Tough Bugs vs Smart Tools
L2/L3 Cache Verification Using SystemVerilog, UVM,
&
Verdi Transaction Debugging

Viba Viswanathan, Doug Reed,
Centaur Technology Inc.
Juliet Runhaar, Jun Zhao,
Synopsys Inc.

September 29, 2016
Austin

Agenda

Design Under Test

UVM Test bench

Verdi Transaction Debug

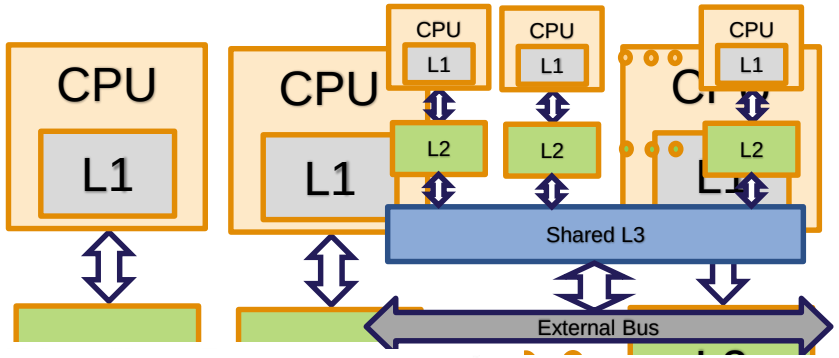
Debug Scenarios

Summary

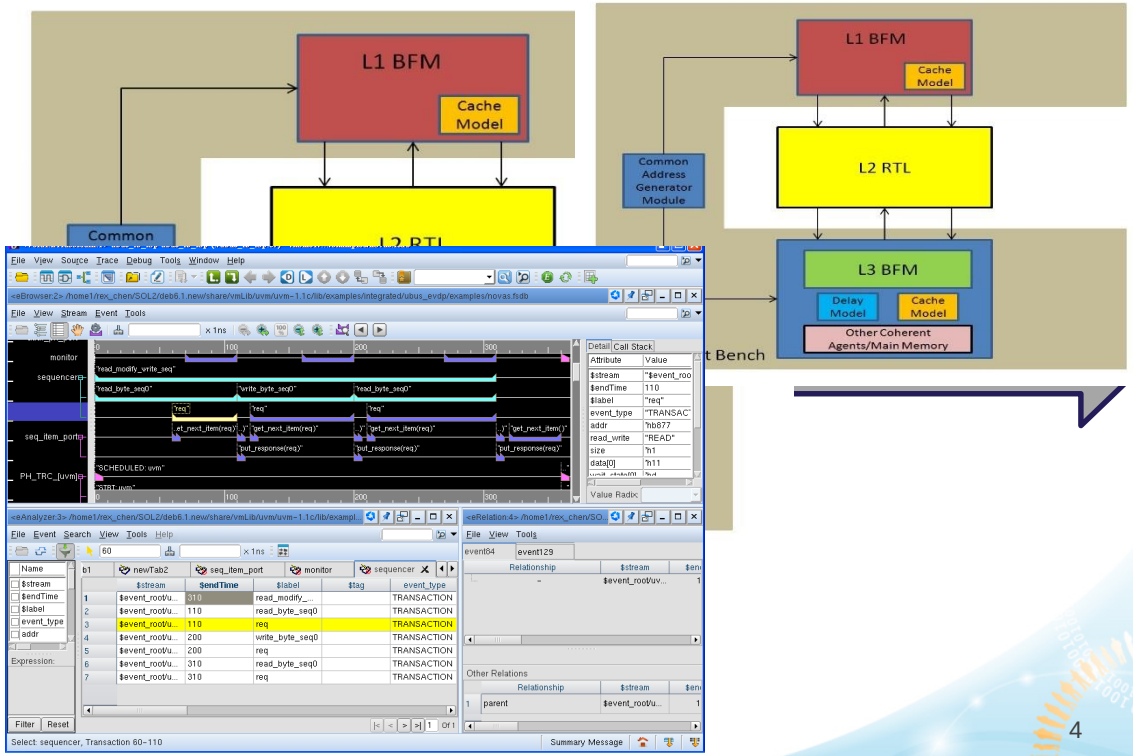
Design Under Test

Centaur Design Environment

- Multi-core and Multi-level cache processor
  - X86 Instruction Set Architecture
  - Coherent L1, L2, L3 Caches
  - External Memory/Bus



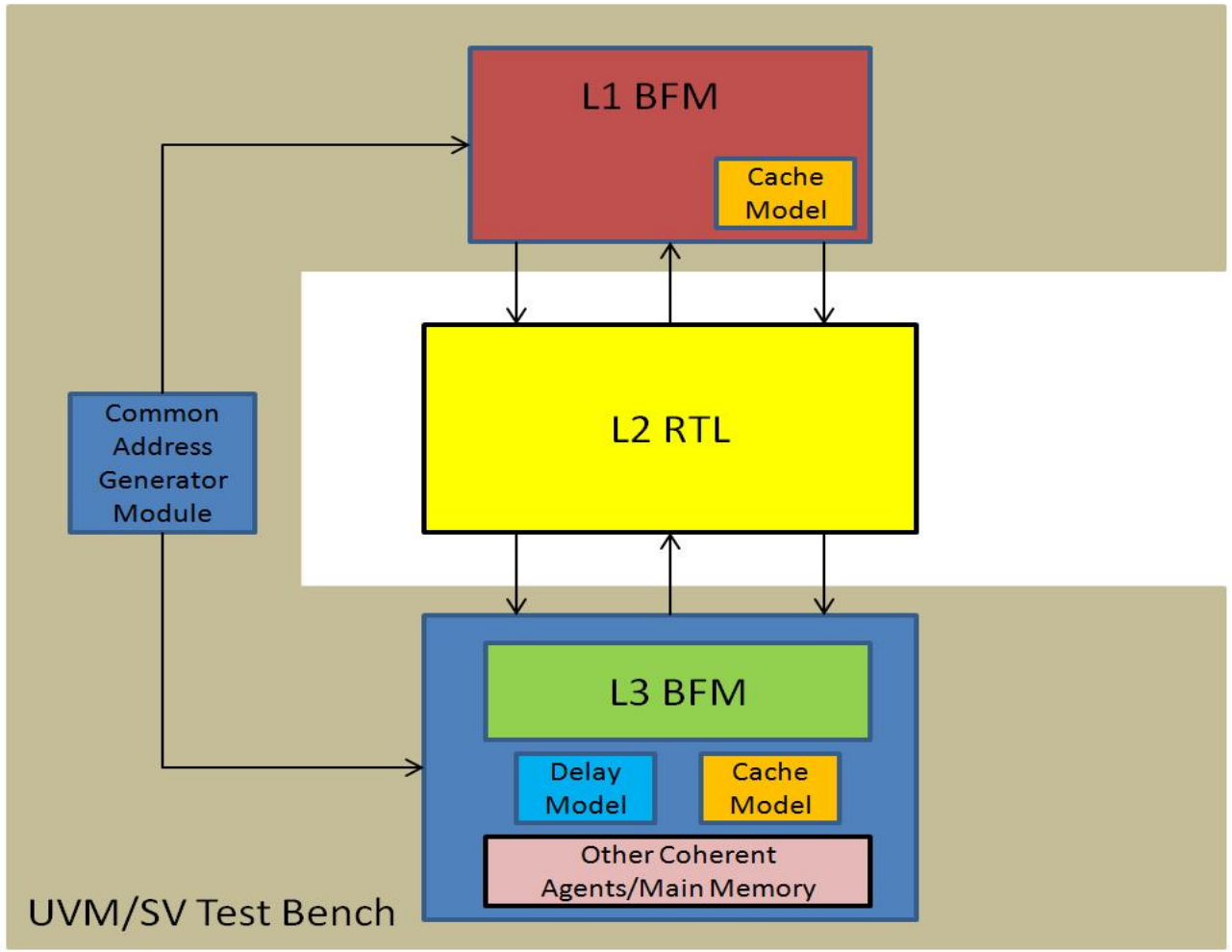
- L2 SV/UVM Test bench
  - Constrained Random Stimulus
  - Cache Coherency Checking
  - Address Generator
  - External Memory/Bus Response Delay Modeling



