CircuitSpace User Guide

Allegro Interface

November 2008

Doc 003.0-002-000-01

DesignAdvance®

Accelerating the component placement process.™

Warranties and Liabilities

DesignAdvance Systems, Inc. does not assume any responsibility for use of this information. Additionally, this information is subject to change without notice and does not represent a commitment of DesignAdvance Systems. DesignAdvance assumes no responsibility for any errors that appear in this document.

CircuitSpace Copyright and Trademark Information

DesignAdvance and CircuitSpace are registered trademarks of DesignAdvance Systems, Incorporated.

All other company and product names are trademarks or registered trademarks of their respective companies.

Third Party Acknowledgements

Portions of the Licensed Products utilize or include third-party software libraries and other copyrighted material. Acknowledgements, licensing terms and disclaimers for such material are contained in the electronic documentation for the Licensed Products, and your use of such material is governed by their respective terms.

Intellectual Property Protection Notice

The documentation contained herein is the intellectual property of DesignAdvance Systems, Incorporated. It is protected under the patent, copyright, trademark, and trade secret laws of the United States, and the Commonwealth of Pennsylvania.

The contents include proprietary information, which, if disclosed, would cause serious adverse consequences for DesignAdvance. The contents may not be reproduced in any fashion or any medium, except as may be expressly permitted by licensing or other contractual provisions granted expressly and in writing by DesignAdvance. No disclosure, reverse engineering, or other indirect reproduction may be made without the express, written permission of DesignAdvance.

Comments

We welcome your comments. Please contact us at support@designadvance.com or at

DesignAdvance Systems, Inc.
The Crane Building
40 24th Street
Pittsburgh, PA 15222

Phone: 412.434.0601
Fax: 412.434.0605
Email: support@designadvance.com

Table of Contents

Table of Figures

CircuitSpace

CircuitSpace (CS) integrates with a designer's EDA tool to assist with component placement on printed circuit boards (PCBs). CircuitSpace uses sophisticated algorithms to produce solutions that adhere to as many design constraints and practices as practical.

The user can always adjust CircuitSpace's solutions, and guide subsequent solutions to achieve the desired results. As the layout of a board progresses, CircuitSpace accepts the updated board structure and net lists, allowing the user to work with partially defined boards, while continually updating CircuitSpace's solutions.

CircuitSpace makes it easy to duplicate layout decisions and apply layouts to other boards without mapping reference designators. Designed and tested subcircuits, including etch, can be reproduced in a design, or even applied to a different board. CircuitSpace helps manage netlist imports. CircuitSpace will tell you what has changed, and identify the modifications to existing clusters.

This *CircuitSpace User Guide* introduces CircuitSpace concepts and vocabulary, interspersed with example scenarios. We strongly suggest reading *Using CircuitSpace Commands* on page 39 before studying the example scenarios.

In addition, the *CircuitSpace Command Reference* provides detailed usage information for CircuitSpace commands; please review that information before using a command.

Concepts: Clusters

CircuitSpace (CS) enables the designer to manage the layout of the printed circuit board at the level of clusters rather than at the level of individual components. A *cluster* is a reusable sub-circuit of components. A cluster can be as simple as a group of parts or it can capture the hierarchical structure and layout of a sub-circuit on a PCB.

CircuitSpace clusters enable you to manage parts you'd like to work with as a group, typically, a circuit that you'd like to place together. You can move the parts as a group without creating temporary groups, use the cluster shape to reserve space on a board, and save a cluster to copy it to other boards. Clusters also provide a way to manage netlist changes.

The real value of clusters is in their replication. A cluster's component grouping and placement can occur on a board multiple times. For example, many designs contain repeating patterns of sub-circuits. CircuitSpace creates clusters from these sub-circuits. Using clusters, multiple instances of the same cluster can be updated as a group. CircuitSpace also enables the designer to reuse portions of a PCB design for multiple projects. For example, you can apply designed and tested sub-circuits, including etch, to other boards.

CS helps you to organize groups of parts into clusters – subcircuits that you want to manage or layout as a unit. CircuitSpace can capture a sub-circuit's structure on a PCB as the cluster's membership; both components and the underlying interconnect. CircuitSpace helps manage netlist changes, highlighting new parts for clusters in *addenda* clusters.

In addition to a cluster's membership, CircuitSpace records the layout of a cluster's components. CircuitSpace can suggest a rough placement for a cluster, organizing the support circuitry around an IC or IO. CircuitSpace can also capture the existing placement of a sub-circuit on a board, including the cluster's etch.

Members

A cluster is a group of components, and optionally, other clusters. The components can be selected by a user, or derived from schematic information. CircuitSpace instantiates a part and its reference designator's text location for both top and bottom silk screen and assembly layers.

Hierarchy

Clusters that have other clusters as members are *hierarchical.* Hierarchical clusters enable the physical representation of a schematic's hierarchy. A cluster that belongs to another cluster is the *child* of the *parent* cluster.

Master Part

A cluster usually has a *master part.* The master part usually is the part that defines the functionality for the group, such as a power supply, a clock driver, etc. A cluster cannot have more than one master part. Certain CircuitSpace functionally requires a master part. CircuitSpace designates the component with the highest I/O count as the cluster's master part. The designer can assign a different part as the master.

Cluster Shape

The bounding shape of a cluster is an Allegro database object that is created by CircuitSpace. Cluster boundaries are represented as an Allegro shape on the 'Board Geometry' class. Cluster shapes contain all of the pertinent information about a cluster's current state.

Cluster Placement

In addition to a cluster's membership, CircuitSpace records the layout of a cluster's components, including the components' silkscreen. CircuitSpace can capture the existing placement of a sub-circuit on a board. Once a placement is captured for a cluster, CircuitSpace can replicate the placement for other clusters with similar membership.

Cluster Properties

Clusters have the following properties: name, master part, membership and placement tags, template tag, and membership and placement protections. See *Concepts: Advanced Topics* on page 45 for a description of CircuitSpace tags and protection properties.

Creating Clusters

The first step when using CircuitSpace is to create clusters for your board. There are several ways to create clusters. CircuitSpace can automatically generate clusters from design information. You can create copies of clusters from other boards using templates or membership files. You can also create clusters manually.

CircuitSpace automatically generates clusters, using the **Autocluster** command. When creating clusters automatically, CircuitSpace analyzes functional specifications to find groups of components and their connectivity representing clusters. CircuitSpace also creates clusters via copying a source cluster or instantiating a *template*, a generic cluster pattern. Clusters can be created directly by the user, using the **Define Cluster** command.

Automatically

Autocluster finds sub-circuits of logical, localized interconnect using native HDL, CIS, or DxDesigner schematics. CircuitSpace captures the sub-circuit's components and their underlying interconnect. CircuitSpace supports both hierarchical and flat schematic designs as input to its clustering algorithms. For designs that use the Allegro *Room* property, CircuitSpace also creates clusters based on components' *Room* property.

Hierarchical Schematic

When using hierarchical schematics, CircuitSpace creates clusters corresponding to the functional blocks, producing a physical representation of a schematic's hierarchy. Clusters created from hierarchical schematics have hierarchy – parent clusters that contain child clusters as members. Clusters based on the same functional block are assigned a unique CircuitSpace identifier, a membership tag (see *User Tags* for details).

Flat Schematic

Clusters based on flat schematics correspond to pages in the design. If parts are referenced on multiple pages, all those pages are assigned to one cluster.

Netlist with Room Property

For netlists that have parts with the Allegro Room property, a cluster is created for each Room value, and the remaining parts are assigned to a "no-room" cluster.

Netlist

For netlists that do not contain the Room property, one cluster is created that contains all the parts.

Addendum Clusters

When autoclustering a board that already contains clusters, new parts for an existing cluster are added to its *addendum* cluster. Any clusters corresponding to the proposed clusters are not modified. Additional parts to be added to an existing cluster are placed in an addendum cluster. Designers can review the proposed additions, and then merge the addendum with the cluster it belongs to. Addendum clusters only have name and *addendum to* properties.

Display Placement

Automatically-generated clusters are displayed using a display placement, where a cluster's components are arranged in rows. A display placement is always single-sided, regardless of the board's method of assembly.

Manually

Often you will want to capture the layout of a completed board. You can define clusters from existing boards, capturing their placement for subsequent reuse. There are two ways to do this.

Define Clusters by Selection

Use the **Define Cluster -> Manually** command to define clusters for an existing board. You must select the components for the cluster, and optionally, set the master part.

Instantiate Clusters from a List

Another method of creating clusters is **Define Cluster -> From File**. This command does not move parts when creating clusters, thus preserving their placement.

You can create the file using a text editor, and following the format specified on the **Autocluster** reference page, or you can request a membership report for an existing cluster using the **Reports -> Clusters Members** command.

After you have created the membership list, use **Define Cluster -> From File** to create the clusters on the placed board.

Scenario: Clustering a Board

Our example is a small, single-sided board design, consisting of 4 ICs. The designer opens the board file, which has a netlist imported. Next he specifies the location of the schematic using the **Board Parameters** command to open the **Board Parameters** panel, or has the Allegro room properties assigned to components. Then the components are quickplaced using Allegro, as illustrated in *Figure 1: Example Board Quickplaced*.

