



Use, Analysis, and Debug of SystemVerilog Assertions



Agenda

- Introduction
- Source Code Tracing
- Assertion Checking
- Analyzing and Debugging
 - Waveform
 - Active Annotation
 - Property Result Table

Standards: The Life blood of Interoperability

- Standards mean (users/vendors)
 - Greater confidence
 - Less risk
- Accellera and IEEE SA allow (users/vendors) early involvement and rapid development
 - E.g. SystemVerilog VPI
 - Have been working jointly with Synopsys VCS team to enhance the VPI beyond the LRM for local variable access – plan to feedback to committees in the future
 - Need other simulators to quickly add support for VPI to accelerate adoption – **customers want interoperability!**

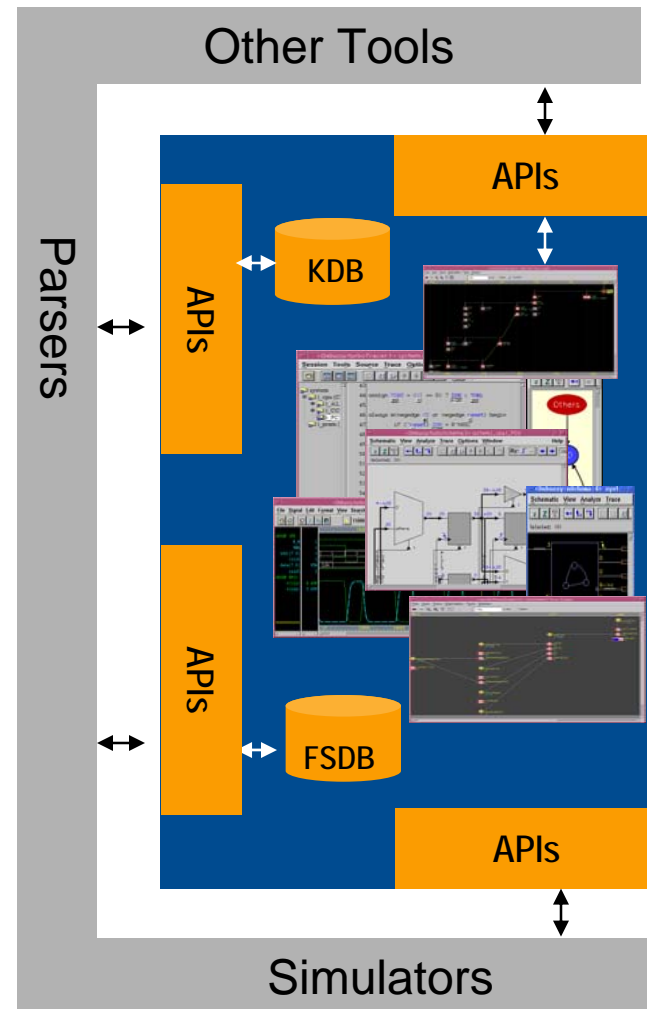


Novas is part of IEEE and Accellera SystemVerilog committees

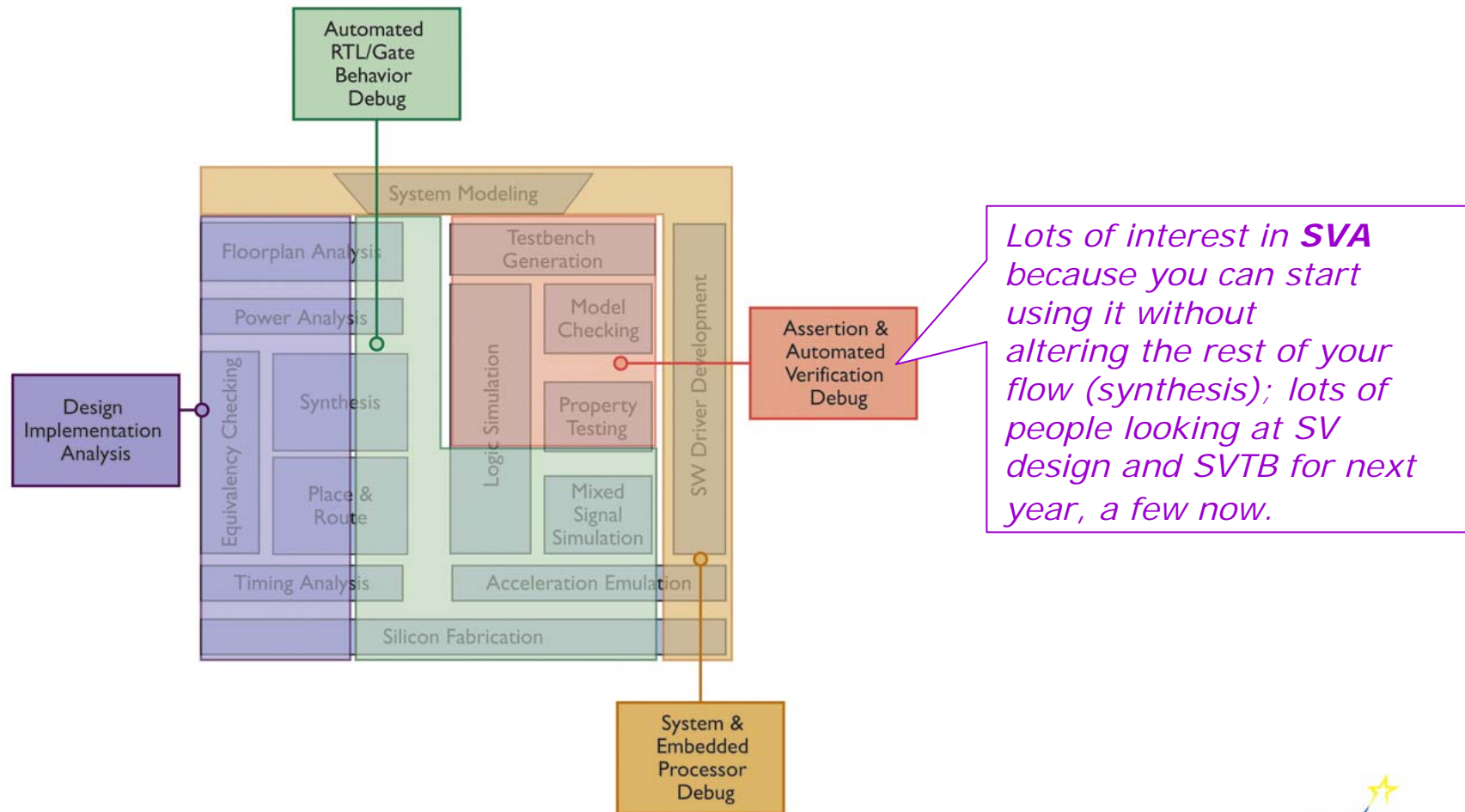


Novas' Open APIs

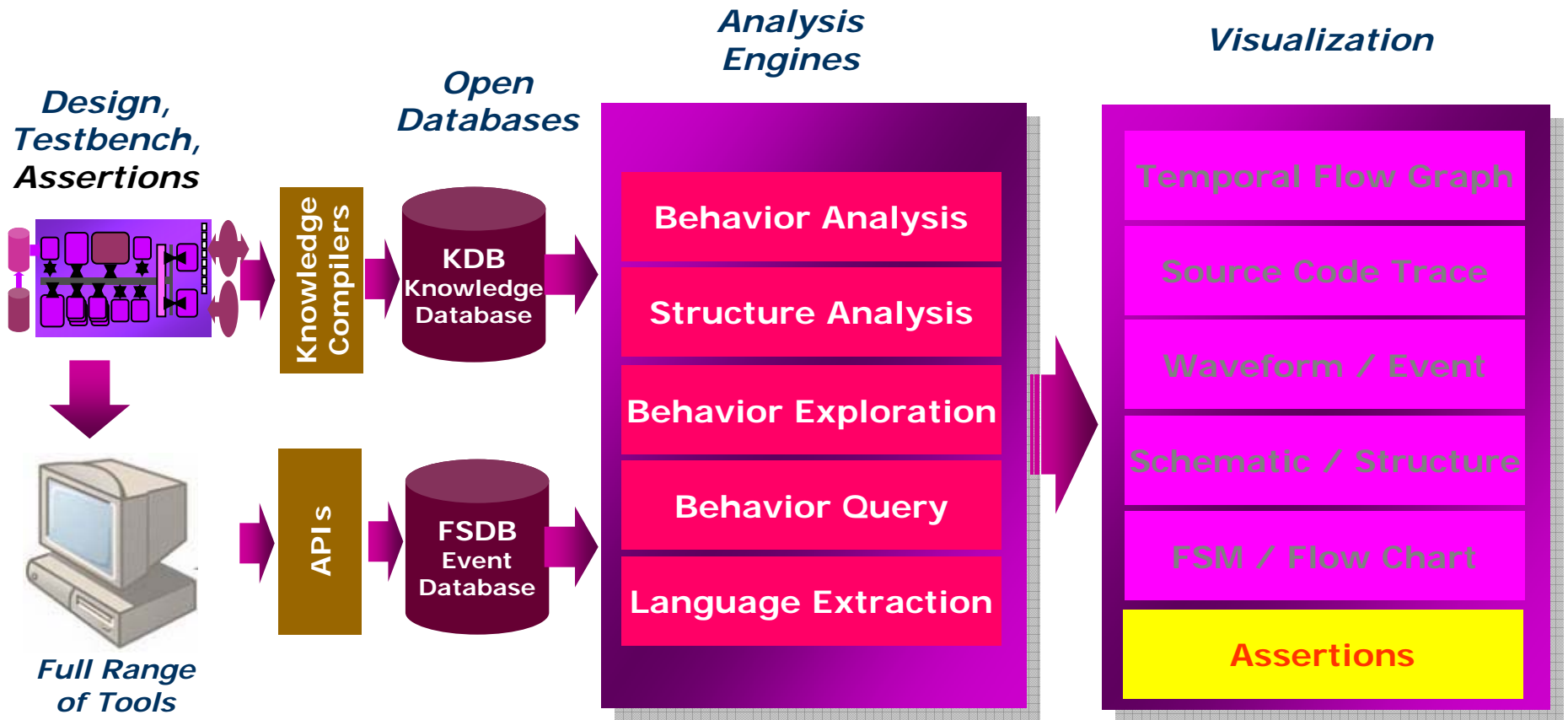
- Waveform database APIs
 - Writer for detection tools
 - Reader for analysis tools
- Design database APIs
 - Writer for adding data
 - Reader for tools
- Visualization tool control API
 - TCL functions for every command, and more
- Simulation control API
 - Debug system as simulator GUI



Debug & Analysis Requirements



Novas Technology Leadership



More Than Simple Visualization



Novas Platform for Assertion-Based Debug

- Wide Language Support – Available Now ...
 - OVA
 - PSL
 - **SVA**
- Source Code Support
 - Unified environment for tracing assertion code or between assertion and design
- Assertion Checking
 - Unique FSDB Checker Engine to check assertions in real-time without re-running simulation
 - Simulation – PLI library support
- Results Analysis
 - Enhanced Waveform for assertions
 - Active Annotation
 - Property Results Table –sorting and filtering of assertion data
 - Specialized treatment for Local Variables



Agenda

- Introduction
- Source Code Tracing
- Assertion Checking
- Analyzing and Debugging
 - Waveform
 - Active Annotation
 - Property Result Table

