# Advanced, High-Throughput Debug From Design to Silicon

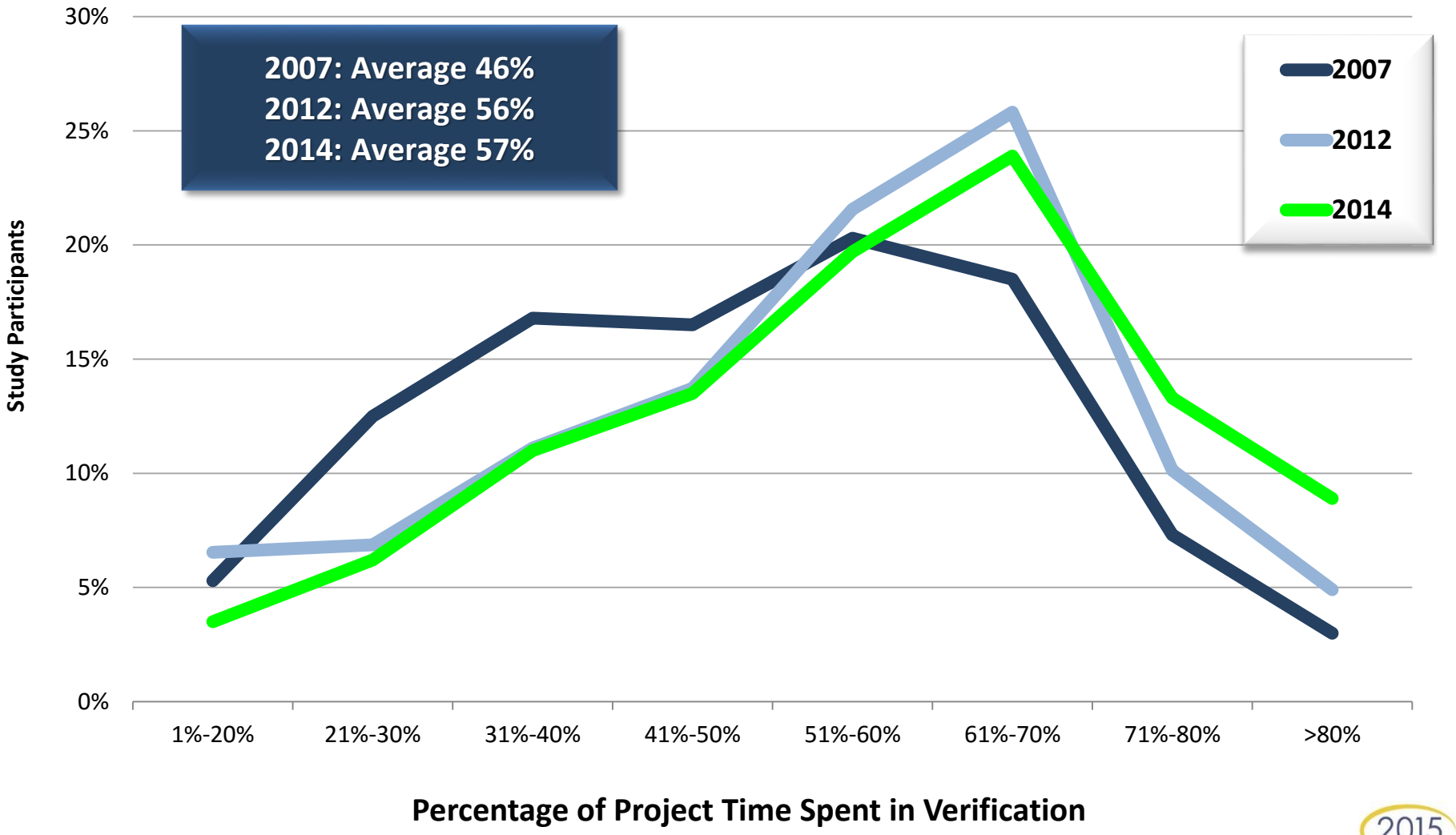
Gordon Allan & Michael Horn
Mentor Graphics Inc



# Agenda

- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

# Verification Consumes Majority of Project Time



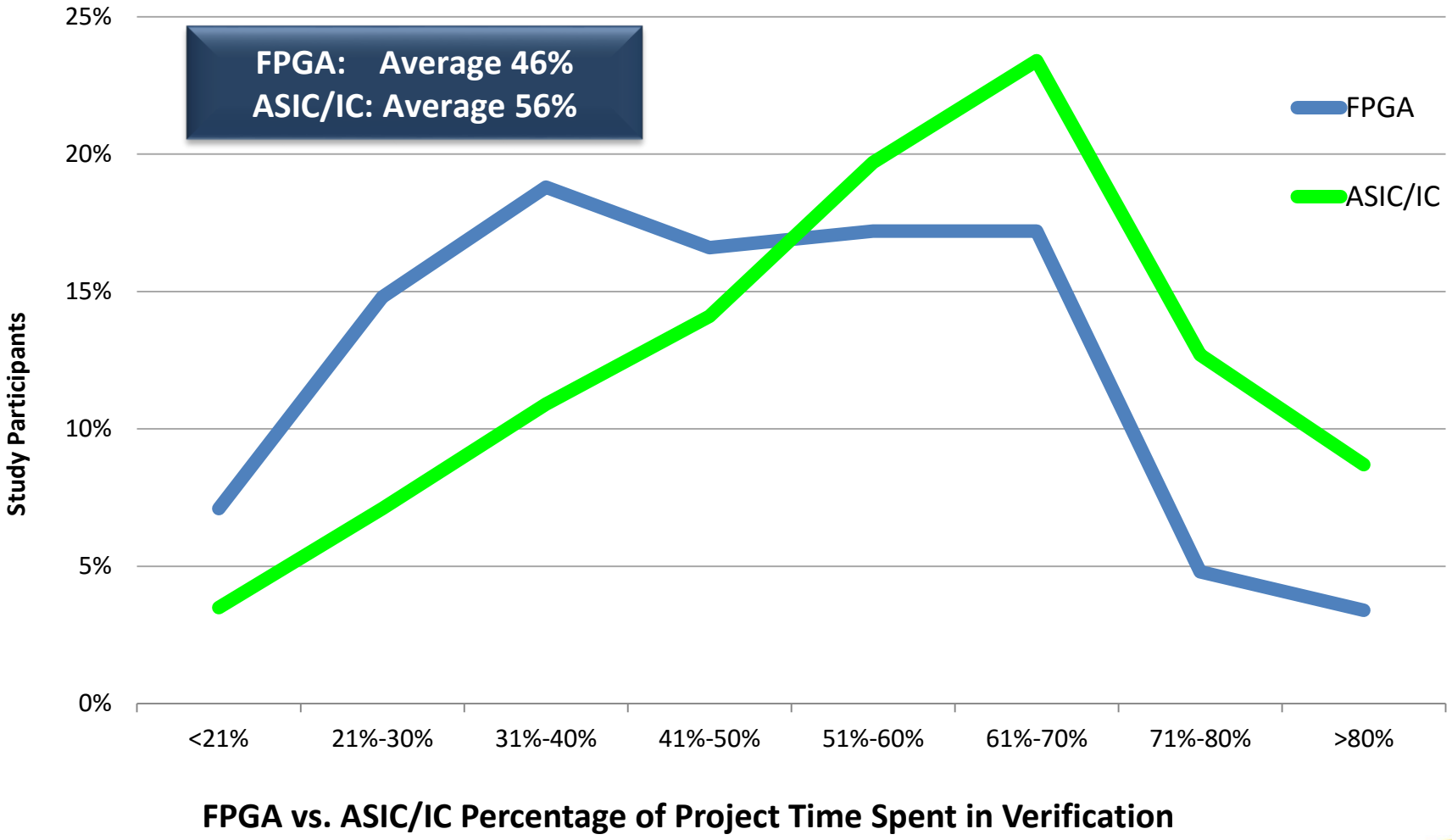

*Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study*



accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Verification Consumes Majority of Project Time for FPGAs too




**FPGA vs. ASIC/IC Percentage of Project Time Spent in Verification**

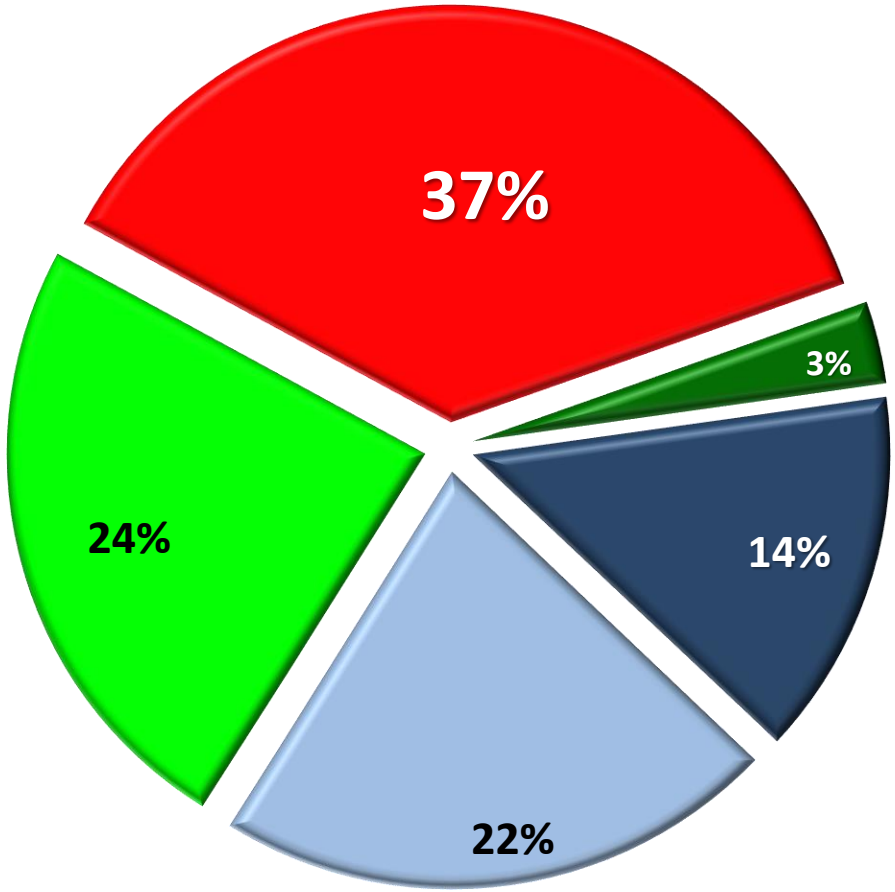
*Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study*



accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Where Verification Engineers Spend Their Time

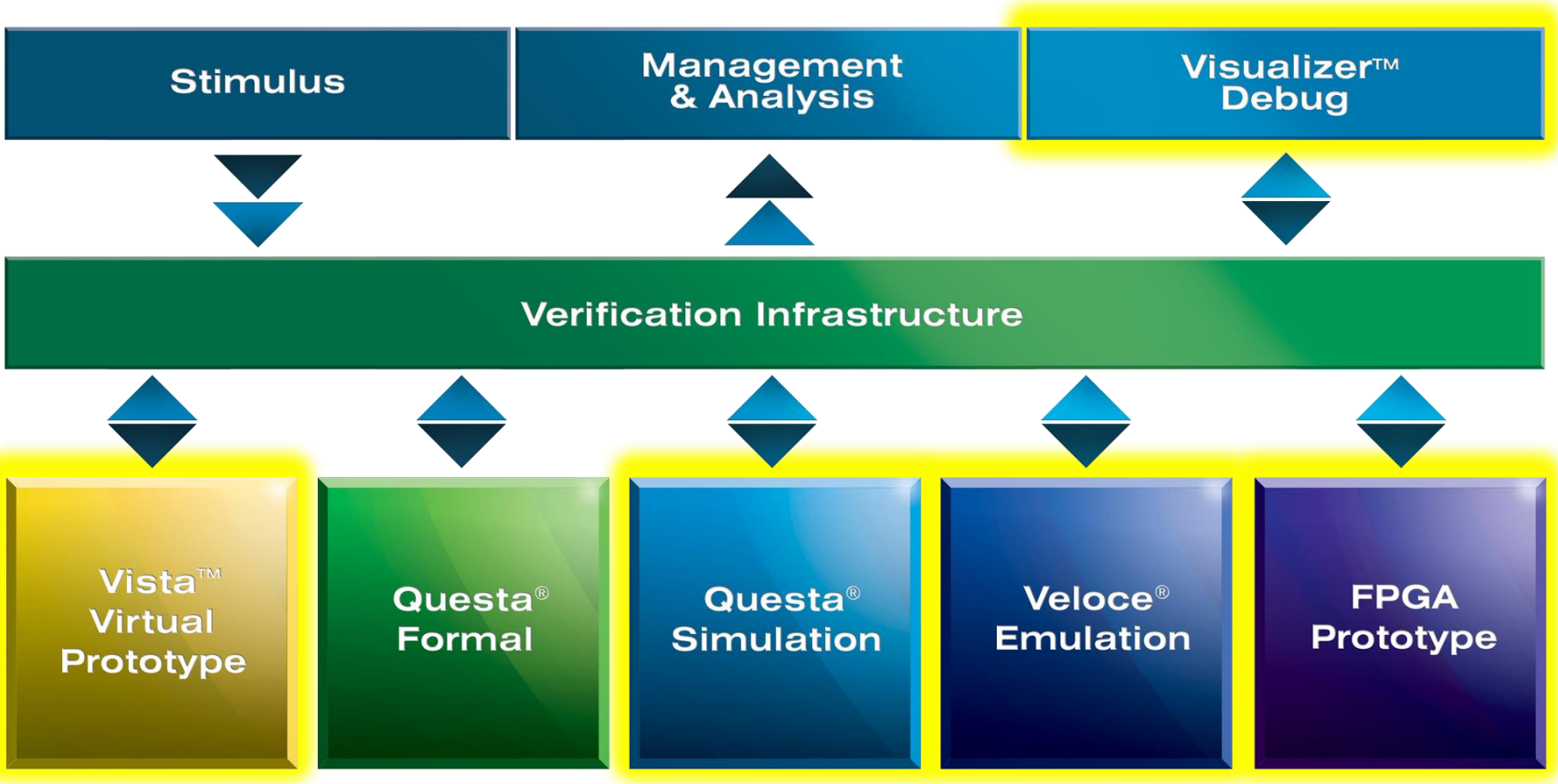



*Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study*

# Key Recommendations

- Find bugs as early as possible
  - Before time-consuming, resource intensive regressions are launched
- Successively refine low power verification in every D&V phase
- Leverage new debug technologies to expedite causality discovery
- Break down the wall between RTL and firmware verification
- Plan ahead for post-silicon debug

# The Enterprise Verification Platform



# Agenda

- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

# Debugging RTL with Visualizer

- Demonstration of example scenarios

# Visualizer Debug Environment Architecture

## *Built for Performance/Capacity for the Largest SoCs*

- Fast GUI data model
  - Technology acquired from Axiom Design Automation
  - Integrated with Questa & Veloce compilation and execution flows
  - Capacity to load 500M+ gate designs in seconds

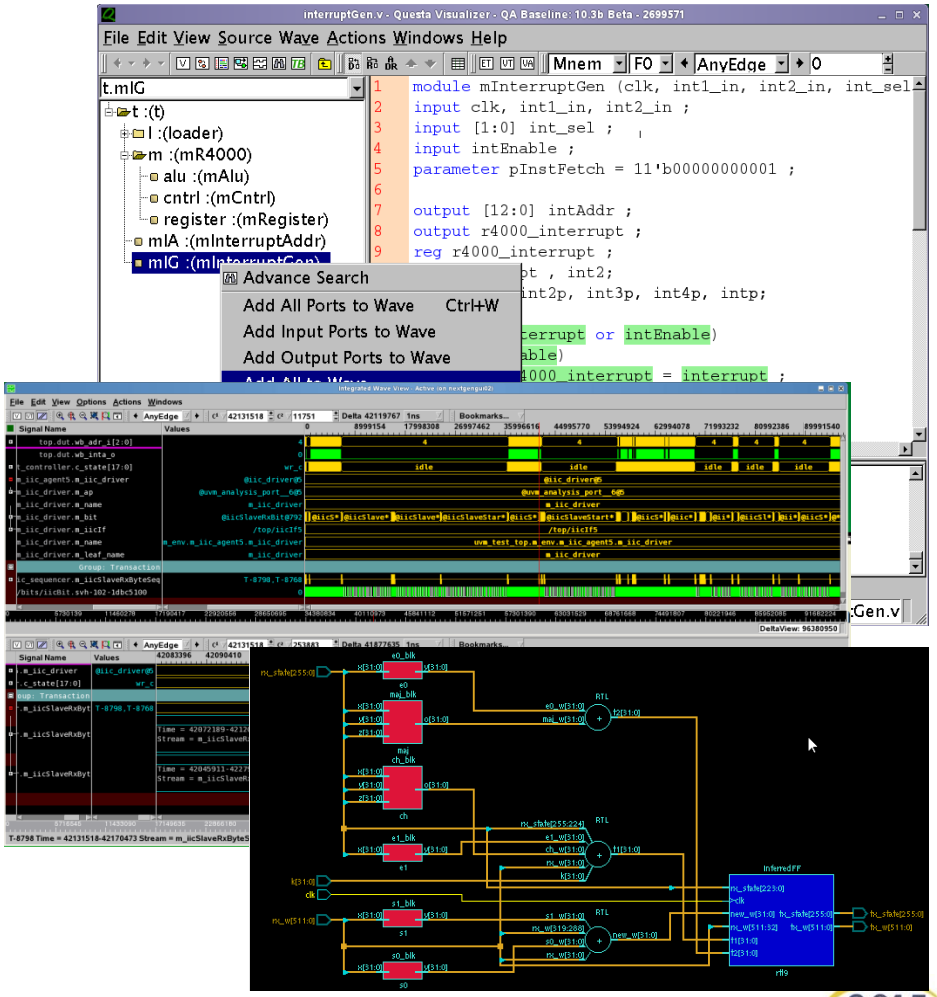
- Fast and efficient waveform database
  - Custom compact database format
  - Standalone option to enable worldwide debug efforts
  - On-demand connectivity for fast loading times

- High performance and capacity
  - “Smarter” integration with QuestaSim for even more capacity
  - Common tasks happen instantly on 200M+ gate designs

# Visualizer is Intuitive for Everyday Debug Tasks

*Fast and efficient where you spend 85% of your time*

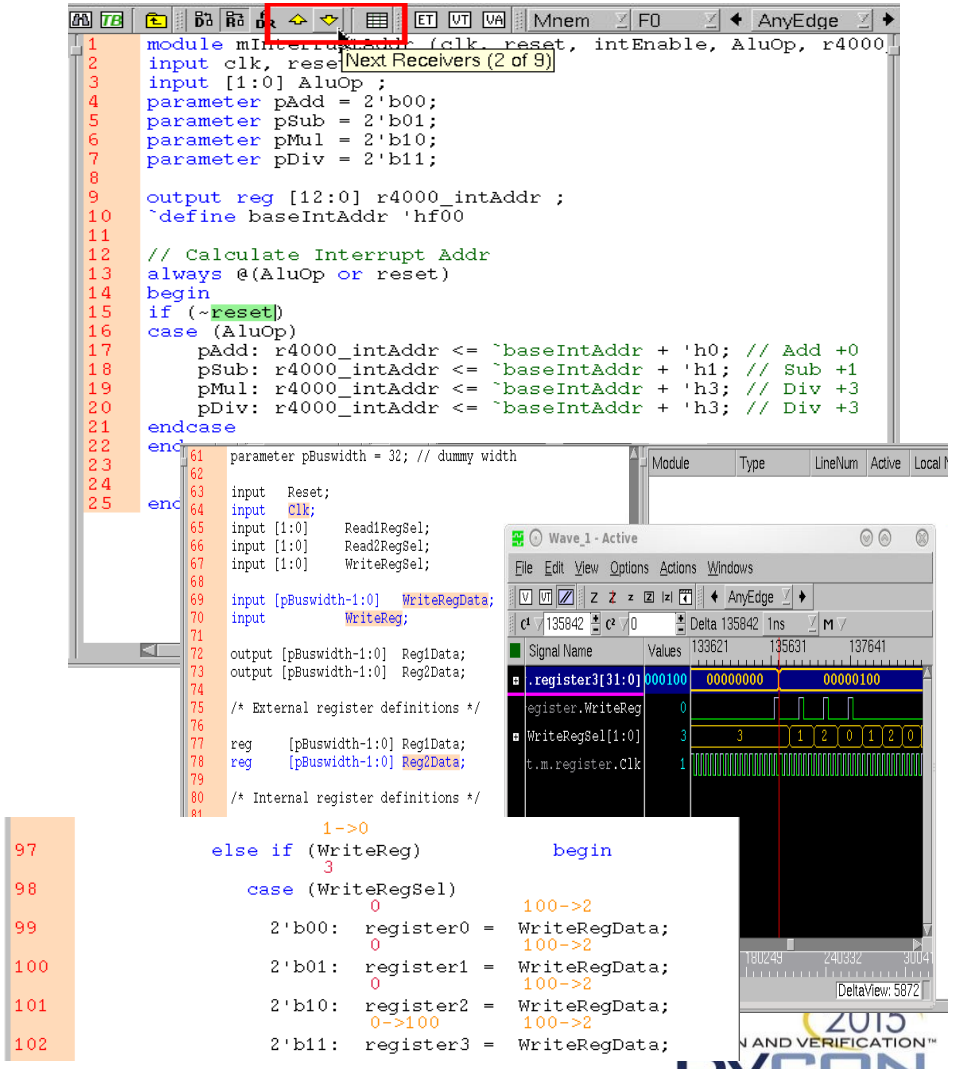
- Intuitive
  - Natural layout
  - Easy navigation
  - Quick mouse-clicks
- Responsive
  - Fast waveform
  - Fast source windows
  - All in time-sync
- Advanced Schematic View
  - Full featured
  - For both RTL and gates



# Powerful Design Comprehension Capabilities

*Finding Drivers, Receivers, and design highlight is fast and easy*

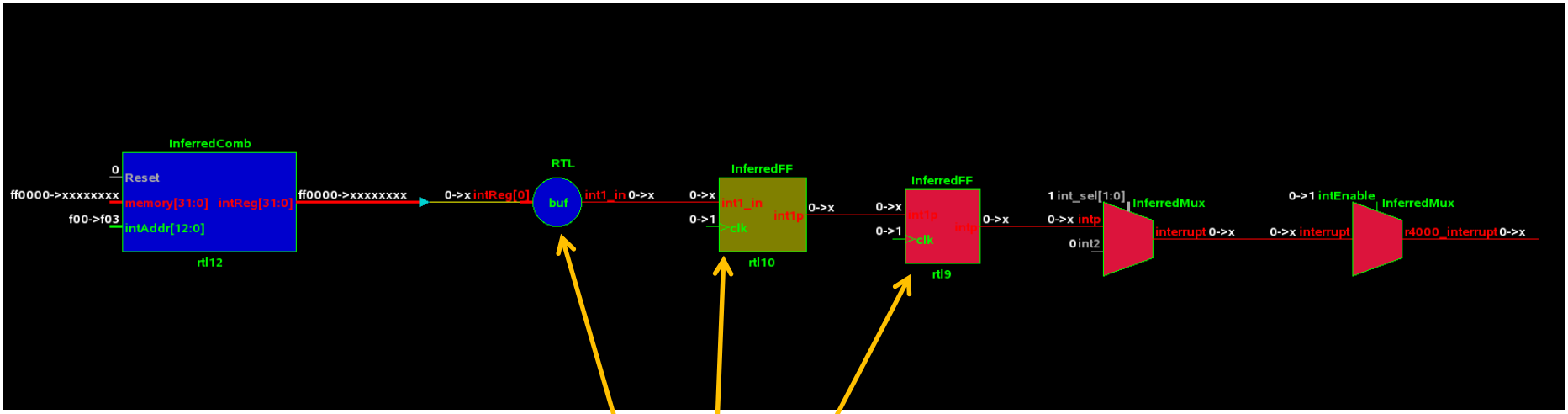
- 3 key navigation capabilities
  - Drivers
  - Receivers
  - Active Drivers / Receivers
- Easy to trace back and forth
  - Single click
- Quickly pinpoint activity in time
  - Execution trace
  - Popup tool tips
  - Value annotation



# Unique TimeCone View to Find Cause Faster

*Quickly trace back through time and view the cause of an event*

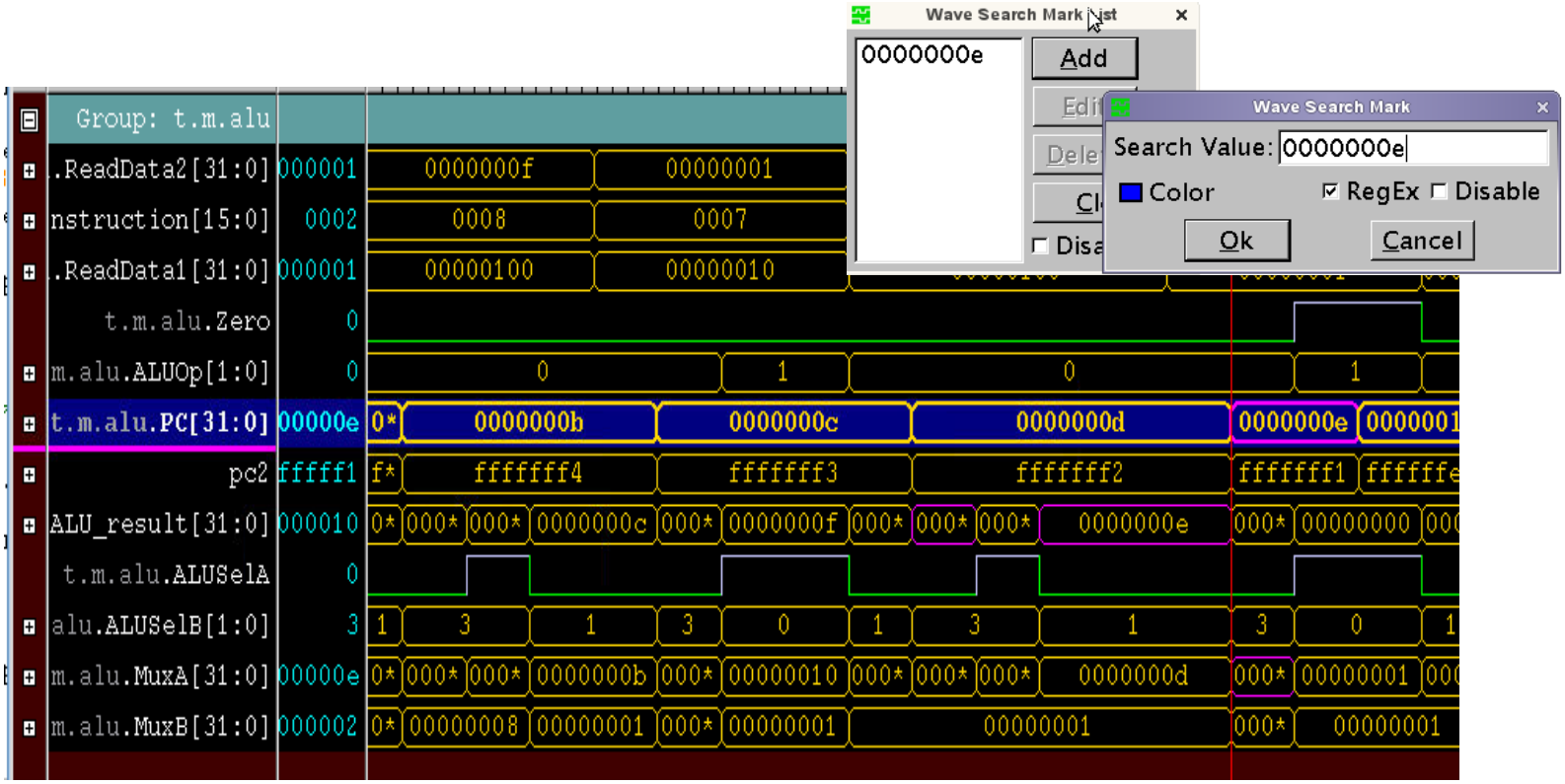
- Trims down to just show the path that is relevant




