Universal Verification Methodology (UVM)

Verifying Blocks to IP to SOCs and Systems

Organizers:
Dennis Brophy
Stan Krolikoski
Yatin Trivedi

San Diego, CA
June 5, 2011

Workshop Outline

10:00am – 10:05am	Dennis Brophy	Welcome
10:05am – 10:45am	Sharon Rosenberg	UVM Concepts and Architecture
10:45am – 11:25am	Tom Fitzpatrick	UVM Sequences and Phasing
11:25am – 11:40am	Break	
11:40am – 12:20pm	Janick Bergeron	UVM TLM2 and Register Package
12:20pm – 12:50pm	Ambar Sarkar	Putting Together UVM Testbenches
12:50pm – 1:00pm	All	Q & A

Workshop Outline

✓10:00am – 10:05am	Dennis Brophy	Welcome
✓10:05am – 10:45am	Sharon Rosenberg	UVM Concepts and Architecture
✓10:45am – 11:25am	Tom Fitzpatrick	UVM Sequences and Phasing
✓11:25am – 11:40am	Break	
✓11:40am – 12:20pm	Janick Bergeron	UVM TLM2 and Register Package
12:20pm – 12:50pm	Ambar Sarkar	Putting Together UVM Testbenches
12:50pm – 1:00pm	All	Q & A

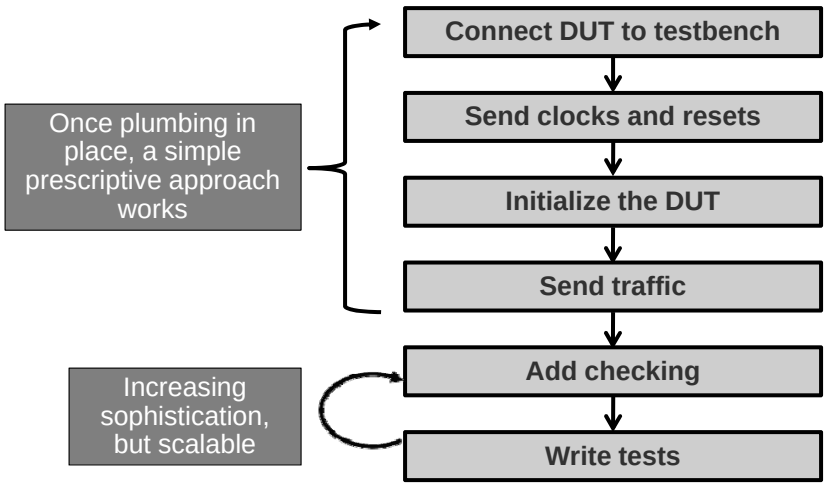
Putting Together UVM Testbenches

Ambar Sarkar
Paradigm Works, Inc.

