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

Learning Objectives

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

- Run Property Density in GUI mode
- Run Over Constraint Analysis in GUI mode
- Run Formal Core in GUI mode
- Run Formal Testbench Analyzer GUI mode
- Run Property Density in batch mode
- Run Over Constraint Analysis in batch mode
- Run Formal Core in batch mode
- Run Formal Testbench Analyzer batch mode

Familiarity with the SystemVerilog Assertion (SVA) language and completion of FPV/FPV_General and FCA lab are required for this lab.

Lab Duration: 60 minutes

Files Location

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

Directory Structure	
FPV_Signoff	Lab main directory
README_VCFormal_FPV_Signoff.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

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. Add path $VC_STATIC_HOME/bin to the PATH environment variable.

3. Change your working directory to FPV_Signoff/run:

```
%cd FPV_Signoff/run
```

Setup File

The VC Formal signoff flow has been developed to allow for measurement of quantifiable metrics on the quality of an FPV environment. The signoff flow is a set of steps which provide different pieces of information and levels of confidence in the quality of a formal environment.

There are five different aspects to the signoff flow:

- Property Density
- Overconstraint Analysis
- Bounded Analysis
- Formal Core
- FTA

In this lab will go through each one of these signoff steps. Each step has individual scripts to be loaded. All the scripts are pre-populated for you to review and run.

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

Step 1 - Property Density

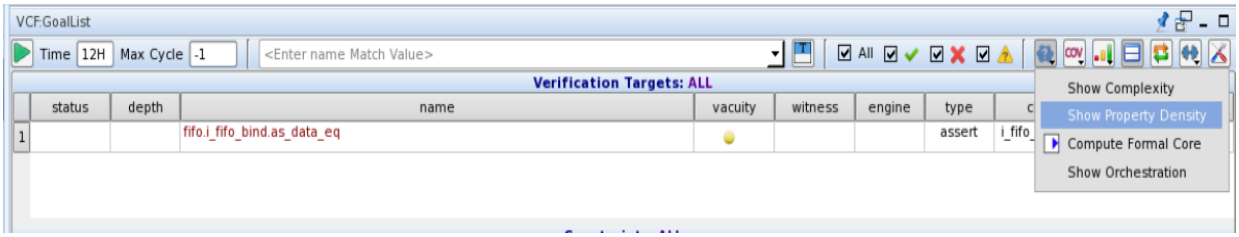
1. Property density performs a structural analysis of the design to determine the Cone of Influence (COI) of the assertions. PD is a check to see if enough properties are present in the testbench. To check the progress of development of assertions, go to work directory and execute run_pd

```
% ./run_pd
```

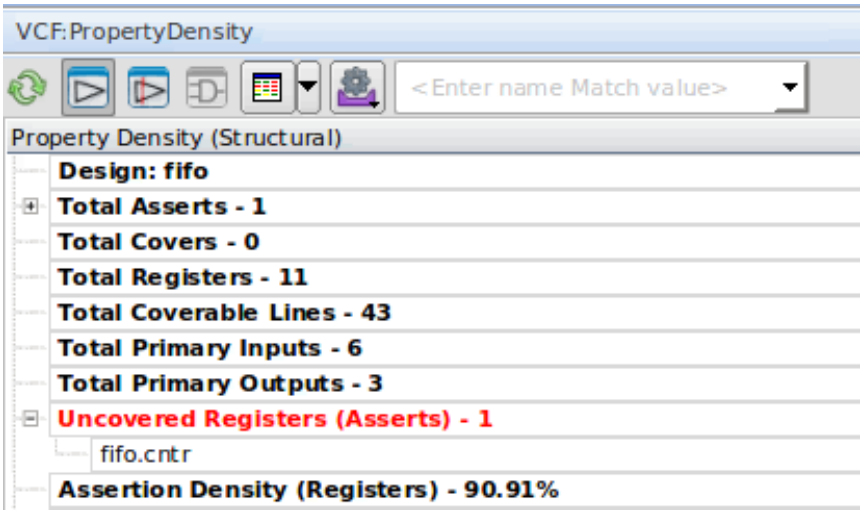
2. This opens VCF Verdi GUI and loads the design with a single assertion:

```
as_data_eq: assert property (`clk_rst $rose(first_pop) |->
data_out_nxt == data_out);
```

3. Generate Property Density report: go to the GoalList window and click on the Show Complexity  icon and choose Show Property Density to generate Property Density report:

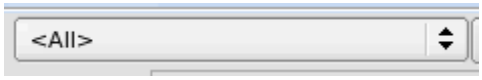


4. Identify "Potential Verification Holes": Expand the Uncovered Registers (asserts) option. Find the register (fifo.cntr) that is outside the COI of all the properties for this run.

