ProGenesis Guide

11.20.07

Prolific, Inc. 39899 Balentine Drive Suite 380 Newark CA 94560

510/252-0490 www.prolificinc.com

Table of Contents

Notices

ProGenesis Guide

Revision for 2006.1 release.

Every effort has been made to supply complete and accurate information. However, Prolific, Inc. assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

This document is considered Prolific Confidential.

Usage

Throughout the ProGenesis Guide’s text, items which you will need to type directly into the command lines will appear in `Courier font.`

Variables you will need to input will appear in *`Courier italic.`*

Getting Started With ProGenesis

Install ProGenesis on your system

Download the gzip'd file corresponding to your operating system.

The examples below are for release version 4.4.36. Other versions have similar organization, but the version number is changed.

Operating System	File
HP-UX 11.00	20061_HPUX11.tar.gz
Linux RedHat 6.x, 7.x	20061_Linux6.tar.gz
Linux RedHat 9.x	20061_Linux9.tar.gz
Linux RedHat Enterprise 3, 4	20061_LinuxRE3.tar.gz
Solaris	20061_Solaris.tar.gz

Table 1

Please write to applications@prolificinc.com for upload instructions.

Unpacking

The commands you use to unpack the software archive files depend on the type of file you downloaded and your operating environment. Typical commands are:

```
gunzip  20061_Solaris.tar.gz

tar xvf 20061_Solaris.tar
```

The file will unpack into the directory "4.4.34" which is the suite release number for the Prolific tools. Typical installations place it under the "prolific" directory and then have a logical link to it:

File	Description
<install_path>/prolific/2006.1	specific release directory
<install_path>/prolific/prod	logical link to prolific/2006.1

Table 2

Of course, this configuration is completely up to you.

Multiple Operating Systems

To use multiple operating system releases just unpack each of the system specific archive files into the same directory tree.   For example, the following unpacks the Solaris, HPUX1100 and Linux6 trees:

```
   cd <install_path>/prolific

   tar xvf 20061_Solaris.tar

   tar xvf 20061_HPUX1100.tar

   tar xvf 20061_Linux6.tar
```

The Prolific tools determine which operating system they are running on and invoke the appropriate executable.

License Management - Existing Customer

Your existing license file will continue to work with the 4.4.36 release of the tools. Simply install your existing license file in the 4.4.36 tree as defined in the next section (License Management - New Customer).

License Management - New Customer

If you have not already sent your license server host-id to Prolific please do so as soon as possible. This information is needed so that we can create a license file. The FLEXlm license file will be emailed to the license manager and should be installed in the following directory with the name <install_path>/prolific/2006.1/lib/prolicense/prolific.lic.

This file may need to be edited depending on your site license management strategies. The FLEXlm license manager (lmgrd), vendor daemon (prolific), and license management utility program (lmutil) are in the OS-specific directory <install_path>/prolific/2006.1/bin/<os>.

Set-Up

To run any of the tools the environment variable PROLIFIC must be set to the software install directory. Using the above path in csh you would set:

```
    setenv PROLIFIC <install_path>/prolific/prod
```

Then add the $PROLIFIC/bin directory to you path. Again, in csh:

```
   set path=($path $PROLIFIC/bin)

   rehash
```

Configuring the environment is easy: the PROLIFIC environment variable must point to the directory where the software has been installed.

For example, if the software were installed in /usr/local/prolific/4.4.36, then you could use the following command to set the environment using the Bourne shell:

```
sh% PROLIFIC=/usr/local/prolific/2006.1

sh% export PROLIFIC
```

For non-Bourne shells, use the syntax appropriate to the shell. For example: using csh, set the PROLIFIC environment variable this way:

```
csh% setenv PROLIFIC /usr/local/prolific/2006.1
```

To check the environment variable, use the following command:

```
csh% echo $PROLIFIC
```

From now on, we'll refer to the PROLIFIC installation directory as $PROLIFIC. All of the tools reside in the $PROLIFIC/bin directory, so an easy way to run them is to add the $PROLIFIC/bin directory to the path. It is most convenient to do this in the shell configuration files. For example: in the csh configuration file, .cshrc, the following command can be added:

```
   set path = (           \

       $PROLIFIC/bin   \

       $path           \

       )
```

Note that if you have multiple architecture binaries installed, the Prolific tools will automatically execute the appropriate version for your machine.

ProGenesis Tool Overview

The basic ProGenesis tool flow is shown in Figure 1, below. ProGenesis is the suite of tools that generates standard cell layout from Spice input. ProGenesis includes three main tools used to create layout from netlists: ProTech, ProSpin, and ProGen. Although other tools are included in the ProGenesis suite, these are the three we'll focus on to get started.

ProTech

ProTech is the tool used to configure fabrication technology specifications, design styles, and layer information. ProTech is used to enter or modify the following types of information:

- Design Options: Transistor layout, compaction and routing options
- Design Rules: Mandatory and preferred rules specified by the fab
- Layer Data: Specify GDS layer data and text
- Cell Template: Cell height, well tie behavior, and rail and well sizes.
- Router Configuration: Number of routing tracks, layers, and costs.

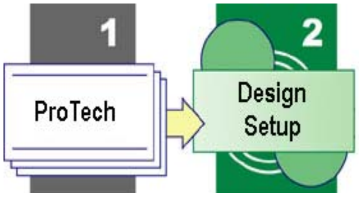


Figure 1: ProTech Flow

ProSpin

ProSpin reads Spice netlists describing individual cells and produces corresponding CEL files to be read by the ProGen tool. A CEL file specifies the generators that should be called to create the final layout data and contains cell-specific information such as transistor sizes and node names.

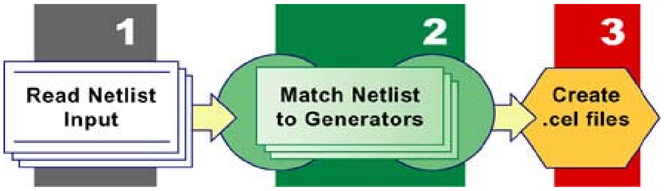


Figure 2: ProSpin Flow

ProGen

ProGen reads in CEL files and produces physical layout based on the cell characteristics, the generators specified, and the technology-specific information created using ProTech. After reading the CEL file, ProGen invokes the proper generators and performs transistor-level synthesis to produce a loose physical layout. ProGen then compacts this initial layout to produce a final cell that is as small as possible while conforming to all design rules and layout constraints.

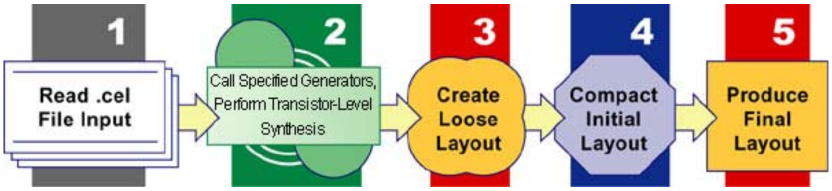


Figure 3: ProGen Flow

Demonstration Data

The software bundle contains a demonstration area which contains everything you need to run the ProGenesis tools. All of the examples in this section use data from the demonstration area $PROLIFIC/demo. The demo area contains the following types of data:

- Demonstration CMOS technology
- Cell Spice netlists
- Sample layout AGD (ASCII GDS, the output from ProGen) data
- Sample ProSticks layout, generators, cells, and output AGD data

Generating Cell Layout From a Netlist

This section describes how to generate an inverter cell from a Spice netlist. This process involves these steps:

1. Create or move to a working directory

2. Run ProSpin to convert the Spice netlist to a CEL file

3. Run ProGen to generate the cell and compact it

4. View AGD data

5. Convert between AGD and GDS

Step 1: Move to a working directory

The ProGenesis flow results in many files being created, so it is wise to run the tools from an appropriate working directory. We suggest creating a parent directory for each library project, with subdirectories for netlist input, CEL files, and process technology information. See "Prepare to Run Tutorial Data" for details.

Step 2: Use ProSpin to convert the netlist to a CEL file

In this step, we run ProSpin from the command line to create a cell description file named inv.cel from a Spice netlist named inv.sp. Use the following command:

```
csh% prospin -l inv $PROLIFIC/demo/spice/inv.sp
```

This should produce the inv.cel file in your working directory. ProSpin automatically creates CEL files to match each subcircuit name requested. For more information, consult the ProSpin Reference Guide.

Step 3: Use ProGen to generate and compact the cell

Now that there's a CEL file corresponding to the inverter cell, we run ProGen to create the cell layout:

```
csh% progen -t $PROLIFIC/demo/tech/demo.db -i inv.cel -o inv.agd
```

This command produces the inv.agd cell file, which contains the layout in AGD format. For more information on ProGen, consult the ProGen Reference Guide.

Step 4: Viewing AGD data

While this step isn't necessary for generating cell layout, it is useful for viewing the layout created in the first three steps. The ProGenesis tools produce AGD (ASCII-GDS) cell data. AGD is an ASCII representation of the standard GDSII data file format. PROLIFIC provides an AGD data file viewer called ProView.

```
csh% proview -t $PROLIFIC/demo/tech/demo.db inv.agd
```

This starts the ProView viewer and loads the AGD file inv.agd. Further information on options available in ProView is available in the ProView Reference Guide.

Step 5: Converting between AGD and GDS

Moving the data from ASCII GDS to a binary GDS file is easy. PROLIFIC provides two easy-to-use utilities for converting layout between AGD and GDS formats. To convert AGD data to GDS, use a2gds:

```
csh% a2gds < inv.agd > inv.gds
```

To convert GDS data to AGD, use gds2a:

```
csh% gds2a < inv.gds > inv.agd
```

GDS output can be generated directly from progen by specifying:

```
_type agds
```

ProGenesis Tutorial Overview

The ProGenesis suite allows designers to quickly and accurately create optimized standard cells. ProGenesis includes the tools shown below.

Tool	Description
ProTech	Design rule and configuration tool
ProSpin	Netlist analysis tool
ProGen	Cell creation and compaction tool
ProSticks	Topology editing and creation tool

Table 1

ProGenesis Flow

ProGenesis can create and manage any number of cells, and can handle cells as part of a library or individually. ProGenesis requires the inputs shown below to create GDS-II output:

Input	Description
Netlist	Single, multiple, or hierarchical HSpice netlist
Design rules	From design rule document
Cell template definition	User-defined cell architecture

Table 2

The cell creation process follows these steps:

1. Define setup information in ProTech design rules and cell template definitions.

Input	Enter design rules and cell architecture settings
Output	A database (.db) file containing design rules, cell template, and library option data

Table 3

2. Use ProSpin to analyze netlists and match their contents to cell generators.

Input	Spice netlists
Output	.cel files containing the generator calls corresponding to each cell

Table 4

3. Run ProGen to create and compact cells.

Input	Output from the previous steps (.cel files and .db file)

Table 5

Output	GDS-II layout

Table 5

ProGenesis Approach

ProGenesis recognizes the basic hierarchical building blocks in a cell, such as NAND, AOI, and TRI-INV. These blocks are represented within the ProGenesis flow as "generators" which can include descriptions of transistor placement, routing configurations, and other layout details.

In general, more layout information contained in a generator results in more predictable GDS-II output. However, it is also possible to include too much information in a generator, which can overconstrain the cell creation process, ruling out desirable solutions. Walk through the ProGenesis Tutorial Guide to see how generators are used.

ProGenesis Tutorial Guide

In the first part of this tutorial, the use of the products is described. In the second part of the tutorial is data that can be used to practice walking through the tools. Sample tutorial data should be included with the ProGenesis tools; the data are also available in the section called "Tutorial Data."

Prepare To Run Tutorial Data

Ensure that the environment and tools are ready:

- Create a working directory you can use without worrying:

```
% mkdir temp
```

- Create a subdirectory structure for data and output:

```
% cd temp

% mkdir agd cel net tech
```

- Check that the environment variable is set correctly:

```
%  echo $PROLIFIC
```

Setting the Environment Variable

If the PROLIFIC environment variable is not set, or points to the wrong directory, follow the setup instructions in the Getting Started guide.

- Check that ProGen runs properly:

```
%  progen -h
```

- Make a local work directory and copy the tutorial data there:

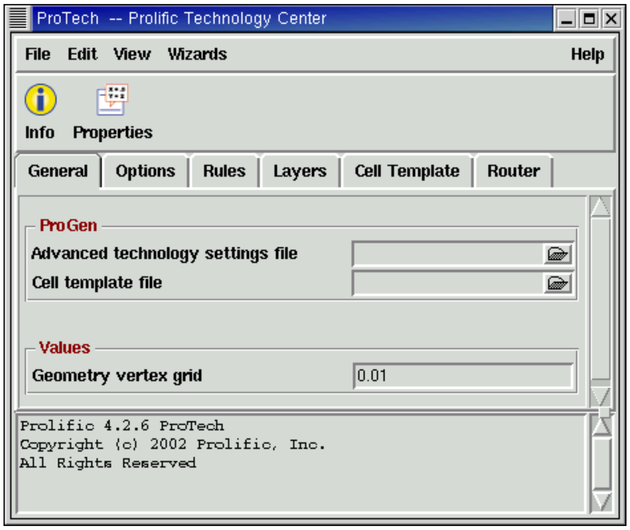
```
%  cp -r $PROLIFIC/tutorial work
```

Starting ProTech

Start ProTech from the command line:

```
% protech
```

The ProTech graphical interface will appear on the screen (see Figure 1). Its tabbed region identifies the different categories of configuration information (Table 1):



ProGen

Advanced technology settings file

Cell template file

Values

Geometry vertex grid: 0.01

```
Prolific 4.2.6 ProTech
Copyright (c) 2002 Prolific, Inc.
All Rights Reserved
```

Figure 1: ProTech Interface

Tab	Description
General	Set manufacturing vertex grid. Set path to advanced settings file.
Options	Configure library-wide options and layout style preferences.
Rules	Enter design rules.
Layer	Define layer-to-GDS mappings.
Cell Template	Select cell template style.
Router	Calibrates router for cell template definition.

Table 1: ProTech tabs

Entering Setup Data

The first time you use ProTech, follow these steps:

Click on the General tab. In the Geometry vertex grid field, enter the manufacturing grid size in microns. For this tutorial, enter 0.005.

You can skip the other General settings and the Options settings. When you initially create cells, you do not have to alter these settings, but on subsequent iterations they may help you to tailor the layout to your needs and preferences.

Click on the Rules tab to enter design rules from a design rule document. For this tutorial, enter the values shown in Table 4. If you want to know more about a particular design rule entry, click on the Info button, then click on the name of any rule for a description and illustration of that rule.

Rule Name	Value to Enter
Min n-well width	0.62
n-well overlap N+ diffusion tie	0.17
n-well overlap P+ diffusion	0.22
Minimum P+ implant width	0.24
P+ implant space to N+ diffusion	0.13
P+ implant space to N+ diffusion gate	0.22
P+ implant space to N+ diffusion tie	0.02
P+ implant overlap P+ diffusion	0.13
P+ implant overlap P+ diffusion gate	0.22
P+ implant overlap P+ diffusion tie	0.02
Minimum diffusion width	0.11
Minimum diffusion spacing	0.14
Diffusion overlap of contact	0.05
Diffusion extension from gate	0.16
Minimum width butted diffusions	0.11
N+ diffusion space to n-well	0.22
Minimum poly width	0.1
Minimum poly spacing	0.14
Minimum poly overlap of contact	0.03 0.05 0.03 0.05
Minimum poly space to diffusion	0.05
Poly-gate endcap extension	0.16
Minimum transistor gate width	0.12
Minimum poly-to-poly gate spacing	0.15
Minimum transistor gate length	0.1
Minimum contact width	0.12
Minimum contact spacing	0.14
Minimum diffusion-contact to gate spacing	0.08
Minimum poly-contact to gate spacing	0.1
Minimum Metal-1 width	0.12
Minimum Metal-1 spacing	0.12
Metal-1 overlap diffusion-contact	0.0 0.05 0.0 0.05
Metal-1 overlap poly-contact	0.0 0.025 0.0 0.025
Metal-1 overlap Via-1	0.005 0.05 0.005 0.05
Metal-1 minimum area	0.059
Minimum Via-1 width	0.13

Table 2: Tutorial Design Rule Settings

Rule Name	Value to Enter
Minimum Via-1 spacing	0.15
Minimum Metal-2 width	0.14
Minimum Metal-2 spacing	0.14
Metal-2 overlap Via-1	0.005 0.05 0.005 0.05
Metal-2 minimum area	0.07

Table 2: Tutorial Design Rule Settings

Now click on the Layers tab, where there are two regions containing data used to generate mappings between GDS-II output numbers and user-defined layer names, and between the user-defined layers and ProGenesis' internal layers.

Layers Section	Description
ProGen Layers	Match the layers from the pulldown menus to ProGen's system layers listed in the left-hand column. The layers in the pulldown menus are defined in the ProView Layers section.
ProView Layers	Assign or alter output GDS numbers and layer colors as needed. Be sure each layer is assigned to a unique GDS number.

Table 3: Layers Descriptions

For the tutorial, set the layers as shown in Tables 4 and 5. Because the ProGen layers are set to match the ProView layers, start by configuring the ProView layers.

Start by adding ProView layers that are missing. Click on the Add ProView Layer button, enter the name of on layer to be added, and click OK. The tutorial layers that need to be added are diffusion, via 2, metal 3.

Set the GDS layer for each of the ProView layers that remain. Simply click in the GDS Layer field and set the layer number as necessary. For the tutorial, refer to the layer numbers in Table 5.

Set the display properties for each layer:

Color/Pattern

Show Fill Pattern?

Show Outline?

Show Layer?

Click on the color bar to alter the display color and hash pattern for each layer, and using the check boxes under Display Fill and Display Outline, set each layer to be displayed with a fill pattern, an outline, or both. Add or remove layers from the display (but not the GDS-II output) by toggling the Display View field. For the tutorial, match the fill and display settings shown in Table 6.

If you wish, alter the order of the ProView layers by clicking on the Change ProView Layer Order button. The current layer list can be sorted as you like by clicking on a layer name and dragging it to a new location.

Once the ProView layers are configured, set the ProGen layers.

Start by matching the ProGen layers to the ProView layers. Click on the drop-down list under ProView Data Layer to set the ProView layer for each ProGen layer. For the tutorial, set n+ diffusion and p+ diffusion to diffusion.

Set the ProView text layer for each of the ProGen layers by clicking on the drop-down list under ProView Text Layer and selecting the desired layer name. For the tutorial, the only change you need to make is to set metal-1 to use metal-1 and set metal-2 to use metal-2. As an added exercise, you can add the via-2 and metal-3 layers that you defined earlier, but it isn't required for the tutorial because we don't use v2/m3.

To exclude a layer from being output, deselect its check box in the Output Layer column. This is used when you have different text layers, otherwise the default text mapping should suffice.

ProGen Layer	ProView Data Layer	ProView Text Layer	Output Layer?
cell border	cell border	text	✓
n-well	n-well	text	✓
n+ implant	n+ implant	text	✓
p+ implant	p+ implant	text	✓
n+ deep implant			
p+ deep implant			
n+ diffusion	diffusion	text	✓
p+ diffusion	diffusion	text	✓
poly	poly	text	✓
contact	contact	text	✓
metal-1	metal-1	metal-1	✓

Table 4: Tutorial ProView Layer Settings

ProGen Layer	ProView Data Layer	ProView Text Layer	Output Layer?
via 1	via 1	text	✓
metal-2	metal-2	metal-2	✓
critical path	critical path	text	✓

Table 4: Tutorial ProView Layer Settings

ProView Layer	GDS Layer	Display Fill?	Display Outline?
cell border	235		✓
n-well	3		✓
diffusion	6	✓	
n+ implant	26		✓
p+ implant	25		✓
poly	17	✓	
contact	30		✓
metal-1	31	✓	
via 1	51	✓	✓

. Table 5: Tutorial ProView Layer Settings

ProView Layer	GDS Layer	Display Fill?	Display Outline?
metal-2	32	✓	
via 2	52	✓	✓
metal-3	33	✓	✓
text	900		✓
critical path	999		✓

. Table 5: Tutorial ProView Layer Settings

The layer setup process allows you to assign more than one internal ProGen layer to the same GDS number. For example: there is more than one ProGen layer for diffusion. But if the process has only one layer for diffusion, first define that layer in the ProView Layers section. Then in the ProGen Layers section, choose that same layer from the pulldown menu for each of the ProGen diffusion layers.

Click on the Cell Template tab. Set the characteristics that apply to all cells here, such as cell height, routing grid, n-well position, and power rail dimensions.

Click on the Router tab, which is used to restrict the router to use routing that will fit the cell height. Click on the Run parameter generation script button only after completing the configuration on the Rules and Cell Template tabs.

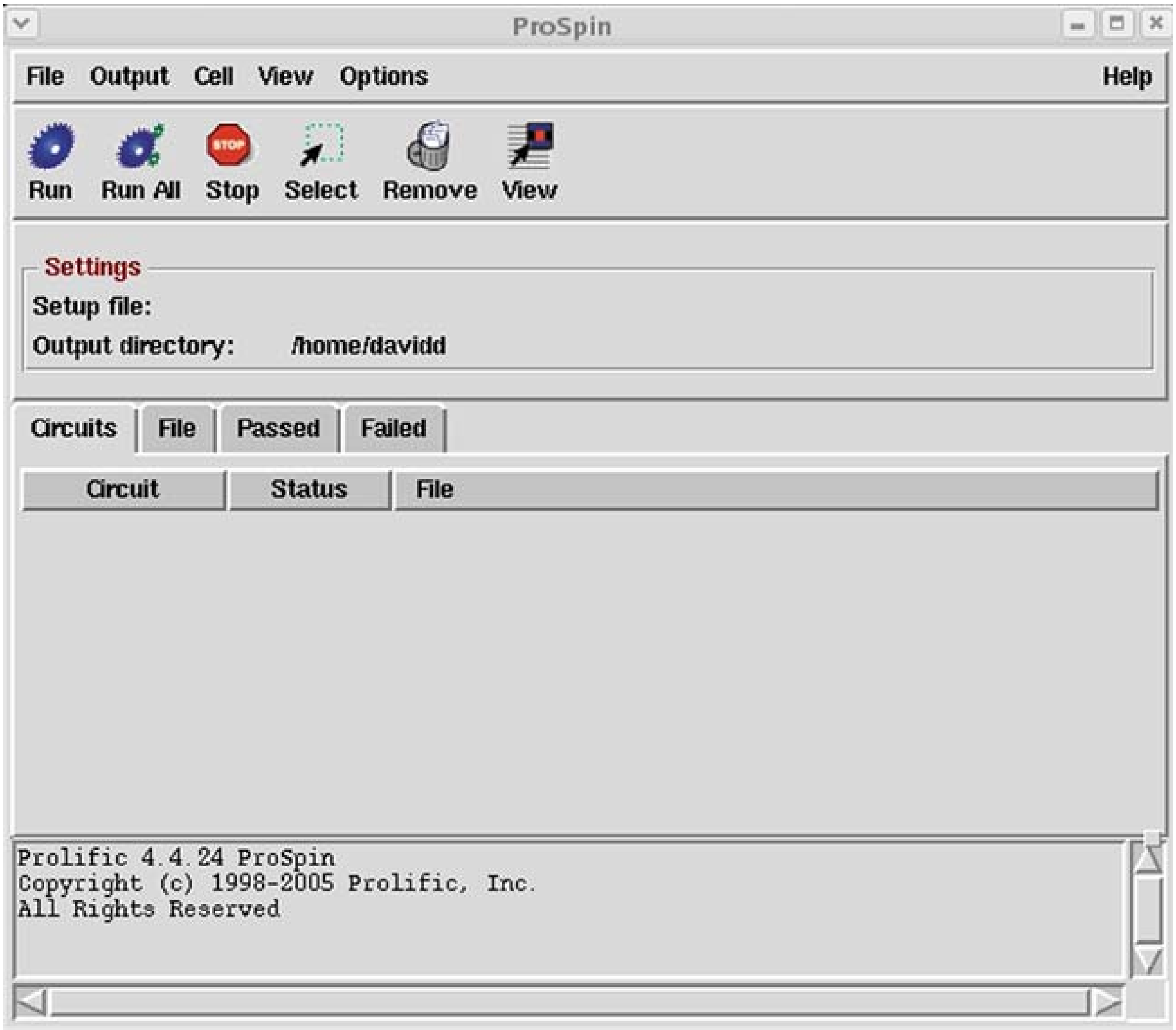
Save the configuration using the File->Save As ... menu item.

Starting ProSpin

Start ProSpin from the command line:

```
    % prospin -g
```

A graphical interface like the one shown in Figure 2 will appear on the screen.



Setup file:

Output directory: /home/davidd

```
Prolific 4.4.24 ProSpin
Copyright (c) 1998-2005 Prolific, Inc.
All Rights Reserved
```

Figure 2: ProSpin Interface

Running ProSpin

ProSpin identifies the structures inside HSpice netlists and maps them to appropriate generators. Because netlists have many different styles, consider the following when starting to use ProSpin:

Are transistor lengths and widths scaled properly? See Scaling Transistors for more details.

Are ProSticks generators referenced? See Referencing ProSticks Generators for more details.

Are names correctly specified to match power and ground nodes? See Specifying Power and Ground Node Names for more details.

Are ProPPR route settings correct? See ProPPR Router Settings for more details.

Scaling Transistors

If the transistors need to be scaled by width or gate length, e.g., when porting netlists from an older technology, place the following commands in prospin.tcl to get linear scaling:

```
prospin_set option width_ratio_pfet x

prospin_set option width_ratio_nfet x

prospin_set option length_ratio_pfet x

prospin_set option length_ratio_nfet x
```

Set x equal to the scaling value:

```
widthdesired=x*widthnetlist
```

Setting these options allows the use of different scaling factors for p- and n-transistors.

Referencing ProSticks Generators

ProSticks is a layout editor that can produce generators. To use one of these generators in place of the default system generator, add these lines to prospin.tcl:

```
prospin_source generator.tcl

prospin_set option search_list {cellname} [list \

[generatorproc_pattern_map]

]
```

Use the following values:

generator.tcl

The name of the Tcl file containing the generator.

cellname

The cell name, which matches the subcircuit name from the netlist.

generatorproc

The procedure name of the ProSticks generator.

In most cases, all three values can be the same - use the cellname for all three variables. If you want to use a ProSticks generator as a subgenerator rather than a generator for the entire cell, add the following line to the end of the list:

```
    [prospin_library basic default_list]
```

To use same generator for cells with different drive strengths (e.g., to apply generator "dff.tcl" with "dff_pattern_map" as its generatorproc_pattern_map to cells dffx1, dffx2, and dffx4), do the following:

```
    prospin_source dff.tcl
    prospin_set option search_list "^dffx1$|^dffx2$|^dffx4$" [list \
        [dff_pattern_map] \
    ]
```

Specifying Power and Ground Node Names

If your netlist has a name for the power and ground rails that ProSpin doesn't recognize, use the following code in prospin.tcl:

```
    prospin_set node_match ground pattern {

      prospin_set node_match power pattern

    }
```

This addition will perform a regular expression match for the node.

ProPPR Router Settings

To enable the ProPPR router, enable it in ProSpin by adding the following to prospin.tcl:

```
    prospin_set generator options "^cellname$" {

    {proppr_route 1}

    }
```

In addition to enabling the router, these are the additional commands that can control the behavior of the ProPPR router.

Setting expected and acceptable costs

Set these when the router runs and finds a solution, but does not finish. Expected cost sets an upper bound on the router's solution: it will not select a solution with a cost greater than the expected cost. Acceptable cost is the cost at which ProPPR will stop looking for solutions; the default is an exhaustive search, so by default, the acceptable cost is set to 0 (zero).

```
prospin_set progen options  "^cellname$" {

pro_set global proppr_acceptable_cost  cost

pro_set global proppr_expected_cost   cost

}
```

Using joggedm1 and joggedpoly

For cases where the cell misses the cell height, or where outer channel m1 routes affect diffusion contact strapping, it may help to use joggedm1 and joggedpoly.

```
prospin_set progen options "^cellname$" {

pro_set global proppr_horz_layernames {poly joggedm1}

}
```

Using ProSpin

Follow these steps to run ProSpin:

Source the technology setup file, if applicable, and set output directory where .cel files will be placed. Use the Options->Options... menu item. For the tutorial, set the

Load the HSpice netlists using the File->Load Spice ... menu item.

Select the netlist file or files to use, and click OK. The subcircuit names will appear in the main window along with the directory path of the netlist file being sourced.

Run ProSpin to map netlist functions to generators. To run ProSpin on all cells, click Run All. To run ProSpin on a subset of the imported cells, and click Run.

After creating the .cel files, select one of the cells and click View to examine the content of its .cel file. The .cel file contains calls to the generators ProGen will use. Each generator call is denoted by flag_call. While generator interfaces differ somewhat, the calls include: transistor widths, inputs, outputs, transistor names, connectivity.

View the contents of the .cel file to verify that a user-specified generator call (e.g., a generator created in ProSticks) is actually being called for the cell.

Starting ProGen

ProGen uses the technology database file and the .cel file to create and compact the cell layout. When ProGen runs on a cell, it first creates a fully-connected but loose topology, then compacts the cell according to the design rules and options in the database file.

ProGen runs from the command line:

```
%  progen -t techfile -i

celfile -o outputfile >&

logfile
```

Evaluating ProGen Output

Once ProGen has created the output file, review the results using ProView. Use the following command to open all the output files in the current directory:

```
%  proview -t techfile *.agd
```

Verify that the options that were set are reflected in the layout, and that the density meets expectations. If a cell appears wider than expected, it is possible to alter the compaction order to reduce the cell's size, or to analyze the critical path to better understand the situation.

Tutorial Data

The information contained in this section can be used to run the sample cells without the hand-holding of the ProGenesis Tutorial Guide. You can make sure you understand the process of creating a library using these few cells:

AND2

LATCH

MUX2

XOR2

Following the flow described in ProGenesis Flow, create the layout:

ProTech

ProSpin

ProGen

Copy Tutorial Data

Copy the tutorial directory to a local area:

```
%  cp -r $PROLIFIC/tutorial

foodir
```

Configure Design Rules and Cell Template

The tutorial area's directory structure is shown below:

```
tutorial

    |-- agd        AGD output from ProGen

    |-- cel       .cel files for use by

ProGen

    |-- net       HSpice netlists

    `-- tech       Technology database file and special option files
```

The design rule document for the tutorial is shown in Tutorial Design Rules, below.

Tutorial Design Rules

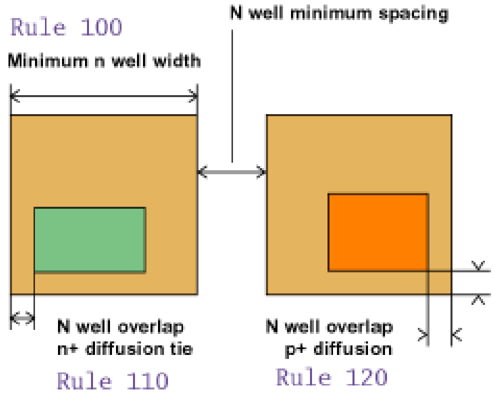


Figure 3: Well Rules

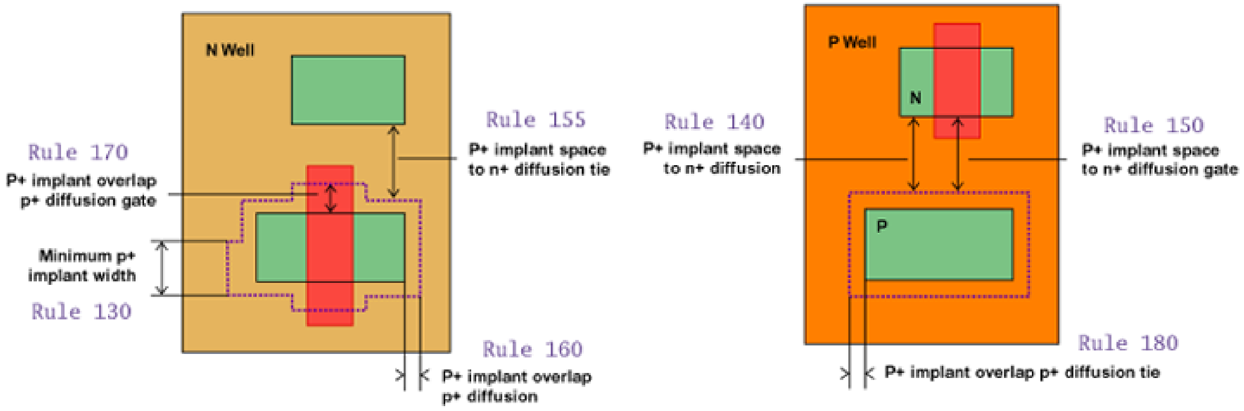


Figure 4: Implant Rules

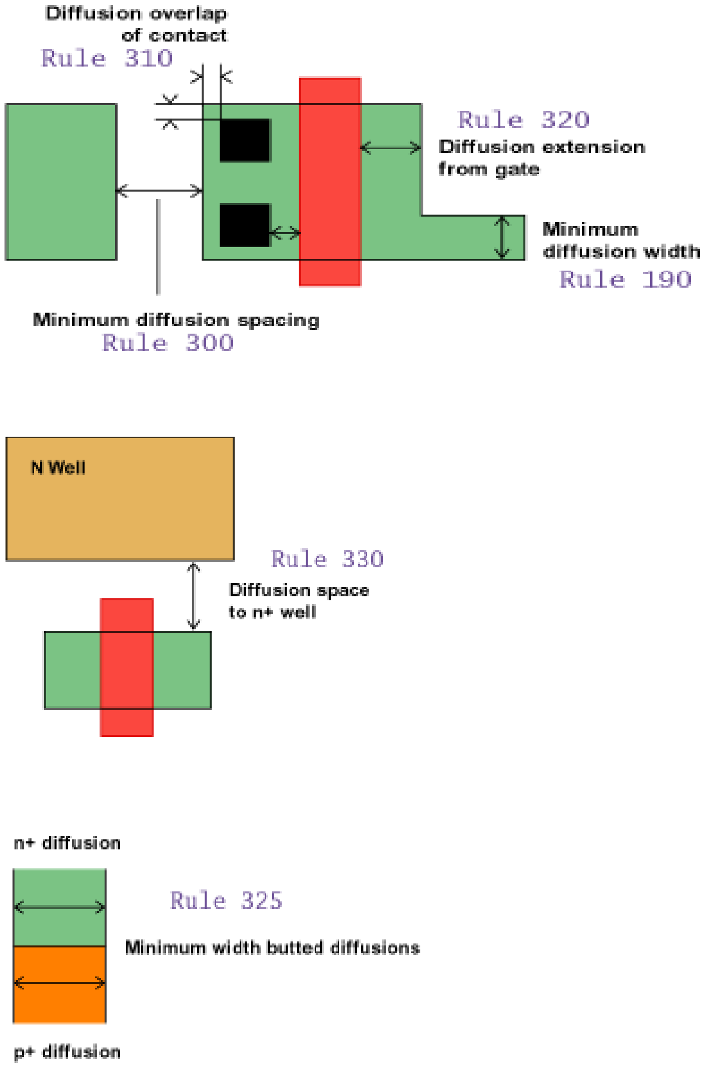


Figure 5: Diffusion Rules

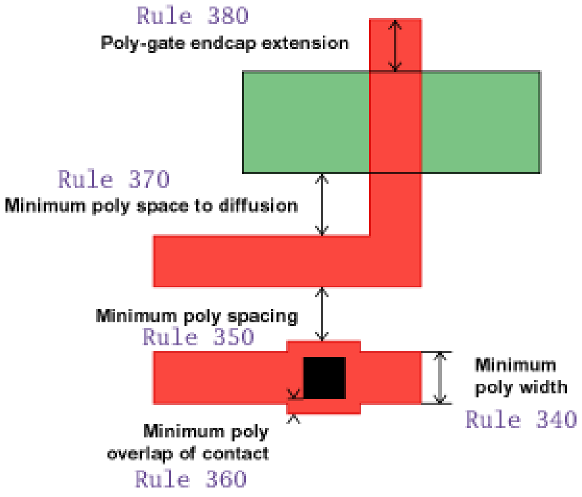


Figure 6: Poly Rules

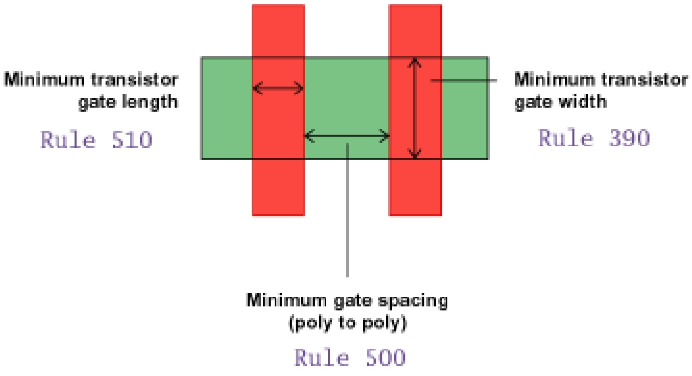


Figure 7: Gate Rules

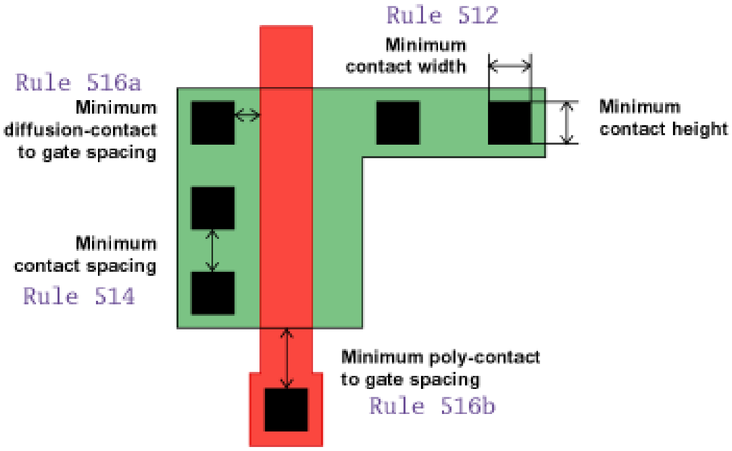


Figure 8: Contact Rules

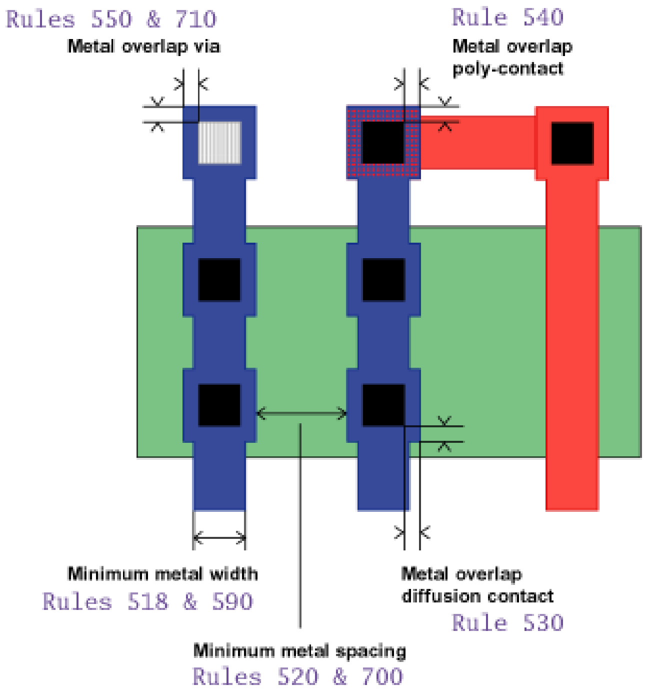


Figure 9: Metal Rules

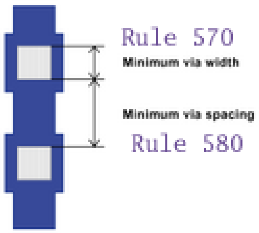


Figure 10: Via-1 Rules

Rule	Description	Value
100	Min n-well width	0.62
110	n-well overlap n+diffusion tie	0.17
120	n-well overlap p+diffusion	0.22
130	Minimum p+implant width	0.24
140	p+implant space to n+diffusion	0.13
150	p+implant space to n+diffusion gate	0.22
155	p+implant space to n+diffusion tie	0.02
160	p+implant overlap p+diffusion	0.13
170	p+implant overlap p+diffusion gate	0.22

Table 6

Rule	Description	Value
180	p+implant overlap p+diffusion tie	0.02
190	Minimum diffusion width	0.11
300	Minimum diffusion spacing	0.14
310	Diffusion overlap of contact	0.05
320	Diffusion extension from gate	0.16
325	Minimum width butted diffusions	0.11
330	n+diffusion space to n-well	0.22
340	Minimum poly width	0.1
350	Minimum poly spcing	0.14
360	Minimum poly overlap of contact	0.03 0.05 0.03 0.05
370	Minimum poly space to diffusion	0.05
380	Poly-gate endcap extension	0.16
390	Minimum transistor gate width	0.12
500	Minimum poly-to-poly gate spacing	0.15
510	Minimum transistor gate length	0.1
512	Minimum contact width	0.12
514	Minimum contact spacing	0.14
516a	Minimum diffusion-contact to gate spacing	0.08
516b	Minimum poly-contact to gate spacing	0.
518	Minimum Metal-1 width	0.12
520	Minimum Metal-1 spacing	0.12
530	Metal-1 overlap diffusion contact	0.0 0.05 0.0 0.05
540	Metal-1 overlap poly-contact	0.0 0.025 0.0 0.025
550	Metal-1 overlap Via-1	0.005 0.05 0.005 0.05
560	Metal-1 minimum area	0.059
570	Minimum Via-1 width	0.13
580	Minimum Via-1 spacing	0.15
590	Minimum Metal-2 width	0.14
700	Minimum Metal-2 spacing	0.14
710	Metal-2 overlap Via-1	0.005 0.05 0.005 0.05
720	Metal-2 minimum area	0.07

Table 6

From the tech directory, follow the instructions in Entering Setup Data to complete the rules and layers sections using the information in Tutorial Design Rules. For the data on the Cell Template page, enter the information in Tutorial Cell Template.

Tutorial Cell Template

Option	Value
Cell Template	basic
Ties	straddle
Cell Height	11 grids
Xgrid	0.28
Ygrid	0.28
Routing Offset	xy-half

Table 7

Save the file as progen.db under the tech directory.

Input Netlist

Following the instructions from Using ProSpin, configure ProSpin to use the parameters shown in Tutorial ProSpin Configuration. Then use ProSpin to process the each of the cells in the netlist.

Tutorial ProSpin Configuration

Option	Value
Output Directory	cel
HSPICE Netlists	net/all_netlists.net
ProSpin Technology File	tech/prospin.tcl

Table 8

Create Layout

Once the .cel files are created, follow the steps outlined in Starting ProGen to create the physical layout for the cells. Use the command line values gathered from Tutorial ProGen Configuration, substituting the name of each cell where cell appears.

These values will create AGD and log files in the agd directory. View the layout as described in Evaluating ProGen Output, using progen.db as the technology file.

Tutorial ProGen Configuration

File	Path
techfile	tech/progen.db
celfile	cel/cell.cel
agdfile	agd/cell.agd
logfile	agd/cell.log

Table 9

Library Construction Guide

This document describes the steps that should be taken to build the layouts for an entire library.

How to Build a Library: Process Overview

1.Gather Data

2.Specify Technology Settings

3.Create the cell template.

4.Configure Post-Processing Scripts

5.Run an Inverter

6.Run All Cells

7.Analyze/Adjust Combinational Gates

8.Analyze/Adjust Mux/XOR/Adder Gates

9.Analyze/Adjust Latch/Flop Gates

10.Run Experiments

11.Summarize All Results

Gather Data

Gather the data necessary for determining the how the ProGenesis tools should be set up. The required data includes:

- Design-rules (temporary ones will do; ProGenesis easily handles changes)
- Cell-template description
- Cell netlists
- Layouts/schematics (optional)

Specify Technology Settings

Use ProTech to enter the process design rules, which will map design rules to ProGenesis primitives. The easiest approach is to start with a similar technology file and make any changes necessary. If you reuse another technology file, be sure to run an inverter through it (see Run an Inverter) to ensure that the technology file that you copied is functional before making changes.

After entering the design rules, enter the GDS layer numbers in ProTech. Also include the rules to the edge of the cell and the layer numbers for text on each layer. Don't forget the critpath layer (usually GDS layer 999) for viewing the critical path when using fast-compact.

Create the Cell Template

Use ProTech to set up the cell template according to your specifications.

Set the design-style options according to the performance, area, power and yield requirements. (All these options should be documented for option-name, allowed values, example layouts for each value.)

Key values to be sure to enter in ProTech include:

- Set the manufacturing grid
- Set the transistor width grid (Generally, this should be twice the manufacturing grid)
- Set the compaction order

Configure Post-Processing Scripts

This step is optional. If setting up a post-processing script is needed, refer to "post-processing" document for details.

Run an Inverter

Run a simple inverter to verify that the technology files are properly set up. The resulting layout should be exactly as desired.

The first step is to run ProSpin to create a .cel file for the inverter. Assuming the cell is called invx1, run the following command:

```
% prospin -t mytech/prospin.tcl -d cel -l invx1 net/invx1.sp
```

which will create cel/invx1.cel

Check the resulting .cel file to make sure that the transistor scaling is correct. Prospin's native unit is microns. Reasonable numbers would be in the range of 100 for a large transitor width down to .07 for the gate length of a cutting edge process.

If the values in the resulting .cel files are extremely large numbers, the Spice decks probably do not have the appropriate scaling numbers embedded in them. Set the following in prospin.tcl to scale netlist values by 10e-6, for example:

```
# corresponding statement in spice netlist:

# option scale=0.000001

prospin_set Spice scale 0.000001
```

If the transistor widths are very small (such as 6e-07), you might be scaling twice (once in the netlist, once in prospin). Remove that statement from the prospin.tcl file and re-run prospin.

Decide whether to use the folding numbers from the Spice netlist, or to let the generators calculate the folding by changing prospin.tcl as follows:

```
prospin_set option use_default_folding 1
```

If the folding was explicitly set in the netlist (i.e., multiple transistor definitions with the same node connections), setting the above option will recombine all of these transistors and progen will do the folding according to the folding settings.

Next, attempt to run the inverter using skipcompact with the following command:

```
%

progen -t mytech/progen.db -i cel/invx1.cel \

-o agd/invx1.agd.skip -d skipcompact >&! log/invx1.log.skip
```

Review the layout using proview:

```
proview -t mytech/progen.db agd/invx1.agd.skip
```

The layout will appear very large -- probably ten times the cellheight in x and y. This is normal and is what the layout looks like before being compacted.

At this point you might want to perform some sanity checks. It would be useful to use the zoom features provided by proview.

Verify that the pre-compacted layout looks reasonable. Some things to check are:

- The n-well around the p-transistors is fully enclosed by the pre-compacted n-well of the cell-template. If it is not already on, you can turn on the fill for the n-well layer to check this.
- The n-transistors are far away from the n-well. With the n-well fill turned on make sure the n-transistors are outside of the n-well and have a healthy margin to the n-well bottom edge.
- The power-rails are the correct width. If you are using a template defined in protech the rails should be the same width that you expect them to be after compaction.
- The well and implant overlaps are correct relative to the cellborder and well. Turn on the fill for the implant layers one at a time an make sure p-implant is over the p transistors and n-implant is over the n-transistors.
- The ties (if in the celltemplate) are being placed. If these are in the celltemplate they will probably be in the corners or top and bottom edges.
- There are no opens or shorts, particularly to the power and ground rails.
- The connections to the power rails are either a diffusion contact connecting right to the transistor, or a dmwire which does a partial diffusion route. This consists of a diffusion contact outside the boundary of the transistors, and the rest is a metal route.

Run All Cells

Once the technology files are properly set up, run the entire library to get a first pass layout for each cell. Alternately, you can just run each family individually as you work on them. At this point, don't worry about cells that fail.

Analyze/Adjust Combinational Gates

Now that the cell-template is all set up, start creating the first families of cells. As always, start with the easiest cells and work up to the most complex cells. This section concentrates on the inverters/NAND/NOR/

AOI/OAI/BUFFER/And/OR/AO/OA cells. These cells are vital because they are used as subgenerators in virtually all other cells.

First, verify that the folding parameters are properly set up. The folding parameters are:

In the Protech>Cell template configuration:

p transistor max width 2.00

n transistor max width 1.00

These set the maximum allowed transistor widths for p- and n-transistors before the transistors must be folded (regardless of routing congestion).

Generally, this is set to the maximum possible transistor sizes that will fit between the edge of the cell and the edge of the n-well. These can be calculated based on rules between the transistor and the tie on the edge of the cell, and the rules between the transistor and the edge of the n-well. Alternately, these can simply be measured from a sample inverter layout that is known to be DRC-clean with maximum possible transistor sizes.

Analyze/Adjust Mux/Xor/Adder Gates

Next, adjust any pass-gate style Mux/Xor/Adder gates that are not mapped using pure combinational gates, if any.

Analyze/Adjust Latch/Flop Gates

Next, run the latches and flops. Create any sub-generators that are needed to match the netlists or adjust the placement/routing.

Run Experiments

Run all of the above cells using various experiments, such as changing design-rules, transistor sizes, cell-template information, etc.

Strategy for Reducing metal-2 usage

Use outer-channel routing. This is the most important strategy for reducing metal-2 usage. Any long input-trunk that crosses feed-trunks that connect between the p-transistors and n-transistors will cause those feed-trunks to be in metal-2. Moving those long input-trunks to the outer-channels (above the p-transistors or below the n-transistors) will enable the feed-trunks to remain in metal-1. Depending on the topology, multiple outer-channel routes are possible. Also, some can be in metal-1, and others can be in poly (depending on whether the dmwire is used for power-rail diffusion connections).

Sometimes, by using the m1over terminal of the transistors (which allow a metal signal to be routed over the gate of the transistor), a more circuitous route can be used that avoids crossing of feed-trunks. The route_outer command cannot do this; route_override or prosticks must be used.

Change stitch-order to avoid letting feed-trunks cross input-trunks. This can be done globally by adjusting the metal-2 weight of the routing optimization. Or, the stitch-order can be overridden on a per-cell basis.

Strategy for Making Cells Fit into the Cell Height

(Merge with info from "Running Existing Layout Generators", which may be out of date.) If the cell doesn't fit into the cell-height, run compaction again, this time with y_fast compaction:

```
% *progen -t mytech/progen.tcl -i cel/invx1.cel \

-o agd/invx1.agd.yfa -f co=y_fast >&! log/invx1.log.yfa*
```

Then review the result with proview. The critical path is highlighted as part of the critpath layer. For more information, look at the log-file (search for "->->") for a description of the forces through the critical path.

The critical path will be listed as a set of forces between edges of the layout. The first and last forces listed are usually border-spacing rules from the cellborder to some layout edges. The rest will be most spacing forces between edges of the layout. If a geometry is flexible, for example a wire, then there will be minimum width forces as well from the low to high edges of the flexible geometry.

Check to make sure that all the design-rules in the critical path are correctly entered in the technology file. In particular, check the border-spacing rules. A border-spacing rule will be listed in the log file as a "<layer> to border spacing force", if it is applying the border-spacing rules found in /mytech//layers.tcl. Alternately, the force can be a spacing rule to one of the geometries found in the cell-template (as described in / mytech//stdcell.tcl).

If the cell doesn't fit, the transistor sizes may be too big for the cell-template. Check "pro_set max_f pdiff_gate" and "ndiff_gate". These set the maximum transistor size before folding. See also ProSpin to let the generators calculate folding vs. spice-based folding. See also "Running Existing Layout Generators" for how folding is calculated and the various folding options.

Increase xdelta_con and xdelta_nocon such that poly-contacts in the outer channels can slide past the poly gate extension. Try to make cells fit in the first y-pass. Trying to get the 2d compactor to make it fit can result in inefficiencies in the x-direction.

Run cells with -f co=y_fast, to get critical path through y. For nets that are on the critical path, try the following to eliminate it from the critical path:

Change track assignment for those nets (and any nets that start to the left of the critical net).

Check to see if any poly-contacts were not merged that can be merged.

Change stitch-order to let nets slide past each other, thus changing the layout shown in Figure 1:

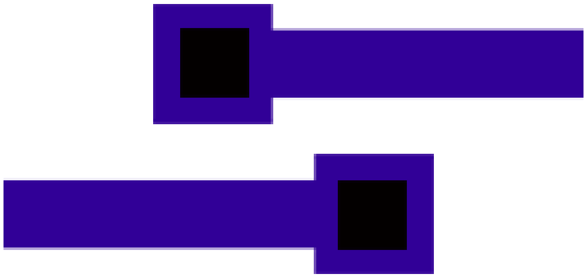


Figure 1

to the layout shown in Figure 2:

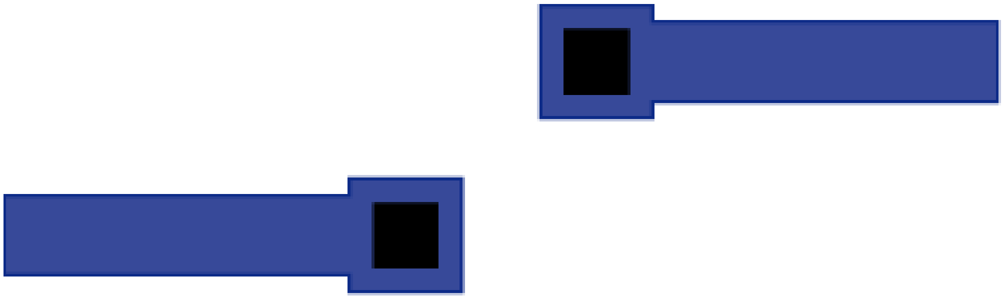


Figure 2

Push signals down to poly or from poly to met1. Use less met1 tromboning (see Figure 3), allow met2 usage instead, as shown in Figure 4 (generally a second-to-last resort).

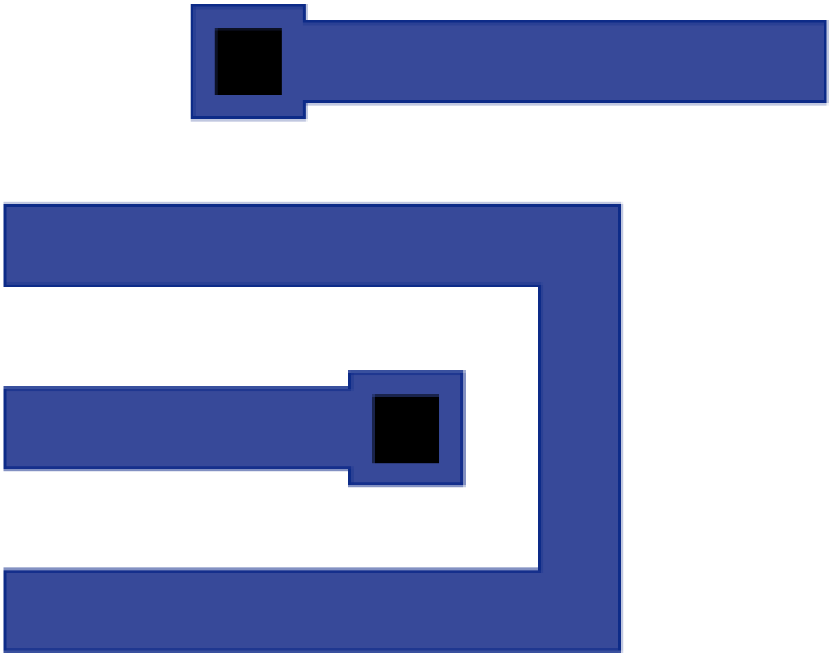


Figure 3: Excessive tromboning in metal-1

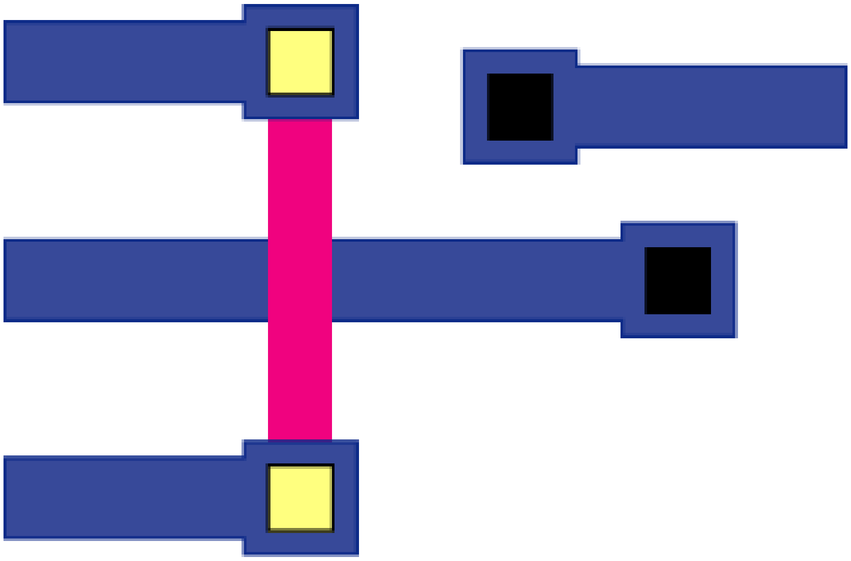


Figure 4: Reduce tromboning by using metal-2

Change input-trunks to metal-3 (generally a last resort, and only if allowed).

Summarize All Results

Summarize all the results and create a report describing the cells created, their area, and any experiments run.

Folding Control FAQ

What are the folding settings when using a fixed n-well?

What are the folding settings when using a notched n-well?

When using system-defined folding rather than requiring ProGenesis to follow the folding in the Spice netlist, the following determines the number of folds for the p- and n-transistors:

If there are no other folding options set, run Protech. Under the cell template tab click Edit for any cell template you have defined. In the fields "N(P)-Transistor maximum width," put the values for the maximum transistors width before folding (see "Analyze/Adjust Combinational Gates").

Alternatively, you can set the same values in the tech files. This has the advantage of being able to use different widths for different cells in the same library.

I'd like better control over folding with a fixed well. What can I do?

When using system-defined folding rather than requiring ProGenesis to follow the folding in the Spice netlist, the following determines the number of folds for the p- and n-transistors:

Use pro_set to set the maximum transistor width. This can be accomplished using ndiff_gate and pdiff_gate, as shown here:

pro_set max ndiff_gate max width in integer units

pro_set max pdiff_gate max width in integer units

Any transistor whose width is larger than the maximum transistor width for its type will be folded into smaller transistors whose width for each fold (also known as a leg) will be less than or equal to the maximum.

For most generators, the p- and n-transistors will be matched as a dual transistor structure. For those cases, the total transistor width will therefore equal total_gate = maxn + maxp

In some situations, the total transistor width is not allowed to be equal to the total of the independent transistor widths. There are several methods available for controlling this behavior.

The maximum total transistor width can be set as follows:

Setting directly

```
pro_set max total_gate <max width in integer units>
```

By setting total_gate, the total width of a dual structure will be less than or equal to total_gate.

Decreasing total_gate to make room for routes
When large input stacks (for example due to a NAND4) require extra routing resources, the total transistor width available can be reduced using free_inputs and excess_input_cost.

The value of the free_inputs field is set by:

```
pro_set celltemplate free_inputs num_routes_fit_max_trans
```

The free_inputs should be set to the number of poly-contacted metal routes that fit in the center channel between the p-transistors and n-transistors when the p-transistors and n-transistors are at their maximum (maxn, maxp). If the depth of the transistor stack exceeds free_inputs and the folding style requires horizontal routes to connect the same-net gates of the folded transistors, then the total transistor width (total_gate) will be reduced by an amount equal to:

```
reduction = (stack depth - free_inputs) * excess_input_cost
```

Note that free_inputs can be a floating point number. The default value for free_inputs is 5, which effectively turns off reduction because stacks are never deeper than 5 in modern processes.

The value for excess_input_cost can be set by:

```
pro_set max excess_input_cost <vertical pitch in integer units>
```

The default value for excess_input_cost is the poly-contact to poly-contact routing pitch. It can also be set to:

```
pro_set max excess_input_cost "cpoly_to_wire"
```

to make it equal to the poly-contact to metal-1 wire pitch.

For a dual structure, the total p-transistor width plus n-transistor width will always satisfy the following:

```
p-width + n-width = total_gate - reduction
```

Where total_gate is either the sum of maxn and maxp, or as set directly by total_gate.

Remember, the reduction is only applied when the folding style dictates it. For example: if you set series and parallel folding to fold transistors in place, then no stacked horizontal routes are necessary to connect the folded transistors; therefore, reduction will not occur in that situation.

```
# Folding style where reduction is not used

pro_set topology fold_ser_in_place 1

pro_set topology fold_par_in_place 1
```

If you have different cells using different folding strategies, then the system will automatically apply the total transistor width reduction where needed. Also note that each stack is treated independently. For example: for an AOI31, the 3-stack has a stack depth of 3 and the single transistor has a stack depth of 1. Therefore, even if the width of the transistors are the same, the 3-stack may be folded more than the single transistor.

In summary, the following constraints are applied:

```
folded n-width = maxn

folded p-width = maxp

folded n-width + folded p-width = total_gate - reduction
```

When given a choice between folding the n-transistors, or folding the p-transistors (to satisfy the maximum total width), the system will fold such that the impact to the cell-width is minimized.

I'm using notched wells and would like better control over folding. What can I do?

If you are using a notched well, and you wish the folding to automatically compensate, set the following:

```
pro_set option npNwellType "notch"
```

Instead of the normal folding constraints, as defined in the previous section, the notched well takes advantage of the well boundary being able to move. This allows n-transistors to push the well boundary up and enables n-transistors that are larger than what would normally be allowed with a straight well edge. Likewise, p-transistors can push the well boundary down and enable larger than normal p-transistors. However, the total transistor width must be maintained, because any increase in an n-transistor means there is less room for the p-transistors, and vice versa.

The new constraints when using a notched well are as follows:

```
folded n-width = 2 * maxn

folded p-width = 2 * maxp

folded n-width + folded p-width = total_gate - reduction
```

Therefore, the n-width and p-width constraints are effectively removed. The total transistor width constraint will still be applied. The total_gate is still set by the maxn + maxp, as defined for the straight well case. The parameters can be overridden as follows:

```
pro_set max ndiff_gate_notch max n-width for a notched well in integer
units

pro_set max pdiff_gate_notch p-width for a notched well in integer units

pro_set max total_gate_notch total width in integer units
```

The above values will set the maximum values only for the notched well case. The default values are:

```
ndiff_gate_notch = 2 * maxn (straight well)

pdiff_gate_notch = 2 * maxp (straight well)

total_gate_notch = total_gate (straight well, which defaults to maxn +
maxp)
```

Note that the reduction due to routing folded stacks is still applied normally.

What alternative folding options exist?

Known reasonable alternates:

The following will fold series-stacks as stacks, but will fold the transistors of parallel-stacks individually in-place:

```
pro_set -nooverride topology fold_ser_in_place 0

pro_set -nooverride topology fold_par_in_place 1
```

Generally, it is recommended to set `fold_ser_in_place` to 0, which will reduce the number of diffusion contacts and often result in fewer diffusion gaps. However, each transistor in a stack is folded the same number of times when folding by stack. Therefore, if the transistors in the stack have radically different transistor sizes, it may be better to set `fold_ser_in_place` to 1.

For example, if a stack contains a large transistor and a minimum size transistor, folding as a stack would cause multiple minimum size transistors, whose total might be greater than the desired total width.

Generally, it is recommended that `fold_ser_in_place` and `fold_par_in_place` are set to the same value for better poly gate alignment and better poly routing.

The following forces the p-transistors and n-transistors to have the exact same number of folds:

```
pro_set -nooverride topology fold_sync 1

pro_set -nooverride topology fold_balance 0
```

The following allows for any mismatch of the number of folds:

```
pro_set -nooverride topology fold_sync 0

pro_set -nooverride topology fold_balance 0
```

The fold_sync and fold_balance options apply to all cells or subcircuits that are of the following five types: INVERTER, NAND, NOR, AOI, and OAI.

The fold_sync option also applies to tri-state inverters.

fold_balance

Tries to balance the widths of transistors between n-transistor and p-transistor stacks by increasing the number of folds. To balance the widths is to try and have the same total width, regardless of the number of folds. Balancing will only be done while the widths remain less then the maximum allowed widths and while the cell size does not increase.

Values:

0 = do not try to balance folds
1 = try to balance stack folds

fold_sync

Forces transistors of corresponding input stacks to have the same number of folds. This will override fold balancing (fold_balance).

Values:

0 = do not try to sync folds
1 = sync folds

Another option helps with the multi-notch folding for cells. It might not be required to get a cell that fits the cell height, but it's a good tool. Combine this with the multi-notched well and the notched folding setting in progen.tcl:

```
   pro_set option require_output_diffusion_island_rectangles on
```

Solutions may vary from cell to cell. Therefore, to get the best results, run multiple versions of the combinational cells with these various options and pick the best result

Navigating ProTech

Introduction

Prolific's ProTech is the tool used to configure process technology settings, including:

- Design rules
- Layer information
- Layout options

These settings are easily configured using ProTech's graphical user interface, which manages a database of the configuration options and provides context-sensitive help for every option. ProTech is an X-Windows application that runs on any ProGenesis platform.

Running ProTech

protech -- Invoke the ProTech graphical technology editor

ProTech is started from the command line by typing protech. A more complete description of the arguments for ProTech is detailed in this section.

Name

protech -- Invoke the ProTech graphical technology editor

Synopsis

protech [-h] [-geometry size] [-i database_file] [-q] [-t database_file] [-v]

Description

The protech command invokes ProTech, a graphical interface for library technology setup. ProTech allows users to select layout styles and options, set or adjust design rules, control layout layers, and perform cell template configuration, all from a simple user interface requiring no programming.

Arguments

-h

Display usage message on stdout

-geometry

Sets the size of the ProTech window.

WxH+X+Y

Sets the ProTech window width to W and its height to H, located X from the left side of the display, and Y from the top of the display.

full

Maximizes the ProTech window.

-i database_file

Opens the specified database_file for editing.

-q

Queues the tool execution for the next available license.

-t database_file

Opens the specified database_file for editing.

-v

Output tool version number.

Files

.prolificrc

Specifies simple setup information for all Prolific tools. See prolificrc(5) for more details.

progen.db

ProTech stores the technology configuration in a database file known by default as progen.db. The actual configuration information may be stored in a .db file with a different name.

Advanced settings file

Users who wish to use low-level configuration options for unusual situations may employ a Tcl file to supplement the .db file.

Cell template file

Importing an existing cell template into ProTech is simply a matter of loading a stdcell.tcl file. The ProTech interface may be used to change the cell template, and the Cell Template Wizard may be used to quickly and easily create a cell template from scratch.

Author

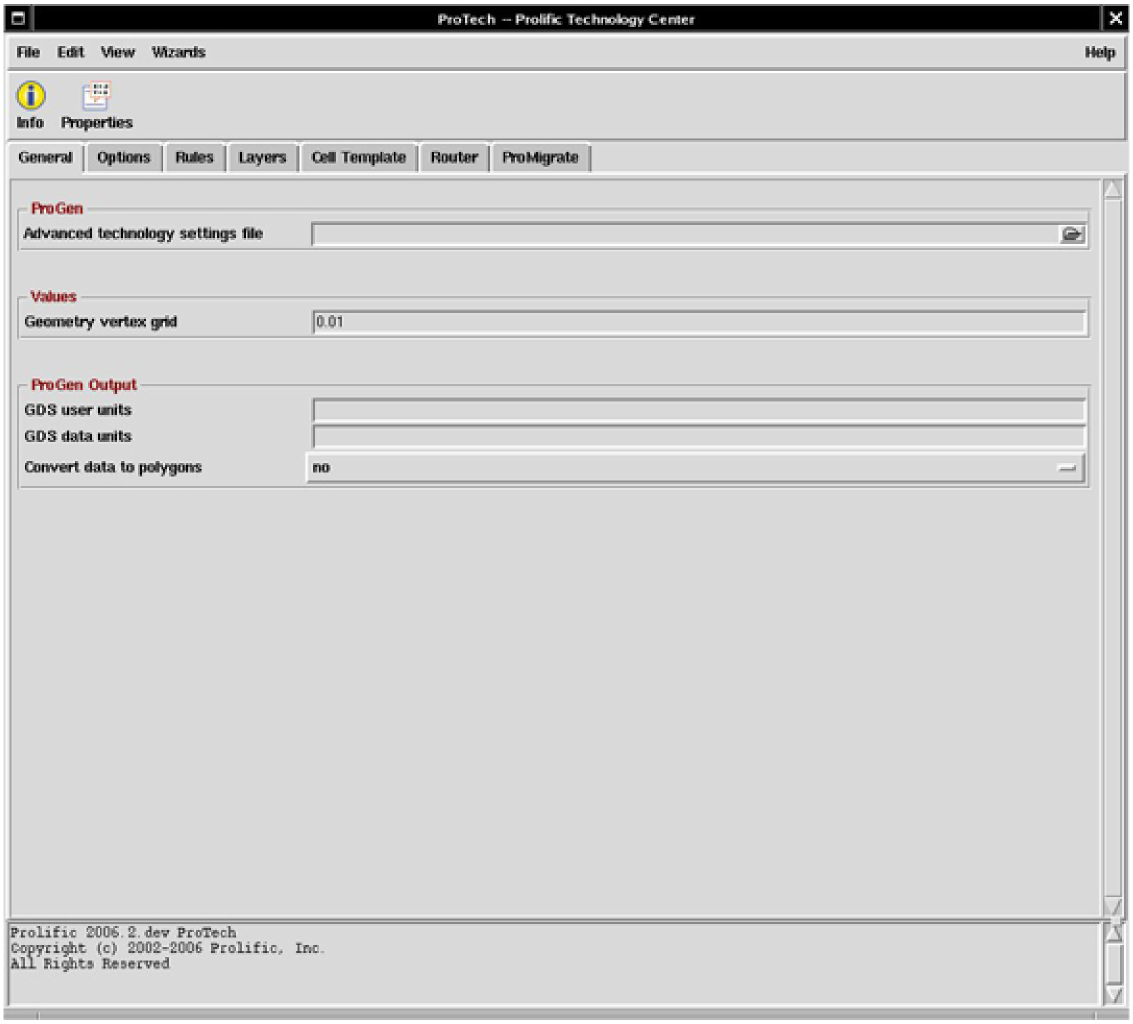
Prolific, Inc. <applications@prolificinc.com>

See Also

.prolificrc(5), PROLIFIC(5)

Navigating ProTech

The ProTech interface includes a menu bar, a tool bar, and several tabbed regions containing technology setup options.



Advanced technology settings file

Geometry vertex grid 0.01

GDS user units

GDS data units

Convert data to polygons no

```
Prolific 2006.2.dev ProTech
Copyright (c) 2002-2006 Prolific, Inc.
All Rights Reserved
```

Figure 1. ProTech Interface

Menu Bar

The menu bar contains tools for configuring ProTech and manipulating its features. There are five menus on the menu bar:

- File
- Edit
- View
- Wizards
- Help

File Menu

The ProTech File menu is used to control the files manipulated within the ProTech interface. The drop-down options are discussed below.

New

The New menu option causes the ProTech options to be restored to their defaults -- as though a new technology is being created. If an existing file has already been loaded when the New command is invoked, the existing filename is retained. To avoid killing an existing file, use the Save As command instead of the Save command to save the new technology to a new .db file.

Open

The Open menu option loads an existing .db file for editing. Changes made in ProTech will be saved back to the existing .db file by default (if the Save command is used rather than the Save As command).

Merge

The Merge option loads the contents of an existing .db file without editing that .db file. This can be useful for creating a new technology by making a copy of an existing technology.

Save

The Save option saves the current settings to a .db file. If the .db filename is known by ProTech because it has already been loaded or saved, the filename will be displayed in parentheses next to the Save command. Selecting the Save option will immediately save the ProTech settings into the .db file. If the .db filename is not known, ProTech will automatically open the Save As window.

Save As

The Save As option allows a new .db file to be created and saved. It may also be used to save the tech settings to an existing .db filename. Save As opens a window in which a directory and filename can be specified for the output .db file.

Save Rules

Because one set of design rules may be used with different technology configurations, and because different technology configurations may use multiple sets of design rules, the design rule portion of the ProTech configuration can be saved to its own .db file for reuse elsewhere using the File menu's Merge command.

Save ProView Layers

Saves layer information set in ProTech to a .db file for use within the ProView layout viewer.

Exit

The Exit option closes ProTech. It will not save changes to open .db files before exiting the application, however.

Edit Menu

The Edit menu is used to manage how ProTech responds to clicks on option labels. If neither of the Edit menu options is selected, clicking on one of the technology entry labels within the tabbed area of ProTech will not have any effect. How ProTech responds to these mouse clicks if one of the options is selected depends on the option:

Properties

The Properties option instructs ProTech to pop up a configuration dialog box when a technology setting's label is clicked. Some settings have options beyond the basic entry specified in the ProTech tab; to access these advanced properties, select the Properties option and click on the settings label.

For example: select Properties, and then click on the Options tab in ProTech. Click on the words Compaction effort to open a dialog box with a drop-down list of compaction effort settings, which are available from the main ProTech window, and a field for entering a custom compaction order, which is not available from the main ProTech window.

Describe

The Describe option instructs ProTech to pop up a help window when a technology setting's label is clicked. Context-sensitive help is available for each configuration setting.

View Menu

The View menu is used to control the display of the ProTech tool button bar. This menu is a tear-off list, so it may be opened as a separate window by selecting the dashed line at the top of the submenu. The four suboptions include:

Icon Bar

This option, which is enabled by default, displays the buttons in the tool bar. De-select this option to remove the tool bar from the ProTech interface. If this option is not selected, the other options on the Icons submenu will not be relevant.

Icon Labels

This option, which is enabled by default, displays text labels on the buttons in the tool bar.

Small Icons

This option, which is enabled by default, displays small button icons in the tool bar. Select Large Icons instead to display bigger button icons.

Large Icons

This option displays large button icons in the tool bar. Select Small Icons instead to display smaller button icons.

Help Menu

The ProTech Help menu offers help on several interface- and content-related subjects.

Contents

This is a top-level list of the help available in ProTech's graphical interface.

Menu Bar

Describes the ProTech menus.

Icon Bar

Describes the function of the buttons on the ProTech tool bar.

Concepts

Help for invoking ProTech and an overview of its layout.

About ProTech

Provides application and contact information.

Tool Bar

The ProTech tool bar is located below the menu bar. The tool bar contains the following tools:

Info

The Info button performs the same function as the Edit menu's Info entry. It instructs ProTech to pop up a help window when a technology setting's label is clicked. Context-sensitive help is available for each configuration setting.

Properties

The Properties button performs the same function as the Edit menu's Properties entry. It instructs ProTech to pop up a configuration dialog box when a technology settings label is clicked. Some settings have options beyond the basic entry specified in the ProTech tab; to access these advanced properties, select the Properties option and click on the settings label.

For example: select Properties, and then click on the Options tab in ProTech. Click on the words Compaction effort to open a dialog box with a drop-down list of compaction effort settings, which are available from the main ProTech window, and a field for entering a custom compaction order, which is not available from the main ProTech window.

ProTech Interface and Options

The ProTech uses a notebook tab interface to organize its contents. There are seven tabs in the ProTech configuration area:

- General
- Options
- Rules
- Layers
- Cell Template
- Router
- ProMigrate (if you have a license for ProMigrate)

General Tab

The General tab is used for system-wide, generic technology properties.

ProGen settings

The General tab includes an area for settings related to Prolific's ProGen tool.

Advanced techology settings file

The advanced technology settings file allows users to specify complex ProGen options not found in the ProTech interface. The value for this option is a path name to the settings file. The file contains the desired ProGen technology setup commands and the commands will override ProTech settings that refer to the same option.

For users converting an existing technology area, this file will be the main technology file for that area. By convention, this file is usually called progen.tcl, or, in older technology areas, tech.tcl. Users may click on the file folder button at the right of the entry field to pop up a file selection dialog box for specifying the settings file.

Values settings

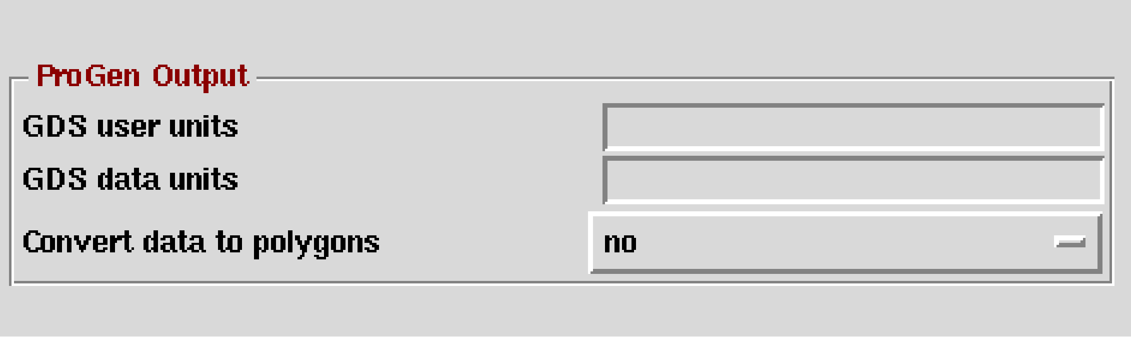
General technology setting values are stored in this section under the General tab.

Geometry vertex grid

The Geometry vertex grid is the manufacturing grid specified for the associated process technology. All geometric features must be on this grid to be fabricated correctly.

This value is set in microns.

ProGen Output



ProGen Output

GDS user units

GDS data units

Convert data to polygons: no

Figure 2

GDS user units and GDS data units

This entry is used in conjunction with the GDS data units entry to allow you to override the units written to the output AGD/GDS cell.

Values for these two entries do not typically need to be specified. ProGen automatically determines the units using the default values (user units =1e-3, data units = 1e-9) unless it needs a higher precision based on the Geometry Vertex Grid.

Values must be specified for both the GDS user units and GDS data units entries to override the system values. Setting only one value will have no effect.

A common reason that is needed to change the units is if a DRC deck is expecting a specific precision and ProGen's output is in a higher precision than that of the DRC deck. This is a scaling issue (set using this option) and if the allowed vertex grid is set correctly, ProGen will not violate it or the required precision.

Convert Data to Polygons

ProTech allows you to convert the fractured AGD data to polygons.   This will create shapes in the layout with more than four vertices. The default is to create fractured data. When the data is fractured the shapes in the layout will be rectangles. The geometries in the shapes are the same using either setting.

Options Tab

The Options tab contains several areas in which different setup options are configured, including compaction, routing, and folding options.

Compaction Related Options

The compaction-related options help tune the compaction process. Several options can be configured.

Compaction effort

The value of this option controls the extent to which ProGen will compact a cell. The option may have one of the following values: low, normal, high, or custom.

The low value directs ProGen to compact cells as quickly as possible while still producing valid cell layout. These layouts will typically not be as dense as layouts produced with the other compaction efforts but can provide useful feedback when initially running a library.

The normal value directs ProGen to compact cells, resulting in dense layout while minimizing run time. The layout will typically be as dense as the high compaction layout cells without the additional run time.

The high value directs ProGen to perform extensive compaction on cell layouts. This directive will result in longer run times but is useful when normal compaction is unable to compact layout in the desired cell size.

The custom value allows the user to control cell compaction based on the following directives:

x

y

x_2d

y_2d

x_skip_grid

y_skip_grid

x_skip_grid_2d

y_skip_grid_2d

x_fast

y_fast

To specify these directives the user must enter the properties dialog of the option, and select custom as the value of Compaction effort.

Diffusion contact growing

This option controls how ProGen will size diffusion contacts during compaction. The option may have one of the following values:

single contact

single contact, in transistor

transistor width

transistor width percentage

delay grow

delay grow, in transistor

The single contact value results in ProGen doing no contact growing. There will be a single contact. If the transistor has a diffusion wire from the source/drain to the contact then this option will attempt to pull the contact into the transistor during compaction but will not insist that its final position be in the transistor.

The single contact, in transistor value is the same as single contact except ProGen will insist that a contact on a diffusion wire be pulled into the transistor source/drain during compaction.

The transistor width value directs ProGen to grow all diffusion contacts to the full width of the corresponding transistor source/drain.

The transistor width percentage allows you to specify a percentage of the source/drain width that the contact should cover. For example: a 70 percent value would result in 70 percent of the transistor's width having contacts. The percentage value is entered as a property of the option.

The default percentage is 100.

The delay grow value causes ProGen to begin with a single contact object before compaction. It then attempts to increase the contact object up to the transistor width during the 2d y-compaction pass. This results in the contact object creating as many tiled contacts as possible without sacrificing compaction results. When a diffusion wire contact is used this option will attempt to pull the contact onto the source/drain region but will not insist on it.

The delay grow in transistor value is the same as delay grow except that ProGen will insist that the diffusion wire contact be pulled onto the transistor source/drain region.

> Delay grow in a transistor will force the contacts to be pulled in or be maintained in the s/d region. Therefore, it is important to make sure that this is possible given the technology settings (especially router settings) before enabling this option.
>
> The user may run into this issue when setting the dmwire contact position to be at midpoint while allowing outer channel routes. In this case the contact position starts outside the s/d region, but because of the "delay grow in transistor" setting will be required to be pulled back into the s/d region. If a routing solution has outer channel routes that cross the diffusion section of the dmwires, then it might not be possible to pull the contacts back. If the contacts must always pull back in, then the user should either disallow outer channel horizontal m1 routes, use jogged metal 1 outer channel routes, have the m1 routes route outside of the power rails (m2 inset rails only), or change the dmwire contact position to be at transistor (preferred).
>
> An error may occur when this when this option is selected and the diffusion contact plus diffusion contact overlap are larger than the transistor width.

Poly contact merging

This option controls how aggressively ProGen attempts to merge poly-contacts in the pre-compacted layout. The option may have one of the following values:

none

transistor pitch

in pairs

threshold

When none is specified, ProGen will not attempt to merge poly-contacts.

The transistor pitch value causes ProGen to merge poly-contacts if their pre-compacted spacing is less than or equal to the transistor pitch. This typically results in a low number of contacts being merged.

The in pairs value causes ProGen to merge every two adjacent poly-contacts.

The threshold value allows the user to control the ProGen poly-contact merging characteristics with the following options in the Properties dialog:

Merge contacts with spacing less than threshold

Directs ProGen to merge poly-contacts which have a pre-compacted spacing less than the value specified.

The value is in microns.

Maximum number of contacts to merge (optional)

Allows you to specify the maximum number of contacts to merge regardless of the merge spacing. ProGen will merge the contacts (within the merge spacing) from left-to-right until the maximum number is reached. It will then start a new contact. If no value is defined then there is no maximum value.

Number of rightmost contacts to merge (optional)

When merging poly-contacts left-to-right there are sometimes times when there a couple at the right end which you would like to have merged with the previous contact instead of starting a new group. The number of right-most contact to merge property allows you to specify how many contacts at the right end should be merged with the previous group instead of the creating a new contact.

Pull transistors

This option allows you to direct ProGen to slightly pull transistors outward or inward during compaction. There are special circumstances in which this could help the compaction process. (For example: if you have a large number of small transistors relative to the cell height with a lot of center channel routing, then pulling the transistor outward slightly could help better compact the routing; or if you have a lot of outer channel routing, then pulling the transistors inward can also help the process.)

The pull transistors option may have one of the following values:

none

inward

outward

The value none results in ProGen doing no special transistor pushing or pulling during compaction. This is the default value.

The inward value causes ProGen to pull transistors toward the cell center slightly during compaction. This can help the compactor some when there is a large amount of outer channel routing.

The outward value causes ProGen to pull transistors toward the top and bottom cell borders. This can help the compactor some when there is a large amount of center channel routing.

Automatic overlap orientation

When there are asymmetric contact/via overlap rules this option turns on and off the capability of having ProGen determine the orientation used per contact/via.

Asymetric rules are specified as four-sided values where the top and bottom sides have one value and the left and right sides have a different value. The orientation is called horizontal when the left and right values are greater than the top and bottom values. It is called vertical when the top and bottom values are greater.

This option has the following values:

yes: ProGen determines overlap orientation

no: ProGen uses orientation defined by rule.

Properties

This compaction pass number determines orientation.

Properties allows you to override at which point ProGen makes the orientation decision. The default is for ProGen to determine the orientation just after the pass where both one x-compaction and one y-compaction have taken place.

Override the decision point by specifying a compaction pass number. The first compaction pass is number 0, the next pass is 1, and so forth.

When compaction is x_skip_grid y_skip_grid x_2d y_2d x y the following pass numbers indicate when the orientation is determined:
0 determines orientation after x_skip_grid
1 determines orientation after x_skip_grid, y_skip_grid
2 determines orientation after x_skip_grid, y_skip_grid, x_2d
3 determines orientation after x_skip_grid, y_skip_grid, x_2d, y_2d
4 determines orientation after x_skip_grid, y_skip_grid, x_2d, y_2d, x
5 determines orientation after x_skip_grid, y_skip_grid, x_2d, y_2d, x, y
In this example, the default is 1. This is the point where one x-compaction and one y-compaction has occured.

Example 1: Orientation Compaction Pass

Layer Options

The layer options are used to configure layer-specific settings.

Minimum area metal rules

The value of this option determines whether or not ProGen will try to resolve minimum area metal rules. If the option value is yes, then the values of the minimum metal area rules must be entered in square microns on the Rules page or in the Properties window.

The default is yes.

Implant layers

The value of this option determines whether or not ProGen will generate n+ and p+ implant layers in the final cell layout.

If the option value is yes, then the corresponding design rules and GDS layer information must be supplied on the Rules page or in the Properties window.

If the option value is no, then n+ and p+ implant layers will not be output in the final cell layout and implant layer rules will not be enforced.

The default value is yes.

Separate N- and P- gate-length rules

The value of this option specifies whether or not the default transistor gate length is the same for n- and p-transistors.

If the option value is yes, then n- and p-transistors have different default gate lengths and these two gate length values must be entered on the Rules page or in the Properties window. If the option value is no, then the single default transistor gate length must be entered on the Rules page or in the Properties window.

The default value is no.

Note that these values are only used when gate length values are not specified in the ProGen input file. Gate lengths of all transistors may be controlled individually in the ProGen input file.

Diffusion contact deep implant

The value of this option specifies whether or not diffusion contacts require a deep implant layer. When the option value is yes, the GDS layer information for the deep implant layers must be in the layer page or in the properties window.

General Options

The general options cover special treatments, (eg, bent gates, rectangular source/drain regions, transistor alignment, and labeling of internal nodes).

Transistor gate bending

The value of this option specifies whether or not transistor gates with 45 degree bends are allowed. If the option value is yes, then design rules specific to bent gates may be specified on the Rules page or in the Properties window.

The default value is no.

Straighten transistor source/drain edges

The value of this option determines whether the contact diffusion overlap will be selectively increased to create transistor source/drain regions which are rectangular.

If the required diffusion extension from gate is less than the sum of the contact-to-gate spacing, the contact width, and the diffusion overlap of contact, then transistor source/drain regions that are not fully contacted may have irregular shapes. The value yes will result in the diffusion areas increasing to make the source/drain regions rectangular. The value no allows the source/drain regions to be irregularly shaped.

The value of this option determines whether the contact diffusion overlap will be selectively increased to create transistor source/drain regions which are rectangular.

If the required diffusion extension from gate is less than the sum of the contact-to-gate spacing, the contact width, and the diffusion overlap of contact, then transistor source/drain regions that are not fully contacted may have irregular shapes. The value yes will result in the diffusion areas increasing to make the source/drain regions rectangular. The value no allows the source/drain regions to be irregularly shaped.

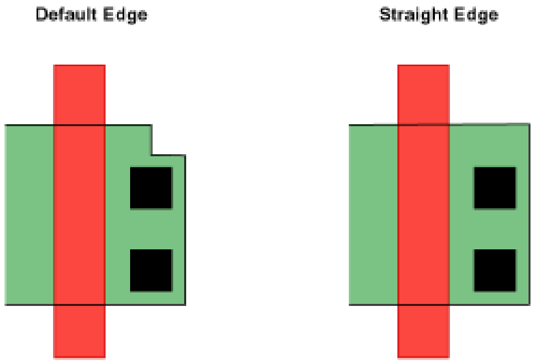


Figure 3

Align same size transistors

The value of this option determines whether or not adjacent transistors of the same width are forced to align their active areas with each other. If the option value is yes, then the transistors must be aligned in the final layout.

Typically, transistors that are aligned may be spaced more closely together. However, some layouts may require non-aligned transistors to achieve the desired result.

The default value is yes.

Text internal nodes

The value of this option determines whether or not internal nodes are labeled with their net names in the final layout. If the option value is yes, then internal nodes will be labeled and the text will generally appear over one of the diffusion nodes of the net. If the option value is no, then only ports will be labeled. This option may also be activated on the ProGen command-line using the -f text flag.

The default value for this option is yes.

Text internal nets

This option turns on and off the creation of internal node text in the final layout. This has no effect on port texting. The following values are available:

yes
Turn on texting of internal nodes.

no (default)
Turn off texting of internal nodes.

This option can also be activated on the ProGen command-line by using the -f text flag:

```
progen -t progen.db -i i.cel -o i.agd -f text
```

When internal node texting is turned on, Using master net names for text is an option which indicates if internal nets should be texted with their master names or their internal names.

Internal names correspond to sub-nets while master names correspond to global net names.

The following values are available:

yes (default)
Use master names for net texting.

no
Use internal names for net texting.

Routing Options

The routing options provide instructions for how the router should treat certain situations, as described in this section.

Split horizontal M2 wires into M1/M2 segments

When there is a vertical M2 wire which connects to a horizontal M1 wire the resulting ProGen route will contain three vias. One in the center of the M2 wire where it connects to the M1 horizontal wire and one at each end where it connects to the M1 of the transistor contacts.

Setting the value of this option to yes results in ProGen attempting to route one half of the M2 wire in M1 so that there are only two vias.

A value of no results in the default behaviour described above.

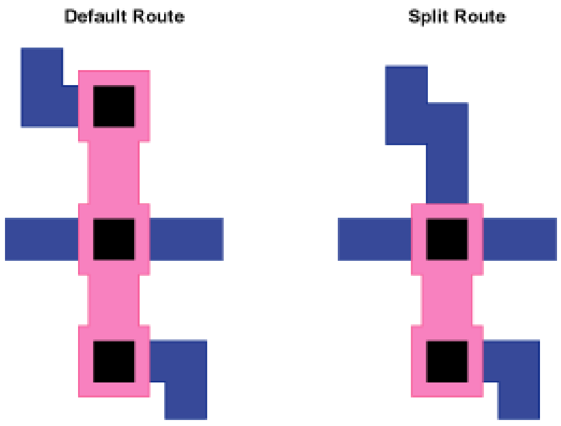


Figure 4

Metal-2 routing usage

This option controls how hard ProGen will try to minimize the use of a second metal layer during routing. The option may have one of the following values: minimal, normal, or freely.

The minimal value directs ProGen to try really hard to not use a second metal layer for routing. This can result in longer routing run times and large amounts of jogging in the first metal layer.

The normal value is the ProGen default. ProGen tries to minimize the use of the second metal layer routing but will not sacrifice cell size.

The freely value allows ProGen to use as much second metal layer routing as needed to route across the channel.

Objects

The objects options relate to the use of special objects under specified conditions.

Diffusion/Metal-1 contact wire

The value of this option specifies whether or not ProGen connects diffusion nodes to the power or ground rails with a combination diffusion and metal wire. If the option value is no, then ProGen makes the connection with a metal wire. If the option value is yes, then ProGen makes the connection with diffusion wire followed by a metal wire.

The default is no.

If the option value is yes, then the following characteristics of the combination diffusion and metal wire may be specified in the Properties window:

Contact initial position

Contact final position

Layer stack

Supply rails overlap cell border

Metal overlap orientation of contact and via-1

Extra metal connection to ties

The initial position of the diffusion contact may be specified by one of the following values: attran, midpoint, or atrail.

The value attran specifies that the initial contact position should be in line with the center of the associated transistor.

The value midpoint specifies that the initial contact position should be half way between the transistor and the rail.

The value atrail specifies that the initial contact position should be at the center of the rail.

The default is midpoint.

The final position of the diffusion contact may be specified by one of the following values: none or atrail.

The value none specifies that the final contact position is not restricted.

The value atrail specifies that the metal overlap of the contact should be fully contained by the rail after compaction. Note that this value is only meaningful if the initial contact position is specified as atrail.

The default is none.

The layer stack property may have one of the following values:

m1

m1 v1 m2

The m1 value means use a metal-1 wire to connect to a metal-1 supply rail.

The m1 v1 m2 value means use a metal-1 wire that connects to a metal-2 supply rail through a via.

The default is m1.

If the power or ground rails overlap the cell border, set that property to yes. If the rails do not overlap the cell border, set the property to no.

The default is yes.

The next four properties control how the metal overlaps of contact and via-1 are oriented. Depending on the value of the layer stack property, only the relevant rotation properties will be active. These properties will only make a difference if the overlaps specified in the rules section have different values for different directions.

The default is no for all the rotate properties, making the overlap orientations as specified in the Rules tab.

The last property specifies whether or not an extra piece of metal-1 will be added to connect to either a well or substrate tie. This property is only available if the layer stack ends at metal-2 and the well and substrate ties in the cell template are defined using the standard pro_tie procedure.

The default is no.

Allow strap-taps

Indicates if strap-taps should be placed in cell layout when cells, generators, or technology files contain directives to place them.

A strap-tap is a tie which abuts a transistor source diffusion, is contacted, and connects to the transistor source contact with metal-1. The system places strap-taps based on directives in the ProSticks generator.

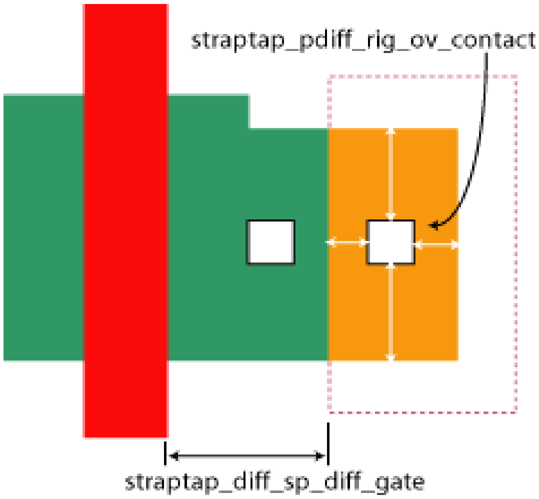


Figure 5: Example of a strap-tap.

The option can have one of the following values:

yes

Create strap-taps in layout when directed. This is the default value.

no

Do not use strap-taps. Ignore references to them in cells, generators, and technology files. ProGen will use the default power and ground contacts in place of the strap-taps.

Folding Options

The folding options control the style and amount of transistor folding performed by the automatic folding algorithms of various generators.

Fold parallel stacks

The value of this option determines the folding style for stacks of parallel transistors. The option may have one of the following values: by stack orby transistor.

The by stack value specifies that each leg of a transistor will be in its own stack such that the unit of repetition is a stack rather than a transistor.

The by transistor value specifies that all the legs of a transistor will be adjacent to each other so that the unit of repetition is a transistor rather than a stack.

The default value is by stack.

Fold series stacks

The value of this option determines the folding style for stacks of series transistors. The option may have one of the following values: by stack or by transistor.

The by stack value specifies that each leg of a transistor will be in its own stack such that the unit of repetition is a stack rather than a transistor.

The by transistor value specifies that all the legs of a transistor will be adjacent to each other so that the unit of repetition is a transistor rather than a stack.

The default value is by stack.

N/P transistor fold balancing

The value of this option controls the relative amount of folding between n- and p-transistors. The option may have one of the following values:

none

same number of folds

fold to fill free space

The none value specifies that n- and p-transistors are folded independently of each other. The number of legs is calculated based on their width and the space available in the cell template.

The same number of folds value specifies that corresponding n- and p-transistors must be folded the same number of times.

The fold to fill free space value specifies that the side (n or p) which uses less of the cell width after folding based on transistor width should be folded more times to use up the empty space in the cell with out increasing the cell width.

Rules Tab

The Rules tab contains the design rule settings used by the technology being defined. This tab is used to configure, modify, or re-configure a design technology.

Information

The Information area contains information about the rule document containing the applicable design rules.

Design rule document name

A place for the user to enter the design rule document name associated with the rules. The value is echoed in the ProGen log file.

Design rule document version

A place for the user to enter the design rule document version number with the rules. The value is echoed in the ProGen log file.

Well Rules

The Well Rules define well size and overlap rules.

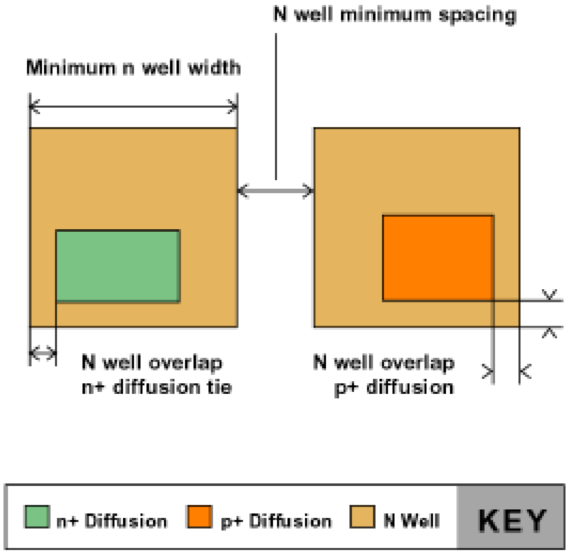


Figure 6

Minimum n-well width

N-well minimum width.

N-well overlap n+ diffusion tie

N-well minimum overlap of n-diffusion (typically a tie/tap).

N-well overlap p+ diffusion

N-well minimum overlap of p-diffusion.

N-well space to p-well

The minimum spacing between n-well and p-well geometries. All values are in microns. The default value is 0.0 microns and does not require any additional settings to have it applied.

This value is ignored for technologies that do not define a p-well layer.

Implant Rules

The Implant rules define implant size, spacing, and overlap rules.

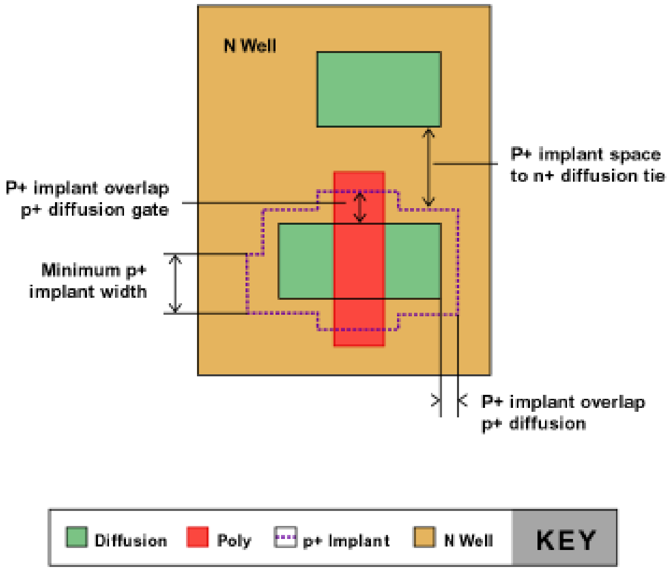

N Well
P+ implant space to n+ diffusion tie
P+ implant overlap p+ diffusion gate
Minimum p+ implant width
P+ implant overlap p+ diffusion
Diffusion
Poly
p+ Implant
N Well
KEY

Figure 7

Minimum p+ implant width

Minimum implant width. The value is used for both n- and p- type implants.

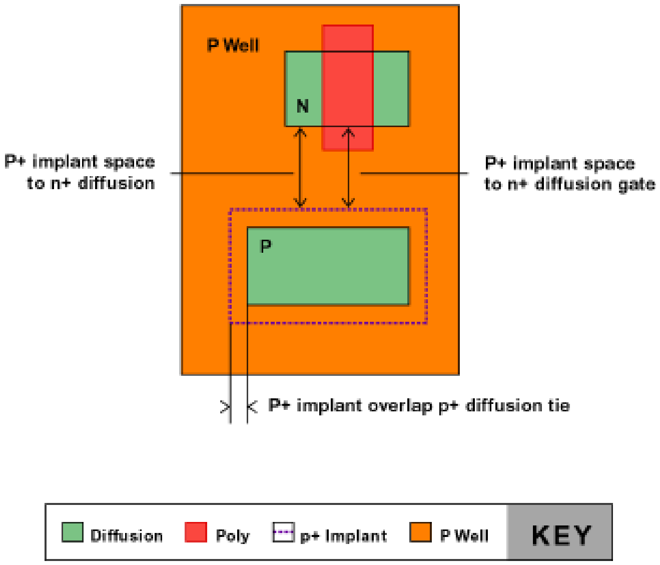

P Well
N
P
P+ implant space to n+ diffusion
P+ implant space to n+ diffusion gate
P+ implant overlap p+ diffusion tie
Diffusion
Poly
p+ Implant
P Well
KEY

Figure 8

P+ implant space to n+ diffusion

P-implant spacing to n-diffusion.

P+ implant space to n+ diffusion gate

P-implant minimum spacing to n-diffusion gate.

P+ implant space to n+ diffusion tie

P-implant minimum spacing to n-diffusion tie.

P+ implant overlap p+ diffusion

Minimum p-implant overlap of p-diffusion.

P+ implant overlap p+ diffusion gate

P-implant minimum overlap of p-diffusion gate.

P+ implant overlap p+ diffusion tie

P-implant minimum overlap of p-diffusion tie.

Diffusion Rules

The diffusion rules define diffusion size, spacing, overlap, and extension.

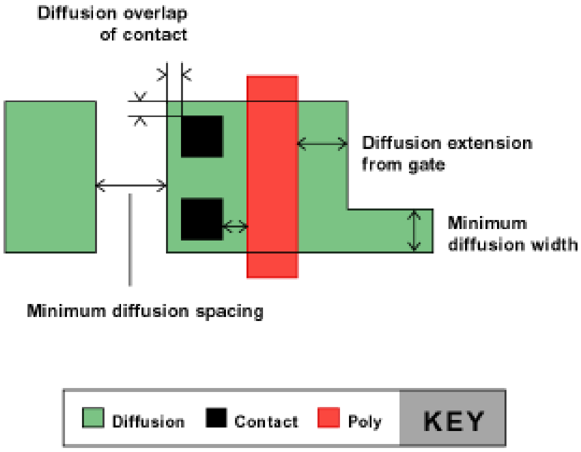

Diffusion overlap of contact
Diffusion extension from gate
Minimum diffusion width
Minimum diffusion spacing
Diffusion
Contact
Poly
KEY

Figure 9

Minimum diffusion width

Minimum diffusion width. This value is applied to both n- and p- type diffusion.

Minimum diffusion spacing

Minimum diffusion-to-diffusion spacing. The spacing is applied to all types of diffusion.

Diffusion overlap of contact

Minimum diffusion overlap of contacts. The value enter can take one of the two forms: a single value or a list of four values. Positive values cause the diffusion to extend outside of the contact. Negative values result in the diffusion being within the contact.

When a single value is specified it is applied to all sides of the diffusion overlap of the contact.

A list of four values specifies the overlaps for each of the four sides of the contacts. The list of overlaps have the order left bottom right top.

Diffusion extension from gate

Minimum diffusion overlap of gate region. This value really defines the minimum amount which source/drain diffusion must extend from the gate poly.

Minimum width butted diffusions (N+ or P+)

This value defines the minimum width of the diffusion where n-diffusion abuts p-diffusion.

N+ diffusion space to n-well

Minimum spacing of n-diffusion to n-well when n-diffusion is outside of n-well.

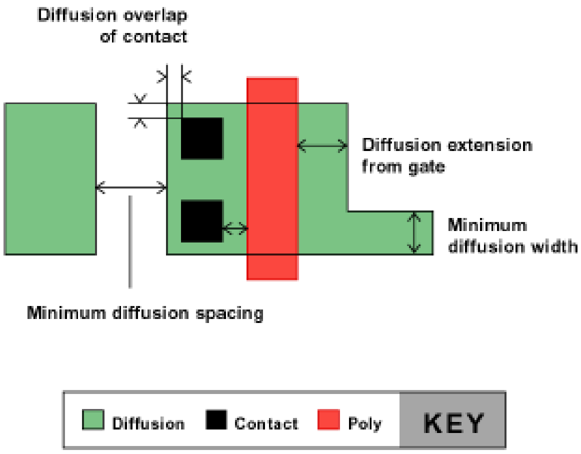

Diffusion overlap of contact
Diffusion extension from gate
Minimum diffusion width
Minimum diffusion spacing
🟩 Diffusion ■ Contact 🟥 Poly
KEY

Figure 10

Minimum poly width

Minimum poly width.

Minimum poly spacing

Minimum poly to poly spacing.

Minimum poly overlap of contact

The value enter can take one of the two forms: a single value or a list of four values. Positive values result in the poly extending outside of the contact. Negative values result in the poly being within the contact.

When a single value is specified it is applied to all sides of the poly overlap of contact.

A list of four values specifies the overlaps for each of the four sides of the contacts. The list of overlaps have the order left bottom right top.

Minimum poly space to diffusion

This value is applied to all diffusion types.

Poly-gate endcap extension

Minimum poly endcap extension beyond gate.

Poly Rules

The poly rules define polysilicon size, spacing, overlap, and extension.

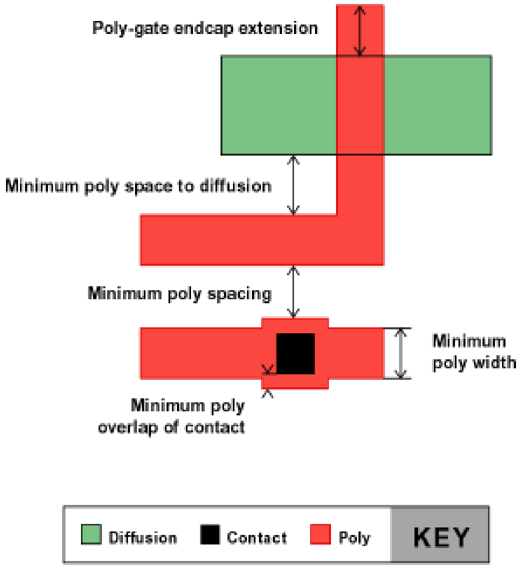

Poly-gate endcap extension
Minimum poly space to diffusion
Minimum poly spacing
Minimum poly width
Minimum poly overlap of contact
Diffusion
Contact
Poly
KEY

Figure 11

N+ diffusion space to n-well

Minimum spacing of n-diffusion to n-well when n-diffusion is outside of n-well.

Minimum poly width

Minimum poly width.

Minimum poly spacing

Minimum poly to poly spacing.

Minimum poly overlap of contact

The value enter can take one of the two forms: a single value or a list of four values.

Positive values result in the poly extending outside of the contact. Negative values result in the poly being within the contact. When a single value is specified it is applied to all sides of the poly overlap of contact.

A list of four values specifies the overlaps for each of the four sides of the contacts.

The list of overlaps have the order left, bottom, right, top.

Minimum poly space to diffusion

This value is applied to all diffusion types.

Poly-gate endcap extension

Minimum poly endcap extension beyond gate.

Gate Rules

The Gate Rules define the size, spacing, and bend restrictions for gates.

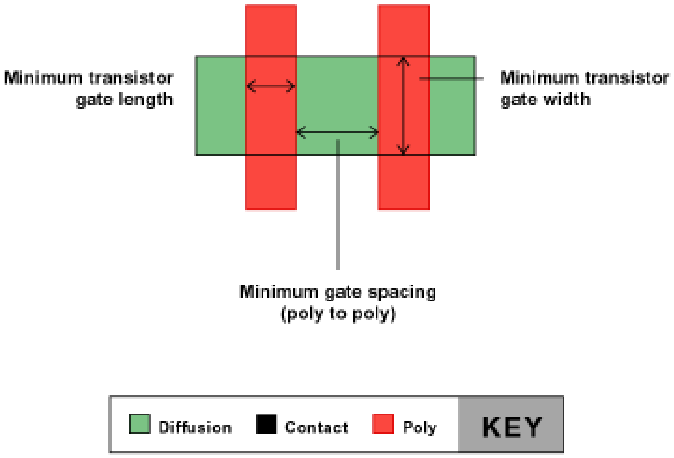

Minimum transistor gate length
Minimum transistor gate width
Minimum gate spacing (poly to poly)
🟩 Diffusion
■ Contact
🟥 Poly
KEY

Figure 12

Minimum transistor gate width

Minimum gate width.

Minimum gate spacing (poly to poly)

Minimum gate to gate spacing.

Minimum transistor gate length

Minimum gate length.

Minimum n-transistor gate length

Minimum gate width for n-diffusion transistors.

Minimum p-transistor gate length

Minimum gate width for p-diffusion transistors.

