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

Learning Objectives

In this lab you will be using FIFO design and learning to do the following:

Step 1: FPV

- Read the spec provided in this document
- Write SVA as per suggestions in the provided checker file
- Setup environment for FPV and verify the design using assertions you have created
- Determine root cause and fix CEXs if any

Step 2: Formal Signoff Dashboard

- Re-visit your FPV setup and instrument Coverage & Faults for analysis
- Re-run FPV and invoke dashboard interface
- Execute effort low
- Execute effort medium
- Execute effort high
- Save and Restore session

Lab Duration: 120 minutes

Familiarity with the SystemVerilog Assertion (SVA) language and completion of FPV/FPV_General 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_Signoff_Dashboard

Directory Structure	
FPV_Signoff_Dashboard	Lab main directory
README_VCFormal_FPV_Signoff_Dashboard.pdf	Lab instructions
design/	Verilog RTL code of the Device Under Test
sva/	SVA properties to check functionality of
run/	Run directory
solution/	Solution directory

Resources

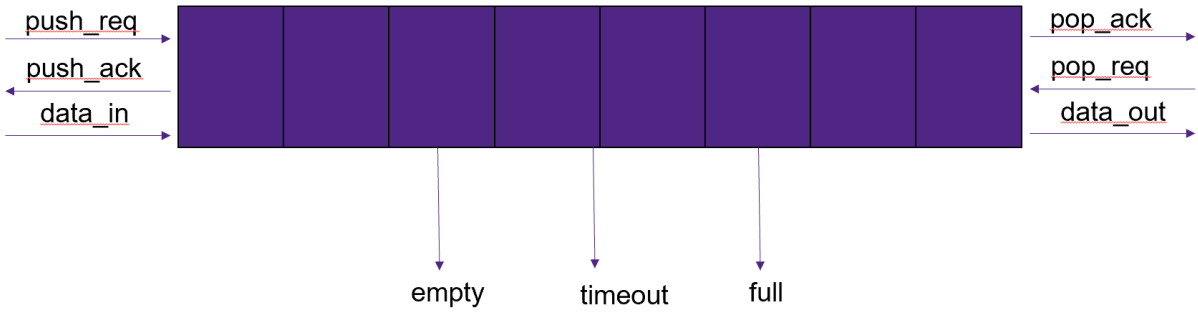
The following resources are available for in-depth guidance regarding VC Formal usage, commands, and variables.

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/

First In First Out (FIFO)



FIFO SPEC

• Design should write data and read data in a First In First Out order and handle up to 16 transfers

• When the design has no more free space the full flag should be raised

• When the full flag is high, push_ack must be low

• When push_req is high and push_ack low, push_req must be kept high and data_in stable

• When the design has no data inside the empty flag should be raised

• When the empty flag is high, **pop_ack** must be low

• When **pop_req** is high and **pop_ack** low, **pop_req** must be kept high

• Data comes out of the pop interface 1 cycle after **pop_req** and **pop_ack** are high together.

• When a gap of 5 or more cycles appears between **push_req**s timeout should be set

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_Dashboard/run:

```
%cd FPV_Signoff_Dashboard/run
```

4. For this signoff dashboard, it is beta feature and need set some environment variables

```
%source ../sva/env.csh
```

Now you are ready to begin the lab.

Create FIFO Checker from Spec

5. Open the "../sva/fifo_sva.sv" file. There are five questions embedded as comments, e.g.

```
// Q1. Complete the assumes described in the labels below
//
// In absence of push_ack, data in and push request
// is held stable
am_push_req_stable_when_no_ack:
assume property (`clk_rst (push_req && !push_ack) |=>
am_data_in_stable_when_no_ack:
assume property (`clk_rst
// If pop_req is asserted and pop_ack is low,
// pop_req should be held stable
am_pop_req_stable_when_no_ack:
assume property (`clk_rst
```

Complete questions Q1, Q2 & Q3 in the file "fifo_sva.sv" by writing assertion and assume expressions as **instructed in the comments above**.