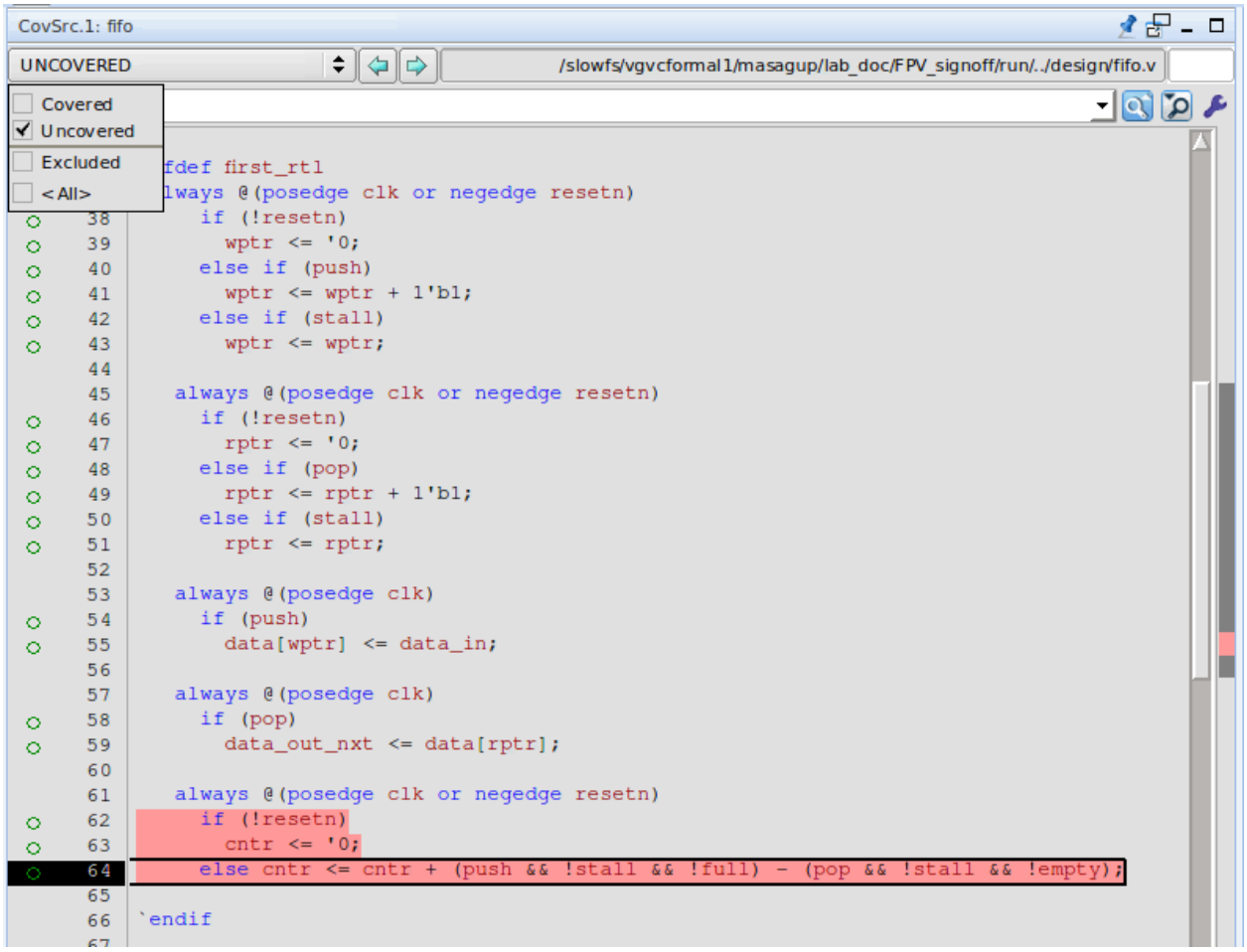


5. To view the property density in coverage, click on the icon  in the Property Density window. Verdi Coverage GUI will pop up. This will show the regions of the RTL that are not covered.

