Verdi^3 Fundamental Training

Based on Verdi3 2012.04

SpringSoft Notifications

Copyright

© 2012 SpringSoft, Inc. All rights reserved.

No part of this training may be reproduced in any form or by any means without written permission of SpringSoft, Inc.

Trademarks

The product names used in this document are the trademarks or registered trademarks of their respective owners.

Confidentiality

The information in this document is confidential and is covered by a license agreement between SpringSoft and your organization. Distribution and disclosure are restricted.

Prerequisites

- Basic HDL/HVL coding skills: Verilog, VHDL, SystemVerilog.
- Familiarity with standard simulators.

Glossary

- RMB = Right Mouse Button
- MMB = Middle Mouse Button
- LMB = Left Mouse Button
- DC = Double-click
- D&D = Drag and Drop
- KDB = Knowledge Database
- FSDB = Fast Signal Database
- TFV = Temporal Flow View
- BA = Behavior Analysis
- BDB = Behavior Database
- VERDI3_HOME = The install directory of *Verdi*^{3}

Objective

- After completing this training, you should be able to…
  - Understand the technology of the Verdi platform
  - Setup the environment for debugging
  - Use tasks and utilities to dump FSDB
  - Import Design into the Verdi platform
  - Debug in Source Code View
  - Debug in Waveform View
  - Debug in Schematic View
  - Debug in FSM view
  - Debug in Temporal Flow View

Overview

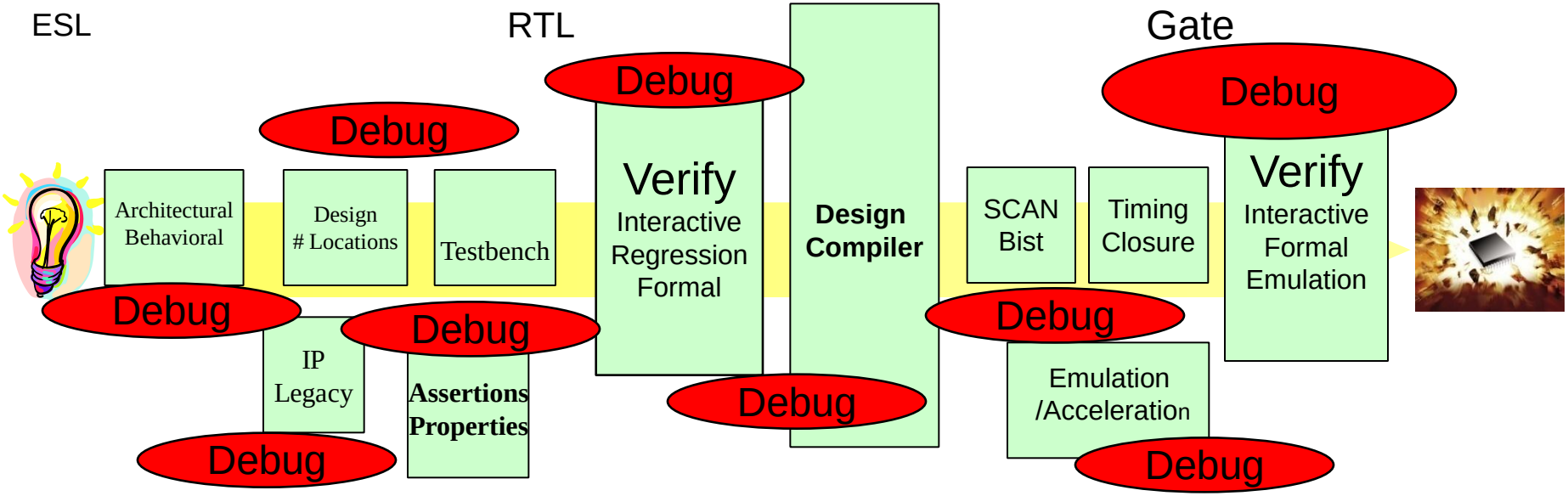
- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM view
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

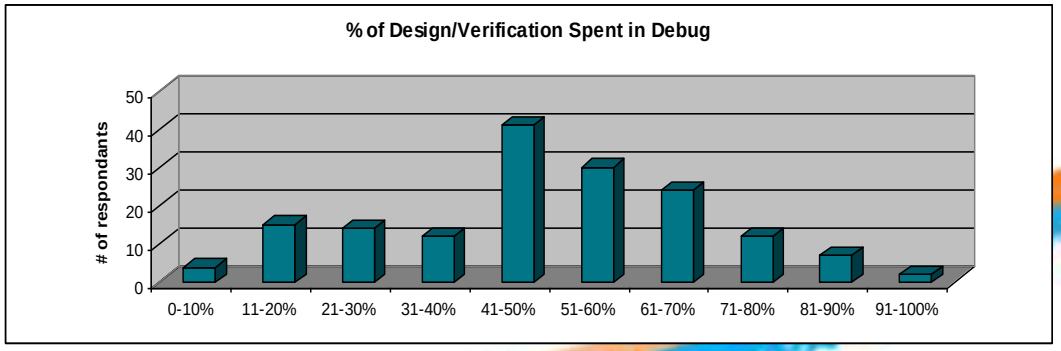
Technology Background

Debug in the Design Flow



Do you know how much time your teams spend debugging?

What else could you be doing with that time?

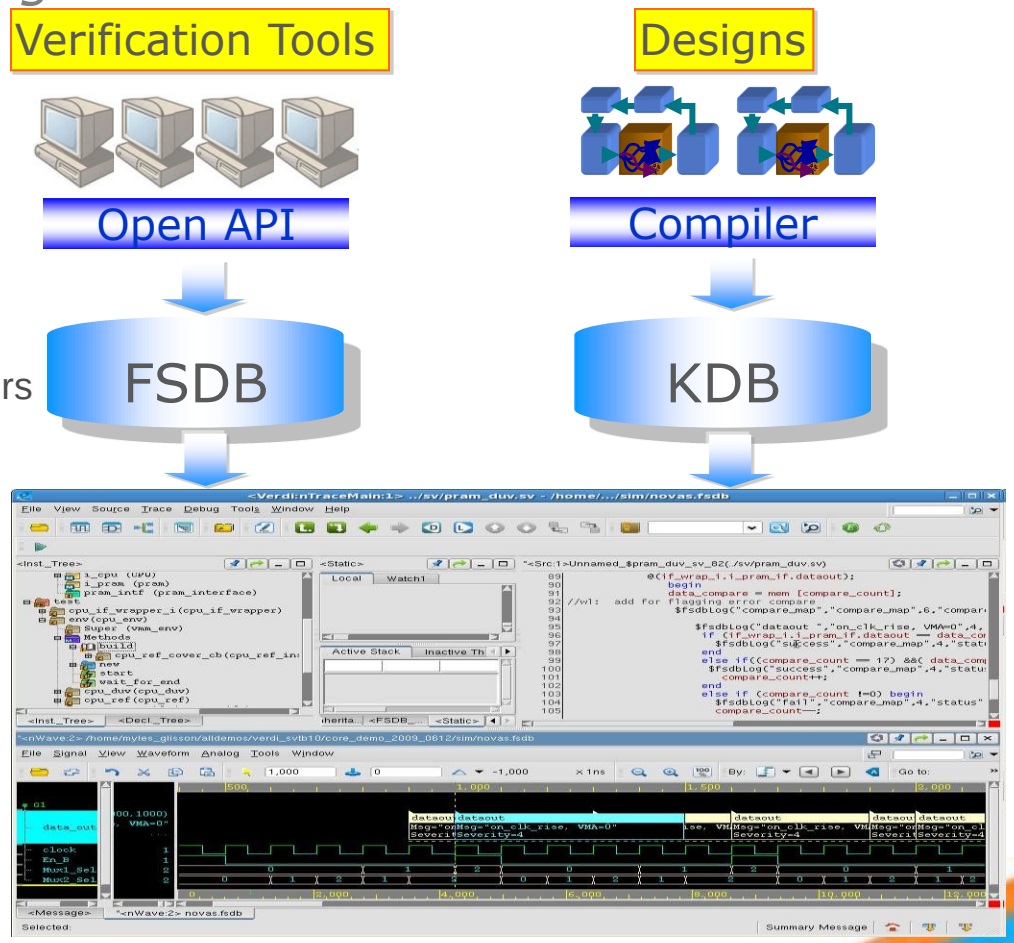


Source: Novas Customer Survey, 161 Respondents

Technology Background

Verdi – Behavior Based Debug Automation

- Automated cause & effect tracing
  - Save significant engineering time
  - Easy visualization of behavior with time and structure
- Complete, easy to use environment
  - Design knowledge activated viewers
    - Waveform, source code, schema, TFV
  - Advanced debugger
    - State, testbench, assertions, transactions
  - Power-aware
  - Clock analysis
  - Post-layout ECO
- Single portal for multiple languages, tools and methodologies
  - Verilog, VHDL, System Verilog
  - Open interface for verification tools



Verdi

SpringSoft
Accelerating Engineers

Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Specify the Path and Environment Variables

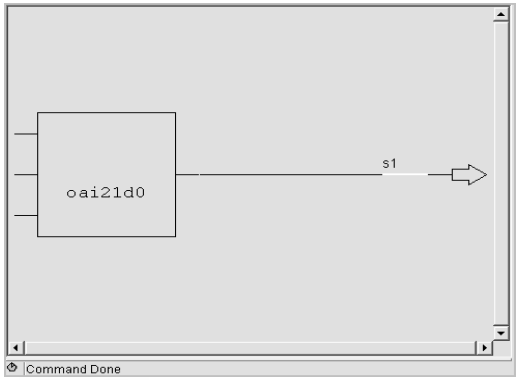
- Binary File
  - *setenv VERDI3_HOME <Path to Novas installation>*
  - *set path = ($VERDI3_HOME/bin $path)*

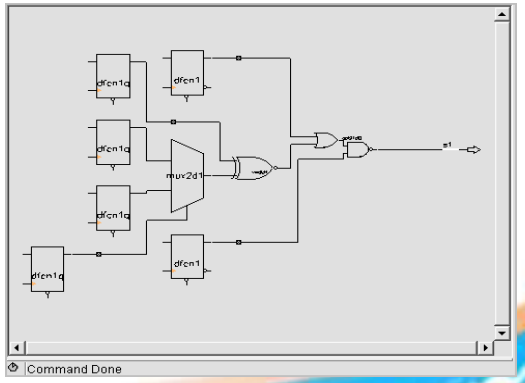
- License File
  - *setenv NOVAS_LICENSE_FILE <license_file>:$NOVAS_LICENSE_FILE*
  - Alternatively use *LM_LICENSE_FILE* or *SPS_LICENSE_FILE*
  - The search priority will be:
    - *SPS_LICENSE_FILE*
    - *NOVAS_LICENSE_FILE*
    - *LM_LICENSE_FILE*

Symbol Libraries Background

- Why use symbol libraries?
  - Important for gate-level designs
  - Provides mapping between source code and logic cell
    - For example: ND2P U235 (.A(n758), .B(n759), .Z(\Y0[6]));
      - What does the ND2P cell look like?
- Without symbol mappings
  - Active fan-in cone, fan-in cone and fan-out cone schematics will **NOT** function properly
  - Schematics will show **ONLY** squares instead of logic cells

Unmapped fan-in cone





Mapped fan-in cone

SpringSoft
Accelerating Engineers

Using Symbol Libraries

Import Novas Symbol Libraries

- Create a symbol library using one of the following methods:
  - **syn2SymDB**: For Synopsys logical library file in ACSII format (.lib)
  - **map2SymDB**: For the prepared map file
    - Check the **Library Developer's Guide** document from <VERDI3_HOME>/doc/Symbol_Library.pdf for detailed formats

- Set the symbol library environment variables

  setenv NOVAS_LIBS "<LIB ROOT#1> <LIB ROOT#2> ..."

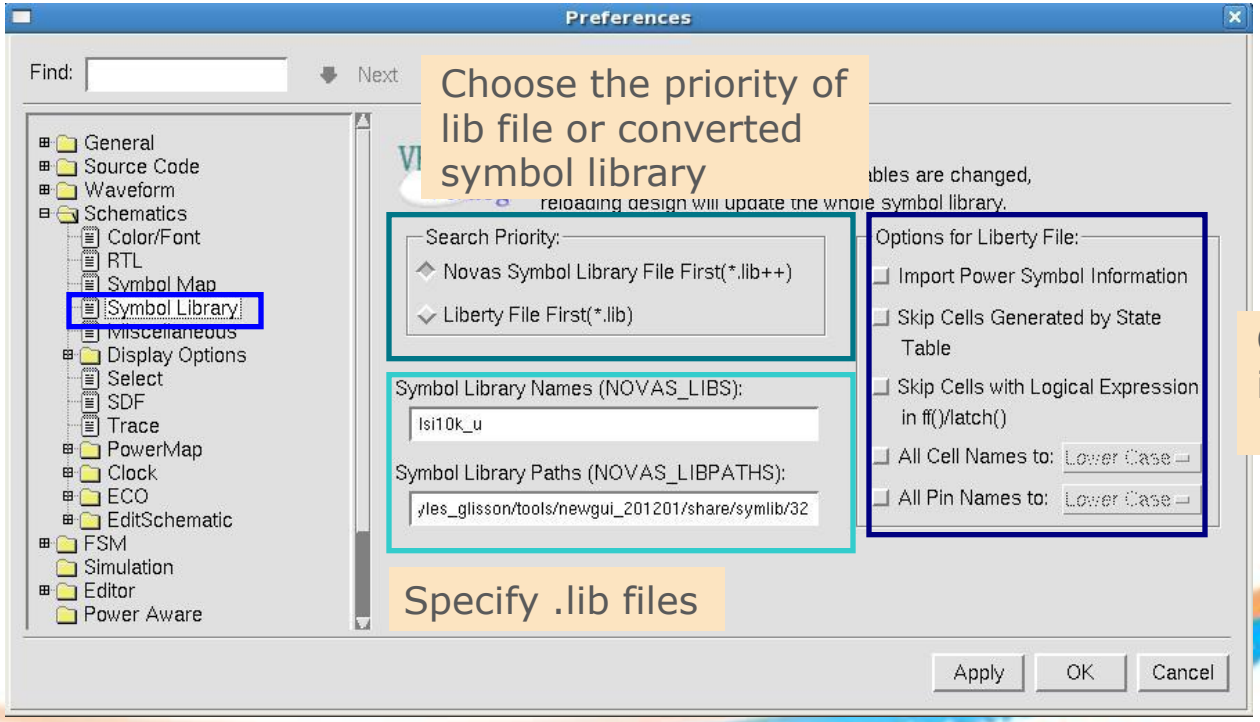
  setenv NOVAS_LIBPATHS "<Directory#1> <Directory#2> …"

- Set the symbol library through the GUI
  - Invoke **Tools → Preferences** command to open *Preferences* form
  - Select the folder **Schematics → Symbol Library**
  - Input the symbol library name in **Symbol Library Names** field
  - Input the path in **Symbol Library Paths** field
  - The setting will be saved into the *novas.rc* resource file

Using Symbol Libraries

Import Liberty File Directly

- The Liberty File (***.lib**) can also be imported directly without pre-compiling to Novas symbol libraries
  - From GUI: invoke **Tools → Preferences** and go to **Schematics → Symbol Library** page, select **Liberty File First** option
  - From command line: set the Liberty File to ***NOVAS_LIBS*** and ***NOVAS_LIBPATHS*** environment variables



Choose the priority of lib file or converted symbol library

Constraints for importing the .lib file

Specify .lib files

SpringSoft Accelerating Engineers

Use Setup File – novas.rc

- Contains:
  - Mapping information for pre-compiled designs.
    - Map a library logical name to a physical location
    - Mapping format

```
[Library]
    logical name =  physical location ( ex, pack = ./work)
    pack = ../library/pack            ( ex, vital = ./work )
```

  - User preferences (set via **Tools → Preferences**).

- Specify Path
  - Using NOVAS_RC environment variable

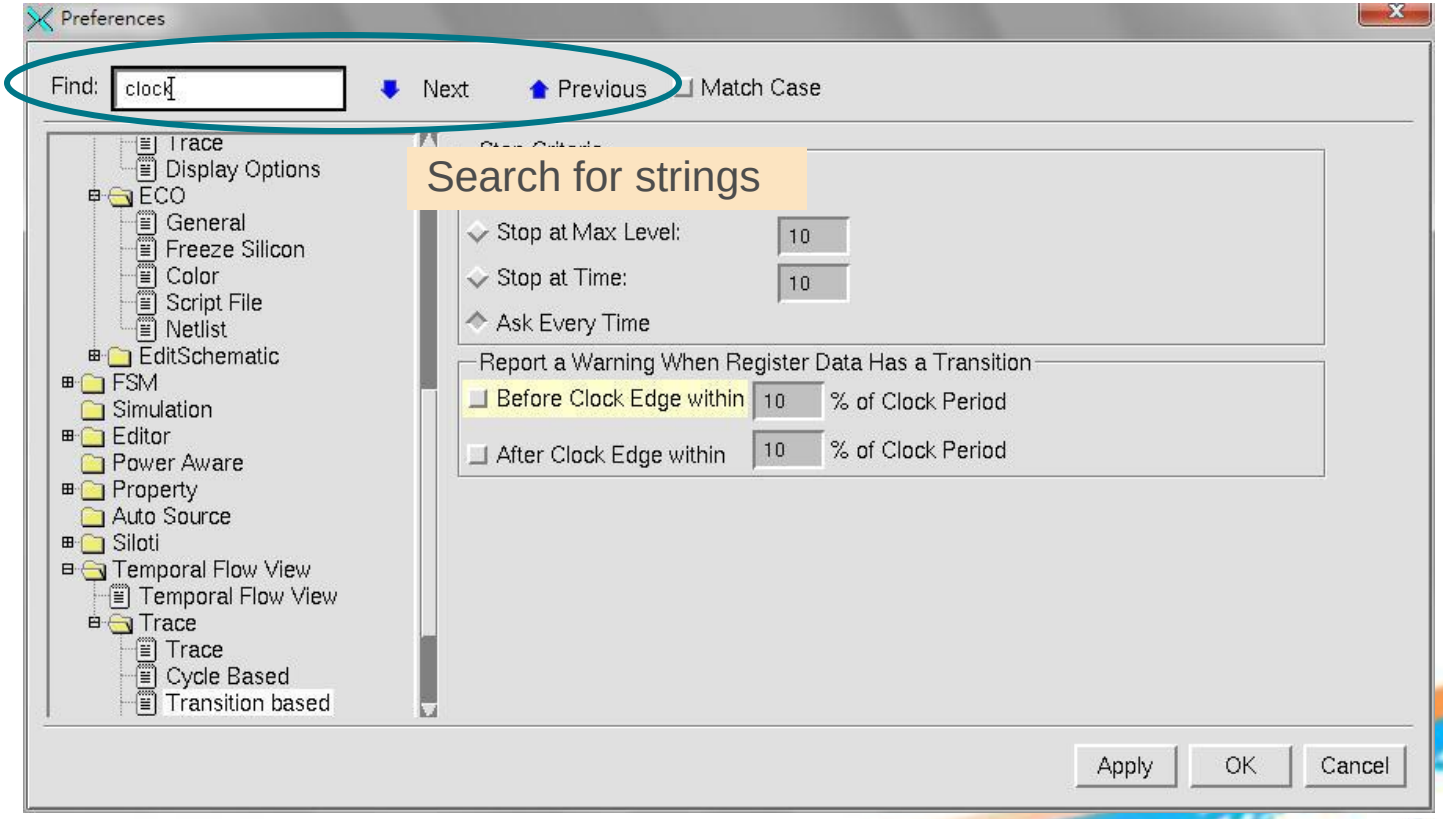
```
setenv NOVAS_RC <path>/novas.rc
```

  - Directly by **-rcFile** <*filename*> on the command line

- Search Sequence
  1. **-rcFile** <*filename*> command line option
  2. NOVAS_RC environment variable
  3. ./novas.rc
  4. $HOME/novas.rc
  5. <VERDI3_HOME>/etc/novas.rc

Specify Preferences

- Open *Preferences* form by invoking **Tools → Preferences** from any window
  - Changes to the form are saved in the *novas.rc* resource file
  - Search for keywords in the **Find** field



SpringSoft Accelerating Engineers

Topics

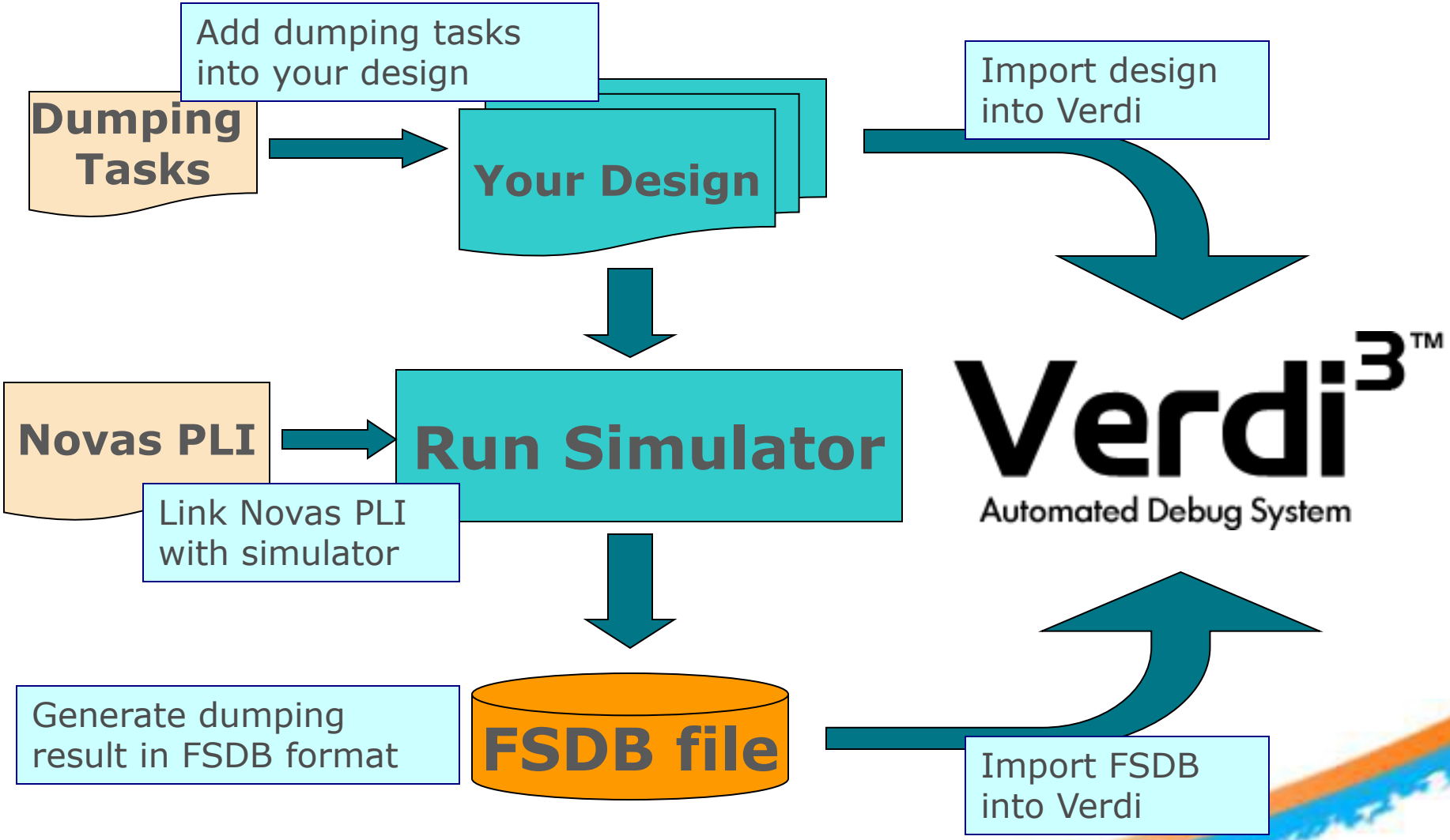
- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Fast Signal Data Base (FSDB)

Introduction

- The FSDB stores the simulation results in an efficient and compact format
- *SpringSoft* provides the object files that can be linked to common simulators to store the simulation results in FSDB format directly
  - Provide links to **VCS**, **IUS**, and **MODELSIM**
- Use system tasks for both VHDL and Verilog to dump values into FSDB files during simulation

Flow to Dump FSDB



SpringSoft
Accelerating Engineers

Link Novas Object File for FSDB Dumping

Run Simulation to Generate FSDB

- Setup tool environment variables:
  - Setup license file for simulator:

    % setenv LM_LICENSE_FILE *<simulator_license_file>*
  - Setup path for simulator:

    % setenv <Simulator>_INST_DIR *<simulator_tool_install>*
  - Setup path for Novas tool:

    % setenv VERDI3_HOME *<Novas_install_path>*
  - Setup **LD_LIBRARY_PATH** to specify the appropriate shared object files:

    % setenv LD_LIBRARY_PATH ${VERDI3_HOME}/share/PLI/*<simulator>*/*<platform>*
    - Supported simulators: **VCS** / **IUS** / **MODELSIM**
    - The correct simulator version will be selected automatically
- Create working library and compile designs for simulation
- Run and call FSDB dump functions on the simulator command prompt

NOTE: For more simulator linking details, refer to the <VERDI3_HOME>/doc/linking_dumping.pdf file

Link Novas Object File for FSDB Dumping

Example – Using VCS (1/3)

- Setup environment variables for VCS and Verdi
  - LM_LICENSE_FILE
    - setenv LM_LICENSE_FILE /dq3/license/third/synopsys/license.reg
    - setenv LM_LICENSE_FILE 5219@sps100:$LM_LICENSE_FILE
  - VCS_HOME
    - setenv VCS_HOME /novas/tools/Synopsys/VCS/<version>
  - VERDI3_HOME
    - setenv VERDI3_HOME /novas/tools/Springsoft/Novas/<version>

- Setup path for VCS and Verdi
  - Setup search path for VCS:
    - set path=($VCS_HOME/bin $path)
  - Setup search path for Verdi:
    - set path = ($VERDI3_HOME/bin $path)

Link Novas Object File for FSDB Dumping

Example – Using VCS (2/3)

- Link Novas PLI with *VCS*:
  - % setenv LD_LIBRARY_PATH \
    ${VERDI3_HOME}/share/PLI/**VCS**/${PLATFORM}:

- Add FSDB Dumping Tasks – The simplest way
  - Generates value changes for all signals in all scopes.
  - A novas.fsdb will be generated (default name).
  - Add following code in your design:

```
initial
  begin
    $fsdbDumpvars;
    #12500 $finish;
  end
```


SpringSoft
Accelerating Engineers

Link Novas Object File for FSDB Dumping

Example – Using VCS (3/3)

- Run *VCS*
  1. Remove previous **simv** if it exists:

     ```
     if(-e simv) then
          \rm -rf simv
     Endif
     ```

  2. Run *VCS* with **–P** option:
     - Point to **novas.tab** and **pli.a** provided by SpringSoft:

       ```
       % vcs  -debug_all –P
           $VERDI3_HOME/share/PLI/VCS/$PLATFORM/novas.tab \
           $VERDI3_HOME/share/PLI/VCS/$PLATFORM/pli.a \
           -f run.f
       % simv
       ```

  - **NOTE**: **-debug_all** is suggested for cases where MDAs or SVA are desired, otherwise **–debug_pp** should be added at least

- Load FSDB into Verdi

  ```
  % Verdi -f run.f -ssf novas.fsdb &
  ```

FSDB Dumping Tasks

Frequently Used Dumping Tasks

- **$fsdbDumpfile** - Specify FSDB file name, limit the FSDB file size
  - Uses the sliding window scheme to keep the last signal values in the FSDB file and drops the old values if file size exceeds the limitation.
- **$fsdbDumpvars** - Dump signal value changes of specified instances and depth
  - Can also specify the FSDB file name; different FSDB file names can be specified in each fsdbDumpvars command.
- **$fsdbDumpon/ fsdbDumpoff** - Turn on/off FSDB dumping
  - Can specify FSDB file name to turn on/off specific FSDB file
- **$fsdbAutoSwitchDumpfile** - Limit FSDB file size and switch dumping to new FSDB file automatically.

NOTE: Refer to *Appendix: Frequently Used Dumping Tasks* section in the end of this slide, to get detail usage of most frequently used system tasks

SpringSoft
Accelerating Engineers

FSDB Utilities

Process FSDB Files in Batch

- **vfast**
  - Convert VCD files to FSDB files
- **fsdb2vcd**
  - Convert FSDB files to VCD files
- **fsdbextract**
  - Extract signals, scopes, time periods from existing FSDB files without re-simulating
- **fsdbmerge**
  - Merge several FSDB files into one
- **fsdbreport**
  - Generates a report of value changes for specified signals

NOTE: Use <utility> -h for a list of all options or see the *Verdi3 and Siloti Command Reference* for more information

NOTE: Refer to the *Utilities* chapter of the *Verdi3 and Siloti Command Reference* for the complete utility list

Labs

Instruction

1. Copy <*Verdi_install*>*/demo/pli/VCS/verilog* to working directory (./verilog)
2. Modify *./verilog/sim/run_VCS.cmd* to generate *novas.fsdb* and load it into Verdi
3. Add/modify system tasks in *./verilog/src/system.v* to control FSDB dumping. Objectives are:
   - Generate a *novas.fsdb* that includes all scopes and signals
   - Generate a *My.fsdb* which only dumps signals under *system.i_cpu.i_ALUB* scope
   - Suppress signals under *system.i_pram* scope for both FSDB files
   - Turn off dumping from 5000ns to 7000ns for *My.fsdb*
   - Total simulation run 12500ns
4. Load both *novas.fsdb* and *My.fsdb* into Verdi, and check the difference between two FSDB files

Labs

Hints

- Add the following code into *./verilog/src/system.v*:

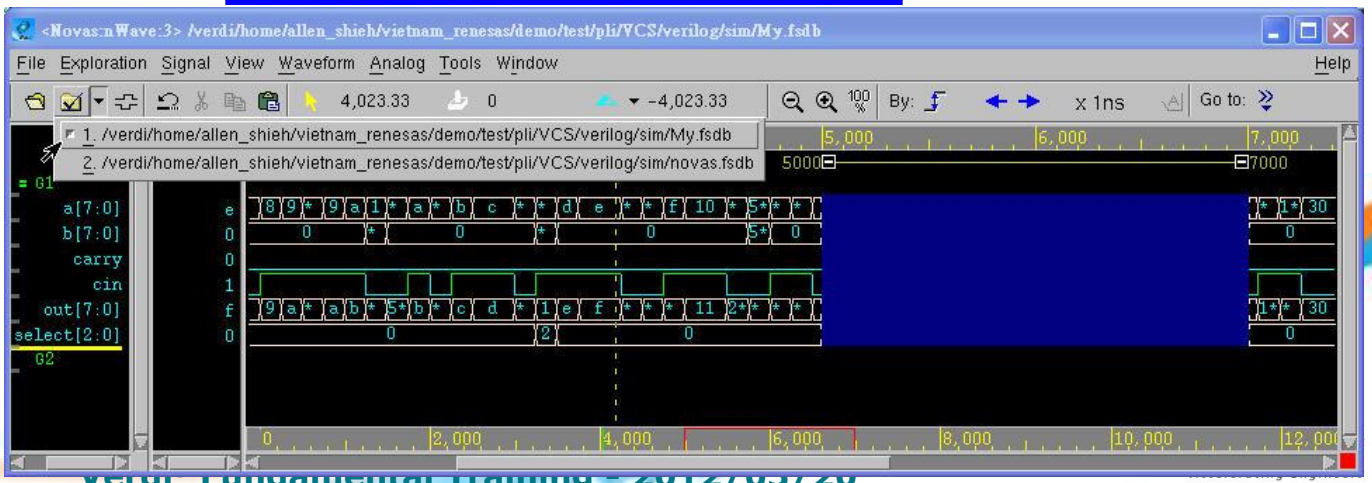
```
initial
  begin
    $fsdbSuppress(
    $fsdbDumpvars;
    $fsdbDumpvars(0, system.i_cpu.i_ALUB, "+fsdbfile+My.fsdb");
    #5000
    $fsdbDumpoff("+fsdbfile+My.fsdb");
    #2000
    $fsdbDumpon("+fsdbfile+My.fsdb");
    #5500 $finish;
  end
```

$fsdbSuppress is a global setting (affect all FSDB files), and must be specified before *$fsdbDumpvars*

Generate My.fsdb, includes all signals under system.i_cpu.i_ALUB scope and its child scopes (depth=0)

Turn off dumping between 5000~7000ns for My.fsdb

Continue dumping till 12500ns



Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Import Design on Command Line

From File

verdi [Verdi options] [<your Verilog options>]

- Reference source files on the command line (Verilog only)

  % verdi <source_file1> <source_file2>…

- Verdi takes all the Verilog command line options

  % verdi –f <file_list> +define+GETTHEMOSTOUTOFYOURTIME

  - Where *file_list* is a file that contains the Verilog source files and any command line switches

  % verdi –ssv –v lib.v +libext+.v –ssy –y /src/abc –y /src/def top.v

  - Where **–ssv** and **–ssy** are needed only once for design modules specified with the **–v** and **–y** and *top.v* is the design file

Import Design on Command Line

From Library

- Compile once, load many times
- Saves load time and overall memory
- Necessary for mixed language or VHDL designs; optional for Verilog only language designs

- Pre-compile design into library: *vericom* for Verilog code, *vhdlcom* for VHDL code

```
% vericom -lib <libName> block1.v block2.v block3.v
% vhdlcom -lib <libName> block1.vhd block2.vhd block3.vhd
```

- Then import the compiled libraries and specify the top module

```
% verdi -lib <libName> -top <TopBlock>
```

For example:

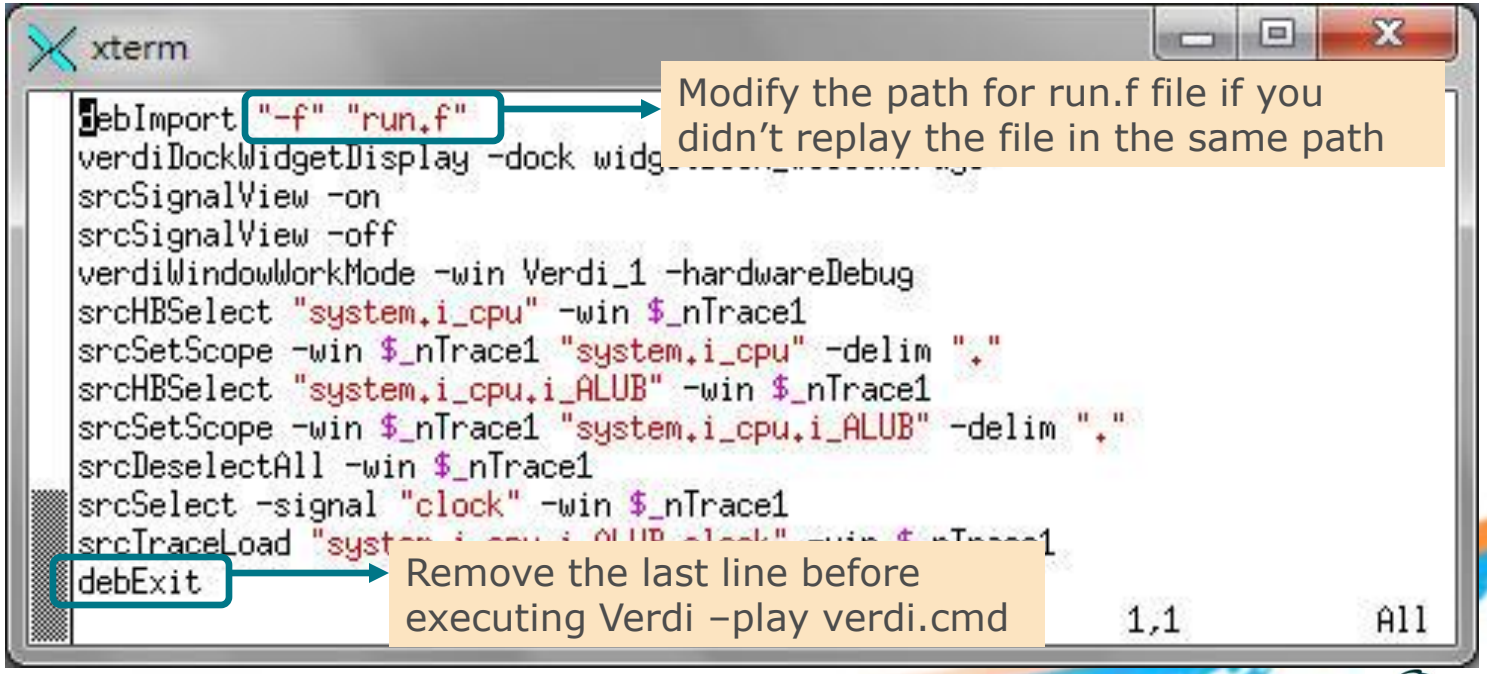
```
% verdi -lib work -top system
```

Import Design on Command Line

By Replaying a Tcl Command File

- Every Tcl command executed in Verdi has an equivalent Tcl command saved in the *./verdiLog/verdi.cmd* file
  - This file may be copied and modified to automate frequent commands
  - The new file may then be replayed on the Verdi command line using:

    % verdi –play <file>.cmd
- Example for how to modify a **verdi.cmd** file:

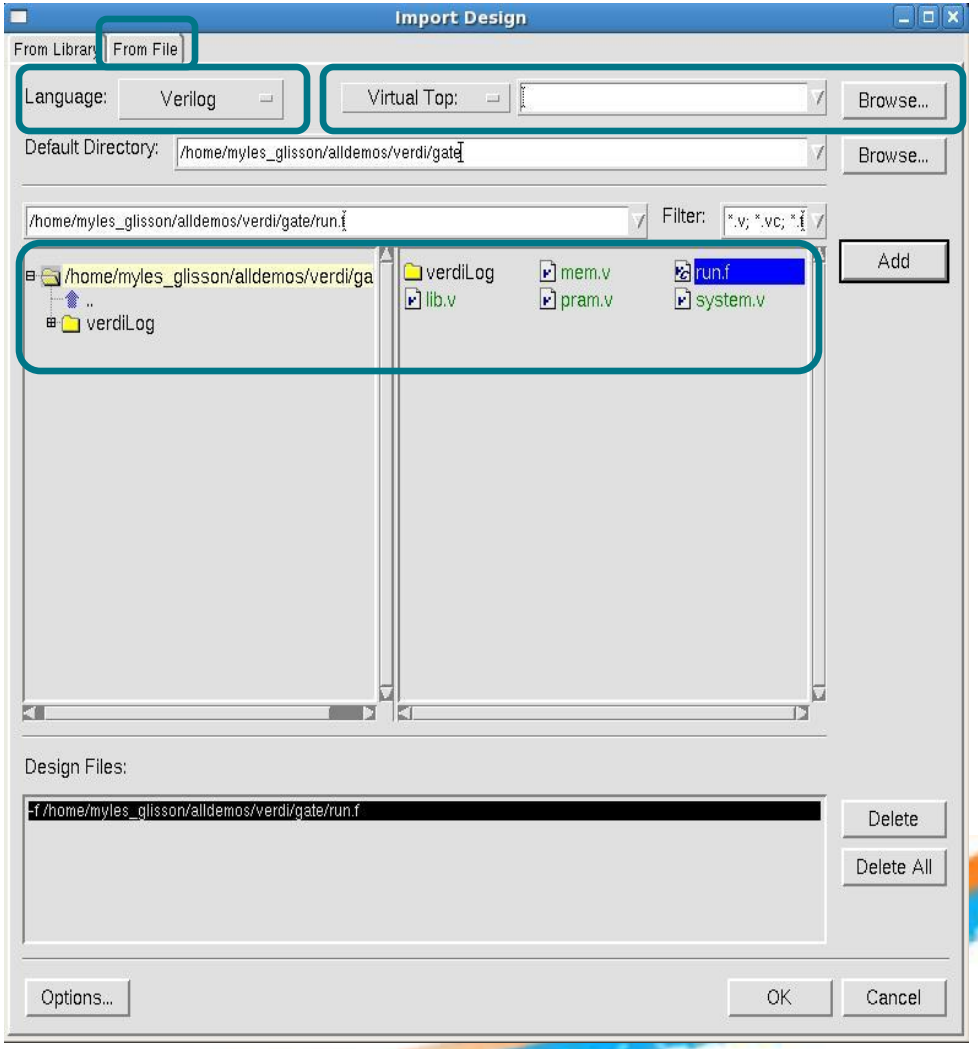


SpringSoft Accelerating Engineers

Import Design from GUI

From File

- Enter **verdi** on command line to launch Verdi
- Use **File → Import Design** or **Import Design** icon to import your design
  - Select **From File** tab
  - Specify language
  - Specify run file or individual source files
  - Supports mapping file to set a Virtual Top
    - Use *Virtual Top* if the design hierarchy does not match the FSDB hierarchy
- Not recommended for VHDL