- Verdi Transaction Debug
  - Visualize, Analyze, and Debug Transactions at High Level of Abstraction

Verification Environment

L2 System Verilog, UVM Unit Level Environment



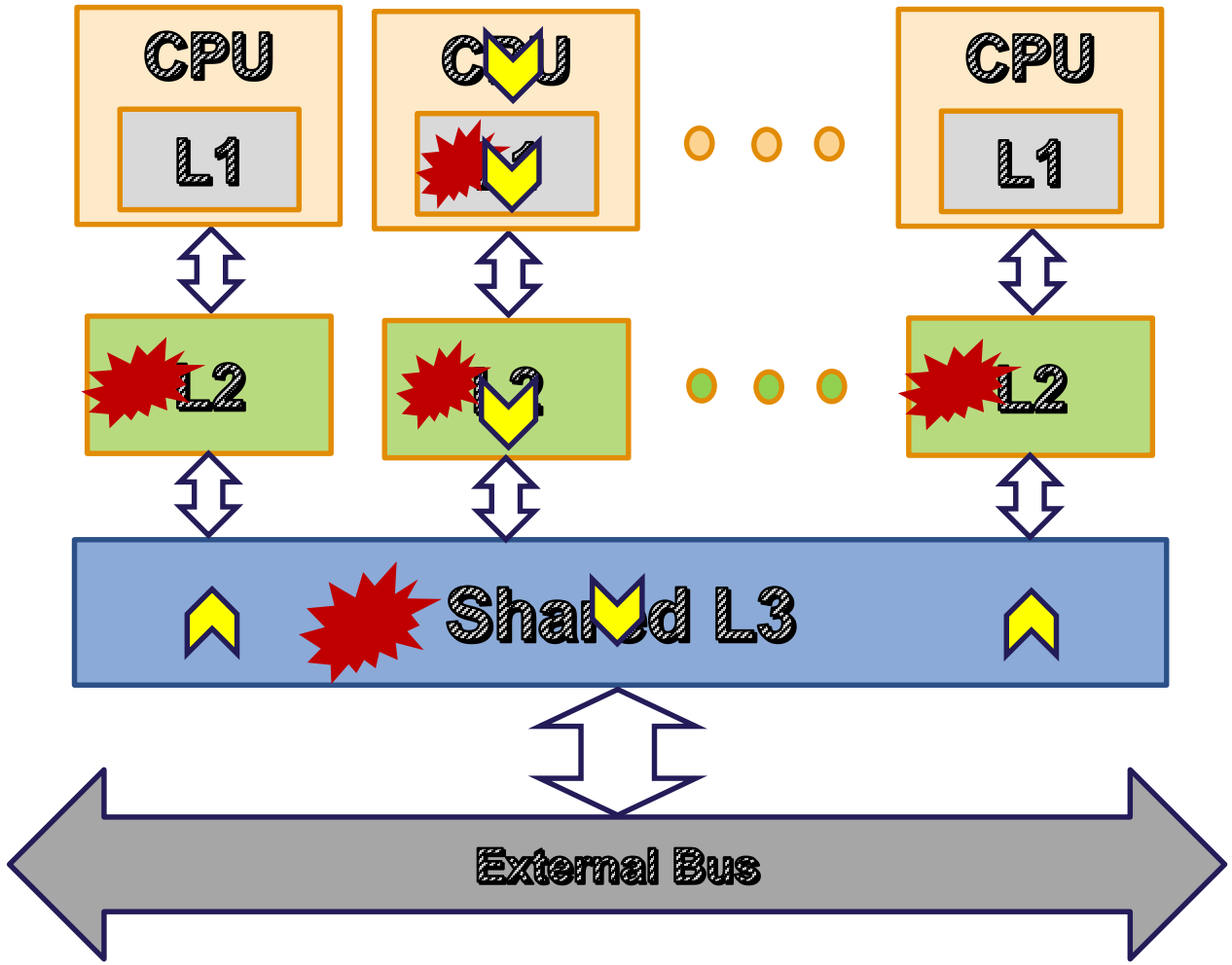
- Constrained Random Environment
  - Load, Evict, Snoop attributes Randomization
- UVM Virtual Sequences to compose stimulus across L1 and L3 Interfaces
- L1 & L3 cache models
  - to track MESI states accuracy
- Address Generator
  - To generate interesting colliding addresses
- Semaphore techniques
  - to model arbitration and resource sharing in UVM stimulus generation
- External Memory/Bus Response Delay Randomization