Different colors denote different simulation times

# Searching is Both Easy and Powerful

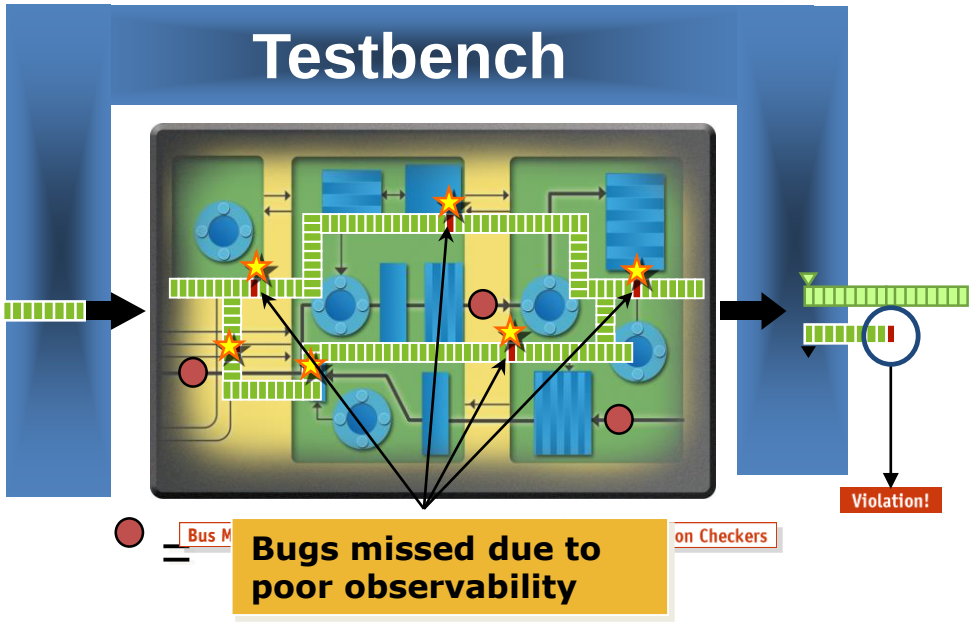
*Both design wide and local*

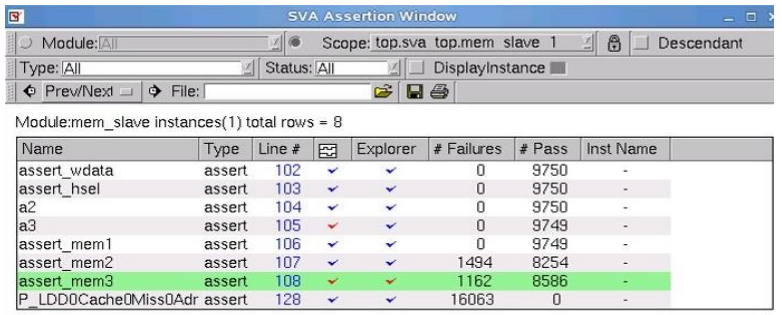


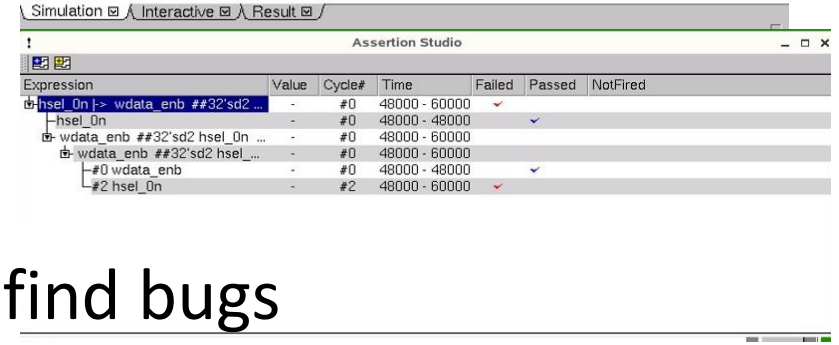

- Easy searching and filtering in any window to find objects
- Biometric search to add a “color” tag to a search so that it stays highlighted to see trends

# Adding Assertions: A Powerful RTL Debug Tool

*Visualizer makes it easier and faster to get them right*




- Assertions make it easier to find bugs
- Easy to create and debug assertions
  - Assertion Explorer
  - Interactive Replay

# Agenda

- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

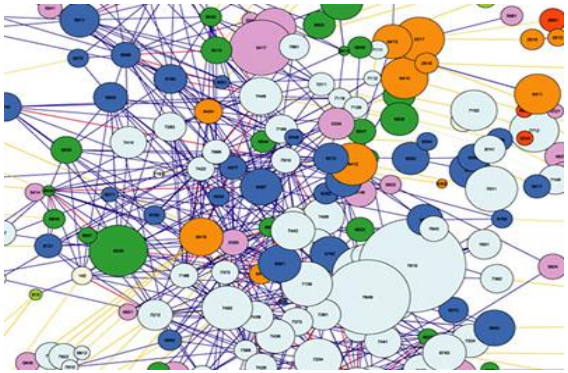
# Debugging Complex Testbenches

- Demonstration of example scenarios

# Modern Testbenches Add Complexity to Debug

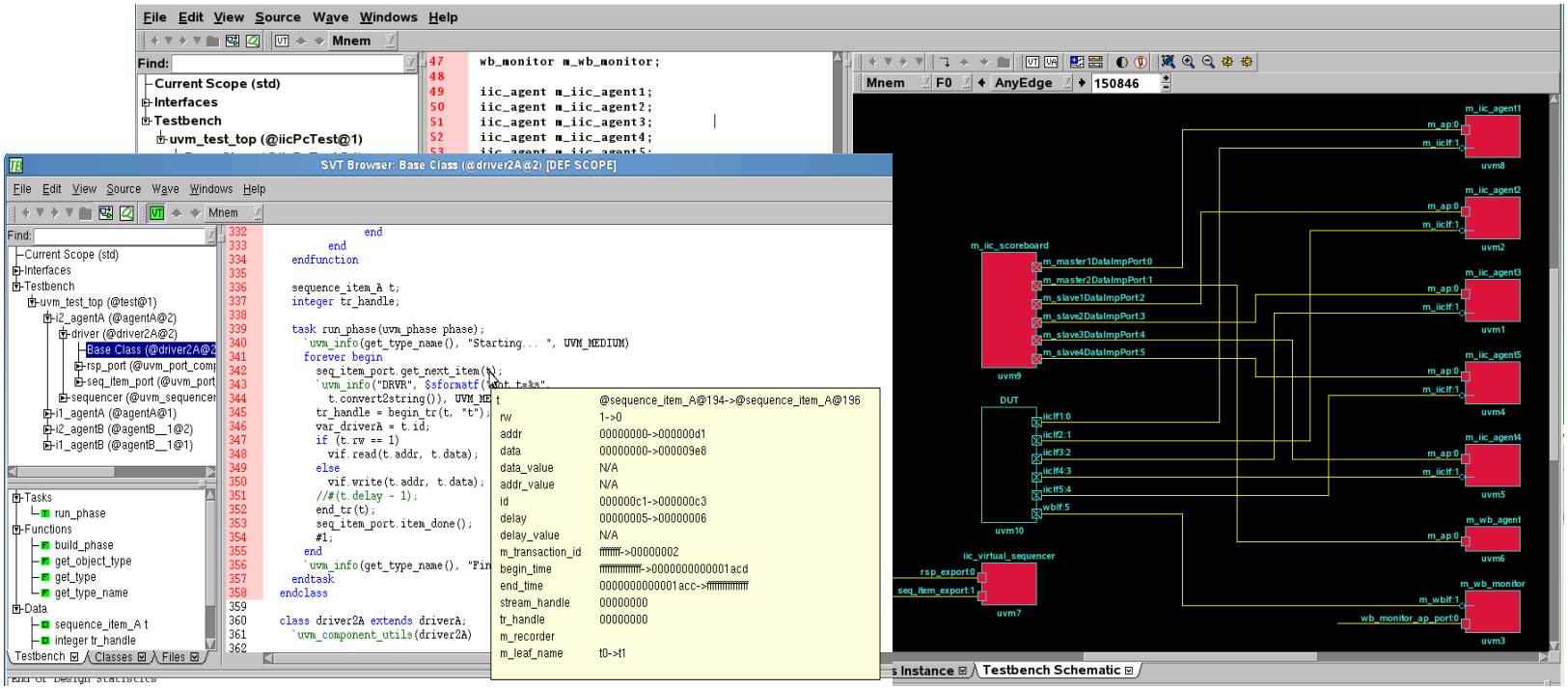
*Users report often 50% of debug time is in testbench*

- Testbench data is dynamic
  - How can you log it?
  - How do you find objects and with what name?
- Testbench is complex structures such as classes
  - How do you look at all of the handles and variables?
  - How do you see what is happening over time inside a class?
- Testbench is object-oriented and multi-threaded
  - How do you see inheritance?
  - How do you breakpoint or view a particular thread?



# Classes and UVM Debug in Post-Sim Mode

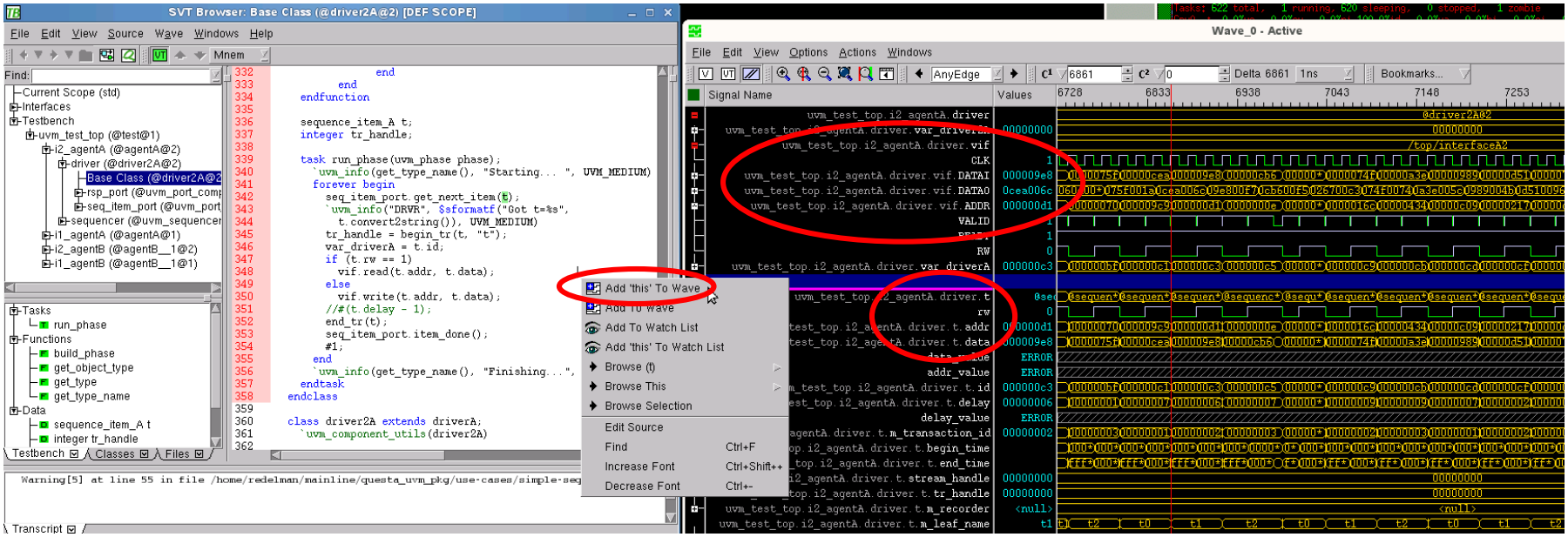
*TB debug features normally only found in interactive debug*




- All data available at no performance cost
- Easy and Powerful Browse and Analysis of Objects
- Full schematic and waveform features
- No more relying on print statements

# Classes/UVM data in Waveform

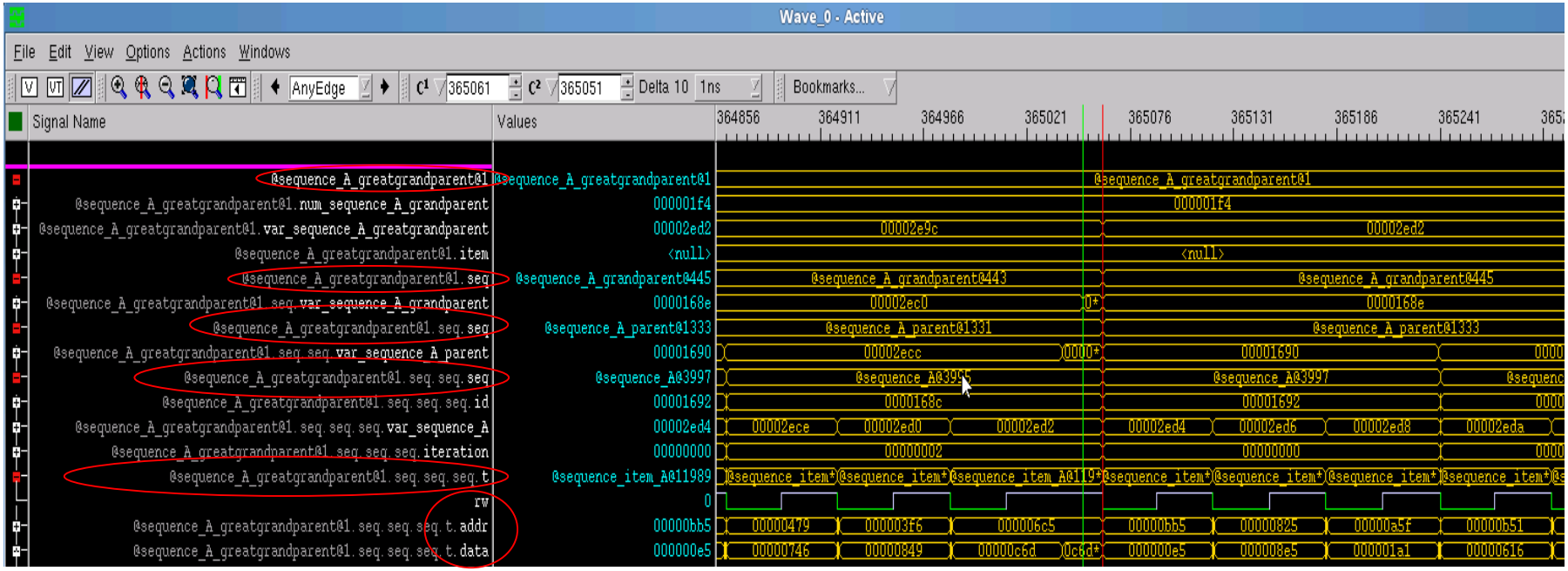
*Easily explore data and member values over time*




- Easy to drag into waveforms from source or object –
- Find ‘driver’ - 1 step to add ‘this’ handle to waveform
  - Expand and see its data, address, etc. Everything over time!
- Simple to see any testbench transaction to your DUT

# Intuitive Debug for Your UVM Sequences

*Easy to track, trace and see your stimulus*

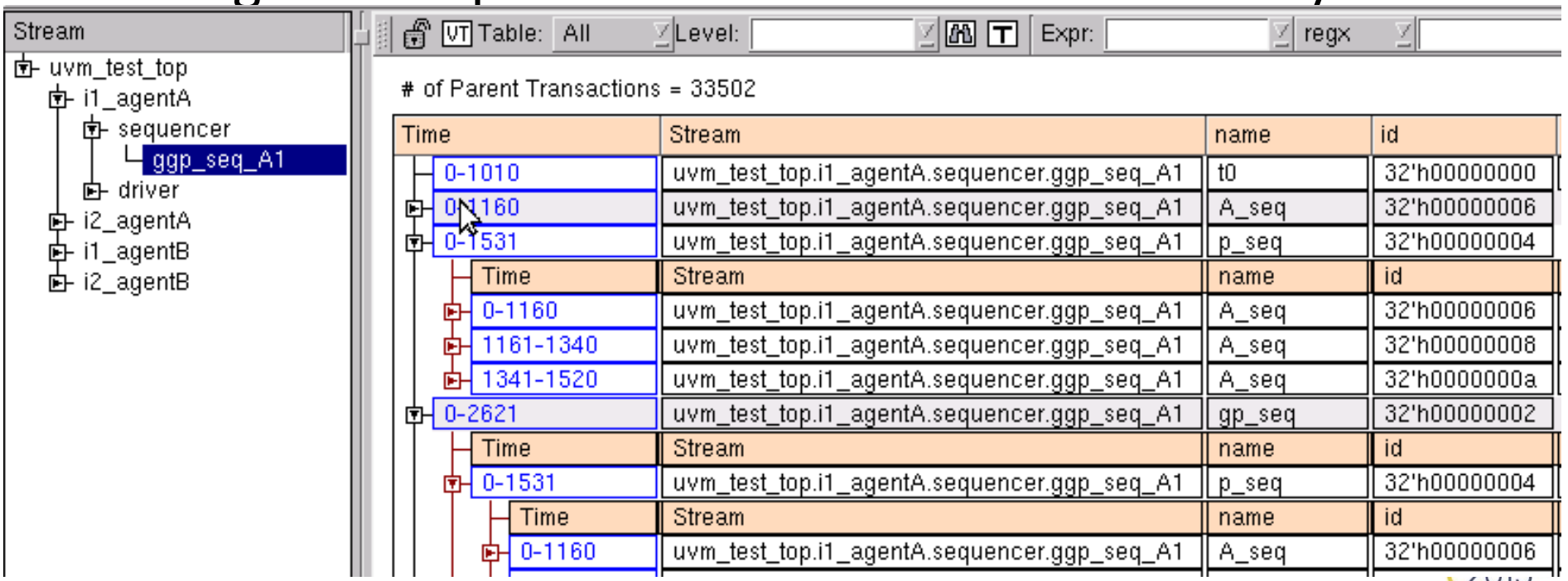



- Visualizer has powerful class-based debug capabilities
  - See parent child relationships, find key sequences, etc.
- Questa automatically records sequences as transactions

# Raising Abstraction: Transaction-Level Debug

*Easy, Powerful Waveforms and Transaction Stripe View*

- Transactions listed in start time order
- Powerful filter and search
- Drag and drop into waveform for detailed analysis



# Testbench Debug Requiring Interactive Mode

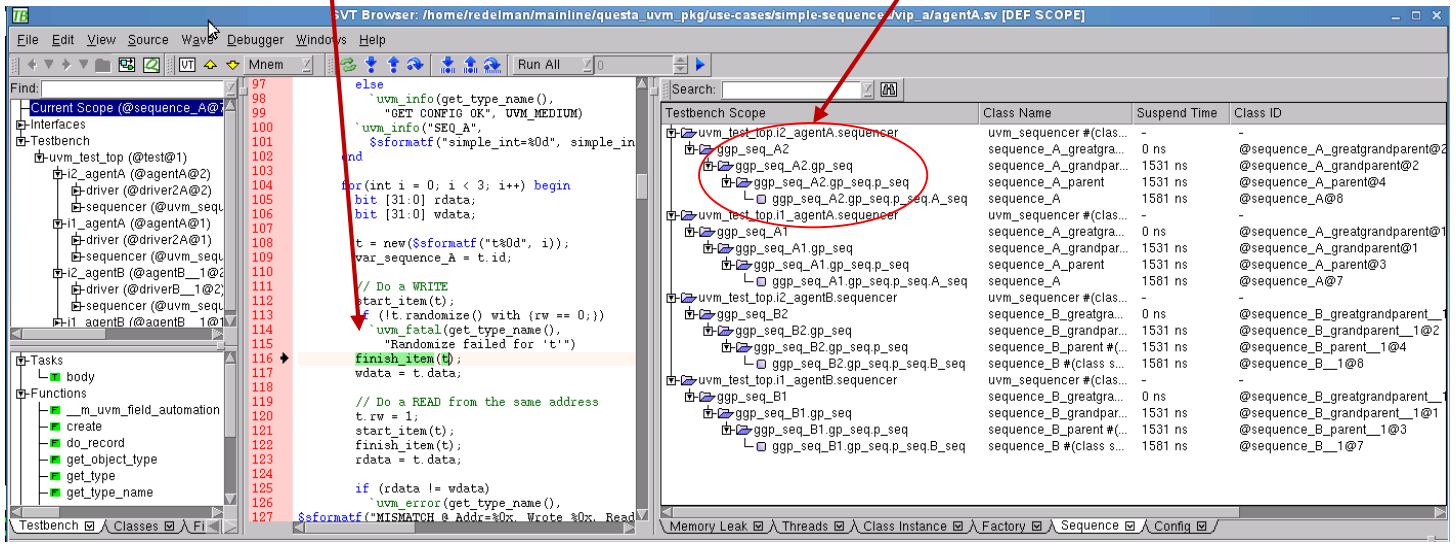
- Just creating the testbench
  and need to slowly step through it?
- Something is happening at initialization
  that is not right?
- Need to run the simulation up to a
  certain point, stop and explore?
- Need to understand how dynamic events
  and data are being created especially the threading?

# UVM and Class Debug in Interactive Mode

*Even more capabilities available in interactive mode*

Set breakpoint on line, class object or in base class object

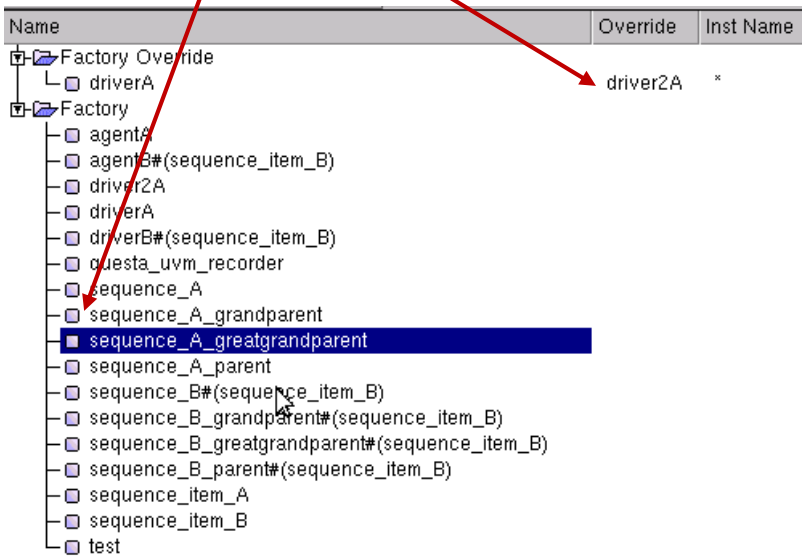
Dynamic sequence hierarchy



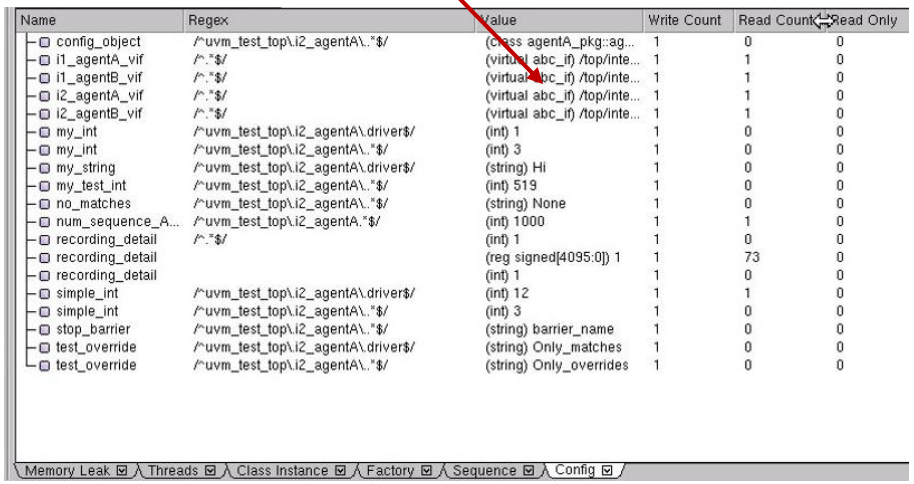

- Interactive mode has everything post-sim does plus…
- Breakpoints – So you can stop the simulation in any object
- Hierarchy of currently active sequences

# UVM Factory and Configuration Debug

Factory parameterized classes and any overrides all in one place




Config database – readers, writers and who set what

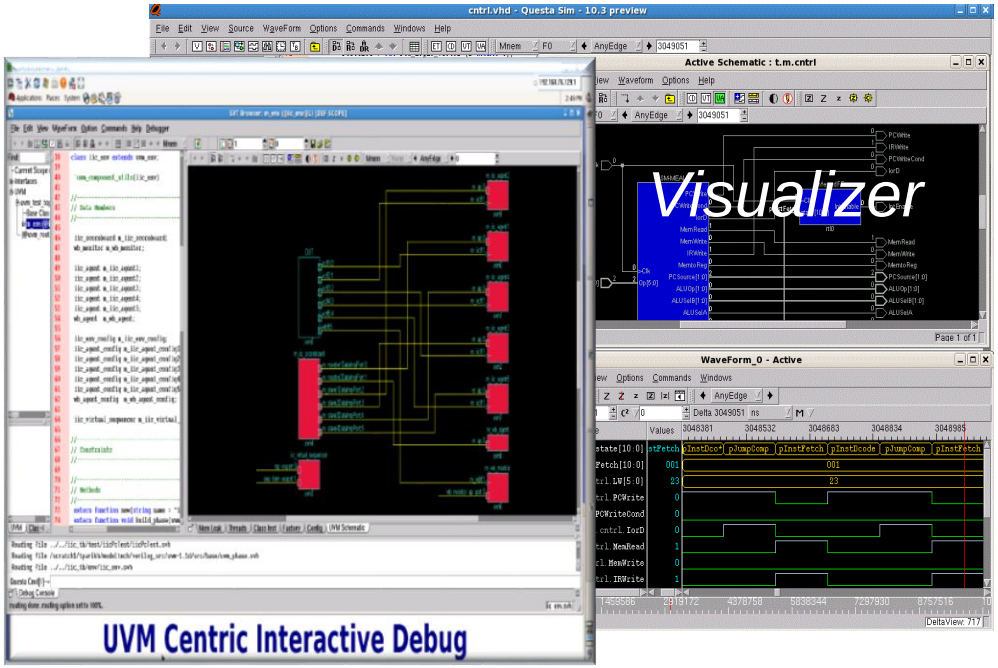



- Explore who, what and where things are set and used
- Resolve common UVM initialization issues

# Visualizer: Unified Debug for Questa and Veloce

*Maximize performance and still have ease of use*

Questa®





- Same debugger for both simulation and emulation
- Unique debug for TBX flow – TB from sim, DUT from emulation
- On-demand data loading for fast response with huge data

accellera SYSTEMS INITIATIVE

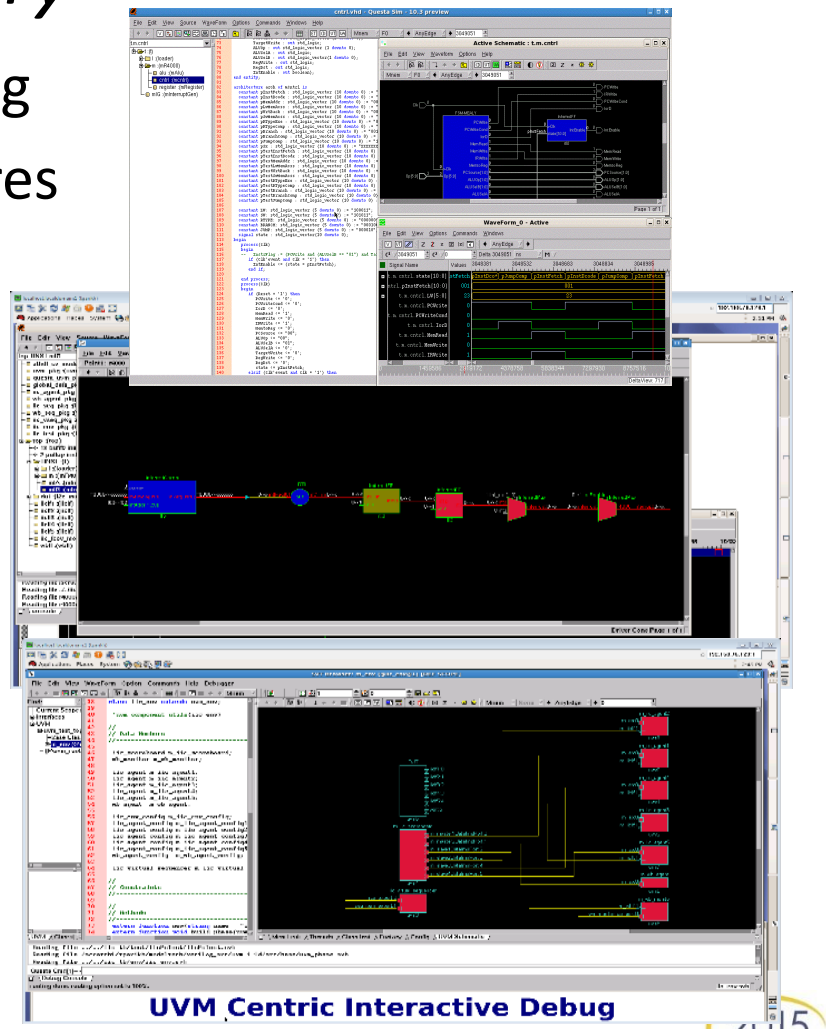


2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE

# Visualizer Debug Environment

## *Summary*

- High Performance/Capacity Debug
- Powerful Advanced Debug Features
- Assertion Debug
- Integrated Testbench
- Transaction Debug
- Full SystemVerilog, VHDL
- Mentor VIP integration
- Post-simulation use model
- O/S: Linux 32-bit, 64-bit



