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:

Steps: Formal Signoff Flow

- Set the FPV environment and instrument the Coverage & Faults for analysis
- Run signoff flow to compute Cone of Influence and Formal Core to assess code coverage of proven assertions
- Root cause Formal Core holes and add assertions if needed
- Exclude and save manual exclusions
- Invoke and Run FTA

PRE-REQUISITES:

- Familiarity with the SystemVerilog Assertion (SVA) language and knowledge of basic formal verification concepts are required for this lab.
- Go through FPV Spec to Signoff Lab prior running this Lab

Lab Duration: 120 minutes

Files Location

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

Directory Structure	
Spec_to_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
solution/	Solution directory

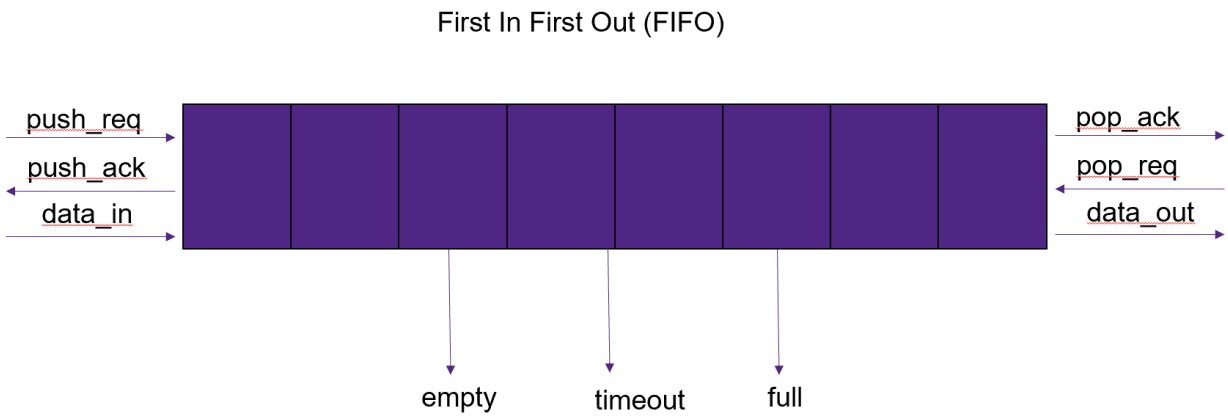
Resources

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/



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 Formal Environment

1. Load VC Formal and Verdi to set up the tool and required licenses

```
% module load vcstatic
% module load verdi
```

2. Change your working directory to run

```
% cd run
```

Now you are ready to begin the lab.

Setup Design for Coverage and FTA

3. Set Signoff configuration command as shown below. Add the below command before "read_file" command in "fifo.tcl."

```
signoff_config -type all
```

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.

