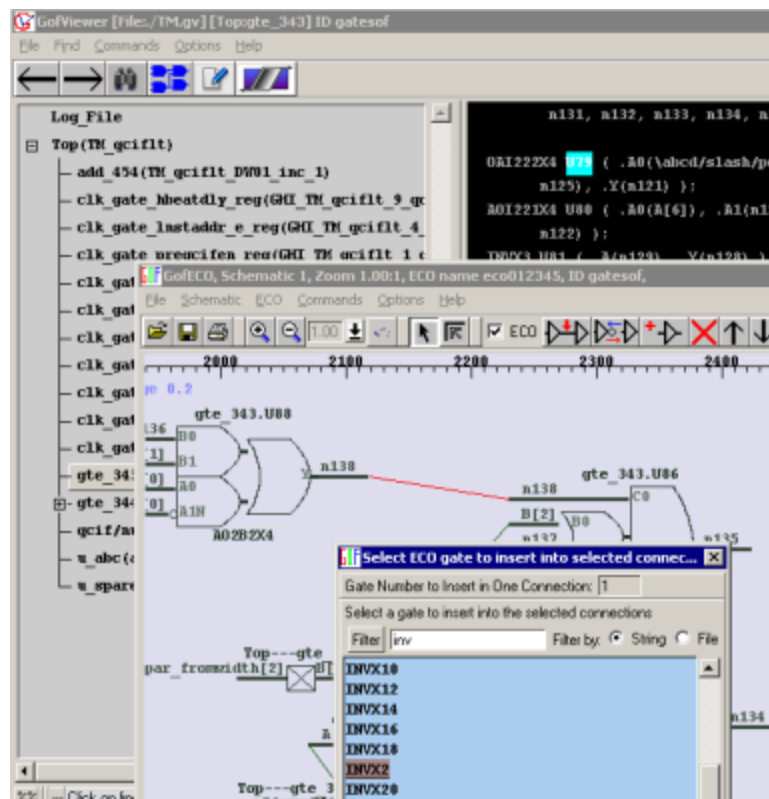


Gates on the Fly Manual

Copyright © 2013, SynaptiCAD Sales, Inc.



Gates on the Fly Manual

Copyright Copyright © 2013, SynaptiCAD Sales, Inc., version 2

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: July 2013 in (whereever you are located)

Netlist Processing and Editing

Gates-on-the-Fly or GOF

This is the manual for SynaptiCAD's Gates-on-the-Fly, GOF, an EDA tool for processing large Verilog netlists and making changes to synthesized code.

Table of Contents

Gates-on-the-Fly Introduction and Quick Start Guide	9
Step 0: Setup Your License.....	10
Step 1: Write a Script File to Launch GOF.....	11
Step 2: Locate Sections of interest in the design in GofViewer.....	11
Step 3: Investigate gates using GofTrace schematic windows.....	12
Step 4: Use GofECO to graphically edit the netlist gates and connections.....	13
Step 5: Use GofCall for automatic ECOs and Checking the Schematic.....	14
Step 6: Check the Design using layout viewer and other interfaces.....	15
Chapter 1: Launching GOF with a Batch script	17
1.1 Writing a Batch Script.....	17
1.2 Concepts for GOF files.....	18
1.3 GOF Command Line Options.....	19
1.4 Example Scripts to Launch GOF.....	21
Chapter 2: GofViewer - text viewer	24
2.1 Moving within GofViewer.....	24
2.2 Search the Netlist.....	27
2.3 Report area, leakage, leafs, & submodules.....	28
2.4 Matching a Net from RTL to Synthesis.....	31
2.5 Diff Utility for Netlist files.....	34
2.6 Analyzing Timing Violations.....	35
2.7 GofViewer Menus and Options.....	38
Chapter 3: GofTrace - schematic viewer	41
3.1 Open a Schematic Window.....	41
3.2 Show Fan In/Out of a Gate.....	45
3.3 Show Circuit between two Gates.....	47
3.4 Search & List Info for Schematic.....	48
3.5 Place and Route Schematic.....	50
3.6 Display Settings and Comments.....	52
3.8 Accessing SDF timing files.....	56
3.9 Layout Viewer.....	57
3.10 Printing a Schematic.....	59
3.11 GofTrace - Menus and Buttons.....	60

Chapter 4: GofECO - schematic editor 65

4.1 Enable ECO mode, Undo, and Save.....	65
4.2 Insert, Replace, or Add a Gate	67
4.3 Add or Delete Connections.....	70
4.4 Drive Strength Adjustment.....	71
4.5 Logic Cone ECO.....	71
4.6 Metal-Only ECOs.....	73

Chapter 5: GofCall - netlist processing 75

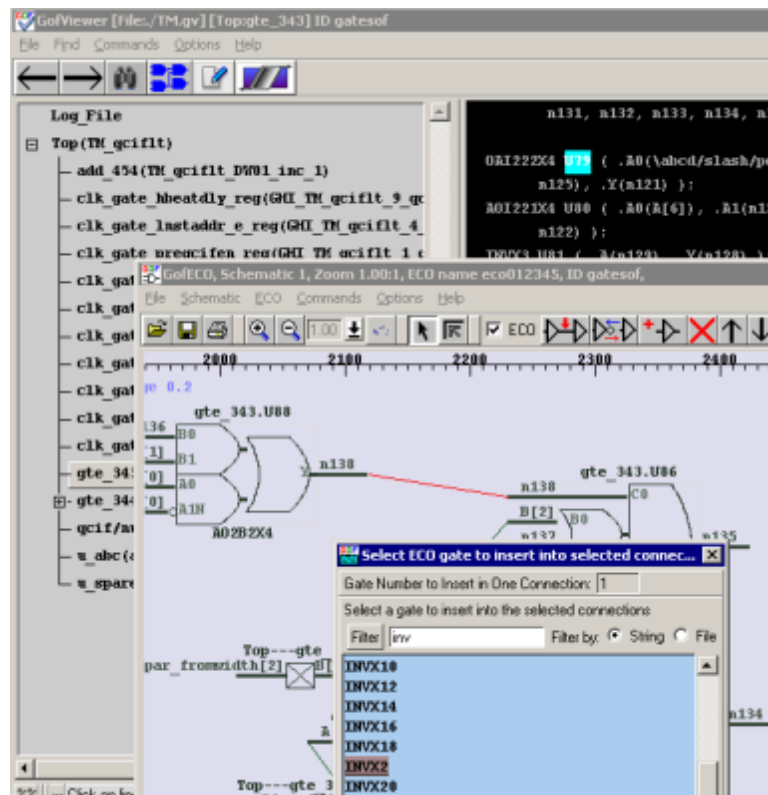
5.1 Interactive window & batch files.....	75
5.2 Command line text mode.....	77
5.3 GofCall API List.....	78
buffer	80
change_gate	80
change_net	81
change_pin	82
change_port	83
check_design	84
compare_nets	84
del_gate	85
del_net	85
del_port	85
exist_inst	85
exist_wire	85
fix_hold	85
fix_logic	86
fix_setup	87
get_cell_info	87
get_cells	88
get_conns	89
get_coord	90
get_definition	90
get_driver	90
get_drivers	91
get_instance	91
get_instances	92
get_leaf_pin_dir	92
get_leafs_count	92
get_leaf_types	93
get_lib_cells	93
get_loads	93
get_logic_cone	94
get_match_nets	94
get_modules	94
get_net_of	95
get_nets	95
get_path	95
get_pins	96
get_ports	96

get_ref	97
get_resolved	97
get_roots	97
get_spare_cells	97
gexit	98
gprint	98
is_leaf	98
is_seq	99
lv_search	99
map_spare_cells	99
new_gate	100
new_net	100
new_port	101
place_gate	102
place_port	102
pop_top	102
push_top	102
read_def	102
read_design	103
read_file	103
read_library	104
rename_net	104
report_eco	104
report_spares	104
replace_logic_cone	105
run	105
sch	105
set_buffer_distance	106
set_exit_on_error	106
set_exit_on_warning	106
set_invert	106
set_keep_format	106
set_leaf	107
set_log_file	107
set_max_lines	107
set_max_loop	107
set_mod2mod	107
set_mu	108
set_power	108
set_preserve	108
set_quiet	108
set_tiehi_net	108
set_tielo_net	108
set_top	109
set_tree	109
set_verbose	109
setup_eco	109
start_gui	110
stich_scan_chain	110
strict_syntax	110
undo_eco	110
write_dcsh	110
write_soc	110
write_spare_file	111

write_tcl	111
write_verilog	111
5.4 Examples of GofCall Scripts.....	112
Chapter 6: Waveform Viewer Support	116
6.1 Setup SynaptiCAD's Waveform Viewer.....	116
6.2 Load VCD and Launch Waveform Viewer.....	117
6.3 Annotate Logic States on Schematic.....	118
6.4 Displaying Waveforms in Waveform Viewer.....	119
6.5 Create Waveform Restore File.....	120
Index	123

Gates-on-the-Fly Introduction and Quick Start Guide

Gates-on-the-Fly (GOF) graphically analyzes and edits large Verilog netlists that have been generated from a synthesis or layout tool. Netlists sometimes require changes to either meet timing closure specifications, fix functional logic bugs, or to repartition a design. Using GOF, you can easily find and view specific logic cones in your design on a schematic to visualize just the paths you need to see without unnecessary clutter. GOF also simplifies mapping from RTL level constructs to their gate-level equivalents, so that you can pinpoint the locations where changes need to be made. GOF's ECO mode supports both graphical and script-based editing features for tracking ECO changes. Metal-only ECO operations are also supported with an automatic spare gates flow.



GOF has four main parts: a netlist browser (GofViewer), a schematic viewer (GofTrace), a netlist editor (GofECO), and a scripting program for automating changes (GofCall). You can also debug a design using one of SynaptiCAD's Waveform Viewers to annotate GofTrace schematics with state information from a waveform file (either from a simulation debug session in progress or from a VCD file created during an earlier simulation of the design). Below is a typical design flow using GOF:

- [Step 0: Setup Your License](#)
- [Step 1: Write a Script File to Launch GOF](#)
- [Step 2: Locate Sections of interest in the design in GofViewer](#)
- [Step 3: Investigate gates using the schematic display of GofTrace](#)
- [Step 4: Use GofECO to graphically add, replace, insert, or delete parts in the netlist](#)
- [Step 5: Use GofCall for automatic ECOs and checking the Schematic](#)
- [Step 6: Check the Design using the layout viewer and other design interfaces](#)

Step 0: Setup Your License

First you should install the Gates-on-the-fly (GOF) software and then install the license.

Installing Node-Locked Licenses

- If you have not received a license file already, you will need to request a license from our sales department at sales@syncad.com. For a node-locked license, you will need the hostid of your system in the email request. To obtain the hostid of your system, run GOF, and select the **Help > Read Ethernet MAC Address** menu.
- Once you've receive a license file (typically named `goflic.dat`), place this file in the SynaptiCAD product suite's top-level install directory. This directory is usually `C:\SynaptiCAD` if you installed in the default location. When you run Gates on the Fly, it will find this license automatically.

Installing Floating Licenses and the License Server

- Download the GOF floating license server.
 - For Linux: `ftp://morgen.syncad.com/pub/synapticad/current/gatesonthe-fly/goflm.tar.gz`
 - For Windows: `ftp://morgen.syncad.com/pub/synapticad/current/gatesonthe-fly/goflm.zip`
- In order to obtain a floating license, you will need the MAC address of the license server. You can get this by running `"goflm -mac_address"`.
- Once you've receive a floating license file (typically named `license_gof.lic`), you may place the license server executable and the license file in any directory. The first line of this file will look like:

```
SERVER <hostname> <hostid> <port>
```

- You should edit this file and change the hostname to the actual hostname of your license server. You may also choose to change the port number (port number defaults to 27200). Do not change the hostid.
- To start the license server, run:

```
goflm -license /path/to/license_gof.lic
```

- The license server has other options as follows:

```
goflm [-help] -license license_file [-o logfile] [-timeout seconds]
```

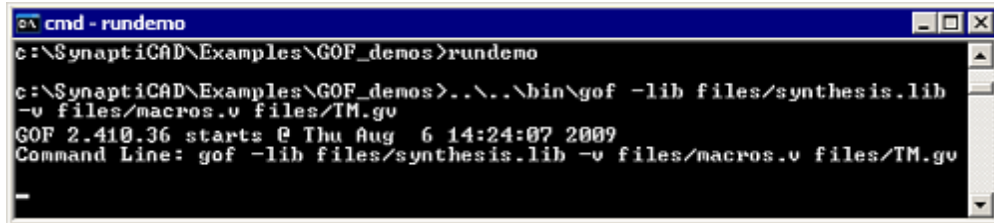
- **-license <file>**: Specify license file location to the license server
- **-help**: Print out this message
- **-mac_address**: Read and display the MAC Address of the current machine
- **-o <logfile>**: Specify log filename (defaults to `goflicense.log`)
- **-timeout seconds**: Specify the timeout (seconds) for the server to kill the client connection. If a client connection has no response within the timeout time specified, the connection will be terminated. The value should be larger than 100.

Using a Floating License

To use a floating license on your local machine, you will need to know the port number and hostname of the license server. On each system where GOF will be run, set the environment variable `GOF_KEY_FILE` to `[port_number]@[hostname]` of the license server. For example, if the license server is running on `license.syncad.com` on port 27200, set `GOF_KEY_FILE=27200@license.syncad.com`.

Step 1: Write a Script File to Launch GOF

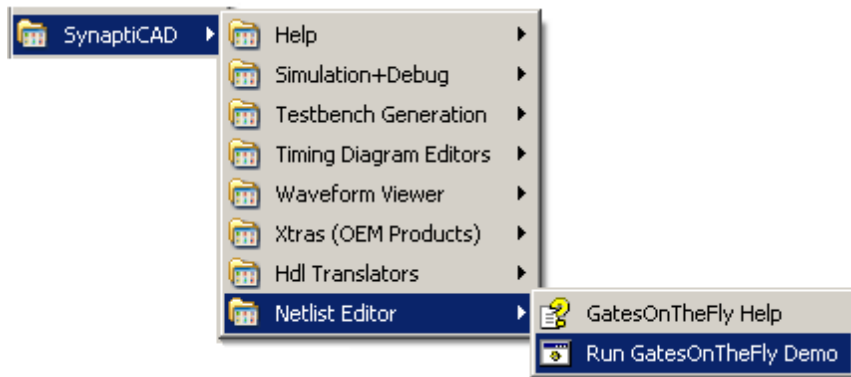
GOF is usually launched with a batch script file that describes the design files, design libraries, and options. The exact syntax and options are covered in the next Chapter [Launching GOF with a Batch script](#). If you are trying GOF for the first time, you can close the tool now and re-launch it using the **rundemo** batch file that is located in the install directory under **Examples > GOF_demos**. This will load GofViewer with an example design.



```
cmd - rundemo
c:\SynaptiCAD\Examples\GOF_demos>rundemo

c:\SynaptiCAD\Examples\GOF_demos>..\..\bin\gof -lib files/synthesis.lib
-v files/macros.v files/TM.gv
GOF 2.410.36 starts @ Thu Aug 6 14:24:07 2009
Command Line: gof -lib files/synthesis.lib -v files/macros.v files/TM.gv
```

The **rundemo** batch script can also be ran from the **Start** menu on windows machines, under the **SynaptiCAD > Netlist Editor > Run GatesOnTheFly Demo** entry.

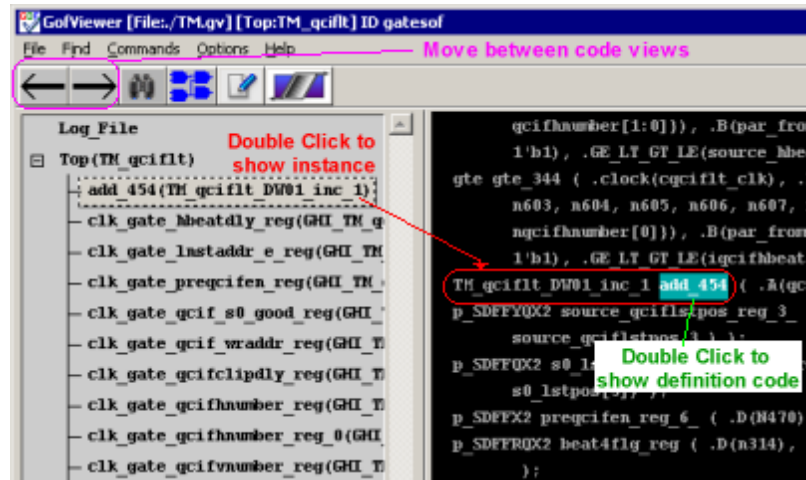


The **rundemo** batch script loads three files. The **-synlib ./synthesis.lib** option loads the technology library that contains physical information for leaf cells in the netlist. The **-v ./macros.v** loads a Verilog simulation library that contains module definitions for leaf cells (leaf cells are primitives that are viewed as uneditable units by GOF). The **./TM.gv** argument loads the Verilog netlist of the design which you will be viewing and performing ECOs on. When writing a batch script, it is important to appropriately specify the type of file being loaded, because this determines how GOF will use the information in the file.

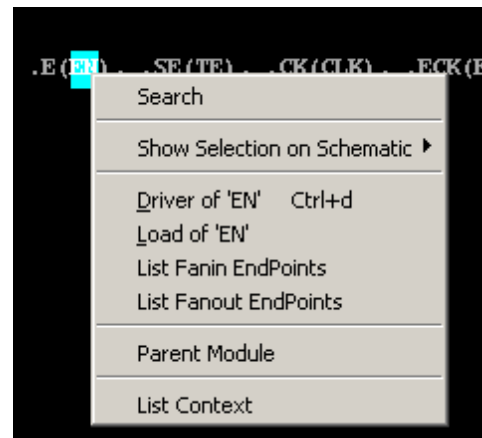
Step 2: Locate Sections of interest in the design in GofViewer

When trying to locate a section of interest (e.g. a logic cone) in a design in GofViewer, try to locate at least one gate or net within the logic cone, then send it to a schematic window. Once you have opened a schematic window with a relevant gate, it takes only a middle mouse click to expand from that gate to show all the drivers of its inputs and the loads that it drives. See [Chapter 2: GofViewer - text viewer](#) for a full feature overview.

- Press the **plus** and **minus** icons to expand/collapse the hierarchy tree under a module instance.
- Double left click** on a hierarchy tree module instance to display the source code where that instance is declared in the netlist text window.



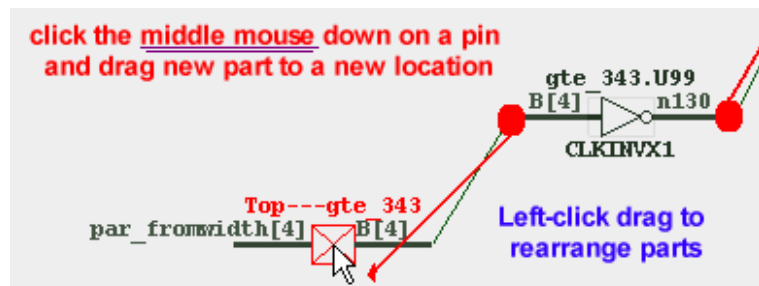
- The big **Arrow** buttons in the top left corner move you back and forth between the different modules that you have viewed, so there is no fear that you will lose your place when looking at a particular object.
- In the netlist text window, **double left click** to view more information about an object. For example, double clicking on a leaf cell will display the gate in a schematic window, double clicking on the type of the leaf cell will open a blue window with the definition of the cell, and double clicking on a hierarchical module instance will take you to the module's definition in the netlist code.
- Right click** to open context menus which can generate reports and send gates to a schematic of your choice.
- You can also select a set of gates and **drag and drop** them onto an open schematic window using the left mouse button.



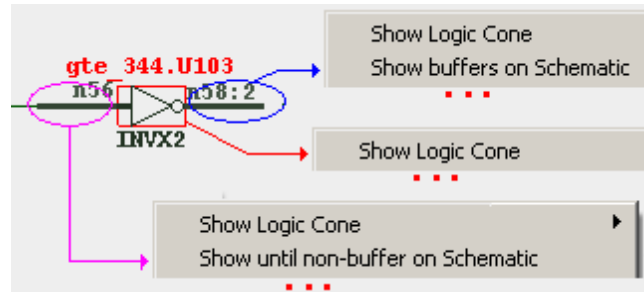
Step 3: Investigate gates using GofTrace schematic windows

Once you get a gate shown on a schematic, you can use the middle mouse button and the right click context **Show** menus to draw more of the gates that are connected to that gate. See [Chapter 3: GofTrace - schematic viewer](#) for a full feature review.

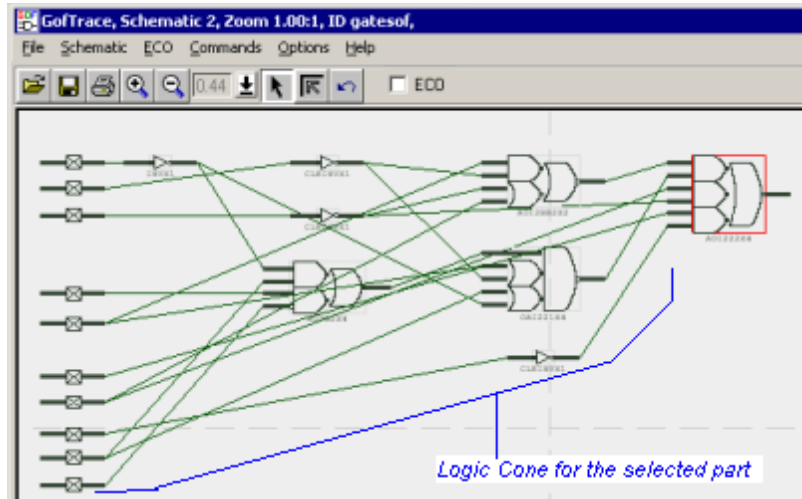
- Press the **middle mouse** down on a **gate pin** to display the fan in/out, then drag the mouse to a new location and release to complete the placement.
- You can rearrange gate placement by **left mouse** click and drag.



- Left click on either an **input pin**, an **output pin** or a **gate symbol itself** to select it.
- Next, right click to open the context menu for that particular point on the symbol. The **Show** menu will add more symbols to the schematic.



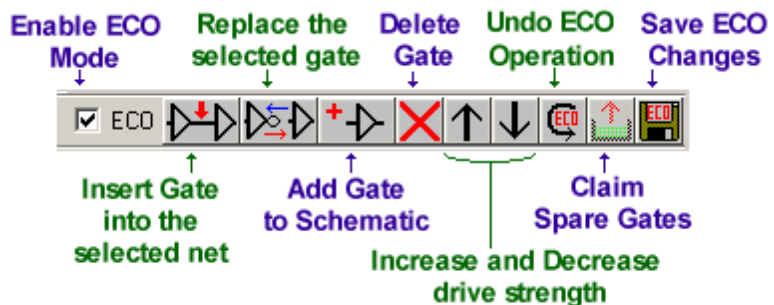
- **Show Logic Cone** adds all of the gates that drive all of the input pins of the selected gate.
- Right click and choose one of the **Auto Place and Route** menus to layout the schematic.



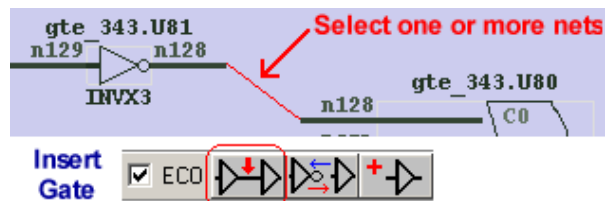
Step 4: Use GofECO to graphically edit the netlist gates and connections

GofECO is a graphical method of changing a netlist. You can add, delete or replace gates. You can also add or delete connections between gates. See [Chapter 4: GofECO - schematic editor](#) for a full feature review.

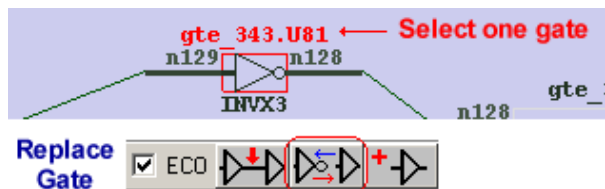
- In a schematic window, check the **ECO** box on the button bar to enable changes to the netlist.
- In this mode, the ECO button bar will appear and the schematic will change colors.



- **To Insert a gate** into a net, select one or more **nets** and then press the **Insert Gate** button. This opens a dialog where you pick the gate to insert from a list of the available leaf cell types.



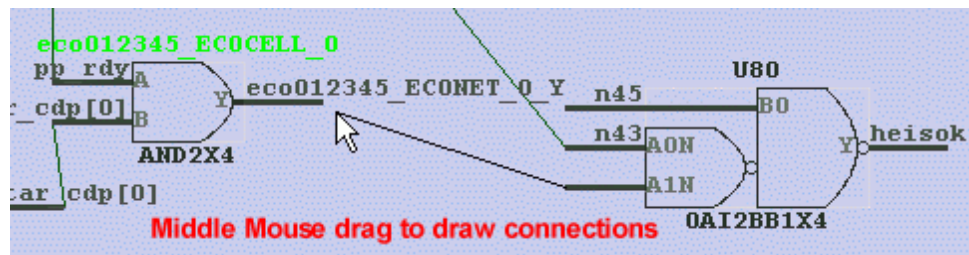
- **To Replace a gate** with another type of gate: select the **gate**, press the **Replace Gate** button, then select the desired leaf cell type from the list that appears.



- **To Add a gate** to the schematic, press the **Add Gate** button. There is no need to select any gates or nets. This opens a dialog where you pick the gate to add from a list.



- Use middle mouse click and drag operations to create connections from gate inputs to the outputs of other gates. Note: the reverse operation, going from a gate output to a gate input is not allowed by GofECO.



Step 5: Use GofCall for automatic ECOs and Checking the Schematic

Using GofCall's programming API, complicated ECOs can be performed with only a few lines of scripting code. The API lets the user focus on the logic changes, without having to specify all the details like adding hierarchical ports or naming new nets. The GofCall API statements can be entered through an Interactive Command window or executed using a batch file.

Below is an example of a three line script that borrows a gate from the spare gate list and makes the connections needed to logically connect up the gate using `change_pin` API call. The first line connects up the A input of `u_spares.so` to the Y output of `gte_344.U101`, the second line connects up the B input of `u_spares.so` to the Y output of `gte_344.U143`, and the third line connects up the D input of `gte_344.gte_con_reg` to the Y output of `u_spares.s0`. GofCall will automatically take care of all the tedious aspects of naming nets and creating ports during the connection process. See [Chapter 5: GofCall -netlist processing](#) for a full list of the GofCall API commands.

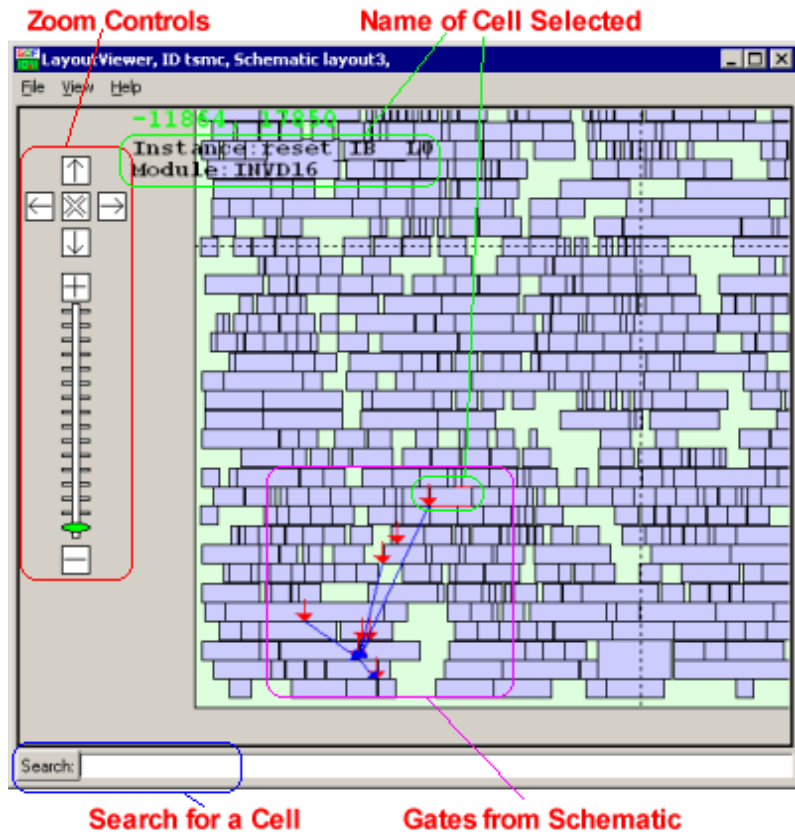
```
change_pin("u_spares/s0/A", "gte_344/U101/Y");
change_pin("u_spares/s0/B", "gte_344/U143/Y");
change_pin("gte_344/gte_con_reg/D", "u_spares/s0/Y");
```

Step 6: Check the Design using layout viewer and other interfaces

GOF has several utilities for viewing a design and finding more information about each cell.

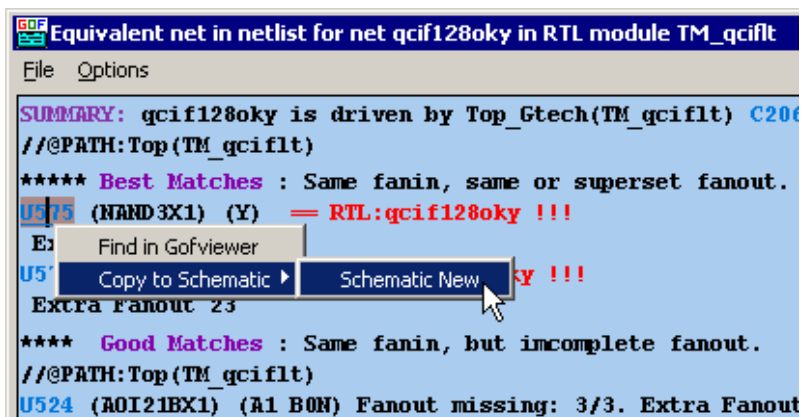
Layout Viewer

Gof can display gates from a schematic in a layout viewer that shows actual gate locations in the backend layout (see [Section 3.9: Layout Viewer](#)).



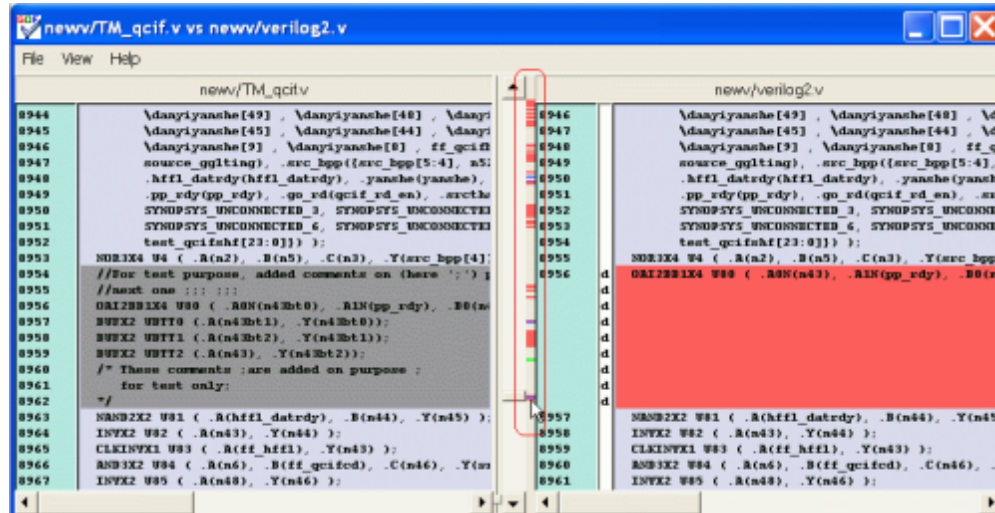
Match RTL nets to netlist nets

GOF has a feature that helps match nets in the RTL code to nets in the netlist code. The results are classified based on how closely they match. See [Section 2.4: Match Net from RTL to Synthesis](#).



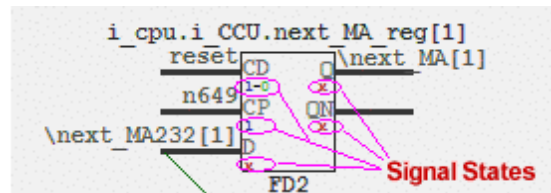
Find the differences between two source code files

GOF has a difference window for comparing netlist code changes (see [Section 2.5: Diff Utility for RTL Files](#)).



Show VCD waveforms and logic states on the schematic

Using one of SynaptiCAD's waveform viewers or editors, GOF can take simulation waveform data and annotate a schematic with the logic values (see [Chapter 6: Waveform Viewer Support](#)).



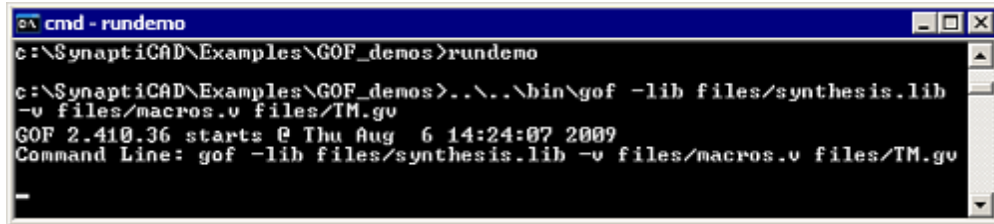
Show Timing Violations on the schematic and other windows

GOF can help analyze timing violations. Once you get a Prime Time file you can use GOF to look at the timing violations using the schematic window, and several different listing windows (see [Section 2.4: Process Timing Violations](#)).



Chapter 1: Launching GOF with a Batch script

GOF is primarily used to investigate the gates of a design after the design has been synthesized or comes back from a layout tool, so it is optimized to work with very large files with many library files. Generally you will load your design using a batch script file that you have created with a text editor. The batch file specifies the information contained in each file that is loaded using command line options. The type of file determines how GOF will treat the information in the file. Very simple designs can be loaded with the menu functions, but a batch script offers more options and flexibility.



```

cmd - rundemo
c:\SynaptiCAD\Examples\GOF_demos>rundemo

c:\SynaptiCAD\Examples\GOF_demos>..\..\bin\gof -lib files/synthesis.lib
-v files/macros.v files/TM.gv
GOF 2.410.36 starts @ Thu Aug 6 14:24:07 2009
Command Line: gof -lib files/synthesis.lib -v files/macros.v files/TM.gv

```

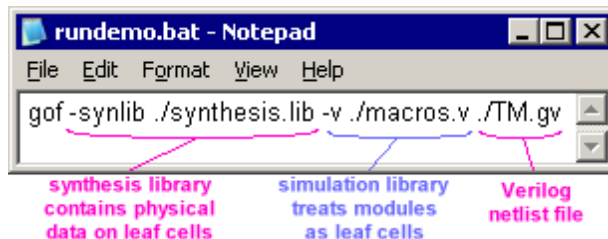
If you are new to GOF, you can use the **rundemo.bat** file to launch GOF with an example design that will allow you to test some of the features of the program. The **rundemo** batch file can also be run from the **Start** menu on windows machines, under the **SynaptiCAD > Netlist Editor > Run GatesOnTheFly Demo** entry.

1.1 Writing a Batch Script

GOF is launched using a batch script file that passes in all of the netlist code files, technology library files (i.e. synthesis libraries), simulation library files, and run time options. When writing the batch file, it is important to specify the type of file that is being loaded because this determines how GOF will treat the information in the file. In particular, Verilog files can be loaded as either netlist files or as simulation library files and there is no way for GOF to determine which way you want to treat such files. Netlist files are editable whereas simulation libraries are not. You also need to indicate whether to treat modules in simulation libraries as hierarchical modules or as leaf cells (see [Section 1.2: Concepts for GOF files](#) for more information on this topic).

Write a batch file:

- Open a text editor (e.g. Notepad or vi) and type in the GOF command which lists all of the files to load and the options in the form of: `gof [options] netlists`



- These options are defined in [Section 1.3: GOF Command Line Options](#)
- Save the file with a **.bat** extension for Windows or a **.sh** extension for Unix.
- From a command prompt, run the script to launch GofViewer with the design loaded.

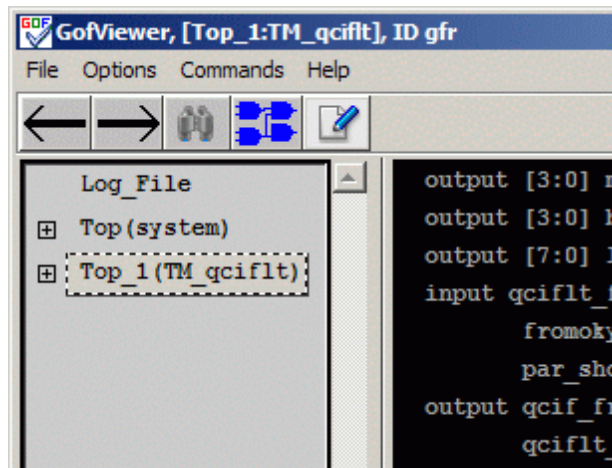
```

c:\SynapticAD\Examples\GOF_demos>rundemo

c:\SynapticAD\Examples\GOF_demos>...\bin\gof -lib files/synthesis.lib
-v files/macros.v files/TM.gv
GOF 2.410.36 starts @ Thu Aug 6 14:24:07 2009
Command Line: gof -lib files/synthesis.lib -v files/macros.v files/TM.gv

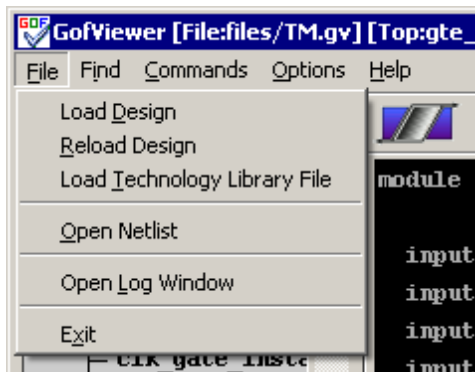
```

- See [Chapter 2: GofViewer](#) for more information on moving through the file. The first top level instance in the netlist will be identified as **Top** followed by its module type in parenthesis (e.g. **Top(system)** in the picture). If the netlist contains more than one top level instance, or additional netlists are loaded subsequently, these top level instances will be numbered consecutively (e.g in the pictured example, **Top_1 (TM_qciflt)**, is a second top level instance).



