

VC Formal Lab

Formal X-Propagation (FXP) App Setup and Standard Usage

Learning Objectives

In this VC Formal lab, you will use a simple valid pipeline example to learn to do the following:

- Set up design and appmode
- Run interactively with Verdi GUI
- Review design complexity statistics and setup
- Set up clocks and resets
- Establish initial state for formal
- Generate property
- Run checks and review results
- Debug failures
- Save and restore session
- Run in interactive shell without Verdi GUI
- Run in batch mode

Familiarity with basic formal verification concepts are required for this lab.



Lab Duration:
30 minutes

Files Location

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

Directory Structure	
FXP	Lab main directory
README_VCFormal_FXP.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/

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 FXP/run:

```
%cd FXP/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 simple pipeline example, used in this lab.

The DUT files are located under FXP/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 FXP App mode:

```
set_fml_appmode FXP
```

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

```
set design pipeline
```

7. Add command to compile DUT and SVA properties :

```
read_file -top $design -format sverilog -sva \  
-vcs { ../design/pipeline.v}
```

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

```
analyze -format sverilog -sva -vcs {../design/pipeline.v}  
elaborate $design
```

8. Save run.tcl file and exit editor.

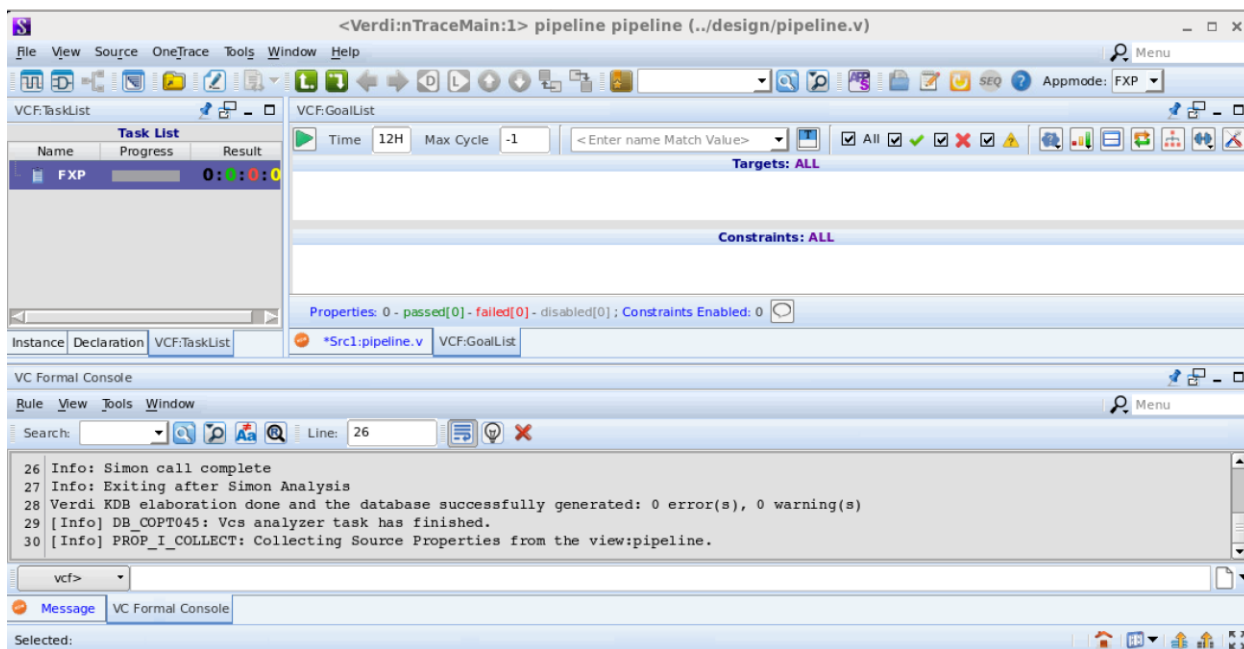
VC Formal can be run in three modes: interactive Verdi GUI mode, interactive without Verdi GUI using shell mode, and non-interactive batch mode.

Mode 1: Start VC Formal in Verdi GUI Mode

9. 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. We had overridden this in our “run.tcl file via command “set_fml_appmode FXP” so GUI has FXP-mode set.

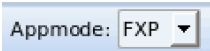



Initial configuration is shown with the “VCF:TaskList” tab on the top left, the “VCF:GoalList” tab on the top right, and the “VC Formal Console” shell at the bottom.

10. 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.

Review Design Information and Setup

11. Review design information and setup  on top menu bar ensuring you are in FXP-mode, and in TaskList you can see 0-FXP property.
12. Click on the Show Complexity 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 resetn is low.

Revise Setup and Review Initial State

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

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

```
create_clock clk -period 100
create_reset resetn -sense low
```

Add commands to initialize the DUT by holding reset active until sequential elements (latches and flip flops) values are stable. And save the initial state.