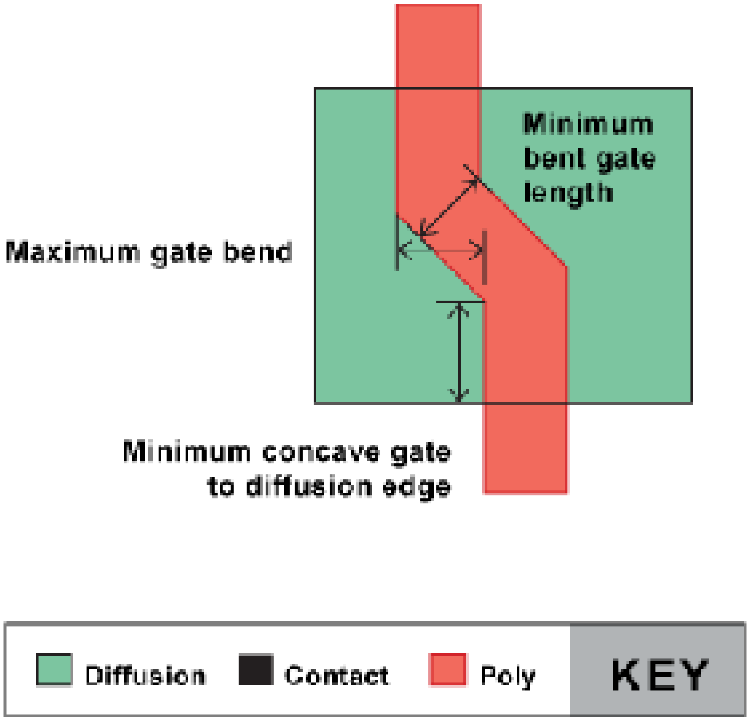


Figure 13

Maximum gate bend (optional)

The maximum amount that a transistor gate is allowed to bend in the orthogonal direction from the transistor width.

In general, the bent gate option is used to reduce the cell area. By bending a gate, the effective transistor width can be larger than the straight line transistor width measured. However, if the orthogonal portion of the gate gets too long, it will actually make the cell width wider.

So the maximum gate bend size is really the maximum useful bend size in order to effectively reduce cell size. The useful bend size can be calculated by comparing the contacted pitch for a parallel stack of transistors and set the maximum bend to the smallest bend that will give you the best pitch with bends.

The max_bend default value = (contacted gate pitch - uncontacted gate pitch) / 2.0.

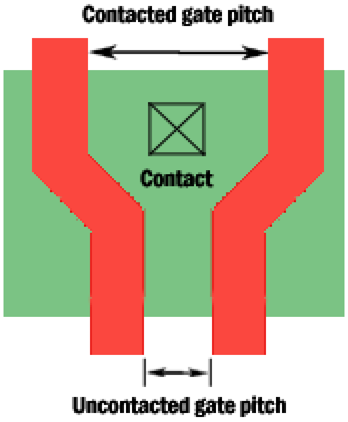


Figure 14

Minimum bent gate length

The minimum bent gate length. This allows you specify a value for bent gates which can be different then the non-bent minimum transistor gate length.

Minimum concave gate to diff edge (optional)

The minimum distance that a bent gate segment must be from the transistor diffusion/poly endcap edges. Specification of this value is optional. If not specified, then the diffusion-over-gate rule value is used.

Contact Rules

The Contact Rules define contact size and spacing.

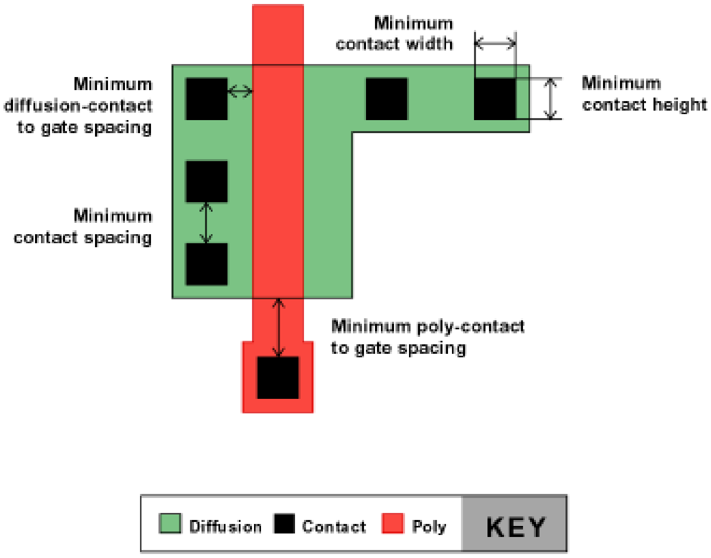


Figure 15

Minimum contact width

The exact size of all diffusion and poly contacts.

If one value is specified, that value will be used as both the minimum height and minimum width of the contact. If two values are specified, the first value specifies the minimum contact width, and the second value specifies the minimum contact height: width, height.

Minimum contact spacing

The minimum contact-to-contact spacing.

Minimum poly-contact to gate spacing

The minimum spacing of diffusion-contact to gate-poly.

Minimum diffusion-contact to gate spacing

The minimum spacing of poly-contact to gate-diffusion.

Metal-1 Rules

The Metal-1 Rules define size, spacing, overlap, and area for metal-1 geometries.

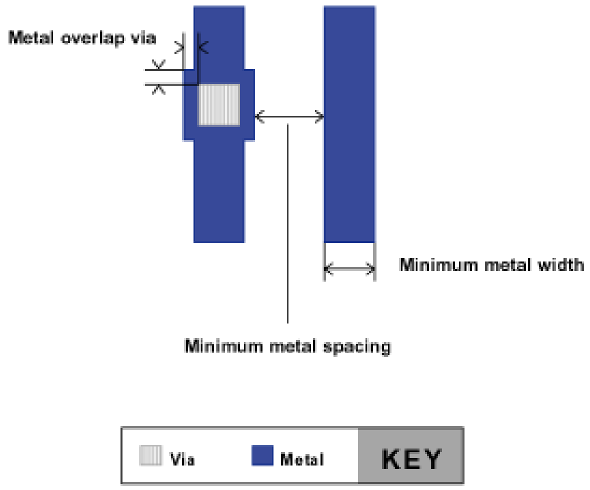


Figure 16

Minimum metal-1 width

The minimum width of metal-1 geometries.

Minimum metal-1 spacing

The minimum metal-1 to metal-1 spacing.

Metal-1 overlap diffusion-contact

Minimum metal-1 overlap of diffusion-contacts. The value entered can take one of the two forms: a single value or a list of four values. Positive values result in the metal extending outside of the contact. Negative values result in the metal being within the contact.

When a single value is specified it is applied to all sides of the metal overlap of the contact.

A list of four values specifies the overlaps for each of the four sides of the contacts. The list of overlaps have the order left bottom right top.

Preferred metal-1 overlap of diffusion-contacts

Value specification is optional.

When a preferred overlap is specified, ProGen will attempt to achieve the overlap during compaction, but it will not impact the final cell size in order achieve the overlap. The minimum overlap rules will still be enforced regardless of whether ProGen can achieve the preferred overlaps or not.

The specified value can take one of two forms: a single value or a list of four values. Positive values result in the metal extending outside of the contact. Negative values result in the metal being within the contact.

When a single value is specified it is applied to all sides of the metal overlap of the contact.

A list of four values specifies the overlaps for each of the four sides of the contacts. The list of overlaps have the order left bottom right top.

Metal-1 overlap poly-contact

Minimum metal-1 overlap of poly-contacts. The value entered can take one of the two forms: a single value or a list of four values. Positive values result in the metal extending outside of the contact. Negative values result in the metal being within the contact.

When a single value is specified it is applied to all sides of the metal overlap of the contact.

A list of four values specifies the overlaps for each of the four sides of the contacts. The list of overlaps have the order left bottom right top.

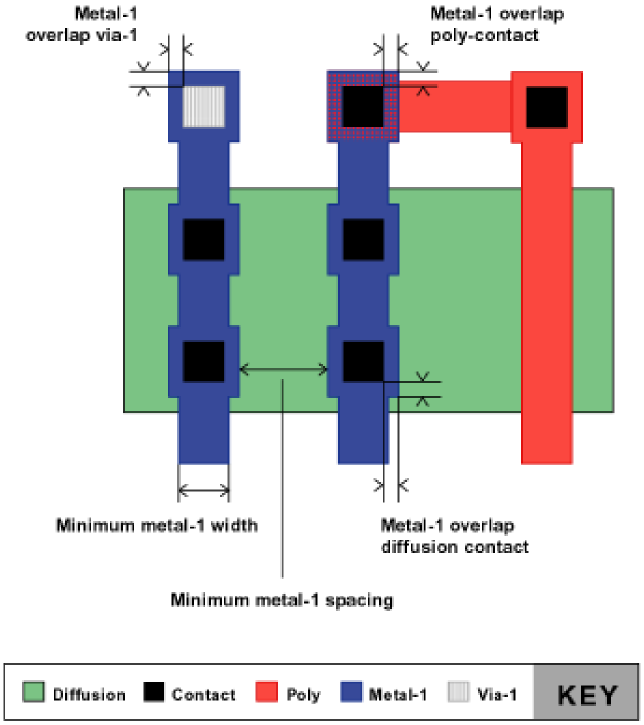


Figure 17

Preferred metal-1 overlap of poly-contacts

Value specification is optional.

When a preferred overlap is specified ProGen will attempt to achieve the overlap during compaction but it will not impact the final cell size in order achieve the overlap. The minimum overlap rules will still be enforced regardless of whether ProGen can achieve the preferred overlaps or not.

The specified value can take one of two forms: a single value or a list of four values. Positive values result in the metal extending outside of the contact. Negative values result in the metal being within the contact.

When a single value is specified, it is applied to all sides of the metal overlap of the contact.

A list of four values specifies the overlaps for each of the four sides of the contacts. The list of overlaps have the order left bottom right top.

Metal-1 overlap via-1

Minimum metal-1 overlap of via-1. The value entered can take one of two forms: a single value or a list of four values. Positive values result in the metal extending outside of the via. Negative values result in the metal being within the via.

When a single value is specified it is applied to all sides of the metal overlap of the via.

A list of four values specifies the overlaps for each of the four sides of the vias. The list of overlaps have the order left bottom right top.

Metal-1 minimum area

The minimum area of metal-1 geometries. The value specified is in square microns and does not have to be on grid.

Via-1 Rules

The via-1 rules define via-1 geometry size and spacing.

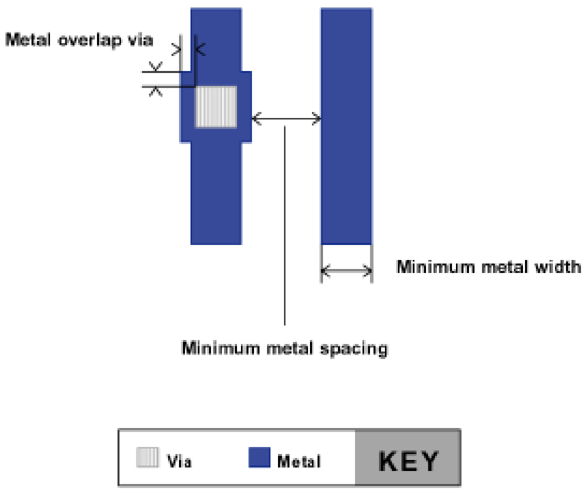


Figure 18

Minimum via-1 width

The exact size of via-1 geometries.

If one value is specified, that value will be used as both the minimum height and minimum width of the via.

If two values are specified, the first value specifies the minimum via width, and the second value specifies the minimum via height: width, height.

Minimum via-1 spacing

The minimum via-1 to via-1 spacing.

Metal-2 Rules

The Metal-2 rules define size, spacing, overlap, and area restrictions on metal-2.

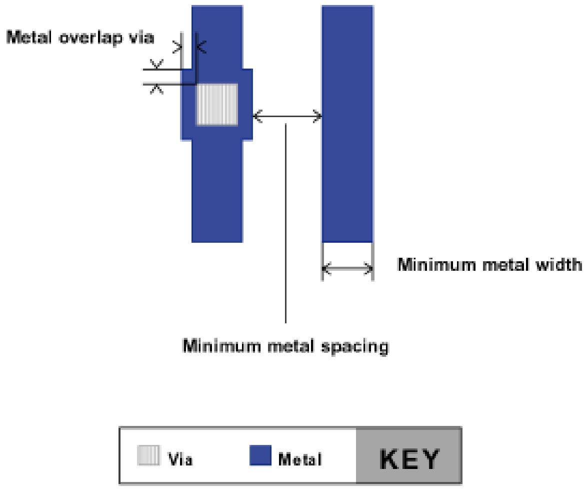


Figure 19

Minimum metal-2 width

The minimum width of metal-2 geometries.

Minimum metal-2 spacing

The minimum metal-2 to metal-2 spacing.

Metal-2 overlap via-1

Minimum metal-2 overlap of via-1. The value entered can take one of two forms: a single value or a list of four values. Positive values result in the metal extending outside of the via. Negative values result in the metal being within the via.

When a single value is specified it is applied to all sides of the metal overlap of the via.

A list of four values specifies the overlaps for each of the four sides of the vias. The list of overlaps have the order left bottom right top.

Metal-2 minimum area

The minimum area of metal-2 geometries. The value specified is in square microns and does not have to be on grid.

Layers Tab

The Layers tab provides access to layer names, mappings, output, and customization.

ProGen/ProView Layer Controls

Just below the row of ProTech tabs are two buttons used to manage layers. The ProGen button allows addition, removal, and renaming of layers, while the ProView button allows addition, removal, and reordering of layers for ProView.

ProGen

Clicking on this button brings up three options.

The Add Layer... option brings up a window from which you can select a new layer to add. (If you are creating a new layer altogether, go to Custom Layers>Add first; it will then appear under the Add Layer... list for selection.)

To rename a layer, select the Rename Layer... option and click on the name of the layer to be renamed. Click OK to be prompted for the new layer name.

To delete a layer, select the Remove Layer... option and select the doomed layer from the list.

ProView

Clicking on this button brings up three options. The New Layer... option prompts for a new layer name.

The Delete Layer... works by offering a list of layers to choose from; select the layer to be deleted and click OK.

The Change Layer Order... opens a list of the layers in their current order; they can be dragged and dropped into any desired order. Click OK when the order is correct.

ProGen Output Layers

This section controls which layers are output. The layer types can be linked to a particular ProView Data Layer and a particular ProView Text Layer from a drop-down list. Set or unset the Output Layer check box to manage output of each layer.

ProView Layers Configuration

The ProView layer information is also arranged in tabular form, with the following columns appearing for each layer:

GDS Data Layer

GDS Data Type

Display Color

Display Fill

Display Outline

Display View

Layout within the ProView layout viewer can be customized through these settings.

Customize ProGen Layers

ProTech supports manipulation of both built-in and customized layers through the interface in this area of the Layers tab. When working with custom layers, the layer name can be added or changed, but when working with built-in layers, the names are fixed. The other elements of the ProTech interface are identical for the two types of layers.

Custom layers

This control provides a way to create new ProGen layers. Layers have the following values that can be set:

Layer name

Cell border spacing rule control

Spacing design rule internal names

Forbidden spacing design rule internal names

Width-dependent spacing design rule internal names

Inhibit spacing design rule internal names

Overlap design rule internal names

Minimum size design rule internal names

Shadow layers

Layers to shadow

Layers that connect

Columns

Layer

Name of custom layer

Controls (at right)

Add

Opens a dialog for creating a new layer and setting layer values. (See “Add ProGen Custom Layer”)

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected, then the remove button has no effect.

Be careful when removing custom layers.

Removing a layer from the list deletes it from the ProTech system. You will have to reenter all values if you delete a layer and then decide you want it.

Built-in layers

This control provides a way to modify characteristics of ProGen built-in layers. The list indicates layers that you have modified. Removing an entry from the list causes it to return to using all built-in default values.

Layers have the following values that can be set:

Cell border spacing rule control

Spacing design rule internal names

Forbidden spacing design rule internal names

Width-dependent spacing design rule internal names

Inhibit spacing design rule internal names

Overlap design rule internal names

Minimum size design rule internal names

Shadow layers

Layers to shadow

Connecting layers

Columns

Layer

Name of ProGen built-in layer

Controls (at right)

Add

Opens a dialog for customizing the characteristics of a ProGen built-in-layer. The dialog will initially fill in the default values for the selected layer in the dialog Layer control selection list.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, the Edit button has no effect. The Edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected, then the remove button has no effect.

Removing a layer from this list does not delete it from the ProTech system. It just means that you are not changing any of the default values.

Layer name

Name of ProGen built-in layer to add/edit. When adding a built-in layer to the list of custom layers, the system will fill in all the built-in layer characteristics once a layer has been selected. When editing a built-in layer previously added to the list of custom layers, you will not be able to change the layer name.

Add ProGen Custom Layer

Add

Layer name

Name of a custom layer to add to ProGen. A custom layer name must not match any existing layer names and cannot contain white space (spaces, tabs, newlines).

When editing a previously created custom layer, the Layer entry will not allow you to change its name.

Cell border spacing

Indicates how ProGen should derive minimum cell border spacing values between geometries and the cell borders. If one of more rule entries exist, the system will first try to base the spacing on the first rule which is assigned valid values. If there are no rule entries or none have valid values, then the first non-rule entry will be used to create the border spacing values.

Controls (at right)

Add

Opens a dialog for adding border spacing definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the Remove button has no effect.

Columns

Type

The type of rule look up ProGen should use to set the border spacing values. It can have one of following values:

none

Do not apply any cell border spacings. This is typically used for layers that contain geometries which must overlap the cell border (any cb_layer layers).

x-axis

Apply minimum cell border spacings in the x-axis only. This results in spacings to the left and right cell border edges but not the top and bottom edges. The value is one half the minimum spacing between geometries on the layer.

y-axis

Apply minimum cell border spacings in the y-axis only. This results in spacings to the top and bottom cell border edges but not the left and right edges. The value is one half the minimum spacing between geometries on the layer.

both axes

Apply minimum cell border spacings in both the x- and y-axis. The value is one half the minimum spacing between geometries on the layer.

rule

Indicates that ProGen should base the spacing on the value associated with the corresponding design rule. A Rule name value must be specified when this option is selected. When more then one rule exists, the first rule that has a valid value is used. If no rules have a valid value, then the first of the above options specified in the Type list is used.

Rule name

Name of a design rule that ProGen will use to look up the minimum border spacing value. This entry is only meaningful when the value of Type is rule. It is ignored for all other types.

Prolific recommends that all border spacing rule names be prefixed with bsp_ (for example: bsp_m1, bsp_ndiff, and bsp_poly). This helps identify it as a border-spacing rule when adding and editing the rule values.

Priority

Priority associated with satisfying the minimum border spacing. The default value guarantees that the spacing will be satisfied. A value in the range of 0 to 5 indicates that spacing should be satisfied if possible, but it is not mandatory. A value in the range of 6 to 9 indicates that it must be satisfied.

Weight

Weight associated with satisfying the minimum border spacing. This can be any positive integer or the value default.

Spacing rules

This is used to enter minimum spacing rule definitions for geometries on this layer.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, then the Edit button has no effect. The Edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

To layer

ProGen will apply minimum spacings between geometries on the main layer (Layer name) and geometries on the To layer. There may be any number of To layer entries and it may be the same as the main layer.

Rule name

One or more names that ProGen will use to look up the minimum spacing values. ProGen looks up each rule in the order listed and the first rule that has a valid value is used to set the minimum spacing.

Rule names cannot have white space (spaces, tabs, newlines) in them. Multiple rule names are separated by a space.

Net type

This indicates when the spacing rule should be applied based on the net connectivity (net type) between geometries. ProGen can be directed to apply spacing rules when the geometries are on the same net (samenets_only or both) or on different nets (diffnets_only or both). The values defined in Connect to layers control net connectivity.

Priority

The priority associated with enforcing minimum spacings; only legal values are 0 through 9. The ProGen default priority guarantees that minimum spacings will be enforced. A priority less than or equal to 5 indicates optional enforcement. A priority greater than or equal to 6 indicates mandatory enforcement.

Optional enforcement means that ProGen will attempt to enforce the spacing if it can, but it will not sacrifice cell size or higher priority rules to satisfy this spacing.

Weight

The weight associated with enforcing minimum spacings. This can be any positive integer or the keyword default.

Forbidden spacing rules

This is used to define forbidden spacing design rule definitions for geometries on the layer.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, the Edit button has no effect. The Edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected, then the Remove button has no effect.

Columns

To layer

ProGen will apply forbidden spacings between geometries on the main layer (Layer name) and geometries on the To layer. There may be any number of To layer entries, and may include the main layer.

Rule name

One or more names that ProGen will use to look up the forbidden spacing values. ProGen looks up each rule in the order listed and the first rule that has a valid value is used to set the spacing.

Prolific suggests the following conventions for forbidden spacing rule names:

Forbidden spacings between geometries on the same layer: layer_fsp. For example: m1_fsp, poly_fsp, and ndiff_fsp.

Forbidden spacings between geometries on different layers: layer1_fsp_layer2. For example:

ndiff_fsp_pdiff, fieldpoly_fsp_gatepoly, and m1_fsp_vssm1.

Rule names cannot have white space (spaces, tabs, newlines) in them. Multiple rule names are separated by a space.

Priority

The priority associated with enforcing forbidden spacings. The ProGen default priority guarantees that forbidden spacings will be enforced. A priority less than or equal to 5 indicates optional enforcement. A priority greater than or equal to 6 indicates mandatory enforcement.

Weight

The weight associated with enforcing forbidden spacings. This can be any positive integer or the keyword default.

Width-dependent spacing rules

This is used to define width-dependent spacing design rule definitions for geometries on the layer.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected, then the Remove button has no effect.

Columns

To layer

ProGen will apply width-dependent spacings between geometries on the main layer (Layer name) and geometries on the To layer. There may be any number of To layer entries and it may be the same as the main layer.

Rule name

One or more names that ProGen will use to look up the width-dependent spacing values. ProGen looks up each rule in the order listed and the first rule that has a valid value is used to set the spacing.

Prolific suggests the following conventions for width-dependent spacing rule names:

Width-dependent spacings between geometries on the same layer: layer_wsp. For example:

m1_wsp, poly_wsp, and ndiff_wsp

Width-dependent spacings between geometries on different layers: layer1_wsp_layer2. For example:

ndiff_wsp_pdiff, fieldpoly_wsp_gatepoly, and m1_wsp_vssm1

Rule names cannot have white space (spaces, tabs, newlines) in them. Multiple rule names are separated by a space.

Priority

The priority associated with enforcing width-dependent spacings. The ProGen default priority guarantees that width-dependent spacings will be enforced. A priority less than or equal to 5 indicates optional enforcement. A priority greater than or equal to 6 indicates mandatory enforcement.

Weight

The weight associated with enforcing width-dependent spacings. This can be any positive integer or the keyword default.

Inhibit spacing when between

Inhibit spacing definitions prevent ProGen from applying minimum spacing rules between Layer 1 and Layer 2 geometries when a geometry on the main layer exists between them.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

Layer 1

One of the layers that would normally have a minimum spacing rule to a Layer 2 geometry.

Layer 2

One of the layers that would normally have a minimum spacing rule to a Layer 1 geometry.

Overlap rules

This is used to specify overlap design rule definitions.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

Overlap layer

The name of the layer being overlapped.

Rule name

One or more names that ProGen will use to look up the overlap values. ProGen looks up each rule in the order listed and the first rule that has a valid value is used to set the overlap.

Prolific suggests that overlap rule names follow the convention: layer1_ov_layer2. For example:

m1_ov_v1, ndiff_ov_cndiff, and poly_ov_cpoly.

Rule names cannot have white space (spaces, tabs, newlines) in them. Multiple rule names are separated by a space.

Preferred overlap rules

This is used to specify preferred-overlap design rule definitions. Preferred overlaps are low priority rules that ProGen will try to satisfy without impacting other cell compaction requirements such as minimizing the cell size.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

Overlap layer

The name of the layer being overlapped.

Rule name

One or more names that ProGen will use to look up the overlap values. ProGen looks up each rule in the order listed and the first rule that has a valid value is used to set the overlap.

Prolific suggests that overlap rule names follow the convention: pref_layer1_ov_layer2. For example:

pref_m1_ov_v1, pref_ndiff_ov_cndiff, and pref_poly_ov_cpoly

Rule names cannot have white space (spaces, tabs, newlines) in them. Multiple rule names are separated by a space.

Priority

The priority associated with enforcing preferred overlaps. The default value results in ProGen using the built-in preferred priority.

Weight

The weight associate with enforcing preferred overlaps. This can be any positive integer or the keyword default.

Minimum width rules

This is used to enter minimum width rule definitions for a layer's geometries. The minimum widths are often used to create the initial size of shapes for just about all objects (such as wires, contacts, vias, diodes, and ports). The values are also used to prevent the ProGen compactor from collapsing the geometries smaller than their minimum design rule sizes.

All non-shadow layers should have at least one minimum width rule defined.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

Rule name

The rule name that ProGen will use to look up the minimum width values. ProGen looks up each rule in the order listed and the first rule that has a valid value is used to set the minimum width.

Prolific suggests that minimum size rule names follow the convention layer_min, where layer is the name of the main layer being defined.

Rule names cannot contain white space, such as spaces, tabs, and newlines.

Priority

The priority associated with enforcing minimum widths. The ProGen default priority guarantees that minimum widths will be enforced. A priority less than or equal to 5 indicates optional enforcement. A priority greater than or equal to 6 indicates mandatory enforcement.

Weight

The weight associate with enforcing minimum widths. This can be any positive integer or the keyword default.

Preferred minimum width rules

This is used to enter preferred minimum width rule definitions for geometries on this layer. Preferred minimum widths are low priority rules that ProGen will try to satisfy without impacting other cell compaction requirements such as minimizing the cell size.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

Rule name

The rule name that ProGen will use to look up the preferred minimum width values. ProGen looks up each rule in the order listed and the first rule that has a valid value is used to set the preferred minimum width.

Prolific suggests that minimum size rule names follow the convention pref_layer_min, where layer is the name of the main layer being defined.

Rule names cannot contain white space (including spaces, tabs, and newlines).

Layers that shadow

This lists the shadow layers of the primary layer. A geometry will be created on each shadow layer every time a non-shadow geometry is created on this layer.

Shadow layers

Whenever a geometry is created on a primary layer, ProGen creates geometries on all the shadow layers of the primary layer. Shadow layer geometries are exactly the same as the corresponding primary layer geometries except that overlap design rule values are applied (if they are on a different layer). During compaction, shadow layer geometries follow the edges of the corresponding primary layer geometries.

ProGen does not create shadow layer geometries of shadow layer geometries. So if layer "A" shadows layer "B," and layer "B" shadows layer "C," the creation of a geometry on layer "C" results in a shadow geometry on layer "B", but not on layer "A". If you want "A" and "B" shapes then they must both be listed as shadow layers of "C".

Shadow layers do not inherit the net number from design layers. They only propagate nets to the same shadow layer.

Connection forces are never applied to shadow layers, unless a geometry has a specified MUST_CONNECT that creates the connection forces.

“Layers that shadow” vs. “Layers to shadow”

Both of these provide the same basic functionality. When one layer shadows many other layers, it is probably easier to specify Layers to shadow for the one layer then to go to all the other layers and add the single layer to the Layers that shadow list of each.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

Layers

Indicates the layers that shadow the primary layer. A geometry will be created on every shadow layer listed.

Layers to shadow

This lists the layers that this layer shadows. A geometry will be created on this layer for every non-shadow geometry created on any of the layers in this list.

Shadow Layers

Whenever a geometry is created on a primary layer, ProGen creates geometries on all the shadow layers of the primary layer. Shadow layer geometries are exactly the same as the corresponding primary layer geometries except that overlap design rule values are applied (if they are on a different layer). During compaction, shadow layer geometries follow the edges of the corresponding primary layer geometries.

ProGen does not create shadow layer geometries of shadow layer geometries. So if layer "A" shadows layer "B," and layer "B" shadows layer "C," the creation of a geometry on layer "C" results in a shadow geometry on layer "B", but not on layer "A". If you want "A" and "B" shapes then they must both be listed as shadow layers of "C".

Connection forces are never applied to shadow layers, unless a geometry has a specified MUST_CONNECT that creates the connection forces.

Shadow layers do not inherit the net number from design layers. They only propagate nets to the same shadow layer.

“Layers that shadow” vs. “Layers to shadow”

Both of these provide the same basic functionality. When one layer shadows many other layers, it is probably easier to specify Layers to shadow for the one layer then to go to all the other layers and add the single layer to the Layers that shadow list of each.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

Layers

Indicates the layers that this layer shadows. Whenever a geometry is created on any of these layers, the system will create a shadow geometry on the primary layer.

Connect to layers

This entry controls how ProGen propagates net connectivity between layers and how it determines to create connections between different geometries.

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected then the Edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the Remove button has no effect.

Columns

Layers

Indicates the secondary layer to which a connection relationship is being defined. Typically, there is an entry to indicate that geometries on the same layer are connected.

Propagate net

When geometries are considered connected (see Dimension), this determines if they are also considered to be on the same net. If so, net information will be propagated between them. Values are

yes

Connected geometries are on the same net and net information will be propagated between the shapes.

no

Do not consider the connected geometries to be on the same net. Do not propagate net information between the shapes.

Shadow layers do not inherit the net number from design layers. They only propagate nets to the same shadow layer.

The Propagate net setting is only part of configuring net connection. See the discussion of the Dimension setting for additional information on how connections are determined.

Add forces

When geometries are considered connected (see Dimension), this determines if ProGen should add "forces" to keep them connected during compaction. Typically, geometries on the same conceptual layer (such as m1 and cb_m1) should be connected while geometries on different layers (such as m1 and v1) should not be connected. The following values are available:

yes

Create connections between the geometries.

no

Do not create connections between the geometries.

Dimension

This indicates when geometries should be considered connected based on their physical placement. The following choices are available:

overlapping

Consider geometries connected if they are overlapping by one or more units. Abutted edges are not considered overlapping.

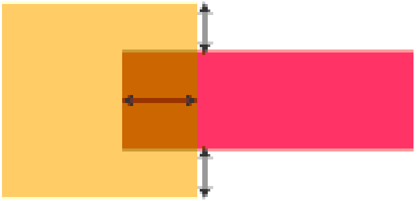


Figure 20: One example of overlapping geometries.

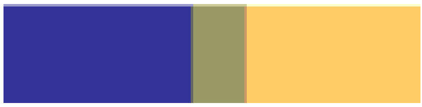


Figure 21: A "follow" example of overlapping geometries.

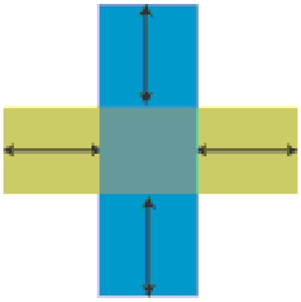


Figure 22: Another example of overlapping geometries.

touching

Consider geometries connected if they are overlapping or if they have abutted edges. They are not considered connected if just their corners are touching.

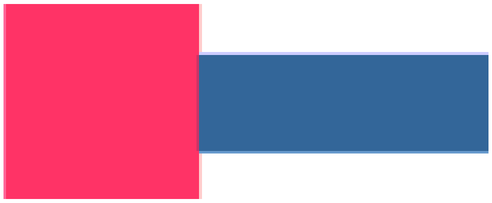


Figure 23: Touching geometries.

point

Consider geometries connected if they overlap, have abutted edges, or if just their corners are touching.

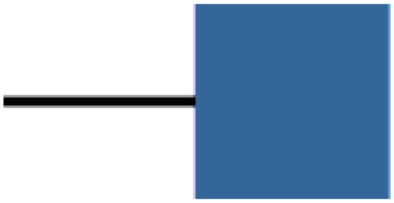


Figure 24: Geometries connected at a single point.

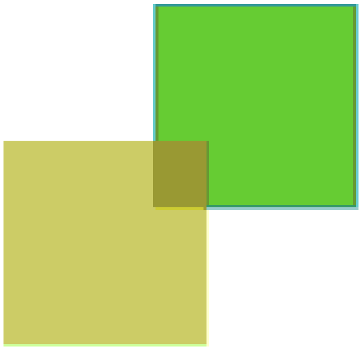


Figure 25: These geometries only overlap at a corner and would not receive connecting forces.

dontconnect

Never consider geometries connected.

Priority

The priority associated with enforcing connections. The ProGen default priority guarantees that connections will be enforced. A priority in the range of 0 to 5 indicates optional enforcement. A priority in the range of 6 to 9 indicates mandatory enforcement.

Weight

The weight associated with enforcing connections. This can be any positive integer or the keyword default.

Dense end-of-line spacing rules

Controls (at right)

Add

Opens a dialog for adding rule definitions.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected then the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected then the remove button has no effect.

Columns

To layer

The name of the layer that dense end-of-line spacings are checked against. This is typically the same layer as the customer layer name.

Rule name

One or more names that ProGen will use to look up the dense end-of-line spacing values. ProGen looks up each rule in the order listed, and the first rule that has a valid value is used to set up the values.

Prolific suggests that following rule naming convention:

When the rule applies to shapes on the same layer:

layer_dense_end_of_line_spacing

When the rule applies to shapes on different layers:

layer1_dense_end-of-line-spacing_layer2

Examples:

```
poly_dense_end_of_line_spacing

m1_dense_end_of_line_spacing_cb_m1
```

Example 2

Rule names can not have white space (spaces, tabs, newlines) in the them. Multiple rule names are separated by a space.

Cell Template Tab

Cell Template settings

Cell Templates list

This entry is used to define one or more cell templates for the technology.

There must always be at least one cell template in this list. Cell templates define the cell height, routing grids, power and ground rails, wells, ties, and other architecture requirements.

Cell Specification of Cell Templates

The cell template that any given cell will use can be specified in the .cel file using the following Tcl statement:

```
pro_set celltemplate dbname CellTemplate
```

The `CellTemplate` value must match one of the entries in the Cell templates list.

When this statement does not exist in the cell file, the system will use the Default cell template.

Controls (at right)

Add

Opens a dialog for adding new cell template definitions. When an entry is selected in the list, the values of the selected entry will be copied to the add dialog.

Edit

Opens a dialog for editing selected entries in the list. If no list entry is selected, then the Edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is selected, then the remove button has no effect.

When the entry being removed is also the Default cell template, the system will make the resulting first entry in the list the default. When the last entry is removed, the Default cell template value is cleared.

Columns

Cell template name

Name of cell template.

Default cell template

This defines the default cell template for the technology. Valid values are any cell templates listed in the Cell templates list. When no value is given, the first entry in the Cell templates will be used.

ProGen uses the default cell template for all cells that don't contain a cell template definition statement in their .cel file.

The following Tcl statement can be included in any .cel file to indicate which cell template to use:

```
pro_set celltemplate dbname CellTemplate
```

The CellTemplate value must match one of the entries in the Cell templates list.

Cell Template setttings

There are many options available when creating or modifying a cell template.

Cell template name

This defines the name to associate with the cell template. The name is case sensitive and must not contain any white space. The name is used to define the technology default cell template and optionally in cell .cel files to indicate a specific cell template for the cell.

When adding a new cell template, the name will be added to the Cell templates list. Duplicate names are not allowed.

When editing an existing cell template, the name cannot be changed.

Additional values file

This entry allows the specification of a file which will be loaded by ProGen to define any other cell template values. This can range from a simple value to a completely new cell architecture.

Cell height in grids

The desired height of the final cell in y-grids. This must be a positive integer value. The system derives the final height by multiplying this value and the Routing grid in Y.

Routing grid in X

The size of the routing grid with respect to the x-axis in microns. The cell border left and right edges are snapped to this grid, the ports are placed in X with respect to it, and any other geometries that have x-axis gridding values are placed with respect to this value.

Setting the value to 0.0 turns it off.

Routing grid in Y

The size of the routing grid with respect to the y-axis in microns. The cell border top and bottom edges are snapped to this grid, the ports are placed in Y with respect to it, and any other geometries that have y-axis gridding values are placed with respect to this value.

Setting the value to 0.0 turns it off.

Offset routing grid from origin in X

The amount to shift the X-grid to the right of the origin in microns.

The default value of 0.0 results in the grid going through the origin in X.

This value is often set to half of the Routing grid in X so that the ports can be placed within a half grid of the cell border left and right edges.

Offset routing grid from origin in Y

The amount to shift the Y-grid upward from the origin in microns.

The default value of 0.0 results in the grid going through the origin in Y.

This value is often set to half of the Routing grid in Y so that the ports can be placed within a half grid of the cell border top and bottom edges.

N-transistor maximum width

The maximum width of n-transistors in microns. This value is used by ProGen to determine when to automatically fold n-transistors.

P-transistor maximum width

The maximum width of p-transistors in microns. This value is used by ProGen to determine when to automatically fold p-transistors.

N-Well type

Controls the type of n-well geometries that will be placed in the cells. The following describes the supported types:

none

Do not create an n-well based on the values specified in the cell template dialog. Assuming that n-well is still needed, it should be defined in the Additional values file.

straight

Creates a simple n-well that has a straight bottom edge through the cell. This is the most common n-well type used in standard cell libraries and hence is the default.

notched

Creates an n-well that can float up and down in the non-edge regions of the cell to adjust for various transistor sizes. The left and right edges of the cell stay at the n-well bottom edge Y value to assure cell-to-cell abutment consistency.

N-well bottom edge Y value

Controls the desired final location of the n-well bottom edge in cells. The value is in microns and with respect to the cell bottom edge. The exact meaning depends on the value of n-well type:

none

There is no n-well. The value is ignored.

straight

A single n-well edge spans the cell width and will be at this value.

notched

The n-well left and right edge pieces will have their bottom edges at this value. The center piece may float.

N-well cell border overlaps

Defines the amount that the n-well should overlap the left, right, and top edges of the cell border. This must actually be entered as a list of four values where the positions are left bottom right top.

The bottom value is ignored.

Add p-well

Instructs the system to add p-well shapes in the cell template that are complimentary to the n-well. The p-well has the same cell border overlaps as the n-well, and the top edge abuts the n-well bottom edge. It can have the following values:

yes

Add p-well shapes to cell template.

no

Do not put p-well shapes in cell template. This is the default since p-well is typically a derived layer.

Rail type

Controls the type of power and ground rails that will be placed in the cells. The following describes the supported types:

none

Do not create rails based on the values specified in the cell template dialog. Assuming that rails are still needed, they should be defined in the Additional values file.

m1

Creates metal-1 rails. These are the most common type used in standard cells and, therefore, are the default.

m2

Creates metal-2 rails. This will result in the metal-1 wires from the transistors with via-1 at the rails to connect.

Rail width

Defines the width of the rails in microns.

Rail center offset in Y from cell border

The default placement of rails is for them to straddle the top and bottom cell edges. The center line of the rail in Y is coincident with the corresponding cell border edges.

This defines the amount to shift the rails from the above default. Positive values shift the rails inward, negative values shift them outward. All values are in microns.

Vss rail text

Defines text to put on the ground rail.

Vdd rail text

Defines text to put on the power rail.

Add ties

Defines the ties to have the system put in the cell template. When no ties are entered, no ties will be in the final cell. Any number of ties may be specified. The following choices are available:

straddle bottom

Add a substrate tie that straddles the bottom edge of the cell border.

straddle top

Add a well tie that straddles the top edge of the cell border.

soft bottom

Add a soft substrate tie that straddles the bottom edge of the cell border. A soft tie is initially just diffusion and implant. The system adds contacts and metal-1 after compaction based on the space availability in the final cell.

soft top

Add a soft well tie that straddles the top edge of the cell border. A soft tie is initially just diffusion and implant. The system adds contacts and metal-1 after compaction based on the space availability in the final cell.

corner

Indicates that corner ties may be specified in .cel input files.

The specification in the .cel file for corner ties, whether built-in or custom, is:

pro_set celltemplate_info add {objects {corner_ties ties...}}

where ties... indicates one or more corner tie object names.

The following eight object names reference corner ties which are built into the ProGen system and may be specified as ties:

cornertie_sub_lef_horz

cornertie_sub_rig_horz

cornertie_sub_lef_vert

cornertie_sub_rig_ver

cornertie_wel_lef_horz

cornertie_wel_rig_horz

cornertie_wel_lef_vert

cornertie_wel_rig_vert

The cell file may also reference custom defined corner ties defined in the cell file or in the technology files.

The following four ties are all substrate p-diffusion corner ties. They are placed in the defined cell template corner with the corresponding orientation. They assume that there are minimum diffusion area rules which will result in the tie having either a horizontal or vertical orientation:

corner bot lef horz

corner bot rig horz

corner bot lef vert

corner bot rig vert

The following four ties are all n-well n-diffusion corner ties. They are placed in the defined cell template corner with the corresponding orientation. They assume that there are minimum diffusion area rules which will result in the tie having either a horizontal or vertical orientation:

corner top lef horz

corner top rig horz

corner top lef vert

corner top rig vert

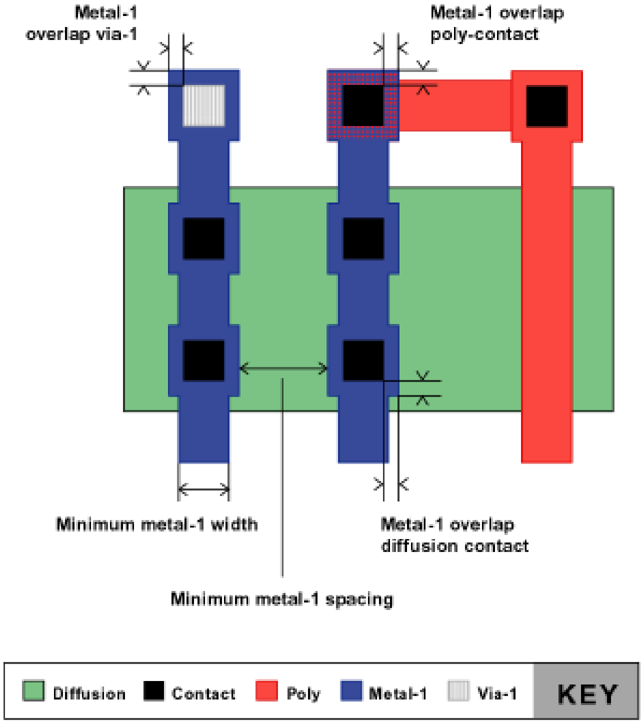


Figure 26

Router Tab

The settings on the Router tab are used to configure the ProPPR router, which is discussed in more detail in Selecting, Configuring and Using ProPPR.

Router configurations

This entry is used to define one or more router configurations for the technology.

There must always be at least one router configuration for the technology. Router configurations define values for routing contraints such as routing layers, cost per layer, tracks per channel, and other information.

Cell specification of router configuration

The router configuration that any given cell will use can be specified in the .cel file using the following Tcl statement:

pro_set cell dbproppr RouterConfiguration

The RouterConfiguration value must match one of the entries in the Router configurations list. When this statement does not exist in the cell file, the system will use the Default router configuration.

Columns:

Router configuration name

Name of router configuration.

Controls:

Add

Pops open a dialog for adding new router configuration definitions. When an entry is selected in the list, the values of the selected entry will be copied to the add dialog.

Edit

Pops open a dialog for editing seelected entries in the list. If no list entry is selected, the edit button has no effect. The edit dialog can also be opened by double-clicking on a list entry.

Remove

Removes selected entries from the list. If no entry in the list is slected, the remove button has no effect.

When the entry being removed is also the Default router configuration, the system will make the resulting first entry in the list the default. When the last entry is removed, the Default router configuration value is cleared.

Default configuration

This defines the default router configuration for the technology. Valid values are any configurations listed in the Router configurations list. If no value is given, the first entry will be used.

ProGen uses the default router configuration for all cells that don't contain a router configuration definition statement in their .cel file. The following Tcl statement can be included in any .cel file to indicate which router configuration to use:

```
pr_set cell dbproppr RouterConfiguration
```

The RouterConfiguration value must match one of the entries in the Router configurations list.

Parameter Generation

The Parameter Generation section allows configuration of where ProGenesis runs to create router values.

Temporary working directory

This is the directory in which ProTech will run ProGen to create the router values. ProTech will create the directory if it does not already exist.

Delete temporary working directory

Setting this option to yes instructs ProTech to delete the temporary directory after it has completed parameter generation. It is not necessary to delete the directory, which will contain a log, .cel files, and agd files, which may be useful if parameter generation fails.

Setting the option to no leaves the directory intact after parameter generation has finished.

Parameter Generation Log

Run Parameter Generation Script

ProMigrate Tab

Transistor scale factor

Scale factor to apply to all transistor widths when migrating from the source technology to the new technology. The default value is 1.0 (no scaling is done).

To apply a 20% decrease to all transistor widths, set the scale factor to 0.80. (So: migrating a transistor with a gate width of 0.65 would result in a transistor with a 0.52 width.)

Source data GDS layers

Identifies the ProMigrate layers to which GDS layers in the source technology are mapped. The system defaults to the GDS values defined for the destination layers (found on the ProTech Layers tab) when no value for one or more layers is defined. In other words, source GDS layers which are the same as the destination technology layers need not be specified.

There are 5 values per entry:

Layer

Data GDS

Data Type

Text GDS

Text Type

Layer is the layer for which the GDS values are being defined.

A GDS layer is defined by a layer number and type number. The values of Data GDS and Data Type map geometric data (non-text) on the corresponding GDS layer to the ProMigrate Layer.

The values of Text GDS and Data Type map text data on the corresponding GDS layer to the ProMigrate Layer.

Non-port text to ignore

List of text that should not be considered a port and will not receive a port object in the destination layout. Each text entry is actually a Tcl regular expression, so you may use wildcards and such in each entry.

Ports usually have via and metal-2 added, are gridded, and have port text. In the case of power and ground text, the port is actually part of the cell-template. Therefore, the power/ground text is typically listed as non-port text.

Scale all data by

Scale factor to apply to all data in source layout. A value of 1.0 (the default) results in no scaling. When migrating to a technology with smaller feature sizes, you should leave this value as 1.0. The system will automatically adjust values to the new feature sizes and ProGen will compact the data according to the destination values.

When migrating to a technology with larger feature sizes, it is suggested to set the scale so the data enlarges initially. Then the system will future-adjust the values during compaction.

Offset all data in X by

Offsets the initial layout of the destination data in the x-axis by the specified amount in microns from its position in the source layout. If a geometry has a lower-left vertex at 100.0 in the source layout, then setting the offset to 50.0 will result in the migrated data initially be placed at coordinate 150.0 in X. The compactor may then move this to make the overall cell smaller.

Offset all data in Y by

Offsets the initial layout of the destination data in the y-axis by the specified amount in microns from its position in the source layout. If a geometry has a lower-left vertex at 100.0 in the source layout, then setting the offset to 50.0 will result in the migrated data initially be placed at coordinate 150.0 in Y. The compactor may then move this to make the overall cell smaller.

Rotate all data

Rotates the source data 90 degrees before migrating.

The ProGen system general creates cells such that the power and ground rails are horizontal. If the source data has vertical rails, then this value should be turned on.

Ignore ties in source

Instructs ProMigrate to ignore tie data in the source layout. It then depends on your setup for the destination technology to create ties in the migrated data. Typically, ties are defined as part of the ProGen cell-template.

Source layers to ignore

List of layers in the source layout that should not be processed by migration. This allows you to ignore data on specific layers. It is recommended to leave this list empty.

For example: ndiff_gate will map to a transistor object, while cpoly_hor will map to a horizontally oriented poly-contact. When a layer appears in this list, those corresponding objects will not be created.

Contact width

The exact size of all diffusion and poly contacts.

If one value is specified, that value will be used as both the minimum height and minimum width of the contact. If two values are specified, the first value specifies the minimum contact width, and the second value specifies the minimum contact height: width, height.

Contact spacing

The minimum contact-to-contact spacing.

Metal-1 overlap contact

Minimum metal-1 overlap of diffusion-contacts. The value entered can take one of the two forms: a single value or a list of four values. Positive values result in the metal extending outside of the contact. Negative values result in the metal being within the contact.

When a single value is specified it is applied to all sides of the metal overlap of the contact.

A list of four values specifies the overlaps for each of the four sides of the contacts. The list of overlaps have the order: left, bottom, right, top.

Preferred metal-1 overlap of diffusion-contacts.

Value specification is optional.

When a preferred overlap is specified ProGen will attempt to achieve the overlap during compaction but it will not impact the final cell size in order achieve the overlap. The minimum overlap rules will still be enforced regardless of whether ProGen can achieve the preferred overlaps or not.

The specified value entered can take one of two forms: a single value or a list of four values. Positive values result in the metal extending outside of the contact; negative values result in the metal being within the contact.

When a single value is specified it is applied to all sides of the metal overlap of the contact.

A list of four values specifies the overlaps for each of the four sides of the contacts. The list of overlaps have the order: left, bottom, right, top.

Setting Up A Technology Area

Q: What is the best approach to create files for a new technology?

The best way to create a new technology is to start with a copy of an existing, similar technology (you can avoid a lot of typing). You may want to edit the following files; the most common changes are indicated.

Filename	Description	Typical Changes
progen.db	Main technology file	Advanced settings
progen.tcl	Advanced settings	Change general system options if needed. Examples include changing the compaction order, setting m2 usage, setting bent gates on or off, or altering the vertex grid.
prospin.tcl	Cell-specific overrides (usually empty, initially)	
stdcell.tcl	Cell template requirements (if any)	Advanced settings for cell template or custom template which can't be set in progen.db

Table 1: Technology Files

Q: What is the best approach for experimenting with layout architectures?

Customers often wish to experiment with different layout architectures to decide which is best for their requirements. Once a technology has been set up, the only changes typically made are to set different options in the options.tcl file, and to modify the basic cell architecture in the stdcell.tcl file.

One example of a common experiment would be turning on transistor bent gates in the options.tcl file. Another example would be changing the cell height from 10 grids to 9 grids in the stdcell.tcl file.

Because the .cel files do not need to be touched when you experiment with a change to the layout architectures, you can just make the changes to the technology files, and then rerun the existing .cel files.

Also note that because all the technology files use the Tcl language, the customer can create new options and set up the technology files so that they can be run with one architecture versus another. For example: the following commands run an inverter once with a 10 grid cell height, and once with a 9 grid cell height. The local technology has a user-defined flag named arch with values of 9grid and 10grid. This flag sets the cell height values.

```
progen -t tech/progen.tcl -i inv.cel -o inv10.agd -f arch=10grid

progen -t tech/progen.tcl -i inv.cel -o inv9.agd -f arch=9grid
```

Example 1: Experimenting with layout architectures using user-defined flags

How you evaluate the results of your experiments is completely up to you and your design requirements. You may wish to select the layout architecture that yields cells with the smallest area, or one that provides the most free routing area, or the one that uses the least m2. The choice is up to you!

Technology Settings Q&A

Q: The dmwire produces diffusion contacts that are larger than the transistors to which they connect. How do I change this?

In mytech/options.tcl, set dmwire_align. (In this case, it was set to 0, so change it to 1, delayed, or delayed_reverse.) Use a statement like this one:

```
   pro_set options dmwire_align delayed_reverse
```

If you use either delayed or delayed_reverse, set the compaction order to include at least two y-passes, such as:

```
   pro_set options compaction_order {y_skip_grid_2d x_2d y}
```

Options for dmwire_align

`0`

never create forces that keep the diffusion contact within the transistor source/drain region

`1`

create the forces and activate them immediately

delayed

create the forces and activate them on the second y-pass

delayed_reverse

save as delayed, except also add forces that pull the diffusion contact as close to the edge of the cell as possible (away from the transistors)

Q: How do I encourage or discourage the use of diffusion routing where the diffusion contact ends up wrapped around the gate-extension of the transistors (above the p-transistors or below the n-transistors)?

Use dmwires for power diffusion contacts.

Pull the diffusion contacts towards the rails as much as possible on the first y-compaction:

```
   pro_set options dmwire_align delayed_reverse
```

Pull the transistors towards the center of the cell as much as possible (in mytech/options.tcl):

```
   pro_set option npPullTransistorsOutward 0
```

To discourage this behavior, set

```
   pro_set option npPullTransistorsOutward 1
```

To make sure that the diffusion contact stays within the source/drain region of the transistor, set:

```
   pro_set options dmwire_align 1
```

...or don't use dmwires at all.

ProTech FAQ

Can the tool handle special gate pattern requirements?

Yes.

How DFM-friendly is the setup?

Very friendly. Just enter DFM rules in ProTech.

How do I display a certain node name in the layout without declaring it as a port?

In ProTech>Options tab, you can set option "Text internal nodes" to yes to get all internal node names written in agd output. However, the option currently only works with non-ProSticks cells and the internal node names are written in m1 text layer.

How do I set gate poly pitch restriction?

The 260 nm gate poly pitch is enforced by applying gridding forces on gate poly. In progen.tcl, search for "grid_x_track_callback", you will find "poly_gate" appeared right after the first occurrence of "grid_x_track_callback". Search the same text again will bring you to the procedure of "grid_x_track_callback".

Inside the procedure, the last two lines are as follows:

```
tech_grid_force $id x $name lo 6 1 $xgrid $offset_lo $offset_lo

tech_grid_force $id x $name hi 6 1 $xgrid $offset_hi $offset_hi
```

Number "6" means priority 6, which is higher than area minimization force priority. If 260nm is not required any more, you should lower this number to be less than or equal to 5.

In addition, since gate poly now can be placed in a region, you can specify Poly forbidden spacing in the ProTech GUI to block out unacceptable regions. Click Info then Poly forbidden spacing for more details.

Why has the cell template changed to be made in straight.ct instead of progen.db?

You need to use the Tcl template file only if the entries in the ProTech template form are not sufficient to completely define the desired template. The Tcl template file can contain only the complete template definitions or the additional definitions that are not covered in ProTech template for. In the case of the complete template definitions, the Tcl template definitions will override those in the ProTech template form.

ProGen Guide

Adding Layers To ProGen

This document describes everything you need to know about adding layers to the ProGen system.

There are many reasons to add new layers. They almost always involve needing tighter control over the behavior of objects during compaction.

When defining a new layer there are four groups of information which should be specified:

- graphical
- design rule
- compaction
- GDS output

Graphical Information

The graphical specification provides ProGen with information on how to present the layer when ProGen is run graphically. Layers have the following graphical characteristics:

- drawing turned on or off
- fill pattern
- fill color
- outline color
- drawing order

Each is controlled by setting values associated with keywords. The following are the keywords and the values which can be defined:

draw *value*

Indicates if a layer should be drawn when running ProGen graphically. This does not affect the layer being output in AGD. Valid values are:

`on` - Draw layer (default if draw not specified)

`off` - Turns off drawing of layer

fill *color*

ProGen draws geometries on the corresponding layer filled in the color specified by color. If this keyword is not specified then geometries are drawn with no fill. Color can be specified in two forms:

- Any of the valid textual names for a color defined in the X11 server's database file, such as Blue or BlanchedAlmond. The location and name of the server database file is system dependent but can often be found in /usr/X11/lib/rgb.txt.
- A numeric specification of the red, green, and blue intensities to use to display the color of one of the forms:

Format	Description
#RGB	4-bit colors
#RRGGBB	8-bit color
#RRRGGGBBB	12-bit color
#RRRRGGGGBBBB	16-bit color

Table 1

Each R, G, and B value represents a single hexadecimal digit. When fewer than 16 bits are provided for each color, they represent the most significant bits of the color.

The following examples specify the same color:

```
{fill CornflowerBlue}

   {fill #6495ed}  Decimal: R=100, G=149, G=237
```

stipple *pattern*

ProGen will stipple the fill pattern of the geometries on the corresponding layer with the stipple pattern pattern. If this keyword is not defined then the fill is drawn solid. If the fill option has not been specified then there is no effect. A specified pattern must use one the following forms:

Value	Description
@filename	File containing a valid X11 bitmap description. Prolific provides many bitmap pattern files already in the $PROLIFIC/lib/proview/bitmaps directory.
name	One of the predefined stipple pattern names: • gray75 (75% gray, 3 of 4 bits on) • gray50 (50% gray, 2 of 4 bits on) • gray25 (25% gray, 1 of 4 bits on) • gray12 (12.5% gray, 1 of 8 bits on)

Table 2

The following are examples of stipple pattern definitions:

```
{stipple @tictactoe.bmp} - A bitmap file I created
   {stipple gray12} - System-defined bitmap
```

outline *color*

The color to outline geometries on the layer. If this keyword is not specified then geometries are drawn with no outline. The format of color is the same as for the "fill" keyword.

draworder *integer*

Layers are drawn one after the other on top of each other. Geometries drawn on a layer which are drawn later then another layer may obscure the geometries on the earlier-drawn layer. The order in which the layers are drawn is based on the "draworder" keyword value. The larger the number relative to other layers, the later the layer is drawn.

The number may be anywhere in the range from 0 to 40000. If this keyword is not defined then the layer is placed at the end of the drawing order. Layers defined after this layer which also don't define this keyword will be on top of this layer.

Design Rule Information

Geometries that are created on a given layer typically inherit the design rules associated with that layer. Layers may have any of the following characteristics defined:

- spacing rules
- overlap (enclosure) rules
- cell border spacing
- minimum size geometries on the layer
- maximum size geometries on the layer

spacing *to_layer rule_list* [flag]

The spacing keyword causes ProGen to automatically create spacing forces between geometries on the corresponding layers. The spacing force will be applied between geometries on the master layer and the layer defined by to_layer.

The rule_list is a list of one or more rule names which ProGen will use as the spacing value; the first rule in the list which returns a non-null value will be used.

The [flag] is an optional value which indicates if the spacing force should be applied to geometries on the same or different nets. The valid values are:

Value	Description
`diffnets_only`	Apply spacing if geometries only on different nets
`samenets_only`	Apply spacing if geometries only on same nets
`both`	Always apply spacing (default)

Table 3

overlap *over_layer rule_list*

The overlap keyword allows you to indicate which layers have overlap rules of other layers. The system will automatically apply overlap rules when creating shadow layers.

The rule_list is a list of one or more rule names which ProGen will look-up to use as the overlap value. The first rule in the list which returns a non-null value will be used.

borderspacing *value*

The border spacing keyword directs ProGen to automatically apply cell border spacing rules to geometries on the corresponding layer. The border spacing rule is one-half the layer spacing rule. The argument value may have one of the following values:

Value	Description
0	No cell border spacing forces created
1	Create cell border spacing forces from all cell sides (same as XY)
XY	Create cell border spacing forces from all cell sides (same as 1)
X	Create left and right cell side spacing forces
Y	Create top and bottom cell spacing forces

Table 4

Most layers don't actually need to have a border spacing defined. A layer which has another layer shadowing it does not need border spacing forces when the shadow-layer has the border spacing forces.

min_size *rule*

Tells ProGen the name of the rule to use to get the minimum size of a geometry on the layer. This is usually used to place minimum size forces on geometries to prevent them from collapsing during compaction.

max_size *rule*

Tells ProGen the name of the rule to use to get the maximum size geometry on the layer. This is very rarely specified.

Compaction

Compaction directives provide information which contribute to how geometries on the layer behave. The following information may be set:

- shadow layers
- auto-connection
- area minimization priority
- area minimization weight

shadowlayers *layers*

Defines layers which will shadow the corresponding (master) layer. A shadow layer is created whenever the master layer is created and the edges of the shadow layer follow the master layer. Layers is a list of one or more layers which should shadow the master layer. Overlaps are automatically created when overlap rules have been defined using the layer overlap keyword.

The following is an example of the use of shadowlayers:

```
{shadowlayers m1_all m1_cb}
```

shadowlayer_of *layers*

Defines layers which the master layer will shadow. This has the opposite affect of the "shadowlayers" keyword. Layers is a list of one or more layers which the master layer will shadow. Overlaps are automatically created when overlap rules have been defined.

A `shadowlayer_of` example:

```
{shadowlayer_of m1_all m1_cb}
```

```
connect to_layer [net_prop] [forces] [dim] [priority] [weight]
```

Causes geometries on the master layer to automatically connect to geometries on the `to_layer` whenever they start out overlapping.

The value of net_prop directs the system to propagate net information between geometries when they start out overlapping. It may have one of the following values:

Value	Description
1	Propogate net information (default and typical value)
0	Don't propgate net information

Table 5

The `forces` value indicates if connection forces should actually be created between the objects. This is typically turned on. Turning it off is a tricky way to get the system to propagate the net numbers without actually connecting the shapes. The value of `forces` may be:

Value	Description
1	Create connection forces (default and typical value)
0	Don't create connection forces

Table 6

The *dim* indicates when connection forces are created between geometries on the layers. The value of *dim* can be any one of the following:

Value	Description
overlapping	Create forces if geometries actually overlap
touching	Create forces if sides are overlapping or just touching; default value if not defined
point	Create forces even if corners are just touching
nottouching	Create connection forces if they are not touching
dontconnect	Don't create connection focres regardless

Table 7

You may optionally assign priorities and weights to the connection forces when they are created with the `priority` and `weight` values. The system default connection values are sufficient to hold geometries together during compaction. Changing the values is usually done to lower the values so the system will try to keep things connected if possible but won't insist on it.

pro_set option width_dependent_end_of_line_spacing [on|off]

```
   layer layer-name
   min_width microns
   max_width microns
   spacing microns
   priority 7
   weight 1
```

This option is used to define width-dependent end-of-line spacings to other geometries on the same layer. In otherwords, the spacing from the route end-of-line is dependent on the width of the route.

The spacing rule is applied when the width of the route is greater than or equal to the minimum width (min_width) value and less than the maximum width (max_width) value.

when (min_width >= width < max_width) apply spacing

This allows you to have multiple ranges of values for the same layer.

Parameters:

on|off - Turns option on or off
layer - ProGen layer name (no default)
min_width - Minimum width of route for spacing to be applied
max_width - Maximum width of route for spacing to be applied
spacing - Spacing design rule value in microns (no default)
priority - Priority of spacing forces (default = 7)
weight - Weight of spacing forces (default = 1)

The block of layer parameter values may be specified as many time as is needed.

To change the compaction pass on which the the option is active, use the following command:

```
pro_set option width_dependent_end_of_line_spacing_start_index pass
```

`pass` = the compaction pass number

Examples:

Set width-dependent eol-spacing for m1 routes

```
pro_set option width_dependent_end_of_line_spacing {on

{layer      m1

min_width  0.12

max_width  0.32

spacing    0.16
```

Example 1

Set width-dependent eol-spacing for m1 and m2 routes

```
pro_set option width_dependent_end_of_line_spacing

on layer   m1
min_width  0.12
max_width  0.32
spacing    0.16

layer      m2
min_width  0.12
max_width  0.32
spacing    0.16
```

Example 2

Set width-dependent eol-spacing for 2 ranges of m1 route widths

```
pro_set option width_dependent_end_of_line_spacing

on layer   m1

min_width  0.12

max_width  0.32

spacing    0.16

layer      m1

min_width  0.60

max_width  0.80

spacing    0.20
```

Example 3

priority *value*

weight *value*

GDS Output

The GDS information describes how ProGen should output the geometries on the corresponding layer. The following must be set:

- GDS data and text layers
- When to output the GDS

gds *data_layer data_type text_layer text_type*

This defines the mapping of the ProGen layer and text data to output output GDS layers. Only layers which are actually going to be output (see `gdsout`) need layer mapping values defined.

`Data_layer` and `data_type` are the GDS layer number and data-type to which ProGen will write all geometries (rectangles and polygons) which are on the corresponding layer.

`Text_layer` and `text_type` are the GDS text layer and text-type to which ProGen will write all text which is attached to geometries on the corresponding layer.

gdsout *value*

Directs ProGen when it should output GDS data for geometries on the layer. `Value` may have one of the following values:

Value	Description
`comp`	Output layer data only when compaction is run
`skip`	Output layer data only when compaction is skipped
`all`	Always output layer data
`off`	Never output layer data

Table 8

ProGen uses many special layers in order to model layout constraints and design rules. It is often not necessary to output the data on all the individual layers. Instead just the data on the shadow-layer can be output. It typically doesn't matter however if both layers are output since ProGen will reduce the geometries before writing to GDS.

How To Set Layer Information

All of the layer information is set using the following command:

```
pro_set [FLAGS] layer_info COMMAND LAYER_SPEC
```

The easiest way to describe this is through an example. The following sets every characteristic possible for a layer called "metal3”:

```
   pro_set layer_info add {
       {metal3
         {fill
                  darkblue}
         {stipple
                  gray50}
         {outline
                  darkblue}
         {draworder
                  600}
         {min_size
                  metal3_min}
         {spacing
                  {metal3      metal3_sp diffnets_only}
                  {metal3_m3  {metal3_cb_sp metal3_sp}}}
         {overlap
                  {via4        metal3_ov_via4}}
         {borderspacing
                  XY}
         {connect
                  metal3_cb
                  via4}
         {shadowlayers
                  metal3_all}
         {priority
                  1}
         {weight
                  30}
         {gds
                  12 1 42 1}
         {gdsout
                  all}
       }

  }
```

Example 4

(Note that you cannot embed comments in the statement, but you can break up the statement to put comments before each block.)

- Geometries are drawn with a dark-blue fill pattern.
- A solid fill pattern is used.
- Geometries are drawn with a dark-blue outline.
- Geometries are drawn at position 600 in the order of layers. The number value is only relative to other layer numbers and does not mean that there are more then 600 layers or that it is the 600th layer drawn.

Why Set Layer Info?

The following are common usages of layer_info:

- Defining new layers.
- Modifing existing system layer properties.
- Describing design rule relationships between layers.

A simple new layer definition is:

```
pro_set layer_info add {

            {new_layer
                  {gdsout all}
                  {gds 200 0 0 0}
                  {overlap {poly_gate new_layer_ov_poly_gate}}
            }
          }
```

Example 5: Define a new layer and describe design rule relationships between layers

Where new_layer is the layer name, gdsout all tells the system that the layer is to be output in both skip compact and compacted results. "gds" tells the system which gds layer to output to; "overlap" tells the system to maintain an overlap of "poly_gate" with "new_layer" using the rule value defined in "new_layer_ov_poly_gate". With a definition like the one above, a new layer is defined, but unless an object in the system or celltemplate uses it or more specific properties are added, you will see nothing different in the layout.

To get the overlap initially, you need to have an object use new_layer overlapping poly_gate or to universally add "new_layer" to poly_gate. Register it as a shadowlayer of poly_gate using the following:

```
pro_set layer_info append {

        {poly_gate

                 {shadowlayers new_layer}

        }

}
```

Notice that layer_info append is used. This is so that no other shadowlayer relationships are interrupted.

The following example features a command which adds a spacing property to a system layer:

```
pro_set layer_info append {

                  {poly_gate
                     {spacing  {pdiff_tie gate_tie_sp}
                         {ndiff_tie gate_tie_sp}
                      }
                  }
         }
```

Example 6: Modifying an existing layer

`Poly_gate` is the system layer and this code adds a spacing definition to the `pdiff_tie` and `ndiff_tie` layers using the design rule called `gate_tie_sp`. `Gate_tie_sp` must be defined somewhere in the tech files using a `pro_set rule gate_tie_sp value` command.

ProGen Database Reference

ProGen maintains an internal database of information which can be accessed by symbolic objects, generators, option routines, and any other routines which interface with the ProGen system.

The `pro_set` command is used to set the database information. There are four directives which control how table data is added and deleted from ProGen's internal database. The directives are:

pro_set Directives

add

Add the values to the table. If the -nooverride flag is specified for pro_set, then column specific values are added only if there is presently no value defined. If the -override flag is specified, then column specific values will overwrite existing values.

append

When a column already has values for a specific row-entry, the new values are appended to the existing values. This can result in duplicate values in the row/column entry.

join

When a column already has values for a specific row-entry, the new values are appended to the existing values if the new values are not already in the existing values.

delete

Delete the entry.

The pro_set command has the following form:

```
pro_set [flag] table directive row-entry-list
```

The pro_set command manipulates the database contents, which are all keyword/values lists having the form:

```
{info_key kv_list}
```

Info_key indicates which class of information is being accessed.

- `grid_info`
- `layer_info`
- `net_info`
- `port_info`
- `tran_info`
- `trunk_info`
- `map_info`
- `rail_info`
- `rule_info`

`kv_list` is the sublist of keyword/values for a given `info_key`.

grid_info

Gridding information.

Attribute	Description
`x value`	x-axis grid size in internal ProGen units
`y value`	y-axis grid size in internal ProGen units
`x_min_offset value`	Minimum x-axis offset from grid in the “hi” direction in internal ProGen units
`y_min_offset value`	Minimum y-axis offset from grid in the “hi” direction in interal ProGen units
`x_max_offset value`	Maximum x-axis offset from grid in the “hi” direction in internal ProGen units

Table 9

Attribute	Description
`y_max_offset value`	Maximum y-axis offset from grid in the “hi” direction in internal ProGen units
`x_priority value`	value is a number from 0 to 9, inclusive, represnting the priority of the gridding force on the x-axis.
`y_priority value`	value is an number from 0 to 9, inclusive, respresenting the priority of the y-axis gridding force.
`x_weight value`	value is an integer representing the weight of the x-axis gridding force.
`y_weight value`	value is an integer representing the weight of the y-axis gridding force.

Table 9

layer_info

Layer information.

gds

This defines the mapping of the ProGen layer and text layer to the output GDS layers. Only those layers which are actually going to be output should have this keyword defined. The definition takes this form:

```
gds data_layer data_type text_layer text_type
```

`data_layer` and `data_type` are the GDS layer number and data-type to which ProGen will write all layer data (rectangles and polygons) on the corresponding layer.

`text_layer` and `text_type` are the GDS text layer and text type to which ProGen will write all text which is attached to data on the corresponding layer.

gdsout

Indicates when ProGen should output GDS data for geometries on the corresponding layer.

gdsout value

`Value` may have one of the following values:

Value	Description
`comp`	Output layer data only when compaction is run
`skip`	Output data only when compaction is skipped
`all`	Always output layer data
`off`	Never output layer data

Table 10

ProGen uses many special layers in order to model layout constraints and design rules. It is often not necessary to output the data on all the individual layers. Instead, just the data on the shadow-layer can be output. It typically doesn't matter however if both layers are output since ProGen will reduce the geometries before writing to GDS.

borderspacing

The border spacing keyword directs ProGen to automatically apply cell border spacing rules to data on the corresponding layer. The border spacing rule is one-half the layer spacing rule.

borderspacing *value*

`value` may have one of the following values:

Value	Description
`0`	No cell border spacing forces created.
`1`	Create cell border spacing forces from all cell sides (same as XY)
`XY`	Create cell border spacing forces from all cell sides (same as `1`)
`X`	Create left and right cell side spacing forces
`Y`	Create top and bottom cell side spacing focres

Table 11

Most layers don't actually need to have a border spacing defined. A layer that has another layer shadowing it does not need border spacing forces when the super-layer has the border spacing forces.

shadowlayers

Defines layers which will shadow the corresponding (master) layer. A shadow layer is created whenever the master layer is created and the edges of the shadow layer follow the master layer.

```
shadowlayers layer_list
```

`layer_list` is a list of one or more layers which should shadow the master layer.

shadowlayer_of

Defines layers which the master layer will shadow. This has the opposite affect of the "shadowlayers" keyword. Overlaps are automatically created when overlap rules have been defined.

```
shadowlayer_of layer_list
```

`layer_list` is a list of one or more layers which the master layer will shadow.

fill

Sets fill color of layer geometries.

```
fill color
```

ProGen will draw geometries on the corresponding layer filled in the color specified by color. Color can be specified in two forms:

Any of the valid textual names for `color` defined in the X11 server's database file, such as Blue, Blanched-Almond, or PapayaWhip. The location and name of the server database file is system dependent but can often be found in /usr/X11/lib/rgb.txt.

A numeric specification of the red, green, and blue intensities used to display the `color` of one of the forms:

Format	Description
#RGB	4-bit colors
#RRGGBB	8-bit color
#RRRGGGBBB	12-bit color
#RRRRGGGGBBBB	16-bit color

Table 12

Each R, G, and B value represents a single hexadecimal digit. When fewer than 16 bits are provided for each color, they represent the most significant bits of the color.

The example below shows two specifications for the same color:

```
{fill CornflowerBlue}

{fill #6495ed}   Decimal: R=100, G=149, G=237
```

Example 7

outline

Sets outline `color` of layer geometries.

```
outline color
```

ProGen outlines all geometries on the corresponding layer in the color specified by color.

`color` can have any values as defined above in the "fill" command.

stipple

Sets stipple pattern of layer geometries.

```
stipple pattern
```

Directs ProGen to stipple fill the geometries on the corresponding layer with the stipple `pattern`. If the "fill" option has not been specified then this option has no effect.

`pattern` must be one the following forms:

Format	Description
`@filename`	File containing a valid X11 bitmap description. Prolific provides many bitmap pattern files already in the $PROLIFIC/lib/proview/bitmaps directory
`name`	One of the predefinied stipple pattern names: • gray75 (75% gray, 3 of 4 bits on) • gray50 (50% gray, 2 of 4 bits on) • gray25 (25% gray, 1 of 4 bits on) • gray12 (12.5% gray, 1 of 8 bits on)
#RRRGGGBBB	12-bit color
#RRRRGGGGBBBB	16-bit color

Table 13

The following are examples of stipple pattern definitions:

`{stipple @tictactoe.bmp}` - A bitmap file I created

`{stipple gray12}` - System defined bitmap

outline *color*

The color to outline geometries on the layer. If this keyword is not specified then geometries are drawn with no outline. The format of color is the same as for the `fill` keyword.

draworder *integer*

Layers are drawn one after the other on top of each other. Geometries drawn on a layer which are drawn later then another layer may obscure the geometries on the earlier-drawn layer. The order in which the layers are drawn is based on the draworder keyword value. The larger the number relative to other layers, the later the layer is drawn.

The number may be anywhere in the range from 0 to 40000. If this keyword is not defined then the layer is placed at the end of the drawing order. Layers defined after this layer which also don't define this keyword will be on top of this layer.

Design Rule Information

Geometries that are created on a given layer typically inherit the design rules associated with that layer. Layers may have any of the following characteristics defined:

- spacing rules
- overlap (enclosure) rules
- cell border spacing
- minimum size geometries on the layer
- maximum size geometries on the layer
- spacing to_layer rule_list [flag]

The `spacing` keyword causes ProGen to automatically create spacing forces between geometries on the corresponding layers. The spacing force will be applied between geometries on the master layer and the layer defined by `to_layer`.

The `rule_list` is a list of one or more rule names which ProGen will use as the spacing value. The first rule in the list which returns a non-null value will be used.

The `[flag]` is an optional value which indicates if the spacing force should be applied to geometries on the same or different nets. The valid values are:

Value	Description
`diffnets_only`	Apply spacing if geometries only on different nets
`samenets_only`	Apply spacing if geometries on on same nets
`both`	Always apply spacing (default)

Table 14

overlap *over_layer rule_list*

The `overlap` keyword allows you to indicate which layers have overlap rules of other layers. The system will automatically apply overlap rules when creating shadow layers.

The `rule_list` is a list of one or more rule names which ProGen will look-up to use as the overlap value. The first rule in the list which returns a non-null value will be used.

borderspacing *value*

The `borderspacing` keyword directs ProGen to automatically apply cell border spacing rules to geometries on the corresponding layer. The border spacing rule is one-half the layer spacing rule. The argument value may have one of the following values:

Value	Description
`0`	No cell border spacing forces created
`1`	Create cell border spacing forces
`XY`	Create cell border spacing
`X`	Create left and right cell spacing forces
`Y`	Create top and bottom cell spacing forces

Table 15

Most layers don't actually need to have a border spacing defined. A layer which has another layer shadowing it does not need border spacing forces when the shadow-layer has the border spacing forces.

min_size *rule*

Tells ProGen the name of the rule to use to get the minimum size of a geometry on the layer. This is usually used to place minimum size forces on geometries to prevent them from collapsing during compaction.

max_size *rule*

Tells ProGen the name of the rule to use to get the maximum size geometry on the layer. This is very rarely specified.

Compaction

Compaction directives provide information which contribute to how geometries on the layer behave. The following information may be set:

- shadow layers
- auto-connection
- area minimization priority
- area minimization weight

shadowlayers *layers*

Defines layers which will shadow the corresponding (master) layer. A shadow layer is created whenever the master layer is created and the edges of the shadow layer follow the master layer. Layers is a list of one or more layers which should shadow the master layer. Overlaps are automatically created when overlap rules have been defined using the layer `overlap` keyword.

The following is an example of the use of `shadowlayers`:

```
{shadowlayers m1_all m1_cb}
```

shadowlayer_of *layers*

Defines layers which the master layer will shadow. This has the opposite affect of the `shadowlayers` keyword. Layers is a list of one or more layers which the master layer will shadow. Overlaps are automatically created when overlap rules have been defined.

A `shadowlayer_of` example:

```
{shadowlayer_of m1_all m1_cb}
```

connect *to_layer* [net_prop] [forces] [dim] [priority] [weight]

Causes geometries on the master layer to automatically connect to geometries on the `to_layer` whenever they start out overlapping.

The value of `net_prop` directs the system to propagate net information between geometries when they start out overlapping. It may have one of the following values:

Value	Description
1	Propagate net information (default and typical value)
0	Don’t propagate net information

Table 16

The `forces` value indicates if connection forces should actually be created between the objects. This is typically turned on. Turning it off is a tricky way to get the system to propagate the net numbers without actually connecting the shapes. The value of `forces` may be:

Value	Description
1	Create connection forces (default type and typical value)
0	Don’t create connection forces

Table 17

The `dim` indicates when connection forces are created between geometries on the layers. The value of `dim` can be any one of the following:

Value	Description
overlapping	Create forces if geometries actually overlap
touching	Create forces if sides are overlapping or just touching (default value)
point	Create forces even if corners are just touching
nottouching	Create connection foces if they are not touching
dontconnect	Don’t create connection forces, regardless

Table 18

You may optionally assign priorities and weights to the connection forces when they are created with the `priority` and `weight` values. The system default connection values are sufficient to hold geometries together during compaction. Changing the values is usually done to lower the values so the system will try to keep things connected if possible but won't insist on it.

pro_set option width_dependent_end_of_line_spacing [on|off]

```
layer layer-name
min_width microns
max_width microns
spacing microns
```

```
   priority 7
   weight 1
```

This option is used to define width-dependent end-of-line spacings to other geometries on the same layer. In other words, the spacing from the route end-of-line is dependent on the width of the route.

The spacing rule is applied when the width of the route is greater than or equal to the minimum width (min_width) value and less than the maximum width (max_width) value.

when (min_width >= width < max_width) apply spacing

This allows you to have multiple ranges of values for the same layer.

Parameters:

`on|off` - Turns option on or off
`layer` - ProGen layer name (no default)
`min_width` - Minimum width of route for spacing to be applied
`max_width` - Maximum width of route for spacing to be applied
`spacing` - Spacing design rule value in microns (no default)
`priority` - Priority of spacing forces (default = 7)
`weight` - Weight of spacing forces (default = 1)

The block of layer parameter values may be specified as many time as is needed.

To change the compaction pass on which the the option is active, use the following command:

```
pro_set option width_dependent_end_of_line_spacing_start_index <pass>
```

`<pass>` = the compaction pass number

Examples:

Set width-dependent eol-spacing for m1 routes

```
pro_set option width_dependent_end_of_line_spacing
{on
{layer       m1
min_width  0.12
max_width  0.32
spacing    0.16
```

Example 8

```
Set width-dependent eol-spacing for m1 and m2 routes

pro_set option width_dependent_end_of_line_spacing

on layer     m1
min_width  0.12
max_width  0.32
spacing   0.16

layer     m2
min_width  0.12
max_width  0.32
spacing   0.16
```

Example 9

```
Set width-dependent eol-spacing for 2 ranges of m1 route widths

pro_set option width_dependent_end_of_line_spacing

on layer      m1
min_width  0.12
max_width  0.32
spacing    0.16

layer      m1
min_width  0.60
max_width  0.80
spacing    0.20
```

Example 10

priority *value*

weight *value*

GDS Output

The GDS information describes how ProGen should output the geometries on the corresponding layer. The following must be set:

- GDS data and text layers
- When to output the GDS

gds *data_layer data_type text_layer text_type*

This defines the mapping of the ProGen layer and text data to output output GDS layers. Only layers which are actually going to be output (see gdsout) need layer mapping values defined.

`data_layer` and `data_type` are the GDS layer number and data-type to which ProGen will write all geometries (rectangles and polygons) which are on the corresponding layer.

`text_layer` and `text_type` are the GDS text layer and text-type to which ProGen will write all text which is attached to geometries on the corresponding layer.

gdsout *value*

Directs ProGen when it should output GDS data for geometries on the layer. `value` may have one of the following values:

Value	Description
`comp`	Output layer data only when compaction is run
`skip`	Output layer data only when copmpaction is skipped
`all`	Always output layer data
`off`	Never output layer data

Table 19

ProGen uses many special layers in order to model layout constraints and design rules. It is often not necessary to output the data on all the individual layers. Instead just the data on the shadow-layer can be output. It typically doesn't matter however if both layers are output since ProGen will reduce the geometries before writing to GDS.

How To Set Layer Information

All of the layer information is set using the following command:

```
pro_set [FLAGS] layer_info COMMAND LAYER_SPEC
```

The easiest way to describe this is through an example. The following sets every characteristic possible for a layer called "metal3":

```
pro_set layer_info add {
{metal3
          {fill
                    darkblue}
          {stipple
                    gray50}
          {outline
                    darkblue}
          {draworder
                    600}
          {min_size
                    metal3_min}
          {spacing
                    {metal3     metal3_sp diffnets_only}
                    {metal3_m3  {metal3_cb_sp metal3_sp}}}
          {overlap
                    {via4       metal3_ov_via4}}
          {borderspacing
                    XY}
          {connect
                    metal3_cb
                    via4}
          {shadowlayers
                    metal3_all}
          {priority
                    1}
          {weight
                    30}
          {gds
                    12 1 42 1}
          {gdsout
                    all}
  }
 }
```

(Note that you can not embed comments in the statement. but you can break up the statement to put comments before each block.)

- 1.Geometries are drawn with a dark-blue fill pattern.
- 2.A solid fill pattern is used.
- 3.Geometries are drawn with a dark-blue outline.
- 4.Geometries are drawn at position 600 in the order of layers. The number value is only relative to other layer numbers and does not mean that there are more then 600 layers or that it is the 600th layer drawn.

Why Set Layer Info?

The following are common usages of `layer_info`:

- Defining new layers;
- Modifing existing system layer properties;
- Describing design rule relationships between layers.

A simple new layer definition is:

```
pro_set layer_info add {

              {new_layer
                  {gdsout all}
                  {gds 200 0 0 0}
                    {overlap {poly_gate
                  new_layer_ov_poly_gate}}
              }
             }
```

Example 11: Define a new layer and describe design rule relationships between layers

Where new_layer is the layer name, gdsout all tells the system that the layer is to be output in both skip compact and compacted results. "gds" tells the system which gds layer to output to; "overlap" tells the system to maintain an overlap of "poly_gate" with "new_layer" using the rule value defined in "new_layer_ov_poly_gate". With a definition like the one above, a new layer is defined, but unless an object in the system or celltemplate uses it or more specific properties are added, you will see nothing different in the layout.

To get the overlap initially, you need to have an object use `new_layer` overlapping `poly_gate` or to universally add `new_layer` to `poly_gate`. Register it as a `shadowlayer of poly_gate` using:

```
pro_set layer_info append {

               {poly_gate
                  {shadowlayers new_layer}
                  }
         }
```

Notice that `layer_info append` is used. This is so that no other shadowlayer relationships are interrupted.

The following command adds a spacing property to a system layer:

```
pro_set layer_info append {

               {poly_gate
                     {spacing  {pdiff_tie gate_tie_sp}
                         {ndiff_tie gate_tie_sp}
                   }
            }
         }
```

Example 12: Modify an existing layer

`Poly_gate` is the system layer and this code adds a spacing definition to the `pdiff_tie` and `ndiff_tie` layers using the design rule called `gate_tie_sp`. `Gate_tie_sp` must be defined somewhere in the tech files using a `pro_set rule gate_tie_sp value` command.

grid_info

Gridding information.

Attribute	Description
`x` *`value`*	x-axis grid size in internal ProGen units
`y` *`value`*	Y-axis grid size in internal ProGen units
`x_min_offset` *`value`*	Minimum x-axis offset from grid in the “hi” direction in internal ProGen units
`y_min_offset` *`value`*	Minimum y-axis offset from grid in the “hi” direction in internal ProGen units
`x_max_offset`	Maximum x-axis offset from grid in the “hi” direction in internal ProGen units
`y_max_offset` *`value`*	Maximum y-axis offset from grid in the “hi” direction in internal ProGen units
`x_priority` *`value`*	`value` is a number from 0 to 9, inclusive, represnting the priority of the gridding force on the x-axis.

Table 20

layer_info

Layer Information.

gds

This defines the mapping of the ProGen layer and text layer to the output GDS layers. Only those layers which are actually going to be output should have this keyword defined. The definition takes this form:

```
gds data_layer data_type text_layer text_type
```

`data_layer` and `data_type` are the GDS layer number and data-type to which ProGen will write all layer data (rectangles and polygons) on the corresponding layer.

`text_layer` and `text_type` are the GDS text layer and text type to which ProGen will write all text which is attached to data on the corresponding layer.

gdsout

Indicates when ProGen should output GDS data for geometries on the corresponding layer.

```
gdsout value
```

Value may have one of the following values:

Value	Description
comp	Output layer data only when compaction is run
skip	Output layer data only when compaction is skipped
all	Always output layer data
off	Never output layer data

Table 21

ProGen uses many special layers in order to model layout constraints and design rules. It is often not necessary to output the data on all the individual layers. Instead, just the data on the shadow-layer can be output. It typically doesn't matter however if both layers are output since ProGen will reduce the geometries before writing to GDS.

borderspacing

The border spacing keyword directs ProGen to automatically apply cell border spacing rules to data on the corresponding layer. The border spacing rule is one-half the layer spacing rule.

```
borderspacing value
```

value may have one of the following values:

Value	Description
0	No cell border spacing forces created
1	Create cell border spacing forces from all cell sides (same as XY)
XY	Create cell border spacing forces from all cell sides (same as 1)
X	Create left and right cell side spacing forces
Y	Create top and bottom cell side spacing forces

Table 22

shadowlayers

Defines layers which will shadow the corresponding (master) layer. A shadow layer is created whenever the master layer is created and the edges of the shadow layer follow the master layer.

```
shadowlayers layer_list
```

layer_list is a list of one or more layers which should shadow the master layer.

shadowlayer_of

Defines layers which the master layer will shadow. This has the opposite affect of the `shadowlayers` keyword. Overlaps are automatically created when overlap rules have been defined.

```
shadowlayer_of layer_list
```

`layer_list` is a list of one or more layers which the master layer will shadow.

fill

Sets fill color of layer geometries.

```
fill color
```

ProGen will draw geometries on the corresponding layer filled in the color specified by color. color can be specified in two forms:

- Any of the valid textual names for a color defined in the X11 server's database file, such as Blue, BlanchedAlmond, or PapayaWhip. The location and name of the server database file is system dependent but can often be found in /usr/X11/lib/rgb.txt.
- A numeric specification of the red, green, and blue intensities used to display the color of one of the forms:

Format	Description
#RGB	4-bit colors
#RRGGBB	8-bit color
#RRRGGGBBB	12-bit color
#RRRRGGGGBBBB	16-bit color

Table 23

Each R, G, and B value represents a single hexadecimal digit. When fewer than 16 bits are provided for each color, they represent the most significant bits of the color.

The following example shows two specifications for the same color:

```
 Color Specification

{fill CornflowerBlue}

{fill #6495ed}   Decimal: R=100, G=149, G=237
```

Example 13: Color specification

outline

Sets outline color of layer geometries.

```
outline color
```

ProGen outlines all geometries on the corresponding layer in the color specified by color. color can have any values as defined above in the `fill` command.

stipple

Sets stipple pattern of layer geometries.

```
stipple pattern
```

Directs ProGen to stipple fill the geometries on the corresponding layer with the stipple pattern. If the `fill` option has not been specified then this option has no effect.

`pattern` must be one the following forms:

Format	Description
`@filename`	File containing a valid X11 bitmap description. Prolific provides many bitmap pattern files already in the $PROLIFIC/lib/proview/bitmaps directory
`name`	One of the predefinied stipple pattern names: • gray75 (75% gray, 3 of 4 bits on) • gray50 (50% gray, 2 of 4 bits on) • gray25 (25% gray, 1 of 4 bits on) • gray12 (12.5% gray, 1 of 8 bits on)
#RRRGGGBBB	12-bit color
#RRRRGGGGBBBB	16-bit color

Table 24

draworder

ProGen draws layer geometries in a specific order. The later a layer is drawn, the easier it will be to view the data on the layer since it be drawn on top of previously drawn layers. This keyword allows you to indicate the drawing order or the layers.

```
draworder integer
```

`integer` is an value between 0 and 40000. Layers are drawn in increasing order. Layers which are not given a number are placed at the end of the drawing order as they are defined.

priority

Default area minimization priority for geometries on this layer

```
priority integer
```

weight

Default area minimization weight for geometries on this layer.

```
weight integer
```

min_size

Defines the design rule name (defined in the rule.tcl file) that will be queried for the minimum geometry size on the corresponding layer.

```
min_size rule_name
```

max_size

Defines the design rule name (defined in the rule.tcl file) which will be queried for the maximum geometry size on the corresponding layer. This is rarely defined.

```
max_size rule_name
```

overlap

`Rule_names` is a list of design rule names (defined in the rule.tcl file) that will be queried to get the layer-overlapping-layer values.

```
overlap list

overlap {enclosed_layer rule_name ... rule_name}
```

The first *`rule_name`* in the list that produces a valid value is used and the remainder are ignored.

The *`enclosed_layer`* is the name of the layer which is overlapped (enclosed) by the main layer.

spacing

Defines spacing forces between geometries on corresponding layers.

```
spacing list

spacing {to_layer rule_name ... rule_name option}
```

connect

Defines connection forces between layers.

```
connect list
```

```
connect {to_layer [netprop] [forces] [dim] [priority] [weight]} ...
```

net_info

Net information.

type

Type values are various net classifications for this net.

```
type TYPE0 TYPE1 ... TYPEn
```

Valid classifications include:

- beentexted
- contacted
- control
- data
- feed
- gnd
- hasintrunk
- input
- internal
- output
- port
- power
- vdd
- widewire

x_grid

Indicates that geometries on layers in `layer_list` that implement the stubtype wires in stubtype-list for this net should be gridded along the x-axis.

```
x_grid layer_list stubtype-list
```

y_grid

Indicates that geometries on layers in `layer_list` that implement the stubtype wires in stubtype-list for this net should be gridded along the y-axis.

```
   y_grid layer_list stubtype-list
```

mastername

Global (master) net name to which a subnet is equivalent.

```
   mastername net_name
```

port_info

Port information; port characteristics can be controlled on per-port basis using the `pro_set port_info` command.

The port access options create port access, which means temporary m1 around the port, extending beyond the other attempted port-spanning, going in up to four directions (N S E W). The port-access is actually the m1_block layer, which has a normal m1 spacing rule around it. The m1_block port-access geometries will try to expand with the given priority/weight and will therefore push neighboring m1 geometries away.

To force a port to span two grids, use

```
   pro_set port_info add {

           {portname

               {span_layers              v1}

               {span_grids               2}

               {span_preferred_direction y}

               {text_float              on}

               {span_priority            6}

               {span_weight            1}

           }
```

`portname` is the port's name.

You may also force the direction to be horizontal or vertical.

To set the proppr expected (first valid solution) and acceptable (to quit looking for more solutions), use

```
pro_set global proppr_acceptable_cost cost_value
```

```
pro_set global proppr_expected_cost cost_value
```

You can get `cost_value` from the final cost values in the log file. It is common to use the same number for both costs.

It is important not to set the acceptable cost higher than the expected cost; also, if the default or user-defined costs change, the same solution will have a different final cost. So if you change the router costs or if a new version of the tool changes the default costs, there could be problems in finding a solution with this option set.

To force the placement from the placeorder in the log file, use

```
prospin_set generator options {^cellname$} {

    {placeorder {4 7 2 3 8 6 0 1 9 10 5 11}}

}
```

The example place order is a list of the order of the generators in the .cel file to be placed from left to right. The starting index of the first generator in the .cel file is 0.

This changes the placement and mirroring of generators. The generator index is the ordering, starting with 0 of the generator calls in the .cel file. Changing the placeorder can be used to ease routing congestion, maintain consistency of the layout in logic families, or to decrease runtime. By default, the placer will try to minimize the number of diffusion breaks. If mirrorgates is omitted the placer will try to find the optimal mirroring for the given placement.

access_layers

List of layers that have associated access properties for this port.

```
access_layers layer_list
```

access_dir_layer

List of directions for which access constraints will be added on layer.

```
access_dir_layer direction_list
```

The `direction_list` can include W, S, E, or N. Default is { }.

The following attributes have list values corresponding to the `access_dir_layer` list. Each value in the following lists corresponds to the direction at the same list position in the `access_dir_layer`. If a list is too short, any remaining unmatched directions get the last value specified.

access_grids_layer

Distance in units of routing grids for which port access must be preserved on layer for the directions specified in access_dir_layer. Default is 1.

access_priority_layer

Priorities at which to force access preservation layer for the directions specified in access_dir_layer. Default is 5.

access_weight_layer

Weight at which to force access preservation layer for the directions specified in access_dir_layer. Default is 1.

access_width_layer

Width for this direction should be a two-element list:

```
widthx widthy
```

The following is an example of creating port access:

```
pro_set port_info add {PG_DEFAULT {access_layers m1}
{access_dir_m1 W E} \
 {access_nettype_m1 input} {access_grids_m1 1 1} \
 {access_priority_m1 4 4} {access_weight_m1 20}  \
 {access_width_m1 {0.20 0.20}}}
pro_set port_info add {Z {access_layers m1} {access_dir_m1
W E} \
 {access_nettype_m1 output} {access_grids_m1 1 2}   \
 {access_priority_m1 4 4} {access_weight_m1 20}  \
 {access_width_m1 {0.20 0.32}}}
```

Example 14

This example creates two `port_access` directions: left (W) and right (E) each has a width of 0.2 in all directions. This is only applied to input ports.

For port Z (if port Z is an output port), the width is 0.20 in x and 0.32 in y, which means that the preferred port-access extends beyond the x-grid by 0.2/2 = 0.10 and will extend vertically by 0.16um. If Z is not an output port, nothing will happen.

access_nettype_layer

Only apply this direction if the port is one of the specified nettypes. Most useful for `PG_DEFAULT`.

bc_feed_y_track

Initial (pre-compaction) port placement on a specific horizontal (y) track of ports on feed trunks. The first track is 0, increasing upward, decreasing downward.

bc_feed_y_track *value*

Value	Description
`incr`	Ports are one track higher than previous
`track-number`	Track number of specific port

Table 25

vert_track

Restricts port placement to specific vertical track. The first track is 1 and increases to the right.

```
vert_track value
```

Value	Description
`any`	No restriction (default)
`track-number`	Restricted to specific track number

Table 26

horz_track

Restricts port placement to specific horizontal track. The first track is 1 and increases upward.

```
horz_track value
```

Value	Description
`any`	No restriction (default)
`track-number`	Restricted to specific track number

Table 27

vert_track_own

Indicates if port owns vertical track.

```
vert_track_own value
```

Value	Description
0	Port does not own vertical track (default)
1	Port owns vertical track
any	Port does not own any tracks (same as 0)
stagger	Don’t own entire track; only own vertical track within one grid above and below port position
integer	The number of grid lines from which the ownership distance is derived. A value of 2 equals one grid.

Table 28

The stagger setting is best used with the default port specification (primary key PG_DEFAULT); otherwise it must be set for all ports which are to be staggered with respect to each other.

horz_track_own

Indicates if port owns horizontal track.

```
horz_track_own value
```

Value	Description
0	Port does not own horizontal track (default)
1	Port owns horizontal track
any	Port does not own any tracks (same as 0)
stagger	Don’t own entire track; only own vertical track within one grid to left and right of port position
integer	The number of grid lines from which the ownership distance is derived. A value of 2 equals one grid.

Table 29

The stagger setting is best used with the default port specification (primary key PG_DEFAULT) otherwise; it must be set for all ports which are to be staggered with respect to each other.

spanaxis

Span port relative to axis.

```
spanaxis axis
```

Value	Description
X	Port spans horizontally along the x-axis
Y	Port spans vertically along the y-axis

Table 30

span_x_grids

Number of grids for port to span with respect to x-axis. values can be set for a specific port or as the default value for all ports using the special port name PG_DEFAULT.

```
span_x_grids integer-list
```

span_y_grids

Number of grids for port to span with respect to y-axis. values can be set for a specific port or as the default value for all ports using the special port name PG_DEFAULT.

```
span_y_grids integer-list
```

span_x_layers

Layers of port to span multiple grids for the x-axis. There must be the same number of entries in layer_list as there are for the corresponding `span_?_grids` integer-list. The special port name PG_DEFAULT may be used to provide a default for all ports.

```
span_x_layers layer_list
```

span_y_layers

Layers of port to span multiple grids for the y-axis. There must be the same number of entries in `layer_list` as there are for the corresponding `span_?_grids` integer-list. The special port name PG_DEFAULT may be used to provide a default for all ports.

```
span_y_layers layer_list
```

span_x_priority

Priority of forces which cause port to span multiple grids in the x-axis. This can be set for a specific port or as the default value for all ports using the special port name PG_DEFAULT. Individual port specifications have precedence over the default specification.

```
span_x_priority integer
```

span_y_priority

Priority of forces which cause port to span multiple grids in the y-axis. This can be set for a specific port or as the default value for all ports using the special port name PG_DEFAULT. Individual port specifications have precedence over the default specification.

```
span_y_priority integer
```

span_x_weight

Weight of forces which cause port to span multiple grids in the x-axis. This can be set for a specific port or as the default value for all ports using the special port name PG_DEFAULT. Individual port specifications have precedence over the default specification.

```
span_x_weight integer
```

span_y_weight

Weight of forces which cause port to span multiple grids in the x- and y-axis. This can be set for a specific port or as the default value for all ports using the special port name PG_DEFAULT. Individual port specifications have precedence over the default specification.

```
span_y_weight integer
```

span_x_forces

Controls when port spanning forces are enabled.

```
span_x_forces_enabled value
```

span_y_forces

Controls when port spanning forces are enabled.

```
span_y_forces_enabled value
```

Value	Description
`after_orthogonal_compaction`	Enable port spanning forces after orthoganal compaction (default)
`immediately`	Enable port spanning forces immediately

Table 31

grid_x_priority

Priority of port gridding forces. This can be for a specific port or as the default value for all ports using the special port name PG_DEFAULT. If not defined, the system uses [pro_get priority x_port_fine 6] and [pro_get priority y_port_fine 6].

```
grid_x_priority integer
```

Gridding can be turned off for the x-axis by setting the priority to 0.

grid_y_priority

Priority of port gridding forces. This can be for a specific port or as the default value for all ports using the special port name PG_DEFAULT. If not defined, the system uses [pro_get priority x_port_fine 6] and [pro_get priority y_port_fine 6].

```
grid_y_priority integer
```

Gridding can be turned off for the y-axis by setting the priority to 0.

grid_x_weight

Weight of port gridding forces. This can be for a specific port or as the default value for all ports using the special port name PG_DEFAULT. If not defined, the system uses [pro_get weight x_port_fine 1] and [pro_get weight y_port_fine 1].

```
grid_x_weight integer
```

grid_y_weight

Weight of port gridding forces. This can be for a specific port or as the default value for all ports using the special port name PG_DEFAULT. If not defined, the system uses [pro_get weight x_port_fine 1] and [pro_get weight y_port_fine 1].

```
grid_y_weight integer
```

grid_x_edge

Edge of port layer object to which the gridding is actually applied. Valid values are: lo | hi.

```
grid_x_edge EDGE
```

grid_y_edge

Edge of port layer object to which the gridding is actually applied. Valid values are: lo | hi.

```
grid_y_edge EDGE
```

stagger_horz_grids

Number of grids to force staggered ports apart. Default is 1. This is only used if "horz_track_own" is set to "stagger."

```
stagger_horz_grids integer-list
```

stagger_vert_grids

Number of grids to force staggered ports apart. Default is 1. This is only used if "vert_track_own" is set to "stagger."

```
stagger_vert_grids integer-list
```

stagger_horz_priority

List of priorities of port staggering forces.

```
stagger_horz_priority priority-list
```

stagger_vert_priority

List of priorities of port staggering forces.

```
stagger_vert_priority priority-list
```

stagger_horz_weight

List of weights of port staggering forces.

```
stagger_horz_weight weight-list
```

stagger_vert_weight

List of weights of port staggering forces.

```
stagger_vert_weight weight-list
```

obj

Force port to use sym_obj symbolic object.

```
obj sym_obj
```

pos

Input trunk contact position from left on which to place port. The first position is 0.

```
pos integer
```

pro_set option require_diffusion_island_rectangles on|off

Require all diffusion islands to be rectangles. A diffusion island is a diffusion mask connected shape; the source and drain of a transistor are part of the same diffusion island.

pro_set option require_output_diffusion_island_rectangles on|off

Require all diffusion islands that contain output nodes to be rectangles. A diffusion island is a diffusion mask-connected shape; the source and drain of a transistor are part of the same diffusion island.

pro_set option skip_detailed_placement *boolean*

This option specifies whether or not detailed placement should be run during routing optimization. The value can be any Tcl boolean value and the default is 0. Turning on this option is useful for speeding up large prosticks cells.

+ pro_set option diffusion_gap_dummy_poly [list <boolean>\

```
?auto_convert<boolean>?\

?extend_connections<boolean>?\

?extension<microns>?\

?minimum_area<square microns>?\

?ndiff_extension<microns>?\

?ndiff_hi_extension<microns>?\

?ndiff_lo_extension<microns>?\

?pdiff_extension<microns>?\

?pdiff_hi_extension<microns>?\

?pdiff_lo_extension<microns>?\

?priority<0-9>?\

?weight<integer>?\

    ]
```

Activating this option requests that minimum width dummy poly shapes be inserted in all internal diffusion gaps of a cell. The option does not add dummy poly shapes on the edges of the cell.

The first element of the list value for this option must be a Tcl boolean. A true value activates the option and a false value deactivates the option.

The dummy poly shapes will extend outside of the gaps beyond the edges of their neighboring diffusion shapes by an amount specified by the extension parameters. A single amount for all dummy poly shapes may be specified in microns using the "extension" parameter. Its default is 0.0.

Different extension amounts can be specified for different diffusion layers using the "ndiff_extension" of "pdiff_extension" parameters. If not specified, the value defaults to the value of the "extension" parameter.

Different extension amounts can be specified for different sides of the neighboring diffusion shapes using the "ndiff_hi_extension", "ndiff_lo_extension", "pdiff_hi_extension" and "pdiff_lo_extension" parameters. Since diffusion gaps are oriented vertically, the *_hi_extension parameters refer to how high above the neighboring diffusion shapes the dummy poly should extend and the *_lo_extension parameters refer to how low below the neighboring diffusion shapes the dummy poly should extend. If not specified, the value defaults to the value of either "ndiff_extension" or "pdiff_extension" as appropriate.

The "auto_convert" parameter specifies whether or not existing pass-through transistors should be used to generate the dummy poly shapes. If "auto_convert" is true, then any existing pass-through transistors that don't have template keys set, will get a dummy poly shape. If "auto_convert" is false, the existing pass-through transistors are left untouched. The default is true.

The "extend_connections" parameter specifies whether or not extension constraints should be applied to the sides of dummy poly shapes on pass-through transistors that are connected to other routes. The default is true. Note that if route connections exist, the dummy poly shape will be generated on the poly layer rather than the dummypoly layer.

The "minimum_area" parameter specifies the minimum area rule in square microns for the dummy poly layer. The default is the value of the poly minimum area rule. If the poly minimum area rule is undefined, the default is 0.0.

The "priority" and "weight" parameters specify the priority and weight of the extension constraint forces. The default is 7,1.

If the dummy poly must extend beyond the neighboring diffusion by 0.04 um, then use:

```
   pro_set option diffusion_gap_dummy_poly [list on extension 0.04]
```

If, in addition, routed connections don't need the extension constraint, then use:

```
   pro_set option diffusion_gap_dummy_poly [list on        \
   extension           0.04                            \
   extend_connections  false                           \
           ]
```

Example 15

text

Text to place on port. The default value is the net name. This allows the port to have a different name than the net.

```
   text STRING
```

channel

Channel of port. Default = 1.

```
   channel integer
```

types

Port types to use for selecting port objects based on type. The value is one or more types which should be defined in the customers technology area. The first type found in the list which has a corresponding SOT is used.

route_overlap

Have specific port rotate the metal-over-via values 90 degrees from the default.

```
rotate_overlap boolean
```

aux_procs

Allows the attachment of additional procedures to call after a port has been created. This allows you to extend the capability of the port object to include additional features.

```
aux_procs PROC_ARGS_list
```

The procedure must have the following interface:

```
proc procname {infolist} {body}
```

Arguments appended to passed in arguments:

```
port_id id

port_namename

port_xx

port_yy

port_anchorsg
```

There is a zero size anchor geometry in all ports which corresponds to the main gridding point of the port. The `port_anchor` is the shape-geo of the object.

Here are examples of additional_procs, Port Staggering, and Port Spanning, respectively:

```
pro_set port_info add {

 {PG_DEFAULT

  {aux_procs

 {pro_strap {layer m2} {orient horz}}

 {pro_strap {layer m2} {orient vert}}

  }

 }

 }
```

Example 16: additional_procs

```
pro_set port_info add {PG_DEFAULT

   {placement_vert  stagger}

   {stagger_vert_grids 1 2}

   {stagger_vert_priority default 4}

   {stagger_vert_weight   default 1}

 }
```

Example 17: Port Staggering

```
pro_set port_info add {PG_DEFAULT

   {span_y_layers v1}

   {span_y_grids   2}

 }
```

Example 18: Port Spanning

tran_info

Transistor information

ProAssemble generates the following information for every transistor. The values are available only after the generator has called ProAssemble. This means that CEL files and generators can not access the information but option routines, symbolic objects, and post-processing routines can.

source

The transistor source net name.

```
source name
```

drain

The transistor drain net name.

```
drain name
```

gate

The net names on the transistor gate. This is a list of three values:

- global-net
- net-lo
- net-hi

The net-lo and net-hi values are the actual names which connect to input trunks. Any of the names may or may not be the same.

```
gate {GLOBAL-NET NET-LO NET-HI}
```

type

General transistor diffusion type. Value values are: n, p, resistor, or { }.

```
type value
```

tkey

Template GSOT keyword name.

```
tkey name
```

row

The initial transistor placement row number. The lowest transistor row is number 0 and the row numbers increase by one.

```
row integer
```

pos

The position of the transistor in the row from the low side. The lowest position is 0 and the positions increase by one.

```
pos integer
```

gate_trunk_lo

Flag indicating if gate connects to trunk on lo side of transistor. value: 1 = gate connects to trunk, 0 = doesn't connect to trunk. The trunk name is the same as the "gate" lo-side name.

```
gate_trunk_lo boolean
```

gate_trunk_hi

Flag indicating if gate connects to trunk on hi side of transistor. Values: 1 = gate connects to trunk, 0 = doesn't connect to trunk. The trunk name is the same as the "gate" hi-side name.

```
gate_trunk_hi boolean
```

gate_length

Transistor gate length in microns. If not defined, then the value of the minimum gate length design rule (`gate_len`) is used.

```
gate_length MICRONS
```

feed_anchor

Indicates a specific transistor node which ProGen should use as a feed-anchor. A feed-anchor is a source/drain contact that must be used as the attachment point for a feed. Assuming there are multiple source/drain contacts that have the same net-name and therefore will be planar-routed together, the default behavior is to connect the feed to the source/drain contact that is closest (delta-x) to the pre-compacted feed location. Setting a feed-anchor will force the feed to connect to a specific source/drain contact.

```
feed_anchor list
```

list = `<source|drain> <lo|hi> 1`

List Value	Description
`source`	Connect to transistor source
`drain`	Connect to transistor drain
`lo`	Apply route in lo channel
`hi`	Apply route in hi channel
`1`	Route to transistor contact
`0`	Feed will connect to any source/drain except the specified transistor

Table 32

prefw_tolerance

When this value is set, the transistors will be flexible. The width of the transistors will prefer to be the specified width. However, the widths of transistors may vary, but will be kept within the tolerance of the specified starting width.

For example: the following setting makes all transistors flexible, with a width tolerance of 20%:

```
pro_set tran_info add \

[list PG_DEFAULT [list prefw_tolerance 0.20]]
```

Example 19

The priority and weight of the preferred width force can be overridden. The default weight is weaker than that of the 2d compaction forces.

The following example sets the default value for the preferred width priority and weight. The example sets the pulling force (preferred width = w).

```
pro_set priority tran_pref_width 4

pro_set weight tran_pref_width 1
```

Example 20

The pushing force is automatically created with a priority of one higher than the pulling force; in this case, the pushing force priority would be 5.