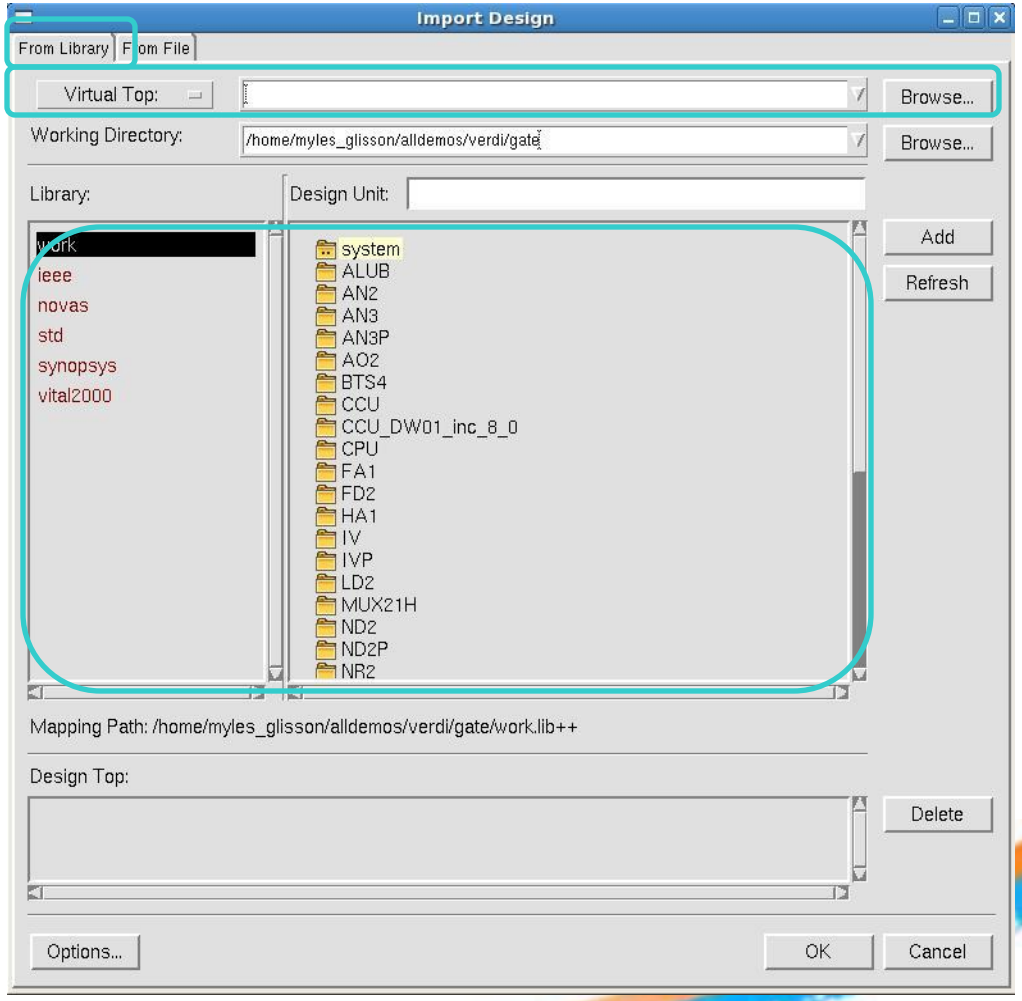


SpringSoft Accelerating Engineers

Import Design from GUI

From Library

- Enter **verdi** on command line to launch Verdi
- Use **File → Import Design** or **Import Design** icon to import your design
  - Select **From Library** tab
  - Specify the library and the top module
  - Supports mapping file to set a Virtual Top
    - Use Virtual Top if the design hierarchy does not match the FSDB hierarchy



SpringSoft Accelerating Engineers

Import Mixed Language Design

Use Mixed Language Compiler – nrun (1/2)

- Compile mixed design files (Verilog and VHDL) to libraries
  - Instead of using both *vericom* and *vhdlcom* utilities
- The type of input files will be determined by its file extension:

Language Type	File Extension
Verilog 95	.v95, .V95, .v95p, .V95P
Verilog (2001)	.v, .V, .vp, .VP, .vs, .VS, vcfg
System Verilog	.sv, .SV, .svp, .SVP, .svi, .svh, .vlib, .VLIB
VHDL (93)	vhd, .VHD, .vhdl, .VHDL, .vhdp, .VHDP, .vhdlp, .VHDLP, .vhcfg

- Example:
  - % nrun a.v95 b.v c.sv d.vhd

    Is equivalent to:
  - % vericom +verilog1995ext+.v95 +systemverilogext+.sv a.v95 b.v c.sv
  - % vhdlcom d.vhd

Import Mixed Language Design

Use Mixed Language Compiler – nrun (1/2)

- Specify the default type with the ***–default_ext <file_type>*** option
  - Supported file types: **verilog95**, **verilog**, **systemverilog**, and **vhdl**

- Use ***-languageType_ext [+]fileext,.[fileext...]***to override the default file extensions; supported *languageType_ext* are:

Language Type	**Options**
Verilog 95	vlog95_ext
Verilog (2001)	vlog_ext / vcfg_ext / vlogext
System Verilog	sysv_ext
VHDL (93)	vhdl_ext / vhcfg_ext / vhdlext

- Refer to the *Utilities* chapter**,** *Compilation* → *nrun* section in the *<VERDI3_HOME>/doc/reference.pdf* for detailed options

SpringSoft
Accelerating Engineers

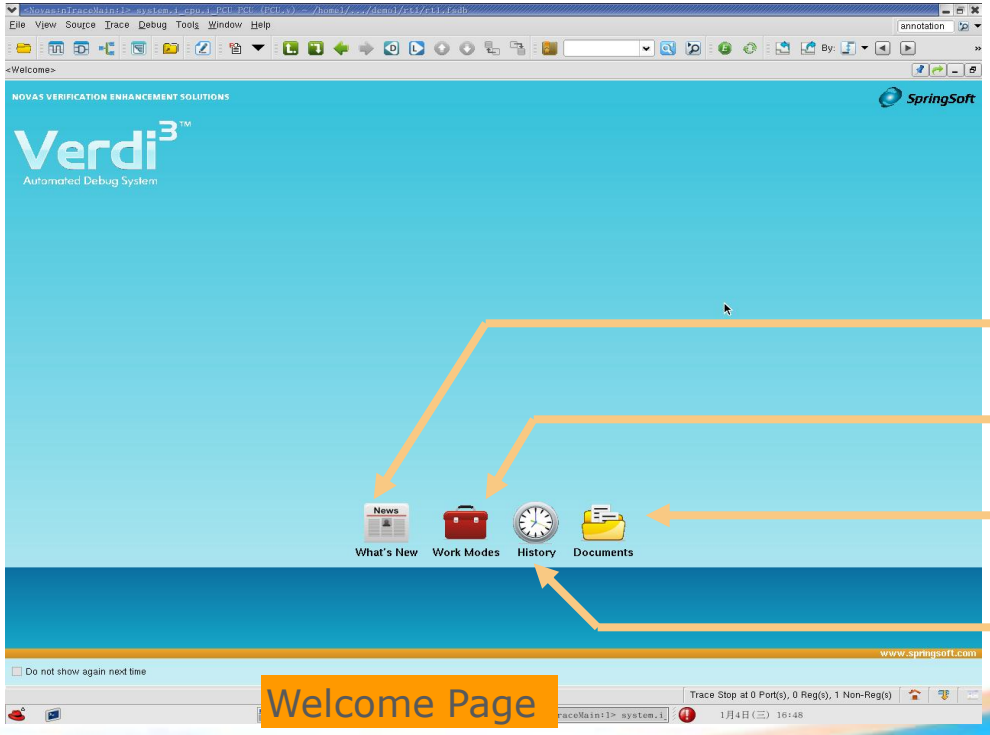
Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Verdi^3 Interface

Welcome Page

- The first page after invoking Verdi^3
  - Quick access to information
    - Direct access to documents and video clips
    - See what's new in the release
  - Easy access to different user modes



What's new

Mode Selection

Documents

History

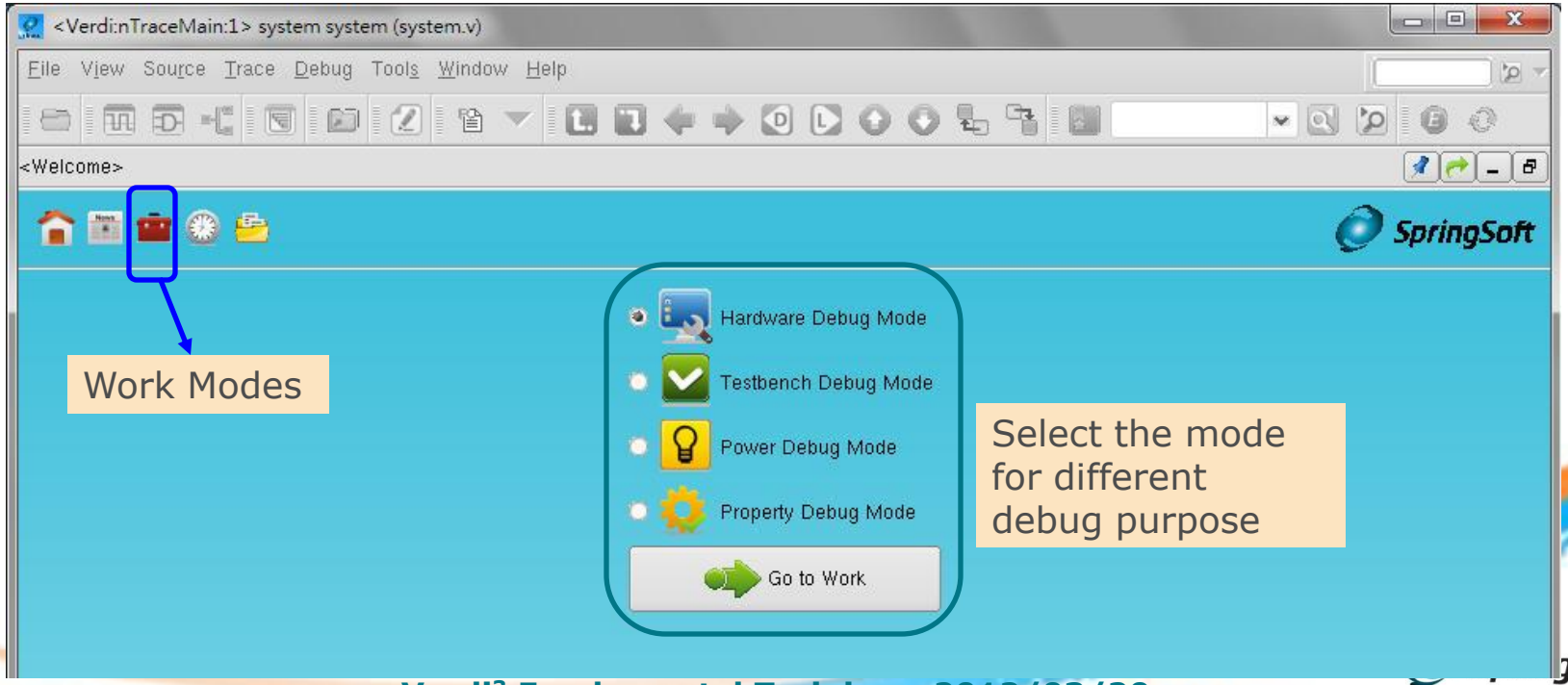
Welcome Page

SpringSoft Accelerating Engineers

Verdi³ Interface

Welcome Page – Work Mode

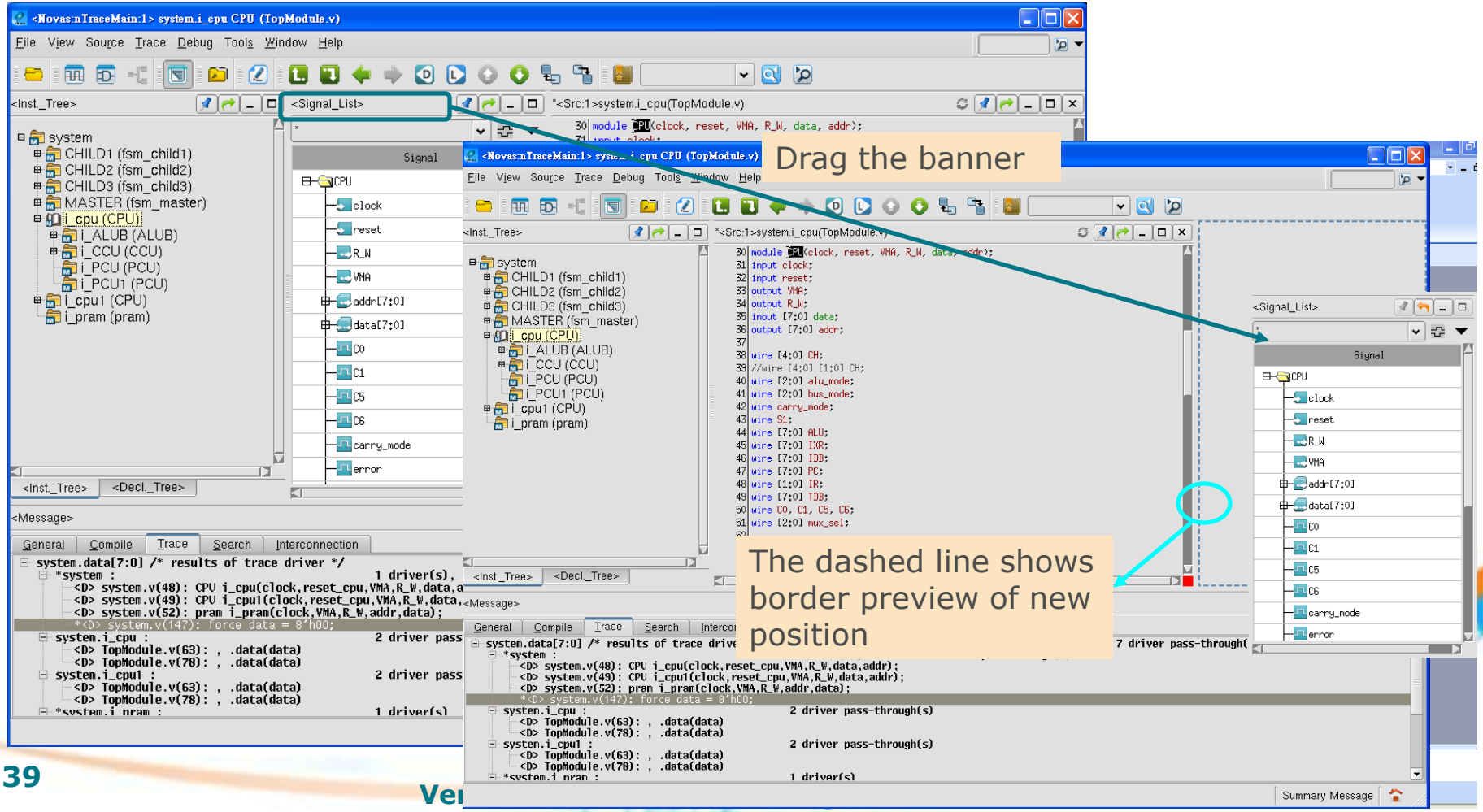
- Click the **Work Modes** icon in the *Welcome Page* to select modes
  - Different layouts for different debug purposes
    - Necessary components will be opened on the same page
  - Available modes: **Hardware Debug**, **Testbench Debug**, **Power Debug**, and **Property Debug**
    - Select the mode and click the **Go to Work** button to open the mode



Verdi³ Interface

Customize Design Views

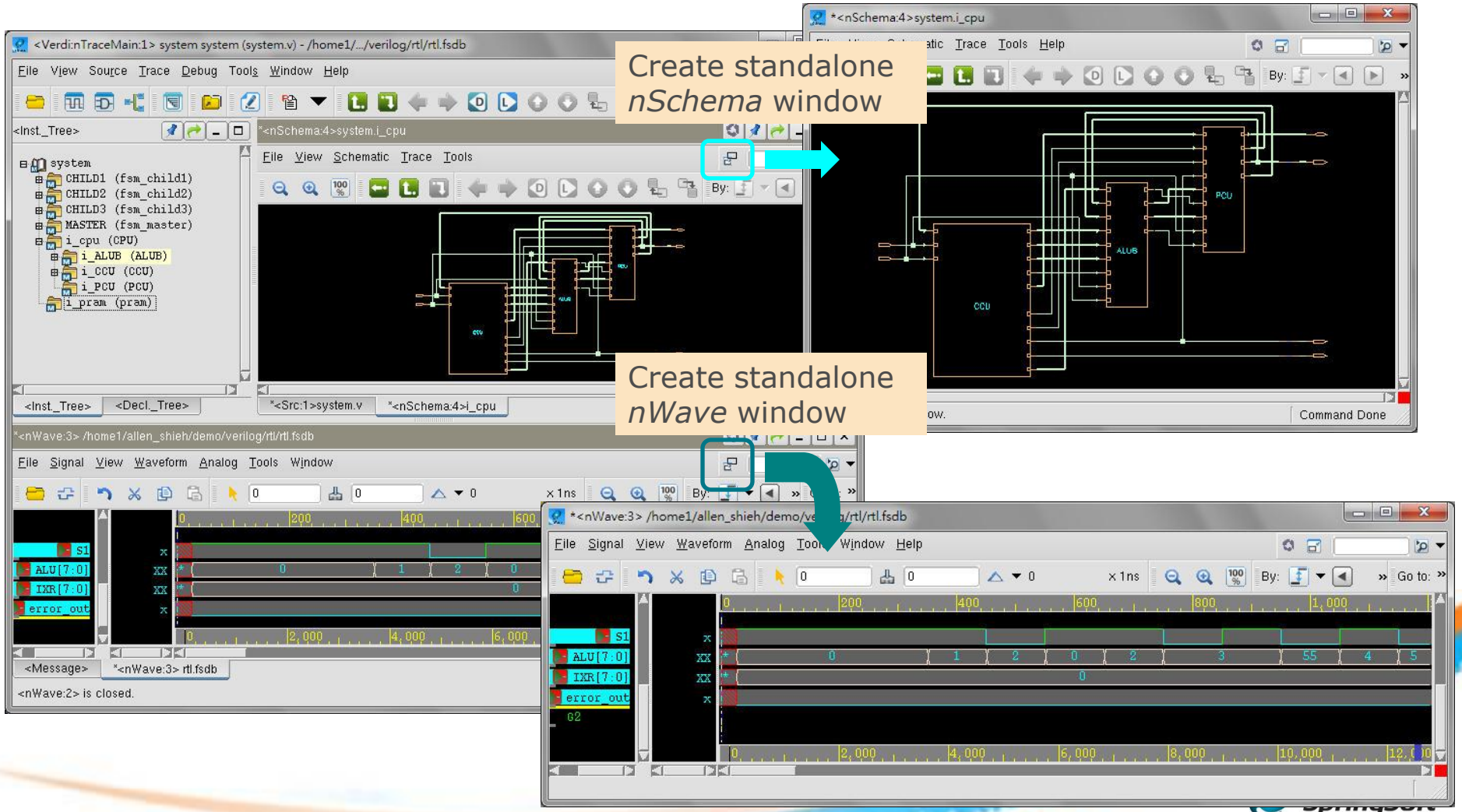
- All frames can be customized to preferred locations
  - Move the frame to the same area of another frame will create a tab



Verdi³ Interface

Customize Design Views – Create Standalone Windows

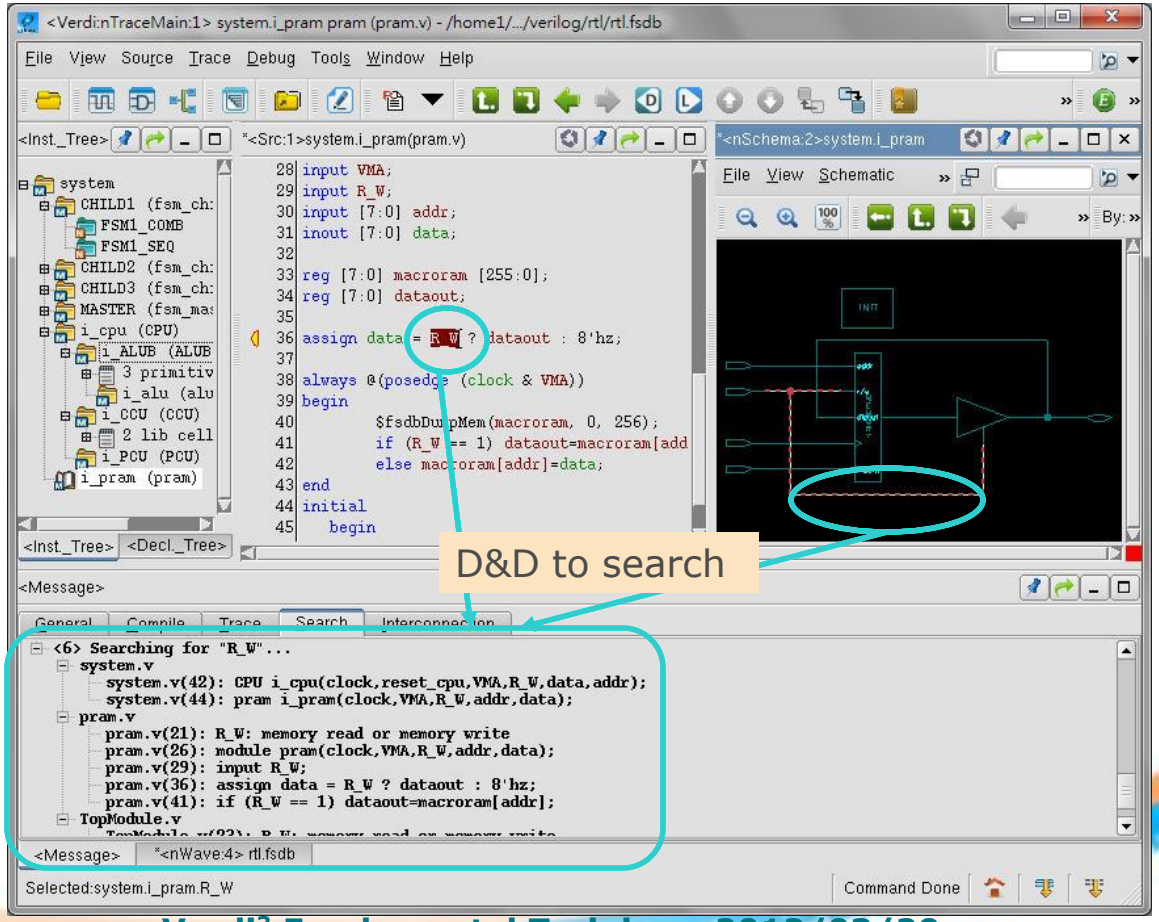
- Click the **Be Window** icon to pull the frame out as a standalone window



Verdi³ Interface

Drag & Drop Between Frames

- Drag and drop can be performed between different frames
  - For example, drag a signal from *nSchema* or *nTrace* to the **Search** tab of the *Message* frame will execute **Search** string

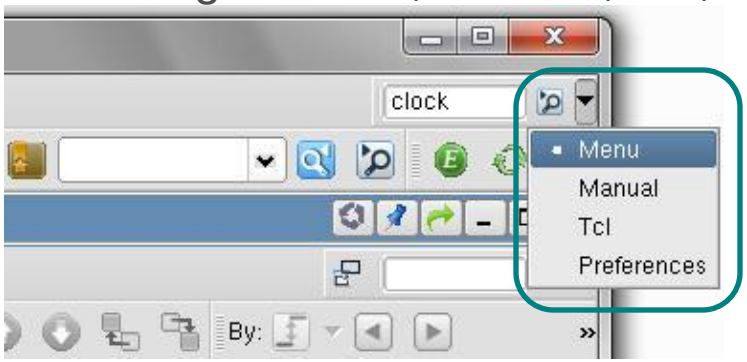


SpringSoft
Accelerating Engineers

Verdi³ Interface

Spotlight Search

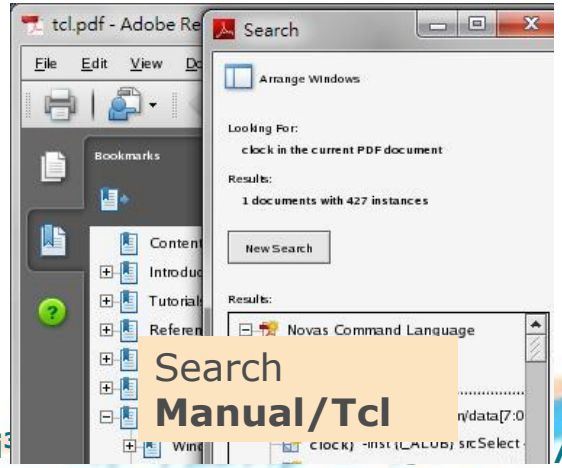
- Click the arrow to the right of the spotlight icon to select search target
  - Select to search string in **Menu**, **Manual**, **Tcl**, or **Preferences**



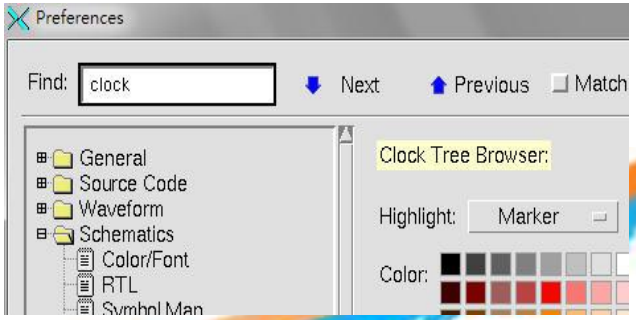
Set search target

- **Menu**: Matched menu commands will be listed under the spotlight icon
- **Manual/Tcl**: Adobe reader will be invoked and search string will be executed
- **Preferences**: *Preferences* form will be opened and search will be executed

Search **Menu**



Search **Manual/Tcl**



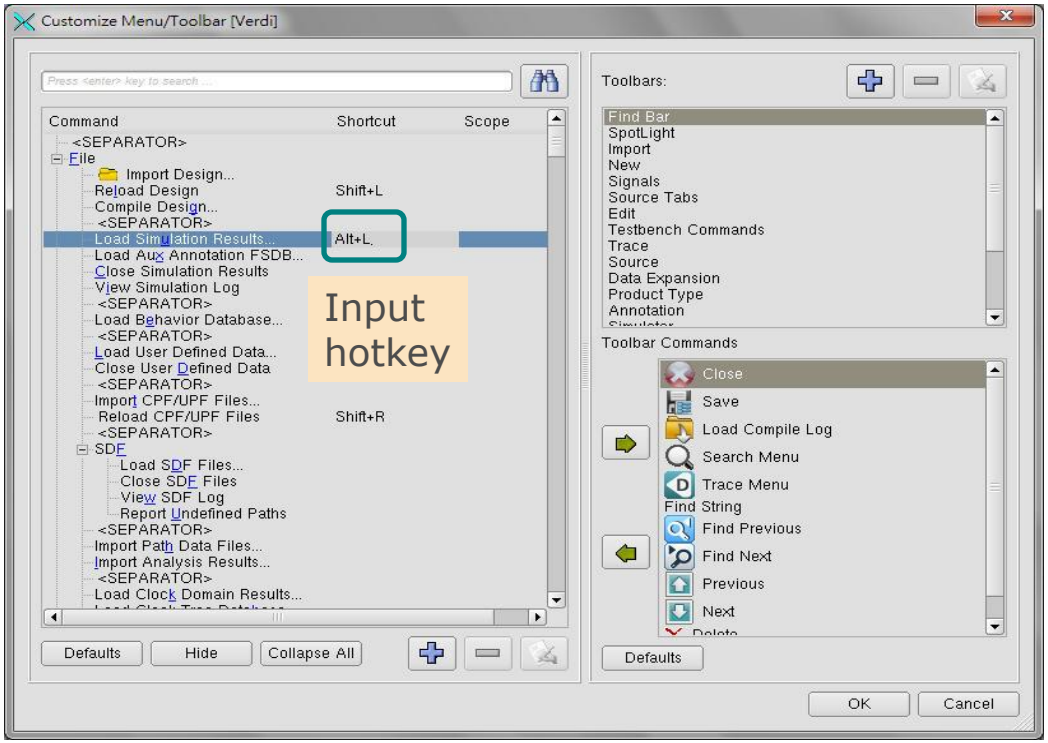
Search **Preference**

SpringSoft
Accelerating Engineers

Verdi³ Interface

Customize Bind Keys

- Invoke **Tools → Customize Menu/Toolbar** from *nTrace*, *nSchema*, *nWave*, or *TFV* frame to open the *Customize Menu/Toolbar* form
  - Double-click the **Shortcut** column to specify a hotkey



- Refer to the *Verdi³ User Interface Training* for other customizations

SpringSoft
Accelerating Engineers

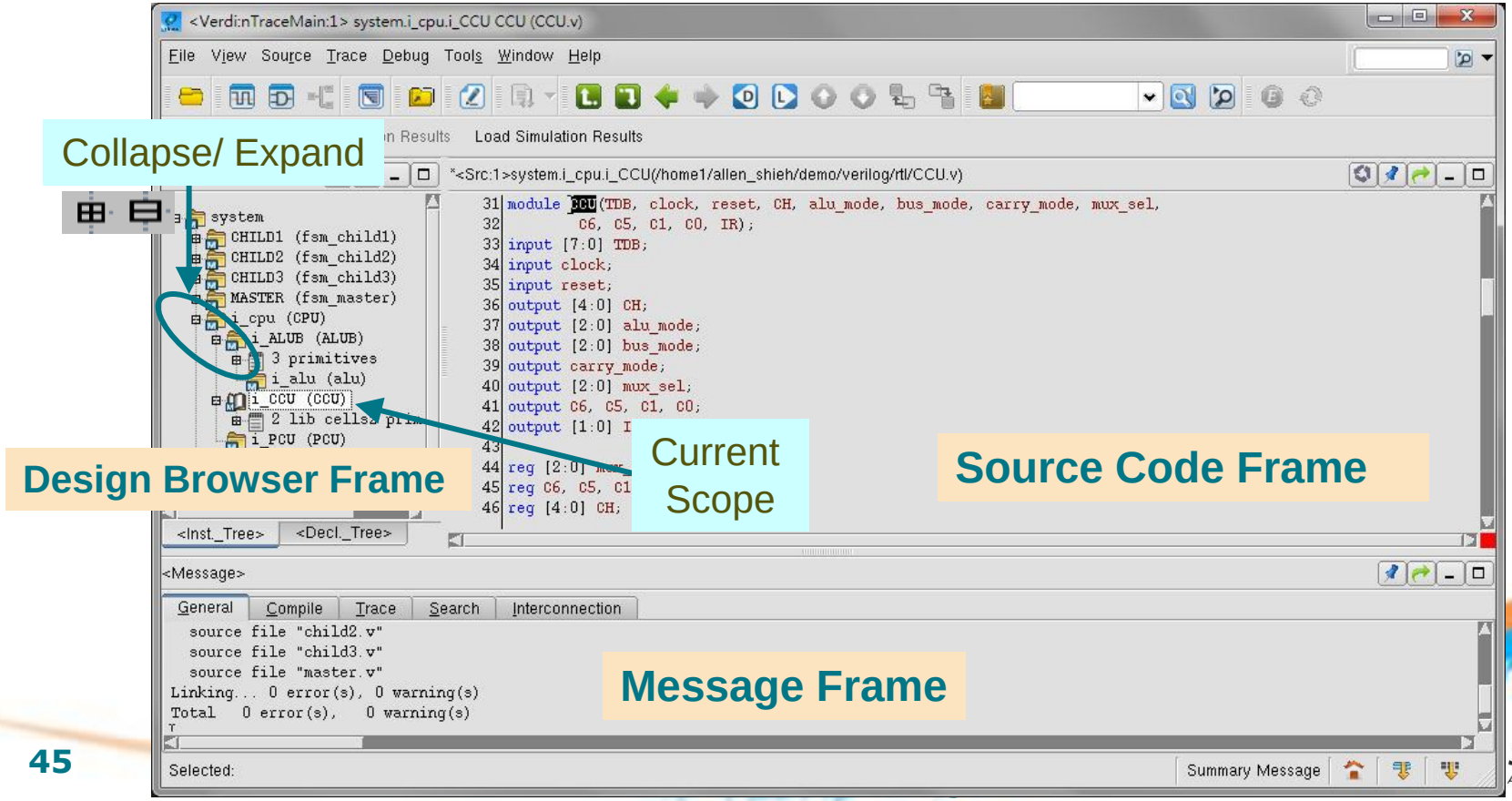
Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Open *nTrace*

Main Window

- The main window is opened after selecting the *Work Mode* in the *Welcome Page*
  - Contains three re-sizable frames (*Hardware Debug Mode*)
    - The layout and frames may be different for different work modes

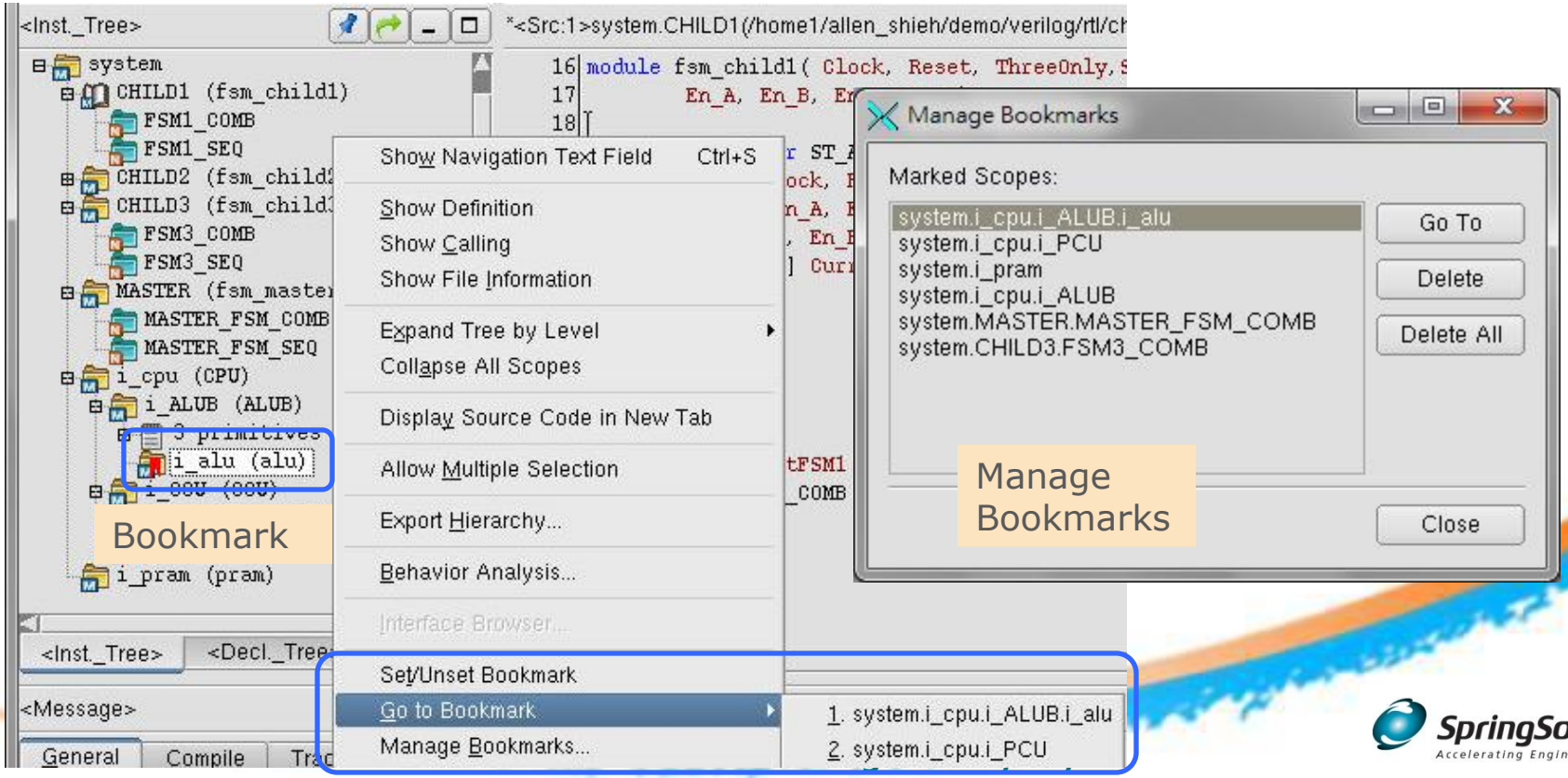


SpringSoft
Accelerating Engineers

Search and Collapse Design Browser

Instance Tree

- Double click a node in the **<Inst._Tree>** tab of the design browser frame to see the source code in the source code frame
- Select a node and invoke **RMB → Set/Unset Bookmark** to set the bookmark
  - Bookmarks will appear under the **RMB → Go to Bookmark** command
  - **RMB → Manage Bookmarks** command to manage bookmarks

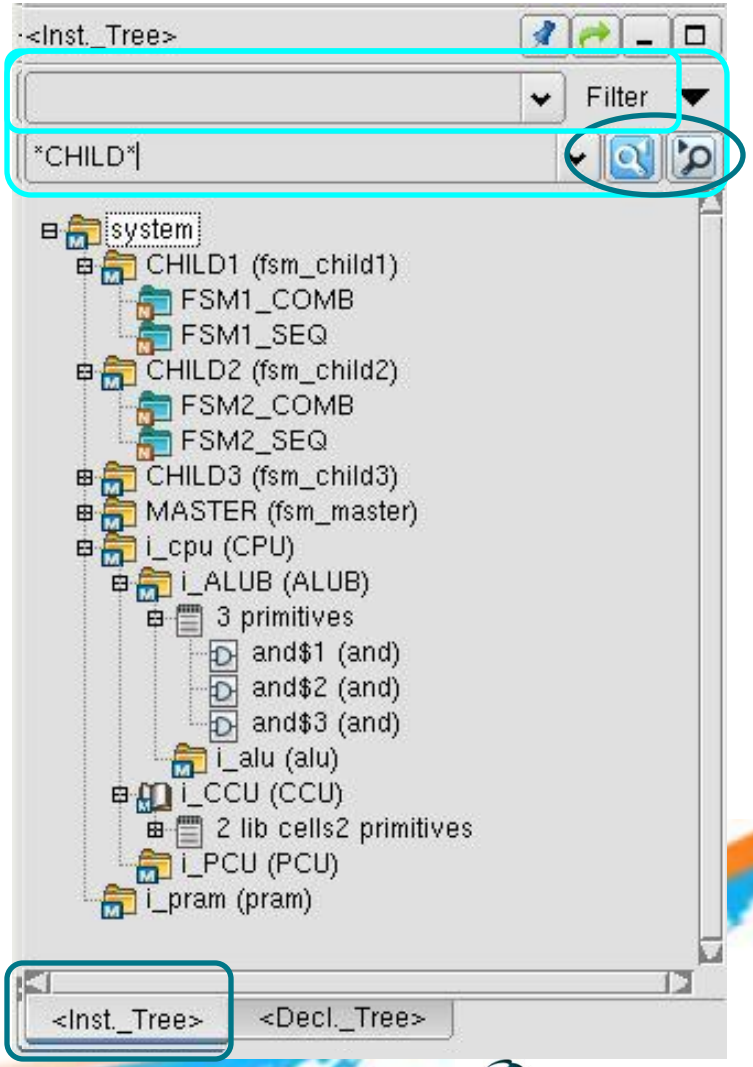


SpringSoft
Accelerating Engineers

Search and Collapse Design Browser

Instance Tree - Search & Filter Scopes

- In the **<Inst_Tree>** tab of the design browser, invoke **RMB → Show Navigation Editbox** to turn on the *Navigation Box*
  - Search Scopes:
    - Type a wildcard string in the *Navigation Box*, and press **Enter**
    - The hierarchy browser jumps to the first matched scope, click on **Next Node** icon to jump to the next match
    - For example, type **CH*** in the *Navigation Box*, **CHILD1**, **CHILD2**, and **CHILD3** will be matched
  - Filter Scopes:
    - Type a wildcard string in the **Node Filter** field, only matched scopes will be shown



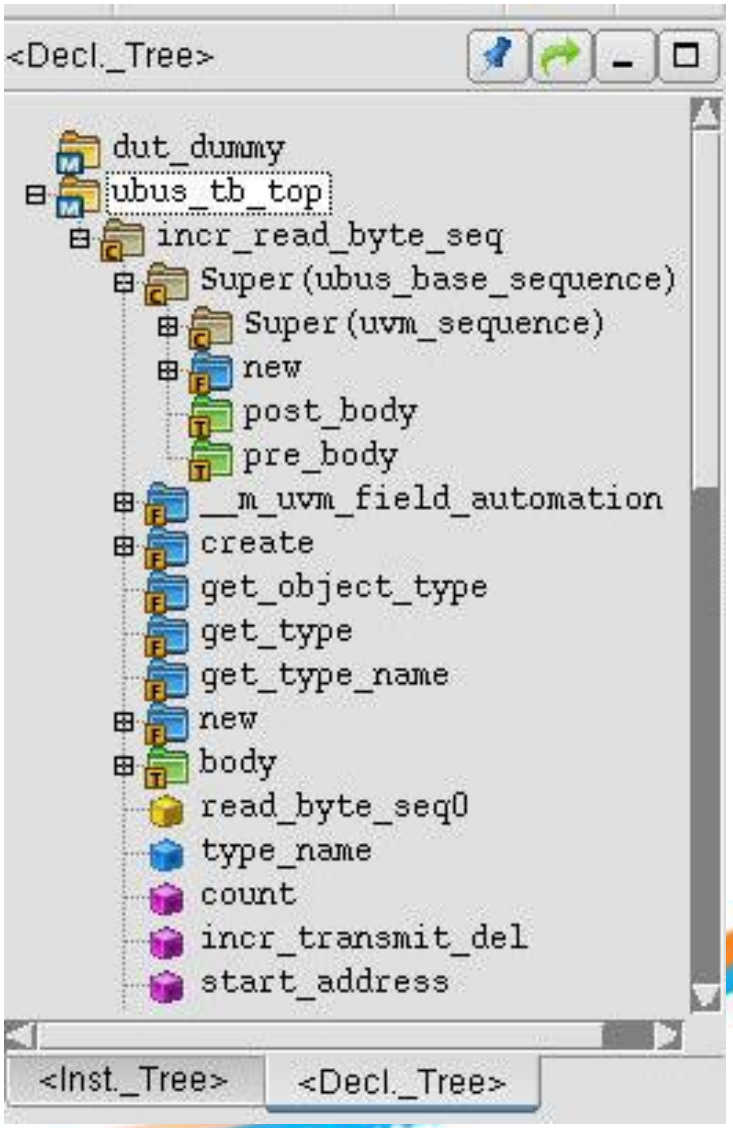
SpringSoft Accelerating Engineers

Search and Collapse Design Browser

Declaration Tree

- Click the **<Decl. Tree>** tab in the design browser to browse the declaration in designs and testbenches
  - The **<Decl. Tree>** tab includes module, entity (vhdl), class, class member data, task, function and package

- The selection in the **<Decl. Tree>** tab is synchronized with **<Inst. Tree>** tab and source code pane
  - Drag and drop items from other views to the **Decl. Tree** tab to identify the declaration