SVA Source Code – nTrace Hierarchy Browser

Hierarchy can be expanded/
collapsed by clicking on +/- sign
or double-clicking

Special icons for SVA asserts,
properties, sequences, covers

Assertion structure represented
hierarchically

The screenshot shows the nTrace Hierarchy Browser interface. The left pane displays a hierarchical tree of SVA assertions. The tree structure is as follows:

- system.i_cpu CPU (TopModule.v) - /home/.../SystemVerilog/sv.fsdB
 - INCPC
 - e_INC
 - e_l
 - e_r
 - COVER_e_r3
 - e_r3
 - INCPC2
 - e_INC2
 - e_l2
 - e_r2
 - LD
 - i_ALUB (ALUB)
 - 3 primitives
 - ALU_SUB
 - ALU_ZERO
 - OF_COVER
 - genblk[0]

The right pane shows the source code for the selected assertion, `INCPC`:

```
125 INCPC: assert property (e_INC);
126
127 // Example of local var
128 sequence e_l2;
129   @(negedge clock) (bus_mode == `INCA);
130 endsequence
131
132 sequence e_r2;
133   logic [7:0] ALU_prev;
134   @(negedge clock) (PC_load, ALU_prev = ALU) ##[1:2] (ALU == ALU_prev + 1);
135 endsequence
136
137 sequence e_r3;
138   logic [7:0] ALU_prev; // init to X
139   bit [7:0] flag; // init to 0 for 2-state
140   @(negedge clock) (PC_load, ALU_prev = ALU) ##[1:2] (ALU == ALU_prev + 1,
141 endsequence
142
143 COVER_e_r3: cover property (e_r3);
144
```

At the bottom of the window, the following text is displayed:

Release 5.4v9p5 (Linux) 07/05/2005 Base Technology 5.4v10_RD
Copyright (C) 1996 - 2005 by Novas Software, Inc.

How about a *global picture* of **all** assertions in design ...



Property Management Window

All assertions in design with *Filtering and Sorting*

Property Management - System Verilog Mode

Property Language: SVA

Filters:

- Type: *
- Name: *
- Module: *
- Instance: *

Ignore Case
 Use RegExp
 Filter Current

Reset Apply

	Type	Name	Module	Instance
<input checked="" type="checkbox"/>	assert	INCPC	CPU	system_i_cpu
<input checked="" type="checkbox"/>	cover	COVER_e_r3	CPU	system_i_cpu
<input checked="" type="checkbox"/>	assert	INCPC2	CPU	system_i_cpu
<input checked="" type="checkbox"/>	assert	LD	CPU	system_i_cpu
<input checked="" type="checkbox"/>	assert	ALU_SUB	ALUB	system_i_cpu.i_ALUB
<input checked="" type="checkbox"/>	assert	ALU_ZERO	ALUB	system_i_cpu.i_ALUB
<input checked="" type="checkbox"/>	cover	OF_COVER	ALUB	system_i_cpu.i_ALUB
<input checked="" type="checkbox"/>	assert	fetch	CCU	system_i_cpu.i_CCU
<input checked="" type="checkbox"/>	cover	cv	ALUB	system_i_cpu.i_ALUB.genblk[0]
<input checked="" type="checkbox"/>	cover	cv	ALUB	system_i_cpu.i_ALUB.genblk[1]
<input checked="" type="checkbox"/>	cover	cv	ALUB	system_i_cpu.i_ALUB.genblk[2]
<input checked="" type="checkbox"/>	cover	cv	ALUB	system_i_cpu.i_ALUB.genblk[3]
<input checked="" type="checkbox"/>	cover	cv	ALUB	system_i_cpu.i_ALUB.genblk[4]
<input checked="" type="checkbox"/>	cover	cv	ALUB	system_i_cpu.i_ALUB.genblk[5]

Sorted By: Instance Ascend Descend More...