Refreshing a design or loading a simple design:

- If the design files get modified during a GOF session, you can use the **File > Reload design** to refresh the netlist.
- If you have launched GOF without using a batch file, then **File > Open netlist** can open a netlist and build a new tree. This is only valid for designs using the same library files.



1.2 Concepts for GOF files

When working with GOF for the first time, understanding a few key concepts discussed below will help you get working productively. For example, when writing your batch files it is important to know that how a file is loaded determines how GOF will treat the code inside it.

Netlist files are loaded by listing them after GOF on the command line. The module definitions inside a netlist file will be treated as hierarchical models and will show up in the tree window in GofViewer.

Hierarchical models can contain a combination of leaf cell instances and instances of other hierarchical models. GofViewer navigates through the netlist code using the hierarchical models tree and the associated netlist text window. You can generate reports about the leaf cells that make up a hierarchical model, but you cannot directly display hierarchical model symbols in a schematic window (only the leaf cells that make up a hierarchical model can be shown as symbols on a schematic).

Leaf cells represent pre-layed out structures of the design. Leaf cell definitions are passed to GOF either from simulation libraries (i.e. Verilog source code files) or from technology libraries (e.g. .lib synthesis files). Library cell definitions are displayed in blue list windows instead of in GofViewer's black netlist window to distinguish them from hierarchical models. Unlike hierarchical models, leaf cell symbols can be shown directly on a schematic. Leaf cells are also the gates that will be available when performing an ECO operation to change the design schematic.

Technology Library files are loaded using the **-synlib** command-line option. Technology library files contain physical information about leaf cells of the design. When leaf cells are defined using a technology library, GOF can use the library data to calculate information about the design such as area and leakage current. GOF also uses technology library data to distinguish between combinational gates and flip-flops for operations such as display of logic cones in schematic windows.

Simulation Library files are loaded using either **-v**, **-vn**, **-y**, or **-yn** command-line options (these options correspond to the common **-v** and **-y** options used by many Verilog simulators). Simulation library files contain Verilog code for module definitions for library models provided by an ASIC/FPGA vendor. These module definitions will be treated as leaf cells if loaded with **-v** or **-y**, or as hierarchical models if loaded with **-vn** or **-yn**.

- The **-v** and **-vn** options assume that the library file contains many modules.
- The **-y** and **-yn** options require a path to a directory containing a library file for each library module definition (with each library filename matching the name of the module it defines). This library directory will be searched for a matching file whenever GOF's Verilog parser encounters a leaf cell type for which it can't find a definition elsewhere.

Note: The key difference between passing Verilog module definitions using the **-vn** simulation library file option instead of passing them as a netlist file is that uncontained modules in a netlist will be treated as top level hierarchical models in the design, whereas uncontained modules in a library file will not appear in the design at all (because the current design doesn't use these models).

1.3 GOF Command Line Options

Below is a list of command line options that you can use in batch scripts that launch GOF:

Loading Netlist files and Library Files

See [Section 1.2: Concepts for GOF files](#) for an explanation of how the models in these files are treated.

netlists	Load Verilog netlist file(s). Multiple netlist files can be opened in GOF by listing them separated by spaces.
-synlib technology_library or -lib technology_library	Specify a technology library (synthesis library) which contains parameters for leaf gates. Multiple -synlib options can be specified if the design has more than one technology library file. The -v option below should only be used to provide definitions for any leaf cells not covered by technology library files. You can also load a technology library after launching GofViewer by selecting the File > Load Technology Library File menu option.
-v simulation_library	Specify a simulation library file which contains verilog definitions for leaf cells (e.g. AND2X4). There can be multiple -v options if the design has more than one simulation library. Both -v and -synlib options are for leaf gate definitions and both options can both be used in a single design, but technology libraries enable more

features in GOF such as logic cone extraction. Simulation libraries should generally only be used for leaf cell definitions when technology libraries are not available for those cells. Multiple -v options can be used.

-vn <i>simulation_library</i>	Similar to the -v option, except that modules defined by this library are treated as hierarchical modules instead of as leaf cells by GOF. Multiple -vn options can be used.
-y <i>library_directory</i>	Specify a "Verilog-style" library directory containing files that define individual leaf cells (one file per leaf cell, with the filename for each cell matching the cell name). +libext+.v should be used with -y option. Multiple -y options can be used.
-yn <i>library_directory</i>	Similar to -y option, except that modules found in this library directory should be treated as hierarchical modules instead of as leaf cells by GOF (e.g GOF can show these modules in schematic form). Multiple -yn options can be used.
+libext+ext1+ext2+...	Specify library file extensions to use when searching for module definitions in library directories. For example, to indicate that .v, .vg and .lib are valid file extensions, use +libext+.v+.vg+.lib. Only useful when at least one -y option has been specified. Multiple +libext options can be used.

Controlling how GOF works:

-h	print out this info
-id mydesign	Optionally specify a design id that will be shown in the title bars of GOF windows.
-o <i>log_file</i>	Sets log file name. By default, the log file will be named gatesof.log
+define	Define one or more preprocessor macros from the command line.
+PARAMETER0	
+PARAMETER1	
-f <i>file_list_file</i>	Load all the files and any command line options listed in specified file.
-textbutton	Replaces the ECO graphical image buttons with a set of buttons that have text displayed on the face of the buttons.
-usermenu	Adds user-created menus to the GofTrace window under a main menu called UserMenu. The file contains a list of the names of the new menus and provides for the passing of information about the schematic to another program. Here is an example of a file:
<i>file_name.post</i>	

```
List Selected---SEL---wire,inst,pin---notepad.exe FILE
List Definition---SEL---def---notepad.exe FILE
```

Layout Viewer Support:

These files are used by the layout viewer to show where the gates appear in the backend layout (see [Section 3.9: Layout Viewer](#))

-def <i>def_file</i>	Load a DEF (Design Exchange Format) file. Multiple -def options can be used. .
-lef <i>lef_file</i>	Load a LEF (Library Exchange Format) file. Multiple -lef options can

be used.

-pdef pdef_file

Load a PDEF (Physical Design Exchange Format) file. Multiple -pdef options can be used.

Waveform Viewer Support:

Waveform files can be used to show logic states on the pins of gates in a schematic. The logic states will also be displayed as waveforms in a SynaptiCAD waveform viewer (see [Chapter 6: Waveform Viewer Support](#)).

-vcd waveform_file

Load in a waveform file for schematic annotation. Note that the file can be in any waveform format recognized by the associated waveform viewer, it does not have to be a VCD file (e.g. BTIM files are supported by SynaptiCAD's waveform viewers).

Match an RTL net to a net in the Synthesized Netlist:

GTECH files are used to help match wires from the RTL code to wires in the synthesized netlist (see [Section 2.4: Match Net from RTL to Synthesis](#)).

-gtech gtech_file

Load a GTECH file for wire matching. Multiple -gtech options can be used.

Metal-Only ECO Support:

GOF has a metal-only mode for doing ECOs where the only allowed changes are in the metal layers. The spare cells list shows unattached gates that exist on the chip (see [Section 4.6: Metal Only ECOs](#)).

-sparelist

spare_cells_list_file

Load in an optional spare cells list from a file. If no spare cell list is loaded, you cannot add new gates during metal-only ECOs.

Logic Cone ECO Support:

Logic Cone ECO scripts change the entire logic cone that drives a particular gate input and replaces it with a newly re-synthesized netlist (see [Section 4.5: Logic Cone ECO](#)).

-ecofile eco_file.eco

To perform a logic cone replacement, setup by eco_file.eco

GofCall Support:

GofCall is a programming interface for performing large complicated ECO changes.

-run script_file.pl

Invoke GofCall to run a netlist processing script. Gof will stay in shell mode when the script finishes (see [Section 5.1: Interactive window & batch files](#)).

-shell

Run in text mode with a shell prompt. GofCall APIs can be run in interactive mode from the shell (see [Section 5.2 Command line text mode](#)).

1.4 Example Scripts to Launch GOF

There are several prewritten scripts that are shipped with GOF that let you launch GOF and test out a specific feature. These are located under the **SynaptiCAD\Examples** directory in either the **GOF_features**, **GOF_demos**, or **GOF_logic_cone_eco** subdirectories.

- **rundemo.bat** is the main demo design that is shipped with GOF. It is a large design that lets

you generate schematics and test most of the features in the Viewer, Trace, and ECO windows. See [Section 1.1: Writing a Batch Script](#).

- **rundemo2.bat** is the same as rundemo except that it uses a simulation library instead of a synthesis library. See [Section 1.2: Concepts for GOF files](#) for a discussion on the difference in the library files.
- **case_gtech.bat** launches GOF with a design that has a gtech file and the RTL Code so that you can experiment with the RTL-to-Netlist Matching feature which eases the process of locating synthesized nets. See [Section 2.4: Match Net from RTL to Synthesis](#).
- **case_layoutview.bat** launches GOF with layout files required to view the physical layout of cells in the design so that you can experiment with the layout viewer. See [Section 3.9: Layout Viewer](#).
- **case_metalonly.bat** launches GOF with a spare files list and layout information for performing a metal-only ECO. See [Section 4.6: Metal-Only ECOs](#).
- **case_multilibs.bat** launches GOF with a design that contains multiple leaf cell libraries. Leaf cells are color-coded based on which library defines them to make it easy to identify which parts of the design use cells from which library. See [Section 3.6: Display Settings and Comments](#) in the subsection *Change the color of cells defined by a particular library*.
- **case_timingvio.bat** launches GOF with a design that has a primetime PT file so that you can experiment with back annotating timing information into GOF. See [Section 2.6: Analyzing Timing Violations](#).
- **case_vcd.bat** launches GOF with a VCD file loaded so that you can do back annotation of waveform data back onto the schematic. See Chapter 6: Waveform Viewer Support form more information. See [Section 6.2: Load VCD and Launch Waveform Viewer](#).
- **case_btim.bat** is basically the same as case_vcd.bat, but it loads a waveform file stored in BTIM format. BTIM files load much faster than VCD files and require far less memory, so we recommend converting larger VCD files to BTIM format if you plan to view them often. To convert a VCD file after loading it into the waveform viewer, select the viewer menu option **File>Save As Timing Diagram**. See [Section 6.2: Load VCD and Launch Waveform Viewer](#).
- **logic_cone_eco.bat** launches GOF and performs a logic cone ECO operation. See [Section 4.5: Logic Cone ECO](#).

Below are some examples of different simple scripts and their behavior:

Simple GOF example:

```
gof -synlib tsmc.lib soc.v
# You have one netlist file 'soc.v' and one technology library, 'tsmc.lib'
```

Loading multiple netlists and multiple technology library files:

```
gof -synlib tsmc_std.lib -synlib tsmc_io.lib top.v part0.v part1.v
# You have three netlists, top.v, part0.v and part1.v, two technology library files
# standard library cell, tsmc_std.lib, IO cells, tsmc_io.lib
```

Loading a macro cell library (library of hierarchical modules):

```
gof -synlib tsmc_std.lib -synlib tsmc_io.lib -vn macros.v top.v part0.v part1.v
# Same as above except a macros.v, which has defined macro cells like sync-cells.
```

Using -v to load a simulation library for some analog cells (treated as leaf cells):

```
gof -synlib tsmc_std.lib -synlib tsmc_io.lib -vn macros.v -v analog_models.v top.v part0.v part1.v
```

Same as above except some analog cells having no technology library,
using '-v' option instead to load in analog simulation library.

Load a simulation library file:

```
gof -v /home/lib/tsmc_cells.v /home/netlist/soc.v  
# Use above to use a simulation library instead of a technology library
```

Loading library where leaf cells are defined in a directory:

```
gof -v /home/lib/tsmc_cells.v -v /home/lib/macro_cells.v +libext+.v -y /home/lib /home/netlist/top.v /  
home/netlist/soc.v  
# Multiple simulation library files, and some leaf cells are defined in '/home/lib' directory,  
# which uses '-y' option to resolve them.
```

Loading macro cell files defined in a directory:

```
gof -v /home/lib/tsmc_cells.v +libext+.v -y /home/lib -yn /home/vmodule /home/netlist/soc.v  
# '/home/vmodule' has some macro cells files.
```

Specifying a title for the design:

```
gof -v /home/lib/tsmc_cells.v /home/netlist/soc0.v /home/netlist/soc1.v -id the_soc_design  
# You would like to display 'the_soc_design' as the title.
```

Loading Layout files:

```
gof -synlib tsmc.lib -def soc.def.gz -lef libcell.lef soc.v  
# Design Exchange Format file soc.def.gz. And library exchange format file. For layout view  
usage.  
# See Section 3.9: Layout Viewer
```

Load a GTECH file to help match wires from the RTL code to wires in the synthesized netlist:

```
gof -v /home/lib/tsmc_cells.v /home/netlist/soc.v -gtech soc_submod.gtech.gv  
# Load in gtech file for RTL wire to netlist mapping  
# See Section 2.4: Match Net from RTL to Synthesis
```

Use a script to process the netlist:

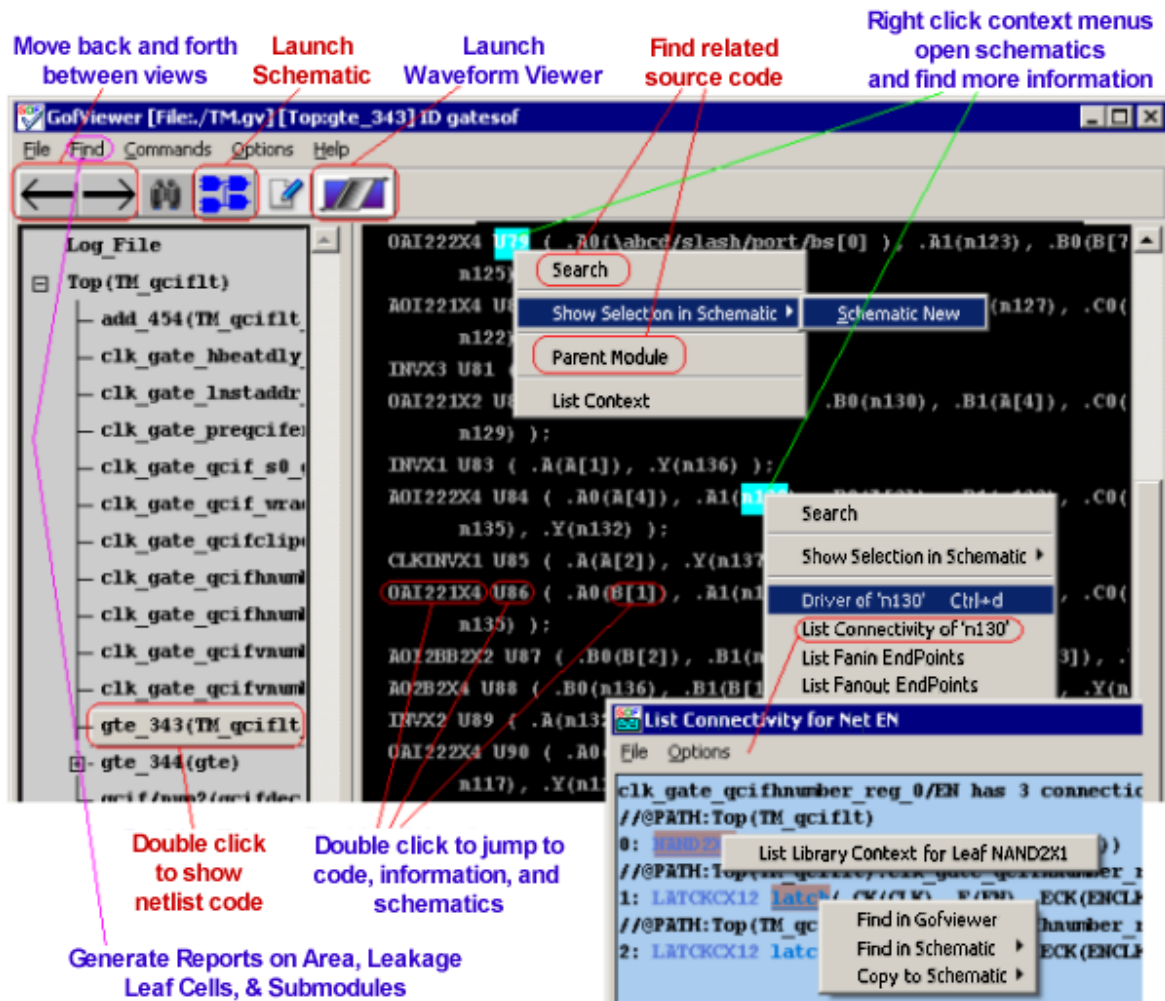
```
gof -synlib tsmc.lib soc.v -run scripts.pl  
# Process netlist with scripts.pl in text mode  
# See Section 5.1: Interactive window & batch files
```

Load a Waveform file to annotate the schematic:

```
gof -v simulation_lib.v top.v netlist.vg -vcd top.vcd  
# Load in a VCD file for schematic annotation  
# See Chapter 6: Waveform Viewer Support
```

Chapter 2: GofViewer - text viewer

GofViewer displays a netlist as a hierarchical tree of module instances with an associated text window. This is the first window opened after launching GOF from the command line (see [Chapter 1: Launching GOF with a Batch script](#)). From this window, the design can be quickly explored by double clicking on the tree and viewing the code. Use the right click context menus and/or double click on objects to quickly find related information such as the drivers and loads of a particular net. GOF is optimized to handle very large netlists, so it only loads the module that you are looking at.

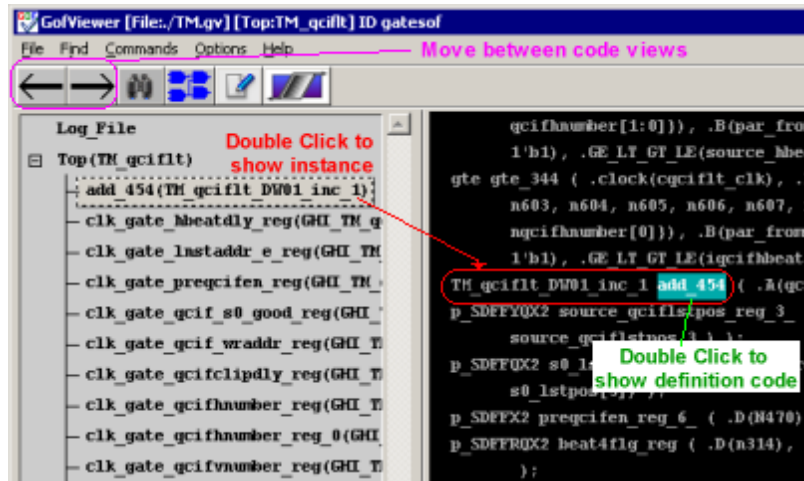


2.1 Moving within GofViewer

Designs are loaded into GofViewer using a batch file as described in the section [Launching GOF with a Batch script](#). The left pane of GofViewer shows a hierarchical view of the loaded design (the module definitions located in the loaded netlist files). The right pane shows the actual netlist code for the design. Nets and modules are hyperlinked so that double clicking will either: open up the definition of the object or display a leaf cell in the schematic window. The right click menu provides functions for finding related loads, drivers, fanin, fanout, parent modules, and the list context.

Double click on left side to display code on the right:

- **Double click** on a node in the hierarchy tree to open netlist code on the right side.
- Press the plus and minus icons to expand the hierarchy tree nodes.



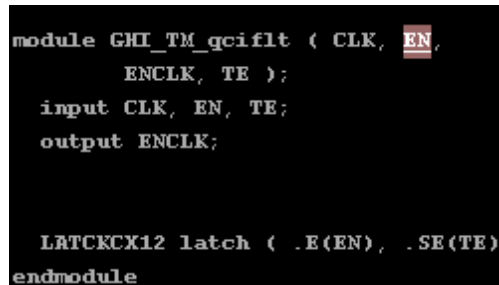
Use Arrow Buttons to jump to the previous and next views:

- The Arrow buttons on the button bar move between the last code that you have viewed and back again.



Left click on code to find hyperlinks then double click or use the context menu to find stuff:

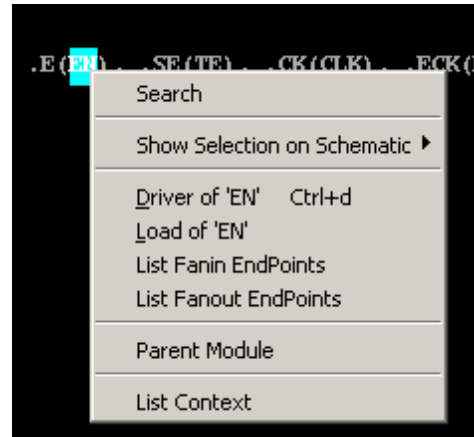
- If you **left click** on text in the RTL window and it immediately highlights itself, then the text is hyperlinked to other parts of the design.
- Try **double clicking** and **right-clicking** on the hyperlinks to get to related code and schematics.



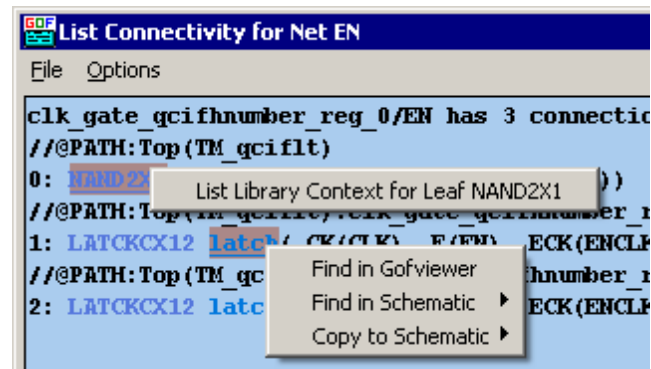
Right-click to search, show on the schematic, and find related gates:

- **Right click** on the highlighted text to open a context menu.

- **Search** will open the Search dialog with the highlighted text entered ([Section 2.2 Search the Netlist](#)).
- **Show Selection on Schematic** will show a leaf cell or net in a schematic window ([Chapter 3: GofTrace](#)).
- **Driver of** will take you to the gate that drives the selected net.
- **Load of** will open a window that displays the fanout of the selected net.
- **List Fanin Endpoints** finds the flipflops and ports that drive the net.

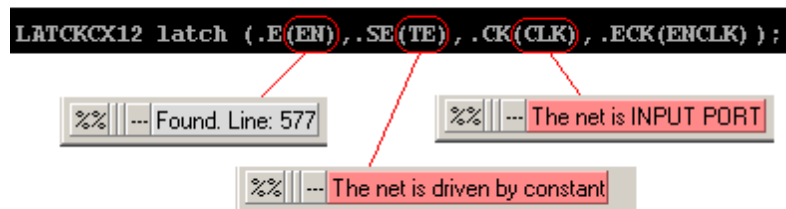


- **List Fanout Endpoints** menu finds the flipflops and ports that the net drives.
- **Parent Module** takes you to the surrounding module that the current text is located in.
- **List Context** shows what the net is connected to and the hierarchical name of the scope where the net was defined.
- Some of the above menus will open other windows to show the relevant information. The text inside these windows is also hyperlinked, so you can click, double click and right click to navigate further into the design.

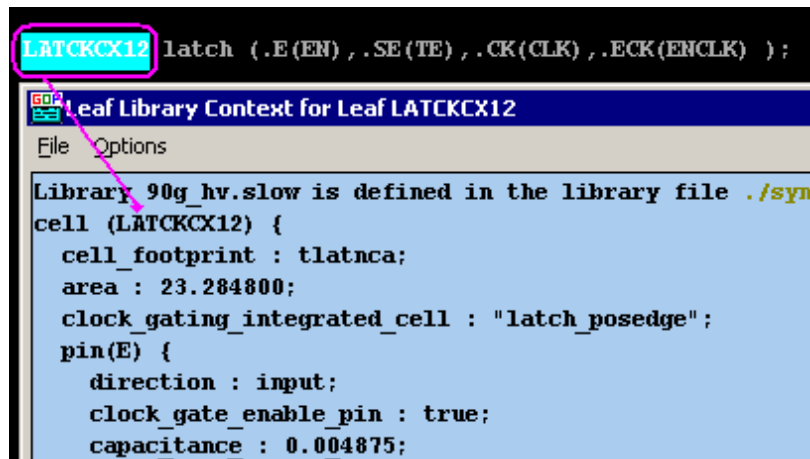
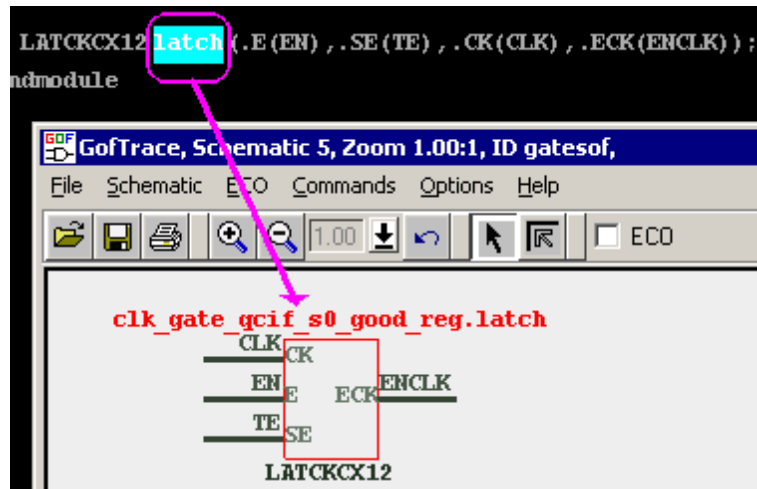


Double Left Click to jump to code, information, and schematics:

- **Double-left click** on a **net** to jump to the code that shows the net's driver. The bottom left corner will show red if there is no driver, or else the driver code will be displayed.



- **Double-left click** on an **instance of a leaf-cell module** to display the part in a schematic window (see [Chapter 3: GofTrace](#)). A leaf-cell is a module that is defined via a synthesis or technology library.
- **Double-left click** on an **instance of a hierarchical module** to display the definition of the module in the netlist text window.
- **Double-left click** on a **leaf cell type** to find information from the library that defines the cell type. The nature of the information displayed depends on whether the cell is from a technology library or a simulation library.



2.2 Search the Netlist

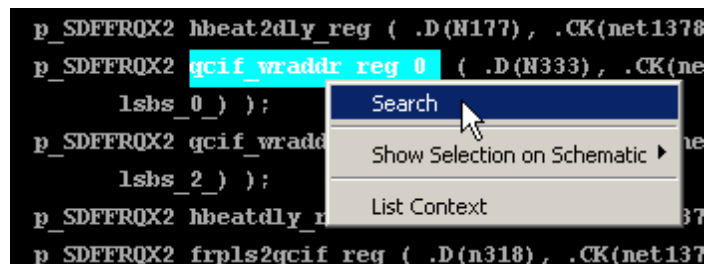
You can search inside the netlist text window of GofTrace using either the <CTRL>-S key combo or the search dialog.

Quick Search with <CTRL>-F keys:

- **Left click** on some text in the netlist window to select it, then press the **CTRL-F** keys to search for other instances of that text. Press <CTRL>-F again to jump to the next instance.

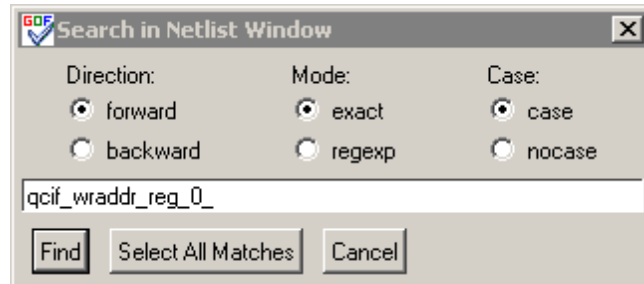
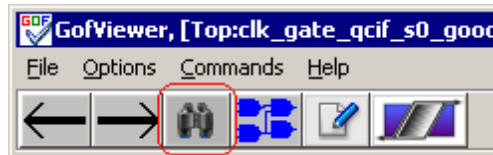
Search with a Dialog:

- Open the search dialog by either:
 - **Right clicking** anywhere in the netlist window and choosing **Search** from the context menu to open the search dialog.
 - Or press the **Search**



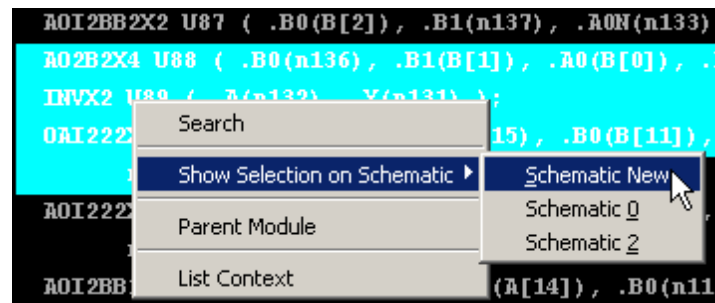
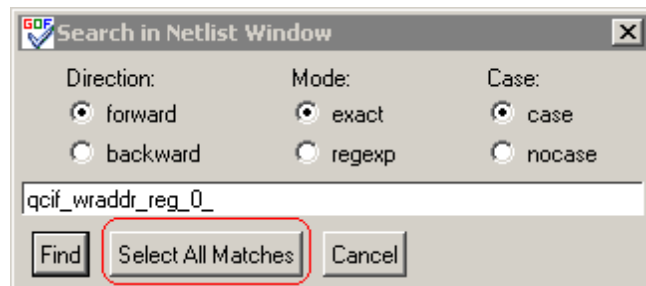
button on the button bar.

- Using either method, any selected text will be inserted automatically into the *Find* field in the dialog.
- Press the **Find** button to start a search for the next occurrence of the string in the netlist.



Highlight all Instances of a module:

- The **Select all matches** button highlights all strings that match the search string so you can easily see them when scrolling through the netlist text window.
- This function also selects the instances into a buffer so that if you right click and choose the **Show Selection on Schematic** menu, all of the selected gates will be displayed in a schematic window.

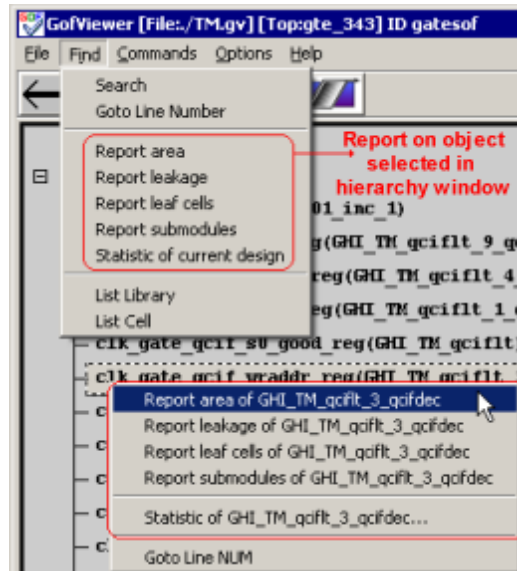


2.3 Report area, leakage, leafs, & submodules

GofViewer can generate several different reports about the entire design or a branch of the hierarchy tree. To generate reports using the Report Area or Report Leakage menus, GOF needs to have access to the technology libraries for the leaf cells of the netlist.

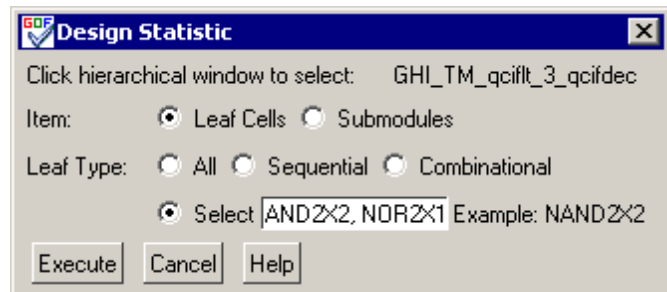
Generate Reports for different modules and parts of the design:

- To generate a report about a particular module, left click on the module name in the hierarchy window to select it, then right click and choose one of the **Report** or **Statistic** menus.
- These menus are also available as context menus in the hierarchy window.

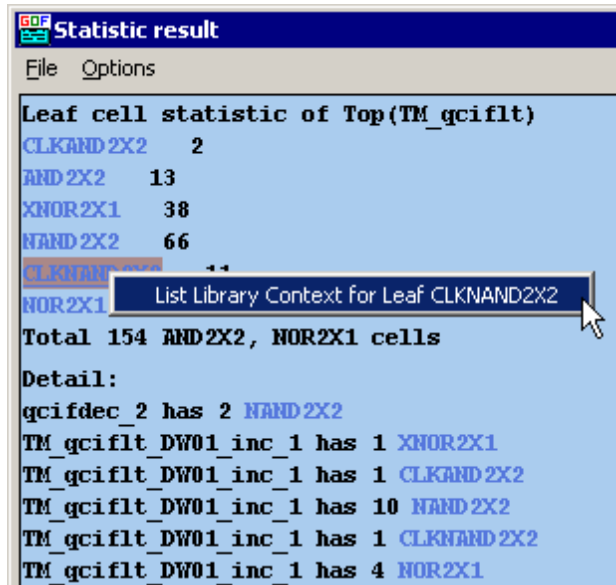


Report LeafCells and Statistic of Design:

- The **Report Leaf Cells** menu generates a report that lists all the leaf cells of a particular module. This report is hyperlinked, so right clicking on the blue entries will give you more information about that item.
- The **Statistic** menu opens a dialog which gives you more control over the reports about **Leaf Cells**.
- If the Leaf Cells option is selected, you can choose to report **all** leaf cells, just **sequential** cells, just **combinational** cells, or just a set of leaf cells that partially match a comma-separated list of strings. GOF requires a technology library to distinguish between sequential and combinational leaf cells.

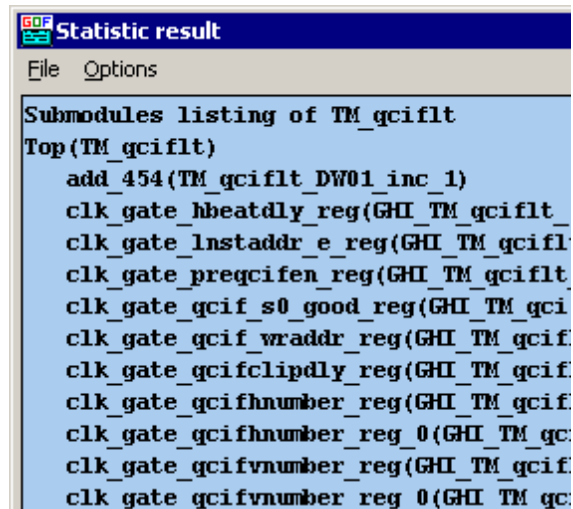


- Press the **Execute** button to generate a report and display it in a *Statistic result* window. The dialog will remain open so that you can quickly generate additional reports by selecting other instances in the design hierarchy and pressing execute again.
- You can **right-click** on leaf cell names in a *Statistic result* window to see their library definitions.



Report on Sub-modules:

- The **Report Submodules** menu generates a list of all of the submodules (non-leaf cells) of a selected instance in the hierarchy tree window.



Reports on Area and Leakage:

One or more technology libraries (synthesis libraries) that define the leaf cells of the design should be loaded before generating area and leakage reports. If a library is missing, a warning will appear and the results will not be accurate. See GofViewer Commands Menu for information on loading the technology library file(s).

- The **Report Area** menu shows the total area of the selected object and then the area on each subcomponent.

MODULE	NUMBER	UNIT-AREA	AREA	PERCENT
GHI_TM_qciflt	1	23.28480	23.28	0.38 %
GHI_TM_qciflt_1_qcifdec	1	23.28480	23.28	0.38 %
GHI_TM_qciflt_2_qcifdec	1	23.28480	23.28	0.38 %
GHI_TM_qciflt_3_qcifdec	1	23.28480	23.28	0.38 %
GHI_TM_qciflt_4_qcifdec	1	23.28480	23.28	0.38 %

- The **Report Leakage** shows the leakage of the selected object and then the leakage on each sub component.

MODULE	NUMBER	UNIT-LEAK	LEAK	PERCENT
GHI_TM_qciflt	1	6.51229	6.51	0.53 %
GHI_TM_qciflt_1_qcifdec	1	6.51229	6.51	0.53 %
GHI_TM_qciflt_2_qcifdec	1	6.51229	6.51	0.53 %
GHI_TM_qciflt_3_qcifdec	1	6.51229	6.51	0.53 %
GHI_TM_qciflt_4_qcifdec	1	6.51229	6.51	0.53 %

2.4 Matching a Net from RTL to Synthesis

One challenge when making a change to a netlist is to find the right spot in the modified netlist code that matches the original RTL code. Constructs such as inputs, outputs and flip-flops are usually preserved during synthesis so they are easy to spot when making a change. However, wires are usually renamed or completely optimized away by the synthesis tool. To manually find these wires is difficult and timing consuming (or even impossible when the net has been completely optimized away).

GofViewer has an RTL-to-Netlist Matching feature which eases the process of locating synthesized nets. To use this feature, you need to load in a GTECH file generated from the specific RTL-level sub-module containing the RTL version of the net. A GTECH file is a generic unoptimized netlist that can be generated using either Synopsys Design Compiler or Cadence synthesis tools from an RTL file.

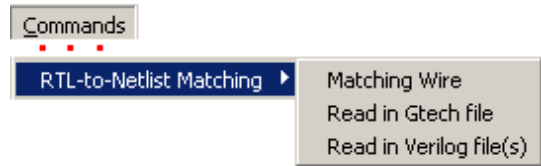
Creating a GTECH file using Synopsys DC Compiler:

- From inside a DC shell, the following DC commands will create a Verilog version of a GTECH file for a given RTL file (you will need a DC license to perform these operations):

```
read -f verilog submodule.v
```

```
write -f verilog -hierarchy submodule -o submodule.gtech.gv
```

- Optionally, the **Commands > RTL-to-Netlist Matching > Read in Verilog files** menu can be used to automatically launch DC and silently perform these commands. DC must be installed on the computer to execute this function.