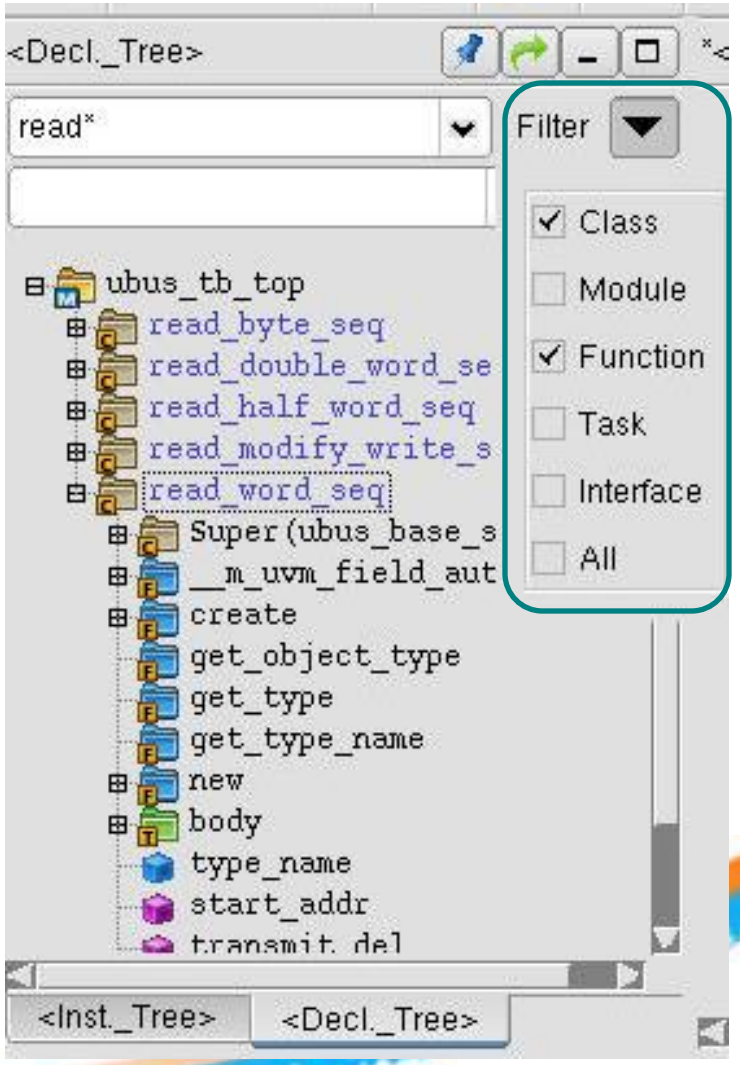


SpringSoft Accelerating Engineers

Search and Collapse Design Browser

Declaration Tree – Search & Filter

- Put mouse cursor in the **Decl. Tree** tab, invoke the **RMB → Show Navigation Text Field** command to display the **Navigation Text Field**
  - Search declarations:
    - Type a wildcard string, and click **Next Node** button to search
  - Filter declarations :
    - Click the arrow to the right of the **Filter** button to specify which type needs to be displayed
      - Click the **Filter** button again to apply the setting
    - Type a string in the **Filter** text field and press ENTER to filter declarations
      - Wildcard is supported
    - A blue node means matched objects
    - The number of matched items will be shown in the *Message* frame

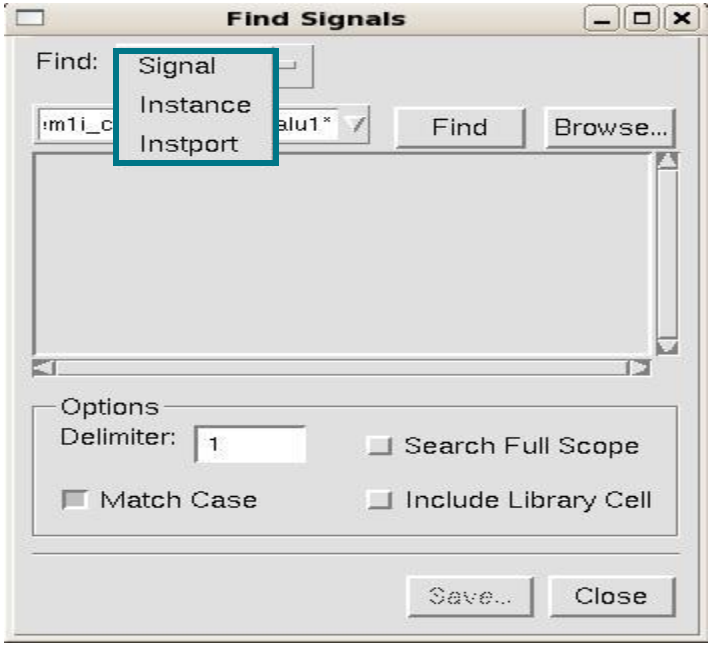


SpringSoft Accelerating Engineers

Traverse Source Code

Find Operations (1/2)

- **Source → Find Signal/Instance/Instport** is a fast way to find signals, instances, or instance ports in full design or selected scope



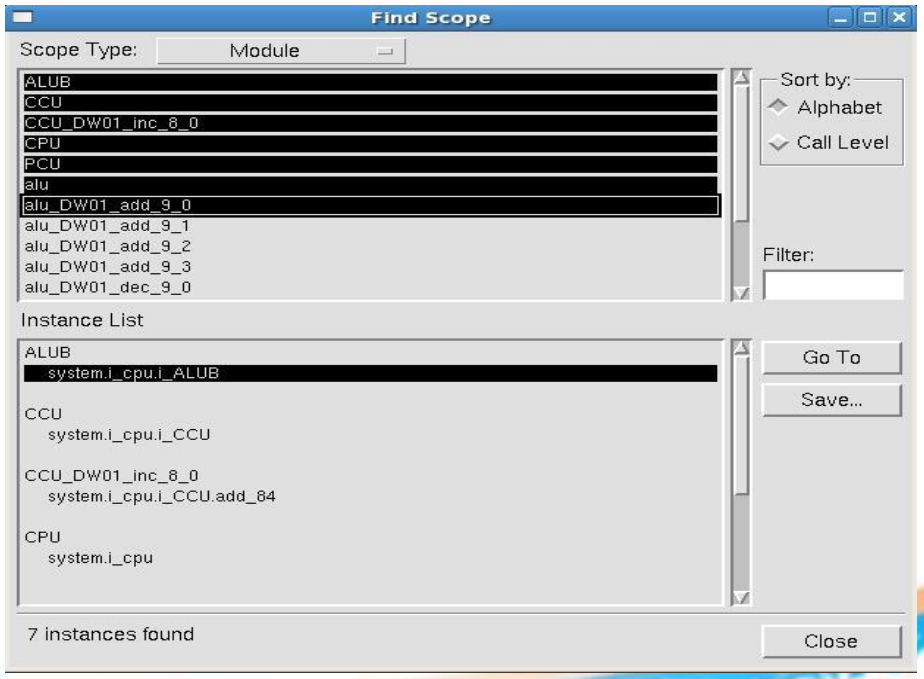
- **Find String** on toolbar quickly locates a string in current scope

SpringSoft
Accelerating Engineers

Traverse Source Code

Find Operations (2/2)

- **Source → Find Scope** is a fast way to directly open a Verilog module or VHDL architecture if the design tree is deep
  - Select scopes to list related instances
  - Multiple selection is allowed
    - Applies to Module/File/Function/Task types in **Scope Type** field
    - The complete **Instance List** can be saved by clicking the **Save** button
  - Only single selection is allowed in **Instance List** since it affects the **Go To** button

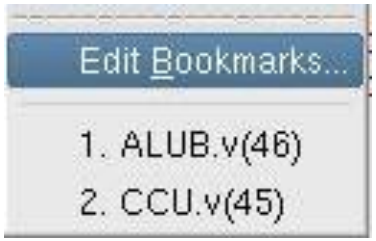


SpringSoft
Accelerating Engineers

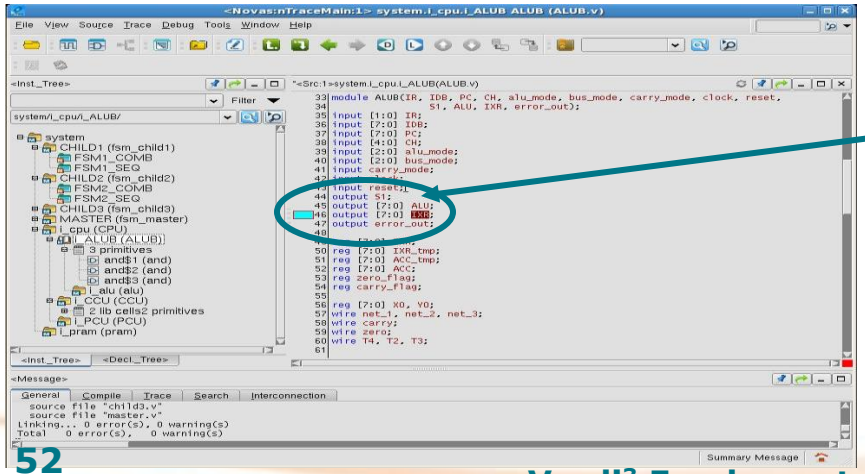
Traverse Source Code

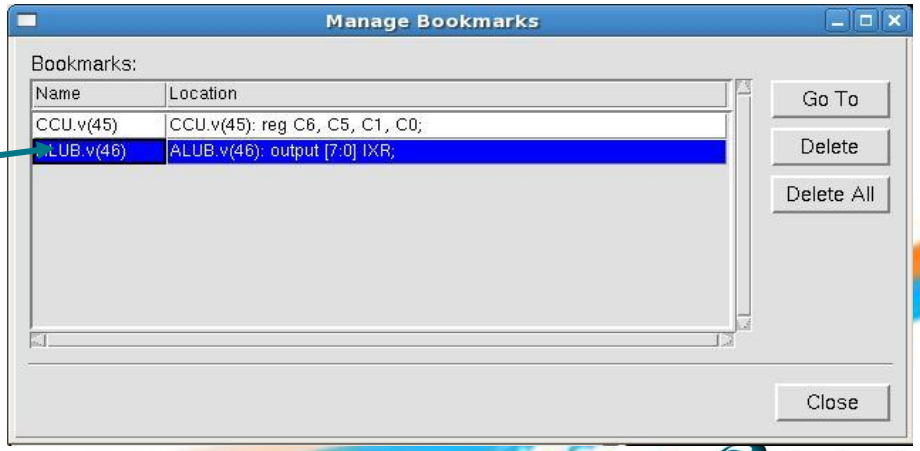
Use Bookmarks

- Add bookmarks:
  - Click on **Bookmark** icon on toolbar to bookmark current line
  - All bookmarks will be listed under **Source → Manage Bookmarks**; click the line to go to bookmarks



- Use **Source → Manage Bookmarks → Edit** command to get list of current bookmarks
  - Enter strings in the **Name** field to name your bookmarks

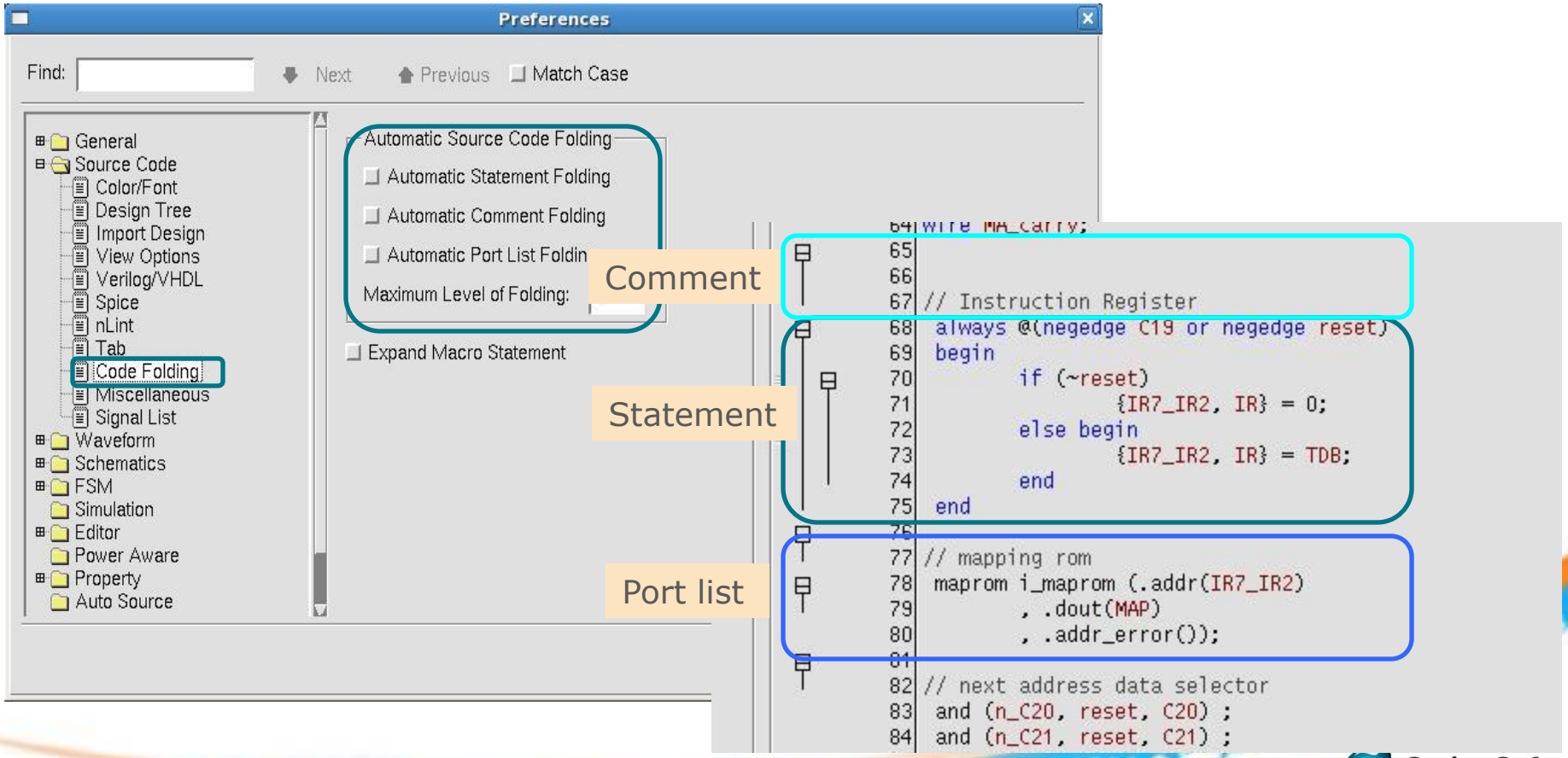




Traverse Source Code

Collapse Source Code (1/2)

- Invoke **Tools → Preferences** to open the *Preferences* form, go to **Source Code → Code Folding** page, three options are available to automatically fold **Statements**, **Comments**, and **Lists**

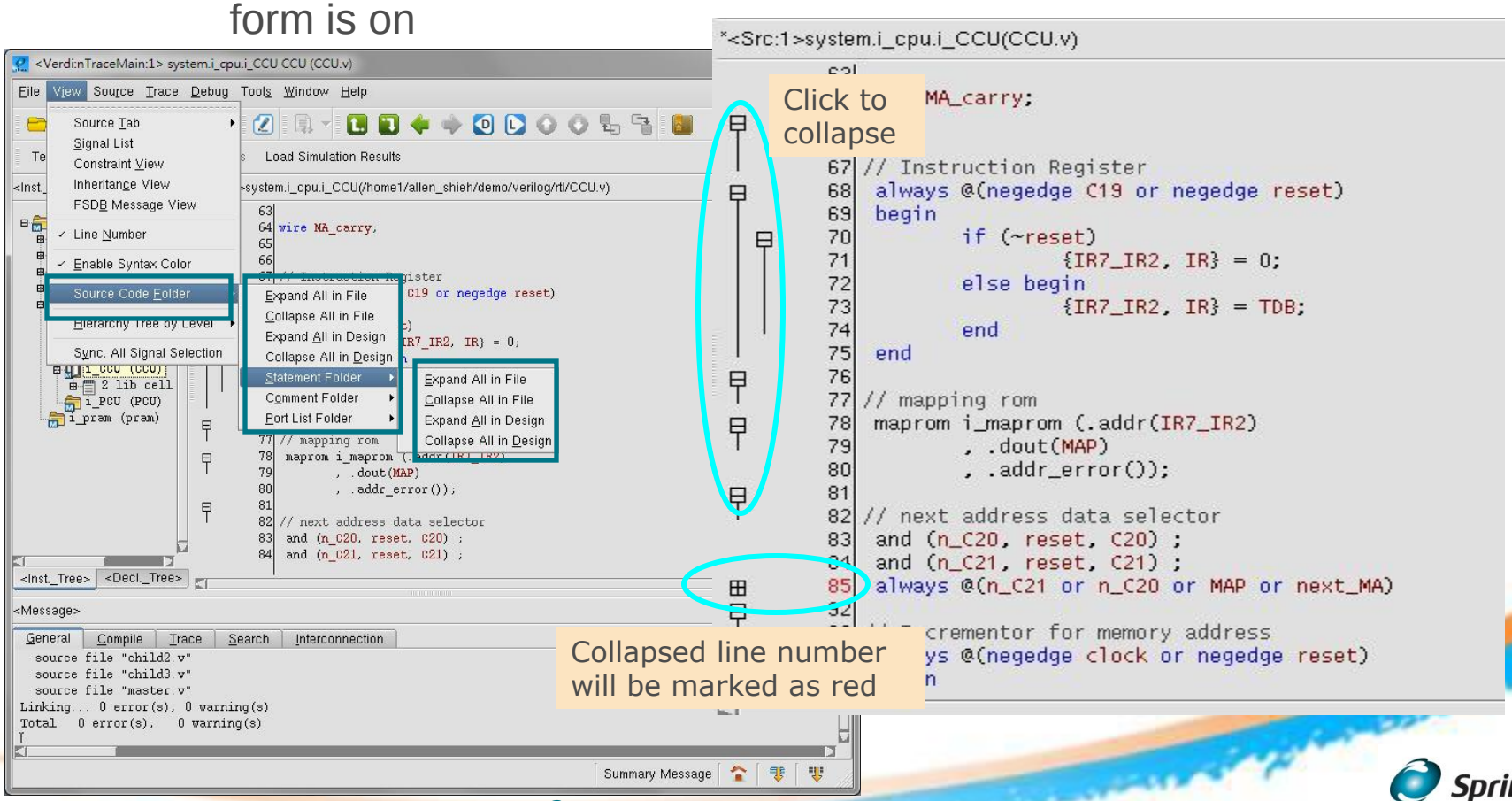


SpringSoft Accelerating Engineers

Traverse Source Code

Collapse Source Code (2/2)

- Use commands under **View → Source Code Folder** or click the mark "**+**" or "**-**" to expand or collapse the folded source code
  - These commands will be available only if the setting in *Preferences* form is on

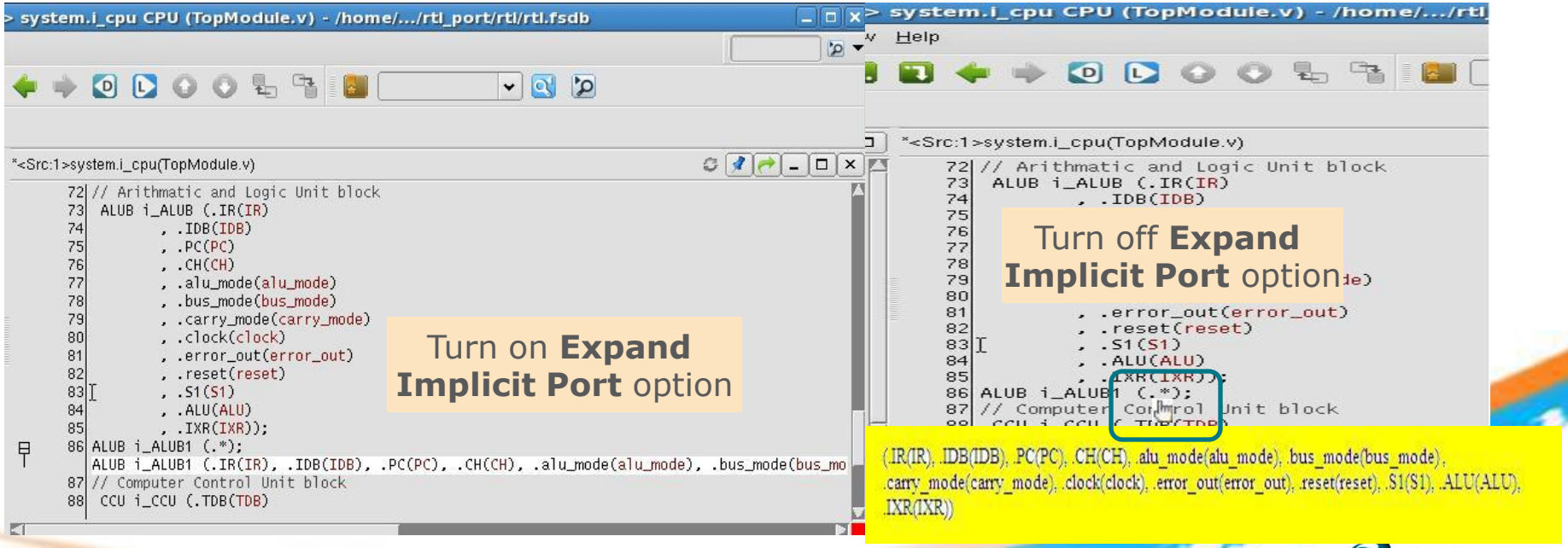


SpringSoft Accelerating Engineers

Traverse Source Code

Expand Implicit Port for SystemVerilog

- In *nTrace*, turn on the **Source → Expand Implicit Port** to expand the *SystemVerilog* implicit port statements
  - Click the ***+/-*** icon to expand/collapse the statement
  - Trace commands and active annotation are available in the expanded statement
- If the **Expand Implicit Port** option is off, a tip will appear when moving the mouse cursor over ***
  - The tip will be disabled when the option is turned on

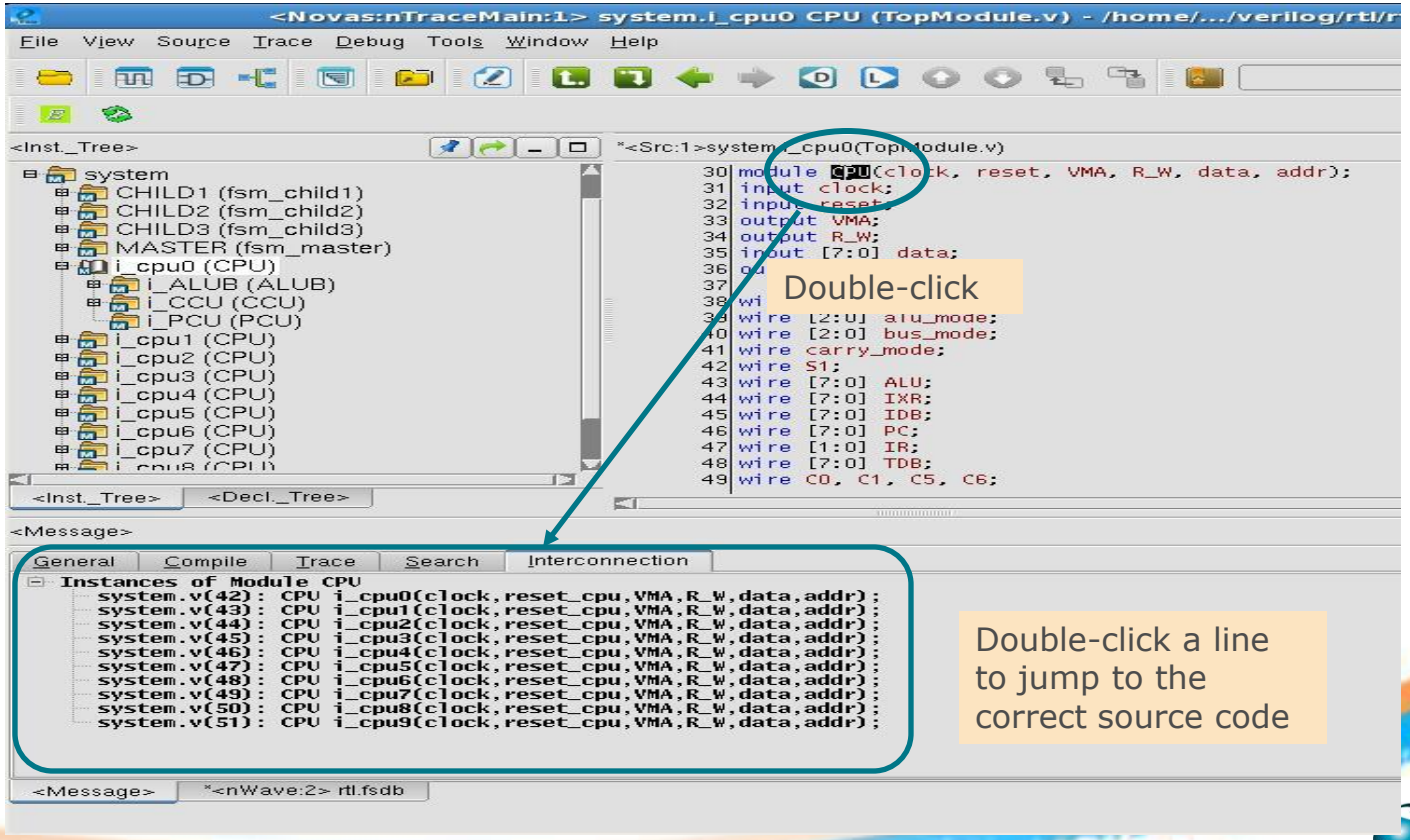


SpringSoft Accelerating Engineers

Traverse Source Code

Double-click a Module Name to List all Instances

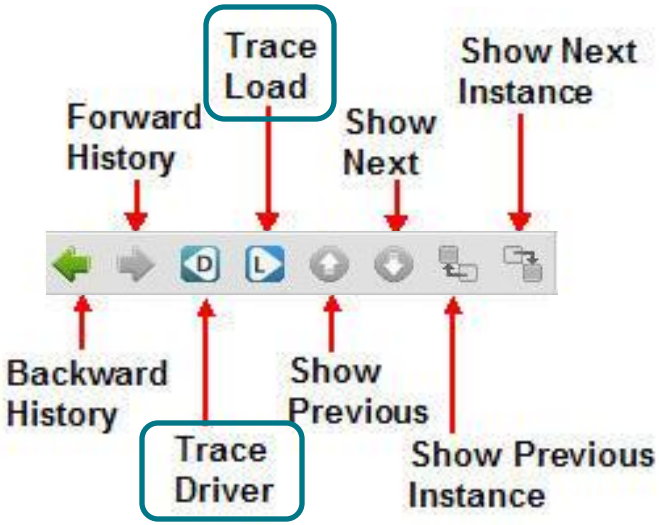
- Double-click a module name in the source code pane
  - All instances will be listed in the message pane
  - Double-click a line in the *Message* frame to jump to the expected instance



Operate on Signals

Trace Driver and Loads

- To quickly find all the ***drivers*** of a signal:
  - DC a signal
  - Select signal and click on **Trace Driver** icon
  - **Trace → Driver** or Right Mouse Button menu



- To quickly find all the ***loads*** of a signal:
  - Select signal and click on **Trace Load** icon
  - **Trace → Load** or Right Mouse Button menu

- To quickly find all the ***drivers and loads*** of a signal:
  - **Trace → Connectivity** or Right Mouse Button menu
  - Dropping a signal in the source window will trace the connectivity of the signal

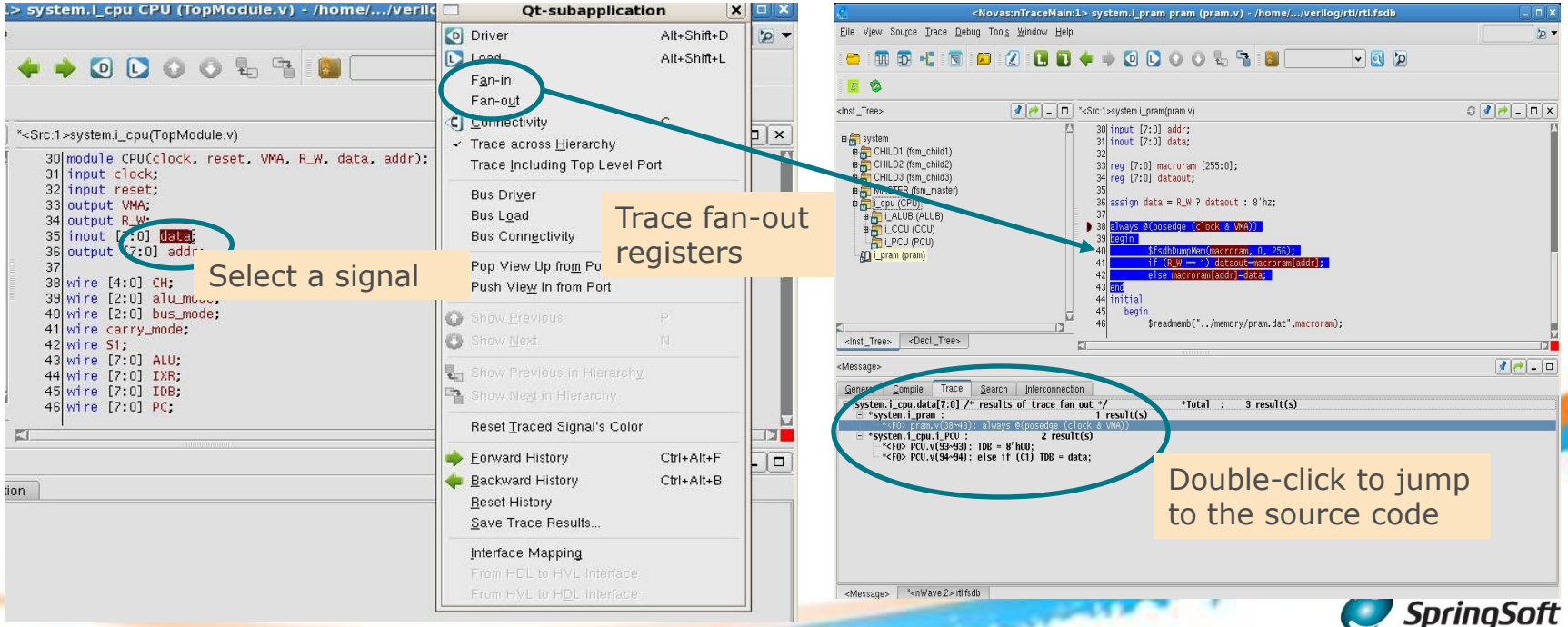
- By default, tracing will cross hierarchical boundaries
  - Disable **Trace → Trace across Hierarchy** to keep within a hierarchy

SpringSoft
Accelerating Engineers

Operate on Signals

Trace Fan-in and Fan-out

- Select a signal in the source code pane and invoke the command **Trace → Fan-in** or **Trace → Fan-out** command
  - Traces fan-in or fan-out registers
  - Jumps to the first trace result in the source code pane
  - Lists all traced results in the message pane
    - Jump to a different trace result in the source code pane by double-clicking in the *Message* frame



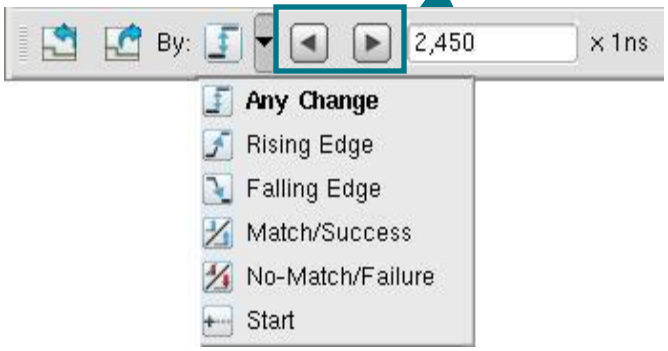
SpringSoft Accelerating Engineers

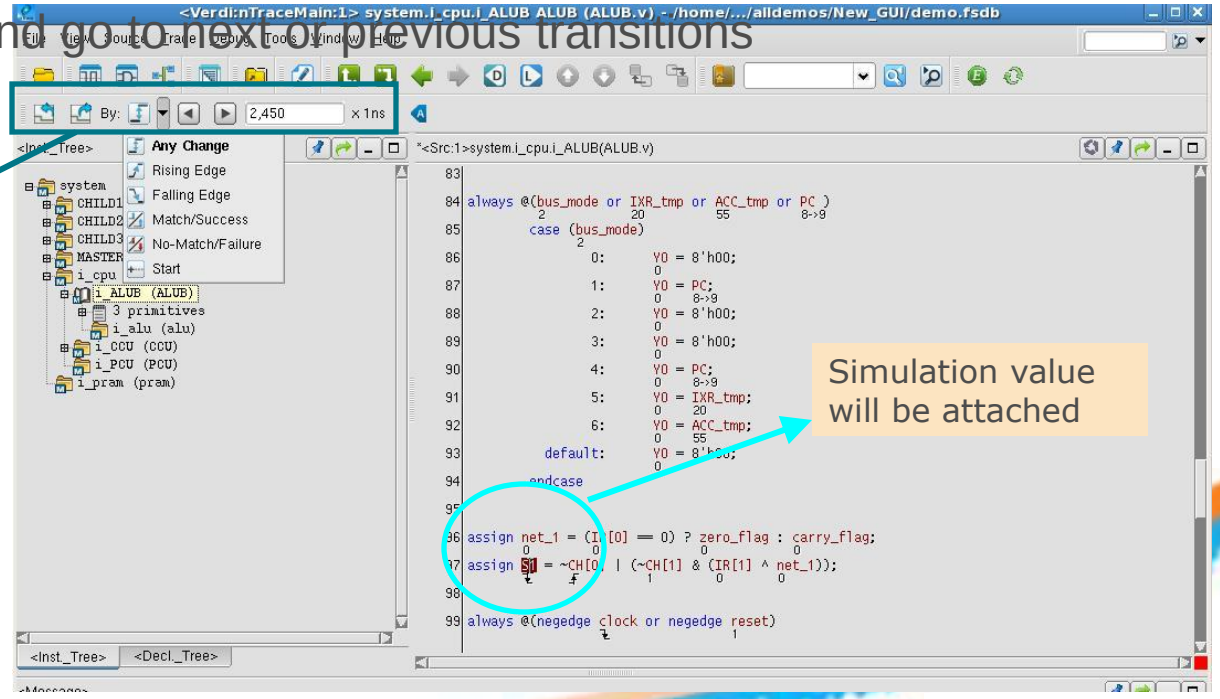
Operate on Signals

Enable Active Annotation

- Use the **File → Load Simulation Results** command to load FSDB file
- Enable **Source → Active Annotation** (or hot key x) in *nTrace*
  - Display simulation results on source
  - Signal values/transitions are synchronized with cursor time
  - Select signal(s) and go to next or previous transitions
    - **Any Change**, **Rising** or **Falling**

Search previous/next





Simulation value will be attached

SpringSoft Accelerating Engineers

Operate on Signals

Perform Active Trace

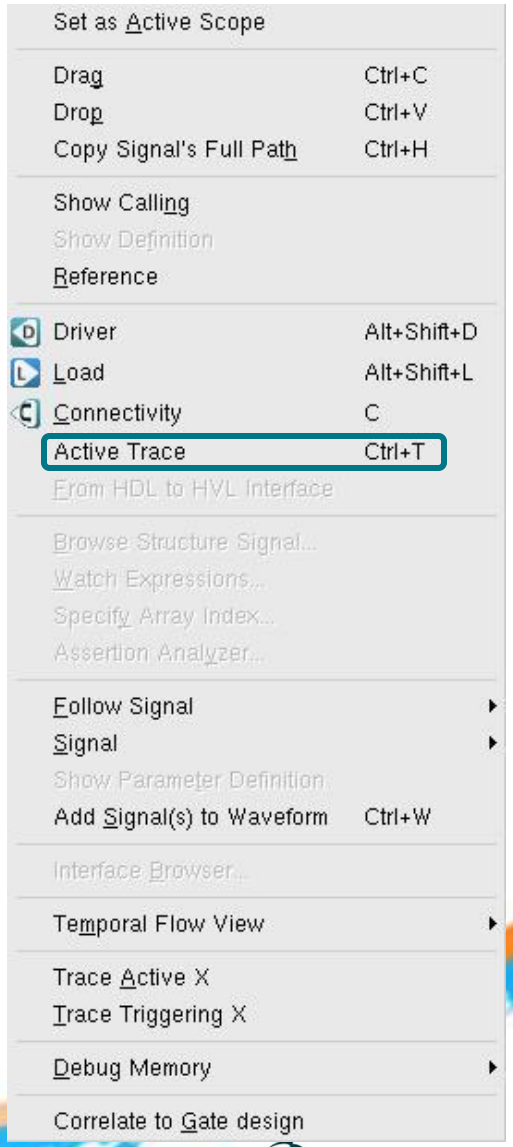
- Locate the active driver for a specific signal
  - Select one signal and invoke **RMB → Active Trace**
  - The actual driving statement at the cursor time will be displayed

```
49 always @(select or a or b or cin)
50    begin
51       #1 case (select)
52          `ADD:     {carry,out} = a + b + cin;
53          `SUB:     {carry,out} = a - b - 1 + cin;
54          `SUB1:    {carry,out} = b - a - 1 + cin;
55          `AND_OP:  {carry,out} = a & b;
56          `OR_OP:   {carry,out} = a | b;
```

```
29 */
30
31 `timescale 1 ns / 1 ns
32
33 module ALUB(IR, IDB, PC, CH, alu_mode, bus_mode, carry_mode, clock, reset,
34                S1, ALU, IXR,
35 input [1:0] IR;
36 input [7:0] IDB;
```

- system
  - CHILD1 (fsm_child1)
  - CHILD2 (fsm_child2)
  - CHILD3 (fsm_child3)
  - MASTER (fsm_master)
  - i_cpu (CPU)
    - i_ALUB (ALUB)
      - 3 primitives
      - i_alu (alu)
    - i_CCU (CCU)
    - i_PCU (PCU)
  - i_pram (pram)

RMB → Active Trace



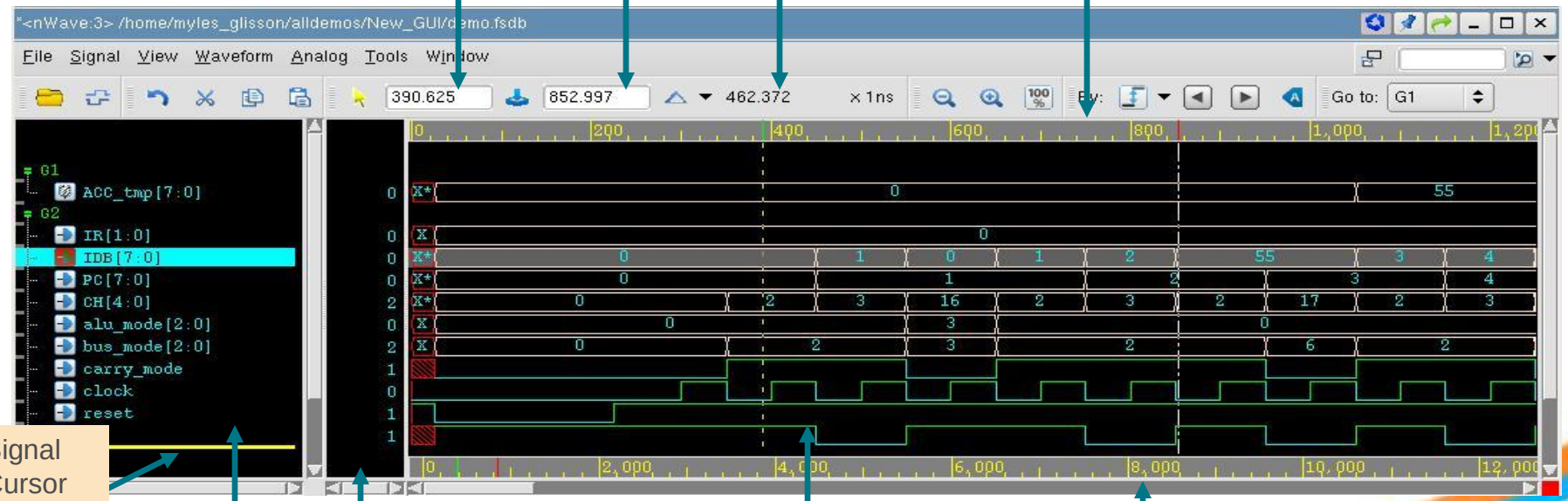
SpringSoft Accelerating Engineers

Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Open *nWave* Frame

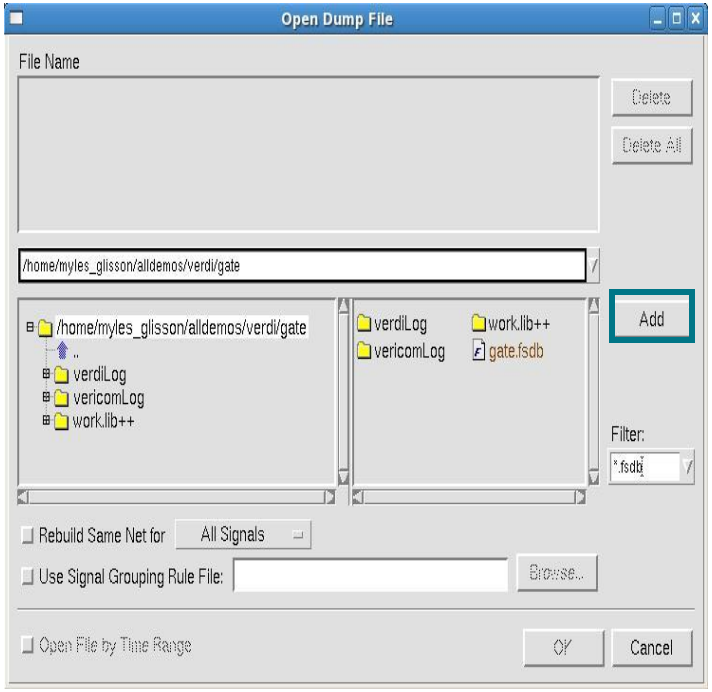
- In the main window, invoke **Tools → New Waveform** or click on the **New Waveform** icon to open *nWave* frame
  - The *nWave* frame is opened in the bottom frame of the main window
    - Click the **Be Window** icon can make it to be a standalone window



SpringSoft Accelerating Engineers

Load Simulation Results

- In *nWave*, use **File → Open** or toolbar icon to load results
  - Load FSDB Files generated by *$fsdbdumpvars*
  - Load VCD files generated by *$dumpvars*
    - VCD files are automatically converted to FSDB at load time
  - Select one or more files to **Add** and open in the same waveform window
    - Use **File → Set Active** to specify which file is active
  - Opening several files in different *nWave* frames/windows is allowed
    - Use **Window → Change to Primary** to specify which window is active
  - The compressed FSDB and VCD file are supported
    - The file name has to be ***.fsdb.gz**, ***.vcd.gz**, or ***.dump.gz**



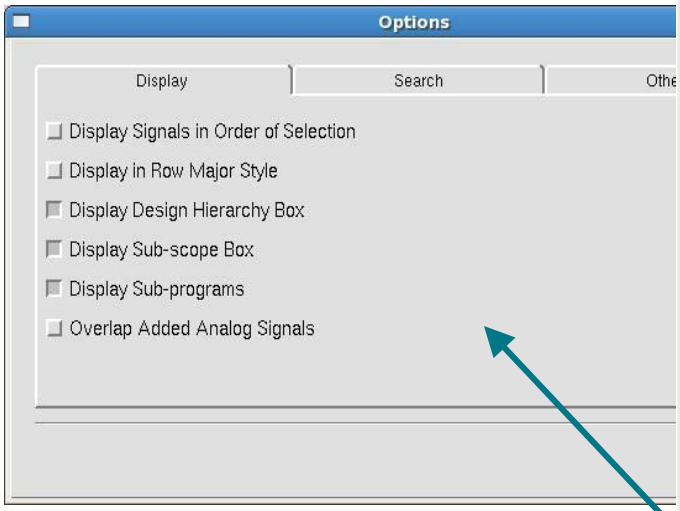
SpringSoft Accelerating Engineers

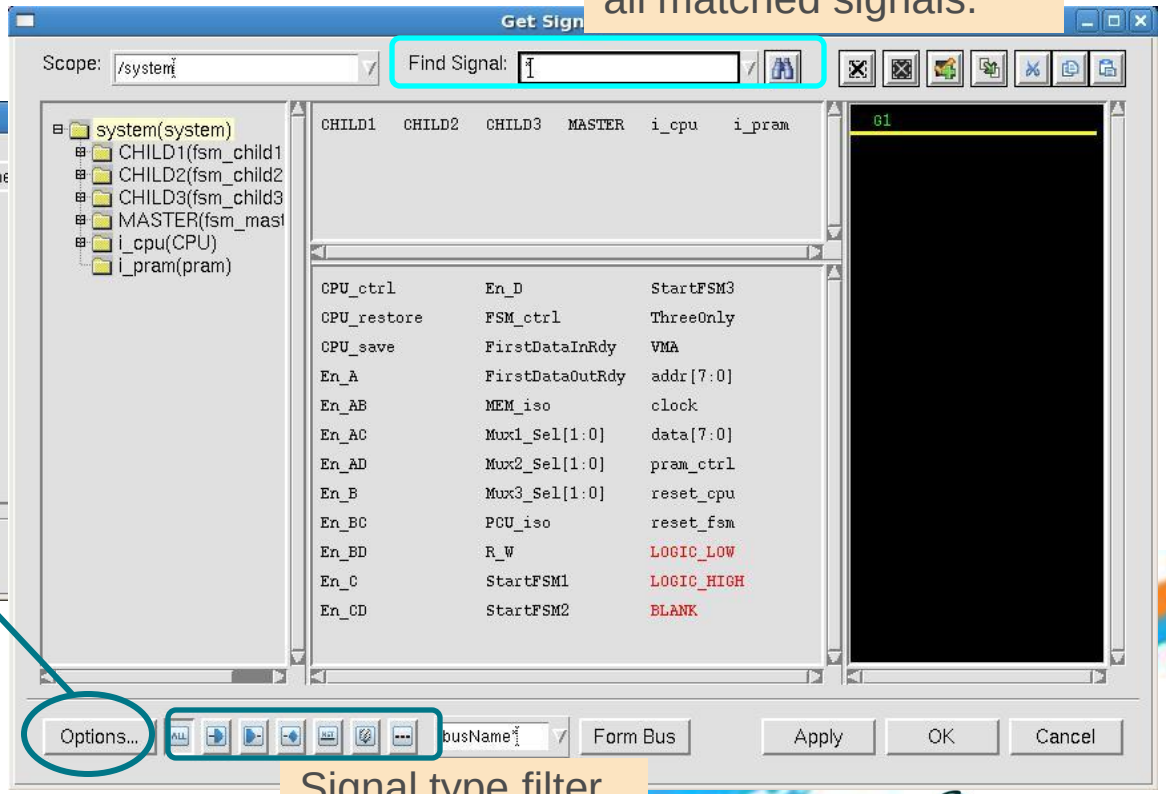
Add Signals

Get Signals from nWave

- Use **Signal → Get Signals** or toolbar icon
  - Click **Options** button to specify conditions
    - For example to specify the display order or type filter

Use wildcard *,? to get all matched signals.