The following example sets the preferred width pushing force to be stronger than the 2d compactor forces.

```
pro_set priority tran_pref_width 4

pro_set weight tran_pref_width 1000
```

Example 21

trunk_info

Input trunk information

channel

Channel number of trunk. Channels start at 0 and incrementally increase by one.

```
channel positive-integer
```

port

Indicates if trunk has a port on it. values: 1 = has port, 0 = no port.

```
port boolean
```

merge_polycon

Forces transistor poly-contacts to merge on the trunk. The value is one or more list of transistor names which should merge. Merging has precedence over non-merging specifications.

```
merge_polycon list-OF-MERGE-list
```

```
{merge_polycon {n1 n2} {n3 n4 n5}}
```

Example 22: merge_polycon

This example forces n1 & n2 to merge; n3, n4 & n5 to merge; allows but doesn't force n2 & n3 to merge.

no_merge_polycon

Forces transistor poly-contacts not to merge on the trunk. The value is one or more list of transistor names which should not merge. Merging has precedence over non-merging specifications.

```
no_merge_polycon list-OF-NO-MERGE-list
```

```
{no_merge_polycon {p1 p2} {p3 p4 p5}}
```

Example 23: no_merge_polycon

This example forces p2 & p3 not to merge; p3, p4 & p5 not to merge; but doesn't force p2 & p3 not to merge (may or may not merge).

rail_info

Power/ground rail information

Every rail in the system is assigned a unique number which allows the system to support multiple rails of the same type. The primary key for the rail_info values is this unique rail number.

obj

Name of object which can be used as a lookup key in the "obj_info" database for power and ground rail information.

```
obj objname
```

map_info

Object mapping information

tran_models

Mapping of transistor model names to tkeys

```
tran_models list-OF-list
```

```
pro_set -nooverride map_info add {

  tran_models {n_default trann}

  {p_default tranp}

  {n   trann}

  {p   tranp}

  {nh  trannimp}

  {ph  tranpimp}

  }
```

Example 24

rule_info

Design rule information

The primary key for the rule_info tables is the design rule name. The system will automatically set the values for all design rules when the option flag rule_info is set to 1. If the flag is set to 0 (default setting), then values will not be set.

```
pro_set option rule_info 1

   ...define design rules...

 pro_set option rule_info 0
```

value

The design rule value in microns.

```
value micron
```

ProGen Predefined Variable Reference

The `pro_set` assigns a value to a ProGen predefined variable. ProGen identifies variables by splitting the specification into two parts: class and key. The `class` identifies the ProGen group to which the identifier belongs. The key is the actual identifier name.

The procedure has the following syntax:

```
pro_set class key value
```

Class Proc: Procedure Settings

These options describe a procedure or a list of procedures used to define advanced settings. The order that these options are specified in the technology file does not matter, but a list of procedures specified by an option will be called in the order they are listed, first to last.

A ***proc_name*** is a single procedure name, while a ***proc_name_list*** is a list of procedures.

celltemplate

Identifies the procedure that sets the initial cell template values. This procedure also identifies the mapping between template objects and symbolic object template mappings.

This command is deprecated; its functionality is handled internally by ProTech.

```
pro_set procs celltemplate proc_name
```

execprocs

Identifies one or more procedures to invoke after outputting the final cell datafile. This allows you to invoke post-ProGen filters on the ProGen output files.

The name of the output cell file and the technology directory are passed to the procedure, so each procedure should have the following form:

```
proc my_execproc

{filename tec_dirname}

 { ...Tcl code... }
```

This command is not required; the default is the empty list.

```
pro_set procs execprocs proc_name_list
```

layermap

Identifies the procedure that sets layout layer mapping information.

This command is not required; the default is the empty list.

```
pro_set procs layermap proc_name_list
```

notchfill

Identifies one or more procedures that implement user notch- or hole-filling. Registering these procedures allows them to be called during post-processing so that the ProGeo evaluation phase at the end of each compaction pass can notch-fill the intermediate layout states in the same way as the final layout, ensuring compatibility with many design-for-manufacturability rules.

Each procedure takes as an argument the current geo list representing the cell layout and returns the geo list corresponding to the notch-filled cell layout:

```
proc proc_name

{      geo_list  }

{    ...Tcl code...       return $modified_geo_list  }
```

This command is not required; the default is the empty list.

```
pro_set procs notchfill proc_name_list
```

objects

Identifies the procedures that contain technology-specific object mappings. Allows use of different symbolic objects than the package defaults allows.

The command is optional; the default is the empty list.

```
pro_set procs objects proc_name_list
```

options

Procedures that set technology option flags and values. All options override package defaults.

This command is optional; the default is the empty list.

```
pro_set procs options proc_name_list
```

overrides

Procedures that should be called to override calculated technology values (after ProGen has calculated technology values but before topology layout and compaction). This provides the ability to query ProGen on the values it calculated and make adjustments if desired.

This command is optional; the default is the empty list.

```
pro_set procs overrides proc_name_list
```

postprocess

Identifies a procedure to invoke just before ProGen outputs the layout data to the AGD file. This provides the ability to do technology-specific processing of the geometric data. Examples include rotating and flipping cell data, notch filling, scaling, etc.

This command is optional; the default is the empty list.

```
pro_set procs postprocess proc_name_list
```

rules

Procedure called to define design rules.

This command is not required; there is no default setting.

```
pro_set procs rules proc_name_list
```

Class Global: Global Settings

These options configure universally-applied technology setup options.

pro_set global auto_select_tie

Automatically determines the optimal tie-type. Requires the corner tie object to work properly.

This command is optional. The default is 1000000, which disables the automatic tie selection feature.

```
pro_set global auto_select_tie 1
```

pro_set global package

Identifies the name of the package from which the system will derive default layout style values. Package names are predefined by Prolific.

This command is optional. The default is basic.

```
pro_set global package package_name
```

pro_set global sotfiles

Identifies symbolic object template definition files from which ProGen should use the objects. The file list is loaded in first-to-last order, and these files are loaded after the package objects, overriding any objects of the same name.

This command is optional; the default is " ".

```
pro_set global sotfiles package_name
```

pro_set global techname

Provides a string recorded in the log file each time ProGen is run with the corresponding technology setup. A descriptive name with a version number to keep track of changes is the best option.

This command is optional; the default is " ".

```
pro_set global techname technology_name
```

pro_set global Xextraspread

Defines an additional amount of transistor spacing, in ProGen internal units.

This command is optional; the default is 0.

```
pro_set global Xextraspread extra_spacing_amount
```

Class Rule: Design Rule Mappings

These commands associate design rule values with design rule document names, and also associate the design rule values with ProGen's standard identifiers. These commands are set using this syntax:

```
pro_set rule rule-name rule-value [comment]

pro_set rule progen-name rule-value
```

The `rule-name` entry corresponds to the design rule document's rule name. The `progen-name` entry specifies the standard ProGen rule name for which a value is being set. The `rule-value` specifies the design rule value, in microns, to be assigned to the ProGen identifier. This is typically accomplished using the `pro_get` command to return a design rule value previously defined. The values can also be set directly, of course:

```
pro_set rule NW100  1.50     "N-well width"

pro_set rule NW114  1.00     "N-well space to P+ substrate tap"

pro_set rule nwell_ov_pdiff  1.00

pro_set rule nwell_min       [pro_get rule NW100]
```

Class Options: Technology Option Settings

While technology options can be set via the ProTech interface, options are also configurable using the pro_set command. These options take the general form:

```
pro_set option option setting
```

Some example options are described in this section:

compaction_order

Specifies the default compaction order and type.

```
pro_set option compaction_order compaction-pass-list
```

girdle

For each specified layer, try to reduce the width of shapes down to the specified size to remove unsightly overlaps between shapes and avoid width-dependent spacing rules.

```
pro_set option girdle layer-list
```

tran_width_vertex

Sets the transistor width vertex grid in microns. Allows transistor widths to be placed on a coarser grid than the general geometry vertex grid.

```
pro_set option tran_width_vertex width
```

routing_scheme

Controls the router gen_multi will use for a cell.

```
pro_set option routing_scheme scheme
```

A value can also be passed in directly to `gen_multi` as the `routing_scheme` argument using prospin. As a `gen_multi` argument, the default value is the special token default, which says to query `pro_get option routing_scheme` to get the value. If the routing scheme option hasn't been set in the database, a default of the empty string is used. This gives backwards-compatible behavior for old tech files. New tech files should have a default set for this option in protech.

The empty string ({ }) means to turn off this option.

All the other values, listed below, consist of three tokens separated by underscores. Each token corresponds to the kind of routing that should be used for each of three scenarios. The first token specifies which router to use for the case of a single subgenerator. The second token specifies which router to use for the case of multiple simple subgenerators. Currently, the definition of "simple" is (the `pasm_complex_size` generator) `inv`, `nand`, `nor`, `aoi`, and `oai`. The third token specifies which router to use for the case of multiple subgenerators of any type.

```
native_native_std

native_native_proppr

native_std_std

native_std_proppr

native_proppr_proppr

std_std_std

std_std_proppr

std_proppr_proppr

proppr_proppr_proppr
```

There are three different kinds of routing. The native setting uses the routing produced by the subgenerators without any further optimization. The std setting uses the standard routing optimizations and proppr means to use the ProPPR router.

The ProGen Exit Status Codes are:

Code	Description
-2	Warnings generated. Cell validity unknown.
-1	Fatal errors generated
0	OK. Cell valid.
6-9	Cell invalid. Exit code reflects highest priority cost after last compaction.

Table 33

Setting up a Cell Template

Each cell template must have a top-level definition which identifies the objects which make up the template and define its attributes.

```
pro_set obj_info add {celltemplate

     {proc     pro_celltemplate}

     {objects {cellborder}}

     {border   cellborder}

   }
```

The `celltemplate` primary key must be specified.

At a minimum, a cell template contains a cell border. The cell border defines the basic compaction region.

Cell Template Attributes

Cell templates have several attributes which must be specified in the celltemplate definition.

```
pro_set obj_info add {celltemplate

     {proc     pro_celltemplate}

     {objects {cellborder}}

     {border   cellborder}

     {xgrid    0.50}

     {ygrid    0.50}

   }
```

border

Indicates which subobject defines the actual cell border. The cell border object dictates the initial size of the template.

xgrid

The xgrid attribute defines the routing grid associated with the template. When a cell is compacted with gridding turned on, the cell boundary will be snapped to a grid multiple of this value in the X-axis. In addition, all objects (such as ports) to be placed on the routing grid will be snapped accordingly.

ygrid

The ygrid attribute defines the routing grid associated with the template. When a cell is compacted with gridding turned on the cell boundary will be snapped to a grid multiple of these values in the corresponding axis. Also, all objects (such as ports) which are to be placed on the routing grid will be snapped accordingly.

xgrid_lo, xgrid_hi (OPTIONAL)

These attributes define the low and high range boundaries of the port grids with respect to the cell left and bottom edges. Default values are 0 (zero). Ports may float within the range.

ygrid_lo, ygrid_hi

These attributes define the low and high range boundaries of the port grids with respect to the cell left and bottom edges. Default values are 0 (zero). Ports may float within the range.

objects

Objects which make up template. Objects are considered part of the template object and do not exist as stand-alone objects in the ProGen system. The physical geometeries are part of the cell template.

subobjs (OPTIONAL)

Subobjects within the template. Subobjects are stand-alone objects which are placed in the template object. They have a unique object ID independent of the template and are created along with the cell template. The benefit of a subobject is that it will design rule and connection forces to the other celltemplate objects and cellborder.

proc (OPTIONAL)

Name of procedure to call to implement cell template object. If not specified, the system will call pro_celltemplate. The proc procedure must have the following calling conventions:

```
proc proc { lef bot rig top [arg...] }
```

init_proc (OPTIONAL)

Name of procedure to call to initialize specific values needed by system before the template is actually created. This is typically used with stacked-generators. Optionally, arguments may be provided after the procedure name and they will be passed to the procedure. The init_proc procedure must have the following calling convention:

```
proc init_proc { lef bot rig top [arg...] }
```

Cell Border

The cell border is a subcomponent of the template and must be defined with a pro_set statement:

```
pro_set obj_info add {cellborder
     {proc       pro_cellborder}
     {bc_rig     100.0}
     {ac_rig     pull self lef}
     {ac_top     exact self bot 8.00 cellborder_maxy}
     {name_x     0.00}
     {name_y     0.00}
     {name_layer cellborder}
   }
```

Currently, due to limitations of ProGen, you can not directly control the initial left, bottom, and top edge positions through this specification. That means that specifying bc_lef, bc_bot, bc_top will have no affect. You can also not control the final left and bottom positions through this specification. This means that specifying ac_lef and ac_bot will have no effect. ProGen always shifts the data such that the bottom-left corner of the cellborder is 0,0.

name_x, name_y

Indicates the placement position of the cell name text relative to the cells bottom-left corner in microns. The default values are 0,0.

name_layer

The layer on which to place the cell name text. The default layer is the cellborder text layer. If a value is given which is different then the cellborder text layer, then a piece of geometry is created on the layer to which the text is attached.

Object Placements

There are two sets of edge specifications which are associated with all objects: initial pre-compaction relative position and post-compaction desired position. Each set has a different specification format.

Pre-Compaction Relative Positions

These coordinates describe how to initially place an object before compaction relative to other objects or in exact coordinates.

The following example illustrates placing an n-well object with respect to the initial cellborder. The left, right, and top edges of the n-well are positioned coincident with the corresponding cellborder edges. The bottom edge of the n-well is placed forty-five microns down from the top edge of the cellborder.

```
pro_set obj_info add {nwell

     {bc_bot   rel_obj cellborder top -45.00}

     {bc_top   rel_obj cellborder top   0.00}

     {bc_lef   rel_obj cellborder lef   0.00}

     {bc_rig   rel_obj cellborder rig   0.00}

   }
```

There are three placement forms which may be specified:

- rel_obj `object edge` [offset]
- value `offset`
- none

rel_obj object edge [offset]

Places an edge relative to the initial placement of another object's edge. An object can reference itself by specifying the keyword self for the reference object. When referencing an object other then itself the reference object must have already been built. Two edges within the same object should not reference each other. The offset value is optional and will default to 0 (zero).

value offset

Place an edge at the exact coordinate value specified. The offset value is not optional.

none

Indicates that no specific pre-compaction placement value is desired. This will result in a value of 0 (same as specifying value 0).

Compaction Placement Positions

These specifications indicate the desired final positions of objects after compaction has completed. Placements may be specified relative to other objects, in absolution coordinates, or not at all.

The following example illustrates a ground rail where the bottom, left, and right edges follow the corresponding edge of the cellborder. The top edge of the rail follows the bottom edge of the rail:

```
pro_set obj_info add {gnd_rail
     {ac_bot   follow cellborder bot}
     {ac_top   follow self       bot}
```

```
        {ac_lef   follow cellborder lef}
        {ac_rig   follow cellborder rig}
    }
```

There are eight compaction placement forms which may be specified:

- follow `object edge`
- exact `object edge` [offset] [priority] [weight] [option]
- pull `object edge` [offset] [priority] [weight] [option]
- push `object edge` [offset] [priority] [weight] [option]
- keepout `object edge` lo hi [priority] [weight]
- grid object edge grid [lo] [hi] [priority] [weight]
- static
- none

follow object edge

Creates forces such that the edge follows the reference object edge.

exact object edge [offset] [priority] [weight] [option]

Creates forces which cause edge to be placed exactly at some offset from the relative edge. The "spacing" priority and weight are used as the default.

An exact force actually creates a pushing force and a pulling force internally. If the priority is 6 or greater then the pushing force has the specified priority and the pulling force is 1 less then the specified priority.

The option allows you to specify that the forces be enabled or disabled after the running of the compaction in the corresponding axis. The option can have one of the following four values:

enable_after

Enable the forces after the first X compaction

enable_after_y

Enable the forces after the first Y compaction

disable_after_x

Disable the forces after the first X compaction

disable_after_y

Disable the forces after the first Y compaction

pull object edge [offset] [priority] [weight] [option]

Creates pulling forces between edges. The offset can be either positive or negative. A positive value places a force which pulls to the right of the relative edge. A negative value places a force which pulls to the left of the relative edge. The default offset is 0. The default priority is 4 and default weight is 1.

An interesting side effect is that negative 0 (-0) is not the same as positive 0. A pulling force of negative 0 causes the edge to be pulled to the left of the relative edge while a positive 0 pulls to the right.

The option allows you to specify that the forces be enabled or disabled after the running of the compaction in the corresponding axis. The option can have one of the following four values:

enable_after

Enable the forces after the first X compaction

enable_after_y

Enable the forces after the first Y compaction

disable_after_x

Disable the forces after the first X compaction

disable_after_y

Disable the forces after the first Y compaction

push object edge [offset] [priority] [weight] [option]

Creates pushing forces between edges. The offset can be either positive or negative. A positive value places a force which pushes to the right of the relative edge. A negative value places a force which pushes to the left of the relative edge. The default offset is 0. The default priority is 4 and default weight is 1.

Negative 0 (-0) is not the same as positive 0. A pushing force of negative 0 causes the edge to be pushed to the left of the relative edge while a positive 0 pushes to the right.

See “pull object edge [offset] [priority] [weight] [option]” above for the four possible values for the option setting.

The following illustrates push and pull forces for two edges of two objects:

```
   {pro_set obj_info add {A
            ...
            {ac_rig {pull B lef -10.0}}  # pull B 10um to the left of A
            ...
        }
        {pro_set obj_info add {A
            ...
            {ac_rig {pull B lef  10.0}} # pull B 10um to the right of A
            ...
        }
        {pro_set obj_info add {A
            ...
            {ac_rig {push B lef -10.0}}  # push B 10um to the left of A
            ...
        }
        {pro_set obj_info add {A
            ...
            {ac_rig {push B lef  10.0}}  # push B 10um to the right of A
            ...
        }
```

Example 25

keepout object edge lo hi [priority] [weight]

Creates a keep-out region about the edge of the reference object. The keep-out region is bounded by the lo and hi values specified. These values may be positive or negative. The default priority is 4 and the default weight is 1. The region defined by the lo and hi values does not include the lo and hi values as part of the keep-out region.

grid object edge grid [priority] [weight]

Creates a gridding force which causes the edge to be positioned as a function of the grid value from the edge of the reference object. The optional lo and hi values allow the specification of a delta range about each grid point in which the edge can fall and still be considered on grid. The default values for lo and hi are zero. The default priority is 4 and the default weight is 1.

static

Edge is static with respect to objects origin. Typically two opposing edges are made static.

none

No forces are put on edge; the result is that the edge will freely float during compaction.

The option allows you to specify that the forces be enabled or disabled after the running of the compaction in the corresponding axis. The option can have one of the following values:

object

Reference object. The special keyword self may be specified which indicates that the object is referencing itself. This has the same affect as specifying the objects name as the reference object.

edge

Reference edge of reference object. Valid values are one of the following: lef, bot, rig, top.

offset

All offset values are specified in microns. A positive value refers to above/right of the reference edge and a negative value refers to below/left of the reference edge. The default value is always 0 (zero).

priority

The priority to use for the force. All specifications have default values which typically make sense. Valid values are an integer in the range of 0 to 9 or a system constant name. If not an integer, then the system looks to see if a priority constant value has been defined using pro_set. If not, then the default value is used.

weight

The weight to apply to the force. Specifications have default values that rarely need changing. Valid values are an integer or a system weight constant. This follows the same lookup rules as priority, above.

Object Procedures

pro_celltemplate

Object Type

The cell template object is a stand-alone object and receives a unique ProGen identifier.

Attributes

objects

Objects to create as part of template. Template is parent.

subobjs

Independent objects to create when template is created.

border

Name of cell border object from above list.

xgrid

X grid size in microns. Cell border is snapped to value.

ygrid

Y grid size in microns. Cell border is snapped to value.

xgrid_lo

Port X gridding low range value from cell left edge in microns. Default value is 0.

xgrid_hi

Port X gridding low range value from cell left edge in microns. Default value is 0. If a value is given for xgrid_lo but not xgrid_hi then xgrid_hi will use xgrid_lo as its value.

ygrid_lo

Port Y gridding low range value from cell bottom edge in microns. Default value is 0.

ygrid_hi

Port Y gridding high range value from cell bottom edge in microns. Default value is 0. If a value is given for ygrid_lo but not ygrid_hi then ygrid_hi will use ygrid_lo as its value.

init_proc

Procedure to set system values as early as possible.

An example of a cell template object definition:

```
 pro_set obj_info add {celltemplate

     {proc      pro_celltemplate}

     {objects  {cellborder vdd_rail gnd_rail}}

     {border    cellborder}

     {xgrid     1.0}

     {ygrid     1.0}

     {xgrid_lo  0.5}

     {xgrid_hi  0.5}

     {ygrid_lo  0.5}

     {ygrid_hi  0.5}

   }
```

Example 26

border

Indicates which subobject defines the actual cell border. The cell border object dictates the initial size of the template.

xgrid micron

The xgrid and ygrid attributes define the routing grid associated with the template. When a cell is compacted with gridding turned on the cell boundary will be snapped to a grid multiple of these values in the corresponding axis. Also all objects (such as ports) which are to be placed on the routing grid will be snapped accordingly.

ygrid micron

The xgrid and ygrid attributes define the routing grid associated with the template. When a cell is compacted with gridding turned on the cell boundary will be snapped to a grid multiple of these values in the corresponding axis. All objects (such as ports) which are to be placed on the routing grid will be snapped accordingly.

xgrid_lo [micron]

One of four attributes defining the low and high range boundaries of the port grids with respect to the cell left and bottom edges. Default values are 0 (zero). Ports may float within the range.

ygrid_lo [micron]

One of four attributes defining the low and high range boundaries of the port grids with respect to the cell left and bottom edges. Default values are 0 (zero). Ports may float within the range.

xgrid_hi [micron]

One of four attributes defining the low and high range boundaries of the port grids with respect to the cell left and bottom edges. Default values are 0 (zero). Ports may float within the range.

ygrid_hi [micron]

One of four attributes defining the low and high range boundaries of the port grids with respect to the cell left and bottom edges. Default values are 0 (zero). Ports may float within the range.

objects LIST-OF-OBJECT-NAMES

Objects which make up template. Objects are considered part of the template object and do not exist as stand-alone objects in the ProGen system. The physical geometeries are part of the cell template.

subobjs LIST-OF-SUBOBJ-NAMES (OPTIONAL)

Subobjects within the template. Subobjects are stand-alone objects which are placed in the template object. They have a unique object ID independent of the template are created along with the cell template. The benefit of a subobject is that it will design rule and connection forces to the other celltemplate objects and cellborder.

proc PROC-NAME (OPTIONAL)

Name of procedure to call to implement cell template object. If not specified, the system will call pro_celltemplate. The proc procedure must have the following calling convention:

```
proc proc { lef bot rig top [arg...] }
```

init_proc PROC-NAME (OPTIONAL)

Name of procedure to call to initialize specific values needed by system before the template is actually created. This is typically used with stacked-generators. Optionally, arguments may be provided after the procedure name and they will be passed to the procedure. The init_proc procedure must have the following calling convention:

```
proc init_proc { lef bot rig top [arg...] }
```

pro_cellborder

Object Type

All geometries created by this procedure are children of the calling object.

Attributes

bc_rig

Initial pre-compaction placement of right edge

ac_top

Final post-compaction placement of top edge

ac_rig

Final post-compaction placement of right edge

name_x

Position of cell name text in X

name_y

Position of cell name text in Y

name_layer

Cell name layer

name_properties

Cell name text properties

xgrid

Cellborder edge gridding value in X

ygrid

Cellborder edge gridding value in Y

xgrid_priority

Cellborder edge gridding priority in X

ygrid_priority

Cellborder edge gridding priority in Y

xgrid_weight

Cellborder edge gridding weight in X

ygrid_weight

Cellborder edge gridding weight in Y

Coordinates

The cellborder is a special object which only allows you to specify the pre-compaction right edge coordinate. The left, bottom, and top edges are calculated by ProGen. Only the post-compaction top and right edge placements may be specified and these should be relative to itself.

name_x, name_y

Indicates the placement position of the cell name text relative to the cells bottom-left corner in microns. The default values are 0,0.

name_layer

The layer on which to place the cell name text. The default layer is the cellborder text layer. If a value is given which is different then the cellborder text layer then a piece of geometry is created on the layer to which the text is attached.

name_properties

Cell name GDS text properties to apply to text. Do not put quotation marks around the properties.

xgrid micron_value

Allows you to define a gridding value for the cellborder edge that is different than the port grids. The default value is the port grid defined by the celltemplate object.

ygrid micron_value

Allows you to define a gridding value for the cellborder edge that is different than the port grids. The default value is the port grid defined by the celltemplate object.

xgrid_priority [0-9]

Priority of cellborder gridding force along the X-axis. Default value is the cellborder_grid priority.

ygrid_priority [0-9]

Priority of cellborder gridding force along the Y-axis. Default value is the cellborder_grid priority.

xgrid_weight positive_integer

Weight of x-axis cellborder gridding forces. Default value is the cellborder_grid weight.

ygrid_weight positive_integer

Weight of y-axis cellborder gridding forces. Default value is the cellborder_grid weight.

```
pro_set obj_info add {cellborder
     {proc     pro_cellborder}
     {bc_rig   value 100.00}
     {ac_rig   pull self lef}
     {ac_top  {pull self bot 10.00 6}
              {push self bot 10.00 7}}
   }
```

Example 27: pro_cellborder

pro_rail

Object Type

All geometries created by this procedure are children of the calling object.

Attributes

type

Type of rail: "gnd" or "vdd'

bc_bot

Initial pre-compaction placement

bc_top

Initial pre-compaction placement

bc_lef

Initial pre-compaction placement

bc_rig

Initial pre-compaction placement

ac_bot

Final post-compaction placement

ac_top

Final post-compaction placement

ac_lef

Final post-compaction placement

ac_rig

Final post-compaction placement

layer

Rail layer

rail_id

Rail ID for routing (optional)

text

Text to place on rail, default = no text

text_x

X-position of text relative rail left edge, default = 0

text_y

Y-position of text relative rail bottom edge, default = 0

text_layer

Text layer, default = rail layer

```
     pro_set obj_info add {gnd_rail
          {proc          pro_rail}
          {type          gnd}
          {bc_bot        rel_obj cellborder bot -0.60}
          {bc_top        rel_obj self       bot  1.20}
          {bc_lef        rel_obj cellborder lef  0.00}
          {bc_rig        rel_obj cellborder rig  0.00}
          {ac_bot        follow cellborder bot}
          {ac_top        follow cellborder bot}
          {ac_lef        follow cellborder lef}
          {ac_rig        follow cellborder rig}
          {layer         cb_m1}
          {text          VSS}
          {text_x        0.80}
          {text_y        0.60}
          {text_layer    cb_m1}

  }
```

Example 28. Rail setup

pro_tie

Object Type

Geometries built by this procedure are stand-alone ProGen objects. They are not children of the calling object. They receive a unique ProGen identifier and get all automatic forces with respect to the cell template and other objects.

Attributes

type

Type of tie: "substrate" or "well"

layers

Tie layers (three or more layers)

bc_bot

Initial pre-compaction placement

bc_top

Initial pre-compaction placement

bc_lef

Initial pre-compaction placement

bc_rig

Initial pre-compaction placement

ac_bot

Final post-compaction placement

ac_top

Final post-compaction placement

ac_lef

Final post-compaction placement

ac_rig

Final post-compaction placement

diff_delta_lef

Adjust diffusion-over-contact extensions delta

diff_delta_bot

Adjust diffusion-over-contact extensions delta

diff_delta_rig

Adjust diffusion-over-contact extensions delta

diff_delta_top

Adjust diffusion-over-contact extensions delta

properties

Properties to associate with layer geometries.

type

Indicates if this is a substrate tie or a well tie. A substrate tie is typically connected to the ground rail and is an p-diffusion based device. A well tie is typically connected to the power rail and placed in the cell template well region is an n-diffusion based device.

layers

List of three or more layers which make up ties. The layers values correspond to diffusion, contact, and metal layers (in that order). The default values depend on the tie type:

```
substrate = pdiff_tie cpdiff_tie cb_m1 [via metal]*

      well      = ndiff_tie cndiff_tie cb_m1 [via metal]*
```

diff_delta_lef microns

Adjust the amount which diffusion extends beyond the contact in respect to the left edge. The amount is added to the design rule amount.

diff_delta_bot microns

Adjust the amount which diffusion extends beyond the contact in respect to the bottom edge. The amount is added to the design rule amount.

diff_delta_rig microns

Adjust the amount which diffusion extends beyond the contact in respect to the right edge. The amount is added to the design rule amount.

diff_delta_top microns

Adjust the amount which diffusion extends beyond the contact in respect to the top edge. The amount is added to the design rule amount.

delta_lef entry

Adjust the amount which geometries on the corresponding layer extend beyond the default extensions. This amount is added to extensions which naturally exist in the tie due to overlaps and minimum size geometries. These values will supersede the "diff_delta_*" values. To extend geometries to the left or downward specify negative values, to extend to the right or upward specify positive values.

delta_bot entry

Adjust the amount which geometries on the corresponding layer extend beyond the default extensions. This amount is added to extensions which naturally exist in the tie due to overlaps and minimum size geometries. These values will supersede the "diff_delta_*" values. To extend geometries to the left or downward specify negative values, to extend to the right or upward specify positive values.

delta_rig entry

Adjust the amount which geometries on the corresponding layer extend beyond the default extensions. This amount is added to extensions which naturally exist in the tie due to overlaps and minimum size geometries. These values will supersede the "diff_delta_*" values. To extend geometries to the left or downward specify negative values, to extend to the right or upward specify positive values.

delta_top entry

Adjust the amount which geometries on the corresponding layer extend beyond the default extensions. This amount is added to extensions which naturally exist in the tie due to overlaps and minimum size geometries. These values will supersede the "diff_delta_*" values. To extend geometries to the left or downward specify negative values, to extend to the right or upward specify positive values.

```
{delta_lef  entry...}

        entry = {layer microns}
```

Coordinates

The coordinates correspond to the center lines of the contact geometry. The left values are half the horizontal minimum contact size to the right of the contact geometry left edge. The right values are half the horizontal minimum contact size to the left of the contact right edge. The bottom values are half the vertical minimum contact size above the contact geometry bottom edge. The top values are half the vertical minimum contact size below the contact geometry top edge.

For a minimum size tie the left and right edges refer to the same point within the tie and the bottom and top edges refer to the same point.

properties

A list of any valid ProGen properties which may be associated with shape-geos. The default is no properties. This command has the form:

```
{properties  entry...}

        entry = property

                | {layer properties}
```

The value of entry has two forms. The first form defines a property which is assigned to all layers. The second form assigns one or more properties to a specific layer. These forms may be mixed in the attribute specification.

Valid properties include:

```
MAKE_DISTRIBUTE_EXACT

MAKE_EXTENDCONNECT

MAKE_JUSTIFY_DOWN

MAKE_JUSTIFY_LEFT

MAKE_JUSTIFY_RIGHT

MAKE_JUSTIFY_UP

MAKE_MUSTCONNECT

MAKE_NOCONNECT

MAKE_NOMERGE

MAKE_NO_COLLAPSE_WARNING

MAKE_TEMP
```

For example: let's look at a substrate tie which overlaps the cell border left, bottom, and right edges and follows the border. The MAKE_TEMP property is assigned to all layer data and other properties are assigned only to data on the cpdiff_tie layer.

```
pro_set obj_info add {subtie
   {proc       pro_tie}
   {type       substrate}
   {layers     pdiff_tie cpdiff_tie cb_m1}
   {bc_lef     rel_obj cellborder lef}
   {bc_rig     rel_obj cellborder rig}
   {bc_bot     rel_obj cellborder bot}
   {bc_top     rel_obj cellborder bot}
   {ac_lef     follow cellborder lef}
   {ac_rig     follow cellborder rig}
   {ac_bot     follow cellborder bot}
   {ac_top     follow cellborder bot}
   {properties MAKE_TEMP {cpdiff_tie MAKE_DISTRIBUTE_EXACT
MAKE_NOCONNECT}}
 }
```

Example 29

pro_shape

Object Type

All geometries created by this procedure are children of the calling object.

Attributes

layer

Layer on which to place shape

sgname

Shape-geo name to assign to shape

properties

ProGen properies with which to tag shape

bc_bot

Initial pre-compaction placement

bc_top

Initial pre-compaction placement

bc_lef

Initial pre-compaction placement

bc_rig

Initial pre-compaction placement

ac_bot

Final post-compaction placement

ac_top

Final post-compaction placement

ac_lef

Final post-compaction placement

ac_rig

Final post-compaction placement

text

Add text to shape

text_x

Position of text in X with respect to shape bot-lef

text_y

Position of text in Y with respect to shape bot-lef

text_properties

Text properties

layer

Layer on which to place shape-geo. Any valid ProGen layer may be specified. There is no default value.

sgname

Shape-geo name to give to geometry which is created. There is no default value.

properties

A list of any valid ProGen properties which may be associated with shape-geos. The default is no properties. Valid values include:

```
MAKE_DISTRIBUTE_EXACT

MAKE_EXTENDCONNECT

MAKE_JUSTIFY_DOWN

MAKE_JUSTIFY_LEFT

MAKE_JUSTIFY_RIGHT

MAKE_JUSTIFY_UP

MAKE_MUSTCONNECT

MAKE_NOCONNECT

MAKE_NOMERGE

MAKE_NO_COLLAPSE_WARNING

MAKE_TEMP
```

text

Text to add to shape geo. Do not enclose text in quotation marks unless you want the quotes in the output.

text_x

Position of text in X with respect to shape-geo bottom-left corner.

text_y

Position of text in Y with respect to shape-geo bottom-left corner.

text_properties

GDS text properties to be associated with text. See GDSII manual for description of properties (such as font, size, etc).

```
      pro_set obj_info add {nwell
           {proc         pro_shape}
           {sgname       cellborder_nwell}
           {layer        nwell}
           {bc_bot       rel_obj self       top -45.00}
           {bc_top       rel_obj cellborder top   0.41}
           {bc_lef       rel_obj cellborder lef  -0.41}
           {bc_rig       rel_obj cellborder rig   0.41}
           {ac_lef       follow cellborder lef}
           {ac_rig       follow cellborder rig}
           {ac_top       follow cellborder top}
           {ac_bot       none}
           {properties   MAKE_MUSTCONNECT}

   }
```

Example 30. Use of pro_shape

pro_well

Object Type

All geometries created by this procedure are children of the calling object.

Attributes

type

Type of well: standard (default), notched

layer

Layer of well (default = n-well)

antilayer

Only used when layer type is notched. Defines complementary layer of main well layer (default = pwell).

sgname

Shape-geo name to assign to shape

bc_bot

Initial pre-compaction placement

bc_top

Initial pre-compaction placement

bc_lef

Initial pre-compaction placement

bc_rig

Initial pre-compaction placement

ac_bot

Final post-compaction placement

ac_top

Final post-compaction placement

ac_lef

Final post-compaction placement

ac_rig

Final post-compaction placement

obj_names

List of notched well components: lef_name rig_name cen_name

lef_name

Left shape name and attributes

rig_name

Right shape name and attributes

cen_name

Center shape name and attributes

type

Indicates type of well geometry: standard or notched.

A standard well is a single rectangle which covers the top half of the celltemplate.

A notched well results in two complementary regions of geometries. The primary well region (typically n-well) has a rectangle on the left and right edges of the cellboarder. These preserve the well interface region when abutted with other cells. There is also a center rectangle which is allowed to float up and down. The following illustrates how the bc/ac coordinates control the edges:

```
:- bc_lef, ac_lef                       :- bc_rig, ac_rig
         :                                       :
         :                                       :
         +-----+---------------------------+-----+ <- bc_top, ac_top
         |.....| ' ' ' ' ' ' ' ' ' ' ' ' ' |.....|
         |..+--|---------------------------|--+..|
         |..|..| ' ' ' ' ' ' ' ' ' ' ' ' ' |..|..|
         |..|..|  ' ' ' ' ' ' ' ' ' ' ' '  |..|..|
  LEFT ---X.|..| ' ' ' ' CENTER RECT ' ' ' |..|.X--- RIGHT
  RECT   |..|..|  ' ' ' ' ' ' ' ' ' ' ' '  |..|..|   RECT
         |..|..| ' ' ' ' ' ' ' ' ' ' ' ' ' |..|..|
         +-----+---------------------------+-----+ <- bc_bot, ac_bot
            |                                 |
            |                                 |

                        |
                  X-- CELL BORDER
```

Example 31

The default amount for overlapping the cellborder for both standard and notched wells is one half the minimum geometry size for the well layer.

layer layer

Primary well layer. For a standard well, this is the layer of the single geometry; for a notched well, this is the layer for the three upper geometries. The default layer is n-well.

antilayer layer|none

Complementary layer for notched wells. This is not used for standard wells. This layer covers everything except for the primary well layer geometries. This may be turned off by specifying the value none. The default layer is pwell.

sgname name

Shape-geo name to give to geometry which is created. This may be specified for each of the notched well subshapes. If no value is defined the system will create an internal name which can not be user-accessed.

obj_names lef_name rig_name cen_name

Three geometric object names must be specified to create a notched well.

The order of the names is important: the first is the left shape, the second the right shape, and the third is the center shape. The names must be unique with respect to all other object names created in the cell template. The default names are: lef_layer, rig_layer, and cen_layer where layer is the value of the primary well layer attribute.

lef_name attributes

Placement attributes of shapes. Both pre-compaction and post-compaction values may be specified. If not specified, the values are taken from the corresponding well object values. The name values must match the values specified in the obj_names attribute (or the default values, if not defined). The before and after compaction specifications can reference the other notch pieces if they have already been defined.

rig_name attributes

Placement attributes of shapes. Both pre-compaction and post-compaction values may be specified. If not specified, the values are taken from the corresponding well object values. The name values must match the values specified in the obj_names attribute or the default values if not defined. The before and after compaction specifications can reference the other notch pieces if they have already been defined.

cen_name attributes

Placement attributes of shapes. Both pre-compaction and post-compaction values may be specified. If not specified, the values are taken from the corresponding well object values. The name values must match the values specified in the obj_names attribute or the default values if not defined. The before and after compaction specifications can reference the other notch pieces if they have already been defined.

Standard n-well which starts out covering from the top of the cell template down 20 microns from the top. After compaction the bottom edge must be exactly 4.50 microns from the cell border bottom edge.

```
pro_set obj_info add {nwell

   {proc        pro_well}

   {type        standard}

   {layer       nwell}

   {bc_bot      rel_obj cellborder top -20.00}

   {ac_bot      exact   cellborder bot   4.50 7}

  }
```

Example 32: Standard n-well

Using the example above, we create a notched well, which allows the well notch to float. We can also control all edges of each notch piece:

```
pro_set obj_info add {nwell
    {proc      pro_well}
    {type      notched}
    {layer     nwell}
    {obj_names lef_nwell rig_nwell cen_nwell}
    {lef_nwell
      {bc_lef  rel_obj cellborder lef -0.2}
      {bc_rig  rel_obj cellborder lef  0.2}
      {bc_top  rel_obj cellborder top  0.2}
      {bc_bot  rel_obj cellborder top 10.0}
      {ac_lef  follow cellborder lef}
      {ac_rig  follow cellborder rig}
      {ac_top  follow cellborder top}
      {ac_bot  exact cellborder top -4.2}
    }
    {rig_nwell
      {bc_lef  rel_obj cellborder rig -0.2}
      {bc_rig  rel_obj cellborder rig  0.2}
      {bc_top  rel_obj cellborder top  0.2}
      {bc_bot  rel_obj cellborder top 10.0}
      {ac_lef  follow cellborder lef}
      {ac_rig  follow cellborder rig}
      {ac_top  follow cellborder top}
      {ac_bot  follow lef_nwell  bot}
    }
    {cen_nwell
      {bc_lef  rel_obj lef_nwell  rig}
      {bc_rig  rel_obj rig_nwell  lef}
      {bc_top  rel_obj cellborder top  0.2}
      {bc_bot  rel_obj cellborder top 10.0}
      {ac_lef  follow lef_nwell  rig}
      {ac_rig  follow rig_nwell  lef}
      {ac_top  follow cellborder top}
      {ac_bot  none}
    }
  }
```

Example 34

pro_centering_obj

Creates a zero-area temporary object which will center itself in the cell along the defined axis. This can be used to drive other objects towards the center (like power rails in datapaths).

Object Type

All geometries created by this procedure are children of the calling object.

Attributes

layer

Layer of shape-geo (default = temp)

sgname

Shape-geo name (default = obj_centerline)

priority

Priority of forces which cause centering (default = 5)

weight

Weight of forces which cause centering (default = 1)

axis

Axis about which to center, values = X, Y, XY (default=XY)

xstep

Center step amount in X (default = celltemplate xgrid)

ystep

Center step amount in Y (default = celltemplate ygrid)

layer

Layer on which to place shape-geo. Any valid ProGen layer may be specified. The default layer is temp.

sgname

Shape-geo name to give to geometry which is created. The default name is obj_centerline where obj is the parent id.

priority [0-9]

Priority of centering forces. Default is 5.

weight *positive_integer*

Weight of centering forces. Default is 1.

axis

Axis about which shape-geo will be centered. Valid values are:

X

Center in X

Y

Center in Y

XY

Center in both X and Y (default)

xstep micron_value

Centering is accomplished by putting a series of stepping forces between the cellborder and the shape-geo. The stepping for each axis can be defined with these properties. The default values are to use the port gridding values (defined by the celltemplate xgrid/ygrid values).

ystep micron_value

Centering is accomplished by putting a series of stepping forces between the cellborder and the shape-geo. The stepping for each axis can be defined with these properties. The default values are to use the port gridding values (defined by the celltemplate xgrid/ygrid values).

User-Defined Flags

User-defined flags can be powerful tools for automating ProGen tasks. Use the pro_user_flag statement to define the flag name and the flag's default value:

```
   [pro_user_flag flagname default]
```

Defining Flags

The flag definition should be placed into the technology area file that corresponds to the flag's purpose. For example: if you wish to define a flag that will determine whether the cell height should be ten grids or nine grids, place the flag definition in the stdcell.tcl file, where the cell height is defined (note that each Y grid is two microns):

```
set ngrid [pro_user_flag arch 10grid]

if [strequ $ngrid 10grid] {

  # 10 grids

  pro_set obj_info add {cellborder

    {proc    pro_cellborder}

    {bc_rig  value 200.0}

    {bc_top  value 300.0}

    {ac_top  exact self bot 20.0}

    {ac_rig  pull self lef]

  }

} elseif [strequ $ngrid 9grid] {

  # 9 grids

  pro_set obj_info add {cellborder

    {proc    pro_cellborder}

    {bc_rig  value 200.0}

    {bc_top  value 300.0}

    {ac_top  exact self bot 18.0}

    {ac_rig  pull self lef}

  }

}
```

Calling Flags

When running ProGen, use the -f flag and specify the flag name and the value you wish to give the flag:

```
% progen  ... other options ...  -f flagname=value
```

For example: in Example 35 below, an inverter cell is run using the arch flag set to 10grid. In Example 36, the same .cel file is run with the arch flag set to 9grid, and the output is sent to a different file.

```
progen -t tech/progen.tcl -i inv.cel -o inv10.agd -f arch=10grid

progen -t tech/progen.tcl -i inv.cel -o inv9.agd -f arch=9grid
```

Expeditions

How can I set the option to make a metal1 rectangle?

To make a metal1 rectangle on input pins, you can use the following example construct in prospin.tcl:

```
prospin_set progen options {^MUX2HD1X$} {
    pro_set port_info add {S0 {vert_track 8} {horz_track 4}}
    ::progen::set_overlap_f m1 v1 {0.055 0.24 0.055 0.24} {net_name S0} {fixed 1}
}
```

Example 1

Where {0.055 0.24 0.055 0.24} means the left, bottom, right, and top sides of metal1 overlapping via1 values. Since via size is 0.19um and the min. metal1 area rule is 0.2 sq-um, you can define any combination of overlap values to come up with a metal1 shape >= 0.2 sq-um.

In this example, 0.055 + 0.19 + 0.055 = 0.30 (um wide), and 0.24 + 0.19 + 0.24 = 0.67 (um tall). The area of this metal1 is 0.30 * 0.67 = 0.201 sq-um.

In ./tech/progen.tcl, we can add a few more options which allow you to select fixed size of rectangles on all metal1 pin shapes:

```
#
# Force m1 overlap v1 to form a vertical rectangle and meet min. area rule
#
  if [pro_get option ud_force_vertical_v1_ov_m1_minarea 0] {
      ::progen::set_overlap_f m1 v1 {0.105 0.155 0.105 0.155} {fixed 1}
  }

  #
  # Force m1 overlap v1 to form a horizontal rectangle and meet min. area
rule
  #

  if [pro_get option ud_force_horizontal_v1_ov_m1_minarea 0] {
      ::progen::set_overlap_f m1 v1 {0.155 0.105 0.155 0.105} {fixed 1}
  }

  #
  # Force m1 overlap v1 to form a square and meet min. area rule
  #

  if [pro_get option ud_force_square_v1_ov_m1_minarea 0] {
      ::progen::set_overlap_f m1 v1 0.13 {fixed 1}
  }
```

To make use of them, apply one of the following controls in prospin.tcl:

```
prospin_set progen options {^<cellname>$} {
  pro_set option ud_force_vertical_v1_ov_m1_minarea 1
}

prospin_set progen options {^<cellname>$} {
  pro_set option ud_force_horizontal_v1_ov_m1_minarea 1
}

prospin_set progen options {^<cellname>$} {
  pro_set option ud_force_square_v1_ov_m1_minarea 1
}
```

You can define any size of rectangle as desired. If the shapes are not big enough to meeting metal1 min. area rule, progen will grow the shape automatically.

What grid offset should I use when using horizontal m2?

In our experience, the best thing to do when the left- and rightmost grids are not valid port locations because of the interaction between design rules and grid size is to make the X grid offset equal to 0 (i.e. the left and right edges of the cell are routing tracks).

The advantage of this approach is that the left- and rightmost grid locations within the cell will be valid port locations, so you still block the Y track above and below the port location. This will cause a lot of congestion around the ports and may impact routability. With the zero offset grid and horizontal metal2 around the vias, you will block the tracks to the left and right of the port location, so be sure to stagger ports; the 2D compactor will detect this and likely stagger anyway. However, the signals generally will be coming into the cell horizontally in metal2, so blocking the grid locations to the left and right of the port has much less impact than blocking the locations above and below the port.

We have seen other libraries with full grid offsets, so you should not have a problem with the place and route tools. Just make sure you tell the P&R tool about the offset so it doesn't think that all ports are off-grid. (Even in that case, that is likely all right because a port would only block two grids – half a grid in either direction – rather than three grids, so it is possible that it is even better to keep the routing grid at half grid offset at the block level and 0 offset at the cell level.)

Also, you'll want to make sure that the ProGenesis port orientation and the P&R vias' orientations are consistent so the P&R tool has room to place the via. In this case, both should be horizontal for m2.

How do I control the number of contact cuts?

You may grow specific number of contacts at particular nodes. Two settings are involved with this control, "net_info" and "char_info".

```
pro_set net_info add {<net_name>
```

```
        {contact_char

            {convdd {vdd <pwr_con_name>}}

            {congnd {gnd <gnd_con_name>}}

            {condifn {output <diff_con_name1>} {input <diff_con_name2>}

                {internal <diff_con_name3>}}

            {condifp {output <diff_con_name1>} {input <diff_con_name2>}

                {internal <diff_con_name3>}}

        }

    }

    pro_set char_info add {

        {<pwr_con_name>

            {min_<dir>_cuts {cndiff <ncuts>} {cpdiff <ncuts>}}

            {min_<dir>_cuts_priority {cndiff <priority>} {cpdiff <priority>}}

            {min_<dir>_cuts_weight {cndiff <priority>} {cpdiff <priority>}}

            {max_<dir>_cuts {cndiff <ncuts>} {cpdiff <ncuts>}}

            {max_<dir>_cuts_priority {cndiff <priority>} {cpdiff <priority>}}

            {min_<dir>_cuts_weight {cndiff <priority>} {cpdiff <priority>}}

        }

        {<gnd_con_name>

            ...

        }

        {<diff_con_name1>

            ...

        }

        {<diff_con_name2>

            ...
        }

        {<diff_con_name3>

            ...

        }

    }
```

where

```
   <net_name> Net name, such as "VDD", "GND", "A",...
```

"PG_DEFAULT" can be used to include all net names.

```
    <pwr_con_name>
    <gnd_con_name>
    <diff_con_name1>
    <diff_con_name2>
    <diff_con_name3>
```

are contact characteristic names. They are identifiers.

Each name should have a corresponding specification in "char_info" section to detail how many contact cuts to be added for the net(s) and how important to add so many cuts.

<dir> Contact growing direction, either "x" or "y".

"x" means growing contacts side way, along X axis.

"y" grows contacts vertically, along Y axis.

<ncuts> Number of cuts, any integer number > 1.

<priority> Importance of the constraint, value can be from 1 to 9.

Value > 5 will compete with cell area minimization force, so don"t specify value > 5 unless it is really necessary.

<weight> Weight of the corresponding constraint, value is any integer number > 1.

```
   max_x_cuts

      max_y_cuts

      min_x_cuts

      min_y_cuts
```

These parameters specify the maximum allowed or minimum required number of horizontal (x) or vertical (y) contact cuts. Default is "no constraint."

```
max_x_cuts_priority

   min_x_cuts_priority

   max_y_cuts_priority

   min_y_cuts_priority
```

These parameters specify the priority of their corresponding maximum or minimum horizontal (x) or vertical (y) cut constraint. Default is 7.

```
max_x_cuts_weight

   max_y_cuts_weight

   min_x_cuts_weight

   min_y_cuts_weight
```

These parameters specify the weight of their corresponding maximum or minimum horizontal (x) or vertical (y) cut constraint. Default is 1.

For example: to have maximum 2 minimum 1 contact cuts on all the power/ground connections in Y direction, 2 contact cuts on all output nodes in Y direction, single cut on all input and internal nodes, specify the constraints as follows:

```
pro_set net_info add {PG_DEFAULT
        {contact_char
                 {convdd {vdd pwr_con}}
                 {congnd {gnd gnd_con}}
                 {condifn {output output_cdiff} {input internal_cdiff}
                          {internal internal_cdiff}}
                 {condifp {output output_cdiff} {input internal_cdiff}
                          {internal internal_cdiff}}
        }
    }

    pro_set char_info add {
        {pwr_con {min_y_cuts {cpdiff 2}}
                 {max_y_cuts {cpdiff 2}}
                 {min_x_cuts {cpdiff 1}}
                 {max_x_cuts {cpdiff 1}}}
        {gnd_con {min_y_cuts {cndiff 2}}
                 {max_y_cuts {cndiff 2}}
                 {min_x_cuts {cndiff 1}}
                 {max_x_cuts {cndiff 1}}}
        {output_cdiff
{min_y_cuts {cndiff 1} {cpdiff 1}}
                 {max_y_cuts {cndiff 2} {cpdiff 2}}
                 {min_x_cuts {cndiff 1} {cpdiff 1}}
                 {max_x_cuts {cndiff 1} {cpdiff 1}}}
        {internal_cdiff
{min_y_cuts {cndiff 1} {cpdiff 1}}
                 {max_y_cuts {cndiff 1} {cpdiff 1}}
                 {min_x_cuts {cndiff 1} {cpdiff 1}}
                 {max_x_cuts {cndiff 1} {cpdiff 1}}}
    }
```

How do I use ProGen objects and SOTs?

The initial layout that ProGen creates is based on the placement of objects. Objects include transistors, wires, transistor-contacts, wire-contacts, and ports. These objects create the actual geometries for each object type and define design rule relationships (forces) for the geometries in the object.

Objects are implemented as Tcl procedures. A single object procedure may produce varying flavors of the same object based on the arguments provided for the object procedure and the system settings.

The definition of an object procedure and its arguments is called a "Symbolic Object Template" (SOT). A SOT is merely a definition and it is only used when it is mapped to specific ProGen object and ProGen uses the object.

Users can create new SOT definitions or override existing definitions. They can then map the SOT definitions to the ProGen objects.

Creating a SOT Definition

The following three statements are used to create a new SOT definition:

```
pro_set  sot  <SOT-NAME>       <SOT-TYPE>

   pro_set  sot  <SOT-NAME>_proc  <PROCEDURE>

   pro_set  sot  <SOT-NAME>_args  <ARGUMENTS>
```

1) The first statement identifies the name of the SOT and the type of object it represents. SOT object types are predefined in ProGen.

The <SOT-TYPE> identifies the general class of a SOT. It indicates if it is a transistor, a transistor diffusion power/ground contact, a horizontal route, a port, etc.

2) The second statement identifies the Tcl procedure name that is used to create an instance of the object in the layout.

3) The third statement identifies argument values passed to the object Tcl procedure.

For example: let’s create a SOT definition for a fictitious transistor rubber ground contact object:

```
pro_set  sot  congnd_rubber_sot       gsot_named_contact

pro_set  sot  congnd_rubber_sot_proc  rubber_contact

pro_set  sot  congnd_rubber_sot_args  {gnd rubber}
```

1) The first statement names the SOT "congnd_rubber_sot" and identifies its type as a "gsot_named_contact". Transistor power/ground diffusion-node contacts are typically of SOT-TYPE "gsot_named_contact".

2) The second statement identifies the Tcl procedure "rubber_contact" as the procedure to call to create an instance of the object.

3) The third statement identifies argument values passed to the SOT Tcl procedure.

At this point the SOT is defined, but it is not mapped to any specific ProGen object.

Mapping SOT to ProGen Object

To have ProGen actually use a SOT to create an object the SOT definition must be mapped to a ProGen object name. This is done by using the following statement:

```
pro_assign_sot  <OBJECT>  <SOT-NAME>  <OPTIONAL-FLAG>
```

Where <OBJECT> is the name that ProGen associates with a specific object. The <OBJECT> value may depend on system option settings. For example: when transistor bent gates is activated the n-transistor <OBJECT> is "tranbendn" instead of "trann". Also, many objects can have their <OBJECT> value explicitly set in ProSticks.

Some of the most common ProGen object names include:

trann - n-diffusion transistor

tranp - p-diffusion transistor

tranbendn - n-diffusion bent-gate transistor

tranbendp - p-diffusion bent-gate transistor

congnd - transistor ground contact

convdd - transistor vdd contact

condifn - transistor non-ground n-diffusion contact

condifp - transistor non-ground p-diffusion contact

The <SOT-NAME> is the name used to lookup the corresponding SOT definition.

The <OPTIONAL-FLAG> indicates if the mapping should override an existing mapping for the <OBJECT>. It may have the value "override" or "nooverride".

For example: to assign the "congnd_rubber_sot" SOT definition for every instance of a transistor ground node:

```
pro_assign_sot congnd congnd_rubber_sot override
```

Modifying an Existing SOT Definition

Users rarely create new SOT definitions to add to the system. More typically they modify an existing definition by changing the values of its SOT procedure arguments.

For example: the following is the default SOT definition for a transistor ground-contact dmwire:

```
pro_set  sot  congnd_dmwire_sot       gsot_contact

pro_set  sot  congnd_dmwire_sot_proc  tech_dcon_to_tie_dmwire

pro_set  sot  congnd_dmwire_sot_args  {gnd m1 atrail}
```

The "atrail" argument value indicates that the diffusion-wire/m1-wire contact should be initially positioned at the ground rail. To change it to be at the midpoint between the transistor and the rail we would specify the following:

```
pro_set  sot  congnd_dmwire_sot_args  {gnd m1 midpoint}
```

When a definition has been modified, it must be remapped to the internal database.

```
pro_assign_sot  congnd  congnd_dmwire_sot  override
```

Note the "override" `<OPTIONAL-FLAG>` value to force it to override the system default settings.

How do I use the dmwire?

To specify the dmwire, set the following in "proc objects" in progen.tcl:

```
pro_set sot congnd_dmwire_sot_args {gnd m1 midpoint}

pro_assign_sot congnd congnd_dmwire_sot override

pro_set sot convdd_dmwire_sot_args {vdd m1 midpoint}

pro_assign_sot convdd convdd_dmwire_sot override
```

To set to an all-diffusion routing (no diffusion contact) to connect by abutment to the diffusion of the well/substrate ties:

```
pro_assign_sot congnd condifntie_sot override

pro_assign_sot convdd condifptie_sot override
```

If there are problems, adjust the bc (before compaction) values in stdcell.tcl to be more reasonable. When in doubt, make the bc numbers larger -- the compactor will take care of compacting the cell down to its final value. The only requirement is that the placement and routing routines (before compaction) have plenty of room to work with.

Next, run the cell with y-compaction only (no gridding):

```
%
```

```
   progen -t mytech/progen.tcl -i cel/invx1.cel \

          -o agd/invx1.agd.ysg -f co=y_skip_grid >&! log/invx1.log.ysg
```

Review the layout. Check for the following:

The y-values of the cellborder, power rails, well, implants and ties are all at their correct value.

- The "Initial cell cost for this direction" is priority 6.
- The "Final cost for last YAXIS compaction" is priority 5.
- The dmwire diffusion contact is near the rails (for delayed or delayed_reverse), or by the transistor (for 1).
- If the cell doesn't fit into the cell-height, run compaction again, this time with y_fast compaction:

```
%

progen -t mytech/progen.tcl -i cel/invx1.cel \

       -o agd/invx1.agd.yfa -f co=y_fast >&! log/invx1.log.yfa
```

Then review the result with proview. The critical path is highlighted as part of the critpath layer. For more detail, see Strategy for Making Cells Fit into the Cell Height in the “Library Construction Guide” section. Once the layout is perfect, you are ready to run ore cells through. If you are using revision control software, this is a good time to check in your work before proceeding with more complex cells.

ProGen FAQ

Regarding butted diffusion as an electrical connection: I see that sometimes there is no diffusion contact for the transistor source (invx2, nand2x4), and instead, the butted diffusion tie is somehow supposed to provide this connection. How does this work?

The connection made by butted diffusion is actually done through the silicide on top of the diffusion (i.e. metal 0). The diffusion makes a reverse-biased diode, so it provides no current. In some technologies, the p/n junction creates some deformation of the silicide, so they are either not recommended or not legal. However, in some technologies, the connection works great and is therefore allowed. The advantage of that type of connection is that you can sometimes save area over using a contact. So either your design-rules allow that type of connection or they don't (or they allow it but prefer you don't make it). ProGenesis allows you to build it either way (or you can try both and see what the area impact is).

How do I control the number of contact cuts?

See this topic in the ProGen “Expeditions” section.

How do I deal with transistor width tolerance? How does the tool interpret tolerance? For example, if a tolerance of 0.05u is given, does the tool find the least width delta to be equal fingers? How about LVS for changed widths?

By default, the total width of transistor fingers (legs) of a gate should be equal to the width in input spice netlist. This means that "transistor_width_tolerance" is '0.0'. If "transistor_width_tolerance" is set to "0.05", the maximum width error allowed after folding is 0.05um. In this case, the total width of the fingers may not be the same as the input.

What grid offset should I use when using horizontal m2?

In our experience, the best thing to do when the left- and rightmost grids are not valid port locations because of the interaction between design rules and grid size is to make the X grid offset equal to 0 (i.e. the left and right edges of the cell are routing tracks).

The advantage of this approach is that the left- and rightmost grid locations within the cell will be valid port locations, so you still block the Y track above and below the port location. This will cause a lot of congestion around the ports and may impact routability. With the zero offset grid and horizontal metal2 around the vias, you will block the tracks to the left and right of the port location, so be sure to stagger ports; the 2D compactor will detect this and likely stagger anyway. However, the signals generally will be coming into the cell horizontally in metal2, so blocking the grid locations to the left and right of the port has much less impact than blocking the locations above and below the port.

We have seen other libraries with full grid offsets, so you should not have a problem with the place and route tools. Just make sure you tell the P&R tool about the offset so it doesn't think that all ports are off-grid. (Even in that case, that is likely all right because a port would only block two grids – half a grid in either direction – rather than three grids, so it is possible that it is even better to keep the routing grid at half grid offset at the block level and 0 offset at the cell level.)

Also, you'll want to make sure that the ProGenesis port orientation and the P&R vias' orientations are consistent so the P&R tool has room to place the via. In this case, both should be horizontal for m2.

How do I turn off port gridding while keeping the cell border on grid?

Turning off port gridding in X or Y direction (or both) is the same as placing ports on manufacturing (database) grids.

By default, progen shares cellborder grids with port grids which are defined as "xgrid" and "ygrid" in "celltemplate" object. If you use the ProTech window to define a Cell Template and you don't have "Additional values file" specified in the Cell Template form, you will have to use the progen command switch "-ct" to write out the Cell Template information to a file in which you will find the "celltemplate" object along with "xgrid" and "ygrid" fields in it. Once you finish modifying the contents of the Cell Template file, remember to specify the Cell Template file name as the value of "Additional value file" in ProTech's Cell Template form.

In "celltemplate" object, the values of "xgrid" and "ygrid" are the routing grids (pitches) in X and Y directions, respectively. Ports are placed on these grids. To turn off port gridding, simply change the value of "xgrid" or "ygrid" to the manufacturing grid in the object specification.

There is another object called "cellborder" in the Cell Template file. By default, there is no "xgrid" or "ygrid" entry in the object. That means “cellborder” will be placed on the routing grids defined in the "celltemplate" object. If you change the "xgrid" or "ygrid" value in the "celltemplate" object to the manufacturing grid to turn off port gridding, you need to add the "xgrid" or "ygrid" entry to the "cellborder" object and set the desired cell border grid number as value.

For example: let's say you want to turn off port gridding in X direction, but want to enforce port gridding in Y direction with grid size of 0.24um. You also want to make sure cell border is placed on X-grid of 0.28um and Y-grid of 0.24um; and the manufacturing grid size is 0.005um. Here is what you would have related to the settings in the Cell Template file:

```
{celltemplate

    {proc pro_celltemplate}

    {border cellborder}

    {xgrid 0.005}          <===

    {ygrid 0.24}

    {xgrid_lo 0.00}

     ...

}
```

```
{cellborder

    {proc pro_cellborder}

    {bc_rig value <value>}

    {bc_top value <value>}

    {xgrid 0.28}            <===

    {ac_rig pull self lef}

    ...

}
```

What is the difference between series folding and parallel folding?

Let's use nand2 -- a simple two-input NAND gate -- as example to describe series and parallel gate folding. In a nand2 layout, the n-fets are series gates (no source/drain contact between two poly-gates), while the p-fets are parallel gates (contain s/d contacts between poly-gates).

If you apply the following control to nand2 with inputs "a" and "b", and the order of the inputs are "a b" before folding, from left to right:

```
pro_set topology fold_ser_in_place 0

   pro_set topology fold_par_in_place 1
```

This will fold series-stacks (n-gates) as stacks, so the input gate order after folding on the n side becomes "a b b a". However, on the p side the parallel-stacks are folded in-place individually, which makes the input gate order be "a a b b".

"set fold_xxx_in_place 0" causes folding transistor stacks (either series-stacks or parallel-stacks) as stacks, so the input order becomes "a b c c b a a b c ..." given a three-input cell, while "set fold_xxx_in_place 1" results in folding transistor stacks in place, such that the input becomes "a a ... b b ... c c ..."

What is the method to determine the maximum n- or p-width to fit in given cell template?

One easy way is to get the values is by measuring the size p- and n-gates from the results of executing Run Parameter Generation Script in ProTech window under the Router tab. Measure the p-gate width in "max_outer_poly_diffnet_row0" for max p, and the n-gate width in "max_outer_poly_diffnet_row1" for max n-width.

ProGenesis Design for Manufacturability

DFM Features

The following DFM features are found in the 4.3 release of the ProGenesis software:

Forbidden spacing design rules

- Width-dependant spacing design rules
- Length of transistor gate extensions (endcap) dependant on spacing to diffusion design rule
- Preferred rules
- Distribute preferred rules option
- Automatically determine metal overlap orientation of contacts and vias
- Straighten transistor gates
- Forbidden spacing design rules

Forbidden spacing rules define ranges of separation that geometries on the same layer are not allowed to occupy.

Greater than	Less than	Meaning
0.10	0.15	forbidden range = 0.10 < spacing < 0.15
0.32	0.40	forbidden range = 0.32 < spacing < 0.40

Table 1: Illustration of a rule with two forbidden spacing ranges defined

Feature Locations

```
ProTech Tab: Rules

      Group: Diffusion Rules -- Design For Manufacturing

      Rule:  Diffusion forbidden spacing

      Group: Poly Rules -- Design For Manufacturing

      Rule:  Poly forbidden spacing

      Group: Gate -- Design For Manufacturing

      Rule:  Gate forbidden spacing

      Group: Metal-1 Rules -- Design For Manufacturing

      Rule:  Metal-1 forbidden spacing

      Group: Metal-2 Rules -- Design For Manufacturing
```

```
Rule:  Metal-2 forbidden spacing
```

Width-dependent spacing design rules

Width-dependent spacing rules define minimum separations for geometries on the same layer that are dependent on the width of the geometries. If either of the geometries are greater than or equal to the width, then the minimum spacing is applied.

Table 2 illustrates a rule with three width-dependent spacing rules defined.

Minimum width	Minimum spacing	Meaning
0.50	0.45	enforce 0.45 minimum spacing when either width ≥ 0.50
0.70	0.65	enforce 0.65 minimum spacing when either width ≥ 0.70
0.90	1.00	enforce minimum spacing when either width ≥ 0.90

Table 2. Width-Dependent Spacing

There is an optional third parameter for width-dependent rules: the threshold. The threshold is measured in the axis orthogonal to the spacing and indicates the minimum length that geometries must overlap before the width-dependent spacing rule is applied.

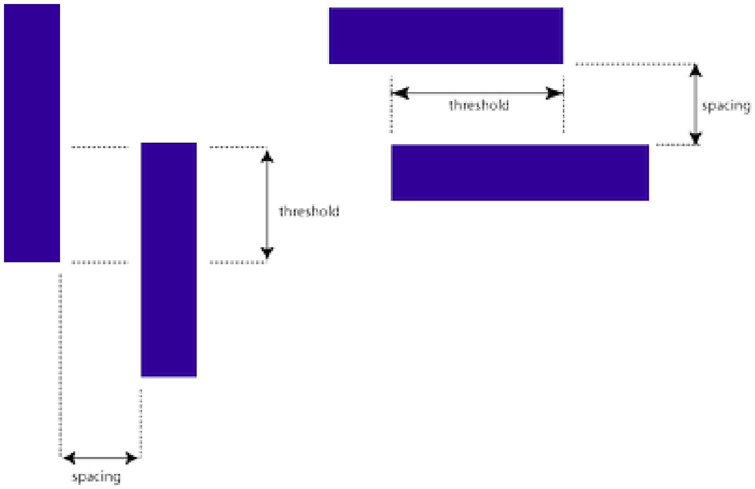


Figure 1. Threshold

Table 3 illustrates a rule with three width-dependent spacing rules defined.

Minimum width	Minimum spacing	Threshold	Meaning
0.50	0.40	0.25	enforce 0.40 minimum spacing when either width ≥ 0.50 and the threshold > 0.25

Table 3. Width-Dependent Spacing

Minimum width	**Minimum spacing**	**Threshold**	**Meaning**
0.50	0.60	0.50	enforce 0.60 minimum spacing when either width ≥ 0.50 and the threshold > 0.50

Table 3. Width-Dependent Spacing

Width-Dependant End of Line Spacing

```
pro_set option width_dependent_end_of_line_spacing {

   on|off
   {
   layer     <layer-name>
   min_width <microns>
   max_width <microns>
   spacing   <microns>
   priority  7
   weight    1
   }

   }
```

This option is used to define width-dependent end-of-line spacings to other geometries on the same layer. In other words, the spacing from the route end-of-line is dependent on the width of the route.

```
     |
     v                 |//|
    ---------------+    |//|
   width  ///////////| <-spacing-> |//|
    ---------------+    |//|
     ^                 |//|
     |
```

Figure 2

The spacing rule is applied when the width of the route is greater than or equal to the minimum width (min_width value and less than the maximum width (max_width) value.

when (min_width >= width < max_width) apply spacing

This allows you to have multiple ranges of values for the same layer.

Parameters:

`on|off` - Turns option on or off
`layer` - ProGen layer name (no default)
`min_width` - Minimum width of route for spacing to be applied
`max_width` - Maximum width of route for spacing to be applied
`spacing` - Spacing design rule value in microns (no default)

`priority` - Priority of spacing forces (default = 7)
`weight` - Weight of spacing forces (default = 1)

The block of layer parameter values may be specified as many times as needed.

To change the compaction pass that the option is activated use the following command:

```
pro_set option width_dependent_end_of_line_spacing_start_index <pass>

<pass> = the compaction pass number
```

```
      # Set width-dependent eol-spacing for m1 routes

pro_set option width_dependent_end_of_line_spacing {on
   {layer      m1
    min_width  0.12
    max_width  0.32
    spacing    0.16
   }
}
```

Example 1

```
# Set width-dependent eol-spacing for m1 and m2 routes

pro_set option width_dependent_end_of_line_spacing {on
   {layer      m1
    min_width  0.12
    max_width  0.32
    spacing    0.16
   }
   {layer      m2
    min_width  0.12
    max_width  0.32
    spacing    0.16
   }
}
```

Example 2

```
# Set width-dependent eol-spacing for 2 ranges of m1 route widths

   pro_set option width_dependent_end_of_line_spacing {on
      {layer     m1
       min_width  0.12
       max_width  0.32
       spacing    0.16
      }
      {layer     m1
       min_width  0.60
       max_width  0.80
       spacing    0.20
      }
   }
```

Example 3

Feature Locations

```
ProTech Tab: Rules

   Group: Diffusion Rules -- Design For Manufacturing
   Rule:  Diffusion width-dependent spacing
   Group: Poly Rules -- Design For Manufacturing
   Rule:  Poly width-dependent spacing
   Group: Gate -- Design For Manufacturing
   Rule:  Gate width-dependent spacing
   Group: Metal-1 Rules -- Design For Manufacturing
   Rule:  Metal-1 width-dependent spacing
   Group: Metal-2 Rules -- Design For Manufacturing
   Rule:  Metal-2 width-dependent spacing
```

Length of transistor gate extensions (endcap) dependent on spacing to diffusion design rule

This rule defines a relationship between the length of a transistor gate extension and its spacing to diffusion.

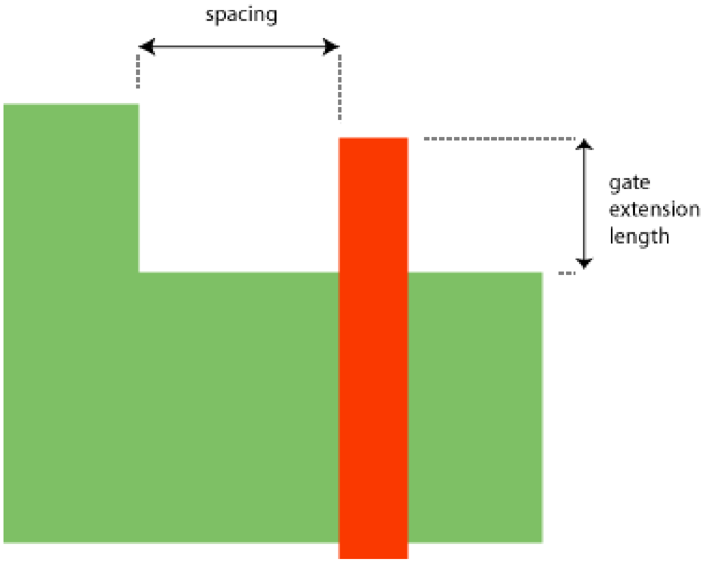


Figure 3. Endcap Length

Spacing less than	Endcap length	Meaning
0.15	0.10	increase spacing to 0.15 or endcap length to 0.10 when spacing <0.15 and endcap length <0.10
0.20	0.12	increase spacing to 0.20 or endacp length to 0.12 when 0.15 ≤ spacing < 0.20 and 0.10 ≤ endcap length ≤ 0.10
ProTech Tab: Rules Group: Poly Rules -- Design For Manufacturing Rule: Poly endcap length / diffusion spacing		

Table 4: Endcap length

Preferred rules

Preferred rules are design rules that are applied after all mandatory rules are satisfied. Preferred rules are not mandatory. If they can be satisfied, they will be. They will not cause the cell size to increase. If they cannot be satisfied without increasing the cell size, then they will remain unsatisfied.

There are three primary types of preferred rules:

Preferred spacing design rules

```
ProTech Tab: Rules
```

All spacing rules allow the specification of a preferred spacing value in the "Preferred Value" column.

Preferred width design rules

```
ProTech Tab: Rules
```

All width rules allow the specification of a preferred width value in the "Preferred Value" column.

Preferred overlap design rules

```
ProTech Tab: Rules
```

All overlap rules allow the specification of a preferred overlap value in the "Preferred Value" column.

Distribute preferred rules option

When ProGen cannot satisfy all of the preferred rules for a cell, it can take two basic approaches to resolve as many as possible: the default summation approach and the distributed approach.

The default approach is based purely on a mathematical summation of the final result. This is fast and easy to evaluate. Figure 4 reflects layout that might result when the minimum spacing is 0.10 and the preferred spacing is 0.50.

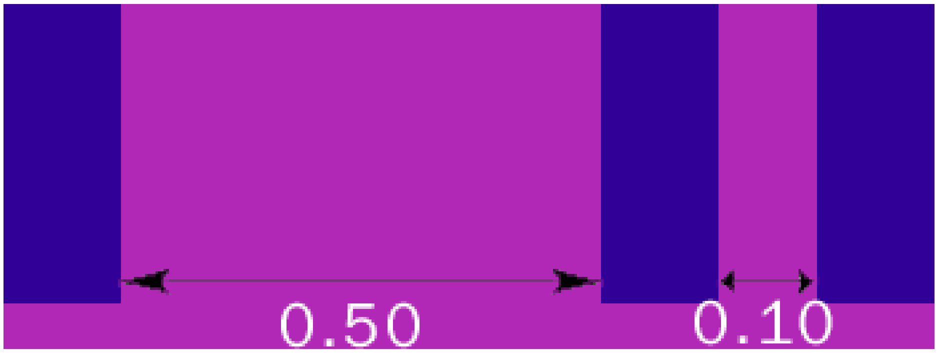


Figure 4. Default Summation Approach

In the summation approach, the result may be any of the following combinations (assuming a 0.10 grid): (0.5, 0.1), (0.4, 0.2), (0.3, 0.3), (0.2, 0.4), or (0.1, 0.5). All of these combinations add up to 0.6.

The alternative approach is to evenly distribute the amount at which all preferred rules are satisfied. Using a distributed approach, the layout in Figure 5 would be more likely to result from compaction.

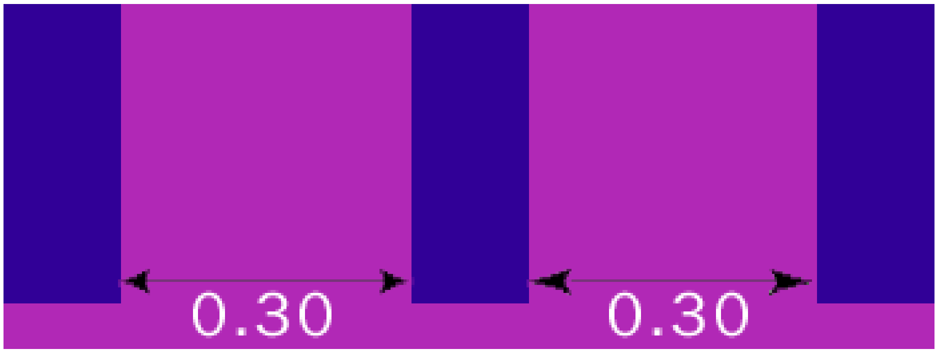


Figure 5. Distributing Preferred Rules

Due to the numerous factors in effect, the distributed approach does not guarantee evenly distributed results, but it makes distributed results more likely. The distributed approach can take longer to run than the summation approach.

The approach the system takes can be controlled for any of the three primary types of preferred rules.

Distributed preferred spacings option

```
ProTech Tab: Options
      Group: Preferred Rule Options
      Rule:  Distribute preferred spacings
```

Distributed preferred widths option

```
ProTech Tab: Options
      Group: Preferred Rule Options
      Rule:  Distribute preferred widths
```

Distributed preferred overlaps option

```
ProTech Tab: Options
      Group: Preferred Rule Options
      Rule:  Distribute preferred overlaps
```

Automatically determine metal overlap orientation of contacts and vias

In the previous releases of the ProGen software, the orientation of asymmetric overlaps was determined by the order of the values defined for the overlap rules. All the overlaps controlled by a given rule would have the same orientation in a cell. For example: if the rule given is 0.10 0.20 0.10 0.20, the overlaps would always look like the one in Figure 6.

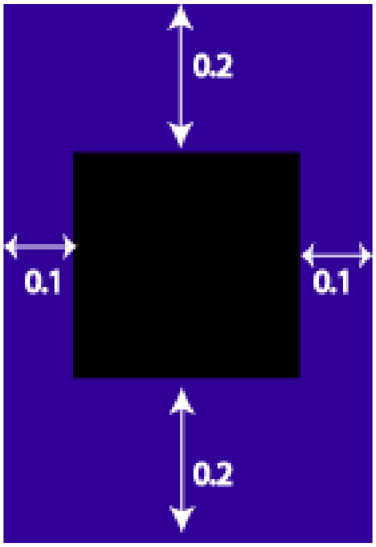


Figure 6. Metal Overlap Orientation

This control allows ProGen to decide which overlap orientation could produce the best compaction results on a per-overlap basis. Each overlap instance is evaluated separately so some contacts may have a vertical metal-1 overlap orientation and some may have a horizontal orientation.

```
ProTech Tab: Options

         Group: Compaction Related Options
         Rule:  Automatic overlap orientation
```

Straighten transistor gates

The ProGen implementation of bent gates has always attempted to straighten the gates if possible. It will only straighten gates if doing so does not cause the cell size to increase. The following ProTech option turns bent gates on and off:

```
ProTech Tab: Options

         Group: General Options
         Rule:  Transistor gate bending
```

Multiple contacts where possible

ProGen has always supported a wide range of features for contacts. In particular, the transistor diffusion contacts offer several features, including redundant contacts versus single contacts. The system can be directed to create connections with a single contact, to always put in as many contacts as will fit in the transistor width, to put in as many contacts as will fit in a percentage of the transistor width, and to put in as many contacts as will fit in the transistor width without sacrificing compaction results.

In the last option, the system starts out with a single contact at every connection, and then adds in contacts during compaction as space permits. This is the most widely used method for multiple contact control.

```
ProTech Tab: Options

        Group: Compaction Related Options
        Rule:  Diffusion contact growing
```

Finding DFM Features

Most of the DFM features can be controlled via settings in ProTech (comprehensively described in ProTech Guide).

1. ProSpin Guide

1.1 Introduction to ProSpin

ProSpin is a very powerful tool from which the library building process begins. It processes an input Spice net-list for use in creating one or more cell layouts using ProGen.

ProSpin examines each circuit in the net-list, identifies known topologies, and maps those topologies to one or more ProGen generators.

ProSpin may also be used to assign the mapping from a subcircuit topology to one or more ProGen generators; net-list patterns may be custom-mapped to generators to optimize the layout.

The ProSpin technology file **prospin.tcl** allows flexible registration and preservation of CEL file customizations so the output can be quickly regenerated with all modifications intact.

ProSpin also provides many useful net-list transformations including device size scaling and node name modification.

The input to ProSpin is a Spice file describing the desired circuit net-list. In general, a Spice file may specify a net-list in terms of zero or more subcircuits using the Spice **.subckt** statement.

Given a Spice file, ProSpin allows users to process the top-level net-list and any number of the defined subcircuits. A separate CEL description is output for the top-level net-list and for each subcircuit requested. The top-level net-list, if it exists, is always processed. Subcircuits are processed only if requested specifically by name.

Spice is case-insensitive. ProSpin ignores case as required by Spice, but preserves in the output any case found in the input. If multiple versions of an identifier are found that differ only with respect to case, the first version encountered will be used in the output.

1.2 Order of address

You have several ways to access instruction you need to get the tool running:

Quick Start
For those with some ProSpin knowledge who want to start immediately

Guided Tour
Detailed, step-by-step instructions for using ProSpin for most applications

Expeditions
Detailed, step-by-step instructions for specialty use of ProSpin

If you're already familiar with ProSpin and its GUI, jump to the Quick Start guide. Within a few pages, you can be on the way to great results in automated library construction.

Need a little more background? The Guided Tour offers direction for common use of ProSpin, but in a more expository fashion than the Quick Start guide. This section is user-friendly for novices and experienced users alike.

For those who want further coaching, Expeditions addresses specialty needs -- many of them culled from the files of our customer support team. If you have what looks like a unique challenge, check the Expeditions and FAQ sections; you may find the resources you need to keep your production apace.

We also offer the ProSpin Reference Guide as an appendix you may find helpful.

If you've consulted these sections and still have a need, please write to us at applications@prolificinc.com, and we'll quickly respond.

2.Quick Start

2.1 Create a subdirectory in the working directory for the input Spice net-lists.

```
% mkdir ./net
```

2.2 Copy all cell net-lists into **./net**

2.3 Start ProSpin.

```
% prospin -d cel -g &
```

2.4 Click **File->Load Spice...**

2.5 Select all the net-list files.

In the **Spice** selection window, double click **Net** under the **Directories** section and then select all net-list files in the **Files** section by clicking on the first net-list file name, dragging through the whole list, and releasing. Click **OK**.

2.6 Click **Run All**

The **./cel subdirectory** will be populated with the .cel files corresponding to the subcircuits in the selected net-list files.

2.7 Click **File->Exit**

Common mistakes/problems

- Using the ProGen technology file instead of the ProSpin technology file. The ProSpin technology file is a separate and distinct file.
- Not specifying the -l option to select a subcircuit when the given Spice net-list does not contain a top-level circuit.
- Input/output port patterns not correct for your library.
- ...need more instruction? Continue on to the Guided Tour.

3. Guided Tour

3.1. Interactive mode

3.2. Batch mode

3.2.1. Top-level net-list examples

In its simplest form, using all the default options, ProSpin may be run with a Spice file name as its single argument.

Suppose the file NAND2.sp contains the following:

```
M0 VSS A n1 VSS N W=1.4U

M1 n1 B OUT n1 N W=1.4U

M2 OUT A VDD VDD P W=2.0U

M3 OUT B VDD VDD P W=2.0U
```

To generate the corresponding CEL file on stdout, use

```
% prospin NAND2.sp
```

To change the cell name from NAND2 to ND2, use

```
% prospin -N ND2 NAND2.sp
```

To put the CEL description in the file ND2.cel instead of on stdout, use

```
% prospin -N ND2 -O ND2.cel NAND2.sp
```

3.2.2. Subcircuit examples

Often, the Spice files of interest only contain subcircuit definitions. Suppose NAND2.sp contains

```
.SUBCKT NAND2 A B OUT

M0 VSS A n1 VSS N W=1.4U

M1 n1 B OUT n1 N W=1.4U

M2 OUT A VDD VDD P W=2.0U

M3 OUT B VDD VDD P W=2.0U

.ENDS
```

Since many subcircuits may be requested at once, ProSpin always outputs CEL subcircuit descriptions in files. The default file name is formed by suffixing the subcircuit name with .cel. To select subcircuits by name from the command line, use the -l option:

```
% prospin -l NAND2 NAND2.sp
```

This command produces a file called "NAND2.cel".

Note that since Spice is case-insensitive, this subcircuit could also have been selected using:

```
% prospin -l NAND2 NAND2.sp
```

The difference is that ProSpin would now use NAND2 as the subcircuit name and the output file (called NAND2.cel) would specify the cell name as nand2 instead of NAND2.

Options are provided to change the location of the output CEL file. For example, a common methodology is to collect all the CEL files for a given library into a subdirectory called “cel”. This is conveniently accomplished using the -d option. The command

```
% prospin -d cel -l NAND2 NAND2.sp
```

generates a NAND2.cel file in the cel subdirectory of the current working directory (cel/NAND2.cel).

Alternately, the more general -o option may be used to format the .cel file name. The -o option takes a Tcl format string as its value. The format string may contain a single “%s” that will be replaced with the subcircuit name.

For example: to put the NAND2 .cel description into the file baz/FOO_NAND2.cel, use

```
% prospin -o baz/FOO_%s.cel -l NAND2 NAND2.sp
```

Note that the option -d cel is equivalent to the option -o cel/%s.cel.

As with the -o option, the -n option may be used to format the cell name specified to progen in the .cel file. If the cell name used by ProGen should be MY_NAND2 instead of NAND2, then use:

```
% prospin -n MY_%s -l NAND2 NAND2.sp
```

Spice files often contain multiple subcircuit definitions. The –l option may be used to select more than one subcircuit by setting the option value to a comma-separated list of subcircuit names. For example, suppose the file cells.sp contains a NAND2 subcircuit definition plus another definition for an inverter called INV:

```
.SUBCKT NAND2 A B Out

M0 VSS A n1 VSS N W=1.4U

M1 n1 B Out n1 N W=1.4U

M2 Out A VDD VDD P W=2.0U

M3 Out B VDD VDD P W=2.0U

.ENDS

.SUBCKT INV IN Out

M0 VSS IN Out VSS N W=1.4U
```

```
M1 Out IN VDD VDD P W=2.0U
.ENDS
```

Now, to generate .cel files for both NAND2 and INV with a single command line, use

```
% prospin -l INV,NAND2 cells.sp
```

Alternately, the list of requested subcircuits may be specified in a file. If the file list contains

```
NAND2
INV
```

then the command line

```
 % prospin -s list cells.sp
```

will also generate the two requested .cel files.

3.2.3. Miscellaneous examples

The following examples apply to both Spice files with top-level circuits and Spice files with subcircuits.

Suppose the file INV.sp contains

```
.SUBCKT INV IN Out
M0 vss! IN Out vss! N W=1.4U
M1 vss! IN Out vss! N W=1.4U
M2 Out IN vdd! vdd! P W=2.0U
M3 Out IN vdd! vdd! P W=2.0U
.ENDS
```

To output all Spice node names in upper case, use

```
% prospin -C -l INV INV.sp
```

To output all Spice node names in lower case, use

```
% prospin -C -l INV INV.sp
```

To ignore the transistor folding specified in the Spice file and have the generators calculate the folding, use

```
% prospin -f -l INV INV.sp
```

To consider vdd! and vss! (independent of case) as the power and ground nets instead of the defaults, use

```
% prospin -P {^vdd\!$} -G {^vss\!$} -l INV INV.sp
```

To use the ProSpin technology file **tech/prospin.tcl**, use

```
% prospin -t tech/prospin.tcl -l INV INV.sp
```

3.2.4. User-defined pattern-mappings

The ProSpin **-p** option does much of the work required to create a user-defined pattern-mapping. A user-defined pattern-map is required when a circuit structure does not match any topology in ProSpin's default search list or in any of the ProSpin libraries. It can also be used to customize the layout of a particular grouping of subgenerators.

The **-p** option causes ProSpin to output a Tcl source file containing two Tcl procedures. The first procedure returns a ProSpin pattern-map entry suitable for use in the ProSpin search_list option. The second procedure is a template of generator code. The user must complete the generator to get the desired layout.

The pattern-map entry procedure is named **cellname_pattern_map**. It names the pattern-mapping cellname and maps the Spice circuit structure to a generator called “cellname” that uses the standard Prolific generator interface. The input Spice circuit specification is the same as during normal ProSpin operation. Either the top-level net-list is used or a subcircuit is selected using the **-l** or **-s** subcircuit options. Similarly, cellname is the name specified for the relevant Spice structure using either the default behavior or the **-N** or **-n** options. The output of the **-p** option goes to stdout by default. It can be directed to a file using the **-O** option.

The **-p** option takes a required value which is either **asm** or **multi**. If the value is **asm**, a ProAssemble style generator is produced. If the value is **multi**, then a gen-multi style generator is produced. As generated by ProSpin, either style of generator can be run through progen, but will require varying degrees of customization to produce the desired results. The ProAssemble style generator produces the transistors in an arbitrary order without any explicit routing. To complete this generator, the user must update the gates list with the desired transistor order and then change node names and add input trunks, feed trunks, and stitch orders to create the desired routing. This generator uses a special progen helper routine that requires that the various ProAssemble information be stored in variables named the same as the ProAssemble arguments. (See the ProAssemble reference guide for details.) The gen-multi style generator template is more complete since it contains routing as specified by the subgenerators. The most common modifications to this generator are changes to the placeorder and mirrorgates arguments and specification of custom routing topologies. If the generated gen-multi style generator requires any user-defined pattern-maps, then these pattern-map entries must be in the search list of a cell called pattern_map.

Need some direction for specialty use? Check out Expeditions.

4. Expeditions

Expeditions are examples of specialty use of Prolific tools. There are currently no Expeditions in this section of the ProGenesis Guide.

5. ProSpin Reference Guide

5.1. Manual page

This manual page describes the available command-line options. Most any option that can be specified on the command line can also be specified in the setup file. The exceptions are -g, -h, -p, -q, and -v.

Name prospin -- Accepts Spice netlist input for ProGenesis processing

Synopsis

prospin [-C] [-c] [-d subckt_dir] [-F values] [-f] [-G ground_regexp] [-g] [-h] [-l subckt_list] [-N cell_name] [-n subckt_cell_name] [-O file] [-o subckt_output_file] [-P power_regexp] [-p gentype] [-q] [-s subckt_file] [-t tech.tcl] [-v] [-w] input

Description

ProSpin reads a Spice deck from the given file or from **stdin** and generates either progen input or Tcl code usable for user-defined pattern-mapping.

In general, a Spice deck may specify a net-list in terms of zero or more subcircuits. Some Spice decks only contain subcircuit definitions and do not specify a top-level net-list. In the following argument descriptions, please note which apply to the top-level net-list if it exists and which apply to any subcircuits that might exist.

Arguments

input
Name of input Spice file; if —, then stdin is read

Options

-C
Convert all Spice node names to upper case

-c
Convert all Spice node names to lower case

-F values
Comma-separated list of ProGen parameter values for fold_par_in_place and fold_ser_in_place in the form: fold_par_in_place,fold_ser_in_place

-f
Use default folding for all generators and subgenerators, ignoring the folding specified in the net-list

-G ground_regexp
Regular expression that matches ground net names. Default is ^(0|vss|gnd)$

-g
Run graphical, interactive interface

-h
Display usage message

-P power_regexp
Regular expression that matches power net names. Default is ^vdd$

-p gentype
Generate Tcl code for use in user-defined pattern-mapping. Two procedures are output. One returns a pattern-map entry and the other is a generator template. Type asm requests a standard ProAssemble generator. Type multi requests a gen_multi style generator.

asm
Requests a standard ProAssemble generator.

multi
Requests a gen_multi-style generator.

-q
When all licenses are in use, queues the job and waits for the next available license

-t tech.tcl
Full path to ProSpin technology file to use

-v
Output tool version number to stdout

-w
Reduce number of warnings output

Subcircuit Options

The input file has one derivative cell request per line. Each line has the form --

```
-d subckt_dir
```

Directory in which to place subcircuit output files. Equivalent to -o subckt_dir/%s.cel

```
-l subckt_list
```

Comma-separated list of subcircuits for which to generate output files. Takes precedence over the -s option.

```
-s subckt_file
```

Path to file of list of subcircuits (one per line) for which to generate output files in addition to the top-level netlist

```
-n subckt_cell_name
```

Tcl format string for cell name declared to progen for each subcircuit. Default is %s

```
-o subckt_output_file
```

Tcl format string for subcircuit output file names. Default is %s.cel
Be sure the -O and -o options do not conflict with each other.

Top-Level Net-list Options

The input file has one derivative cell request per line. Each line has the form:

-N cell_name
Cell name declared to ProGen for the top-level netlist. The default is the file name portion of input minus the last . suffix, or cell if input is stdin.

-O file
Path to output file. The default is stdout. This option is also valid when using the -p option to generate patterns.

Files

.prolificrc

Specifies simple setup information for all Prolific tools.   See prolificrc(5) for more details.

Environment

Prolific

Provides the location of the Prolific tools

Author

Prolific, Inc. <applications@prolificinc.com>

See also progen(1), Prolific(5), prolificrc(5) for more details.

5.2 ProSpin Setup File (prospin.tcl) settings

This reference section describes the options available within the **prospin.tcl** configuration file.

Spice-related Parameters

The Spice parameters allow you to supply Spice options that are missing from the input deck fed to ProSpin. This option allows you to use an incomplete Spice deck and fill in any necessary information. The parameters are described in the following list:

global_nodes
Specifies a list of node names to be handled as global nodes.

hierarchy_separator
Specifies the separator character for use with hierarchicial names.

nmos_models
Specifies the models that are nmos.

parhier
Specifies global or local parameter scoping.

pmos_models

Specifies pmos models.

scale

Specifies the Spice scale parameter.

global_nodes

There is no default **global_nodes** setting. To set the node names you wish to be considered global nodes, use the following syntax:

```
prospin_set spice global_nodes [list n1 n2]
```

This is equivalent to the following Spice option:

```
.global n1 n2
```

hierarchy_separator

ProSpin uses the underscore character as the default separator for building hierarchical names. The syntax for setting the separator is:

```
prospin_set spice hierarchy_separator _
```

Note that the ProSpin default is different from the period character used as the standard Spice separator.

nmos_models

This parameter allows you to specify which models are nmos. If the nmos_models and pmos_models options are not set and if no model card exists for the model name in the input Spice deck, ProSpin makes a guess based on the first or the last letter of the model name. The Spice parameter

```
prospin_set spice nmos_models {small mos1}
```

is equivalent to this Spice deck entry:

```
.model small nmos

.model mos1 nmos
```

parhier

ProSpin sets the default parameter scoping to global, which would be specified as:

```
prospin_set spice parhier global
```

This is equivalent to the Spice statement:

```
.options parhier=global
```

pmos_models

This parameter allows you to specify which models are pmos. If the nmos_models and pmos_models options are not set and if no model card exists for the model name in the input Spice deck, ProSpin makes a guess based on the first or the last letter of the model name. The Spice parameter

```
prospin_set spice pmos_models big
```

is equivalent to the Spice entry:

```
.model big pmos
```

scale

ProSpin defaults to a scaling value of one, which would be set using this statement:

```
prospin_set spice scale 1
```

This is equivalent to the Spice setting:

```
.options scale=1
```

Node Reformatting Options

The node_format parameters allow node names to be reformatted in the output using a format string for each of the following node types:
input

- output
- internal
- power
- ground
- The format string uses the syntax of the Tcl format command. If no format string is specified for a particular node type, the node name is transcribed literally after any command-line-specified conversions.

For example, to add the token prefix to input signals (at input ports), use

```
prospin_set node_format input prefix%s
```

Note that the dollar sign character (**$**) has special meaning in Tcl and cannot be successfully passed through ProGen. If you wish to have dollar signs in your layout node names, you can use another character sequence in the node_format string and then postprocess the ProGen output to change that sequence to **$**.

Node Type Classification Options

Users who specify the node name format will probably also want to use **node_match** to classify node names into node types. Each **node_match** entry for a node type is a list of pairs. The first element of the pair is a regular expression that matches some set of cell names. The second element is a regular expression that matches some set of node names.

To check if a node **node** of a cell **cell_name**, is a particular type **type**, the list of pairs of type is searched in order until a pair is found in which the cell name regular expression of the pair matches **cell_name**. If no matching pair is found, then node is not of type **type**. If a matching pair is found, then node is of type **type** if the node regular expression of the pair matches node. The cell name and node name matching are case insensitive. The defaults for the **node_match** options are:

```
        prospin_set node_match power {
            {{.} {^(1|vdd)$}}
        }
        prospin_set node_match ground {
            {{.} {^(0|vss|gnd)$}}
        }
        prospin_set node_match input {
            {{.} {^([a-kmr-w][^ou]*|x[0-9]*|pre|no.*)$}}
        }
        prospin_set node_match output {
            {{.} {^([lopqsyz][a-z0-9]*|[a-z]o[a-z0-9]*)$}}

    }
```

Any nodes that do not match this default description are classified as type internal.

ProSpin Execution Options

General ProSpin execution parameters are specified using option. The general form of option values in this category is a list of pairs. The first element in each pair is a regular expression that is matched against the cell name. The second element in each pair is the option value to set if the first element matched the current cell. Options that always use a single value can be specified as the single value rather than as a list of pairs. These single-value options are applied to all cells.

case

force_gen_multi_usage
length_grid
length_max_nfet
length_max_pfet
length_min_nfet
length_min_pfet
length_ratio_nfet
length_ratio_pfet
width_grid
width_max_nfet
width_max_pfet
width_min_nfet
width_min_pfet
width_ratio_nfet
width_ratio_pfet
output_file
pattern_map_type
post_normalization_proc
post_normalization_args
reduce_warnings
search_list
source_path
subcircuit_cell_name
subcircuit_output_file
use_default_folding
use_spice_stack_folding
verbose
case

This option specifies the case used for Spice node names in the output. The possible values are shown below.

Option	Description	Equivalence
tolower	Use all lowercase	Same as using prospin -c
ProSpin	Use all uppercase	Same as using prospin -C

Therefore, to set the output node names to use all lower-case letters, use this syntax:

```
prospin_set options case tolower
```

subcircuit_cell_name

This option is used to specify a format string used to generate the cell name that will be declared to ProGen for any subcircuits. The syntax of the format string is the same as the Tcl format command. Consider this example:

```
prospin_set option subcircuit_cell_name prefix%s
```

This statement will declare to ProGen that the cell name of the subcircuit subckt is prefix_subckt.

subcircuit_output_file

This option specifies a format string used to generate the subcircuit output file names. The syntax of the format string is the same as the Tcl format command. Consider this example:

```
prospin_set option subcircuit_output_file cel/%s.cel
```

This statement will cause the subcircuit output files to be written to the cel subdirectory with the suffix .cel. The cel subdirectory is created if it doesn't exist.

pre_patterns
This option specifies patterns to match before attempting to match an AOI/OAI topology, including the degenerate NAND, Nor, and inverter cases. The option values for this parameter must be specified in the general format as a list of pairs. The second element of each pair is a list of pattern entries. Entries may be selected from the ProNet pattern library or may be user-defined using the Tcl procedure.

prospin_pattern_entry
Described in the User-Defined Conversions section of this document. A sample usage of the pre_patterns option is:

```
prospin_set option pre_patterns [list \

      [list {^latch.*$} [list [pronet_latch_iii]]] \

      [list {^flop.*$}  [list [pronet_latch_xni] \

                              [pronet_latch_xin]]] \

      [list (^mux.*$}   [list [pronet_mux21_ii] \

                              [pronet_mux21_ix] \

                              [pronet_mux21_xx]]] \

]
```

In Example 1, cell names or subcircuit names starting with the string flop are matched against pronet_latch_xni and pronet_latch_xin before being matched against AOI/OAI topologies.

post_patterns
This option specifies patterns to match after attempting to match an AOI/OAI topology, including the degenerate NAND, Nor, and inverter cases. The option values for this parameter must be specified in the general format as a list of pairs. The second element of each pair is a list of pattern entries. Entries may be selected from the ProNet pattern library or may be user-defined using the Tcl procedure **prospin_pattern_entry**.

verbose
This option turns verbose mode on or off.

Main	Description
0	Turn verbose mode off (default)
1	Turn verbose mode on

The statement syntax is:

```
prospin_set option verbose 0
```

Verbose mode provides maximum feedback while ProSpin is running.

width_*
There are seven width_ options, each used to change device widths on input:

width_grid

width_max_nfet

width_max_pfet

width_min_nfet

width_min_pfet

width_ratio_nfet

width_ratio_pfet
These options change the device widths by a specified ratio for each device type. The width of each type of device is multiplied by the ratio the user specifies. If width_grid is specified, the width will be snapped to the closest multiple of width_grid that is greater than or equal to the device's modified width value. Minimum and maximum device widths may also be set per device type.

To alter the width of all nfet type devices by 0.80, use this syntax:

```
prospin_set width_ratio_nfet 0.80
```

length_*
There are seven length_ options, each used to change device lengths on input:

length_grid

length_max_nfet

length_max_pfet

length_min_nfet

length_min_pfet

length_ratio_nfet

length_ratio_pfet
These options change the device lengths by a specified ratio for each device type. The length of each type of device is multiplied by the ratio the user specifies. If length_grid is specified, the length will be snapped to the closest multiple of length_grid that is greater than or equal to the device's modified length value. Minimum and maximum device lengths may also be set per device type.

To alter the length of all pfet type devices by 0.65, use this syntax:

```
prospin_set length_ratio_pfet 0.85
```

ProGen Input Options

The progen parameters allow you to make the ProGen input generated by ProSpin consistent with your ProGen technology file. The subset of ProGen parameters required to ensure this compatibility is supported. The progen options are listed below:

fold_par_in_place

fold_ser_in_place
Number of routing tracks used before number of diffusion breaks outweighs routing density

scale

fold_par_in_place
This parameter specifies how parallel stacks should be folded. This should be set to match the configuration used in the target ProGen technology file. The possible values are:

Value	Description
0	The whole stack of transistors will be folded
1	Individual transistors will be folded inside the stack of transistors (default)

To override the default and fold the whole stack of transistors, use this setting:

```
prospin_set progen fold_par_in_place 0
```

fold_ser_in_place
This parameter specifies how serial stacks should be folded. This value should also be set to match the configuration in the target ProGen technology file. The possible values shown in Table 4 apply to serial stacks.

ProSpin assumes fold_ser_in_place should be set to 1 unless it determines from the net-list that it must be set to 0.

Value	Description
0	The whole stack of transistors will be folded
1	Individual transistors will be folded inside the stack of transistors (default)

scale
This parameter declares the number of ProGen internal database units per nanometer. The default is:

```
prospin_set progen scale 1
```

This corresponds to the ProGen technology file specification:

```
pro_set options scale 1
```

Compaction Options

This option allows further customization of the generated ProGen input. Values are specified relative to a regular expression matched against the cell name, so options may be applied to a single cell or a set of cells. Each cell gets all options for which its cell name matches the associated regular expression. These options can be placed in any order.

This option allows customization of the compaction order, to alter the default compaction specification. To change the strategy to compact first the X-axis, then the Y-axis, and then the X-axis again for a set of AOI gates, use this syntax:

```
prospin_set compaction_options {^aoi[0-9]*$} {x y x}
```

Generator-Related Options

The following three options are used to control which generators are called and what options they use:

```
generator_options

generator_options

generator_setup
```

Generator Name Specification Options

This option allows further customization of the generated ProGen input. Values are specified relative to a regular expression matched against the cell name, so options may be applied to a single cell or a set of cells. Each cell gets all options for which its cell name matches the associated regular expression. These options can be placed in any order.

This option indicates the name of a Tcl generator routine to call instead of the default routine. To use a special generator routine, gen_foo, for the cell baz, include the following statement:

```
prospin_set generator_name {^baz$} gen_foo
```

Setting Generator Options

This option allows further customization of the generated ProGen input. Values are specified relative to a regular expression matched against the cell name, so options may be applied to a single cell or a set of cells. Each cell gets all options for which its cell name matches the associated regular expression. These options can be placed in any order.

The value for this option is a list of lists. Each of the sublists has exactly two elements. The first element is the name of an argument to the generator function. The second element is the value for that argument.

To change the track assignment for one particular version of an XOR cell, xor21, use this syntax:

```
        prospin_set generator_options {^xor21$} {
                {ytrackbynet ?{{@i1 1} {@i2 2} {@@n2 0}}?}

    }
```

An exception to this rule occurs when the argument name is **gatecalls**. In the output, the value of **gatecalls** is a list of **flag_call** statements. To allow user modification of these **sub-flag_call** commands, a special format is used for the second element. This element is a list of pairs, where the first element is the index (starting at zero) of one of these **sub-flag_call** statements. The second element is the list of lists to specify generator arguments for that particular **flag_call**. These specifications can replace or add to the **flag_call** arguments passed to the cell generator.

Overriding CEL File Settings

This option allows further customization of the generated ProGen input. Values are specified relative to a regular expression matched against the cell name, so options may be applied to a single cell or a set of cells. Each cell gets all options for which its cell name matches the associated regular expression. These options can be placed in any order.

These values will be included in the section immediately before the call to the generator. They are used to override settings from the CEL file that cant be done directly with proset.

For example, to make the metal-1 priority the same as the poly priority for some set of NAND gates, use this syntax:

```
prospin_set generator_setup {nand[0-9*]} {

        tech_set_priority m1 [tech_get_priority poly]

}
```

5.3 Graphical Interface Reference

Navigating ProSpin

Menu Bar

The ProSpin menu bar provides access to the various functions available via ProSpin's graphical interface. The menu bar contains the following entries:

- File Menu
- Output Menu
- Cell Menu
- View Menu
- Options Menu
- Help Menu

File Menu

The File menu is used to load circuits or exit the application.

Load Spice

This menu option opens a dialog box that allows loading of one or more input Spice files. To select multiple files, hold down the Ctrl button while clicking the desired filenames. To select a sequence of files, click the first filename, then press the Shift button while clicking the last filename in the sequence.

Exit

Closes ProSpin.

Output Menu

The Output menu specifies the type of output expected from ProSpin.

ProGen input

When ProSpin is run, instruct it to generate a CEL cell definition file from each of the processed circuits. generator for each input cell.

Hierarchical generator and pattern-mapping

Instruct ProSpin to generate a pattern-mapping procedure and hierarchical generator (gen_multi-style)

Flat generator and pattern-mapping

Create a pattern-mapping procedure and a ProAssemble-style generator for each input cell.

Cell Menu

The Cell menu contains options that process circuits/cells selected in ProSpin's circuit list.

Clear Status

Select one or more circuits, then use the Clear Status option to unset any status information related to them. The circuits can be selected from any notebook tab. As ProSpin processes the circuits, it will scroll through the list and update the status information for each cell.

Remove

Removes the selected circuits from all views of the circuit list.

Run

Creates the specified output for the selected circuits. The output is specified from the Output menu.

Run All

Creates the specified output for all listed circuits (not just the selected circuits). The output is specified from the Output menu.

Select

Opens a dialog box in which cells can be selected using Tcl regular expressions. A pulldown menu maintains a history of regular expressions.

The Include circuits that match pattern radio button indicates that only circuits that match the specified regular expression will be selected.

The Exclude circuits that match pattern radio button indicates that only circuits that do not match the specified regular expression will be selected.

Clicking the Apply Pattern button determines what circuits should be included based on the regular expression and the selection criteria. These circuits are added to any currently selected in the notebook view; the current selection is not reset first.

Click the Deselect All button to deselect all circuits in the notebook view. This button can be used to clear the existing selection before applying a new pattern.

Click the Close button to close the selection dialog box. This button will not cause any other actions to be taken; to select circuits based on the pattern, use the Apply Pattern button instead.

Stop

Use the Stop option to cancel the processing of the current set of cells. This is useful when things are not progressing as desired (a Run or Run All command is placing output files in the wrong location, the wrong setup file is being used and all the cells are failing, or the wrong type of output is being produced).

View Menu

The View menu is used to display output or error information for any selected cells. One or more cells must be selected for the View options to display information.

ProGen input

This option brings up a window showing the CEL file ProSpin produced for each of the selected circuits.

Generator and pattern-mapping

This entry displays the generator and pattern-mapping information output by ProSpin for each selected circuit. If ProSpin has not been instructed to produce a generator and pattern-mapping, an error message will appear with a warning that a .tcl file is not readable. To produce this file, set the output option to create a generator and pattern-mapping for ProGen.

Errors

Select this menu item to view any errors produced by ProSpin for each of the selected circuits. No results will be displayed for any circuits that did not produce errors.

Results

Selecting this entry opens windows containing whatever results were produced by ProSpin for each of the selected circuits.

Icons

The Icons entry is used to control the display of the ProSpin tool button bar. This menu option is a tear-off list, so it may be opened as a separate window by selecting the dashed line at the top of the submenu. The four suboptions include:

Icon Bar

This option, which is enabled by default, displays the buttons in the tool bar. Deselect this option to remove the tool bar from the ProSpin interface. If this option is not selected, the other options on the Icons submenu will not be relevant.

Icon Labels

This option, which is enabled by default, displays text labels on the buttons in the tool bar.

Small Icons

This option, which is enabled by default, displays small button icons in the tool bar. Select Large Icons instead to display bigger button icons.

Large Icons

This option displays large button icons in the tool bar. Select Small Icons instead to display smaller button icons.

Settings

This option, which is enabled by default, displays the current setup file location and output directory. Deselect the option to remove the settings area from the ProSpin interface, which will free some screen real estate within ProSpin.

Options Menu

This menu allows configuration of the settings area parameters.

Options

Select the Option entry from the Options menu to pop up a window in which the setup file and output directory may be configured. Click the Close button when the desired settings are entered.

Setup file

Enter the path to the ProSpin setup file (usually prospin.tcl), or click on the folder icon to browse the directory structure to find and select the file. If ProSpin is run with the -t option set, the filename specified on the command line will be the initial setup file.

Output directory

Enter a directory name or click on the button to browse the directory structure. If the entered directory does not exist, it will be created. The output directory defaults to the directory from which ProSpin was started.

Help Menu

Use the Help menu to find help related to the interface and function of ProSpin.