```
sim_run -stable
sim_save_reset
```

14. Generate injection and observation points for FXP checking. In this example just use the default points, to see other options, type *fxp_generate -help* in the vcf prompt

```
fxp_generate
```

15. Save edited run.tcl Tcl file:

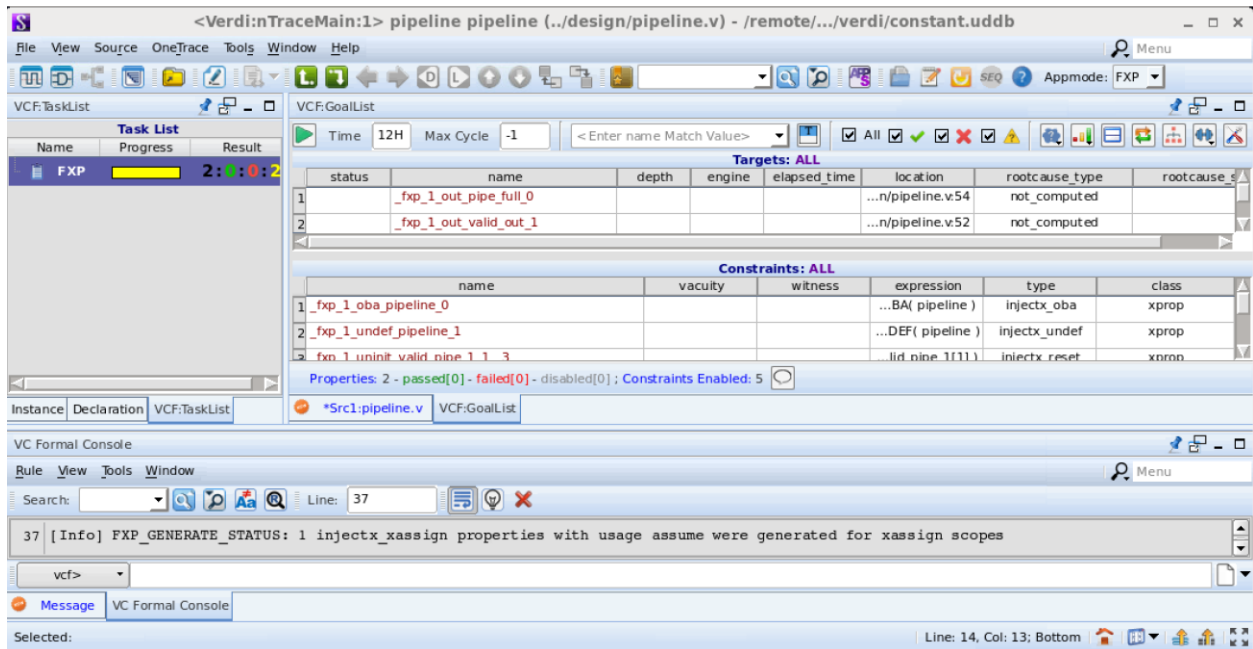
Click on the Save icon .

16. Restart VC Formal:

Click on the Restart VCST icon .


17. Check setup and design information:

There are 2-FXP properties show in TaskList, and the status for these properties is blank due to checks has not been run yet.



Run FXP check and Review Results

18. Start formal x-prop verification:



Click on the Start Check icon .

19. Hide the constraints table:

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


20. Check result in Targets:ALL:

2-FXP: 1 proven , 1 falsified .

Targets: ALL							
	status	name	depth	engine	elapsed_time	location	rootcause_type
1		_fxp_1_out_pipe_full_0	1	t1	00:00:01	...n/pipeline.v:54	not_computed
2		_fxp_1_out_valid_out_1	3	rf1	00:00:02	...n/pipeline.v:52	not_computed

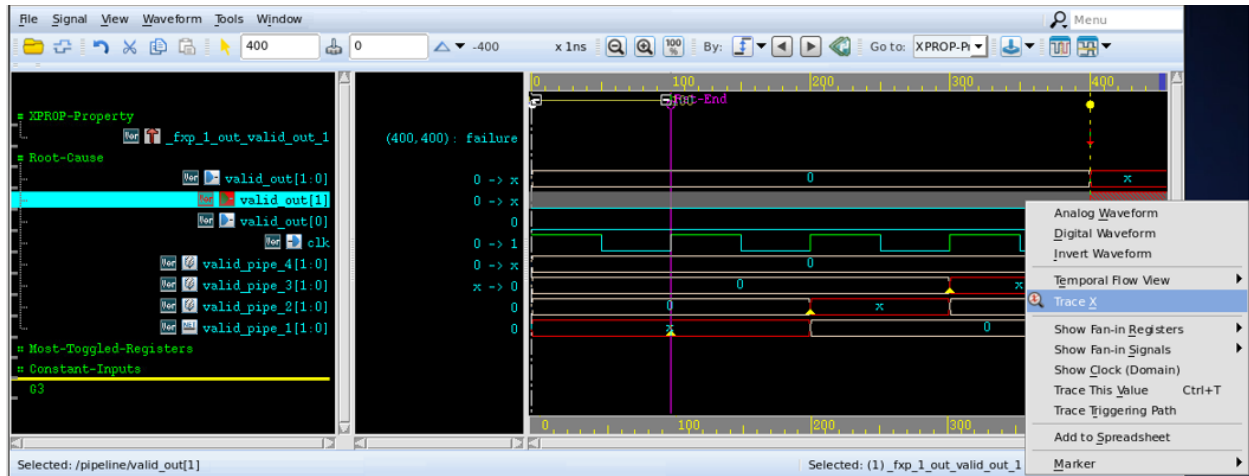
Can click  to filter status.

Debug Failure

21. Double click on  under the status of property “_fxp_1_out_valid_out_1” to open a counter-example waveform.
(Note that a double click on the “name” of a property will open the property in the source

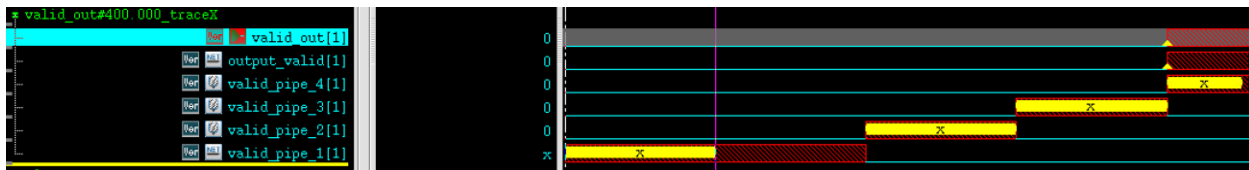
code window instead.)

22. Double click on the valid_out signal where the X has occurred to expand the bus. Click on the waveform to move the cursor to the point of this X transition. Right click on the waveform and select “Trace X”.

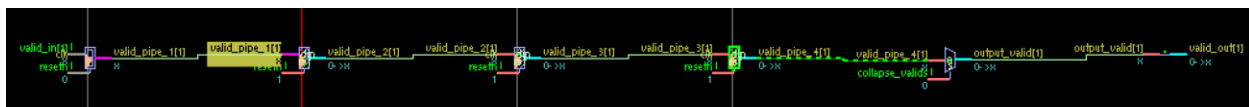


23. The tool will trace the X back to its source and display the point of failure by mark it yellow in waveform. Also, tool will generate temporal flow view for schematic.

In waveform



In schematic (Temporal Flow View)



24. Drag the signal “valid_pipe_1” from waveform to source code and can see only bit 0 of valid_pipe_1 is reset and the X can propagate through the pipeline and appear on the output.

