

Adding a module in netlist by using Gates On the Fly

Contents

Introduction	1
Prepare the netlist	2
Analyze the ECO changes.....	3
ECO on schematic	4
Load ECO gates	4
Connect up cells.....	6
Save ECO result	8
Do ECO by Perl script calling GofCall APIs.....	8
ECO script and run command	8
Script dissection	9
Debug and verification method	10
Bring up GUI.....	10
Run script in GUI window.....	11
Interactive with schematic.....	11
Conclusion.....	12

Introduction

One common scenario in netlist ECO is to add a small re-synthesized module. The module can be a multiplier, adder or other circuit missed in the original netlist.

Figure 1 shows one use case which has a new block inserted to an existing bus by multiplexers

Gates On the Fly Use Case: Add a module in netlist ECO

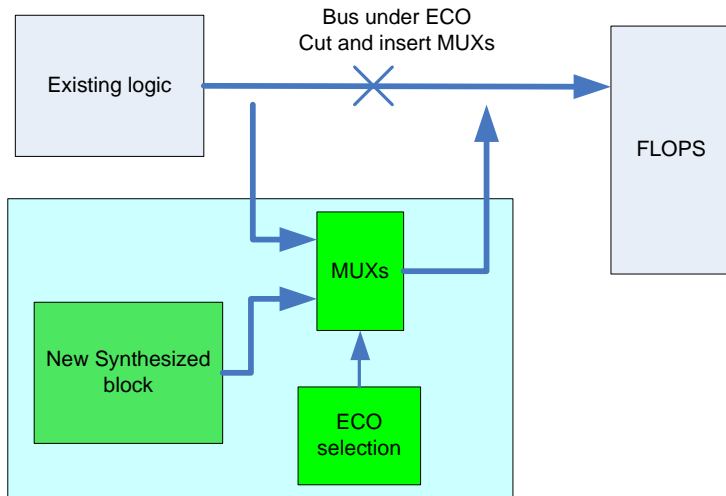


Figure 1

The following paragraph will show how to do ECO in GUI mode and in script mode by Gates On the Fly.

Prepare the netlist

Create a tiny RTL file with only the necessary logic. In this case, it's an incrementor-by-one module. If the green box 'ECO selection' in figure 1 has complicated logic, it can be added into the tiny RTL module as well. Now assume it is simple enough.

```
module eco2135_DW01_inc_1(A,SUM);  
input [14:0] A;  
output [15:0] SUM;  
assign SUM = A + 1;  
endmodule
```

After synthesis, the netlist is generated which has 50 gates. Save the netlist to mymacro.v

```
module eco2135_DW01_inc_1 ( A, SUM );  
input [14:0] A;  
output [15:0] SUM;  
wire n19, n20, n26, n27, n28, n31, n37, n41, n42, n45, n46, n50, n55, n56,  
n60, n61, n64, n65, n66, n70, n71, n76, n77, n79, n95, n96, n97, n98,  
n99, n100, n101, n102, n103, n104, n105, n106;  
NAND2X4 U39 ( .A(n46), .B(n27), .Y(n79) );  
XOR2X2 U80 ( .A(n77), .B(A[14]), .Y(SUM[14]) );  
AND2X4 U81 ( .A(A[4]), .B(A[5]), .Y(n95) );  
CLKIN VX16 U82 ( .A(n95), .Y(n37) );  
NOR2X2 U83 ( .A(n28), .B(n41), .Y(n42) );  
NAND2X2 U84 ( .A(A[8]), .B(A[9]), .Y(n56) );  
NOR2X2 U85 ( .A(n79), .B(n66), .Y(n103) );
```

Gates On the Fly Use Case: Add a module in netlist ECO

```
XOR2X1 U86 ( .A(n101), .B(A[6]), .Y(SUM[6]) );  
XOR2X1 U87 ( .A(n102), .B(A[10]), .Y(SUM[10]) );  
...  
endmodule
```

Analyze the ECO changes

Start up GOF to load the netlist under ECO by the following command

```
gof -lib art.lib TM.gv
```

Find the related signal for 'ECO selection', in this case the MUXs selection signal is 'qciflt_mode'. Click the signal to mark it, click 'New Schematic' button to bring up a schematic.

