VC Formal Lab
Assertion IP (AIP)
Setup and Standard Usage

Learning Objectives

In this VC Formal lab, you will use a simple AHB Lite to APB bridge example to learn to do the following:

- Create a bind file to connect AIPs to the DUT
- Select the AIP and agent type for each interface
- Connect and configure AIP instantiations
- Create a Tcl script to compile DUT and AIPs
- Set up clocks and resets
- Establish initial state for formal
- Run interactively with Verdi GUI
- Run protocol checks from connected AIPs
- Debug failures

Lab Duration:
30 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/AIP/

Directory Structure	
AIP	Lab main directory
README_VCFormal_AIP.pdf	Lab instructions
design/	Verilog RTL code of the Device Under Test (DUT)
sva/	Bind file to connect AIPs to 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/

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.

Now you are ready to begin the lab.

Create a Bind File for AIP Setup

VC Formal verification using AIPs requires the user to create a bind file to connect the AIPs to the DUT.

AIPs are located under $VC_STATIC_HOME/packages/aip/ and you select the appropriate interface protocol and version from the ones available.

In addition to the AIP protocol and version, the agent type may affect the AIP you select. For AMBA interfaces, the agent type is determined by the direction of the DUT control signals e.g., valid and ready signals, or write and read signals.

The DUT uses a Slave AHB Lite interface thus the AHB Lite AIP agent type is Master. Conversely, the DUT uses a Master APB interface thus the connected APB AIP agent type is Slave.

3. Change your working directory to AIP/sva:

```
     %cd AIP/sva
```

4. Open file bind_ahb_apb_bridge.sv provided as initial template using any text editor:

```
     %vi bind_ahb_apb_bridge.sv
```

5. Step 1 - Select AIPs to connect to the DUT:

   Replace “<Step 1>” in the bind file with the name of the matching AIP to connect to the DUT for the AMBA AHB Lite and AMBA APB interfaces respectively.

   From:

```
     // AMBA AHB Lite AIP instantiation
     bind ahb_apb_bridge snps_<Step 1>
       #(.AGENT_TYPE       (snps_ahb_aip_pkg::<Step 2>),
     ...
```

```
  // AMBA APB AIP instantiation
  bind ahb_apb_bridge snps_<Step 1>
    #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
      .AGENT_TYPE      (snps_apb_aip_pkg::<Step 2>),
  ...
```

To:

```
  // AMBA AHB Lite AIP instantiation
  bind ahb_apb_bridge snps_ahb_lite_master_aip
    #(.AGENT_TYPE      (snps_ahb_aip_pkg::<Step 2>),
  ...

  // AMBA APB AIP instantiation
  bind ahb_apb_bridge snps_apb_aip
    #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
      .AGENT_TYPE      (snps_apb_aip_pkg::<Step 2>),
  ...
```

6. Step 2 - Specify the agent type for each AIP instance:

   Replace "<Step 2>" in the bind file with the agent type selected for each AIP instantiation; the AGENT_TYPE parameter controls the AIP behavior and the properties created.

   From:

```
  // AMBA AHB Lite AIP instantiation
  bind ahb_apb_bridge snps_ahb_lite_master_aip
    #(.AGENT_TYPE      (snps_ahb_aip_pkg::<Step 2>),
  ...

  // AMBA APB AIP instantiation
  bind ahb_apb_bridge snps_apb_aip
    #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
      .AGENT_TYPE      (snps_apb_aip_pkg::<Step 2>),
  ...
```

   To:

```
  // AMBA AHB Lite AIP instantiation
  bind ahb_apb_bridge snps_ahb_lite_master_aip
    #(.AGENT_TYPE      (snps_ahb_aip_pkg::MASTER),
  ...

  // AMBA APB AIP instantiation
  bind ahb_apb_bridge snps_apb_aip
    #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
      .AGENT_TYPE      (snps_apb_aip_pkg::SLAVE),
  ...
```

7. Step 3 - Complete port connections:

   Replace “<Step 3>” with the DUT input/output port names that connect to the AIP signals explicitly listed.

