



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

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

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

Automated Changes (Configuration)

set_config_string exception for sequences

UVM-EA	UVM-1.x
<pre>set_config_string("my_sequencer", "default_sequence", "my_random_seq");</pre>	<pre>uvm_config_db#(class_name)::set(this, "my_sequencer.main_phase", "default_sequence", my_random_seq::type_id::get());</pre>

Automated Changes (Phases)

UVM phase changes

UVM-EA	UVM-1.x
<code>function void build()</code>	<code>function void build_phase(uvm_phase phase)</code>
<code>class::build()</code>	<code>class::build_phase(uvm_phase phase)</code>
<code>endfunction: build</code>	<code>endfunction: build_phase</code>
<code>super.build()</code>	<code>super.build_phase(phase)</code>
<code>task run()</code>	<code>task run_phase(uvm_phase phase);</code>

Automated Changes (Phases)

Exception for test classes

- `run()` converted to `main_phase(uvm_phase)`

Automated Changes (Deprecated Code)

Sequence(r) utility macros deprecated

UVM-EA	UVM -1.X
<code>`uvm_sequence_utils(seq, sequencer)</code>	<code>`uvm_object_utils(seq) `uvm_declare_p_sequencer(sequencer)</code>
<code>`uvm_sequence_utils_begin(seq, sequencer)</code>	<code>`uvm_object_utils_begin(seq) `uvm_declare_p_sequencer(sequencer)</code>
<code>`uvm_sequence_utils_end</code>	<code>`uvm_object_utils_end</code>
<code>`uvm_sequence_param_utils(seq, sequencer)</code>	<code>`uvm_object_param_utils(seq) `uvm_declare_p_sequencer(sequencer)</code>
<code>`uvm_sequencer_utils</code>	<code>`uvm_component_utils</code>
<code>`uvm_sequencer_utils_begin</code>	<code>`uvm_component_utils_begin</code>
<code>`uvm_sequencer_utils_end</code>	<code>`uvm_component_utils_end</code>

Automated Changes (Deprecated Code)

Additional deprecated code

UVM-EA	UVM-1.x
<code>`uvm_update_sequence_lib_and_item</code>	<code>// ea_to_1x Commented</code>
<code>`uvm_update_sequence_lib</code>	<code>// ea_to_1x Commented</code>
<code>global_stop_request</code>	<code>// ea_to_1x Commented</code>
<code>set_global_timeout</code>	<code>// ea_to_1x Commented</code>
<code>set_global_stop_timeout</code>	<code>// ea_to_1x Commented</code>

Automated Changes (Miscellaneous)

Additional changes

- better performance
- consistency

UVM-EA	UVM-1.x
<pre>uvm_report_info uvm_report_warning uvm_report_error uvm_report_fatal</pre>	<pre>`uvm_info `uvm_warning `uvm_error `uvm_fatal</pre>
<pre>\$psprintf</pre>	<pre>\$sformatf</pre>
<pre>\$display</pre>	<pre>`uvm_info</pre>
<pre>p_sequencer.uvm_report_info</pre>	<pre>`uvm_info</pre>

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.

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:

```
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:

```
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

```
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

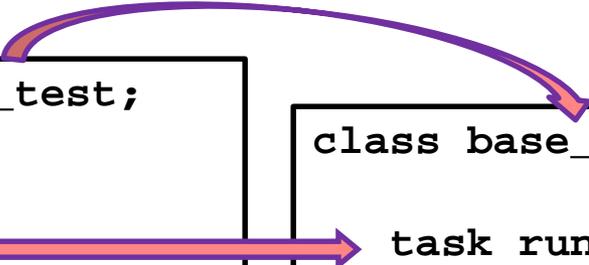
```
if(!uvm_config_db#(virtual sys_if)::get(this,"*", "sys_if0", sys_if0))
    `uvm_error (get_name(), "Unable to get sys_if0")
```

Non-Automated Changes (Test Class)

Changes to test cases and base test:

```
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
```



Non-Automated Changes (Test Class)

Base_Test Changes

UVM-EA

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

UVM-1.x

```
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_objection(this, "main_phase");  
    run_seq.start(sqr_db);  
    phase.drop_objection(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

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

UVM-1.x

```
task run_phase(uvm_phase phase);  
    phase.raise_objection(this, {get_name(), "_", phase.get_name()});  
    my_seq.start(sqr_db);  
    phase.drop_objection(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.