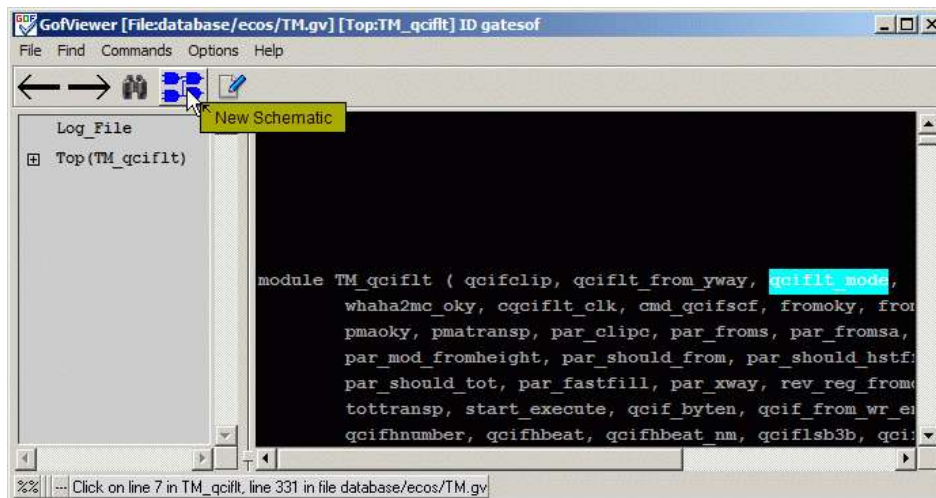


Figure 2

Press ctrl-g to load flops involved in this ECO. Wild card is acceptable, key in 'qcifvnumber_reg_*' and click OK.

Gates On the Fly Use Case: Add a module in netlist ECO

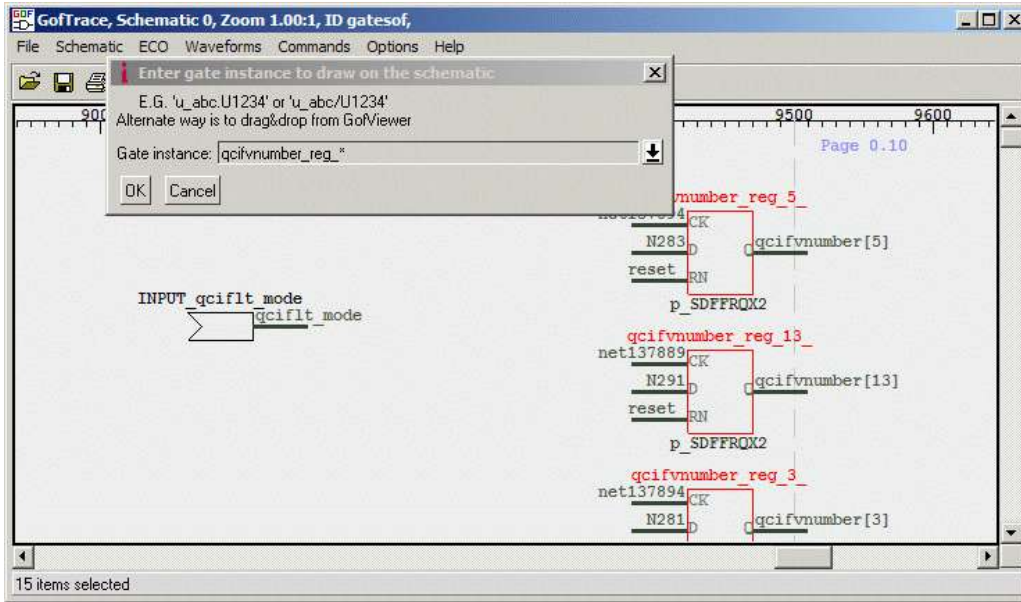


Figure 3

The new block will be inserted to the 'D' pins of the flops with MUXs.

ECO on schematic

Load ECO gates

Use mouse middle button click on D inputs of the flops to expand the schematic. Click 'ECO' checkbox to enable ECO.

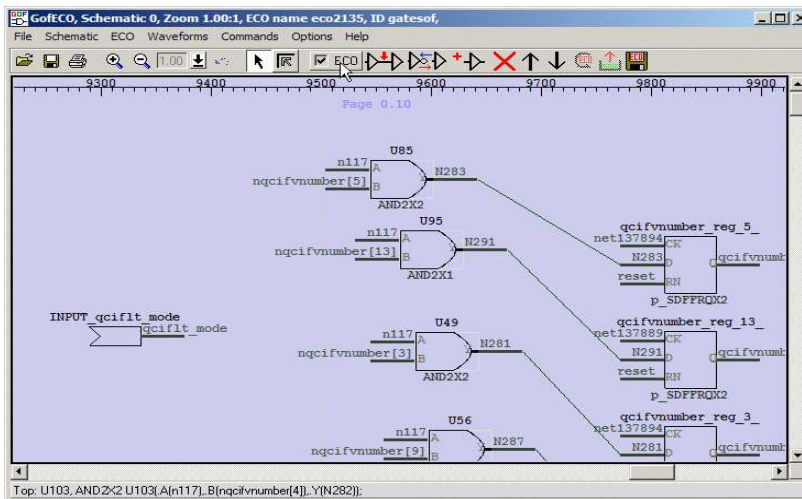


Figure 4

Press 'Add gates' button to add the synthesized block 'eco2135_DW01_inc_1' as a leaf cell.