# Agenda

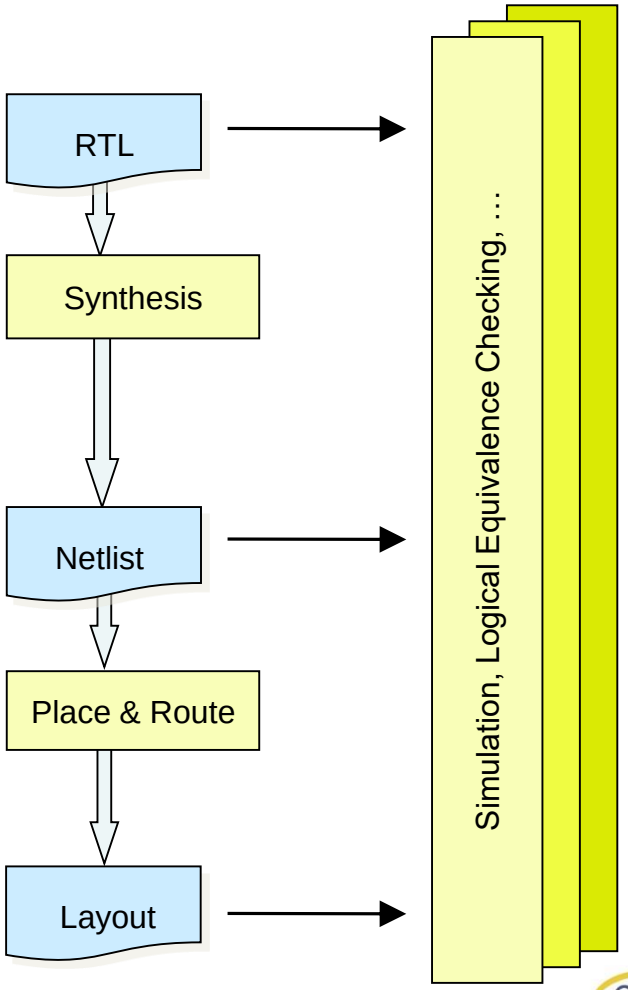
- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

# Debugging Low Power Designs

- Demonstration of example scenarios

# Traditional Design and Verification Flow

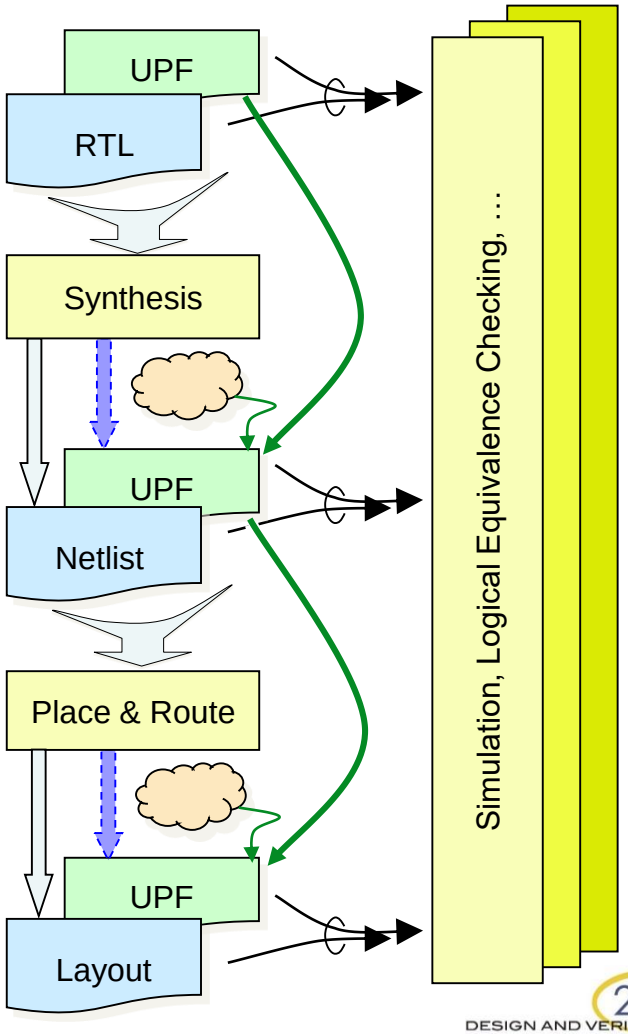
- RTL design
  - Captures design intent
  - Drives functional verification
  - Drives synthesis
- Logical implementation
  - Using standard cells and macros
  - Further modified for test, ECOs, etc.
- Physical implementation
  - Place & Route completes implementation
  - Produces manufacturing data
- Power management
  - Inherently part of physical implementation
  - Not represented in earlier stages
  - Not easy to optimize or redesign at this point



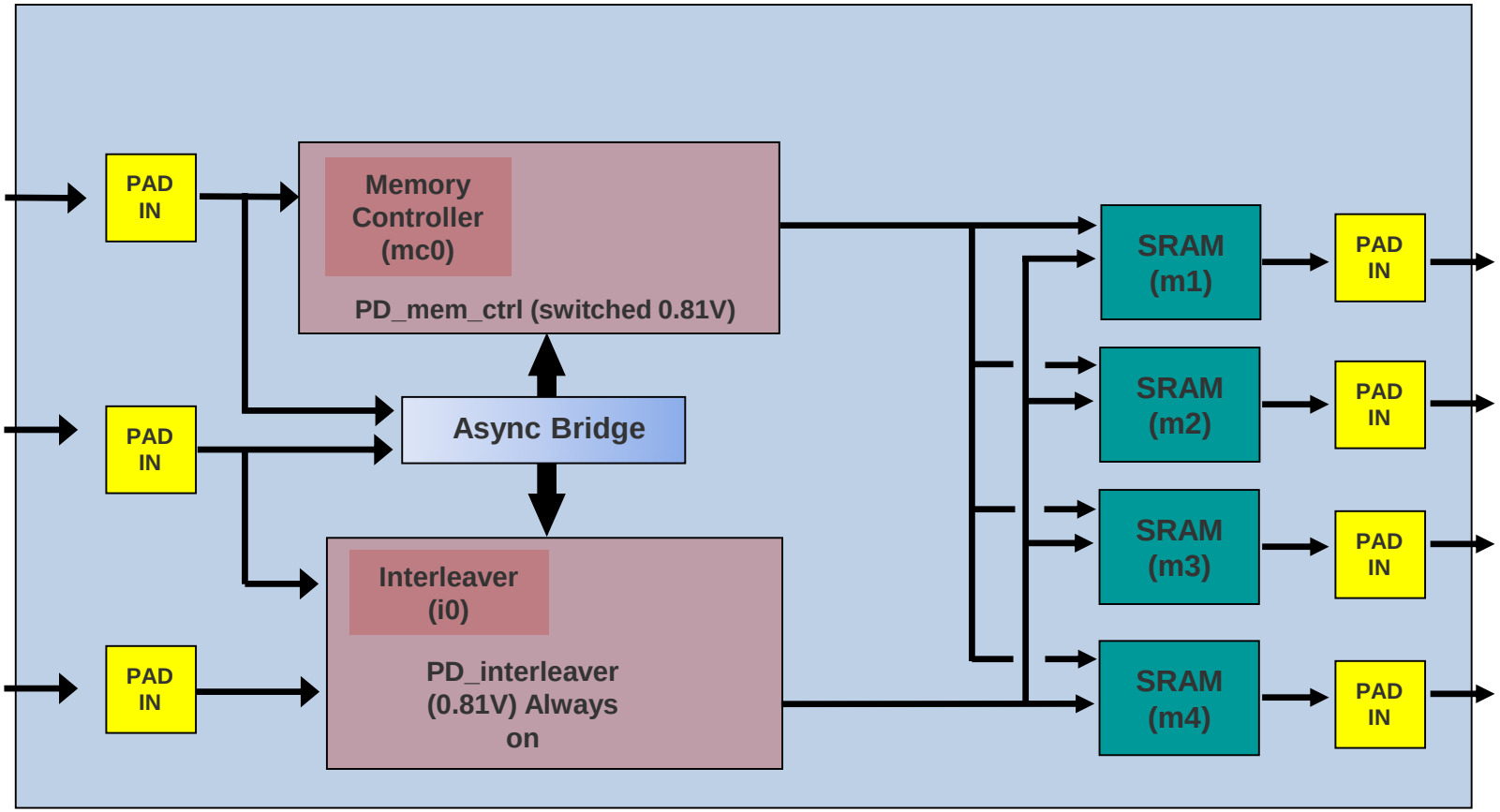

Can we verify power management earlier in the flow?

# UPF-Based Design and Verification Flow

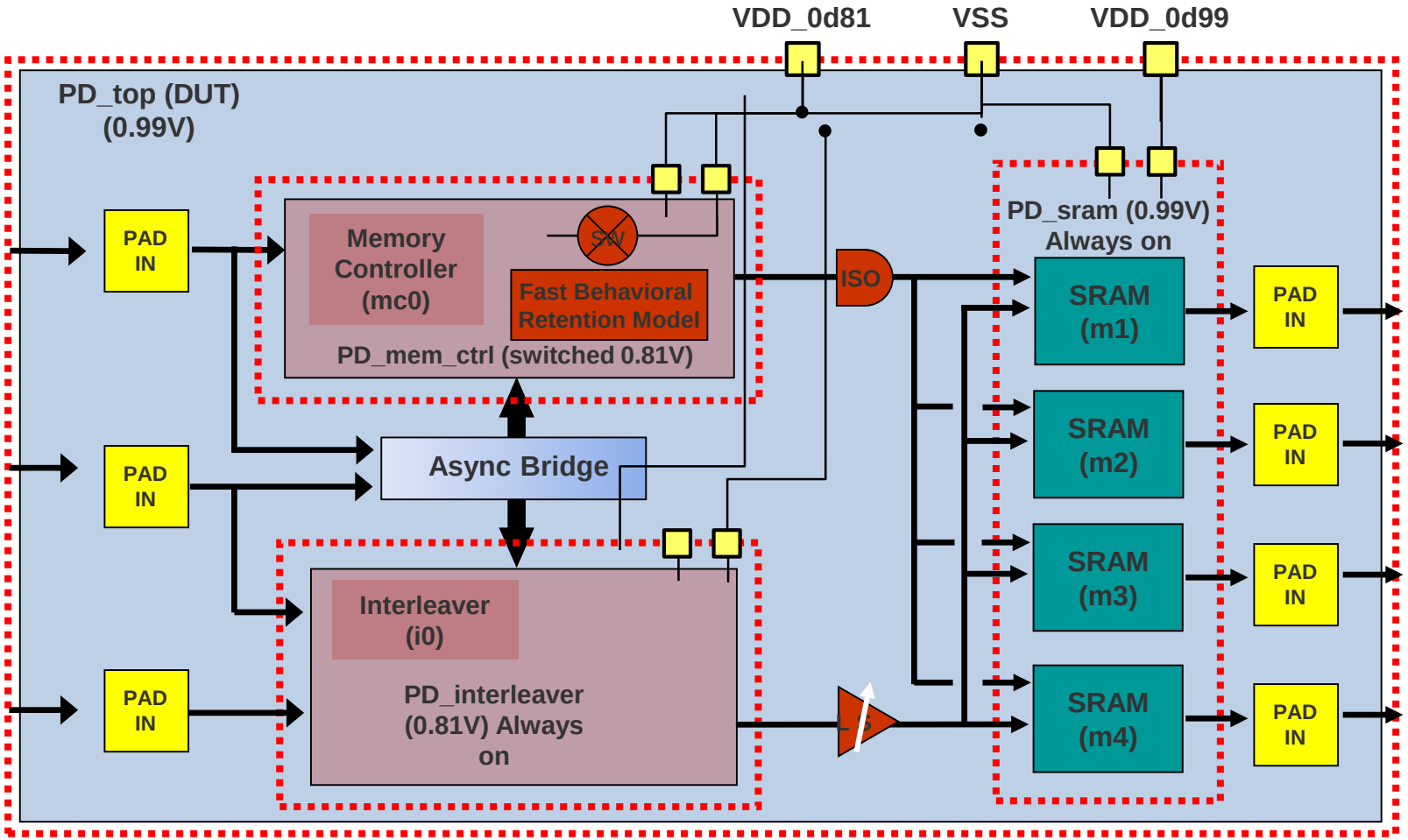
- RTL is augmented with UPF
  - To define power management architecture
- RTL + UPF verification
  - To ensure that power architecture completely supports planned power states of design
  - To ensure that design works correctly under power management
- RTL + UPF implementation
  - Synthesis, test insertion, place & route, etc.
  - UPF may be updated by user or tool
- NL + UPF verification
  - Power aware equivalence checking, static analysis, simulation, emulation, etc.



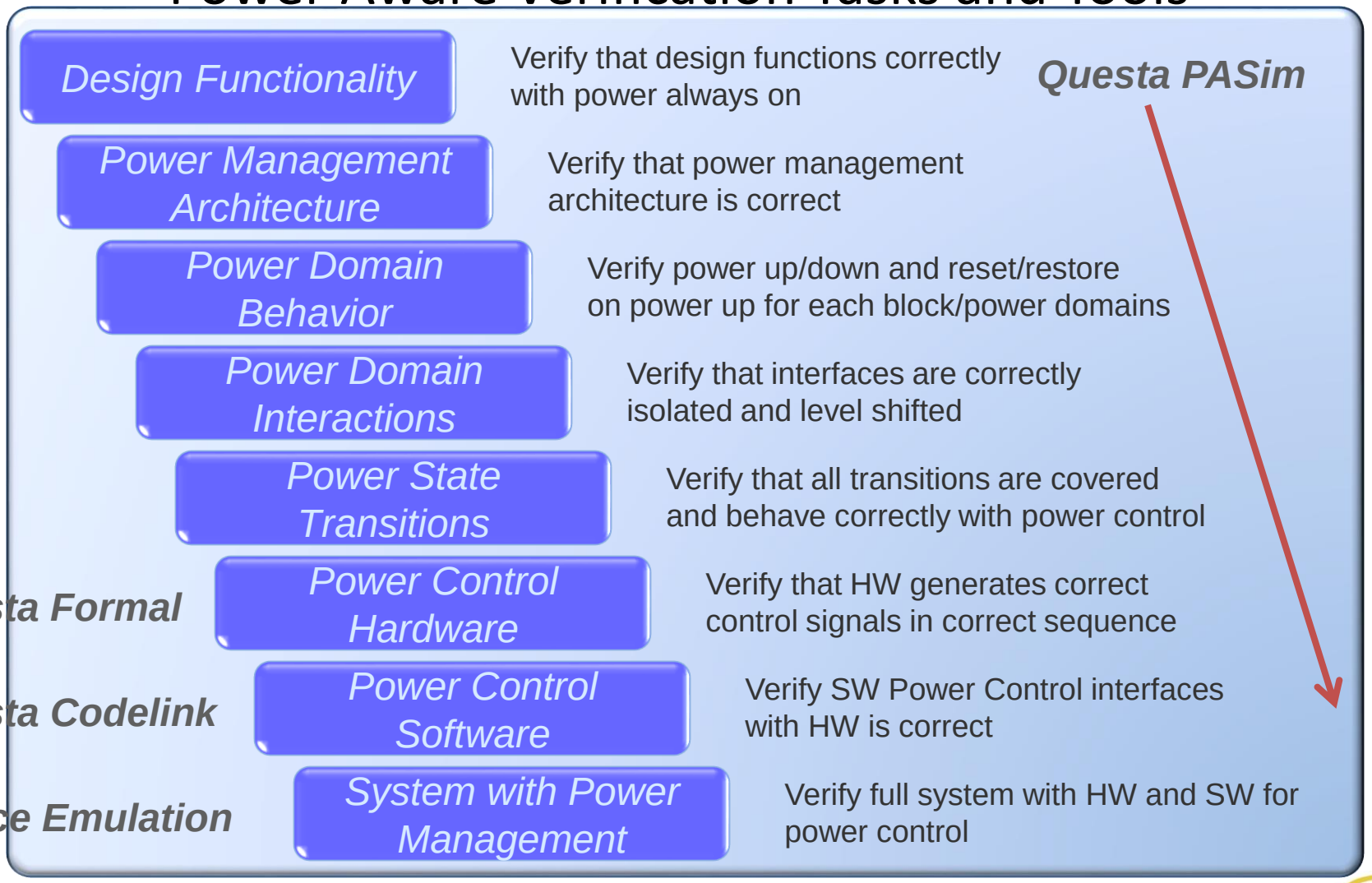
# Example Design



# Example Design With Power Intent



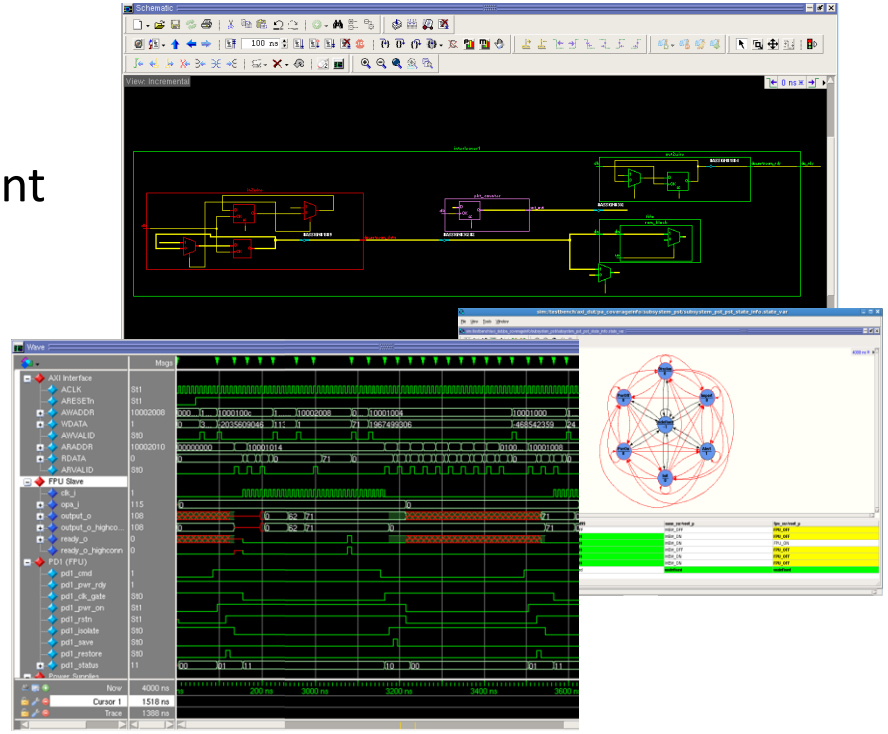
# Power Aware Verification Tasks and Tools



# Questa Power Aware Simulation

IEEE Std 1801™-2009

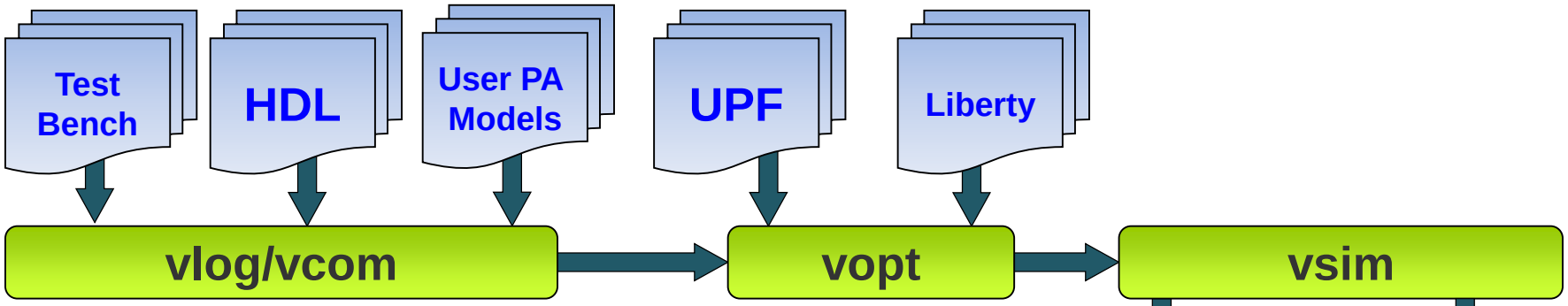
- Visualization of power management structure and behavior
- Automatic detection of power management errors
- Automatic power management coverage
- Automatic test plan generation
- High performance and efficiency
- Extensive IEEE 1801 UPF support



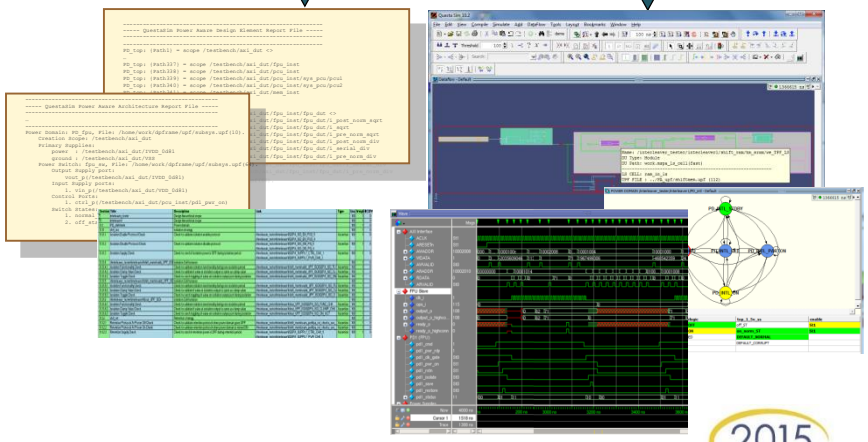
Enables power management architecture verification with RTL and Gate-Level power aware simulation

# Questa PASim Native Simulation of RTL+UPF

***Just a command-line switch on normal Questa simulation***



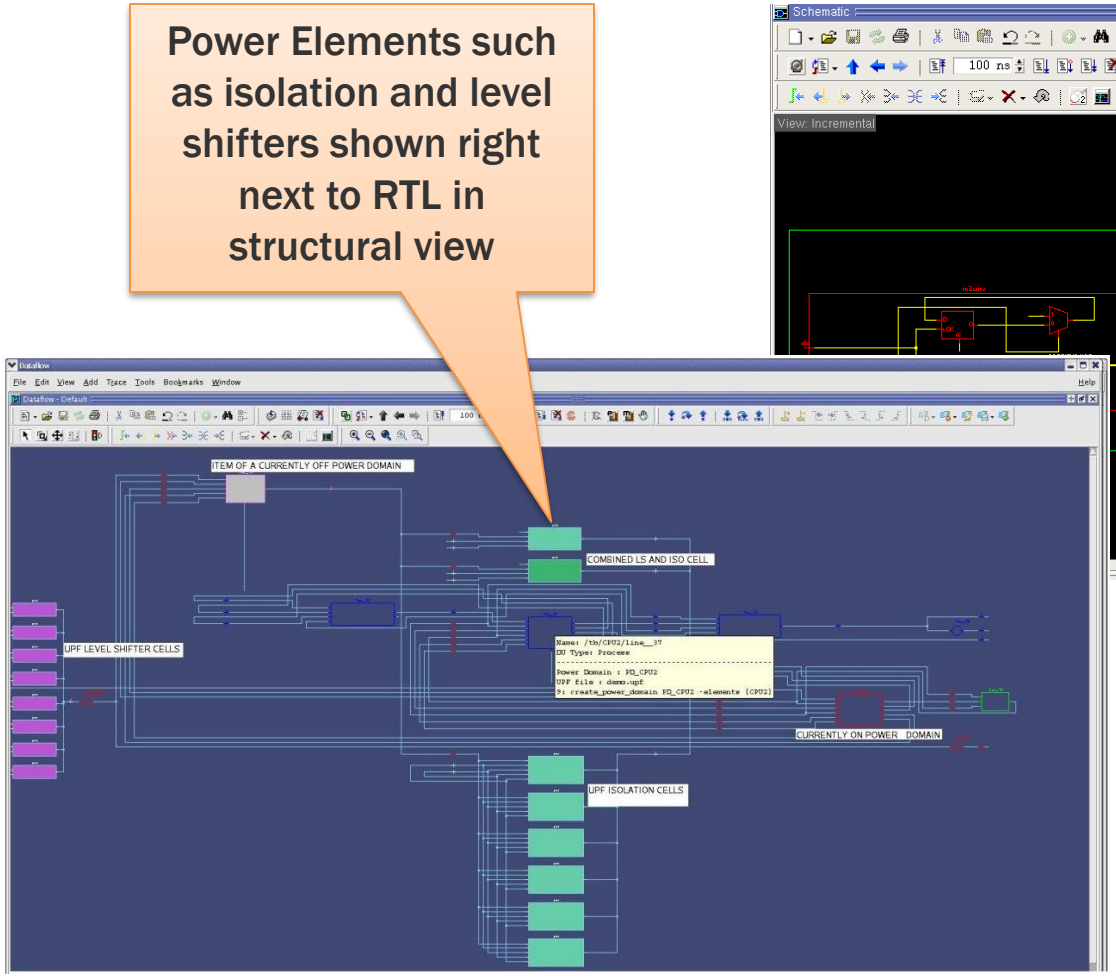

- Compile as usual
  - No change in source code
- Optimize with PA options
  - Process UPF power intent
  - Read Liberty libraries as needed
  - Run static PA checks
- Simulate with PA options
  - Generate reports/Testplan
  - Visualize/debug power-managed behavior

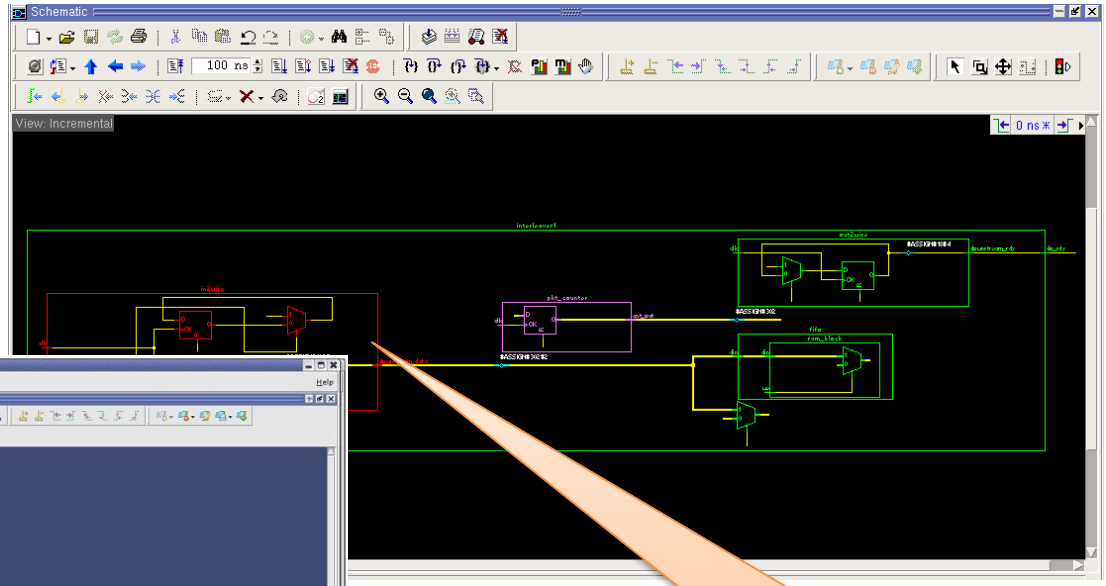


# Visualizing Power Domains and Structure

*See colorized domains and inserted power objects natively*

Power Elements such as isolation and level shifters shown right next to RTL in structural view



