Computer-Aided VLSI System Design

Fall 2011, National Taiwan University

EC Lab: Design Verification

===

Motivation and Importance of Equivalence Checking

As a golden model is constructed, we prefer to check the equivalence between golden model and the revised model, which may be a modification of golden model and more probable to be a different abstraction level design from design flow synthesis, instead of running simulation on it. This prevent from contiguously simulation on these designs throughout the design flow, which may cost extremely long time and delay the design closure. Therefore, we may expect to perform equivalence checking between design versions throughout the design flow to enhance the efficiency of consistency check instead of simulation on every revised version.

Main Objectives of this Lab:

To understand how to use **Conformal LEC** to *formally* check the equivalence between your VLSI designs in the design flow. This Lab includes :

1. Check the equivalence between two VLSI designs in different abstraction levels.

2. Find the reason of non-equivalence and fix the problem from LEC supplements.

Lab 1: RTL Verilog v.s. Synthesized Gate-Level Verilog Designs

Objectives:

1. Understanding LEC basic interface (modes) and command format.
2. Setup environment for simple equivalence checking task.
3. Run LEC in shell mode / script mode.

Related Files:

In this Lab, we intend to perform equivalence checking between ***Verilog designs from your HW***, which are written and generated by yourselves before. These mandatory files are listed here:

File Name	Description
CTE.v	Your Verilog Code in HW2

CTE_syn.v	Your Synthesized Gate-Level Verilog in HW5
tsmc13.v	TSMC Cell Library Verilog File

In case you have little confidence to your HW designs, you can do this Lab with the designs in ~cvsd/CUR/Verify/LEC_Lab.tar.gz. But we strongly recommend you to use your own designs so that you can see whether your design remains functionally correct after synthesis.

Note: If you don't want to operate on GUI mode, you can type commands after LEC prompt instead. For commands, you can refer to *Command* within the following descriptions.

Step 0. Generate a directory, named "*LEC_Lab1*" for all three files to put if it is not ready.

Linux Shell Command : *mkdir LEC_Lab1*

cp <file> LEC_Lab1

Set current directory to *LEC_Lab1*.

Source LEC licence file : *csh; source /usr/cadence/CIC/confrml.csh*

Step 1. Start Cadence Conformal LEC from GUI mode.

Linux Shell Command : *lec&*

Don't forget to start X-window before GUI mode.

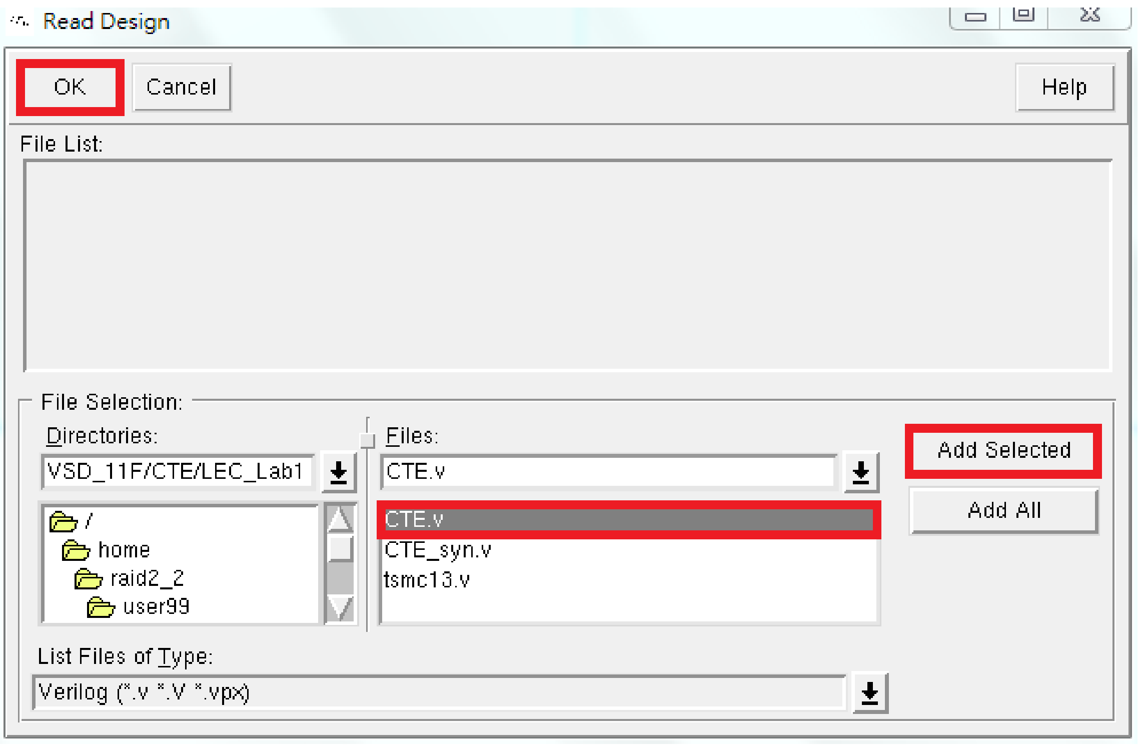
Command-Line LEC: lec –nogui

Step 2. Setup Log File. (Optional)

LEC Command : *set log file LEC_Lab1.log*

Step 3. Read RTL Verilog design as Golden Model.