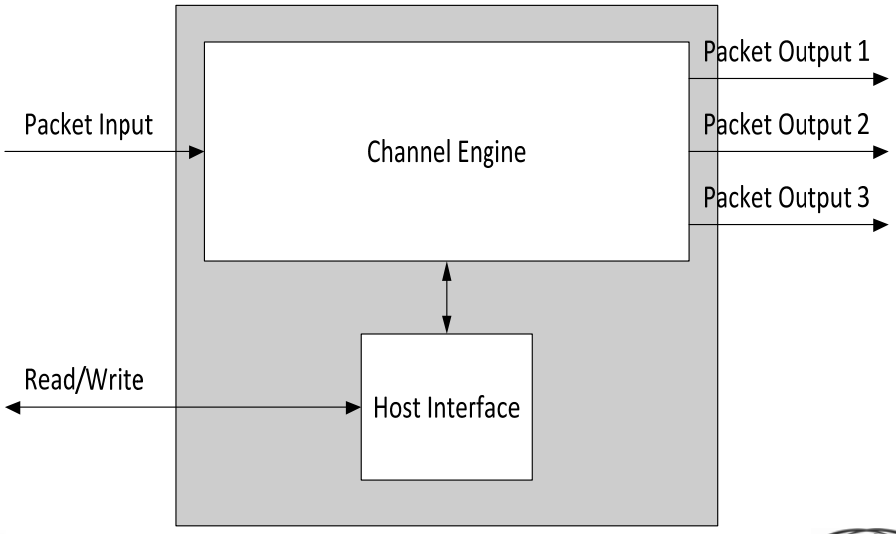
Agenda

- Case studies
  - Several UVM1.0 environments deployed
  - Currently in production
  - Novice to sophisticated teams
- Getting started with UVM is relatively easy
  - Basic tasks remain simple
  - Were able to use a “prescriptive” approach
  - Iteratively developed and scales to any complexity
- Advanced uses
  - Unit to system-level
  - VIP Stacking/Layering

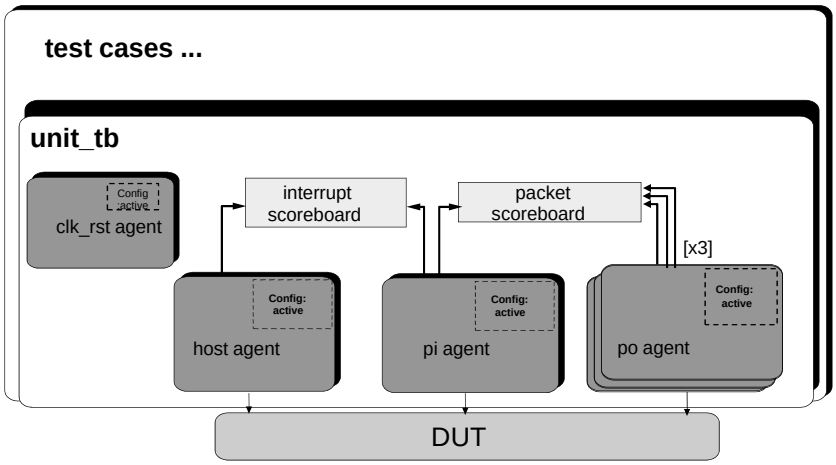
Implementing Basic Steps



Example



Example Environment in UVM



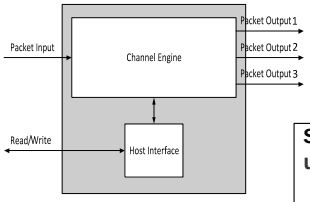
Connect DUT to Testbench

Always use SystemVerilog interface

- *Use clocking blocks for testbench per interface*
- *Use modports for DUT*

Pass interface to the environment

- *Use **uvm_config_db***




```
Set (Top level initial block)
uvm_config_db#(virtual host_if)::set(
                null, “my_tb.*”,“vif”, vif);

Get (In build phase of the agent)
if(!uvm_config_db#(virtual host_if)::get(
                    this,,"vif",vif))
        `uvm_fatal(“NOVIF”, . . . );
```


Connect Clock and Resets

Combine clock and reset as agents in one env

```
class clk_rst_env extends uvm_env;
   clk_rst_cfg cfg;   //! VIP configuration object
   //! bring ports out for convenience
   uvm_analysis_port #(uvm_transaction) rst_mon_ap_out;

   //! Agents in env
   clk_agent clk_agt;
   rst_agent rst_agt;
. . .
```

Define reset as sequences

```
task clk_rst_reset_pulse_sequence::body();

    `uvm_do_with(change_sequence, { level == 0;
                                hold_cycles == init_cycles;
    `uvm_do_with(change_sequence, { level == 1;
                                hold_cycles == assert_cycles;
    `uvm_do_with(change_sequence, { level == 0;
                                hold_cycles == negate_cycles;
```

Initializing the DUT

Add transaction definitions for configuration interface (host)

