

Advanced, High-Throughput Debug From Design to Silicon

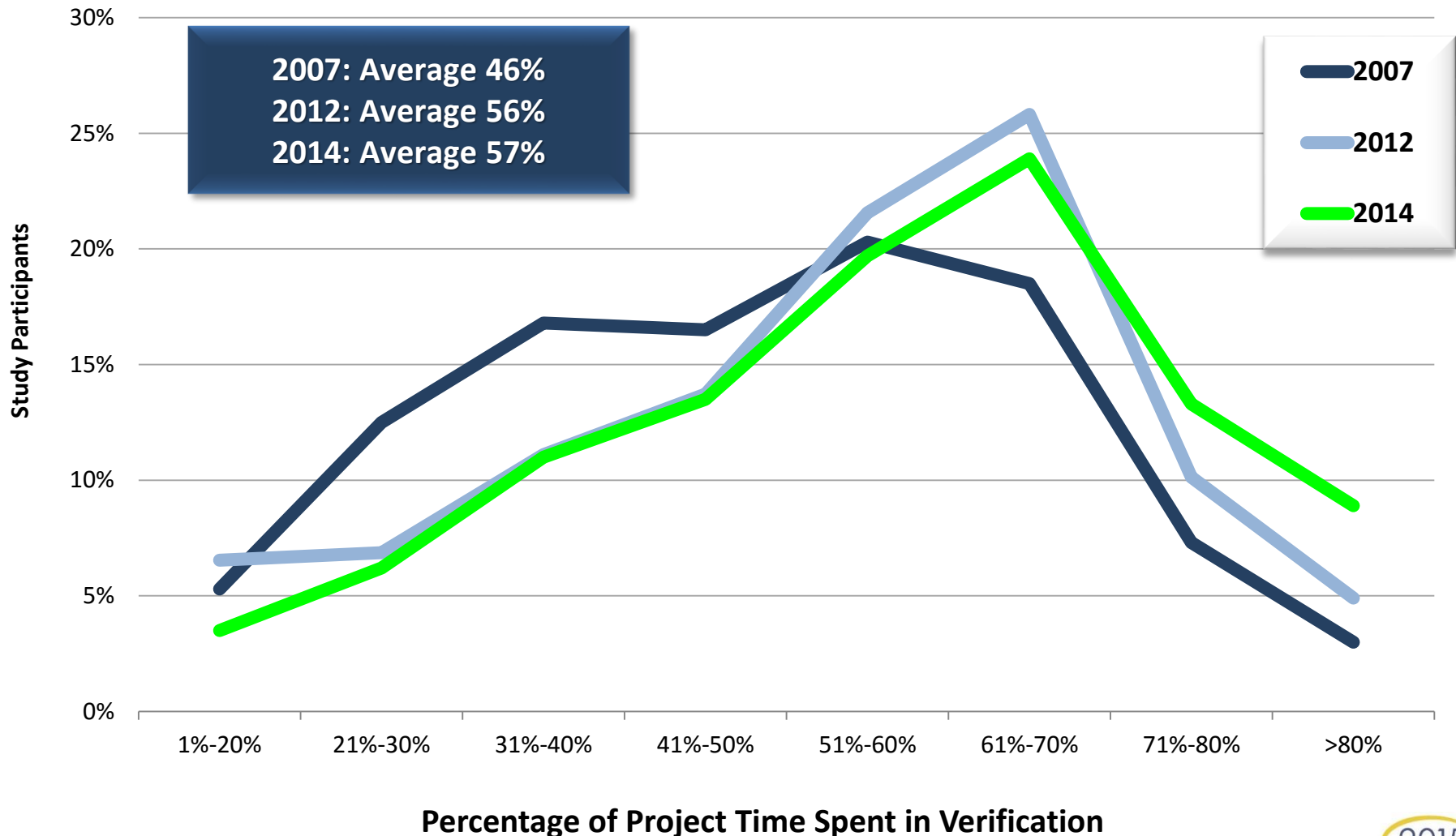
Gordon Allan & Michael Horn
Mentor Graphics Inc



Agenda

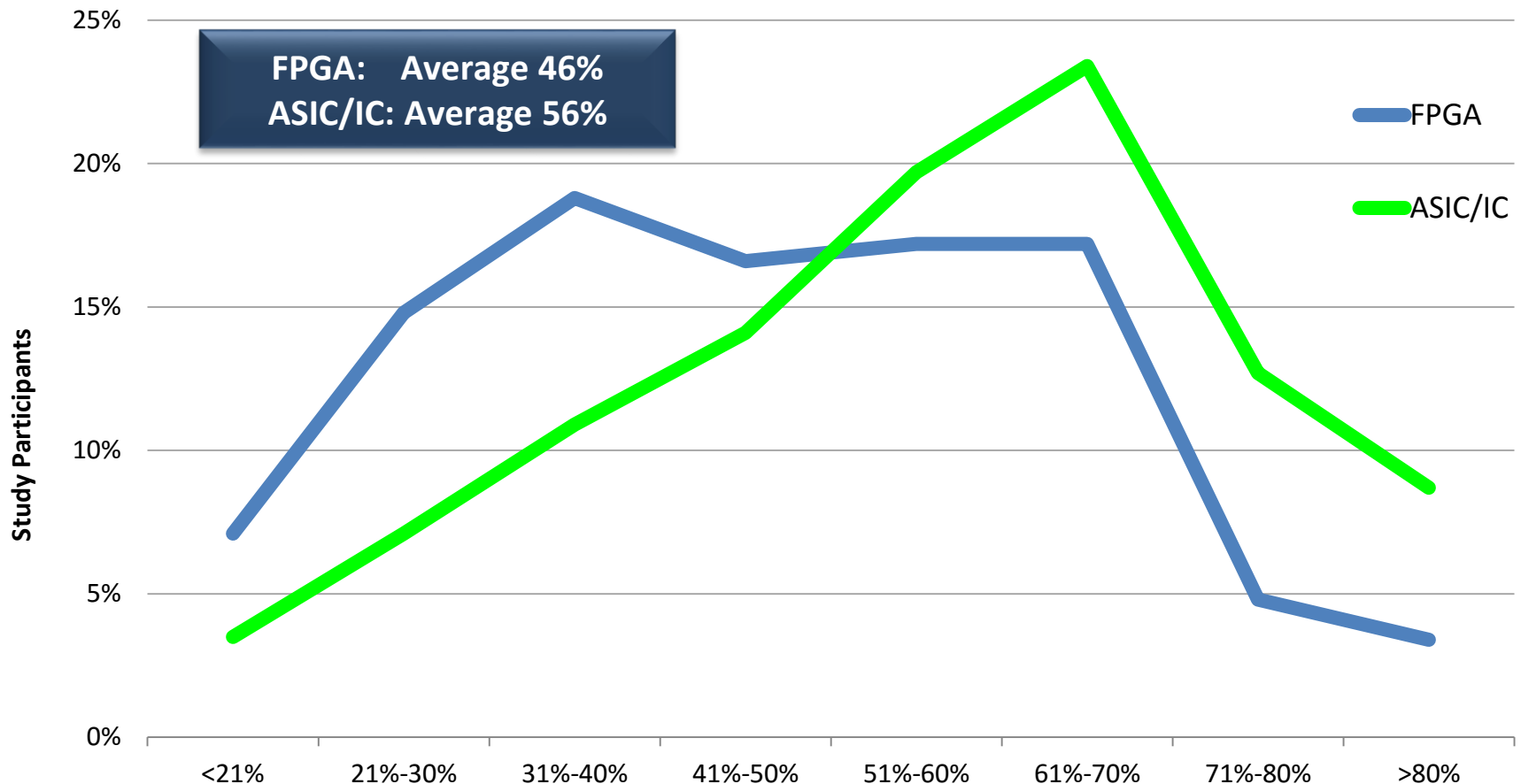
- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

Verification Consumes Majority of Project Time



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

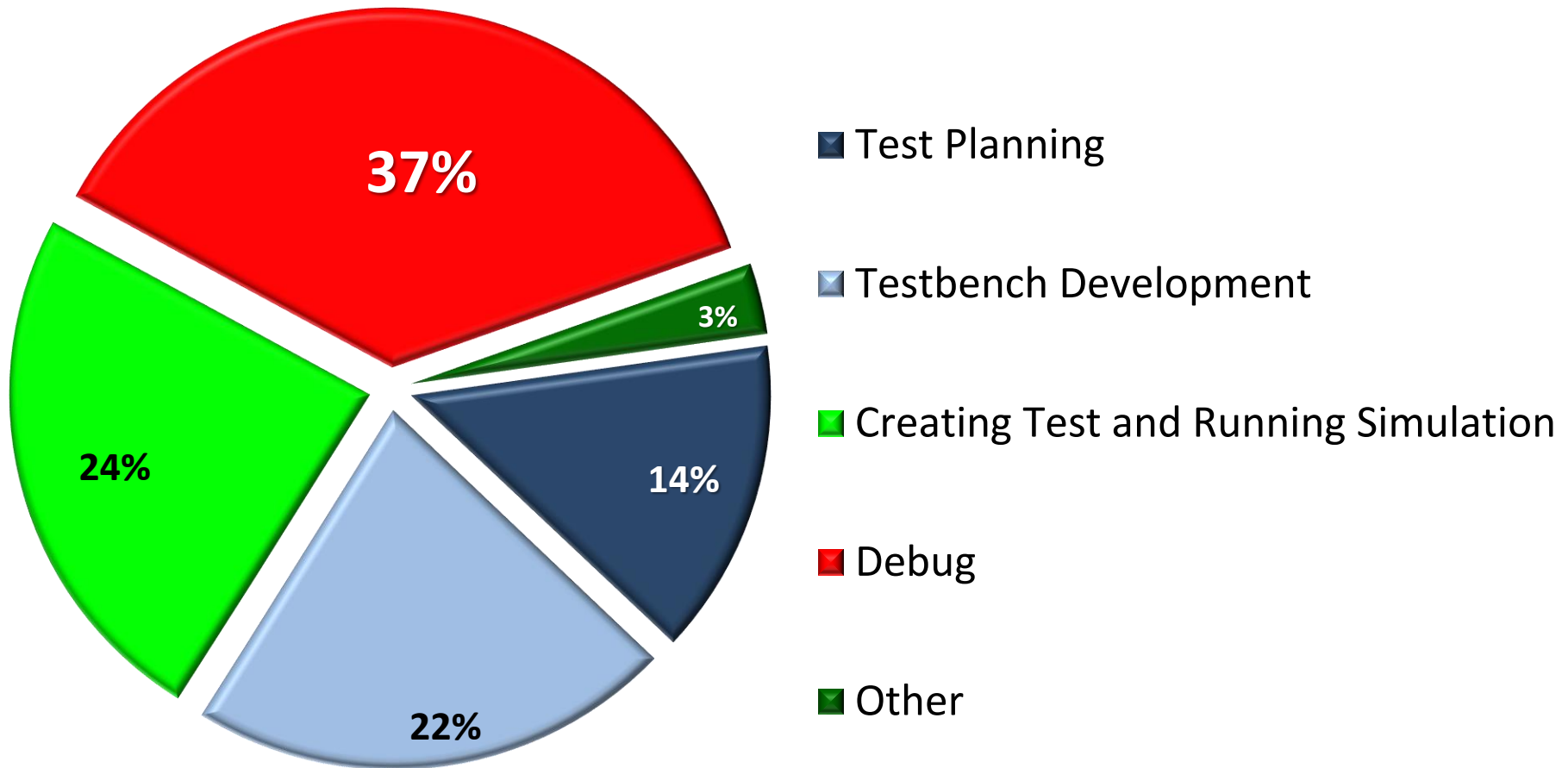
Verification Consumes Majority of Project Time for FPGAs too



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

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

Where Verification Engineers Spend Their Time

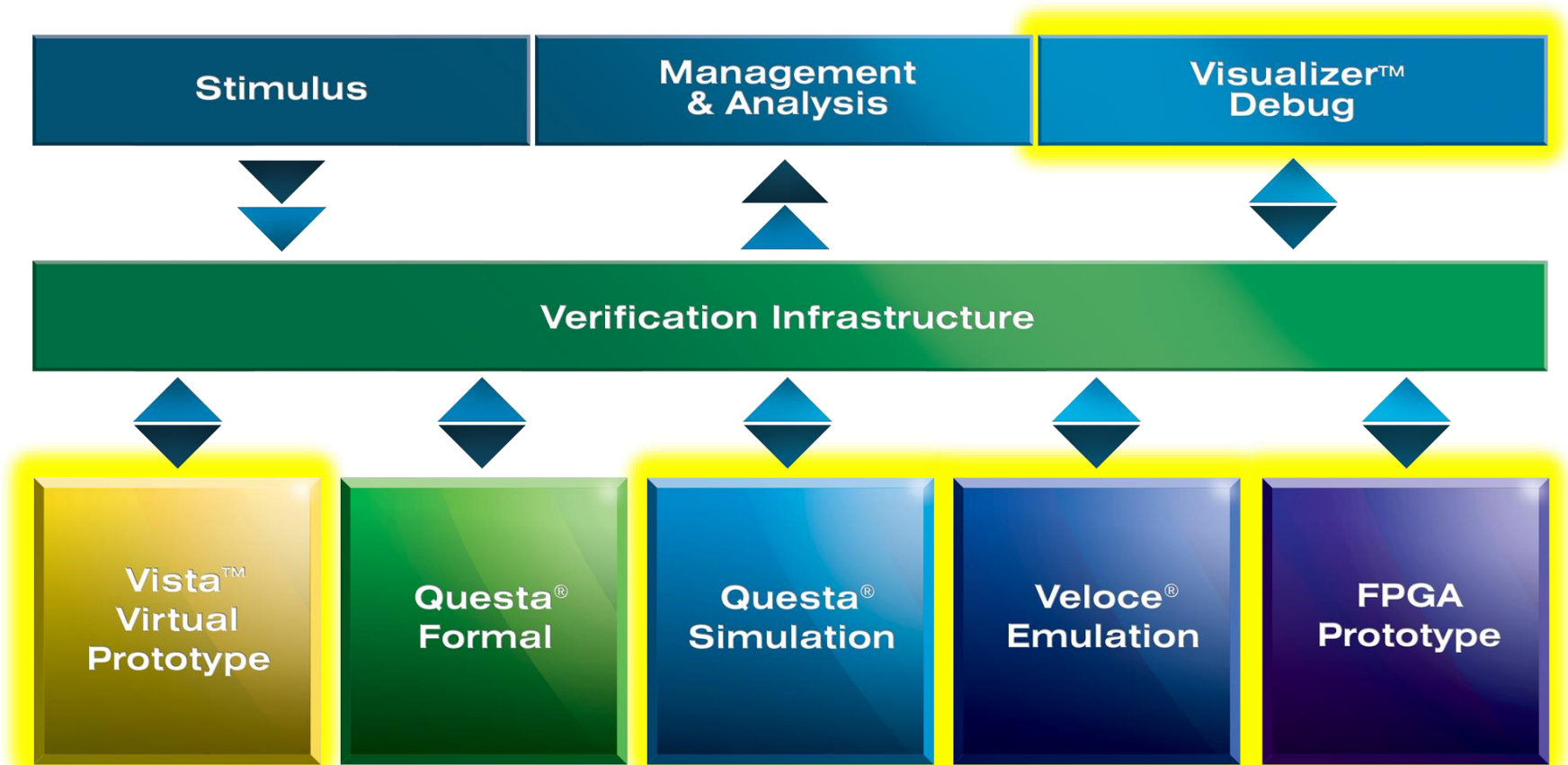


Source: Wilson Research Group and Mentor Graphics, 2014 Functional Verification Study

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

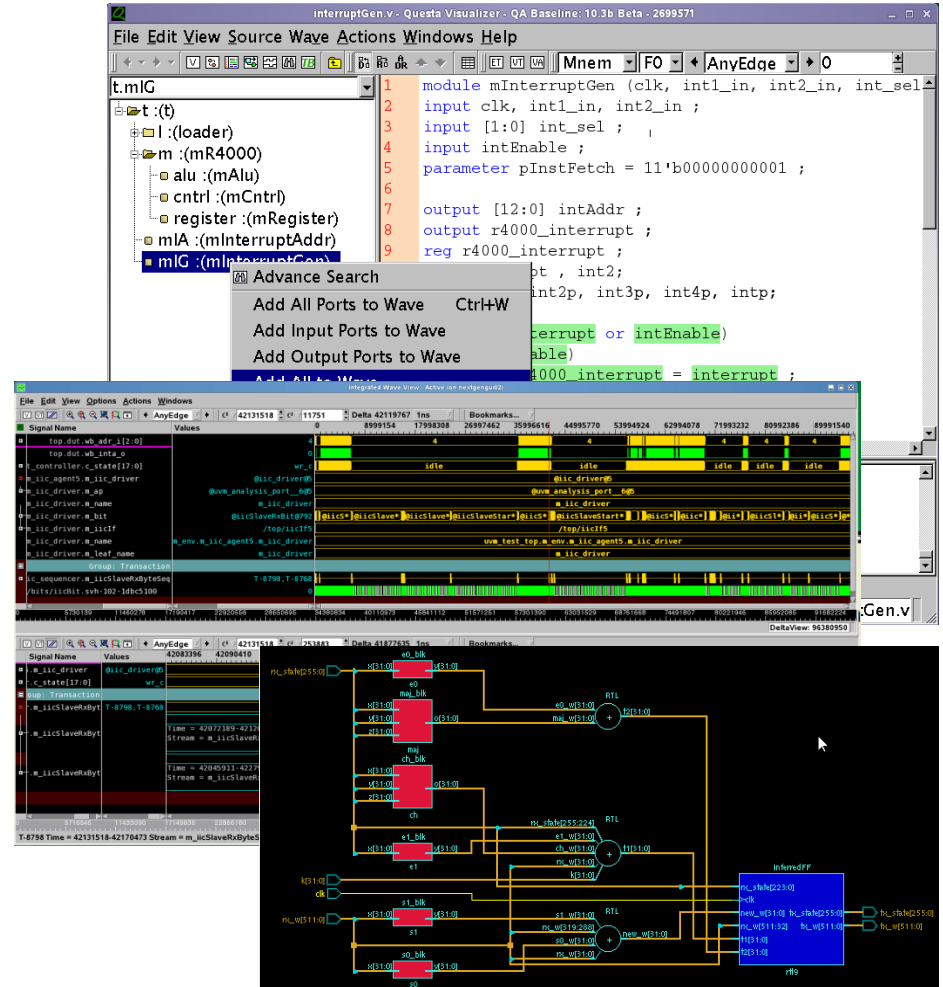
- Fast GUI data model
 - Technology acquired from Axiom Design Automation
 - Integrated with Questa & Veloce compilation and execution flows
 - Capacity to load 500M+ gate designs in seconds
- Fast and efficient waveform database
 - Custom compact database format
 - Standalone option to enable worldwide 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

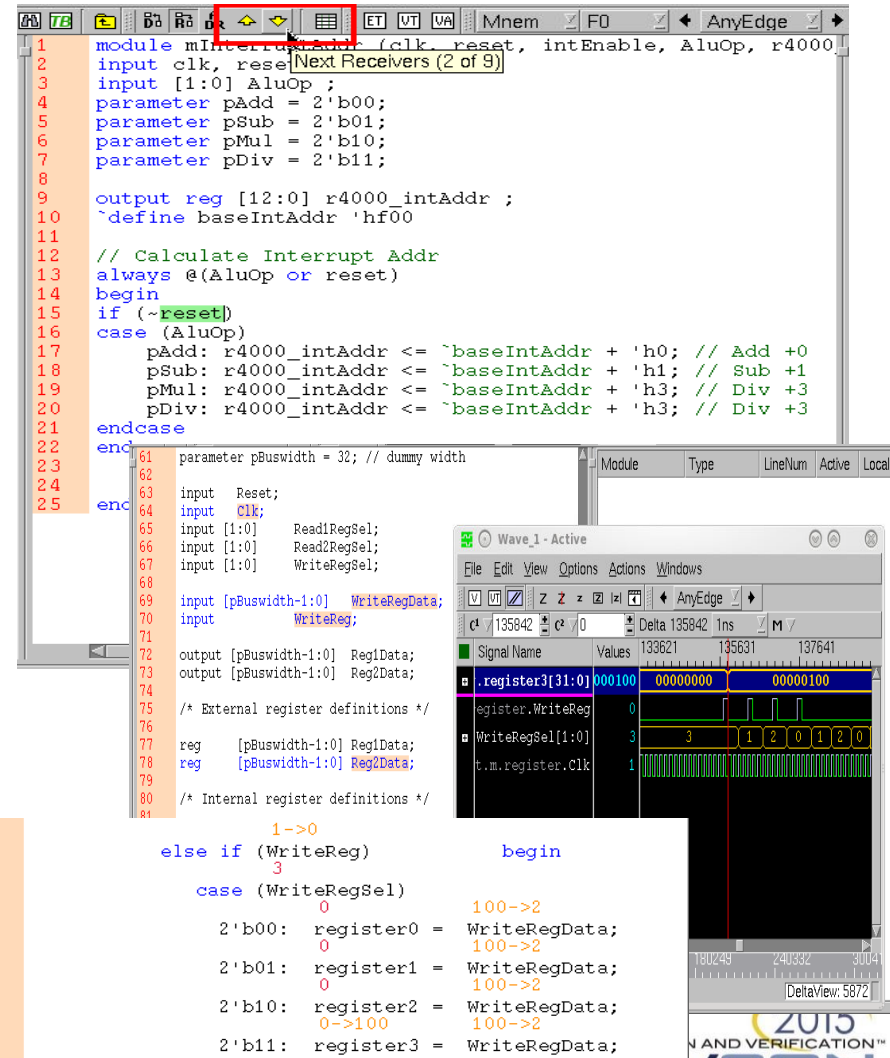
- Intuitive
 - Natural layout
 - Easy navigation
 - Quick mouse-clicks
- Responsive
 - Fast waveform
 - Fast source windows
 - All in time-sync
- Advanced Schematic View
 - Full featured
 - For both RTL and gates



Powerful Design Comprehension Capabilities

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

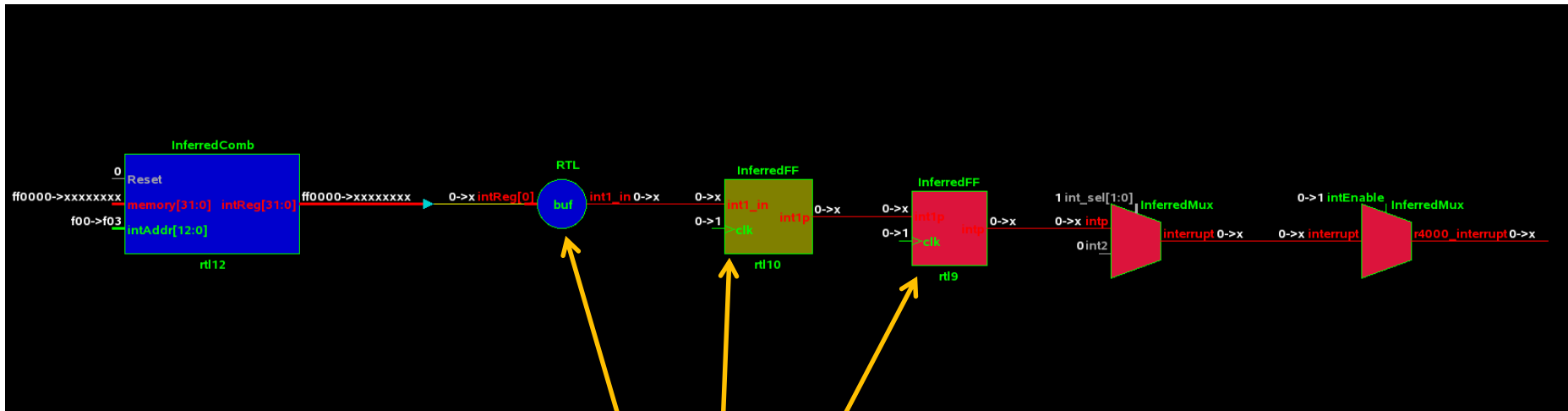
- 3 key navigation capabilities
 - Drivers
 - Receivers
 - Active Drivers / Receivers
- Easy to trace back and forth
 - Single click
- Quickly pinpoint activity in time
 - Execution trace
 - Popup tool tips
 - Value annotation