From the GUI window, click on the icon . Then "Read Design" dialog window will pop out as shown below:



Double click your ***CTE.v*** as golden design, press OK and leave.

Note: You can see some warning and information messages from LEC standard output, and make sure your design has been read successfully.

Command: read design CTE.v –golden

Step 4. Read Gate-Level Verilog design as Revised Model.

From the GUI window, click on the icon again. Then "Read Design" dialog window will pop out as shown above.

Now, select your ***CTE_syn.v*** as revised design, press OK and leave. Revised

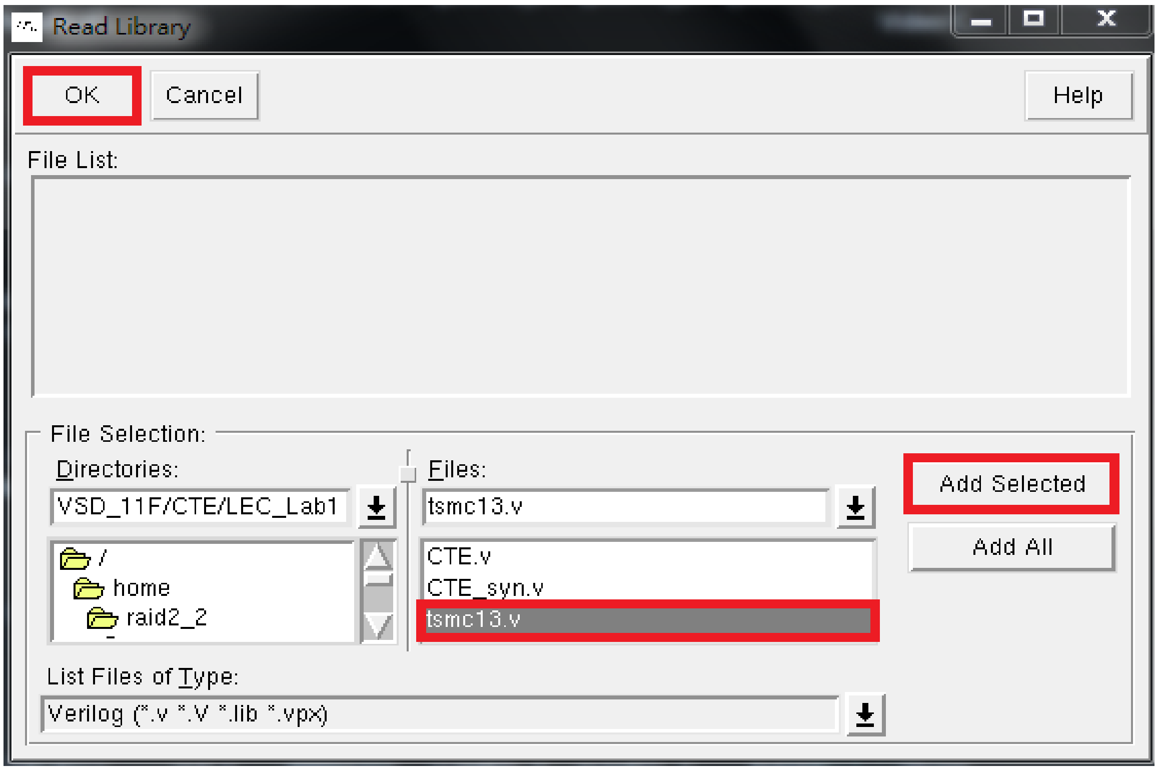
However, you may found some error message outputs from LEC RTL checker. What's the problem?!

Have you ever specified cell libraries instantiated in your gate-level design to LEC?!

Command: read des CTE_syn.v –revised

Step 5. Read Verilog Library File for Gate-Level Revised Model.

From the GUI window, click on the icon . Then "Read Library" dialog window will pop out as shown below:



Double click ***tsmc13.v*** as the library file, which must correspond to the library used in synthesis before. Note that you can either read the library for Revised Revised Model only or for Both Both Models in this Lab. (Why?!)

Command: read library tsmc13.v –verilog –both

Step 6. Read Gate-Level Verilog design as Revised Model again.

As the library file has been read in, we may expect to successfully read in the synthesized

Verilog now. Please perform *Step 4* again and your design should be read successfully now.

Step 7. Modes in LEC:

LEC has two modes : ***Setup Mode*** and ***LEC Mode*** for enabling different operations:

1. *Setup* ***Setup Mode :***

   Mainly for design / constraints / rename / black box setup before compare.

2. *LEC* ***LEC Mode :***

   Compare mode for adding compare points and report compare results.