Add driver code for host

Setup host initialization seq

```
class host_transaction extends uvm_sequence_item;
   ...
   rand bit[31:0] hi_addr;
   rand bit[7:0]  hi_wr_data;
   . . .
task pwr_hi_master_driver::drive_transaction(host_transaction trans);
   if (trans.trans_kind == PWR_HI_WR) begin
       intf.hif_address = trans.hi_addr;
       intf.hif_data = trans.hi_wr_data;
   . . .

task my_env_init_sequence::body();
      regmodel.R1.write(...);
      regmodel.R2.read(...);
   . . .
```


Sending traffic

Similar to initialization

Sequences differ

```
//! sequence body.
task pi_sequence::body();
   pi_transaction#(. . .) req_xn;
   cfg_inst = p_sequencer.cfg;
   forever begin
      p_sequencer.peek_port.peek(req_xn);
      if (!this.randomize()) begin
         `uvm_fatal({get_name(), "RNDFLT"}, "Can't randomize");
      end
      `uvm_do_with(this_transaction,
                  {   trans_kind == req_xn.trans_kind;
                      read_vld_delay inside { [0:15] };
                      . . .
                   });
       . . .
```

Checking and Writing Tests

Add checking in the environment

Add monitor code for all components

Connect the scoreboards

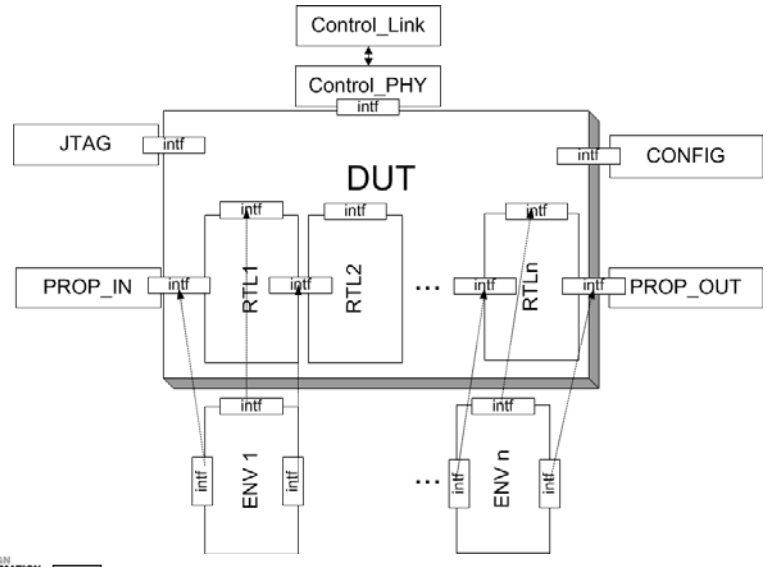
```
class my_env extends uvm_env;
  unit_scoreboard#(uvm_transaction, uvm_transaction) sb;

function void my_env::build_phase(uvm_phase phase);
  ...
  sb = unit_scoreboard#(uvm_transaction, uvm_transaction)
       ::type_id::create({get_name(), "_sb"}, this);

function void pwc_env::connect_phase(uvm_phase phase);
  . . .
  pi_mon.mon_ap_out.connect(sb.post_export);
  po_mon.mon_ap_out.connect(sb.check_export);
```

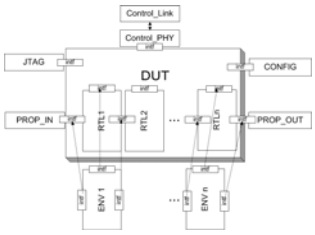
Add test-specific checking in the test as needed

Connecting Unit to System Level



Connecting Unit to System Level: Reuse

- Scoreboarding
  - Reuse SB encapsulated within sub-envs
  - Chain the scoreboards
- Functional Coverage
  - Needed to filter coverage points
- Reuse monitors
  - Avoid duplicated instances
- Gate Simulations
  - Disable monitors
  - Disable internal scoreboards
- Registers
  - Register abstraction reused, but different interfaces used
- Configurations
  - Defined separate system configuration
  - Top level instantiated sub-env configurations
- Sequences
  - Virtual sequencer only at the system level
  - Initialization sequences reused from unit-level
  - Traffic sequences created from scratch at the system level