Verify Design using VC Formal

6. Verify the FIFO design using the checker you have written.

```
% vcf -f fifo.tcl -gui &
```

Fix any compilation warnings and errors in your checker and run proof of all assertions in your checker. Debug and fix any CEX/Failures in the design and checkers you have written.

Check Data Integrity of FIFO

7. Now try Part B of the lab. This has questions Q4 & Q5 to complete. Pass the "+define+LAB_PART_B" option to the "read_file" command.

```
read_file -top $top -format sverilog -sva \
-vcs {-f ../design/filelist.flist +define+LAB_PART_B}
```

Fix any compilation warnings and errors in your checker and run proof of all assertions in your checker. Debug and fix any CEX/Failures in the design and checkers you have written.

Setup Design for Coverage and FTA

8. Source options/settings required for coverage generation. Add the below command before "read_file" command in "fifo.tcl".

```
source ../sva/cov_options.tcl
```

9. Pass the signoff configuration for "all" which defines some types of coverage and faults in "fifo.tcl" before the "read_file" command.

```
signoff_config -type all
```

For coverage, it can support in "line", "cond", "toggle", "branch" and "cg".
For fault types, it can support in "fault_rtl" and "fault_conn" which means RTL and Connecticity in faults.

```
signoff_config -type "line cond toggle branch cg fault_rtl fault_conn"
```

Configure Fault Injection for FTA

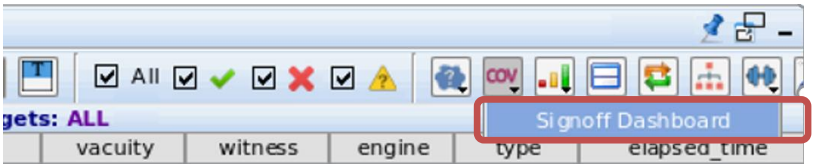
10. FTA uses Synopsys Certitude to instrument faults in the design. You can specify modules where faults need to be injected by using the "fta_init" command.

    Add the below command before "read_file" command.

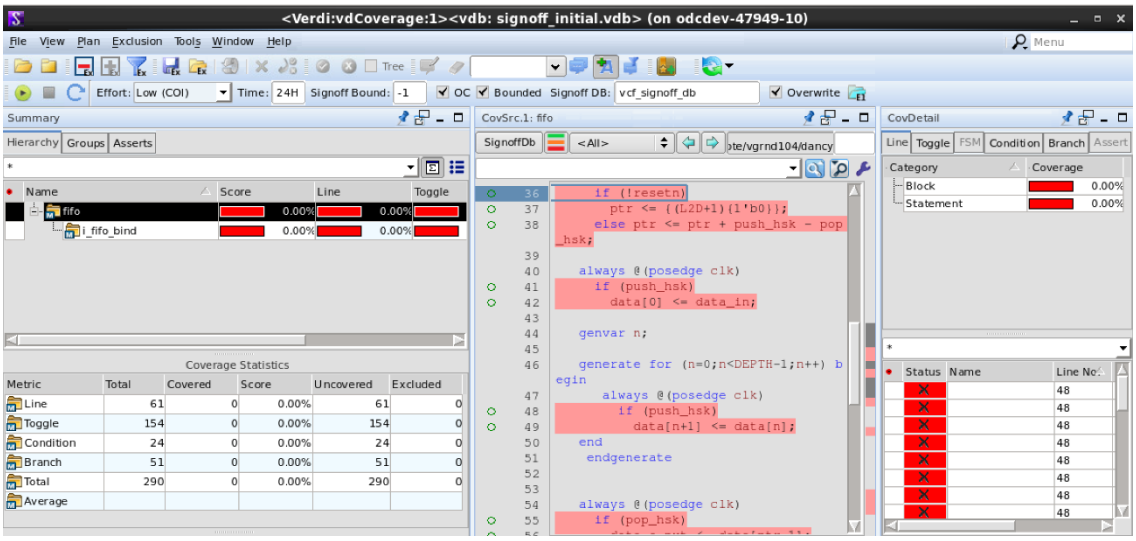
```
fta_init -scope {fifo}
```

Formal Signoff – Dashboard invoke

11. Restart tcl file and rerun the design by click this icon  .
12. After the tool finish running, invoke Formal Signoff Dashboard by click COV => Signoff Dashboard :