L2 Verification Challenges

- MESI State Bugs
  - Load, Evict, and Snoop dependencies
  - Out-of-order L3 cache miss responses
    - Response, Self-Snoop, Data, Hit-Assert
- Interface Protocol Errors
- Hangs, Timeouts, and Deadlocks
  - New load requests before MESI state update
    - L2 pipes occupied by new load requests from CPU
    - Potential deadlock as L2 is waiting for self-snoop and data to previous Cache-Miss to clear the pipes
    - Potential hangs/deadlocks overlooked due to Evicting-Snoops
  - New load requests after MESI state update
    - Out-of-order Cache-Miss response: Data arrives before Self-Snoop

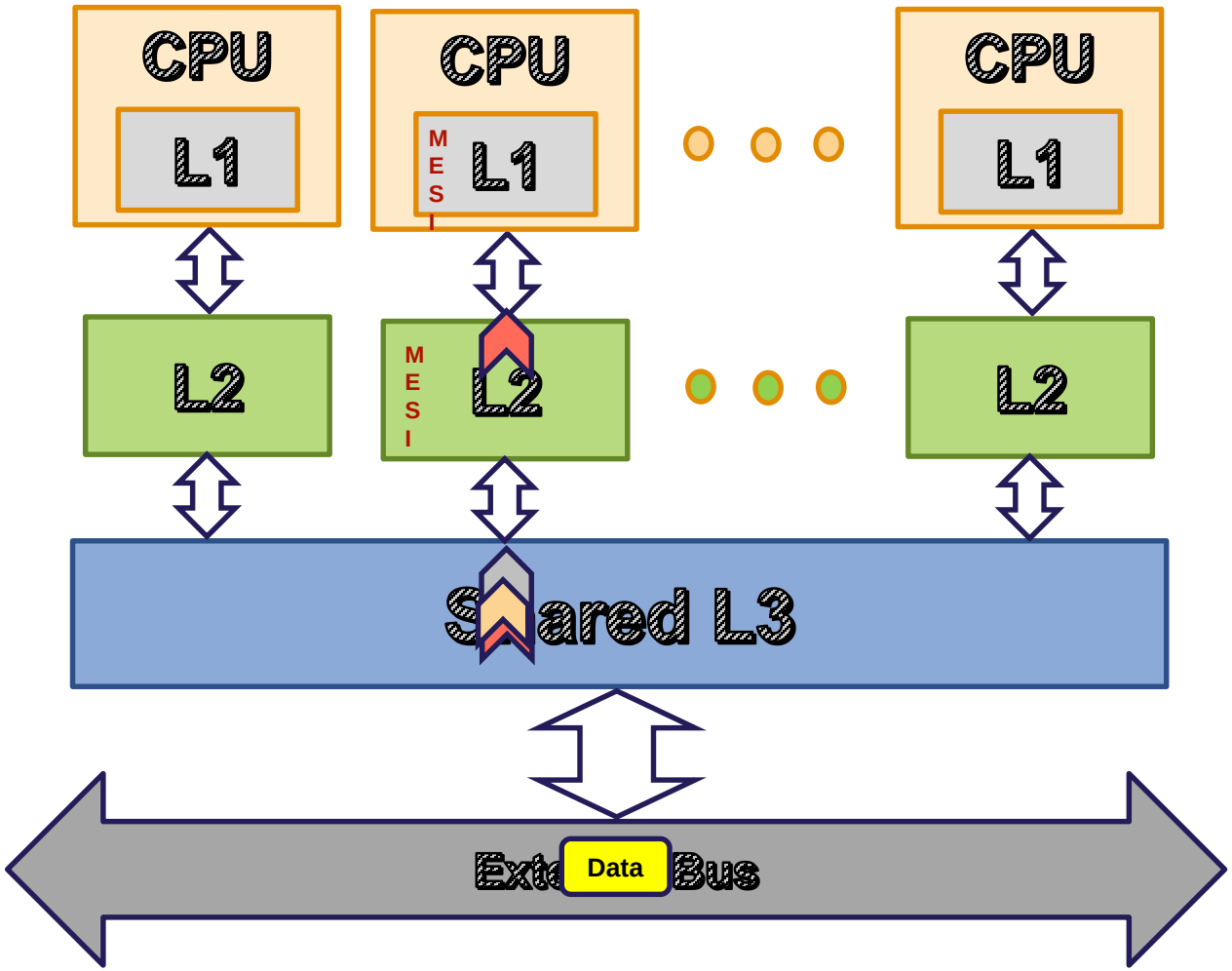
Multi-Core, Multi-Level Cache Architecture

Cache Line Miss Scenario



- L1, L2, and L3 cache hierarchy
- Load, Store, Evict, and Snoop ports
- Load request forwarded to other coherent agents with an L3 cache-miss
- Data received from other agents or the main memory

