SYNOPSYS®

VC Formal Lab
Automatically Extracted Property (AEP) 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 appmode
- Run interactively with Verdi GUI
- Review design complexity statistics and setup
- Enable AEP switch when read design
- Set up clocks and resets
- Establish initial state for formal
- Generate property
- Run checks and review results
- Filter specified properties name
- Debug failures
- Add FSM fairness constraint for inputs
- Configure FSM property generation
- Waive for confirmed falsified properties
- Save and restore session
- Run in interactive shell without Verdi GUI
- Run in batch mode

Lab Duration: 30 minutes

Familiarity with basic formal verification concepts is required for this lab.

Files Location

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

Directory Structure	
AEP	Lab main directory
README_VCFormal_AEP.pdf	Lab instructions
design/	Verilog RTL code of the Device Under Test (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.

3. Change your working directory to AEP/run:

```
    %cd AEP/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.

The DUT files and filelist are located under AEP/design.

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

```
    %vi run.tcl
```

5. Add command to enable AEP App mode :

```
    set_fml_appmode AEP
```

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

```
    set design traffic
```

7. Add fml var for unique name presentation, easier for waiver reuse after design modify:

```
    set_fml_var fml_aep_unique_name true
```

8. Add app var for compilation to dump more FSM info to help potential FSM issue debug:

```
    set_app_var fml_enable_fsm_report_complexity true
    set_app_var fml_trace_auto_fsm_state_extraction true
```

9. Add command to compile DUT and SVA properties:

```
    read_file -top $design -format verilog \
     -vcs {-f ../design/filelist}
```

Or it can be separated into 2-step flow using the analyze and elaborate commands. This is suitable for complex designs or mix-language designs.

```
    analyze -format verilog -vcs {-f ../design/filelist}
    elaborate $design
```

10. Save run.tcl file and exit editor.

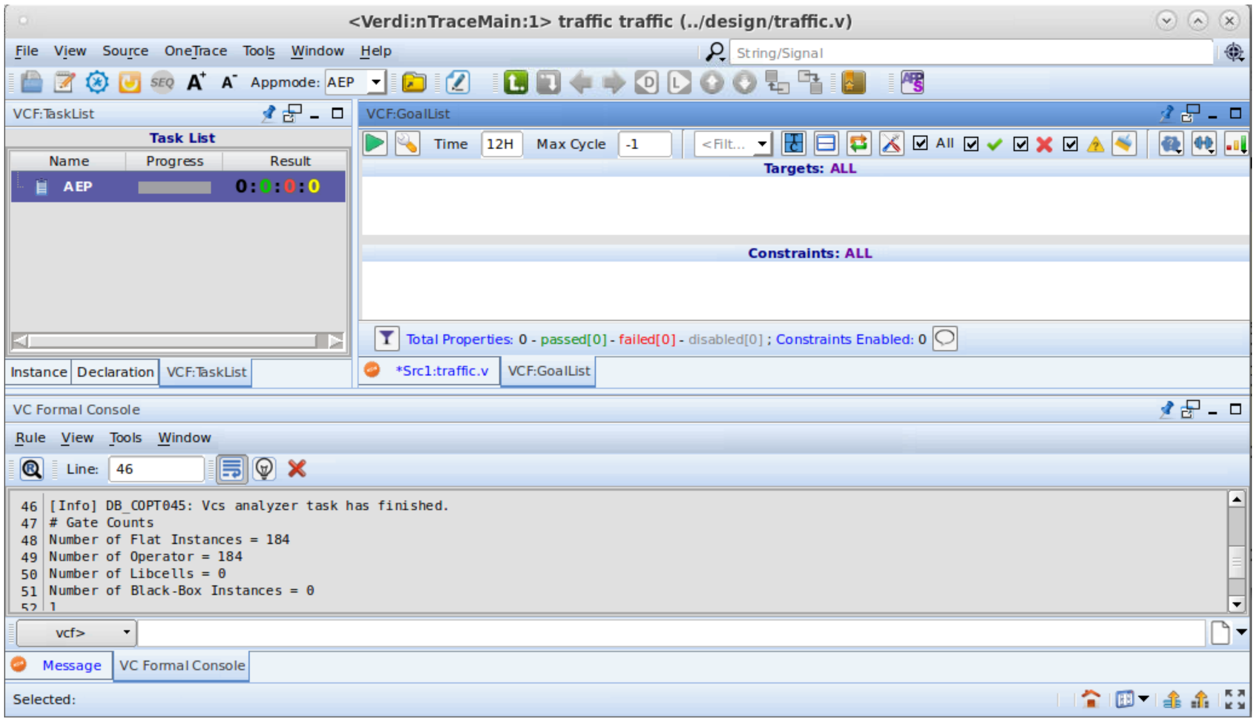
VC Formal can be run in three modes: interactive Verdi GUI mode, interactive without Verdi GUI using shell mode, and non-interactive batch mode.

Mode 1: Start VC Formal in Verdi GUI Mode

11. 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. We had overridden this in our "run.tcl file via command "set_fml_appmode AEP" so GUI has AEP-mode set.



<Verdi:nTraceMain:1> traffic traffic (../design/traffic.v)
File View Source OneTrace Tools Window Help
String/Signal
Appmode: AEP
VCF:TaskList
Task List
Name Progress Result
AEP 0:0:0:0
VCF:GoalList
Time 12H Max Cycle -1
<Filt...
All
Targets: ALL
Constraints: ALL
Total Properties: 0 - passed[0] - failed[0] - disabled[0] ; Constraints Enabled: 0
Instance Declaration VCF:TaskList
*Src1:traffic.v VCF:GoalList
VC Formal Console
Rule View Tools Window
Line: 46
vcf>
Message VC Formal Console
Selected:

```
[Info] DB_COPT045: Vcs analyzer task has finished.
# Gate Counts
Number of Flat Instances = 184
Number of Operator = 184
Number of Libcells = 0
Number of Black-Box Instances = 0
1
```

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.

12. 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.

Review Design Information and Setup

13. Review design information and setup Appmode: AEP on top menu bar ensuring you are

in AEP-mode, and in TaskList you can see 0-AEP property.

14. Click on the "Setup Assist" icon on the upper right above the property table and choose "Show Complxity". Examine items under "Missing Clock" and "Missing Reset" and trace from the source file to see that the active phase of rst is high.

Revise Setup and generate AEP

15. Click on the Edit Tcl Project File icon on the upper left and add the missing switch "-aep all" to the existing command "read_file" in the Tcl file to generate AEP properties.

```
     read_file -top $design -format verilog -aep \
all+fsm_deadlock+fsm_livelock \
     -vcs {-f ../design/filelist}
```

Or for the 2-step design:

```
     elaborate $design -aep all+fsm_deadlock+fsm_livelock
```

Note: if design is huge and more complex, it's not suggested to set "all" type at initial run. You can select needed types below by using "+": arith_oflow, bounds_check, x_assign, set_reset, full_case (priority_case), parallel_case (unique_case), multi_driver, conflict_driver, floating_bus. For example: -aep arith_oflow+x_assign

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

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

17. Add commands to initialize the DUT by holding reset active until sequential elements (latches and flip flops) values are stable. And save the initial state.

```
     sim_run -stable
     sim_save_reset
```

18. Save edited run.tcl Tcl file:

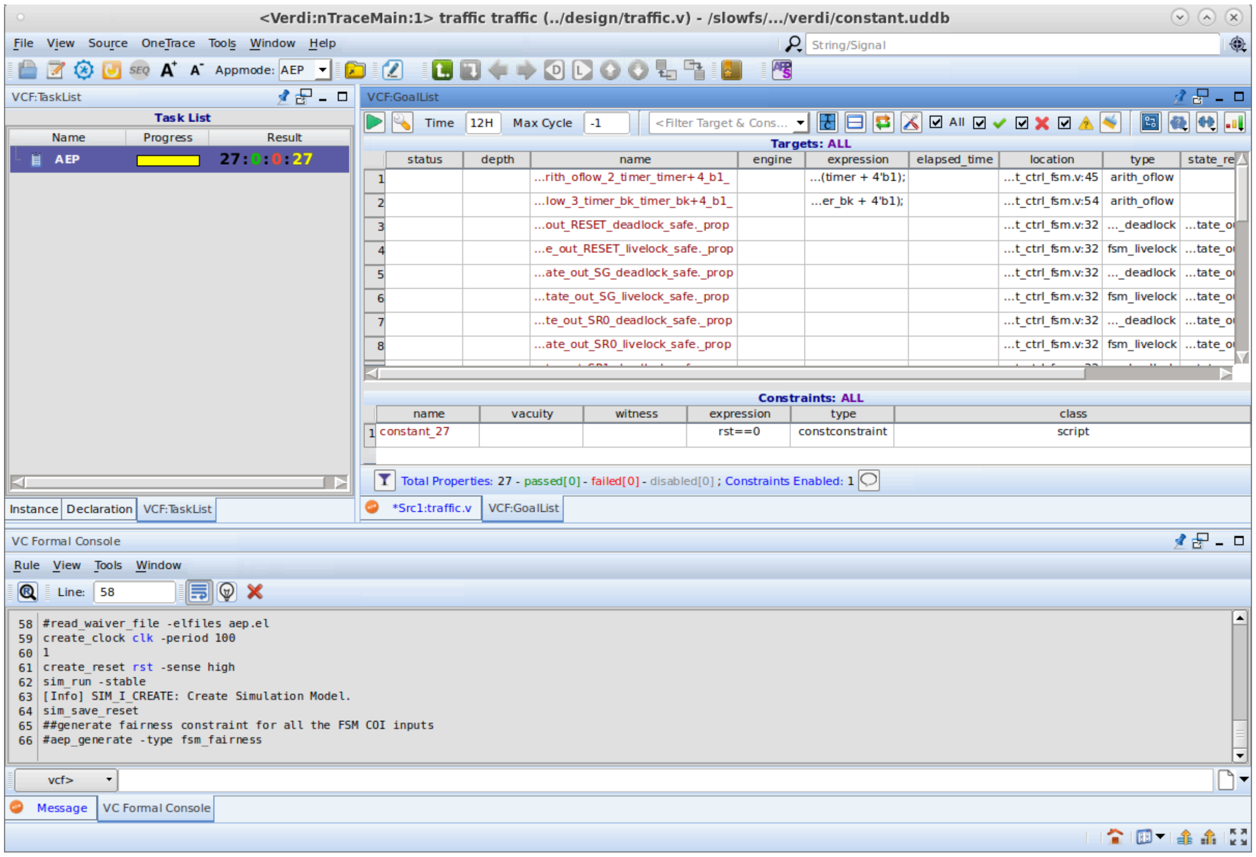
    Click on the Save icon .

19. Restart VC Formal:

    Click on the Restart VCST icon .

20. Check setup and design information:

There are 27-AEP properties show in TaskList, and the status for these properties is blank due to checks has not been run yet.



<Verdi:nTraceMain:1> traffic traffic (../design/traffic.v) - /slowfs/.../verdi/constant.uddb

VCF:TaskList — Task List

Name	Progress	Result
AEP		27:0:0:27

VCF:GoalList — Time 12H, Max Cycle -1

Targets: ALL

	status	depth	name	engine	expression	elapsed_time	location	type	state_re
1			...rith_oflow_2_timer_timer+4_b1_		...(timer + 4'b1);		...t_ctrl_fsm.v:45	arith_oflow	
2			...low_3_timer_bk_timer_bk+4_b1_		...er_bk + 4'b1);		...t_ctrl_fsm.v:54	arith_oflow	
3			...out_RESET_deadlock_safe._prop				...t_ctrl_fsm.v:32	..._deadlock	...tate_o
4			...e_out_RESET_livelock_safe._prop				...t_ctrl_fsm.v:32	fsm_livelock	...tate_o
5			...ate_out_SG_deadlock_safe._prop				...t_ctrl_fsm.v:32	..._deadlock	...tate_o
6			...tate_out_SG_livelock_safe._prop				...t_ctrl_fsm.v:32	fsm_livelock	...tate_o
7			...te_out_SR0_deadlock_safe._prop				...t_ctrl_fsm.v:32	..._deadlock	...tate_o
8			...ate_out_SR0_livelock_safe._prop				...t_ctrl_fsm.v:32	fsm_livelock	...tate_o

Constraints: ALL

	name	vacuity	witness	expression	type	class
1	constant_27			rst==0	constconstraint	script

Total Properties: 27 - passed[0] - failed[0] - disabled[0] ; Constraints Enabled: 1

VC Formal Console

