NanDigits
Design Automation

A Practical Way to Do Netlist ECO

Gates On the Fly User Manual V3.0

https://nandigits.com/gof_manual.php

Table of Contents

1 Introduction

1.1 Overview

GOF, Gates On the Fly, provides complete netlist solutions to accommodate various netlist ECO and debug scenarios.

- Automatic functional ECO uses the Reference Netlist to fix the Implementation Netlist
- Built-in logic equivalence check engine makes the ECO self contained
- Parallel processing fully utilizes multiple CPU cores to decrease the ECO run time
- Standard spare cells in Metal only ECO remaps only spare gates in post-mask ECO
- Metal Configurable Gate Array Spare Cells makes larger Metal Only ECO possible
- Auto mode ECO mixed with GUI and Script mode ECO optimizes ECO patches to the full extent
- ECO retargeting achieves huge netlist ECO in short period of time
- DFT friendly maintains test logic untouched to avoid second time ECO in later stage

1.2 Netlist ECO Solutions

GOF implements several cutting edge ECO methodologies. Netlist ECO varies in size and complexity from case to case. Design process changes from company to company. GOF gives users flexibility to choose one or several of the methodologies depending on the size and complexity of the changes.

- Automatic ECO and Manual ECO
- Script Mode and GUI Mode
- Metal Only ECO and All Layers ECO

- **Automatic mode ECO**

The automatic functional ECO is done by a GofCall ECO script. The flow needs Implementation Netlist which is under ECO, and Reference Netlist which is re-synthesized from modified RTL with the same constraints as the pre-layout netlist. The API 'fix_design' performs a top down global ECO. GOF uses the built-in Logic Equivalence Check engine to find and analyze the non-equivalent points in the top level module and its sub-modules. Logic patches are created to fix the non-equivalent modules. The final patches are optimized circuits with minimum gate count that make Implementation Netlist equal to Reference Netlist. The patches can be mapped to spare-type-gates by 'map_spare_cells' API.

- **Manual mode ECO**

When ECO changes are known and ECO size is small or the operations are repetitive like adding inverts on a bus, manual mode ECO is a better choice. Since it is more efficient and the final gates touched can be less than automatic mode ECO. Moreover, automatic and manual mode ECO can be interleaved in one GofCall ECO script.

- **Metal Only ECO**

When ECO is done in either automatic mode or manual mode, 'map_spare_cells' command is run to convert the newly added cells to spare gate types cells. Users can control only spare gate type cells being used in manual mode ECO, so that the converting stage can be bypassed. The flow supports both standard spare cells and gate array spare cells.

- **Hierarchical ECO**

GOF supports hierarchical ECO by set the ECO scope to the sub-modules. Some Logic Equivalence Check cases can only be resolved in flatten mode. Since GOF only focuses on the modules or spots that user specifies, it can avoid to get false non-equivalence in hierarchical netlist.

- **GUI mode ECO**

GUI mode ECO has advantage of fast ramping up. It’s good for small size ECOs. The incremental schematic feature is very helpful for analyzing the netlist before the next step is decided.

- **Integrated environment**

The incremental schematic is very useful in pinpointing the logic issues. GOF supports many useful APIs to fast access the database.

1.3 Download and Install GOF

GOF release package can be found in

https://nandigits.com/download.php

The tool supports Linux 64bits OS. Download the release package and unzip to a directory. Set 'the_64bit_install_path/GOF64/bin' in search path.

1.4 License and Setup

Visit https://nandigits.com/support.php?type=license to request an evaluation license. Or email support@nandigits.com for more information. Without license, the tool can support netlist size less than 500K bytes. There are two license modes, fixed node mode and floating node mode.

- Fixed node license: Copy the license file to "the_install_path/GOF64/bin" and restart GOF.
- Floating node license: Please refer to this page to install floating license https://nandigits.com/floating_license_setup_example.htm

2 Netlist ECO Flows

2.1 Automatic Full-Layers Functional ECO Flow

2.1.1 Overview

Full layers functional ECO can add or delete gates freely. The ECO operations are done in a GofCall script which is compatible with Perl, and it uses exported APIs to access, modify and save the netlist database. GOF reads in two netlist files, Implementation Netlist which is under ECO and Reference Netlist which is re-synthesized from modified RTL with the same constraints as the pre-layout netlist. In the GofCall script, the API 'fix_design' is used to fix the top level module and its sub-modules in global mode. GOF uses the built-in Logic Equivalent Check Engine to figure out the non-equivalent points. And optimized minimum size gate patches are applied to fix the non-equivalent modules.

As shown in Figure 1, two logic cones are extracted from the Implementation and Reference Netlist for the same compare point. The implementation point mismatches the reference point initially. GOF compares the two points and generated a patch from Reference logic cone and applies to Implementation Netlist. After the patching, the two points become equivalent.

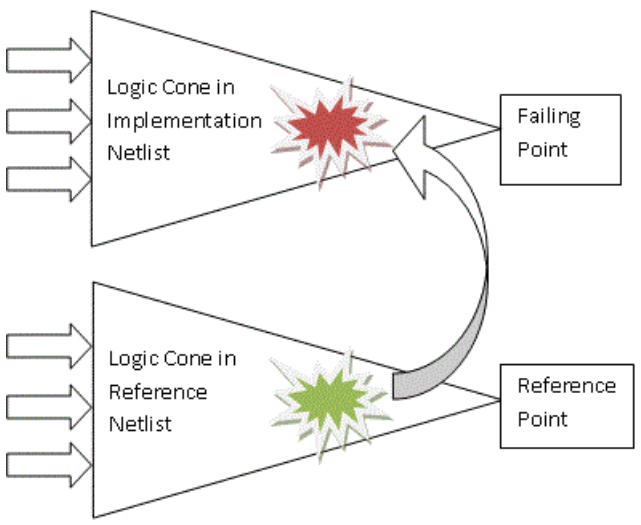

Figure 1: Logic Cone Optimization

GOF does logic cone analysis and optimization for each failing point found in top down logic equivalence check. The failing point is in format of output port or sequential element's input pin, such as flop's D input. The final patch has the minimum number of gates to make the implementation logic cone equal to the reference logic cone.

The flow chart is shown in Figure 2.

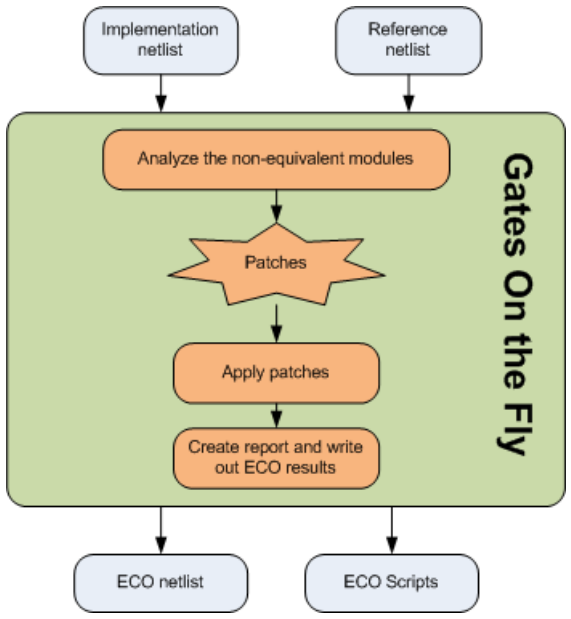

Figure 2: Automatic functional ECO flow

2.1.2 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.lib'
- Other Verilog libraries files if not covered in '.lib' files
- Implementation Netlist on which ECO is done
- Reference Netlist synthesized with the same constraints as the pre-layout netlist
- The level module name under ECO

2.1.3 Steps to do automatic functional ECO

Steps for an automatic functional ECO:

- Modify RTL
- Synthesize the new RTL to get Reference Netlist
- Create a GofCall ECO script:
  - Specify ECO name in 'setup_eco'
  - Load Standard Cell libraries and Verilog libraries
  - Load Reference Netlist and Implementation Netlist
  - Add fix command 'fix_design'
  - Report ECO status and write out ECO results
- Run the above script

2.1.4 Automatic Functional ECO example script

The GofCall script has exact the same syntax of Perl script. It runs the exported APIs that access the netlist database and modify the netlist.

The following is an example script for an automatic functional ECO:

```
# GofCall ECO script, run_example.pl
use strict;
undo_eco; # Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("art.5nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("topmod");# Set the top module
# Preserve DFT Test Logic
set_ignore_output("scan_out*");
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit; # Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.1.5 Run and debug

The GofCall Script can be run by '-run' option.

```
gof -run run_example.pl
```

Check Run and debug GofCall script section in User Manual for more detail

2.1.6 Partial Mode

If all RTL changes are known to be contained in one sub-block and its sub-modules, the top level scope can be set to the sub-block. 'fix_design' can apply to the block and its sub-modules.

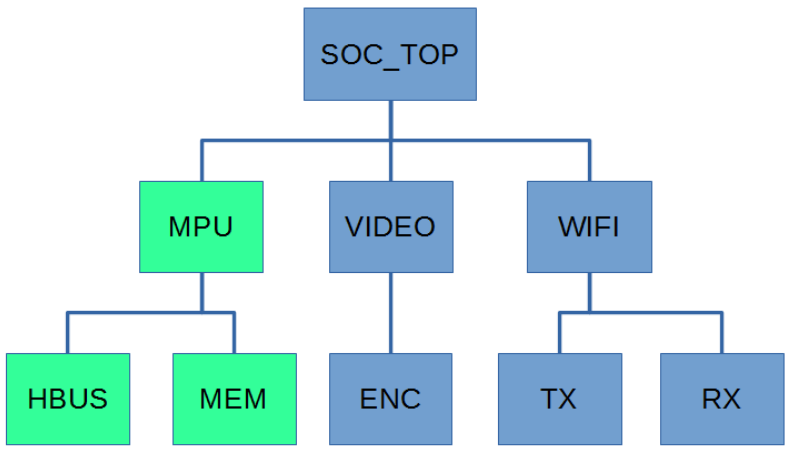

Figure 3: Partial Mode

For example, the above design has only MPU/HBUS/MEM modified. The top level can be set to MPU by 'set_top("MPU")'

```
# GofCall ECO script, run_example_partial.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("tsmc.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("MPU");# Set the top module to MPU instead of the most top module SOC_TOP
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
set_top("SOC_TOP"); # Set the top module to the most top module
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.1.7 ECO Retargeting

ECO retargeting is to run ECO on the block level netlist and dump the ECO script. The ECO script is retargeted to the post-layout netlist by applying changes on the block in the large netlist. The ECO script is in Perl syntax written out by 'write_perl' command.

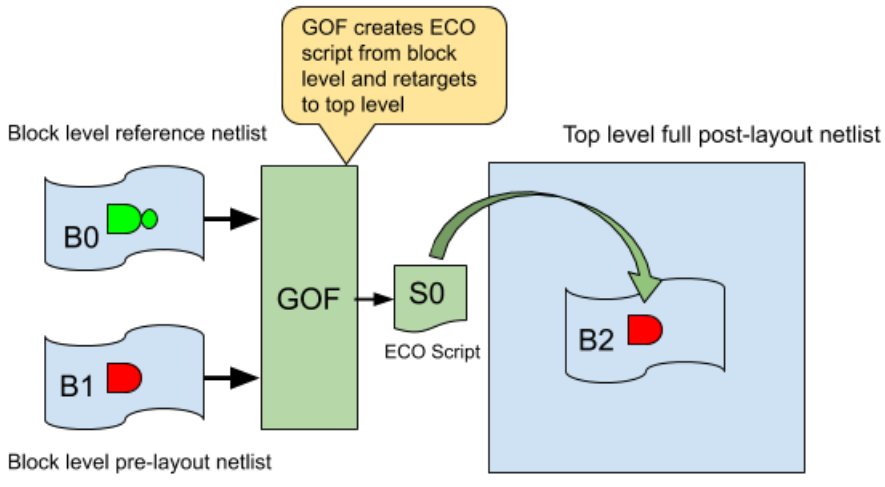

Figure 4: ECO Retargeting in Functional Netlist ECO

As shown in Figure 4, B0 is the block level reference netlist which is synthesized after RTL change; B1 is the block level pre-layout netlist which doesn't have boundary optimized done by the back-end tool; B2 is the block inside the large post-layout netlist. B1 and B2 are equivalent, but they maybe different in structure due to the optimization done by the back-end tool.

Synthesizing the full design to get the large complete new netlist would take long time, the block level B0 is much easier to be generated through synthesis. Doing ECO between B0 and B1 is much faster than between the two full netlists.

The first round ECO is run on block level to generate block level ECO script:

```
# GofCall ECO script, run_example_eco_retarget1.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("tsmc.lib");# Read in standard library
read_design('-ref', 'mpu_block_new_syn.v');# Read in block level Reference Netlist, B0 in Figure 4
read_design('-imp', 'mpu_block_pre_layout.v');# Read in prelayout block level Implementation Netlist, B1 in Figure
4
set_top('MPU');# Set the top module to MPU
set_ignore_output("scan_out*"); # The block level DFT signals may have different names
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
write_perl('mpu_block.gpl'); # ECO script will apply to large post-layout netlist
exit;
```

The second round ECO is to apply block level ECO script to the large top level post-layout netlist:

```
# GofCall ECO script, run_example_eco_retarget2.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("art.5nm.lib");# Read in standard library
read_design("-imp", "top_level_implementation.gv");# Read in Implementation Netlist Which is under ECO, B2 is the
block inside shown in Figure 4
set_top('MPU'); # Set the top module to sub-block MPU
run('mpu_block.gpl'); # The ECO script, S0 in Figure 4, is retargeted on the sub-block MPU in the large post-
layout netlist
set_top("SOC_TOP"); # Set the top module to the most top
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

One disadvantage is some instances appeared in the ECO script 'mpu_block.gpl' may have been optimized away by back-end tool. In this rare case, the ECO script should be manually adjusted and apply it to the netlist again.

To debug and search for the right instance or net in the netlist, GOF Incremental Schematic feature can be used. Refer to Incremental Schematic for more detail.

2.1.8 DFT Constraints

DFT logic should be constrained to avoid false non-equivalence in LEC and ECO. In the traditional DFT flow, the DFT logic is normally created in the RTL design which shows up in both Reference Netlist and Implementation Netlist. In the modern DFT flow, in which IEEE1687 and IEEE1500 standards are supported, the DFT logic is normally inserted into Implementation Netlist by the DFT tool, for example Mentor Tessent. And Logic Equivalence Check needs to be run to check Implementation Netlist which has DFT logic inserted by the DFT tool against Reference Netlist which has no DFT logic. To avoid false and redundant ECO fix, the automatic functional ECO should properly constrain the DFT logic.

In the traditional DFT flow, as shown in left side of the Figure 5, the constraints are set on the ports. For example, setting DFT control signals like TEST_EN to zero and leave the normal functional ports unconstrained.

```
# Set DFT Constraints
set_pin_constant("TEST_EN", 0);
set_pin_constant("scan_mode", 0);
set_ignore_output("scan_out*");
```

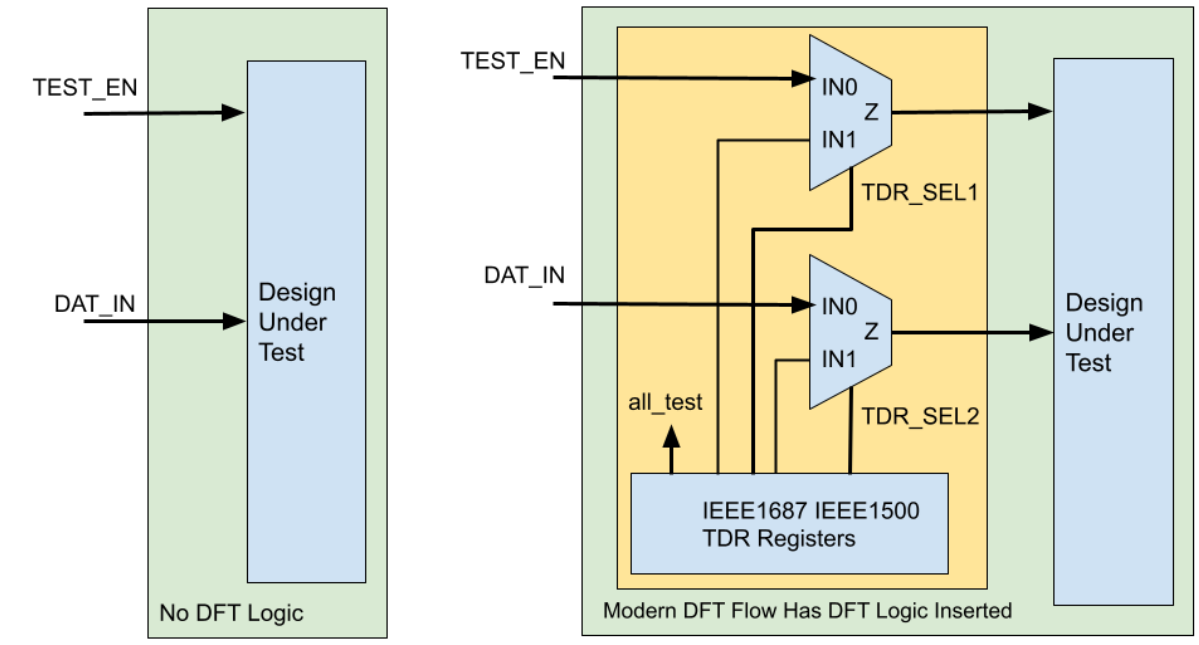

Figure 5: DFT Constraints in Automatic Functional ECO

In the modern DFT flow, these inserted DFT logic by the DFT tool as shown in the right side of the Figure 5 should be constrained to be in inactive state. The control signals driven by TDR registers should be constrained to zeros.

GOF provides several APIs to constrain the DFT logic, set_ignore_output, set_pin_constant and set_net_constant. The API set_net_constant can be used to constrain the TDR registers signals. Since TDR registers are not ports, so they have be treated as nets.

```
# Set DFT Constraints for the modern DFT flow
set_pin_constant("TEST_EN", 0);
set_pin_constant("scan_mode", 0);
set_ignore_output("scan_out*");
set_net_constant("TDR_SEL0", 0, "-imp"); # TDR register net only exists in Implementation Netlist
set_net_constant("TDR_SEL1", 0, "-imp");
set_net_constant("all_test", 0, "-imp");
```

The full script with constraints on the traditional DFT flow is shown below:

```
# GofCall ECO script, run_example_exclude_test_logic.pl
# The SOC_TOP design should have scan insertion test logic excluded in ECO.
# The scan out bus pin has naming of scan_out[199:0] and API set_ignore_output can be used to exclude LEC check on
scan_out in ECO.
# And TEST_EN and scan_mode are two scan set up signals which can be forced to zeros by API set_pin_constant.
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("tsmc.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("SOC_TOP"); # Set the top to the most top module SOC_TOP
set_ignore_output("scan_out*");
set_pin_constant("TEST_EN", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

The full script with constraints on the modern DFT flow is shown below:

```
# GofCall ECO script, dft_constraints_on_inserted_test_logic.pl
# set_net_constant is used to constrain TDR register nets to zeros
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("tsmc.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("SOC_TOP"); # Set the top to the most top module SOC_TOP
set_ignore_output("scan_out*");
set_pin_constant("TEST_EN", 0);
set pin constant("scan mode", 0);
set_net_constant("TDR_SEL0", 0, "-imp"); # TDR register net only exists in Implementation Netlist
set_net_constant("TDR_SEL1", 0, "-imp");
set_net_constant("all_test", 0, "-imp");
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.1.9 No Exact Pin Match

Physical Synthesis is more and more popular in logic synthesis. Physical Synthesis tool, DCT in topographical mode for example, may add hierarchical pins that are not in RTL code and it may cause mapping issue when Implementation Netlist is comparing with Reference Netlist in ECO.

For example, DCT may add 'IN0', 'IN1', 'IN2', 'IN2_BAR' ... to hierarchical modules. The new added pins are not necessarily matching to each other in Implementation Netlist and Reference Netlist. That is, IN0 in module A in Reference Netlist maybe a different signal from IN0 in module A in Implementation Netlist.

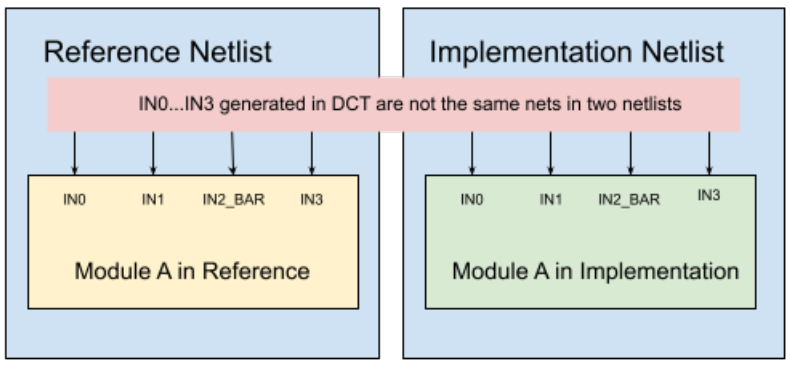

Figure 6: No Exact Pin Match

These pins are randomly named in each run. They won't affect logic equivalence check, but they need to be excluded in pin matching in ECO. Otherwise, the ECO tool would insert redundant logic or wrong logic.

API set_noexact_pin_match can be used to resolve the mapping issue between Implementation Netlist and Reference Netlist.

By adding the port naming regular expression in the API argument, set_noexact_pin_match('\bIN\d+(_BAR)?\b'), these ports will be remapped.

Note: This API should be run before reading designs.

```
# GofCall ECO script, run_example_noexact_pin_match.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
set_noexact_pin_match('\bIN\d+(_BAR)?\b'); # The argument is in REGEX format to detect IN0/IN0_BAR/IN1...
# Note: set_noexact_pin_match API should be run before reading designs!
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("SOC_TOP"); # Set the top to the most top module SOC_TOP
set_ignore_output("scan_out*");
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.1.10 Flip-flop Phase Inverted

In power timing critical designs, Place and Route tool may change some flip-flops' phase by adding inverters in input pin and output pin.

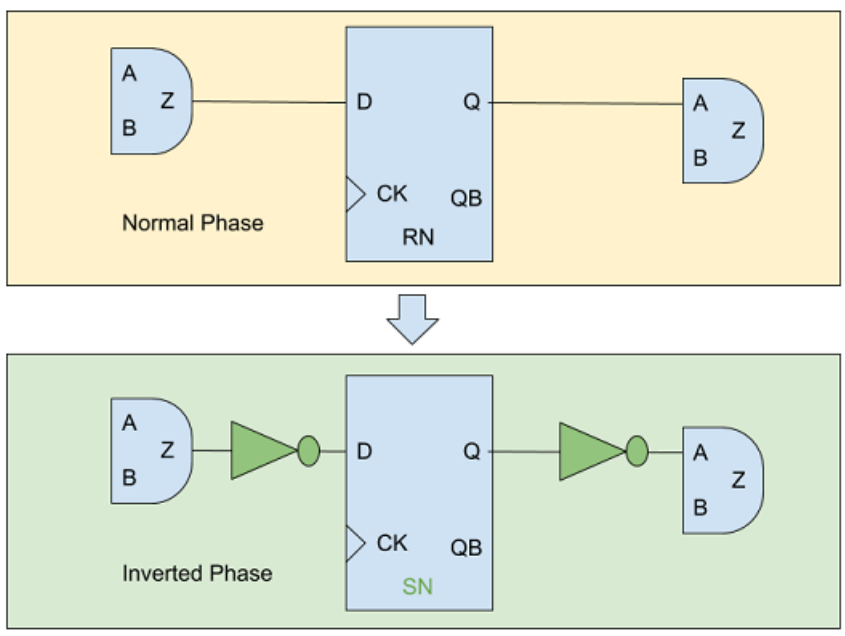

Figure 7: Flip-flop Phase Inverted

API set_mapping_method('-phase') is used to handle these cases.

```
# GofCall ECO script, run_example_ff_phase_inverted.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("art.90nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("SOC_TOP"); # Set the top to the most top module SOC_TOP
set_mapping_method('-phase'); # Check flop phase during LEC
set_ignore_output("scan_out*");
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.1.11 Logic Equivalence Check Engine

2.1.11.1 LEC for Two Netlists

The built-in Logic Equivalence Check Engine checks the equivalence of two netlists. The speed and memory usage are similar to Conformal LEC and Formality.

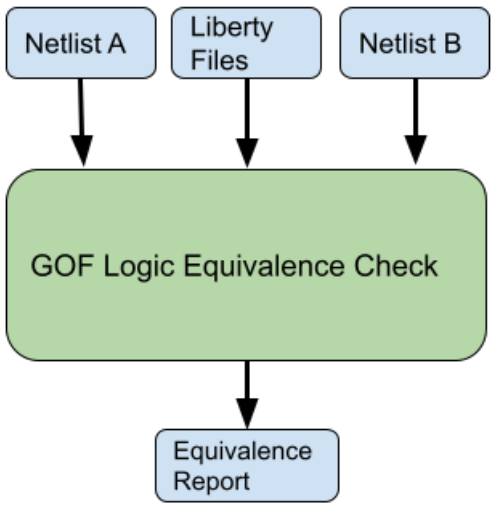

Figure 8: GOF LEC Engine

Two netlists can be checked if they are equivalent by run_lec API.