Unique TimeCone View to Find Cause Faster

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

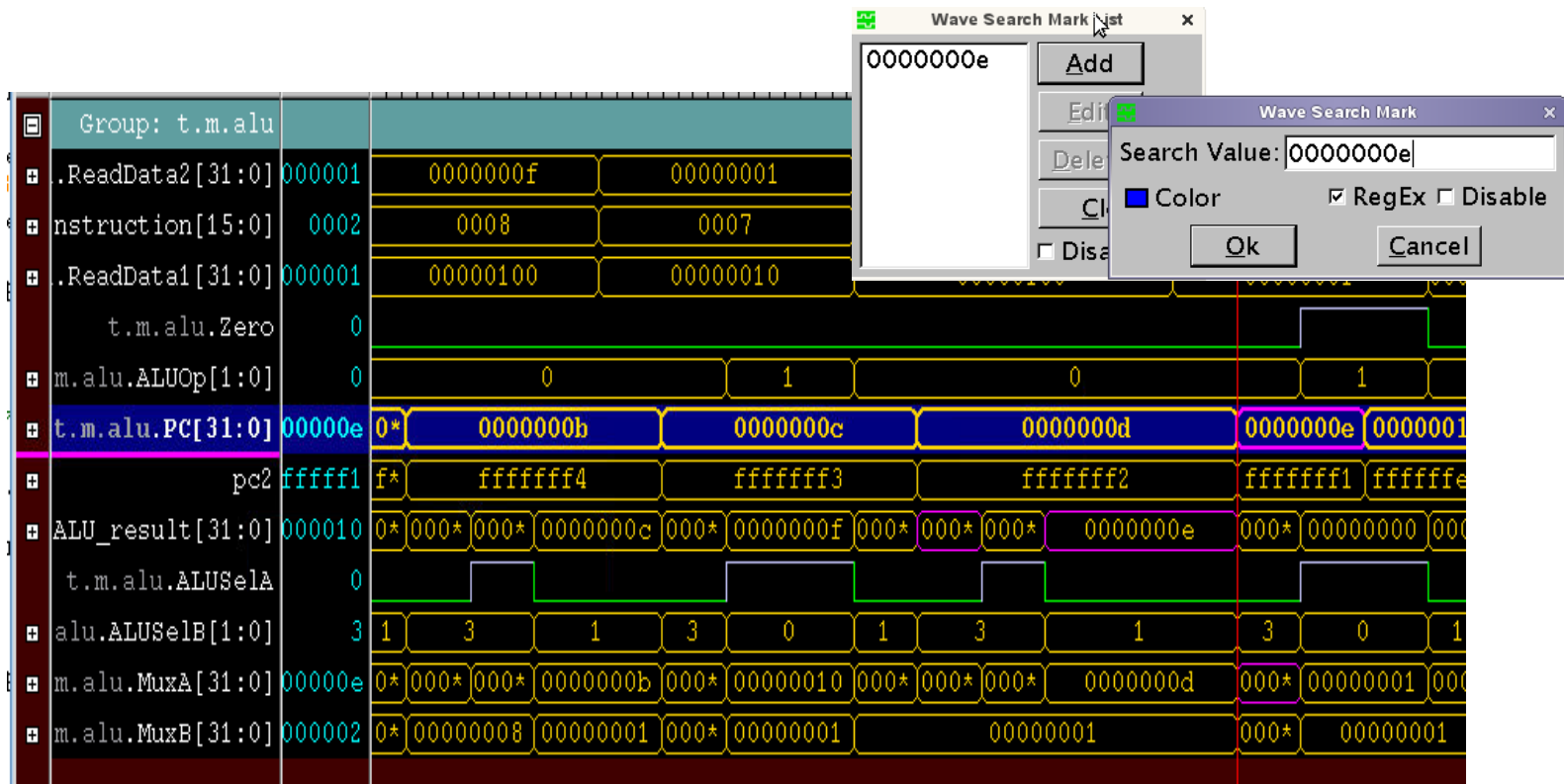
- Trims down to just show the path that is relevant



Different colors denote different simulation times

Searching is Both Easy and Powerful

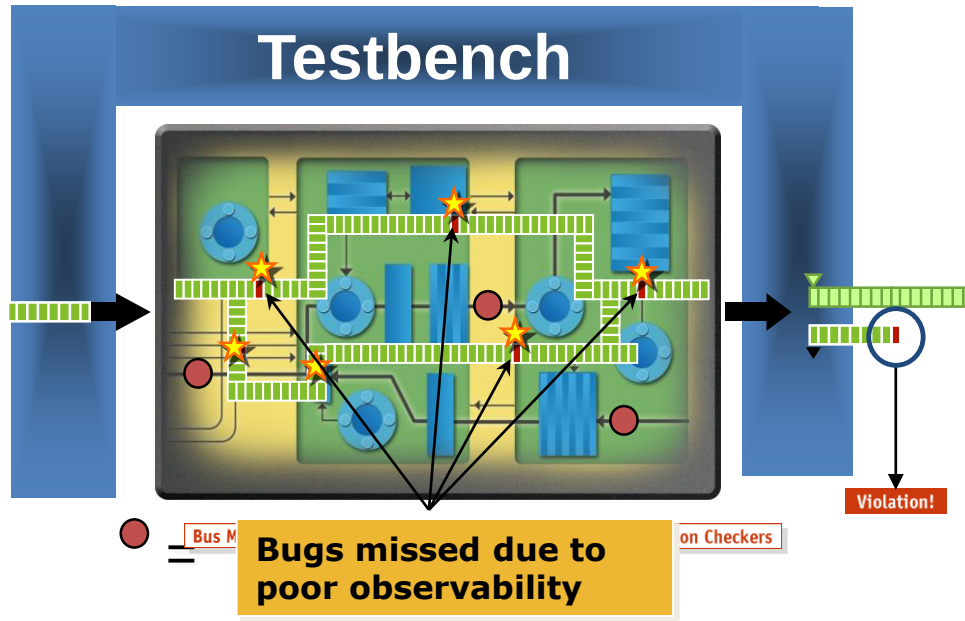
Both design wide and local



- 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



SVA Assertion Window

Module: [All] Scope: top.sva top.mem slave 1 Descendant

Type: [All] Status: [All] DisplayInstance

Prev/Next File:

Module:mem_slave instances(1) total rows = 8

Name	Type	Line #	Explorer	# Failures	# Pass	Inst Name
assert_wdata	assert	102	✓	0	9750	-
assert_hsel	assert	103	✓	0	9750	-
a2	assert	104	✓	0	9750	-
a3	assert	105	✓	0	9749	-
assert_mem1	assert	106	✓	0	9749	-
assert_mem2	assert	107	✓	1494	8254	-
assert_mem3	assert	108	✓	1162	8586	-
P_LDD0Cache0Miss0Adr	assert	128	✓	16063	0	-

Simulation Interactive Result

Assertion Studio

Expression	Value	Cycle#	Time	Failed	Passed	NotFired
hsel_0n > wdata_enb ##32'sd2 ...	-	#0	48000 - 60000	✓		
hsel_0n	-	#0	48000 - 48000		✓	
wdata_enb ##32'sd2 hsel_0n ...	-	#0	48000 - 60000			
wdata_enb ##32'sd2 hsel_0n ...	-	#0	48000 - 60000			
#0 wdata_enb	-	#0	48000 - 48000		✓	
#2 hsel_0n	-	#2	48000 - 60000	✓		

- 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

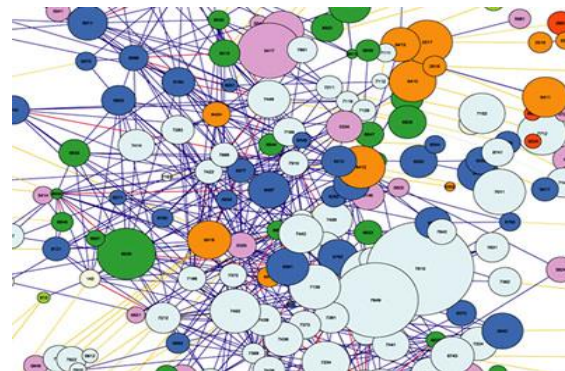
Debugging Complex Testbenches

- Demonstration of example scenarios

Modern Testbenches Add Complexity to Debug

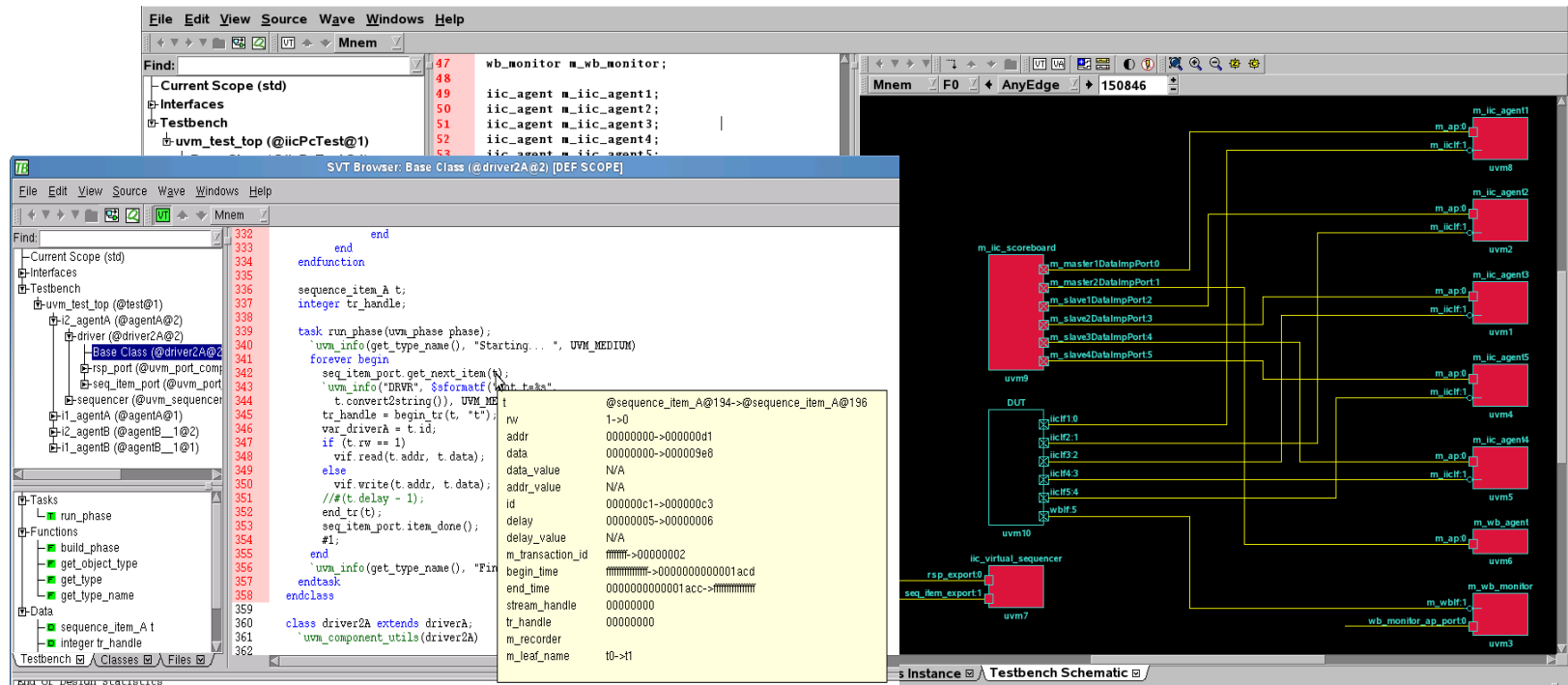
Users report often 50% of debug time is in testbench

- 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

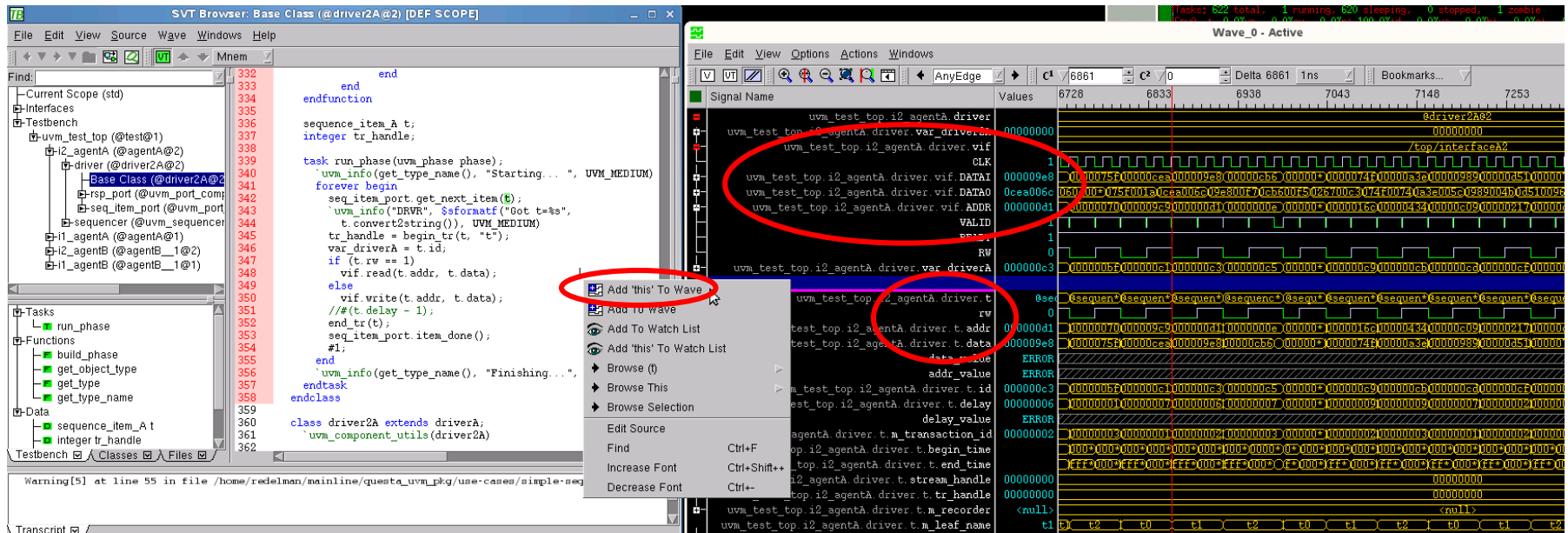
TB debug features normally only found in interactive debug



- 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

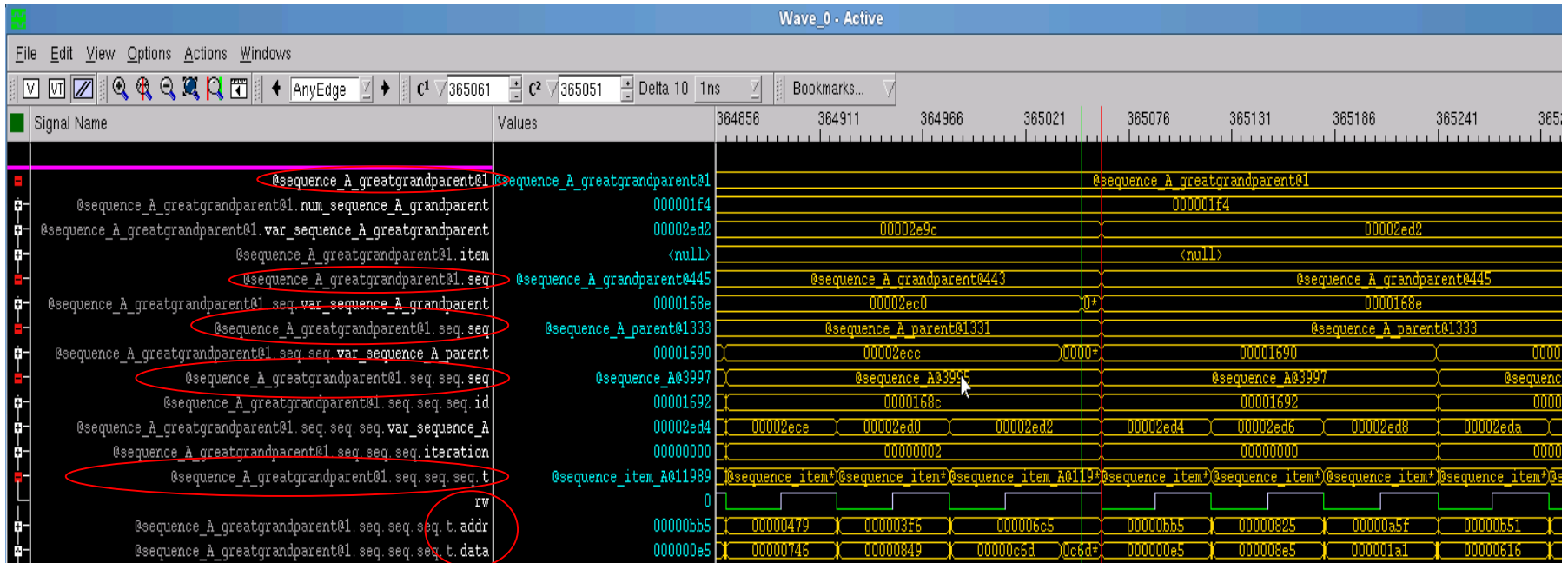
Easily explore data and member values over time



- 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

Stream

- uvm_test_top
 - i1_agentA
 - sequencer
 - ggp_seq_A1**
 - driver
 - i2_agentA
 - i1_agentB
 - i2_agentB

of Parent Transactions = 33502

Time	Stream	name	id
0-1010	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	t0	32'h00000000
0-1160	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	A_seq	32'h00000006
0-1531	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	p_seq	32'h00000004
0-1160	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	A_seq	32'h00000006
1161-1340	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	A_seq	32'h00000008
1341-1520	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	A_seq	32'h0000000a
0-2621	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	gp_seq	32'h00000002
0-1531	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	p_seq	32'h00000004
0-1160	uvm_test_top.i1_agentA.sequencer.ggp_seq_A1	A_seq	32'h00000006

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

The screenshot displays the SVT Browser interface with the file `/home/redeiman/mainline/questa_uvm_pkg/use-cases/simple-sequence/vip_a/agentA.sv` open. The left pane shows the 'Current Scope' and 'Testbench' hierarchy. The main pane shows the source code with a breakpoint set at line 117, `finish_item(t);`. The right pane shows the 'Testbench Scope' table, which lists the dynamic sequence hierarchy.

Testbench Scope	Class Name	Suspend Time	Class ID
uvm_test_top.i2_agentA.sequencer	uvm_sequencer #(class...	-	-
ggp_seq_A2	sequence_A_gratgra...	0 ns	@sequence_A_gratgrandparent@2
ggp_seq_A2_gp_seq	sequence_A_grandpar...	1531 ns	@sequence_A_grandparent@2
ggp_seq_A2_gp_seq_p_seq	sequence_A_parent	1531 ns	@sequence_A_parent@4
ggp_seq_A2_gp_seq_p_seq_A_seq	sequence_A	1581 ns	@sequence_A@8
uvm_test_top.i1_agentA.sequencer	uvm_sequencer #(class...	-	-
ggp_seq_A1	sequence_A_gratgra...	0 ns	@sequence_A_gratgrandparent@1
ggp_seq_A1_gp_seq	sequence_A_grandpar...	1531 ns	@sequence_A_grandparent@1
ggp_seq_A1_gp_seq_p_seq	sequence_A_parent	1531 ns	@sequence_A_parent@3
ggp_seq_A1_gp_seq_p_seq_A_seq	sequence_A	1581 ns	@sequence_A@7
uvm_test_top.i2_agentB.sequencer	uvm_sequencer #(class...	-	-
ggp_seq_B2	sequence_B_gratgra...	0 ns	@sequence_B_gratgrandparent_1
ggp_seq_B2_gp_seq	sequence_B_grandpar...	1531 ns	@sequence_B_grandparent_1@2
ggp_seq_B2_gp_seq_p_seq	sequence_B_parent #(...	1531 ns	@sequence_B_parent_1@4
ggp_seq_B2_gp_seq_p_seq_B_seq	sequence_B #(class s...	1581 ns	@sequence_B_1@6
uvm_test_top.i1_agentB.sequencer	uvm_sequencer #(class...	-	-
ggp_seq_B1	sequence_B_gratgra...	0 ns	@sequence_B_gratgrandparent_1
ggp_seq_B1_gp_seq	sequence_B_grandpar...	1531 ns	@sequence_B_grandparent_1@1
ggp_seq_B1_gp_seq_p_seq	sequence_B_parent #(...	1531 ns	@sequence_B_parent_1@3
ggp_seq_B1_gp_seq_p_seq_B_seq	sequence_B #(class s...	1581 ns	@sequence_B_1@7

- 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

Name	Override	Inst Name
Factory Override		
driverA		
Factory		
agentA		
agentB#(sequence_item_B)		
driver2A	driver2A *	
driverA		
driverB#(sequence_item_B)		
questa_uvm_recorder		
sequence_A		
sequence_A_grandparent		
sequence_A_gratgrandparent		
sequence_A_parent		
sequence_B#(sequence_item_B)		
sequence_B_grandparent#(sequence_item_B)		
sequence_B_gratgrandparent#(sequence_item_B)		
sequence_B_parent#(sequence_item_B)		
sequence_item_A		
sequence_item_B		
test		