Out-of-Order Responses

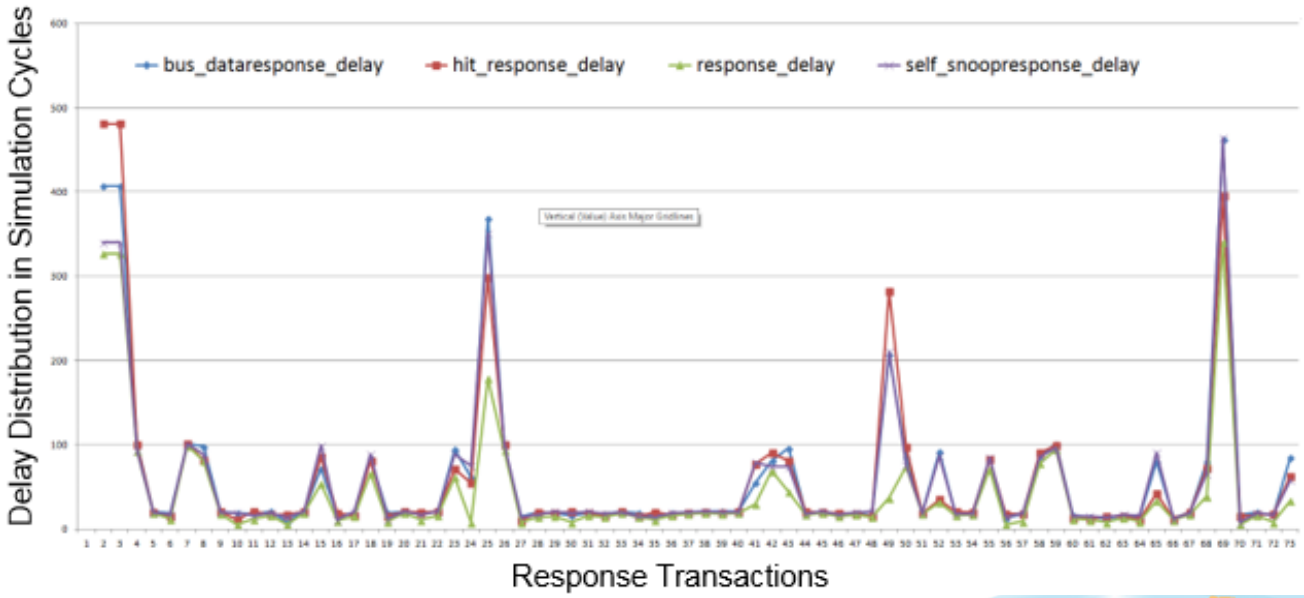


- Miss Load-Request results in four out-of-order responses
  - Response
    - MESI State Response
  - Self-snoop
    - Marks the latest MESI state
  - HitAssert
    - Data came from other coherent agents or external memory
  - Data

External Memory/Bus Response Delay Modeling

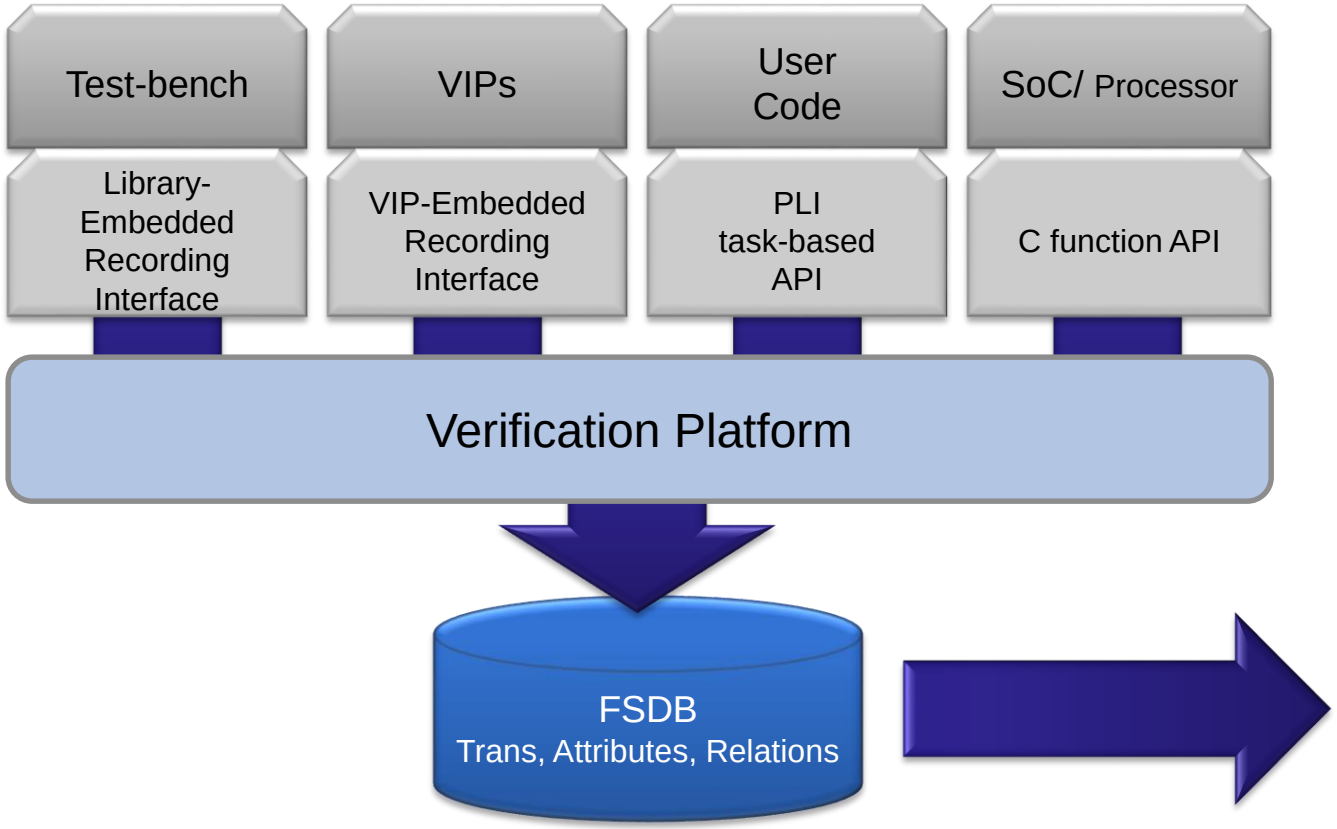
- Models out of order response delays distributed across three buckets: SHORT, MEDIUM, LONG
- SHORT, MEDIUM, LONG are biased with configurable weights: C1, C2, C3.
- Example Scenario
  - SHORT (5–20 Clocks) (C1=7->70%)
  - MEDIUM (5–100 Clocks) (C2=3->30%)
  - LONG (5–500 Clocks) (C2=1->10%)

```
class response_delay_biasing;
  rand int index;
  rand int delay [3];
  int c1 = 7;
  int c2 = 3;
  int c3 = 1;

  constraint distribution_c { index dist { 0 := c1, 1  := c2, 2 := c3 }; }
  //constraint delay_c { delay[0] == 70 && delay[1] == 40 && delay[2] == 10 ; }
  constraint delay_c { delay[0] < 20 && delay[1] < 100 && delay[2] < 500 &&
                       delay[0] > 5 && delay[1] > 5 && delay[2] > 5; }

  constraint order_c { solve index before delay ; }

  function new(int i_C1 = 1, int i_C2 = 1, int i_C3 = 1);
      this.c1 = i_C1;
      this.c2 = i_C2;
      this.c3 = i_C3;
  endfunction

  ..........
  ..........
endclass
```