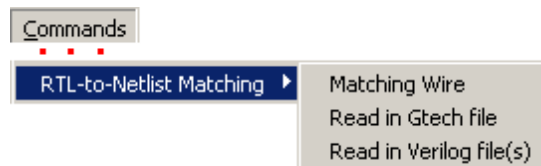
Load the GTECH file into GOF:

A GTECH file from either Synopsys or Cadence can be loaded at GOF startup time using a **-gtech** option or loaded later using the **Read in Gtech file** menu. Shown below are examples of both methods of loading a GTECH file. Note that GOF also requires access to the design's technology library(s) to setup a register pattern for the RTL to Synthesis matching function.

- Loading a GTECH file at GOF startup time:

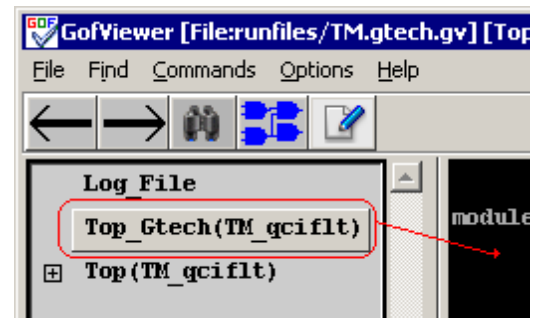
```
gof -v library.v top_netlist.gv -gtech submodule.gtech.gv -synlib synthesis.lib
```

- The **Commands > RTL-to-Netlist Matching > Read in Gtech file** can be used to load in a GTECH file after GOF launch.

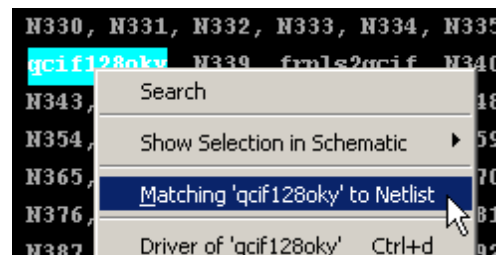


Matching a Wire from the RTL to the Netlist:

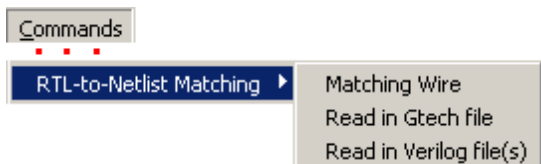
- When a GTECH file is loaded, GofViewer will display two trees hierarchies in the left side window.
- Double click the GTECH tree to load the GTECH file into GofViewer.



- Find the GTECH net that needs to be matched to the design's netlist, then right click on it and choose the **Matching net_name to Netlist** from the context menu to open a window with the results.

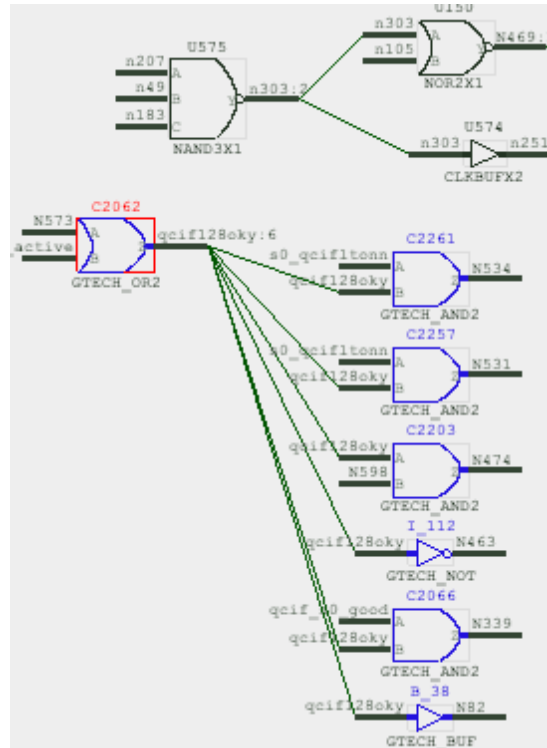


- There is also a Commands menu, **Matching Wire**, that opens a dialog that lets you type in the net name that will be matched.



- The results are classified into different levels: **Best**, **Good** (same fan-in but incomplete fan-out), **Good2** (different fan-in but exact fan-out). If a netlist net exactly matched the RTL net, it

- GTECH gates are displayed in a different color in the schematic to prevent confusion with the real (optimized) netlist gates.
- Right click on a pin and choose **List Fanin Endpoints** to open a window that shows all the fanin gates for that pin. **List Fanout Endpoints** is also helpful in comparing different parts.



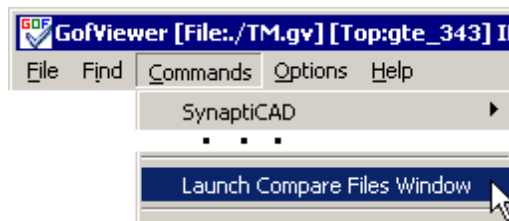
Example of Matching Net from RTL to Synthesis Features:

In the **SynaptiCAD > Examples > GOF_features** directory, the **case_gtech.bat** file launches GOF with a design that has a gtech file and the RTL Code so that you can experiment with the features in this section.

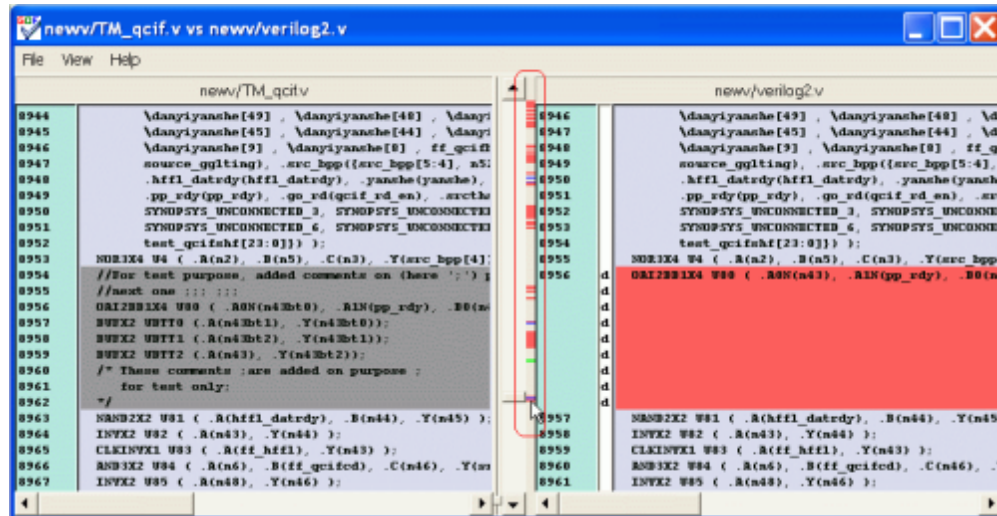
2.5 Diff Utility for Netlist files

GofViewer can launch a file comparison utility that shows the differences between two netlist files or other text files. To use this function, you need to have a "diff" program installed. Diff is a standard program installed by default on most Unix system. On Windows, you can install the diff program from www.cygwin.com to enable this functionality.

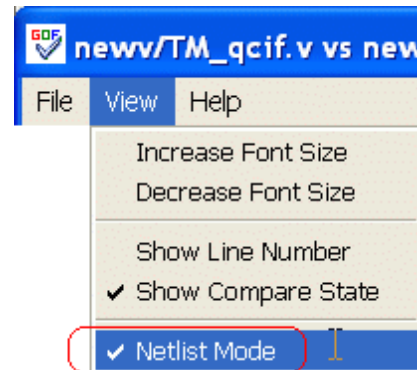
- Choose **Commands > Launch Compare Files Window** menu to open the *Diff two text files* dialog.



- Choose the files to perform the diff on and press the **RunDiff** button to open the Diff window and display the results in either netlist mode or line-by-line difference mode (set by checkbox).



- The differences are marked in red. Click on the colored bar in the middle to move between differences.
- Once a comparison has been performed, you can switch between netlist mode and line-by-line mode by checking the **View > Netlist mode** menu.



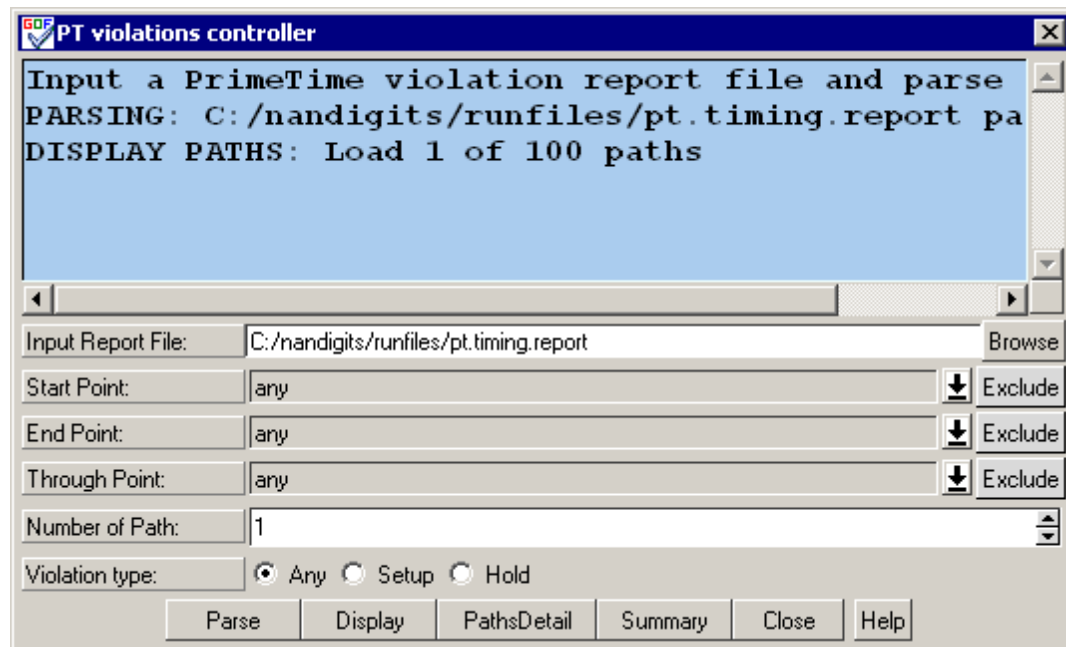
2.6 Analyzing Timing Violations

GOF can help investigate timing violations. Once you get a Prime Time report file, you can use GOF to look at the timing violations using the schematic window and several different listing windows.

View Timing Violations:

- In GOF, choose the **Commands > Process Timing Violations** menu to open the *PT violations controller* dialog.





- Enter the name of a timing report file in in the **Input Report File** edit box.
- The **Start Point** and **End Point** boxes let you specify a place in the hierarchy to start or stop the search for timing violations
- The **Through Point** box can specify a point in the hierarchy that timing violations should be associated with.
- The **Number of Paths** indicates the number of paths to display in a list window or draw on a schematic.
- Press the **Parse** button to find the timing violations that meet the settings in the dialog.
- Once the Parse operation has completed, view the information using the **Display**, **Paths Detail**, and **Summary** buttons.
- The **Display** button shows the timing violation on a schematic.



- The **Paths Detail** button opens a listing window with timing information about all the nets and gates in each path.

List detail of paths

File Options

Startpoint: uinst_hshf/curindex_reg_3_ (rising edge-triggered flip-flop clocked by clk2gb_gcifr_clk)
 Endpoint: uinst_hshf/yanshe_reg_27_ (rising edge-triggered flip-flop clocked by clk2gb_dstw_clk)
 Path Group: clk2gb_dstw_clk
 Path Type: max

Des/Clust/Port	Wire Load Model	Library
TM_gcif	MH59JDC_36000	artisan_nec90g_hv.synth

Point	Fanout	Cap	Trans	Incr	Path
clock clk2gb_gcifr_clk (rise edge)				0.00	0.00
clock network delay (ideal)				0.00	0.00
uinst_hshf/curindex_reg_3 /CK (p_SDFFRHQX4)			0.00	0.00	0.00 r
uinst_hshf/curindex_reg_3 /0 (p_SDFFRHQX4)			0.34	0.42	0.42 r
uinst_hshf/test_shf[3] (net)	11	0.07		0.00	0.42 r
uinst_hshf/U4957/A (NAND2X4)			0.34	0.00	0.42 r
uinst_hshf/U4957/Y (NAND2X4)			0.12	0.17	0.59 f
uinst_hshf/n6581 (net)	2	0.01		0.00	0.59 f
uinst_hshf/U5077/B (NOR2X4)			0.12	0.00	0.59 f
uinst_hshf/U5077/Y (NOR2X4)			0.15	0.16	0.76 r

- The **Summary** button opens a listing window with summary information about each path (e.g. total delay time of the path).

Violations summary

File Options

List paths:

0	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_27	VI0: -1.25 max
1	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_25	VI0: -1.25 max
2	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_27	VI0: -1.24 max
3	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_25	VI0: -1.24 max
4	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_27	VI0: -1.23 max
5	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_25	VI0: -1.23 max
6	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_21	VI0: -1.22 max
7	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_19	VI0: -1.22 max
8	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_7	VI0: -1.22 max
9	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_6	VI0: -1.22 max
10	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_4	VI0: -1.22 max
11	S:uinst_hshf/curindex_reg_3	E:uinst_hshf/yanshe_reg_3	VI0: -1.22 max

Example of Analyzing Timing Violations Features:

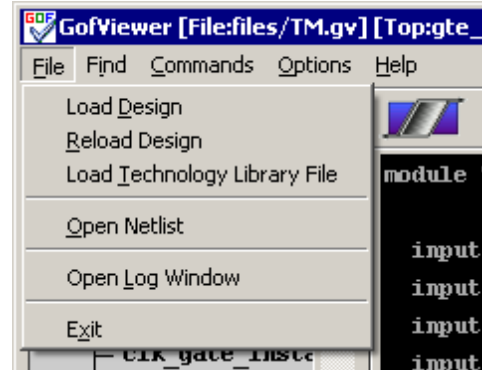
In the **SynaptiCAD > Examples > GOF_features** directory, the **case_timingvio.bat** file launches GOF with a design that has a primetime PT file so that you can experiment with the features in this section.

2.7 GofViewer Menus and Options

The following is an overview of the menu commands in the GofViewer window.

File Menu:

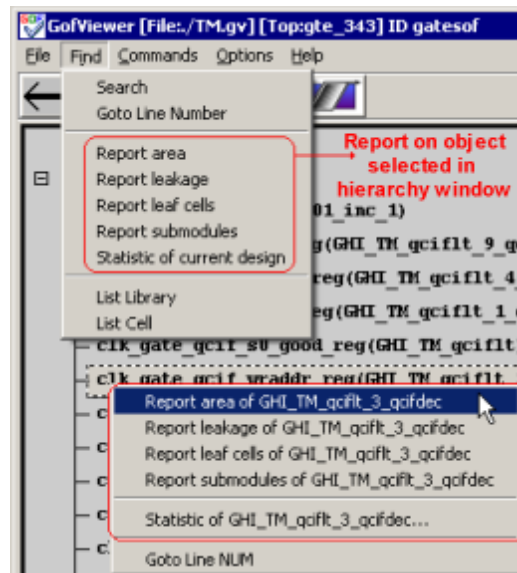
- **Load Design** loads libraries, netlists, and other options.
- **Reload design** gets the modified design files after making changes with the GofECO feature and refreshes the screen.
- **Load Technology Library File** loads into memory a technology library which contains parameters for leaf gates. This performs the same function as the -synlib option that is used in the batch file that launches GOF (see [Chapter 1: Launching GOF with a Batch script](#)).



- **Open netlist** can open a single netlist file, but not library files. GOF usually should be launched using a batch file as described in [Chapter 1: Launching GOF with a batch script](#).
- **Open Log Window** launches a list window that shows all of the GOF commands that have been performed.
- **Exit** closes GOF and all GOF sub-windows (such as the schematic windows and list report windows).

Find Menu:

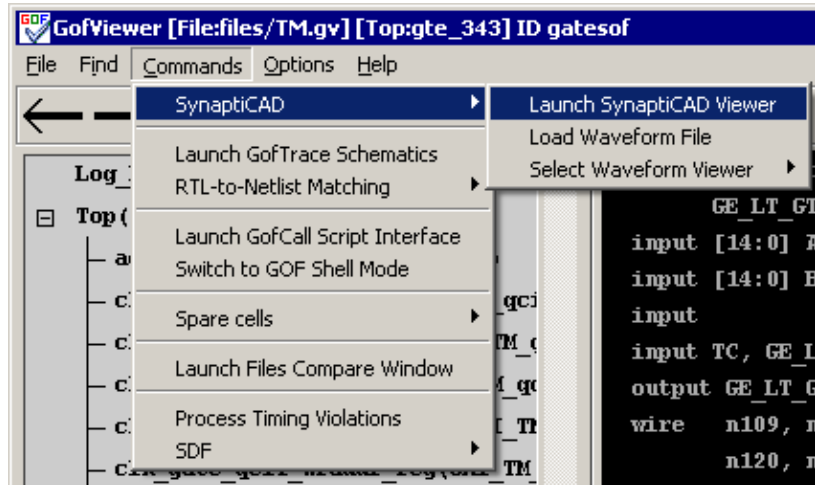
- **Search** opens a dialog that lets you find strings that match a pattern (see [Section 2.2 Search the Netlist](#)).
- **Goto Line Number** opens a dialog which accepts the line number to scroll to.
- **Report** and **Statistic** menus open list windows with the associated information (see [Section 2.2 Report area, leakage, leafs & submodules](#)). These are also available in the right click context menu.
- **List Library** opens a list window that shows all of the cell types defined in the loaded technology libraries. Technology libraries are loaded using the -synlib command (see [Section: Launching GOF with a Batch script](#)) or with the **File > Load Technology Library File** menu.



- **List Cell** opens a dialog which lets you type in the name of a cell and then opens the library context for that cell. If you do not know the cell name, choose the **List Library** menu and find the cell in the resulting list, then right click to get the list context for the cell.

Commands Menu:

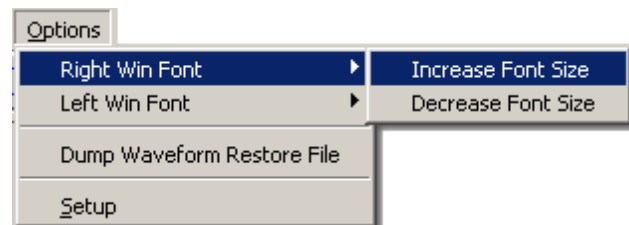
- **SynaptiCAD** menus launch the waveform viewer, load the waveform file and also set which viewer will be used (see [Section 6.2: Show logic states from VCD file](#)).
- **Launch GofTrace Schematics** opens a blank schematic window (see [Chapter 3: GofTrace](#)).



- **RTL-to-Netlist Matching** maps a net in a Gtech file to a netlist net (see [Section 2.4: Match Nets from RTL to Synthesis](#)).
- **Launch GofCall Script Interface** opens a window that runs Netlist Processing APIs in interactive mode (see [Section 5.1: Interactive window & batch files](#)).
- **Switch to GOF Shell Mode** closes the graphical GOF interfaces (GofViewer and GofTrace windows) and enters into the GOF's shell script mode where you can directly enter GofCall API commands. You can switch back to GUI mode from shell script mode with the shell command `run start_gui` (see [Section 5.2: Command line text mode](#)).
- **Spare cells** menus are used to create a spare cells file and load it. This is used in metal-only ECOs (see [Section 4.6: Metal Only ECOs](#)).
- **Launch Compare Files Window** opens a window that will compare two netlists and show the differences between them (see [Section 2.5: Diff Utility for RTL files](#)).
- **Process Timing violations** displays timing violations from a Prime Time file using the schematic window and several listing windows (see [Section 2.6: Process Timing Violations](#)).
- **SDF** creates an index for an SDF file and loads the SDF index file. GOF uses SDF index files to speed access to SDF timing data (see [Section 3.8: SDF timing](#)).

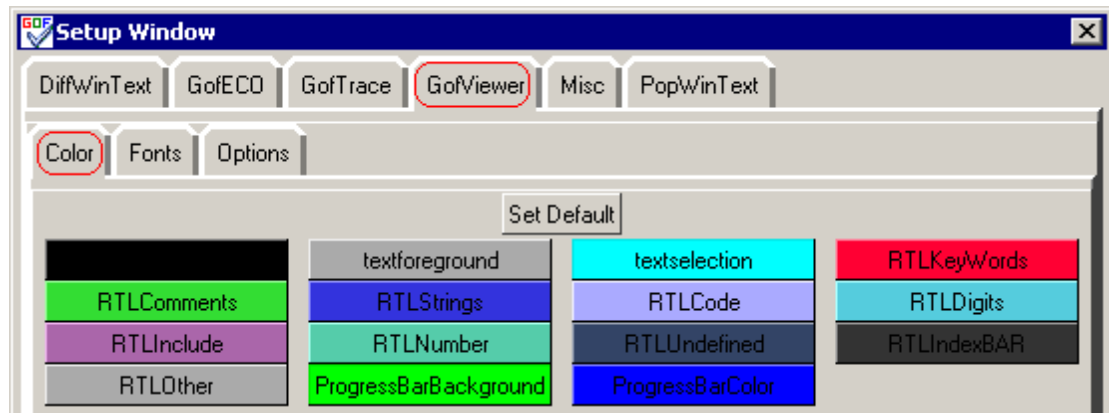
Options Menu:

- **Right Win Font** and **Left Win Font** control the font size for the left and right windows of the GofViewer. The actual font types are controlled using the GOF Setup dialog (see below).

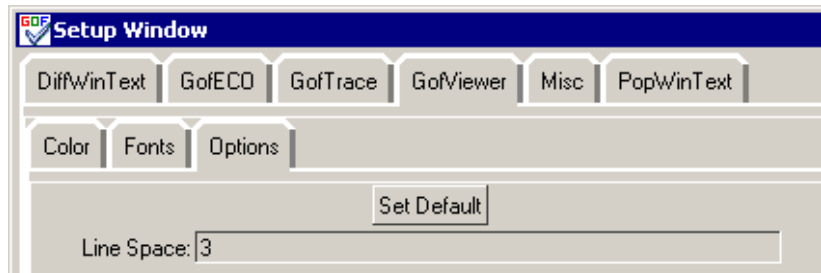


- **Dump Waveform Restore File** will create a filter file that can be read by Verdi and other programs (see [Section 6.3: Create Waveform Restore File](#)).

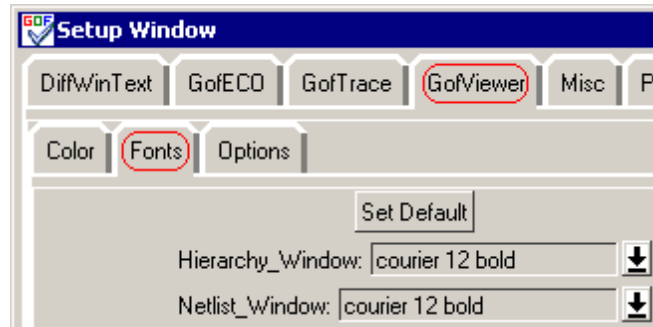
- **Setup** opens a dialog that lets you set the colors and fonts for each of the windows in the GOF tool suite.



- The **Options Tab** controls line spacing between lines in the source code window.

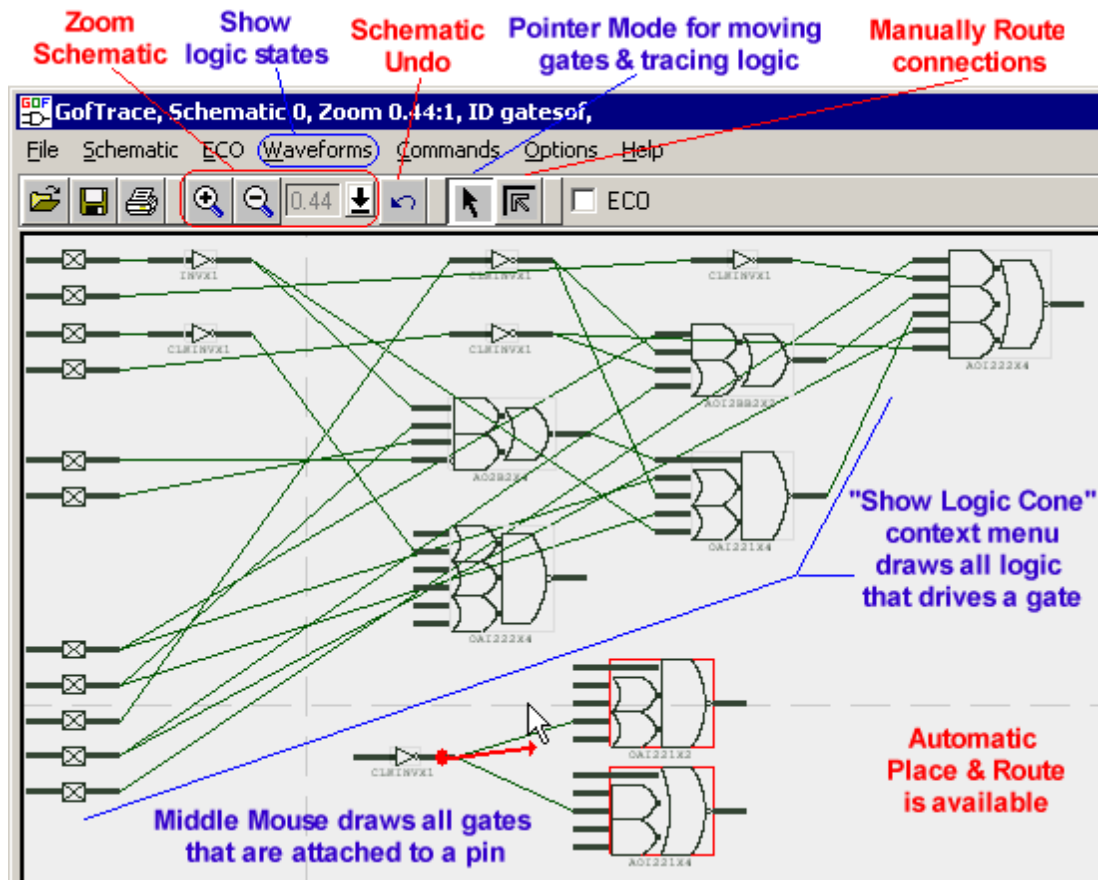


- The **Fonts Tab** sets the font for each type of GOF window.



Chapter 3: GofTrace - schematic viewer

Once a netlist is loaded, you can view sections of the design as a schematic using a GofTrace schematic window. GofTrace allows you to quickly display only the portion of logic that you are interested in by focusing on tracing just the fan in/out of a particular path, unlike other schematic tools which force a fixed placement of the gates that makes it difficult to rearrange the gates to visualize just the area of interest.



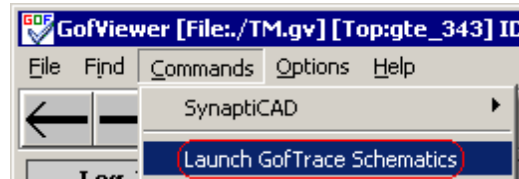
3.1 Open a Schematic Window

There are several different ways that you can launch a GofTrace window and fill it with a partial schematic. Once you get a single gate into a schematic (either by double left clicking, dragging-and-dropping, or using the right click menu), you can use the commands in the next section ([Section 3.2: Draw Fan In/Out of a Part](#)) to trace the buffers and logic attached to the selected part.

It is important to remember that schematics only display leaf cells that are defined in the *Technology* or *Simulation libraries* that you loaded into GOF. Module definitions in the *netlist* are treated as hierarchical models and can be viewed in GofViewer, but will not be displayed in the schematic as a distinct part. Only the leaf cells and their connections will be displayed in the schematic, because that is what can be changed during an ECO. See [Section 1.2: Concepts for GOF files](#) for more information on this topic.

Launch GofTrace with Command Menu or Double Left Click:

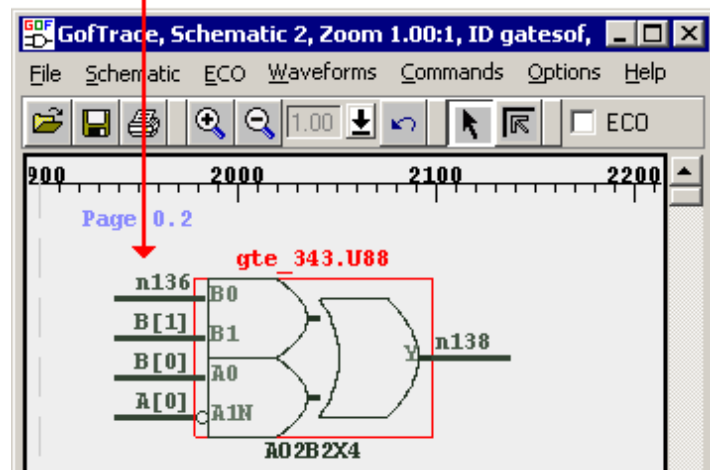
- To open a blank schematic window either:
 - Press the **GofTrace** button on the GofViewer window.
 - Or choose the **Commands > Launch GofTrace Schematics** menu.



- To open a schematic window with a gate symbol loaded, **double left click** on a module instance in the netlist text window. If the double click takes you to a new section of code instead of opening a schematic, then the module is a hierarchical model and you need to drill down to find the leaf cells to display on the schematic.
- Drag-and-Drop one or more selected leaf cell from the viewer window to an open schematic window using either the **left** or **middle mouse button**.

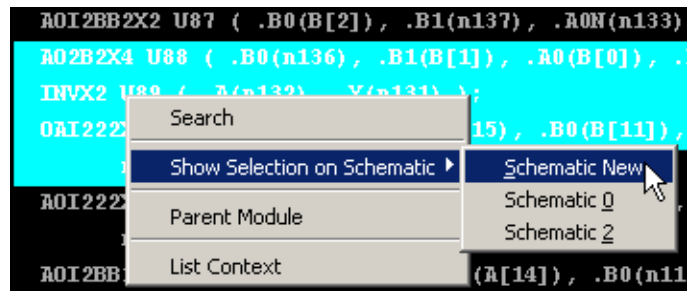
A02B2X4 **U88** (.B0(n136), .B1(B[1]), .A0(B[0]), .A1N(A

Double Click to display in a new schematic
or
Drag-and-Drop to an open schematic

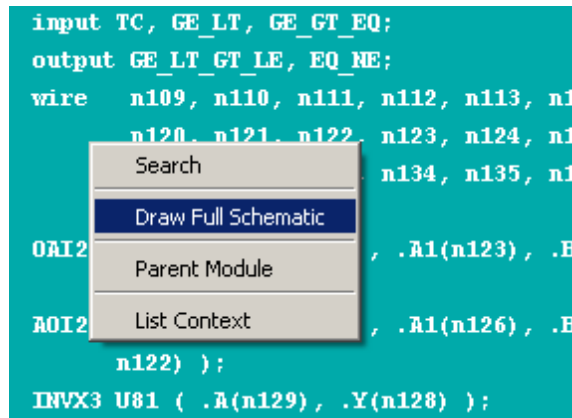


Send Several or all Gates to a schematic using the right click menu:

- Select one or more lines of code, right click and choose one of the **Show Selection on Schematic > Schematic #** to load the selected gate or gates into a schematic. Selecting **Schematic New** will open a new schematic window (the other entries in this popup menu are for existing schematic windows).

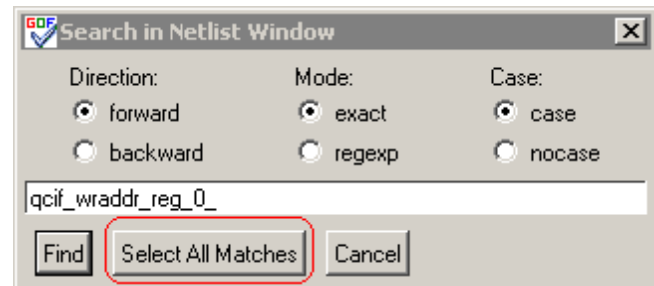


- For small modules, **<CTRL>-A** will select all of the text. This will enable a new context menu option called **Draw Full Schematic** that sends all of the selected gates and the nets connecting them to the schematic and places and routes the schematic.

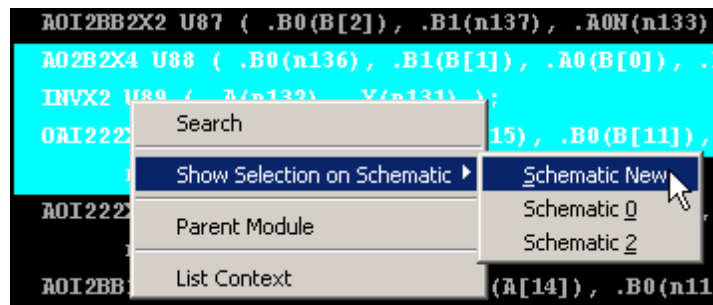


Sending all Instances of a module to a schematic window:

- In the GofViewer window, right click and open the Search dialog.
- Type in a module definition name and press the **Select all matches** button to highlight all strings that match the search string so you can easily see them when scrolling through the file.

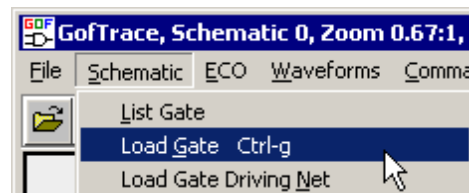


- All the selected instances can then be displayed in a schematic window using the **Show Selection on Schematic** menu.

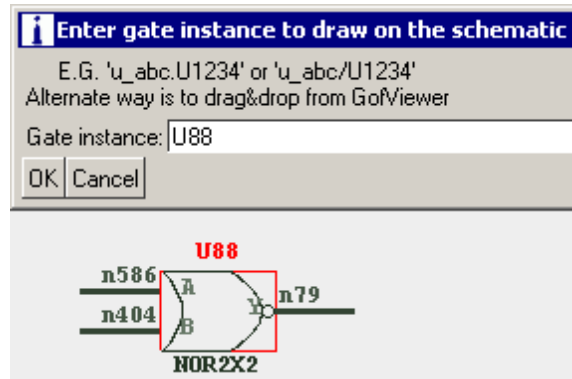


Use GofTrace Load Gate menu to load a gate or net into the schematic:

- From inside a GofTrace schematic window, choose either the **Schematic > Load Gate** or the **Schematic > Load Gate Driving Net** menu to open a dialog for specifying the object to display on the schematic.

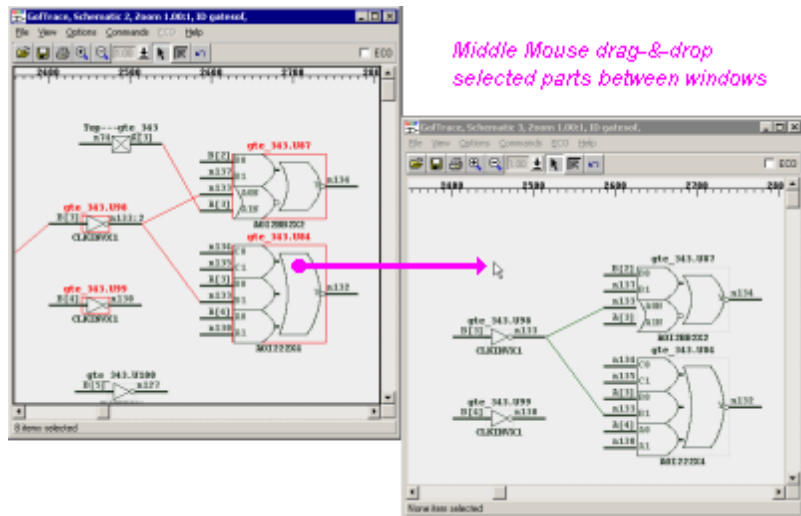


- Enter the name of the gate or the net depending on which dialog you open. Wild cards * are allowed in the name. The path can be specified using '.' or '/' characters. The search is CASE SENSITIVE.
- Press the **OK** button to display the gate/net on the schematic.

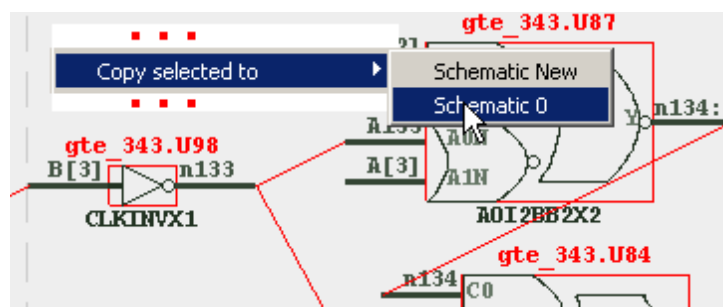


Transfer Gates between Schematics by drag-&-drop or context menu

- For either method of transferring gates, you must first specify the gates to transfer by selecting them. Multiple gate selection can be achieved by **left clicking down and drag** on an empty part of the window to open a selection box, and then dragging until the selection box surrounds the parts to be selected. More parts can be added to the selection by pressing the **<CTRL>** key while **left clicking** on objects. **<CTRL>-A** will select all the gates in the active schematic window.
- Press the **middle mouse button** down on one of the selected gates, then drag the mouse to the new schematic window and release the mouse button. During the drag you will see the symbol "Drag & Drop Circuit" moving along with the mouse cursor.



- Another way to transfer gates is to **right click** and choose **Copy Selected to > Schematic #** from the context menu. This menu will only be visible if you have gates selected.

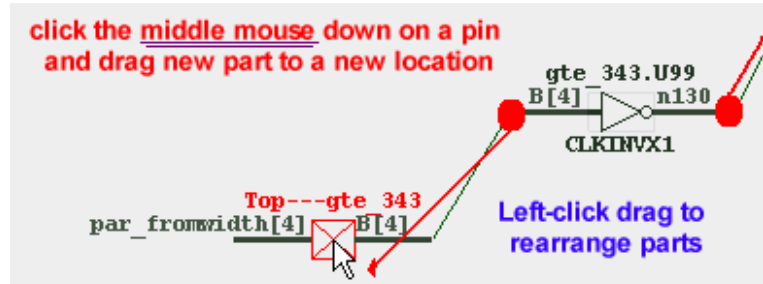


3.2 Show Fan In/Out of a Gate

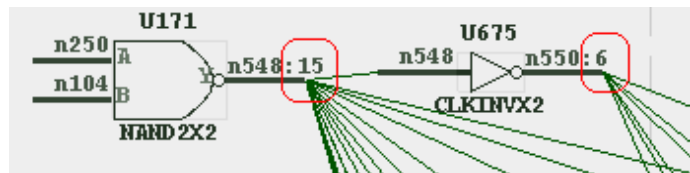
GofTrace is optimized to allow you to quickly trace a gate's fan-in and fan-out so that you can determine how the gate is connected in the design. Once a gate is loaded into the schematic as described in [Section 3.1: Open a Schematic Window](#), use the middle mouse button on the pins of the part to display the parts connected to the pin. If a technology file was loaded with the netlist, there are also several **Show** context menu functions that can display special types of trace paths when a pin or gate is selected (e.g. show all pure buffer paths driven by a pin).