Enable Disable Enable All Disable All

Total: 16 Enabled: 16 Disabled: 0

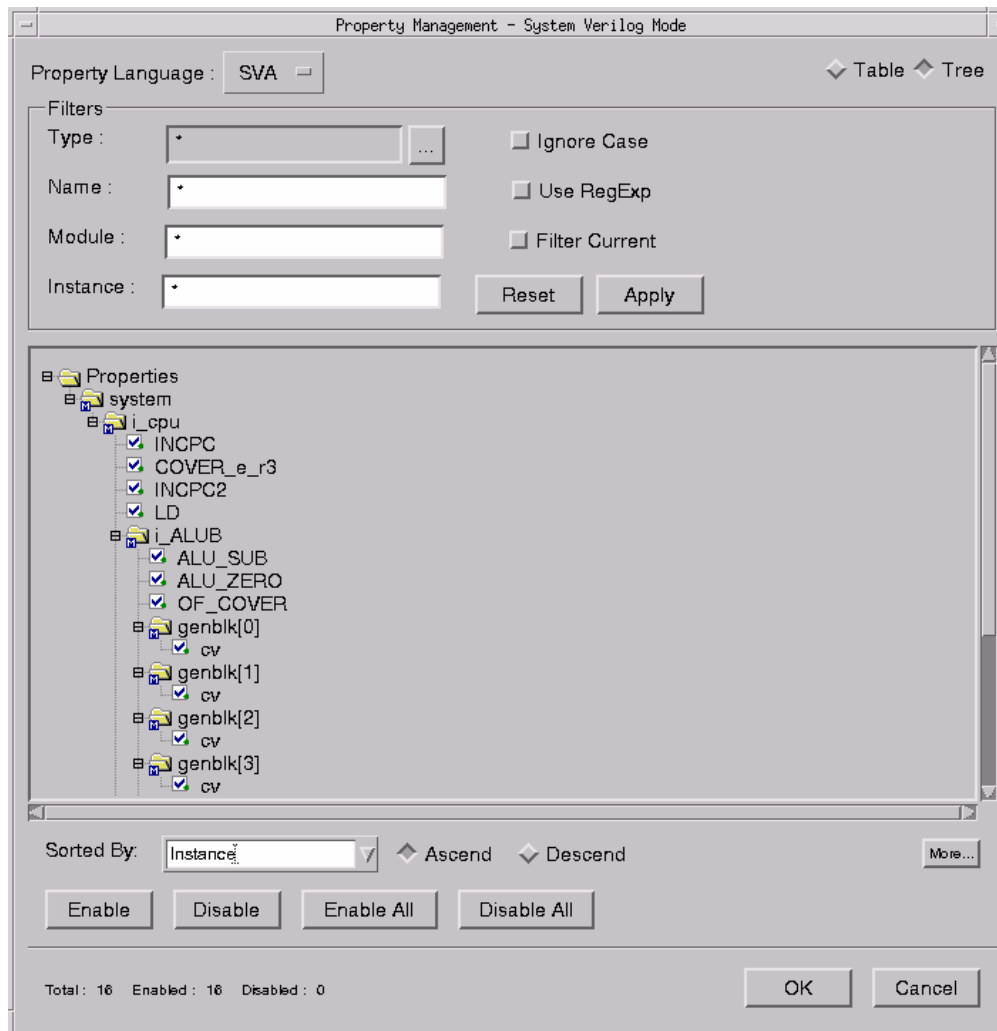
OK Cancel

Default is table view

D&D from table to source code or waveform



Property Management Window – Tree View



SVA Source Code Tracing (I)

Focus on an assertion (or property or sequence) by DC'ing in HB or D&D from some other view

```
<Verdi:nTraceMain:1> system.i_cpu CPU (TopModule.v) - /home/.../SystemVerilog/sv.fsdb
File Exploration View Source Trace Tools Window Help
121 property e_INC;
122 @(<negedge clock> e_l |-> e_r;
123     property
124
125 INCPC: assert property (e_INC);
126
127 // Example of local var
128 sequence e_l2;
129 @(<negedge clock> (bus_mode == `INCA);
130 endsequence
131
132 sequence e_r2;
133     logic [7:0] ALU_prev;
134 @(<negedge clock> (PC_load, ALU_prev = ALU) ##[1:2] (ALU == ALU_prev + 1);
135 endsequence
136
137 sequence e_r3;
138     logic [7:0] ALU_prev; // init to X
139     bit [7:0] flag; // init to 0 for 2-state
140 @(<negedge clock> (PC_load, ALU_prev = ALU) ##[1:2] (ALU == ALU_prev + 1, flag = 1);
141 endsequence
142
```

Release 5.4v9p5 (Linux) 07/05/2005 Base Technology 5.4v10_RD
Copyright (C) 1996 - 2005 by Novas Software, Inc.

SVA Source Code Tracing (II)

```
<Verdi:nTraceMain:1> system.i_cpu CPU (TopModule.v) - /home/.../SystemVerilog/sv.fsdb
File Exploration View Source Trace Tools Window Help
MASTER (fsm_master)
├── i_cpu (CPU)
│   ├── INCPC
│   │   ├── e_INC
│   │   │   ├── e_l
│   │   │   └── e_r
│   │   ├── COVER_e_r3
│   │   │   ├── e_r3
│   │   └── INCPC2
│   │       ├── e_INC2
│   │       │   ├── e_l2
│   │       └── e_r2
│   ├── LD
│   ├── i_ALUB (ALUB)
│   │   ├── 3 primitives
│   │   ├── ALU_SUB
│   │   ├── ALU_ZERO
│   │   └── OF_COVER
121 property e_INC;
122   @(negedge clock) e_l |-> e_r;
123 endproperty
124
125 INCPC: assert property (e_INC);
126
127 // Example of local var
128 sequence e_l2;
129   @(negedge clock) (bus_mode == `INCA);
130 endsequence
131
132 sequence e_r2;
133   logic [7:0] ALU_prev;
134   @(negedge clock) (PC_load, ALU_prev = ALU) ##[1:
135 endsequence
136
137 sequence e_r3;
138   logic [7:0] ALU_prev; // init to X
139   bit [7:0] flag;      // init to 0 for 2-state
```

```
<Verdi:nTraceMain:1> system.i_cpu.INCPC.e_INC e_INC (TopModule.v) - /home/.../SystemVerilog/sv.fsdb
File Exploration View Source Trace Tools Window Help
MASTER (fsm_master)
├── i_cpu (CPU)
│   ├── INCPC
│   │   ├── e_INC
│   │   │   ├── e_l
│   │   │   └── e_r
│   │   ├── COVER_e_r3
│   │   │   ├── e_r3
│   │   └── INCPC2
│   │       ├── e_INC2
│   │       │   ├── e_l2
│   │       └── e_r2
│   ├── LD
│   ├── i_ALUB (ALUB)
│   │   ├── 3 primitives
│   │   ├── ALU_SUB
│   │   ├── ALU_ZERO
│   │   └── OF_COVER
120
121 property e_INC;
122   @(negedge clock) e_l |-> e_r;
123 endproperty
124
125 INCPC: assert property (e_INC);
126
127 // Example of local var
128 sequence e_l2;
129   @(negedge clock) (bus_mode == `INCA);
130 endsequence
131
132 sequence e_r2;
133   logic [7:0] ALU_prev;
134   @(negedge clock) (PC_load, ALU_prev = ALU) ##[1:2] (ALU == ALU_prev +
135 endsequence
136
137 sequence e_r3;
138   logic [7:0] ALU_prev; // init to X
139   bit [7:0] flag;      // init to 0 for 2-state
```