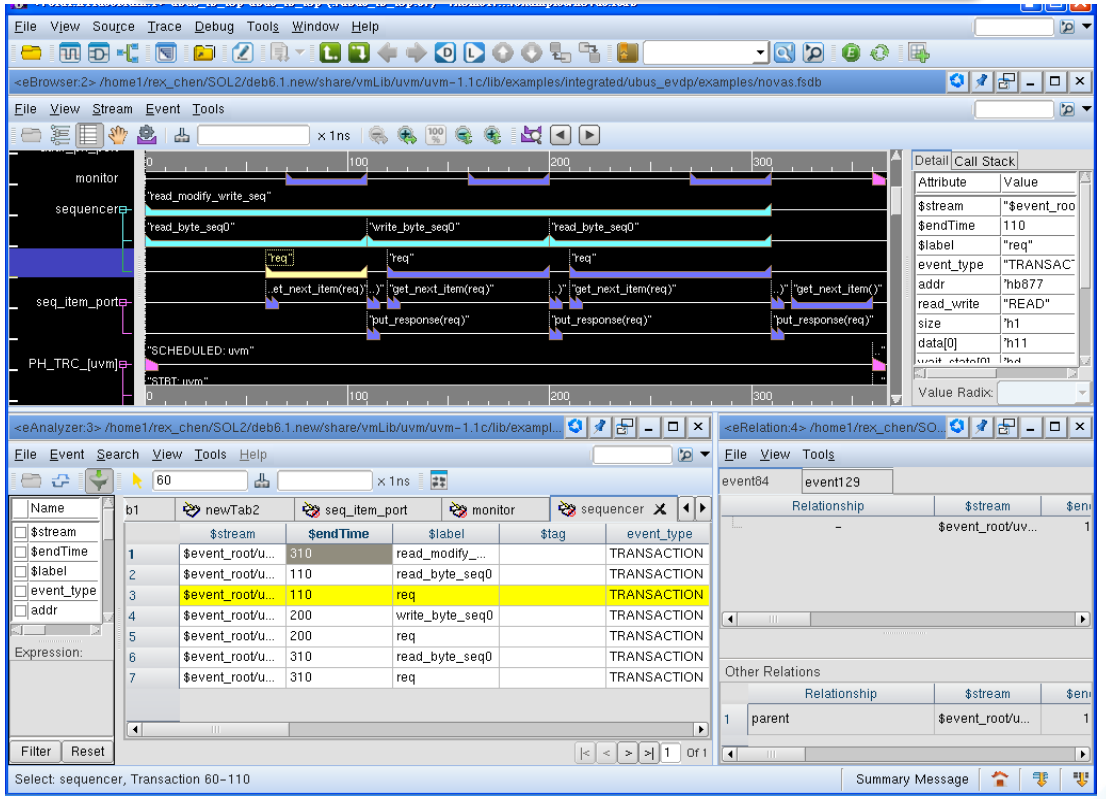


Verdi Transaction Debug

Verdi Transaction Debug



Verdi Transaction Based Debug



UVM Transactions Recording

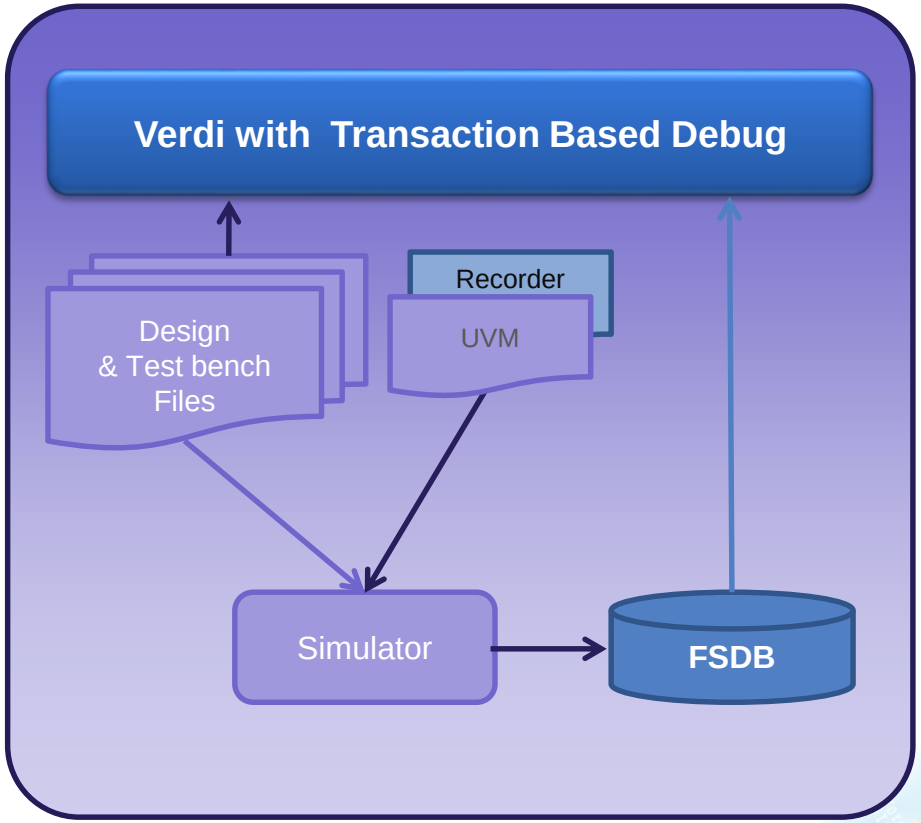
Automatic Transaction Recording

- Recorders Integrated inside Verdi UVM Libraries

- Automatic Recording of Transactions into FSDB
  - Sequencer Transactions
  - TLM Port Transactions
  - System Messages
  - UVM Messages

- No Requirements on User's Test bench code
  - Add following runtime switches

```
+UVM_VERDI_TRACE
+UVM_TR_RECORD
+UVM_LOG_RECORD
```



UVM Transactions Recording

Non-Sequence Transaction Recording

- Visibility into non-sequence components crucial to L2 verification
  - Drivers driving transactions into multiple interfaces
  - Monitors monitoring transactions received from multiple interfaces

- Verdi provides APIs to record non-sequence transactions

```
uvm_component::begin_tr()
uvm_component::begin_child_tr()
uvm_component::end_tr()
```

- begin_tr or begin_child_tr must be used in pair with end_tr

```
// Add at beginning of transaction for transaction record
    void'(this.begin_tr(tr));
    …
// Add at transaction end for transaction record
    this.end_tr(tr);
```

