Practical Scheme to Enhance Verification Turn-Around-Time by Using Reusable Harness Interface (RHI)

Jong pil Jung, Hyunju Lee, Jaejin Ha, and Yonghee Im
Agenda

• Introduction
  – Verification Turn-Around-Time(TAT) Issue

• Proposed Scheme
  – Merge DPI-C Sequences into UVM Environment - Step1
  – Reusable Harness Interface(RHI) - Step2

• Experiments

• Conclusions
Introduction: Verification TAT Issue

• Resource Trends

2016 Where ASIC/IC Verification Engineers Spend Their Time

- Test Planning: 39%
- Testbench Development: 22%
- Creating Test and Running Simulation: 22%
- Debug: 14%
- Other: 7%

Design Verification Turn-Around-Time (TAT)
- Testbench build-up TAT
- Scenario development TAT
- Debugging TAT

Introduction : Verification TAT Issue (Cont.)

• Verification Language and Methodology Trends

One of drawbacks
  • Irritating re-compiling TAT while developing scenarios

Proposed Scheme:

Step 1 - Merge DPI-C Sequences into UVM Environment
Merge DPI-C Sequences into UVM Environment

• Testbench with DPI-C sequence and UVM environment combined.

Conventional Harness:
To bridge DUT and UVM

Proposed Harness:
A new role is added – linking DPI-C and UVM

Benefits:
- Faster builds
- Dynamic linking
Merge DPI-C Sequences into UVM Environment (Cont.)

• How to combine DPI-C sequences and UVM environment

<table>
<thead>
<tr>
<th>C domain</th>
<th>Verilog domain</th>
<th>UVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test FW</td>
<td>Harness Interface</td>
<td>Test</td>
</tr>
<tr>
<td>Tester</td>
<td>Get handle</td>
<td>Build Phase</td>
</tr>
<tr>
<td>Driver</td>
<td>Call imported C-function</td>
<td>Run Phase</td>
</tr>
<tr>
<td>DPI-C</td>
<td>Convert UVM sequence</td>
<td>Sequence</td>
</tr>
</tbody>
</table>

- **C domain**
  - Test FW
  - Tester: `test_main()`, `test_scenarios`
  - Driver: `HAL`, `LLD`
  - DPI-C: `ahb_read()`, `ahb_read()`, `axi_read()`, `axi_read()`...

- **Verilog domain**
  - Harness Interface
  - Get handle: `function connect(uvm_env env)
  - Call imported C-function: `task run_dpic();
  - Convert UVM sequence: `task v_ahb_write(input int addr, input int data);

- **UVM**
  - Test Build Phase
  - `function build_phase();
  - Run Phase
  - `task run_phase();
  - Sequence
  - `ahb_sequence axi_sequence user_defined_sequence...`
Proposed Scheme:

Step2 - Reusable Harness Interface (RHI)
Reusable Harness Interface (RHI)

- How to apply RHI to each different hierarchical testbench

C based sequence

Reusable Harness Interface (RHI)

Block level
Sub-system level
Top level
Reusable Harness Interface (RHI) (Cont.)

- Reusable Harness Interface (RHI) is an interface format in which aforementioned harness interface can be systematically reused in a difference level.

- RHI consists of three major parts:
  - Harness interface
  - Harness binding
  - Path define

REUSABLE : Harness Interface + Harness Binding
Only path define file is required
Reusable Harness Interface (RHI) (Cont.)

- **Harness interface**
  - Connecting DUT and UVM environment

```verilog
/** HDL defines */
`include hdl_defines.h

/** Import components */
import ahb_env_pkg::*;
import ahb_seq_lib_pkg::*;

/** Interface declaration */
interface ahb_harness( parameter string ENV_PATH_PARAM = "uvm_test_top.xxx.xxx")();

/** Next slide */
endinterface : ahb_harness
```
Reusable Harness Interface (RHI) (Cont.)

- Harness interface (Cont.) – Inside of interface declaration

```c
/** Handle of components */
ahb_env dpic_ahb_env;

/** Declaration of imported and exported functions and tasks */
import "DPI-C" context task main(int arg0, int arg1, ... , int arg3);
export "DPI-C" task v_ahb_swrite;
export "DPI-C" task v_ahb_sread;

/** Description imported and exported functions and tasks */
task v_ahb_swrite(input int size, intput int addr, input int wdata);
    /** Convert to uvm_sequence */
    ...
endtask : v_ahb_swrite
```

...
**Instantiation of virtual interface */
ahb_interface #() u_ahb_interface (  
    .xxx ( `DUT_TOP.xxx ),  
    ...  
);

/** Set VIF for ENV */
initial begin
    uvm_config_db#(ahb_interface)::set(uvm_root::get(), ENV_PATH_PARAM,  
        "u_ahb_vif", u_ahb_interface);
end

/** Get ENV component */
initial begin
    uvm_config_db#(ahb_env)::get(uvm_root::get(), ENV_PATH_PARAM,  
        "dpic_ahb_env", dpic_ahb_env);
end
Harness binding

- Instantiates harness interface and binds it to `TB_TOP

```vhdl
/** import components */
import xxx_env_pkg::*;

/** HDL path defines */
'include "path_defines.sv"

/** Interface to connect with C sequence */
'include "ahb_harness.sv"

/** Binding interface */
bind `TB_TOP ahb_harness#(`ENV_PATH) u_ahb_harness ();
```
Reusable Harness Interface (RHI) (Cont.)

• Path define
  – Hierarchical HDL path defines such as block, sub-system and top level

```vhdl
/** Define design hierarchy path for block level */
`define `TB_TOP block_tb_top
`define `DUT_TOP `TB_TOP.u_block

/** Define ENV path for block level */
`define `ENV_PATH "uvm_test_top.m_block_env"
```

```vhdl
/** Define design hierarchy path for subsystem level */
`define `TB_TOPsubsystem_tb_top
`define `DUT_TOP `TB_TOP.u_subsystem.u_block

/** Define ENV path for subsystem level */
`define `ENV_PATH "uvm_test_top.m_subsystem_env.m_block_env"
```
• Path define (Cont.)
  – Hierarchical HDL path defines such as block, sub-system and top level

```verilog
/** Define design hierarchy path for top level */
define `TB_TOP top_tb_top
define `DUT_TOP `TB_TOP.u_top.u_subsystem.u_block

/** Define ENV path for TOP level */
define `ENV_PATH "uvm_test_top.m_top_env.m_subsystem_env.m_block_env"
```
Experiments : NAND controller

- Block diagram for test experiments

- Host protocol design of a NAND controller
- DPI-C test is translated into AXI/AHB UVM sequences and they are fed to AXI/AHB Agent
Experiments: NAND controller

• Results
  – The merged testbench can give you more benefit in TAT.

<table>
<thead>
<tr>
<th>Details</th>
<th>Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>UVM based</td>
<td></td>
</tr>
<tr>
<td>Verilog compile time x Iteration</td>
<td>8260</td>
</tr>
<tr>
<td>Run time x Iteration</td>
<td>16020</td>
</tr>
<tr>
<td>Total sequence develop TAT</td>
<td>24280</td>
</tr>
<tr>
<td>DPI-C + UVM based</td>
<td></td>
</tr>
<tr>
<td>Verilog compile time</td>
<td>413</td>
</tr>
<tr>
<td>C compile time x Iteration</td>
<td>80</td>
</tr>
<tr>
<td>Run time x Iteration</td>
<td>16020</td>
</tr>
<tr>
<td>Total sequence develop TAT</td>
<td>16513</td>
</tr>
</tbody>
</table>

※ Iteration Cost: “The amount of time required to iterate once in sequence development TAT.”
Assumes 20 as the average iterations
Conclusions

• This paper presents a practical scheme to enhance verification turn-around-time (TAT)

• Reduction of sequence development TAT in case of merging DPI-C based codes into UVM testbench via harness interface

• Reusable Harness Interface (RHI)
  – Harness interface, Harness binding, and Path define

• Reduction of debugging TAT in case of using across hierarchy via RHI
Thank you