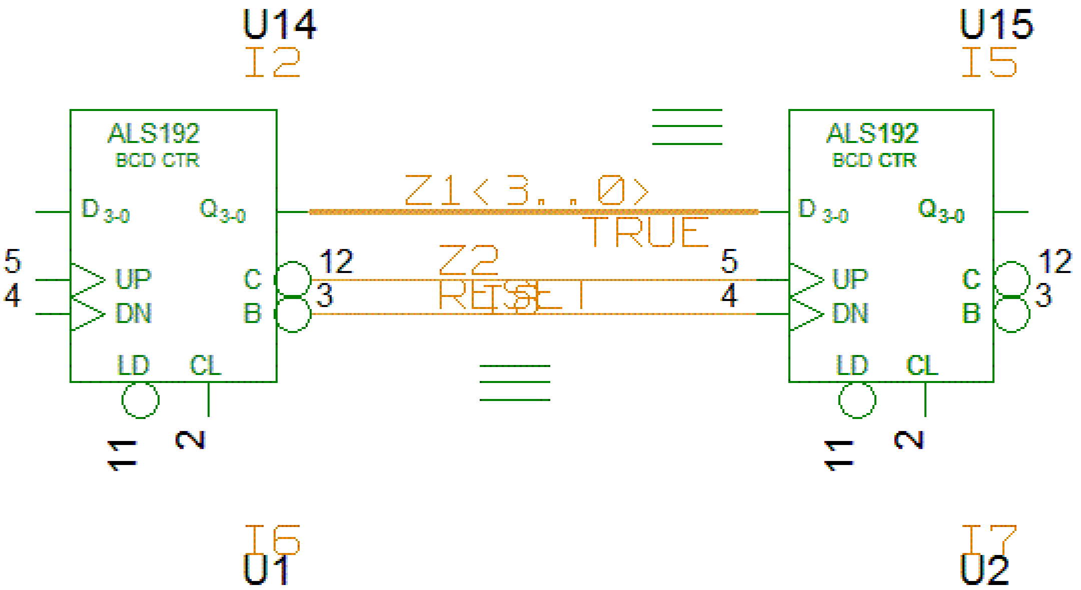
Part Name	MODEL
ynonym	No parts found, click here to view the details of component.

nber of rows in result: 1 of 1

3. In the *Library* box, select the *standard* library
4. In the scroll area, select *synonym*.
5. Click *Add.*

   The component is attached to the cursor.

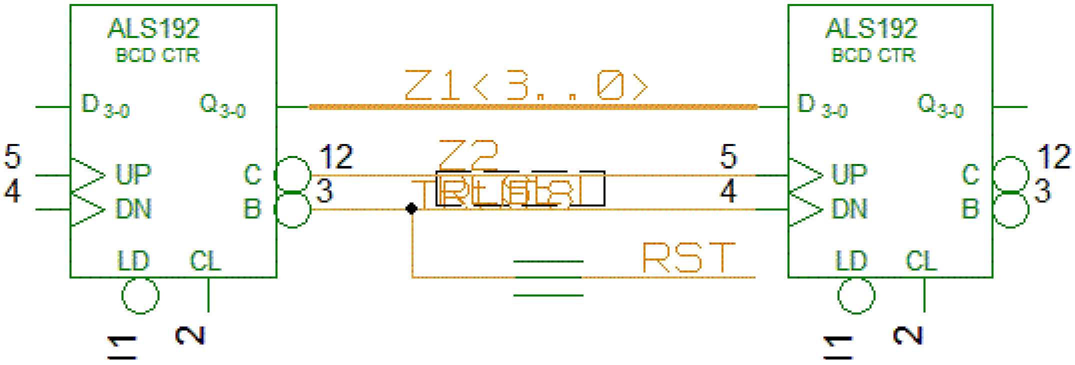
6. Place the component below the net that you want to rename.



For the purpose of this document assume that the net you want to rename is RESET.

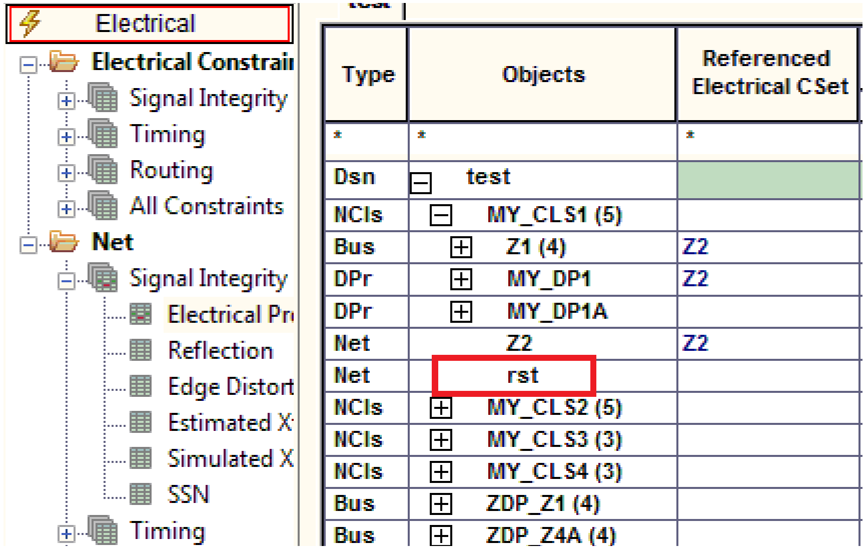
To rename the net, alias the old net to the new net that is added from the component browser.

7. Connect net RESET with the component that you just added.
8. Choose *Wire – Draw.*
9. Draw a wire from the net RESET to the component.
10. Choose *Wire – Signal Name.*
11. Type *RST* in the *Signal Name* box.



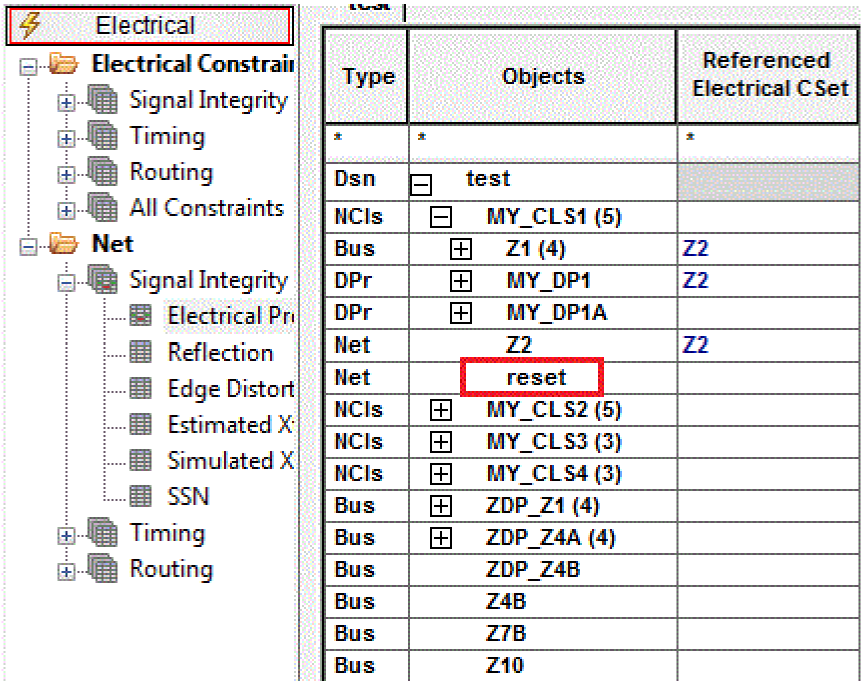
12. Click Ok.

Launch Constraint Manager. The renamed net is displayed as RST because it is lexicographically smaller. Observe that the renamed net is still a member of the net class and is being referenced by the ECSet. Also, all the constraints are preserved on the net after renaming.



Type	Objects	Referenced Electrical CSet
*	*	*
Dsn	⊟ test	
NCls	⊟ MY_CLS1 (5)	
Bus	⊞ Z1 (4)	Z2
DPr	⊞ MY_DP1	Z2
DPr	⊞ MY_DP1A	
Net	Z2	Z2
Net	rst	
NCls	⊞ MY_CLS2 (5)	
NCls	⊞ MY_CLS3 (3)	
NCls	⊞ MY_CLS4 (3)	
Bus	⊞ ZDP_Z1 (4)	
Bus	⊞ ZDP_Z4A (4)	

However if you want RESET to be the winning net, then add the *MAKE_BASE* property to it and save the design. Launch Constraint Manager and RESET is the winning value and is displayed as the net name.



Type	Objects	Referenced Electrical CSet
*	*	*
Dsn	⊟ test	
NCls	⊟ MY_CLS1 (5)	
Bus	⊞ Z1 (4)	Z2
DPr	⊞ MY_DP1	Z2
DPr	⊞ MY_DP1A	
Net	Z2	Z2
Net	reset	
NCls	⊞ MY_CLS2 (5)	
NCls	⊞ MY_CLS3 (3)	
NCls	⊞ MY_CLS4 (3)	
Bus	⊞ ZDP_Z1 (4)	
Bus	⊞ ZDP_Z4A (4)	
Bus	ZDP_Z4B	
Bus	Z4B	
Bus	Z7B	
Bus	Z10	

Note: If the old net is lexicographically smaller, add the *MAKE_BASE* property to the net and save the design. Launch Constraint Manager and the old net is the winning value and is displayed as the net name.

Note: Renaming nets with alias only works for scalar nets.

Option 3

1. Open the `.brd` file in PCB Editor and launch Constraint Manager.
2. Choose *File – Export Constraints*.

   This creates the `.dcf` file.
3. Perform the steps required to rename nets in the schematic.
4. Run Export Physical and import new netlist into the board.
5. Edit the `.dcf` file from step 2:
   1. Replace old net names with new.

      Note: This could probably be done with a script and list of old and new nets. The `pxl.chg` file provides a list of nets deleted or added.
6. Launch Constraint Manager connected to PCB Editor.
7. Choose *File – Import Constraints* to import the `.dcf` file from step 5.

   Note: When you use the Import Constraints option to reset constraint information in PCB Editor, perform the following steps:

   a. Import DCF (Overwrite mode),

   b. Close Constraint Manager (not PCB Editor),

   c. Open Constraint Manager.

   d. Import the `.dcf` file (Overwrite mode, Report-only).

      The report should be empty.

Resolving Constraints

When two (or more) nets are aliased together, they retain their logical names but they become connected and are merged into one physical net. If the properties on aliased nets do not match, it is considered a property conflict. To help you identify and fix conflicts, a checking trigger, a visual indicator of a mismatch and a resolution method is provided.

You can view conflicts and their value in a design, by using the *Audit – Alias Property Conflicts* option. The *Audit Constraints* option only shows the net level overrides and constraint inconsistencies. With this option only the constraint properties that are overridden are reported, their values are not reported.

Note: Conflicts are not reported on aliasing buses in DE-HDL and on pre-16.5 designs. Existing designs (pre-16.5 designs) are not upreved to re-merge interface nets to show conflicts for existing designs across hierarchy.

The conflict resolution feature reports conflicts in the following cases:

- Flat Designs
- Hierarchical Designs

Flat Designs

Conflict resolution shows constraints nets where different constraints were aliased using "Alias" or "Synonym" body in the DE-HDL canvas.

Hierarchical Designs

In a hierarchical design scenario where some nets having constraints defined in a lower level block, and the net is aliased or connected with an existing top level block that has different constraints, then no constraints are reported. However, in this scenario the constraints are considered resolved, because the top level constraints get precedence over the lower level constraints. So in this case, if you view *Audit – Alias Property Conflicts*, the constraint is shown as a resolved constraint and you can change the constraint here. You can also change the constraint by clicking the constraint cell. Clicking the constraint cell will display the Conflict Resolution window. You can change the constraint from this window.

Take another scenario, where the interface net that has some constraints defined in the lower level block, is connected to a top level net with no constraints, and the constraints are changed at the top level, then no conflicts are reported. Here, the constraints are modified knowingly after the nets are merged or connected. Conflicts are reported only when nets that have different constraints are connected and as a result the constraints are merged.

Now, consider a scenario where the interface nets with constraints (defined in the lower level), from block LOW1 are connected to interface nets (constraints defined in lower level) from Block LOW2, the conflicts are reported automatically. Here, the constraint values are shown in red indicating a conflict. Here, both the interface nets have inherited constraints in the top block and therefore when the nets are merged, conflicts are reported.

Note: Restore from definition does not work for interface objects.

5

Physical and Spacing Constraints

Physical and spacing constraints impact the physical layout of a PCB board or of a SiP, and are therefore, usually captured and modified during the layout design stage by layout engineers. However, stack-up information and some physical and spacing constraints are finalized during the design capture stage itself. For example, physical constraints, such as minimum line width and maximum line width, that have an impact on the manufacturing as well as the functioning of the design. Such constraints are specified at the design stage and can then be modified while creating the physical layout. Similarly, information such as layers to be used by signal groups and trace proximity is also decided at the design stage.

This chapter describes how Constraint Manager, launched from a design capture tool can be used to capture, edit, and view physical and spacing constraints.

The topics covered in this chapter are:

Note: To know about the physical and spacing constraints and corresponding CSets, see Physical Constraints, Spacing Constraints, and Physical and Spacing CSets.

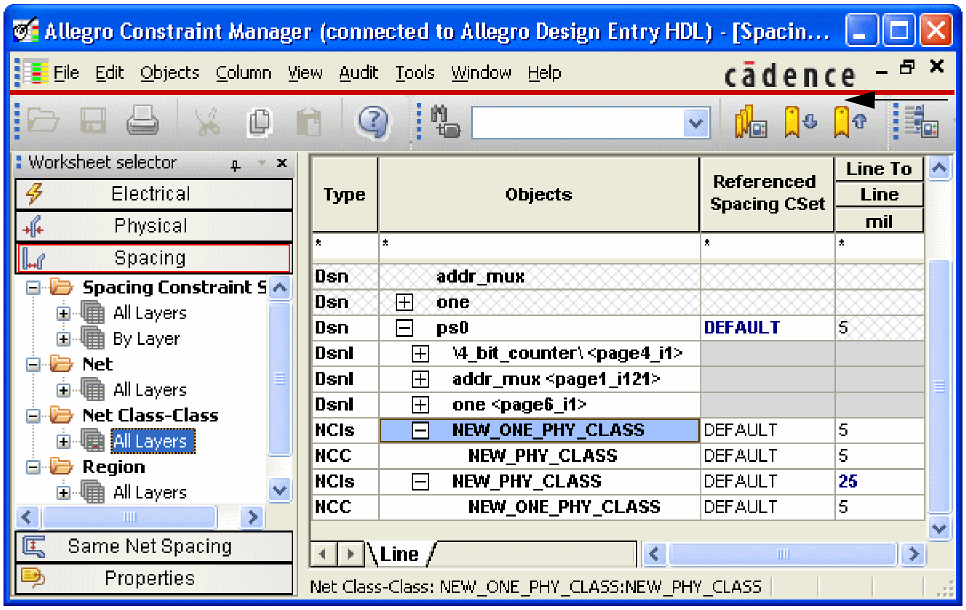
Working with Net Classes

A net class is created by grouping the objects with common features and similar constraints. Using classes allows you to quickly apply similar constraints on all the objects in the same class using CSets.

Using Net Classes

Depending on whether the net class is in the Electrical, Physical, or the Spacing domain, you can apply different CSets to each class. The CSets available depend on whether Constraint Manager is launched from a design capture tool or from a board design tool.

For example, when you launch Constraint Manager from a design capture tool, you can create new net classes in the Physical and Spacing domain. You can also modify the membership of these classes by adding or removing nets from these classes.

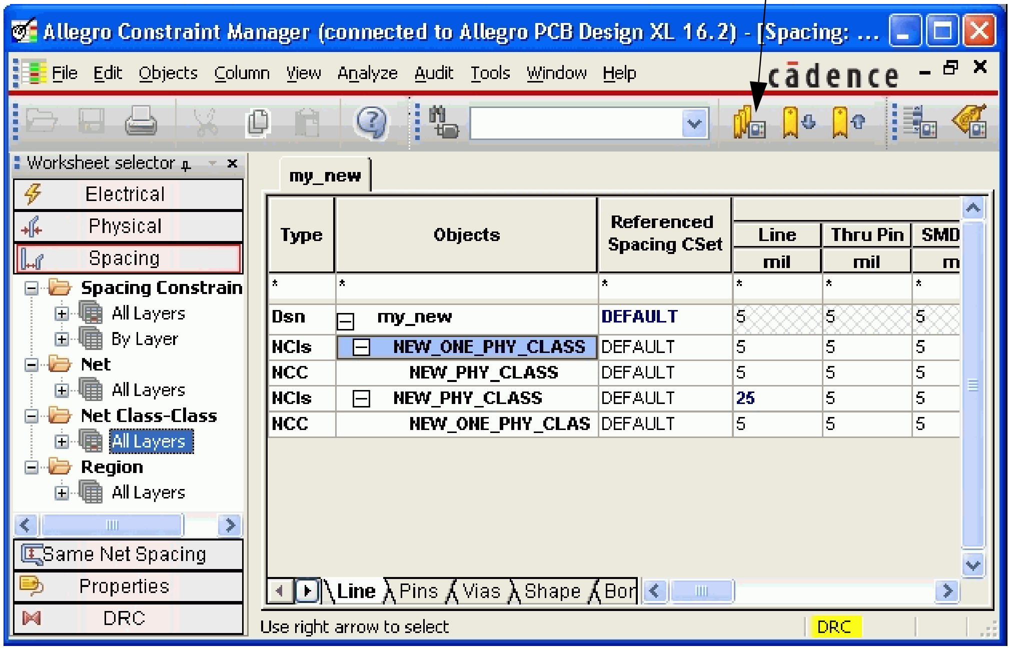


Type	Objects	Referenced Spacing CSet	Line To Line mil
*	*	*	*
Dsn	addr_mux		
Dsn	one		
Dsn	ps0	DEFAULT	5
Dsnl	\4_bit_counter\<page4_i1>		
Dsnl	addr_mux <page1_i121>		
Dsnl	one <page6_i1>		
NCls	NEW_ONE_PHY_CLASS	DEFAULT	5
NCC	NEW_PHY_CLASS	DEFAULT	5
NCls	NEW_PHY_CLASS	DEFAULT	25
NCC	NEW_ONE_PHY_CLASS	DEFAULT	5

Net Class-Class: NEW_ONE_PHY_CLASS:NEW_PHY_CLASS

Constraint Manager Interface when launchec from Design Entry HDL

Constraint Manager Interface when launched from



Type	Objects	Referenced Spacing CSet	Line mil	Thru Pin mil	SMD m
*	*	*	*	*	*
Dsn	my_new	DEFAULT	5	5	5
NCls	NEW_ONE_PHY_CLASS	DEFAULT	5	5	5
NCC	NEW_PHY_CLASS	DEFAULT	5	5	5
NCls	NEW_PHY_CLASS	DEFAULT	25	5	5
NCC	NEW_ONE_PHY_CLAS	DEFAULT	5	5	5

Use right arrow to select DRC

The net classes supported when you launch Constraint Manager from a design capture tool, are:

- Electrical Net Classes

- Physical and Spacing Net Classes

To know about the operations that can be performed on these net classes, see the *Constraint Manager Reference*. To know more about the new constraint objects, see *Allegro Constraint Manager User Guide.*

Electrical Net Classes

These are the net classes created for electrical constraints. Net classes created for electrical constraints are considered as local classes and support a logical name, which is then used to instantiate a unique net class in the higher-level design.

Physical and Spacing Net Classes

The physical and spacing net classes are considered global classes. There is only one instance of these classes in the design. Physical and Spacing net classes use physical names for processing lower-level constraints. By using these classes, you can now define physical constraints in hierarchical designs as well.

When Constraint Manager is invoked from a design capture tool, you can only add or modify the membership of the physical and spacing net classes. Applying constraints to these classes is supported only if physical and spacing constraints are enabled for the design. See Enabling Physical and Spacing Constraints.

If capturing physical and spacing constraints in not enabled, invoke Constraint Manager from Allegro PCB Editor and then capture physical and spacing constraints.

Creating Net Classes

To create a new net class in Constraint Manager, complete the following steps.

1. Select a workbook in the domain for which net class is to be created.
2. Choose *Object – Create – Net Class*.
3. In the *Create Net Class* dialog box, specify the name of the net class.
4. Click *OK*.

For more details, see *Constraint Manager Reference*.

Editing Net Class Membership

1. In the worksheet, right-click on the net class.
2. From the pop-up menu, choose *Membership – Net Class*.
3. In the dialog box, add or remove objects from the selected net class.

Capturing Physical and Spacing Constraints

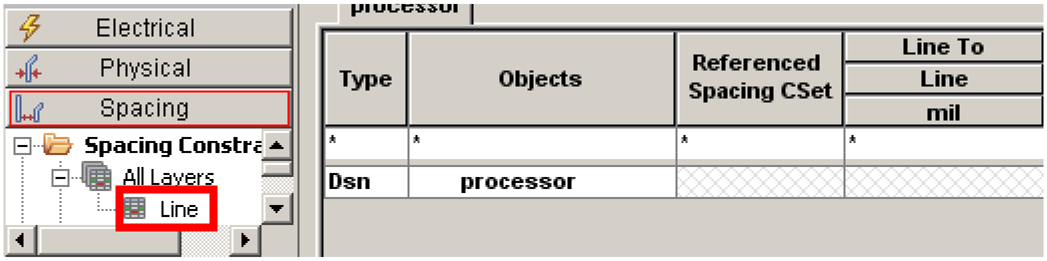
Now, Constraint Manager launched from logic design tools can be used to view, capture, or edit the physical and spacing constraints. By default, only those constraints that impact the functioning of design are displayed in Constraint Manager launched from the design capture tool.

The Physical constraints that are visible by default are:

- Minimum and maximum line width
- Minimum Width and the maximum length of the Neck
- Primary gaps and neck gaps for differential pairs
- Allow Etch and Ts

The Spacing constraints that are visible by default are:

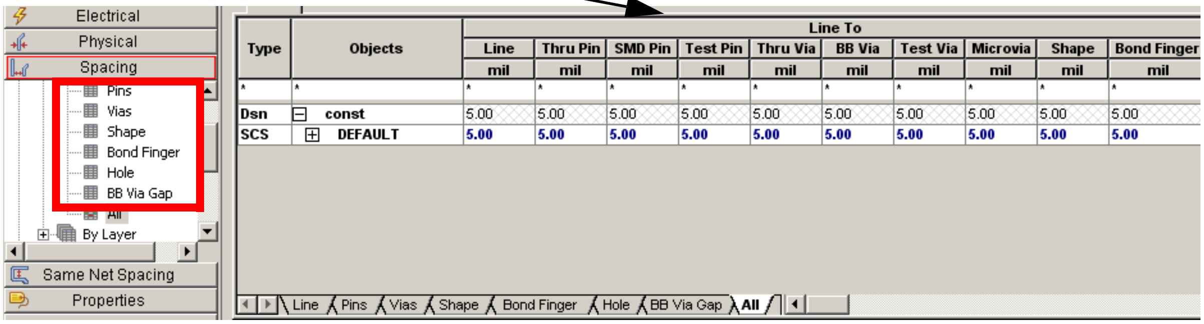
- Line-to-line spacing



Type	Objects	Referenced Spacing CSet	Line To Line mil
*	*	*	*
Dsn	processor		

Default Spacing Constraints in Constraint Manager launched from design capture tools

Spacing Constraints in Constraint Manager launched from physical layout tool



Type	Objects	Line To									
		Line	Thru Pin	SMD Pin	Test Pin	Thru Via	BB Via	Test Via	Microvia	Shape	Bond Finger
		mil	mil	mil	mil	mil	mil	mil	mil	mil	mil
*	*	*	*	*	*	*	*	*	*	*	*
Dsn	⊟ const	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00
SCS	⊞ DEFAULT	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00	5.00

Modes for Physical and Spacing Constraints

When Constraint Manager is launched from the design capture tools, physical and spacing constraints are available in two modes; Read-only mode and Edit mode.

Important

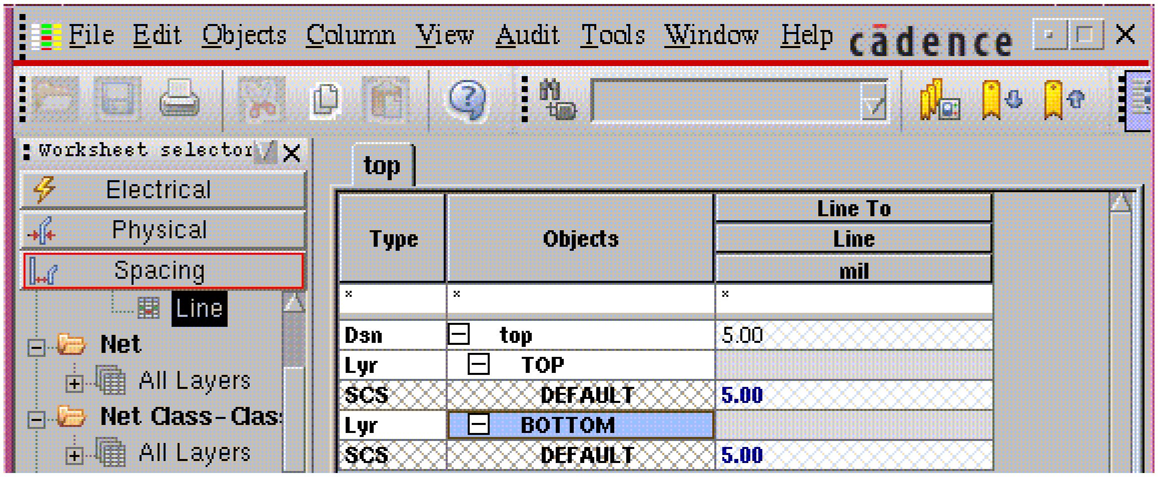
> The read-only and edit modes discussed in this section are valid only for the physical and spacing constraints. These are not applicable to the electrical constraints displayed in Constraint Manager.

Read-Only Mode

In the read-only mode, you can only view the physical and spacing constraints in the Constraint Manager. In this mode, edit and delete operations are not supported for physical and spacing constraints.

If you run the import physical command in the read-only mode, modifications made to the constraints during the physical design stage are visible in Constraint Manager launched from

DE-HDL but cannot be modified. Similarly, on running the export physical command, the physical and spacing constraints are not transferred to the physical layout of the design.

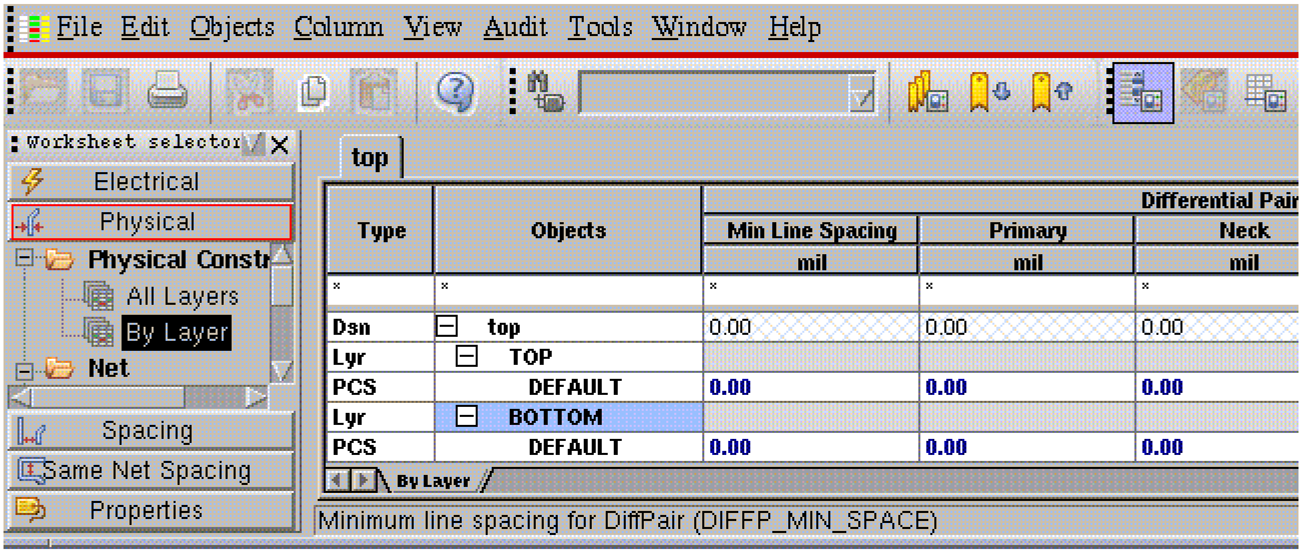


Type	Objects	Line To Line mil
*	*	*
Dsn	⊟ top	5.00
Lyr	⊟ TOP	
SCS	DEFAULT	5.00
Lyr	⊟ BOTTOM	
SCS	DEFAULT	5.00

Edit Mode

In this mode, you can view the constraints as well as modify them in Constraint Manager.

As you implement design logic in the design capture tool, you can also capture or modify physical and spacing constraints in Constraint Manager. You can add new physical and spacing constraints and also delete and modify the existing constraints. In the Edit mode, you can apply Constraint Sets (CSets) on the objects of physical and spacing worksheets. These constraints can then be passed on to the layout engineer.



Type	Objects	Differential Pair		
		Min Line Spacing	Primary	Neck
		mil	mil	mil
*	*	*	*	*
Dsn	⊟ top	0.00	0.00	0.00
Lyr	⊟ TOP			
PCS	DEFAULT	0.00	0.00	0.00
Lyr	⊟ BOTTOM			
PCS	DEFAULT	0.00	0.00	0.00

Minimum line spacing for DiffPair (DIFFP_MIN_SPACE)

Modifying physical and spacing constraints specified during the layout design stage is also supported.

Important

> Layer information cannot be modified. On running Export Physical command to update an existing board, only constraint data is modified. Layers added or deleted during the design capture process are ignored.

Table 5-1 on page 126 lists the possible ways in which Constraint Manager can be launched from a design capture tool and it's impact of information flow.

Table 5-1 Constraint Manager Launched From Design Capture Tool

Capturing Physical & Spacing Constraints...	Constraint Manager Mode	Flow of Physical & Spacing Constraints
Enabled	Read-Only	Constraint information updated in the back-to-front flow.
Enabled	Edit	Constraint information shared in the front-to-back flow as well as in the back-to-front flow.

Enabling Physical and Spacing Constraints

This section describes the steps to be performed to enable capturing or modifying of physical and spacing constraints using Constraint Manager launched from the design capture tool.

Though, you can create physical and spacing net classes and also define their membership, but actual capturing or modifying of physical and spacing constraints using Constraint Manager launched from a design capture is enabled if the following two conditions are satisfied.

- In the *project*.cpm file, the value of the EDIT_PHYSICAL_SPACING_CONSTRAINTS directive must be set to ON.

  The .cpm file must have the following lines.

  ```
  START_CONSTRAINT_MGR
  EDIT_PHYSICAL_SPACING_CONSTRAINTS 'ON'
  END_CONSTRAINT_MGR
  ```

  The stackup data for the board file is available in Design Entry HDL. The layer definition or the stackup data has an impact on the physical and spacing constraints. Therefore, these constraints are available in Constraint Manager only if the layer information is available in the design.

  To see how layer information can be made available in design capture tools, see Importing Stackup Data.

Important

In designs created in the 16.2 release, physical and spacing constraints are enabled by default. For these designs, the default value of the EDIT_PHYSICAL_SPACING_CONSTRAINTS directive is ON. However, if you create a new design in Design Entry HDL and launch Constraint Manager, physical and spacing constraints are not visible until layer information is imported into the design.

Importing Stackup Data

Capturing and editing of physical and spacing constraints is enabled only if the layer or the stackup information is available during the design capture stage. This information can be made available using one of the following methods.

- Importing Physical Data
- Importing Technology File

Importing Physical Data

This method is recommended for designs for which the physical layout has already been created and the logical and physical designs are in sync. In case of such designs, it is recommended that you run Import Physical and update the logical design with changes in the physical design.

Along with layer information, Physical and Spacing CSets are also imported. The information imported in the logical design can be categorized as follows.

- Layer Stackup Changes
- Physical CSet Differences
- Spacing CSet Differences
- Same Net Spacing CSet Differences
- Physical CSet Association Differences
- Spacing CSet Association Differences
- Same Net Spacing CSet Association Differences

Importing Technology File

For new designs, layer information can be included ny importing the technology file that is to be used for creating the physical design.

1. Launch Constraint Manager.
2. Choose *File – Import Technology File*.
3. Specify the technology file to be imported.
4. Select the mode in which the technology file is to be imported.

Tip

If you want to view the modification that will be made before performing the actual import, select the *Report Only* check box.

5. Click *Open*.

Note: 16.3 onwards, SigXplorer is enhanced to import and create multi-layer stacks so that topology can be simulated with traces defined on different layers based on a layer set, Therefore, you can now define the layer stackup from SigXplorer and export it as a technology file. You need not invoke PCB Editor to get the technology file with the layer stackup.