6. To view only the Uncovered RTL, click on the drop-down option at



and select Uncovered.



Increasing Property Density:

7. Go to the work directory and execute

```
% ./run_pd_oc
```

8. A second assertion has been added to close the Verification hole for “cntr” register.

9. Re-generate Property Density reports (steps 3 to 6)

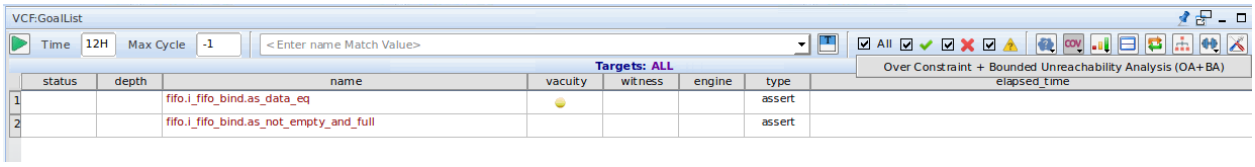
10. Observation: Is all of RTL covered?

Step 2 – Over Constraint Analysis

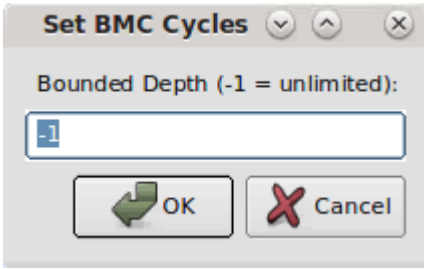
11. Over constraint analysis is designed to ensure that there are no constraints in the design preventing legal areas of code from being exercised. To run Over constraint analysis on this design, go to the work directory and execute

```
% ./run_pd_oc
```

12. Run Over Constraint analysis: In the GoalList window click on the icon Coverage Analysis icon  and select Over Constraint and Bounded Unreachability Analysis

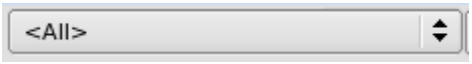


Set BMC Cycles to -1 for over constraint analysis

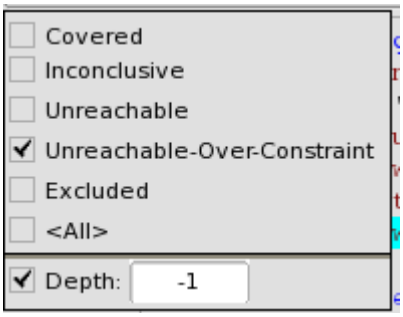


Note: To run bounded analysis, specify the depth for the analysis in the "Set BMC Cycle" field above.

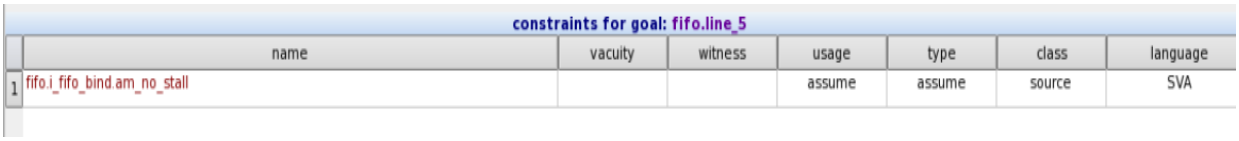
13. Verdi Coverage GUI will pop-up. Manually browse color coded source code to identify RTL that cannot be reached due to constraints.
Or in the source code window select on the drop-down option