```
#read_waiver_file -elfiles aep.el
create_clock clk -period 100
1
create_reset rst -sense high
sim_run -stable
[Info] SIM_I_CREATE: Create Simulation Model.
sim_save_reset
##generate fairness constraint for all the FSM COI inputs
#aep_generate -type fsm_fairness
```

Run AEP check and Review Results

21. Start auto-extracted property verification:

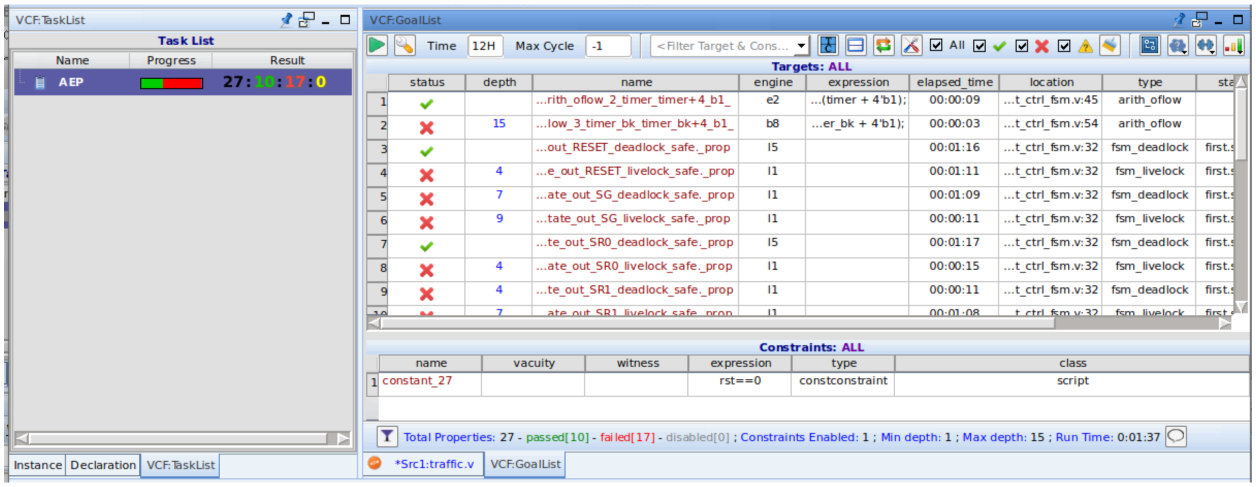
    Click on the Start Check icon ▶.

22. Hide the constraints table:

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

23. Check result in Targets:ALL:

    27-AEP: 10 proven ✓, 17 falsified ✗.



Name	Progress	Result
AEP		27:10:17:0

Time 12H Max Cycle -1

Targets: ALL

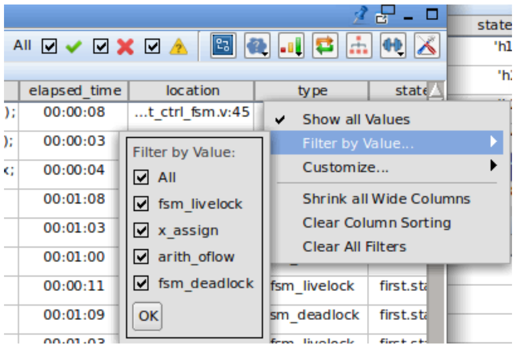
	status	depth	name	engine	expression	elapsed_time	location	type	sta
1	✓		...rith_oflow_2_timer_timer+4_b1_	e2	...(timer + 4'b1);	00:00:09	...t_ctrl_fsm.v:45	arith_oflow	
2	✗	15	...low_3_timer_bk_timer_bk+4_b1_	b8	...er_bk + 4'b1);	00:00:03	...t_ctrl_fsm.v:54	arith_oflow	
3	✓		...out_RESET_deadlock_safe._prop	l5		00:01:16	...t_ctrl_fsm.v:32	fsm_deadlock	first.s
4	✗	4	...e_out_RESET_livelock_safe._prop	l1		00:01:11	...t_ctrl_fsm.v:32	fsm_livelock	first.s
5	✗	7	...ate_out_SG_deadlock_safe._prop	l1		00:01:09	...t_ctrl_fsm.v:32	fsm_deadlock	first.s
6	✗	9	...tate_out_SG_livelock_safe._prop	l1		00:00:11	...t_ctrl_fsm.v:32	fsm_livelock	first.s
7	✓		...te_out_SR0_deadlock_safe._prop	l5		00:01:17	...t_ctrl_fsm.v:32	fsm_deadlock	first.s
8	✗	4	...ate_out_SR0_livelock_safe._prop	l1		00:00:15	...t_ctrl_fsm.v:32	fsm_livelock	first.s
9	✗	4	...te_out_SR1_deadlock_safe._prop	l1		00:00:11	...t_ctrl_fsm.v:32	fsm_deadlock	first.s
10	✗	7	...ate_out_SR1_livelock_safe._prop	l1		00:01:08	...t_ctrl_fsm.v:32	fsm_livelock	first.s

Constraints: ALL