13. The Dashboard interface appears in below with initial vdb loaded for 0% coverage.



Formal Signoff Dashboard – Interface Introduction

14. Dashboard icon list

 Compute Signoff : Start signoff validation

 Stop Compute Signoff : Stop signoff validation by ctrl-c, the database will not be loaded when interrupt

Reload Database

Effort: Low (COI) Signoff effort : There has 3 efforts which can choose for run, low effort for COI, medium effort for FC and High effort for FC plus FTA. The default is low effort.

Time: 24H Max time : The timeout setting for signoff flow, default is 24Hrs

Signoff Bound: -1 Signoff Bound : bound to compute signoff metrics, default -1(unlimited)

OC Over Constraint invoke : default turn on

Bounded Bounded results for FC and FTA, default turn on

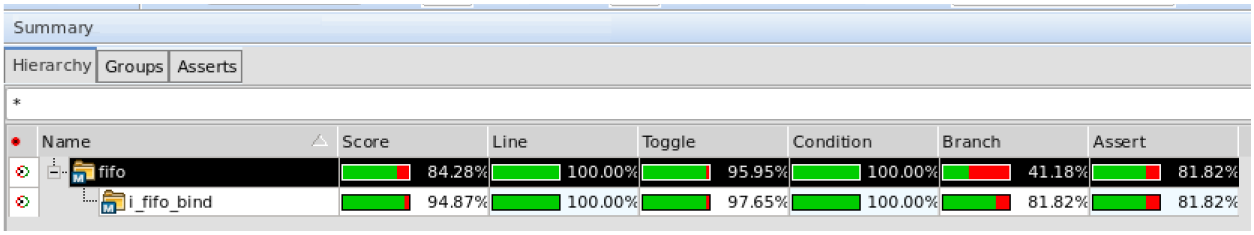
Signoff DB: vcf_signoff_db Signoff DB : Signoff database save location, default is vcf_signoff_db

Overwrite Overwrite : Re-calculate and overwrite the database at each stage, default on

Exclusion files loaded : Load exclusions before execute signoff effort stage, and it will be excluded after that effort

15. Summary

It presents the design coverage score which is enable for the coverage type in hierarchical



16. Coverage Statistics

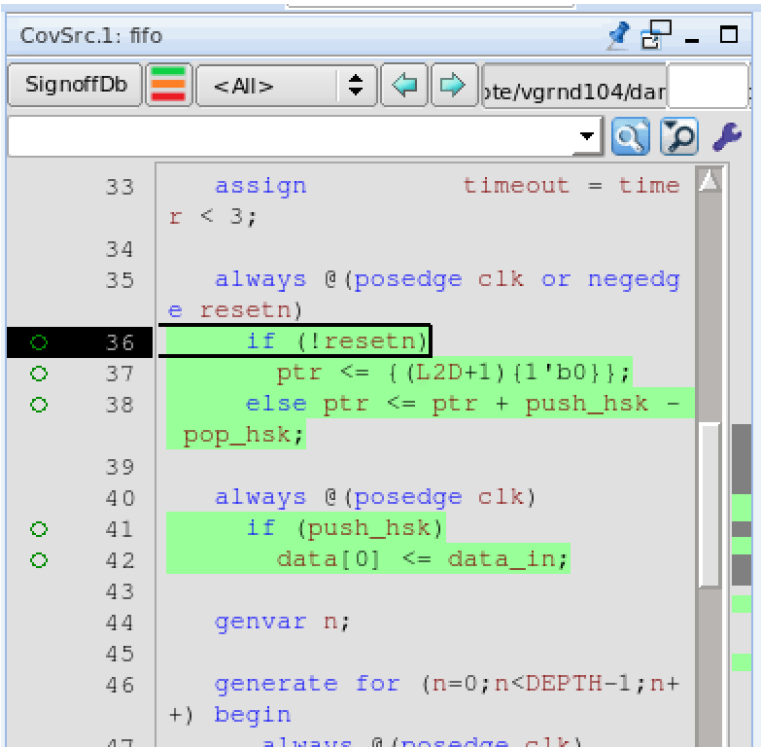
It presents the metric score which is covered/uncovered for all enabled coverage type by current database.
It also includes excluded part.

