VC Formal Lab
Formal Property Verification (FPV) App Setup and Standard Usage

Learning Objectives

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

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

Lab Duration: 40 minutes

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

Files Location

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

Directory Structure	
FPV	Lab main directory
README_VCFormal_FPV.pdf	Lab instructions
design/	Verilog RTL code of the Device Under Test (DUT)
sva/	SVA properties to check functionality of 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 Guide:
$VC_STATIC_HOME/doc/vcst/VC_Formal_Docs/VC_Formal_Quick_Reference_Guide.pdf

VC Formal Apps Tcl Templates:
$VC_STATIC_HOME/doc/vcst/VC_Formal_Docs/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 FPV/run:

```
%cd FPV/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 traffic light controller, used in this lab.

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

```
%vi run.tcl
```

5. Add command to enable FPV App mode (default when starting VC Formal):

```
set_fml_appmode FPV
```

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

```
set design traffic
```

7. Add command to compile DUT and SVA properties:

The DUT files and filelist are located under directory FPV/design. The assertion and bind files are located under directory FPV/sva.

```
read_file -top $design -format sverilog -sva \
   -vcs {-f ../design/filelist +define+INLINE_SVA \
   ../sva/traffic.sva ../sva/bind_traffic.sva}
```

Since the DUT includes inline properties, the "+define+INLINE_SVA" string is added to the compilation command.

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 +define+INLINE_SVA \
  ../sva/traffic.sva ../sva/bind_traffic.sva}
elaborate $design -sva
```

8. Save Tcl file run.tcl 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.

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.
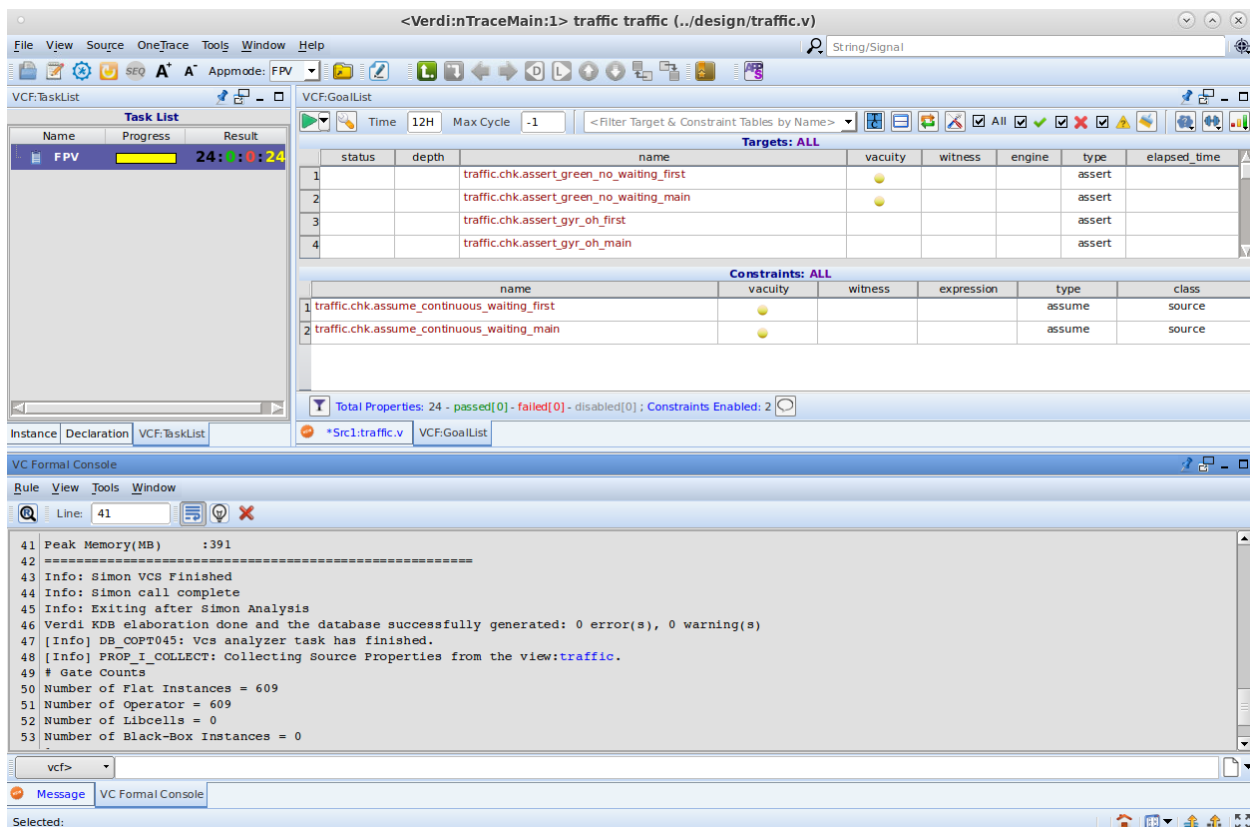
10. Get familiar with the Verdi GUI instance:

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.
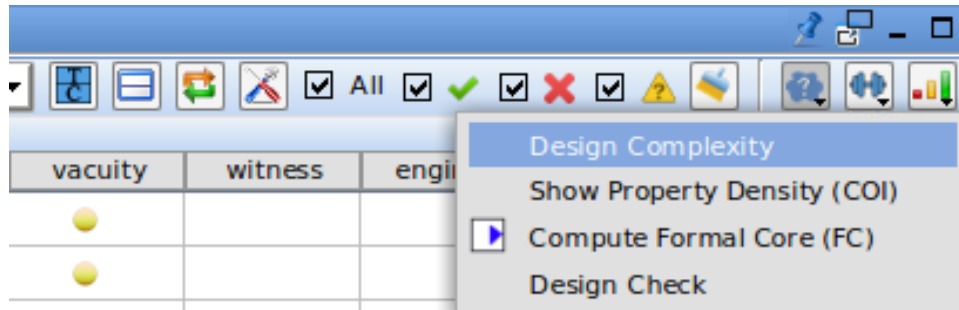
If desired, customize the columns shown by clicking on the Preference Settings icon
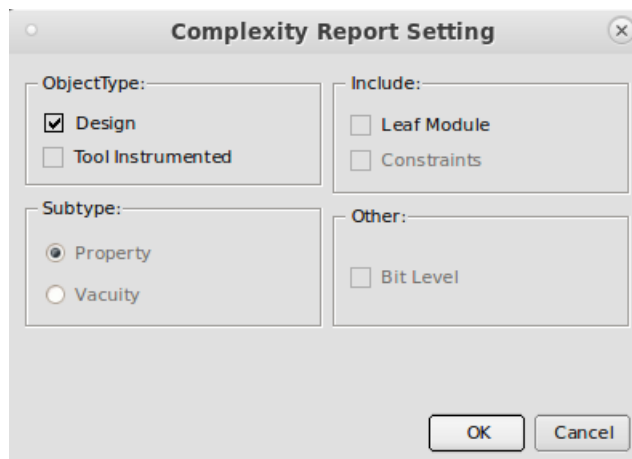
located above the property table.

Review Design Information and Setup

11. Review design information:

Click on the Setup Assist icon ![icon] located above the property table and select Design Complexity.

Click OK in the Complexity Report Setting dialog.

Equivalent command to report design complexity:
```
report_fv_complexity
```

12. Analyze results for missing clock and reset information:

Examine items under "Missing Clock" and "Missing Reset" and trace from the source code window to determine the clock sampling edge is posedge and the active phase of rst is high.

VCF:ComplexityReport

Name	Bits	Initial State	
⊞ **Primary Input: 4**	4		
⊞ **Primary Output: 6**	6		
⊞ **Counter: 2**	8		
⊞ **Flip flop: 3**	11		
⊞ **Latch: 6**	6		
⊟ **Missing Clock: 1**			
clk	1		
⊟ **Missing Reset: 1**			
rst	1		
⊞ **Net: 7**			
⊞ **Port: 40**			
⊞ **Additional Operators**			
⊞ **Arithmetic Operators**			

