

ELECTRA

User Guide

Version 7.xx



Copyright Notice

Copyright © 2003-2018 by KONEKT, SPRL. All rights reserved.

No part of this work may be reproduced without receiving the prior permission of the copyright owner.

All brand or product names mentioned in this guide are trademarks or registered trademarks of their respective owners.

In particular:

SPECCTRA® is a registered trademark of Cadence, Inc.

Windows is a trademark of Microsoft Corp.

Disclaimer

KONEKT SPRL AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

1	Overview of ELECTRA Routing Technology	5
2	Main Features	8
3	CAD Flow Integration	9
4	AutoRouting with the GUI	10
5	AutoRouting with commands or DO file.....	11
6	Basic Routing Strategy	11
6.1	Strategy Dialog Editor.....	13
6.2	Reporting of routing via styles	15
6.3	Analyzing Routing Results	16
6.4	Improving routability	17
6.5	Restarting the autorouter.....	18
6.6	Mouse driven Viewing Navigation	19
6.7	Constraints Editor	20
7	Advanced Rules Hierarchy.....	21
	Clearance Types	23
7.1	Class Editor	26
7.2	Differential Pair Autorouting	27
7.3	Length Constrained Autorouting.....	29
7.4	Autorouting by Fence.....	30
7.5	DRC Violation Browser	30
7.6	Preview DO file	30
7.7	Area Rules support.....	31
8	Guided Autorouting.....	34
	Tutorial on Guided Autorouting	35
	Tutorial on High Speed Design with ELECTRA.....	39
9	Quick Command Reference	50
9.1	Notations	50
9.2	Descriptors.....	50
9.3	Types:	52
10	Command Reference	53
	add.....	53
	autopair	53
	autoroute	53
	bestsave	53

bus	54
check.....	54
circuit	54
clean	56
cost	56
direction	58
fanout	58
filter.....	59
gloss	60
grid.....	60
limit	60
lock	61
lroute	61
miter	62
recorner	63
route	64
rule.....	64
select	65
status_file.....	65
tax.....	65
unit.....	66
unlock	67
unprotect	67
unselect	67
write	67
11 Knowledge Base of Tech Tips	68
11.1 Constraining Via Type Usage.....	68
11.2 Cost and Tax usage	71
11.3 Route command <start_pass> parameter	73
11.4 Microvias.....	74
11.5 Power and Ground planes.....	75
11.6 Mixed Power Planes	76
11.7 Autorouting Decoupling Capacitors	77
11.8 Recorner expert	78
11.9 Filter 5	79
12 Appendix.....	80
Autorouting Techniques/Using ELECTRA with Protel	80

1 Overview of ELECTRA Routing Technology

The large majority of commercial autorouters for PC boards use a grid-based search model and place the interconnects on a fixed grid. They also route each net sequentially without allowing DRC conflicts. These routers very quickly reach a deadlock, creating long detours and a large number of vias.

With the advent of component dimensions in mixed units (1990's) and complex constraints, a grid-based model for autorouting is not satisfactory and has discouraged many PCB designers who had no choice other than routing manually.

ELECTRA™ is a new generation of Shape-Based Autorouting software for PC boards. In contrast with traditional gridded maze autorouters, a shape-based approach allows for more efficient use of routing area and is more suited to handle complex design rules requirements of high density PCB designs.

In addition to its shape-based architecture, ELECTRA routes with conflicts and uses a multi-pass cost-based conflict reduction algorithm to find a routing solution adapting to the natural flow of the nets.

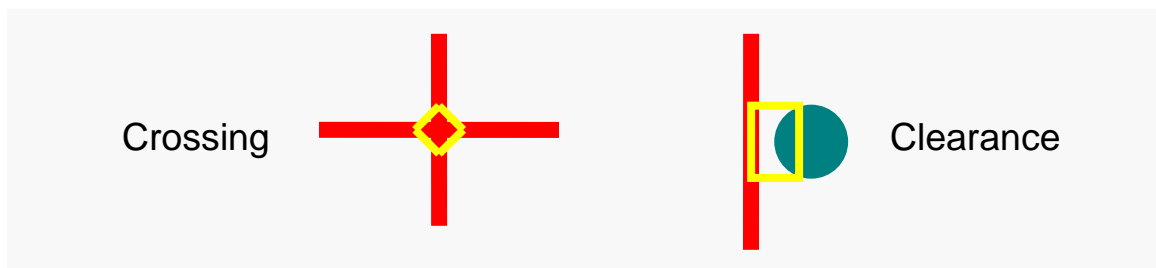


Figure 1

On the initial passes the router assigns a relatively low cost for accepting a routing path solution with crossing and clearance conflicts (figure 1). Internal costing is then slowly increased on subsequent passes, as illustrated on figure 2. This all happens automatically!

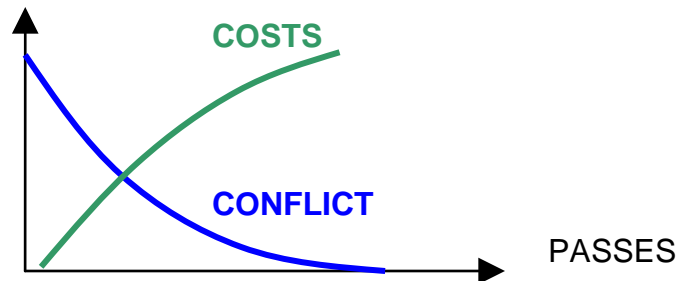
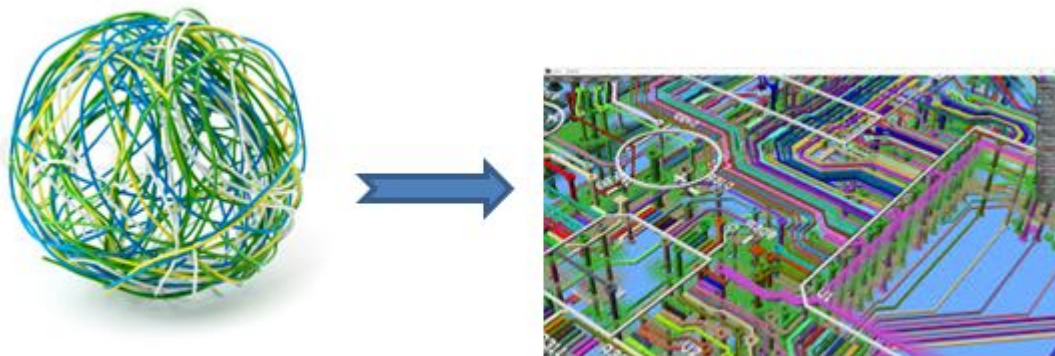


Figure 2

By allowing nets to be routed with conflicts, autorouted wires will initially start with an ideal path.

By analogy with nature, a simulated annealing algorithm is applied to resolve the conflicts while adapting to the presence of other nets. You can intuitively picture the idea by thinking of a ball of wires that you try to flatten into a lasagna of layers by untangling and reducing local crossings while still retaining the whole interconnect topology.



The final routing solution closely mimics interactive routing results in terms of optimal wire length and number of vias.

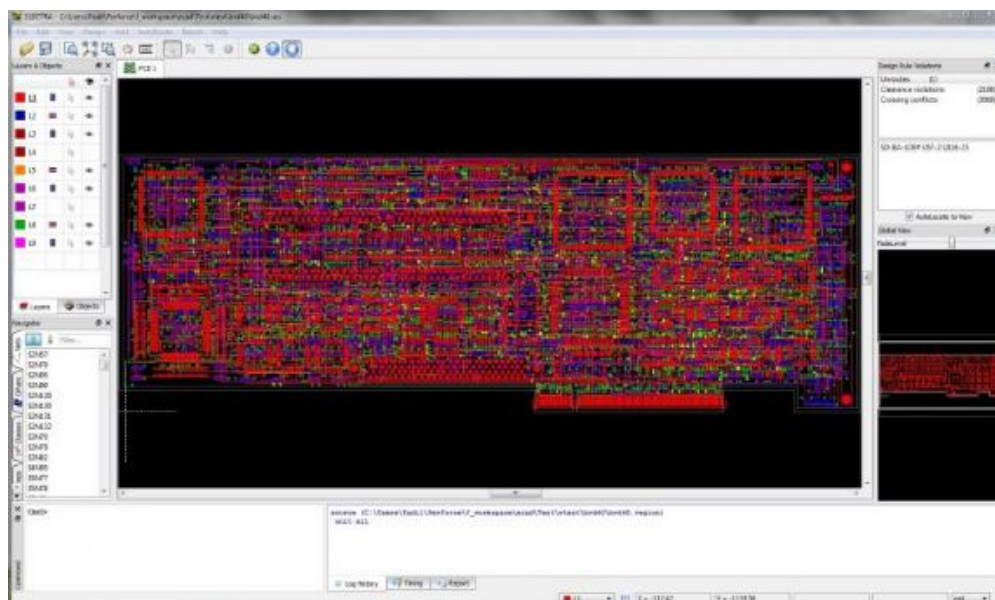
As the router iterates through the passes, it uses conflict information to generate a solution that resolves the conflicts and untangles the routed wires.

When the design is easily routable, the conflict curve drops down very rapidly.

If the design is unroutable, the conflict reduction rate will exhibit oscillations.

Routing history can be monitored to predict and qualify the routability of a design. (Read 6.1 for more details on Analyzing Routing Results)

Adaptive routing algorithms are a proven approach to systematically reach high completion rate on complex high density PCB designs.



8 layers, 2 power/ground planes, 3380 pins, 2276 wires, routed to 100% in 12 minutes

2 Main Features

- Gridless routing of up to 256 layers
- Shape-based architecture
- Differential pairs autorouting
- Length matching autorouting
- Autorouting to target length
- Wiring and Clearance rule by layer, net intra-classes and inter-classes
- Via size and use_layer rule by net class
- Width and spacing rule by area
- Split/full power and ground plane support
- SMD escape fanout control
- Routes SMDs on both sides of the board
- Blind via and buried via support
- Support for embedded components
- Split Power/Ground Planes support
- AutoRouting by polygonal keep-in fences
- Memory routing pass
- Supports pre-defined fanout patterns
- Customization of cost factors
- Post-route cleanup optimization
- Real-time display of routing progress
- Shadow mode display on selection
- DRC Violation browser
- Preview DO file
- Batch routing option
- TCL Scriptable routing strategy (DO file)

3 CAD Flow Integration

ELECTRA supports a broad spectrum of PCB CAD systems by reading the widely-employed Design File (DSN) format. ELECTRA's routing results are saved into standard route file format (RTE) or session files (SES). ELECTRA is thus a plug-and-play autorouter for any existing PCB CAD system environment that has a SPECCTRA® design file interface.

The design flow for employing ELECTRA is illustrated by the diagram below:

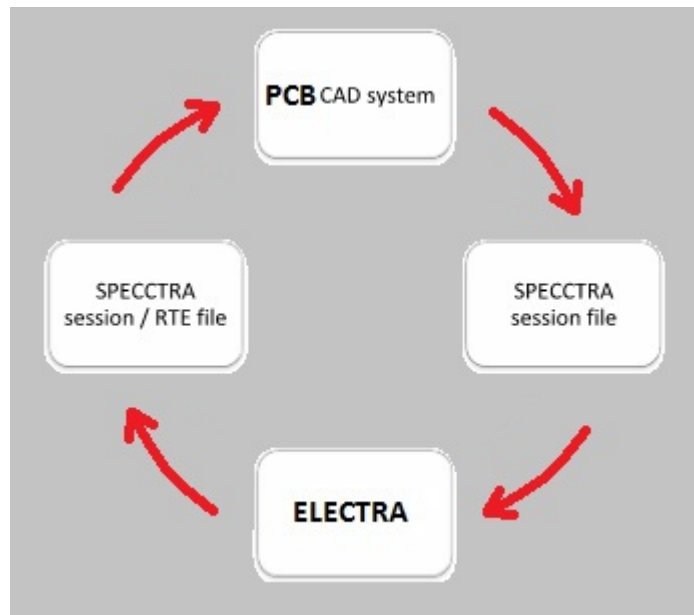
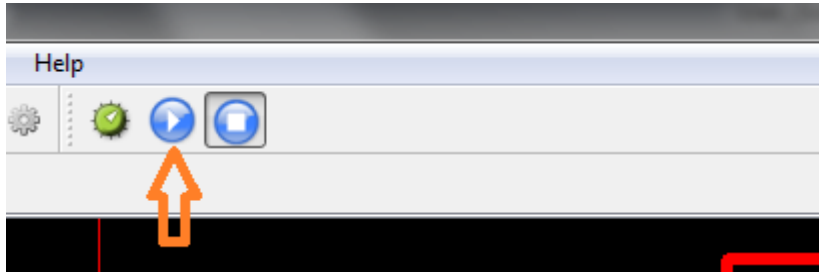


Figure 3

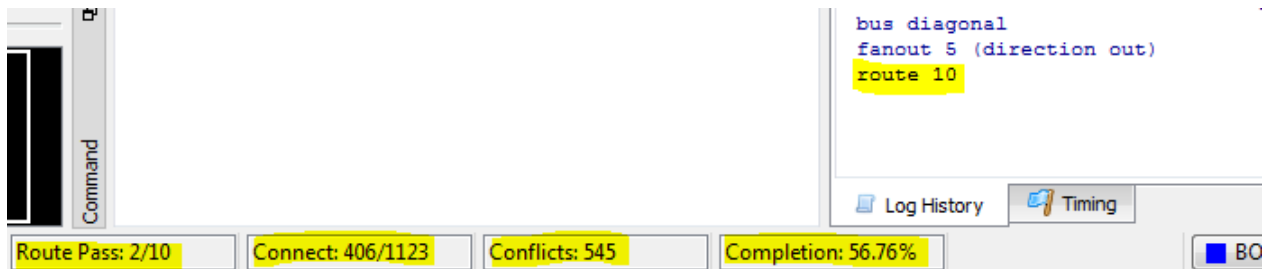
4 AutoRouting with the GUI



The router can be controlled by either using the GUI or by a text-based DO file.

1. Start ELECTRA, select the menu **File / Load** and load the demo design located in <install_dir>\Demo\Demo.dsn
2. Select the icon as shown above to run the pre-defined routing strategy. You can now watch the router in action.

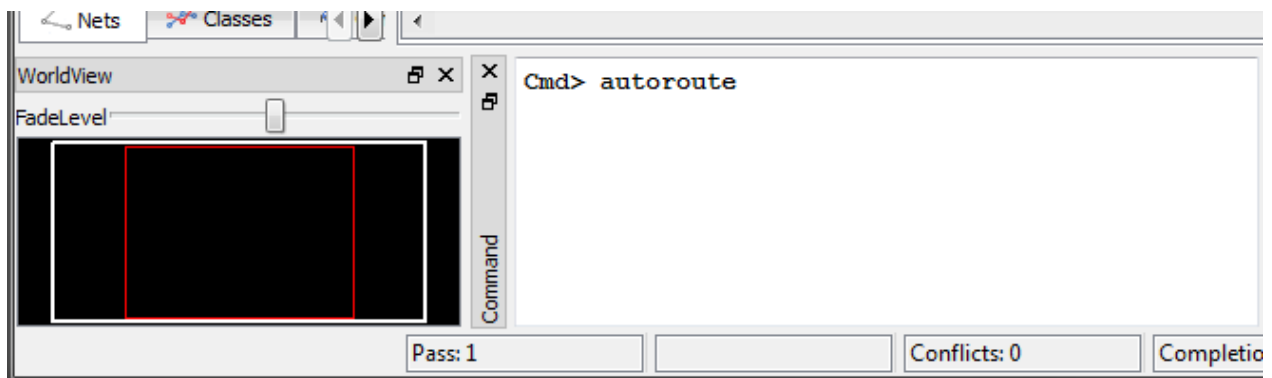
ELECTRA provides immediate feedback on the routing progress and conflict reduction rate, key parameters can be monitored on the status bar and the log panel. See text in yellow:



5 AutoRouting with commands or DO file

Electra can also be controlled through typed-in commands or through the direct execution of a command file (DO file). Text based DO file method allows for unattended batch operation as well.

The equivalent command to run the pre-defined routing strategy is named **autoroute**. Click in the command entry panel and type in *autoroute*.



6 Basic Routing Strategy

The adaptive routing strategy method generally requires many routing passes and involves selection of the right commands to build the best strategy.

Sequence of commands can be saved into a text DO file for future reuse. Use the "File/Execute Do file..." menu to execute a Do file.

A pre-defined routing strategy DO file can be found in the executable directory, it is called "basic.do". The strategy is as follows:

```
#=====
# BASIC DO file
#=====
# Phase 1 - Initial
#-----
bus diagonal
fanout 5
```

```
# Phase 2
#-----
route 20
clean 2
```

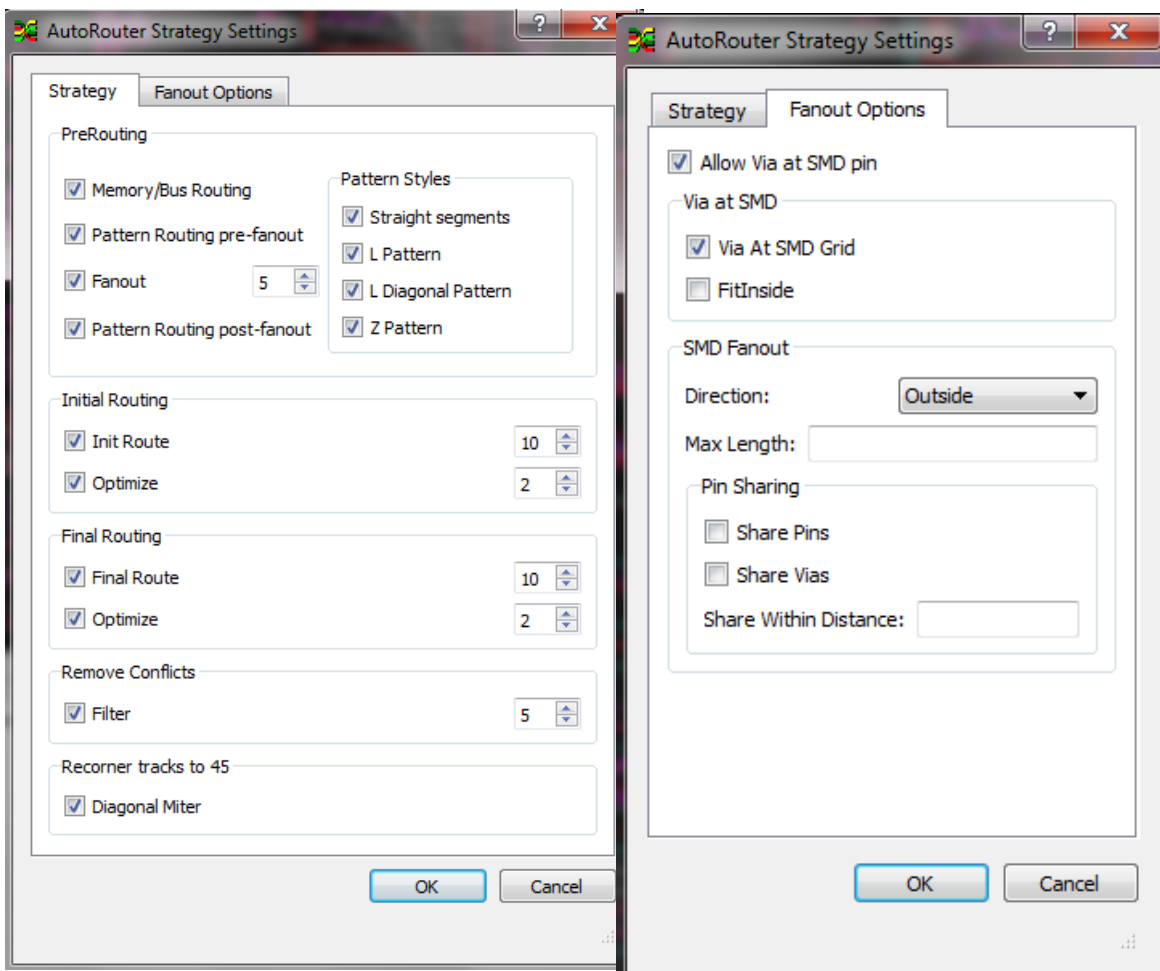
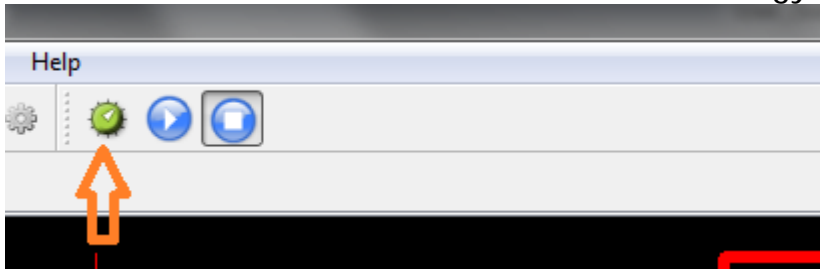
```
# Phase 3
#-----
route 25 16
clean 2
```

```
# Phase 4
#-----
filter 5
```

```
# Phase 5 - Final
#-----
recorner diagonal
status_file
```