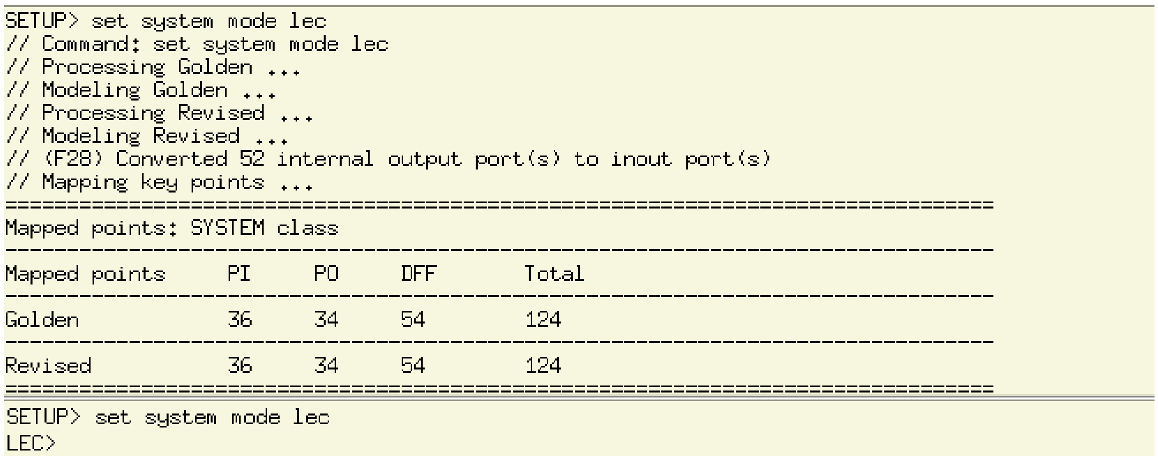
Some commands are available in both modes; you can type *help* to see all commands.

As all setup constraints has been applied, for this is a basic and simple Lab. Now we're going to change current mode from ***Setup*** to ***LEC*** for further compare:

Click on the icon *LEC* at the right-top of the GUI window.

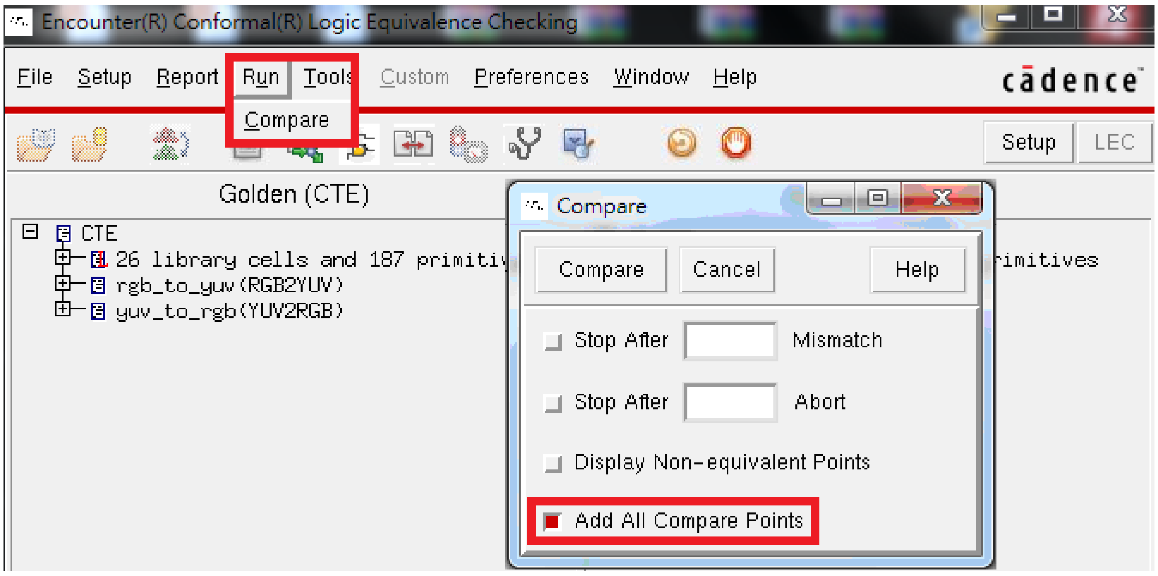
Command: set system mode lec

You can see LEC start automatically find compare points and makeup constraints from setup mode specifications, and finally output the statistical information of mapped points to the output window, as shown below.



```
SETUP> set system mode lec
// Command: set system mode lec
// Processing Golden ...
// Modeling Golden ...
// Processing Revised ...
// Modeling Revised ...
// (F28) Converted 52 internal output port(s) to inout port(s)
// Mapping key points ...
================================================================================
Mapped points: SYSTEM class
--------------------------------------------------------------------------------
Mapped points     PI      PO      DFF        Total
--------------------------------------------------------------------------------
Golden            36      34      54         124
--------------------------------------------------------------------------------
Revised           36      34      54         124
================================================================================
SETUP> set system mode lec
LEC>
```

Step 8. Add Compare Points:


Encounter(R) Conformal(R) Logic Equivalence Checking
File Setup Report Run Tools Custom Preferences Window Help
cādence
Compare
Setup LEC
Golden (CTE)
CTE
26 library cells and 187 primiti
rgb_to_yuv(RGB2YUV)
yuv_to_rgb(YUV2RGB)
imitives
Compare
Compare Cancel Help
Stop After Mismatch
Stop After Abort
Display Non-equivalent Points
Add All Compare Points

We should tell LEC what are the points in both designs that we want to compare. In general, we may specify all PO and DFF to be compared. However, you can manually add or delete compare points for LEC on your purpose. In this Lab, we simply add all PO and DFF as compare points.