```
# LEC script, run_example_lec.pl
use strict;
read_library("art.5nm.lib"); # Read in standard library
read_design('-ref', 'AI2021_top_syn.v'); # Read in the Reference Netlist, prelayout netlist
read_design('-imp', 'AI2021_top_pr.v'); # Read in the Implementation Netlist, postlayout netlist
set_top("AI2021_top"); # Set the top module
set_ignore_output("scan_out*");
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
my $non_equal = run_lec; # Run logic equivalence check on the two netlists
if($non_equal){
  gprint("LEC failed with $non_equal non-equivalent points");
}else{
  gprint("LEC passed");
}
```

2.1.11.2 Equivalent Nets Searching

In ECO process, GOF searches for equivalent nets in Implementation Netlist to optimize the patch circuit. The searching process is global.

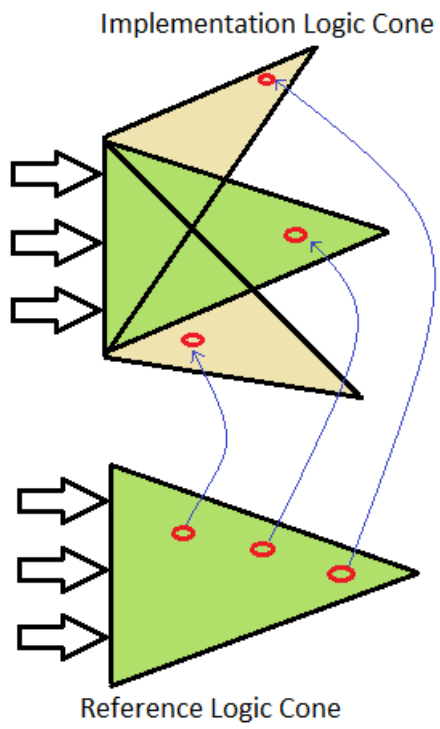

Figure 9: Equivalent Nets Searching

Any two nets in the Reference and Implementation Netlists can be checked if they are equivalent. The API 'comare_nets' can be used to compare any two nets in Reference Netlist and Implementation Netlist.

Check equivalence of two nets in the reference and implementation netlist

```
Usage: my $result = compare_nets($net0, $net1);
$net0: The net in Reference Netlist.
$net1: The net in Implementation Netlist.
$result: If 1, they are equal, if 0, they are not equal.
Examples:
# Compare reg1/D in the reference and reg1/D in Implementation Netlist
compare_nets("reg1/D", "reg1/D");
```

2.1.12 Stitch new flops into scan chain

New flops inserted in an ECO should be stitched into existing scan chains to avoid DFT coverage loss. From the industrial data, 100 new non-scan flops in a design with 100K flops can cause more than 0.1% DFT coverage loss. For the high-reliability chips like Automobile IC, the DFT coverage loss is not acceptable. So if there are new flops in a functional ECO, the scan chain should be redone to include the new flops.

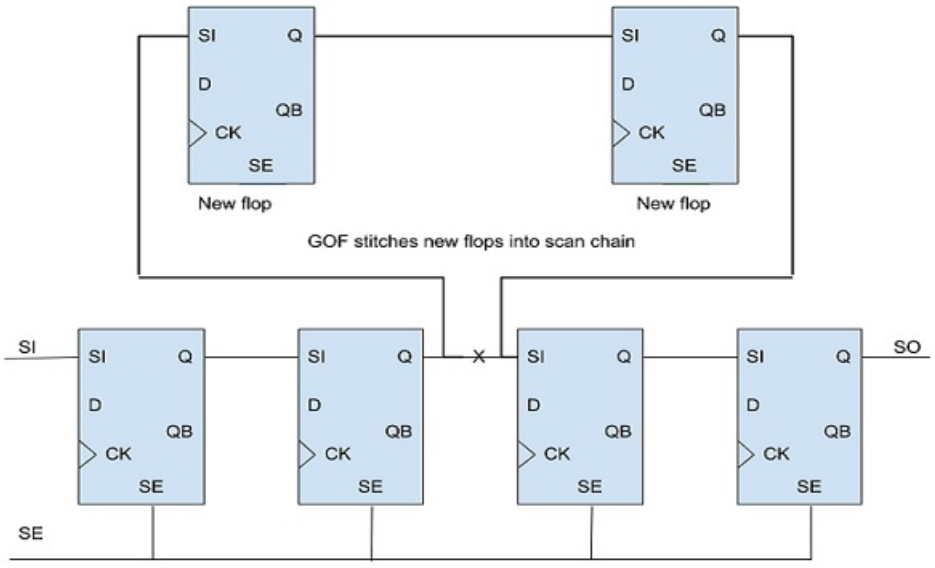

Figure 10: Stitch scan chain

GOF provides several ways to insert the new flops into scan chains. The API 'stitch_scan_chain' can be used to automatically stitch scan chains by inserting the new flops. A manually way is supported by using several netlist processing APIs.

Automatic mode to insert flops into a scan chain in the local modules

For example, eight new flops 'state_new_reg_0' to 'state_new_reg_7' are added in fix_design command. To insert them into scan chain in the local module:

```
# API stitch_scan_chain without any argument to insert new flops in the local modules
stitch_scan_chain();
```

Automatic mode to insert flops before one flop

Users can specify one flop instance name, so that GOF can insert all new flops to the scan chain before the flop instance.

For example, insert all new flops to the scan chain before instance 'u_pixel_ctrl/pulse_reg':

```
# API stitch_scan_chain with -to option
stitch_scan_chain('-to', 'u_pixel_ctrl/pulse_reg');
```

Manual mode to connect up all new flops

The scan chain can be re-connected up manually by ECO APIs. And new scan in/out ports are created.

```
# GofCall ECO script, run_manual_stitch_scan_chain_example.pl
use strict;
undo_eco; # Discard previous ECO operations
setup_eco("eco_manual_stitch_scan_chain_example");# Setup ECO name
read_library("art.5nm.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("topmod");# Set the top module
set_ignore_output("scan_out*");
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
set_error_out(0); # Don't exit if finds error
my @flops = get_cells("-hier", "-nonscan"); # Find all new flops that are not in scan chain yet
# @flops can be defined by reading a list file
if(scalar(@flops)){ # If there are new flops, start the work
  new_port("so1", "-output"); # New a scan out port so1
  new_port("si1", "-input"); # New a scan in port si1
  my $cnt = 0;
  my $now_si;
  foreach my $flop (@flops){
    $cnt++;
    if(is_scan_flop($flop)==0){
      my $flop_name = get_ref($flop);
      my $scanflop = get_scan_flop($flop_name); # If the flop is not scan type, change to scan type flop
      change_gate($flop, $scanflop);
    }
    if($cnt==1){
      change_port("so1", "$flop/Q"); # The first flop drives the new scan out port
    }else{
      change_pin($now_si, "$flop/Q");
    }
    $now_si = "$flop/SI";
    change_pin("$flop/SE", "te"); # All scan enable pin is connected to scan enable signal
  }
  change_pin($now_si, "si1"); # The last flop has the new scan in port driving SI pin
}
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;
```

2.1.13 Add a new module

The module mentioned in the section above can have hierarchy kept instead of flatten, and being written into ECO netlist as whole. This flow needs the module and its sub-modules written out in a separate verilog file, then uses read_library to load the file with '-vmacro' option. GOF treats the module as a leaf cell.

An example for adding a new module:

```
# GofCall ECO script, run_new_module_example.pl
use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_hier_example");# Setup ECO name
read_library("tsmc.lib");# Read in standard library
read_library("-vmicro", "syn_macro.v"); # The syn_macro module is added into the netlist
read_design("-ref", "reference.gv");# Read in the Reference Netlist
# Read in the implementation netlist which is under ECO
read_design("-imp", "implementation.gv");
set_top('top');# Set the top module
set_ignore_output("scan_out*");
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
report_eco();# ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
```

The content in file syn_macro.v is written into the ECO file eco_verilo.v as a whole. The corresponding instance is created as well with ports connected correctly according to Reference Netlist.

2.1.14 Note in RTL modification and re-synthesis

When modifying RTL and do re-synthesis, care should be taken to maintain the database as much alike Implementation Netlist as possible.

2.1.14.1 Keep sequential signal name

A common problem in modifying RTL is having sequential signal name changed, which appears in Reference Netlist as a different flop instance. For example

```
always @(posedge clk) abc <= abc_next;
```

It creates a flop instance 'abc_reg' in synthesis. If the ECO in RTL change this to

```
always @(posedge clk) abc_new <= abc_next;
```

After synthesis, a new flop instance 'abc_new_reg' is created. GOF may fail to find that 'abc_new_reg' being able to merge with 'abc_reg', due to other non-equivalent points present, which brings a redundant fix in the new register creation.

So it is highly recommended to keep the sequential signal names in re-synthesis.

2.1.14.2 Use the same synthesis constraints

When do re-synthesis, the same constraints should be used as what has been used in Implementation Netlist synthesis. If any hierarchy is not present in Implementation Netlist, it's better to flatten the module in synthesis to maintain the same hierarchies.

2.1.15 Check design after ECO

It is highly recommended to run 'check_design' after ECO, to speed up, users can specify '-eco' option,

```
check_design('-eco')
```

It can detect if there is any floating or multiple drivers after ECO.

2.2 Standard Cells Automatic Metal Only ECO Flow

2.2.1 Overview

In Metal Only ECO, the design has completed place and route. Any new gates added should map to spare gates that located in the design. GOF supports Standard Spare Cells and Metal Configurable Gate Array Spare Cells post-mask metal only ECO.

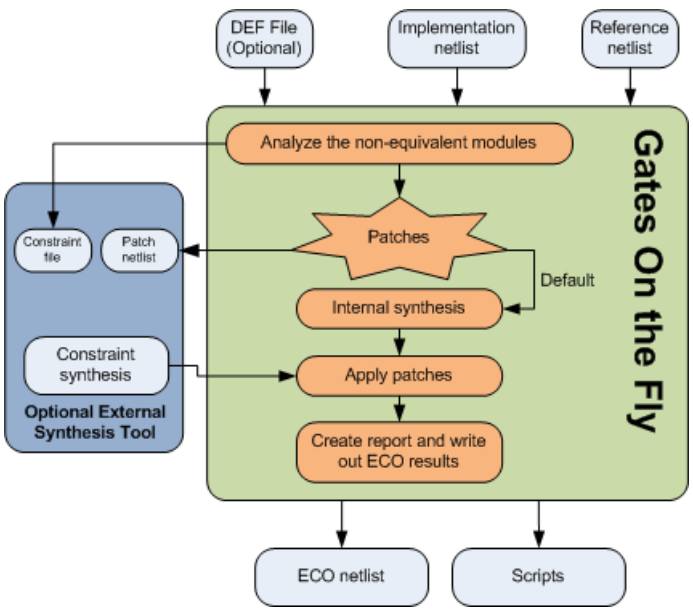

Figure 11: Metal Only ECO

2.2.2 Standard Cells Spare Gates Mapping

GOF uses internal synthesis engine or external synthesis tool to map patch logic to spare gates. The spare gate list requires one or both of the following spare type combinations.

- Two ports 'and/or' gates, 'inv' gates and flops, 'mux' is optional.
- Two ports 'nand/nor' gates, 'inv' gates and flops, 'mux' is optional.

In the Figure 12, the circuit generated by ECO in the left side has random standard cells. The mapping process maps the MUX and flop type gates directly to the spare gates, since they have one to one matching gate in the spare gate list. For the complicated cell type AO32, it has to be synthesized and mapped to 3 AND gates and one NOR gate.

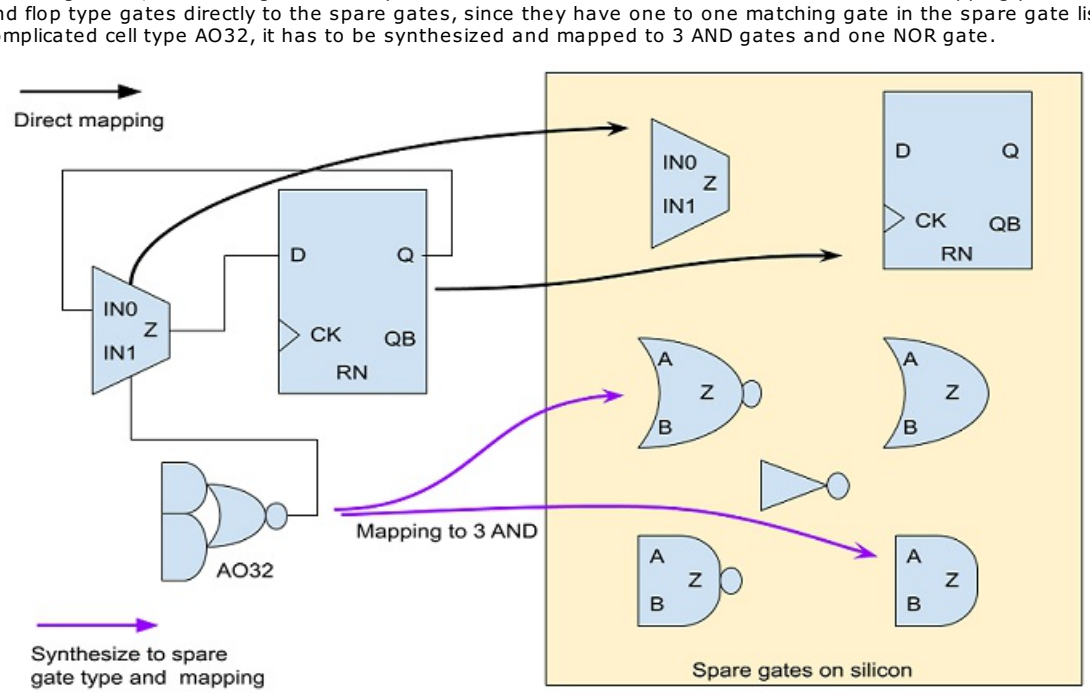

Figure 12: Standard Cells Spare Gates Mapping

2.2.3 Spare Gates Synthesis

A Design Exchange Format file is needed to map new instances to the closest spare gate instances. If DEF file is not loaded, GOF processes the ECO with gates type from the spare list without mapping to the exact spare instances. P&R tool like SOC Encounter maps the new instances in the new netlist to the closest spare gates.

In 'fix_design' command, GOF analyzes the top level module and its sub-modules to isolate the non-equivalent points and optimize the logic cone to find the minimum gate count patch circuit.

2.2.4 External Synthesis Tool for Spare Gates Synthesis

The flow can use external Synthesis Tool as well. The executable synthesis command should be in the search path. The supported Synthesis Tool is RTL Compiler from Cadence and Design Compiler from Synopsys.

GOF writes out the patch in Verilog file and a TCL script for external Synthesis Tool if it's enabled. The TCL script is to constrain the Synthesis Tool to use spare gates only when remapping the gates in the patch file. The Synthesis Tool is run with the Verilog file and TCL script as inputs, and it writes out remapped Verilog patch file which has only spare gate types.

When the spare-only patch file is created, user can pause the flow by '-pause' in 'map_spare_cells' command. User can either modify the patch file manually or tunes up the constraint file to rerun the synthesis for several iterations until the patch netlist meets the requirement. Then press 'n' key to resume the flow.

GOF reads back the spare-only patch file and fits the circuit into Implementation Netlist to fix the logic cones.

When ECO is done, a report can be created and ECO netlist/ECO script can be written out for the back end tool and LEC tool.

2.2.5 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.lib'
- Other Verilog libraries if '.lib' files can't cover
- Implementation Netlist
- Reference Netlist
- DEF (Design Exchange Format) file. It's optional. If it is not loaded, GOF won't map the spare gate type cells to the exact spare instances
- Spare gates pattern. It is in 'hierarchical_instance/leaf_instance' format. It has wild card '*' to match the spare gates in Implementation Netlist
- Spare gates list file. If several users work on the same Implementation Netlist, the initial spare gates list file should be generated only once. And new spare gates list file must be created every time an ECO is done

2.2.6 Steps to do automatic Metal Only ECO

A typical process for an automatic Metal Only ECO:

- Change RTL modules
- Synthesize the new RTL to get Reference Netlist
- Create a GofCall ECO script:
  - Load Standard Cell libraries and Verilog libraries
  - Load Reference Netlist and Implementation Netlist
  - Add 'fix_design' command
  - Load DEF file, optional
  - Load LEF file, optional. It's useful in LayoutViewer feature
  - Create Spare Gates List by Spare Gates pattern or by reading in spare list file
  - Run 'map_spare_cells' to remap the patch from 'fix_design' command to all spare -type gates patch netlist and select the closest spare instances for each gate in the patch netlist
  - Report ECO status and write out ECO results
- Run the above script

2.2.7 Example GofCall script for Metal Only ECO

The GofCall script has the exact same syntax of Perl script. It runs the exported commands that access the netlist database and modify the netlist.

The following shows an example of an automatic Metal Only ECO:

```
# GofCall ECO script, run_metal_only_example.pl
use strict;
undo_eco;# Discard previous ECO operations
# Setup ECO name
setup_eco("eco_metalonly_example " );
read_library("tsmc.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("topmod");# Set the top module that ECO is working on
set_ignore_output("scan_out*");
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
# The following is metal ECO related
read_def("topmod.def");# Read Design Exchange Format file, optional
# Specify spare cell pattern, when 'map_spare_cells' is done, a new spare list file is written out
# with updated spare list.
get_spare_cells("*/*_SPARE*");
# Comment the above line and use the following line to use spare list file
# if the spare list file has been generated already and gone through other ECOs
# get_spare_cells("-file", "spare_list_file.txt");
# set_constraints("-num", "and<20"); # set_constraints is optional to control AND cell usage under 20 counts
map_spare_cells();
# Use one of the following lines if external Synthesis Tool is used
#map_spare_cells ( "-syn", "rc" );
#map_spare_cells ( "-syn", "dc_shell" );
report_eco(); # ECO report
check_design("-eco");# Check if the ECO causes any issue, like floating
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.2.8 Run and debug

The GofCall Script can be run by '-run' option.

gof -run run_metal_only_example.pl

User can insert 'die' command to let GOF stop in some point and do interactive debugs when 'GOF >' shell appears. GUI mode can be enabled by run 'start_gui' command.

Check Run and debug GofCall script section for more detail

2.2.9 Gated clocks in Automatic Metal Only ECO

If the automatic metal only ECO has new gated clock cells added while the spare gates list doesn't have gated clock cell, "convert_gated_clocks" API should be run to convert gated clock cells to 'MUX' type logic. GOF maps the 'MUX' type logic to the spare type gates in 'map_spare_cells' API.

```
get_spare_cells("*/*_SPARE*");
convert_gated_clocks();
map_spare_cells();
```

2.3 Metal Configurable Gate Array Cells ECO Flow

2.3.1 Overview

Metal configurable gate array cells are specially developed for Metal Only ECO. There are two types of gate array cells used in different backend stages. The first type is gate array spare cells which are normal filler or decap cells in the original flow. In the backend P&R stage, gate array spare cells, like GFILL/GDCAP, are added and scattered over the design. The second type is gate array functional cells. In post-mask ECO, gate array spare cells are swapped out by gate array functional cells, like GAN2, GND2, GXOR2.

2.3.2 Gate Array Cell Base Tile

The base unit of gate array cell is a tile. Every gate array cell consists of one or more tiles. Use one 5nm standard library as

example:

Tile Numbers	Spare Cells	Functional Cells
1	GFILL1	GTIE GINVD1 GND2D1 GNR2D1
2	GFILL2	GBUFD1 GAN2D1 GOR2D1 GAOI21D1 GDN3D1
3	GFILL3	GAO21D1 GAN4D1 GOR4D1
4	GFILL4	GINVD8 GAN2D4
5	GFILL5	GMUX2D1 GXOR2D1 GXNOR2D1
6	GFILL6	GBUFD8 GSDFFRQD1 GSDFFSQD1
8	GFILL8	GINVD16
12	GFILL12	GCKLNQD6

Table: Tile Numbers in Gate Array Spare Cells and Functional Cells

Gate array cells are larger than the normal standard cells. For example, GFILL1 is four times larger than FILL1 and GND2D1 is 25% larger than ND2D1. The power consumption and timing are similar.

Each gate array spare cell has the location defined by DEF file. As shown in Figure 13, one GFILL8 spare cell has location defined as (Xg, Yg). The tile width is the width of GFILL1.

The tiles on the GFILL8 can be regrouped and rewired in metal layers to form different functional cells. For example, GBUFD1 takes two tiles and implements buffer function and GAN4D1 takes 3 tiles to form 4 inputs AND function.

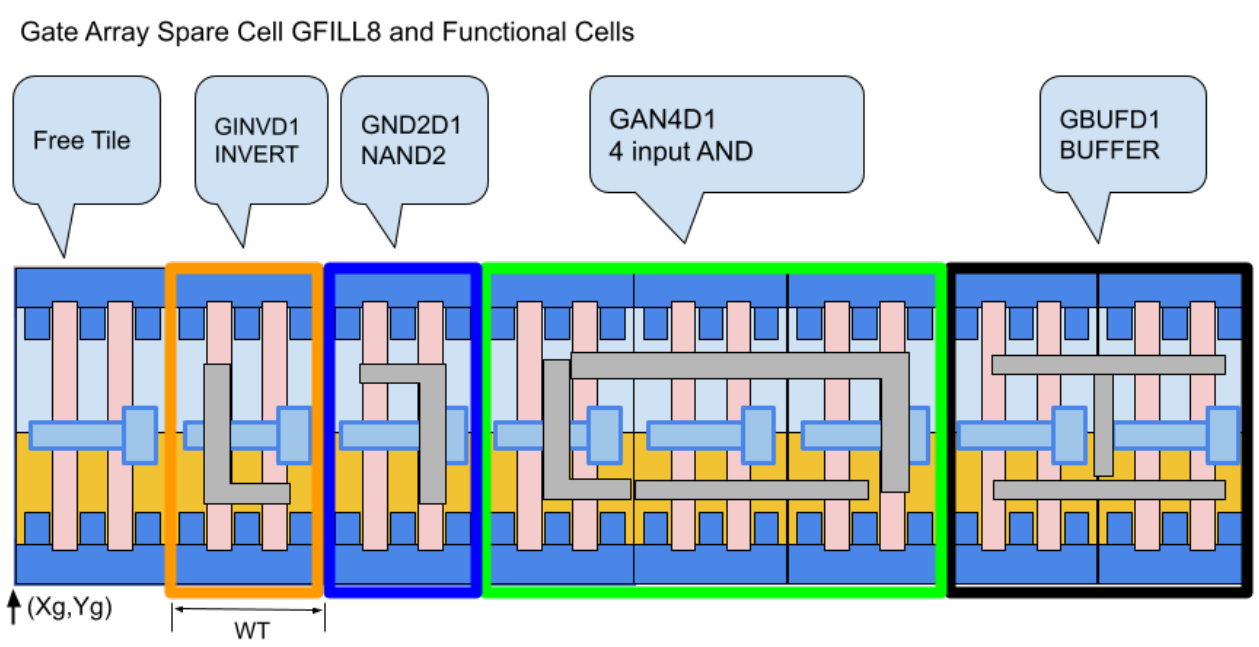

Figure 13: Gate Array Spare Cell GFILL8 Regrouped Tiles to Form Functional Cells

2.3.3 Automatic Mapping

When GOF generates a patch, it synthesizes the patch to gate array functional cell types only. Then the gate array functional cells are mapped to the optimal close by gate array spare cells with minimum wire connection costs.

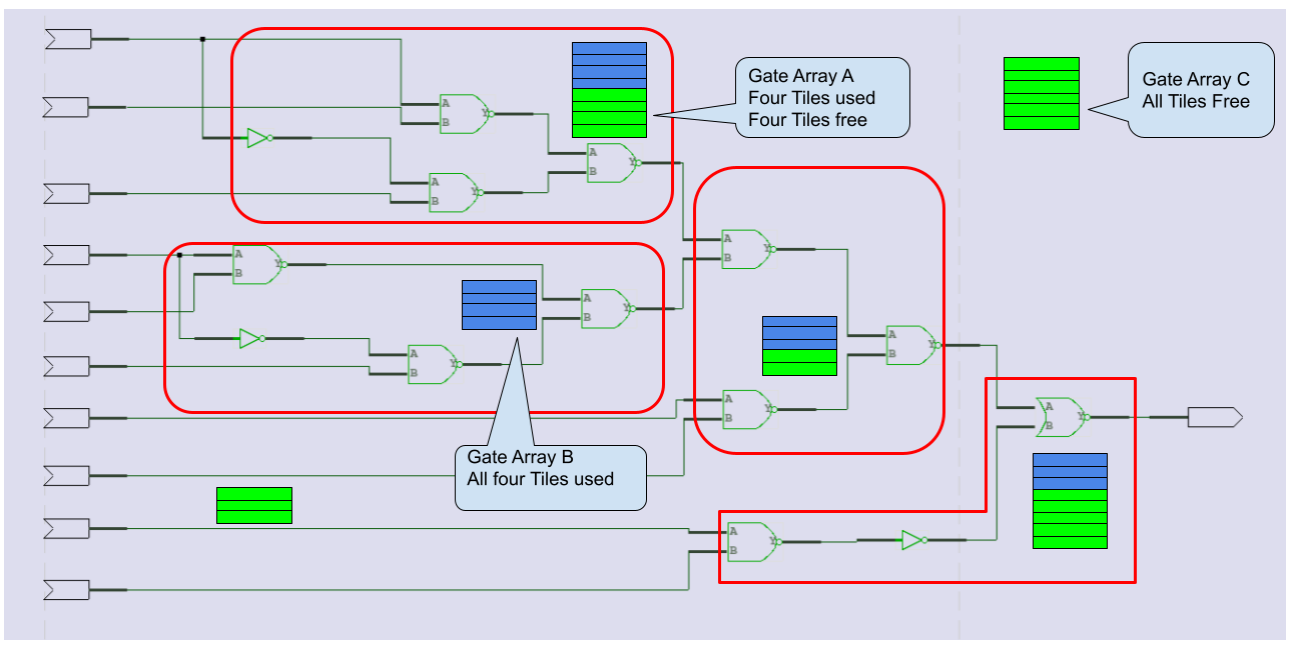

Figure 14: Gate Array Spare Cells Mapping to Functional Cells

After mapping and swapping, as shown in Figure 14, some gate array spare cells have portion of tiles being used by several functional cells. When saving ECO results, these gate array spare cells should have type changed, for instance, gate array A should have type changed from GFILL8 to GFILL4. Those used up gate array spare cells should be deleted, for instance gate array B has type GFILL4 and all four tiles are used.