Coverage Statistics (OC.vdb)

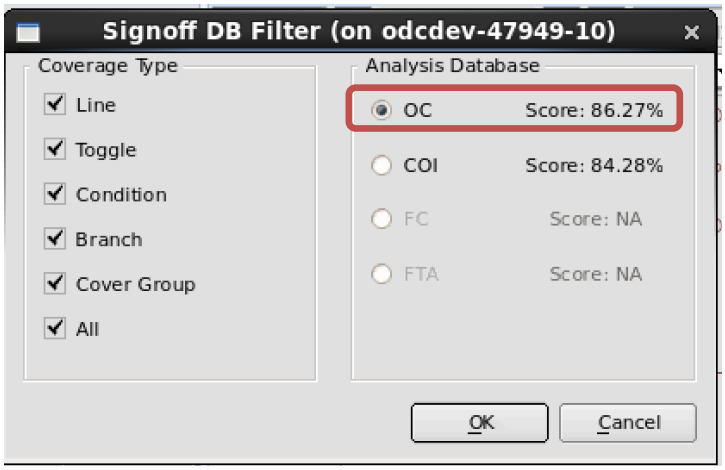
Metric	Total	Covered	Score	Uncovered	Excluded
Line	61	61	100.00%	0	0
Toggle	148	148	100.00%	0	6
Condition	18	18	100.00%	0	6
Branch	51	23	45.10%	28	0
Total	278	250	89.93%	28	12
Average			86.27%		

17. Coverage Source (CovSrc)

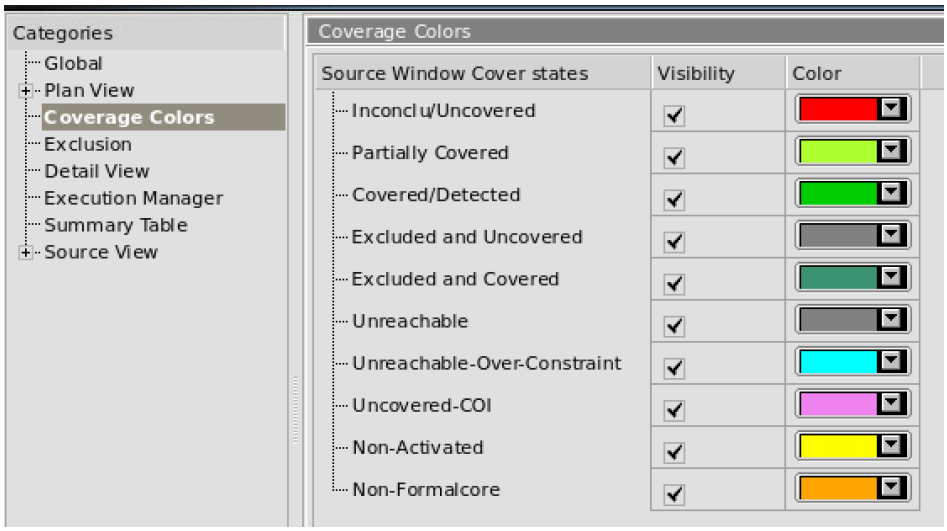
It presents the source code for coverage.



