Overview of Formality Basic Labs

Purpose: These labs are designed for you to become familiar with using Formality.

Content: There are four labs using Synopsys training and public-domain RTL source. The netlists were generated using DesignCompiler 2006.06-SP5 software. Be aware that there are additional SVF enhancements created by later versions of DC. (Using newer SVF files with the Auto Setup Mode in Formality will reduce or eliminate the issues in these labs.)

Procedure:

- Follow the instructions for each lab.
- There is a ".solution" sub-directory with the correct result. Please compare your result with the correct result as you finish each lab.

Invoke Formality in this manner to bring up the GUI:

"fm_shell -gui -f runme.fms |tee runme.log" or

"formality -f runme.fms |tee runme.log"

FM Lab1: Basic Formality Flow

Objective: This lab will introduce you to the Formality GUI by manually reading-in two designs for verification. Then, you will create a Tcl script to perform the same verification.

Lab flow:
All of the necessary reference and implemention files, and libraries are included in sub-directories.

Use the Formality GUI to verify the golden Verilog RTL against the gate-level Verilog netlist produced by DC. Be sure to include the SVF file. Afterward, modify the resulting "fm_shell_command.log" file to become a Formality TCL script.

a.) Bring up the Formality GUI by using the command "formality" or "fm_shell –gui". Follow the flow buttons on the GUI to read-in SVF, reference designs, implementation design with library, and then verify.

b.) The DC produced SVF file "default.svf" is located under the sub-directory "netlist_w_svf". This file is necessary for correct setup.

c.) The Verilog RTL files are located under the directory "rtl". The top-level design name is "mR4000".

d.) The Verilog gate-level netlist "mR4000.gates.v" is located under the directory "netlist_w_svf". The top-level design name is "mR4000".

e.) The library file "tc6a_cbacore.db" is located under "lib". This is needed for the netlist. There are no Verilog simulation libraries, just the .db file for this lab.

f.) Use the GUI flow buttons: Guidance, Reference, Implementation, Setup (not needed), Match, Verify, Debug (not needed).

g.) After completing a successful verification using the GUI, edit and transform the "fm_shell_command.log" file into a Formality "runme.fms" Tcl script.

h.) Use the script to run verification: "fm_shell -f runme.fms |tee runme.log".

i.) Experiment with Formality commands and variables. Try some of the following commands:

- *help report**
- *report_passing_points*
- *man set_top*
- *man verify*
- *report_status*

j.) Exit Formality.

FM Lab2: Recognizing Simulation/Synthesis Mismatch Errors

Objective: This lab will give you practice in writing a Formality Tcl script. This design is VHDL and contains several potential differences between simulation and synthesis in their code interpretation which Formality will flag. You will need to direct Formality to ignore the differences and continue verification.

By default, Formality is conservative in its interpretation of RTL. It will stop processing if it finds a difference between simulation and synthesis. You can direct it to continue by turning these error messages into warning messages. Use the following variable to accomplish this:

```
set hdlin_warn_on_mismatch_message "FMR_VHDL-1002 …"
```

Use this variable before reading in the RTL into a container.

(Note: If you have these types of mismatch issues using your design at work, you need to make sure that these conditions are investigated before taping-out your design.)

This lab requires the use of "hdlin_dwroot". Please edit the runme.fms under the …fm_basic/labs/lab2/scripts directory to correctly set the variable "hdlin_dwroot" to the top-level location of the DC software.

Lab flow:

1.) Use the script ./scripts/runme.fms as a starting point to create your Formality Tcl script. Copy it to the lab2 working directory.

2.) You will need to complete the following:

   a. Read and link the reference VHDL files from the directory src.

   b. Read and link the Verilog gate-netlist.

3.) Run an initial verification. Look at the error message ID.

4.) Include the error message ID in the Formality script using the variable hdlin_warn_on_mismatch_message to turn it into a warning message instead.

5.) Continue this until you get a successful verification.

6.) There are two other ways to accomplish this conversion of sim/synth messages to warnings. See the ./.solution/runme.fms script.

FM Lab3: Missing Part of the Design

Objective: This lab shows an example of what happens in verification if a piece of the reference design is missing while the implementation design remains complete. The point to this lab is to review transcript messages. These messages can provide hints to correct problems. You must change the "runme.fms" FM TCL script to get a successful verification.

Lab flow:

1.) Run the verification using the existing "runme.fms" script.