	name	vacuity	witness	expression	type	class
1	constant_27			rst==0	constconstraint	script

Total Properties: 27 - passed[10] - failed[17] - disabled[0] ; Constraints Enabled: 1 ; Min depth: 1 ; Max depth: 15 ; Run Time: 0:01:37

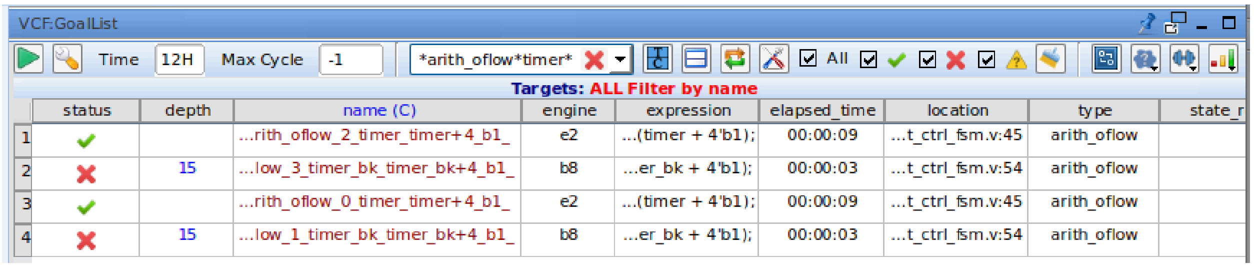
Can click ☑ All ☑ ✓ ☑ ✗ ☑ ⚠ to filter status or right click any title of Targets column, for example, right click "type" and select "Filter by Value", can choose "fsm_livelock" type, "x_assign" type or "arith_oflow" type and others or all.



Filter out Specific Properties Name

24. For unique name setting, it's easier to filter the property name.

It's based on design hierarchy, AEP types and expression for property naming.



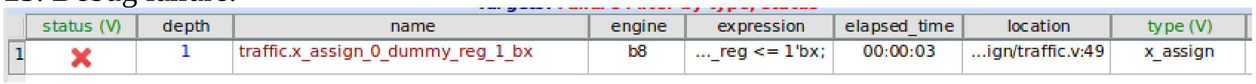
Time 12H Max Cycle -1 *arith_oflow*timer*

Targets: ALL Filter by name

	status	depth	name (C)	engine	expression	elapsed_time	location	type	state_r
1	✓		...rith_oflow_2_timer_timer+4_b1_	e2	...(timer + 4'b1);	00:00:09	...t_ctrl_fsm.v:45	arith_oflow	
2	✗	15	...low_3_timer_bk_timer_bk+4_b1_	b8	...er_bk + 4'b1);	00:00:03	...t_ctrl_fsm.v:54	arith_oflow	
3	✓		...rith_oflow_0_timer_timer+4_b1_	e2	...(timer + 4'b1);	00:00:09	...t_ctrl_fsm.v:45	arith_oflow	
4	✗	15	...low_1_timer_bk_timer_bk+4_b1_	b8	...er_bk + 4'b1);	00:00:03	...t_ctrl_fsm.v:54	arith_oflow	

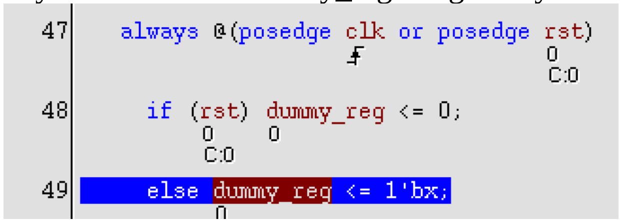
Debug x_assign Failure

25. Debug failure:



	status (V)	depth	name	engine	expression	elapsed_time	location	type (V)
1	✗	1	traffic.x_assign_0_dummy_reg_1_bx	b8	..._reg <= 1'bx;	00:00:03	...ign/traffic.v:49	x_assign

Double click on the "name" of property "traffic.x_assign_0_dummy_reg_1_bx" in the source code window.
The reason for property fails is due to dummy_reg assigned by x at line#49.