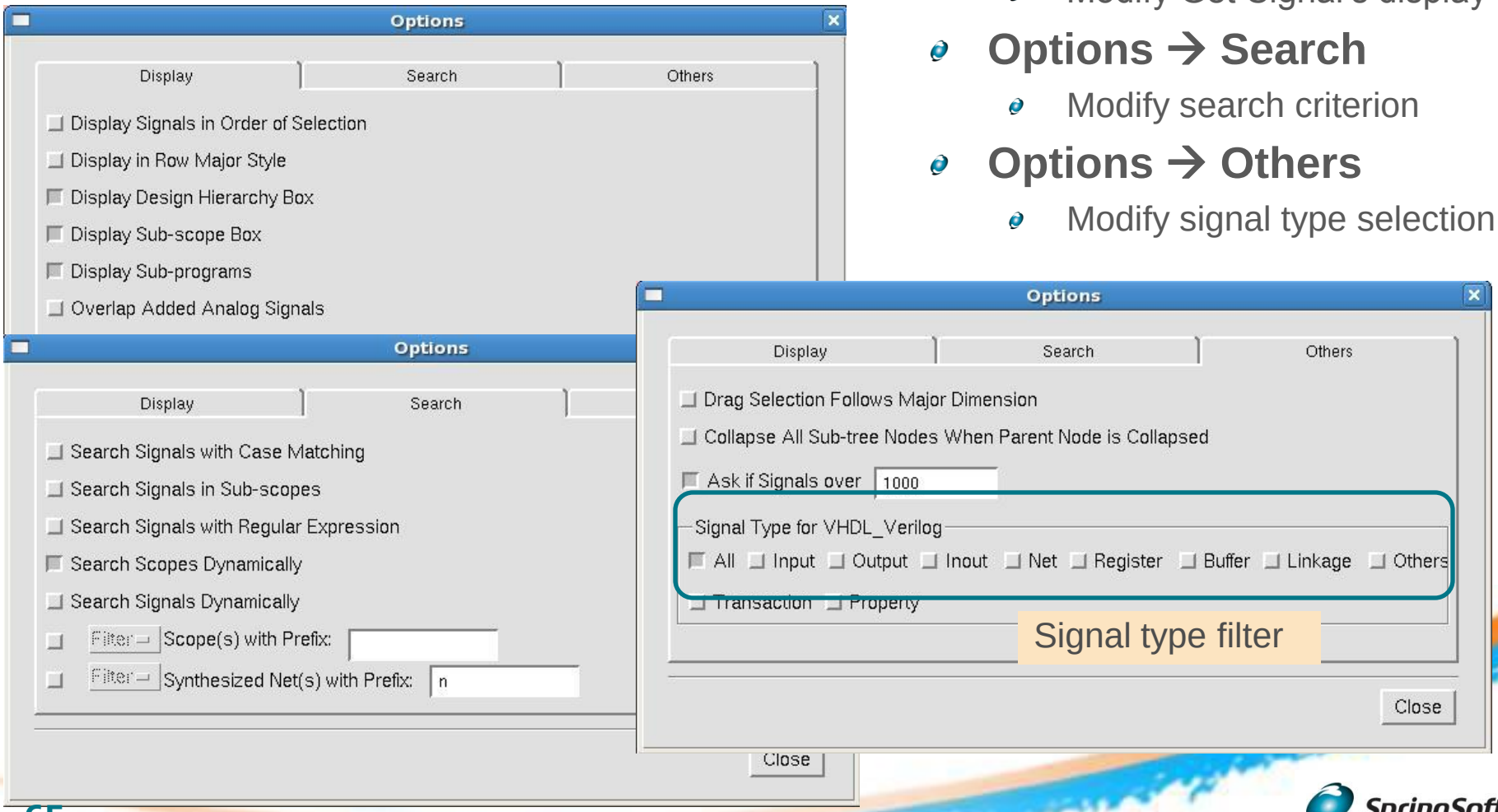
Signal type filter

SpringSoft Accelerating Engineers

Add Signals

Get Signals from nWave

- Use **Signal → Get Signals → Options**
- **Options → Display**
  - Modify Get Signal's display
- **Options → Search**
  - Modify search criterion
- **Options → Others**
  - Modify signal type selection

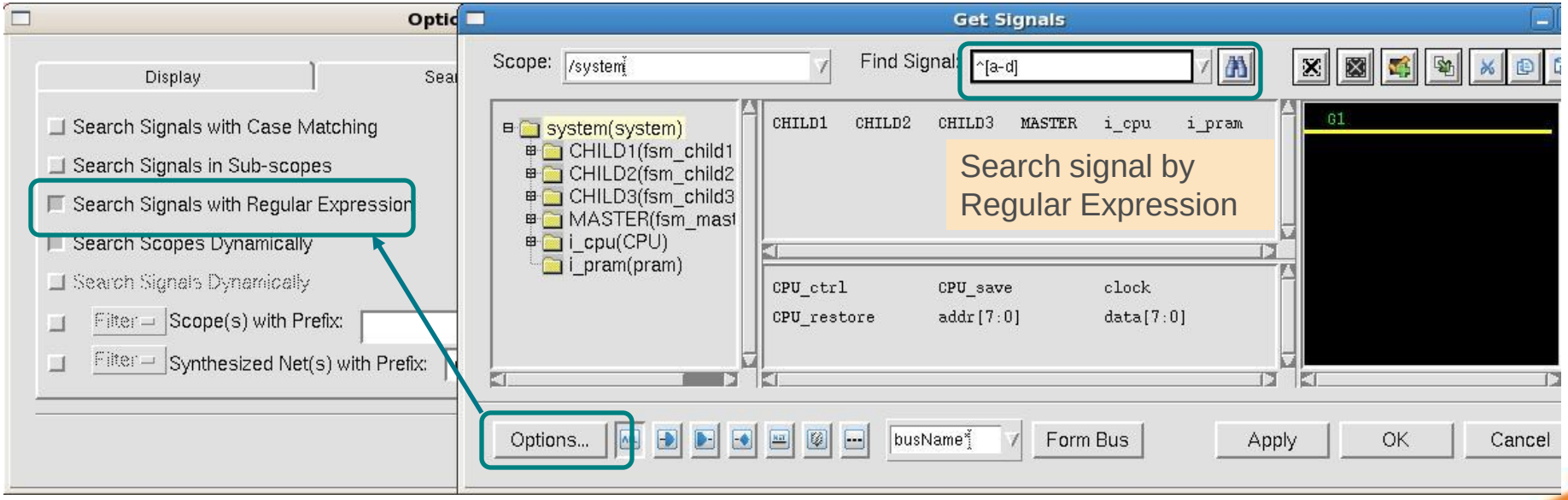


SpringSoft Accelerating Engineers

Add Signals

Search Signals with Regular Expression

- Click **Options** button in *Get Signals* form
- Turn on the **Search Signals with Regular Expression** option in the **Search** tab



SpringSoft Accelerating Engineers

Add Signals

Add Signals from Other Windows

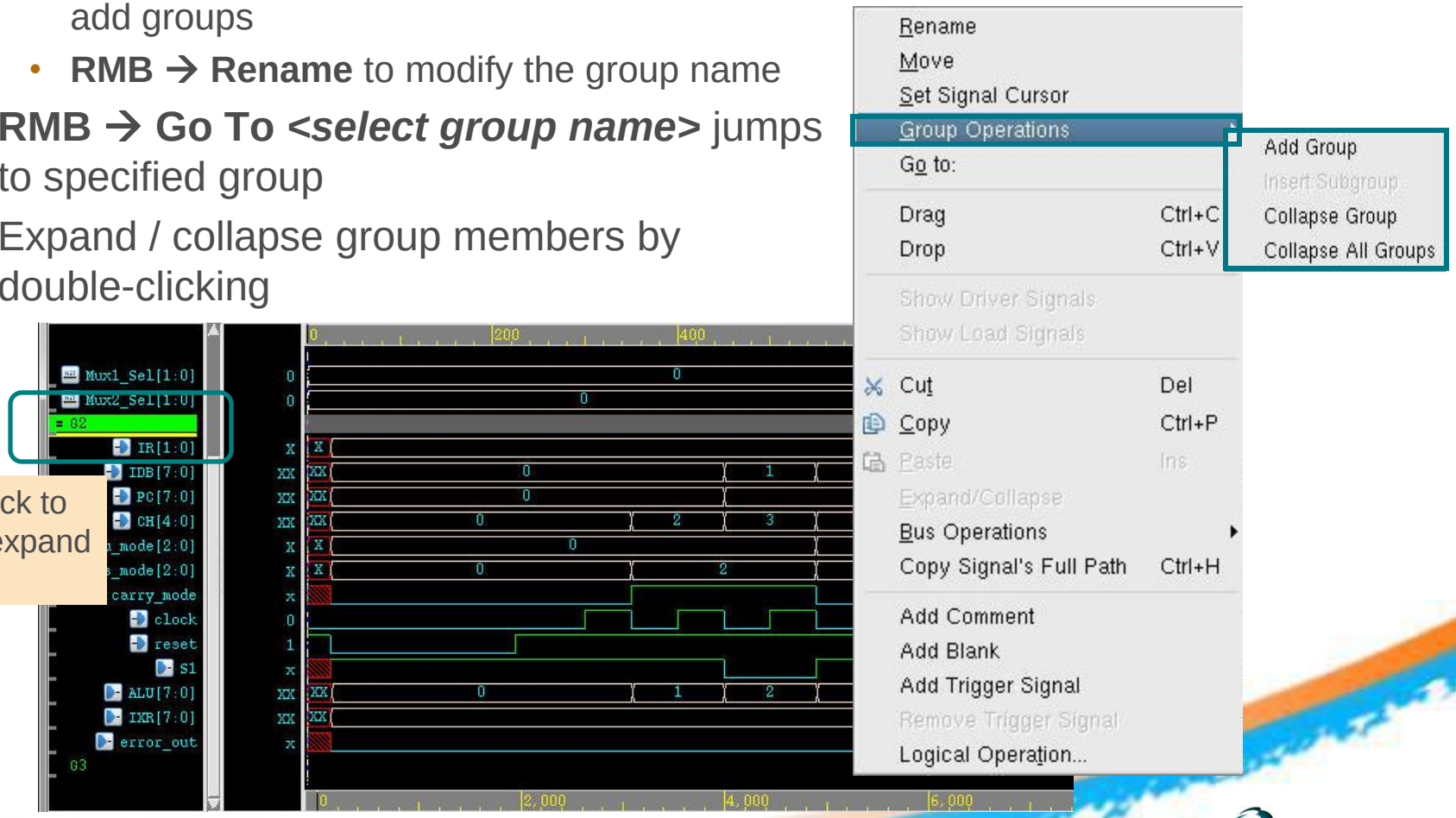
- D&D signals/instances from other frames/windows to *nWave*
  - All instance I/O
  - Selected signal(s)
  - Signals in text region
  - Schematic hierarchical blocks or logic gates

- Select signals and use **RMB → Add Signal(s) to Waveform** in *nTrace*
- Select objects and use **RMB → Add Selected to Waveform** in *nSchema*
- After executing a trace command in *nSchema*, use **Trace → Add Results to Waveform**
  - The command is available in *Full Hierarchical Window* only

Organize Signals

Create Group for Signals

- Select a group in signal window:
  - **RMB → Group Operations → Add Group** to add groups
  - **RMB → Rename** to modify the group name
- **RMB → Go To <*select group name*>** jumps to specified group
- Expand / collapse group members by double-clicking



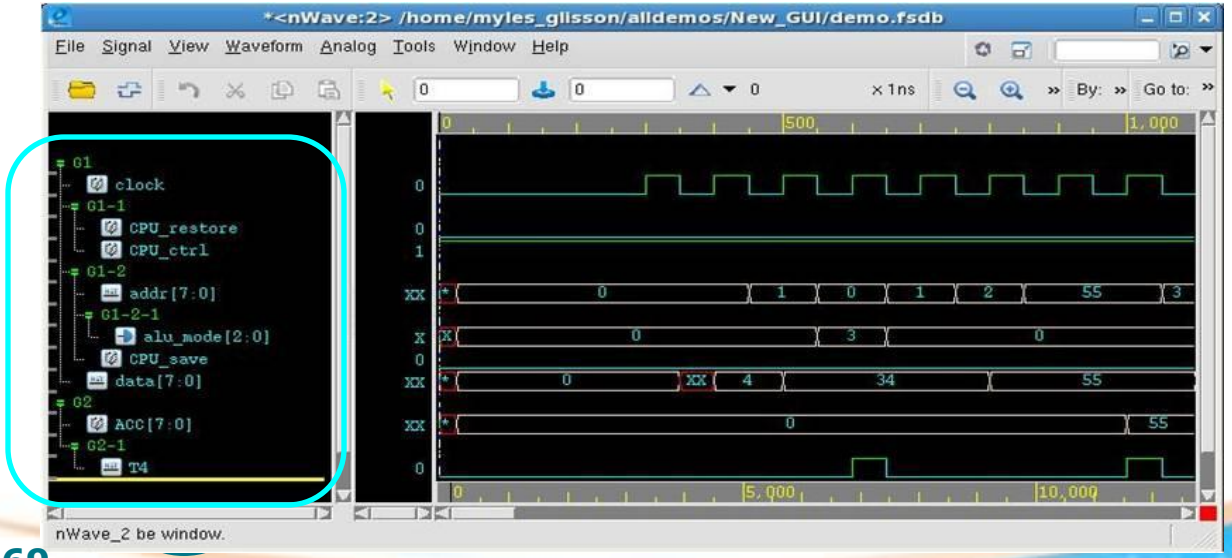
Double click to collapse/expand groups

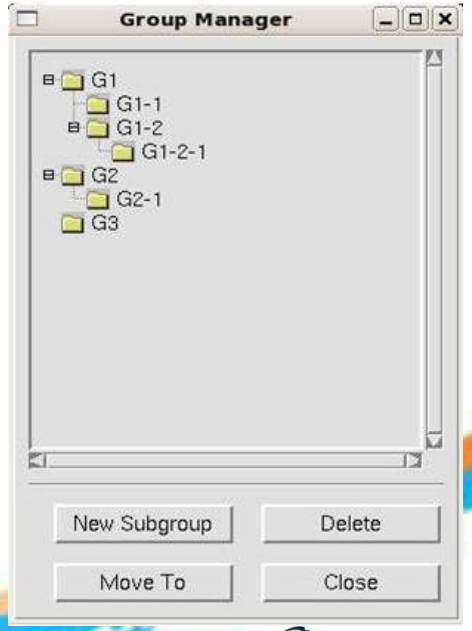
SpringSoft Accelerating Engineers

Organize Signals

Group Operations

- Invoke **RMB → Group Operations → Insert Subgroup** to create sub-groups of selected groups
- Each group/sub-group can be renamed with **RMB → Rename**
- Invoke **View → Group Manager** to open the *Group Manager* form
  - Select a group in *Group Manager* form and click **New Subgroup** to add a new group under the selected group
  - Click the **Move To** button to move the selected group/signals in *nWave* into the selected group in the *Group Manager*
  - Delete the selected group and subgroups

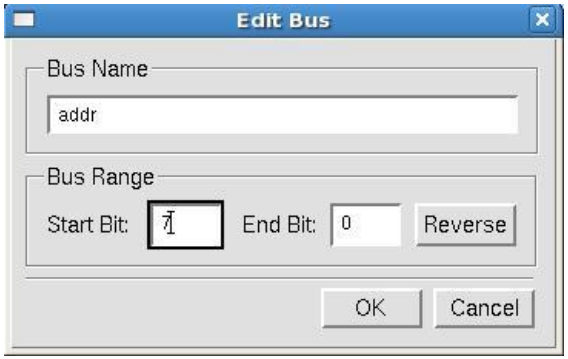


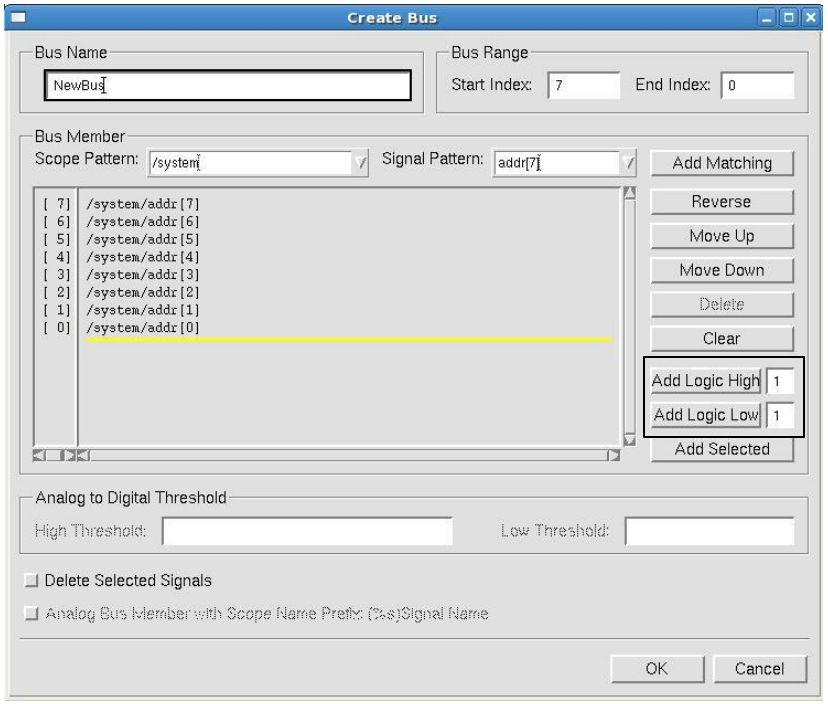


Operate on Signals

Create and Edit Buses

- Invoke **Signal → Bus Operations -> Edit Bus** or **RMB → Bus Operations → Edit Bus** in signal pane to edit selected bus





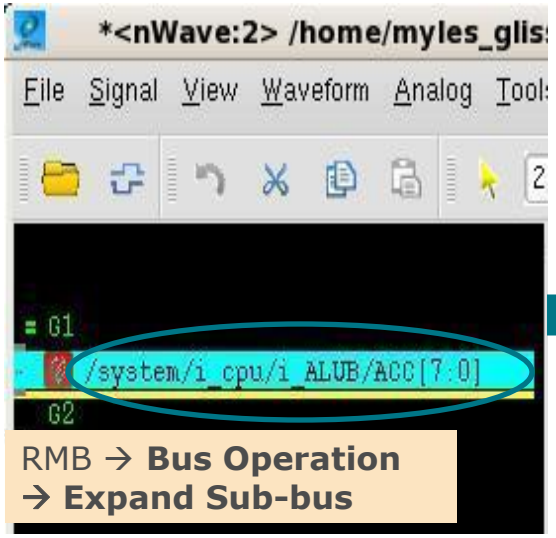
- Create bus from selected signals
  - Invoke **Signal → Create Bus** or **RMB → Bus Operations → Create Bus** in signal pane
  - Add Logic 0 or Logic 1 as place holders

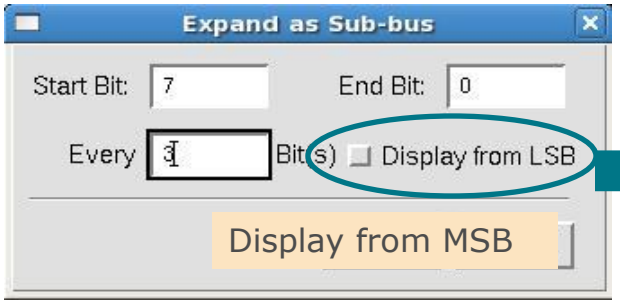
SpringSoft
Accelerating Engineers

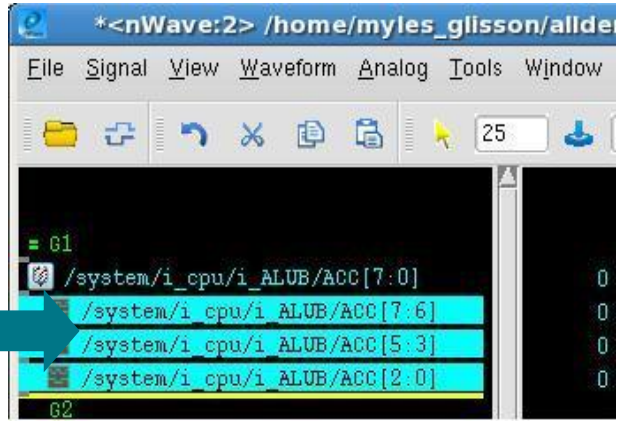
Operate on Signals

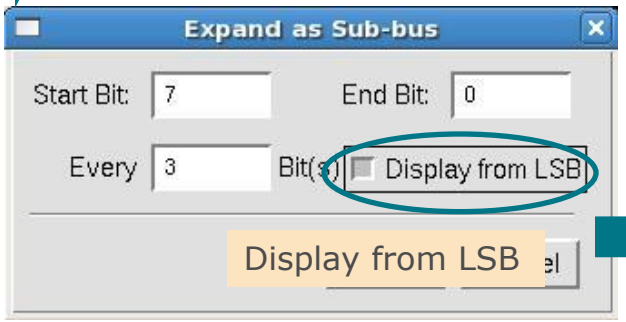
Expand Buses

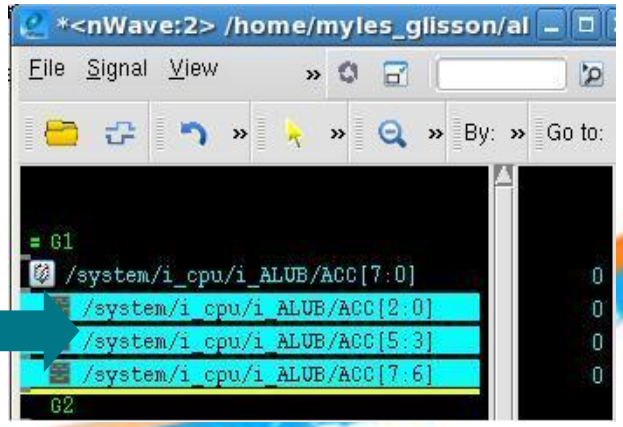
- Expand / Collapse bus by double-clicking
- Select a bus in the *nWave* signal pane and invoke **RMB → Bus Operation → Expand Sub-bus** to open *Expand Bus* form
  - Specify number of bits to divide the bus by
  - Choose to display from LSB or MSB







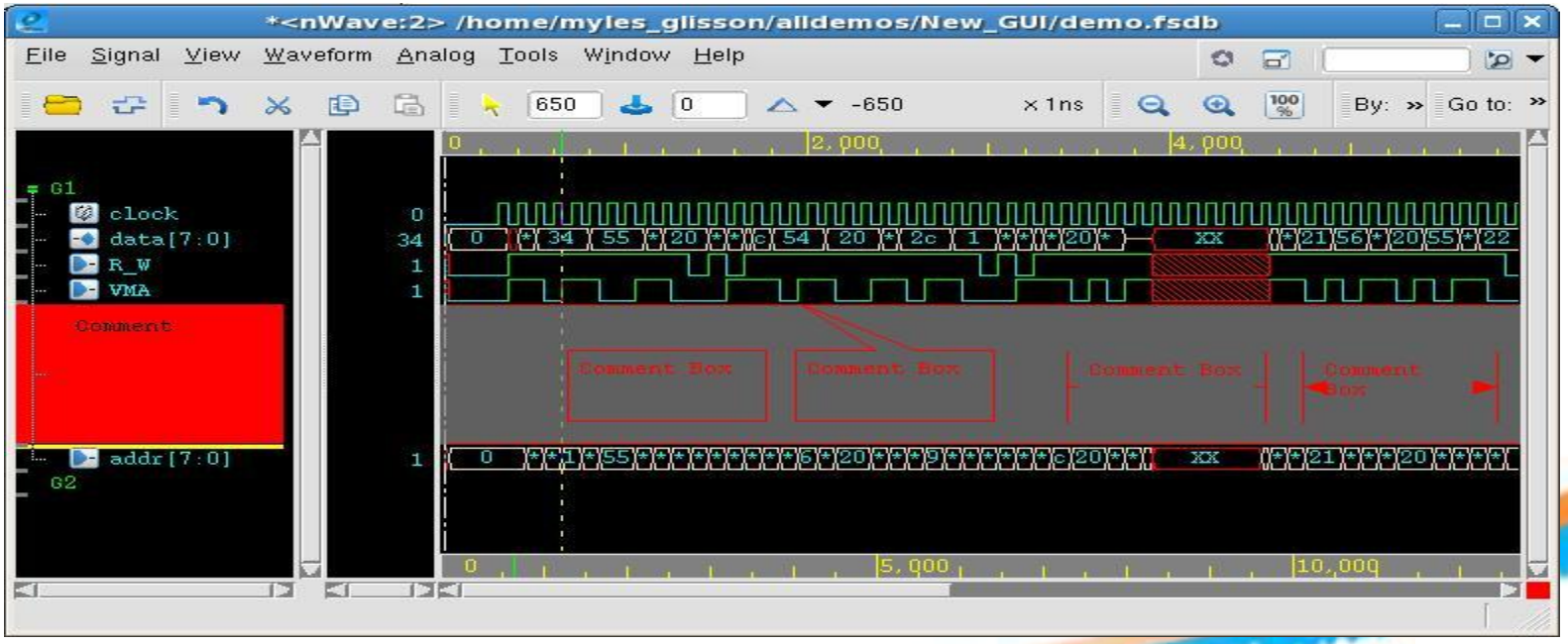




Operate on Signals

Add Comments to Signals

- Use **Signal → Comment → Insert** to insert a comment field at cursor position
- Use **Signal → Comment → Add Square Box** or other sub-commands to add different types of comment boxes
  - Four comment box types are provided: **Square Box**, **Attached Square Box**, **Period Box** and **Arrow Period Box**

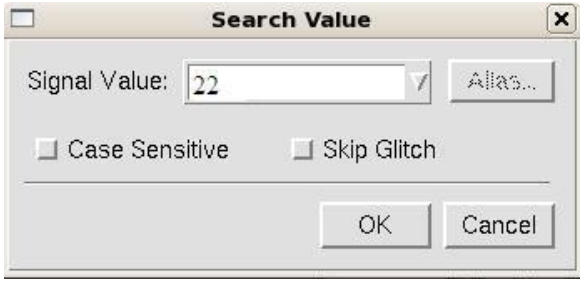


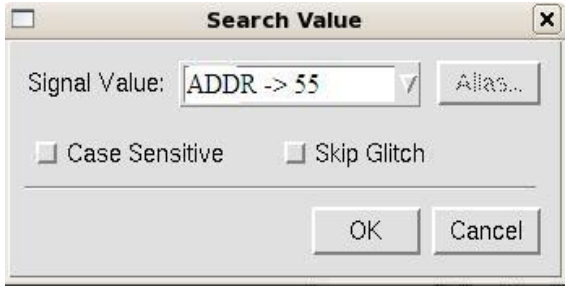
SpringSoft Accelerating Engineers

Operate on Signals

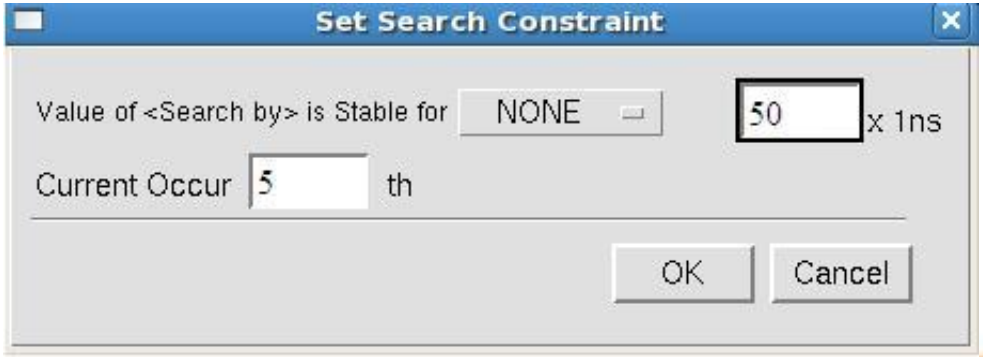
Search for Values

- Use **Waveform → Set Search Value** to search signal values
  - Search value accepts mnemonics, wild cards and transitions





- Use **Waveform → Set Search Constraint** to constrain search to a duration of time (set to <= 0 for glitches) or nth occurrence of a condition

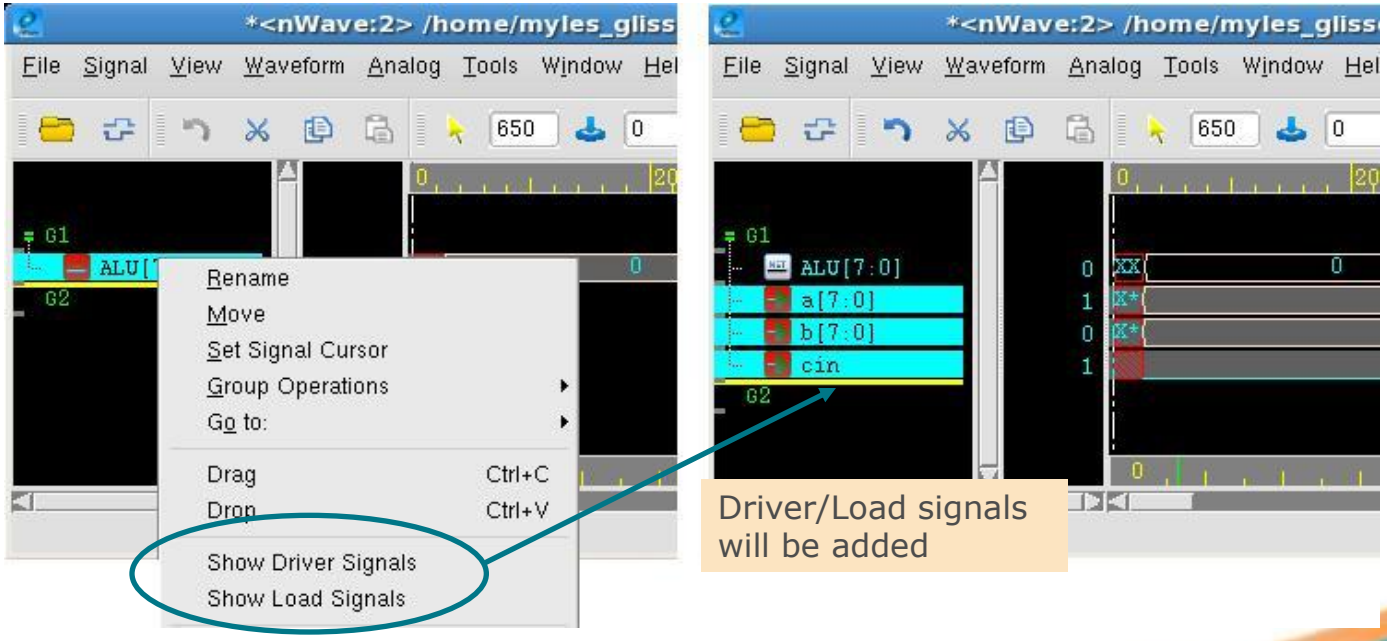


SpringSoft
Accelerating Engineers

Operate on Signals

Show Driver and Load Signals

- Select a signal in *nWave* signal pane, invoke **RMB → Show Driver Signals** or **Show Load Signals** commands to trace driver or load
  - Driver/load signals will be added in signal pane
  - The command is only valid when the design is also imported – not valid in standalone *nWave*

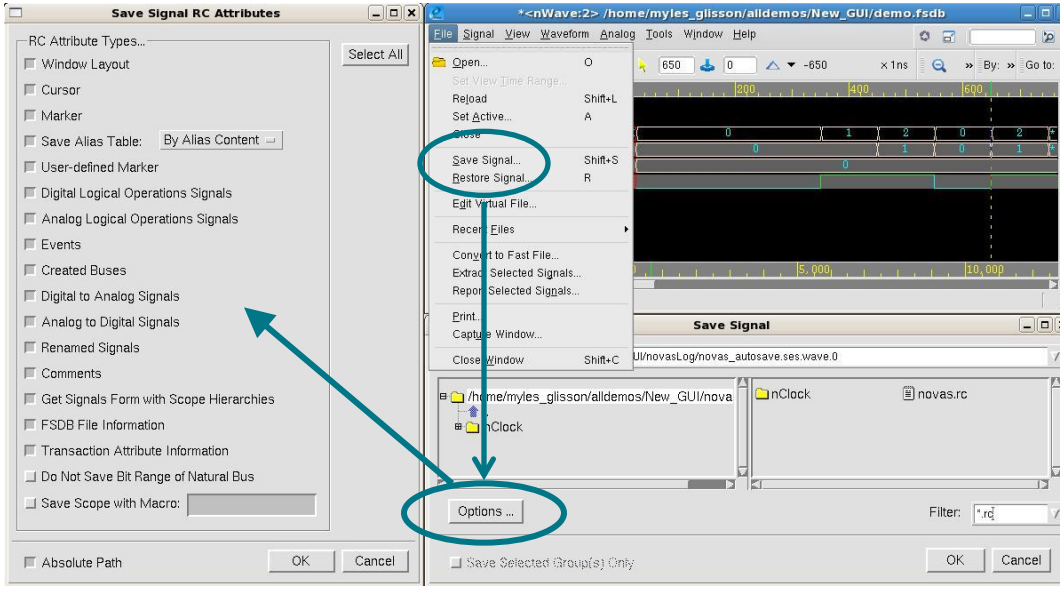


SpringSoft Accelerating Engineers

Operate on Signals

Save and Restore Signals

- Use **File → Save Signal** to save all signals to a loadable file
  - Click **Options** button to specify signals attributes
  - **NOTE**: turn on **Do Not Save Bit Range of Natural Bus** option when debugging parameterized designs (with the same signal name but different bus range)



- In *nWave*, use **File → Restore Signal** to load the saved file
- On the command line, use **–sswr <file>.rc** to restore

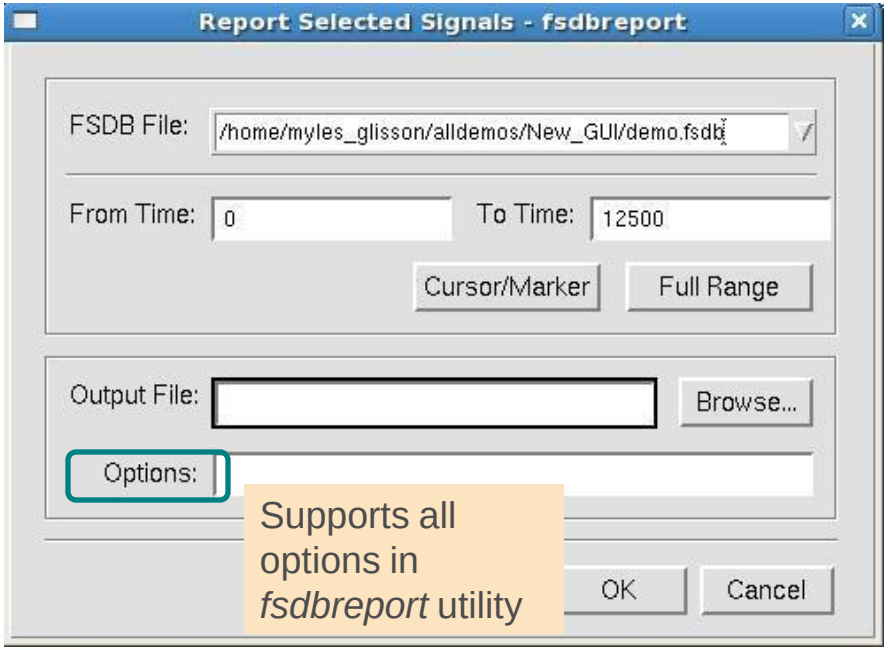
NOTE: If there is a FSDB file open in *nWave*, the **Restore Signal** command will not overwrite the loaded simulation results

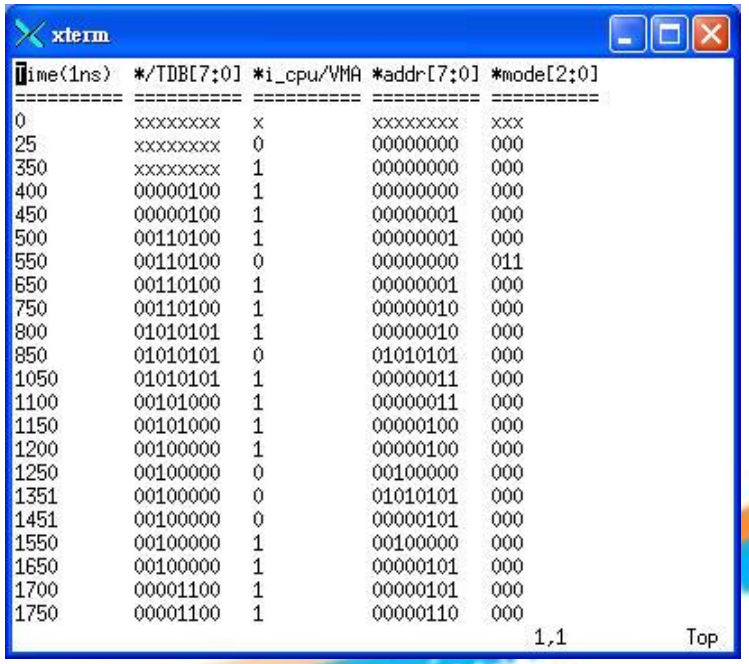
SpringSoft Accelerating Engineers

Operate on Signals

Report Selected Signals

- Select signals in *nWave*, invoke the command **File → Report Selected Signals**
  - Report selected signals to an *ACSII* file
  - Will actually invoke **fsdbreport** utility in batch
  - **NOTE**: Selected signals should belong to the same FSDB file



