Gates-on-the-Fly fixes
Logic Equivalence Check Failures

Logical Equivalence Checking software like Cadence's Conformal and Synopsys' Formality create detailed reports of differences and errors, but it is often difficult to find, view, and fix the logic cones involved with the errors. SynaptiCAD's Gates-on-the-Fly (GOF) can be used to easily find and view these specific logic cones on a schematic so that you can visualize just the paths you need to see without unnecessary clutter. GOF also simplifies mapping from RTL level constructs to their gate-level equivalents, so that you can pinpoint the locations where changes need to be made. And GOF's ECO mode supports both graphical and script-based editing features for tracking ECO changes. Metal-only ECO operations are also supported with an automatic spare gates flow.

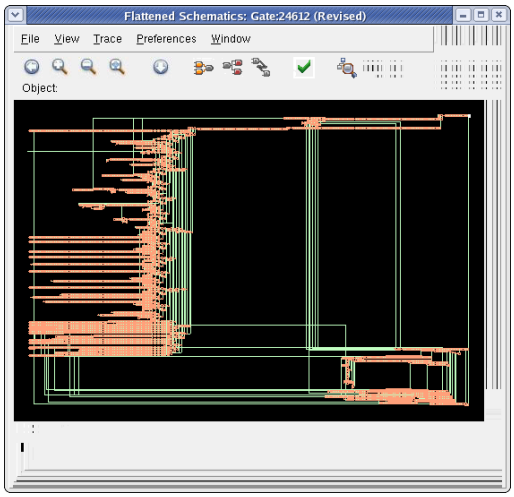
The following whitepaper shows how to use GOF to track down Logic Equivalence Check (LEC) failures identified by Cadence's Conformal LEC tool. The design example discussed in this white paper is from a real world debugging session by a GOF customer.

Conformal Logic Equivalence Check Results:

After running the design through Conformal, the results showed 661 non-equivalent points. Primary analysis showed each non-equivalent point had the same fanin endpoints in both the revised and the golden netlist, so the failure must be inside the logic cone.

Compared points	PO	DFF	DLAT	BBOX	Total
Equivalent	553	42253	98	62	42966
Non-equivalent	0	646	0	15	661

Conformal LEC GUI debug tool gave some useful information, but the schematic displayed too many gates and connections. The culprit gates were hiding somewhere, but it was too difficult to find them in this full-blown schematic.



LEC diagnosis

LEC diagnosis result gave some clues in the potential error candidates, but it was not clear how to use them:

```
Diagnosis for Non-equivalent key points:
 (G) + 1703      DFF  /core/crossbar/cfrg_xhncc_0
 (R) + 10964     DFF  /core/crossbar/cfrg_xhncc_0

Diagnosis points: [CLOCK]
 (G) + 736683    AND  /core/crossbar/FE_RC_10964_0
 (R) + 2279312   BUF  /core/crossbar/FE_OFC2712_xt_xhncc_txd_37_

Non-equivalent signal and its error candidates

   ID (R)    Type         Likelihood  Name
--------------------------------------------------------------------------------
523002      DFF                                      /core/crossbar/cfrg_xhncc_0
------- Candidates ----------------------------------------------------------
1:  113766    BUF_X16M_RVT  1.00      /core/crossbar/FE_OFC75663_scanmode_from_pad
2:  113767    BUF_X3M_RVT  1.00      /core/crossbar/FE_OFC341157_scanmode_from_pad
3:  114588    BUF_X13M_RVT  1.00      /core/crossbar/FE_OFC34939_xt_sel_2_
4:  114601    NOR2XB_X0P7M_RVT  1.00 /core/crossbar/u_mux_p214748365A28
5:  114610    NOR2B_X3M_RVT  1.00      /core/crossbar/u_mux_p214748365A29
6:  114825    INV_X13M_RVT  1.00      /core/crossbar/FE_OFC11881_jtag_mode
7:  117468    INV_X2P5M_RVT  1.00      /core/crossbar/u_gpio_p214748365A20030
8:  114592    BUF_X3M_RVT  0.99      /core/crossbar/FE_OFC339934_scanmode_from_pad
9:  114593    INV_X13M_RVT  0.97      /core/crossbar/FE_OFC75793_scanmode_from_pad
```

Note: two instances have been marked in red. These will be the two tracing points for Gates On the Fly.

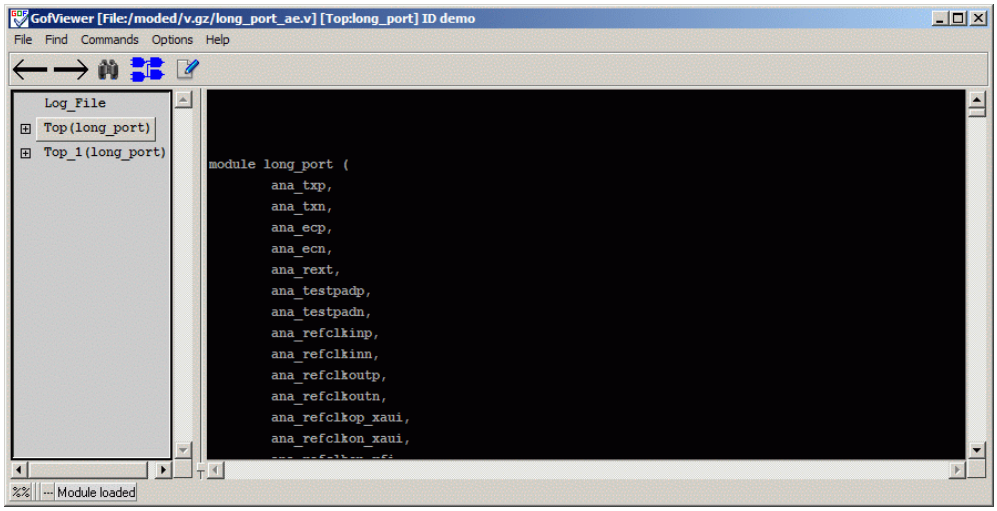
The GOF way

To track down the offending gates we loaded both the revised and golden netlists into GOF using the following command:

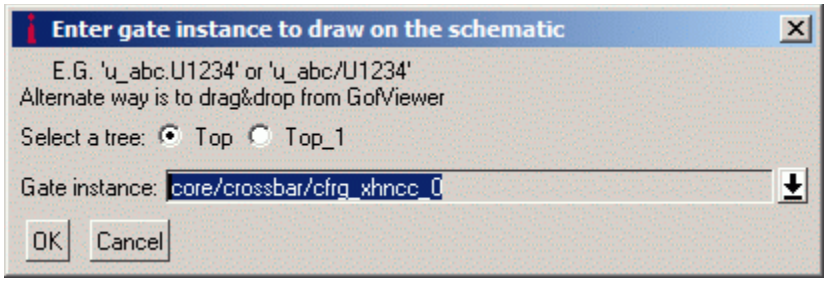
```
gof –lib hvt.lib –lib rvt.lib –Top_1 golden_netlist.v revised_netlist.v
```

- **-lib** option loads library files
- **-Top_1** option specifies the golden netlist to load. Additional netlists can be loaded by “-Top_2” and so on.
- The revised netlist filename has no option before it.

GOF first displays the net list code in a text viewing window called GofViewer. The left hand side has the two netlists' hierarchies that were loaded into GOF. The right hand side lists the modules of the selected netlist.

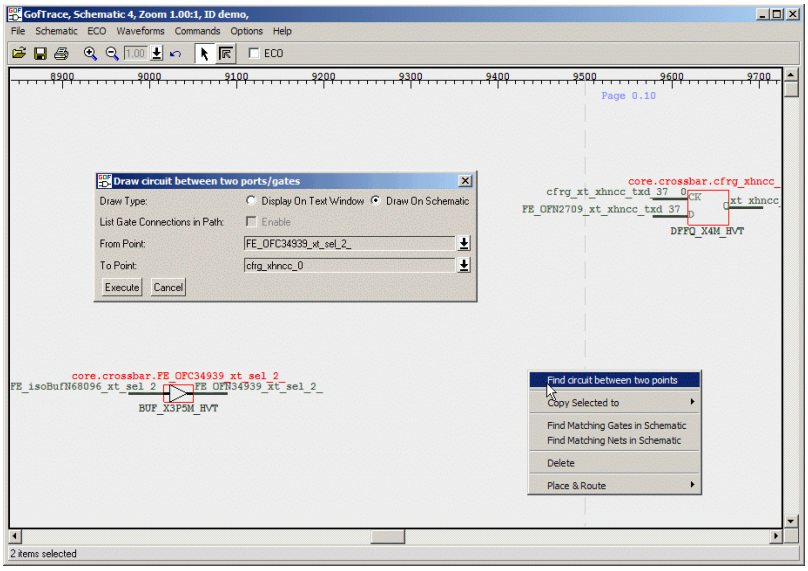


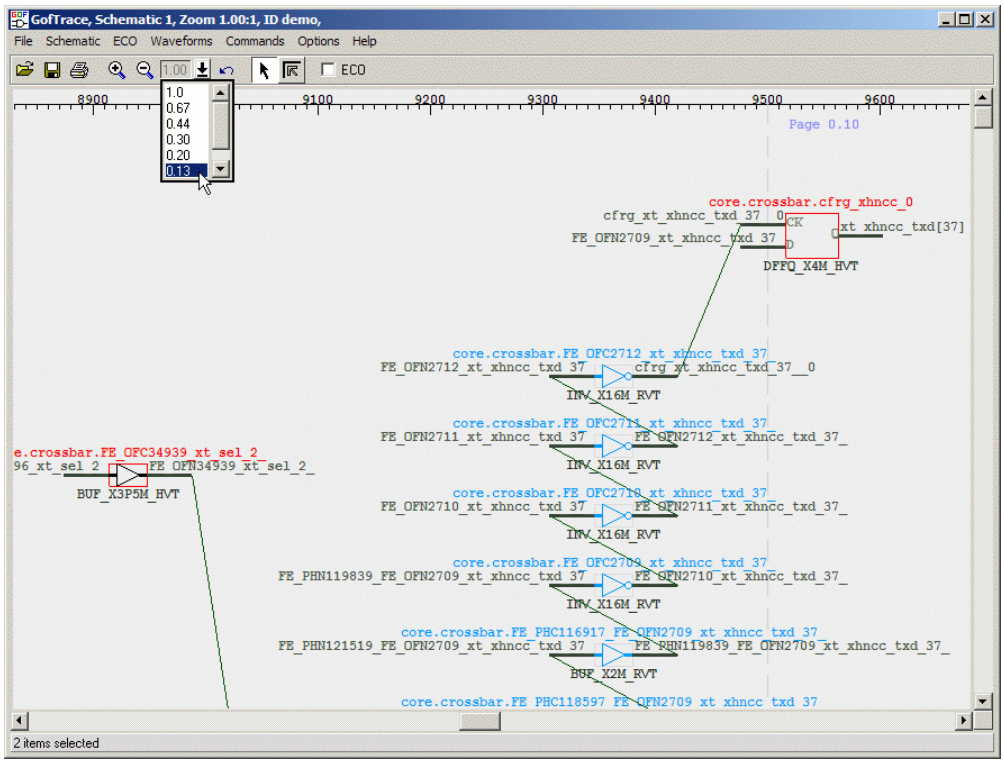
With GofViewer, the design can be quickly explored by double clicking on the tree and viewing the code. The right click context menus and double clicking on objects lets you find related information such as the drivers and loads of a particular net. But since we already knew some gate instance names from the LEC report, we just pressed the CTRL-G key combination to open a search dialog and typed in the instances to show on the schematic. We loaded two gates onto the schematic: **core/crossbar/cfrg_xhncc_0** and **core/crossbar/FE_OFC34939_xt_sel_2_**.