Contents

This is a top-level list of the help available in ProSpin's graphical interface.

Menus

Describes the ProSpin menus.

Buttons

Describes the function of the buttons on the ProSpin tool bar.

Tabs

Explains the notebook tab region of the ProSpin graphical interface.

About ProSpin

Provides application and contact information.

Tool Bar

The ProSpin tool bar features a set of buttons that can be used to access frequently used features.

Run

The Run tool duplicates the operation of the Cell menu's Run option: it creates the specified output for the selected circuits. The output is specified from the Output menu.

Run All

Clicking Run All causes ProSpin to process all listed circuits to produce the output specified from the Output menu. This duplicates the operation of the Cell menu's Run All option.

Stop

Use the Stop button to cancel the processing of the current set of cells. This is useful when things are not progressing as desired (a Run or Run All command is placing CEL files in the wrong location, the wrong setup file is being used and all the cells are failing, or the wrong type of output is being produced).

Select

Use the Select button to select cells using Tcl regular expressions. A pulldown menu maintains a history of regular expressions. This duplicates the operation of the Cell menu's Select option.

Remove

Use the Remove button to remove any selected cells from the list in the tabbed notebook region.

View

Use the View button to view output from a ProSpin job. The View button will attempt to display the type of output currently specified by the Output menu setting; if there is no such output, an error message is displayed.

Settings Area

The Settings area displays the current setup file (also referred to as the technology file) and output directory.

Setup File

The setup file is typically called prospin.tcl. Usually located in the technology area directory, this file contains configuration information for ProSpin. Its contents can be configured from the Options menu. See the prospin.tcl section (5.2) for more information.

Output Directory

The output directory is where ProSpin will store the files it generates. By default, the output directory is the directory from which the ProSpin tool was launched. If ProSpin was launched using the -d or -o options, another output directory will be designated.

Configuring the Settings Area

The Settings area is included by default when ProSpin runs, but it may be turned off via the View menu's Settings option.

Tabbed Notebook Region

The notebook region of the ProSpin graphical interface is used to list the input Spice net-list circuits and their corresponding status information. The information displayed depends on which tab is selected.

Circuits Tab

The Circuits tab lists all circuits loaded from an input net-list. These circuits are displayed in rows, showing the following information for each circuit:

Circuit name
Circuit status
Circuit source file

File

The File tab shows the same information as the Circuits tab, but arranges it in a slightly different format. Instead of a row-based format of all loaded circuits, the File tab displays a directory tree showing the files containing any loaded circuits. The directories and files in the structure can be expanded and collapsed by clicking the + and - symbols next to their icons.

Passed

The Passed tab filters the circuit list to show only circuits that were successfully processed by ProSpin. This tab displays the circuit name and net-list file name for each of these circuits.

Failed

The Failed tab filters the circuit list to show only circuits that ProSpin failed to successfully process. This tab displays the circuit name and net-list file name for each of these circuits.

5.4 Manual Conversion Definitions

The recommended way to specify a conversion is to use ProSticks. However, it is possible for users to define any number of conversions manually. Conversions are registered with ProSpin by setting either the pre_patterns or post_patterns options. The pattern entries specified for these options can be generated using prospin_pattern_entry.

prospin_pattern_entry Interface

The prospin_pattern_entry procedure controls the generation of pattern entries for user-defined conversions. The values used by the procedure are:

Parameter	Description
`namedesc`	Conversion name or list of conversion name and pattern description
`definition`	String containing circuit topology definition
`generators`	List of generators to call for this pattern
`genArgFormats`	List of list of argument name-format pairs
`diffusions`	List of un-mirrored left, right diffusion net pairs by row
`mirroring`	List of default mirroring for each generator call
`internals`	List of relative port/pin internal positions from left to right in the un-mirrored pattern

The procedure is:

```
      proc prospin_pattern_entry {
            namedesc
            definition
            generators
            genArgFormats
            diffusions
            {mirroring 0}
            {internals { }}

   }
```

Circuit Topology Definition

Circuit topologies are defined in a string containing lines of connect and property statements.

connect

The connect statements feature the following syntax:

connect {{inst deviceType} {{termType netName} ...}

The values of the parameters in the connect statement are:

Parameter	Description
`inst`	The instance name of a device
`deviceType`	The type of inst; usually nfet or pfet
`termType`	A terminal type; srcdrn or gate
`netName`	The net node connected to the terminal of termType

property

The property statements use this syntax:

property net netName type netType

The values for the parameters in the property statement are shown in Table 7. Note that net and type are not variables; the literal strings net and type must appear in the property statements.

Parameter	Description
`netName`	A net name found in a connect statement
`netType`	A net type: power, ground, input, output, or internal

All nets in the pattern must have their types specified.

The string shown in Example 2 can be used to describe an inverter:

```
set def {
      connect {{I1 pfet} {{srcdrnVDD} {srcdrn out} {gate
in}}}
      connect {{I0 nfet} {{srcdrn VSS} {srcdrn out} {gate
in}}}
      property net VDD type power
      property net VSS type power
      property net in type input
      property net out type output

}
```

Generator Argument Format Strings

flag_call

The actual cell generator routine is called using a helper function called flag_call. This function allows users to specify the arguments to the generator routine in any order. In general, if a generator routine, generator, has an argument, argN, then that argument can be specified as argValue by calling:

```
flag_call generator ?argN argValue
```

Arguments that have default values need not be specified to flag_call.

genArgFormats

The genArgFormats argument is a list of list of lists. Each sub-list corresponds to a generator call. Each sub-sub-list has exactly two elements. The first element is the name of an argument to the specified generator and the second element is a format string specifying the value for that argument. Special formatting commands are provided to substitute net names, device widths, and other attributes from the input net-list. These commands are:

%f device
This command returns the number of legs of the device, device.

%n net
This command returns the Spice node name that corresponds to net. One of the sub-sub-lists for the inverter described in Example 2 in the Section called "Example" might be:

{namess {list [list [%n in]]}}

%r fmt args
This command unfolds folded devices for matching. The %r command provides a way to format generator arguments based on the original folded structure. The fmt parameter is a Tcl format string, and args is a variable number of pattern net names.

The %r command returns a list where each element corresponds to a fold or leg in the original net-list and is formatted according to fmt with values mapped from the pattern net names in args back to the original folded net names.

%u token

This command returns the string formed by appending an underscore character and a unique number to token. For example, the syntax below shows a sub-sub-list that could return prefix_0.

{namess {list [list [%u prefix]]}}

%W device

This command returns the width of the device, device, in ProGen internal units (by default, these units are set as nanometers).

%w device

This command returns the width of the device, device, in microns.

Diffusion Node Placement

The placement algorithm requires the net names of the leftmost and rightmost n and p diffusion nodes corresponding to each pattern. The generator must place these nodes in an un-mirrored instance of the cell. The net name information is specified as a list of pairs.

Each pair corresponds to a ProGen row, starting with row 0 and going in increasing order. The first element of each pair is the name of the leftmost diffusion node for that row. The second element of each pair is the name of the rightmost diffusion node for that row.

Mirroring

The placement algorithm requires the default mirroring of each of the constituent generator calls for the pattern. This value is a list with an element for each generator call. The element is 0 if the call does not mirror that component and 1 if it does mirror the component.

Internal Port and Pin Positions

The relative internal positions of ports and pins may be specified to the placement algorithm in order to improve its mirroring decisions. This parameter is an ordered list of lists. The sub-lists correspond to topological horizontal positions and are ordered from left to right. The elements of the sub-lists are the names of nets that have ports or pins at the corresponding position.

```
#
# Specify inverter net-list
#
set def {
connect {{P0 pfet} {{srcdrn VDD} {srcdrn out} {gate in}}}
      connect {{N0 nfet} {{srcdrn VSS} {srcdrn out} {gate in}}}
      property net VDD type power
      property net VSS type power
      property net in  type input
      property net out type output
}
#
```

```
# Generator list
#
set generators pasm_complex_size
#
# Specify flag_call arguments for the inverter generator
#
set argFormats {
    {isnor             {list inv}}
    {nums              {list 1}}
    {nwidths           {list [%w N0]}}
    {pwidths           {list [%w P0]}}
    {nfolds            {list [%f N0]}}
    {pfolds            {list [%f P0]}}
    {namess            {list [list [%n in]]}}
    {outname           {%n out}}
    {nnodenames        {list [list [%n VSS] [%n out]]}}
    {pnodenames        {list [%n VDD] [%n out]}}
    {fold_ser_in_place {list 1}}
    {fold_par_in_place {list 1}}
}
#
# Specify the leftmost diffusion node and the rightmost
# diffusion node by row for an un-mirrored instance of
# this cell.
#
set diffusions {list [list [%n VSS] [%n out]] [list [%n VDD] [%n out]]}
#
# Specify mirroring - unmirrored
#
set mirror 0
#
# Specify internal port/pin positions
#
set internal {list [%n in] [%n out]}
#
# Generate a pattern entry
#
set entry [prospin_pattern_entry {inv {user inverter pattern}} \
                           $def $generators [list $argFormats] \
                           $diffusions $mirror $internal]
```

ProSpin FAQ

What is the difference between series folding and parallel folding?

Let's use nand2 -- a simple two-input NAND gate -- as example to describe series and parallel gate folding. In a nand2 layout, the n-fets are series gates (no source/drain contact between two poly-gates), while the p-fets are parallel gates (contain s/d contacts between poly-gates).

If you apply the following control to nand2 with inputs "a" and "b", and the order of the inputs are "a b" before folding, from left to right:

```
pro_set topology fold_ser_in_place 0

    pro_set topology fold_par_in_place 1
```

This will fold series-stacks (n-gates) as stacks, so the input gate order after folding on the n side becomes "a b b a". However, on the p side the parallel-stacks are folded in-place individually, which makes the input gate order be "a a b b".

"set fold_xxx_in_place 0" causes folding transistor stacks (either series-stacks or parallel-stacks) as stacks, so the input order becomes "a b c c b a a b c ..." given a three-input cell, while "set fold_xxx_in_place 1" results in folding transistor stacks in place, such that the input becomes "a a ... b b ... c c ..."

How do I deal with unroutable cells? What about placement order information?

If the cell template allows more routing tracks, increase routing track numbers:

```
pro_set global proppr_numtracks_<layer>_chan<#>  <num_of_tracks>
```

For example, to increase the number of metal1 routing tracks in upper channel to 2, and increase the number of poly routing tracks in lower channel to 2;

```
pro_set global proppr_numtracks_m1_chan2    2

    pro_set global proppr_numtracks_poly_chan0  2
```

If increasing the number of routing tracks is not an applicable solution, change the placement order (place-order) of the cell to help ProPPR router find routing solution.

There are several ways to change the placeorder:

(1) Check for any cell in the same functional family that can be routed. If there is, copy the placeorder from its log file and apply the order to the current cell with the following control in prospin.tcl:

```
        prospin_set generator options {^cellname$} {
        {placeorder {<placeorder>}}
    }
```

Re-run prospin and progen.

(2) Set "placement_cost_model" to "congestion" to generate placements that are easier to route, but may be a larger area, as follows:

```
   prospin_set generator options {^cellname$} {
   {placement_cost_model congestion}
   }
```

(3) Increase diffusion break tolerance to make placements more routable:

```
   prospin_set generator options {^cellname$} {
   {diffusion_break_tolerance <value>}
   }
```

To find out the existing diffusion break number, search for "Running placement algorithm" in progen log file.

The line after "Running placement algorithm" is a list of costs. The fourth entry in the list is the number of diffusion breaks of the current placeorder.

For example, if you find the following in a log file:

```
Running placement algorithm ...

   cost = 0 0 0 2 4 7232 4 4
```

The diffusion break number is "2". To increase the tolerance to 3:

```
   prospin_set generator options {^cellname$} {
   {diffusion_break_tolerance 3}
    }
```

(4) Manually specify custom placeorder to help find routing solution:

```
   prospin_set generator options {^cellname$} {
   {placeorder {<custom_placeorder>}}

}
```

How do I maximize poly contacts for big drive inv and buffers? What about EM considerations?

One easy way is to change the Poly contact merging value in ProTech under the Options tab. You can also assign a special layer, "m1poly", to a particular route segment to have both metal1 and poly routes strapped with poly contacts.

How is M=x interpreted by the tool?

The transistor size is multiplied by "x".

Is there any lint checker for netlist?

No.

What are the restrictions on netlist generation?

Prospin does not support the ".include" statement. Prospin does not make use of R and C statements.

ProPPR Guide

Cell Creation Using ProPPR

Once the `ProTech` and `.cel` file setup is complete, ProPPR is ready to create cells.

Testing Routing Configurations

The additional setup in the technology area is intended to give ProPPR the information for a particular cell template and technology to determine which routing configurations are most likely to fit in the cell height. This is done by running a couple of test cases with different routing configurations and passing the resulting transistor sizes to the system. The test cases are run and transistor sizes extracted from ProTech.

Tuning Cell Configuration

Additional setup options may be placed in the set_options procedure.

Enabling Verbose Mode

To enable ProPPR's verbose mode, use the following command:

```
pro_set proppr_debug 1
```

Routing Layers

You can instruct ProPPR to use only poly and metal-1 as the routing layers, or to use metal-1, poly, and metal-2. The proppr_numlayers parameter controls the routing layers, as shown:

Parameter	Description	Default
`proppr_numlayers`	Number of routing layers: if set to 2, then poly and metal-1 are routed; if set to 3, then poly, metal-1, and metal-2 are routed	2

Table 1: Set number of routing layers

It's important to be aware of which routing layers you are using since some of the other router controls are layer-dependent. Specifically, there are cost-per-layer and tracks-per-layer controls. Allowing ProPPR to use metal-2 does not mean metal-2 will necessarily be used. It depends on the cost-per-layer settings.

The following example sets ProPPR to use poly, metal-1, and metal-2 routing layers:

```
pro_set global proppr_numlayers 3
```

Setting the value to anything other than 2 or 3 will result in the following error in the log file:

```
proppr_numlayers must be either 2 or 3
```

Routing Channels

The ProPPR router divides the cell area into channels. Figure 4 illustrates the basic standard cell layout which is made up of 3 channels. The bottom channel (number 0) is below the n-transistors, the center channel (number 1) is between the n- and p-transistors, and the top channel (number 2) is above the p-transistors.

Several of the controls are channel-specific. ProPPR uses the channel number to indicate to which channel the control is being applied.

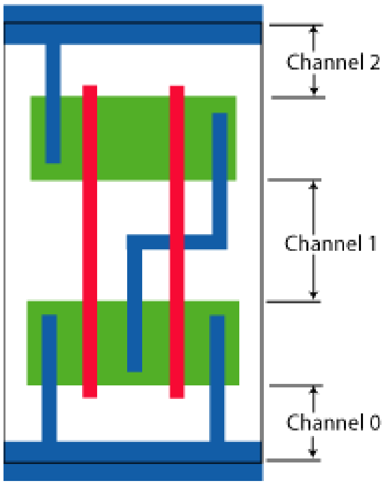


Figure 4. Routing Channels

Controlling Routing Tracks

Within a channel ProPPR uses the concept of horizontal routing tracks to represent the number of parallel horizontal routes which can fit in the channel height. Each routing layer has its own number of routing tracks per channel. Routing tracks of different layers can overlap. Any number of non-overlap route wires on the same layer can exist on a single routing track. Figure 5 shows a cell with two routing tracks for metal-1 routes in the outer channels and three routing tracks for metal-1 routes in the center channel.

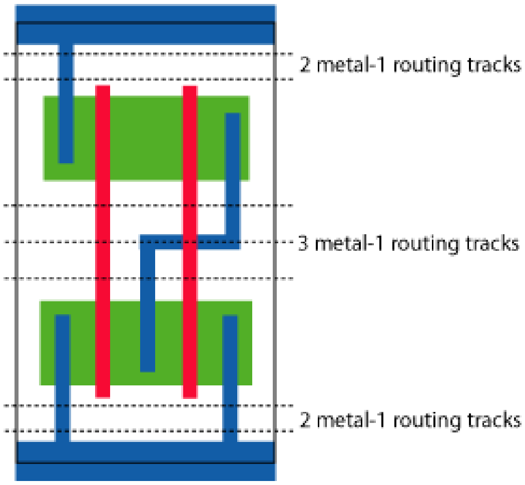


Figure 5. Routing Tracks

The number of tracks can be increased using the numtracks parameters:

Parameter	Description	Default
`proppr_numtracks_m1_chan0`	Number of m1 routing tracks in the lower channel	1
`proppr_numtracks_m1_chan1`	Number of m1 routing tracks in the center channel	2
`proppr_numtracks_m1_chan2`	Number of m1 routing tracks in the upper channel	1
`proppr_numtracks_m2_chan0`	Number of m2 routing tracks in the lower channel	1
`proppr_numtracks_m2_chan1`	Number of m2 routing tracks in the center channel	2
`proppr_numtracks_m2_chan2`	Number of m2 routing tracks in the upper channel	1
`proppr_numtracks_poly_chan0`	Number of poly routing tracks in the lower channel	1
`proppr_numtracks_poly_chan1`	Number of poly routing tracks in the center channel	2
`proppr_numtracks_poly_chan2`	Number of poly routing tracks in the upper channel	1

Table 2: Set number of tracks

Here is an example of the numtracks parameters in action:

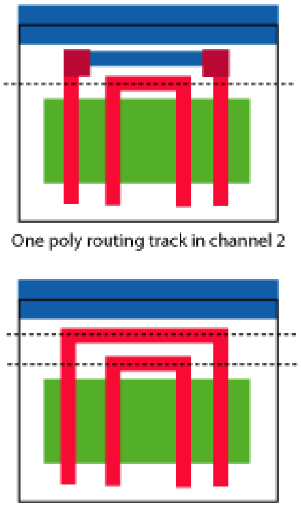
```
pro_set global proppr_numtracks_poly_chan1  3


pro_set global proppr_numtracks_m1_chan1 3

pro_set global proppr_numtracks_m1_chan2 2
```

Example 1: Specifying the number of tracks

Setting the numtracks parameters does not magically provide additional space in the cell template, so defining more routing tracks than the template can support may cause the cell to miss height.

The numtracks parameters supersede the deprecated proppr_numycoords parameter, which is set to the sum of the maximum number of tracks for each layer for each channel, plus 2 for y_p and y_n.

Figure 6 demonstrates how a routing topology can be changed if the number of poly tracks is increased from one to two. In the top layout there is only one poly routing track, so the outer gates are connected with a metal-1 route which can float over the inner gate poly routes. In the bottom layout there are two poly routing tracks, so ProPPR routed both the inner and outer gates in poly.



One poly routing track in channel 2

Two poly routing tracks in channel 2

Figure 6. Routing Track Example

Controlling Routing Cost

ProPPR determines the quality of routing solutions based on the total cost of the solution. The routing solution with the lowest cost is considered the best. The total cost of the solution is derived by summing up the cost of the individual routes within the solution. Every routed net has an individual cost associated with it. The following factors have cost associated with them that ProPPR uses when summing up the total cost:

layers of vertical routing segments (poly, metal-1, metal-2)

- layers of horizontal routing segments
- length of routing segments
- number of contacts in route
- number of vias in route
- ProPPR provides the following controls to override the system default values:

Parameter	Description	Default
`proppr_layerhor_costs`	List of horizontal routing segment costs, by layer: {<polycost> <m1cost> <m2cost>}	{40 10 90}
`proppr_layerver_costs`	List of vertical routing segment costs, by layer: {<polycost> <m1cost> <m2cost>}	{15 10 50}
`proppr_via_cost`	List of contact/via cost, by layer: {<contact via>}	{100 100}

Table 3. Options for Controlling Routing Cost

proppr_layerhor_costs

```
proppr_layerhor_costs {polyCost m1Cost m2Cost}
```

This option allows you to control the cost of all horizontal routing segments based on the layers of the segments.

```
pro_set global proppr_layerhor_costs {50 10 15}
```

When ProPPR needs to change from the metal-1 layer to another with a horizontal routing segment, the above example encourages it to go to metal-2 instead of to poly.

These values can instead be specified as a list of lists, where each entry of the list is the cost per channel. The following example will set the hor poly cost for channel 2 (above the p-transistors) to 100:

```
pro_set global proppr_layerhor_costs {{40 40 100} 10 90}
```

Furthermore, each track can be set individually as well. If there are two poly tracks in channel 2, the following will increase the cost of just the track which is farthest from the transistor:

```
pro_set global proppr_layerhor_costs {{40 40 {40 100} 10 90}
```

Note that -- like the max_outer descriptions -- tracks are numbered from closest to transistor to farthest from the transistors. Therefore, to change the cost of the lowest track in channel 0 (the channel below the N-transistors) use:

```
pro_set global proppr_layerhor_costs {{{40 100} 40 40} 10 90}
```

proppr_layerver_costs

```
proppr_layerver_costs {<cost_poly> <cost_m1> <cost_m2>}
```

This option allows you to control the cost for all vertical routing segments based on the layers of the segments. Usually metal-1 will have the lowest cost, poly is next, and metal-2 is the highest.

When ProPPR needs to change from the metal-1 layer to another with a vertical routing segment, the following example encourages it to go to metal-2 instead of to poly:

```
pro_set global proppr_layerver_costs {50 10 15}
```

Further, the cost per layer can be a list, including one entry per channel:

```
 {<cost_poly_chan0> <cost_poly_chan1> <cost_poly_chan2>}
```

The cost per layer, per channel can be further sub-divided by y-track, but that is not recommended. For example:

```
pro_set global proppr_layerver_costs {{75 15 75} 10 50}
```

Example 2

will set the vertical poly cost for the outer channels to 75, and leave the rest at their defaults (10 for m1, 50 for m2.

```
proppr_layerver_gate_costs {<cost_poly> <cost_m1> <cost_m2>}
```

Example 3

Results as above, except the vertical cost is only applied for slices (x-values) that align with gates.

Other slices will use the values in proppr_layerver_costs.

This defaults to proppr_layerver_costs.

```
proppr_layerver_sd_costs {<cost_poly> <cost_m1> <cost_m2>}
```

As above, except the vertical cost is only applied for slices (x-values) that align with source/drains (not gates).

Other slices will use the values in proppr_layerver_costs.

This defaults to proppr_layerver_costs. For example:

```
pro_set global proppr_layerver_sd_costs {infinity 10 50}
```

Example 4

will disallow vertical poly on the non-gate slices, and leave the rest at their defaults (10 for m1, 50 for m2). Vertical poly on the gate slices will be as defined by either proppr_layerver_gate_costs (if any) or proppr_layerver_costs.

Note that the slices on the left and right edges of the cell are not considered gate or source/drain slices for this purpose and will always use the values of proppr_layerver_costs.

proppr_via_cost

```
proppr_via_cost {contactCost viaCost}
```

This option allows you to control the cost of all contacts and vias. In general, the cost of contacts and vias is high compared to the horizontal and vertical routing segment cost. This prevents the system from hopping excessively between layers.

```
pro_set global proppr_via_cost {100 10}
```

This setting encourages ProPPR to hop from metal-1 to metal-2 more freely since the cost of vias is so low. This would probably be used with low cost settings for metal-2 route segments.

proppr_layercost_overrides

There is also a fully customizable interface to control the costs of each location. This is set as follows:

```
   pro_set global proppr_layercost_overrides {over1 over2 ...}
```

where over1, over2, etc. consist of 2-item sublists, for example:

```
   {cost_type ver} {cost infinity} {layer m1} {y all} {slice 1}
```

which will set the vertical layer cost of m1 to infinity at slice 1, for all y-values.

The allowed values are:

`cost_type`	one of: `ver hor bend via`
`cost`	either `infinity`, or an integer >= 0
`layer`	one of: `poly m1 m2`
`y`	either `all`, or list of integers and/or symbols:
`y_gnd`	ground rail
`y_vdd`	vdd rail
`y_gnd offset <offset>`	y of gnd + `<offset>` (integer)
`y_vdd offset <offset>`	y of vdd + `<offset>` (integer)
`y_row <n>`	transistor row <n> where <n> = 0 is the lowest row
`y_row <n> offset`	transistor row <n> + <offset>
`y_channel <n>`	all y-values in channel <n> where <n> = 0 is the channel below the lowest transistor row
`slice`	either `all`, or an integer >= 0
`slice_repeat`	apply to slice + <n> * slice_repeat, where n>=0
`slice_begin`	only apply to slices >= slice_start

`slice_start`	only apply to slices >= slice_start	alternate name for `slice_begin`
`slice_end`	only apply to slices <= slice_end	
`slice_finish`	only apply to slices <= slice_end	alternate name for `slice_end`

For slices:

0 is the left-most slice

1 is the left-most source/drain slice

2 is the left-most gate slice

3 is the 2nd source/drain slice, and so on.

`begin`	equivalent to 0
`leftmost_sd`	`1`
`leftmost_gate`	`2`
`rightmost_gate`	far right gate slice
`rightmost_sd`	far right source/drain slice
`end`	to the right of rightmost_sd, these may be from 0 to 2 slices beyond the rightmost_sd slice.

any of the above symbols can be followed by:

```
offset <offset>
```

Where <offset> will be added to the value

For example:

```
slice_end end offset -5
```

will set the value up to (and including) the slice which is 5 slices to the left of the right-most slice.

The following have no default values and must be specified:

```
cost_type, cost, layer
```

The default values are as follows:

`y`	all
`slice`	all slices from `slice_start` to `slice_end`
`slice_start`	0

slice_end	end
slice_repeat	no repeat

If both slice and slice_repeat are specified, then slice_begin and slice_end can be used to further restrict which slices get applied.

If only slice is specified, then slice_begin and slice_end are ignored, and the cost is only applied to the specified slice.

The segment specified is always going from the location specified in the positive direction. For example, the ver segment goes from the y specified to y+1, the hor segment goes for the slice specified to slice + 1, and the contact segment goes from the layer specified to the next higher layer.

Any locations not covered by an override will continue to use the current method for determining costs.

For example, let's say a ProPPR setup contains metal-1 (m1) tracks as follows:

```
#   pro_set global proppr_numtracks_m1_chan2      1

#   pro_set global proppr_numtracks_m1_chan1      4

#   pro_set global proppr_numtracks_m1_chan0      1
```

To discourage vertical m1 routing on all the source/drain slices (which are odd number slices and starts from slice 1) in the center channel, add the following override_costs:

```
set override_costs {}

lappend override_costs \

{{layer m1} {cost_type ver} {slice_begin 3} \

{slice_end rightmost_gate} {slice_repeat 2} {y {3 4 5 6}} \

{cost 100000} {comment {Discourage vertical M1 on s/d slice}}}

# Add the override costs to proppr layercost list

pro_set global proppr_layercost_overrides $override_costs
```

Specifing layer cost overrides by net_names or net_types

All costs (but not overlaps) in proppr_layercost_overrides can be set either for only certain net-names or net-types (but not both).

For example, this can be used to set the m2 cost very high for all non-port nets, and normal for port nets. Alternately, it can be used to encourage/discourage certain nets in certain tracks, or force a net to use a specific location for horizontal tracks or contact/via location by making it cheap, and/or disallowing all other locations -- basically pre-routing.

For example, to allow m2 to be used only for port nets (assuming only y=4 and y=5 are the only m2 tracks available):

```
lappend override_costs [list \
          [list cost_type hor] \
          [list slice     all] \
          [list y         {4 5}] \
          [list layer     m2] \
          [list cost      1000000] \
          [list comment   "no m2 for non-ports"] \
          ]


lappend override_costs [list \
          [list cost_type hor] \
          [list slice     all] \
          [list y         {4 5}] \
          [list layer     m2] \
          [list net_types {input output}] \
          [list cost      20] \
          [list comment   "m2 for ports only"] \
          ]
set  proppr_layercost_overrides $override_costs
```

Limit Solution Searching

ProPPR finds many routing solutions. Each solution has a total cost as described in the previous sections. It is possible to set cost limits to improve the performance of the system. These limits cause ProPPR to stop searching for more solutions when an acceptable solution is found. The following options control the searching cost:

Parameter	Description	Default
`proppr_acceptable_cost`	The cost at which ProPPR will stop looking for solutions	0
`proppr_expected_cost`	The minimum cost that must be found before any solution is considered valid	undefined

Table 4

Parameter	Description	Default
`proppr_max_router_iter`	Limit the number of routing iterations	5000000

Table 4

proppr_acceptable_cost

proppr_acceptable_cost value

This option allows you to specify a cost at which ProPPR will stop looking for routing solutions. It will accept the first solution which has a cost less than or equal to value. When a cost is never found which is lower than value, the lowest cost found is considered good.

This can be useful when you have run a cell several times and you know from the log file at which solution it can stop.

In the following example, the first routing solution found which has a total cost of 2000 or less will be used as the solution:

```
pro_set global proppr_acceptable_cost 2000
```

proppr_expected_cost

```
proppr_expected_cost value
```

This option allows you to specify the minimum cost value must be found before any solution is considered valid. When no cost is found which is less than or equal to value, then no solutions are considered valid.

The setting in the following example instructs ProPPr not to consider any routing solutions as valid until one is found which is less than or equal to 3268.

This setting should only be used to speed up the search when you've already run ProPPR before. Generally, the acceptable and expected are set to the known solution cost.

```
pro_set global proppr_expected_cost 3268
```

proppr_max_route_iter

```
proppr_max_route_iter value
```

One of the advantages of ProPPR is its ability to find good routing solutions without searching the complete space of all possible solutions. It prunes large number of possible solutions based on heuristics. Because of this it has a built in limit to the number of solution spaces (iterations) it will check. The limit minimizes the run time while allowing it to find optimal solutions.

This option allows you to control the maximum number of solutions spaces the router will check before it exits. The default value is 5000000. Set the value to infinity to keep it searching until it has exhausted all possible solutions. For complex cells, setting the value to infinity can result in very long run times. In general, you should not adjust this number. If you think that there is a better solution that ProPPR has still not found, then you can try increasing this value. More likely, adjusting some of the cost information will result in a solution more to your liking, and much quicker.

The following example would increase the number or router iterations to 7,500,000 to see if there's a lower solution out there.

```
pro_set global proppr_max_route_iter 7500000
```

The following setting forces the router to run until it has searched all possible solutions.

```
pro_set global proppr_max_route_iter infinity
```

> There is another related option to limit memory usage: proppr_max_route_iter_after_soln. The option limits the number of routing solutions searched after a solution has been found. This can be used when a very large number of iterations are necessary to find solutions for large complex cells, but you do not want to wait that long for cells that can find a solution earlier.
>
> The number of transistors is not the only factor impacting the memory usage. The number of routing tracks available, of all the routing layers defined, plays an important role as well.

Proppr channel crossing costs

These costs control the router's preference for routing between channels. All of these can be set in the tech file with pro_set global <cost>.

proppr_polyover_cost - Cost of routing through a transistor
proppr_polyovergap_cost - Cost of routing through the gate of a passthru (routing in poly between two channels using the gate of a passthru)
proppr_m1over_cost - Cost of a route using a route over point
proppr_m1overgap_cost - Cost of using the route over point of a passthru

pro_set global proppr_m1over_2sd_cost 40

m1over_<n>sd_cost and m1overgap_<n>sd_cost are additional controls over the m1over_cost and m1overgap_cost, used to specify the cost as a function of how many source/drain contacts are immediately adjacent to the gate that ProPPR is routing over.

The variable names are:

pro_set global proppr_m1over_cost 20 ;# original variable, default is 20

pro_set global proppr_m1over_0sd_cost <val> ;# series stack on both sides
pro_set global proppr_m1over_1sd_cost <val> ;# series stack on one side
pro_set global proppr_m1over_2sd_cost <val> ;# no series, contact on both sides

Likewise, for routing over a passthrough gap:

```
pro_set global proppr_m1overgap_cost 20     ;# org var, default: same as m1over
pro_set global proppr_m1overgap_0sd_cost <val> ;# series stack on both sides
pro_set global proppr_m1overgap_1sd_cost <val> ;# series stack on one side
pro_set global proppr_m1overgap_2sd_cost <val> ;# no series, contact on both sides
```

Likewise, the same can be done for proppr_polyover_<n>sd_cost. For example, if there is not enough room to route a m1 over a transistor if there are source/drain contacts on both sides, you can make the the m1over in that case illegal as follows:

```
pro_set global proppr_m1over_2sd_cost    "infinity"
pro_set global proppr_m1overgap_2sd_cost "infinity"
```

This would still allow m1 to be routed over the gate of a transistor if the transistor is part of a series stack on one or both sides.

ProPPR port locations

ProPPR will pre-assign a port location, prior to determining the detailed routing, as follows:

For ports that connect to both a p-transistor drain and a n-transistor drain (i.e. output ports), no port pre-assignment occurs, because that route will always contain metal-1 and therefore the port location can be determined after routing.

For ports that have only gate connections (i.e. input ports), the following location is pre-assigned:

The y-value is the lowest y-value in the center channel that allows horizontal poly.

The x-value is the left-most x-value where both the p-transistor gate and n-transistor gate are connected to the port. If there is a group of multiple pairs all of which are connected to the input-port (for example a set of folded transistors), then the x-value will be left-most transistor of the largest such group.

If there are no p-transistor/n-transistor gate pairs, then the left-most transistor, whether it be a p-transistor or an n-transistor, will be used for the pre-assigned port location.

After routing, proppr will move the port from its pre-assigned location if it can eliminate a poly-contact by doing so.

Also, the new port optimization option, which can be used with or without proppr, will improve the port location.

Control ProPPR port locations

port location

```
pro_set port_info add {<portname>

{channel <ch>}

{bc_proppr_x_track <xnum>}

{bc_proppr_y_track <ynum>}
```

where <portname> is the name of the signal with the port

<ch>   is the channel number 0, 1, or 2

<xnum>   is the x-track to place the m1 port

the first gate is at 2, subsequent gates (or gaps)

x-locs

are at 4, 6, 8, etc. s/d locations are odd

<ynum>   is the track index for that channel

For example, if numtracks = 3, y can be 0, 1 or 2 default is highest usable track (largest index).

Both channel and bc_proppr_x_track and/or bc_proppr_y_track must be specified. For example:

```
pro_set port_info add {A {channel 1} {bc_proppr_y_track 1}}
```

sets port "A" to the second track in Y direction in center channel (channel 1) before compaction.

Also, bc_proppr_y_track_input and bc_proppr_y_track_output can be specified to use different bc_proppr_y_track locations for input and output ports. For example:

```
pro_set port_info add {PG_DEFAULT {bc_proppr_y_track_input  {1 2}}

{bc_proppr_y_track_output 0}

}
```

will cause proppr to reserve a staggered port location for all input ports, alternating between y-tracks 1 and 2 (y_n + 2, and y_n + 3) while the output port will reserve a location at y-track 0 (y_n + 1).

bc_proppr_y_track is allowed to be a list. If it is a list, the ports will be staggered between those tracks.

Ports are allowed to be placed on otherwise "unusable" tracks. If that happens, the port can be placed there, but no other routes. Vias to those ports (and possibly horizontal wires if the port spans more than one grid using bc_proppr_x_span_lo/hi) are also okay.

If you set up multiple templates in ProTech, you should use the following control in prospin.tcl to tell progen which template to use. Having multiple versions of progen.db for different templates is not necessary and hard to maintain.

```
#*
#* Select a cell template for a cell
#* templates names are 'multi-notched' and 'straight'.
#* Default is 'straight'.
#*

   prospin_set progen options "^FAHDLX$" {
       pro_set celltemplate dbname multi-notched
   }
```

Using "multi-notched" template is not necessary to give you better area density comparing to "straight" template because of the penalty you pay on the left and right edge of nwell interface. You should try both templates out to decide which one is better for a particular cell.

Fast Routing and Simple Nets

The fast routing option allows you to specify the nets that should be simple nets, which has only a single horizontal crossing of a "slice."

What is a ProPPR simple net? A simple net is one where there is only one routing wire for the net at any given X-value in the layout. The simple net may be made up of multiple wires and go across many layers, it just doesn't cross the same X-value more than once. A non-simple net has more than one routing wire at any given X-value.

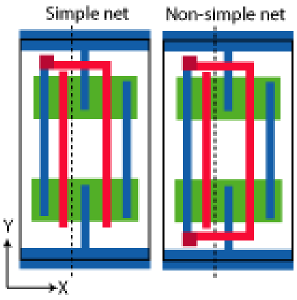


Figure 7: Simple vs. Non-Simple Nets

The default for ProPPR is to use both simple and non-simple nets when evaluating routing solutions. You can speed up the process by limiting ProPPR to just simple nets. It will then use the lowest cost routing solution consisting of only simple nets. If no routing solution exists with just simple nets, then it will automatically start searching for solutions with non-simple nets. The proppr_simple_nets option controls this setting. The following options control simple nets:

Parameter	Description	Default
`proppr_simple_nets`	List of nets to keep simple. Value of proppr_all_nets sets all nets to simple.	No simple nets

Table 5: Simple Nets option

proppr_simple_nets

This option allows you to turn on and off the use of simple nets, either for all nets or just for specific nets. Setting its value to the keyword proppr_all_nets tells it to try and use simple nets for all nets. Otherwise you can set it to the names of all the nets for which you want it to try and keep simple. Here are examples:

Turn on using simple nets for all nets:

```
pro_set global proppr_simple_nets proppr_all_nets
```

Example 6

Force ProPPR to try and keep the clock net simple:

```
pro_set global proppr_simple_nets clock
```

Example 7

Force ProPPR to try and keep the clock, Q, and Qbar nets simple:

```
pro_set global proppr_simple_nets {clock Q Qbar}
```

Example 8

Diffusion/Metal Wire Contact Objects (`dmwire`)

ProGen is an object-based system. When it creates the pre-compacted layout it is actually placing and connecting transistor objects, transistor-contact objects, routing wire objects, wire-contact objects, diode objects, tie objects, port objects, etc. Even the cell template is one large object in which all the other objects are placed. The specific objects used are based on system defaults, technology file settings, generator settings, and cell settings.

ProGen generally connects transistor sources to the power and ground rails using transistor diffusion-contact objects. The basic contact object is made up of diffusion, contact, and metal-1. Figure 8 shows a basic transistor diffusion contact connecting the transistor source to the power rail.

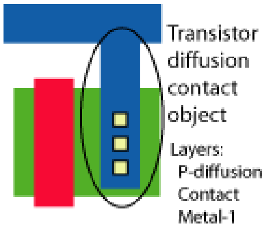


Figure 8: Transistor diffusion contact object

An alternative transistor-contact object is the dmwire object. This object starts out as a diffusion wire at the transistor source and then switches to metal-1 outside of the transistor. Figure 9 shows one of the possible layouts of a pre-compacted dmwire object. After compaction it may look the same as the basic transistor-contact object.

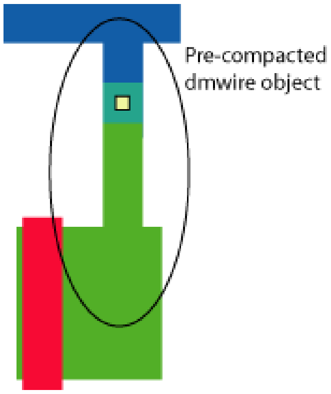


Figure 9: Transistor dmwire contact object

ProPPR is automatically set for dmwire objects and generally needs no adjustment. ProPPR automatically takes advantage of not having metal directly over the transistor initially when looking for a routing solution.

The following ProPPR dmwire controls are available:

Parameter	Description	Values	Default
proppr_dmwire	Turn dmwire use on and off	0 or 1	Same as cell template setting
proppr_dmwire_pos	Pre-compacted position of contact	midpoint or atrail or attran	Same as cell template setting
proppr_dmwire_midcon_ytrack_from_rail	Track number of contact	0, 1, 2...	Automatically calculated

Table 6: ProPPR dmwire Controls

proppr_dmwire

```
proppr_dmwire value
```

This control tells ProPPR if you are using dmwire objects. A value of 0 indicates that you are not using dmwire objects. A value of 1 indicates that you are using dmwire objects.

The default value is automatically determined and does not use dmwire objects. It is the same as the cell template setting. If the dmwire usage is off then the other parameters are ignored.

proppr_dmwire_pos

```
proppr_dmwire_pos value
```

This control tells ProPPR the pre-compacted position of the diffusion/metal contact of the dmwire. The values are:

`midpoint`
contact is at midpoint between transistor and rail

`atrail`
contact is centered in rail

`attran`
contact is on transistor source

Figure 10 illustrates pre-compacted layout for each of the above position types.

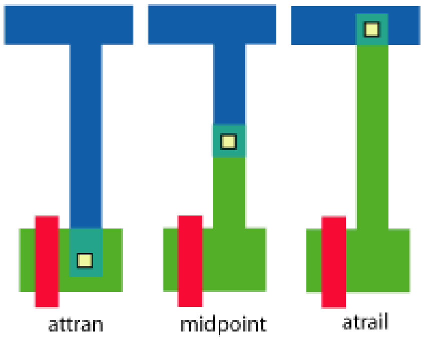


Figure 10. Precompacted position of the dmwire contact

The following example illustrates using a dmwire object initially located halfway between the rail and the transistor, demonstrating setting values specifying that a dmwire object is being used and that its initial diffusion/metal-1 contact position is halfway between the rail and the transistor.

```
pro_set global proppr_dmwire  1

  pro_set global proppr_dmwire_pos  midpoint
```

Example 9

The position specification is really not needed since midpoint is the default value.

In addition, the use of dmwires must be enabled in the option under the **Options** tab in ProTech with the **Diffusion/Metal contact wire** setting matching the one specified above.

proppr_dmwire_midcon_ytrack_from_rail

```
proppr_dmwire_midcon_ytrack_from_rail tracks
```

The value of tracks indicates the number of horizontal tracks between the dmwire contact position and the rail to which it connects in the pre-compacted layout. It provides a means to tell ProPPR exactly the location of the contact instead of the symbolic values specified with the proppr_dmwire_pos option.

This option does not control the location of the dmwire contact. It only tells ProPPR where it is located. The dmwire object has its own settings that allow you to control its behavior. See the ProTech and ProGen documentation for more details of advanced settings of the dmwire object.

This option is rarely used. It is typically only needed when you have overridden the starting location of the dmwire contact object and you need to tell ProPPR where it is.

Restricting Track Usage

ProPPR allows you to specify that a given y-track on a given layer is not usable. This can be used to create jumpers between the p-transistors and n-transistors in metal, and prevent poly from using the center track.

To specify the unusable tracks for a layer, set

```
proppr_unusabletracks_layer_channum tracknum_list
```

tracknum_list is a list of zeroes and ones. The length of this list should match the number of tracks for that layer and channel. The first entry on the list corresponds to the lowest track in the channel. For each 1 in the list, the corresponding y-track will be unusable for that layer (all slices).

By default, all tracks are usable.

Here is an example in which three poly tracks are defined in the center channel (chan1), but the center track in the channel is unusable:

```
pro_set global proppr_numtracks_poly_chan1 3

pro_set global proppr_unusabletracks_poly_chan1 {0 1 0}
```

Example 10. Setting the center track unusable for the poly layer

Similar restrictions are available:

```
proppr_unusable_hor_layer_channum tracknum_list
```

Set the track to be unsliceable, so no horizontal wires are allowed for that y-location

`proppr_unusable_via_layer_channum tracknum_list`
Disallow vias for the entire track specified

`proppr_unusable_via_by_sd_layer_channum tracknum_list`
Disallow vias for all source/drain slices in the entire track specified, but not gate slices

`proppr_unusable_via_by_gate_layer_channum tracknum_list`
Disallow vias for all gate slices in the entire track specified, but not source/drain slices

For example:

> In progen.tcl, two tracks of poly and m1 are defined in the lower channel (chan0), but only the lower track is usable by default for poly, m1, and cpoly. The specification is:
>
> pro_set global proppr_unusable_via_poly_chan0 {0 1}
>
> pro_set global proppr_unusable_hor_poly_chan0 {0 1}
>
> pro_set global proppr_unusable_hor_m1_chan0 {0 1}

Support for m2 Routing

The ProPPR router supports metal-2 routing. However, by default, only poly and m1 are used for routing. Specify the following parameters to control how m2 is routed.

Parameter	Description	Default
`proppr_numlayers`	Number of routing layers. A value of 2 means that routes are only made in poly and m1, while a value of 3 means routes caan also be made in m2	2
`proppr_layerver_costs`	List of vertical route costs, by layer: {`polycost m1cost m2cost`}	`{15 10 50}`
`proppr_layerhor_costs`	List of horizontal route costs, by layer: {`polycost m1cost m2cost`}	`{40 10 90}`
`proppr_via_costs`	List of contact and via costs: {`contactcost viacost`}	`{100 100}`
`proppr_numtracks_m2_chan0`	Number of m2 routing tracks in the lower channel	1
`proppr_numtracks_m2_chan1`	Number of m2 routing tracks in the center channel	2
`proppr_numtracks_m2_chan2`	Number of m2 routing tracks in the upper channel	1

Table 7. Metal-2 Routing Parameters

Routing wire objects

ProGen builds cell layout from objects. You can tell ProPPR which objects to use for horizontal and vertical routing wires in the outer channels. The following options control this:

Parameter	Description	Default
`proppr_horz_layernames`	List of objects to use for horizontal routing wires for each layer (poly, metal-1, metal-2); can be specified per channel, as described in Layer cost commands, below.	`{poly m1 m2}`
`proppr_vert_layernames`	List of objects to use for vertical routing wires for each layer (poly, metal-1, metal-2); can be specified per channel, as described in Layer cost commands, below.	`{poly m1 m2}`

Table 8. Outer channel routing control options

Layer cost commands

The cost per layer, per channel can be further sub-divided by y-track, but that is not recommended.

```
pro_set global proppr_layerver_costs {{75 15 75} 10 50}
```

Example 11

Example 11, then, will set the vertical poly cost for the outer channels to 75, and leave the rest at their defaults (10 for m1, 50 for m2).

```
   layerver_gate_costs {<cost_poly> <cost_m1> <cost_m2>}
```

Example 12

Example 12 results as above, except the vertical cost is only applied for slices (x-values) that align with gates.

Other slices will use the values in proppr_layerver_costs.

This defaults to proppr_layerver_costs.

```
   pro_set global proppr_layerver_costs {{75 15 75} 10 50}
```

Example 13

Example 13 results as above, except the vertical cost is only applied for slices (x-values) that align with source/drains (not gates).

Other slices will use the values in proppr_layerver_costs.

This defaults to proppr_layerver_costs. For example:

```
pro_set global proppr_layerver_sd_costs {infinity 10 50}
```

Example 14

Example 14 will disallow vertical poly on the non-gate slices, and leave the rest at their defaults (10 for m1, 50 for m2). Vertical poly on the gate slices will be as defined by either proppr_layerver_gate_costs (if any) or proppr_layerver_costs.

Note that the slices on the left and right edges of the cell are not considered gate or source/drain slices for this purpose and will always use the values of proppr_layerver_costs.

proppr_horz_layernames

`proppr_horz_layernames {polyObj m1Obj m2Obj}`

This option tells the system which objects will be used for horizontal routing wires for each of the layers: poly, metal-1, and metal-2. Of course, if metal-2 is not being used, then that value is ignored.

Presently, the main use of this option is to have the system use the jogged metal-1 wire object instead of the default straight wire. This jogged object has a lot of extra jogs, so the wires post-compacted layout may bend around contacts and other obstructions.

The following example sets ProPPR to using the default poly object for horizontal poly wires and the special jogged metal-1 wire object for horizontal metal-1 routes. In this example metal-2 is not turned on (proppr_numlayers option) so no metal-2 value was needed to be specified:

```
pro_set global proppr_horz_layernames {poly joggedm1}
```

Example 15

proppr_vert_layernames

`proppr_vert_layernames {polyObj m1Obj m2Obj}`

This option tells the system which objects will be used for vertical routing wires for each of the layers: poly, metal-1, and metal-2. If metal-2 is not being used, its value is ignored.

The following example sets ProPPR to using the default poly object for vertical poly wires and the special jogged metal-1 wire object for vertical metal-1 routes. In this example metal-2 is turned on (proppr_numlayers option), so the metal-2 value needed to be specified:

```
pro_set global proppr_vert_layernames {poly joggedm1 m2}
```

Example 16

Specify Early Termination

The proppr_max_route_iter global variable sets the maximum number of routing iterations that are searched before the router exits. The default value is 5000000. Use the value infinity to keep searching until all memory is consumed.

```
pro_set global proppr_max_route_iter infinity
```

ProGen Output For ProPPR Cells

Layout is created in ProGen with ProPPR in the same manner as cells that don't use ProPPR. While running, there will be some additional information output to the log file.

```
proppr_dmwire set to 1
```

reports that the PPR is using dmwire.

```
proppr_dmwire_pos set to midpoint
```

reports relative location of dmwire contact.

```
proppr_dmwire_midcon_ytrack_from_rail, set to 104
```

reports location of dmwire contact from rail set to 104.

Reading the ProGen Log File

There is information in the ProGen log file which is useful for tuning the router using the control options. The following is an excerpt from the ProGen log file when ProPPR is being used.

```
      #####
      proppr_numlayers  = 2
      proppr_numtracks_poly_chan0 = 1
      proppr_numtracks_poly_chan1 = 2
      proppr_numtracks_poly_chan2 = 1
      proppr_numtracks_m1_chan0 = 1
      proppr_numtracks_m1_chan1 = 2
      proppr_numtracks_m1_chan2 = 1
      proppr_numtracks_m2_chan0 = 1
      proppr_numtracks_m2_chan1 = 2
      proppr_numtracks_m2_chan2 = 1
      proppr_layerver_costs = 15 10 50
      proppr_layerhor_costs = 40 10 90
      proppr_via_costs  = 100 200
      proppr_dmwire = 1
      proppr_dmwire_pos = midpoint
      proppr_dmwire_midcon_ytrack_from_rail = 151
      acceptable_cost = 0
```

```
proppr_simple_nets  = clock
Calling proppr_route_engine
final cost = 2010
final cost = 1975
final cost = 1870
final cost = 1835
final cost = 1760
final cost = 1700
final cost = 1695
final cost = 1640
final cost = 1610
Highest slice processed = 14 of 14
Final cost = 1610
Number of branches searched = 224304
```

Look familiar? These values have been explained in the previous sections. Review the lines and see how well you understand the report before the following analysis.

Analysis

```
proppr_numlayers = 2
```

There are two routing layers being used: poly and metal-1. Metal-2 is not considered as a possible routing layer.

```
proppr_numtracks_poly_chan0 = 1

proppr_numtracks_poly_chan1 = 2

proppr_numtracks_poly_chan2 = 1
```

For poly routes, ProPPR assumes there is space for one horizontal track in the lower channel, two tracks in the center channel, and one track in the upper channel.

```
proppr_numtracks_m1_chan0 = 1

proppr_numtracks_m1_chan1 = 2

proppr_numtracks_m1_chan2 = 1
```

For metal-1 routes, ProPPR assumes there is space for one horizontal track in the lower channel: two tracks in the center channel, and one track in the upper channel.

```
proppr_numtracks_m2_chan0 = 1

proppr_numtracks_m2_chan1 = 2

proppr_numtracks_m2_chan2 = 1
```

If the router was using metal-2, then it would assume there is space for one metal-2 horizontal track in the lower channel, two tracks in the center channel, and one track in the upper channel.

```
proppr_layerver_costs = 15 10 50
```

The cost of vertical poly routes is 15, vertical metal-1 is 10, and vertical metal-2 is 50. This means that metal-1 routes are the least expensive and will be used most frequently, poly routes are little more expensive then metal-1, and metal-2 routes (if they were turned on) are very expensive.

```
proppr_layerhor_costs = 40 10 90
```

The cost of horizontal poly routes is 40, horizontal metal-1 is 10, and horizontal metal-2 is 90. This means that metal-1 routes are the least expensive and will be used most frequently, poly routes are will be used second, and metal-2 routes (if they were turned on) would be used least.

```
proppr_via_costs = 100 200
```

The cost of using contacts is 100, and using vias is 200. So, going from poly to metal-1 (which uses a contact) is less expensive then going from metal-1 to metal-2 (which uses a via).

Also notice that the cost of contacts and vias is quite a bit higher than the routing cost. This prevents the system from hopping back and forth between layers if it really isn't needed.

```
proppr_dmwire = 1
```

Use of the diffusion-metal wire contact is turned on (1). If the value was off (0), then the next two log lines would be meaningless.

```
proppr_dmwire_pos = midpoint
```

The pre-compacted location of the diffusion-metal wire contact is the midpoint between the transistor diffusion and the corresponding power/ground rail.

```
proppr_dmwire_midcon_ytrack_from_rail = 151
```

The number of pre-compacted tracks between the diffusion/metal contact and the rail center is 151.

acceptable_cost = 0

The value of zero indicates that there was no minimum limit set for calculating the final routing solution with the proppr_acceptable_cost option. ProPPR searched every solution trying to find the one with the lowest cost.

```
proppr_simple_nets = clock
```

Force the clock net to try and be a simple net.

```
Calling proppr_route_engine

  final cost = 2010

  final cost = 1975

  final cost = 1870

  final cost = 1835

  final cost = 1760
  final cost = 1700

  final cost = 1695

  final cost = 1640

  final cost = 1610

  Highest slice processed = 14 of 14

  Final cost = 1610
```

Here ProPPR was evaluating routing solutions and outputting the cost of different complete routes. The first solution it found had a total cost of 2010. The next better solution found had a cost of 1975, then 1870, down to 1610. The final and best routing solution had a cost of 1610.

```
Number of branches searched = 224304
```

This indicates the number of "iterations" that the router searched to find a solution. Anything less then the system default indicates a search of the complete solution space. See the description of proppr_max_router_iter if you need to increase the value beyond the system default.

Selecting, Configuring and Using ProPPR

Introduction

ProGen allows use of two routing methods: generator-defined routing, and the ProPPR router.

Generator-defined routing is the default method ProGen uses. This method makes use of the routing defined by the cell generators. When a cell is derived from multiple generators via the gen_multi generator, ProGen uses the routing within each generator for the intra-generator nets and derives the routes for the inter-generator nets.

The ProPPR router is much more powerful than the generator-defined router, and it ignores the routing defined in the generators to derive all the routes for the cell. ProPPR provides controls for directing and overriding routes.

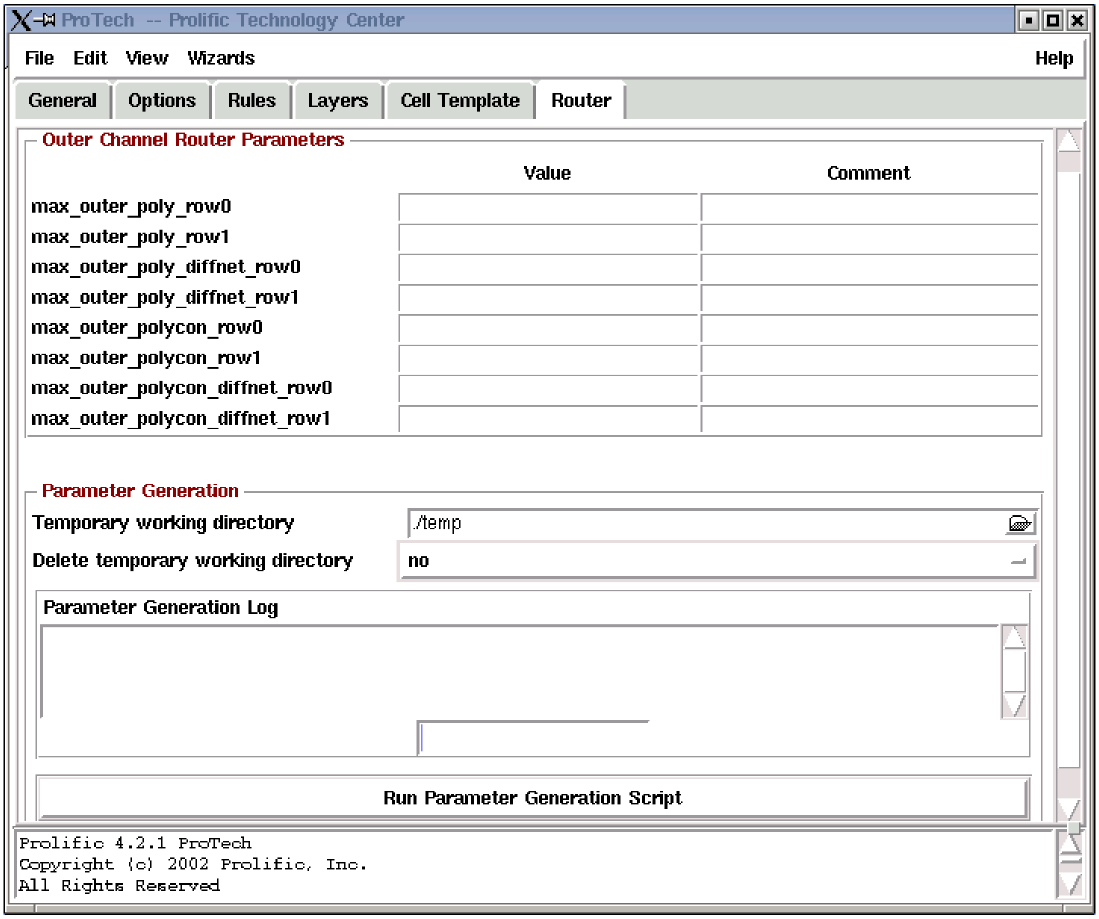
This section describes how to select a router, how to set up the system to use the ProPPR router, and how to control its behavior.

ProPPR Setup

Before you can use the ProPPR router, you must set routing parameter values in ProTech. These parameters are discussed in more detail later in this document; for setup purposes, there are three ways of generating these values:

- ProTech can generate these values
- The values can all be user-entered
- The values can be generated by ProTech, and some values can be user-overridden

ProTech-Generated Router Values



	Value	Comment
max_outer_poly_row0		
max_outer_poly_row1		
max_outer_poly_diffnet_row0		
max_outer_poly_diffnet_row1		
max_outer_polycon_row0		
max_outer_polycon_row1		
max_outer_polycon_diffnet_row0		
max_outer_polycon_diffnet_row1		

Temporary working directory: ./temp

Delete temporary working directory: no

```
Prolific 4.2.1 ProTech
Copyright (c) 2002 Prolific, Inc.
All Rights Reserved
```

Figure 3. ProTech Router Interface

The router values ProTech creates depend on all the other values defined for the technology: options, design rules, layers, and cell template values. For ProTech to generate the values, you must already have entered all these other data and saved them to a .db file, performing the following steps in ProTech:

1.Select the **Router** tab

2.In the **Parameter Generation** region:

a.Set the value of **Temporary working directory.**

This is the directory in which ProTech will run ProGen to create the router values. ProTech will create the directory if it doesn't already exist.

b.Set the value of **Delete temporary working directory**

Enabling this option instructs ProTech to delete the temporary directory after it has completed parameter generation. It is not necessary to delete the direction, however, and it will contain a log, cell (.cel) files, and agd files, which may be useful if parameter generation fails.

c.Click the **Run Parameter Generation Script**

ProTech will start generating parameters in the background. The current status is displayed in the **Parameter Generation Log** window, where the small status bar will move while the generation is running.

When the parameter generation is complete, ProTech will automatically fill in the **Outer Channel Router Parameter** values.

Manually Setting Routing Parameter Values

In ProTech, open the **Router** tab.

There are two areas containing configuration tables: one labeled **Outer Channel Router Parameters** and the other **Parameter Generation**. The value field of the **Outer Channel Router Parameters** table sets the maximum transistor size that is allowed for a particular routing configuration. There are four unique parameters that apply separately to the bottom row, located below the n-transistors, and the upper row, located above the p-transistors. In this context, the bottom row is denoted by the term row0, while the upper row is referred to as row1. The definitions for each of the parameters are shown below.

These outer channel parameters are numbered differently from normal channel references. The parameter names ending in 0 (zero) designate the bottom channel's settings, and parameter names ending in 1 designate the top channel's settings.

`max_outer_poly_row`

maximum transistor size to allow a route in poly above/below the transistor

`max_outer_poly_diffnet`

maximum transistor size to allow a different net signal to cross above/below the transistor

`max_outer_polycon`

maximum transistor size to allow a poly contact above/below the transistor

`max_outer_polycon_diffnet`

maximum transistor size to allow a route in m1 around a poly contact placed above/below the transistor

Whenever design rule values or cell template values change, the router values may need to be updated. For example, if the poly-to-diffusion spacing changes, then the router parameter values should be regenerated, since that changes the amount of space required to do outer-channel poly routing. Likewise, if the cell template height changes it may result in more or less routing area available.

All the values derived from running the test cases may be user-overridden by entering new values in the parameter fields. It's a good idea to enter a comment reflecting the fact that the value was overridden.

Identify overridden parameters with a comment for future reference and troubleshooting!

Once the router values have been entered, save the technology file before changes take effect.

Each of the max_outer_poly variables can have a list of maximum allowed values to support multi-track poly routing topology. They are ordered from lowest y to highest y; so in the lower channel, they should be decreasing, while in the upper channel, they should be increasing. A large transistor will first knock out the track location closest to the transistor, then knock out the track location closer to the rail. A medium-sized transistor can still allow the outermost poly route, but not the track closest to the transistor.

```
pro_set global max_outer_polycon_diffnet_row1 {6.9  9.4}

  pro_set global max_outer_poly_diffnet_row0    {2.2  1.6}
```

Pre-routing in PPR

There is now limited support for pre-routing in ProPPR. The pre-routed signals must be single-trunk nets.

Pre-routing is a way to route specific nets before others. By specifying the net, layer, and y-track, proppr will reserve the track for that net before other nets are routed. This is useful for cases in which the user has regular structured connections for a net, such as connections to guard bars in the center and edge of the cell.

```
# Format for proppr_preroute_info

# {

# {netname trunk_layer trunk_layer_name ytrunk trunk_must_use}

# {...}

# }
```

netname - name of net to be pre-routed

trunk_layer - proppr layer number to route main trunk

- 0=polydiff, 1=m1, 2=m2

trunk_layer_name - progen wire layer name for main trunk

ytrunk - proppr y-coord, depends on number of tracks set

trunk_must_use - limit the router such that it must use the pre-routed locations, instead of just reserving the locations for the net; if in doubt, set to 0.

The following example pre-routes the gnd and vdd nets in the center channel, connecting to the ndiff_tie and pdiff_tie wires that are pre-defined in the cell-template. When pre-routing vdd/gnd in the center channel, you must also pre-route the outer-channel vdd/gnd.

```
pro_set global proppr_preroute_info {

   {VDD 2 m2 12 0}

   {VDD 0 ndiff_tie 7 0}

   {VSS 0 pdiff_tie 5 0}

   {VSS 2 m2 0 0}
}
```

Setting Cells To Use ProPPR

Because ProPPR is not the default router, its use must be specified. This can happen in the .cel file or in the ProGen technology setup files. Whenever options are set this way, they are expressed in the same Tcl form:

```
pro_set global option value
```

Where option is any of the control options defined in this document and value is the corresponding control value. Here are some examples:

```
pro_set global proppr_dmwire          1

pro_set global proppr_dmwire_pos       midpoint

pro_set global proppr_numlayers        3

pro_set global proppr_horz_layernames  {poly joggedm1 m2}
```

Example 1

To apply the settings to a single cell, or to some subset of the cells in your library, specify them in the .cel file. If you want to apply the settings to all cells, specify them in the .cel file or ProGen technology files.

Setting options in prospin.tcl

To apply the control settings to a subset of all the cells in the library, define the settings in the ProGen input .cel files. Since ProSpin is typically used to create all the .cel files, put the settings in the ProSpin technology file.

This file is typically called prospin.tcl and is in the technology directory. Here is an example of a ProSpin technology file entry for a single cell:

```
    prospin_set progen options {^xor$} {
       pro_set global proppr_numlayers   3
       pro_set global proppr_simple_nets proppr_all_nets

   }
```

Example 2. Single-cell ProPPR configuration in prospin.tcl

Notice that the ProPPR settings are the same form as at the top of this section except that they are enclosed in ProSpin commands which specify in which cell to add the settings.

To apply the control settings to all the cells in the library, there are two equally effective techniques: Either set them in the ProSpin technology file as was done above for all cells, or set them in the ProGen Tcl technology file. The following example applies options to all cells in the ProSpin Tcl technology file:

```
prospin_set progen options {^.*$} {

  pro_set global proppr_numlayers   3

  pro_set global proppr_simple_nets proppr_all_nets

    }
```

Example 3. Configuring ProPPR for all cells in prospin.tcl

Setting options in progen.tcl

As just stated, applying the control options to all the cells being created using a specific technology can be done by setting the values in the ProSpin Tcl technology file or in the ProGen Tcl technology file. To apply options in the ProGen Tcl technology file, you just need to specify them in the options section of the technology file. A typical file may have the following settings:

```
proc set_options { } {

  pro_set global proppr_numlayers 3

  pro_set global proppr_simple_nets proppr_all_nets

}
```

Example 4. Configuration ProPPR for all cells in progen.tcl

As with the ProSpin example, the option settings follow the simple form and are enclosed in some ProGen code.

Router Selection

ProGen creates a routing topology in a cell using one of its two router engines. The default router is the Standard router, a "generator-defined channel router." The alternate router is the ProPPR router (nicknamed "proper"), which is an optimized exhaustive search router.

The user controls the router used for any given cell. This section discusses each router, explaining their differences as well as their strengths and weaknesses.

The routing topology created in a cell by ProGen is determined by one of the two routers built into ProGen. The Standard router is the default router and is described as a generator-defined channel router. The alternative is the ProPPR router and is an optimized exhaustive search router.

Standard Router

The Standard router uses the routing defined by cell generators to route nets internal to the generators. When multiple generators are used to create a single cell it uses a center channel routing algorithm to do the inter-generator routing. The center channel is defined as the open space between the nMOS and pMOS transistor rows. The Standard router provides very fast routing because it is primarily concerned with inter-generator routes. The algorithm considers primarily metal-1 and metal-2 routing layers with a little poly routing. It will use metal-2 routing when necessary for complex cells that require many generators and have little routing space available. It should be noted that most cells in a library are simple in nature (such as NAND, NOR, AND, OR, AO, OA, AOI, OAI, buffer, and inverter cells) and the routing defined in the generators to produce them is optimized to provide high density/high performance layout using the Standard router.

Because the Standard router is the default router, there are no special controls required to activate it. It supports numerous options for extended control of routes. See ProGen Guide for more information on using the Standard router.

ProPPR Router

The ProPPR router is a dynamic router that exhaustively searches all viable routing solutions. It is guaranteed to find a solution if one exists. The ProPPR router algorithm does a full route of all nets in the cell, discarding all generator-defined routes before starting. It considers all channels and will route across channel boundaries. It considers all valid routing layers, including poly routes. Its default behavior is to minimize the use of higher level metal usage. It also checks for transistor sizes in finding routing solutions to ensure that the layout after compaction will meet target cell height. The nature of the exhaustive search algorithm in general takes more time and memory to run than the Standard router.

Flops, latches, XORs, XNORs and muxes are examples of cells that are typically created by combining several different generators. The inter-generator routing can be complex and congested. These cells are ideal candidates for ProPPR.

To use ProPPR you must first set the ProTech technology Router information. You then need to add control information to all ProGen input CEL files. You can set-up ProSpin to put this information in automatically when the CEL files are created from the netlist. Continue reading this chapter for more information on using the ProPPR router.

Router Examples

The following two illustrations demonstrate potential differences in layout from the two routers. The examples are based on the same function, design rules, and cell template. Note that in the Standard router layout, almost all the routing is in the center channel and one metal-2 route was used. In the ProPPR layout, a poly route was used in the lower channel and no metal-2 was used. Also note the metal-1 routes that cross the transistor rows.

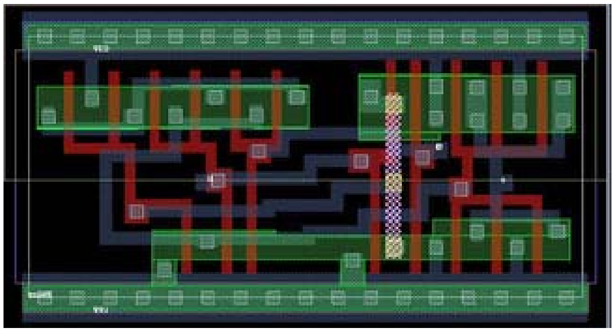


Figure 1. Standard Router Layout Example

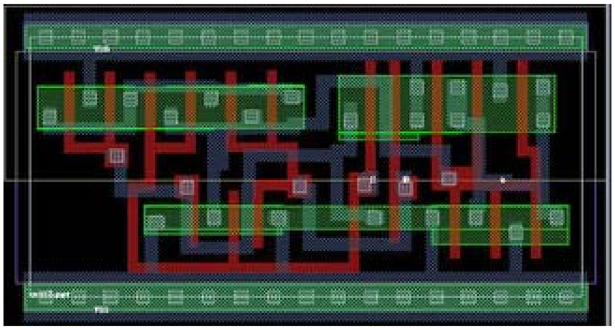


Figure 2. ProPPR Router Layout Example

Standard Router Costs: Changing the behavior of the standard router.

Unless specified otherwise, progen will use the standard router to generate a topology for a cell.

The following are costs that affect the behavior of the standard router. The higher the cost, the less likely the standard router is going to do what that cost corresponds to. For example, one of the costs is "edge_not_poly_port". This is the cost for not having an input port on the edge of the cell. By increasing this cost, you are likely to have input ports on the edge of the cell, and other routing will try to move to accommodate this. These costs are usually set in the options procedure in progen.tcl and can be set using the form:

```
pro_set costs  cost cost_value
```

Cost: first_cross

Default value: 1000

Description: Cost of first wire crossing. This generally results in m2, so the cost for using m2 is represented by this cost.

```
pro_set costs first_cross 1000
```

Cost: extra_cross

Default value: 101

Description: Cost of additional feed/input wire crossings.

```
pro_set costs extra_cross 101
```

Cost: distance_sd

Default value: 100

Description: Cost of distance to transistor source/drain connection. This is distance in the stitch-order list; if feed is between the leftmost and rightmost SD connection, this distance/cost will be 0.

```
pro_set costs distance_sd 100
```

Cost: distance_trunk

Default value: 10

Description: Cost of distance to center of input-trunk, numbered in units of entries in stitch-order list.

```
pro_set costs distance_trunk 10
```

Cost: feed_congestion

Default value: 501

Description: Cost of feed congestion. If two feeds must wrap around each other, this cost is added. This will prevent the feed from traveling too far away from its SD connections. Once the feed crosses over the SD of another feed, this cost is incurred.

```
pro_set costs feed_congestion 501
```

Cost: split_polycons

Default value: 1000

Description: Cost of splitting adjacent poly-contacts which are on the same net. For an inverter example, the stitch-order {lo f hi} would incur this cost, but {lo hi f} will not.

```
pro_set costs split_polycons 1000
```

Cost: edge_not_poly_port

Default value: 5

Cost: edge_not_feed_port

Default value: 5

Cost: edge_not_poly

Default value: 0

Cost: edge_not_feed

Default value: 0

Description: Cost of not having a port on the edge of the cell. For both the left and right edges, if the edge-most entry in the stitch-order is not a "lo" or "hi" of a port net, then the cost of edge_not_poly_port is added. If the edge-most entry in stitch-order is not a "f" of a port net, then the cost of edge_not_feed_port is added. If the edge-most entry in stitch-order is not a "lo" or "hi" of any net, then the cost of edge_not_poly is added. If the edge-most entry in stitch-order is not a "f" of any net, then the cost of edge_not_feed is added.

Example: {f lo hi f hi lo}

First entry of stitchorder is an "f" of an internal net, (feed). Last entry of stitchorder is a "lo" of a port net (poly port). Therefore, the cost is: left: edge_not_poly_port + edge_not_feed_port + edge_not_poly right: + edge_not_feed_port + edge_not_feed.

```
pro_set costs edge_not_poly_port 5

pro_set costs edge_not_feed_port 5

pro_set costs edge_not_poly 0

pro_set costs edge_not_feed 0
```

Cost: split_subcell_polycons

Default value: 0

Description: Cost of splitting adjacent poly-contacts that are in the same sub-cell.

```
pro_set costs split_subcell_polycons 0
```

Cost: horizontal_constraint

Default value: 99999

Description: This is the cost of violating user/system-specified constraints.

```
pro_set costs horizontal_constraint 99999
```

ProPPR FAQ

Why does bc_proppr_x_track or bc_proppr_y_track not output a port at desired location?

Using bc_proppr_x_track and bc_proppr_y_track, proppr will place a port on the input-trunk that is at the grid location specified. However, a port on an input-trunk (horizontal trunk) will always be placed at the left-most location of the input-trunk. This means that if the horizontal input-trunk that crosses the specified grid location continues to the left of that grid location, then the port will actually be placed on the far left location of that input-trunk. To further control the location of port, use "pos."

ProSticks Introduction

Prolific's ProSticks is a graphical cell editor that can be used to create or edit a cell's topology. It may also be used to create or modify generators. ProSticks is an X-Windows application that runs on any ProGenesis platform.

Running ProSticks

ProSticks is started from the command line using the **prosticks**, described here:

Name
prosticks -- Create or edit cell layout graphically

Synopsis
```
prosticks [-h] [-q] [-v] [pscfile]
```

Description
ProSticks is a graphical cell layout tool which allows creation or editing of cell layout from scratch or from a Spice netlist. ProSticks can save layout in its own format (.psc files) for future editing, but can also create generator (.cel) files. ProSticks can be used as a graphical, cell-based front end to ProGen as well.

Arguments
-h
Display usage message on stdout

-v
Output tool version number.

pscfile
Specify an existing ProSticks file to open. If the specified pscfile does not exist, a File->Open dialog will open when ProSticks starts up. If no pscfile is specified, ProSticks opens with an empty work area.

Files
```
.prolificrc
```

Specifies simple setup information for all Prolific tools. See prolificrc(5) for more details.

Author
Prolific, Inc. <applications@prolificinc.com>

See Also
.prolificrc(5), PROLIFIC(5)

ProSticks Hotkeys

Though this guide refers to menu selections and icons as means of accessing features, you may prefer to use hotkeys. Default hotkeys are listed below, followed by instructions so the keys may be programmed however you wish. For the sake of clarity, we do not often refer to each possible method of selecting any given feature, and we hope that you agree that this streamlines the user process. If you ever do run into trouble, click **Help** and you should find your way in short order.

Feature	Hotkey
Auto-route	a
Delete	d
Flip transistor	f
Move	m
n-diode	Alt-n
n-transistor	n
Pan	A
p-diode	Alt-p
Port	i
Properties	P
p-transistor	p
Resistor	R
Route	r
Scale transistors	s
Trace route	t
Undo	u
x-transistor	x
Zoom	z
Zoom in 2x	X
Zoom max	B
Zoom out 2x	O

Table 1

```
#

#  Prolific tools rc file (.prolificrc)

#

#*****************************************************************************
#

#*
                                                                          *#
```

```
#*  ProView
                                                                  *#

#*
                                                                           *#

#******************************************************************************
#

namespace eval PROVIEW {

  set key_bindings {

    {c         {panel display cross toggle}}

    {C         {panel display coordinates toggle}}

    {d         {panel distance}}

    {g         {panel display grid toggle}}

    {G         {panel display grid2 toggle}}

    {s         {panel select}}

    {w         {panel draw_wire}}

    {a         {panel draw_circle}}

    {l         {panel draw_line}}

    {Escape    {panel keyboard_entry}}

    {colon     {panel keyboard_entry}}

    {Left      {panel pan left}}

    {Right     {panel pan right}}

    {Up        {panel pan up}}

    {Down      {panel pan down}}

    {KP_Left   {panel pan left}}

    {KP_Right  {panel pan right}}

    {KP_Up     {panel pan up}}

    {KP_Down   {panel pan down}}

    {p         {panel pan select}}

    {o         {panel zoom x .5}}
    {x        {panel zoom x 2}}
    {z        {panel zoom select}}
    {B        {panel zoom max}}
    {b        {panel zoom backward}}
    {f        {panel zoom forward}}
    {0        {panel display depth 0}}
    {1        {panel display depth all}}
  }
  set control_key_bindings {
    {c         exit}
    {l        {panel refresh}}
    {KP_Left  {panel pan delta -1  0}}
    {KP_Right {panel pan delta  1  0}}
    {KP_Up    {panel pan delta  0  1}}
    {KP_Down  {panel pan delta  0 -1}}
  }
  set alt_key_bindings {
  }
}
#*****************************************************************************
#
#*
                                                                          *#
#*  ProSticks                                                        *#
#*
                                                                          *#
#*****************************************************************************
#
if 0 {
```

```
namespace eval PROSTICKS {
  set key_bindings {
    {n      {addobject ntran}}
    {p      {addobject ptran}}
    {t      {addobject xtran}}
    {R      {addobject resistor}}
    {P       property}
    {r       route}
    {m       move}
    {f       fliptran}
    {d       delete}
    {a       autoroute}
    {s       scaletran}
    {Left     {pan left}}
    {Right    {pan right}}
    {Up       {pan up}}
    {Down     {pan down}}
    {KP_Left  {pan left}}
    {KP_Right {pan right}}
    {KP_Up    {pan up}}
    {KP_Down  {pan down}}
    {A        {pan select}}
    {o        {zoom x .5}}
    {x        {zoom x 2}}
    {z        {zoom select}}
    {B        {zoom max}}
 }
```

```
  set control_key_bindings {
    {p       property}
    {r       {macro start}}
    {o       {macro end}}
    {t       {macro reset}}
    {s       {macro save}}
    {e       {macro run}}
  }
  set alt_key_bindings {
    {n       {addobject ndiode}}
    {p       {addobject pdiode}}
  }
     set log_file 0
}
}
```

ProSticks Menus

Index

    - Select Outlines
    - Icons
        - Icon Bar
        - Icon Labels
        - Small Icons
        - Large Icons
- Tools menu
    - ProGen
    - Net Browser
    - Check Opens
    - Check Ports
- Format menu
    - Add transistor row
    - Delete transistor row
    - Move transistor row
    - Add rail
    - Channel spread trunks (overlapped)
    - Channel spread trunks (no overlaps)
    - Channel height minimize
- Help menu
    - Contents
    - Concepts
    - Buttons
    - Menus
    - About ProSticks

Menu Bar

The ProSticks menu bar provides access to the various functions available via ProSticks' graphical interface. The menu bar contains the following entries:

- File Menu
- Edit Menu
- View Menu
- Tools Menu
- Format Menu
- Help Menu

File Menu

The File menu is used to manipulate cell files, generator files, and SPICE files.

New
Opens a new cell in the layout window.

All existing layout will be deleted before this operation runs. ProSticks will prompt you to save existing layout when it has been modified since the last save.

Open Cell

Open an existing cell file by selecting Open Cell and selecting the desired directory and file. You can navigate the directory structure by clicking on the Directory list and selecting an option, or by clicking on the parent directory icon to move up a level. When you've selected the file you wish, click Open. The selected file will be opened in the ProSticks interface. The status line will be updated to show that the file has been opened.

You can now manipulate the contents of the cell as you wish.

Save Cell

To save the updated layout, select the Save Cell option from the File menu. The status line indicates that the file has been saved. If the layout in ProSticks has never been saved before, the Save As dialog appears. If ProSticks has been used to edit existing layout, the layout is saved in the same file last saved or opened.

Save Cell As

To save layout under a different filename, use the Save Cell As option. This option allows you to specify the name into which you wish to save the layout.

You can select the directory and filename you wish to use to save the layout. Click on the Save button to complete the Save Cell As operation.

Open Recover File

Loads a ProSticks editing session recover file. A recover file is created for every cell being editing when the .prolificrc file option is enabled.

```
    set recover_on 1
```

All existing layout will be deleted before this operation runs. ProSticks will prompt you to save existing layout when it has been modified since the last save.

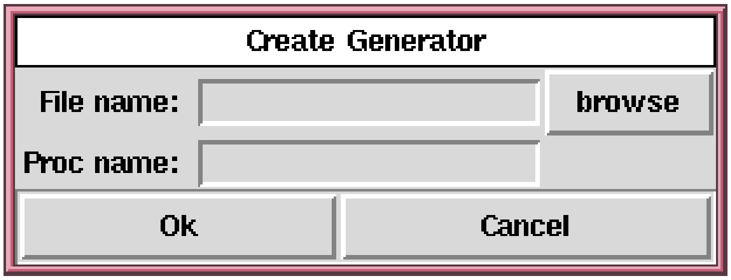
Load ProGen Interface File

Loads a ProGen interface file into ProSticks. A ProGen interface file contains the precompacted layout topology of a cell from a ProGen run. This layout can then be loaded into ProSticks, edited, and used to create a generator for a subsequent ProGen run.

All existing layout will be deleted before this operation runs. ProSticks will prompt you to save existing layout when it has been modified since the last save.

Create Generator

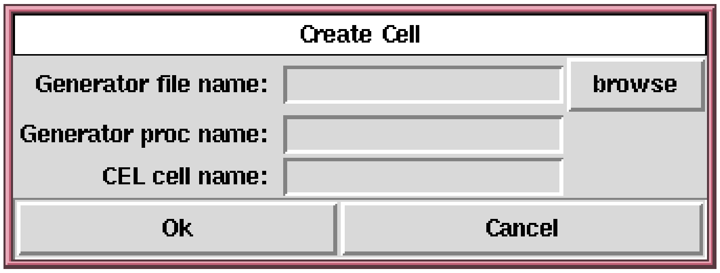
The Create Generator command produces a file containing the ProGen generator Tcl code. This makes ProSticks an easy way to create custom layout generators. Selecting this command opens the following dialog window:



File name
The name of the file to contain the generator Tcl code based on the ProSticks layout.

Proc name
The name of the generator Tcl procedure in the generator file. This name will be referenced in the .cel that you create either using ProSpin, ProSticks, or by hand.

Create Cell
The Create Cell command creates a ProGen .cel file that corresponds to the ProSticks layout. The .cel file identifies the generator that ProGen is to use to create the layout. Selecting this command opens the following dialog window:


Create Cell

Generator file name:

Generator proc name:

CEL cell name:

Generator file name
The name of the file containing the generator Tcl code based on the ProSticks layout. The generator file does not need to exist at the time that the .cel file is created.

Generator proc name
The name of the generator Tcl procedure in the generator file.

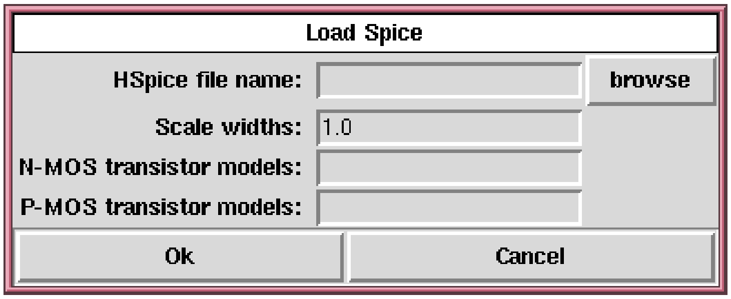
CEL cell name
This is the name of the cell that corresponds to the ProSticks layout. It is used to create the name of ProGen .cel file (*name*.cel) and is the cell name in the file. The file is always created in the working directory.

Create PostScript
This option allows you to output a graphical version of your ProSticks layout to a file for printing or viewing in PostScript format.

Spice

This menu option has three cascading sub-options: Load, Merge, and Create. These options can be used to import a netlist into a new ProSticks layout, merge a netlist into an existing layout, or create an HSpice netlist file from the ProSticks layout.

Load
The Load command opens the following dialog window that allows you to import an HSpice netlist into ProSticks to create a generator.



Scale widths: 1.0

HSpice file name
The file containing the HSpice to be loaded. This file may contain one or more circuits. When more then one circuit exists in the file, a circuit selection dialog will be displayed.

Scale widths
This entry allows you to specify the transistor width scaling factor to change the values from the netlist input values.

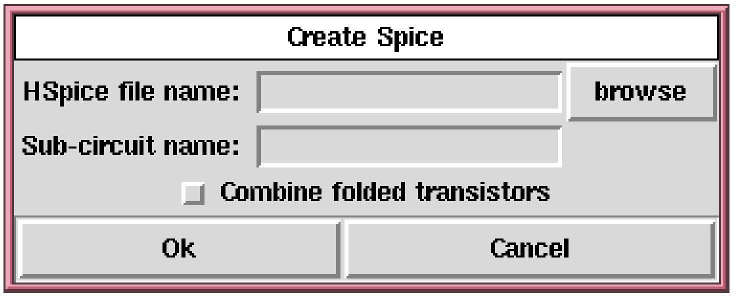
N-MOS transistor models
ProSticks will automatically identify NMOS transistors when the transistor model name begins with the letter “n.” This entry allows you specify one or more model names that do not start with the letter “n” so that they can be identified as NMOS transistors.

P-MOS transistor models
ProSticks will automatically identify PMOS transistors when the transistor model name begins with the letter “p.” This entry allows you specify one or more model names that do not start with the letter “p” so that they can be identified as PMOS transistors.

Merge
Use this command to merge a netlist into an existing ProSticks cell. Transistors in the netlist that don't have a corresponding named transistor in the ProSticks layout are added as new transistors. Transistors with matching names are not added but will have gate width and length values updated.

This command opens the same dialog window as the Load command above.

Create
The Create command opens the following dialog window that allows you to export the netlist defined by the ProSticks layout to a HSpice file.



□ Combine folded transistors

HSpice file name
The name of the file to which the HSpice netlist is to be written.

Sub-circuit name
The name of the circuit for the HSpice.

Combine folded transistors
By default, ProSticks does not evaluate the netlist to determine which transistors are folded. The netlist will have an entry for every transistor in the ProSticks layout. Enabling this option results in ProSticks determining which transistors are folded and outputting only a single transistor for each fold group.

Technology
The ProSticks technology file allows the user control some aspects of the ProSticks interface such as object colors and the list of ProGen symbolic objects available for the ProSticks objects. The system default settings can be found in:

```
$PROLIFIC/lib/prosticks/default.tec
```

The technology file can also be specified on the command-line when invoking ProSticks with the -t filename option:

```
prosticks -t tech/prosticks.tec inv.psc
```

For a description of the contents of technology files, see ProSticks - Technology Files.

Load
The Load command allows you to interactively load a technology file into ProSticks. All technology values are restored to the system default values before loading a new technology file.

Reset Defaults
This command restores ProSticks to its default technology settings. All settings loaded from a users technology file will be reset.

Exit
This command simply closes ProSticks.

ProSticks will prompt you to save existing layout when it has been modified since the last save.

Edit Menu

The Edit menu is a tear-off menu, which means you can keep the window open and move it around the screen This is convenient because the Edit menu is the one you'll use most as you lay out your cell generators. To tear off the menu, select the dashed line at the top of the menu. The menu will be placed in its own window, which you can move around the screen to wherever is convenient.

Undo
The first Edit menu option is Undo. Use this option to undo the last operation you performed. This is particularly handy when you mistakenly delete a component or decide you didn't like the results of the last move you made. The Undo function can be invoked multiple times to undo multiple actions - you can undo the operations back to the point when you created or opened the file, whichever happened most recently.

Undo Stack

This command opens a window that displays the complete list of undo commands. The list is ordered from the most recent editing command at the top of the list to the oldest at the bottom of the list, hence a stack. When you press the Undo button in the window with no entries in the list selected, ProSticks removes the top entry from the list and performs the undo operation. When you select an entry in the list and press the Undo button, ProSticks performs all the undo commands from the top of the list down to and including the selected entry. Double-clicking on an entry in the list has the same effect as selecting the entry and pressing the Undo button.

Undo All

This command will undo all operations on the existing layout back to the point where the layout was created of opened from a file, whichever happened most recently.

N-Transistor

After the the N-Transistor option is selected, N-transistors appear as green-colored blocks when you click on the ProSticks workspace.

If you import a netlist, the transistors will appear automatically. If you place them using these Edit menu options, they will snap to the dotted lines in the ProSticks workspace. This means if you attempt to drop a transistor somewhere other than a dotted line, ProSticks will pull it to the nearest dotted line. If you want more than two rows of transistors in the cell, see the description of the Add Transistor Row option from the Format menu option.

P-Transistor

Selecting P-Transistor allows you to create P-transistors, which are colored orange.

If you import a netlist, the transistors will appear automatically. If you place them using these Edit menu options, they will snap to the dotted lines in the ProSticks workspace. This means if you attempt to drop a transistor somewhere other than a dotted line, ProSticks will pull it to the nearest dotted line. If you want more than two rows of transistors in the cell, see the description of the Add Transistor Row option from the Format menu option.

X-Transistor

The X-Transistor option allows you to create white pass-through transistors.

If you import a netlist, the transistors will appear automatically. If you place them using these Edit menu options, they will snap to the dotted lines in the ProSticks workspace. This means if you attempt to drop a transistor somewhere other than a dotted line, ProSticks will pull it to the nearest dotted line. If you want more than two rows of transistors in the cell, see the description of the Add Transistor Row option from the Format menu option.

Route

The next two tools allow you to route the components of your cell. If you select the Route option, you can connect the elements you've placed in the cell. Route makes connections manually; after you select this option, click a gate, source, or drain and drag to another cell element to route it to.

Auto-Route

This option allows you to make use of network names to automatically generate routes connecting all common names.

N-Diode

This option allows the placement of N-type diodes for technologies that require them. To place a diode, click on the appropriate diode option from the Edit menu, and then click on the trunk where you wish to place the diode.

Note that the diode is connected to an extension of the poly. Because of the diode's placement requirements, you must use the Route function to add a trunk to the end of the poly on the transistor.

P-Diode

This option allows the placement of P-type diodes for technologies that require them. To place a diode, click on the appropriate diode option from the Edit menu, and then click on the trunk where you wish to place the diode.

Note that the diode is connected to an extension of the poly. Because of the diode's placement requirements, you must use the Route function to add a trunk to the end of the poly on the transistor.

Resistor

The Resistor option allows the placement of resistors where needed. Select this menu item and click and drag to place the resistor.

Port

The Port command allows the placement of ports in the layout. Ports can only be placed at the intersection of vertical and horizontal routes (feeds and trunks respectively). They can not be placed on power/ground rails. Also, the net name of the route must be defined before a port can be placed on it.

When the Port command is activated, the system will highlight the horizontal trunks as the mouse pointer passes over them on which ports may be placed.

Move

The next Edit menu option is Move. This function allows you to select any element in the workspace and move it somewhere else. When you click on the Move option, you will notice as you pass the cursor over the workspace, a dotted line will appear around elements on the screen.

To move a highlighted object, simply click it and drag it to the new location. You can use this technique to improve placement and routing in the cell to reduce cell size or optimize for materials.

Moving a single object often causes routes to cross, which means you'll have to make additional moves.

Flip Transistor

The Flip Transistor option is useful when you wish to align the connections between two transistors. Transistor flipping is also useful for aligning left and right feeds of N- and P-transistors to minimize jogs.

Transistor flipping is performed automatically when you use the Move command.

Properties

This menu command allows you to view and edit properties associated with layout elements. Elements that have properties include transistors, resistors, diodes, horizontal routes (trunks), vertical routes (feeds), and ports. Different element types have different properties. Properties include information such as net names, net types, and layers.

When the Properties command is activated, ProSticks will highlight the element as the mouse pointer passes over it that may be selected for property editing. When the mouse button is pressed, the edit properties dialog window will be opened for that element.

Transistors

Transistors have the following editable properties:

- Gate net

  Name of net connected to transistor gate.

- Left net
Name of net connected to transistor diffusion on the left side of the gate.

- Right net
Name of net connected to transistor diffusion on the right side of the gate.

- Gate width
Gate width in microns.

- Gate length
Gate length in microns.

- Template key
Override default ProGen object for transistor.

- Net-list include
Indicates if transistor should be included in spice netlist generation and ProSpin pattern generation. Setting the value to no results in the transistor not being included in the above and the transistor is drawn in a lighter then normal color. The default value is yes.

Resistors
Resistors have the following editable properties:

- High net
Name of net connected to high/top side of resistor.

- Low net
Name of net connected to low/bottom side of resistor.

- Template key
Override default ProGen object for resistor.

Ports
Ports have the following editable properties:

- Symbolic object
Override default ProGen object for port.

- Net type
Type of net that port is associated with. Built-in net types are input and output.

- Diodes
Diodes have the following editable properties:

- Diode type
Indicates if diode should be created with n-diffusion or p-diffusion.

- Trunk side
Diodes can only be placed on the left (lo) or right (hi) side of horizontal wires (trunks).

- Extend direction
Diodes connect to the trunk on which they are placed and extend either to the left (lo) or right (hi).

- Grid span

This indicates the number of routing grids that the diode diffusion should span. The default of 1 results in a minimum size diffusion contact diode.

Trunks

Trunks have the following editable properties:

- Net name

Name of net associated with trunk route.

- Layer

Force ProGen to create trunk route on the specified layer. Leave this the empty string to let ProGen determine the layer.

- Width

Force the width of the route to be other then the design rule minimum width for the layer.

- Track

Force ProGen to place the trunk on the specified track number in the precompacted layout.

Power/Ground/Rail Trunks

Power, ground, and rail trunks have the following editable properties:

- Net name

Name of net associated with trunk.

- Layer

Force ProGen to create rail on the specified layer. Leave this the empty string to have ProGen use the layers specified in the cell template definition. Rails that are not defined in the cell template must have layer specified here.

- Width

Force trunks of type rail to width in microns. Ignored for power and ground trunks.

- Map ID

Power/ground/rail trunks defined in the cell template have an unique identifier associated with them that is used to map the ProSticks trunk to the cell template trunk. This is used when there are more then the default 2 rails (power and ground) in the cell template. The ID is ignored for rails not defined in the cell template.

- Type

Indicates if trunk is a power, ground, or rail type.

- Defined

Indicates if the trunk is defined in the cell template definition or should be added to the cell generator. The typical case is for power and ground rails to be defined by the cell template.

Jogs

Jogs are the routes that connect transistor and resistors to horizontal trunks. They have the following editable properties:

- Net name

Name of net associated with jog routes.

- Contact symbolic object

  Force ProGen to use the object as the transistor/resistor contact instead of letting it decide which to use.

- Feed layer

  Force the layer of the route from the contact to the horizontal trunk.

Feeds

Feeds are vertical routes between 2 horizontal trunks. They have the following editable properties:

- Net name

  Name of net associated with feed route.

- Symbolic object

  Force ProGen to use the object to create the feed route instead of letting it decide which to use.

Trace Route

The Trace Route command is used to highlight named nets in the ProSticks layout. As the mouse pointer passes over named nets it highlights all routes and nodes with matching net names. The name of the net is displayed in the ProSticks bottom-right status window. Clicking on a highlighted net will leave the net highlighted until the trace route command is terminated or another net is clicked on.

Trunk Split

This command is used to split a horizontal trunk into 2 pieces and connect them with a vertical feed. Power/ground rails can not be split. When the command is activated, the trunk to be split will be highlighted as the mouse pointer passes over it. The split will be placed at the point on the trunk where the mouse pointer is located.

Delete

The final two options on the Edit menu allow you to delete objects from the workspace. The Delete command allows you to select individual objects for deletion.

If you wish to undo the last deletion, you can select the Undo command from the Edit menu. Remember that this will only undo the last action you took.

Deleting one item often implies deletion of another, and ProSticks will delete any objects that are dependent upon an item you're deleting.

Delete All

If you wish to delete everything you've done, the last option on the menu is the best approach. Simply select Delete All from the Edit menu. A warning is generated if the layout has been changed since the last save; to continue the deletion, click OK, to completely clear the workspace of objects. This will delete all of them. This step cannot be undone, so be careful, okay?

Scale Transistors

The Scale Transistors command is used to scale the width of all transistors in the layout. The width of each transistor is multiplied by the scale value. For example, set the scale to 0.9 to resize all transistors to be 90% of their current width. Set the value to 2.0 to double the size of all transistor widths.

View Menu

The View menu allows you to set some ProSticks configuration options.

The dotted line means that this is another tear-off menu, so if you select the dotted line, a separate Options menu window will be opened.

Pan

The Pan option centers the layout on the next point clicked. Select Pan and click on the point that should be at the center of the layout area.

Zoom

The Zoom option allows selection of the viewable area. Select Zoom, then click in the layout area and drag to create a rectangular region. Release to make the box the entire viewable area.

Zoom In 2X

The Zoom In 2X option sets the zoom level to twice the current setting, giving a more detailed view.

Zoom Out 2X

The Zoom Out 2X option is used to zoom out by a factor of two, giving a view of a larger area.

Zoom Max

The Zoom Max option sizes the layout to completely fill the viewable area.

Node Names

The Node Names option is a toggle. If the checkbox is marked, node names are displayed in the ProSticks work area. If the checkbox is unmarked, node names are not displayed. This can be useful when examining routing in large cells.

Device Names

The Device Names option is a toggle. If the checkbox is marked, device names are displayed in the ProSticks work area. If the checkbox is unmarked, device names are not displayed.

Select Outlines

The third item on the Options menu is Select Outlines. The default setting causes an outline to appear around cell objects when you've selected a menu option that acts on cell objects. This is usually a useful feature because it helps you see what element you will be performing the action on. If you wish, you may turn this off by unchecking the Select Outlines box.

Icons

The Icons entry is used to control the display of the ProSticks tool button bar. This menu option is a tear-off list, so it may be opened as a separate window by selecting the dashed line at the top of the submenu. The four suboptions include:

Icon Bar

This option, which is enabled by default, displays the buttons in the tool bar. Deselect this option to remove the tool bar from the ProSticks interface. If this option is not selected, the other options on the Icons submenu will not be relevant.

Icon Labels

This option, which is enabled by default, displays text labels on the buttons in the tool bar.

Small Icons

This option, which is enabled by default, displays small button icons in the tool bar. Select Large Icons instead to display bigger button icons.

Large Icons

This option displays large button icons in the tool bar. Select Small Icons instead to display smaller button icons.

Tools Menu

ProGen

This option allows you to run ProGen using the generator you've created in ProSticks, allowing you to stay in ProSticks while you test the output of your generator. Once you see the ProGen results, you can immediately make changes to the generator from within ProSticks and run ProGen again.

Select the directory and filename of the tech file you wish to use. Specify the generator file name and procedure name as described in the Create Generator option described previously. Remember to use names you'll be able to identify later.

Note that there is a line of dashes at the top of this menu. This means that this is a tear-off menu, or one that can be left open and moved around the screen. We will see other tear-off menus in ProSticks; whenever you see the line of dashes at the top of a menu, you know you can tear off the menu by selecting the dashed line.

The options listed on this menu allow you to specify easily the ProGen command-line debug options. For more detailed information on these options, refer to the Running Existing Layout Generators.

After you have set the debug options as you wish, you can return to the Run ProGen window to set the compaction options by clicking the compact options button.

The option you select controls the order of the axis of compaction.

After you have set the compaction options, click OK on the Run ProGen menu to start the layout and compaction. As ProGen starts, the layout will be placed and compacted.

As the compaction process progresses, the size of the cell will decrease in the ProGen window. Click the resize button to expand the view to make the cell layout more visible.

You can save a copy of the finished cell as a PostScript file by clicking the print button. This will bring up a file-saving window.

You can view the ProGen progress by looking at the log file. Simply press the log button.

Net Browser

This command opens a window that list all the named nets in the layout with the following information:

- Nets
  Name of nets

- Status
  Indicates if the net is fully connected (no entry) or open.

- Type
  Indicates the type of net. Values are: input, output, power, ground, rail, or internal (no entry).

The nets window can remain open at all times and its contents will be automatically updated during editing operations.

You can select nets in the list and they will be displayed selected in the layout. You can edit net names and types in the dialog and it will update all the routes and objects in the layout with the corresponding net names.

Check Opens

The Check Opens tool searches the current ProSticks cell for open nets. Signals that are not fully routed are identified.

Check Ports

Use this command to have ProSticks report all input/output nets that do not have ports. ProSticks outputs the names of missing ports in the status window. Use the Tools -> Net Browser command to highlight the nets.

Ports are not required to produce valid generators in ProSticks.

Format Menu

The next menu allows you to modify aspects of the default workspace. The options on the Format menu allow you to alter the layout of the transistor rows and the power and ground rails.

This is another tear-off menu, so you can select the dashed line to open an independent window containing the Format menu options. The first three menu items allow you to add, remove, or move transistor rows from the workspace.

Add Transistor Row

You might wish to add additional transistor rows if you want to stack transistors. You can add a row simply by clicking Add transistor row and then clicking on the workspace. A dotted line representing the transistor row appears.

The first transistor you add to the row determines whether it is a row of N-transistors or P-transistors. You are free to add additional transistors to the row, just as you would to the two that ProSticks provides by default. The white X-transistors are used to route across the transistor rows. Any feed wire crossing a transistor row must do so through a pass-through transistor, also known as an X-transistor. These objects maintain valid routing when there are more than two transistor rows.

Delete Transistor Row

If you wish to remove a transistor row, simply click the Delete transistor row option and click on the row you wish to remove. Note that the row must be empty before it can be deleted, so you must use the Delete or Move command to remove the transistors from the row before you can remove the row. Also note that the Delete All command does not delete transistor rows.

Move Transistor Row

If you wish to move a transistor row, you can click on the "Move transistor row" command and then click and drag the row to its new location. Channels are only enlarged when transistor rows are moved. To decrease the size of a channel use the "Channel height minimize" command first and then enlarge using this command.

Add Rail

The remaining option on the Format menu allows you to add additional power or ground rails. While this feature is not often required, you do have full control over the environment. To add a new rail, click Add Rail, then click on the workspace where you'd like to place the rail. The new rail can be manipulated using the Edit menu commands such as Move and Delete. Use the Properties command to identify a rail's properties, including whether it was defined in the cell template or added in ProSticks.

Channel spread trunks (overlapped)

The "Channel spread trunks (overlapped)" command distributes the horizontal trunk and jog wires in the selected channel such that they are evenly distributed across the height of the channel. Wires that are overlapping will remain overlapping. The bottom-to-top order of the trunks and jogs relative to each other will not be changed.

Channel spread trunks (no overlaps)

The "Channel spread trunks (no overlaps)" command attempts to distribute the horizontal trunk and jog wires in the selected channel such that they are evenly distributed across the height of the channel. Wires that are overlapping will be separated. The bottom-to-top order of the trunks and jogs relative to each other will not be changed (except to eliminate overlaps).

The command may not be able to evenly distribute the routes if it requires more tracks than are available because of separating overlapping routes. It will display an error message that indicates how many tracks are needed. You should enlarge the channel by moving one of the transistor rows that borders the channel and then redo the spread command.

Channel height minimize

The "Channel height minimize" command decreases the height of the selected channel to what it considers the minimum height without overlapping horizontal wires that are not already overlapping. Horizontal wires that are already overlapping will remain overlapping. The bottom-to-top order of trunk and jog wires is not changed.

Help Menu

Use the Help menu to find help related to the interface and function of ProSticks.

Contents

This is a top-level list of the help available in ProStick's graphical interface.

Concepts

A high-level view of why to use ProSticks.

Menu Bar

Describes the ProSticks menus.

Tool Bar

Describes the function of the buttons on the ProSticks toolbar.

About ProSticks

Provides application and contact information.

ProSticks Technology File

The ProSticks technology file provides a way for users to add information to the ProSticks interface and to change display settings. The file is a Tcl script that ProSticks sources. The following types of information (options) may be specified in the technology file:

- Display colors
- List of possible ProGen symbolic objects for ProSticks objects
- Generator creation options for ProSticks jog/contact objects

The default values for all ProSticks technology file settings can be found in the file:

```
$PROLIFIC/lib/prosticks/default.tec
```

Where `$PROLIFIC` is the directory where the Prolific tools are installed.

Technology File Input

There are several ways to load a technology file into ProSticks. The technology file can be loaded from the user interface menu commands, it can be specified on the command line, and it can be specified in the user's .prolificrc file.

Menu Command

The technology file can be loaded from the ProSticks menu command File>Technology>Load.... This command first resets all technology values to the system defaults and then loads the new technology file settings.

Command Line

The technology file can be specified on the ProSticks command line with the -t filename option. For example:

```
prosticks -t prosticks.tec inv.psc
```

A technology file specified on the command line supersedes one specified in the .prolificrc file.

Prolific RC file

A technology file may be specified in the user's .prolificrc file in the PROSTICKS namespace section using the following command:

```
set technology_file tech-file
```

A technology file specified on the command line supersedes one specified in the .prolificrc file.

Basic Commands

Before describing the details of each specific setting the following describes the basic commands used to set values in the technology file. There are two commands: tech_set and tech_add. The commands have the same syntax but behave slightly differently.

The specification form of each command is:

```
        ::prosticks::tech_add option {

           { option-entry }

           ...

        }

        ::prosticks::tech_set option {

           { option-entry }

           ...

        }
```

tech_add

The tech_add command is used to append new values to existing option-entry values. When no option entries match, the values are set the same as the tech_set command. Use this command when you don't wish to lose the systems existing values but want to add new values to them.

tech_set

The tech_set command is used to set new option-entry values and to replace existing values. It will either overwrite the existing values or replace them.

Display Colors

Many of the colors used by ProSticks to display objects may be changed to suit your personal preferences using the `colors`, `colors_layer`, and `colors_symobj` options.

tech_set colors

The `colors` option sets the default colors for objects. The command specification syntax is:

```
   ::prosticks::tech_set colors {

      {type type   color color}

      ...

   }
```

Note that `tech_set` should always be used for the colors option because you must replace the system default values.

The value of *color* may be specified by name or in the X11 #rrggbb hexadecimal form. The system will report an error if the name is not defined in X11.

The following system colors can be changed in the technology file:

Type	Description
background	main window background, {} = default background color
grid	transistor row grid lines
ndiff	n-diffusion for transistors and diodes
pdiff	p-diffusion for transistors and diodes
xdiff	x-diffusion for transistors and diodes
gate	transistor gates and jogs connected to gates
gate-alt	jogs connected to gates when ProGen object is changed
feed	non-gate jogs and feeds
feed-alt	non-gate jogs & feeds when ProGen object is changed
port	ports
port-alt	port when ProGen object is changed
res-body	resistor body (squiggly line)
res-con	resistor connection wire (vertical line)
trunk	input trunks and rails
trunk-alt	trunks connected to gates when ProGen object is changed

Example:

The following demonstrates setting the background window to white, the n-transistor diffusion to lightgreen, and all transistor gates to the color corresponding to the hexadecimal code #7a5b00:

```
::prosticks::tech_set colors {

   {type background  color white}

   {type ndiff       color lightgreen}

   {type gate        color #7a5b00}
}
```

tech_set colors_layer

The `colors_layer` option sets the default color of objects that allow the specification of the object layer in the Properties dialog. The command specification syntax is:

```
::prosticks::tech_set colors_layer {

   {layer layer   color color}

   ...

}
```

Note that `tech_set` should always be used for the `colors_layer` option because you must replace the system default values.

The value of color may be specified by name or in the X11 `#rrggbb` hexadecimal form.

The following are the system defaults:

Layer	Color
poly	red
m1	blue
m2	navyblue
m3	orange

Example:

The following demonstrates setting the color of all m2 to darkgreen and m3 to pink:

```
::prosticks::tech_set colors_layer {

   {layer m2  color white}

   {layer m3  color pink}

}
```

tech_set colors_symobj

The `colors_symobj` option sets the default color of objects that allow the specification of the ProGen symbolic object in the Properties dialog. This setting supersedes the `colors_layer` setting. The command specification syntax is:

```
::prosticks::tech_set colors_symobj {
        {obj obj   color color}
        ...
    }
```

The value of color may be specified by name or in the X11 #rrggbb hexadecimal form.

There are no system default settings.

Example:

The following demonstrates setting the color for several common symbolic object names:

```
::prosticks::tech_set colors_layer {
        {obj congnd    color yellow}
        {obj convdd    color pink}
        {obj joggedm1  color #3a3b7c}
}
```

ProGen Symbolic Objects

ProSticks knows about all of the built-in ProGen symbolic objects that may be associated with the varying types of ProSticks objects. Occasionally users create custom ProGen objects to perform some specialized behavior. These new ProGen objects can be added to the technology file so that they show up in the ProSticks Properties dialog's list of choices for the corresponding ProSticks object.

For example, we may want to create a ProGen feed (a vertical route) that routes in 2 metal layers at the same time on top of each other (lets call it feed_m1_m2). In ProSticks, the ProGen feed can be specified in the "Symbolic object" entry of the Properties dialog for any feed. But it must by hand-typed every time we wish to specify it. By adding the following entry to the technology for the feed we can get the name to show up in the pull-down list for the Properties dialog "Symbolic object" entry so that we don't have to hand-type it every time.

```
        ::prosticks::tech_add symobjs {
            {type feed obj feed_m1_m2}
        }
```

All the objects are specified with the symobjs option:

```
        ::prosticks::tech_add symobjs {
            object specifications ...
        }
```

ProGen symbolic objects can be specified for the following ProSticks objects:

- feed
- jog
- port
- resistor
- transistor

There are slight variations between the object specification for some of the objects.