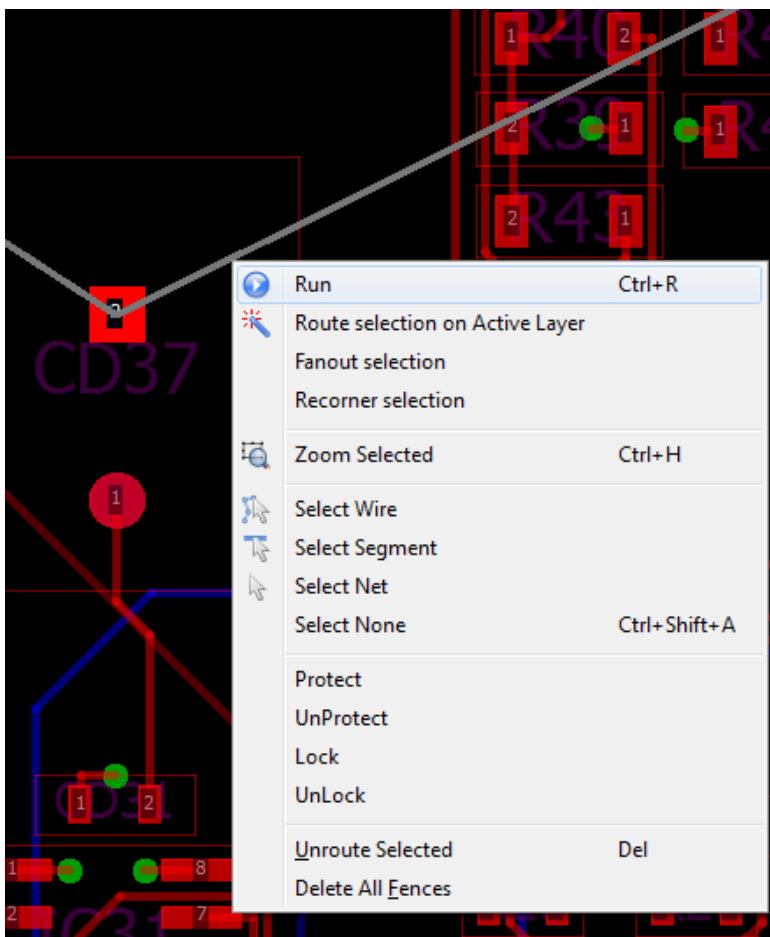
6.1 Strategy Dialog Editor

Select the toolbar button to invoke the strategy editor.



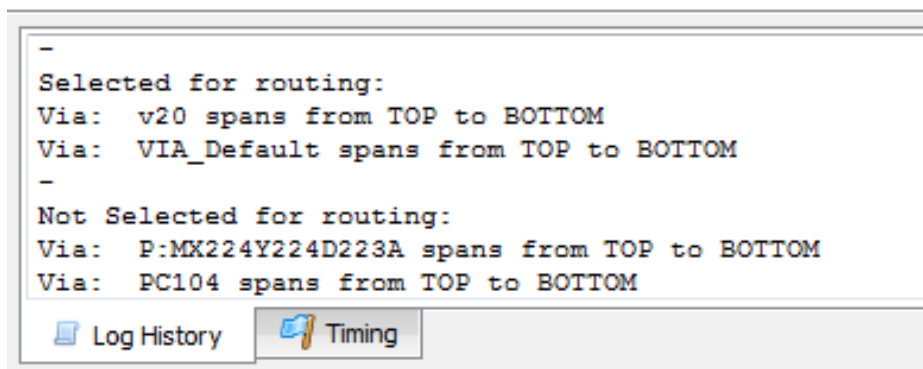
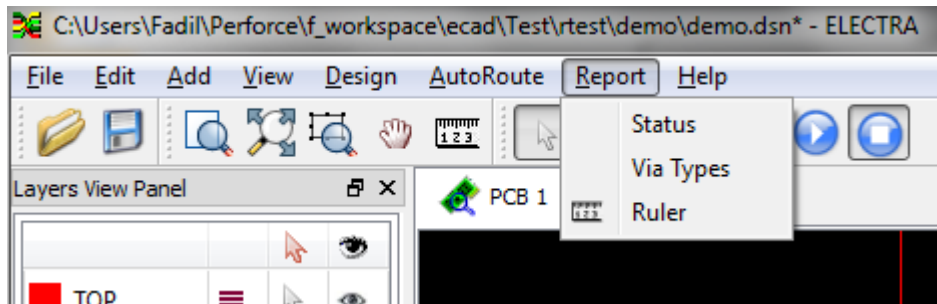
A basic routing strategy can now be defined and reused without having to "text edit" a DO file. The strategy settings are automatically saved and restored the next time you reload the design. This is accomplished, behind the scene, by saving the strategy settings in the design directory. The strategy file is named after the design name with the .rtr extension appended. For more advanced requirements, the use of a DO file is of course still available.

You can autoroute selected nets only, and if no nets are preselected then the autorouter works on all nets. You can also invoke the autorouter by bring the popup menu at the cursor location (RightMouseButton) and select "**Run**". The key shortcut is "Ctrl + R" to run the autorouter on selected nets.



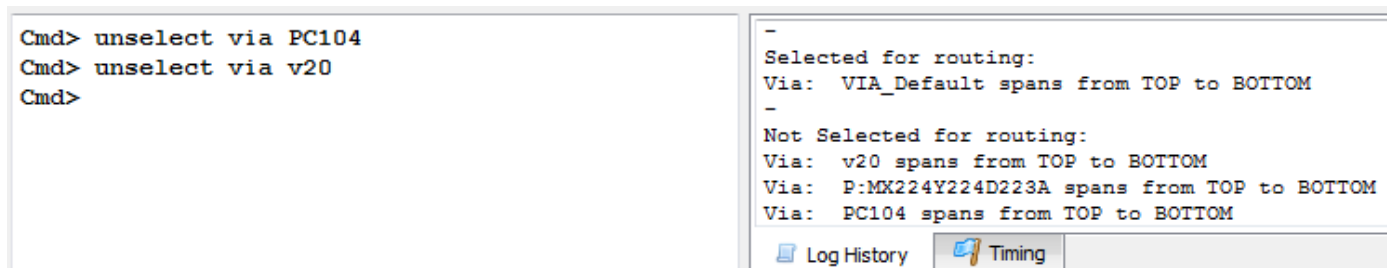
6.2 Reporting of routing via styles

Reports the vias that are selected for routing. You can verify that the via defined in the DSN file are as expected.

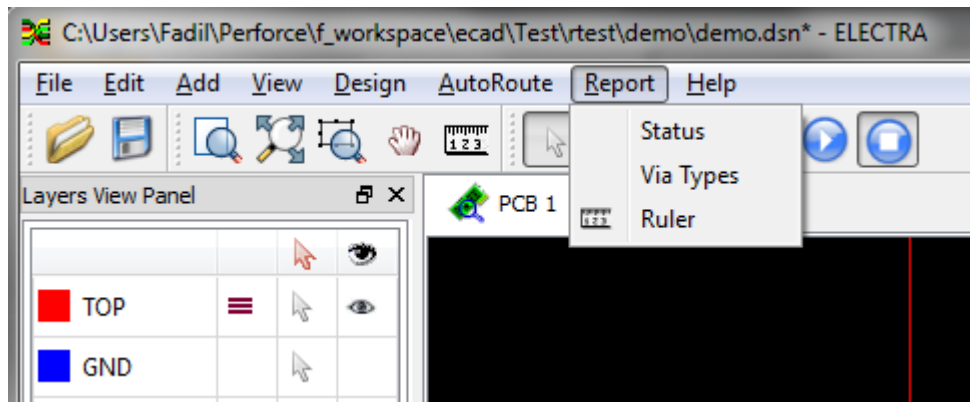


You can also select/unselect a via style as needed. For example here is the effect of typing the following command:

unselect via v20



6.3 Analyzing Routing Results



Select the top bar menu as shown to get a routing report in the log window. Alternatively you can use the command **report status** to generate a report.

To determine if a design is routable, check the status report for convergence signs, specifically that conflict reduction rate is higher than 15% after passes 2-5

Here is an example with high conflict **reduction rate** highlighted in yellow:

```
ELECTRA Version  3_beta.02
Design           = C:\Users\Fadil\Test\rtest\demo\demo.dsn
Report Time      = 17:35:41
Nets             = 392
Components       = 223
Connections      = 1036
Vias             = 904
Routed Length    = 1145722.7 mil
Signal Layers    = 4
Power Layers     = 2
Total Pins       = 1570
Completion       = 100.00%
```


Unroutes = 0
Routing Time = 00:06:17

#	Algo	Shorts	DRC Errs	Failures	Unroutes	Vias	Reduc	Time
Total								
0	Bus	0	0	32	1028	5		0:00:01
0:00:01								
1	Fanout	0	0	0	796	1093		0:00:15
0:00:16								
2	Fanout	0	0	0	796	1093		0:00:01
0:00:17								
3	Route	472	73	0	0	947	0	0:00:52
0:01:09								
4	Route	179	14	0	0	998	64	0:00:52
0:02:01								
5	Route	41	2	0	0	1037	77	0:00:55
0:02:56								
6	Route	9	0	0	0	1067	79	0:00:14
0:03:10								
7	Route	1	0	0	0	1080	88	0:00:06
0:03:16								
8	Route	0	0	0	0	1081	100	0:00:02
0:03:18								
9	Clean	0	0	23	0	956		0:00:43
0:04:01								
10	Clean	0	2	15	0	930		0:00:41
0:04:42								
11	Route	0	0	0	0	929	0	0:00:01
0:04:43								
12	Clean	0	0	12	0	912		0:00:39
0:05:22								
13	Clean	0	0	25	0	904		0:00:41
0:06:03								
14	Filter	0	0	0	0	904	0	0:00:01
0:06:04								
14	Recorner	0	0	0	0	0		0:00:05
0:06:09								

6.4 Improving routability

1. An unroute is a connection where the autorouter has not been able to find a path. This can be caused by a related pin that cannot be accessed, the presence of some blockage, or a high congestion area.

2. High crossing counts, indicative of insufficient signal layers and/or via starvation
3. High clearance counts may come from design rule errors or improper layer route direction
4. Going gridless will help in achieving higher completion rate.
5. Verify layer directions
6. On 2 layer boards you might have to run many passes (~100). Note that passes above pass 5 run faster since only nets that are in conflict are rerouted.

6.5 Restarting the autorouter

When the number of completed passes (n) is less than 15, the <start_pass> should be n+1.

When the number is higher than 15, then the <start_pass> should be 16.

The syntax of the route command allows control of the starting pass count: **route <passes> <start_pass>**

6.6 Mouse driven Viewing Navigation



The mouse navigation methods are designed for a three button mouse, and involve either a click, or a drag operation.

The **mouse wheel** can be rolled to zoom in and out.

You can “drag-pan” the design by using the right mouse button and holding it down while moving the mouse.

The Ruler tool is used to measure a distance in the design window. Click the left mouse button on the screen to designate the first point of measurement, drag the mouse to the last point to perform the distance calculation. The total distance and the intermediate distances will be displayed next to the cursor.

- **Zoom To Area using the bird's eye view:**

Drag the left mouse button in an upward motion from the lower-left or lower-right starting corner until the desired view is within the elastic bounding box...





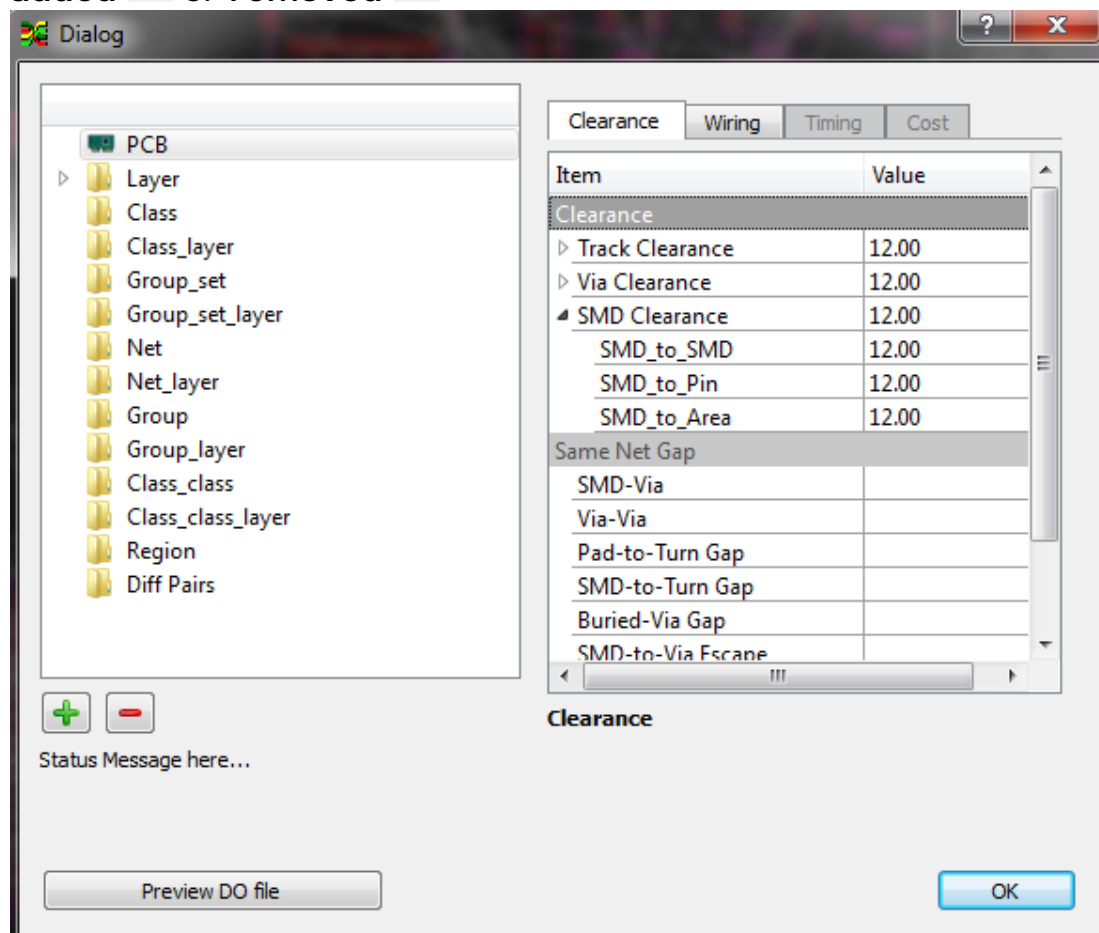
6.7 Constraints Editor

In comparison to the writing of a DO file with a text editor, the Electra constraint editor is a graphical widget method for defining the constraints.

The user interface allows for quick creation of design rules in a tree-table view environment.

The tree view shows the rules in their order of precedence of the hierarchy.

Rule values are presented in a table format. High speed constraints are defined around the net connections (pin to pin from-to) and signal paths (group and group_set of connections). Each design object type can have associated rules which can be added  or removed .



7 Advanced Rules Hierarchy

ELECTRA is driven by DFM and high speed layout rules. Each interconnect object can have its own minimum clearance and wiring constraints. The autorouter combines the rules of all design objects based on their precedence in the hierarchy. Net classes and group of connections can be constrained to be routed on specific layers (impedance control) and use different rules for each of the layers. Different via type can be assigned to each interconnect; these could be used for example for power and ground current carrying requirements. The autorouter finds a solution that simultaneously respects all the user defined rules constraints.

Starting with default values for the PCB, critical nets can be grouped into a class and assigned rules that override the defaults. Within that class, a net may be assigned a value overriding the PCB default.

Minimum clearances can be set by layer and for classes and nets. These clearances are measured from the edge of the wire to specific objects (via, wire, pin, smd, via or board outline/keepout area). For fine-pitch smd parts, wires can be routed on internal layers with larger clearances only on the routing layers, allowing a small length to a fanout via on outer layers unaffected by the clearance rule.

Rule-by-net class applies to the elements of the class. Class-to-Class rules apply between selected classes. This allows isolation between classes that should not couple and is a preferred alternative to forcing these classes onto different layers.

Class_class C1 C2 C3 = apply between all combinations but not itself. Combinations are C1-C2, C1-C3 and C2-C3

Class-to-Class C1 C1

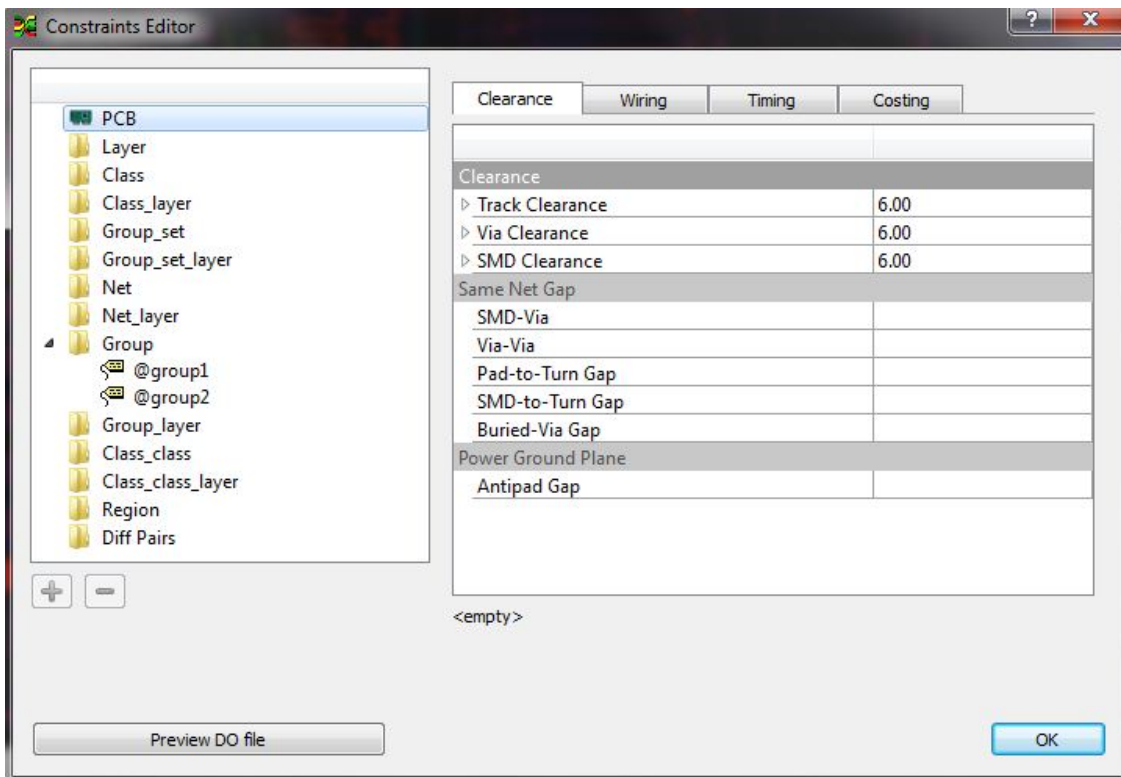
Apply only between elements of C1. For busses, a class-to-itself

clearance can be defined, allowing a data or address bus that switches simultaneously to route closer to itself than to other nets.

ELECTRA reads the rules constraints out from the DSN file. The host CAD system interface embeds the design rules into the transferred DSN file.