Config database – readers, writers and who set what

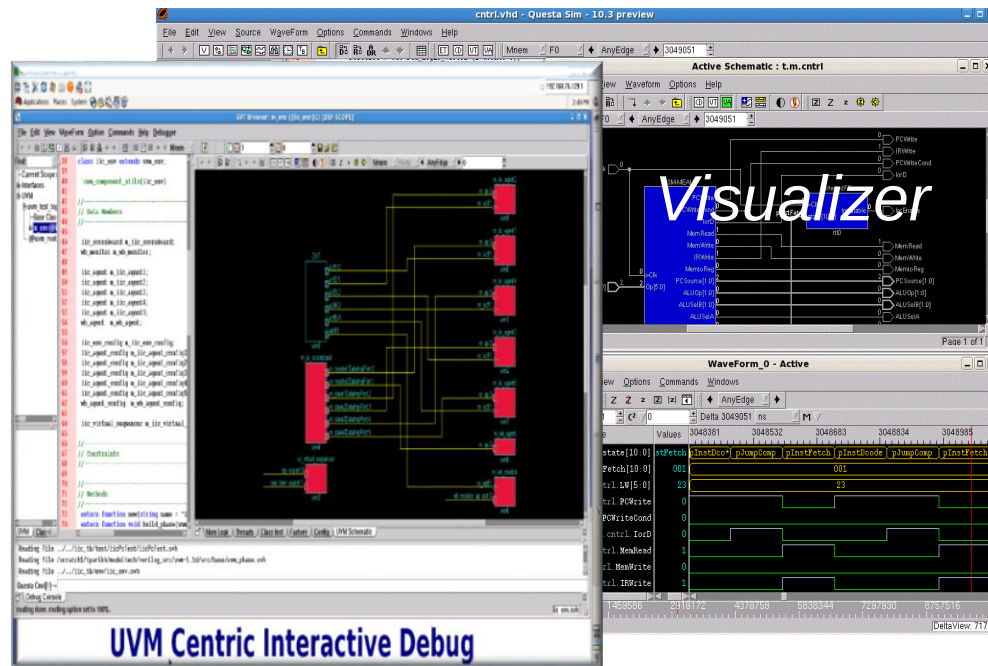
Name	Regex	Value	Write Count	Read Count	Read Only
config_object	/uvm_test_top/i2_agentA.*\$/	(class agentA_pkg:ag...	1	0	0
i1_agentA_vif	/.*\$/	(virtual abc_if /top/inte...	1	1	0
i1_agentB_vif	/.*\$/	(virtual abc_if /top/inte...	1	1	0
i2_agentA_vif	/.*\$/	(virtual abc_if /top/inte...	1	1	0
i2_agentB_vif	/.*\$/	(virtual abc_if /top/inte...	1	1	0
my_int	/uvm_test_top/i2_agentA\driver\$/	(int) 1	1	0	0
my_int	/uvm_test_top/i2_agentA.*\$/	(int) 3	1	0	0
my_string	/uvm_test_top/i2_agentA\driver\$/	(string) Hi	1	0	0
my_test_int	/uvm_test_top/i2_agentA.*\$/	(int) 519	1	0	0
no_matches	/uvm_test_top/i2_agentA.*\$/	(string) None	1	0	0
num_sequence_A...	/uvm_test_top/i2_agentA.*\$/	(int) 1000	1	1	0
recording_detail	/.*\$/	(int) 1	1	0	0
recording_detail		(reg signed[4095:0]) 1	1	73	0
recording_detail		(int) 1	1	0	0
simple_int	/uvm_test_top/i2_agentA\driver\$/	(int) 12	1	1	0
simple_int	/uvm_test_top/i2_agentA.*\$/	(int) 3	1	0	0
stop_barrier	/uvm_test_top/i2_agentA.*\$/	(string) barrier_name	1	0	0
test_override	/uvm_test_top/i2_agentA\driver\$/	(string) Only_matches	1	0	0
test_override	/uvm_test_top/i2_agentA.*\$/	(string) Only_overrides	1	0	0

- 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

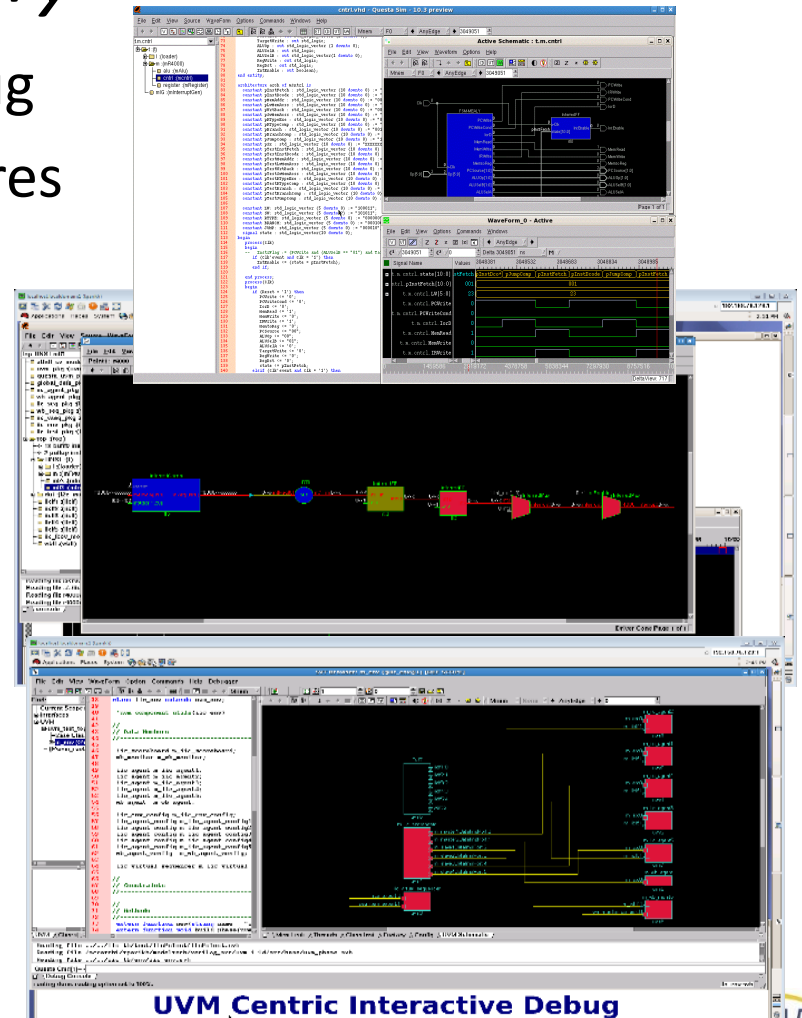
Questa®



- 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

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



UVM Centric Interactive Debug

Agenda

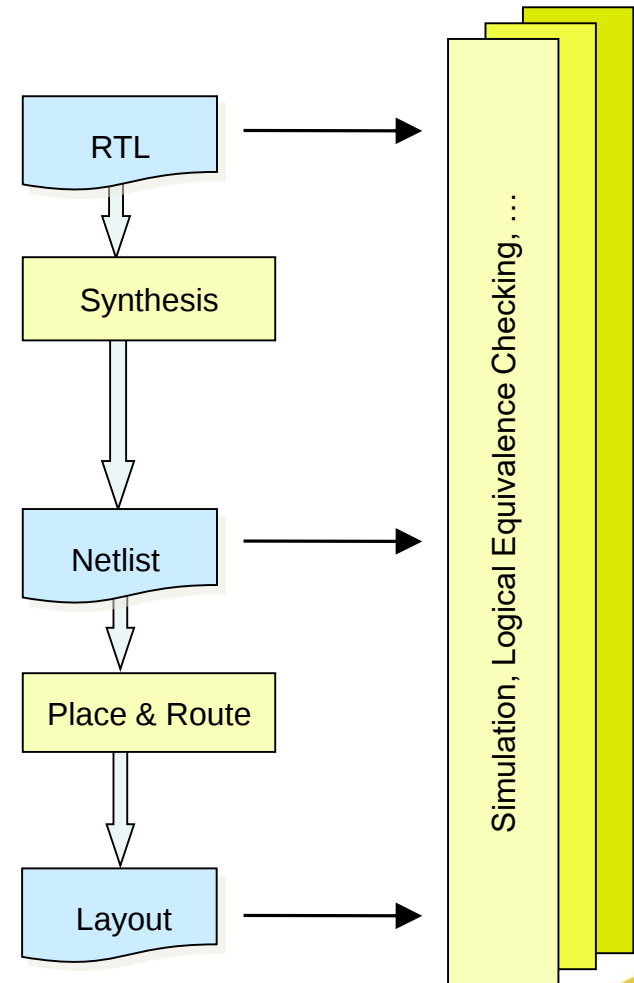
- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- **Power-aware Verification Debug**
- Integrated Hardware/Software Debug
- Post-Silicon Debug Solutions

Debugging Low Power Designs

- Demonstration of example scenarios

Traditional Design and Verification Flow

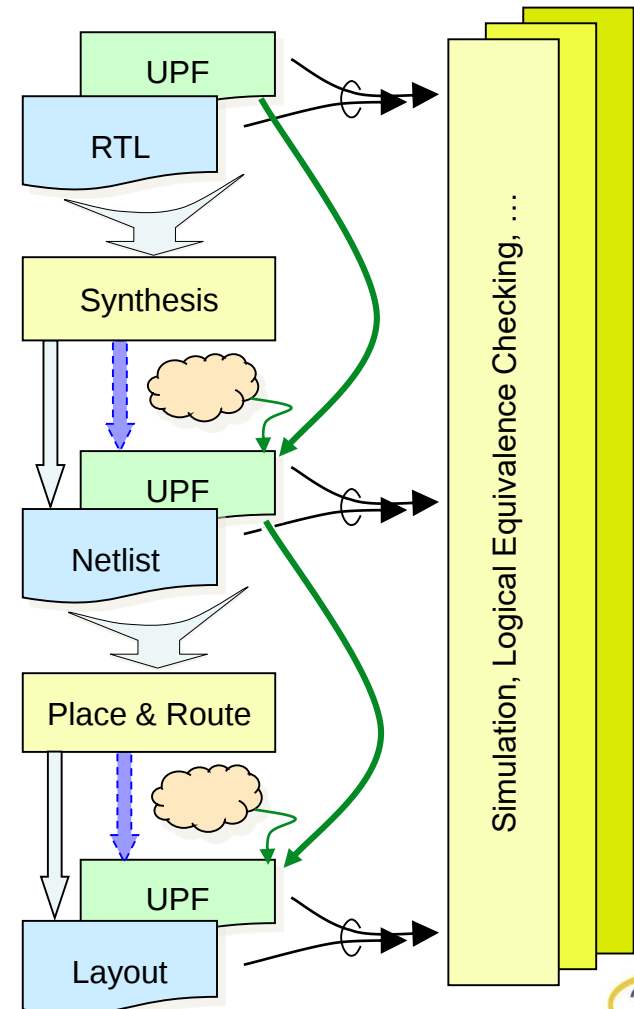
- RTL design
 - Captures design intent
 - Drives functional verification
 - Drives synthesis
- Logical implementation
 - Using standard cells and macros
 - Further modified for test, ECOs, etc.
- Physical implementation
 - Place & Route completes implementation
 - Produces manufacturing data
- Power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



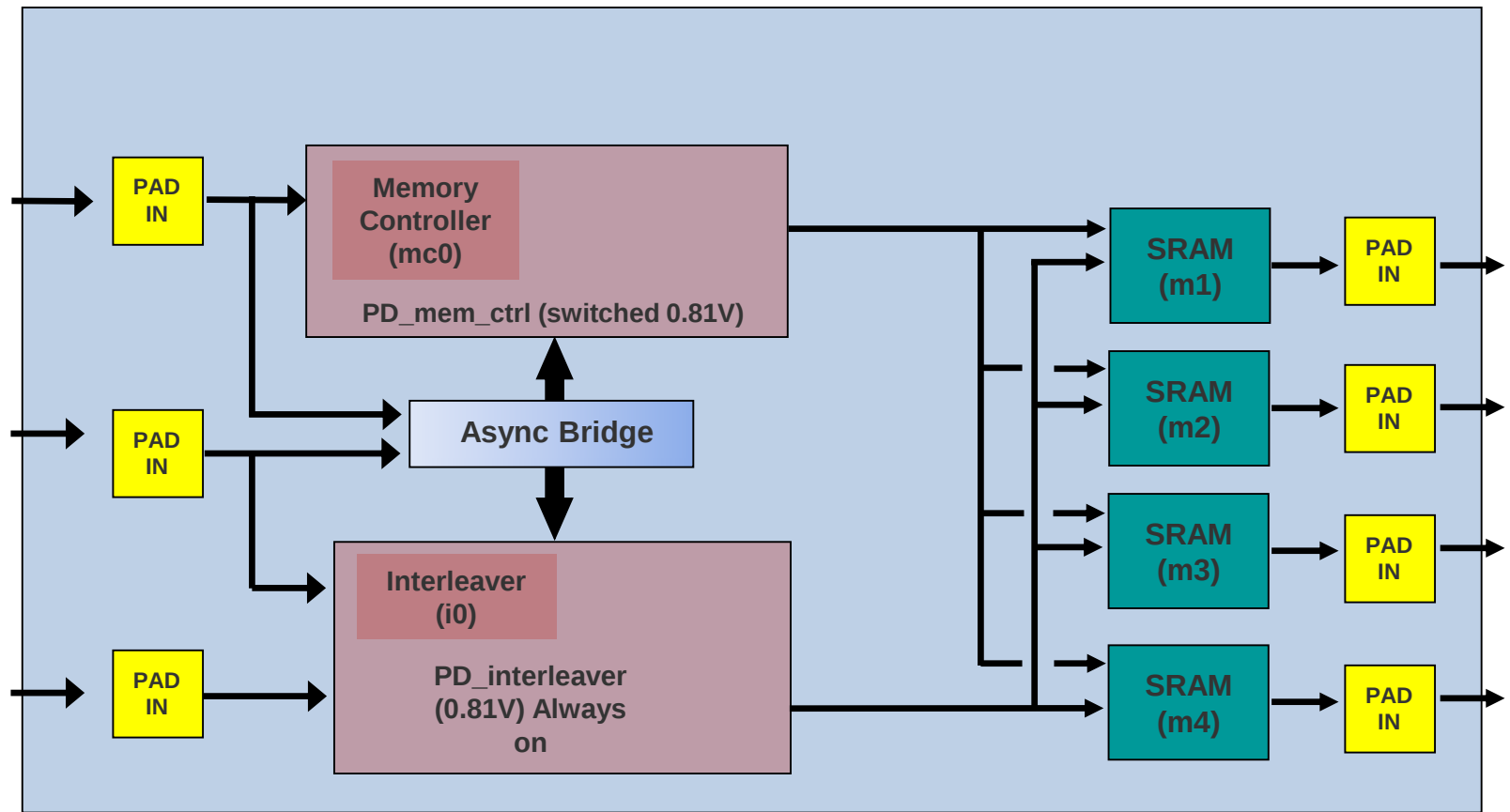
Can we verify power management earlier in the flow?

UPF-Based Design and Verification Flow

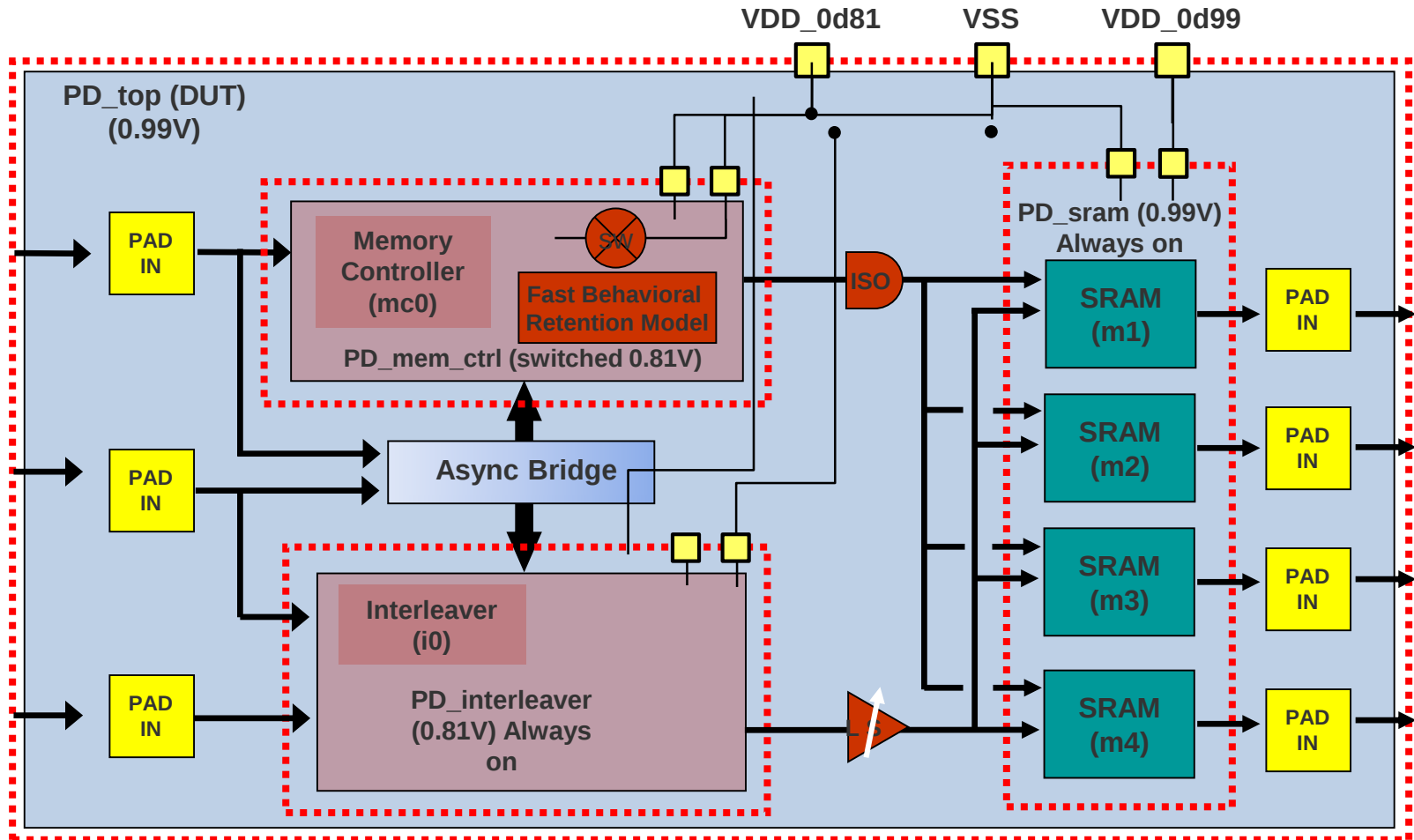
- RTL is augmented with UPF
 - To define power management architecture
- RTL + UPF verification
 - To ensure that power architecture completely supports planned power states of design
 - To ensure that design works correctly under power management
- RTL + UPF implementation
 - Synthesis, test insertion, 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

Design Functionality

Verify that design functions correctly with power always on

Questa PAsim

Power Management Architecture

Verify that power management architecture is correct

Power Domain Behavior

Verify power up/down and reset/restore on power up for each block/power domains

Power Domain Interactions

Verify that interfaces are correctly isolated and level shifted

Power State Transitions

Verify that all transitions are covered and behave correctly with power control

Power Control Hardware

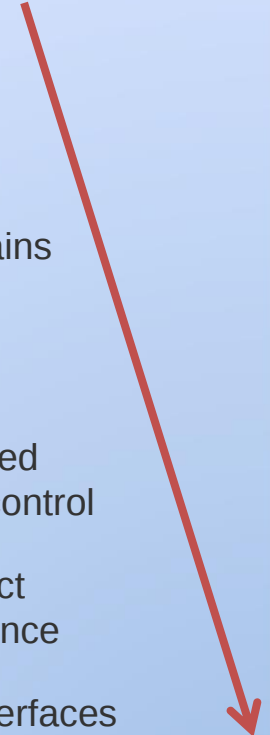
Verify that HW generates correct control signals in correct sequence

Power Control Software

Verify SW Power Control interfaces with HW is correct

System with Power Management

Verify full system with HW and SW for power control



Questa Formal

Questa Codelink

Veloce Emulation

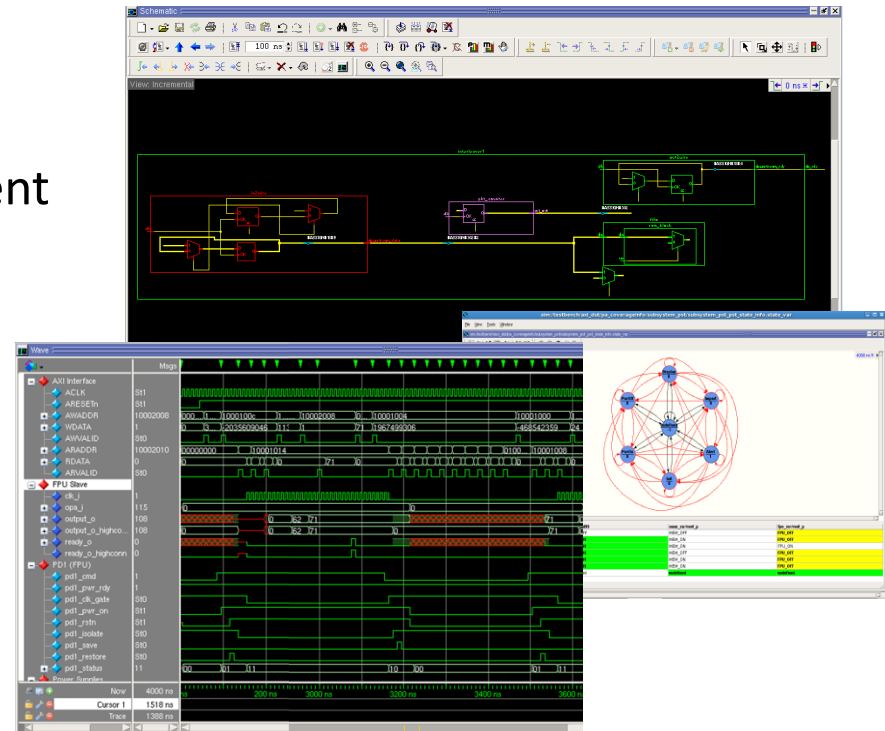
Questa Power Aware Simulation



IEEE Std 1801™-2009



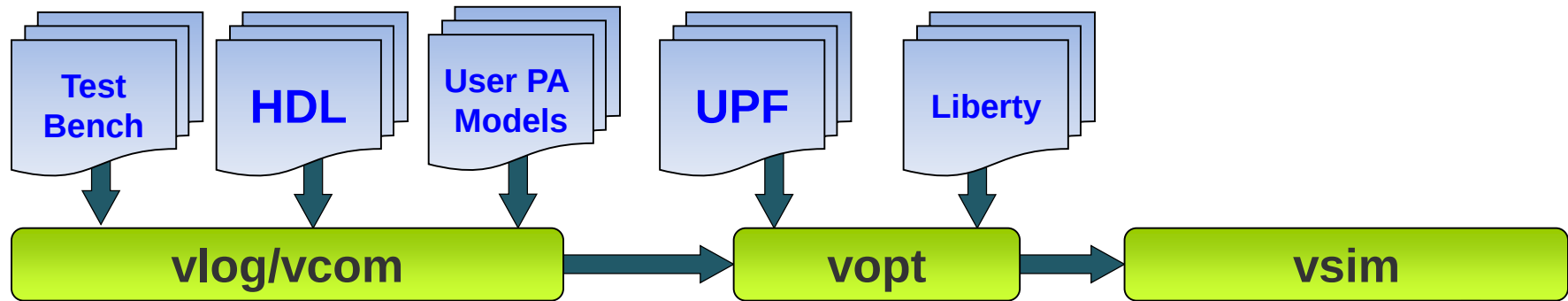
- Visualization of power management structure and behavior
- Automatic detection of power management errors
- Automatic power management coverage
- Automatic test plan generation
- High performance and efficiency
- Extensive IEEE 1801 UPF support



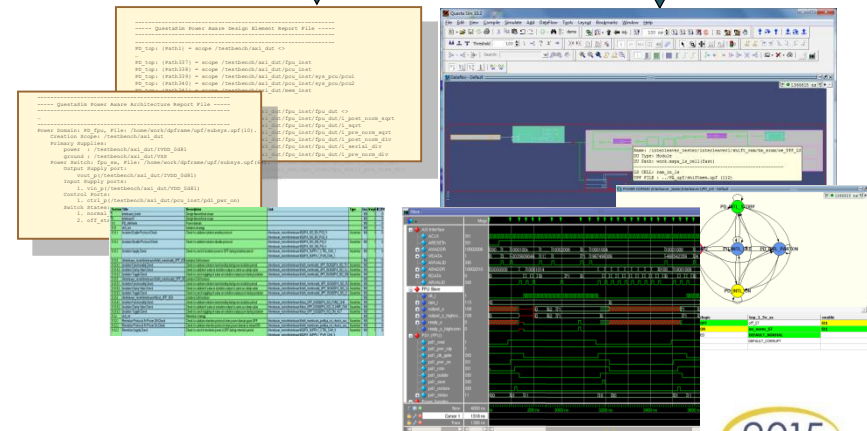
Enables power management architecture verification
with RTL and Gate-Level power aware simulation