IMPORTANT: Tracing more gates with the middle mouse button:

- Press the **middle mouse** down on a **pin** to display the fan in/out, then drag the mouse to a new location and release to complete the placement. If you don't have a middle mouse button, press both the **Left and the Right mouse buttons at the same time**.



- Use the **left mouse** button to rearrange existing gates.
- When you trace the fanout on a pin, the number of gates driven is displayed as a number after the pin name.



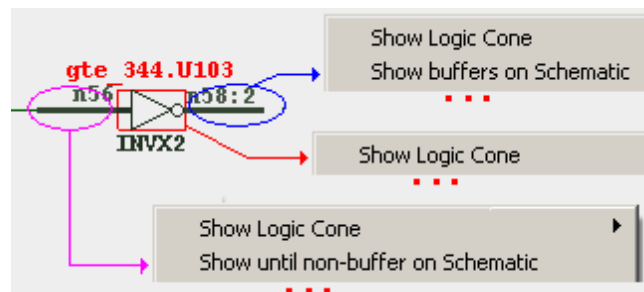
Automatically trace special gate paths using the right click Show menus

- To use the **Show** features, a technology library must be loaded (instead of a simulation library). The technology library file can be loaded at GOF startup with the **-synlib** command option or later using the **File > Load Technology Library File** menu function.

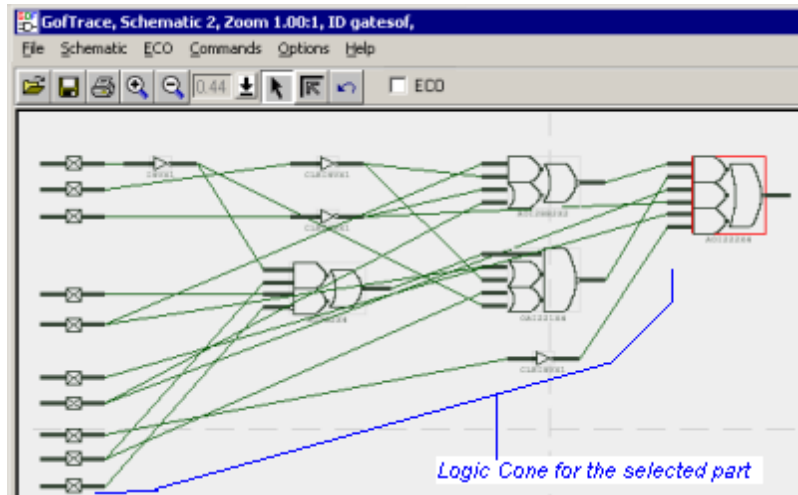
```
gof -synlib tsmc65.lib dsp_chip.v
```



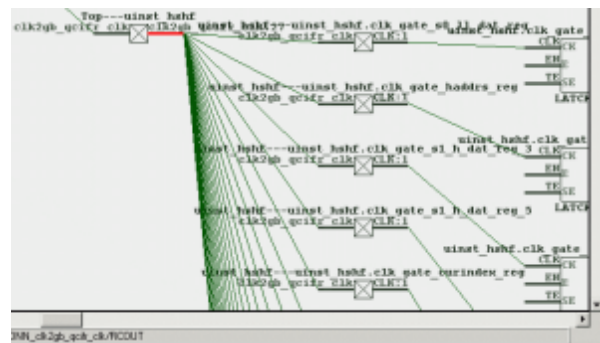
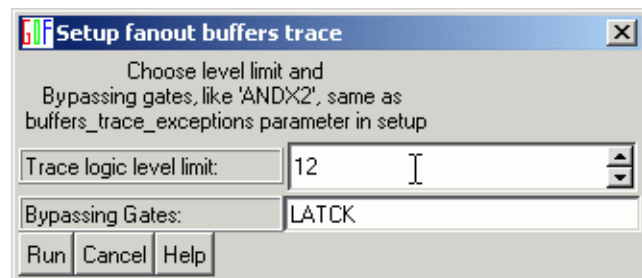
- Left click on either an **input pin**, an **output pin**, or the **gate itself** to select it.
- Then right click to open the context menu for that particular point on the part and select the **Show** menu for the desired type of trace path to display in the schematic.



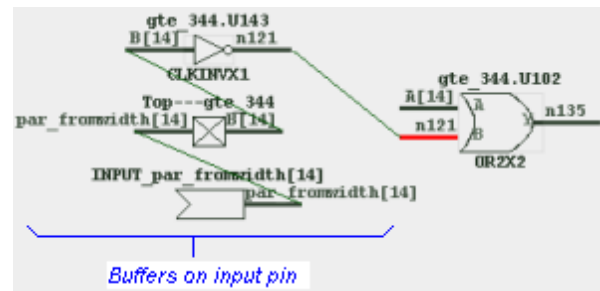
- **Show Logic Cone** displays all of the gates that drive all of the input pins of the selected gate. This function will trace through all combinational logic back to a port or flip-flop. This menu is available on both pins and gates.
- If the resulting logic cone is very large, then a warning dialog may pop-up and recommend that you view it as text.



- **Show Buffers on Schematic** is available only on **output pins**. It draws all of the buffers and inverters that are driven by the pin until it reaches a non-buffer-type gate. This menu opens the *Setup Fannout Buffers Trace* dialog.
- **Trace logic level limits** accepts a number that sets the maximum number of logic levels to investigate.
- **Bypassing Gates** allows you to exclude certain gates from the function. This is useful when tracing buffers on a clock pin, where the pin may drive some gated clock cells along with clock buffers. By entering the gated clock cell name, the Show Buffers feature will not stop on these gates.



- **Show until non-buffer on Schematic** is only available on **input pin menus**. It draws all the buffers and inverters that drive the pin up to and including the first logic gate.

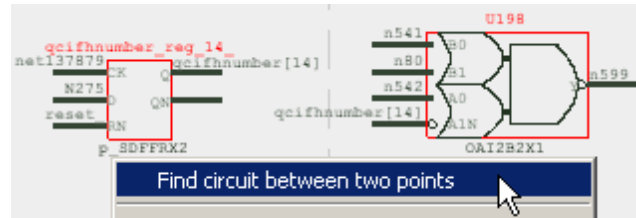


3.3 Show Circuit between two Gates

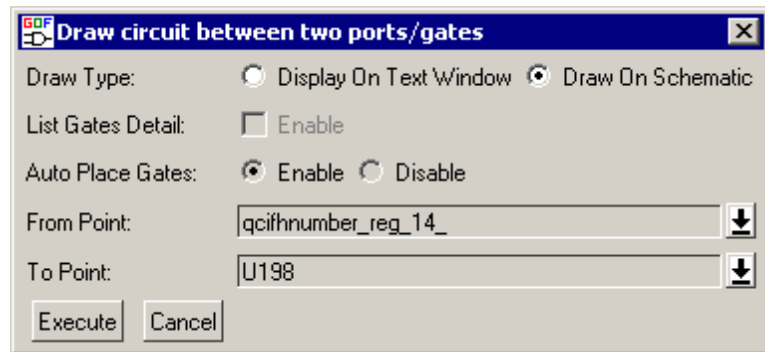
This function can draw the circuit between two gates or ports. For example, it can find out if there is a path from a flipflop to another flipflop. The function does not work across connector parts (marked with an 'X') which have different hierarchies on the left and right sides.

Find Circuit between two parts:

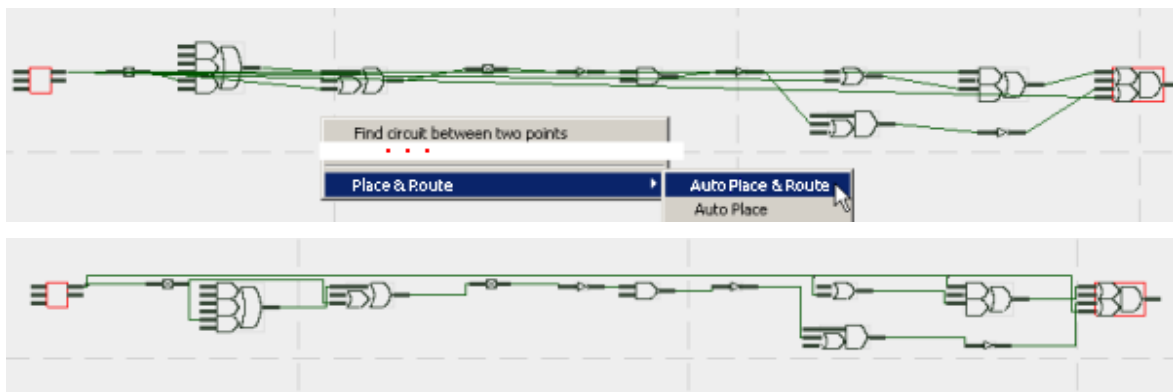
- Select just two gates or ports in the schematic.
- Right click and choose the **Find Circuit between two points** menu to open a dialog box (this menu is only available when two or more gates are selected).



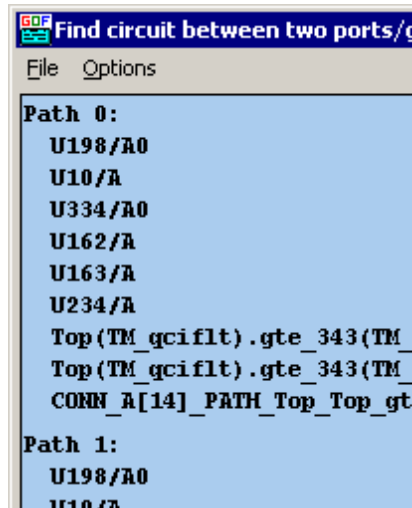
- **Draw Type** determines whether the results are sent to a schematic or to the netlist text window.
- **From Point** and **To Point** show the gates that the function will look for paths between.
- Press **Execute** to perform the search.



- If **Draw On Schematic** is selected, you can also choose to enable **Auto Place Gates** to automatically place the gates in a reasonable manner. To route the resulting nets, right click and choose **Place & Route > Auto Place & Route** from the context menu.



- If **Display On Text Window** is selected, then you have a choice of whether to enable or disable **List Gates Details** to control the amount of information that is generated. If enabled, a text window will pop up that lists all connections from the starting leaf cell to the ending cell.

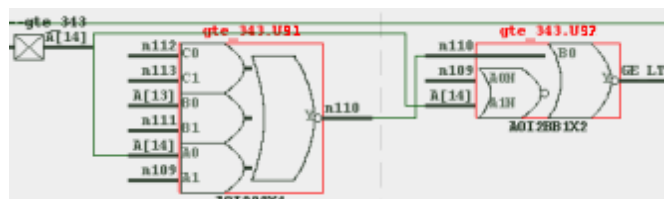
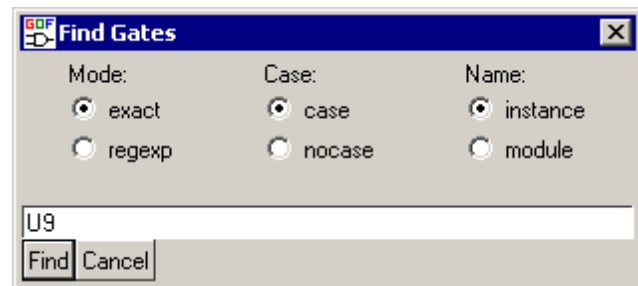
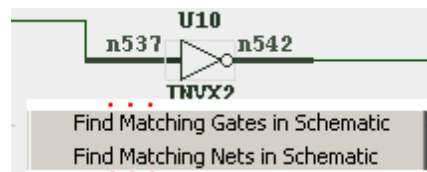


3.4 Search & List Info for Schematic

Use the right click context menus to search or find out more information about a particular gate. You can search for a gate or net inside the schematic by choosing one of the **Find Matching** menus. The **Find in GoViewer** menu will take you back to the source code displayed in GoViewer. The **List** menus will display text information about the particular gate or pin that is selected. The **List** windows display information that is hyperlinked and gates in the List windows can be sent to a schematic window.

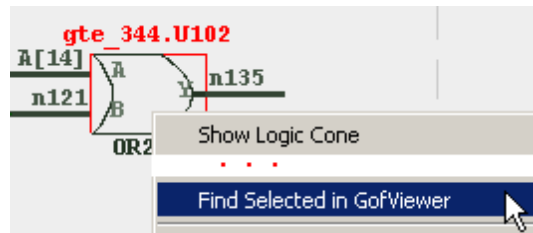
Search for Gates or Nets in a Schematic

- Right click anywhere in a schematic window and choose one of the **Find Matching** menus to open a *Find* dialog.
- Enter a partial search string. Any gates or nets in the schematic that partially match the search will be selected and the schematic will be adjusted to show the gates.
- In this example, the search string U9 found two matching gate instances: gte_343.U91 and gte_343.U97.



View the code for a particular gate:

- Left click on one gate (not the pins) to select it, then right click and choose **Find Selected in GofViewer**.



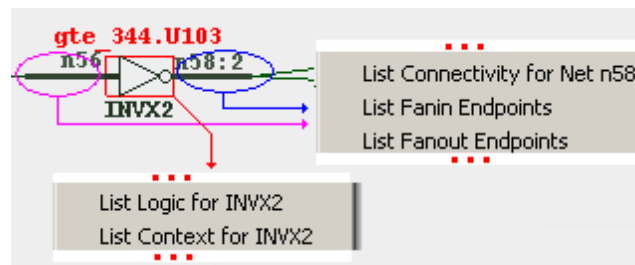
- This action will bring GofViewer to the top and highlight the code for the gate that you have selected.



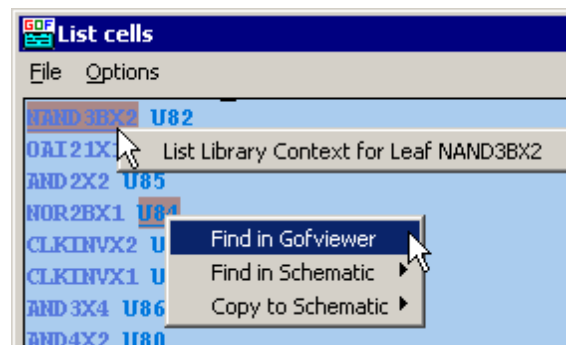
```
AND2X2 U101 ( .A(A[14]), .B(n121), .Y(n138) );
OR2X2 U102 ( .A(A[14]), .B(n121), .Y(n135) );
INVSX2 U103 ( .A(n56), .Y(n58) );
NAND2X2 U104 ( .A(n124), .B(n132), .Y(n70) );
OAI21X2 U105 ( .A0(n128), .A1(n129), .B0(n105),
```

Finding Information Using the List Menus:

- The context menus for pins and gates have several different **List** menus which provide information about the selected object.

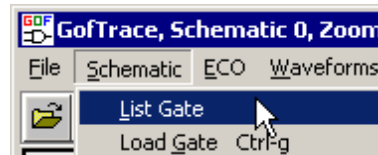


- List Connectivity** shows a list of gates that touch the pin.
- List Fanin Endpoints** show a list of gates that can affect the value of this pin.
- List Fanout Endpoints** shows a list of gates that are directly affected by this net.
- List Logic** shows the gate logic function for the selected gate.
- List Context** shows file information about the definition of the selected gate.
- Each List window has text that is hyperlinked to more information about a particular gate or leaf cell. **Right click** on the highlighted text to open the context menu for that type of object and choose what to look at.

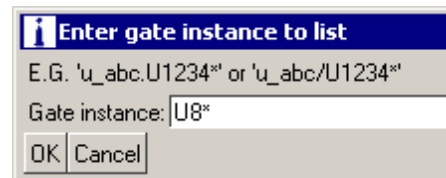


Find All the gates with a similar name and send them to a schematic:

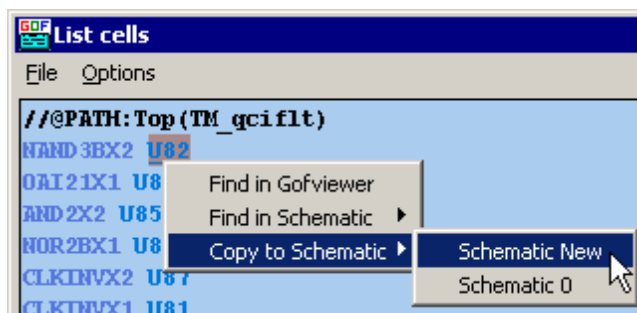
- Choose the **Schematic > List Gate** to open the *Enter gate instance to list* dialog.



- Type in a search string. Wild cards * are allowed. The path can be specified using '.' or '/' characters. The search is CASE SENSITIVE.
- Press **OK** to perform the search and open a list window.



- All instances matching the name are listed in a new window.
- To show an instance on a schematic, right click on an instance name and choose **Copy to Schematic > Schematic #**.



Selecting All Gates and Nets in the schematic:

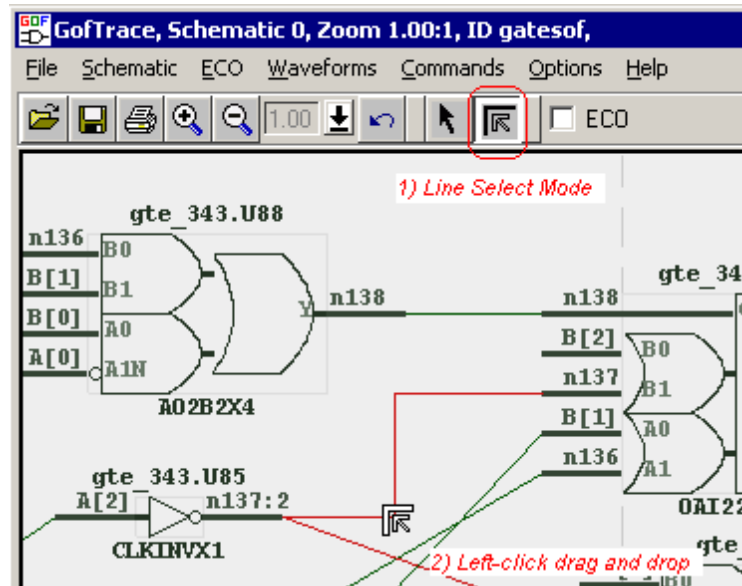
- Pressing the **CTRL** key and the **A** key at that same time will select all objects in the schematic window. **CTRL-A**

3.5 Place and Route Schematic

Initially a schematic is drawn with straight wires between connections so that you can quickly view the circuit and get an idea of the gates and buffers that have been added by the synthesis tool. The nets can then be automatically routed using the **Place & Route** menus, manually routed with the **Line Select Mode**, or a combination of manual and automatic techniques.

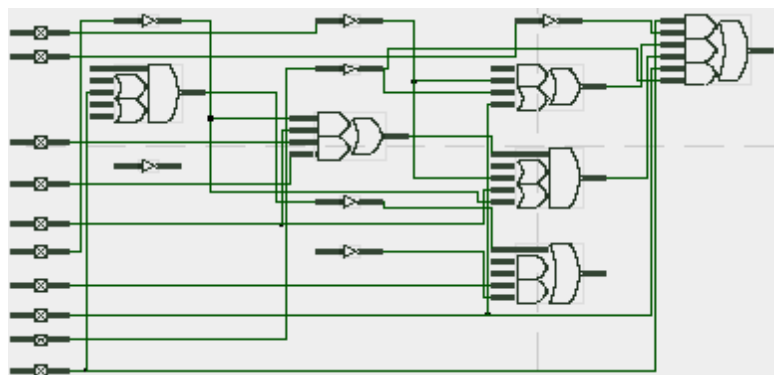
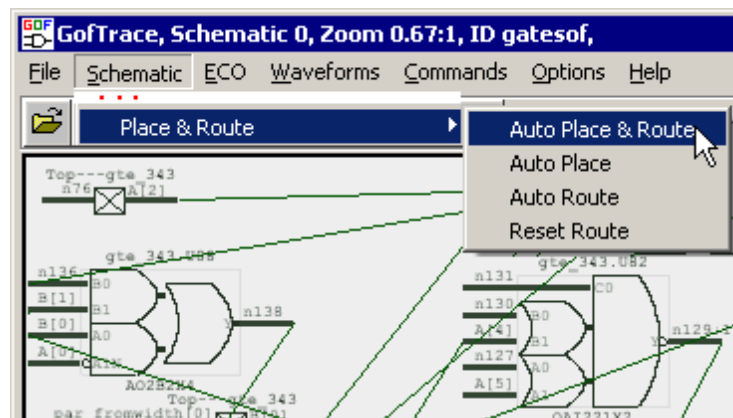
Manually Route a Net:

- Press the **Line Select Mode** button on the button bar, so that the left mouse button will be able to route the nets. Right-click menus and gate dragging are NOT available in this mode.
- **Left mouse click down** on a net to insert a corner point **and drag** to route the net.
- **Right mouse click** on a corner point to remove it.
- To discard the routing of a net: first select it, press the delete key to remove it, then **middle mouse click** on a pin to redisplay the unrouted net.

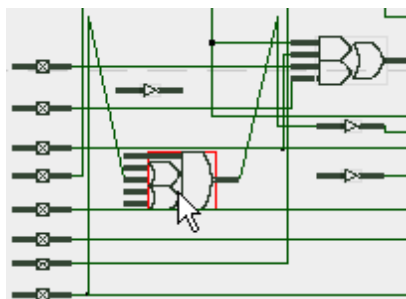


Automatically Place and Route an Entire Schematic:

- The **Place & Route** menus operate on the entire schematic and are located under the **Schematic** main menu or any of the right click context menus.
- **Auto Place and Route** rearranges the gates and routes the nets.
- **Auto Place** rearranges the gates without routing the nets.
- **Auto Route** routes the nets without changing gate placement.
- **Reset Route** returns the nets to the default straight connection display (unrouted state).



- After a place and route, you can still rearrange gates by left-click drag and drop. Then when the gates are placed, you can choose to **Auto Route** to preserve the placement and just route the new nets.

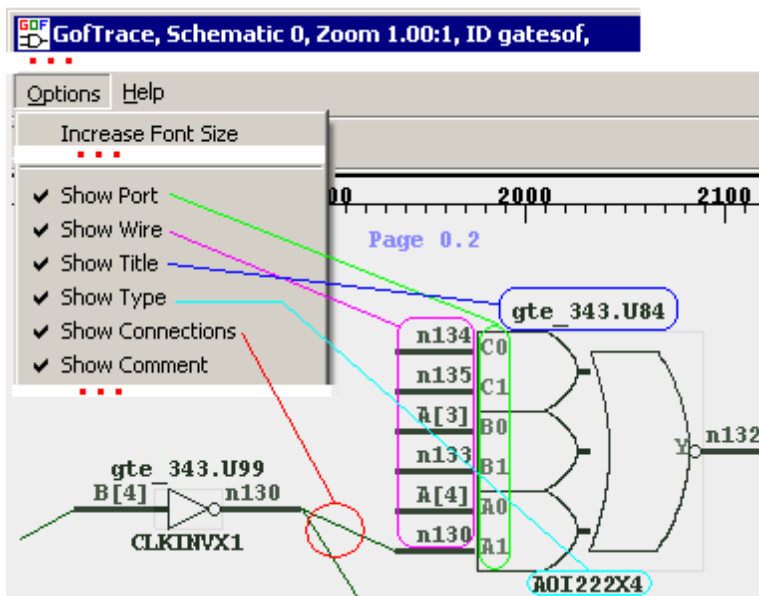


3.6 Display Settings and Comments

The schematic display is fully customizable, so you can control what types of objects are displayed, how the logic is drawn, and the colors for different objects. Comments can also be added to a gate or to the schematic sheet.

Control the types of objects shown on the schematic:

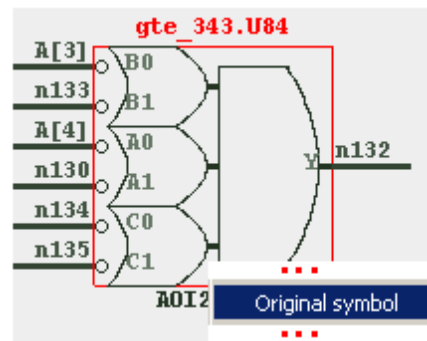
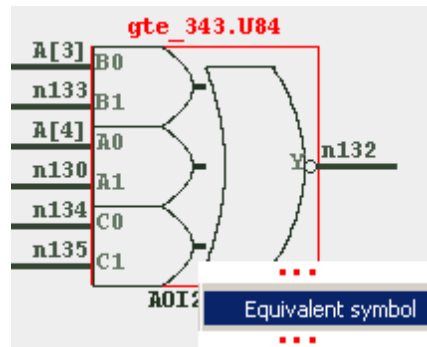
- The **View** menu controls what types of objects are displayed in the schematic window.



Control the Logic Display for each Gate

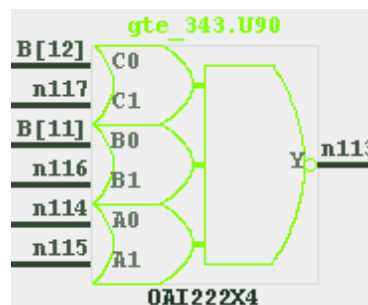
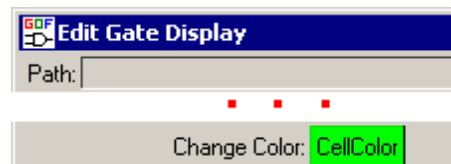
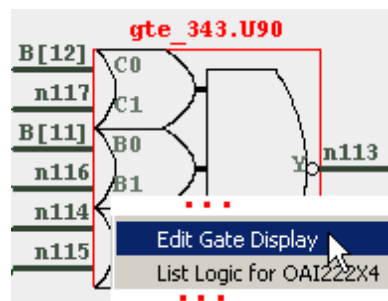
To make the logic easier to follow, sometimes it is helpful to switch a cell to an equivalent gate symbol, so that the symbol is composed of either AND gates or OR gates. Using DeMorgan's Laws, GofTrace can switch any two or more input cell from its original symbol to its equivalent representation.

- Select a cell, then right click and choose **Equivalent Symbol** from the context menu. This menu is only available when the logic symbol's components are ANDs and ORs.
- You can switch back to the other symbol by selecting **Original Symbol** from the right click context menu.



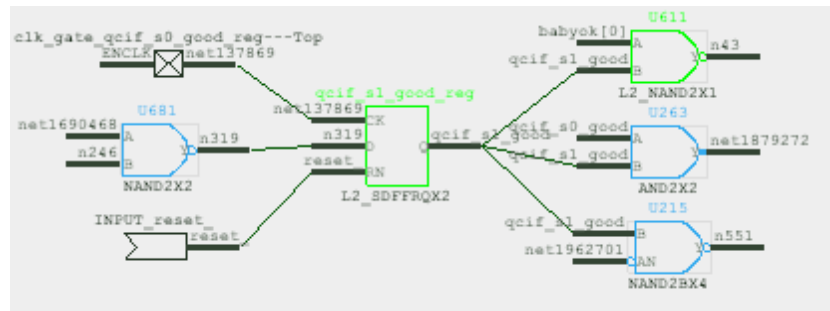
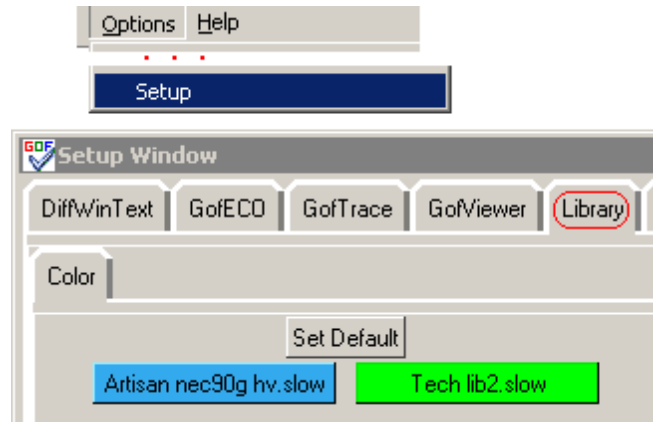
Change the Color of a Gate:

- Right click on a gate and choose **Edit Gate Display** from the context menu to open the *Edit Gate Display* dialog.
- Press the **Cell Color** button to open the *Choose Color* dialog.
- Choose a color and close both dialogs. The gate will be redrawn with the color that you chose.



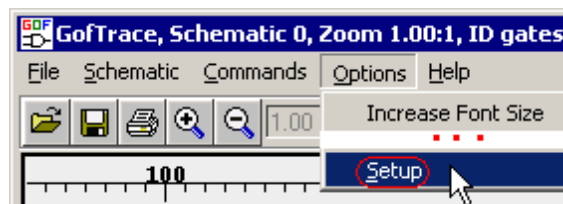
Change the color of cells defined by a particular library:

- The *Setup Window* dialog has a **Library** tab that controls the colors for cells in a particular library. Open this dialog using the **Options > Setup** menu.
- Click on the colored button to change the color for the library.

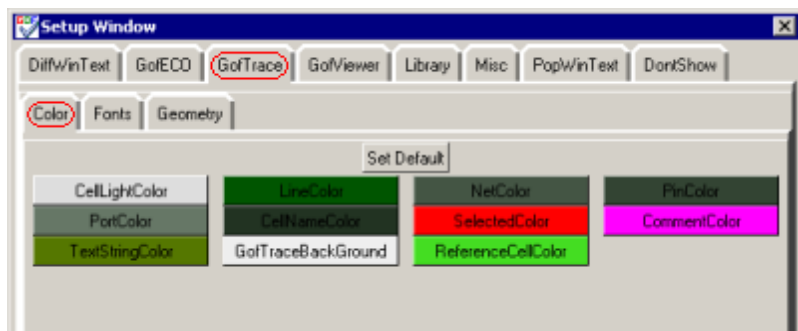


Control the color of comments and different elements within GofTrace

- Choose the **Options > Setup** menu to open the *Setup* window.

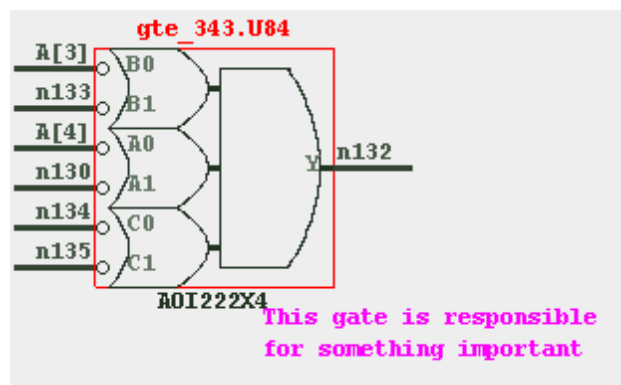
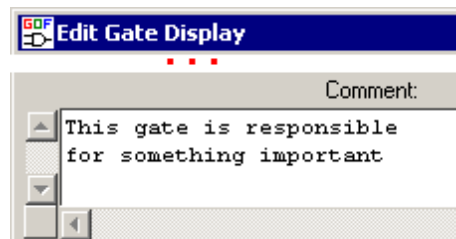
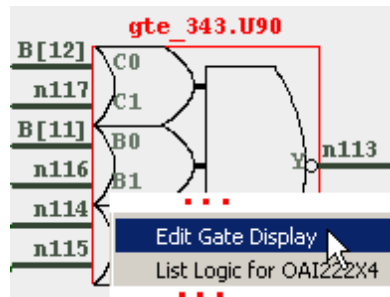


- Choose the **GofTrace** tab, then the **Color** tab, and then press on any button to change the color of the stated object.



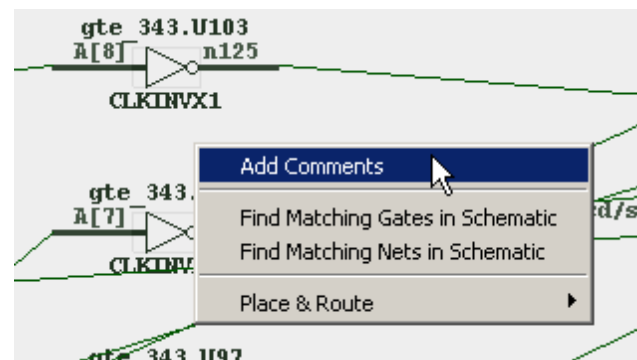
Add a Comment to a Gate

- Right click on a gate and choose **Edit Gate Display** from the context menu to open the *Edit Gate Display* dialog.
- Type a comment into the Comment Edit box and click OK to close the dialog.
- The resulting comment is attached to the gate and will keep the same distance when the gate is moved.

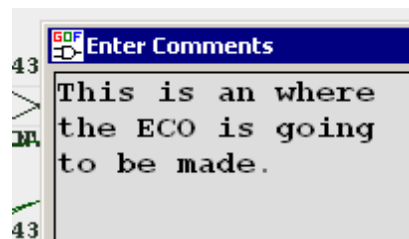


Add a Comment to the schematic sheet

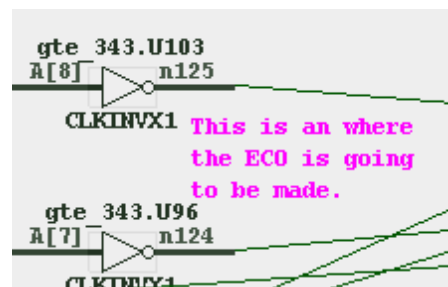
- Left click into a blank space on the schematic, so that no gates or nets are selected.
- Right click and choose **Add Comments** from the context menu to open the *Enter Comments* dialog. This menu is only available when no gates or nets are selected.



- Type in the comment into the edit box of the dialog.
- Press **OK** to close the dialog.



- The comment can be moved by left clicking and dragging it around.



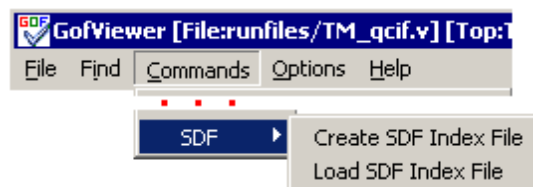
3.8 Accessing SDF timing files

From within a schematic, you can get the timing information for each gate or net in the schematic if you have a Standard Delay Format (SDF) file for the netlist. SDF files are timing files generally created by timing estimation tools after a chip has been placed and routed. These are the same SDF files used by Verilog simulators to perform timing simulations on a synthesized design.

Create and Load an SDF Index file:

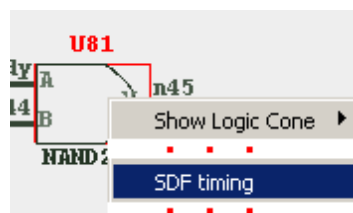
Since SDF files can be very large, GOF does not load an SDF file directly. Instead, GOF creates an SDF index file so that it can quickly find timing information on demand as required.

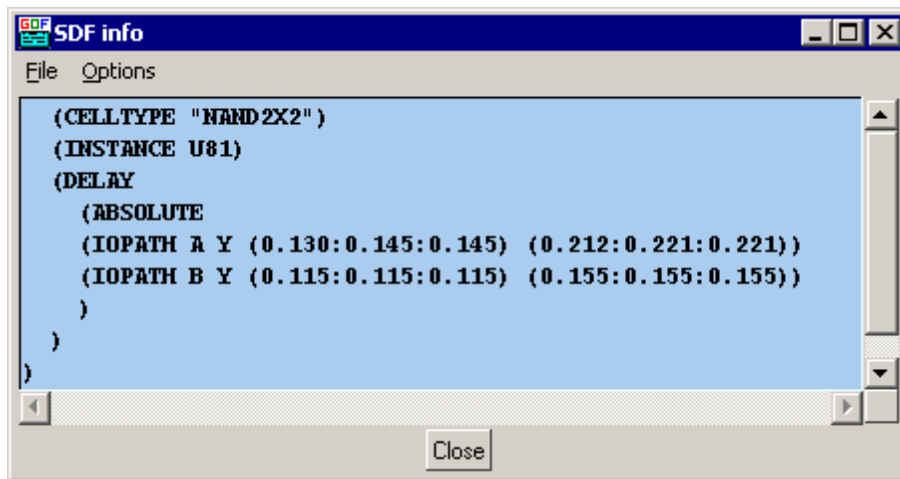
- In GofViewer, choose the **Commands > SDF > Create SDF Index File** menu to open a File dialog.
- Locate the SDF file for your design and press OK to close the dialog, create the index file with a **.SIF** extension, and then load it into GOF.
- The next time GOF is launched, if the SDF file is the same, then you can just use the **Commands > SDF > Load SDF Index File** to load the index file and skip the creation step.



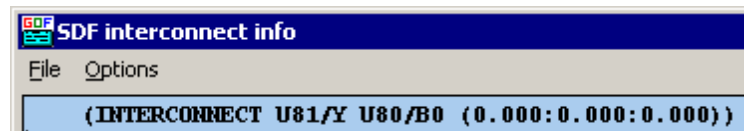
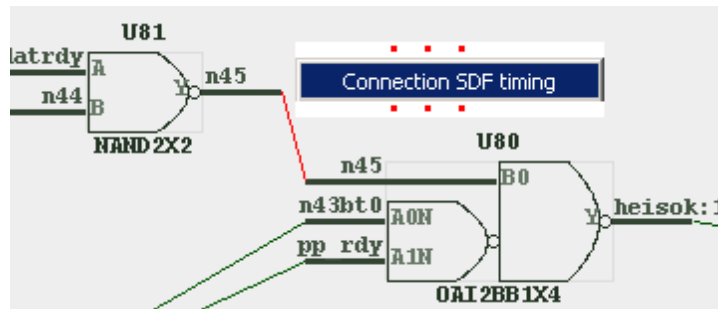
SDF Timing for Gates and Nets:

- To get timing information on a selected gate, right click and choose **SDF timing** to open a listing window that shows the delay information.





- To get information on a selected net, right click and choose **Connection SDF timing** from the context menu.



