FlexRay™ Conformance Testing using OVM

Mark Litterick
Co-founder & Verification Consultant
Verilab
Introduction

• FlexRay overview
• What is conformance testing
• Open Verification Methodology (OVM)
  – architecture and solutions
  – mapping of tests to layered sequences
  – using the factory for configuration
• Some project development statistics
• Conclusion
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 V3.0 Conformance Test Specification
  – set of directed test cases and methods
  – verifies conformance of communication controllers with FlexRay Protocol Specification
  – devices must pass these tests to claim conformance
  – specification is independent of implementation

• hardware-based tests on real devices, or
• simulation-based tests pre-silicon

Conformance Testing is a subset of Verification
Requirements

• Support **multiple tools**
  – identical test results

• Test **different IUTs**
  – different languages
  – different suppliers
  – future devices

• Flexible & extensible
  – evolving test specification

• Maintainable
  – manage configurations
  – manage test volume
  – clear test mapping to spec

![Diagram showing Conformance Test Environment with IUT-1 (SystemC), IUT-2 (verilog), and IUT-3 (VHDL) connected to Cadence IUS™ and Mentor Questa™.](image-url)
SystemVerilog & OVM

- Conformance Test Environment
- Open Verification Methodology (OVM)
- SystemVerilog IEEE1800
- Questa™
- IUS™
- VCS™

- FlexRay-specific code
- Uses OVM building blocks
- Open source (Apache)
- Class library
- Consistent methodology
- Facilitates interoperability
- Supported by all simulators

- Multi-language simulators
- VHDL, Verilog, SystemVerilog, SystemC
Conformance Test Environment

Conformance Tests

Conformance Test Environment

CONFORMANCE TEST ENVIRONMENT

RESULTS PASS/FAIL

Implementation Under Test (IUT)

CHI Adaption Layer

LOG

Conformance Tests

IUT Implementation
- Capabilities
- Features

PASS

RESULTS PASS/FAIL

LOWER TESTER (LT)

UPPER TESTER (UT)

TESTER

TESTER

TESTER

TESTER

TESTER

TESTER

TESTER

TESTER

TESTER
OVM Architecture

- OVM Layered Sequences
- ovm_component Building Blocks
- OVM Factory

- OVM Event Pool
- OVM Comparator
- TLM Communication

Diagram showing various components and their interactions:
- Test
- Test Sequencer
- Upper Tester Agent
- Lower Tester Agent
- Channel Agent
- Timebase Agent
- Scoreboard
- Event Pool
- CAPI
- Adaptation Layer
- IUT
Test Example

- Upper Tester stimulus – host software interface
- Lower Tester stimulus – emulates cluster traffic
- Status, buffer, frame content and timing checks
- Configuration setup for test repetition
N generated configuration classes

1 test file

N generated specialized tests

run **test_1_2_3_bc1a_cha_ii**

run_conformance **list.all.txt**

include "config_lib/config_1_2_3.sv"

class test_1_2_3_seq extends base_sequence

\`ovm_sequence_utils(test_1_2_3_seq, test_seq)

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_scsc_i(test_1_2_3, i)
\`fr_test_all_scsc_i(test_1_2_3, ii)
\`fr_test_all_scsc_i(test_1_2_3, iii)

include classes from configuration library

test body is config independent

macros generate all versions of test
The IUT is configured to receive frames in slot 4 and to transmit additional frames in dynamic slots 3 and 5.

### Preamble (setup state)

Preamble II.

### Test Execution

1. In cycle 7, the LT simulates an additional frame in dynamic slot 4. It is verified (LT) that the IUT transmits its frames in slots 1, 3 and slot 5 correctly, i.e. the frame ID, the cycle count, the payload length, the indicator and the payload preamble.

   - automatic scoreboard check – all IUT Tx frames are checked on LT

2. In the NIT of cycle 7, it is verified that the frame as simulated by the LT

### Postamble

The UT sets the IUT into POC:halt state.

### Pass Criteria

Pass: The IUT transmits its frames correctly and receives the LT’s frame in slot 4 correctly.

- automatic scoreboard check – all Rx buffers are checked on UT

```
`ovm_do(preamble2_seq) // C0-C6
```

```
`ovm_do_on_with(dyn_slot_seq, `LT, { cycle == 7; sstat == 2; kstat == STARTUP_NULL; sdyn == 4; kdyn == PAYLOAD; })
```

```
// automatic scoreboard check – all IUT Tx frames are checked on LT

dt_3_5 = 2*(`ceil((`gdTSSTransmitter+`cdFSS+50+1*20+30+`cdFES+1)*`gdBit)/ (`gdMinislot*`gdMacrotick)) + `gdDynamicSlotIdlePhase+1
`ovm_do_on_with(time_check_seq, `LT, { cycle == 7; start == 3; end == 5; interval == dt_3_5; tolerance ==`Xi; })
```

```
// automatic scoreboard check – all Rx buffers are checked on UT

`ovm_do(postamble_freeze_seq)
```
Sequence Hierarchy

```
#ovm_do(preamble3_seq)

fork
  #ovm_do_on_with (ut_seq, `UT, {count==1;})
  #ovm_do_on_with (lt_seq, `LT, {...; error==CRC;})
join

fork
  #ovm_do_on_with (ch_seq, `CHA, {...; error==CRC;})
  #ovm_do_on_with (tb_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
```
Configuration with OVM Factory

• Each test must run under multiple configurations
  – 5x basic configurations
  – 3x channel applicability
  – test-specific modifications
  – 60+ configuration parameters
  – total **10164 runs** for **432 conformance tests**

• Three configuration challenges:
  – Generation of configurations
  – Replication of tests for different configurations
  – Propagation of configuration to all components

auto-script
SV macros
OVM factory
Configuration Hierarchy

// 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;}
Example Configuration

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Modification to Basic Configuration 1a</th>
</tr>
</thead>
<tbody>
<tr>
<td>gdDynamicSlotIdlePhase [Minislot]</td>
<td>I II III</td>
</tr>
<tr>
<td>gMacroPerCycle [μT]</td>
<td>0 2095 2</td>
</tr>
<tr>
<td>gNumberOfStaticSlots</td>
<td></td>
</tr>
<tr>
<td>gPayloadLengthStatic [two-byte word]</td>
<td></td>
</tr>
<tr>
<td>gdCycle [μs]</td>
<td></td>
</tr>
<tr>
<td>pClusterDriftDamping [μT]</td>
<td></td>
</tr>
<tr>
<td>pdListenTimeout [μT]</td>
<td></td>
</tr>
<tr>
<td>pMicroPerCycle [μT]</td>
<td></td>
</tr>
<tr>
<td>pOffsetCorrectionStart [μT]</td>
<td></td>
</tr>
<tr>
<td>pRateCorrectionOut [μT]</td>
<td></td>
</tr>
<tr>
<td>pPayloadLengthDynMax [two-byte word]</td>
<td></td>
</tr>
</tbody>
</table>

// 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_bc1aCha_i_config extends flexray_config_bc1a_p2;
`ovm_object_utils(test_1_2_3_bc1aCha_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
Configuration Macros & Factory

```
`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)
```

**test file**
simple and uncluttered

**macros**
verbose but hidden

**factory**
override config with test-specific class

```
define fr_test_all_scdc_i(TEST,INDEX) 
`fr_test_config_ch_i_utils(TEST,bc1a,cha, INDEX) 
`fr_test_config_ch_i_utils(TEST,bc1a,chb, INDEX) 
... (15x fr_test_config_ch_i_utils in total)
`fr_test_config_ch_i_utils(TEST,bc3,chab,INDEX)
```

```
define fr_test_config_ch_i_utils(TEST,CONFIG,CH,INDEX) 
class TEST`_`CONFIG`_`CH`_`INDEX extends flexray_base_test; 
`ovm_component_utils(TEST`_`CONFIG`_`CH`_`INDEX)
function void build(); 
super.build();
set_type_override_by_type(  
  flexray_config::get_type(), TEST`_`CONFIG`_`CH`_`INDEX`_`config::get_type();
);
endfunction : build
endclass
```

Whole environment sees the change
Code Distribution

- Small OVM overhead
- Very Complex Application-Specific Modelling and Checking
- Mainly Automatically Generated Code
- Test-Specific but Easy
- Medium Complexity lots of reuse Sequences

- Tests
- Sequence Library
- Config Library
- ENV Components
- OVM Infrastructure
Effort & Interoperation

- Simulator interoperation
- OVM = effortless
- SystemVerilog = effort
  - each simulation different:
    - IEEE spec interpretation
    - non-supported syntax
    - complex constraint resolution
- Code compromises
  - all resolved with code rewrite
  - no simulator `ifdefs used
  - identical behaviour achieved
Project Development Chart

- OVM environment running early
- Forget OVM is even there and focus on application problems
- OVM facilitated parallel test development
Conclusion

• FlexRay Consortium project goals were met
  – Improved quality of 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 & 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