Gates On the Fly Use Case: Add a module in netlist ECO

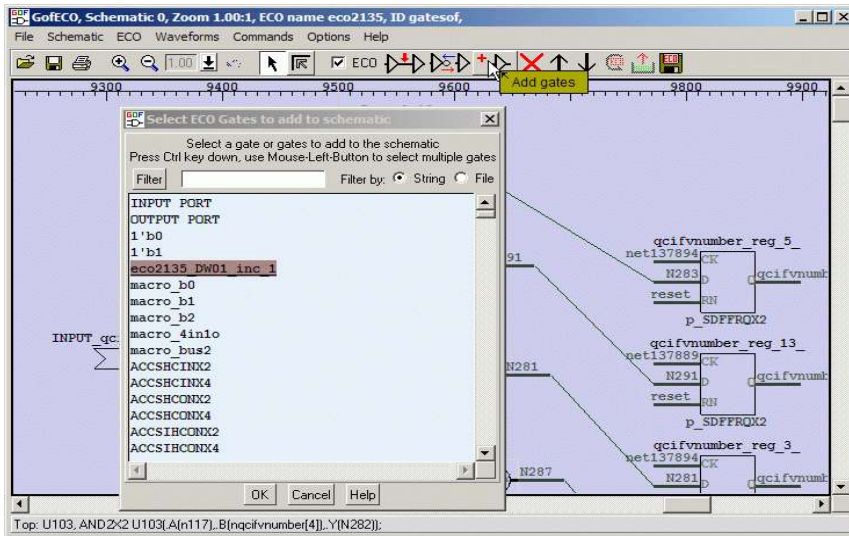


Figure 5

Select all lines to 'D' inputs. There are two ways to select multiple lines.

1. Press 'ctrl' key and mouse-left-click on the lines.
2. Press mouse-left-button and scroll down to select all the lines and release the mouse button.

Click 'Insert gates into connections' button and select 'MX2X4', click OK. Click OK in the next 'Specify pin connection' window

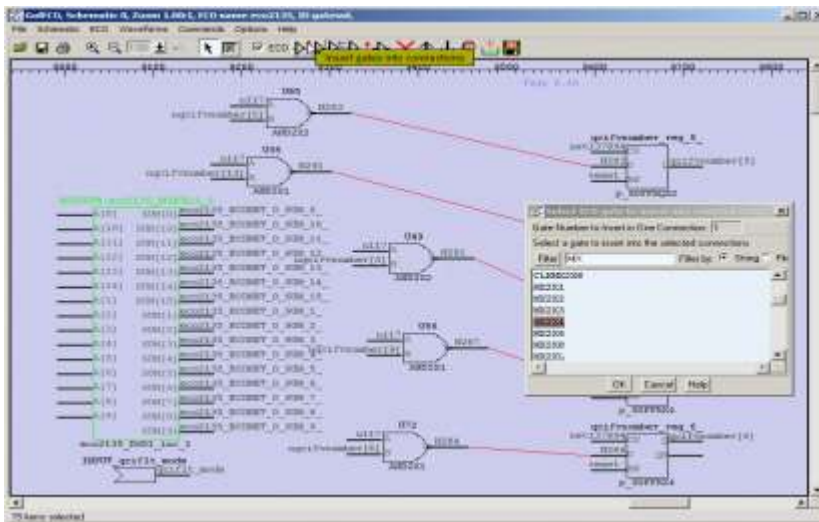


Figure 6

Figure 7 shows the re-synthesis blocks and MUXs are loaded into the schematic. Now it's time to connect them up.

Gates On the Fly Use Case: Add a module in netlist ECO

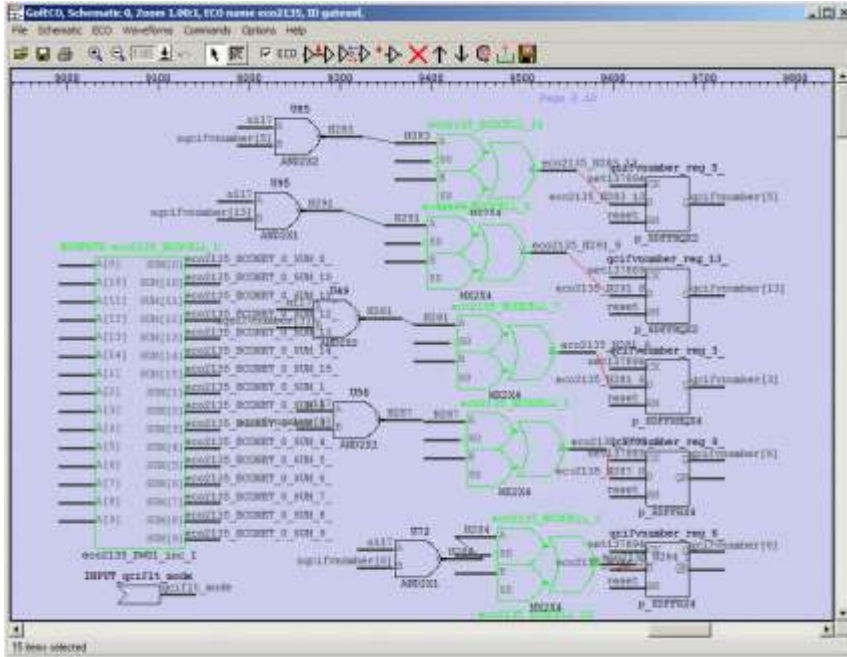


Figure 7

Connect up cells

The connections to be done are listed as below:

- Connect MUXs' S0 pins to qciflt_mode as the 'ECO selection' green box shown in Figure 1.
- Connect SUM output pins of the ECO block to MUXs' B pins.
- Connect FLOPs' Q pins to input pins of ECO block.