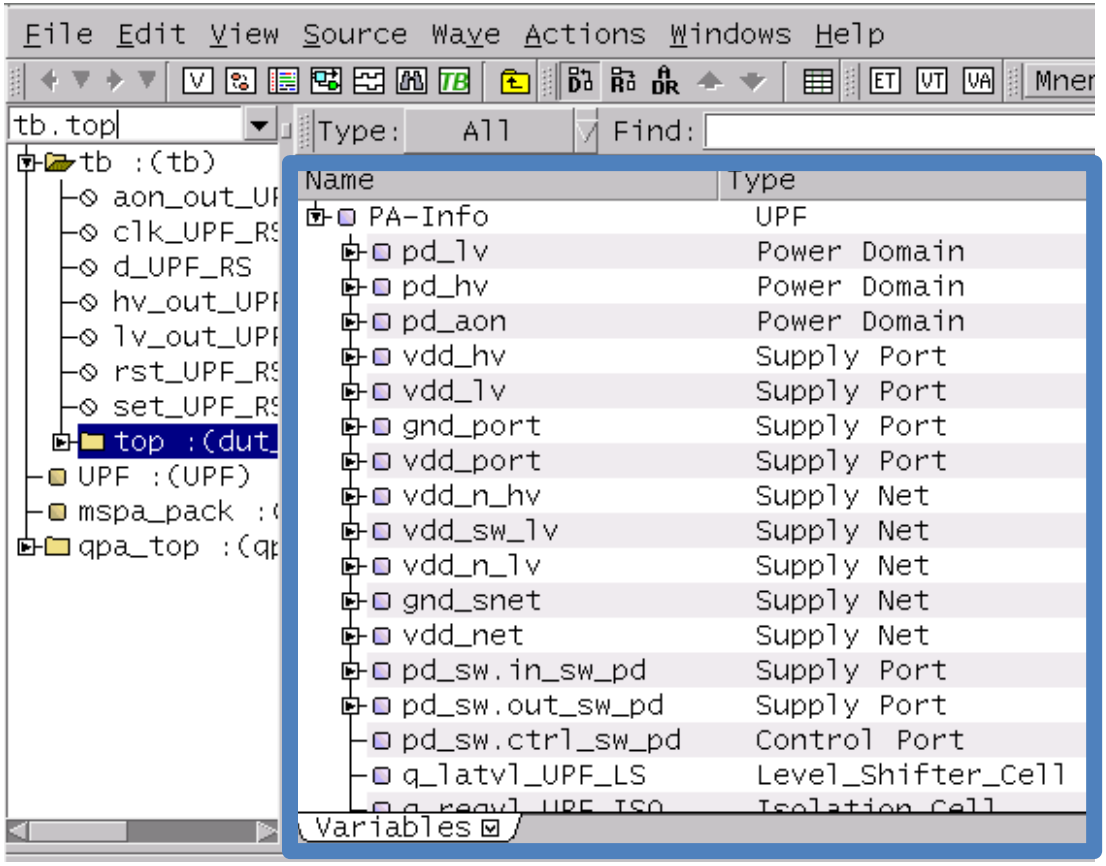




Power Domain and its instances is displayed in a unique color in structural view

# PA Debug - UPF Object Visualization

*Explore all UPF objects from one place*

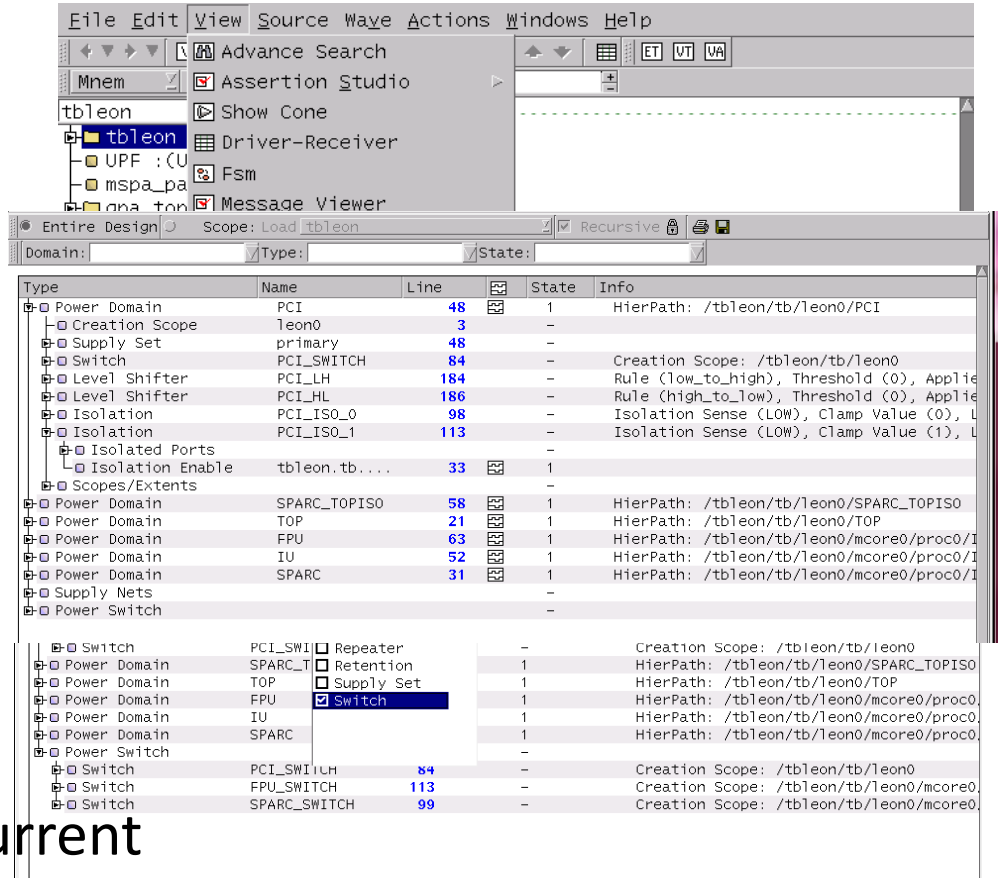
- View all PA objects in variable window
- Clubbed in PA Info
- All UPF objects
  - Power Domain
  - Supply Port
  - Supply Net
  - Supply Sets
  - Logic Nets/Ports
  - Add these to wave
  - Do cone analysis
- Inserted PA Cells



# Power Domains & UPF Object List

*Understand all objects associated with each Power Domain*

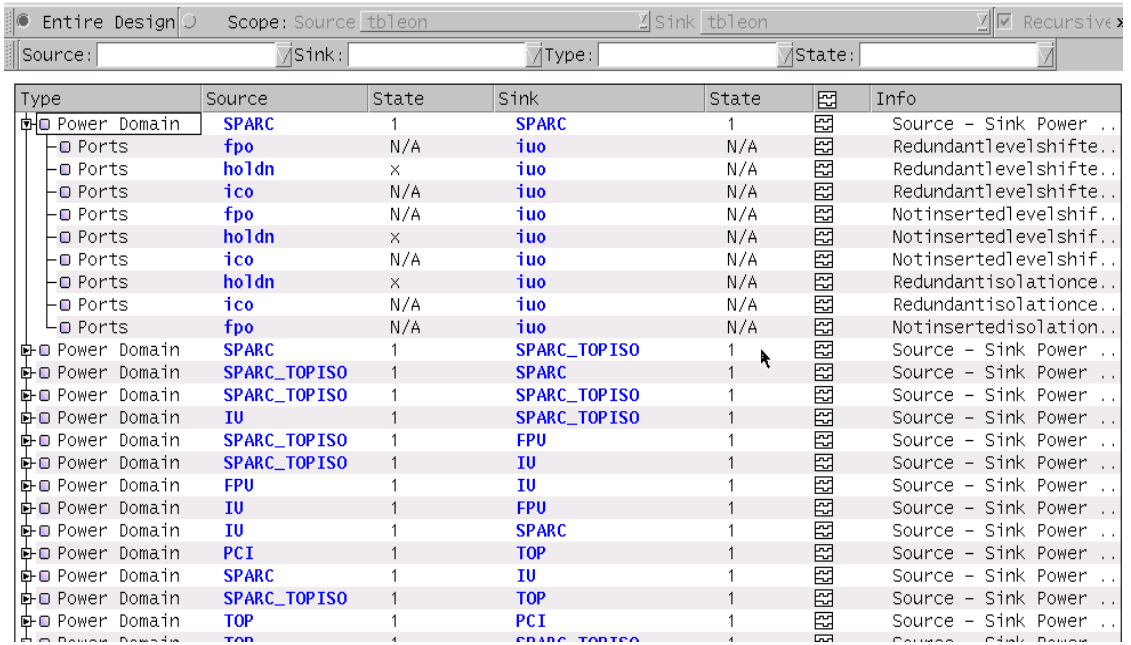
- View list of all Power Domains & UPF Objects
- Whole power management architecture visible
  - Primary Supplies, Power Management Strategies, Isolated/Level Shifted Ports …
- Option to add objects to wave
- View current state/simstate
- In sync with wave window
- Filtering of objects based on current scope/type/names



# Power Domains Crossings

## *Comprehending Power Domain Connectivity*

- View connected power domains
- List of all signals flowing from one domain to other
- Any violation/check in the source to sink path
- Filter/select source-sink domains
- Filtering based on violation type
- Add signals to waveform

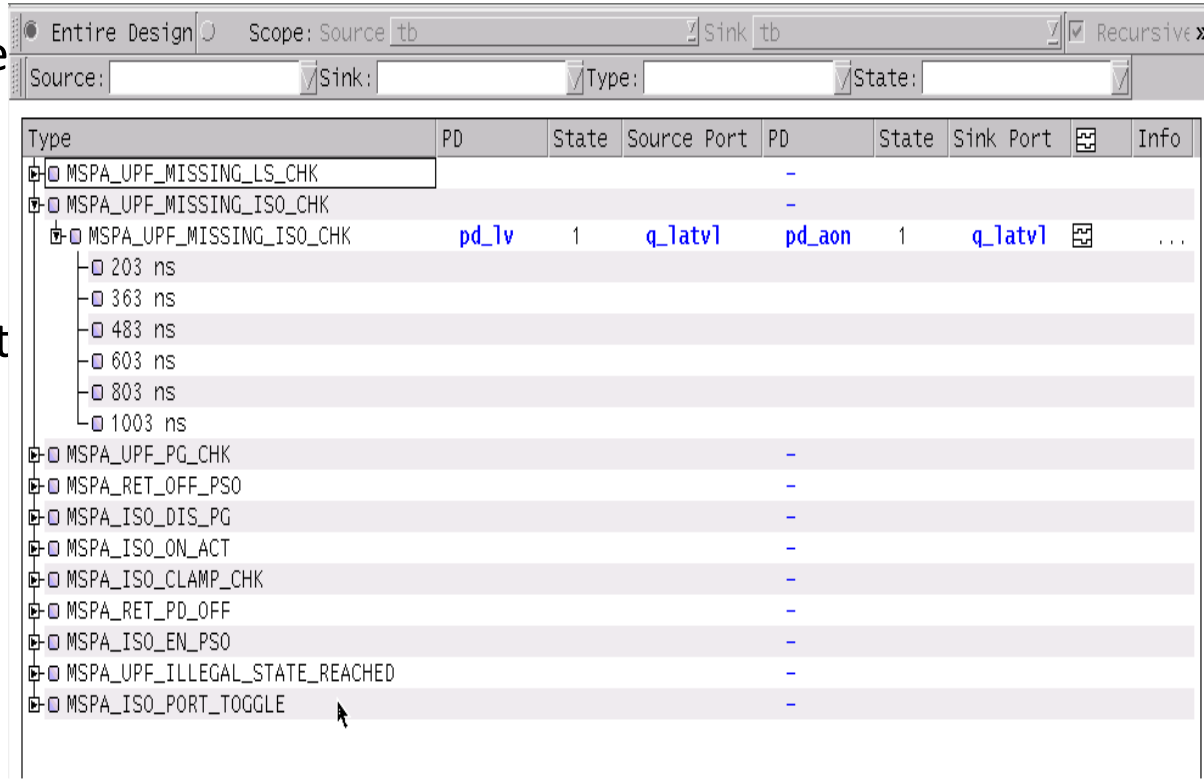



| Type | Source | State | Sink | State | Info |
|---|---|---|---|---|---|
| Power Domain | SPARC | 1 | SPARC | 1 | Source - Sink Power .. |
| Ports | fpo | N/A | iuo | N/A | Redundantlevelshifte.. |
| Ports | holdn | x | iuo | N/A | Redundantlevelshifte.. |
| Ports | ico | N/A | iuo | N/A | Redundantlevelshifte.. |
| Ports | fpo | N/A | iuo | N/A | Notinsertedlevelshif.. |
| Ports | holdn | x | iuo | N/A | Notinsertedlevelshif.. |
| Ports | ico | N/A | iuo | N/A | Notinsertedlevelshif.. |
| Ports | holdn | x | iuo | N/A | Redundantisolationce.. |
| Ports | ico | N/A | iuo | N/A | Redundantisolationce.. |
| Ports | fpo | N/A | iuo | N/A | Notinsertedisolation.. |
| Power Domain | SPARC | 1 | SPARC_TOPISO | 1 | Source - Sink Power .. |
| Power Domain | SPARC_TOPISO | 1 | SPARC | 1 | Source - Sink Power .. |
| Power Domain | SPARC_TOPISO | 1 | SPARC_TOPISO | 1 | Source - Sink Power .. |
| Power Domain | IU | 1 | SPARC_TOPISO | 1 | Source - Sink Power .. |
| Power Domain | SPARC_TOPISO | 1 | FPU | 1 | Source - Sink Power .. |
| Power Domain | SPARC_TOPISO | 1 | IU | 1 | Source - Sink Power .. |
| Power Domain | FPU | 1 | IU | 1 | Source - Sink Power .. |
| Power Domain | IU | 1 | FPU | 1 | Source - Sink Power .. |
| Power Domain | IU | 1 | SPARC | 1 | Source - Sink Power .. |
| Power Domain | PCI | 1 | TOP | 1 | Source - Sink Power .. |
| Power Domain | SPARC | 1 | IU | 1 | Source - Sink Power .. |
| Power Domain | SPARC_TOPISO | 1 | TOP | 1 | Source - Sink Power .. |
| Power Domain | TOP | 1 | PCI | 1 | Source - Sink Power .. |

# Power Aware Dynamic Messages

## *Interactively Debug Errors from Automatic Checks*

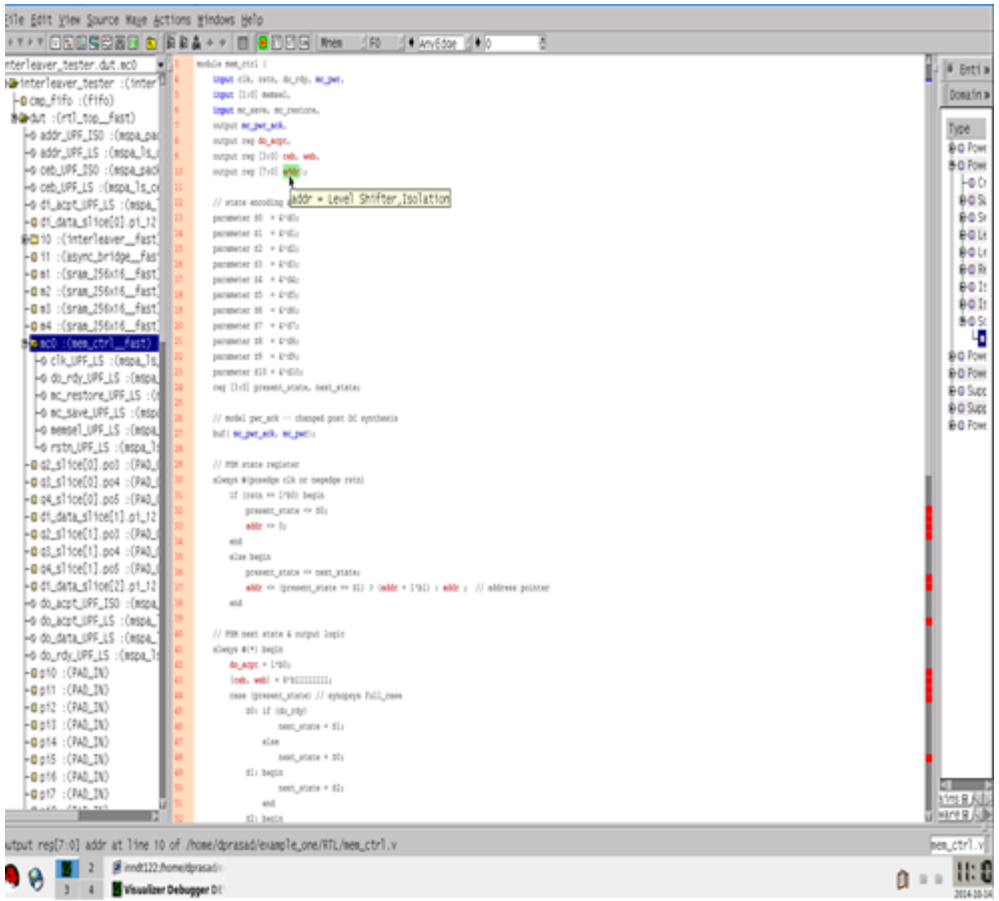
- View and debug all PA Dynamic Messages
- View time at which message
  - Synced with waveforms
- Add signals/domains to waveform
- Filtering based on message t



# HDL Annotation

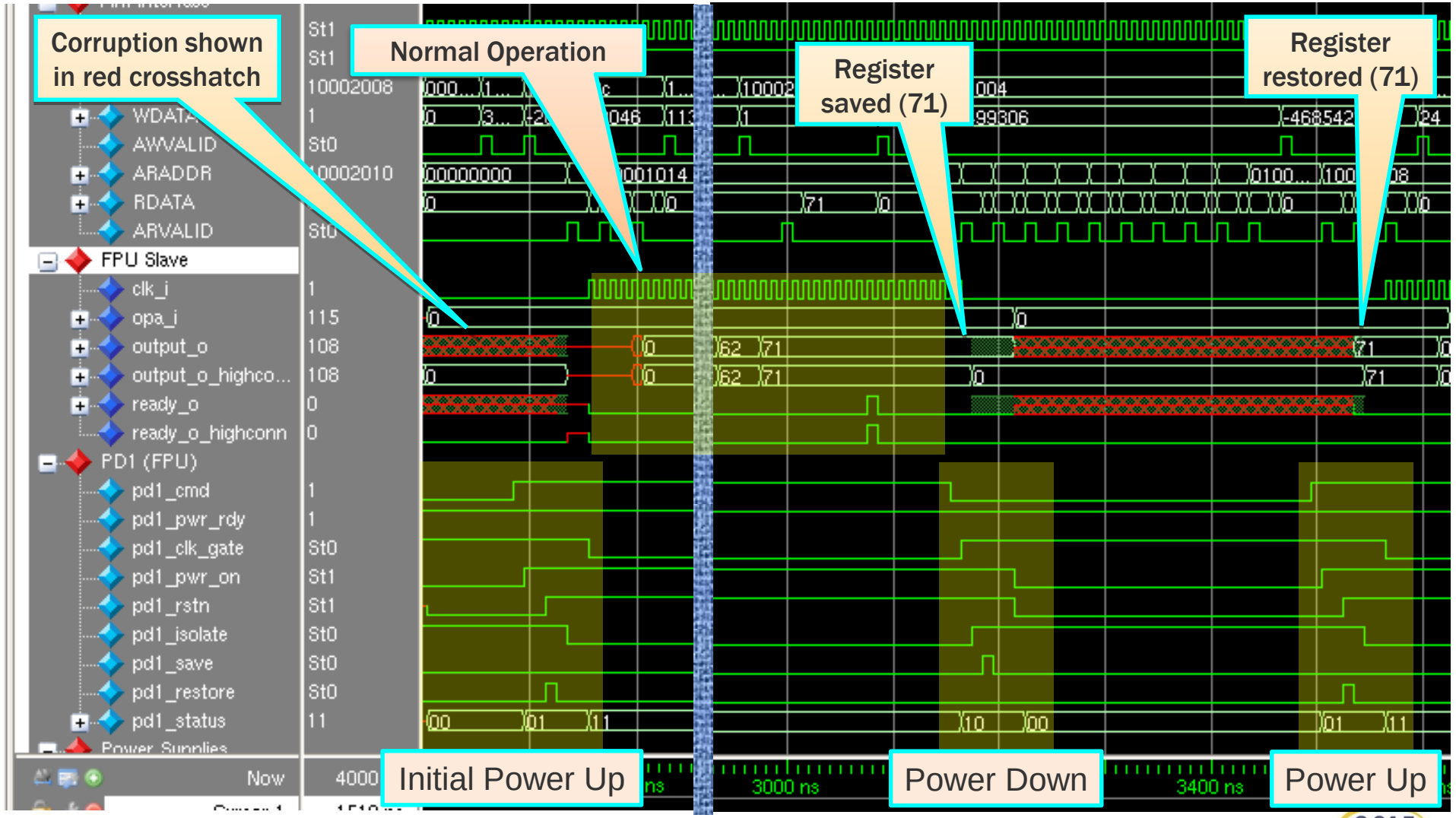
*View all UPF added information in your HDL Source Context*

- HDL signal coloring
  - Isolation(Blue)
  - Level Shifter(Blue)
  - Buffer(Blue)
  - Corruption(red)
- Hovering information
  - Level Shifter
  - Isolation
  - Buffer
  - Corruption



# Visualizing Power Domain Behavior in waveform

*Special patterns for corruption and isolation*



# Questa PASim Automatic Checks

*Automatically detect Power Management Errors*

**Power Architecture Checks:**

Isolation cells

- Isolation cells are inserted where required by power state definitions
- Isolation cells correctly handle dynamic signal behavior

Level shifters

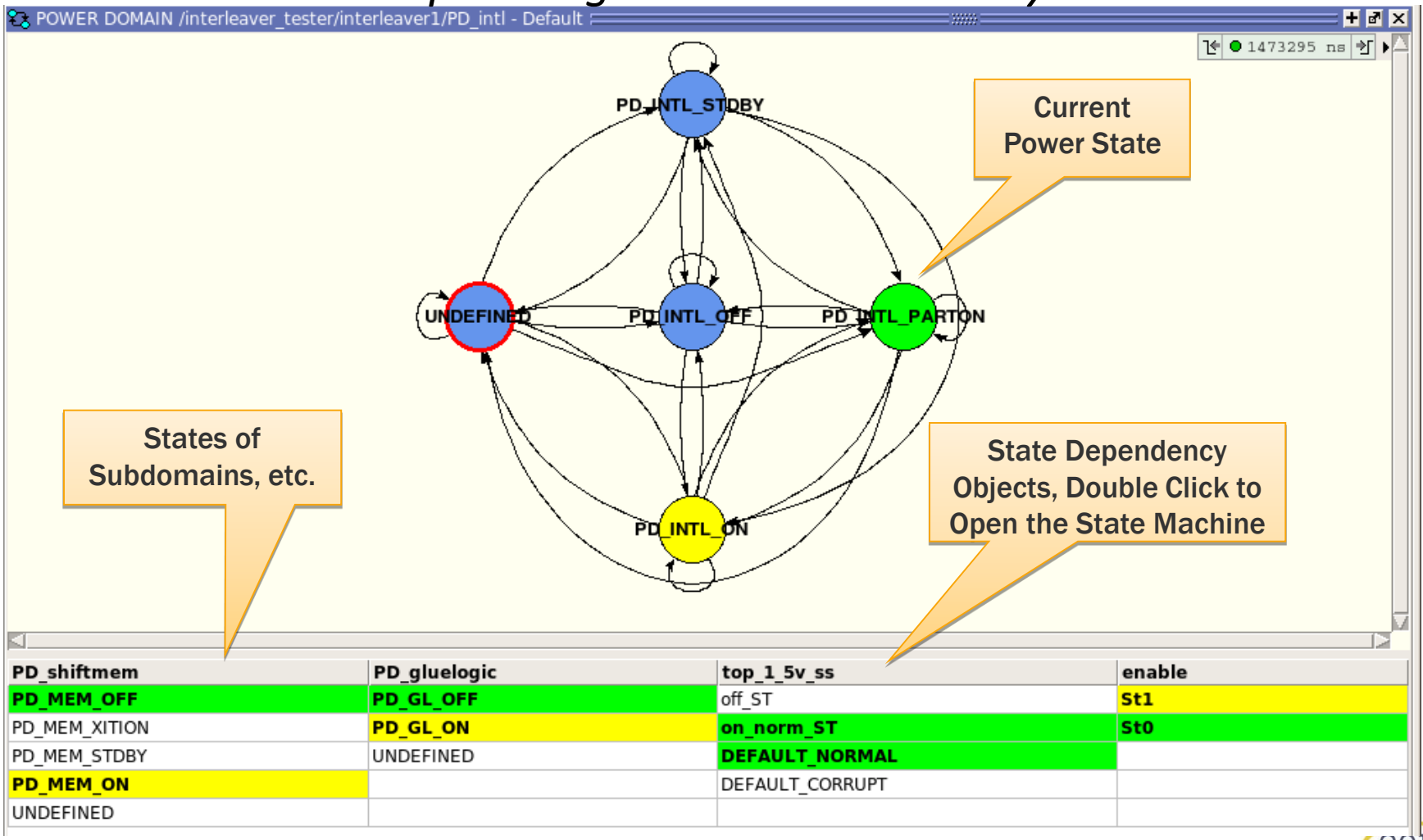
- Level shifters are inserted where required by power state definitions
- Level shifters shift in correct direction
- Level shifters correctly handle dynamic signal behavior

**Power Control Sequence Checks:**

- Clock is disabled during power down
- Isolation is enabled during power down
- Inputs do not toggle during power down
- Retention/Isolation power is on and stable during power down
- Retention registers are saved before power down
- Latch enable is correctly set when retention occurs
- Primary power is on and stable during power up
- Non-retention registers are reset at power up
- Power control signals do not glitch

# Visualize Power States and Transitions

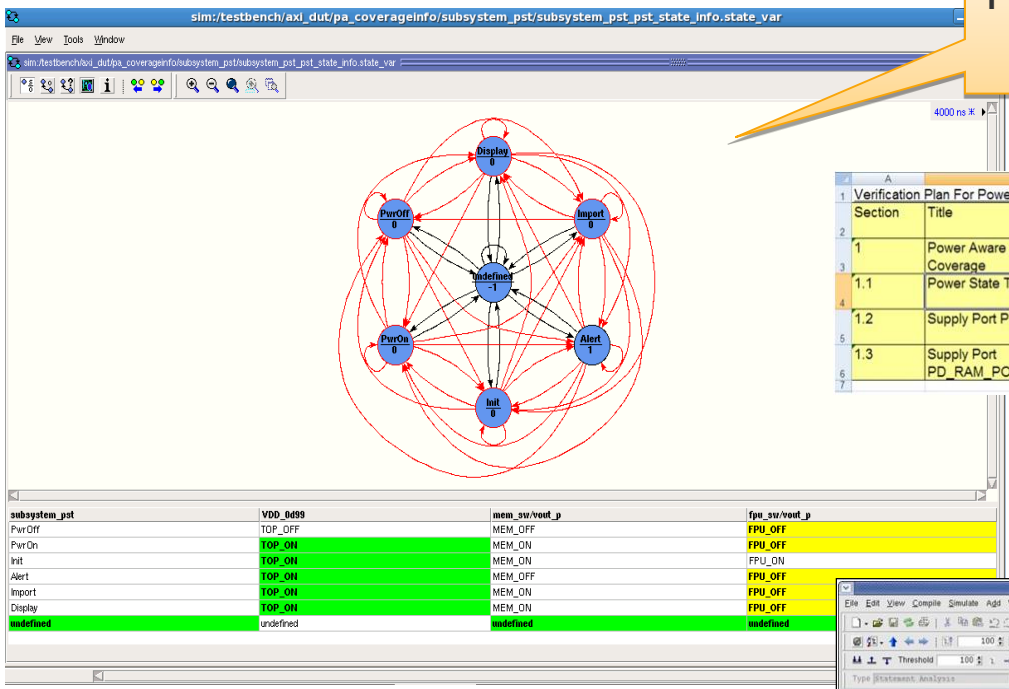
*Step through states interactively*




| PD_shiftmem | PD_gluelogic | top_1_5v_ss | enable |
|---|---|---|---|
| **PD_MEM_OFF** | **PD_GL_OFF** | off_ST | **St1** |
| PD_MEM_XITION | **PD_GL_ON** | **on_norm_ST** | **St0** |
| PD_MEM_STDBY | UNDEFINED | **DEFAULT_NORMAL** | |
| **PD_MEM_ON** | | DEFAULT_CORRUPT | |
| UNDEFINED | | | |

# Automatic Power State Coverage

*Coverage Information Created Directly from UPF Power States*

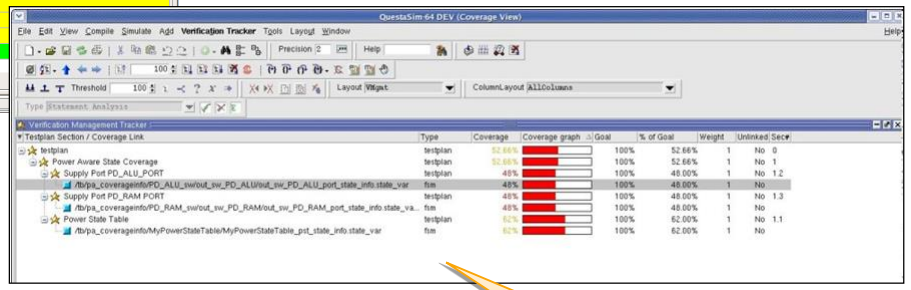



Power states and transitions ...