4. Add the below command before "read_file" command.

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

5. Specify the read_file command as shown below.

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

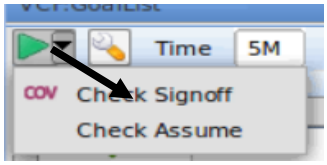
Running Signoff Flow using VC Formal

6. After all the settings, verify the FIFO design using the checkers present in sva directory and run the signoff flow.

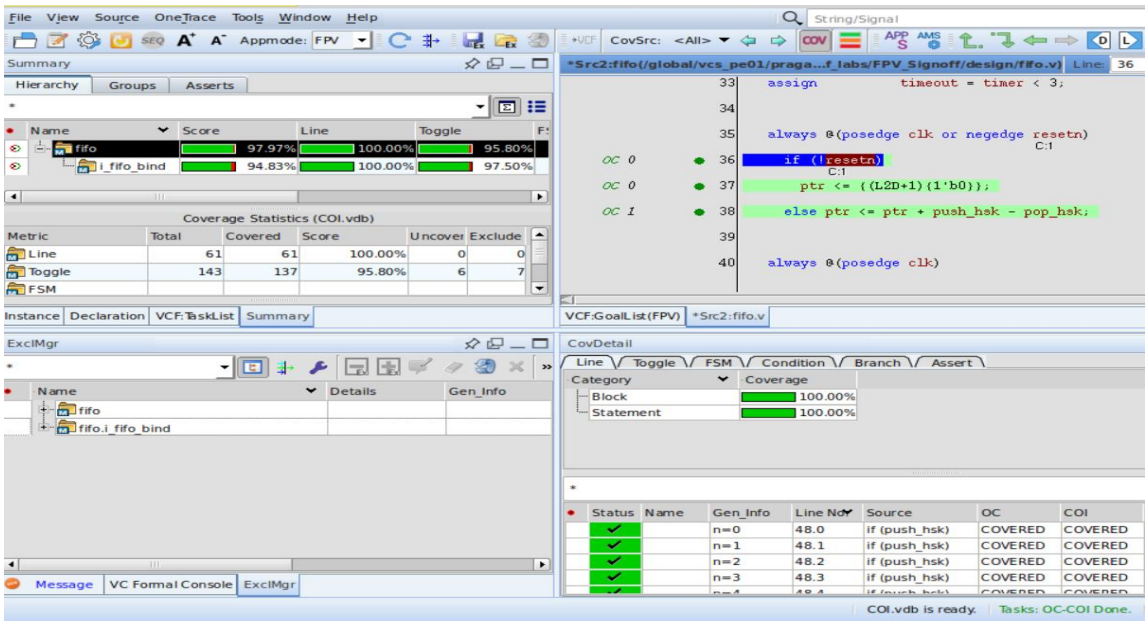
```
% vcf -f fifo_signoff.tcl -verdi &
```

Formal Signoff – Low Effort – Cone of Influence (COI) Analysis

7. Once the tool has finished running FPV, select “Check Signoff” option present in the drop down menu of the play button.



This will run Cone-of-Influence (COI) + Over Constraint (OC) analysis and exclude any cover items which are Unreachable in the design.



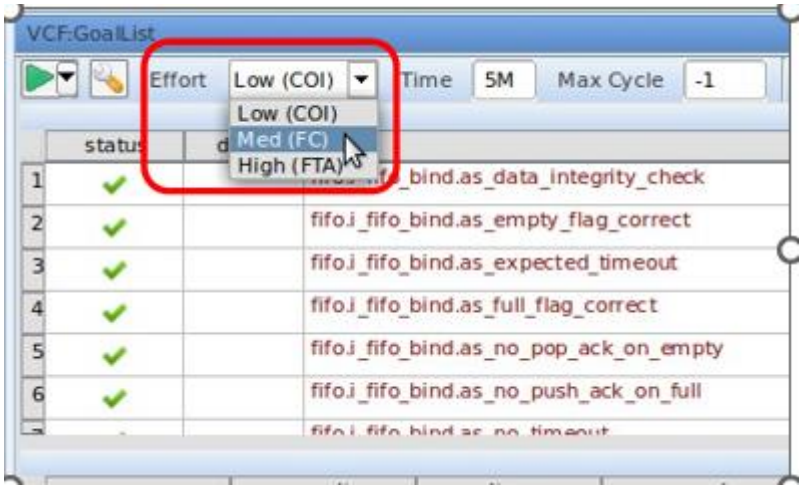
The tool will show the COI + OC coverage highlighted in the above image as 97.97%. The other highlighted red box in CovSrc window indicates the covered or uncovered status from different analysis. By default, OC is done in each effort level.

You can also run low effort or COI analysis from the VC Formal Console using the command below.

