

VC Formal Lab

Connectivity Check (CC) App Setup and Standard Usage

Learning Objectives

In this VC Formal lab, you will use the multiplier example to learn to do the following:

- Setup compile
- Setup CSV file parameters
- Load csv file
- Establish initial state for formal
- Run checks
- Debug failure
- Run in interactive shell without Verdi GUI
- Run in batch mode

Familiarity with the SystemVerilog Assertion (SVA) language and knowledge of basic formal verification concepts are required for this lab.



Lab Duration:
30 minutes

Testcase Location

All files for this VC Formal lab are in directory:

\$VC_STATIC_HOME/doc/vcst/examples/CC/

Directory Structure	
CC	Lab main directory
README_CC_setup.pdf	Lab instructions
design/	Verilog RTL code of the Device Under Test (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_Forma Docs/VC_Forma_UG.pdf

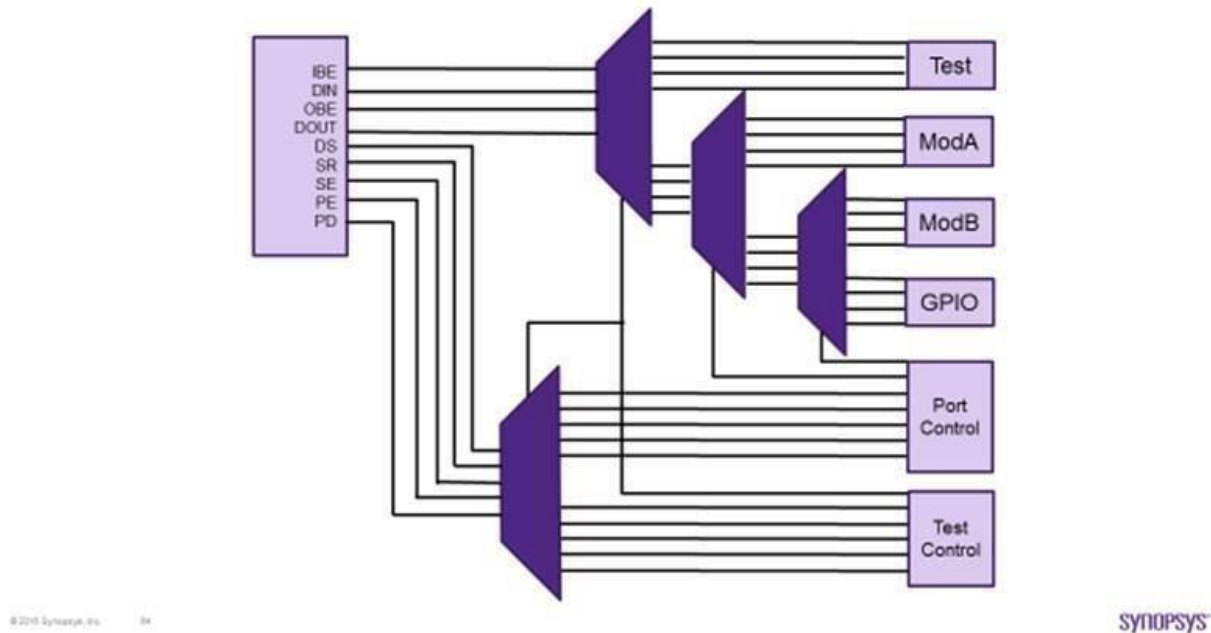
VC Formal Apps Quick References Guides:

\$VC_STATIC_HOME/doc/vcst/VC_Forma Docs/Quick_Reference_Guides/

VC Formal Apps Tcl Templates:

\$VC_STATIC_HOME/doc/vcst/VC_Forma Docs/Quick_Reference_Guides/vcf_tcl_templates/

Design schematic



Prepare your Environment

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

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

2. Add path \$VC_STATIC_HOME/bin to the PATH environment variable.
3. Change your working directory to CC/run:

```
%cd CC/run
```

Now you are ready to begin the lab.