Questa PAsim Native Simulation of RTL+UPF

Just a command-line switch on normal Questa simulation



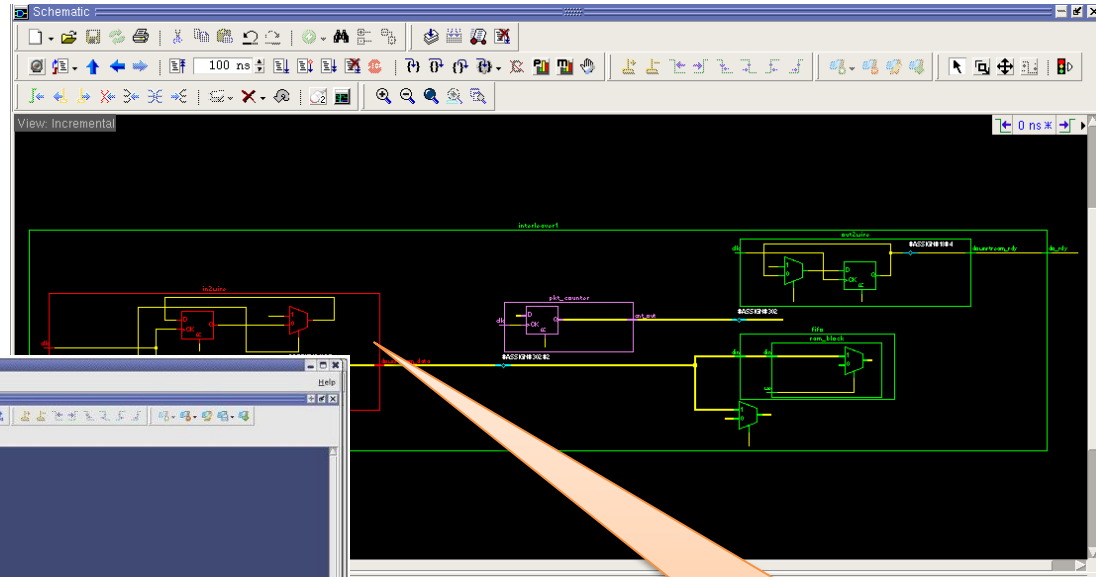
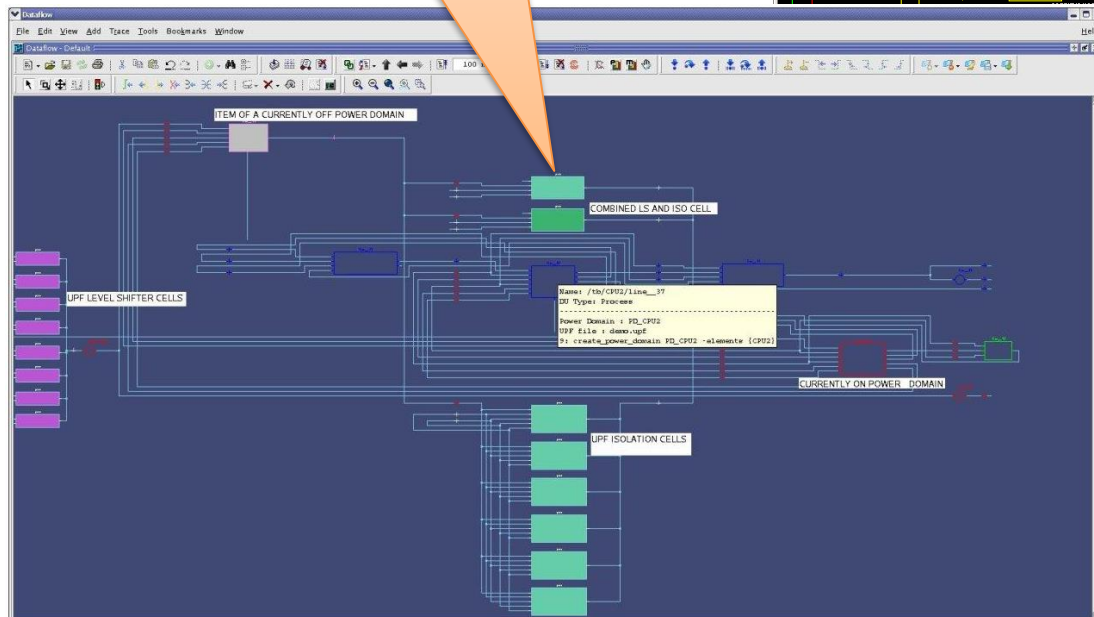
- Compile as usual
 - No change in source code
- Optimize with PA options
 - Process UPF power intent
 - Read Liberty libraries as needed
 - Run static PA checks
- Simulate with PA options
 - Generate reports/Testplan
 - Visualize/debug power-managed behavior



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

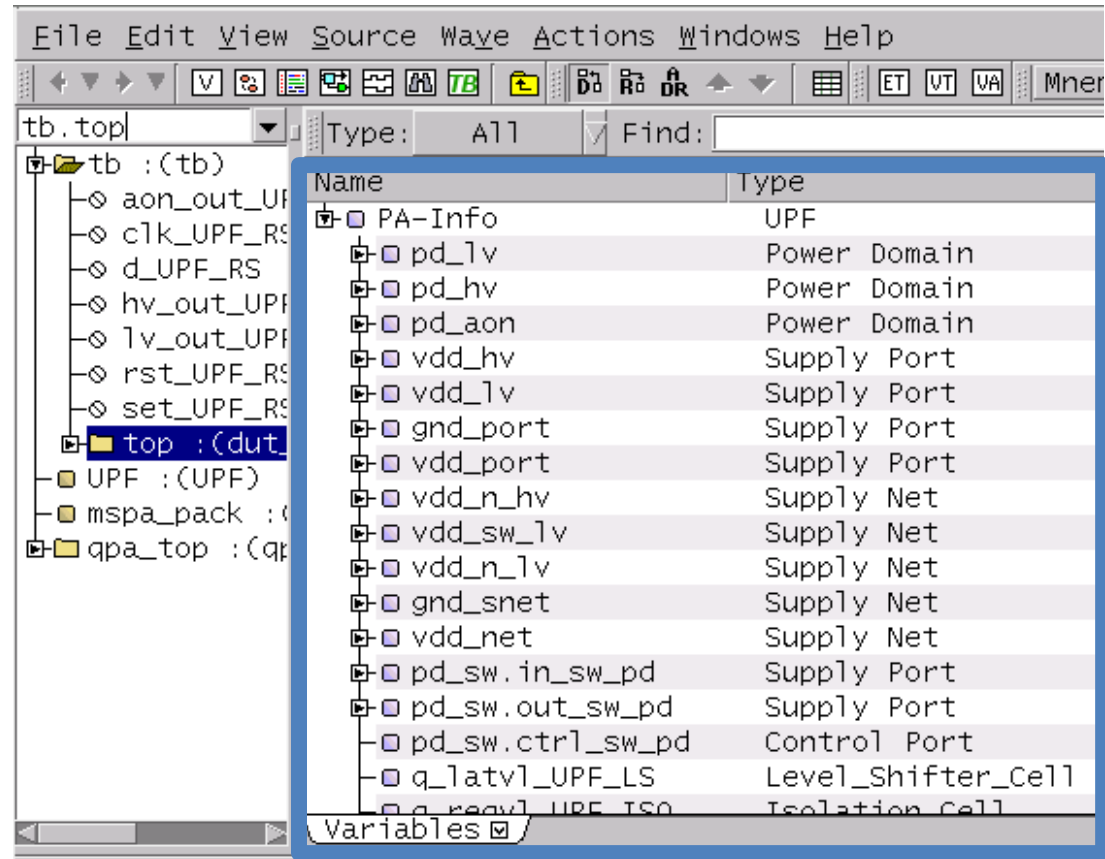


Power Domain and its instances is displayed in a unique color in structural view

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

Type	Name	Line	State	Info
Power Domain	PCI	48	1	HierPath: /tbleon/tb/leon0/PCI
Creation Scope	Leon0	3	-	
Supply Set	primary	48	-	
Switch	PCI_SWITCH	84	-	Creation Scope: /tbleon/tb/leon0
Level Shifter	PCI_LH	184	-	Rule (low_to_high), Threshold (0), Applie
Level Shifter	PCI_HL	186	-	Rule (high_to_low), Threshold (0), Applie
Isolation	PCI_ISO_0	98	-	Isolation Sense (LOW), Clamp Value (0), L
Isolation	PCI_ISO_1	113	-	Isolation Sense (LOW), Clamp Value (1), L
Isolated Ports				
Isolation Enable	tbleon.tb...	33	1	
Scopes/Extents				
Power Domain	SPARC_T0PISO	58	1	HierPath: /tbleon/tb/leon0/SPARC_T0PISO
Power Domain	TOP	21	1	HierPath: /tbleon/tb/leon0/TOP
Power Domain	FPU	63	1	HierPath: /tbleon/tb/leon0/mcore0/proc0/I
Power Domain	IU	52	1	HierPath: /tbleon/tb/leon0/mcore0/proc0/I
Power Domain	SPARC	31	1	HierPath: /tbleon/tb/leon0/mcore0/proc0/I
Supply Nets				
Power Switch				
Switch	PCI_SWI		-	Creation Scope: /tbleon/tb/leon0
Power Domain	SPARC_T		1	HierPath: /tbleon/tb/leon0/SPARC_T0PISO
Power Domain	TOP		1	HierPath: /tbleon/tb/leon0/TOP
Power Domain	FPU		1	HierPath: /tbleon/tb/leon0/mcore0/proc0
Power Domain	IU		1	HierPath: /tbleon/tb/leon0/mcore0/proc0
Power Domain	SPARC		1	HierPath: /tbleon/tb/leon0/mcore0/proc0
Power Switch				
Switch	PCI_SWITCH	84	-	Creation Scope: /tbleon/tb/leon0
Switch	FPU_SWITCH	113	-	Creation Scope: /tbleon/tb/leon0/mcore0
Switch	SPARC_SWITCH	99	-	Creation Scope: /tbleon/tb/leon0/mcore0

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

Entire Design Scope: Source: tbleon Sink: tbleon Recursive

Source: Sink: Type: State:

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 Domain	TOP	1	SPARC_TOPISO	1	Source - Sink Power ..

Power Aware Dynamic Messages

Interactively Debug Errors from Automatic Checks

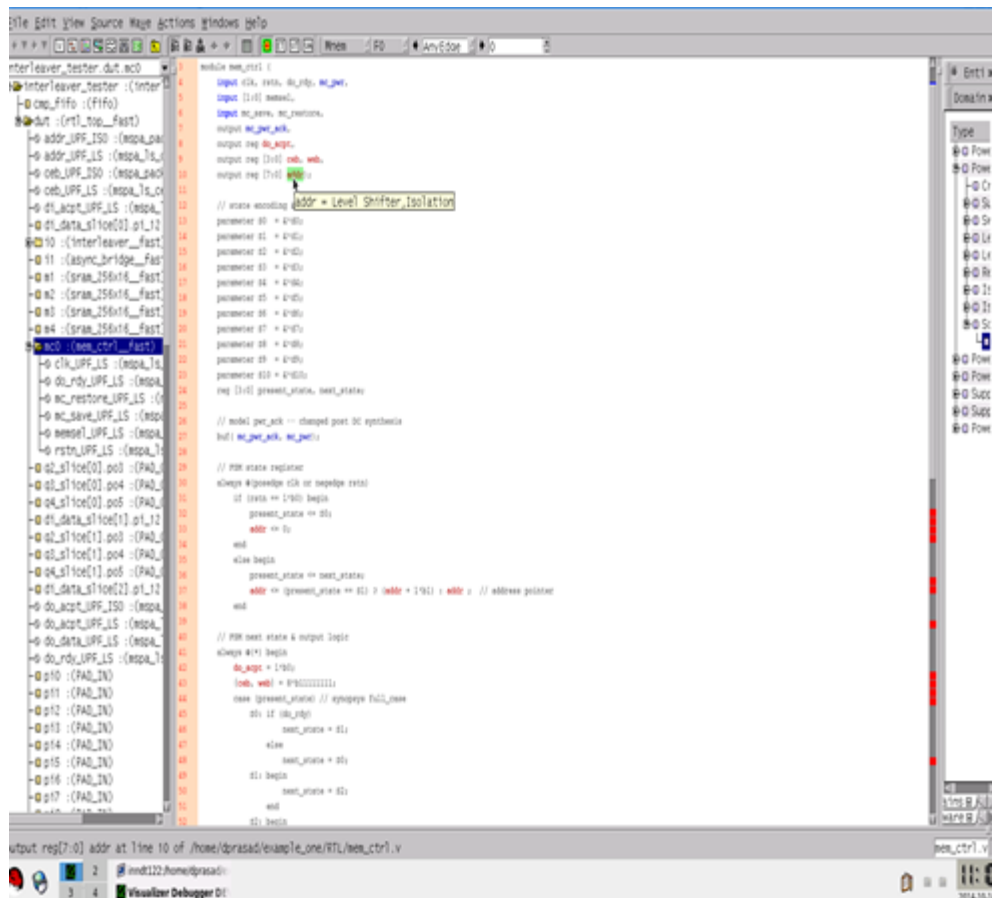
- View and debug all PA Dynamic Messages
- View time at which message
– Synced with waveforms
- Add signals/domains to waveform
- Filtering based on message t

Type	PD	State	Source Port	PD	State	Sink Port	Info
MSPA_UPF_MISSING_LS_CHK							
MSPA_UPF_MISSING_ISO_CHK							
MSPA_UPF_MISSING_ISO_CHK	pd_lv	1	q_latv1	pd_aon	1	q_latv1	...
203 ns							
363 ns							
483 ns							
603 ns							
803 ns							
1003 ns							
MSPA_UPF_PG_CHK							
MSPA_RET_OFF_PSO							
MSPA_ISO_DIS_PG							
MSPA_ISO_ON_ACT							
MSPA_ISO_CLAMP_CHK							
MSPA_RET_PD_OFF							
MSPA_ISO_EN_PSO							
MSPA_UPF_ILLEGAL_STATE_REACHED							
MSPA_ISO_PORT_TOGGLE							

HDL Annotation

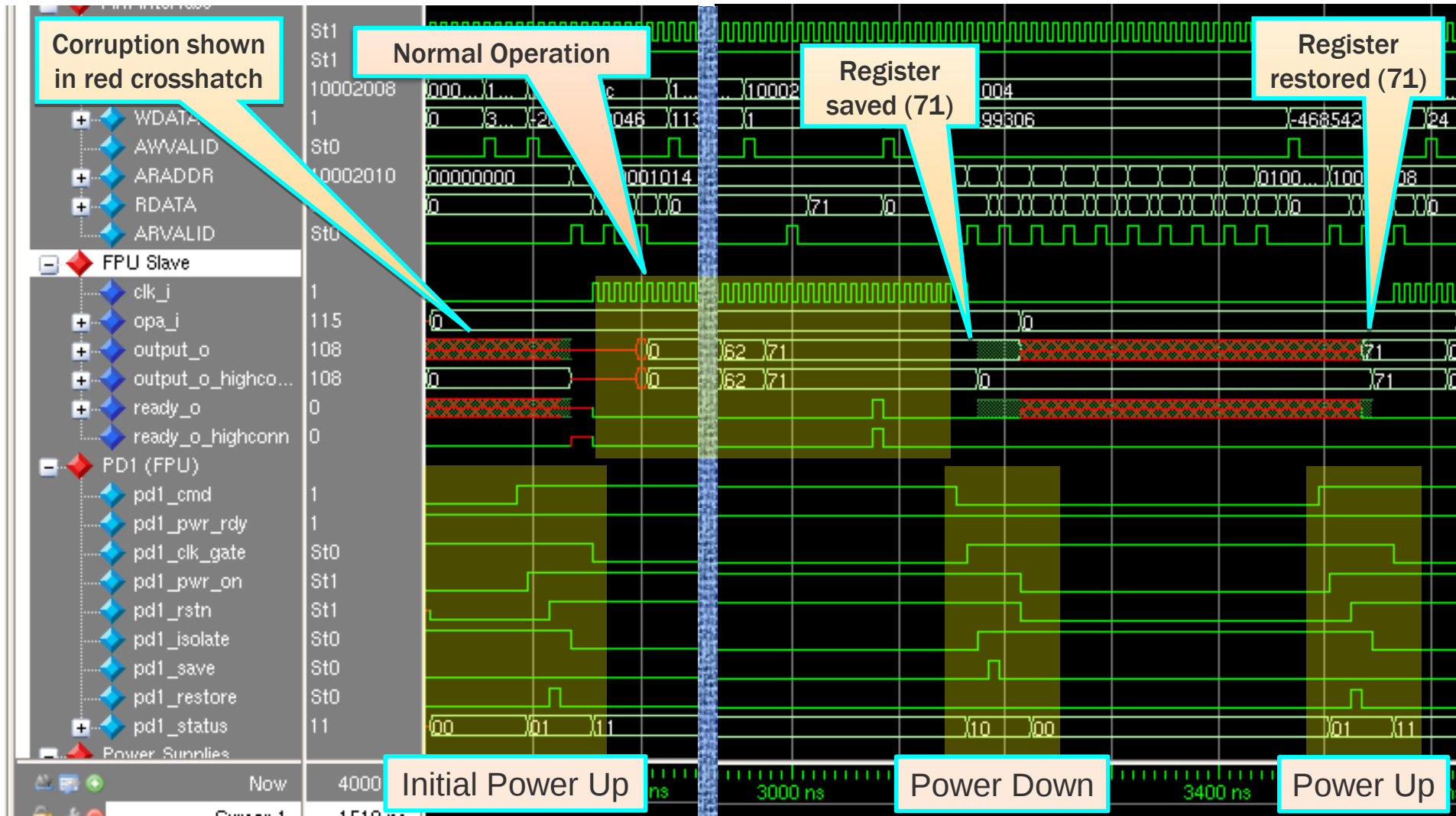
View all UPF added information in your HDL Source Context

- 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

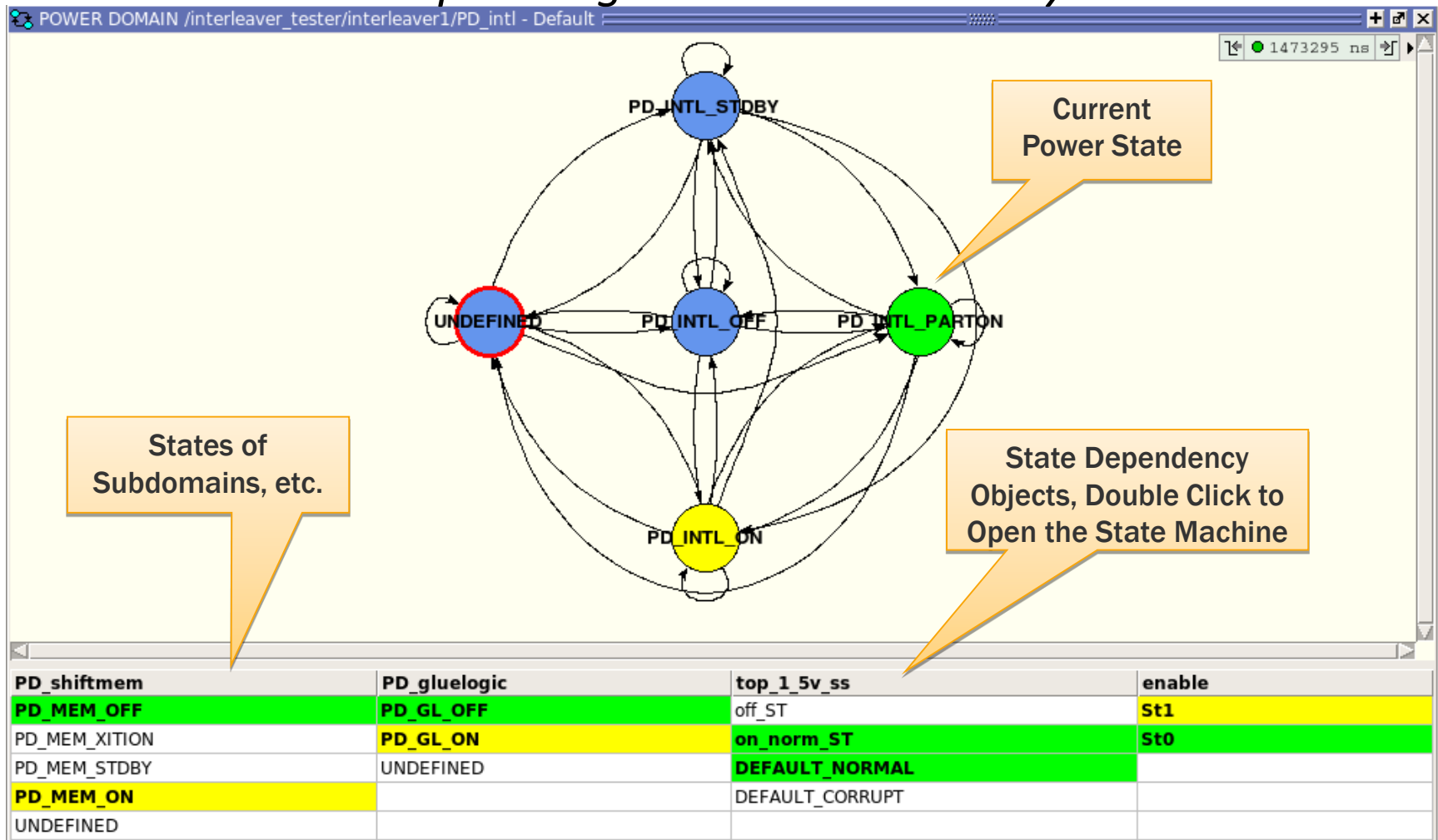
- Level shifters are inserted where required by power state definitions
- Level shifters shift in correct direction
- Level shifters correctly handle dynamic signal behavior

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

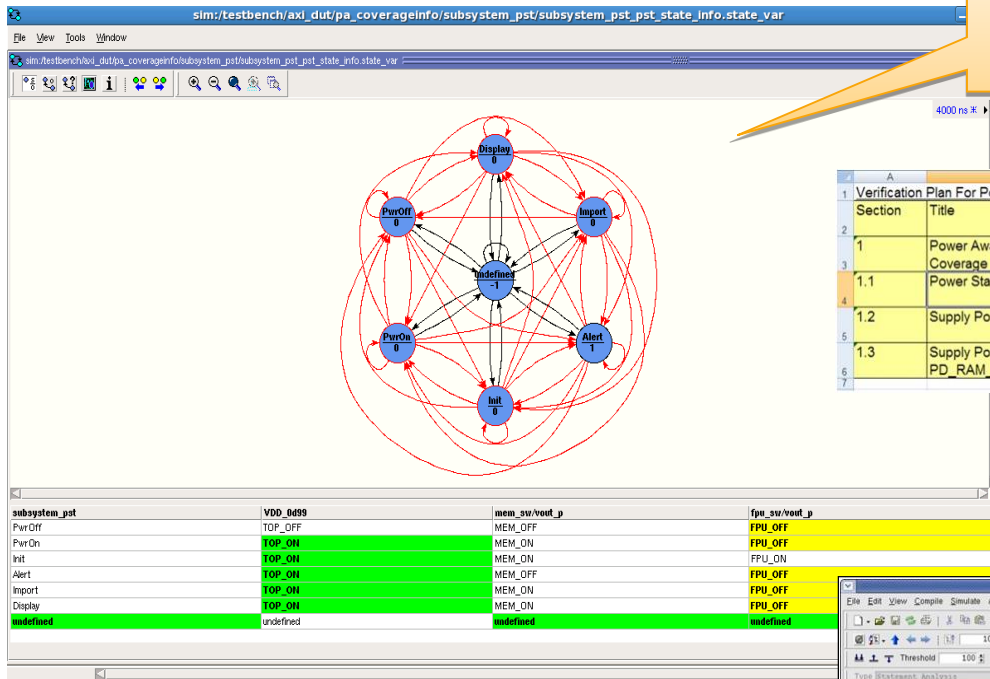
Step through states interactively



Automatic Power State Coverage

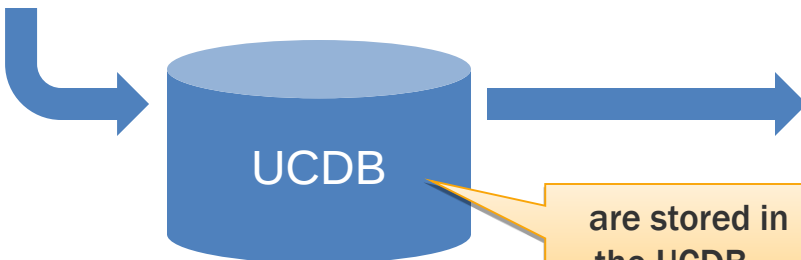
Coverage Information Created Directly from UPF Power States

Power states and transitions ...