```

21  always @(posedge clk or negedge resetn)
           1           0
           C:1
22  if (!resetn)
           0
           C:1
23  valid_pipe_1[0] <= 1'b0;
           0
24  else
25
25  valid_pipe_1 <= valid_in;
           x           x


```

25. The root cause for type and related signal also show in Target table after generating counter-example waveform.

Targets: ALL							
	status	name	depth	engine	elapsed_time	location	rootcause_type
1	✓	_fxp_1_out_pipe_full_0		t1	00:00:01	...n/pipeline.v:54	not_computed
2	✗	_fxp_1_out_valid_out_1	3	rf1	00:00:02	...n/pipeline.v:52	uninitialized_register
							valid_pipe_1

Correct Erroneous in RTL Design

26. To correct error, we need fix the RTL-source file ‘pipeline.v’.

The design can be edit on the GUI interface by click on Verdi “Edit Source File” icon  and modify the file below:

Original design:

```

always @(posedge clk or negedge resetn)
  if (!resetn)
    valid_pipe_1[0] <= 1'b0;
  else
    valid_pipe_1 <= valid_in;

```

Modified design:

```

always @(posedge clk or negedge resetn)
  if (!resetn)
    valid_pipe_1 <= 2'b0;
  else
    valid_pipe_1 <= valid_in;

```

27. Save file:


Click on the Save icon .

Restart the Run and Verify Fix

28. Restart VC Formal with the modified assertion:

Click on the Restart VCST icon .

29. Re-run property verification:

Click on the Start Check icon .

Observe that there are no falsified properties and all are proven.

Save Session

30. 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.

31. Exit VC Formal:

Click on File → Exit

Restore Session

32. Start VC Formal using previous saved session:

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

33. Review setup and results then exit:

Click on File → Exit

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

34. Invoke VC Formal without Verdi GUI:

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

35. Enter run check command in VC Formal Shell:

```
vcf> check_fv
```

36. 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}
```

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

```
vcf> start_gui
```

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_gui` 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.

38. Exit VC Formal:

```
vcf> quit
```

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

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

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

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

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

41. Add command to save session:

```
save_session -session batch_results
```

42. Save run_batch.tcl file and exit editor.

43. 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 and no need to add “quit” command in the end of Tcl file.