Message | VC Formal Console | VCF:ComplexityReport

Run Design Checks and Analyze Results

13. Run design checks:

    Click on the Setup Assist icon ⬚ located above the property table and select Design Check
    to run the full set of design checks to validate the current design and setup for potential
    issues such as missing clocks, missing resets, oscillating combinational loops, etc.
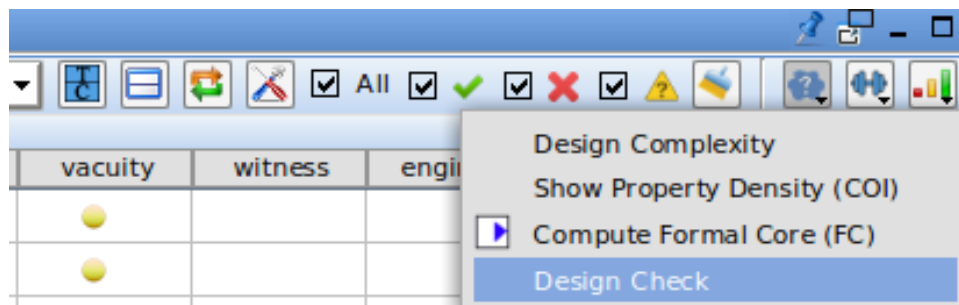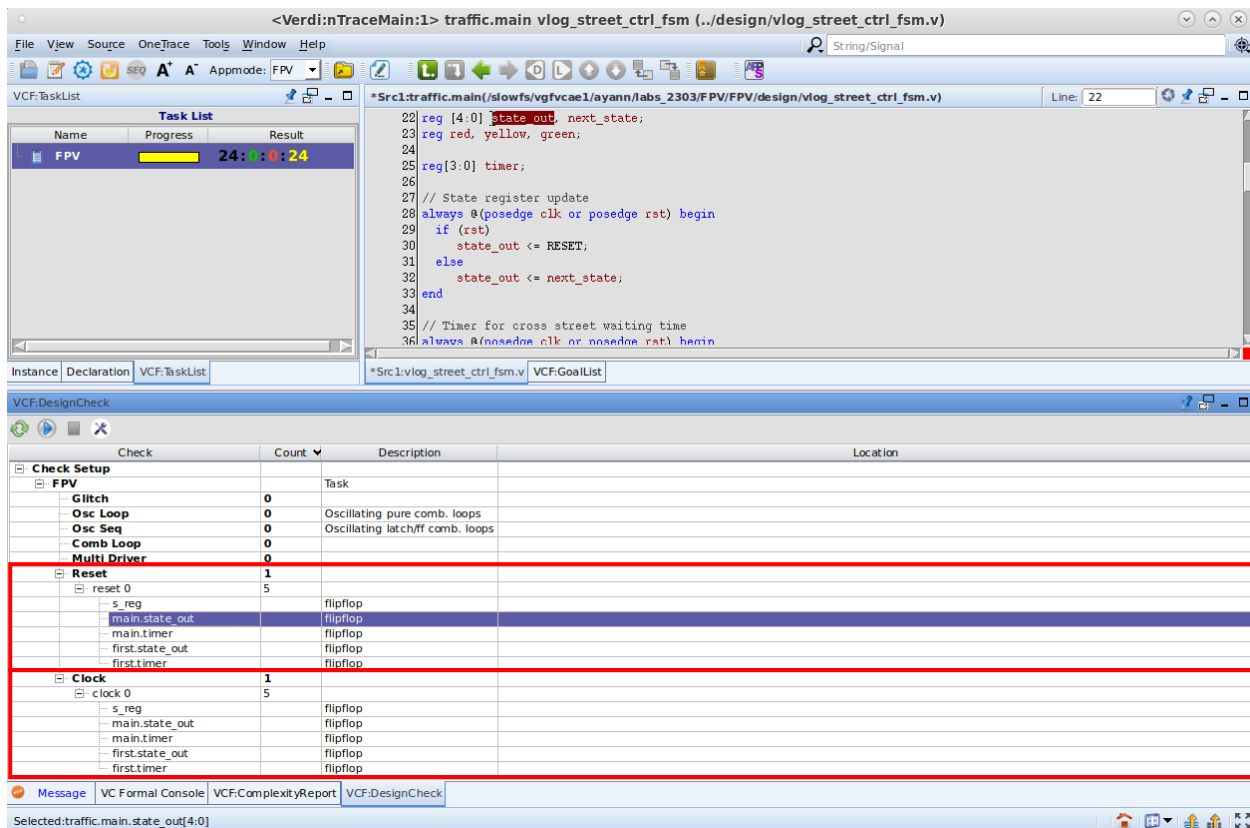

    Equivalent commands to run design checks and report results:
    ```
    check_fv_setup
    report_fv_setup
    ```

14. Analyze results for missing clock and reset information:

    Examine items under "Reset" and "Clock" and double-click on signals names to open the
    source code window to determine the clock sampling edge is posedge and the active phase
    of rst is high.

