2024
DESIGN AND VERIFICATION™
DVCON
CONFERENCE AND EXHIBITION
UNITED STATES
SAN JOSE, CA, USA
MARCH 4-7, 2024

A UVM SystemVerilog Testbench for Directed & Random Testing of an AMS LDO Voltage Regulator

Charles Dančak, Betasoft Consulting, Santa Clara

Betasoft
Consulting

accellera
SYSTEMS INITIATIVE

Verify an LDO Regulator's Functions

```
--Repeat till no more vectors:
  while (VECTORS_LEFT > 0) do
--Wait for trigger from system:
  TRIG_RECD = DIGIO.trigger[3]
  delay(0.0005) --Settling time.
  .  .  .  .
```

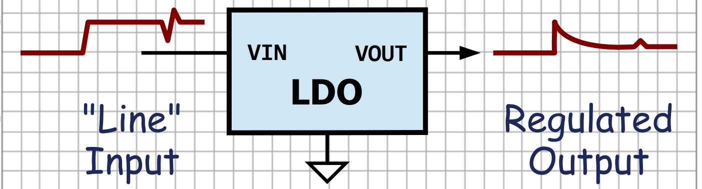
Eight Directed Tests:

- Line/load regulation.
- Line/load transient.
- PSRR (hum); IDDQ.
- Trim VOUT in steps.
- Control modes.

TITLE: 1

Verification Goals

- A UVM testbench for an AMS DUT.
- Run 8 benchtop-style directed tests to verify that voltage is regulated.
- Randomize a sequence of such tests to reach unanticipated corner cases.



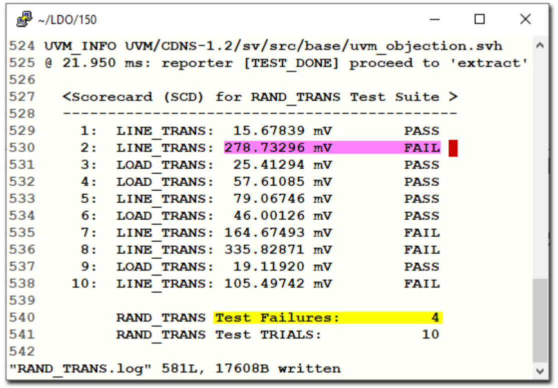
Coding Tactics

- Factory drivers/monitors to isolate UVM classes from testing details.
- UVM events for testbench timing.
- Use randsequence to generate test sequence, with no random instability.

SIGNATURE : DATE : WITNESSED BY : DATE :

LDO Regulator Specs

Test Type	Design Requirement
LINE_REGLN	[ΔVOUT / ΔVDD18] ÷ 1.8 ≤ 5%/V
. . . .	. . . .
PSRR_RATIO	20 log(HUM_{out}/HUM_{in}) ≤ −5 dB
IDDQ_LEVEL	Iddq ≤ 1000 µA
LINE_TRANS	Vover, Vunder ≤ 100 mV
LOAD_TRANS	



```
UVM_INFO UVM/CDNS-1.2/sv/src/base/uvm_objection.svh
@ 21.950 ms: reporter [TEST_DONE] proceed to 'extract'

  <Scorecard (SCD) for RAND_TRANS Test Suite >
  -----------------------------------------------
     1:  LINE_TRANS:  15.67839 mV        PASS
     2:  LINE_TRANS: 278.73296 mV        FAIL
     3:  LOAD_TRANS:  25.41294 mV        PASS
     4:  LOAD_TRANS:  57.61085 mV        PASS
     5:  LINE_TRANS:  79.06746 mV        PASS
     6:  LOAD_TRANS:  46.00126 mV        PASS
     7:  LINE_TRANS: 164.67493 mV        FAIL
     8:  LINE_TRANS: 335.82871 mV        FAIL
     9:  LOAD_TRANS:  19.11920 mV        PASS
    10:  LINE_TRANS: 105.49742 mV        FAIL

         RAND_TRANS Test Failures:          4
         RAND_TRANS Test TRIALS:           10

"RAND_TRANS.log" 581L, 17608B written
```