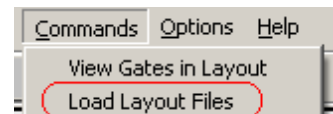
3.9 Layout Viewer

GOF can display gates from a schematic in a layout viewer that shows where the gates are placed in the backend layout. To enable this feature, you must load the files generated by your layout tool that contain the layout information into GOF. The following layout file types are supported:

- LEF** library exchange format, loaded with the **-lef** command.
- DEF** design exchange format, loaded with the **-def** command.
- PDEF** physical design exchange format, loaded with the **-pdef** command.

Load the LEF, DEF, or PDEF files:

- In GofTrace, choose the **Commands > Load Layout Files** to load in the layout information from a LEF, DEF, or PDEF file.

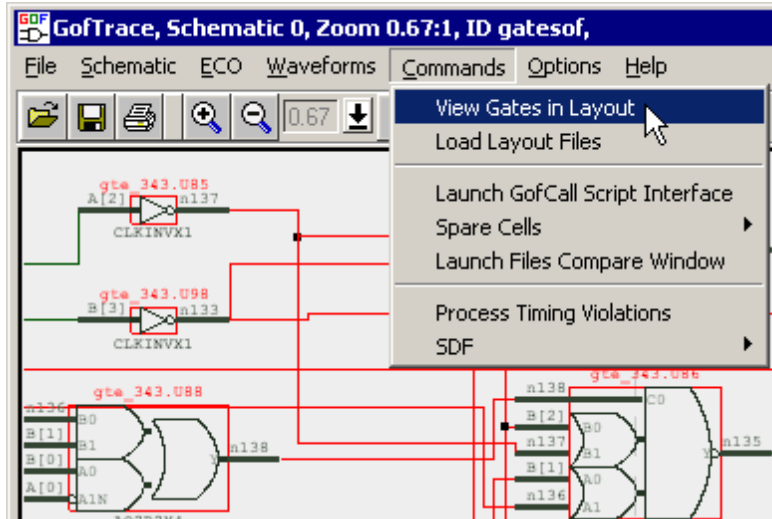


- The layout files can also be added to the project when launching GOF using the **-lef**, **-def**, or **-pdef** command line option that matches the type of file you are loading (see [Chapter 1: Launching GOF with a Batch script](#)). For example:

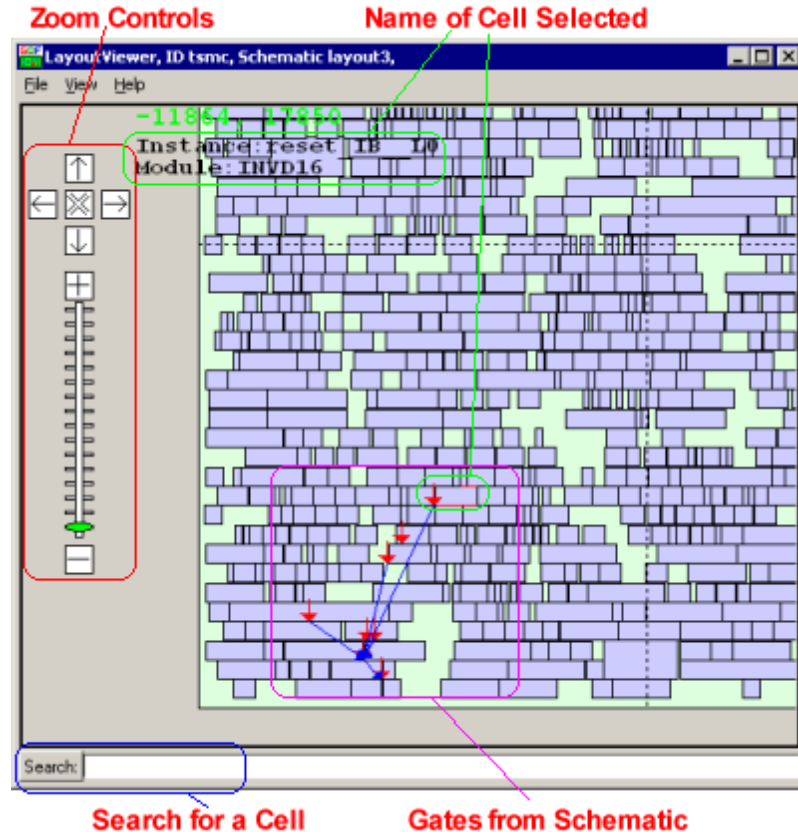
```
gof -synlib synthesis.lib mydesign.v -def mydesign.def -lef mylib.lef
```

Select Gates and View in Layout:

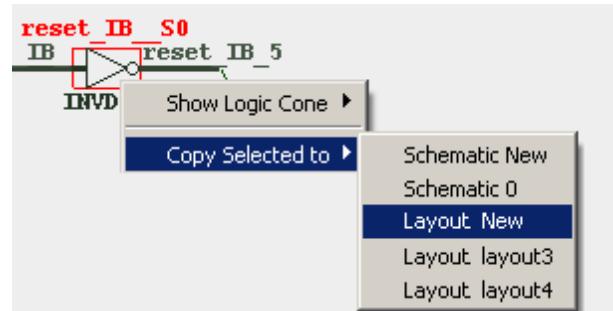
- Select some gates in a schematic. Multiple gate selection can be achieved using **area selection**. To select an area, **left click down and drag** on an empty space in the schematic to open a selection box, then drag to widen the box until the selection surrounds the gates of interest. More gates can be added to the selection by pressing the **CTRL** key while **left clicking** on objects. Alternatively, **<CTRL>-A** can be used to select all the gates in a schematic.



- Once the gates are selected, choose **Commands > View Gates in Layout** to open the Layout viewer. The selected gates will be shown with red arrows pointing to them.



- You can also show selected gates and nets in a Layout Viewer window using the right-click context menu **Copy Selected to > Layout 0**.



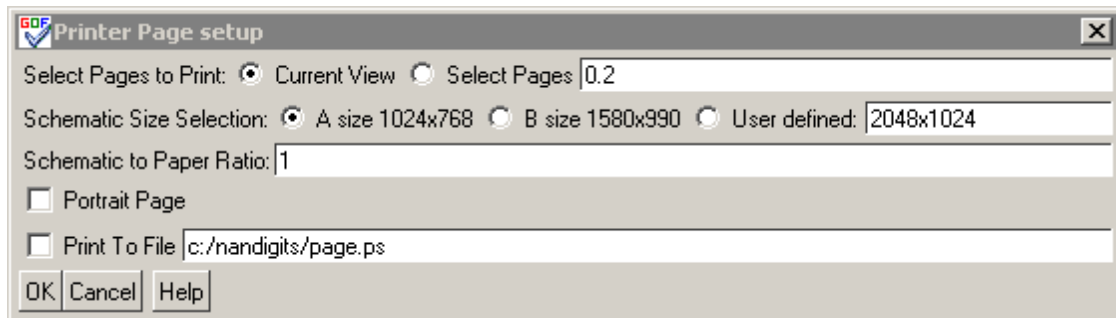
- The **Search** box searches for a cell in the layout window. The search string accepts wildcards (e.g. NAND*).

Example of Layout Features:

In the **SynaptiCAD > Examples > GOF_features** directory, the **case_layoutview.bat** file launches GOF with design layout files loaded so that you can experiment with the features in this section.

3.10 Printing a Schematic

To print a GofTrace schematic, choose the **File > Print** menu to open the *Printer Page Setup* dialog. Below are a list of the options you can set when printing a schematic:



- Select Pages to Print** determines what will be printed. The **Current View** button will print so that the top left corner of the current schematic view will line up with the top left of the printer paper. Selecting the **Select Pages** radio button allows you to enter a comma separated list of the pages to print (e.g. "0.2,1.2,2.3" will print three pages).
- Schematic Size Selection** specifies the size of paper that will be printed. The default is 1024X768 which corresponds to an A4 paper size of 11x8.5 if the *Schematic to Paper Ratio* is 1. You can choose a larger schematic size and shrink it down to fit on a smaller paper size by changing the schematic to paper ratio.
- Schematic to Paper Ratio** determines whether the schematic will be shrunk down to fit on the paper. At the default value of 1, the schematic is printed at full size on the paper. If you choose a **User Defined** size of 2048x1024, you can set the **paper ratio** to 0.5 to fit a big schematic onto an A4 paper size.
- Portrait Page**, if checked, will print with portrait orientation instead of landscape.
- lpr command (Linux version only)** passes additional command-line options to the printer (e.g. -Pmyprinter specifies to print to a printer called myprinter). See your Linux manual for available **lpr** options.
- Print to File**, if checked, creates a Postscript file instead of printing to the printer.

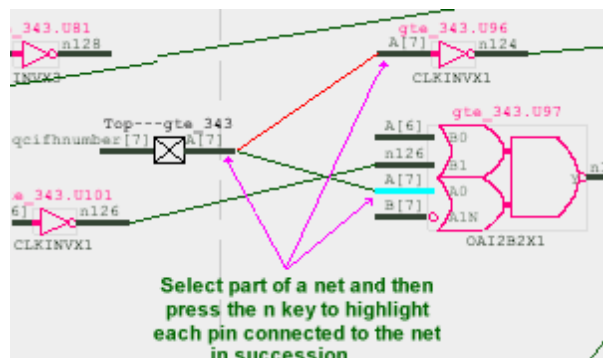
3.11 GofTrace - Menus and Buttons

Below is an overview of the menus and buttons in GofTrace schematic windows.

Key Commands:

- Pressing **<CTRL>-A** key combo selects all objects in the active schematic window.
- When one net is selected, press **'n'** key to highlight connected pins and keep on pressing 'n' to jump between all pins that the selected net connects.

<CTRL>-A to select all



- When a set of gates or nets are selected, press **<CTRL>-digit** (where digit is 0 to 9) to save the selection for later re-use.
- You can restore a saved selection at any time by pressing the appropriate **digit** (without the **<CTRL>** key).
- Press **<SHIFT>** and roll mouse middle wheel to move schematic window horizontally
- <CTRL>-C** and **<CTRL>-V** copies between schematics

<CTRL> and 0, 1, 2, 3, 4, 5, 6, 7, 8, or 9

To save a selection

0, 1, 2, 3, 4, 5, 6, 7, 8, or 9

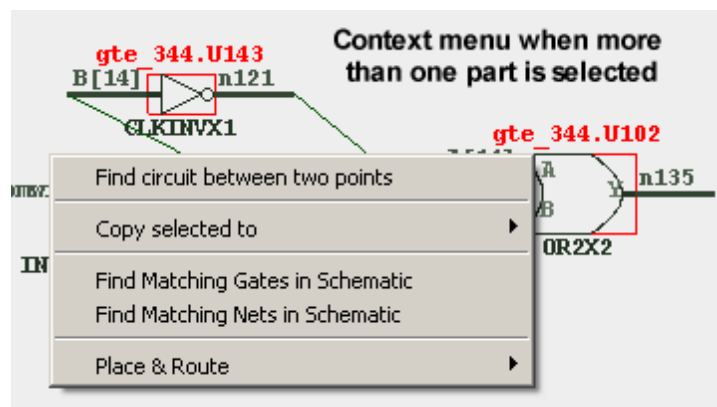
To restore a saved selection

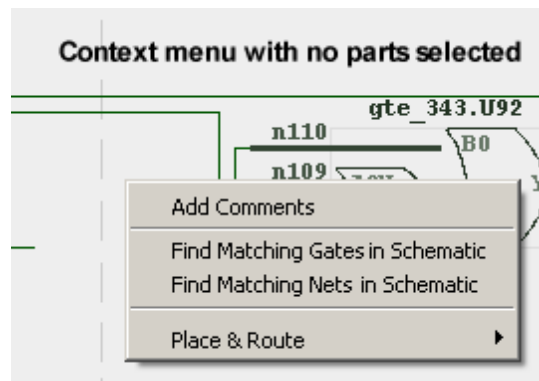
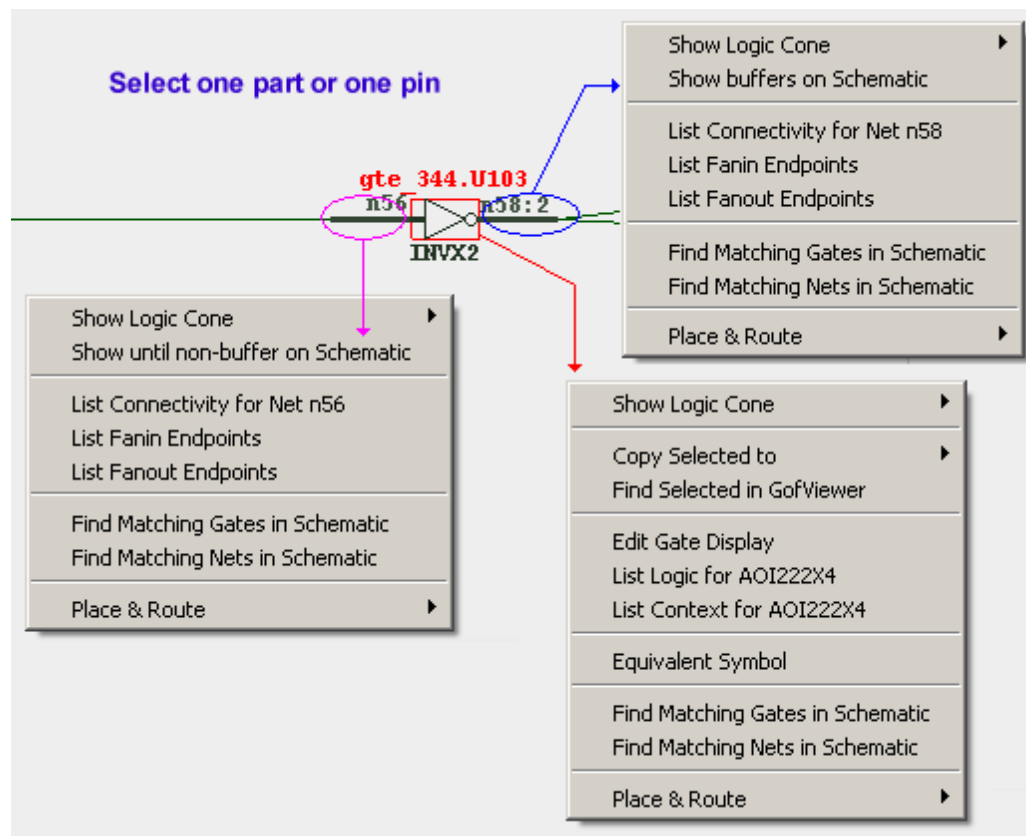
SHIFT and mouse roller to horizontally scroll

<CTRL>-C and <CTRL>-V to copy and paste

Context Menus:

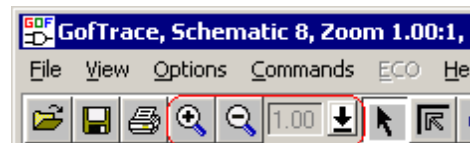
The context menus for GofTrace change depending on what is selected. Below are some images that will give you an idea of where the different commands are located.



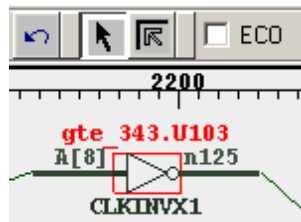


GofTrace Buttons:

- The **zoom** buttons change the view size of the screen. Maximum zoom is 1. The little display shows the numeric zoom level



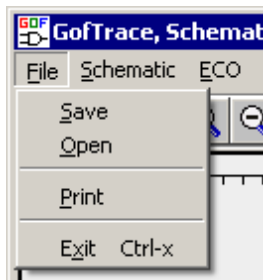
- The **Undo** button reverts the schematic view back to it's previous state (i.e. undoes your last change to how schematic is displayed). This button is grayed out if you are in ECO mode (in ECO mode, you must use the ECO Undo button, see [Section 4.1: Enable ECO](#)).
- The **Pointer Mode** button is the default editing mode where right click menus and drag-and-dropping of gates work.
- The **Line Selection Mode** button is used to manually route nets on the schematic (see [Section 3.5: Place and Route Schematic](#)).
- Checking the ECO box puts the schematic in a state that allows the netlist to be altered (see [Section 4.1: Enable ECO](#)). In this mode, gates can be added, replaced, and deleted. Changes to an ECO Schematic will change the netlist itself, instead of just how the netlist is displayed.



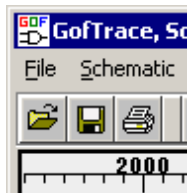
File Menu

The GofTrace **File** menus operate on the schematic image only. If you have been making ECO changes to the netlist, then use the saving features in [Section 4.1: Enable ECO, Undo, and Save](#). The schematic image and place-and-route information are stored in files with a **.st** extension.

- **Save** writes out the schematic image and place and route information to a file with an extension of **.st**.
- **Open** loads a **.st** file and displays the schematic.
- **Print** sends the image to the printer.
- **Exit** closes the GofTrace window.

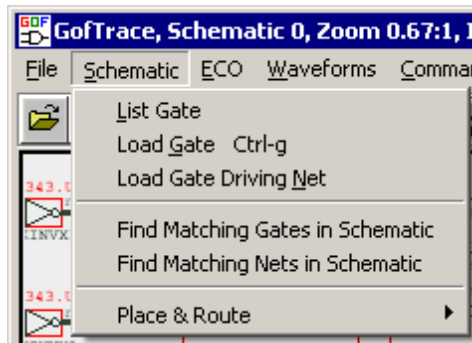


- The three buttons on the left side of the button bar also perform the Schematic **Open**, **Save**, and **Print** functions.



Schematic Menu:

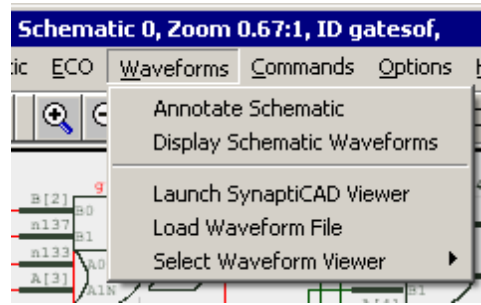
- **List Gate** shows a list of all gates that match a case sensitive search (see [Section 3.4: Search & List Info for Schematic](#)). These gates can subsequently be sent to a schematic using the right-click context menus.
- **Load Gate** and **Load Gate Driving Net** open a dialog that lets you enter a search string. All gates that match the search string (or gates driving nets that match the search string) will be displayed in the schematic (see [Section 3.1: Open a Schematic Window](#)).



- The **Find Matching Gates/Nets in Schematic** will highlight matching gates or nets and rearrange the schematic so that you can see them (see [Section 3.4: Search and List Info for Schematic](#)).
- The **Place and Route** menus rearrange the schematic and draw the nets using horizontal and vertical lines rather than straight lines (see [Section 3.5: Place and Route Schematic](#)).

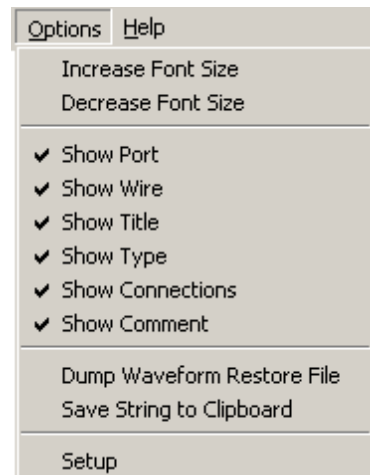
Waveforms Menu

SynaptiCAD's Waveform viewers can be used with GOF to show logic states on the schematic. These menus are covered in [Chapter 6: Waveform Viewer Support](#).

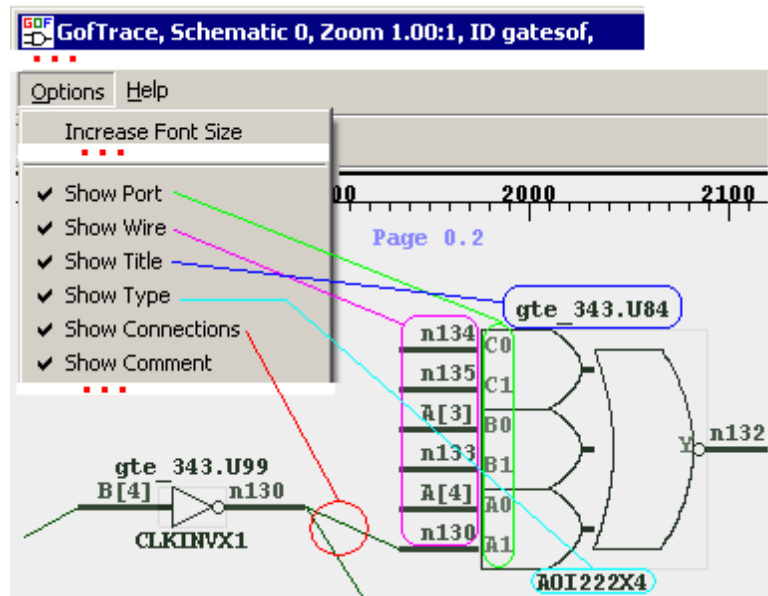
- **Annotate Schematic** writes the logic states from simulation data onto the pins of the gates.
 - **Display Schematic Waveforms** opens the waveform viewer with the nets that are show on the schematic.
 - **Launch SynaptiCAD Viewer** opens the waveform window with all signals hidden.
- 
- **Load Waveform File** loads the VCD file into the Viewer and prepares the schematic to be annotated.
 - **Select Waveform Viewer** specifies which of SynaptiCAD's viewers to load.

Options Menu

- **Increase Font Size** and **Decrease Font Size** affect the font size in the schematic window.
- **Dump Waveform Restore File** is an older method of working with waveform data (see [Section 6.3: Create Waveform Restore File](#)). A new graphical method is covered in [Section 6.2: Show logic states from VCD file](#).
- **Save String to Clipboard** puts GOF in a mode so that when you click on an instance name or a net name the name will be copied to the clipboard (no need to press **Ctrl-C** to copy). The **Ctrl-V** will paste into edit boxes or other text based tools.
- **Setup** opens a dialog that lets you set the colors of each type of object in a schematic (see [Section 3.6: Display Settings and Comments](#)).

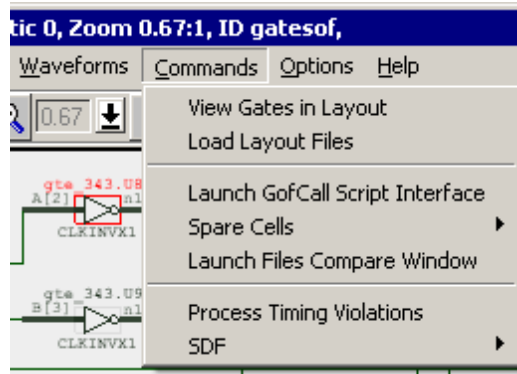


- The **Show** menus control which object types are shown on the schematic.



Commands Menu

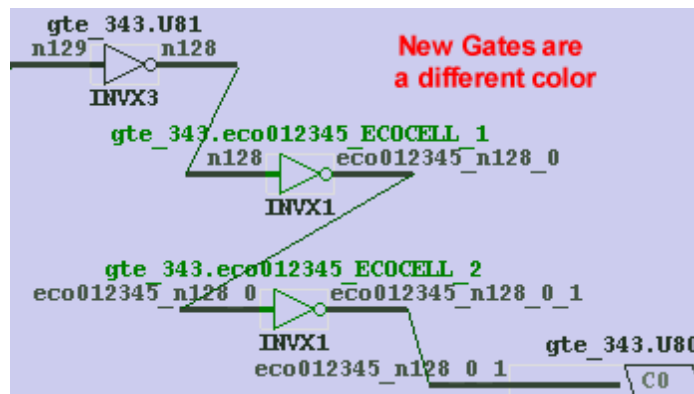
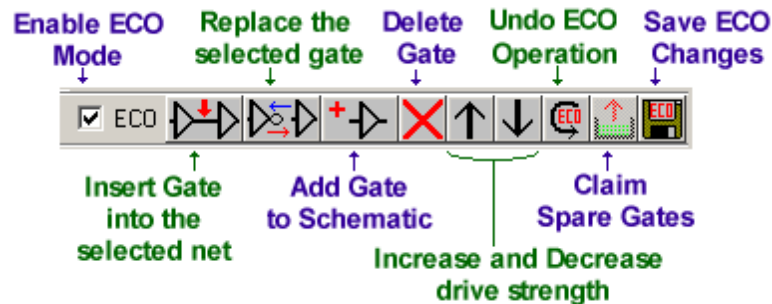
- **View Gates in Layout** shows the selected gates in a layout window so that you can see the physical placement of the gates (see [Section 3.9: Layout Viewer](#)).
- **Load Layout Files** will load the LEF, DEF, or PDEF file needed for the layout viewer (see [Section 3.9: Layout Viewer](#)).



- **Launch GofCall Script Interface** opens a window that runs Netlist Processing APIs in interactive mode (see [Section 5.1: Interactive window & batch files](#)).
- **Spare cells** menus are used to create the spare cells file and load it. This is used in metal only ECOs (see [Section 4.6: Metal Only ECOs](#)).
- **Launch Compare Files Window** opens a window that will compare two netlists and show the differences between them (see [Section 2.5: Diff Utility for RTL files](#)).
- **Process Timing violations** displays timing violations from a Prime Time file using the schematic window and several listing windows (see [Section 2.6: Process Timing Violations](#)).
- **SDF** creates an index for SDF file and load SDF index file. When SDF file is large, indexing the file first and load in the index file will be much faster (see [Section 3.8: SDF timing](#)).

Chapter 4: GofECO - schematic editor

GofECO is a special mode of the schematic window that lets you make changes to the netlist. Normally, when you are working in a GofTrace schematic window, actions like adding or deleting gates and nets only affects what gets displayed in the schematic and does not change the actual netlist. When you put the schematic in ECO mode, such changes will actually modify the netlist. Once ECO mode is activated, the ECO buttons will take over many of the schematic features like saving files, undoing operations, and adding or deleting gates and nets, to ensure you are aware that the actual netlist is being changed and not just the schematic view.



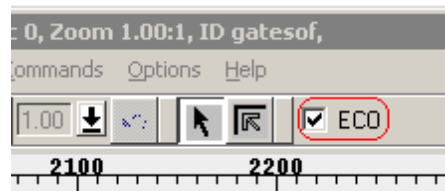
In addition to graphically performing ECOs, there are also two scripting methods for performing ECO operations: **Logic Cone ECO** and **GofCall API scripts**. The Logic Cone ECO interface accepts a script that changes the entire logic cone that drives a particular part and replaces it with a newly re-synthesized netlist (see [Section 4.5: Logic Cone ECO](#)). The GofCall API mimics most of the graphical ECO commands (see [Chapter 5: GofCall - netlist processing](#)).

4.1 Enable ECO mode, Undo, and Save

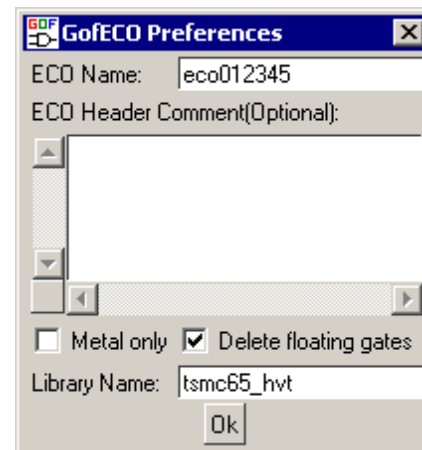
To make changes to the netlist you must first put GOF into ECO mode. After the tool enters ECO mode, the schematic background will change to blue and the ECO button bar will appear. From this point forward, any changes that you make will be recorded and written out to the ECO files.

Enable ECO Mode:

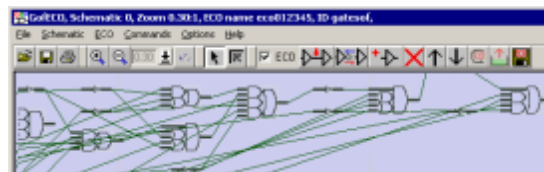
- Check either the **ECO** box on the button bar or the **ECO > Enable ECO Mode** menu to open the *GofECO Preferences* dialog. This dialog is used to specify the different options for the ECO. During the ECO if you need to change these options than this dialog can be reopened by choosing **ECO > ECO Preferences** menu.



- ECO Name** is a string that is used as a prefix to name new gates and nets that are added during the ECO session. It should be unique to avoid a conflict with any existing string in the netlist. For example, it can be a bug ID string, like **bug23456**.



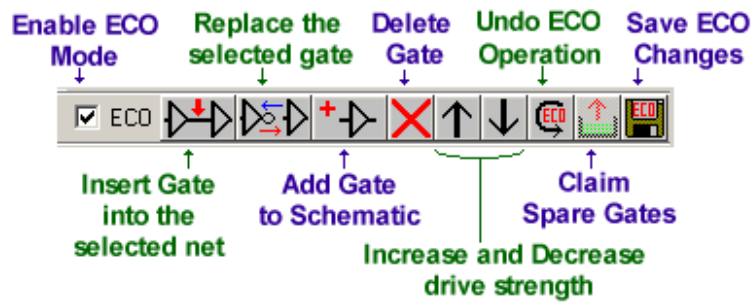
- ECO Header Comment** is a comment that will be inserted into the ECO result files.
- Delete floating gates** deletes gates that no longer have any fanout after the ECO is performed (i.e. the nets connecting to the output pins of those gates were deleted during the ECO so these gates no longer have any purpose in the design).
- Library Name** is used when writing out the ECO results as a Synopsys DCSH/TCL file. This library name is used as a prefix to leaf cell types written out in the ECO results file. If you don't need to specify a particular library name, leave it with the default library name.
- Metal only** option is for making changes to silicon when only metal layers can be changed (see [Section 4.6: Metal Only ECOs](#)).
- Notice that the schematic background is now blue to indicate that ECO mode is enabled. The background color is controlled through the **Options > Setup** menu and the **GofECO** tab in the *Setup Window* dialog.



ECO Mode buttons for Undo and Save:

Enabling ECO mode also causes the ECO button bar to appear.

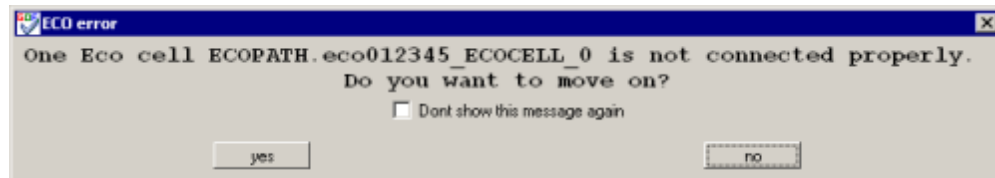
- While in ECO mode the Undo function is controlled by the **Undo ECO** button. The regular schematic Undo will be disabled.



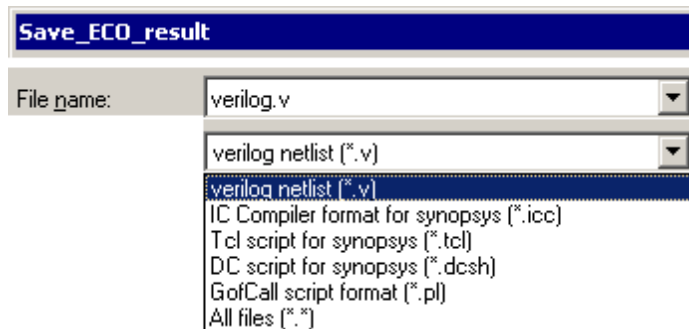
- Only the **Save ECO** button or menu will save the ECO operations that you make in the ECO mode. The regular **File Save** button and menu will only save the schematic, not the ECO operations.



- Before saving ECO changes, GofECO will check if any pins are disconnected, and show a warning if more changes are needed.



- You can choose to save ECO results to a Verilog file, a Synopsys TCL script file, or to a DC script file.



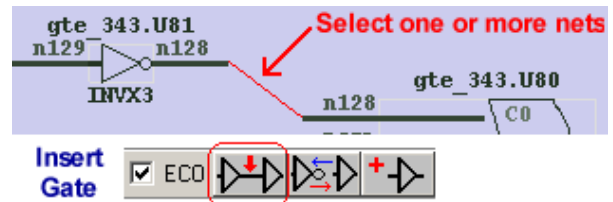
- After saving the ECO results, it is a good idea to use an equivalence checker or perform a post-ECO simulation to verify the modified netlist.

4.2 Insert, Replace, or Add a Gate

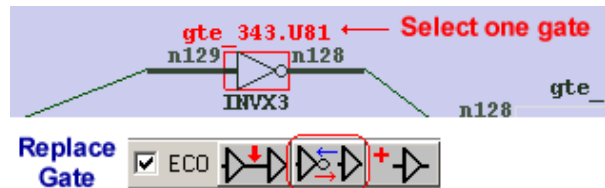
ECO gates can be inserted into net connections between gates, replace existing gates, or just added to the schematic. These actions are done by selecting the object to act on and pressing the **Insert Gate**, **Replace Gate**, or **Add Gate** button. A selection dialog will allow you to choose the gate type, and a connection dialog will allow you to hook the gate up. After a gate is added to the schematic, you can use the techniques in [Section 4.3: Add or Delete Connections](#) to change how the gate is connected.

Press the **Insert**, **Replace**, or **Add** button to start the ECO change:

- To **Insert a gate** into a net, select one or more **nets** and then press the **Insert Gate** button.



- To **Replace a gate** with another type of gate, select the **gate** and press the **Replace Gate** button.

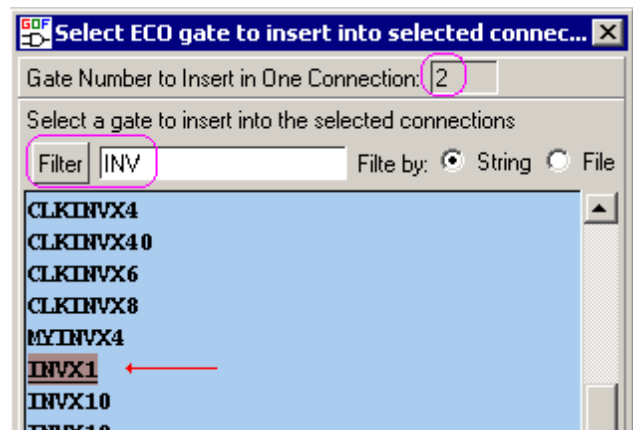


- To **Add a gate** to the schematic, press the **Add Gate** button (there is no need to select any gates or nets first).



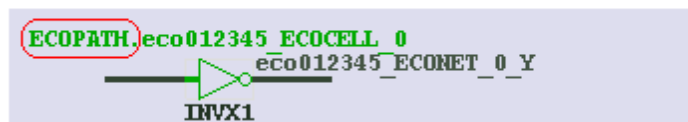
Select the gate to insert, replace, or add to the schematic:

- Pressing the Insert, Replace or Add button will open a selection dialog.
- If you are **Inserting**, you can choose to insert multiple gates by entering a number into the **Gate Number to Insert in One Connection** box. In this example, two inverters will be inserted in series into each selected net.
- Select a gate from the leaf cell library list. The list can be filtered by typing in a string and pressing the **Filter** button.
- Press the **OK** button to close the selection dialog.

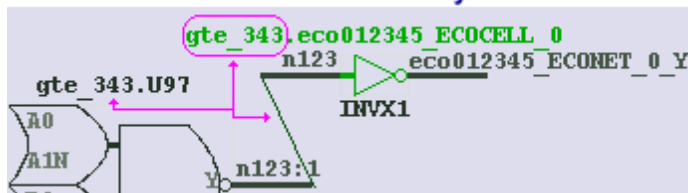


- If you are **Adding** a gate, closing the selection dialog will add the gate. The hierarchical scope is not known for newly added gates, so the first connection you make will set the hierarchical level for the new gate. The next section will show you how to make connections to the gate using the middle mouse button.

Undefined Hierarchy

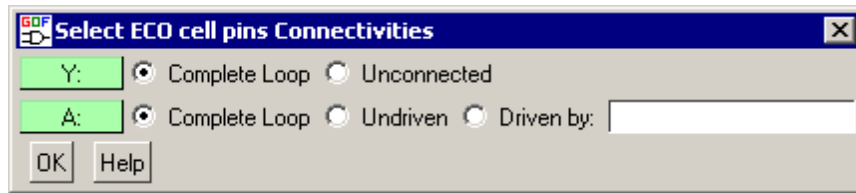


Connection Defines Hierarchy

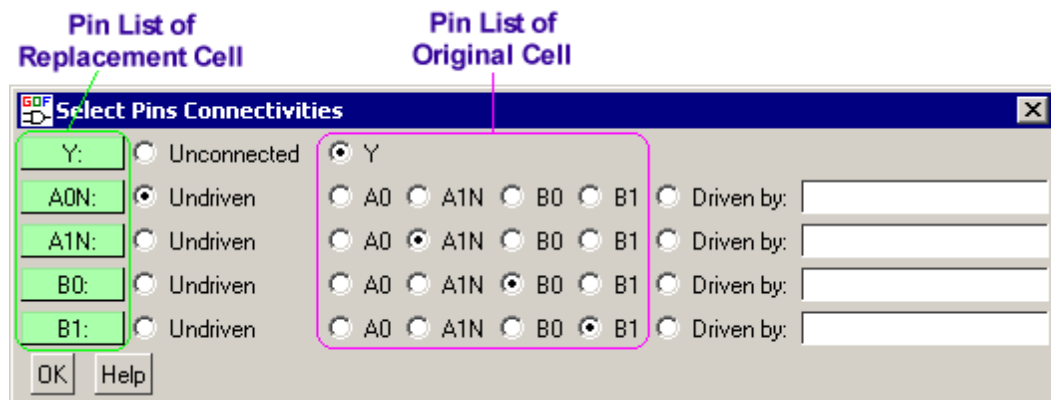


Specify the connections for each pin:

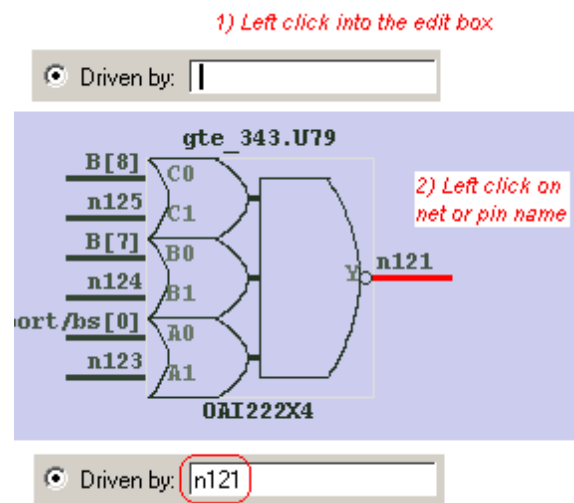
- After the selection dialog closes, a Connections dialog will open. Each pin of the new gate will be listed in the dialog.