Rules can be set globally (PCB) or specifically to layers, net classes or nets. When rules are applied at these different levels, a rule's hierarchy precedence takes place.

The constraint editor GUI has organized the rule hierarchy as follows:



The autorouter rule precedence is as follows

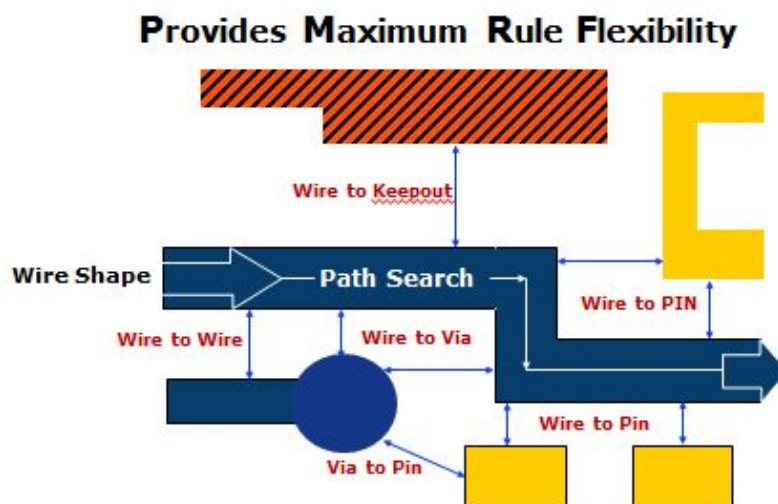
PCB < Layer < Class < Class_layer < Group_set <
Group_set_layer < Net < Net_layer < Group < Group_layer
< Class_class < Class_class_layer < Region

PCB rules have the lowest priority and region rules the highest.

Clearance Types

pin_pin
pin_smd
smd_smd
pin_wire
smd_wire
wire_wire
smd_via
pin_via
via_wire
via_via

For example, the **pin_via** clearance is defined as the minimum clearance from the pin to any via. The **via_via** clearance is defined as the minimum clearance between any two vias on the same net and the same layer.



Here is the list of rules that are embedded in the design file (DSN) and that are supported by ELECTRA.

There are two categories of rules type: clearance and wiring rules as shown by the rule descriptor below.

```
<rule_descriptor> ::=
  [(clearance <positive_dimension>
    [(type {<clearance_type>})]) |
   (junction_type [term_only | all]) |
   (limit_bends [<positive_integer | -1]) |
   (limit_crossing [<positive_integer | -1]) |
   (limit_vias [<positive_integer | -1]) |
   (limit_way [<positive_integer | -1]) |
   (max_stub [<positive_dimension> | 0]) |
   (max_total_vias [<positive_integer | -1]) |
   (reorder <positive_integer>) | [(tjunction [on | off]) |
   (via_at_smd [off | on [(grid [on | off]) [(fit [on | off])]]) |
   (width <positive_dimension>)]
```

Separate edge to edge clearance can be assigned by object type such as wire_smd (wire against SMD pad). The object types are:

- Wire (routing tracks)
- Pin (through hole pin)
- Smd (surface mount pad)
- Via (Via defined in DSN file)
- Area (Copper area, keepout, board outline)

Examples:

1. Setting PCB width and clearance:

```
rule pcb (width 8)
```



```
rule pcb (clearance 8)
rule pcb (clearance 6 (type wire_via))
```

2. Setting a layer rule applying to all nets in a design:

```
rule layer s1 s2 (width 6) (clearance 6)
```

3. Setting a special net rule:

```
rule net sig1 (clearance 12)
rule net sig2 (clearance 10 (type via_pin))
```

4. Limiting wrong way routing distance:

```
rule net clk (limit_way 150)
```

This will force the router to use a via if it exceeds the wrong way limit.

5. Limiting the number of vias per connection:

```
rule pcb (Limit_vias 4)
rule class fast (limit_vias 1)
```

Prevent via use:

```
rule class c1 (limit_via 0)
```

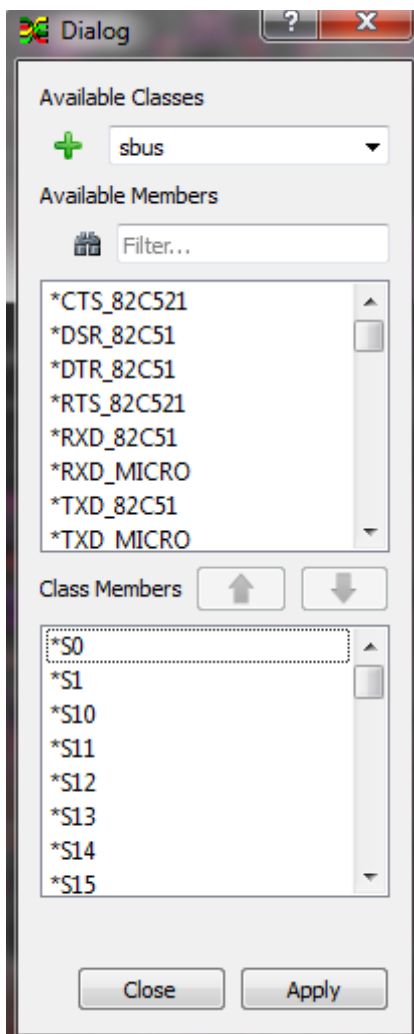
On entire design:

```
unselect all vias
```

7.1 Class Editor

A Class conveniently allows for managing multiple nets at the same time. All operations and rule definitions on a class are automatically applied to all its member nets.

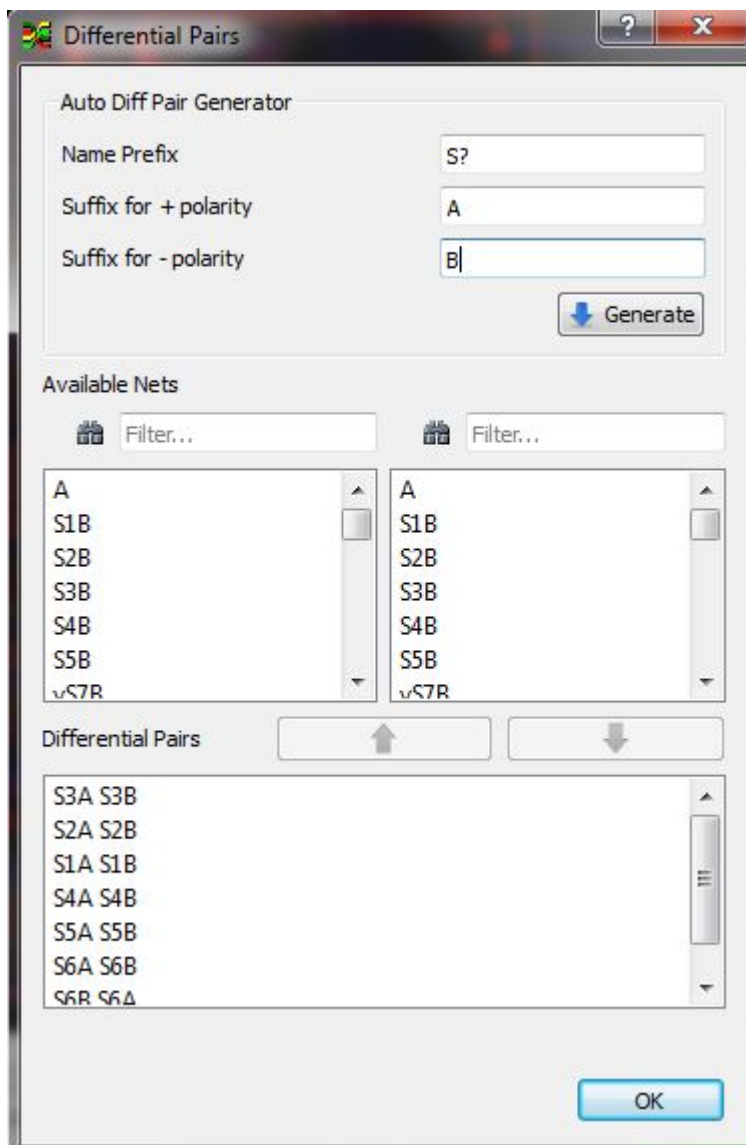
Available from the menu Design/Class.



7.2 Differential Pair Autorouting

Differential pairs can be user defined or automatically generated based on filters for the positive and negative member names:

From the GUI:



At the command level:

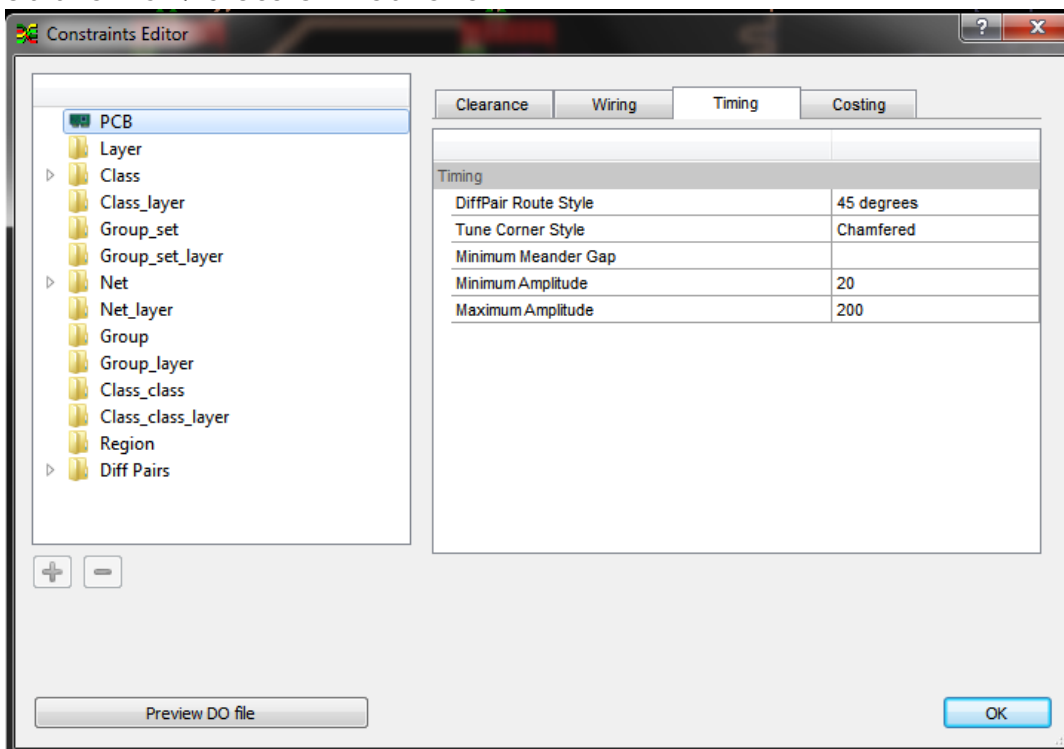
define pair <net1> <net2>

*autopair *_?N* *_?P**

Differential pair routing process starts by identifying gather points at the proximity of the fanout vias. The main part of the connections are then autorouted between the gather points. The autorouter applies the same multipass conflict reduction strategy to the differential pair nets and length constrained nets in order to reach the highest completion rate.

No T-junction or vias are used during routing of the differential pairs.

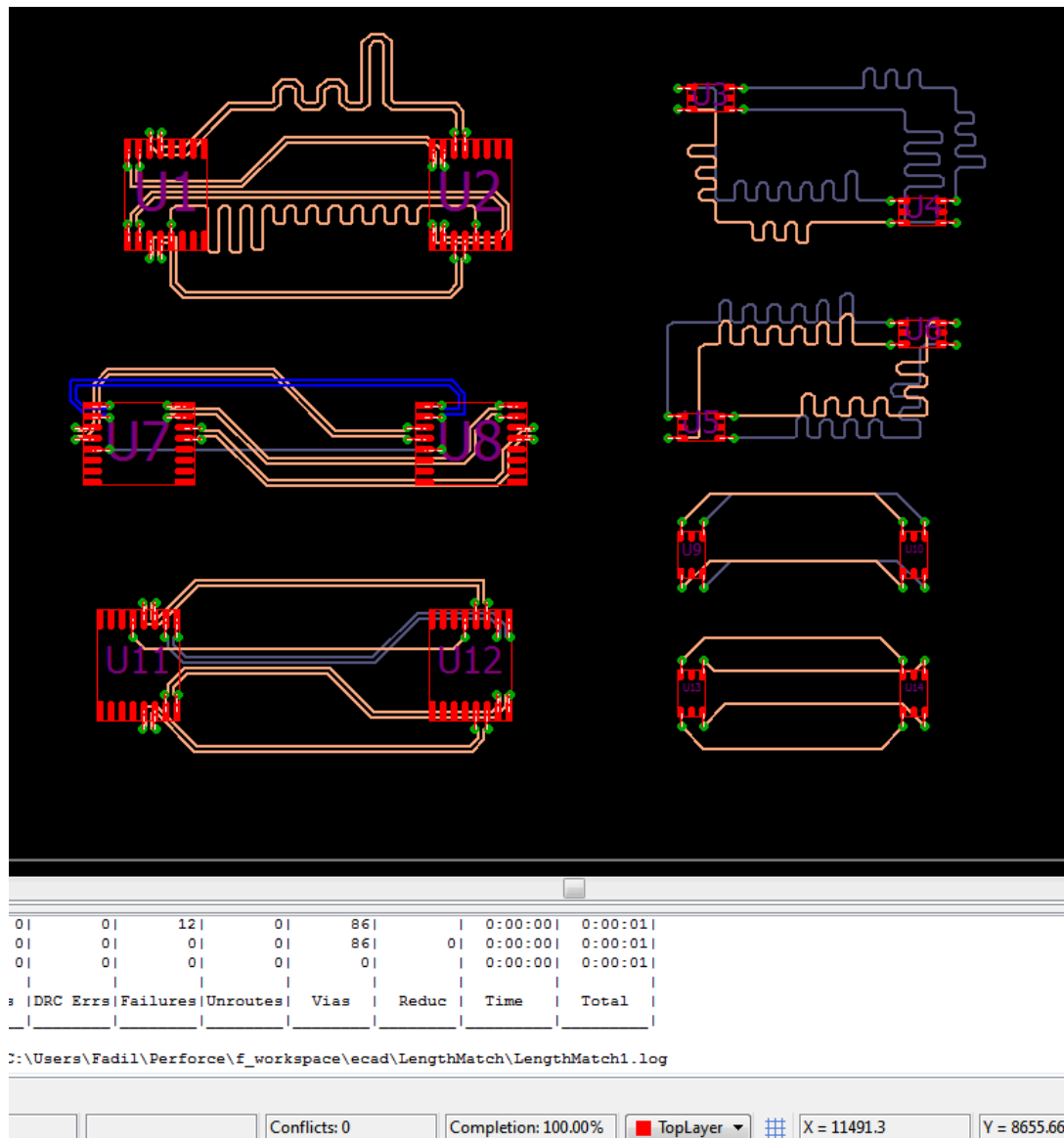
Diff pair 45 degree routing style, serpentine corner style, meander gap and min/max amplitude can be defined as “Timing” constraints at the PCB/Class or Net level.



7.3 Length Constrained Autorouting

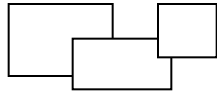
For min length and match length constraint, the router meanders routed wires to increase length by following an accordion pattern.

Includes the ability to define Signal Path containing connections within a net of a set of connections of different nets.



7.4 Autorouting by Fence

Especially useful to force the routing of sensitive nets to stay inside the fence area being a rectilinear polygonal shape. Use the menu Add/Fence to create overlapping rectangles resulting in a rectilinear polygon.

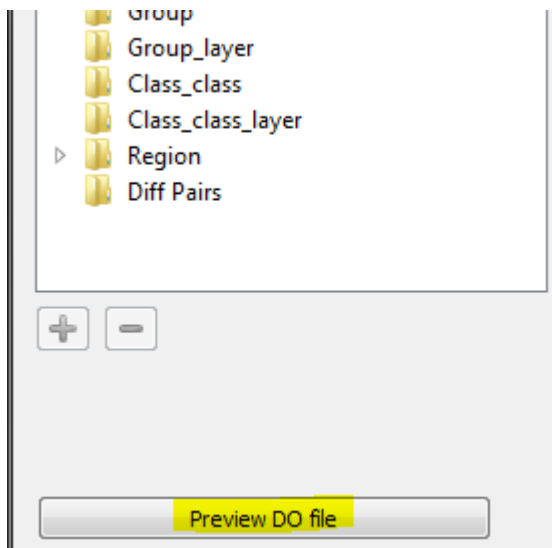


7.5 DRC Violation Browser

The violation browser shows the number of violations for every rule. Actual lengths of signal path (groups) are displayed, with automatic view-fit of the selected item.

7.6 Preview DO file

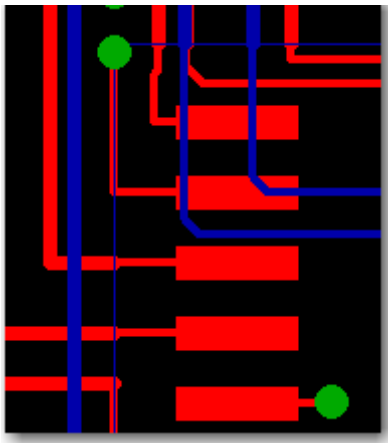
Provides the ability to control the intent before executing the DO file. Invoke it from the Constraints Editor by clicking on the [Preview DO file] button. A text editor with file load/save capabilities allows for managing storage and retrieval of DO files.



7.7 Area Rules support

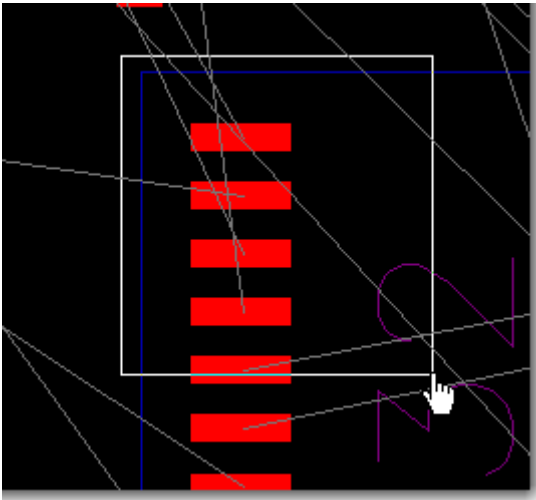
This is especially useful around small complex components such as BGAs and SMDs.

Also called region rule, it is a user defined rectangular area specified on specific layer(s) and a user defined track width and clearance within the area. The automatic router will adhere to the defined area rules during autorouting of nets that are crossing the region.