Feed

Field	Values	Description
type	feed	Apply to all feeds
obj	*ProGen object list*	List of one or more ProGen symbolic objects

Example:

Add the two feed objects feed_m1_m2 and feed_m2_m3 to the display list of all feeds.

```
::prosticks::tech_add symobjs {

   {type feed  obj {feed_m1_m2 feed_m2_m3}}

}
```

Jog

Field	Values	Description
type	jog	Jog that connects to transistor
dif	n | p | x	Type of transistor
obj	*ProGen object list*	List of one or more ProGen symbolic objects

Note! See the Jog/Contact Symbolic Object Generator Options section below for addition information that may need to be specified when jog objects are added to the system.

Example:

Add the special left/right jog/contact objects to the display list of all jogs that connect to the corresponding diffusion type.

```
::prosticks::tech_add symobjs {

   {type jog  dif n  obj {conndif_lef conndif_rig}}

   {type jog  dif p  obj {conpdif_lef conpdif_rig}}
```

```
      }
```

Port

Field	Values	Description
type	port	Apply to all ports
obj	*ProGen object list*	List of one or more ProGen symbolic objects

Example:

Add the single port object port_3grid to the display list of all ports.

```
      ::prosticks::tech_add symobjs {

         {type port  obj port_3grid}

      }
```

Resistor

Field	Values	Description
type	resistor	Apply to all resistors
obj	*ProGen object list*	List of one or more ProGen symbolic objects

Example:

Add the res_3ohm and res_5ohm objects to the display list of of all resistors.

```
      ::prosticks::tech_add symobjs {

         {type resistor  obj {res_3ohm res_5ohm}}

      }
```

Transistor

Field	Values	Description
type	tran	Apply to transistors
dif	n | p | x	Type of transistor
obj	*ProGen object list*	List of one or more ProGen symbolic objects

Example:

Add variations of n, p, and x-transistors to the display list of the corresponding diffusion type of all transistors.

```
::prosticks::tech_add symobjs {

   {type tran  dif n  obj {trann_lef trann_rig}}

   {type tran  dif p  obj {tranp_lef tranp_rig}}

   {type tran  dif x  obj  tranx_rot}

}
```

Jog/Contact Symbolic Object Generator Options

When custom jog symbolic objects are added using the symobjs option there are additional properties related to those that correspond to ProGen transistor diffusion contacts that ProSticks needs to know to properly create a generator from them. These properties are defined using the difcon_gen_options option. It has the specification form:

```
::prosticks::tech_set difcon_gen_options {

   {obj obj  option property  value value}

   ...

}
```

The value of obj is the name of the object that is typically also defined with the symobjs option. The corresponding values of property and value are:

Property	Value=1	Value=0	Default
unique_intrunk	Create unique input trunk for each instance of jog/ feed	Do not create unique input trunks for multiple instances	0
can_merge	Jog can be merged with other jogs/feeds when aligned	Do not merge jogs/feeds	0
has_intrunk	Always set net_info type to “hasintrunk”	May or may not set net_info type	

Setting the value of Property to { } (empty list) means to set the corresponding value for all jogs that have no symbolic object override specified in the ProSticks layout.

Example:

The following sets the value for the conndif_lef and conpdif_rig objects to true (1) for all properties:

```
::prosticks::tech_set difcon_gen_options {
   {obj conndif_lefoption unique_intrunkvalue 1}

   {obj conpdif_rigoption unique_intrunkvalue 1}

   {obj conndif_lefoption can_merge  value 1}

   {obj conpdif_rigoption can_merge  value 1}

   {obj conndif_lefoption has_intrunkvalue 1}

   {obj conpdif_rigoption has_intrunkvalue 1}

}
```

Navigating ProSticks

The ProSticks interface features several components: the Menu Bar, Tool Bar, Status Line, Workspace, and Message Area.

Menu Bar

The menu bar (Figure 1) contains the tools for configuring and using ProSticks. Its six menus are:

- File
- Edit
- View
- Tools
- Format
- Help

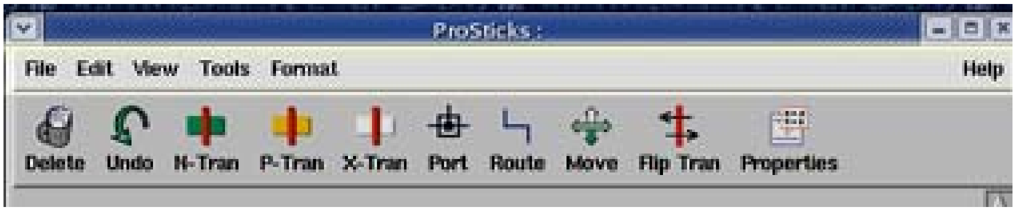


Figure 1

File Menu

Several options may be selected from the File menu: New, Open Cell..., Save Cell, Save Cell As..., Open Recover File..., Load Progen Interface File..., Create Generator..., Create PostScript..., Spice, or Exit.

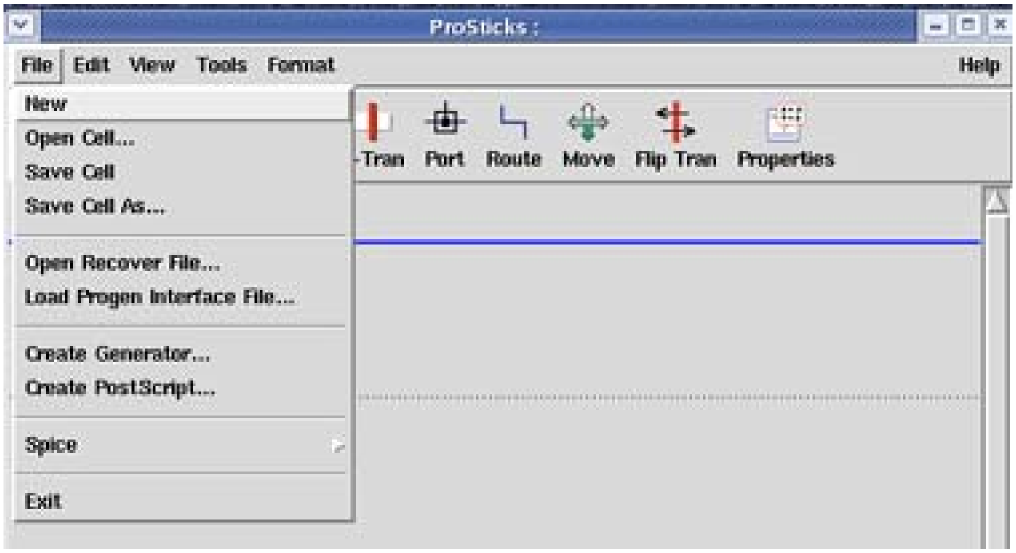


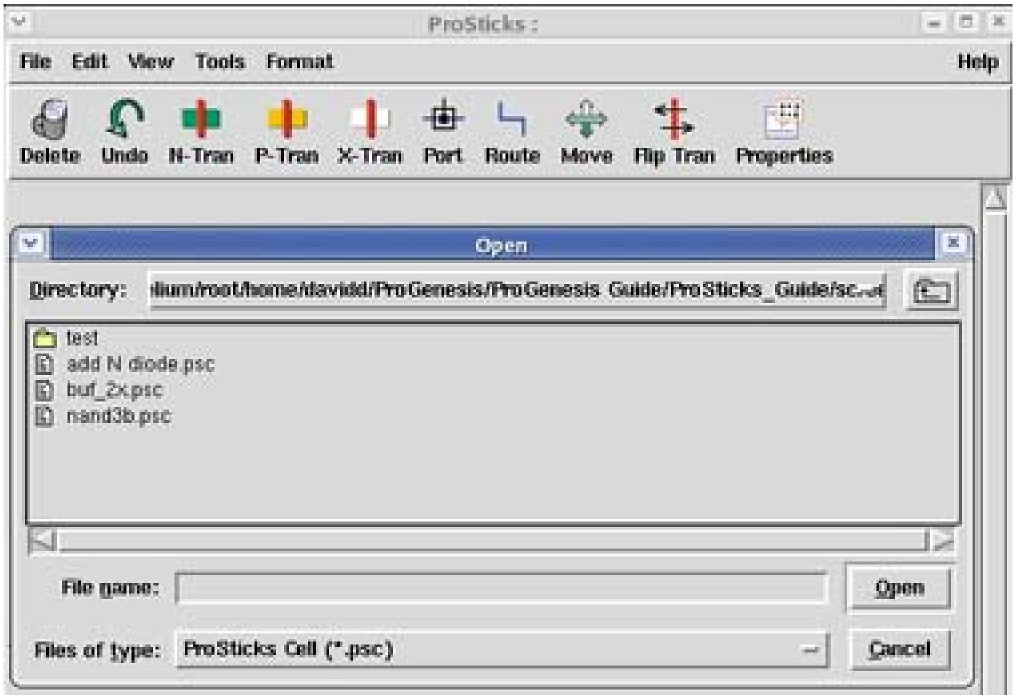
Figure 2

New

Clears workspace to start a new cell.

Open Cell...

This menu option opens an existing cell file. A dialog box, shown in Figure 3, allows the selected file to be opened.



Directory: llium/root/home/davidd/ProGenesis/ProGenesis Guide/ProSticks_Guide/sc...

test
add N diode.psc
buf_2x.psc
nand3b.psc

File name:

Files of type: ProSticks Cell (*.psc)

Figure 3: Selecting File>Open Cell... to open an existing file

In Figure 3, you'll see that you can navigate the directory structure by clicking on the directory list and selecting a file, or by clicking on the folder icon to move up a directory level. (The default file type is .psc.) When you've selected the file you wish, click Open. The selected file will be opened in the Pro-

Sticks interface as shown in Figure 4. Notice that the status window has been updated to show that the file has been opened.

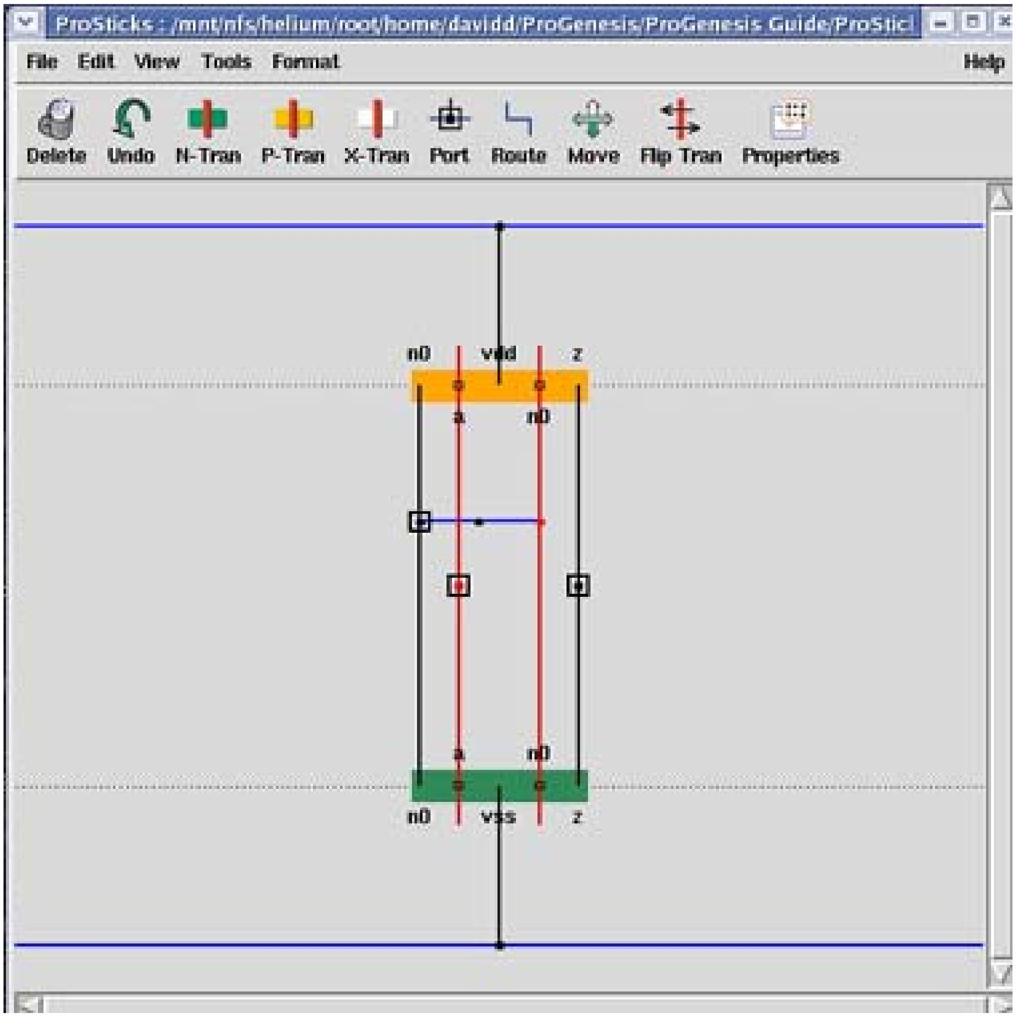


Figure 4: Opening a cell in ProSticks

You can now manipulate the contents of the cell as you wish.

Save Cell

To save the updated layout, select the Save Cell option from the File menu. The status window indicates that the file has been saved. If you've created a new layout, the Save As... dialog appears. If you've edited existing layout, it is saved in the same file you last saved or opened.

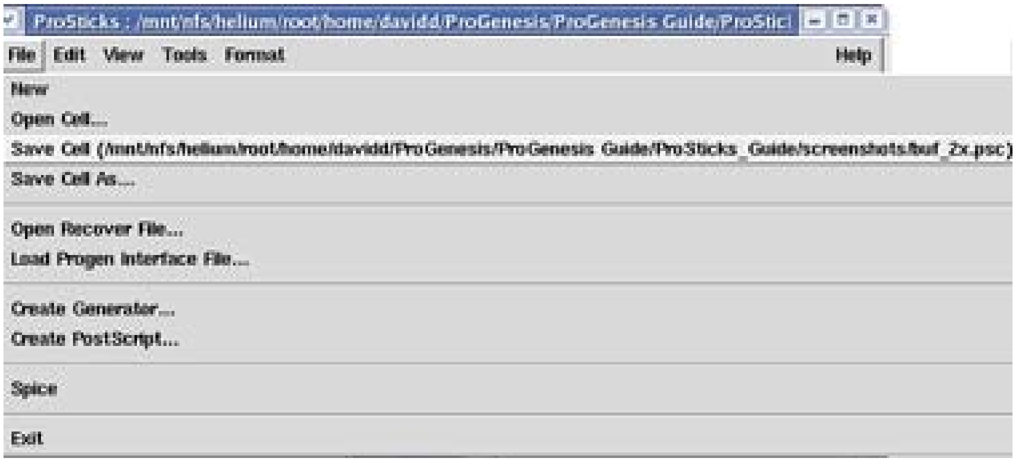
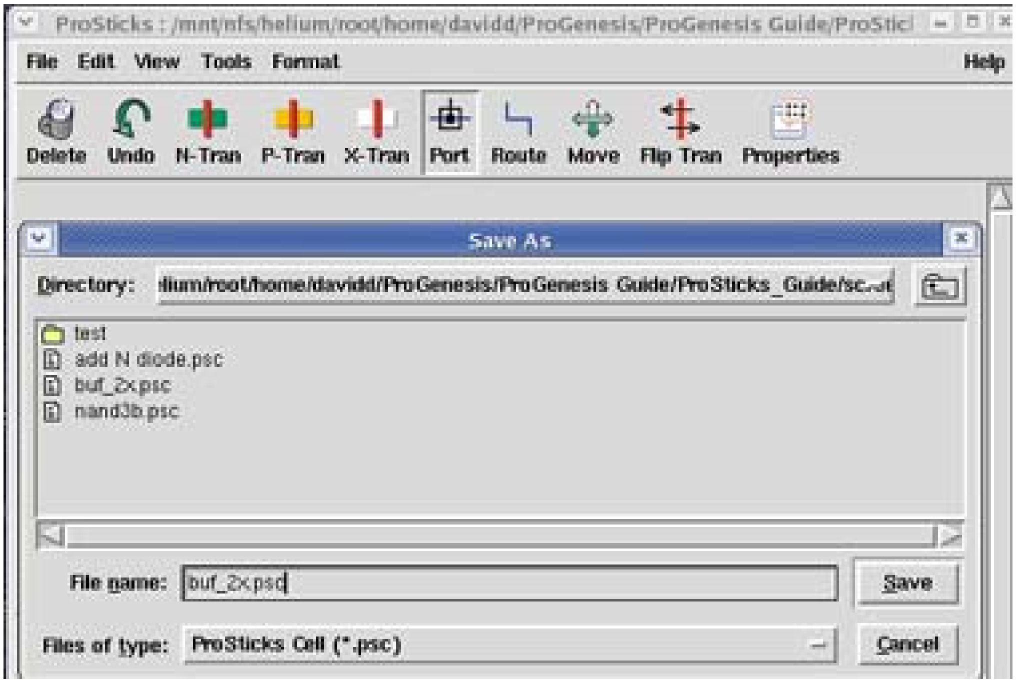

ProSticks : /mnt/nfs/helium/root/home/davidd/ProGenesis/ProGenesis Guide/ProStic
File
Edit
View
Tools
Format
Help
New
Open Cell...
Save Cell (/mnt/nfs/helium/root/home/davidd/ProGenesis/ProGenesis Guide/ProSticks_Guide/screenshots/buf_2x.psc)
Save Cell As...
Open Recover File...
Load Progen Interface File...
Create Generator...
Create PostScript...
Spice
Exit

Figure 5

Save Cell As...

If you wish to save the layout under a different name, you may use the Save Cell As... option from the File menu. This option allows you to specify the name into which you wish to save the layout. When you select File Save Cell As..., a window like the one shown in Figure 6 opens up.



Directory: ılium/root/home/davidd/ProGenesis/ProGenesis Guide/ProSticks_Guide/sc...

test
add N diode.psc
buf_2x.psc
nand3b.psc

File name: buf_2x.psc

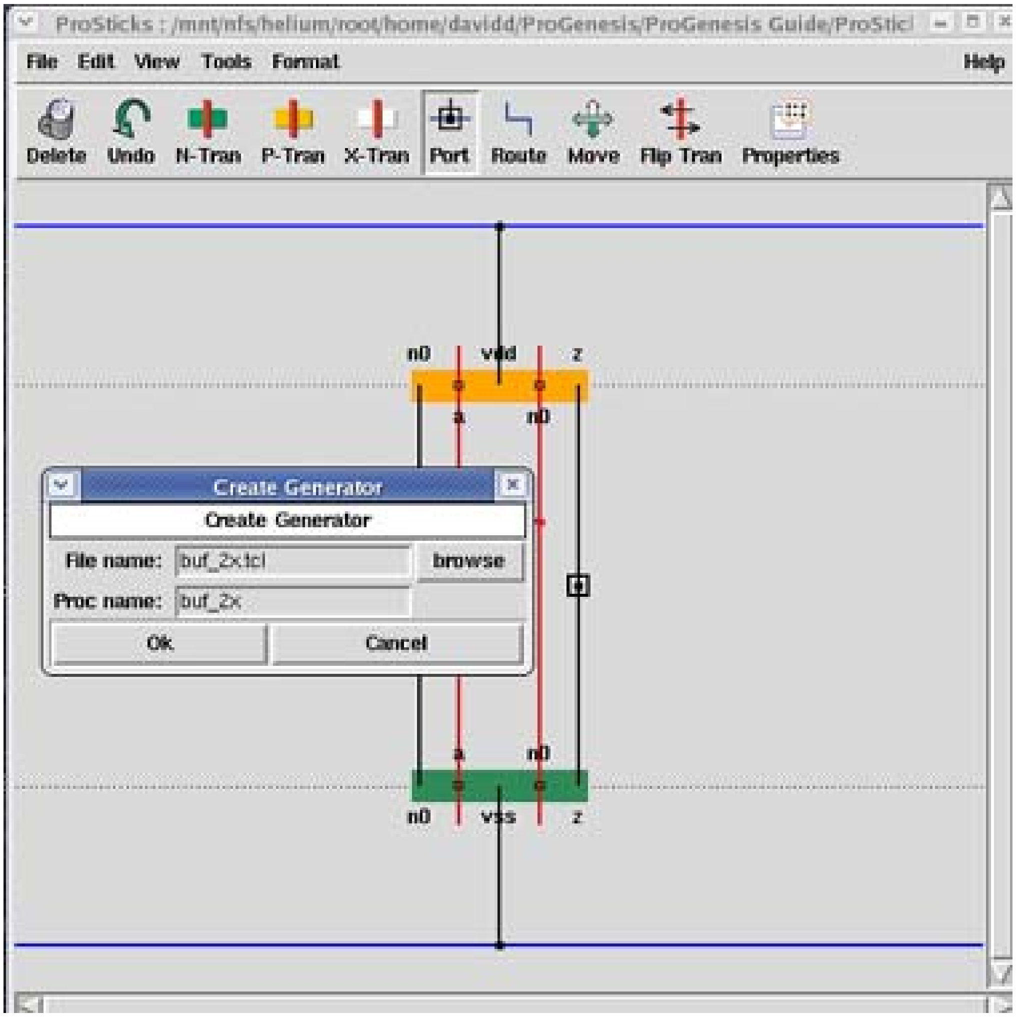
Files of type: ProSticks Cell (*.psc)

Figure 6: Use Save Cell As ... to specify a new file name for the layout

You can select the directory and filename you wish to use to save the layout. In Figure 6, the ProSticks layout will be saved as /prolific/psc/buf_2x.psc. Click on the Save button to complete the Save Cell As... operation.

Create Generator

The next option on the File menu is Create Generator.... This option produces a Tcl file describing the generator for ProGen without forcing you to know Tcl. You can always modify the output if you're familiar with Tcl. This makes ProSticks an easy way to create custom layout generators. To use this option, select Create Generator... from the File menu. This brings up the Create Generator window shown in Figure 7.

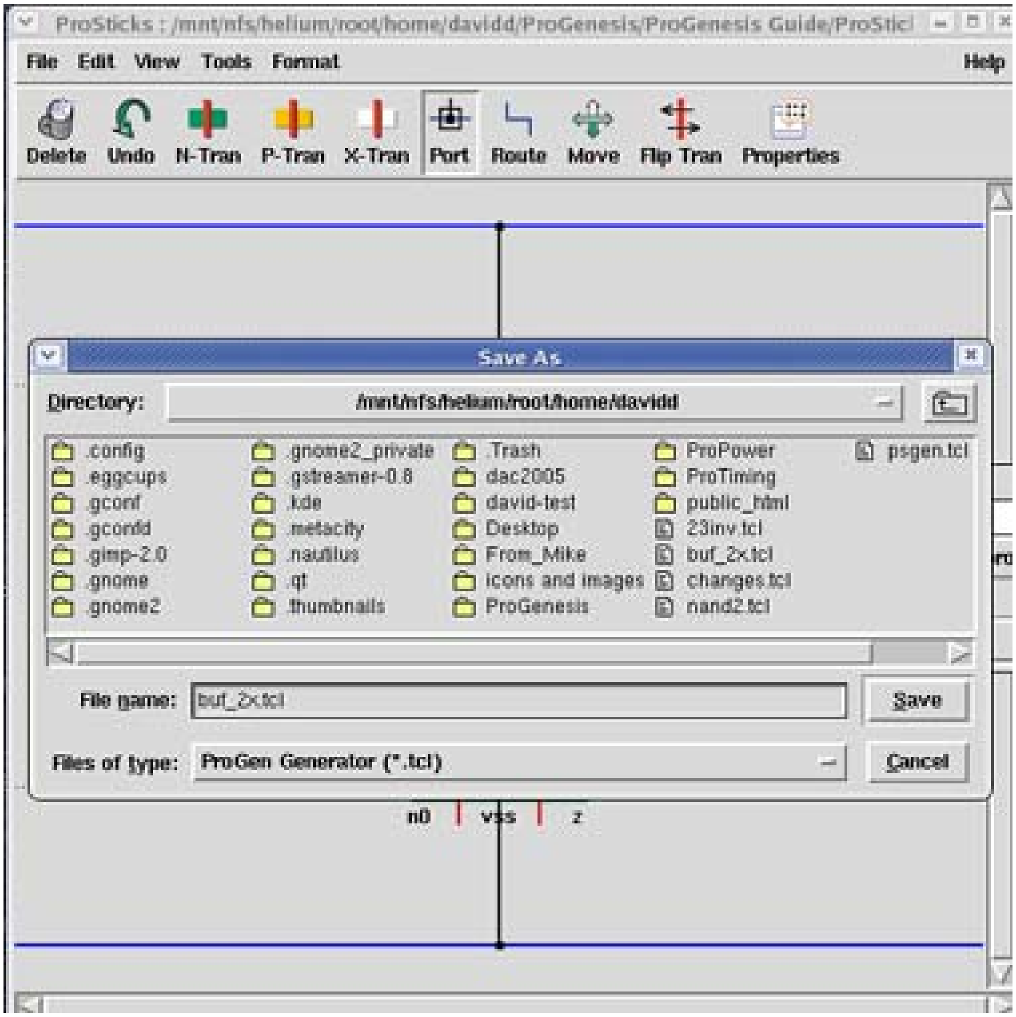


File name: buf_2x.tcl

Proc name: buf_2x

Figure 7: Use the Create Generator ... option to output a layout generator Tcl file

Once you've opened this window, you can enter the desired filename for the layout generator. Use the browse button to find the directory in which you wish to locate the file, as shown in Figure 8.



ProSticks : /mnt/nfs/helium/root/home/davidd/ProGenesis/ProGenesis Guide/ProStic

Save As

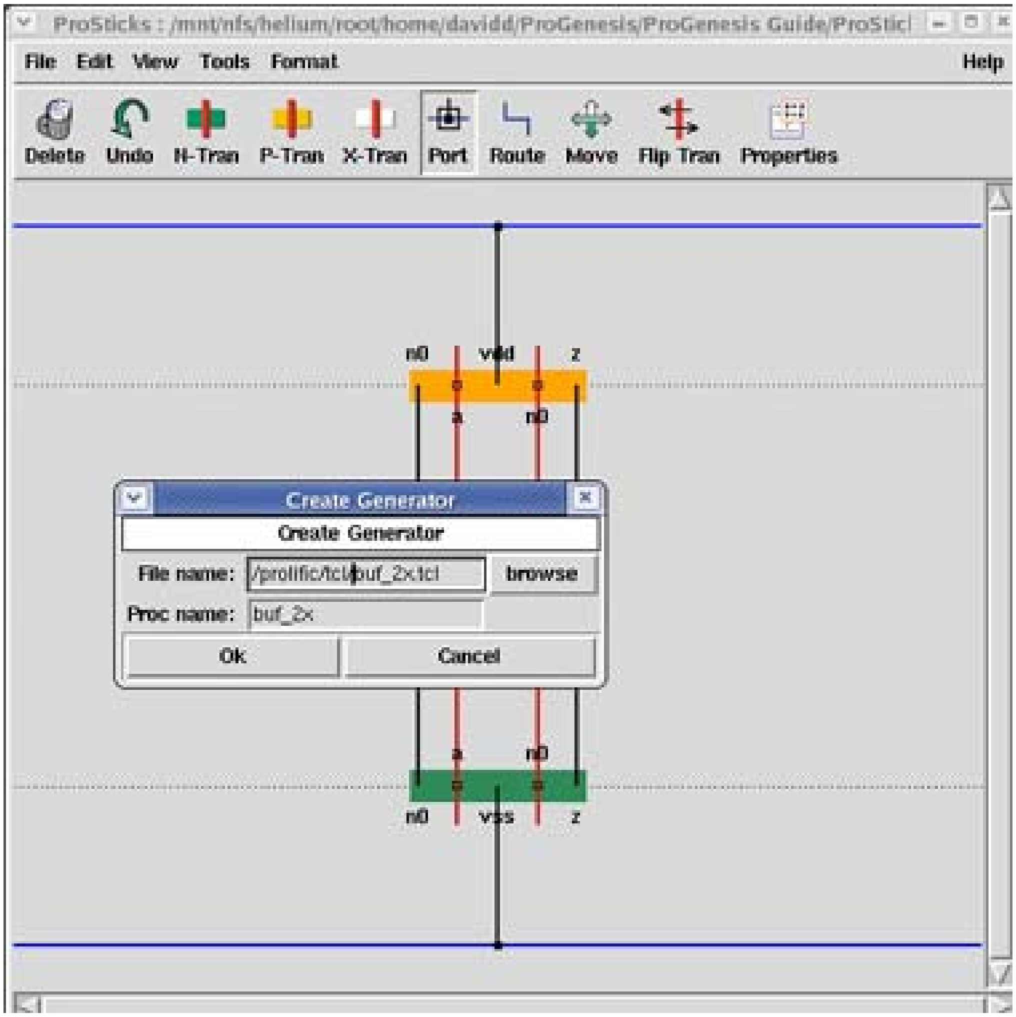
Directory: /mnt/nfs/helium/root/home/davidd

File name: buf_2x.tcl

Files of type: ProGen Generator (*.tcl)

Figure 8: Select the directory in which you wish to place the new layout generator

Enter the name that will call this procedure in the Proc name: field. These values will default to psgen.tcl and psgen as shown in Figure 8. Although these defaults will function properly, you may wish to use more descriptive names. For example, our cell might use a filename like buf_2x.tcl and a procedure name of gen_buf_2x or gen_buf, as shown in Figure 9. The name in the Proc name: field is the actual generator name to be called by ProGen. The File name: field simply indicates the name of the file in which that procedure is defined.



Create Generator

File name: /prolific/tcl/buf_2x.tcl

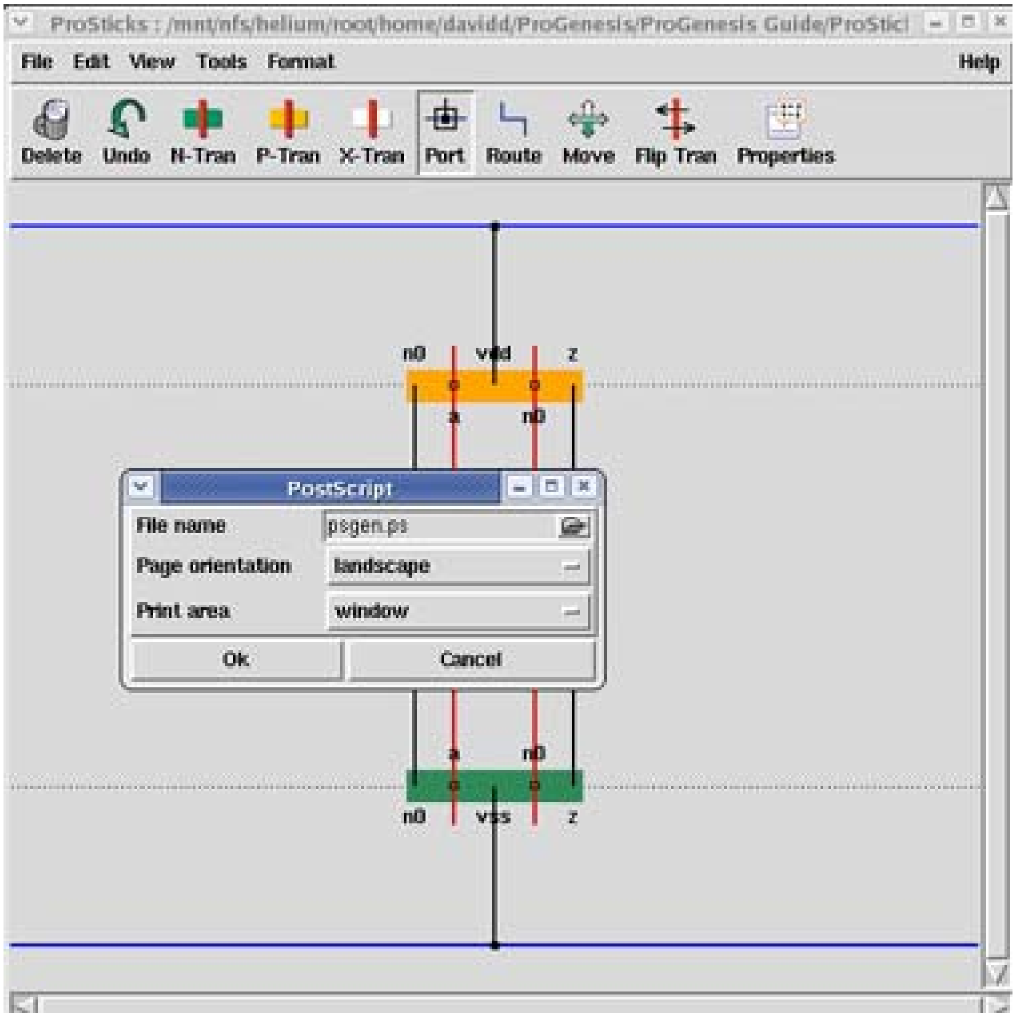
Proc name: buf_2x

Figure 9: Use descriptive names for your generator file and procedure name

When you've entered the values you want in the Create Generator window, click OK to continue.

Create PostScript...

The next File menu option is Create PostScript.... This option allows you to output a graphical version of your ProSticks creation for printing or viewing. When you select Create PostScript..., you will see a dialog box similar to the one shown in Figure 10.



File name: psgen.ps
Page orientation: landscape
Print area: window

Figure 10

Navigate through directories, enter a filename as usual, and click Save to finish.

Spice

The next option on the File menu is the Spice submenu. These options allow you to import an HSPICE netlist into ProSticks to create a generator. When the Spice menu is chosen, three suboptions may be selected:

- Load...
- Merge...

- Create...

Load...

You may click the Browse button to locate the filename you wish to load. You can also specify the transistor scaling value to increase or decrease transistor sizes from the netlist input values. Click OK when you are finished.

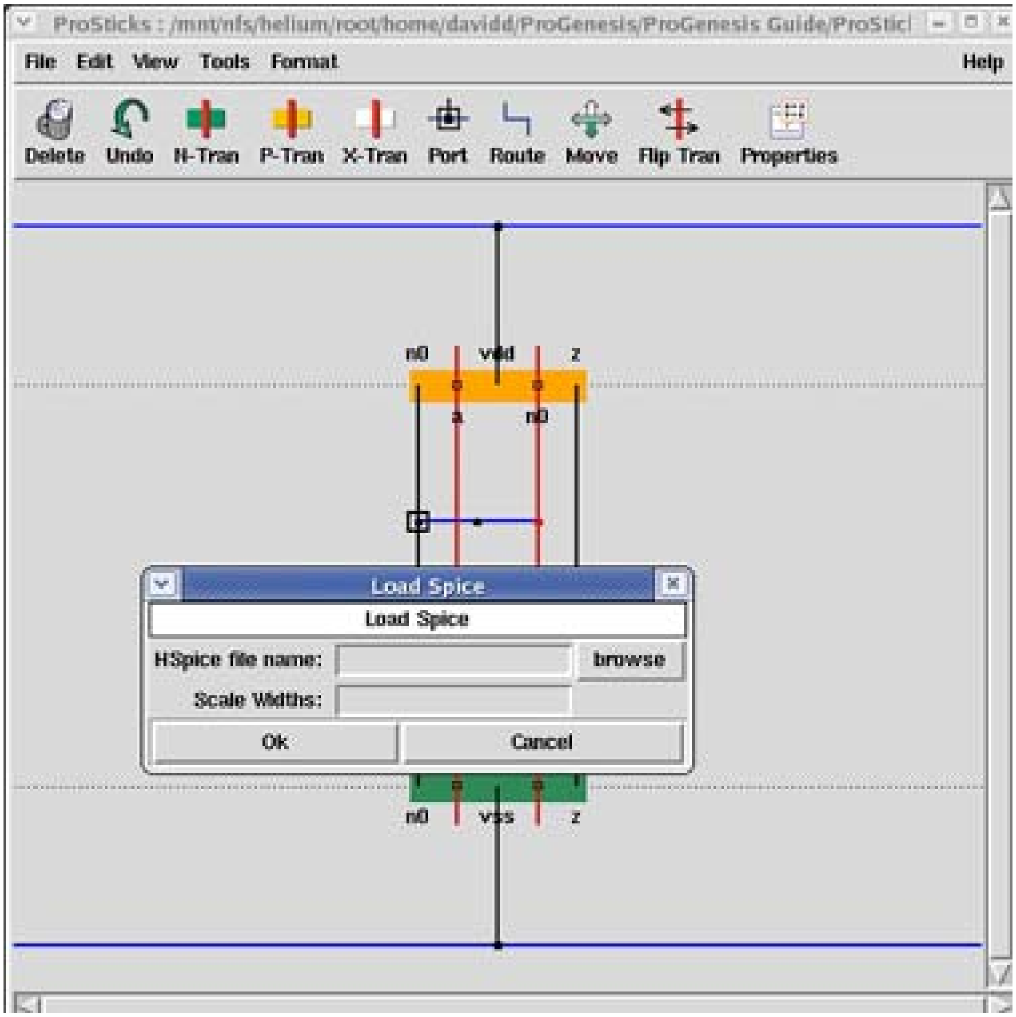


Figure 11

In Figure 12, an HSPICE file describing a buffer has been imported, ready to be laid out and routed.

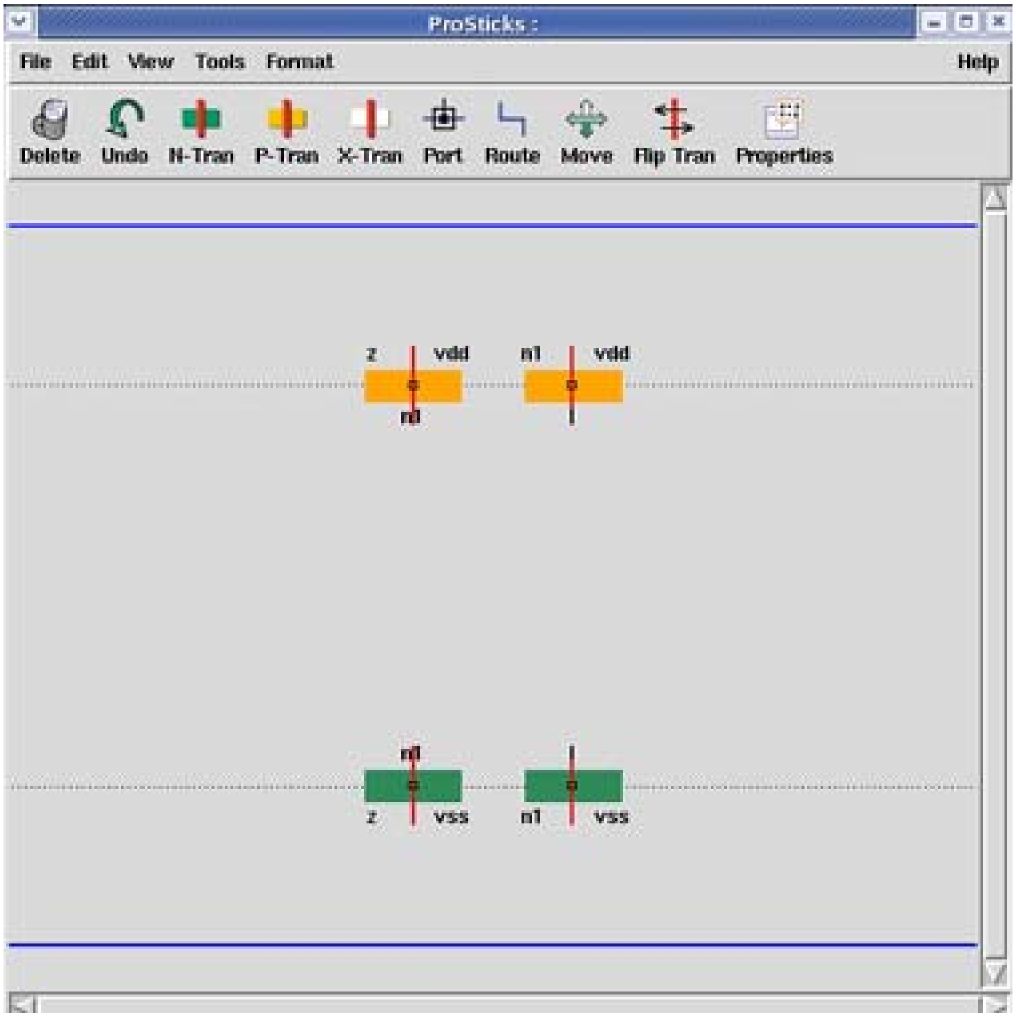


Figure 12: Importing an HSPICE file into ProSticks creates and labels the transistors

Merge...

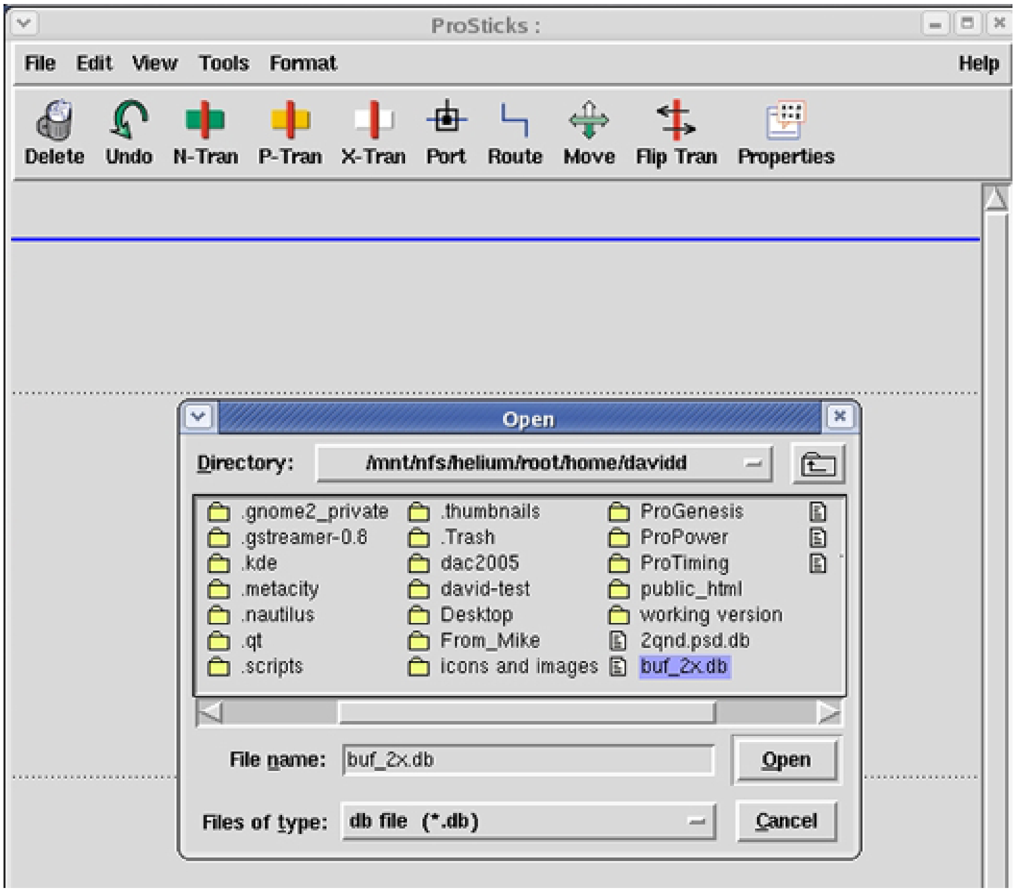
Merge adds the transitions from a netlist into the workspace.

Create...

Creates a new netlist

Load ProGen Interface File...

This option allows you to load a ProGen interface file (a .db file). This file is created by progen and is the ProSticks representation of progen's result.



Directory: /mnt/nfs/helium/root/home/davidd

File name: buf_2x.db

Files of type: db file (*.db)

Figure 13

Exit

The final option on the File menu is Exit, which simply allows you to close ProSticks. If you have modified the layout since the last time you saved it, ProSticks will ask if you want to save your work before exiting.

Edit Menu

The next menu is the Edit menu, which is illustrated in Figure 14.

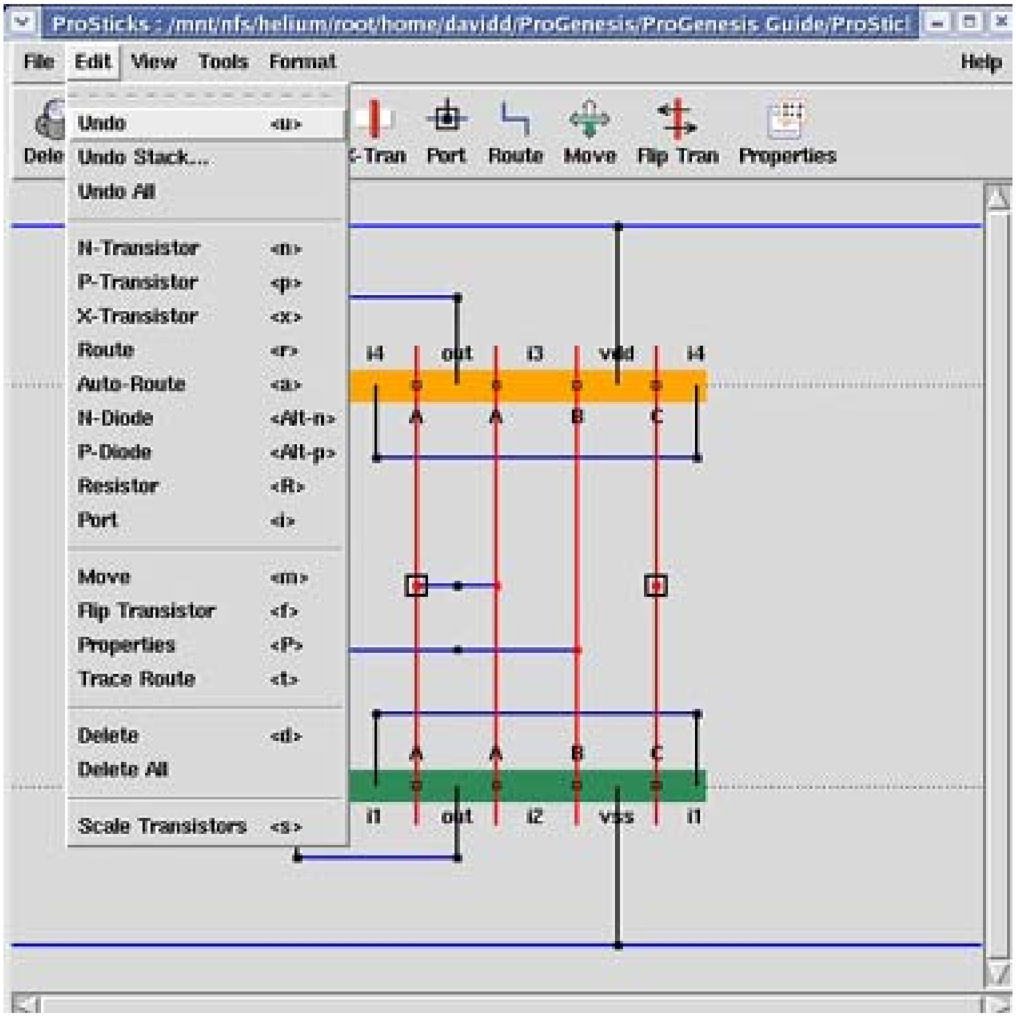


Figure 14: Use the tear-off Edit menu to lay out the generator

This is another tear-off menu, which means you can keep the window open and move it around the screen. This is convenient because the Edit menu is the one you'll use most as you lay out your cell generators. To tear off the menu, select the dashed line at the top of the menu. The menu will be placed in its own window, which you can move around the screen to wherever is convenient. Figure 15 shows the ProSticks desktop with the Edit menu window placed on top.

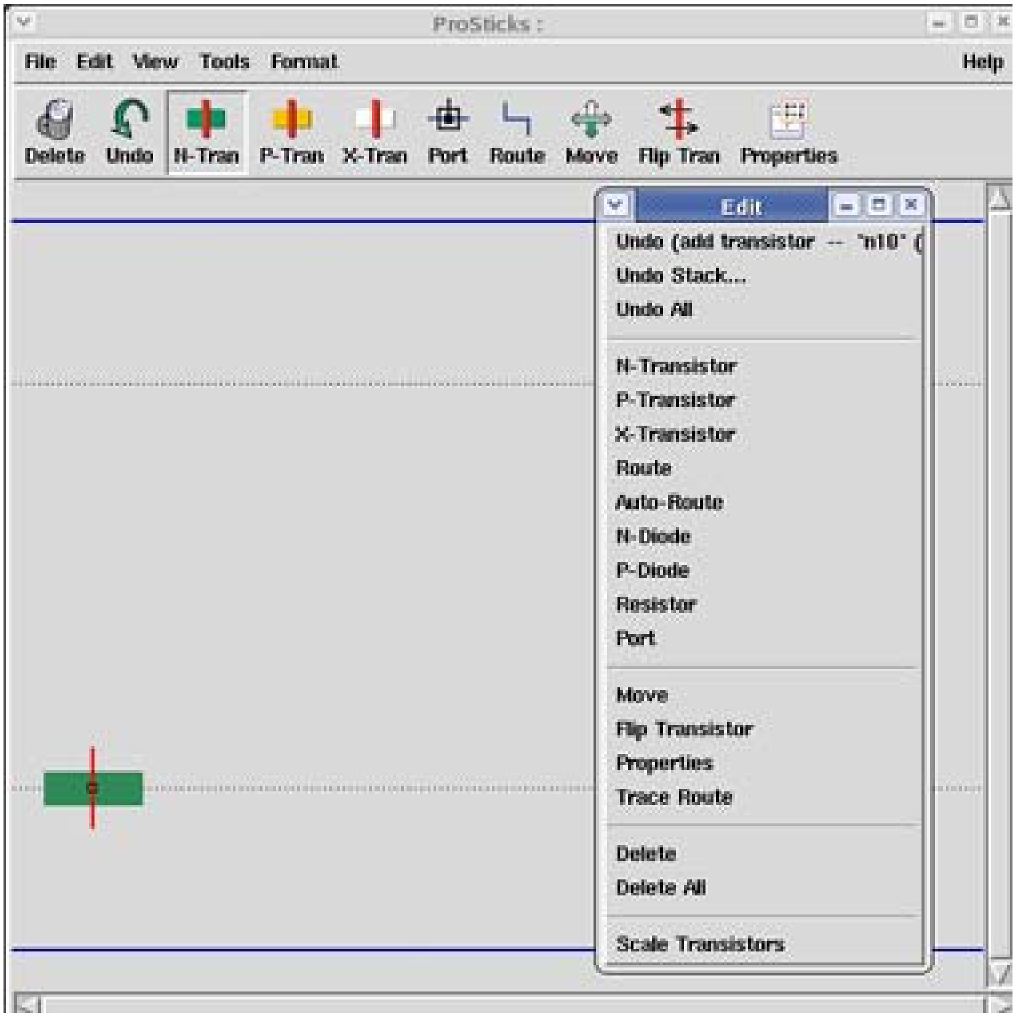


Figure 15: Tear off the Edit menu to place it in its own window

Undo

The first Edit menu option is Undo. Use this option to undo the last operation you performed. This is particularly handy when you mistakenly delete a component or decide you didn't like the results of the last move you made. The Undo function can be invoked multiple times to undo multiple actions; you can undo the operations back to the point where you created, opened, or saved the file -- whatever happened in succession most recently.

N-transistor, P-transistors, and X-transistors

The next three items on the menu allow you to place transistors in the cell. If you click on the N-transistor option, n-transistors appear as green-colored blocks when you click on the ProSticks workspace. Selecting P-transistor allows you to create p-transistors, which are colored orange. The X-transistor option allows you to create white pass-through transistors.

If you import a netlist, the transistors will appear automatically. If you place them using these Edit menu options, they will snap to the dotted lines in the ProSticks workspace. This means that if you attempt to drop a transistor somewhere other than a dotted line, ProSticks will pull it to the nearest dotted line. If you want more than two rows of transistors in the cell, see the description of the Add Transistor Row option from the Format menu later in this guide.

Route

The next two tools allow you to route the components of your cell. If you select the Route option, you can connect the elements you've placed in the cell. Route makes connections manually; after you select this option, click a gate, source, or drain and drag to another cell element; now they're routed. Figure 16 shows a route being dragged between two transistor ates.

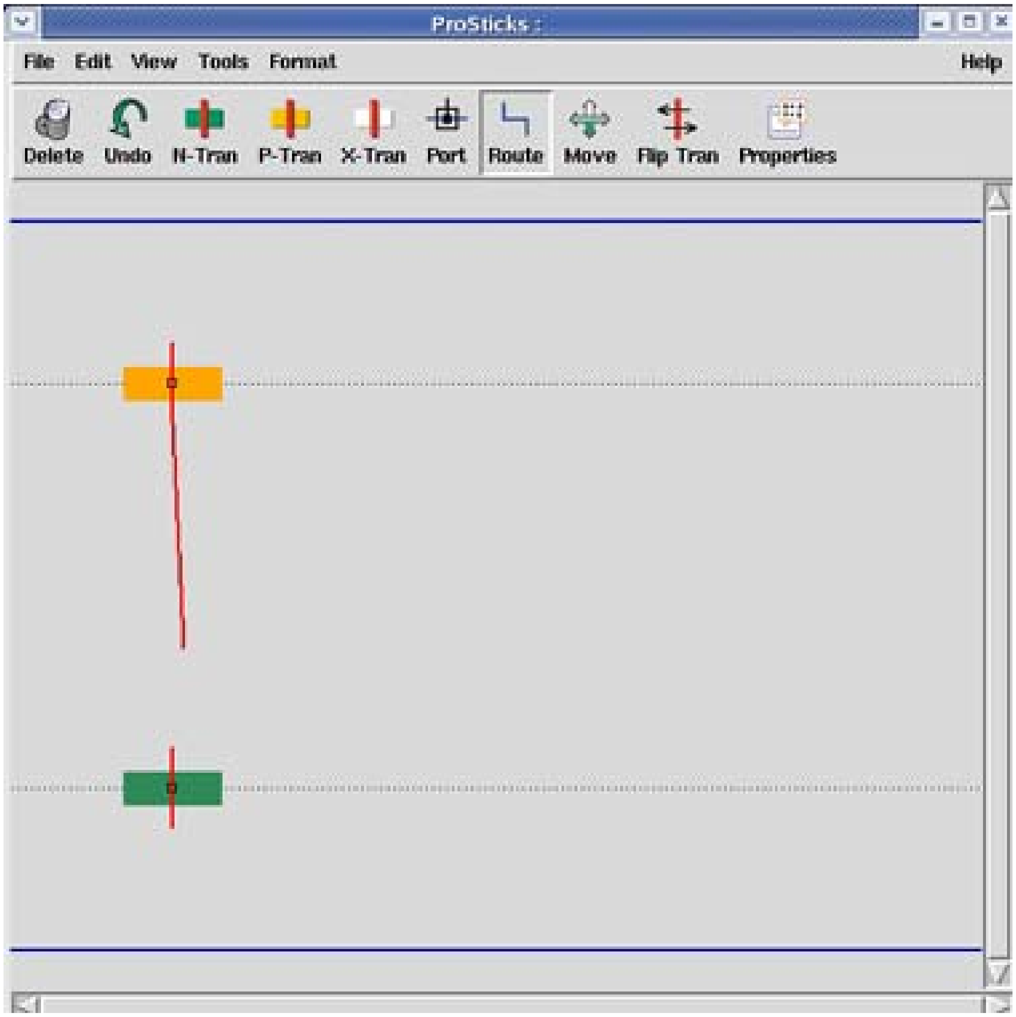


Figure 16: Drag to route between two cell elements

Once you have finished dragging the route, the two gates are connected as shown in Figure 17.

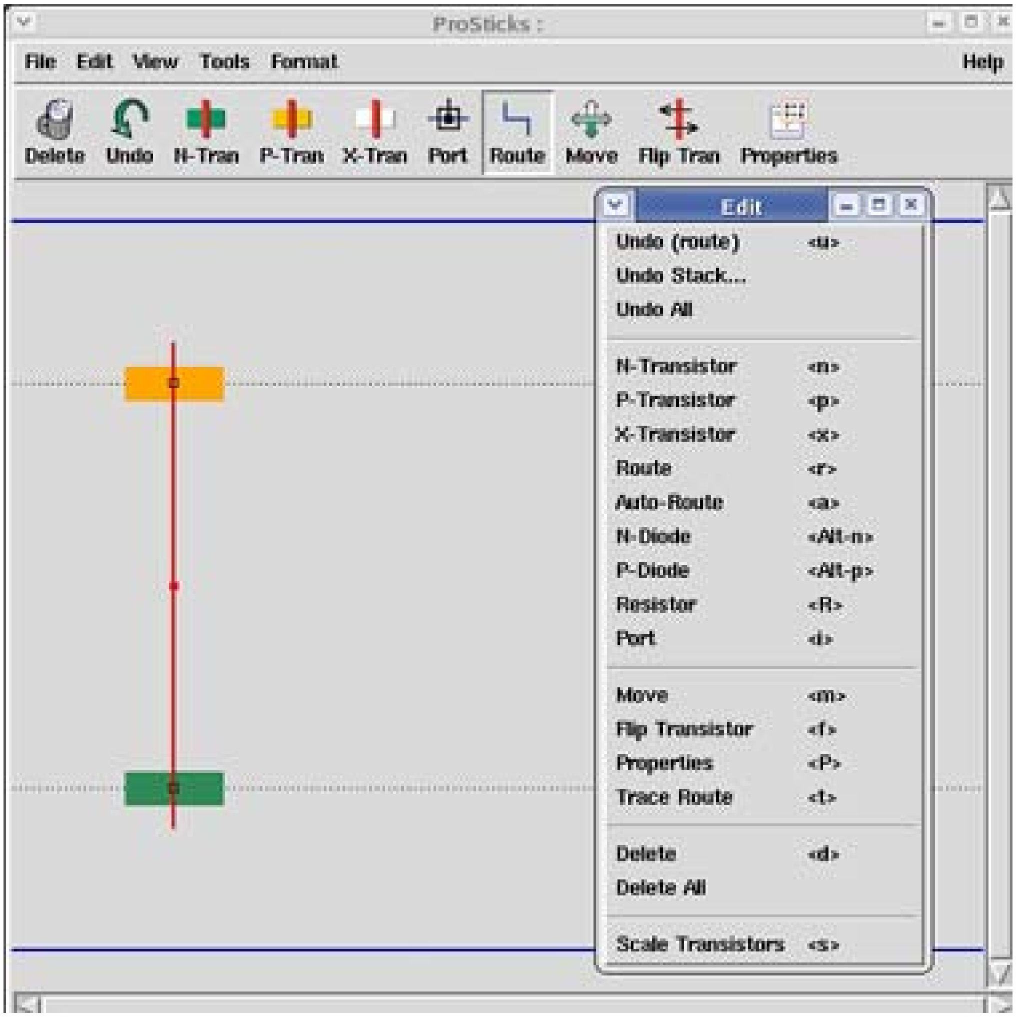


Figure 17: The transistor gates have been routed

Auto-Route

The next option allows you to make use of network names to automatically generate routes connecting all common names. In Figure 18, the inverter's two transistors have gates and right nets in common, so selecting Auto-Route and clicking on any of the common labels will connect the labels.

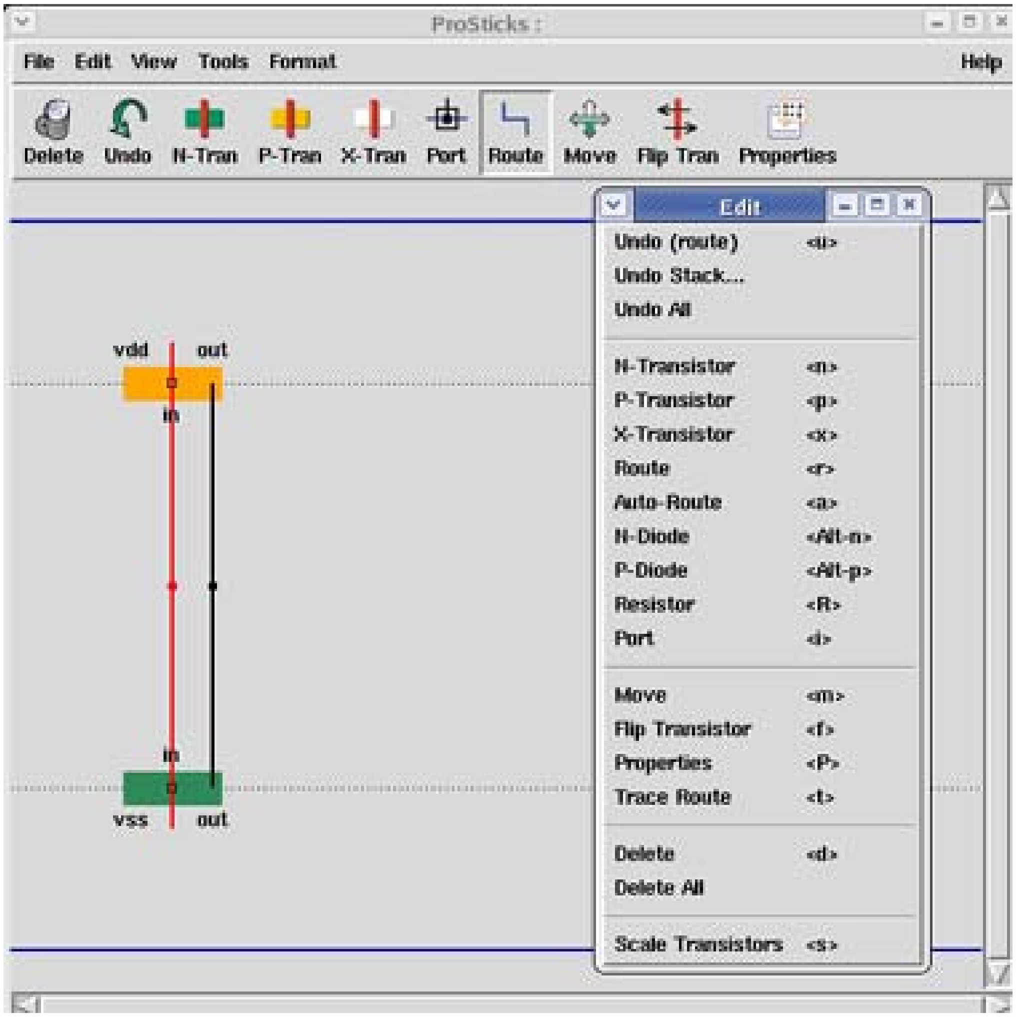


Figure 18: Use Auto-Route to connect all elements with the same net names

In Figure 18, the in and out connections were routed by clicking on Auto-Route and then clicking the in and out nodes.

N-diode and P-diode

The next two options allow the placement of n- and p-type diodes for technologies that require them. To place a diode, click on the appropriate diode option from the Edit menu, and then click on the trunk where you wish to place the diode. Figure 19 shows an n-diode placed in the p-well of an inverter.

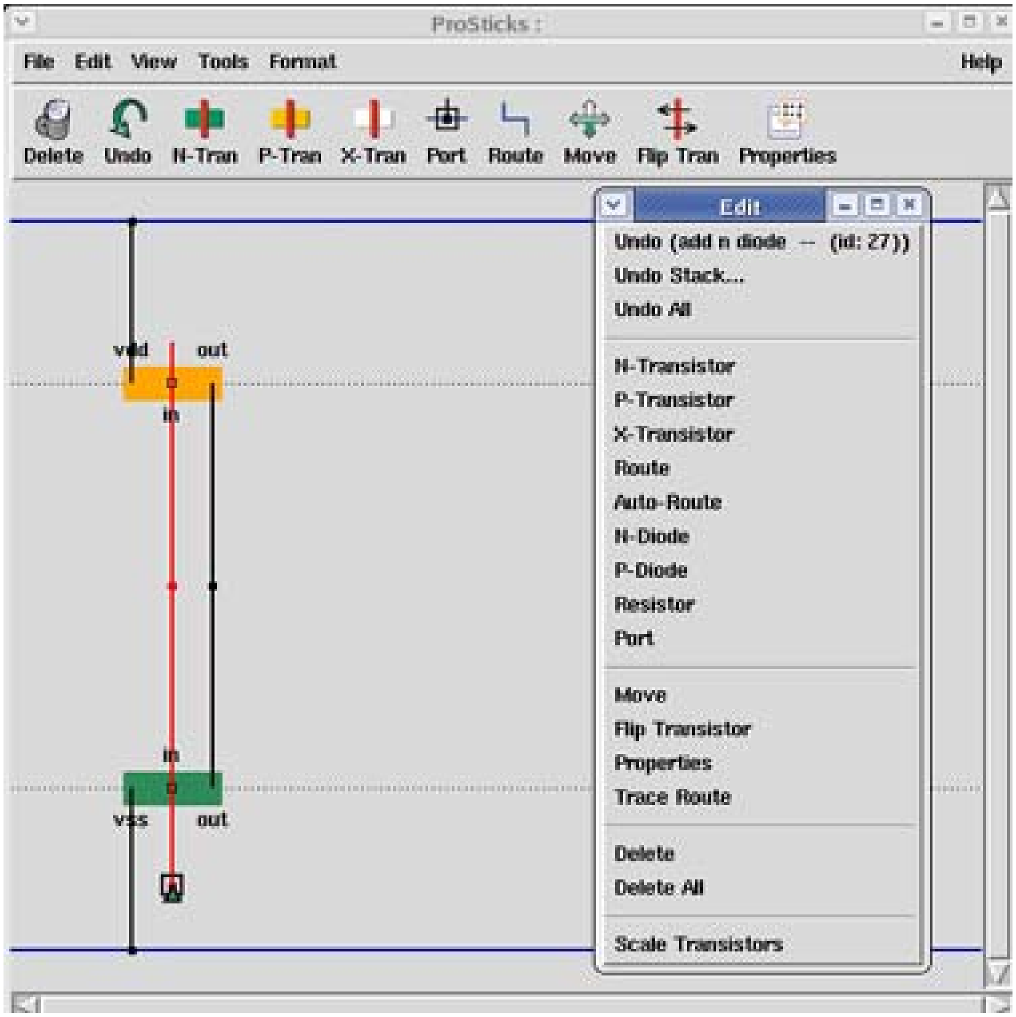


Figure 19: If diodes are required, place them on poly trunks

Note that the diode is connected to an extension of the poly. Because of the diode's placement requirements, you must use the Route function to add a trunk to the end of the poly on the transistor.

Move

This function allows you to select any element in the workspace and move it somewhere else. When you click on the Move option, you will notice a dotted line appear around elements on the screen as you pass the cursor over the workspace (as shown around the p-transistor in Figure 20).

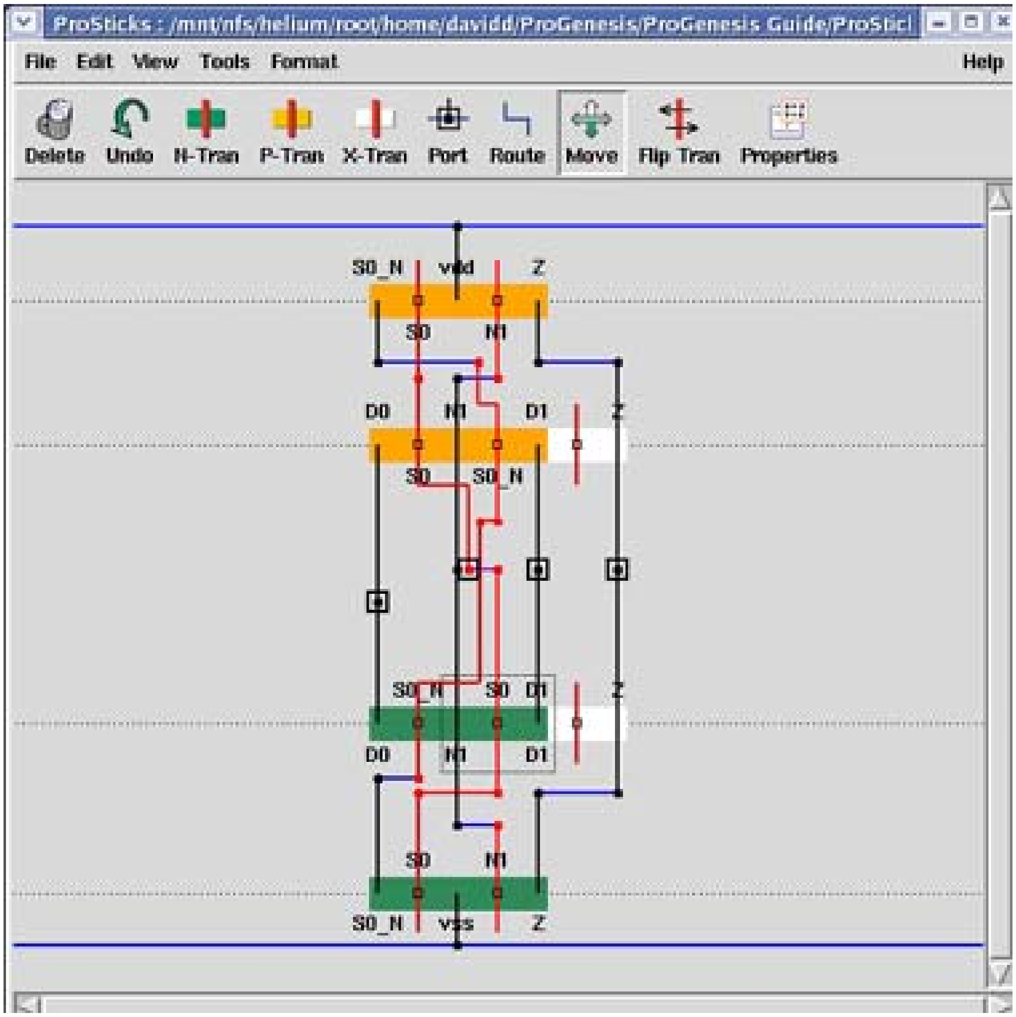


Figure 20: The dotted outline shows which element can be moved

To move a highlighted object, simply click it and drag it to the new location, as shown in Figure 21. You can use this technique to improve placement and routing in the cell to reduce cell size or optimize for materials.

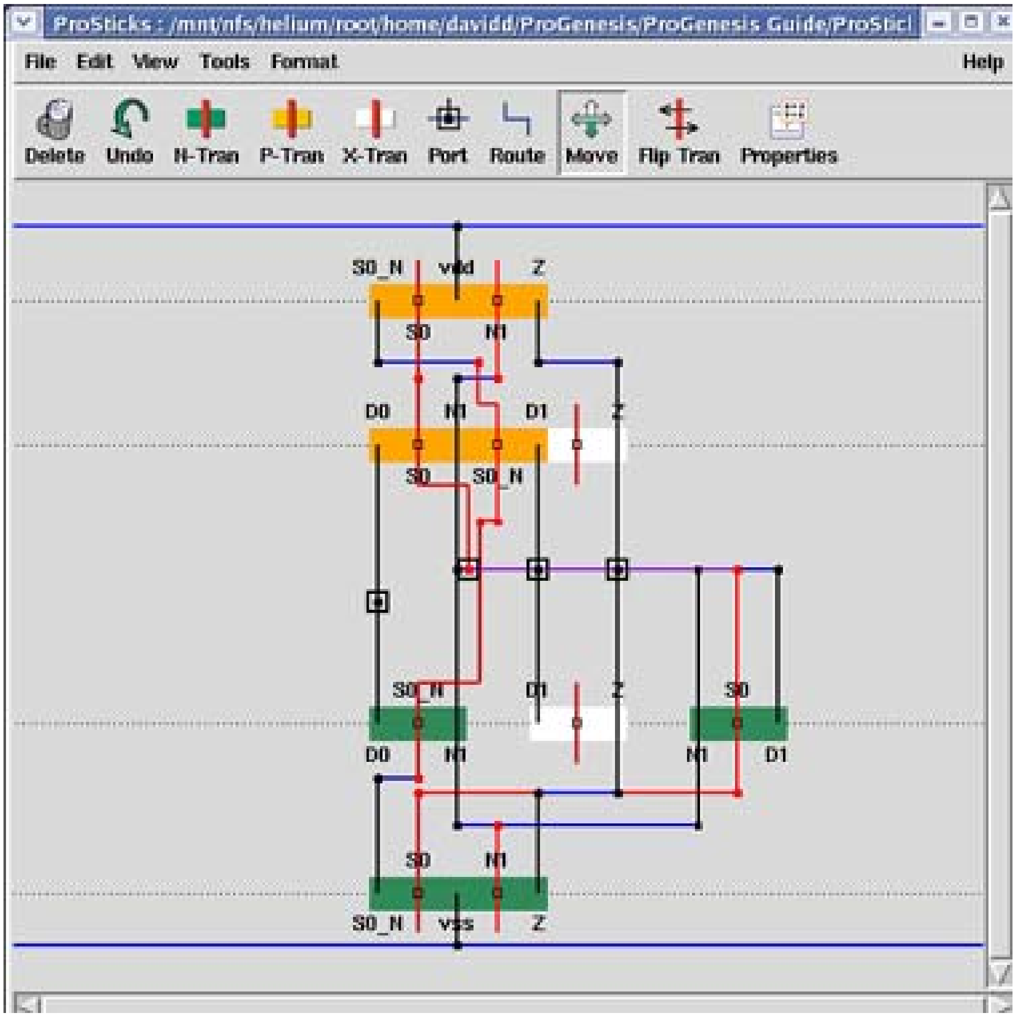


Figure 21: Simply click an object and drag it to use the Move option

Moving a single object often causes routes to cross, which means you'll have to make additional moves. Notice that in Figure 22, the poly from SO and the poly from Z (highlighted in yellow) now overlap because of a transistor move. (ProSticks won't highlight the overlap; we've retouched the screen capture to help you see the area in question.)

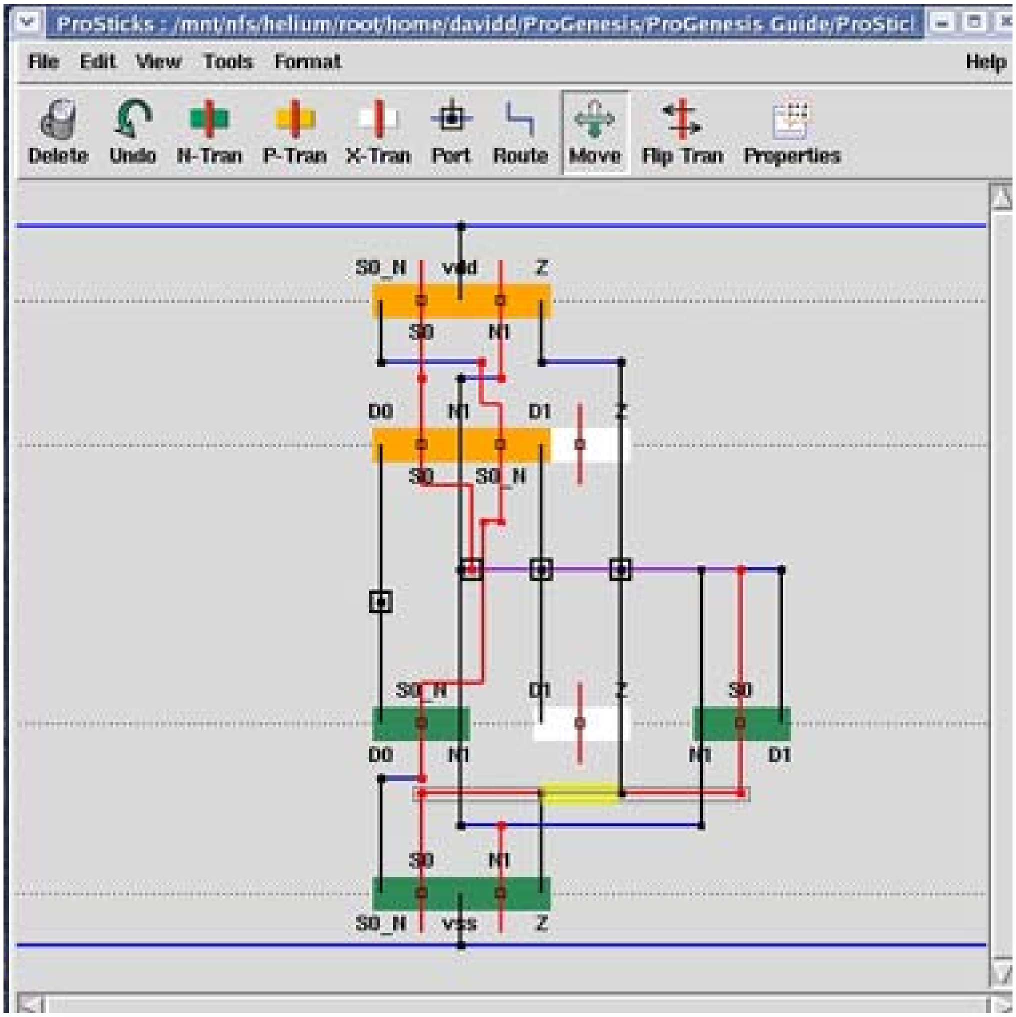


Figure 22: Moving one element often requires additional moves to avoid overlap

You might solve this problem by selecting Edit>Move, and clicking and dragging the poly S0 out of the way, as shown in Figure 23.

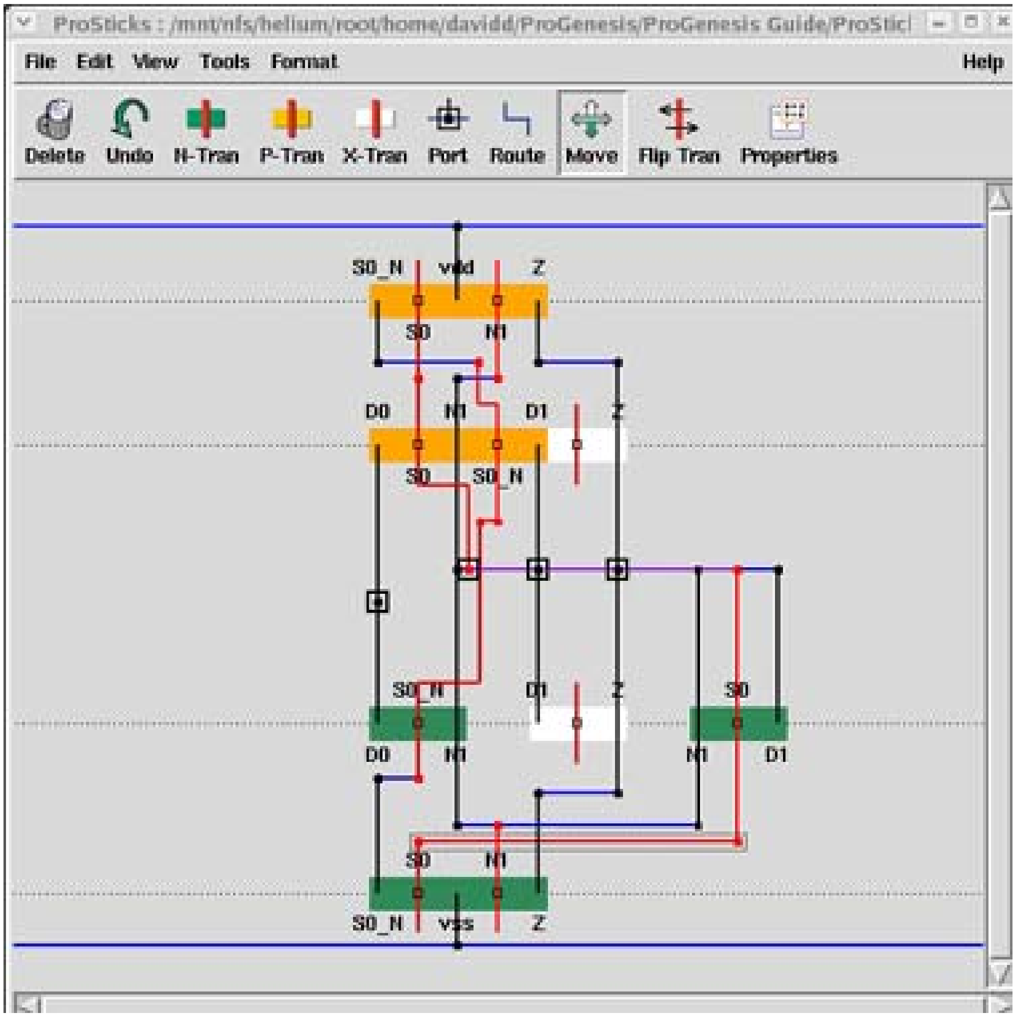
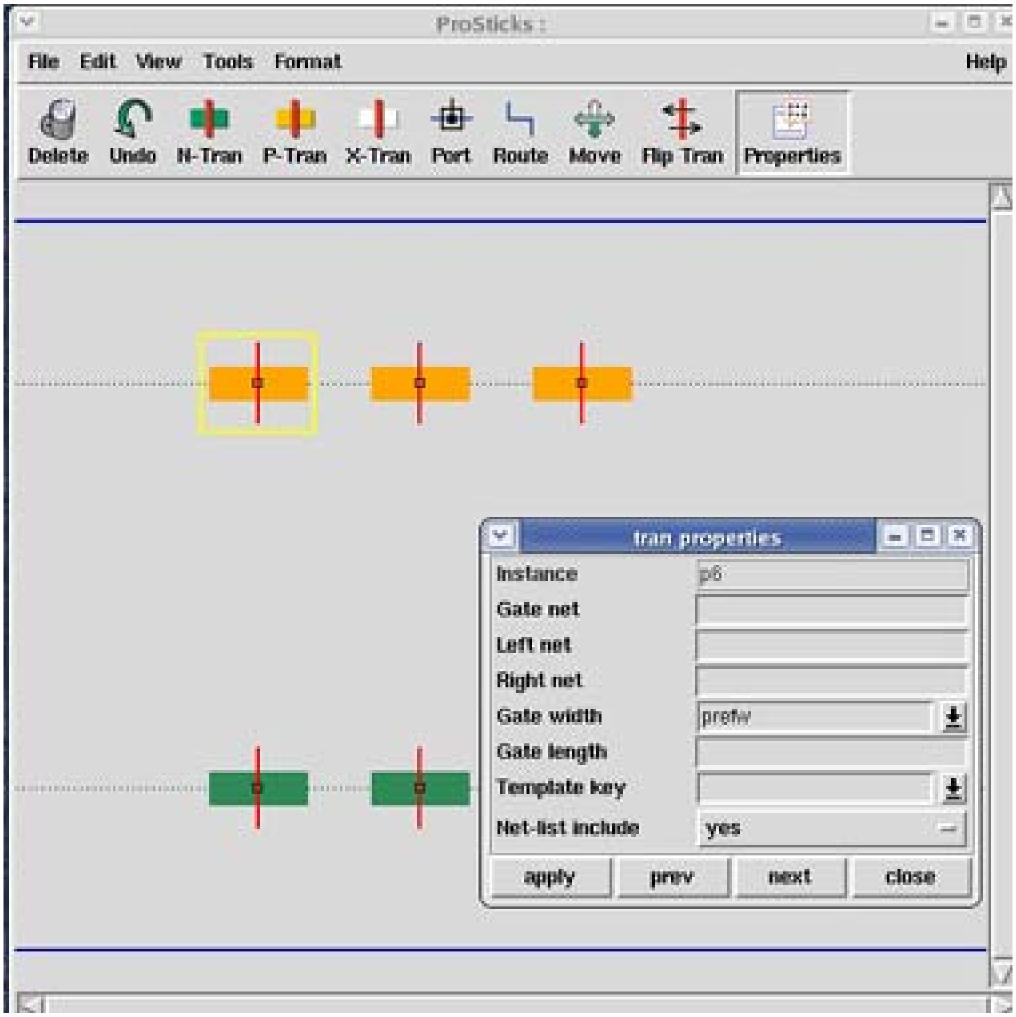


Figure 23: Jog a poly route to avoid the overlap caused by moving a transistor

Name Nodes

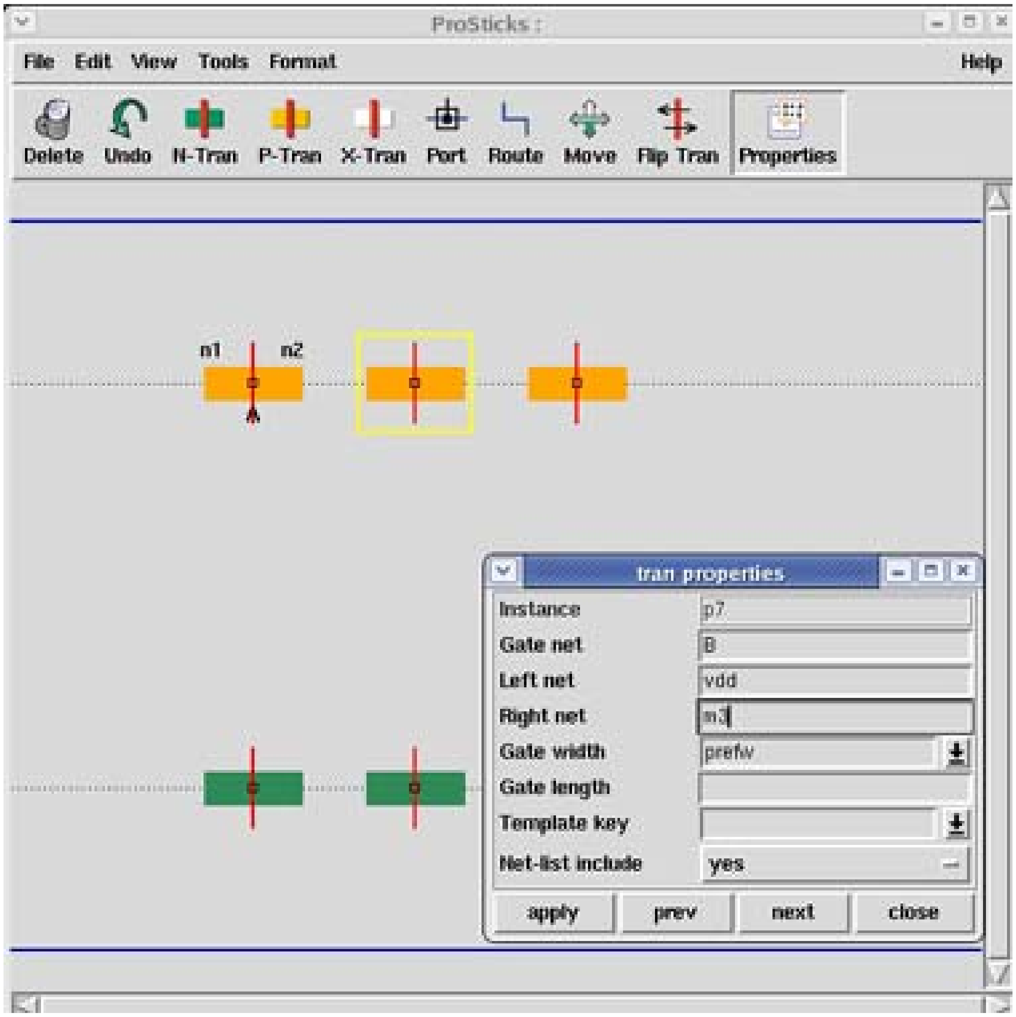
The next item on the menu is Name Nodes. This option allows you to cycle through the transistors in the workspace and provide names for the gate and diffusion contacts. Simply click on the Name Nodes option on the Properties menu and then click on an element on the workspace. Figure 24 shows a cell under development with the Node Names box ready for input.



Instance	p6
Gate net	
Left net	
Right net	
Gate width	prefw
Gate length	
Template key	
Net-list include	yes

Figure 24: Enter or modify transistor node names using the Name Nodes function

Notice the yellow line around the transistor whose nodes are being named. Enter names in the gate, left, and right fields, and then click the Apply button to attach those node names to the corresponding parts of the highlighted transistor. To move on to the next transistor, click the Next button. Figure 25 shows the cell after the first transistor node names have been entered, the Apply and Next buttons have been clicked, and the second transistor node names have been entered. The Apply button has not yet been clicked again.



Instance	p7
Gate net	B
Left net	vdd
Right net	m3
Gate width	prefw
Gate length	
Template key	
Net-list include	yes

Figure 25: Continue entering node names for each of the transistors in your cell

In the layout shown in Figure 25, if you click the Apply button, the node names that have been entered will appear on the outlined transistor. In addition, the routed nets B and m3 will have their node names propagated to all the other nodes on those nets. If you subsequently route a node you've named, the name will be propagated where you route it.

Click the Next button to move on to the next transistor, or click the Prev button to move to the previous transistor. You can continue entering node names until all transistors have been fully named. Then click the Close button to return to the workspace.

Flip Transistors

The next option is to Flip Transistors. This is useful when you wish to align the connections between two transistors. In Figure 26, the transistor selected on the bottom right of the workspace has the ground connection on its right side; if this transistor were flipped over, its diffusion could be joined with the transistor next to it.

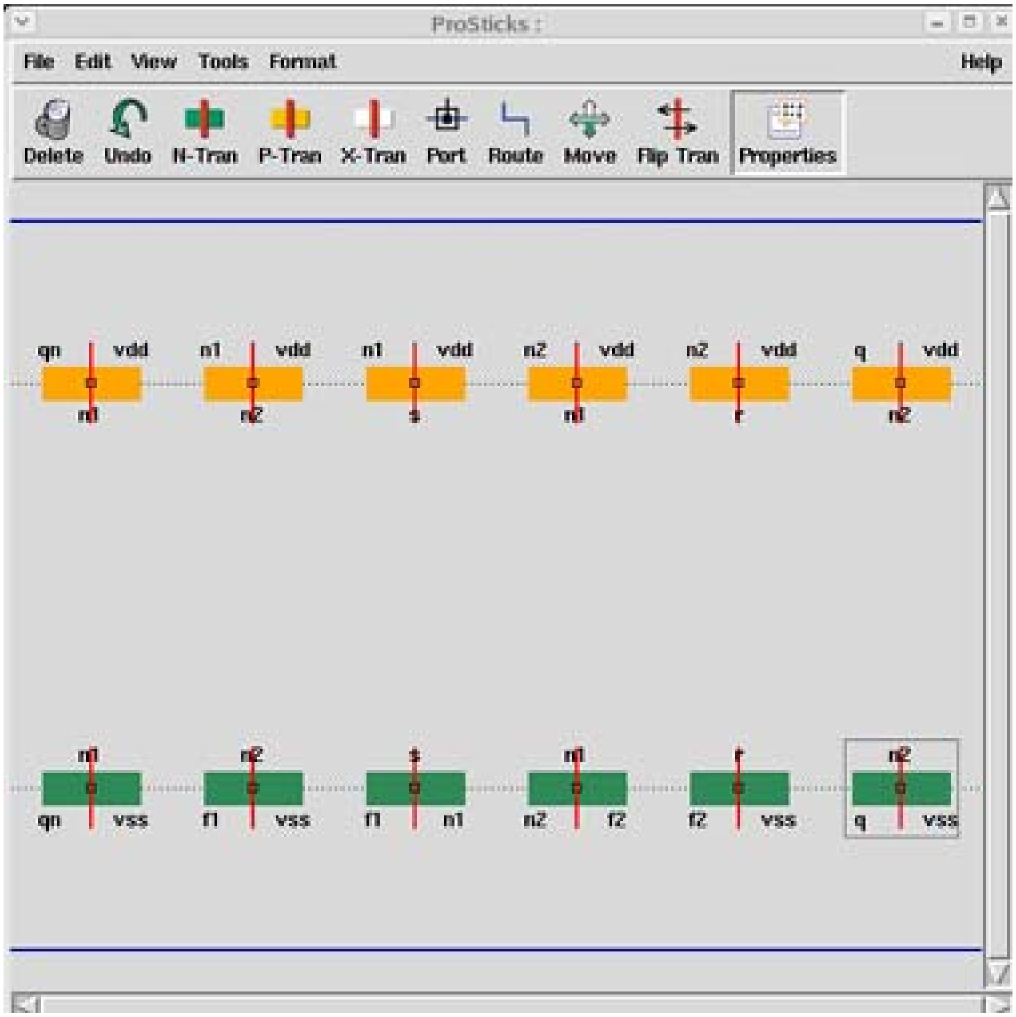


Figure 26: Select a transistor to flip to share diffusion

By selecting Flip Transistors from the Edit menu and clicking on the transistor, you can reorient the transistor, as illustrated in Figure 27.

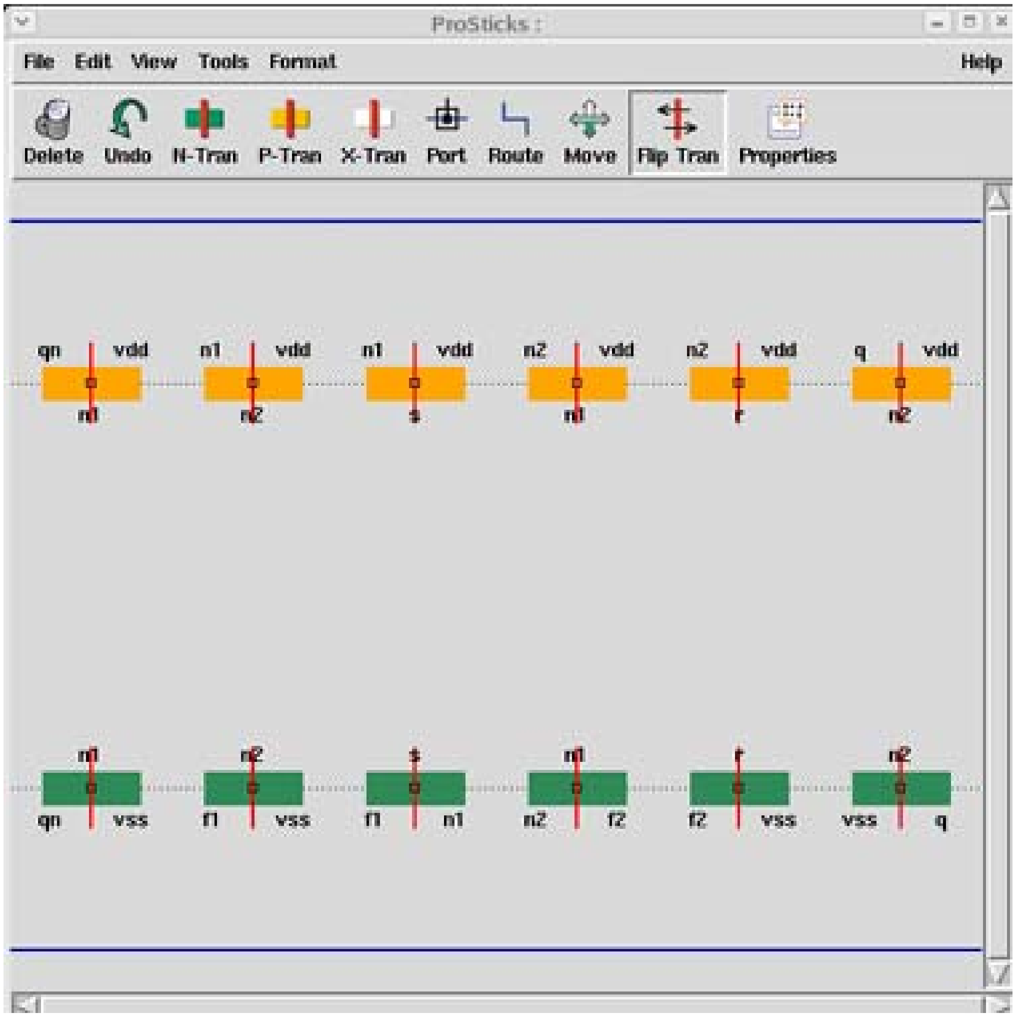


Figure 27: The flipped transistor is now oriented to attach to its neighbor

Transistor flipping can be performed automatically when you use the Move command. When you move one transistor next to another, the transistor will be flipped, if needed, to ensure that the diffusion contact is shared. Manual flipping is also useful for aligning left and right feeds of n- and p-transistors to minimize jogs.

Auto Flip

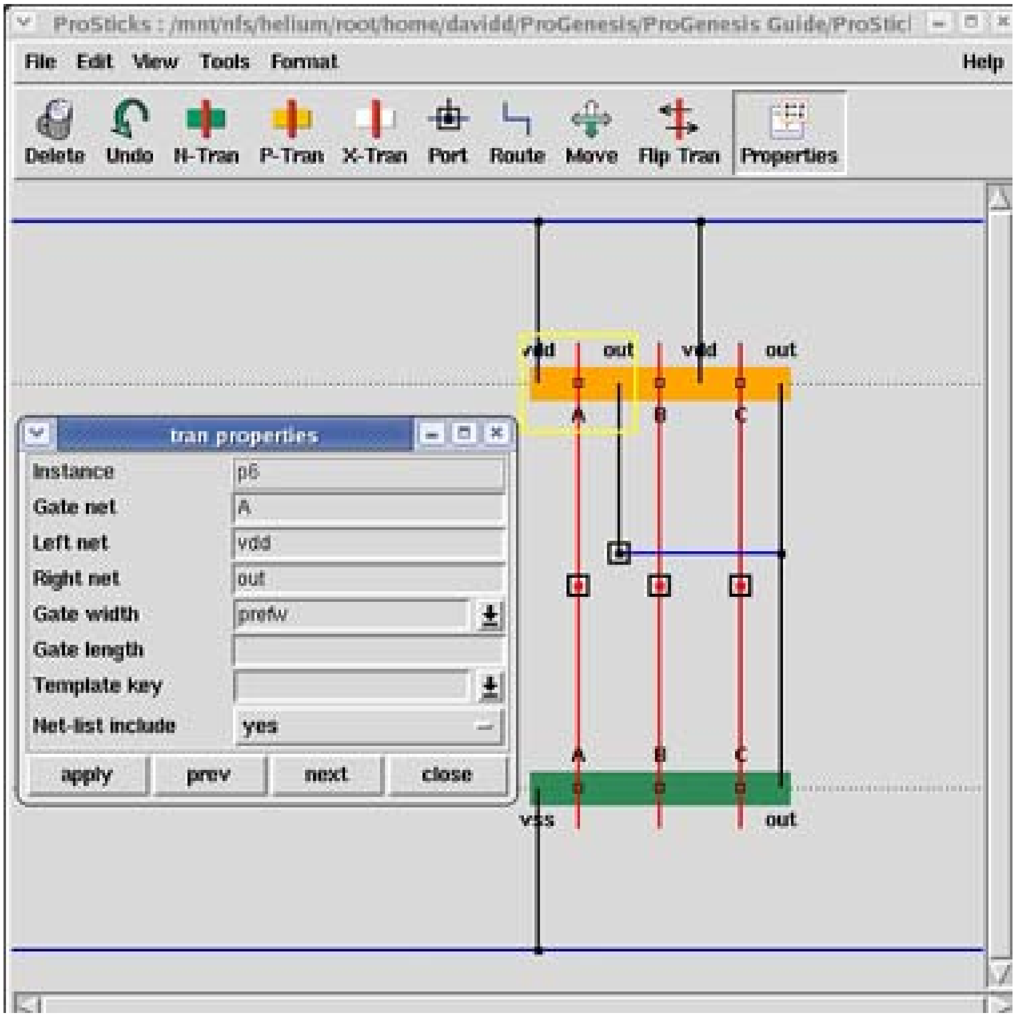
When you move a transistor next to another transistor with a common diffusion contact net name, ProSticks assumes you want to join the transistors together to share diffusion. It will automatically flip the transistor you're moving if that makes it possible to merge the common contacts.

Trunk Split

Trunk split splits a horizontal trunk at the point selected by the user. A vertical feed connects the two trunk pieces. Both pieces of the split trunk get the property values of the original trunk. Feeds, jogs, diodes, and ports to the left of the split point are placed on the new left trunk and those to the right go to the new right trunk.

Properties

The next Edit menu option allows you to see the properties of a cell element. Unlike Name Nodes, which only works on transistors, properties can also be entered for poly or metal trunks. Simply click the Properties entry on the Edit menu and then click an element. Figure 28 shows the properties entries for a transistor.



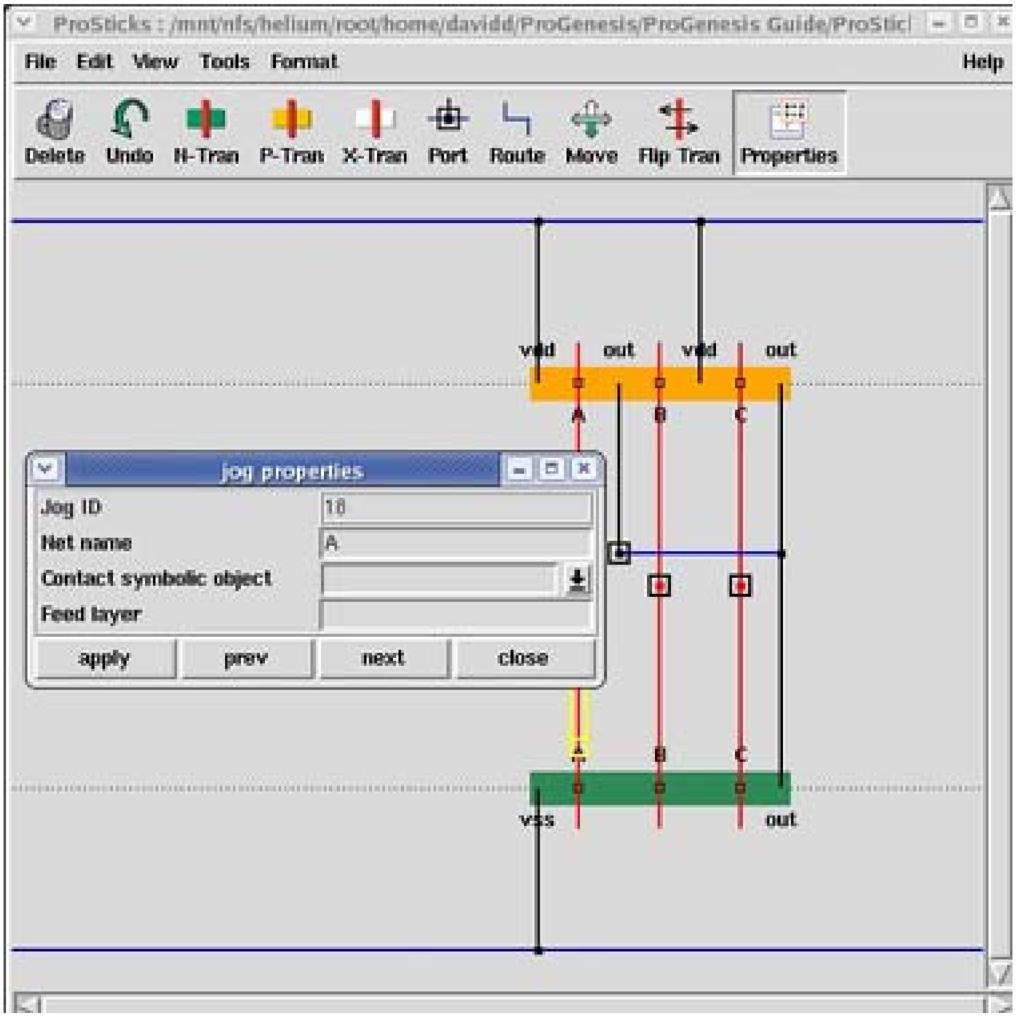
Instance	p6
Gate net	A
Left net	vdd
Right net	out
Gate width	prefw
Gate length	
Template key	
Net-list include	yes

Figure 28: Transistor options can be displayed using the Properties command

The Instance field indicates the unique name of the element you've selected. This name identifies the component for ProGen, which builds the cell. The instance names are assigned in the order the elements are placed on the workspace.

In Figure 28, the highlighted transistor's name is p6. The next three properties are the same ones shown by Name Nodes. The gate, left net, and right net fields designate names for the gate and the left and right transistor poly contacts. The last field indicates the width of the element; width defaults to prefw, which is an arbitrary default width. If you import a netlist, each transistor will have a value defined already; if you manually create the transistors, you can enter the width here.

Figure 28 illustrates the properties available for a transistor. Other ProSticks elements have different available properties. In Figure 29, a poly route has been selected and its properties are being displayed.



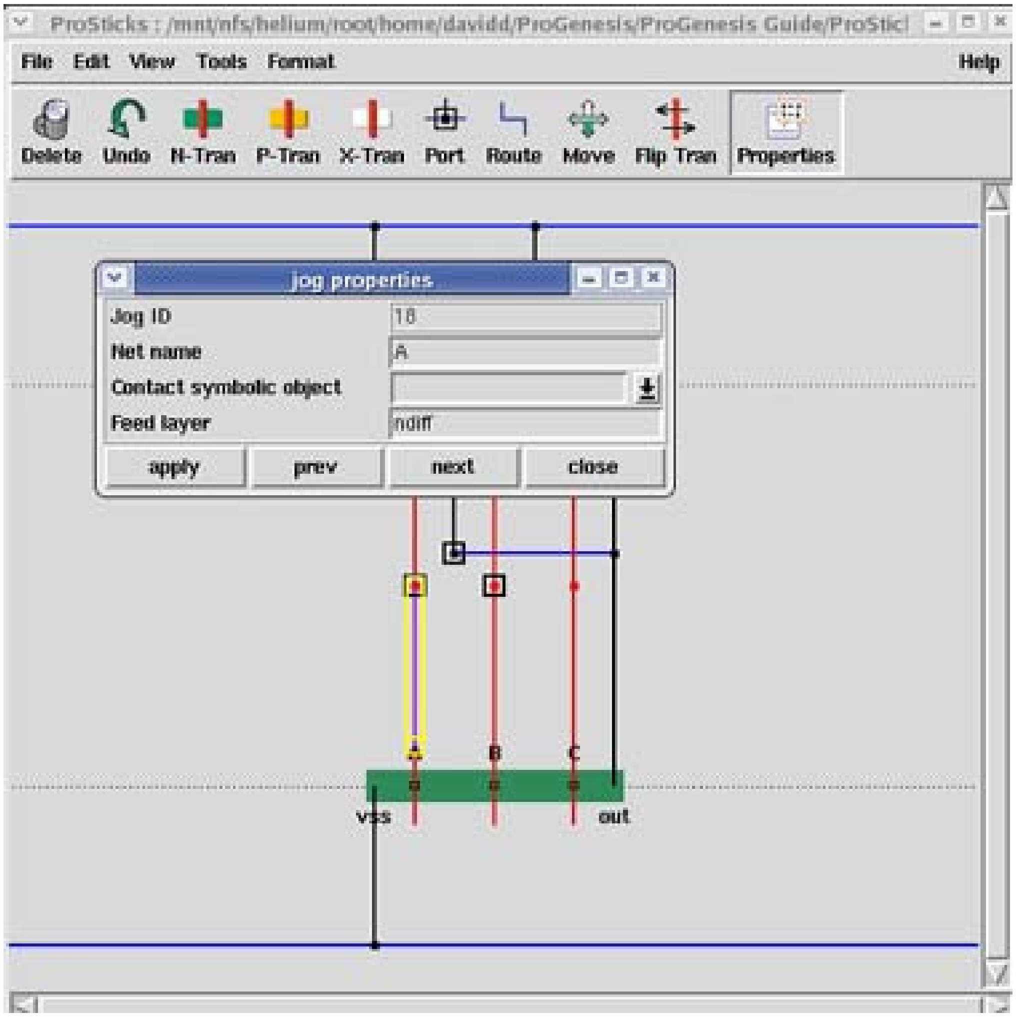
ProSticks : /mnt/nfs/helium/root/home/davidd/ProGenesis/ProGenesis Guide/ProStic

jog properties

Jog ID	18
Net name	A
Contact symbolic object	
Feed layer	

Figure 29: A different set of properties is shown for a poly route

In Figure 29, the jog has a single net name. The other options indicate the object used to define the contact and the layer that this jog will use for routing.