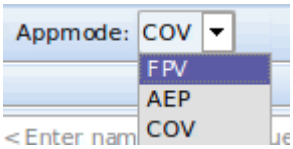


and select only Unreachable-Over-Constraint option

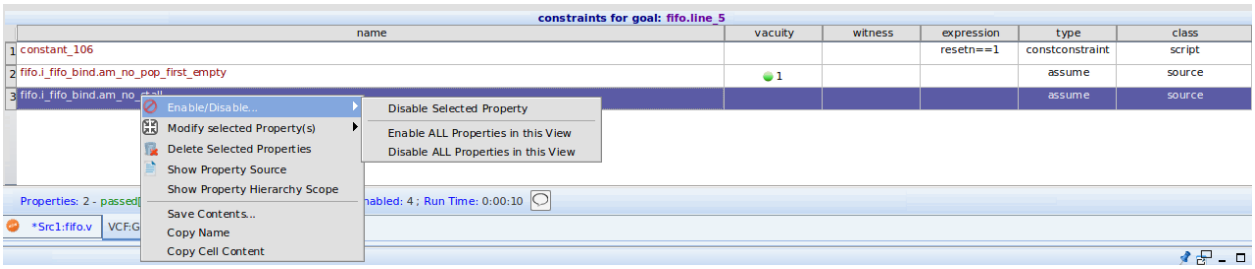


14. Identify the constraint: In Verdi Coverage GUI, select the unreachable line and click on the Show Reduced Constraints icon  in the Menu Bar. The computation depicts responsible constraints on VCF.GoalList Constraints Pane in VC Formal Verdi GUI



15. In VC Formal Verdi GUI, change the appmode to FPV 

. Disable the constraint: select the constraint and RMB, and disable the constraint



Re-run the proofs and observe that the constraint was causing the unreachability.

Fixing the Constraints:

16. To fix the over constraining issue, the constraint has been removed. To run the fix, go to the work directory and execute

```
%./run_pd_oc_fc
```

17. Run the proofs – the goals should both be proven

18. Run over constraint analysis should show no over constraining (steps 12-14)

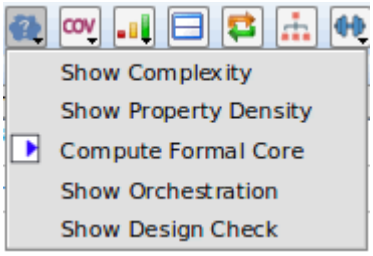
Step 3 – Formal Core

19. Formal core is a more accurate measure of which areas of the design have been involved in the proof (or bounded proof) of a property. The formal core will be a subset of the COI and is more accurate than property density. To run Formal Core, go to the work directory and execute

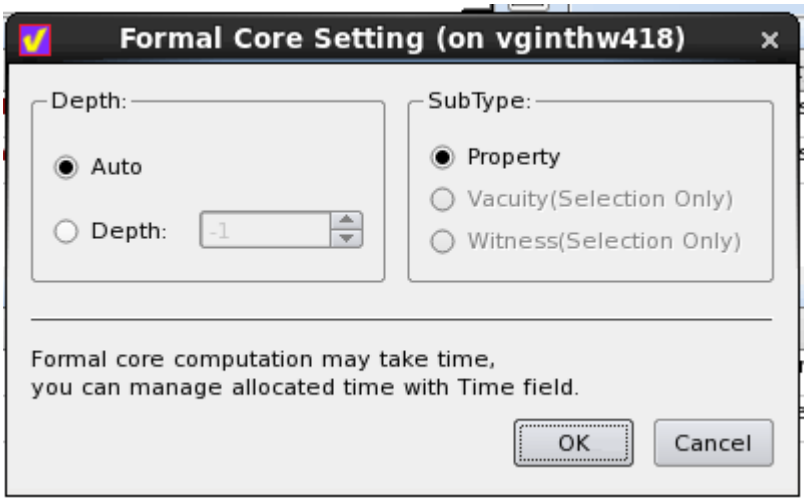
```
%./run_pd_oc_fc
```

20. Since Formal Core results are for Proven (or bounded proven) property, first run the FPV. This can be done using  in the task bar of GoalList window or by using command "check_fv" in the Console.

21. Run Formal Core analysis: in the GoalList window click on the the Show Complexity  icon and choose Compute Formal Core.