Verification Plan For Power Aware State Coverage

| Section | Title | Description | Link | Type | Weight | Goal |
|---|---|---|---|---|---|---|
| 1 | Power Aware State Coverage | Coverage plan for PA State machines | | | | |
| 1.1 | Power State Table | State coverage data for PST PASM | /tb/pa_coverageinfo/MyPowerState Table | FSM | 1 | 100 |
| 1.2 | Supply Port PD_ALU_PORT | State coverage data for supply port PASM | /tb/pa_coverageinfo/PD_ALU_sw/out_sw_PD_ALU | FSM | 1 | 100 |
| 1.3 | Supply Port PD_RAM_PORT | State coverage data for supply port PASM | /tb/pa_coverageinfo/PD_RAM_sw/out_sw_PD_RAM | FSM | 1 | 100 |

to measure power state coverage ...




UCDB

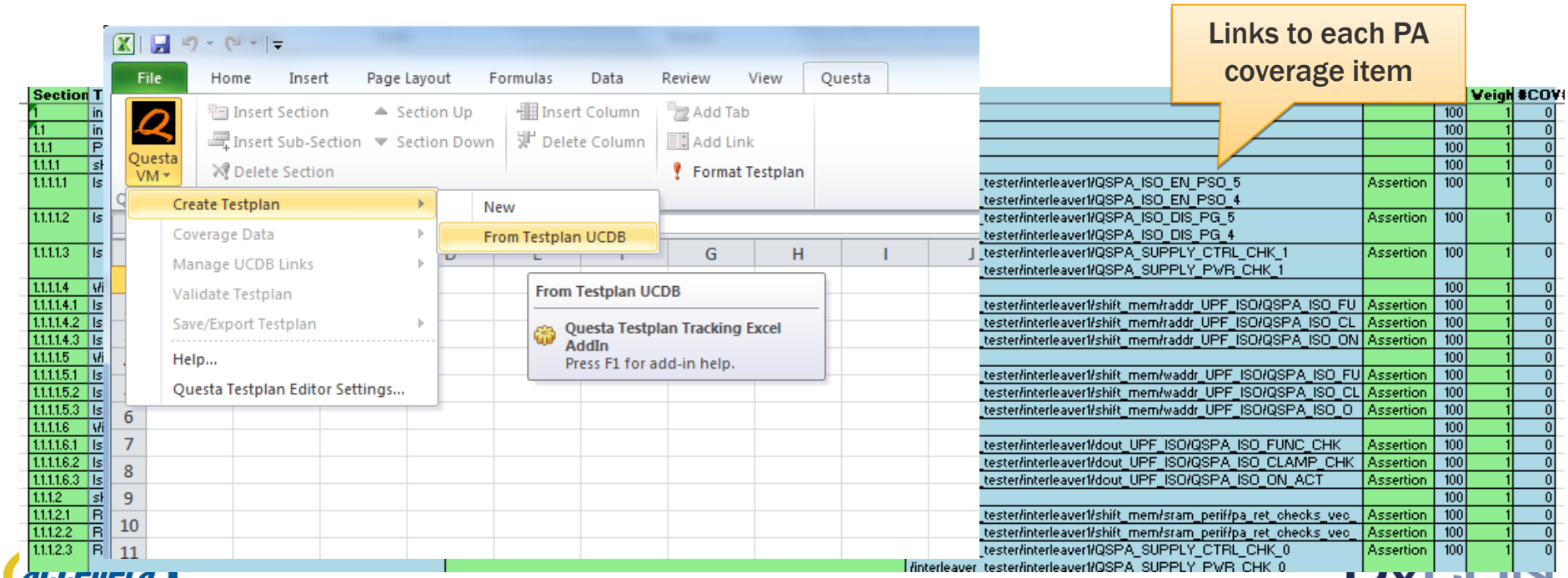
are stored in the UCDB ...

... for coverage closure.

# PA Testplan Generation

*Automatic Flow for Low Power Coverage Closure*

- Vopt automatically generates QuestaPowerAwareTestplan.ucdb
- Includes both power states and dynamic tool generated PA assertions
- Coverage data stored in UCDB can be merged with testplan UCDB
- Uses Excel add-in to generate XML testplan from UCDB



# Low Power Coverage Closure Report

- Tracks power related coverage data
  - Links PA testplan items with PA coverage items in ucdb
  - Used standa in top tplan

Track coverage of PA State and Transition

Track coverage of PA enable assertion passes

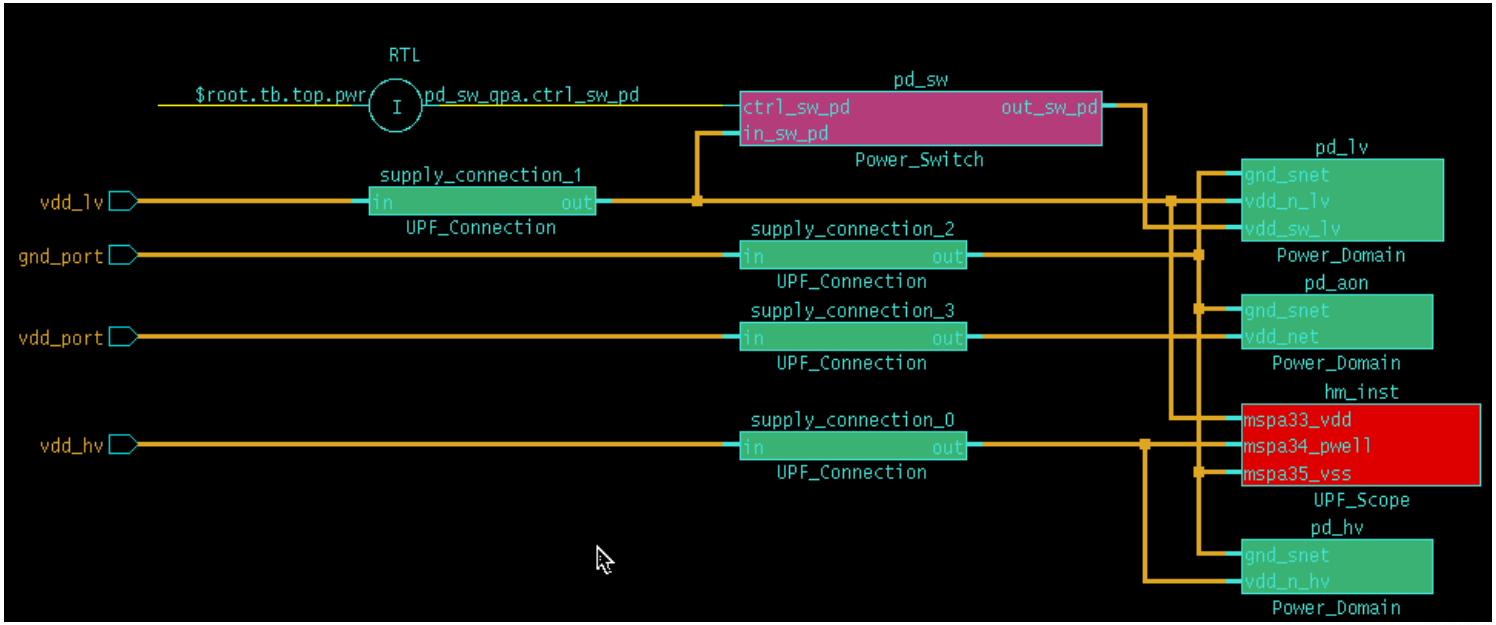
Verification Management Tracker

| Sec# | Testplan Section / Coverage Link | Type | Coverage graph | Goal | % of Goal | Weight | Unlinked |
|---|---|---|---|---|---|---|---|
| 0 | testplan | testplan | 7% | - | 100% | 1 | No |
| 1 | interleaver_tester | testplan | .37% | 1% | 100% | 100 | No |
| 1.1 | interleaver1 | testplan | 63.37% | 1% | 100% | 100 | No |
| 1.1.1 | PD_shiftmem | testplan | 61.11% | 1% | 100% | 100 | No |
| 1.1.1.1 | shft_iso | testplan | 72.22% | 1% | 100% | 100 | No |
| 1.1.1.2 | shft_ret | testplan | 50% | 1% | 100% | 100 | No |
| 1.1.1.2.1 | Retention Supply Chec | testpla | 50% | 1% | 100% | 100 | No |
| | /interleaver_teste erleaver1/pa_upf_ret_supply_check_1/mspa_supply_chk_1 | assert | 100% | 100% | 100% | 1 | No |
| | /interleaver_te /interleaver1/pa_upf_ret_supply_check_1/mspa_supply_chk_2 | assert | 0% | 100% | 0% | 1 | No |
| 1.1.2 | PD_ram | testplan | 66.66% | 1% | 100% | 100 | No |
| 1.1.3 | PD_intl | testplan | 57.5% | 1% | 100% | 100 | No |
| 1.1.3.1 | Power States | testplan | 72.5% | 1% | 100% | 100 | No |
| 1.1.3.1.1 | States | testplan | 100% | 1% | 100% | 100 | No |
| | /interleaver_tester/interleaver1/pa_coverageinfo/PD_intl/PD_intl_STATE_COVERAGE/intl_off | coverpoint | 100% | 100% | 100% | 1 | No |
| | /interleaver_tester/interleaver1/pa_coverageinfo/PD_intl/PD_intl_STATE_COVERAGE/intl_parton | coverpoint | 100% | 100% | 100% | 1 | No |
| | /interleaver_tester/interleaver1/pa_coverageinfo/PD_intl/PD_intl_STATE_COVERAGE/intl_standby | coverpoint | 100% | 100% | 100% | 1 | No |
| | /interleaver_tester/interleaver1/pa_coverageinfo/PD_intl/PD_intl_STATE_COVERAGE/intl_on | coverpoint | 100% | 100% | 100% | 1 | No |
| 1.1.3.1.2 | Transitions | testplan | 45% | 1% | 100% | 100 | No |
| | /interleaver_tester/interleaver1/pa_coverageinfo/PD_intl/PD_intl_TRANS_COVERAGE/PD_intl_TRA... | coverpoint | 45% | 100% | 45% | 1 | No |
| 1.1.3.2 | Missing Level Shifter Check | testplan | 0% | 1% | 0% | 100 | No |
| | /interleaver_tester/interleaver1/pa_missing_ls_check_5/mspa_upf_missing_ls_chk | assert | 0% | 100% | 0% | 1 | No |
| 1.1.3.3 | Missing Isolation Check | testplan | 100% | 1% | 100% | 100 | No |
| | /interleaver_tester/interleaver1/pa_missing_iso_check_6/mspa_upf_missing_iso_chk | assert | 100% | 100% | 100% | 1 | No |
| | /interleaver_tester/interleaver1/pa_missing_iso_check_7/mspa_upf_missing_iso_chk | assert | 100% | 100% | 100% | 1 | No |

# PA Debug – Supply Network Debug

*Visualizer Explore the Supply Network*

- Explore the power connectivity and power intent of the design (Visualize UPF)
  - Power Domains, Switches, PA Strategies, Supply Ports/Nets, Logic Ports/Nets …
  - View state & voltage values of UPF supplies
  - View simstate of power domain
- Debug supply network created in UPF
  - Do cone analysis (and time cone) on any UPF objects
  - Trace back and forth on UPF supplies, trace UPF to HDL connections



# Summary

- The UPF is the leading industry standard defines Power Management Architecture independent from RTL and physical design
- Debugging a Low Power/UPF enabled design requires visualization, exploration, checking and coverage tightly integrated with the HDL/gates
- Mentor provides the most comprehensive native low power debugging environment for UPF

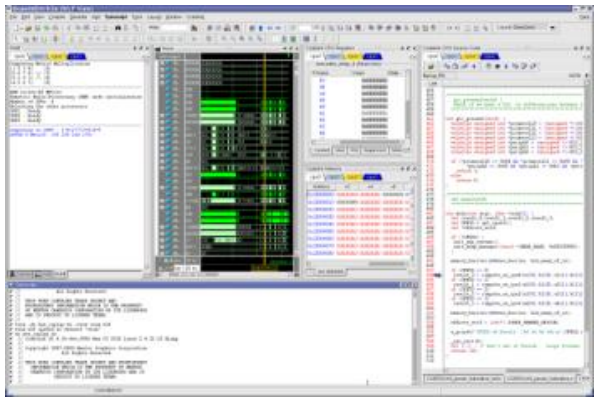
# Agenda

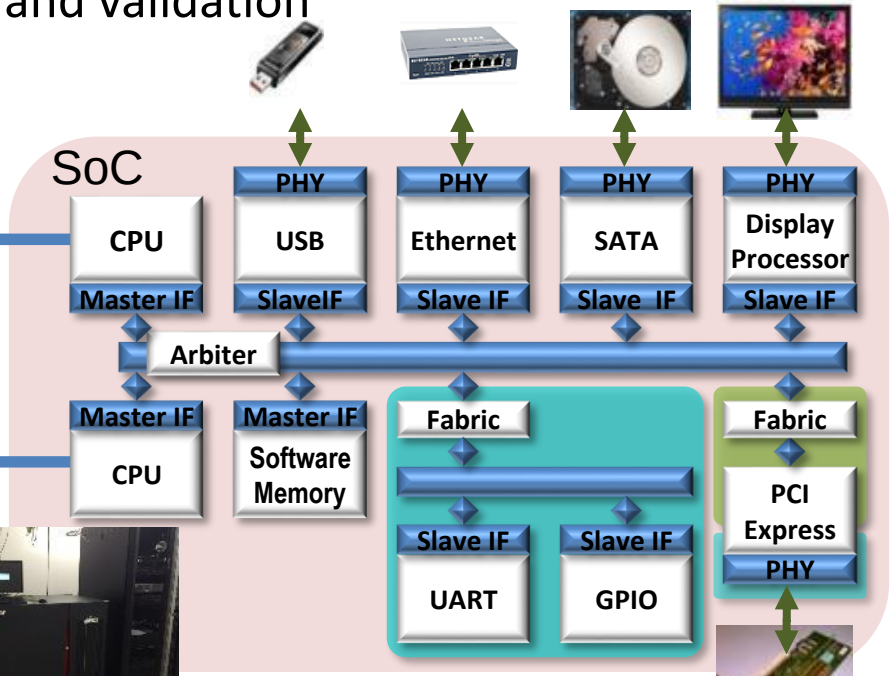
- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

# Veloce2 for Full-Chip Verification and Validation Boot OS, Drivers Validation

- Co-Emulation for UVM/SystemC testbench Acceleration
- OS and device drivers run on RTL or ISS processor models
- VirtuaLAB and physical peripherals exercise interfaces
- Golden Model for silicon bring-up and validation

Multi-core SW/HW debug





# Veloce OS3 Fusion ICE and Virtual Within One Solution

Veloce2+ OS3

In Circuit Emulation (ICE)

**The Best of Both Worlds**

Virtual Peripheral & Transactors

**A Door to real traffic**
- No testbench creation
- Near real speed emulation

Quattro

Maximus

**TBX the pathway to a spectrum of verification possibilities**
- Predictable/repeatable behavior
- No speed compromise

**Versatile and Easy to Use Verification and Validation Solution**

accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE



53

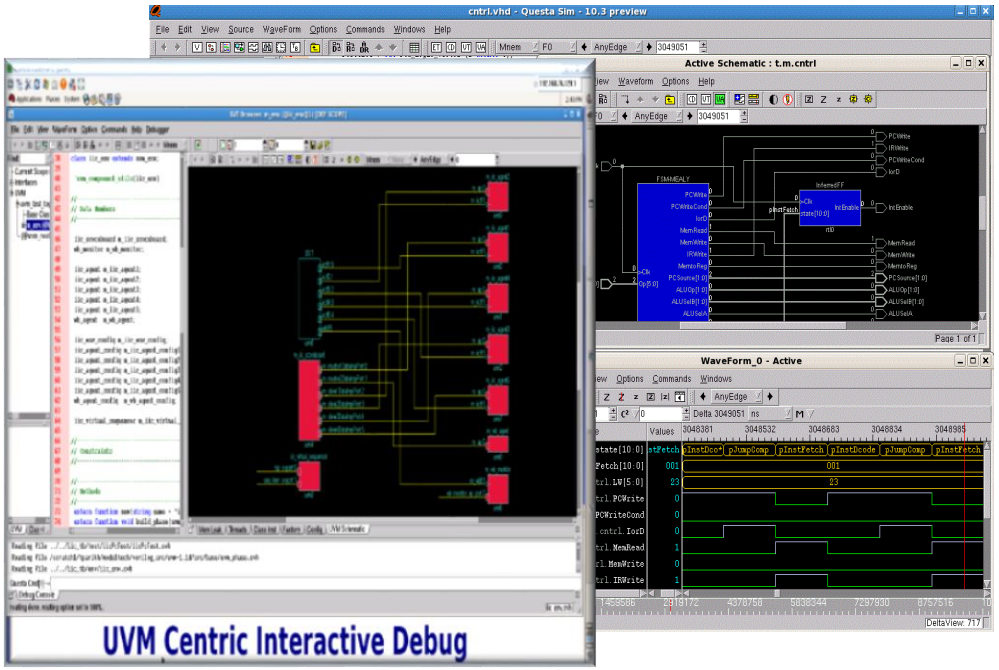
# Advanced Methodologies – UVM/ SV


- Advanced methodologies enable verification productivity
  - Faster to develop reusable application level testbenches
  - Reuse from block to system level and across the teams/projects
  - Single testbench for simulation and emulation

# Unified Debug for Questa and Veloce

*Maximize performance and still have ease of use*

Questa®

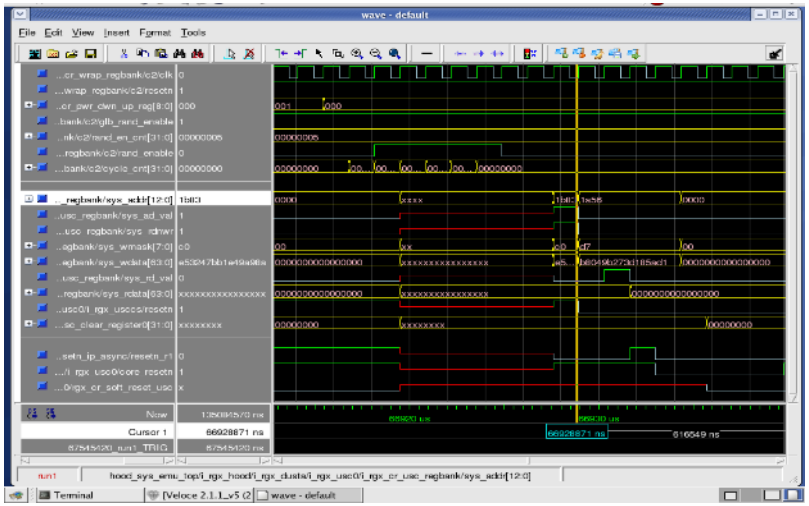



Veloce2

- Full signal visibility in the hardware offers fast TAT
- Same debugger for the whole simulation and emulation environment
- On-demand data loading for fast response with huge data

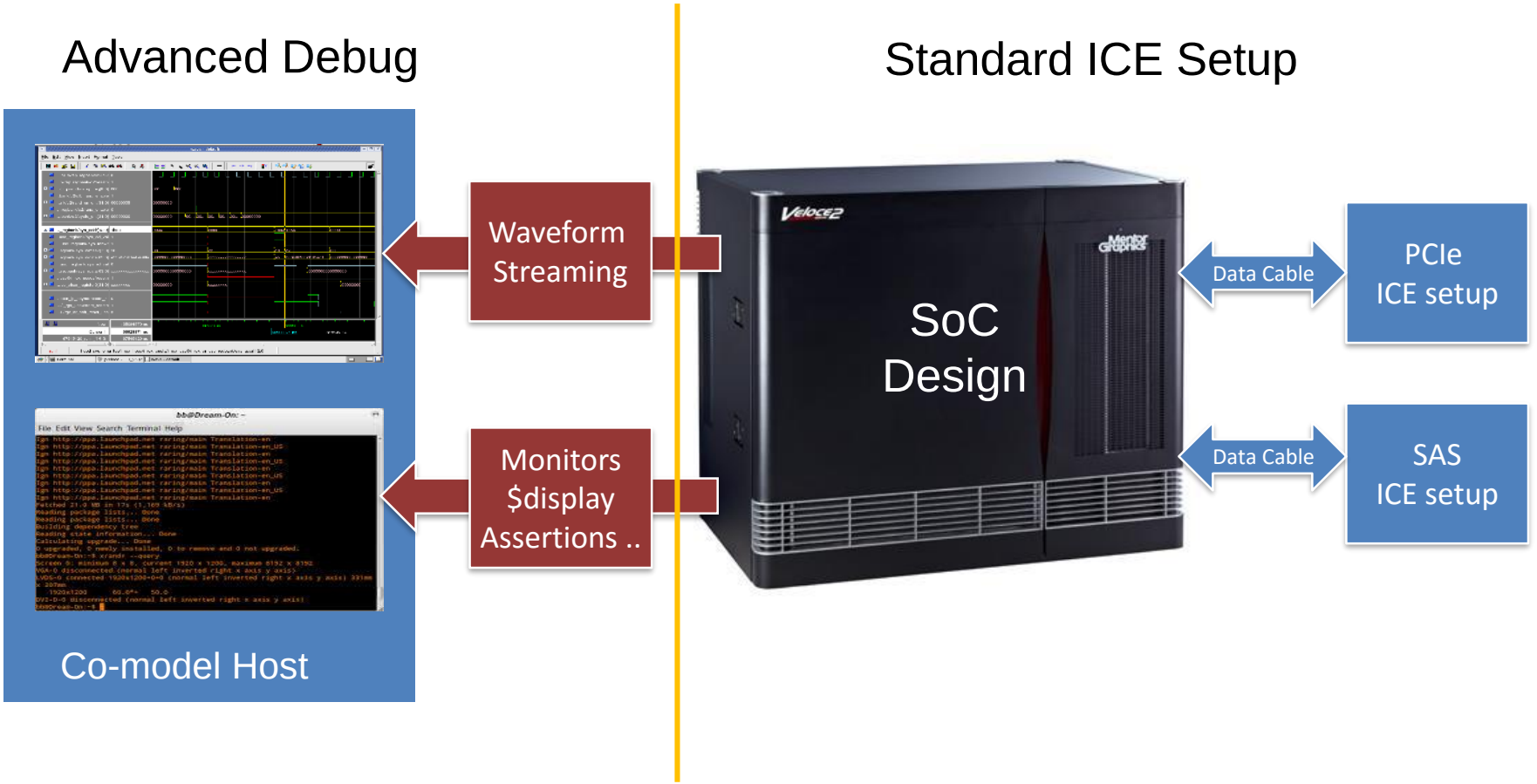
# On the Fly Waveform Streaming For Fast Bug Identification

- Waveform generation of key signals/ interfaces for entire emulation run to identify problem area
- Monitor key interfaces for long emulation runs
- Find the failure time point very quickly with fewer signal

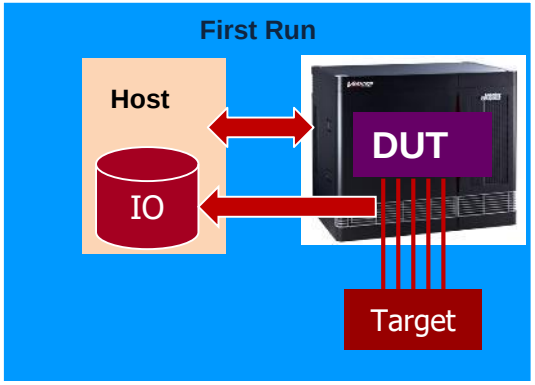



TBX Streaming

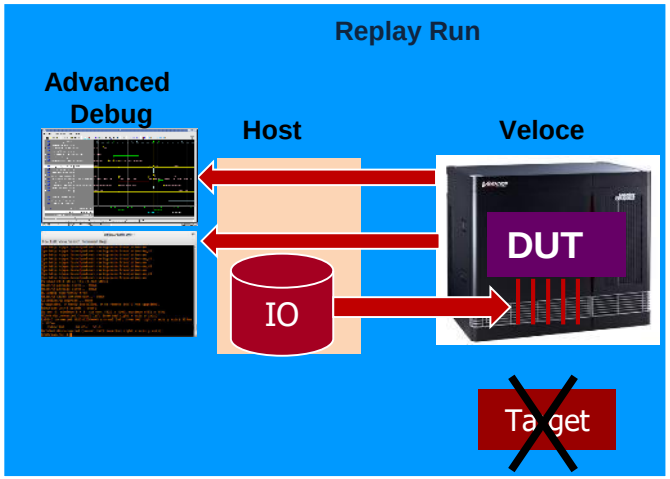
# Veloce Advanced Debug for ICE



# Replay Dynamic ICE Environment Real-Time TBX




- Run emulation with ICE target connected
- Captures IO activity (stimulus) of the target on disk




- Replay the captured stimulus
  - ICE Targets not needed
  - Repeatable behavior
- Use TBX capabilities
  - Assertion and coverage
  - Monitors and $display
  - Interval replay for productive debug

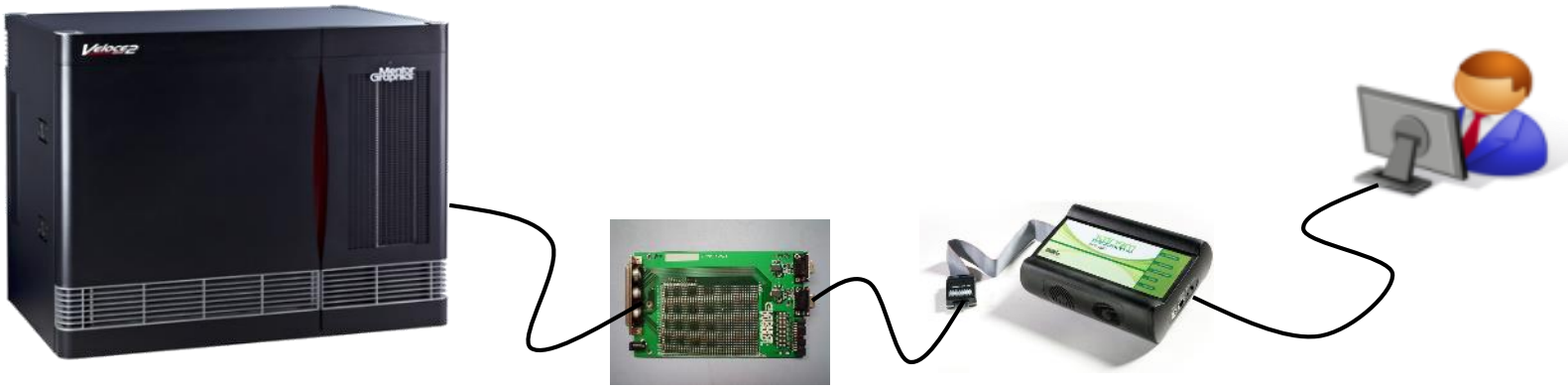
**Reproduce non deterministic Scenarios**
**Power Analysis with ICE environment**