Sequence Transcript

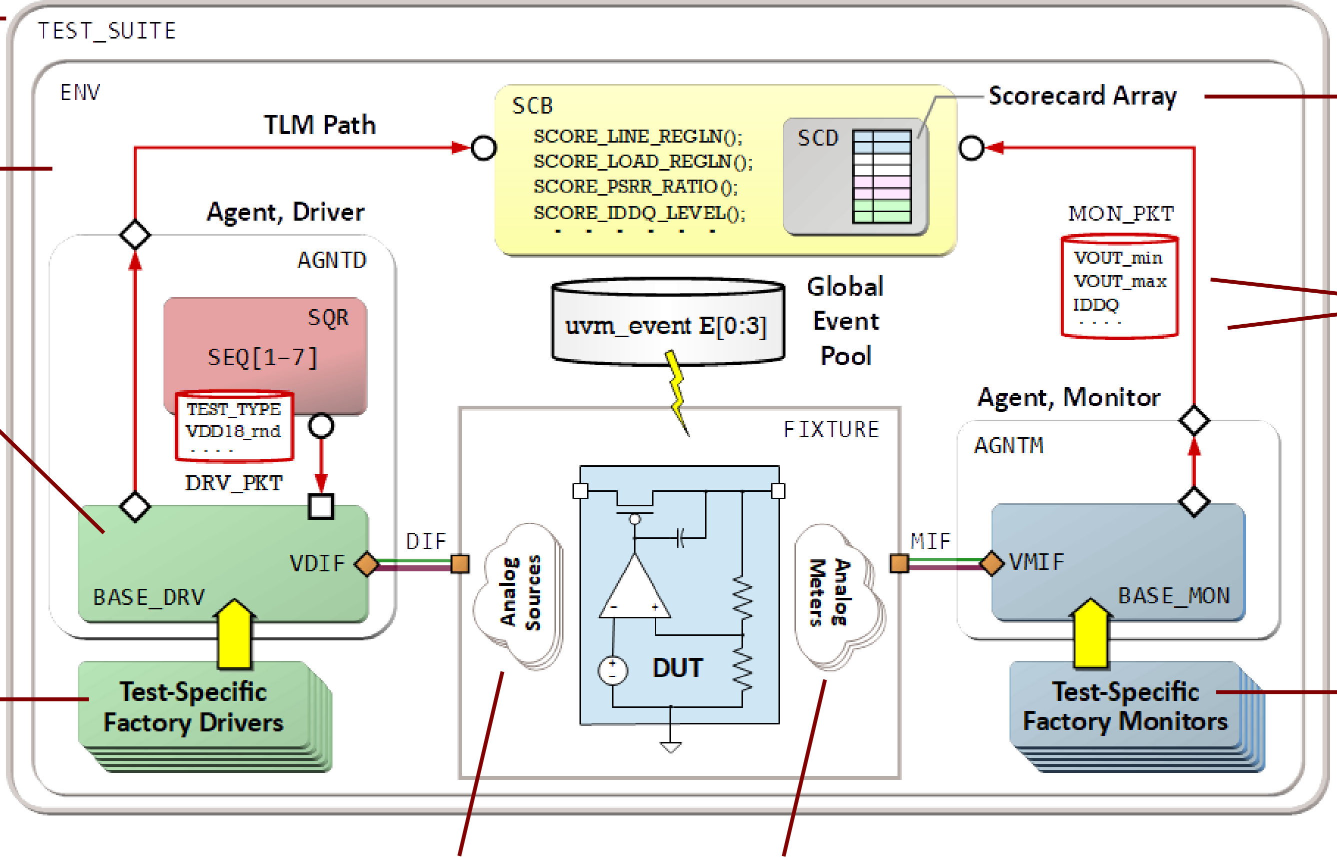
UVM Testbench Organization

Top-Level Test

Environment Isolated from Low-Level Tests

Test-Neutral Base Driver

Test-Specific UVM Factory Replacement Drivers



Sequence Transcript Scorecard

DRV and MON Packets Per Directed Test

Factory Replacement Monitors for Response

Analog Stimulus

Analog Response

LDO Voltage Regulator DUT

Unregulated Line [~1.8 V]

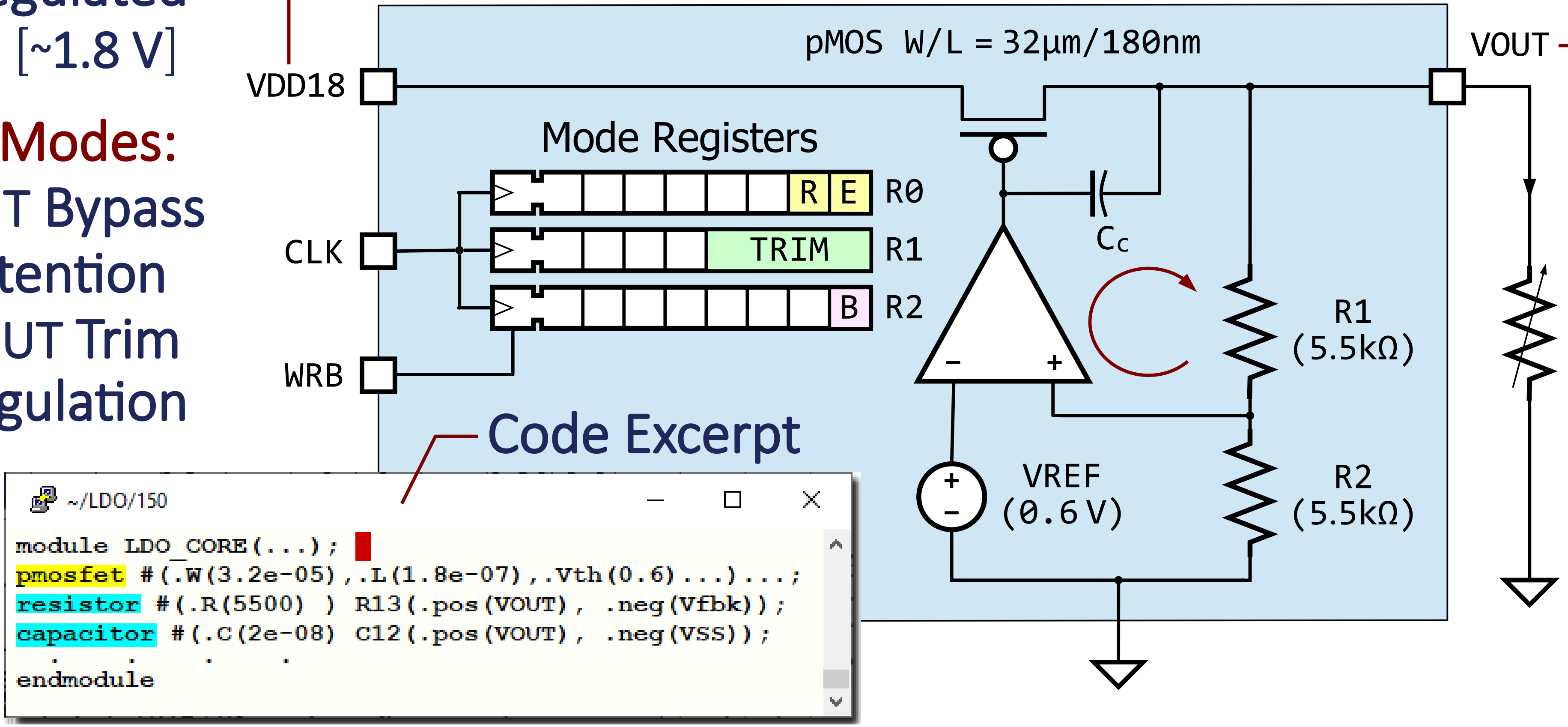
LDO Modes:

- DUT Bypass
- Retention
- VOUT Trim
- Regulation

Regulated Output

Adjustable Current Load

Code Excerpt



```
module LDO_CORE(...);
pmosfet #(.W(3.2e-05),.L(1.8e-07),.Vth(0.6)...)...;
resistor #(.R(5500) ) R13(.pos(VOUT), .neg(Vfbk));
capacitor #(.C(2e-08) C12(.pos(VOUT), .neg(VSS));
  .   .   .   .
endmodule
```

DUT is modeled in pure SystemVerilog, using parameterized XMODEL primitives.

Enumerated Type Specifies Active Test

Idle After Each Test

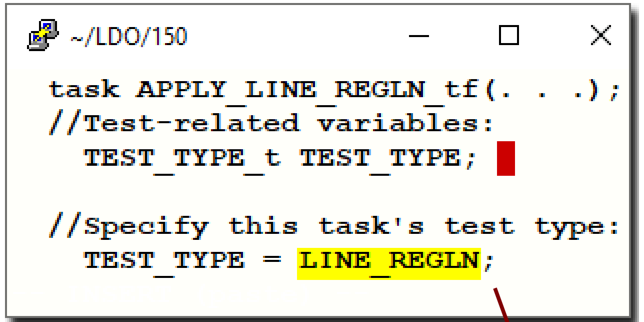
Individual Directed Tests

Random Sequence

Voltage Regulator Test Types		
Code	**Test Type**	**Analog Quantity Measured**
4'h0	IDLE_SUITE	Default enumerated test type.
4'h1	LINE_REGLN	*ΔVOUT*, as *VDD18* varies from min to max over input range.
4'h2	LOAD_REGLN	*ΔVOUT*, as *ILOAD* varies from max to min over load range.
4'h3	PSRR_RATIO	Rejection of 10-kHz input hum.
4'h4	IDDQ_LEVEL	Quiescent *IDDQ* under low load.
4'h5	LINE_TRANS	*VOUT* overshoot, undershoot.
4'h6	LOAD_TRANS	*VOUT* undershoot, overshoot.
4'h7	TRIM_LEVEL	Trimmed *VOUT* levels [*n* = 1–16].
4'h8	CTRL_MODES	*VOUT* for ENA, RET, BYP modes.
4'h9	RAND_TRANS	Run LINE_TRANS, LOAD_TRANS tests in a random sequence.

```
/* ENUMERATED TEST_TYPE
 * Declared in package TYPES.
 */
 typedef
   enum bit [3:0] {
   //-Test Type- -Code-
     IDLE_SUITE = 4'h0, //Default.
     LINE_REGLN = 4'h1,
     LOAD_REGLN = 4'h2,
     PSRR_RATIO = 4'h3,
     IDDQ_LEVEL = 4'h4,
     LINE_TRANS = 4'h5,
     LOAD_TRANS = 4'h6,
     TRIM_LEVEL = 4'h7,
     CTRL_MODES = 4'h8,
     RAND_TRANS = 4'h9
   } TEST_TYPE_t;
```

Testbench-Wide Test Types



```
task APPLY_LINE_REGLN_tf(. . .);
//Test-related variables:
  TEST_TYPE_t TEST_TYPE;

//Specify this task's test type:
  TEST_TYPE = LINE_REGLN;
```