Draw Schematic between two parts

The GofTrace interface is optimized to quickly display only the portion of logic that you are interested in by focusing on tracing just the fan in/out of a particular path. If we were just investigating one particular path, we could expand the schematic by ***middle mouse clicking*** and dragging on a port to see more of the schematic. In this case, we wanted see the entire path between the two parts and this can be done using a context menu command. As shown below, we selected the two gate instances and then right clicked and selected **Find Circuit between two points** from the context menu. This opened a dialog where we chose to show the circuit on the schematic and then executed the function.

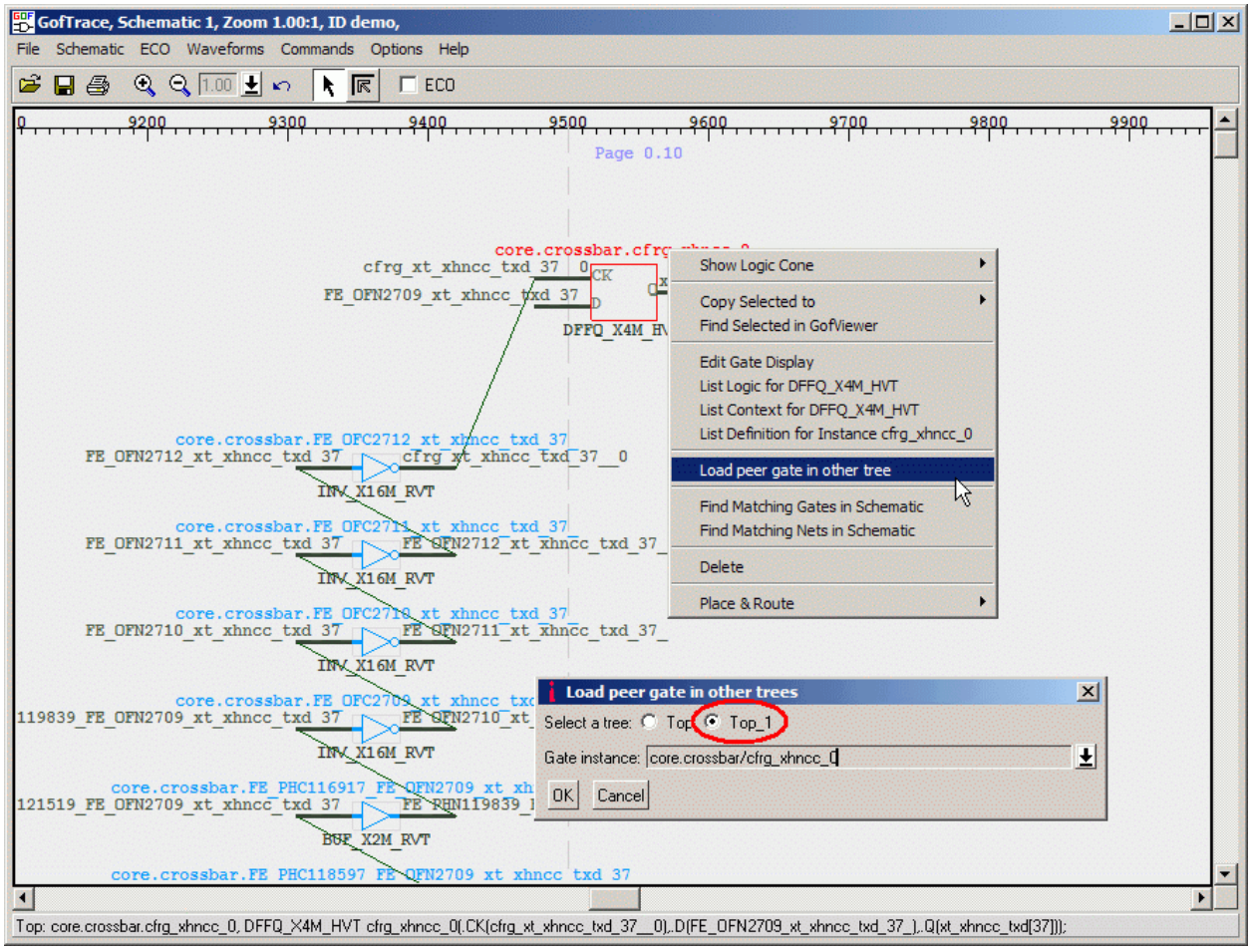




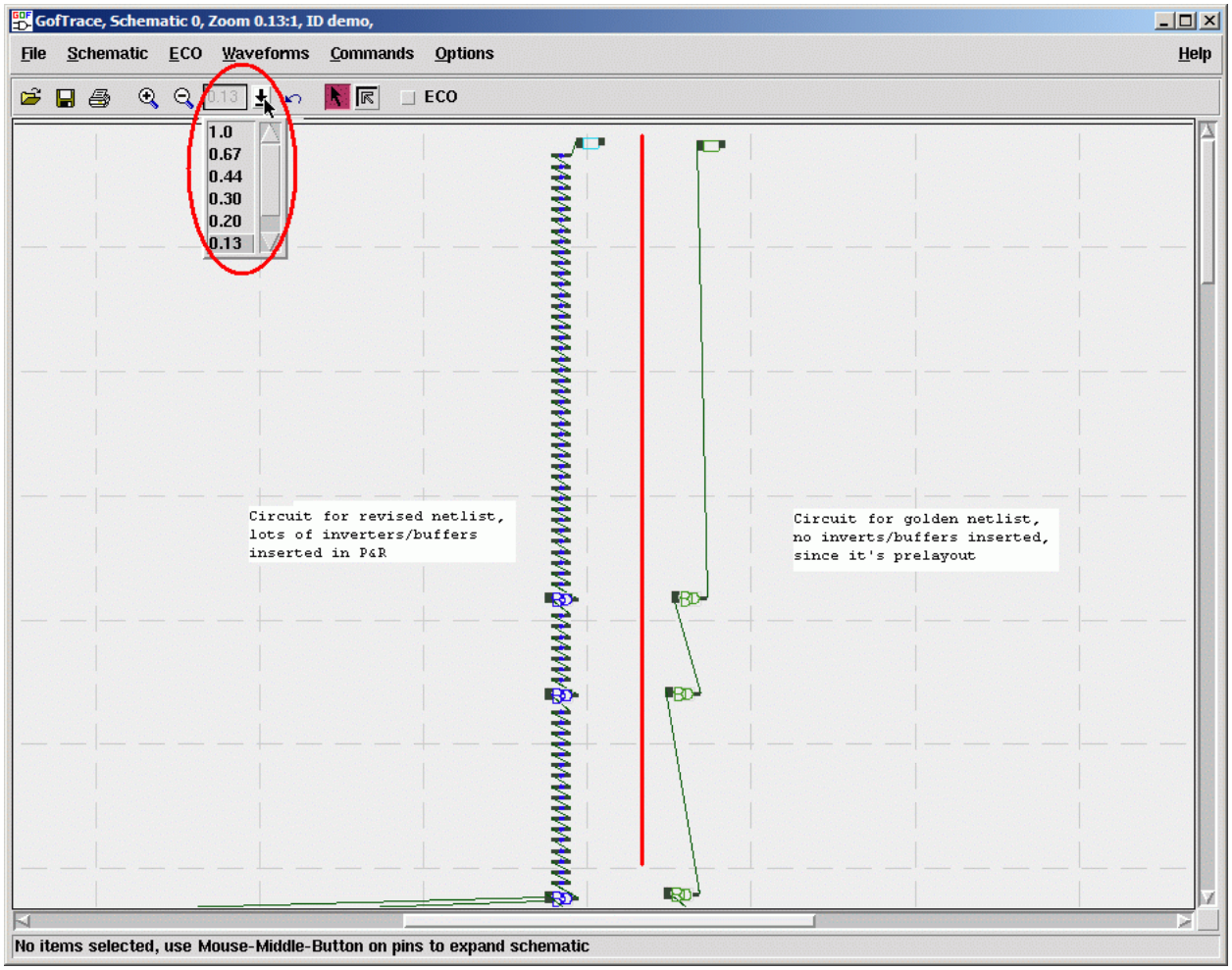
The schematic display is fully customizable, so you can control what types of objects are displayed, how the logic is drawn, and the colors for different objects. Comments can also be added to a gate or to the schematic sheet. The schematic can be zoomed in/out to view the whole circuit or a particular area. Any cell on the schematic can be moved around to a new proper position for easy viewing by left clicking and dragging. Another useful tracing feature during LEC analysis is **Show until non-buffer on Schematic**. This renders all driving buffers/inverters until a non-buffer or non-inverter cell is encountered on the selected path.

Create same path for golden netlist

Next we want to add the same path from the golden netlist to the schematic viewer window so that we can visually compare the differences. We can do this either from the GOFViewer or from the GofTrace windows. In GofViewer, we would select the golden netlist tree and use Ctrl-g to load instances from that netlist like we did in the previous section. In GofTrace, we just select an instance, right click, and choose **Load peer gate in other tree** from the context menu. This opens a dialog where you can choose the golden netlist by selecting 'Top_1' for the other tree.