2.) Find clues to indicate the potential problem.

2a) Transcript messages:

Formality debugging involves collecting information that may point to the reason why the design fails verification. Always look at the transcript messages first.

Note the following warning message in the transcript:

```
Warning: Cannot link cell '/WORK/fifo/memory' to its reference design
'DW_ram_r_w_s_lat'. (FE-LINK-2)
Status:  Implementing inferred operators...
Status:  Creating black-box designs...
Created technology library 'FM_BBOX' in container 'r' for black-box designs
Created black-box design 'DW_ram_r_w_s_lat' in library 'FM_BBOX'
Warning: 1 blackbox designs were created for missing references. (FM-064)
```

Formality is creating a black-box to represent a missing piece of the design. The missing piece is the DW part "DW_ram_r_w_s_lat".

This is only a warning instead of an error because the customer included this variable setting in the FM TCL script:

```
set hdlin_unresolved_modules black_box
```

Otherwise, Formality would have stopped processing the design.

The problem is resolved by setting a variable to point to the top-level of the DesignCompiler tree. Formality can then find the correct DW information to generate the missing part.

The Formality variable is hdlin_____.

The DC location (for Synopsys internal training) is */global/apps/syn_2007.12-SP2*.

*** It is important to note that if you are paying attention to the transcript, you would stop here and correct the problem with the missing piece of the design. However, since the script continued to

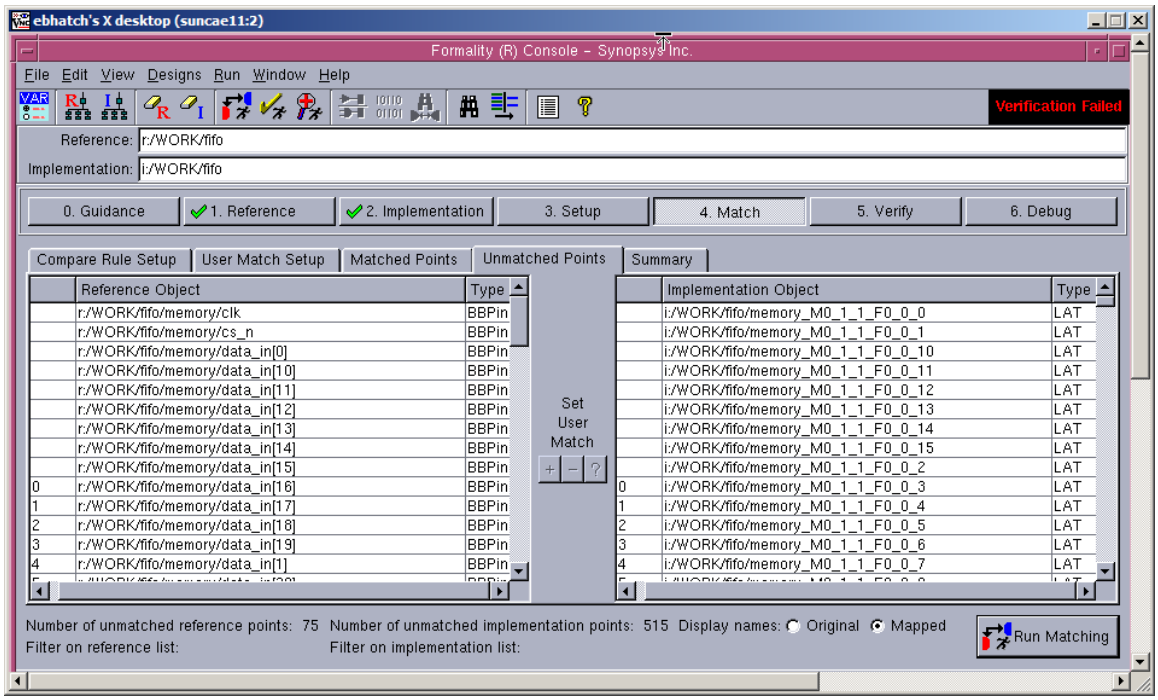
finish the verification, there are some additional clues to see if you skipped the warning message.

2b) Messages from matching:

If you proceeded through matching, you will see several unmatched, black-box input pins. Since there are no unmatched implementation black-boxes, this means that the reference design has a black-box that the implementation does not have.

Additionally, the implementation seems to have latches that the reference design does not have. You can draw the conclusion that a big design piece is missing in the reference design.