The mapped gate array functional cells should be moved to the locations of the corresponding gate array spare cells, and horizontal location X should be adjusted according to the starting tile location. As shown in Figure 13, GINVD1 instance should be moved to (Xg+TW, Yg) and GBUFFD1 instance should be moved to (Xg+TW*6, Yg).

GOF writes out ECO verilog file and Backend tools ECO scripts. In Verilog file, the newly added gate array functional cells have location written in comments. GOF supports Synopsys ICC script and Cadence Encounter script. Both scripts have cell location placement support.

For example, saving the result in ICC TCL script, the cells in Figure 13 have the following commands:

```
size_cell GFILLER_7256 GFILL1 # The original GFILL8 resized
create_cell eco_3821_ubuf GBUFD1
create_cell eco_3821_uan4 GAN4D1
create_cell eco_3821_und2 GND2D1
create_cell eco_3821_uinv GINVD1
set_cell_location -ignore_fixed -coordinates "255.02 413.28" eco_3821_ubuf # Xg+WT*6, Yg
set_cell_location -ignore_fixed -coordinates "254.42 413.28" eco_3821_uan4 # Xg+WT*3, Yg
set_cell_location -ignore_fixed -coordinates "254.22 413.28" eco_3821_und2 # Xg+WT*2, Yg
set_cell_location -ignore_fixed -coordinates "254.02 413.28" eco_3821_uinv # Xg+WT*1, Yg
```

Encounter script format:

```
ecoChangeCell -inst GFILLER_7256 -cell GFILL1 # The original GFILL8 resized
```



```
addInst -loc 255.02 413.28 -inst eco_3821_ubuf -cell GBUFD1 # Xg+WT*6, Yg
addInst -loc 254.42 413.28 -inst eco_3821_uan4 -cell GAN4D1 # Xg+WT*3, Yg
addInst -loc 254.22 413.28 -inst eco_3821_und2 -cell GND2D1 # Xg+WT*2, Yg
addInst -loc 254.02 413.28 -inst eco_3821_uinv -cell GINVD1 # Xg+WT*1, Yg
```

Note:Tile size assumed to be 0.20 X 0.22; GFILL8 location (Xg, Yg)=(253.82, 413.28)

2.3.4 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.lib'
- Other Verilog libraries if '.lib' files can't cover
- Implementation Netlist
- Reference Netlist
- LEF files
- DEF (Design Exchange Format) files

2.3.5 Steps to do gate array spare cells ECO

A typical process for gate array spare cells ECO:

- Change RTL modules
- Synthesize the new RTL to get Reference Netlist
- Create a GofCall ECO script:
  - Load Standard Cell libraries and Verilog libraries
  - Load Reference Netlist and Implementation Netlist
  - Add 'fix_design' command
  - Load LEF files
  - Load DEF files
  - Extract gate array spare cells and functional cells by 'get_spare_cells'
  - Run 'map_spare_cells' to convert the patch all gate array functional cells type and map to optimal gate array spare cells
  - Report ECO status and write out ECO results
- Run the above script

2.3.6 Example GofCall script for gate array cells ECO flow

The GofCall script has the exact same syntax of Perl script. It runs the exported commands that access the netlist database and modify the netlist.

```
# GofCall ECO script, run_gate_array_cells_eco_example.pl
use strict;
undo_eco;# Discard previous ECO operations
# Setup ECO name
setup_eco("eco_gate_array_example" );
read_library("tsmc.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in Implementation Netlist Which is under ECO
set_top("topmod");# Set the top module that ECO is working on
set_ignore_output("scan_out*");
set_pin_constant("scan_enable", 0);
set_pin_constant("scan_mode", 0);
fix_design;
save_session("current_eco_name"); # Save a session for future restoration
read_lef("tsmc.lef"); # Read LEF
read_def("topmod.def"); # Read Design Exchange Format file
# Specify gate array cells, spare and functional
# set dont_use command can be used to exclude some gate array cells
get_spare_cells("-gate_array", "G*", "-gate_array_filler", "GFILL*"); # Gate array cells extraction
map_spare_cells();
report_eco(); # ECO report
check_design();# Check design
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
write_tcl("eco_icc.tcl");# Write out TCL script for ICC
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.3.7 TCL output file format

```
current_instance
current_instance up_ma/utx_afe_if
create_net eco_ganet_wire270244
create_net eco_ganet_gofrev_net_on19915
create_net eco_ganet_wire270246
create_net eco_ganet_gofrev_net_on19913
disconnect_net [get_nets n_223] [get_pins slow_cnt_reg_1/D]
connect_net [get_nets eco_ganet_wire270244] [get_pins slow_cnt_reg_1/D]
create_cell eco_gacell_gofrev_inst_19916 GND2D1
create_cell eco_gacell_gofrev_inst_19914 GOR2D1
create_cell eco_gacell_inst270245 GMX2D1
create_cell eco_gacell_gofrev_inst_19912 GAN2D1
connect_net [get_nets eco_ganet_gofrev_net_on19915] [get_pins eco_gacell_gofrev_inst_19916/A]
connect_net [get_nets slow_cnt_2] [get_pins eco_gacell_gofrev_inst_19916/B]
connect_net [get_nets eco_ganet_wire270244] [get_pins eco_gacell_gofrev_inst_19916/Y]
connect_net [get_nets eco_ganet_gofrev_net_on19913] [get_pins eco_gacell_gofrev_inst_19914/A]
connect_net [get_nets xg_prbs_0] [get_pins eco_gacell_gofrev_inst_19914/B]
connect_net [get_nets eco_ganet_gofrev_net_on19915] [get_pins eco_gacell_gofrev_inst_19914/Y]
connect_net [get_nets n_221] [get_pins eco_gacell_inst270245/A]
connect_net [get_nets slow_cnt_0] [get_pins eco_gacell_inst270245/B]
connect_net [get_nets fast_data_9] [get_pins eco_gacell_inst270245/S0]
connect_net [get_nets eco_ganet_wire270246] [get_pins eco_gacell_inst270245/Y]
connect_net [get_nets n_223] [get_pins eco_gacell_gofrev_inst_19912/A]
connect_net [get_nets eco_ganet_wire270246] [get_pins eco_gacell_gofrev_inst_19912/B]
connect_net [get_nets eco_ganet_gofrev_net_on19913] [get_pins eco_gacell_gofrev_inst_19912/Y]
set_cell_location -ignore_fixed -coordinates "253.84 413.28" eco_gacell_inst270245
set_cell_location -ignore_fixed -coordinates "250.42 390.60" eco_gacell_gofrev_inst_19912
set_cell_location -ignore_fixed -coordinates "288.04 497.70" eco_gacell_gofrev_inst_19914
set_cell_location -ignore_fixed -coordinates "204.25 267.12" eco_gacell_gofrev_inst_19916
current_instance
size_cell FILLER_impl0_7256 GFILL3
size_cell FILLER_impl1_30700 GFILL2
current_instance
remove_cell FILLER_impl1_20939
remove_cell FILLER_impl1_40219
```

2.3.8 Run and debug

The GofCall Script can be run by '-run' option.

gof -run run_gate_array_cells_eco_example.pl

User can insert 'die' command to let GOF stop in some point and do interactive debugs when "GOF > " shell appears. GUI mode can be enabled by run 'start_gui' command.

Check Run and debug GofCall script section for more detail

2.4 Script Mode Full Layers Manual ECO Flow

2.4.1 Overview

In many cases, the ECO operations are well known by users. They can be inserting buffers to a 128bits bus, or adding isolation AND gates to all outputs of a module. In these cases, manual ECO by scripts is more efficient and resource saving.

GOF exports many APIs for ECO operations in GofCall script.

2.4.2 Steps to do Manual ECO In Scripts

A typical situation for a Manual ECO:

- Run LEC on modified RTL to Implementation Netlist
- Collect the failing points in the above run
- Create a GofCall ECO script:
  - Define ECO name in 'setup_eco'
  - Load Standard Cell libraries and Verilog libraries
  - Load Implementation Netlist
  - Locate ECO point
  - Use ECO APIs to fix the logic
  - Report ECO status and write out ECO results
- Run the script

2.4.3 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.lib'
- Other Verilog libraries
- Implementation Netlist
- ECO locations

2.4.4 ECO APIs list

These APIs change Implementation Netlist

```
buffer: ECO command. Buffer high fanout ECO nets
change_gate: ECO command. Modify an instance in ECO
change_net: ECO command. Change a existing net's driver
change_pin: ECO command. Modify pin connection in ECO
change_port: ECO command. Change an output port's driver
del_gate: ECO command. Delete gate
del_net: ECO command. Delete net
del_port: ECO command. Delete port
new_gate: ECO command. Create new gate
new_net: ECO command. Create a new net
new_port: ECO command. Create a new port for the current top level module
```

For the full list of the APIs, user can type 'help' in 'GOF >' shell.

For the individual API, type 'help api_name' . For example:

```
GOF > help new_port
Help for new_port
new_port: ECO command. Create a new port for the current top level module
ECO command. Create a new port for the current top level module
Usage: new_port($name, @options);
$name: Port name
@options:
-input: New an input port
-output: New an output port
-inout: New an inout port
Note: The port name has to be pure words or with bus bit, like, abc[0], abc[1]
Examples:
new_port('prop_control_en', '-input'); # create an input port naming prop_control_en
new_port('prop_state[2]', '-output'); # create an output port with bus bit prop_state[2]
new_port('prop_state[3]', '-output'); # create an output port with bus bit prop_state[3]
```

2.4.5 Example GofCall script for Manual ECO

```
# GofCall ECO script, run_example.pl use strict;
undo_eco;# Discard previous ECO operations
setup_eco("eco_example");# Setup ECO name
read_library("tsmc.lib");# Read in standard library
read_design("-ref", "reference.gv");# Read in Reference Netlist
read_design("-imp", "implementation.gv");# Read in implementation Netlist Which is under ECO
set_top("topmod");# Set the scope to the module that ECO is working on
# The following API adds a mux in flop 'state_reg_0_' D input pin,
# and connect up the original connection to pin 'A',
# pin 'B' connect to net 'next_state[7]', and pin 'S' to net 'sel_mode'
# the net can be replaced by format of 'instance/pin' , E.G. '.S(state_reg_2_/Q)'
change_pin("state_reg_0_/D", "MX2X4", "", ".A(-),.B(next_state[7]),.S0(sel_mode)");
report_eco();
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.4.6 Run and debug

Check Run and debug GofCall script section for more detail

2.4.7 Handle repetitive work

A Perl 'for' or 'foreach' loop can handle repetitive work efficiently. For example, to add a 'AND' isolation gate for every output port of a module.

```
# GofCall ECO script, add_ands.pl
use strict;
undo_eco; # Discard previous ECO operations
setup_eco("eco_example"); # Setup ECO name
read_library("tsmc.lib"); # Read in standard library
read_design("-ref", "reference.gv"); # Read in Reference Netlist
read_design("-imp", "implementation.gv"); # Read in implementation Netlist which is under ECO
set_top("topmod"); # Set the top module that ECO is working on
my @ports = get_ports("-output"); # Get all output ports of module 'topmod'
# For each output port of 'topmod', insert an 'AND' gate to enable it only when 'enable_out' is high
my $cnt = 0;
foreach my $port (@ports){
  change_port($port, "AND2X2", "eco_add_and_$cnt", "-,enable_out");
  $cnt++;
}
report_eco();
write_verilog("eco_verilog.v");# Write out ECO result in Verilog
exit;# Exit when the ECO is done, comment it out to go to interactive mode when 'GOF >' appears
```

2.4.8 Special character

The special character '-' is used to represent existing connection. For example

```
change_pin("U0/A", "BUFFX1", "eco_buf","-");
```

A buffer is inserted into A pin of instance U0. The old existing net drives the new buffer now.

The special character '.' is used in ECO new instance name if the new instance needs to be in the same hierarchy as the ECO spot.

```
change_pin("u_qcif/u_num2/u_spare1/B", "AOI21X2",".", "net1,net2,net3");
```

If the instance is empty, GOF creates 'AOI21X2' in the current top level. With ".", GOF creates 'AOI21X2' new instance in

hierarchy "u_qcif/u_num2/u_spare1".

2.5 Script Mode Metal Only Manual ECO Flow

2.5.1 Overview

In Manual Metal Only ECO, any new added gates are automatically mapped to spare gate instances by 'map_spare_cells' command. A Design Exchange Format file has to be loaded for the tool to find optimal spare instances. If the file is not present, the mapping is skipped.

2.5.2 Files and data requirements

- Standard library (Synopsys Liberty) files with extension '.'lib'
- Other Verilog libraries
- Implementation Netlist
- DEF (Design Exchange Format) file. If it is not loaded, GOF won't map the spare gate type cells to the exact spare instances
- Spare gates pattern. It is in 'hierarchical_instance/leaf_instance' format. It has wild card '*' to match the spare gates in Implementation Netlist
- Spare gates list file. If several users work on the same Implementation Netlist, the initial spare gates list file should be generated only once. And a new spare gates list file should be created every time ECO is done
- ECO locations

2.5.3 Example GofCall script for Manual Metal Only ECO

```
# Manual Metal Only ECO, manual_metal_eco.pl
use strict;
undo_eco;
setup_eco("metal_eco0123");
set_log_file("metal_eco0123.log");
read_library("/prj/lib/tsmc40.lib");
read_design("-imp", "/prj/netlist/imp_net.v");
set_top("mtop");

new_port("nout7", "-output");# Create a new port 'nout7'
# Place the port to 60000, 1000000. It's approximate position, the main purpose is for
# spare instances selection
place_port("nout7", 60000, 100000);
new_port("nout8", "-output");# Create another port
place_port("nout8", 120000, 81000);
# 'nout8' is driven by an invert first, and the invert's input is driven by pin 'cmpmod/rego/QN'
change_port("nout8", "INV_X1M", "", "cmpmod/rego/QN");
# Drive the 'nout7' by 'INV_X1M' and leave the input unconnected, but the mapped
# spare instance name is returned.
my $inst = change_port("nout7", "INV_X1M", "", "");
# Drive the new instance's input by a flop, and specify the flop's connection in the 4th argument
change_pin("$inst/A", "SDFFRPQ_X4M", "", \n".CK(cmpmod/rego/CK),.D(cmpmod/rego/QN),.R(1'b0),.SE(1'b0),.SI(1'b0)");

read_def("/prj/def/imp_net.def");
get_spare_cells("Spare_*/*_SPARE_GATE*");
# Before mapping to spare gates, set a large number in buffer distance, so that GOF does not
# add buffers for long connections.
set_buffer_distance(9999999);
# The following 'map_spare_cells' command maps the three new ECO instances to the optimal
# spare instances.
map_spare_cells;

report_eco;
write_verilog("imp_eco0123.v");
```

2.5.4 Run and debug

The GofCall Script can be run by '-run' option.

gof -run manual_metal_eco.pl

Check Run and debug GofCall script section for more detail

2.6 GUI Mode Full Layers ECO Flow

2.6.1 Overview

The following paragraph demonstrates how to insert buffers and inverters into a circuit in GUI mode.

2.6.2 Start up GOF in GUI Mode

Start up Gates On the Fly by the command line

```
gof -lib t65nm.lib -lib io.lib netlist_port.v
```

For detail usage, visit this link

https://nandigits.com/usage.htm

In GofViewer netlist window, press ctrl-g or menu commands->'Launch GofTrace with gate'. Fill in the instance name that needs ECO.

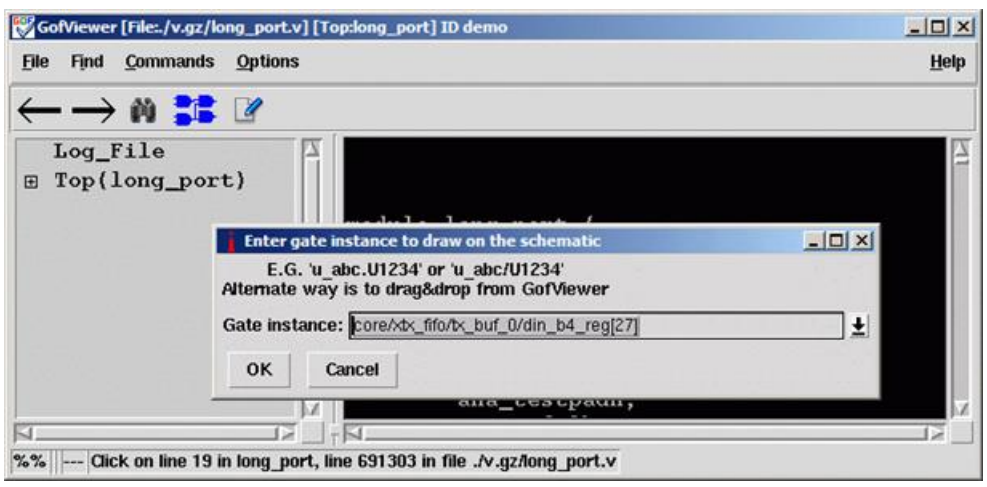

Figure 15: Load gate to schematic

2.6.3 Create Partial Schematic

In GofTrace schematic window, use mouse middle button to expand the schematic. In this case, pin D of the flop should be inserted an invert.

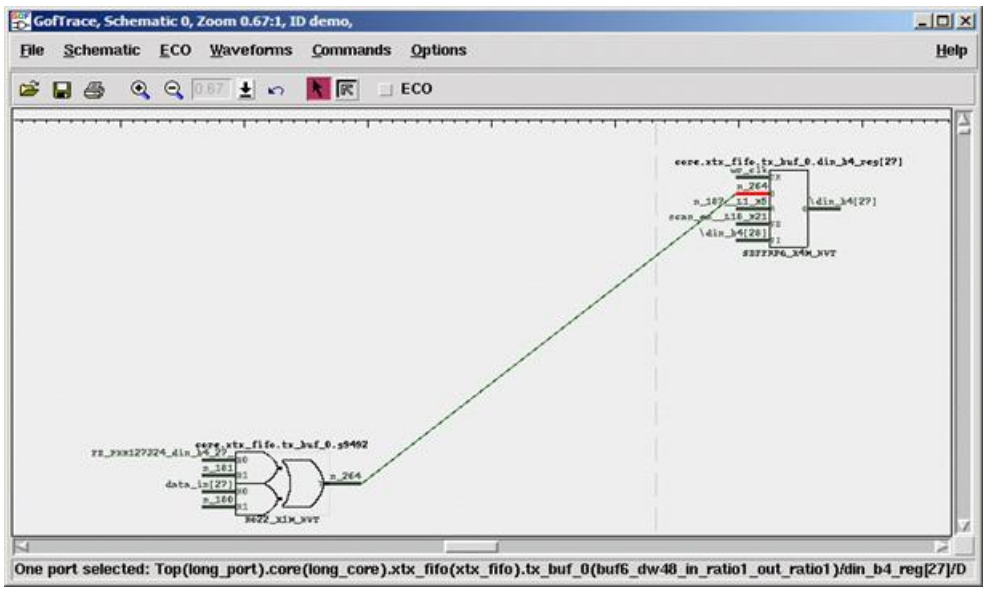

Figure 16: Partial Schematic for GUI ECO

2.6.4 Do ECO on schematic

Check ECO button to enable ECO mode

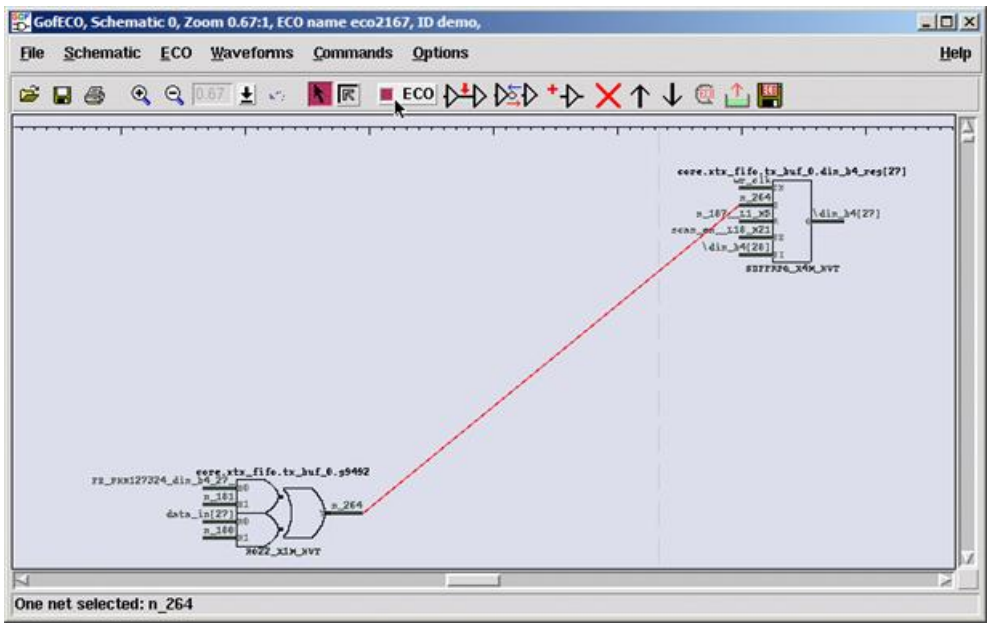

Figure 17: Schematic in ECO Mode

Press mouse-left-button on the wire to select it. Click ECO button 'Insert gates into connections', select the right invert in the gate type selection window.

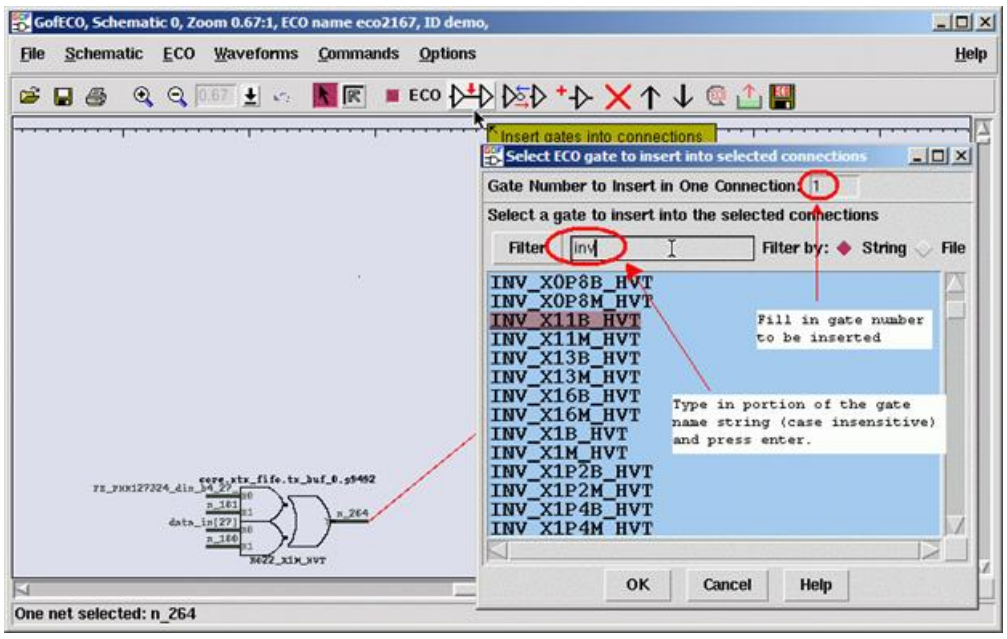

Figure 18: Select Gate in GUI ECO

In 'Pin Connections' setup window, use default 'Complete Loop' option, so that the gate can be inserted in the net.

Figure 19: New Cell Pin Connection Selections

Click OK and the invert is inserted.

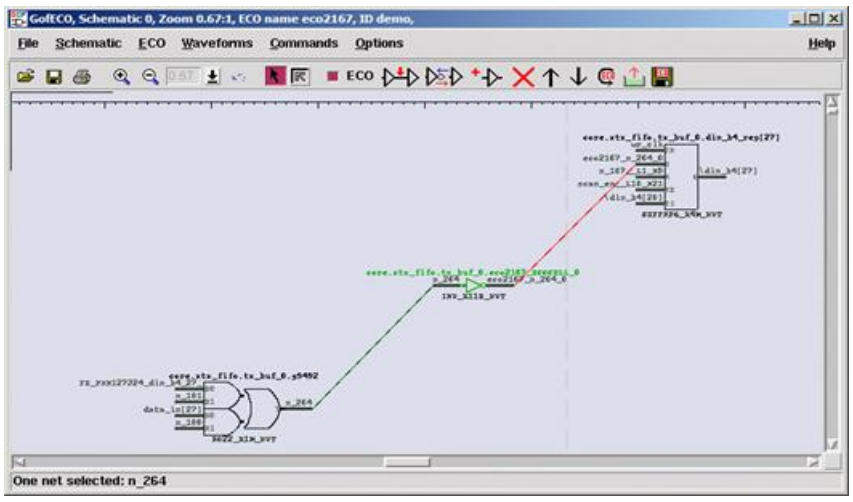

Figure 20: Manual ECO with New Gate Inserted

2.6.5 Save ECO

Press ECO button 'Save ECO result to file'. And select the format to be saved. The supported formats include verilog netlist, SOC Encounter ECO script, GofCall Script, TCL script and DCShell script.