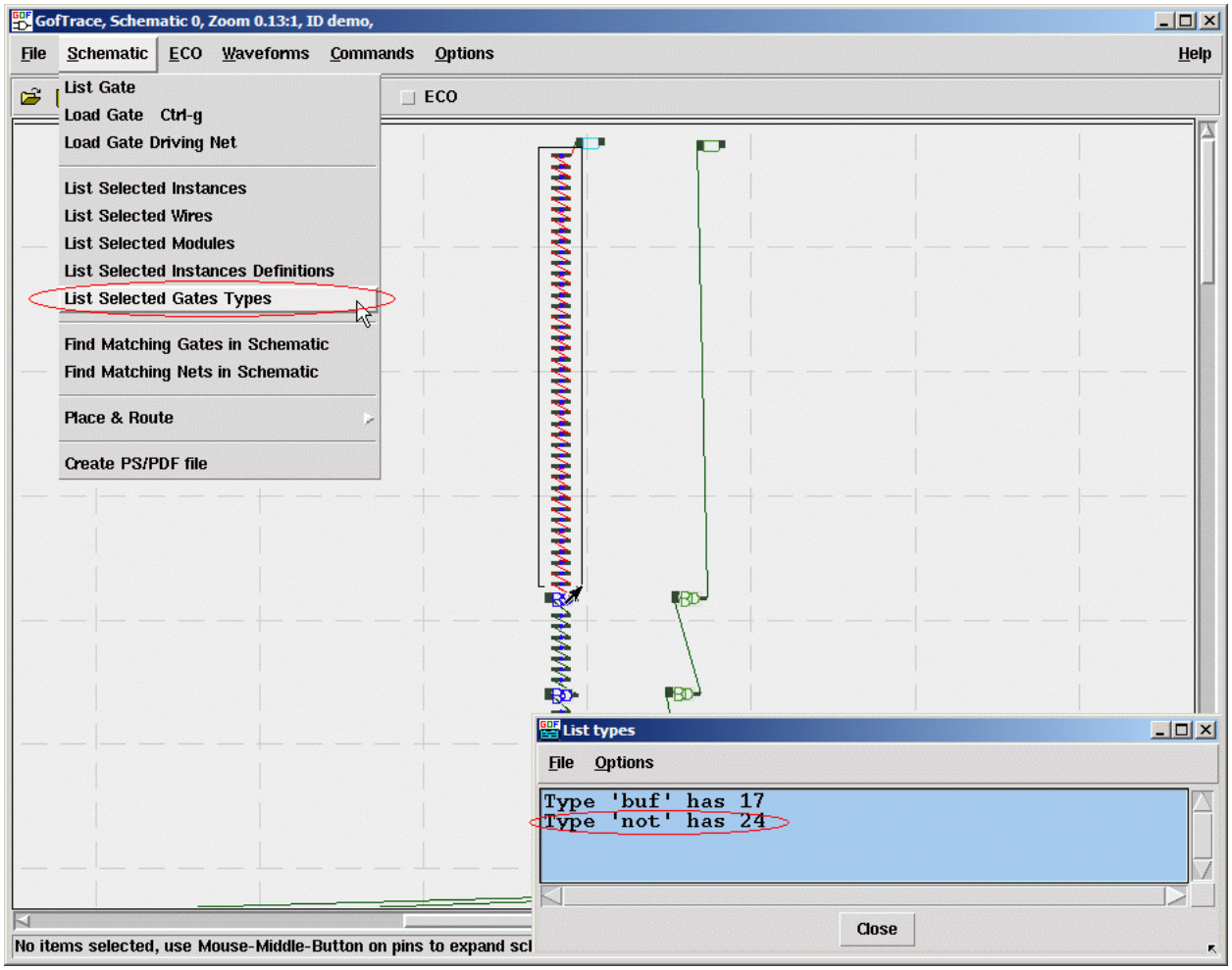


This will load one gate from the golden netlist. More gates can be added by *mouse middle button clicking* on the gate's input/output pins. In a few clicks, we have the two circuits displayed side-by-side. Since all the non-buffer/inverter paths are the same, the mismatch must be in the buffer/inverter paths.