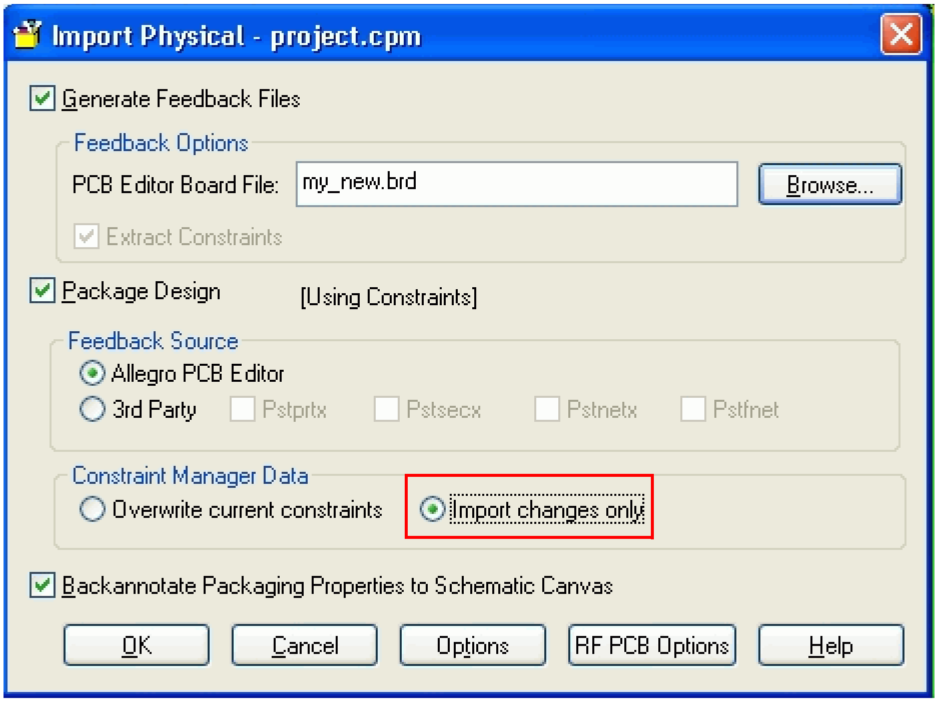
See *Allegro SI SigXplorer Reference* for more information.

Capturing Constraints in Pre-16.2 Designs

To enable physical and spacing constraints in designs that were created using old SPB releases, perform the following sequence of tasks.

1. Open the `.cpm` file in a text editor, add the `EDIT_PHYSICAL_SPACING_CONSTRAINTS` directive, set its value to ON, save and close the file.
2. Open the design in the design capture tool.
3. Use the import physical command to import layer information.

   Note: Ensure that the *Import changes only* option is selected under Constraint Manager Data when you run Import Physical.



Import Physical - project.cpm

☑ Generate Feedback Files

Feedback Options

PCB Editor Board File: my_new.brd

☑ Extract Constraints

☑ Package Design [Using Constraints]

Feedback Source

● Allegro PCB Editor

○ 3rd Party □ Pstprtx □ Pstsecx □ Pstnetx □ Pstfnet

Constraint Manager Data

○ Overwrite current constraints ● Import changes only

☑ Backannotate Packaging Properties to Schematic Canvas

The design is updated with the layer stackup data, PCSets and ECSets

Editing Physical and Spacing Constraints

If the conditions listed in the Enabling Physical and Spacing Constraints section are satisfied, the physical and spacing constraints are visible in Constraint Manager launched from the design capture tool. These constraints can also be edited.

If you now run the Export Physical command to synchronize the logical and the physical design, all the physical and spacing constraints captured or modified in the design capture tool will be available in the Constraint Manager launched from the physical layout tool.

Important

> Layer information cannot be modified. On running Export Physical command to update an existing board, only constraint data is modified. Layers added or deleted during the design capture process are ignored.

Table 5-2 Design Tasks performed in the 16.2 release

Design Task	Design State	P&S Constraints in Constraint Manager
Create a design	■ `EDIT_PHYSICAL_SPACING_CONSTRAINTS` set to `ON` ■ Layer information missing	■ Not visible
Import Technology File	■ `EDIT_PHYSICAL_SPACING_CONSTRAINTS` set to `ON` ■ Layer information available	■ Visible ■ Editable
Change the edit-mode to view-only mode	■ `EDIT_PHYSICAL_SPACING_CONSTRAINTS` set to OFF ■ Layer information available	■ Visible ■ Not Editable

Editing Constraints in a Hierarchical Design

Which capturing physical and spacing constraints for an hierarchical design, take care of the following:

- It is recommended that the layer stackup of the lower-level blocks should be same as the layer stackup for the root design.

- In case the layer stackup for lower-level blocks is different from the layer stackup for root design, following rules are applied for merging physical and spacing constraints.
  - Name mapping is used to identify the layers.
  - Addition or deletion of layers is not supported.
  - If a layer is available for a lower-level block but is missing from the stackup of the root design, the constraint information in the layer is ignored.

Switching Modes for Physical and Spacing Constraints

For a design created in SPB 16.2, physical and spacing constraints are enabled by default, and are in the edit-mode. To switch Constraint Manager from edit-mode to read-only mode and vice-versa requires the *`project.cpm`* file to be edited manually.

Switching From Edit to Read-Only Mode

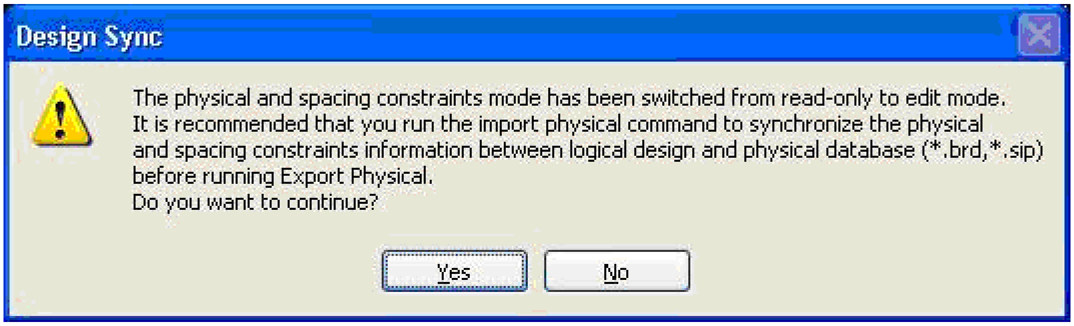
In Read-Only Mode, the physical and spacing constraints information in logical design is not shared with the physical design. Therefore, it is recommended that before you switch from edit mode to read-only mode, run the Export Physical command to ensure that the physical design is updated with the constraint modifications made in the logical design.

To switch from edit mode to read-only mode, edit the *`project.cpm`* file and set the value of the `EDIT_PHYSICAL_SPACING_CONSTRAINTS` to OFF.

Switching From Read-Only to Edit Mode

When you change from read-only mode to edit mode, a physical or a spacing constraint can be modified while making changes to the logical design as well as while modifying the physical design. Depending on whether you export the design changes or import the modifications made to the physical design, there are changes that a actual constraint value is overwritten by a stale value.

To prevent such scenarios, it is recommended that before you switch from the read-only mode to edit mode, use the *import physical* command in the changes only mode to update the logic design with the latest constraint values from the physical design. The message displayed when you run Export Physical command for the first time after switching modes, reiterates this recommendation.



The physical and spacing constraints mode has been switched from read-only to edit mode.
It is recommended that you run the import physical command to synchronize the physical and spacing constraints information between logical design and physical database (*.brd,*.sip) before running Export Physical.
Do you want to continue?

Caution

Changing modes of Constraint Manager launched from design capture tool requires editing of the .cpm file. Therefore, frequent switching of modes is not recommended. Instead, it is recommended that the `EDIT_PHYSICAL_SPACING_CONSTRAINTS` ***directive must be specified at the site level in site.cpm and directive locking should be used to prevent frequent changing of modes.***

6

ECSet in SigXplorer

Why SigXplorer?

While capturing design constraints in Constraint Manager, you can view the entire topology on a object in SigXplorer, which is the tool used to create, modify, simulate, and save prototypes of net topologies. A topology can be defined as a representation of how signals are physically and electrically connected. When you launch SigXplorer on an object in Constraint Manager, the driver-receiver relationship along with its connectivity details are created in SigXplorer. While capturing your designs, if you create a topology that meets the circuit specifications, you can extract the topology in SigXplorer, and save it as a topology template for reuse in future.

For detailed information on topology templates in Constraint Manager, see *ECSets and Topology Templates* in *Allegro Constraint Manager User Guide*.

For topology extraction and constraint modification, you can launch SigXplorer on a design object. You can also validate and generate electrical constraint sets (ECSets) in SigXplorer. These ECSets can then be applied on other XNets or net objects in Constraint Manager and validated. When Constraint Manager is launched from a design capture tool, you can launch SigXplorer on the following design objects.

- ❑ nets
- ❑ XNets
- ❑ pin pairs
- ❑ Buses (SigXplorer displays the topology on the first bit)
- ❑ classes
- ❑ differential pairs
- ❑ ECSets

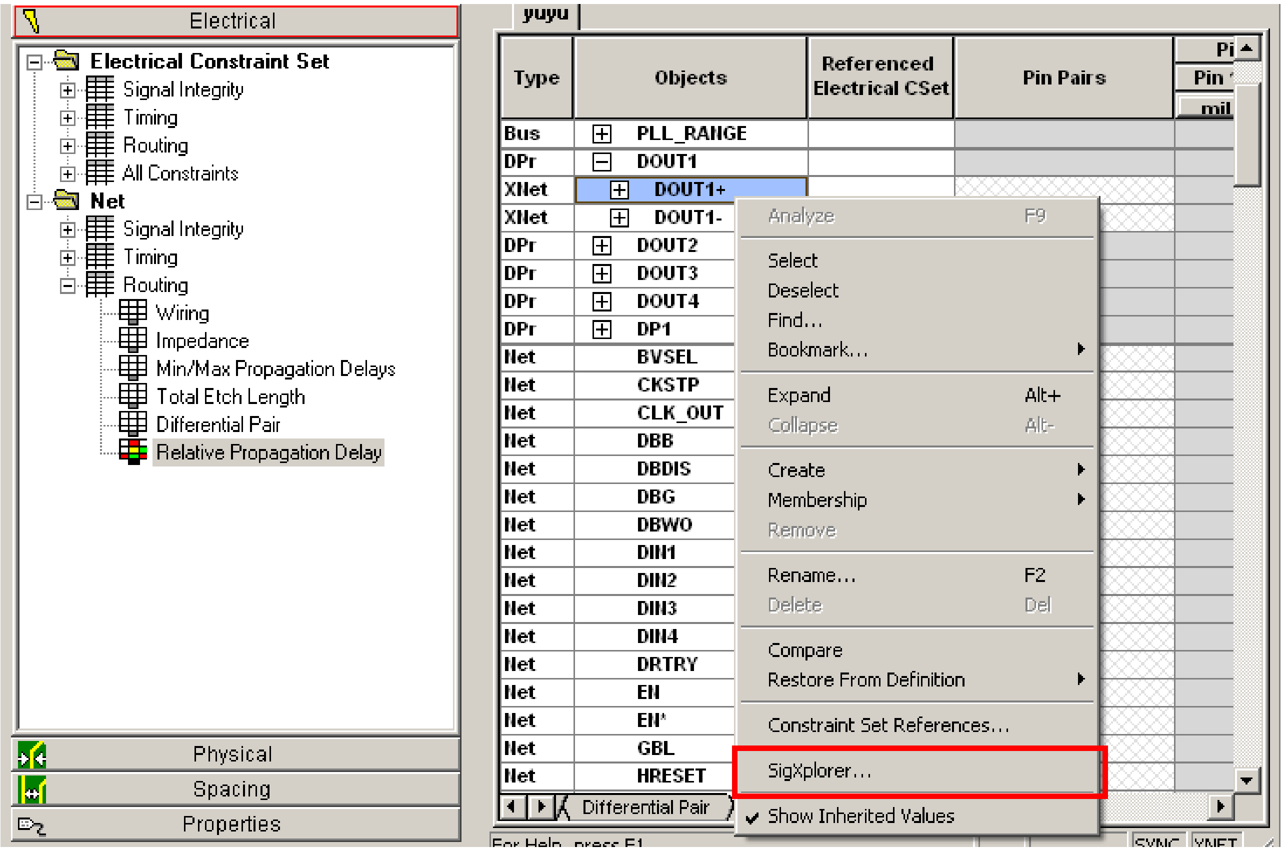
Important

To launch SigXplorer on model-defined differential pair objects, it is recommended that you use a SigXplorer license that is higher than the default license shipped with design capture tools. To change the product suite, set the value of the environment variable SIGXP_TIER to the product number for the high-end license. For example, `SIGXP_TIER = PA5630`

Viewing Topology in SigXplorer

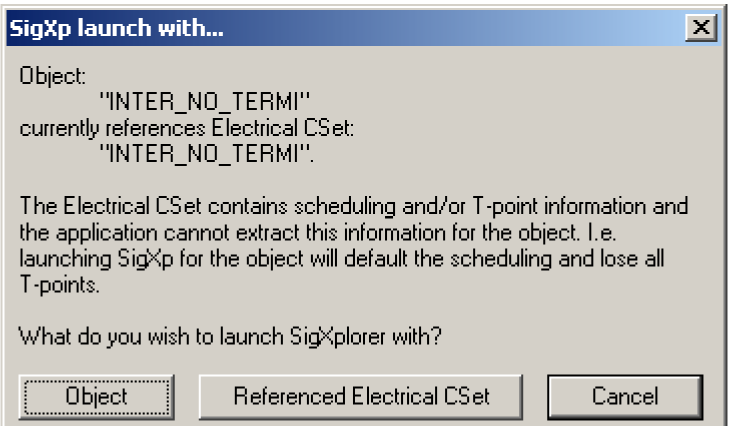
Constraint Manager provides support for viewing valid models in the SigXplorer canvas. To launch SigXplorer on a design object in Constraint Manager, do the following:

➤ Right-click on the design object and choose *SigXplorer* from the pop-up menu.



Type	Objects	Referenced Electrical CSet	Pin Pairs	Pi Pin mil
Bus	PLL_RANGE			
DPr	DOUT1			
XNet	DOUT1+			
XNet	DOUT1-			
DPr	DOUT2			
DPr	DOUT3			
DPr	DOUT4			
DPr	DP1			
Net	BVSEL			
Net	CKSTP			
Net	CLK_OUT			
Net	DBB			
Net	DBDIS			
Net	DBG			
Net	DBWO			
Net	DIN1			
Net	DIN2			
Net	DIN3			
Net	DIN4			
Net	DRTRY			
Net	EN			
Net	EN*			
Net	GBL			
Net	HRESET			

If you have an ECSet applied to the design object, the following dialog box appears.



SigXp launch with...

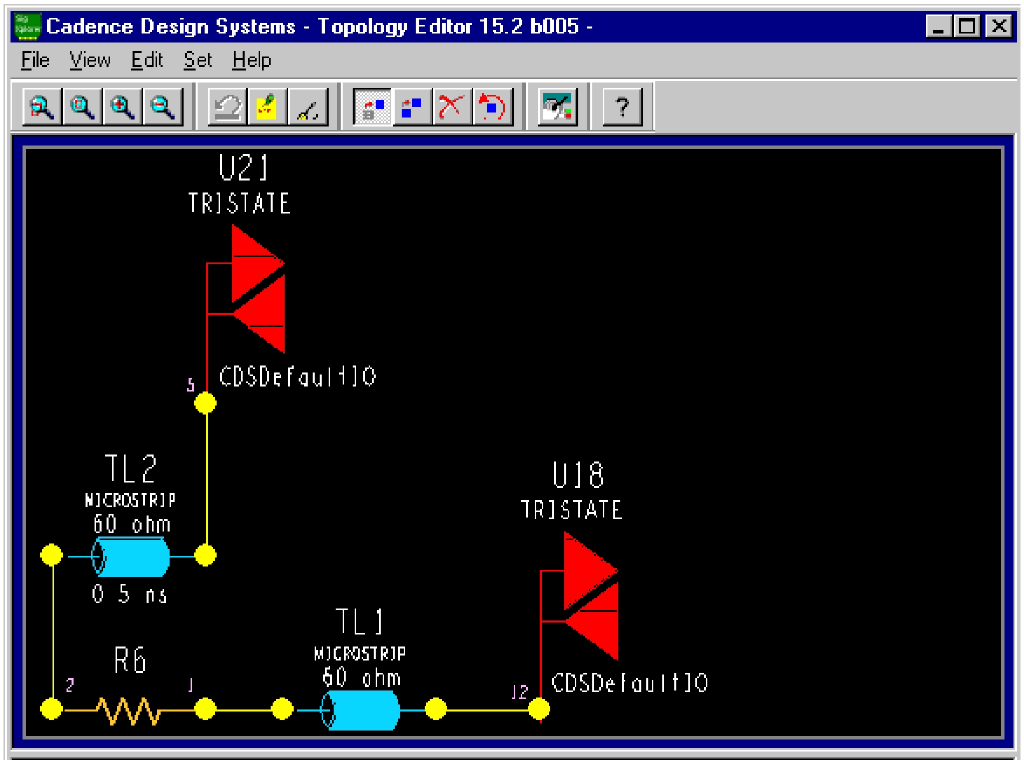
Object:
"INTER_NO_TERMI"
currently references Electrical CSet:
"INTER_NO_TERMI".

The Electrical CSet contains scheduling and/or T-point information and the application cannot extract this information for the object. I.e. launching SigXp for the object will default the scheduling and lose all T-points.

What do you wish to launch SigXplorer with?

- ❑ Select *Object*, if you want to launch SigXplorer on the object.
- ❑ Select *Referenced Electrical Set* if you want to modify the topology and constraints defined in the ECSet.

SigXplorer opens and displays the design object and valid models assigned to the devices.



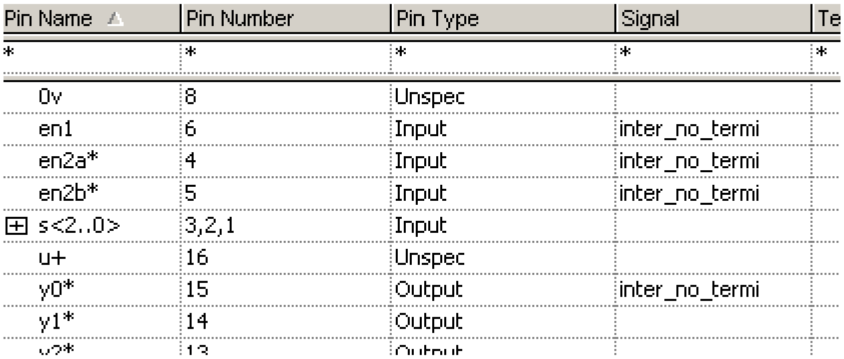
Note: If a discrete device is not assigned a valid signal model, SigXplorer will not launch on the XNet.

Inserting T-Points in SigXplorer

SigXplorer reads electrical constraints on a net or XNet, which you can modify. You can also create new topology files to capture electrical constraints in SigXplorer.

During the design capture stage, if you have a one-to-many connection from a driver to receiver, you can use the SigXplorer to modify the topology, such that all receivers receive the signal in parallel at with the same delay value.

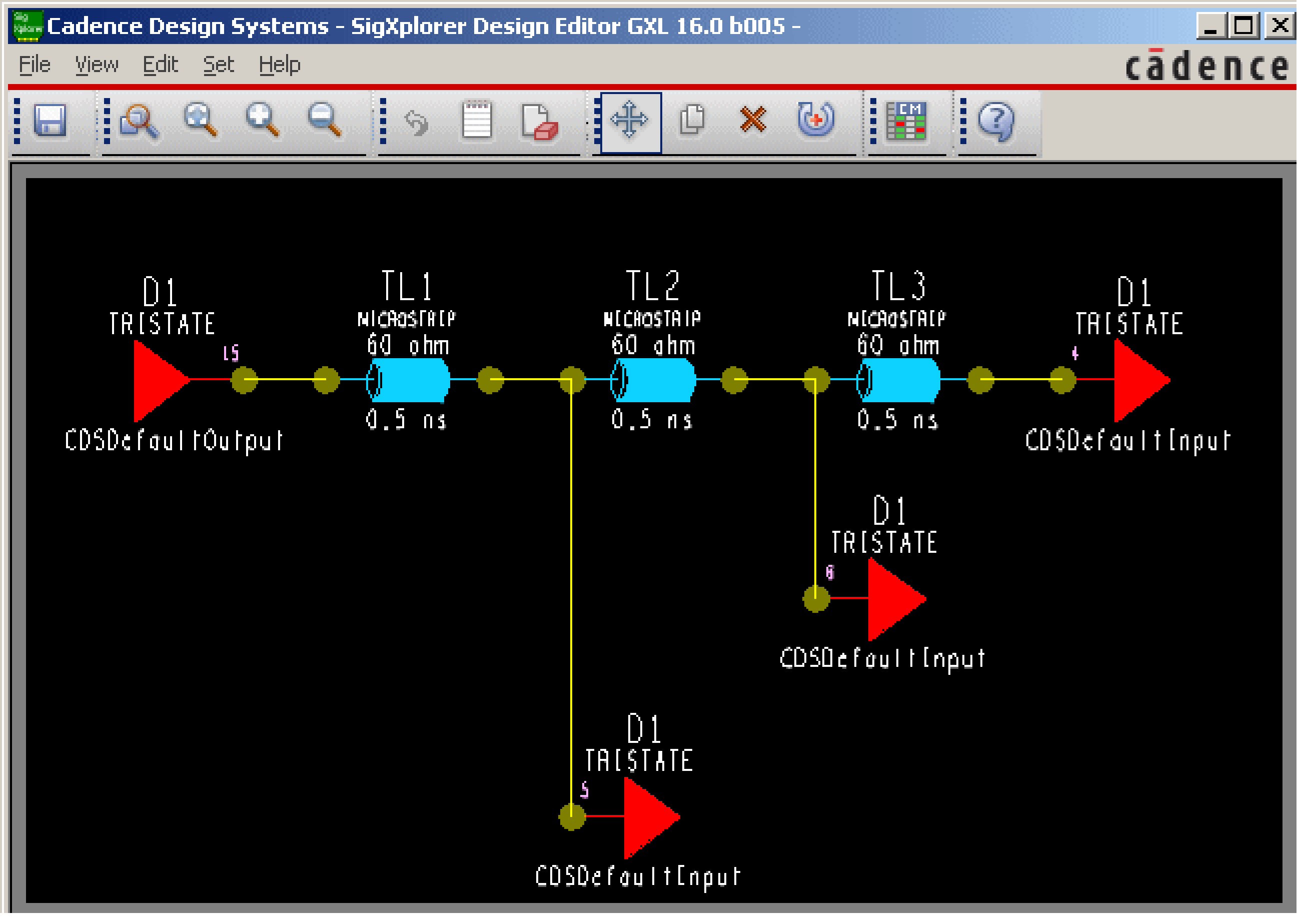
A design that has one driver connected to multiple receivers is shown in the following figure:



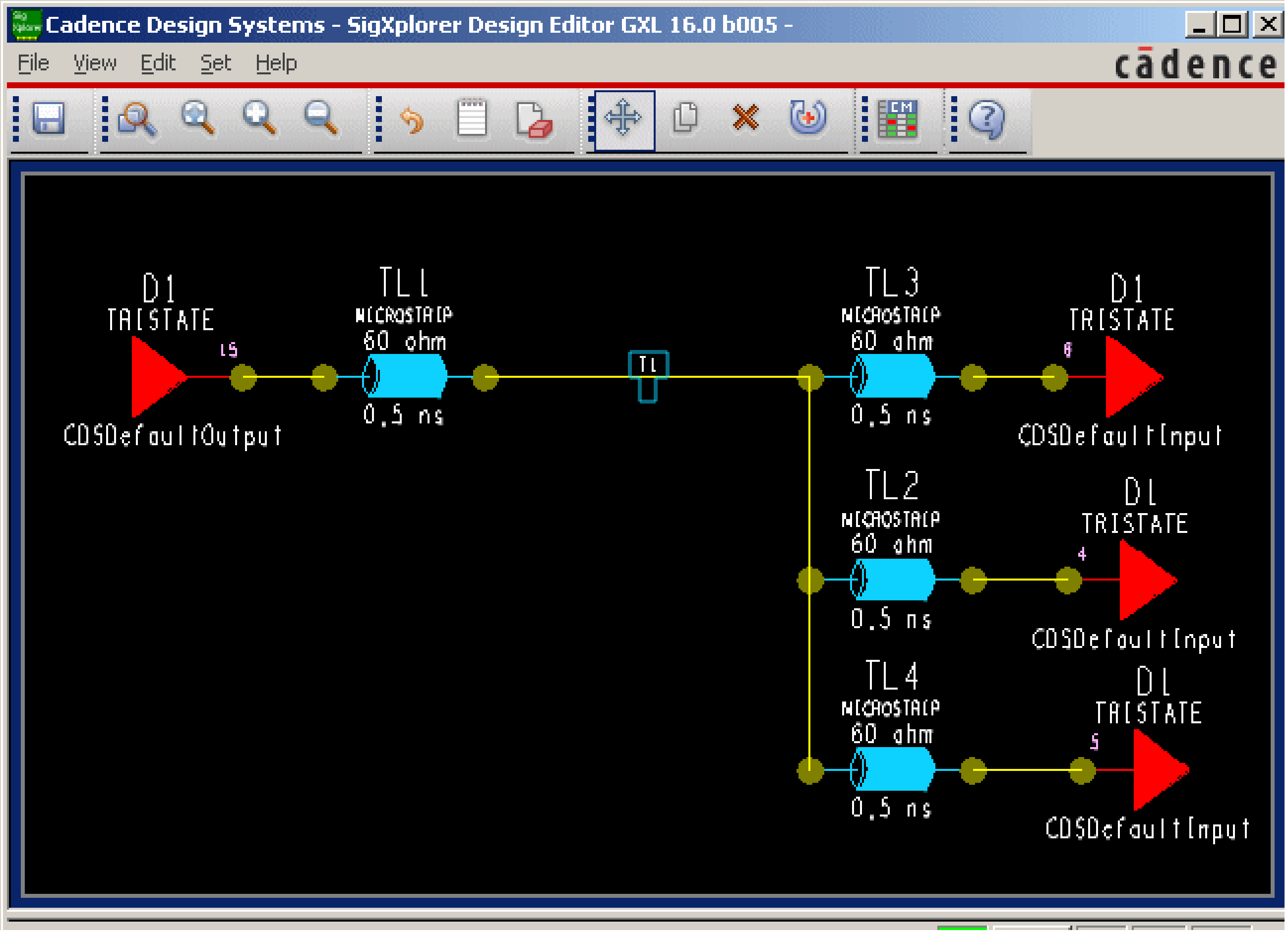
Pin Name	Pin Number	Pin Type	Signal	Te
*	*	*	*	*
0v	8	Unspec		
en1	6	Input	inter_no_termi	
en2a*	4	Input	inter_no_termi	
en2b*	5	Input	inter_no_termi	
⊞ s<2..0>	3,2,1	Input		
u+	16	Unspec		
y0*	15	Output	inter_no_termi	
y1*	14	Output		
y2*	13	Output		

Figure 6-1 Design with One-To-Many Driver to Receiver connection

If you now open Constraint Manager and launch SigXplorer on the net, the topology of the net is as shown in the following figure:



If you now modify the topology, the final topology will be as illustrated in the following figure:



As you modify the topology, note that a new element, T-point, is added automatically. The topology modifications are saved as a topology (`.top`) file. The advantage of using T-points is that you can specify different separate propagation delay constraints for net segments from driver to t-point and from t-point to each receiver.

If you now update Constraint Manager (see Updating Constraint Manager with Changes in SigXplorer) with these changes, they are imported as an ECSet. The next time when you launch SigXplorer on the ECSet, the topology of the net is the same as that shown in the figure above.

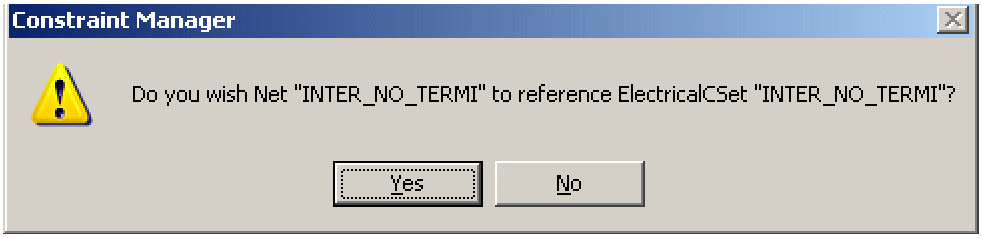
Updating Constraint Manager with Changes in SigXplorer

After you have made changes to design constraints in SigXplorer, you can import these changes to Constraint Manager. All topology changes in SigXplorer are imported into Constraint Manager as ECSets.

To update Constraint Manager with the constraint modifications in SigXplorer, do the following:

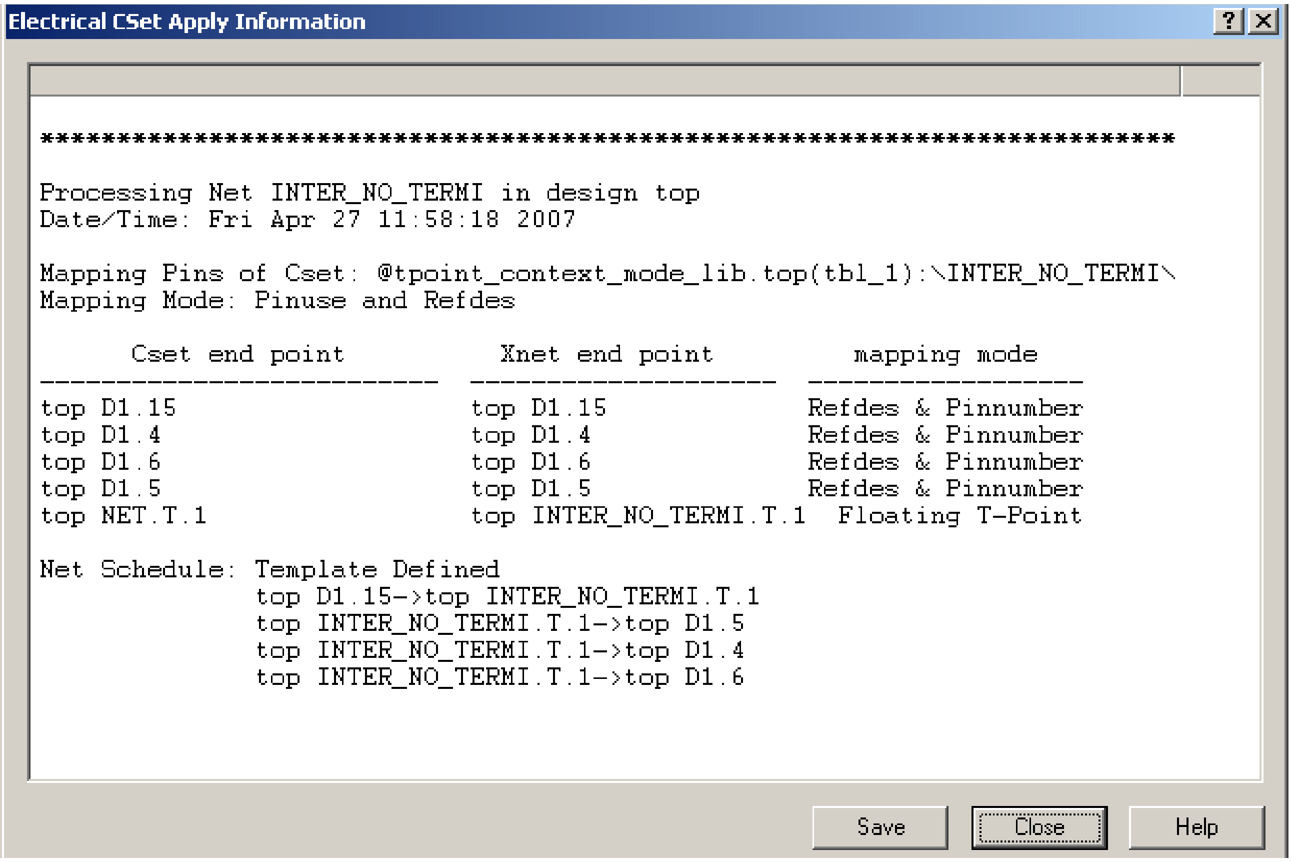
➤ In SigXplorer, choose *File – Update Constraint Manager.*

A new ECSet is created in Constraint Manager. A message box is displayed asking whether the new ECSet should be applied on the object on which SigXplorer was initially launched.



Do you wish Net "INTER_NO_TERMI" to reference ElectricalCSet "INTER_NO_TERMI"?