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 multiplier, used in this lab.

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

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

```
%vi run.tcl
```

5. Add command to enable CC App mode:

```
set_fml_appmode CC
```

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

```
set design chip_top
```

7. Add command to compile DUT:

```
read_file -top $design -format verilog \  
-vcs {../design/cctest.v}
```

Or it can be separate into 2-step: analyze and elaborate
This is suitable for complex design or mix-language design.

```
analyze -format verilog -vcs {../design/cctest.v}  
elaborate $design
```

8. Add commands set csv file parameters. It gives information to the tool how to read the connections from csv file.

```
load_cc_set_param csv_enable "%1%"  
load_cc_set_param csv_from "%2%"  
load_cc_set_param csv_to "%3%"  
load_cc_set_param csv_prop_name "%4%"  
load_cc_set_param csv_start_line "2"
```

9. Add command load connections from csv file .

```
load_cc -format csv cctest.csv
```

10. 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.

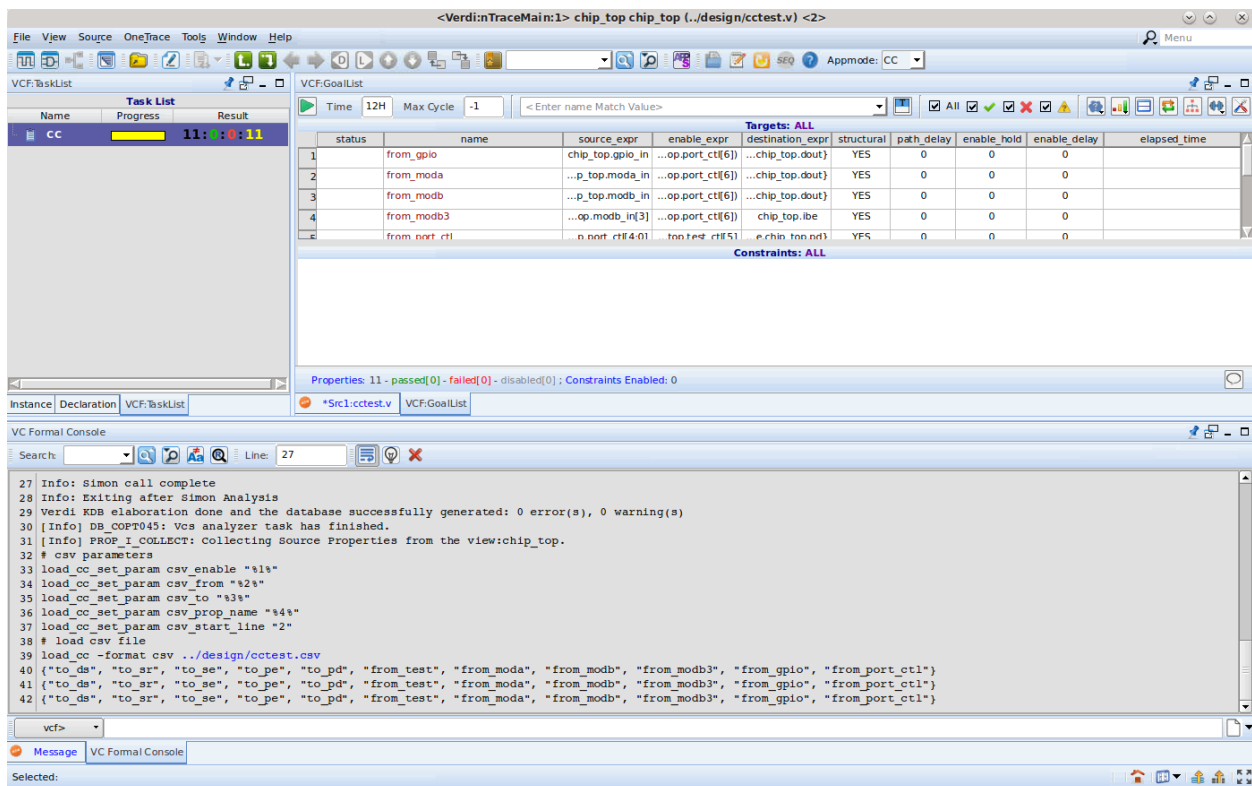
Mode 1: Start VC Formal in Verdi GUI Mode

11. 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 property verification with the FPV App. The App mode is set to FPV by default. Using CC app mode variable in step 2 it will start in CC mode

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.