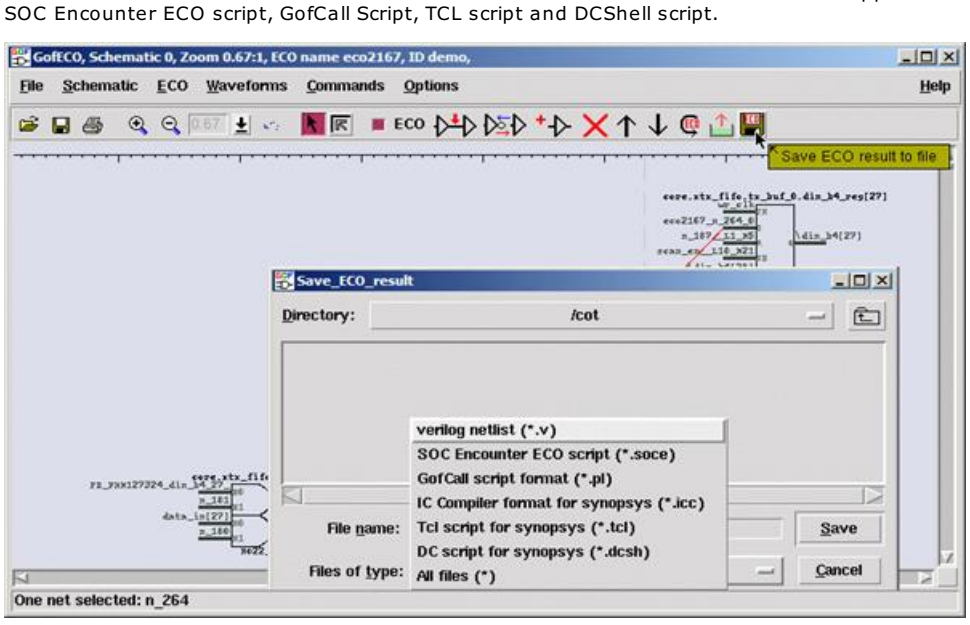

Figure 21: Save ECO in GUI Mode

2.7 GUI Mode Metal Only ECO Flow

2.7.1 Overview

Metal ECO can only use existing spare gates on the silicon. Gates On the Fly controls how to use these spare gates.

2.7.2 Methods for Metal Only ECO

Four methods are supported in Metal Only ECO:

1. User can add any type of gates and let the tool map to the spare type gates, Place and Route tool should map the spare type gates to the exact spare gate instances
2. User can add any type of gates and let the tool map to the exact spare gate instances
3. User can add only spare type gates and let the tool map to the exact spare gate instances
4. User can pick the exact spare gate instances, and connect and disconnect up the instances in ECO

Note: 'Spare type gate' refers to the gate type, 'INVX2', 'NAND2X2'. 'Exact spare gate instance' refers to the spare instances in the design, E.G. 'spare1/spare_invx2'

2.7.3 Setup and use cases

The detail setup for four method can be found in GofECO Metal Only ECO. Use cases can be found in online document.

3 Script Mode Detail Features

3.1 GofCall Script

GofCall is the Perl Script Interface which can access internal exported APIs.

3.1.1 Get Help

- Type 'help' in interactive shell 'GOF >' to list all APIs
- Type 'help set_*' to list all APIs matching 'set_', like 'set_top', 'set_invert'
- Type 'help individual-API' to list detail of the API
- Visit https://nandigits.com/gof_manual.php for online manual

3.1.2 Feature list

- Compatible with Perl
- Automatic ECO with fix_design/fix_modules APIs
- Rich ECO APIs to do manual ECO
- ECO operations are reversible
- ECO result can be loaded in schematic on the fly
- Integrated commands to browser netlist and check design status

3.1.3 API list

```
buffer: ECO command. Buffer high fanout ECO nets
change_gate: ECO command. Modify an instance in ECO
change_net: ECO command. Change a existing net's driver
change_pin: ECO command. Modify pin connection in ECO
change_port: ECO command. Change an output port's driver
check_design: Check if the netlist status
compare: Logic equivalence check on output port and register input pins
compare_nets: Check equivalence of two nets in the reference and implementation netlist
convert_gated_clocks: ECO command. Convert gated clocks to MUX logic.
current_design: Set the current top level module
current_instance: Set the current instance
del_gate: ECO command. Delete gate
```



```
del_net: ECO command. Delete net
del_port: ECO command. Delete port
exist_inst: Check if an instance exists
exist_wire: Check if a wire exists
fix_design: ECO command. Fix the whole design in global mode
fix_logic: ECO command. Fix listed points
fix_modules: ECO command. Fix modules listed in the arguments
flatten_modules: Flatten hierarchical modules in reference netlist
get_cell_info: Get information of a module or instance
get_cells: Get all cells in the current module or sub-modules
get_conns: Get connections of net or pin in the top level module
get_coord: Get an instance's coordination
get_definition: Get instantiation of instance
get_driver: Get the driver of a net or pin
get_drivers: Get the drivers of a net or pin
get_instance: Get instance in the top level module
get_instances: Get all hierarchical instances in the top level module
get_leaf_pin_dir: Get leaf cell pin's direction input/output/inout
get_leafs_count: Get all leaf cells name and count in the top level module
get_lib_cells: Get leaf gates in libraries
get_loads: Get loads of net in the top level module
get_logic_cone: Get logic cone of nets or pins
get_modules: Get all hierarchical modules under current module
get_net_of: Get net name connecting to a pin
get_nets: Get nets that matching pattern
get_path: Get current hierarchical path
get_pins: Get pins of instance or module
get_ports: Get all ports in the current top level module
get_ref: Get the reference of the instance
get_resolved: Resolve the relative path to module and leaf item
get_roots: Get root designs name
get_scan_flop: Get scan flop for the non scan flop
get_spare_cells: ECO command. Get spare cells
gexit: Exit the GofCall interactive mode
gprint: Print the message and save to log file
is_leaf: Check if a module or instance is leaf cell
is_scan_flop: Check if an instance is scan flop
is_seq: Check if an instance is a specific sequential cell
map_spare_cells: ECO command. Map all new created cells to spare cells
new_gate: ECO command. Create new gate
new_net: ECO command. Create a new net
new_port: ECO command. Create a new port for the current top level module
place_gate: ECO command. Place gate position
place_port: ECO command. Place port position
pop_top: Pop out the saved top level module from the stack and discard the current setting
push_top: Set the current top level module and push the previous setting to stack
read_def: Read DEF file
read_design: Read verilog netlist
read_file: Read timing violation report file
read_lef: Read LEF file
read_library: Read standard library or verilog library files
read_svf: Read Synopsys SVF text files
rename_net: ECO command. Rename a net name
report_eco: Report ECO
report_spares: Report Spare cells
restore_session: Restore ECO session
run: Run GofCall script
run_lec: Run Logic Equivalence Check on Implementation Netlist and Reference Netlist
save_session: Save ECO session
sch: Launch schematic to verify ECO
set_auto_fix_floating: ECO setting. Enable automatic fixing floating input ports after fix_modules
set_bfix: Enable or disable BFIX features
set_blackbox: Set Blackbox on Modules
set_bound_opti: Enable or disable boundary optimization check. Enabled by default
set_buffer: Set buffer type. The tool automatically picks one if the command is not called
set_buffer_distance: Set distance limit for inserting buffer
set_constraints: Set constraints for map_spare_cells command
set_cutpoint_thresh: Set Cutpoint Threshold
set_cutpoint_ultra: Enable or disable CutPoint Ultra Effort
set_dont_fix_modules: Set dont fix property on modules
set_dont_use: Set dont use property on library cells
set_eco_effort: ECO setting. Set ECO effort
set_equal: ECO setting. Set two points to be equivalent in the Reference and Implementation Netlists
set_error_out: Set error out setting
set_exit_on_error: Whether the tool should exit when the script runs into an error
set_exit_on_warning: Whether the tool should exit when the script runs into a warning
set_floating_as_zero: Set floating net as constant zero
set_high_effort: Set high ECO effort on modules
set_ignore_instance: ECO setting. Set ignored sequential or blackbox instances in ECO
set_ignore_network: ECO setting. Set ignore network in ECO
set_ignore_output: ECO setting. Set ignore output ports
set_inside_mod: Set fix scope inside the current module
set_inst: Set the current instance
set_inv: ECO setting. Set two points to be inverted in the Reference and Implementation Netlists
set_invert: Set invert type. The tool automatically picks one if the command is not called
set_keep_format: Set keep format of the original verilog when ECO is done
set_keep_tree: Set keeping buffer tree
set_keypoints_rep_in_ref: ECO setting. Replace keypoints naming in Reference Netlist.
set_leaf: Set a hierarchical module to be leaf. Useful to stub hierarchical instances
set_log_file: Set log file name
set_low_effort: Set low ECO effort to speed up ECO process
set_mapped_point: ECO setting. Set two points mapped in Reference and Implementation Netlists
set_mapping_method: LEC setting. Detecting flop phase inversion.
set_max_lines: Set max output lines
set_max_loop: Setup max loop
set_mod2mod: Set reference module mapping to implementation module
set_mu: MU configuration
set_multibit_blasting: Set blasting on multibit flops.
set_net_constant: Set net to a constant value
set_noexact_pin_match: ECO setting. Don't match some special pins
set_phase_adjust_en: Enable phase adjusting
set_phase_inv: ECO setting. Set flops invert phase in the Reference and Implementation Netlists
set_pin_constant: Set pin to a constant value
set_power: Set power pins connections for leaf cell
set_preserve: Set preserve property on instances. The tool does not remove them in ECO
set_quiet: Run script in quiet mode
set_recovery_distance: Set distance limit for gates recovery in ECO
set_remove_undsc_in_ref: ECO setting. Remove last '_' in flop instance in Reference Netlist
set_tiehi_net: Set tiehi net name
set_tielo_net: Set tielo net name
set_top: Set the current top level module
set_tree: Set the current tree
set_user_match: Set match between multi-bit flops to multi-bit flops
set_verbose: Run script in verbose mode
setup_eco: ECO command. Setup ECO
source: Run GofCall script
start_gui: Start GUI windows
stitch_scan_chain: ECO command. Stitch scan chain
suppress_warnings: Suppress warning messages
swap_inst: ECO command. Swap two instances with same input/output pins.
undo_eco: ECO command. Undo eco operations
write_dcsh: ECO command. Write ECO result in Design Compiler dcsh script format
write_perl: ECO command. Write GofCall ECO script compatible with Perl
write_soce: ECO command. Write ECO result in Cadence SOC Encounter script format
```



```
write_spare_file: ECO command. Write spare cells list to a file
write_tcl: ECO command. Write ECO result in Design Compiler tcl script format
write_verilog: ECO command. Write ECOed netlist to a verilog file
```

3.1.4 API grouping

3.1.4.1 Netlist Browse APIs

One key element to do efficient manual ECO is to isolate the ECO spots quickly. The following APIs are for fast Netlist Browsing.

```
get_cells: Get all cells in the current module or sub-modules
get_conns: Get connections of net or pin in the top level module
get_driver: Get the driver of a net or pin
get_drivers: Get the drivers of a net or pin
get_instance: Get instance in the top level module
get_instances: Get all hierarchical instances in the top level module
get_lib_cells: Get leaf gates in libraries
get_loads: Get loads of net in the top level module
get_modules: Get all hierarchical modules under current module
get_net_of: Get net name connecting to a pin
get_nets: Get nets that matching pattern
get_pins: Get pins of instance or module
get_ports: Get all ports in the current top level module
get_ref: Get the reference of the instance
```

For example, to get data pins of flops in one module. The script can use these browse APIs

```
my @flop_data_pins;
set_top("module_name");
my @flops = get_cells("-type", "ff");
foreach my $flop (@flops){
  my @dpins = get_pins("-data", $flop);
  push @flop_data_pins, $dpins;
}
```

After run the script, @flop_data_pins have all data pins of all flops in the module.

3.1.4.2 Automatic ECO APIs

These APIs are for Automatic ECO

```
fix_design: ECO command. Fix the whole design in global mode
fix_modules: ECO command. Fix modules listed in the arguments
fix_logic: ECO command. Fix listed points
map_spare_cells: ECO command. Map all new created cells to spare cells
```

Combining netlist browsing APIs, users can come up a short script to do complicated changes.

For example, to fix all modules named "tx_machine_*"

```
my @modules = get_modules("-hier", "tx_machine_*");
fix_modules(@modules);
```

3.1.4.3 File IO APIs

These APIs are for reading/writing files.

```
read_def: Read DEF file
read_design: Read verilog netlist
read_file: Read timing violation report file
read_lef: Read LEF file
read_library: Read standard library or verilog library files
restore_session: Restore ECO session
save_session: Save ECO session
write_dcsh: ECO command. Write ECO result in Design Compiler dcsh script format
write_perl: ECO command. Write GofCall ECO script compatible with Perl
write_soce: ECO command. Write ECO result in Cadence SOC Encounter script format
write_spare_file: ECO command. Write spare cells list to a file
write_tcl: ECO command. Write ECO result in Design Compiler tcl script format
write_verilog: ECO command. Write ECOed netlist to a verilog file
```

3.1.4.4 Manual ECO APIs

These are APIs for Manual ECO.

```
buffer: ECO command. Buffer high fanout ECO nets
change_gate: ECO command. Modify an instance in ECO
change_net: ECO command. Change a existing net's driver
change_pin: ECO command. Modify pin connection in ECO
change_port: ECO command. Change an output port's driver
del_gate: ECO command. Delete gate
del_net: ECO command. Delete net
del_port: ECO command. Delete port
new_gate: ECO command. Create new gate
new_net: ECO command. Create a new net
new_port: ECO command. Create a new port for the current top level module
```

Combining netlist browsing APIs, a short GofCall script can do very efficient ECOs.

For example, to add isolation cells for all output ports of a module.

```
set_top("module_name");
my @out_ports = get_ports("-output");
foreach my $out (@out_ports){
  change_port($out, "AND2X2", "", "-,net_iso_enable");
}
```

3.1.5 API usage

For detail of APIs visit Appendix A

3.2 String Handling In Script Mode

3.2.1 Single quote and double quote

Any string in GofCall script for module/instance/wire/pin/port should be enclosed by single quote or double quote. When a Perl variable is used, double quote should be used

```
my $inst = "state_reg_0";
change_pin("inst/D","1'b0");
```

3.2.2 Instance and net with backslash

Instance with backslash should be either put in single quote and with a space in the end.


```
change_pin('\u_abc/u_def/state_reg[0] /RN', "1'b0");
```

Net name with backslash should keep the backslash and space. For example

```
DFFQ_X4M \u_abc/u_def/state_reg[0] (.D(\u_abc/u_def/net123 ), .Q(\u_abc/u_def/state[0] ));
```

The net '\u_abc/u_def/net123 ' should have backslash and space kept in API, for example:

```
change_net("\u_abc/u_def/net123 ", "INVX1", "", "-");
```

3.3 Run and debug GofCall script

3.3.1 Command line

In Linux Shell, the GofCall Script can be run by '-run' option.

gof -run run_example.pl

3.3.2 GOF Shell

If '-run' option is present in the command line, and 'exit' or 'gexit' is not in the script, or GOF meets error when executing the script, GOF goes into interactive mode with GOF shell 'GOF >'.

```
GofCall, Netlist Processing Script APIs, Interactive Mode
Run 'start_gui' to launch GUI window
Run 'help' to list API calls
GOF >
```

Individual command can be executed in GOF shell. The command can be in nested mode

```
GOF > set_top(get_ref("u_rxbuf"))
```

3.3.3 Run in GUI mode

GofCall scripts can be run in GUI window. In GofViewer, click Menu Commands->'Launch GofCall Script Interface' to launch GofCall GUI window.

Type 'help' in the shell entry for help information. Scripts can be run by 'run' command in the shell entry

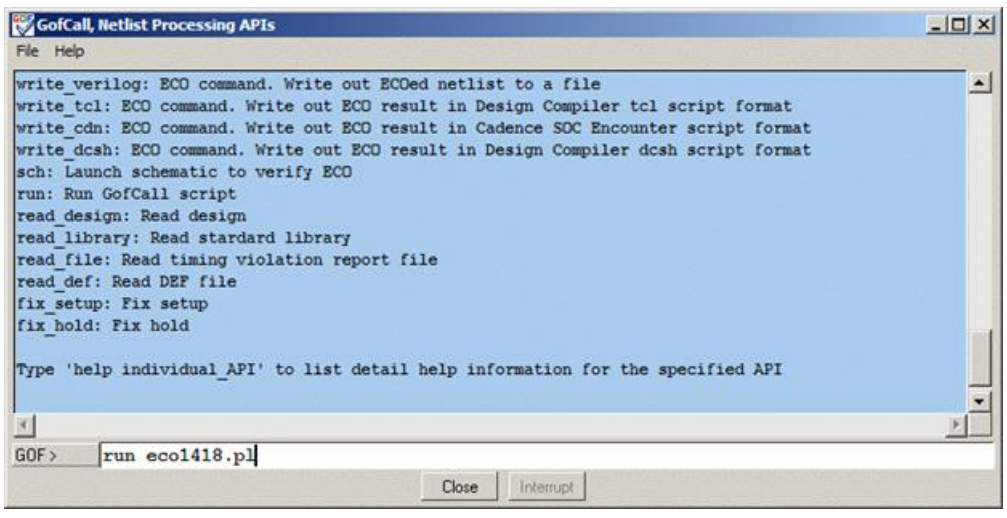

Figure 22: GofCall window

3.3.4 Fast schematic launch

In GOF shell, GUI windows can be launched by 'start_gui' or 'sch' commands.

'start_gui' launches netlist view window first and user can bring up schematic window from netlist view window.

'sch' command only launches schematic window, and it doesn't enable netlist view window. So it has fast turnaround in GUI interactive debug.

For example,

After the following command is done,

```
change_pin("u_top/u_core/u_regmod/state_reg/D", "XOR2X2", "", "-,new_enable");
```

Run 'sch' in 'GOF >'

```
GOF > sch("u_top/u_core/u_regmod/state_reg")
```

The instance is loaded into a schematic and user can click on the instance's pins to trace fanin/fanout on the schematic to see if the ECO is done as expected.

3.3.5 Break points for debug

'sch' fast schematic launch command can be used as break points for debug. For example, 'sch' commands are inserted in GofCall script, when the tool runs to the point, a schematic is launched.

```
…
setup_eco("eco_3821");
set_log_file("t_eco_3821.log");
read_library("art.m.simple.lib");
read_design("-imp", "./cdir/imp_name.v");
change_pin("state_reg_0_/D", "MX2X4", "eco_inst_1", ".A(-),.B(next_state[7]),.S0(sel_mode)");
sch("state_reg_0_");
…
```

On the schematic, user can use mouse-middle-button clicking on the pin 'D' to see if the ECO is done as expected.

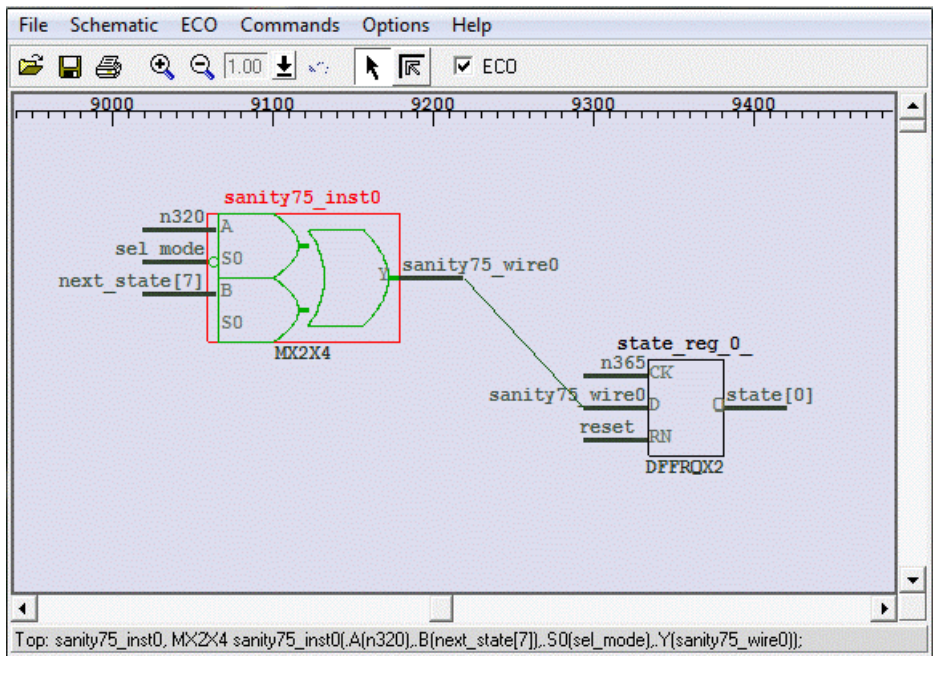

Figure 23: Launch schematic at break point

Note: 'ECO' check-button is enabled automatically, since there is ECO having been done.
To compare with the logic before ECO, launch a new schematic by menu Schematic->'New Schematic'. On the new schematic, press 'ctrl-g' or by menu Schematic->'Load Gate' to load in the flop under ECO.

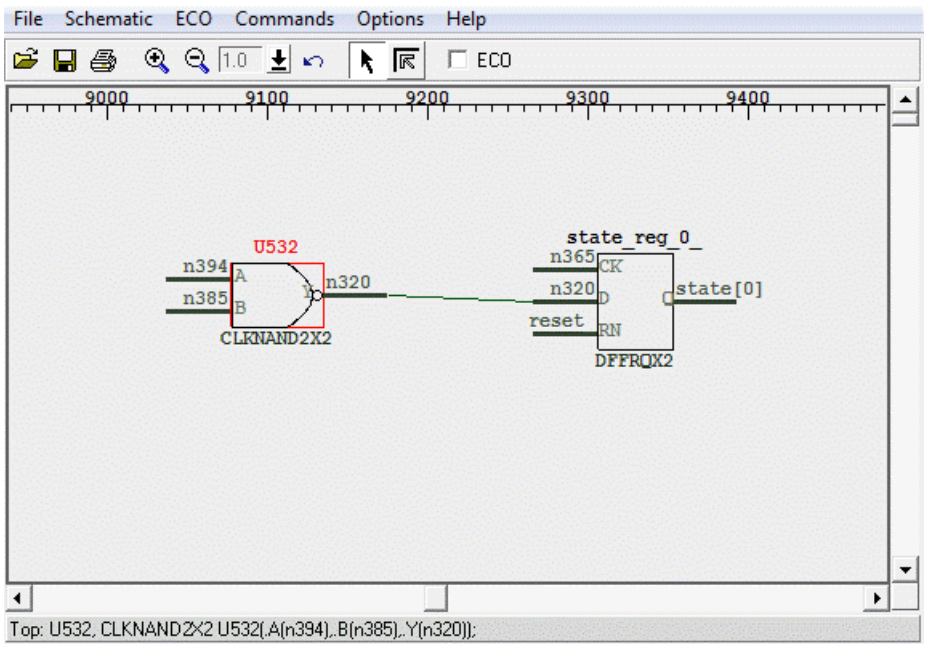

Figure 24: Launch schematic before ECO

Note: 'ECO' check-button is un-checked.

3.4 Typical Manual ECO operations

3.4.1 Insert gate to port

Both input port and output port have the same operation

3.4.1.1 Insert an invert to input port

```
# Insert to input port 'in_enable'
change_port("in_enable", "INVX1", "inst_name", "-"); # 'inst_name' can be empty
```

3.4.1.2 Insert to output port

```
# Insert AND2X1 to output port 'out_enable', one pin connects to the original driver,
# the other pin is driven by 'scan_mode'
change_port("out_enable", "AND2X1", "", "scan_mode,-");
```

3.4.1.3 Insert inverts to multiple ports

```
# Find all ports matching string "abcde" and insert invert to each port
my @ports = get_ports("*abcde*");
foreach my $port (@ports){
  change_port($port, "INVX1", "", "-");
}
```

3.4.2 Insert gate to register instance pin

3.4.2.1 Insert invert to flop data pin

```
# Insert an invert to 'D' pin of flop 'abc_reg'
change_pin("abc_reg/D", "INVX1", "", "-");
```

3.4.2.2 Insert invert to flop output pin

```
# Insert an invert to 'Q' pin of flop 'abc_reg'
change_pin("abc_reg/Q", "INVX1", "", "-");
```

3.4.2.3 Insert MUX to data pin of multiple flops

```
# Find all flops matching string "cnt_reg" and insert MUX to 'D' pin of each flop,
# so that each flop is preset to 'preset_val' when 'preset_enable' is high
my @flops = get_cells("*cnt_reg*");
foreach my $flop (@flops){
  change_pin("$flop/D", "MUX2X2", "", ".A(-),.B(preset_val),.S(preset_enable)");
}
```

3.4.3 Change flops to other type

3.4.3.1 Change non-reset flop type to resettable flop


```
# Find all flops matching string "cnt_reg" and change each flop to resettable flop
my @flops = get_cells("*cnt_reg*");
foreach my $flop (@flops){
  change_gate($flop, "DFFRQX2", ".RD(reset_n)");
}
```

3.4.4 Insert gate to hierarchical instance pin

3.4.4.1 Insert inverts to hierarchical instance pins

```
# Find all instances matching "tx_mac" in module "abc_mod" and insert invert to 'loop_en' pin
set_top("abc_mod");
my @insts = get_instances("*tx_mac*");
foreach my $inst (@insts){
  change_pin("$inst/loop_en", "INVX1", "", "-");
}
```

3.4.4.2 Insert AND to hierarchical instance pins

```
# Find all instances matching "tx_mac" in module "abc_mod" and
# AND all output pins with "power_on" signal.
set_top("abc_mod");
my @insts = get_instances("*tx_mac*");
foreach my $inst (@insts){
  my @pins = get_pins("-output", $inst);
  foreach my $pin (@pins){
    my $net = get_net_of($pin); # Only add AND to those out pins driving nets
    if($net){
      change_pin($pin, "AND2X2", "", ".A(-),.B(power_on)");
    }
  }
}
```

4 GUI Mode Detail Features

4.1 GofViewer

When GOF is run without '-run' or '-shell' option, it goes into GUI mode.

gof -lib t65nm.lib -lib io.liblong_port.v

GofViewer is the first window after GOF starts up GUI interface.