- If you are **Inserting**, the **Complete Loop** option inserts that pin into the selected net.
- Unconnected** and **Undriven** leave the pin unhooked. You can use the techniques in [Section 4.3: Add or Delete Connections](#) to hook the pins up at a later time.
- If you are **Replacing**, GOF will attempt to match the pins from the original and new gate using the pin names.

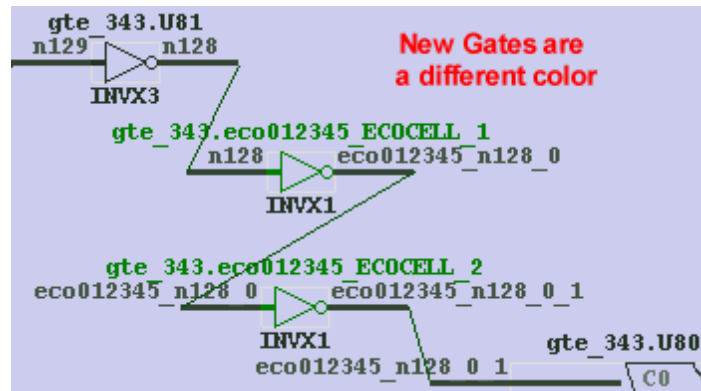


- To connect to a net that is different than the selected net, choose **Driven By** and type in the name of the net to connect (e.g. **n121**).
- Or use the **fast way**, by left clicking into the **Driven by** edit box to activate it, then left clicking on the net or pin that should drive the new pin.



- Press the **OK** button to close the dialog and complete the ECO change.

- The new gates will be displayed in green to indicate they are ECO gates.
- Newly created nets are named using a combination of the **ECO Name** (specified in the GoFECO Setup dialog), the original schematic net name, and an index number.

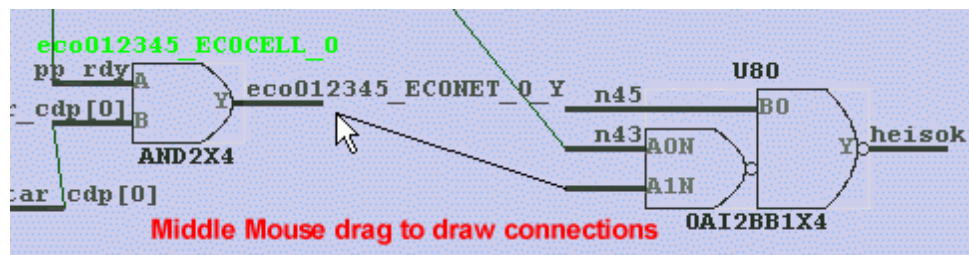


4.3 Add or Delete Connections

Nets can be deleted and gates connected to new pins.

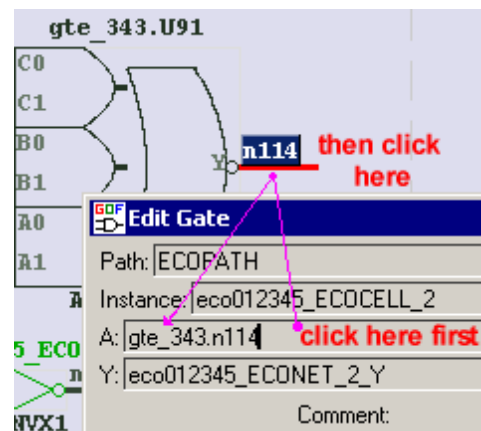
Use the middle mouse button to add net connections

- **Middle Mouse click** on an **input pin** of a gate and **drag** the mouse to an **output pin** and drop to make a connection. An output pin on one gate must be connected to an input pin on another gate. Output pins cannot be connected to other output pins, and input pins cannot be connected to other input pins.



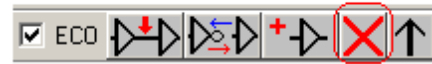
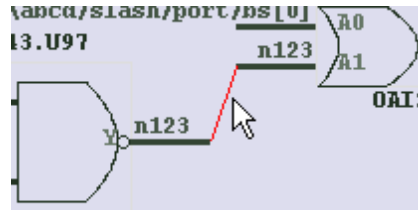
Edit Gate dialog

- **Left click** on a gate to select it, then **right click** and choose **Edit gate** from the context menu to open the *Edit Gate* dialog.
- This dialog lists the pin connections for the gate. For unconnected pins, you can type in the name of the net to connect to. Another way to get the pin name is to first **click into the edit box**, then **click on the pin to connect to** and the name will automatically appear in the box.



Select and press the Delete Key

- Select a net or a gate and press the ECO **Delete** button or the **Del** key on the keyboard.
- The item deletion will be recorded in the ECO result file.



- If you make a mistake, you can use the **Undo ECO** button to go back to the previous change.

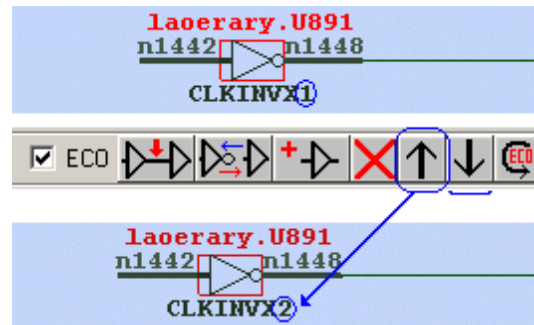
Undo ECO
Operation



4.4 Drive Strength Adjustment

The up and down arrow buttons on the ECO button bar are used to change the drive strength of the cell. To use this function on a gate, the library must contain multiple versions of the selected gate with different driver strengths.

- Select the driver by left clicking on it.
- Press the **Up arrow** button to increase the driver strength.
- Press the **Down arrow** button to decrease the driver strength.
- Notice that the part type for the driver changes as you press the Up and Down arrow buttons.



4.5 Logic Cone ECO

When a change involves many gates, sometimes it is easier to perform the ECO using scripts rather than using the graphical commands discussed earlier in this chapter. There are two methods for writing ECO scripts: Logic Cone ECO and GofCall. The GofCall API is covered in [Chapter 5: GofCall - netlist processing](#) and mimics most of the graphical ECO commands. Another type of script is a Logic Cone ECO script that changes the entire logic cone that drives an input pin and replaces it with a new set of logic from a newly re-synthesized netlist.

Write a Logic Cone Script file:

- Use a text editor to write the script file and save it with an ***.eco** file extension.

Below is an example of a Logic Cone script file called **eco_file_sc.eco**. Notice that no semicolons are needed at the end of the commands. The first line in the file is a comment line (indicated by the # prefixing the line). The script below will replace the logic cones to the D inputs of three registers.

```
#This is an ECO file for GOF
set_eco_number ec056789
```

```
current_design send_controller
replace_register state_reg_0_ -pin "D"
replace_register state_reg_1_ -pin "D"
replace_register state_reg_2_ -pin "D"
write_eco -f verilog neweco_sdc.v
```

Close Gof and execute the script using the command line:

Below is an example of how to launch GOF in batch mode to execute a logic cone ECO script.

```
gof -o sc_cone.log -lib technology.lib -imp netlist_under_eco.v
    -ref netlist_reference.v -ecofile logic_cone_eco.eco
    -nogui 0
```

Description of options passed to GOF:

-o *sc_cone.log*: Write ECO results to *sc_cone.log* file

-lib *technology.lib*: Loads the synthesis library file.

-imp *netlist_under_eco.v*: loads original netlist which needs some logic cones to be replaced.

-ref *netlist_reference.v*: Newly synthesized netlist with replacement logic cones.

-ecofile *logic_cone_eco.eco*: Load and run the ECO script. The script filename must have a file extension of **.eco*.

-nogui 0: No GUI is set to false to display the GUI windows while the ECO script commands are performed. By default, the GUI is disabled when an ECO script is executed.

In the above example, GOF will replace some logic in *netlist_under_eco.v* with new logic from *netlist_reference.v*. The logic cones replaced are defined by *logic_cone_eco.eco* script.

Available Commands for Logic Cone ECOs

Below is a list of commands that can be used in a **.eco* batch file. There should be one command on each line. No semicolons are needed at the end of each line.

set_eco_number *eco_prefix_name*

Sets the prefix for newly added cell instances or nets in the modified netlist file. Example:

```
set_eco_number eco_01234
```

current_design *module_name*

Defines the module instance containing the logic cone that will be replaced by **replace_register** commands. Example

```
current_design tx_controller
```

replace_register *instance_name* -pin *pin_name*

Tells GOF to replace the logic cone driving *instance_name*'s specified input pin. Example:

```
replace_register state_reg_2_ -pin "D"
```

write_eco -f verilog *new_netlist_name*

Write the current ECO result to a new netlist file (using Verilog netlist syntax). Do not use the same filename as the original netlist. Example:

```
write_eco -f verilog new_controller.v
```


4.6 Metal-Only ECOs

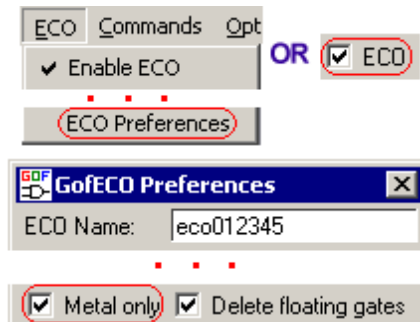
GOF has a metal-only mode for doing ECOs in which the only allowed changes are in the metal layers. In this mode, you can change the wiring and connect up existing spare gates, but you cannot add new gates that are not available on the spare gate list. When you add a spare gate to the netlist, GOF will find the closest spare, claim it, and remove it from the spare gate file. The spare gate list is shared across multiple ECOs to prevent the same spare gate from being used by two different ECOs.

When you commit the changes from an ECO, any spare gates used by the ECO will be commented out in the spare gates text file. If you need to redo an ECO, just reuse the same ECO Name when you start the new ECO attempt, and GOF will automatically recover the spare gates used during the previous attempt at the ECO.

If you plan to keep track of spare gates manually (e.g. by loading the spare gates instances on the schematic before performing any ECOs and then avoid adding or replacing any gates), you do not need to check the **Metal only** option.

Check Metal Only when enabling the ECO Mode:

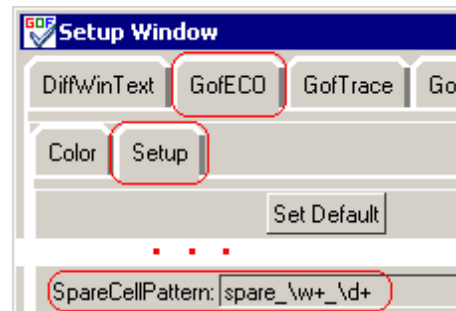
- Open the *GofECO Preferences* dialog either by:
 - Choosing the **ECO > ECO Preferences** menu,
 - Or if the menu is grayed out, check the **ECO** button on the button bar.
- Check the **Metal only** box, then close the dialog. This will limit your ECOs to metal-only changes.



Set up the Spare Gate naming pattern

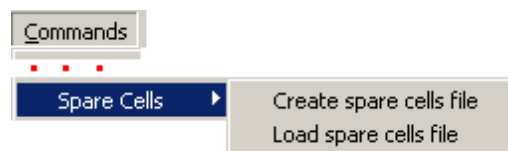
The metal-only flow requires spare gate names to match a regular expression specified in GofECO's Setup tab in GOF's *Setup Window*. Below are the steps to set this SpareCellPattern regular expression.

- Choose the **Options > Setup** menu to open the *Setup Window* dialog.
- Choose the **GofECO** tab, and the **Setup** tab.
- Type in the spare gate pattern into the **SpareCellPattern** box. For example, the regular expression shown in the picture, **spare_\w+_d+**, will match gates named spare_NANDX2_42 and spare_REGD42_2 in the netlist and identify them as spare gates.

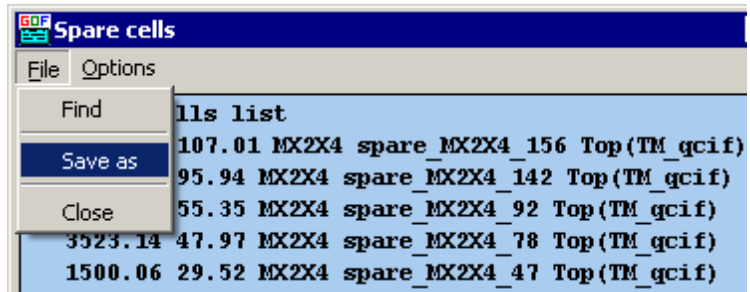


Create and Load the Spare Gate file.

- **Commands > Spare Cells > Create spare cells file** creates the spare cells file for this design (finds all cell instances that match the SpareCellPattern and puts them in this file).



- The spare gates file will open in a blue file window. Choose the **File > Save as** menu to save the file.



- Commands > Spare Cells > Load spare cells file** will load a spare cell file that was previously created during a GOF ECO. The Claim Spare Gates button on the ECO button bar does the same function as this menu.

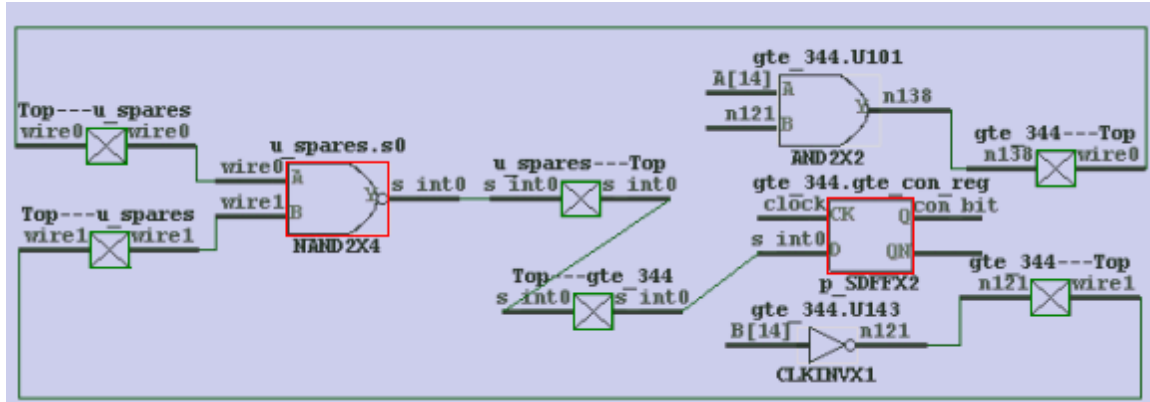


Example of Metal-Only ECO Features:

In the **SynaptiCAD > Examples > GOF_features** directory, the **case_metalonly.bat** file launches GOF with a design that has a spare gates list so that you can experiment with the features in this section.

Chapter 5: GofCall - netlist processing

GOF provides a programming interface, GofCall, that is useful for checking a design and performing substantial ECO changes on a design. GofCall scripts are executed by an embedded Perl interpreter which has been extended to provide commands for netlist processing. GofCall API commands can be entered into an Interactive Command Console, grouped together in a perl script, or typed into a shell version of GOF. For relatively simple ECO changes, you can use the graphical interface described in [Chapter 4: GofECO - schematic editor](#).



The GofCall API lets the user focus on a set of desired logic changes, without having to specify all the low-level details involved in making the changes (such as adding hierarchical ports or naming newly created nets). In the example above, a designer needed to change a flip-flop's D input pin so that the pin was a NAND of two other nets. However, the NAND gate was not available in the module that the flip-flop was in, so the gate needed to be borrowed from another hierarchical module (in this case, using the module that contains the spare gates). A total of 6 ports and several wires with multiple connections had to be created in the two sub-instances (gte344 and u_spares) to complete this ECO. However, this change was specified with only three lines of code in a GofCall script (the green lines in picture above show the new net connections added by GOF while performing this ECO):

```
change_pin("u_spares/s0/A", "gte_344/U101/Y");
change_pin("u_spares/s0/B", "gte_344/U143/Y");
change_pin("gte_344/gte_con_reg/D", "u_spares/s0/Y");
```

GofCall scripts can be used to nest ECO changes and add multiple gates in one line. All actions of a GofCall script are reversible. The resulting ECOs can be loaded into the schematic on the fly.

5.1 Interactive window & batch files

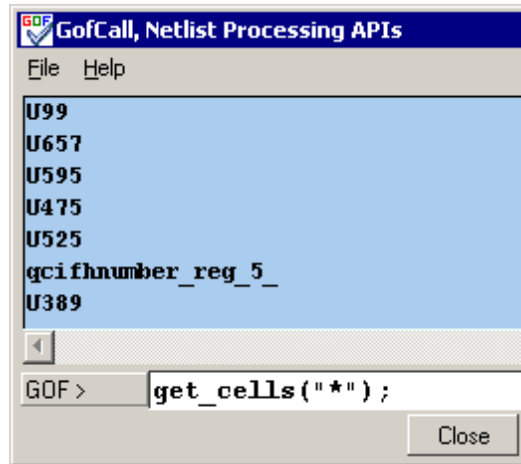
GofCall API commands can be entered individually into the GofCall Interactive window so you can check the result of each command as you enter it. The API commands can also be grouped together inside a *.pl script file and executed either through the GofCall Interactive window or through a command-line option to GOF at startup. See [Section 5.3: GofCall API List](#) for a complete list of GofCall commands.

Execute individual GofCall commands using the GofCall Interactive window:

- In GofViewer, choose the **Commands > Launch GofCall Script Interface** to open the GofCall Interactive window.



- In the **GOF >** edit box, enter GofCall commands and press the return key to execute them.
- For example, enter **get_cells("*");** and press return. This causes the instances in the current top level to be displayed in the GofCall log window.

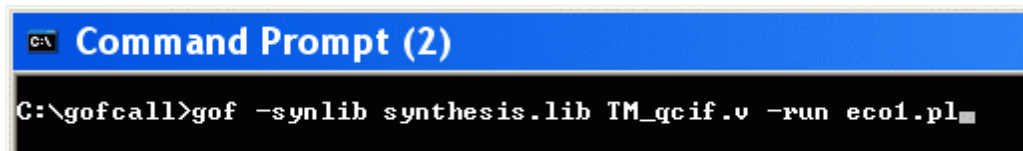


Execute Multiple GofCall Commands using a Script file:

- Use a text editor to create a Perl script file containing GofCall commands. The script filename must have a file extension of **.pl**. For example, the following 4-line script was saved to **eco1.pl**:

```
undo_eco;
set_top("TM_qcif");
change_pin("u_abc/u0/A", "AOI21X2", "u_abc/new_eco_inst", "frpls2qcif,beat4flg,Instaddr_e_3_");
write_verilog("TM.ecoed.v");
```

- The script file can be launched from the GofCall Interactive window using the **run** command.
- Scripts can also be launched from within the batch file that launches GOF using a **-run** command line option.



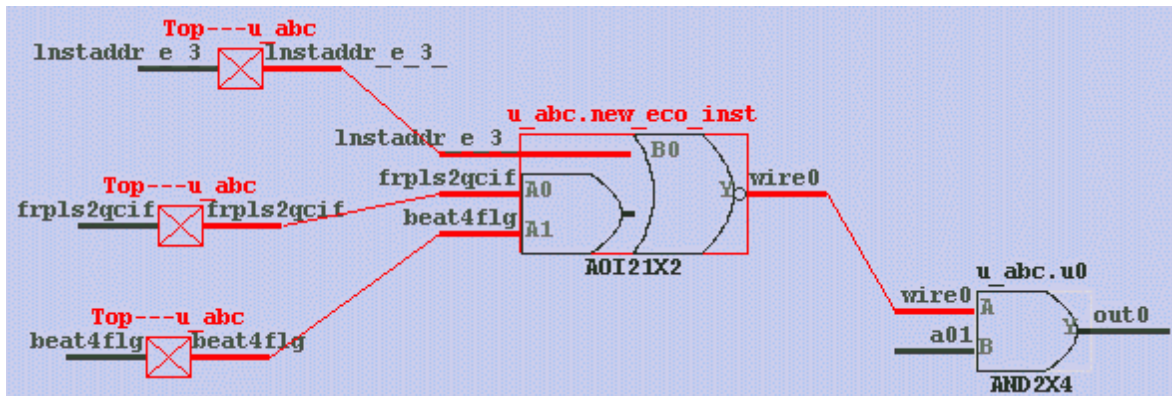
- When the script runs, GOF loads in the design and reports any errors or warnings if there are issues with the design. If there are errors, fix them and try again.

Verify ECO by viewing in the schematic:

- You can verify your ECO by viewing the affected leaf cells in the schematic window. One way to do this with GofCall is to use the **sch leafcell_name** command from the interactive

command window.

- In the below example, we used the command "**sch u_abc/u0**" to display one leaf cell in the schematic. Then we used *middle-mouse* click and drag operations on the pins of the cell to display the fanin and fanout of the cell.



- All the items marked in red were created by a single GofCall command: **change_pin("u_abc/u0/A", "AOI21X2", "u_abc/new_eco_inst", "frpls2qcif,beat4flg,Instaddr_e_3_");**

5.2 Command line text mode

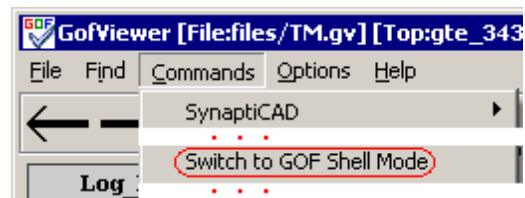
In addition to the GofCall Interactive window and batch scripts covered in the previous section, GofCall commands can also be entered from the command line when GOF is launched in the shell mode rather than the normal graphical mode.

Launching GOF in text mode:

- When launching GOF, if you put a **-shell** command in the batch file, GOF will open in text mode and will not open the GofViewer window.

```
C:\gofcall>gof -synlib synthesis.lib TM_qcif.v -shell
```

- OR, if you have already launched the graphical GOF, then choose the **Commands > Switch to GOF Shell Mode** menu. This will close the GofViewer and the GofTrace windows and enter the shell mode.



Entering command in the GOF in text mode:

- In text mode, GOF will load the design, report any errors, and then give you a **Gof >** command prompt.

```

C:\gofcall>gof -synlib synthesis.lib TM_qcif.v -shell
GOF starts @ Fri Aug 1 17:52:20 2008
Command Line: gof -synlib synthesis.lib TM_qcif.v -shell

Load synthesis library file synthesis.lib
Done loading synthesis.lib
Top tree creating...
Parsing TM_qcif.v
Loading TM_qcif.v...
ERROR! Leaf gate LATCKCX12 doesn't have pin SE
Topdesign is TM_qcif in Top tree
Total 575885 bytes netlist loaded
The design has total cell counter 9407.
GOF U2.400.02 is up @ Fri Aug 1 17:52:21 2008
GofCall, Netlist Processing APIs, Interactive Mode

Warning! There are errors in the design, run 'check_design' for detail
GOF > █

```

- The **Gof>** command prompt accepts any of the GofCall Api commands.
- To see a list of available commands, type **help**. To see the syntax and example usage for a specific command, type in **Help api_command_name** to get help for that API command!
- You can always start the graphical portion of GOF from the GOF shell mode using the command **start_gui**.

5.3 GofCall API List

This is a complete list of GofCall commands grouped by function. After this section is an alphabetical list of the commands and their syntax (in Windows help, shown in topics tree to the left instead).

Controlling GofCall

- [run\(\\$script_name\)](#): Run GofCall script
- [read_design\(\\$file_name, @options\)](#): Read in a Verilog design file
- [read_library\(\\$file_name\)](#): Read in a standard Verilog file
- [strict_syntax](#): Use strict syntax
- [set_quiet](#): Run script in quiet mode
- [set_verbose](#): Run script in verbose mode
- [set_max_lines\(\\$num\)](#): Set max output lines, default is 500 lines
- [set_keep_format\(\\$value\)](#): Set to true to keep format of the original netlist when ECO is done
- [gexit](#): Exit the GofCall interactive mode
- [gprint\(\\$info\)](#): Print a message

ECO Commands

- [setup_eco\(\\$eco_name, \\$library_name\)](#): Setup the ECO naming prefix and the technology library
- [set_inverter\(\\$inverter_name\)](#): specifies the inverter to use during subsequent ECOs
- [change_pin\(\\$pin_name, \\$net\)](#): Modify pin connection
- [change_pin\(\\$pin_name, \\$leaf_cell, \\$new_instance, \\$connection\)](#): Modify pin connection
- [change_port\(\\$port, \\$gate, \\$instance, \\$connections\)](#): Add a gate to a port

- [`change_gate\(\$instance, \$new_reference, \$pin_mapping\)`](#): Modify an instance
- [`change_gate\(\$instance, \$pin_connections\)`](#): Modify an instance
- [`change_net\(\$net, \$gate, \$instance, \$connections\)`](#): Change the driver to an existing net
- [`new_net\(\$new_net, \$reference, \$new_instance, \$connections\)`](#): Create a new net
- [`new_gate\(\$new_net, \$reference, \$new_instance, \$connections\)`](#): Create a new gate
- [`new_port\(\$name, \$direction\)`](#): Create a new port for the current top level module
- [`replace_logic_cone\(\$pin_list, @options\)`](#): Replace logic cone
- [`set_power\(\$leaf_cell, \$connections\)`](#): Set power pins connections for a leaf cell
- [`del_gate\(\$inst\)`](#): Delete a gate
- [`del_net\(\$net\)`](#): Delete a net
- [`del_port\(\$port\)`](#): Delete a port
- [`buffer\(\$net_names, \$buffer_name, \$fanout\)`](#): Add buffers to high fanout ECO nets
- [`undo_eco`](#): Undo an ECO
- [`write_verilog\(\$verilog_file\)`](#): Write out the changed netlist to a Verilog file
- [`write_tcl\(\$tcl_script_name\)`](#): Write out ECO result to Design Compiler tcl script format
- [`write_dcsh\(\$dc_script_name\)`](#): Write out ECO result to Design Compiler dcsh script format
- [`write_cdn\(\$soc_encounter_script_name\)`](#): Write ECO results to Cadence SOC Encounter format

Fixing Setup and Hold Timing Violations

- [`read_file`](#): Reads a static timing analyzer file into memory for the `fix_hold` and `fix_setup` commands
- [`fix_hold`](#): Inserts buffers to fix hold violations detected by static timing analyzers
- [`fix_setup`](#): Fixes setup violations detected by static timing analyzers by making paths faster.

Checking the ECO

- [`sch\(@instances_or_nets\)`](#): Launch schematic to verify ECO
- [`check_design`](#): Check if the design has unresolved modules or other errors

Top-level Module Commands

- [`set_top\(\$module\)`](#): Set the current top level module
- [`push_top\(\$module\)`](#): Set the top level module and push the previous setting to stack
- [`pop_top`](#): Pop out the saved top level module from the stack and discard current top level module setting

Getting information about different types of objects:

- [`get_conns\(\$net_or_pin\)`](#): Get connections of a net or pin in the top level module
- [`get_leaf_types\(\$leaf_cell_name\)`](#): Get leaf cell type
- [`get_drivers\(\$point\)`](#): Get the drivers of a net or pin
- [`get_net_of\(\$pin\)`](#): Get the name of the net that connects to a specified pin
- [`get_cell_info\(\$leaf_inst\)`](#): Get the cell connections for a leaf cell instance
- [`get_cells\(\$pattern\)`](#): Get all cells in \$path

- [`get_nets\(\$pattern\)`](#): Get nets that match the pattern
- [`get_def\(\$inst\)`](#): Get line number where instance is instantiated
- [`get_modules\(\$pattern\)`](#): Get all hierarchical modules in the top level module
- [`get_instances\(\$pattern\)`](#): Get all hierarchical instances in the top level module
- [`get_logic_cone\(\$net or pin_list, @options\)`](#): Get the logic cone of the listed pins
- [`get_resolved\(\$relative_path\)`](#)
- [`get_roots`](#): Get name(s) of the root module(s) of the design
- [`get_ref\(\$instance\)`](#): Get the leaf cell name or hierarchical module name of an instance
- [`get_ports\(\$direction, \$pattern\)`](#): Get all ports matching the direction in the top level module
- [`get_pins\(\$direction, \$instance\)`](#): Get pins of instance
- [`is_leaf\(\$name\)`](#): Check if a module or instance is a leaf cell
- [`is_seq\(\$leaf_name, @type_of_gate\)`](#): Check if the leaf cell is of the specified gate type
- [`exist_inst\(\$inst\)`](#): Check if the instance exists
- [`exist_wire\(\$wire\)`](#): Check if the wire exists
- [`get_lib_cells\(\$pattern\)`](#): List gates in libraries
- [`get_lib_cells\(\$pattern\)`](#): Get leaf gates in libraries
- [`get_leafs_count`](#): Get all leaf cell names and count in the top level module
- [`get_leaf_pin_dir\(\$pin, \$leaf_name\)`](#): Get leaf cell pin's direction input/output/inout

buffer

This is an ECO command that buffers high fanout ECO nets.

Usage:

```
buffer($net_names, $buffer_name, $fanout);
```

\$net_names: Net names to be buffered. Use "," to separate multiple nets, like "eco1_net1,reset2"

\$buffer_name: The buffer module name from library, leave it blank to automatically extracted.

\$fanout: How many fanout to insert a buffer.

Examples:

- 1) For every 10 fan outs of test_mode add a BUF6 buffer:

```
buffer("test_mode", "BUF6", 10);
```
- 2) For every 10 fanouts of 'clock', add repeaters, INV2,INV16

```
buffer("clock", "INV2,INV16", 10);
```
- 3) Let the tool pick a buffer

```
buffer("clock", "", 10);
```

change_gate

This is an ECO command that modifies a gate's type or pin connections depending on the arguments.

Usage:

```
change_gate($instance, $new_reference, $pin_mapping);
change_gate($instance, $pin_connections);
```

\$instance: The instance under ECO. Support hierarchical name, "u_abc/U123"

\$new_reference: The new reference which the instance will change to, E.G. 'AND3X1'. If no reference is present, the ECO operation is assumed to change the instance's pin connections.

\$pin_mapping: Input pins mapping, "new=>old" or ".new(old)" like "A1=>A,B1=>B", ".A1(A),.B1(B)" if two references have same input pins. The option can be empty

\$pin_connections: New pin connections, ".A(n242)", the unspecified pins will keep the original connection.

Examples where U123 is an OR3X1 with input pins, A,B,C:

```
#change U123 to an AND3X1, where all input pins are the same.
change_gate('U123', 'AND3X1', "");

#change U123 to an AND2X1, keep A and B connections, discard C
change_gate('U123', 'AND2X1', "");

#keep A, new B connects to old C's connections, discard old B
change_gate('U123', 'AND2X1', "B=>C");

# keep A B & C the same, and new D pin connects to net n123
change_gate('U123', 'AND4X1', "D=>n123");

#AO21X1 has input pins, A0, A1 and B0
change_gate('U123', 'AO21X1', "A0=>A,A1=>B,B0=>C");

# change U123 A to n123, B to n124, keep C connection
change_gate("U123", ".A(n123),.B(n124)");

# Rotating A/B/C connections
change_gate("U123", ".A(B),.B(C),.C(A)");

# Rotating A/B/C connections.
change_gate("U123", "A=>B,B=>C,C=>A");
```

change_net

This is an ECO command, that changes the driver of an existing net.

Usage:

```
change_net($net, $gate, $instance, $connections);
```

\$net: The net to be changed

\$gate: New leaf gate to drive the net

\$instance: The instance name of the new gate. If this is empty, GOF will assign a name.

\$connections: The new gate input pins connections. These can be specified using the following formats:

1. Very detailed: **.A(net0),.B(net1),.C(net2)**

2. Connect to the pins in alphabetical sequence (**net1,net0,net2**) indicating A->net1,B->net0,C->net2.
3. Other instance/pin "**U408/Y,U409/Y,net2**" indicating A->U408/Y,B->U409/Y,C->net2
4. Special character '-' is used to connect up the original connection

Examples:

- 1) Drive n123 with BUFX2 driven by n40

```
change_net("n123", "BUFX2", "", "n40");
```

- 2) Drive n123 with AND2X2 driven by n40 and original n123 driver

```
change_net("n123", "AND2X2", "", "-,n40");
```

- 3) Insert a buffer into n123

```
change_net("n123", "BUFX2");
```

change_pin

This is an ECO Command that modifies a pin connection. This command accepts two types of arguments, which GOF automatically detects the type by checking the reference of the second argument.

Usage 1: Connect pin to another net.

```
change_pin($pin_name, $net);
```

Usage 2: Insert a new leaf cell to drive the pin

```
change_pin($pin_name, $leaf_cell, $new_instance, $connection);
```

\$pin_name: In the format of "**instance/pin**", e.g. "U123/A". Multiple pins connecting to the same net can be listed and separated by a comma, e.g. "U123/A,U345/B". Hierarchical naming style is supported as well, "u_abc/U123/A".

\$net: The net name the pin will connect to. Hierarchical naming style is supported, "u_abc/net123". When the pin and the net are in different hierarchies, ports will be added automatically. See examples section a different hierarchy examples.

\$leaf_cell: The name of the leaf cell type that will drive the \$pin_name

\$new_instance: The instance name for the new inserted leaf cell. This is optional, and if blank Gof will assign a name. If use '.', the instance will be added to the same hierarchy as the \$pin_name.

\$connection: Describes how to connect the pins connection for the new cell. Supported formats:

1. Detail format: "**A(net0),.B(net1),.C(net2)**"
2. Simple format: Connect to the pins in alphabetical sequence "net1,net0,net2"
3. Mixed format: "u_abc/U123/Y,.B(net1),net2"
4. Special format: "-" in the pin connections, to connect up the original net.
5. Advanced nesting format: change_pin("U189/A", "AOI21X2", "", "U190/Y,,BUFX6(BUFX6(BUFX6(n412)))");

Note: All strings should be quoted by ' or " to avoid syntax error or undesired effects. If net or instance name has '\', single quotation has to be used. The best choice is to omit the '\' in the name, the tool can automatically decide if '\' is needed in the name.

Examples for **Usage 1** where U123 has input pins A,B,C, and U234 has input pins A0,A1,B:

```
# Change A pin of U123 to net12345
```

```

change_pin("U123/A", "net12345");

# Change B pin of U123 to $net which is defined in the ECO script.
change_pin("U123/B", $net);

# Change A pin of U123 and B pin of U234 to net12345
change_pin("U123/A,U234/B", "net12345");

#When the pin and the net are in different hierarchies,
#ports will be added automatically. The following will
#create 4 ports across the hierarchies to connect the net to the pin.
change_pin("u_abc/u_cde/U200/A", "u_xyz/u_stv/net300");

#The tool will get the net tie to Y pin of U300 and
#do the the same as the previous example.
change_pin("u_abc/u_cde/U200/A", "u_xyz/u_stv/U300/Y");

```

Examples for Usage 2 using same gates as above

```

# Insert "NAND2X2 eco12345_U0(.A(net1234),.B(net5678));"
# to drive U123/A
change_pin("U123/A", "NAND2X2", "eco12345_U0", "net1234,net5678");

# Same as above, with more detail of pin connections
change_pin("U123/A", "NAND2X2", "eco12345_U0",
           ".A(net1234),.B(net5678)");

#Insert a buffer to U123 A pin
change_pin("U123/A", "BUFFX4", "", "-");

#Insert NAND2X1 to drive CK pin and new A connects to the original net
change_pin("abc_reg_1_/CK", "NAND2X1", "", ".A(-),.B(1'b1)");

# Do hierarchical connection
change_pin("u_abc/u_cde/U200/A", "u_xyz/u_stv/U300/Y");

#Advance pin-connection mode
change_pin("qcif/num2/u_spare1/B", "AOI21X2", "eco_inst_on_top1",
           "NAND2X2(gte_344/u_smod/U100/Y, gte_344/n114),
           gte_343/U111, BUFX6(BUFX6(n105))");

```

change_port

This is an ECO command that changes an output port's driver, or adds a gate after an input port.

Usage 1:

```
change_port($port, $gate, $instance, $connections);
```

Usage 2:

```
change_port($port, $inst_pin);
```

\$port: The port that will be changed

\$gate: A new leaf gate that will become be attached to the port. If it is an output port then the gate will drive the port. If it is an input port, then gate will be added after the port.

\$instance: The instance name for the new leaf cell. If this parameter is left empty, then GOF will automatically assign a name.

\$inst_pin: In the format of 'u1234/Y', instance-name/pin-name

\$connections: The new gate's input pin connections. If this parameter is empty, the gate will be inserted in the connection. Supported formats:

1. Named association: ".A(net0),.B(net1),.C(net2)"
2. Connect nets to the pins in alphabetical sequence: "net1,net0,net2" indicating A->net1,B->net0,C->net2
3. Other instance/pin: "U408/Y,U409/Y,net2" indicating A->U408/Y,B->U409/Y,C->net2

Note: The difference of change_net and change_port command

change_net changes all drains of the net.

change_port changes only the port driver.

Examples: Add buffer to output port 'out1'

```
change_port("out1", "BUFX1", "eco_buf0", "-");
```

check_design

Check the design to see if it has unresolved modules or other errors, like floating multidrivers.

Usage:

```
check_design(@options);
```

@options:

-ignore list: Ignore the issues matching the list, E.G. 'FE_UNCONNECT*,SCAN_*'.

-eco: Only check instances/wires having been done ECO. Default check all instances/wires

-fixfile filename: Create ECO fix file

Examples:

```
check_design;
check_design('-ignore', 'FE_UNCONNECT*');
check_design('-ignore', 'FE_UNCONNECT*,SCAN_*');
check_design("-eco");
```

compare_nets

Check equivalence of two nets in the reference and implementation netlist

Usage:

```
my $result = compare_nets($net0, $net1, @options);
```

\$net0: The net in the reference netlist.

\$net1: The net in the implementation netlist.

@options:

-flatten: Work in the flatten, by default, the check is done in current module scope

\$result: If 1, they are equal, if 0, they are not equal.

