Generation of UVM compliant Test Benches for Automotive Systems using IP-XACT with UVM-SystemC and SystemC AMS

Ronan LUCAS (Magillem)
Philippe CUENOT (Continental)
Agenda

• Introduction

• IP-XACT extension for UVM

• Test bench generation

• Traceability

• Conclusion
Introduction : Automotive Design

Vehicle System Sub-System Electronic Unit

with impact on ... with impact on ... with impact on ...

Electronic Development

ASIC µP SW
Introduction: Virtual Prototype

- Make ECU model available before HW delivery
  - HW-SW Co-design and function exploration
- System simulation for V&V cycle
  - Digital simulation is state of the art (System-C)
  - Mixed Signal ASICs ⇒ SystemC-AMS as Executable Spec.
  - Golden reference ⇒ Investigate UVM (VERDI UVM-SC-AMS)

Focus on ASIC

© Accellera Systems Initiative
Introduction : System Under Test

- ASIC for Engine Management System

Smart Power Supply as Design Under Test example

with Communication interface

© Accellera Systems Initiative 2014
Introduction: Test Bench Organization

Agent = ASIC feature  -  Test = Test Case
IP-XACT for Specification driven Verification Flow

➢ From the specification to the UVM code, based on IEEE1685 IP-XACT standard

➢ to provide a unified specification for a verification component

➢ to exchange and share compatible components from multiple companies or services.

➢ To enable efficient assembly and configuration of test bench, test and top level elements by generating the relevant SystemC and SystemC-AMS views
IP-XACT extensions for UVM

• Assembly
  – Hierarchy and interconnections inside testbench follow the existing IEEE1685 standard
  – Extension to identify: UVM test, virtual interface
  – Connection of DUT to the testbench
IP-XACT extensions for UVM

- Virtual Interface as an IP-XACT component:
  - Bus Interface definition: Logical representation of interconnection with the UVC through the UVM configuration mechanism
  - Ad hoc connections to represent the connections to the DUT through signals
IP-XACT extensions for UVM

• Configuration:
  – Extension to store information in the central resource: UVM configuration database

```cpp
// record the configuration
uvm::uvm_config_db<testbench_config*>::set(this,"m_tb0.*","my_tb_config",m_tb_config);
```
IP-XACT extensions for UVM

- Agent of configuration:
  - `uvm_object` defined by a name and instance name
  - To configure each UVC in testbench through dedicated method defined by an unbounded list of parameters
  - Contained: nested configuration description, virtual interface
  - Configuration parameter is defined by a name, a value and a type

```cpp
virtual void configure_spi_agent_config(
    spi_agent_config* config,
    uvm::uvm_active_passive_enum active,
    std::string in_file_name,
    spi_trans_config* m_spi_trans_cfg);
```

```cpp
spi_trans_config* m_spi_trans_cfg;
spi_agent_config* m_spi_agent_cfg;
```

```cpp
m_spi_agent_cfg = spi_agent_config::type_id::create("m_spi_agent_cfg", this);
```

```cpp
configure_spi_agent_config(m_spi_agent_cfg, UVM_ACTIVE,
    m_tb_config->spi_file_concat, m_spi_trans_cfg);
```
Testbench generator

- Generator architecture
  - TGI API to get meta-datas
  - Template based to customize the outputs
  - Generated files:
    - sc-main
    - test.h
    - test.cpp
    - testbench.h
    - testbench.cpp
    - testbench_cfg.h
    - virtual_sequence.h
Template mechanism

- **Text** to customize the output: header, fix library, comments
- **key words** replaced by generator from an elaborated IP-XACT description

```
#include <systemc>
#include <systemc-ams>
#include <uvm.h>

#include "ena/v_regulator_if_ena.h"
#include "vin/v_regulator_if_vin.h"
#include "spi/v_regulator_if_spi.h"
#include "vout/v_regulator_if_vout.h"
#include "test.h"
#include "reading_method.h"
```

// Connect the Virtual Interfaces to the dut
@dut->Enable_V(vif_v_regulator_ena->sig_Enable_V);
dut->Trans_spi_cs_port(vif_v_regulator_spi->sig_Tspi_CS_V);
dut->Trans_spi_sdi_port(vif_v_regulator_spi->sig_Tspi_SDI_V);
....
Traceability of test requirements

- Fragments the Test specification in Minimum Reusable Unit during the import
- Builds IP-XACT test-bench
- Creates links between fragments and ip-xact elements of test-bench
- Netlists UVM test environment
- Traces the test requirements
- Analyzes the impact of change request
- Generates reports
Treacability of test requirements

- Ensure the verification of all requirements defined in component specification
- **Use case 1**: Evaluate the coverage of the requirements
- **Use case 2**: Evaluate the impact of a change request on tests or DUT and identify non-regression test suite
Use Case 1: Coverage report

- Requirements and tests are linked
- Coverage report shows:
  - the requirements covered by the verification flow
  - the last test execution

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Test</th>
<th>Stimuli</th>
<th>Test Equip</th>
<th>trace</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQT_22</td>
<td>Test1, Test2</td>
<td>Test1/ana_stim1, Test1/spi_stim1, Test2/spi_stim2</td>
<td>E33220 NI-8452</td>
<td>Test1/trace1</td>
<td>05/10</td>
</tr>
<tr>
<td>RQT_23</td>
<td>Test1</td>
<td>Test1/ana_stim1, Test1/spi_stim1</td>
<td>E3664 NI-8452</td>
<td>???</td>
<td></td>
</tr>
<tr>
<td>RQT_24</td>
<td>test3</td>
<td>Test3/ana_stim1, Test3/spi_stim3</td>
<td>E33220 NI-8452</td>
<td>Test3/trace2</td>
<td>05/31</td>
</tr>
<tr>
<td>RQT_25</td>
<td>???</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Use Case 2: Impact Report

- **A change request** in the specification can impact test suite, DUT, UVCs ...
- **Impact report** shows the list of test involved by a change request or the consolidation of a change request

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Test</th>
<th>Test Effort</th>
<th>Test Equip.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RQT_23</td>
<td>test1</td>
<td>2h</td>
<td>E33220 NI-8452</td>
</tr>
<tr>
<td>RQT_24</td>
<td>test3</td>
<td>0.2h</td>
<td>E3664 NI-8452</td>
</tr>
</tbody>
</table>
Conclusion: Verification

• The IP-XACT description offers:
  – A **central repository** to exchange test cases between UVM expert and Verification team
  – **Simplify test case execution** for verification team (user interface used only for configuration)
  – An easy integration for **traceability flow**

• The UVM infrastructure offers:
  – **Organization of test** versus IP features (reuse still to be investigated for interface configuration)
  – **Joining with the validation** phase (today only stimuli)
Thank you for your attention

Questions ?