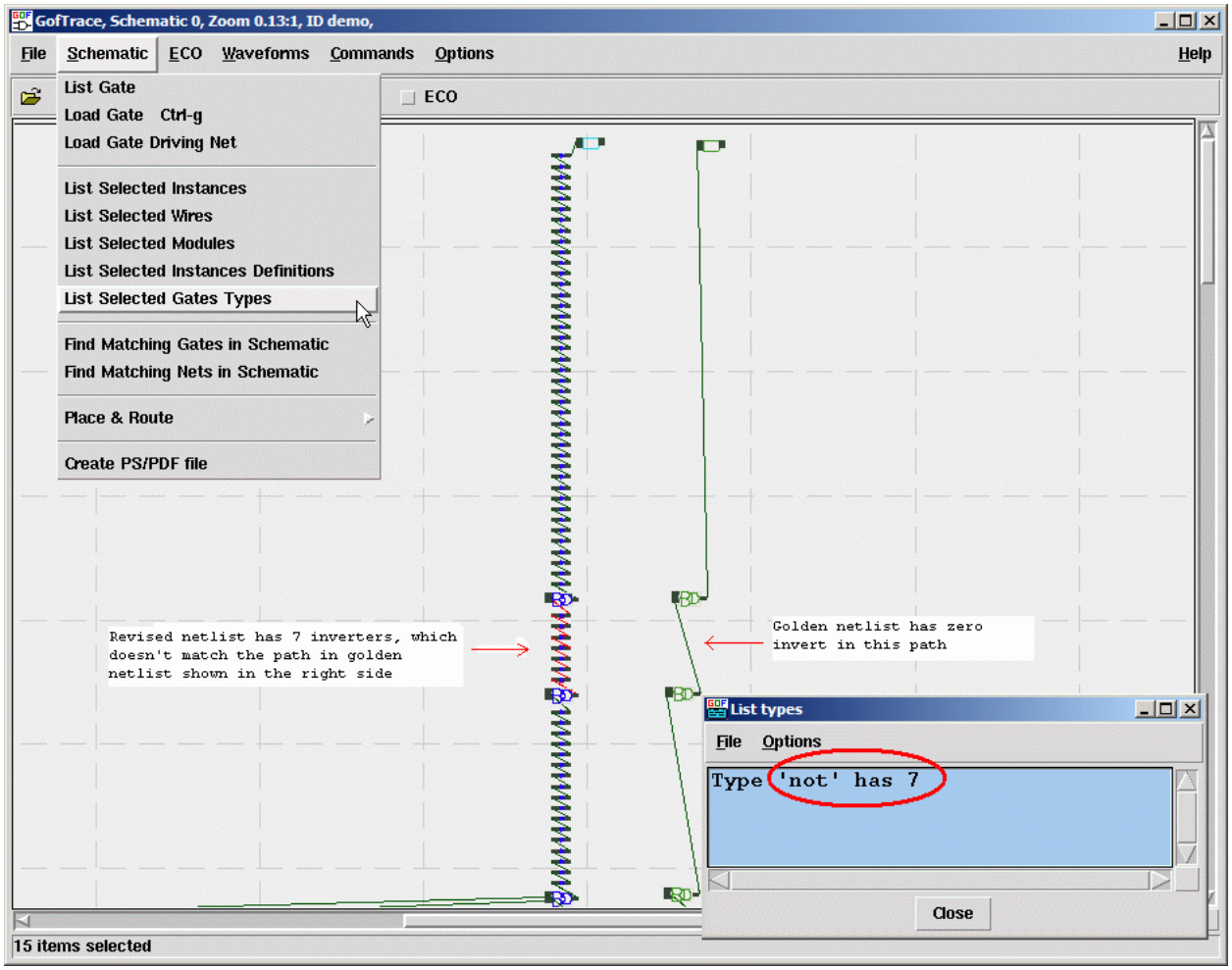
Analyze segment by segment

Select the first group of buffers/inverters driving the sink by pressing *left mouse button* near the top of the chain and dragging it to the end of the chain. When all the cells in the group are selected, choose the **Schematic >List Selected Gates Types** menu. This opens the ***List types*** dialog which shows that there is an even number of inverters which means it matches the path in the golden netlist.

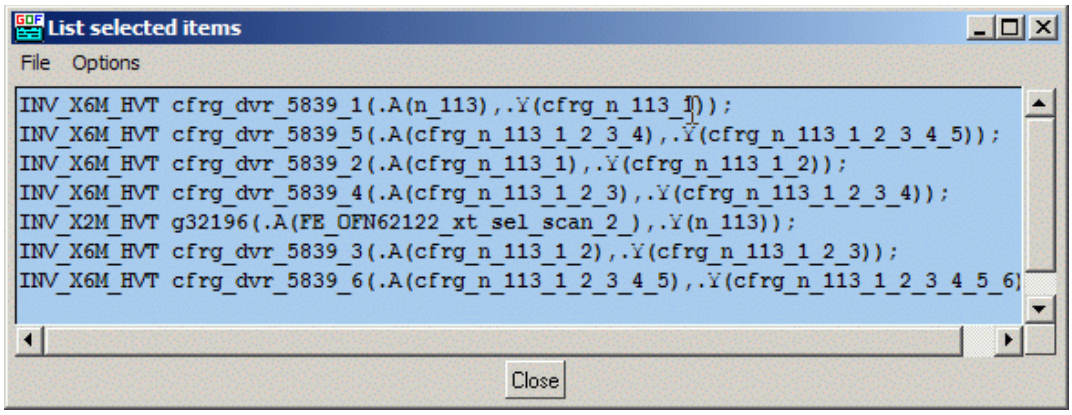


Catch the issue

Using the same analysis technique on the second set of buffers, we find that the inverter count is 7, which doesn't match golden path! The revised netlist from Physical Design Team has gone through several ECO scripts, so some ECO operations must have swapped the gate type.

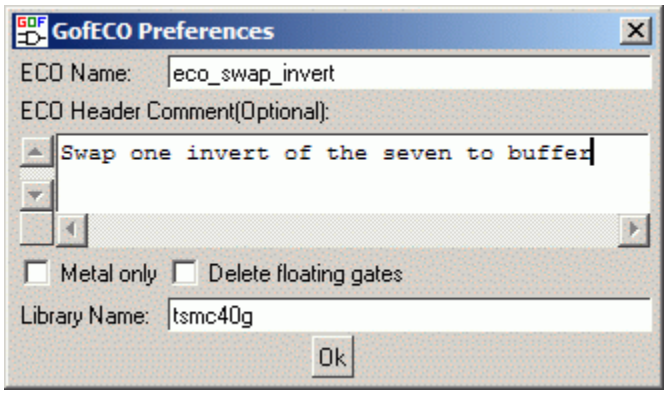


To get a list of the 7 problematic inverters to send to the Physical Design Team, we used the **Schematic >List Selected Instances Definitions** menu, to open a text dialog with the information.