# Goals of New HW Debug Paradigm

- Full signal visibility built on hardware accelerates turn around time
- Augmented ICE debug combining transactor with ICE
- Homogeneous debug solution from simulation to emulation

# Verification Demands New SW Debug Methods

- JTAG reasonably productive at 1 GHz
- Painfully slow at simulation and emulation speeds



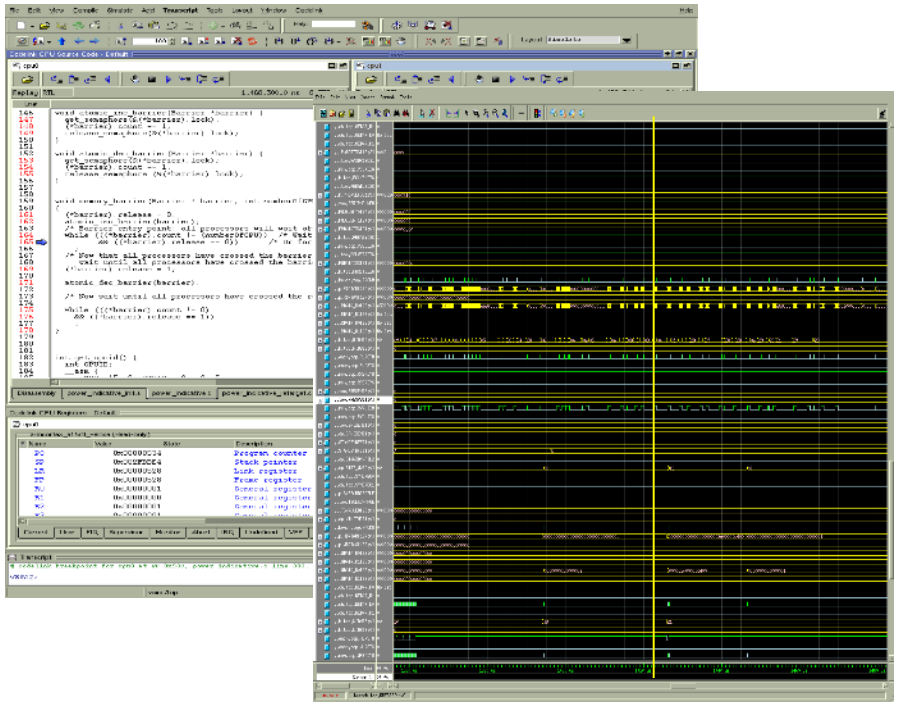
Emulator

JTAG Speed Adaptor

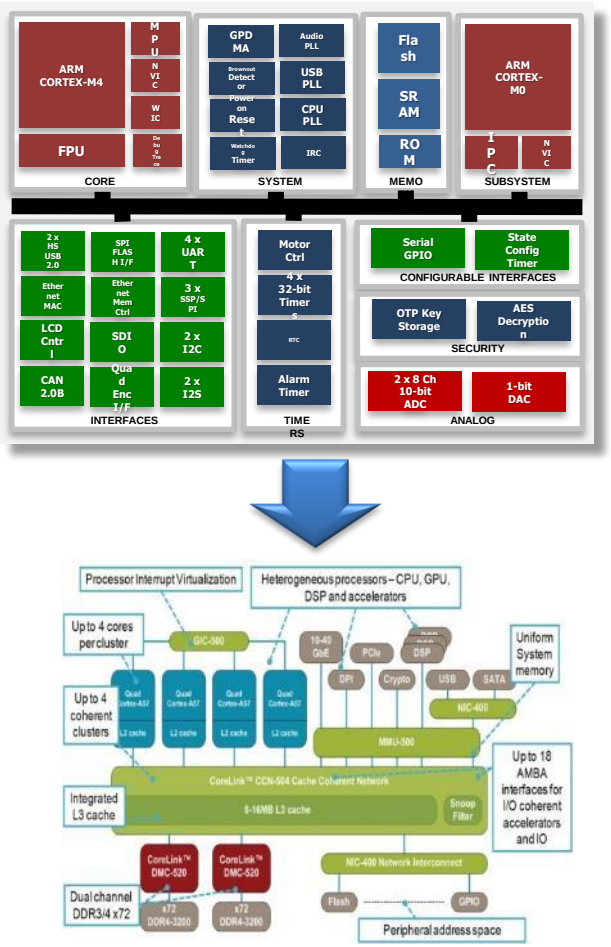
JTAG Probe

# Veloce New Software Debug Solutions

- Virtual Probes
  - Traditional interactive debug using JTAG transactors
  - No physical JTAG probe needed – all virtual
  - Full software debug capabilities
- Codelink
  - Off-line post emulation debug
  - Enables efficient sharing of emulation resource
  - Full software debug capabilities
- Warpcore
  - Integrates Virtual Machine with Veloce
  - Enables 100 MIPS+ performance of software execution
  - Full software debug capabilities

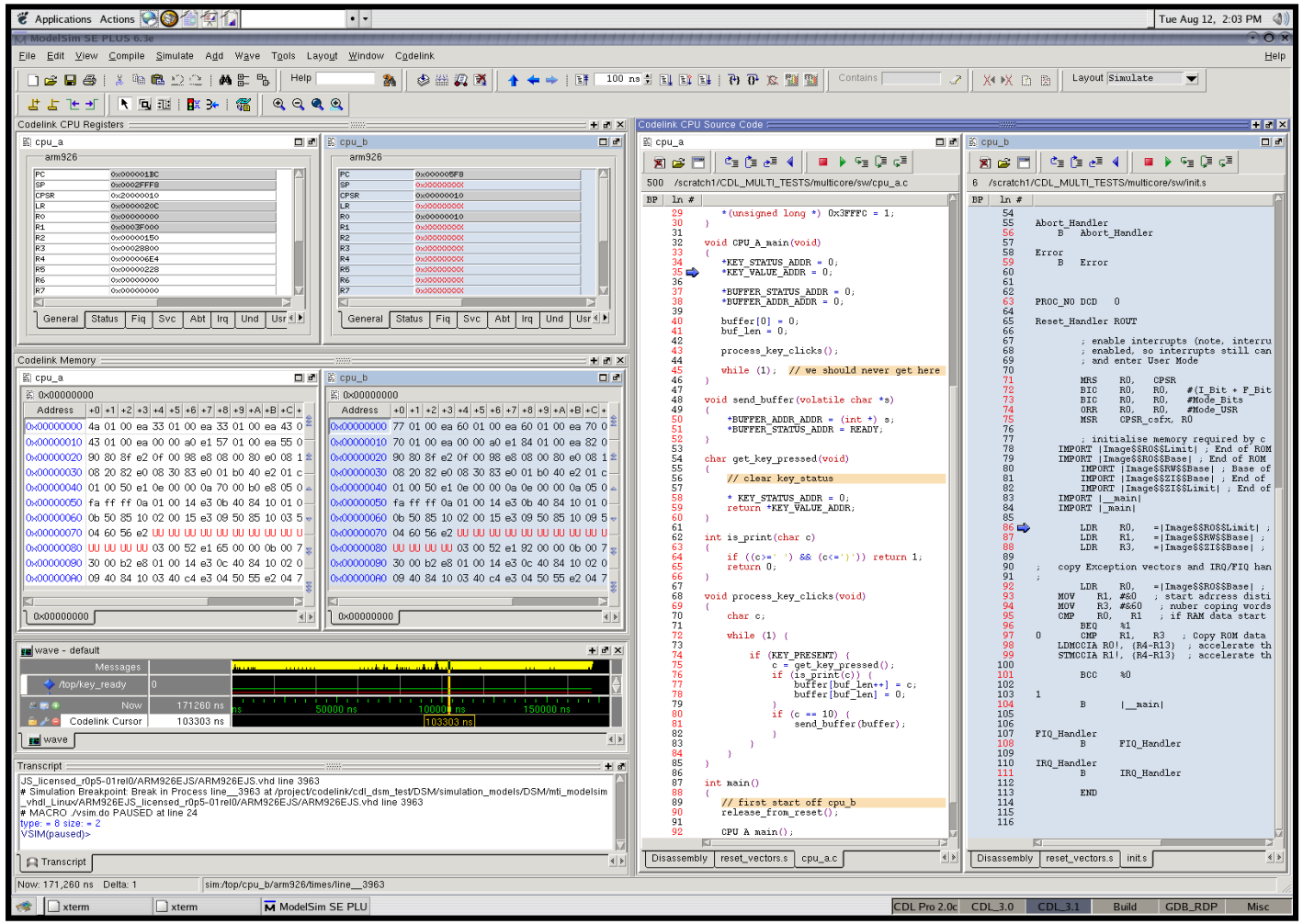


# Every Design is Now Multi-Core



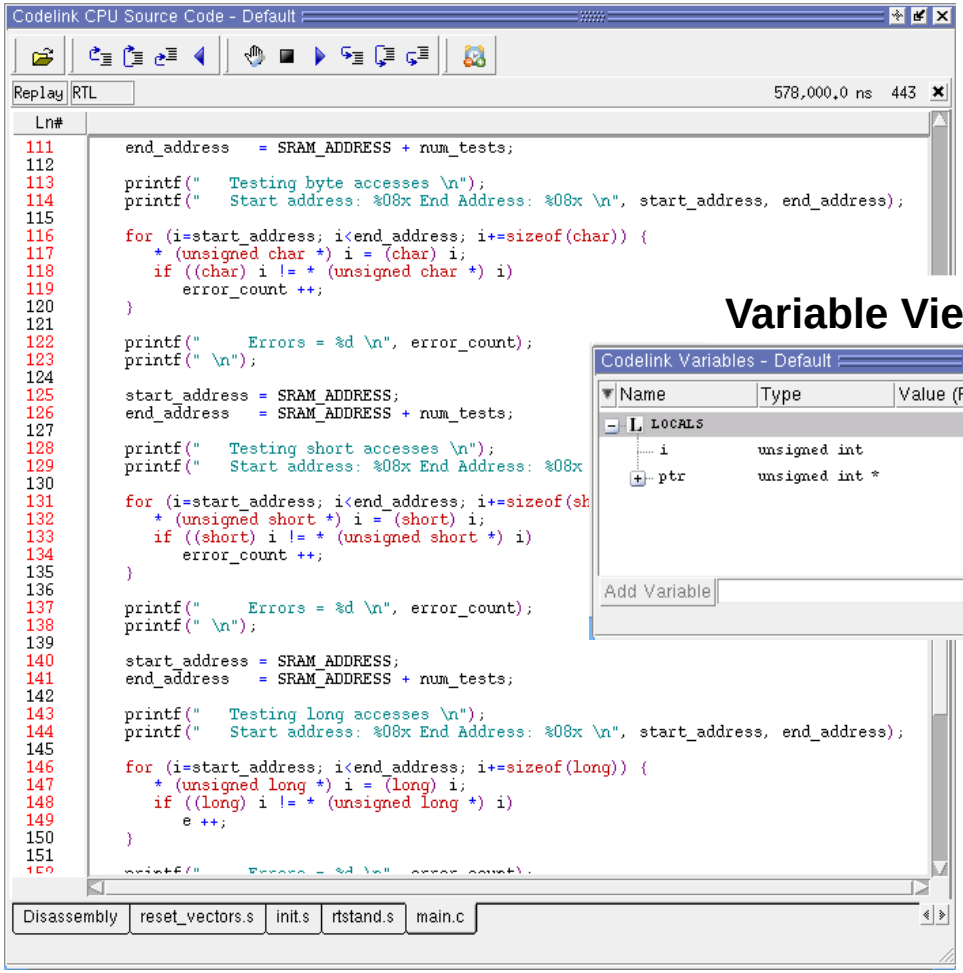

- Multiple Embedded Cores
- Heterogeneous Protocols
- Complex Interconnect
- Embedded Software

# Codelink Non-Intrusive Multi-Core Support

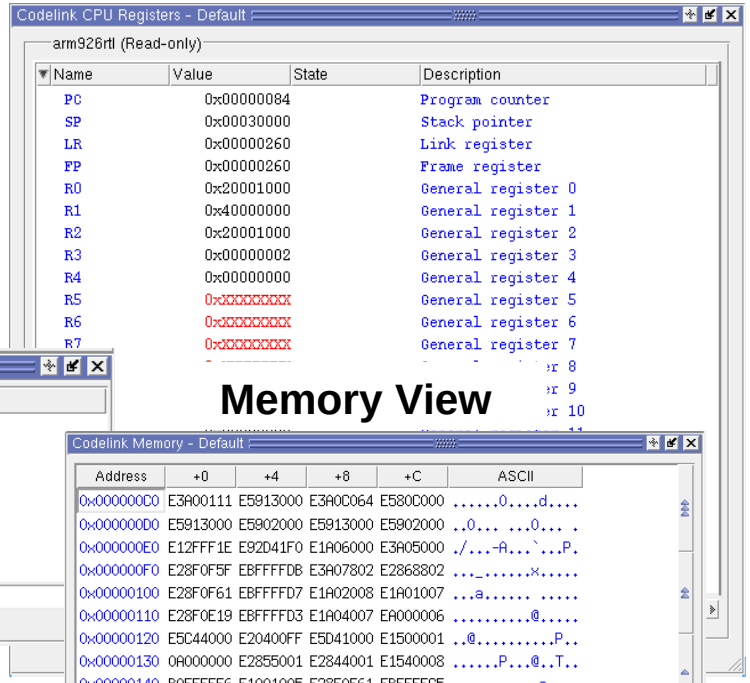


# Full Support of Classic SW Debug Views

## Source & Assembly View



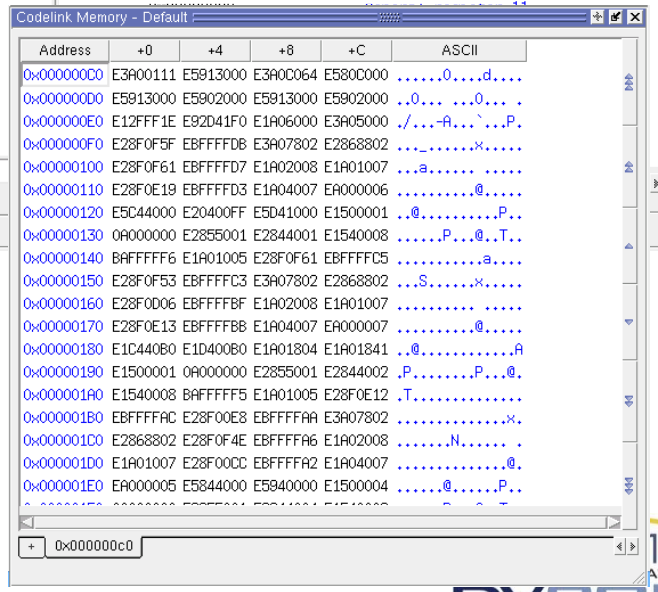

## Register View



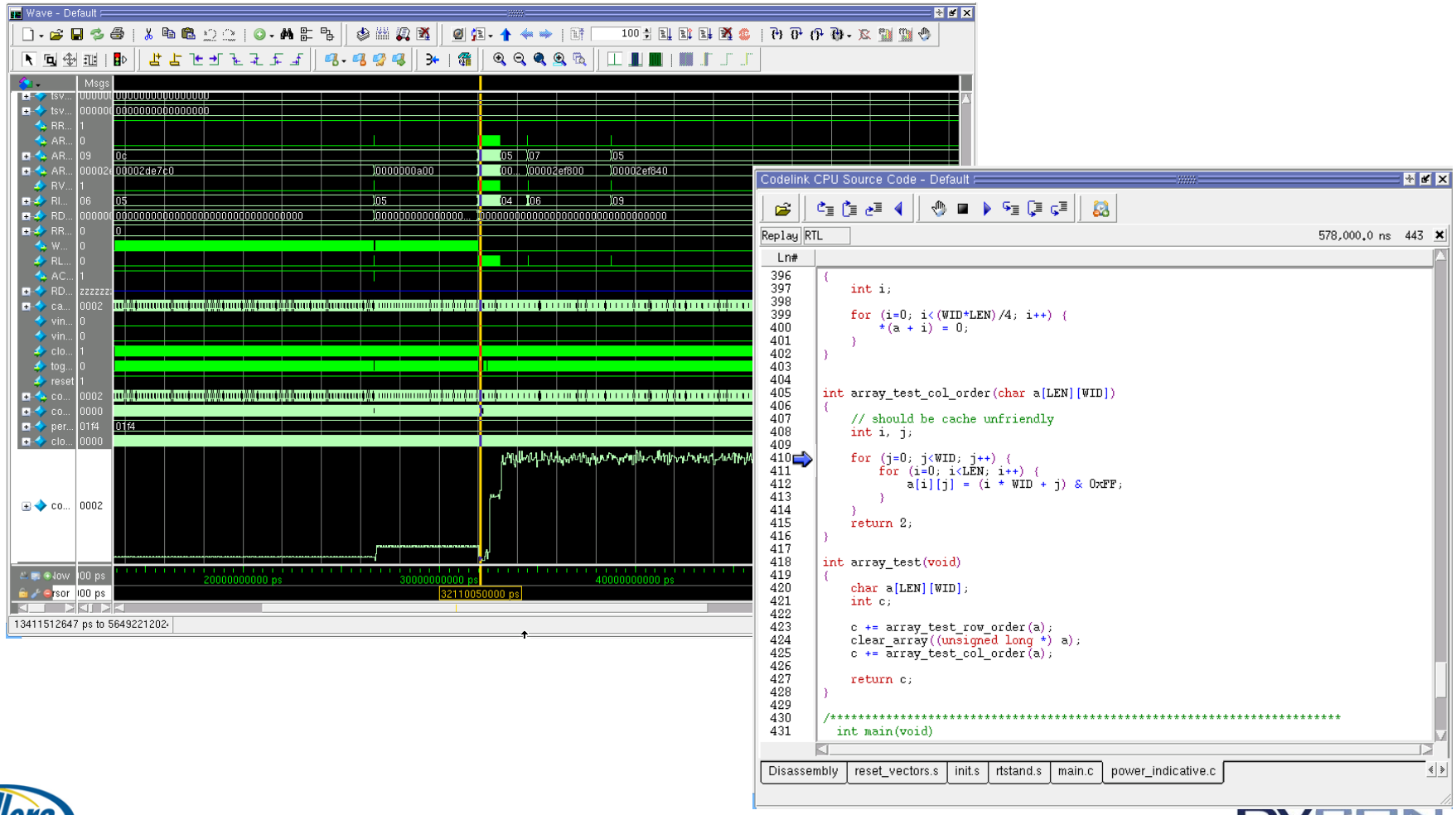

## Variable View




## Memory View



# Synchronized Hardware and Software Views



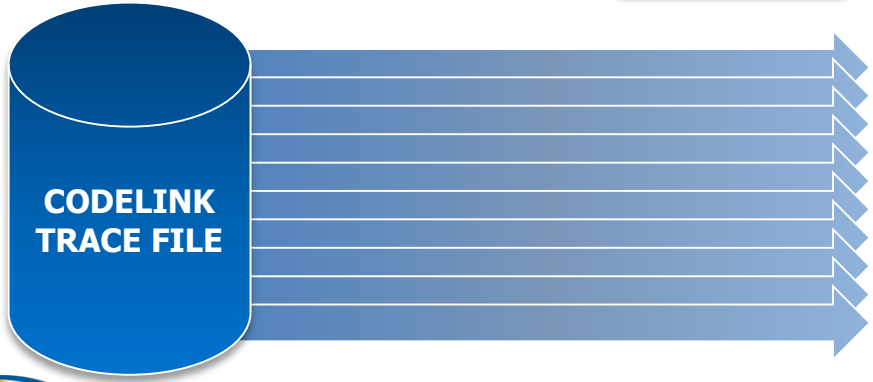
# New Paradigm:
# Post-Run Software Debug


JTAG
1 MHz

Single User

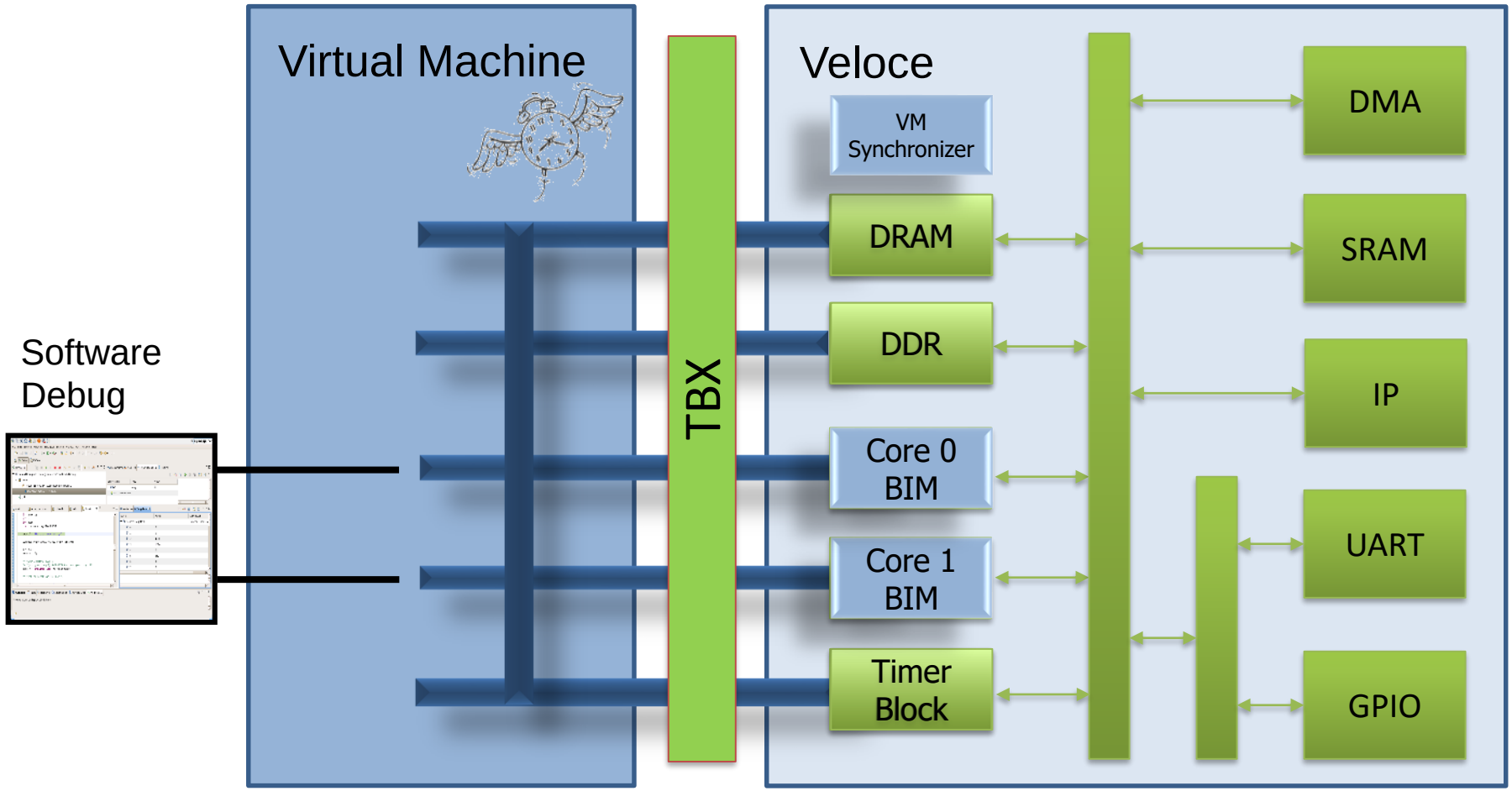
Post-Emulation

Codelink
50 MHz



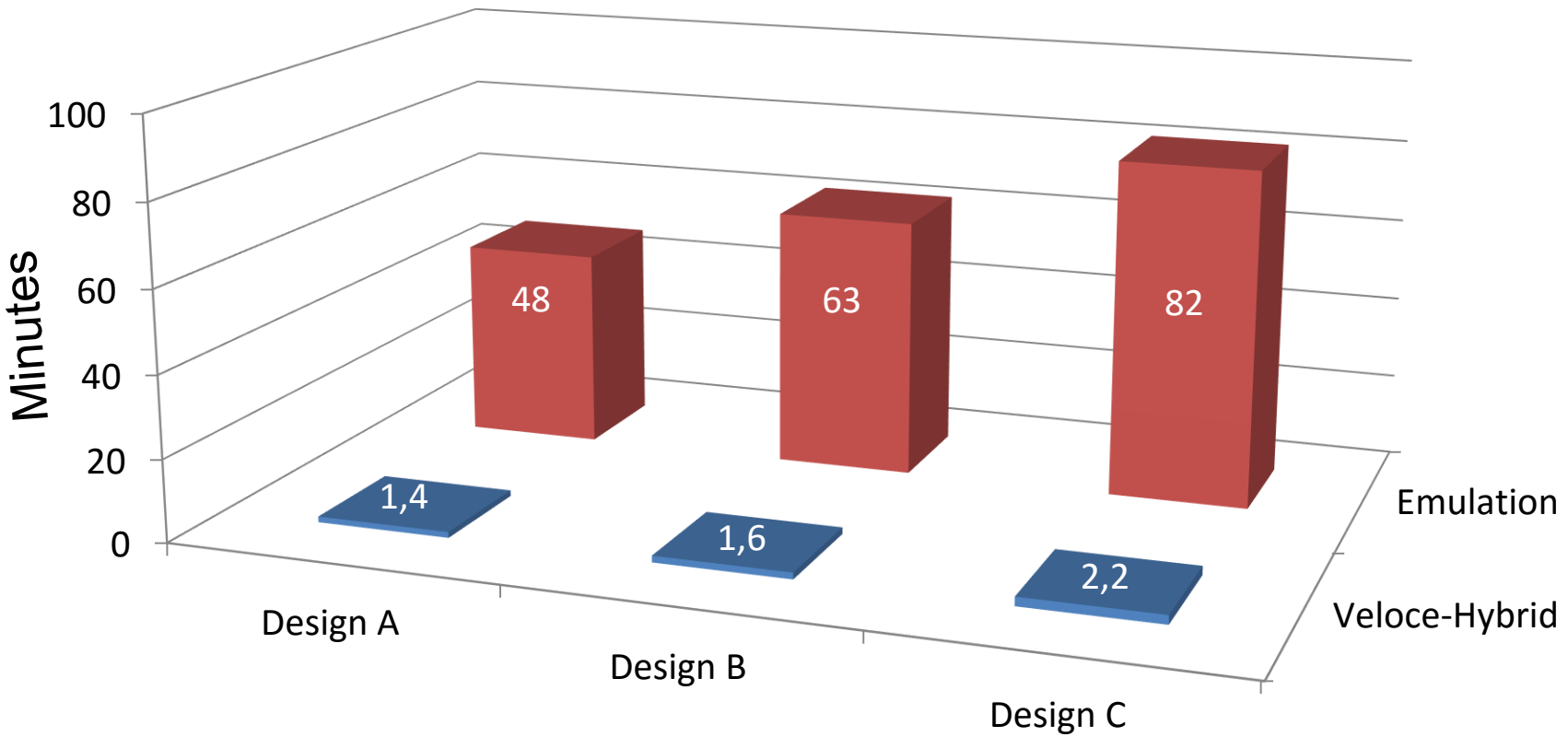

~10 Simultaneous users

# WarpCore: Veloce Hybrid Emulation




- Moves CPU/memory subsystem from the emulator into a virtual machine
- Speeds up execution by 50X and more

# Real World Results




Linux Boot Performance

Often, the interesting things happen after the OS boot

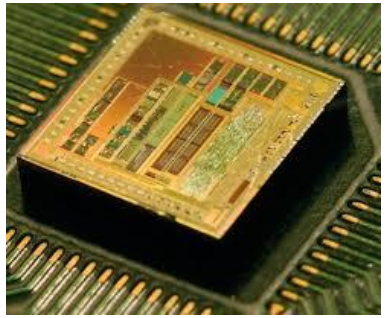
# Goals of New SW Debug Paradigm

- Non intrusive multi-core debug
- Process-level multi-core debug
- Backup rather than rerun from time zero
- Add “printf” without recompile and rerun
- Virtual machine for fast OS boot

