Using Test-IP Based Verification Techniques in a UVM Environment

Vidya Bellippady  Sundar Haran  Jay O’Donnell
Microsemi Corporation  Microsemi Corporation  Mentor Graphics
San Jose, CA  Hyderabad, India  Seattle, WA
Directed & CRT Test Limitations
Driving Test-IP Development

- Must know low-level VIP & bus protocol to write
- Directed sequences time-consuming to write
- CRT sequences difficult to constrain to application
  - Many seeds/sims may be req’d to hit cases
  - Lots of redundant CRT sequence code
  - Many apps not compatible w/CRT so Directed only option
- Significant user effort to write sequences
- Many simulations/tests to run to reach coverage
Test-IP Benefits compared to CRT and Directed Methods

• Shortens and simplifies test development
  – No need to know VIP & protocol details
  – No need to write sequences, simply write UVM tests
  – Test-IP tests described in a simple UVM config class

• Hits coverage goals in fewer simulations in less time
  – Leverages Test-IP graph-based stimulus targeting
Test-IP Architecture vs DT/CRT

- Test-IP implemented as a UVM sequence
  - Sequence code never changes, simply instantiate
  - Internally leverages graph-based technology (inFact)
  - Behaves like a black box, no need to understand internals

- Test-IP behavior specified in Cfg class
  - Various integral controls (~50) and address range specifiers
Test-IP Configuration

- **Create in UVM test**
  - During build phase
  - Collect in a cfg class
  - Register w/config_db

- **Cfg controls include:**
  - Address map, 1..32 rngs
  - ~50 global controls

- **Common Cfg architecture**
  - Across AXI|AXI4|AHB|ACE
  - Similar features
  - AXI & AHB variant support

- **User requirements**
  - Learn controls, see docs
  - No inFact knowledge reqd
  - No VIP knowledge reqd

```vhls
// Function: do_infact_axi_sequence_config
// This code should be called during the build phase of the UVM test.
function void infact_basic_test::do_infact_axi_sequence_config();
    infact_axi_controls e enable_mask;
    infact_axi_full_protocol_seq_cfg cfg = new("M0_static.cfg"); // Construct the cfg object

    // Add up to 32 addr rngs the Test-IP can target and qualify the traffic configuring an enable mask.
    // configuring an enable mask. Rngs are numbered (arg0), have base & upper addr (arg1 & 2), strin
g eqn the msb configurable capabilities. Different enable mask settings can be used for
    // enable mask = infact_axi_controls e(iax Normal|iax incre|iax bytes 4|iax Norm Sec Data|iax nono
    // cfg.add address range(0,1h600,1hfff,1hDRC2,enable mask) // 4k addr range
    // cfg.add address range(1,1h000,1hfff,1hDRC2,enable mask) // 4k addr range
    // cfg.add address range(2,1h000_0000,1hfff_0000,1hDRC2,enable mask) // 8K

    // controls below are AXI Test-IP defaults unless indicated
    cfg.en_stim_cov = 1; // typically enabled, inFact stimulus coverages manage burst gen
    cfg.en_axi_incr = 1; // global ctrl, restrict types of bursts generated, override addr
    cfg.en_axi_fixed = 1; // enable this, master will use the same ID for every burst
    cfg.en_axi_normal = 1; // if enabled, this master will use the same ID for every burst
    cfg.en_axi_exclusive = 1; // if enabled, this master will use the same ID for every burst
    cfg.en_axi_locked = 1; // if enabled, this master will use the same ID for every burst
    cfg.en_axi_masked = 1; // if enabled, this master will use the same ID for every burst
    cfg.en_axi_id_val = 6; // initial ID value to use when rotating thru IDs, or fixed ID if
    cfg.en_axi_min_read_width = 8; // global controls that restricts (burst) sizes for all addr ranges
    cfg.en_axi_max_read_width = 8; // global controls that restricts (burst) sizes for all addr ranges
    cfg.en_axi_max_write_width = 8; // global controls that restricts (burst) sizes for all addr ranges
    cfg.en_axi_max_write_length = 16; // global ctrl, restricts burst length
    cfg.en_axi_max_lock_length = 16; // global ctrl, restricts burst length
    cfg.en_axi_max_word_length = 1; // # consecutive locked accesses, including the unlock access
    cfg.en_axi_wstrobe_all = 1; // various write strobe settings available..subset shown
    cfg.en_axi_wstrobe_lshift = 1; // ... unclear what this does
    cfg.en_axi_wstrobe_inverse = 1; // ... unclear what this does
    cfg.en_axi_addrRanges = 1; // if 1, cover all legal burst constructions specified per addr range
    cfg.en_axi_addr_range_sequence = 1; // if 1, specify config.addr_range_sequence = (1,2,3);
    cfg.en_axi_shared_data = 1; // if 0, data uses an incrementing count pattern
    cfg.en_axi_fixed_trans_gap = 6; // transaction gaps between bursts, or if en phase seq, between ID
    cfg.en_axi_trans_gap = 8; // gap in axi clocks when enabled, up to 1024 axi clocks allowed
    cfg.en_axi_variable_trans_gap = 1; // config.variable_trans_gap = 8; // max gap value, up to 1024 axi
    cfg.en_axi_max_outstanding_accesses = 1; // Increase +1 to enable interleaved transactions
    cfg.en_axi_phase_seq = 0; // default is 0, create burst-level transactions
    cfg.en_axi_outstanding_ph_reads = 2; // config.max_outstanding_ph_reads = 2; // enable if slave supports dat
    cfg.en_axi_max_rd_data_waits = 4; // config.max_wdata_waits = 4; config.max_rdop_waits = 4;
    cfg.en_axi_phase_adr = 1; // phase sequence: address->data->response
    cfg.en_axi_phase_dar = 1; // phase sequence: address->response->data
    cfg.en_axi_forceAlignedAddr = 0; // to tighten up certain AXI bursts that normally allow un-aligned
    cfg.en_axi_increAddressIn_cmt = 1; // default is 0, enable to reduce scoreboard errors during overlap
    cfg.en_axi_incrLimit = 0; // 0-value lets rule constraints decide increment, recommended in
    cfg.en_axi_write_before_read = 0; // useful to suppress OVL warnings about accessing un-initialized m
    cfg.en_axi_custom_constraint = 0; // advanced feature, used to restrict generation of an AXI subset
    cfg.en_axi_haltOnCoverage = 1; // default enabled
    cfg.en_axi_haltOnBarrier = 6; // advanced feature, terminate on shared UVM barrier testing fabric
    cfg.en_axi_haltOnIteration = 0; // tests to run before finishing

    uvm_config_db #((infact_axi_full_protocol_seq_cfg).::set(this, "#env.axi_slave_agent_seqencer", CFG
endfunction.
```
Test-IP Implementation Details