Set Active Test

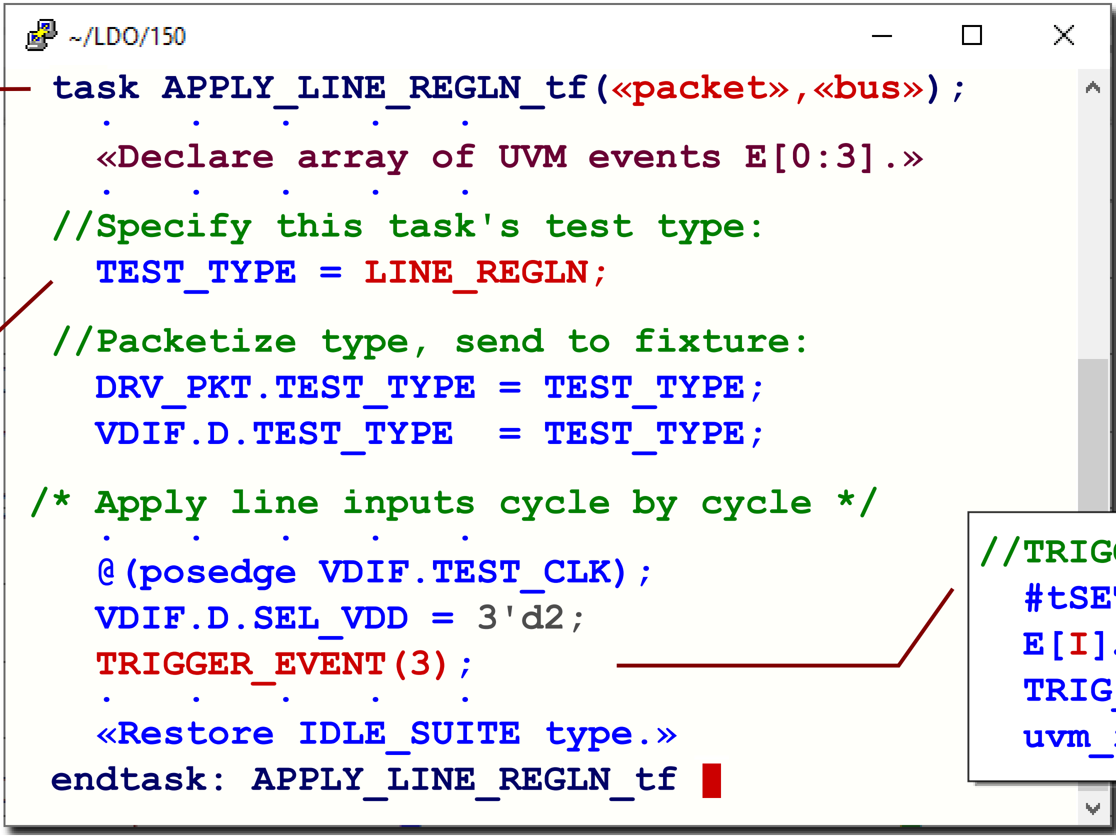
A Driver-Side Command Task

Factory Driver Calls This Task

Test-Specific Factory Driver

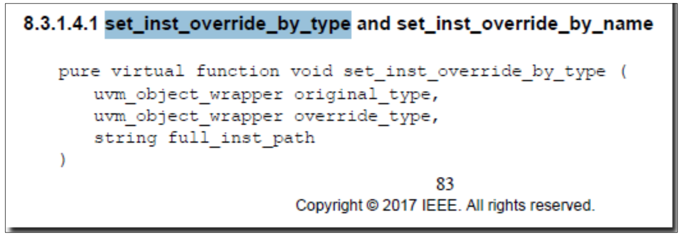
Sets Test to **LINE_REGLN**

Driver Activity During a Cycle



```
task APPLY_LINE_REGLN_tf(«packet»,«bus»);
   .   .   .   .   .
   «Declare array of UVM events E[0:3].»
   .   .   .   .   .
 //Specify this task's test type:
   TEST_TYPE = LINE_REGLN;

 //Packetize type, send to fixture:
   DRV_PKT.TEST_TYPE = TEST_TYPE;
   VDIF.D.TEST_TYPE  = TEST_TYPE;

/* Apply line inputs cycle by cycle */
   .   .   .   .   .
   @(posedge VDIF.TEST_CLK);
   VDIF.D.SEL_VDD = 3'd2;
   TRIGGER_EVENT(3);
   .   .   .   .   .
   «Restore IDLE_SUITE type.»
endtask: APPLY_LINE_REGLN_tf
```

Factory Replacement



8.3.1.4.1 set_inst_override_by_type and set_inst_override_by_name

```
pure virtual function void set_inst_override_by_type (
   uvm_object_wrapper original_type,
   uvm_object_wrapper override_type,
   string full_inst_path
)
```

