



Simulation-Based FlexRay™ 3.0 Conformance Testing – An OVM Success Story



Jason Sprott, Verilab
CTO

DAC 2010, OVM Booth

Agenda

- Verilab Introduction
- The FlexRay Project
- What OVM Brings to the Party
- Layered Sequences
- Using the Factory for Configurations
- Some Project Development Statistics
- Conclusions

About Verilab

- Verification consulting
- Turnkey verification and VIP development
- Staff augmentation and team mentoring
- Over 150 verification projects since 2000
- US, UK, Germany
- Driving verification standards (IEEE, Accellera)

All verification, all the time!

verilab



Ethernet

TCP/IP



C/C++

Automotive

FBDIMM

Vera



Comms

WiMax

Military/Defense



VIP



MEMBIST

Training

Methodology

CPU

Ref Model/Spec Validation

AXI

DSP



Verilog/VHDL

OCP

DDR3

Video

Specman/e

3G Wireless

Audio

What is FlexRay?

“The FlexRay™ Communications System is a robust, scalable, deterministic, and fault-tolerant serial bus system designed for use in automotive applications”

- Basic characteristics include:
 - synchronous and asynchronous frame transfer
 - guaranteed frame latency and jitter
 - prioritization of frames
 - single or multi-master clock synchronization
 - time synchronization across multiple networks
 - error detection and signalling
 - scalable fault tolerance
- Supports new applications such as drive-by-wire

The Problem

- Hardware based conformance testing only
- **Specification and tests validated late**
- Expensive and slow for end users to setup
- Conformance testing late in development cycle
- Difficult to debug issues

