VC Formal
Formal Coverage Analyzer (FCA) App
Lab 1: Unreachability Analysis

Learning Objectives

In this VC Formal lab, you will use a traffic light controller example to learn to do the following:

- Import a coverage database generated by simulation
- Set up design
- Run interactively with Verdi GUI
- Run FCA on goals not yet covered in simulation
- Generate updated coverage database
- Generate exclusion file for items found
- View updated coverage in Verdi Coverage GUI
- Save and restore session
- Run in interactive shell without Verdi GUI
- Run in batch mode

Lab Duration:
30 minutes

Familiarity with VCS coverage, knowledge of the basic formal verification concept and basic setup using VC Formal are required for this lab

Files Location

All files for this VC Formal lab are in directory:
    $VC_STATIC_HOME/doc/vcst/examples/FCA/FCA_with_SimCovdb/

Directory Structure	
FCA_with_SimCovdb	Lab main directory
README_VCFormal_FCA_Lab1.pdf	Lab instructions
design/	Verilog RTL code of the Device Under Test (DUT)
TB/	Directed simulation testbench for the DUT
run/	Run directory
solution/	Solution directory

Resources

The following resources are available for in-depth guidance regarding VC Formal usage, commands, and variables.

VC Formal User Guide:
$VC_STATIC_HOME/doc/vcst/VC_Formal_Docs/VC_Formal_UG.pdf

VC Formal Apps Quick References Guides:
$VC_STATIC_HOME/doc/vcst/VC_Formal_Docs/Quick_Reference_Guides/

VC Formal Apps Tcl Templates:
$VC_STATIC_HOME/doc/vcst/VC_Formal_Docs/Quick_Reference_Guides/vcf_tcl_templates/

Prepare your Environment

1. Set environment variable pointing to your VC Formal installation directory:

```
%setenv VC_STATIC_HOME /tools/synopsys/vcstatic
```

2. Set environment variable pointing to your VCS installation directory:

```
%setenv VCS_HOME <path of VCS-MX installation>/vcs-mx
```

3. Add paths $VC_STATIC_HOME/bin and $VCS_HOME/bin to the PATH environment variable.

```
%set path=($VCS_HOME/bin $VC_STATIC_HOME/bin $path)
```

4. Change your working directory to FCA/FCA_with_SimCovdb/run:

```
%cd FCA/FCA_with_SimCovdb/run
```

Now you are ready to begin the lab.

Generate Simulation Coverage Database

5. Open file run_sim.csh (any arbitrary name is ok to use) using any text editor:

```
%vi run_sim.csh
```

6. Enter command for VCS simulation:

The DUT files and filelist are located under FCA_with_SimCovdb/design.

```
vcs -f ../design/filelist_sim -cm line+cond+tgl+fsm -R
```

7. Save run_sim.csh file and exit editor.

8. Run VCS simulation:

```
%chmod +x run_sim.csh
%./run_sim.csh
```

The simulation coverage database to read in the FCA App is generated.

Create a run.tcl Setup File

VC Formal has a Tcl-based command interface. It is common to start with a Tcl file to set up

and compile a design. In this step, you will create a VC Formal Tcl file for the DUT, a traffic light controller, used in this lab.

9. Open file run.tcl (any arbitrary name is ok to use) using any text editor:

```
%vi run.tcl
```

10. Add command to enable FCA App mode:

```
set_fml_appmode COV
```

11. Specify DUT top level module name as Tcl variable:

```
set design traffic
```

12. Add command to load simulation coverage database:

```
read_covdb -cov_input simv.vdb -cov_dut TB.T1
```

The `-cov_dut` switch is used to specify the DUT top level instance in the testbench.

13. Add command to compile DUT and enable automatic code coverage extraction:

The DUT files and filelist are located under directory FCA_with_SimCovdb/design.

```
read_file -top $design -format sverilog \
  -vcs {-f ../design/filelist} -cov all
```

Note: Switch `-cov all` instruments the RTL for line, condition, fsm_state, fsm_trans and toggle coverage metrics. To enable a single coverage metric, specify `-cov <metric>`. To specify multiple coverage metrics, specify `-cov <metric1 + metric2>`, e.g., `-cov line` or `-cov line+cond`.