You might or might not choose to apply ECSet on the object. If you apply the ECSet a message box displaying the status appears.



```
****************************************************************************

Processing Net INTER_NO_TERMI in design top
Date/Time: Fri Apr 27 11:58:18 2007

Mapping Pins of Cset: @tpoint_context_mode_lib.top(tbl_1):\INTER_NO_TERMI\
Mapping Mode: Pinuse and Refdes

      Cset end point          Xnet end point         mapping mode
--------------------------  --------------------  ------------------
top D1.15                   top D1.15             Refdes & Pinnumber
top D1.4                    top D1.4              Refdes & Pinnumber
top D1.6                    top D1.6              Refdes & Pinnumber
top D1.5                    top D1.5              Refdes & Pinnumber
top NET.T.1                 top INTER_NO_TERMI.T.1  Floating T-Point

Net Schedule: Template Defined
              top D1.15->top INTER_NO_TERMI.T.1
              top INTER_NO_TERMI.T.1->top D1.5
              top INTER_NO_TERMI.T.1->top D1.4
              top INTER_NO_TERMI.T.1->top D1.6
```

Caution

Termination points created in SigXplorer are not backannotated to Constraint Manager or Design Entry HDL.

For more information of SigXplorer, refer to Allegro SI SigXplorer User Guide.

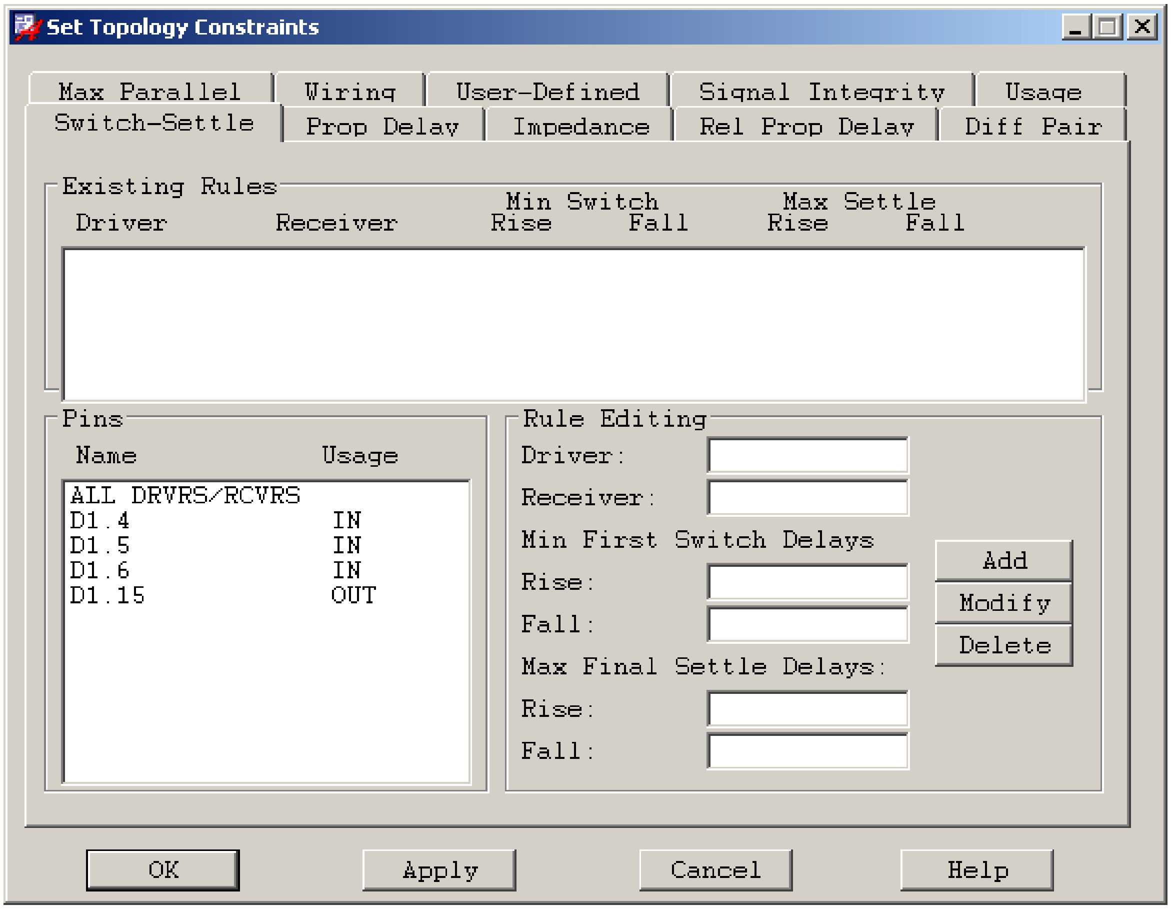
Modifying Electrical Constraints in SigXplorer

Besides modifying the topology, SigXplorer also provides support for modifying constraints on the topology. In SigXplorer, you can generate or modify electrical constraints as a topology file. The constraints added or modified in SigXplorer, as a topology template, is imported in 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.

To add constraint information of an ECSet in SigXplorer, perform the following steps:

1. Launch SigXplorer on ECSet.
2. Choose *Set – Topology Constraints.*

   The Set Topology Constraints dialog box is displayed for you. You can use the tabbed pages of this dialog box, to set various constraints such as minimum and maximum propagation delay, impedance, differential pair constraints, and so on.



Name	Usage
ALL DRVRS/RCVRS	
D1.4	IN
D1.5	IN
D1.6	IN
D1.15	OUT

3. Select the required tab.

   For example, to specify the differential pair constraints, select the *Diff Pair* tab.
4. Specify the constraints values.

   For example, in the Diff Pair page, add values in the Line Width text box to specify the line width of a differential pair signal.
5. Click *Apply*.

If required, select any other tab and specify the constraint values. While adding new constraints in the Switch-Settle, Prop Delay, Impedance, Max Parallel and User Defined pages of the Set Constraint dialog box, you need to specify the constraint details in the Rule Editing section and then click the *Add* button.

Similarly, while modifying constraints, select an entry in the Existing Rules section, modify the constraint values in the Rule Editing section, and then select the *Modify* button.

6. Click *OK*.
7. To save the constraint information as a topology (.top) file, choose *File – Save* or *File – Save As*.

To know more about specifying constraints in SigXplorer, see to *Allegro SI SigXplorer User Guide*.

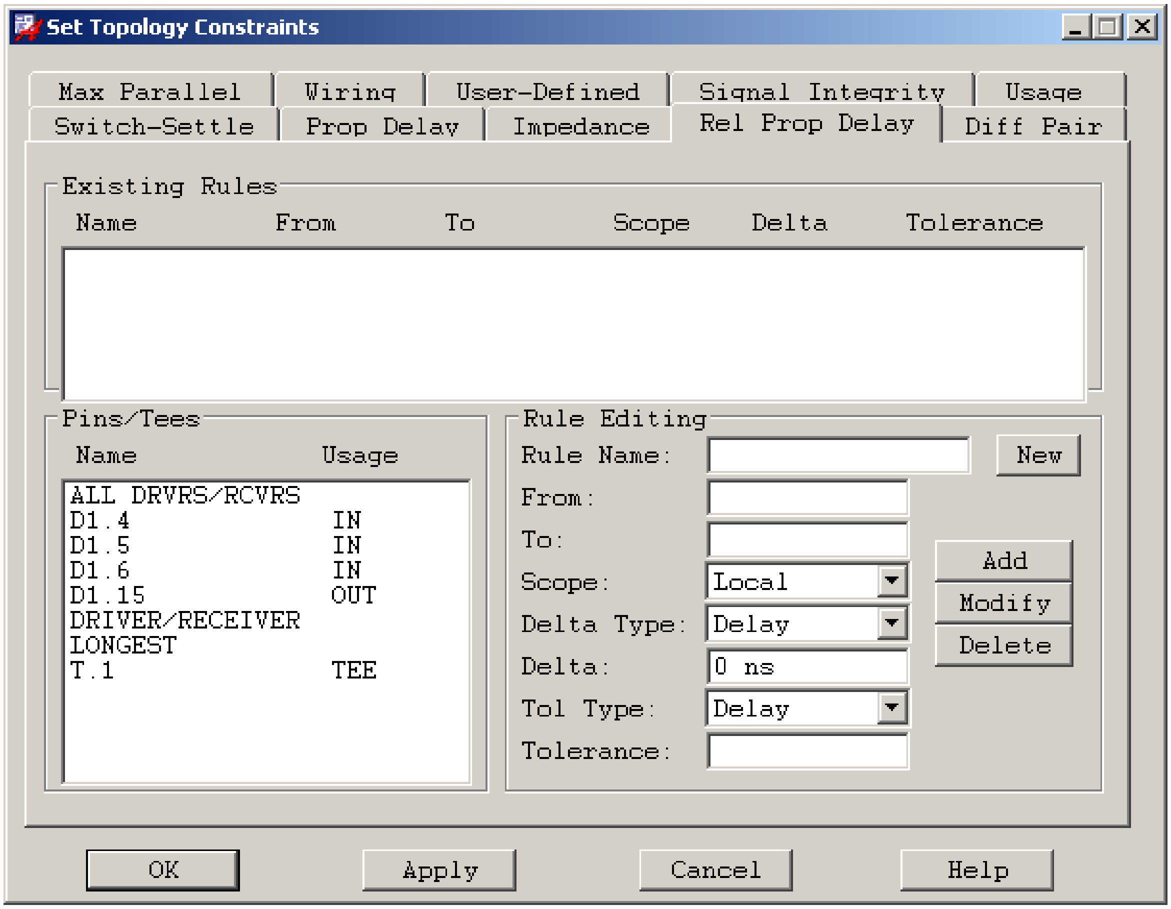
Creating ECSet Generated Match Groups in SigXplorer

While editing constraints in SigXplorer, if you modify or add a rule to specify the relative propagation delay between a pin-pair, and then update Constraint Manager with these changes, a new ECSet and a match group are created in Constraint Manager.

While you modify the relative propagation delay constraints, you can also add new rule that will create ECSet generated match groups, using the following steps.

1. Launch SigXplorer on a design object.
2. Choose *Set – Constraints*.

3. In the Set Topology Constraints window, select the *Rel Prop Delay* tab.



Existing Rules

Name	From	To	Scope	Delta	Tolerance

Pins/Tees

Name	Usage
ALL DRVRS/RCVRS	
D1.4	IN
D1.5	IN
D1.6	IN
D1.15	OUT
DRIVER/RECEIVER	
LONGEST	
T.1	TEE

Rule Editing

Rule Name:
From:
To:
Scope: Local
Delta Type: Delay
Delta: 0 ns
Tol Type: Delay
Tolerance:

4. Click *New*.

   The Rule Name field in the Rule Editing section gets populated.

5. To specify the pins between which you want to specify the relative propagation delay, select appropriate pins from the Pin/Tee section for From and To fields.

6. Similarly, specify other values for all other fields in the Rule Editing section and click *Add*.

   The relative propagation delay constraint that you have created gets added in the Existing Rules section.

7. Click *OK*.

8. Update Constraint Manager with these modifications.

   To view the steps for updating Constraint Manager with modifications in SigXplorer, see Updating Constraint Manager with Changes in SigXplorer.

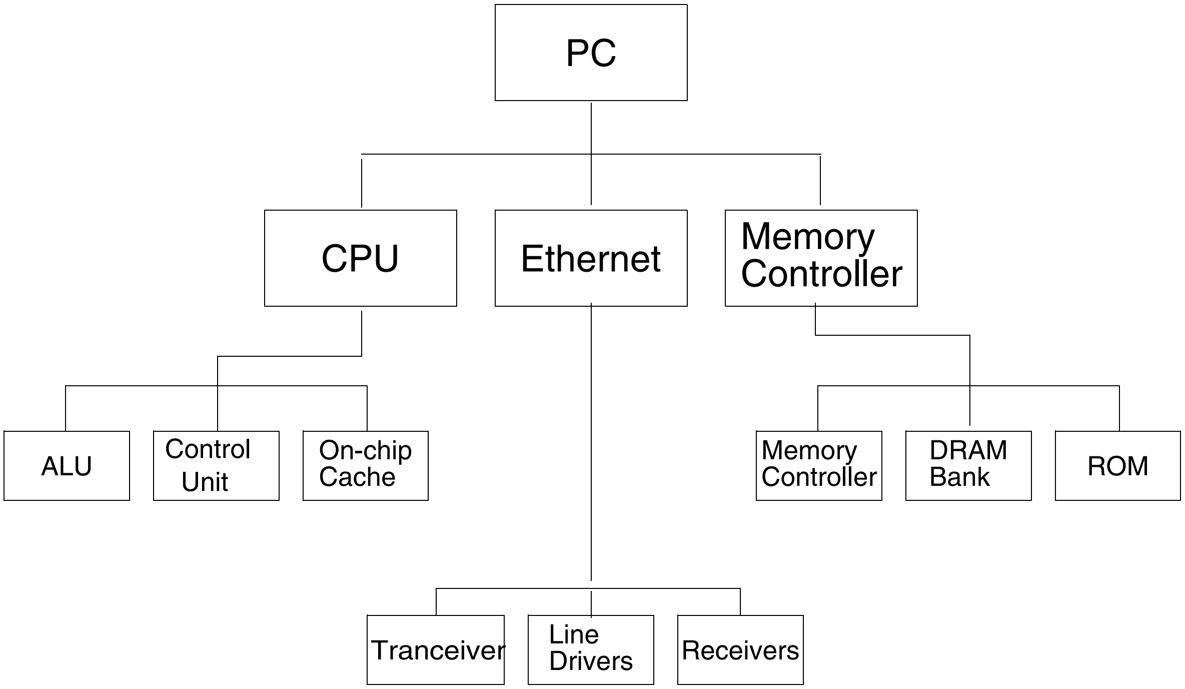
After you have updated Constraint Manger with the topology and constraint modifications, select the Relative Propagation Delay worksheet in the Net folder of the Electrical tab. The pin pair created in the *Rel Prop Delay* page of the Set Constraints dialog box in SigXplorer, is listed. Similarly, you can create multiple pin pairs by adding new rules in SigXplorer.

7

Constraints in Hierarchical Designs

Hierarchical Designs Overview

A hierarchical design is a large and complex design divided into subdesigns (hierarchical blocks). Each of the subdesigns can be further divided into subdesigns. For example, you might have a hierarchical design called PC that contains the following subdesigns: CPU, Ethernet, and Memory Controller.



The hierarchical design method is typically followed for large and complex designs. These designs are divided into individual modules where each module represents a logic function.

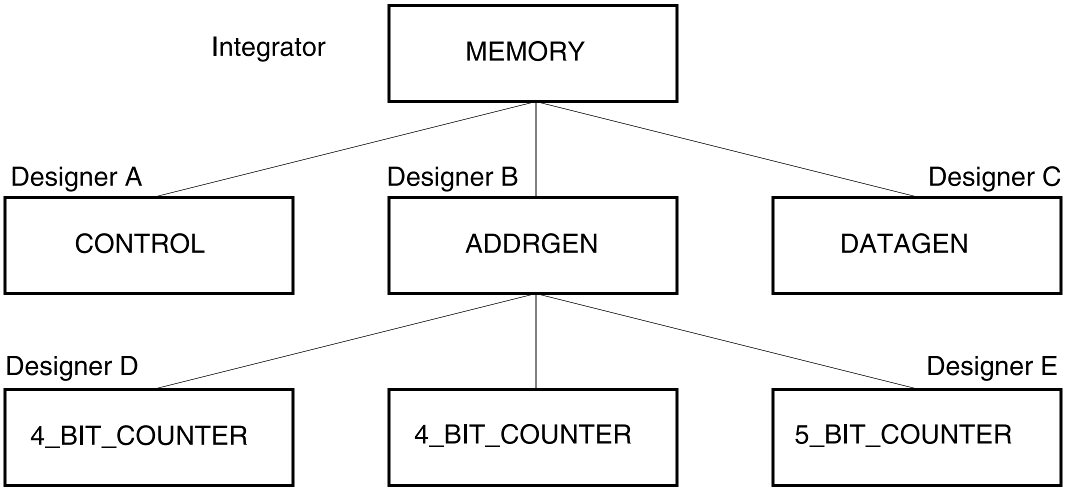
The chapter discusses the following:

- Team Design on page 146

- Assigning Constraints in Hierarchical Designs on page 146

Team Design

Teams of designers might work on a hierarchical design consisting of various blocks. In this team design environment, each designer could work on a block in the hierarchical design. After all the designers working on the blocks have completed their work, the Integrator (a Team Leader or a designated person) integrates all the blocks into the top-level design.



In the preceding example of a hierarchical design (`MEMORY`), teams of designers work on different blocks of the design and the Integrator integrates all the blocks into the top-level design `MEMORY`.

If a block is instantiated more than once in a design, it is called a reuse or replicated block. For example, `4_BIT_COUNTER`, is a replicated block because it is instantiated twice in the schematic for the `ADDRGEN` block.

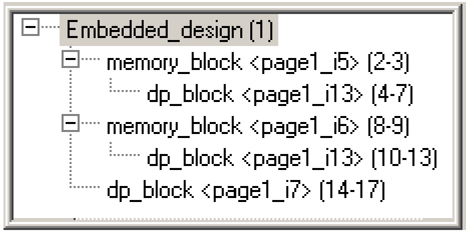
Assigning Constraints in Hierarchical Designs

You can adopt one of the following two methods to assign constraints in hierarchical designs:

1. Assign constraints in the context of the top-level design.

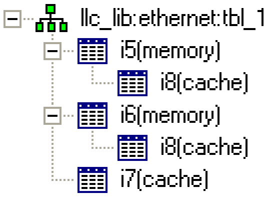
   You can adopt this method if you are not creating the hierarchical design in a team design environment, or if you do not want to create physically reusable blocks. In this method, you assign constraints for all the lower-level designs in the context of the top-level design.

For example, if you have a hierarchical design similar to the following one, you should assign constraints to the `memory_block` lower-level design when the top-level design named `Embedded_design` is set as the root design.



```
Embedded_design (1)
    memory_block <page1_i5> (2-3)
        dp_block <page1_i13> (4-7)
    memory_block <page1_i6> (8-9)
        dp_block <page1_i13> (10-13)
    dp_block <page1_i7> (14-17)
```

Hierarchical Design in Design Entry HDL



```
llc_lib:ethernet:tbl_1
    i5(memory)
        i8(cache)
    i6(memory)
        i8(cache)
    i7(cache)
```

Hierarchical Design in SCM

2. Assign constraints in the individual blocks (sub-designs) in the hierarchical design.

   You can adopt this method if you are creating the hierarchical design in a team design environment, or if you want to create physically reusable blocks.

   In this method, you set the block in which you want to assign constraints as the root design for the project and then assign constraints in the block. When you add the block in a design, the constraints in the block appear as inherited constraints in the design. You can then choose to override the inherited constraints from the block or retain them in the design. The constraints inherited from lower-level designs are referred to as *lower-level constraints*.

   For example, if a hierarchical design named `Embedded_design` with two blocks named `memory_block` and `dp_block` is being created in a team design environment, the designers working on the blocks assign constraints in each block when the block is set as the root design. When the owner of the top-level design named `Embedded_design` uses the blocks, the constraints in the blocks appear as inherited constraints in `Embedded_design`. The owner of `Embedded_design` can choose to override the constraints inherited from the blocks or retain them. Constraints can also be assigned on the lower-level design in the context of the top-level design, `Embedded_design`.

Note: In Constraint Manager, MMDP/LLDP will be visible in context of the root design if the nets are interfaced nets and will be visible under design instances if they are local nets. In case MMDP/LLDP exists for local nets, and the nets are converted to interface nets, then MMDP/LLDP continue to be visible under design instance.

Caution

Constraints overridden in the top-level design (for example, `Embedded_design)` ***cannot be pushed down to lower-level designs***. ***If you need to override constraints in all the occurrences of the lower-level design (***`memory_block` or `dp_block)`***, you should make the changes in the lower-level design.***

Working with Constraints in Lower-Level Designs

This section explains how lower-level constraints appear in a top-level design and how you can use them effectively. The Design Entry HDL-Constraint Manager flow includes support for handling constraints in schematic blocks in hierarchical designs. You can capture constraints in a schematic block and later pull the constraints into a top-level design by instantiating the block in the top-level design. These constraints are visible and can be edited in the context of the top-level design.

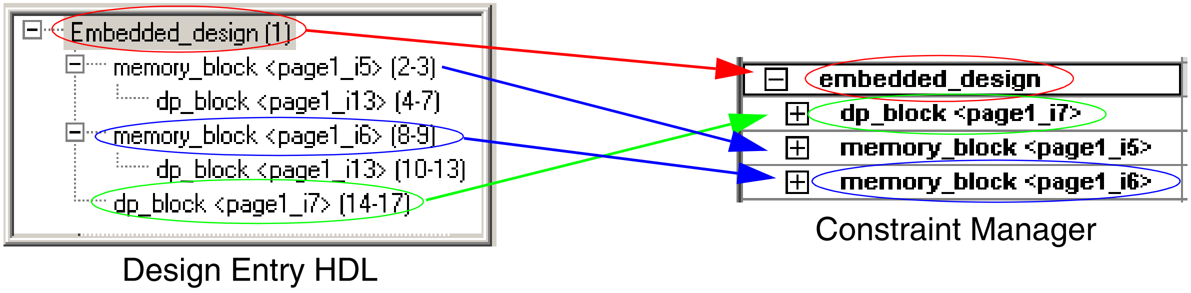
Constraint changes made in a top-level design will override constraints coming from the lower-level design in the context of the top-level design. You can work with lower-level designs to do the following:

- view constraints as defined in the lower-level design in the context of the top-level design.
- make changes that can be reflected in all instances of the lower-level design.
- refresh a higher-level writable block with the current information from a lower-level design.

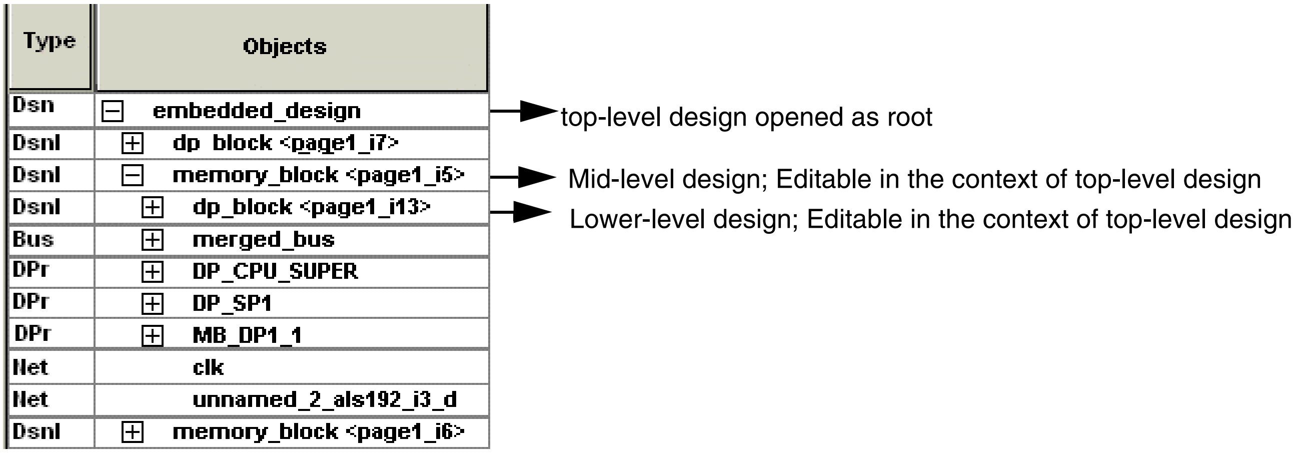
Understanding How Hierarchical Designs Appear in Constraint Manager

Constraint Manager displays all lower-level design blocks in a hierarchical design, separately. Constraint Manager treats the root design as the top-level design and displays all the lower-level designs below it.

For instance, in the following example, `Embedded_design` is the top-level design and it includes two instances of `memory_block` and one instance of `dp_block`. Constraint Manager displays the design in a hierarchical view.


Embedded_design (1)
memory_block <page1_i5> (2-3)
dp_block <page1_i13> (4-7)
memory_block <page1_i6> (8-9)
dp_block <page1_i13> (10-13)
dp_block <page1_i7> (14-17)
Design Entry HDL
embedded_design
dp_block <page1_i7>
memory_block <page1_i5>
memory_block <page1_i6>
Constraint Manager

The root design is treated as an instance under `PS System`. For example, note that while `Embedded_design` is the root design, it appears as an instance under `PS System`.



Type	Objects
Dsn	⊟ embedded_design
Dsnl	⊞ dp_block <page1_i7>
Dsnl	⊟ memory_block <page1_i5>
Dsnl	⊞ dp_block <page1_i13>
Bus	⊞ merged_bus
DPr	⊞ DP_CPU_SUPER
DPr	⊞ DP_SP1
DPr	⊞ MB_DP1_1
Net	clk
Net	unnamed_2_als192_i3_d
Dsnl	⊞ memory_block <page1_i6>

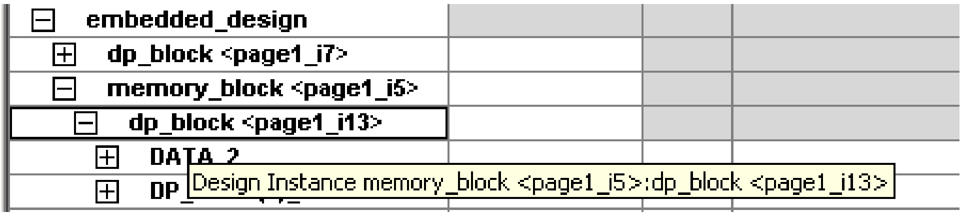
top-level design opened as root

Mid-level design; Editable in the context of top-level design

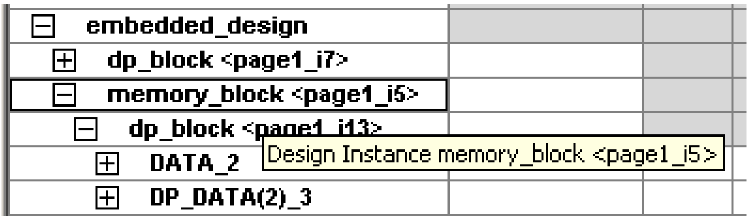
Lower-level design; Editable in the context of top-level design

Each design has an associated tooltip and its path details are displayed in the tooltip as well as on the status bar. Lower-level designs included under the top-level design appear with the tooltip displaying “design”. Lower-level designs included within another lower-level design are treated as instances and the tooltip displays “design instance”. The name of the design and its path is also mentioned. Further, the Type column lists the type of the object; for example, *Dsn* represents the root design, and *DsnI* represents an instance of a lower-level design.

If a design instance has more blocks, then the tooltip has the syntax `Design instance \<design name>\<Location>: <sub_design name>\<Location>`. For a design with instantiated lower-level designs, the lower-level design names appear before the net names or constraint names.

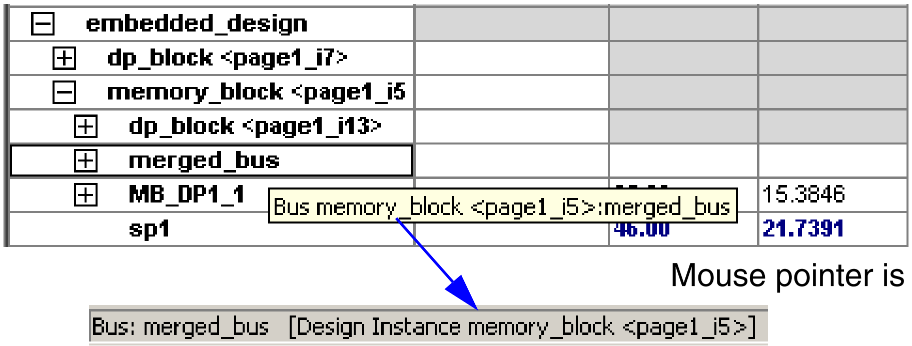


Design Instance memory_block <page1_i5>:dp_block <page1_i13>



Design Instance memory_block <page1_i5>

You can find the source information for an object by placing the mouse pointer on the object. For example, the following figure shows the message displayed on the status bar when the mouse pointer is placed on a bus, `merged_bus`.



Bus memory_block <page1_i5>:merged_bus

Bus: merged_bus [Design Instance memory_block <page1_i5>]

Mouse pointer is displayed on `merged_bus`

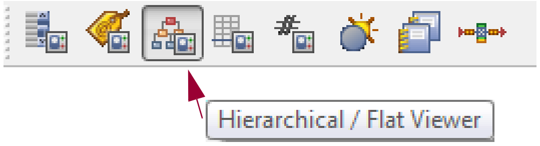
and the status bar displays object's details.

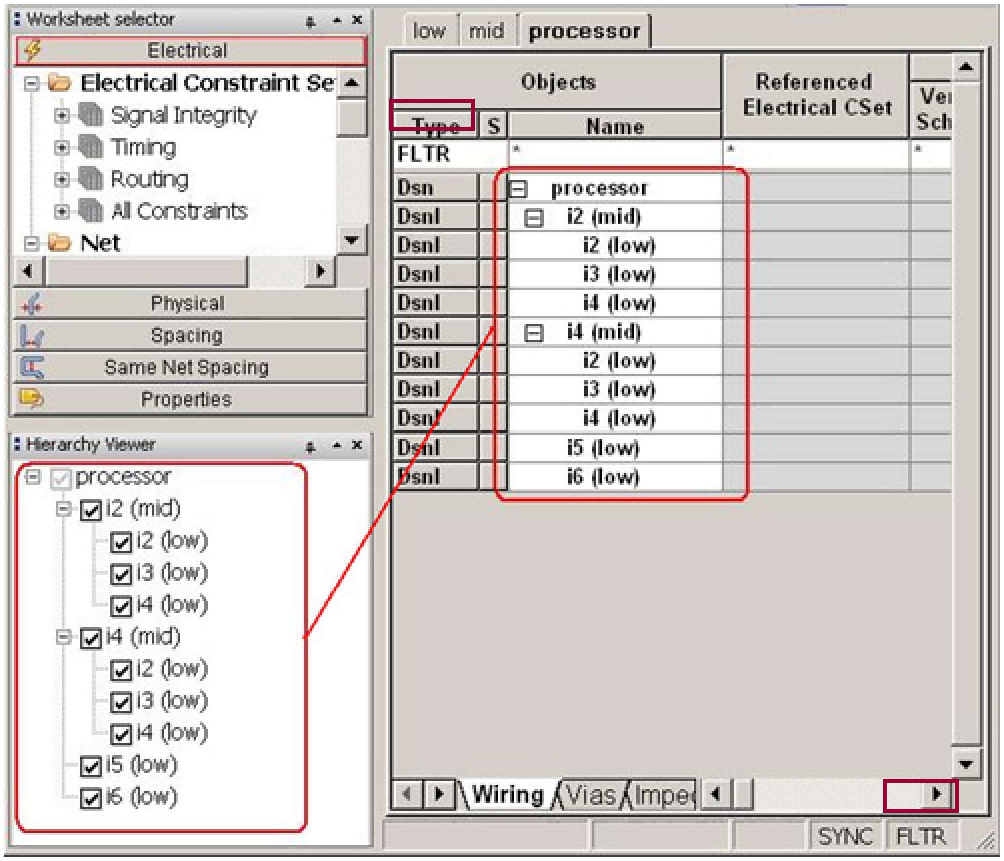
Flat and Hierarchical Display for Hierarchical Designs

When working with hierarchical designs, you might want to filter XNets, nets and differential pairs based on the top-level design name and view these objects in a flat structure. For instance, if a top-level net is connected to an XNet at a lower level, or is part of a differential pair at a lower level, you might want to view only the top net and not the XNet or differential pair. You might want to view a top-level net only in the top-level design, without any lower-level object association.

Using the Design Instance/ Block filter panel, available by default in the Worksheet Selector, and the Hierarchical/Flat Viewer toggle button, you can filter the objects in the design and view the design in the hierarchical, or flat mode.

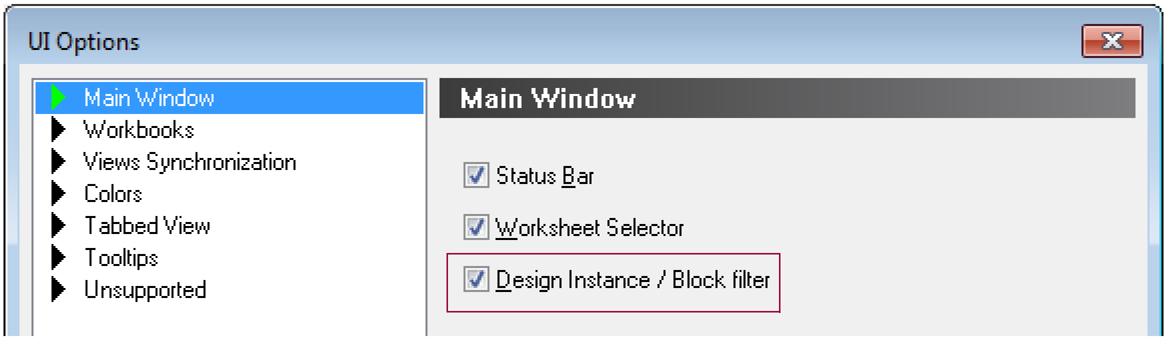
This feature helps you easily locate objects in hierarchical designs. For example, top-level objects might not be visible in lower levels of the hierarchy, or top-level nets are connected to lower-level nets and the lower-level objects are grouped. In such cases, you can use the toggle button to find objects easily.