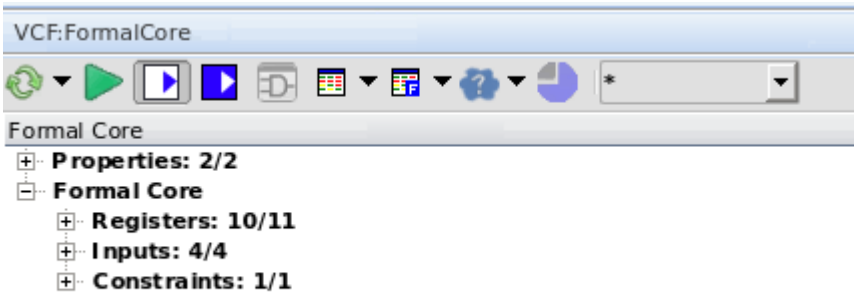


And click OK for the Formal Core Setting

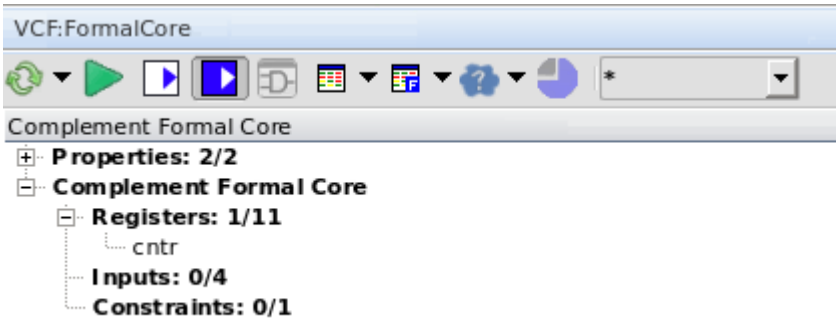


Formal Core window will pop-up

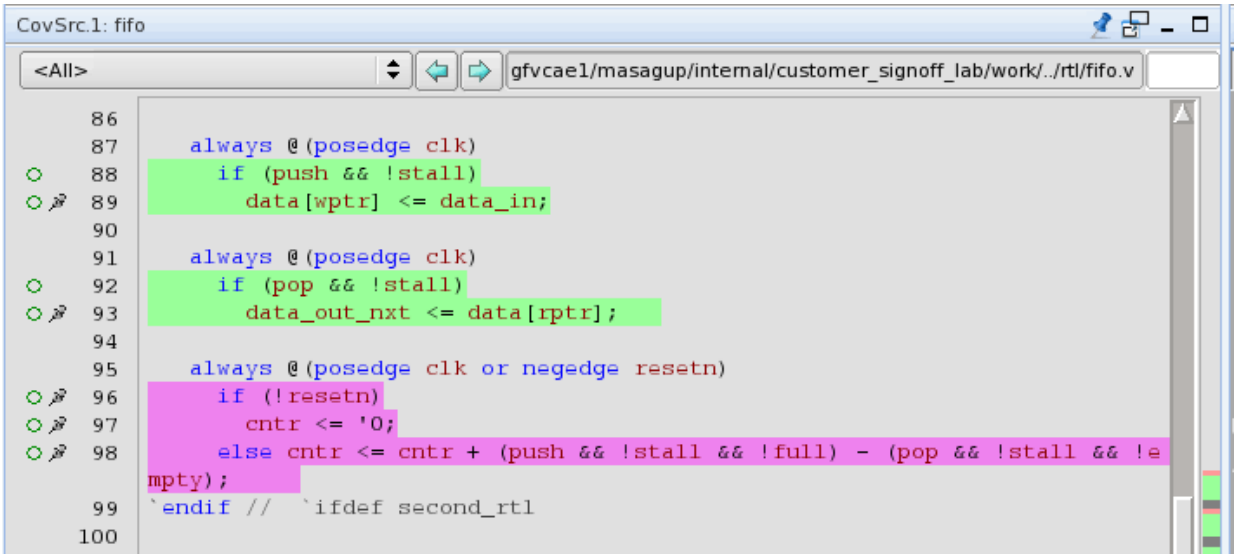
22. The Formal Core report provides a summary of the:
    The number of properties the compute_formal_core command is working/worked on
    A set of registers in the design used by the formal engines to prove the property
    A set of constraints used to prove the property.
    A set of inputs involved to prove the property.



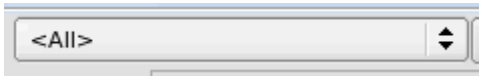
To see the complement of Formal Core, select icon  on the Formal Core window taskbar. This option provides, a collection of registers, inputs, constraints that are not present in the formal core is reported.