Note: To use unified usage model to compile design, use these commands instead of `read_file` to compile design and SVA properties:

```
analyze -format sverilog \
  -vcs {-f ../design/filelist}
elaborate $design -sva -cov all
```

14. Save run.tcl file and exit editor.

VC Formal can be run in three modes: interactive Verdi GUI mode, interactive without Verdi GUI using shell mode, and non-interactive batch mode.

Mode 1: Start VC Formal in Verdi GUI Mode

15. Start the tool in Verdi GUI mode:

```
%vcf -f run.tcl -verdi
```

VC Formal starts in the Verdi GUI mode, with icons, tables, tabs, and windows especially designed for verification with the FCA App.

Initial configuration is shown with the "VCF:TaskList" tab on the top left, the "VCF:GoalList" tab on the top right, and the "VC Formal Console" shell at the bottom.
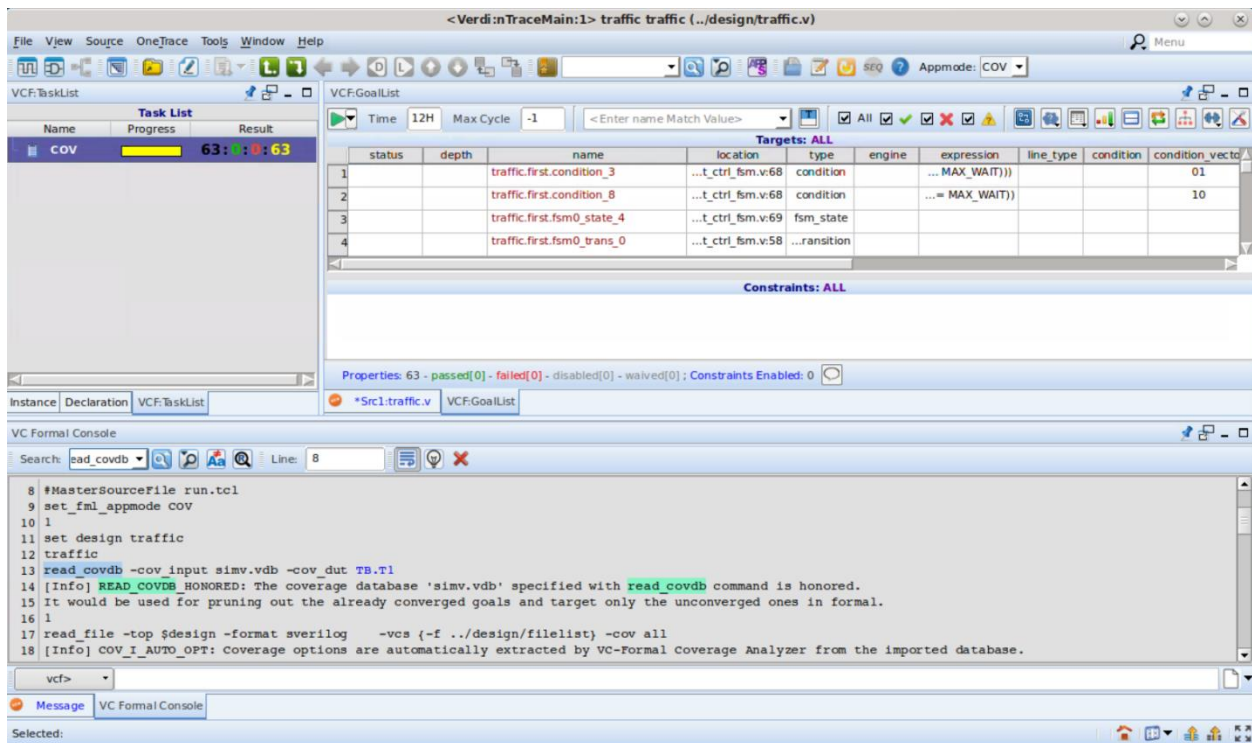
16. Get familiar with the Verdi GUI instance:

Check the source file tabs, properties to be checked under the "Targets: ALL" table as well as properties specified as constraints in the "Constraints: ALL" table.

If desired, customize the columns shown by clicking on the Customize View Settings icon at the upper right of the property table.

Review Design Information and Setup

17. Review design information:

    Click on the Show Complexity icon ![icon] on the upper right above the property table.
    Examine items under "Missing Clock" and "Missing Reset" and trace from the source file to
    see that the active phase of rst is high.