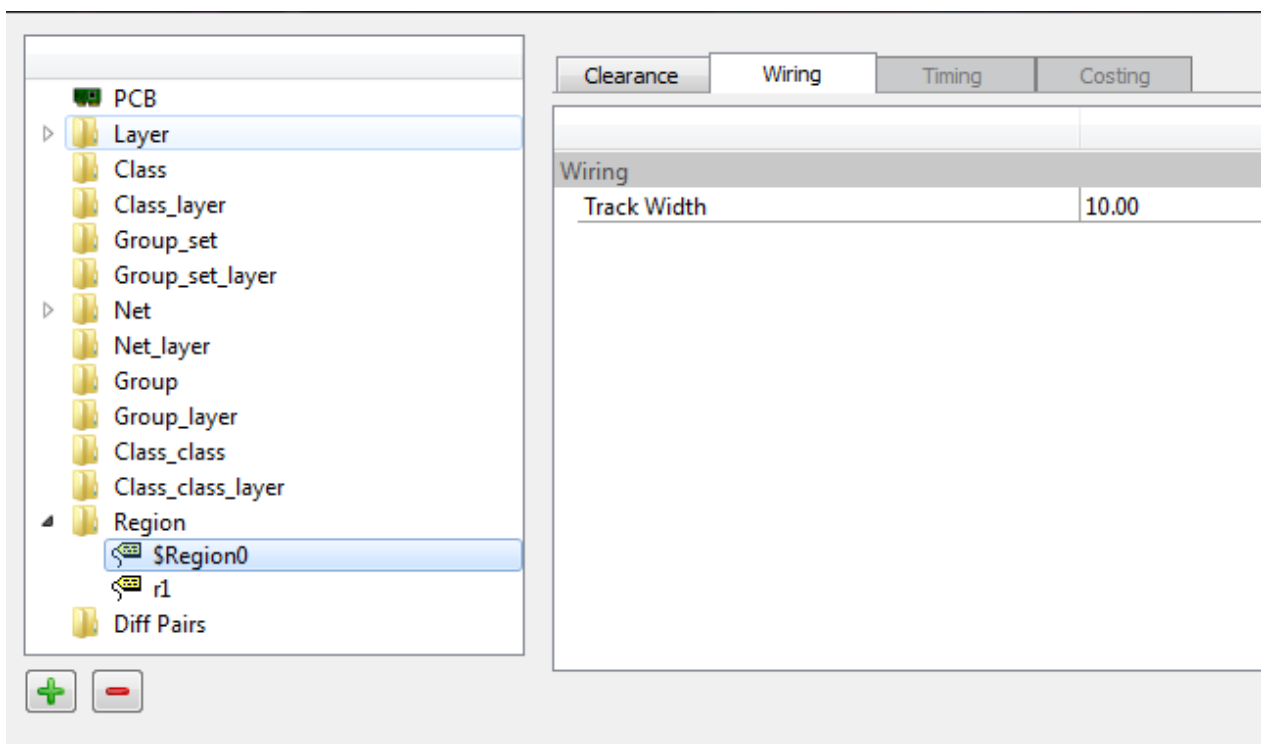


How to use it:

1. From the top menu bar, select Add/Region...
2. A cursor appears and you are prompted to define a rectangular area by dragging from one corner to the other corner of the rectangle. Make sure that the active layer shown on the layer Panel as the **layer name** in bold is the one desired for the region to be active on.



3. After the area is defined, you can change its width/clearance rule inside the Constraint Editor.



Note that the regions are automatically saved and restored the next time you reload the design. This is accomplished by

saving the user defined region definitions in the design directory. The region file is named after the design name with the .region extension appended. You can freely edit the file. See the following example:

```
demo.region
#
# Region file created by ELECTRA V2.00
#
# Syntax: add region <layer_name> width clearance x1 y1 x2 y2
#
unit mil
add region BOTTOM    6.000   12.000 9549.583 8638.587 10083.456 9306.394
add region TOP      6.000   12.000 9549.583 8638.587 10083.456 9306.394
|
```

8 Guided Autorouting

ELECTRA supports batch routing flow and active control of the autorouter. During batch routing, the order of the connections is determined by the autorouter and it is typically based on a sorting by length algorithm, while guided autorouting is about assisting the router to route structures in sequence.

As a preliminary step to batch autorouting, guided routing can be useful when a number of connections have to be routed as a topological group on specific layers or as a group of connections of predefined length range. For example, the user assists the router by defining bus structures to be routed, or short fromto's interconnecting nearby components on top/bottom layers to be routed first, then long tracks on inner layers.

Different ways to pre-select before routing:

1. Select a terminal to **route the connected net**
2. Select a Component to **route the attached nets**
3. Select a **Class** to route the net members
4. Select **fromto's by length** to route them
5. Select **fromto's by window** to route them

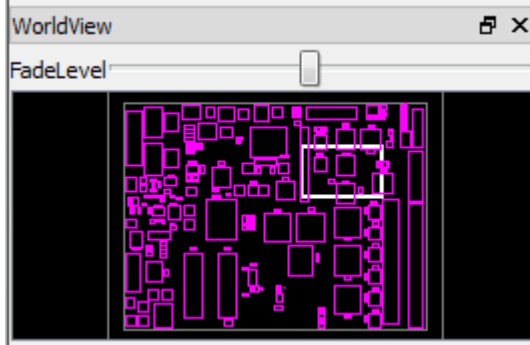
The last two methods are not net based but connection based. The selection of the pin to pin connections (fromto's) can be performed in two ways:

- a. By defining a rectangular area intersecting the ratline of the connections.
- b. By invoking a command to select the fromto based on minimum and maximum length criteria.

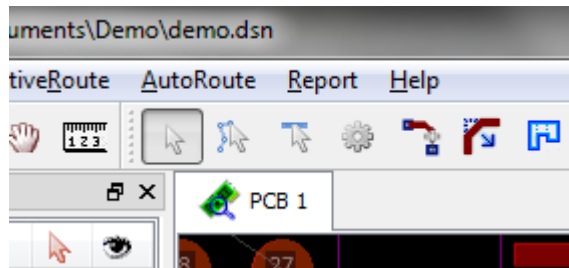
select fromto (length <min_val> <max_val>)

Tutorial on Guided Autorouting

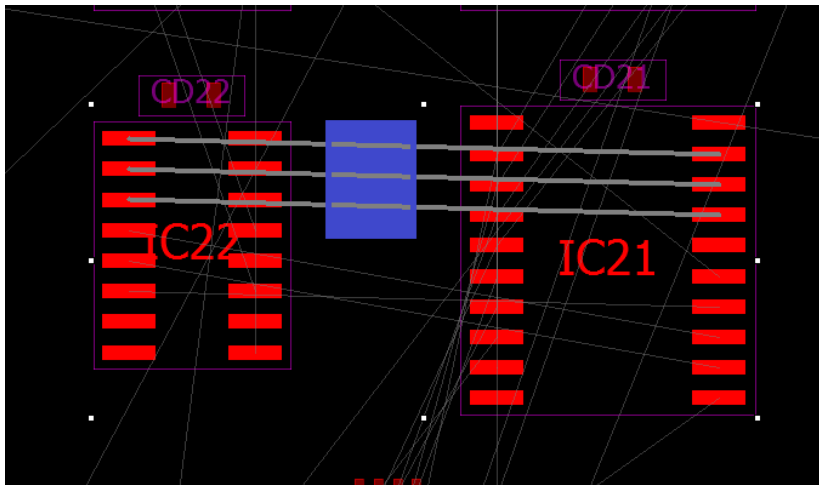
1. Load demo.dsn
2. Zoom to area including IC22 and IC21



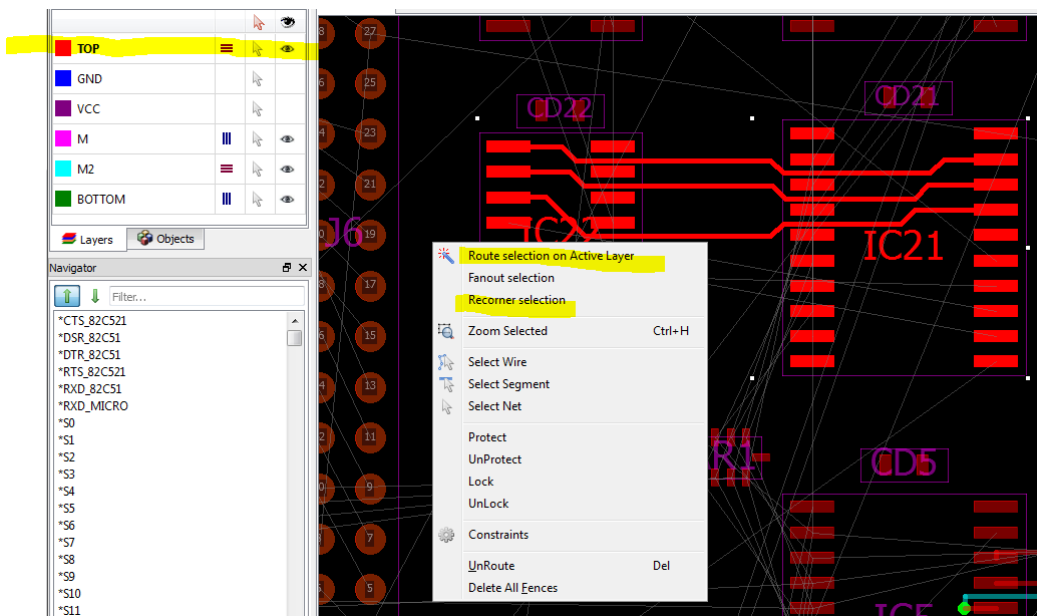
3. Toolbar default is the "select mode" cursor:



4. Select the ratlines by defining the rectangular area by dragging from one corner to the other corner of the rectangle, shown below in blue.
Make sure that the active layer shown on the layer Panel as the **layer name** in bold is the one desired to be active on.

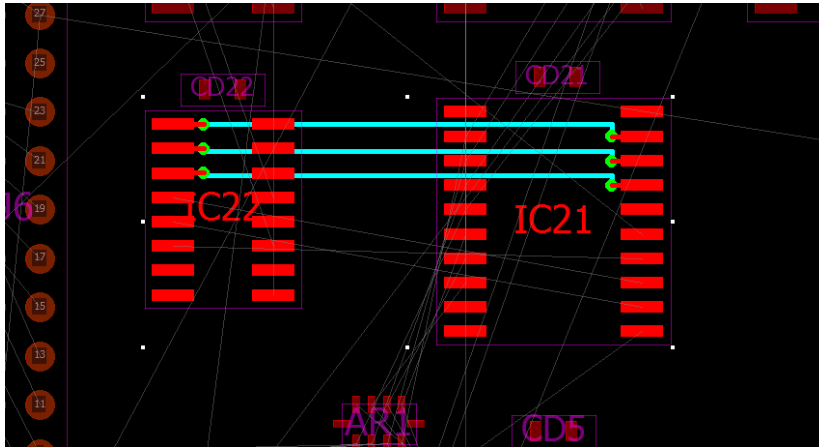


5. The 3 connections are highlighted. Press RMB to bring up the context popup menu and select **Route selection on Active Layer** to route the selected connections on a single layer. Finish with RMB **Recorner selection**.

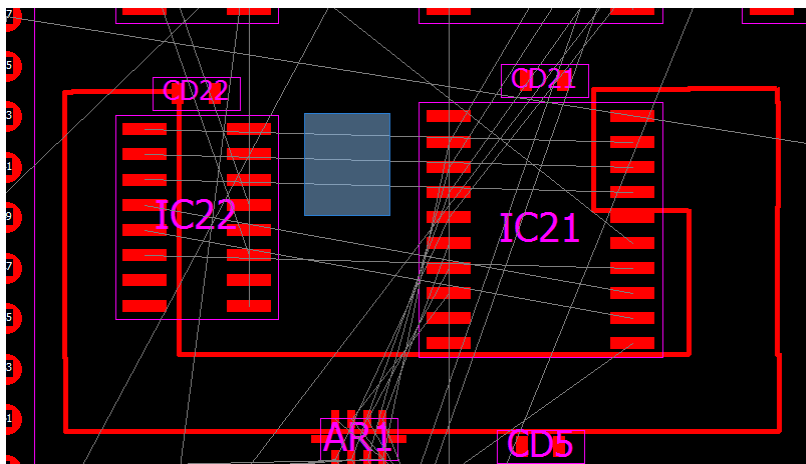


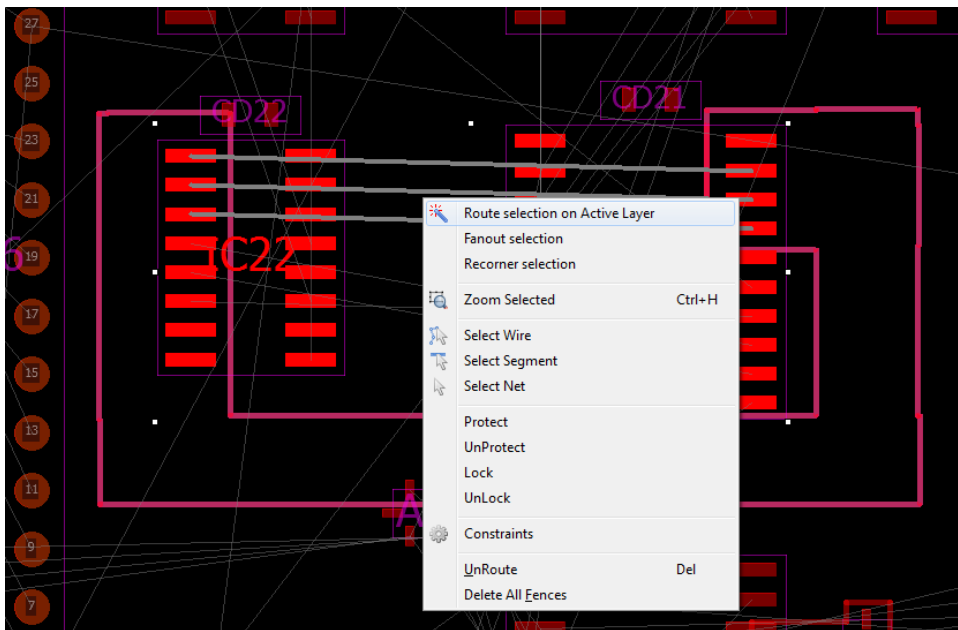
6. Try routing on bottom layer. RMB Unroute. From the layer panel, change active layer to BOTTOM. RMB Fanout selection. RMB Route selection.

7. Compare with multilayer routing: Select the routed nets by window by defining a rectangle surrounding any group of 3 pins, RMB **Unroute**. Type in the command: **route**. The solution is multilayered and follows the predominant layer directions:

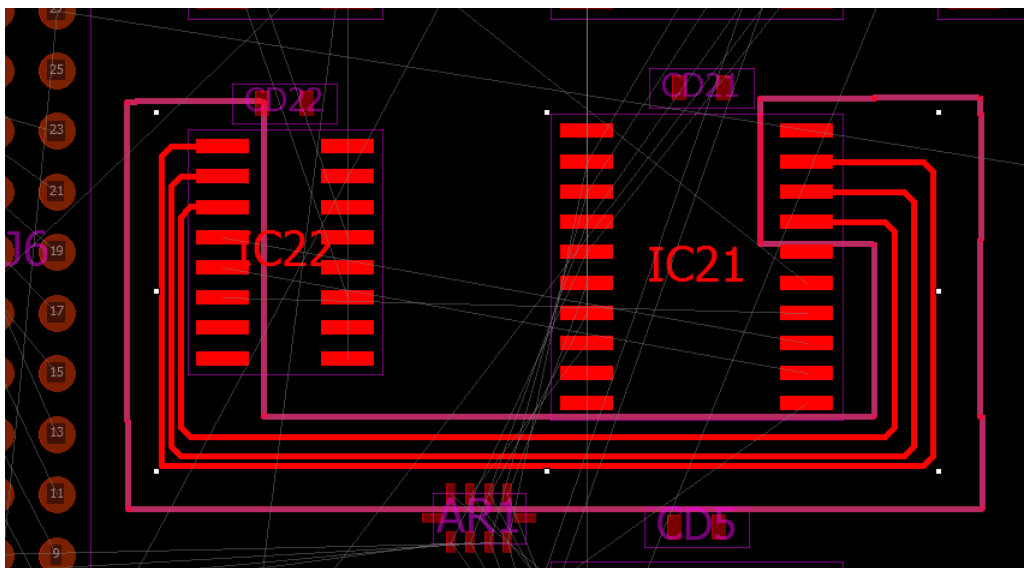


8. Right mouse button to bring up the context popup menu and select **Unroute** to remove the selected routes.
9. Let's define a keep-in zone. Select again the 3 connections and Use the menu Active Route/ Draw Fence as shown below and RMB to invoke **Route selection on Active Layer**





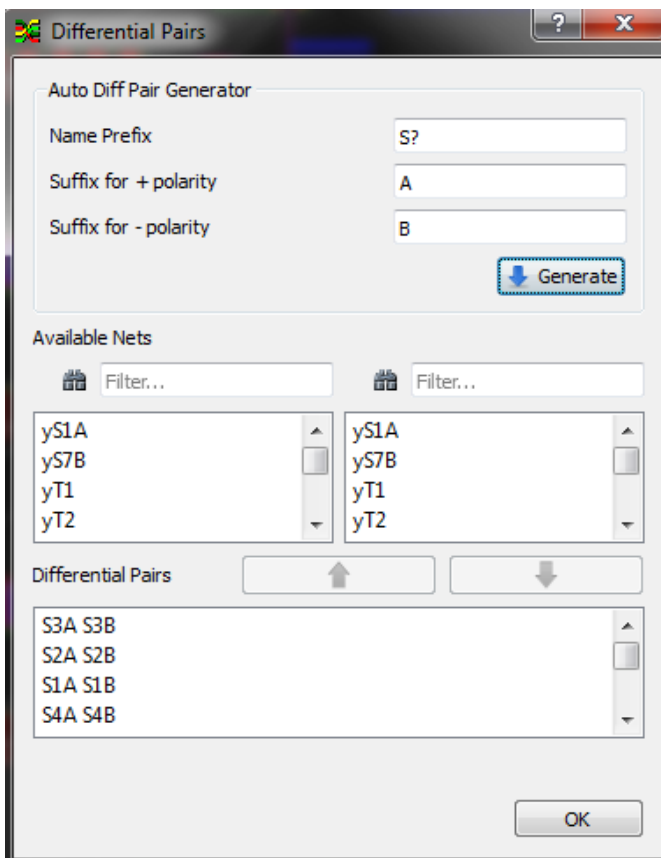
Final results showing the group of fromto auto-steered within the fence zone:



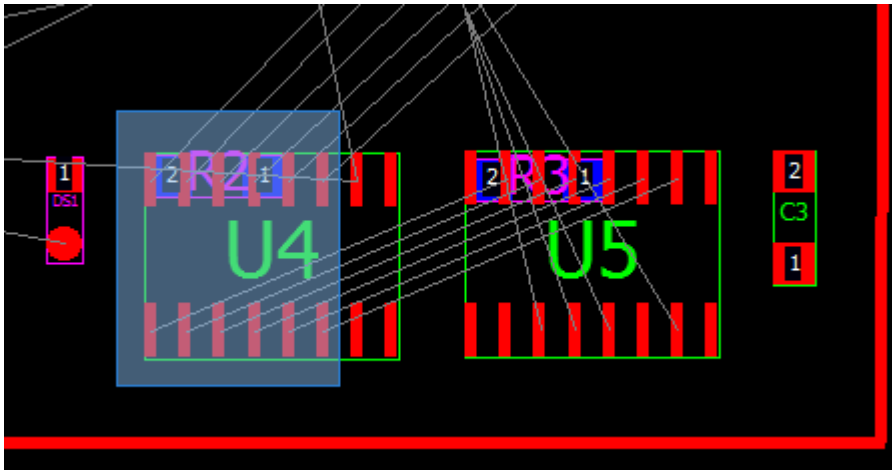
Selected routes can easily be protected or locked. When fences are not needed anymore, they can be removed with the menu (Edit / Delete All Fences).

Tutorial on High Speed Design with ELECTRA

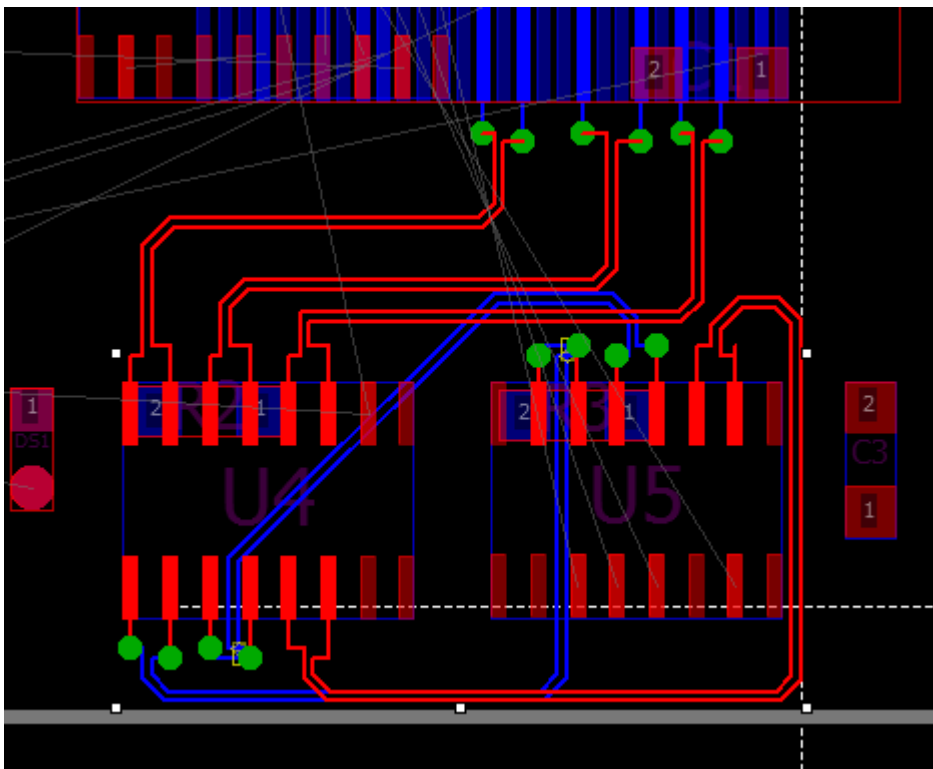
1. Start ELECTRA, select the menu **File / Load** and load the demo design located in <install_dir>\Demo\Fast1.dsn
2. Select the menu **Design / Differential Pairs**. A dialog box is shown as below. Type in **S?** for the name prefix and **A** for the + polarity postfix and **B** for the - polarity postfix. Click on the “**Generate**” button. A list of differential pair nets satisfying these conditions are automatically generated and results are shown at the bottom listbox. You can also manually select the nets to be removed or paired up, by using the **up** and **down** arrow buttons.