To connect up pins, use mouse middle button to press on the floating input pin, don't release, move the mouse to the proper output pin and release the mouse. A wire connects up the output pin to the input pin. And the input pin will be assigned the same net name as the output pin.

Figure 8 shows how to connect B pin of a MUX to SUM[10] pin of the re-synthesized block.

Gates On the Fly Use Case: Add a module in netlist ECO

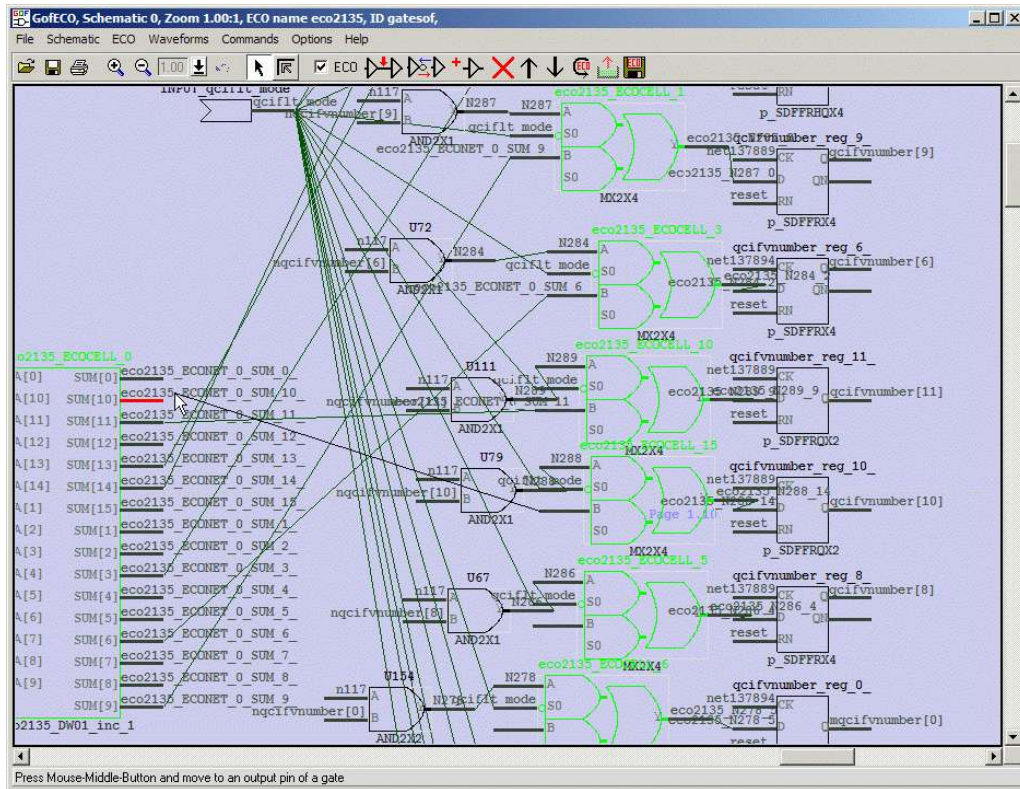


Figure 8

Any gate on the schematic can be moved around for convenient connecting. Figure 9 is the final schematic with all pins connected up.