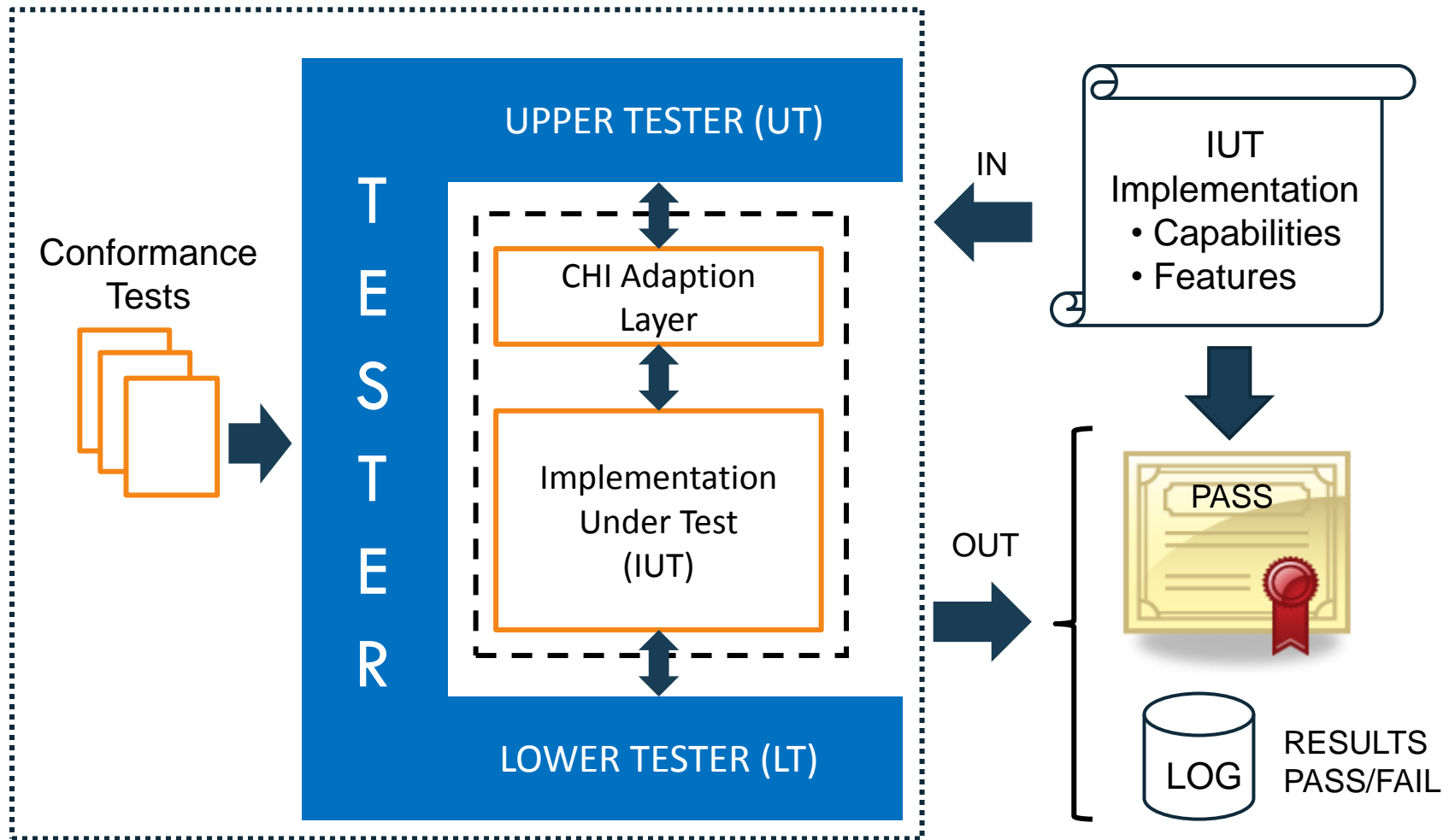
Too late and costs too much

The Solution

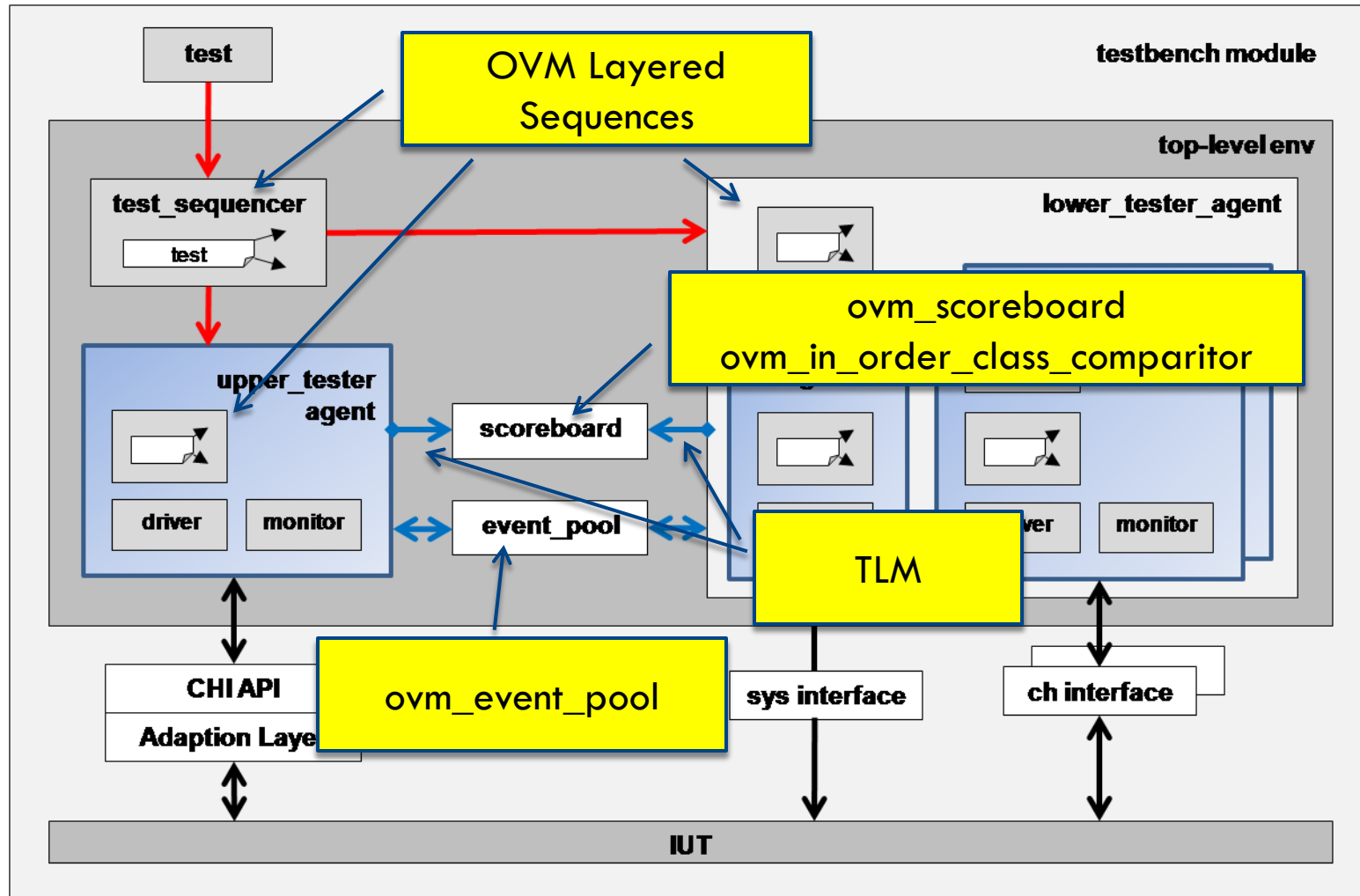
- Simulation-based conformance testing
- **Specification and tests validated early**
- Easy for end users to setup
- Spot and debug issues during development
- Simulator independence via SystemVerilog
- **Industry standard implementation via OVM**