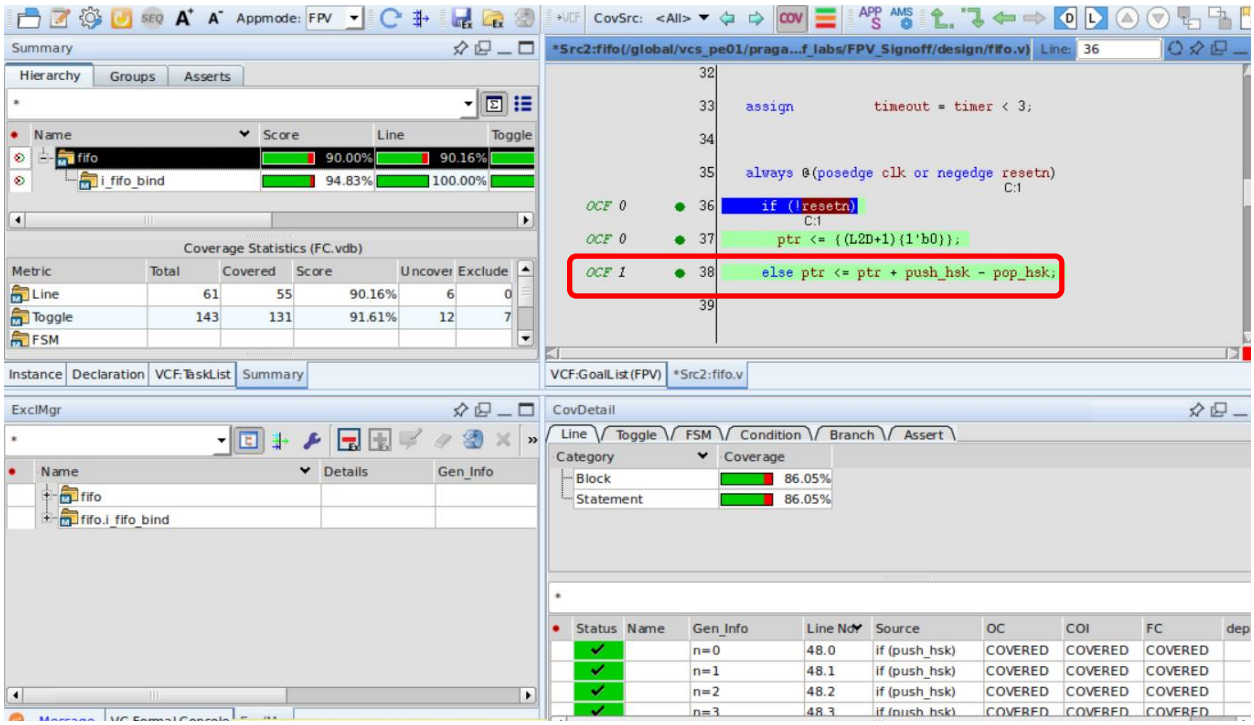
```
signoff_check -effort low -time 5M -signoff_bound -1 \
     -signoff_dashboard
```

Formal Signoff – Med Effort – Formal Core (FC) Analysis

8. Once effort Low has been run, you can run Formal Core analysis with effort Med to increase the coverage granularity of the verification performed. Select "Med (FC)" effort as shown below and run "Check Signoff"

