Agenda: Day 3

DAY 3

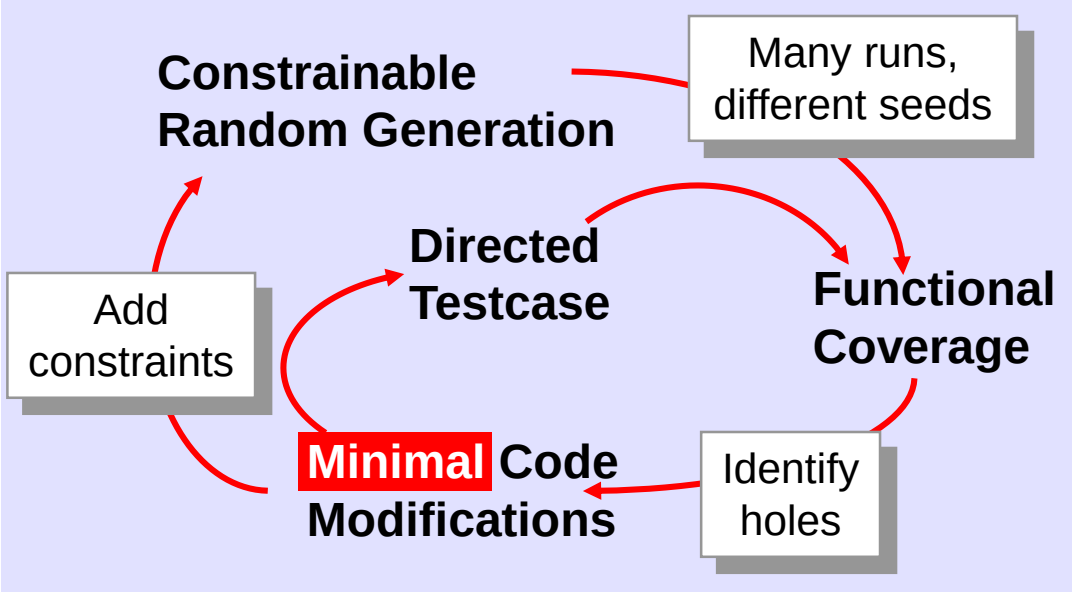
9	UVM Advanced Sequence/Sequencer
10	UVM Phasing and Objections 
11	UVM Register Abstraction Layer (RAL)
12	Summary 

Key Elements of UVM

- Methodology
- Scalable architecture
- Standardized component communication
- Customizable component phase execution
- Flexible components configuration
- Flexible component search & replace
- Reusable register abstraction
- Command line debug