Objects			Referenced Electrical CSet	Ver Sch
Type	S	Name		
FLTR		*	*	*
Dsn		⊟ processor		
Dsnl		⊟ i2 (mid)		
Dsnl		i2 (low)		
Dsnl		i3 (low)		
Dsnl		i4 (low)		
Dsnl		⊟ i4 (mid)		
Dsnl		i2 (low)		
Dsnl		i3 (low)		
Dsnl		i4 (low)		
Dsnl		i5 (low)		
Dsnl		i6 (low)		

This option can be enabled in the UI Options dialog (*View — Views Options*).



UI Options

- Main Window
- Workbooks
- Views Synchronization
- Colors
- Tabbed View
- Tooltips
- Unsupported

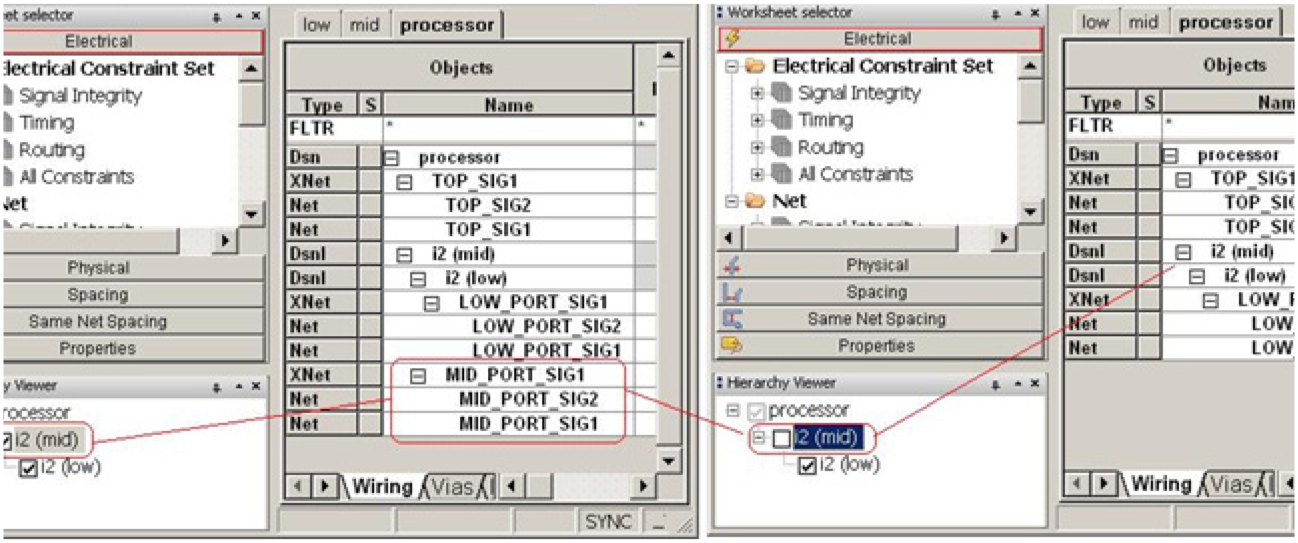
Main Window

☑ Status Bar

☑ Worksheet Selector

☑ Design Instance / Block filter

When you work in the hierarchical mode, Constraint Manager displays design instances, but not the objects that belong to the design instances.



Type	S	Name
FLTR		*
Dsn		⊟ processor
XNet		⊟ TOP_SIG1
Net		TOP_SIG2
Net		TOP_SIG1
Dsnl		⊟ i2 (mid)
Dsnl		⊟ i2 (low)
XNet		⊟ LOW_PORT_SIG1
Net		LOW_PORT_SIG2
Net		LOW_PORT_SIG1
XNet		⊟ MID_PORT_SIG1
Net		MID_PORT_SIG2
Net		MID_PORT_SIG1

Type	S	Nam
FLTR		*
Dsn		⊟ processor
XNet		⊟ TOP_SIG1
Net		TOP_SIG
Net		TOP_SIG
Dsnl		⊟ i2 (mid)
Dsnl		⊟ i2 (low)
XNet		⊟ LOW_I
Net		LOW
Net		LOW

Note: If you switch tabs, say from Wiring to Vias, the same filtered-out Design Instances as in the previous tab continue to be displayed.

Inherited and Overridden Constraints

- Inherited constraint —It is a property value which is inherited from a parent object. Lower-level constraints are called inherited constraints. By default, a cell which is populated and colored regular black reflects that the value is inherited from a parent object.
- Overridden constraint — Constraints that are modified in the context of the top-level design are called overridden constraints. As such overridden constraints belong to the design specified as the root design. By default, a worksheet cell is colored bold blue if the value it contains is overridden.

You can identify whether a constraint is inherited or overridden by its color. Within a design, constraints may be displayed in blue color or gray color.

- Blue color—Indicates a constraint added on an object in the top-level design or an inherited constraint that is overridden in the top-level design.
- Gray color—Indicates a constraint inherited from a lower-level block or from an ECSet. You can override the inherited constraints in the top-level design.

Note: If you create an object, such as a differential pair, match group or ECSet, the constraints assigned to the object are inherited by its member nets. If you change the constraints on the member nets, Constraint Manager treats the change as constraint overrides and displays the overrides in blue color.

Rules For Merging Lower-Level Constraints

General Rules

1. If you delete a pin pair, differential pair or match group in a lower-level block (by setting the lower-level block as the root design), the pin pair, differential pair or match group gets deleted in context of the top-level design even if you have overridden the constraints on the pin pair, differential pair or match group in context of the top-level design.

For example, assume that you have a pin pair in a lower-level block named `memory`. Override the constraints on the pin pair in a higher level design named `ethernet` (that is set as the root design). If you now set the `memory` block as the root design, delete the pin pair and then set the `ethernet` design as the root design, the pin pair is no longer displayed in context of the `ethernet` design, even though the constraints on the pin pair were overridden in context of the `ethernet` design.

2. When you apply ECSets in the design by running the *File – Update Constraint Manager* command in SigXplorer or the *File – Import – Electrical CSets command in Constraint Manager*, if the ECSet apply results in addition or deletion or terminations (coming from the topology) the addition or deletion of terminations will be done only for the nets in the root design and not for the nets in the lower-level blocks.

Local Nets, XNets and Buses

Local nets, XNets, and buses belong to individual designs. To assign constraints to these objects, you need to open the design as the root design.

1. Constraints assigned to local nets, XNets, and buses in a lower-level design are visible at higher levels. If you change the constraint value for a local net, XNet or bus in the lower-level design, the constraint changes are visible at the top-level design also.
2. Constraints added in lower-level designs appear as inherited values in the top-level design. You can override these inherited values in the top-level design. If you override an inherited constraint in the top-level design and then change the same constraint in the lower-level design, the overridden constraint value in the top-level design is not impacted. The constraints defined in the top-level design wins.
3. You can set different overrides for different instances of a reused block at different levels of hierarchy.

   For example, assume that a block named `memory_block` has the `MAX_OVERSHOOT` constraint with the value `5100:-610`. If you add two instances of this block in a design named `Embedded_design`, the `MAX_OVERSHOOT` constraint in `memory_block` appears as an inherited constraint in the context of the top-level design, `Embedded_design`. You can override the constraint on each instance of the block by assigning different values. For example, you can override the constraint on the first instance of `memory`_block by specifying the value `5000:-600` and override the constraint on the second instance of the block by specifying the value as `5300:-630`.
4. A constraint coming from a lower-level design when deleted at the top-level design makes the lower-level value stale. Such constraints are not pushed to the top-level design even when changed at the lower-level design. It is recommended that you take caution while deleting or changing values of lower-level constraints in the top-level design as the

changes in lower-level constraints are not automatically rolled back to the top-level design. To manually roll back changes, use one of the following two methods:

a. Manually edit the value to match the value in the lower-level design. When the values are in-sync, any future edits done in the lower-level design will ripple up to the top-level.

b. Delete and add the block instances again.

5. Changes made to an object in a higher level design are ignored if the object is deleted from lower-level designs. For example, if you create a pin-pair in a lower-level design and assign constraints to it, then the pin-pair definition and constraint values ripple up the hierarchy. You can override the constraint values in higher level design. If you now delete the pin-pair from the lower-level design, then the same is also deleted from the higher level design.

Interface Nets

1. Constraints assigned on interface nets are merged with the base net and appear in the context of the top-level design. If the interface nets have the same constraint with different values, any one of the constraint values will be used.
2. If an interface net in a lower-level design is connected to an XNet in a higher-level design, the interface net becomes a member of the XNet.

Global Nets and Buses

1. Constraints assigned to global signals in lower-level designs are rippled up and displayed in the top-level design.

Adding/base

Consider a scenario where an eight bit bus MERGED_BUS_ALIAS<7..0>, is aliased with ADDR<3..0> and SUPER_SIMPLE_BUS_ALIAS<3..0>. Now, save this design and open Constraint Manager. You will notice that the MERGED_BUS_ALIAS does not show as a complete bus. If you want to show MERGED_BUS_ALIAS a the winning bus, then you can make the MERGED_BUS_ALIAS a Base using \Base - then this bus becomes the winning bus for all the eight bits. However if the MERGED_BUS_ALIAS is made an Interface MERGED_BUS_ALIAS >7..0>\I - then this bus wins and not ADDR.

Differential Pairs

1. Constraints assigned to differential pairs in lower-level designs appear as inherited values in the top-level design.

2. If there are conflicts in differential pairs because the same differential pair name exists across multiple instances of a design imported in a top-level design, then the differential pair names are renamed automatically. The differential pair in the first merging design retains its name while the second differential pair is renamed using a numerical suffix.

3. If you rename a differential pair at any level of a hierarchical design, the new name is displayed in the top-level design.

   For example, assume that the block `dp_block` has a differential pair named `CAC_DP1`. If you set `memory_block` as the root design and rename the differential pair to `CACHE_DP1`, and then set `Embedded_design` as the root design, the differential pair is displayed as `CACHE_DP1` in the design. If you now set `dp_block` as the root design and rename the differential pair to `CAC_DP1`, and then set `Embedded_design` as the root design, the differential pair is displayed as `CAC_DP1`.

   Note: Renaming a differential pair does not affect inheritance of constraints from lower-level designs.

4. If you make any changes to constraint values in a differential pair in a lower-level design, even after renaming the differential pair in the top-level design, the changes to constraint values in the lower-level design are propagated to the top-level design.

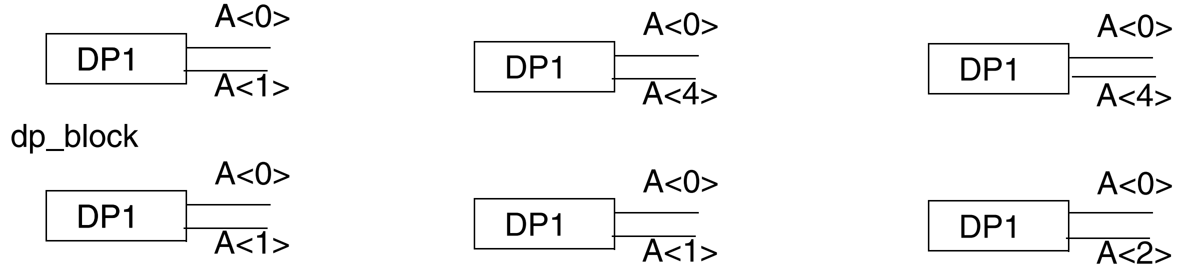
5. If you have an interface net differential pair which has nets connected from a lower-level design to a top-level design, the differential pair appears in the lower-level design and it shows nets that are available at the top-level design. For example, assume you have an interface net differential pair `DP1`, which contains nets `sp1` and `sp2`, in the design `DP_block`. Assume that these nets are connected in the top-level design `memory_block` to nets `p1` and `p2`. In Constraint Manager, a differential pair `DP1` containing the nets `p1` and `p2` will appear in the (lower-level) design `DP_block`.

6. If you have an inherited interface net differential pair and you rename it in a top-level design, and then rename the differential pair in the lower-level design, the renamed lower-level differential pair is not rippled up to the top-level design.

7. If you change the membership of a differential pair object in the context of the top-level design, and then set the block in which the differential pair exists as the root design and

change the membership of the differential pair again, the membership changes you made in the context of the top-level design wins. To understand this, see the example below:

`memory_block` design: It contains an instance of `dp_block`



Step 1
`dp_block` is root
Differential pair ripples
to `memory_block`

Step 2
`memory_block` is root
Member net changed
in `memory_block`

Step 3
`dp_block` is root
Member net is changed
However, nets in
`memory_block` win

Match Groups

1. Constraints assigned to match groups in lower-level designs ripple up in hierarchy and appear as inherited values in the top-level design.
2. A match group assigned in a lower-level design is reflected in the top-level design with the same target net relationship. However, you can change the target net in match groups inherited from lower-level designs in the context of the top-level design. This will assign an overridden value to the original target net.

   For example, assume that you have a match group `MG_RST_BUS` in `memory_block` with `rst_bus<0>`, `rst_bus<1>`, and `rst_bus<3>` as member nets and `rst_bus<0>` as the target net. When you make `embedded_design` as the root design, the match groups in the two instances of `memory_block` appear as `MG_RST_BUS` and `MG_RST_BUS_1`. You can set a different target net in these match groups. For instance, you can make `rst_bus<2>` as the target net in `MG_RST_BUS` match group.
3. If there are conflicts in match groups in case the same match group name exists across multiple instances of a design instantiated in a top-level design, the match groups are renamed automatically. The match group in the first merging design retains its name while the second match group is renamed using a numerical suffix.

   Note: ECSet-generated match groups in a lower-level design are not renamed in the top-level design. as such match groups are considered as global objects. See Special Cases Where Rules for Constraint Merging Are Overridden on page 159 for more details.

4. If you rename a match group in a design, the renamed match group ripples up to higher-level designs.

   For example, assume that the block `dp_block` has a match group named `MG_RST_BUS`. If you set `memory_block` as the root design and rename the match group to `MG_RST`, and then set `Embedded_design` as the root design, the match group is displayed as `MG_RST` in the design. If you now set `dp_ block` as the root design and rename the match group to `MG_RST_LOW`, and then set `Embedded_design` as the root design, the match group is displayed as `MG_RST_LOW` in `Embedded_design`.

5. A match group in a lower-level design is displayed in the context of the top-level design. If you now add a net existing in the top-level design as a member of the match group, the net appears as a member of the match group in the lower-level design. For example, assume that a net `clk` in the top-level design named `Embedded_design` is added as a member of a match group named `MG_RST_BUS` in `memory_block`. The `MG_RST_BUS` shows `clk` as its member in lower-level design (`memory_block`) when `Embedded_design` is set as the root design. The net `clk` is no longer displayed under the root design `Embedded_design`.

6. If you rename a match group rippled up from a lower-level design in a higher-level design and if you again rename the match group in the lower-level design, the renamed lower-level match group ripples up and overrides the match group name changes done at the top-level.

   For example, assume that you have a 3-level design with `embedded_design` at the highest level, `memory_block` instantiated in `embedded_design` and `dp_block` instantiated in `memory_block`. You have a match group named `MG1_DP` in `dp_block`. This match group ripples up and appears in `memory_block` when it is made the root design. If you now rename this match group to `MG1_MEM` and make the `embedded_design` design as root, the match group will ripple up. If you make `dp_block` as the root design and change the match group name from `MG1_DP` to `MG1_DP1`, then the renamed match group name again ripples up the hierarchy. The name changes in mid- and top- levels are ignored and the last changed name wins as it ripples up in hierarchy.

7. If you add a new member net from the top-level design in a match group, which is part of a lower-level design, the new net appears in the match group under the lower-level design. For example if a net named `clk` is part of `embedded_design`, which is the top-level design and is added in the match group, `MG_RST_BUS`. `MG_RST_BUS` shows the net `clk` as a member and continues to appear under the lower-level design (`memory_block`).

Note: In pre-15.5 releases, if there were multiple replicated blocks at one level of hierarchy, you could select nets from different blocks to create differential pairs and match groups. Post release 15.5, you can create differential pairs and match groups for nets from different blocks only in the top-level design. For example, assume that you have a top-level design named

TOP that contains two instances of design MID – MID1 and MID2. Assume that MID1 contains net1 and MID2 contains net2 and both these nets are part of a differential pair DP1. In the previous releases, you could create such differential pairs in MID as TOP was treated as a flat design for the purpose of constraint-assignment. However, to create such differential pairs now, you would need to make TOP as the root design and then create differential pairs. You cannot create such differential pairs across lower-level design MID.

ECSet

1. If you make a definition change to an ECSet, it is automatically propagated up in the hierarchy.
2. If you reference an ECSet on an interface net, the constraints in the ECSet are re-applied to the aliased base net in the top-level design. In fact, all ECSet references on all objects can be re-applied at higher-levels.
3. If ECSets with the same name across different blocks have the same topology, the constraints on the ECSets that have the same topology are merged in the context of the top-level design. However, if such ECSets have a different topology, Constraint Manager retains the name for one of the ECSets. For the other ECSets, it adds a suffix of _*n* where n is a number.

   For example, if you add memory_block that has an ECSet named ECSet1 with an associated topology in the root design named Embedded_design, the ECSet appears as ECSet1 in the context of embedded_design. If you now add an instance of a block named dp_block that has an ECSet named ECSet1 but with a different associated topology, the ECSet on block dp_block appears as ECSet1_1 in the context of embedded_design.
4. If you want ECSets with the same names to be treated differently, rename them to have a unique name in each block. For example, you can append all ECSets from lower-level designs with a unique suffix.

Special Cases Where Rules for Constraint Merging Are Overridden

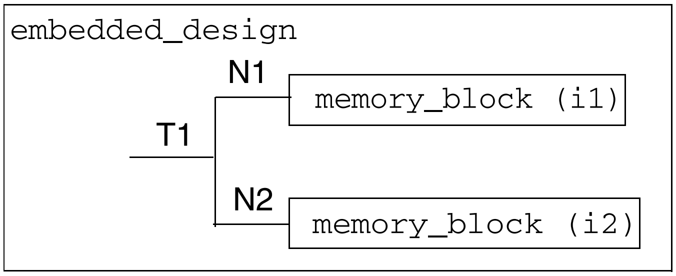
The following section describes certain situations where Constraint Manager may make a variation in processing the rules described in Rules For Merging Lower-Level Constraints on page 153.

1. If two or more nets having the same constraint with different values are aliased, the constraint on the base net wins. For example, if net p1 having the MAX_OVERSHOOT constraint value of 5100:-610 is aliased to a net sp1 which has the MAX_OVERSHOOT constraint value of 4900:-610, the MAX_OVERSHOOT constraint with the value 5100:-610 will be applied on both the nets because p1 is the base net.

2. If two or more nets are aliased and if the base net does not have the same constraint that exists on the aliased nets, the constraint on the aliased nets will be merged on the base net. If the aliased nets have the same constraint with different values, the constraint value on one of the aliased nets (on a random basis) will be merged on the base net.

3. Constraint changes made to a pin-pair in a higher-level design are ignored if the pin-pair is deleted from lower-level designs. For example, if you create a pin-pair in a lower-level design and assign constraints to it, the pin-pair and its constraint values ripple up the hierarchy. You can override the constraint values in the higher level design. If you now delete the pin-pair from the lower-level design, then the pin-pair is also deleted from the higher level designs.

4. If you have a design containing interface nets instantiated multiple times in a top-level design, the top-level design may have a base net that is connected with multiple interface nets. For example, assume that you have a design named `embedded_design`, with two instances of `memory_block`. Each instance of `memory_block` has interface nets, `NI` and `N2` connected to the net `T1` of `embedded_design`.



Constraint Manager checks whether the top-level net, `T1`, has any constraints defined. If `T1` has constraints defined on it, these constraint values win. If `T1` does not have any constraints defined on it, constraint values assigned to `NI` or `N2` win based on which design is merged first in the top-level design.

5. If you have ECSets with the same names but different constraint definitions across two levels of hierarchy, constraint differences between two ECSets may be ignored while merging at the top-level. For instance, assume that you have a design, `dp_block`, instantiated in `memory_block`. Both `dp_block` and `memory_block` designs have an ECSet named `ECSet1` with different constraint definitions. When `memory_block` is launched as the root design, Constraint Manager checks if the topology file associated with the ECSet is different. Here constraint differences between `dp_block:ECSet1` and `memory_block:ECSet1` may be ignored when the first block is added.

6. If you create a match group or differential pair in the context of the root design, the match group or differential pair is displayed under the root design and not under the lower-level design in which it was added in the context of the root design.

7. A match group or differential pair existing in a lower-level design will continue to be displayed under the lower-level design even if the interface nets that are members of the match group or differential pair are aliased to base nets in the top-level design.

8. If an ECSet exists in a lower-level design, it appears in the context of the top-level design. However, if you set the lower-level design as the root design and delete the ECSet, the ECSet will continue to appear in the context of the top-level design.

   For example, assume that you have an ECSet in the lower-level design `memory_block` added in `embedded_design`. The ECSet is displayed in the context of `embedded_design` (when set as the root design). If you now set `memory_block` as the root design, delete the ECSet, and then set `embedded_design` as the root design, the ECSet continues to be displayed in the context of `embedded_design`.

9. If two or more blocks have ECSets with the same name and if the ECSets do not have an associated topology, the constraints on the ECSets are merged in the top-level design.

10. In pre-15.5 releases, you could backannotate constraints added in lower-level designs in a top-level design by making the top-level design as the root design and running the *Tools – Constraints – Update Schematic* command in Design Entry HDL. Release 15.5 onwards, you cannot backannotate lower-level constraints in the top-level design.

Recommendations for Optimum Handling of Lower-Level Constraints

1. It is recommended that you use ECSets to capture constraints in lower-level blocks.

2. Exercise caution while deleting or changing values of lower-level constraints at top-level design as you cannot rollback the changes in lower-level constraints back to the top-level design.

3. If you have set the top-level design as root, you can override the values of constraints in lower-level designs, but cannot change values of constraints in lower-level designs directly. To make any changes to a lower-level design, make it the root design.

4. It is recommended that you make changes to constraints on interface nets only at the top-level design.

5. Always create ECSets using interface or global objects in the top-level design and not in the lower-level design.

   This is because, if you create an ECSet that has one driver and two receivers in a lower-level design using interface or global nets and then connect the interface or global net to a signal in the top-level design, the ECSet will become invalid in the context of the top-level design because it now has three receivers.

6. If you want to apply the same ECSets across different blocks, it is recommended that you apply them at the top-level design.
7. Add signal models to all the nets, which you want to connect to interface ports.
8. Always clean up obsolete objects (signals and buses) in a lower-level design before merging it in a higher level design.

Handling Constraints in Hierarchical Designs

This section contains the following topics:

- Handling Differential Pairs in Hierarchical Designs

Handling Differential Pairs in Hierarchical Designs

In hierarchical designs, the `DIFFERENTIAL_PAIR` property on a lower-level design is updated in the context of the top-level design. The constraint is not backannotated to the lower-level design on the schematic and remains in Constraint Manager.

Migration requirement

To ensure that the lower-level constraints of a design created in previous releases are read into top-level designs in the current release, save the old design in the current release.

8

Restoring Constraints from Definition

Overview

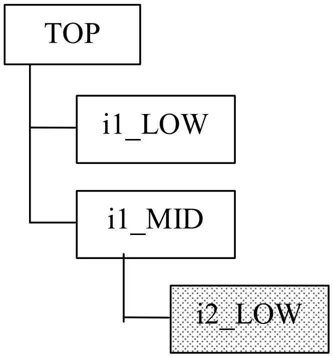
The *restore from definition* functionality enables you to restore constraint values in a hierarchical design from a lower-level (schematic) block. This functionality is helpful when you override constraint values in the context of a top-level design and want to revert to the original value stored in the lower-level block.

The restore from definition functionality restores values for objects from the immediate child block. Therefore, the object whose value is to be restored must be under the active design.

Note: Restore from definition is not supported for Component and Pin properties.

Example

Consider the example of a hierarchical design with TOP, MID, and LOW blocks, each representing its place value in the hierarchy.



In this example, restoring *NET1* in *i2_LOW* restores the value from the design *MID*, which represents the next level of hierarchy. However, if the design *TOP* is active, you cannot restore *i2_LOW* with constraint information directly from *LOW*. If design *TOP* is active, restoring *i2_LOW* would restore the values from design *MID*. To restore *i2_LOW* from its

LOW definition overridden in *MID*, you must open *MID* as the root design and restore the instance of *LOW* in *MID*.

This section covers:

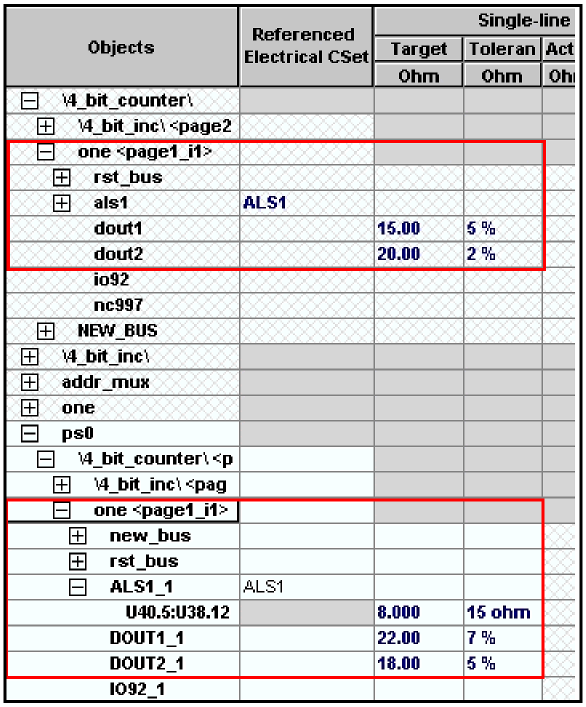
- Restoring a Constraint from its Definition
- Use Models
- Restoring Objects from their Definitions
- Other Cases

Restoring a Constraint from its Definition

Note: All the examples in this section are taken from a design named *ps0*, which includes, among other components and blocks, two instances of a schematic block, *one*.

To restore constraints on a net from its original definition, perform the following steps:

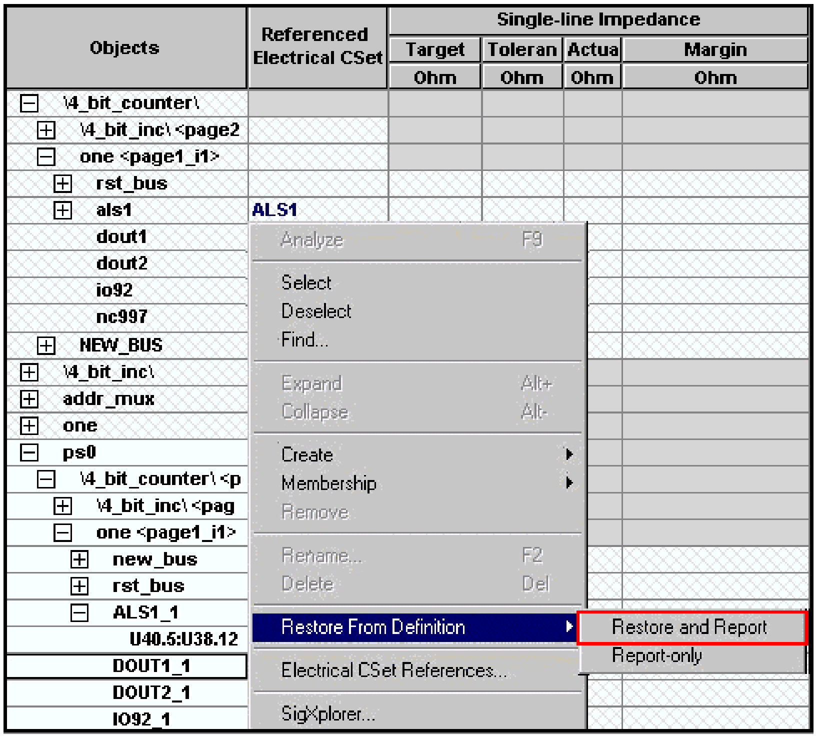
1. Select the appropriate workbook under the Net worksheet in Constraint Manager.



Objects	Referenced Electrical CSet	Single-line		
		Target	Toleran	Act
		Ohm	Ohm	Oh
⊟ \4_bit_counter\				
⊞ \4_bit_inc\ <page2				
⊟ one <page1_i1>				
⊞ rst_bus				
⊞ als1	ALS1			
dout1		15.00	5 %	
dout2		20.00	2 %	
io92				
nc997				
⊞ NEW_BUS				
⊞ \4_bit_inc\				
⊞ addr_mux				
⊞ one				
⊟ ps0				
⊟ \4_bit_counter\ <p				
⊞ \4_bit_inc\ <pag				
⊟ one <page1_i1>				
⊞ new_bus				
⊞ rst_bus				
⊟ ALS1_1	ALS1			
U40.5:U38.12		8.000	15 ohm	
DOUT1_1		22.00	7 %	
DOUT2_1		18.00	5 %	
IO92_1				

Figure 8-1 Lower-level constraints from the block *one* are overridden in the context of the top-level design *ps0*.

2. Right-click on the object or the constraint whose value you want to restore from the child block. In the example shown above, we will restore the constraints on the net *DOUT1* in the block *\4_BIT_COUNTER\ONE* under the design *ps0*.



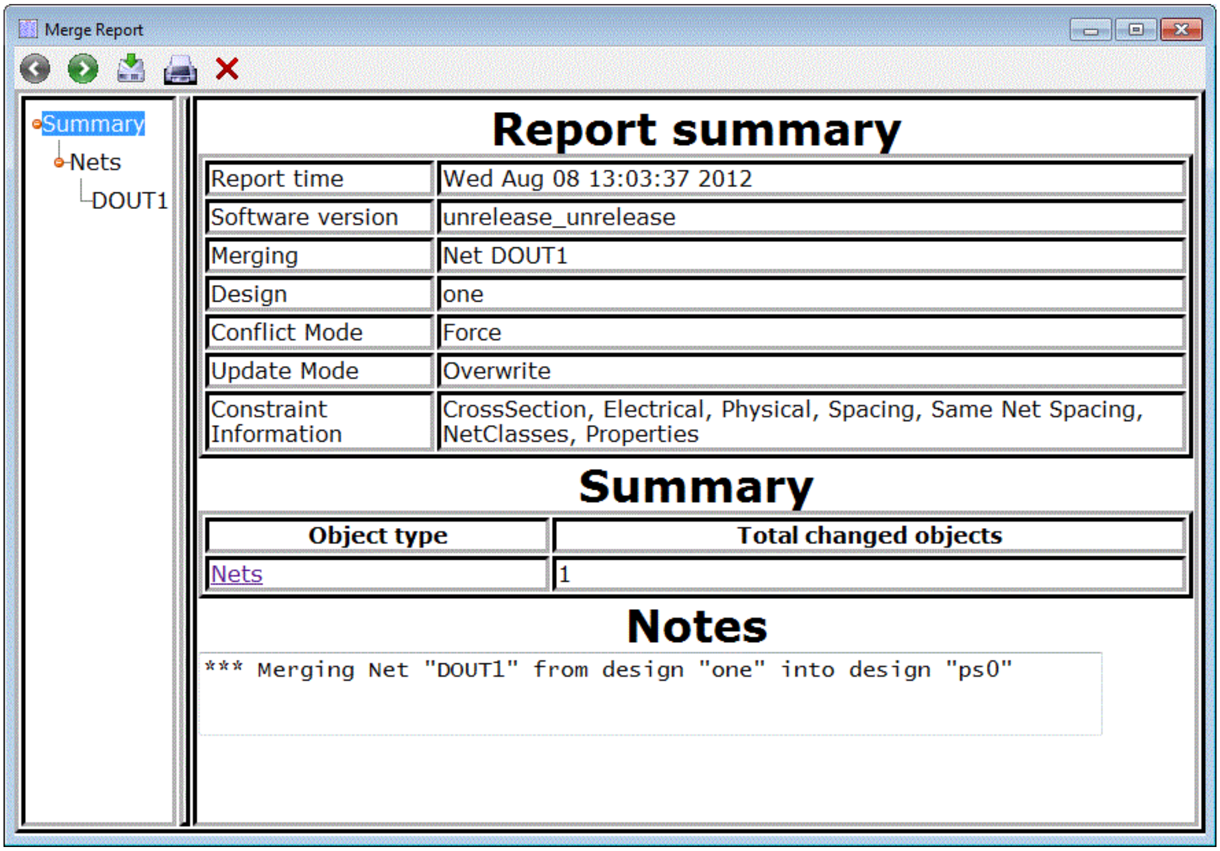
Objects	Referenced Electrical CSet	Single-line Impedance			
		Target	Toleran	Actua	Margin
		Ohm	Ohm	Ohm	Ohm
⊟ \4_bit_counter\					
⊞ \4_bit_inc\ <page2					
⊟ one <page1_i1>					
⊞ rst_bus					
⊞ als1	ALS1				
dout1					
dout2					
io92					
nc997					
⊞ NEW_BUS					
⊞ \4_bit_inc\					
⊞ addr_mux					
⊞ one					
⊟ ps0					
⊟ \4_bit_counter\ <p					
⊞ \4_bit_inc\ <pag					
⊟ one <page1_i1>					
⊞ new_bus					
⊞ rst_bus					
⊟ ALS1_1					
U40.5:U38.12					
DOUT1_1					
DOUT2_1					
IO92_1					

Figure 8-2 The Restore From Definition option

3. Choose *Restore From Definition > Restore and Report*.

   Note: If you want to view a report of the differences between the original definition and the overridden values without restoring the values, you can choose *Restore From Definition > Report only*.