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/MyPowerStateTable	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 ...



are stored in the UCDB ...

Type	Statement	Analysis	Coverage	Coverage graph	Goal	% of Goal	Weight	Unlinked Secs
testplan	Testplan		52.66%		100%	52.66%	1	No 0
testplan	Power Aware State Coverage		52.66%		100%	52.66%	1	No 1
testplan	Supply Port PD_ALU_PORT		48%		100%	48.00%	1	No 1.2
testplan	/tb/pa_coverageinfo/PD_ALU_sw/out_sw_PD_ALU/port_state_info_state_var		48%		100%	48.00%	1	No
testplan	Supply Port PD_RAM_PORT		48%		100%	48.00%	1	No 1.3
testplan	/tb/pa_coverageinfo/PD_RAM_sw/out_sw_PD_RAM/port_state_info_state_var		48%		100%	48.00%	1	No
testplan	Power State Table		62%		100%	62.00%	1	No 1.1
testplan	/tb/pa_coverageinfo/MyPowerStateTable/MyPowerStateTable_pst_state_info_state_var		62%		100%	62.00%	1	No

... for coverage closure.

5

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

Links to each PA coverage item

Section	Testplan	Weight	Coverage
1.1.1.1	tester/interleaver/QSPA_ISO_EN_PSD_5	100	1
1.1.1.1	tester/interleaver/QSPA_ISO_EN_PSD_4	100	1
1.1.1.1	tester/interleaver/QSPA_ISO_DIS_PG_5	100	1
1.1.1.1	tester/interleaver/QSPA_ISO_DIS_PG_4	100	1
1.1.1.1	tester/interleaver/QSPA_SUPPLY_CTRL_CHK_1	100	1
1.1.1.1	tester/interleaver/QSPA_SUPPLY_PWR_CHK_1	100	1
1.1.1.1	tester/interleaver/shift_mem/raddr_UPF_ISO/QSPA_ISO_FU	100	1
1.1.1.1	tester/interleaver/shift_mem/raddr_UPF_ISO/QSPA_ISO_CL	100	1
1.1.1.1	tester/interleaver/shift_mem/raddr_UPF_ISO/QSPA_ISO_ON	100	1
1.1.1.1	tester/interleaver/shift_mem/waddr_UPF_ISO/QSPA_ISO_FU	100	1
1.1.1.1	tester/interleaver/shift_mem/waddr_UPF_ISO/QSPA_ISO_CL	100	1
1.1.1.1	tester/interleaver/shift_mem/waddr_UPF_ISO/QSPA_ISO_O	100	1
1.1.1.1	tester/interleaver/dout_UPF_ISO/QSPA_ISO_FUNC_CHK	100	1
1.1.1.1	tester/interleaver/dout_UPF_ISO/QSPA_ISO_CLAMP_CHK	100	1
1.1.1.1	tester/interleaver/dout_UPF_ISO/QSPA_ISO_ON_ACT	100	1
1.1.1.1	tester/interleaver/shift_mem/sram_perif/pa_ret_checks_vec	100	1
1.1.1.1	tester/interleaver/shift_mem/sram_perif/pa_ret_checks_vec	100	1
1.1.1.1	tester/interleaver/QSPA_SUPPLY_CTRL_CHK_0	100	1
1.1.1.1	tester/interleaver/QSPA_SUPPLY_PWR_CHK_0	100	1

Questa Testplan Tracking Excel Addin
Press F1 for add-in help.

47

accutera
SYSTEMS INITIATIVE

EUROPE

Low Power Coverage Closure Report

- Tracks power related coverage data
 - Links PA testplan items with PA coverage items in ucdb
 - Used standard testplan structure in top

Track coverage of PA
State and Transition

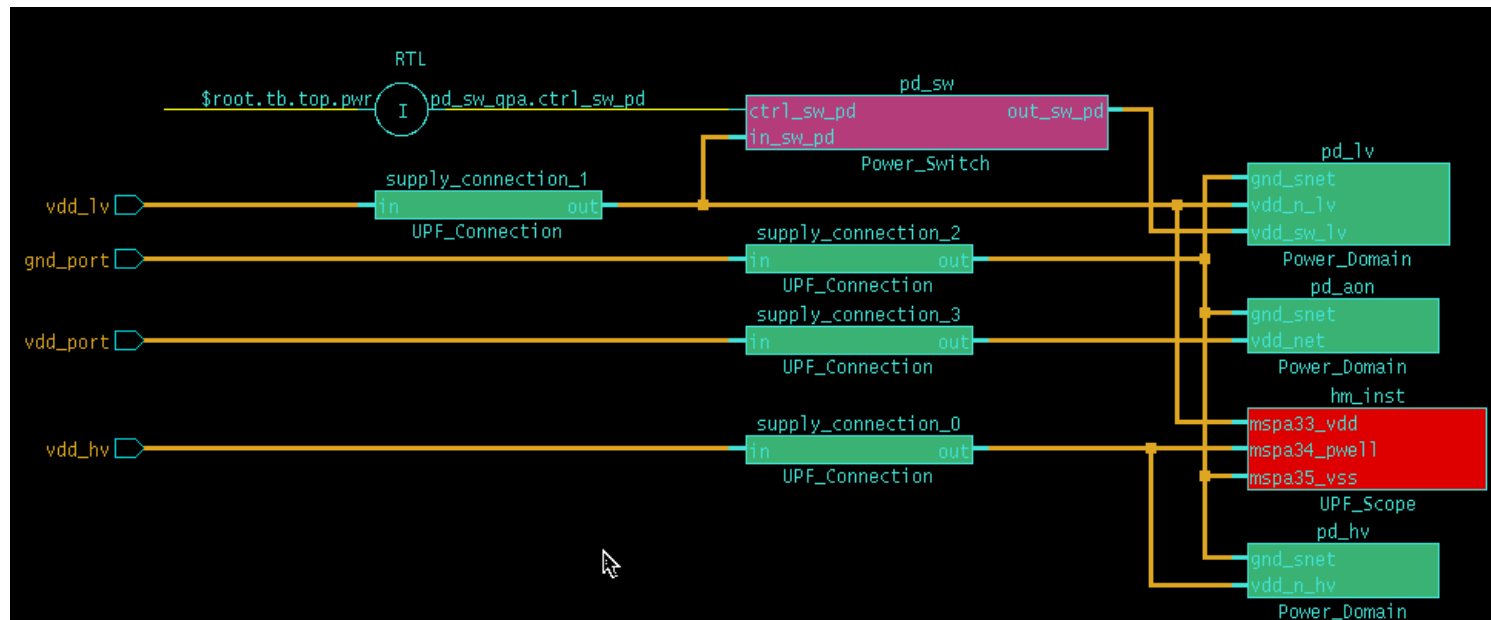
Track coverage of PA
enable assertion passes

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

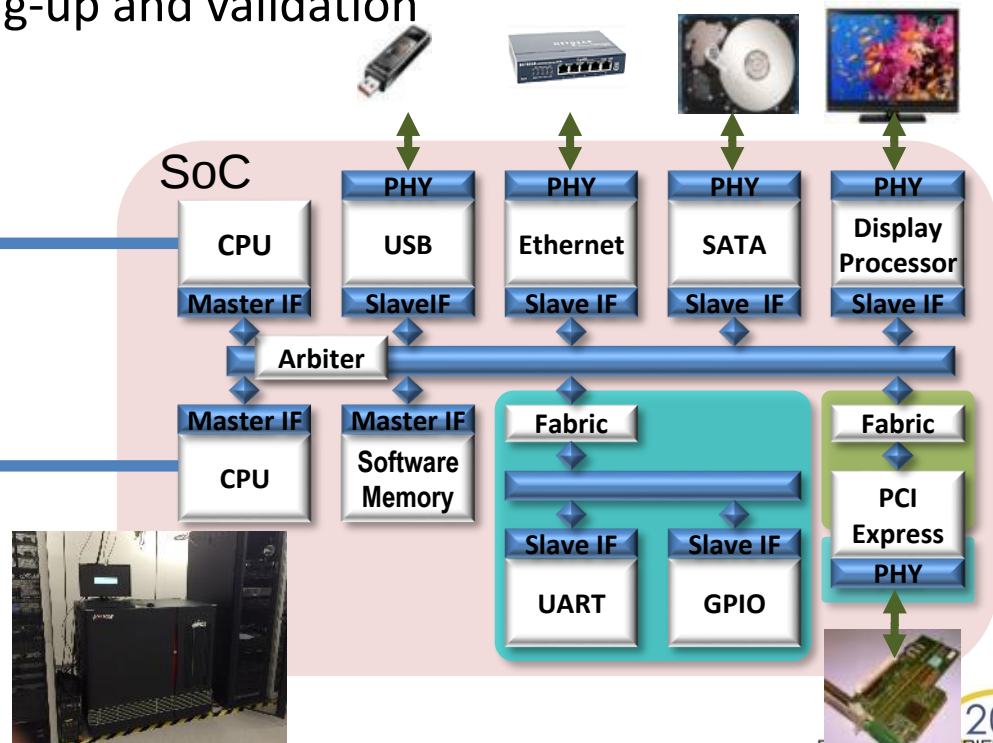
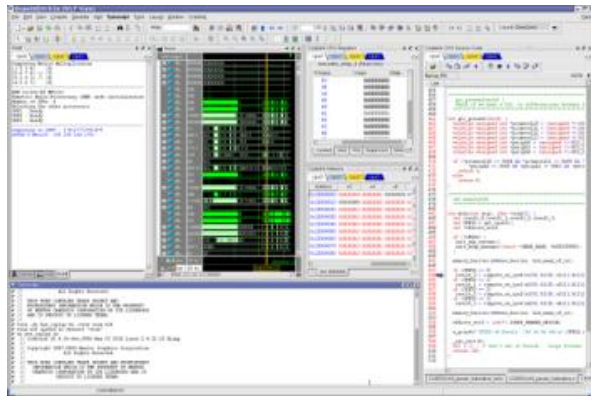
Agenda

- 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

***VELOCE*2+ OS3**



**In Circuit Emulation
(ICE)**



**Virtual Peripheral
& Transactors**

**The Best of Both
Worlds**

A Door to real traffic

- No testbench creation
- Near real speed emulation



Quattro



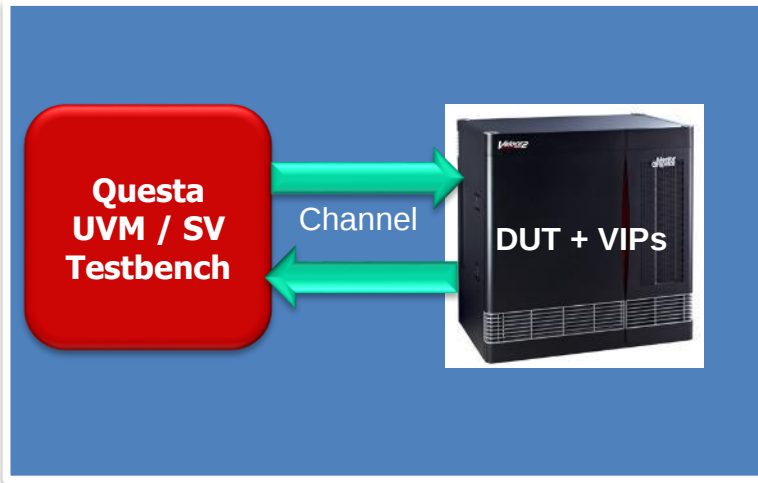
Maximus

TBX the pathway to a spectrum of verification possibilities

- Predictable/repeatable behavior
- No speed compromise

Versatile and Easy to Use Verification and Validation Solution

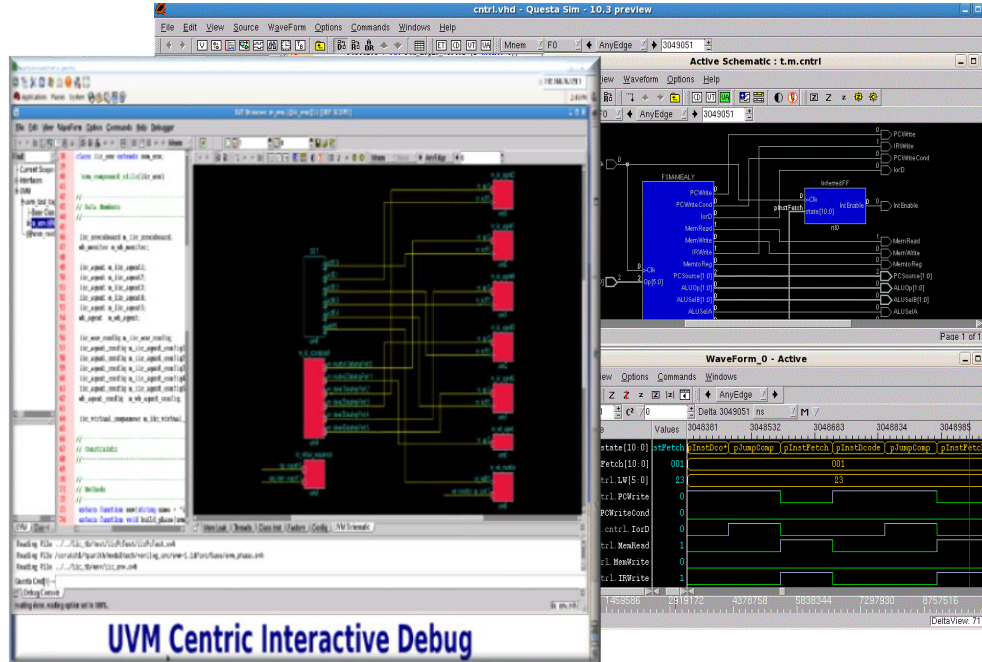
Advanced Methodologies – UVM/ SV



- 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



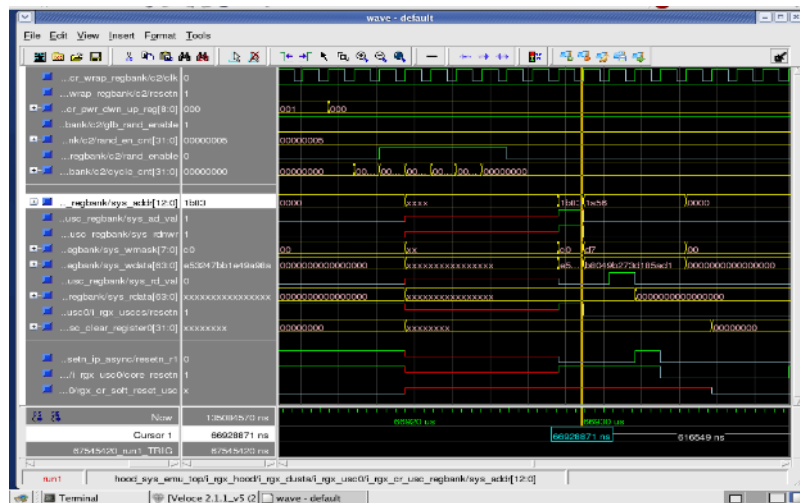
Maximize performance and still have ease of use



- 

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



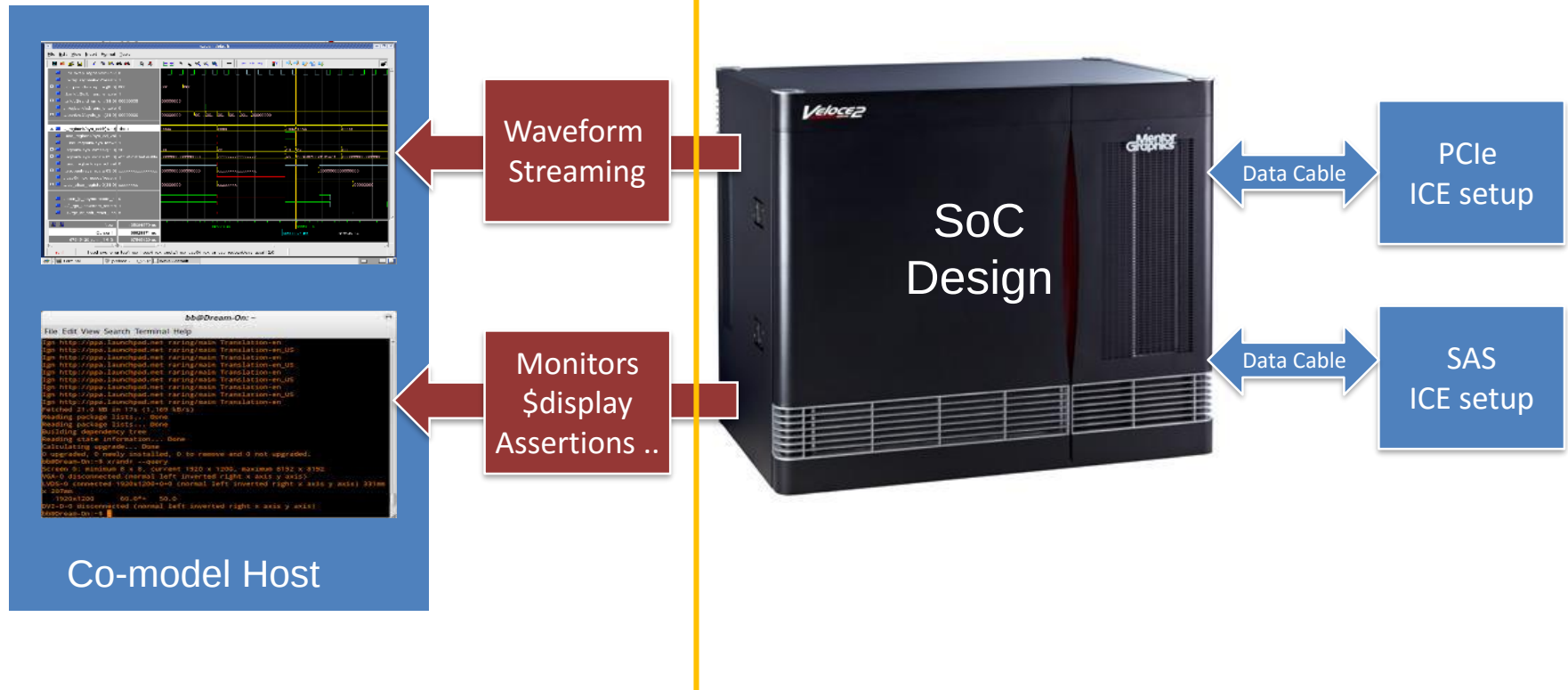
TBX Streaming



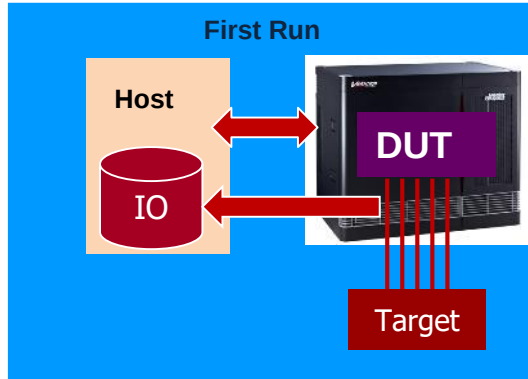
Veloce Advanced Debug for ICE

Advanced Debug

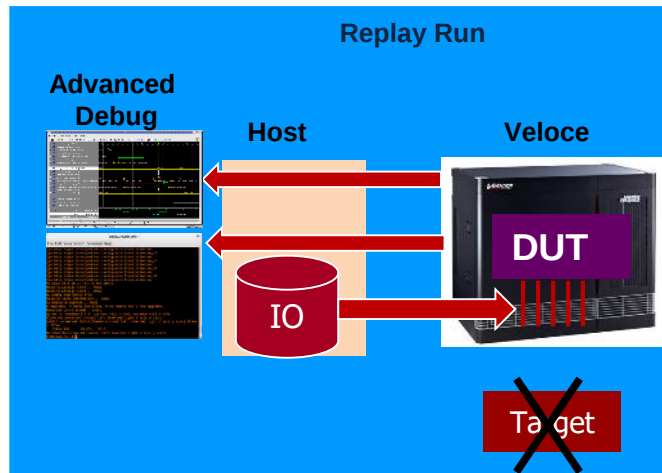
Standard ICE Setup



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**

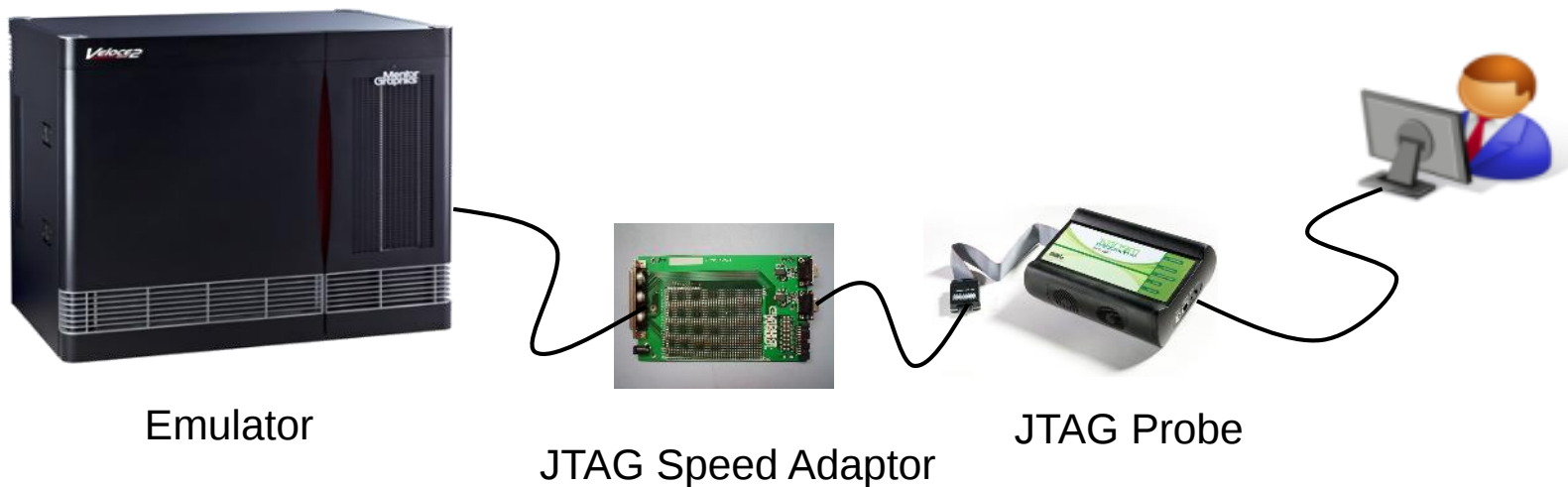
Goals of New HW Debug Paradigm

- 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

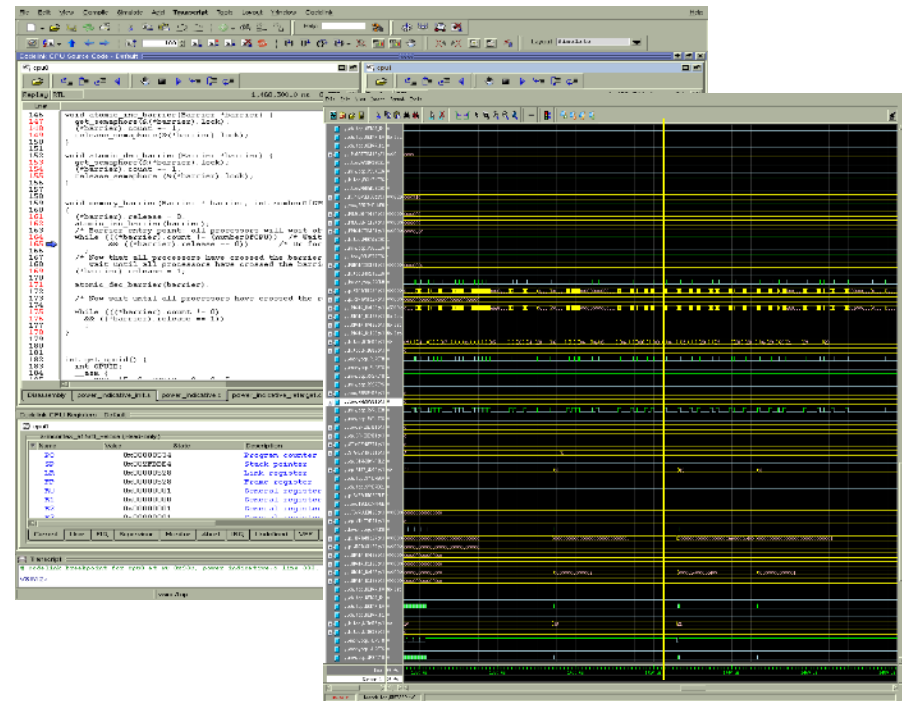
New SW Debug Methods

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



Veloce New Software Debug Solutions

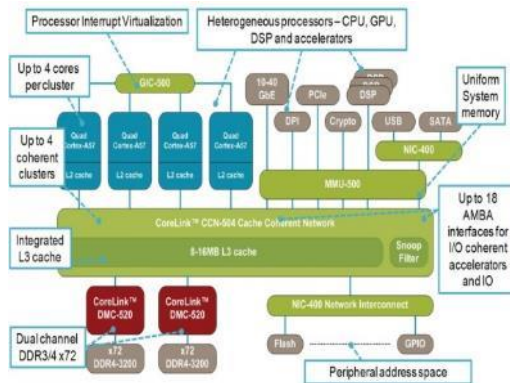
- Virtual Probes
 - Traditional interactive debug using JTAG transactors
 - No physical JTAG probe needed – all virtual
 - Full software debug capabilities
- Codelink
 - Off-line post emulation debug
 - Enables efficient sharing of emulation resource
 - Full software debug capabilities
- Warpcore
 - 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