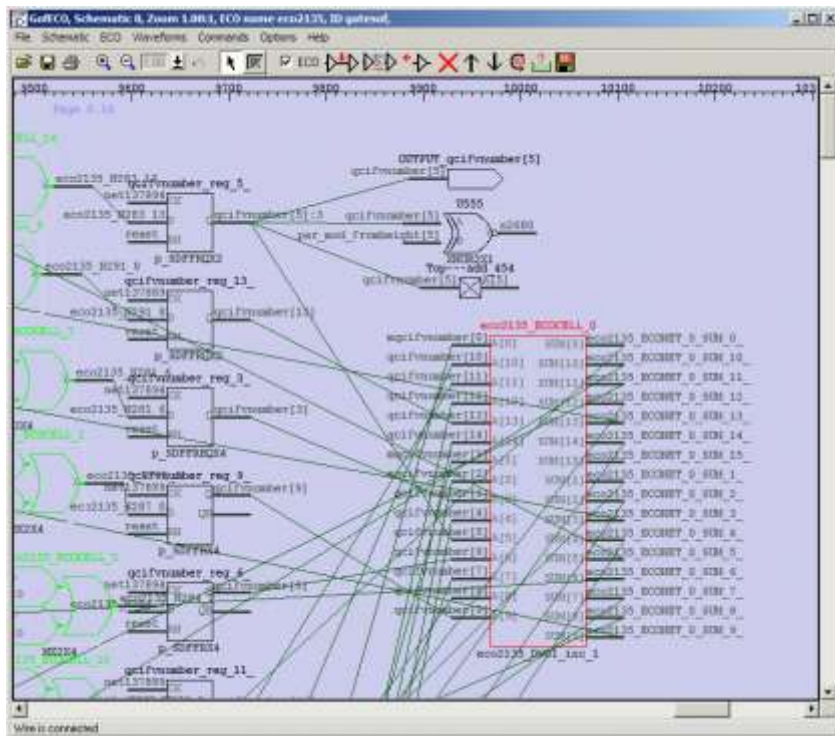


Figure 9

Save ECO result

Click 'Save ECO result to file' button to save ECO in verilog netlist. Currently these formats are supported:

- Verilog netlist
- SOC Encounter ECO script
- GofCall Perl script
- TCL script
- DCSHELL script

However, 'vmacro' option only supports verilog netlist written out.

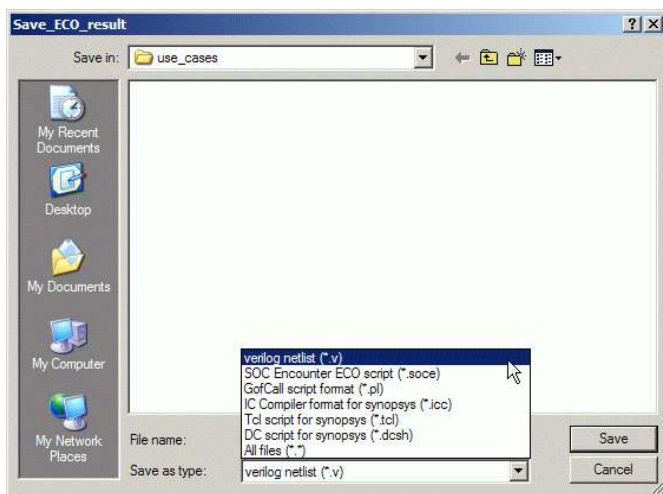


Figure 10

Do ECO by Perl script calling GofCall APIs

ECO script and run command

It's a little tedious to manually connect up the pins in GUI mode especially when the ECO gates number is large. It is more efficient to use Perl script calling GofCall APIs.

```
1. use strict;
2. undo_eco();
3. setup_eco("eco2135");
4. my $coinst = "eco2135_DW01_u";
5. new_gate("", "eco2135_DW01_inc_1", $coinst, "");
6. for(my $index=0;$index<15;$index++){
7.   change_pin("qcfvnumber_reg_${index}_/D", "MX2X4", "eco2135_mux_${index}", ".A(-).B().S0()");
8.   change_pin("eco2135_mux_${index}/S0", "qcfilt_mode");
9.   change_pin("eco2135_mux_${index}/B", "$coinst/SUM[${index}]");
10.  change_pin("$coinst/A[${index}]", "qcfvnumber_reg_${index}_/Q");
11. }
12. write_verilog("eco2135_net.v");
```


Gates On the Fly Use Case: Add a module in netlist ECO

Save the script to a file, name it as macro.pl. Run Gates On the Fly with the script by the following command. Please note `-vmacro` option is used.

```
gof -lib art.lib TM.gv -vmacro mymacro.v -run macro.pl
```

After the run, ECO netlist `eco_2135_net.v` is created.

Script dissection

```
1. use strict;
```

Use strict syntax check

```
2. undo_eco();
```

Always undo the previous ECO operation, since the script is normal run several times after the database is loaded

```
3. setup_eco("eco2135");
```

ECO name, new nets and instances automatically created by GOF will have ECO name as prefix

```
4. my $ecoinst = "eco2135_DW01_u";
```

```
5. new_gate("", "eco2135_DW01_inc_1", $ecoinst, "");
```

Create a new instance for the re-synthesis block. The block is treated as a leaf cell. The API `new_gate` has the first argument defined as new net name, the second as module name, the third as instance name, and the fourth as connection. For the detail usage of GofCall APIs, type 'help' in GOF shell. It is covered in the other section below.

```
6. for(my $index=0;$index<15;$index++){
```

Start of the loop, it runs 15 times to cover the 15 ECO points

```
7. change_pin("qcfvnumber_reg_${index}_/D", "MX2X4", "eco2135_mux_${index}", ".A(-).B(),.S0()");
```

The API `change_pin` has two usages. Two arguments and Four arguments. Two arguments have the first one as instance/pin, and the second one as net or instance/pin. Four arguments have the first one as instance/pin, the second as leaf cell name, the third as the instance name and the fourth as the connection. Note, '-' in the connection means taking the original wire connection of the first argument "qcfvnumber_reg_\${index}_/D". The connection can be further simplified as ".,." by omitting '.A()' '.B()' and '.C()'