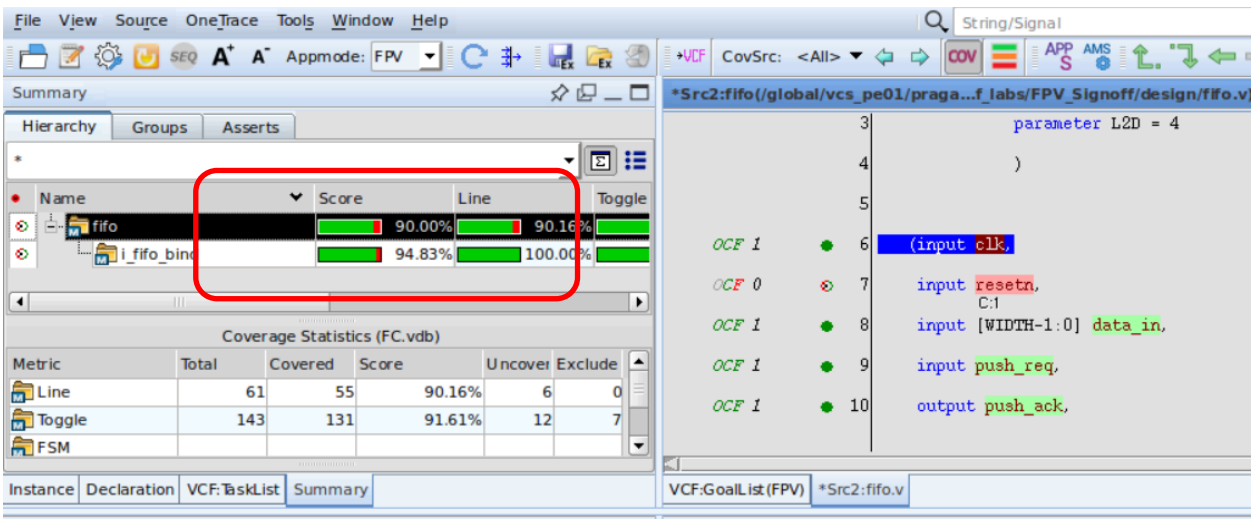


9. Once Formal Core is computed, it will give you a pop-up asking if you should load the FC coverage database. Click "Ok" to load the Formal Core coverage %. In this case, the FC coverage is computed as 90%.



Here, in CovSrc window, the highlighted red box shows that the line number 38 is covered in both OC, COI, and FC.

10. In the Source Browser of the Signoff Dashboard window, you can see code coverage in the Formal Core of the assertions that have been proven.
    a. Green → In the Formal Core
    b. Red → Outside the Formal Core



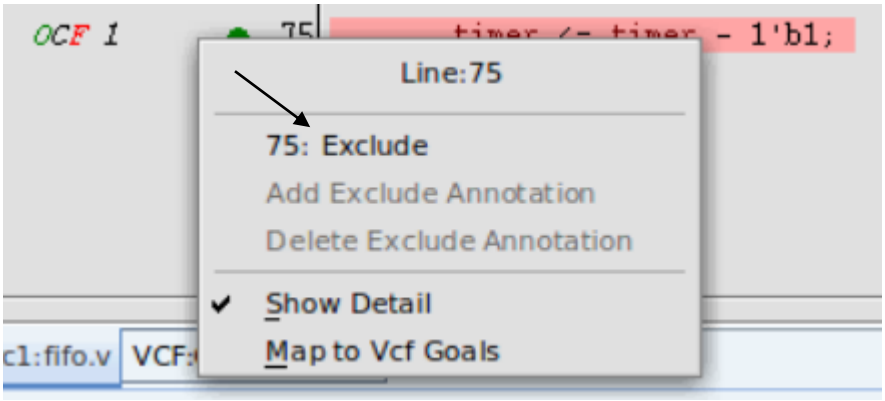
Code coverage outside the Formal Core means a portion of the code is not being tested. Make sure this is justified. If not, please add more assertions to achieve 100% Formal Core coverage.

You can also run med effort or FC analysis from the VC Formal Console using the command below.

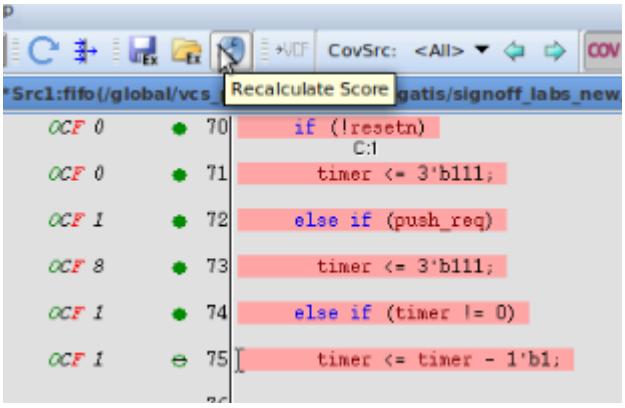
```
signoff_check  -effort med -time 5M -signoff_bound -1 \
 -signoff_dashboard
```