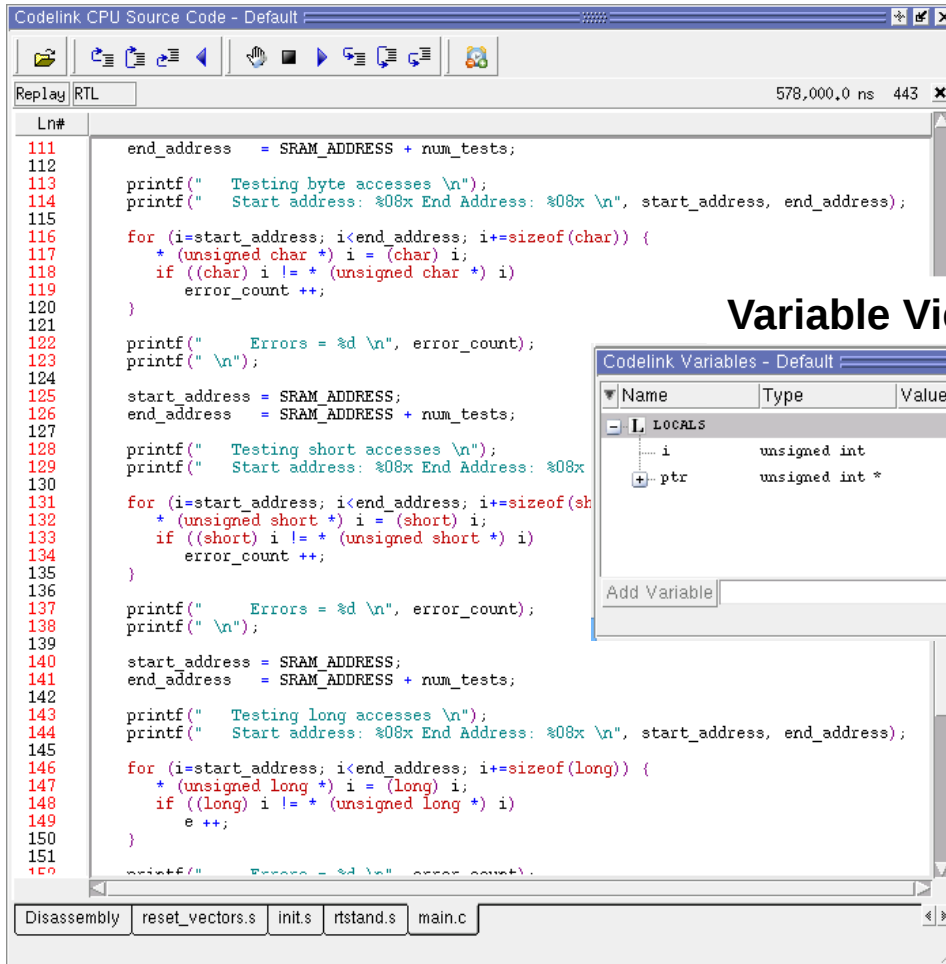
The screenshot displays the ModelSim SE PLUS 6.3e interface with the Codelink Non-Intrusive Multi-Core Support project. The interface is divided into several panes:

- Codelink CPU Registers:** Shows the state of CPU registers for two cores, `cpu_a` and `cpu_b`. The registers include PC, SP, CPSR, LR, R0, R1, R2, R3, R4, R5, R6, and R7. The values are displayed in hexadecimal.
- Codelink CPU Source Code:** Displays the source code for the two cores. The code for `cpu_a` is shown in the left pane, and the code for `cpu_b` is shown in the right pane. The code includes initialization, a main loop, and a reset handler.
- Codelink Memory:** Shows the memory contents for the two cores. The memory is displayed in hexadecimal, with addresses ranging from 0x00000000 to 0x0000000F.
- wave - default:** A timing diagram showing the execution of the code. The wave is divided into segments for `cpu_a` and `cpu_b`, with a time scale of 100 ns.

The status bar at the bottom indicates the current time is 171,260 ns, and the simulation is paused at line 3963 of the file `sim/cpu_b/arm926/line_3963`.

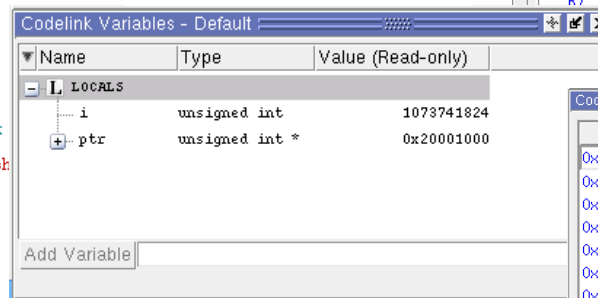
Full Support of Classic SW Debug Views

Source & Assembly View



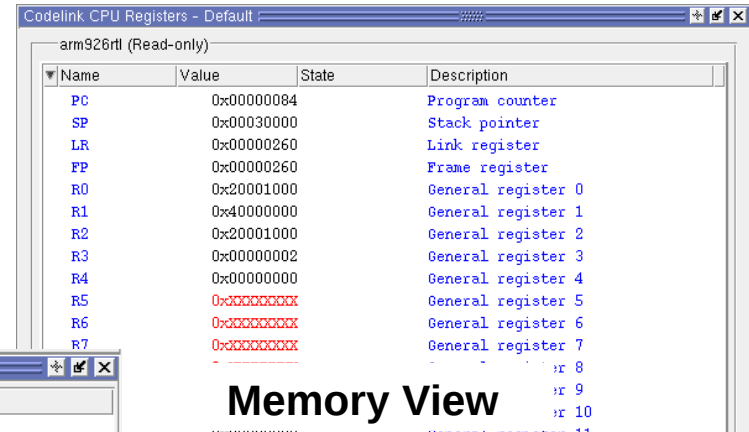
```
Ln#
111 end_address = SRAM_ADDRESS + num_tests;
112
113 printf(" Testing byte accesses \n");
114 printf(" Start address: %08x End Address: %08x \n", start_address, end_address);
115
116 for (i=start_address; i<end_address; i+=sizeof(char)) {
117     * (unsigned char *) i = (char) i;
118     if ((char) i != * (unsigned char *) i)
119         error_count ++;
120 }
121
122 printf(" Errors = %d \n", error_count);
123 printf(" \n");
124
125 start_address = SRAM_ADDRESS;
126 end_address = SRAM_ADDRESS + num_tests;
127
128 printf(" Testing short accesses \n");
129 printf(" Start address: %08x End Address: %08x \n", start_address, end_address);
130
131 for (i=start_address; i<end_address; i+=sizeof(short)) {
132     * (unsigned short *) i = (short) i;
133     if ((short) i != * (unsigned short *) i)
134         error_count ++;
135 }
136
137 printf(" Errors = %d \n", error_count);
138 printf(" \n");
139
140 start_address = SRAM_ADDRESS;
141 end_address = SRAM_ADDRESS + num_tests;
142
143 printf(" Testing long accesses \n");
144 printf(" Start address: %08x End Address: %08x \n", start_address, end_address);
145
146 for (i=start_address; i<end_address; i+=sizeof(long)) {
147     * (unsigned long *) i = (long) i;
148     if ((long) i != * (unsigned long *) i)
149         e ++;
150 }
151
152 printf(" Errors = %d \n", error_count);
```

Variable View



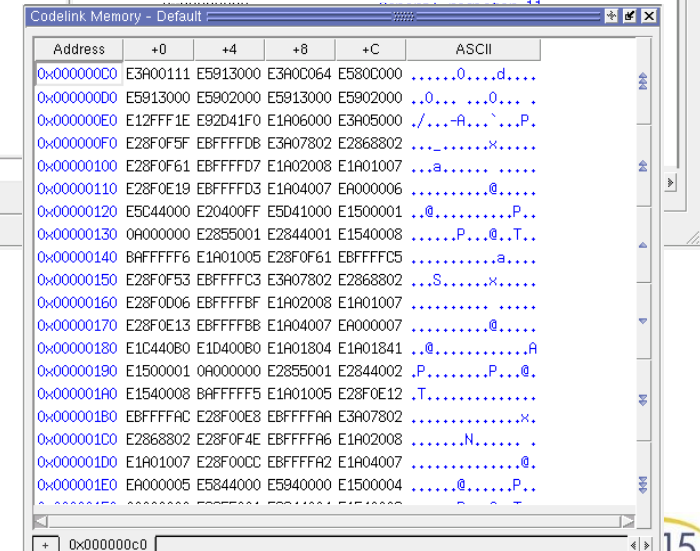
Name	Type	Value (Read-only)
LOCALS		
i	unsigned int	1073741824
ptr	unsigned int *	0x20001000

Register View



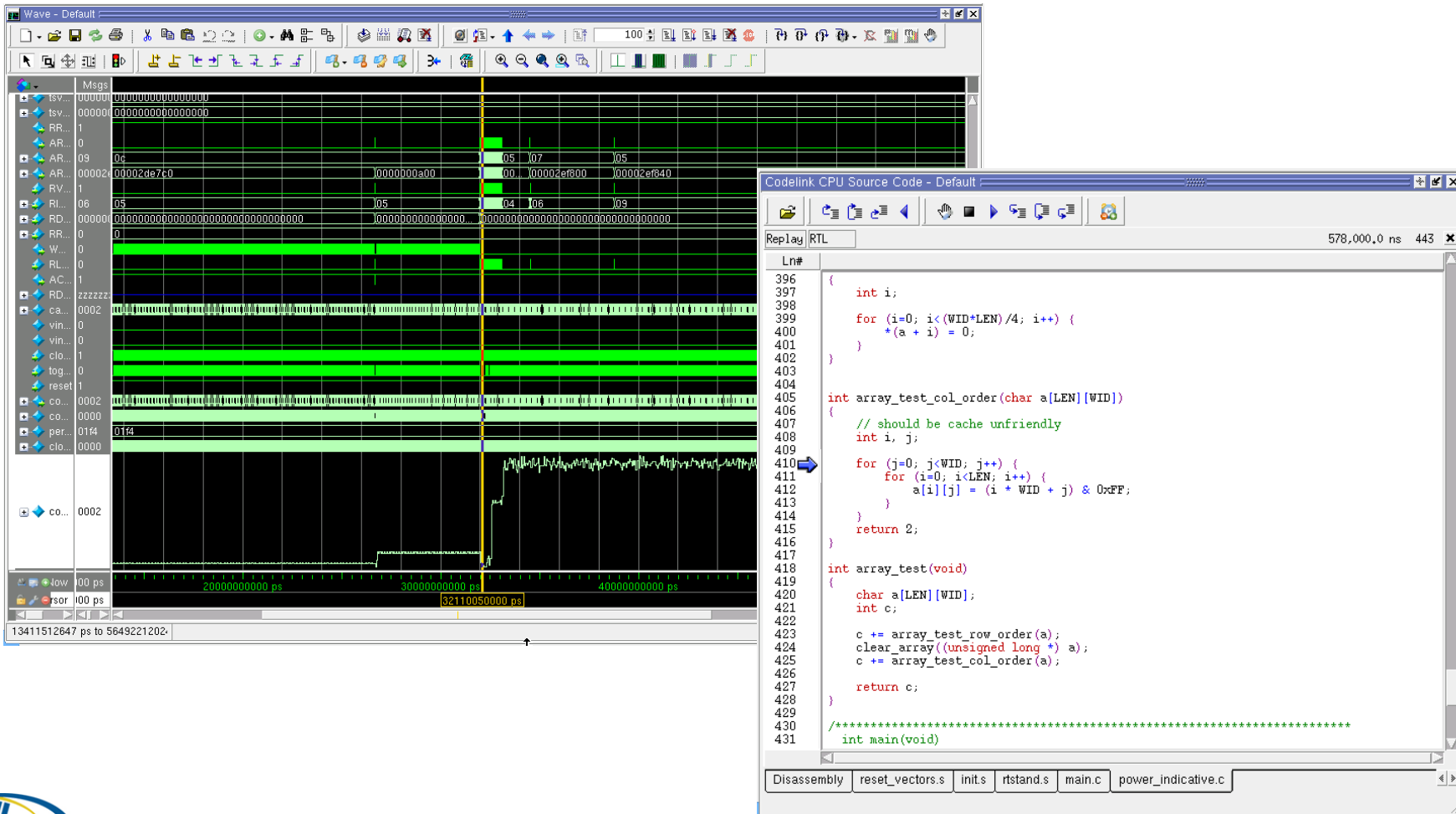
Name	Value	State	Description
PC	0x00000084		Program counter
SP	0x00030000		Stack pointer
LR	0x00000260		Link register
FP	0x00000260		Frame register
R0	0x20001000		General register 0
R1	0x40000000		General register 1
R2	0x20001000		General register 2
R3	0x00000002		General register 3
R4	0x00000000		General register 4
R5	0x00000000		General register 5
R6	0x00000000		General register 6
R7	0x00000000		General register 7
R8	0x00000000		General register 8
R9	0x00000000		General register 9
R10	0x00000000		General register 10
R11	0x00000000		General register 11
R12	0x00000000		General register 12
R13	0x00000000		General register 13
R14	0x00000000		General register 14
R15	0x00000000		General register 15

Memory View



Address	+0	+4	+8	+C	ASCII
0x00000000	E3A00111	E5913000	E3A0C064	E580C000	0...d...
0x00000004	E5913000	E5902000	E5913000	E5902000	..0... ..0...
0x00000008	E12FFF1E	E92D41F0	E1A06000	E3A05000	./...A...`...P.
0x0000000C	E28F0F5F	EBFFFFD8	E3A07802	E2868802	X.....
0x00000010	E28F0F61	EBFFFFD7	E1A02008	E1A01007	...a.....
0x00000014	E28F0E19	EBFFFFD3	E1A04007	E9A00006	@.....
0x00000018	E5C44000	E20400FF	E5D41000	E1500001	..@.....P..
0x0000001C	0A000000	E2855001	E2844001	E1540008	P...@...T..
0x00000020	BAFFFFFF	E1A01005	E28F0F61	EBFFFFC5	a.....
0x00000024	E28F0F53	EBFFFFC3	E3A07802	E2868802	...S.....x.....
0x00000028	E28F0D06	EBFFFFB8	E1A02008	E1A01007	@.....
0x0000002C	E28F0E13	EBFFFFB8	E1A04007	E9A00007	@.....
0x00000030	E1C440B0	E1D400B0	E1A01804	E1A01841	..@.....A
0x00000034	0x00000001	0A000000	E2855001	E2844002	..P.....P...@..
0x00000038	E1540008	BAFFFFFF	E1A01005	E28F0E12	..T.....
0x0000003C	EBFFFFAC	E28F00E8	EBFFFFAA	E3A07802	x.....
0x00000040	E2868802	E28F0F4E	EBFFFFA6	E1A02008	N.....
0x00000044	E1A01007	E28F00C0	EBFFFFA2	E1A04007	@.....
0x00000048	E9A00005	E5844000	E5940000	E1500004	@.....P..