   From:

```
// AMBA AHB Lite AIP instantiation
bind ahb_apb_bridge snps_ahb_lite_master_aip
  #(.AGENT_TYPE      (snps_ahb_aip_pkg::MASTER),
...
snps_ahb_lite_m
  (.hclk    (<Step 3>),
   .hresetn (<Step 3>),
   .*);

// AMBA APB AIP instantiation
bind ahb_apb_bridge snps_apb_aip
  #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
    .AGENT_TYPE      (snps_apb_aip_pkg::SLAVE),
...
snps_apb_s
  (.pclk    (<Step 3>),
   .presetn (<Step 3>),
   .pselx   (<Step 3>),
   .*);
```

   To:

```
// AMBA AHB Lite AIP instantiation
bind ahb_apb_bridge snps_ahb_lite_master_aip
  #(.AGENT_TYPE      (snps_ahb_aip_pkg::MASTER),
...
snps_ahb_lite_m
  (.hclk    (hclk),
   .hresetn (hresetn),
   .*);

// AMBA APB AIP instantiation
bind ahb_apb_bridge snps_apb_aip
  #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
    .AGENT_TYPE      (snps_apb_aip_pkg::SLAVE),
...
snps_apb_s
  (.pclk    (hclk),
   .presetn (hresetn),
   .pselx   (psel),
   .*);
```

8. Step 4 - Complete parameter values:

   Replace "<Step 4>" with parameter values that match the DUT input/output port sizes.

   From:

```
  // AMBA AHB Lite AIP instantiation
  bind ahb_apb_bridge snps_ahb_lite_master_aip
    #(.AGENT_TYPE      (snps_ahb_aip_pkg::MASTER),
      .ADDR_WIDTH      (<Step 4>),
      .DATA_WIDTH      (<Step 4>),
  ...

  // AMBA APB AIP instantiation
  bind ahb_apb_bridge snps_apb_aip
    #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
      .AGENT_TYPE      (snps_apb_aip_pkg::SLAVE),
      .PSEL_WIDTH      (<Step 4>),
      .ADDR_WIDTH      (<Step 4>),
      .WDATA_WIDTH     (<Step 4>),
      .RDATA_WIDTH     (<Step 4>),
  ...
```

   To:

```
  // AMBA AHB Lite AIP instantiation
  bind ahb_apb_bridge snps_ahb_lite_master_aip
    #(.AGENT_TYPE      (snps_ahb_aip_pkg::MASTER),
      .ADDR_WIDTH      (HADDR_WIDTH),
      .DATA_WIDTH      (DATA_WIDTH),
  ...

  // AMBA APB AIP instantiation
  bind ahb_apb_bridge snps_apb_aip
    #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
      .AGENT_TYPE      (snps_apb_aip_pkg::SLAVE),
      .PSEL_WIDTH      (PSEL_WIDTH),
      .ADDR_WIDTH      (PADDR_WIDTH),
      .WDATA_WIDTH     (DATA_WIDTH),
      .RDATA_WIDTH     (DATA_WIDTH),
  ...
```

9. Save bind_ahb_apb_bridge.sv file and exit editor.

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 AHB Lite to APB bridge, used in this lab and add the necessary commands to load the AIPs and associated bind file.

10. Change your working directory to AIP/run:

```
%cd ../run
```

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

```
%vi run.tcl
```

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

```
set_fml_appmode FPV
```

13. Add command to load AIPs instantiated for verification of the DUT:

```
aip_load -protocol {APB AHB}
```

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

```
set design ahb_apb_bridge
```

15. Add command to compile DUT and bind file:

    The DUT files and filelist are located under directory AIP/design.

```
read_file -top $design -format sverilog -sva -vcs \
  {-f ../design/filelist ../sva/bind_ahb_apb_bridge.sv}
```

    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 ../sva/bind_ahb_apb_bridge.sv}
elaborate $design -sva
```

16. Add clock and reset information:

```
create_clock hclk -period 100
create_reset hresetn -sense low
```

17. Add design initialization commands:

```
sim_run -stable
sim_save_reset
```

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

18. Save run.tcl file and exit editor.

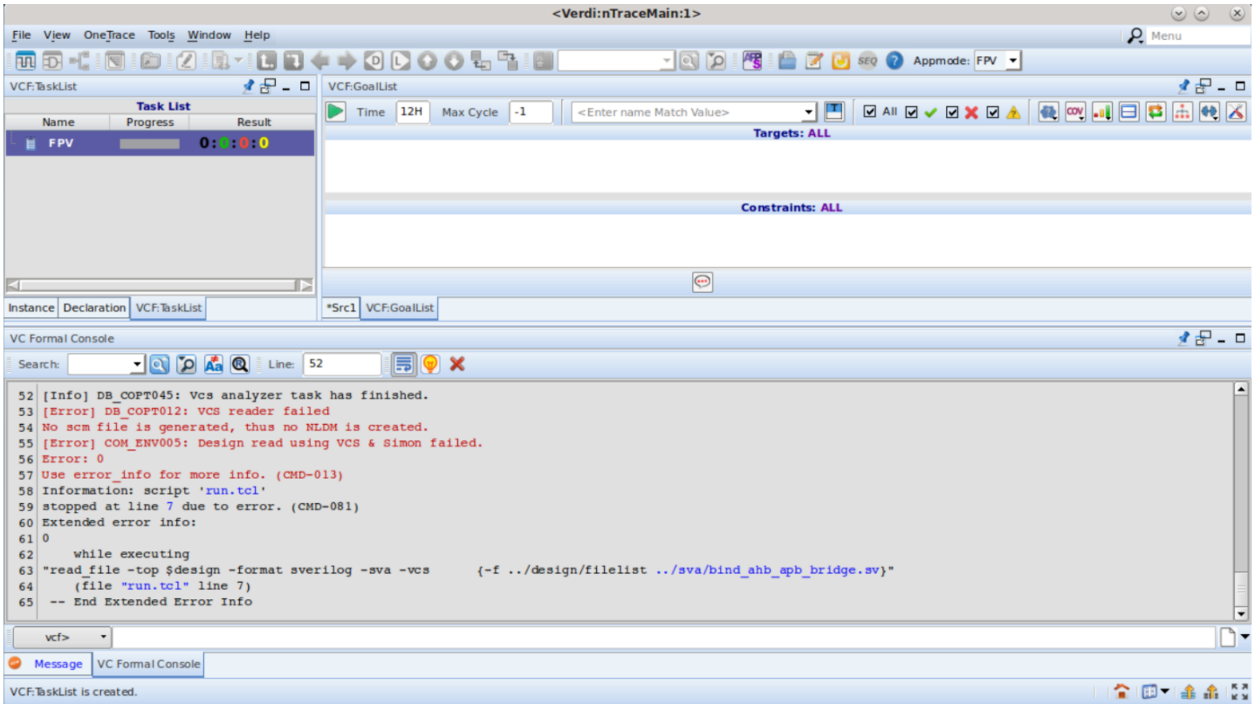
Start VC Formal in Verdi GUI Mode

19. Start the tool in Verdi GUI mode:

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

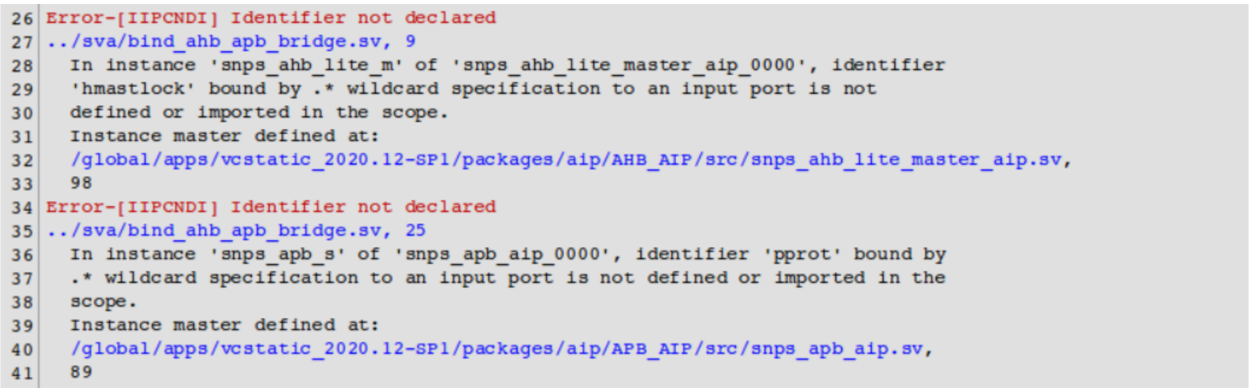
VC Formal starts in the Verdi GUI mode, with icons, tables, tabs, and windows especially designed for property verification with the FPV App. FPV is the default App mode.

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.