Revise Setup and Review Initial State

18. Add missing clock and reset information to the Tcl file:

    Click on the Edit Tcl Project File icon ![icon] on the upper left and add the following
    commands to the Tcl file.

    ```
    create_clock clk -period 100
    create_reset rst -sense high
    ```

19. Add design initialization commands:

    ```
    sim_run -stable
    sim_save_reset
    ```

    These commands initialize the DUT by holding reset active until sequential elements
    (latches and flip flops) values are stable.

20. Save edited run.tcl Tcl file:

    Click on the Save icon ![icon].

21. Restart VC Formal:

    Click on the Restart VCST icon ![icon].

22. Check setup and debug initial state:

    Click on the Show Complexity icon ![icon] on the upper right to see design information and
    confirm there are more missing clock and reset shown.

    Examine the initial state of latches and flip flops to check if anything is unexpected.

23. Exit VC Formal:

    Click on File → Exit.

Run Formal Proofs and Review Results

24. Start VC Formal with existing Tcl file:

    Now that you have a correct setup, start VC Formal pre-reading the existing run.tcl file.

    ```
    %vcf -f run.tcl -verdi
    ```

25. Enable trace generation for covered goals:

    ```
    vcf> set_fml_var fml_cov_gen_trace on
    ```

26. Start property verification:

    Click on the Start Check icon ▶.

27. Hide the constraints table:

    Click on the Targets+Constraints icon 🔲 at the top right.

28. Filter "Targets" table to keep <u>uncoverable</u> targets:

    Select to view only uncoverable targets ☐ All ☐ ✔ ☑ ✖ ☐ ⚠.

29. Filter "Targets" table to keep <u>covered</u> targets:

   Select to view only covered targets ☐ All ☑ ✔ ☐ ✖ ☐ ⚠.

View Trace for Covered Targets

30. View trace for covered target:

    Double click on ✔ under the status of property "traffic.first.condition_3" to open a waveform showing how the condition was covered.
    (Note that a double click on the "name" of a property will open the source code window instead.)

    Alternatively, open a trace using the right-click menu: View Trace → Property+Reset
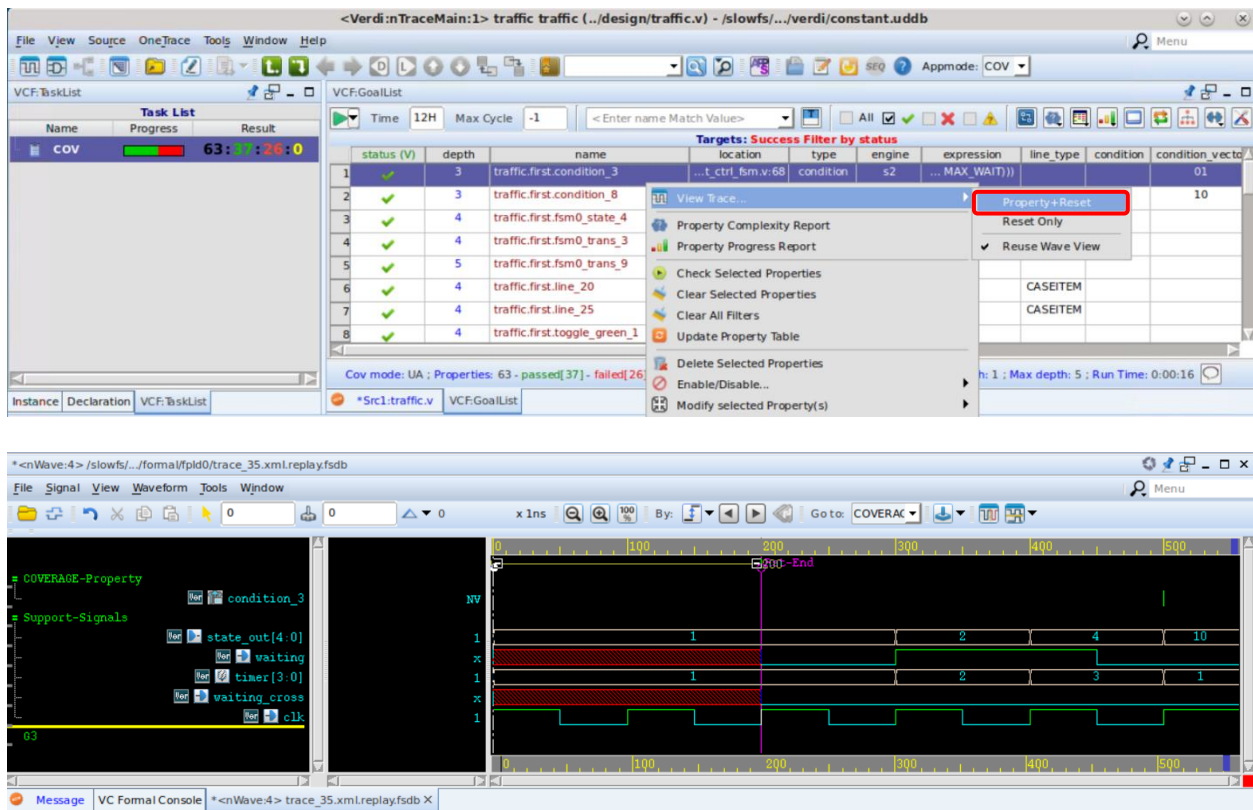
View Coverage in Verdi Coverage GUI