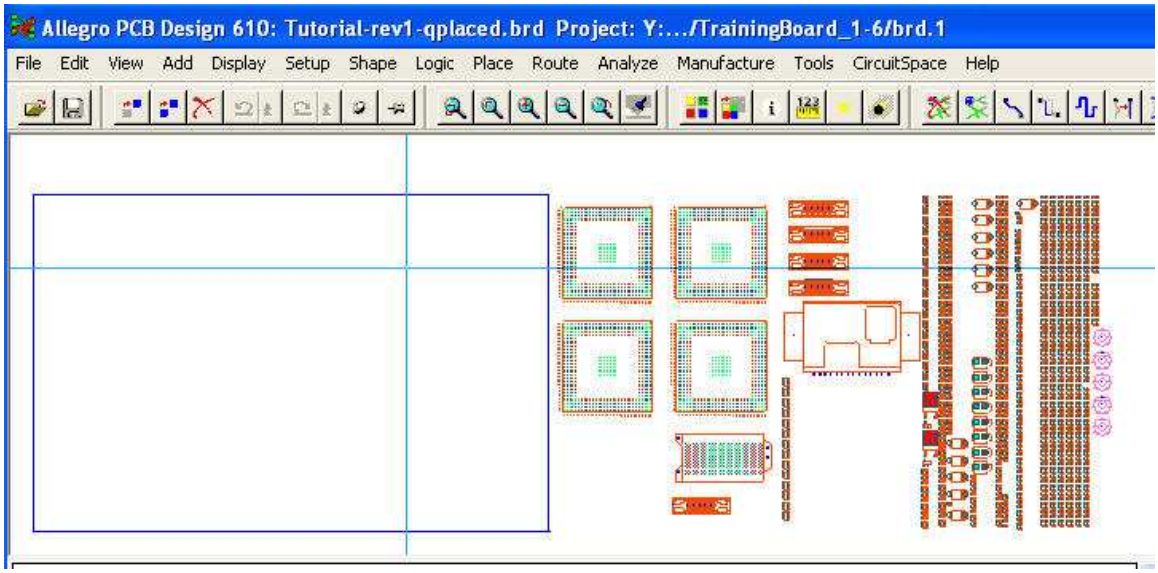

Figure 1: Example Board Quickplaced

The designer then uses CircuitSpace to autocluster the board. CircuitSpace forms functional groups of components based on a hierarchical schematic's functional blocks, a flat schematic's pages, or a netlist with Allegro's Room property. These functional groups are called *clusters* and usually consist of an IC and its supporting circuitry. The resulting clusters are placed at the top of the board. *Figure 2: Example Board Autoclustered* illustrates the solution, which contains 4 *matching* clusters, one for each IC.

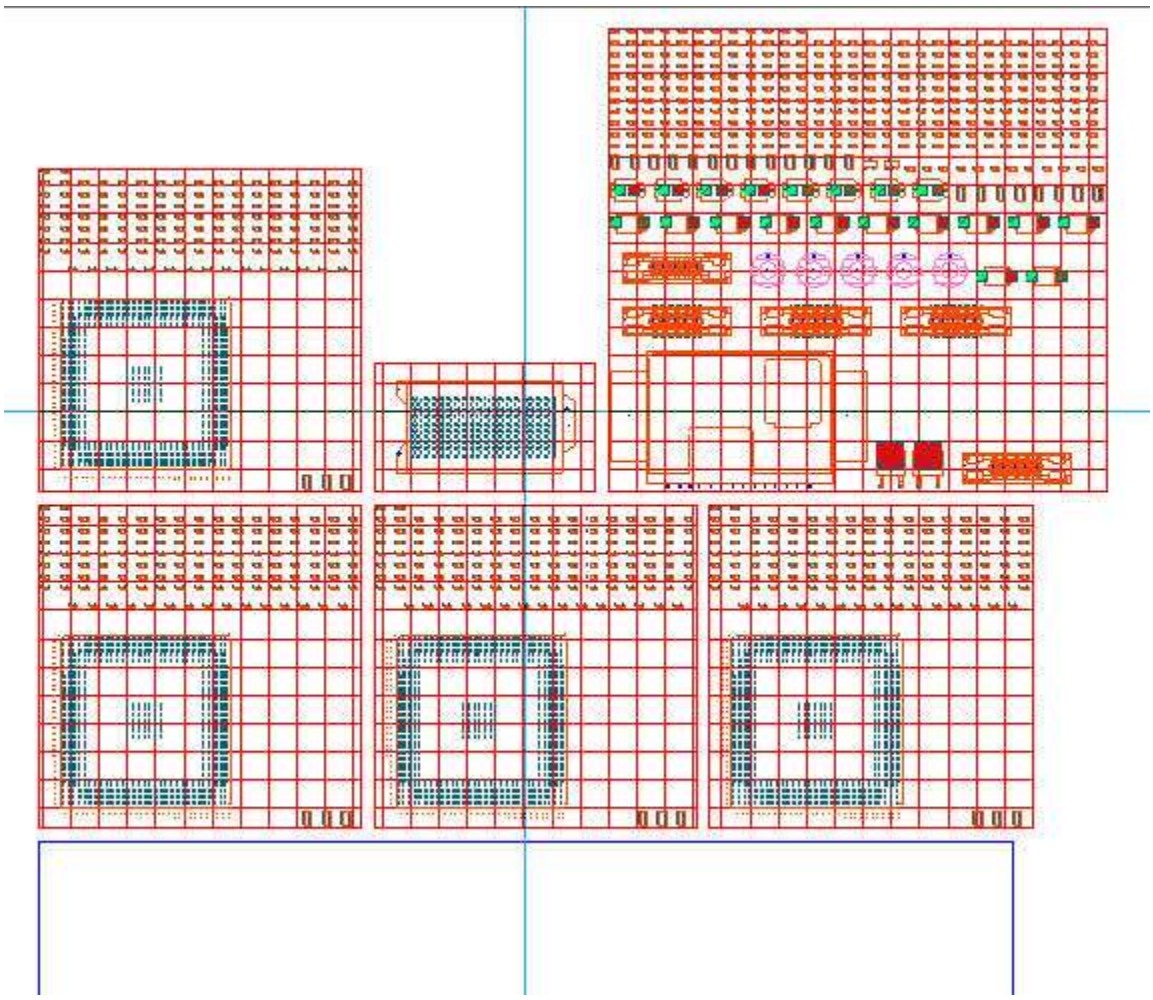

Figure 2: Example Board Autoclustered

These autogenerated clusters use *display* placement – the components are simply lined up in rows inside the cluster boundary. The designer uses the **Modify Properties** command to view a cluster's properties: name, master part, protections, and tags, illustrated in *Figure 3: Cluster Properties*.

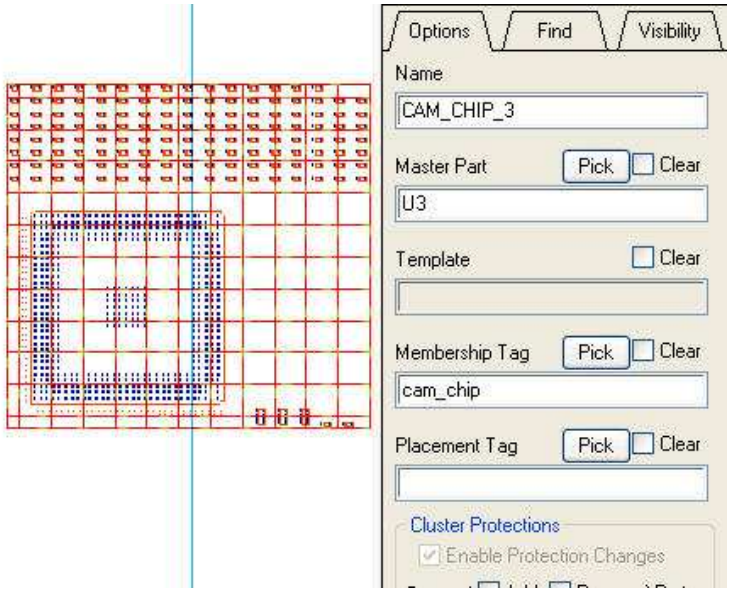

Figure 3: Cluster Properties

The four clusters created from the same functional block are assigned the same membership tag.

In the next scenario, the designer creates a rough placement for one of the IC clusters using the **Autoplace Local** command.

Concepts: Automatic Component Placement

Autoplacement produces a rough placement for a cluster's members, simplifying the designer's manual placement task. After creating clusters, CircuitSpace's **Autoplace Local** command can perform rough placement, where CS places components relative to one another within a cluster. With CircuitSpace's easy-to-use set-up, you simply define a few parameters for CircuitSpace to start the process. Based on those parameters, CircuitSpace then places the components within a cluster.

Rough Placement

A cluster must have a major part, such as an IC or an IO, in order to be autoplaced. For rough placement, CircuitSpace places the major part's support circuitry in rows around it. Bypass and filter capacitors are placed closest to the IC or IO, then resistors, then other parts except bulk caps, and finally, bulk caps are placed in the outmost perimeter.

CircuitSpace arranges components in rows to preserve routing channels. Parts are arranged to minimize the number of rows. Component locations and orientations are selected to minimize airwire crossings and lengths.

Spacing

CircuitSpace creates a *fanout estimate* for each component. *Fanout estimates* help determine a component's required placement area. Fanout is estimated based on the size and location of the component pads and the following spacing rules:

- Fanout type: Thru hole
- Fanout Directional Preference: Out and Staggered
- Pad to Via: 5 mils
- Via to Via: 6 mils
- Pad to Pad: 7 mils
- Trace to Pad: 5 mils
- Trace to Trace: 6 mils
- Trace width: 5 mils

- Via Diameter: 22 mils
- 0 tracks per channel

CircuitSpace uses following rules for double-sided boards:

- 60 mil bottom height restriction for double-sided boards

For each row, parts are placed to satisfy CircuitSpace's spacing rule, lie on the placement grid, eliminate place-bounds intersections, and satisfy pad-to-pad spacing.

Capacitors

Bulk caps are distinguished from other capacitors on the basis of their size, and inferred power and ground nets. Capacitors with an area greater than 40,000 square mils are considered bulk caps.

Bypass and filter capacitors are placed closest to the major part; bulk capacitors are placed last.

Multiple ICs

For clusters with multiple ICs, CircuitSpace places the support circuitry around each IC, forming a *primitive cluster* for each IC. Then the primitive clusters are tiled in a rough square.

Placement Options

You can specify the following options for the **Autoplace Local** command.