Signoff database switcher : Can choose which database to load when done and which coverage type to present.



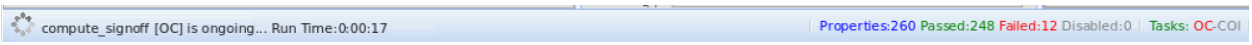

Coverage colors : To change the color for default setting.



Formal Signoff Dashboard – Low effort

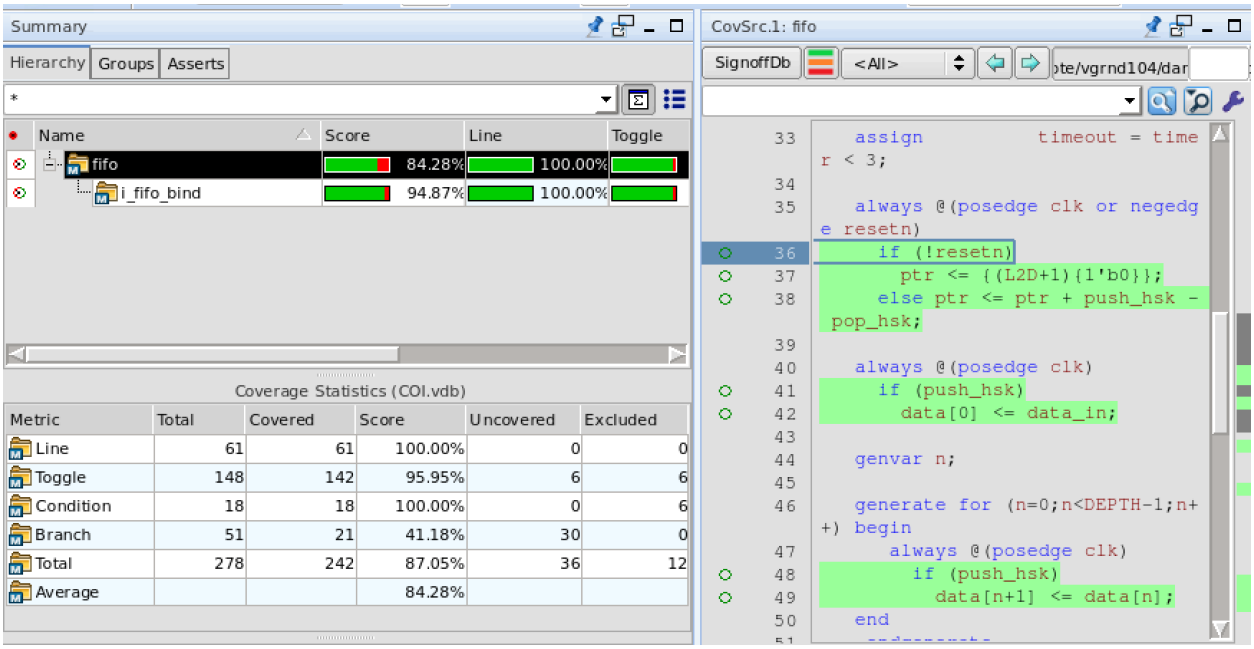
18. Choose effort “low” (default) and pass “Compute Signoff” icon.

19. Status will show in the bottom message ribbon, for the ongoing task will show it’s run time, property results. And for the Tasks : OC-COI which is the flow for low effort.



Red task : Ongoing task
Grey task : Not-go task
Green task : Done task

20. When done, it will pop out one window for load database.

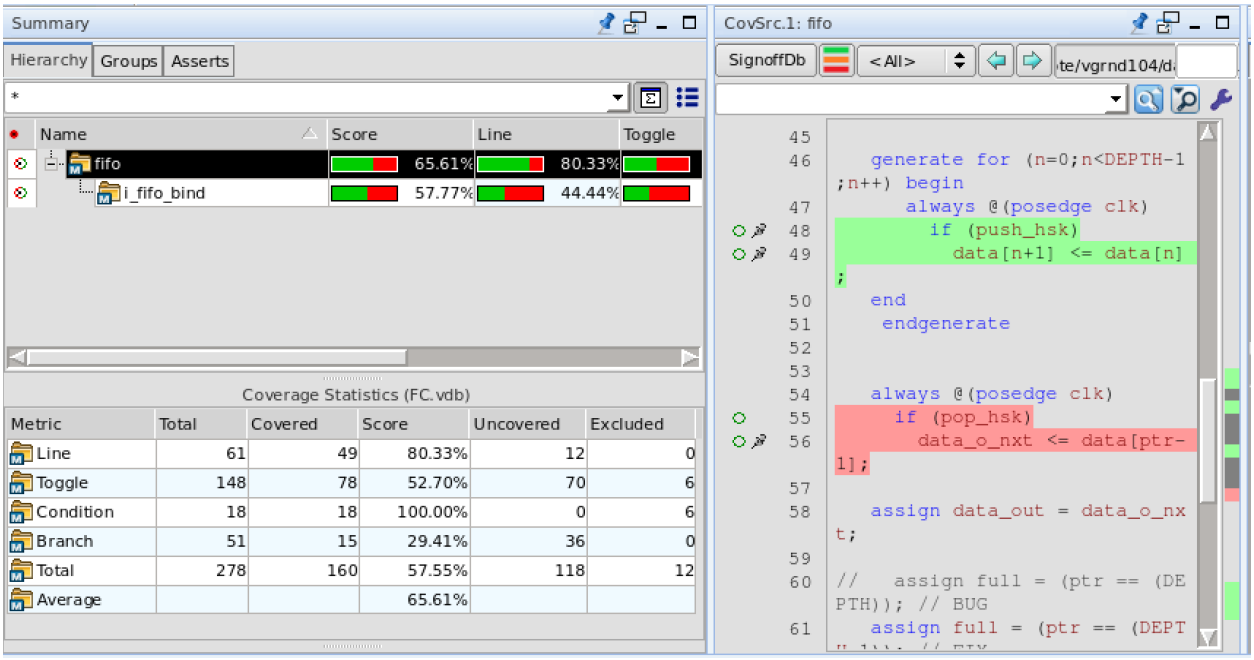


Formal Signoff Dashboard – Medium effort

21. Choose effort “Med” and pass “Compute Signoff” icon.

22. When done, you can see code coverage that is in the Formal Core of the assertions that have been proven.

    Green → In the Formal Core
    Red → Outside the Formal Core



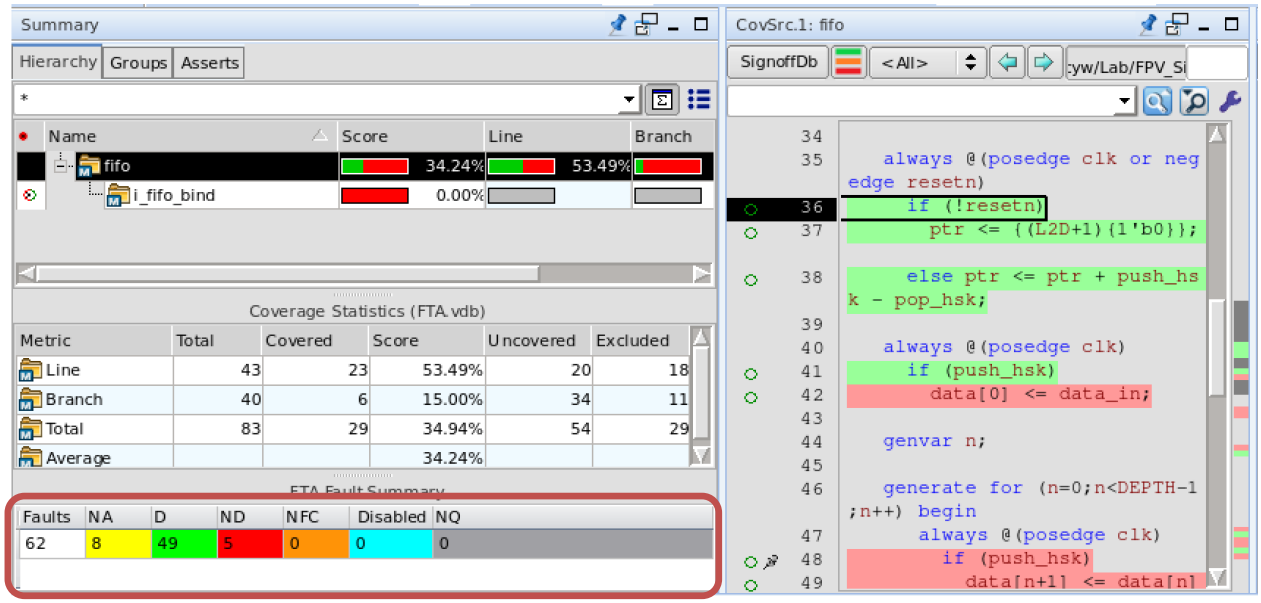
Code coverage outside the Formal Core means a portion of the code is not being tested. Make sure this is justified. If not, please add more assertions to achieve 100% Formal Core coverage.

Formal Signoff – High effort

23. Choose effort “High” and pass “Compute Signoff” icon.

24. When done, you can see FTA coverage result.

    The difference from other efforts is tool will show FTA Fault Summary under coverage statistics.

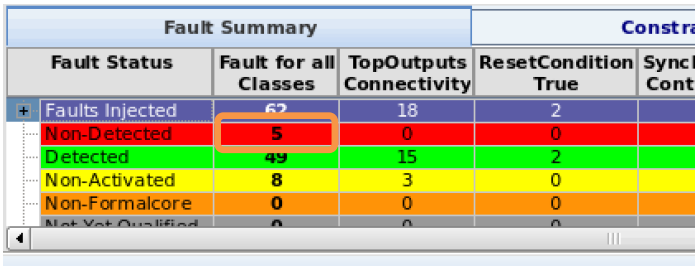


25. Once the FTA run is finished, debug Non-Activated and Non-Detected assertions.

    Non-Activated (NA) : No assertion is checking this code. Outside COI of all assertions.
    Non-Detected (ND) → In the COI of one of more assertion, but definition of the assertions is **inadequate** to catch a fault/bug introduced. Modify assertion or enhance testbench with more assertions

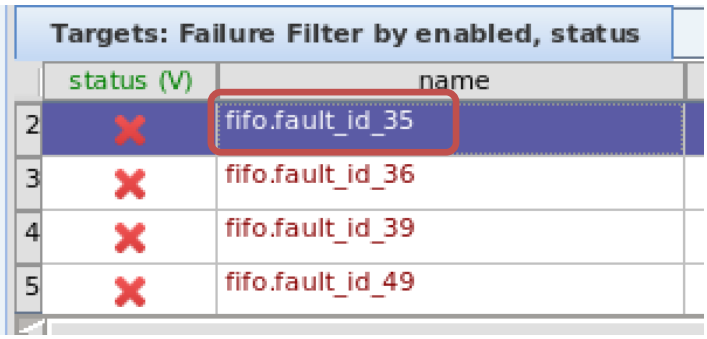
26. Debug the ND and NA faults

Change to original GoalList/TaskList window and double click on individual Non-Activated and Non-Detected faults to launch FTA Fault View. This should help you understand the fault that has been introduced and the code that is not being tested.