UVM Transactions Recording

L2 Driver Transaction Recording (1)

```
Class l2_l3_load_driver extend …;
  task run_phase (uvm_phase phase);
    …
    phase.raise_objection (this)
    fork
       get_load_request();
       send_load_response();
    join
    phose.drop_objection(this);
  endtask: run_phase

  task automatic send_load_response();
    …
    p_handle = this.begin_tr(load_req);
    load_req.p_handle = p_handle;
    …
    this.end_tr(load_req, 0, 0);
  end task
endclass
```

- Verdi APIs are called directly in the driver code to record transactions
- load_req modeled as a parent transaction

UVM Transactions Recording

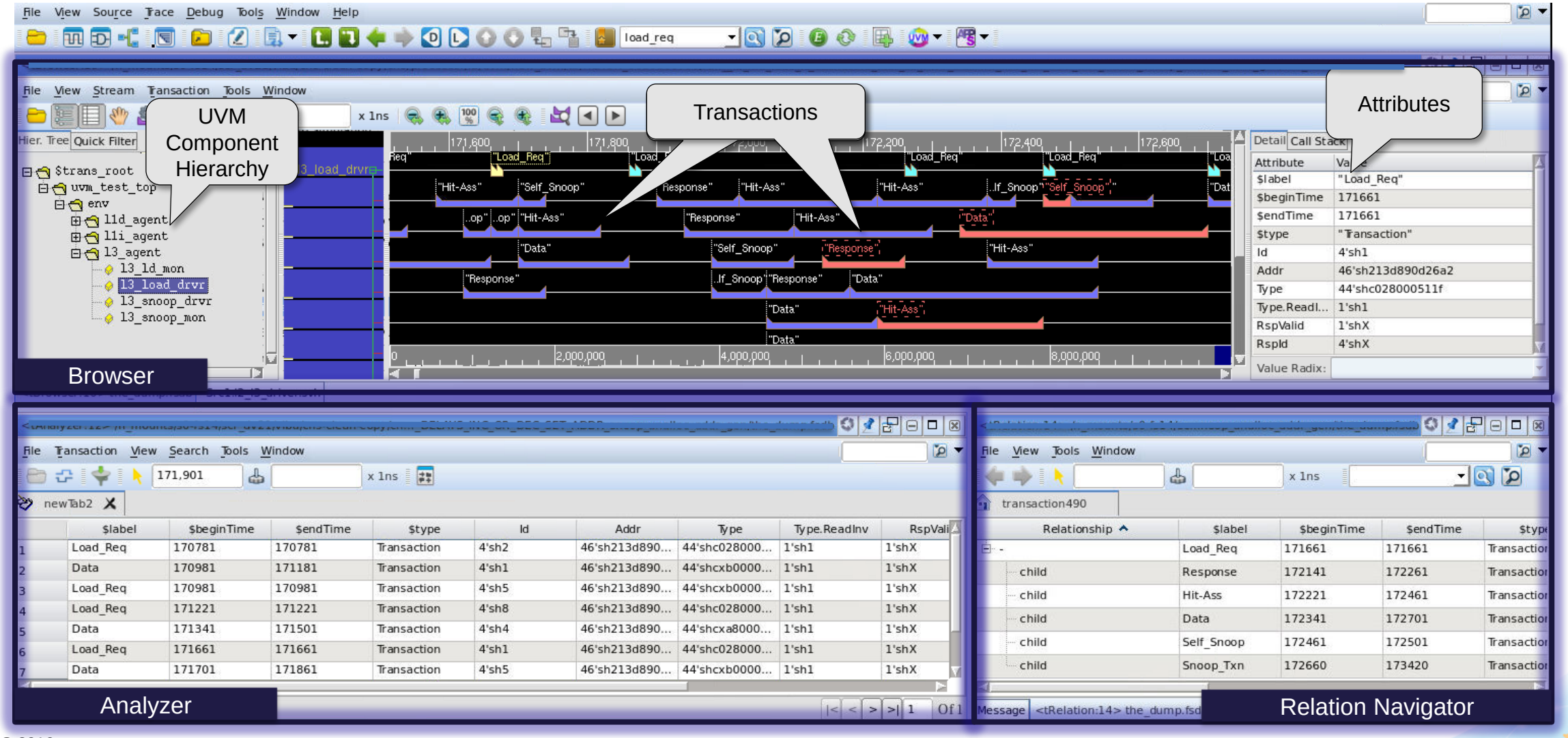
L2 Driver Transaction Recording (2)

```
virutal task automatic driver_self_snoop (…, input int
p_handler);
    …
      snoop_req =l2_l3_snoop_sequence_item::type_id::create("Snoop_Txn");
      ld_resp_t ld_resp;

      If (delay != 0) begin
        ld_resp = new("Self_Snoop");
        …
      ld_resp.p_handle = p_handle;
      this.begin_child_tr (ld_resp.l3_load, p_handle);
      ld_snoopQ.push_front(ld_resp);
    …
   end
endtask
```

```
task automatic driver_self_snoop_bus (…, integer
p_handle);

  l2_l3_snoop_sequence_item snoop_req;
  …
  ld_resp_t ld_resp_item;
  …
  snoop_req = l2_l3_snoop_sequence_item::type_id::create("Snoop_Txn");
  …
  this.end_tr(load_req, 0,0);

endtask
```