83

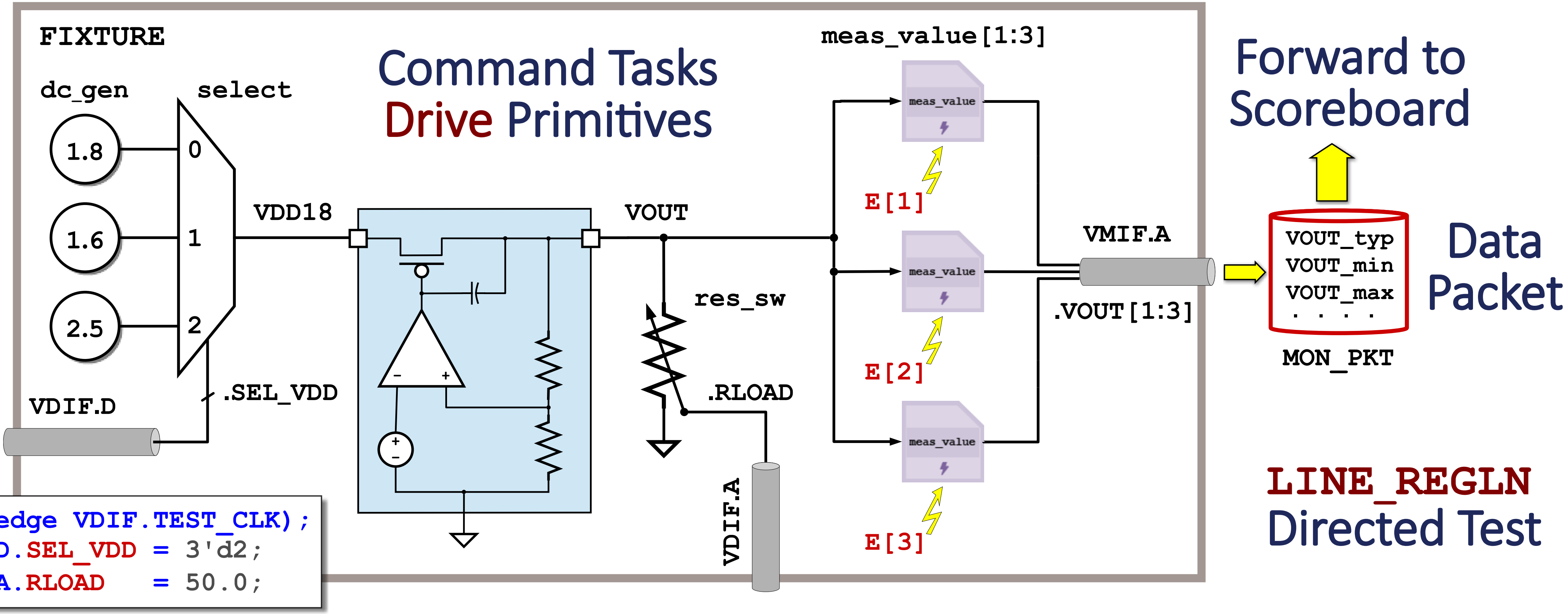
Copyright © 2017 IEEE. All rights reserved.

Test-Related Command

```
//TRIGGER_EVENT() command code:
  #tSETTLE;   //Analog settling time.
  E[I].trigger(); //Trigger UVM event.
  TRIG_TIME = E[I].get_trigger_time();
  uvm_report_info(...);
```

Driver-side command task applies test-specific stimuli to DUT for a directed test.