Examples:

```
# Compare reg1/D in the reference and reg1/D in the implementation netlist
compare_nets("reg1/D", "reg1/D");
# Compare reg1/D in flatten mode
compare_nets("reg1/D", "reg1/D", "-flatten");
```

del_gate

This is an ECO command that deletes the specified gate.

Usage:

```
del_gate($inst);
```

\$inst: The instance name of the gate to be deleted.

del_net

This is an ECO command that deletes the specified net.

Usage:

```
del_net($net);
```

\$net: The net to be deleted.

del_port

This is an ECO command that deletes the specified port.

Usage:

```
del_port($port);
```

\$port: The port to be deleted.

exist_inst

Checks to see if the specified instance exists in the netlist.

Usage:

```
my $ret = exist_inst($inst);
```

\$inst: The instance name that will be checked

\$ret: If ret=1 then the instance exists, if ret=0 then the instance does not exist

exist_wire

Checks to see if the specified wire exists.

Usage:

```
my $ret = &exist_wire($wire);
```

\$wire: The wire name that will be checked

\$ret: If ret=1 then wire exists, if ret=0 then wire does not exist

fix_hold

Inserts buffer delays to fix hold violations detected by static timing analyzers like Accucore and Primetime. Timing violation report file has to be read by read_file API before calling this API.

Usage:

```
my $status = fix_hold(@options);
```

@options: list of comma separated pairs where the first term is the type of operation to perform and the second is the value to use during the operation.

"-margin", margin_value: If any timing path has a margin lower than this value in nanoseconds, then the gates in the data path will be upsized or faster gates inserted to meet the specified margin requirement. Defaults to a required margin of 0 when no value is given.

"-buffer", name_of_buffer: Name of buffer to be inserted

"-bufdelay", value: For AccuCore report file. Estimated buffer delay value in nanoseconds (1.0ns by default). For example, -bufdelay 0.9 indicates that each buffer inserted adds around 0.9ns of delay.

\$status the return value: If 1, then fix_setup was successful. If 0, then fix_setup failed.

For Example:

```
#1) read in the prime time report file
my $status = read_file("soc_primetype_hold.report", "-format", "pt");
# The goal is to make all paths have hold slack above 0.5ns
$status = fix_hold("-margin", 0.5, "-buffer", "BUFX10");
write_verilog("soc_hold_fix.v");

#2) Or for AccuCore timing report file
my $status = read_file("soc_accucore_hold.report", "-format", "accu");
$status = fix_hold("-margin", 0.5, "-buffer", "BUFX10", "-bufdelay", 0.9);
write_verilog("soc_hold_fix.v");
```

fix_logic

ECO command. Fix the implementation netlist logic by Reference Netlist

Usage:

```
fix_logic(@pin_port_list, @options);
```

@pin_port_list: List of the pins or ports whose logic will be fixed by the reference logic in reference netlist.

- The format is "sic_cnt_reg_0/D","sic_cnt_reg_1/D","bbr_ccd_reg[0]/D","out_port"
- The '\' should be dropped if the instance has '\' as prefix. For example 'bbr_ccd_reg[0]' has real name '\bbr_ccd_reg[0]' in the netlist

@options:

- help:** Print out this information
- keepfreed:** Keep freed instances, by default all output floating gates caused by ECO will be removed
- noopt:** No optimization on front side
- recover value:** The effort to recover removed gates, default 9, highest effort, valid from 0 to 9

Examples:

```
#1. Fix state_regs's D inputs
fix_logic("state_reg_0/D", "state_reg_1/D");
#2. Fix state_regs's D inputs and one output port
fix_logic("state_reg_0/D", "state_reg_1/D", "out_port");
#3. Fix state_regs's D inputs and one output port and don't delete floating gates
```

```
fix_logic("state_reg_0/D", "state_reg_1/D", "out_port", "-keepfreed");
```

fix_setup

This function fixes setup violations detected by static timing analyzers like Accucore and Primitime. The timing analyzer report file has to be read first by making a call to the **read_file** function before calling this function. Fix_setup speeds up violated data paths by swapping gates in the path for faster versions of the gates in the gate library. If this is not enough to correct the timing, it will next try to upsize the gates to meet the margin requirement.

Usage:

```
my $status = fix_setup("-margin", $value);
```

-margin: type of timing requirement to correct

\$value: The value of the margin required in nanoseconds. If any timing path has a margin lower than this value, then the gates in the data path will be upsized or faster gates inserted to meet the specified margin requirement. Defaults to a required margin of 0 when no value is given.

\$status the return value: If 1, then fix_setup was successful. If 0, then fix_setup failed.

Example:

```
my $status = read_file("soc_primitime_setup.report", "-format", "pt");
# The goal is to make all paths have setup slack above 0.2ns
$status = fix_setup("-margin", 0.2);
write_verilog("soc_setup_fix.v");
```

get_cell_info

This call returns a hash data structure containing the module name, instance name, and the connections for the specified cell.

Usage:

```
$hash = get_cell_info($module_or_inst, @options);
```

\$module_or_instance: the module or instance's name.

@options:

-help: Print out this information

-conns: Get Connections of the item, only when it's instance

-type: Get the item's type information. It can be 'ff','cg','latch','buf',

run 'get_lib_cells -type_info' for all existing type in the current libraries

An array will be returned if this option is present

-libname: Get the library name that the cell is in

-area: Get the area of the item

-size: Get the size of the item

-leakage: Get the leakage of the item

-ref. Same as 'get_ref instance' if the item property is instance

-context: Get detail library information

-attribute attribute_name: Check if the cell has the attribute set. 0 or 1 is returned

\$hash: Returned data in hash format. It has the following data structure:

```

my $module = $hash->{module};
my $instance: $hash->{instance};
foreach my $port (keys %{$hash->{connections}})
    { my $net = $hash->{connections}{$port}; }

```

If no option is present, it return the item's property:
 leaf_instance leaf_module hierachical_instance hierarchical_module

Examples:

```

my $area = get_cell_info("AND2X2", "-area");
my $is_iso = get_cell_info("ISOX2", "-attribute", "is_isolation_cell");

```

get_cells

Get all cells that match \$pattern

Usage:

```
my @cells = get_cells($pattern, @options);
```

\$pattern: The pattern matching instance name, '*', 'U*', 'U123' or '/UI_.*_./'

It can have path, 'u_clk/*', 'u_abc/u_def/*'

@options:

-help: Print out this information

-hier: Or -h, do the command hierarchically

-ref ref_pattern: Get cells that has reference matching ref_pattern, E.G. -ref OAI*

-type type_pattern: Type_pattern can be 'ff','latch','cg','hot' ...

run 'get_lib_cells -type_info' for all existing type in the current libraries

-type_match type_pattern: Get cells that have one of the types matches the type_pattern

-leaf: Only leaf cells

-verbose: To print out reference with instance

-dotpath: Path delimit is '.' instead of '/'

-nobackslash: Remove backslash

@cells: Reference of an array with all instances in "sub_module"

Examples:

1. Get all instances in the current module

```
get_cells('*');
```
- #2. Get all instances in the current module

```
get_cells();
```
- #3. Get all instances matching 'U234*' in the current module

```
get_cells('U234*');
```
- #4. Regular expression. Get all instances starting with U and followed by
 # two characters, like U10, U99

```
get_cells('/U../');
```
- #5. Get all instances matching *reg_*_ hierarchically

```
get_cells('*reg_*_', '-hier');
```
- #6. Get all instances hierarchically and having reference matching DFF*

```
get_cells('*', '-hier', '-ref', 'DFF*');
```
- #7. Get all instances in 'u_kb'

```
get_cells('u_kb/*');
```


get_conns

Get connections of net or pin in the top level module. The function returns the leafs and the hierarchical connections as a two dimensional array.

Usage:

```
@result = get_conns($net_or_pin, @options);
```

\$net_or_pin: The net name or pin name that needs to get connections.

@options:

-pin: Return inst/pin format

-driver: Return driver only

-load: Return load only

-count: Return connections count

@result: a two dimensional array where each element has the form of **[instance_name, port_name, pin_or_port, load_or_driver, is_it_a_leaf]**

@result = ([instance_0, pin_0, 'pin', 'load', 1], ...)

Example:

#1) n599 has three connections, U198 is the driver

```
@result = get_conns("n599");
```

```
#Returns @result = ( [gte_344 A[14] pin load 0],
#                    [U198 Y pin driver 1],
#                    [U94 AN pin load 1] )
```

#2) qcifhbeat has three connections, it is the output port
of the current top level module

```
@result = get_conns qcifhbeat
```

```
#Returns @result = ( [ qcifhbeat port load]
#                    [ U80 A pin load 1],
#                    [ qcifhbeat_reg Q pin driver 1]
```

#3) The argument is inst/pin format

```
GOF > get_conns U187/A
```

```
Executing get_conns("U187/A");
```

```
U294 A1 pin load 1
```

```
U187 A pin load 1
```

```
U80 Y pin driver 1
```

#4) Return connections count

```
get_conns("U187/A", "-count");
```

```
3
```

#5) Return load only and in pin format

```
get_conns("U187/A", "-pin", "-load");
```

```
U294/A1
```

```
U187/A
```

get_coord

Get an instance's coordination

Usage:

```
my ($x, $y) = get_coord($instance);
```

\$instance: Instance name

Examples:

```
my ($x, $y) = get_coord("xbar/U1234");
# $x=100, $y=200 in um
```

get_definition

Returns the line where the specified instance is instantiated.

Usage:

```
my $line = get_definition($inst);
```

\$inst: Instance name.

\$line: The instantiating line

Example

```
# returns "AND2X1 U78(.A(n1), .B(n2), .Z(n3));"
my $line = get_def('U78');
```

get_driver

Get the driver of a net or pin

Usage:

```
@driver = get_driver($point, @options);
```

\$point: net name or pin name, 'n12345' or 'U12345/A1'

@options:

-pin: Return in "inst/pin" format, E.G. "state_reg/Q"

Return an array if '-pin' is not present

@driver: The driver in array format, if '-pin' is not present.

If the point is floating, @driver is empty,

Index 0: instance, it will be empty if the driver is port

Index 1: pin or port name, if the driver is a port, it has the port name

Index 2: string "pin" or "port" depending on the driver is port or leaf cell

Note:

1. If '-pin' is present, return a scalar, \$driver = get_driver("n12345", "-pin");
2. Use 'get_drivers' if there are multiple drivers, the return data has different structure

Examples:

```
#1.
@driver = get_driver("net12345");
@driver has content of ("U1247", "Y", "pin");
```

```
#2. port_abc is input port
```

```
@driver = get_driver("port_abc");
@driver has content of ("", "port_abc", "port");

#3. Get pin format
$driver = get_driver("net12345", "-pin");
$driver has content of "U1247/Y"
```

get_drivers

Get the drivers of the net or pin and return them as an array.

Usage:

```
@drivers = get_drivers($point,@options);
$point: net name or pin name, 'n12345' or 'U12345/A1'
```

@options:

-nonbuf: Trace the drivers until none buffer

@drivers: An array of the drivers. If the net or pin is floating, then @drivers will be empty. If the net or pin has multiple drivers, then @drivers has more than one items. For each item in @drivers

Index 0: instance, it will be empty if the driver is port

Index 1: pin or port, if the driver is port, return port

Index 2: string "pin" or "port" depending on the driver is port or leaf cell

If 'nonbuf' is present, the last item in @drivers is the non-buffer driver

So '\$nonbuf = pop @drivers' can get the non-buffer driver

Note: Use 'get_driver' instead if the net has only one driver and 'nonbuf' option is not used

Examples:

```
#1.
@drivers = get_drivers("net12345");
@drivers has content of (["U1247", "Y", "pin"]);

#2. port_abc is input port
@drivers = get_drivers("port_abc");
@drivers has content of (["", "port_abc", "port"]);

#3. Buffers in the path
@drivers = get_drivers("state_reg/CK", "-nonbuf");
@drivers has content of
(
  ["buf_inst0", "Y", "pin"],
  ["buf_inst1", "Y", "pin"],
  ["and_inst2", "Y", "pin"]
)
```

get_instance

Get instance in the top level module

Usage:

```
my $instance = get_instance($pattern, @options);
```

\$pattern: Match pattern, can have wildcard "*", if it is empty, it will be treated as ""

@options:

-module: module name to have its instance name found

\$instance: Return the first instance matching

Examples:

```
# The first hierarchical instance matching 'ui_*'.
$instance = get_instance("ui_*");

# Find the instance name of module 'enet_control'
$instance = get_instance("-module", "enet_control");
```

get_instances

Get all hierarchical instances in the top level module

Usage:

```
my @instances = get_instances($pattern);
```

\$pattern: Match pattern, can have wildcard "*", if it is empty, it will be treated as ""

@instances: Array of the hierarchical instances

Examples:

```
# Any hierarchical instances with UI_ as prefix.
@instances = get_instances("UI_*");

# All hierarchical instances.
@instances = get_instances;
```

get_leaf_pin_dir

Get leaf cell pin's direction input/output/inout

Usage:

```
my $dir = &get_leaf_pin_dir($pin, $leaf_name);
```

\$pin: pin name, E.G. A or B or Y

\$leaf: Leaf cell name, E.G. NAND2X2

\$dir: return direction is either *input*, *output*, or *inout*.

Examples:

```
my $dir = get_leaf_pin_dir("NAND2X2/A");
```

get_leafs_count

Return a two dimensional array of each leaf cell and then number of that type contained in the top-level module.

Usage:

```
@leaf_count = get_leafs_count;
```

@leaf_count: Array of leaf name and count ([leaf0, cnt0], [leaf1, cnt1], ...)

Example

```
@leaf_count = get_leafs_count;
foreach my $leaf_point (@leaf_count)
{
    my $leaf_name = $leaf_point->[0];
    my $count = $leaf_point->[1];
    print "LEAF: $leaf_name has $count cells\n";
}
```

get_leaf_types

Get leaf cell type, return an array

Usage:

```
my @types = get_leaf_types($leaf_cell_name);
```

\$leaf_cell_name: Leaf cell name E.G. OAI32X2

@types: return an array, ('nand','or'), and ('ff' if it's flipflop then the -synlib technology_library option should be used in command line.

Examples of command line output:

```
GOF > get_leaf_types NAND2X2
Executing get_leaf_types("NAND2X2");
nand
```

```
GOF > get_leaf_types INVX2
Executing get_leaf_types("INVX2");
not
```

get_lib_cells

Get leaf gates in libraries

Usage:

```
my @cells = &get_lib_cells($pattern, options);
```

\$pattern: Library cell name pattern, can have '*'.

@cells: Return array with name matching

@options:

-help: This information

-char: All cells characterization

-type leaf_type: Get leaf gates matching type.

Leaf_type can be 'ff', 'latch', 'cg', 'buf', 'not', 'and' ...

-type_info: List all types in the current loaded libraries

-verbose: If \$pattern matchs only one lib cell, print the cell lib information

get_loads

Get loads of net or pin in the top level module, return the leafs connections

Usage:

```
@result = get_loads($net_or_pin, @options);
```

\$net_or_pin: The net name or pin name that needs to get fanouts.

@options:**-nonbuf:** Trace the loads until none buffer**-bypbuf:** Don't include buffer/inverter in the return array**@result:** a two dimension array

```

@result = ([instance_0, pin_0],
           [instance_1, pin_1],
           ...
          )

```

get_logic_cone

Get logic cone of nets or pins

Usage:

```
$result = get_logic_cone($net_or_pin_list, @options);
```

\$net_or_pin_list: the net or pin list should be in format of "U1/A,abc_reg/D,cde_reg/D"**@options:** Currently only one option, **-array**, which indicates the result value should be returned as an array of instances instead of as a string.**\$result:** a string in Verilog module format

Examples:

```
my $logic_cone = get_logic_cone("abc_reg/D,de_reg/D");
```

get_match_nets

Get logic equivalent nets in implementation netlist

Usage:

```
my @nets = get_match_nets($reference_net);
```

\$reference_net: Net name in reference netlist**@nets:** Equivalent nets in implementation netlist**get_modules**

Get all hierarchical modules in the top level module

Usage:

```
@modules = get_modules($pattern, @options);
```

\$pattern: Match pattern, can have wildcard "*", if it is empty, it will be treated as ""**@options:****-help:** Print out this information**-hier:** Or -h, do the command hierarchically**@modules:** Array ("module0", "module1", ...)

Examples:

```
# Any hierarchical modules with TM in the name.
@modules = get_modules("*TM*");

# All hierarchical modules
@modules = get_modules;

# All hierarchical modules and sub-modules under current module.
@modules = get_modules("-hier");
```

get_net_of

Get the name of the net that connects to a specified pin.

Usage:

```
my $net = get_net_of($pin);
    $pin: The pin of an instance (e.g 'U1234.A1' or 'U1234/A1')
    $net: Returns the net name connected to the specified pin.
```

get_nets

Get nets that match the pattern

Usage:

```
@nets = get_nets($pattern, @options);
    $pattern: naming pattern.
    @options:
        -const0: Get all constant zero nets
        -const1: Get all constant one nets
    @nets: returned net array.
```

Examples:

```
# get all nets.
@nets = get_nets("*");

# all nets with dbuffer as prefix
@nets = get_nets("dbuffer_*");

#Get constant nets
@nets = get_nets("-const");
```

get_path

Get current hierarchical path

Usage:

```
$path = get_path();
    $path: The current path
```

get_pins

Get pins of instance or module

Usage:

```
@pins = get_pins($name, @options);
```

\$name: The instance or module name, it can be hierarchical or leaf

@options: If no option is present, get all pins

-input: Get input pins

-output: Get output pins

-inout: Get inout pins

-clock: Get clock pin, only valid for sequential cell, flop latch and gated-clock-cell

-reset: Get reset pin, return "" if it doesn't exist

-set: Get set pin, return "" if it doesn't exist

-data: Get data pins

-attribute attribute: Get pins with the attribute

-nextstate_type type: Get pins matching the type

which can be 'data', 'load', 'scan_in', 'scan_enable'

This option is only valid for sequential cell, flop, latch and gated-clock-cell

@pins: All pins returned, in 'instance/pin' format

Examples:

```
my @pins = get_pins("-input", "u_abc/U123");
Result @pins = ("u_abc/U123/A", "u_abc/U123/B")
```

```
@pins = get_pins("AND2X2");
Result @pins = ("A", "B", "Y")
```

get_ports

Get all ports matching the direction in the top level module

Usage:

```
@matching_ports = get_ports($direction, @options);
```

\$direction: Port direction, "input", "output", "inout" or "". If it is empty, it will be treated as ""

@options: If no option is present, get all ports

-input: Get input ports only

-output: Get output ports only

-inout: Get inout ports only

-bus: Get ports in bus format instead of bit blast.

The API returns an array point if this option present

The item in the array has format of [port, IsBus, MaxIndex, MinIndex]

if IsBus == 1, MaxIndex is the Max Index of the bus, E.G, 7 if the bus is port_a
[7:0]

if ISBus==0, MaxIndex and MinIndex are not defined

@matching_ports: Return ports matching the pattern and the option specified in the current top level module

Examples:

```
# Get input ports with "dsp2mc_" as prefix
@ports = get_ports("input", "dsp2mc_*");
```

```
# Get all ports
@ports = get_ports;
```

get_ref

Returns the leaf cell name or hierarchical module name of the specified instance.

Usage:

```
$reference = get_ref($instance);
```

\$instance: Instance name, "U123"
\$reference: Return reference name, "NAND2X4"

get_resolved

Take the relative path name and resolve it into the module name and the leaf name.

Usage:

```
my ($module, $leaf) = get_resolved($relative_path);
```

\$relative_path: Relative path, like "u_abc/u_def/U456"
\$module: Resolved module name, like "def"
\$leaf: Resolved leaf name, like U456

Examples:

```
my ($module, leaf) = get_resolved("u_abc/u_def/U456");
# in this example $module has a value "def"
# and $leaf has a value "U456"
```

get_roots

Get the name of the root design.

Usage:

```
my @rootdesigns = get_roots;
```

@rootdesign: returned root designs name

get_spare_cells

ECO command. Get spare cells

Usage:

```
get_spare_cells($pattern,@options);
```

\$pattern: Spare leaf cell instance pattern, E.G. 'spare_inst*/spare_gate*'

Extract spare cells from the database with the pattern

The first half before '/' is hierarchical instance pattern, it is '*' for top level

The second half after '/' is leaf instance pattern

It is ignored if -file option is present

@options:

-o file_name: Write updated spare cell list to the file, by default, it has name 'spare_cells_scriptname.list'

-file spare_list_file: Load in spare cell list file instead of extracting from the database

Examples:

```
#1. Extract spare cells from the database, matching instances
#   like "SPARE_u0"
get_spare_cells("*/SPARE_*");

#2. Matching hierarchical instance "someSpare_*" and
#   leaf instance "spr_gate*"
get_spare_cells("someSpare_*/spr_gate*");

#3. Extract spare cells from file "spare_cells_list.txt"
get_spare_cells("-file", "spare_cells_list.txt");
```

gexit

Exit the GofCall interactive mode

Usage:

```
gexit;
```

gprint

Print out the message and save to log file

Usage:

```
gprint($info);
$info: message to be printed.
```

is_leaf

Check if a module or instance is leaf cell.

Usage:

```
my $leaf = is_leaf($name);
```

\$name: The module or instance under check

\$leaf: if 0, then it's a hierarchical module or the module is not defined. If 1, then it's a leaf cell like, NAND4X8.

is_seq

Checks whether the specified leaf module is a sequential gate of the type specified by the sequential gate option.

Usage:

```
my $isseq = is_seq($name, @options);
```

\$name: The leaf module under check

@options:

-ff: Check if it is a flipflop

-latch: Check if it is a latch

-cg: Check if it is a gated clock

-help: This information

\$isseq: function returns a 1 if the leaf module is the type of the specified option, and a 0 if it is not.

lv_search

Search instances in LayoutViewer window

Usage:

```
my $ret = lv_search($pattern, @options);
```

\$pattern: The pattern matching instance name, '*', 'u_spare*/U*'

@options:

-help: Print out this information

-spare: Unused spare cells only

map_spare_cells

ECO command. Map all new created cells to spare cells

Usage:

```
map_spare_cells;
```

@options:

-help: Print out this information.

-syn Synthesis-command-line:

By default, a built-in Synthesis Engine is used.

External Synthesis tool can be picked by this option

RTL Compiler and Design Compiler are supported.

E.G. "map_spare_cells('-syn', 'rc')" to pick RTL compiler

"map_spare_cells('-syn', 'dc_shell')" to pick Design Compiler

User can specify more values in the synthesis command

E.G. '-rc', "rc -E -use_lic RTL_Compiler_Physical"

-nofreed: Dont add freed gates for synthesis.

- nobuf:** Dont insert buffers/repeaters in long wires.
- notielow:** Dont tie low of the input pins of output floating gates, delete them instead
- pause:** Pause the tool before apply the patch
- exact:** Map to the exact name of spare cell, by default the tool picks up a spare cell with the same function, for example, pick up 'INVX2' for 'INVX4'

Note: A DEF file is needed for mapping to exact spare instances.

Examples:

- #1. Map to spare cells and use the built-in Synthesis Engine
`map_spare_cells;`
- #2. Use extra 'rc' option
`map_spare_cells('-syn', "rc -E -use_lic RTL_Compiler_Physical")`
- #3. Don't add freed cells for synthesis
`map_spare_cells('-syn', "rc -E -use_lic RTL_Compiler_Physical", "-nofreed")`

new_gate

This is an ECO command that creates a new gate.

Usage:

```
@return = new_gate($new_net, $reference, $new_instance, $connections);
```

Note:

- If the command is called in the context of return a scalar, the new created instance name will returned back.
- The usage is the same as [new_net](#), except **\$reference** has to be defined, and return back instance if scalar present.
- Run "help new_net" for detail in shell "GOF >"

new_net

This is an ECO command that creates a new net.

Usage:

```
@return = new_net($new_net, $reference, $new_instance, $connections);
```

\$new_net: The new net to be created, if not defined, the tool will assign one automatically

\$reference: The leaf gate name to drive the net.

\$new_instance: The instance name the new cell will have. Automatically assigned a name if left empty.

\$connections: The new gate input pins connections Supported formats:

1. Detail format: ".A(net0),.B(net1),.C(net2)"
2. Simple format: Connect to the pins in alphabetical sequence "net1,net0,net2" indicating A=>net1,B=>net0,C=>net2
3. Mixed format: "instance/pin" and net, "U408/Y,U409/Y,net2" indicating A=>U408/

Y,B=>U409/Y,C=>net2

4. The "instance/pin" can have sub-instance hierarchy, "u_abc/U408/Y"

@return: is a two element array, where \$return[0] = is the new instance, and \$return[1] is the name of the new net.

Note: Hierarchical path is supported in any net or instance in the command, for instance, new_net('u_abc/net124', ...). If the command is called in the context of return a scalar, the new created net name will returned back. The new net is assumed to be driven in the path it is created, for instance, new_net('u_abc/eco12345_net124'); eco12345_net124 should be driven in sub-instance u_abc after it is created.

Examples:

```
# NAND2x2 instance name 'U_eco_123' driving new net 'net123'
new_net("net123", "NAND2X2", "U_eco_123", ".A(n200),.B(n201)");

# IN VX2 with instance name 'U_inv' is created in u_abc sub-instance
# and the input pin of the invert is driven by n200 in current top level
# port would be created in n200 doesn't have connection to u_abc
new_net("u_abc/net123", "INVX2", "u_abc/U_inv", "n200");

# Create a new net "net500"
new_net("net500");
new_net("n200");    # Error, n200 already exists

# Create a new instance with new net tied to output pin,
# and the input pin is floating.
@return = new_net("", "INVX2", "", "");
# $return[0] is new created instance,
# $return[1] is new created net.
```

new_port

This is an ECO command that creates a new port for the current top level module.

Usage:

```
new_port($name, @options);
```

\$name: This is name of the new port. The port name has to be pure words or with bus bit, like, abc[0], abc[1]

@options:

- input:** New an input port
- output:** New an output port
- inout:** New an inout port

Examples:

```
# create an input port naming 'prop_control_en'
new_port('prop_control_en', 'input');

# create an output port with bus bit 'prop_state[2]'
new_port('prop_state[2]', 'output');
```

```
# create an output port with bus bit 'prop_state[3]'
new_port('prop_state[3]', 'output');
```

place_gate

ECO command. Place gate position

Usage:

```
place_gate($inst, $x, $y);
```

\$inst: The instance to be placed

\$x,\$y: The coordinate

Note: This command affects the spare gate mapping of the instance.

Examples:

```
# A flop is added and placed in some location
# In 'map_sparee_cells' command, the flop is mapped to a
# spare flop closest to the location
change_pin("U123/A", "DFFX1", "eco_dff_reg", ".D(-),.CK(clock)");
place_gate("eco_dff_reg", 100, 200); # location, 100um, 200um
map_spare_cells;
```

place_port

ECO command. Place port position

Usage:

```
place_port($port, $x, $y);
```

\$port: The port to be placed

\$x,\$y: The coordinate

Note: This command has effect on change_port ECO command

pop_top

Pop out the saved top level module from the stack and discard current top level module setting.

Usage:

```
pop_top;
```

push_top

Set the top level module and push the previous setting to stack, pop_top can retrieve it

Usage:

```
push_top($module);
```

\$module: Set the *\$module* as the top level module, push the previous setting to the stack.

read_def

Read DEF file

Usage:

```
my $status = read_def(@files);
```

@files: DEF files

\$status: If zero, the files are read in successfully
if one, failed to read in the files

Examples:

```
# Read in soc_top.def
my $status = read_def("soc_top.def");

# Multiple DEF files
my $status = read_def("soc_top1.def", "soc_top2.def");
```

read_design

Read in a Verilog netlist file

Usage:

```
my $status = read_design($file_name, @options);
```

\$file_name: Verilog netlist name

@options: If no -imp or -ref option is provided, the netlist is assumed 'implementation'

-imp: The netlists are implementation design

-ref: The netlists are reference design

-Top_1: Read design to create Top_1 tree database

-Top_2: Read design to create Top_2 tree database

-once: Read the file only once, the file will be bypassed if the command is executed again

\$status: If zero, the file is read in successfully. If one, then it failed to read in the file

Example:

```
my $status = read_design("-imp", "soc_design_resynthesized.gv");
my $status = read_design("-ref", "soc_design_released.gv");
my $status = read_design("-ref", "soc_design_released.gv", "soc_io.gv");
```

read_file

Reads a file of the specified type into memory.

Usage:

```
my $status = read_file($file_name, @options);
```

\$file_name: name of file to load

@options: a list of optional values, specified in pairs. Currently only one option is supported:
"-format". This option specifies the file format of the file to be loaded. Two file formats are currently supported:

"accu" - Accucore timing report file.

"pt" - Prime Time report file which should generated by the following command and options:

```
report_timing -nosplit -path full -delay max/min -input_pins \
-nets -nworst 5 -max_paths 10000 -transition_time -capacitance
```

Return value: If 1, the file was successfully read in. If 0, then reading the file failed.

Usage in a script:

```
read_file("prime_time.report", "-format", "pt");
```

Usage in a command shell:

```
"read_file prime_time.report -format pt"
```

read_library

Read standard library

Usage:

```
my $status = read_library($file_name, @options);
```

\$file_name: Standard library file name, having .lib extension

@options:

-v: Treat the @files as verilog library files

-lib: Treat the @files as standard library files

-vmacro: Treat the @files as macro library files which will be used as macro cell in ECO

\$status: If zero, the file is read in successfully. If one, then failed to read in the file

Example:

```
my $status = read_library("arm_40_hvt.lib", "arm_40_io.lib");
my $status = read_library("analog_stub.v", "analog_stub2.vlib");
my $status = read_library("-v", "analog_stub.gv");
my $status = read_library("-vmacro", "macrocell.v");
```

rename_net

ECO command. Rename a net name

Usage:

```
rename_net($oldname, $newname);
```

\$oldname: Old net name

\$newname: New net name

report_eco

Report ECO

Usage:

```
report_eco($filename);
```

If \$filename is not present, print to screen

report_spares

Report Spare cells

Usage:

```
report_spares;
```


replace_logic_cone

Replace logic cone specified in the list of register/pins with the netlist contained in the reference netlist specified by the **read_design** call. Before calling **replace_logic_cone**, you must call **read_design** once with the original netlist and the **-golden** option, and once with the name of the new netlist to be generated using the **-revised** option.

Usage:

```
replace_logic_cone($pin_list, @options);
```

\$pin_list: an ordered list register/pin pairs. Each pair is separated by a comma. Inside the pair the register name and pin name are separated by a forward slash. For example, "reg_0/D,reg_1/D,ccd_reg[0]/D" is a list of three pairs. If a register instance has a backwards slash, '\', as a prefix, then the slash should be dropped (e.g use *bbr_ccd_reg[0]* even though the netlist name is *bbr_ccd_reg[0]*).

@options: **-keepfreed** keeps freed instances, instead of automatically removing any floating gates caused by the ECO. Also a **-help** to print out this information.

Example:

```
undo_eco; # Undo all eco done by previous run
setup_eco("eco_rep_logic_cone");
# the above prefix will appear on all new instances and nets
set_inverter("INVX2");
my $status = read_design("-golden", "soc_design_resynthesized.gv");
$status = read_design("-revised", "soc_design_released.gv");
set_top("codec_top");
replace_logic_cone("sic_cnt_reg_0/D,sic_cnt_reg_1/D,bbr_ccd_reg[0]/D");
write_verilog("soc_eco.gv");
```

run

Runs a GofCall script.

Usage:

```
run($script_name);
```

Example:

```
run("eco2.pl");
```

sch

Launch schematic to verify ECO

Usage:

```
sch(@instances_or_nets, @options);
```

@instances_or_nets: Instances or nets to be displayed on the schematic

@options:

-set value: Set a value when launch the schematic

-to value: To existing schematic

Examples:

```
sch("U123", "U456", "inst0");
sch("clk")
sch("in1", "-set", "1");
sch("in1", "-to", "1"); # No action if schematic 1 doesn't exist
```

set_buffer_distance

Set distance limit for inserting buffer

Usage:

```
set_buffer_distance($distance_val);  
    $distance_val: distance to insert buffer, in um
```

set_exit_on_error

Whether the tool should exit when the script runs into an error

Usage:

```
set_exit_on_error($error, $bit);  
    $error: Error pattern, wild card support. 'E-001', 'E-'  
    $bit: 1, Exit on the error, default  
          0, Don't exit on the error
```

set_exit_on_warning

Whether the tool should exit when the script runs into a warning

Usage:

```
set_exit_on_warning($warning, $bit);  
    $warning: Warning pattern, wild card support. 'W-001', 'W-'  
    $bit: 1, Exit on the warning  
          0, Don't exit on the warning, default
```

set_invert

This command specifies the inverter to use during subsequent ECOs. If an inverter is not set, then GOF will automatically pick one.

Usage:

```
set_invert($inverter);  
    $inverter: Library cell name for inverter
```

Example:

```
set_invert("INVX2");
```

set_keep_format

When set this keeps the original Verilog when the ECO is done.

Usage:

```
set_keep_format($value);  
    $value: If 0, then disable. If 1, then enable.
```

set_leaf

Set a hierarchical module to be leaf. Useful to stub hierarchical instances.

Usage:

```
set_leaf($module_name, $value);
```

\$module_name: The module to be set leaf or not set to leaf

\$value: 1 or larger than 1, set the module as leaf. 0 not set to leaf.

If \$value is not present, the default value is 1.

Examples:

```
set_leaf($module_a); # set $module_a as a leaf
set_leaf($module_a, 1); # same as the above
set_leaf($module_a, 0); # remove the leaf setting
```

set_log_file

Set log file name

Usage:

```
set_log_file($filename);
```

\$filename: Log file name

set_max_lines

Set max output lines.

Usage:

```
set_max_lines($num);
```

\$num: New max lines number. The default is 500.

set_max_loop

Setup max loop, GOF stops logic optimization when max loop number is reached

Usage:

```
set_max_loop($value);
```

\$value: Setup BDD threshold, default 40000

set_mod2mod

Set reference module mapping to implementation module

Usage:

```
set_mod2mod($refmod, $impmod);
```

\$refmod: The reference module name

\$impmod: The implementation module name

Note: The command is used when reference netlist is partial

set_mu

MU configuration, setup MU value for BDD threshold

Usage:

```
set_mu($value);
```

\$value: Setup BDD threshold, default 12000

set_power

Set power pin connections for the leaf cell.

Usage:

```
set_power($leaf_cell, $connections);
```

\$leaf_cell: Leaf cell name. Like NAND2X4

\$connections: Power pins connections, like ".GND(GND),.VDD(VDD)"

set_preserve

Set preserve property on instances. The tool will not remove them in ECO

Usage:

```
set_preserve(@instances);
```

@instances: Instances to be preserved in the current module

Accept wild card '*'

Examples:

```
set_preserve("u_donttouch0", "u_1000");  
set_preserve("DONT*");
```

set_quiet

Run script in quiet mode

Usage:

```
set_quiet;
```

set_tiehi_net

Set tiehi net name, it will be used if tiehi net is needed in ECO

Usage:

```
set_tiehi_net($netname);
```

\$netname: Tiehi net name, E.G. '___logic1___'

set_tielo_net

Set tielo net name, it will be used if tielo net is needed in ECO

Usage:

```
set_tielo_net($netname);
```

\$netname: Tielo net name,

Examples:

```
set_tielo_net("___logic0___");
```

```
set_tielo_net("TIE_HILO_TIELO_NET");
```

set_top

Set the current top level module

Usage:

```
set_top($module);
```

\$module: Set *\$module* as the current top of the design. If the argument is missing, return the current setting *".."* set to parent module

Note: It can be reset to the root top module by 'undo_eco'

set_tree

Set the current tree, if there are more than one sets of databases

Usage:

```
set_tree($tree);
```

\$tree: It can be the default tree 'Top'.

Or 'Top_1' if you use -Top_1 option to load in other design

Or Top_ref in when using read_design("-ref", reference_netlist)

If \$tree is not defined, the current database name is returned

Note:

- Implementation tree 'Top' has aliases of 'imp', 'IMP'
- Reference tree 'Top_ref' has aliases of 'ref', 'REF'

Examples:

```
set_tree("Top");
```

```
set_tree("IMP"); # Same as the above
```

```
set_tree("Top_ref"); # Set to reference tree
```

```
set_tree("ref"); # Same as the above, set to reference tree
```

```
set_tree(); # Return the current database name. E.G. 'Top_ref'
```

set_verbose

Run script in verbose mode.

Usage:

```
set_verbose;
```

setup_eco

This is an ECO command, that sets up the preferences for the ECO.

Usage:

```
setup_eco($eco_name, @options);
```

\$eco_name: The ECO name that will be used as a prefix to name new gates and nets like: eco01234.

@options:

-help: Print out this information.

-comments comments: Comments to appear at the beginning of ECO netlist.

Examples:

```
#1. Setup ECO name
setup_eco('eco1234')
#2. Setup ECO name with comments
setup_eco('eco1234', '-comments', 'Fix abc_state state machine');
```

start_gui

Start GUI windows

Usage:

```
start_gui;
```

stich_scan_chain

ECO command. Stitch scan chain

Usage:

```
stich_scan_chain($inst);
```

\$inst: The instance name to stitch in all new flops scan chain. Create a new sub-scan-chain for the all new flops, break the chain connecting the instance, and stitch in the new sub-scan-chain.

strict_syntax

Use strict syntax, has same effect as 'use strict;' in perl script.

Usage:

```
strict_syntax;
```

undo_eco

This is an ECO command, that undos the eco and restores the database.

Usage:

```
undo_eco;
```

write_dcsh

This is an ECO command that writes the ECO results to Design Compiler dcsh script format.

Usage:

```
write_dcsh($dc_script_name);
```

\$dc_script_name: Synopsys Design Compiler dcsh script name.

Examples:

```
write_dcsh("eco12345.dcsh");
```

write_soc

This is an ECO command that writes the ECO results to a Cadence SOC Encounter script format.

Usage:

```
write_soc($soc_encounter_script_name);
```

\$soc_encounter_script_name: Cadence SOC Encounter script name.

Examples:

```
write_soc("eco12345.soc");
```

write_spare_file

ECO command. Write out spare cells list file

Usage:

```
write_spare_file($filename);
```

\$filename: Spare cells file name to be written out

Note: Any used spare cell has '#' in the start of the line

write_tcl

This is an ECO command that writes the ECO changes to Design Compiler tcl script format.