Revise Setup and Review Initial State

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

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

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

16. Add design initialization commands:

```
sim_run -stable
sim_save_reset
```

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

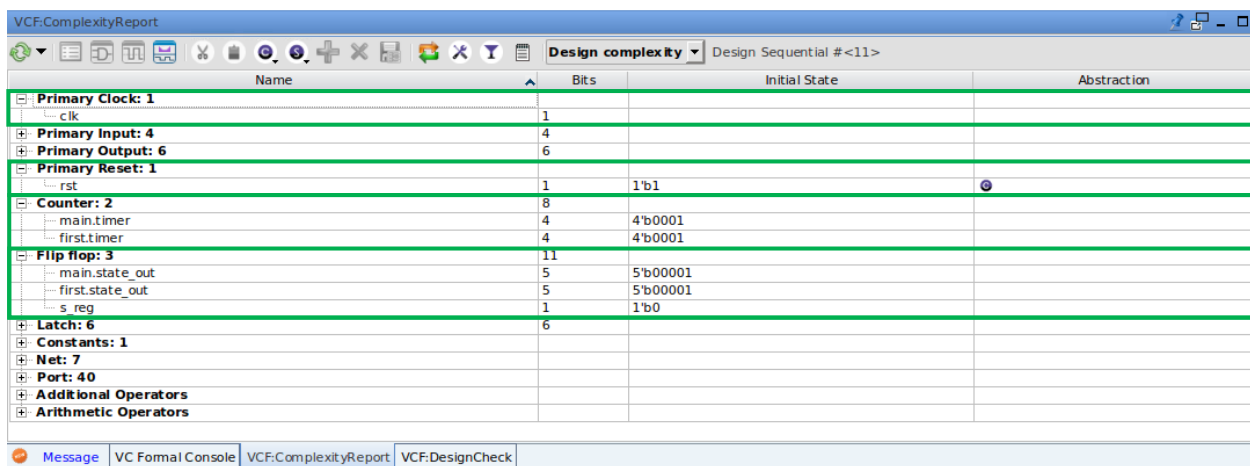17. Save edited Tcl file run.tcl:

Click on the Save icon .

18. Restart VC Formal:

Click on the Restart VCST icon ![icon].

19. Check updated setup and debug initial state:

Click on the Setup Assist icon ![icon] located above the property table and select Design Complexity. After clicking OK in the Complexity Report Setting dialog, check the Complexity Report window and confirm there are no more missing clock and reset shown.

Optionally, examine the Initial State for flip flops, latches, and other sequential elements such as counters to check if their initial value is as expected.

20. Exit VC Formal:

Click on File → Exit.