Placement Grid

A placement grid must be specified for the board before using the **Autoplace Local** command. Specify the placement grid, in terms of x-grid-size and y-grid-size, and the unit of measure using the **Board Parameters** command to open the **Board Parameters** panel. Components will be placed such that their distance from their respective IC is a multiple of the grid-size.

Double-sided Placement

You must specify whether the board is single- or double-sided for the **Autoplace Local** command. If double-sided placement is specified, parts may be placed back-to-back.

Special Cases

Fixed Parts

If a cluster has one fixed part, the cluster will be placed relative to that fixed part.

If a cluster has multiple fixed parts, it will be placed relative to one of the fixed parts. The fixed part used is chosen in the following order: the master part, the IC or IO with the most pins, and finally the non-IC or IO with the most pins. The other fixed parts are not moved.

If a cluster has multiple fixed ICs, the support circuitry will be placed around each IC. Any non-fixed primitives are tiled around the fixed primitives. Parts from different primitive clusters may overlap if their ICs are fixed too close to each other.

Protected Boundary

If a cluster's shape and position is protected, the cluster shape is not changed. Parts may overlap or be placed outside the boundary.

Unconnected Parts

If a part does not have a routable path through other cluster members back to a major part, it is not placed, but tiled beside the primitive clusters.

Scenario:
Autoplacement

Returning to our example board, the designer autoplaces one of the clusters. Automatically generated clusters use a simple display placement, shown on the left in *Figure 4: Autoplacing a Cluster*. CircuitSpace's autoplacement provides a rough placement the designer can manually refine, shown on the right.

The support circuitry is grouped around the IC. Bypass and filter caps are placed closest to the major part; bulk caps are placed farthest from it. CircuitSpace determines bypass and bulk capacitors based on their shapes, sizes, and inferred power and ground nets.

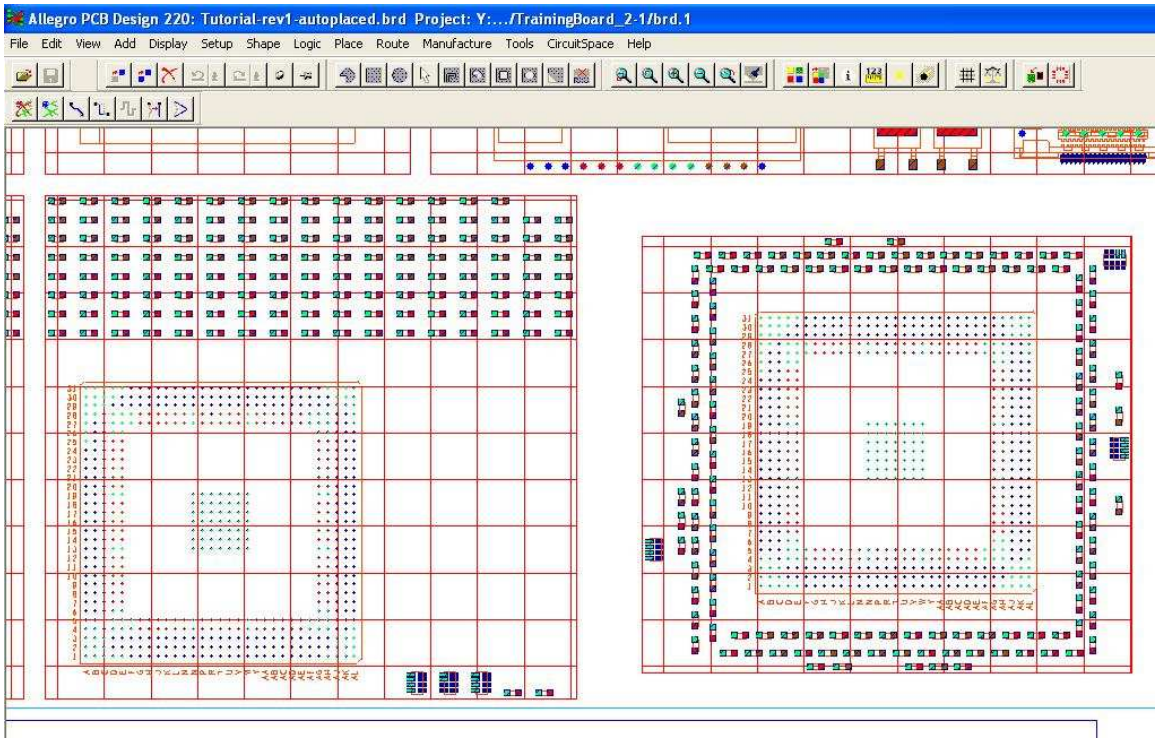

Figure 4: Autoplacing a Cluster

In the next scenario, the designer will adjust the placement of one the clusters, then replicate it, efficiently re-using his placement work and maintaining the matching clusters.

Concepts: Matching Clusters

A cluster's circuitry can occur multiple times on a circuit board. To create "matching" clusters for this circuitry, CircuitSpace instantiates new clusters or modifies existing clusters to match the cluster specified by the user, depending on the user request. CircuitSpace instantiates a source cluster's circuit topology using components on the board.

Much of CircuitSpace's functionality is based on matching two clusters, or a cluster and a template. For example, when replicating a cluster, new clusters are created that match the source cluster in members and connectivity. When propagating a change, target clusters are modified to match the source cluster.

Types of Matches

Any cluster to be matched, either source or target, must have a major part, usually an IC or IO. Once clusters meet that basic criterium, clusters are said to match when they have the same members with the same connectivity. CircuitSpace uses two levels of match: *strict*, and *partial*, and two levels of connectivity: *strict* or *equivalent*, each further defined below.

CircuitSpace always returns *strict* matches using *strict* connectivity. You can request matches that use *equivalent* connectivity. You may also specify that CircuitSpace returns *partial* matches, with fewer parts or weaker connectivity than a strict match.

Strict Matches

By default, CircuitSpaces uses strict matches with strict connectivity – all the parts match and all the nets match. In more details, a cluster matches a source cluster or template strictly when the following criteria are met:

One-to-one part correspondence

Each component in the source has a corresponding component in the target.

Components match if they have the same symbol and electrical part type, as defined in the Reference Designator Mappings. Parts match if they have different device types.

Notes:

- Electrical part type is based on the Reference Designator Mappings defined by the **Board Parameters**.

- Capacitors must have the same capacitance values to be considered a match.
- When propagating membership or placement, a “matching” target cluster can have extra parts due to its protections. This exception will be noted in the match report.

Alternate Symbols

When a source cluster uses a primary symbol, its targets must have the same primary symbol in order to match. However, an alternate symbol in the source can be matched with its primary symbol in the target, if the alternate symbol is not available for the target.

Strict Connectivity: One-to-one net correspondence

Each net in the source has a corresponding net in the target that is on corresponding pins of the corresponding parts. The source cluster’s connections to the outside correspond to the target cluster’s connections to the outside.

For nets connected to a discrete array, such as a resistor array, resistor net, capacitor net, etc, all the nets connected to one discrete component are matched as though they are one composite net.

CircuitSpace allows the following exceptions:

- Single-pin nets and unused pins are ignored.
- Single-part nets are ignored, that is, a component’s internal connections are ignored.

These exceptions will be noted in the match report.

Equivalent Connectivity

You can request matches that use *equivalent* connectivity. Equivalent connectivity uses pin-to-pin correspondence, a less strict connectivity than net-to-net correspondence.

Pin-to-pin correspondence is defined as for each set of pins connecting two parts in the source cluster, the same pins in the target cluster must connect the corresponding parts. The source cluster may have subnets that the target cluster does not. However, the same pins must be connected.

Also, passive components and arrays, like resistors, capacitors, ferrite beads, or resistor arrays, may have swapped pins.

Figure 5: Match Using Equivalent Connectivity illustrates a match using equivalent connectivity between Cluster A and Cluster B. Cluster A and B have identical components. The same sets of pins are connected in A and B. The pins on Net J are connected in cluster B on Net M, as are the pins on Net K, so there is an equivalent connectivity match.

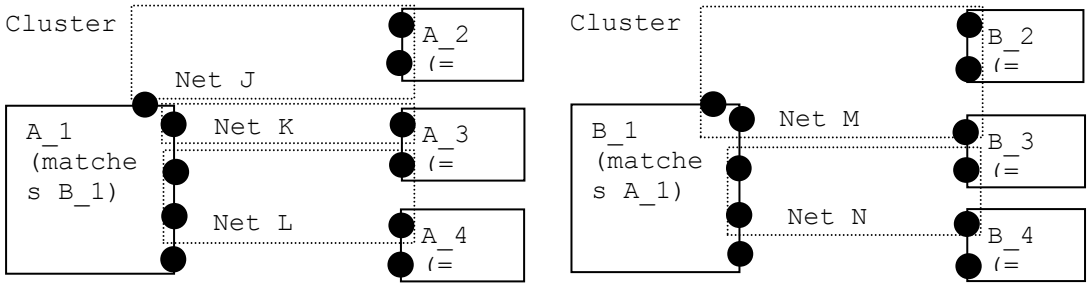

Figure 5: Match Using Equivalent Connectivity

The association between the nets is not one-to-one. Note too that Cluster B matches Cluster A, but Cluster A does not match Cluster B; equivalent connectivity is not commutative.

Partial Matches

You may also request partial matches. Partial matches relax the membership and, optionally, connectivity rules. You can specify that partial matches use either the same connectivity as strict matches, or accept incomplete nets.

Subset of Matching Parts:
For partial matches, the source and target only need to have a subset of matching parts; either cluster could have unmatched parts. In fact, a partial match is only required to have the same major part as the source.

Weaker Connectivity:
You can choose to use weaker connectivity criteria. When using the weaker matching rules, only 80% of the nets connecting two parts in the target must match the source cluster.

Create Generic Symbols for Missing Parts:
If using partial match, you can choose to create generic symbols for missing parts. If a part is not available in the scope, CircuitSpace uses a generic symbol. You can use the **Sync Clusters** command to match these generic parts with parts in a scope.