The Merge Report is displayed summarizing the changes that have taken place on restoring the values.



Report summary

Report time	Wed Aug 08 13:03:37 2012
Software version	unrelease_unrelease
Merging	Net DOUT1
Design	one
Conflict Mode	Force
Update Mode	Overwrite
Constraint Information	CrossSection, Electrical, Physical, Spacing, Same Net Spacing, NetClasses, Properties

Summary

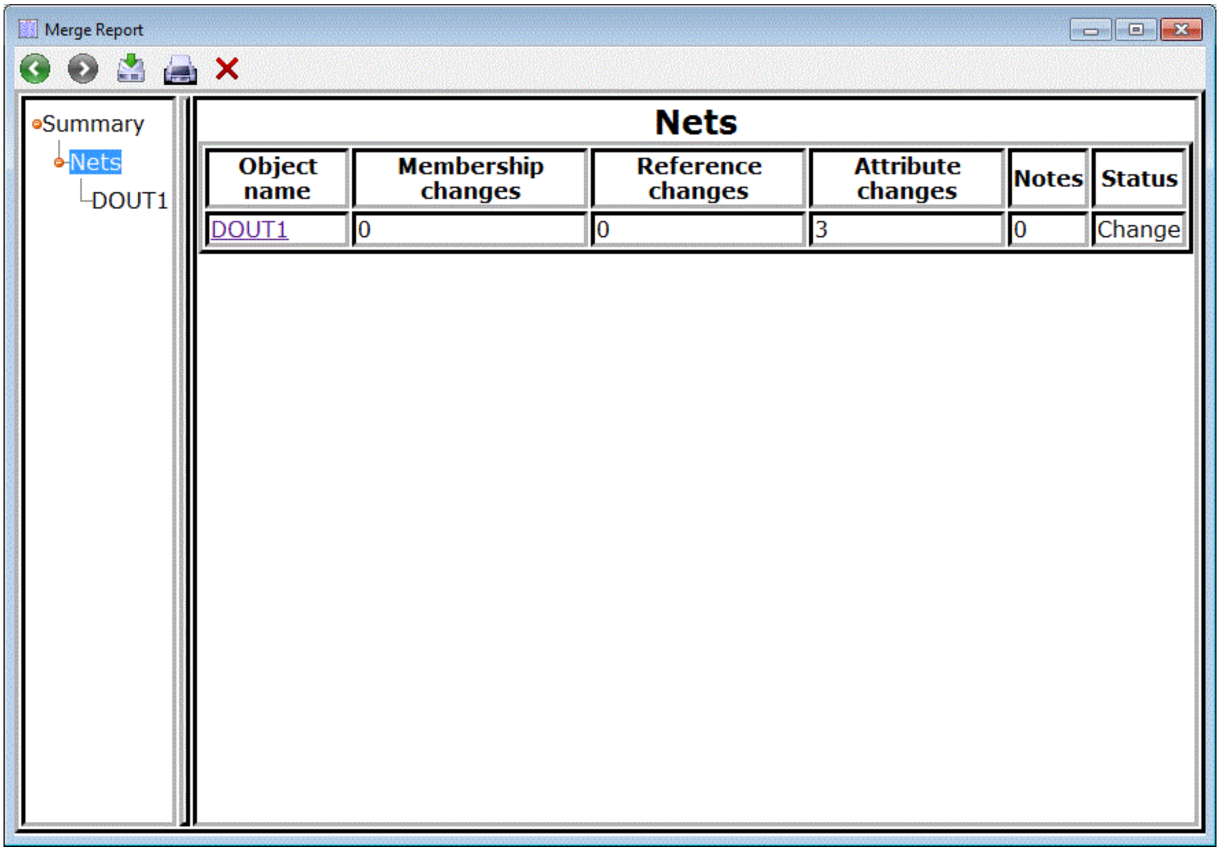
Object type	Total changed objects
Nets	1

Notes

```
*** Merging Net "DOUT1" from design "one" into design "ps0"
```

Figure 8-3 The Merge Report window

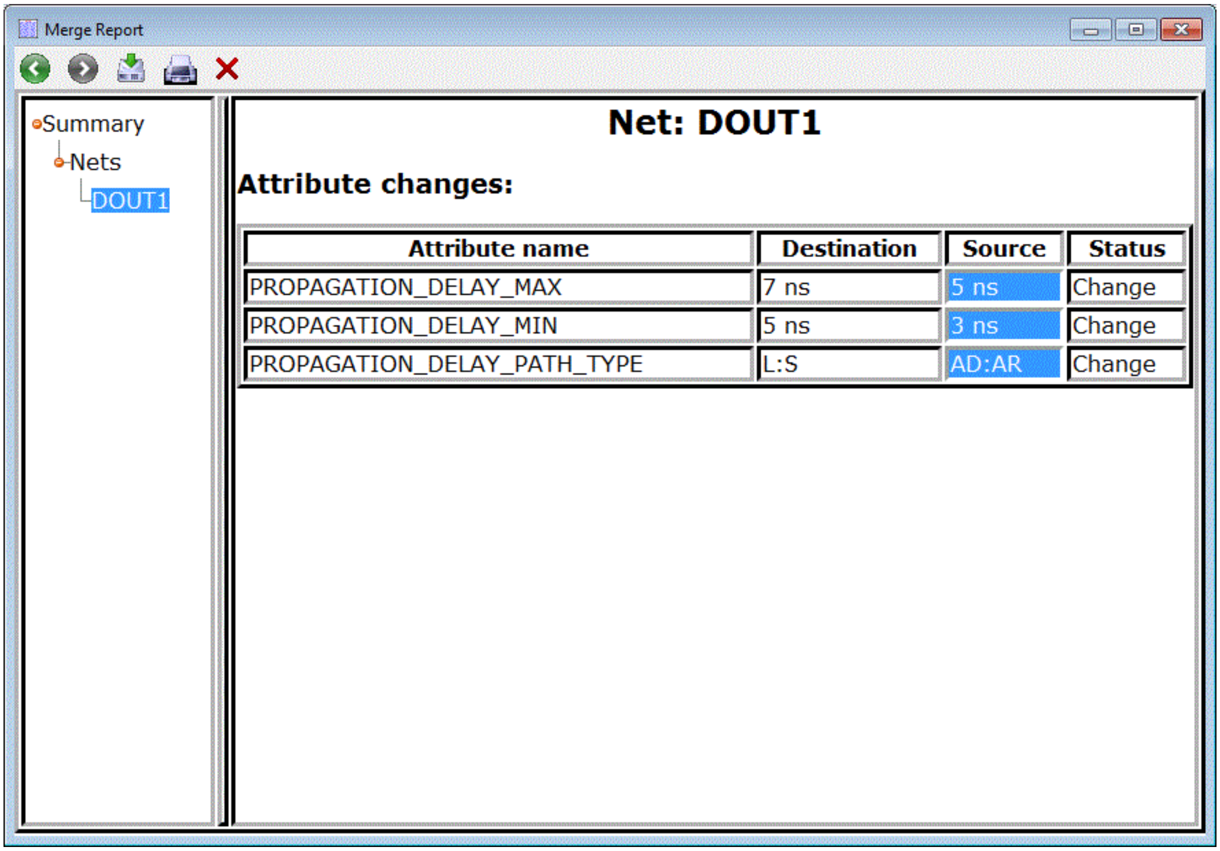
You can click *Nets* to see the detailed report



Nets

Object name	Membership changes	Reference changes	Attribute changes	Notes	Status
DOUT1	0	0	3	0	Change

Click DOUT1 to see the detailed report.



Net: DOUT1

Attribute changes:

Attribute name	Destination	Source	Status
PROPAGATION_DELAY_MAX	7 ns	5 ns	Change
PROPAGATION_DELAY_MIN	5 ns	3 ns	Change
PROPAGATION_DELAY_PATH_TYPE	L:S	AD:AR	Change

For more information on Generating and Viewing constraint differences refer to the *Allegro Constraint Manager User Guide*.

Use Models

Restore All the Constraints on an Object

As shown in the example, performing the restore from definition operation on an object restores all the constraints set on that object.

Restore a Hierarchical Block

Performing the restore from definition operation on a hierarchical block restores all the net level constraints in the block and properties from the child block.

Restore a Specific Constraint in Constraint Manager

You can also restore a specific constraint from the child block.

Interface nets, global buses, or XNets cannot be restored. However, pin pairs for these objects may be restored.

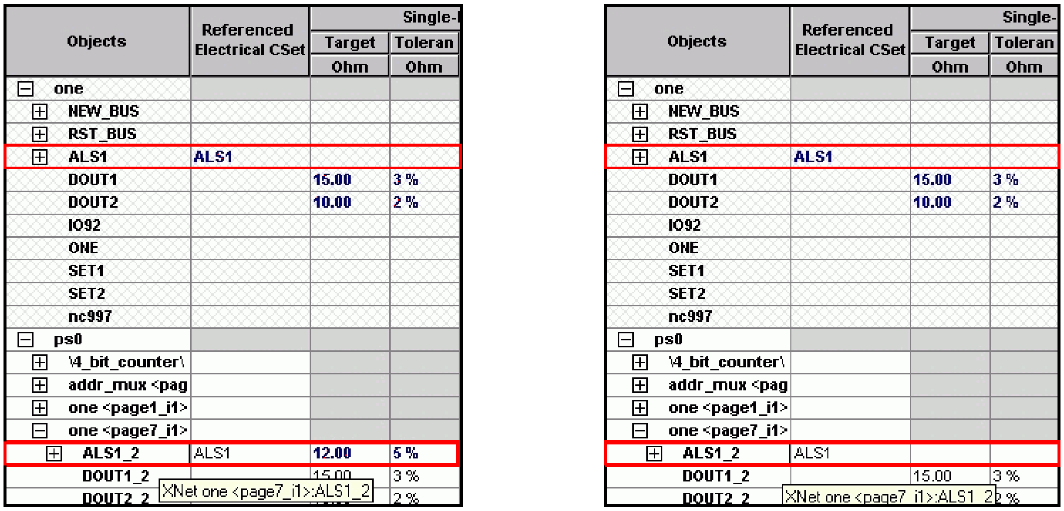
Note: Restoring constraints may cause violation in the Min:Max Propagation delay limits. For example, if the Min:Max Propagation delay limits on a lower-level block is set to 10:20 and in the root-design the constraint definition is overridden, such that Min:Max Propagation delay limits for the nets is set to 3:4. If you now restore the minimum propagation delay from the definition, the Min:Max limits will be modified to an unrealistic figure of 10:4.

Restoring Objects from their Definitions

This section shows the effect of performing the restore from definition operation on various objects in Constraint Manager. XNets/nets, differential pairs, bus-level and user-defined constraints, and pin pair and non-pin pair constraints are restored from their child blocks on restoring from the definition.

- Restoring XNet Constraints
- Restoring Differential Pair Constraints
- Restoring Constraints in a Hierarchical Block
- Restoring Constraints in a Matched Group
- Restoring Bus-Level Constraints

Restoring XNet Constraints

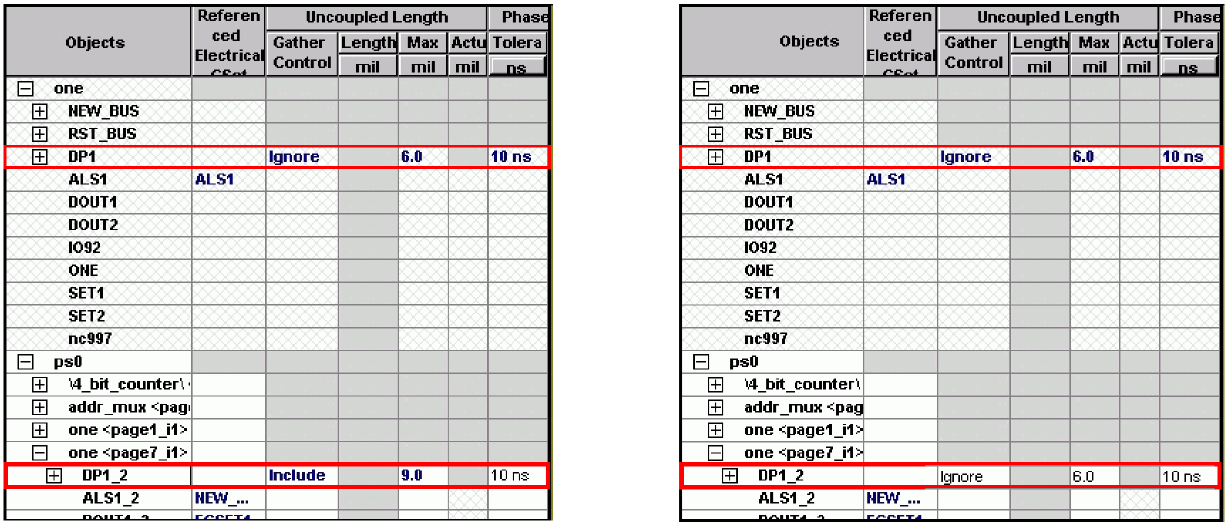


Objects	Referenced Electrical CSet	Single-I	
		Target	Toleran
		Ohm	Ohm
⊟ one			
⊞ NEW_BUS			
⊞ RST_BUS			
⊞ ALS1	ALS1		
DOUT1		15.00	3 %
DOUT2		10.00	2 %
IO92			
ONE			
SET1			
SET2			
nc997			
⊟ ps0			
⊞ \4_bit_counter\			
⊞ addr_mux <pag			
⊞ one <page1_i1>			
⊟ one <page7_i1>			
⊞ ALS1_2	ALS1	12.00	5 %
DOUT1_2		15.00	3 %
DOUT2_2	XNet one <page7_i1>:ALS1_2		2 %

Objects	Referenced Electrical CSet	Single-	
		Target	Toleran
		Ohm	Ohm
⊟ one			
⊞ NEW_BUS			
⊞ RST_BUS			
⊞ ALS1	ALS1		
DOUT1		15.00	3 %
DOUT2		10.00	2 %
IO92			
ONE			
SET1			
SET2			
nc997			
⊟ ps0			
⊞ \4_bit_counter\			
⊞ addr_mux <pag			
⊞ one <page1_i1>			
⊟ one <page7_i1>			
⊞ ALS1_2	ALS1		
DOUT1_2		15.00	3 %
DOUT2_2	XNet one <page7_i1>:ALS1_2		2 %

Figure 8-4 XNet ALS1_2 before and after restoring the constraints from the child block *one*

Restoring Differential Pair Constraints



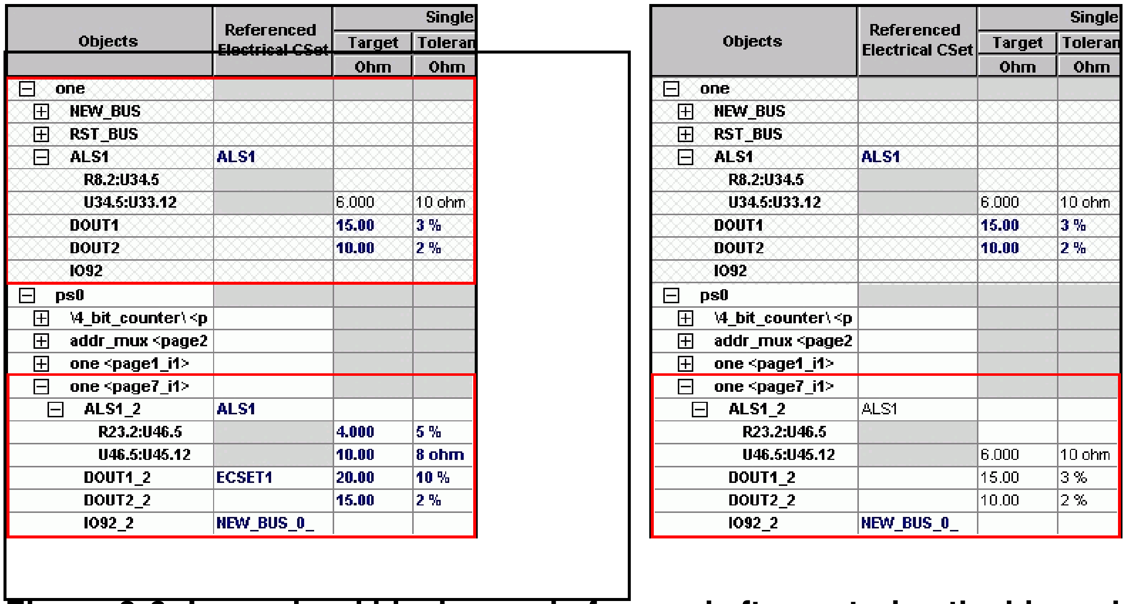
Objects	Referenced Electrical CSet	Uncoupled Length				Phase
		Gather Control	Length	Max	Actu	Tolera
			mil	mil	mil	ns
⊟ one						
⊞ NEW_BUS						
⊞ RST_BUS						
⊞ DP1		Ignore		6.0		10 ns
ALS1	ALS1					
DOUT1						
DOUT2						
IO92						
ONE						
SET1						
SET2						
nc997						
⊟ ps0						
⊞ \4_bit_counter\						
⊞ addr_mux <pag						
⊞ one <page1_i1>						
⊟ one <page7_i1>						
⊞ DP1_2		Include		9.0		10 ns
ALS1_2	NEW_...					
DOUT1_2	ECSET1					

Objects	Referenced Electrical CSet	Uncoupled Length				Phase
		Gather Control	Length	Max	Actu	Tolera
			mil	mil	mil	ns
⊟ one						
⊞ NEW_BUS						
⊞ RST_BUS						
⊞ DP1		Ignore		6.0		10 ns
ALS1	ALS1					
DOUT1						
DOUT2						
IO92						
ONE						
SET1						
SET2						
nc997						
⊟ ps0						
⊞ \4_bit_counter\						
⊞ addr_mux <pag						
⊞ one <page1_i1>						
⊟ one <page7_i1>						
⊞ DP1_2		Ignore		6.0		10 ns
ALS1_2	NEW_...					
DOUT1_2	ECSET1					

Figure 8-5 Differential pair *DP1_2* before and after restoring the constraints from the child block, *one*

Restoring Constraints in a Hierarchical Block

When you perform the restore from definition operation on a hierarchical block, all the constraints and the objects are restored. This includes all the objects and constraints added, modified, or deleted in the top-level block.



Objects	Referenced Electrical CSet	Single	
		Target	Toleran
		Ohm	Ohm
one			
NEW_BUS			
RST_BUS			
ALS1	ALS1		
R8.2:U34.5			
U34.5:U33.12		6.000	10 ohm
DOUT1		15.00	3 %
DOUT2		10.00	2 %
IO92			
ps0			
\4_bit_counter\ <p			
addr_mux <page2			
one <page1_i1>			
one <page7_i1>			
ALS1_2	ALS1		
R23.2:U46.5		4.000	5 %
U46.5:U45.12		10.00	8 ohm
DOUT1_2	ECSET1	20.00	10 %
DOUT2_2		15.00	2 %
IO92_2	NEW_BUS_0_		

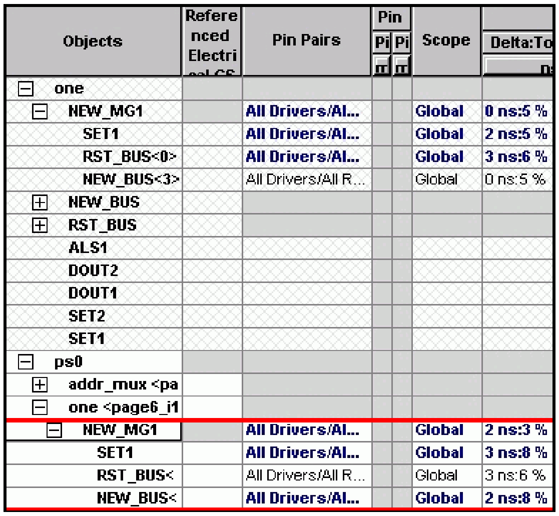
Objects	Referenced Electrical CSet	Single	
		Target	Toleran
		Ohm	Ohm
one			
NEW_BUS			
RST_BUS			
ALS1	ALS1		
R8.2:U34.5			
U34.5:U33.12		6.000	10 ohm
DOUT1		15.00	3 %
DOUT2		10.00	2 %
IO92			
ps0			
\4_bit_counter\ <p			
addr_mux <page2			
one <page1_i1>			
one <page7_i1>			
ALS1_2	ALS1		
R23.2:U46.5			
U46.5:U45.12		6.000	10 ohm
DOUT1_2		15.00	3 %
DOUT2_2		10.00	2 %
IO92_2	NEW_BUS_0_		

Figure 8-6 Lower-level block, *one*, before and after restoring the hierarchical block from its definition

Note: When you restore lower-level constraints on a Design Entry HDL block, a number of false pin-pair messages are reported. This is because, restoring lower-level constraints recreates pin-pairs.

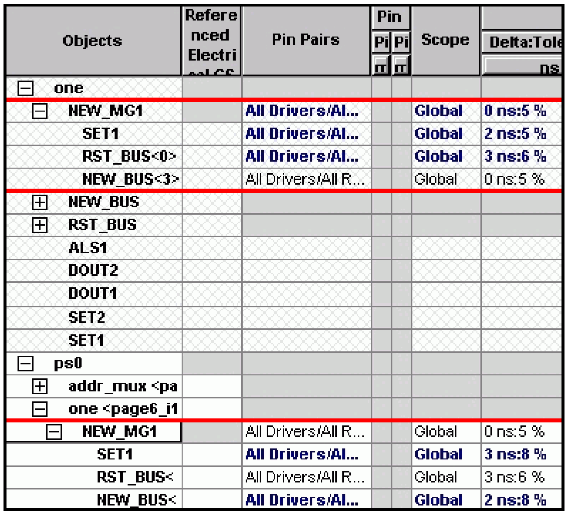
Restoring Constraints in a Matched Group

The Relative Propagation Delay constraint value and the members are restored. In our example, if the matched group member nets were changed in the context of the top-level design, *ps0*, performing the restore operation on the matched group object would restore the original member nets from the child block. However, if there are constraint overrides on the member nets set in the context of *ps0*, the restore operation on the matched group object will preserve the overridden net/XNet constraints.



Objects	Refere nced Electri	Pin Pairs	Pin Pi	 Pi	Scope	Delta:To ns
⊟ one						
⊟ NEW_MG1		All Drivers/Al...			Global	0 ns:5 %
SET1		All Drivers/Al...			Global	2 ns:5 %
RST_BUS<0>		All Drivers/Al...			Global	3 ns:6 %
NEW_BUS<3>		All Drivers/All R...			Global	0 ns:5 %
⊞ NEW_BUS						
⊞ RST_BUS						
ALS1						
DOUT2						
DOUT1						
SET2						
SET1						
⊟ ps0						
⊞ addr_mux <pa						
⊟ one <page6_i1						
⊟ NEW_MG1		All Drivers/Al...			Global	2 ns:3 %
SET1		All Drivers/Al...			Global	3 ns:8 %
RST_BUS<		All Drivers/All R...			Global	3 ns:6 %
NEW_BUS<		All Drivers/Al...			Global	2 ns:8 %

Figure 8-7 Matched group NEW_MG1 with overridden constraints at the matched group- and member net- level

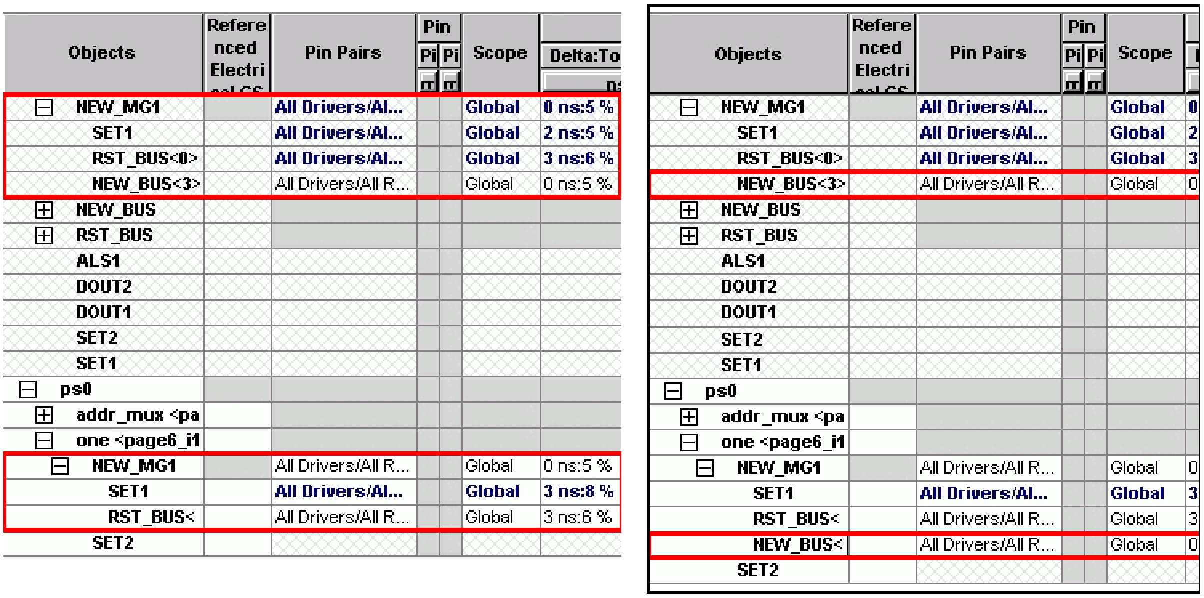


Objects	Refere nced Electri	Pin Pairs	Pin Pi	 Pi	Scope	Delta:Tol ns
⊟ one						
⊟ NEW_MG1		All Drivers/Al...			Global	0 ns:5 %
SET1		All Drivers/Al...			Global	2 ns:5 %
RST_BUS<0>		All Drivers/Al...			Global	3 ns:6 %
NEW_BUS<3>		All Drivers/All R...			Global	0 ns:5 %
⊞ NEW_BUS						
⊞ RST_BUS						
ALS1						
DOUT2						
DOUT1						
SET2						
SET1						
⊟ ps0						
⊞ addr_mux <pa						
⊟ one <page6_i1						
⊟ NEW_MG1		All Drivers/All R...			Global	0 ns:5 %
SET1		All Drivers/Al...			Global	3 ns:8 %
RST_BUS<		All Drivers/All R...			Global	3 ns:6 %
NEW_BUS<		All Drivers/Al...			Global	2 ns:8 %

Figure 8-8 Matched group-level constraints of *NEW_MG1* after restoring

Note: If you want to restore a specific object or a member net in a matched group, then select the object and perform the restore operation on the object separately. Remember, restoring the object restores all the constraint information for the object, not just the constraints which are displayed in the active worksheet. If you do not want to restore all the constraints, perform the restore operation on the specific cell(s) that are overridden.

Similarly, if a matched group contains a net/XNet/pin pair and the object is removed in the context of the top-level design, on running the restore operation, the object is restored in the matched group in the top-level design. For all the newly added objects, the restore should be run explicitly on the objects to ensure that their relative propagation delay constraints reflect what is stored in the definition. If you do not restore the objects, they will inherit the matched group-level values.



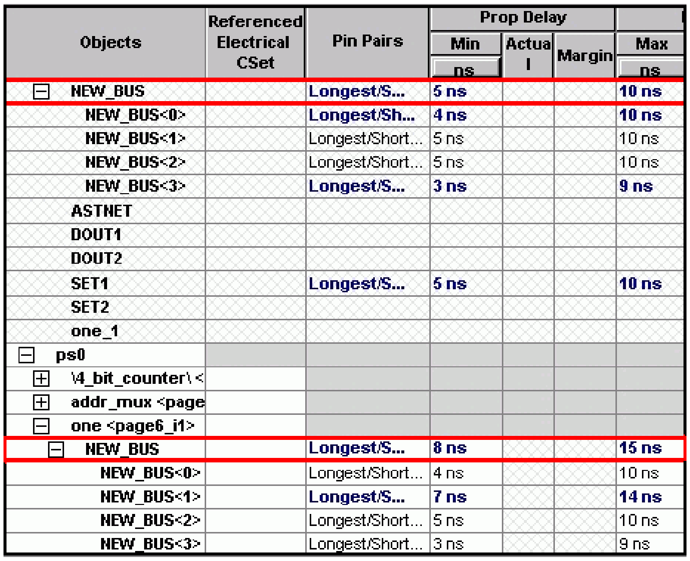
Objects	Referenced Electri	Pin Pairs	Pin Pi	Pi	Scope	Delta:To
⊟ NEW_MG1		All Drivers/Al...			Global	0 ns:5 %
SET1		All Drivers/Al...			Global	2 ns:5 %
RST_BUS<0>		All Drivers/Al...			Global	3 ns:6 %
NEW_BUS<3>		All Drivers/All R...			Global	0 ns:5 %
⊞ NEW_BUS						
⊞ RST_BUS						
ALS1						
DOUT2						
DOUT1						
SET2						
SET1						
⊟ ps0						
⊞ addr_mux <pa						
⊟ one <page6_i1						
⊟ NEW_MG1		All Drivers/All R...			Global	0 ns:5 %
SET1		All Drivers/Al...			Global	3 ns:8 %
RST_BUS<		All Drivers/All R...			Global	3 ns:6 %
SET2						

Objects	Referenced Electri	Pin Pairs	Pin Pi	Pi	Scope	
⊟ NEW_MG1		All Drivers/Al...			Global	0
SET1		All Drivers/Al...			Global	2
RST_BUS<0>		All Drivers/Al...			Global	3
NEW_BUS<3>		All Drivers/All R...			Global	0
⊞ NEW_BUS						
⊞ RST_BUS						
ALS1						
DOUT2						
DOUT1						
SET2						
SET1						
⊟ ps0						
⊞ addr_mux <pa						
⊟ one <page6_i1						
⊟ NEW_MG1		All Drivers/All R...			Global	0
SET1		All Drivers/Al...			Global	3
RST_BUS<		All Drivers/All R...			Global	3
NEW_BUS<		All Drivers/All R...			Global	0
SET2						

Figure 8-9 *NEW_BUS<3>*deleted from the matched group *NEW_MG1* in the context of the top-level design *ps0* is restored after performing the restore operation on *NEW_MG1*

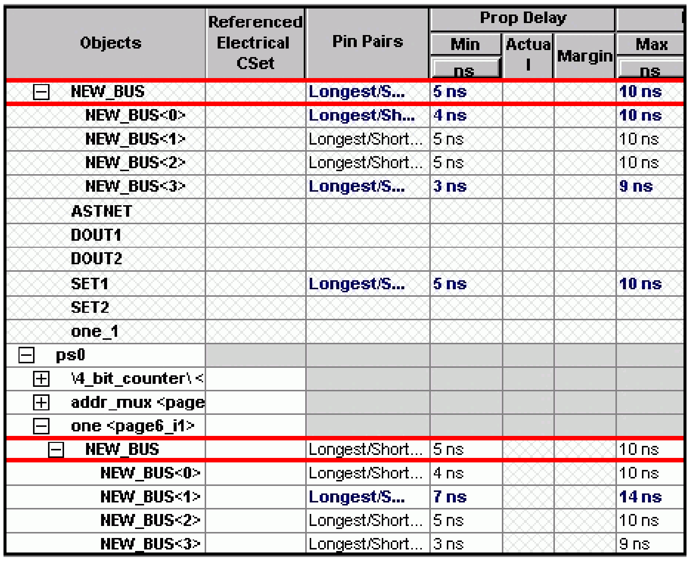
Restoring Bus-Level Constraints

A restore operation performed on a bus restores only bus-level constraints.



Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay			
			Min ns	Actual	Margin	Max ns
⊟ NEW_BUS		Longest/S...	5 ns			10 ns
NEW_BUS<0>		Longest/Sh...	4 ns			10 ns
NEW_BUS<1>		Longest/Short...	5 ns			10 ns
NEW_BUS<2>		Longest/Short...	5 ns			10 ns
NEW_BUS<3>		Longest/S...	3 ns			9 ns
ASTNET						
DOUT1						
DOUT2						
SET1		Longest/S...	5 ns			10 ns
SET2						
one_1						
⊟ ps0						
⊞ \4_bit_counter\ <						
⊞ addr_mux <page						
⊟ one <page6_i1>						
⊟ NEW_BUS		Longest/S...	8 ns			15 ns
NEW_BUS<0>		Longest/Short...	4 ns			10 ns
NEW_BUS<1>		Longest/S...	7 ns			14 ns
NEW_BUS<2>		Longest/Short...	5 ns			10 ns
NEW_BUS<3>		Longest/Short...	3 ns			9 ns

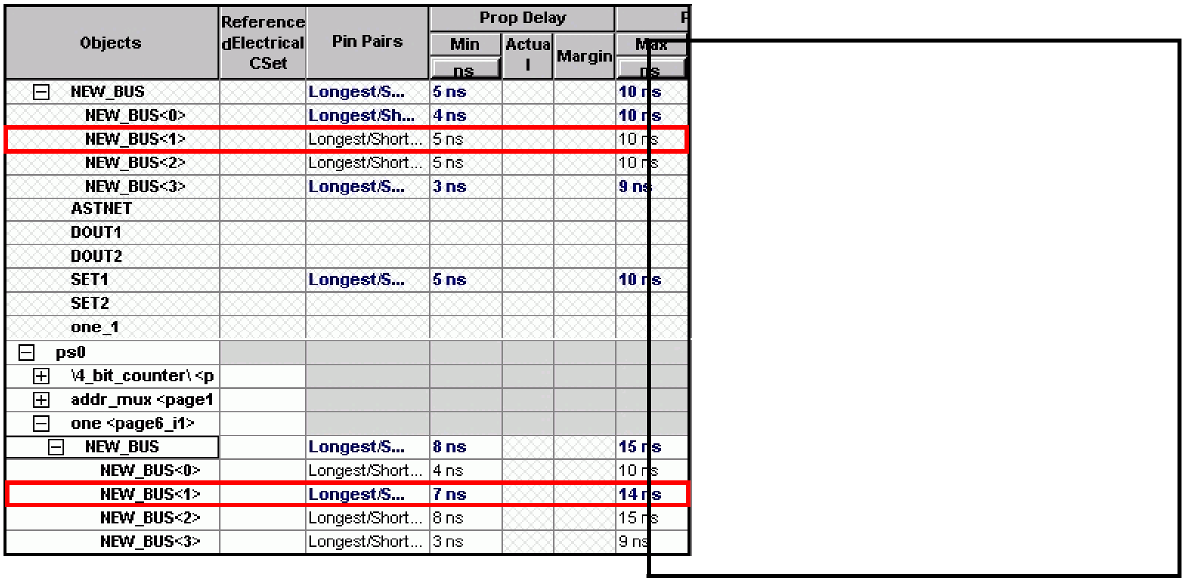
Figure 8-10 Bus-level and member-net level constraints of the bus *NEW_BUS* overridden in *ps0*



Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay			
			Min ns	Actual	Margin	Max ns
⊟ NEW_BUS		Longest/S...	5 ns			10 ns
NEW_BUS<0>		Longest/Sh...	4 ns			10 ns
NEW_BUS<1>		Longest/Short...	5 ns			10 ns
NEW_BUS<2>		Longest/Short...	5 ns			10 ns
NEW_BUS<3>		Longest/S...	3 ns			9 ns
ASTNET						
DOUT1						
DOUT2						
SET1		Longest/S...	5 ns			10 ns
SET2						
one_1						
⊟ ps0						
⊞ \4_bit_counter\ <						
⊞ addr_mux <page						
⊟ one <page6_i1>						
⊟ NEW_BUS		Longest/Short...	5 ns			10 ns
NEW_BUS<0>		Longest/Short...	4 ns			10 ns
NEW_BUS<1>		Longest/S...	7 ns			14 ns
NEW_BUS<2>		Longest/Short...	5 ns			10 ns
NEW_BUS<3>		Longest/Short...	3 ns			9 ns

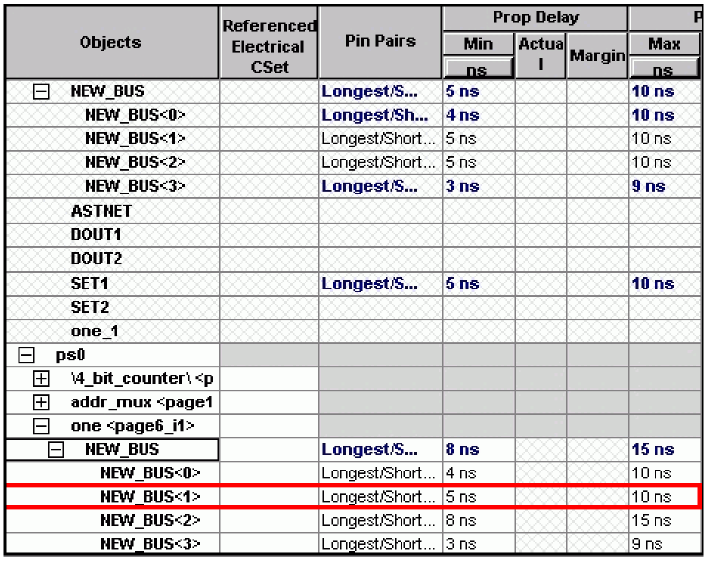
Figure 8-11 Bus-level constraints are restored

Note: To restore constraints on bus bits or member nets, you need to restore the constraints on the member net.



Objects	Reference dElectrical CSet	Pin Pairs	Prop Delay			P
			Min	Actual	Margin	Max
			ns			ns
⊟ NEW_BUS		Longest/S...	5 ns			10 ns
NEW_BUS<0>		Longest/Sh...	4 ns			10 ns
NEW_BUS<1>		Longest/Short...	5 ns			10 ns
NEW_BUS<2>		Longest/Short...	5 ns			10 ns
NEW_BUS<3>		Longest/S...	3 ns			9 ns
ASTNET						
DOUT1						
DOUT2						
SET1		Longest/S...	5 ns			10 ns
SET2						
one_1						
⊟ ps0						
⊞ \4_bit_counter\<p						
⊞ addr_mux <page1						
⊟ one <page6_i1>						
⊟ NEW_BUS		Longest/S...	8 ns			15 ns
NEW_BUS<0>		Longest/Short...	4 ns			10 ns
NEW_BUS<1>		Longest/S...	7 ns			14 ns
NEW_BUS<2>		Longest/Short...	8 ns			15 ns
NEW_BUS<3>		Longest/Short...	3 ns			9 ns

Figure 8-12 Member-net level constraints overridden in *ps0*



Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay			P
			Min	Actual	Margin	Max
			ns			ns
⊟ NEW_BUS		Longest/S...	5 ns			10 ns
NEW_BUS<0>		Longest/Sh...	4 ns			10 ns
NEW_BUS<1>		Longest/Short...	5 ns			10 ns
NEW_BUS<2>		Longest/Short...	5 ns			10 ns
NEW_BUS<3>		Longest/S...	3 ns			9 ns
ASTNET						
DOUT1						
DOUT2						
SET1		Longest/S...	5 ns			10 ns
SET2						
one_1						
⊟ ps0						
⊞ \4_bit_counter\<p						
⊞ addr_mux <page1						
⊟ one <page6_i1>						
⊟ NEW_BUS		Longest/S...	8 ns			15 ns
NEW_BUS<0>		Longest/Short...	4 ns			10 ns
NEW_BUS<1>		Longest/Short...	5 ns			10 ns
NEW_BUS<2>		Longest/Short...	8 ns			15 ns
NEW_BUS<3>		Longest/Short...	3 ns			9 ns

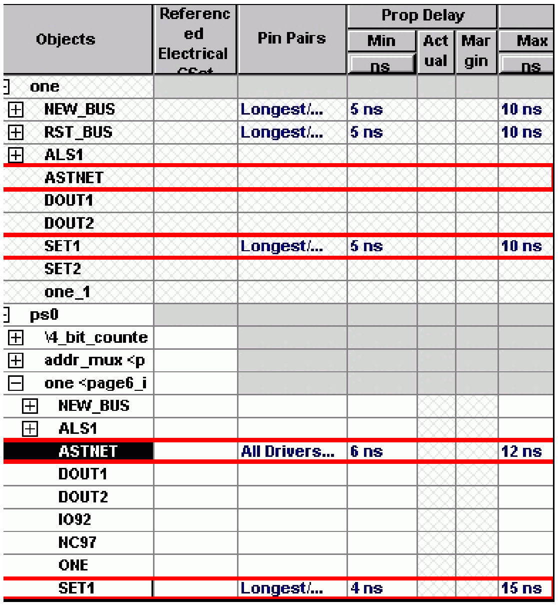
Figure 8-13 Constraints restored for the member net *NEW_BUS<1>* **after performing the restore operation on it.**

Other Cases

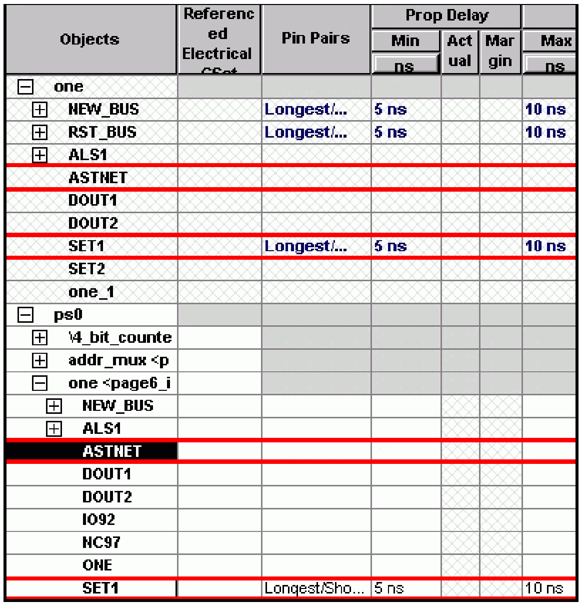
- Restoring Multiple Selections
- Restoring Specific Constraint Value
- Restoring Deleted Objects
- Restoring Renamed Objects
- Restoring in Replicated Blocks
- Restoring Electrical Constraints Sets

Restoring Multiple Selections

In case of multiple selections, all the selected objects are restored.



Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay			
			Min ns	Actual	Margin	Max ns
⊟ one						
⊞ NEW_BUS		Longest/...	5 ns			10 ns
⊞ RST_BUS		Longest/...	5 ns			10 ns
⊞ ALS1						
ASTNET						
DOUT1						
DOUT2						
SET1		Longest/...	5 ns			10 ns
SET2						
one_1						
⊟ ps0						
⊞ \4_bit_counte						
⊞ addr_mux <p						
⊟ one <page6_i						
⊞ NEW_BUS						
⊞ ALS1						
ASTNET		All Drivers...	6 ns			12 ns
DOUT1						
DOUT2						
IO92						
NC97						
ONE						
SET1		Longest/...	4 ns			15 ns

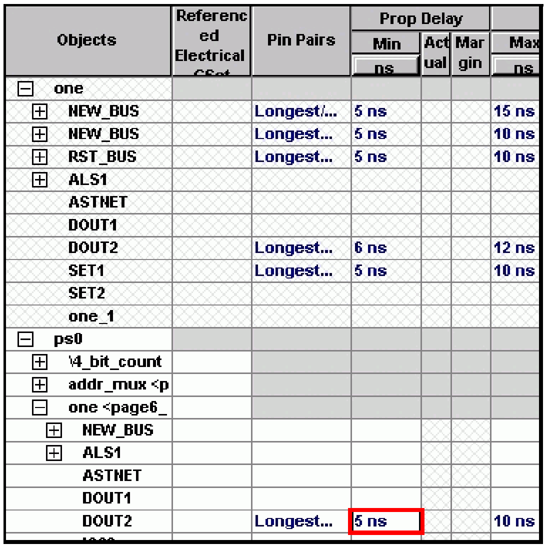


Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay			
			Min ns	Actual	Margin	Max ns
⊟ one						
⊞ NEW_BUS		Longest/...	5 ns			10 ns
⊞ RST_BUS		Longest/...	5 ns			10 ns
⊞ ALS1						
ASTNET						
DOUT1						
DOUT2						
SET1		Longest/...	5 ns			10 ns
SET2						
one_1						
⊟ ps0						
⊞ \4_bit_counte						
⊞ addr_mux <p						
⊟ one <page6_i						
⊞ NEW_BUS						
⊞ ALS1						
ASTNET						
DOUT1						
DOUT2						
IO92						
NC97						
ONE						
SET1		Longest/Sho...	5 ns			10 ns

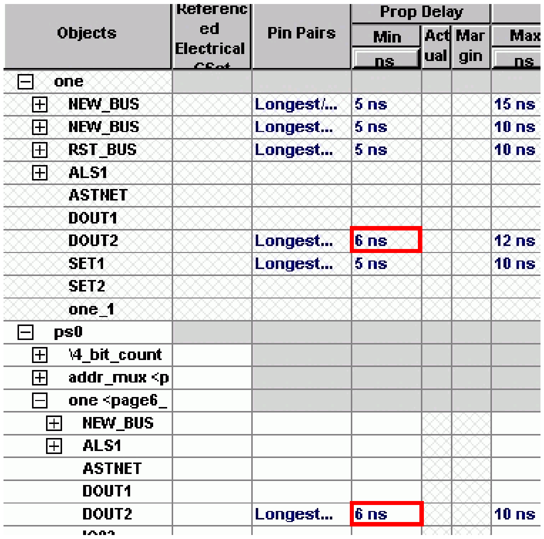
Figure 8-14 Constraints on the nets *ASTNET* and *SET1* restored from the schematic block *one*.

Restoring Specific Constraint Value

To restore a specific constraint value, select the cell and restore the value.



Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay Min	Actual	Margin	Max
			ns			ns
⊟ one						
⊞ NEW_BUS		Longest/...	5 ns			15 ns
⊞ NEW_BUS		Longest...	5 ns			10 ns
⊞ RST_BUS		Longest...	5 ns			10 ns
⊞ ALS1						
ASTNET						
DOUT1						
DOUT2		Longest...	6 ns			12 ns
SET1		Longest...	5 ns			10 ns
SET2						
one_1						
⊟ ps0						
⊞ \4_bit_count						
⊞ addr_mux <p						
⊟ one <page6_						
⊞ NEW_BUS						
⊞ ALS1						
ASTNET						
DOUT1						
DOUT2		Longest...	5 ns			10 ns



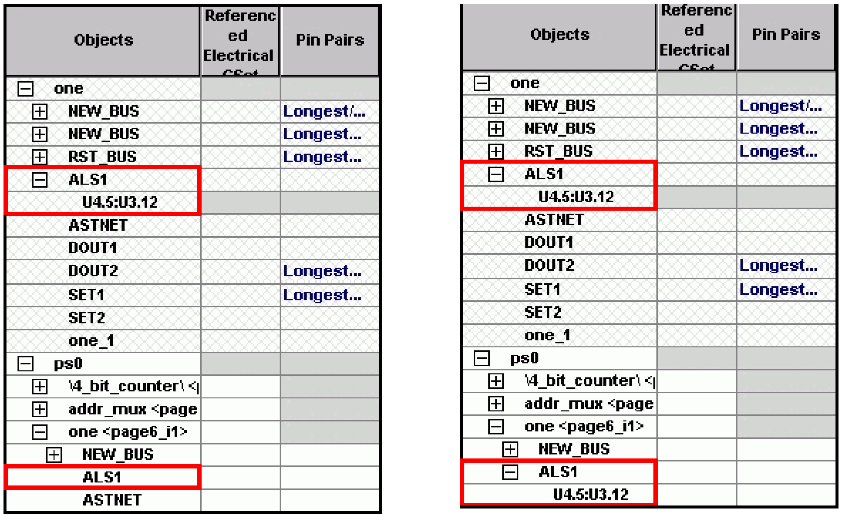
Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay Min	Actual	Margin	Max
			ns			ns
⊟ one						
⊞ NEW_BUS		Longest/...	5 ns			15 ns
⊞ NEW_BUS		Longest...	5 ns			10 ns
⊞ RST_BUS		Longest...	5 ns			10 ns
⊞ ALS1						
ASTNET						
DOUT1						
DOUT2		Longest...	6 ns			12 ns
SET1		Longest...	5 ns			10 ns
SET2						
one_1						
⊟ ps0						
⊞ \4_bit_count						
⊞ addr_mux <p						
⊟ one <page6_						
⊞ NEW_BUS						
⊞ ALS1						
ASTNET						
DOUT1						
DOUT2		Longest...	6 ns			10 ns

Figure 8-15 The Minimum Propagation Delay constraint for net *DOUT2* overridden in *ps0* is restored from the lower-level block *one*.

Note: When you perform the restore operation on a specific constraint value in the context of the top-level design, constraint values are pulled from the `.dcf` file of the lower-level block and added to the `OPF` of the top-level design.

Restoring Deleted Objects

Any lower-level object or constraint deleted in the context of the top-level design can be restored.



Objects	Referenced Electrical CSet	Pin Pairs
⊟ one		
⊞ NEW_BUS		Longest/...
⊞ NEW_BUS		Longest...
⊞ RST_BUS		Longest...
⊟ ALS1		
U4.5:U3.12		
ASTNET		
DOUT1		
DOUT2		Longest...
SET1		Longest...
SET2		
one_1		
⊟ ps0		
⊞ \4_bit_counter\ <|		
⊞ addr_mux <page		
⊟ one <page6_i1>		
⊞ NEW_BUS		
ALS1		
ASTNET		

Objects	Referenced Electrical CSet	Pin Pairs
⊟ one		
⊞ NEW_BUS		Longest/...
⊞ NEW_BUS		Longest...
⊞ RST_BUS		Longest...
⊟ ALS1		
U4.5:U3.12		
ASTNET		
DOUT1		
DOUT2		Longest...
SET1		Longest...
SET2		
one_1		
⊟ ps0		
⊞ \4_bit_counter\ <|		
⊞ addr_mux <page		
⊟ one <page6_i1>		
⊞ NEW_BUS		
⊟ ALS1		
U4.5:U3.12		

Figure 8-16 A pin pair, *U33.5:U32.12*, deleted in the context of *ps0* is restored from the lower-level block, *one*, along with constraint values

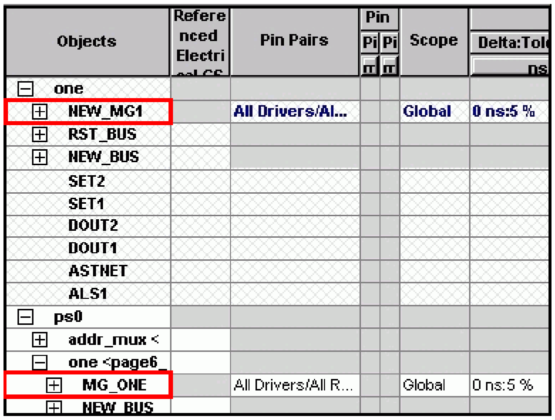
Note: Deleted objects are only re-created by restoring their parent object. When restoring the parent (for example, net *ALS1*), all the constraint information for the parent will also be restored, in addition to the pin pair(s).

The only way to restore deleted matched groups and differential pairs is to restore the block instance from which they originated. As a result, all the constraint information from the block will also be restored.

Restoring Renamed Objects

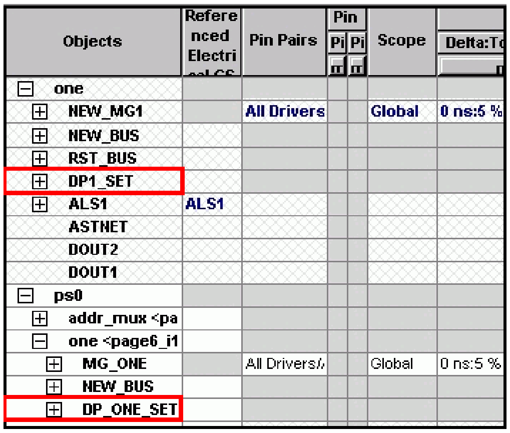
On restoring a matched group or a differential pair renamed in the context of the top-level design, the constraints are restored, but the name is preserved. Also, if you rename a matched group and perform the restore operation on it, you cannot rename it back to the original name.

Note: If you rename an ECSet generated matched group in the ECSet worksheet and restore the ECSet, the name is also restored along with constraints from the child block.



Objects	Refere nced Electri	Pin Pairs	Pin Pi	Pi	Scope	Delta:Tol ns
one						
NEW_MG1		All Drivers/Al...			Global	0 ns:5 %
RST_BUS						
NEW_BUS						
SET2						
SET1						
DOUT2						
DOUT1						
ASTNET						
ALS1						
ps0						
addr_mux <						
one <page6_						
MG_ONE		All Drivers/All R...			Global	0 ns:5 %
NEW_BUS						

Figure 8-17 Matched group *NEW_MG1* renamed as *MG_ONE* in the context of *ps0*

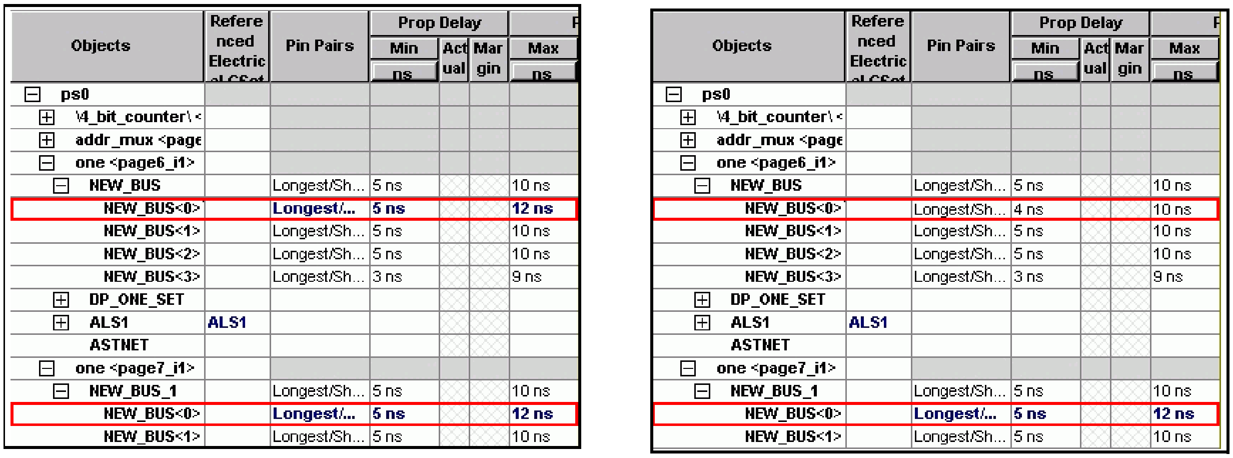


Objects	Refere nced Electri	Pin Pairs	Pin Pi	Pi	Scope	Delta:Tc
one						
NEW_MG1		All Drivers			Global	0 ns:5 %
NEW_BUS						
RST_BUS						
DP1_SET						
ALS1	ALS1					
ASTNET						
DOUT2						
DOUT1						
ps0						
addr_mux <pa						
one <page6_i1						
MG_ONE		All Drivers/			Global	0 ns:5 %
NEW_BUS						
DP_ONE_SET						

Figure 8-18 Differential pair *DP1_SET* renamed as *DP_ONE_SET* in the context of *ps0*

Restoring in Replicated Blocks

In case of replicated blocks, restoring overridden constraints on one block will restore the values only in that block and not across instances.



Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay Min ns	Actual	Margin	Max ns
⊟ ps0						
⊞ \4_bit_counter\ <						
⊞ addr_mux <page						
⊟ one <page6_i1>						
⊟ NEW_BUS		Longest/Sh...	5 ns			10 ns
NEW_BUS<0>		Longest/...	5 ns			12 ns
NEW_BUS<1>		Longest/Sh...	5 ns			10 ns
NEW_BUS<2>		Longest/Sh...	5 ns			10 ns
NEW_BUS<3>		Longest/Sh...	3 ns			9 ns
⊞ DP_ONE_SET						
⊞ ALS1	ALS1					
ASTNET						
⊟ one <page7_i1>						
⊟ NEW_BUS_1		Longest/Sh...	5 ns			10 ns
NEW_BUS<0>		Longest/...	5 ns			12 ns
NEW_BUS<1>		Longest/Sh...	5 ns			10 ns

Objects	Referenced Electrical CSet	Pin Pairs	Prop Delay Min ns	Actual	Margin	Max ns
⊟ ps0						
⊞ \4_bit_counter\ <						
⊞ addr_mux <page						
⊟ one <page6_i1>						
⊟ NEW_BUS		Longest/Sh...	5 ns			10 ns
NEW_BUS<0>		Longest/Sh...	4 ns			10 ns
NEW_BUS<1>		Longest/Sh...	5 ns			10 ns
NEW_BUS<2>		Longest/Sh...	5 ns			10 ns
NEW_BUS<3>		Longest/Sh...	3 ns			9 ns
⊞ DP_ONE_SET						
⊞ ALS1	ALS1					
ASTNET						
⊟ one <page7_i1>						
⊟ NEW_BUS_1		Longest/Sh...	5 ns			10 ns
NEW_BUS<0>		Longest/...	5 ns			12 ns
NEW_BUS<1>		Longest/Sh...	5 ns			10 ns

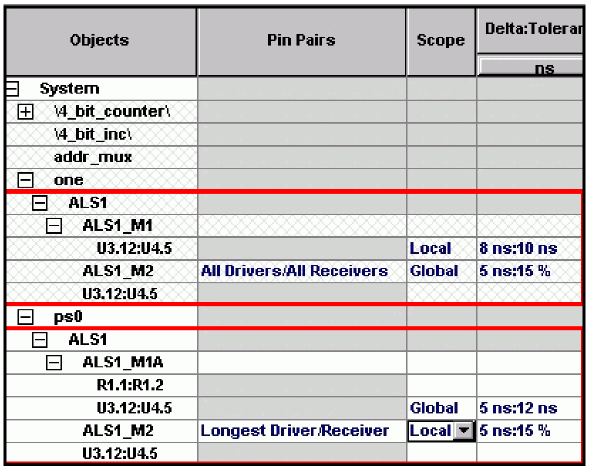
Figure 8-19 Restoring the constraint value of *NEW_BUS<0>* on `page 6` **does not restore the value on** `page 7`

Restoring Electrical Constraints Sets

On restoring an ECSet in the ECSet worksheet, the entire ECSet definition is restored, including the constraints, pin pair, and matched group information. Consider the example shown below. The ECSet *ALS1* is overridden in the context of the top-level design *ps0*:

- Matched group *ALS1_M1* is renamed as *ALS1_M1A*
- A pin pair *R1.1:R1.2* is added to the ECSet

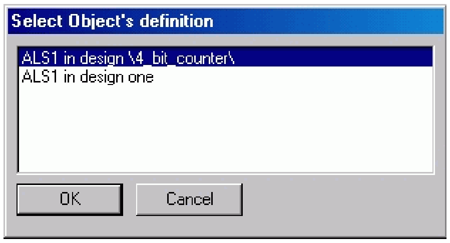
- The `Scope` and `Delta:Tolerance` values are changed for the pin pair *U3.12:U4.5*



Objects	Pin Pairs	Scope	Delta:Toleran
			ns
⊟ System			
⊞ \4_bit_counter\			
\4_bit_inc\			
addr_mux			
⊟ one			
⊟ ALS1			
⊟ ALS1_M1			
U3.12:U4.5		Local	8 ns:10 ns
ALS1_M2	All Drivers/All Receivers	Global	5 ns:15 %
U3.12:U4.5			
⊟ ps0			
⊟ ALS1			
⊟ ALS1_M1A			
R1.1:R1.2			
U3.12:U4.5		Global	5 ns:12 ns
ALS1_M2	Longest Driver/Receiver	Local	5 ns:15 %
U3.12:U4.5			

Figure 8-20 ECSet definitions overridden in *ps0*

In case of replicated blocks, if a block is instantiated at multiple levels of hierarchy, the Select Object's definition dialog box is displayed. In this dialog box, you choose the level from which you want the definition to be restored. In our example, the lower-level block *one* is instantiated in *ps0*. At the same time *one* is also instantiated in *4_bit_counter*, which is another child block of *ps0*.

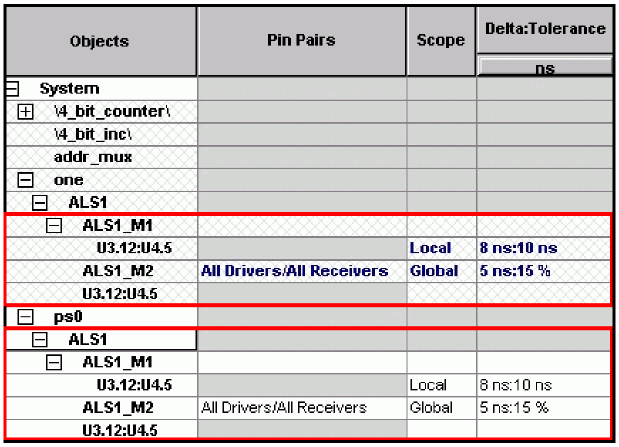


Select Object's definition

ALS1 in design \4_bit_counter\
ALS1 in design one

Figure 8-21 Select Object's definition dialog box

The ECSet definition is restored.



Objects	Pin Pairs	Scope	Delta:Tolerance
			ns
⊟ System			
⊞ \4_bit_counter\			
\4_bit_inc\			
addr_mux			
⊟ one			
⊟ ALS1			
⊟ ALS1_M1			
U3.12:U4.5		Local	8 ns:10 ns
ALS1_M2	All Drivers/All Receivers	Global	5 ns:15 %
U3.12:U4.5			
⊟ ps0			
⊟ ALS1			
⊟ ALS1_M1			
U3.12:U4.5		Local	8 ns:10 ns
ALS1_M2	All Drivers/All Receivers	Global	5 ns:15 %
U3.12:U4.5			

Figure 8-22 ECSet definition restored from the lower-level block *one*.

Note: If you restore a net that is a member of an ECSet-generated matched group and pin pair of that net is a member of another ECSet-generated matched group, restore on net in the matched group also restores the pin pair lying in the other ECSet-generated matched group.

9

Working with Properties

This chapter describes the following sections:

- Using Constraint Manager to Manage Properties on page 183.
- Procedures for Working with Properties in Constraint Manager on page 184.

Refer to the Tools – Customize command in the *Constraint Manager Reference* for more information.

Using Constraint Manager to Manage Properties

Using Constraint Manager you can add, modify, or delete properties for an object. When invoked from System Connectivity Manager, Constraint Manager, can be used to modify the properties of nets, components, and pins. To do this, select the property worksheet for the object in the Properties Tab.

For more information on opening worksheets with property information for objects, see:

- Working with Properties on Nets on page 183

The object you selected in System Connectivity Manager is highlighted in the property worksheet for the object in Constraint Manager.

Important

If Constraint Manager is launched from Design Entry HDL, only the Net property worksheet is available. Modifying component and pin properties is Constraint Manager is not support for Design Entry HDL- Constraint Manager flow.

Working with Properties on Nets

To work with properties on nets, perform the following steps.

➤ In Constraint Manager, select the *Properties* tab.

➤ Expand the *Net* object folder.

➤ Select the *General Properties* workbook to display the *General Properties* worksheet.

Procedures for Working with Properties in Constraint Manager

The following sections describe how you can work with properties in Constraint Manager:

- Adding Properties in Constraint Manager on page 184
- Editing Properties in Constraint Manager on page 185
- Working with User-Defined Properties in Constraint Manager on page 188
- Sorting Properties Values in Constraint Manager on page 185
- Managing Columns for Properties in Constraint Manager on page 186

Adding Properties in Constraint Manager

To add a property on an object, do the following:

1. Open the property worksheet for the object.
2. Locate the object on which you want to add the property.

   Note: You can perform cross probing from System Connectivity Manager to Constraint Manager to quickly locate objects on which you want to add properties in Constraint Manager.

3. Enter the value of the property in the corresponding cell in the column for the property. For example, if you want to enter the value for the ROOM property, enter the value in the corresponding cell in the ROOM column.

   Note: If the column for the property does not exist in the worksheet, you must add a column for the property in the worksheet. For more information, see Adding a Column for a Property in a Worksheet in Constraint Manager on page 185.

Note the following when working with properties in Constraint Manager:

- Nets are displayed in Constraint Manager using physical (packaged) net names.
- You cannot add properties on a bus (vectored signal) or a vectored pin. You can only add properties on the bits of a bus or on the bits of a vectored pin.

- If a net is aliased to another net or nets, only the base net is displayed in Constraint Manager. The base net inherits all the properties that exist on the nets aliased to it. A property you add on a base net also applies to the nets aliased to it.

Note: Buses with voltage property is not supported in the flow.

Editing Properties in Constraint Manager

You can quickly edit, delete, cut, copy or paste property values.

Note: Undo and redo of properties and constraints is not supported in Constraint Manager.

Tip

You can perform cross probing from System Connectivity Manager to Constraint Manager to quickly locate objects on which you want to edit properties in Constraint Manager.

Sorting Properties Values in Constraint Manager

To sort properties,

➤ Double-click on the column header of the property that you want to sort.

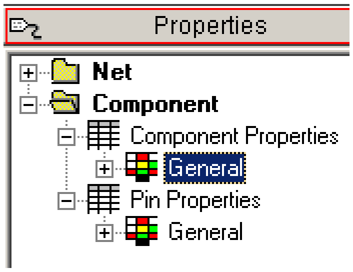
The column is sorted in the ascending or descending order.

Adding a Column for a Property in a Worksheet in Constraint Manager

To be able to add a property on an object, you must add a column for the property in Constraint Manager. For example, if you want to add a property on components in the design, you must add a column for the property.

1. In Constraint Manager, choose *Tools – Customize*.
2. Expand the workbook that contains the property worksheet for the object.

For example, if you want to add a column for a component property, expand the *Component* object folder in the properties tab, then expand the *Component Properties* workbook to display the *General* worksheet, as shown below:



3. Select the property worksheet for the object, right-click and choose *Add Column*.

   The *Add Column* dialog box appears.