Analog Sources and Meters



Select Line Input

Command Tasks Drive Primitives

Forward to Scoreboard

Data Packet

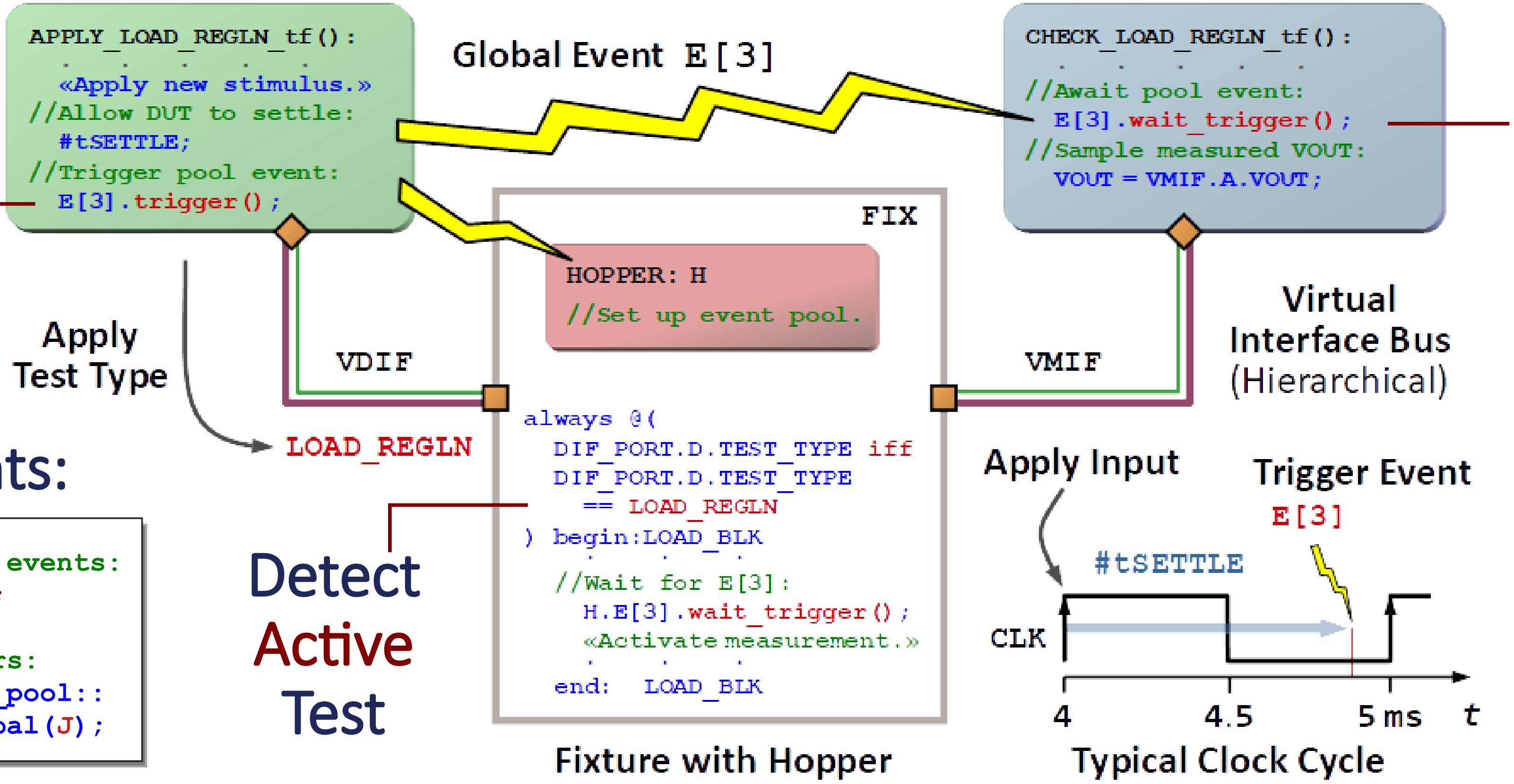
LINE_REGLN Directed Test

```
@(posedge VDIF.TEST_CLK);
VDIF.D.SEL_VDD = 3'd2;
VDIF.A.RLOAD    = 50.0;
```

Fixture instantiates XMODEL primitives: `dc_gen`, `select`, adjustable `res_sw`, …

Sending Events Across the Testbench

Driver Task
Triggers Each
UVM Event



```
APPLY_LOAD_REGLN_tf():
   .   .   .   .   .
   «Apply new stimulus.»
//Allow DUT to settle:
   #tSETTLE;
//Trigger pool event:
   E[3].trigger();
```

```
CHECK_LOAD_REGLN_tf():
   .   .   .   .   .
//Await pool event:
   E[3].wait_trigger();
//Sample measured VOUT:
   VOUT = VMIF.A.VOUT;
```

```
HOPPER: H
//Set up event pool.
```

```
always @(
  DIF_PORT.D.TEST_TYPE iff
  DIF_PORT.D.TEST_TYPE
    == LOAD_REGLN
) begin:LOAD_BLK
   .   .   .
  //Wait for E[3]:
    H.E[3].wait_trigger();
    «Activate measurement.»
    .   .   .
  end:  LOAD_BLK
```

Monitor Task
Awaits Event

Detect
Active
Test

Timing
Paradigm

Declare Events:

```
//Declare array of events:
  uvm_event E[0:3];
  foreach (E[J])
//Get event pointers:
  E[J] = uvm_event_pool::
              get_global(J);
```

Global UVM event array provides crisply-synchronized stimulus/response timing.