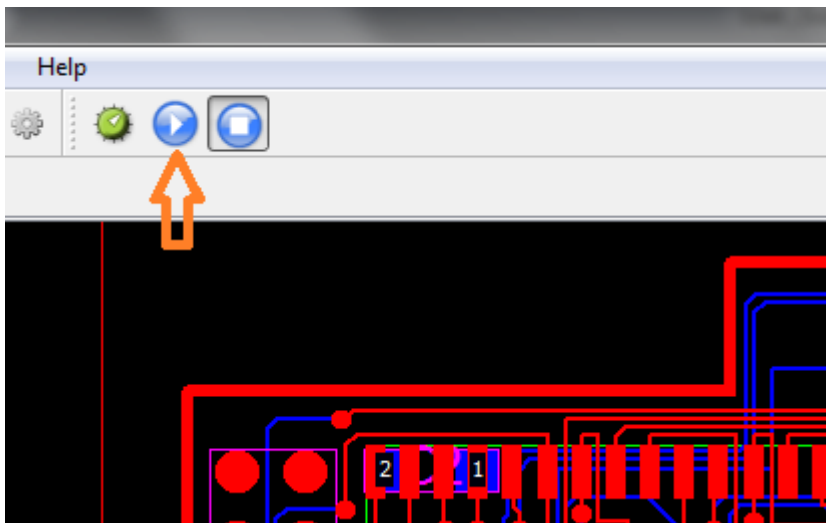
3. Select the differential pairs by dragging a window around U4 as shown below:



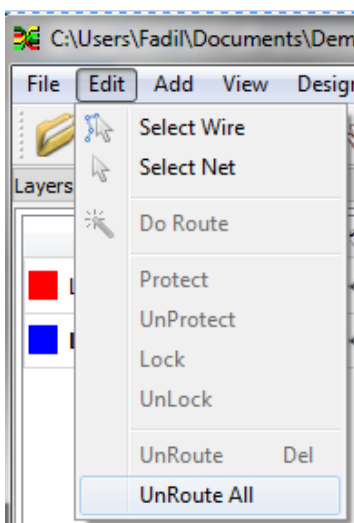
4. Right mouse button to bring up the context popup menu and select **Do Route** to route the selected connections in one pass. You should get routed differential pairs like this:



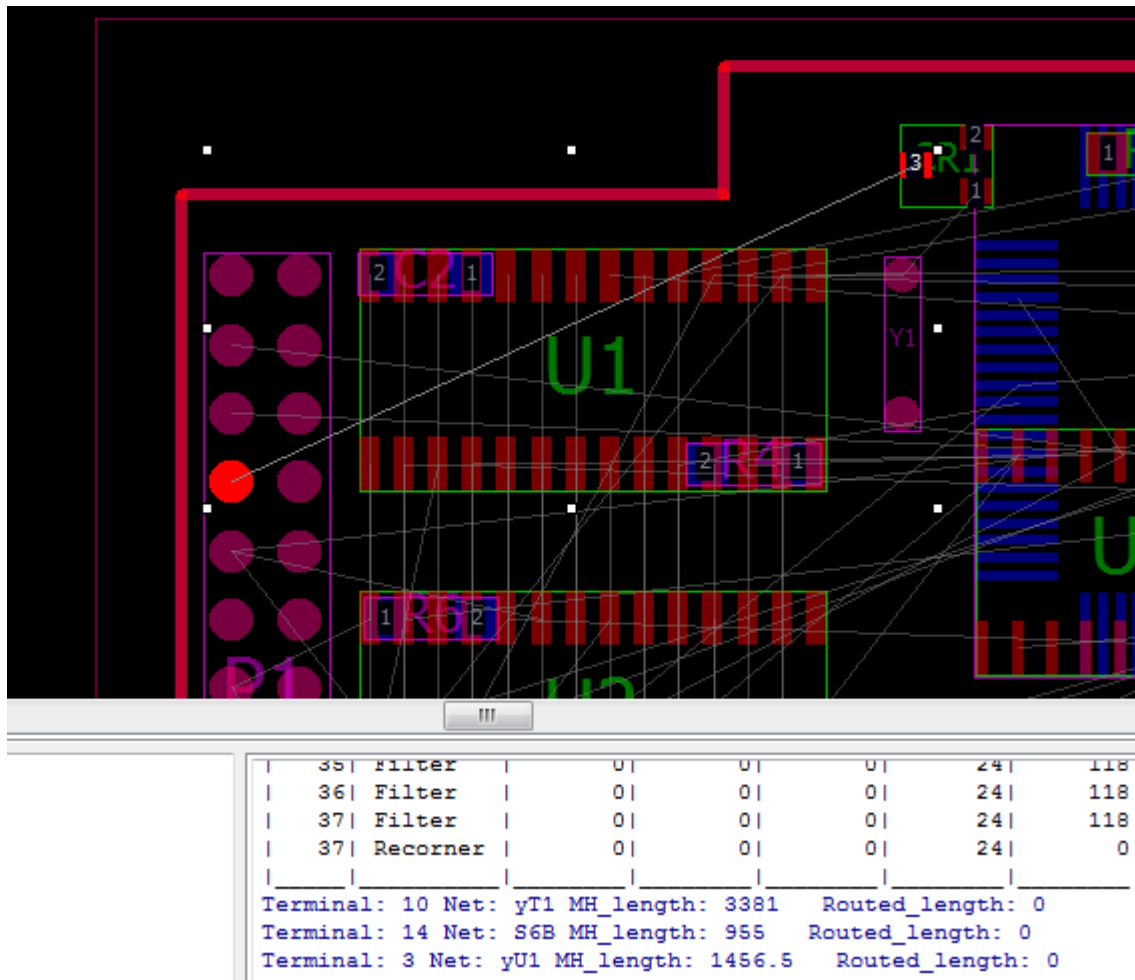
5. Differential pairs are treated as signal nets and **take part into** the adaptive autorouting strategy with conflict resolution. Click in an empty area to unselect all items and invoke the autorouter from the toolbar by clicking on the **play** button.




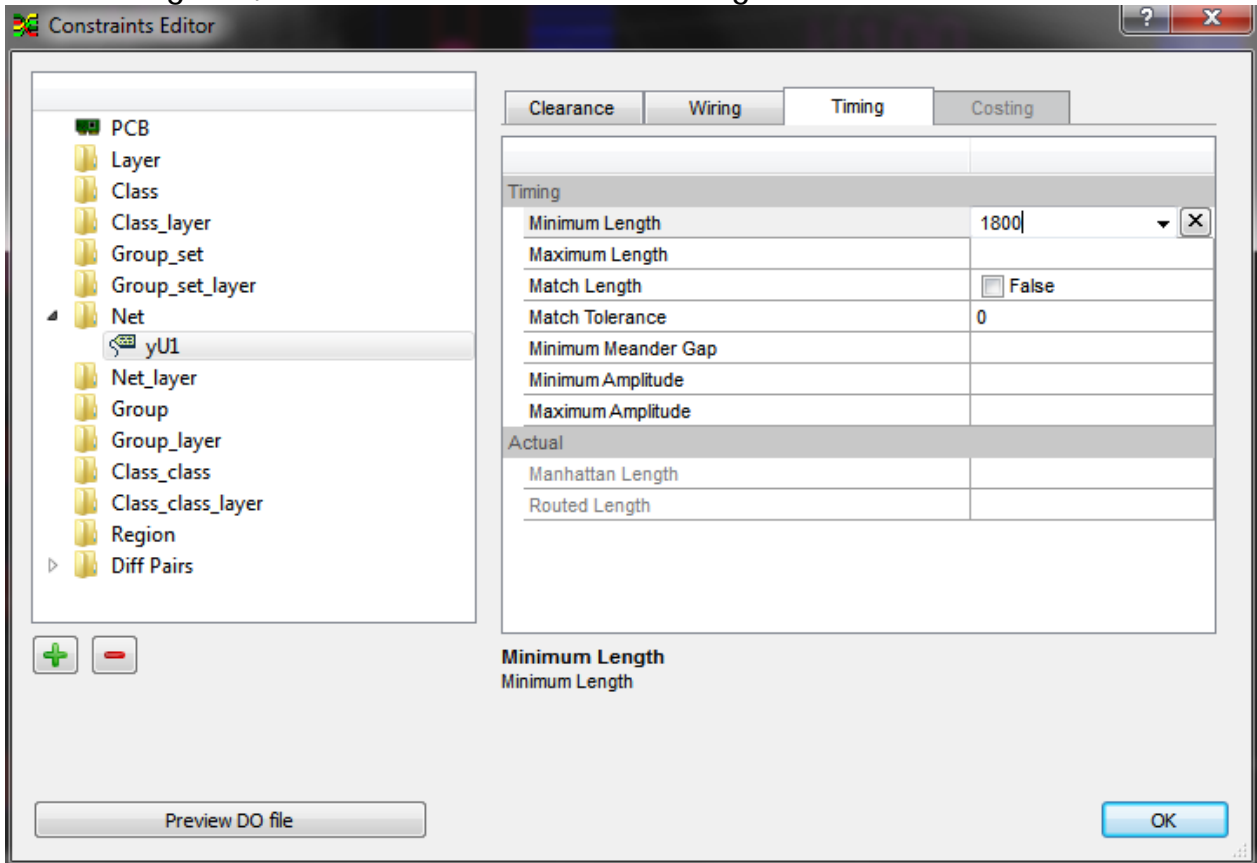
6. Select the menu Edit/Unroute All to unroute all unprotected tracks:



7. Specify a constraint to route a net at a **minimum of length**:
Select net yU1, note that its Manhattan length is reported as being 1456.5 mils. We will specify that a minimum of 1800 mils is required.



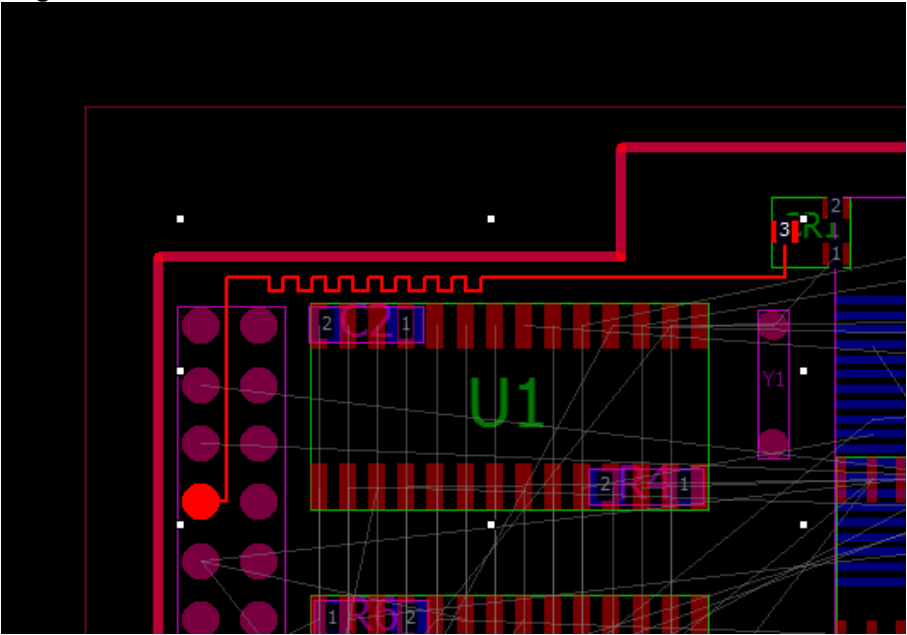
From the menu Design/Constraints we will add a net constraint by clicking on the Net branch and adding a net rule . A list of nets will be shown, with the preselected net as the default selection. In the Timing tab, enter 1800 as minimum length.



Outer and inner layers have different impedance characteristics. Layer-based constraints for a set of connections are typically used in order to meet target impedance. A set of connections can belong to design objects such as nets, classes, class_class, groups or group_sets; their rule type names are postfixed with the "_layer" text.

In addition to meeting electrical constraints, the ability to assign specific rules for each design class item provides optimal usage of routing resource and conserves design space.

Right mouse menu, select Do Route



8. Review **Timing** report. Select the Timing tab at the command panel at the bottom. Because we are within the tolerance, the routed length is colored in Green

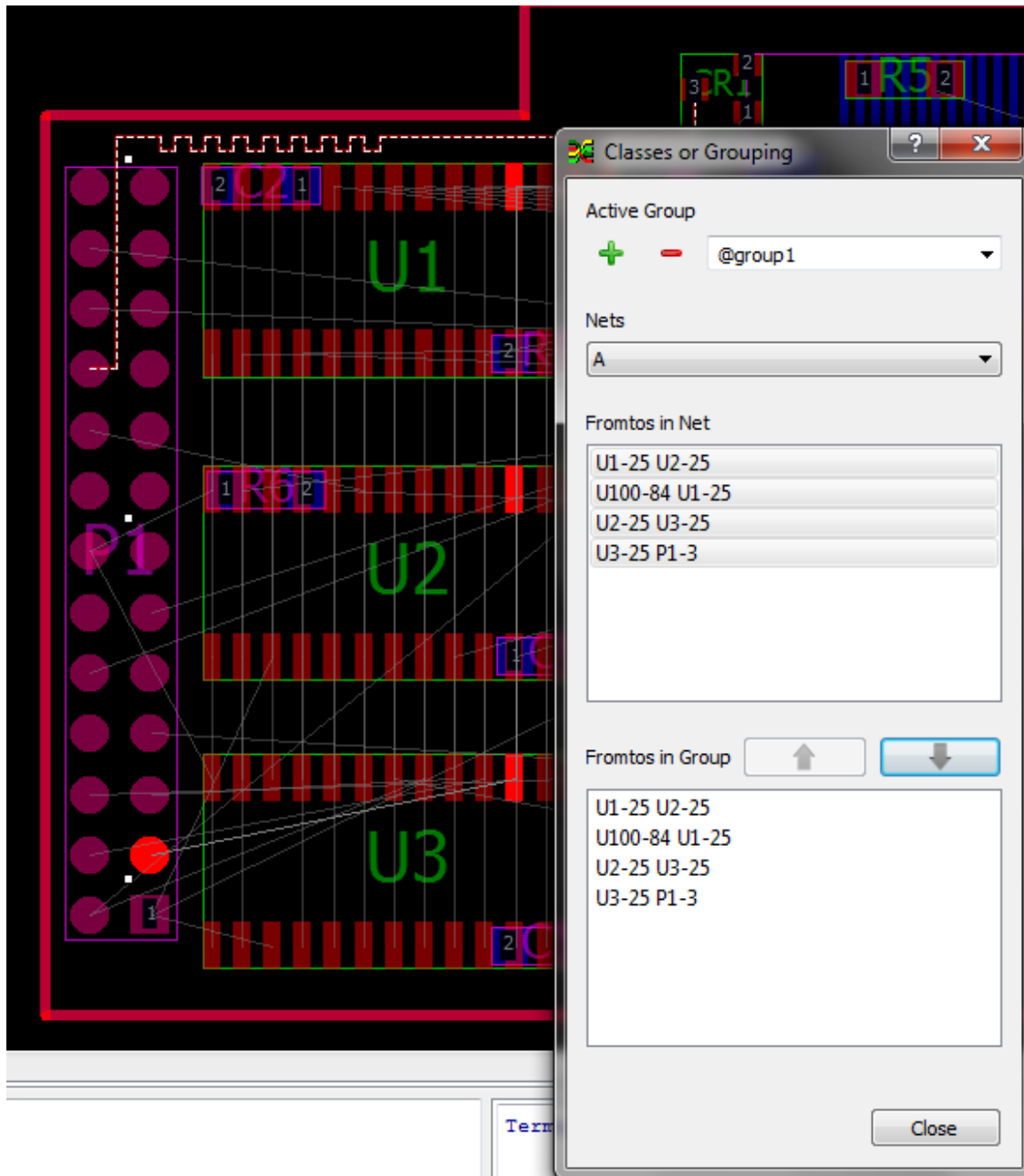
A screenshot of the same PCB layout with the Timing report window open. The window shows a table with the following data:

Design Object	Manhattan Length	Minimum Length	Routed Length	Maximum Length	Match	Tolerance
1 yU1	1456.5	1800	1840.5			10%

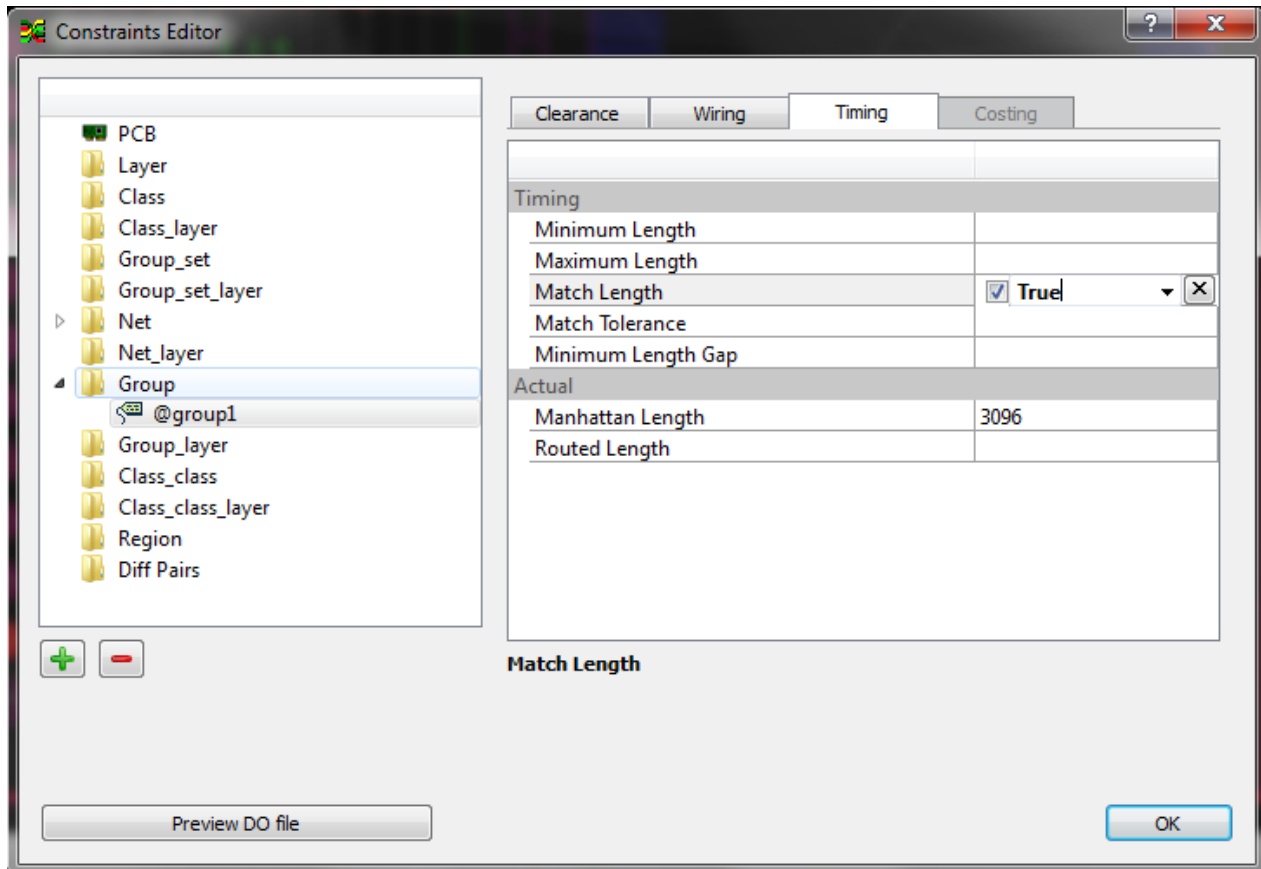
The Routed Length (1840.5) is highlighted in green. The Minimum Length (1800) is also highlighted in green. The Match column is empty. The Tolerance is 10%. The window also shows a list of components on the right and a command panel at the bottom with tabs for Log History and Timing.