```
[Info] DB_COPT045: Vcs analyzer task has finished.
[Error] DB_COPT012: VCS reader failed
No scm file is generated, thus no NLDM is created.
[Error] COM_ENV005: Design read using VCS & Simon failed.
Error: 0
Use error_info for more info. (CMD-013)
Information: script 'run.tcl'
stopped at line 7 due to error. (CMD-081)
Extended error info:
0
    while executing
"read_file -top $design -format sverilog -sva -vcs      {-f ../design/filelist ../sva/bind_ahb_apb_bridge.sv}"
    (file "run.tcl" line 7)
 -- End Extended Error Info
```

The “VC Formal Console” window shows that the compilation failed with four errors due to ports in the AIPs that do not have equivalent signals in the DUT. The errors point to signals hprot and hmastlock for the AHB Lite AIP, and pprot and pstrb for the APB AIP.



```
Error-[IIPCNDI] Identifier not declared
../sva/bind_ahb_apb_bridge.sv, 9
  In instance 'snps_ahb_lite_m' of 'snps_ahb_lite_master_aip_0000', identifier
  'hmastlock' bound by .* wildcard specification to an input port is not
  defined or imported in the scope.
  Instance master defined at:
  /global/apps/vcstatic_2020.12-SP1/packages/aip/AHB_AIP/src/snps_ahb_lite_master_aip.sv,
  98
Error-[IIPCNDI] Identifier not declared
../sva/bind_ahb_apb_bridge.sv, 25
  In instance 'snps_apb_s' of 'snps_apb_aip_0000', identifier 'pprot' bound by
  .* wildcard specification to an input port is not defined or imported in the
  scope.
  Instance master defined at:
  /global/apps/vcstatic_2020.12-SP1/packages/aip/APB_AIP/src/snps_apb_aip.sv,
  89
```

Add Missing Ports to Bind File

20. Open file bind_ahb_apb_bridge.sv using any text editor:

```
%vi ../sva/bind_ahb_apb_bridge.sv
```

21. Add missing ports to the bind statements by declaring them explicitly and tying them to zero:

From:

```
// AMBA AHB Lite AIP instantiation
bind ahb_apb_bridge snps_ahb_lite_master_aip
  #(.AGENT_TYPE      (snps_ahb_aip_pkg::MASTER),
...
snps_ahb_lite_m
  (.hclk     (hclk),
   .hresetn (hresetn),
   .*);

// AMBA APB AIP instantiation
bind ahb_apb_bridge snps_apb_aip
  #(.PROTOCOL_VER    (snps_apb_aip_pkg::AMBA3),
    .AGENT_TYPE      (snps_apb_aip_pkg::SLAVE),
...
snps_apb_s
  (.pclk     (hclk),
   .presetn (hresetn),
   .pselx   (psel),
   .*);
```

To:

```
// AMBA AHB Lite AIP instantiation
bind ahb_apb_bridge snps_ahb_lite_master_aip
  #(.AGENT_TYPE      (snps_ahb_aip_pkg::MASTER),
...
snps_ahb_lite_m
```

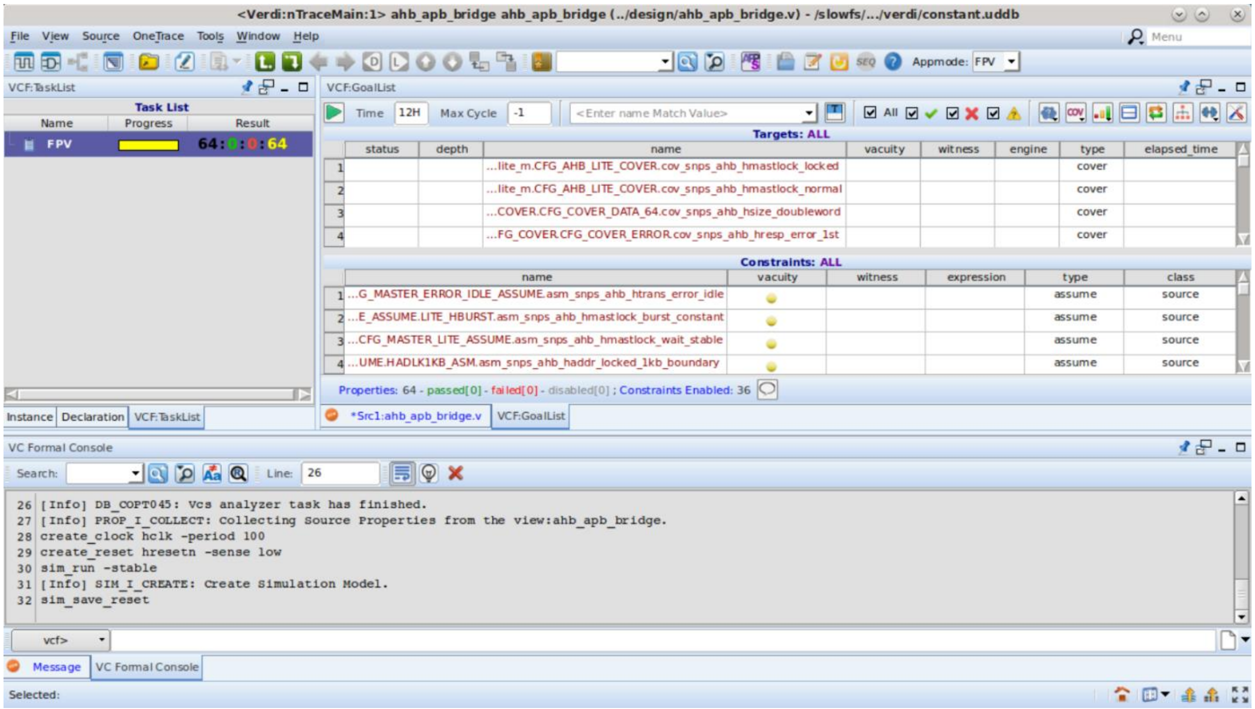
```
        (.hclk     (hclk),
         .hresetn  (hresetn),
         .hprot    (4'b0),
         .hmastlock (1'b0),
         .*);