Exclude and save manual exclusion

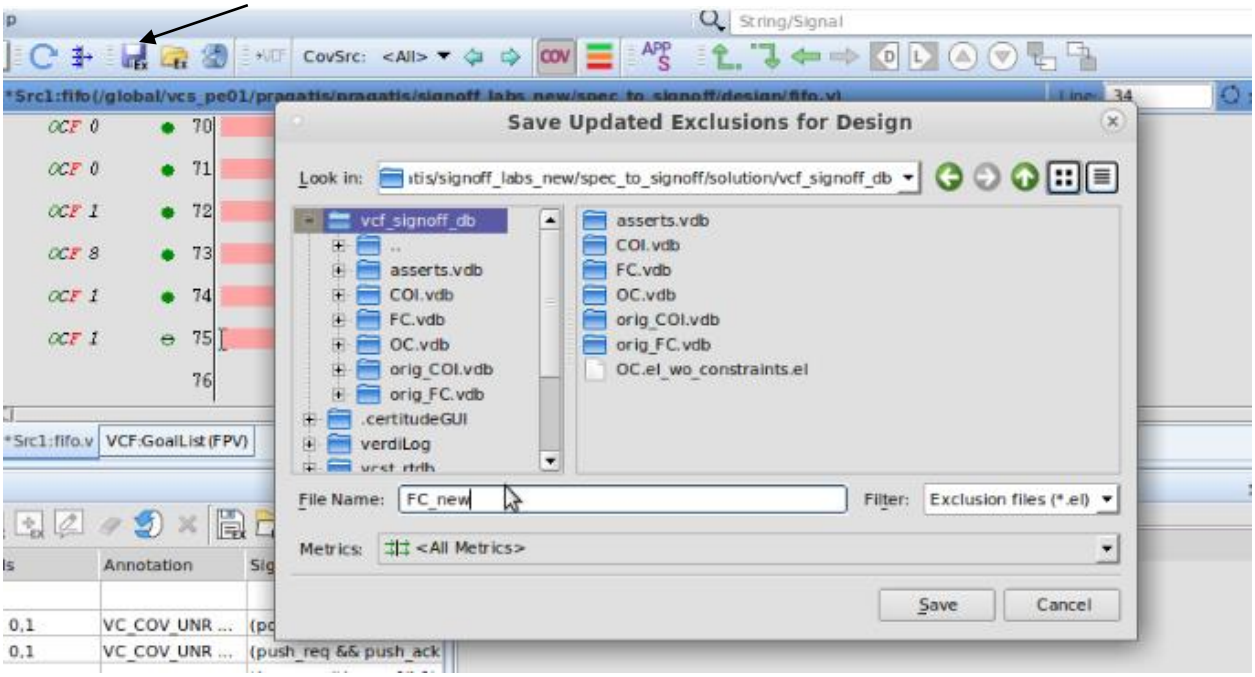
11. To mark a goal as excluded, Right Click on the respective line, and select the “Exclude” option.