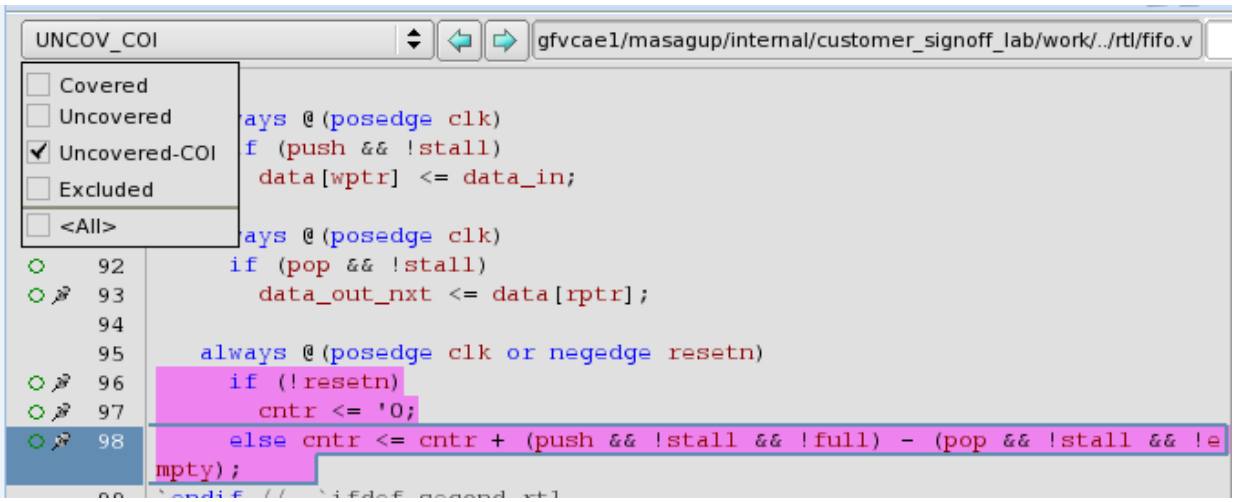
23. To view Formal core in coverage, click on the icon  in Formal Core window.
    Verdi Coverage GUI will pop-up showcasing the RTL coverage. Browse color coded source code to identify if RTL coverage is 100%. Observe that “cntr” is in the Property Density report [COI]. It is outside of the formal core [Uncovered-COI]. This is still a verification hole.



The Uncovered-COI RTL can only be selected by using the drop-down option at



and selecting Uncovered-COI.



24. To run reachability test, click on view Formal analysis coverage from Formal Core task bar, the icon  . Please note that there will be performance hit when Formal coverage analysis is run.

Increasing the Formal Core Score:

25. To increase the formal core score, an additional assertion has been added
    Go to the work directory and execute

```
% ./run_pd_oc_fc_fta
```

26. Run the assertions and check the formal core, it should now be at 100% (steps 20 to 23)

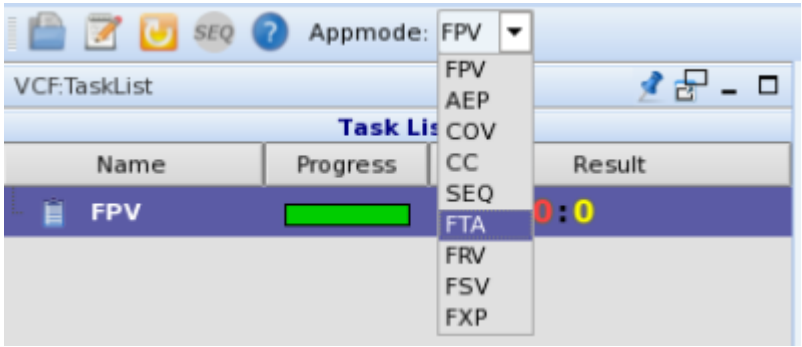
Step 4 – Formal Testbench Analyzer

27. The FTA app has been developed to perform fault injection analysis on a design to check whether the formal environment can catch injected issues. If a property fails in the presence of an artificial fault, then this is a good sign. To run FTA, go to the work directory and execute

```
% ./run_pd_oc_fc_fta
```

Run the properties using check_fv command or  icon on the task bar

28. Change the Appmode to FTA using the Appmode drop-down option



Run FTA using the Start Check icon 

This will generate Fault summary. Although Formal Core was at 100% there are still 22 non-detected faults representing verification holes

Fault Summary					Constraints: ALL		
Class Name	Faults in Design	Faults in Report	Non-Activated	Detected	Non-Detected	Disabled By User	Not Yet Qualified
All Fault Classes(9)	**64**	**64**	**0**	**42**	**22**	**0**	**0**
TopOutputsConnectivity	9	9	0	9	0	0	0
ResetConditionTrue	3	3	0	1	2	0	0
SynchronousControlFlow	20	20	0	10	10	0	0
InternalConnectivity	0	0	0	0	0	0	0
SynchronousDeadAssign	2	2	0	1	1	0	0
ComboLogicControlFlow	0	0	0	0	0	0	0
SynchronousLogic	25	25	0	18	7	0	0
ComboLogic	2	2	0	2	0	0	0
OtherFaults	3	3	0	1	2	0	0

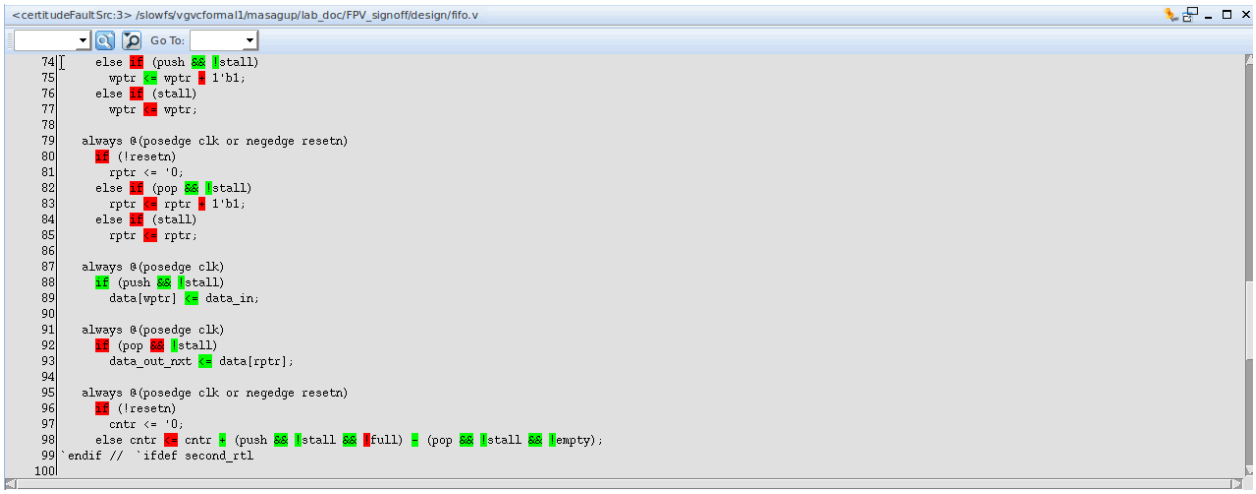
The severity of the faults is reported in the order of highest to lowest in the rows. The non-detected faults are of interest here as the current properties cannot detect these faults.

29. Select only Failures from the GoalList taskbar 

.

Targets: Failure Filter by status					FPV Task Properties							
	status (V)	name	elapsed_time	location	mutated_code	fault_type	fault_class	Prop ID	signals	waived	fault_cluster_id	fault_inst
1	✗	fifo.fault_id_19		...esign/fifo.v:74	...de removed */	ElseDead	...usControlFlow	56		false		
2	✗	fifo.fault_id_23	00:00:03	...esign/fifo.v:75	-	Operator	...chronousLogic	61		false		
3	✗	fifo.fault_id_24		...esign/fifo.v:76	1'b0	ConditionFalse	...usControlFlow	62		false		
4	✗	fifo.fault_id_25		...esign/fifo.v:76	1'b1	ConditionTrue	...usControlFlow	63		false		
5	✗	fifo.fault_id_26		...esign/fifo.v:76	!(stall)	...atedCondition	...usControlFlow	64		false		
6	✗	fifo.fault_id_27		...esign/fifo.v:77	...de removed */	DeadAssign	...chronousLogic	65		false		
7	✗	fifo.fault_id_29		...esign/fifo.v:80	1'b1	ConditionTrue	...tConditionTrue	66		false		