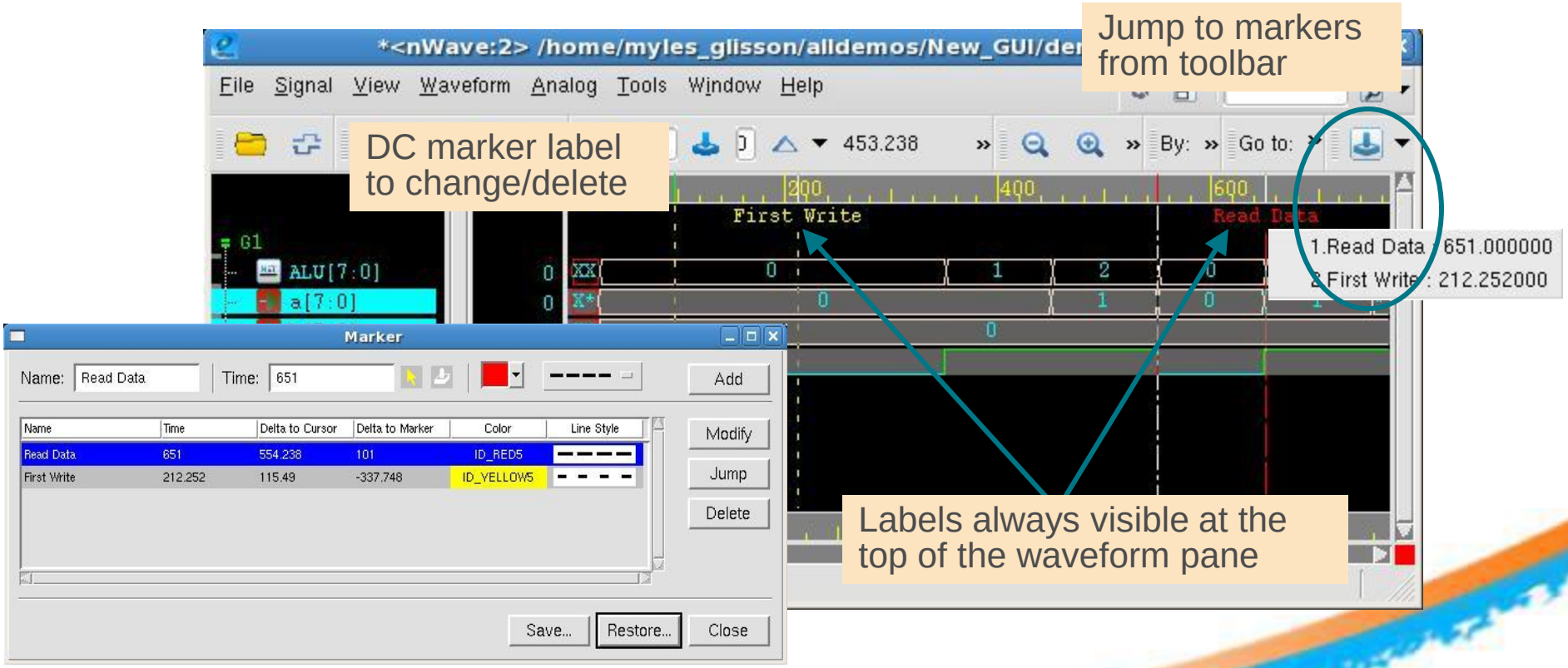


SpringSoft Accelerating Engineers

Manipulate Waveform Display

Set Markers

- Use **Waveform → Marker** to name and place markers
- Multiple markers are supported, markers can be saved and restored
- Specify name, color, and line style for labels to differentiate markers



SpringSoft Accelerating Engineers

Manipulate Waveform Display

Waveform Alias (1/3)

- Load design from file:
  - Use **–autoalias** option on Verdi command line for automatic mnemonic recognition for **'defines** and parameters

    *% **verdi -f run.f –autoalias***

- Load design from library:
  - Use **aliasextract** to extract an alias file from a compiled library. Default file name is *extracted.alias*

    *% **vericom –f run.f –lib work***

    *% **aliasextract -lib work***
  - Use **–aliasFile** option on Verdi command line to load an alias file

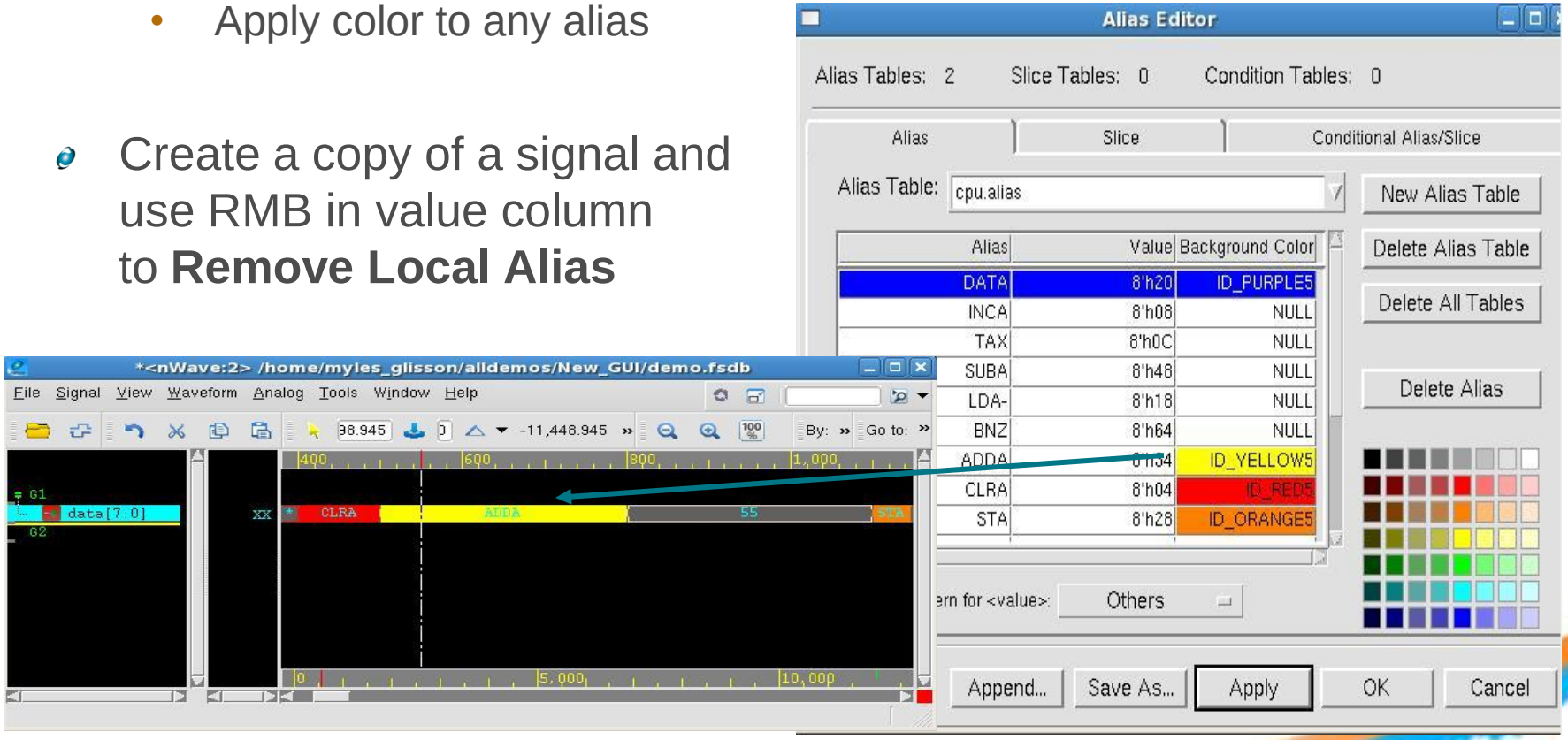
    *% **verdi -top system -aliasFile extracted.alias***

SpringSoft
Accelerating Engineers

Manipulate Waveform Display

Waveform Alias (2/3)

- Use **Waveform → Signal Value Radix → Edit Alias** to create/update an alias file and apply to a signal
  - Apply color to any alias
- Create a copy of a signal and use RMB in value column to **Remove Local Alias**

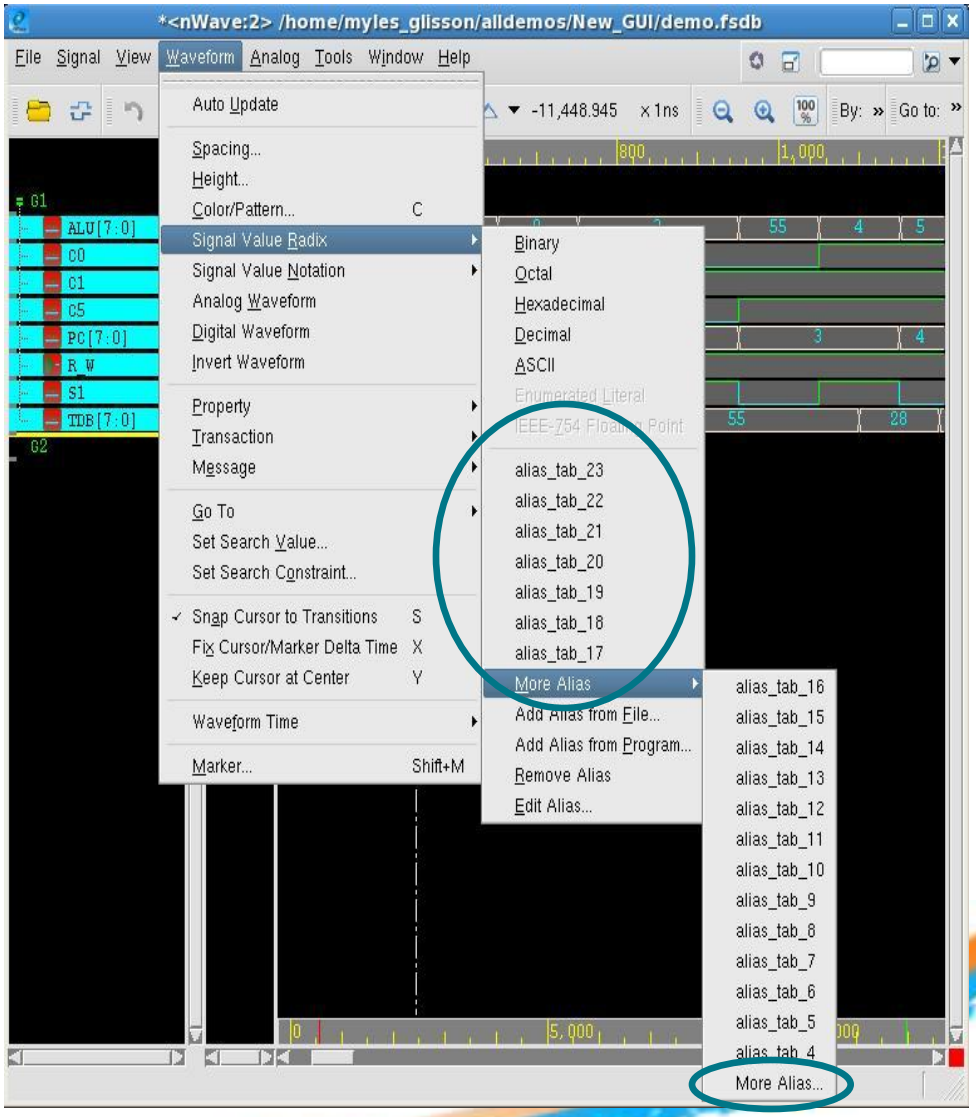


SpringSoft Accelerating Engineers

Manipulate Waveform Display

Waveform Alias (3/3)

- Supports up to 20 aliases in a list
  - Select **More Alias** in the first level to see the second level aliases
  - Select **More Alias** in the second level will bring up ***Alias Editor***
    - No number limitation in this form

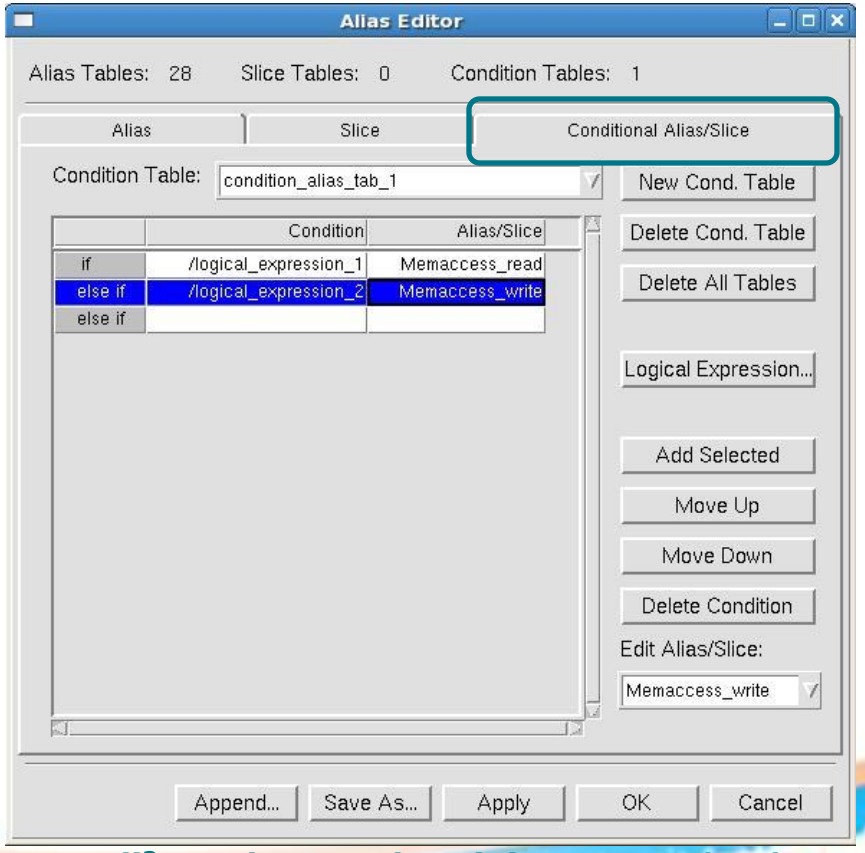


SpringSoft Accelerating Engineers

Manipulate Waveform Display

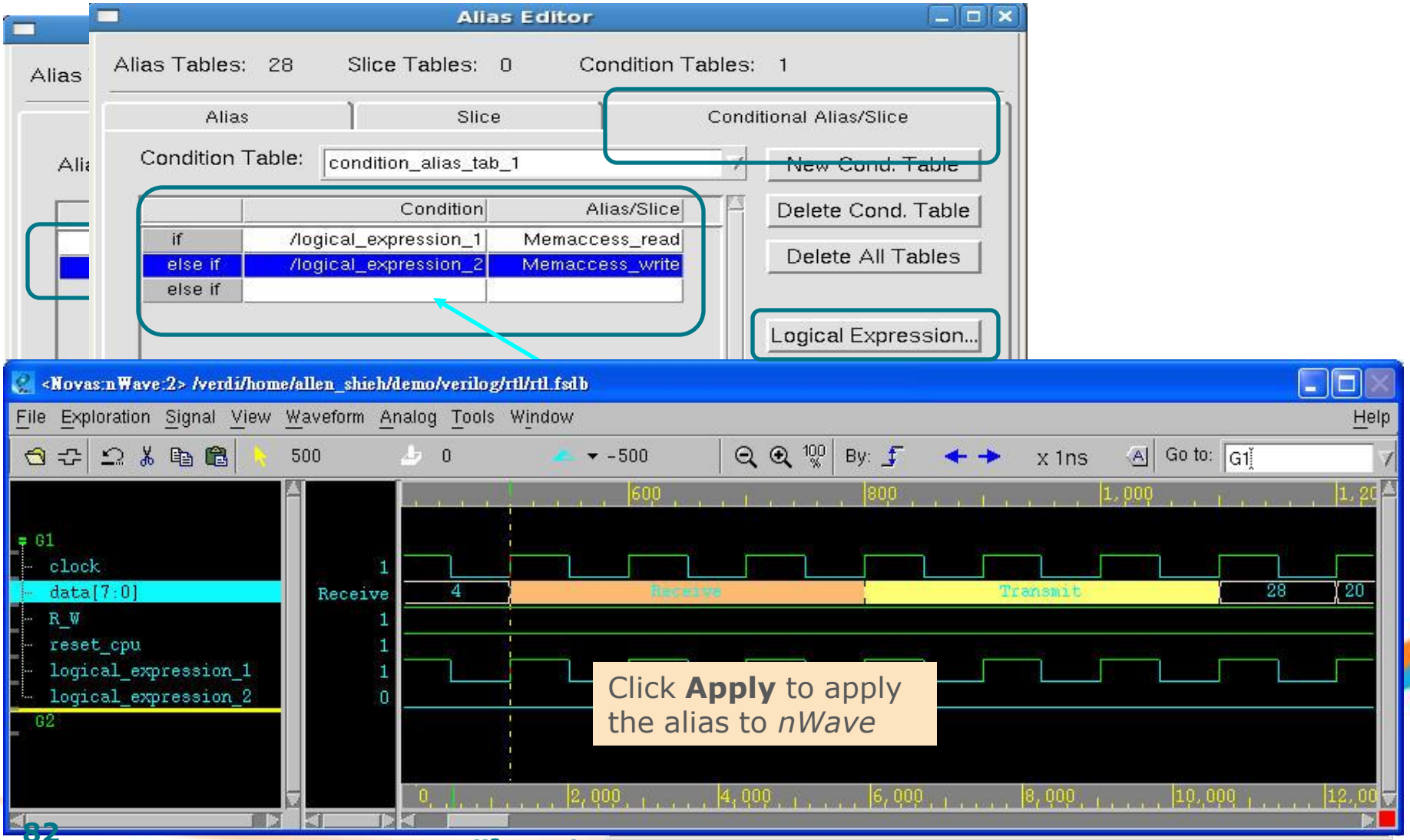
Conditional Alias (1/2)

- In *nWave*, invoke **Waveform → Signal Value Radix → Edit Alias** command to open the *Alias Editor*
  - Click **Conditional Alias/Slice** tab to specify conditions for applying multiple alias tables



Manipulate Waveform Display

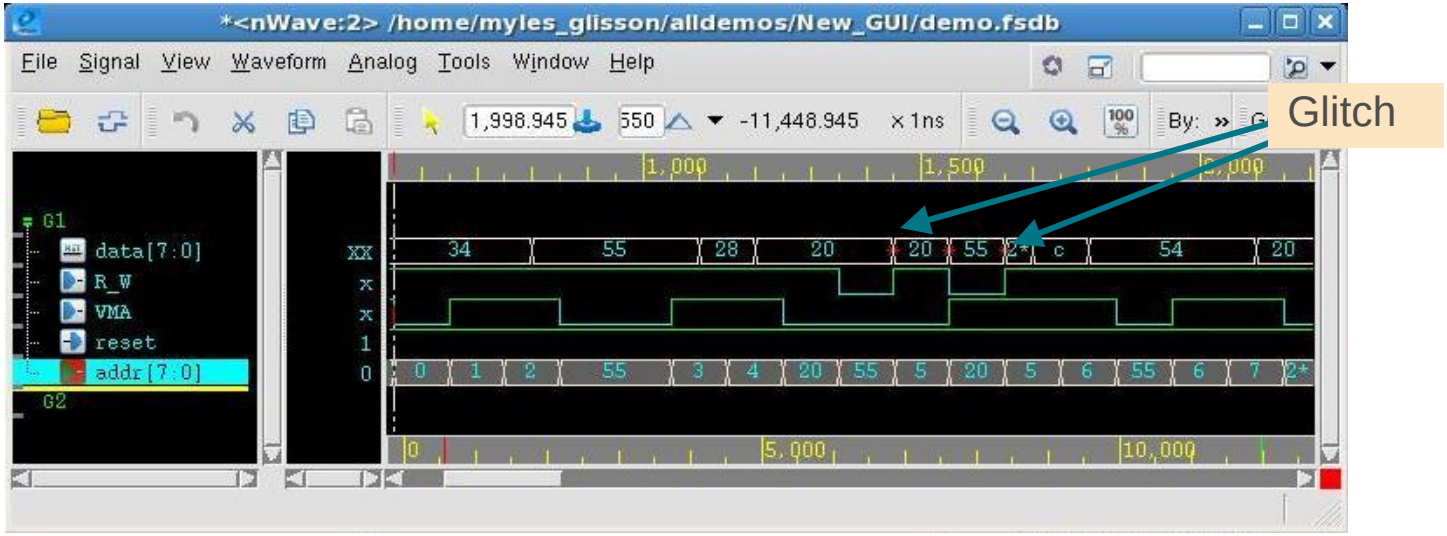
Conditional Alias (2/2) - Example



Manipulate Waveform Display

Display Glitches

- Use **View → Display Glitch** to identify glitch location



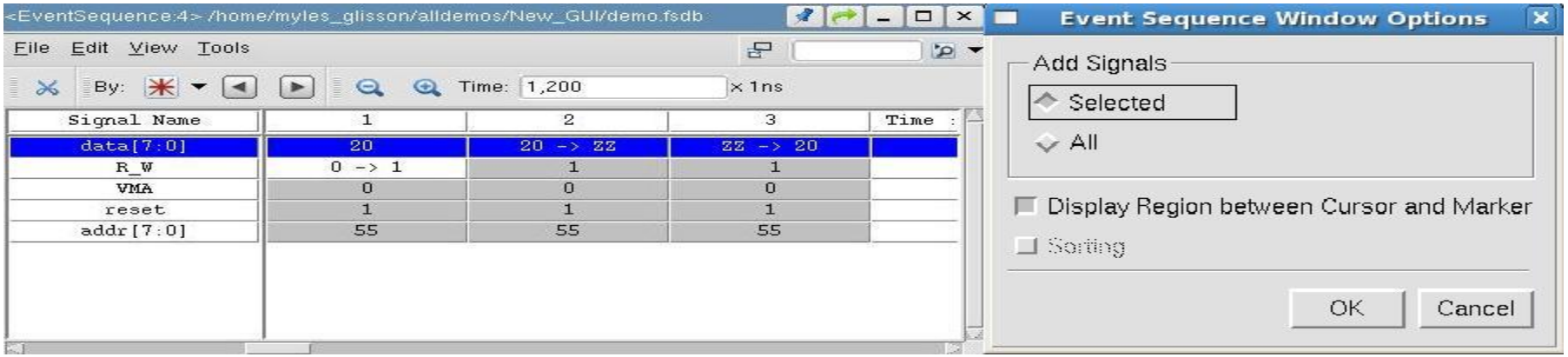
- Use **Waveform → Set Search Constraint** to search for glitch events
  - Set **Value of <Search By> is stable for** to **<=** and enter **0** in the time field

NOTE: Simulation results file must be generated with the following environment variable set:
setenv NOVAS_FSDB_ENV_MAX_GLITCH_NUM 0

Manipulate Waveform Display

Event Sequence

- Use **Tools → Event Sequence** to display order of events on selected signals in a single time step
  - Sync time with cursor
  - Drag and drop additional signals as needed
  - Switch display between waveform and grid mode

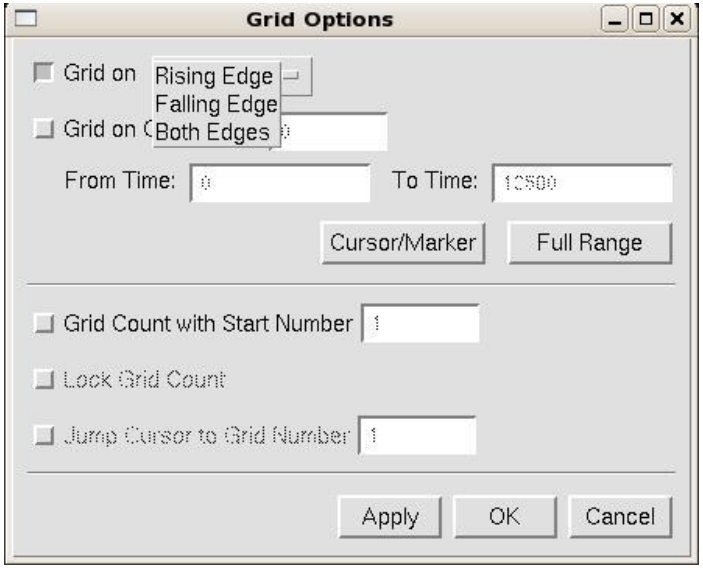


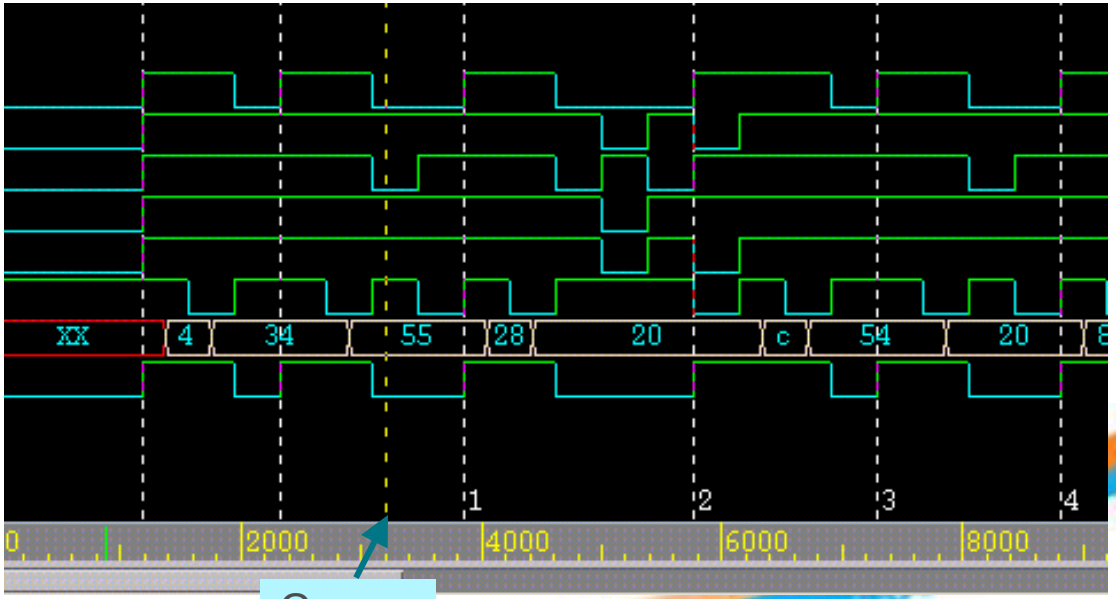
NOTE: Simulation results file must be generated with the following environment variable set:
setenv NOVAS_FSDB_ENV_DUMP_SEQ_NUM on

Manipulate Waveform Display

Display Grid

- In *nWave*, use **View → Grid Options** to open the *Grid Options* form
  - Add grids based on the rising/falling edges of the selected signal
  - Add grids based on user defined cycle in a specific time range
  - Count number for grids
    - Can specify starting number
    - Counting will start on the first grid after the current cursor time





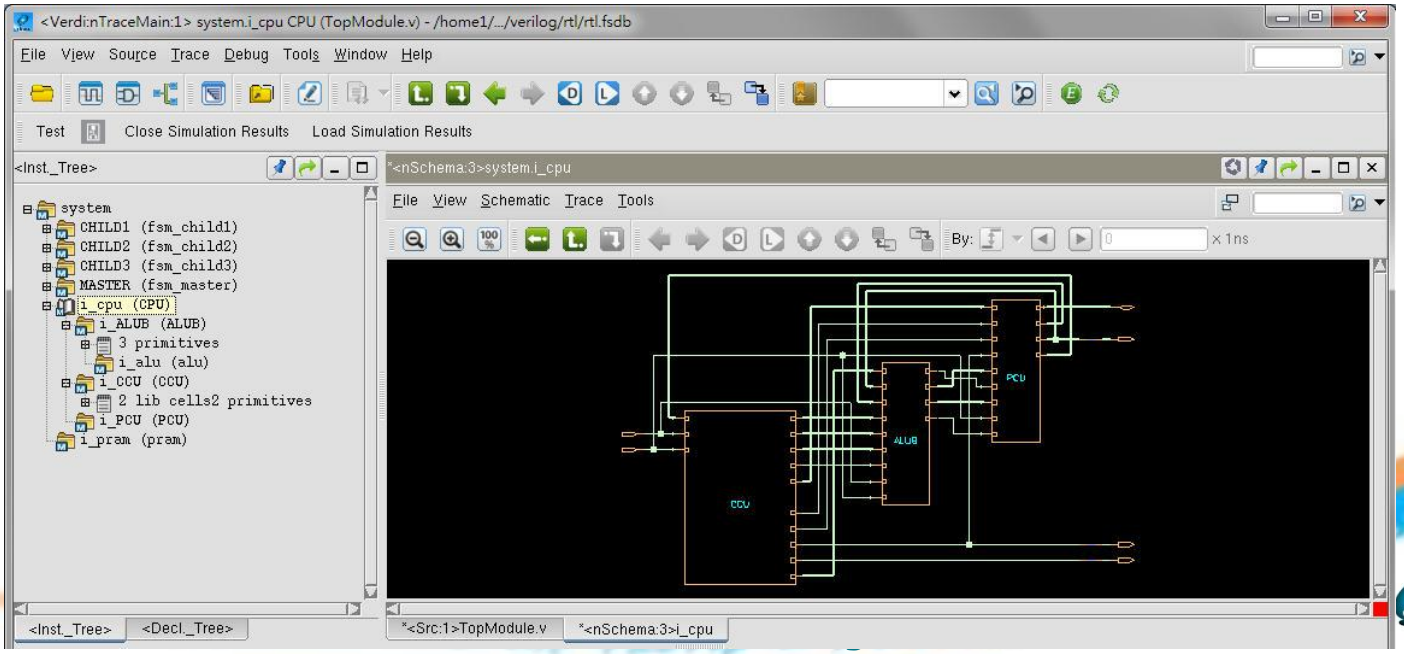
Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Open *nSchema*

Open a Full Hierarchy Window – Default Schematic View

- Select an instance in the *nTrace* design browser and click the **New Schematic** icon
- Or, **D&D** an instance from the design browser or a Verilog module or VHDL architecture from the *nTrace* source code frame to **New Schematic** icon
- Or, execute **Tools → New Schematic → Current Scope** in *nTrace*
  - The *nSchema* frame opens in the right frame of the main window
    - Click the **Be Window** icon can make it to be a standalone window

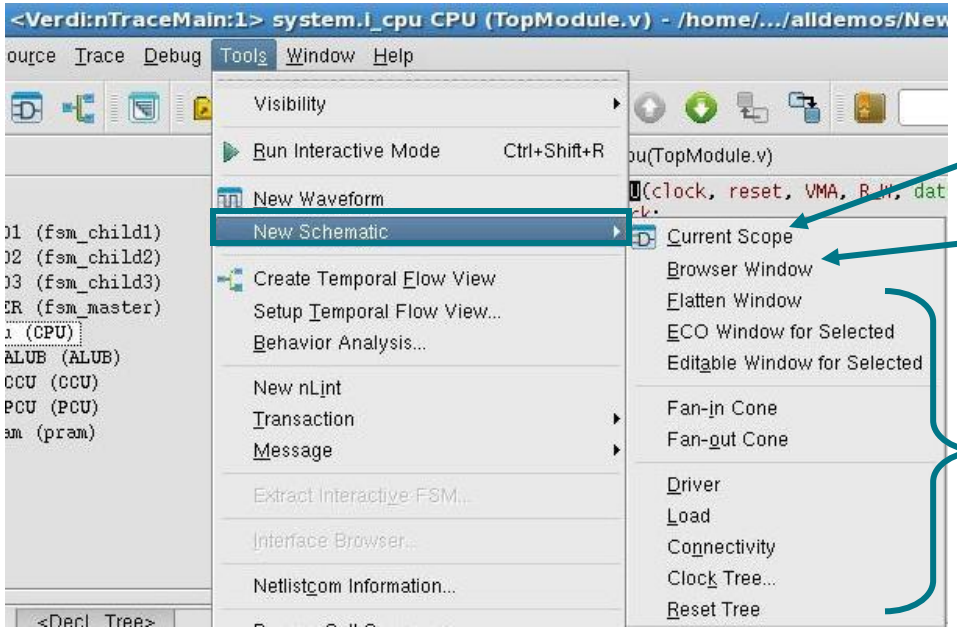


SpringSoft
Accelerating Engineers

Open *nSchema*

Different Schematic Window Types

- **Full Hierarchical Window** – Shows complete objects in specific scope. Default schematic view
- **Browser Window** – Shows partial schematic in specific scope
- **Flatten Window** – Shows partial schematic in flatten view, schematics in the *Flatten Window* cross all scopes



Full Hierarchical Window

Browser Window

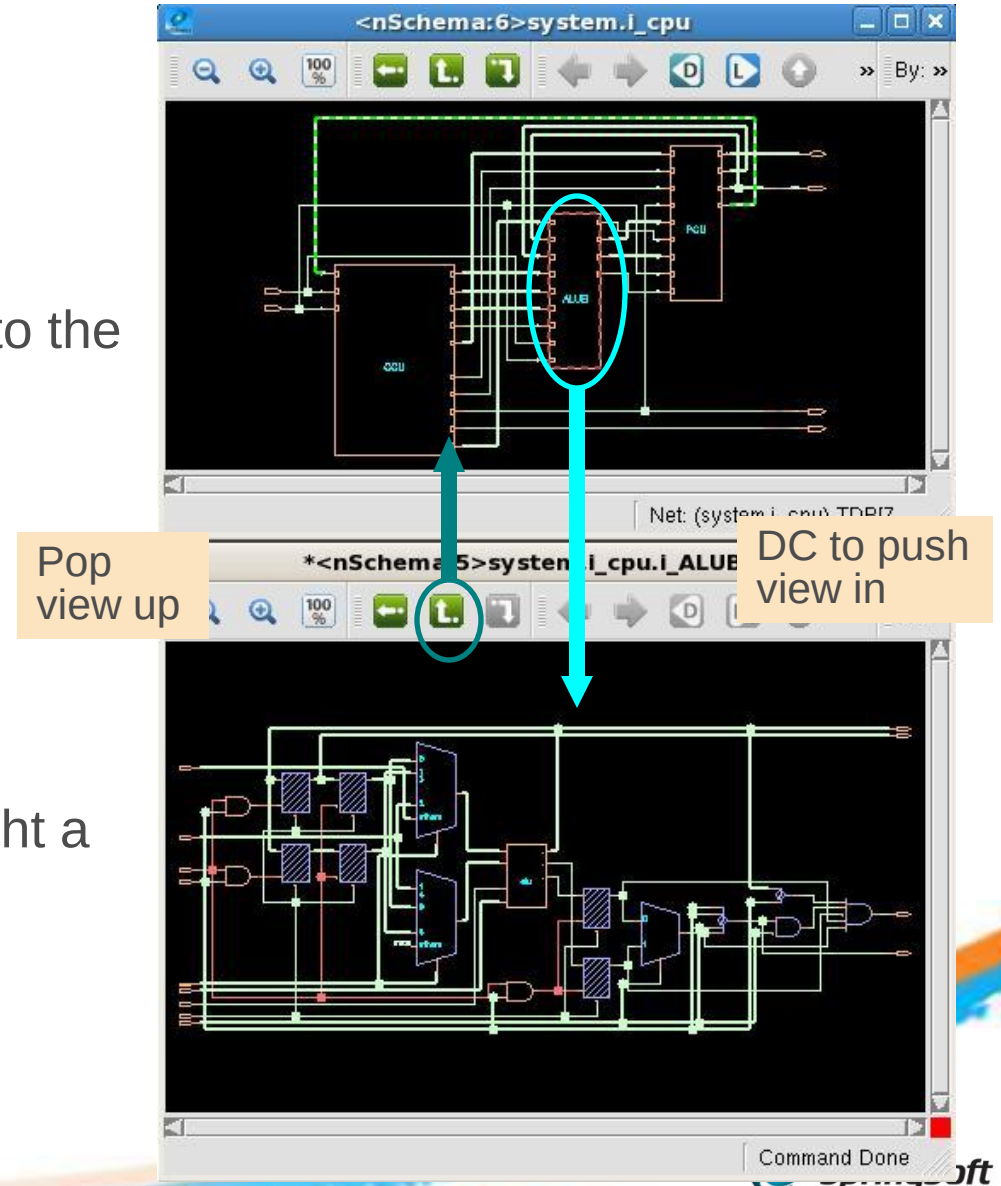
Flatten Window

SpringSoft Accelerating Engineers

Open *nSchema*

Full Hierarchical Window

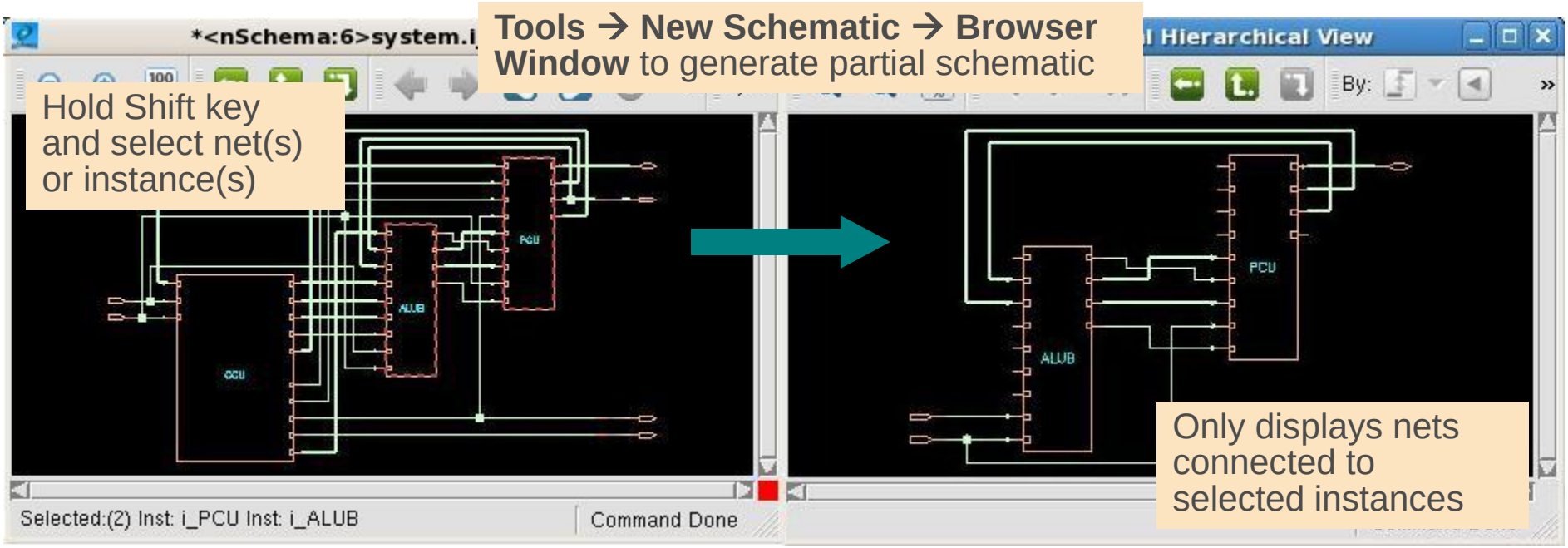
- The complete design can be traversed hierarchically
- **DC** a symbol port to push into the lower level and highlight the associated net
- **DC** on a net to highlight all instances that connect to it
- Trace commands will highlight a variety of different objects in current window



Open *nSchema*

Browser Window

- Focus on specified instances or nets for debugging



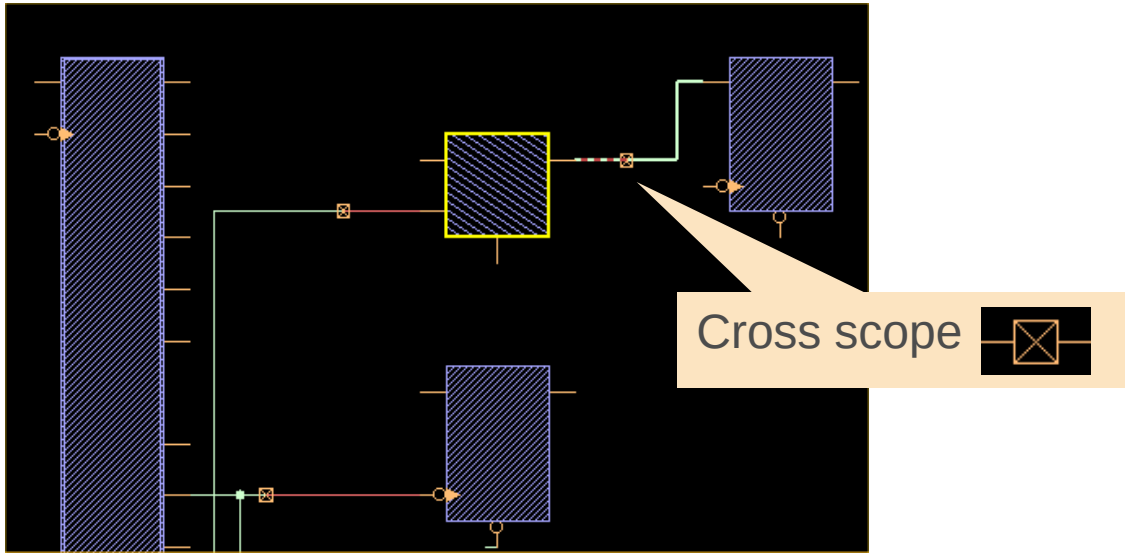
- DC on symbol port to push in, pop up or expand logic
- Use Delete key or to remove object(s)
- Undo / redo edits
- No trace commands in this view

SpringSoft
Accelerating Engineers

Open *nSchema*

Flatten Window

- Focus on a specified primitive or net for debugging

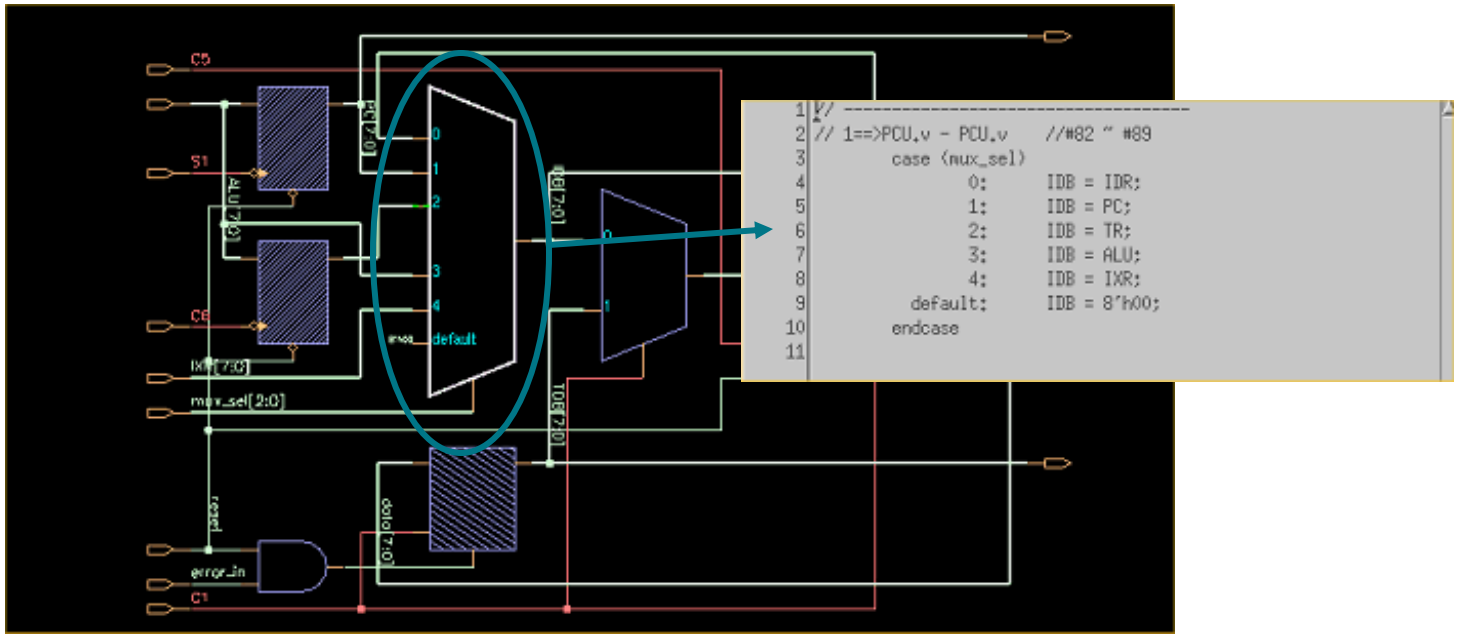


- DC on instance pin or D&D from other windows to add objects
- Use <Delete> or to remove selected object(s)