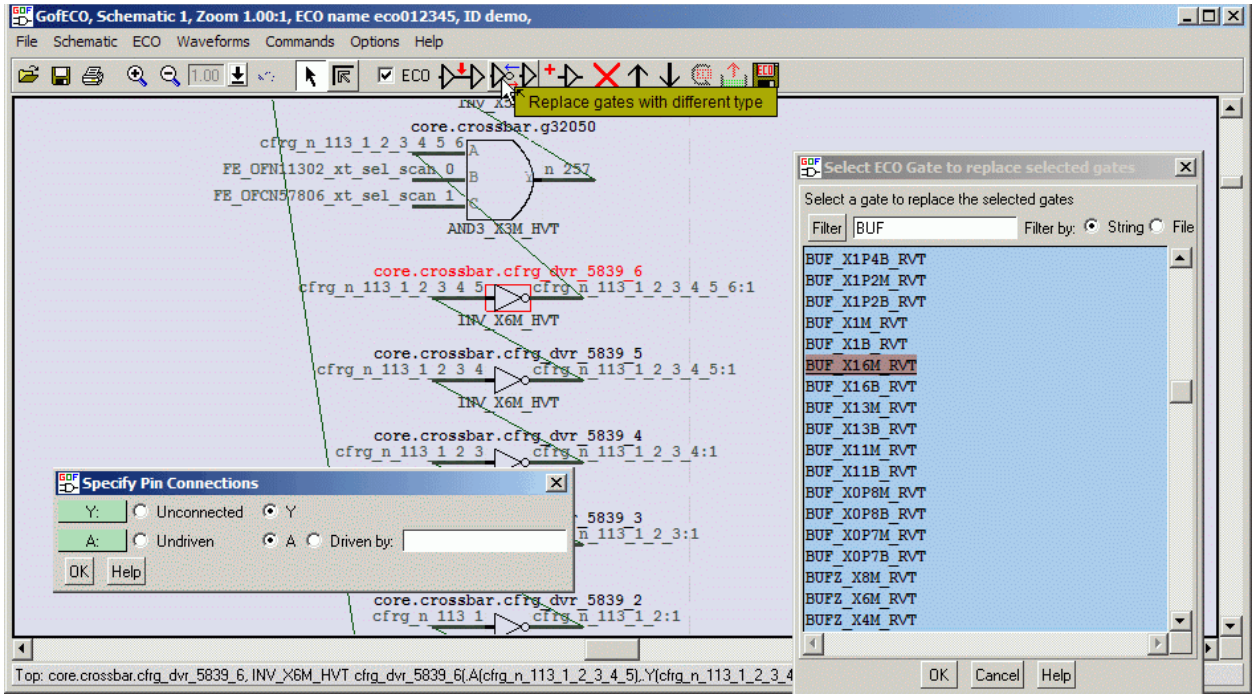


Experimental fix analysis and GOF ECO

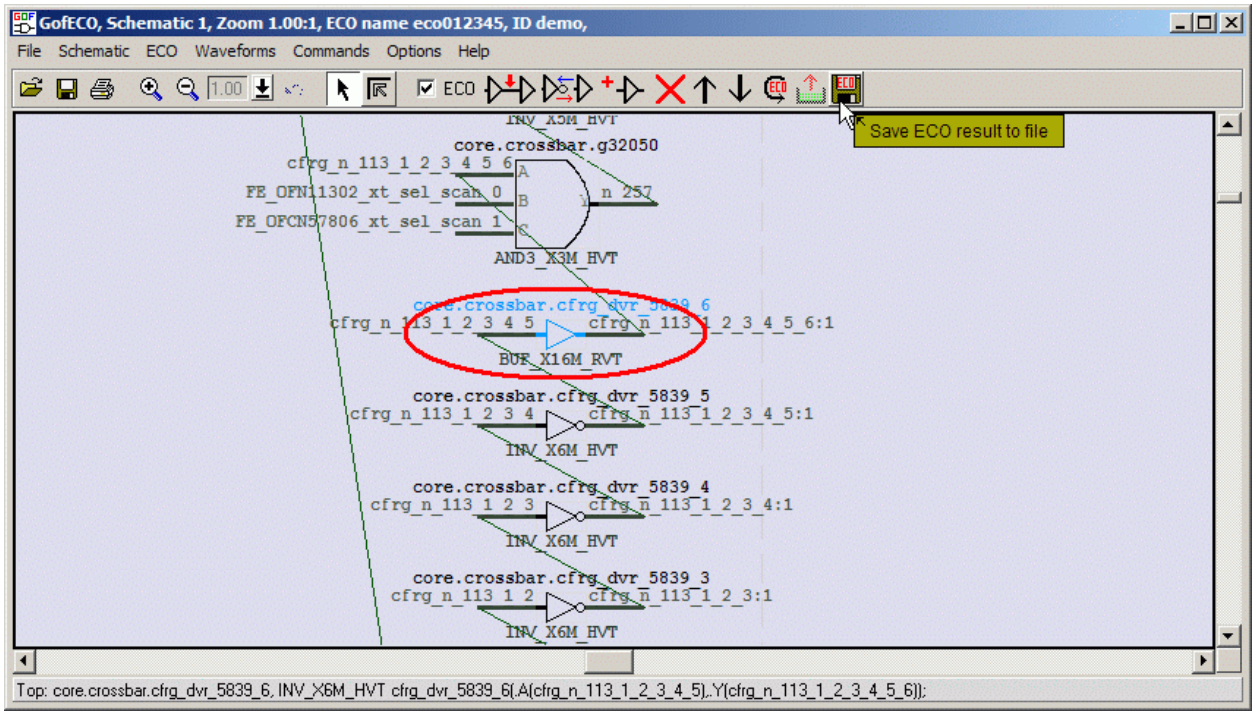
GOF can also be used to perform the ECO and fix the error. Since the inverters all have only one fanout, we can swap in one inverter for a buffer to fix the logic error. Click the **ECO** button in GofTrace window to open the ***GofECO Preferences*** dialog. This lets you control the type of ECO, the prefix for new gates added to the design by the ECO, and some other settings.



To perform the ECO, select one of the seven inverters, then press the **Replace Gates with different type** button. This will open a series of dialogs that will let you pick the new gate and control the connections of the gate. We picked **BUF_X16M** and used the default connections.



When the ECO is performed, the new gates are displayed in a different color, so it is easy to see how the change affected the original circuit. Press the **Save ECO** button to save the ECO result to a Verilog netlist, a SOC Encounter ECO script, a Synopsys TCL format, or other supported format.