      // AMBA APB AIP instantiation
      bind ahb_apb_bridge snps_apb_aip
        #(.PROTOCOL_VER     (snps_apb_aip_pkg::AMBA3),
          .AGENT_TYPE       (snps_apb_aip_pkg::SLAVE),
      ...
      snps_apb_s
        (.pclk     (hclk),
         .presetn  (hresetn),
         .pselx    (psel),
         .pprot    (3'b0),
         .pstrb    (8'b0),
         .*);
```

22. Save bind file and exit editor.

Restart VC Formal in Verdi GUI Mode

23. Restart VC Formal with the modified bind file:

    Click on the Restart VCST icon.



<Verdi:nTraceMain:1> ahb_apb_bridge ahb_apb_bridge (../design/ahb_apb_bridge.v) - /slowfs/.../verdi/constant.uddb
File View Source OneTrace Tools Window Help
Menu
Appmode: FPV
VCF:TaskList
VCF:GoalList
Time 12H Max Cycle -1
<Enter name Match Value>
All
Instance Declaration VCF:TaskList
*Src1:ahb_apb_bridge.v VCF:GoalList
VC Formal Console
Search:
Line: 26
vcf>
Message VC Formal Console
Selected:

Task List

Name	Progress	Result
FPV		64:0:0:64

Targets: ALL

	status	depth	name	vacuity	witness	engine	type	elapsed_time
1			...lite_m.CFG_AHB_LITE_COVER.cov_snps_ahb_hmastlock_locked				cover	
2			...lite_m.CFG_AHB_LITE_COVER.cov_snps_ahb_hmastlock_normal				cover	
3			...COVER.CFG_COVER_DATA_64.cov_snps_ahb_hsize_doubleword				cover	
4			...FG_COVER.CFG_COVER_ERROR.cov_snps_ahb_hresp_error_1st				cover	

Constraints: ALL

	name	vacuity	witness	expression	type	class
1	...G_MASTER_ERROR_IDLE_ASSUME.asm_snps_ahb_htrans_error_idle	●			assume	source
2	...E_ASSUME.LITE_HBURST.asm_snps_ahb_hmastlock_burst_constant	●			assume	source
3	...CFG_MASTER_LITE_ASSUME.asm_snps_ahb_hmastlock_wait_stable	●			assume	source
4	...UME.HADLK1KB_ASM.asm_snps_ahb_haddr_locked_1kb_boundary	●			assume	source

Properties: 64 - passed[0] - failed[0] - disabled[0] ; Constraints Enabled: 36

```
[Info] DB_COPT045: Vcs analyzer task has finished.
[Info] PROP_I_COLLECT: Collecting Source Properties from the view:ahb_apb_bridge.
create_clock hclk -period 100
create_reset hresetn -sense low
sim_run -stable
[Info] SIM_I_CREATE: Create Simulation Model.
sim_save_reset
```

24. 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 Customize View Settings icon at the upper right of the property table.

    Instantiating the AMBA AHB Lite and AMBA APB AIPs resulted in the generation of 64 targets (assertions and cover properties) as well as 35 constraints (or assumptions) with minimal setup effort.

Run Formal Proofs and Review Results

25. Start property verification:

    Click on the Start Check icon ▶.

26. Hide the constraints table:

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

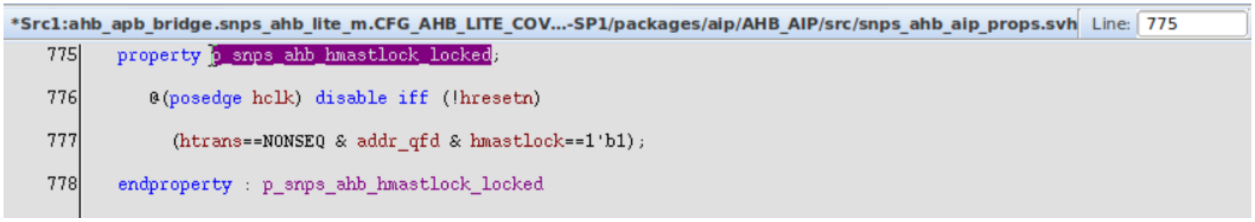
27. Filter “Targets” table to keep failed targets:

    Select to view only failed targets □ All □ ✓ ☑ ✗ □ ⚠.

Debug Failures – Uncoverable Cover Properties

28. Debug failures for uncoverable cover properties:

    Double click on the name for property “ahb_apb_bridge.snps_ahb_lite_m.CFG_AHB_LITE_COVER.cov_snps_ahb_hmastlock_locked” to open the source code window and trace the property to confirm that the expression is indeed uncoverable due to signal hmastlock being tied to 1’b0 in the bind file.



*Src1:ahb_apb_bridge.snps_ahb_lite_m.CFG_AHB_LITE_COV...-SP1/packages/aip/AHB_AIP/src/snps_ahb_aip_props.svh Line: 775