Line 7 is equivalent to these two lines.

```
my $net = get_net_of("qcfvnumber_reg_${index}_/D");
```

```
change_pin("qcfvnumber_reg_${index}_/D", "MX2X4", "eco2135_mux_${index}", ".A($net).B(),.S0()");
```

```
8. change_pin("eco2135_mux_${index}/S0", "qcfilt_mode");
```

Connect MUX's S0 pin to ECO selection 'qcfilt_mode'

```
9. change_pin("eco2135_mux_${index}/B", "$ecoinst/SUM[${index}]");
```

Connect MUX's B pin to the corresponding output pin of the re-synthesized block

```
10. change_pin("$ecoinst/A[${index}]", "qcfvnumber_reg_${index}_/Q");
```

Connect the re-synthesized block's input pin to the corresponding flop's Q output

```
11. }
```

```
12. write_verilog("eco2135_net.v");
```

Save the ECO in verilog, the re-synthesized macro module is written out in the beginning of the file.

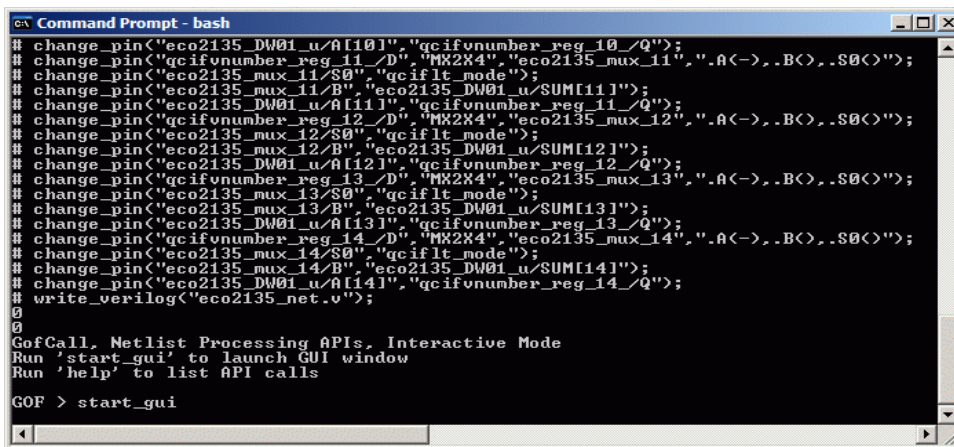
Debug and verification method

Normally, you need several iterations to make a script to work. The best practice is to run the script in GUI mode and interact with schematic.

Bring up GUI

After the command line “gof -lib art.lib TM.gv -vmacro mymacro.v -run macro.pl” is run, the program stops at shell, “GOF >”

Type ‘help’ to list all APIs. To enter GUI mode type ‘start_gui’.

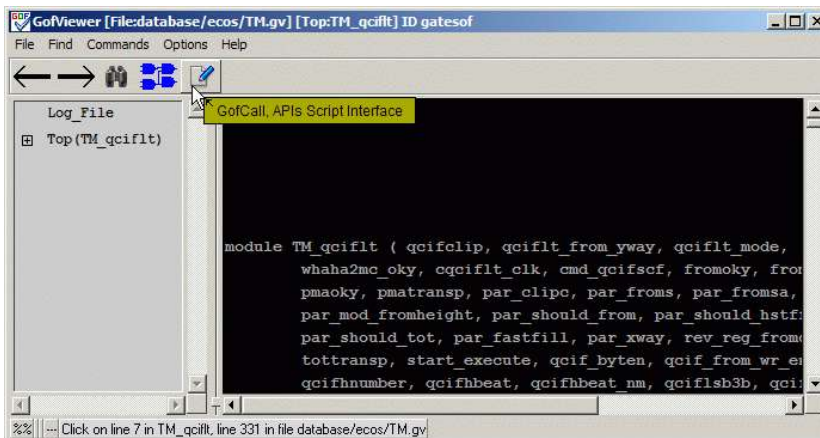


```
Command Prompt - bash
# change_pin("eco2135_DW01_u/A[10]", "qcifnumber_reg_10/Q");
# change_pin("qcifnumber_reg_11/D", "MX2X4", "eco2135_mux_11", ".A(-),.B(,),.S0(<");
# change_pin("eco2135_mux_11/S0", "qciflt_node");
# change_pin("eco2135_mux_11/B", "eco2135_DW01_u/SUM[11]");
# change_pin("eco2135_DW01_u/A[11]", "qcifnumber_reg_11/Q");
# change_pin("qcifnumber_reg_12/D", "MX2X4", "eco2135_mux_12", ".A(-),.B(,),.S0(<");
# change_pin("eco2135_mux_12/S0", "qciflt_node");
# change_pin("eco2135_mux_12/B", "eco2135_DW01_u/SUM[12]");
# change_pin("eco2135_DW01_u/A[12]", "qcifnumber_reg_12/Q");
# change_pin("qcifnumber_reg_13/D", "MX2X4", "eco2135_mux_13", ".A(-),.B(,),.S0(<");
# change_pin("eco2135_mux_13/S0", "qciflt_node");
# change_pin("eco2135_mux_13/B", "eco2135_DW01_u/SUM[13]");
# change_pin("eco2135_DW01_u/A[13]", "qcifnumber_reg_13/Q");
# change_pin("qcifnumber_reg_14/D", "MX2X4", "eco2135_mux_14", ".A(-),.B(,),.S0(<");
# change_pin("eco2135_mux_14/S0", "qciflt_node");
# change_pin("eco2135_mux_14/B", "eco2135_DW01_u/SUM[14]");
# change_pin("eco2135_DW01_u/A[14]", "qcifnumber_reg_14/Q");
# write_verilog("eco2135_net.v");
0
0
GofCall, Netlist Processing APIs, Interactive Mode
Run 'start_gui' to launch GUI window
Run 'help' to list API calls

GOF > start_gui
```