Usage:

```
write_tcl($tcl_script_name);
```

\$tcl_script_name: Synopsys Design Compiler tcl script name.

Examples:

```
write_tcl("eco12345.tcl");
```

write_verilog

This is an ECO command that writes the changed netlist to a file to a Verilog file.

Usage:

```
write_verilog($verilog_file, @options);
```

\$verilog_file: Name of the file to write to. It should be different from any existing netlist file name.

@options:

-help: Print out this information

-all: Keep the modules in the netlist file even they are not the sub-modules of the top module

\$verilog_file: Write out verilog file name, should be different from existing netlist file name.

Note: When the design is read in by multiple netlist files, set_top command should be used to make the correct file saved out.

Examples:

```
#1. The design is read in by 'gof -lib tsmc.lib ethernet_top.v'
```

```
# After ECO, to write out ECO netlist use command
```

```
write_verilog("ethernet_top_eco.v");
```

```
#2. The design is read in by multiple netlist files,
```

```
# 'gof -lib tsmc.lib mem_control.v dsp.v ethernet_top.v'
```

```
# The ECO is done on 'mem_control' module, to save the netlist
```

```

set_top("mem_control");
write_verilog("mem_control_eco.v");

#3. The design is read in by 'gof -lib tsmc.lib ethernet_top.v',
#   ethernet_top.v has 'mem_control' and 'dsp' sub-modules
#   The following commands will only write out 'mem_control' and
#   its sub-modules
set_top("mem_control");
write_verilog("ethernet_top_eco.v");

#4. The design is read in by 'gof -lib tsmc.lib ethernet_top.v'.
#   ethernet_top.v has 'mem_control' and 'dsp' sub-modules
#   The following commands will write out all modules in ethernet_top.v
set_top("mem_control");
write_verilog("ethernet_top_eco.v", "-all");

#5. The design is read in by 'gof -lib tsmc.lib ethernet_top.v'.
#   ethernet_top.v has 'mem_control' and 'dsp' sub-modules
#   The following commands will write out all modules in ethernet_top.v
set_top("ethernet_top");
write_verilog("ethernet_top_eco.v");

```

5.4 Examples of GofCall Scripts

Below are a few examples of how GofCall scripts can be used.

1) Metal-only ECO using a Spare Gate

Perform a metal-only ECO to use an AND gate from the spare gates list with a two line script (A TCL script to do the equivalent would be 20 lines long). The AND spare gate is in sub-hierarchy "u_spares", instance "S2". Two ports have to be created to connect two signals "n242" and "n243" to the AND gate. Another port has to be created to connect back the AND gate's output and tie it to U187's A pin.

```

change_gate("u_spares/s2", "n242,n243");
change_pin("U187/A", "u_spares/s2/Y");

```

2) Switch a NAND gate's three inputs.

```

change_get("U188", "A=>B,B=>C,C=>A");

```

3) Disable 128 bit data bus by "disable_bus" signal

```

change_net("abc_bus[$_]", "AND2X2", "",
           "-", disable_bus) foreach (0..127);
buffer("disable_bus", "BUFX6", 16); # automatically buffer disable_bus

```

4) Use the power of Perl to add a clamp gate to specially named ports.

```

my @ports = get_ports("output", "*");
my @clampgates = get_cells("AND_clamped_*");
foreach my $port (@ports)
{
    if($port =~ m/clamp_\d+/)
    {

```



```

my @drivers = get_drivers($port);
my $drive_inst = $drivers[0][0];
if($drive_inst !~ m/AND_clamped_\d+/)
{ # missing clamp
my $num = 0;
my $clamp_instance = "AND_clamped_$num";
while(grep($clamp_instance eq $_, @clampgates))
{
$num++;
$clamp_instance = "AND_clamped_$num";
}
change_net($port, "AND2X2", $clamp_instance, "-", clamp_signal);
}
}
}

```

5) Change a flip-flop's D input.

Below is a script that adds 3 gates, one port, 4 new wires, 13 connections, and 1 disconnection, and it only takes 4 lines of code. If this were written in TCL format, it would be 20 lines long.

```

undo_eco;
set_top("TM_qciflt");
setup_eco("eco012345");
change_pin("frplx_reg/D", "NAND3X4", "",
    "-",
    MXI2X2(qciflsb3b_e,gte_344/gte_con_reg/Q,e_bonn),
    BUF6(qcif_s0_good)");

```

- The first argument is the instance pin, "frplx_reg/D"
- The second argument is the leaf cell "NAND3X4" to drive the D input pin.
- The third argument is the instance for the leaf cell, it's empty "", letting the tool pick one instance name.
- The fourth argument is the pin-connections. NAND3X4 has three input pins (A,B,C), so the pin-connections have three parts. Separated by " , "
 - Part 1 is "-", which means the original D input wire will be used to connection to A pin of NAND3X4. Read GofCall APIs for more detail.
 - Part 2 is "MXI2X2(qciflsb3b_e,gte_344/gte_con_reg/Q,e_bonn)". A new gate MXI2X2 is inserted to drive B pin of NAND3X4, the input pins of MXI2X2 is driven by three signals "qciflsb3b_e", "gte_344/gte_con_reg/Q" and "e_bon". Out of these three signals, "gte_344/gte_con_reg/Q" is a hierarchical signal which needs a port to pull it to the current hierarchy and the port will be automatically created.
 - Part 3 is "BUF6(qcif_s0_good)". Wire "qcif_s0_good" is buffered to driver C pin of NAND3X4.

The nested long line can be separated into three small operations, but more manual intervenes are needed to specify the new net names or instance names.

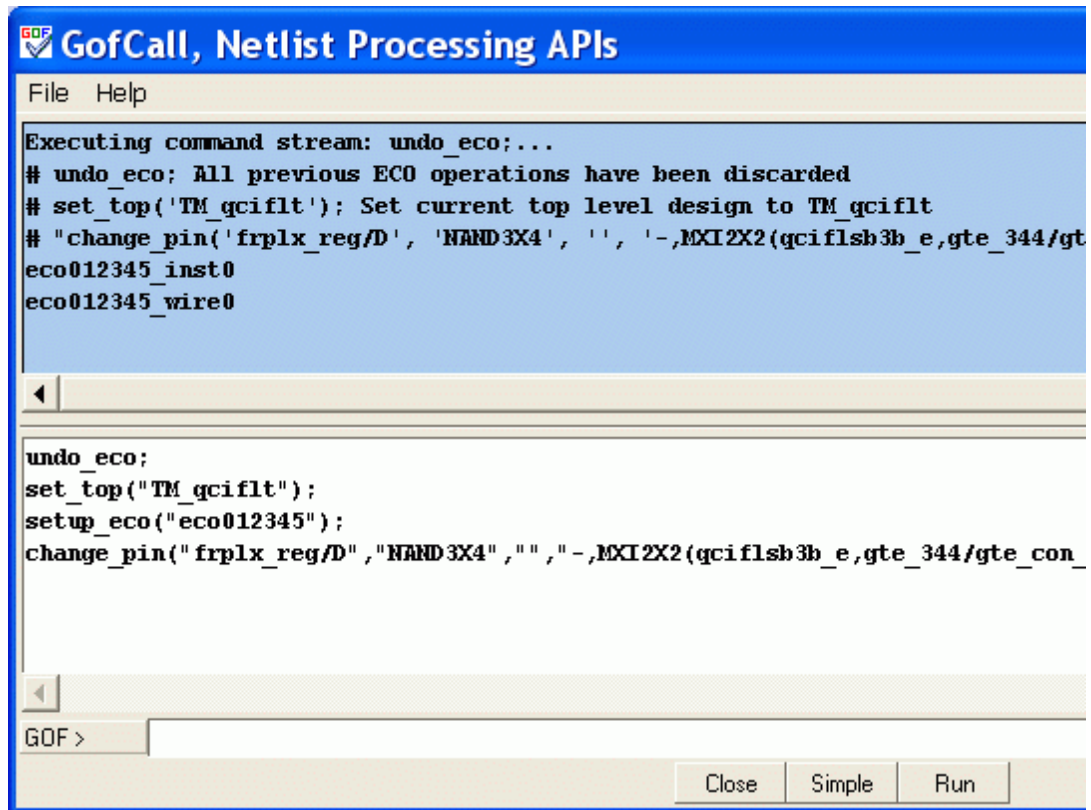
```

new_net("eco012345_mx_out", MXI2X2, "", "qciflsb3b_e,gte_344/gte_con_reg/Q,e_bonn");
new_net("eco012345_buf_qcif_s0_good", "BUF6", "", "-");
change_pin("frplx_reg/D", "NAND3X4", "", "-", eco012345_mx_out, eco012345_buf_qcif_s0_good);

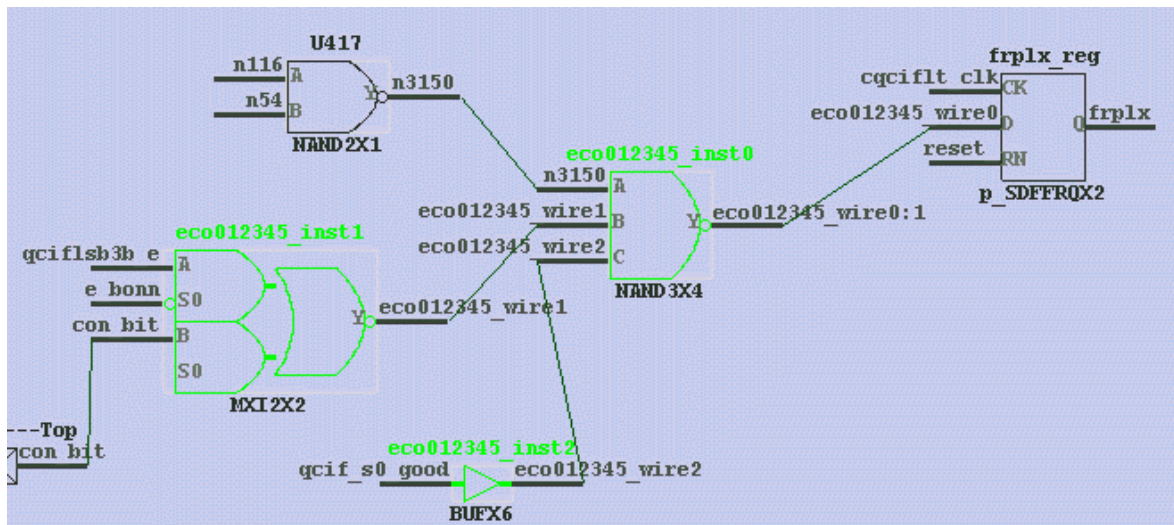
```

To run the script, you can use GofCall Graphical mode. In GofViewer, Menu Actions->"GofCall, APIs interactive" to get GofCall window. Press "Advance" button to run in code snip. Or save to a file, key

in "run_the_script_file" in "GOF > " shell entry to run in batch mode.



Key in "sch eco012345_inst0" in "GOF > " shell entry to launch schematic to verify the ECO on the fly.



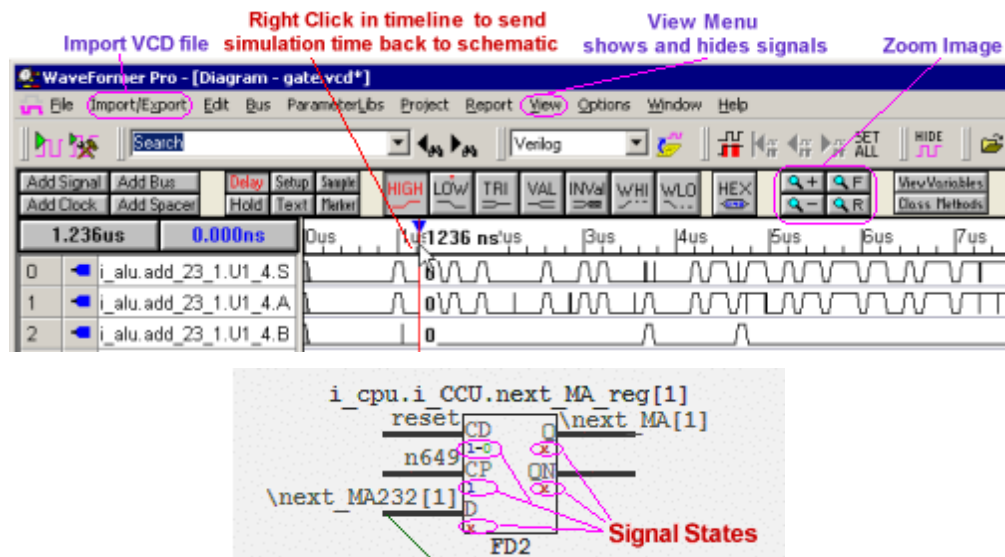
After ECO, three new gates, "eco012345_inst0", "eco012345_inst1" and "eco012345_inst2" are created. One port is added to connect "con_bit" from sub-instance "gte_344". Four new wires are created "eco012345_wire0", "eco012345_wire1", "eco012345_wire2" and "con_bit". 13 pin connections are added. And D input of "frplx_reg" is disconnected and reconnected. If converted to Design Compiler TCL script format, there are more than 20 lines!

```
current_design gte
```

```
create_port -direction out con_bit
connect_net [get_nets con_bit] [get_ports con_bit]
current_design TM_qciflt
create_net eco012345_wire0
create_net eco012345_wire1
create_net con_bit
create_net eco012345_wire2
create_cell eco012345_inst0 tmc65nm/NAND3X4
connect_net [get_nets n3150] [get_pins eco012345_inst0/A]
connect_net [get_nets eco012345_wire1] [get_pins eco012345_inst0/B]
connect_net [get_nets eco012345_wire2] [get_pins eco012345_inst0/C]
connect_net [get_nets eco012345_wire0] [get_pins eco012345_inst0/Y]
create_cell eco012345_inst1 tmc65nm/MXI2X2
connect_net [get_nets qciflsb3b_e] [get_pins eco012345_inst1/A]
connect_net [get_nets con_bit] [get_pins eco012345_inst1/B]
connect_net [get_nets e_bonn] [get_pins eco012345_inst1/S0]
connect_net [get_nets eco012345_wire1] [get_pins eco012345_inst1/Y]
create_cell eco012345_inst2 tmc65nm/BUFX6
connect_net [get_nets qcif_s0_good] [get_pins eco012345_inst2/A]
connect_net [get_nets eco012345_wire2] [get_pins eco012345_inst2/Y]
disconnect_net [get_nets n3150] [get_pins frplx_reg/D]
connect_net [get_nets eco012345_wire0] [get_pins frplx_reg/D]
connect_net [get_nets con_bit] [get_pins gte_344/con_bit]
```

Chapter 6: Waveform Viewer Support

Using one of SynaptiCAD's waveform viewers, you can view waveforms from a simulation or a waveform file (e.g. a VCD file) and also show specific logic states annotated on GOF schematic Windows. The schematic and the waveform displays are linked so that you can quickly display just the waveforms that match the nets of your schematic. You can also right click in the waveform window to send logic states at a particular simulation time back to the schematic.

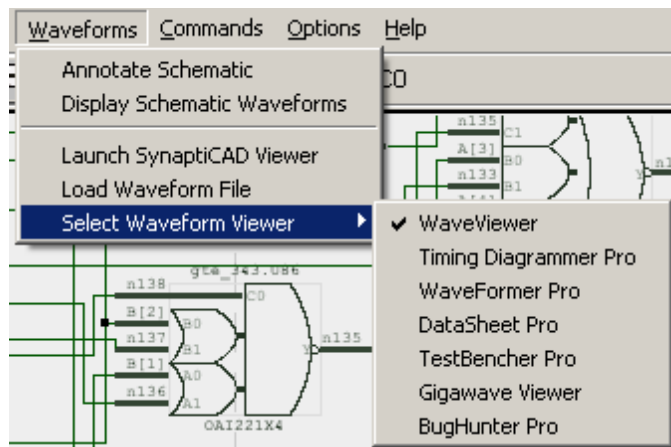


6.1 Setup SynaptiCAD's Waveform Viewer

Several of SynaptiCAD's products contain a waveform viewer that can be used with GOF to display VCD waveforms and send logic state values to GofTrace Schematic windows. If you do not own one of SynaptiCAD's waveform viewers, you can download SynaptiCAD's free WaveViewer software.

Check to Select the Viewer to use with GOF:

- In GofTrace, choose the **Waveforms > Select Waveform Viewer** menu and check the name of the viewer to use with GOF. Choose **WaveViewer** if you intend to use the free waveform viewer.



If needed, download the free waveform viewer:

If you own a license, then GOF works with v14 or higher of TimingDiagrammer Pro, WaveFormer Pro, DataSheet Pro, TestBench Pro, Gigawave Viewer, or BugHunter Pro (graphical front end to

VeriLogger Extreme). If you do not own a license, then you will need to download a free waveform viewer from the SynaptiCAD website.

- Go to the SynaptiCAD website at **www.syncad.com**.
- Click on the **Downloads** link on left side of the window.
- Fill out the form and check the box that says you are interested in the **WaveViewer** software.
- The download includes all of SynaptiCAD's waveform viewer products, so it's simple to later upgrade to a commercial version of the software if you need additional functionality beyond that of the viewer.



- Install the waveform viewer and you are ready to work with GOF. If you would also like to evaluate one of the timing diagram editors, the Verilog simulator, or the commercial version of the waveform viewer then, you can choose **Help > Request License** menu to request a 2 week evaluation license of that product.

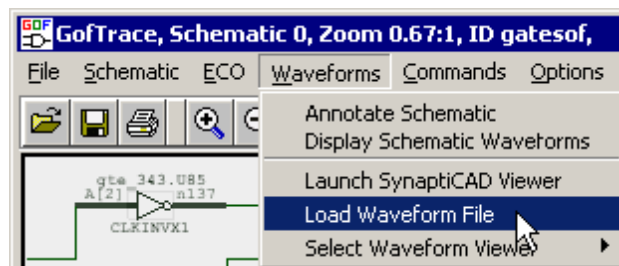
6.2 Load VCD and Launch Waveform Viewer

To establish the communications path between GOF and the SynaptiCAD waveform viewer, one of the tools needs to launch the other tool. This section describes the process of GOF launching the waveform viewer with a preselected waveform file (e.g. a VCD or BTIM file). You can also load waveform files into the waveform viewer after GOF has launched it.

An alternate way to establish communication between GOF and the waveform viewer, is to run BugHunter (which supports waveform viewing) and then use the Project window context menu to launch GOF. This is covered in Chapter 10 of the BugHunter manual.

From within GOF, load the waveform file and launch the waveform viewer:

- In GofTrace, choose the **Waveforms > Load Waveform File** menu to load in the waveform information from a waveform file.

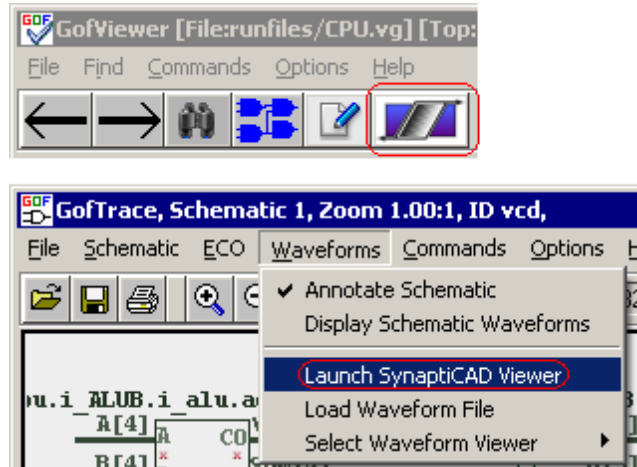


- The waveform file can also be added to the project when launching GOF using the **-vcd**

command line option as shown in [Chapter 1: Launching GOF with a Batch script](#). An example of the option is:

```
gof -v simulation_lib.v top_tb.v netlist.v -vcd top_tb.vcd
```

- After loading the waveform file, launch the waveform viewer using either the **SynapticAD** button in GofViewer, or the GofTrace menu **Waveforms > Launch SynapticAD Viewer**. If the waveform file is loaded into GOF, then GOF will pass the file to the Waveform Viewer with all of signals hidden. The next section will show you how to selectively hide or show the waveforms from the schematic.



Directly Load a VCD file or other waveform file into the SynapticAD Waveform Viewer:

After GOF has launched your SynapticAD product that contains the waveform viewer you can load any waveform file for back annotation into GOF

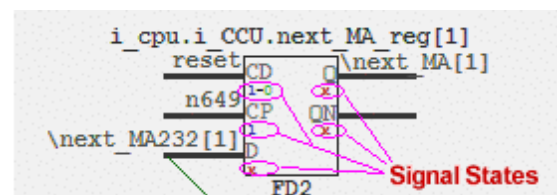
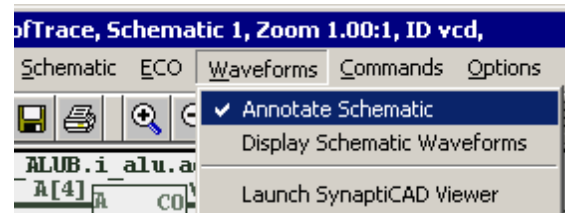
- In the waveform viewer, select the **Import/Export > Import Timing Diagram From** menu to launch the *Open Timing Diagram* dialog.
- Browse to find your waveform file and load it into the viewer.

6.3 Annotate Logic States on Schematic

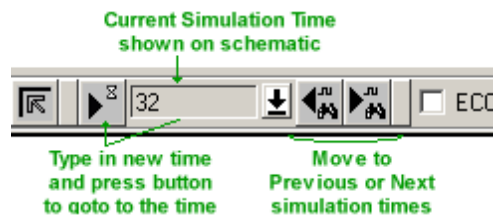
In the **Annotate Schematic** mode, GofTrace can display logic states at a given time on the pins of the schematic. You can also send states at a specific simulation time to the active schematic window directly by right clicking in the timeline of SynapticAD's waveform viewer.

Turn on the Annotate Schematic Feature:

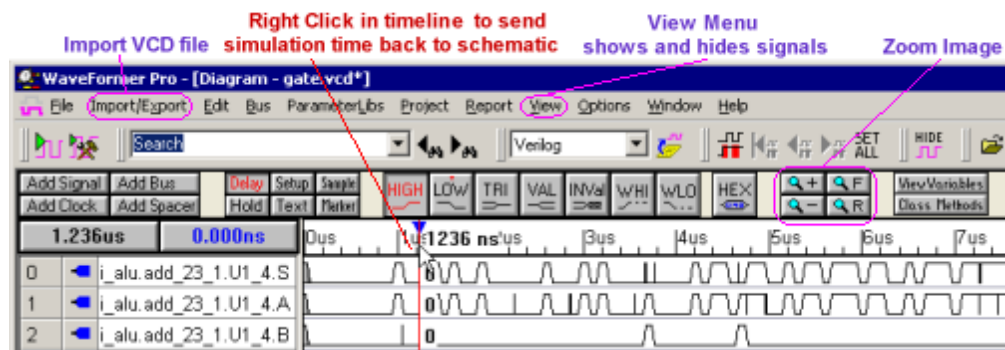
- In a schematic window, check the **Waveforms > Annotate Schematic** menu to show the logic states on the schematic gate pins and to enable the time buttons.
- The gate pins will show either 0,1,x,z or two state values if the pin is currently in transition (e.g. pin CD in the picture to the right is changing from 1 to 0).
- This feature is very useful for 'x' tracing in gate level simulations.



- The **Previous** and **Next** buttons change the simulation time to the next time where one of the pins on the schematic has a state transition.
- The **time box** shows the current simulation time.



- The time box is also an edit box that lets you enter a new simulation time and then press the **Goto Time** button to change the simulation to that exact simulation time.
- In the Waveform Viewer, you can **right click** in the timeline to send states at that simulation time back to the schematic. If the schematic is very large this may take a moment to update the schematic.



Example of Schematic Annotation Violations Features:

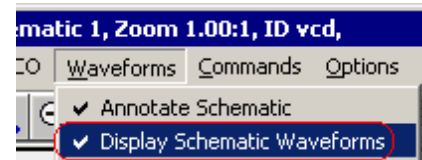
In the **SynaptiCAD > Examples > GOF_features** directory, the **case_vcd.bat** file launches GOF with a design that has a vcd file so that you can experiment with the features in this section and the next section.

6.4 Displaying Waveforms in Waveform Viewer

There are two ways to show waveforms for nets in the schematic in the Waveform viewer: (1) using the **Display Schematic Waveforms** mode to display waveforms for all the nets or (2) individually displaying the waveforms of selected nets.

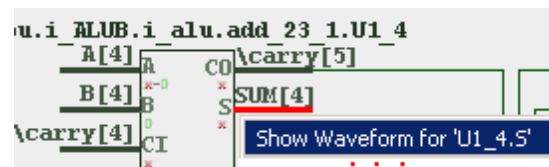
Displaying ALL net waveforms in the Waveform Viewer:

- To display all the waveforms for all the pins on a schematic, check the **Waveforms > Display Schematic Waveforms** menu. In this mode, any new gates added to the schematic will also result in new waveforms being added to the waveform viewer.



Displaying ONE net waveform in the Waveform Viewer:

- To display a waveform for a specific net, right click on a pin and choose the **Show Waveform** from the context menu.

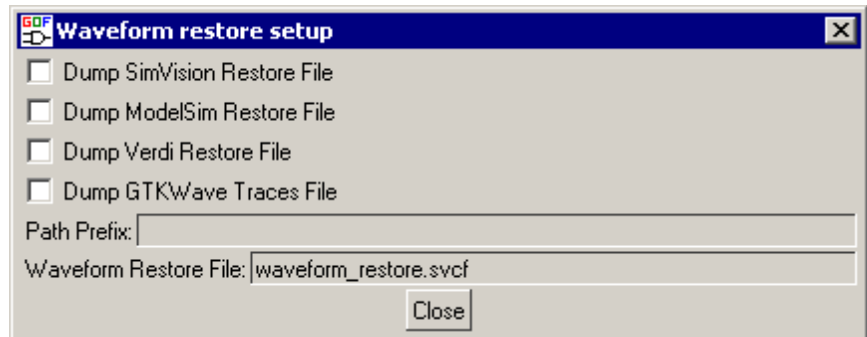


6.5 Create Waveform Restore File

The GOF tool suite works interactively with SynaptiCAD's waveform viewers as described in [Section 6.2: Show logic states from VCD file](#). However, GOF also has a method of statically working with other waveform viewers (Verdi, SimVision, ModelSim, and GTKWave). In this static method, GofViewer creates a waveform restore file that lists the waveforms to load into the waveform viewer.

Setup up the Options for the Waveform Restore File:

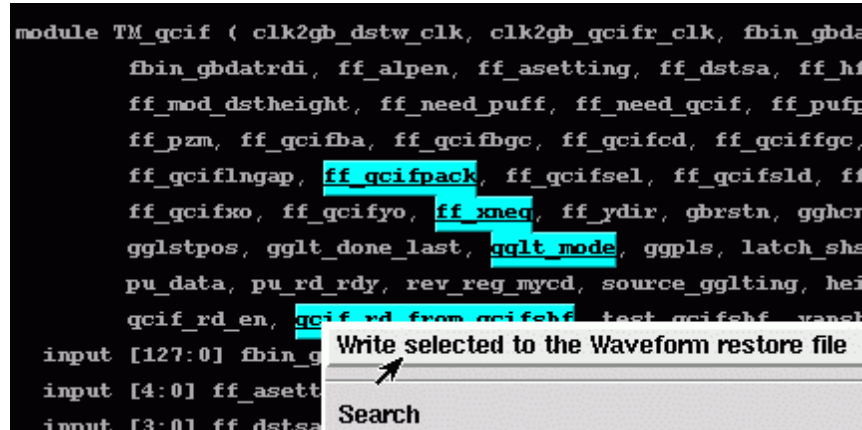
- In GofViewer, choose the **Options > Dump Waveform Restore File** menu to open the *Waveform Restore Setup* dialog.



- Use the check boxes to choose the type of Restore file to generate.
- In the **Path Prefix** box, you may enter a prefix that will be added to the beginning of each signal name in the waveform restore file. For example, if GOF contains a signal named **strbltcmd** and the path prefix is specified as **top.testop.**, the signal will be renamed **top.testop.strbltcmd** in the waveform restore file.
- In the **Waveform Restore File** box, enter the name of the file to generate.

Select the Signals to add to the Waveform Restore File and Generate the file:

- Left click to select the nets to be added to the waveform restore file. Hold the <CTRL> key while clicking to select multiple wires.
- Right click and choose **Write selected to Waveform restore file** menu to generate the file.



- In the Waveform Viewer specified in the Waveform restore setup dialog, load the Waveform Restore file that you just created. In some tools this may be called the **Restore Waveform** option. The signals you selected should be displayed in your waveform viewer.

Example of the code Generated

Below is an example of a Waveform Restore File created in GTKWave's Trace File format.

```
<?XML version "1.0"?>
<!-- GTKWAVE saved traces, create by GOF
```



```
<!-- at Mon Sep 11 17:47:57 2006
<config>
<traces>
<trace name=" WRAP.u_TM_qciff_xneg" mode="bin" rjustified="yes" selected="yes">
<signal name=" WRAP.u_TM_qciff_xneg"/>
</trace>
<trace name=" WRAP.u_TM_qciff_qcifpack" mode="bin" rjustified="yes" selected="yes">
<signal name=" WRAP.u_TM_qciff_qcifpack"/>
</trace>
<trace name=" WRAP.u_TM_qcifqcif_rd_from_qcifshf" mode="bin" rjustified="yes" select
<signal name=" WRAP.u_TM_qcifqcif_rd_from_qcifshf"/>
</trace>
<trace name=" WRAP.u_TM_qcifggt_mode" mode="bin" rjustified="yes" selected="yes">
<signal name=" WRAP.u_TM_qcifggt_mode"/>
</trace>
</traces>
</config>
```

Below is a waveform restore file saved in Verdi's Restore file format.

```
addSignal -h 15 WRAP/u_TM_qciff_xneg
addSignal -h 15 WRAP/u_TM_qciff_qcifpack
addSignal -h 15 WRAP/u_TM_qcifqcif_rd_from_qcifshf
addSignal -h 15 WRAP/u_TM_qcifggt_mode
```


Index

- + -

+define launch command 19
+libext launch command 19

- A -

Add Gate 67
Annotate Schematic 118
Area 28
Arrow Buttons 24
Auto Place & Route menu 50
Auto Place menu 50
Auto Route menu 50

- B -

BAT Files
 case_metalonly.bat 73
Batch Files 17
 examples 21
buffer 80
 example 112

- C -

case_btim.bat 21
case_gtech.bat 21
case_layoutview.bat 21
case_metalonly.bat 21, 73
case_multilibs.bat 21
case_timingvio.bat 21
case_vcd.bat 21
change_gate 80
 example 112
change_net 81
 example 112
change_pin 82
check_design 84
Claim Spare Gates Button 73
Clock Tree Trace menu 60
Color

Gate color 52
GofTrace color 52
Library color 52

Commands

Load Technology Library File 45

Commands menu

GofTrace Schematics 41

Comments

Gate level 52
Schematic level 52

compare_nets 84

Complete Loop 67

Copy Selected to Schematic context menu 41

CTRL-A 41

CTRL-S 27

current_design 71

- D -

-def launch command 19
del_gate 85
del_net 85
del_port 85
Delete Connection 70, 71
Delete floating gates 65
DeMorgans Law Displays 52
Design Statistic 28
Diff two text files dialog 34
Double Click GofViewer 24
Draw circuit between two ports/gates dialog 47
Draw fan in/out 45
Driver of net 24
Dump Waveform Restore File menu 120

- E -

ECO

Add Gate 67
Complete Loop 67
Delete Connection 70, 71
Delete floating gates 65
ECO Button Bar 65
ECO Header 65
ECO Name 65
Edit Gate dialog 70, 71
example bat file 73
Insert Gate 67

ECO

- Library for Synposys 65
- Metal Only 73
- Middle Mouse Connections 70, 71
- Replace Gate 67
- SpareCellPattern 73
- ecofile eco_file_sc.eco 71
- ecofile launch command 19
- Edit Gate dialog 70, 71
- Edit Gate Display dialog
 - cell color 52
- Equivalent symbol menu 52
- exist_wire 85
- Exit 38

- F -

- f launch command 19
- Find Circuit Between Two Points context menu 47
- Find in GofViewer 27
- Find Matching Gates in Schematic context menu 48
- Find Matching Nets in Schematic context menu 48
- Find Selected in GofViewer 48
- fix_logic 86

- G -

- get_cell_info 87
- get_cells 88
 - example 112
- get_cells example 75
- get_conns 89
- get_coord 90
- get_definition 90
- get_driver 90
- get_drivers 91
 - example 112
- get_instances 92
- get_leaf_pin_dir 92
- get_leaf_types 93
- get_leafs_count 92
- get_lib_cells 93
- get_loads 93
- get_modules 94
- get_nets 95
- get_pins 96

- get_ports 96
 - example 112
- get_refs 97
- get_roots 97
- gexit 98
- GofCall
 - API by groups 78
 - command scripts 75
 - Interactive Window 75
 - Overview 75
 - run command 75
 - script example 75
 - Script Examples 112
 - Text Mode 77

- GofTrace
 - menu overview 60
- GofTrace Schematics menu 41
- Goto Line Number 38
- gprint 98
- gtech launch command 19
- GTKWave 120

- H -

- h launch command 19
- help command
 - example 77
- Hierarchical model
 - definition 18
 - Show schematic 41

- I -

- id launch command 19
- imp sdc_ori.flat.gv 71
- Insert Gate 67
- is_leaf 98
- lv_search 99

- L -

- Launch Compare Files Window 34
- Launch GofCall Script Interface menu 75
- Launch Waveform Viewer 118
- Launching GOF 17
- LayoutViewer
 - searching 99

- Leaf Cell report 28
- leaf cells 18
- Leakage 28
- lef launch command 19
- Left click GofViewer 24
- Left Win Font menu 38
- Library for Synposys 65
- Line Select mode 50
- Line Selection Mode button 60
- List Connectivity context menu 48
- List Context 24
- List Context context menu 48
- List Fanin EndPoints 24
- List Fanin Endpoints context menu 48
- List Fanout EndPoints 24
- List Fanout Endpoints context menu 48
- List Gate menu 48
- List Logic context menu 48
- Load Design menu 38
- Load Gate Driving Net 41
- Load Gate menu 41
- Load of net 24
- Load Technology Library File 45
- Load Technology Library File menu 38
- Load Waveform File 118
- Loading files into GOF 17
- Logic Cone 45
- Logic Cone ECO 71
- logic_cone_eco.bat 21

- M -

- map_spare_cells 99
- Metal Only 73
- Metal Only ECO
 - example 112
- middle mouse button 45
- Middle Mouse in ECO 70, 71
- ModelSim 120

- N -

- Net
 - delete connection 70, 71
- net_net 100
- netlist files 18
- new_gate 100

- new_port 101

- O -

- o launch command 19
- o sc_cone.log 71
- Open Log Window menu 38
- Open netlist 17
- Open netlist menu 38
- Options for GofViewer 38
- Original symbol menu 52

- P -

- Parent Module 24
- Path Prefix 120
- pdef launch command 19
- Place & Route menu 50
- place_gate 102
- place_port 102
- Pointer Mode button 60
- pop_top 102
- Prime Time 35
- Process Timing Violations 35
- push_top 102

- R -

- read_def 102
- ref sdc_mod.flat.gv 71
- Reload Design 17
- Reload Design menu 38
- rename_net 104
- Replace Gate 67
- replace_logic_cone 105
- replace_register 71
- Report area menu 28
- Report Leaf Cells menu 28
- Report Leakage menu 28
- Report submodules menu 28
- report_eco 104
- report_sparees 104
- Reset Route menu 50
- Right Win Font menu 38
- Route
 - Automatically 50
 - Manually 50

RTL-to-Netlist Matching menu 31
 run 105
 -run command
 example 75
 -run launch command 19
 rundemo.bat 21
 rundemo2.bat 21
 RunDiff button 34

- S -

Save String to Clipboard menu 60
 Saving Schematics (non-eco) 60
 sch 105
 sch command
 example 75
 Schematic
 Display Part 41
 Draw circuit between gates 47
 Open Window 41
 Schematic menu
 List Gate 48
 Load Gate 41
 Load Gate Driving Net 41
 Search GofViewer 27
 Select ECO Gate dialog 67
 Select Line Edit Mode 50
 Select Pin Connections dialog 67
 Select Waveform Viewer 118
 set_buffer_distance 106
 set_eco_number 71
 set_exit_on_warning 106
 set_exit_or_error 106
 set_invert 106
 set_keep_format 106
 set_leaf 107
 set_log_file 107
 set_max_lines 107
 set_max_loop 107
 set_mod2mod 107
 set_mu 108
 set_power 108
 set_preserve 108
 set_quiet 108
 set_tiehi_net 108
 set_tielo_net 108
 set_top 109
 set_tree 109

set_verbose 109
 Setup Fannout Buffers Trace dialog 45
 Setup Menu
 Gof Viewer 38
 setup_eco 109
 example 112
 shell command
 example 77
 -shell launch command 19
 Show Buffers on Schematic context menu 45
 Show Comment menu 52
 Show Connections menu 52
 Show Logic Cone context menu 45
 Show Port menu 52
 Show Selection on Schematic context menu 41
 Show Title menu 52
 Show Type menu 52
 Show until non-buffer on Schematic 45
 Show Wire menu 52
 Simulation Library files
 theory 18
 SimVision 120
 Spare Cells Menu 73
 Spare Gate List 73
 SpareCellPattern 73
 -sparelist launch command 19
 start_gui 110
 example 77
 Starting GOF 17
 Statistic of current design 28
 stich_scan_chain 110
 strict_syntax 110
 Submodule report 28
 Switch to GOF Shell Mode 38, 77
 SynaptiCAD Waveform Viewer 118
 -syncad launch command 19
 -synlib launch command 19
 -synlib synthesis.lib 71

- T -

Technology Library files
 theory 18
 -textbutton launch command 19
 Timing Violations 35
 Trace Buffers 45

- U -

Undo Schematic 60
undo_eco 110

- V -

-v launch command 19
-vcd launch command 19
Verdi 120
-vn launch command 19

- W -

Waveform Restore File 120
write_dcsh 110
write_eco 71
write_spare_file 111
write_tcl 111
write_verilog 111

- Y -

-y launch command 19
-yn launch command 19

- Z -

Zoom Buttons 60