```
property p_snps_ahb_hmastlock_locked;
   @(posedge hclk) disable iff (!hresetn)
     (htrans==NONSEQ & addr_qfd & hmastlock==1'b1);
endproperty : p_snps_ahb_hmastlock_locked
```

The four other cover properties are uncoverable due to signal hprot being tied to 4’b0.

29. Disable debugged uncoverable cover properties:

Select the five uncoverable cover properties and disable them.

30. Add command to disable uncoverable cover properties to the Tcl file:

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

```
fvdisable {
ahb_apb_bridge.snps_ahb_lite_m.CFG_AHB_LITE_COVER.cov_snps_ahb_hmastlock_locked
ahb_apb_bridge.snps_ahb_lite_m.CFG_COVER.cov_snps_ahb_hprot_data_access
ahb_apb_bridge.snps_ahb_lite_m.CFG_COVER.cov_snps_ahb_hprot_privileged_access
ahb_apb_bridge.snps_ahb_lite_m.CFG_COVER.cov_snps_ahb_hprot_bufferable
ahb_apb_bridge.snps_ahb_lite_m.CFG_COVER.cov_snps_ahb_hprot_cacheable
}
```

31. Save edited run.tcl Tcl file:

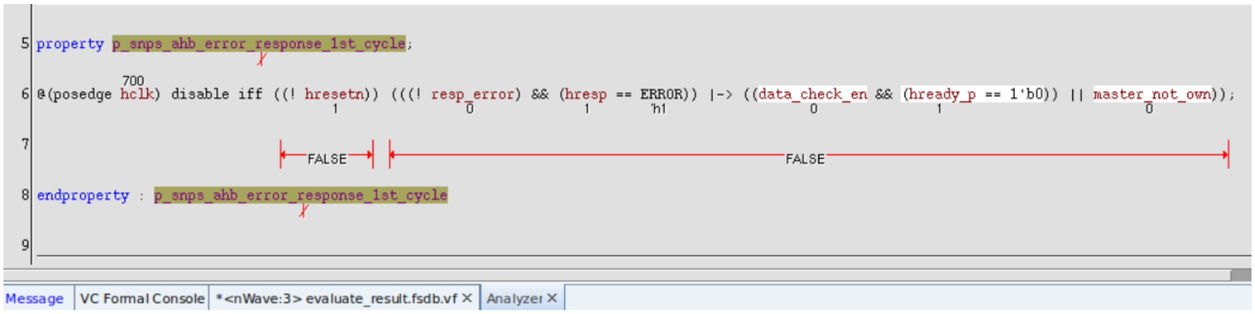
Click on the Save icon 💾.

Debug Failures – Falsified Assertions

32. Debug falsified assertion:

Double click on ✖ under the status of property "ahb_apb_bridge.snps_ahb_lite_m.CFG_MASTER_ASSERT.CFG_MASTER_ERROR_ASSERT.ast_snps_ahb_error_response_1st_cycle" 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.)

Click on "Analyzer" tab at the bottom to see the property expression.



```
property p_snps_ahb_error_response_1st_cycle;
@(posedge hclk) disable iff ((! hresetn)) (((! resp_error) && (hresp == ERROR)) |-> ((data_check_en && (hready_p == 1'b0)) || master_not_own));

endproperty : p_snps_ahb_error_response_1st_cycle
```

The reason this property fails is because there is a constraint missing in the setup. Since there is a single AHB Lite Slave, signal hready of the AHB Lite Master AIP should be same as signal hreadyout of the DUT.

33. Add constraint for signal hready to the Tcl file:

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

```
  fvassume asm_hready_hreadyout -expr {hready==hreadyout}
```

34. Save edited run.tcl Tcl file:

    Click on the Save icon 💾.

Restart the Run and Verify Fix

35. Restart VC Formal with the modified Tcl file:

    Click on the Restart VCST icon ↻.

36. Re-run property verification:

    Click on the Start Check icon ▶.

    Observe that all properties are now proven and only the five disabled uncoverable cover properties remain with status “not_run”.

37. Exit VC Formal:

    Click on File → Exit