4. Depending on whether you want to add a column for pre-defined or user-defined property, do one of the following:

   ❑ To add a column for a predefined property, from the *Type* drop-down list choose *Pre-defined*. The list of predefined properties are displayed.

   ❑ To add a column for a user defined property, click the *Type* drop-down list and choose *User-defined*. The list of existing user defined properties are displayed.

5. Select the property for which the column is to be added.

6. Column heading for the column is reflected in the *Name* text field.

   Modify the column name, if required.

7. Click *OK.*

The column for the property is displayed on all the relevant worksheets. For example, if a property can be added on components and on pins, a column for the property is displayed in the *General Properties* worksheet in the *Component* folder and in the *Pin Properties* worksheet.

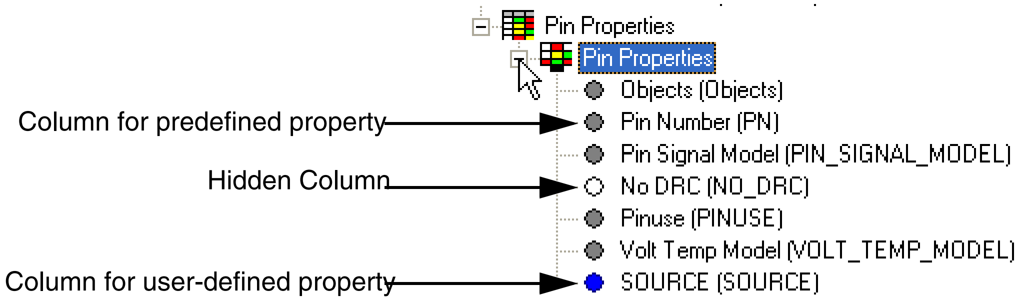
Managing Columns for Properties in Constraint Manager

You can customize the worksheets in Constraint Manager such that only the required properties are visible in the worksheet. Constraint Manager provides support to hide, show, or delete columns for properties.

1. In Constraint Manager, choose *Tools – Customize*.

2. Expand the worksheet in which you want to hide, show, or delete the column for a property.

For example, if you want to hide the column for a pin property, expand the *Component* object folder, expand the *Pin Properties* workbook, then expand the *Pin Properties* worksheet, as shown below:


Pin Properties
Pin Properties
Objects (Objects)
Column for predefined property
Pin Number (PN)
Pin Signal Model (PIN_SIGNAL_MODEL)
Hidden Column
No DRC (NO_DRC)
Pinuse (PINUSE)
Volt Temp Model (VOLT_TEMP_MODEL)
Column for user-defined property
SOURCE (SOURCE)

In the above figure, the gray circle indicates a column for a predefined property, the circle with white color indicates a hidden column, and the blue circle indicates a column for a user-defined property.

- ❑ To hide a column, select the column for the property, right-click and choose *Visible*.
- ❑ To show a column that is hidden, select the column, right-click and choose *Visible*.
- ❑ To delete a column, select the column, right-click and choose *Delete Column*.

Tip

You can also select the column head for a property in a worksheet, right-click and choose *Hide Column* to hide a column.

Note the following:

- You can only delete the columns for user-defined properties. You cannot delete the columns for predefined properties.

  To know more about user defined properties, see Working with User-Defined Properties in Constraint Manager.

- When you hide or delete the column for a property, the property is not deleted from the objects on which it is added.

Working with User-Defined Properties in Constraint Manager

Constraint Manager lets you define user-defined properties. You can use an user defined property to capture a characteristic of an object.

The following topics provide information on working with user defined properties:

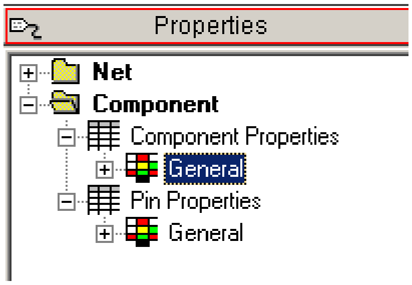
- Defining User-Defined Properties in Constraint Manager on page 188
- Modifying the Definition of User-Defined Properties in Constraint Manager on page 191
- Deleting the Definition of User-Defined Properties in Constraint Manager on page 192

You can also define user-defined properties in System Connectivity Manager.

Defining User-Defined Properties in Constraint Manager

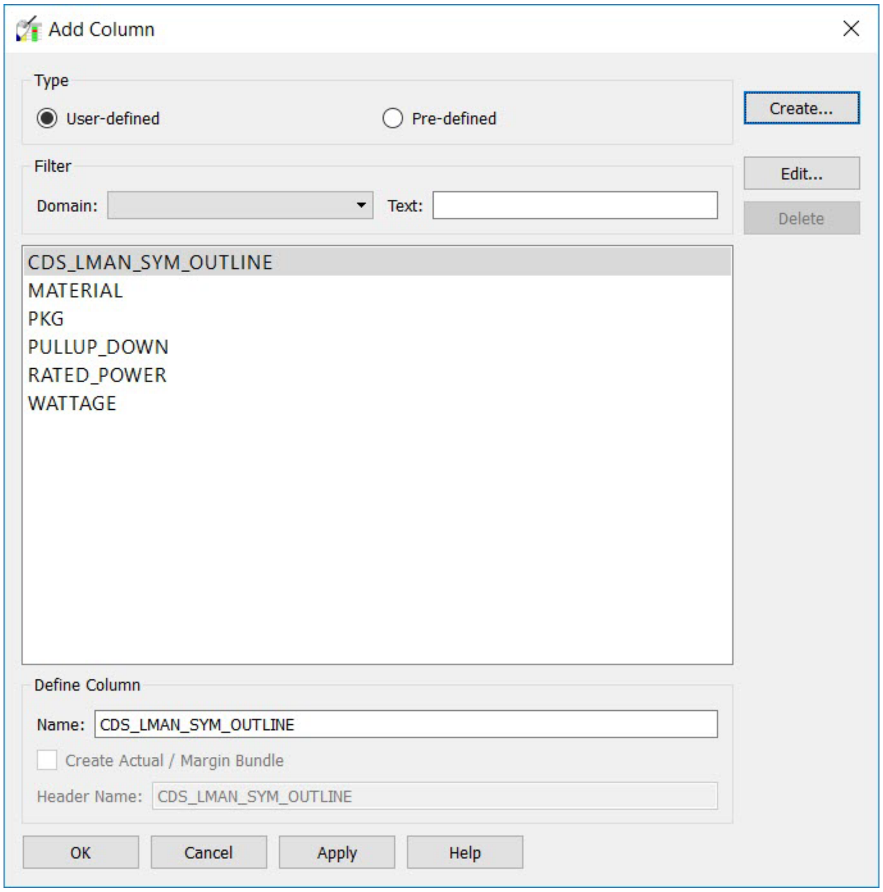
1. In Constraint Manager, choose *Tools – Customize Worksheets*.
2. Expand the workbook that contains the property worksheet for an object.

   For example, expand the *Component* object folder, then expand the *General Properties* workbook to display the *General* worksheet, as shown below:



3. Select the property worksheet for the object, right-click and choose *Add Column*.

The *Add Column* dialog box appears.



Type

● User-defined ○ Pre-defined

CDS_LMAN_SYM_OUTLINE
MATERIAL
PKG
PULLUP_DOWN
RATED_POWER
WATTAGE

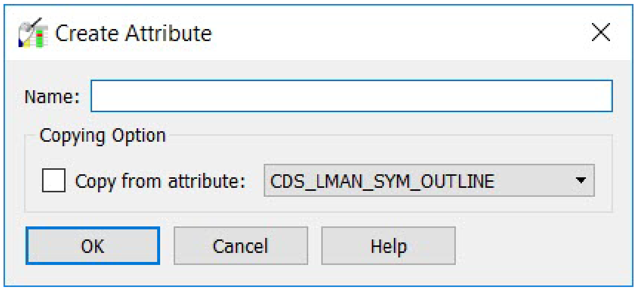
Name: CDS_LMAN_SYM_OUTLINE

□ Create Actual / Margin Bundle

Header Name: CDS_LMAN_SYM_OUTLINE

4. Select the attribute type as *User-defined*.
5. Click *Create*.

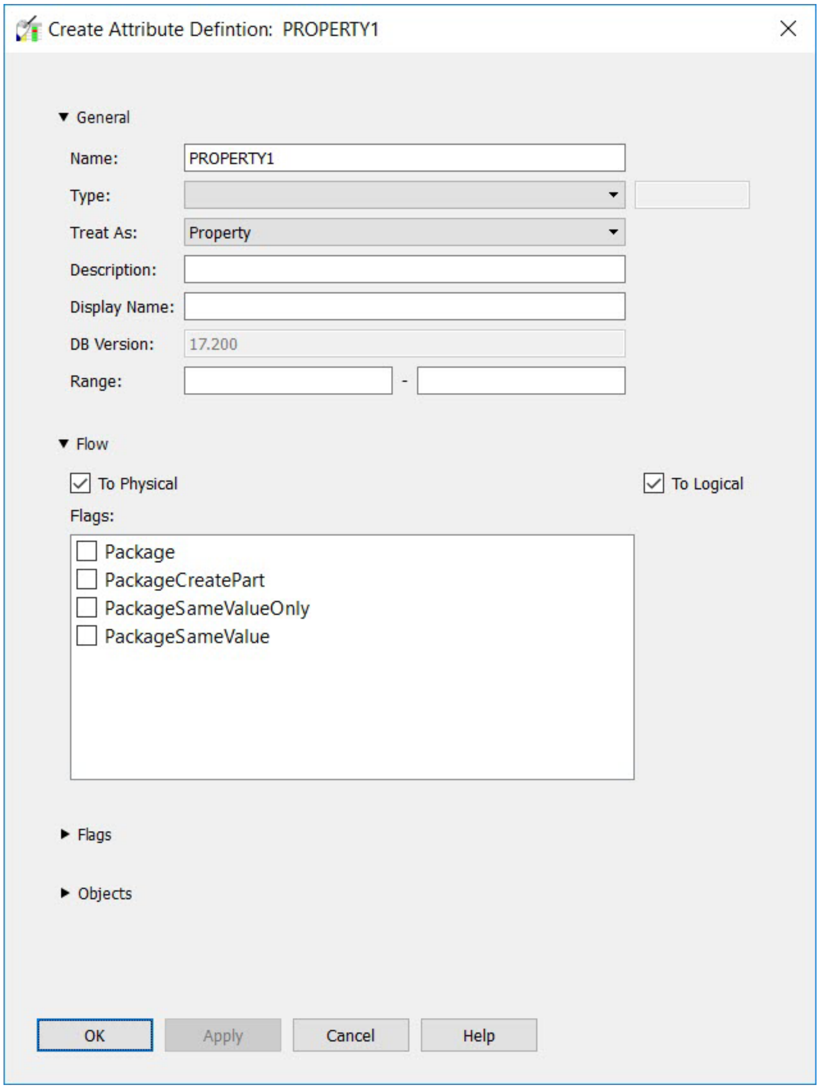
   The *Create Attribute* dialog box appears.



□ Copy from attribute: CDS_LMAN_SYM_OUTLINE

6. In the *Name* field, specify the name of the attribute to be added.
7. Click *OK*.

The *Create Attribute Definition* dialog box appears.



Create Attribute Defintion: PROPERTY1

▼ General

Name: PROPERTY1

Type:

Treat As: Property

Description:

Display Name:

DB Version: 17.200

Range: -

▼ Flow

☑ To Physical ☑ To Logical

Flags:

□ Package
□ PackageCreatePart
□ PackageSameValueOnly
□ PackageSameValue

8. From the *Type* drop-down list, select the data type to be used for specifying the value of the attribute.

9. In the *Objects* list box, select the check box next to the object you want to add to the property.

10. In the *Range* field, specify the range of values that are acceptable for the property.

    When you enter a property value, Constraint Manager displays an error message if the value is not within the specified range.

11. In the *Description* field, enter a description for the property.

12. In the *Flow* list box, select the *To Physical* or *To Logical* check box if you want the property to be transferred between Design Entry HDL and Allegro PCB Editor along with the netlist when you run the *Export Physical* command.
13. In the *Flags* list box, under *Flow*, select the check box next to the flag you want to add to the property.

    For more information about flags, refer to the *Create Attribute Definitions Dialog Box* section of the *Allegro Constraint Manager Reference* guide.
14. Click *OK*.

    The Add Column dialog box appears displaying the new user defined property.
15. Click *OK*.

The column for the property is displayed on all the relevant worksheets. For example, if a property can be added on components and on pins, a column for the property is displayed in the *General Properties* worksheet in the *Component* folder and in the *Pin Properties* worksheet.

Modifying the Definition of User-Defined Properties in Constraint Manager

Caution

You cannot modify the definition of a predefined property.

To modify the definition of a user-defined property:

1. In Constraint Manager, choose *Tools – Customize*.
2. Expand the workbook that contains the property worksheet for an object.

    For example, expand the *Component* object folder, then expand the *General Properties* workbook to display the *General* worksheet.
3. Select the property worksheet for the object, right-click and choose *Add Column*.

    The *Add Column* dialog box appears.
4. Click the *Type* drop-down list and choose *User-defined*.

    The list of existing user-defined properties are displayed.
5. Select the property you want to modify and click *Edit*.

    The *Edit Attribute Definition* dialog box appears.

6. Modify the user defined property and click *OK*.
7. Click *OK.*

   The *Add Column* dialog box appears.
8. Click *Cancel*.

Deleting the Definition of User-Defined Properties in Constraint Manager

Caution

If you delete the definition for a user-defined property, the property is deleted from all the objects in the design on which it exists.

Note: You cannot delete the definition of predefined properties. If you do not want to see the column for a predefined property, you can hide it. For more information, see Managing Columns for Properties in Constraint Manager on page 186.

To delete the definition of a user-defined property:

1. In Constraint Manager, choose *Tools – Customize*.
2. Expand the workbook that contains the property worksheet for an object.

   For example, expand the *Component* object folder, then expand the *General Properties* workbook to display the *General* worksheet.
3. Select the property worksheet for the object, right-click and choose *Add Column*.

   The *Add Column* dialog box appears.
4. Click the *Type* drop-down list and choose *User-defined*.

   The list of existing user defined properties are displayed.
5. Select the property you want to delete and click *Delete*.
6. If the property is being used, a message appears stating that all instances of the property will be deleted.

   Click Yes.

   The property is deleted from all the objects in the design. The column for the property is also deleted from the relevant worksheets. For example, if a property can be added on components and on pins, the column for the property is deleted from the *General* worksheet in the *Component Properties* workbook and from the *General worksheet* in the *Pin Properties* workbook.

10

Migrating from Previous Releases to the Current Releases

This appendix contains the following sections:

Opening Pre-16.0 Designs in Later Releases

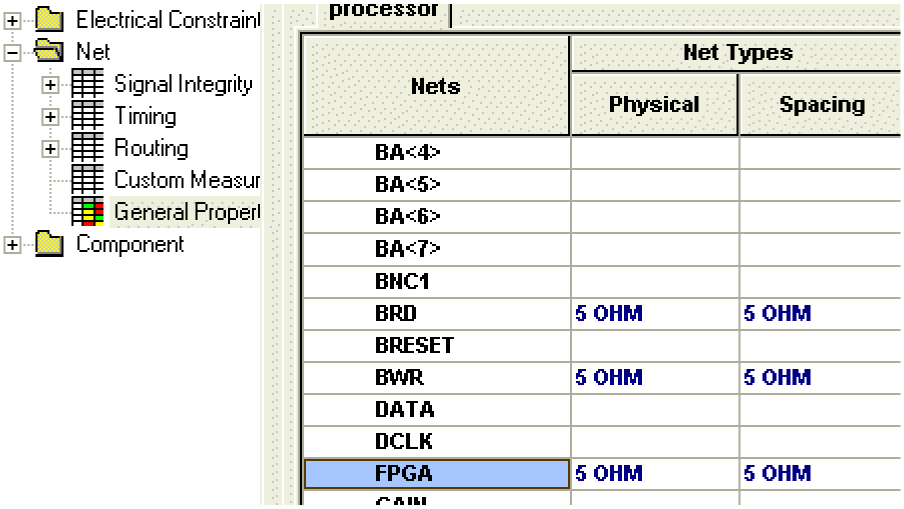
If you open a pre-16.0 design in later releases, the net physical type and net spacing type constraints in the design are converted to corresponding physical and spacing classes in Constraint Manager. The `NET_PHYSICAL_TYPE` and `NET_SPACING_TYPE` values on the nets are now listed as classes in Constraint Manger. The nets to which these properties were attached are listed as class members.

Scenario 1

If you have a 15.7 design for which the logical design is in sync with the board design, and you now open the logical design in a 16.0 or later release and launch SCM, the following modifications are visible:

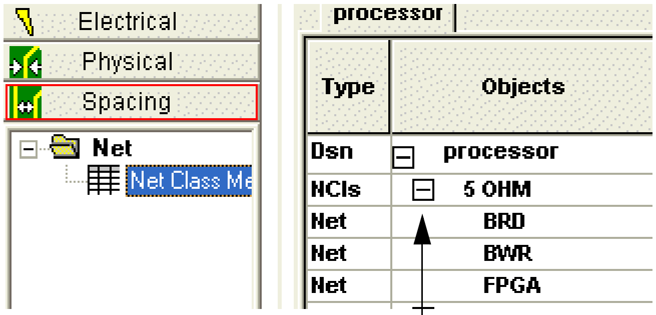
- The values assigned to NET_PHYSICAL_TYPE and NET_SPACING_TYPE properties are converted to net classes listed in the Physical and Spacing domain respectively.

- All nets with the same value for net physical and spacing constraints are listed as member nets of the same net class object.

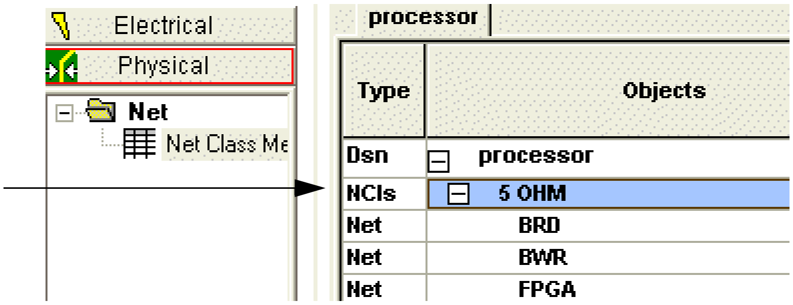


Nets	Net Types	
	Physical	Spacing
BA<4>		
BA<5>		
BA<6>		
BA<7>		
BNC1		
BRD	5 OHM	5 OHM
BRESET		
BWR	5 OHM	5 OHM
DATA		
DCLK		
FPGA	5 OHM	5 OHM

Net physical and spacing constraints in release 15.x



Type	Objects
Dsn	⊟ processor
NCls	⊟ 5 OHM
Net	BRD
Net	BWR
Net	FPGA

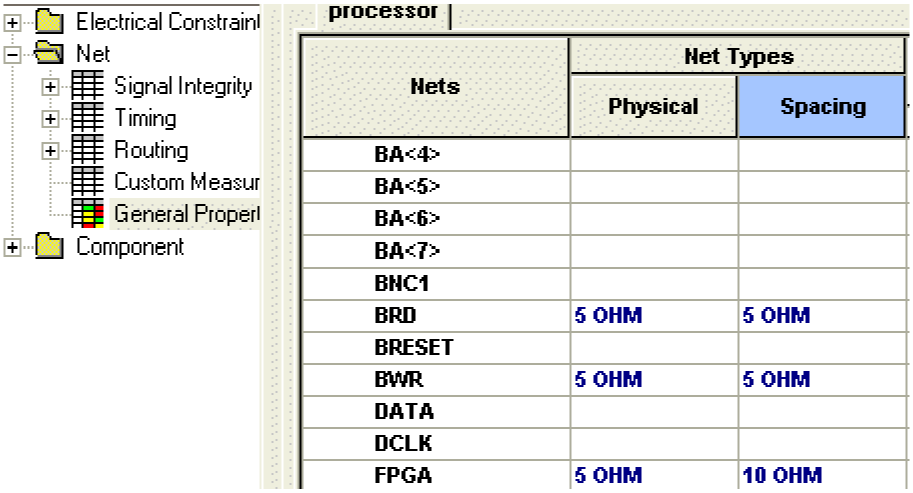


Type	Objects
Dsn	⊟ processor
NCls	⊟ 5 OHM
Net	BRD
Net	BWR
Net	FPGA

Net physical and spacing classes in release 16.0

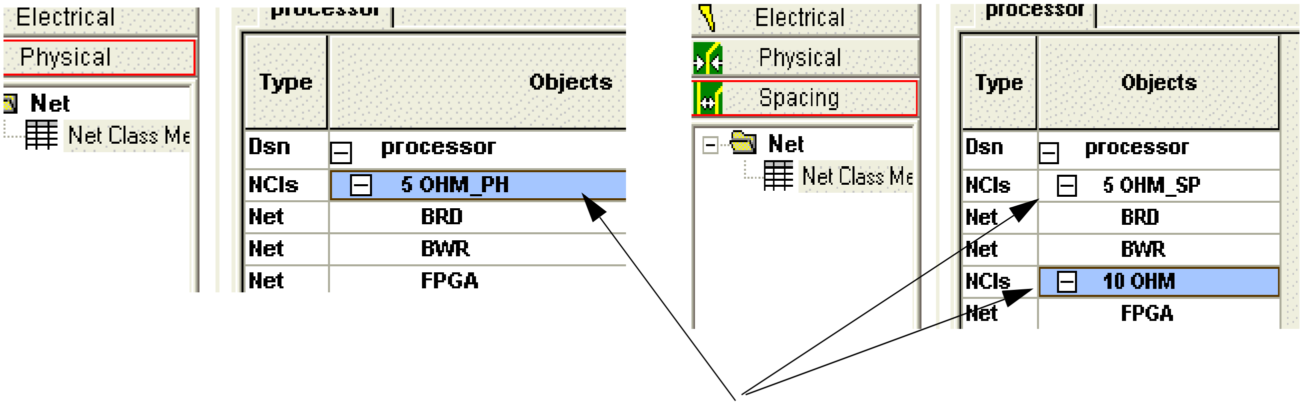
Scenario 2

- By default, Constraint Manager creates physical and spacing classes with the same names. Therefore, in a design if the number of nets with the same physical constraint is different from the number of nets with a similar spacing constraint, the net physical class created in the 16.0 or later release has `_PH` attached as a suffix to the net class name. Similarly, the net class created in the spacing domain has `_SP` as a suffix.



Nets	Net Types	
	Physical	Spacing
BA<4>		
BA<5>		
BA<6>		
BA<7>		
BNC1		
BRD	5 OHM	5 OHM
BRESET		
BWR	5 OHM	5 OHM
DATA		
DCLK		
FPGA	5 OHM	10 OHM

Separate values for net physical and spacing constraints in 15.x release



Type	Objects
Dsn	processor
NCls	5 OHM_PH
Net	BRD
Net	BWR
Net	FPGA

Type	Objects
Dsn	processor
NCls	5 OHM_SP
Net	BRD
Net	BWR
NCls	10 OHM
Net	FPGA

Net classes in the physical and spacing domain in the 16.0 release

Front-to-back flow

Consider the following situation:

You have a design created in a previous release, where the logical design is in sync with the board design and the board design has nets with an overridden physical constraints value. If you now open the logical design in SCM or DE-HDL and open the board in Allegro PCB Editor, you will see that additional classes generated in Allegro PCB Board when compared to SCM or DE-HDL, giving you the impression that the design and board are out of sync. To ensure that the additional classes are available in the front-end tools, you need to take your design through a complete front-to-back flow using *the Import Physical* command.

Migrating a Design from pre-15.2 to a Higher Version

To support the migration of pre-15.2 designs to later versions, Design Entry HDL provided the *retain existing XNets and diff-pairs* mode. In this mode, all the existing XNets and model-defined diff-pairs in your design are retained and you can see them in Constraint Manager. This mode is set by default when you open a pre-15.2 design in later releases.

Retain Existing XNets and Diff-Pairs Mode

In the *retain existing XNets and diff-pairs* mode, Constraint Manager does not validate or recognize the signal models you assign in Design Entry HDL. All the existing XNets and model-defined differential pairs in your design are retained and you can see them in Constraint Manager. However, in this mode, you cannot define new XNets or differential pairs. To define XNets and differential pairs in 15.2 and 15.5, you need to enable the Signal Integrity analysis features by switching to the *non-retain* mode.

Note: Before you migrate your 14.2 designs with XNet constraints to 15.2 or 15.5, you need to perform the *Import Physical* operation in Design Entry HDL keeping the *Electrical Constraints* check box selected. Otherwise, XNet constraints will be lost when you invoke Constraint Manager. The *Import Physical* operation brings all the constraints in OPF. Therefore, before you move such designs to 15.2 or 15.5 and switch off the default *Retain Existing XNets and Diffpairs* directive, you must run the *Import Physical* command.

Non-Retain Mode

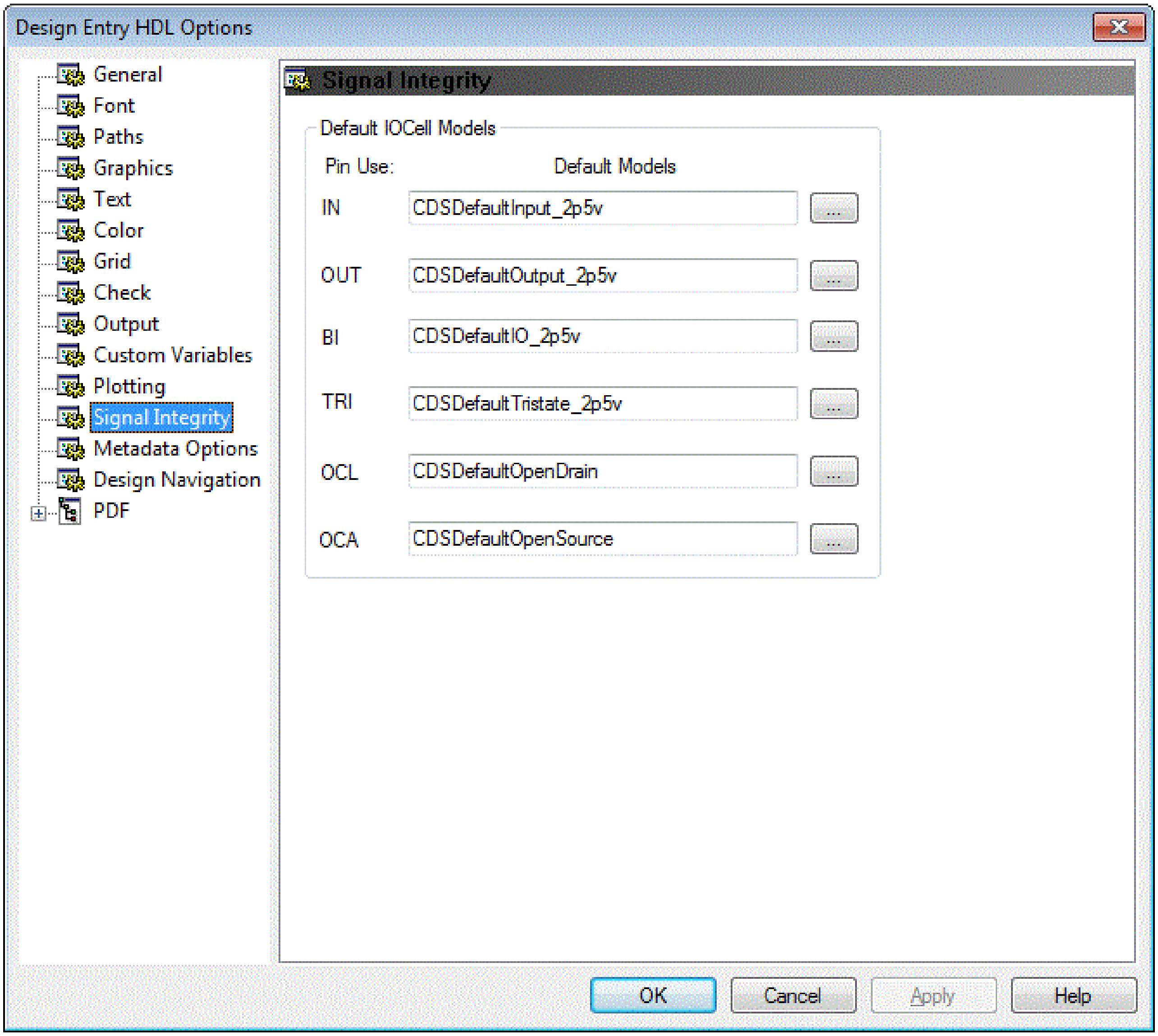
In the non-retain mode, Constraint Manager validates signal models that you assign in Design Entry HDL. All the XNets and model-defined differential pairs are retained based on the validation results.

Note: Before you start defining XNets and differential pairs in 15.2 or 15.5 by assigning signal models to devices, you must ensure that you are not in the *retain existing XNets and diff-pairs* mode.

Enabling Signal Integrity Analysis Features

The *retain existing XNets and diff-pairs* mode is the default mode when you open a pre-15.2 design in 15.2 or 15.5. To create new XNets in Design Entry HDL, you need to enable Signal Integrity analysis features by switching to the *non-retain* mode. To switch to the *non-retain* mode, perform the following steps:

1. In Design Entry HDL, choose *Tools – Options*.
2. Choose the *Signal Integrity* tab.



Default IOCell Models

Pin Use:	Default Models
IN	CDSDefaultInput_2p5v
OUT	CDSDefaultOutput_2p5v
BI	CDSDefaultIO_2p5v
TRI	CDSDefaultTristate_2p5v
OCL	CDSDefaultOpenDrain
OCA	CDSDefaultOpenSource

3. Click OK.

Important

Before you switch to the *non-retain* mode, ensure the following:

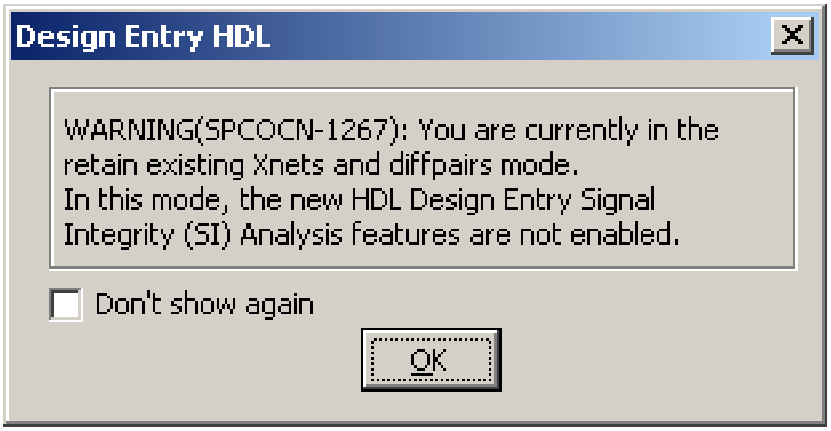
- ❑ If you are migrating your design to 15.2 or 15.5, all signal models assigned in PCB Editor are ported to Design Entry HDL by running the *Import Physical* command.
- ❑ Signal models are assigned to instances, and the path to device model libraries (`.dml` files) is set up correctly. If the .dml file is missing, then you will get an error on launching CM. to avoid this error, set the following directive to 'ON":

```
START_ECSET_MODELS

retain_existing_xnets_and_diffpairs 'ON'

END_ECSET_MODELS
```

Once this directive is set to "ON", the error is not displayed and you can launch CM. To confirm if the directive is set, the following message is displayed when you launch CM:



WARNING(SPCOCN-1267): You are currently in the retain existing Xnets and diffpairs mode.
In this mode, the new HDL Design Entry Signal Integrity (SI) Analysis features are not enabled.

□ Don't show again

Note: If valid signal models are not present in the `physical` folder, you can also create a dump of models from back-end to a new signal integrity model library. To create a dump of models, select the *Analyze – SI/EMI Sim – Model Dump/Refresh* menu option from the main menu of either your PCB- or package- editor. For more information, refer to the refer to the *Model and Library Management* chapter of Allegro PCB SI User Guide.

This section describes how Design Entry HDL behaves at different stages in the two modes.

Constraint Manager Connected to Design Entry HDL

Retain existing XNets and diff-pairs mode

- All the existing XNets and model-defined diff-pairs and their constraints will be visible. Constraint Manager will not validate whether signal models are found or not.
- In this mode, you will be able to apply pin-pair constraints on XNets. When you apply an ECSet on an XNet, it will appear as an invalid application in the worksheet if the XNet exists in the `.dcf` file but the signal model is not defined in the schematic. Similarly, if the signal model exists on a discrete in the schematic and the ECSet contains pin-pair constraints, pin-pairs for the XNet will be applied when you apply the ECSet on the XNet.

Non-retain mode

- Constraint Manager will validate all the signal models assigned in the schematic. If the SIGNAL_MODEL property is present on the schematic, chips.prt, or the ptf but the model is not found in the device models library, you will be prompted to abort the session.