CircuitSpace Commands

The following commands use CircuitSpace's matching functionality:

Replicate: CircuitSpace creates clusters that match the source cluster, using the parts available in the scope.

Modify Membership: CircuitSpace modifies selected *target* clusters to match the source cluster, using the parts available in the scope. See *Concepts: Modifying Clusters, Propagating Changes* for details.

Propagate Placement: CircuitSpace modifies selected *target* clusters to match the source cluster's placement. See *Concepts: Modifying Clusters, Propagating Changes* for details.

Apply Templates: CircuitSpace creates clusters that match the selected template, using

the parts available in the scope.

The matched clusters contain matching components. Any exceptions are noted in the match report. For usage details, please see the applicable command reference page.

Match Options

When replicating, applying templates, modifying membership, or propagating placement, you have several match options.

- Use equivalent connectivity for strict matches.
- Use partial match, in addition to strict match.
- Use a weaker connectivity rule for partial matches and allow incomplete nets.
- Create generic symbols for missing parts.

Match Reports

CircuitSpace provides detailed information for matching. The progress report summarizes the results: types of matches and errors. CircuitSpace also produces a detailed csv file. Below is an example of the progress match report.

```
====================
RQ000005 Progress:
--------------------
Applying template '1','example|cam_chip_4_additional_parts', with '161'
members and master part 'U3'
Finding exact [STRICT] matches
Found a match with 161 members and master part 'U3'
Placement matching was successful

-- Summary Report --
There were 4 matches found for 161-part source
example|cam_chip_4_additional_parts of which 1 was [STRICT] and 3 were
[PARTIAL].
  example|cam_chip_4_additional_parts - 1 [STRICT] match.
     U3_CLUSTER_1 matched 161 parts out of 161
  example|cam_chip_4_additional_parts - 3 [PARTIAL] matches.
     U4_CLUSTER_1 matched 156 parts out of 161
        - 5 parts in the source example|cam_chip_4_additional_parts
were not matched. R181  R182  RA12  RA14  RA15
     U2_CLUSTER_1 matched 155 parts out of 161
        - 6 parts in the source example|cam_chip_4_additional_parts
were not matched. R167  R168  RA12  RA14  RA15  RA18
     U1_CLUSTER_1 matched 153 parts out of 161
        - 8 parts in the source example|cam_chip_4_additional_parts
were not matched. R167  R168  R169  R170  RA12  RA14  RA15  RA18
Writing report to: C:\pcbhome\CSRun\cs_tutorial_start\RQ000005.csv
```

Scenario: Replicating a Cluster

After adjusting the rough placement of a cluster, the designer replicates it, selecting it as the source cluster. In *Figure 6: Replicating the Placed Cluster*, the source cluster is located within the board shape. The designer changes to **Select Scope** mode and in the **Parts Selection** pane of the *Options* panel adds **Clustered Off Board** parts to the scope.

CircuitSpace creates as many instances of the source cluster as possible from available parts in the scope.

In this example, parts from the three *matching* clusters, which were included in the scope, are used to create new clusters that match the cluster selected as the source. In effect, all the parts from the original clusters are removed and used in the replicants. When a cluster is emptied in the scope, the cluster is deleted. The replicated clusters are displayed at the bottom of the board. The replicated clusters are assigned the same membership and placement tags as the source cluster.

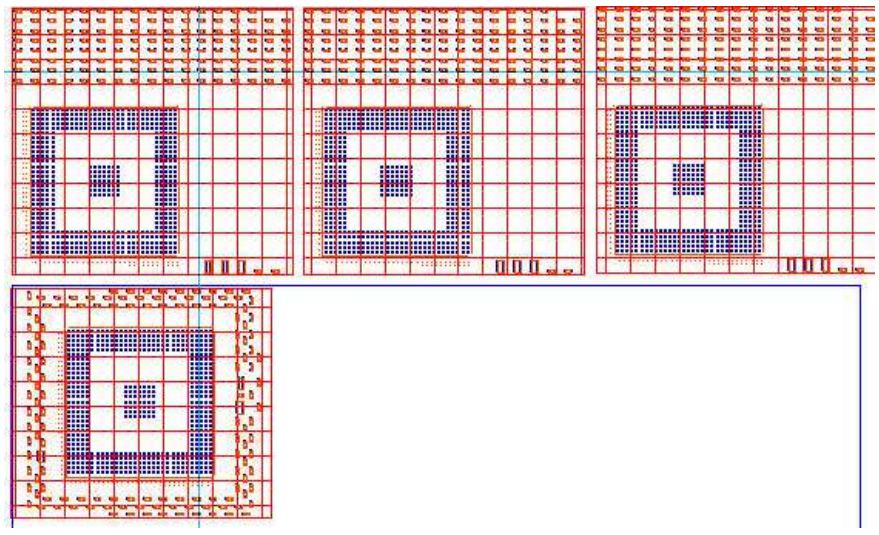

Figure 6: Replicating the Placed Cluster

The replicants match the source cluster strictly; all the parts were available for each of the clusters.

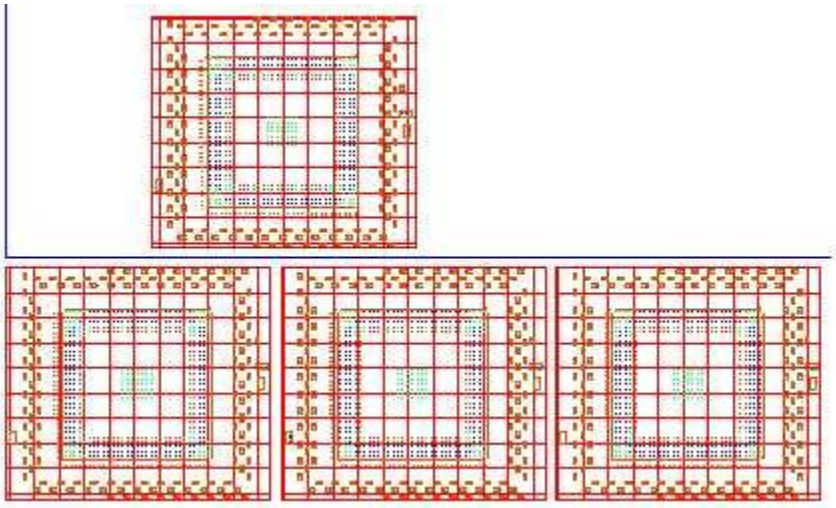

Figure 7: Replicated Clusters

This information is contained in the progress report, available on the **Batch Control** panel, illustrated in *Figure 8: Replicate Match Report*. If parts were not available, and a partial match had been requested, the missing parts would be listed on the match report.

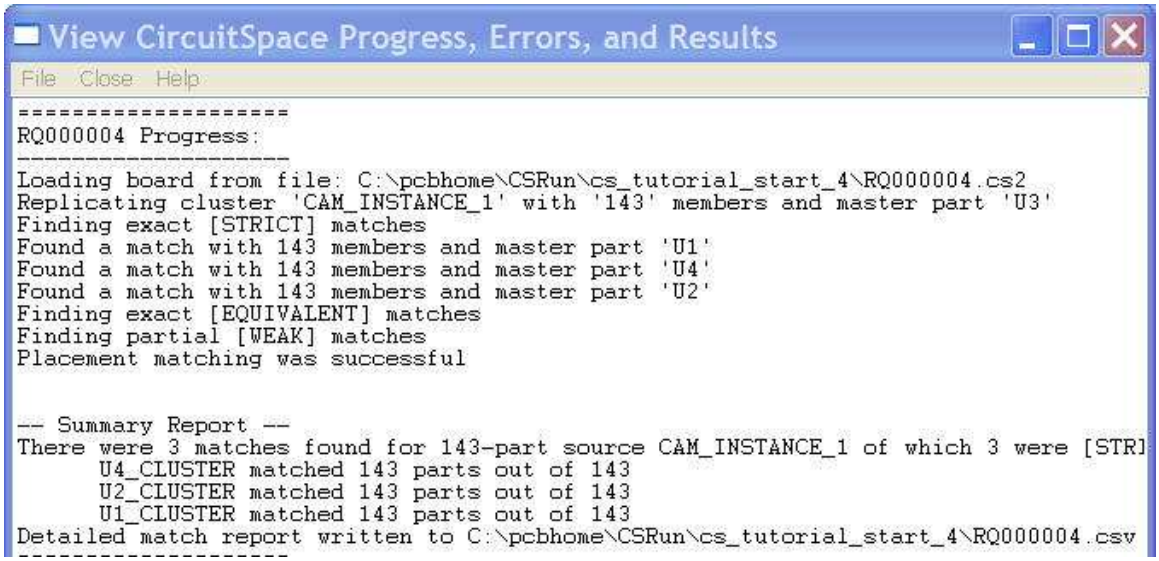

Figure 8: Replicate Match Report

In the next scenario, the designer receives a revised netlist, containing additional parts.

Concepts: Managing Modifications

Designs are revised over time, during the course of a project, and from board to board. CircuitSpace identifies changes and helps you manage them. CircuitSpace provides several tools to help you do this, including the following:

- *Checkpoint reports* describe a board's current state, enumerating clusters, members, and nets.
- *Comparison reports* identify the differences between a checkpoint report and a board, or another checkpoint report. For example, when a modified netlist includes additional parts and nets, those additions are highlighted on the comparison report.
- *Addendum clusters* contain new parts for existing clusters. You can review the additions to a cluster's membership before modifying the actual cluster.

When you receive a revised netlist, first, create a checkpoint for the current board. Then import the modified netlist, and schematics, if applicable. To see a summary of the modifications, create a comparison report against the modified netlist. To see changes for clustering, autocluster the board.

Checkpoints

CircuitSpace can create a *checkpoint* file for the current board with the **Reports > Create Checkpoints** command. A checkpoint captures a board's state, including information about its components, nets, and clusters. See the **Reports** reference page for usage details.