Figure 11

Press ‘GofCall, APIs Script Interface’ button to get GofCall window.

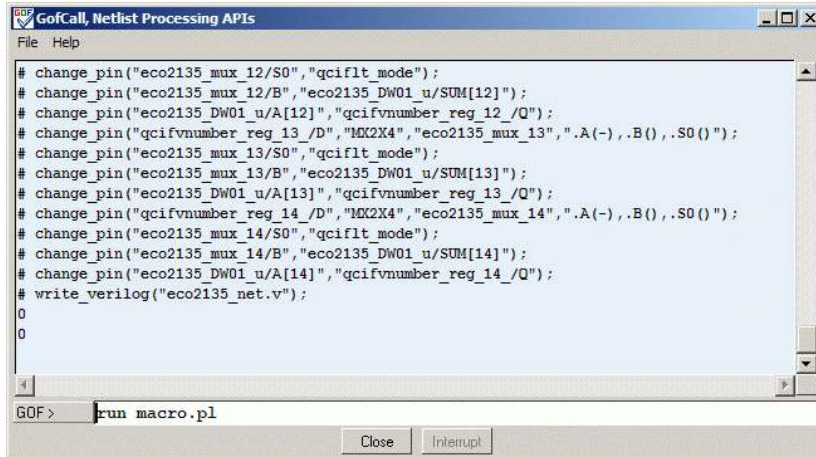


```
GofViewer [File:database/ecos/TM.gv] [Top:TM_qciflt] ID gatesof
File Find Commands Options Help
Log_File
Top(TM_qciflt)
GofCall, APIs Script Interface
module TM_qciflt ( qcifclip, qciflt_from_yway, qciflt_mode,
whaha2mc_oky, qciflt_clk, cmd_qcifscf, fromoky, from
pmaoky, pmatransp, par_clipc, par_froms, par_fromsa,
par_mod_fromheight, par_should_from, par_should_hstf;
par_should_tot, par_fastfill, par_xway, rev_reg_from
tottransp, start_execute, qcif_byten, qcif_from_wr es
qcifhnumber, qcifhbeat, qcifhbeat_nm, qciflsb3b, qci
```

Figure 12

Run script in GUI window

In 'GOF >' shell entry, type 'run macro.pl' and press enter. The script will be executed. It will stop if errors are found. Change the script to fix errors and rerun it again, until it runs to the end without errors. To debug the errors, you can bring up schematic from GofCall window.



The screenshot shows a window titled "GofCall, Netlist Processing APIs". The main area contains a script with the following content:

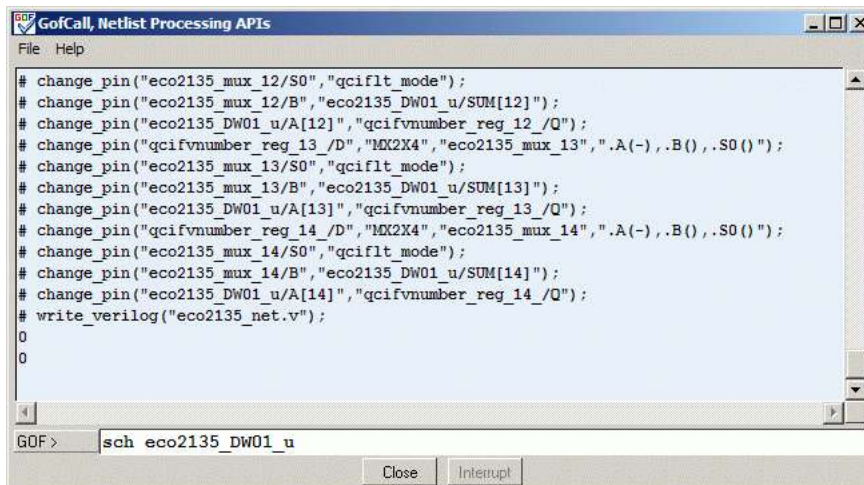
```
# change_pin("eco2135_mux_12/S0", "qcfilt_mode");  
# change_pin("eco2135_mux_12/B", "eco2135_DW01_u/SUM[12]");  
# change_pin("eco2135_DW01_u/A[12]", "qcfvnumber_reg_12/Q");  
# change_pin("qcfvnumber_reg_13/D", "MX2X4", "eco2135_mux_13", ".A(-), .B(), .S0()");  
# change_pin("eco2135_mux_13/S0", "qcfilt_mode");  
# change_pin("eco2135_mux_13/B", "eco2135_DW01_u/SUM[13]");  
# change_pin("eco2135_DW01_u/A[13]", "qcfvnumber_reg_13/Q");  
# change_pin("qcfvnumber_reg_14/D", "MX2X4", "eco2135_mux_14", ".A(-), .B(), .S0()");  
# change_pin("eco2135_mux_14/S0", "qcfilt_mode");  
# change_pin("eco2135_mux_14/B", "eco2135_DW01_u/SUM[14]");  
# change_pin("eco2135_DW01_u/A[14]", "qcfvnumber_reg_14/Q");  
# write_verilog("eco2135_net.v");  
0  
0
```