Export Physical

Retain existing XNets and diff-pairs mode

- All the existing XNets and model-defined diff-pairs and the constraints will be retained.

Non-retain mode

- The existing XNets and model-defined diff-pairs are not validated.

Import Physical

Retain existing XNets and diff-pairs mode

- In case all the models are found and backannotated to Design Entry HDL, you are prompted to switch to the *non-retain* mode.

Non-retain mode

- In case the SIGNAL_MODEL property is present on the schematic, but the signal model is not found in the device models library, the process will abort and the `.dcf` file will not be updated with back-end changes.

Model Assignment UI

Retain existing XNets and diff-pairs mode

- When you launch the Model Assignment UI, you are prompted to switch to the *non-retain* mode, so that the signal models can be validated by Constraint Manager when launched from Design Entry HDL. On closing the Model Assignment UI, you are prompted to switch to the *non-retain* mode if you are still in the *retain existing XNets and diff-pairs* mode.

Non-retain mode

- You can assign signal models to devices in this mode.

Signal Integrity Messages

Design Entry HDL prompts you to switch modes and take appropriate actions at various stages so that you can use the new signal integrity analysis features effectively. These messages are listed below.

- On launching the Model Assignment window in the *retain existing XNets and diff-pairs* mode:

  ```
  You are currently in the retain existing XNets and diffpairs mode. In this mode, the
  new HDL Design Entry Signal Integrity (SI) Analysis features are not enabled.
  To enable the Signal Integrity (SI) Analysis features, deselect the Retain
  Existing XNet and DiffPairs option in the Signal Integrity page of the Design
  Entry HDL Options dialog box.
  ```

- On deselecting the *Retain Existing XNets and DiffPairs* option in the *Signal Integrity* page of the Design Entry HDL Options dialog box:

  ```
  You are enabling the new HDL Design Entry Signal Integrity (SI) Analysis
  features. Please ensure that all signal models are correctly assigned.
  Otherwise, you might lose your existing XNets and diffpairs. To enable the
  Signal Integrity (SI) Analysis features, deselect the Retain Existing XNet and
  DiffPairs option in the Signal Integrity page of the Design Entry HDL Options
  dialog box.
  ```

- On launching Constraint Manager or run Export Physical and some of the models are not found in the *non-retain* mode:

  ```
  Model <model name> on instance <instance name> were not found. This can lead to
  loss of XNet and diffpair information. Processing Aborted. For a complete list
  of missing models, refer to concept2cm.log.
  ```

- On running Import Physical and some of the models are not found in Design Entry HDL in the *non-retain* mode:

  ```
  Some of the signal models in Allegro PCB Editor were not found in Design Entry
  HDL. This can lead to potential loss of XNets and diffpairs. Launch Model
  Assignment in Design Entry HDL and assign the missing models or set up the
  library path for the missing models.
  Do you wish to see a list of missing models?”
  ```

- On running Import Physical and all the models are found in Design Entry HDL in the *retain existing XNets and diffpairs* mode:

  ```
  All signal models in Allegro PCB Editor were found in Design Entry HDL. However,
  you are currently in the retain existing XNets and diffpairs mode. In this mode, the
  ```

```
new HDL Design Entry Signal Integrity (SI) Analysis features are not enabled.
To enable the Signal Integrity (SI) Analysis features, deselect the Retain
Existing XNet and DiffPairs check box in the Signal Integrity page of the Design
Entry HDL Options dialog box.
```

- The following message appears when you run Import Physical and some of the models are not found in Design Entry HDL in the *retain existing XNets and diffpairs* mode:

```
Some of the signal models in Allegro PCB Editor were found in Design Entry HDL.
The new HDL Design Entry Signal Integrity (SI) Analysis features are not enabled
because you are currently in the retain existing XNets and diffpairs mode. To enable
the Signal Integrity (SI) Analysis features, deselect the Retain Existing XNet
and DiffPairs check box in the Signal Integrity page of the Design Entry HDL
Options dialog box.
```

```
Do you wish to see a list of missing models?
```

11

Frequently Asked Questions About the Constraint Manager Flow

Why is the Autogenerate button not enabled for discrete components?

This situation can arise because of the following reasons:

- The CLASS property has a value other than DISCRETE in the chips.prt file (CLASS='DISCRETE' statement is not present).
- The PHYS_DES_PREFIX property has a value other than *R* in the chips.prt file (PHYS_DES_PREFIX='R' statement is not present).

Why is SigXplorer not launching for specific nets/XNets?

This can happen because of the following reasons:

- The signal model applied is not valid.
- PIN_TYPE information is not extracted from the chips.prt file.

Why are all the XNet segments in my design not appearing in Constraint Manager connected to Design Entry HDL?

If there are more than ten segments in an XNet, Constraint Manager connected to Design Entry HDL will not show the XNets. If your design contains pull up and pull down resistors without the VOLTAGE property assigned to them, they will appear as XNets thereby using up the place of valid XNet segments. You must add the VOLTAGE property to the pull down and pull up resistors, so that they do not appear as XNets. This will free up space and valid XNet segments can appear as XNets in Constraint Manager.

There is an option in the Electrical Checks section of the Checks page of the Design Entry HDL Options dialog box, *Voltage on HDL Symbols*. This option checks for the presence of the VOLTAGE property on an HDL_POWER symbol. If the VOLTAGE property is not present, a warning message is displayed.

Note: There are certain net names such as GND and VCC, which are never treated as XNets even if the `VOLTAGE` property is not present on them.

What will happen to XNets and model-defined differential pairs and their constraints when the signal model is not in the path?

If the signal model library is not in the path, XNets and model-defined differential pairs will not be created. You will receive an warning message prompting you to change the mode.

Refer to Migrating a Design from pre-15.2 to a Higher Version on page 196 for more details

12

Glossary

Overshoot

Overshoot is the maximum voltage swing above the input voltage. It specifies the acceptable voltage limits of logic families.

Noise Margin

Noise margin is the voltage difference between the maximum voltage dip and the active high threshold or between the maximum voltage dip and the active low threshold.

Jitter

Jitter is the deviation in pulse width of a clock cycle, keeping the clock cycle same.

Sensitive Edge

Sensitive edge signals are those that drive receivers by their edge thresholds. A typical example of a sensitive edge signal is a clock signal.

First Incident Switch

First incident switching is the switching voltage of sufficient amplitude at the initial rise of a signal which is sufficient to drive receivers.

Propagation Delay

Propagation delay is the summation of all calculated transmission line delays along the shortest path between two points. The default unit for propagation delay is ns.

Settle Time

Settle time is the time required for a ringing signal to stabilize to within a specified range of the final value.

Minimum First Switch Time

Specifies the maximum transmission line wire delay plus distortions differing from the nominal driver rise-or-fall time seen in the receiver rise-or-fall.

Duty Cycle

The portion of the time the pulse stimuli is held in the high state as a fraction of the entire pulse period. A value of 0.5 represents equal high and low portions of the cycle period.

Simultaneous Switching Noise

When a number of drivers switch simultaneously in a digital system, a sudden change in current occurs through the power and ground connections to the die. Because of the parasitic inductance that exists in this path, any current change produces a temporary fluctuation in the power and ground voltages as seen by the die. This is typically referred to as Simultaneous Switching Noise (SSN), or Ground Bounce. Simultaneous switching noise can cause noise at the output of non-switching drivers. This noise will then propagate to loads on the net and potentially cause false switching.

Impedance

Impedance is the ratio of input voltage to input current for a transmission line (Z0 = V/I). When a source sends a signal down a line, this is the impedance it must drive. The Source will not see a change in its loading Impedance until 2*TD, where TD is the delay of the line.

Reflection

A reflection on a transmission line is an echo. A portion of the signal power (voltage and current) transmitted down the line goes into the load, and a portion is reflected. Reflections are prevented if the load and the line have the same impedance.

Setup Time

The time for which a digital signal A must be stable and unchanging prior to another digital signal of interest B.

Setup time is most often associated with activity of digital signals immediately before a clock event when these signals must be stable and ready as necessary inputs to clocked circuits, especially latches.

Hold Time

The time for which a digital signal A must be stable and unchanging following the change of another digital signal of interest B.

This parameter is very important with synchronous state machines employing feedback logic that can change as a result of the clock.

Clock Skew

Clock skew is the difference in arrival time between clock and data at a logic gate.

Differential Pair

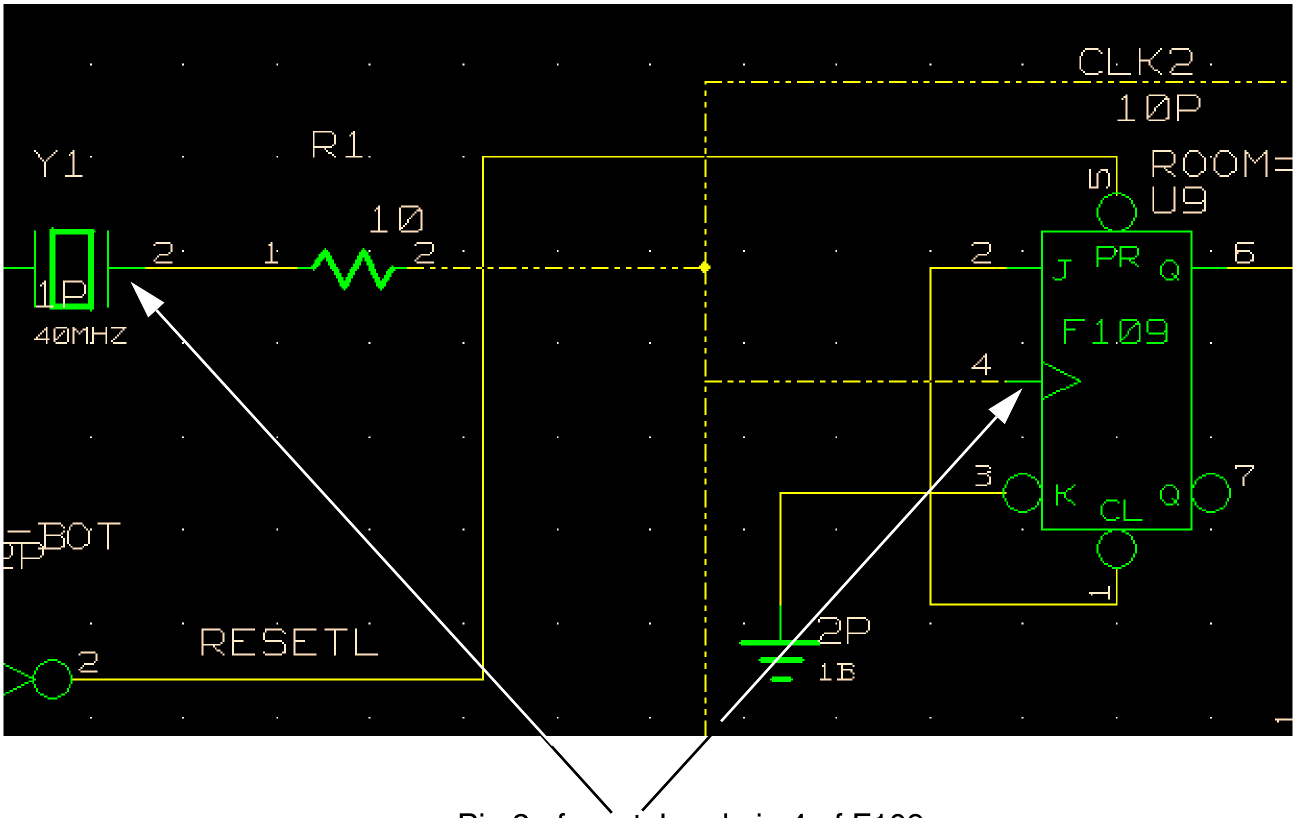
A differential pair represents a pair of nets or XNets that will be routed in a way that the signals passing through them are opposite in sign with respect to the same reference. This ensures that any electromagnetic noise in the circuit is cancelled out.

Pin-Pair

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 be on the same net or XNet. A pin-pair for a net connecting component 1 and component 2 is represented as follows:

```
reference designator of component 1.pin number : reference designator of component
     2.pin number
```

For example in the following figure, a pin-pair for net CLK2 is:



Pin 2 of crystal and pin 4 of F109
form a pin-pair for net CLK2.

You can specify a pin-pair explicitly, or it can be derived based on the length of the physical net between the pins forming the pin-pair. The length of the net is determined when the board for the schematic is placed and routed in PCB Editor. Accordingly, the pin-pairs are categorized as follows:

- **longest/shortest pin-pair**

  Out of all the possible pairs of pins that a net connects, the longest pin-pair is the one between whose pins the length of the connecting net is maximum. Similarly, the shortest pin-pair is the one between whose pins the length of the connecting net is minimum.

- **longest/shortest driver-receiver**

  Out of all the pairs of pins for a net where one pin outputs the signal (driver) on the net and the other takes input (receiver), the longest driver-receiver is the one between whose pins the length of the connecting net is maximum. Similarly, the shortest driver-receiver is the one between whose pins the length of the connecting net is minimum.

- **all driver-receiver pin-pairs**

  This refers to all possible pairs of pins for a net such that one pin outputs the signal (driver) on the net and the other takes input (receiver).

Electrical Constraint Set

An electrical constraint set (ECSet) is a collection of constraints and their default values. An ECSet reflects a particular design requirement. You can capture any or all electrical constraints in an ECSet.

ECSets reside in the Electrical Constraint Set object folder. You can create ECSets for signal integrity, timing, and routing constraints.

Primary Gap

Indicates the ideal edge-to-edge spacing between the pair of nets in the differential pair that should be maintained for the entire length of the pair.

13

Troubleshooting Design Entry HDL to Constraint Manager Flow

This appendix lists the solutions to some common or intermittent problems encountered while using Allegro Design Entry HDL.

- Unable to launch Constraint Manager from specific design in Allegro Design Entry HDL
- Incorrect property definitions causing Netrev errors while importing a netlist in the board

Unable to launch Constraint Manager from specific design in Allegro Design Entry HDL

Description

When you launch Constraint Manager from few designs in Design Entry HDL 15.2 or 15.5, the Constraint Manager window may appear briefly and then disappear. The following MPS error is returned by Design Entry HDL.

```
MPS Error: MPSC: bad Handle: 0xbe52e0
```

Reason

It is possible that you have placed instances of a component in the schematic in Design Entry HDL and have not yet added any wires to connect them.

Solution

Ensure that component pins are connected with wires and launch Constraint Manager again from Design Entry HDL. It should run.

Incorrect property definitions causing Netrev errors while importing a netlist in the board

Description: You may find the following Netrev (305) error while importing a netlist in a template board file in Allegro. This error will occur for every component in the netlist.

```
ERROR (305) Device/Symbol check error detected..... 'Attribute Definitions are
incompatible for attribute ....
```

Reason

This problem may occur if

- there is an improperly defined user-defined property in the Allegro board file to which the netlist is being imported.
- footprints are not upreved to the latest version.

Solution

To verify bad property definition:

1. Choose *Setup – Property Definition* from the main Allegro PCB Editor menu.

   The errant property definition is displayed in the *Available Properties* pane of the Define User Properties dialog box.

2. Select the property.

   The Data Elements and Data Type of the property are displayed in the right hand pane of the Define User Properties dialog box.

To correct the problem:

➤ Select at least one data type for the errant property and *Apply* the change.

You should now be able to import the netlist successfully.

14

Property Mapping

Topics in this appendix include

Overview

Constraint Manager organizes constraints by tabs, workbook, then by worksheet, then by cell. Workbooks can be generic, such as those under the Electrical CSet folder, or they can be net-related.

This appendix depicts the cell name, as shown in Constraint Manager, with the associated property that is stored in the schematic, board, or package database. Consult the *Allegro® Platform Properties Reference* for property definitions.

- Tables B1, B2, and B3 present constraints contained in the Electrical CSet folder
- Tables B4, B5, B6, and B7 present constraints contained in the Net folder

You cannot edit cells that contain *Actual* or *Margin* calculations; therefore, these cells are not presented in the worksheets under the Net folder.

The All Constraints workbook in the Electrical CSet folder presents a flattened view of all constraints under the Electrical CSet folder. Because they are described elsewhere, cells in the All Constraints folder are not presented in these tables.

Worksheets In the Net Folder

Properties in the Electrical Properties Worksheet

Table 14-1 Electrical Properties Worksheet

Column Heading	Cell Label	Property Name
Frequency	Not Applicable	`PULSE_PARAM_FREQUENCY`
Period	Not Applicable	`PULSE_PARAM_PERIOD`
Duty Cycle	Not Applicable	`PULSE_PARAM_DUTY_CYCLE`
Jitter	Not Applicable	`PULSE_PARAM_JITTER`
Cycle to Measure	Not Applicable	`PULSE_PARAM_MEASURE_CYCLE`
Offset	Not Applicable	`PULSE_PARAM_COFF`
BitPattern	Not Applicable	

Properties in the General Properties Worksheet

Table 14-2 General Properties Worksheet

Column Heading	Cell Label	Property Name
	Voltage	VOLTAGE
	Weight	WEIGHT
	No Rat	NO_RAT
Route	*Priority*	ROUTE_PRIORITY
	to shape	ROUTE_TO_SHAPE
Route Restrictions	*Fixed*	FIXED
	No Route	NO_ROUTE
	No Ripup	NO_RIPUP
	No Pin Escape	NO_PIN_ESCAPE
Testpoints	*Prohibit*	NO_TEST
	Quantity	TESTPOINT_QUANTITY
	Probe Number	PROBE_NUMBER
Backdrill	*Max PTH Stub*	BACKDRILL_MAX_PTH_STUB
	No Gloss	NO_GLOSS
Shield	*Shield*	SHIELD_NET
	Type	SHIELD_TYPE

Worksheets In the Component Folder

Properties in the Component Properties Worksheet

Table 14-3 Component Properties

Column Heading	Property Name
Part name	
Alt Symbol	
JEDEC TYPE	
GROUP	
ROOM	
HEIGHT	
VALUE	
TOL	
Component Name	

Table 14-3 Component Properties

Column Heading	Property Name

Properties in the Pin Properties Worksheet

Table 14-4 Pin Properties

Column Heading	Property Name
Pin Number	`PN`
Pinuse	`PINUSE`
PinDelay	`PIN_DELAY`
No DRC	`NO_DRC`
PIN_TEXT	`PIN_TEXT`
TT	`TT`
QW	`Q`

15

Recommendation for Allegro Design Entry HDL — Constraint Manager Flow

Overview

This document defines the Cadence-recommended flow for a constraint-enabled schematic design captured in Allegro Design Entry HDL and Allegro PCB Editor. Recommendations are made for library organization, constraint capture methodology, and the sequence of steps for capturing constraints. The document is targeted at CAD administrators and schematic designers who work with Design Entry HDL to capture schematic connectivity and constraints. The document will also provide recommendations for migrating designs from an earlier release of SPB to the latest release

Terms Used

Synchronization Properties

The constraints which can be captured on the schematic canvas and are required to be synchronized between Design Entry HDL and Constraint Manager. These constraints are defined in the file synch_props.cfg.

Non- Synchronization Properties

The constraints which reside in the Constraint Manager database only (`.dcf` file) and cannot be captured on the schematic canvas. These constraints are not synchronized between Design Entry HDL and Constraint Manager. You can convert non-synch constraints to synch constraints by adding them to the synch_props.cfg file. However, pin pair constraints can not be captured on the schematic canvas, even if mentioned in the synch_props.cfg file.

Synch Constraints

All The constraints included in the synch_props.cfg file are considered as synch constraints and can be written on to the schematic. You can add constraints to the configuration file and then write them on to the schematic. These constraints are synchronized with the Constraint Manager database, which means that the value of the constraints can be edited on the schematic canvas or in the Constraint Manager spreadsheet and is always in synch. The DIFFERENTIAL_PAIR and VOLTAGE properties are the default synch constraints. Synch constraints are visible in the Attributes form in the Hierarchy mode.

Non-Synch Constraints

All the constraints, which are not listed in the configuration file, are considered as non-synch and are stored in the Constraint Manager database. If there are non-synch constraints present on the schematic canvas, introduced by means of a copied or imported sheet, the Synchronize utility moves those constraints to the Constraint Manager database. Non-synch constraints are visible in the Attributes form only in the Expanded or Occurrence Edit mode. You can convert non-synch constraints to synch constraints by adding them to the synch_props.cfg file.

Recommendations for Capturing Constraints

General Recommendations

It is recommended that:

- The schematic designers and PCB designers have access to the same SI model libraries.
- Model assignment should be done in the schematic or as an injected part table property in scenarios where there are a large number of physical parts associated with the same logical component. Examples include resistor and capacitor networks.
- Constraints should be captured in Constraint Manager and not on the schematic canvas.
- Placeholders should be used on the schematic canvas if constraints need to be seen on printed schematic documentation.
- Synchronization properties should be kept to a minimum. This maximizes performance by minimizing schematic to Constraint Manager synchronization. By default, `VOLTAGE` and `DIFFERENTIAL_PAIR` are configured to be synchronization properties since these

are commonly captured on the schematic. If it is not your practice to capture these properties on the schematic, remove them from the synch_props.cfg file.

These recommendations are applicable for flat or hierarchical designs.\

Signal Models

Signal models are used for creating Extended Nets (XNets) and differential pairs (Diff Pairs) and for controlling how ECSets are applied on nets. If you are using SI models in the flow, the same models must be made available to both Design Entry HDL and PCB Editor. Models once used must be available to all the applications in the flow. Missing models can cause loss of constraints, although some basic safeguards are provided.

- It is recommended that you assign models using the Model Assignment dialog or by librarian addition to part table files as injected properties.
- Using injected PTF properties is recommended in cases where model assignment need not to be changed or where the librarian would like to provide a default signal model value. It is important to note that any changes made by the schematic or board designer will mask any future changes made by the librarian.
- Models can also be pre-assigned in the chips.prt file, but this prevents any change in PCB Editor for the same reason as mentioned above.
- Models assigned on the schematic canvas are the most flexible. Models can also be defined on symbols but this practice is not recommended.

Capturing Signal Models in Design Entry HDL-Constraint Manager

The following points should be considered before capturing signal models in the design.

Setting the Signal Model Path

- DE HDL and PCB Editor read signal model paths from the signoise.run folder. This folder is created, by default, in the physical folder of the root design.
- In case of hierarchical designs the signal model path needs to be set for each level of hierarchy as the signal model path is written in the signoise.run folder present in the physical folder of the design. To overcome this limitation, we recommend that the signal model files be defined in the variable `SIG_DEVLIBS` in the `env` file in the pcbenv folder. When the value of this variable changes by means of addition or deletion of a new model file, remove the signoise.run folder from the physical folder so that the values defined in the signoise.run folder are not used.

Where to define the SIGNAL_MODEL property:

- Signal models can be assigned in any one of the following three locations:
  - In the chips.prt file located in the chips view
  - In the part table file (`ptf`)
  - On the schematic canvas
- Define signal models in the chips.prt file only if the value is not likely to be changed. This is a likely situation for signal models on established library components such as ICs.
- Define signal models in the Part Table File as an injected property if the signal model assignment is likely to be changed from Design Entry HDL only and not in PCB Editor. Signal models in this case are defined in the library. Therefore, it should be changed by changing the PTF row by Part Manager.
  - If the signal model is annotated on the schematic canvas, the value on the schematic canvas becomes the winning value. The value present in the PTF is no longer used. Even if the PTF value is changed in the file, the schematic value wins.
  - Similarly, if the signal model is changed in PCB Editor, it appears as a component instance property and backannotation on schematic canvas has the same effect as with a property annotated on the schematic canvas.
- Defining signal models on schematic canvas enables you to update them in Design Entry HDL as well as in PCB Editor. This includes Espice models for resistor packs or default generated models. In case the signal models are updated in PCB Editor, make sure that pxlba.txt file has an entry of SIGNAL_MODEL as the component property.

What happens if assigned signal models are not found?

- The Export and Import Physical processes fail and the constraints view is not updated. The flow will not be enabled until the models are found.
- Constraint Manager cannot be launched from Design Entry HDL.

Capturing Constraints

- All constraints should be captured in Constraint Manager.
- Constraints should be captured using ECSets to the maximum extent possible.
- VOLTAGE should be captured on canvas.

Recommendations while Capturing Pin-pair Constraints in Constraint Manager

Before capturing pin-pair constraints in Constraint Manager, make sure that the design is packaged so that all the packaged data is available in Constraint Manager database

Recommendations for Synchronizing Constraints between the Schematic, Constraint Manager, and the PCB Editor

Constraints captured on the schematic canvas can be updated using Constraints Manager in PCB Editor or Constraints Manager in Design Entry HDL. In case constraints are updated in PCB Editor, the back-annotation process will update the schematic canvas value and synchronize it with changes done in Constraint Manager from DE HDL,

Using Place Holders:

- You can use the Attributes form in the following two ways to create schematic placeholders:
  - ❑ Assign a soft property like `$DIFFERENTIAL_PAIR = ?`
  - ❑ Change the visibility of the existing property from None to any other value.
- For non-synch constraints, create placeholders only if it is required to view the constraint on the schematic canvas for plotting. Synch constraints as per definition are already present on the schematic canvas. Therefore, you need not create any placeholders for them.

Recommendations while capturing Differential Pairs on schematic canvas

In case of XNets, it is recommended that the main segment of the XNet be named, and the other segments of the X-Net are left unnamed. If you want to name the other segments of the XNet as well, the net name should be such that it comes lower in the lexicographical order.

Example

```
Segment 1:DDR
Segment 2:<Unnamed Net>
XNet Name:DDR
Segment 1:DDR
Segment 2:ADDR
XNet Name:ADDR
```

As A comes higher in the lexicographical order, the XNet is named ADDR

In case of XNet members, DIFFERENTIAL_PAIR property should be assigned to the main segment net of both the members:

- Assign DIFFERENTIAL_PAIR property only on the main XNet segment. You need not assign the property on the other segments of the XNet. To assign the property on all the segments of the XNet, ensure that while updating the property is updated on all the segments.
- When deleting differential pair member nets, orphaned differential pair properties should be cleaned up so the schematic doesn't have stale values.
- While renaming the DIFFERENTIAL_PAIR property, make sure the new name is updated on each member net. It is also important that the new name be checked for uniqueness. If a differential pair of this name already exists, an error will be generated which you will need to resolve manually
- Do not rename differential pairs on the schematic canvas if the differential pair object contains constraints in Constraint Manager. In the Constraint Manager database, the differential pair objects are identified by their names. If the name of the differential pair object is changed outside the Constraint Manager database, the constraints on the differential pair object are lost.
- Avoid renaming nets and buses that have been constrained in Constraint Manager for the same reason provided for differential pairs.

Capturing Electrical Constraint Sets

Carefully select the pin mapping mode in ECSets. You can set the mapping mode to PINUSE, REFDES, both, or nothing. The mapping mode should not be selected if the same ECSet has to be applied to signals with different pin types

Capturing Constraints on Bus and Bus Bits on the Schematic Canvas

- Bus constraints should NOT be captured on the schematic canvas for the following reasons:
  - ❑ Constraints captured on a bus in the schematic do not appear on the bus object in Constraint Manager. They appear on each bit of the bus.
  - ❑ Constraints captured on a bus in schematic canvas can only be updated by editing them on the schematic canvas and not in Constraint Manager.
  - ❑ Constraints applied to a bus in Constraint Manager are not back annotated to the schematic canvas making the constraint on the schematic canvas stale.

- Constraint Manager provides the expected inheritance between the bus and its bit members. A differential pair can be formed by using the bits of a two-bit bus by applying the DIFFERENTIAL_PAIR property to the two bus bits on the schematic canvas. In this case, if the differential pair name is updated in Constraint Manager, the same value is not backannotated to the schematic canvas.

Recommendations for Working with Lower-Level Blocks

- Assign models at the block level instead of at the top-level design. This ensures that XNet information is available when you capture constraints at a lower-level.
- Define constraints on interface signals at the top-level design. This avoids random selection of a winning value seen when there are conflicts between net aliases across the hierarchy.
- Apply ECSets to interface signals at the top-level design for the reason cited above.
- Interface X-nets should use the winning XNet name as the interface signal name.
- Once interface XNet are constrained at the top-level, do not rename the member signal names.
- The units and precision used at all the levels of the hierarchal design should be consistent to ensure that there are no unit conversion issues. It is recommended that you first set the units at the top-level design so that when you launch Constraint Manager with top-level design as the root design containing a lower-level block, you get the correct units already set.
- Do not manually modify the matched groups created by ECSet application by adding more members to the group.

Restoring Lower-Level Block Constraints

- Constraint Manager provides a restore operation for net and block objects. Care should be taken when restoring an entire block as this will override ALL constraints in the top-level that were captured on lower-level nets.
- Restoring constraints on a single net should always be done using the net-level restore and not the block level-restore.
- The restore from definition feature is available only for local signals and not for global or interface signals.
- Renaming of objects like matched groups and differential pairs at the top level cannot be restored with the names at the lower-level block

Creating the Netlist for PCB Editor

- When you export a schematic design, the Export Physical process creates pst*.dat files (this includes `pstdmlmodels.dat`) in the `packaged` folder, and `devices_dump.dml` in the `physical` folder which contains all the signal models used in Design Entry HDL. The `pstdmlmodels.dat` file is a copy of `devices_dump.dml`. PCB Editor users should have access to the same signal model files used by Design Entry HDL users. Therefore, it is recommended that the `pstdmlmodels.dat` file, along with the pst*.dat files, be passed to layout engineers.
- Performing Export Physical from Design Entry HDL or Import Logic in PCB Editor provides two options for processing constraints: Overwrite and Changes Only. It is recommended that the Overwrite mode be used unless merging concurrent constraint changes in Design Entry HDL and PCB Editor.

  For example if the constraints are captured in Design Entry HDL-Constraint Manager and are changed in PCB Editor -Constraint Manager also before updating the board:

  - In the *Overwrite* mode, Export Physical updates the board file with constraints from Design Entry HDL-Constraint Manager and changes done in PCB Editor-Constraint Manager are lost.
  - In the *Change Only* mode, Export Physical updates the board file with constraints from Design Entry HDL-Constraint Manager and the changes in constraints done in PCB Editor-Constraint Manager are also preserved. In case the same constraint for the same net is updated at both the places, the updated value from Design Entry HDL-Constraint Manager wins.
- Exit Constraint Manager before running Export Physical command from Design Entry HDL

Backannotating from PCB Editor to Design Entry HDL

- Define the `SIGNAL_MODEL` property in the `pxlBA.txt` file under the Component section.
- Signal model (`dml`) files must be available when running Import Physical and while launching Constraint Manager from Design Entry HDL. If the signal model (`dml`) files are not available, Constraint Manager launch fails and invalid ECSet assignments can take place due to pin type mismatches.

Constraints Captured Exclusively in PCB Editor.

- If you do not want to display constraints in Design Entry HDL, use the traditional flow (non-CM flow) in Design Entry HDL. In such a case, Constraint Manager should not be invoked from Design Entry HDL. Also, while performing the Importing Design operation from PCB Editor, you must ensure that the Constraint Manager-enabled flow is not selected.
- Assign signal models only in PCB Editor and remove the SIGNAL_MODEL property from pxlBA.txt file.

Migration from Pre-157 designs

Synchronizing the Old Design for 157 Release

- Always decide beforehand which constraint are to be captured on the schematic canvas. Such constraints should be configured in the synch_props.cfg file.
- The signal models should always be in the path.
- Launch Constraint Manager to synchronize constraints.
- In case the design is from a pre 15.5.1 release, synchronization will first create the constraint folder at each block level and then clean up the constraints from the schematic canvas and OPF. This makes migration to 15.7 release a one-step process irrespective of the previous release (SPB1551 or SPB152).
- Constraints integrity can be verified by creating the *extracta* output file for pre 15.7 board file and the new board file created in 15.7. `extracta` is a utility to extract information from a board file as per the report template used. To extract board file information using extracta, perform the following steps:
  - ❑ Export the pre-15.7 design into a new board with the name pre_157.brd.
  - ❑ Run the command `extracta` with the following syntax to generate a report file pre_157.txt containing all the constraints present in the design. For this, you need to prepare a report file template file. A sample report file template is available at the end of this document.

    ```
    extracta <board_file_name> <report_file_template>
    <output_report_file>
    ```

  - ❑ Migrate the design to release 15.7 or later.
  - ❑ Export the 15.7 (or later) design into a new board with the name 157.brd.

- ❑ Run the command `extracta` with the same syntax to generate a report file 157.txt containing all the constraints present in the design. For this, you need to use the same report file template as used earlier.
- ❑ Compare the pre_157.txt and 157.txt report files to verify that all the constraints have been successfully migrated from pre-15.7 release to 15.7 (and later) release.

Sample Report File Template

```
# This is an extract command file

# generated by the Extracta utility.

#

NET

NET_NAME != ''

NET_SPACING_TYPE

NET_PHYSICAL_TYPE

NET_ELECTRICAL_CONSTRAINT_SET

NET_DIFFERENTIAL_PAIR

NET_STUB_LENGTH

NET_PROPAGATION_DELAY

NET_RELATIVE_PROPAGATION_DELAY

NET_VOLTAGE

DIFF_PAIR_GRP_NAME

MATCH_GROUP_GRP_NAME

XNET_GRP_NAME

END
```

Index