Overview of Formality Jumpstart Labs

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

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 FM 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 gate-level Verilog netlist against the golden Verilog RTL. Be sure to include the SVF file. Afterward, modify the resulting "*fm_shell_command.log*" file to become a Formality TCL script.

1) Bring up the Formality GUI by using the command "*formality*" or "*fm_shell -gui*". Follow the flow buttons on the GUI to read in the SVF file, the reference designs, the implementation design with library. Then run verify.

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

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

4) The Verilog gate-level netlist "*mR4000.vg*" is located under the directory "*netlist_w_svf*". The top-level design name is "*mR4000*".

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

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

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

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

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

   - *help report**
   - *report_passing_points*
   - *printvar verification**
   - *report_status*
   - *man set_top*

10) Exit Formality

FM Lab2: Recognizing Simulation/Synthesis Mismatch Errors

Objective: This lab will give you practice in writing a Formality Tcl script. This design consists of several VHDL modules, and contains potential differences between simulation and synthesis in 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 verification by converting those error messages into warning messages. Use the following variable to accomplish this:

set hdlin_warn_on_mismatch_message "FMR_VHDL-1002 etc…"

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

(Note: If you see these types of mismatch messages using your own design, please ensure these conditions are investigated before taping-out your design.)

Lab flow:

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

2) You will need to complete the following:

   a. Read and link the reference VHDL files from the directory *rtl*.

   b. Read and link the DC netlist from the directory *netlist_w_svf*

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

```
Warning: Out of range write possible, may cause simulation and synthesis
mismatch. (File: /remote/users/lab2/rtl/predict.vhd Line: 194)  (FMR_ELAB-146)

Warning: Out of range write possible, may cause simulation and synthesis
mismatch. (File: /remote/users/lab2/rtl/predict.vhd Line: 195)  (FMR_ELAB-146)

Error: Unsuppressed RTL interpretation message(s) :
         FMR_ELAB-146 FMR_VHDL-1014 FMR_ELAB-147 FMR_VHDL-1002
         were produced during link. (FM-262)

Error: Failed to set top design to 'r:/WORK/minimips' (FM-156)
```

4) Include the pertinent error message ID's in the Formality script using the variable *hdlin_warn_on_mismatch_message* to turn them into warning messages instead.

5) Continue this process until you get a reference container successfully built, and a final successful verification.

6) There are two other ways to convert simulation/synthesis mismatch error messages into warnings. You should make sure that the RTL is correctly specified before using either of these variables. They will globally convert all mismatch messages into warnings.

   - "set hdlin_error_on_mismatch_message false"

   - "set synopsys_auto_setup true" - This variable enables the Auto Setup Mode which also sets the previous variable's value.

7) Another way to automatically create a FM Tcl script is to use the UNIX command *fm_mk_script*. Try the following:

   fm_mk_script ./netlist_w_svf/minimips.svf

8) View the resulting FM Tcl script *fm_mk_script.tcl*, and try it out with Formality.

9) Exit Formality.

FM Lab3: Basic Debugging

Objective: 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 potential problems. For this lab, ignore any simulation/synthesis mismatch messages.

2a) Transcript messages:

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

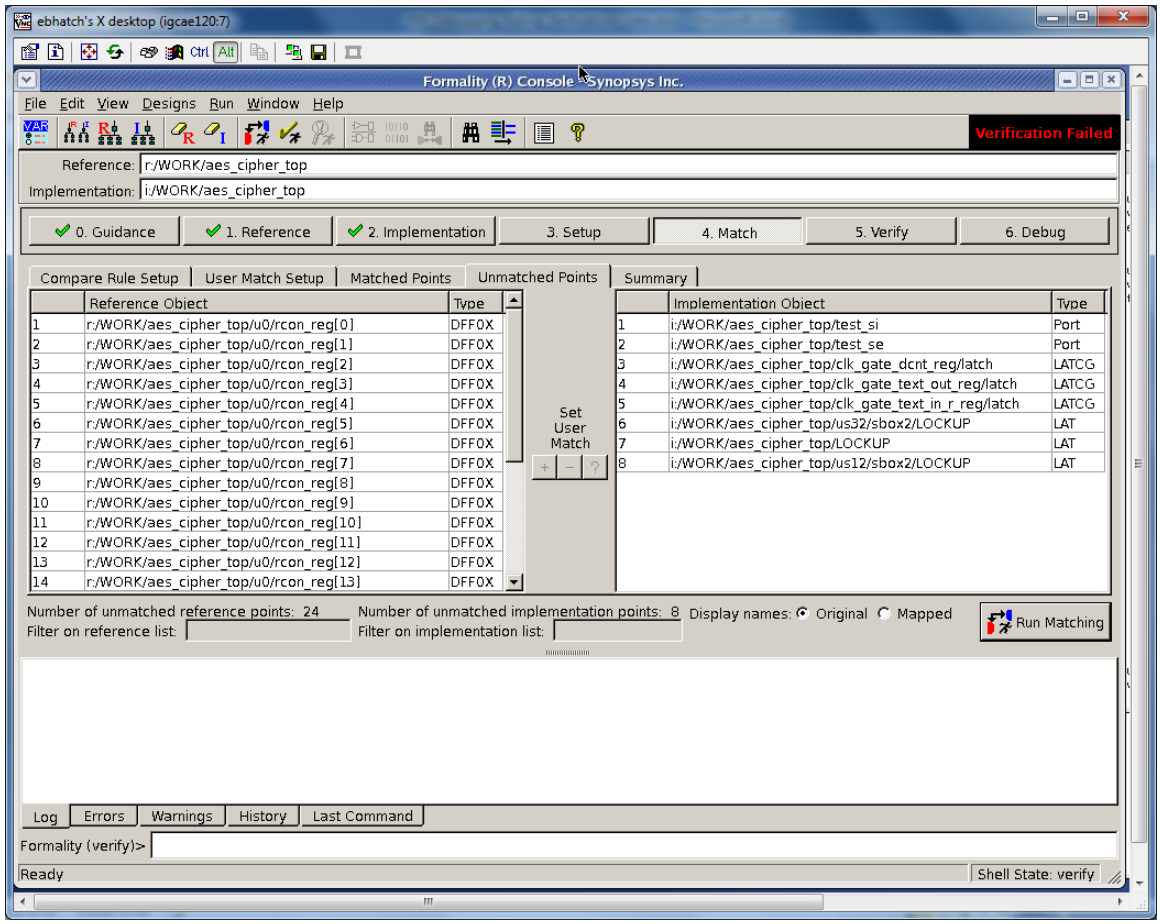
Note the SVF Guidance Summary in the transcript:

```
***************************** Guidance Summary ******************************
                                        Status
Command                 Accepted   Rejected  Unsupported  Unprocessed  Total
-------------------------------------------------------------------------------
change_names        :         23          1            0            0         24
environment         :          4          0            0            0          4
instance_map        :          3          0            0            0          3
inv_push            :         16          0            0            0         16
mark                :          4          0            0            0          4
reg_constant        :         24          0            0            0         24
scan_input          :          0          1            0            0          1
uniquify            :         33          0            0            0         33
```

Note that unless the Auto Setup Mode is enabled, Formality will ignore scan_input guidance found in the SVF file.

2b) Messages from matching:

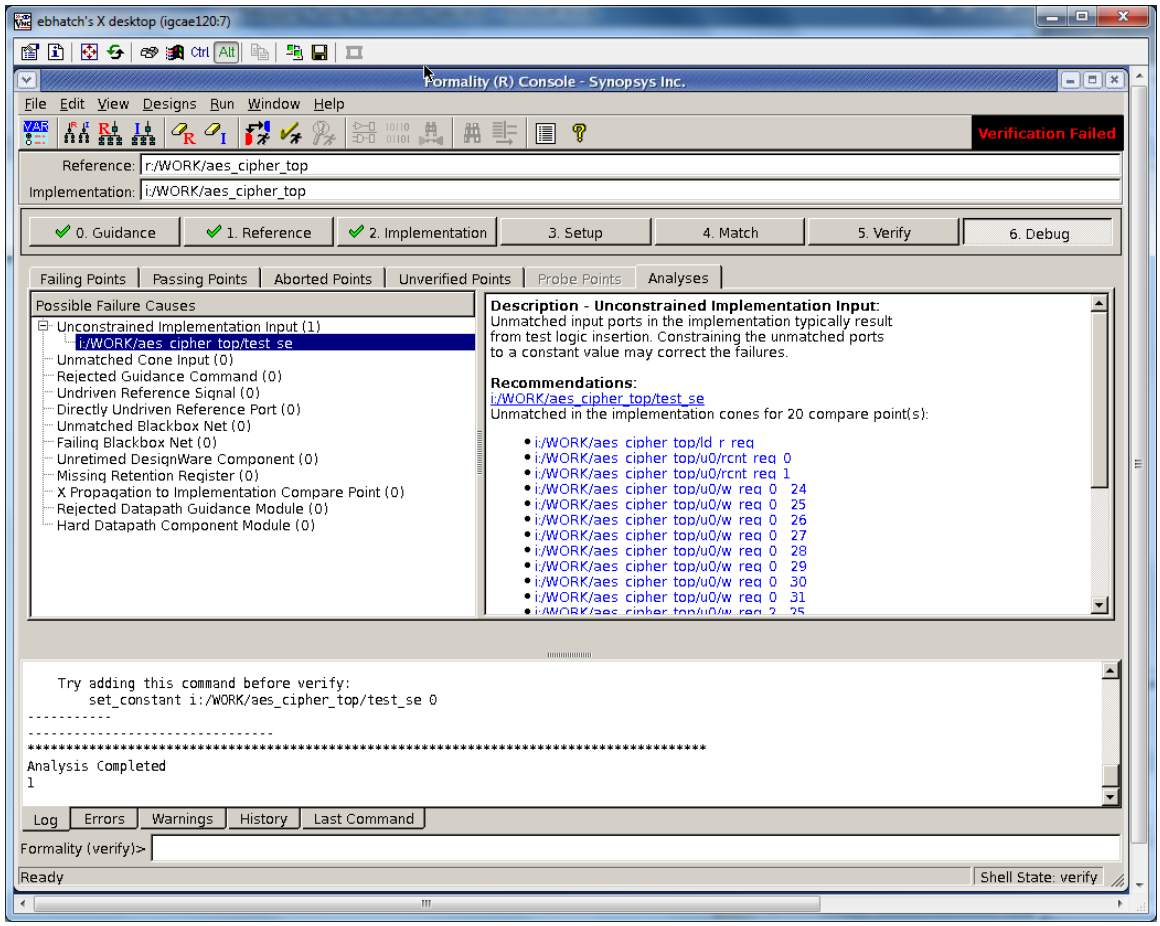
Bring up the GUI and view the Match tab. Here is a picture of the GUI unmatched points tab.



Note the unmatched input ports in the implementation design. This is an indication of test inputs in the netlist. At least one of these *test…* inputs will probably need to be set to a constant to disable scan mode.

2c) Verification debugging:

Under the Debug tab, run "Analyze". Formality provides hints concerning some of the failing compare points. It suggests the syntax for disabling scan mode for this testcase. This is the correct solution to resolve the problem; however, let’s continue on in using some additional debugging tools.