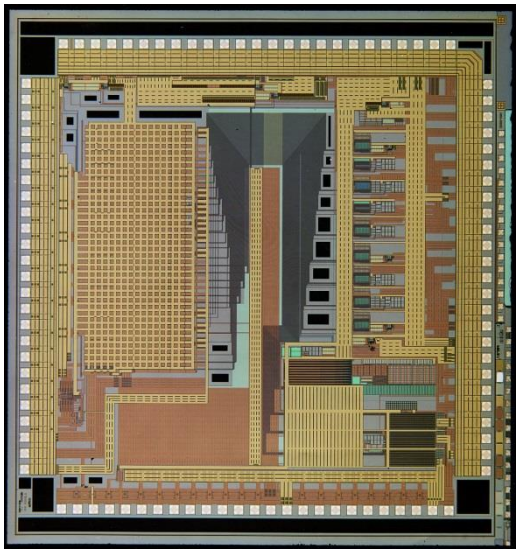
# Agenda

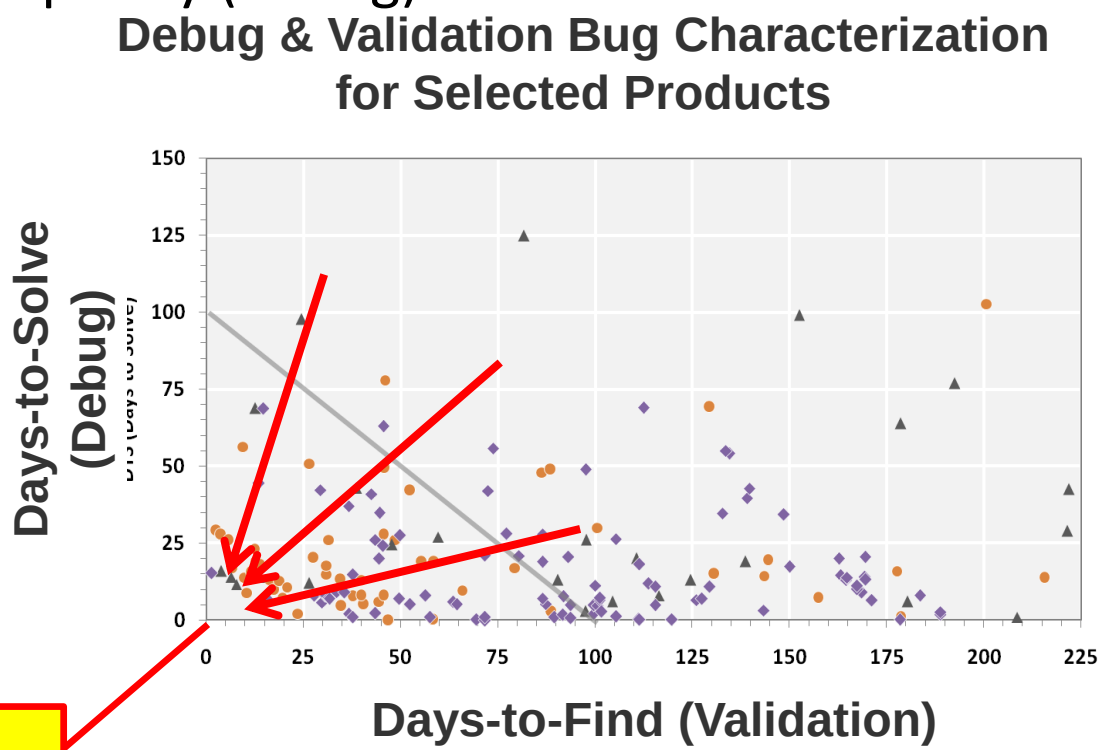
- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

# Goals of Post-Silicon Validation/Debug



- From the time silicon comes back:
  - Find <u>all</u> bugs quickly (validation)
  - Root-cause <u>all</u> bugs quickly (debug)
- Just make it easy

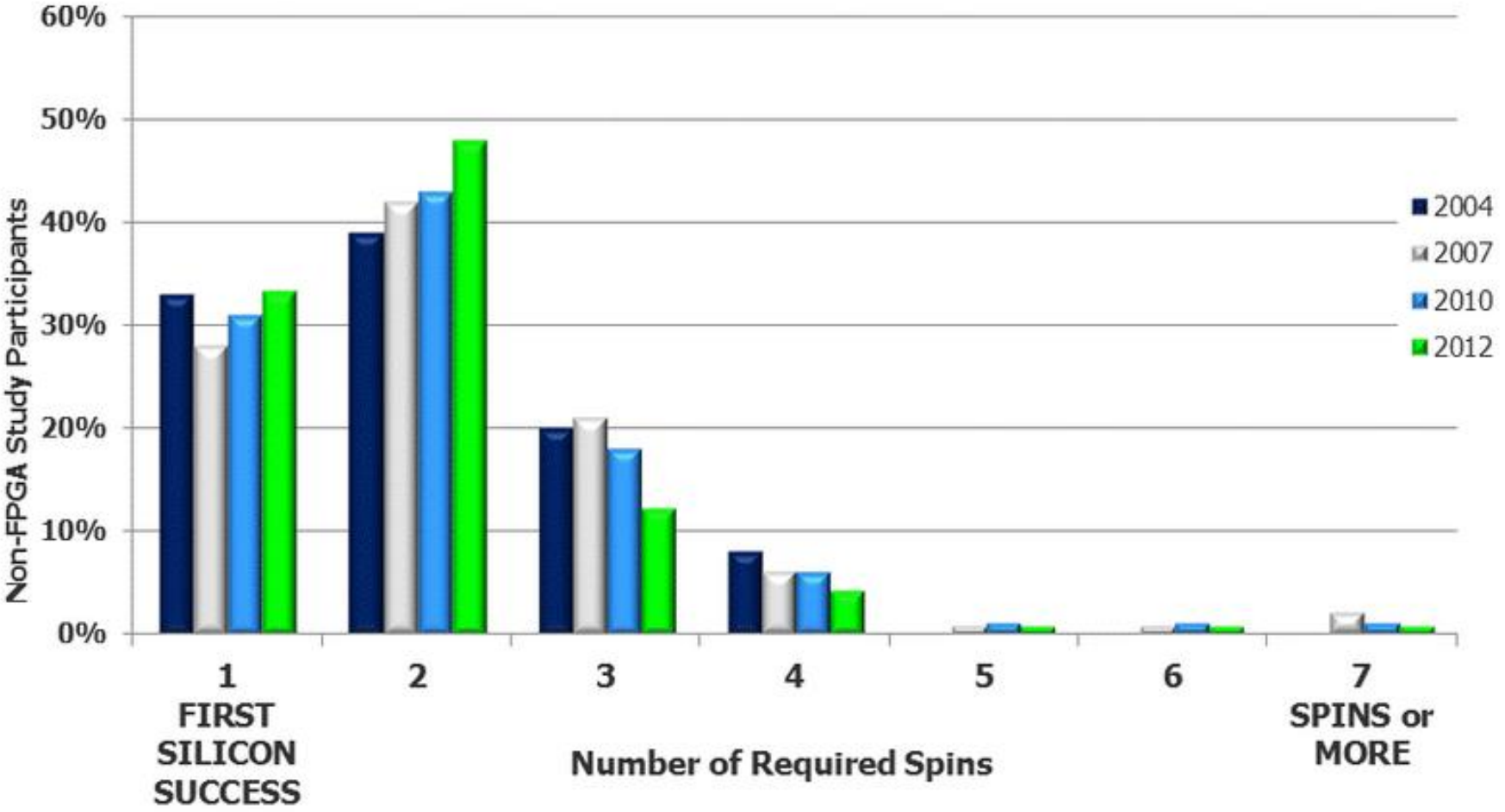





Drive bugs down

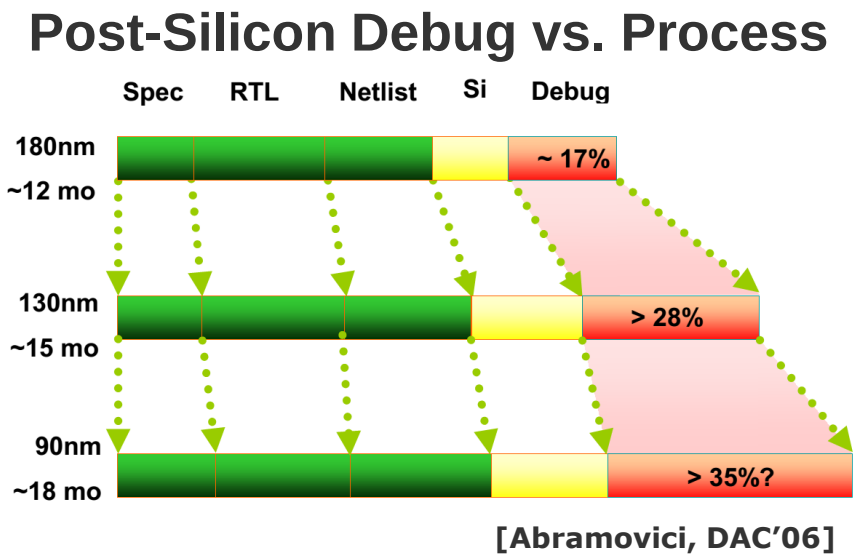
# Evolution of SOC Design

*Unplanned Respins => Missed Revenue!*

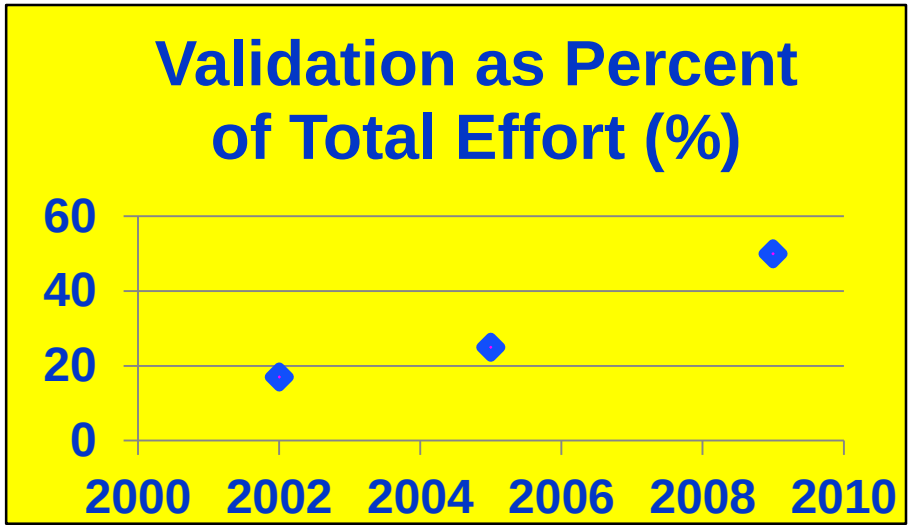



Wilson Research Group and Mentor Graphics, 2012 Functional Verification Study, Used with permission

# Post-Silicon Back-End Effort Growing



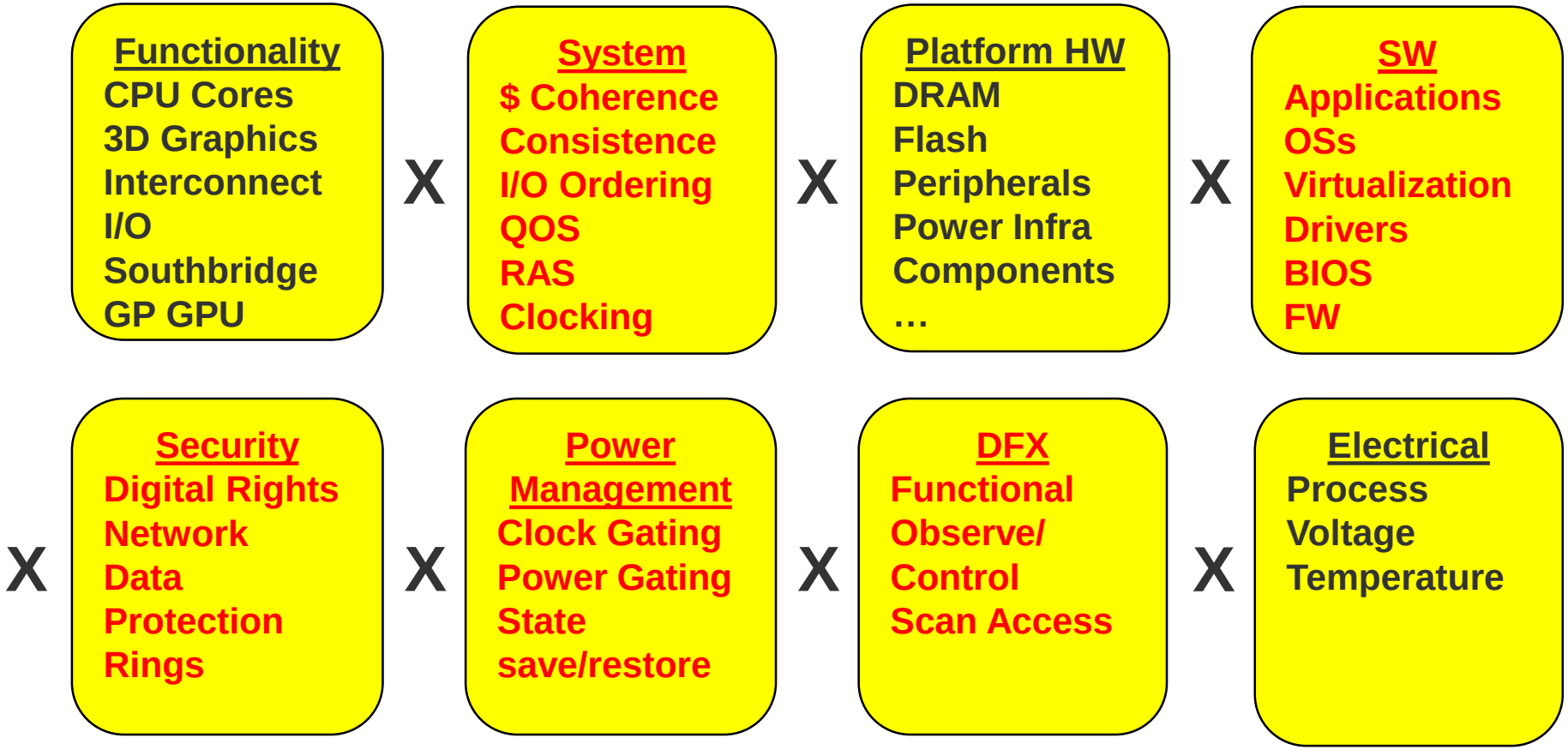

[Abramovici, DAC'06]




[Tim Cheng: "The Changing Roles of Verification and Test in the Late-Silicon Era," 2009 PROFIT]

- Post-silicon continues to require growing amount of total effort
- Data shows problem getting worse

# SOC Validation Space Challenge

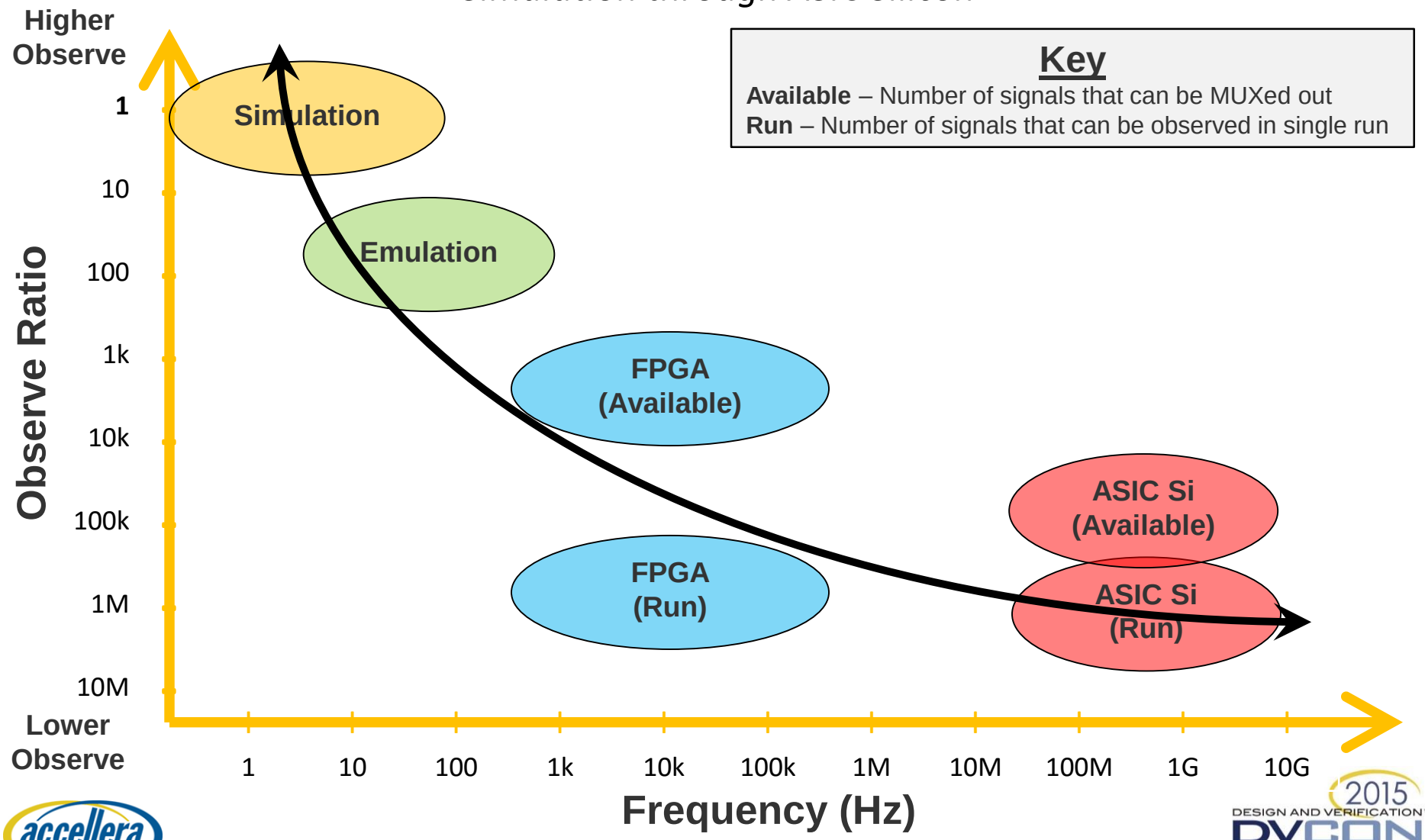



***What does all this add up to?***

***We're forced to debug post-silicon!***

# Observability Versus Frequency

*Simulation through ASIC Silicon*



# Post-Silicon System-Level Validation/Debug

*High-Level Feature Summary Flyover*

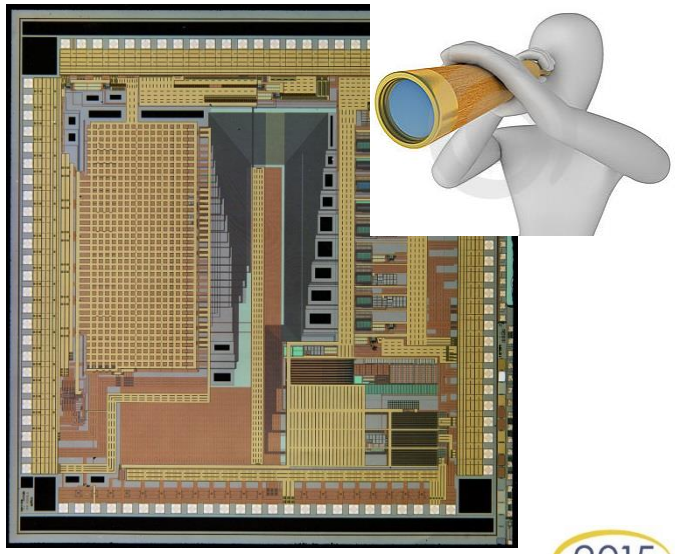
| Feature(s) | Comments |
|---|---|
| Program Flow Trace | •ARM: Coresight ETM PC trace<br>•AMD: BTHB<br>•Intel: Processor Trace |
| HW History Buffers | Small trace buffers in HW: THBs, EHB, LBR buffers, LTSSM buffers |
| Scandump[1] | •ATPG scanchains configured as single-chain, TDI to TDO<br>•Debug of hardlocks and more<br>•"bread and butter" feature for many companies<br>•Destructive in nature<br>•Analogous to "read-back" on FPGAs |
| Performance Monitor & Error Observe | Observe on pin, trigger ext. instruments, selective program flow trace |
| HW Flow Control | Breakpoint/single-step resume |
| HW Assertions | Active in parallel, no programming required |
| Bug Replay (Si=>tester, sim.) | Really both legacy and emerging<br>Expensive and custom per company |

***Where do you turn for more difficult SOC interaction bugs?***

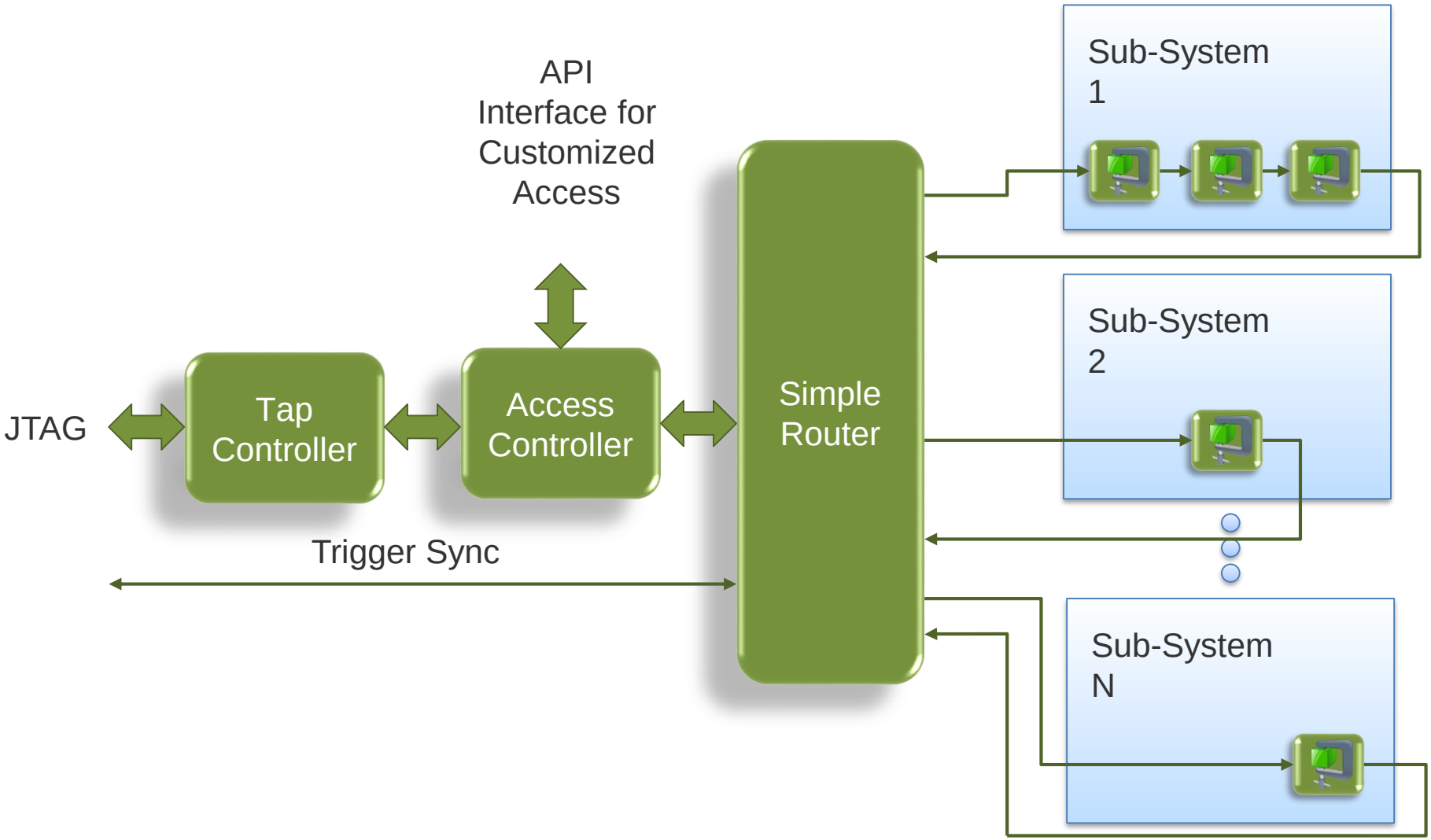
1. Array and main memory dump applies in addition here too.