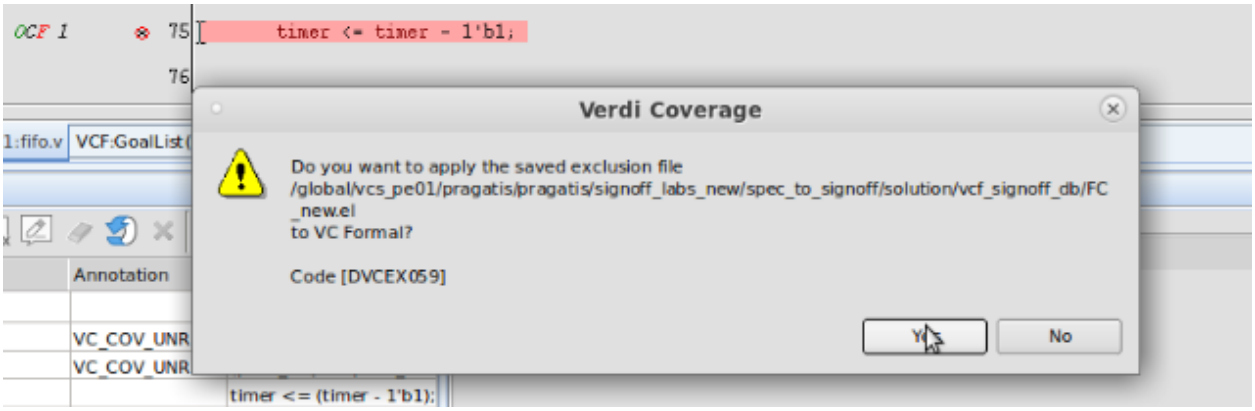
12. Once the goal is excluded, recalculate the score to apply the exclusion.



13. Once the coverage score is recalculated, save the exclusion file as shown below.

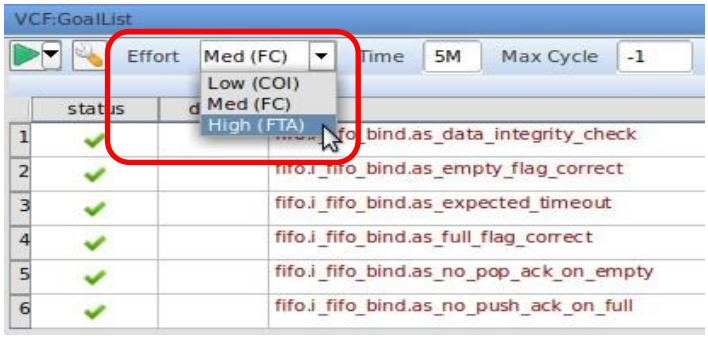


14. Once the exclusion is saved, tool will pop-up with the message if you need to apply the saved exclusion or not. Click “Yes” to apply it.

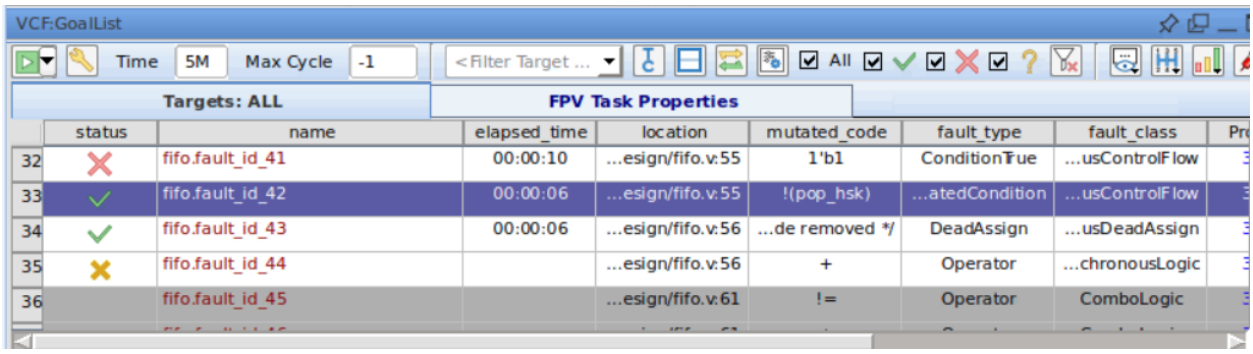


Formal Signoff – High Effort – Formal Testbench Analyzer (FTA)

