Simulation-Based FlexRay™ Conformance Testing using OVM

Mark Litterick
Senior Consultant and Co-Founder, Verilab
Agenda

- FlexRay overview
- What we mean by conformance testing
- What OVM brings to the party
- Layered sequences
- Using the factory for configuration
- Using TLM ports for scoreboard checker
- Some project development statistics
- Conclusions
FlexRay™ Overview

- automotive communications system
- robust, deterministic & fault-tolerant
- supports applications like drive-by-wire

CC = Communication Controller
FlexRay™ Configurations

- scalable & flexible
- many network topologies
- different data rates
- synchronous and asynchronous transfer
- single or multi-master clock synchronization
- time synchronization across networks
- scalable fault tolerance
- over 60 node and cluster parameters
Conformance Testing

- FlexRay Conformance Test Specification
  - defines a set of directed test cases and methods
  - verifies conformance of communication controllers with FlexRay Protocol Specification
  - devices must pass these tests to claim conformance
  - test specification is independent of test environment implementation
  - test environment implementation can be hardware or simulation-based

Conformance Testing is a subset of Verification
Simulation-Based Testing

- Simulation-based verification is standard for all complex semiconductor design
- Advantages include:
  - validate behaviour before manufacture
  - find and correct defects earlier in design cycle
  - improved controllability and observability
  - lower-cost and lower-risk for supplier and end user
  - improved quality
- Additional benefits to Conformance Test project include:
  - concluded prior to any V3.0 controllers being available
  - enabled validation of evolving Conformance Test Specification
  - also found several defects in new Protocol Specification features
  - verification of FlexRay Executable Model
SystemVerilog & OVM

- Multi-language simulators
  - VHDL, Verilog, SystemVerilog, SystemC
- FlexRay-specific code
- Uses OVM building blocks
- Open source (Apache)
- Class library
- Consistent methodology
- Facilitates interoperability
- Supported by all simulators
Conformance Test Environment

Upper Tester (UT)

Lower Tester (LT)

Conformance Tests

CHI Adaption Layer

Implementation Under Test (IUT)

LOG

RESULTS PASS/FAIL

IUT Implementation
- Capabilities
- Features

PASS

EDA Tech Forum
Technical Resource for the EE Community

Verilab
OVM Architecture

- OVM Layered Sequences
- ovm_component Building Blocks
- TLM communication

- ovm_scoreboard
- ovm_comparator

- ovm_event_pool

- OVM Factory
OVM Sequences

- OVM sequences are used for all stimulus
- Concise description of test stimulus
- Example:

  In cycle 9, the LT simulates a startup frame in slot 1 with the wrong header CRC...

```c
`ovm_do_on_with(error_seq,`LT,{
  lt_ch    == FR_AB;
  lt_cycle == 9;
  lt_kind  == FR_STARTUP_PAYLOAD;
  lt_slot  == 1;
  lt_error == FR_HEADER_CRC;
})
```
Sequence Hierarchy

[Diagram showing sequence hierarchy with nodes and arrows representing dependencies and connections between components such as test, test_sequencer, upper_tester_agent, driver, monitor, scoreboard, event_pool, time_base_agent, driver, monitor, channel_agent, sys_interface, ch_interface, and IUT.]
Layered Sequences

`ovm_do(preamble3_seq)

fork
  `ovm_do_on_with (ut_exe_seq, `UT, {count==1;})
  `ovm_do_on_with (lt_error_seq, `LT, {ch==AB; kind==STARTUP; cycle==8; error==CRC;})
join

fork
  if (ch inside{A, AB}) `ovm_do_on_with (ch_x_seq, `CHA, {...; error==CRC;})
  if (ch inside{B, AB}) `ovm_do_on_with (ch_x_seq, `CHB, {...; error==CRC;})
  `ovm_do_on_with (tb_cycle_seq, `TB, {count==1;})
join

`ovm_create(item)
  item.c_error.constraint_mode(0); // disable
  item.randomize() with {...; item.error==CRC;}
  `ovm_send(item)

if (item.error==CRC) generate_bad_crc();
else generate_good_crc();
ch_interface.RxD = crc[N]; // drive signals
Using OVM Factory for Configuration

- FlexRay protocol has more than 60 node and cluster configuration parameters
- Each test must run under multiple configurations
  - 5x basic configurations
  - 3x channel applicability
  - preamble & test-specific modifications
  - require 10164 total runs for 432 conformance tests
- Configuration information used all over the testbench (drivers, monitors, sequencers)

OVM Factory solves the problem...
// define all 60+ parameters
rand int pKeySlotID;
// define valid range under all conditions
constraint c_range_pKeySlotID {pKeySlotID inside {[0:1023]};}
// invalid constraints must be overloaded in derived classes
constraint c_pKeySlotID {0;}

// redefine constraints for all parameters
constraint c_pKeySlotID {pKeySlotID == 1;}

// redefine only required constraints for each test
constraint c_pKeySlotID {pKeySlotID == 2;}

Conformance Test Specification
(ODT File)
test_1_2_3
• 10 parameters
• 3 modifications sets
• 5 basic config
• 3 ch applicable (A,B,AB)
=> 45x config classes

EDA Tech Forum
Technical Resource for the EE Community

odt2cfg script
configuration library
// test file
// single line to include configuration
`include "config_lib/config_1_2_3.sv"

// configuration library file
// 45 config classes similar to this one (for this test)
class test_1_2_3_bc1a_cha_i_config extends flexray_config_bc1a_p2;
`ovm_object_utils(test_1_2_3_bc1a_cha_i_config)
`fr_config_mod_cha
constraint c_gdDynamicSlotIdlePhase { gdDynamicSlotIdlePhase == 0 ; }
constraint c_gMacroPerCycle   { gMacroPerCycle   == 2095 ; }
constraint c_gNumberOfStaticSlots { gNumberOfStaticSlots == 2 ; }
constraint c_gPayloadLengthStatic { gPayloadLengthStatic == 0 ; }
constraint c_gdCycle         { gdCycle         == 2095 ; }
constraint c_pClusterDriftDamping { pClusterDriftDamping == 0 ; }
constraint c_pdListenTimeout { pdListenTimeout == 167702; }
constraint c_pMicroPerCycle  { pMicroPerCycle  == 83800 ; }
constraint c_pOffsetCorrectionStart { pOffsetCorrectionStart == 2085 ; }
constraint c_pRateCorrectionOut { pRateCorrectionOut == 51 ; }
constraint c_pPayloadLengthDynMax { pPayloadLengthDynMax == 1 ; }
endclass
test file simple and uncluttered

macros verbose but hidden

override all config with test-specific configuration

Whole environment sees the change
`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, test_sequencer)
  virtual task body();
  // preamble
  `ovm_do (preamble3_seq)
  // execute
  fork
    `ovm_do_on_with (exe_seq,`UT, {count==1;})
    `ovm_do_on_with (err_seq,`LT, {ch==AB; kind==...})
  join
  // postamble
  `ovm_do (postamble_seq)
endtask

class
  `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)

run test_1_2_3_bc1a_cha_ii

run_conformance list.all.txt

include classes from configuration library

1 test file

N generated configuration classes

N generated specialized tests

test body is config independent

macros generate all versions of test
## Scoreboard Checker

### UT Sequence

<table>
<thead>
<tr>
<th>Cycle</th>
<th>No schedule</th>
<th>Cycle 0</th>
<th>Cycle 1</th>
<th>Cycle 2</th>
<th>Cycle 3</th>
<th>Cycle 4</th>
<th>Cycle 5</th>
<th>Cycle 6</th>
<th>Cycle 7</th>
<th>Cycle 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### CHI Commands

- PREAMBLE
- EXECUTE
- POSTAMBLE

### IUT

- DC
- CR
- IL
- IS
- ICC
- NA

### Channel

- CAS

### Cluster Traffic

- CAS

### CHI Status

- STARTUP

### Sequence

- EXECUTE

### Commands

- CHI

### Expect + Observe = Compare
SB Operation & TLM Ports

- UT & LT sources *publish* transactions
  - instantiate OVM TLM analysis *port*
  - write expected or observed transactions to port
    - IUT transmit: UT expects buffer to be transmitted, LT observes bus traffic
    - IUT receive: LT expects traffic to be received, UT observes buffer status

- Scoreboard *subscribes* to transactions
  - instantiate OVM TLM analysis *export*
  - implement export-specific write method
  - send transactions to comparator

- Transaction comparator
  - compares pairs of transactions
  - comparison rules implemented in transaction
Scoreboard Implementation

```verilog
ovm_analysis_port #(flexray_bus_transaction) m_chi_bus_port;
transaction = generate_transaction();  // expect
m_chi_bus_port.write(transaction);  // publish

ovm_analysis_port #(flexray_bus_transaction) m_cha_bus_port;
transaction.field = ch_interface.TxD;  // observe
m_cha_bus_port.write(transaction);  // publish

ovm_analysis_imp_chi_bus #(flexray_bus_transaction, fr_scoreboard) m_chi_bus_export;

ovm_analysis_imp_cha_bus #(flexray_bus_transaction, fr_scoreboard) m_cha_bus_export;
m_cha_iut_tx_comparator.before_export.write(transaction);  // in write_chi_bus()
m_cha_iut_tx_comparator.after_export.write(transaction);  // in write_cha_bus()

m_upper_tester.m_chi_bus_port.connect(m_scoreboard.m_chi_bus_export);
m_lower_tester.m_cha_bus_port.connect(m_scoreboard.m_cha_bus_export);

function bit flexray_bus_transaction::comp(flexray_bus_transaction transaction);
    if (compare_failed) `fr_report_error(...)
```

UT CHI ovm_monitor expected transaction

LT CH ovm_monitor observed transaction

ovm_scoreboard subscribes TLM export & implements write()

write to ovm_in_order_class_comparator

TLM port and export connected in env

actual comparison defined in transaction
Code Distribution

- Tests: 32%
- Sequence Library: 11%
- Config Library: 11%
- ENV Components: 7%
- OVM Infrastructure: 11%

EDA Tech Forum
Technical Resource for the EE Community
Project Development Chart

OVM environment running early

Forget OVM is even there and focus on application problems

OVM facilitated parallel test development
Conclusions

- FlexRay Consortium project goals were met
  - Improved quality of *FlexRay V3.0 Conformance Test Specification*
  - FlexRay *Executable Model* rigorously debugged
  - More cost effective Conformance Test Environment
  - Conformance testing earlier in development cycle
  - Operation validated with multiple target IUTs
- OVM increased development productivity and consistency
  - On-time delivery for all project milestones
- OVM really *facilitates* interoperation
  - interoperation of non-OVM SystemVerilog code still a lot of effort!
  - achieved identical conformance results on different simulators
Questions?

mark.litterick@verilab.com