Fault Summary				Constra
Fault Status	Fault for all Classes	TopOutputs Connectivity	ResetCondition True	Sync Cont
Faults Injected	62	18	2	
Non-Detected	5	0	0	
Detected	49	15	2	
Non-Activated	8	3	0	
Non-Formalcore	0	0	0	

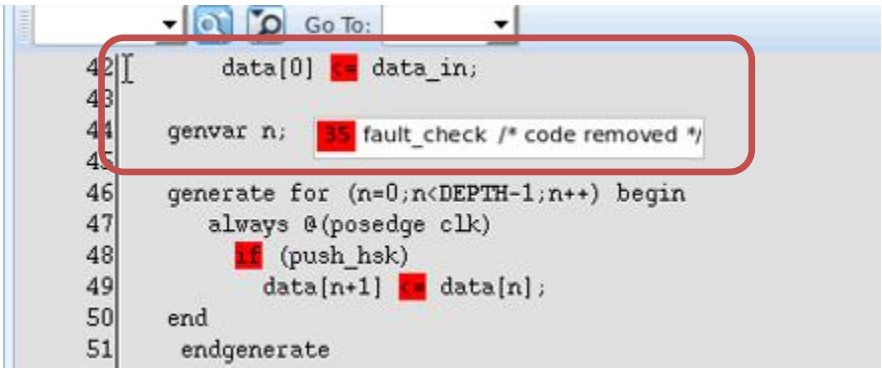
Double click fault name :



Targets: Failure Filter by enabled, status

	status (V)	name
2	✗	fifo.fault_id_35
3	✗	fifo.fault_id_36
4	✗	fifo.fault_id_39
5	✗	fifo.fault_id_49

The sync fault source view :

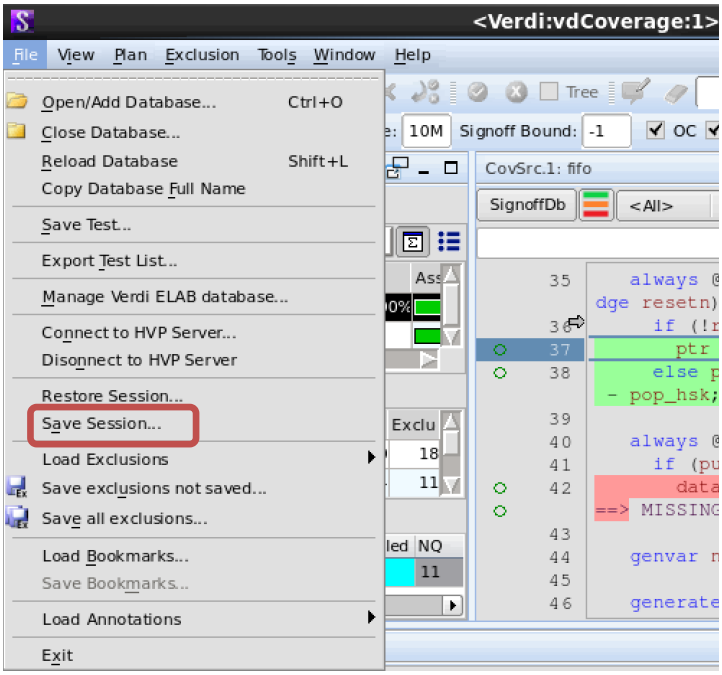


```
Go To:
42        data[0] <= data_in;
43
44    genvar n;   35 fault_check /* code removed */
45
46    generate for (n=0;n<DEPTH-1;n++) begin
47       always @(posedge clk)
48         if (push_hsk)
49           data[n+1] <= data[n];
50    end
51     endgenerate
```

Non-Activated and Non-Detected faults mean that there are portions of the code that are not being tested and point to the inefficacy of your assertions and Formal testbench. All these faults should be justified. If not, please add/edit assertions to achieve 0% Non-Activated and Non-Detected faults.

Save Session

27. Save session by click “ File” and choose “ Save Session”.



28. Exit VC Formal Signoff Dashboard :

Click on File → Exit

Restore Session

29. Open signoff dashboard interface

30. Restore session by click “File” and choose “Restore Session”