Comparison Reports

The **Reports > Compare Checkpoints** command generates a checkpoint comparison report that details the differences between checkpoints or a checkpoint and the current board. It includes information on components, nets, and clusters. Each category details additions, deletions, and modifications. In addition, modifications to clusters are broken down into modified master part, and the addition, deletion, or modification of cluster members. See the **Reports** reference page for usage details.

Addendum Clusters

After importing a modified netlist, the designer can autocluster the board to see the

changes. Existing clusters will not be modified or moved. Deleted parts will be removed from the board. New members for an existing cluster will be placed in an *addendum*, which is a special type of cluster. Addendum clusters are containers of new parts for an existing cluster, enabling modifications to cluster membership to be managed easily. Addenda have an *addendum to* property, identifying the cluster to which they belong. See the **Autocluster** reference page for details.

CircuitSpace provides specialized sub-commands for moving and merging addendum clusters. See the **Merge Clusters** and **Move Clusters** reference pages for details.

Revised Netlists

A revised netlist can contain new parts or nets, modifications to symbols and circuits, and may delete parts or nets. The comparison report identifies the modifications to a netlist.

New Parts

New parts are placed in the queue and clustered if they are part of the scope. If a new part is an addition to an existing cluster, it is placed in an addendum.

Revised Symbols

If a symbol changes from the original netlist to the revised netlist, but its reference designator remains the same, the part will be placed back in the queue, and will loose its cluster membership information. The changed part is treated as a "new" part by CircuitSpace.

Deleted Parts

Symbols that are deleted or changed from a previous netlist are silently removed from the board.

Nets

Any modifications to the board's nets, such as new nets, deleted nets, or revised nets, are listed in the comparison report.

Scenario: Using a Modified Netlist

The designer receives a revised netlist for our example board. Before importing the modified netlist, the designer creates a checkpoint for the current board with the **Reports > Create Checkpoint** command. He then imports the new netlist, and schematics, if available. To easily see a summary of the modifications, he requests a comparison report between the checkpoint for the original board and the new netlist, shown in *Figure 9: Comparison Report*.

Summary Information

Item	Base Checkpoint	Delta Checkpoint
Components	943	952
Components Added		25
Components Deleted		16
Components Modified		10
Nets	520	516
Nets Added		6
Nets Deleted		10
Nets Modified		56
Clusters	6	6
Clusters Added		0
Clusters Deleted		0
Clusters With Modified Master Components		0
Clusters With Added Members		0
Clusters With Deleted Members		1
Clusters With Modified Members		0

Figure 9: Comparison Report

After reviewing the changes, the designer autoclusters the board. Autoclustering does not modify or move existing clusters. The deleted part is removed from the board. The new parts are added to addenda clusters, placed along the left of the board. *Figure 10: Addenda Clusters Containing New Parts* shows an addenda cluster for each of the four matching clusters. Each addenda contains three new parts. Clicking on a **Delta**

Checkpoint column link takes you to the details for that category.

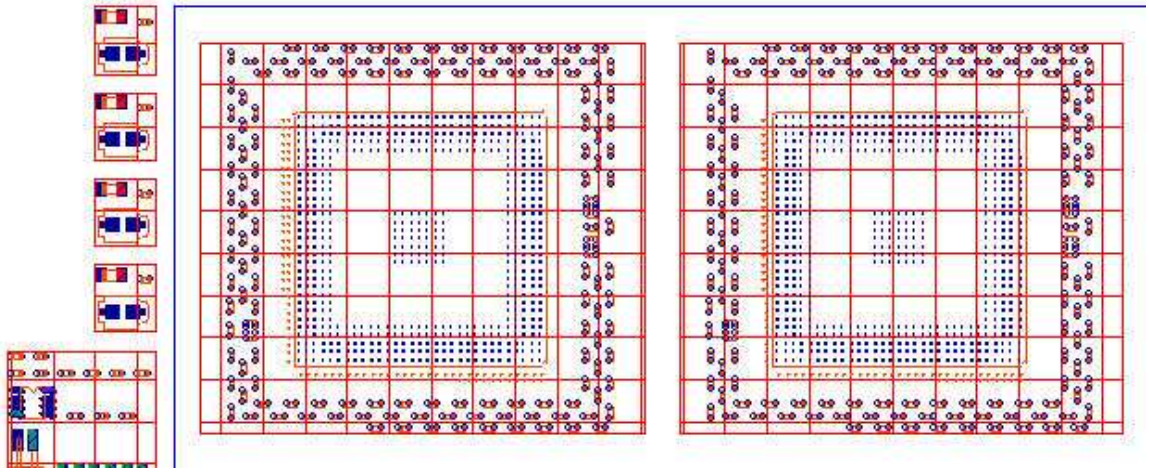

Figure 10: Addenda Clusters Containing New Parts

Reviewing the addenda clusters, the designer merges one addendum with the cluster it belongs to, using the **Merge Clusters > Merge Addenda** command.

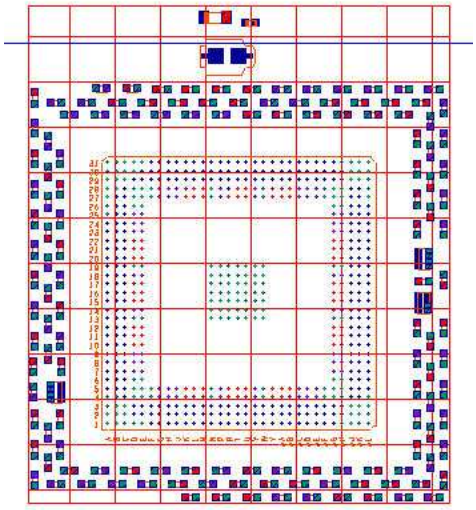

Figure 11: Cluster with Merged Addendum

Next the designer will place the new parts, then easily apply those modifications to the other clusters.

Concepts: Modifying Clusters, Propagating Changes

You can modify a cluster's membership, its placement, and its etch.

- Membership: Using the **Modify Membership** command you can add and remove components from a cluster. After you modify a cluster, you can propagate the change to other clusters.
- Placement: Using the **Propagate Placement** command, you can copy a cluster's placement to other clusters that have the same master part.
- Etch: Using the **Modify Membership > Edit Etch** command, you can add and remove etch from a cluster.

When propagating changes, CircuitSpace determines how to modify a target cluster so that it matches the source, using the rules described in *Concepts: Matching Clusters*.

Any changes to text location and text size on the reference designator layers are propagated to target clusters.

Modify Membership

Using the **Modify Membership** command, you can change a cluster's membership by adding and removing selected components from a cluster. When a cluster's membership is changed, CircuitSpace recalculates the cluster boundary.

As part of the modification, you can choose to propagate the change to other clusters. After successfully propagating a change, the target clusters match the source cluster in membership, protections, tags, and if possible, placement. For example, a designer adds a component to the U1_cluster, and propagates that change to two target clusters. The corresponding parts are available in the scope, and are added to the target clusters. If the needed parts are not available, the default behavior is that the targets will not be modified, and an error will be written. If you use the *Create Generic Parts* option, CircuitSpace will create generic parts for any parts that are not available in the scope. The generic parts will be noted in the match report.

Note that membership changes usually require placement changes. We recommend moving the parts to be added or removed before running the **Modify** command. For example, before adding a new part to a cluster, move it near or within the cluster's boundary. Then the new part's placement is propagated to the target clusters, if the targets have the same placement tag. (If the targets have a different placement tag, the new part is placed outside the cluster boundary.) See *User Tags* on page 46 for further

information on CircuitSpace tags.

Propagate Placement

Likewise, you can modify a cluster's placement, and propagate those changes to other clusters. After changing a cluster's placement using Allegro commands, you can propagate the changes to the selected target clusters. CircuitSpace does not recalculate the cluster boundary when propagating placement changes. Because the cluster's boundary is also propagated, use the **Autosize Cluster** command to modify it, if needed, before propagating. Parts in the target cluster are given the same placement and orientation (with respect to the master part) as the corresponding parts in the source cluster.

Modify Etch

You can add or remove etch from a cluster. Like changing a cluster's membership, this does not create or delete etch; it modifies the etch that is assigned to a cluster. Changes to a cluster's etch are not propagated. See *Etch* on page 48 for further details.

User Tags

For information on how CircuitSpace manages user tags when modifying clusters and propagating changes, see *User Tags* on page 46.

Scenario: Propagating Changes to Other Clusters

Next, the designer refines the placement of the new parts that were merged into one of the clusters. Now he is ready to apply those changes to the other matching clusters. Using the **Modify Membership** command, the designer can add the new parts from the addenda clusters to their owners, and copy the placement from the source cluster. *Figure 12: Placed Parts* illustrates the placement of the new parts, previously merged from the addenda.

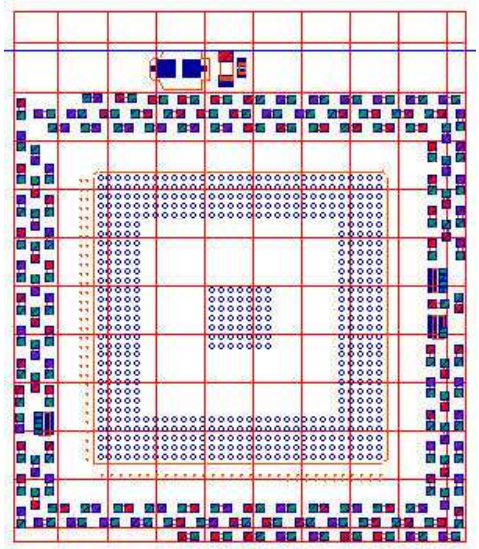

Figure 12: Placed Parts

Now the designer calls **Modify Membership > Revise and Propagate**, and selects the cluster with the new parts as the source. He then chooses **Choose Targets** from the right mouse button menu, and selects the cluster's membership tag in **Cluster Select** pane on the *Options* panel. This selects the other three matching clusters as targets. Then he selects **Scope** mode, and selects the three remaining addenda clusters as the scope.