Click Run → Compare at the top function button list, and a compare window will pop out as shown above.

Command: add compare points –all

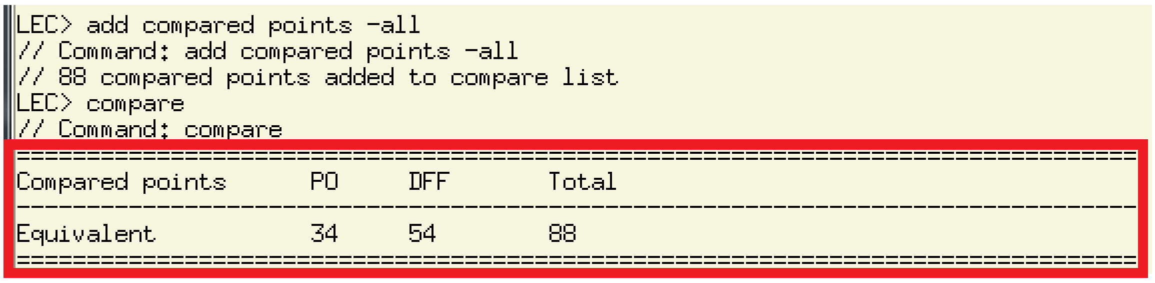
Step 9. Compare:

For we've set up all indications to LEC currently, we are now starting to perform EC.

Press the compare icon in the GUI window to start compare.

Command: compare

After a few minutes, you may found LEC compare results shown directly on the screen :



```
LEC> add compared points -all
// Command: add compared points -all
// 88 compared points added to compare list
LEC> compare
// Command: compare
================================================================================
Compared points       PO      DFF        Total
--------------------------------------------------------------------------------
Equivalent            34      54         88
================================================================================
```

Did you see any non-equivalent point?!   What's the message about?!

Step 10. Exit LEC.

As your result is equivalent, you've completely prove that your synthesis is sound and the functionalities of synthesized design is the same as original Verilog.

Now, you can click on File → Exit → Yes to exit LEC forcedly.

Command: exit –f

What have you learnt from this Lab?!
What are the improvements of applying equivalence checking compared to repeated simulation that you've found in this Lab?!

In case you've set the log file in *Step 2*, you can see all messages on your screen output in that file.

For script generation, you can click File → Save Dofile, and all commands related to what you've done will be stored in that dofile.
Command: save dofile LEC_Lab1.script
Next time you can simply run that script by File → Do Dofile.
Command: dofile LEC_Lab1.script
If you want to run script only, you can also pass dofile as parameter for LEC in *Step 1*.
Linux Shell Command : *lec -dofile LEC_Lab1.script&* (GUI Mode)
lec –nogui -dofile LEC_Lab1.script (SHELL Mode)

Lab 2: Synthesized Gate-Level Verilog Design v.s. Scan-chain Inserted DFT Design

Objectives:

1. Understanding how to perform EC between circuits before and after scan-chain insertion.
2. Perform simple diagnosis functionalities supported by LEC to fix the non-equivalence.

Related Files:

Extend from previous Lab, now we're going to perform equivalence checking between ***Synthesized gate-level Verilog design and DFT design***, which are generated by yourselves before. As we've checked that your synthesized design is equivalent to your Verilog design, we expect to continue on checking the equivalence of scan-chain inserted design to one of the previous equivalent designs. (Here, we take gate-level one for better analysis later.) All mandatory files are listed here:

File Name	Description
CTE.v	Your Verilog Code in HW2
CTE_syn.v	Your Synthesized Gate-Level Verilog in HW5
CTE_dft.v	Your Synthesized Verilog after DFT (scan-chain insertion) in HW5
tsmc13.v	TSMC Cell Library Verilog File

The setup and compare of the two designs in this Lab is similar to that in the previous Lab;

hence, the following procedure will not be explained in detail. You can refer to steps in the previous Lab and concerned with differences only. However, we also expect you to operate on your designs.

Step 0. Generate a directory, named "*LEC_Lab2*" for all three files to put if it is not ready.