Correlate Other Views

View Detail Source Code for RTL Blocks

- DC on a bottom level RTL Block to show its corresponding source code



SpringSoft Accelerating Engineers

Manipulate Viewing

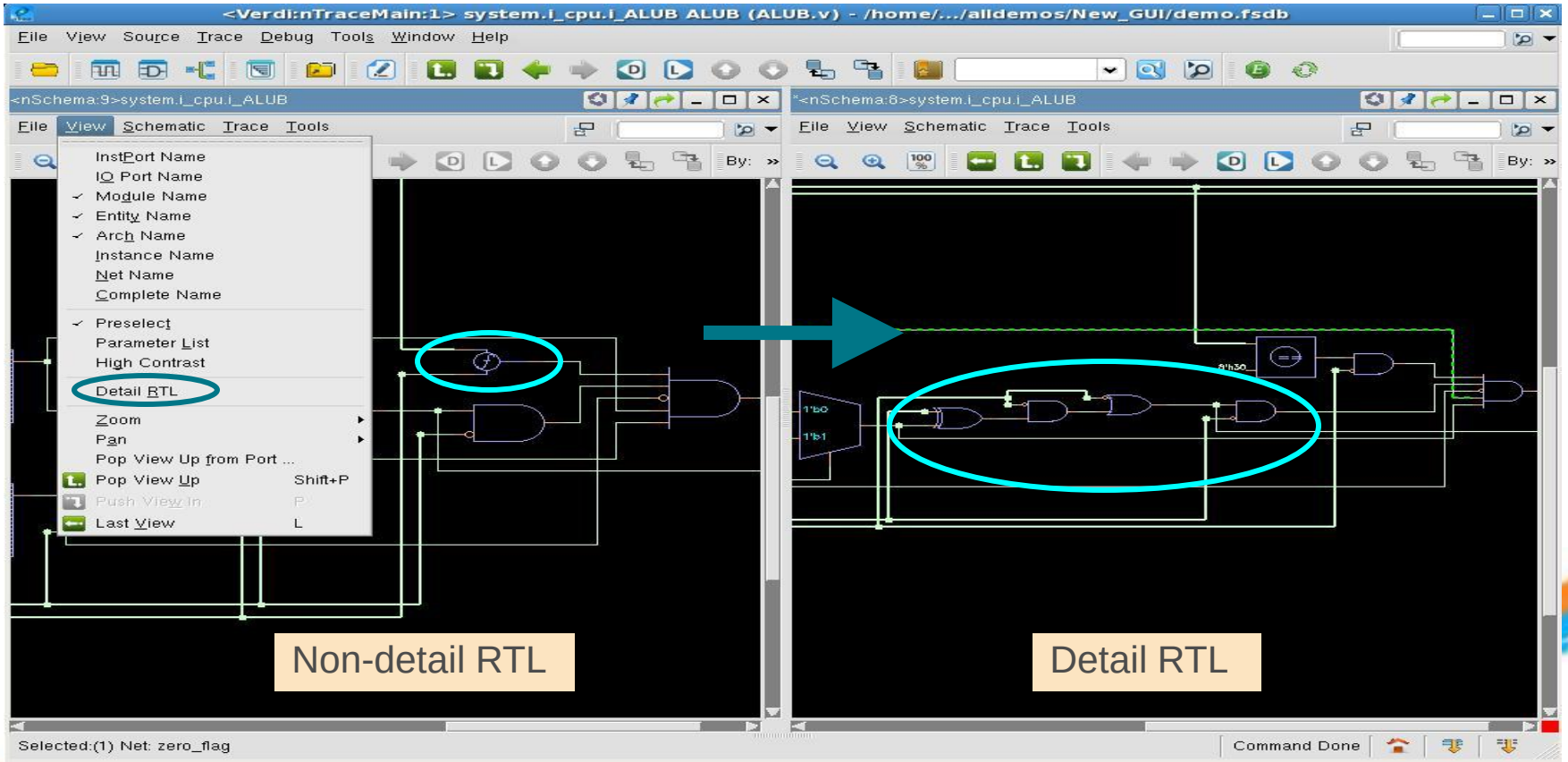
Miscellaneous Viewing Commands

- **View → Preselect**
  - Highlight signals/blocks when the cursor moves over them
- **View → InstPort Name**
  - Turn on/off port names of instances
- **View → IO Port Name, Instance Name, Net Name**
  - Turn on/off pin, instance, local net names
- **View → High Contrast**
  - Display all schematic in dark blue except the selected objects which are highlighted in white
- **Tools → Preferences → Schematics → Display Options → Viewing**
  - Change default viewing attributes when opening new *nSchema* windows

Manipulate Viewing

View Detail RTL on a Window Basis

- Invoke **View → Detail RTL** to show Boolean equivalent of function symbols
  - Effective in the current window only

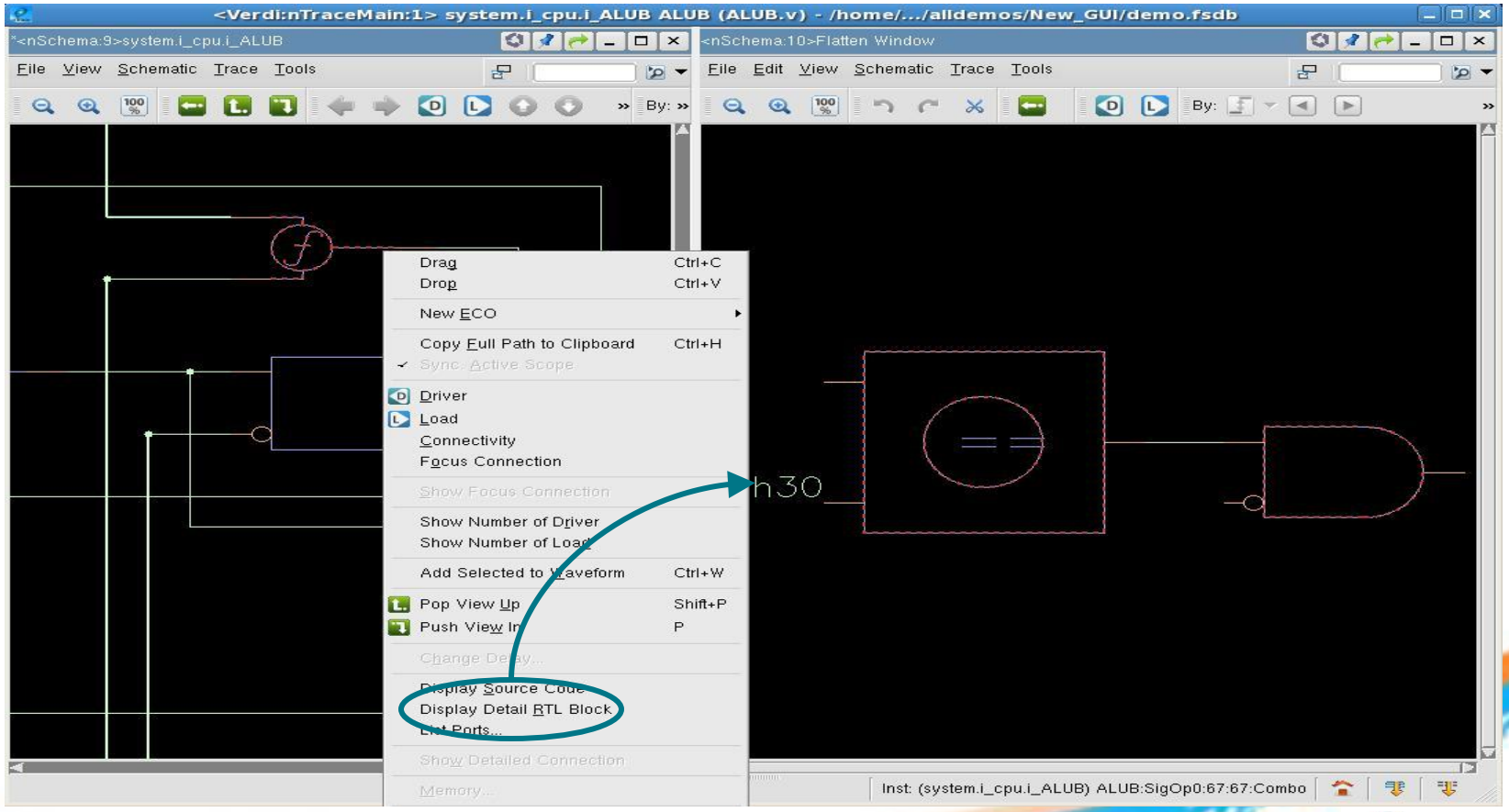


SpringSoft Accelerating Engineers

Manipulate Viewing

View Detail RTL on Instance Basis

- On RTL block, invoke RMB → **Display Detail RTL Block**
  - New window with details will open

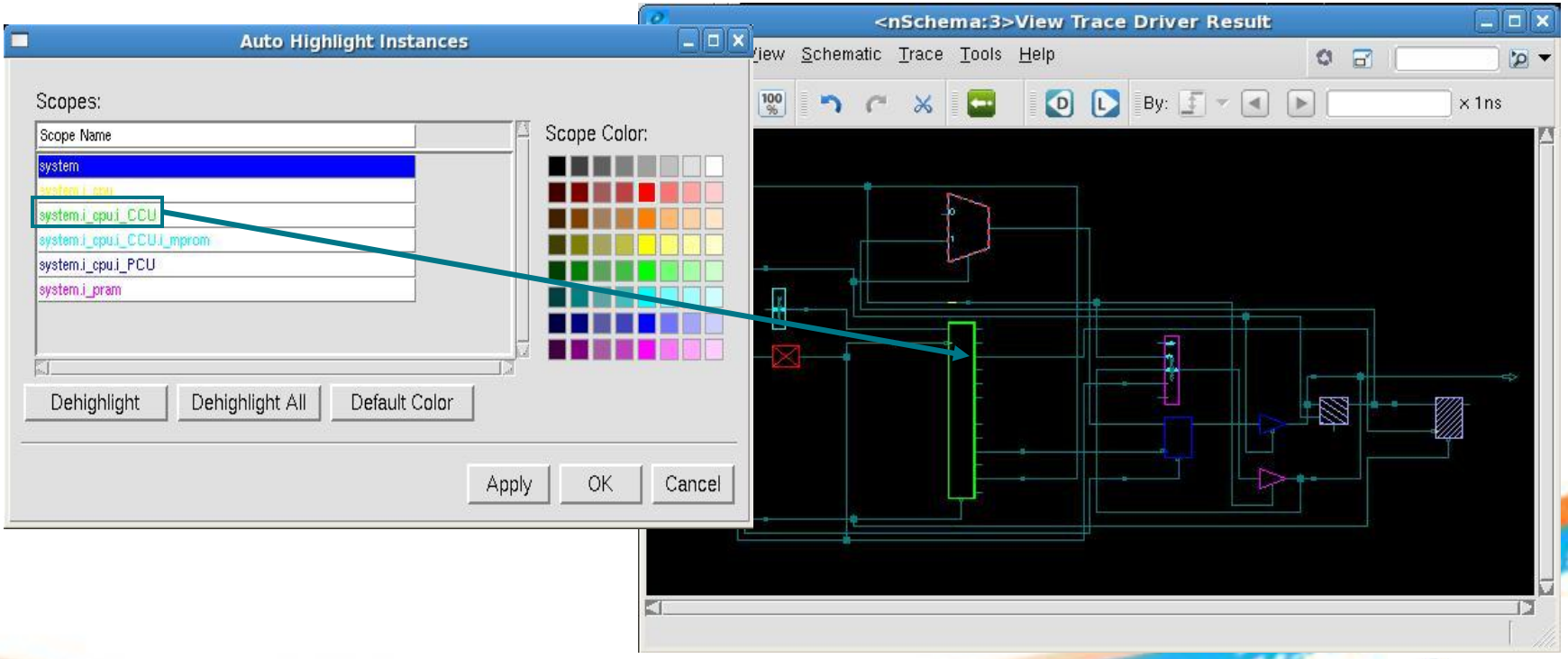


SpringSoft Accelerating Engineers

Manipulate Viewing

Colorize Different Hierarchies in Flatten Window

- Enable **Automatically Color Instances in Flatten Window** option on the **Schematics → Display Options → Schematic** folder of the *Preferences* form (invoked with **Tools → Preferences**)
- In flatten schematic window, invoke **Schematic → Color Instances by Scope** to change default scope colors. Use **View → High Contrast**

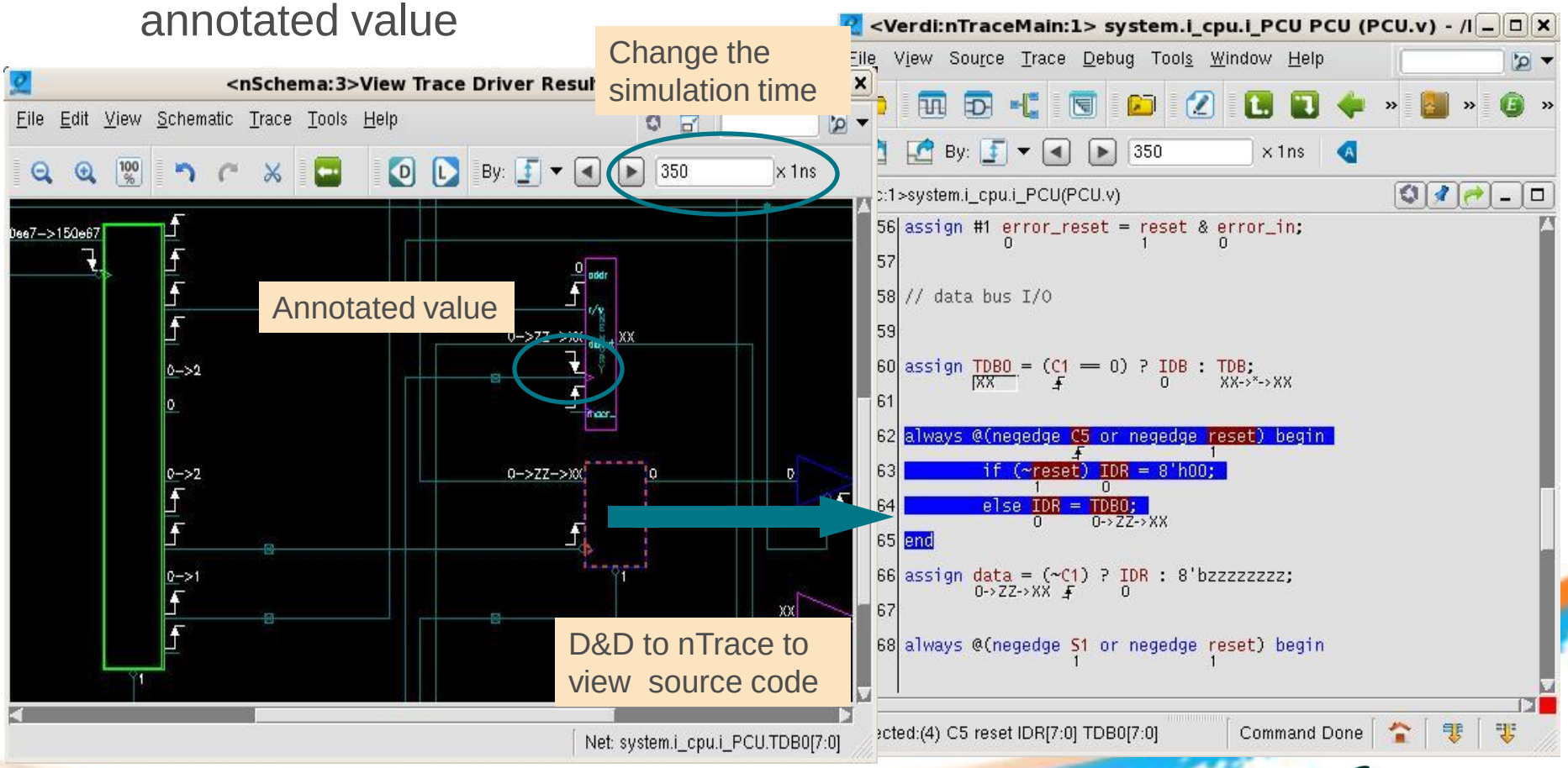


SpringSoft Accelerating Engineers

Manipulate Viewing

Annotate Signal Value in nSchema

- In *nTrace*, **File → Load Simulation Result** to load the FSDB
- In *nSchema*, **Schematic → Active Annotation** to display the annotated value

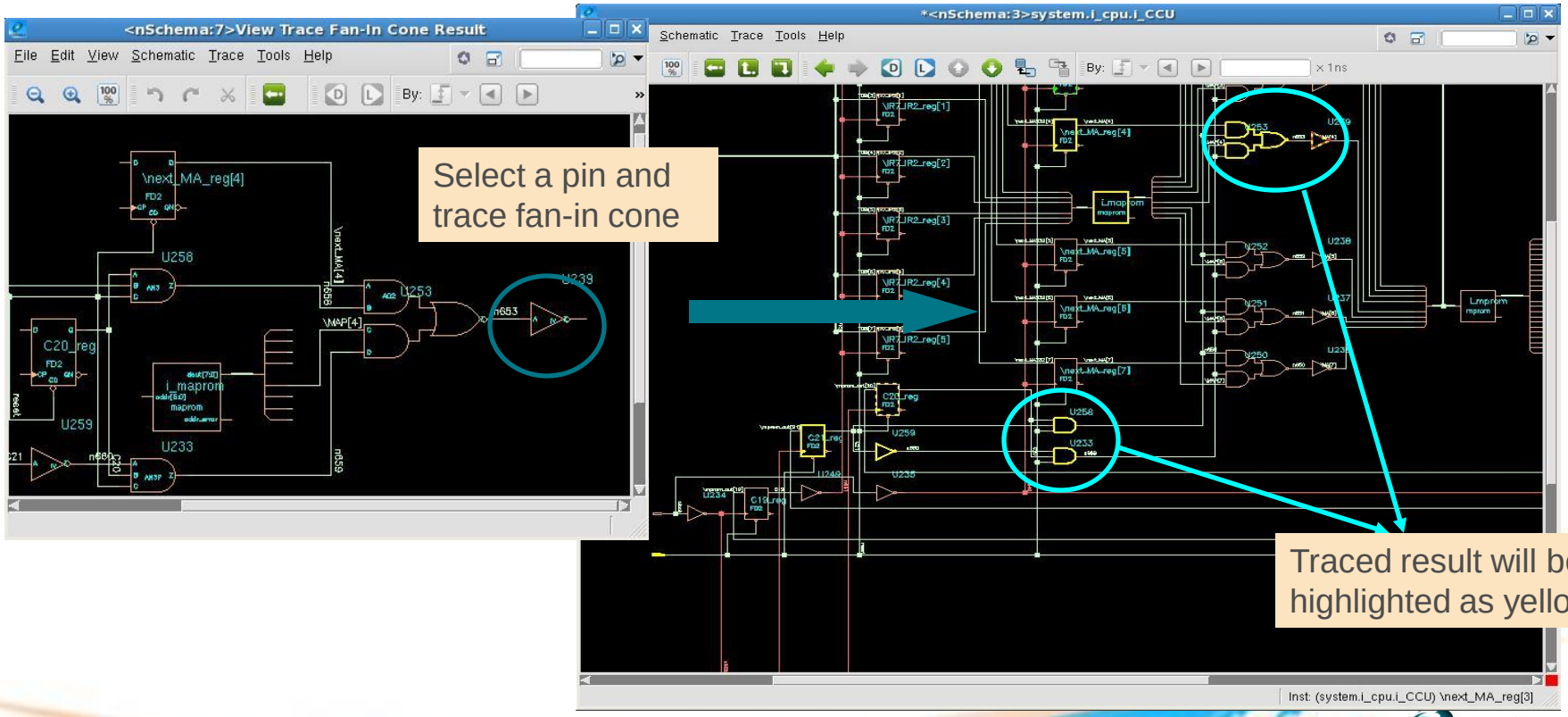


SpringSoft Accelerating Engineers

Trace Signals

Trace for Fan-in in Full Hierarchy Window

- Select a pin/signal/instance in *Full Hierarchy Window*, invoke **Trace → Fan-in Cone**
  - Shows all the logic affecting the selected net
  - Will trace drivers indepth until reaching the storage element

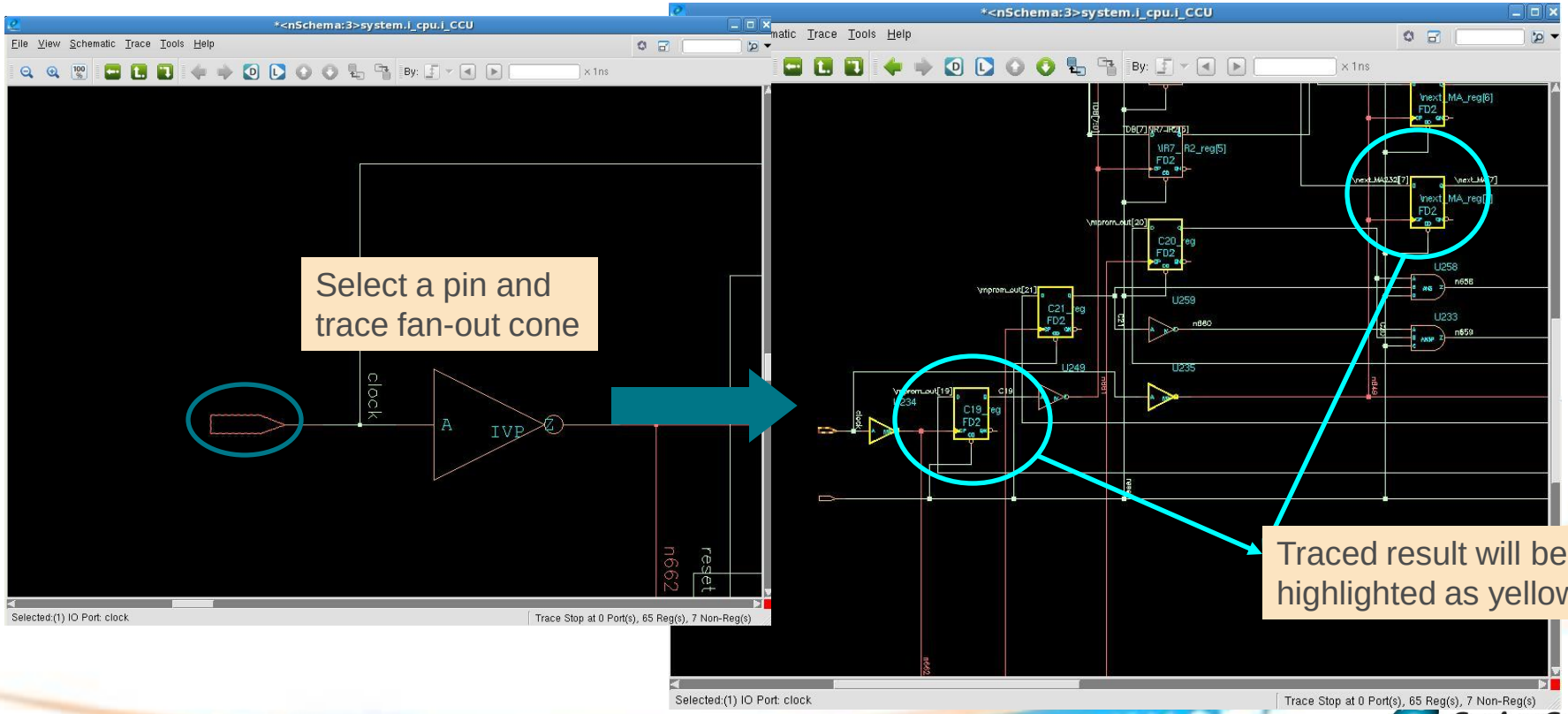


SpringSoft Accelerating Engineers

Trace Signals

Trace for Fan-out in Full Hierarchy Window

- Select a pin/signal/instance in *Full Hierarchy Window*, invoke **Trace → Fan-out Cone**
  - Shows all the loading logic
  - Useful to see global signal distribution (e.g. clock, reset, etc.)

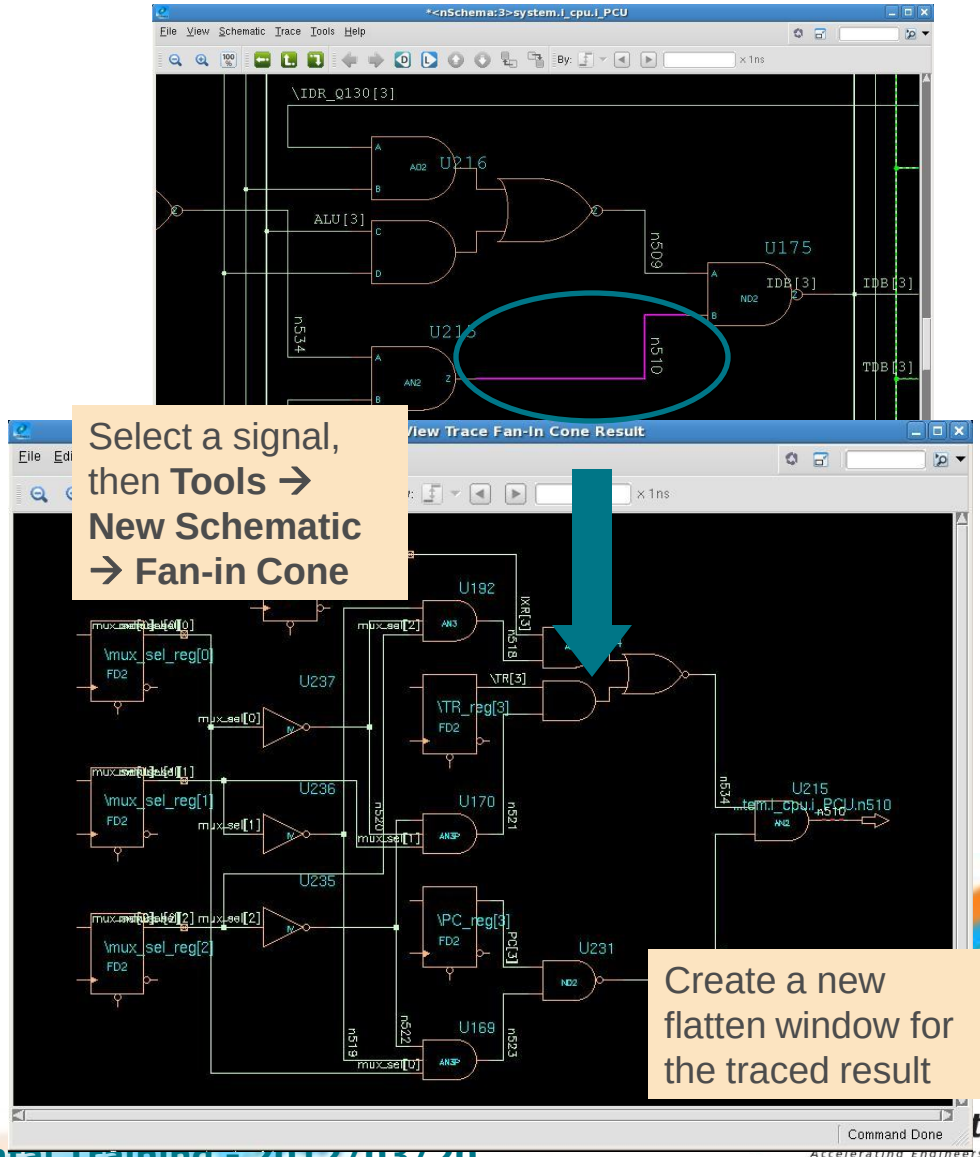


SpringSoft Accelerating Engineers

Trace Signals

Show Traced Result in a New Flatten Window

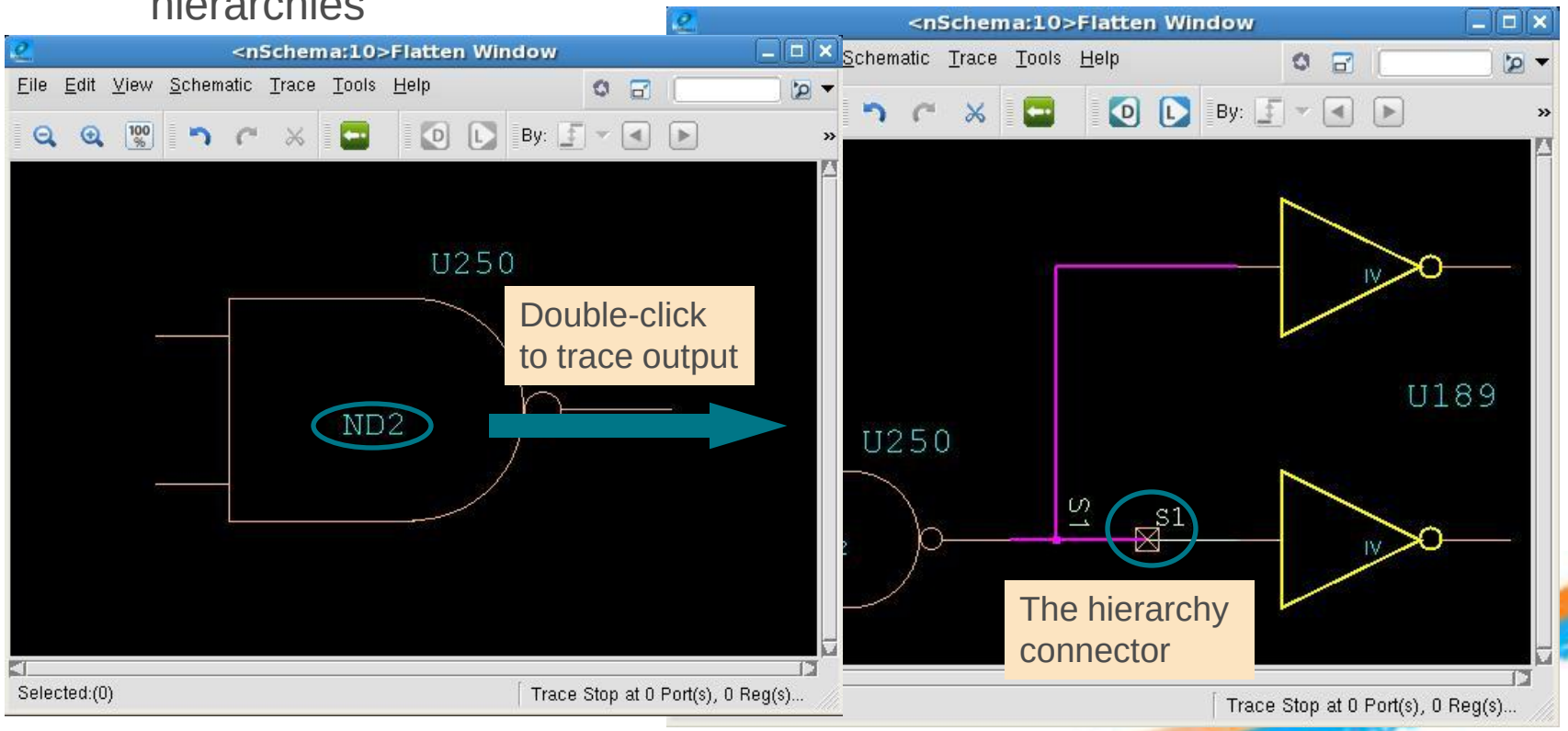
- Invoke **Tools → New Schematic → Fan-in Cone / Fan-out Cone**
  - Generate new schematic window to show all the fan-in or fan-out logic
  - Trace net to register / tri-state output or design boundary
- Trace fan-in/out more then one level
  - Specify the level number in the **Fan-In/Out Level** field on the **Trace** page under the **Schematics** folder of the *Preferences* form (invoked with **Tools → Preferences**)



Trace Signals

Expand the schematic in Flatten Window

- In **Tools → New Schematic → *Flatten Window***, double-click on the instance pin to expand one level
  - The hierarchical connector will be added if tracing through hierarchies

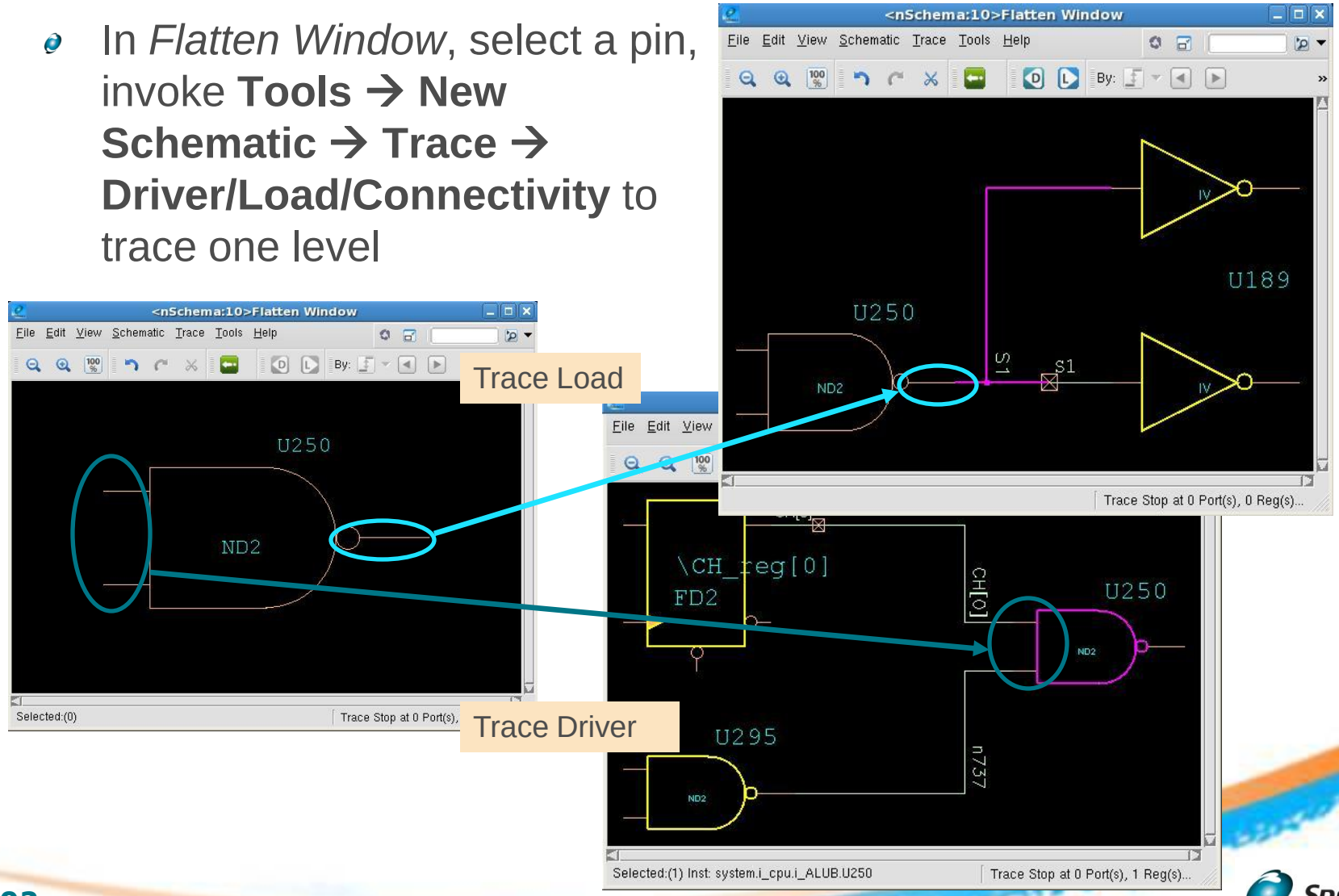


SpringSoft Accelerating Engineers

Trace Signals

Trace Load/Driver/Connectivity in Flatten Window

- In *Flatten Window*, select a pin, invoke **Tools → New Schematic → Trace → Driver/Load/Connectivity** to trace one level

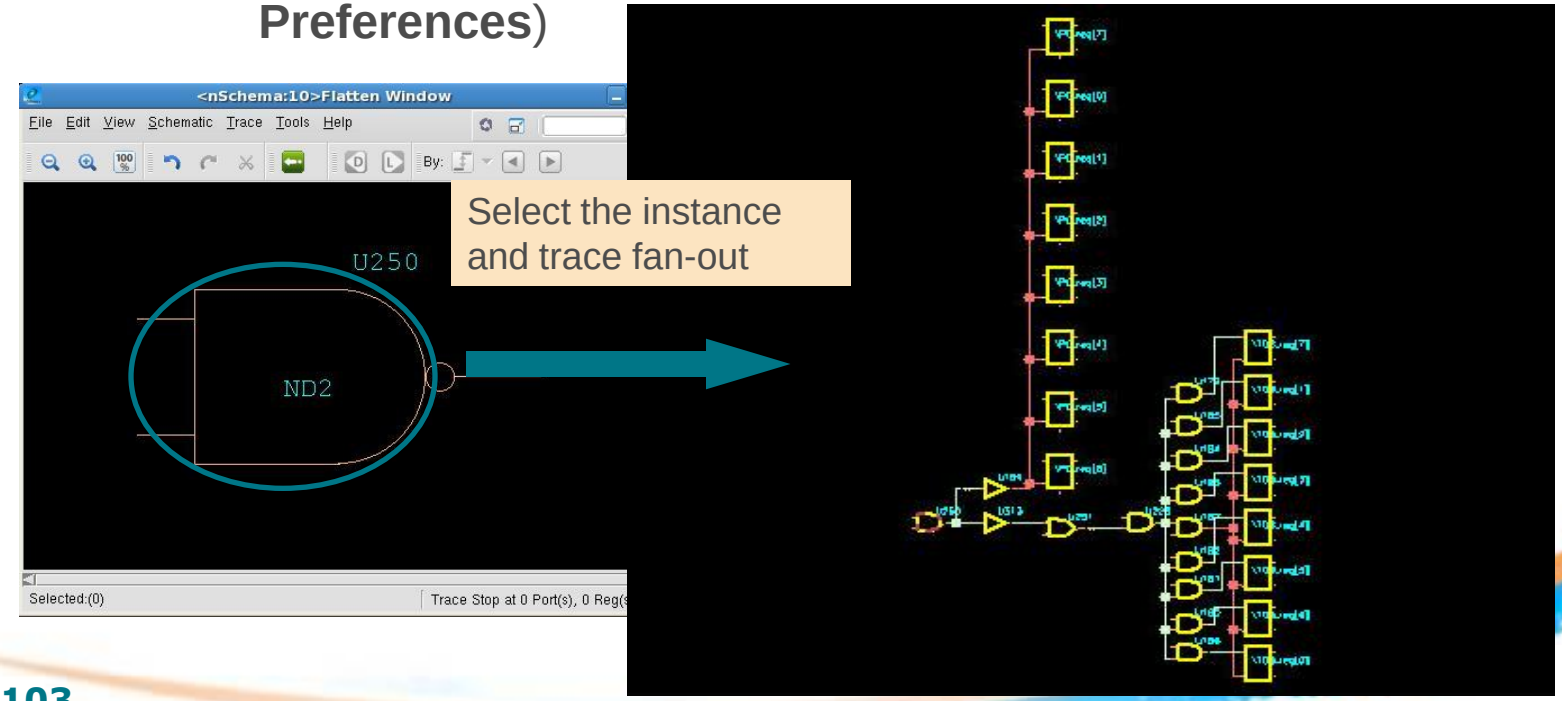


SpringSoft Accelerating Engineers

Trace Signals

Trace Fan-out in Flatten Window

- In *Flatten Window*, select an instance or pin, invoke **Trace → Fan-out Cone**
  - Trace to storage elements
  - Useful to see global signal distribution (e.g. clock, reset, etc.)
  - Specify the level number in the **Fan-In/Out Level** field on **Schematics → Trace** folder of the *Preferences* form (invoked with **Tools → Preferences**)