Synchronized Hardware and Software Views



New Paradigm: Post-Run Software Debug

Single User

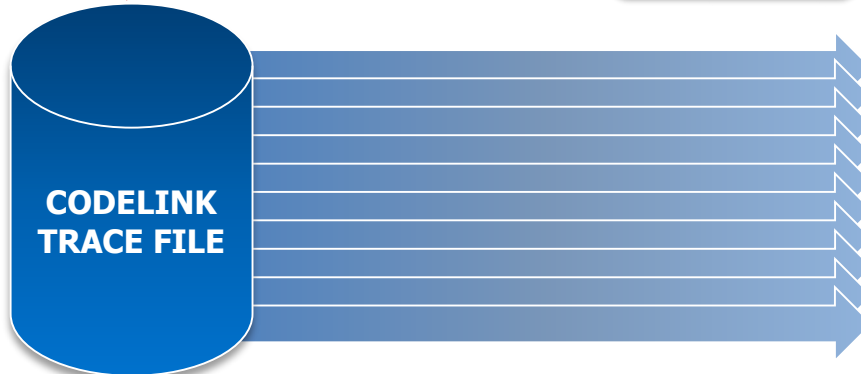


JTAG
1 MHz



Post-Emulation

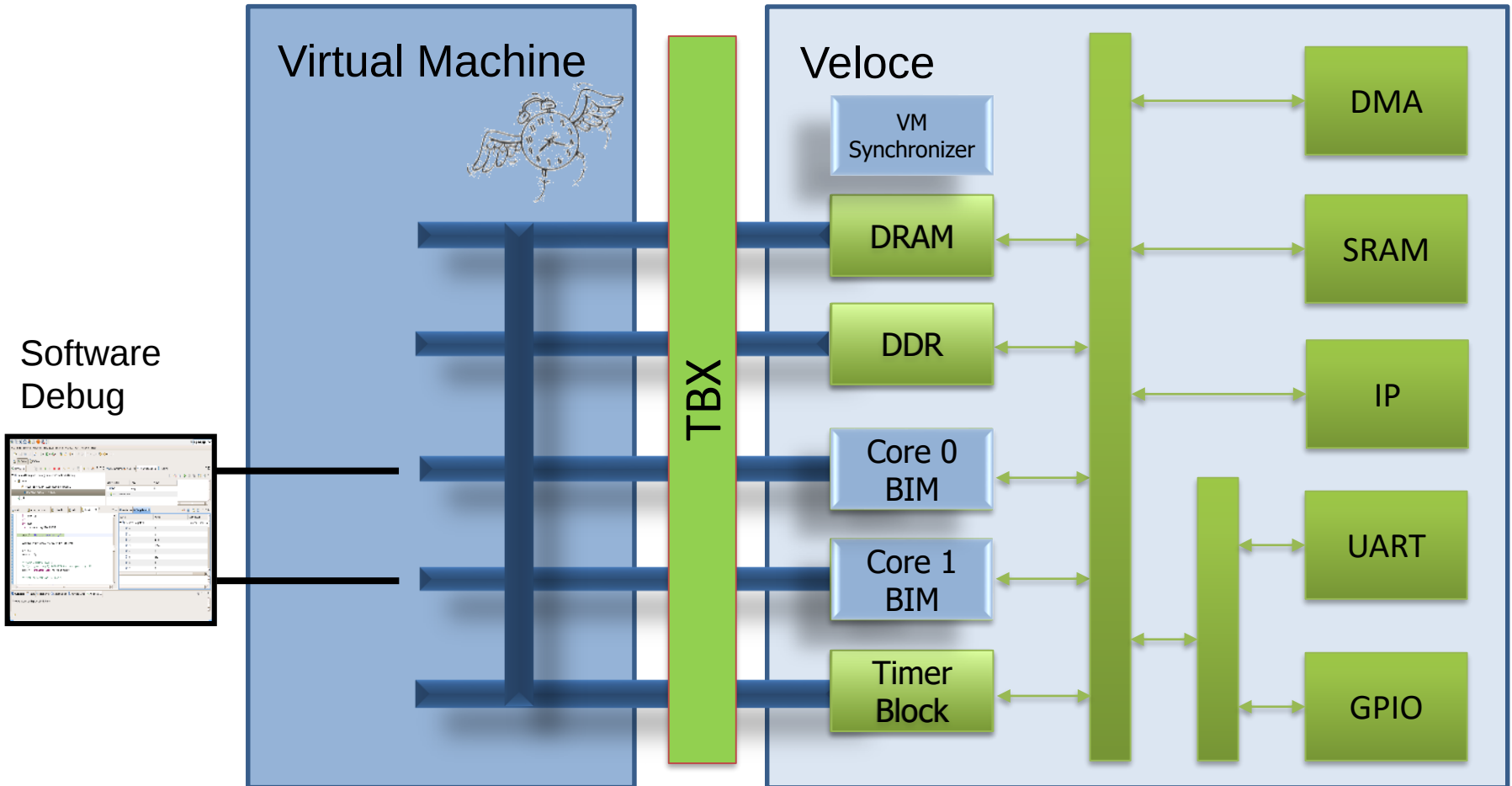
Codelink
50 MHz



~10 Simultaneous
users

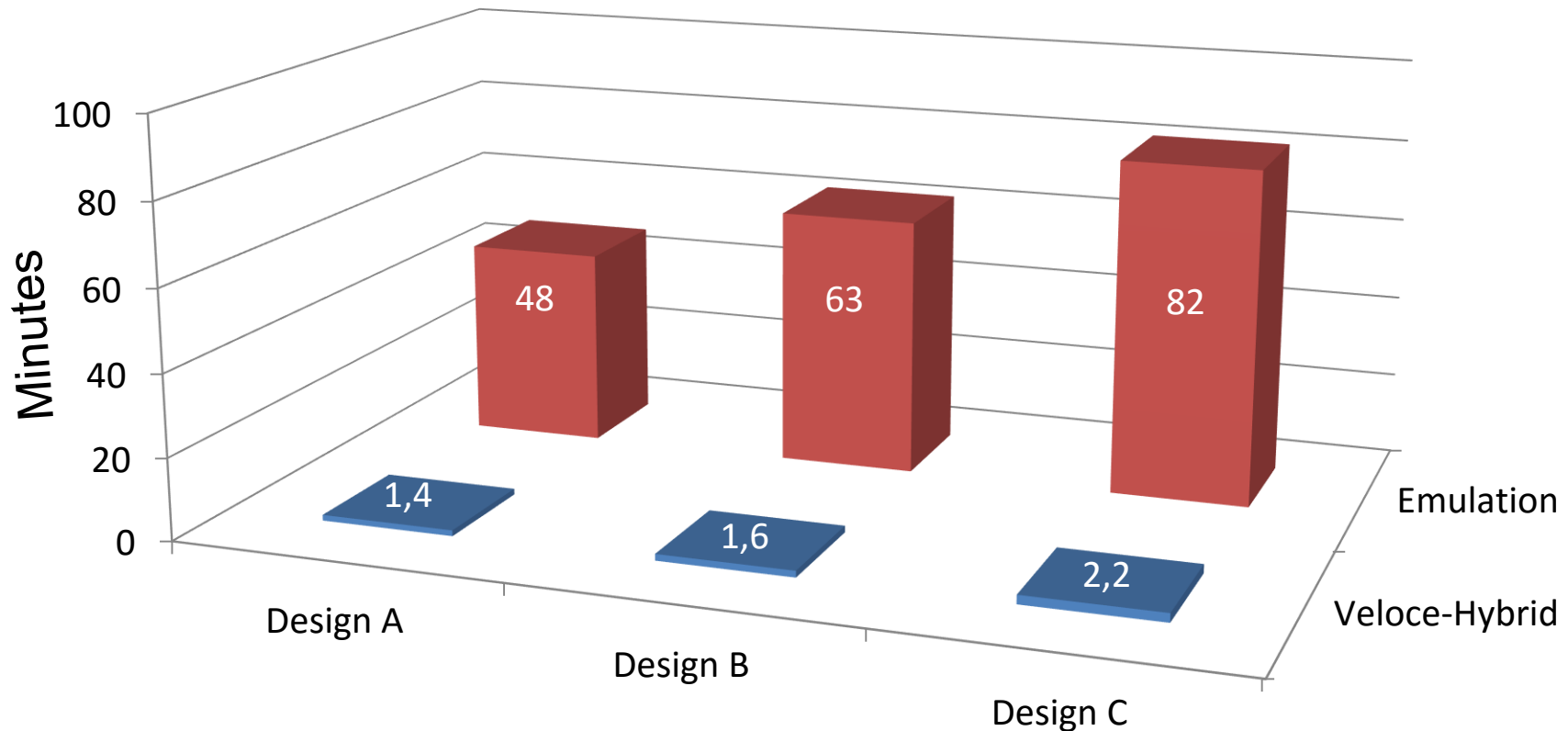


WarpCore: Veloce Hybrid Emulation



- 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

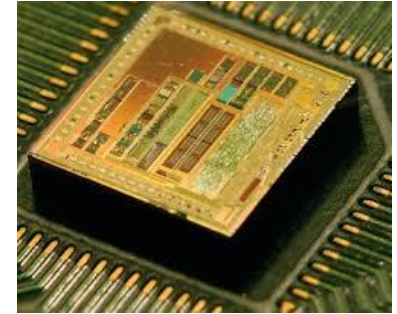
- Non intrusive multi-core debug
- Process-level multi-core debug
- Backup rather than rerun from time zero
- Add “printf” without recompile and rerun
- Virtual machine for fast OS boot

Agenda

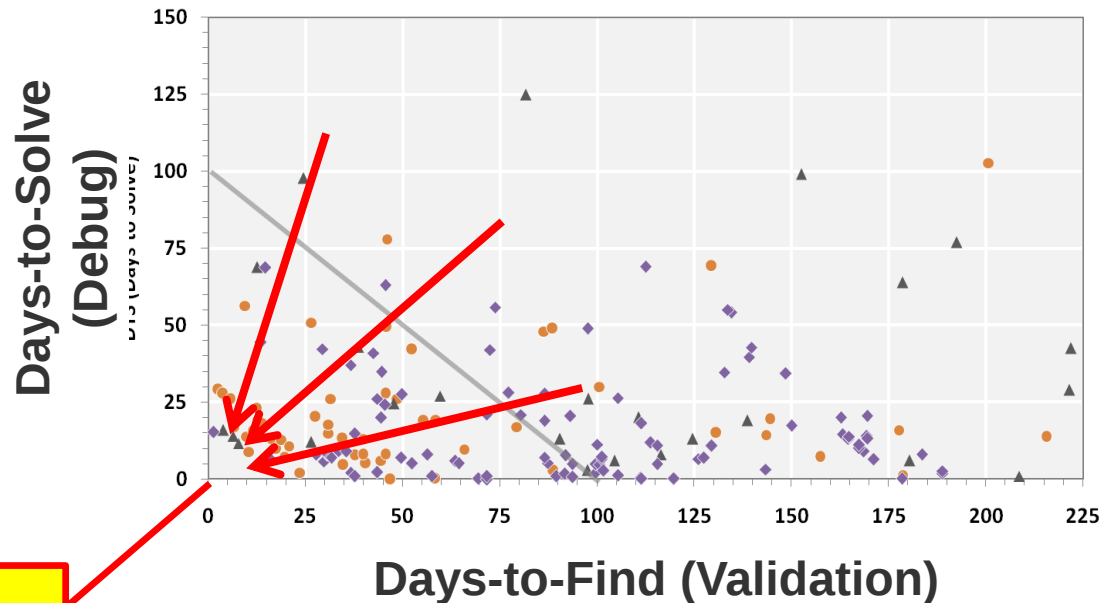
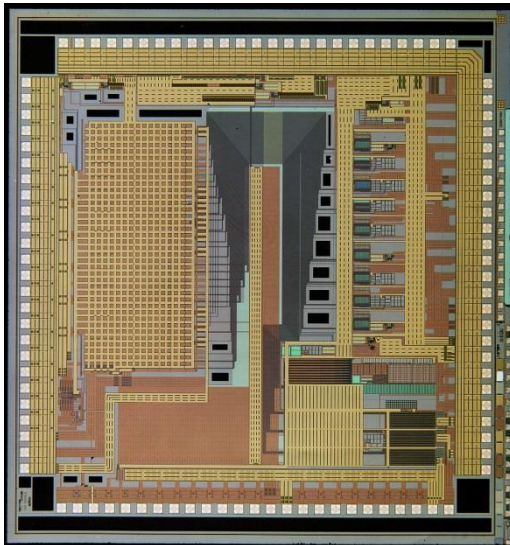
- Introduction – the Debug challenge
- Advanced RTL Debug Scenarios
- Debug of Complex Testbenches
- Power-aware Verification Debug
- Integrated Hardware/Software Debug
- **Post-Silicon Debug Solutions**

Goals of Post-Silicon Validation/Debug

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

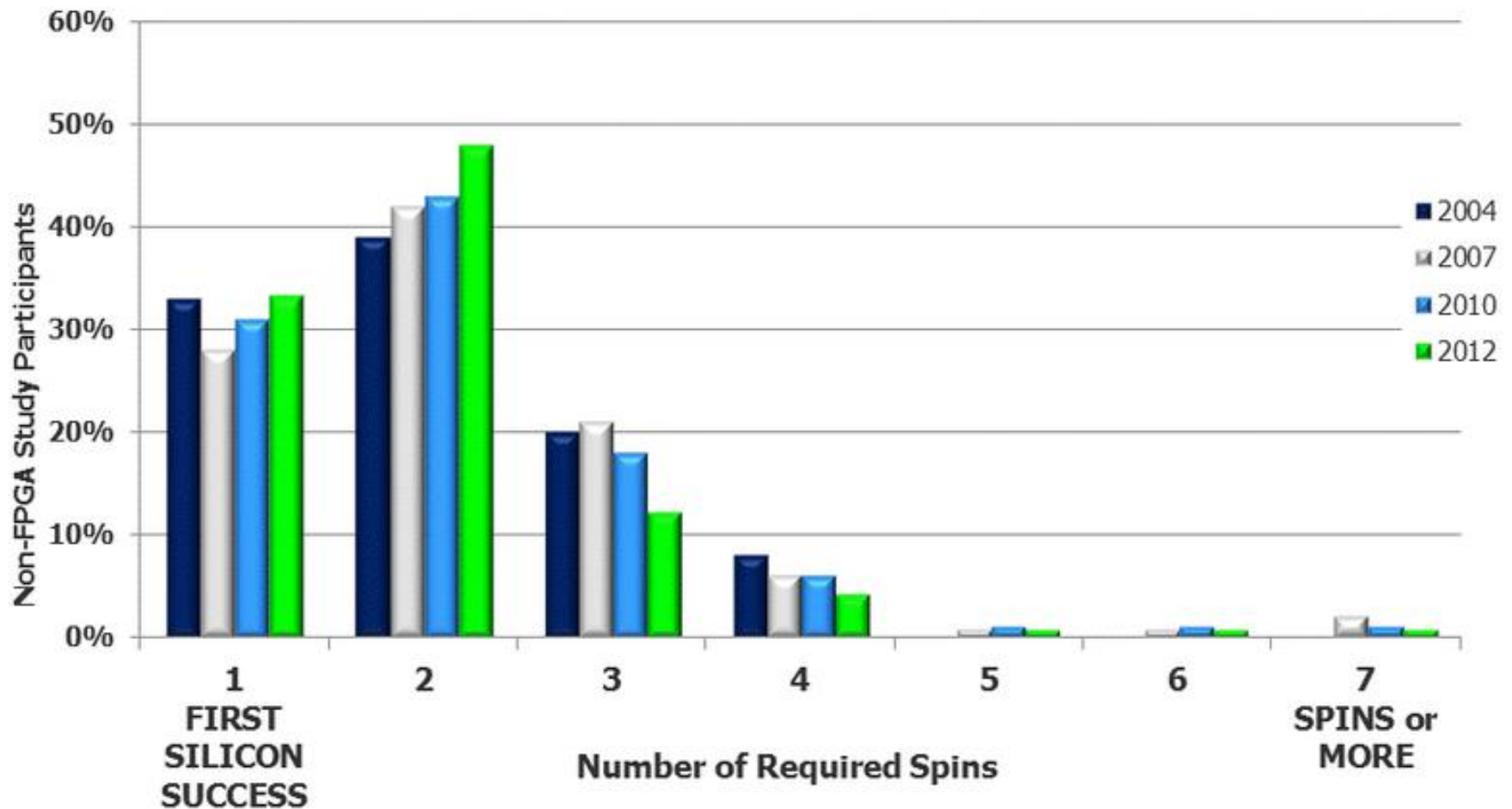


Debug & Validation Bug Characterization
for Selected Products



Evolution of SOC Design

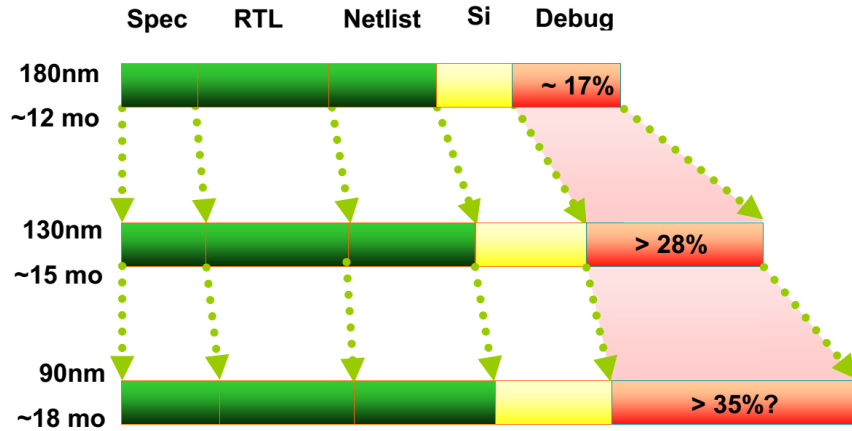
Unplanned Respins => Missed Revenue!



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

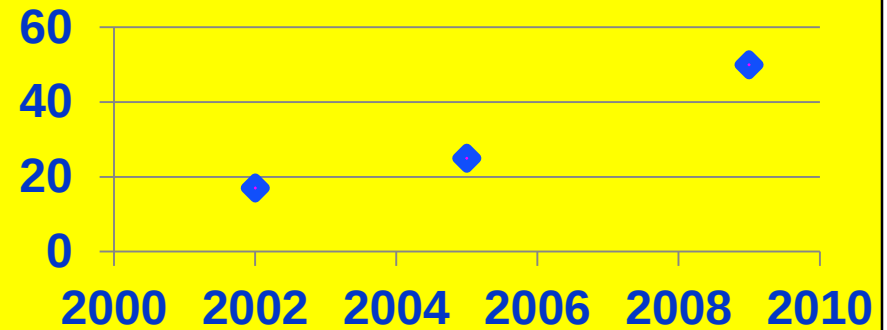
Post-Silicon Back-End Effort Growing

Post-Silicon Debug vs. Process



[Abramovici, DAC'06]

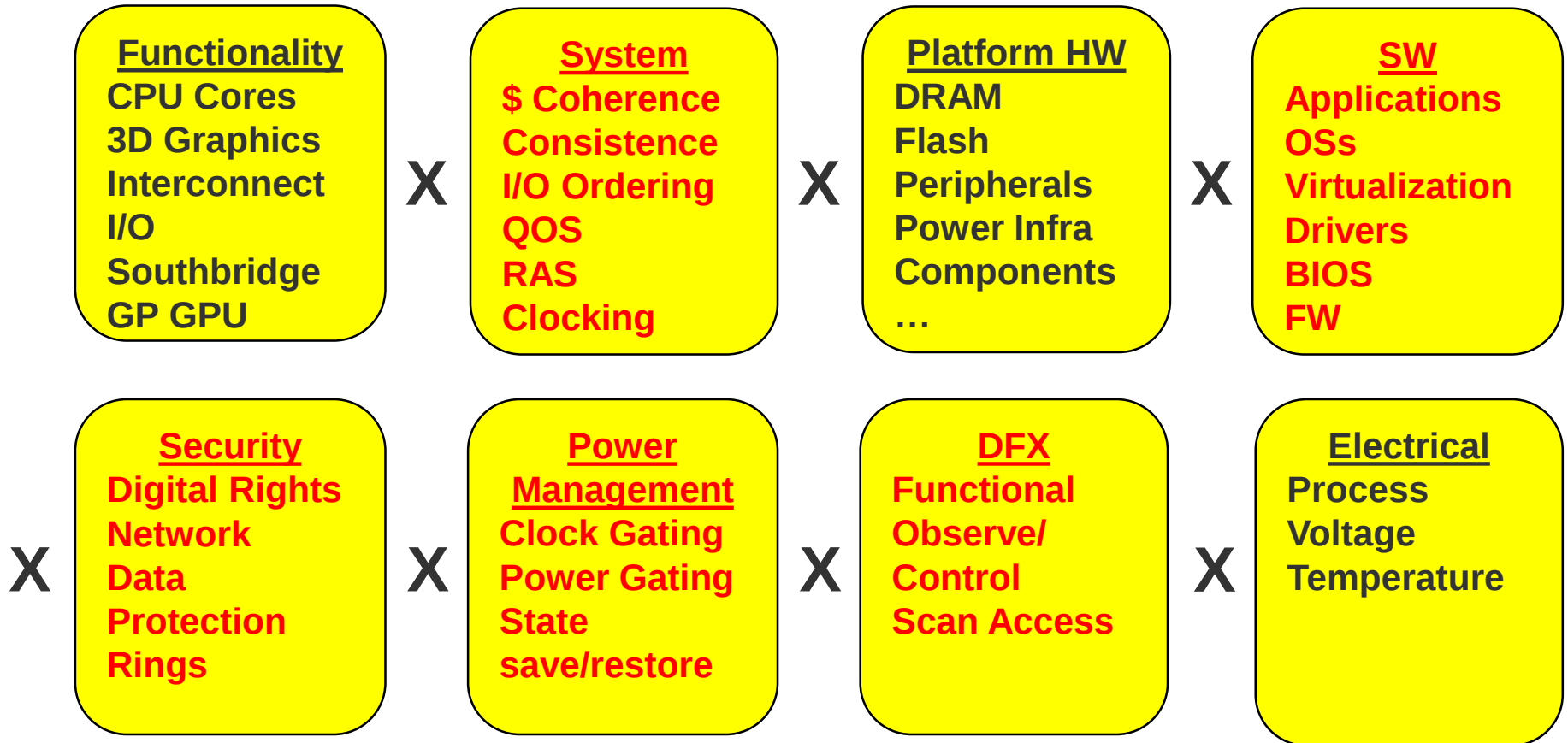
Validation as Percent of Total Effort (%)



[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

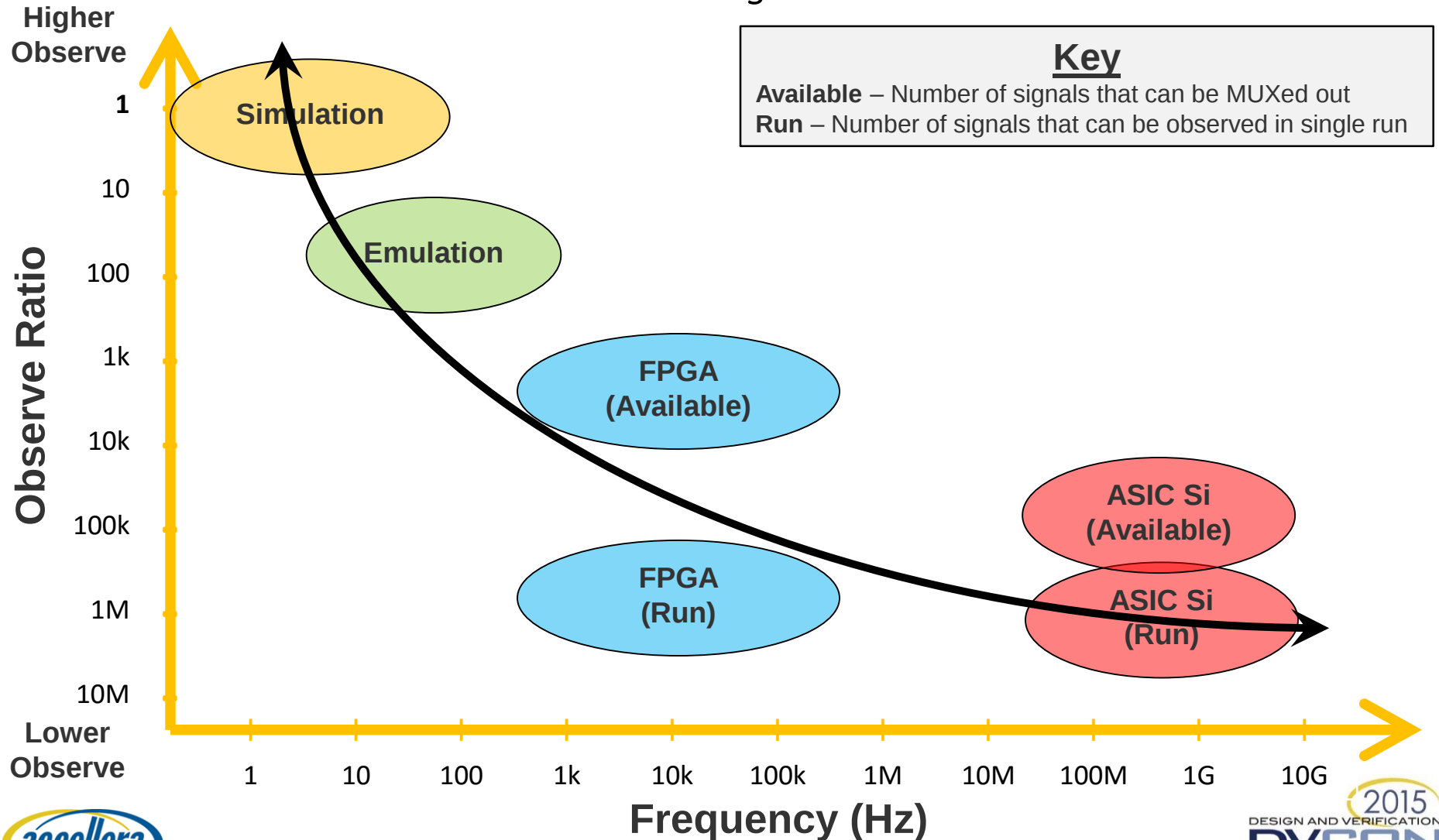


What does all this add up to?

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

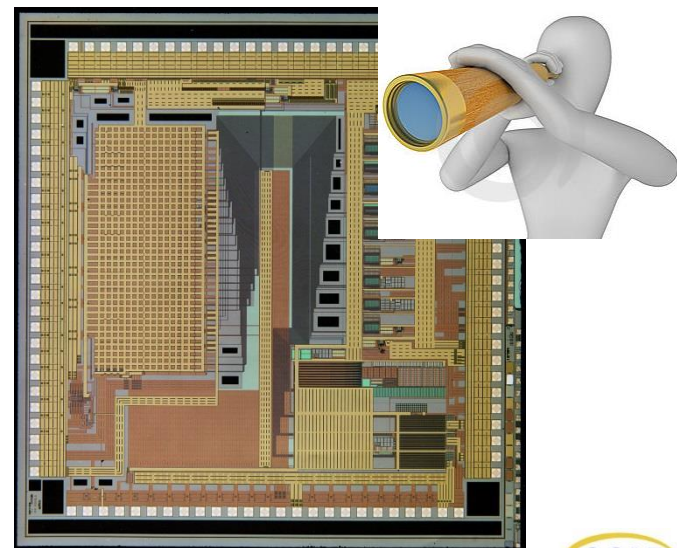
Feature(s)	Comments
Program Flow Trace	•ARM: Coresight ETM PC trace •AMD: BTHB •Intel: Processor Trace
HW History Buffers	Small trace buffers in HW: THBs, EHB, LBR buffers, LTSSM buffers
Scandump ₁	•ATPG sanchains configured as single-chain, TDI to TDO •Debug of hardlocks and more •“bread and butter” feature for many companies •Destructive in nature •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 Expensive and custom per company

Where do you turn for more difficult SOC interaction bugs?