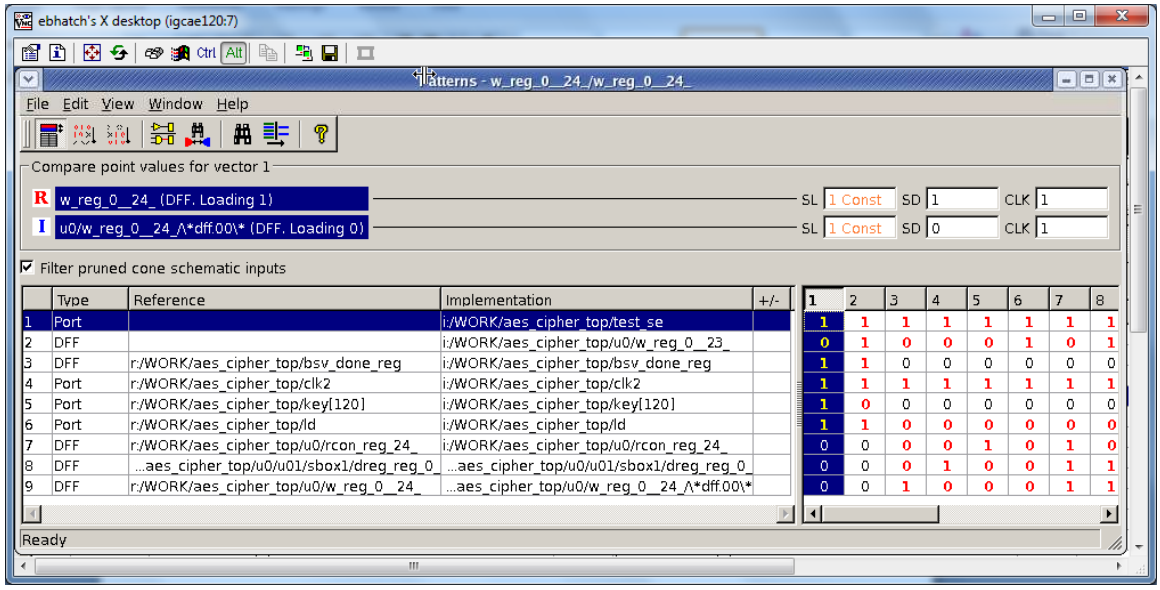


2d) Pattern Viewer:

View the pattern window of one of the failing compare points.

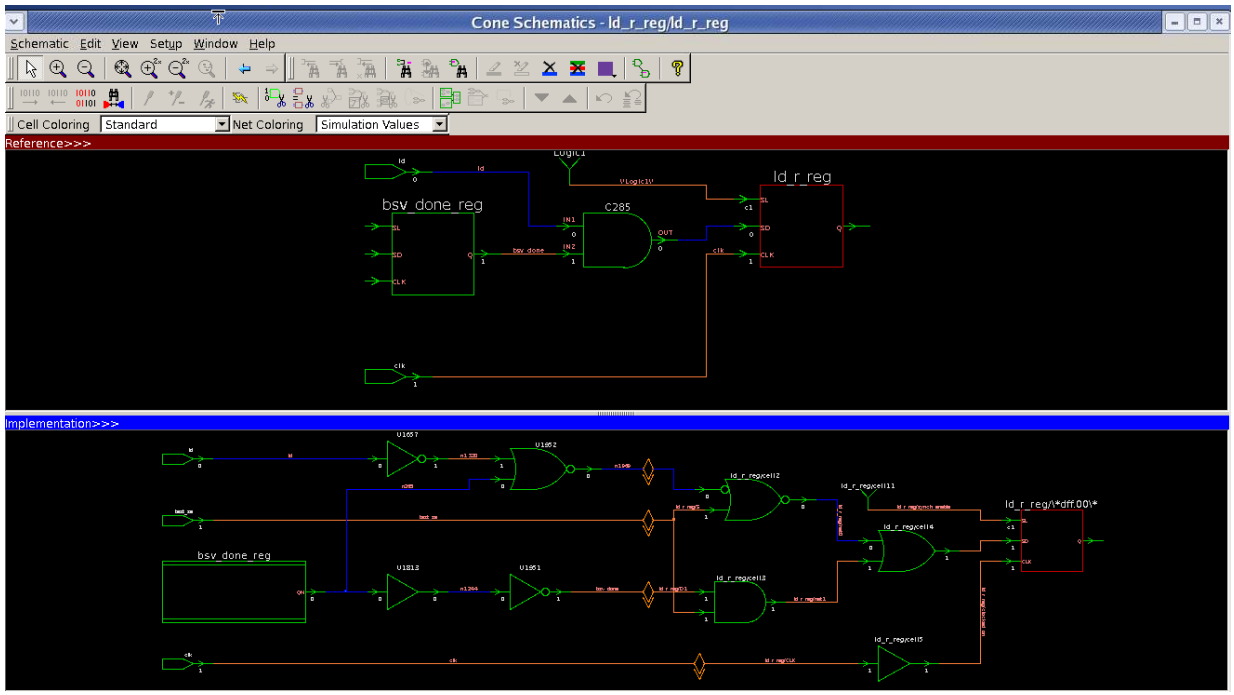
Notice there is the unmatched implementation input port "test_se". Verification fails every time test_se is a “1”.