```
47    always @(posedge clk or posedge rst)
48      if (rst) dummy_reg <= 0;
49      else dummy_reg <= 1'bx;
```

Correct Erroneous in RTL Design

26. To correct error, we need fix the RTL-source file 'traffic.v'.

The design can be edit on the GUI interface by click on Verdi "Edit Source File" icon and modify the file below:

Original design:

```
  always @(posedge clk or posedge rst)
  if (rst) dummy_reg <= 0;
  else dummy_reg <= 1'bx;
```

Modified design:

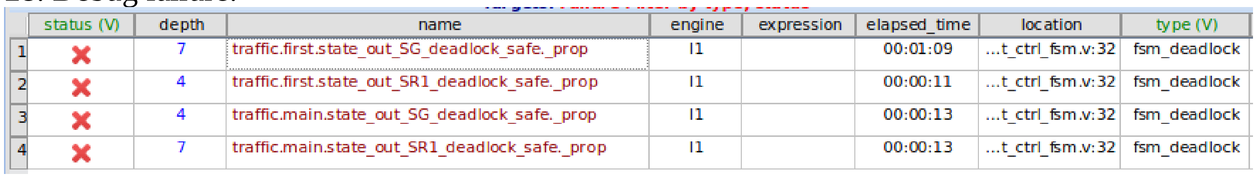
```
  always @(posedge clk or posedge rst)
  if (rst) dummy_reg <= 0;
  else dummy_reg <= 1'b1;
```

27. Save file:

Click on the Save icon .

Debug fsm_deadlock Failure

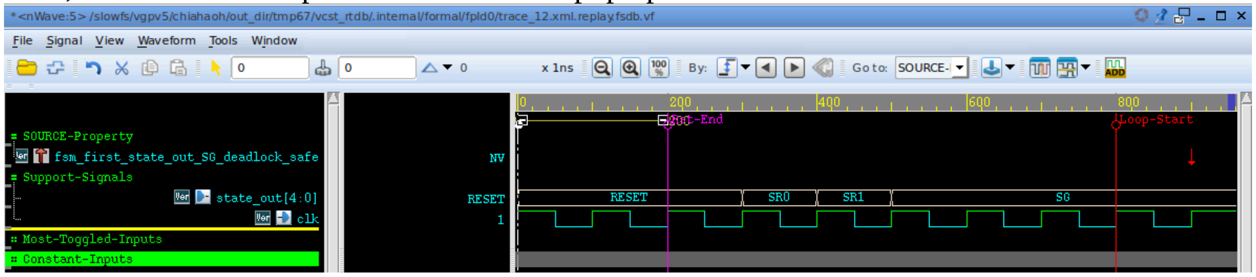
28. Debug failure:



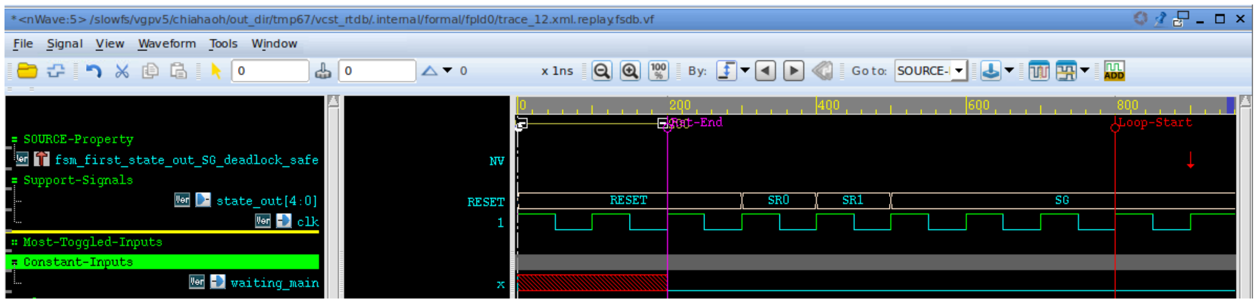
	status (V)	depth	name	engine	expression	elapsed_time	location	type (V)
1	✗	7	traffic.first.state_out_SG_deadlock_safe._prop	l1		00:01:09	...t_ctrl_fsm.v:32	fsm_deadlock
2	✗	4	traffic.first.state_out_SR1_deadlock_safe._prop	l1		00:00:11	...t_ctrl_fsm.v:32	fsm_deadlock
3	✗	4	traffic.main.state_out_SG_deadlock_safe._prop	l1		00:00:13	...t_ctrl_fsm.v:32	fsm_deadlock
4	✗	7	traffic.main.state_out_SR1_deadlock_safe._prop	l1		00:00:13	...t_ctrl_fsm.v:32	fsm_deadlock

Double click on the status ✗ of property "traffic.first.state_out_SG_deadlock_safe._prop"

, the counter example waveform would pop up


*<nWave:5> /slowfs/vgpv5/chiahaoh/out_dir/tmp67/vcst_rtdb/.internal/formal/fpld0/trace_12.xml.replay.fsdb.vf
File Signal View Waveform Tools Window
SOURCE-Property
fsm_first_state_out_SG_deadlock_safe
Support-Signals
state_out[4:0]
clk
Most-Toggled-Inputs
Constant-Inputs
NV
RESET
1
200
400
600
800
Loop-End
Loop-Start
RESET
SR0
SR1
SG

29. To further review the inputs of this FSM, to find that the reason for this deadlock issue is "waiting_main" never goes high the FSM would get stuck at "SG" state


*<nWave:5> /slowfs/vgpv5/chiahaoh/out_dir/tmp67/vcst_rtdb/.internal/formal/fpld0/trace_12.xml.replay.fsdb.vf
File Signal View Waveform Tools Window
SOURCE-Property
fsm_first_state_out_SG_deadlock_safe
Support-Signals
state_out[4:0]
clk
Most-Toggled-Inputs
Constant-Inputs
waiting_main
NV
RESET
1
x
200
400
600
800
Loop-End
Loop-Start
RESET
SR0
SR1
SG

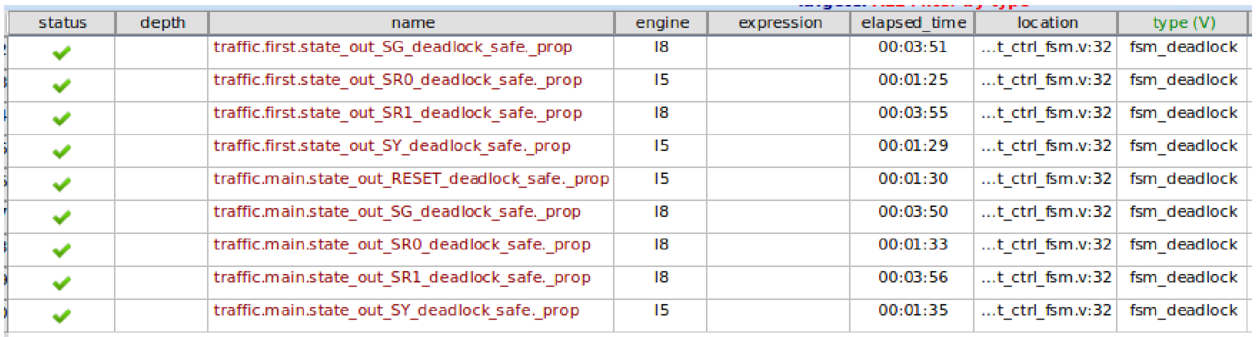
Add fairness Constraint for Inputs of FSM COI

30. To solve this fsm_deadlock falsified result, we need to add fairness constraints to model the inputs behavior by the following command to assume these inputs should have a change to toggle eventually, invoke it in the console:

```
    vcf> aep_generate -type fsm_fairness
```

We can also add it into the tcl file then click on to restart

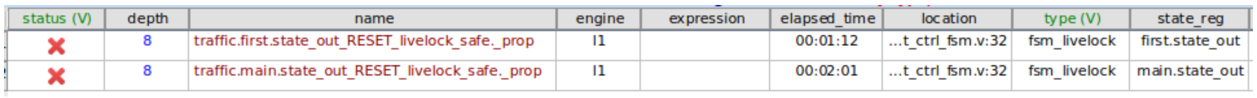
31. Then click on to rerun, the new result shows all the deadlock failures solved



status	depth	name	engine	expression	elapsed_time	location	type (V)
✓		traffic.first.state_out_SG_deadlock_safe._prop	l8		00:03:51	...t_ctrl_fsm.v:32	fsm_deadlock
✓		traffic.first.state_out_SR0_deadlock_safe._prop	l5		00:01:25	...t_ctrl_fsm.v:32	fsm_deadlock
✓		traffic.first.state_out_SR1_deadlock_safe._prop	l8		00:03:55	...t_ctrl_fsm.v:32	fsm_deadlock
✓		traffic.first.state_out_SY_deadlock_safe._prop	l5		00:01:29	...t_ctrl_fsm.v:32	fsm_deadlock
✓		traffic.main.state_out_RESET_deadlock_safe._prop	l5		00:01:30	...t_ctrl_fsm.v:32	fsm_deadlock
✓		traffic.main.state_out_SG_deadlock_safe._prop	l8		00:03:50	...t_ctrl_fsm.v:32	fsm_deadlock
✓		traffic.main.state_out_SR0_deadlock_safe._prop	l8		00:01:33	...t_ctrl_fsm.v:32	fsm_deadlock
✓		traffic.main.state_out_SR1_deadlock_safe._prop	l8		00:03:56	...t_ctrl_fsm.v:32	fsm_deadlock
✓		traffic.main.state_out_SY_deadlock_safe._prop	l5		00:01:35	...t_ctrl_fsm.v:32	fsm_deadlock

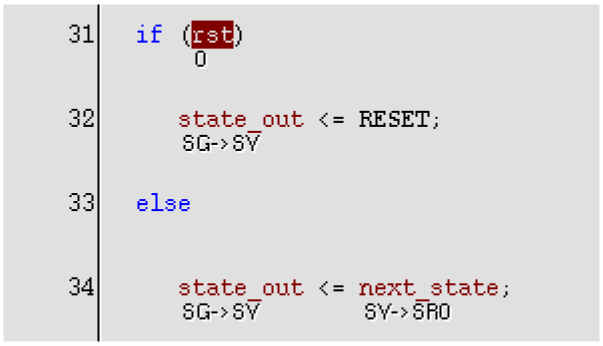
Debug fsm_livelock Failure

32. Debug these failures:



status (V)	depth	name	engine	expression	elapsed_time	location	type (V)	state_reg
✗	8	traffic.first.state_out_RESET_livelock_safe._prop	l1		00:01:12	...t_ctrl_fsm.v:32	fsm_livelock	first.state_out
✗	8	traffic.main.state_out_RESET_livelock_safe._prop	l1		00:02:01	...t_ctrl_fsm.v:32	fsm_livelock	main.state_out

Double click on the name of property "traffic.first.state_out_RESET_livelock_safe._prop" , to cross probe to the FSM segment in the RTL, and find that this FSM could go to RESET state only as "rst" goes to 1'b1.