Here is a picture of the GUI unmatched points tab:

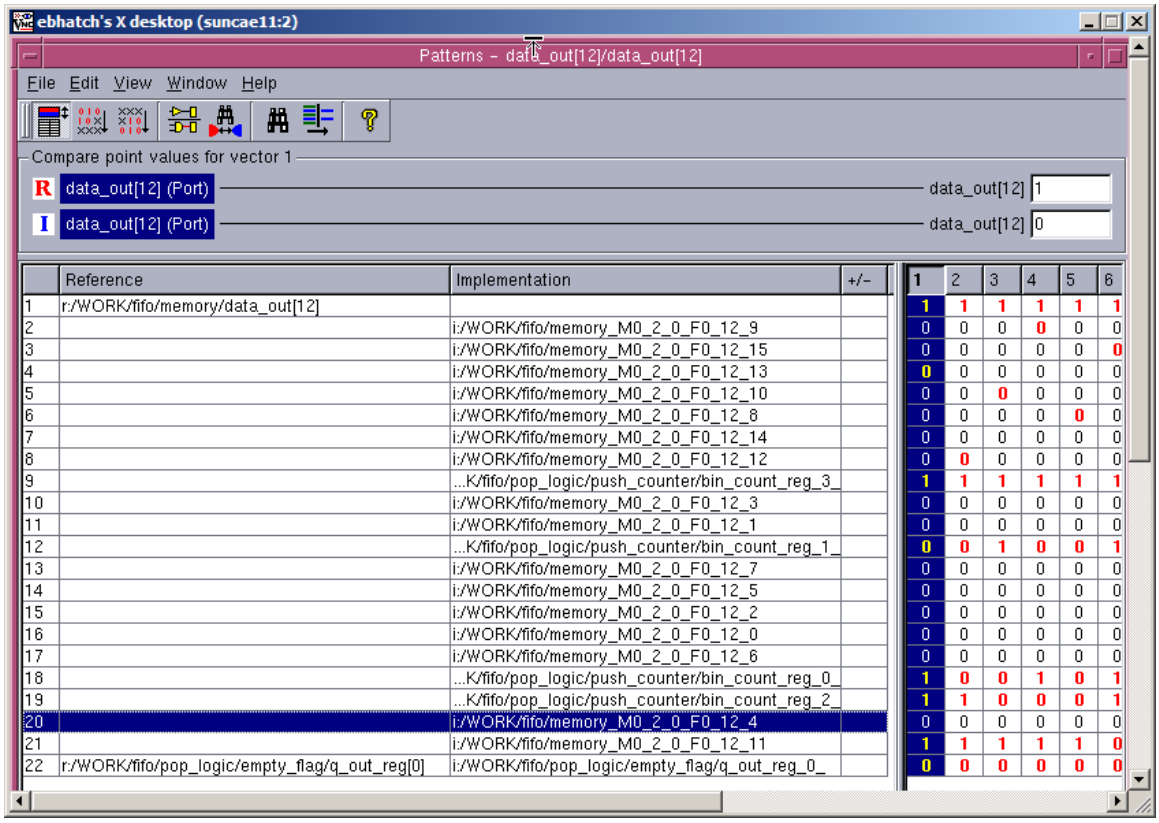


2c) Graphical debugging:

After running the verification and getting failing compare points, we normally would run "Diagnose". However, we already know that a large piece of the design is missing.

It would be interesting to quickly view the pattern window to see how the missing design piece affected the logic cone of a failing compare point.

Notice that several logic cone inputs seem to be missing in the pattern window:



3.) Include the missing variable for specifying the DC tree in the FM TCL script and re-run verification. If you do not get a successful verification, view the .solution directory.