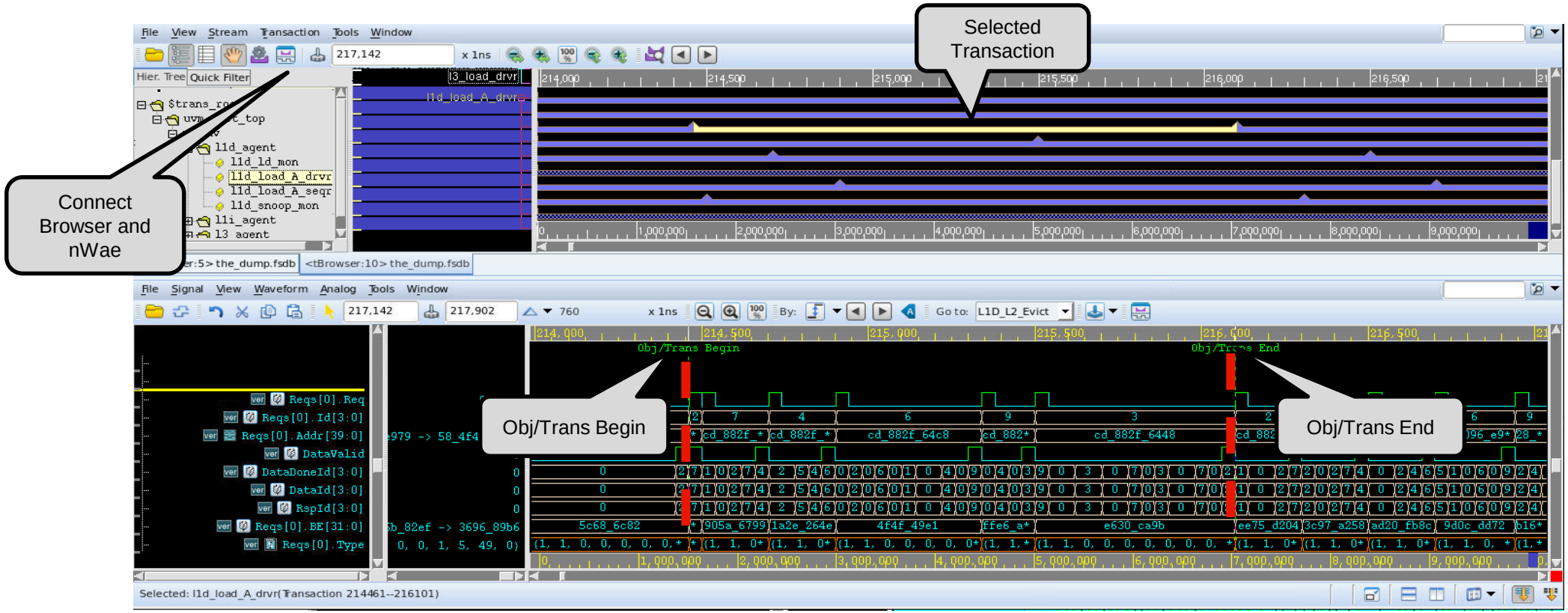
- Linking a child-transaction to a parent transaction

Verdi Transaction Debug



Transaction Browser & Waveform Viewer Correlation

- Builds a bridge between high level transactions and low level transactions (RTL)



- Quickly determine simulation time range when related RTL signals are active

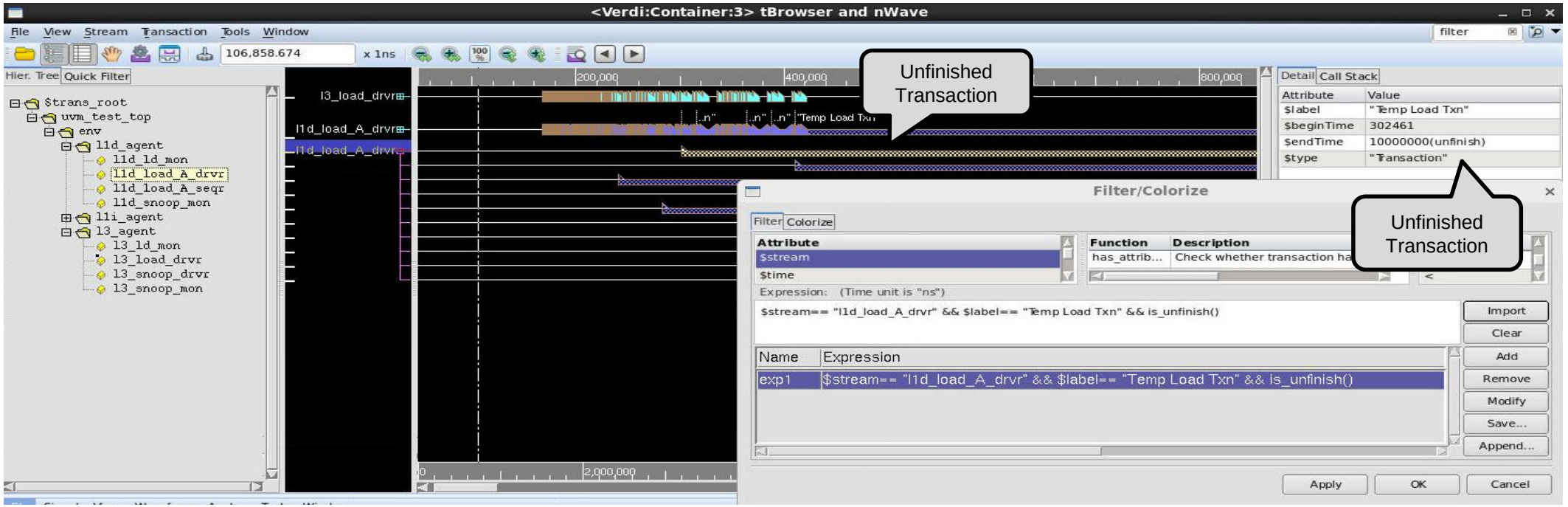
L2 Debug Scenarios

Unfinished Transactions

- Incomplete L1 transactions resulted in simulation hangs
- Difficult to discern culprit TB object or RTL component causing a hang
- Traditional debug methods are inefficient
  - Searching through UVM messages in log files for a clue
  - Relying on good debug messaging
  - Interactive Debug is slow, hence not ideal for debugging hangs that occur hours into simulation

Visualizing Unfinished Transactions

- Special display pattern in Browser pane and ‘unfinished’ suffix in Attribute pane indicate unfinished transactions
- Powerful filtering mechanism isolate unfinished transactions



- Quickly diagnose hang/deadlocks by isolating unfinished transactions and correlating RTL code and signals