The parts to be added are available in the scope, and are added to the matching clusters with the source's placement.

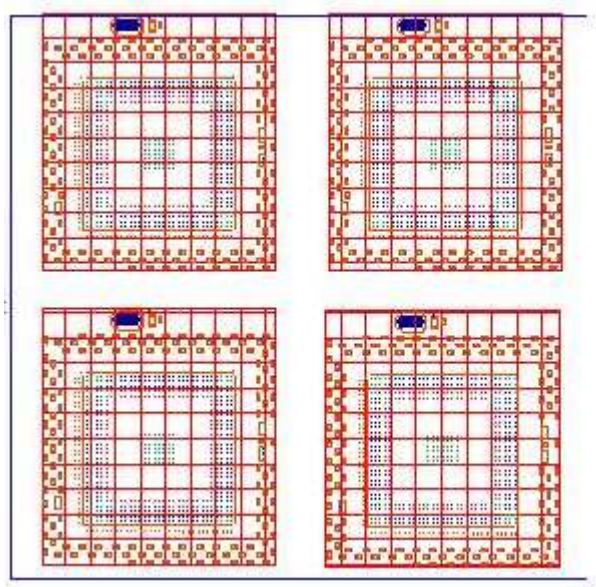

Figure 13: Modified Targets for Modify Membership

In the next scenario, the designer will use a template to create the matching clusters.

Concepts: Templates

Reusable Circuits

Clusters represent the design rules for a set of components. A *template* abstracts this information into a physical design reuse (PDR) pattern for creating additional clusters containing the same type of components with the same generic connectivity and the same placement. Templates store this abstracted information in a file saved independently from Allegro. Once saved, templates can be applied to any board, provided the sub-circuits described by the template are present on the target board.

A template defines a cluster's *membership* -- the components in a cluster and their connectivity. It also defines a cluster's *placement* -- the layout of a cluster and the placement of its components, relative to a master part. Thirdly, it defines a cluster's *etch*. This information can be applied to any board with similar circuits and the same master parts available.

Templates are true generic objects; CS maintains generic reference designators and net names. When a template is applied to a board, its clusters adopt the correct reference designators and net names.

Saving Templates

A template is a file containing a cluster's membership, placement, and etch. When saving a template, a cluster must have a master part.

Template Library

The template library is specified in **User Settings**. By default, this is `$home/CSTemplates`.

Directory Structure

Templates are organized in subdirectories, usually according to membership. You specify the subdirectory and the file name when saving a template.

Etch

If the cluster has etch, it is saved in the template. The etch, vias, and fills within the cluster boundary are saved. See *Etch* on page 48 for details about etch.

You can apply a template's etch to a cluster using the **Apply Template Etch** command.

Applying Templates

CircuitSpace creates clusters matching the specified templates. The matching clusters will have the same master part, same members, and same connectivity as the template. When applying templates, CircuitSpace instantiates matching clusters using the rules described in *Concepts: Matching Clusters*.

Template Tags

A *template tag* is a CircuitSpace identifier assigned to clusters that were created from the same template.

Templates do not have membership tags and placement tags. When a template is applied, the template's subdirectory is used as the membership tag, and the template name is used as the placement tag. Any existing membership and placement tags are overwritten. See *User Tags* for more information on CircuitSpace user tags.

Applying Template Etch

The etch within a cluster's boundary is saved as part of its template. A template's etch can be applied to a cluster on the board. Etch is applied separately because cluster creation occurs at the start of the design process, while etch is designed after component placement. See *Etch* on page 48 for details about etch.

Scenario: Saving Templates from a Placed Board

Next, the designer creates a template from a circuit on a finished board, thus enabling the placement work to be reused. First, he creates a cluster representing the finished circuit. There are two methods to do this: define the cluster manually, and define the cluster using a list. Then he must save the cluster's template.

To define the cluster manually, the designer opens the finished board and locates the IC for the circuit. Using the **Define Cluster > Manually** command, he selects the parts comprising the circuit, and then selects the IC as the master part.

The other method of creating the cluster uses **Define Cluster -> From File**. This command does not move parts when creating clusters, thus preserving their placement. This takes several steps.

To generate the cluster membership list used by **Define Cluster -> From File**, first the designer autoclusters the board. This creates clusters, but does not maintain the placement of the components. Then, he requests a membership report using the **Reports -> Clusters Members** command. He saves the report to a text file; this is the membership list.

After creating the membership list, the designer is ready to recreate the clusters on the placed board, using **Define Cluster -> From File**, which preserves the part placement. The designer reopens the placed board and calls **Define Cluster -> From File**. He specifies the cluster membership report file created in the previous step. The **Define Cluster -> From File** command creates the clusters from the file on the board, preserving their part placement.

Now he calls **Save Template** and selects the cluster. The cluster, including its etch, is saved to the template file.

Scenario: Applying Templates

Now that the designer has a template from a placed board, he wants to apply it to the example board. Calling up the original example board, the designer once again specifies the location of the schematic using the **Board Parameters** command. The components are quickplaced using Allegro. The designer then selects a template to apply to the board, using the **Apply Templates** command. *Figure 14: Apply Templates dialog* illustrates the **Apply Templates** dialog.

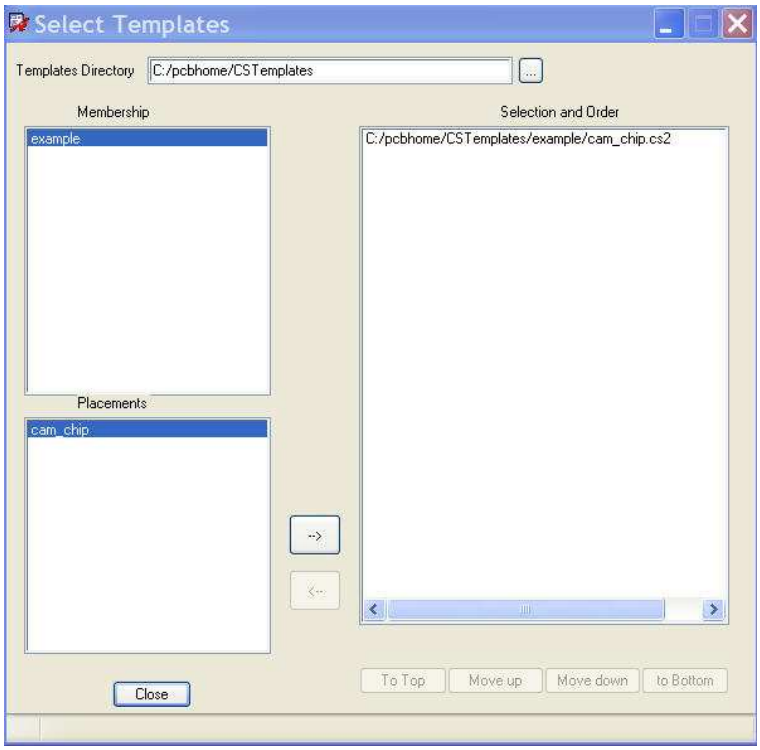

Figure 14: Apply Templates dialog

Four matching clusters are created from the available parts in the default scope. Applying the solution, shown in *Figure 15: Apply Template Results,* the new clusters are displayed below the board. The matching clusters have the same members, master part, relative part placement, and cluster shape as the template. Exact matches are assigned the template tag.

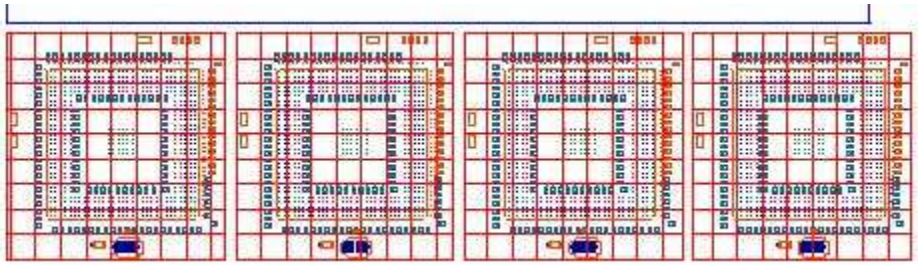

Figure 15: Apply Template Results

The designer reviews the match report, illustrated in *Figure 16: Applied Templates Match Report*.

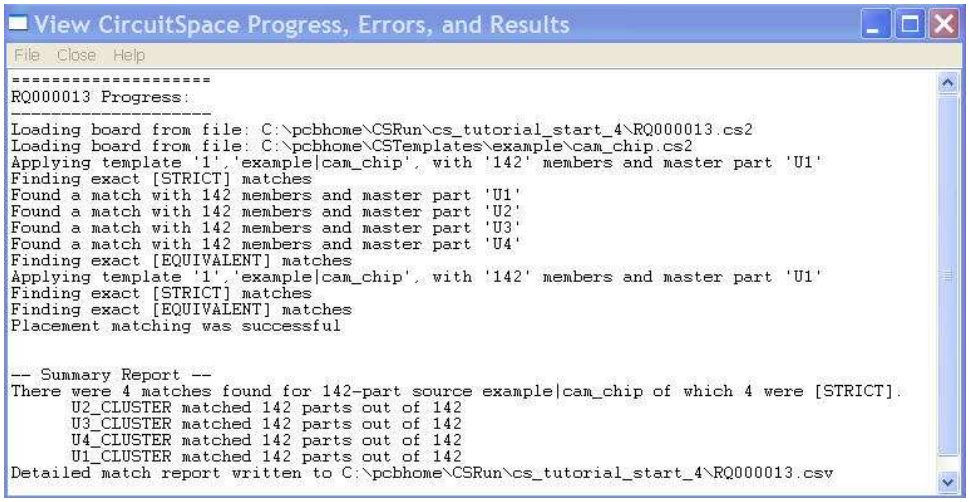

Figure 16: Applied Templates Match Report

The designer now reopens the example board, and applies the etch to the clusters, once again from a template. Selecting the **Apply Template Etch** command, the designer confirms the template name, and then selects a matching cluster. The etch is applied to the selected cluster. He then selects the next cluster, until the four clusters now have the etch and vias from the template.