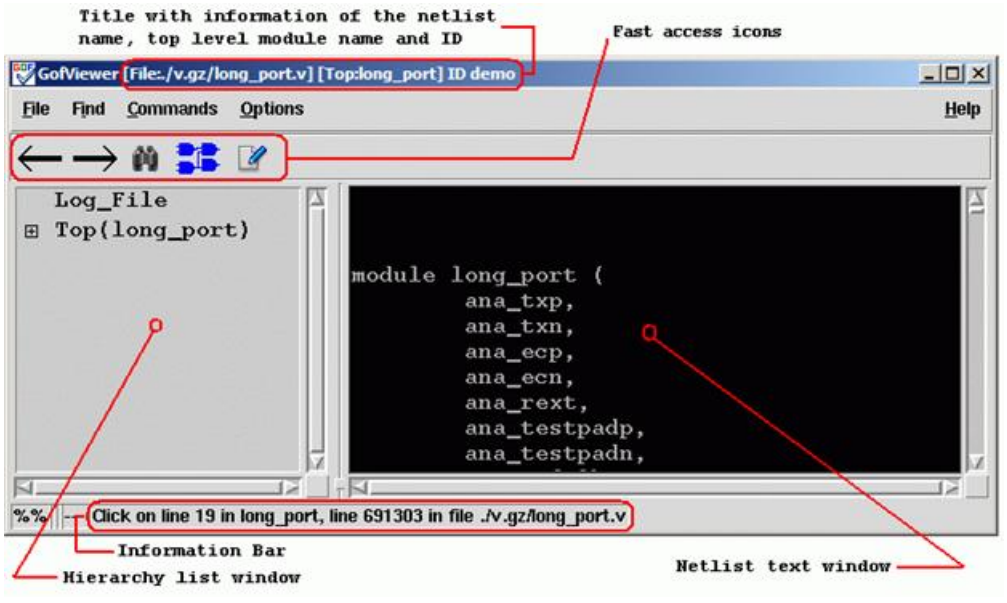

Figure 25: GofViewer Window

4.1.1 Log Window

If there errors or warnings in loading the database, Log Window pops up

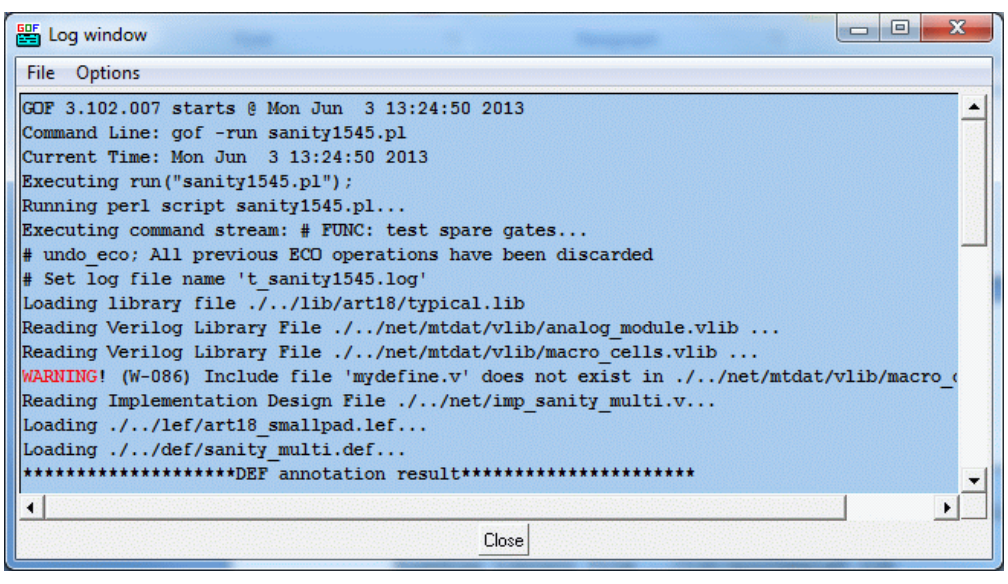

Figure 26: Log Window

4.1.2 File Menu

4.1.2.1 Load Design

Users can input netlist files and design files through the Load Design command.

4.1.2.2 Reload Design

If any netlist file or design file has been updated during GOF session, this command can be used to reload the design.

4.1.2.3 Open Other Netlist

This command loads another netlist file to create a new hierarchical tree. The hierarchy tree is listed in the hierarchy list

window. The command is useful when users want to draw circuits from different netlist file on the same schematic which is good for logic comparison in netlist debug scenario such as LEC failures analysis.

4.1.2.4 Open Log Window

The command opens log file in a text window.

4.1.2.5 Exit

Exit command.

4.1.3 Find Menu

4.1.3.1 Search

This command searches for the matching string in the netlist text window.

4.1.3.2 Goto Line Number

GOF loads only one module in the netlist text window when the netlist file is hierarchical with multiple modules. The command loads the corresponding module into the text window and highlight the line with the specific number in the netlist file.

4.1.3.3 Report Area

This command reports the design area. The command requires standard library files to be loaded which include leaf cell area information.

4.1.3.4 Report Leakage

This command reports the leakage power in the design. Same as the Report Area command it requires standard libraries.

4.1.3.5 Report Leaf Cells

This command reports the leaf cell type and numbers in the design.

4.1.3.6 Report Submodules

This command reports the hierarchical sub-modules in the design.

4.1.3.7 Statistic of Current Design

This command reports the statistic of the current design. It pops up an option window for interactivity from users.

4.1.3.8 List Library

The command lists the libraries and leaf cells in each library.

4.1.3.9 List Context for Leaf Cell

This command pops up an entry window for users to input leaf cell name string, wild card can be accepted. All leaf cells matching the string is listed. If there is only one cell matched, the detail property is listed.

4.1.4 Commands Menu

4.1.4.1 Launch GofTrace Schematic

This command launches GofTrace Schematic, if any instance or net string is highlighted in the netlist window, the instance or the driver of the net is drawn on the schematic. Otherwise, the schematic is empty.

4.1.4.2 Launch GofTrace with Gate

This command pops up an entry window for users to input a string to load a specific instance. For example, ‘u_abc/U123’. Click ‘OK’ button on the pop window, GofTrace Schematic is launched.

4.1.4.3 Launch Layout Viewer

This command launches Layout Viewer window, if any instance or net string is highlighted in the netlist window, the instance or the driver of the net is highlighted on the Layout Viewer window. The command requires that physical files to be loaded. Both def and lef files should be loaded before launching Layout Viewer, otherwise a warning window pops up for the missing physical files.

4.1.4.4 Launch GofCall Script Interface

This command launches GofCall window to run scripts or other interactive command.

4.1.4.5 Spare Cells

This command group processes Spare cells in metal ECO. Warning! GUI metal ECO is used for visually checking the possibility of metal ECO. The script mode metal ECO is recommended.

- Create Spare Cells File

  This command extracts spare cells from netlist file. A pop window appears for spare gates pattern. The default is ‘spare_*/*’.

Figure 27: Spare Cell Pattern

Click 'OK' to extract spare instances from the netlist, and a pop text window appears to list all spare gate instances. Save the list to a spare list file for later usage.

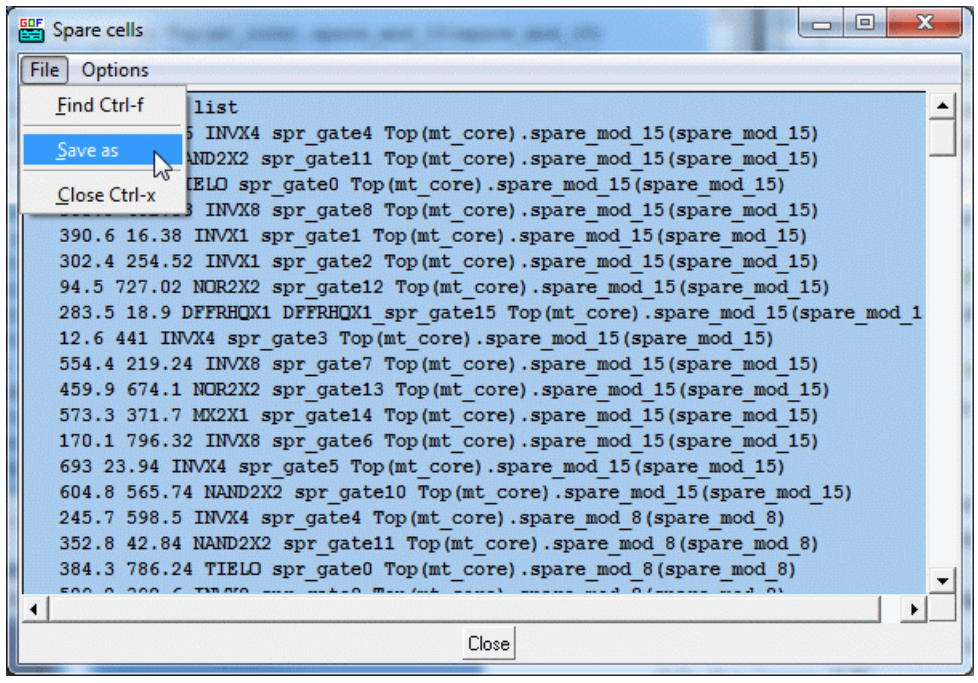

Figure 28: Spare cell list

- Load Spare Cells File

  This command loads in the spare cells file created by the above command.

4.1.5 Options Menu

4.1.5.1 Hierarchy Window Font

Check GofViewer for the hierarchy list window position

- Increase Font Size

  Increase font size in the hierarchy list window.

- Decrease Font Size

  Decrease font size in the hierarchy list window.

4.1.5.2 Netlist Window Font

Netlist window locates in the right side of GofViewer window. Check GofViewer for the netlist text window position

- Increase Font Size

  Increase the font size in netlist window.

- Decrease Font Size

  Decrease font size in netlist window.

4.1.5.3 Dump Waveform Restore File

An option window pops up for users to choose which dump restore file to be saved. It's useful for netlist simulation debug. When one format box is checked in pop menu, 'Write Waveform Restore File' item is presented on the top when one net is selected in the netlist window.

4.1.5.4 Setup

Integration of various setup information.

4.1.6 Help Menu

4.1.6.1 General

General help information.

4.1.6.2 About

About Gates On the Fly.

4.1.6.3 nandigits.com/gof_manual.php

Visit the website for this manual.

4.1.6.4 Read Ethernet Mac Address

Read out MAC address. When users decide to purchase licenses or ask for evaluation licenses, MAC address is required to generate GOF licenses.

4.1.7 Keyboard Shortcuts

4.1.7.1 Access Menu

Press key 'Alt' and underlined letter in menu.

4.1.7.2 Functions access

- Ctrl-a: Select all text lines
- Ctrl-c: Copy the marked (highlighted) string
- Ctrl-v: Paste the content in clipboard
- Ctrl-d: Trace driver of the marked net
- Ctrl-f : Search function
- Ctrl-g: Load instance to schematic
- Ctrl-s: Emacs style search forward
- Ctrl-r: Emacs style search backward
- Ctrl-w: Write to waveform dump restore file
- Ctrl-x: Exit the current window

4.1.8 Selection Status

Click mouse-left-button on netlist text window, the object which can be net, instance or module under the cursor is highlighted. Netlist window pop menu has different content according to the selection status. Pressing keys Ctrl-a can have all content in the netlist window selected. Press mouse-left-button and don't release, move mouse down to select multiple lines.

4.1.9 Netlist Window Pop Menu

Click mouse-right-button and release, a pop menu appears under the cursor. The menu content varies with the selection status in the netlist window.

4.1.9.1 Search

Search for a string in the netlist window. Keyboard shortcut is Ctrl-f.

4.1.9.2 Copy Selected to

Copy the selected object (net or instance) to new schematic window or existing schematic window.

- Schematic New

  Copy the selected object to a new schematic.

- Schematic #number

  Copy the selected object to an existing schematic window.

4.1.9.3 Driver of the selected net

Trace to the driver of the selected net. The netlist window shows the instance that drives the net and mark the driven net.

4.1.9.4 List Connectivity of the selected net

Pop up a window to list the connectivity of the selected net.

4.1.9.5 List Fanin EndPoints

Pop up a window to list the fanin endpoints including flops and input ports that drive the selected net.

4.1.9.6 List Fanout EndPoints

Pop up a window to list the fanout endpoints including flops and output ports that are driven by the selected net.

4.1.9.7 Parent Module

Go to the definition location of the parent module calling the current module. It's only active in sub-modules, not in root top level module.

4.1.9.8 List Context

List the context of the selected object which can be net, instance or module.

4.1.10 Hierarchy Window Pop Menu

Click mouse-right-button and release, a pop menu appears under the cursor. The menu content varies with the selection status in the hierarchy window.

4.1.10.1 Show Definition

Open the module content and display it in the netlist window.

4.1.10.2 Show Calling

Open the parent module and highlight the instantiation location.

4.1.10.3 Report Area of the selected design

See Report Area

4.1.10.4 Report Leakage of the selected design

See Report Leakage

4.1.10.5 Report Leaf Cells of the selected design

See Report Leaf Cells

4.1.10.6 Report Submodules of the selected design

See Report Submodules

4.1.10.7 Statistic of the selected design

See Statistic of Current Design

4.1.10.8 Edit Module of the selected design

Edit the module by using edit tool defined in menu Options->Setup->Misc->'Edit tool'. It asks for a directory for storing temporary files.

4.1.10.9 Save Module of the selected design

After editing, the edited modules can be saved into a file.

4.1.10.10 Goto Line Number

4.2 GofTrace

GofTrace is an incremental schematic engine. Users control how to expand the schematic by clicking the input/output pins of gates with mouse-middle-button. Users can adjust the positions of the gates on the schematic any time by mouse-left-button.

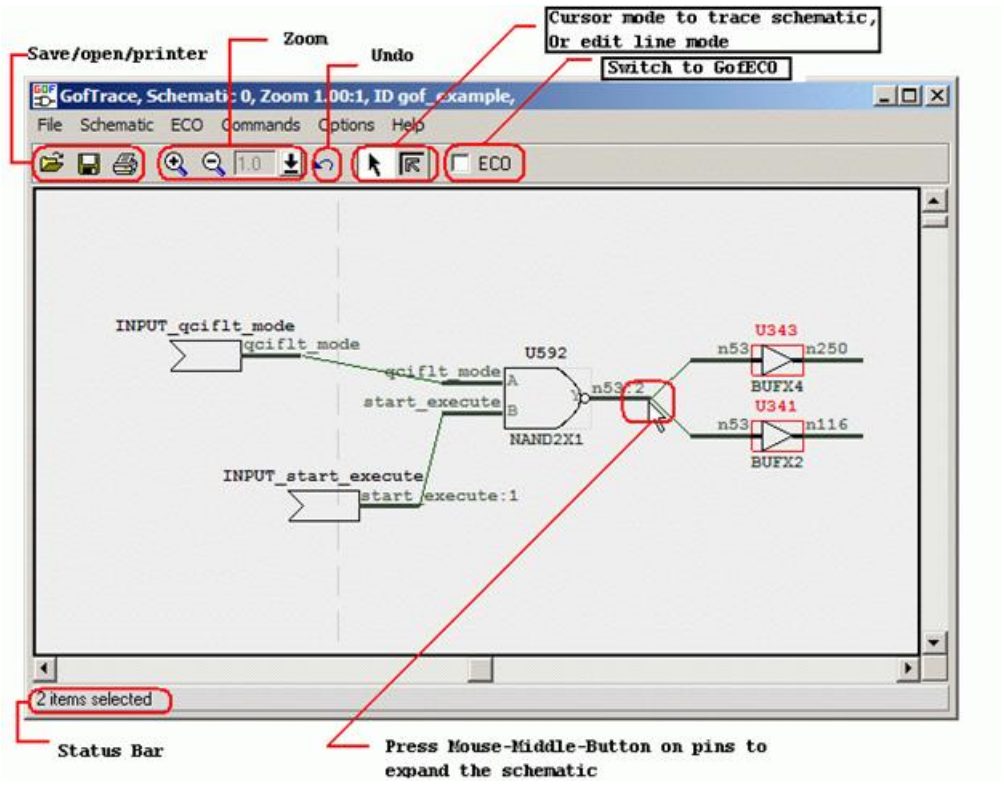

Figure 29: GofTrace Window

4.2.1 Mouse buttons usage

4.2.1.1 Mouse Left Button

Mouse left button is used to select object. Click on any object, it is highlighted to indicated being selected. Press 'ctrl' key and click on objects to select multiple objects. Press mouse left button and move the mouse to select multiple objects at one time.

4.2.1.2 Mouse Middle Button

Mouse middle button is used to trace the schematic. Click on input/output pins to expand the schematic. It used to do drag-drop function as well. In ECO mode, it's used to connect floating input pin to existing nets.

4.2.1.3 Mouse Right Button

Mouse right button is to popup menu.

4.2.2 File Menu

4.2.2.1 Save

Save the schematic to a file for future usage. The saved file has extension '.st' which can only be used by GOF in 'Open' schematic command shown below.

4.2.2.2 Open

Open schematic stored by Save command above.

4.2.2.3 Print

Print schematic to a printer or file. Printer Page Setup window pops up for the print scope setup. In Windows platform, users can select one of the printers configured in the system. In Linux platform, make sure 'lpr' command works.

4.2.2.4 Exit

Exit GofTrace window.

4.2.3 Schematic Menu

4.2.3.1 New Schematic

This command launches a new GofTrace schematic window.

4.2.3.2 List Gate

This command pops up a window for user to enter a string into the entry to find the matching instances. It accepts wildcards in both hierarchy name and instance name. For example, there are four hierarchical instances u_lane0, u_lane1, u_lane_2, u_lane3, each instance has spare modules with instance naming 'u_spare*', and in each spare module AND gate has instance naming '*AND*'. In order to find all spare AND gates, one can enter a string 'u_lane*/u_spare*/*AND*'.

4.2.3.3 Load Gate

This command pops up a window for user to enter a string into the entry to load the matching instances onto the schematic. Same as 'List Gate' command above, it accepts wildcards. However, the total number of gates drown on the schematic should not exceed the threshold defined in Menu Options->Setup->Misc->'Gates number limit'.

4.2.3.4 Load Gate Driving Net

This command pops up a window for user to enter a string into the entry as the net name. The tool finds the driver of the net and draw the driver on the schematic.

4.2.3.5 List Selected Instances

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list all the selected instances' full hierarchical names in a pop window.

4.2.3.6 List Selected Wires

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list all the selected wires' full hierarchical names in a pop window.

4.2.3.7 List Selected Modules

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list all the selected gates' module name in a pop window.

4.2.3.8 List Selected Instances Definitions

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list all the selected instances' full definitions in a pop window.

4.2.3.9 List Selected Gates Types

Use mouse-left-button to select a bunch of objects (Instances or wires) on the schematic. Click this command to list logic type numbers of all the selected gates in pop window. For example, 'AND' gate has type 'and', inverter has type 'not'. The pop window can have information such as "Type 'not' has 11".

4.2.3.10 Zoom In

This command can zoom in the schematic view. The maximum zoom in ratio is 100%. Keyboard shortcut for this command is key '+'.

4.2.3.11 Zoom Out

This command can zoom out the schematic view. The minimum zoom out ratio is 13%. Key board shortcut for this command is key '-'.

4.2.3.12 Zoom to

This command can directly select zoom ratio, the valid values are 100%, 67%, 44%, 30%, 20% and 13%.

4.2.3.13 Find Gates on Schematic

This command pops up a window for users to enter a string to find the matching instances on the schematic. It matches portion of the full name. For example, 'U' matches 'U0', 'U1' and 'U222'.

4.2.3.14 Find Nets on Schematic

This command pops up a window for users to enter a string to find the matching wires on the schematic. It matches portion of the full name. For example, 'Net0' matches 'Net0', 'Net011' and 'Net023'.

4.2.3.15 Undo Schematic Operations

This command is to undo schematic operations. Keyboard shortcut is Ctrl-z.

4.2.3.16 Place and Route

This command group is to automatically place the gates on the schematic and automatically route the wires.

- Auto Place and Route

  This command is to do both placement and routing automatically.

- Auto Place

  This command is to do automatic placement only.

- Auto Route

This command is to do automatic routing only.

- Reset Route

  This command is to reset all existing routes, all routed wires become straight.

4.2.3.17 Create PS/PDF File

This command is to create Postscript file or PDF file for the current view of the schematic. In Windows platform, only Postscript is support. On Linux platform both Postscript and PDF are supported.

4.2.4 Commands Menu

4.2.4.1 View Gates in Layout

This command launches layout viewer window. If some gates and wires are selected on the schematic, they are highlighted on the layout viewer. It requires DEF and LEF physical design files to be loaded.

4.2.4.2 Load Layout Files

This command is to load layout physical design files. They include DEF and LEF files. The command can be run several times to load the physical design file one by one. DEF and LEF files can be loaded by command line with -def and -lef options. Or they can be read in by API 'read_def' and 'read_lef' in GofCall script.

4.2.4.3 Launch GofCall Script Interface

This command launches GofCall Script Interface window.

4.2.4.4 Spare Cells

This command group handles spare cells in automatic metal ECO flow.

- Create Spare Cells File

  This command creates spare cells file.

- Load Spare Cells File

  This command loads the spare cells file created by the command above.

4.2.5 Options Menu

4.2.5.1 Increase Font Size

This command increases the font size on the schematic.

4.2.5.2 Decrease Font Size

This command decreases the font size on the schematic.

4.2.5.3 Show Port

This option makes port name visible.

4.2.5.4 Show Wire

This option makes wire name visible.

4.2.5.5 Show Title

This option makes gate title visible.

4.2.5.6 Show Type

This option makes gate type visible.

4.2.5.7 Show Connections

This option makes wires visible.

4.2.5.8 Show Comment

This option makes comments visible.

4.2.5.9 Dump Waveform Restore File

This command pops up a window to setup simulation waveform restore file. Four waveform restore file formats are supported.

- SimVision Restore File
- ModelSim Restore File
- Verdi Restore File
- GtkWave Restore file

If one or more of the formats are selected, GofViewer and GofTrace pop menus have 'Write Selected Nets to the Waveform Restore File' as the first item, when a net is selected.

4.2.5.10 Save String to Clipboard

This option enables saving string to clipboard when a wire or instance name is clicked by mouse-left-button.

4.2.5.11 Cursor Mode

This is normal mode of the schematic tracing.

4.2.5.12 Line Edit Mode

This mode sets cursor in editing wire connections mode. Press mouse-left-button on the straight wire connection and move, the line is pulled by the cursor until the mouse button is released.

4.2.5.13 Setup

The command pops up configuration window for the tool setup.

4.2.6 Help Menu

4.2.6.1 General

General help information.

4.2.6.2 About

About Gates On the Fly.

4.2.6.3 nandigits.com/gof_manual.php

Visit the website for the manual.

4.2.7 Keyboard Shortcuts

4.2.7.1 Access Menu

Press key 'Alt' and underlined letter in menu.

4.2.7.2 Functions access

- Ctrl-a: Select every object on the schematic

- Ctrl-c: Copy the selected objects
- Ctrl-v: Paste the content in clipboard copied by Ctrl-c
- Ctrl-g: Load instance to schematic
- Ctrl-w: Write to waveform dump restore file
- Ctrl-x: Exit the current window
- Ctrl-digit: Save the selection location as the digit indication, the schematic view moves the current saved location when the digit is pressed later

4.2.8 Selection Status

Click mouse-left-button on the schematic window, the object which can be net, instance under the cursor is highlighted. GofTrace pop menu has different content according to the selection status. Pressing keys Ctrl-a can have all selected on the schematic. Press mouse-left-button on empty space, and don't release, move mouse down to select multiple objects.

4.2.9 GofTrace Pop Menu

Click mouse-right-button on GofTrace schematic, a menu pops up. The content of the menu varies as the selection status on the schematic.

4.2.9.1 Driver Until Non Buffer

Trace driver of an input pin. If the driver is a buffer or invert, the tracing will continue on the input pin of the buffer or invert, until the driver is non-buffer/invert. The feature can be used to trace the clock tree cells of a flop's clock input.

4.2.9.2 Drivers of Logic Cone

Logic Cone is the logic cluster between flops and ports, as shown in the following figure. Users should select the output flop or its pins to do logic cone extraction.