Missing Transactions

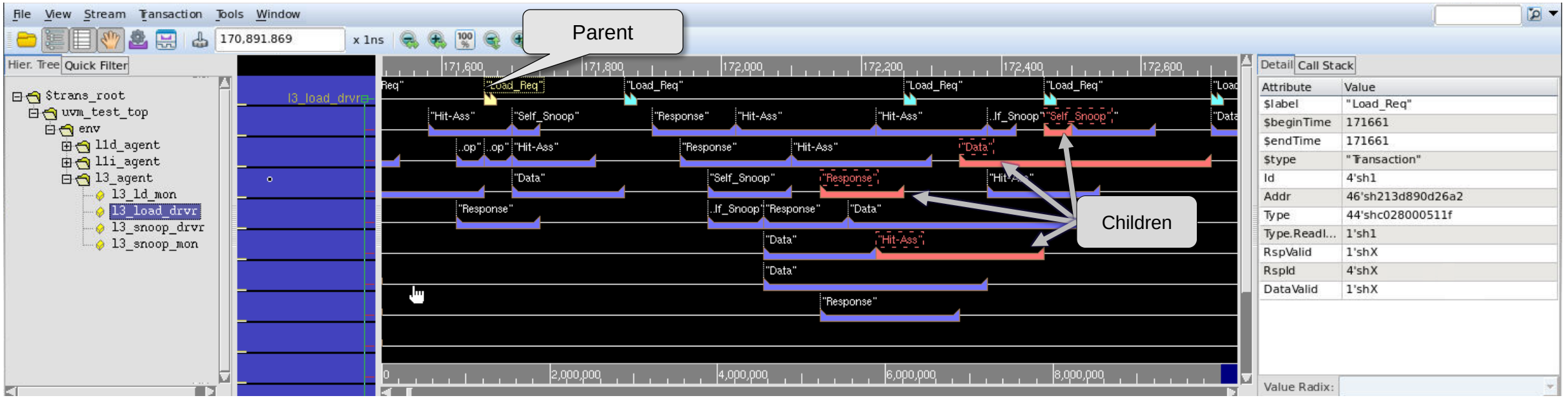
- A missing response-transaction resulted in regression failures
  - An L3 request to the external bus expects four responses:
    - Response, Self-Snoop, HitAssert, and Data
    - Response Transactions are received out-of-order and are spread over 100s of clock cycles

- Traditional debug methods are tedious and inefficient
  - Searching log files
  - Relying on good debug messaging
  - Manually linking transactions

Verdi Transaction Browser

Missing Transactions

- Highlighted linked-transactions provided an easy way to quickly find the missing transaction

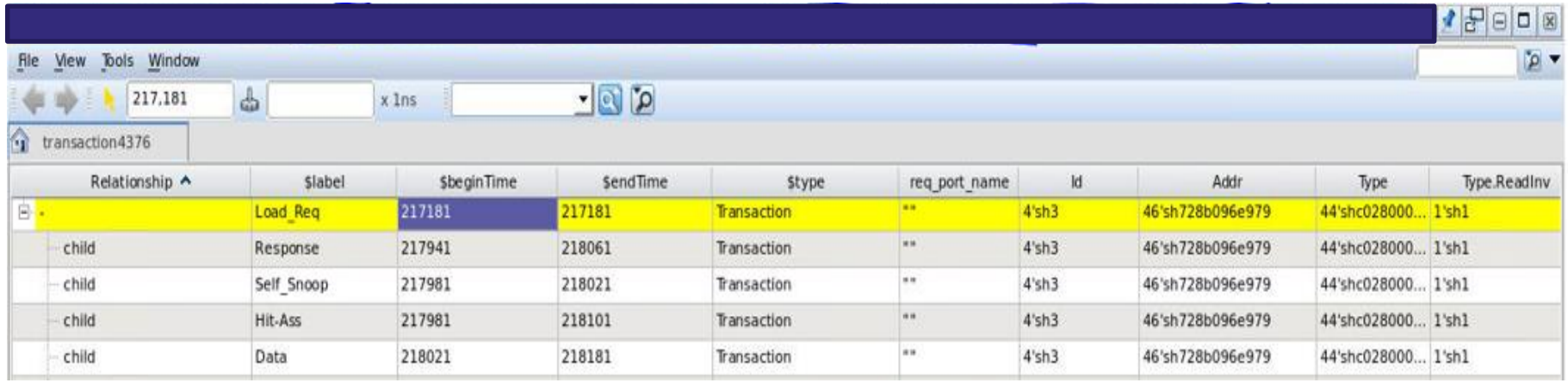


- Visualization of related transaction helps to quickly find parent-child transaction spread across hundred clock cycle

Verdi Transaction-Relation Navigator

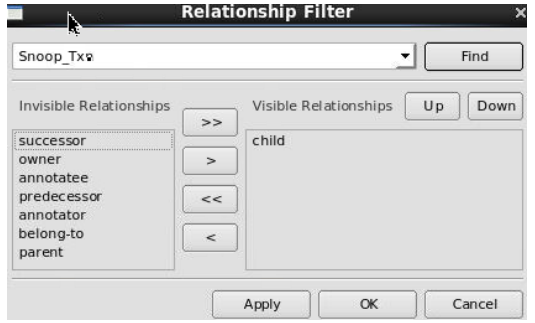
Missing Transactions

- Spreadsheet-like view to quickly trace the relations and find the missing transactions
  - Transactions listed in a relation-hierarchy format
  - Attributes of each transaction are displayed



Relationship	$label	$beginTime	$endTime	$type	req_port_name	Id	Addr	Type	Type.ReadInv
-	Load_Req	217181	217181	Transaction	""	4'sh3	46'sh728b096e979	44'shc028000...	1'sh1
child	Response	217941	218061	Transaction	""	4'sh3	46'sh728b096e979	44'shc028000...	1'sh1
child	Self_Snoop	217981	218021	Transaction	""	4'sh3	46'sh728b096e979	44'shc028000...	1'sh1
child	Hit-Ass	217981	218101	Transaction	""	4'sh3	46'sh728b096e979	44'shc028000...	1'sh1
child	Data	218021	218181	Transaction	""	4'sh3	46'sh728b096e979	44'shc028000...	1'sh1

- Filtering mechanism provided to customize the view



Wish List For Synopsys

- Instrument warning messages in simulation log file to indicate unfinished transactions
  - Quickly confirm/rule out existence of unfinished transactions in simulation without invoking GUI
  - Implemented and delivered in Verdi 2015.09-SP2

- Display attributes of unfinished transactions
  - attributes for unfinished transactions are not being recorded since *end_tr* is not reached
  - Record attributes before transaction end (*end_tr()*)
    - Will populate attribute pane even for unfinished transactions

Summary

- Unit level UVM based constrained random-stimulus generation exposed hard to reach corner case bugs
  - MESI state, hangs, and timeouts

- Verdi Transaction Debug key to debugging complex bugs in a high traffic and complex interface environment
  - Debugging transactions at higher level of abstraction
  - Quickly discern incomplete transactions in Transaction Browser with special display pattern
  - Quickly determine time range where RTL signals are active with Transaction Browser and Waveform Viewer Correlation
  - Quickly find parent-child transaction relations with simple UVM API hooks *begin_child_tr* and *end_tr*

CenTaur technology

snug
Synopsys Users Group
Austin

Thank You