Double-click on a property or sequence reference to go (change scope) to its 'definition'

SVA Source Code Tracing (III)

Design Variables referenced inside properties and sequences can be traced “as usual”

Trace Drivers & Loads

The screenshot displays the SVA source code tracing interface. The source code editor shows the following code:

```
115 endsequence
116
117 sequence e_r;
118 @(negedge clock) (`TRUE) ##[1:2] ((ALU) := $past(ALU))
119 endsequence
120
121 property e_INC;
122 @(negedge clock) e_l l-> e_r;
123 endproperty
124
125 INCPC: assert property (e_INC);
126
127 // Example of local var
```

The trace output window shows the following results:

```
1> system.i_cpu.ALUE7:01 /* results of trace driver */
<ID> TopModule,v(85); , .IXR);
system.i_cpu : 1 driver pass-through(s)
<ID> ALUB,v(89); ,out(ALU) ,
system.i_cpu.i_ALUB : 1 driver pass-through(s)
*ID> ALUB,v(233): ADD: {carry,out} = op,a + op,b + op,cin;
*ID> ALUB,v(234): SUB: {carry,out} = op,a - op,b - 1 + op,cin;
*ID> ALUB,v(235): SUB1: {carry,out} = op,b - op,a - 1 + op,cin;
```

Agenda

- Introduction
- Source Code Tracing
- Assertion Checking
- Analyzing and Debugging
 - Waveform
 - Active Annotation
 - Property Result Table

Assertion Checking

The screenshot displays the VeriTraceMain interface for assertion checking. The top window shows the source code for the `CHECK_SUB` property:

```

91 .zero(zero));
92
93 //SVA: Example of vacuous success
94 sequence e_SUB;
95   @(posedge system_clock) alu_mode == `SUB1;
96 endsequence
97 property CHECK_SUB;
98   @(posedge system_clock) e_SUB |-> <ALU==Y0-X0-1+carry_flag>;
99 endproperty
100
101 ALU_SUB: assert property <CHECK_SUB>;
102

```

The `Assertion Check on FSDB` dialog box is open, showing configuration options:

- Don't record success/match
- Don't filter trivial success/match
- Don't filter 'cover' failures
- Bring up waveform after check
- Load result as aux annotation
- Merge result into design trace
 - Use small range merge
- Flag potential race in sampling (PSL)

The `Save result in:` field is set to `imo_home/demos/core_demo/rtl/vdac_result.fsdb`. The `Check` button is highlighted.

A yellow box labeled `D&D` (Design and Debug) points to the `FSDB Checker...` option in the project tree and the waveform viewer. The waveform viewer shows a clock signal and a failure event.

The diagram on the right illustrates the workflow:

- Simulator** (green box) outputs results to **FSDB** (blue cylinder) via `$fsdbDumpSVA`, `$fsdbDumpPSL`, etc.
- FSDB** outputs results to **VeriTraceMain** (grey window).

FSDB Checker Support in Verdi – Why?

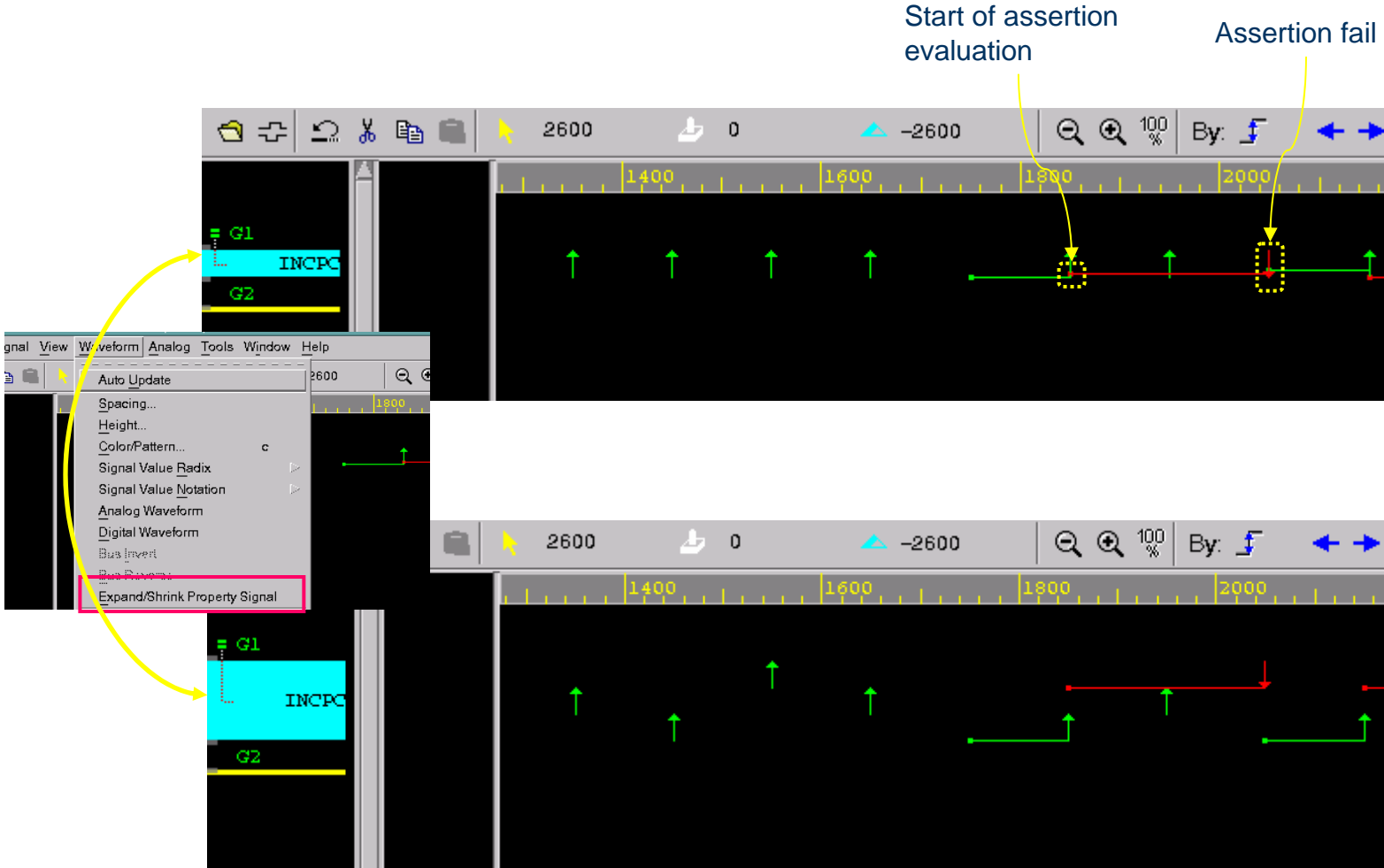
- Idea: Perform **Q&A with engineer** using assertion language. Verdi can check the assertion “questions”.
 - ✓ **Flexibility:** Quickly check the assertion no need to re-compile design and re-simulate.
 - ✓ **Speed:** Check low-level “implementation” assertions instead of slowing doing system simulation
 - ✓ **Power:** Has complete FSDB time, no history (future) limitations
 - ✓ **Detail:** Compute all the dependent events and sequences for easy debug
 - ✗ **Trace data:** Need to pre-generate design trace data for analysis
- Assertion Language Support
 - OVA
 - OVA 1.3
 - PSL
 - PSL 1.01/1.1 “simulation subset”
 - SVA
 - SVA 3.1a