The command prompt at the bottom shows "GOF > run macro.pl". There are "Close" and "Interrupt" buttons at the bottom right of the window.

Figure 13

Interactive with schematic

You can type 'sch instance_name' or 'sch net_name' to bring up schematic to check if the ECO is done properly or any error occurs with an instance or a net. Please note, the instance should be in the current top module. Type 'set_top' in 'GOF >' to what is the current top module. By default, it's the top level module, if you never use 'set_top' in the script.



The screenshot shows the same "GofCall, Netlist Processing APIs" window. The script content is identical to Figure 13. The command prompt now shows "GOF > sch eco2135_DW01_u". There are "Close" and "Interrupt" buttons at the bottom right of the window.

Figure 14

Gates On the Fly Use Case: Add a module in netlist ECO

Figure 15 is the schematic with the instance 'eco2135_DW01_u' loaded. Use mouse-middle-button to expand the schematic. After a few clicks, you will see if the ECO is done correctly.

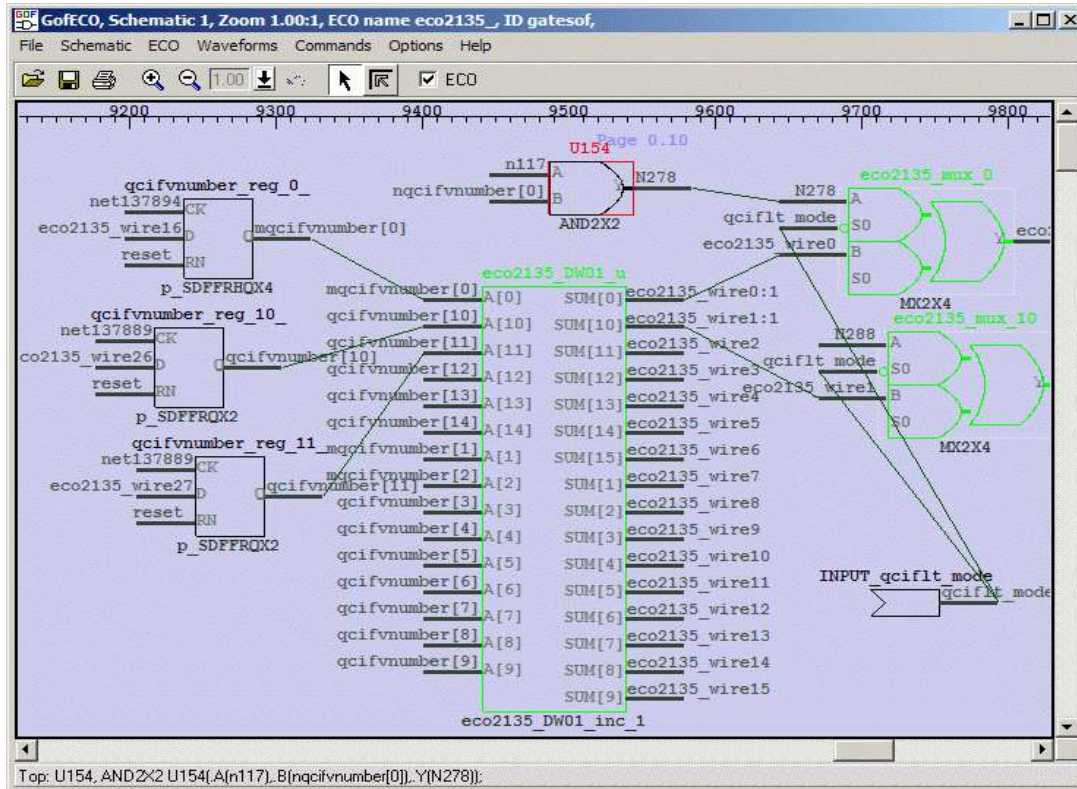


Figure 15

It's good practice to load the ECO netlist into GOF again. And use schematic to check if the connections are good.

```
gof -lib art.lib eco2135_net.v
```

Remember to run logic equivalence check to verify the final ECO netlist.

Conclusion

Combining GUI and script modes, Gates On the Fly provides a more flexible ECO flow. It cuts ECO iteration time significantly.