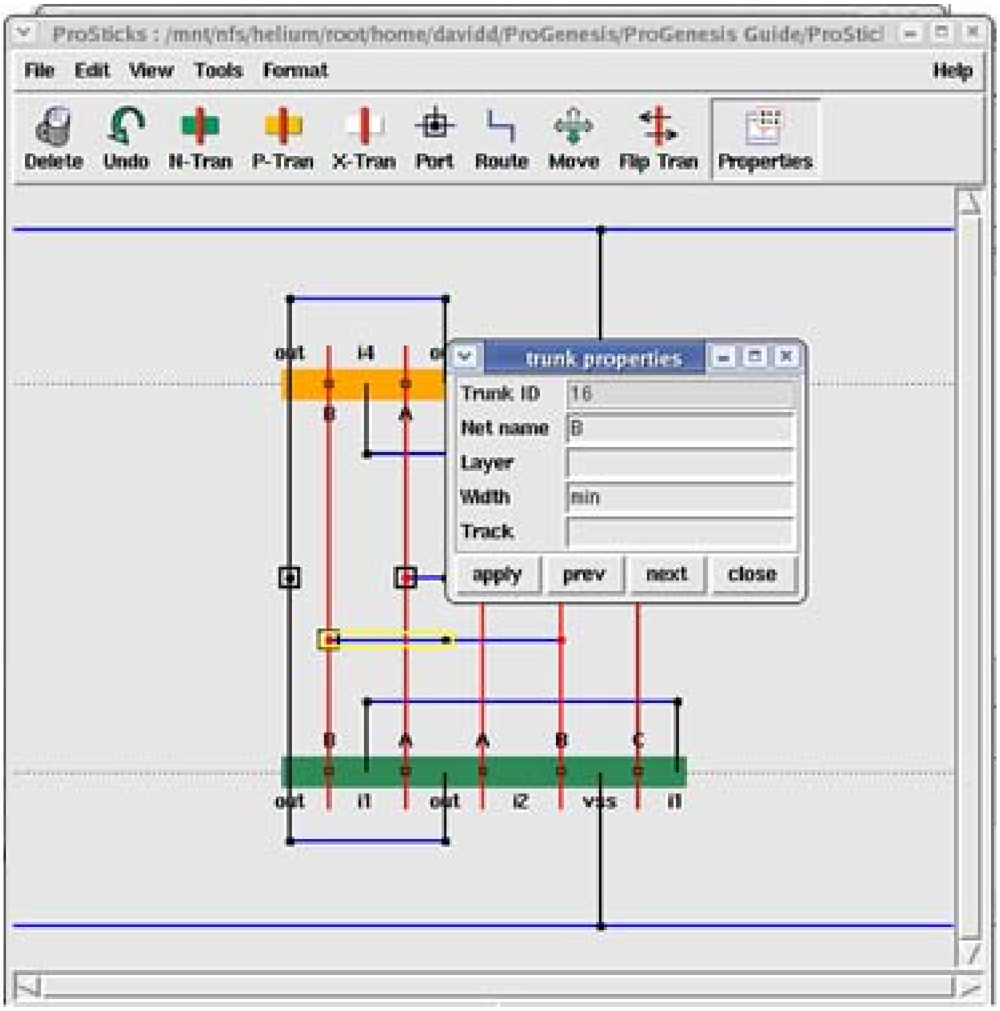


Jog ID 18
Net name A
Contact symbolic object
Feed layer ndiff

Figure 30: Move a feed from metal to diffusion using the Edit Properties function

Notice that the net name is still defined by the transistor to which the contact the wire is routed. You can set the layer definition for this wire from the feed layer dropdown list. In Figure 30, the jog properties indicate that this is a diffusion wire. That doesn't make much sense in this case, but if you want to route in diffusion to avoid using extra metal layers, you can use the Properties option to force diffusion routing.

In Figure 31, a horizontal trunk has been selected, and it has another set of available options.



Trunk ID: 16
Net name: B
Layer:
Width: min
Track:

Figure 31: Trunk settings can be viewed using the Edit Properties function

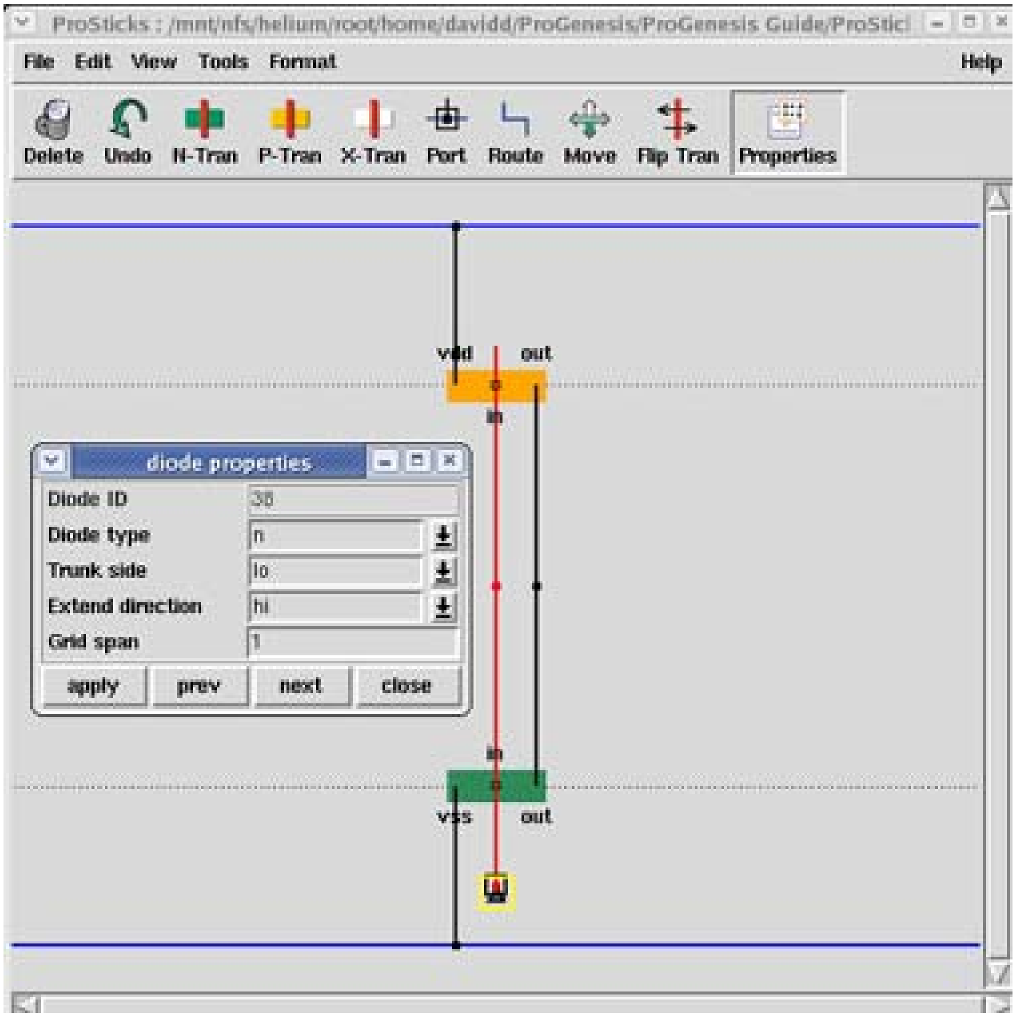
The trunk properties include the net name, which is determined by the feeds the trunk connects. The Trunk Layer field is usually used to force a trunk from metal 1 to poly, but you can change the layer to metal 3 or whatever you need it to be.

Notice that the Port check box is filled. This ensures that a port object is placed somewhere on the trunk; if there are multiple possible locations for the trunk, ProSticks will select one that will be optimized when ProGen runs.

If you select a port, the properties shown are for the trunk on which the port resides.

The Trunk Properties window in Figure 32 is the same as the one in Figure 31 because selecting the port is really the same thing as selecting the whole trunk. In this example, the port on net B has been selected (the yellow highlight is visible inside the black square representing the port). If you wished to remove this port to place one somewhere else, you could unmark the Port check box.

You can also view properties for a diode, as shown in Figure 33.



Diode ID 38
Diode type n
Trunk side lo
Extend direction hi
Grid span 1

Figure 33: Modify diodes using the Edit Properties function

The properties listed for a diode include the type (n-diode or p-diode), the orientation of the diode, and the spacing between the diffusion and the diode's gate contact. In Figure 33, note that the trunk side is set to lo and the extend direction is set to hi, making use of the free space on the hi side of the diode. In Figure 33, the standard grid span value of 1 is fine, but in cases where you wish to control the location of the diffusion (for example, to take advantage of space between transistors), you can alter this value as necessary.

Delete

The final two options on the Edit menu allow you to delete objects from the workspace. The Delete command allows you to select individual objects for deletion. In Figure 34, Delete has been chosen and the n-diode is being selected.

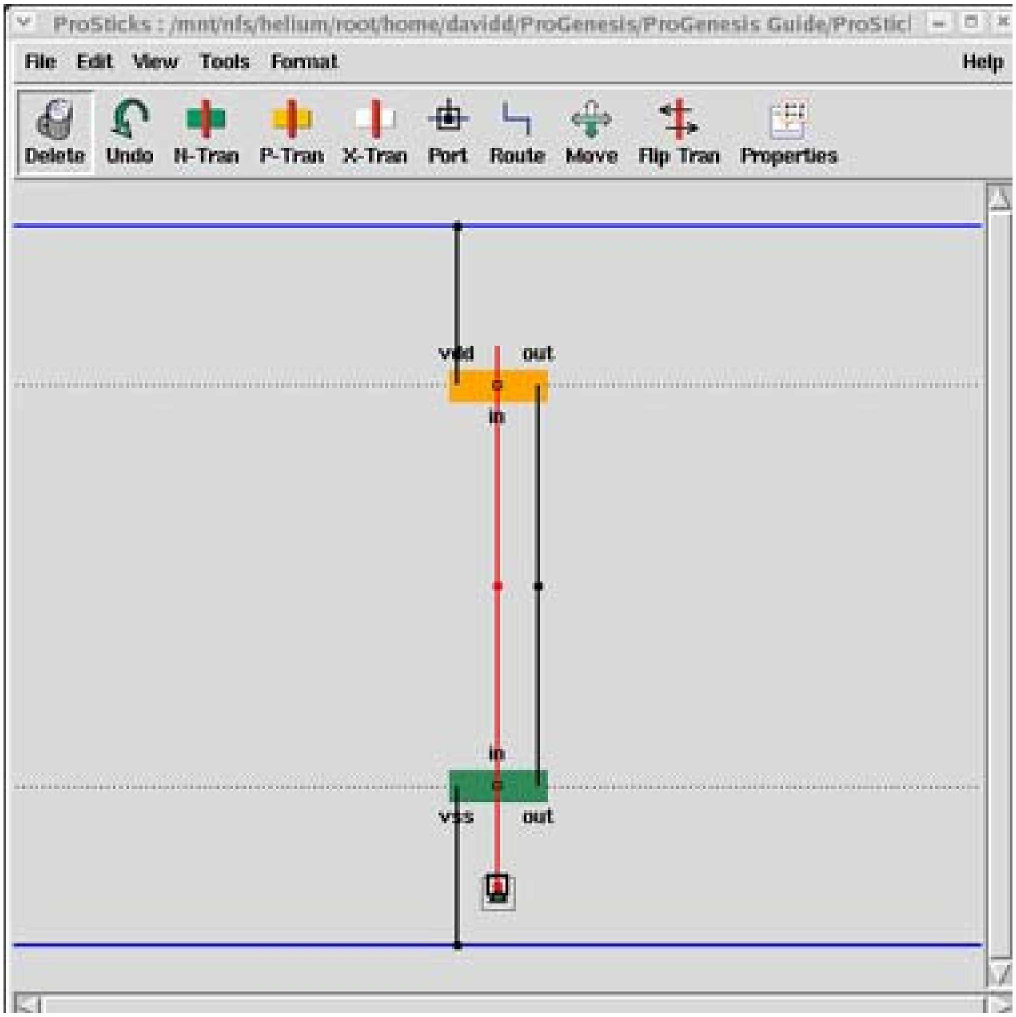


Figure 34: Select the diode to be deleted

Once you click on the diode, it is deleted as shown in Figure 35.

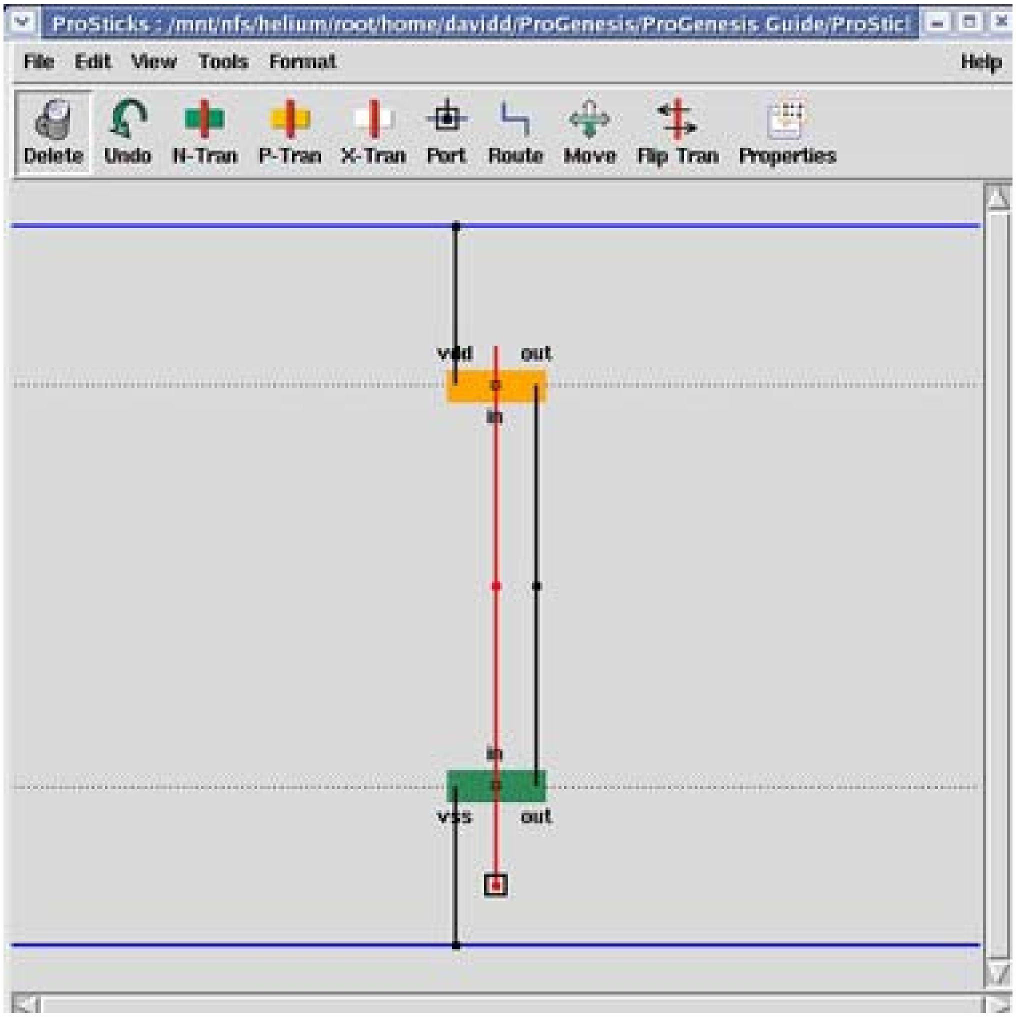


Figure 35: Use the Delete tool to remove the diode

If you wish to undo the last deletion, you can select the Undo command from the Edit menu.

Deleting one item often implies deletion of another, and ProSticks will delete any objects that are dependent upon an item you're deleting. For example, the same inverter cell with a diode is illustrated in Figure 36, but now the poly feed from the transistor gate to the diode has been selected.

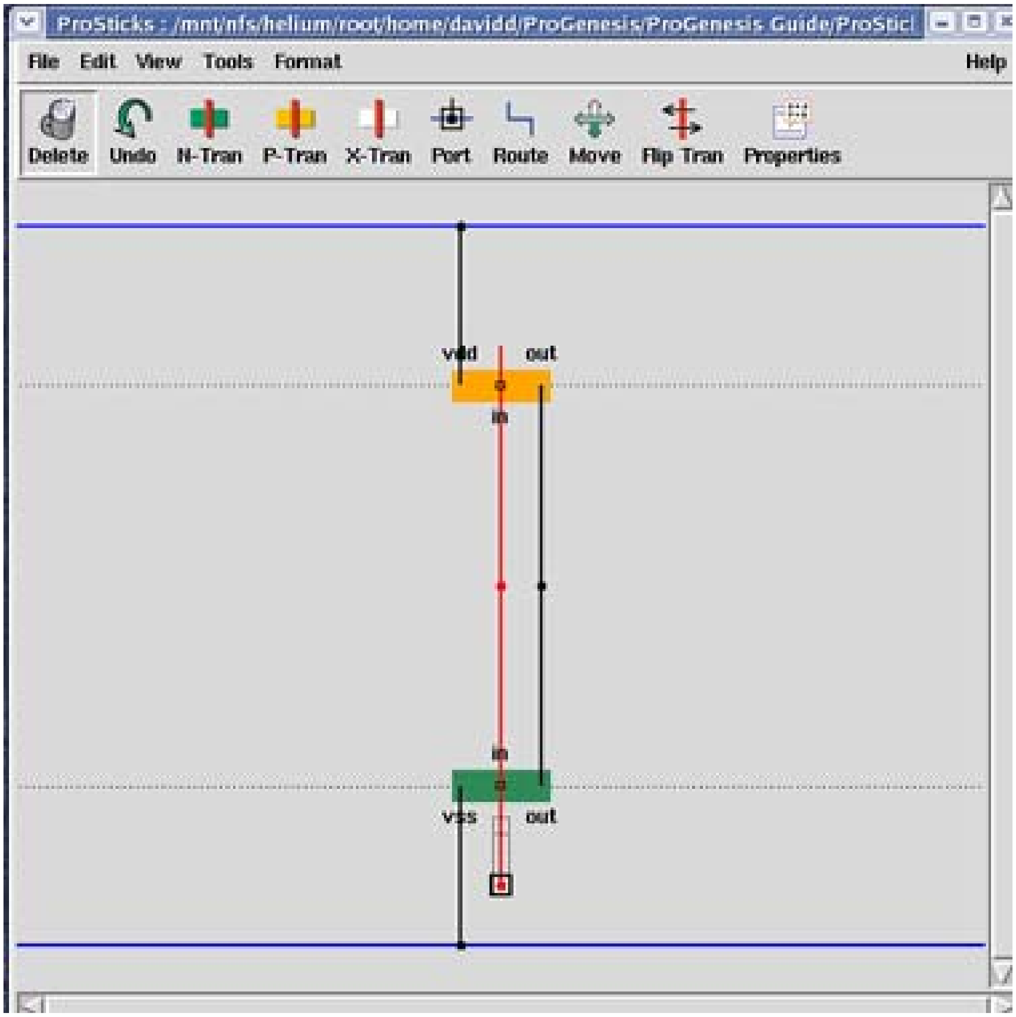


Figure 36: Select the poly feed to delete it

If this poly feed is deleted, as shown in Figure 37, the diode is also deleted. Similarly, if you select half a feed, the whole feed will be deleted.

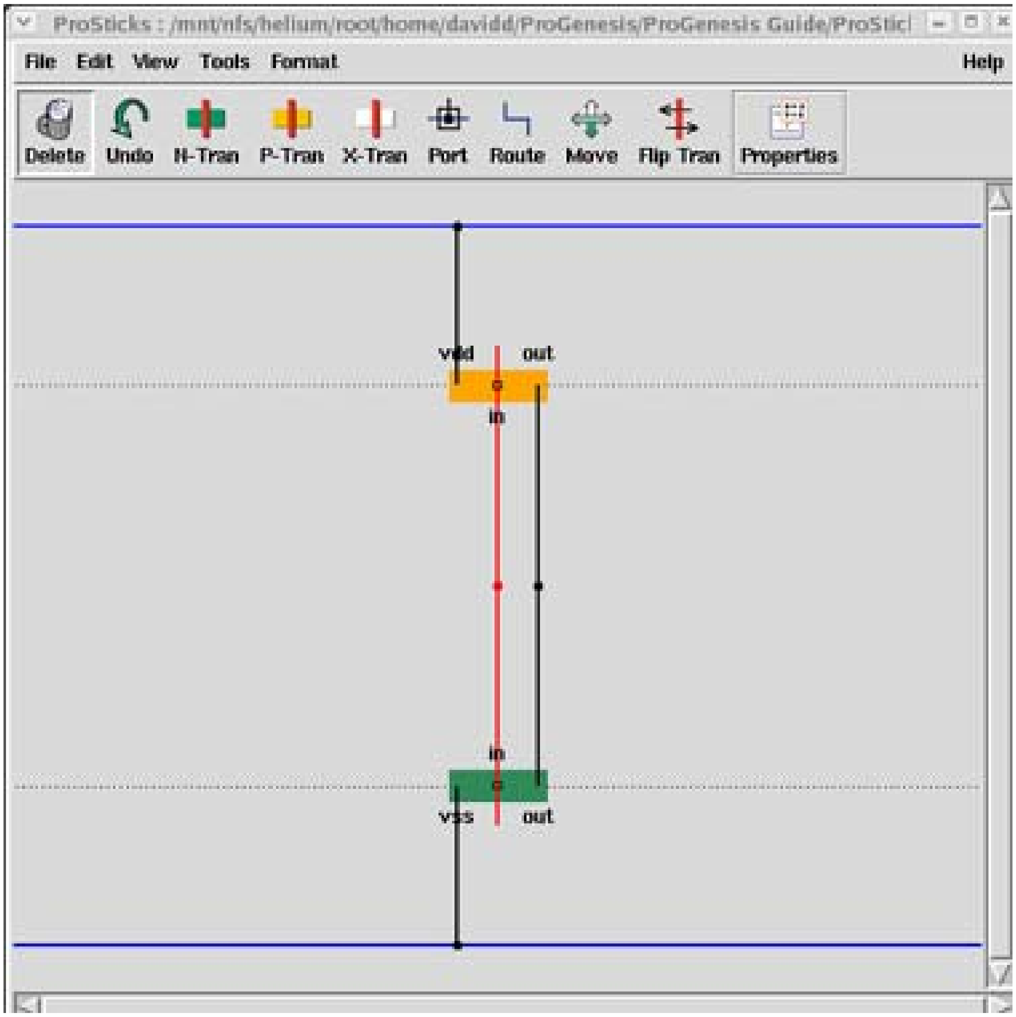
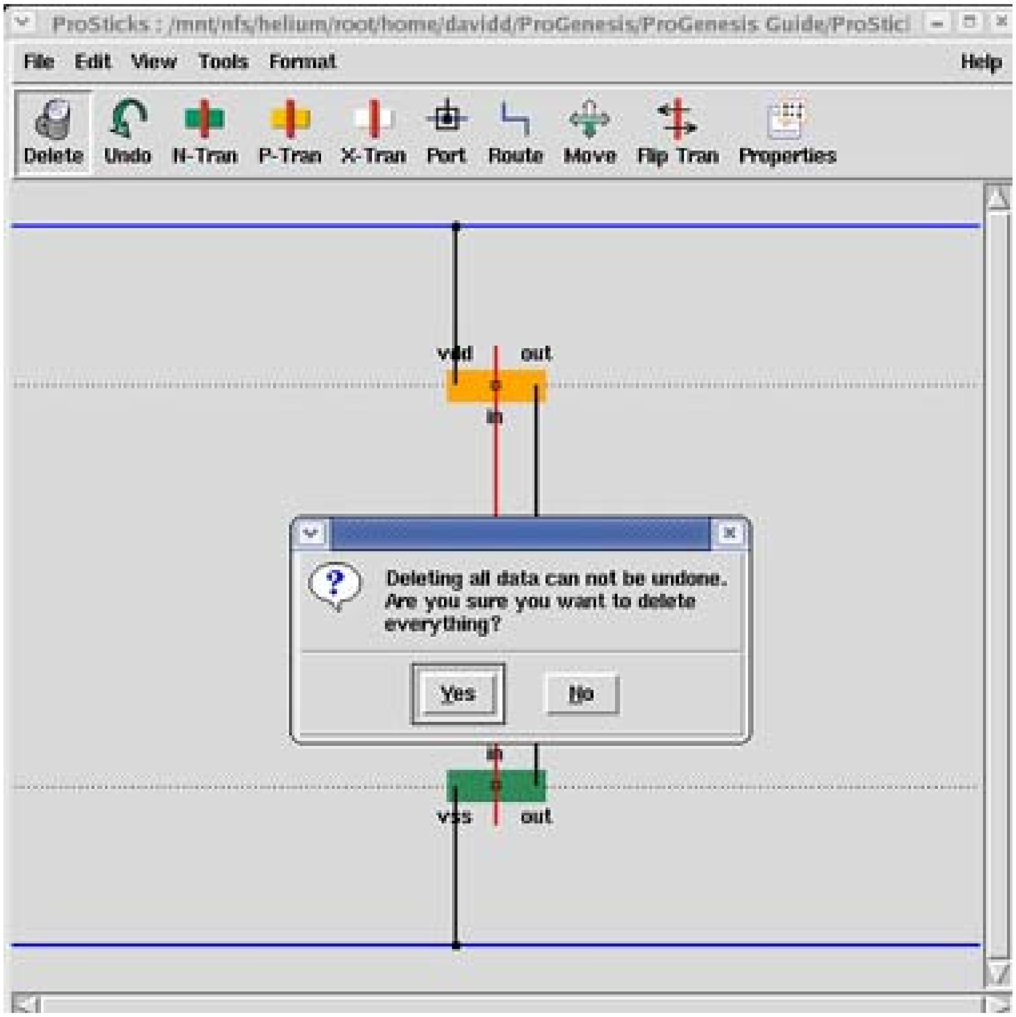


Figure 37: Deleting the poly feed also deleted the diode attached to it

Delete All

If you wish to delete everything you've done, the last option on the menu is the best approach. Simply select Delete All from the Edit menu. You'll get a warning like the one shown in Figure 38.



Deleting all data can not be undone. Are you sure you want to delete everything?

Figure 38: ProSticks will prompt you to make sure you wish to "Delete All"

If you click OK, you'll completely clear the workspace of objects -- all of them. This step cannot be undone. No amount of wishing can make it so, so be careful!

Scale Transistors

The Scale Transistors command is used to scale the width of all transistors in the layout. The width of each transistor is multiplied by the scale value. For example, set the scale to 0.9 to resize all transistors to be 90% of their current width. Set the value to 2.0 to double the size of all transistor widths.

View Menu

The View menu allows you to set some ProSticks configuration options. This menu is shown in Figure 39.

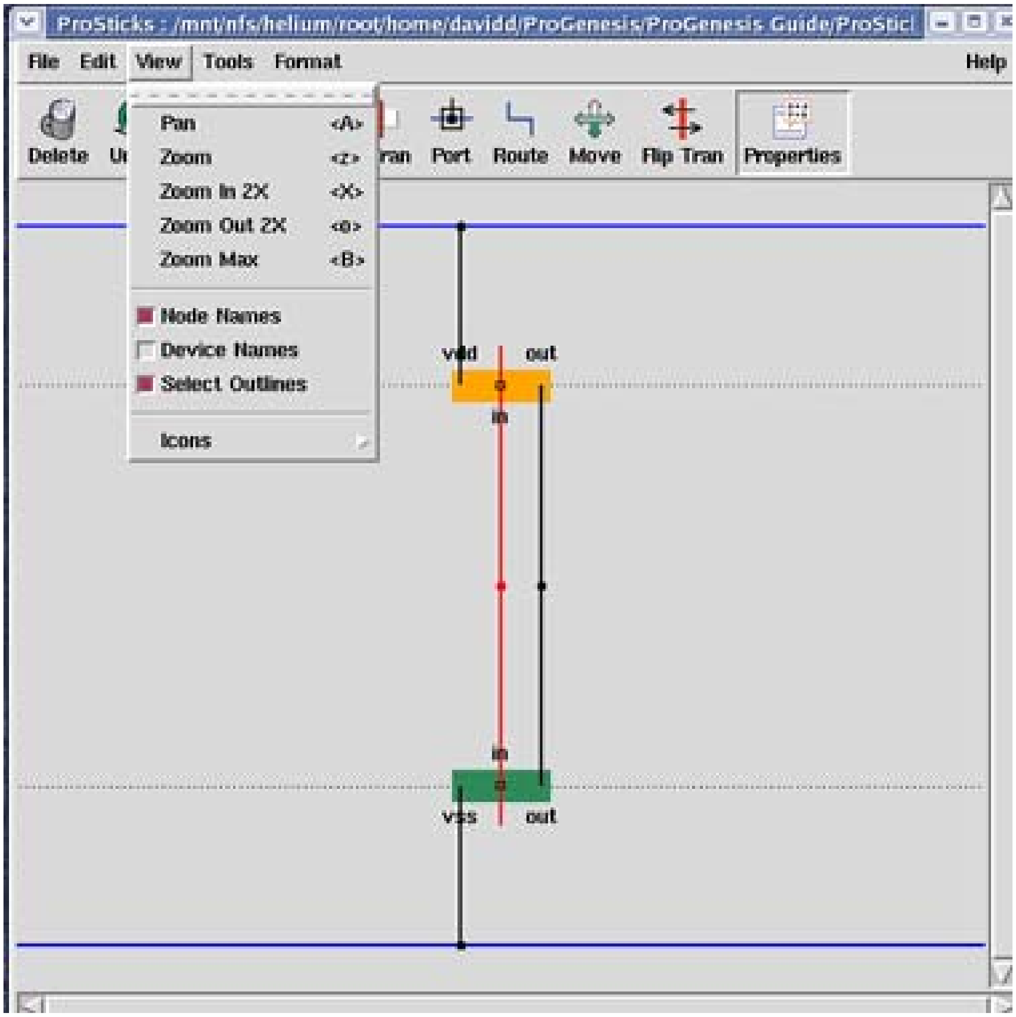


Figure 39: Control ProSticks configuration from the options menu

The dotted line means that this is another tear-off menu, so if you select the dotted line, a separate View menu window will be opened.

Node Names

One item on the View menu is Node Names. By default, this option is turned on; this allows the display of node names within the cell. If you deselect the box, the names will disappear. They are still retained within ProSticks, and you can see them using the Name Nodes or Properties commands from the Edit menu, but they will not appear in the workspace. This is especially useful in large, complex cells with long node names; it's sometimes easier to work out placement when node names aren't obscuring your view of the cell routing.

Select Outlines

Another item on the View menu is Select Outlines. The default setting causes an outline to appear around cell objects when you've selected a menu option that acts on cell objects. This is a useful feature because it helps you see on which element you will be performing the action. If you wish, you may turn this off by deselecting the Select Outlines box.

Icons

Look in the Icon submenu for various Icon options:

Icon Bar

This option, enabled by default, displays the buttons in the tool bar. Deselect this option to remove the tool bar from the ProTech interface. If this option is not selected, the other options on the Icons submenu will not be relevant.

Icon Labels

This option, enabled by default, displays text labels on the buttons in the tool bar.

Small Icons

This option, enabled by default, displays small button icons in the tool bar. Select Large Icons instead to display bigger button icons.

Large Icons

This option displays large button icons in the tool bar. Select Small Icons instead to display smaller button icons.

Tools Menu

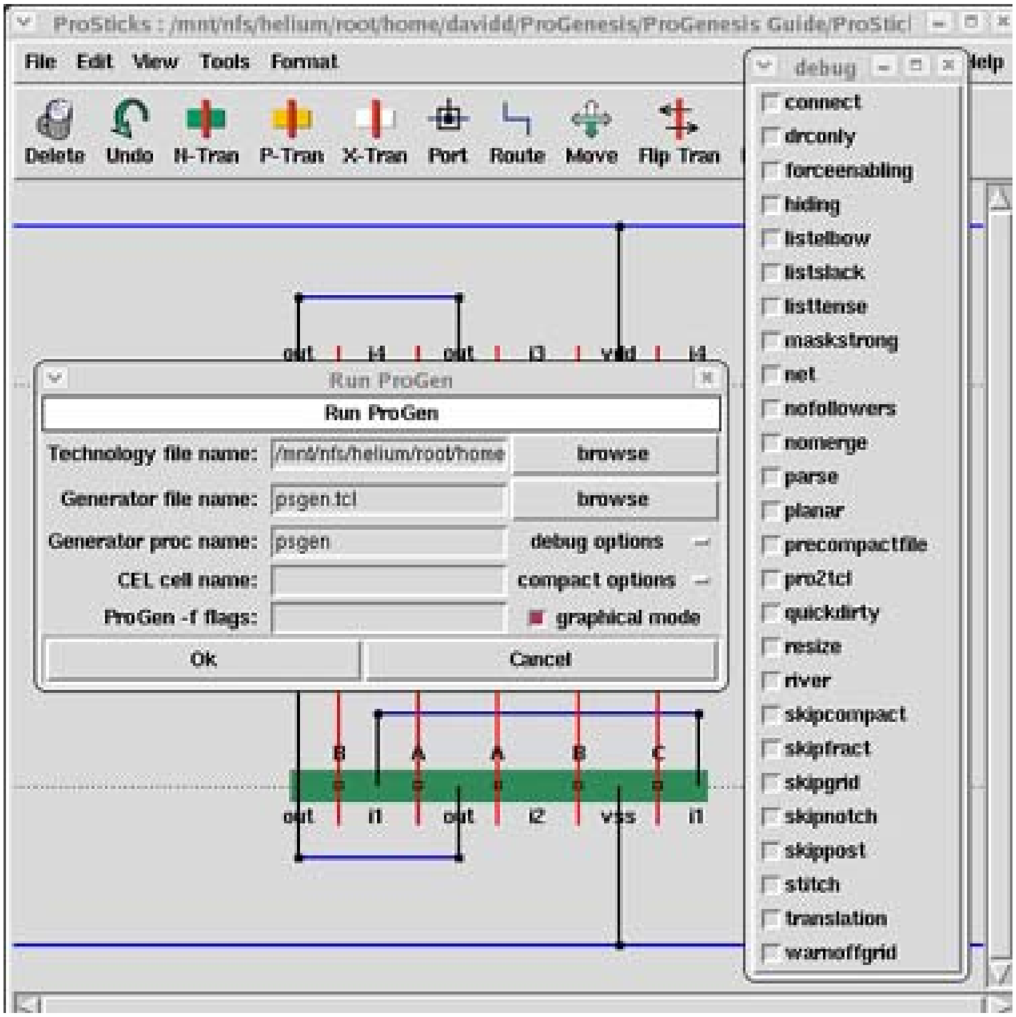
The Tools menu provides access to utilities that can be used on the cell design.

ProGen...

You should specify the generator file name and procedure name as described in the Create Generator option. Remember to use names you'll be able to identify later.

The next item in the Run ProGen window shown in Figure 13 is the Debug options button. When you click on that button, it brings up the list of options shown in Figure 40.

The ProGen option creates the generator and the ProGen PSD cell file before starting ProGen itself. This means that when you select the ProGen command, it performs the same functions as the Create Generator command described earlier before it.



Run ProGen

Technology file name: /mnt/nfs/helium/root/home

Generator file name: psgen.tcl

Generator proc name: psgen

CEL cell name:

ProGen -f flags:

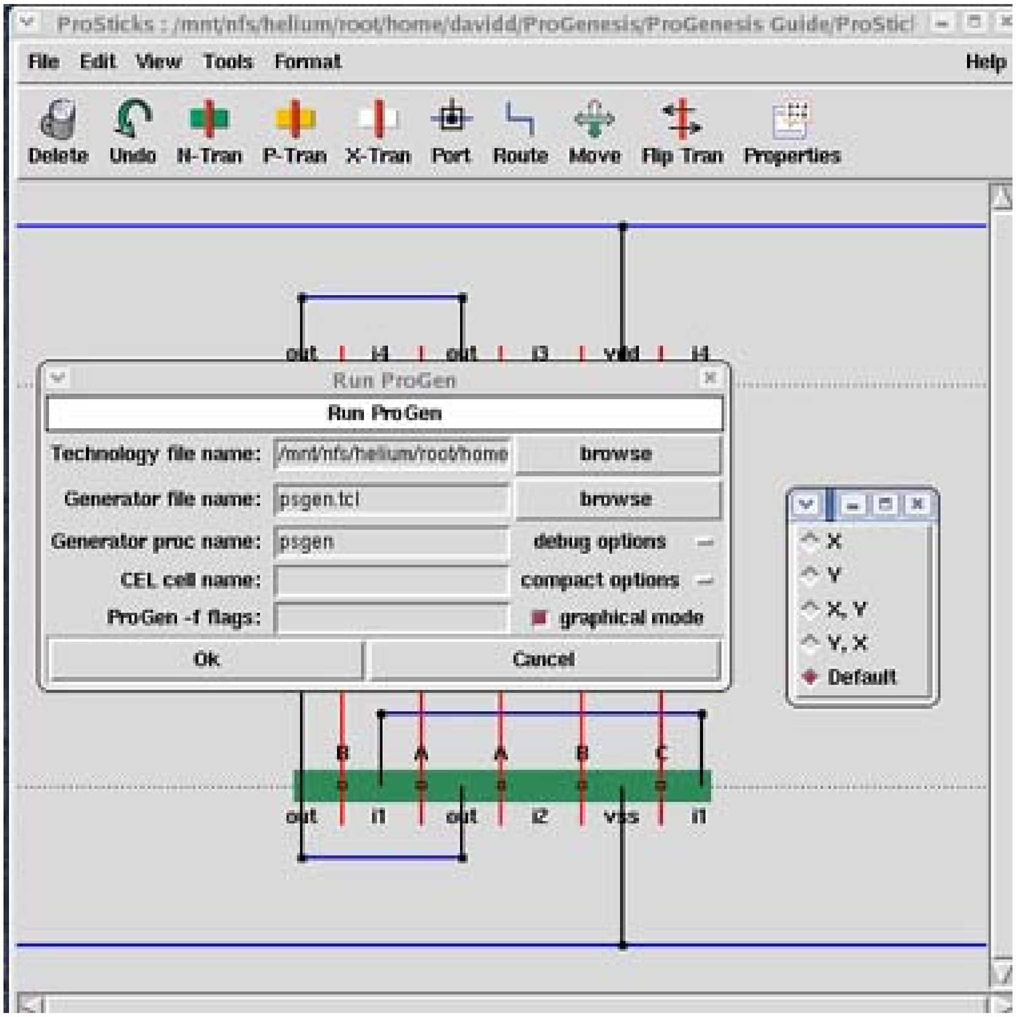
■ graphical mode

Figure 40: Select the debug options you wish to use while running ProGen

Note that there is a line of dashes at the top of this menu. This means that this is a tear-off menu and can be left open and moved around the screen.

The options listed on this menu allow you to easily specify the ProGen command-line debug options. For more detailed information on these options, refer to Running Existing Layout Generators.

After you have set the debug options as you wish, you can return to the Run ProGen window to set the compaction options by clicking the compact options button. This button will bring up the tear-off menu shown in Figure 41.



Run ProGen

Technology file name: /mnt/nfs/helium/root/home [browse]

Generator file name: psgen.tcl [browse]

Generator proc name: psgen [debug options]

CEL cell name: [compact options]

ProGen -f flags: ■ graphical mode

[Ok] [Cancel]

Figure 41: Select the compaction order from the compact options tear-off menu

The option you select controls the order of the axis of compaction. See Running Existing Layout Generators for a full description of compaction ordering.

After you have set the compaction options, click OK on the Run ProGen menu to start the layout and compaction. As ProGen starts, you'll see the layout appearing and being manipulated as shown in Figure 42.

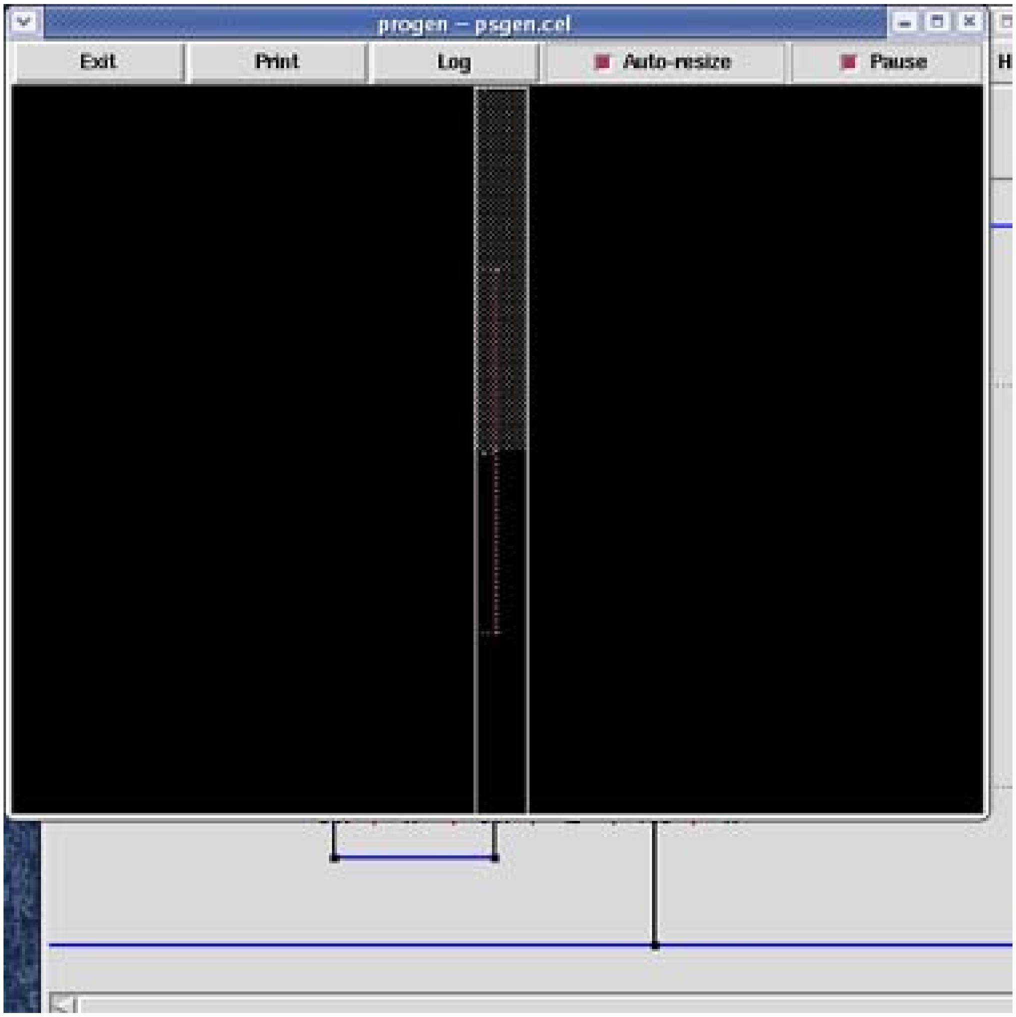


Figure 42: ProGen starts laying out the elements specified in ProSticks

As the ProGen run continues, the elements will all be placed and the compaction process will begin. The cell in Figure 43 is in the layout stage.

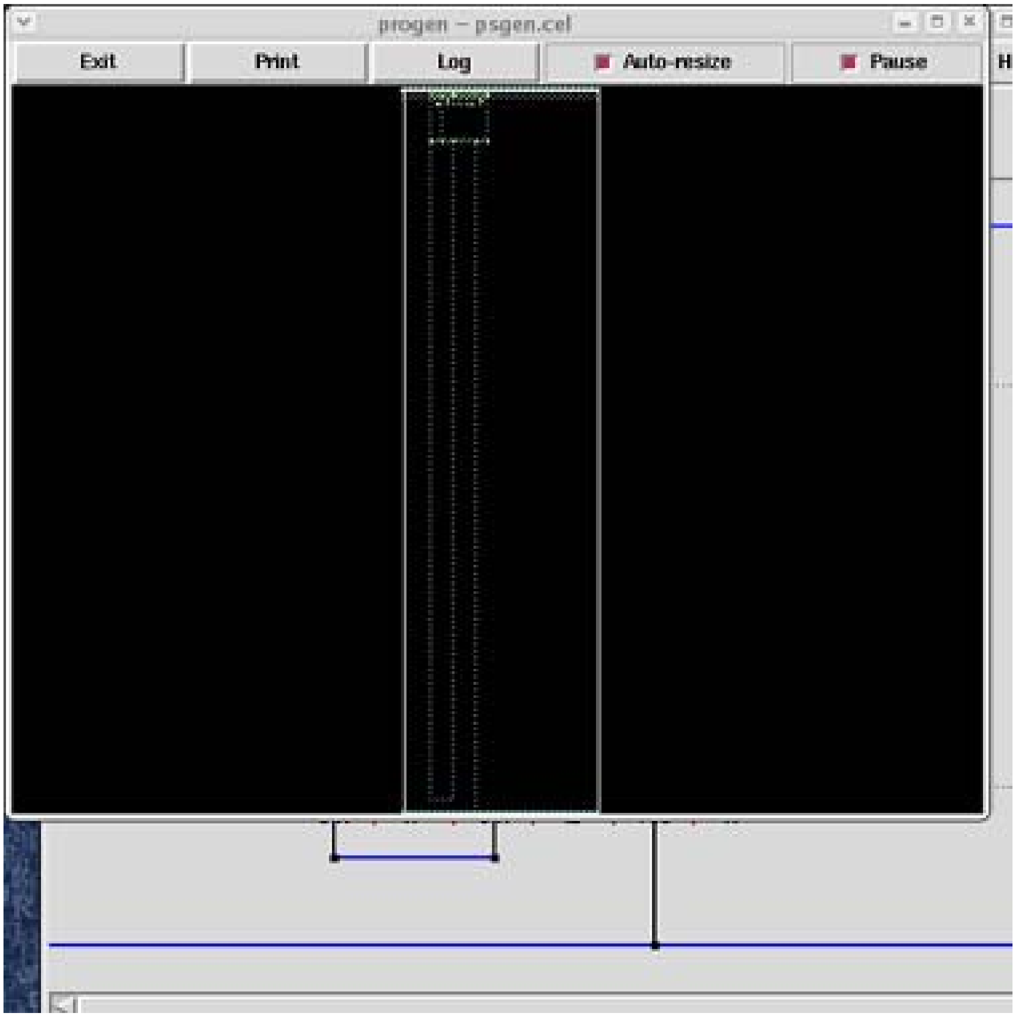


Figure 43: When all the elements have been placed, ProGen begins compaction

As the compaction process progresses, the size of the cell will decrease in the ProGen window. Click the Resize button to expand the view to make the cell layout more visible. A completed and resized cell is shown in Figure 44.

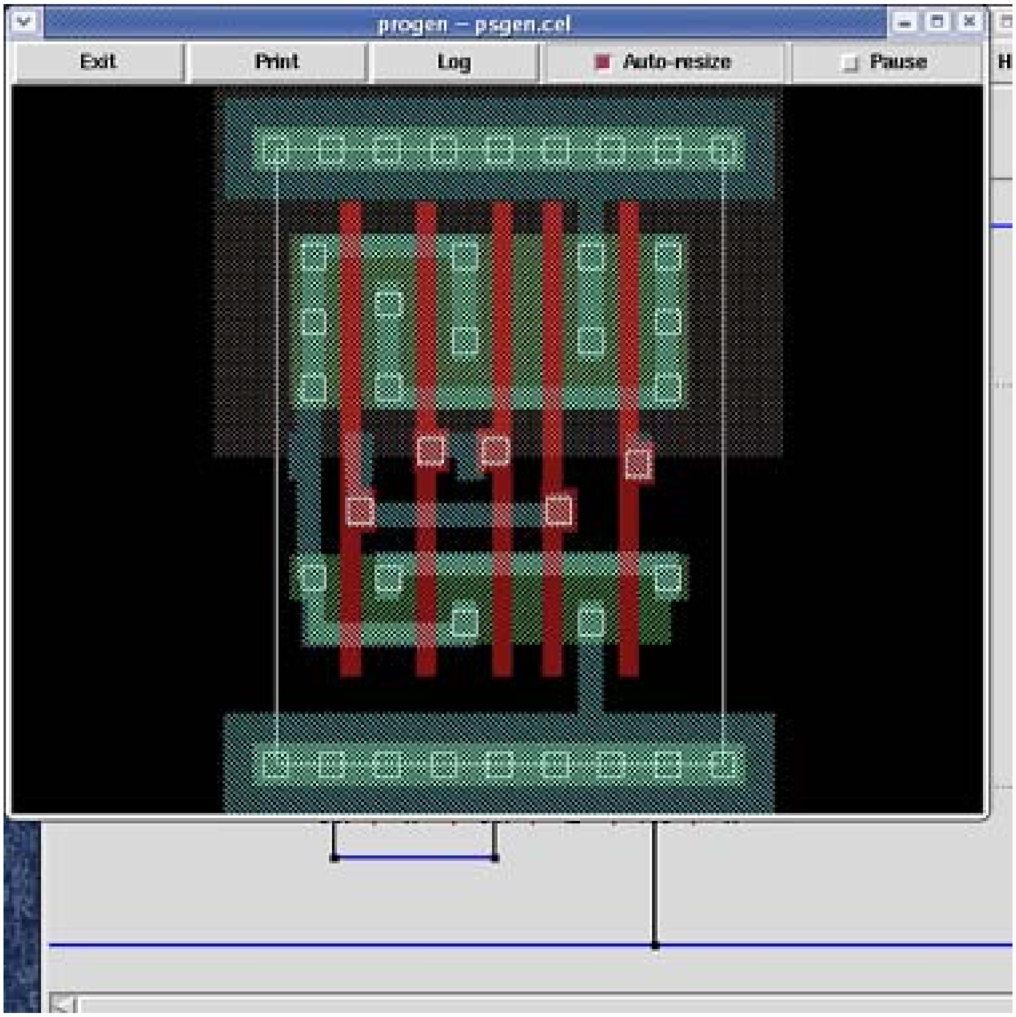
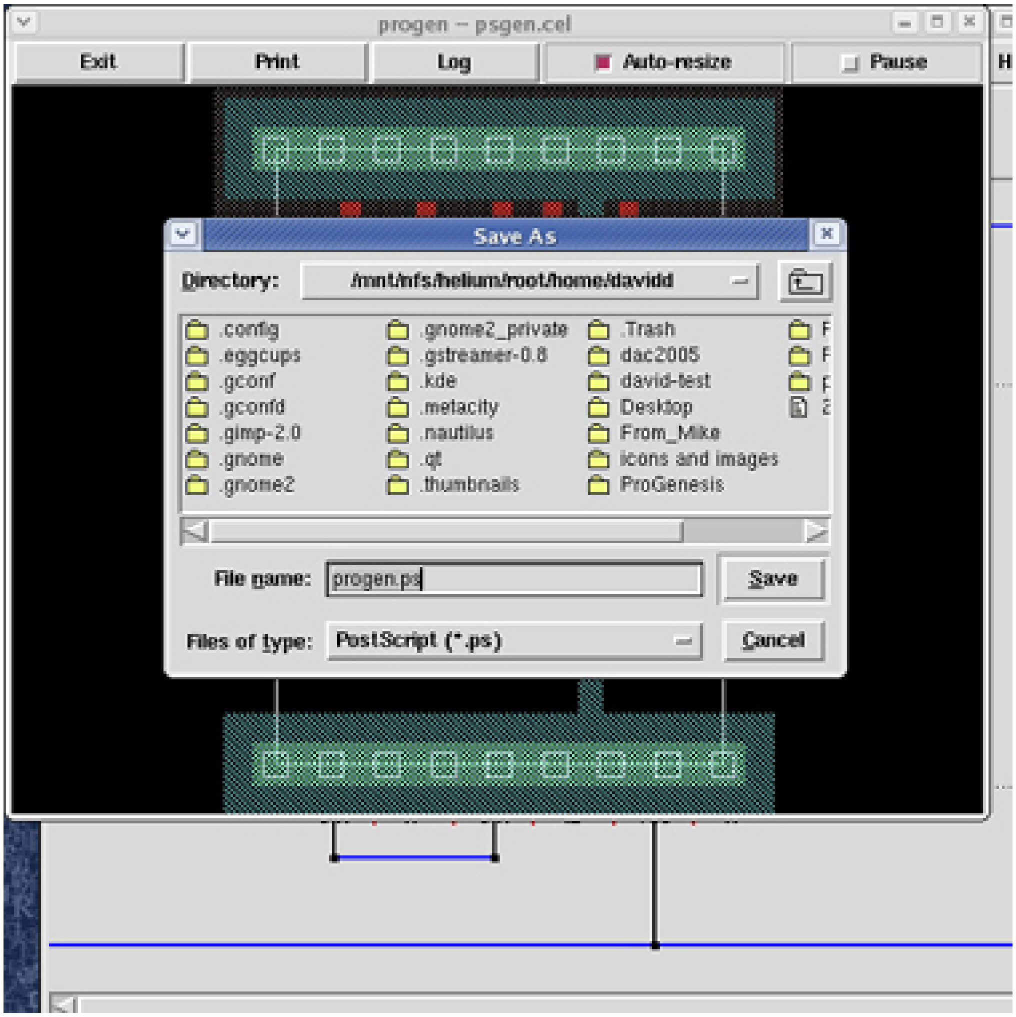


Figure 44: ProGen finishes compacting the cell

You can save a copy of the finished cell as a PostScript file by clicking the print button. This will bring up a file-saving window as shown in Figure 45.



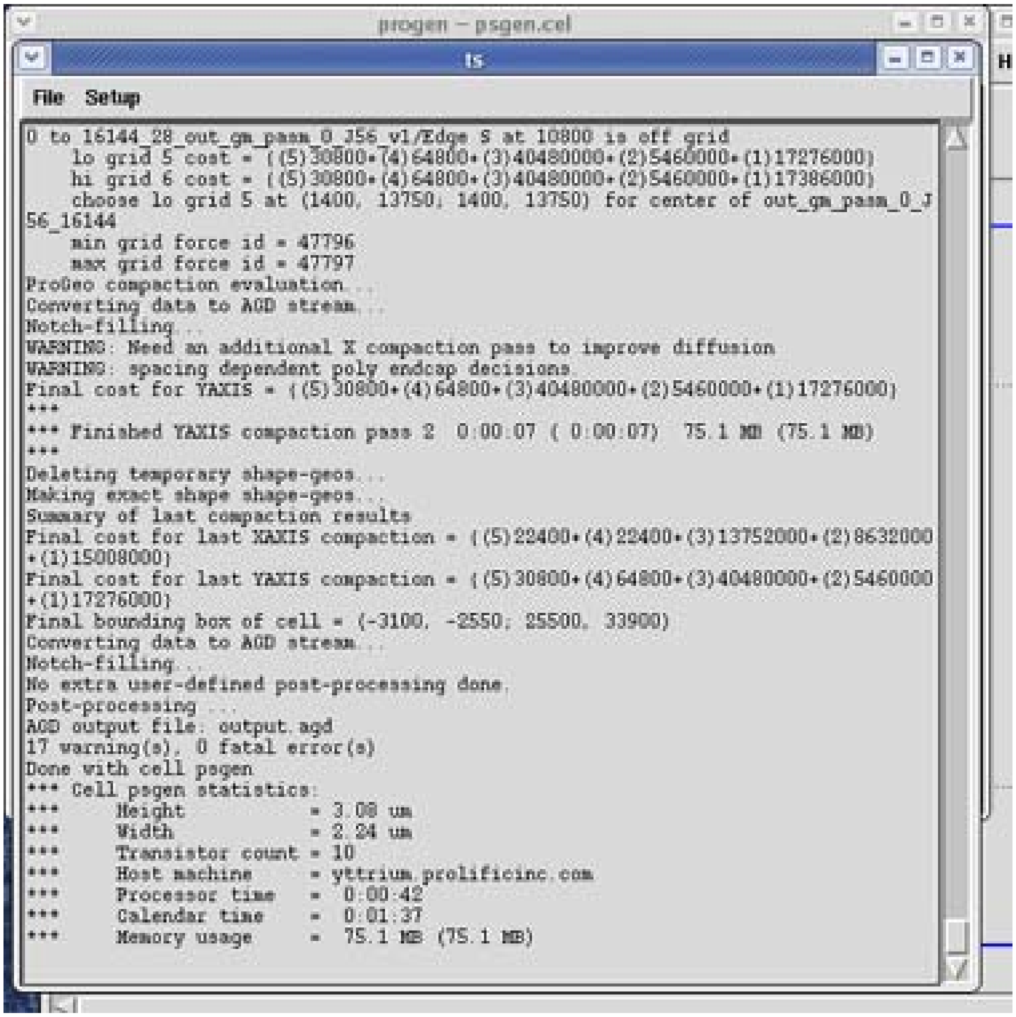
Directory: /mnt/nfs/helium/root/home/davidd

File name: progen.ps

Files of type: PostScript (*.ps)

Figure 45: Output the final ProGen layout to a PostScript file

You can view the ProGen progress by looking at the log file. Simply press the log button, which will bring up the window shown in Figure 46.



```
0 to 16144_28_out_gm_pasm_0_J56_v1/Edge 5 at 10800 is off grid
    lo grid 5 cost = ((5)30800+(4)64800+(3)40480000+(2)5460000+(1)17276000)
    hi grid 6 cost = ((5)30800+(4)64800+(3)40480000+(2)5460000+(1)17386000)
    choose lo grid 5 at (1400, 13750; 1400, 13750) for center of out_gm_pasm_0_J
56_16144
    min grid force id = 47796
    max grid force id = 47797
ProGeo compaction evaluation...
Converting data to AGD stream...
Notch-filling...
WARNING: Need an additional X compaction pass to improve diffusion
WARNING: spacing dependent poly endcap decisions.
Final cost for YAXIS = ((5)30800+(4)64800+(3)40480000+(2)5460000+(1)17276000)
***
*** Finished YAXIS compaction pass 2  0:00:07 ( 0:00:07)  75.1 MB (75.1 MB)
***
Deleting temporary shape-geos...
Making exact shape shape-geos...
Summary of last compaction results
Final cost for last XAXIS compaction = ((5)22400+(4)22400+(3)13752000+(2)8632000
+(1)15008000)
Final cost for last YAXIS compaction = ((5)30800+(4)64800+(3)40480000+(2)5460000
+(1)17276000)
Final bounding box of cell = (-3100, -2550; 25500, 33900)
Converting data to AGD stream...
Notch-filling...
No extra user-defined post-processing done.
Post-processing ...
AGD output file: output.agd
17 warning(s), 0 fatal error(s)
Done with cell psgen
*** Cell psgen statistics:
***     Height            = 3.08 um
***     Width             = 2.24 um
***     Transistor count  = 10
***     Host machine      = yttrium.prolificinc.com
***     Processor time    =  0:00:42
***     Calendar time     =  0:01:37
***     Memory usage      =  75.1 MB (75.1 MB)
```

Figure 46: View the progress of ProGen by looking at the log file

The cell in Figure 43 is finished. Notice that the last two lines indicate no warnings or fatal errors occurred and that the cell had been created. There is additional information in this log file, including the final compaction costs.

That's all there is to running ProGen from ProSticks!

Check Opens

This feature looks at all the named nets in the cell and reports any that are not fully connected.

Nets

Opens net information dialog.

The net dialog displays a list of the named nets in the layout with the following information:

Nets - name of net
Status - indicates if net is "open", value of " " is closed
Type - input, output, power, ground, rail, " " (internal)

The dialog can remain open during all editing operations and it will automatically update its information.

You can select nets in the dialog and they will be displayed selected in the layout. You can edit the name and type in the dialog and it will change all routes/objects in the layout with the corresponding net names to the new values.

This command is very useful when loading spice to identify all nodes on a given net.

Check Ports

Reports input/output nets not having a port.

This command lists all input/output nets that do not have a port on them. These nets do not have to have ports to create valid generators so this acts as general information. Only those nets identified as input/output in the net dialog are considered as port candidates.

Format Menu

The next menu allows you to modify aspects of the default workspace. The options on the Format menu allow you to alter the layout of the transistor rows and the power and ground rails. The Format menu is shown in Figure 47.

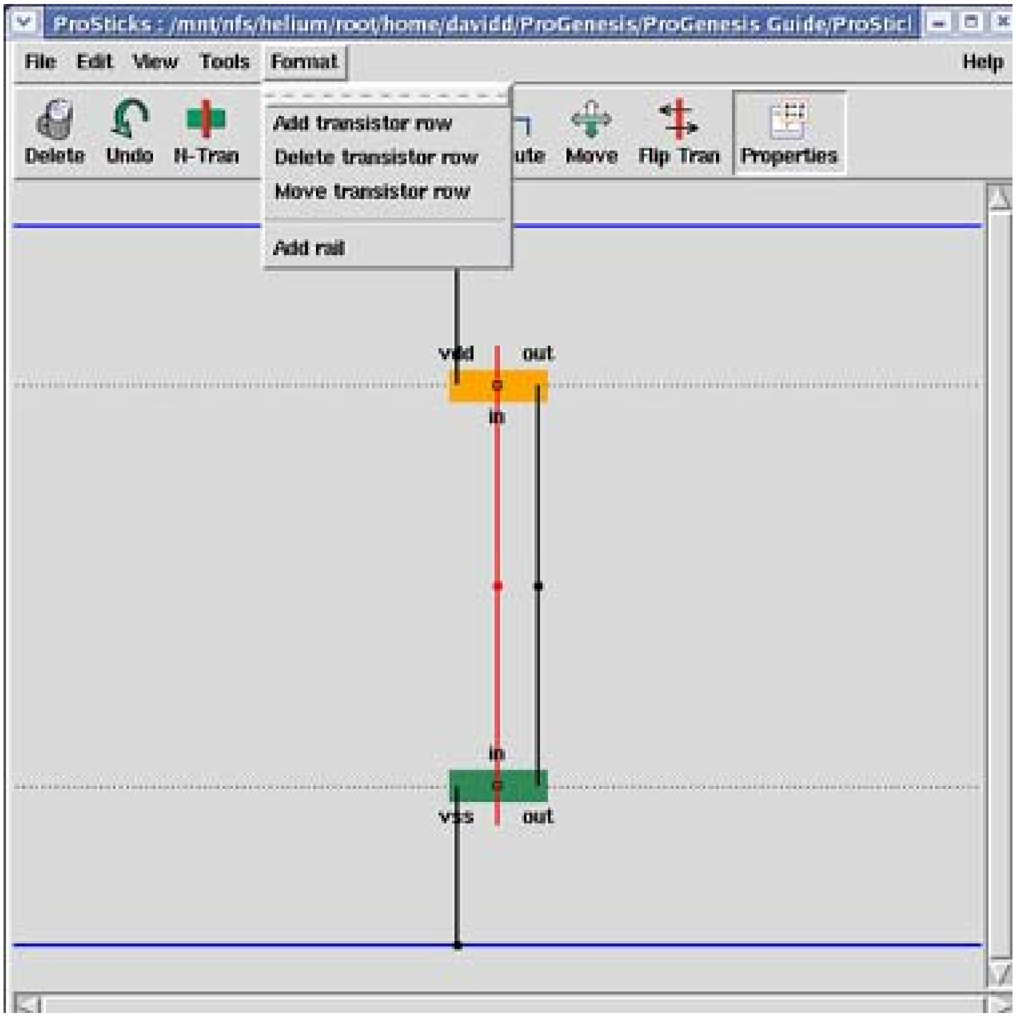


Figure 47: Use the format menu to add rails and alter transistor rows

This is another tear-off menu, so you can select the dashed line to open an independent window containing the Format menu options. The first three menu items allow you to add, remove, or move transistor rows from the workspace.

Add Transistor Row

You might wish to add additional transistor rows if you want to stack transistors. In Figure 48, an additional transistor row has been added to allow the transistors to be stacked. You can add a row simply by clicking Add transistor row and then clicking on the workspace. Dotted lines representing the added transistor rows appear as shown in Figure 48.

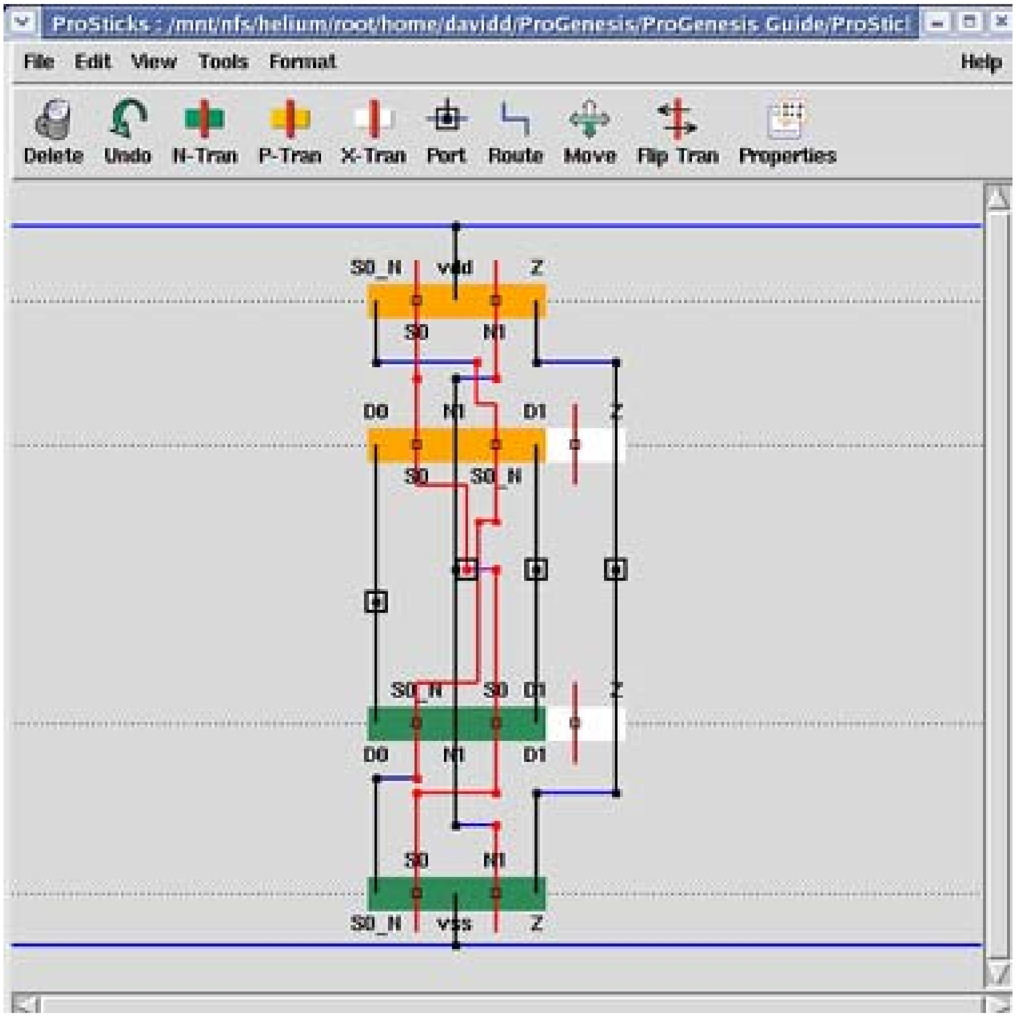


Figure 48: Adding transistor rows allows cells with stacked transistors

The first transistor you add to the row determines whether it is a row of n-transistors or p-transistors. You are free to add additional transistors to the row, just as you would to the two that ProSticks provides by default. Notice the white x-transistors used to route across the transistor rows. Any feed wire crossing a transistor row must do so through a pass-through transistor, also known as an x-transistor. These objects maintain valid routing when there are more than two transistor rows.

Delete Transistor Row

If you wish to remove a transistor row, simply click the Delete Transistor Row option and click on the row you wish to remove. Note that the row must be empty before it can be deleted, so you must use the Delete or Move command to remove the transistors from the row before you can remove the row. Also note that the Delete all command does not delete transistor rows.

Move Transistor Row

If you wish to move a transistor row, you can click on the Move transistor row option and then click and drag the row to its new location. You can only move an empty row, so you must delete or move away any transistors on the row first.

Add Rail

The remaining two options on the Format menu allow you to add additional power or ground rails. (We certainly hope you don't have to do this, but we want to give you full control over the environment.)

To add a new ground rail, click Add ground rail, then click on the workspace where you'd like the rail to run. To add a new power rail, click Add power rail, then click on the workspace where you'd like the rail to run. The new power and ground rails can be manipulated using the Edit menu commands such as Move and Delete.

Routing across transistor rows with transistor route-overs

ProGen is a channel-based system (ProSticks models ProGen). The transistors are initially placed in rows and routed together in the channel between the rows. To bridge from one channel (say, below the n-transistors) to the next (say, the center channel), the routes have to connect to something along the transistor row. This can be a dummy transistor (a pass-through transistor in ProSticks), or it can be a real transistor source/drain/gate connection, or you can "route over" an existing transistor gate region using a metal layer connecting to the transistor route-over point.

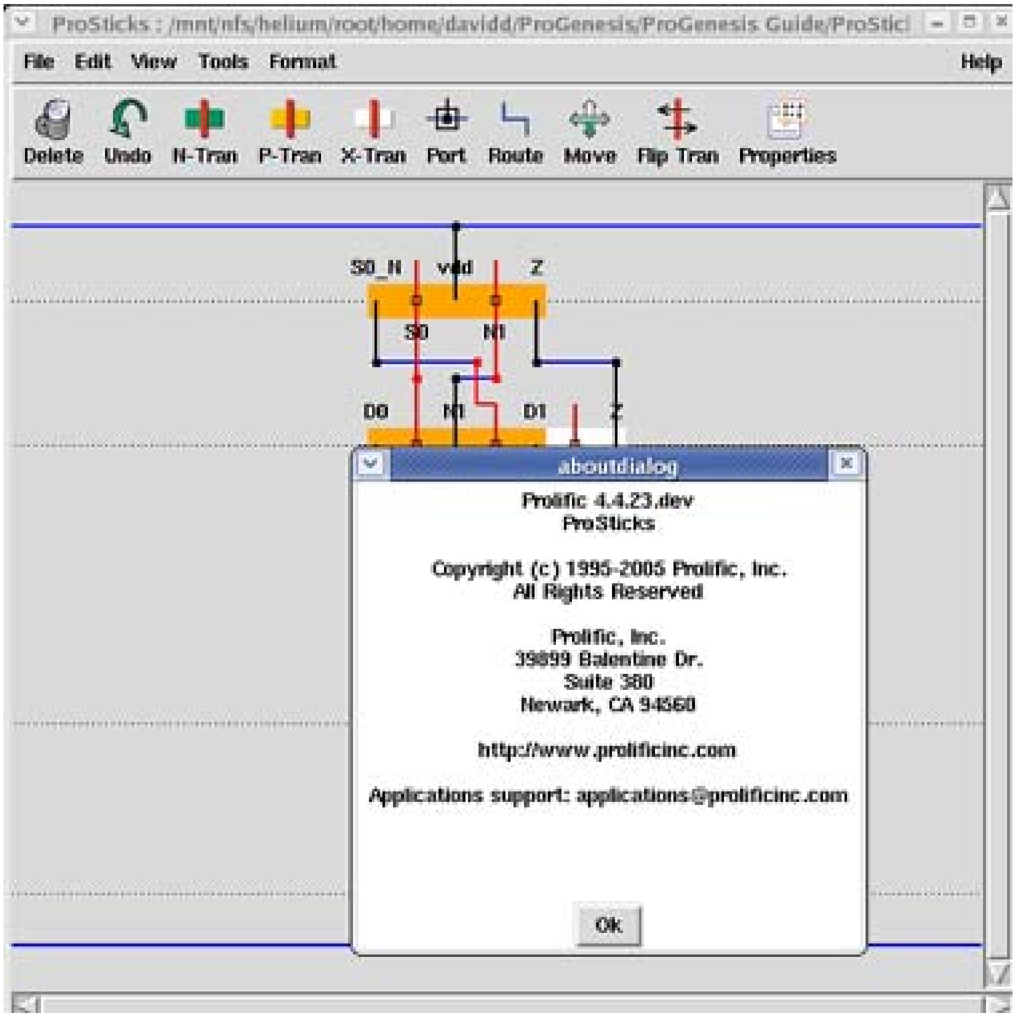
You’ll notice that, in ProSticks, each transistor has a small black square in its center on top of the gate. This is the route-over contact point. You can route to this point from one side of the channel and route from this point to the other side of the channel, thus bridging the channels. ProGen will initially route over the transistor gate, between the contacts, in a metal layer (typically m1).

Help Menu

The last menu is the Help menu.

- Contents...
- Menu Bar...
- Icon Bar...
- Concepts...
- About ProSticks...

This menu is mostly useful for identifying the version of ProSticks you're using once you've started running the tool (the status line tells you the version number when you start ProSticks). Select About ProSticks to bring up a window like the one shown in Figure 49.



Prolific 4.4.23.dev
ProSticks

Copyright (c) 1995-2005 Prolific, Inc.
All Rights Reserved

Prolific, Inc.
39899 Balentine Dr.
Suite 380
Newark, CA 94560

http://www.prolificinc.com

Applications support: applications@prolificinc.com

Figure 49: Find the version, copyright, and Prolific's address from About ProSticks

Toolbar

The ProTech Toolbar is located below the menu bar. The Toolbar contains the following tools:

- Delete
- N-Transistor
- P-Transistor
- X-Transistor
- Route
- Move
- Flip Transistor
- Properties

Delete

The Delete tool duplicates the operation of the Edit menu's Delete option: it deletes the specified element. If you wish to undo the last deletion, you can select the Undo command from the Edit menu. Remember that this will only undo the last action you took.

Deleting one item often implies deletion of another, and ProSticks will delete any objects that are dependent upon an item you're deleting.

N-Tran

After you select this button, n-transistors appear as green-colored blocks when you click on the ProSticks workspace. This duplicates the operation of the Edit menu's n-transistor option.

P-Tran

After you select this button, p-transistors appear as orange-colored blocks when you click on the ProSticks workspace. This duplicates the operation of the Edit menu's p-transistor option.

X-Tran

After you select this button, x-transistors appear as green-colored blocks when you click on the ProSticks workspace. This duplicates the operation of the Edit menu's x-transistor option.

Route

This button allows you to route the components of your cell. If you select this tool, you can connect the elements you've placed in the cell. Route makes connections manually; after you select this option, click a gate, source, or drain and drag to another cell element to route them together. This duplicates the operation of the Edit menu's Route option.

Move

This tool allows you to select any element in the workspace and move it somewhere else. When you click on this icon, you will notice that as you pass the cursor over the workspace a dotted line will appear around elements on the screen.

To move a highlighted object, simply click it and drag it to the new location. This functionality is the same as that of the Edit menu's Move option.

Flip Tran

This tool is useful when you wish to align the connections between two transistors. Transistor flipping is also useful for aligning left and right feeds of n- and p-transistors to minimize jogs. This is equivalent to using the Flip Transistor option from the Edit menu.

Properties

The Properties tool, like the Edit menu's Properties option, allows the properties of layout elements to be seen, and in some cases, to be edited. Properties can be entered for poly or metal trunks and feeds, transistors, resistors, and diodes. Simply click the Properties button and then click an element.

A window will appear showing the properties for the selected element. Properties vary from element type to element type, but such attributes as net names, layer, and object type are available for different elements.

ProSticks Guide

Building a Cell

The features discussed in the ProPPR section are simply tools you can use to produce the layout you want. In the next two examples, we'll go step-by-step through the process of creating a topology and running ProGen to produce layout. First, we'll look at what to do to produce layout from a netlist; in the second example, we'll generate the layout from scratch.

For our purposes, topology is the logical and physical organization of a cell design.

Both examples are intended to show how to make practical use of the functions described in the Navigating ProSticks section. We'll be producing simple cells (a two-input mux from the netlist, and, manually, a two-input NAND) because the point isn't to produce the most frightening and impressive latch ever seen, but to demonstrate how to create layout. You should know enough when we're through to get as complex as you need to in ProSticks. So let's begin with the netlist!

Example 1: Imported Netlist

ProSticks users often wish to work from an HSPICE netlist. This has the advantage of creating transistors with the correct widths and node names. We will work with the netlist for a generic two-to-one mux, as shown in Example 1 -- a minimal netlist with no extraneous declarations.

```
******************
* HSPICE NETLIST *
* MUX21          *
******************
mP1 N1 S0 D0 1 P2 W=2.0 L=0.24
+ AD=  0.95 AS=  0.95 PD=  4.16 PS=  4.16
mP2 N1 S0_N D1 1 P2 W=2.0 L=0.24
+ AD=  0.95 AS=  0.95 PD=  4.16 PS=  4.16
mP4 S0_N S0 vdd 1 P2 W=2.0 L=0.24
+ AD=  1.02 AS=  1.02 PD=  4.36 PS=  4.36
mP3 Z N1 vdd 1 P2 W=2.0 L=0.24
+ AD=  2.86 AS=  2.86 PD=  9.76 PS=  9.76
mN2 N1 S0_N D0 0 N2 W=1.0 L=0.24
+ AD=  0.95 AS=  0.95 PD=  4.16 PS=  4.16
mN1 N1 S0 D1 0 N2 W=1.0 L=0.24
+ AD=  0.95 AS=  0.95 PD=  4.16 PS=  4.16
mN4 S0_N S0 vss 0 N2 W=1.0 L=0.24
+ AD=  0.68 AS=  0.68 PD=  3.36 PS=  3.36
mN3 Z N1 vss 0 N2 W=1.0 L=0.24
+ AD=  1.90 AS=  1.90 PD=  6.96 PS=  6.96
```

Example 1: Netlist for a 2-to-1 mux.

We can tell from the netlist that the n-transistors are all one micron wide, and the p-transistors are all two microns wide. There are eight transistors defined, so we will expect to see eight transistors when we import this netlist into ProSticks.

To begin, we'll start ProSticks with an empty workspace; if ProSticks is already running, we use Edit>Delete All to remove everything; otherwise, we simply start it with an empty cell:

```
> prosticks &
```

This starts ProSticks with an empty workspace as shown in Figure 1:

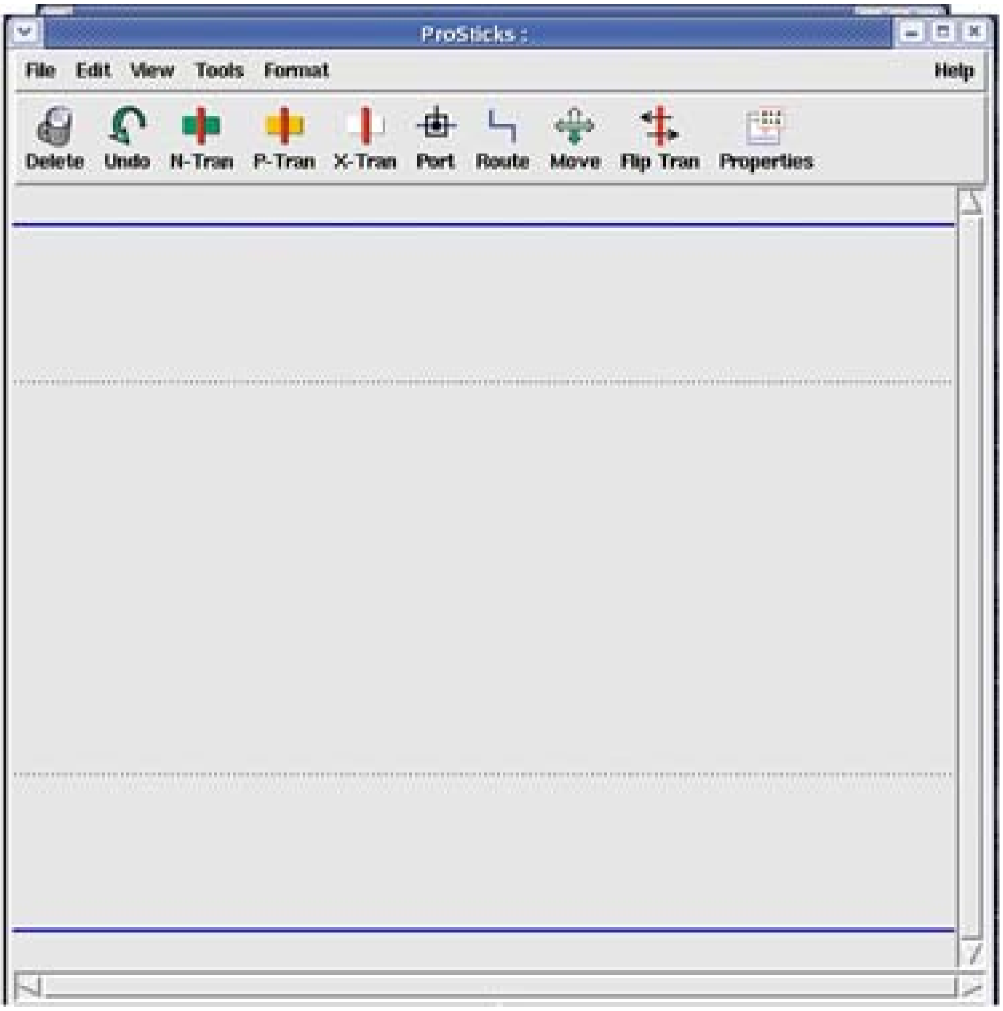


Figure 1: Start ProSticks with an empty workspace

Once the workspace is open, we can import the netlist. First, go to the File menu, which is illustrated in Figure 2.

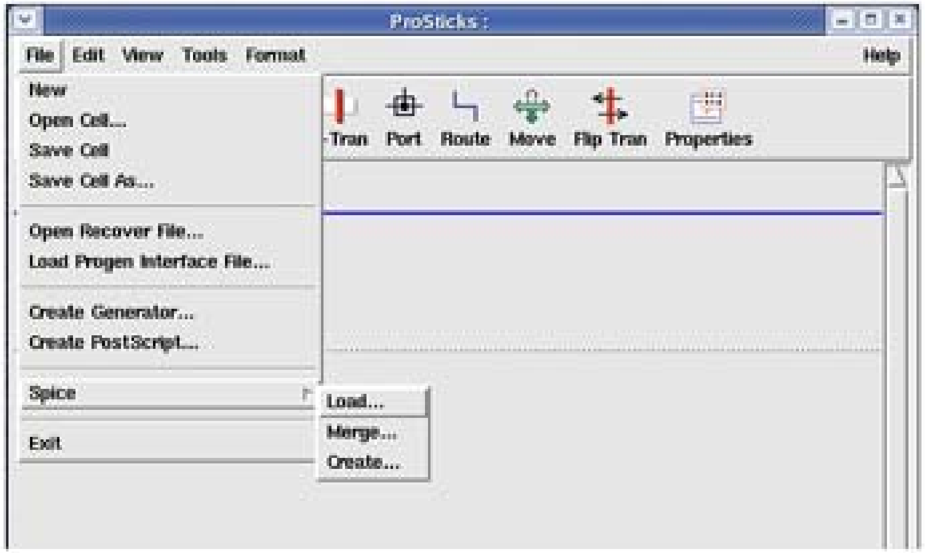


Figure 2: Select Spice>Load from the ProSticks File menu

This action will bring up the window shown in Figure 3.

.

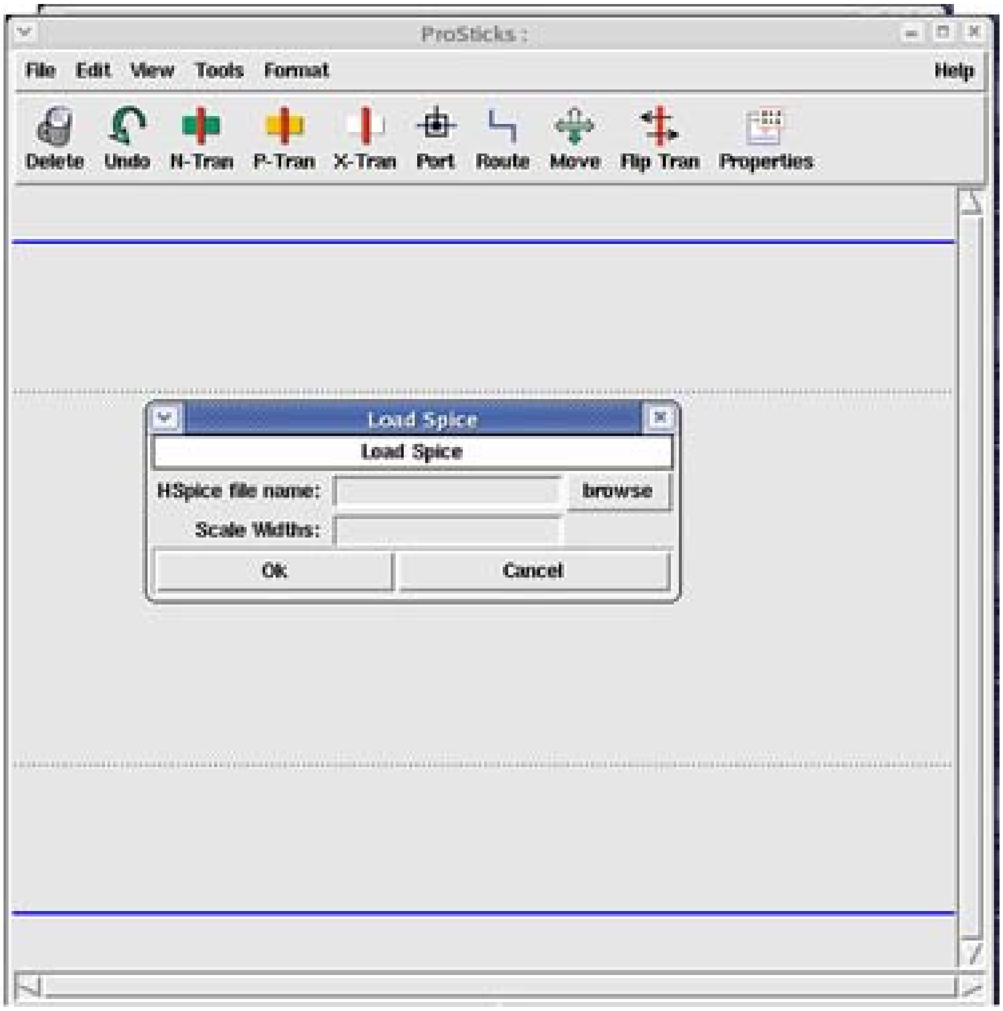
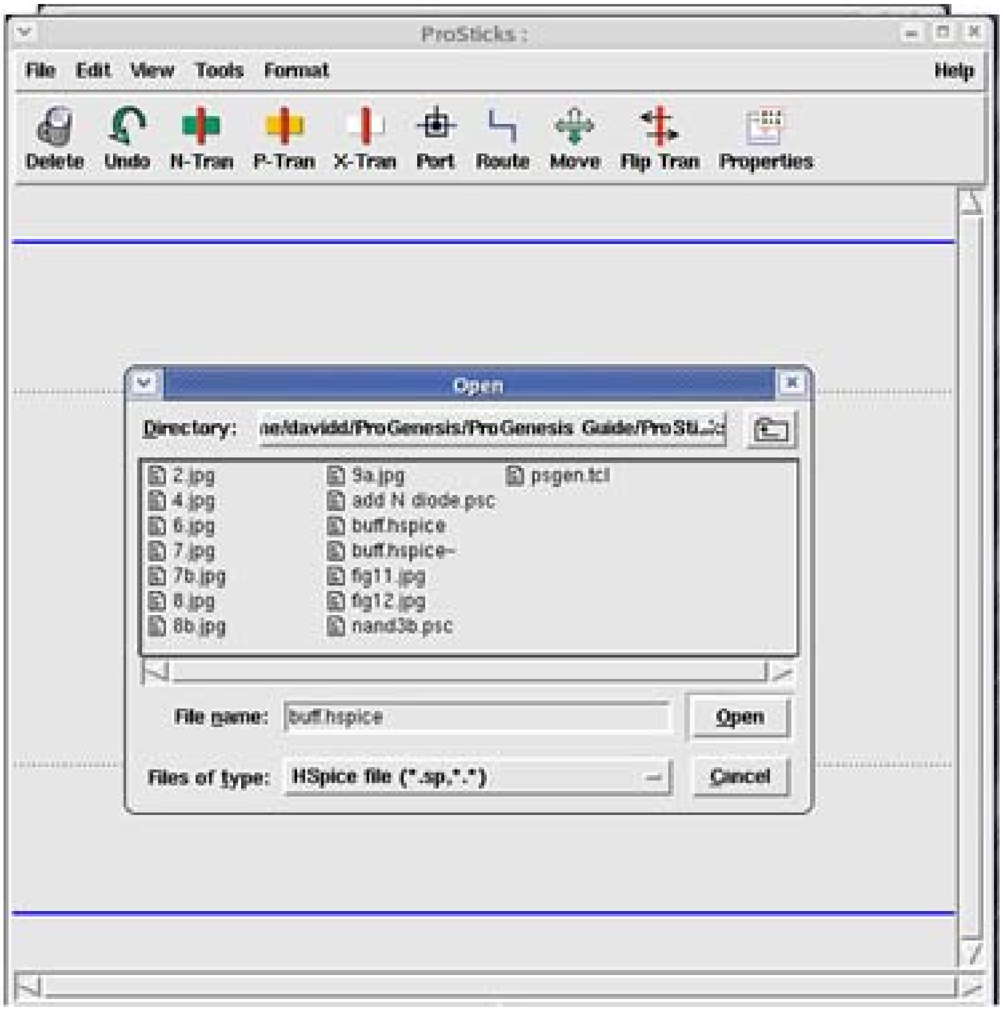


Figure 3: Select the file and scaling factor from the Load Spice window

To select the example HSPICE file name, we click the Browse button to bring up the window shown in Figure 4.



Directory: ne/davidd/ProGenesis/ProGenesis Guide/ProSti...

2.jpg
4.jpg
6.jpg
7.jpg
7b.jpg
8.jpg
8b.jpg
9a.jpg
add N diode.psc
buff.hspice
buff.hspice~
fig11.jpg
fig12.jpg
nand3b.psc
psgen.tcl

File name: buff.hspice

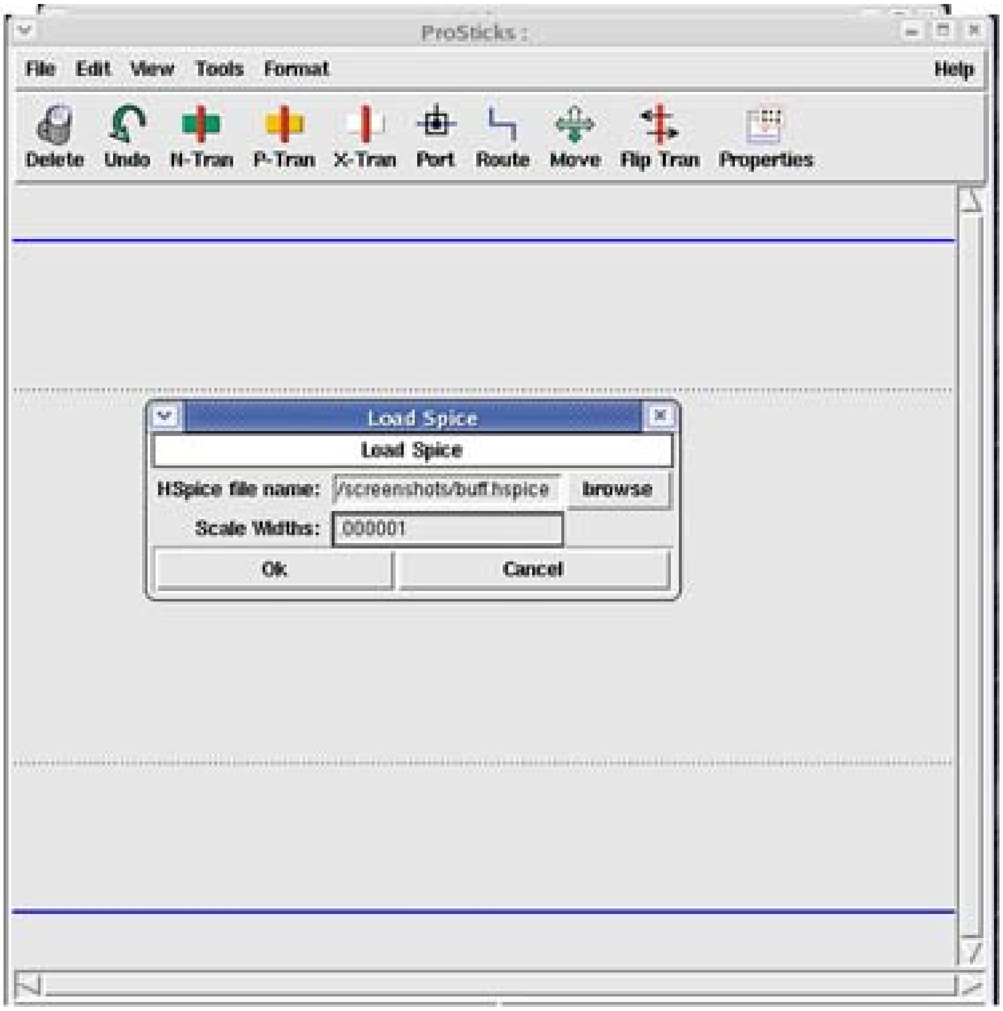
Files of type: HSpice file (*.sp,*.*)

Figure 4: Browse the directory structure to find the HSPICE file to use

Here, we've selected the file we want: /ProGenesis.../buff.hspice. All we have to do to select the file is click the Open button. When we do, the Load Spice window reappears with the HSPICE file name field populated.

Now we just have to scale the transistor widths. We know from looking at the contents of the netlist in Example 1 that the transistor widths are in microns. ProSticks doesn't assume we mean microns, however; it uses the

Spice standard width units, which are meters. The Scale Widths field depicted in Figure 52 has been set to 0.000001, which will scale the units to the correct size.



Load Spice

HSpice file name: /screenshots/buff.hspice

Scale Widths: .000001

Figure 5: Set the scale widths to convert the units to meters

Now that the Load Spice fields have been populated, we can click OK to finish the process. When we do so, the transistors described in the file are imported into the workspace as shown in Figure 6.

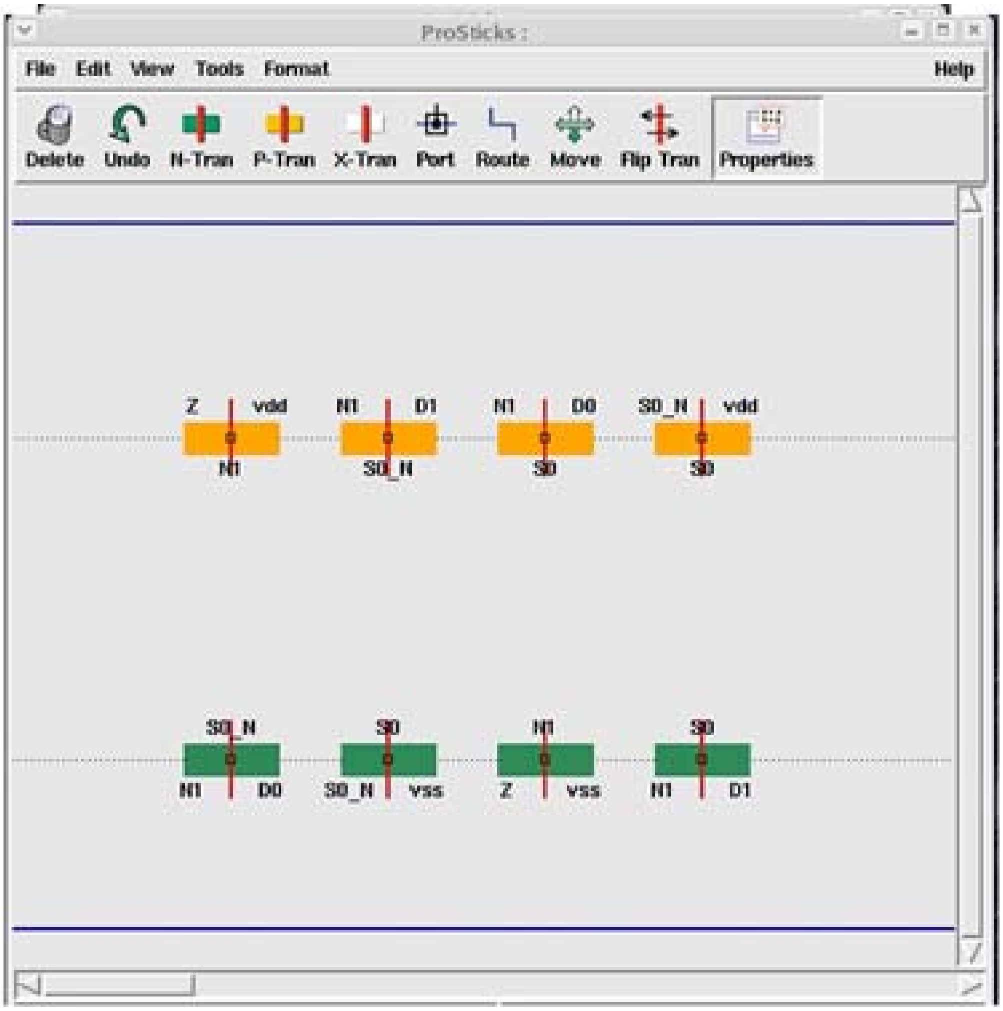


Figure 6: The imported file populates the workspace with transistors

Now we're ready for some fun. We could produce layout using the two default rows of transistors, but instead of doing that, let's live on the edge a little and stack these transistors. We may be able to produce layout that takes up less space along the x-axis by doing so.

First, we'll add the additional transistor rows. We do this by going to the **Format** menu and selecting **Add Transistor Row.** Now with each click on the workspace another transistor row is added, indicated by the dotted lines as shown in Figure 7.

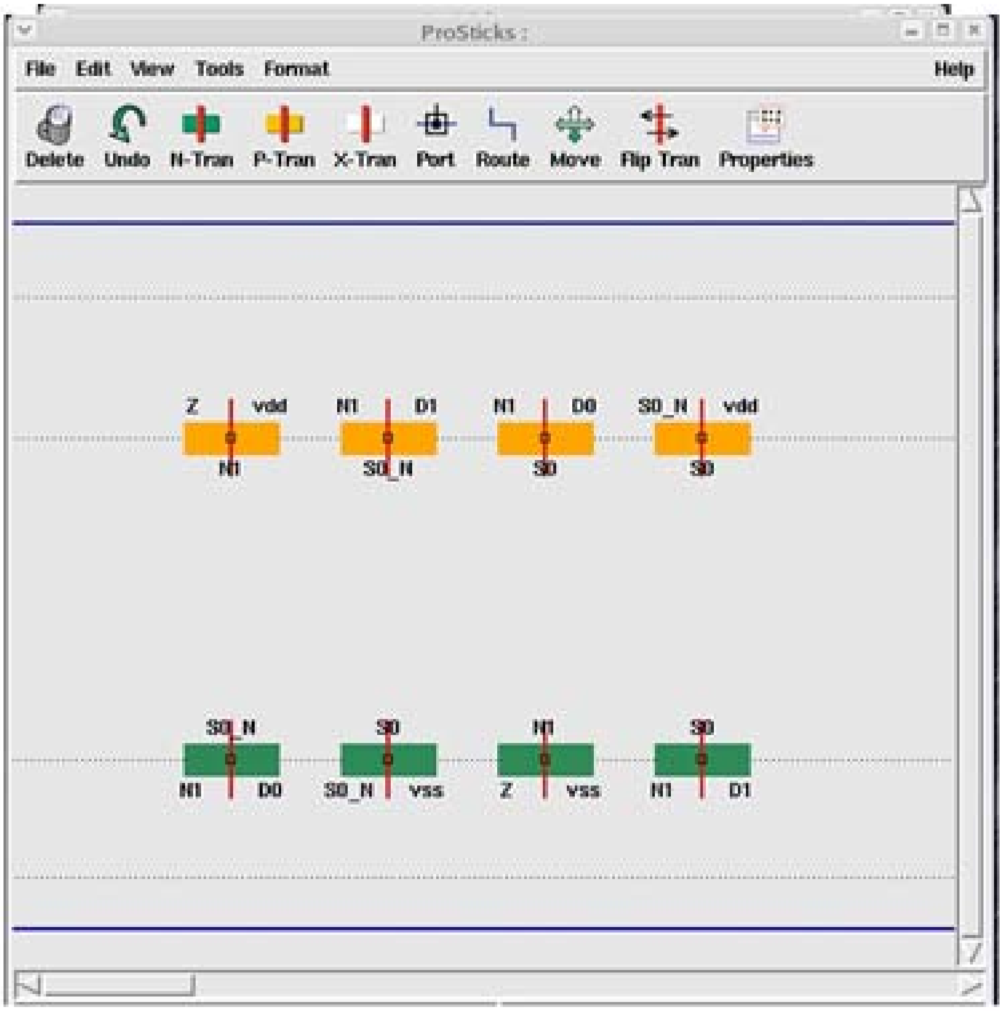


Figure 7: Add additional transistor rows to support stacked transistors

The layout shown in Figure 54 has two additional transistor rows, so we're ready to move the transistors into place. We start by going to the toolbar and selecting **Move** (also available from the **Edit** menu). Now whenever we click on an object in the workspace, we can drag it to another location.

The first move we'll make is to drop a transistor to the new transistor row. In Figure 8, an n-transistor has been moved to the lower transistor row. Why did we select this transistor? Because it needs to connect to the ground rail, so we'll minimize routing that way.

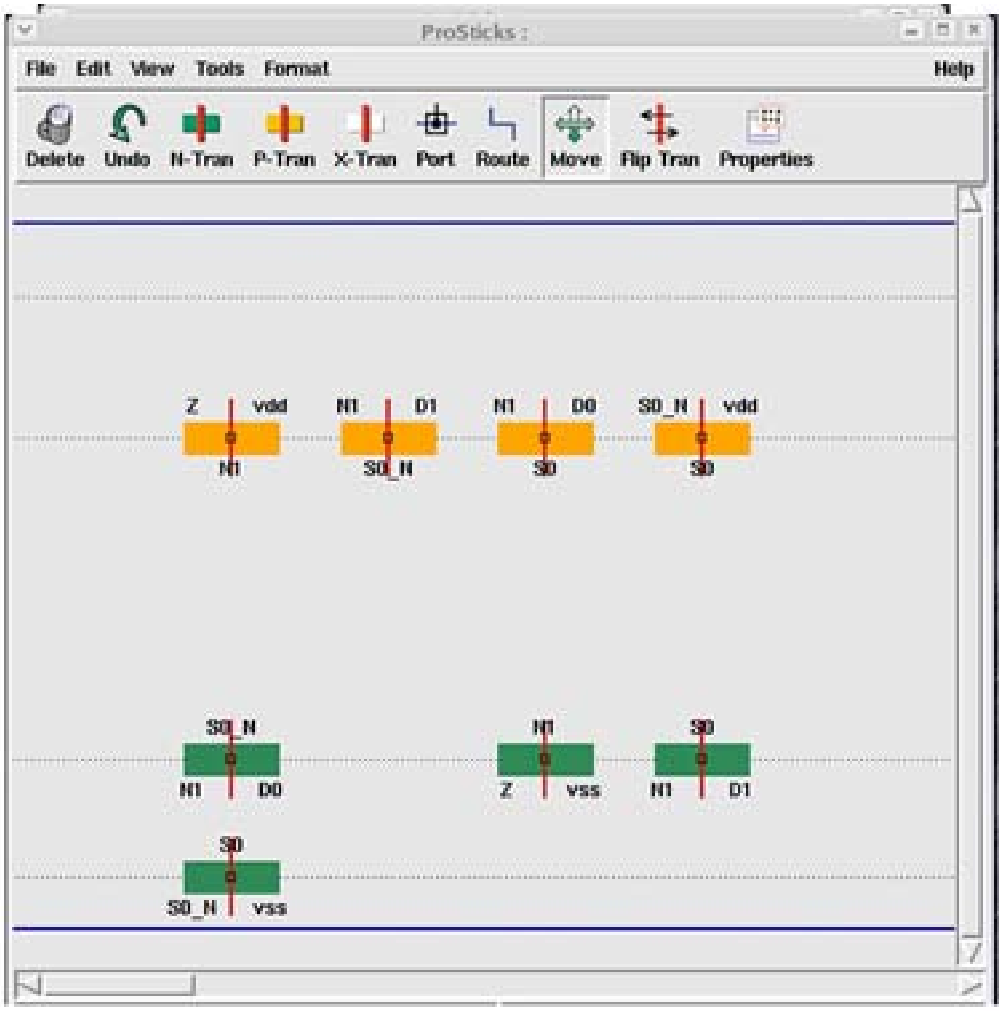


Figure 8: Start by moving a transistor to the new row

Next, we'll move another transistor down to the lower row. We have one that needs to connect to the ground rail, so that looks like a pretty good bet. In Figure 9, we've moved the transistor to be near the rail and to share diffusion with the transistor we already moved. Looks good so far!

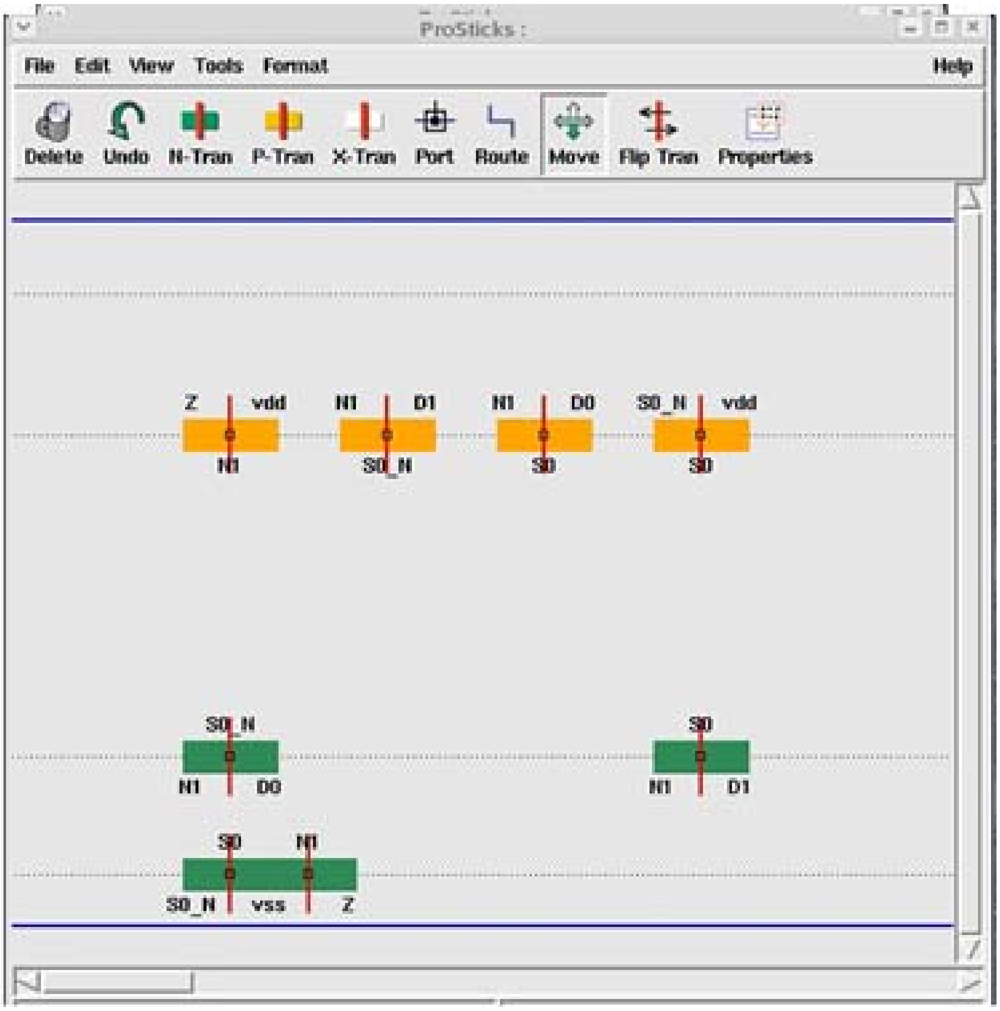


Figure 9: Move another transistor to share diffusion contacts

Notice that in Figure 8, the vss contact on the transistor we moved was on the right side of the transistor. How did it end up on the left?

Autoflipping is enabled by default, which means if the transistor you're moving can share diffusion with the node you drop it on, it will flip the transistor.

Now we'll consolidate the transistors on the upper n-row. The leftmost of these transistors has a contact N1 that could be shared with the other transistor. However, it's on the wrong side of the transistor. Since we won't be moving this transistor, autoflipping won't help us. Instead, we select Flip Transistor from the toolbar (also available from the Edit menu) and then click on the transistor, which produces the layout shown in Figure 10.

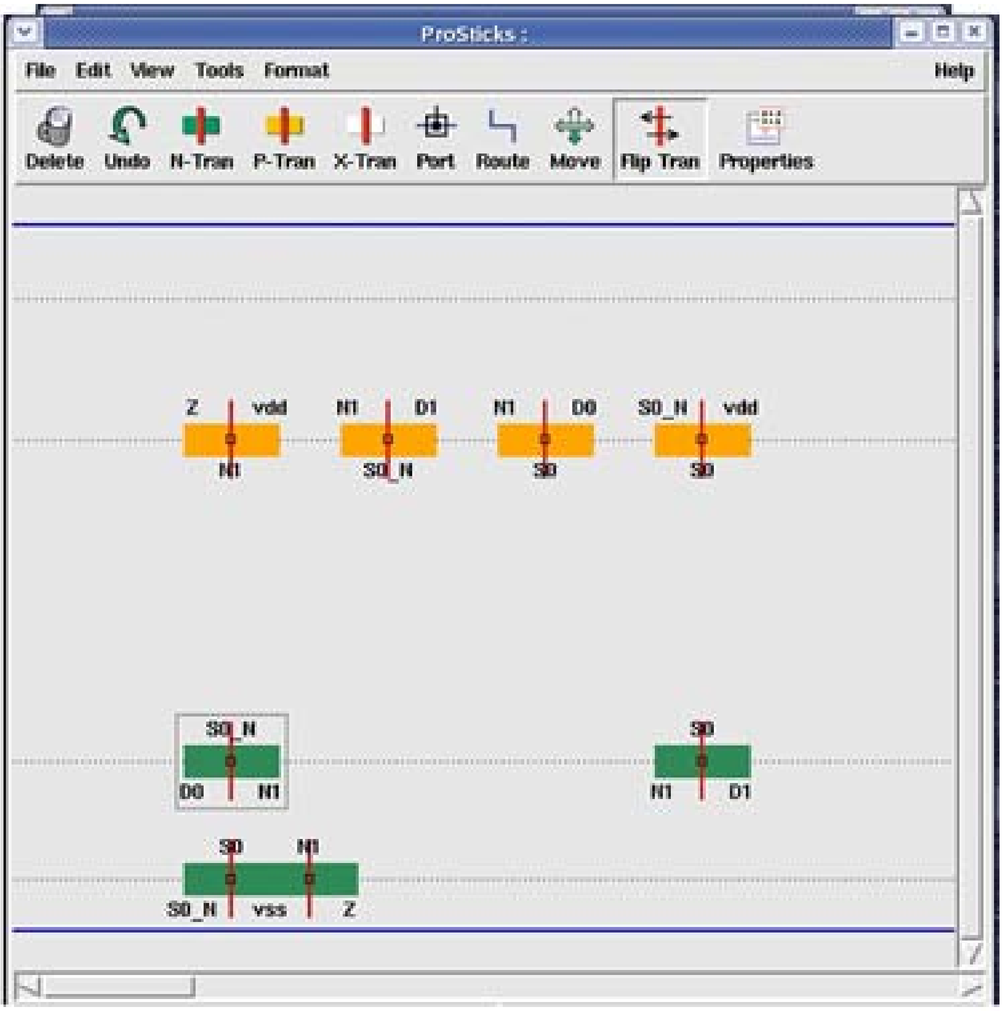


Figure 10: Flip a transistor to allow it to share diffusion contacts

Now we're ready to move that last n-transistor. We've got to re-select the correct tool by activating Move -- otherwise, we're just going to flip the transistor -- then we click on it and drag it over, as shown in Figure 11.