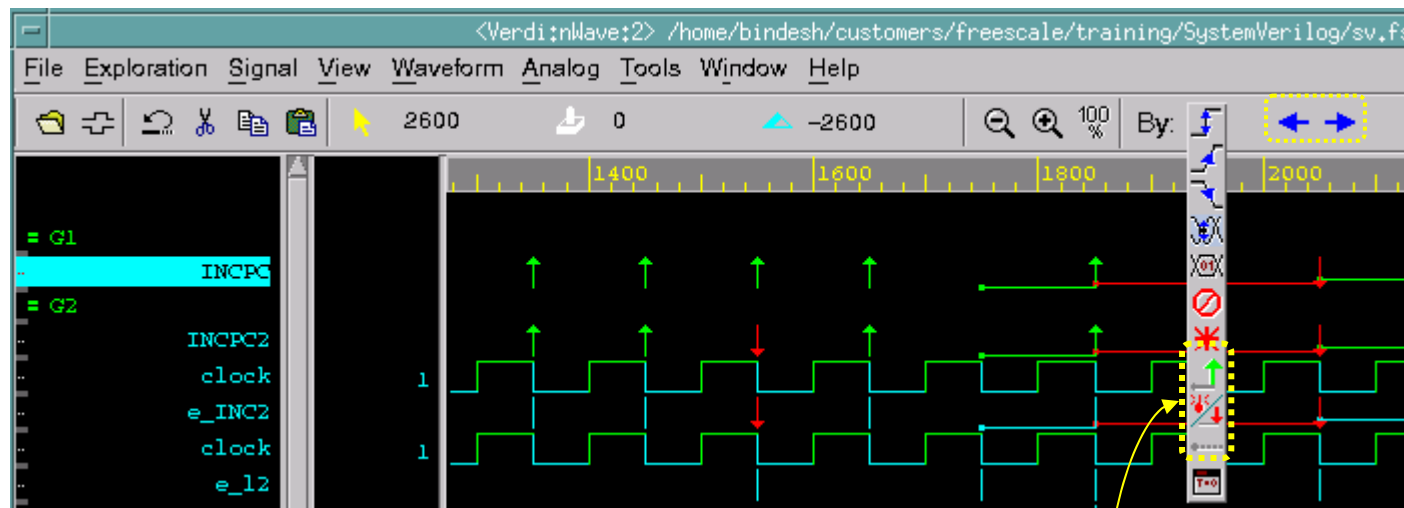
Agenda

- Introduction
- Source Code Tracing
- Assertion Checking
- Analyzing and Debugging
 - Waveform
 - Active Annotation
 - Property Result Table