Linux Shell Command : *mkdir LEC_Lab2*

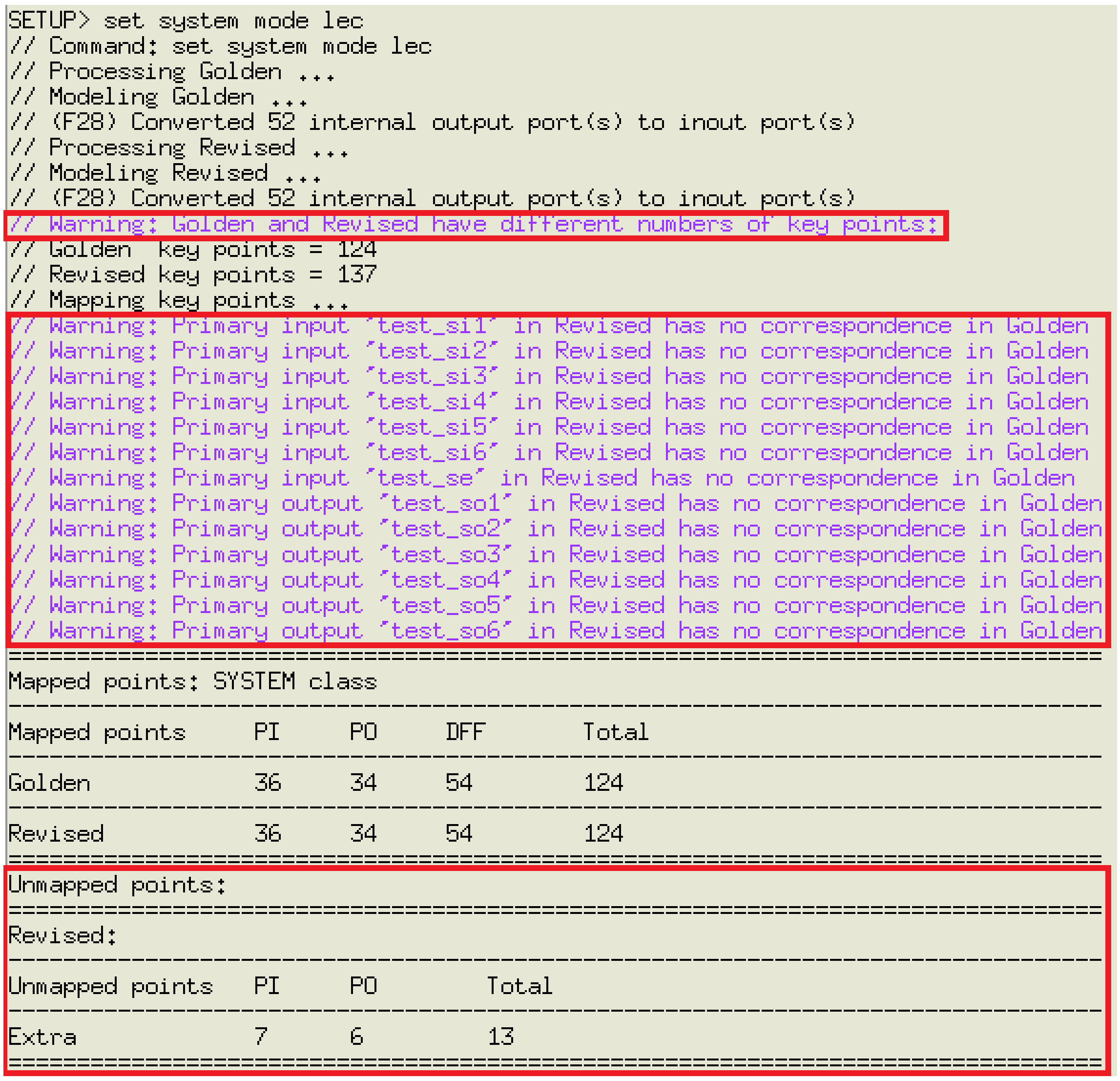
cp <file> LEC_Lab2

Set current directory to *LEC_Lab2*

Source LEC licence file : csh; source /usr/cadence/CIC/confrml.csh

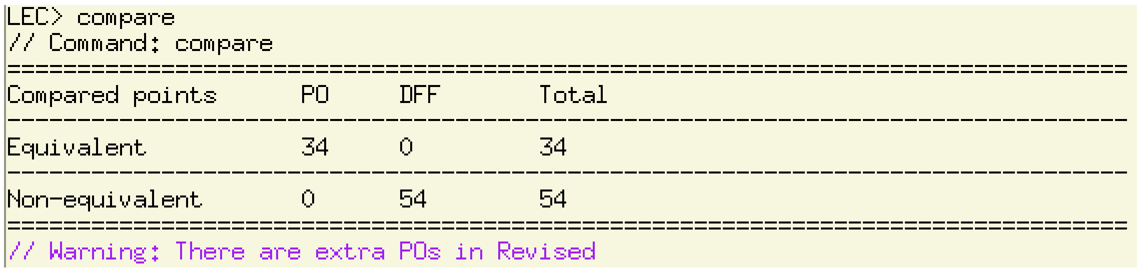
Step 1. Start LEC, read golden design as ***CTE_syn.v*** and revised design as ***CTE_dft.v***. Note that all of them rely on the Library file ***tsmc13.v***, which must be read in prior to the two designs. Is there any error message?! *Hint: Both of them need the library file in this Lab.*