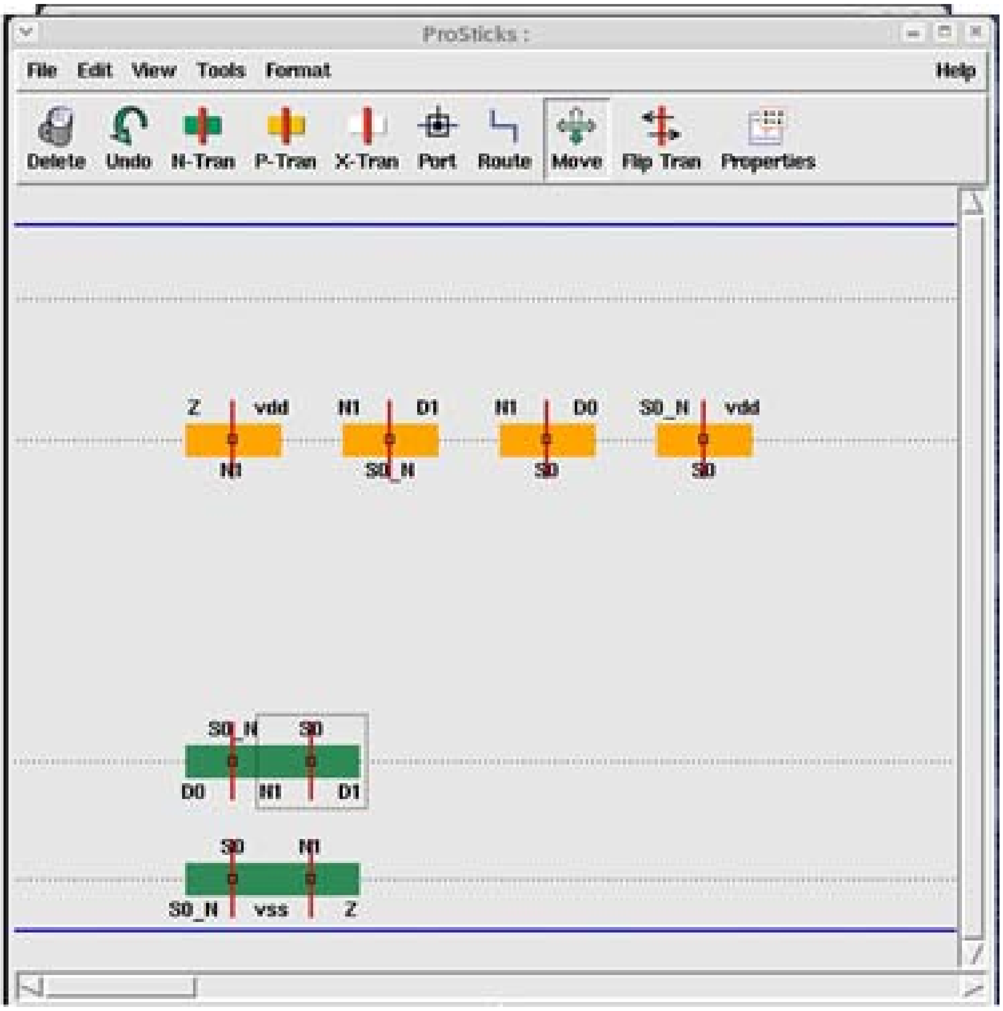


Figure 11: Move the last n-transistor into place

Now we're ready to move the p-transistors into place. We can start by moving the transistors that need to tie to the power rail.

In Figure 12, one of the p-transistors with a vdd node has been moved to the upper transistor row.

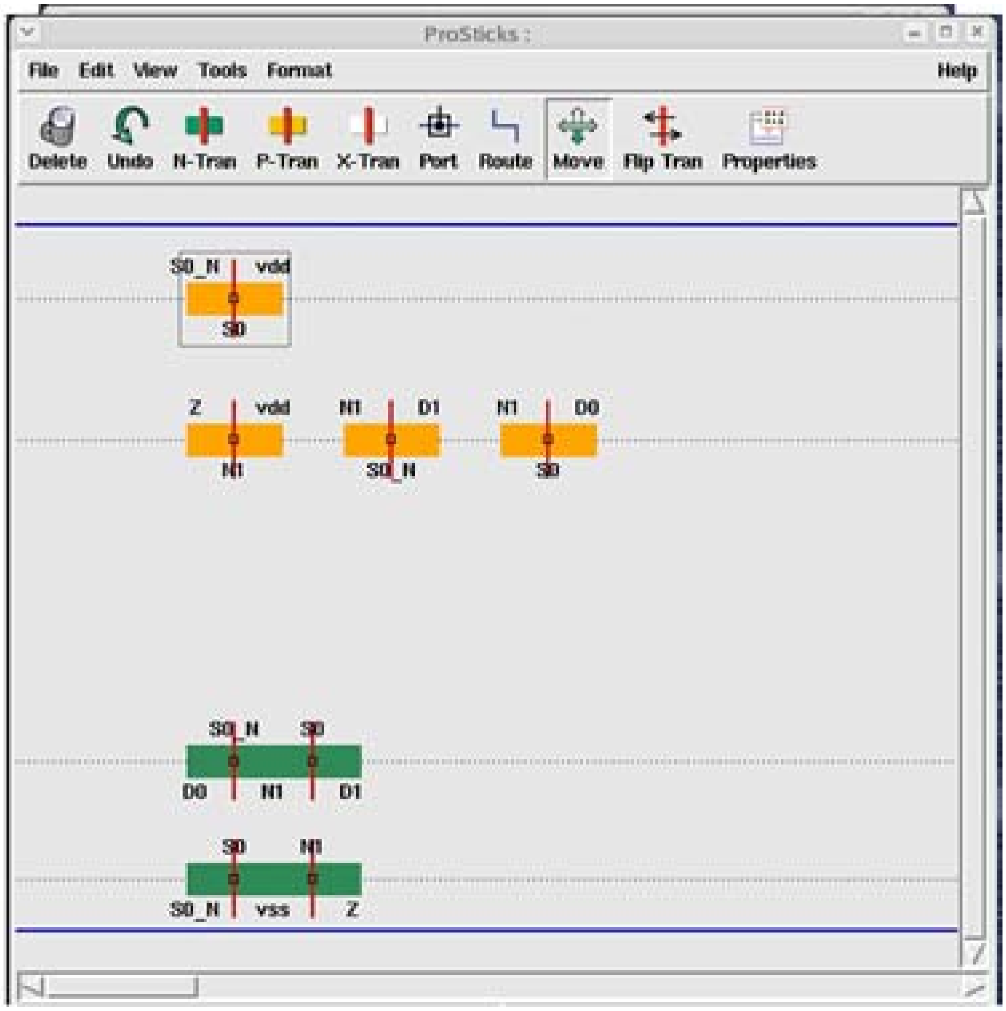


Figure 12: Move transistors that connect to the power rail closer

Now we can add the second p-transistor with a vdd contact. Notice that the nodes we want to merge are on the wrong sides of the transistors, but we can rely on ProSticks to autoflip the one we're moving so that everything goes well, as it does in Figure 13:

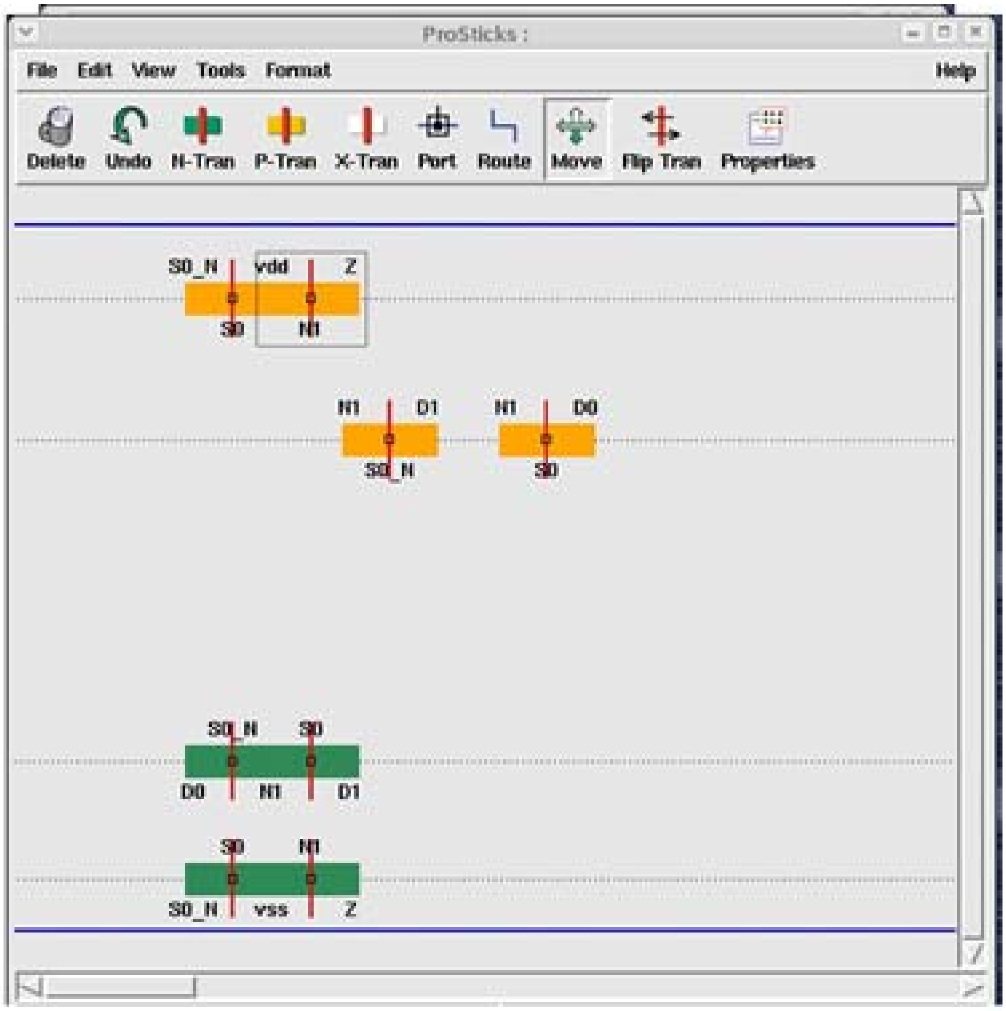


Figure 13: The transistor autoflips to share diffusion contacts

We're on the home stretch now. We move another transistor, as shown in Figure 14.

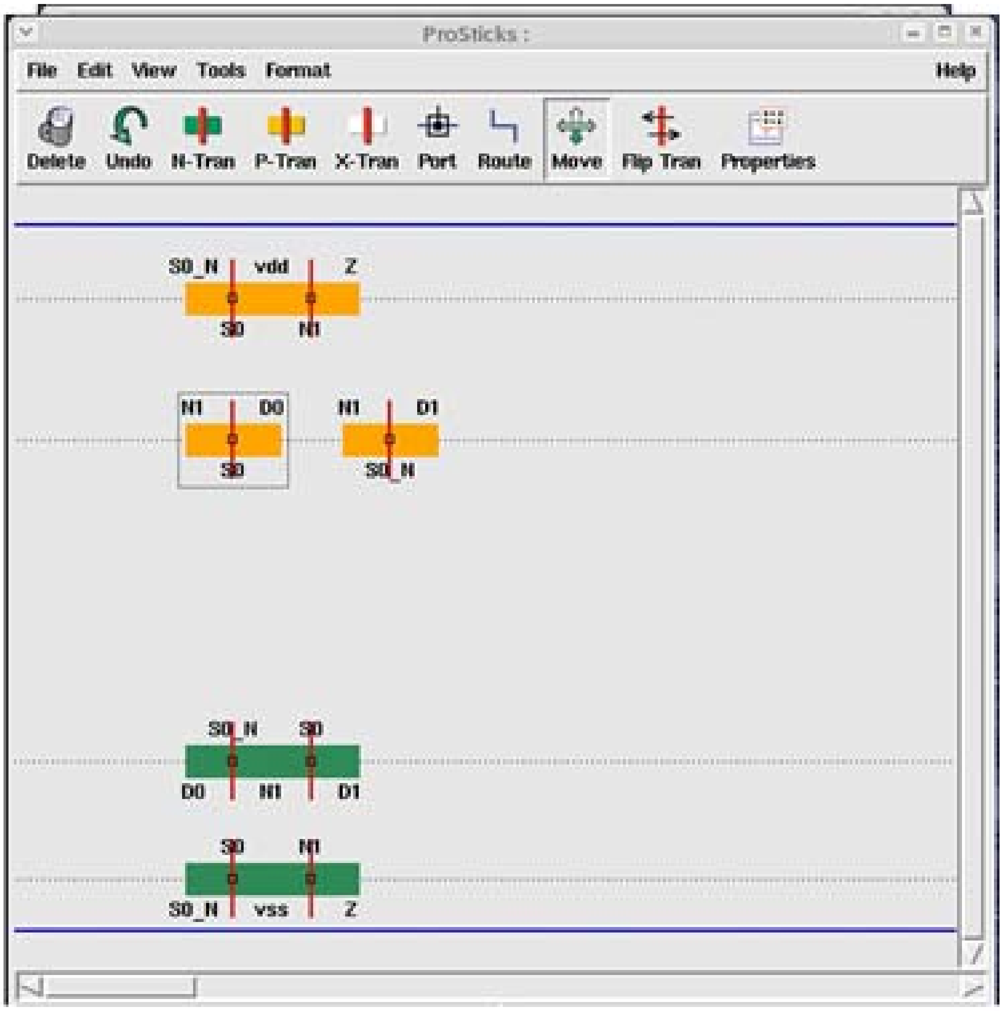


Figure 14: Move the transistor to line up with the others for easy routing

Notice that in Figure 14, the n- and p-transistor gates don't line up perfectly. We're attempting, instead, to line up the diffusion nodes. But if that's our objective, we'll need to flip the transistor we just moved. That will also allow sharing of the N1 diffusion contact, so it's a good move all around.

Once again, we go back to the toolbar or **Edit** menu and select **Flip Transistor.** Then we click on the transistor we just moved, which flips it to the orientation illustrated in Figure 15.

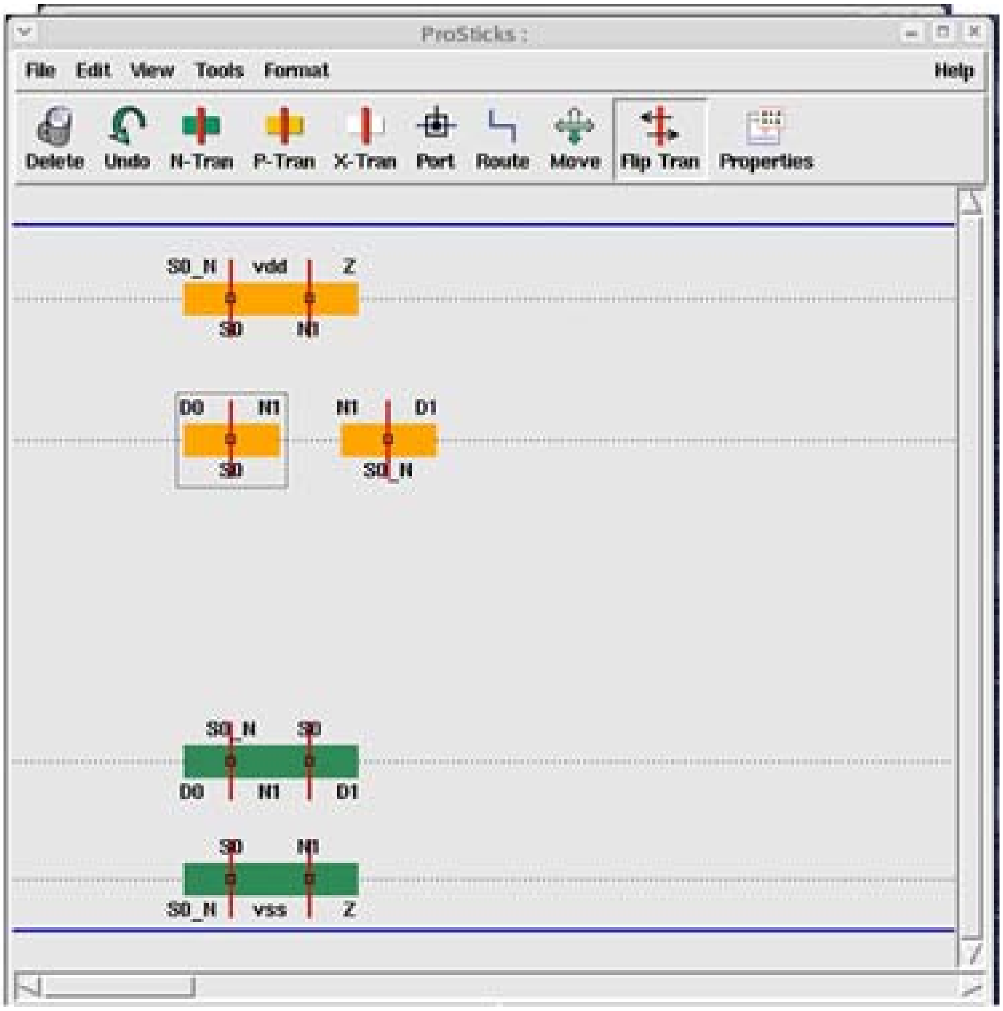


Figure 15: Flip the transistor to line up the diffusion contacts

Once we're done flipping the transistor, we again need to select the **Move** tool.

At last, we're ready to move the final transistor into place, as shown in Figure 16.

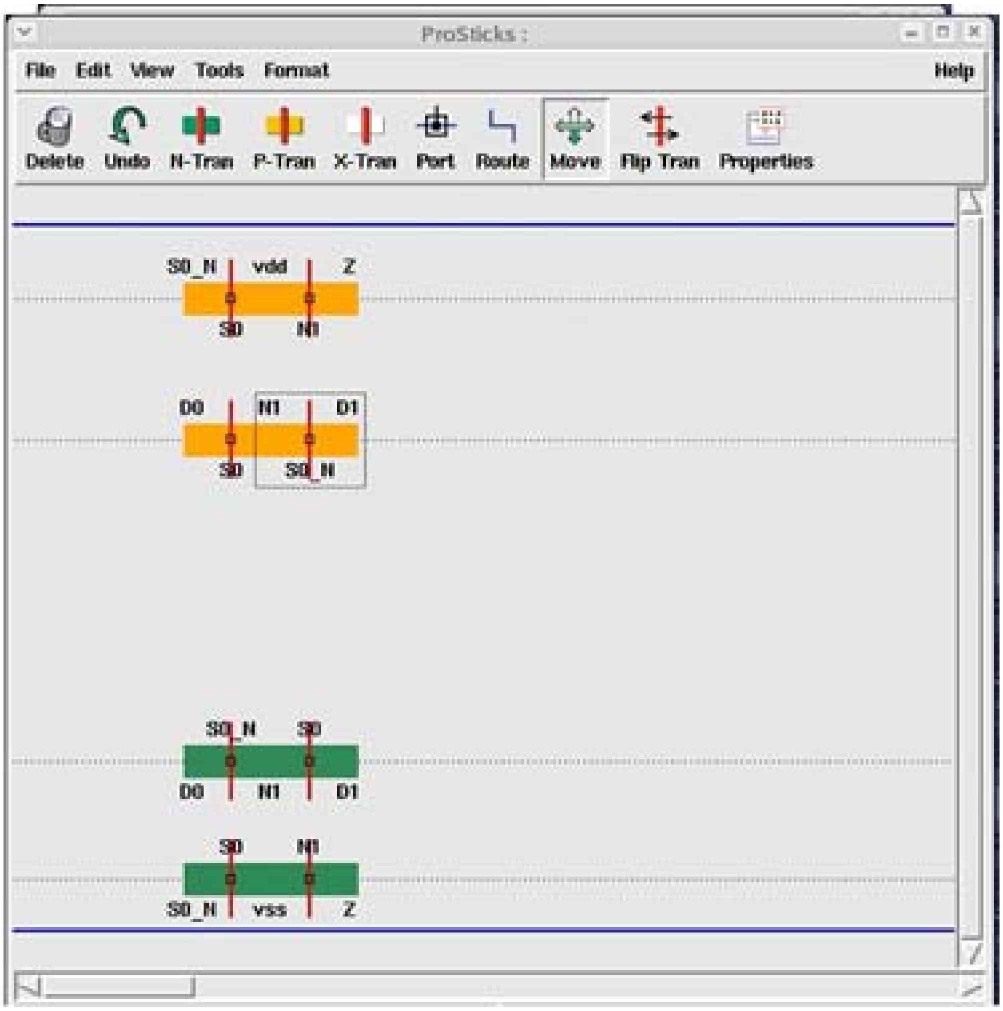


Figure 16: Move the final transistor into place

This looks good: we're sharing diffusion nodes as much as possible, we've got a straight shot for routing several of the diffusion contacts, and none of the nets has contacts spread all over the layout. We're ready to route!

To begin routing, we go to the **Edit** menu or toolbar and select the **Route** tool. We can route in any order we want to, but in this case we'll go with the easiest routes first: power and ground. We simply click on the vss diffusion node , then drag to the ground rail. Then we click on the vdd diffusion node and drag to the power rail. Figure 17 shows the cell after the power and ground connections have been made.

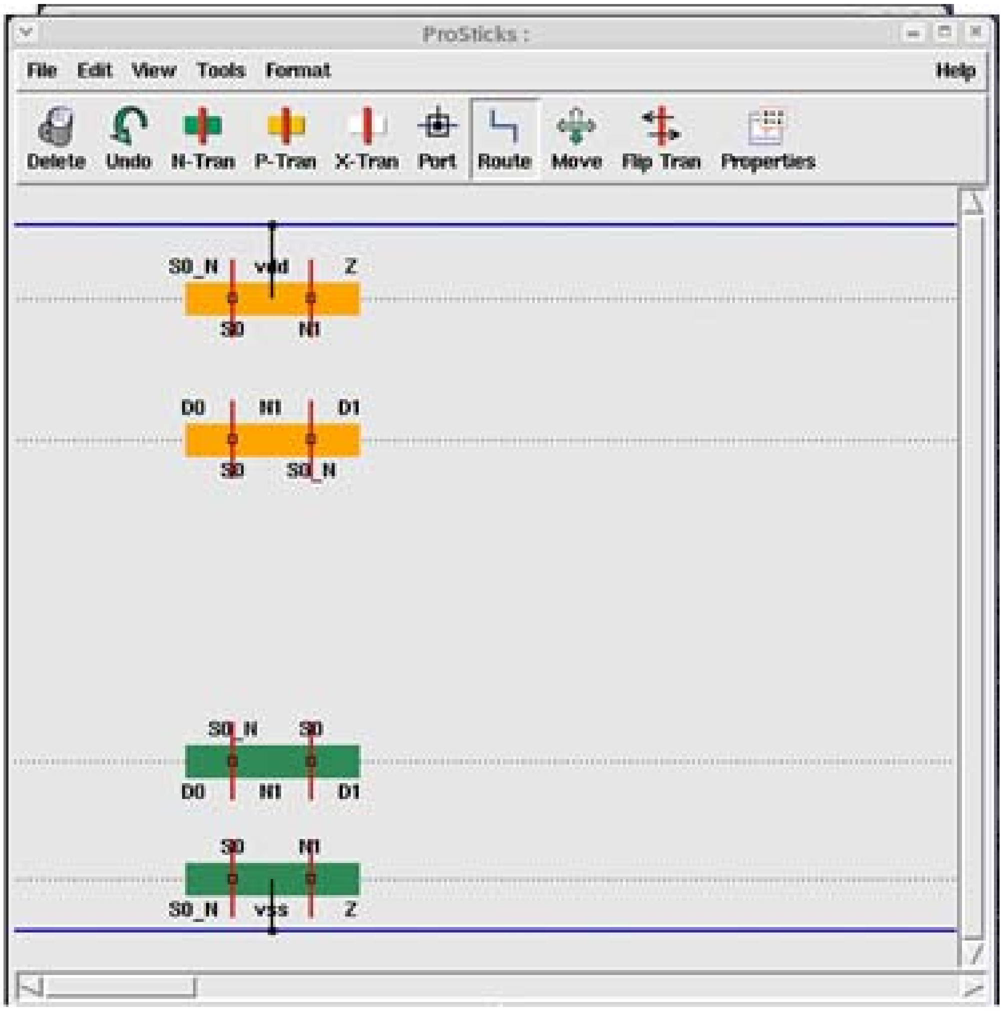


Figure 17: Use the route tool to connect power and ground

Now we're ready to move onto some additional nets. We took such care lining up the input data lines D0 and D1 that we might as well get them routed. To do so, we click on the D0 diffusion contact on one transistor and drag to the D0 diffusion contact on the other transistor. After we repeat this for the D1 contacts, we get layout that looks like Figure 18.

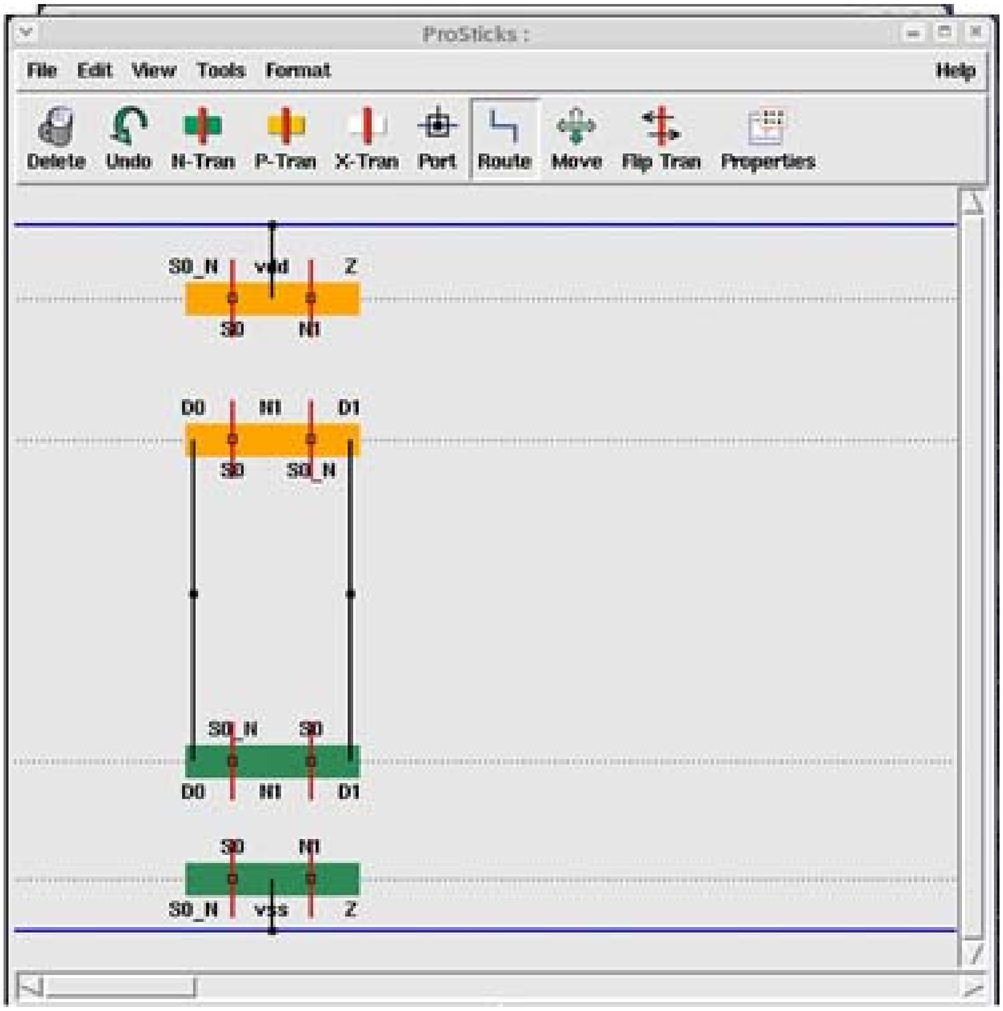


Figure 18: Route D0 and D1

Now we'll start some more complicated routing. We arbitrarily choose the SO_N net to route next (but we could really choose any net we find interesting). We click on the diffusion node on the bottom left transistor and drag to the gate of the transistor above it, creating the connection shown in Figure 19.

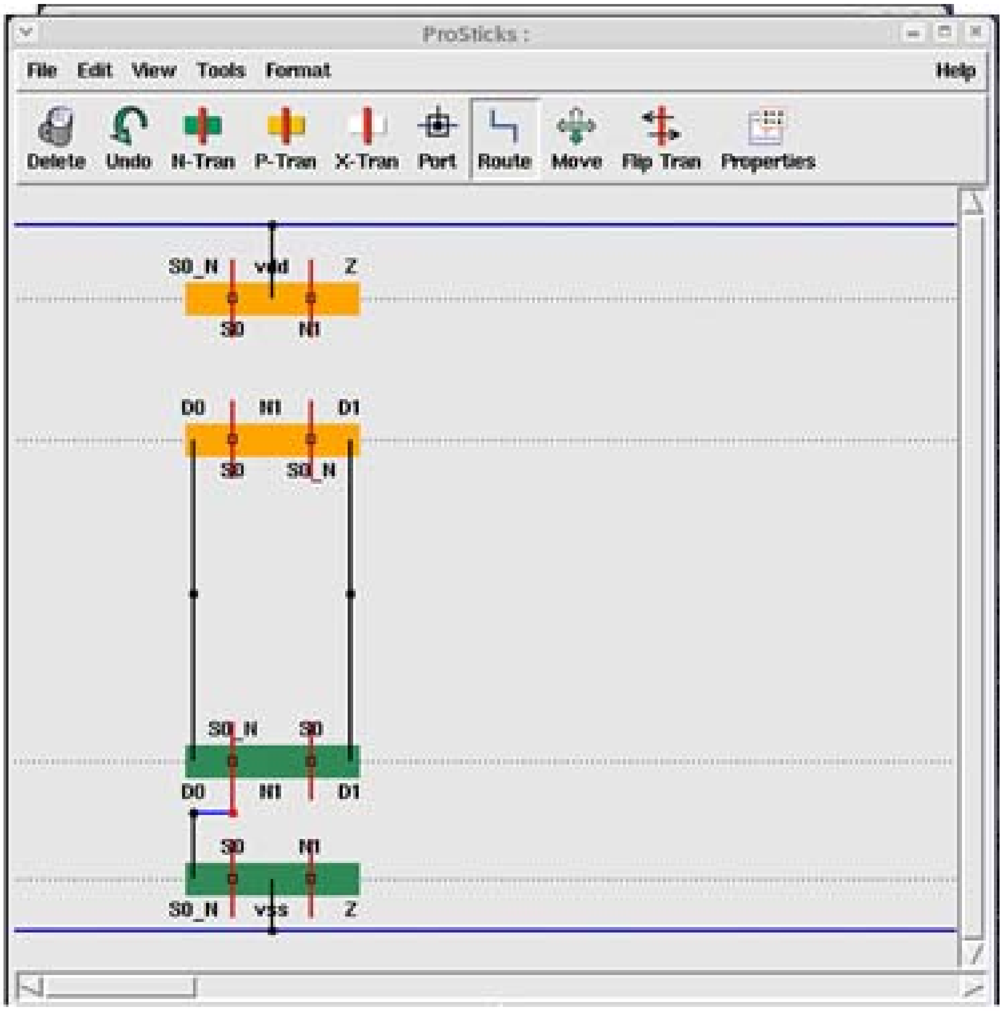


Figure 19: Begin routing the S0_N net

ProSticks automatically places the jog midway between the objects above and below. This looks tidy now but may cause some confusion when we route the S0 net later. It may be useful to use Edit>Move to push the jog upward or downward. We'll leave that until later.

Now we're ready to route the S0_N gates through channel 1 to connect the n- and p-type transistors. We click on one of the gates and drag to the other gate. Getting the hang of this? The result is shown in Figure 20.

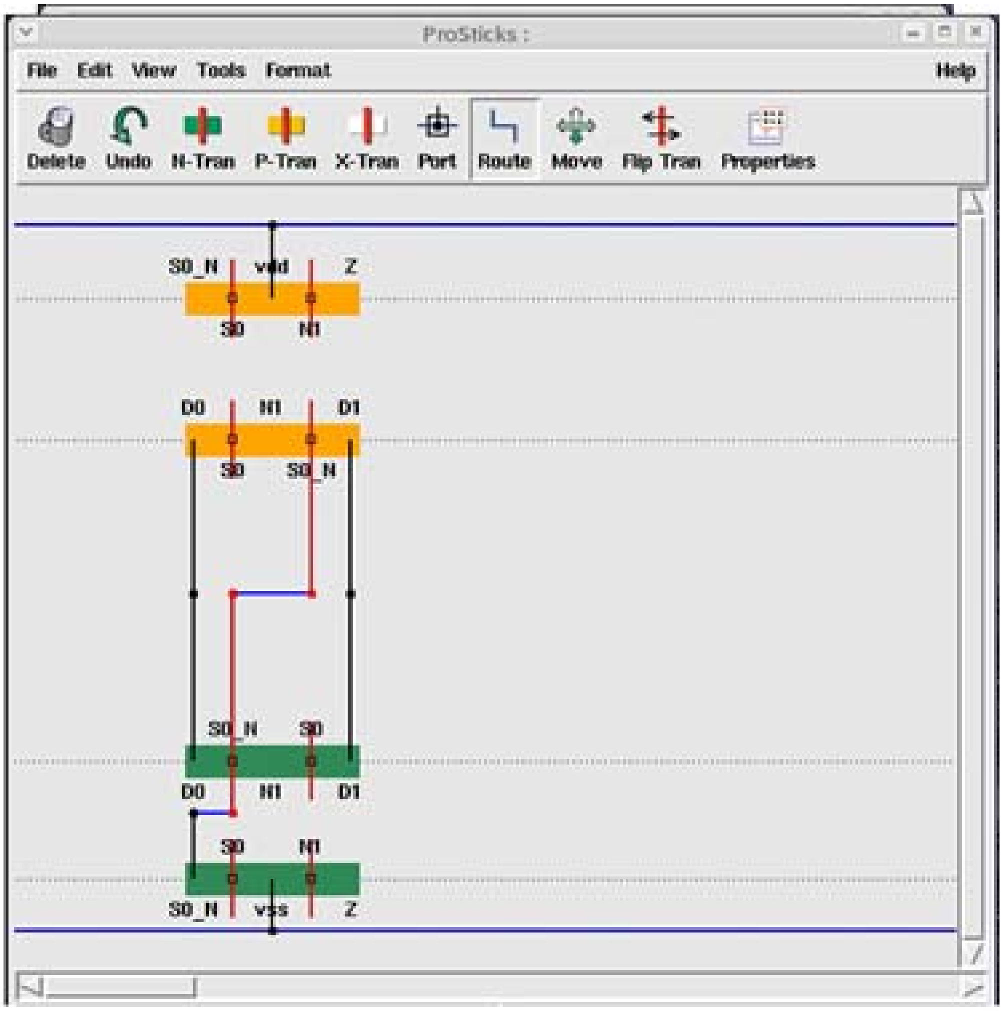
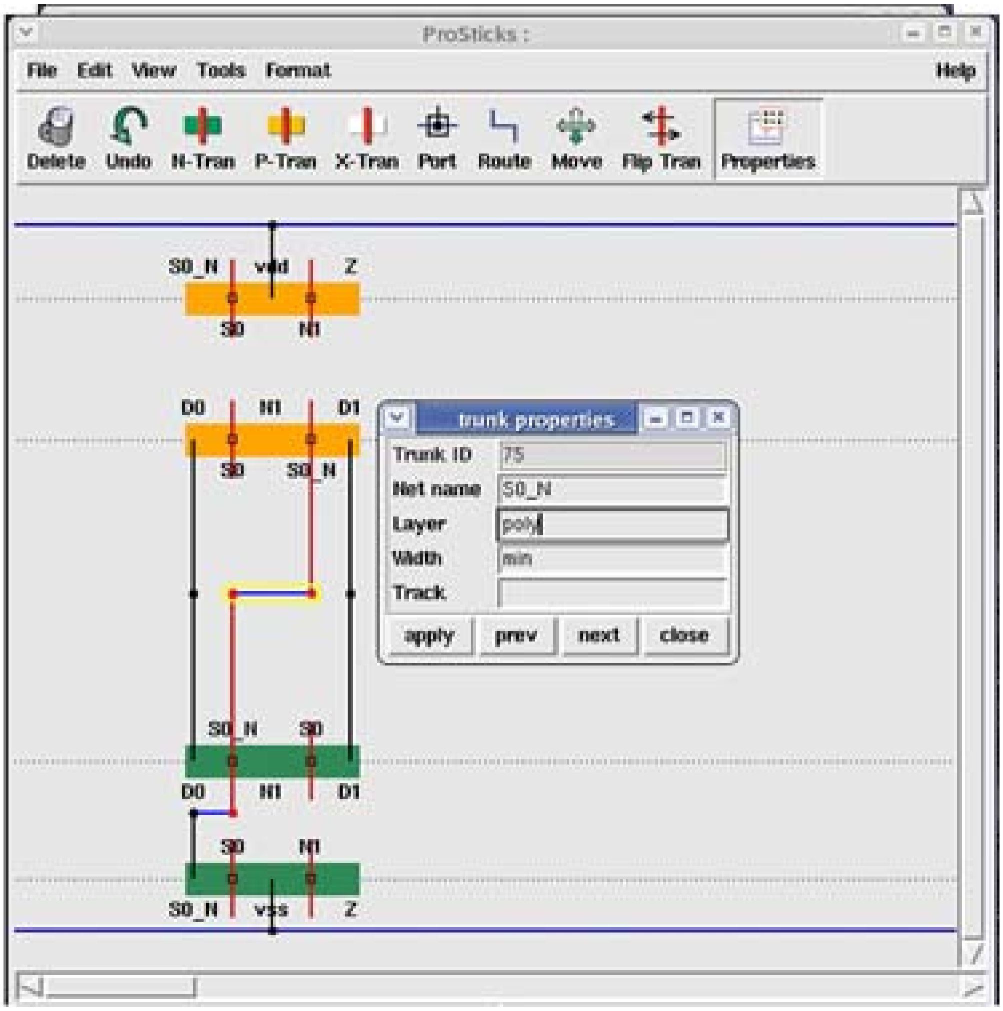


Figure 20: Route the S0_N gates

Notice that the jog wire is blue. This doesn't necessarily tell us what layer the jog is on, but we're pretty sure it's not going to be poly. We know there will be some congestion in this channel later on, so we might as well force this jog to the poly layer so we can try to avoid metal-2 altogether.

To manually select the layer used by this element, we go to the toolbar (or the **Edit** menu) and select **Properties**. Then we click the jog. In Figure 21, the jog is outlined in yellow, indicating that its properties are being displayed.

Notice that the **Layer** field has the entry poly. This field was initially blank; we entered the new value to push the jog into the layer we chose.



Trunk ID 75
Net name S0_N
Layer poly
Width min
Track

Figure 21: Use Properties to force the jog to poly

When we click **Apply** and **Close**, the jog will appear red and will have been designated to be routed in poly. (The M1 or default will be blue; everything else will be purple.)

We need to return to routing, so we go back to the **Edit** menu and select **Route** again, or hit that hotkey.

We continue routing the S0_N net by connecting the S0_N gate with the S0_N diffusion node. This completes that net's routing, as shown in Figure 22.

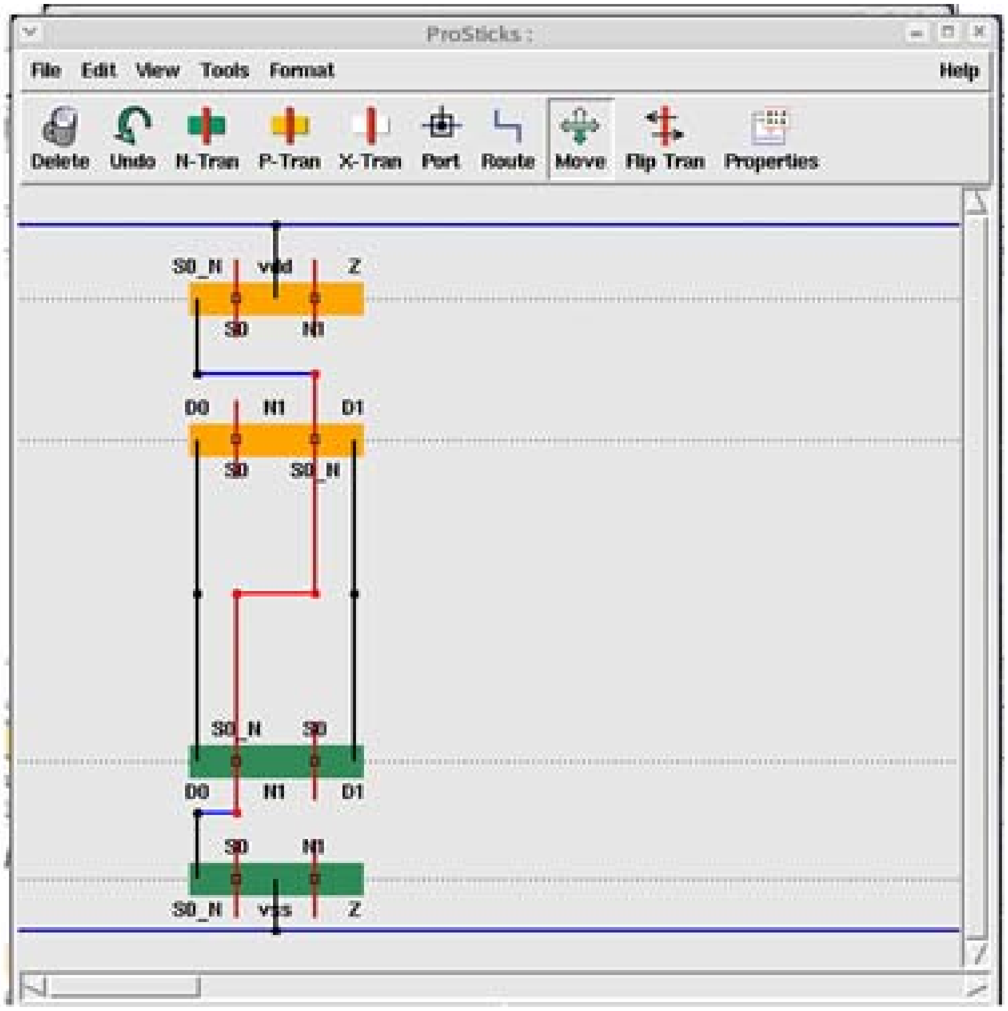


Figure 22: Complete the S0_N routing to the p-transistor diffusion contact

We're ready to move on to the next net. If we start in the upper left corner this time, we see that S0 is unrouted. So we click on the upper S0 gate and drag it to the S0 gate below it, producing the layout shown in Figure 23.

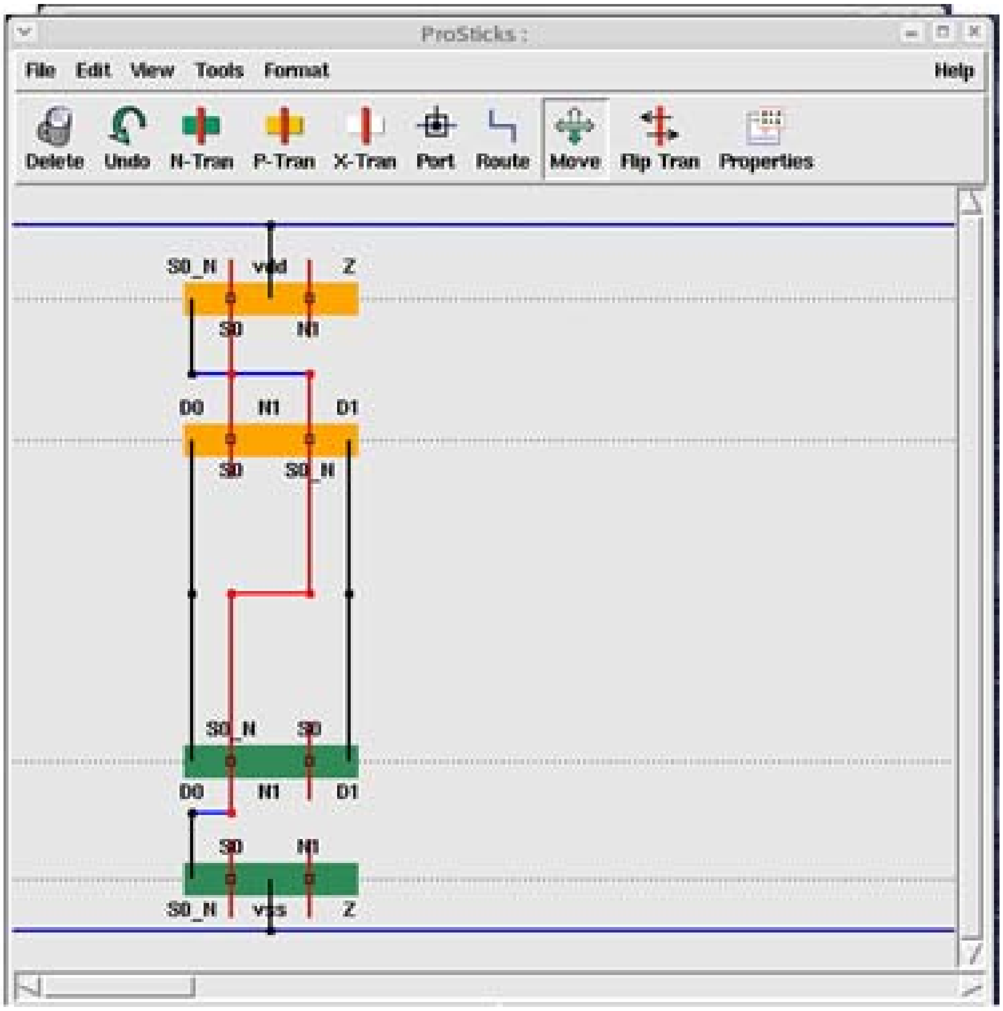


Figure 23: Begin routing the S0 net

This poly connection just crosses metal, so we're looking good.

Next, we click on the S0 gate again and drag to the upper n-transistor's S0 gate, which produces an odd-looking situation in our layout, as we can see in Figure 24.

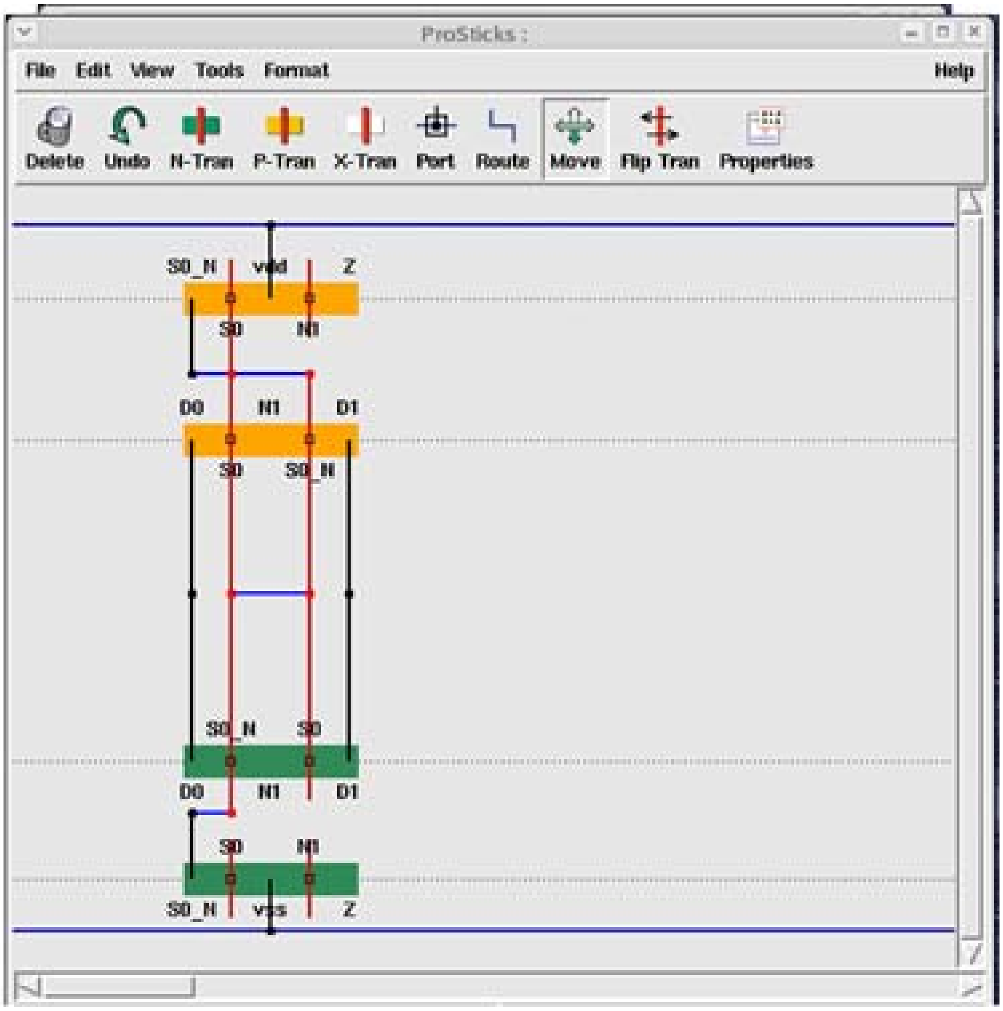


Figure24: Routing the S0 gates appears to have connected S0_N

Although S0 and S0_N appear to have been routed together, they do not really connect. Because ProSticks places jogs halfway between the objects above and below, it has lined up both jogs in the same place.

We can solve this problem by moving the S0 connections around. We select **Move**, and then alter the S0 and S0_N routing until we get the result shown in Figure 25.

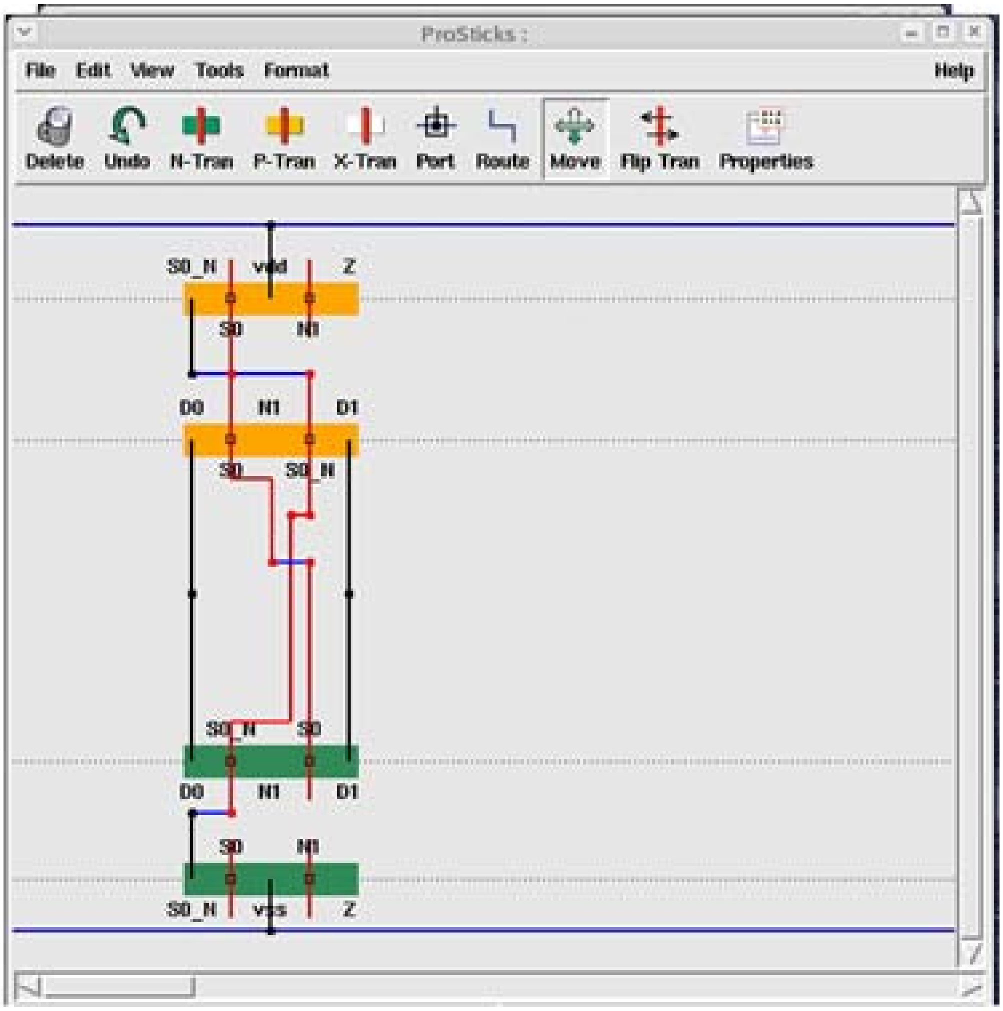


Figure 25: Move the overlapping jogs to produce clean layout

When we're satisfied, we continue by selecting **Route.** (Notice that the poly does not overlap, and there's only one layer of metal so far.)

All we have to do to finish routing the S0 net is connect the last two gates. We click on one gate, drag to the next, and we've got the layout pictured in Figure 26.

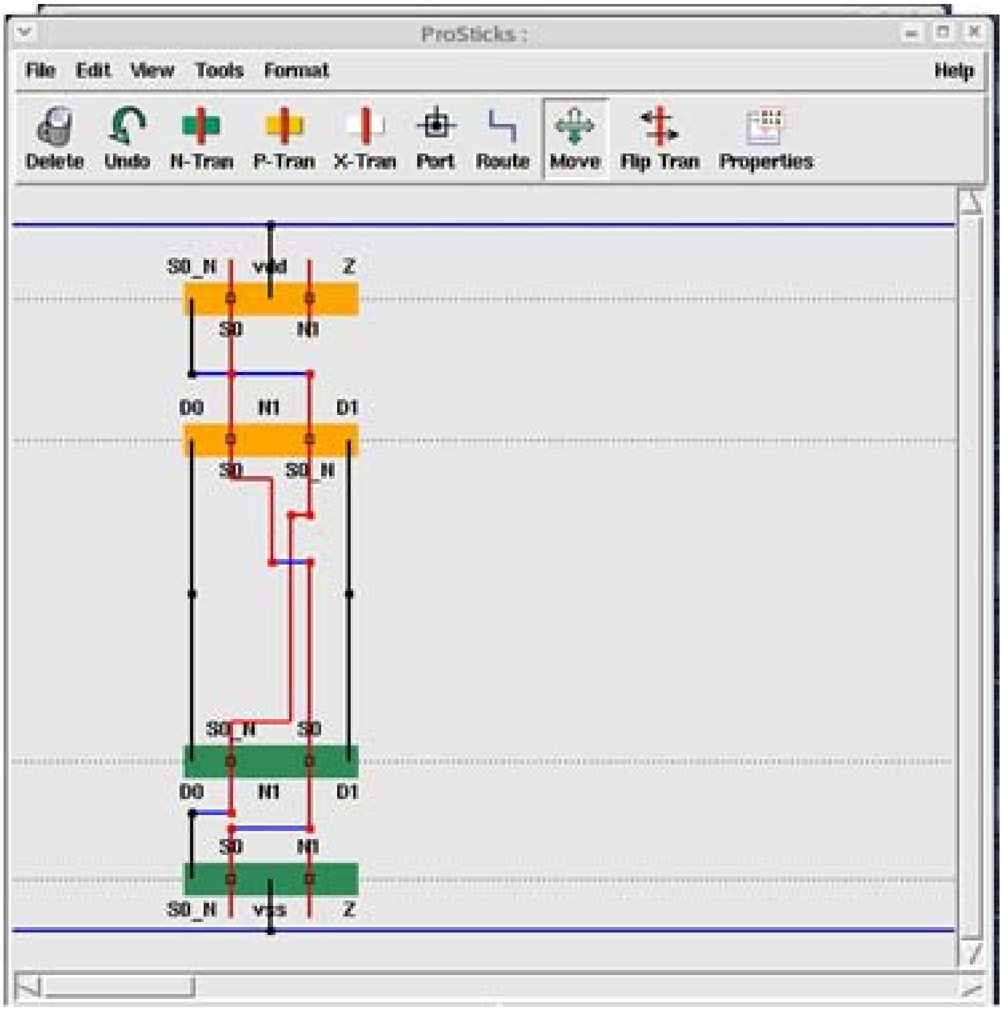
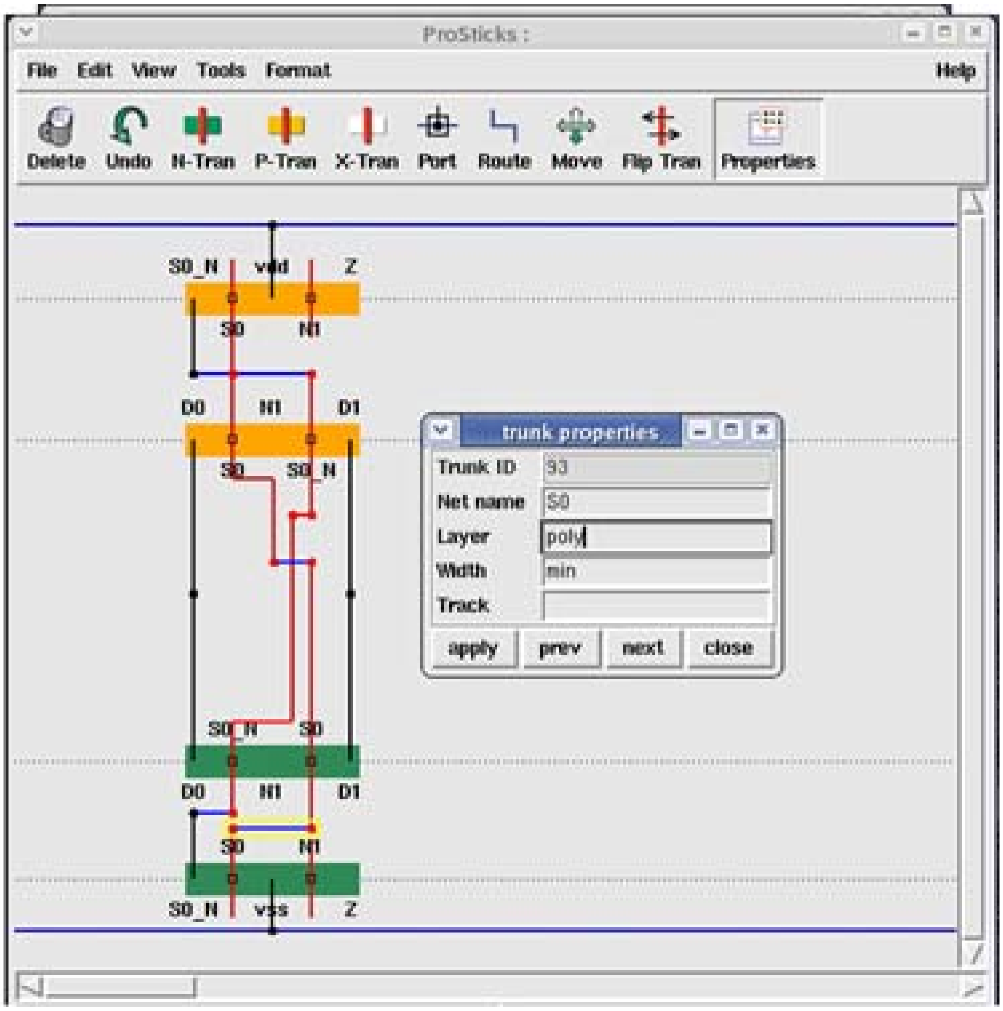


Figure 26: Connect the last two gates to finish the S0 net

Now, we've cheated a little bit in Figure 26. As ProSticks placed the new S0 jog, it appeared to connect with the S0_N jog. So we used Edit>Move to push the S0_N jog upwards a little bit, where it's not in the way.

Looking at this layout, though, we can see a problem. When we route the N1 gate on the lower n-transistor row to the poly contact on the row above it, we'll be using metal. Right now, the jog in S0 is metal, and since the two would be overlapping, we would be forced to employ metal-2. Now, we could arrange these connections to avoid this circumstance, but it will be easier to force the S0 jog into poly. If you want to avoid routing in poly for performance reasons, you might want to route differently.

As we did before, we'll go to the toolbar (or **Edit** menu) and select **Properties**. Then we click the S0 jog. In Figure 27, the jog is outlined in yellow, so we know its properties are the ones being shown.



Trunk ID 93
Net name S0
Layer poly
Width min
Track

Figure 27: Force the S0 jog to poly

We've entered poly in the **Trunk Layer** field; now all we have to do is click **Apply** and the change will take place. Then we click **Close** to quit the **Properties** window.

Since we're ready to resume routing, we select **Route** from the toolbar (or **Edit** menu) again. Since you're probably getting the hang of this by now, we'll do more steps at once. We route from the N1 gate on the lowest transistor row to the N1 diffusion contact on the upper n-transistor row, and then to the N1 diffusion contact on the lower p-transistor row. We accomplish each routing step by clicking on one point and dragging to the next, and we produce the layout shown in Figure 28.

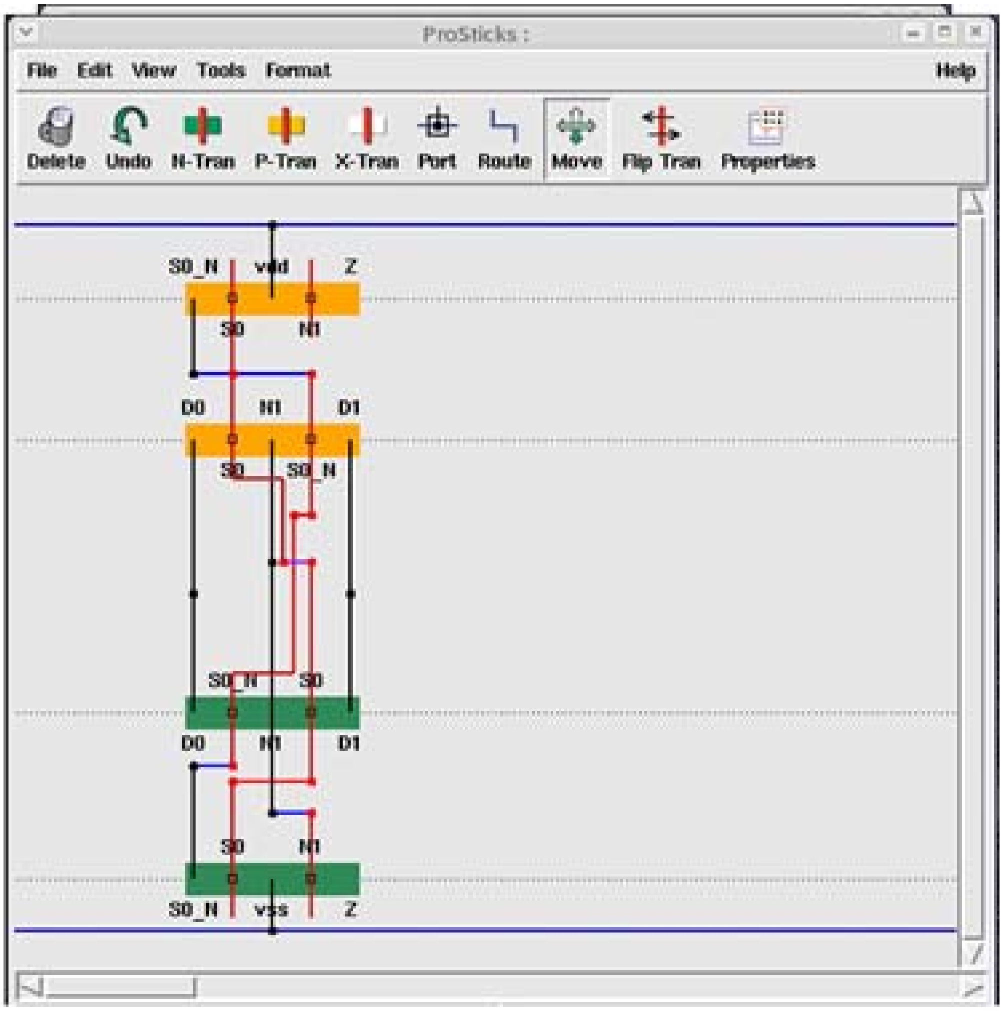


Figure 28: Route half the N1 net

To finish the routing of the N1 net, we'll have to get a little trickier. If we're not careful, we'll end up with metal crossing the metal jog in S0_N, and that will ruin our plans for a single metal layer. To avoid this, we'll route across the poly portion of the S0_N net. We'll connect the net as usual, by clicking on the contact and releasing on the gate. Then we use Edit>Move to push the S0_N jog wire aside and create the layout shown in Figure 29.

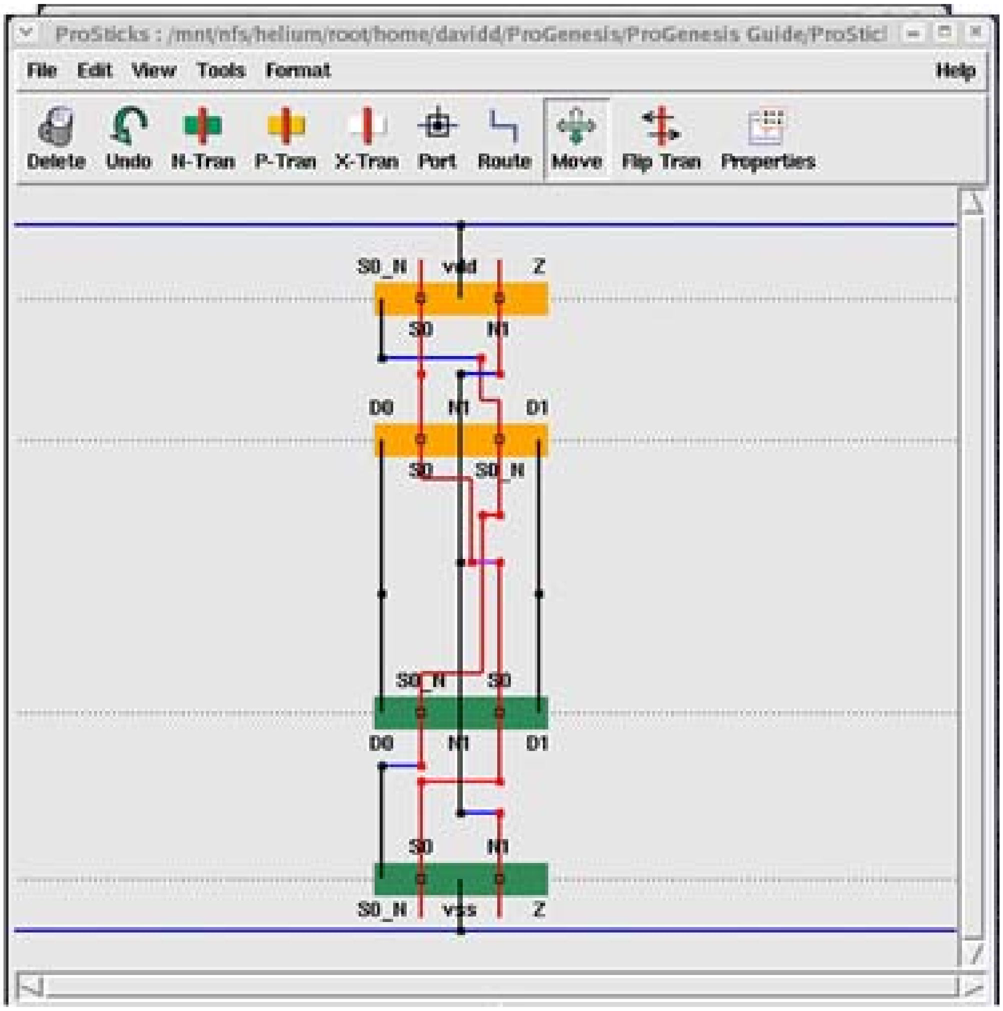


Figure 29: Finish routing N1 and move S0_N to avoid metal crossing metal

We're almost done now. The only remaining net to route is Z, the output feed. But notice that if we connect the two Z contacts directly, the feed will cross over two transistor rows. If you route between transistor rows without using a p- or n-transistor route point, you must use a passthrough or x-tran, which acts as a placeholder.

Adding these pass-through transistors is simple: select x-transistor from the Edit menu and click on the inner two transistor rows to place the transistors as shown in Figure 30.

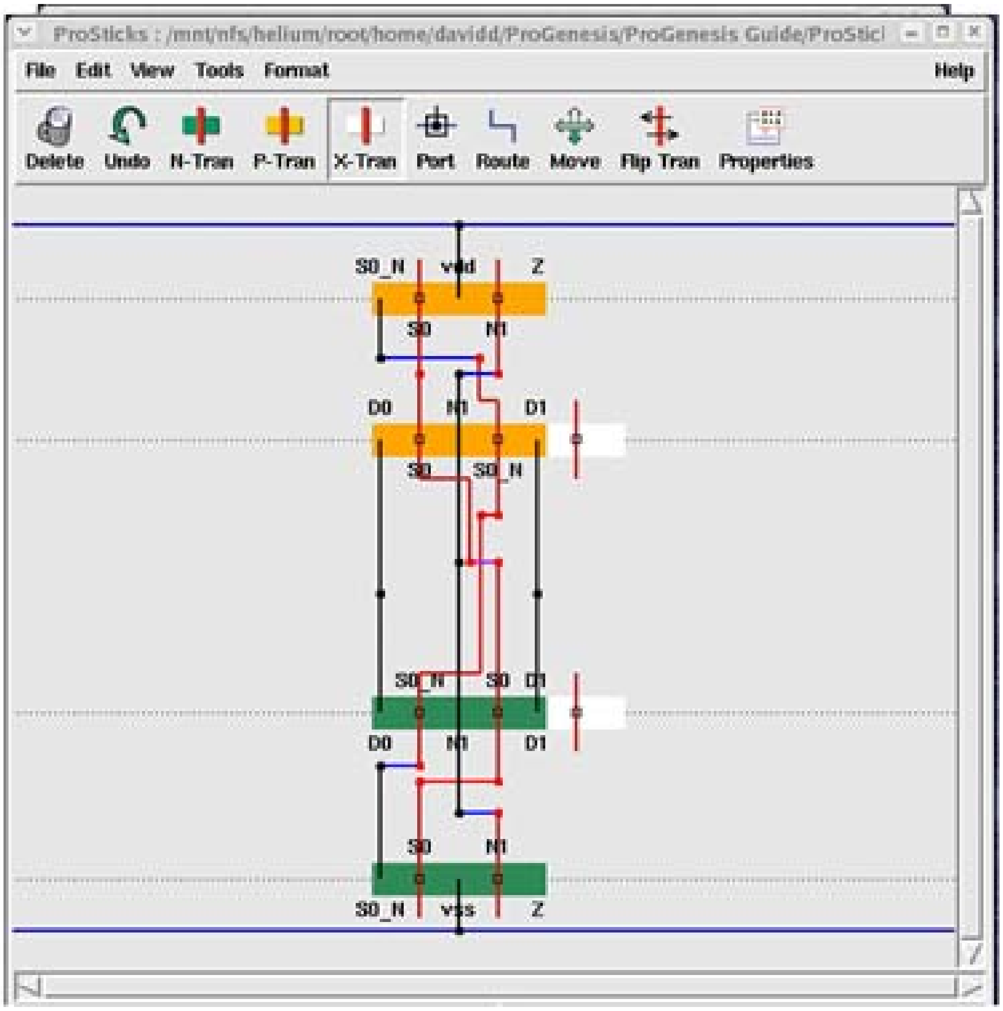


Figure 30: Use the x-transistor option to place pass-through transistors

Transistor rows can only be crossed through a transistor of some kind; these pass-through transistors are used in stacked cells like the one we're building to indicate where the feed wire will cross the extra transistor rows. (As an aside: it's possible to stack transistors without using pass-through transistors in certain cases, such as some two-input XOR cells. But most of the time, you'll be using Edit>X-transistor.)

Now that we have placed the pass-through transistors, we can route through them. We do this as we would with a p- or n-type transistor; we select Edit>Route, click on the Z diffusion contact, and then click on the unused side of the x-transistor. The result is shown in Figure 31.

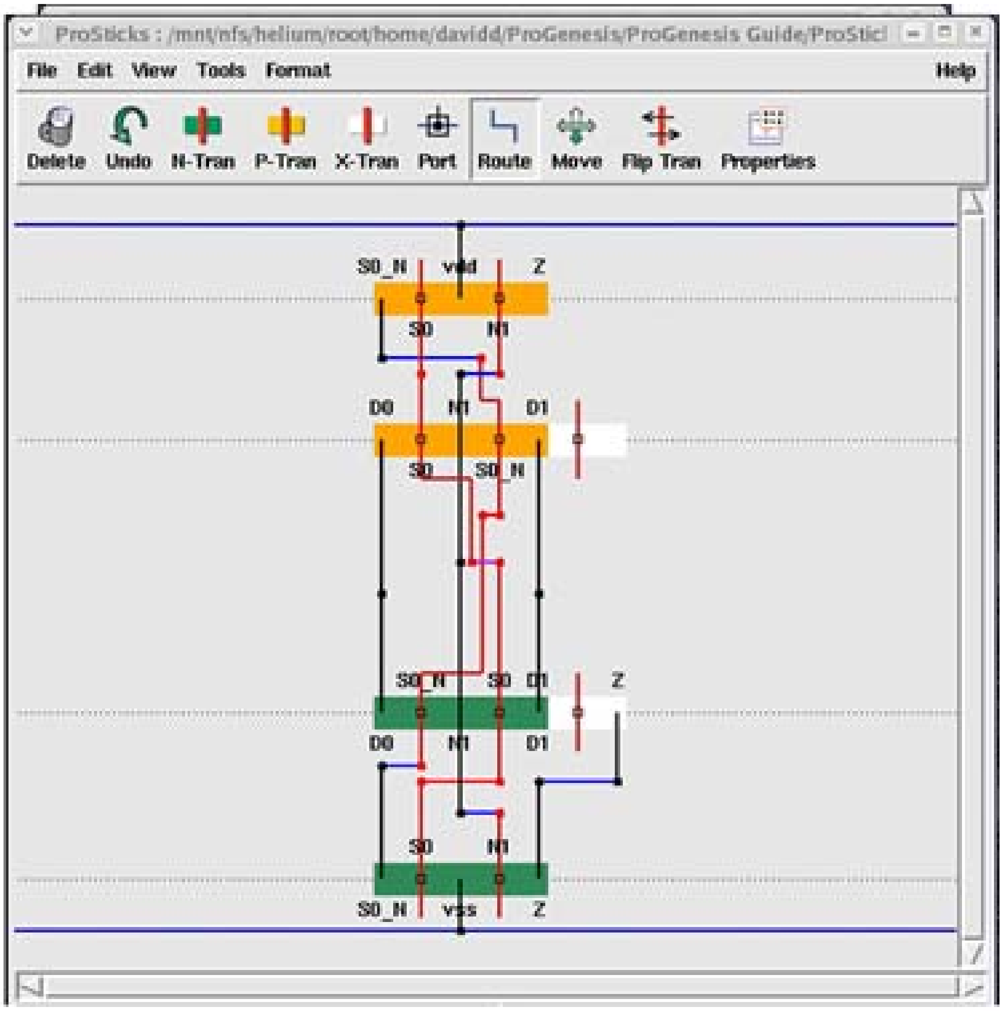


Figure 31: Route the Z net to a pass-through transistor

We can now finish routing Z. We click on the lower pass-through transistor's Z contact, then drag to the upper pass-through transistor and release on the right contact. Then we click on the upper pass-through transistor's right contact (which is now labeled Z because we've connected it to a named net) and drag to the topmost transistor row, where we release onto the Z contact. The resulting diagram is displayed in Figure 32.

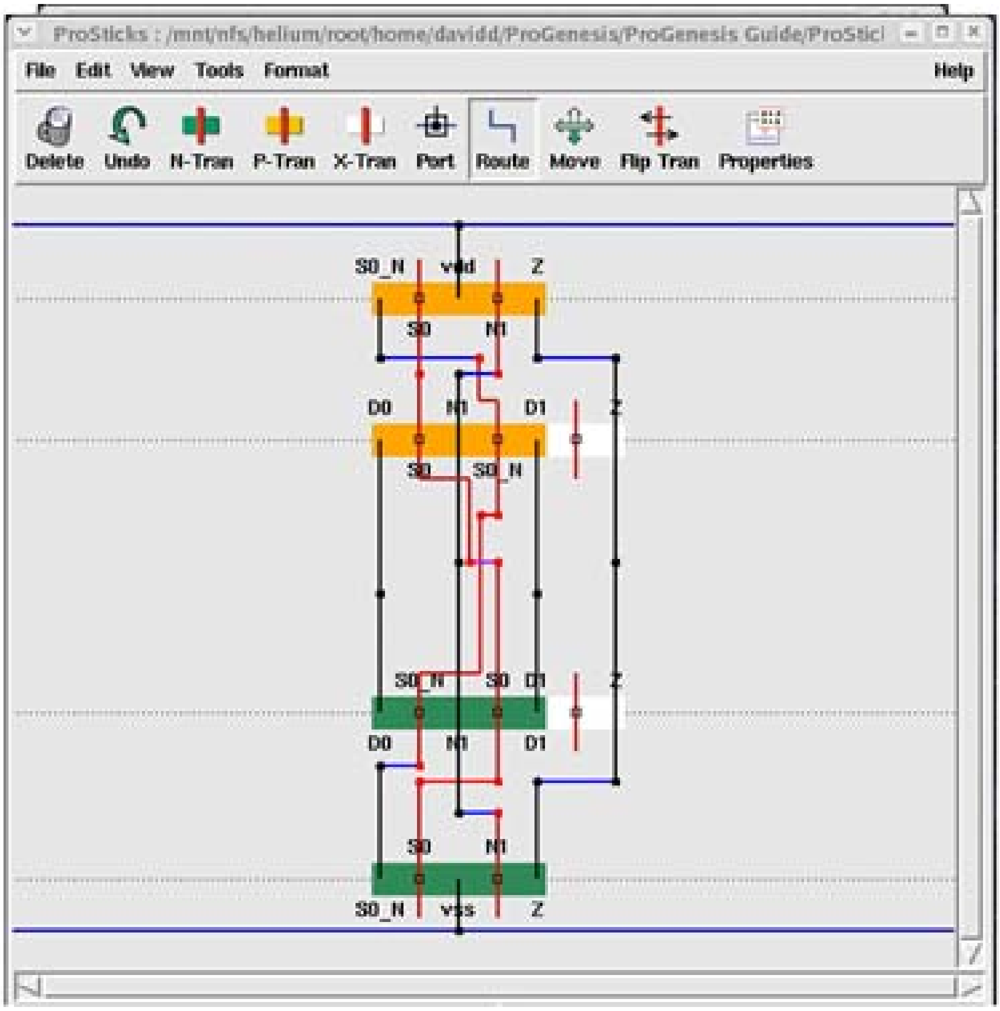


Figure 32: Finish routing the Z net

Ports are the only things missing from our layout. Adding them is easy. Go to **Edit>Port,** then click on a feed that will need a port, such as D0. See the example shown in Figure 33.

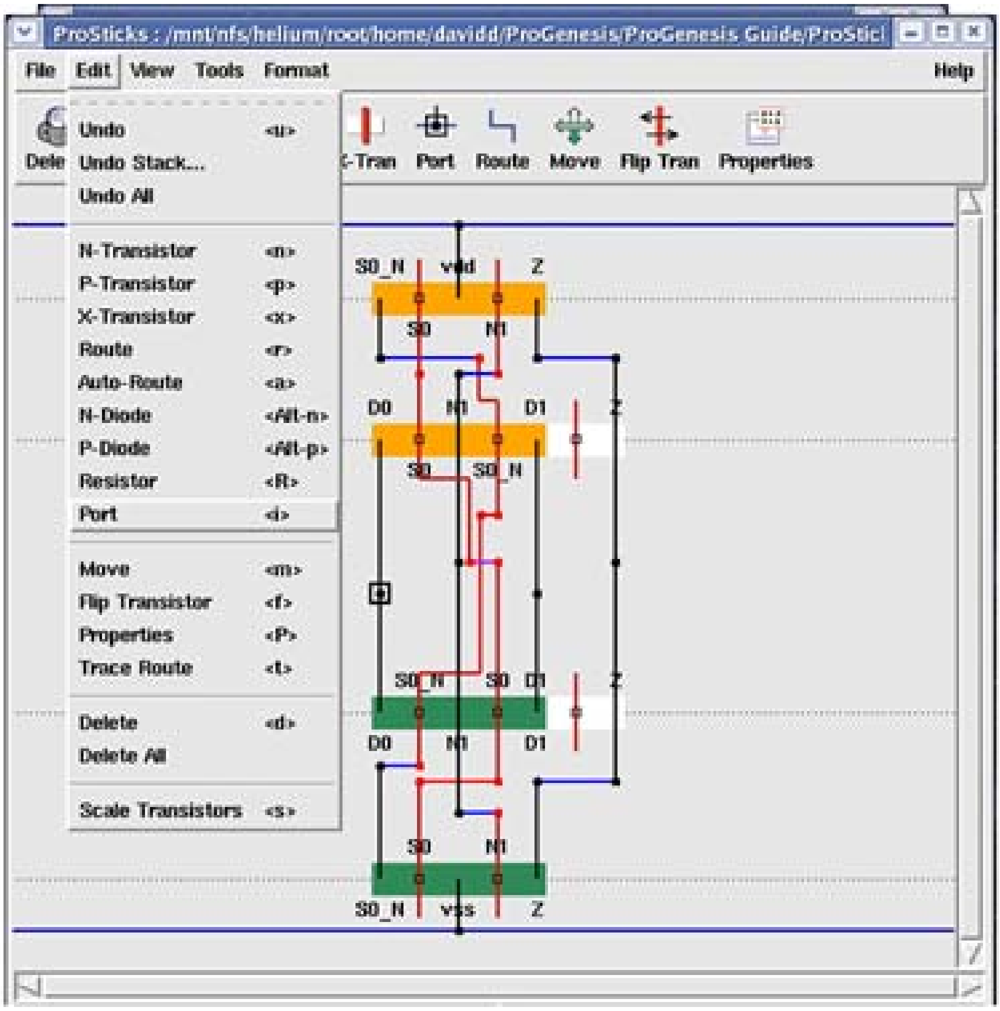


Figure 33: Use Edit>Port to add ports

To finish, use the same technique to add ports to D1, S0, and Z. This will produce the final diagram shown in Figure 34.

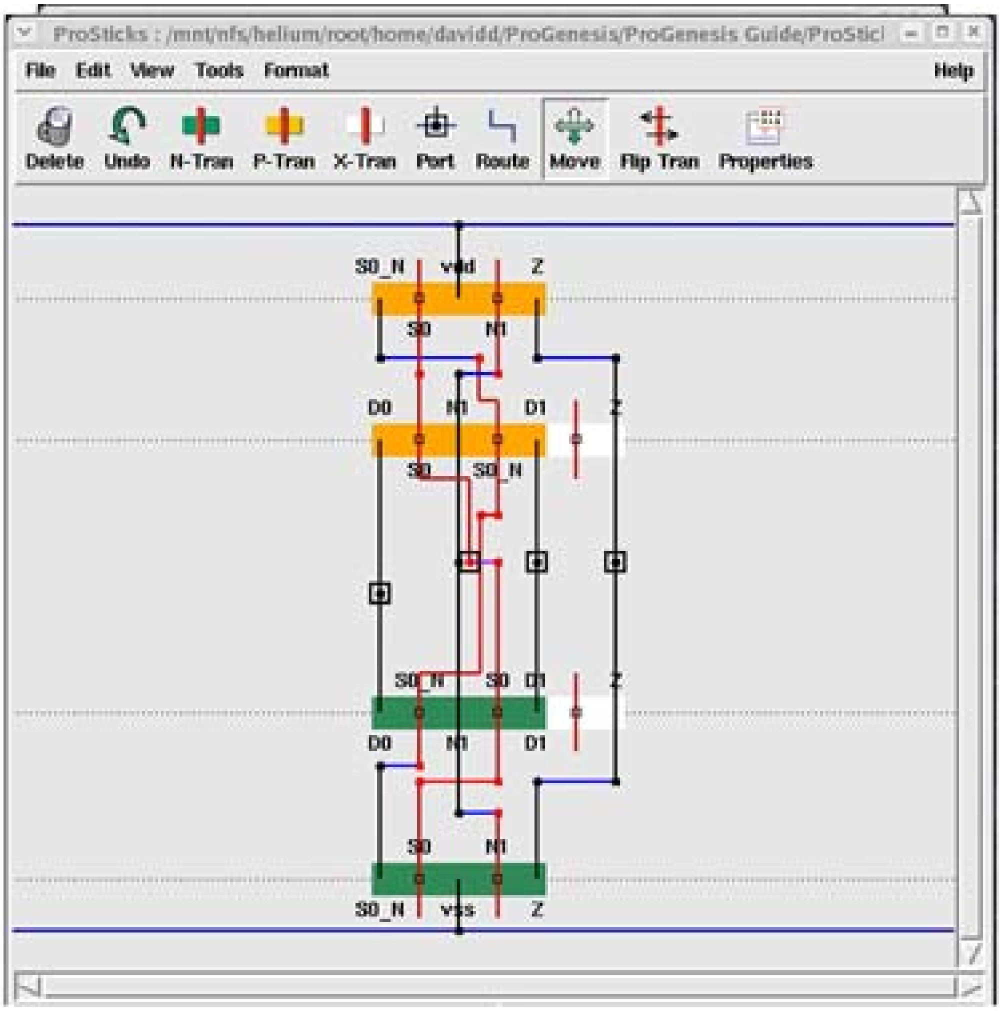
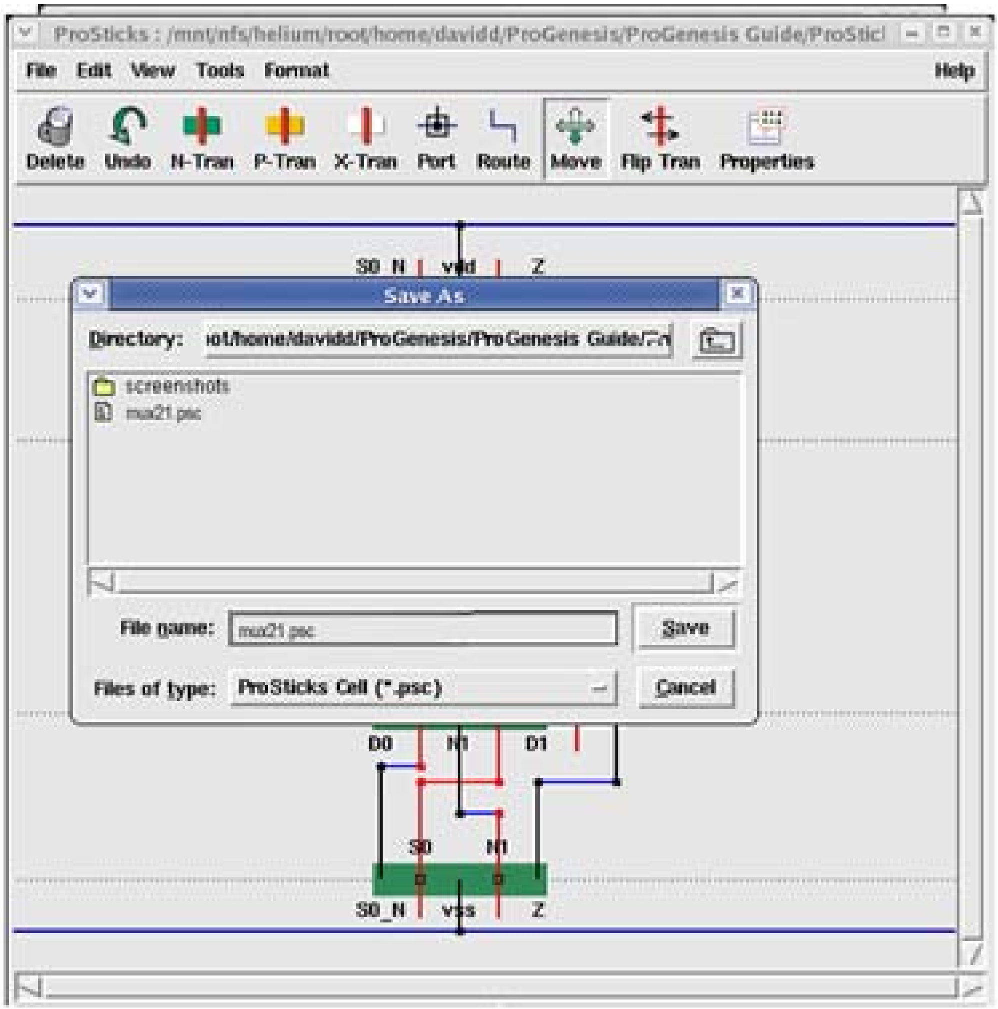


Figure 34: The final cell layout is fully routed and has all its ports

There are many other ways of using ProSticks to produce layout that would work for this generator, but we're satisfied with what we've got. We want to save our work in case we ever want to modify it, so we go to the File menu and select **Save**. Because this is our first time saving this cell, it opens the **Save As** window, which prompts us to enter a .psc filename, as shown in Figure 35.



ProSticks : /mnt/nfs/helium/root/home/davidd/ProGenesis/ProGenesis Guide/ProSticl
File Edit View Tools Format Help
Delete Undo N-Tran P-Tran X-Tran Port Route Move Flip Tran Properties
S0_N vdd Z
Save As
Directory: ›ot/home/davidd/ProGenesis/ProGenesis Guide/
screenshots
mux21.psc
File name:
Files of type:
Save
Cancel
D0 N1 D1
S0 N1
S0_N vss Z

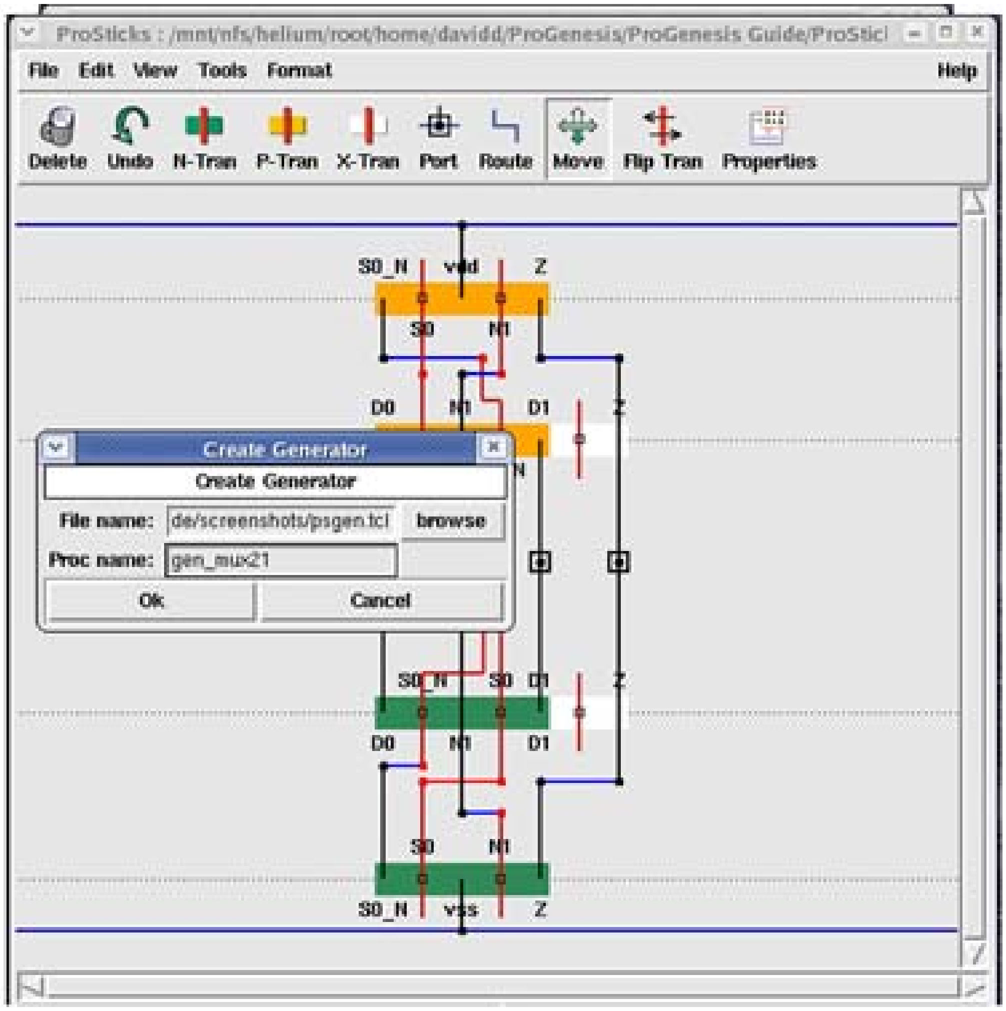
File name: mux21.psc

Files of type: ProSticks Cell (*.psc)

Figure 35: Use the Save As dialog to save the layout for the first time

We're going to save the file as mux21.psc, so we enter it and click **Save**. If we find that there's fine-tuning to do later, we can always reopen this file and modify the layout we've created.

Finally, we'll produce a generator by going to the **File** menu and selecting **Create Generator**. A generator file is a representation of the topology that is used by progen to create the final layout. We see the window pictured in Figure 36.



ProSticks : /mnt/nfs/helium/root/home/davidd/ProGenesis/ProGenesis Guide/ProStic

Create Generator

File name: de/screenshots/psgen.tcl

Proc name: gen_mux21

Figure 36: Output a generator file using the Create Generator option

We've already entered the filename we want to use for the generator, mux21.tcl. The default entry, psgen, will work, but we want to know what the generator creates even if we misplace it. We've also entered a more useful procedure name than the default, which is also psgen. The gen_mux21 procedure name will let us know that its function is to call the two-to-one mux generator. If we click **OK**, the generator is created as specified, and we're finished.

Example 2: Manually Created

Creating generators from netlists is easy. But how about creating a generator completely from scratch? That's easy, too. In fact, most of the steps are the same as working from a netlist; you just have to add the transistors yourself. In this section, we're going to create another generator from scratch, and this time we'll run ProGen and take a look at the results. We'll use a different type of cell so that you have another example.

We begin by starting ProSticks with an empty workspace. If you've finished a previous cell, you can use **Edit>Delete all** to remove all the transistors, routers, and diodes from the workspace. The empty workspace is shown in Figure 37.

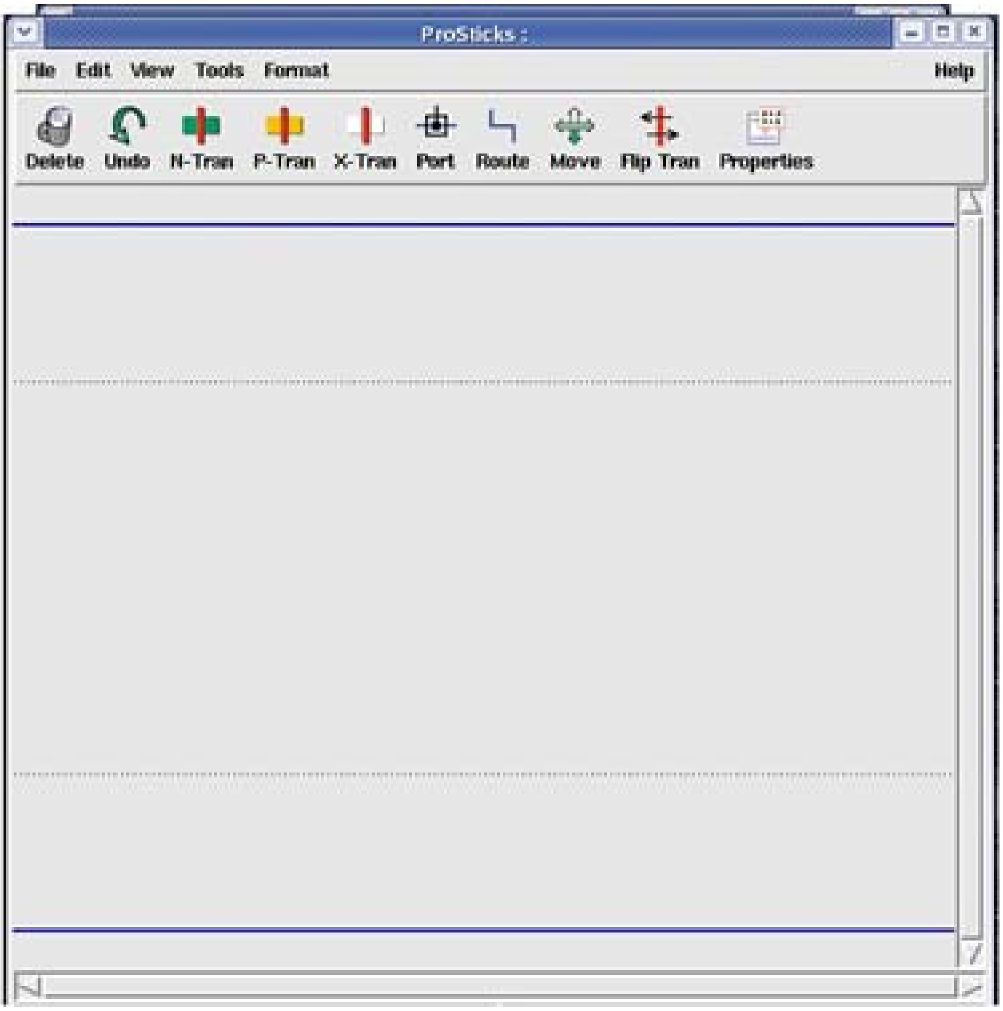


Figure 37: Start ProSticks with an empty workspace

If your previous cell had additional transistor rows (represented by dotted lines), use **Format>Delete transistor row** to remove them.

To begin creating the generator, let's make things easier by tearing off the **Edit** menu. Simply go to **Edit** and select the dashed line. In Figure 38, the **Edit** menu has been torn off the menu bar and is sitting above the workspace. You can place your **Edit** menu there, but if you're working on a large cell, it might be better to place it outside the ProSticks window so you don't obscure the workspace. The sample cell we'll be creating is simple enough that we'll just leave the Edit menu where it is while we work.

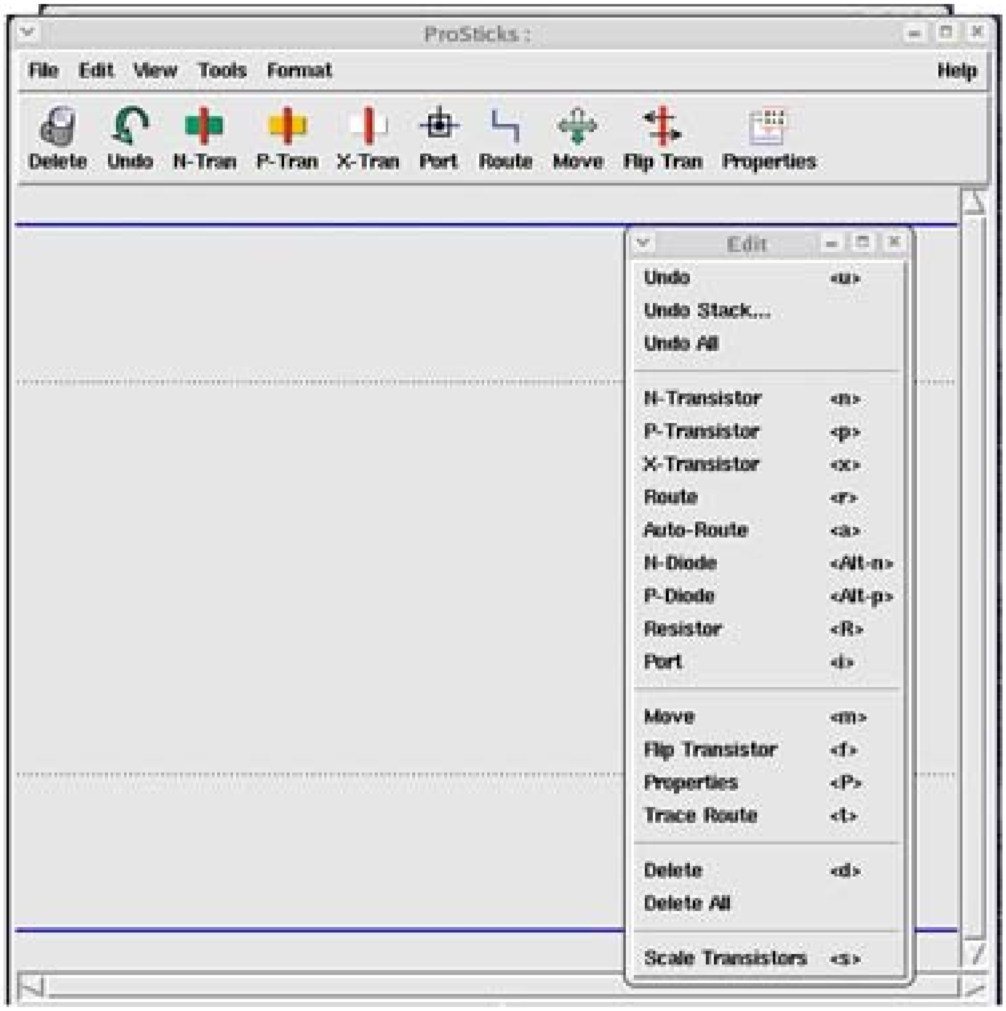


Figure 38: Tear off the Edit menu to prepare to create a generator

Now let's get started building the generator itself. We'll be creating a NAND2, which has four transistors. First, we click on n-transistor from the Edit menu. Then we click twice on the lower transistor row to place the transistors, as shown in Figure 39.

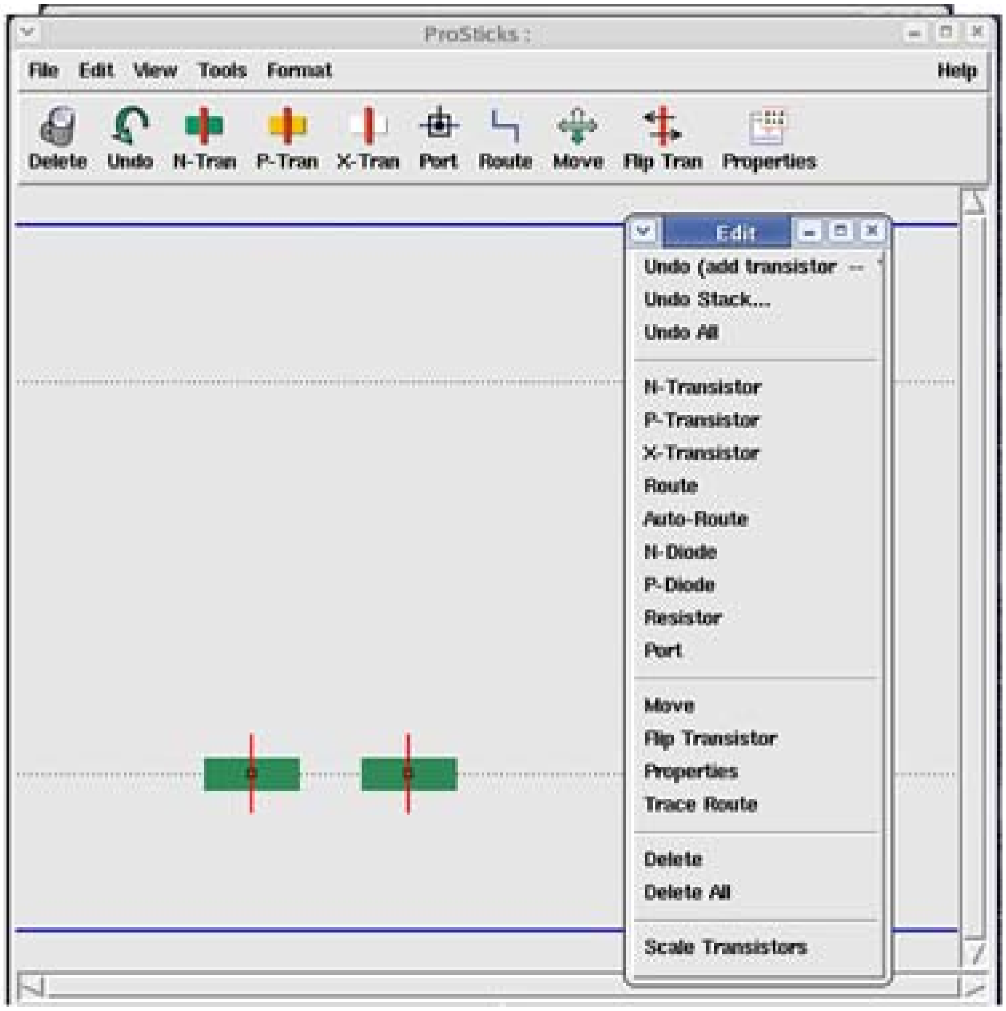


Figure 39: Create the two n-transistors for the generator

Unlike the transistors placed by importing a netlist, these transistors have no node names. Although you can't tell by looking at this view, they also have no transistor widths. We'll fix that using the Properties option a little later.

In the meantime, we need to place the p-transistors as well. This isn't difficult; we simply click on P-transistor on the Edit menu and then click twice on the upper transistor row to create the transistors as shown in Figure 40..

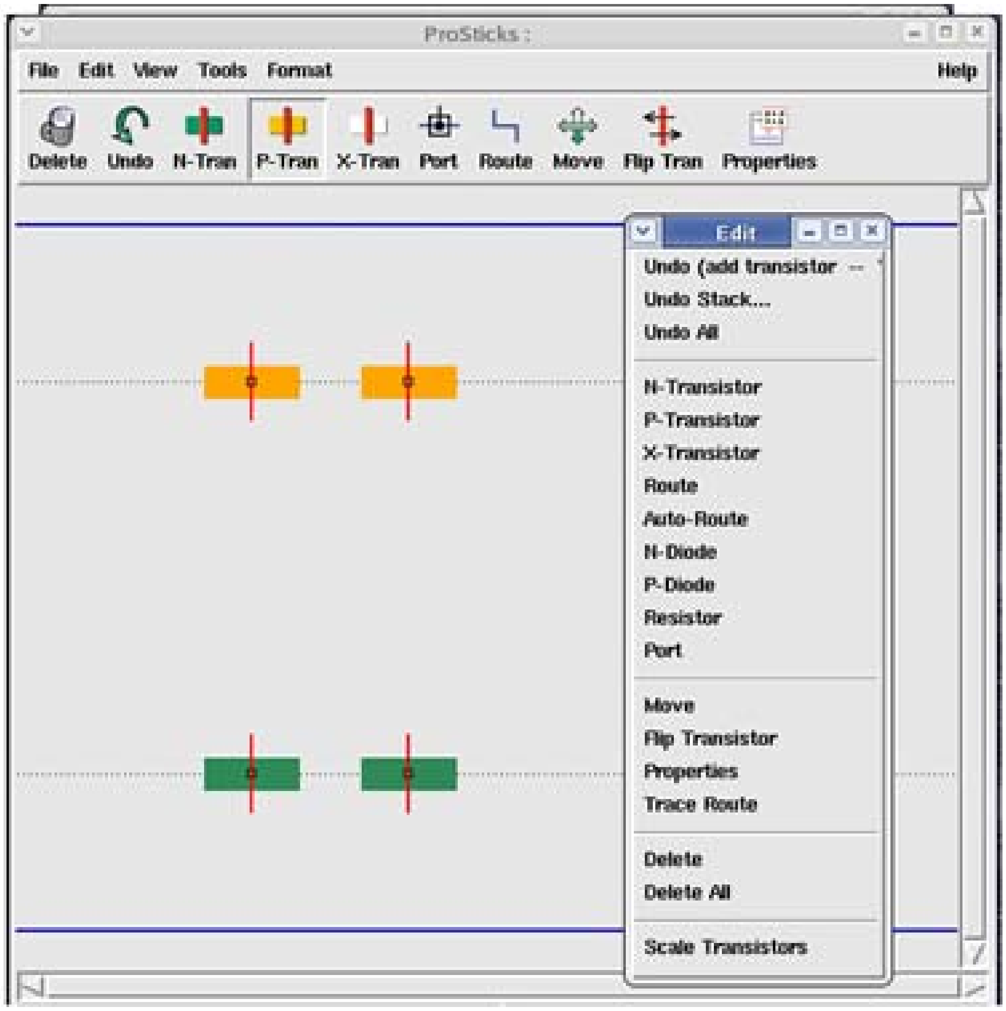


Figure 40: Place the p-transistors for the generator

You might expect that we'd label the transistors now by adding the node names, and we could certainly do that. In this case, though, we're going to start by routing two of the transistors together to show how this speeds node naming.