Waveform Basics & Expanding/ Shrinking



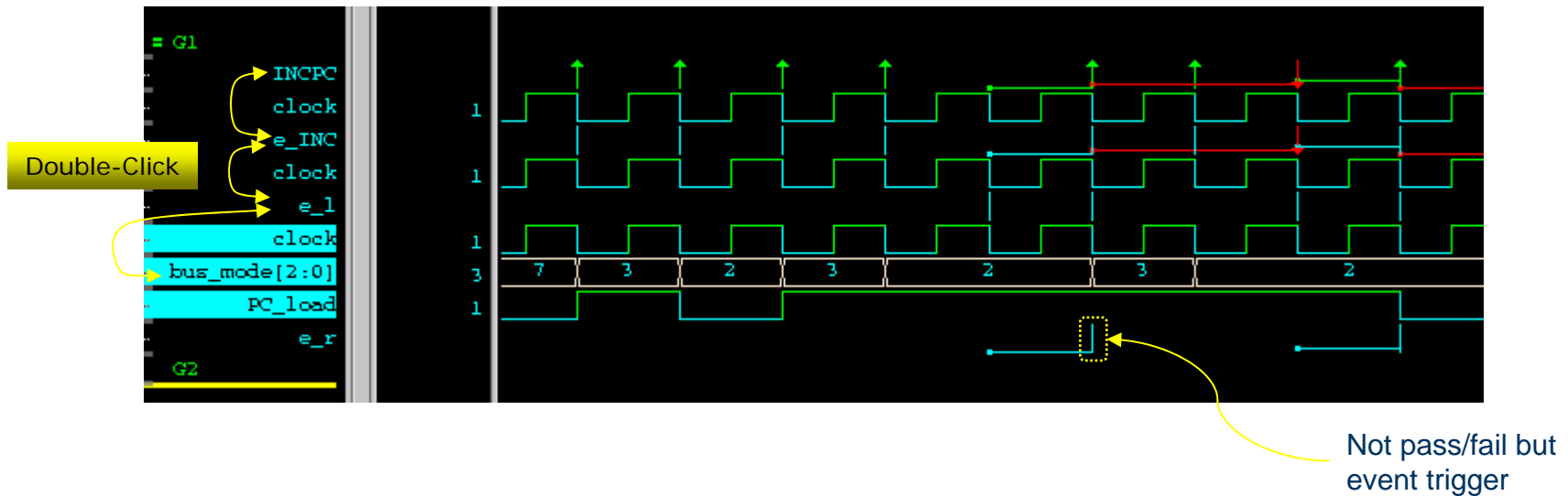
Searching



Search criteria: **start**,
fail, **success/match**

Associated Data Tagging

Underlying “building block” data **tagged** for easy double-click access ...



... all the way down to signals and local variables

Local Variables

```

130 endsequence
131
132 sequence e_r2:
133     logic [7:0] ALU_prev;
134     @(negedge clock) <PC_load, ALU_prev = ALU> ##[1:2] <ALU == ALU_prev + 1>;
135 endsequence
136
137 sequence e_r3:
138     logic [7:0] ALU_prev; // init to X
139     bit [7:0] flag; // init to 0 for 2-state
140     @(negedge clock) <PC_load, ALU_prev = ALU> ##[1:2] <ALU == ALU_prev + 1, flag = 1>;
141 endsequence
142

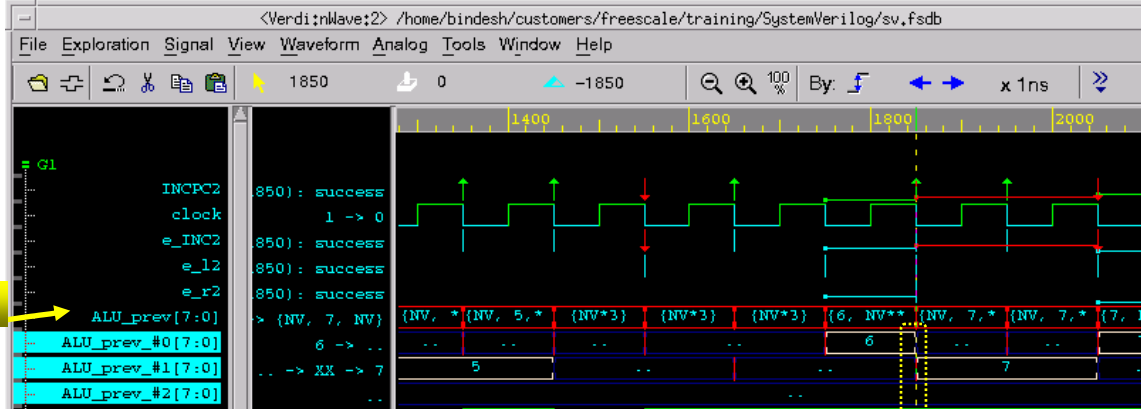
```

Have been working jointly with Synopsys to enhance the VPI beyond the LRM for local variable access from VCS – plan to feedback to Accellera & IEEE committees in the future

```

logic [7:0] ALU_prev;
{6, NV*2}->{NV, XX, NV}->{NV, 7, NV}

```



Double-Click

Multiple attempts



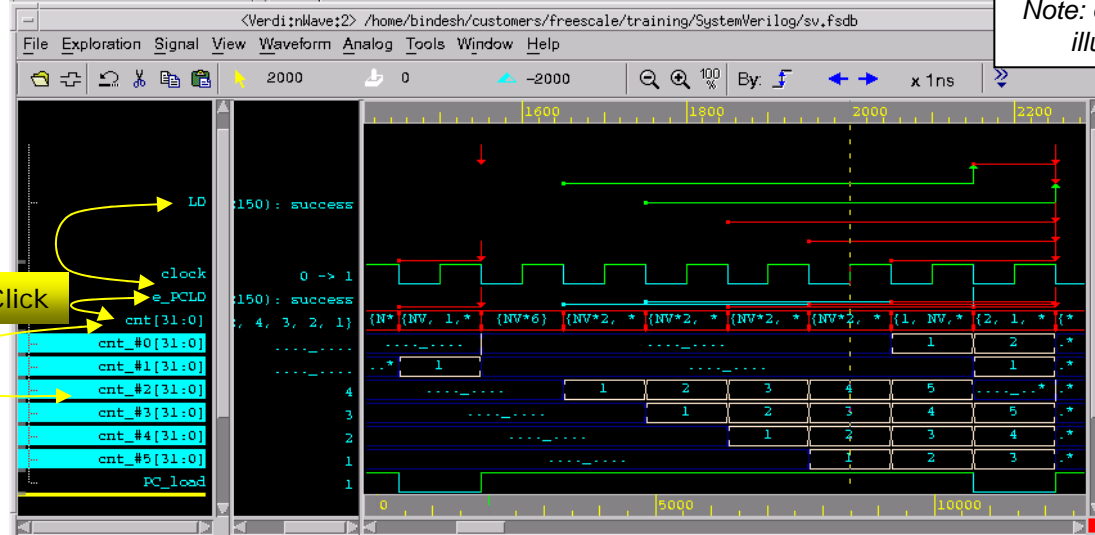
Local Variables – Another Example

UE = Under Evaluation
 SE = Start of Evaluation
 NV = No Value
 NF = Not Found
 not in FSDB file (not in dumped scope, or (see gotchas...))

```

151 property e_PCLD;
152   UE
153   int cnt;
154   @(negedge clock) (<TRUE, cnt = 0)
155   ##0
156   <PC_load, cnt = cnt + 1>[*5]
157   ##1
158   <cnt == 5>;
159   endproperty
160 LD: assert property (<e_PCLD>;
161   UE
162 endmodule
  