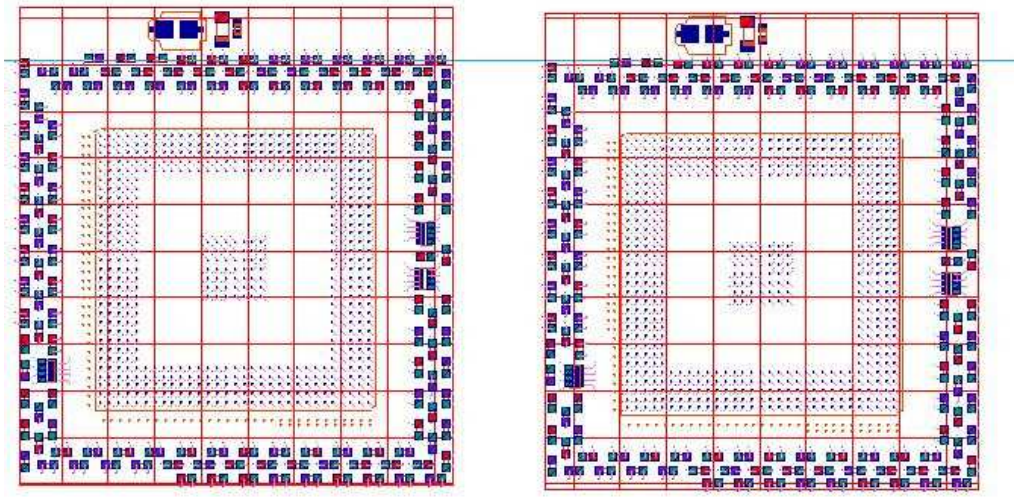

Figure 17: Etch Applied from a Template

Concepts: Cross Probing

DesignAdvance's Cross Probing enables you to select objects in either Allegro PCB Editor/Physical Viewer or Adobe's PDF Reader and have the corresponding objects selected in the other application. Cross probing is bi-directional: selecting components, nets, or pins in one application selects the corresponding object in the other application.

Initializing Cross Probing

DesignAdvance's cross probe tool includes plugins for Adobe Reader and Allegro. The CircuitSpace cross probing plug-in is added to Adobe Reader when you install CircuitSpace, or you can install it manually.

To initialize cross probing, you must start cross probing in Allegro, then turn on Adobe's cross probe selection tool.

In Allegro, CircuitSpace adds the following commands that take objects selected in Adobe Reader as input:

Define Cluster Keep Positions

Define Cluster Reposition (Allegro PCB Editor only)

In Adobe PDF Reader, CircuitSpace adds the following commands to the CircuitSpace menu:

Cross Probe Selection Tool: turns cross probing on. You can choose to navigate by object or by page.

- *By object* displays the first object in the selection set, centered and zoomed. Selection Go then displays the next or previous object in the set.
- *By page* displays all the objects in the selection set that are on the same page. Selection Go then displays the next or previous page in the set.

Selection Go: Activates if more than one object or page is selected. Moves to the next or previous object or page in the selection set.

Selection Type: Choose the type of object to select, either components, nets, or pins. The toolbar icon corresponding to the selected object type is blue. The same type of object must be selected in Allegro's Find Filter. The default object type is components.

Freeze Zoom Level: By default, when crossprobing, Adobe Reader zooms in to the

objects selected in Allegro. If you select Freeze Zoom Level from the Cross Probe menu, the zoom is set to the magnification when the command was run. This is similar to Allegro's no_zoom_to_object user setting.

Selecting Objects

After communication has been established between Allegro and Adobe, you can select symbols, nets, or pins in either application and they will be selected in both applications.

To select objects. The objects are highlighted in the native application. In the other application, the selected objects are highlighted, and, by default, centered in the window, and zoomed. The same type of object, either component, net, or pin, must be specified in both applications.

Selecting Objects in Allegro

To select objects in Allegro, you must be in a command's selection mode, such as Show Element.

The same type of object, either components, nets, or pins, must be specified in both applications. In Allegro, the Find Filter specifies the object type.

Note: Due to inconsistencies in Allegro's general edit mode in 16.0, we recommend using its Info mode as well.

If you are selecting objects in Adobe Reader and Allegro is not in a command's selection mode, that is, it's in Idle mode, you can access the cross probe commands from the 3rd button menu. To access the 3rd button menu in 16.0, you must set Allegro's application mode to None, using the Setup->Application Mode command.

Selecting Objects in Adobe

In Adobe, the CircuitSpace > Selection Type menu item specifies the object type. The same type of object, either component, net, or pin, must be specified in Allegro. You can use the icons to select the object type, as show in Figure 18.

Figure 18: Adobe Cross Probing Selection Icons

To select objects in Adobe, you can

- draw a selection window around the objects
- click on an object
- press the Shift key while clicking to add additional objects to the selection set
- press the Control key while clicking to add or remove additional objects to the selection set
- select CrossProbe Add Pages… and CrossProbe Rem Pages… from the Options menu

on the Navigation pane or from the 3rd button menu, activated by pressing the Shift key and the right mouse button. This enables you to add or remove pages from the selection set, either selected pages, or a hierarchical selection.

Working with Selected Objects

CircuitSpace provides two commands for working with selected objects: **Define Cluster Keep Positions** and **Define Cluster Reposition.** Both of these commands require parts that are selected via cross-probing. **Define Cluster Keep Positions** draws a cluster around the parts without moving them. **Define Cluster Reposition** tiles the selected parts on your cursor, and redraws them where you select.

Ending Cross Probing

When finished cross probing, in Allegro, select **Stop Cross Probing** to exit CircuitSpace's cross probing mode. Then in Adobe, turn off cross probing.

Using CircuitSpace Commands

CircuitSpace plugs into Allegro's PCB Editor. The CircuitSpace menu is available to the left of the Allegro Help menu.

Command Structure

CircuitSpace uses the same "verb/noun" command structure as Allegro (the command-then-object method of command execution). CircuitSpace also implements the right mouse button pop-up menu, which contains modes for the current command. After supplying the required information for a command, select **Done** from the pop-up menu.

Command Modes

Many CircuitSpace commands have modes, like **Select Scope** or **Select Source Cluster**. A mode requires a specific input. While in a mode, you can select a cluster or part for that input. When you select a different mode from the pop-up menu, you have finished the current mode and are in a different mode. You can select a mode again and its previous selection will be highlighted and can be changed. If you select **Done** before specifying all the required information, you will be prompted for the missing information.

Select Scope Mode

CircuitSpace commands that create or modify clusters, like **Replicate** and **Autocluster,** enable you to select a scope for the command. The scope is the set of clusters and parts that provides parts for the command.

Selecting a cluster is a short-cut for selecting that cluster's parts; selecting a cluster for the scope selects all of its parts, including any parts that belong to child clusters. However, if a cluster has the *Do Not Remove* property, its parts are not added to the scope. If CircuitSpace processing removes all the parts from a cluster in the scope, either the parent, or a child, that cluster is deleted.

For commands that do not take targets, like **Replicate** and **Apply Templates**, the default scope is all off-board parts. For commands that take targets, like **Modify Membership**, the default scope is the target clusters. You can use the *Part Selection* and *Cluster Selection* options as short-cuts to select objects in Allegro, and then you can select or

deselect clusters and unclustered parts from the Allegro Drawing window.

Select Source Cluster Mode

Most commands have a **Select Source Cluster** mode, which requires you to select a source cluster.

Select Target Clusters Mode

Most commands have a **Select Target Clusters** mode, which requires you to select one or more target clusters. You can use the *Cluster Selection* options to specify a tag, name, or list of clusters. By default, parts from the targets are in scope.

Options

Options for CircuitSpace commands can be specified on the Allegro *Options* tab.

Part Selection

Part selection enables you to select broad categories of parts easily. Categories include unclustered parts, clustered parts, on-board parts, and off-board parts. You can further select or deselect directly in the Allegro window. The *Part Selection* option is available only in **Select Scope** mode.

Cluster Selection

Cluster selection enables you select groups of clusters by tag, name, or list.

Match Type

You can specify the type of membership matching on the Connectivity for Exact Match section of the *Option* tab. The default is strict connectivity for exact matches, and no partial matches.

CircuitSpace returns strictly matched clusters (see *Concepts: Matching Clusters* on page 19 for details). You can choose to use equivalent connectivity for exact matches. Equivalent connectivity allows the following conditions:

- Pins in a target may be swapped.
- A target may have extra pins on corresponding nets.

You can also choose to return partial matches. A partial match is required to have only the same master part as the source. For partial matches, you can choose to use the same type of connectivity as exact match or use the weaker rule, incomplete connectivity, where only 80% of the pins between two parts are connected.

CircuitSpace always returns exact matches. You can specify whether exact matches should adhere to the strictest connectivity rules, or use equivalent connectivity. You may also specify that a command should return partial matches, and the connectivity to use for the partial matches, either the same connectivity as exact matches, or to allow incomplete nets.

Batch Commands

CircuitSpace commands that require engine processing are *batched*. Control is returned to the Allegro Edit window, and the command continues processing in the background. When the command finishes, the user must apply the solution to the Allegro Edit window. You should apply the results of a command before issuing another command that will use the results of the first command. See the **Batch Control** command reference page for details.

Working with Clusters

Modify a Cluster Shape

To modify a cluster's shape, use the **Allegro Edit Shape > Boundary** command.

Modifying Cluster Properties

You can modify a cluster's properties: change its name, its master part, its protections, or its tags. See the **Modify Properties** reference page for usage details.

Deleting Clusters

Use the **Delete Clusters** command to delete one or more clusters. See the **Delete Clusters** reference page for usage details.