FM Lab4: Beginning Debug

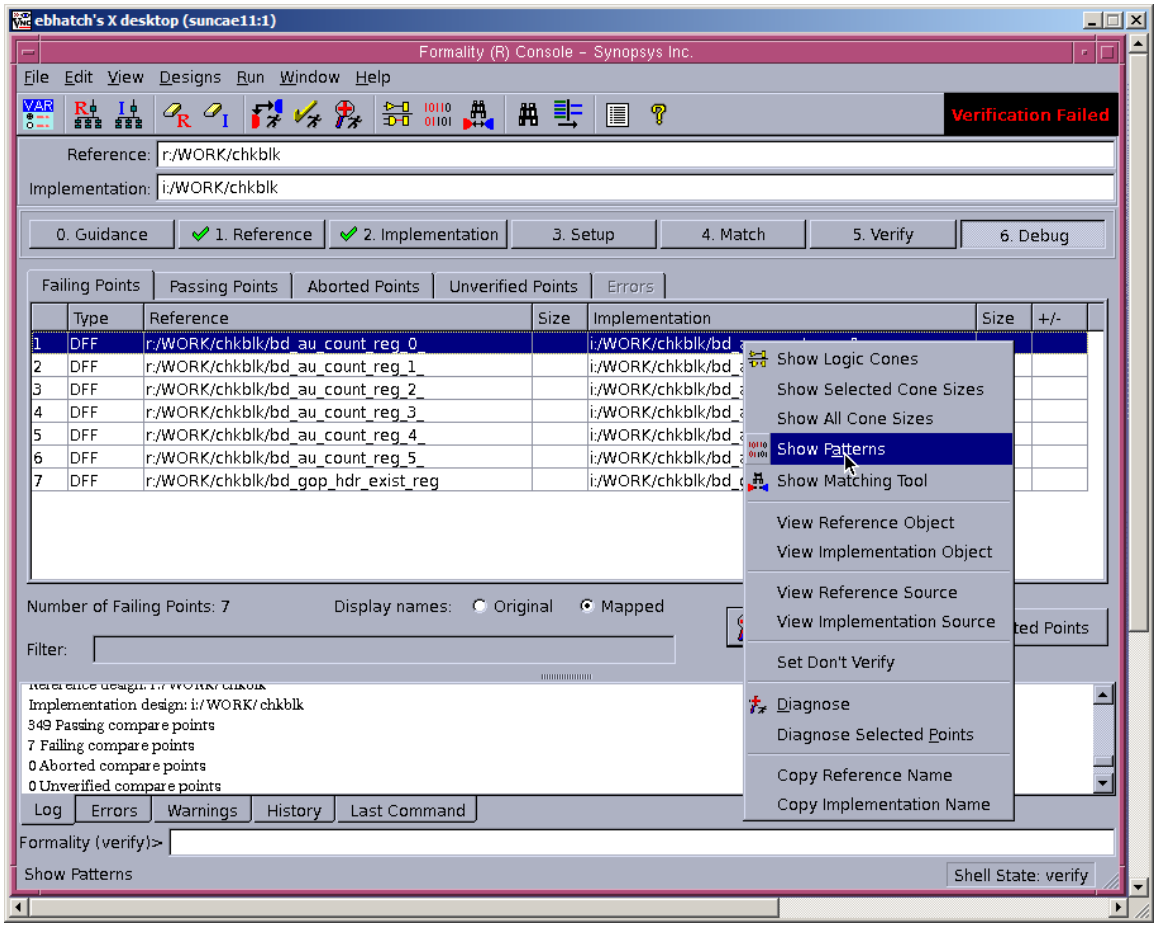
Objective: This lab will give you practice in using the Diagnose command and in viewing failing logic cones. This is a gate-netlist vs gate-netlist ECO verification. The ECO was done manually, and you are checking the resulting netlist against the original netlist.

Lab flow:

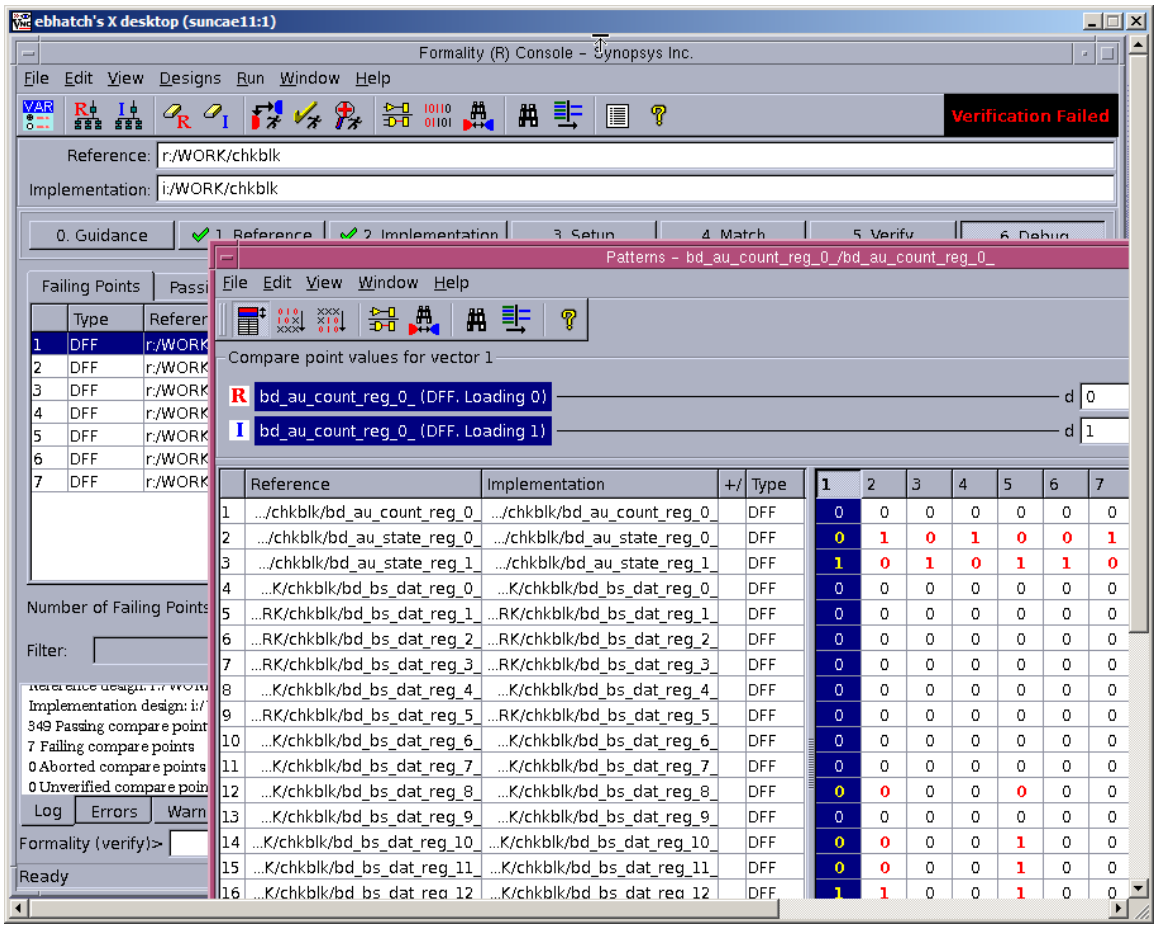
a. Run Formality script fm.tcl.
   - formality -f fm.tcl |tee fm.log

b. Start the GUI.

c. Right mouse click on the first failing compare point and select Show Patterns.



d. This brings up the Pattern Viewer. The Pattern Viewer is verify useful in quickly identifying obvious differences in inputs to logic cones. We recommend using the Pattern Viewer first when doing graphical debugging.



e. In this case there is nothing obvious that is different between the ref/impl logic cone inputs other than the same pattern produces different results at the compare point. Therefore, there is some difference in the combinational logic in the two cones.

f. Close the Pattern Viewer.

g. Click Diagnose button. This runs the diagnosis algorithm on all failing compare points. You should get 1 error detected and 2 error candidates.

h. The GUI automatically changes to Error Candidate window. Right click on failing cell U4072.

i. Select Show Logic Cones.

j. Select first failing compare point and click OK. The schematic will highlight the error candidate (implementation) and the matching region (reference).

k. Notice that U4072 is a 3-input AND in the reference and a 3-input OR in the implementation. This is the ECO problem. Also notice that the logic cone is large. The Error Candidates are highlighted in orange.

l. Click on the Error Candidate Pruning button (or F8). The implemenation will be pruned.

m. Click on the reference window and click again on the Error Candidate Pruning button (or F8). Now the reference is pruned. The resultant logic cones are easier to deal with. (No action needed in this lab to correct the design.)

n. Close the schematic window. Click on the failing points tab.

o. Select the first two failing points (bd_au_count_reg_0_ and bd_au_count_reg_1_) and click Diagnose Selected Points button.

p. Note that there is still 1 error detected but now there are 6 error candidates.

q. Right click on failing cell U4072.

r. Select Show Logic Cones. Note that the Select Failing Compare Point window shows the 7 failing points but only 2 have been diagnosed.

s. Select first failing compare point and click OK.

t. Prune (F8) the reference and implementation schematic windows. Note that there are more possible failures candidates. This is due to selecting a subset of the failing points.