After performing the ECO, we would run another round of LEC to see if the ECO netlist has fixed some failures. The debugging/fixing process normally has multiply iterations.

Script mode ECO

After several iterations, the total failures were down to 92 points and all of them were flip-flop clock inputs. We decided to use GOF's script mode ECO to insert an inverter before each clock pin of the flip-flops. We wrote the ECO script below and we dumped the failing end points to a file named "**non_eq.pnt**" which was processed by the ECO script to extract the flip-flop instance names.

```
1.  # File name: insert_invs.pl
2.  use strict;
3.  undo_eco;
4.  open(FIN, "non_eq.pnt");
5.  open(FOUT, ">fix_92.for_soc");
6.  while(<FIN>){
7.      my ($flop) = (m/(\w+_reg(_\d+)?)/);   # Get flop instance name
8.      print "$flop\n";
9.      my @pins = get_pins("-input", "$flop");  # The clock pin can be CKN or CK
10.     print "@pins\n";
11.     if(grep($_ =~ m/CKN/, @pins)){
12.         change_pin("$flop/CKN", "INV_X16M_RVT", "", "-");
13.         print FOUT "ecoAddRepeater -term $flop/CKN -cell INV_X16M_RVT\n";
14.     }else{
```

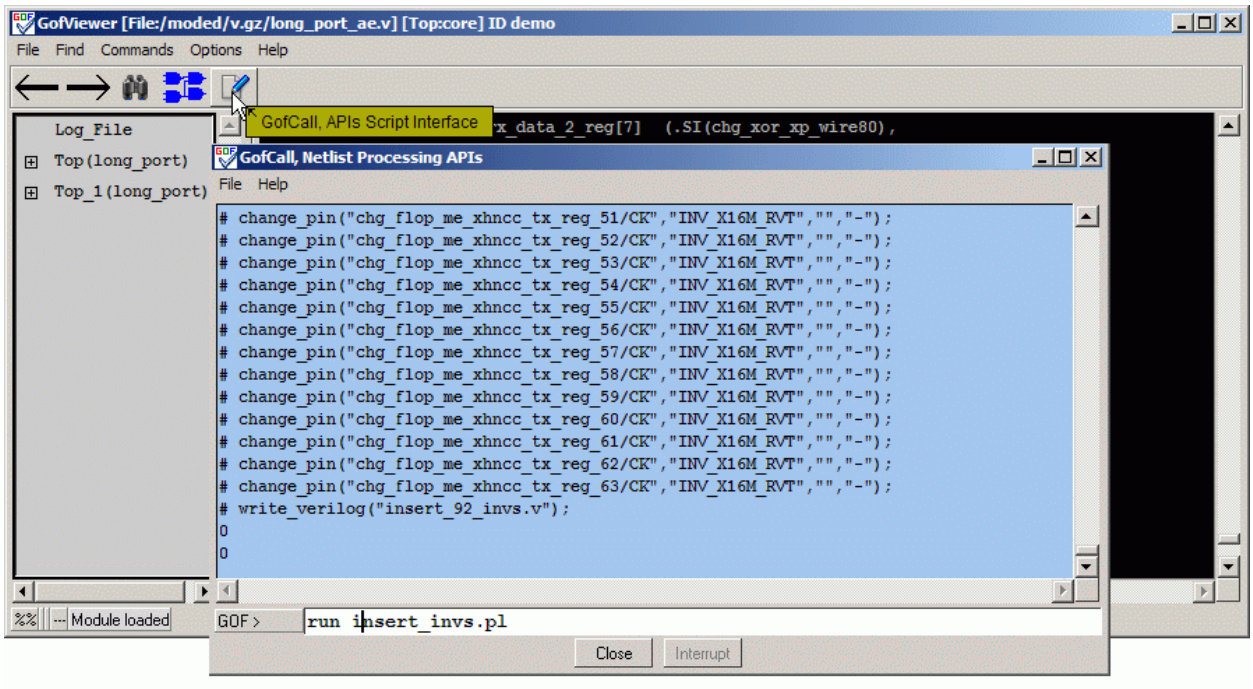
```
15.             change_pin("$flop/CK", "INV_X16M_RVT", "", "-");
16.             print FOUT "ecoAddRepeater -term $flop/CK -cell INV_X16M_RVT\n";
17.         }
18.   }
19.   close(FIN);
20.   close(FOUT);
21.   write_verilog("insert_92_invs.v");  # Save the ECO result to verilog netlist
```

In the script file, we used several GOF APIs: undo_eco, get_pins, change_pin, write_verilog. These are documented in the online help under the GofCall chapter. The script can either be run in a GofCall window by typing in the script command:

```
run insert_invs.pl
```

Or it can be executed from an OS shell by starting GOF with the "-run" option:

```
gof -lib libs.lib netlists.v -run insert_invs.pl
```



After running the script, we re-ran the LEC check on the ECO'd netlist "insert_invs.v" against the golden netlist and found no more mismatches.

Conclusion

GOF's rich schematic features make it a good choice in debugging complicated equivalence check failures. The built-in "On the fly" ECO functions decreases turn-around time for large netlist processing.

GOF's Main features:

- Flexible and partial schematic display makes it easy to isolate only the related path
- Built-in functions like tracing path from point to point and logic cone extraction
- ECO in GUI mode or script mode
- Logic cone replacement for large netlist changes
- Netlist tracing by using exported APIs
- Add annotations to schematics and PDF for easy documentation
- Prime Time report file analysis and timing fix automation
- Placement view for timing path analysis with each segment highlighted with different color