Running the UVM-ML Demos

UVM-ML version 1.5.1

23 August, 2015

Notices

Questions or suggestions relating to this document or product can be sent to

support_uvm_ml@cadence.com

Contents

Introduction

This document provides you with the information you need to run and view the results of the demos that are supplied with the UVM-ML OA package (under ***examples*** directory).

If you need an introduction to UVM-ML OA, read the "Basic Concepts" chapter in the *UVM-ML OA Reference*. For more information about UVM, go to UVM World at: http://www.uvmworld.org.

first_time_demo.sh: Producer / Consumer SC-SV-*e*

The purpose of this demo is to demonstrate a multiple tops environment and usage of TLM ports. This "first-time" demo is a side by side example (parallel trees) as opposed to unified hierarchy (one tree with multiple frameworks). The demo connects all three framework environments (SyetemVerilog, SystemC and *e*) using blocking and non-blocking TLM1 and TLM2 interfaces. The demo can be invoked with $UVM_ML_HOME/ml/examples/first_time_demo.sh.

Note that you can additionally find small versions of side-by-side environments between two frameworks under $UVM_ML_HOME/ml/examples/features/tlm1/prod_cons and $UVM_ML_HOME/ml/examples/features/tlm2/prod_cons.

The following figure illustrates the architecture of this SC-SV-*e* example.