9. Right mouse button menu, select Protect.

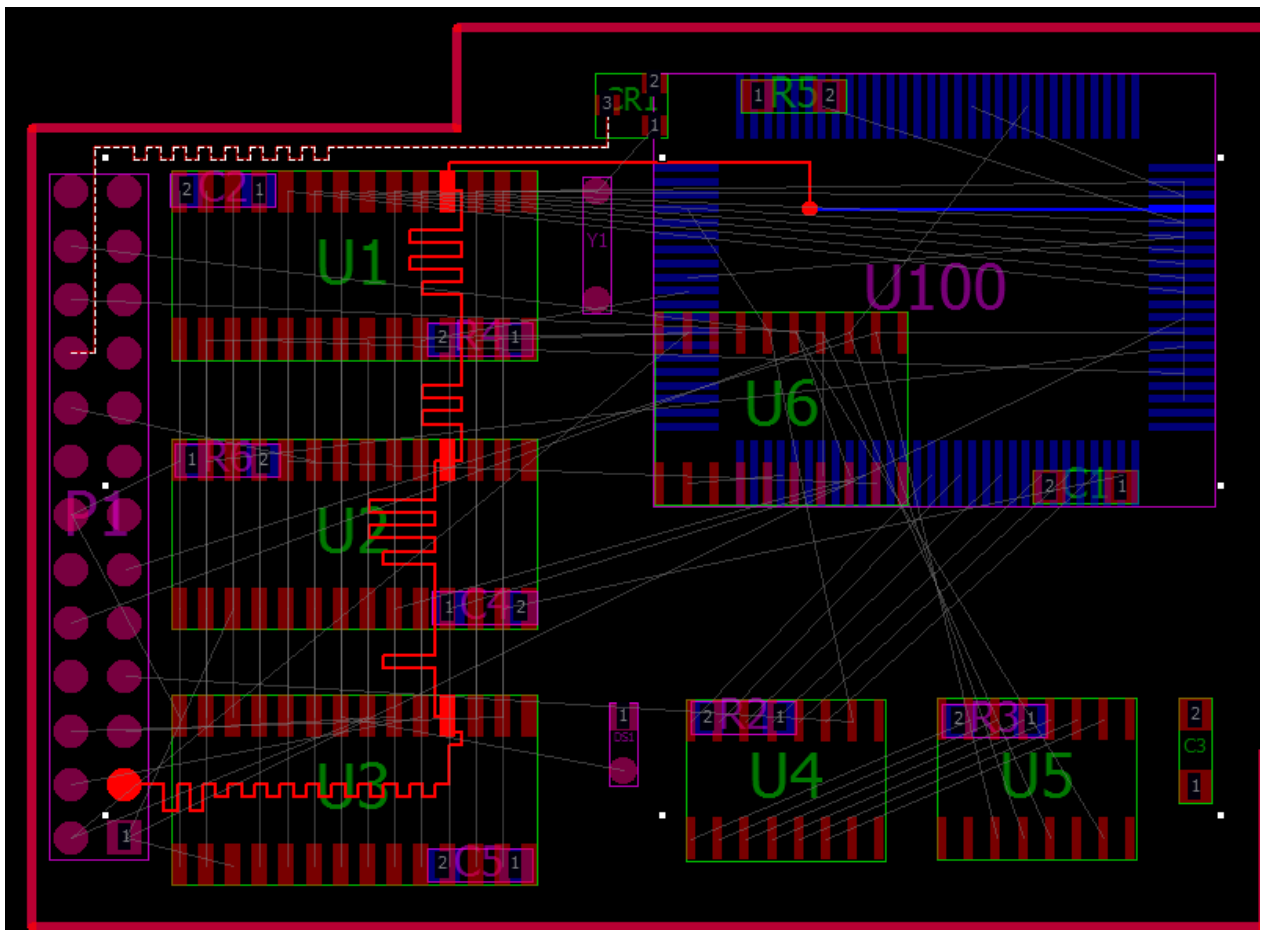
Now we will specify a length matching rule. From the left Net panel, select the first one, net A. Bring up the Group editor and assign the net fromto's to the default named group (@group1). (See screenshot)



Go to the Constraints Editor to add a group rule for @group1 and check the Match length rule in the timing tab, as shown on the next screenshot.



Close the dialog box and invoke one route pass Do Route from the popup menu.

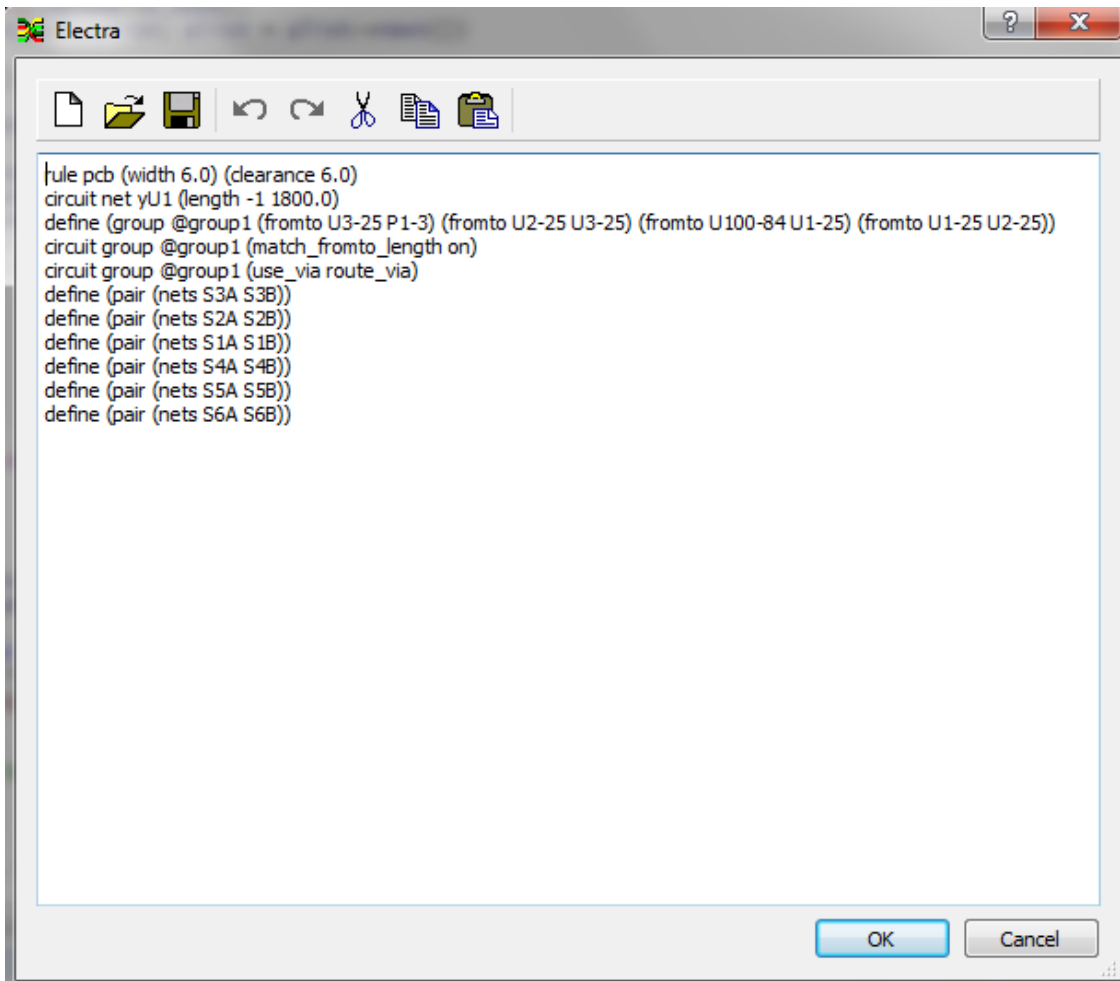


Group	Design Object	Manhattan Length	Minimum Length	Routed Length	Maximum Length	Match	Tolerance
Group Set	1 @group1 U3-25 P1-3	725		1397		<input checked="" type="checkbox"/>	10%
Diff Pair	2 @group1 U2-25 U3...	475		1391		<input checked="" type="checkbox"/>	10%
Net	3 @group1 U100-84 ...	1396		1502		<input checked="" type="checkbox"/>	10%
Class	4 @group1 U1-25 U2...	500		1366		<input checked="" type="checkbox"/>	10%

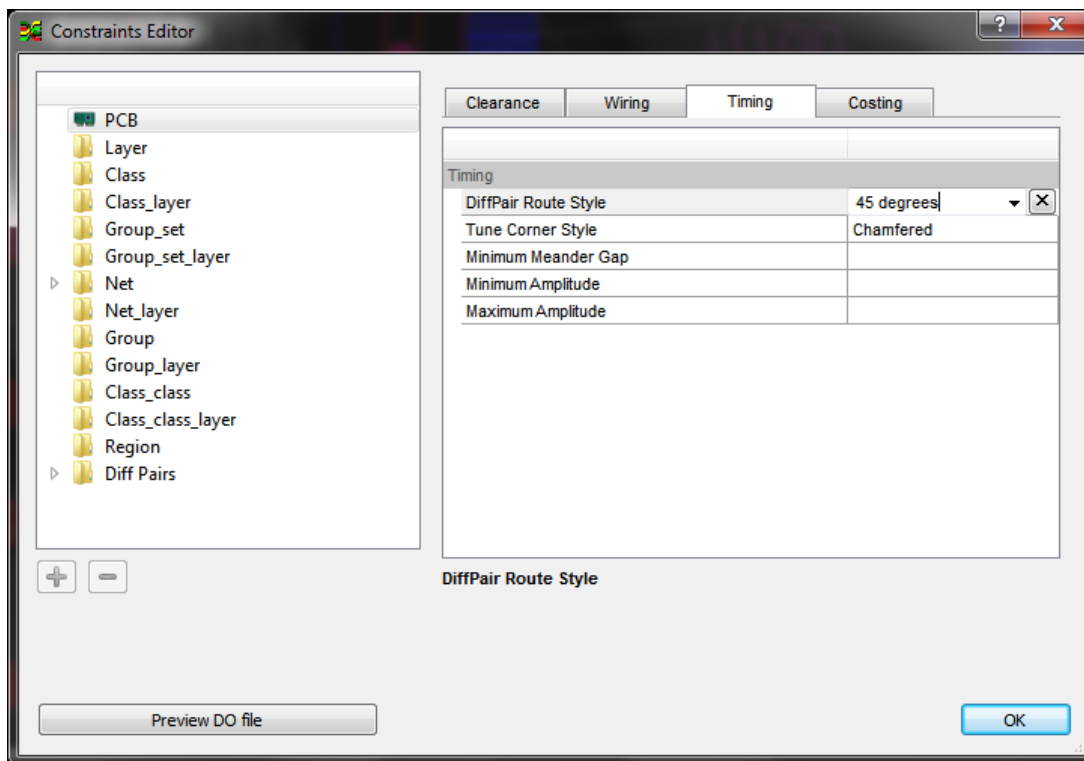
Log History Timing

With the net A still selected, protect the routed wires by using the « Protect » menu item.

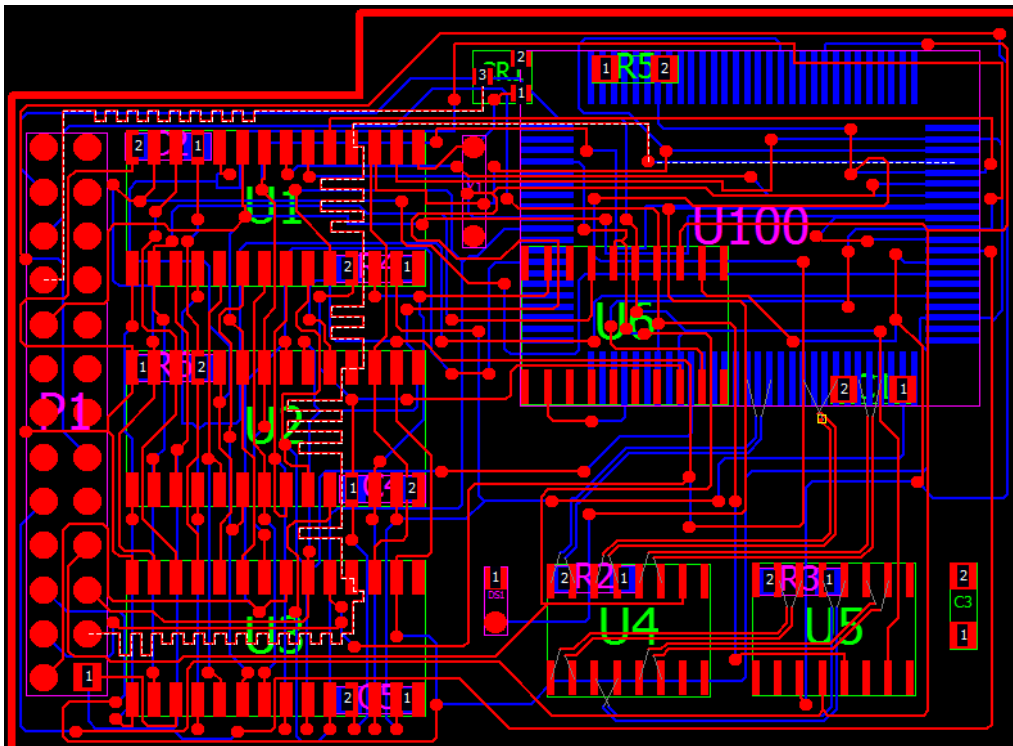
From the Constraints Editor, there is a button to [Preview DO file], it allows for file management of the constraints as a DO file for future reuse.



Prerouted nets with length constraints are now protected. Before proceeding with the autorouting of all the remaining nets, we will set the preferred routing of diff pairs to be using 45 degrees, and this under the Constraints Editor:



See the autorouter results:



9 Quick Command Reference

9.1 Notations

[] = option
| = OR
{ } = parameter inside can appear multiple times
<> = includes a descriptor

9.2 Descriptors

<circuit_descriptor> ::=
 [<length_descriptor> |
 <match_fromto_length_descriptor> |
 <match_group_length_descriptor> |
 <match_net_length_descriptor> |
 (priority <positive_integer> |
 (use_via {<padstack_id>}) |
 (use_layer {<layer_name>})]

<class_descriptor> ::=
 (class <class_id> [{<net_id>}]
 [(circuit {<circuit_descriptor>})]
 [(rule <rule_descriptor>)]
 [{<layer_rule_descriptor>}])

<class_class_descriptor> ::=
 (class_class (classes <class_id> {<class_id>})
 [(rule <rule_descriptor>)]
 [{<layer_rule_descriptor>}])

<cost_descriptor> ::=
 [forbidden | high | medium | low | free | <positive_integer> |
 -1]

```

<group_descriptor> ::=
    (group <group_id>
      {(fromto [<component_id>] [<component_id>]
        [<circuit_descriptor>]
        [<rule_descriptor>]
        [<layer_rule_descriptor>])

<group_set_descriptor> ::=
    (group_set <group_set_id> {[<group_id>]}
      [<circuit_descriptor>]
      [<rule_descriptor>]
      [<layer_rule_descriptor>])

<layer_rule_descriptor> ::=
    (layer_rule { <layer_name>} <rule_descriptor>)

<length_descriptor> ::=
    (length <max_length> [<min_length>])

<match_fromto_length_descriptor> ::=
    (match_fromto_length [off | on] [(tolerance <pos_val>)])

<match_group_length_descriptor> ::=
    (match_group_length [off | on] [(tolerance <pos_val>)])

<match_net_length_descriptor> ::=
    (match_net_length [off | on] [(tolerance <pos_val>)])

<pair_descriptor> ::=
    (pair {[netA netB]})

<rule_descriptor> ::=
    [(clearance <positive_dimension>
      [(type {<clearance_type>})]) |
      [(junction_type [term_only | all])] |
      [(limit_bends [<positive_integer | -1])] |

```

```

[(limit_crossing [<positive_integer | -1])] |
[(limit_vias [<positive_integer | -1])] |
[(limit_way [<positive_integer | -1])] |
[(max_total_vias [<positive_integer | -1])] |
[(reorder <positive_integer>] | [(tjunction [on | off])] |
[(via_at_smd [off | on [(grid [on | off])] [(fit [on | off])]]] |
[(width <positive_dimension>)]

```

9.3 Types:

```

<clearance_type> ::=
    [<object_type>_<object_type> |
     smd_via_same_net | via_via_same_net | buried_via_gap |
     antipad_gap | pad_to_turn_gap | smd_to_turn_gap | drill_gap]

```

```

<cost_type> ::=
    [way | cross | via | off_grid | off_center | side_exit | squeeze]

```

```

<object_type> ::=
    [pin | smd | via | wire | area]

```

```

<order_type> ::=
    [starburst]

```

10 Command Reference

add

add region <region_name> <layer> <width> <x1> <y1> <x2> y2>

Add a region to define an area where crossing wires require different width and clearance.

autopair

autopair <N polarity net expr> <P polarity net expr>

Automatic net pairs generator based from expression

autoroute

Invokes a general purpose routing strategy.
The strategy can be customized by changing the command file located in the executable directory. The command file is named "basic.do".

bestsave

bestsave on | off

Controls automatic saving of the best routing solution during a multi-pass routing run. Routing result is saved into

a file named "bestsave.rte" at the same location as the design file.

bus

bus [diagonal]

Invokes a "collinear pins" routing pass only on regular array of pins with collinear connections where the pins share a common X or Y coordinate. This is particularly effective on memory arrays. In this mode, the router will not generate conflicts, so rules must allow for sufficient space. By default, traces are routed orthogonally unless the diagonal option is specified.

check

This command can be used to run a DRC (design rules check) and visually tag the violations. This is used in particular when a rule is changed. A check is automatically invoked after every routing pass. The total number of violations is shown on the status line and visual feedback is added to the layout view to indicate conflict locations.

circuit

*class <class_id> | net <net_id> | group | group_set
{<circuit_descriptor>}*

Used to schedule the routing order priorities, specific vias to be used and allowed routing layers amongst nets and net classes.

<circuit_descriptor> ::=
 [<length_descriptor> |

```

<match_fromto_length_descriptor> |
<match_group_length_descriptor> |
<match_net_length_descriptor> |
(priority <positive_integer> |
(use_via {<padstack_id>}) |
(use_layer {<layer_name>}))

```

The value of **priority** ranges from 0 to 255, the default is 10.

The **use_via** rule assigns one or more via padstacks to a class or a net. If more than one padstack is defined, the autorouter will favor the smallest padstack in size.

The **use_layer** rule assigns routing layers where nets and classes must be routed. Note that the **use_layer** rule will override a layer unselection rule.