Step 2. As all designs and library have been successfully read in, change to LEC Mode for design compare. Similarly, you should add compare points and perform compare then. However, you may see some error messages and un-mapped points after change to LEC Mode, as shown below. Please ignore them at this moment.



```
SETUP> set system mode lec
// Command: set system mode lec
// Processing Golden ...
// Modeling Golden ...
// (F28) Converted 52 internal output port(s) to inout port(s)
// Processing Revised ...
// Modeling Revised ...
// (F28) Converted 52 internal output port(s) to inout port(s)
// Warning: Golden and Revised have different numbers of key points:
// Golden  key points = 124
// Revised key points = 137
// Mapping key points ...
// Warning: Primary input 'test_si1' in Revised has no correspondence in Golden
// Warning: Primary input 'test_si2' in Revised has no correspondence in Golden
// Warning: Primary input 'test_si3' in Revised has no correspondence in Golden
// Warning: Primary input 'test_si4' in Revised has no correspondence in Golden
// Warning: Primary input 'test_si5' in Revised has no correspondence in Golden
// Warning: Primary input 'test_si6' in Revised has no correspondence in Golden
// Warning: Primary input 'test_se' in Revised has no correspondence in Golden
// Warning: Primary output 'test_so1' in Revised has no correspondence in Golden
// Warning: Primary output 'test_so2' in Revised has no correspondence in Golden
// Warning: Primary output 'test_so3' in Revised has no correspondence in Golden
// Warning: Primary output 'test_so4' in Revised has no correspondence in Golden
// Warning: Primary output 'test_so5' in Revised has no correspondence in Golden
// Warning: Primary output 'test_so6' in Revised has no correspondence in Golden
================================================================================
Mapped points: SYSTEM class
--------------------------------------------------------------------------------
Mapped points      PI      PO      DFF        Total
--------------------------------------------------------------------------------
Golden             36      34      54         124
--------------------------------------------------------------------------------
Revised            36      34      54         124
================================================================================
Unmapped points:
================================================================================
Revised:
--------------------------------------------------------------------------------
Unmapped points    PI      PO         Total
--------------------------------------------------------------------------------
Extra              7       6          13
================================================================================
```

The comparison result contains non-equivalent points, as shown below. Hence, two designs are found non-equivalent!



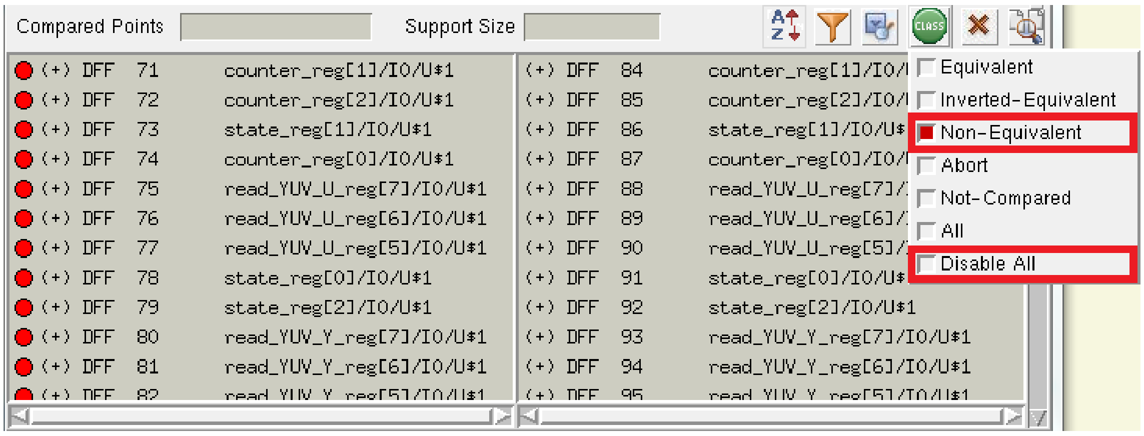
```
LEC> compare
// Command: compare
================================================================================
Compared points      PO      DFF        Total
--------------------------------------------------------------------------------
Equivalent           34      0          34
--------------------------------------------------------------------------------
Non-equivalent       0       54         54
================================================================================
// Warning: There are extra POs in Revised
```

We can see all PO are equivalent, while all DFF are non-equivalent! Note that even if all PO are equivalent cannot imply that two designs are equivalent! (For LEC view DFF as cut points, and compare PO within a smaller cut cone with inputs to be PI or DFF as boundary. So it's easy to prove the equivalence of some (and maybe all) PO pairs while the two designs are strictly different.)

Can you figure out some problems here?! <u>*Hint:*</u> *From warning messages.*

Step 3. Try Fixing the Non-equivalence.