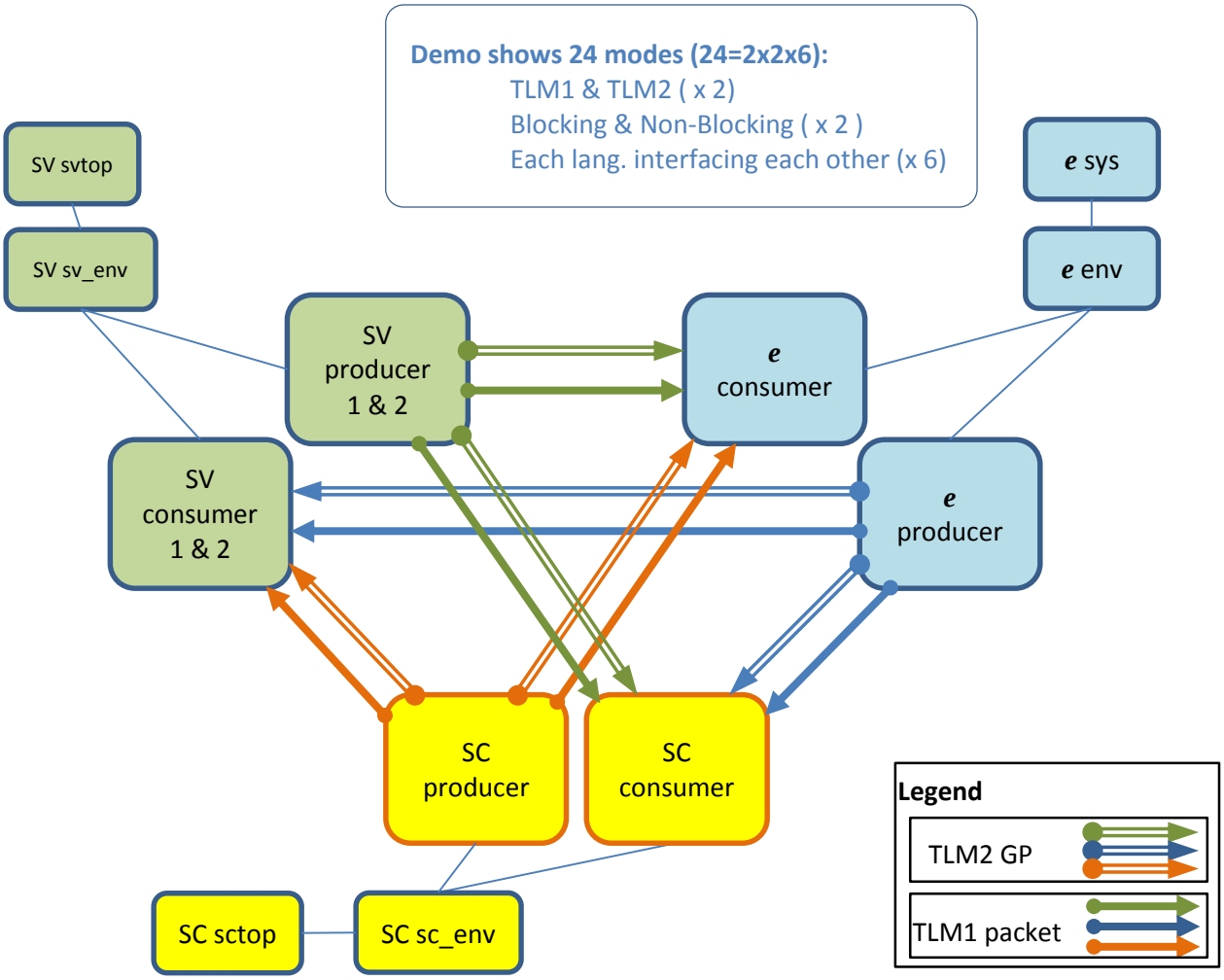

Figure 1: "First-Time" Demo Architecture (SV+SC+e)

Running the demo

Before you run this demo, ensure that the installation and setup steps described in *README_INSTALLATION.txt* were successful. This file can be found at the top of the installed package, in the $UVM_ML_HOME/ml/ directory.

Execute the example as follows:

```
% source $UVM_ML_HOME/ml/examples/first_time_demo.sh <IES|VCS|QUESTA> [-gui]
```

To see what steps are taken by the demo script, go to the directory of the example itself and use the -dry option:

```
% $UVM_ML_HOME/ml/examples/use_cases/side_by_side/sc_sv_e /demo.sh <IES|VCS|
QUESTA> -dry
```

Content of the demo directory

first_time_demo.sh runs the example which resides under $UVM_ML_HOME/ml/examples/use_cases/side_by_side/sc_sv_e. This directory contains the following files:

- PACKAGE_README.txt—Instructions for running the demo
- consumer.e – ***e*** code for the consumer
- consumer.h – SystemC code for the consumer
- consumer.sv - SystemVerilog code for the consumer
- demo.sh—A shell script to invoke the demo
- packet.e – ***e*** code for the TLM1 packet
- packet.h - SystemC code for the TLM1 packet
- packet.sv – SystemVerilog code for the TLM1 packet
- producer.e – ***e*** code for the producer
- producer.h – SystemC code for the producer
- producer.sv – SystemVerilog code for the producer
- sctop.cpp – SystemC code for the top components
- svtop.sv—SystemVerilog code for the top components
- top.e — ***e*** code for the top components
- Makefile—The *makefile* to build and run the demo
- Makefile.ies—A *makefile* used for IES simulation
- Makefile.vcs—A *makefile* used for VCS simulation
- Makefile.questa – A *makefile* used for Questa simulation

Demo Scenario

The demo executes in batch mode, and the output shows six sections of transactions passing from producer to consumer, which represent all combinations of blocking/non-blocking, TLM1/TLM2, and SystemVerilog, SystemC and ***e*** as either initiator or target. This means that this includes the following transactions from producer to consumer:

- non-blocking TLM2 transactions from producer to consumer
- blocking TLM2 transactions from producer to consumer
- non-blocking TLM1 transactions from producer to consumer
- blocking TLM1 transactions from producer to consumer

while there are six combination of producer and consumer: (e/SystemVerilog, e/SystemC, SystemVerilog/e, SystenVerilog/SystemC, SystemC/e, SystemC/SystemVerilog).