Examples:

```
# Routing a net / class to specific layers
```

```
net sig1 (circuit (use_layer L1 L2))
```

```
class fast (circuit (use_layer M1 M2))
```

```
# Assigning routing priority
```

```
net sig1 (circuit(priority 200))
```

```
# Routing a defined class to specific layers and using
specific vias
```

```
(class special net1 net2 net3 net4 net5
```

```
  (circuit
```

```
    (use_via via.VIA1__1_10.bv1_10
```

```
via.VIA3__4_5.bv4_5 via.VIA3__6_7.bv6_7)
```

```
    (use_layer SIGNAL_1 SIGNAL_3 SIGNAL_4 SIGNAL_5
```

```
    SIGNAL_6 SIGNAL_2)
```

```
  )
```

```
  (rule
```

```

        (width 0.007)
        (clearance 0.007)
        (clearance 0.001 (type via_via_same_net))
        (clearance 0.001 (type smd_via_same_net))
    )
)

```

clean

clean [*<passes>*]

The clean command reroutes all the connections with higher costs settings and helps achieve:

- Wire optimization with minimum bends
 - Vias minimization
 - Less off-center SMD pad entry
 - Exit SMD pad on long edge
-

cost

cost *<cost_type>* *cost_descriptors* /[(*type*[*length*|*way*]]] /-
1]]

User adjustable routing costs. This command sets the internally defined costs to a fixed value. By default, some of the cost values get internally modified during autorouting. It is not recommended to change cross and squeeze costing specifically.

Cost values can range from 0 to 100. A value of -1 will reset the cost value. Predefined cost description values can be used:

<u>Cost</u>	<u>Value</u>
Forbidden	100
High	50
Medium	25
Low	8
Free	0

The following costs can be set:

<u>Option</u>	<u>Description</u>
cross	crossing conflict
squeeze	wire to wire clearance conflict
via	wire to via clearance conflict
way	cost of routing in non preferred layer direction
off_grid	Cost of routing off grid if a grid was specified
off_center	Cost of entering or exiting a SMD pad off center
side_exit	Cost to exit SMD pads on long side
layer	If type is length it is the cost of using the layer. If type is way , it is the cost of routing on non-preferred direction

Examples:

cost layer S1 forbidden
cost layer S2 high (type way)
cost via high

define

Class <<class_descriptor> | <group_descriptor> |
<group_set_descriptor> | <class_class_descriptor> |
<pair_descriptor>

delete

Delete all [wires | fences]
Removes all existing wiring, except for the protected wires.

design <filename>

Load design file

direction

direction <layer_name>
[horizontal | vertical | orthogonal | off]

Changes preferred routing direction by layer

fanout

fanout [<passes>]

```
[(direction[in_out | in | out])]
[(pin_share [on | off])] [(via_share [on | off])]
{[(pin_type [active | signal | power | unused | all | single])]}
[(max_len <positive_dimension>)]
```

AutoRoutes short escape wires with a via from SMD pads. Recommended on SMD boards having more than 2 routing layers. Fanout direction can be set so that fanout vias are added inside SMD components, and/or outside. Fanout can be limited by pin type, for example pins connected to power nets only.

Examples:

```
# 5 fanout passes
fanout 5
```

```
# Depth for blind & buried vias
fanout (depth opposite 2) (share_len 500)

fanout 5 (pin_type signal) (via_share on)
```

```
# Sets via to microvia & grid 25
grid via 25 MICROVIA
```

```
# Fanout using a grid of 25
fanout (via_grid 25)
```

filter

```
filter [<passes>]
```

At the end of a routing session you may end up with wires/vias in conflicts (crossing or too close to each other), so in order to remove

these conflicts the router has to unroute the minimum number of connections that get into conflict, "Filter 5" will do that.

gloss

gloss

Post route cleanup pass to remove extra bends and redundant vias, without rerouting the net.

grid

grid [*via* <positive_value> [<via_id>] |
wire <positive_value> [<layer_name>]

Specifies wire and via grid spacing.

Examples:

```
# use a routing grid of 8.333
grid wire 8.333
# use a different routing grid 5 on layer 1
grid wire 5 layer 1
```

limit

limit [*cross* [<positive_integer> | -1] |
via [<positive_integer> | -1] |
bend [<positive_integer> | -1] |
way [<positive_dimension> | -1]]

The limit command sets absolute limit values to be applied to each connection. Control is provided to limit maximum allowed number of intersecting wire, number of vias per connection, number of bends, and the maximum distance of non preferred (wrong-way) routing. The range of limit for *<positive_integer>* is 0 through 255. You can set limit values, perform some routing passes, and return to the default system values by executing a limit command with a value of -1. If you don't supply limit values, computed default values are used by the autorouter.

Examples:

```
limit via 2
limit way 200
# resets routing limit to default value
limit way -1
```

lock

lock net <name>

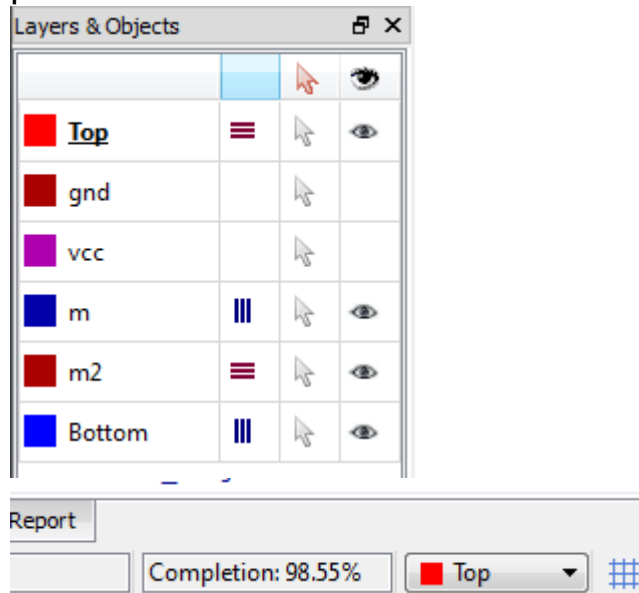
Prevents router from changing or deleting locked wires. Note that the router is not allowed to connect to locked wires.

lroute

lroute

Single layer router. Routes selected nets without using via, on the current active layer. Use the command "set active_layer <signal_layer_name>" to change the current active layer.

The active layer can be changed from the GUI on the layer panel and/or on the status bar as follows:



miter

*miter [diagonal] <pin_setback> <slant_setback> <tjunction
setback> <bend_<start_setback> <end_setback>>*

The miter command changes 90 degree wire corners to 135 degrees. It is performed on wire corners exiting pins and vias, as well as bend and slant wire configuration.

Pin_setback, slant_setback, tjunction and bend_setback options can be used to control which corner type is changed and you can override the default setback values as well:

```
miter pin .2
miter slant .2
miter bend .2
miter tjunction .2
```

In comparison with the recorner command, the miter functionality adds control for tjunction and start/end setback value for the bends.

recorner

```
recorner [diagonal] <pin_setback> <slant_setback>  
<bend_setback>
```

The recorner command changes 90 degree wire corners to 135 degrees. It is performed on wire corners exiting pins and vias, as well as bend and slant wire configuration.

Pin_setback, slant_setback and bend_setback options can be used to control which corner type is changed and you can override the default setback values as well:

```
recorner pin .2  
recorner slant .2  
recorner bend .2
```

Pin option is for wires that are connected to a pin or via.

Slant is two consecutive 90 degree bends

```
recorner diagonal .25 .25 .25
```

You can make multiple calls with decreasing values.

```
recorner diagonal .50 .50 .50  
recorner diagonal .25 .25 .25  
recorner diagonal .10 .10 .10
```

route

route [*<passes>* [*<start_pass>*]]

When the number of completed passes (n) is less than 15, the *<start_pass>* should be n+1.

When the number is higher than 15, then the *<start_pass>* should be 16.

rule

rule [*pcb* | *layer* *<layer_name>* | *class* *<class_id>* | *net* *<net_id>* |] {*<rule_descriptors>*}

Rules can be set globally (PCB) or specifically to layers, net classes or nets.

There are two categories of rules type: clearance and wiring rules as shown by the rule descriptor below.

<rule_descriptor> ::=
[(clearance *<positive_dimension>*
[(type {*<clearance_type>*})]) |
(junction_type [term_only | all]) |
(limit_bends [*<positive_integer* | -1]) |
(limit_crossing [*<positive_integer* | -1]) |
(limit_vias [*<positive_integer* | -1]) |
(limit_way [*<positive_integer* | -1]) |
(max_total_vias [*<positive_integer* | -1]) |
(reorder *<positive_integer*) | ((tjunction [on | off]) |
[via_at_smd [off | on [(grid [on | off]) [(fit [on | off])]]) |
[width *<positive_dimension>*])]

select

```
select [layer | via] {<id>} |  
      [all [nets | vias | layers | classes | wires | fromto] |  
      [class] {<class_name>} |  
      [comp] {<comp_name>} |  
      [net] {<net_name>} |  
      [fromto ( length <min_val> <max_val>)]  
      [fromto Cmp1-pinname Cmp2-pinname]
```

Enables the autorouter to use specific layers and vias for routing. Allows for selecting nets, fromto's, components and classes for routing.

Note: Names are case sensitive

status_file

Creates a status file with routing history and located in the design directory. The status filename is named after the design name, with the extension .sts.

tax

```
tax [way | cross | via | off_grid | off_center | side_exit |  
squeeze | layer <layer_name>] [<real> | positive_integer]
```

This is an alternative method of using the cost command. This is the recommended way to adjust costing. The tax command applies a multiplier to control internal costing. The default value for tax is 1.

The following costs can be set:

<u>Option</u>	<u>Description</u>
cross	crossing conflict
squeeze	wire to via clearance conflict
via	usage of a via
way	cost of routing in non preferred layer direction
off_grid	Cost of routing off grid if a grid was specified
off_center	Cost of entering or exiting a SMD pad off center
side_exit	Cost to exit SMD pads on long side
layer	If type is length it is the cost of using the layer. If type is way , it is the cost of routing on non preferred direction

Examples:

```
tax layer S1 1.2
tax way 1.2
tax via .5
```

unit

unit [inch | mil | cm | mm | um]

Set unit used from now until next unit change.

unlock

unlock net <name> / [all [layers | vias]]

Unlock existing tracks.

unprotect

Unprotect existing tracks. Allows router to reconsider them.

unselect

unselect [layer | class | net | via] {<id>} / [all [layers | vias]]

Disallow the autorouter the usage of specific layers and vias for routing. This command can be used to reduce the number of routing layers than originally specified.

Example:

unselect layer s1 s2

Note: Names are case sensitive

write

write [routes | session | script] [<filename>]

Command to save results to a routes file or a session file. The session file provides an integrated file to manage design data and routing data.

11 Knowledge Base of Tech Tips



11.1 Constraining Via Type Usage

Background:

ELECTRA can be constrained to use specific through-all, blind and buried vias. The via types that are defined in the PCB layout system are available in the DSN file and can be selected for specific requirements by using a DO file.

Selecting a via type for routing:

The select via command can be used to select the via type available for routing.

The default behavior is that all vias defined in a design file are selected and available for routing. Preventing the use of any via type(s) can be done with the unselect via command.

When there is a choice between multiple via types, the autorouter priority is to maximize routability and thus will use the via type having the fewest layers (in presence of partial via type) and also the via stack having the smallest size.

Example:

```
unselect all vias  
select via v1
```

From this point on, only the via v1 is used by the router.

Constraining the use of a via type by net and by net class:

The select via command is used on a global PCB scope. By default all via types are available for the router.

A use_via rule is available to force the use of specific via type(s) on a net and net class basis.

Example:

```
circuit net s1 (use_via V1)  
circuit class fast (use_via V1 V2)
```

Blind and buried via support:

A blind or buried via is considered as a via with shapes partially present on the layer stack. Blind and buried via type can be constrained in the same way as through hole vias and are identified by their padstack name. In addition, for blind and buried vias, you can also control the minimum distance allowed between the partial vias located on different layers. Note that same layer via clearance can be controlled with the usual via_via type rule.

Example:

```
rule pcb (clearance 25 (type buried_via_gap))
```

Buried_via_gap can be used to prevent the router from placing vias on the same Z axis and places them apart by a minimum distance between the shape edges.

This is an interlayer rule and does not affect same-layer via spacing rule, which is controlled with via_via type clearance rule.

```
---+  
  |  
+---
```

←→ MINIMUM GAP

```
---+  
  |  
+---
```

Inside the DSN file:

A padstack will exist for thru hole vias, buried vias and component pins.

For using specific via types, such as blind and buried vias, they need to be defined at 3 different locations in the DSN file:

At the top, simply by expanding the list of vias in the structure section:

```
(structure
  (boundary
    (path pcb 0.0 1.75555 1.98605 1.7568 1.98655 1.75695 1.98665 1.7571
    1.9867
    etc...)
    (via via_rnd_30_h15 via_rnd_30_h15_2_5.bv2_5 via_rnd_30_h15_5_6.bv5_6)
```

In the padstack section, the via type shapes are described:

```
....
(padstack via_rnd_30_h15_5_6.bv5_6
  (shape (circle SIGNAL_4 0.03 0.0 0.0))
  (shape (circle SIGNAL_2 0.03 0.0 0.0))
)
...
```

And assigning a wiring rule in the net section, by class and/or by net:

By class:

```
(class ADD 'ADD[0]' 'ADD[1]' 'ADD[2]' 'ADD[3]' 'ADD[4]' 'ADD[5]' 'ADD[6]'
'ADD[7]'
  (circuit
    (use_via via_rnd_30_h15 via_rnd_30_h15_2_5.bv2_5
    via_rnd_30_h15_5_6.bv5_6)
    (use_layer SIGNAL_1 SIGNAL_3 SIGNAL_4 SIGNAL_2)
  )
  (rule
    (width 0.005)
    (clearance 0.009)
    (clearance 0.006 (type via_via))
    (clearance 0.008 (type via_via_same_net))
    (clearance 0.008 (type smd_via_same_net))
  )
)
```

By net:

```
(net 0
  (pins
    U2-1
    U1-3
    U1-8
```

```

        U2-12
        U2-3
        U2-2
        U1-9
        U2-5
    )
    (rule
        (width 0.01)
    )
    (circuit
        (use_via via2)
    )
)

```



11.2 Cost and Tax usage

When and how to use the Cost command:

Usage of "cost" and "tax" commands requires special attention.

Before using the cost command, it is important to know how costing is used internally by the router. For each routing pass, a set of cost factors are generated; at the early stage of routing the cost of creating spacing conflicts and crossings is low and this increases with the number of passes. The cost function adjustment is not linear, it is a bit more complicated...

The "cost" command is used to fix a constant cost value, but the "tax" command is a cost multiplier that is applied to the internally calculated cost.

The router uses internal costing that varies based on the routing pass, so it is not recommended to change all cost parameters that are directly related to routing, such as "squeeze" and "cross", for the other parameters such as cost "via", "way", "off_center" or "side_exit" it is OK to change them as needed.

Here are some applications where the cost function can be used:

To minimize wire routing on a particular layer:

cost layer Bottom high (type length)

Usage of the Tax command:

Instead of using cost, it is advised to use the tax command if possible, this way the router still follows its internal cost schedule, which is key to converging on a solution.

Example:

```
tax squeeze .5
tax cross 1.3

cost way 50
cost via 8
cost off_center 2
cost side_exit 2

route 20
clean 2
```

Routing a 2-layer board with a minimum number of vias:

If you are routing a 2 layer board and looking at minimizing the # of vias, don't hesitate to run hundred of passes:

```
cost way low
cost via 70
route 20
clean 2
route 50 16
clean 2
route 100 16
clean 2
route 200 16
clean 2
route 300 16
clean 2
```

Another 2-layer board strategy:

```
rule pcb (via_at_smd on)
```



```
cost way low
cost via 70
route 20
clean 2
route 50 16
clean 2
route 100 16
clean 2
route 200 16
clean 2
route 300 16
clean 2
recorner diagonal
```

For dense 2 layers board, "many many" passes helps, but ONLY for 2 layers.



11.3 Route command <start_pass> parameter

Why specifying a start_pass parameter for the route command:

```
route 50 16
```

Start_pass allows presetting the basis for the pass number used to generate the cost factors. If you "route 10" and then another "route 10" without setting the start_pass the router will route again with low conflicts cost and it won't converge.

Example:

```
route 10
clean 2
route 50 16
clean 2
```

How many clean passes during routing?

```
route 10
```

```
clean 2  
route 50 16  
clean 2
```

There is no need to run more than 4 clean passes. These clean passes allow for optimizing the current route and also change their position, perturbing enough of the problem so that subsequent routing passes have better chances for converging.



11.4 Microvias

The router can be controlled to use specific via style during routing. Vias can be directed to be placed under the SMD pads with the following command:

```
rule pcb (via_at_smd on)
```

```
rule pcb (via_at_smd on (fit on))
```

With this “fit” option, you simply specify that the via shape needs to be fully enclosed within the smd pad.

To control the via style used for fanout:

```
select via (via_top_dyco1_signal_underpad  
via_bottom_dyco10_signal_underpad  
via_top_dyco1_ground_underpad  
via_bottom_dyco1_ground_underpad)  
fanout 5  
select all vias
```

Then proceed with autorouting...



11.5 Power and Ground planes

Autorouting with full planes and split planes are both supported. In case of a full plane, there is no need to specify a polygonal region, the board outline is used as limiting region.

In presence of split planes on same layer, polygonal areas need to be defined along with the associated use_net (see example below)

Here is an example of polygon outlines defining split plane areas, in the structure section, the router understands the presence of those polygonal areas and will assume connectivity just by plugging the proper via within the allowed area:

Inside the DSN file:

```
(structure
  (rule (width .010) (clearance .010))
)
(via p12 p13)
(grid wire .01)
(grid via .1)
  (boundary (path pcb 0.016
    1.200 0.120 3.090 0.120 3.090 0.540 3.290 0.540
    3.290 0.120 6.490 0.120 6.490 0.545 6.864 0.545
    6.864 0.320 7.380 0.320 7.380 4.480 0.800 4.480
    0.800 0.585 1.200 0.585 1.200 0.120))
  (boundary (path signal 0.008
    3.042 .170 1.242 .170 1.242 .642 1.092 .642
    1.092 .842 .892 .842 .892 4.142 1.092 4.142
    1.092 4.392 7.080 4.392 7.080 3.792
    6.892 3.792 6.892 3.442 7.292 3.442
    7.292 2.792 6.892 2.792 6.892 2.142
    7.292 2.142 7.292 .642 6.942 .642
    6.942 .592 6.442 .592 6.442 .170 3.342 .170
    3.342 .592 3.042 .592 3.042 .170))

(layer top (type signal))
(layer int1 (type signal))
(layer pwr (type power) (use_net +5V +12V))
(layer int2 (type signal))
(layer int3 (type signal))
(layer gnd (type power) (use_net GND AGND))
(layer int4 (type signal))
```

(layer bottom (type signal))

```
(plane +5V (polygon pwr 0.010 3.22 4.35 7.05 4.35 7.05 3.82 6.85 3.82 6.85 3.40 7.25 3.40
7.25 2.80 6.88 2.80 6.88 2.10 7.25 2.10 7.25 0.68 6.90 0.68 6.90 0.62
6.40 0.62 6.40 0.19 3.38 0.19 3.38 0.62 3.02 0.62 3.02 0.19 1.27 0.19
1.27 0.68 1.12 0.68 1.12 0.88 0.95 0.88 0.95 2.98 1.65 2.98 1.65 2.92
3.05 2.92 3.05 3.80 3.22 3.80 3.22 4.35))
(plane +12V (polygon pwr 0.010 1.125 4.35 3.15 4.35 3.15 3.875 2.975 3.875
2.975 3.0 1.725 3.0 1.725 3.05 .925 3.05 .925 4.1
1.125 4.1 1.125 4.35))
(plane AGND (polygon gnd 0.010 0.900 4.125 1.1 4.125 1.1 4.375 3.175 4.375
3.175 3.85 3.0 3.85 3.0 2.975 1.7 2.975 1.7 3.025
.9 3.025 .9 4.125))
(plane GND (polygon gnd 0.010 0.925 3.0 0.925 0.85 1.1 0.85 1.1 0.65 1.25 0.65
1.250 0.145 3.05 0.145 3.05 0.6 3.35 0.6 3.35 0.145
6.425 0.145 6.425 0.6 6.925 0.6 6.925 0.65 7.275 0.650
7.275 2.125 6.9 2.125 6.9 2.775 7.275 2.775 7.275 3.425
6.875 3.425 6.875 3.8 7.075 3.8 7.075 4.375 3.2 4.375
3.2 3.825 3.025 3.825 3.025 2.950 1.675 2.95 1.675 3.0
0.925 3.0))
```



11.6 Mixed Power Planes

How to mix polygon supply and tracks on same layer. This is what is called a "mixed" layer type in the DSN language.