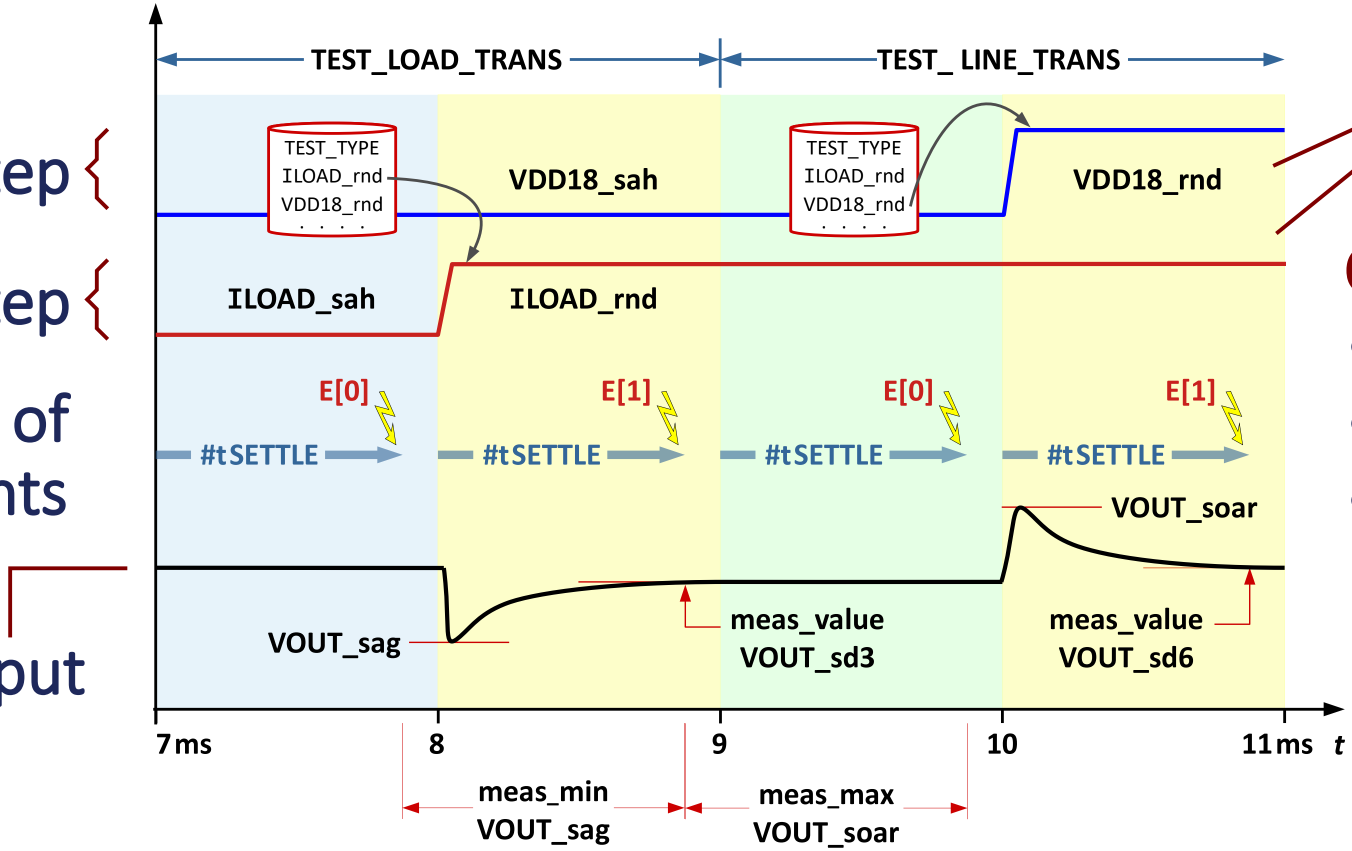
Random Line/Load-Transient Sequence

Random **Line** Step {

Random **Load** Step {

Triggering of Array Events

Regulated Output



Abrupt Change in Line or Load

Causes of Failure:

- Slow feedback loop.
- Loop instabilities.
- Other design flaws.

Alternating **RAND_TRANS** Test Suite

DUT may **pass** each directed test—yet **fail** to regulate under consecutive tests.

Generating the Alternating Sequence

Top-Level
Test Class

```
//RAND_TRANS constructor:
  function new(. . .);
   .   .   .   .
  //Generate random ORDER of test types:
    for (int I = 1; I <= TRIALS; I++)
    begin:RS_LOOP
      randsequence(TRANSIENT)
        TRANSIENT:  RS_LINE_TRANS := 7
                  | RS_LOAD_TRANS := 3;
        RS_LINE_TRANS: {ORDER[I] = LINE_TRANS;};
        RS_LOAD_TRANS: {ORDER[I] = LOAD_TRANS;};
      endsequence

    end:  RS_LOOP
  endfunction: new
```

Two
Rules

Loop

Rule's Action:
Append test type.