12. 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.

Revise Setup and Review Initial State

13. Add commands to set up reset state.

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


```
sim_save_reset
```

14. Save edited run.tcl Tcl file:

Click on the Save icon .

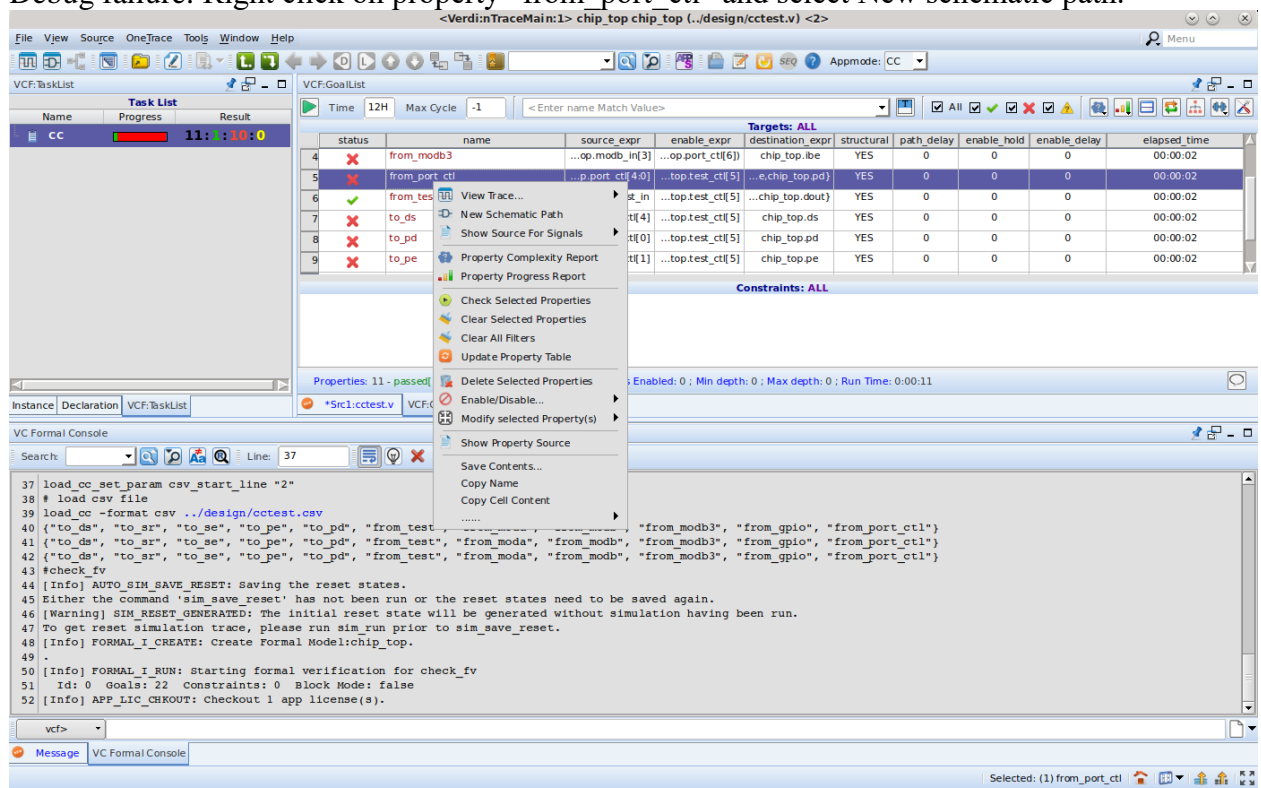
15. Restart VC Formal:

Click on the Restart VCST icon .