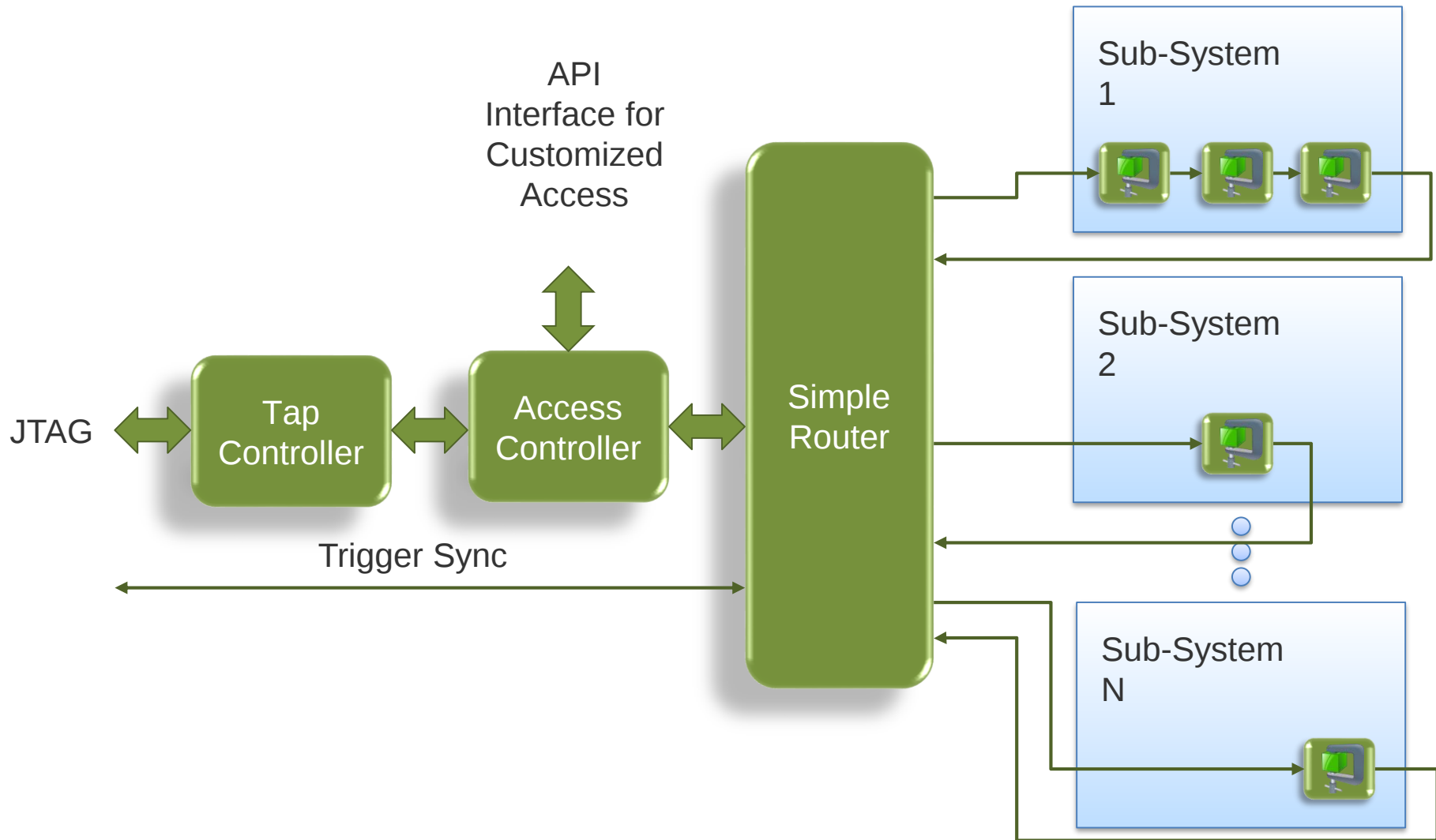
1. Array and main memory dump applies in addition here too.

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

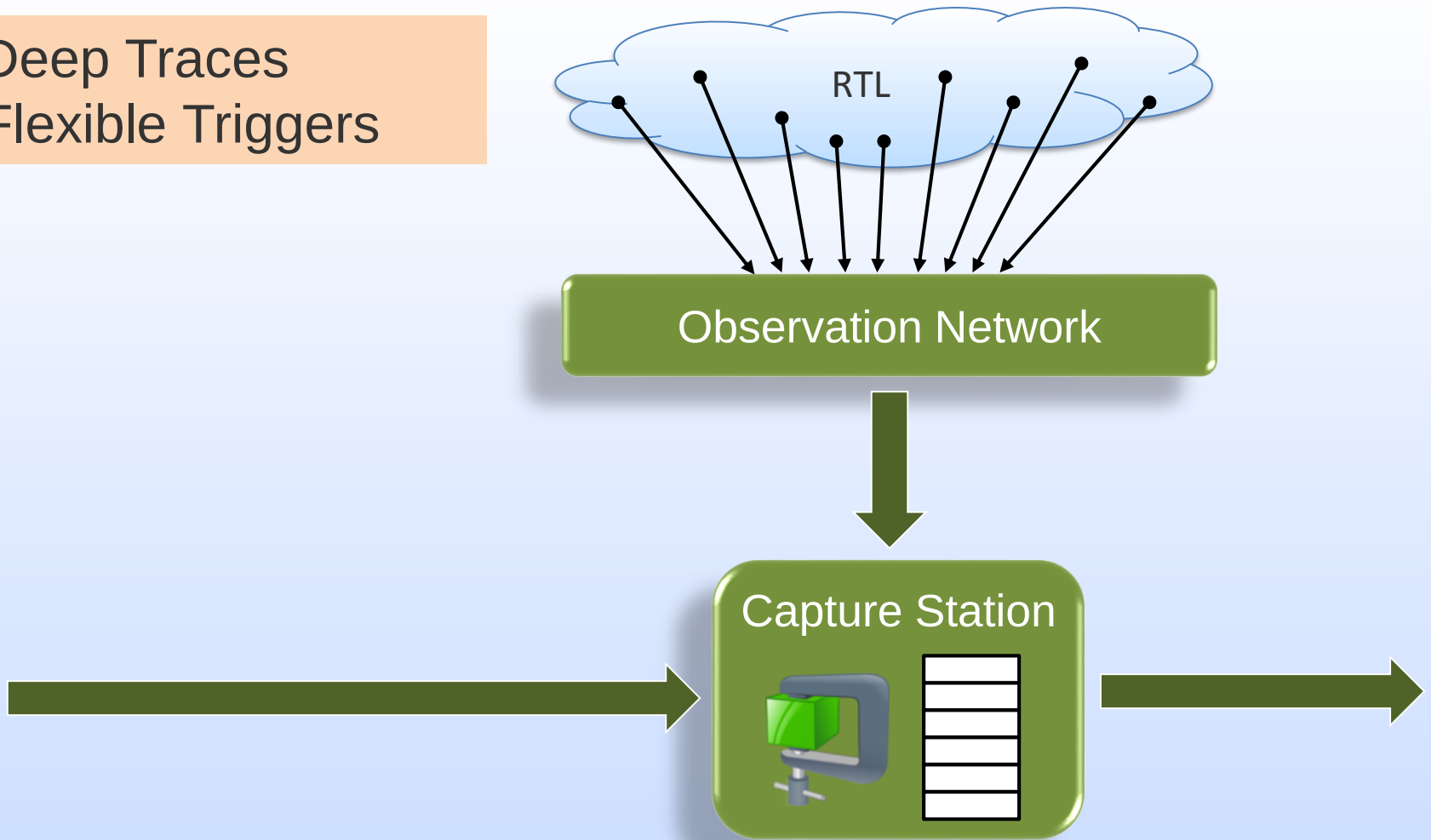


Architected for Post Silicon Debug



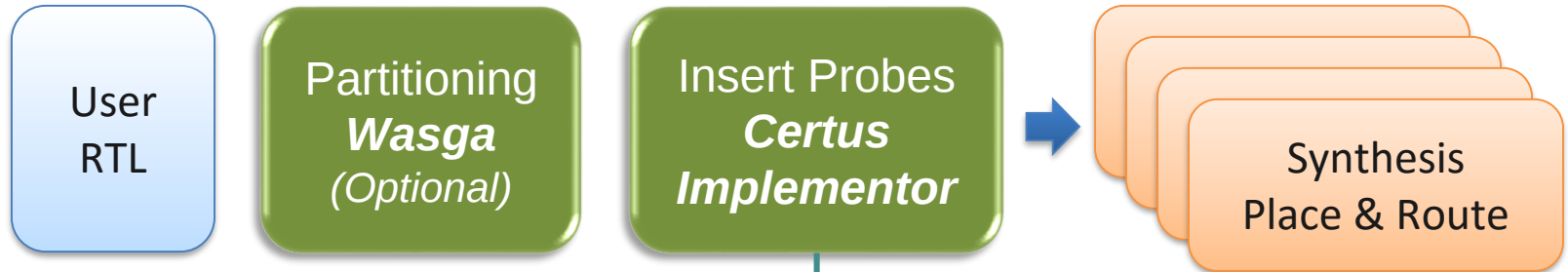
Capture Station

- Deep Traces
- Flexible Triggers



Certus User Flow

Design



File Interchange
(.VFE)

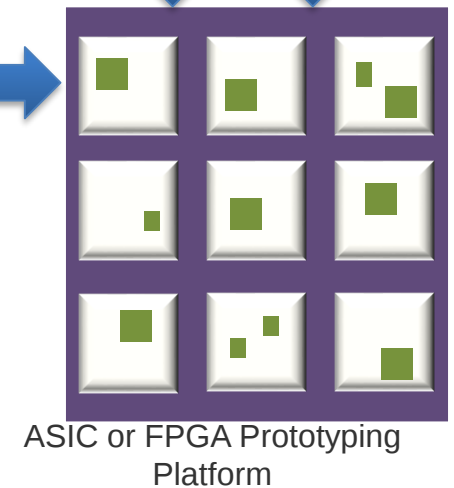
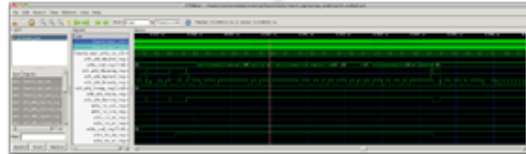
FPGA Bitfile(s) /
Gate Netlist

Lab



Debug
Cycle

Lab Bench PC
(Windows, Linux)



Instrumented Signal Selection

Certus Implementor: /home/shaynal/streaming/siice3/certus1.ipj

File View Help

Design:

- (slice) 473
 - bufg_inst (BUFG) 0
 - FCLK (IBUFGDS) 0
 - mmi64_m_upstream...4_m_upstreamif) 0
 - profpga_clocksync...rofpga_clocksync) 0
 - si (invaders_ps2) 67303 (1041)
 - U_PROFPGA_CTRL (profpga_ctrl) 0
 - UCLK (IBUFGDS) 0
 - upstream_data_ge..._data_generator) 0
 - VCLK (IBUFGDS) 0

Groups:

Infrastructure:

- Summary
 - Signals Selected: 1041
 - Equivalent Signals Selected: 1766
 - Signals Available: 67776
 - LUT Cost: 6.6k
 - RAM Cost: 32.8k
- Routers
- Stations
- Obs Networks

Project **Signals** **Infrastructure** **Insert**

* req	sig	in	out	inout	mem	ff	state	enun	OptiScore
[=] si/kbd/RX_ShiftReg[7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsCL[7][7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsCL[6][7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsCL[5][7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsCL[4][7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsCL[3][7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsCL[2][7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsCL[1][7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsCL[0][7:0]	signal						flip-flop		★★★
[=] si/core/u_mw8080/u_8080/u0/RegAddrC[2:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/RegAddrB_r[2:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/RegAddrA_r[2:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Read_To_Reg_r[4:0]	signal						flip-flop		★★★
[=] si/core/u_mw8080/u_8080/u0/DI_Reg[7:0]	signal								★★★
[=] si/core/u_mw8080/u_8080/u0/DI_Reg[7:0]	signal						flip-flop		★★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsBH[0][7:0]	signal						flip-flop		★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsAL[7][7:0]	signal						flip-flop		★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsAL[6][7:0]	signal						flip-flop		★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsAL[5][7:0]	signal						flip-flop		★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsAL[4][7:0]	signal						flip-flop		★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsAL[3][7:0]	signal						flip-flop		★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsAL[2][7:0]	signal						flip-flop		★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsAL[1][7:0]	signal						flip-flop		★★
[✓] si/core/u_mw8080/u_8080/u0/Regs/RegsAL[0][7:0]	signal						flip-flop		★★

Equivalent Signals:

hdl/ps2kbd.vhd RX_ShiftReg[0]

```

signal PS2_Sample : std_logic;
signal PS2_Data_s : std_logic;

signal RX_Bit_Cnt : unsigned(3 downto 0);
signal RX_Byte : unsigned(2 downto 0);
signal RX_Change : std_logic_vector(7 downto 0);
  
```

Import Select Deselect Verify

INFO - Elaborating (phase 5).
 INFO - Running OptiRank
 INFO - OptiRank done
 INFO - Adding clock 'FPGACKL' with source 'fpgackl'.
 INFO - Setting working directory to /home/shaynal/streaming/siice3/certus_work

All (76) Error (0) Warning (5) Info (71)

Infrastructure

Certus Implementor: /home/shaynal/streaming/siice3/certus1.ipj

File View Help

Design:

(siice)	473
bufg_inst (BUFG)	0
FCLK (IBUFGDS)	0
mmi64_m_upstream...4_m_upstreamif)	0
profpga_clocksync...rofpga_clocksync)	0
si (invaders_ps2)	67303 (1041)
U_PROFPGA_CTRL (profpga_ctrl)	0
UCLK (IBUFGDS)	0
upstream_data_ge..._data_generator)	0
VCLK (IBUFGDS)	0

Groups:

Infrastructure:

- Summary
 - Signals Selected: 1041
 - Equivalent Signals Selected: 1766
 - Signals Available: 67776
 - LUT Cost: 6.6k
 - RAM Cost: 32.8k
- Routers
- Stations
- Obs Networks

Infrastructure:

Clock name:	Clock source:
FPGACLK	fpgackl

Infrastructure:

[-] Router:	CERTUS_RTR
clock	FPGACLK
JTAG interface	xilinx_7_tap
interrupt type	internal

[-] Station:	FPGACLK_STN
clock	FPGACLK
width	64
depth	512

[-] Network:	FPGACLK_STN
clock	FPGACLK
width	64
pipeline stages	3
observables	(1041)

Selected Signals

Selected Signals	Station
si/AD[15:0]	FPGACLK_STN
si/Button_Coin	FPGACLK_STN
si/Button_F	FPGACLK_STN
si/Button_L	FPGACLK_STN
si/Button_P1	FPGACLK_STN
si/Button_P2	FPGACLK_STN
si/Button_R	FPGACLK_STN
si/Buttons[5:0]	FPGACLK_STN
si/core/D5[15:0]	FPGACLK_STN
si/core/EA[2:0]	FPGACLK_STN
si/core/GDB0[6:3]	FPGACLK_STN
si/core/GDB1[7:0]	FPGACLK_STN
si/core/GDB2[2]	FPGACLK_STN
si/core/GDB2[6:4]	FPGACLK_STN
si/core/GDB[7:0]	FPGACLK_STN
si/core/PortWr[6:2]	FPGACLK_STN
si/core/S[7:0]	FPGACLK_STN
si/core/Sample	FPGACLK_STN
si/core/u_mw8080/ClkEnCnt[2:0]	FPGACLK_STN
si/core/u_mw8080/CntD5[3:0]	FPGACLK_STN
si/core/u_mw8080/CntE5[4:0]	FPGACLK_STN
si/core/u_mw8080/CntE7[4]	FPGACLK_STN
si/core/u_mw8080/DBin	FPGACLK_STN
si/core/u_mw8080/Hold	FPGACLK_STN
si/core/u_mw8080/Hold_n	FPGACLK_STN
si/core/u_mw8080/Int	FPGACLK_STN
si/core/u_mw8080/IntTrig	FPGACLK_STN
si/core/u_mw8080/IntTrigOld	FPGACLK_STN
si/core/u_mw8080/Sel[1:0]	FPGACLK_STN
si/core/u_mw8080/ivid	FPGACLK_STN
si/core/u_mw8080/RR[9:0]	FPGACLK_STN
si/core/u_mw8080/Skip[7:0]	FPGACLK_STN

INFO - Running OptiRank

INFO - OptiRank done

INFO - Adding clock 'FPGACLK' with source 'fpgackl'.

INFO - Setting working directory to /home/shaynal/streaming/siice3/certus_work

INFO - Signal selection is valid.

All (77) Error (0) Warning (5) Info (72)

Signal Selection

Certus Analyzer: /export/shaynal/icedemo485new/test2.apj (on hitlab4)

File View Help

Design:

- (siice) 44
 - UCLK (IBUFGDS) 1
 - si (invaders_ps2) 1721 (59)

Configuration: config0

config0

- (siice) 64 (59)
 - FPGACKL_STN

Project **Signals** **Trigger** **Stations** **Configure**

	sig	in	out	inout	Station
Blue					(siice) FPGACKL_STN
buttons[6:3]					(siice) FPGACKL_STN
fpgackl					(siice) FPGACKL_STN
Green					(siice) FPGACKL_STN
hsync					(siice) FPGACKL_STN
LED[6:3]					(siice) FPGACKL_STN
LED[15:8]					(siice) FPGACKL_STN
pixel					(siice) FPGACKL_STN
Red					(siice) FPGACKL_STN
Sel_KB					(siice) FPGACKL_STN
vsync					(siice) FPGACKL_STN
si/AD[15:0]					(siice) FPGACKL_STN
si/Buttons[5:0]					(siice) FPGACKL_STN
si/Buttons_n[5:0]					(siice) FPGACKL_STN
si/CSync					(siice) FPGACKL_STN
si/DIP[8:1]					(siice) FPGACKL_STN
✓ si/Hsync					(siice) FPGACKL_STN
si/IB[7:0]					(siice) FPGACKL_STN
si/LED_OUT[7:0]					(siice) FPGACKL_STN
✓ si/Pixel					(siice) FPGACKL_STN
si/Press					(siice) FPGACKL_STN
si/PS2_Data_s					(siice) FPGACKL_STN
si/PS2_Sample					(siice) FPGACKL_STN
si/RAB[12:0]					(siice) FPGACKL_STN
si/RDB[7:0]					(siice) FPGACKL_STN

Equivalent Signals:

- (top)

+ Select - Deselect Verify

INFO - Hardware(enum, part=0): loop = 0
 INFO - Exact match for part 0.
 INFO - Loading project data...
 INFO - Signal database complete

All (23) Error (0) Warning (1) Info (22)

Set the Trigger Condition

The screenshot displays the Certus Analyzer software interface. The title bar reads "Certus Analyzer: /export/shaynal/icedemo485new/test2.apj (on hitlab4)". The interface is divided into several sections:

- Design:** A tree view on the left showing the project hierarchy. It includes components like UCLK (IBUFGDS), si (invaders_ps2), core (invaders), kbd (ps2kbd), u_RAM (siram), u_ROM (sirom), and u_vga (video_display_vga_bb) with their respective line counts.
- Configuration:** A section below the Design tree showing the selected configuration "config0" and its components, including "FPGACKL_STN" with 64 (59) lines.
- Trigger Station:** The main configuration area on the right. It shows the "Trigger Station" set to "(si) FPGACKL_STN". The "Radix" is set to "Hex". Under "State 0", the condition is configured as: "Add: < All Signals >" followed by "If si/core/u_mw8080/u_8080/u0/PC[15:0] = 0A62" and "is True" for "1" clock cycles then "trigger".
- Verify:** A button at the bottom right of the Trigger Station configuration area.
- Log:** A bottom section showing a list of messages: "INFO - Exact match for part 0.", "INFO - Loading project data...", "INFO - Signal database complete", and "INFO - 59 Signals Selected".
- Status Bar:** At the very bottom, it shows "All (24)", "Error (0)", "Warning (1)", and "Info (23)".

Run Content

Certus Analyzer: /export/shaynal/icedemo485new/test2.apj (on hitlab4)

File View Help

Design:

- (- (slice) 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

- (- slice 64 (59)
 - FPGACKL_STN

Waveform Viewer

GTKWave Modelsim Other:

Station Status

Configure Run Halt

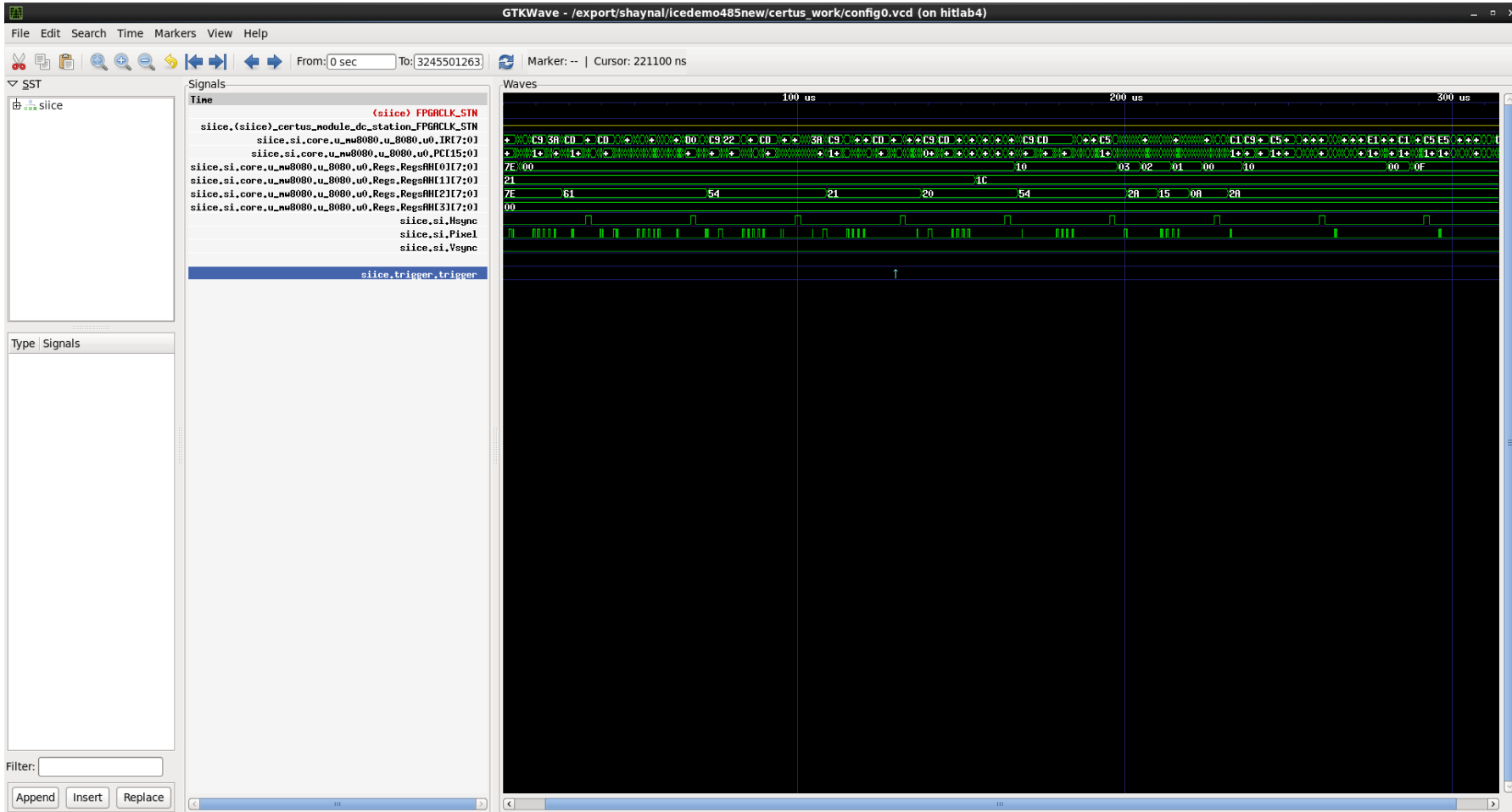
Part	Station	Status
0 - xc7vx485t	(slice) FPGACKL_STN	20.00 MHz 0.0% 9.0E+07 b/s 1.2E+09 bits 13.2:1 compression

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?