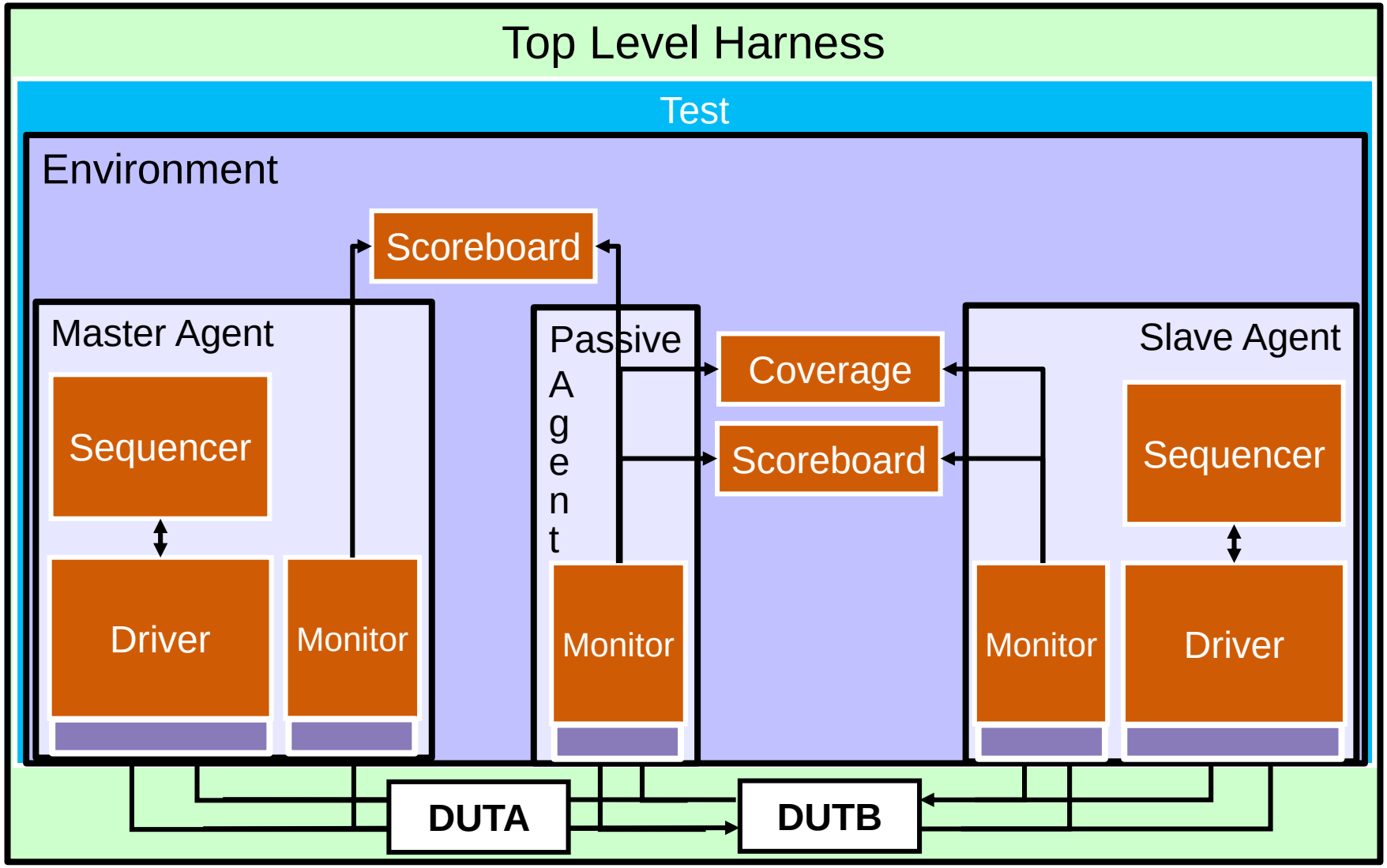
UVM Methodology Guiding Principles

- **Top-down implementation methodology**
  - Emphasizes "*Coverage Driven Verification*"
- **Maximize design quality**
  - More testcases
  - More checks
  - Less code
- **Approaches**
  - Reuse
    - Across tests
    - Across blocks
    - Across systems
    - Across projects
  - One verification environment, many tests
  - Minimize test-specific code