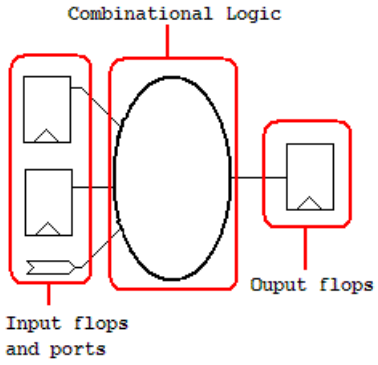

Figure 30: Logic Cone

- On the Schematic

  Draw the whole logic cone on the schematic.

- In Text Mode

  Display the whole logic cone in a pop up text window.

4.2.9.3 Copy Selected to

This command group does interactions between GofTrace windows and LayoutViewer windows.

- Schematic New

  Copy the selected items to a new schematic.

- Schematic Number#

  Copy the selected items to an existing schematic identified by ID Number.

- Layout New

  Copy the selected items to a new launched LayoutViewer window. The selected circuit is marked on the LayoutViewer window.

- Layout Number#

  Copy the selected items to an existing LayoutViewer window indentified by ID Number.

4.2.9.4 Trace Scan Chain

When the selected item is a flop or latch, the command item appears in the pop menu. The command is to trace the scan chain starting from this register's scan input pin or data pin. The scan chain list will be displayed on a popup window.

4.2.9.5 Nets Equivalence Check

The command needs Reference Netlist loaded.

- Reference Netlist can be loaded by '-ref' option in command line. For example,
  ```
  gof -lib tsmc.lib implementation.v -ref reference.v
  ```
- Reference Netlist can be loaded by 'read_design' with '-ref' switch in GofCall script. For example,
  ```
  read_design("-ref", "reference.v");
  ```

Use mouse-left-button to select on a pin in implementation netlist and press 'ctrl' key to click mouse-left-button on the other pin in reference netlist. So that one pin in implementation netlist and the other comparing pin in reference netlist are selected at the same time.

Click mouse-right-button to popup menu and select "Equivalence Check for 'neta' vs 'netb'" command.

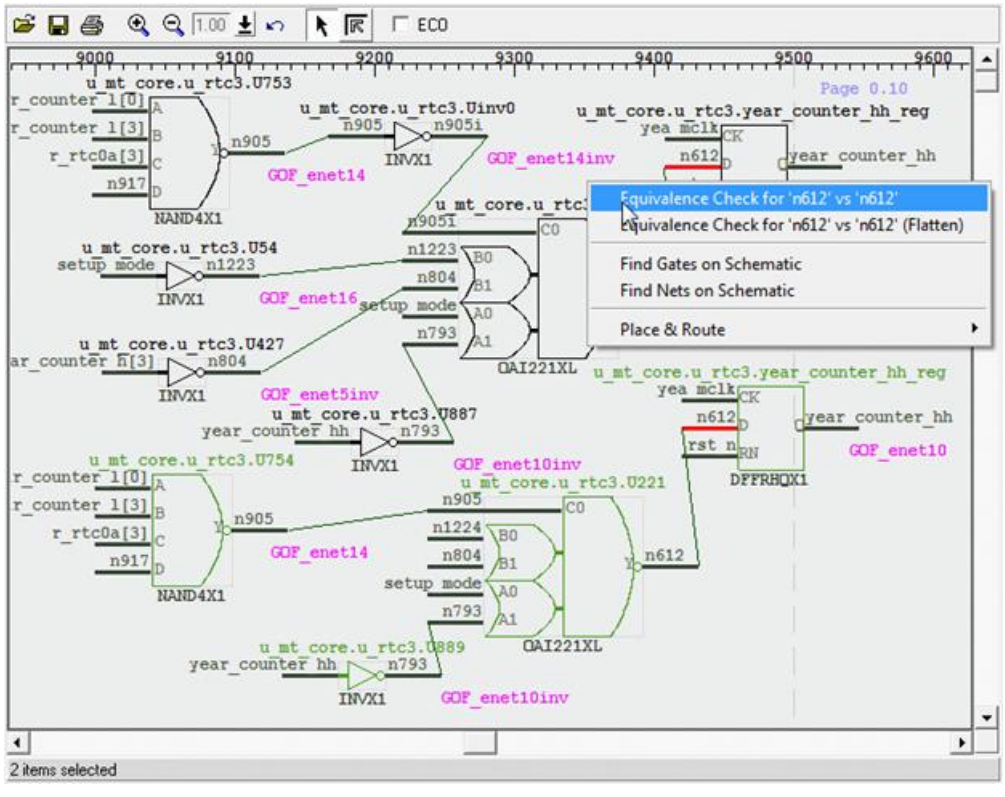

Figure 31: Nets Equivalence Check

When the check is done, a pop window shows if the nets are equivalent.

4.2.9.6 Add Comments

This command adds comments entered by users on the schematic.

4.2.9.7 Find Gates on Schematic

This command pops up a window for users to enter a string to find the matching instances on the schematic. It matches portion of the full name. For example, 'U' matches 'U0', 'U1' and 'U222'.

4.2.9.8 Find Nets on Schematic

This command pops up a window for users to enter a string to find the matching wires on the schematic. It matches portion of the full name. For example, 'Net0' matches 'Net0', 'Net011' and 'Net023'.

4.2.9.9 Place and Route

This command group is to automatically place the gates on the schematic and automatically route the wires.

- Auto Place and Route

  This command is to do both placement and routing automatically.

- Auto Place

  This command is to do automatic placement only.

- Auto Route

  This command is to do automatic routing only.

- Reset Route

  This command is to reset all existing routes, all routed wires become straight.

4.2.9.10 Find selected in GofViewer

This command finds the selected instance back in GofViewer netlist window, and highlights the instance in the netlist window.

4.2.9.11 Edit Gate Display

This command pops up a window for users to add or change comments associated with the gate and change the color of the gate.

4.2.9.12 List Logic for the Selected Leaf Cell

This command pops up a text window to list the logic of the selected leaf cell.

4.2.9.13 List Context for the Selected Leaf Cell

This command pops up a text window to list the library content of the selected leaf cell. The content includes the cell's pin definitions, area and timing.

4.2.9.14 List Definition for the Selected Instance

This command pops up a text window to list the instantiation of the selected instance.

4.2.9.15 Load Instance Similar to the Selected Instance

This command pops up an entry window with the current selected instance name pasted in the entry. So that user can do simple change to load other similar naming style instance onto the schematic.

4.2.9.16 Equivalent Symbol

This command changes the selected gate symbol display to the equivalent symbol according to DeMorgan's Laws. For example, NAND symbol is equivalent to Inputs Inverted OR symbol.

4.2.9.17 Delete

This command deletes the selected objects on the schematic. The object can be gates, wires and comments.

4.3 GofECO

GofECO uses the same window as GofTrace by enable ECO mode. The background color changes to light blue by default. The color can be configured by Menu Setup->GofECO->Color->BackGround. The ECO operation icons appear on the tool bar. GofECO uses the same menus GofTrace uses, besides the contents in ECO menu being activated.

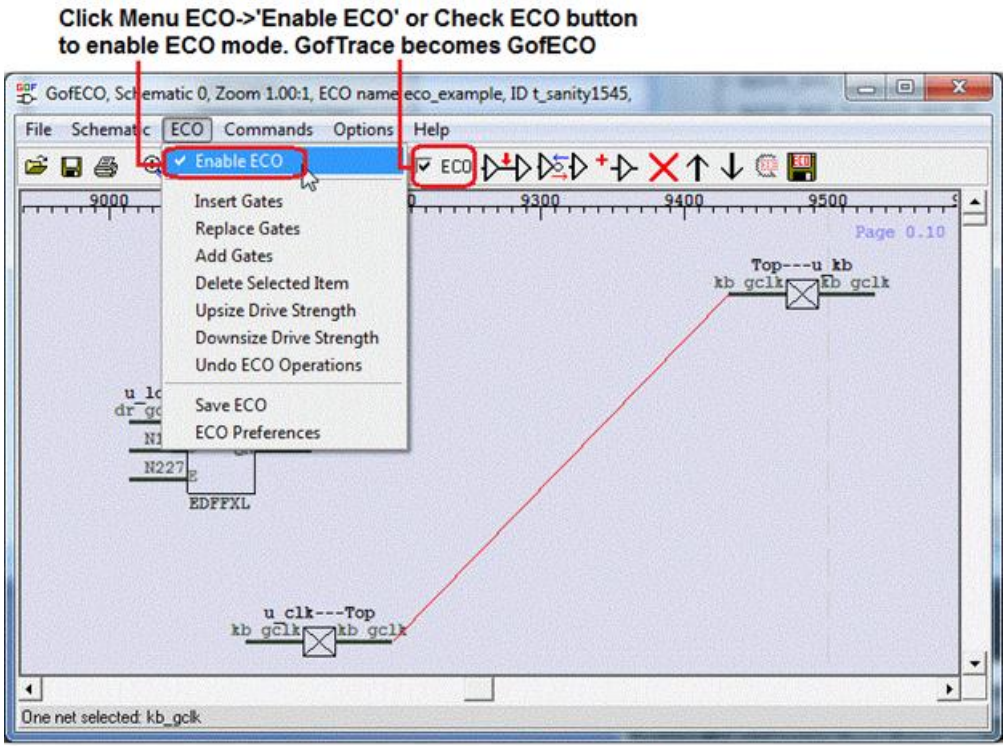

Figure 32: GofECO Window

4.3.1 ECO Menu

4.3.1.1 Enable ECO and ECO Preferences

This option enable ECO mode, GofTrace switches to GofECO. A pop up window appears for inputting ECO setups.

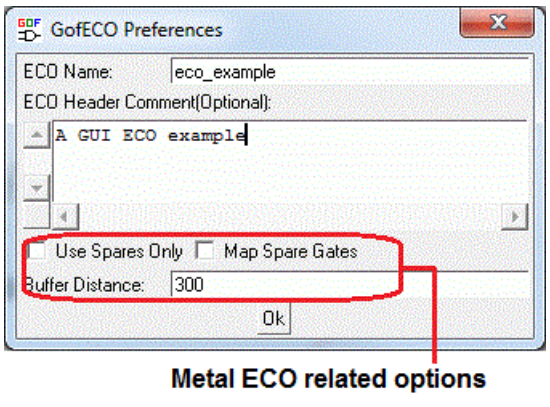

Figure 33: ECO Preferences

- ECO Name should be unique, so that name confliction can be avoided
- ECO Header Comment is optional, which appears at the beginning of ECO netlist file
- Checkbuttons 'Use Spares Only' and 'Map Spare gates' and 'Buffer Distance' entry are for Metal Only ECO. Their usages are:
- Use Spares Only' is to use spare type gates only, a spare gate list file must be loaded with this option enabled.
- Map Spare Gates' is to let the tool mapping any type gates to either the spare type gates or the exact spare instances in the design.
- Buffer Distance' entry is to tell the tool add buffers/repeaters when the connection distance is larger than the limit. Inputting a large number can disable adding buffers. The corresponding script command is 'set_buffer_distance'.

4.3.1.2 Insert Gates

This command inserts gates in the selected wires. It requires one or more wires being selected on the schematic, before inserting gates. A 'Gate selection' window pops up for users to select proper type of gates and gate number. When multiple wires are selected and some wires have the same drivers, users can choose either one gate driving all shared wires or one gate driving each wire. Users are asked to choose the pin connections in 'Specify pin connections' window. The default pin connections setup can be used and users can modify the connections later on the schematic. Read this PDF use case for more detail.

gof_insert_buffers_inverters.pdf

4.3.1.3 Replace Gates

This command replaces the selected gates with a different type of gates. It requires one or more gates being selected on the schematic. If two or more than two gates are selected, they should have the same type. A 'Gate selection' window pops up for users to select proper type of gates to replace the selected ones. Users are asked to choose the pins connections in 'Specify pin connections' window. The default pin connections setup can be used and users can modify the connections later on the schematic.

4.3.1.4 Add Gates

This command adds new ECO gates on the schematic. A 'Gate selection' window pops up for users to select proper type of gate to add onto the schematic. The new ECO gates appear as output driving a new net and input floating. The hierarchy of the gate is undefined. When users connect one of the input pins to another existing gate or connect other gate's floating input to the ECO gate's output pin, the ECO gate gets the same hierarchy as the other gate. Read Add Connection for more detail.

4.3.1.5 Delete Selected Items

This command deletes the selected items. Users would be warned for deleting multiply objects at the same time.

4.3.1.6 Upsize Drive Strength

This command upsizes the selected gate to a higher drive strength gate with the same type. If there is no higher drive strength gate available, users would be warned with a pop up information window.

4.3.1.7 Downsize Drive Strength

This command downsizes the select gate to a lower drive strength gate with the same type. If there is no lower drive strength gate available, users would be warned with a pop up information window.

4.3.1.8 Undo ECO Operations

This command undoes the previous ECO operation, until no more ECO operation is in the pipeline.

4.3.1.9 Add Connection

There is no operation button/icon for Add Connection operation. Adding connection can only be done from a floating input pin to a output pin. User can press mouse-middle-button on a floating input pin, and don't release the mouse. Then move mouse to the destination output pin of the instance that user would like the wire connected to, release the button to make the connection to be created.

4.3.1.10 Save ECO

This command saves ECO result to a file. The supported file formats:

- Verilog netlist
- GofCall Perl Script
- SOC Encounter ECO script
- Tcl script for Synopsys
- DC script for Synopsys

4.3.2 Metal Only ECO

Metal ECO only touches metal layers. Gates On the Fly provides four Metal Only ECO modes by combinations of setting up the options in ECO preference and loading DEF file.

4.3.2.1 Metal ECO, mode 1

User can add any type of gates and let the tool map to the spare type gates, Place and Route tool should map the spare type gates to the exact spare gate instances.

The setup for this mode:

- Spare gate list file should be created and loaded.
- DEF file should NOT be loaded.
- 'Use Spares Only' is NOT checked.
- 'Map Spare Gates' is checked.

4.3.2.2 Metal ECO, mode 2

User can add any type of gates and let the tool map to the exact physically existing spare gate instances.

The setup for this mode:

- Spare gate list file should be created and loaded.
- DEF file should be loaded.
- 'Use Spares Only' is NOT checked.
- 'Map Spare Gates' is checked.

4.3.2.3 Metal ECO, mode 3

User can add only spare type gates and let the tool map to the exact spare gate instances.

The setup for this mode:

- Spare gate list file should be created and loaded.
- DEF file should be loaded.
- 'Use Spares Only' is checked.
- 'Map Spare Gates' is checked.

4.3.2.4 Metal ECO, mode 4

User can pick the exact spare gate instances, and connect and disconnect up the instances in ECO.

The setup for this mode:

- Spare gate list file has no need to be created and loaded.
- DEF file should be loaded.
- 'Use Spares Only' is NOT checked.
- 'Map Spare Gates' is NOT checked.

4.4 LayoutViewer

LayoutViewer window displays partial physical placements. The circuit drawn on the schematic can be highlighted on LayoutViewer. It has full interactivity with GofTrace. It requires physical design files including DEF and LEF files to be loaded.

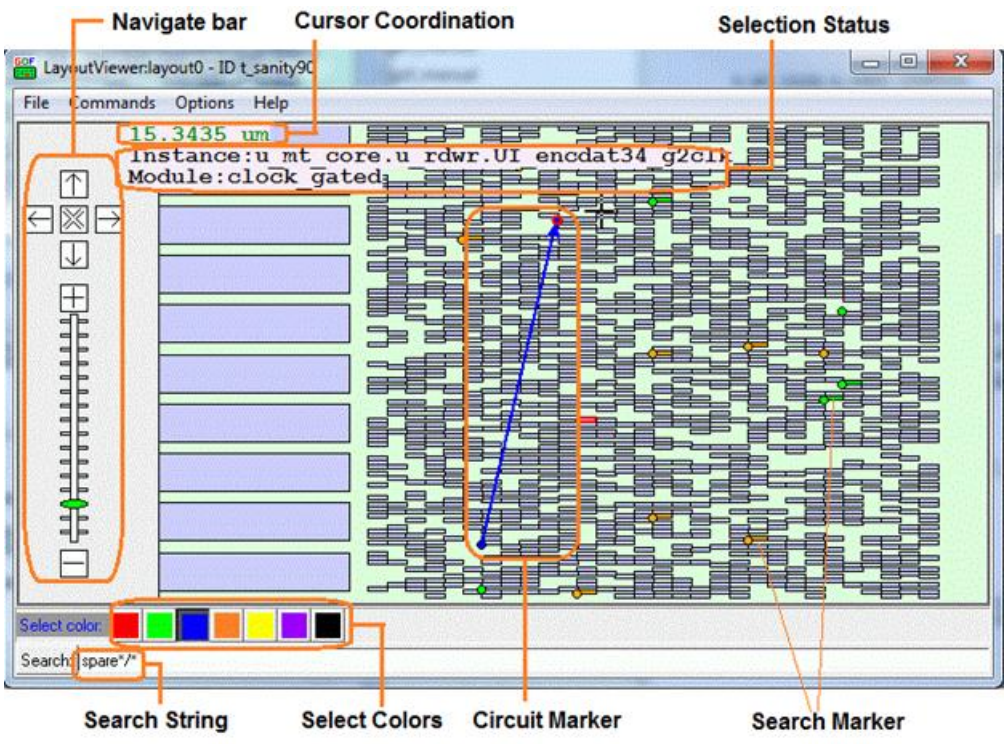

Figure 34: LayoutViewer Window

4.4.1 File Menu

4.4.1.1 Capture in PDF

This command captures the current LayoutViewer display to PDF file. PDF is only supported in Linux Platform. In Windows Platform, the captured display is saved in PostScript format.

4.4.1.2 Exit

Exit LayoutViewer.

4.4.2 Commands Menu

4.4.2.1 Clear Circuit Markers

Clear circuit markers which can be created by Drag-And-Drop from GofTrace Partial Schematic.

4.4.2.2 Clear Search Markers

Clear search markers which are those highlighted cells matching the searching string in search entry.

4.4.2.3 New Schematic

When cells or markers are selected in LayoutViewer, this command can launch a schematic with selected instances on it.

4.4.3 OptionsMenu

4.4.3.1 Show Grid

This option shows grid on LayoutViewer.

4.4.3.2 Show Instance

This option shows instance name on LayoutViewer. Zoom in scale should be large enough to show instance names.

4.4.3.3 Show Module

This option shows module name on LayoutViewer. Zoom in scale should be large enough to show module names.

4.4.3.4 Setup

LayoutViewer setups which include maximum search matching number and placement display zone area size.

4.4.4 Help Menu

4.4.4.1 Help on LayoutViewer

Visit NanDigits web site for Gates On the Fly manual section LayoutViewer.

4.4.5 LayoutViewer Pop Menu

Click mouse-right-button to pop up the menu.

4.4.5.1 Clear Circuit Markers

Clear circuit markers which can be created by Drag-And-Drop from GofTrace Partial Schematic .

4.4.5.2 Clear Searching Markers

This command clears searching markers which were activated by search function.

4.4.5.3 Copy Selected to

This command copies the selected gates to the following destination:

- Back to GofViewer
- A new Schematic
- An existing Schematic indentified by Number ID

4.4.6 Keyboard and mouse combination

4.4.6.1 Ctrl key to measure length

Press 'Ctrl' key and move mouse, the Cursor Coordination displays the length cursor moves in unit of 'um'.

4.4.6.2 Shift key to select multiple markers

Press 'Shift' key and press mouse-left-button, move mouse to draw a virtual rectangle. When release the mouse-left-button, those markers in the virtual rectangle are all highlighted. Click mouse-right-button to pop menu, those selected instances can be sent to other schematics or GofViewer the netlist view window by 'Copy Selected to' command.

4.4.7 Mouse operations

- Mouse-middle-wheel: Roll up to zoom in and roll down to zoom out the LayoutViewer window.
- Mouse-left-button: Click and release to select cells or markers. Press on LayoutViewer window to move it around.
- Mouse-middle-button: Drag-And-Drop selected instances.
- Mouse-right-button: Release to pop up menu.

4.4.8 Select color buttons

Click color buttons in 'Select color:' bar to select the current color. 'Select Color:' string changes to the current selected color. Any new Circuit Markers and Search Markers have this color.

4.4.9 Search function

Type search string in Search Entry to highlight the leaf instances matching the string on the LayoutViewer. The search string is in 'path/instance' string format, separated by '/'. Wildcard can be used in path and instance names. The markers have the color selected in 'Select color' bar.

The search string takes these options:

- -spare: When spare gate list is loaded by -sparelist option or get_spare_cells command.
- -type type-name: Only search those instances with specified type, 'nand' for example.
- -hier: Search all leaf instances under the specified hierarchy. For example, 'u_clk/* -hier'.
- -ref ref-name: Only search those instances with specified reference, 'NAND2X2' for example.

Examples:

```
'u_rtc/*' : Search leaf instances in hierarchy 'u_rtc'.
'* -hier -type nand': Search all leaf instances with 'nand' type in the design.
'u_clk/* -hier': Search all leaf in hierarchy 'u_clk' and its sub-hierarchies.
```

5 Appendix A

5.1 APIs Detail Usage

buffer

```
ECO command. Buffer 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 let the tool pick one.
              It supports repeater case by ",", for example, "INVX1,INVX16" would have 'INVX1'
              drives 'INVX16' and 'INVX16' drives the fanouts.
$fanout: How many fanout to insert a buffer.

Examples:

#1. For every 10 fanout of test_mode, add a buffer, BUFX6
buffer("test_mode", "BUFX6", 10);

#2. For every 10 fanout of 'clock', add repeaters, INVX2,INVX16
buffer("clock", "INVX2,INVX16", 10);

#3. Let the tool pick a buffer
buffer("clock", "", 10);
```

change_gate

```
ECO command. Modify an instance in ECO
Two types of usages
Usage1: change_gate($instance, $new_reference, $pin_mapping);
```



```
$instance: The instance under ECO. Support hierarchical name, "u_abc/U123"
$new_reference: The new reference name which the instance changes to, E.G. 'AND3X1'.
                If no reference is present, the ECO operation is assumed to
                  change the instance's pin connections.
$pin_mapping: Input pins mapping, ".new(old)", E.G. ".A1(A),.B1(B)"
              if two references have same input pins. The option can be empty
Usage2: change_gate($instance, $pin_connections);
$pin_connections: New pin connections, ".A(n242)".
                  The unspecified pins keeps the original connection.
                  E.G. pin 'B' connection is unchanged.

Examples:

#1. U123 has reference OR3X1 with input pins, A,B,C originally
# change U123 to AND3X1, all input pins are the same.
change_gate('U123', 'AND3X1', "");

#2. A and B keep the connections, discard C
change_gate('U123', 'AND2X1', "");

#3. A keeps the connections, B connects to what the old C connects. And discard old B
change_gate('U123', 'AND2X1', ".B(C)");

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

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

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

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

change_net

```
ECO command. Change a existing net's driver
Usage: change_net($net, $gate, $instance, $connections);
$net: The net to be ECOed
$gate: New leaf gate to drive the net
$instance: The instance name of the new gate. Optional, if it is empty, assigned by the tool
$connections:  The new gate input pins connections. If it is empty, the gate is inserted in the net
   Supported formats, 1. Very detail  ".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

```
ECO command. Modify pin connection in ECO
Two types of usages.

Usage1: change_pin($pin_name, $net);
Change pin's connection to a net
$pin_name: In the format of "instance/pin", can be more than one pins separated by ",",
             "instance1/pinA,instance2/pinB", E.G. "U123/A", "U123/A,U345/B"
             Hierarchical naming style is supported as well, "u_abc/U123/A"
             The pins have to be input in this mode.
$net: The net name the pin connects to.
      Hierarchical naming style is supported, "u_abc/net123"
      When the pin and the net are in different hierarchies, ports are added automatically
      E.G.
      # The tool creates 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 gets 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");

Usage2:my $inst = change_pin($pin_name, $leaf_cell, $new_instance, $connection);
Insert a new leaf cell to drive the pin
$inst: Return new instance name if new gate is created in the command.
$pin_name: In the format of "instance/pin", E.G. U123/A  Hierarchical naming is supported, u_abc/U123/A
           The pin can be output in this mode. The tool gets the net the pin drives,
             and change the command to
             change_net($thenet, $leaf_cell, $new_instance, $connection);
$leaf_cell: The leaf cell name to drive the $pin_name
$new_instance: The instance name for the new inserted leaf cell.
               The option is optional, the tool assigns one if it's empty
               If use '.', the instance is added to the same hierarchy as the $pin_name
$connection: 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 character '-' is used to connect up the original connection
                      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.


Examples:

#1. U123 has input pins A,B,C, U234 has input pins A0,A1,B
# Change A pin of U123 to net12345
change_pin("U123/A", "net12345");

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

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

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

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

#6.Insert a buffer to U123 A pin
```



```
change_pin("U123/A", "BUFX4", "", "-");

#7. 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)");

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

#9. Nested connection
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

```
ECO command. Change an output port's driver, or add gate after input port
Usage1: change_port($port, $gate, $instance, $connections);
$port: The port under ECO, can be input port or output port
$gate: New leaf gate to drive the port if the port is output
       Or add the gate after input port
$instance: The instance name for the new leaf cell, optional, the tool assigns one if it's empty
$connections: The new gate input pins connections. If it is empty, the gate is inserted in the net
   Supported formats, 1. Very detail  ".A(net0),.B(net1),.C(net2)"
                      2. Connect to the pins in alphabetical sequence
                          "net1,net0,net2" indicating A->net1,B->net0,C->net2
                      3. Other instance/pin "U408/Y,U409/Y,net2" indicating A->U408/Y,B->U409/Y,C->net2
                      4. Special character '-' is used to connect up the original connection
Usage2: change_port($port, $inst_pin);
$port: The port under ECO, output port
$inst_pin: In the format of 'u1234/Y', instance-name/pin-name
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:

#1. Add buffer to output port 'out1'
change_port("out1", "BUFX1", "eco_buf0", "-");
```

check_design

```
Check if the netlist status, searching for unresolved modules, floating and multi-drivers
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
   -nouniquify: Dont check uniquify

Examples:

check_design;
check_design('-ignore', 'FE_UNCONNECT*');
check_design('-ignore', 'FE_UNCONNECT*,SCAN_*');
check_design('-ignore', 'W-108');
check_design("-eco");
```

compare

```
Logic equivalence check on output port and register input pins
Usage: my $no_eq_num = compare(@nets, @options);
@options:
   -help: Print this info
$no_eq_num: Return back non-equivalent number

Examples:

#1. Check if output port 'state_out' is equivalent in IMP/REF netlists
compare("state_out");

#2. Check if 'state_reg_0_/D' and 'state_reg_1_/D' are equivalent in IMP/REF netlists
compare("state_reg_0_/D", "state_reg_1_/D");
```

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

Examples:

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

convert_gated_clocks

```
ECO command. Convert gated clocks to MUX logic.
In metal ECO, if gated clock cell is not in spare gate list, this command should run before map_spare_cells
Usage: my $cnt = convert_gated_clocks();
$cnt: The number of gated clock cells having been converted
```

current_design

```
Set the current top level module
Usage: current_design($module);
$module: Set $module as the current top level module.
         If the argument is missing, return the current setting
         ".." set to parent module, "~" set to the most top level module
Note: It can be reset to the root top module by 'undo_eco'. It is alias command of 'set_top'
```

current_instance

```
Set the current instance, alias of 'set_inst'
Usage: current_instance($instance);
$instance: Set $instance as the current instance.
         If the argument is missing, return the current setting
         ".." set to parent module, "~" set to the most top level module
Note: It can be reset to the root top module by 'undo_eco'. It has same effect of 'set_top' and 'current_design'
```

del_gate

```
ECO command. Delete gate
Usage: del_gate($inst);
$inst: The instance to be deleted.
```

del_net

ECO command. Delete net
Usage: del_net($net);
$net: The net to be deleted.

del_port

ECO command. Delete port
Usage: del_port($port);
$port: The port to be deleted.

exist_inst

Check if an instance exists
Usage:my $ret = exist_inst($inst);
$inst: The instance for checking
$ret: 1, the instance exists 0, the instance does not exist

exist_wire

Check if a wire exists
Usage:my $ret = exist_wire($wire);
$wire: The wire name for checking
$ret: 1: exists 0: not exist

fix_design

ECO command. Fix the whole design in global mode
Usage: fix_design(@options);
@options:
    -help: Print this information

Examples:

#1. Fix design on module 'VIDEO_TOP' and its sub-modules
```
set_top('VIDEO_TOP');
set_ignore_output("TEST_SO*");
set_pin_constant("TEST_EN", 0);
fix_design;
save_session("this_eco");
my $error = LEC;
```

fix_logic

ECO command. Fix listed points
Usage: fix_logic(@pin_port_list);
@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"
     '\' should be kept if the instance has '\' as prefix.
     E.G. '\bbr_ccd_reg[0] ' instance has '\' and last space in the name.

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. Add one new flop, input pins have the same connections as the Reference Netlist
#  and the output is floating, -recover option sets to 0
fix_logic('new_flop_reg/D', 'new_flop_reg/CK', 'new_flop_reg/RB');
```

fix_modules

ECO command. Fix modules listed in the arguments
Usage: fix_modules(@modules, @options);
@modules: The modules to be fixed
@options:
    -all: Fix all hierarchical modules under the current top module
    -exclude points: Exclude the specific points.
                     For example, "state_reg_0/D,state_reg_1/D", or "scan_in*", "scan_out*"
    -recover value: Recover option for fix_logic
Note: The command can be replaced by
```
foreach my $mod (@modules){
   set_top($cmod);
   fix_logic;
}
```

Examples:

#1. Fix module 'abc_control'
```
fix_modules('abc_control');

#2. Fix modules '*zebra_control*'
my @modules = get_modules('*zebra_control*', '-hier');
fix_modules(@modules);

#3. Exclude some points
fix_modules('abc_mod', '-exclude', "state_reg_0/D,state_reg_1/D,scan_out*");

#4. Fix all modules under top
set_top('top');
fix_modules('-all');
```

flatten_modules

Flatten hierarchical modules in reference netlist
Usage: flatten_modules(@module_names);
@module_names: List of modules to be flatten

Examples:

```
flatten_modules("retime_1", "sync_cell_0");
```

get_cell_info

Get information of a module or instance
Usage: $data = get_cell_info($module_or_inst, @options);
$module_or_inst: The module or instance's name
@options:
    -help: Print 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 is 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
    -fun:  Get the function string 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
$data: Returned data, if '-attribute' option is present, $data is 0 or 1
In option is '-conns' case,
It is a hash having 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 hierarchical_instance hierarchical_module
```

Examples:

```
#1. Get area of one leaf cell
my $area = get_cell_info("AND2X2", "-area");

#2. Get an attribute of one leaf cell
my $is_iso = get_cell_info("ISOX2", "-attribute", "is_isolation_cell");
```

get_cells

```
Get all cells in the current module or sub-modules
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 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','itiming','cg','not','rom','ram' ...
                        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
    -new: Only new created ECO instances
    -verbose: To print out reference with instance
    -dotpath: Path delimit is '.' instead of '/'
    -nobackslash: Remove backslash
    -nonscan: Flops/sync-cells not in scan chain including those scan pins tied off
@cells: Returned array with all instances matched
```

Examples:

```
#1. Get all instances in the current module
my @cells = get_cells('*');

#2. Get all instances in the current module
my @cells = get_cells();

#3. Get all instances matching 'U234*' in the current module
my @cells = get_cells('U234*');

#4. Regular expression. Get all instances starting with U and followed by
#    two characters, like U10, U99
my @cells = get_cells('/U../');

#5. Get all instances matching *reg_*_ hierarchically
my @cells = get_cells('*reg_*_', '-hier');

#6. Get all instances hierarchically and having reference matching DFF*
my @cells = get_cells('*', '-hier', '-ref', 'DFF*');

#7. Get all instances in 'u_kb'
my @cells = get_cells('u_kb/*');

#8. Get all flops, sync-cells not in scan chain
my @cells = get_cells('-hier', '-notscan');
```

get_conns

```
Get connections of net or pin in the top level module, return the leafs and the hierarchical connections
Usage: @result = get_conns($net_or_pin, @options);
$net_or_pin: The net name or pin name that needs to get connections.
@options:
   -driver: Return driver only
   -load: Return load only
   -count: Return connections count
@result: a two dimension array
         instance, port_name, pin_or_port, load_or_driver, is_it_a_leaf,
@result = ([instance_0, pin_0, 'pin', 'load', 1],
           ...
          )
```

Examples:

```
#1. Net 'n599' has three connections, instance 'U198' is the driver
get_conns("n599");
gte_344 A[14] pin load 0
U198 Y pin driver 1
U94 AN pin load 1

#2. Net 'qcifhbeat' has three connections, it is output port of the current top level module
get_conns("qcifhbeat")
 qcifhbeat port load
U80 A pin load 1
qcifhbeat_reg Q pin driver 1

#3. The argument in inst/pin format
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
```

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

```
Get instantiation of instance
Usage: my $line = get_definition($inst);
$inst: Instance name.
$line: The instantiating line

Examples:

get_definition('U78');
Returns "AND2X1 U78(.A(n1), .B(n2), .Z(n3));"
```

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
   -nonbuf: Trace the drivers until none buffer
@driver: The driver in array format, if '-pin' is not present.
         If the point is floating, @driver is empty,
         $driver[0]: Driver instance name. It is empty if the driver is port
         $driver[1]: Driver pin or port name. If the driver is a port, it is the port name
         $driver[2]: Driver type. It is string "pin" or "port" depending on the driver is port or leaf cell
         $driver[3]: Driver phase. It is valid when -nonbuf is present,
                     indicating if the driver path is inverted
                     0: not inverted 1: inverted
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. Get driver of a net
@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. Return in instance/pin format
$driver = get_driver("net12345", "-pin");
$driver has content of "U1247/Y"
```

get_drivers

```
Get the drivers of a net or pin
Usage: @drivers = get_drivers($point, @options);
$point: net name or pin name, 'n12345' or 'U12345/A1'
@options:
   -nonbuf: Trace the drivers until none buffer
@drivers: An array of the drivers, if the point is floating, @drivers is empty,
         if the point has multi-drivers, @drivers has more than one items.
         For each item in @drivers
         Index 0: instance, it is empty if the driver is port
         Index 1: pin or port, if the driver is port, return port
         Index 2: string "pin" or "port" depending on the driver is port or leaf cell
         Index 3: indicating drive path inverted or not
         If 'nonbuf' is present, the last item in @drivers is the non-buffer driver
         So '$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. Get drivers of a net
@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"],
    ["inv_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 is treated as "*"
@options:
    -module: module name to have its instance name found
$instance: Return the first instance matching

Examples:

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

#2. 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 is treated as "*"

@instances: Array of the hierarchical instances

Examples:

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

get_leaf_pin_dir

Get leaf cell pin's direction input/output/inout
Usage: my $dir = get_leaf_pin_dir("$leaf_name/$pin");
$pin: pin name, E.G. A or B or Y
$leaf: Leaf cell name, E.G. NAND2X2
$dir: return direction, input/output/inout

Examples:

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

get_leafs_count

Get all leaf cells name and count in the top level module, return an array
Usage: @leaf_count = get_leafs_count;
@leaf_count: Array of leaf name and count
 ([leaf0, cnt0], [leaf1, cnt1], ...)

Examples:

```
@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
";
}
```

get_lib_cells

Get leaf gates in libraries
Usage: my @cells = get_lib_cells($pattern, @options);
@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 matches only one lib cell, print the cell lib information
```
$pattern: Library cell name pattern, can have '*'.
@cells: Return array with name matching

get_loads

Get loads of net 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
   -fanend: Fanout endpoints, flops or ports
```
@result: a two dimension array. Each item has format of 'instance' and 'pin_name' if
         the load is leaf cell. Or 'port_name' and 'GOF_PIN_IN'
```
@result = ([instance_0, pin_0],
           [instance_1, pin_1],
           [port_name, GOF_PIN_IN],
           ...
          )
```

get_logic_cone

Get logic cone of nets or pins
Usage: $result = get_logic_cone(@InstancePinList, @options);
@InstancePinList: Instance/pin and net list.
$result: 1, the command fails. 0, the command completed successfully
@options:
```
    -o file_name: Write output to the file. Default logic_cone.v
```

Examples:

```
my @InstPin = ('abc_reg/D', 'n12345');
my $ret = get_logic_cone(@InstPin, '-o', 'MyLogicCone.v');
# The logic cone is written out to verilog file 'MyLogicCone.v'
```

get_modules

Get all hierarchical modules under current module
Usage: @modules = get_modules($pattern, @options);
$pattern: Match pattern, can have wildcard "*", if it is empty, it is treated as "*"
@options:
```
    -help: Print this information
    -hier: Get all modules hierarchically
```
@modules: Modules list, ("module0", "module1", ...)

Examples:

```
@modules = get_modules("*TM*");  # Any hierarchical modules with TM in the name.
@modules = get_modules;          # All hierarchical modules.
@modules = get_modules("-hier"); # All hierarchical modules and sub-modules under current module.
```

get_net_of

Get net name connecting to a pin
Usage: my $net = get_net_of($pin);
$pin: The pin of the instance, 'U1234.A1' or 'U1234/A1'
$net: The net name connecting to the pin

get_nets

Get nets that matching pattern
Usage: @nets = get_nets($pattern, @options);
$pattern: The net naming pattern, "*" or empty for all nets
@options:;
```
   -const0: Get all constant zero nets
   -const1: Get all constant one nets
```
@nets: returned net array.

Examples:

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

2#. All nets with 'dbuffer' as prefix
@nets = get_nets("dbuffer_*");

3#. Get constant nets
@nets = get_nets("-const0");
```


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:
    -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
If no option is present, get all pins
@pins: All pins returned, in 'instance/pin' format

Examples:

```
#1. Get input pins of a hierarchical instance
my @pins = get_pins("-input", "u_abc/U123");
Result @pins = ("u_abc/U123/A", "u_abc/U123/B")

#2. Get pins of a leaf cell
@pins = get_pins("AND2X2");
Result @pins = ("A","B","Y")
```

get_ports

Get all ports in the current top level module
Usage: @matching_ports = get_ports($pattern, @options);
$pattern: Match pattern, can have wildcard "*". If it is empty, it is treated as "*"
@options:
          -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

If no option is present, get all ports

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

Examples:

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

get_ref

Get the reference of the instance, return leaf cell name or hierarchical module name
Usage: $reference = get_ref($instance);
$instance: Instance name, "U123"
$reference: Return reference name, "NAND2X4"

get_resolved

Resolve the relative path to module and leaf item
Usage: ($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");
$module has value "def"
$leaf has value "U456"
```

get_roots

Get root designs name
Usage: my @rootdesigns = get_roots;
@rootdesign: returned root designs name

get_scan_flop

Get scan flop for the non scan flop
Usage: my $scanflop = get_scan_flop($nonscanflop);

Examples:

```
# Get a corresponding scan flop for non scan flop DFFHQX1
my $scanflop = get_scan_flop("DFFHQX1");
```

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
    -gate_array gate_array_naming: Gate Array naming style, like 'G*', most standard libraries have Gate Array cells naming starting with 'G'
                  This option enables Metal Only Gate Array Spare Cells ECO flow
    -gate_array_filler gate_array_filler_naming: Gate Array Filler naming style, like 'GFILL*',
                  or 'GFILL*|GDCAP*' to include both GA Filler and GA DCAP

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");

#4. Enable Gate Array Spare Cells Metal Only ECO Flow, map_spare_cells will map to Gate Array Cells only
get_spare_cells("-gate_array", "G*", "-gate_array_filler", "GFILL*|GDCAP*");
```

Note:

```
The API has to run on top level,
set_top('most_top_module')
get_spare_cells("someSpare_*/spr_gate*");
```

gexit

```
Exit the GofCall interactive mode
Usage: gexit;
```

gprint

```
Print the message and save to log file
Usage: gprint($info);
$info: The 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: 0, it's a hierarchical module, (Or the module is not defined)
       1, it's leaf cell. Like, NAND4X8
```

is_scan_flop

```
Check if an instance is scan flop
Usage my $isscan = is_scan_flop($name);
$name: The cell name or instance name
$isscan: 0, it's not a scan flop
         1, it's a scan flop
```

is_seq

```
Check if an instance is a specific sequential cell
Usage: my $isseq = is_seq($name, @options);
$name: The instance under check
@options:
    -help:  This information
    -ff:    Check if it's a flipflop
    -latch: Check if it's a latch
    -cg:    Check if it's a gated clock
    -rom:   Check if it's a rom
    -ram:   Check if it's a ram
$isseq: 0, it is not the specific sequential cell
        1, it is the specific sequential cell
```

map_spare_cells

```
ECO command. Map all new created cells to spare cells
Usage: map_spare_cells;
@options:
    -help: Print 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"
    -lib_header file_name: This option is Valid when '-syn' option is present. To insert the content of
                           file 'file_name' to the header of synthesis tcl script. So that '.lib' file to
                           '.db' conversion can be avoided in Design Compiler case.
                           For example, in Design Compiler case, the file content should have
                             set_search_path [list /project/lib/synopsys_db]
                             set_target_library [list art40_hvt art40_svt]
                             set link_library [list art40_hvt art40_svt]
    -nofreed: Don't add freed gates for synthesis.
    -nobuf: Don't insert buffers/repeaters in long wires.
    -notielow: Don't 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

```
ECO command. Create 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
        returns.
      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 the shell "GOF >"
```

new_net

```
ECO command. Create 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 assigns one automatically
$reference: The leaf gate name to drive the net.
$new_instance: The instance name of the new cell, or leave it empty to get automatically assigned.
$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 to U408/Y, B to U409/Y and C to net2
                    4. The "instance/pin" can have sub-instance hierarchy, "u_abc/U408/Y"
@return: Have the new created instance and net name.
$return[0] : New created instance.
$return[1] : New created 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 is returned.
      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:

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

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

#3. Create a new net "net500"
new_net("net500");

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

new_port

```
ECO command. Create a new port for the current top level module
Usage: new_port($name, @options);
$name: Port name
@options:
    -input: New an input port
    -output: New an output port
    -inout: New an inout port

Note: The port name has to be pure words or with bus bit, like, abc[0], abc[1]

Examples:

new_port('prop_control_en', '-input');  # create an input port naming 'prop_control_en'
new_port('prop_state[2]', '-output');   # create an output port with bus bit 'prop_state[2]'
new_port('prop_state[3]', '-output');   # create an output port with bus bit 'prop_state[3]'
```

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_spare_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
This command has effect on change_port ECO command
```

pop_top

```
Pop out the saved top level module from the stack and discard the current setting
Usage: pop_top;
```

push_top

```
Set the current 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, @options);
@files:  DEF files
@options:
   -defverbose: Report all DEF parsing warnings and errors
$status: If zero, the files have been read in successfully
         if non-zero, failed to read in the files

Examples:

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

read_design

```
Read verilog netlist
Usage: my $status = read_design(@files, @options);
@files: Verilog netlist files
@options:
    -imp: The netlists are implementation design which are under ECO. Normally they are pre-layout netlists.
    -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 is bypassed if the command is executed again
Note: If no -imp or -ref option is provided, the netlist is assumed 'implementation'
$status: If zero, the file is read in successfully
         if one, failed in reading the file

Examples:

#1. Read in an implementation netlist file
```



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

#2. Read in a reference netlist file
my $status = read_design("-ref", "soc_design_released.gv");

#3. Read in two reference netlist files
my $status = read_design("-ref", "soc_design_released.gv", "soc_io.gv");
```

read_file

```
Read timing violation report file
Usage: my $status = read_file($file_name, @options);
$file_name: file name
@options:
    -format format: accu/pt
                    accu --- Accucore report file.
                    pt   --- Prime Time report file
$status: If zero, the file is read in successfully
         if one, failed to read in the file

Examples:

my $status = read_file("soc_primetime_hold.report", "-format", "pt");

Note: Prime Time timing report file should be generated by report_timing command with these options
      report_timing -nosplit -path_type full_clock_expanded -delay max/min -input_pins \
                    -nets -max_paths 10000 -transition_time -capacitance
```

read_lef

```
Read LEF file
Usage: my $status = read_lef(@files);
@files:  LEF files
$status: If zero, the files are read in successfully
         if one, failed to read in the files

Examples:

my $status = read_lef("soc_top.lef");  # Read in soc_top.lef
my $status = read_lef("soc_top.lef", "soc_top1.lef", "soc_top2.lef");  # Read in multiple LEF files
```

read_library

```
Read standard library or verilog library files
Usage: my $status = read_library(@files, @options);
@options:
   -v: Treat the @files as verilog library files
   -lib: Treat the @files as standard library files
   -f library_list_file: Load library files from list file, the list file has format of
                         -v verilog_lib0.v
                         -v verilog_lib1.v
                         -lib tsmc40.lib
   -vmacro: Treat the @files as macro library files which are used as macro cell in ECO
   -rtl: Treat as RTL format
   -gate: Treat as gate format, if not specify -rtl or -gate, the tool automatic picks one
@files: Standard library files, or verilog library files

Note: The three options, '-v' '-lib' and '-vmacro' don't coexist.
      If the file has .lib extension, '-lib' can be omitted, and it is treated as standard library file.
      If the file has .v or .vlib extension, '-v' can be omitted, and it is treated as verilog file.
$status: If zero, the file is read in successfully
         if one, failed to read in the file

Examples:

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");
my $status = read_library("-f", "lib_files.list");
```

read_svf

```
Read Synopsys SVF text files
Usage: my $status = read_svf(@options, @files);
@files: SVF text files
@options:
    -imp: The SVF file is for the Implementation netlist
    -ref: The SVF file is for the Reference Netlist
    -Top_1: The SVF file is for Top_1 tree database
    -Top_2: The SVF file is for Top_2 tree database

$status: If zero, the files have been read in successfully
         if non-zero, failed to read in the files

Note: SVF should be in text format
```

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, @options);
$filename: Write report to the file name. If $filename is not present, print to screen
@options:
    -help: Print this information
    -simple: Print in simple format
```

report_spares

```
Report Spare cells
Usage: report_spares;
```

restore_session

```
Restore ECO session
Usage: restore_session("$directory/$session_name");
$directory: The directory that the session has been saved
$session_name: The session name

Examples:

# To restore the session "myeco" in sub-directory "mach_ai"
restore_session("mach_ai/myeco");
```


run

Run GofCall script
Usage: run($script_name);

Examples:

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

run_lec

Run Logic Equivalence Check on Implementation Netlist and Reference Netlist
Usage: LEC(@options);
@options:

save_session

Save ECO session
Usage: save_session("$directory/$session_name");
$directory: The directory that the session should be saved
$session_name: The session name

Examples:

```
# To save a session "my_eco" in sub-directory "mach_ai"
save_session("mach_ai/my_eco");
```

sch

Launch schematic to verify ECO
Usage: sch(@instances, @options);
@instances: Instances or nets in the current module to be displayed on the schematic
@options:
    -set value: Set a value when launch the schematic
    -to value: To existing schematic
    -both: Load the item in both implementation and reference netlist

Examples:

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

set_auto_fix_floating

ECO setting. Enable automatic fixing floating input ports after fix_modules
By default, it is enabled.
Usage: set_auto_fix_floating(0); --- Disable automatic fixing floating input ports.

set_bfix

Enable or disable BFIX features
Usage: set_bfix($val);
$val: Default 0x3
      Bit 0, Set one to enable Reorder Method
      Bit 1, Set one to enable Cutpoint Method
      bit 2, Set one to force using Reorder/Cutpoint Method instead of Structure Method

set_blackbox

Set Blackbox on Modules
Usage: set_blackbox(@modules);
@module: Module names to be set as blackbox, accept wild card '*'

Examples:

```
#1. Set Blackbox on DW modules
set_blackbox("*DW_pipe*");

#2. Set Blackbox on 'ABC' and 'DEF' modules
set_blackbox("ABC", "DEF");
```

set_bound_opti

Enable or disable boundary optimization check. Enabled by default
Usage: set_bound_opti($val);
$val: 1, enable boundary optimization check. 0, disable.

set_buffer

Set buffer type. The tool automatically picks one if the command is not called
Usage: set_buffer($buffer);
$buffer: Lib cell name for buffer

Examples:

```
set_buffer("BUFX2");
```

set_buffer_distance

Set distance limit for inserting buffer
Usage: set_buffer_distance($distance_val);
$distance_val: distance to insert buffer, in um

set_constraints

Set constraints for map_spare_cells command
Usage: set_constraints(@options);
@options:;
   -type type_constraint : Set spare cell type constraint, type_constraint is a string
                           listing spare cells separated by ','
   -num num_constraint   : Set spare cell number constraint, num_constraint is a string
                           in format of 'mux<16,nand<18'

Note: The number constraint only controls the number of spare types to be used. The spare gates list should have
      'nand/and', 'nor/or' and 'inv' types of leaf cells for synthesis mapping, and have spare flops for direct mapping, 'mux' is optional.
      If used with get_spare_cells command, this command should be used after get_spare_cells, check example #3

Examples:

```
#1. Use less than 16 'mux' and less than 18 'nand' spare gates in map_spare_cells
set_constraints('-num', 'mux<16,nand<18');
map_spare_cells;


#2. Use NAD2X1 NOR2X1 INVX1 and MUX2X1 as spare type gates
set_constraints('-type', 'NAND2X1,NOR2X1,INVX1,MUX2X1');
map_spare_cells;
```



```
#3. Set constraint after spare list created
get_spare_cells("u_Spare*/*spr_gate*");
set_constraints('-num', 'and<1'); # So that no AND spare gate will be used
map_spare_cells;
```

set_cutpoint_thresh

```
Set Cutpoint Threshold
Usage: set_cutpoint_thresh($val);
$val: Threshold value, default 100
```

set_cutpoint_ultra

```
Enable or disable CutPoint Ultra Effort
Usage: set_cutpoint_ultra($val);
$val: 0, Disable CutPoint Ultra, by default
      1, Enable CutPoint Ultra
```

set_dont_fix_modules

```
Set dont fix property on modules
Usage: set_dont_fix_modules(@modules);
@module: Module names not to be fixed

Example:

#1. Set dont fix on pcie_ctrl and pcie_top module
set_dont_fix_modules("pcie_ctrl", "pcie_top");
```

set_dont_use

```
Set dont use property on library cells
Usage: set_dont_use(@cell_list);
@cell_list: List of the dont use cells which is not used in automatic ECO. Wild card '*' is supported

Examples:

set_dont_use("INVX30","AND2X24");
set_dont_use("PWR_*");
```

set_eco_effort

```
ECO setting. Set ECO effort
Usage: set_eco_effort($effort);
$effort: One of the three choices, high, medium and low. By default, high effort is used

Examples:

#1. Change ECO effort to medium
set_eco_effort("medium");
```

set_equal

```
ECO setting. Set two points to be equivalent in the Reference and Implementation Netlists
             The points can be input port, flop instance or black-box's output pin.
             The point names should have 'i:' or 'r:' as prefix to indicate they are for Reference or Implementation, or
             the first point is assumed as Reference and the second Implementation.
             Both of the points can be from Reference or Implementation
Usage: set_equal($ref_point, $imp_point);
$ref_point: The point in the Reference Netlist. It should be the first argument if it doesn't have 'i:' or 'r:' as prefix
$imp_point: The point in the Implementation Netlist. It should be the second argument if it doesn't have 'i:' or 'r:' as prefix

Examples:

#1. Input port 'in_a' in Reference Netlist is equivalent to input port 'in_b' in Implementation Netlist in top module
set_top('top_module');
set_equal('r:in_a', 'i:in-b');

#2. Flop instance 'subinst/flopa_reg' is equivalent to input port 'IN0' in the Implementation Netlist
set_top('top_module');
set_equal('i:subinst/flopa_reg', 'i:IN0');
fix_design();
```

set_error_out

```
Set error out setting
Usage: set_error_out($value);
$value: 1, Abort the program when APIs have run error, default setting
        0, Ignore the error and continue the program

Examples:

# Program continues when there is error in change_pin
set_error_out(0);
change_pin("nonexisting_instance/A", "1'b0"); # It will continue, even though nonexisting_instance is not in the database
```