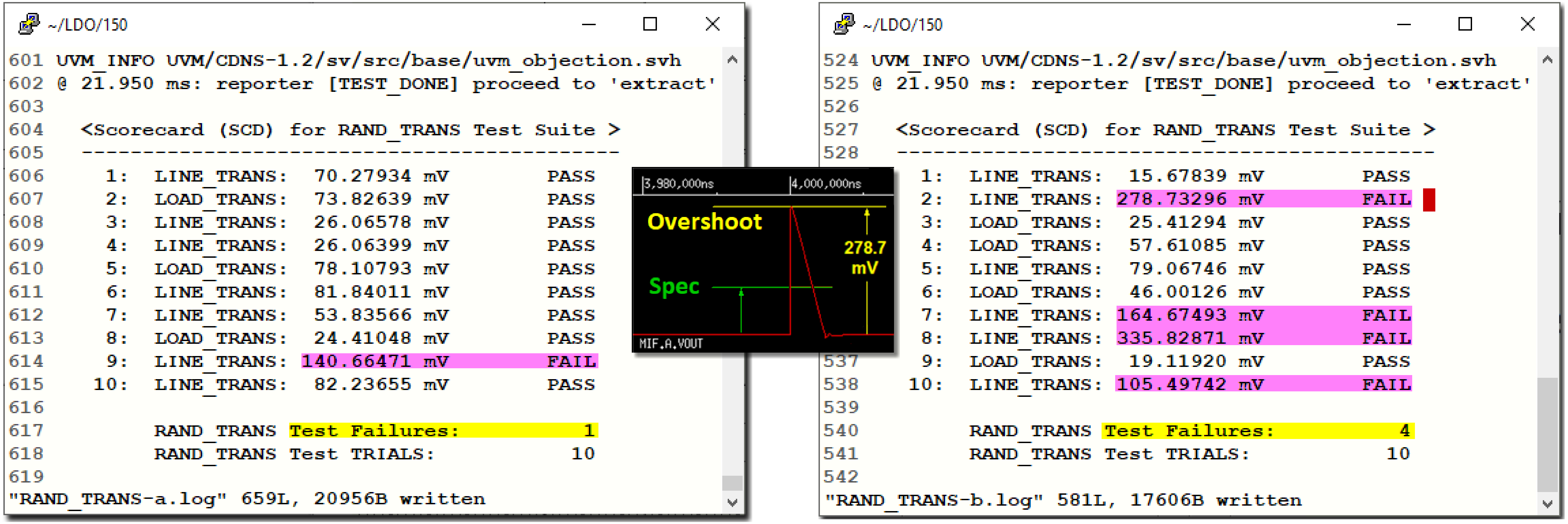
```
/* FACTORY RAND_TRANS_DRIVER CODE
 * Alternate two driver command tasks
 * in prescribed randsequence ORDER:
 */
foreach (ORDER[T])
begin:XORDER
  «get_next_item(DRV_PKT)»
  .   .   .   .
//Command task by type:
  case (ORDER[T])
    LINE_TRANS:
      APPLY_LINE_TRANS_tf(...);
    LOAD_TRANS:
      APPLY_LOAD_TRANS_tf(...);
  endcase
  .   .   .   .
end:  XORDER
```

Same Array
Passed to
DRV, **MON**

Alternate
Tests

Loop shown builds a random array **ORDER** of alternating line or load test types.

Simulated **RAND_TRANS** Results



```
UVM_INFO UVM/CDNS-1.2/sv/src/base/uvm_objection.svh
@ 21.950 ms: reporter [TEST_DONE] proceed to 'extract'

  <Scorecard (SCD) for RAND_TRANS Test Suite >
  -------------------------------------------
     1:  LINE_TRANS:  70.27934 mV         PASS
     2:  LOAD_TRANS:  73.82639 mV         PASS
     3:  LINE_TRANS:  26.06578 mV         PASS
     4:  LINE_TRANS:  26.06399 mV         PASS
     5:  LOAD_TRANS:  78.10793 mV         PASS
     6:  LINE_TRANS:  81.84011 mV         PASS
     7:  LINE_TRANS:  53.83566 mV         PASS
     8:  LOAD_TRANS:  24.41048 mV         PASS
     9:  LINE_TRANS: 140.66471 mV         FAIL
    10:  LINE_TRANS:  82.23655 mV         PASS

         RAND_TRANS Test Failures:           1
         RAND_TRANS Test TRIALS:            10

"RAND_TRANS-a.log" 659L, 20956B written
```

-svseed 9437

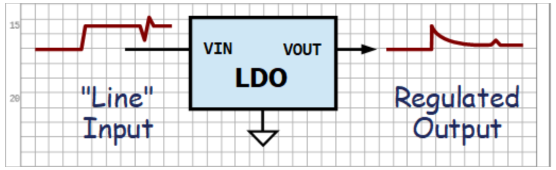
```
UVM_INFO UVM/CDNS-1.2/sv/src/base/uvm_objection.svh
@ 21.950 ms: reporter [TEST_DONE] proceed to 'extract'

  <Scorecard (SCD) for RAND_TRANS Test Suite >
  -------------------------------------------
     1:  LINE_TRANS:  15.67839 mV         PASS
     2:  LINE_TRANS: 278.73296 mV         FAIL
     3:  LOAD_TRANS:  25.41294 mV         PASS
     4:  LOAD_TRANS:  57.61085 mV         PASS
     5:  LINE_TRANS:  79.06746 mV         PASS
     6:  LOAD_TRANS:  46.00126 mV         PASS
     7:  LINE_TRANS: 164.67493 mV         FAIL
     8:  LINE_TRANS: 335.82871 mV         FAIL
     9:  LOAD_TRANS:  19.11920 mV         PASS
    10:  LINE_TRANS: 105.49742 mV         FAIL

         RAND_TRANS Test Failures:           4
         RAND_TRANS Test TRIALS:            10

"RAND_TRANS-b.log" 581L, 17606B written
```

-svseed 3947

Varying random seed changes **ORDER**, and may yield more failures: 4 of 10 trials.

Conclusions and Questions

- Analog/mixed-signal DUT has been verified by a UVM-compliant testbench, including eight low-level benchtop-style tests, and a random-order test suite.



- Isolated UVM components from test-specific details via factory and other tactics.
- Reached unanticipated corner cases by randomly alternating transient tests. Generated alternating series using `randsequence` to build a test-type array.
- XMODEL-based flow verified AMS DUT with standard UVM testbench—entirely in SystemVerilog, using a commercial logic simulator such as Xcelium.

Click to download UVM testbench code package!