For example for a two layer design, here is an example of layer structure with a "mixed" layer type at the bottom:

```
(layer "Top" (type signal) (direction horizontal)
(layer "Bottom" (type mixed) (direction orthogonal))
```

You can also tell the router to minimize usage of bottom layer for wires, here is the command before autorouting:

cost layer Bottom high (type length)



11.7 Autorouting Decoupling Capacitors

Controlled flow to route decoupling capacitor with tracks and the rest with via to planes. Redefine fromto's between major component power/ground pins and the nearby DCAP's with the "select fromto" command which creates the fromto if it does not exist and adds the fromto to the selection list, ready to be routed.

```
select fromto U1-4 P2-10  
select fromto U1-4 P2-10
```

```
setr active_layer TOP  
unit mm  
rule pcb (clearance 0.254 (type smd_to_turn_gap))
```

```
### Single layer detail router  
lroute
```

```
rule pcb (clearance -1 (type smd_to_turn_gap))  
protect selected  
unselect all wires
```

```
### Regular router, the power type nets will be power-  
route 5  
clean 2  
recorner diagonal
```

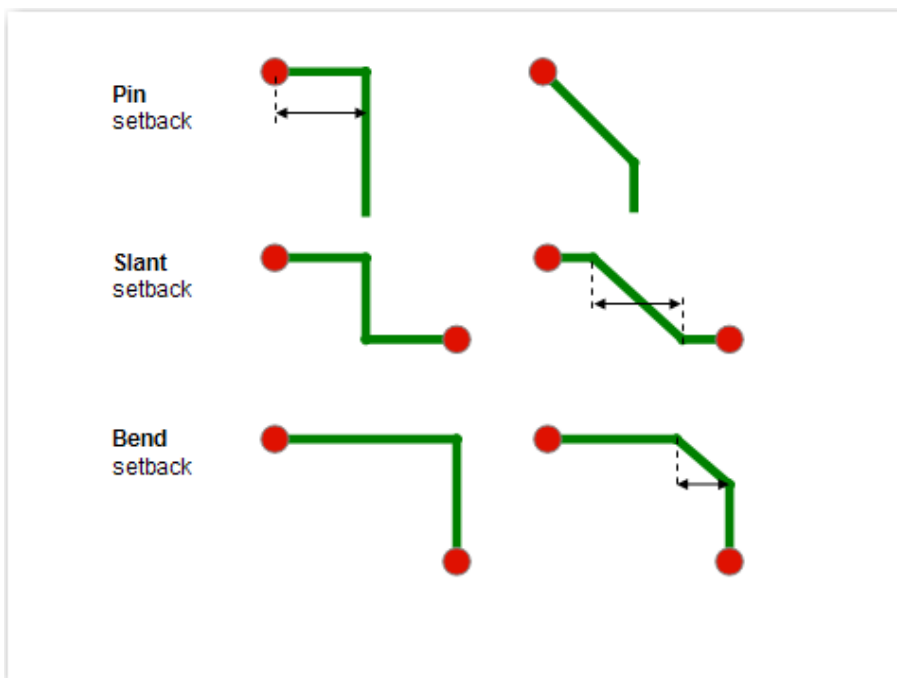
11.8 Recorner expert

With this command you can get 90 degree wire corners converted to 45 degrees (actually 135 degrees). It is performed on wire corners exiting pins and vias, as well as bend and slant (successive bends) wire configuration.

Pin_setback, slant_setback and bend_setback options can be used to control which corner type is changed and you can override the default setback values as well:

```
recorner pin .2
recorner slant .2
recorner bend .2
```

Here is an illustration of the 3 configurations:



Pin option is for wires that are connected to pin or via. Slant is two consecutive 90 degree bends You can specify the setback values for the recorner command.

A short version of the command:

recorner diagonal <pin_setback> <slant_setback> <bend_setback>

Successive calls should give better results, for example:

```
unit inch  
recorner diagonal 1 1 1  
recorner diagonal .5 .5 .5  
recorner diagonal .25 .25 .25  
recorner diagonal .2 .2 .2  
recorner diagonal .1 .1 .1
```



11.9 Filter 5

Removes the minimum number of wires that are in conflict. This removes tracks and vias but does not affect connectivity. Filter works in multiple passes in order to gradually remove the minimal number of wires that are in conflict. It is recommended leaving the number of passes to 5.

12 Appendix

Autorouting Techniques/Using ELECTRA with Protel

by Michael Reagan
C.I.D. EDSI Frederick, Maryland

Artwork and Autorouting

It is interesting that many engineers and designers believe that high-speed designs still require all manual wiring. Many also believe the "auto" in autorouting is the same functionality as the automatic transmission in a car - simply place the shifter into gear and depress go. Perhaps autorouters should have been named "interactive routers". Then they would perhaps find wider acceptance and would be used correctly more often.

Another common misconception among designers seems to be that autorouting does not create artistic designs, therefore the routing is inferior. After all, it is still artwork, is it not? No. For instance, traces tightly wrapped around each other might create eye-catching patterns, but those tightly-coupled traces may also cause a reset line to trigger, resulting in PCB Rev B. High-speed designs should generally be routed to meet electrical requirements and give less emphasis on eye appeal.

In this article and tutorial, I will try to cover the steps necessary for preparing a Protel PCB file for autorouting. These steps include:

- 1. Placement of major components for routability*
- 2. Interactive fanout of SMD components*
- 3. Taking advantage of Protel and 2004 design rules*
- 4. Defining keepouts, using steering graphics, protecting preroutes*
- 5. Pitfalls to avoid*
- 5. Creating route DO files*
- 6. Running the autorouter: Checking progression*
- 7. Retrieving and reviewing route results*

It is assumed that the reader has sufficient knowledge of the Protel 99SE or 2004 Design platforms.

Placement and Routing Strategy

Complex multilayer designs require a designer to think about routing strategy early in the design. It is critical to any design that components are placed to meet electrical performance requirements. With large devices such as BGAs and FPGAs, placement has to accommodate power distribution, as well as wiring. Component orientation should be determined by the best wiring solution, not the best artistic or assembly fit.

Correct component orientation is achieved when the rats-nest appears to represent a matrix of horizontal and vertical connections. Straight lines will allow the autorouter to complete connections using the shortest path. The use of this strategy makes it easy to understand why ConnectEDA's ELECTRA autorouter behaves and routes in the same logical manner as we think. If we position components to allow for line of site wiring, the ELECTRA router will achieve efficient, dense routes, normally reaching 100% completion.

A common problem with many designers is not understanding or "thinking" like the router. Designers goals should be to find the shortest path, use the minimal vias and the least number of layers. With high-density designs, the only proven method of achieving these goals is to route traces parallel to each other (reference the route pattern below). Note that these routing patterns may seem contrary to artistic design. However, they are functional and efficient, and are proven with high-speed digital logic.

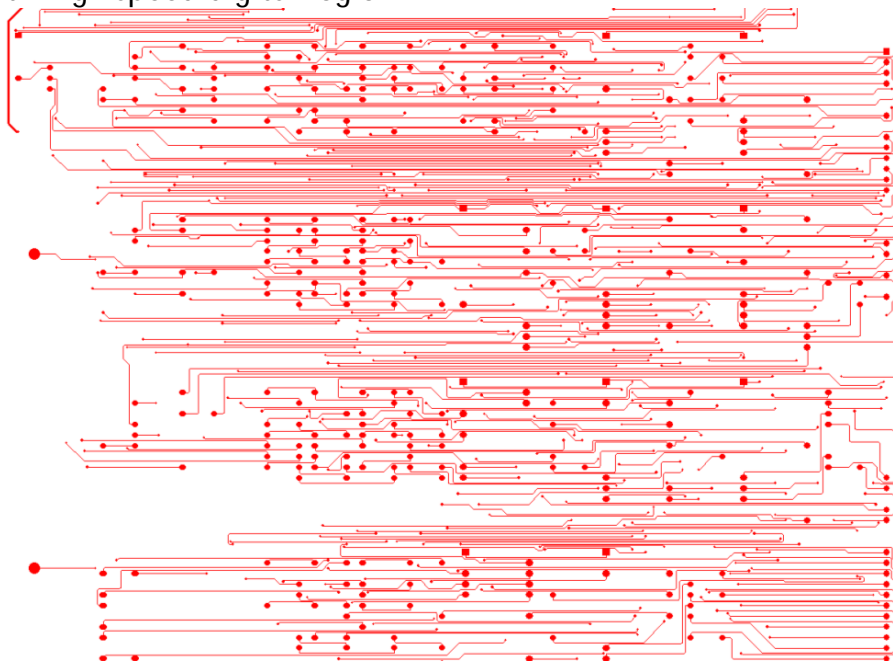


Fig. 1 - Clean and straight autorouted internal layer

Components pads and vias are obstacles to routing, regardless if the routing is performed manually or "automatically". Fewer obstacles exist on the inner layers than on the outer layers. Thus we must look to the internal layers to carry long connections. Internal layers provide the best opportunity for long connections to remain unobstructed. Since all the SMD pads on the outer layers are obstructions, the outer layers serve better for short connections.

As we have divided our connections into short and long traces, it is reasonable to create design rules that would include and exclude nets into outer and inner layer *routes*. You can choose to hand route some of the short nets on the outer layers during the fanout preparation stage.

Large components should always be placed first. After the large components are placed, discrete components such as decoupling capacitors and termination resistors can be added to your design.

Fan Out and Interactive Wiring

Second only to the component location, via fanout is the most important preparation task to achieve your routing goals. Careful attention to fanout will determine how quickly and efficiently the router achieves completion. Most often, when the router has not reached 100 percent completion, the problem can be traced to poor fanout. I do not recommend using automatic fanout routines for several reasons.

Performing manual fanout also allows the designer to review, rotate and move discrete components as required. A good fanout has every component pad tied to a via (reference the fanout pattern which follows). Vias should be positioned on grid and must allow for routing channels. A quick method for fanning components is to use pre-fanned components as library parts.

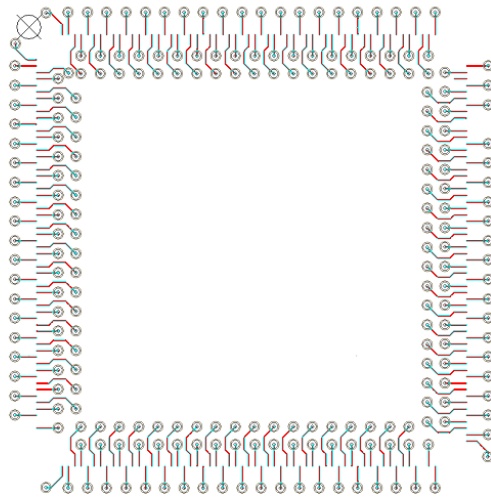


Fig. 2 - Pre-fanned QFP library part

One drawback to this approach is unused pads will contain stubs with a via such as unused pins in a BGA pattern. The extra vias do not pose a manufacturing problem. However, they always remain as an obstacle to the router. My preferred method and more tedious is to create fanout patterns for each size and pitch component.

The fanout libraries should contain patterns for SOICs, QFPs, BGAs, etc. After a placement review, copy these patterns from your FANOUT.LIB and snap them to the footprint pads.

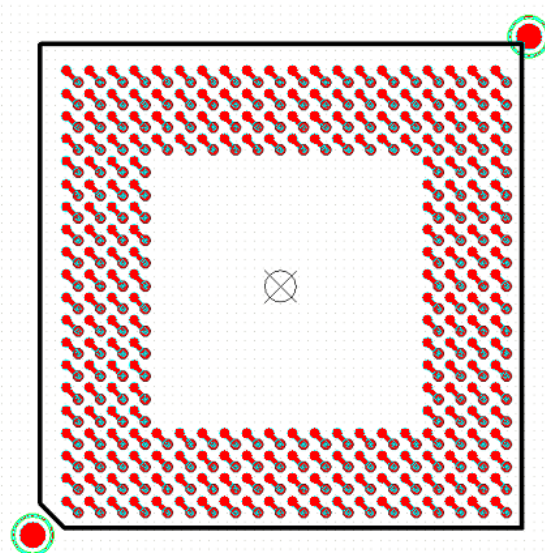


Fig. 3 - Pre-fanned BGA library part

After the entire board is fanned-out, globally remove and delete unnecessary pads, traces and vias from unused pads. Depending on the complexity and density of a design, fanout of the large components should take no more than a few hours maximum.

Design Rules

The objective in creating design rules is to translate the routing requirements in to design rules that the router can follow. Few engineers choose to embed routing rules within a schematic, so it is frequently left to the designer to extrapolate the rules. Design rules are not available in the Protel library editors. Therefore create your patterns on a working PCB file using via-to-via clearances, wire-width and trace clearance rules. A via-to-via clearance rule can be added by selecting object kind A, via and object kind B to via. Then set a clearance at $(2 \times \text{space} + \text{line} + 2 \text{ mils})$. For example, if the line and space are 5/5, then the total clearance will be 17. The extra two mils guarantees the router will slip traces through easily. If larger clearances are used, then valuable routing channels will be lost. It is important that via-to-via clearances are optimized to allow routes with little real estate loss. On very dense designs, every lost or gained mil becomes critical.

Unfortunately, the Protel DSN file will not contain your via-to-via clearance rule, so the via-to-via clearance must be added separately in either the command line or a DO file. The ELECTRA command is: `rule pcb (clearance 17 (type via_to_via))`

Copy your patterns to a library for future use. For those that insist on hand routing, I recommend hand routing every discrete component to its connecting pin and fanning pads out manually. The autorouter is capable of connecting discrete components. However, if you design like I do, you might be surprised how many components require tweaking by moving a few mils or 180-degree rotation. Anything gained by last minute optimization will help the router reach maximum completion.

When the fanout is completed, every used pad on every part should have a short connection to a via or discrete components should be routed to their respective pads. If a discrete component requires a connection to the opposite length side of the board, fanout and terminate with a via. This is the best technique to allow the autorouter to easily connect the longer traces.

One question frequently asked by designers is how to prevent vias out of specific areas. One method is to place lines and fills on any unused layers. Keep-out lines between thru hole pads will prevent the router from placing a via between connector pins. Use keep-out fills between discrete component pads to prevent

the router from placing vias between SMD pad. This will increase yields during reflow by preventing prevent short circuits between pads. Just make sure that this keep-out layer is OFF and not used for either routing or fabrication purposes.- If this layer is not disabled, the router will use it to place traces.

Translate electrical requirements into routing requirements by creating useful and meaningful instructions. One technique is to create and use net classes. Net classes can be included or excluded into routing commands. Assign layers in your layer stack for critical nets or use these layers for controlled impedance wiring.

Assign a net class to specific layers with specific instructions. When a DSN file is exported from PROTEL 2004, the class rules are flattened into individual net instructions. PROTEL 2004 will export the net class name but this action is not functional in PROTEL 99SE. Avoid assigning nets a "priority" number since this does little to instruct the router how to place and route a trace. Another approach is to identify all of the non-critical nets. LEDs, switches, low speed control, even signal requiring large clearances can be assigned to low priority layers in Protel. Typically on a 6-layer design, I might designate two internal layers for critical nets, and two for non-critical nets.

Controlled Autorouting

On larger designs, preparation can take several days to a week. Implementing a routing strategy complete with design rules requires time but much less time than manually routing.

There are four methods that can be used for controlled or interactive autorouting:

1. Design rules in Protel
2. Graphics for steering
3. DO files
4. Commands within ELECTRA

Take advantage of the design rules and graphics in Protel to set up your design file since these are the easiest to use. Use keepouts as guides for routing. Routing topologies can be improved on by using routing or steering channels. Use fills and other objects to prevent vias in specified areas. Use design rules to assign specific via sizes to certain nets. Use net classes to set up clearances and trace widths. The design rules will also allow specific routing layers.

There are several rules in addition to via-to-via clearance that do not export well to DSN. These are matched lengths, differential pairs and parallel rules. Most connections will require one layer pair, which will require one via. A good rule is to use max_via to control the maximum vias allowed by net. The finished results will always be cleaner. For example if a connector has line of sight to a component, and does not require any additional vias, set up design rule for a net class to contain zero vias.

Always make a back up copy of your PCB before importing and exporting to the router. To create a DSN file in DXP/2004, SAVE AS, change the extension to DSN, select all of the export options. Make sure "Protect Preroute" is checked. Do not use the DO file created by PROTEL 2004. Most of your design rules are contained within the PCB file. A relatively small "DO" file can be created using a text editor like WordPad. Another good practice is to review the DSN file in WordPad before loading in the router. Since the PCB file contains most of the rules, my DO files are fairly simple. ELECTRA does an excellent job of protecting nets. Nets are protected by Protel or the command line can be used.

Normally I use the following DO file for routing:

```
LIMIT WAY 250
rule pcb (clearance 17 (type via_via)
SELECT ALL WIRES
UNSELECT CLASSSS ( )
ROUTE 5
FILTER 1
CLEAN 3
RECORNER DIAGONAL.
```

Restricting the routers direction with LIMIT WAY, along with graphic steering and layer assignment will force the router into consistent routing topologies. Consistent topologies yield efficient routes. As a rule, I try routing 97 percent in 5 passes. Import the RTE file from ELECTRA back to Protel. If a trace is not complete, review the routing channels. Estimate how many routed lines are required for your address and data busses, then compare that to how many routing channels are open. If more channels are needed then consider adding layers.

In low-volume designs, adding layers is a cheap compromise. Routes can be reviewed using several methods. A length report is available in both PROTEL 99SE and 2004, or you can toggle through each net using the PCB panel. This allows you to quickly view the topology of each net. A properly routed net should run horizontal and vertical without large loops, or unnecessary stubs and

vias. If nets are consistently straying from a short horizontal/vertical routing path, additional layers might be required. After importing a RTE file from ELECTRA, all the vias will contain 28 mil holes. Globally select these vias and change them to their original hole sizes. This is normal behavior seen in several design systems including PADS, Protel and Orcad. The via-to-via design rule should be enabled to locate overlapping vias. Duplicate or overlapping vias can be found on pre-fanned components like BGAs. Globally select these vias from the PCB file, then delete them.

Several passes at the router may be required to get a desired result. Monitor the routing progress. ELECTRA is fast. If the router is stalling out and not making progress, save, then import the RTE file into Protel for review. Slow performance can only be caused by poor layout, insufficient routing layers, conflicts in design rules, or closed routing channels. If these steps are followed, very little or no post clean up is required.

Pitfalls to Avoid

On rare occasion, a DSN file may not load. All of these problems can be repaired in the PCB file. First, insure the working space with templates, graphics, and text, included does not exceed 50 inches (127 cm) square.

This is a fairly large area that should reasonably accommodate almost any design. Insure that all component attributes including reference designators, and comments are also contained within this area. On occasion, a corrupted PCB database will place these attribute outside the Protel workspace. To correct this problem, Select ALL, then Copy everything into a clean PCB sheet, then reload the netlist. The PCB data must also lay in the lower left quadrant of the Protel workspace. The origin can be set either to the workspace or the PC Board outline. Do not use split planes that are not assigned to net.

There are ongoing debates in manufacturing over the advantages of using round pads versus square pads. Avoid using square pads wherever possible. When a square pad is not assigned to net, on occasion, ELECTRA might place a via over this pad, sometimes requiring minor clean up.

Conclusion

EDA software enhances designers' productivity and it allows them to do more in a shorter space of time, and with a better quality result.

The same is true of autorouters such as ELECTRA. They are a productivity tool that allow PCB designers to perform routing automatically. A proper methodology to implement the use of the autorouter into a PCB design implementation flow provides high productivity gain by shortening the design time and delivers repeatable results.

autoroute, 53
bestsave, 53
bus, 54
check, 54
circuit, 54
circuit_descriptor, 50, 51, 54
class_descriptor, 50
clean, 56
Clearance, 5
clearance_type, 52
cost, 56
cost_descriptor, 50
cost_descriptors, 56
cost_type, 52
Crossing, 5
direction, 58
fanout, 58
filter, 59
grid, 60

layer_rule_descriptor, 51
limit, 60
object_type, 52
order_type, 52
priority, 50, 55
protect, 53, 61, 67
recorner, 62, 63
route, 64
routes, 67
rule, 64
rule_descriptor, 24, 51, 64
select, 65
session, 67
status_file, 65
tax, 65
unprotect, 66, 67
unselect, 67
use_via, 50, 55
write, 67