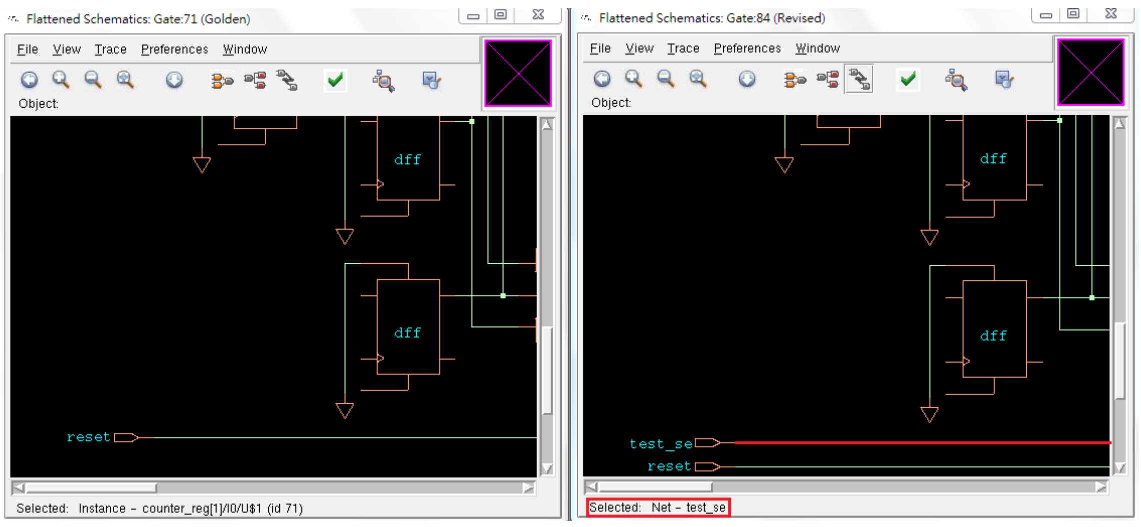
Firstly, we resort to some exist debugging functionalities supported by LEC to have a first perspective of the problem. Click button for opening mapping manager, then a mapping manager window will pop out, as shown below:



```
Compared Points                    Support Size

● (+) DFF  71   counter_reg[1]/I0/U$1          (+) DFF  84   counter_reg[1]/I0/
● (+) DFF  72   counter_reg[2]/I0/U$1          (+) DFF  85   counter_reg[2]/I0/
● (+) DFF  73   state_reg[1]/I0/U$1            (+) DFF  86   state_reg[1]/I0/U$
● (+) DFF  74   counter_reg[0]/I0/U$1          (+) DFF  87   counter_reg[0]/I0/
● (+) DFF  75   read_YUV_U_reg[7]/I0/U$1       (+) DFF  88   read_YUV_U_reg[7]/
● (+) DFF  76   read_YUV_U_reg[6]/I0/U$1       (+) DFF  89   read_YUV_U_reg[6]/
● (+) DFF  77   read_YUV_U_reg[5]/I0/U$1       (+) DFF  90   read_YUV_U_reg[5]/
● (+) DFF  78   state_reg[0]/I0/U$1            (+) DFF  91   state_reg[0]/I0/U$
● (+) DFF  79   state_reg[2]/I0/U$1            (+) DFF  92   state_reg[2]/I0/U$1
● (+) DFF  80   read_YUV_Y_reg[7]/I0/U$1       (+) DFF  93   read_YUV_Y_reg[7]/I0/U$1
● (+) DFF  81   read_YUV_Y_reg[6]/I0/U$1       (+) DFF  94   read_YUV_Y_reg[6]/I0/U$1
● (+) DFF  82   read_YUV_Y_reg[5]/I0/U$1       (+) DFF  95   read_YUV_Y_reg[5]/I0/U$1
```

At the bottom table, please firstly disable all compare points listed here and next assign non-equivalent points for debug only. For all DFF are non-equivalent, we may infer their reasons should be similar. Please select one of the DFF, for instance, counter_reg[1] for example, and click on *right* mouse button to select ***schematic*** view for debugging. Both golden and revised schematics will pop out as follows:


Flattened Schematics: Gate:71 (Golden)
File View Trace Preferences Window
Object:
dff
dff
reset
Flattened Schematics: Gate:84 (Revised)
File View Trace Preferences Window
Object:
dff
dff
test_se
reset

Selected: Instance – counter_reg[1]/I0/U$1 (id 71)

Selected: Net – test_se

From schematic views of corresponding DFF, we can found that in the revised schematic, an extra input *test_se* is found at the input cone of that DFF. Recall that this input is added for test mode toggling in DFT design, with fixed value 1 for scan mode instead of function mode of the design. What happened once LEC set this input to be 1?! Will the two DFF report the same result?! Will they be proved equivalent?! Why?!

Are there more signals that may also result to the same problem?!

Step 4. Refine Settings:

The biggest difference between the two designs is the number of design I/O: for DFT circuit, we may add scan-inputs as PI and scan-outputs as PO at the design boundary. However, LEC should not know these. Therefore, we must inform LEC by commands as settings.

1. **Scan outputs:** For we've added all PO and DFF as mapping points previously, and we're now desire to remove this output in DFT design (revised design). Therefore, LEC will not report this PO as un-mapped point.

   Now, type command: *add ignored outputs test_so* –revised* at the command input after prompt to ignore all scan outputs in both the mapping list and the compare point list. However, you may see the error message of "*This command is not allowed in current system mode*". That's because the command *add ignored output* can only be specified in setup stage of LEC, but we're now intend to add this constraint after compare error. Therefore, you should firstly change mode back to **Setup** by clicking the icon, and then type this command to ignore scan outputs from mapping.