Earlier and lower cost conformance testing

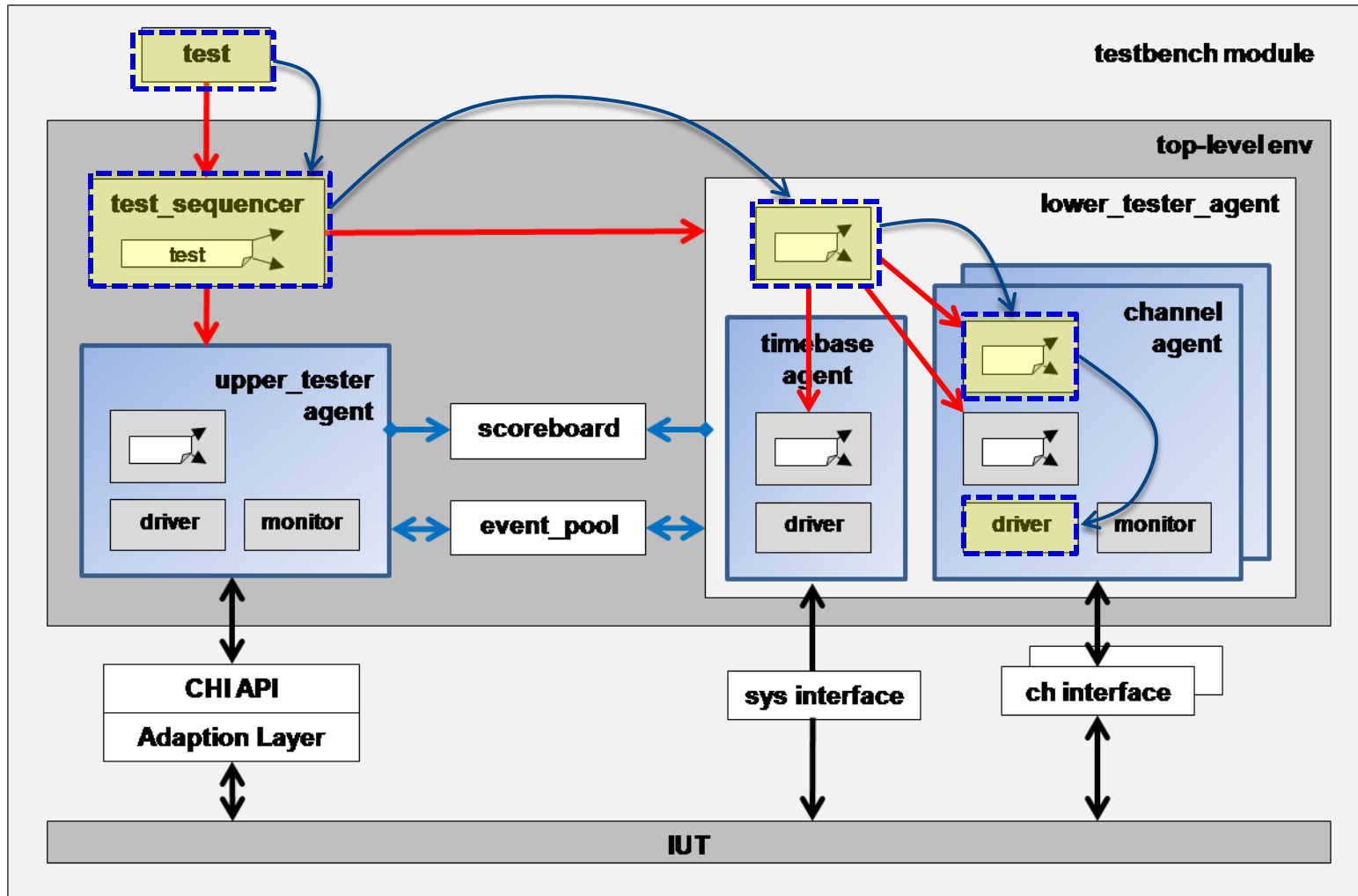
Conformance Test Environment



What OVM Brings to the Party



Toplevel Sequence Layer



Using OVM Sequences for Tests

```
`fr_do_lt_with(lt_er_seq, {  
    lt_ch      == FR_AB;  
    lt_cycle  == 9;  
    lt_kind   == FR_STARTUP_PAYLOAD;  
    lt_slot   == 1;  
    lt_error  == FR_HEADER_CRC;  
})
```

- Example:
 - In cycle 9, the LT simulates a startup frame in slot 1 with wrong header CRC
- Checks fully automatic
- Concise description of test stimulus

Toplevel Sequence Layer Code

```
class execute_seq extends flexray_sequence;

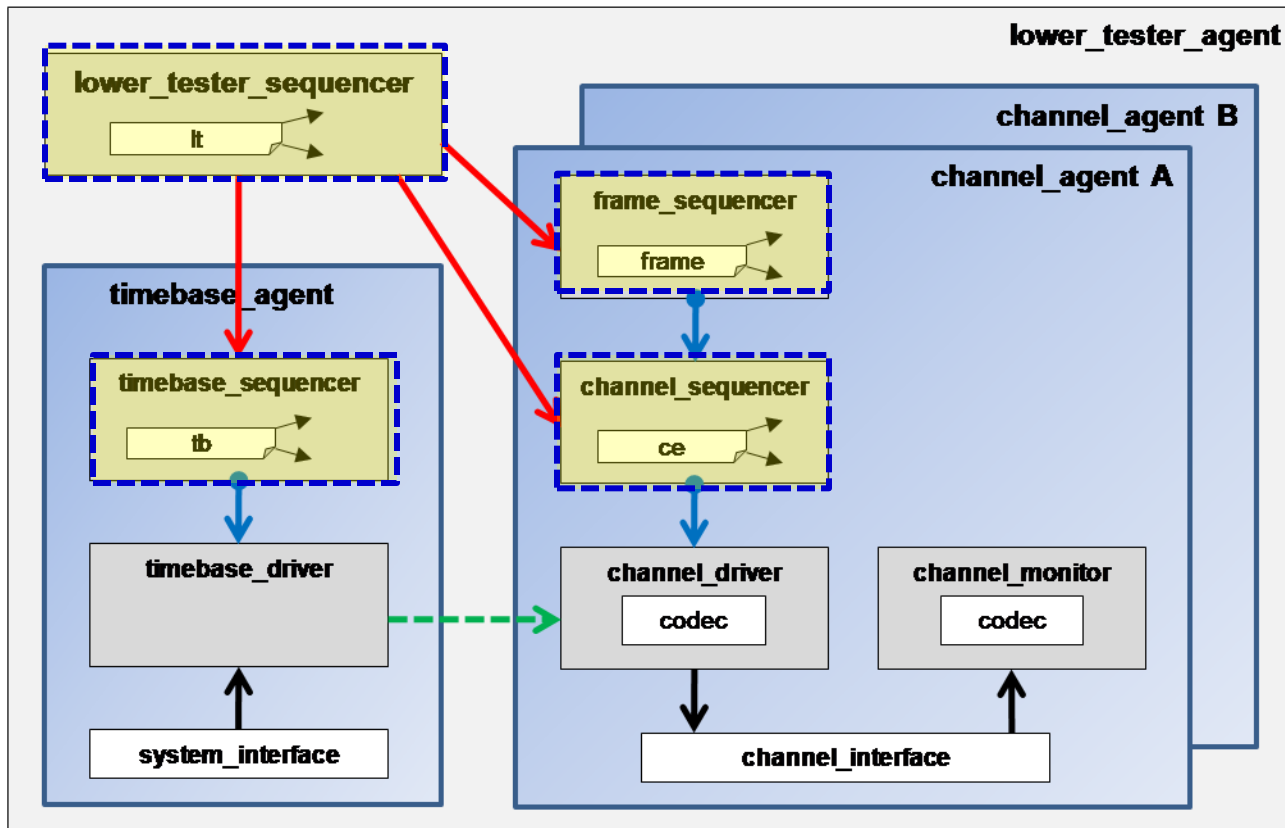
    ut_execute_seq ut_seq;
    lt_execute_seq lt_seq;

    rand flexray_channel_id_enum t_ch;
    rand int t_count;
    ...
    virtual task body();
        ...
        fork
            `ovm_do_on_with(ut_seq, p_sequencer.m_chi_sequencer,
                           {ut_seq.count==t_count;})
            `ovm_do_on_with(lt_seq, p_sequencer.m_lt_sequencer,
                           {lt_seq.lt_count==t_count;lt_seq.lt_ch==t_ch;})
        join
            `fr_report_info("done", OVM_HIGH)
        endtask : body
        ...
    endclass : execute_seq
```

do Upper Tester sequence

do Lower Tester sequence

Next Layer Down: The Lower Tester



Lower Tester Sequence Code

```
class lt_execute_seq extends flexray_sequence;
...
virtual task body();
  fork
    if (fr_inside(lt_ch, '{FR_A, FR_AB}))
      `ovm_do_on_with(ch_seq, p_sequencer.m_ch_a_sequencer,
                    {ch_seq.ch_e_count==lt_count;...})
    if (fr_inside(lt_ch, '{FR B, FR AB}))
      `ovm_do_on_with(ch_seq, p_sequencer.m_ch_b_sequencer,
                    {ch_seq.ch_e_count==lt_count;...})
    `ovm_do_on_with(tb_seq, p_sequencer.m_tb_sequencer,
                  {tb_seq.tb_count==lt_count;})
  join
  ...
endtask : body
...
endclass : lt_execute_seq
```

do Channel A
sequence

do Channel B
sequence

do Timebase
sequence

Bottom Level: Item Sent to Driver

```
class ch_slot_x1_seq extends flexray_sequence;
...
virtual task body();
...
`ovm_create(ce_item)
if (!ce_item.randomize() with {...})
    ovm_report_error("RNDFLD", "Randomization failed for ce_item");
ce_item.m_frame = new[ce_item.m_slot.size()];
foreach (ce_item.m_frame[i]) begin
    ...
    ce_item.m_frame[i] = frame_item;
end
`ovm_send(ce_item)
endtask : body
...
endclass : ch_slot_x1_seq
```

Randomize the item

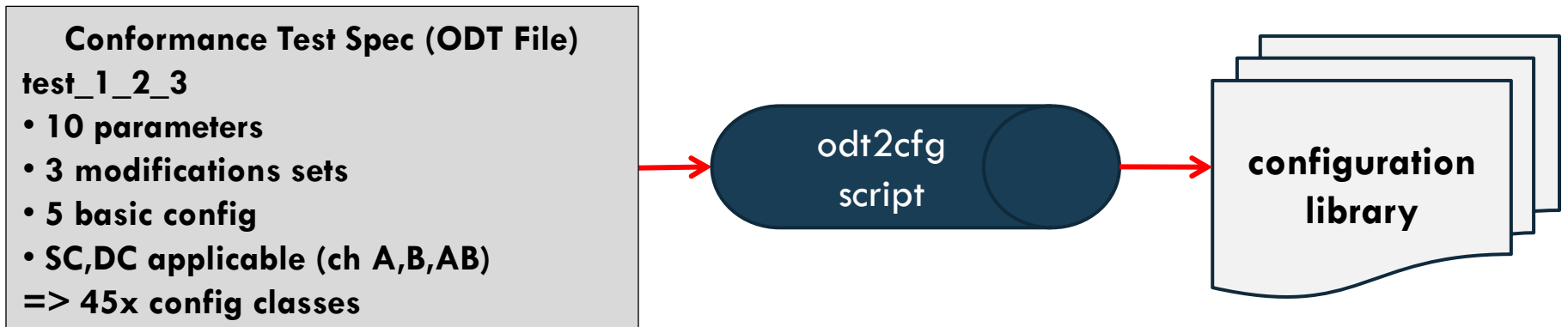
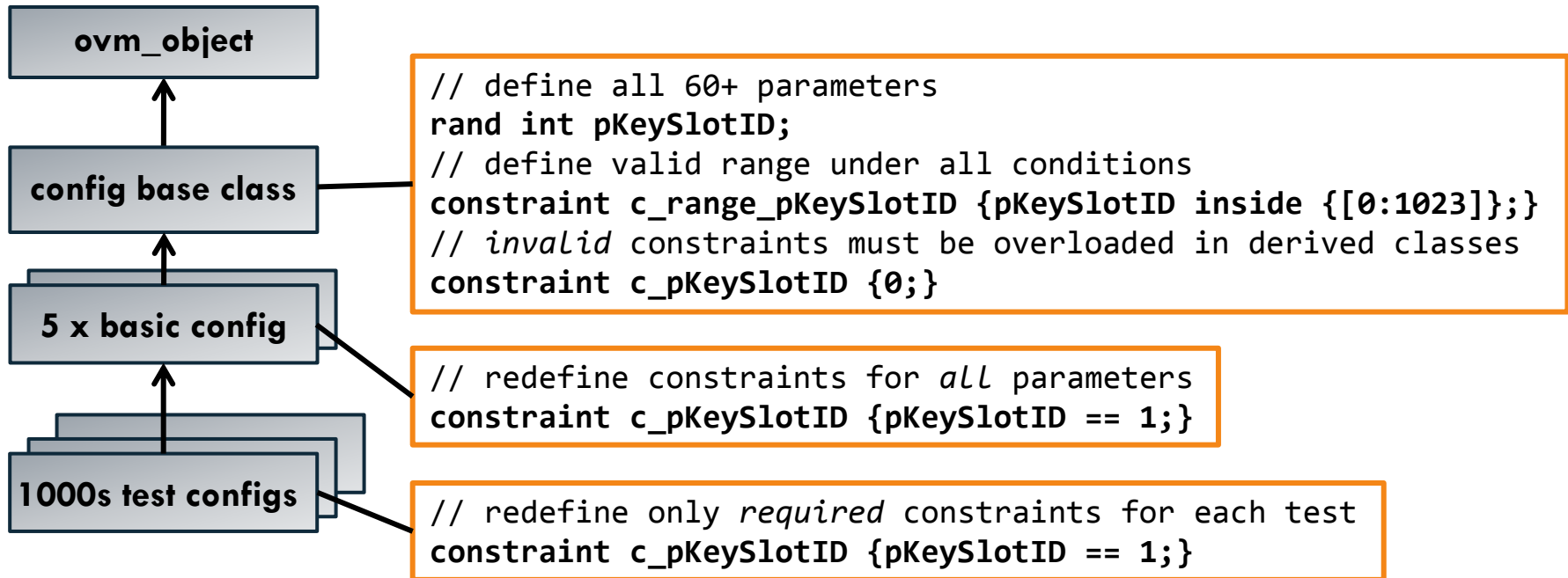
Send item to the driver

Using OVM Factory for Configuration

- FlexRay Conformance Tests
 - Have more than 60 node and cluster related configuration parameters
 - Plus additional test specific modifications
 - Require >10,000 total test runs
- Each test specializes the configuration
- Configuration information used all over the testbench

Enter the OVM Factory ...

Configuration Hierarchy



Test File Structure

```
`include "config_lib/config_1_2_3.sv"

class test_1_2_3_seq extends base_test_seq;
  `ovm_sequence_utils(test_1_2_3_seq, flexray_test_sequencer)
  virtual task body();
    `fr_preamble
    `fr_do (preamble3_seq)
    `fr_execute
    fork
      `fr_do_ut_with (ut_execute_seq, {count==5;})
      `fr_do_lt_with (lt_error_seq, {ch==AB; ... error==CRC;})
    join
      // do sequence with constraints on LT sequencer
      `define fr_do_lt_with(OVM_SEQUENCE_ITEM, CONSTRAINTS) \
        `ovm_do_on_with(OVM_SEQUENCE_ITEM, \
          p_sequencer.m_lt_sequencer, CONSTRAINTS)
    `fr_postamble
    `fr_do (p)
  endtask
endclass

`fr_test_all_scdc_i(test_1_2_3, i)
`fr_test_all_scdc_i(test_1_2_3, ii)
`fr_test_all_scdc_i(test_1_2_3, iii)
```

include classes from configuration library

test body is config independent

macros generate all versions of test with specialized config

Using OVM Factory Override

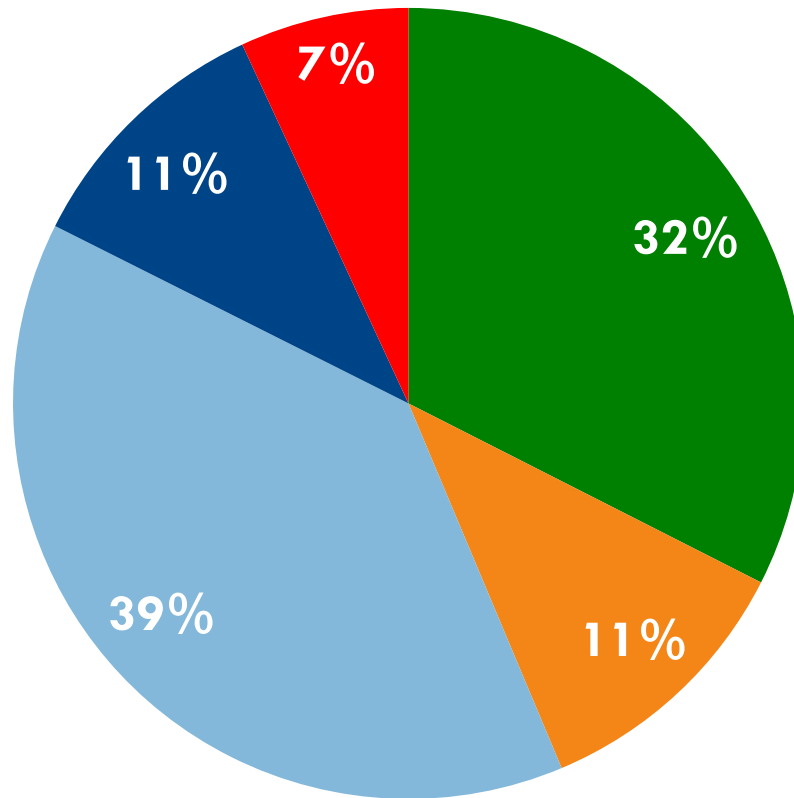
```
`define fr_test_utils(TEST,ROLE) \  
class TEST extends flexray_base_test; \  
... \  
`ovm_component_utils(TEST) \  
  
function void build(); \  
super.build(); \  
set_type_override_by_type(  
    flexray_config::get_type(),TEST`_config::get_type()) \  
); \  
... \  
set_config_int("*","m_iut_role",ROLE); \  
... \  
endfunction : build \  
endclass
```

Macro used for type specific test

Override with the test specific configuration

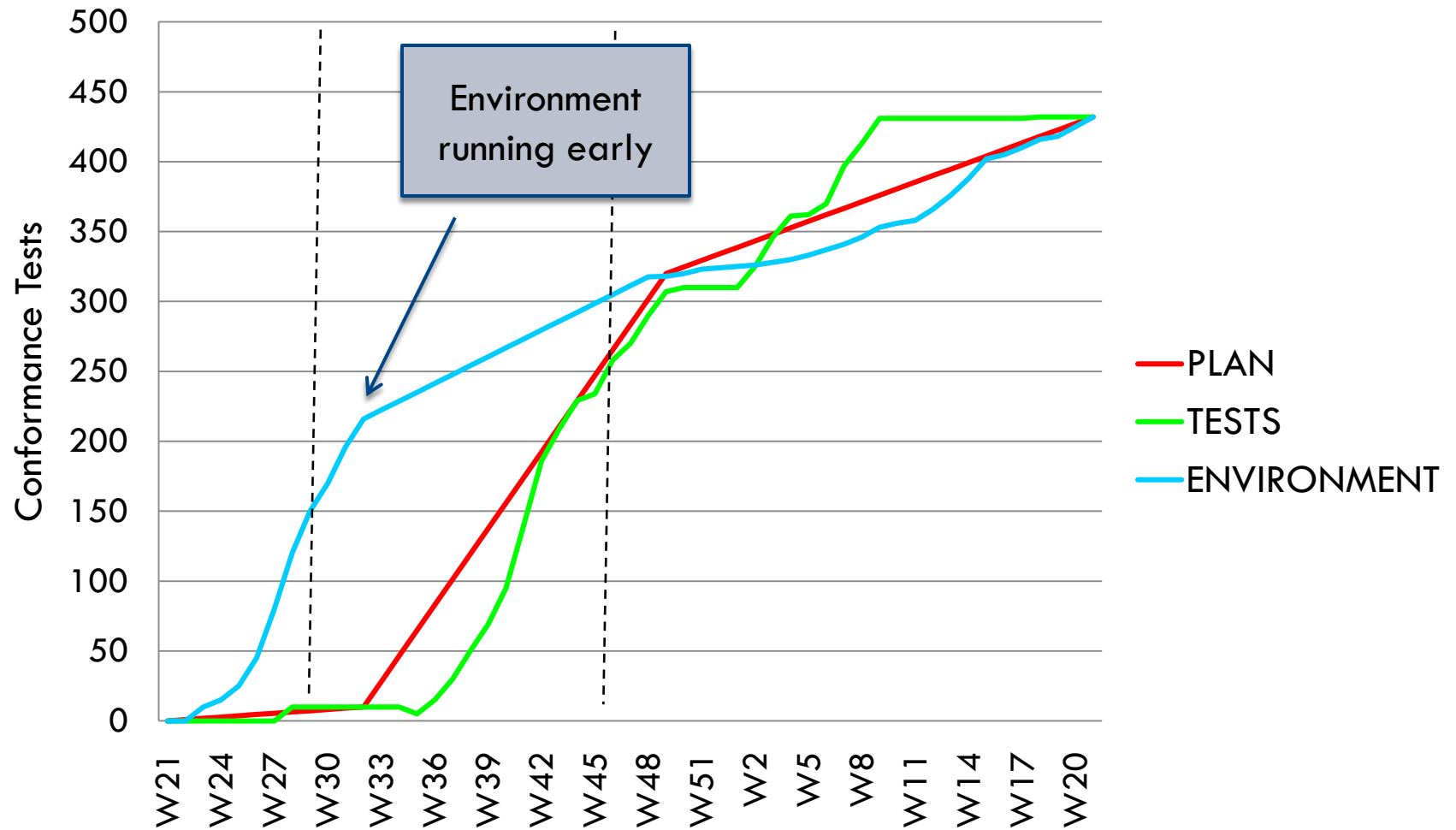
Whole environment sees the change

Code Distribution



- Tests
- Sequence Library
- Config Library
- ENV Components
- OVM Infrastructure

Project Development Chart



Conclusions

- FlexRay Consortium goals were met
 - FlexRay 3.0 Conformance Test Specification rigorously pipe-cleaned
 - FlexRay Executable Model rigorously debugged
 - More cost effective Conformance Test Environment
 - Conformance testing earlier in development cycle
 - Multi-simulator support
- OVM is truly interoperable
- OVM increased development productivity and consistency