```

“Make sure PC_load stays high for 5 consecutive cycles”



Note: example is not real-life but illustrates the key points



Multiple Threads

Why? Cover of a sequence does not do first_match like property/assert, it tracks all matches

The screenshot shows the Verdi tool interface. The top window displays the source code for a sequence cover:

```

137 sequence e_r3:
138   logic [7:0] ALU_prev; // init to X
139   bit [7:0] flag; // init to 0 for 2-state
140   @(negedge clock) (PC_load, ALU_prev == ALU) ##[1:2] (ALU == ALU_prev + 1, flag = 1);
141 endsequence
142
143 COVER_e_r3: cover property (e_r3);
144
145 property e INC2:
    NF
  
```

The bottom window shows a waveform for the sequence cover. The signal 'e_r3' is shown as a pulse at time 2050. The waveform also shows the values of 'ALU_prev', 'flag', and 'PC_load' over time. A yellow box highlights the 'flag' signal, and a red box highlights the 'Set of Set Notation' text.

Note: example is not real-life but illustrates the key points

Assign to the local var after the branching results in 2 threads

flag = 1);
 {{0, 0}, {0, 0}, {NV*2}}->{{1, 0}, {NV*2}, {0, 0}}

Set of Set Notation – each attempt has 2 branches or threads



'generate'

```
<Verdi:nTraceMain:1> system.i_cpu.i_ALUB.genblk[2] genblk[2] (ALUB.v) - /home/.../SystemVerilog/sv.  
File Exploration View Source Trace Tools Window Help  
By: f 1000 x 1ns  
LD 109 // SVA generate  
e_PCLD 110  
i_ALUB (ALUB) 111 genvar i;  
3 primitives 112  
ALU_SUB 113 generate  
ALU_ZERO 114  
CHECK_ALU_ 115 for (i = 0; i < 8; i = i + 1)  
OF_COVER 116 begin : genblk  
add_overflow 117 cv: cover property(@(posedge T2) (ALU[i] == 1) );  
genblk[0] 118 end  
cv  
genblk[1]  
cv  
genblk[2]  
cv  
genblk[3]  
genblk[4]  
genblk[5]
```

Value for "Current" scope will be annotated

Enable computation of genvars,
Source → Parameter Annotation



Beyond Waveforms – View as a Detailed List ...

Sorting and

... Filtering

Property Result Table

Filters | **Sorting Criteria**

Filters

Status : Success Failure Unknown No Hit All

Type : ...

Property : Ignore Case

FSDB File : Use RegExp

Instance :

Instance	UnitName	Property	Type	StartTime	EndTime
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	0 x 1 ns	0 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	350 x 1 ns	350 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	450 x 1 ns	550 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	550 x 1 ns	750 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	650 x 1 ns	650 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	750 x 1 ns	850 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	850 x 1 ns	1050 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	950 x 1 ns	950 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1050 x 1 ns	1050 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1150 x 1 ns	1250 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1250 x 1 ns	1450 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1350 x 1 ns	1350 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1450 x 1 ns	1450 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1550 x 1 ns	1550 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1650 x 1 ns	1650 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1750 x 1 ns	1850 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1850 x 1 ns	2050 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	1950 x 1 ns	1950 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	2050 x 1 ns	2150 x 1 ns
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	2150 x 1 ns	2350 x 1 ns

Hide Show All Save as Text ... Page : 1

View As : Coverage List

Options... Close

Property Result Table - Preference Setting

Options for the result form

1. Page size : records per page

2. Start result form with : Coverage List mode

3. Highlight record to sync with primary nWave :
 Enable Disable

4. Behavior when dropping properties on result form :
 Locate in Add to the result form

Default Apply OK Cancel



... Or Coverage

Property Result Table

Filters | Sorting Criteria

Filters

Status : Success / Failure / Unknown

Type : ...

Property : Ignore Case

FSDB File : Use RegExp

Instance :

Default

Apply

Instance	UnitName	Property	Type	FSDB	Success	Failure
system.i_cpu	i_cpu	INCPC	ASSERT_CHECK	sv.fsdb	96	26
system.i_cpu	i_cpu	INCPC2	ASSERT_CHECK	sv.fsdb	80	42
system.i_cpu.i_ALUB	i_ALUB	OF_COVER	COVER	sv.fsdb	2	0
system.i_cpu.i_ALUB.genblk[0]	genblk[0]	cv	COVER	sv.fsdb	3	0
system.i_cpu.i_ALUB.genblk[1]	genblk[1]	cv	COVER	sv.fsdb	5	0
system.i_cpu.i_ALUB.genblk[2]	genblk[2]	cv	COVER	sv.fsdb	4	0

Summary

- Standards are the Lifeblood for Interoperability
- Novas Provides Essential Support for Productive Analysis & Debug of SVA
 - Intelligent Assertion Source Code Tracing
 - Quick Assertion Checking
 - Advanced Visualization, Analysis, & Debug
- *Meeting* the [Debug] Challenges
 - Local Variables
 - Multiple Attempts
 - Multiple Threads

Novas Solutions → Verdi™ Automated Debug System