   Command: set system mode setup

2. **Scan inputs**: There should be multiple scan in and scan enable inputs in DFT design for scan-chain functionalities in test mode; however, in golden design, it does not support scan-chain enabled mode. But we have no idea to remove those pins from the model, and they will not be mapping points. However, they affect the compare results.

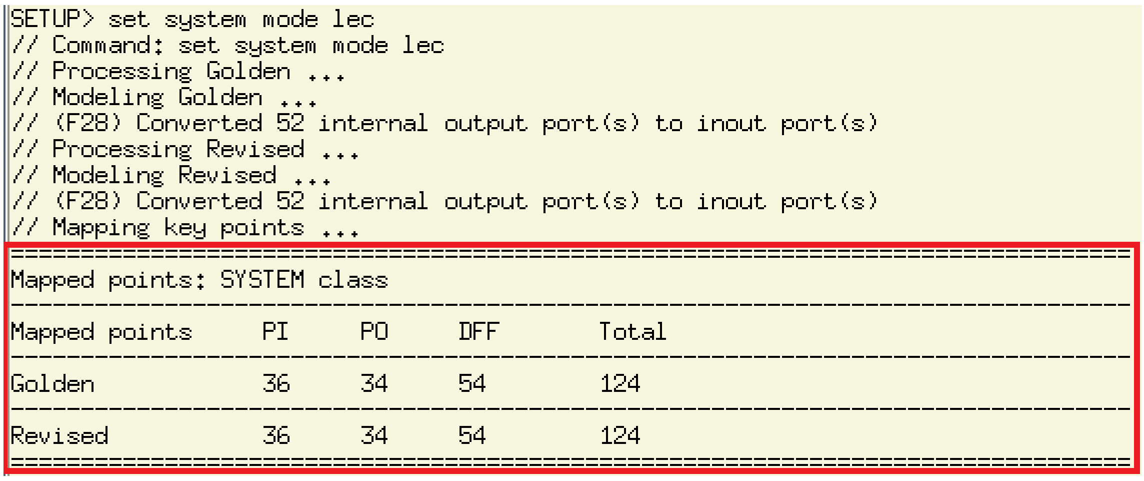
   Note that when scan enable signal is tied to zero, the design after DFT works the

same as original circuit, i.e. not in scan mode. Based on this concept, we can add the command to set fixed values for those scan input pins.

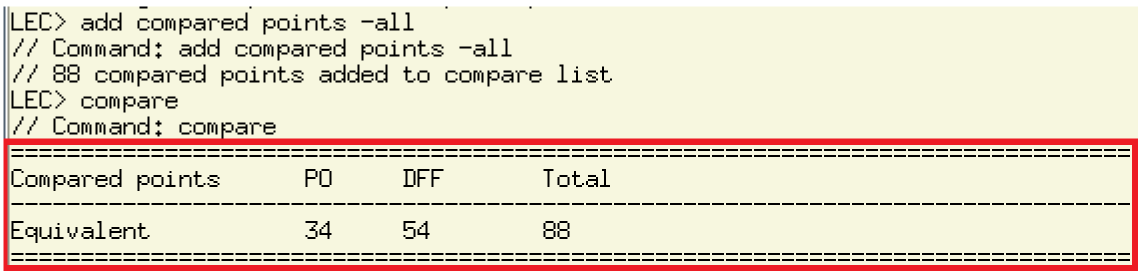
The command is: *add pin constraints 0 test_si* test_se –revised*

Step 5. Continue on comparison:

Supposedly, commands for setup refinements are properly added to LEC, we can continue on compare now. Change system mode back to LEC then add all compare points again followed by compare. Note that you can see all pins are mapped successfully. Now, the two designs will be checked equivalent. A sample output result is shown below:



```
SETUP> set system mode lec
// Command: set system mode lec
// Processing Golden ...
// Modeling Golden ...
// (F28) Converted 52 internal output port(s) to inout port(s)
// Processing Revised ...
// Modeling Revised ...
// (F28) Converted 52 internal output port(s) to inout port(s)
// Mapping key points ...
================================================================================
Mapped points: SYSTEM class
--------------------------------------------------------------------------------
Mapped points      PI     PO     DFF       Total
--------------------------------------------------------------------------------
Golden             36     34     54        124
--------------------------------------------------------------------------------
Revised            36     34     54        124
================================================================================
```



```
LEC> add compared points -all
// Command: add compared points -all
// 88 compared points added to compare list
LEC> compare
// Command: compare
================================================================================
Compared points       PO     DFF       Total
--------------------------------------------------------------------------------
Equivalent            34     54        88
================================================================================
```

Step 6. Exit LEC.

As your result is equivalent, you've completely prove that the functionality of scan-chain inserted DFT design is equivalent to that in synthesized gate-level design, and also to the original Verilog written by yourself.

Now, you can click on File → Exit → Yes to exit LEC forcedly.

Command: exit –f

What have you learnt from this Lab?!

1st Version: 2010.11.29 by Cheng-Yin Wu, NTUGIEE and Yu-Fu Yeh, NTUGIEE

2nd Version: 2011.12.14 by Cheng-Yin Wu, NTUGIEE and Wei-Hsun Lin, NTUGIEE