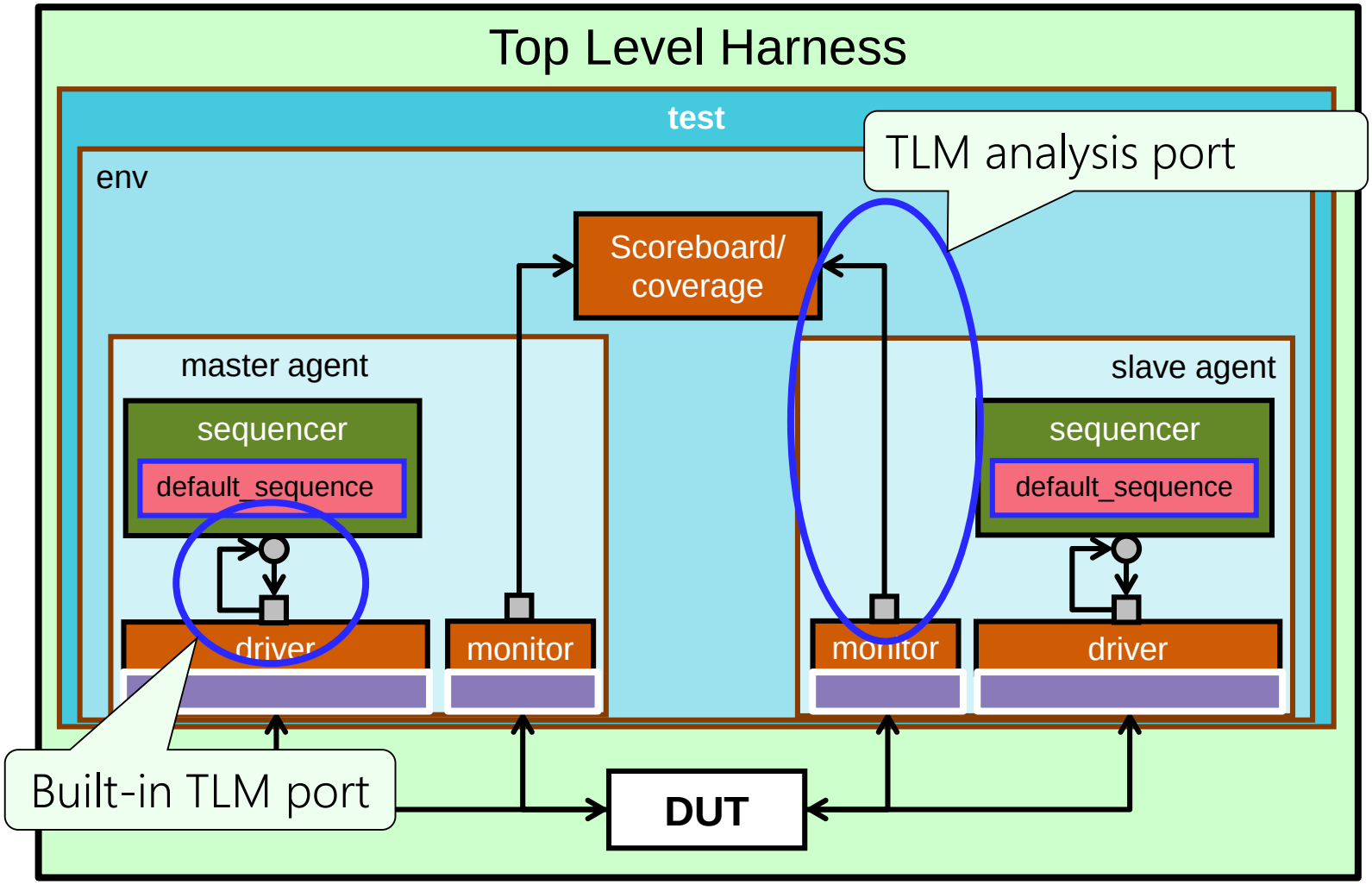
Scalable Architecture

- **Interface based agent enables block to system reuse**



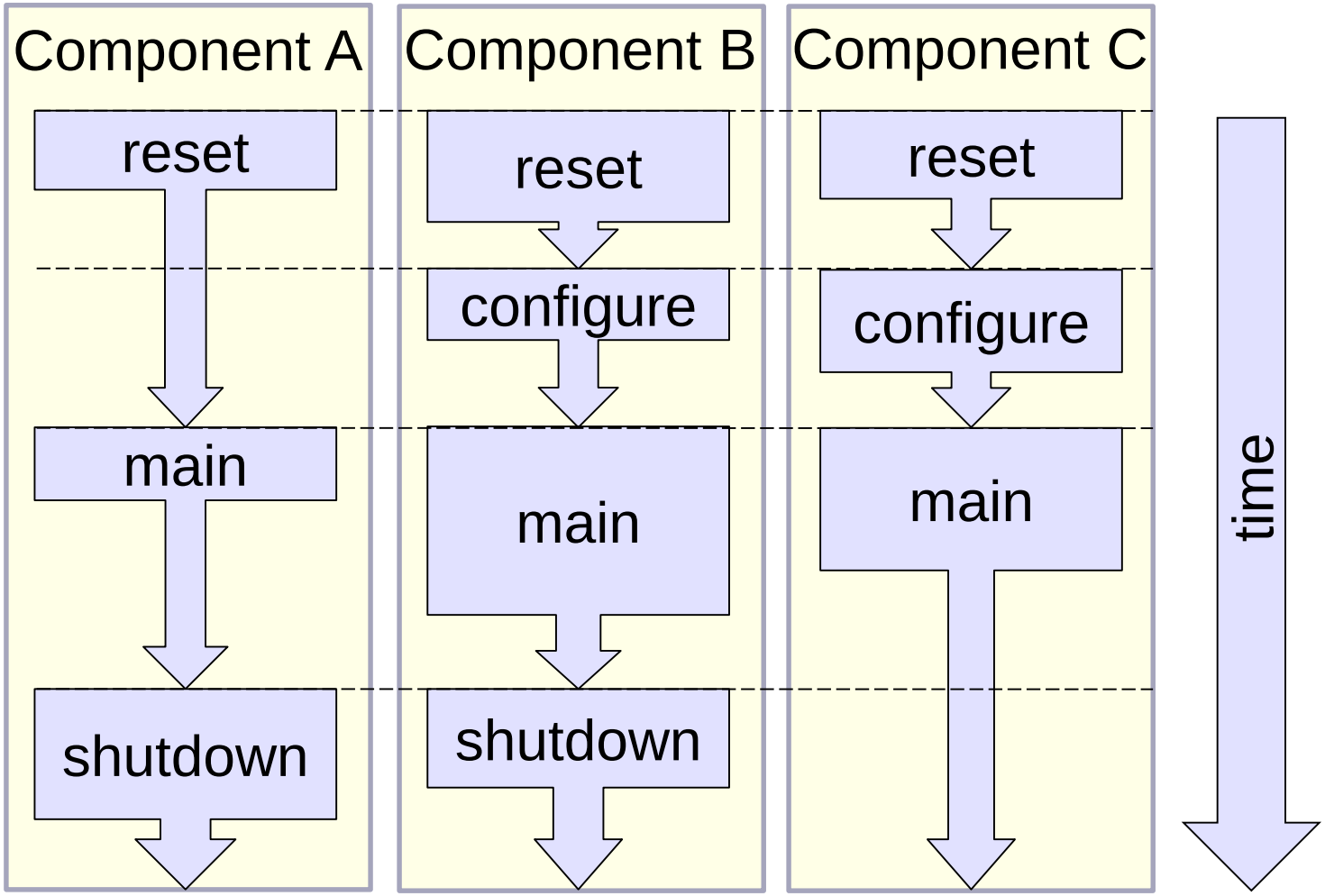
Standardized Component Communication

- **TLM, TLM 1.0, TLM 2.0**



Customizable Component Phase Execution

- **Component phases are synchronized**
  - Ensures correctly organized configuration and execution
  - Can be customized



Flexible Components Configuration (1/2)

- **uvm_config_db#(_type)::get(...)**

```
class driver extends …; // simplified code
  virtual router_io vif;
  function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    if(!uvm_config_db#(virtual router_io)::get(this,"","vif",vif))
      `uvm_fatal("CFGERR", "Driver DUT interface not set");
  endfunction
endclass
```

- **uvm_config_db#(_type)::set(...)**

```
class test_base extends …; // simplified code
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    uvm_config_db#(virtual router_io)::set(this,"env.*","vif", vif);
  endfunction
endclass
```

Flexible Components Configuration (2/2)

- **Sequence execution phase can be specified**
  - **`uvm_config_db#(...)::set(...)`** executes sequence in chosen phase

```
class reset_sequence extends uvm_sequence#(reset_tr); // other code not shown
  function new(string name="reset_sequence");
    super.new(name);