Run Formal Proofs and Review Results

21. 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
```
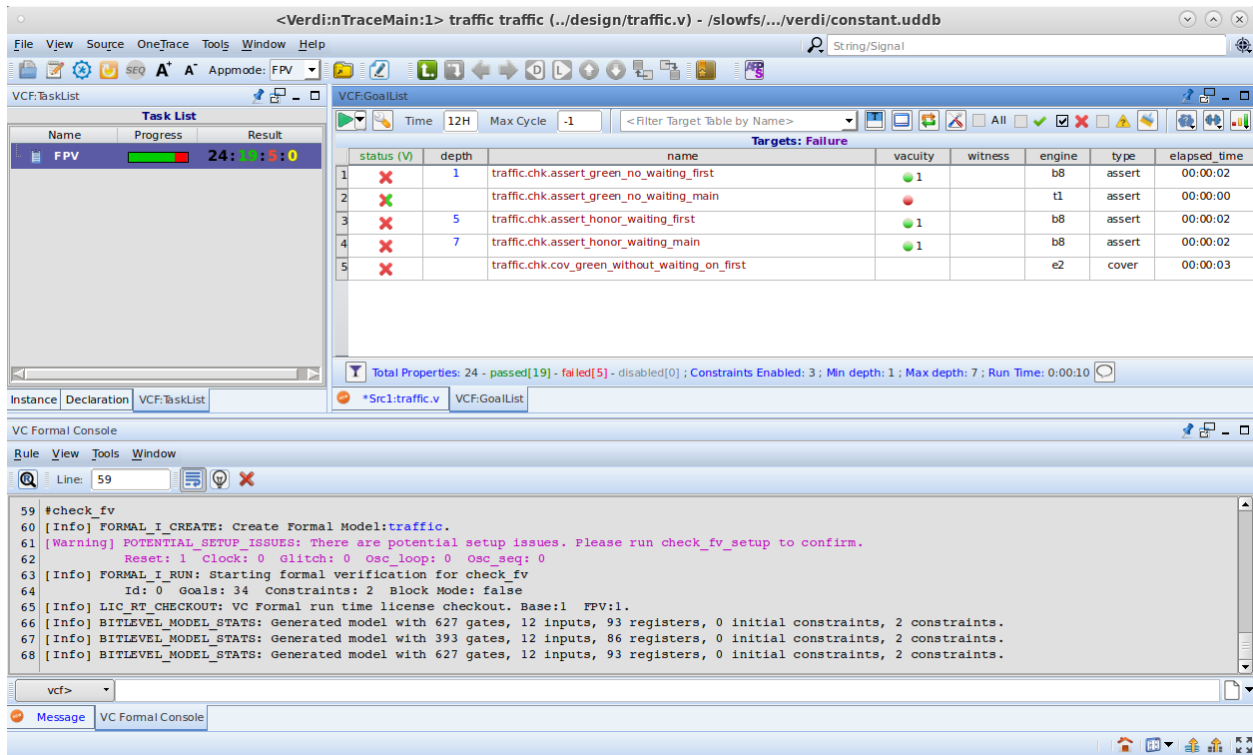
22. Start property verification:

Click on the Start Check icon ![icon] located above the property table.

23. Hide the constraints table:

Click on the Targets+Constraints icon ⊟ located above the property table.

24. Filter "Targets" table to keep failed targets:

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


Debug Failure

25. Debug failure:

Double click on ✖ under the status of property "assert_green_no_waiting_first" to open a counter-example waveform.
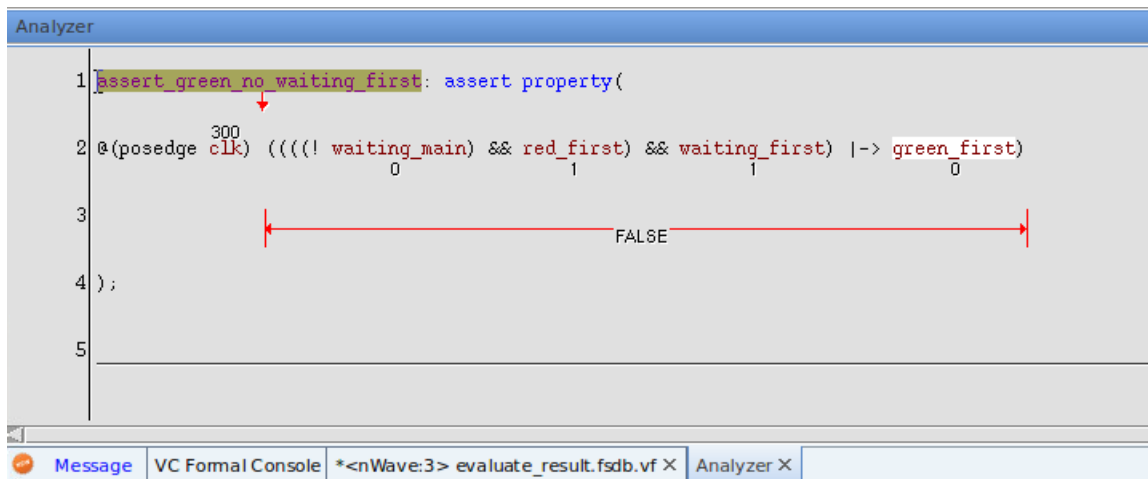(Note that a double click on the "name" of a property will open the property in the source code window instead.)

Equivalent command to open waveform for target property:
```
view_trace -property
  {traffic.chk.assert_green_no_waiting_first} -composite
```
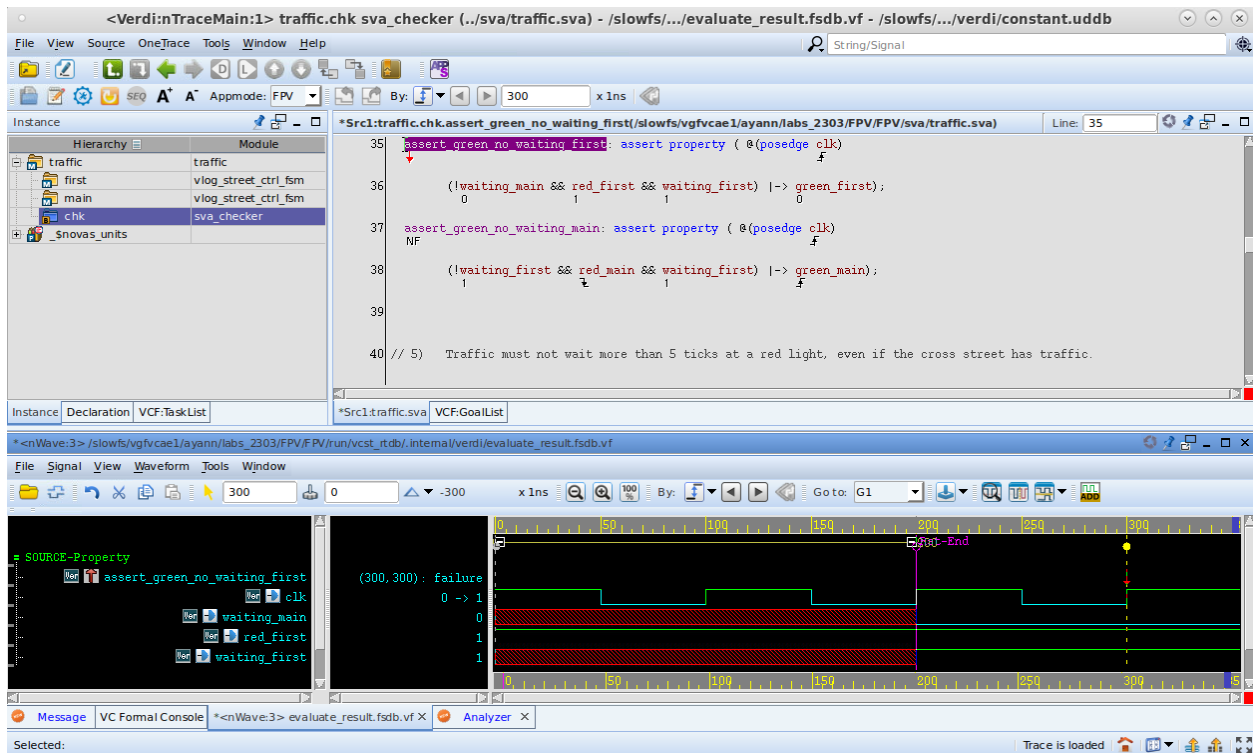
Click on "Analyzer" tab at the bottom to see the property expression.

The reason this property fails is because there is an error in the assertion. It is not possible to have red_first and green_first at the same time. The controller is supposed to wait 1 to 2 cycles to let the red light cycle through before issuing green_first.

Correct Erroneous Assertion

26. Modify assertion:

Click on "Edit Source File" icon and modify the assertion.

Original assertion:
```
   (!waiting_main && red_first && waiting_first) |-> green_first);
```

Modified assertion:
```
   (!waiting_main && green_main && red_first && waiting_first)
   |-> ##[1:2] green_first);
```

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 located above the property table.

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 assertions can be found in file: ../solution/traffic.sva.

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_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.

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 Tcl file run_batch.tcl 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 file.