At this point we know that we can try setting a constant “0” on port “test_se” will probably solve the issue.

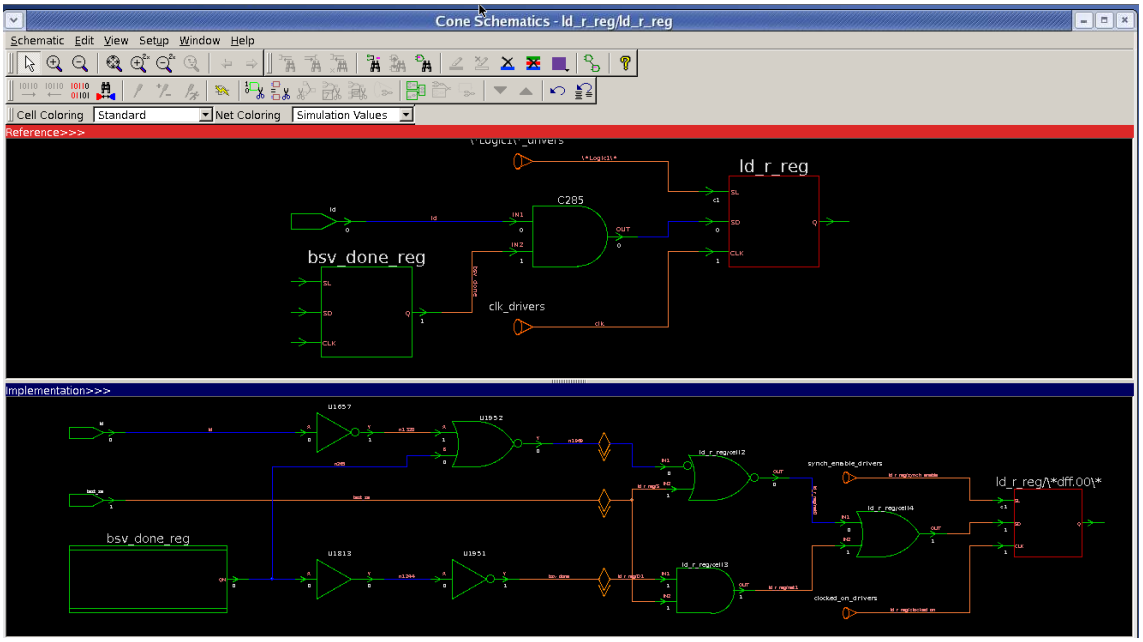


3) Logic Cone Viewer: Bring up the logic cone of one of the compare points by double-clicking on the failing compare point "r:/WORK/aes_cipher_top/ld_r_reg". Or, select this point, right-mouse click to bring up the popup menu. Select "Show Logic Cones".

Try some of the zoom features. Try using the keyboard shortcut keys "i", "o", "f", "."



3a) Use either the icon, or the popup menu, to "Remove Equal Valued CP Inputs". Formality then shows only the differences between the logic cones. This is very helpful in debugging large logic cones.



3b) Click the "10110" icon to bring up the pattern viewer again. Check the box in the pattern viewer for "Filter pruned cone schematic inputs". If you were to debug a larger logic cone, this correlation between pruning in the logic cone and pattern viewer would help you more quickly isolate differences between the designs.

3c) In the pattern viewer, double-click the *test_se* port name. Formality will switch back to the logic cone viewer and will highlight the port.

3d) In the logic cone viewer, select the "test_se" port in the implementation design. Then, use the pull-down menu for "Setup" at the top of the logic cone window to set a constant 0 on the port.

3e) This will bring up the command queue window. When ready, hit the "execute queue" button. This will take Formality back to "setup" mode and will set the constant.

If you were to do this manually, the equivalent commands are the following:

```
setup; set_constant i:/WORK/aes_cipher_top/test_se 0
```

4) Re-run "verify" to get a successful verification.

5) Another way to disable scan is to use the Auto Setup mode. Modify the FM Tcl script to use "set synopsys_auto_setup true". Re-run verification. Formality will disable scan by using the information in the SVF file.