```
31   if (rst)
         0

32      state_out <= RESET;
        SG->SY

33   else


34      state_out <= next_state;
        SG->SY       SY->SR0
```

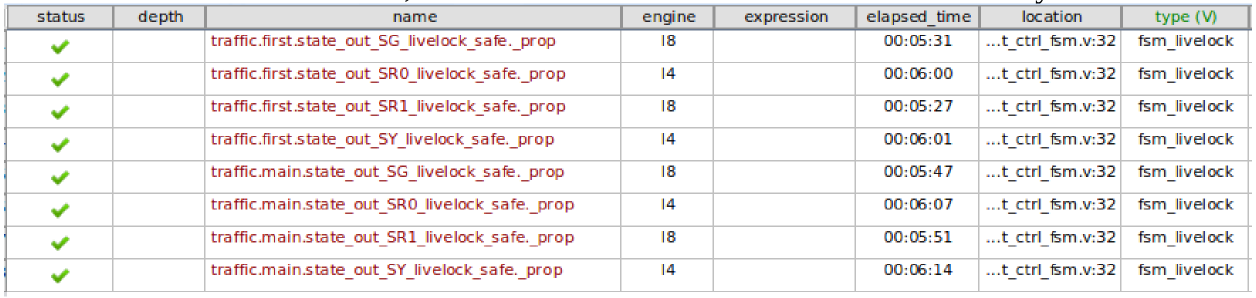
Since "rst" is already constrained by "create_reset" as an active high reset, so it would be kept as 1'b0 in functional phase.

Configure fsm_livelock Property Generation

33. After reviewing the design intent, if the conclusion is that it's meaningless to check livelock for RESET state, then we can configure AEP not to generate such property for it by inserting this command in the tcl, please note it has to be added before "read_file/analyze" command to get it work