Moving Clusters

You can move or rotate a cluster. If the cluster has etch associated with it, any etch within the cluster boundary is moved; etch outside the cluster boundary is not moved, although it still belongs to the cluster. See the **Move Clusters** reference page for usage details.

Aligning Clusters

You can align selected clusters with a source cluster, either horizontally or vertically. See the **Align Clusters** reference page for usage details.

Resize Cluster Boundaries

You can cause CircuitSpace to recalculate the selected cluster's boundary. See the **Autosize Cluster** reference page for usage details.

Concepts: Advanced Topics

This chapter provides detailed discussion on *cluster protections*, *tags*, and *etch*.

Cluster Protections

Protection flags enable designers to identify clusters that CircuitSpace should not modify. There are two types of protections: *membership protections*, which control the addition and deletion of parts, and *placement protection*, which controls how the parts are placed relative to each other. These protections are specified by setting *protection flags*.

You may change the protection flags of a cluster using the **Modify Properties** command.

Membership Protections

CircuitSpace provides two membership protection flags: **Do not add parts** and **Do not remove parts**. If the **Do not add parts** flag is set, CircuitSpace cannot add parts to the cluster. If the **Do not remove parts** flag is set, CircuitSpace cannot remove parts from the cluster. You can finely tune how clusters are managed using these flags.

When a cluster is *unprotected*, that is, neither membership protection flag is set, CircuitSpace can alter the cluster membership by adding and removing parts.

When a cluster is *protected,* with both membership protection flags set, CircuitSpace cannot make changes to the cluster membership.

When the **Add Parts** and the **Do not remove parts** properties are set, parts can be added to a cluster, but current members cannot be removed. This protects clusters from losing parts during CircuitSpace operations.

When the **Do not add parts** flag and **Remove parts** properties are set, parts are only removed from the cluster.

Placement Protection

CircuitSpace provides the **Relative Placement** flag. When set, the cluster's parts cannot move relative to each other. New parts may be placed within the cluster, as long as the existing parts are not moved.

Select **Protect relative placement** in the Allegro *Options* panel to set the **Relative Placement** flag.

Boundary Protection

When set, the **Shape-Position** flag prevents CircuitSpace from modifying the cluster's boundary or changing its position on the board. If you modify a cluster's boundary, even by rotating the cluster, you must set the **Protect cluster shape-position** flag, or the cluster will be redrawn parallel to the x,y axis by CircuitSpace.

Use the **CS Modify Properties** command to set or view the **Shape-Position** setting. Select **Protect cluster shape-position** on the Allegro *Options* panel to prevent CircuitSpace from modifying the cluster's boundary.

How Protections Are Visualized

A cluster's cross-hatch pattern signifies its membership protections; its color signifies its placement protection. CircuitSpace uses the following patterns to display a cluster's membership protections:

- Can remove, Can add (unprotected): **horizontal and vertical** cross-hatch.
- Can remove, Can't add: **vertical** cross-hatch.
- Can't remove, Can add: **horizontal** cross-hatch.
- Can't remove, Can't add (protected): **no** cross-hatch.

Cluster placement protection is represented by a cluster's color. CircuitSpace assigns clusters with the **Relative Placement** flag set to a unique Allegro color slot. A cluster whose **Relative Placement** flag is not set is assigned to another color slot. You can use the Allegro **Color and Visibility** command to view or modify the colors assigned to CircuitSpace on the Board Geometry layer.

How Protections Are Used

CircuitSpace uses protections to determine processing for two types of clusters: target clusters that are being modified, and clusters in the scope that donate parts.

When modifying target clusters, you specify whether to obey or ignore the target cluster protections. When honoring target cluster protections, if **Do not add parts** is set, new parts are placed in an addendum cluster. If **Do not remove parts** is set, the matching report details the parts that could not be removed because of the target's protection.

When components are needed to create clusters or add to clusters, CircuitSpace only uses components from clusters in the scope that have the **Remove Parts** property set.

User Tags

CS provides two types of tags that users can assign to groups of clusters: membership tags and placement tags. These tags act as "sticky notes" – the designer can attach them to clusters, and remove them.

User tags are an easy mechanism for selecting a user-defined group of clusters. A designer can specify a user tag to identify the target clusters for a command, or to specify clusters for the scope for a command. For example, a designer may assign a group of clusters the same membership tag, and then select this group via its tag.

CircuitSpace does not enforce how clusters are related when assigning tags; for convenience we call them membership and placement tags. However, clusters with the same tags are usually related. The user can assign a membership tag or placement tag to one or more clusters with the **Modify Properties** command. Existing tags are replaced with the new tag.

When creating new clusters, CircuitSpace assigns the same tag to matching clusters. CircuitSpace assigns clusters created from the same template the template's membership and placement tags. Likewise, CircuitSpace assigns clusters derived from the same functional block during autoclustering the same membership tag. And when replicating a cluster, that is, creating copies of a source cluster, the new clusters are assigned the same membership tags and placement tags as the source cluster. If the source cluster does not have tags, CircuitSpace generates them.

When modifying clusters, CircuitSpace assigns the source cluster's tags to the target clusters. When modifying a cluster's membership using the **Modify Membership** command, all targets are assigned the source cluster's membership tag, replacing their original membership tag. Clusters with the source cluster's original membership tag that are not selected as targets are given a different membership tag. Placement tags are not modified.

If the target cluster has the same placement tag as the source, CS attempts to propagate placement as well as the membership change. However, the targets' placement tags are not modified.

When propagating a cluster's placement with the **Propagate Placement** command, the source's placement tag is assigned to the target clusters. Clusters that have the same placement tag as the source cluster, but are not selected as targets, are assigned a different placement tag. The membership tags are not modified.

Tag Versioning

As you modify clusters, you must decide if you want to revise the cluster's tag, or create a new *version* of the tag. Revising a tag overwrites the current version. If you choose not to overwrite the tag, a new version of the tag is created.

For example, if you improve a cluster's placement you'll want to revise the tag – you have no use for the old placement. If you revise a cluster, it keeps its original tags. Any clusters with the same tag that the change was not propagated to receive a new version of the tag; that is, the revised cluster retains the original tags, and the unchanged original clusters receive new versions of the tags.

However, if you have similar placements, you'll probably want to track different versions of the placements. For example, if you have five clusters with the same membership, and three are to have one placement, while the other two will have a different placement, you can version the placement tag to track these differences. All five clusters will have the same membership tag. Three of the clusters will have one placement tag, for example, *Placement*, and the other two will have the versioned tag, *Placement_v1*.

Cluster Modifications

If a cluster loses a part, it still keeps any tags that it has, even though the cluster no longer corresponds to other clusters with the same tag. For example, if the cluster is in scope and CircuitSpace removes a part to use elsewhere, the cluster's tags are not modified.

Etch

You can associate etch with a cluster if it is within the cluster boundary. A cline is within a cluster's boundary if both ends are within the boundary. If only one end is within the boundary, the cline is clipped at the boundary. If a cline starts and ends outside the boundary, for example, it passes through the cluster, it is not within the cluster boundary.

The etch associated with a cluster will be moved and saved with the other cluster members. Any etch outside of the cluster boundary is not saved and loses its cluster assignment when the cluster is moved. The etch outside the boundary is not moved and no longer belongs to the cluster.

Cluster etch is assigned to a cluster by either saving a template that has etch and then applying the template with **Apply Template Etch** command, or adding etch directly to the cluster with the **Modify Membership -> Edit Etch** command.

Adding Etch to a Cluster Manually

To add selected etch to a cluster manually, use the **Modify Membership -> Edit Etch** command. Select the cluster, then select the etch. Any etch that extends outside the boundary will not be saved with that cluster's template, and will not be moved with the cluster. You can change the cluster shape to enclose the desired etch and the etch will maintain its cluster assignment.

Saving and Applying Etch Via Templates

When saving a template, CircuitSpace saves the etch that is assigned to a cluster and within the cluster's boundary, even if the etch is not connected to cluster components. If there is no etch defined for a cluster, CircuitSpace will automatically derive the cluster's etch, only saving etch connected to pins of cluster members and within the cluster boundary. Etch is clipped at the cluster's boundary. An area fill must be entirely within a cluster's boundary in order to be saved. You can adjust the cluster boundary so the fill is within the boundary before saving.

The **Save Template** command has an option to force recalculation of etch. If you choose to recalculate the etch, any previously-defined etch is ignored, and etch connected to pins within the cluster's boundary is saved.

After saving a template, you can apply a template's etch to clusters on the board.

When applying etch, you choose to apply one of three levels: fanout, in-cluster, or all.

- **fanout** consists of the etch from a component pin to the first via inside the cluster, including the via.
- **in-cluster** consists of the trace and vias connecting any two cluster pins,

inclusive of fanout. Also, all etch shapes are included.

- **all** is all the trace, vias, and fills that were saved in the template.

When applying etch, if the template's cluster shape is different than the cluster on the board, the cluster on the board is modified to match the template's shape as part of applying etch to the cluster.

CircuitSpace accesses a layer's etch by the layer's name. By default, the template layer is mapped to the board layer of the same name. You can change this mapping. Ideally, the board and template will have the same layers, both the same names and the same functionality. If this is not the case, please note the following scenarios:

- If the board does not have a layer name that is saved in the template, you can map the template layer to a different board layer, or you can ignore the template layer and not apply it.
- If the layers have been reordered on the board, you must map them to the board layers preserving the original order. Else, the template layers are applied in the new order.
- If the board layer and the template layer have the same name, but the function of the layer is different, you will get unacceptable results.

Modifying Etch

You can add or remove etch from a cluster with the **Modify Membership > Edit Etch** command. Just as adding or removing members from a cluster changes the cluster membership, adding and removing etch to a cluster does not create or delete the etch. Instead, it assigns existing etch to a cluster, or removes etch from a cluster; that is, the etch no longer belongs to the cluster. Any etch that is added to a cluster and is within the cluster boundary is saved with that cluster's template, even if the etch is not connected to cluster components.