Double click on any fault id to go into the source browser



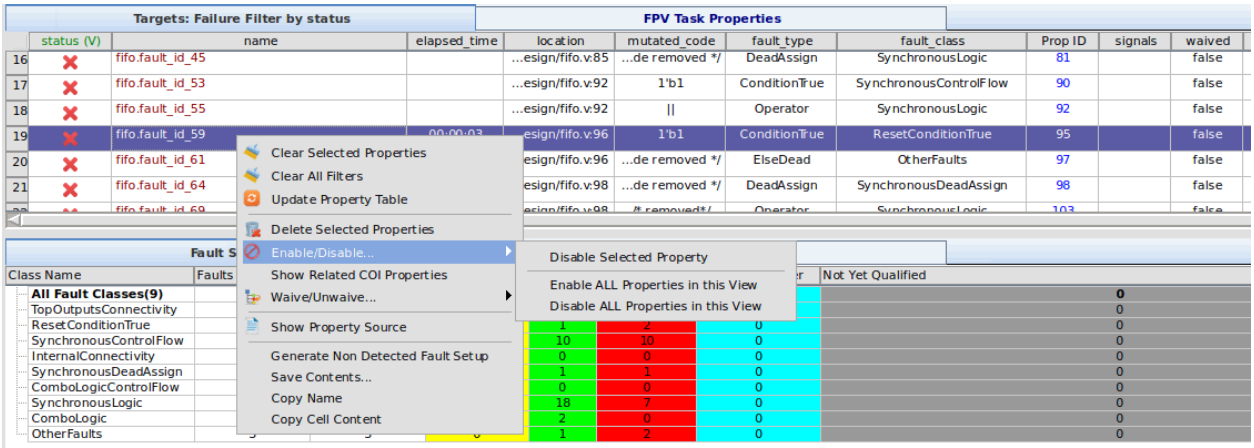
Red indicates that a fault is non-detected. If you mouse hover over the red code it will show you the non-detected faults

30. To map the FTA results to coverage, click on the  icon. Verdi coverage GUI will pop up.

```
87    always @(posedge clk)
88      if (push && !stall)
89        data[wptr] <= data_in;
90
91    always @(posedge clk)
92      if (pop && !stall)
93        data_out_nxt <= data[rptr];
94
95    always @(posedge clk or negedge resetn)
96      if (!resetn)
97        cntr <= '0;
98      else cntr <= cntr + (push && !stall && !full)
    - (pop && !stall && !empty);
99  `endif //  `ifdef second_rtl
100
```

Waiving the faults:

31. For the Formal analysis, the reset doesn't toggle, and we can disable it.
    To disable the fault, RMB of the fault property → Enable/Disable →Disable Selected Property



Step 5 – Finish Signoff

32. Additional assertions have been added, to decrease non-detected faults. To run these, go to the work directory and execute

```
% ./run_signoff
```

    you should now see only two non-detected faults remain

33. See if you can figure out why two non-detected faults remain!

Mode 2: Start VC Formal Signoff in Batch Mode

Step 1 – Property Density

34. Go to the work directory and execute

```
% ./run_pd_batch
```

The script generates a vdb (PD.vdb) file for Property Density Coverage

35. Open pd_uncovered_report.txt to view results
Results show that "cntr" register is outside of the COI of properties

36. To view Property Density Coverage in GUI

```
% $VC_STATIC_HOME/verdi/bin/verdi -cov -covdir PD.vdb
```

Increasing Property Density:

37. A second assertion has been added to close the Verification hole for "cntr" register, to run this, go to the work directory and execute

```
% ./run_pd_oc_batch
```

Step 2 – Over Constraint Analysis

38. Go to the work directory and execute

```
% ./run_pd_oc_batch
```

Script runs over-constraint analysis and dumps a vdb (FPV_OA.vdb) and exclusion file (FPV_OA_unr.el_w_constraints.el)

39. To view Over-Constraint Coverage analysis in GUI:

```
% $VC_STATIC_HOME/verdi/bin/verdi -cov -covdir FPV_OA.vdb -
elfile FPV_OA_unr.el_w_constraints.el
```

Step 3 – Formal Core

40. Go to the work directory and execute

```
% ./run_pd_oc_fc_batch
```

A formal core report of any registers outside the formal core will be generated. Review outside_formal_core.txt to see results. It shows that the cntr register is outside the formal core

41. To view Formal Core Coverage analysis in GUI:

```
$VC_STATIC_HOME/verdi/bin/verdi -cov -covdir FPV_FC.vdb
```

Step 4 – Formal Testbench Analyzer

42. Go to the work directory and execute

```
% ./run_pd_oc_fc_fta_batch
```

Review non_detected_faults.txt – This shows that 22 faults were non-detected even after formal core is at 100%

Step 5 – Finish Signoff

43. Additional assertions have been added, to decrease non-detected faults. To run these, go to the work directory and execute

```
% ./run_signoff_batch
```

you should now see only two non-detected faults remain

See if you can figure out why two non-detected faults remain!