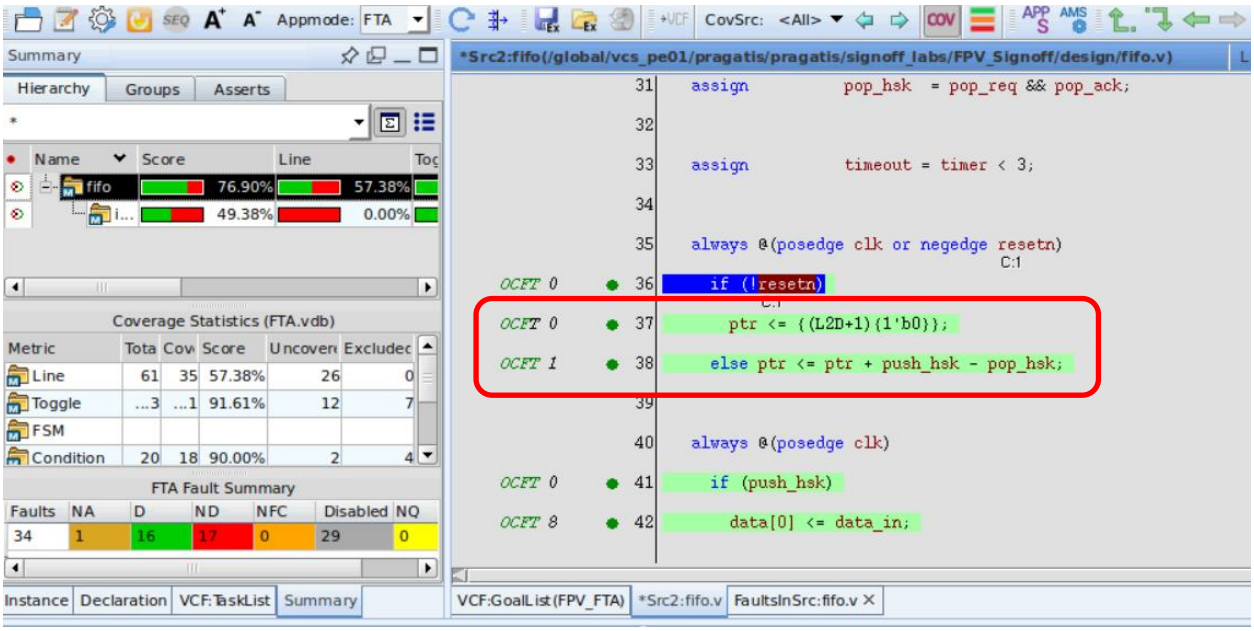
15. To further increase granularity and confidence in Signoff, you can invoke FTA by selecting effort High in VC Formal GUI.



16. This will launch the FTA GUI as shown below. You could get 50-70 faults depending on how you have implemented the SVA and the auxiliary RTL. So please do not worry about the exact number shown here in the screenshot.



17. Wait for the FTA run to finish. Once the run is finished the tool automatically loads the FTA coverage in the GUI as shown below. The tool also opens a Fault Summary View as shown below to list the outcome of the FTA run.



Here, in CovSrc window, the highlighted red box shows that the line 38 is covered in OC, COI, FC, and FTA. While line 37 does not have a fault instrumented.

You can also run high effort or FTA analysis from the VC Formal Console using the command below:

```
signoff_check -effort high -time 5M -signoff_bound -1 \
   -signoff_dashboard
```

Non-Activated means faults which are outside the COI
Non-Formalcore means faults which are outside formal core but inside COI.
Non-Detected faults means these faults were not detected by the assertions written in the code.

Overall, it shows that there are portions of the code which 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, Non-Formalcore and Non-Detected faults.

From a Code Coverage perspective, the goal is 100%. Please review the Non-Activated (yellow), Non-Formalcore (orange) and Non-Detected (red) faults and ensure they have been accounted for in the Source Browser.