To route together the gates of the transistors on the left-hand side of the workspace, we click on the **Route** option on the **Edit** menu. Then we click on the gate (the red portion) of one of the transistors, and drag to the gate of the other transistor. Figure 41 shows the resulting route.

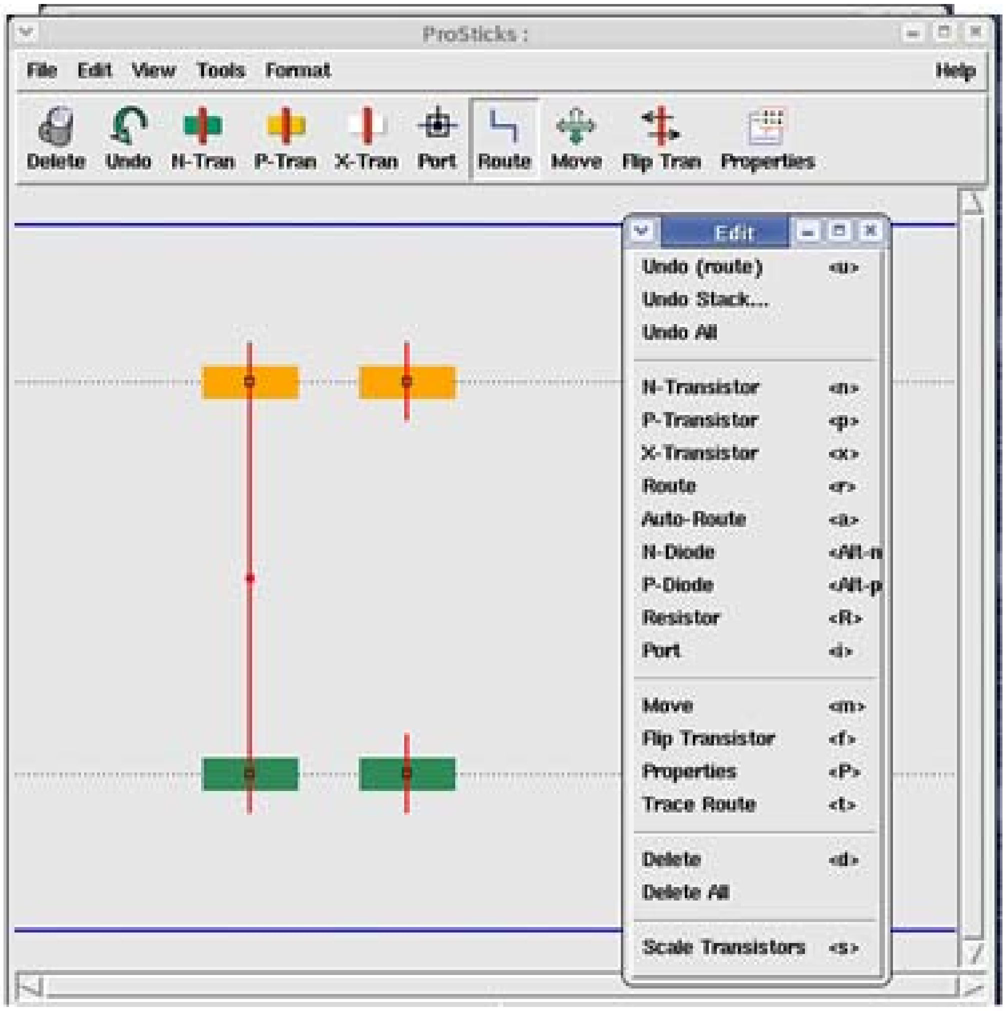
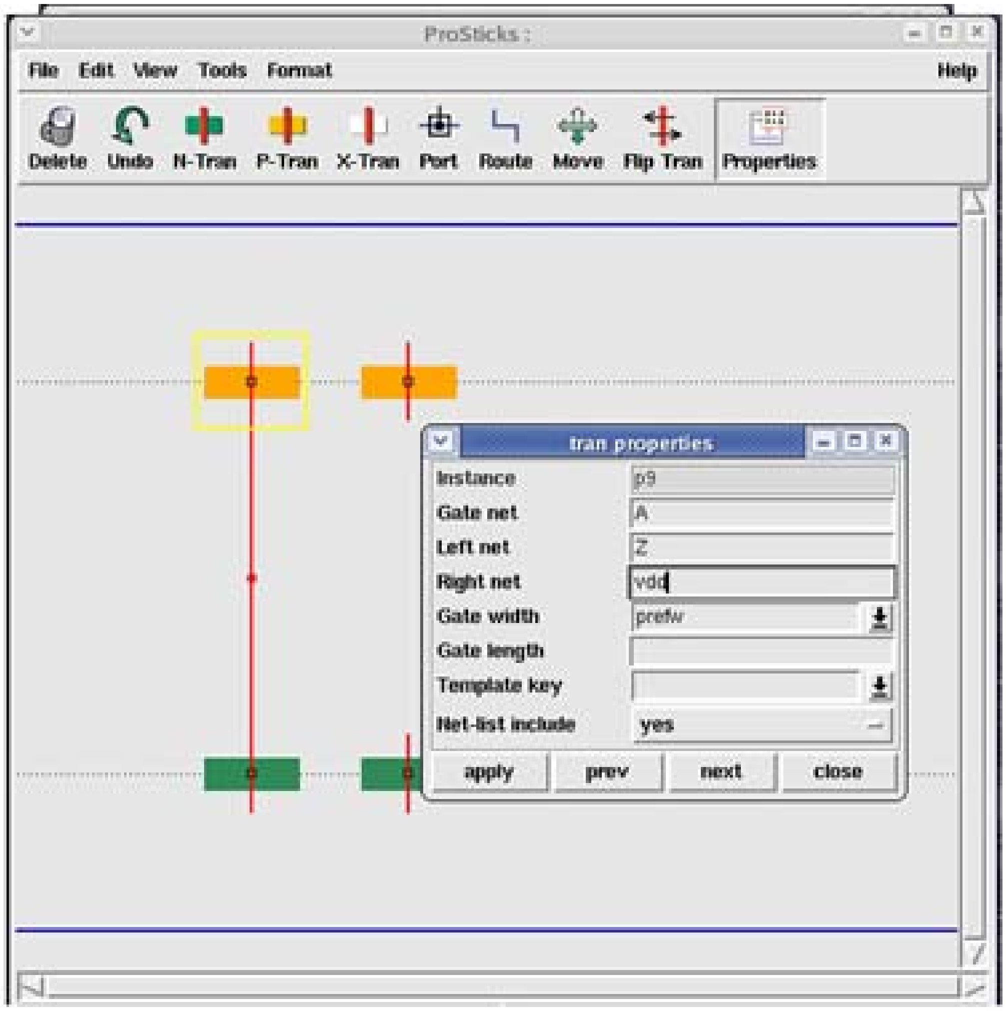


Figure 41: Select Route, then click and drag from one gate to another

Now that we have the gates routed together, we start naming nodes. To accomplish this, we simply select **Properties**, then click on one of the transistors. In Figure 42, the p-transistor in the upper left corner has been selected, and the gate and left and right node names have been entered. Notice that the labels have not yet been placed on the ProSticks workspace.

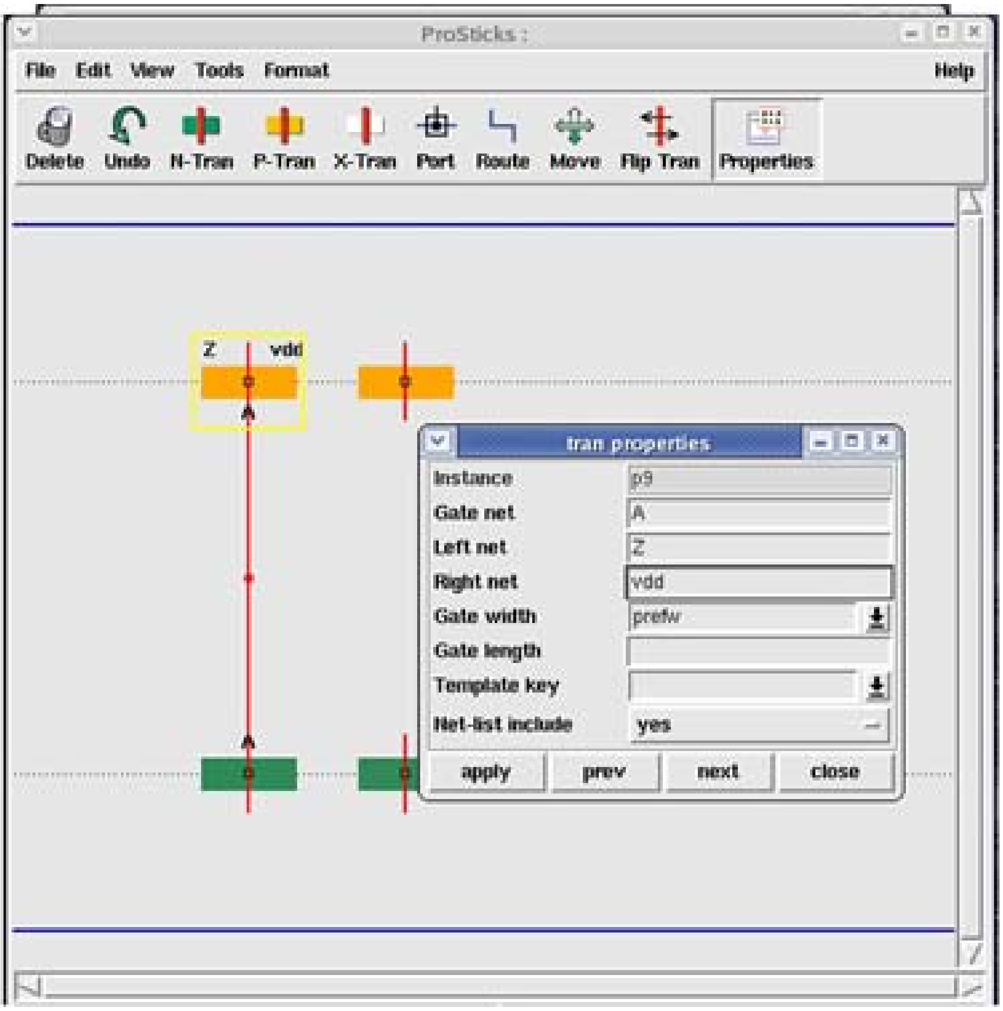


Instance	p9
Gate net	A
Left net	Z
Right net	vdd
Gate width	prefw
Gate length	
Template key	
Net-list include	yes

Figure 42: Enter the transistor's node names

To cause the names to appear on the workspace, click the **Apply** button in the **Transistor Properties** window. Until you apply a node name, it isn't actually associated with the transistor; so to avoid losing work, always click the **Apply** button after you have entered the node names.

Figure 43 shows the updated cell once the node names have been applied. Notice that the left net name is Z, the right net name is vdd, and the gate is A, as specified in the Node Names window. But also notice that the gate on the connected n-transistor on the bottom left has been named A as well.



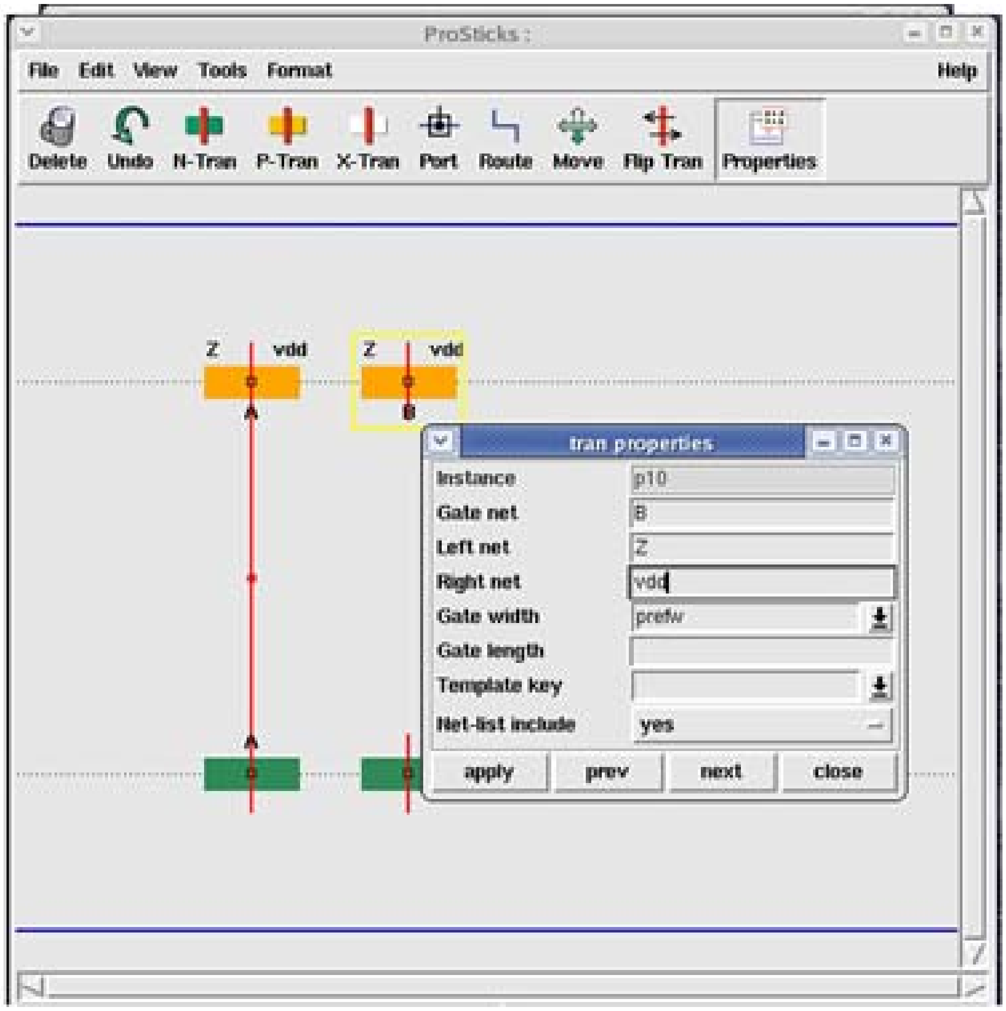
Instance p9
Gate net A
Left net Z
Right net vdd
Gate width prefw
Gate length
Template key
Net-list include yes

Figure 43: Click the Apply button to confirm the node names and label the transistor

This is one way to save time when creating a familiar cell: route the cell first, then start naming nodes. ProSticks will carry over the node names from each transistor to each routed node.

To name the nodes of the next transistor, simply click the Next button in the Transistor Properties window, and the yellow box highlighting the transistor will move to the next transistor. Enter the gate, left, and right node names for that transistor, and click the Apply button.

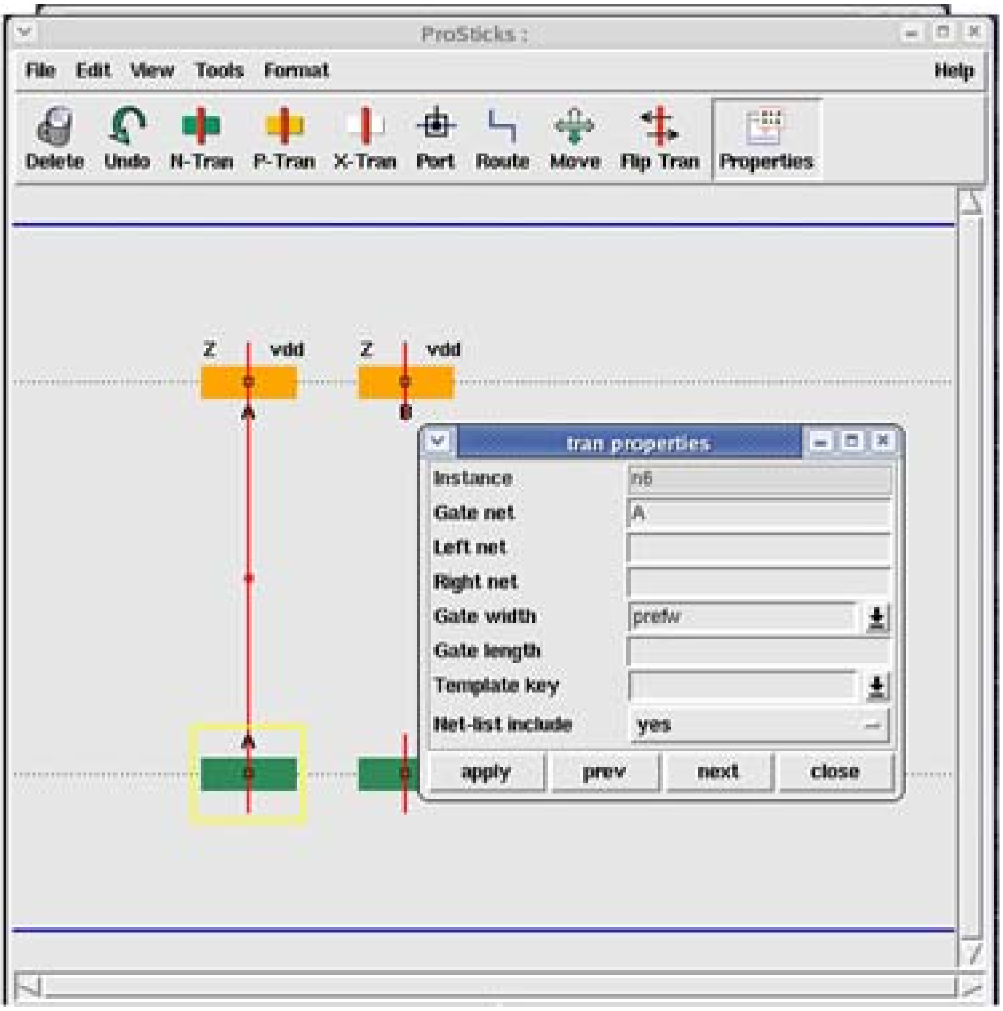
In Figure 44, the rightmost p-transistor has been named. Its gate net is B, its left net is Z, and its right net is vdd.



Instance	p10
Gate net	B
Left net	Z
Right net	vdd
Gate width	prefw
Gate length	
Template key	
Net-list include	yes

Figure 44: Click Next and name the next transistor's nodes

If we click the **Next** button again, the yellow highlight box moves to the leftmost n-transistor. Notice that the gate is already named A, as shown in Figure 45. As we mentioned previously, this gate was already connected to the gate of the p-transistor, so it appropriates the same node name.

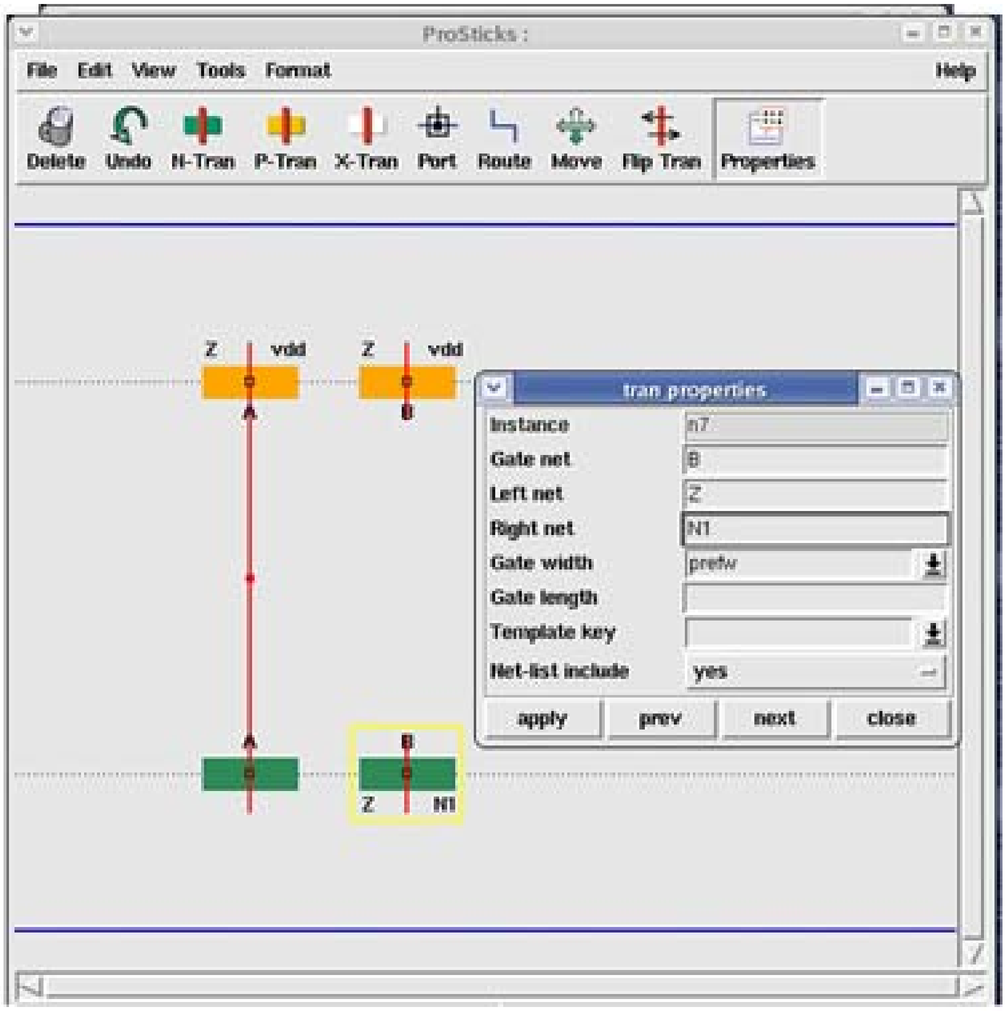


Instance: n6
Gate net: A
Left net:
Right net:
Gate width: prefw
Gate length:
Template key:
Net-list include: yes

Figure 45: Nodes connected to previously named nets will already have names

We'll fill in the other nodes by naming the left net N1 and the right net vss. We click **Apply** to make the node name addition take place, and then click **Next** to move to the last transistor.

Add the gate name B, the left node name Z, and the right node name N1, and click **Apply.** The result appears as shown in Figure 46.



Instance	n7
Gate net	B
Left net	Z
Right net	N1
Gate width	prefw
Gate length	
Template key	
Net-list include	yes

Figure 46: Enter the last transistor's node names

Now that the nodes have all been named, we're ready to start the placement process. We know that sharing diffusion is good, so we look at the p-transistors and notice that they can share either the Z node or the vdd node. Since the p-transistors don't force a particular placement, we look at the n-transistors and see that the only node name they have in common is N1. We'll have to move the transistors around a little bit, but in this case, it's pretty simple.

First, we click **Close** to shut the **Transistor Properties** names window.

Next, we go to the toolbar (or **Edit** menu) and click **Move.** We then click on the rightmost p-transistor (the one with the gate name B), and drag it to the left side of the other, already-routed p-transistor.

When we let go of the transistor, it snaps into place as shown in Figure 47. This illustrates the autoflipping of transistors feature: the transistor we were moving started with Z on the left and vdd on the right, but it flipped over so the Z contact could be shared with the other transistor.

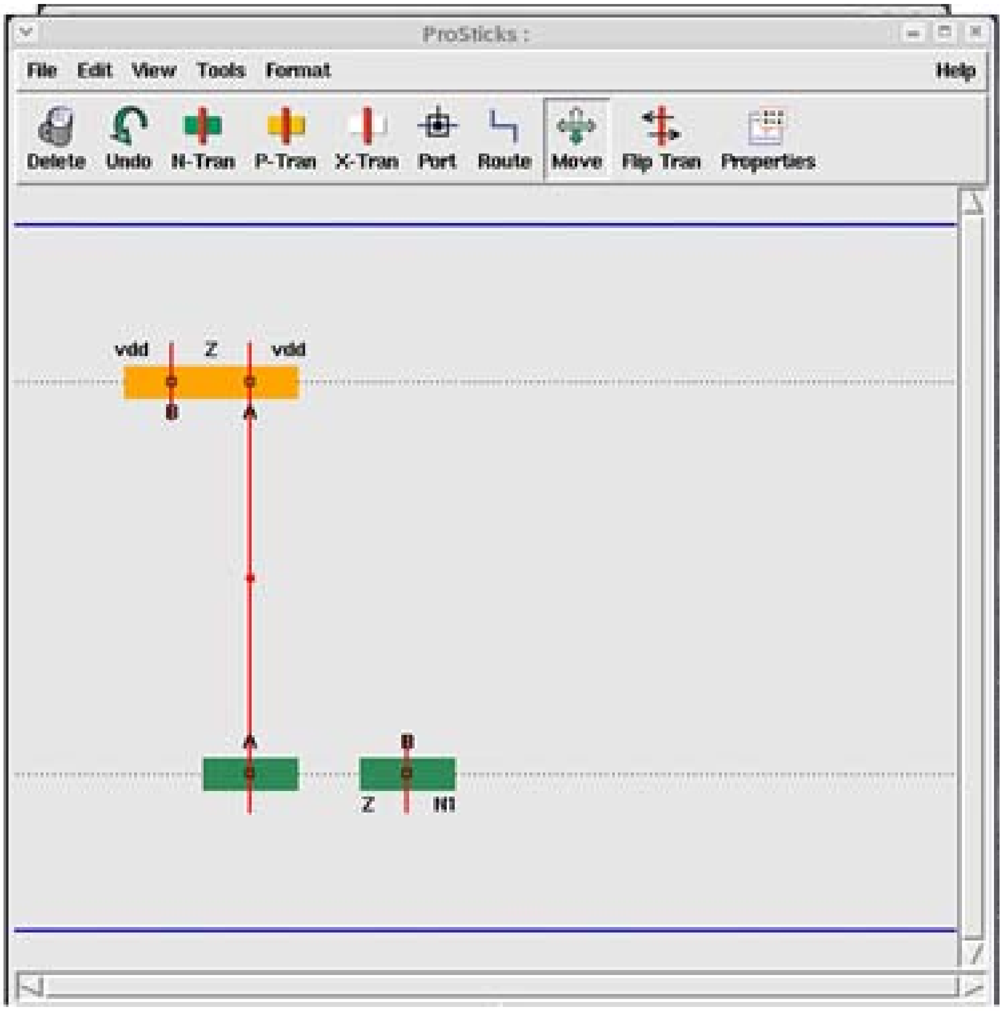


Figure 47: Move the p-transistor to share diffusion

The autoflipping behavior is enabled by default; you can turn it off by deselecting Auto Flip on the View menu.

Next, we'll move the rightmost n-transistor to the left side of the workspace to share diffusion with the other n-transistor, as shown in Figure 48. This will line up its gate with the p-transistor we just moved, so we won't need a jog in the poly connecting those gates.

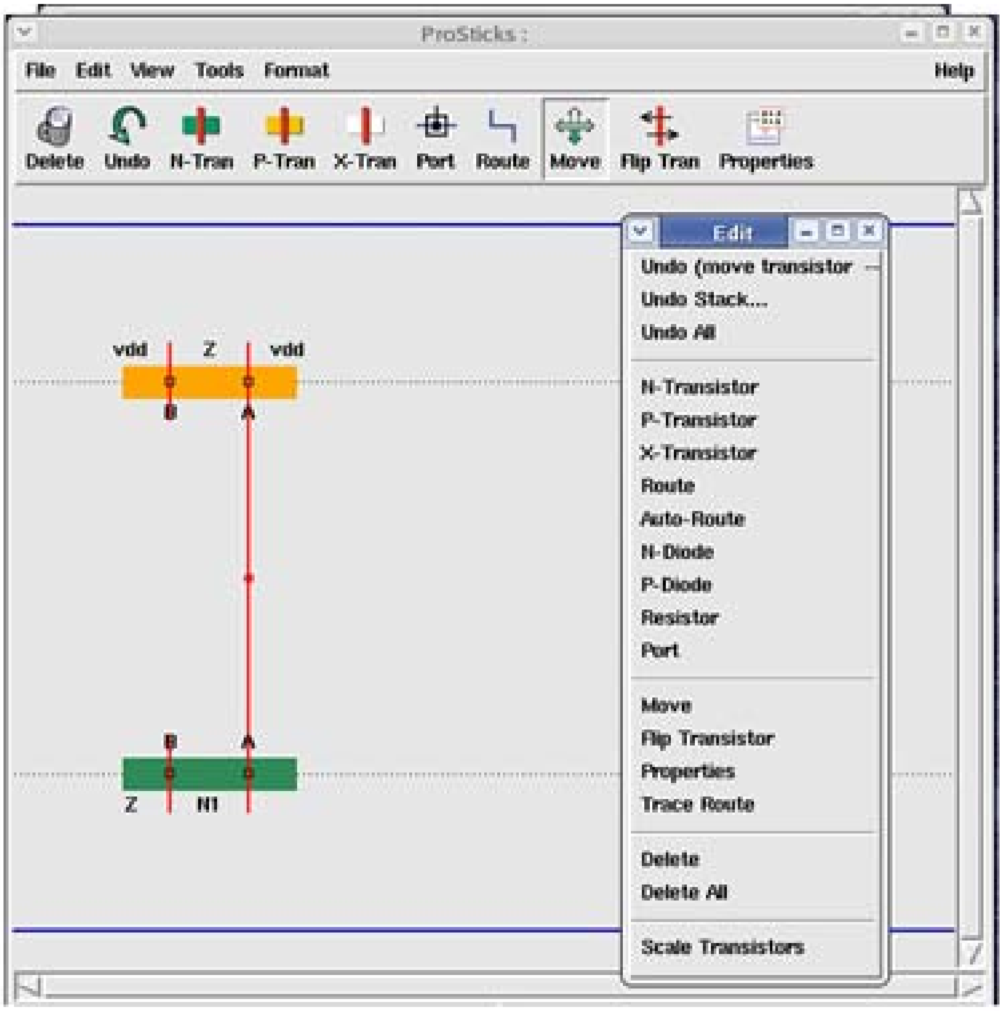


Figure 48: Move the n-transistor to share diffusion and allow a poly connection without jogs

Now that we have the transistors where we want them, we're ready to begin routing them up. We'll start by connecting the gates on the leftmost transistors.

To start the routing, we click on Route from the toolbar or Edit menu. Then we click on the gate, B, of one of the transistors (it doesn't matter which one) and drag to the corresponding gate (B) of the other transistor. Figure 49 shows the resulting layout.

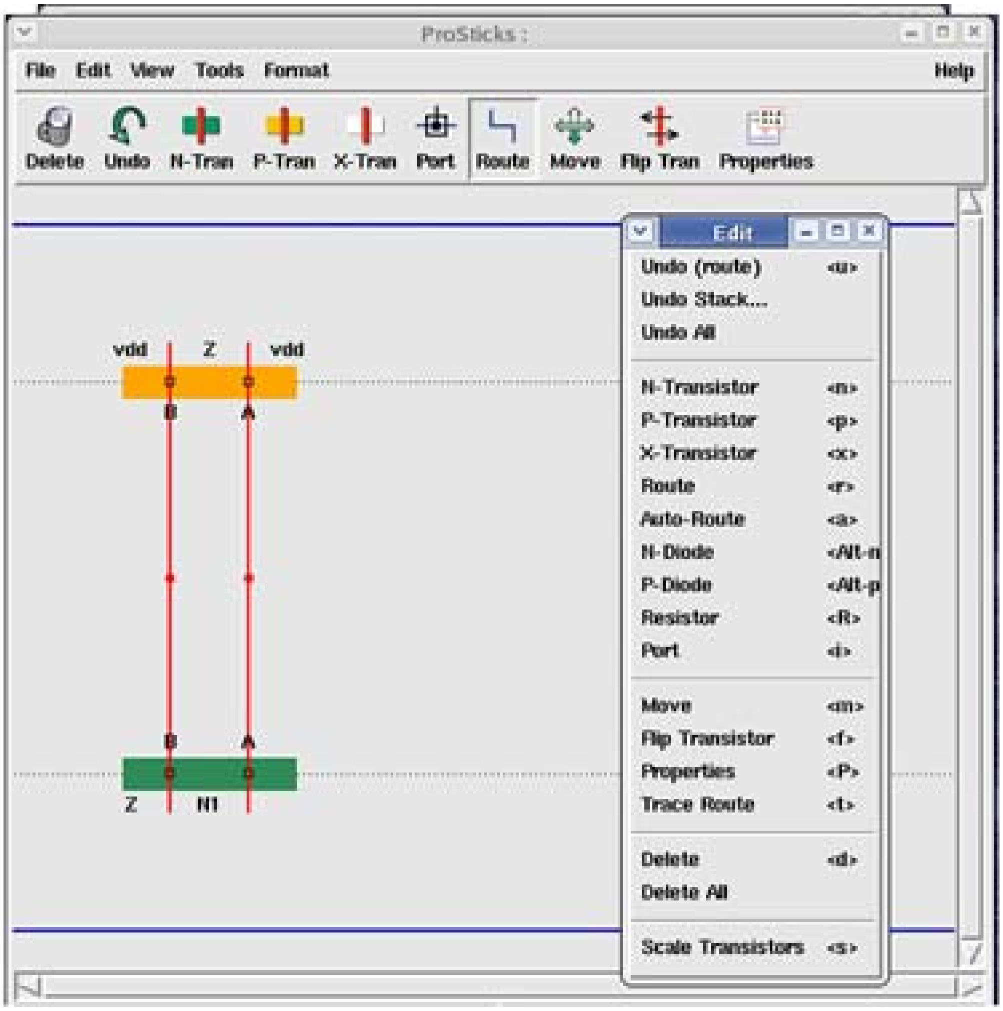


Figure 49: Route the B gates of the n- and p-transistors

Now we're ready to connect the transistors to the power rail. We've already selected Route -- you can check the status line (just beneath the menu bar) to be certain.

To route to the power rails, then, we simply click on a vdd contact and drag up to the power rail at the top of the workspace. In our sample cell, we've got two power connections, so we route them both and the layout appears as shown in Figure 50.

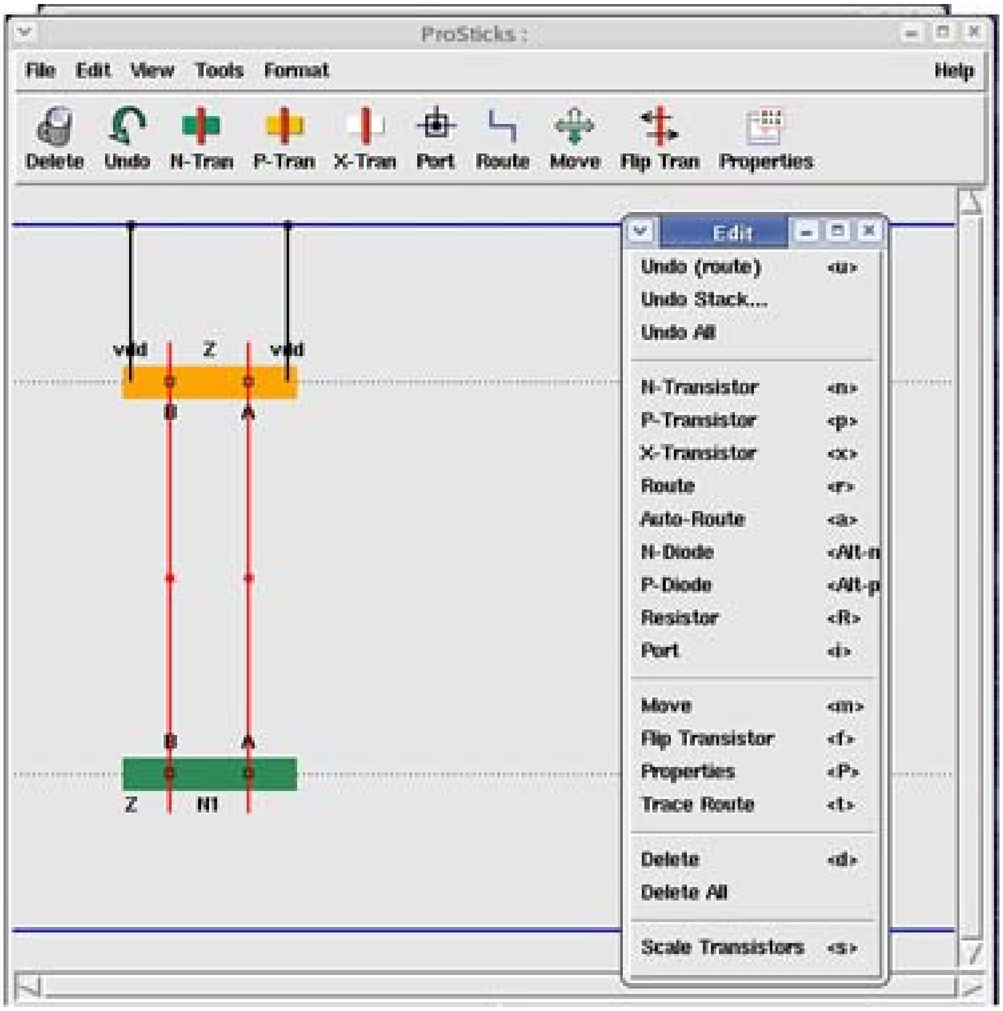


Figure 50: Add the feeds to the power rail

Adding the feed to the ground rail is just as simple; just click on the vss contact and drag to the ground rail at the bottom of the workspace.

Next, we'll add the output routing. With the Route tool still selected, we click on one of the Z nodes and drag to the other Z node. The resulting feed line has a jog across the poly of net B, as shown in Figure 51.

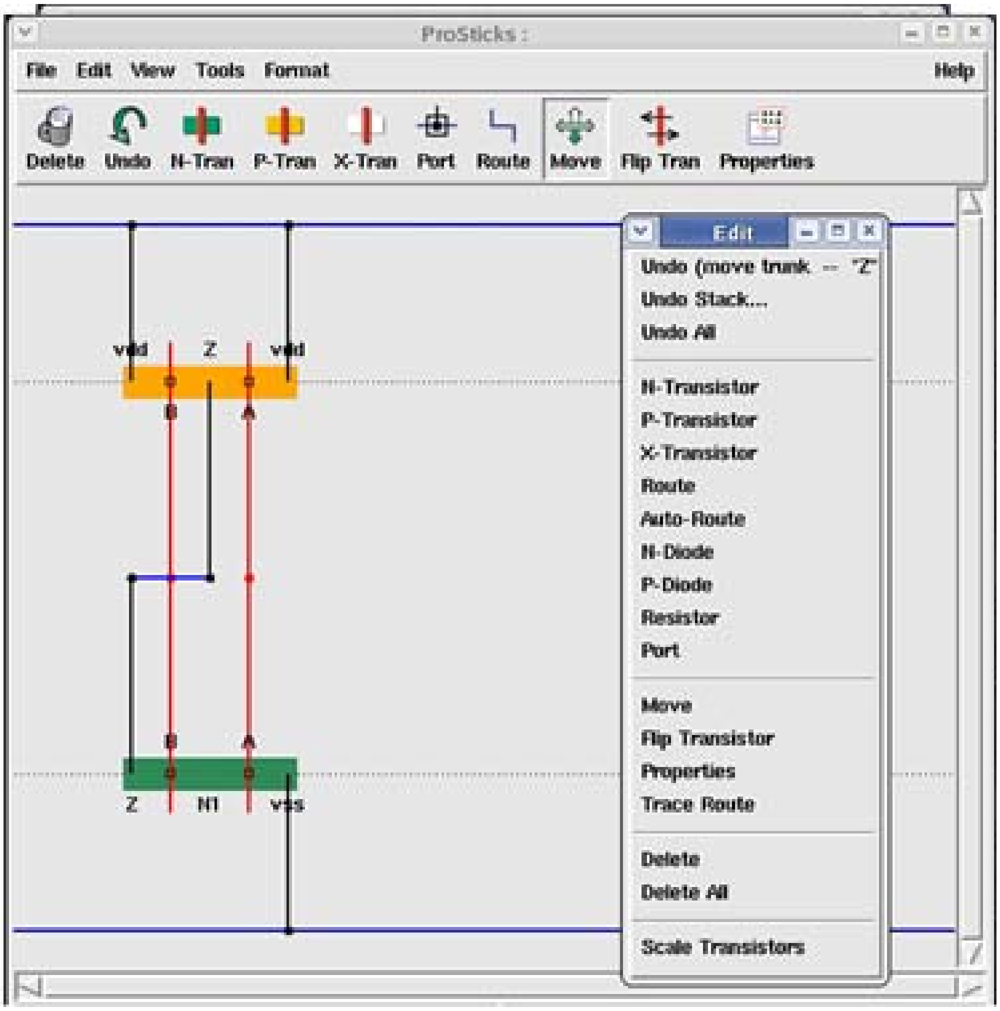


Figure 51: Add the output feed by routing between the Z nodes

Wire Colors

Please note that the colors of the wires don't necessarily indicate what layer they're using. In this case, the feed wires are black and the jog is blue, but they can both use metal-1.

In this example cell, there isn't much routing going on in channel 1, so there isn't much congestion. Notice that in Figure 98, the jog is placed in the center of the channel. This is the default ProSticks behavior; if you wish a jog to use another track, you'll have to move it. This is another simple operation, and we'll illustrate it:

First, we click on Move from the Edit menu. Then we simply move the cursor over the jog we wish to move so that it's outlined, then click it and drag it where we want it. In this case, we'll move it up a bit and place it as shown in Figure 52.

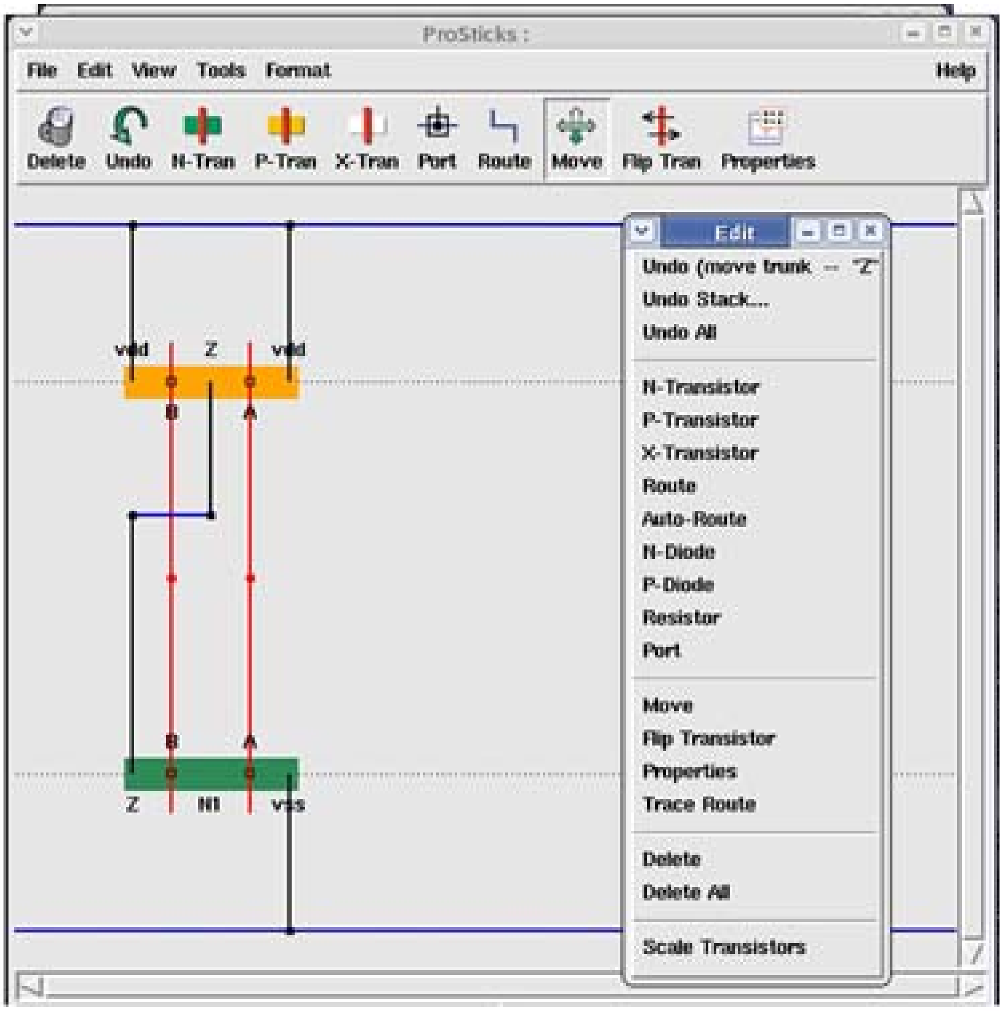
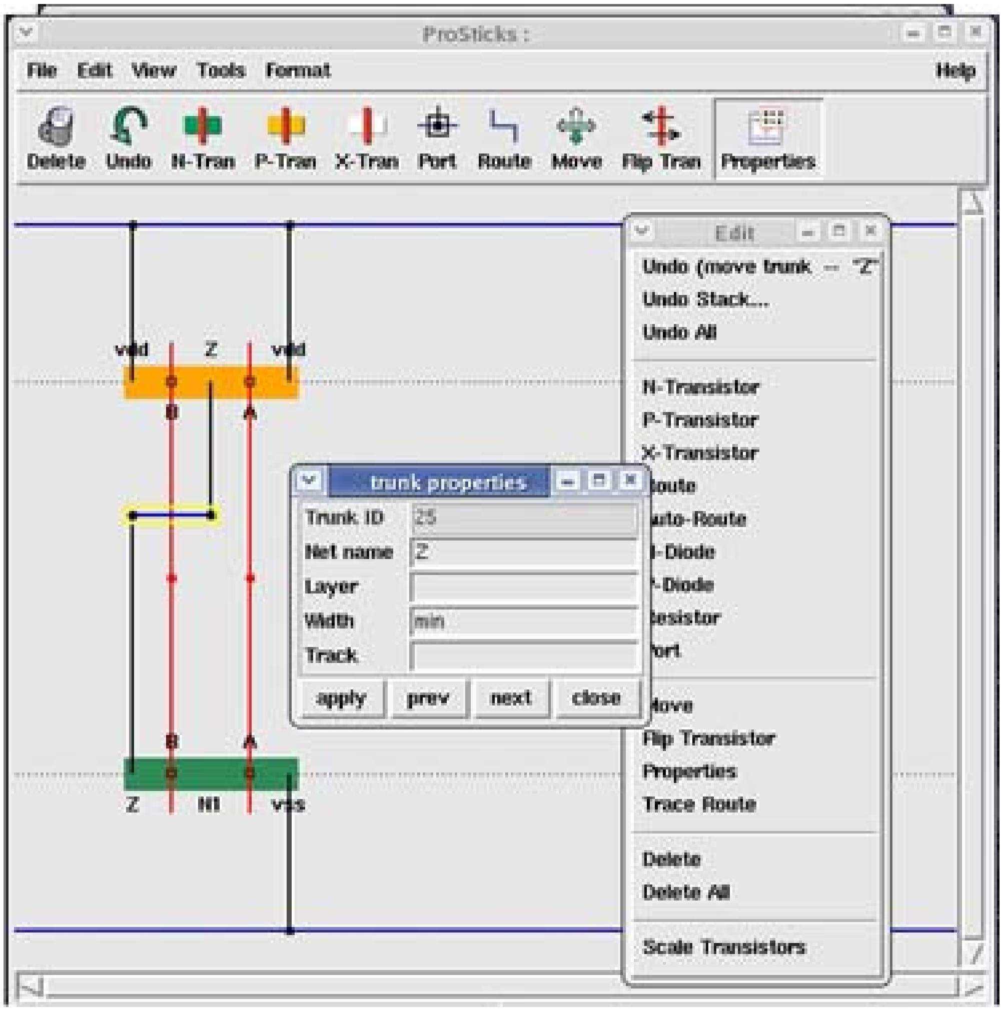


Figure 52: Move the jog to another track in channel 1

We're ready to add ports; then our physical layout will be ready for testing.

Select **Port**, and then the horizontal trunk wire segment. (Select the dot if the wire doesn't have a horizontal tag).



Trunk ID 25
Net name Z
Layer
Width min
Track

Figure 53: Check the Port property to add the port

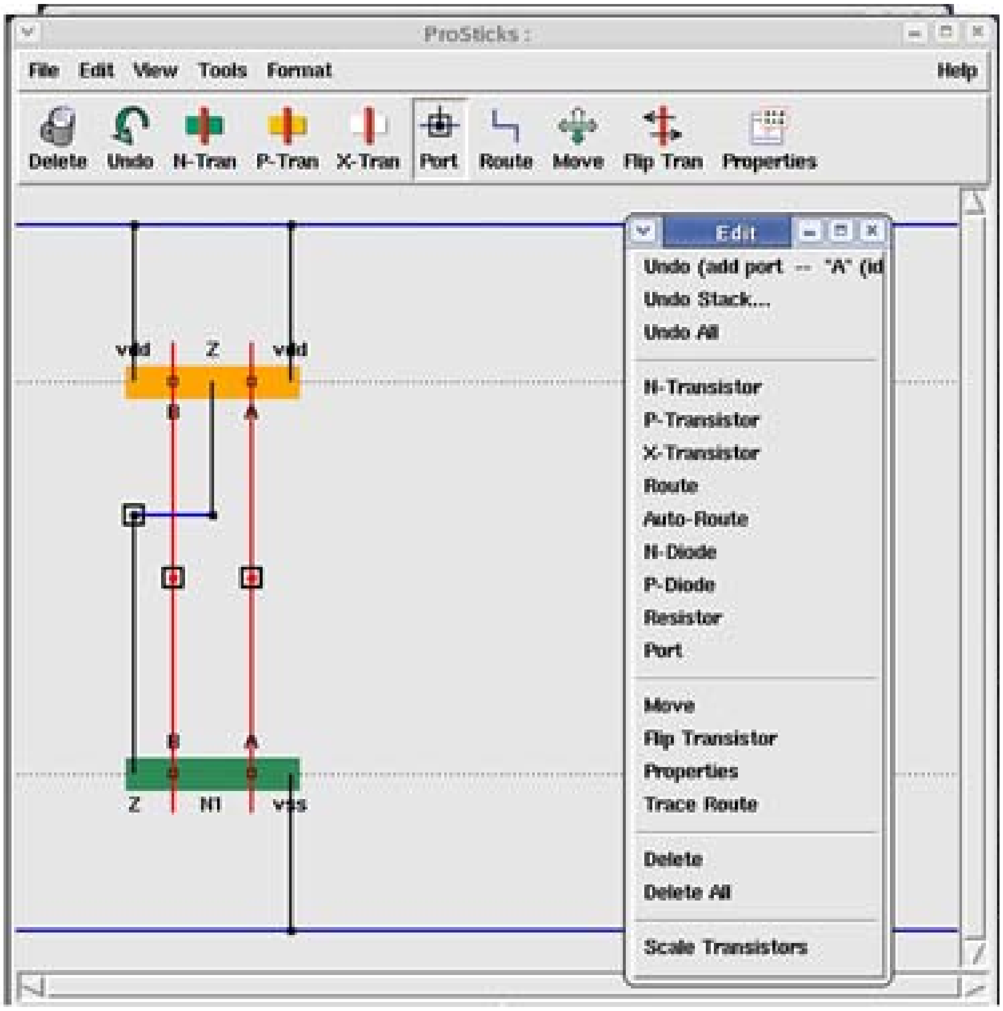
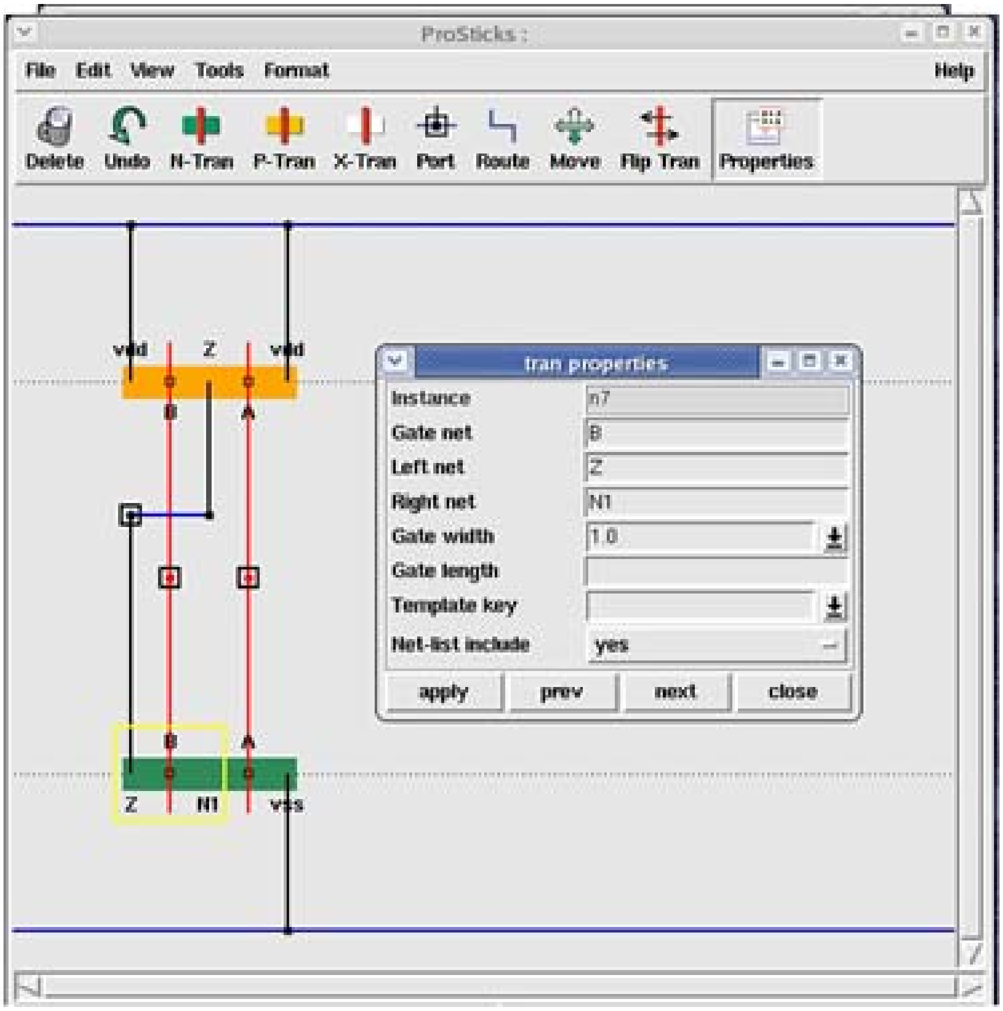


Figure 54: The completed layout is fully routed and populated with ports

We're not quite finished here, however. When we import a netlist, the transistor widths are determined in the Spice listing. When we create layout by hand, however, the transistor widths are not determined. We need to add them, too, if our output is going to be useful.

We can use the **Properties** tool to set the transistor widths. We simply click on a transistor and add the correct width in microns. In Figure 102, the p-transistor on the top left has been selected and its width is being set at 2.0 microns.

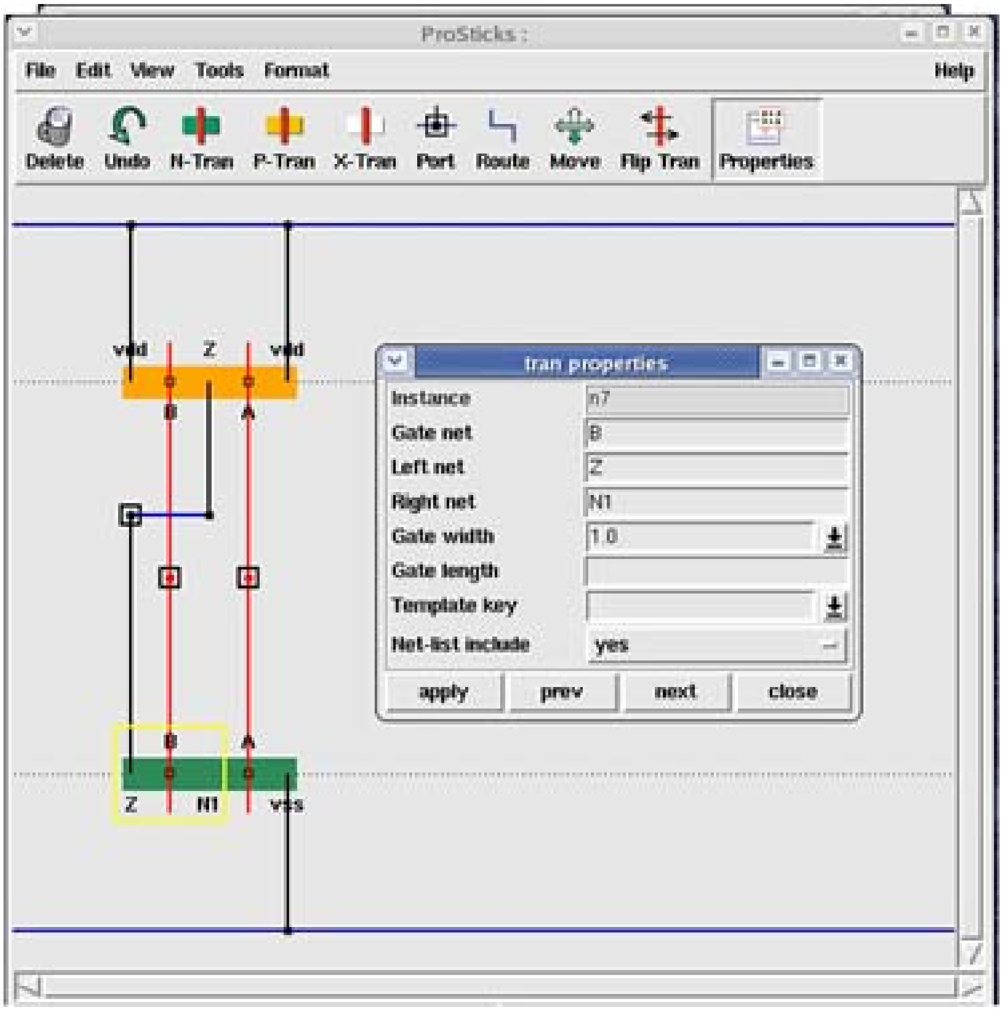


Instance: n7
Gate net: B
Left net: Z
Right net: N1
Gate width: 1.0
Gate length:
Template key:
Net-list include: yes

Figure 55: Set the transistor widths from the transistor Properties window

Be sure to click **Apply** to update the transistors with their new widths. Net names are displayed on the workspace (unless you turn off the display by deselecting the **View>Node Names** check box); but you may not notice that you're missing a transistor width or have set a width incorrectly.

Once you have applied the first transistor width, click on the next transistor and enter its width. You may want to leave the Transistor Properties window open while you click on each of the transistors to ensure that they all have width values. The n-transistor in Figure 103 has a width of 1.0 micron.



Instance	n7
Gate net	B
Left net	Z
Right net	N1
Gate width	1.0
Gate length	
Template key	
Net-list include	yes

Figure 56: Be sure all the transistors have the proper widths using the Properties tool

Once we've clicked close from the **Transistor Properties** window, our layout is finished, as shown in Figure 104. We're ready to test it out by running ProGen.

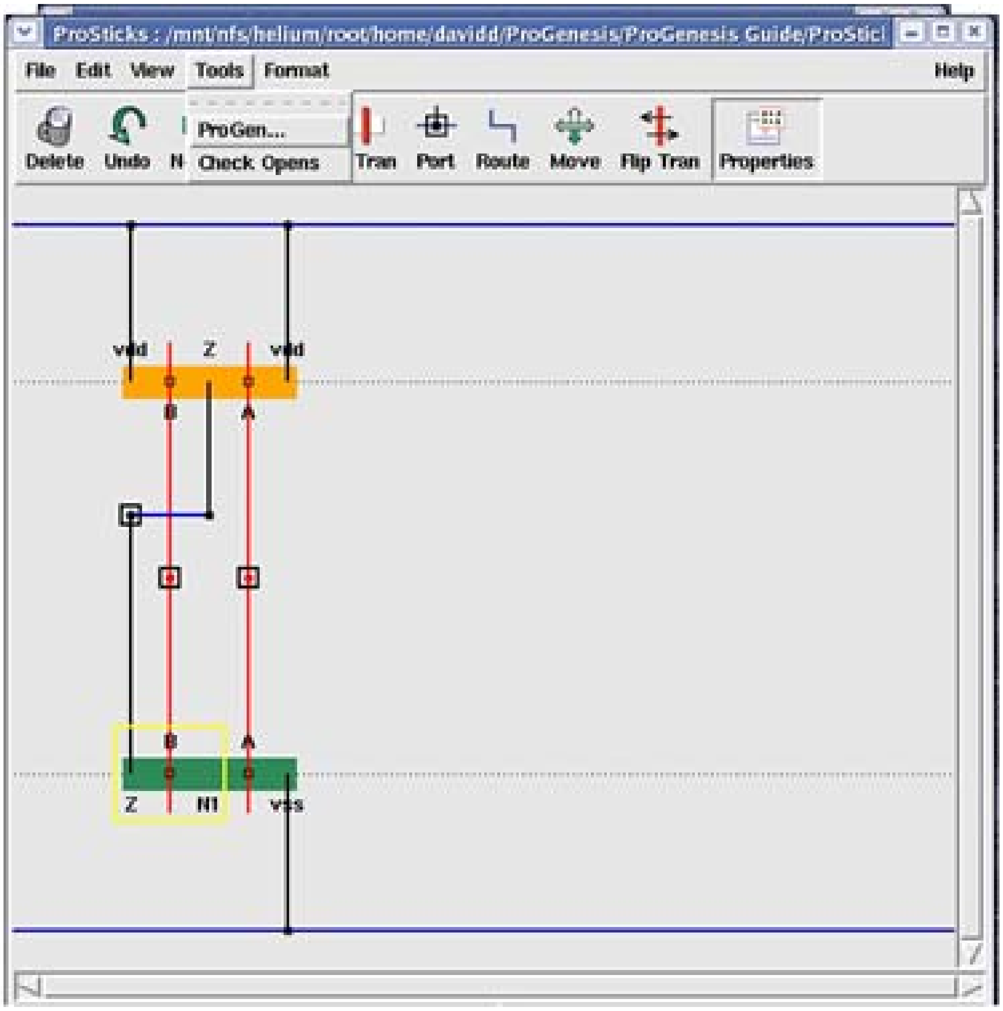
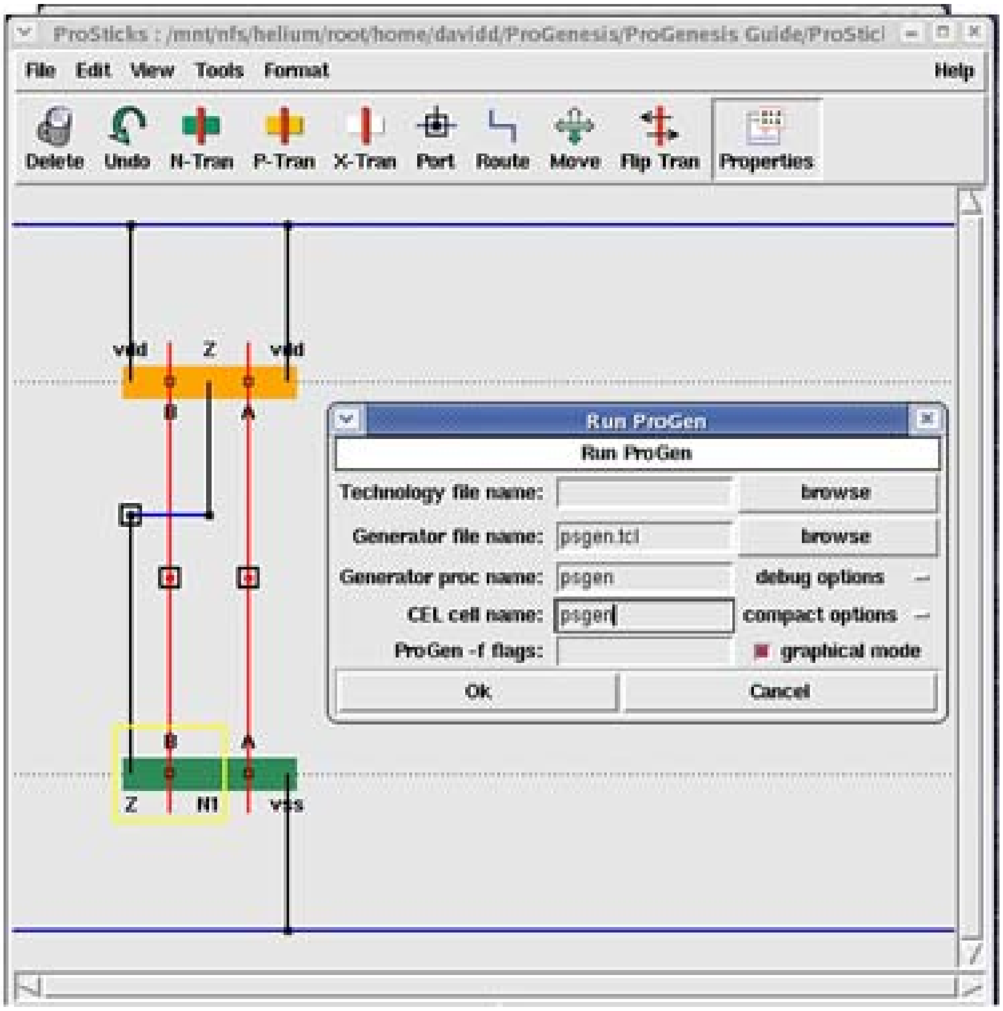


Figure 57: The layout is ready for testing

Before we go any further, we suggest that you take this opportunity to save your work. Saving the ProSticks file (using the .PSC format) will preserve the layout as you've arranged it to this point, and you'll be able to edit it after you run ProGen if you want to optimize things.

(...And if your office experiences a power-terminating cataclysm as ProGen runs, you won't lose any work. It's just a good idea.)

To run ProGen, we go to the **File** menu and select the **ProGen** option. This opens up the **Run ProGen** window shown in Figure 58.



Run ProGen

Technology file name:

Generator file name: psgen.tcl

Generator proc name: psgen

CEL cell name: psgen

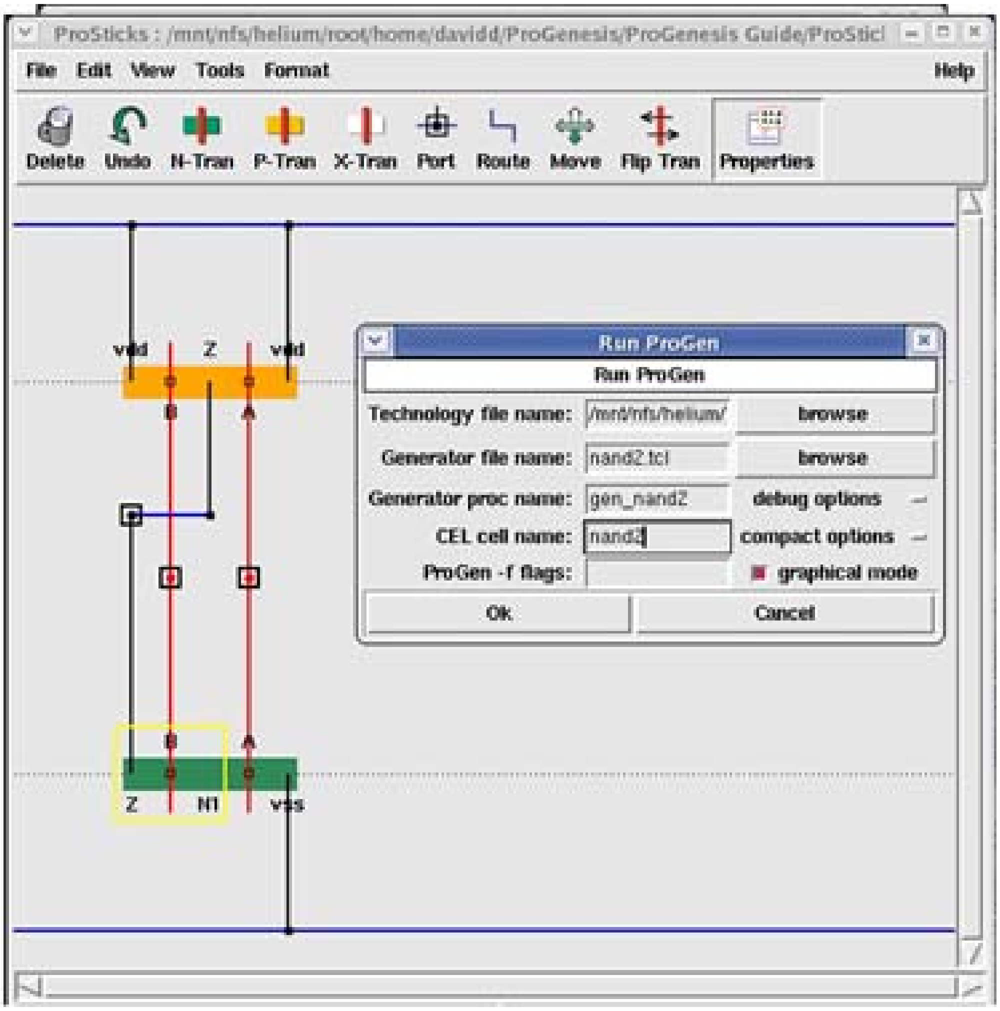
ProGen -f flags:

☒ graphical mode

Figure 58: Enter the filenames and select options to run ProGen

The options shown in Figure 58 are the defaults; we could fill in the technology file name and run ProGen with these values. However, we won't know what the files are by looking at them, as psgen isn't a very descriptive name. So we'll point at a tech file and set the names to more descriptive ones, and then we'll be ready to go.

We start by clicking the **Technology File Browse button** in the Run ProGen window (the one to the left of the **Technology file name** field). We move through the directory structure until we find the technology file we're using. Then we click on it and click **Open** to add its location to the **Technology file name** field. In Figure 59, the path to the technology file starts with /mnt/nfs/helium.



Run ProGen

Technology file name: /mnt/nfs/helium/

Generator file name: nand2.tcl

Generator proc name: gen_nand2

CEL cell name: nand2

ProGen -f flags:

■ graphical mode

Figure 59: Use descriptive names that uniquely identify the generator, procedure, and PSD file

Once the technology file is selected, we can set the generator name. The generator is the .tcl file created by ProSticks to generate cells of the type in the layout. Because this is the layout for a 2-input NAND generator, we've named the generator nand2.tcl, as shown in Figure 59.

The generator filename's default is psgen, but we have the option to name it whatever we like. For ease of future reference, we name the procedure nand2 in "CEL cell name" (because it calls the 2-input NAND generator) as shown in Figure 59.

At this juncture, we could select items from the **Debug Options** menu. If this were a large, complicated cell, we might opt to skip compaction the first time around and make sure everything worked properly. But because this is a simple cell, we're just going to run ProGen without any options set. Likewise, we don't need to alter the compaction options. So we click OK and ProGen starts. As the layout elements are placed and routed per our ProSticks drawing, we see something like the layout shown in Figure 60.

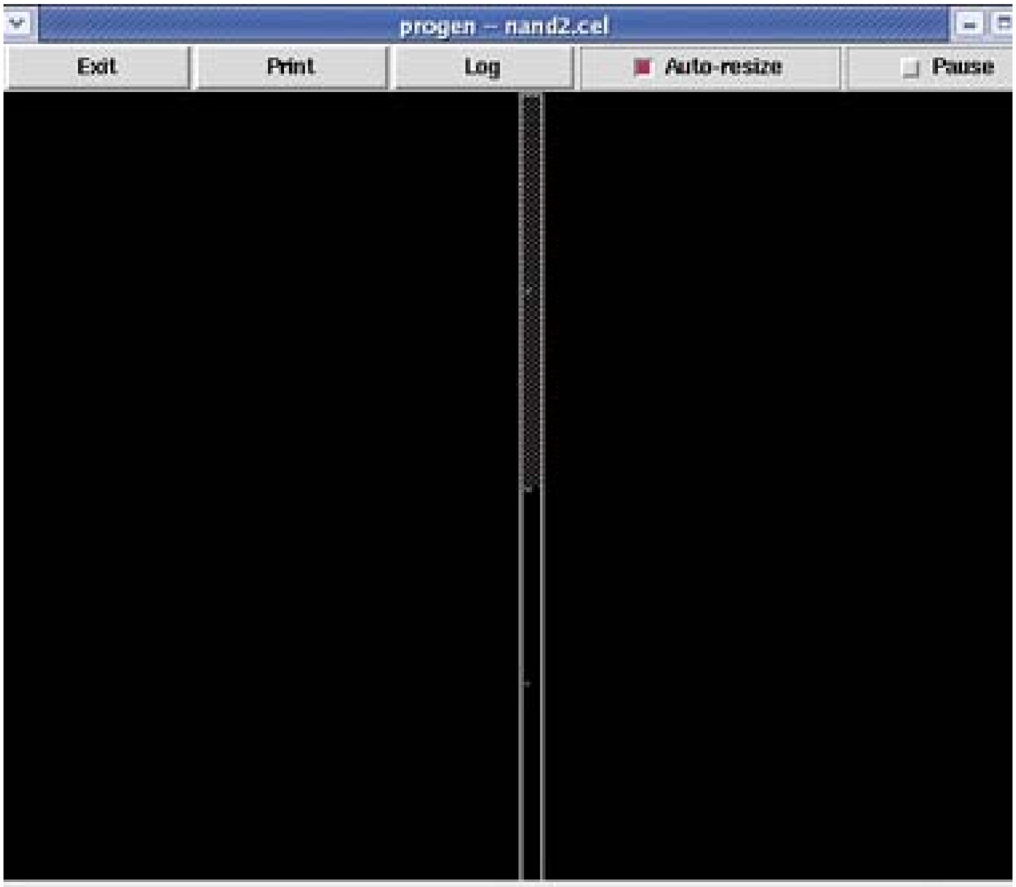


Figure 60: ProGen begins placing and routing the layout elements

Notice how the rails and wells initially extend far beyond the cell width.

When the layout has finished and compaction begins, the wells and rails compress down to the size needed for the cell layout. Since the default compaction order starts with the x-axis, the wells and rails shrink down right-to-left.

As compaction continues, you can click Resize to enlarge the layout to fill the window. The compaction process will continue to reduce the layout size.

In Figure 61, the layout -- resized to fill the ProGen window -- has finished.

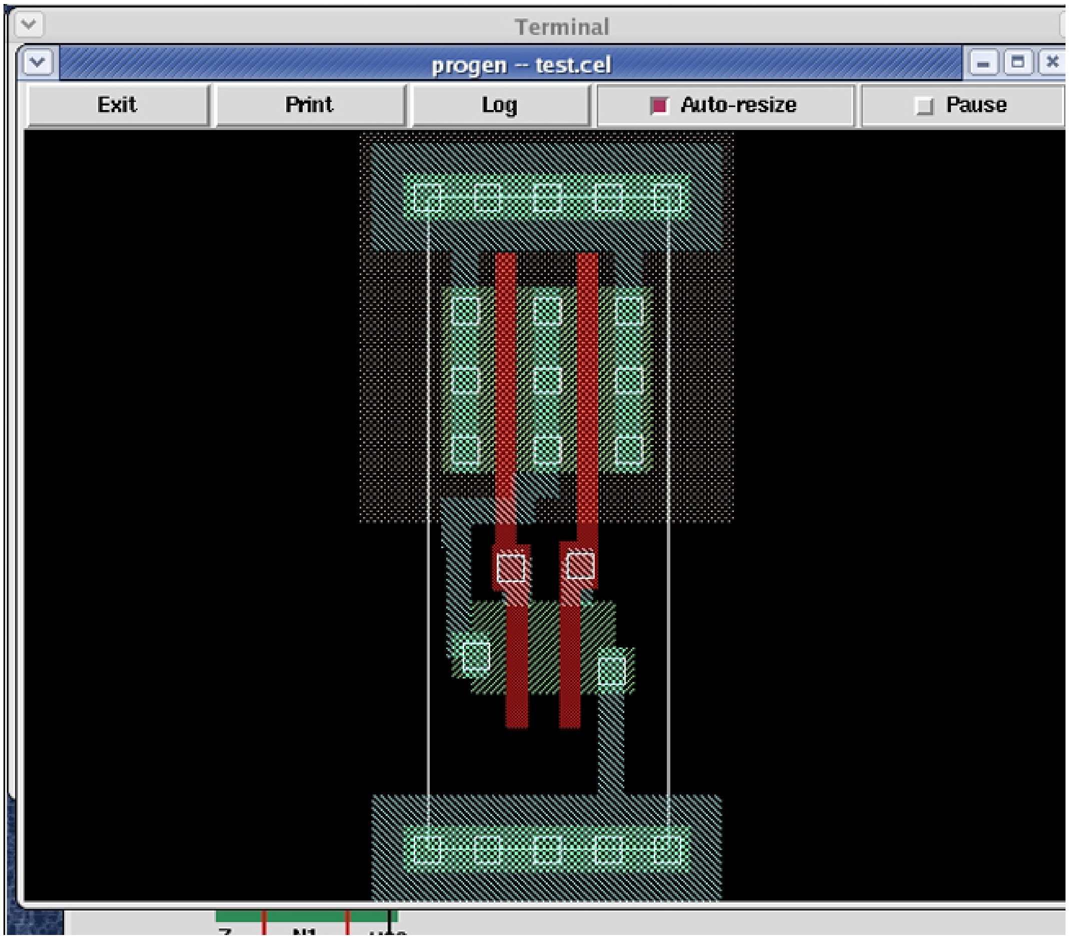


Figure 61: ProGen produces the final output

If you wanted to make layout modifications based on this result, you could go back to the ProSticks workspace and change the file, then run ProGen again. It is possible to try a variety of layout options in this way. Once you have the generator the way you want it, you're done!

ProSticks FAQ

Do I need to create generators for each drive strength of the cell? -- or is the tool able to handle it? I tried a 4x generator for a 6x cell but it doesn't work.

You need to create a ProSticks generator variation only if the existing one will cause missing cell height when you use it with a bigger drive strength version of the function.

Why aren't the changes to the transistor properties being updated as I enter them?

Be sure you're clicking Apply after each change.

The colors don't always match what I think they should. What gives?

The colors used in layout routes aren't strictly controlled. To be sure you know what material is being used, check the route element's properties using Edit>Properties.

What are the different elements that can be created and manipulated in ProSticks layouts?

The elements that can be created and manipulated are:

- Transistor Rows
- Rails
- Transistors
- Diodes
- Trunks
- Jog Wires (Feeds)
- Ports

ProGen doesn't run properly after I imported a netlist.

Be sure you're using the right scaling values. ProSticks assumes the netlist you're using specifies transistor sizes in meters; if this isn't true, and if your scaling value isn't specified in the netlist, you could be trying to create cells that are far too large for ProGen to handle (over two meters).

ProGen doesn't run properly after I stack transistors.

Make sure you don't just plunk the x-transistors down; you must route to and from them like you would with any other transistor.

ProAssemble Reference

ProAssemble underlies the operations of ProSticks. It provides a programming interface for creating generators. It allows the designer to focus on circuit topology and avoid low-level details such as which generator routines to call and when.

Two basic sets of information need to be defined for ProAssemble: the transistor topology and the routing topology.

Transistor Topology

Transistor topology defines the transistor placement order, orientation, gate types (inverter, passgate, etc), node names (source, drain, gate), transistor row placement, gate widths, and folding.

gates

The transistor order is defined by specifying a list of gate type/names. Transistors will be placed left-to-right corresponding to the list order. The first character of the gate name indicates its type. Remaining characters help to uniquely define the gate name. The six supported gate types are shown in Table 1.

Gate type	Description
n	n-transistor
p	p-transistor
x	pass-through transistor object
I	inverter
P	passgate
T	transistor pair

Table 1. Specify the transistor order using a list of gate types

The last three gate types (I,P,T) result in both n- and p-transistors being placed in their respective transistor rows. The details of each type are defined below in the nodes section. A simple specification of an inverter followed by a passgate is shown in Example 2.

```
lappend gates I1 P2
```

Example 2. Create a gate list for an inverter and passgate

nodes

All transistors must have source, drain, and gate node names defined. These provide connectivity information for source/drain sharing and routing. Each gate type has a specific order, as shown in Table 2.

Type	Node name order
n	source drain gate
p	source drain gate
x	source drain gate
I	gate output vss vdd
P	input output n-gate p-gate
T	n-source n-drain n-gate p-source p-drain p-gate

Table 2. Source, drain, and gate node orders depend upon the gate type

The nodes argument is a list-of-lists. Each sublist specifies the node names associated with the corresponding entry in the gates list.

Node names do not have to be unique between transistors. Common source and drain names within the same transistor row will be routed together automatically. Transistor gate names may be specified as a three-element sublist defining separate names corresponding to the channels below and above the gate. The form is:

```
{gatename lo-gatename hi-gatename}
```

The details of each gate type are described below.

n: N-Transistor

The n-transistor specification is:

```
{source drain gate}
```

where the gate element may be defined as the three-element sublist:

```
{general-gate down-gate up-gate}
```

Example 3 specifies two individual n-transistors.

```
# Simple n-transistor specification

lappend nodes {NIN NOUT SEL}

# n-transistor specification using gate sublist

lappend nodes {NIN NOUT {SEL NSEL NPSEL}}
```

Example 3: Specify n-transistors in two ways

p: P-Transistor

The p-transistor specification is:

```
{source drain gate}
```

where the gate element may defined as the three-element sublist:

```
{general-gate down-gate up-gate}
```

Example 4 specifies two individual p-transistors.

```
   # Simple p-transistor specification
   lappend nodes {PIN POUT SELBAR}
   # Less simple specification
   lappend nodes {PIN POUT {SELBAR PSELBAR NPSELBAR}}
```

Example 4. Specify P-transistors

x: Pass-through Transistor

The x-transistor specification is:

```
{source drain gate}
```

The possible values for these variables are shown in Table 3.

Variable	Value
source	{ }
	source-name
drain	{ }
	drain-name
gate	{ }
	gate-name
	{general-gate down-gate up-gate}

Table 3. Specify an x-transistor using these values.

The pass-through transistor provides a way to route feed wires and input wires between transistors within a row. It is used primarily in multi-row layouts (more than one row each of n- and p-transistors). The pass-through transistor has no physical features associated with it -- just three nodes. It does occupy the space of a transistor, which results in a gap between the transistors on both sides of it.

The x-transistor allows node names to be specified as null ({ }). This indicates that there is no wire being routed through the object for that specific node. Example 5 shows a pass-through transistor that routes nothing through the source node and routes SELBAR through the drain and SEL through the gate.

```
Pass-through SEL and SELBAR signals

lappend nodes {{ } SELBAR SEL}
```

Example 5. Specify an x-transistor to pass through SEL and SELBAR signals

I: Inverter

The inverter specification is:

```
{gate output ground power}
```

where the gate element may be defined as:

```
{np-gate n-down-gate p-up-gate}
```

The gate element is the inverter's input net name. When only one value is specified, it is applied to both the n- and p-transistor gates. Separate names may be specified for the net connecting the n- and p-transistors (np-gate), the n-transistor net being routed downward (n-down-gate), and the p-transistor net being routed upward (p-up-gate).

The output element may be defined in the inverter specification as:

```
{n-feed p-feed}
```

The output element is the inverter's output net name. When only one value is specified, it is applied to both the n- and p-transistors. Optionally, different net names may be given to each transistor.

The ground value is applied to the n-transistor and the power value to the p-transistor.

In the single gate name form the name is assigned to both the n- and p-transistor gates and is used to connect the transistors to input trunks in the channel between them.

In the multi-gate name form the np-gate name is assigned to the n- and p-transistor gates for connecting to input trunks in the channel between them. The n-down-gate value is used to connect the n-transistor gate to input trunks in the channel below it, and the p-up-gate value is used to connect the p-transistor gate to input trunks in the channel above it. Example 6 shows two different ways of specifying inverters.

```
# Simple inverter specification

lappend nodes {CLK CLKBAR VSS VDD}

# Less simple inverter specifications

lappend nodes {{CLK NCLK PCLK} {NCLKBAR PCLKBAR} VSS VDD}
```

Example 6. Specify inverters using the nodes argument

Inverter n- and p-transistors can only be placed in consecutively numbered rows.

P: Passgate

The passgate specification is:

```
{input output n-gate p-gate}
```

Table 4 describes the values for these passgate arguments.

Variable	Value
`input`	`{n-feed p-feed}`
`output`	`{n-feed p-feed}`
`n-gate`	`{general-gate n-down-gate n-up-gate}`
`p-gate`	`{general-gate p-down-gate p-up-gate}`

Table 4. Specify a passgate using these arguments

In the simple form, the input and output names are applied to the source/drain feeds of both the n- and p-transistors. The n-gate name is applied to the n-transistor gate name and p-gate name is applied to the p-transistor. Example 7 demonstrates two ways of specifying passgates.

```
# Simple passgate specification

lappend nodes {DA DB SEL SELBAR}

# Less simple specification

lappend nodes {{NDA PDA} {NDB PDB} {SEL NSEL SEL} SELBAR}}
```

Example 7. Specify passgates using the nodes argument

Passgate n- and p-transistors can only be placed in consecutively numbered rows.

T: Transistor Pair

The transistor pair specification is:

```
{n-source n-drain n-gate p-source p-drain p-gate}
```

The arguments are shown in Table 5.

Argument	Value
`n-gate`	`{general-gate n-down-gate n-up-gate}`
`p-gate`	`{general-gate p-down-gate p-up-gate}`

Table 5. Specify a transistor pair using these arguments.

This provides shorthand for the specification of n- and p-transistors separately. It's most useful for when you want better control over objects such as inverters. You use a transistor-pair and specify all nodes individually. Example 8 illustrates two methods for specifying transistor pairs.

```
# Simple transistor pair specification

lappend nodes {NIN NOUT SEL PIN POUT SELBAR}

# Less simple specification

lappend nodes {NIN NOUT {SEL NSEL NPSEL} PIN POUT SELBAR}
```

Example 8. Specify transistor pairs using the nodes argument

Transistor pair n- and p-transistors can only be placed in consecutively numbered rows.

rows

ProAssemble places transistors in rows. A layout may contain any number of transistor rows. Rows are specified by row-number. Row numbers start with zero at the bottom of the layout and increase by one upward for each subsequent row (the second row from the bottom is row 1 and the next is row 2, etc). n-transistors and p-transistors cannot be placed in the same row. All the n-transistor rows must be below the p-transistor rows.

The rows argument is a list-of-lists. Each sublist contains row numbers associated with the corresponding entry in the gates list. When a gate type contains both n- and p-transistors, the n-transistors are specified first.

Specification of the rows argument to ProAssemble is optional. When not specified, the default behavior is for all n-transistors to be placed in row 0 and p-transistors in row 1. Example 9 places the first n-transistor in the bottom row, the first p-transistor in the top row, the second n-transistor in the middle row, and the second p-transistor in the top row.

```
# First N in row 0, first P in row 2

# Second N in row 1, second P in row 2.

lappend rows "0 2" 1 2
```

Example 9. Specify transistor placement using the rows argument

The n- and p-transistors of inverter, passgate, and transistor pair gates can only be placed in consecutively numbered rows. The transistors of n-transistor, p-transistor, and x-transistor gates can be placed in any rows.

widths

Transistor widths should be defined for all transistors. The widths argument is a list-of-lists. Each sublist contains transistor widths associated with the corresponding entry in the gates list. When a gate type contains both n- and p-transistors, the n-transistors are specified first.

The width values are pre-folding. Therefore, when transistors are folded, the widths are reduced to produce the effective width specified. Transistor widths are specified in microns.

The special keyword prefw may be defined for any transistor width in which case ProAssemble will look-up the preferred width in the technology file corresponding to the transistor type. If no widths are provided, ProAssemble will use the preferred width for all transistors.

Pass-through transistors must have a width assigned, but the value is not used, so 0.0 is fine. In Example 10, the first n-transistor and p-transistor are set to the preferred width, while the second n-transistor and p-transistor are set to specific values.

```
lappend widths "prefw prefw" "1.20 3.60"
```

Example 10. Set transistor widths using the widths argument

folds

Any transistor can be folded. Folding is done on each transistor separately treating them as a parallel transistor stack.

The folds argument is a list-of-lists. Each sublist contains the number of folds per transistor associated with the corresponding entry in the gates list. When a gate type contains both n- and p-transistors, the n-transistors are always specified first.

The value indicates the number of transistors in the fold. A value of 1 means "no folds" (one transistor) and a value of 2 means "folded once to produce two transistors." In the statement shown in Example 11, the first n-transistor and p-transistor are not folded, while the second n-transistor and p-transistors are folded once.

```
lappend folds "1 1" "2 2"
```

Example 11. Specify folding instructions using the folds argument

feed_directions

All contacted sources and drains have small feed tabs that go either up or down. The direction depends on which side of the transistor the routing to the feeds will occur. You can let ProGen determine the routing side (and thus the feed side) or force it using the feed_directions argument.

The feed_directions argument is a list-of-lists which defines the feed name, the transistor row, and the direction to force the feed. Each entry in the list has the form:

```
feed-name transistor-row direction
```

Table 6 shows the arguments and describes their values for use with feed_directions.

Argument	Value	Description
`feed-name`	node-name	Source/drain node name
`transistor-row`	number	Transistor row number containing feed-name
	n	row 0
	p	row 1
`direction`	down	Force feed down
	up	Force feed up
`I`	Inverter	
`P`	Passgate	
`T`	Transistor pair	

Table 6. Use feed_directions to specify the direction of feed tabs

Entries are only required for feed directions you want to force. In practice, it is best to avoid specifying feed directions unless ProGen won't do what you want it to do. Example 12 illustrates how to force VSS feeds downward and VDD feeds upward.

```
lappend feed_directions "VSS 0 down" \

                                       "VDD 1 up"
```

Example 12. Use the feed_directions argument to control feed directions

force_src_side

The force_src_side argument provides a way to force the transistor orientation such that the source of the transistor is to the low (left or bottom) or high (right or top) side. This argument is not typically specified, which allows ProGen to decide on the best orientation.

The force_src_side argument is a list-of-lists. Each sublist contains the placement directive for a specific transistor. Each sublist has the following form:

```
gate-name side
```

Table 7 indicates the possible values for each component of the sublists.

Variable	Value	Description
`gate-name`	gate	Name of gate to force
`side`	lo	Force source to low side
	hi	Force source to high side

Table 7. Specify the gate name and the direction to force the transistor orientation.

The gate-name must be a value defined in the gates argument. If the gate name corresponds to a gate type that has both n- and p-transistors then all corresponding transistors are oriented according to the specification.

The source node of the gate is defined in the nodes specification. Example 13 forces the inverter I1 sources to the high side and forces the passgate P2 sources to the low side.

```
lappend force_src_side "I1 hi" \

                                         "P2 lo"
```

Example 13. Use the force_src_side argument to specify transistor orientation

Routing Topology

The routing topology is defined by specifying the channels in which to route, input trunk names and locations, feed names and locations, port information, and the left-to-right order in which to route contacts and feeds (stitch-order). Each of the following ProAssemble arguments contributes to this specification.

chan_ids

The chan_ids argument is a list indicating in which channels routing will be done. Channel numbers begin with zero at the bottom of the layout and incrementally increase by one. Channel 0 is below the first transistor row (row 0), channel 1 is above the first transistor row and below the second transistor row (row 1), channel 2 is above the second transistor row, and so on, depending on how many transistor rows exist.

All other routing arguments are arrays-of-list where each array index indicates the channel in which the information applies. The array index must be a value specified in the chan_ids argument. In Example 14, routing is performed in channel 0 (below N-transistors) and in channel 1 (between N-transistors and p-transistors).

```
lappend chan_ids 0 1
```

Example 14. Specify channels in which routing will be done using chan_ids

chan_in_trunks

This is an array of list-of-lists. Each array entry (list-of-lists) defines the input-trunk information for a channel. The array index indicates the channel in which the list values apply; it must be one of the values defined in the chan_ids argument.

Input trunks require that at least two pieces of information be defined: a trunk name and a track row number. Additional arguments allow the specification of port and trunk type information. The rel_rail argument makes ytrack relative to the rail with id rel_rail.

trunk_name ytrack [port_obj [trunk_obj [width [rel_rail]]]]

Table 8 indicates the arguments that can be specified for each input trunk.

Argument	Value	Description
ytrack	number	Channel track number
	gnd	Track in center of ground rail
	pwr	Track in center of power rail
port_obj	{ }	No port on trunk
	port	Use default port for object port
	obj	Use %Template(port${obj}) as object
trunk_obj	{ }	Use default oject for trunk
	obj	Use $Template(wire${obj}) as object
width	{{ }	Use default trunk width
	default	Use default trunk width
	val	Trunk width in microns
rel_rail	id	Rail id number defined in standard cell template

Table 8. Specify the input-trunk information for a channel using chan_in_trunks

chan_feed_trunks

Array of list-of-lists. Each array entry defines the feed trunk information for a channel. The array index indicates which channel the list applies. The array index must match one of the values defined in the chan_ids argument.

The simplest feed specification contains only a feed name. The system will automatically create a feed between an n- and p-transistor if the node names match the feed name.

Feeds can also attach to input trunks by specifying the input trunk name immediately after the feed name.

Ports can be placed on feeds by specifying port as the third argument in the feed sublist.

Finally, you can override the default feed layer by specifying the layer as the fourth argument in the feed sublist.

feed_name [in_trunk [port [feed_layer]]]

Table 9 lists the arguments for chan_feed_trunks and describes their potential values.

Argument	Value	Description
in_trunk	{ }	Route to input trunk with same name
	default	Route to input trunk with same name
	in-trunk	Input trunk name to route to

Table 9. Use chan_feed_trunks to route channel feeds

Argument	Value	Description
port	{}	No port on feed
	port	Place port on feed
feed_layer	default	Default feed layer
	layer	Force feed on named layer

Table 9. Use chan_feed_trunks to route channel feeds

chan_stitch_order

The stitch order defines the placement order of the poly-contacts of the input trunks and the feed trunks. All transistors that have gate-poly routed to an input trunk within a given channel must have a stitch order specification. Likewise, all source/drain contacts that have a feed through a channel must also have a stitch order specification.

The chan_stitch_order argument is an array-of-lists. Each list defines the stitch order for the corresponding routing channel. The array index indicates the channel to which the list applies; it must match one of the values defined in the chan_ids argument. The stitch order values are shown in Table 10.

Value	Description
lo	Transistor gate route at bottom of channel
hi	Transistor gate route at top of channel
f	Source/drain feed route

Table 10. Use the stitch order values to define placement of poly-contacts

An example of the chan_stitch_order argument is shown in Example 15.

```
lappend chan_stitch_order(1) lo hi f lo hi f
```

Example 15. Specify center channel stitch order for an inverter and passgate

Other Information

ProAssemble allows you to specify a few other miscellaneous things.

cell_width

The cell_width argument allows you to override the default cell width that ProAssemble defines. Most generators allow you to specify this value and pass it through to ProAssemble.

template_type

The template_type argument allows you to override the default object used for the cell border template. Its value is simply a template name that is defined in the technology file.

contacts

The contacts argument provides a way to override the default diffusion contact objects which ProAssemble uses. This is a list-of-lists where each sublist has the form:

```
gate node contact_obj
```

The gate identifies the ProAssemble gate in which the contact will be changed. This must be a value defined in the ProAssemble gate argument list. The node specifies the node in the gate to which the contact change will be applied and contact_obj provides the name of the contact object to use. The contact object must be defined in the technology. An example of the contacts argument is shown in Example 16.

```
lappend contacts {P2 PIN diff_wire_contact} \

                 {P2 POUT diff_wire_contact}
```

Example 16. Switch the passgate contacts to use a special diffusion wire contact

polygate_contacts

The polygate_contacts argument provides a way to override the default transistor gate contacts used for specific transistor gates. The specification is a list-of-lists. Each sublist has the following form:

```
gate node contact_tkey
```

The value of gate identifies which group of transistors defined by the ProAssemble gates argument contains the specific node to override the contact. The node value indicates the specific transistor poly-gate node for which the contact override will be applied. The value of contact_tkey indicates the contact template key (defined in the technology file) to use to look up the override contact object. An example of the polygate_contacts argument is shown in Example 17.

```
lappend polygate_contacts {I1 SELECT conpolyrot}
```

Example 17. Switch the poly contacts on the inverter to use a rotated contact

antenna_diodes

The antenna_diodes argument defines a list of input trunks on which antenna diodes are to be placed. Diodes are placed on either the far left or the far right contacts of the input trunk. Diodes always extend from the placement contact towards the other end of the input trunk. A diode that is directed to extend toward the left is placed on the far right of the trunk. A diode that is directed to extend toward the right is placed on the far left of the trunk. The specification is a list-of-lists where each sublist has the form:

trunk type extdir

Each of these values is described in Table 11.

Argument	Value	Description
`trunk`	name	Name of input trunk on which to place diode
`type`	n	n-type diode
	p	p-type diode
`extdir`	left	Diode extends to the left
	right	Diode extends to the right

Table 11. Specify the antenna diode input trunk, type, and direction

An example of the antenna_diodes argument is shown in Example 18.

```
lappend antenna_diodes {SELECT n left}
```

Example 18. Place an n-diode on the left side of the inverter input trunk

fixed_ports

ProAssemble allows you to specify that a port falls at a specific grid row or column and indicates whether it is the only port that may occupy that row or column. The fixed_ports argument is a list-of-list. Each sublist contains the fixed-port information for a single port. Each sublist has the following form:

port-name {vert-track [ownership]} {horz-track [ownership]}

The arguments for these sublists are described with their possible values in Table 12.

Argument	Value	Description
`port-name`	node-name	Name of node containing port; must match one of the input-trunks or feeds that has a port
`vert-track`	number	Force port on track column
	any	Port in any column (default)
`horz-track`	number	Force port on track now
	any	Port in any row (default)
`ownership`	own	Only this port is allowed on this track

Table 12. Set the port location requirements

Argument	Value	Description
	any	Other ports may use this track (default)

Table 12. Set the port location requirements

The statement in Example 19 forces port IN to reside on vertical track 5 and stipulates that no other ports can reside on that track. It also puts port OUT on horizontal track 3 and allows other ports on that track.

```
lappend fixed_ports "IN {5 own} {any any}"   \

                                    "OUT {any any} {3 any}"
```

Example 19. Use fixed_ports to specify port locations

Procedure Interface

The usage of ProAssemble, the underlying interface used by ProSticks for rapid creation of layout generators, is shown with its defaults in Table 13.

Procedure	Option	Default
ProAssemble	`-gates`	No default value
	`-nodes`	No default value
	`-rows`	{}
	`-cell_width`	default
	`-widths`	prefw
	`-folds`	1
	`-temaplate_type`	stdcell
	`-fixed_ports`	{}
	`-feed_directions`	{}
	`-force_src_side`	{}
	`-antenna_diodes`	{}
	`_contacts`	{}
	`-polygate_contacts`	{}
	`-chan_ids`	{}
	`-chan_in_trunks`	{}
	`-chan_feed_trunks`	{}
	`-chan_stitch_order`	{}

Table 13. ProAssemble usage and defaults

Example: ProAssemble inverter generator

The specification of a sample inverter generator is shown in Example 20.

```
 proc gen_demo_inv {
} {
  #  Identify power and ground nets and force feed directions
  set_nettype vdd "VDD"
  set_nettype gnd "VSS"
  lappend feed_directions "VSS n down" "VDD p up"
  #  Define the transistor topology:
  #
  #     1) place just a simple inverter
  #     2) inverter node names: in=SELECT, out=SELECTBAR,
  #                             gnd=VSS, pwr=VDD
  #     3) transistor sizes (widths)
  #     4) don't fold
  #     5) put N-transistors in row 0 and p in row 1 (same as
default)
  lappend gates I1
  lappend nodes  "SELECT SELECTBAR VSS VDD"
  lappend widths "prefw prefw"
  lappend folds {1 1}
  lappend rows {0 1}
  #  Define the routing topology as:
  #
  #     1) all routing will be done in center channel (1).
  #     2) input trunk with default port object on track #3 to
connect
  #        gates
  #     3) output feed with default port object
  #     4) stitch order = n-tran (lo), p-tran (hi), output-
feed
  lappend chan_ids 1
  lappend chan_in_trunks(1) "SELECT 3 port"
  lappend chan_feeds(1) "SELECTBAR { } port"
  lappend chan_stitch_order(1) lo hi f
  #  Call ProAssemble
  ProAssemble -gates             $gates           \
              -nodes             $nodes           \
              -rows              $rows            \
              -widths            $widths          \
              -folds             $folds           \
              -feed_directions   $feed_directions \
              -chan_ids          $chan_ids        \
              -chan_in_trunks    chan_in_trunks   \
              -chan_feed_trunks  chan_feeds       \
              -chan_stitch_order chan_stitch_order

}
```

Example 20. Inverter Generator

Utility Procedures

Generators can call several procedures before calling ProAssemble. These procedures do not use the flag/value calling convention that ProAssemble uses.

set_nettype

The set_nettype procedure has the following interface:

```
set_nettype nettypes nodes
```

The nettypes argument is a list consisting of any of the items shown in Table 14.

Value	Description
`vdd`	Power net
`gnd`	Ground net
`input`	Input to cell
`output`	Output to cell
`internal`	Internal node
`feed`	Has connected feed trunk
`hasintrunk`	Has connected input trunk (used with power-ground rails)
`contacted`	Must have diffusion contact on node
`beentexted`	Do not text node
`widewire`	Wire has larger width than minimum
`data`	Data signal (datapath cells only)
`control`	Control signal (datapath cells only)

Table 14. The nettypes argument to set_nettype can include these values

The nodes argument is a list of one or more node names to which to apply the net types. The values of nodes must be defined in the ProAssemble nodes argument.

Appendix

Post-Processing Application Note

This document describes the concept behind manipulating layout geometries in post-processing.

Progen has the flexibility to add, change, delete and otherwise modify geometries after compaction in a post-processing phase. Post-processing in progen most commonly is used as a bridge between the implementation of progen features and creating creating legal layout results.

When a user wants to have a particular behavior be evident in the layout that is not explicitly supported, there are usually two choices:

1.)Modify existing system objects and procedures

> Advantages: Most complete way of affecting system behavior. Changes are evident during compaction and guaranteed in the final result.
>
> Disadvantages: Requires fairly extensive knowledge of object interaction and default behavior. Debugging can be complicated by the interaction of many objects with the user defined changes.

2.)Use progen-explicit features to guarantee compatibility with a desired feature, then complete the legal behavior in post-processing.

> Advantages: Very flexible. Post-processing is acted only on geometrical shapes and does not depend on objects properties. Many helper procedures exist to aid with geometry manipulation.
>
> Disadvantages: If setup through explicit progen features causes incompatibility with desired result, there might not be a way of fixing it.

Commonly used post-processing actions:

- Layer adding, fill, overlap
- Tie insertion
- Well jogging
- Contact tiling
- Pin layer,shape definition

To use post-processing, the technology file created by protech must point to an advanced settings file with the following line outside of any procedure. If you didn't use protech these are set in the main technology file.

```
        pro_set procs postprocess   post_process_layout
```

postprocess is a system keyword and this statement associates the procedure name "post_process_layout" as being the post-processing procedure. The user can change the name of "post_process_layout" as long as the name that is used is the name of the procedure for post-processing.

The system will pass in two variables. The cellname and the geolist, therefore the beginning of the post process procedure will look like:

```
proc post_process_layout {

      cellname

      geolist

} {
```

The cellname, as you would expect, is the cellname and the geolist is the collection of geos on all layers from the layout. All of the code below goes in the post_process_layout procedure.

To begin modifying the geometries, it is necessary to get the geometry information from the geolist.

The following two commands are useful for queries to the geolist.

```
pro_gds_spec <layer>
```

and

```
geos_geo $geolist "<gds_number> <gds_type>"
```

The first command returns the gds number and data type for layer. The second command returns the geo in the geolist for that particular layer.

Using the commands above one could get the geo information for nwell, by:

```
set gds(nwell)        [pro_gds_spec nwell]

set geo(nwell)        [geos_geo $geolist $gds(nwell)]
```

Which gets the gds value for nwell in $gds(nwell) and puts the nwell geo information in $geo(nwell).

It is worth noting that geos in post-processing don't have any layer associations themselves. Only when a geo is associated to a layer in the geolist does it have any meaning for output. In the practical sense this means one could use the same geo for making shapes on different layers.

Once the system geo information is obtained the idea is to use the helper procedures to modify the existing geo, or to create additional geos that can be used as masks and use logical operations to create the desired geometry information.

Example: Using existing helper procedure to modify a geo.

```
set gds(nwell) [pro_gds_spec nwell]

set geo(nwell) [geos_geo $geolist $gds(nwell)]

set sized_nwell  [$geo(nwell) size 300]
```

This will upsize geometries in $geo(nwell) by 300 progen units on all sides and place the new geo in $sized_nwell

Example: Using logical operations to get P transistor diffusion geometries only.

```
set get_layers {nwell pdiff}
```

```
foreach x $get_layers {

   set gds($x) [pro_gds_spec $x]

   set geo($x) [geos_geo $geolist $gds($x)]

}

set only_pdiff_tran_and_not_tie_pdiff [$geo(nwell) and $geo(pdiff)]
```

Example: Creating an area around P Transistors so that design rules are not violated on a subsequent fill. (keepout_rule is defined in the design rules procedure, and fill_layer information will need to be set up in the layer mapping procedure)

```
set get_layers {nwell pdiff cellborder fill_layer}

foreach x $get_layers {

   set gds($x) [pro_gds_spec $x]

   set geo($x) [geos_geo $geolist $gds($x)]

}

#get keepout rule

set keepout_rule [scale_float_to_int [pro_get rule keepout_rule]]

#get P transistor diff

set only_pdiff_tran_and_not_tie_pdiff [$geo(nwell) and $geo(pdiff)]

#size pdiff by the keepout rule

set keepout_geo [$only_pdiff_tran_and_not_tie_pdiff size $keepout_rule]

#define fill as being everything inside the cellborder but the keepout
geo

set fill_geo [$geo(cellborder) minus $keepout_geo]

#replace the fill layer geo with the new geo

set geolist [geos_replace $geolist $fill_gdslayer $fill_geo]
```

Example: Make a mask by creating a geo with a geometry for which you know the coordinates.

```
set make_my_geo [geo]

$make_my_geo add_rect -300 -300 300 300
```

Creates a box centered around the orgin 600 progen units on each side [left bottom right top]

After the geo manipulations are complete it is necessary to write back the changes to the system geolist with the following command.

```
   pro_set_geolist $geolist
```

With some knowledge of the technology and some creativity on the part of the user to identify the geometries of interest, post-processing is a very powerful way of making changes to the compacted result.

Putting it all together

The following will replace all diffusion on a single gds number. Normally progen will have seperate layers with seperate rules for pdiff and ndiff. Some technologies use only one diffusion layer. The following code will take all of the diffusion geometries and place them on one diffusion layer. This is useful for when p and n transistors have different rules (whereby the user can use the system disticintion of two seperate layers) but the diffusion is required to be on one layer.

```
proc post_process_layout {

   cellname

   geolist

} {

#get layer gds and geo information

set gds(pdiff)        [pro_gds_spec pdiff]

set geo(pdiff)        [geos_geo $geolist $gds(pdiff)]

set gds(ndiff)        [pro_gds_spec ndiff]

set geo(ndiff)        [geos_geo $geolist $gds(ndiff)]

#Geo that will be the new geo for a single diffusion layer (which in this
case happens to be the same as ndiff)

set alldiff [$geo(pdiff) or $geo(ndiff)]

set geolist [geos_replace $geolist $gds(ndiff) $alldiff]

#One way of deleting the geo on pdiff

set blank [geo]

set geolist [geos_replace $geolist $gds(pdiff) $blank]

#Update the system geolist

pro_set_geolist $geolist

}
```