... AXI Example, if you want to know

- Graph reads cfg info
  - Adjusts graph based on cfg

- All burst options configured
  - Atomic and burst branches
    - Normal | Exclusive | Lock
  - Burst size/length/cache/prot
  - Write strobes/ID selection
  - Addresses in range(s)
  - Data incrementing or random

- Phase-level option
  - ADR | DAR | DADR phases, waits
  - Multiple outstanding rd&wr
  - Out-of-order rd&wr

- Stimulus coverages
  - Highlighted on graph
  - Graph traversed to meet goals
  - Size adapts based on cfg
Using Test-IP to verify an AXI DDR Controller – Before/After Results

<table>
<thead>
<tr>
<th>Metric</th>
<th>CRT/Directed</th>
<th>Test-IP Approach</th>
<th>Benefit</th>
</tr>
</thead>
<tbody>
<tr>
<td>#lines user testbench code</td>
<td>40,000</td>
<td>850</td>
<td>47x less</td>
</tr>
<tr>
<td>#OVM tests</td>
<td>93</td>
<td>7</td>
<td>13x fewer</td>
</tr>
<tr>
<td>Simulation time to coverage</td>
<td>17hrs</td>
<td>15min</td>
<td>68x faster</td>
</tr>
</tbody>
</table>
Test-IP Applications

• AMBA fabrics (ahb | axi | axi4 | ace.. available)
• AXI DDR controllers, routers, switches
• Test-IP supporting AMBA slaves possible
• Other bus protocols where traffic important
  – Pcie, usb, ...
• Application-Specific data generation
  – Currently random or incrementing payloads
  – Could be enhanced to be data-driven (TLM fifo)
  – Could be layered under higher-level sequences
Findings/Lessons Learned Developing and Using Test-IP

- Multiple successes with AMBA fabrics/devices

- Enhancements – developed
  - Concurrent interleaved bursts, same master
  - Sequential address accesses

- Enhancements -- identified
  - Multi-master accesses to same addr avoidance
  - Profile-based traffic generation
  - Test-IP implemented to model bus slaves

- Should be treated like a software product
  - Releases, bug fixes, docs, examples, support...
  - Features and applications will evolve with use