The following listing is an excerpt from the log file that is generated by the demo.

```
Running the test ...
UVM_INFO @ 0: uvm_test_top.sv_env.producer_1 [producer_1]

*** Starting non-blocking TLM2 transactions from SV to e
UVM_INFO @ 0: uvm_test_top.sv_env.producer_1 [producer_1] SV producer sends
64: 33 cb 67 08
[0] consumer-@7: Received nb_transport_fw WRITE 0x33 0xcb 0x67 0x08
UVM_INFO @ 5: uvm_test_top.sv_env.producer_1 [producer_1] SV producer sends
64: 00 00 00 00
[5] consumer-@7: Received nb_transport_fw READ 0x33 0xcb 0x67 0x08
UVM_INFO @ 5: uvm_test_top.sv_env.producer_1 [producer_1] SV producer received
64: 33 cb 67 08
UVM_INFO @ 10: uvm_test_top.sv_env.producer_1 [producer_1]

…
*** Starting Blocking TLM1 transactions from SC to SV
[400 ns] SC producer::sending T packet: 17
UVM_INFO @ 400: uvm_test_top.sv_env.consumer_2 [consumer_2] SV consumer::put          17
UVM_INFO @ 401: uvm_test_top.sv_env.consumer_2 [consumer_2] SV consumer::put returns
17
[401 ns] SC producer::sent T packet: 17
[405 ns] SC producer::sending T packet: 18
UVM_INFO @ 405: uvm_test_top.sv_env.consumer_2 [consumer_2] SV consumer::put          18
UVM_INFO @ 406: uvm_test_top.sv_env.consumer_2 [consumer_2] SV consumer::put returns
18
[406 ns] SC producer::sent T packet: 18
[410 ns] SC producer::sending T packet: 19
UVM_INFO @ 410: uvm_test_top.sv_env.consumer_2 [consumer_2] SV consumer::put          19
UVM_INFO @ 411: uvm_test_top.sv_env.consumer_2 [consumer_2] SV consumer::put returns
19
[411 ns] SC producer::sent T packet: 19
...
UVM_INFO @ 600: uvm_test_top.sv_env.consumer_1 [consumer_1] ** UVM TEST PASSED **
```

In each section you can see messages from two language frameworks, including the time stamps indicating coordinated simulation.

The first section in the listing above, shows non-blocking TLM2 transactions generated in the SystemVerilog framework, targeted to the ***e*** consumer. The transactions containing random address and data are written to the ***e*** framework and then read back, comparing the result to the original transaction.

Additional Demos in the Examples Directory

The $UVM_ML_HOME/ml/examples directory contains in addition to **first_time_demo.sh** numerous other examples, organized into three sub-directories:

use_cases directory

The use-case examples utilize multiple features of UVM-ML OA using two different topologies: side-by-side (parallel trees) and unified hierarchy (single tree). The unified-hierarchy multi-language environments, e_over_sv and sv_over_e, demonstrate also sequence layering and hierarchical configuration. Each example has a document describing the example and how to run it.

This is the structure of the directory:

- use_cases/
  - e_over_sv/ - sequence layering in hierarchical environment
  - side_by_side/ - multiple verification components communicating via TLM
    - sc_sv/ - producer consumer example
    - sc_sv_e/ - producer consumer example with 3 frameworks
    - sv_e/ - producer consumer example
    - sc_reusable_transactor_with_uvm_reg/ - SV uvm_reg with a SC transactor
  - sv_over_e/ - sequence layering in hierarchical environment

Features directory

This directory contains small feature oriented examples that demonstrate how to use specific features provided by UVM-ML OA.

This is the structure of the directory:

- features/
  - configuration/ - ML configuration
    - e_sv/ - *e* code configuring SystemVerilog
      - generation_control/ - *e* code configuring build time parameters of SystemVerilog
    - sv_e/ - SystemVerilog code configuring *e*
      - sv_get_config/ - *e* code configuring SystemVerilog
      - sv_set_config/ - SystemVerilog code configuring *e*
    - sv_sc/ - SystemVerilog code configuring SystemC
      - sc_top_sv_subtree/ - SystemC code configuring SystemVerilog
      - sv_top_config_db_ud_types/ - SystemVerilog code configuring SS using uvm_config_db
      - sv_top_sc_subtree/ - SystemVerilog code configuring SystemC
  - phasing/ - ML phasing
    - sv_sc/
      - sc_phasing/- Alignment of the pre-run, runtime and post-run phases between SystemVerilog and SystemC
  - sequences/ - pointer to sequence layering examples in use_cases
  - stub_unit/ - stub unit example for e environment instantiated under SystemVerilog
    - sv_e/
  - synchronization/ - SystemVerilog-SystemC synchronization capabilities
    - sc_sv
      - barrier/ - using barriers
      - event/ - using events
      - time/ - demonstrates fine synchronization with OSCI SystemC