31. Open updated coverage database in Verdi Coverage GUI:

    Click on [icon] to dump a coverage database updated with FCA results and an elfile (exclusion file) for the uncoverable targets. The Verdi Coverage GUI automatically comes up with the updated coverage and exclusion file information.

    You can also apply the exclusion file to the original simulation coverage database to refine the coverage score.

Save Session

32. Save session using default name from the VC Formal Shell:

```
vcf> save_session
```

    Alternatively, click on File → Save Session… to open the "Save Session" dialog window.

33. Exit VC Formal:

    Click on File → Exit

34. Start VC Formal using previous saved session:

```
%vcf -restore -verdi
```

35. Review setup and results then exit:

Click on File → Exit

36. Invoke VC Formal without Verdi GUI:

```
%vcf -f run.tcl
```

37. Enable trace generation for covered goals:

```
vcf> set_fml_var fml_cov_gen_trace on
```

38. Enter run check command in VC Formal Shell:

```
vcf> check_fv
```

39. When check completes, report results, save coverage database and exclusion file:

```
vcf> report_fv -list > results.txt
vcf> save_covdb -name cov -status covered -overwrite
vcf> save_cov_exclusion -file unr.el
```

Alternatively, enter run check command with callback task:

```
vcf> check_fv -run_finish {
report_fv -list > results.txt
save_covdb -name cov -status covered -overwrite
save_cov_exclusion -file unr.el
}
```

40. Start the Verdi GUI from within the VC Formal Shell:

```
vcf> start_verdi
```

To analyze traces of covered goals, it is recommended to use the Verdi GUI.

Note that the VC Formal Shell panel will not be available in the Verdi GUI. Any Tcl command will need to be entered from the `vcf>` prompt where you used the `start_verdi` command.

Also, the debug waveform may not be embedded in the same window as before. You can always click on ⊞ to dock (or undock) a window.

41. View coverage report from within the VC Formal Shell:

```
vcf> view_coverage -cov_input cov.vdb -elfile unr.el
```

42. View trace of covered goal "traffic.first.condition_3" from within the VC Formal Shell:

```
vcf> view_trace -property traffic.first.condition_3
```

Optionally, add switch -composite to show the reset phase in the trace.

43. Exit VC Formal:

```
vcf> quit
```

44. Copy Tcl file run.tcl to run_batch.tcl:

```
%cp run.tcl run_batch.tcl
```

45. Edit run_batch.tcl and enable trace generation for covered goals:

```
set_fml_var fml_cov_gen_trace on
```

46. Add commands to run and save results:

```
check_fv -block
report_fv -list > results.txt
save_covdb -name cov -status covered -overwrite
save_cov_exclusion -file unr.el
```

47. Add command to save session:

```
save_session -session batch_results
```

48. Save run_batch.tcl file and exit editor.

49. Start VC Formal in batch mode with switch -batch:

```
%vcf -f run_batch.tcl -batch
```

Note that in batch mode VC Formal exits automatically after the execution of the Tcl command file.