```
     aep_config -dont_generate -type fsm_livelock \
{first.state_out:RESET main.state_out:RESET }
```

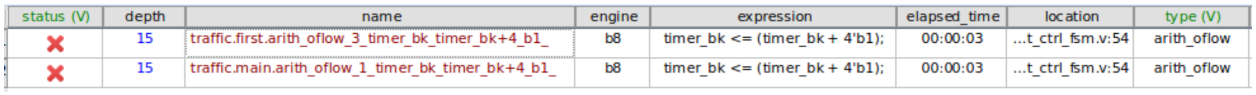
34. Once the tcl updated, click on to restart, it would show there is no property to check RESET state livelock now, and click to run and see there is no failure anymore



status	depth	name	engine	expression	elapsed_time	location	type (V)
✓		traffic.first.state_out_SG_livelock_safe._prop	I8		00:05:31	...t_ctrl_fsm.v:32	fsm_livelock
✓		traffic.first.state_out_SR0_livelock_safe._prop	I4		00:06:00	...t_ctrl_fsm.v:32	fsm_livelock
✓		traffic.first.state_out_SR1_livelock_safe._prop	I8		00:05:27	...t_ctrl_fsm.v:32	fsm_livelock
✓		traffic.first.state_out_SY_livelock_safe._prop	I4		00:06:01	...t_ctrl_fsm.v:32	fsm_livelock
✓		traffic.main.state_out_SG_livelock_safe._prop	I8		00:05:47	...t_ctrl_fsm.v:32	fsm_livelock
✓		traffic.main.state_out_SR0_livelock_safe._prop	I4		00:06:07	...t_ctrl_fsm.v:32	fsm_livelock
✓		traffic.main.state_out_SR1_livelock_safe._prop	I8		00:05:51	...t_ctrl_fsm.v:32	fsm_livelock
✓		traffic.main.state_out_SY_livelock_safe._prop	I4		00:06:14	...t_ctrl_fsm.v:32	fsm_livelock

Waive for confirmed falsified properties

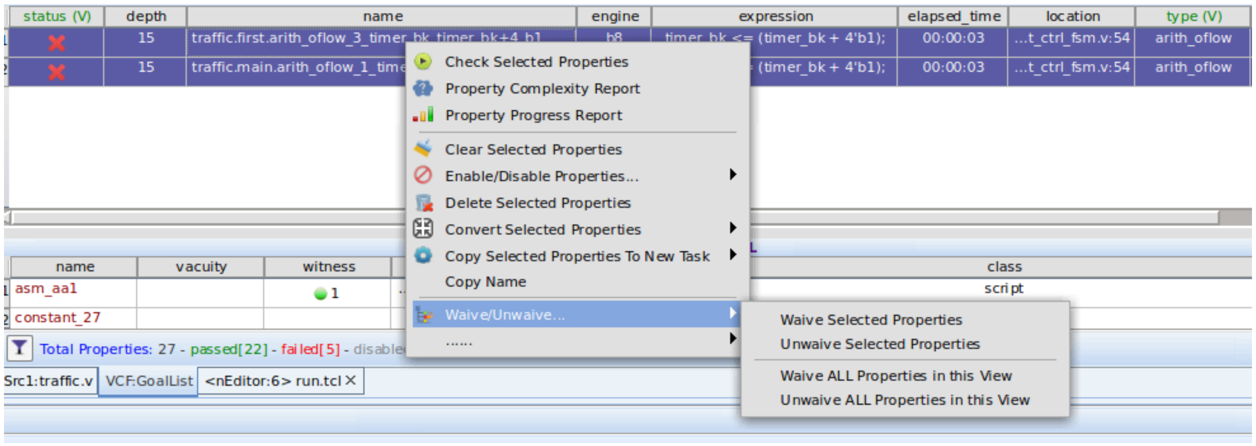
35. For below 2 falsified properties, it can be waived due to these are free run counters, so overflow is inevitable.



status (V)	depth	name	engine	expression	elapsed_time	location	type (V)
✗	15	traffic.first.arith_oflow_3_timer_bk_timer_bk+4_b1_	b8	timer_bk <= (timer_bk + 4'b1);	00:00:03	...t_ctrl_fsm.v:54	arith_oflow
✗	15	traffic.main.arith_oflow_1_timer_bk_timer_bk+4_b1_	b8	timer_bk <= (timer_bk + 4'b1);	00:00:03	...t_ctrl_fsm.v:54	arith_oflow

36. How to waive such falsified properties that user has confirmed?

    In GUI, choose property and right click => choose "Waive/Unwaive" => choose "Waive Selected Properties"


Check Selected Properties
Property Complexity Report
Property Progress Report
Clear Selected Properties
Enable/Disable Properties...
Delete Selected Properties
Convert Selected Properties
Copy Selected Properties To New Task
Copy Name
Waive/Unwaive...
......
Waive Selected Properties
Unwaive Selected Properties
Waive ALL Properties in this View
Unwaive ALL Properties in this View
Total Properties: 27 - passed[22] - failed[5] - disable
Src1:traffic.v
VCF:GoalList
<nEditor:6> run.tcl

status (V)	depth	name	engine	expression	elapsed_time	location	type (V)
✗	15	traffic.first.arith_oflow_3_timer_bk_timer_bk+4_b1	b8	timer_bk <= (timer_bk + 4'b1);	00:00:03	...t_ctrl_fsm.v:54	arith_oflow
✗	15	traffic.main.arith_oflow_1_time		(timer_bk + 4'b1);	00:00:03	...t_ctrl_fsm.v:54	arith_oflow

name	vacuity	witness	class
asm_aa1		● 1	script
constant_27			

The waived properties were hidden after operation and they were also excluded in TaskList No.

Related TCL command:

```
  fvwaive { <prop_name1> <prop_name2> ……}
```

To save waiver file for reusing, use below commands:

```
  save_waiver_file -file <AEP_exclusion_file>.el
```

It can be read back next time in TCL script after compile design:

```
  read_file ……
  read_waiver_file -elfiles <AEP_exclusion_file>.el
```

Restart the Run and Verify Fix

37. Restart VC Formal with the modified assertion:

    Click on the Restart VCST icon .

38. Re-run property verification:

Click on the Start Check icon ▶.

Observe that there are no falsified properties and all are proven.

Save Session

39. 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.

40. Exit VC Formal:

Click on File → Exit

Restore Session

41. Start VC Formal using previous saved session:

```
    %vcf -restore -verdi
```

42. Review setup and results then exit:

Click on File → Exit

Mode 2: Start VC Formal in Interactive Non-Verdi GUI Mode

43. Invoke VC Formal without Verdi GUI:

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

44. Enter run check command in VC Formal Shell:

```
vcf> check_fv
```

45. 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}
```

46. Start the Verdi GUI from within the VC Formal Shell:

```
vcf> start_gui
```

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_gui` 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.

47. Exit VC Formal:

```
vcf> quit
```

Mode 3: Set Up and Run VC Formal in Batch (Regression) Mode

48. Copy Tcl file run.tcl to run_batch.tcl:

```
%cp run.tcl run_batch.tcl
```

49. Edit run_batch.tcl and add commands to run and save results:

```
check_fv -block
report_fv -list > results.txt
```

50. Add command to save session:

```
save_session -session batch_results
```

And this session can be restored back by using below command:

```
%vcf -restore -session batch_results
```

51. Save run_batch.tcl file and exit editor.

52. 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 command file and no need to add "quit" command in the end of Tcl file.