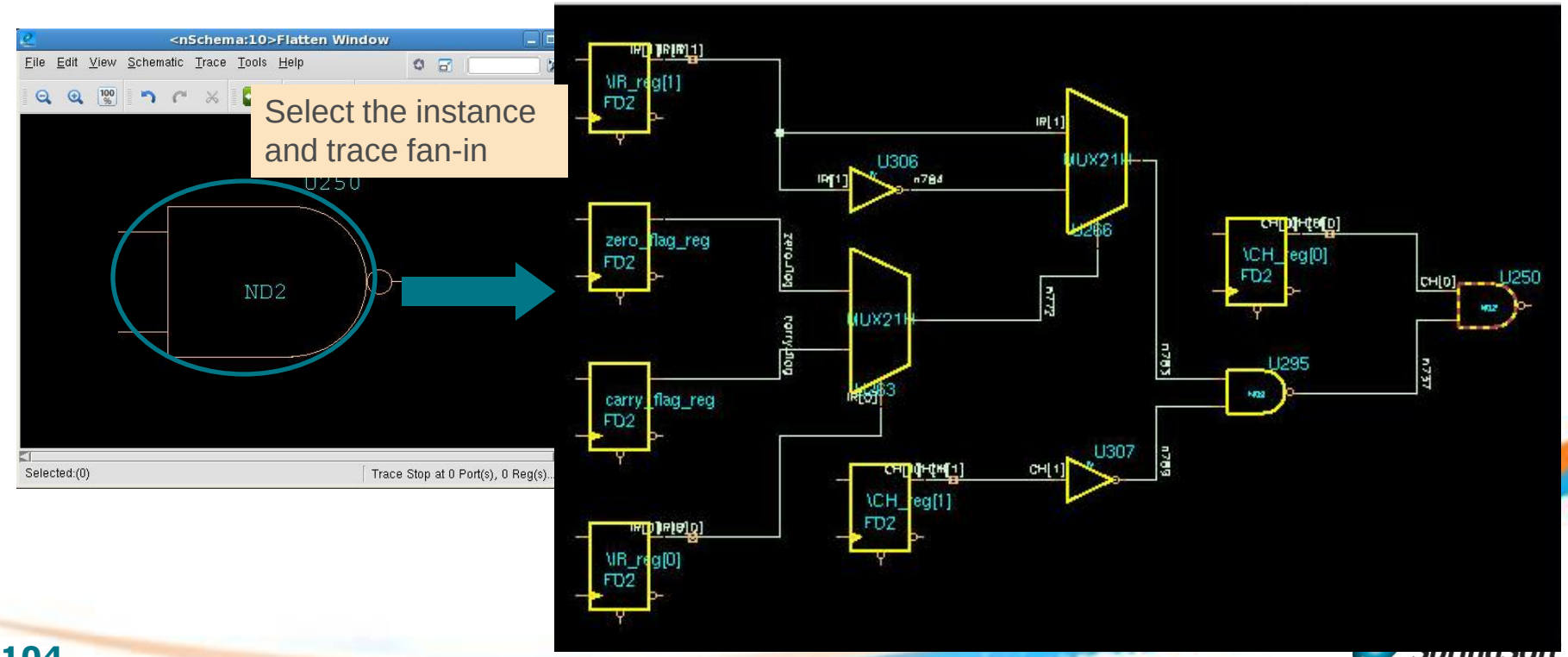


SpringSoft Accelerating Engineers

Trace Signals

Trace Fan-in in Flatten Window

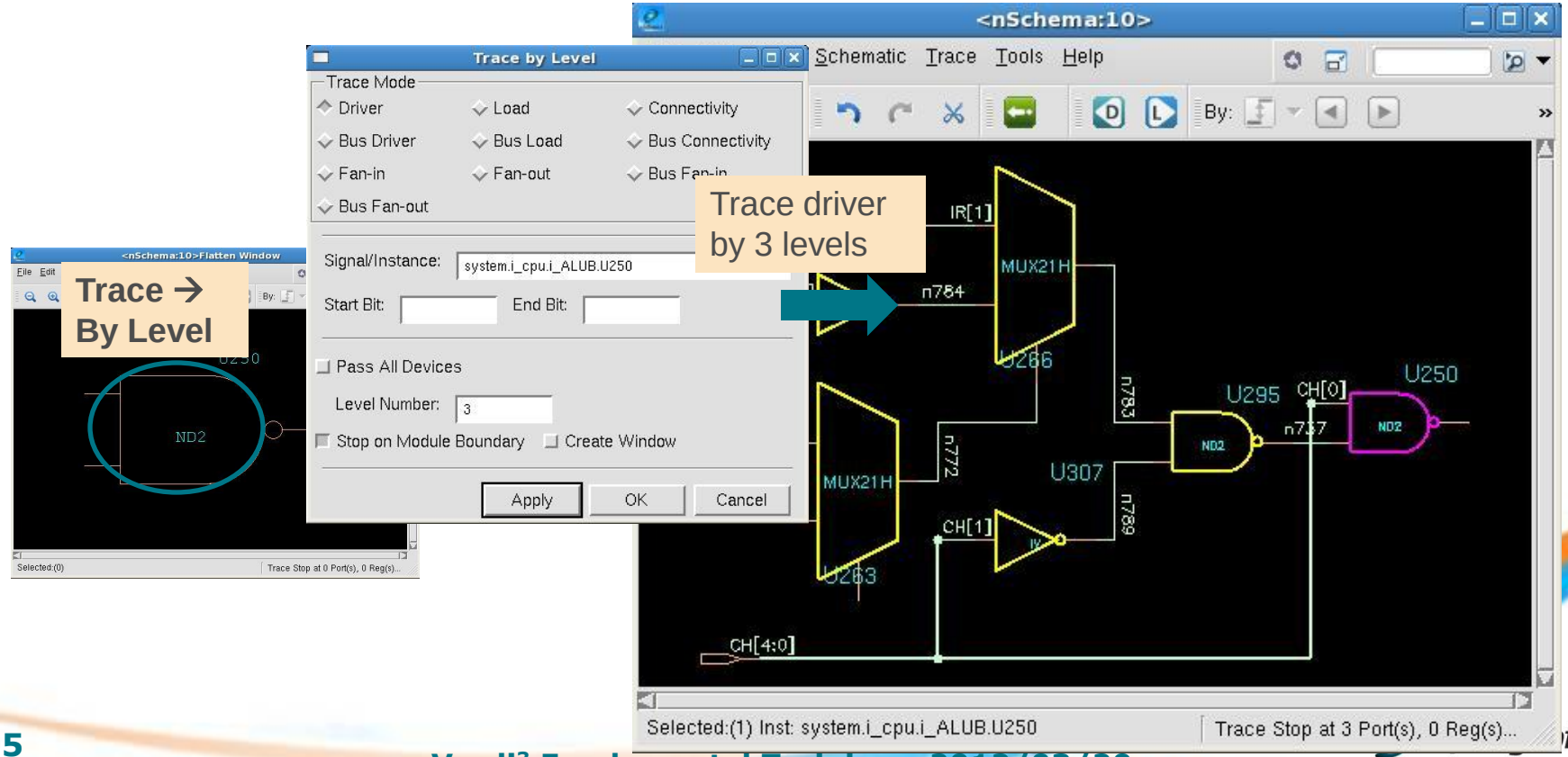
- In *Flatten Window*, select an instance or pin, invoke **Trace → Fan-in Cone**
  - Trace to storage elements
  - Useful to trace drivers in depth to find the cause of error
  - Specify the trace level in the *Preferences* form



Trace Signals

Trace by Level in Flatten Window

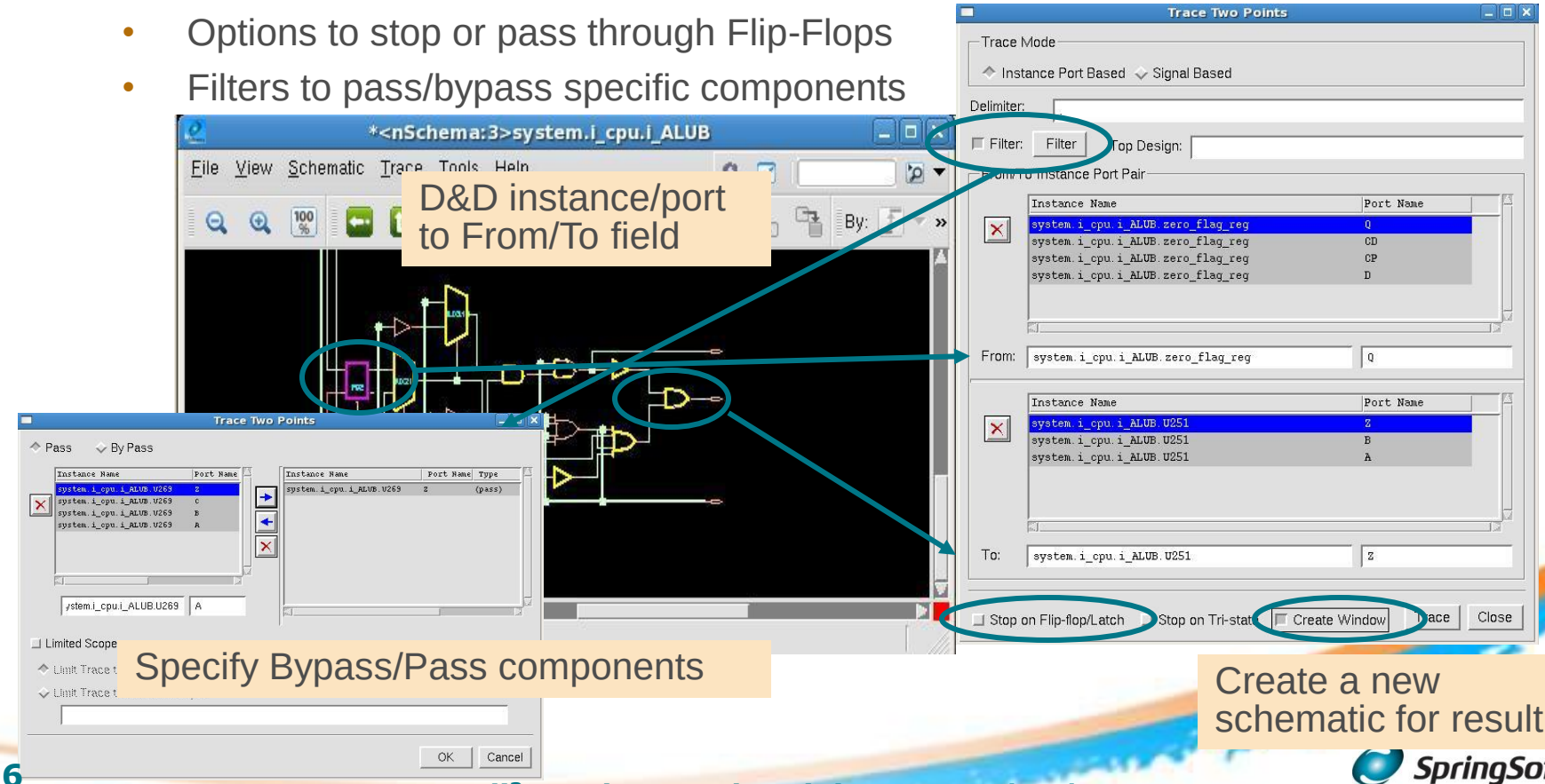
- In *Flatten Window*, select an instance or pin, invoke **Trace → By Level**
  - Specify the Trace Mode (e.g. Driver, Load, Fan-in…etc)
  - Specify Level and other constraints



Trace Signals

Trace All Paths Between Two Points (1/2)

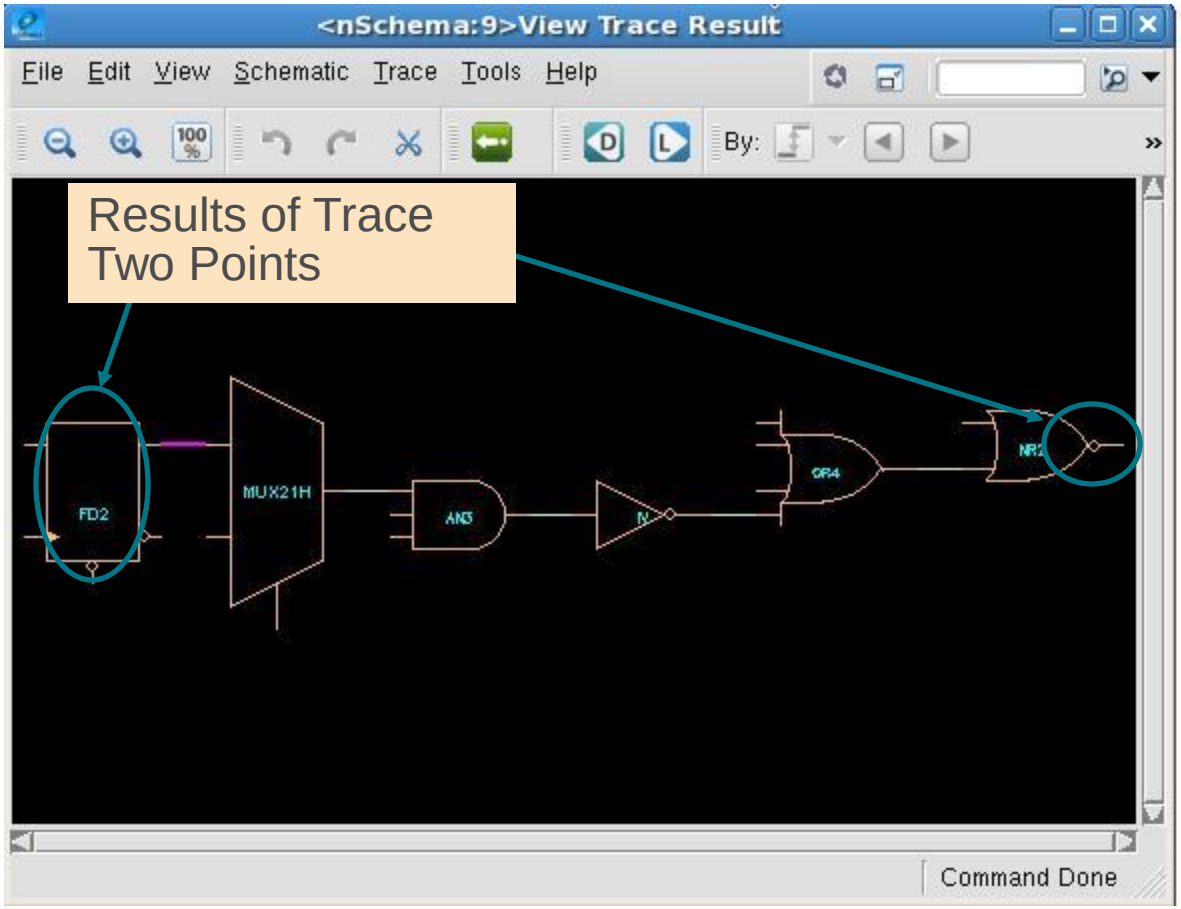
- Invoke **Trace → Two Points** and select **Instance Port Based** option to trace between ports
  - Specify start/end points by entering the port names or D&D from *nSchema* or *nTrace*
  - Options to stop or pass through Flip-Flops
  - Filters to pass/bypass specific components



SpringSoft Accelerating Engineers

Trace Signals

Trace All Paths Between Two Points (2/2)



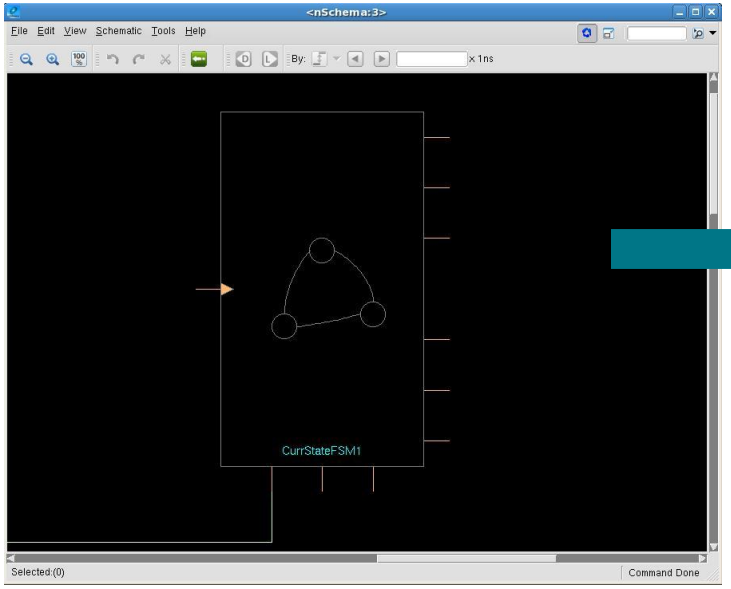
SpringSoft
Accelerating Engineers

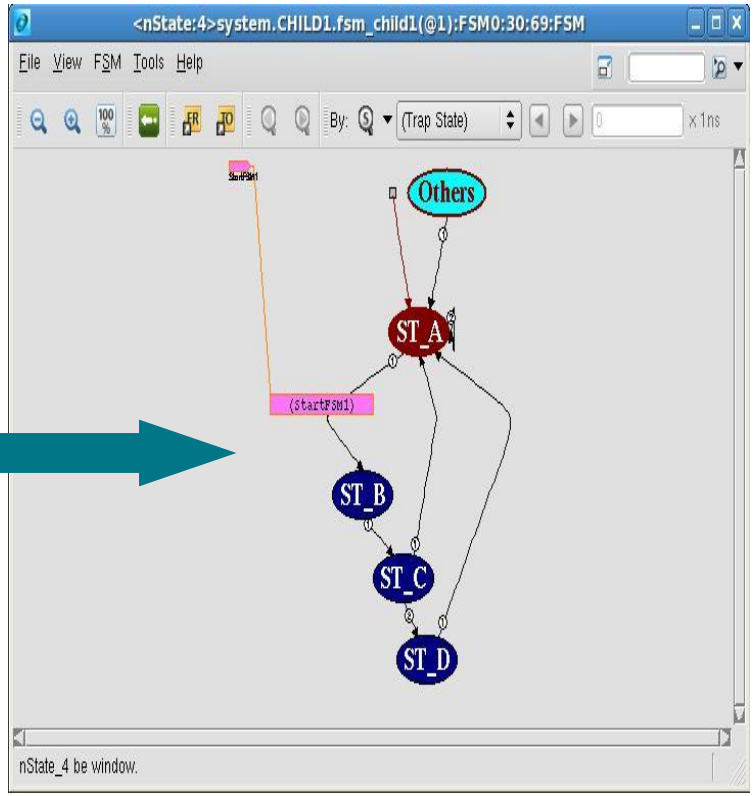
Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Open *nState* Frame

- DC on the state symbol to open *nState*
  - The *nState* frame appears as a new tab in the same location as the *nSchema* frame





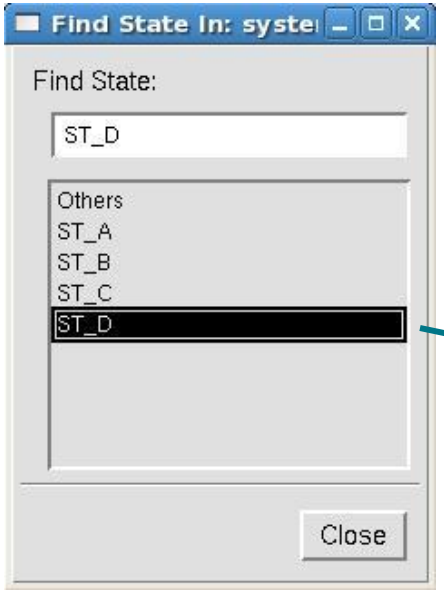
- Verdi automatically identifies FSM code for 1-process, 2-process, or 3-process with conventional encoding of the state variables
- **NOTE:** Remember to enable **FSM Recognition** option on the **RTL** page under the **Schematics** folder of the *Preferences* form (invoked with **Tools → Preferences**) otherwise not all FSM will be extracted

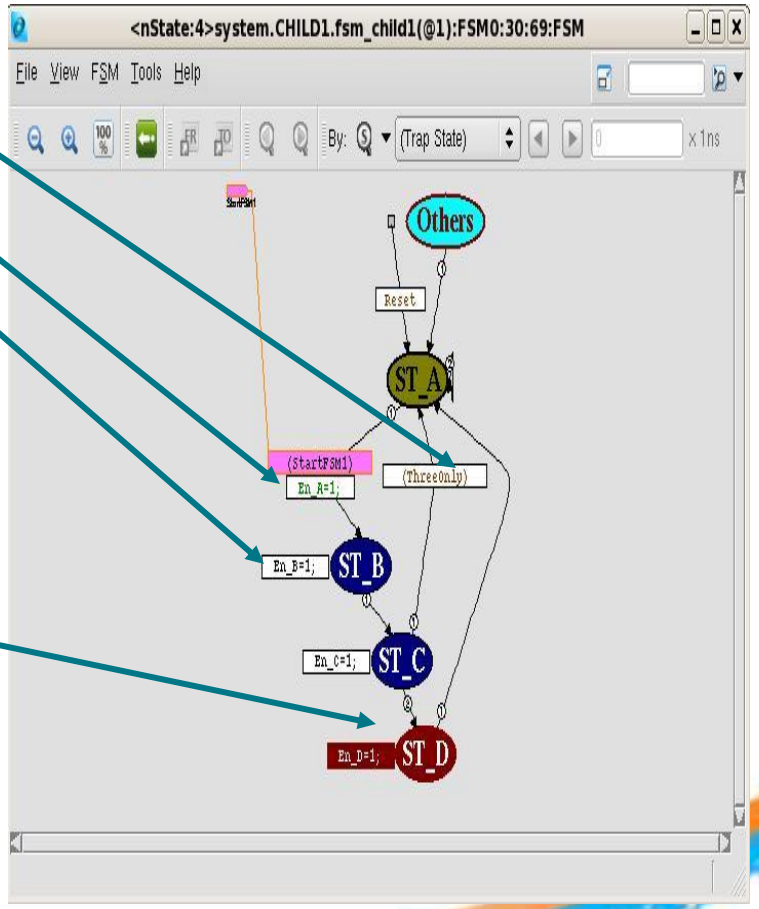
SpringSoft
Accelerating Engineers

Manipulate FSM View

View Actions & Find States

- **View → State Action**
- **View → Transition Condition**
- **View → Transition Action**
- **FSM → Find State**



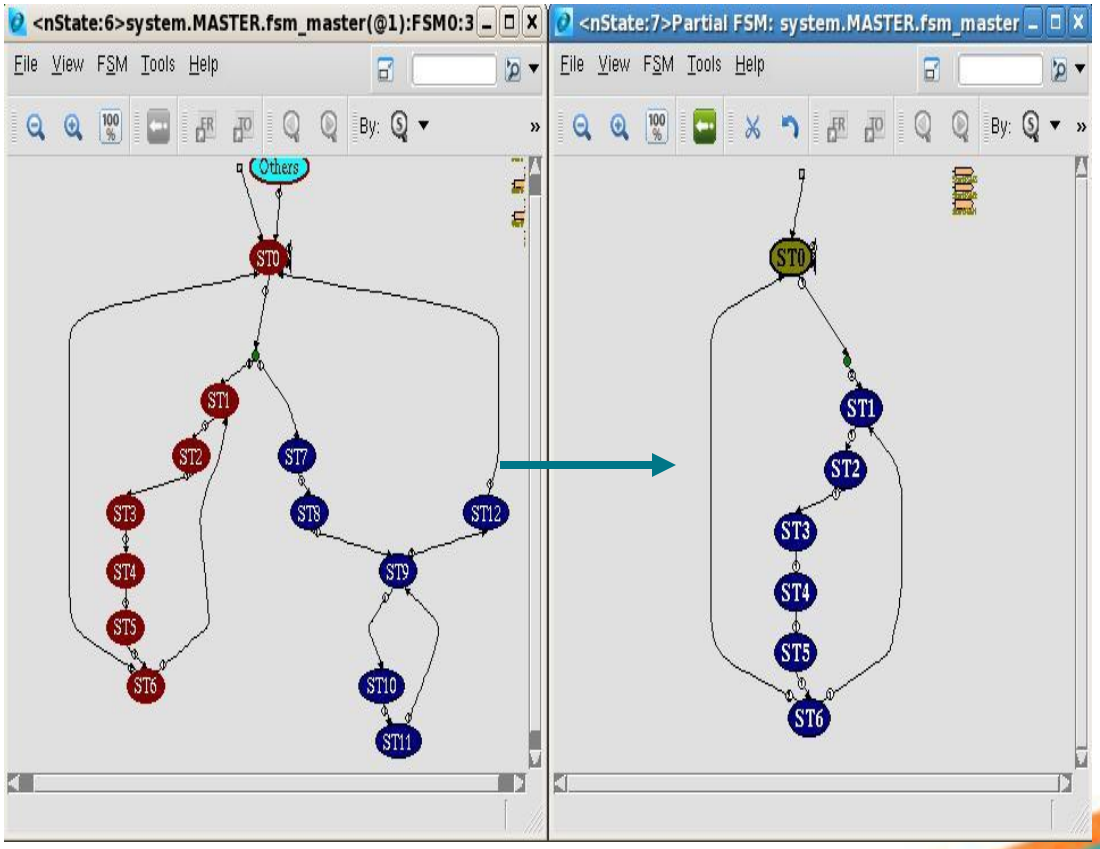


SpringSoft Accelerating Engineers

Manipulate FSM View

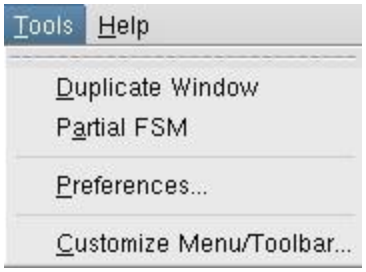
Create Partial FSM

- Select states in *nState* and invoke window from **Tools → Partial FSM**
- Expand adjacent states
  - DC on states
  - Select a state and invoke **RMB → Expand**
- D&D to the partial FSM window to add states



Complete FSM

Partial FSM



SpringSoft Accelerating Engineers

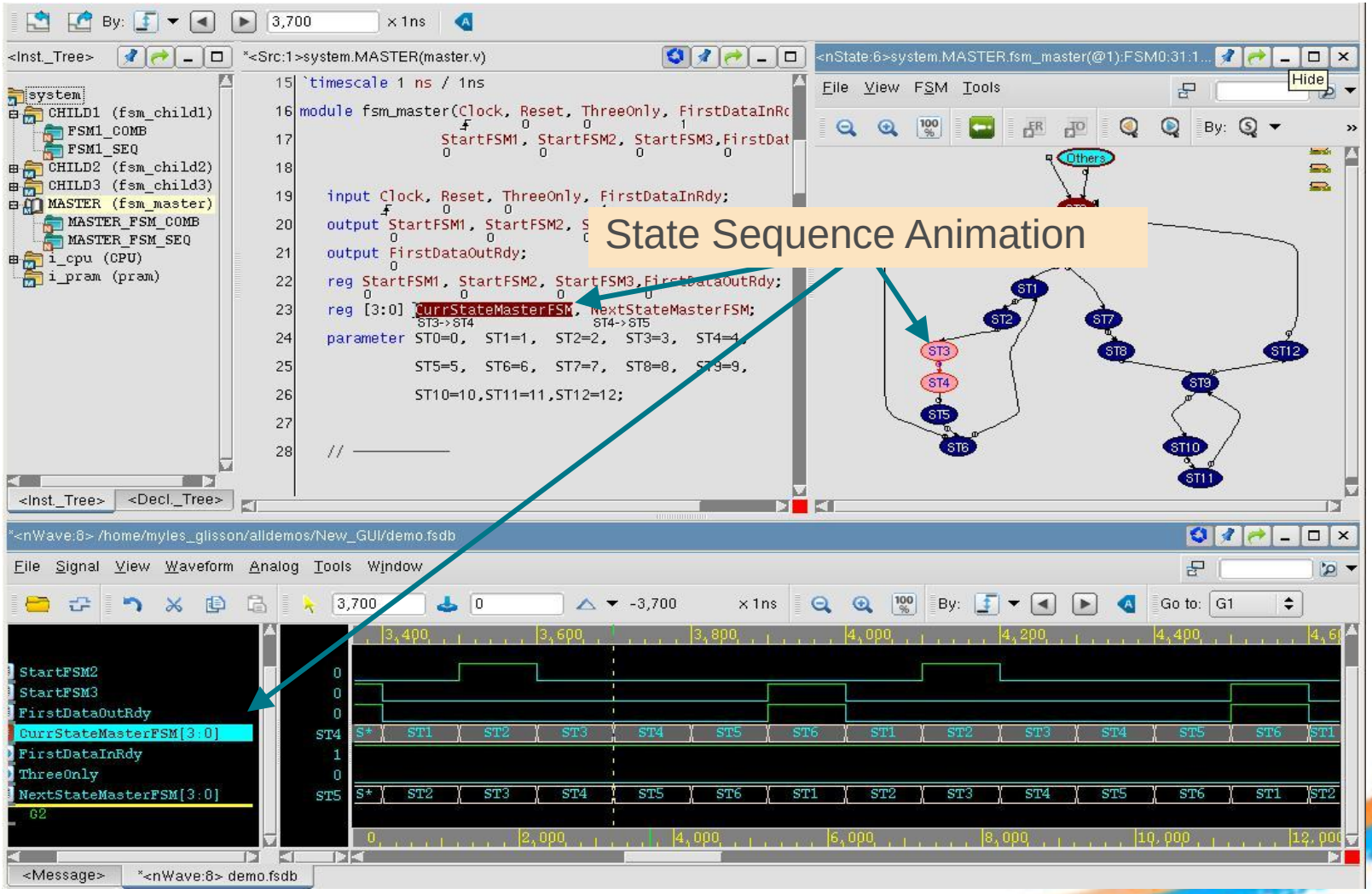
Manipulate FSM View

State Animation (1/2)

- Load simulation results
- Enable **FSM → State Animation**
- Use **Previous State / Next State** toolbar icons to step through the states
- Use **FSM → Edit Search Sequence** to create a sequence of states to search on

Manipulate FSM View

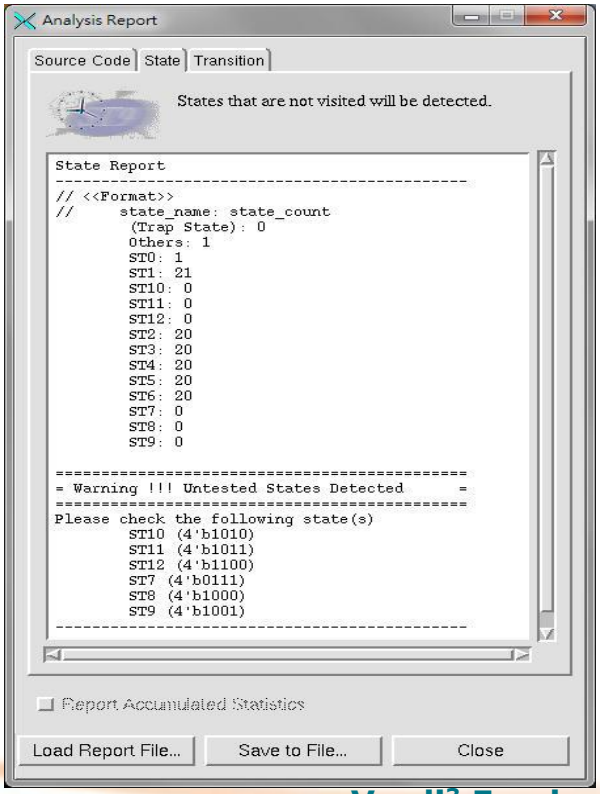
State Animation (2/2)

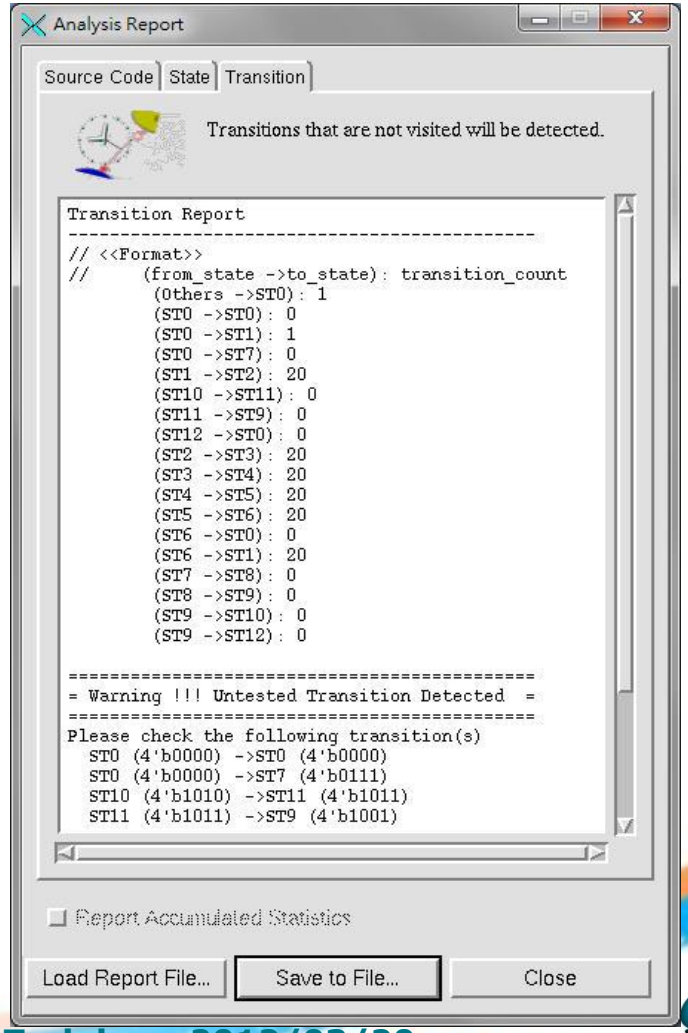


SpringSoft Accelerating Engineers

Analyze State Machines

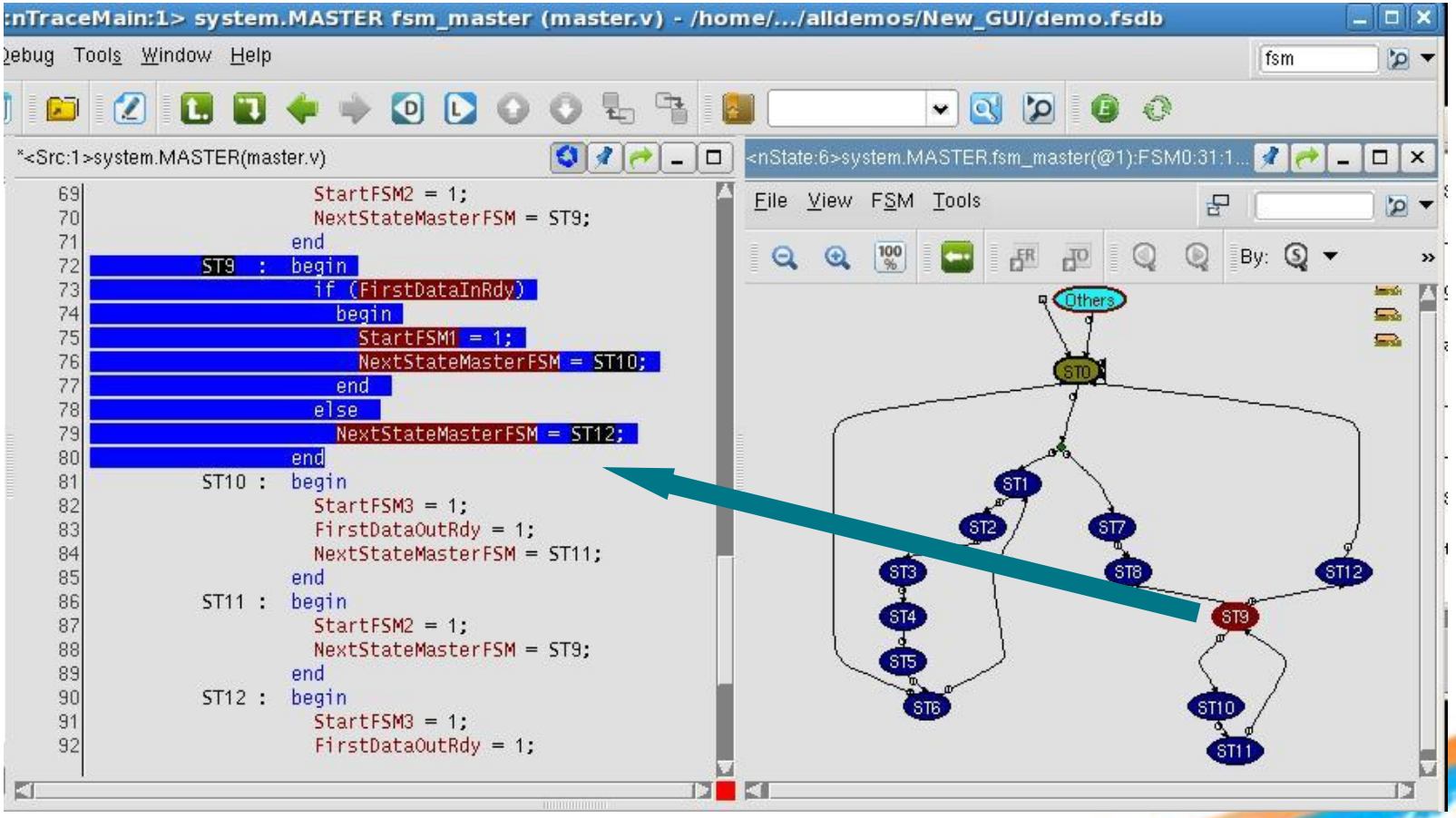
- Use **FSM → Analysis Report** to display analysis report
  - Details of FSM interpretation
  - State coverage report
  - Transition coverage report
- **Save to File** to save the report





Correlate Other Views

- D&D the state or junction into *nTrace*



SpringSoft Accelerating Engineers

Topics

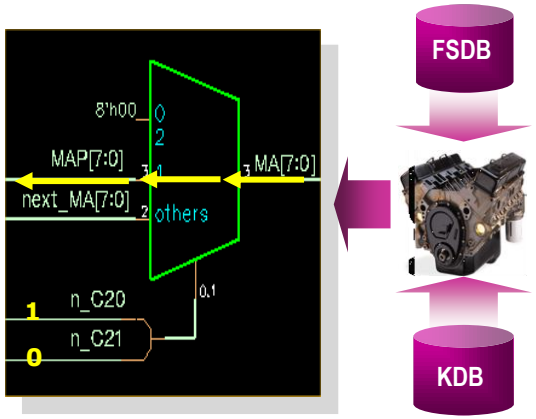
- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

What is a Temporal Flow View?

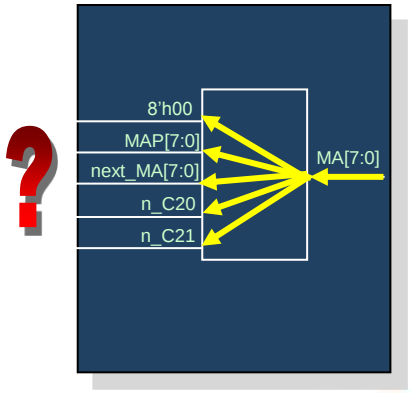
Behavior Analysis Engine

- From the KDB and FSDB, the Verdi platform builds an internal model of actual design behavior using synthesis and formal technology
  - Unroll function over time
  - Differentiate data from control signals
  - Determine clocking
  - Prune inactive elements

Automated Debug ***vs.*** ***Manual Debug***



Verdi



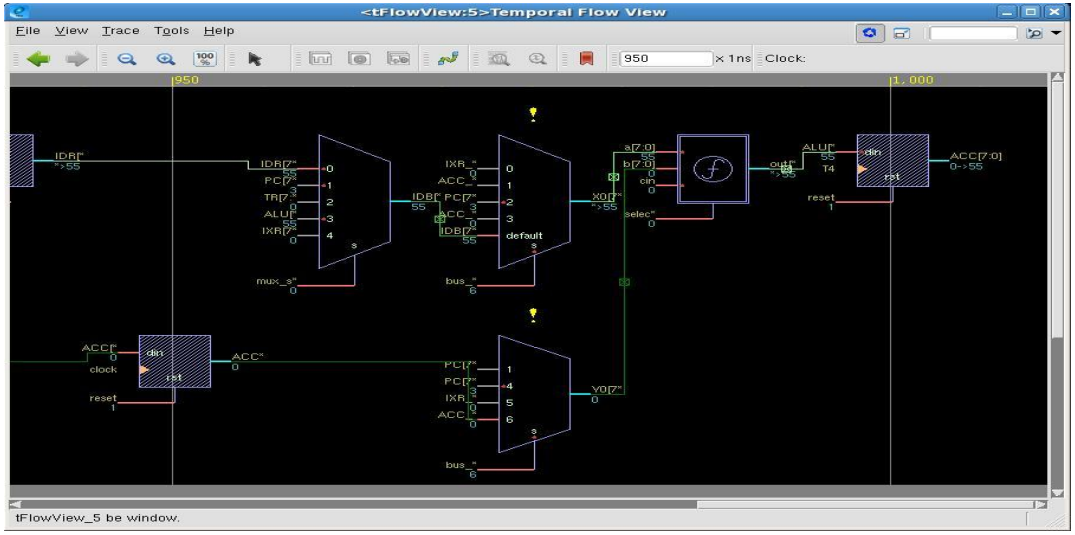
Traditional

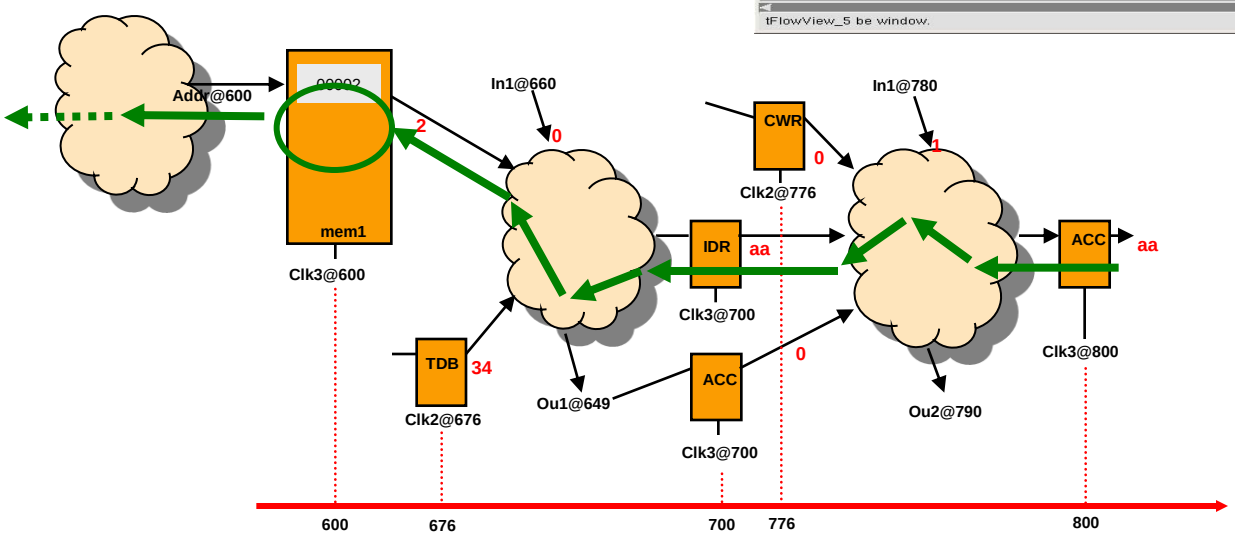
SpringSoft Accelerating Engineers

What is a Temporal Flow View?