16. Start property check: Click on  run to finish. This takes about 20-30 seconds

Debug Failure

17. Debug failure: Right click on property “from_port_ctl” and select New schematic path.



The screenshot shows the Synopsys VC Formal interface. The 'VCF:TaskList' pane on the left shows a task 'CC' with a status of '11: 11: 0: 0'. The main console area displays a list of properties and their status. A context menu is open over the 'from_port_ctl' property, showing options like 'New Schematic Path'. The console output shows the following messages:

```

37 load_oc_set_param csv_start_line "2"
38 # load csv file
39 load_oc -format csv ../design/cctest.csv
40 {"to_ds", "to_sr", "to_se", "to_pe", "to_pd", "from_test", "from_modb3", "from_gpio", "from_port_ctl"}
41 {"to_ds", "to_sr", "to_se", "to_pe", "to_pd", "from_test", "from_moda", "from_modb", "from_modb3", "from_gpio", "from_port_ctl"}
42 {"to_ds", "to_sr", "to_se", "to_pe", "to_pd", "from_test", "from_moda", "from_modb", "from_modb3", "from_gpio", "from_port_ctl"}
43 #check_fv
44 [Info] AUTO_SIM_SAVE_RESET: Saving the reset states.
45 Either the Command "sim_save_reset" has not been run or the reset states need to be saved again.
46 [Warning] SIM_RESET_GENERATED: The initial reset state will be generated without simulation having been run.
47 To get reset simulation trace, please run sim_run prior to sim_save_reset.
48 [Info] FORMAL_I_CREATE: Create Formal Model:chip_top.
49 .
50 [Info] FORMAL_I_RUN: Starting formal verification for check_fv
51 Id: 0 Goals: 22 Constraints: 0 Block Mode: false
52 [Info] APP_LIC_CHKOUT: checkout 1 app license(s).

```


Correct Erroneous Connection

20. For this connection, source is `chip_top.port_ctl[4:0]`, destination is `{ds,sr,se,pe,pd}` and enable condition is `~chip_top.test_ctl[5]`. From schematic, mux enable signal is 0 hence source is not equal to destination.
21. To make this connection pass, edit `../design/cctest.csv` file and replace `~chip_top.test_ctl[5]` with `chip_top.test_ctl[5]` (last line of csv file).

Restart the Run and Verify Fix

22. Restart VC Formal with the modified assertion:

Click on the Restart VCST icon .

23. Re-run property verification:

Click on the Start Check icon .

Observe that the same property is now proven.

Note that there are still more falsified properties. You may choose to debug them on your own. The corrected connections can be found in file: `CC/solution`.

Save Session

24. 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.

25. Exit VC Formal:

Click on File → Exit

Restore Session

26. Start VC Formal using previous saved session:

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

27. Review setup and results then exit:

Click on File → Exit

Mode 2: Start VC Formal in Interactive Non-Verdi GUI Mode

28. Invoke VC Formal without Verdi GUI:

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

29. Enter run check command in VC Formal Shell:

```
vcf> check_fv
```

30. When check completes, report results:

```
vcf> report_fv -list
```

Alternatively, enter run check command with callback task:


```
vcf> check_fv -run_finish {report_fv -list > results.txt}
```

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

```
vcf> start_verdi
```

To debug failures, 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.

32. Exit VC Formal:

```
vcf> quit
```

Mode 3: Set Up and Run VC Formal in Batch (Regression) Mode

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

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

34. Edit run_batch.tcl and add commands to run and save results:

```
check_fv -block  
report_fv -list > results.txt
```

35. Add command to save session:

```
save_session -session batch_results
```

36. Save run_batch.tcl file and exit editor.

37. 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.