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_floating_as_zero

```
Set floating net as constant zero
Usage: set_floating_as_zero($value);
$value: 0, disable
        1, enable (default)
```

set_high_effort

```
Set high ECO effort on modules
Usage: set_high_effort(@options);
@options:
    -help: Print this information
```



```
    -include module_list: Only set high ECO effort on the modules listed,
                          module_list has format of module names separated by ',', wild card is acceptable
                          For example, 'mem_control,dma_*'
    -exclude module_list: Exclude high ECO effort on the modules listed
                          module_list has format of module names separated by ',', wild card is acceptable
                          For example, 'mem_control,dma_*'
    -timeout time_in_seconds: Set time out for each run, default to time out in 900 seconds
                          time_in_seconds is an integer indicating time out in seconds
```

Examples:

```
#1. Set ECO high effort on all modules under ECO
set_high_effort();

#2. Set ECO high effort on module 'mem_control_1'
set_high_effort('-include', 'mem_control_1');

#3. Set ECO high effort on modules matching 'mem_control_*' and modules matching 'dma_*'
set_high_effort('-include', 'mem_control_*,dma_*');

#4. Enable ECO high effort, but excluding module 'mem_control_1  '
set_high_effort('-exclude', 'mem_control_1');

#5. Enable ECO high effort with time out in 600 seconds
set_high_effort('-timeout', 600);
```

set_ignore_instance

```
ECO setting. Set ignored sequential or blackbox instances in ECO
Usage: set_ignore_instnace(@ignored_instances)
@ignored_instances: Instances to be ignored in ECO, accept wild card '*'
```

Examples:

```
#1. Ignore instances matching RAND_CNT_reg* in ECO
set_top('VIDEO_TOP');
set_ignore_instance('RAND_CNT_reg*');
set_top('DESIGN_TOP');
fix_design();

#2. Ignore instances matching current_state_reg* in instance u_video
set_top('DESIGN_TOP');
set_ignore_instance('u_video/current_state_reg*');
fix_design();
```

set_ignore_network

```
ECO setting. Set ignore network in ECO
Usage: set_ignore_network(@ignored_nets, @options)
@ignored_nets: Net and its network to be ignored in ECO, accept wild card '*'
@options:
   -help: Print this information
   -pin:  @ignored_nets are in pin format, for example, 'DONT_mux_clk/PIN_Y'
```

Examples:

```
#1. Ignore scan_en and scan_in
set_ignore_network('scan_en*', 'scan_in*');

#2. Ignore PAD PAD_SCAN_EN's output pin 'core' and its network
set_ignore_network('PAD_SCAN_EN/core', '-pin');
```

set_ignore_output

```
ECO setting. Set ignore output ports
Usage: set_ignore_output(@ignored_ports, @options)
@ignored_ports: Output ports to be ignored, accept wild card '*'
@options:
   -help: Print this information
   -both: Apply to both Reference and Implementation Netlist. Enabled by default
   -ref:  Apply to Reference Netlist
   -imp:  Apply to Implementation Netlist
```

Examples:

```
#1. Ignore output ports matching *scan_out* in ECO
set_top('design_top');
set_ignore_output('*scan_out*');
set_pin_constant('scan_en', 0);
fix_design();

#2. Ignore output ports matching *TSTCON* in Implementation Netlist
set_top('CHIP_TOP');
set_ignore_output('*TSTCON*', '-imp');
```

set_inside_mod

```
Set fix scope inside the current module
If set to 1, the tool only use resource inside the current module to fix the non-eq points.
By default, it is disabled.
Usage: set_inside_mod($val);
$val: 0, disable 1, enable
```

set_inst

```
Set the current instance, alias of 'current_instance'
Usage: set_inst($instance);
$instance: Set $instance as the current instance.
         If the argument is missing, return the current setting
         ".." set to parent, "~" set to the most top level module
Note: It can be reset to the root top module by 'undo_eco'. It has same effect of 'set_top' and 'current_design'
```

set_inv

```
ECO setting. Set two points to be inverted in the Reference and Implementation Netlists
             The points can be input port, flop instance or black-box's output pin.
             The point names should have 'i:' or 'r:' as prefix to indicate they are for Reference or Implementation, or
             the first point is assumed as Reference and the second Implementation.
             Both of the points can be from Reference or Implementation by using 'i:' or 'r:' on both point names.
Usage: set_inv($ref_point, $imp_point);
$ref_point: The point in the Reference Netlist. It should be the first argument if it doesn't have 'i:' or 'r:' as prefix
$imp_point: The point in the Implementation Netlist. It should be the second argument if it doesn't have 'i:' or 'r:' as prefix
```

Examples:

```
#1. Input port 'in_a' in the Reference Netlist is inverted to input port 'in_a_BAR' in the Implementation Netlist in top module
set_top('top_module');
set_inv('r:in_a', 'i:in_a_BAR');
```

set_invert

Set invert type. The tool automatically picks one if the command is not called
Usage: set_invert($invert);
$invert: Lib cell name for invert

Examples:

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

set_keep_format

Set keep format of the original verilog when ECO is done
Usage: set_keep_format($value);
$value: 0: disable, 1 : enable.

set_keep_tree

Set keeping buffer tree, so that buffer tree won't be removed in ECO
By default , it is disabled.
Usage: set_keep_tree($val);
$val: 0, disable 1, enable

set_keypoints_rep_in_ref

ECO setting. Replace keypoints naming in Reference Netlist.
Keypoints naming matching the first argument, and replace the matched string by the second argument
Usage: set_keypoints_rep_in_ref($match_string, $rep_string);
$match_string: Keypoints naming matching this string
$rep_string: To replace the matched string by this string

Note: The command only apply to Reference Netlist

Examples:

```
#1. Replace the last '_' in Keypoints naming in Reference Netlist
set_keypoints_rep_in_ref('_$', '');

#2. Replace the last '0' in Keypoints naming in Reference Netlist
set_keypoints_rep_in_ref('0$', '');
```

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_low_effort

Set low ECO effort to speed up ECO process
Usage: set_low_effort(@options);
@options:
    -help: Print this information

Examples:

```
#1. Set ECO low effort on all modules under ECO
set_low_effort();
```

set_mapped_point

ECO setting. Set two points mapped in Reference and Implementation Netlists
Usage: set_mapped_point($ref_point, $imp_point, @options);
$ref_point: Register instance or output port in Reference Netlist
$imp_point: Register instance or output port in Implementation Netlist
@options:
    -invert: The two points are expected to be inverted

Examples:

```
#1. Two outputs are mapped key points
set_mapped_point("ref_sync", "imp_sync");
```

set_mapping_method

LEC setting. Detecting flop phase inversion.
Usage: set_mapping_method("-phase");

set_max_lines

Set max output lines
Usage: set_max_lines($num);
$num: New max lines number. Default to be 500

set_max_loop

Setup max loop, the tool 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_multibit_blasting

Set blasting on multibit flops.
set_multibit_blasting($enable);
$enable: 1 to enable the setting
Note: This command should run before read_design

Examples:

```
set_multibit_blasting(1);
read_design("-ref", "ref.v");
read_design("-imp", "imp.v");
```

set_net_constant

Set net to a constant value
Usage: set_net_constant($net, $value, @options);
$net: Net name. It can be a bus.
$value: Decimal value that the pin should be set
@options:
    -help: Print this information
    -both: Set the net to the constant value on both Implementation and Reference. Enabled by default.
    -imp: Set the net to the constant only on Implementation
    -ref: Set the net to the constant only on Reference

Examples:

```
#1. Set all_test net to zero in Implementation Netlist
set_top('DESIGN_TOP_DFT_WRAPPER');
set_net_constant('all_test', 0, '-imp');
set_ignore_output('PIN_EDT_CHANNEL_OUT*', '-imp');
fix_design();
```

set_noexact_pin_match

ECO setting. Don't match some special pins
These pins normally don't exist in RTL but added by Synthesis, DFT or other tools.
Usage: set_noexact_pin_match($pattern);
$pattern: Pin pattern in regular expression, '\bIN\d+\b'

Note: The command only apply to Reference Netlist. It should be run before reading reference netlist

Examples:

```
#1. Don't match pins like IN0, IN1, IN2
set_noexact_pin_match('\bIN\d+\b');
read_design('-ref', 'ref_netlist.v');
```

set_phase_adjust_en

Enable phase adjusting
Usage: set_phase_adjust_en($val);
$val: 0, Disable phase adjusting
      1, Enable phase adjusting, by default

set_phase_inv

ECO setting. Set flops invert phase in the Reference and Implementation Netlists
Usage: set_phase_inv($flop1, $flop2 ...);
$flop1, $flop2: Flop instance list in full path

Examples:

```
#1. Set flop instance u_ip/u_control/a_reg to have invert phase
set_top('top_module');
set_phase_inv('u_ip/u_control/a_reg');

#2. Set flop instances u_ip/u_control/a_reg and u_ip/u_control_b/b_reg to have invert phase
set_top('top_module');
set_phase_inv('u_ip/u_control/a_reg', 'u_ip/u_control_b/b_reg');
```

set_pin_constant

Set pin to a constant value
Usage: set_pin_constant($pin, $value, @options);
$pin: Input pin name. It can be a bus, or an instance pin.
$value: Decimal value that the pin should be set
@options:
    -help: Print this information
    -both: Set the pin to the constant value on both Implementation and Reference. Enabled by default.
    -imp: Set the pin to the constant only on Implementation
    -ref: Set the pin to the constant only on Reference

Examples:

```
#1. Set test scan test pin to zero
set_top('DESING_TOP');
set_pin_constant('PIN_SCAN_TEST', 0);
set_ignore_output('PIN_SCAN_SO*');
fix_design();

#2. Set one bus port to all ones on Implementation
set_top('DESING_TOP');
set_pin_constant('PIN_CONTROL[3:0]', 15, '-imp');
fix_design();
```

set_power

Set power pins connections for 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 does 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_recovery_distance

Set distance limit for gates recovery in ECO
Usage: set_recovery_distance($distance);
$distance: Distance to recover deleted gate, in unit of 'um'

set_remove_undsc_in_ref

ECO setting. Remove last '_' in flop instance in Reference Netlist
It's a special command to remove the last '_' in flop instance in Reference Netlist
   to match Implementation Netlist.
Usage: set_remove_undsc_in_ref($value);
$value: 1, enable. 0, disable

Note: The command only apply to Reference Netlist

set_tiehi_net

Set tiehi net name, it is 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 is 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 level module.
         If the argument is missing, return the current setting
         ".." set to parent module, "~" set to the most top level 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_user_match

Set match between multi-bit flops to multi-bit flops, and between multi-bit flops to single bit flop
Usage: set_user_match($inst1, $inst2);
$inst1: The first flop instance, in the format of 'r:reg_1_0A/*dff.00.0*' if it is multibit
           or 'r:reg_1A' if it is single bit
$inst2: The second flop instance, in the format of 'i:reg_1_0A/*dff.00.0*' if it is multibit
           or 'i:reg_1A' if it is single bit

Examples:

```
set_user_match('r:reg_1_0A/\*dff.00.0\*', 'i:reg_0A');
set_user_match('r:reg_1_0A/\*dff.00.1\*', 'i:reg_1A');
set_user_match('r:reg_2_1A/\*dff.00.1\*', 'i:reg_1_0A/\*dff.00.0\*');
```

set_verbose

Run script in verbose mode
Usage: set_verbose($num);
$num: Verbose level, higher to be more verbose

setup_eco

ECO command. Setup ECO
Usage: setup_eco($eco_name, @options);
$eco_name: ECO name, like eco01234
@options:
    -help: Print 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');
```

source

Run GofCall script
Usage: source($script_name);

Examples:

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

Note:
It has the same behavior as 'run' command

start_gui

Start GUI windows
Usage: start_gui;

stitch_scan_chain

```
ECO command. Stitch scan chain
Usage: stitch_scan_chain(@options);
@options:
   -to $flop_inst: Stitch all new flops into the flop_inst or stitch each module's new flops into one flop in this module
Note: If -to option doesn't exist, the new flops in each module are connected up in one chain and stitched into one existing scan flop

Examples:

stitch_scan_chain("-to", "abc_reg"); # Insert new flops' scan chain into the existing flop 'abc_reg'
stitch_scan_chain();                 # Stitch the new flops into local scan chains
```

suppress_warnings

```
Suppress warning messages
Usage: suppress_warnings(@messages)
@messages: Warning messages. 'W-001', 'W-002'
```

swap_inst

```
ECO command. Swap two instances with same input/output pins.
Usage: swap_inst($inst1, $inst2);
$inst1,$inst2: Swap these two instances.
Note: $inst1 and $inst2 should have the same input/output pins.

Examples:

swap_inst("spare1/spr_and0", "spare2/spr_and1");
```

undo_eco

```
ECO command. Undo eco operations, restore the database to the original state.
Usage: undo_eco();
```

write_dcsh

```
ECO command. Write ECO result in 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_perl

```
ECO command. Write GofCall ECO script compatible with Perl
Usage: write_perl($gofcall_script_name);
$gofcall_script_name: GofCall ECO script name
Note: The command can be used after 'fix_logic' API. Detail ECO operations are written out.
```

write_soce

```
ECO command. Write ECO result in Cadence SOC Encounter script format
Usage: write_soce($soc_encounter_script_name);
$soc_encounter_script_name: Cadence SOC Encounter script name.

Examples:

write_soce("eco12345.soce");
```

write_spare_file

```
ECO command. Write spare cells list to a 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

```
ECO command. Write ECO result in 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

```
ECO command. Write ECOed netlist to a verilog file
Usage: write_verilog($verilog_file, @options);
@options:
    -help: Print this information
    -all:  Keep the modules in the netlist file even they are not the sub-modules of the top module
$verilog_file: Write 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 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 only write '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 write all modules in ehternet_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 write all modules in ethernet_top.v
set_top("ethernet_top");
write_verilog("ethernet_top_eco.v");
```


6 Appendix B

6.1 GOF Command Options

```
Usage: gof [options] netlists
netlists
   Netlist files to be loaded. There can be multiple netlist files listed,
   if the design has more than one netlist files.
options:
-h
   Print out this info.

-lib
   Provides liberty file (technology library).
   There can be multiple -lib options,
   if the design has more than one technology library files.

-v
   Specifies simulation library file name which has verilog definition
   for leaf gates, like AND2X4.
   There can be multiple -v options, if the design has more than one simulation library.
    -lib should be used unless the leaf cells defined in simulation library are true black box

-vmacro
   For ECO purpose. Each module in the file appears as leaf cell, and it can be
   added like other leaf cell in ECO. When write out ECO netlist, the file content appears
   in the beginning of ECO netlist. And the ECO cell is added as a hierarchical sub-block.

-run
   Provides ECO script name. The script is compatible with Perl syntax.
   GOF stays in shell mode when the script finishes.

-shell
   Runs in text mode with shell prompt, GofCall APIs can be run in interactive mode in shell.

-o
   Specifies log file name, default gatesof.log.

-Top_1
   Specifies another netlist files to build Top_1 tree. The hierarchy will shown up in left
   side of GofViewer window. -Top_2 -Top_3 ... can be used to load more netlist files.
   Note, when this option takes all netlist files followed, so the main netlist files
   should appear before this option. For example,
   'gof -lib tsmc.lib imp_netlist1 imp_netlist2 -Top_1 ref_netlist1 ref_netlist2'
   will create two trees in the left side of GofViewer window.
   While, 'gof -lib tsmc.lib -Top_1 imp_netlist1 imp_netlist2 ref_netlist1 ref_netlist2'
   will build only one tree, since Top_1 option takes up all of the netlist files,
   the main tree is gone.

-ref
   Specifies reference netlist files.
   +define+PARAMETER0+PARAMETER1
   Defines PARAMETER0 PARAMETER1.

-id
   Specifies design name. The name appears on GUI Window tile bar.

-def
   Specifies DEF file (Design Exchange Format).
   There can be multiple -def options,
   if the design has more than one def files.

-defverbose
   Reports all def error, otherwise only first 10 are reported.

-lef
   Specifies Library Exchange Format file.
   There can be multiple -lef options,
   if the design has more than one lef files.

-sparelist
   Specifies spare cells list file.

-parallel
   Define parallel processing CPU Core number.
   Set the number to zero to disable parallel processing.
   By default, the tool picks a optimal number according to the host CPU setting.

-f
   Loads all the files and options in the file_list_file

-session
   Loads saved session

-vcd
   Specifies VCD file for schematic annotation

-textbutton
   Text mode button instead of image mode button in ECO operations

-version
   Prints out current version and exits.

-licquery
   Queries license usage.
```

6.2 Command line Examples

```
gof -lib tsmc.lib soc.v
   Loads one netlist file 'soc.v' and one technology library, 'tsmc.lib'
gof -lib tsmc_std.lib -lib tsmc_io.lib top.v part0.v part1.v
   Loads three netlists, top.v, part0.v and part1.v, two liberty files
   tsmc_std.lib, IO cells, tsmc_io.lib
gof -lib tsmc_std.lib -lib tsmc_io.lib -v analog_models.v top.v part0.v part1.v
   Loads analog cells in verilog library file analog_models.v all analog cells are black boxes.
gof -lib tsmc_std.lib -lib tsmc_io.lib -vn macros.v -v analog_models.v top.v part0.v part1.v
   Loads macros.v as macro cell
gof -lib tsmc.lib -def soc.def.gz -lef libcell.lef soc.v
   Loads Design Exchange Format file soc.def.gz. And library exchange format file for layout view usage.
gof -lib tsmc.lib soc.v -run scripts.pl
   Processes netlist with scripts.pl. Scripts.pl is in Perl syntax and support GOF APIs
gof -lib tsmc.lib top.v netlist.vg -vcd top.vcd
   Loads VCD file for schematic annotation.
gof -lib tsmc.lib imp_netlist.v -ref ref_netlist.v
```


Loads both implementation netlist and reference netlist, can be used in netlist comparison.

7 Appendix C

7.1 Fatal codes

```
F-000: License failed
F-001: Time out in adding ports in hierarchies
F-002: Empty ID for nets
F-003: Pin connections processing fatal error
F-004: Net id not defined
F-005: Net is not in EpHash
F-006: Instance has not been mapped position in AUTO ECO
F-007: Instance has no name mapping in AUTO ECO
F-008: No net found for ECO instance/pin
F-009: Unknown connection type of instance/pin in AUTO ECO
F-010: Net has no name mapping in AUTO ECO
F-011: Failed to initialize database
F-012: MCell get sub-chains error
F-013: No tree has been defined
F-014: No ID for leaf cell pin
F-015: Undefined subroutine in GofCall script
F-016: Global symbol requires explicit package name
F-017: Syntax Error
F-018: Illegal Division by zero
F-019: Bare word not allowed
F-020: Can't locate Perl module
F-021: File size too large for evaluation mode
F-022: Internal error in make miss
```

7.2 Error codes

```
E-001: Reference netlist has not been loaded
E-002: DEF file has missing section
E-003: Command line needs an option for a switch
E-004: Liberty files have not been loaded
E-005: Library cell doesn't exist
E-006: Delete middle bit in a bus
E-007: Unknown command line option
E-008: Win32 doesn't support .gz file
E-009: DEF file doesn't have DIEAREA item
E-010: Files loading sequence
E-011: Instance or pin or port can't be found in module
E-012: Net doesn't exists in module
E-013: Tree name doesn't exist
E-014: Hierarchical module name doesn't exist
E-015: Miss argument
E-016: Module stack is empty, too many pop_top
E-017: 'instance/pin' has wrong format
E-018: Instance or module doesn't exist
E-019: Instance doesn't have pin
E-020: Item is a black box
E-021: Missing DEF file
E-022: No reference for instance
E-023: 'leaf/pin' doesn't exist
E-024: Power connection format is wrong
E-025: Spare cell pattern is not specified
E-026: Spare list file doesn't exist
E-027: 'get_spare_cells' run before 'map_spare_cells'
E-028: 'instance/pin' is floating
E-029: New instance conflicts with existing one
E-030: Specify leaf:num in more than one output leaf
E-031: Instance should be leaf in change_gate
E-032: Syntax error in pin mapping
E-033: The new gate type should be different from the old one in change_gate
E-034: Leaf cell doesn't exist in libraries
E-035: Net doesn't have a driver
E-036: Instance name has special character that the tool doesn't support
E-037: Wrong argument in ECO APIs
E-038: Net has multiple drivers
E-039: Not a port
E-040: New port conflicts with existing one
E-041: Single bit wire can't be expanded to a bus
E-042: New port direction conflicts with existing one
E-043: Commands loading sequence
E-044: Nets in one ECO command should be in the same hierarchy
E-045: Missing scan control pins
E-046: Reference netlist is not loaded
E-047: Fail to open file for write
E-048: Fail to open file for read
E-049: Unable to recognize file format
E-050: Command line option needs a value
E-051: Path doesn't exist
E-052: Leaf should have only one output pin
E-053: New net conflicts with existing one
E-054: Instance ECO result not consistent
E-056: Net has no driver
E-057: Net has invalid BDD
E-059: Not enough resource to run synthesis
E-060: Not valid patch file
E-061: No spare cell for one gate type
E-062: Output port is driven by input port
E-063: Reference register doesn't exist in implementation netlist
E-064: No inverter in the database
E-067: Should add instance into fix_logic argument
E-071: Port doesn't exist in hierarchical instance
E-072: Black box instance doesn't exist in implementation netlist in AUTO ECO
E-076: Spare cells list file has Wrong format
E-080: GOF_KEY_FILE variable has not been defined
E-081: Use '-run' to run Perl script
E-082: Gtech file doesn't exist
E-085: Syntax error in netlist
E-101: No hierarchical path is used
E-102: Interrupt GUI operation by user
E-103: 'read_def' should be run before 'get_spare_cells'
E-105: Load specific file without the right option
E-106: Source ID can't be deleted
E-109: Found combinational loop
E-110: Implementation Netlist has not been loaded
E-112: Can't find pin direction
```


7.3 Warning codes

```
W-001: Bypass already loaded file
W-002: DEF has some section missing
W-003: DEF has module not resolved
W-004: No ECO pin specified for ECO instance
W-005: Not enough spare cells
W-006: DEF file not loaded
W-007: Leaf cell doesn't have timing table
W-023: 'leaf/pin' doesn't exist
W-028: 'instance/pin' is floating
W-038: Net has multiple drivers
W-054: Instance ECO result not consistent
W-055: Net ECO result not consistent
W-056: Net is not driven
W-060: Invalid patch file
W-061: No spare cell for one gate type
W-065: Tie floating input pin to zero
W-066: New port created in AUTO ECO
W-068: Hierarchical cell is created in AUTO ECO
W-069: Set don't touch Warning
W-070: Can't find repeaters
W-073: 'instance/pin' is inverted but being forced to be equal by user
W-074: 'instance/pin' is forced to be inverted by user
W-075: Net returned wrong BDD
W-077: No size information for a leaf
W-078: Module is redefined
W-079: Instance can't be resolved in GTECH
W-080: Leaf cell can't be resolved in module
W-083: Can't read MAC Address
W-084: Sub-module can't be resolved
W-086: Include file doesn't exist
W-087: Bit-width mismatch in instantiation
W-088: Zero fanin endpoint
W-089: Can't find ECO instance position
W-090: Empty instance name in patch file
W-091: ECO net has no fanout
W-092: New input port created and needs to be connected
W-093: New ID created for end point
W-094: Can't detect port phase in module
W-095: Port or net is forced to be equal by user
W-096: Port and net has mismatching bit-width
W-097: Schematic only feature
W-098: Force to use 1'b0/1'b1 in AUTO ECO
W-099: Can't fix timing, since lacking valid points
W-100: No lib name for a leaf cell
W-104: Module is defined as leaf cell but has definition in the netlist
W-107: Module is set as a leaf by user
W-108: Module is not uniquified
W-111: No need to set path prefix
W-112: Can't find pin direction
W-113: Different types of flops in IMP and REF
```

7.4 GUI warning codes

```
GW-001: Don't connect net to a new created connector
GW-002: Don't connect two ECO connectors
GW-003: Don't drive an output port by a cell in different hierarchy in ECO
GW-004: Forward trace a port's driver before insertion
GW-005: Net doesn't exist in design
GW-006: Can't load cell to schematic
GW-007: Trace output pin before delete the gate
GW-008: Can't delete a gate which drives an output port
GW-009: Can't delete a wire which drives an output port
GW-010: Need select a gate to do a operation
GW-011: Can't change ECO gate size
GW-012: No larger size gate in library
GW-013: No smaller size gate in library
GW-014: Connect other side of ECO port first
GW-015: Path is not allowed in port connection
GW-016: Can't disable ECO mode
GW-017: No more ECO operations in undo
GW-018: Need select a pin to do listing endpoints
```