# The Solution: On-Chip Logic Analyzer

- On-chip IP and software
- Software tools for:
  - Instrumenting design
  - Run-Time Programming
    - Signal observe & triggering
    - Trace dump/waveform view
- For use on FPGAs and ASICs
- Why?
  - Faster bug root-cause
  - Shorter time-to-market

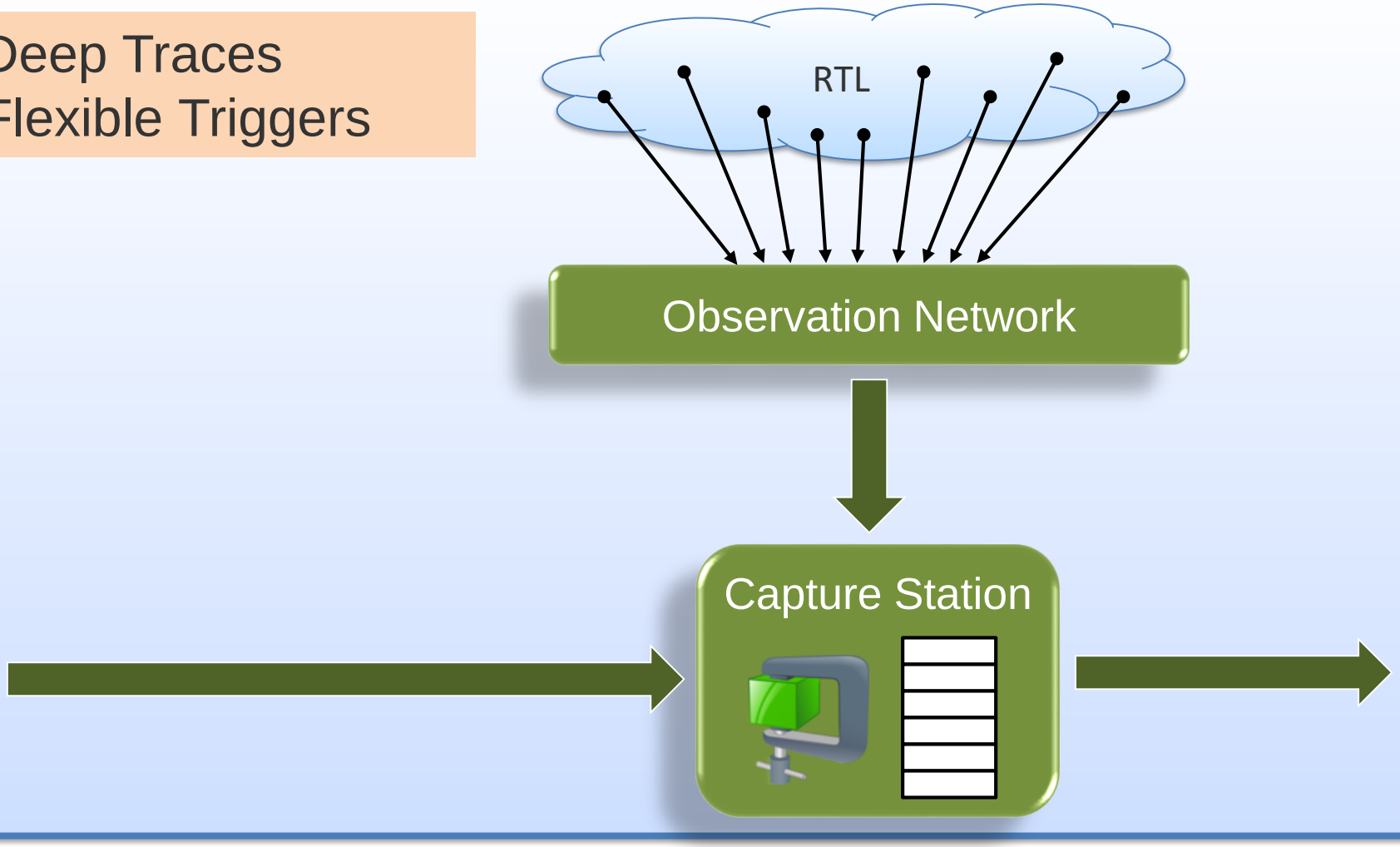


# Architected for Post Silicon Debug

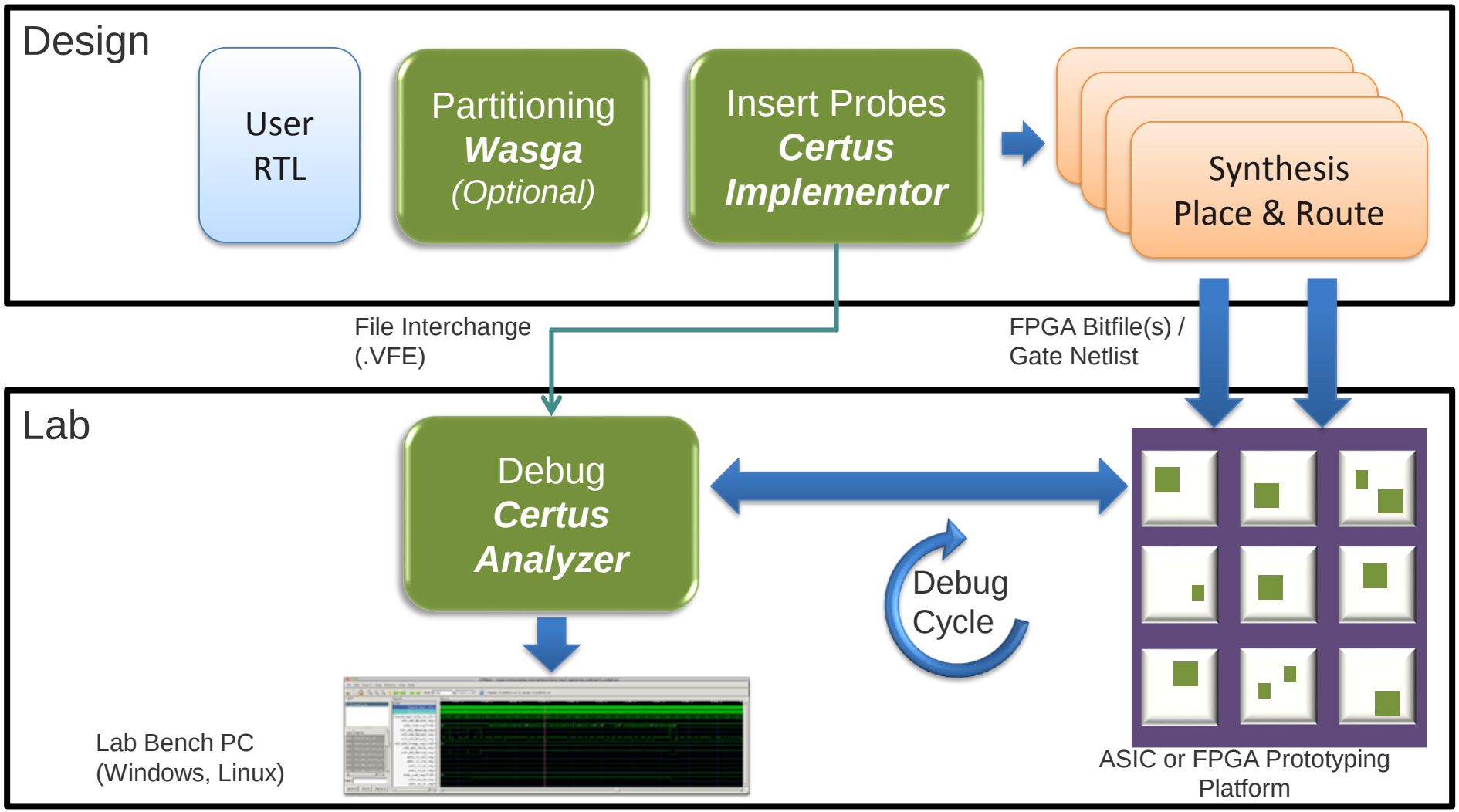


# Capture Station

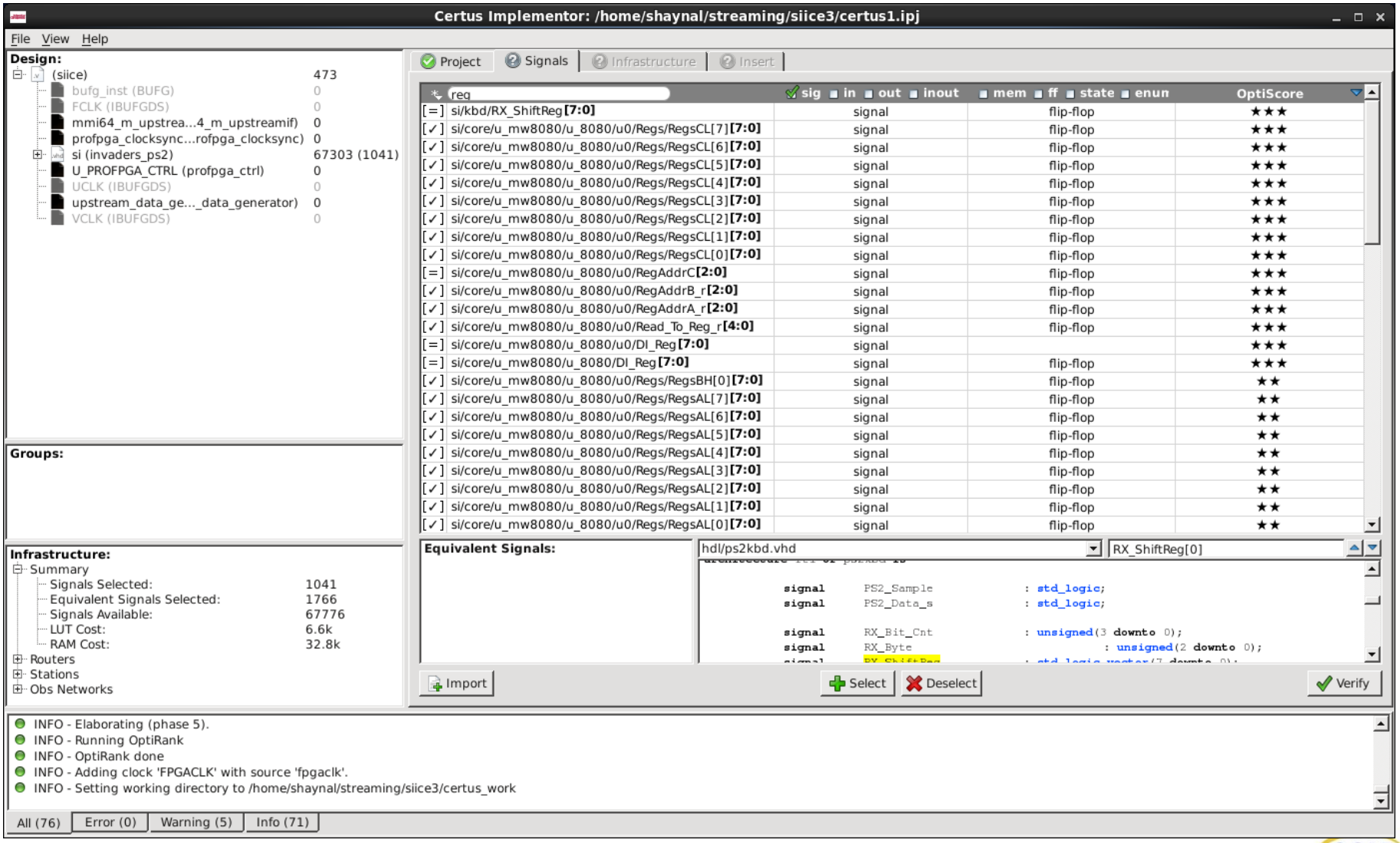
- Deep Traces
- Flexible Triggers



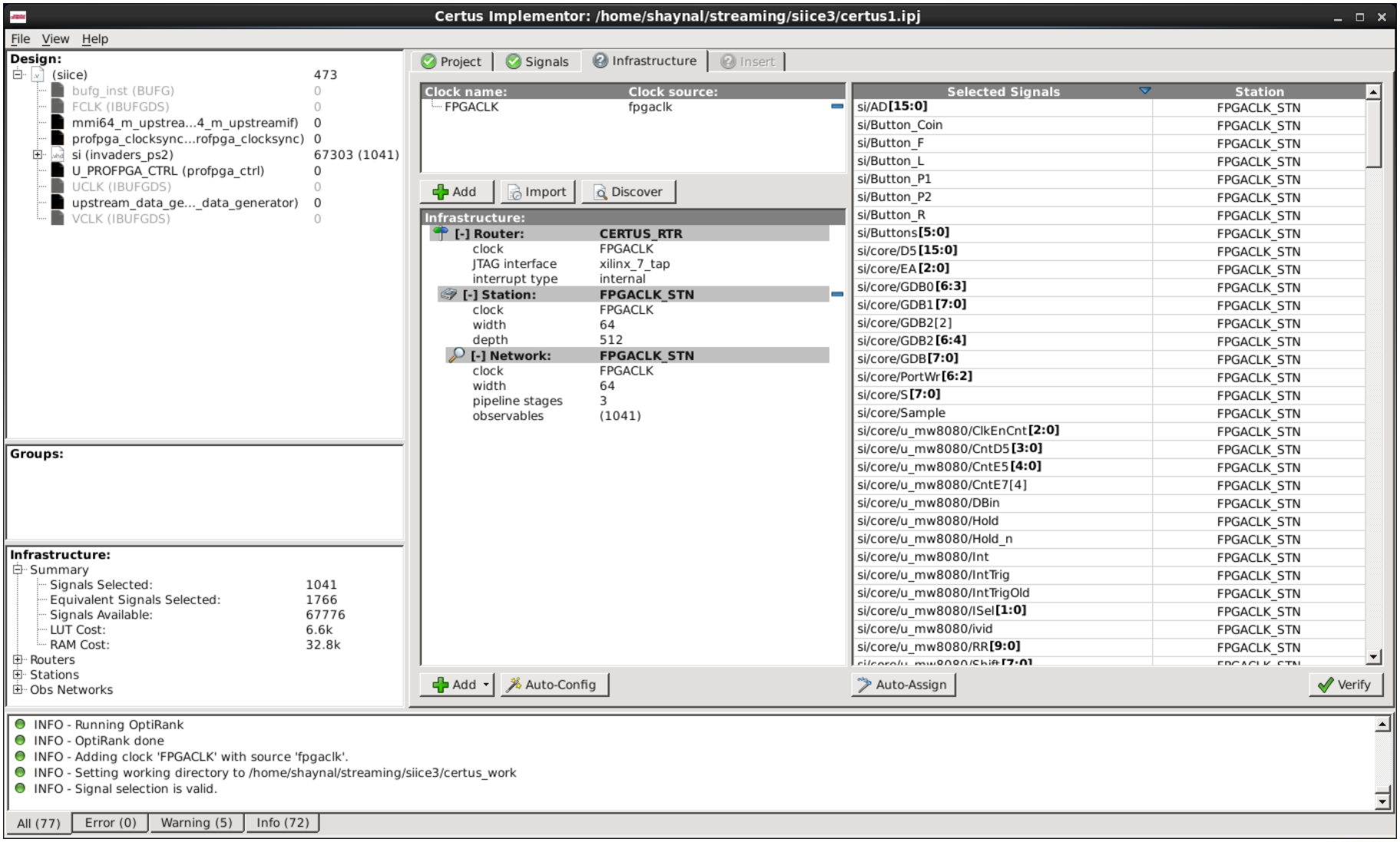
# Certus User Flow



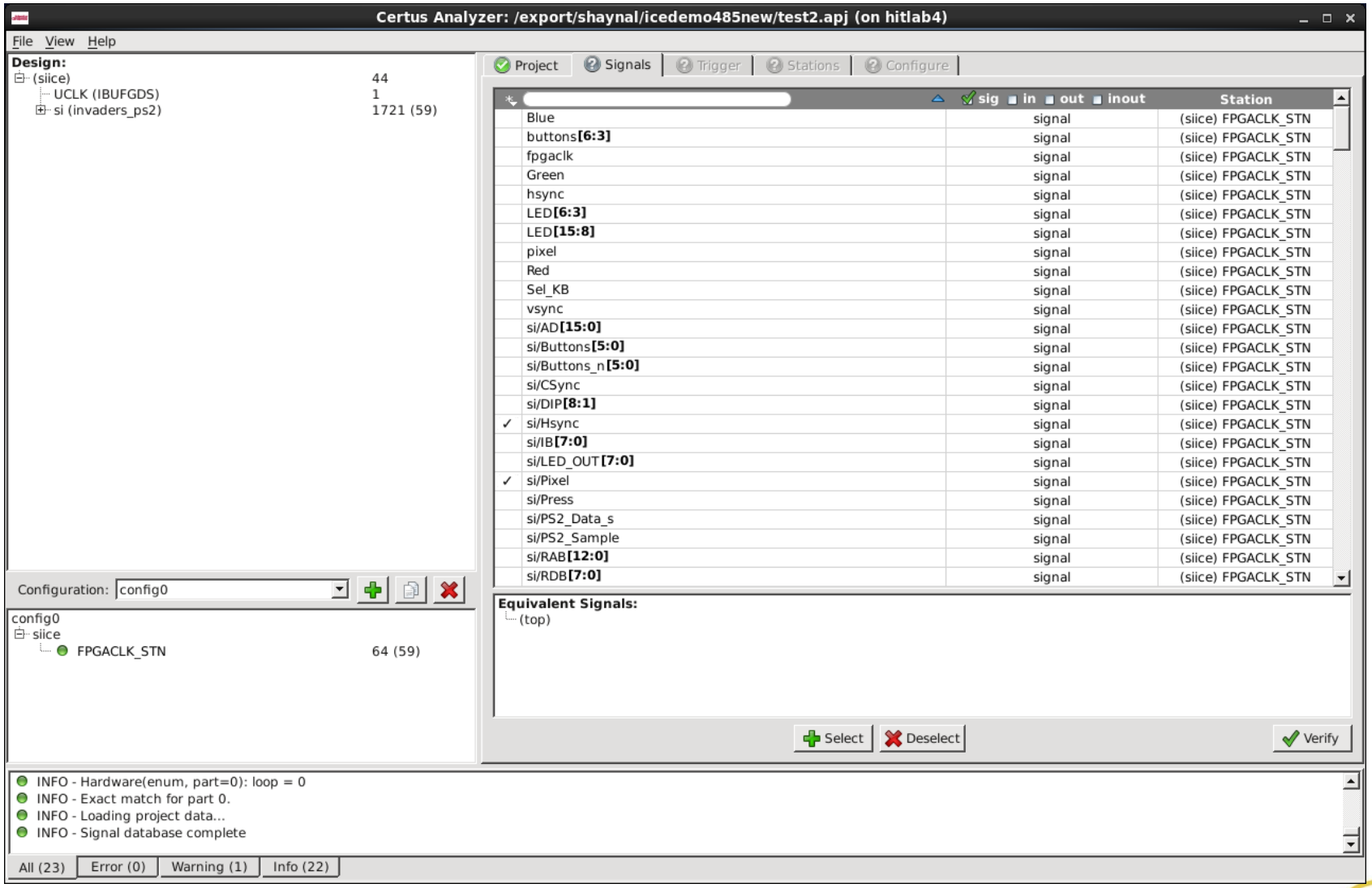
# Instrumented Signal Selection



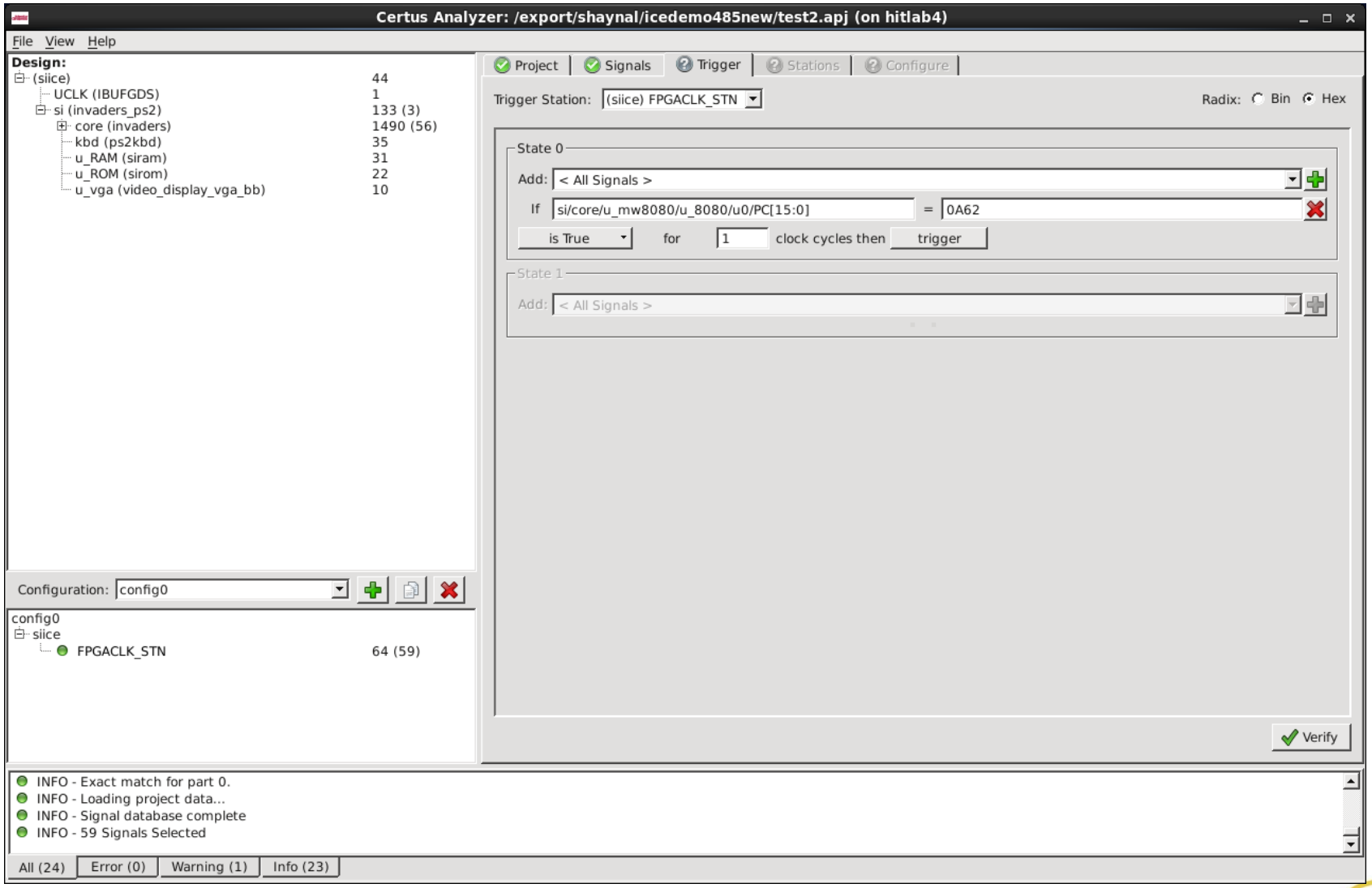
# Infrastructure



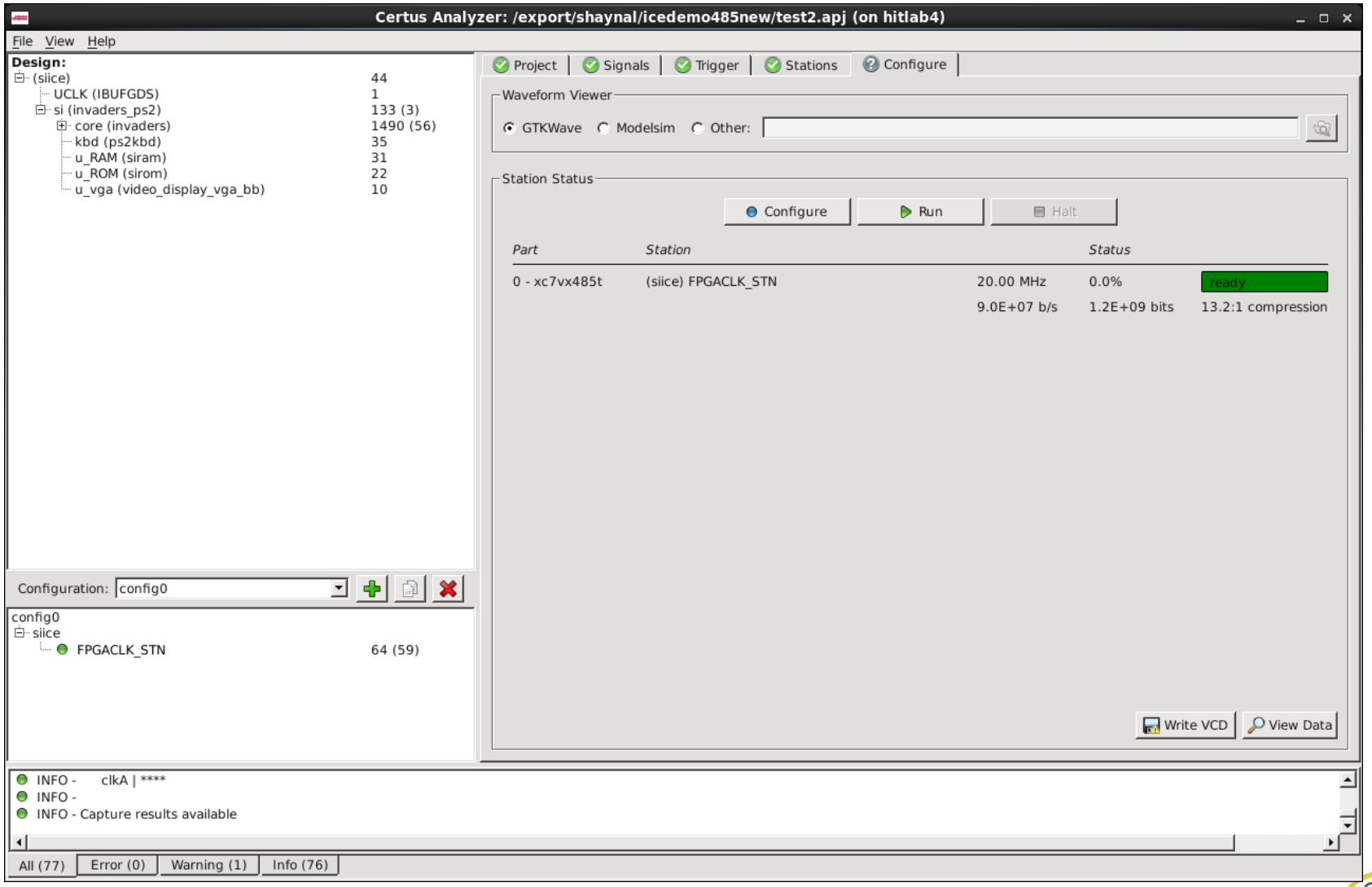
# Signal Selection



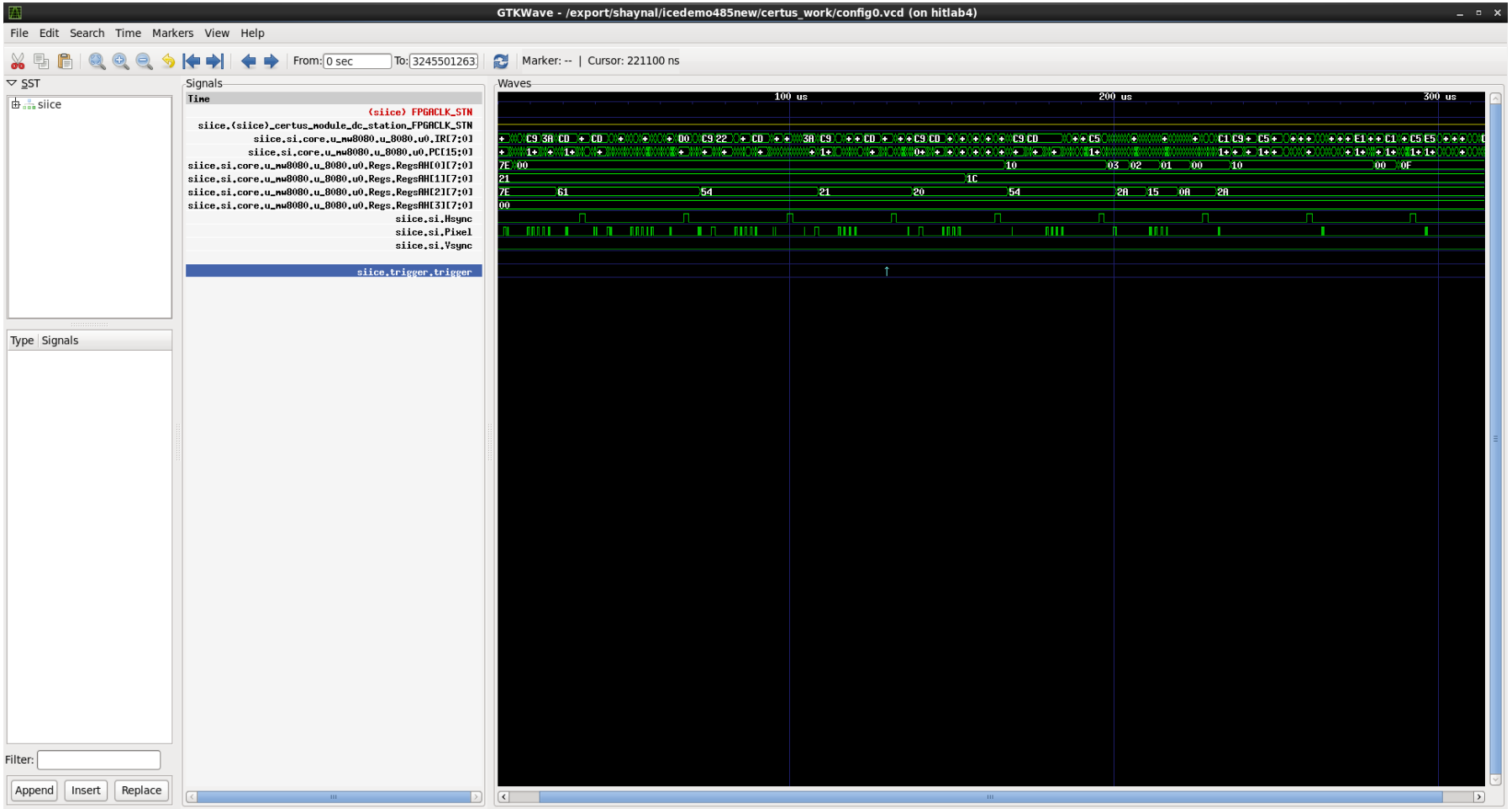
# Set the Trigger Condition



# Run Content


Certus Analyzer: /export/shaynal/icedemo485new/test2.apj (on hitlab4)
File View Help
Design:
(siice) 44
UCLK (IBUFGDS) 1
si (invaders_ps2) 133 (3)
core (invaders) 1490 (56)
kbd (ps2kbd) 35
u_RAM (siram) 31
u_ROM (sirom) 22
u_vga (video_display_vga_bb) 10
Configuration: config0
config0
siice
FPGACLK_STN 64 (59)
Project
Signals
Trigger
Stations
Configure
Waveform Viewer
GTKWave
Modelsim
Other:
Station Status
Configure
Run
Halt
Part
Station
Status
0 - xc7vx485t
(siice) FPGACLK_STN
20.00 MHz
0.0%
ready
9.0E+07 b/s
1.2E+09 bits
13.2:1 compression
Write VCD
View Data
INFO - clkA | ****
INFO -
INFO - Capture results available
All (77)
Error (0)
Warning (1)
Info (76)

# Post-Silicon Waveform Viewing is Key



# Mentor Graphics Certus Solution

- SW & HW solution for FPGA & silicon validation
  - Broad visibility and deep traces, minimal resource cost
  - Make post-silicon easy
- Shorter back-end time-to-market & lower schedule risk
  - Flexibility to meet design/business needs
  - Enterprise-level EDA SW support
  - Productivity efficiency
  - Enables increased focus on core business
- Mentor Graphics serious about post-silicon space

# Summary - Key Recommendations

- Find bugs as early as possible
  - Before time-consuming, resource intensive regressions are launched
- Successively refine low power verification in every D&V phase
- Leverage new debug technologies to expedite causality discovery
- Break down the wall between RTL and firmware verification
- Plan ahead for post-silicon debug

# Agenda

- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

Thank You!

Any Questions?

accellera SYSTEMS INITIATIVE

2015 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE