February 28 – March 1, 2012
Experiences in Migrating a Chip-Level Verification Environment from UVM-EA to UVM-1.x

Authors: Ashish Kumar, S Manikandan, Sasidhar Dudyala, Srishan Thirumalai, Dave Stang
LSI Corporation
Agenda

- Introduction
- Motivation
- Steps Involved
- Recommendations
- Conclusion
Introduction

Reasons for moving to UVM-EA

- Tool independent (maintained by Accellera)
- Open sourced
- Feature rich (for challenging chip verification)
- Portability
- Minimal learning curve
Motivations for moving from UVM-EA to UVM-1.x

• New features which reduce both effort and maintenance
  - New run phases
  - Sequence Library
  - Objections
  - config_db
  - RAL
  - Command Line Processor
Initial Assessment

• The team started listing down the changes required to move to the newer version of UVM
  - There were deprecated features
  - There were optional changes
  - Some changes were implementation specific
Steps involved in moving from UVM-EA to UVM-1.x

- Changes Classified into two sections
  - Automated Changes
  - Non Automated Changes
## Automated Changes (Configuration)

### Passing Configurations

<table>
<thead>
<tr>
<th></th>
<th>UVM-EA</th>
<th>UVM-1.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_config_int</td>
<td>uvm_config_db#(int)::get</td>
<td></td>
</tr>
<tr>
<td>get_config_string</td>
<td>uvm_config_db#(string)::get</td>
<td></td>
</tr>
<tr>
<td>get_config_object</td>
<td>uvm_config_db#(uvm_object)::get</td>
<td>manual intervention required</td>
</tr>
<tr>
<td>set_config_int</td>
<td>uvm_config_db#(int)::set</td>
<td></td>
</tr>
<tr>
<td>set_config_string</td>
<td>uvm_config_db#(string)::set</td>
<td></td>
</tr>
<tr>
<td>set_config_object</td>
<td>uvm_config_db#(uvm_object)::get</td>
<td>manual intervention required</td>
</tr>
</tbody>
</table>

```bash
$ch =~ s/get_config_int\((.*)\)/uvm_config_db#(int)::get(this,\"\",$1)/;
```
Automated Changes (Configuration)

set_config_string exception for sequences

<table>
<thead>
<tr>
<th></th>
<th>UVM-EA</th>
<th>UVM-1.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>set_config_string</td>
<td>&quot;my_sequencer&quot;, &quot;default_sequence&quot;, &quot;my_random_seq&quot;;</td>
<td>uvm_config_db#(class_name)::set(this, &quot;my_sequencer.main_phase&quot;, &quot;default_sequence&quot;, my_random_seq::type_id::get());</td>
</tr>
</tbody>
</table>
## Automated Changes (Phases)

### UVM phase changes

<table>
<thead>
<tr>
<th></th>
<th>UVM-EA</th>
<th>UVM-1.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>function void build()</td>
<td>function void build_phase(uvm_phase phase)</td>
<td></td>
</tr>
<tr>
<td>class::build()</td>
<td>class::build_phase(uvm_phase phase)</td>
<td></td>
</tr>
<tr>
<td>endfunction: build</td>
<td>endfunction: build_phase</td>
<td></td>
</tr>
<tr>
<td>super.build()</td>
<td>super.build_phase(phase)</td>
<td></td>
</tr>
<tr>
<td>task run()</td>
<td>task run_phase(uvm_phase phase);</td>
<td></td>
</tr>
</tbody>
</table>
Automated Changes (Phases)

Exception for test classes
• run() converted to main_phase(uvm_phase)
## Automated Changes (Deprecated Code)

**Sequence(r) utility macros deprecated**

<table>
<thead>
<tr>
<th>UVM-EA</th>
<th>UVM-1.X</th>
</tr>
</thead>
<tbody>
<tr>
<td>`uvm_sequence_utils(</td>
<td>`uvm_object_utils(seq)</td>
</tr>
<tr>
<td>seq,</td>
<td>`uvm_declare_p_sequencer(sequencer)</td>
</tr>
<tr>
<td>sequencer)</td>
<td></td>
</tr>
<tr>
<td>`uvm_sequence_utils_begin(seq,</td>
<td>`uvm_object_utils_begin(seq)</td>
</tr>
<tr>
<td>sequencer)</td>
<td>`uvm_declare_p_sequencer(sequencer)</td>
</tr>
<tr>
<td>`uvm_sequence_utils_end</td>
<td></td>
</tr>
<tr>
<td>`uvm_sequence_param_utils(</td>
<td>`uvm_object_param_utils(seq)</td>
</tr>
<tr>
<td>seq,</td>
<td>`uvm_declare_p_sequencer(sequencer)</td>
</tr>
<tr>
<td>sequencer)</td>
<td></td>
</tr>
<tr>
<td>`uvm_sequencer_utils</td>
<td>`uvm_component_utils</td>
</tr>
<tr>
<td>`uvm_sequencer_utils_begin</td>
<td>`uvm_component_utils_begin</td>
</tr>
<tr>
<td>`uvm_sequencer_utils_end</td>
<td>`uvm_component_utils_end</td>
</tr>
</tbody>
</table>
## Automated Changes (Deprecated Code)

### Additional deprecated code

<table>
<thead>
<tr>
<th>UVM-EA</th>
<th>UVM-1.x</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>uvm_update_sequence_lib_and_item</code></td>
<td>// ea_to_1x Commented</td>
</tr>
<tr>
<td><code>uvm_update_sequence_lib</code></td>
<td>// ea_to_1x Commented</td>
</tr>
<tr>
<td>global_stop_request</td>
<td>// ea_to_1x Commented</td>
</tr>
<tr>
<td>set_global_timeout</td>
<td>// ea_to_1x Commented</td>
</tr>
<tr>
<td>set_global_stop_timeout</td>
<td>// ea_to_1x Commented</td>
</tr>
</tbody>
</table>
Automated Changes (Miscellaneous)

Additional changes
- better performance
- consistency

<table>
<thead>
<tr>
<th>UVM-EA</th>
<th>UVM-1.x</th>
</tr>
</thead>
<tbody>
<tr>
<td>uvm_report_info</td>
<td>`uvm_info</td>
</tr>
<tr>
<td>uvm_report_warning</td>
<td>`uvm_warning</td>
</tr>
<tr>
<td>uvm_report_error</td>
<td>`uvm_error</td>
</tr>
<tr>
<td>uvm_report_fatal</td>
<td>`uvm_fatal</td>
</tr>
<tr>
<td>$psprintf</td>
<td>$sformatf</td>
</tr>
<tr>
<td>$display</td>
<td>`uvm_info</td>
</tr>
<tr>
<td>p_sequencer.uvm_report_info</td>
<td>`uvm_info</td>
</tr>
</tbody>
</table>
Automated Changes (Summary)

Summary

• Script was provided with -t option for testcases.
• The script was easy to write and helped in automating more than 90% of the required changes.
Non-Automated Changes (Virtual Interfaces)

Passing Virtual Interface handles

- UVM-1.x provides new method to pass virtual interface handles.

```vhdl
UVM-EA Interface Wrapper Class:

class vif_wrap #(type vif_type=int) extends uvm_object;
  vif_type vif_inst;

  function void set_vif(vif_type virtual_if);
    vif_inst = virtual_if;
  endfunction : set_vif

  function vif_type get_vif();
    return (vif_inst);
  endfunction : get_vif

endclass : vif_wrap
```

UVM-1.x:
Not required.
Non-Automated Changes (Virtual Interfaces)

Setting the Virtual Interface

**UVM-EA:**

```vhdl
sys_if sys_if0;
...
vif_wrap #(virtual sys_if) vif_sys;
vif_sys = new();
vif_sys.set_vif(sys_if0);
set_config_object("*", "sys_if0", vif_sys, 0);
```

**UVM-1.x:**

```vhdl
sys_if sys_if0;
...
uvm_config_db #(virtual sys_if)::set(null, "*", "sys_if0", sys_if0);
```
Non-Automated Changes (Virtual Interfaces)

Getting the Virtual Interface

**UVM-EA**

```verilog
class UVM-EA {
    uvm_config temp_obj;
    vif_wrap #(virtual sys_if) vif_sys;
    if(!get_config_object(“vif_sys”, temp_obj, 0))
        uvm_report_fatal(get_name(), “Unable to get vif_sys”)
    if(!$cast(vif_sys, temp_obj))
        uvm_report_fatal(get_name(), “Unable to cast vif_sys”) sys_if0 = vif_sys.get_vif();
}
```

**UVM-1.x**

```verilog
if(!uvm_config_db#(virtual sys_if)::get(this,”*”,”sys_if0”,sys_if0))
    `uvm_error (get_name(), “Unable to get sys_if0”)```
**Non-Automated Changes (Test Class)**

**Changes to test cases and base test:**

```plaintext
class test extends base_test;
    task run();
        super.run();
        run_seq.start(sqr_db);
    endtask : run
endclass : test

class base_test extends uvm_test;
    task run();
        reset_seq.start(sqr_db);
        cfg_sys_seq.start(sqr_db);
    endtask : run
endclass : base_test
```
Base_Test Changes

UVM-EA

```plaintext
task run();
    reset_seq.start(sqr_db);
    cfg_sys_seq.start(sqr_db);
endtask : run
```

UVM-1.x

```plaintext
task reset_phase(uvm_phase phase);
    phase.raise_objection(this, "reset_phase");
    reset_seq.start(sqr_db);
    phase.drop_objection(this, "reset_phase");
endtask : reset_phase

task config_phase(uvm_phase phase);
    phase.raise_objection(this, "config_phase");
    cfg_sys_seq.start(sequencer_db);
    phase.drop_objection(this, "config_phase");
endtask : config_phase
```
Non-Automated Changes (Test Class)

Test Changes (script performed automatically)

```
UVM-EA
 task run();
   super.run();
   run_seq.start(sqr_db);
endtask : run

UVM-1.x
 task main_phase(uvm_phase phase);
   phase.raise_objectection(this, "main_phase");
   run_seq.start(sqr_db);
   phase.drop_objectection(this, "main_phase");
endtask : main_phase
```
Non-Automated Changes (Stopping)

Changes in way to stop simulation

- `global_stop_request();` ➔ deprecated
- Raise and drop objections instead

---

**UVM-EA**

```plaintext
task run();
   my_seq.start(sqr_db);
   global_stop_request();
endtask : run
```

**UVM-1.x**

```plaintext
task run_phase(uvm_phase phase);
   phase.raise_object(this,{get_name(), "_", phase.get_name()});
   my_seq.start(sqr_db);
   phase.drop_object(this,{get_name(), "_", phase.get_name()});
endtask : run_phase
```
**Recommendations**

- If starting with OVM, convert to UVM-EA first with the public domain script.
- Avoid wildcards in the scope argument of `uvm_config_db`.
- Objections must be raised & dropped in each phase that consumes time.
- Objections should be used judiciously.
- Use unique string for `raise_objection` and `drop_objection` description argument (helps in debug).
- Run simulation with `+define+UVM_NO_DEPRECATED`.
- Print the UVM version in the log `UVM_VERSION_STRING`.
Conclusion

• The effort required to develop the automation script was minimal and helped us convert more than 90% of the code from UVM-EA to UVM-1.x with ease.
• The move was justified by the many new features of UVM-1.x which were beneficial in improving the quality of our testbench.