Connecting Unit to System Level: Prescription

For each sub-env class

Extend sub-env base class

Make all internal components passive

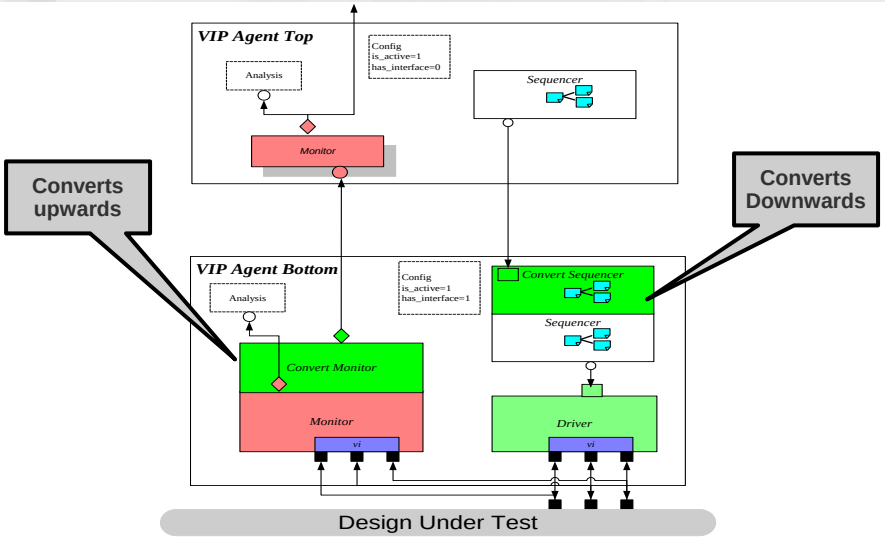
Added sub-env config object to your system config

Declare at system-level:

```
unit_env_cfg       unit_env_cfg_inst;
`uvm_field_object(unit_env_cfg_inst, UVM_REFERENCE)
```

Turn off virtual sequencer at the unit level

VIP Stacking/Layering



Summary

- Getting started with UVM was relatively easy
  - Once initial plumbing in place
    - Basic tasks remain simple
    - Were able to use a “prescriptive” approach
- Able to iteratively develop testbench
    - Scales to any complexity
    - Unit to system
    - Stacking of VIPs
- Deployed across projects and simulation vendors
  - Worked with minor gotchas
    - No UVM issues found
    - Some SystemVerilog support issues among vendors
      - e.g. Inout ports and modports and clocking blocks

Workshop Outline

✓10:00am – 10:05am	Dennis Brophy	Welcome
✓10:05am – 10:45am	Sharon Rosenberg	UVM Concepts and Architecture
✓10:45am – 11:25am	Tom Fitzpatrick	UVM Sequences and Phasing
✓11:25am – 11:40am	Break	
✓11:40am – 12:20pm	Janick Bergeron	UVM TLM2 and Register Package
✓12:20pm – 12:50pm	Ambar Sarkar	Putting Together UVM Testbenches
12:50pm – 1:00pm	All	Q & A

Questions?

- Download UVM from www.accellera.org
  - Reference Guide
  - User Guide
  - Reference Implementation
  - Discussion Forum

Accellera at DAC

- **Accellera Breakfast at DAC: *UVM User Experiences***
  - **An Accellera event sponsored by Cadence, Mentor, and Synopsys**
  - **Tuesday, June 7th, 7:00am-8:30am, Room 25AB**
- **Accellera IP-XACT Seminar**
  - **An introduction to IP-XACT, IEEE 1685, Ecosystem and Examples**
  - **Tuesday, June 7th, 2:00pm-4:00pm, Room 26AB**
- **Birds-Of-A-Feather Meeting**
  - **Soft IP Tagging Standardization Kickoff**
  - **Tuesday, June 7, 7:00 PM-8:30 PM, Room 31AB**

Lunch in Room 20D

Show your Workshop Badge

for entry

Thank You