- tcl_debug/ - debug capabilities
  - print_tree/ - printing the ML hierarchy
  - stop_phase/ - managing phase breakpoints
- tlm1/ - TLM1 communication between frameworks
  - prod_cons/ - side by side producer consumer examples
    - e_sv_tlm1/ - blocking and non-blocking from e to SystemVerilog
    - sc_sv_tlm1/ - blocking and non-blocking from SystemC to SystemVerilog
    - sv_e_tlm1/ - blocking and non-blocking from SystemVerilog to e
    - sv_sc_tlm1/ - blocking and non-blocking from SystemVerilog to SystemC
  - sc_sv/
    - TLM1_all_if/ - all TLM1 interfaces
    - analysis_if/ - simple example of analysis port
    - sv_with_sc_ref_model/ - SystemC reference model in SystemVerilog environment
  - sv_e/
    - e_sv_blocking_get/ - *e* code getting packet from SystemVerilog
    - queue_template/ - *e* code transferring complex data structure containing queue
- tlm2/ - TLM2 communication between frameworks
  - prod_cons/ - side by side producer consumer examples
    - e_sv_tlm2/ - blocking and non-blocking from e to SystemVerilog
    - sc_sv_tlm2/ - blocking and non-blocking from SystemC to SystemVerilog
    - sv_e_tlm2/ - blocking and non-blocking from SystemVerilog to e
    - sv_sc_tlm2/ - blocking and non-blocking from SystemVerilog to SystemC
  - sc_sv/
    - sc_initiator_sv_target/ - blocking and non-blocking from SystemC to SystemVerilog
    - sc_initiator_sv_target_sc_connect/ - connect in constructor
    - sc_initiator_sv_target_sc_connect_static_vars/ - using DPI-C to coordinate end of test
    - sc_initiator_sv_target_w_ext/ - using generic payload extensions
    - sv_initiator_sc_target/ - blocking and non-blocking from SystemVerilog to SystemC
  - sv_e/
    - e_initiator_sv_target/ - non-blocking transaction from e to SystemVerilog
  - sv_sv/ - TLM2 between two SystemVerilog environments
    - sv_sv_blocking/ - blocking transactions
    - sv_sv_nonblocking/ - non-blocking transactions
- unified_hierarchy/ - unified hierarchy demonstrated

- sc_e/
  - e_top_sc_subtree/ - SystemC component in e environment with analysis port
  - sc_top_e_subtree/ - *e* code instantiated under SystemC with analysis port
- sc_sv/
  - prod_cons/ - SystemC component in SystemVerilog environment with TLM2
  - sc_top_sv_subtree/ - SystemVerilog verification component in SystemC environment with analysis port
  - sv_top_sc_subtree/ - SystemC verification component in SystemVerilog environment with analysis port
- sv_e/
  - e_top_sv_subtree/ - SystemVerilog verification component in e environment with analysis port
  - prod_cons/ - *e* verification component in SystemVerilog environment with TLM2
  - sv_top_e_subtree/ - *e* verification component in SystemVerilog environment with analysis port
  - e_top_sv_tlm2 - SystemVerilog code in e environment using TLM2

ex_single_lang_uvcs_lib

This directory contains example single-language UVCs. These are used by some of the ML examples to demonstrate reuse of UVCs in multi-language environments.

This is the structure of the directory:

- ex_single_lang_uvcs_lib/ - example UVCs used in other examples
  - ubus_sv/ - UBUS from UVM-SystemVerilog
  - xbus_simple/ - XBUS from UVM-*e*