Automatic Cause and Effect Tracing

- Complete, easy to use environment
- Easy visualization of behavior with time and structure



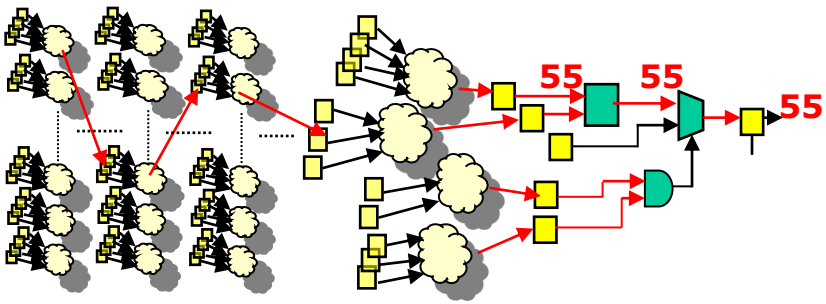


SpringSoft Accelerating Engineers

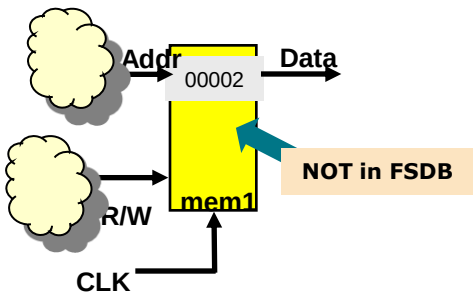
What is a Temporal Flow View?

Key Applications of Temporal Flow View Debugging

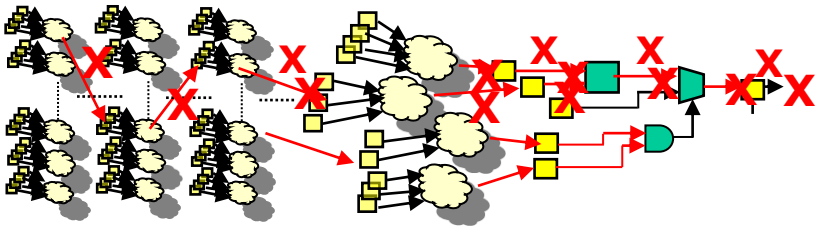
- Understand design behavior and locate the **root cause of a wrong value**



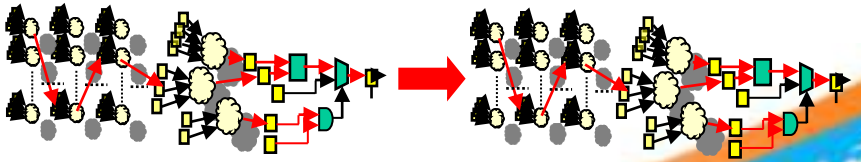
- **Trace memory** content NOT in FSDB and locate memory write



- Locate the root cause of **unknown (X) values**



- Locate the **cause of different results** between two simulation runs

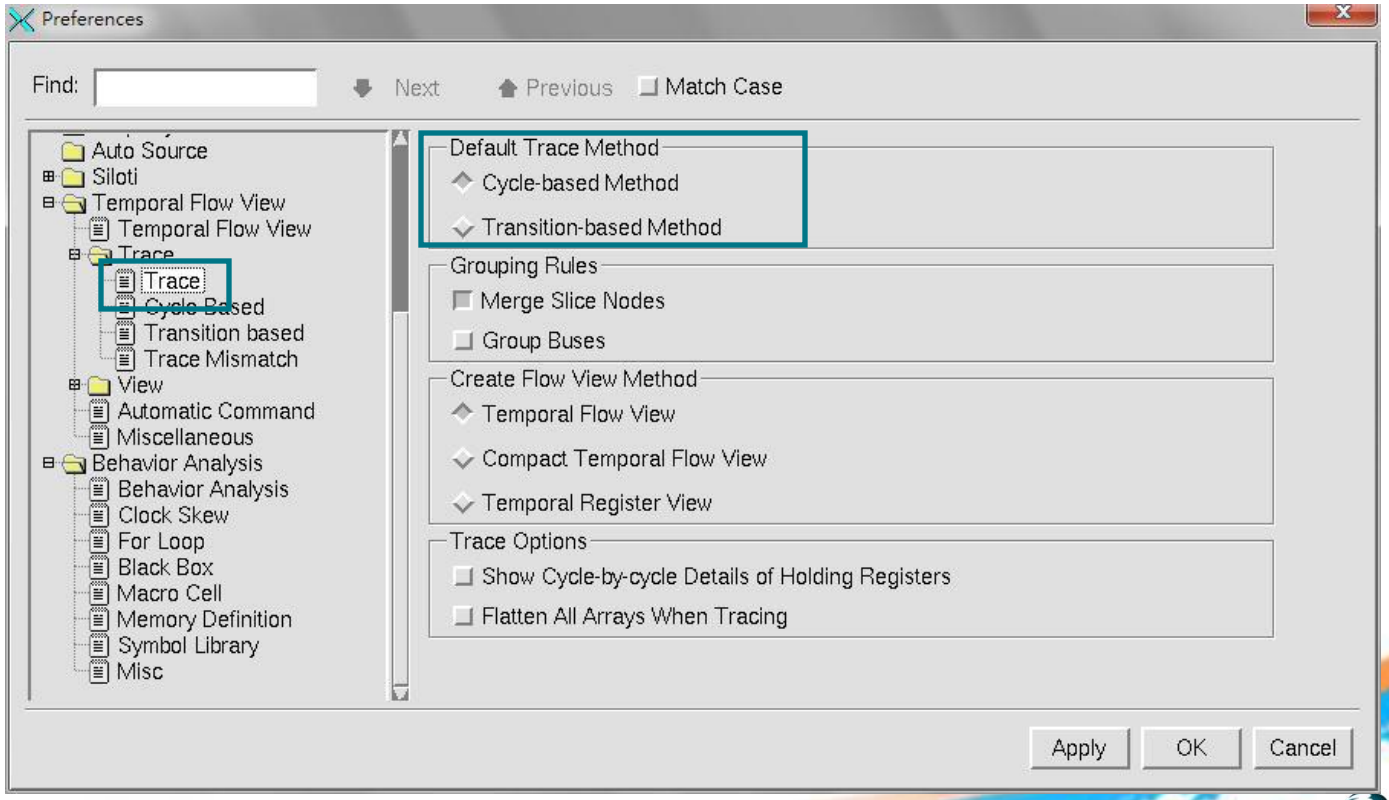


SpringSoft Accelerating Engineers

Open a Temporal Flow View Frame

Setup the Temporal Flow View

- Specify tracing by cycle or transition in **Tools → Preferences**
  - Select in **Default Trace Method** section of **Temporal Flow View → Trace** page
  - Default is **Cycle Based Method**

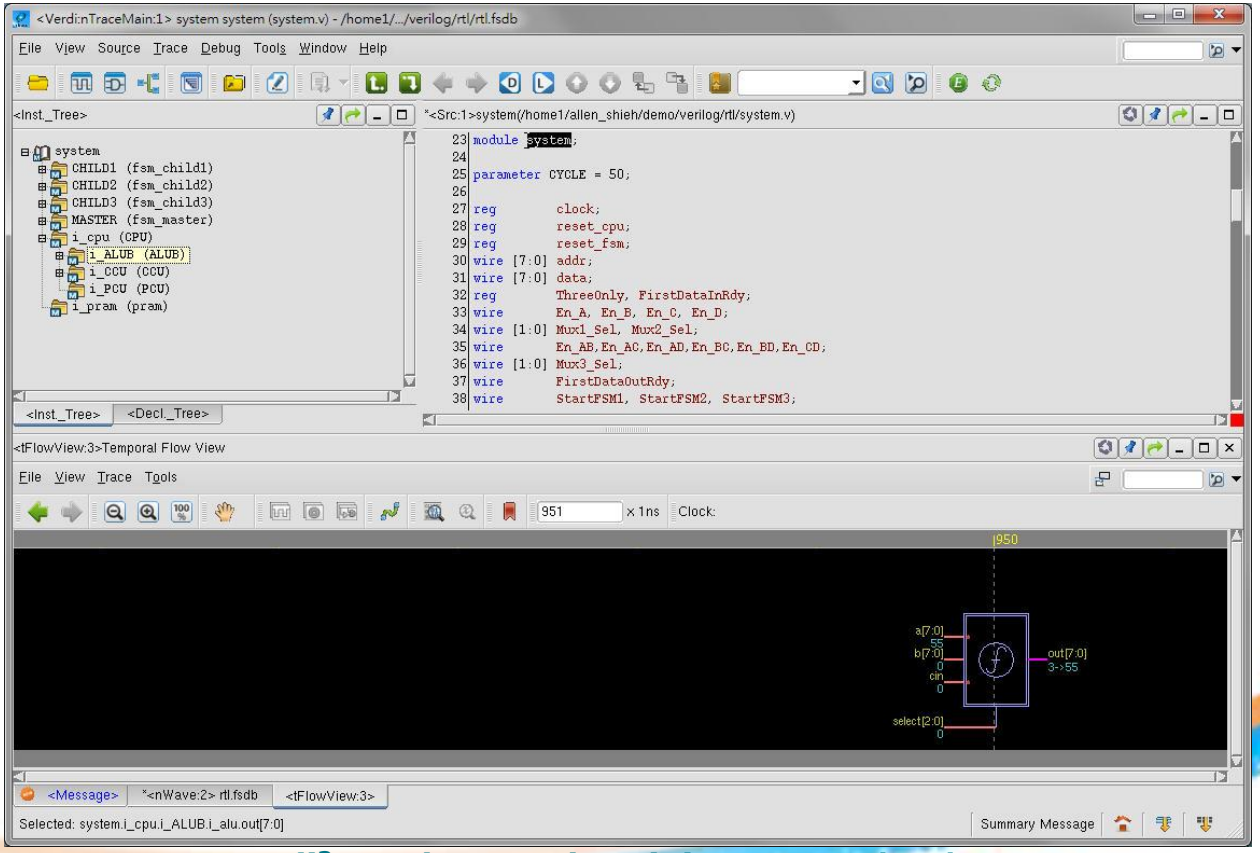


SpringSoft
Accelerating Engineers

Open a Temporal Flow View Frame

Create the Temporal Flow View

- In *nWave* or *nTrace*, select a signal transition and invoke the RMB → **Create Temporal Flow View** command
  - The *Temporal Flow View* frame appears in the bottom of the main window (same location as the *nWave* frame)

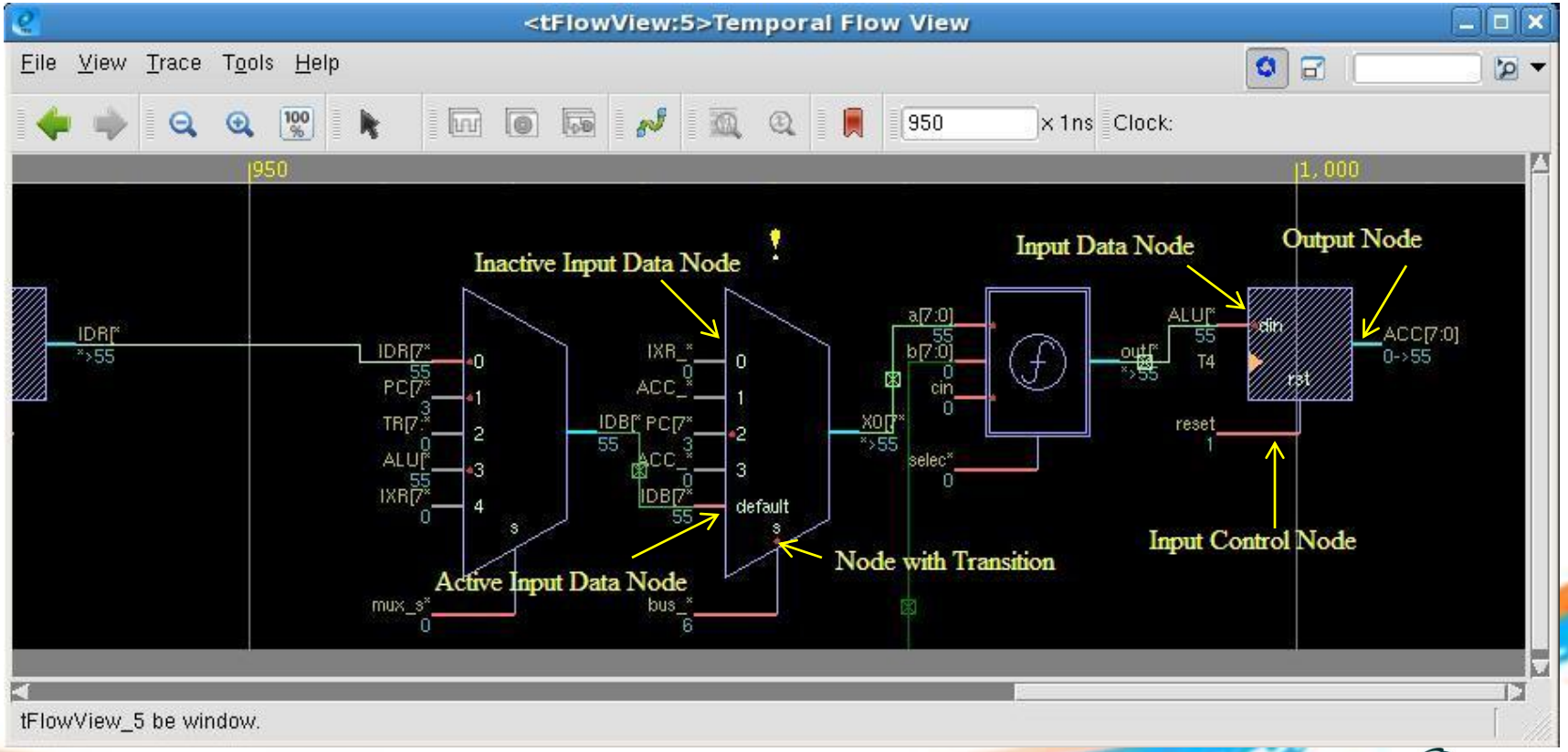


SpringSoft
Accelerating Engineers

Open a Temporal Flow View Frame

Cycle Based

- Structure and time in single view
- Dynamic analysis finds the active path when tracing
- Each structure corresponds to one statement

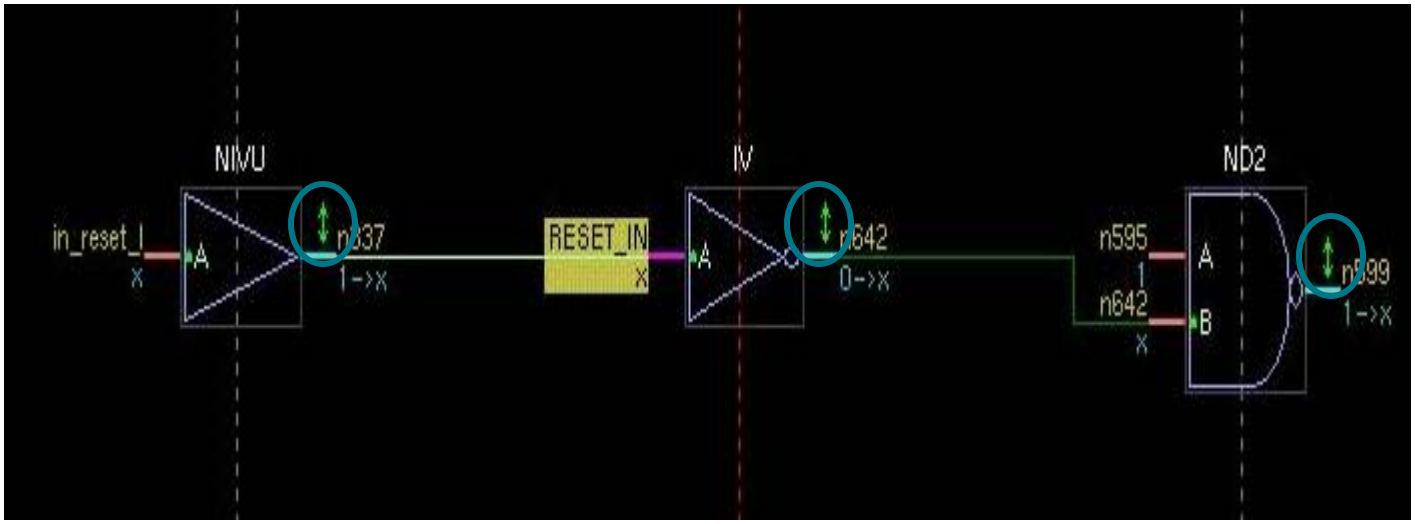


SpringSoft Accelerating Engineers

Open a Temporal Flow View Frame

Transition Based

- distinguishes transition based tracing from cycle-based tracing in the Temporal Flow View

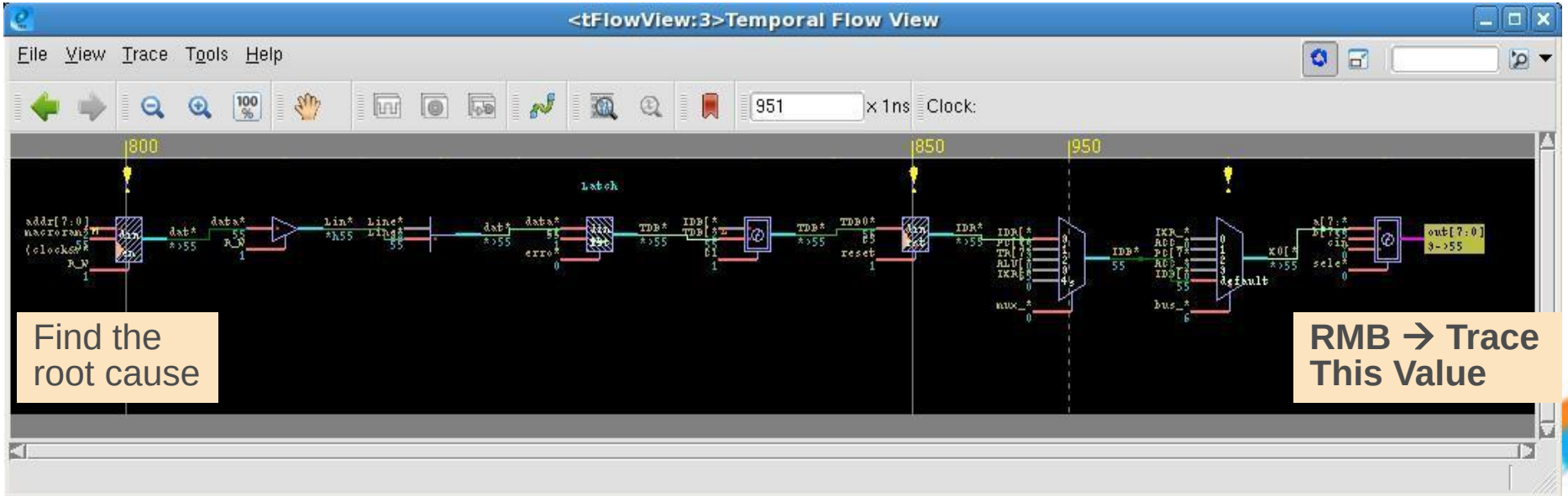


SpringSoft
Accelerating Engineers

Operate in the TFV

Trace This Value

- Use **Trace → Trace This Value** to automatically trace back the data path
  - Unroll the path on time axis, and find the root cause
  - Use **View → Active Data Nodes Only** or **View → Active Nodes Only** to simplify the view and display active nodes only

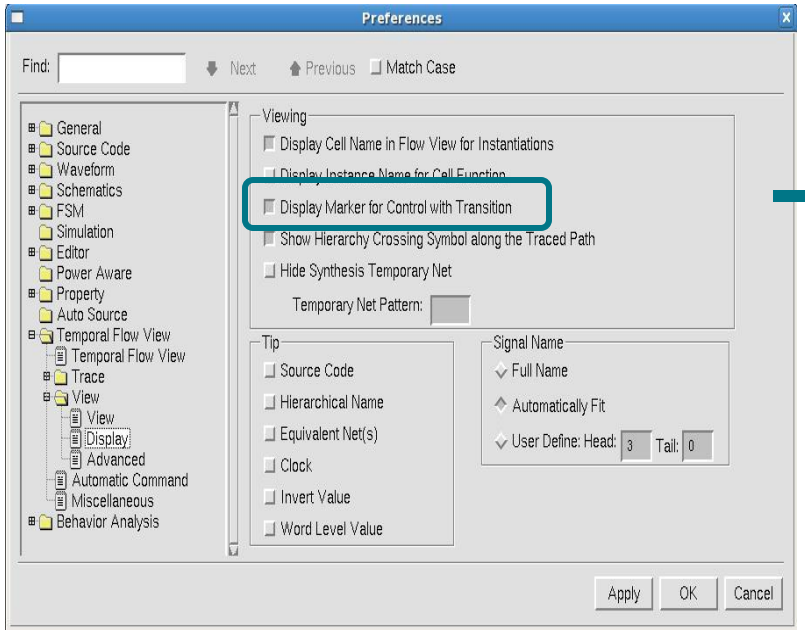


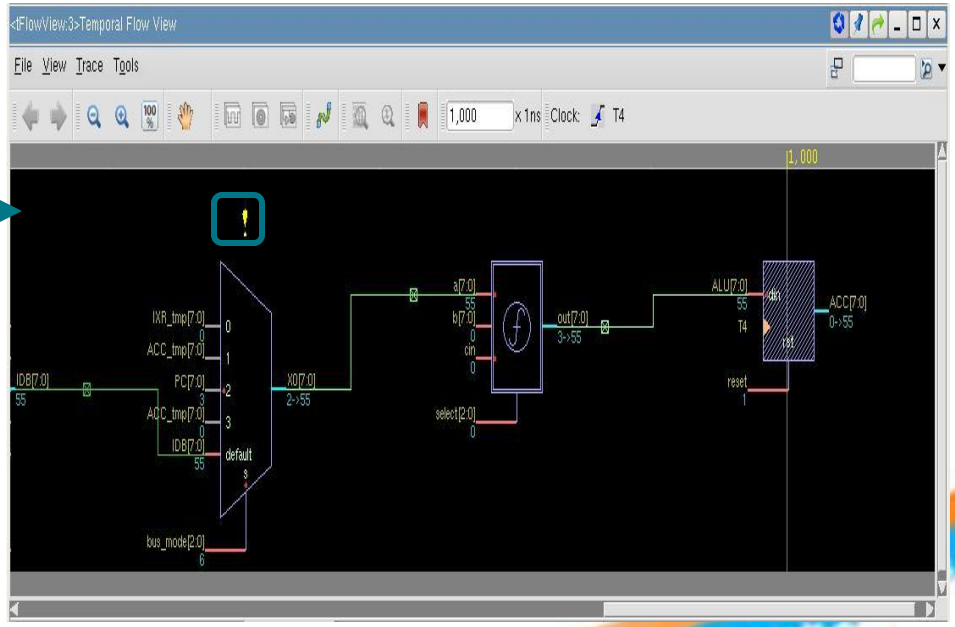
SpringSoft Accelerating Engineers

Operate in the TFV

Show Marker for Control with Transition

- Show the output nodes with transition on the control signal in the tracing path
  - Enable the **Display Marker for Control with Transition** option from **Tools → Preferences → Temporal Flow View → View → Display**
  - A yellow exclamation point is placed over output nodes



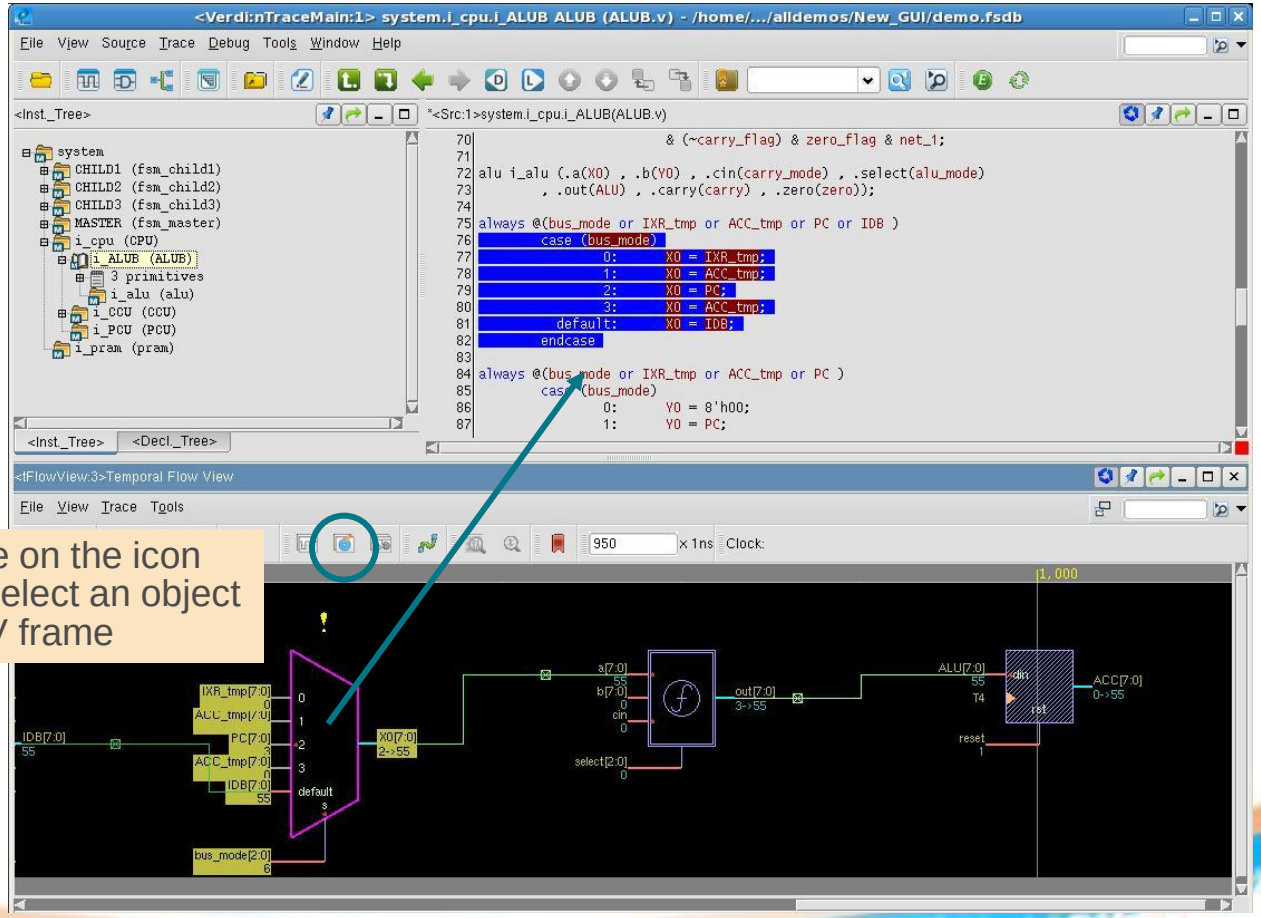


SpringSoft Accelerating Engineers

Correlate Other Views

Automatically Display Source Code

- Toggle on the **Show Source Code Automatically** icon or the **View → Show Source Code Automatically** option to view corresponding source code for a selected node

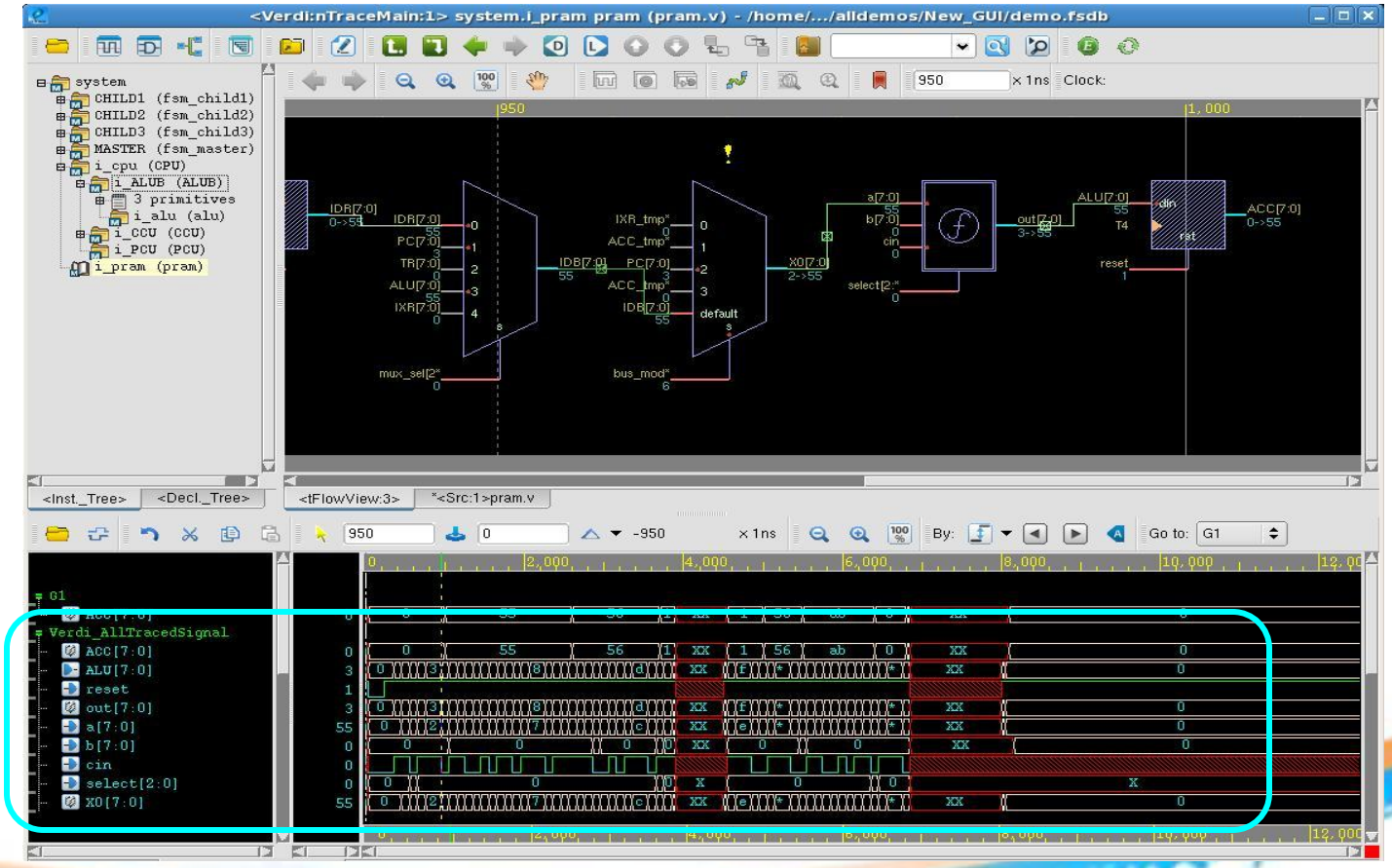


SpringSoft
Accelerating Engineers

Correlate Other Views

Show Traced Signals in nWave

- To show traced signals in the waveform:
  - In the Temporal Flow View, invoke **Tools → Show All Traced Signals on nWave** to add traced results to *nWave*

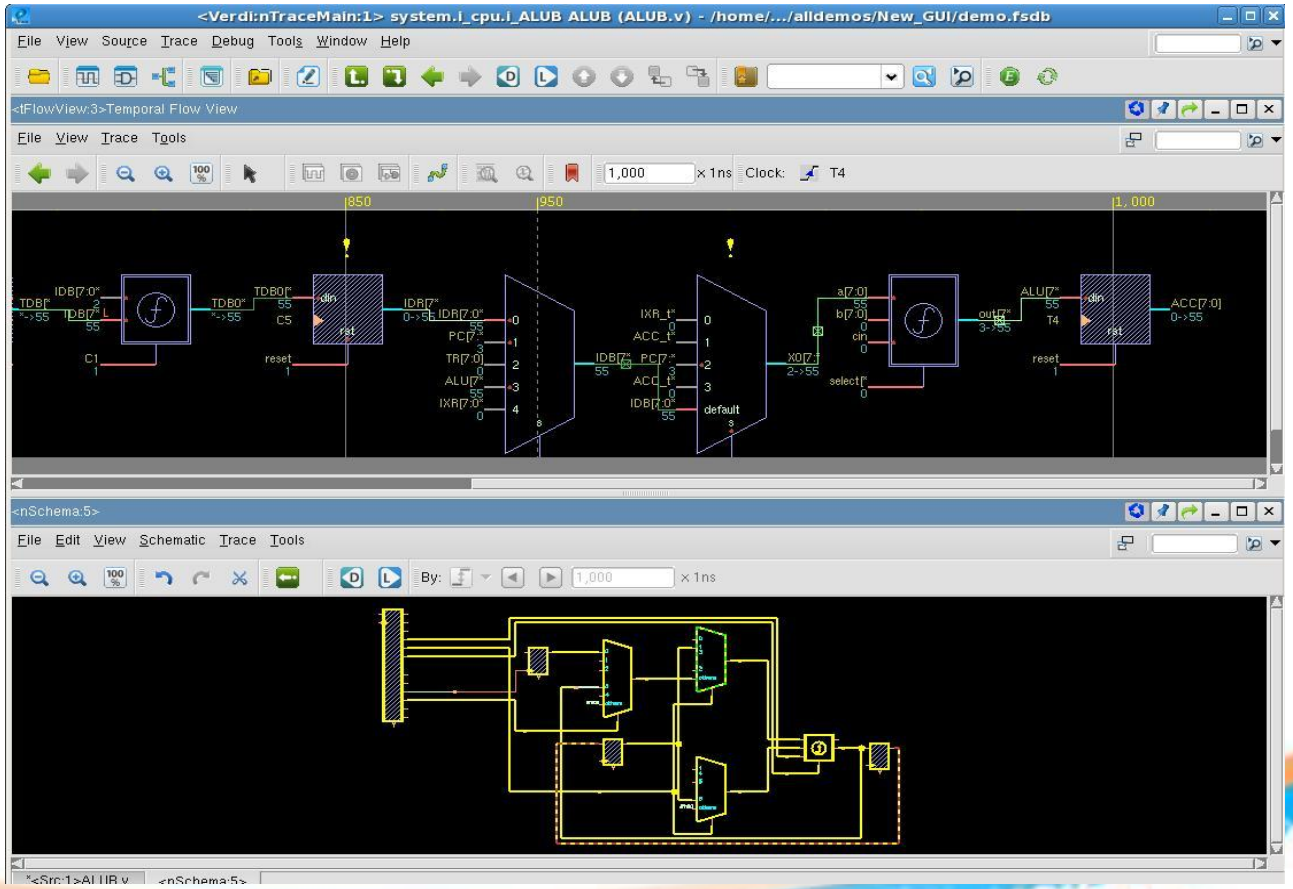


SpringSoft
Accelerating Engineers

Correlate Other Views

Show Fan-in Signals in nSchema

- To view a signal's fan-in in *nSchema*:
  - Select the signal
  - Use **Tools → Show Fan-ins on nSchema → Active Only**



SpringSoft
Accelerating Engineers

Labs

- Practice using the Verdi platform:
  - Refer to the *Verdi User's Guide and Tutorial* document in <VERDI3_HOME>/doc/VerdiTut.pdf
  - Follow the steps in all *Tutorial* chapters to practice using the Verdi platform
    - nTrace Tutorial
    - nSchema Tutorial
    - nWave Tutorial
    - nState Tutorial
    - Temporal Flow View Tutorial
    - Debug a Design with Simulation Results Tutorial
    - Application Tutorials

Summary

- In this training, you have learned…
  - About the Verdi platform technology
  - How to setup the environment for debugging
  - How to use tasks and utilities to dump FSDB
  - How to import the design into Verdi
  - How to debug in source code view
  - How to debug in waveform view
  - How to debug in schematic view
  - How to debug in FSM view
  - How to debug in Temporal Flow View

Topics

- Technology Background
- Set Up the Environment
- Understand FSDB Dumping Tasks and Utilities
- Import Design
- Verdi^3 Interface
- Debug in Source Code View
- Debug in Waveform View
- Debug in Schematic View
- Debug in FSM View
- Debug in Temporal Flow View
- Appendix: Frequently Used System Tasks

Appendix: Frequently Used Dumping Tasks

$fsdbDumpfile

- Syntax:
  - $fsdbDumpfile("FSDB_Name" | FSDB_Name_var [,Limit_Size | ,Limit_Size_var]);

- Arguments:
  - **FSDB_Name**
    - Specify the name of the FSDB file generated by the Novas object files for FSDB dumping.
  - **FSDB_Name_var**
    - Specify the FSDB file name in a variable.
  - **Limit_Size**
    - Specify the maximum FSDB file size in megabyte.
    - **NOTE:** The minimum file size of the FSDB file is 10M. If you set the size to less than 10M, it will be set to 10M instead.
  - **Limit_Size_var**
    - Specify the maximum FSDB file size in a variable.

Appendix: Frequently Used Dumping Tasks

$fsdbDumpfile - Examples

- Specify dump file as **novas.fsdb** and limit to 32 megabytes
  - $fsdbDumpfile("novas.fsdb", 32);

- Use variable:
  - reg [15:0] limit_size_reg = 1000;
  - string newDumpfile_string = "novas_new.fsdb";
  - reg [1023:0] newDumpfile_reg = "novas_new.fsdb";

  - $fsdbDumpfile(newDumpfile_string, 1000);
  - $fsdbDumpfile(newDumpfile_reg, limit_size_reg);

Appendix: Frequently Used Dumping Tasks

$fsdbDumpvars

- Syntax:
  - $fsdbDumpvars([depth, | "level=",depth_var,]
    [instance | "instance=",instance_var]
    [,"option" | ,"option=",option_var]*);

- Arguments:
  - **Depth**
    - 0: all signals in all scopes
    - 1: all signals in current scope
    - 2: all signals in the current scope and all scopes one level below
    - n: all signals in the current scope and all scopes n-1 levels below
  - **"level=“, depth_var**
    - Specify levels of hierarchy in a variable

Appendix: Frequently Used Dumping Tasks

$fsdbDumpvars

- **instance**
  - Gives either the module scope or signals in full hierarchy format
- **"instance=“, instance_var**
  - Gives either the module scope or signal by instance_var
- **option**
  - **"+all"**
    - Dump all signals including the memory, MDA, packed array, structure, union and packed structure signals. For VCS users, the VCS option *"+memcbk"* may be needed when compiling the design
  - **"+IO_Only"**
    - Only IO port signals will be dumped
  - **"+Reg_Only"**
    - Only reg type signals will be dumped
  - **"+strength"**
    - Enable strength dumping
  - **"+fsdbfile+*filename*"**
    - Specify the FSDB file name
  - **NOTE:** Only some options are listed above, check *$VERDI3_HOME/doc/linking_dumping.pdf* for all options

Appendix: Frequently Used Dumping Tasks

$fsdbDumpvars - Examples

- Dump all signals in the module instances below this module
  - $fsdbDumpvars;
- Dump all signals in the module instances below system to the specified FSDB file named “My.fsdb“
  - $fsdbDumpvars(0, system, "+fsdbfile+My.fsdb");
- The following two commands are equal:
  - $fsdbDumpvars(0, design_top, "+fsdbfile+novas.fsdb");
  - $fsdbDumpfile("novas.fsdb"); $fsdbDumpvars(0, design_top);

Appendix: Frequently Used Dumping Tasks

$fsdbDumpon, $fsdbDumpoff

- Syntax:
  - $fsdbDumpon(["option"| option_var]);
  - $fsdbDumpoff(["option"| option_var]);

- Arguments:
  - option:
    - +fsdbfile+*filename*
      - For *$fsdbDumpon*, specify the target FSDB file name for “Dump On”. If not specified, dumping for all FSDB files will be turned on
      - For *$fsdbDumpoff*, specify the target FSDB file name for “Dump off”. If not specified, dumping for all FSDB files will be turned off
  - option_var:
    - Specify the FSDB file name option in a variable

Appendix: Frequently Used Dumping Tasks

$fsdbDumpon, $fsdbDumpoff - Examples

- Control dumping off/on for default novas.fsdb:

```
initial
begin
#50000
$fsdbDumpvars;
#50000
$fsdbDumpoff;
#50000
$fsdbDumpon;
#50000
$finish;
end
```

- Dump off **My.fsdb** only (when specifying multiple FSDB files in a simulation run):
  - $fsdbDumpoff("+fsdbfile+My.fsdb");

SpringSoft
Accelerating Engineers

Appendix: Frequently Used Dumping Tasks

$fsdbSuppress

- Syntax:
  - $fsdbSuppress("suppress_file" | "file=",file_var | [instance]*, | "instance=" [, instance_var]* | "module_file=", "filename" | "module_base="[, "module_name"]*);
- Arguments:
  - **suppress_file**
    - Specify the file containing the names of the scopes that will not be dumped
  - **Instance**
    - Specify one or more instances that will not be dumped. Note that the "instance" includes module scope and signals
  - **"module_file="**
    - Specify the file containing the names of the modules that will not be dumped
  - **"module_base="**
    - Specify the module from which the instantiated scopes will not be dumped
- **NOTE:** *$fsdbSuppress is a global setting (affect all FSDB files), and* must be specified before *$fsdbDumpvars*, *$fsdbDumpvarsByFile*, and *$fsdbDumpMDA*

Appendix: Frequently Used Dumping Tasks

$fsdbSuppress - Examples

- Suppress signals under top.u1.u2 and top.x1.x1:
  - $fsdbSuppress(top.u1.u2, top.x1.x2);
- Suppress scope from a list:
  - $fsdbSuppress("./suppress.lst");
  - where **suppress.list** contains:

    system.i_cpu.C0
    system.i_cpu.C1
    system.i_cpu.C5
    system.i_cpu.C6
- Suppress modules:
  - $fsdbSuppress("module_base=", "module_1", "module_2", "module_3");
- Suppress signals:
  - $fsdbSuppress(“instance=", "system.U1.sub.reg1", "system.U1.sub.reg2");

Appendix: Frequently Used Dumping Tasks

$fsdbAutoSwitchDumpfile

- Automatically switch to a new dump file when the working FSDB file hits the specified size limitation
  - Syntax:
    - $fsdbAutoSwitchDumpfile(File_Size | File_Size_var, "FSDB_Name" | FSDB_Name_var, Number_of_Files | Number_of_Files_var [,"log_filename" | ,log_filename_var]);
  - Arguments:
    - **File_Size**
      - The FSDB file size limit. The unit is megabyte
      - The minimum file size of the FSDB file is 10M
    - **FSDB_Name**
      - Specify the FSDB file name
    - **Number_of_Files**
      - The maximum number of FSDB files to generate. 0 means no limitation

Appendix: Frequently Used Dumping Tasks

$fsdbAutoSwitchDumpfile - Example

- Limit each FSDB file size to 10 MB, and limit the file number to 20
  - $fsdbAutoSwitchDumpfile(10, "test.fsdb", 20);
  - $fsdbDumpvars(0, system);
    - The first FSDB file will be named *test_000.fsdb*. The second FSDB file *test_001.fsdb* will be created when the file size of *test_000.fsdb* reaches near 10MB. The third FSDB file *test_002.fsdb* will be created when the file size of *test_001.fsdb* reaches near 10MB and so on.
    - The first FSDB file *test_000.fsdb* will be overwritten when the file size of *test_019.fsdb* reaches near 10MB.