    set_automatic_phase_objection(1);  // UVM-1.2 Only
  endfunction
  virtual task body();
    `uvm_do(req);
  endtask
endclass
```

```
class router_env extends uvm_env; // simplified code
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    uvm_config_db#(uvm_object_wrapper)::set(this,"agt.sqr.reset_phase",
                    "default_sequence", reset_sequence::get_type());
  endfunction
endclass
```

Flexible Component Search & Replace

```
class test_new extends test_base;  ...
  `uvm_component_utils(test_new)
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    set_inst_override_by_type("env.agt", agent::get_type(),
                              new_agt::get_type());
  endfunction
endclass
```

Simulate with:
`simv +UVM_TESTNAME=test_new`

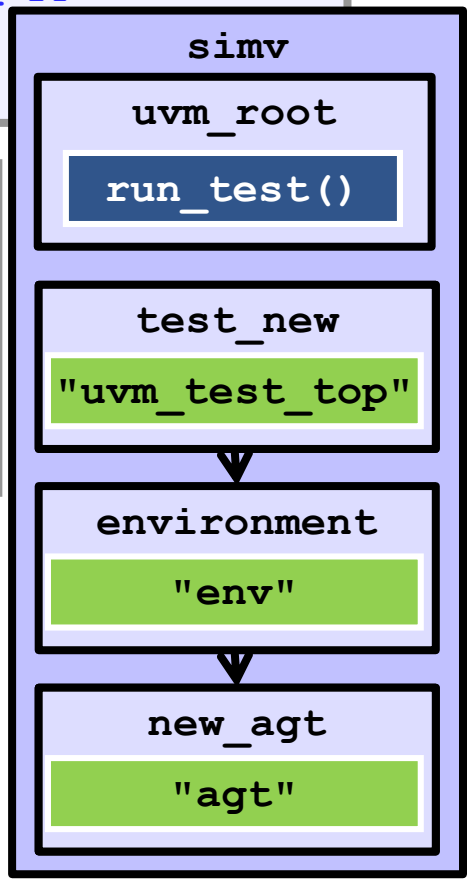
Use component hierarchical path to do component overrides

```
class environment extends uvm_env; ...
  virtual function void build_phase(uvm_phase phase);
    super.build_phase(phase);
    agt = agent::type_id::create("agt", this);
  endfunction
endclass
```

create() used to build component

```
class new_agt extends agent;
  `uvm_component_utils(new_agt)
  function new(string name, uvm_component parent);
  virtual task class_task(…);
    // modified component functionality
  endtask
endclass
```

Modify operation



Standardized Register Abstraction

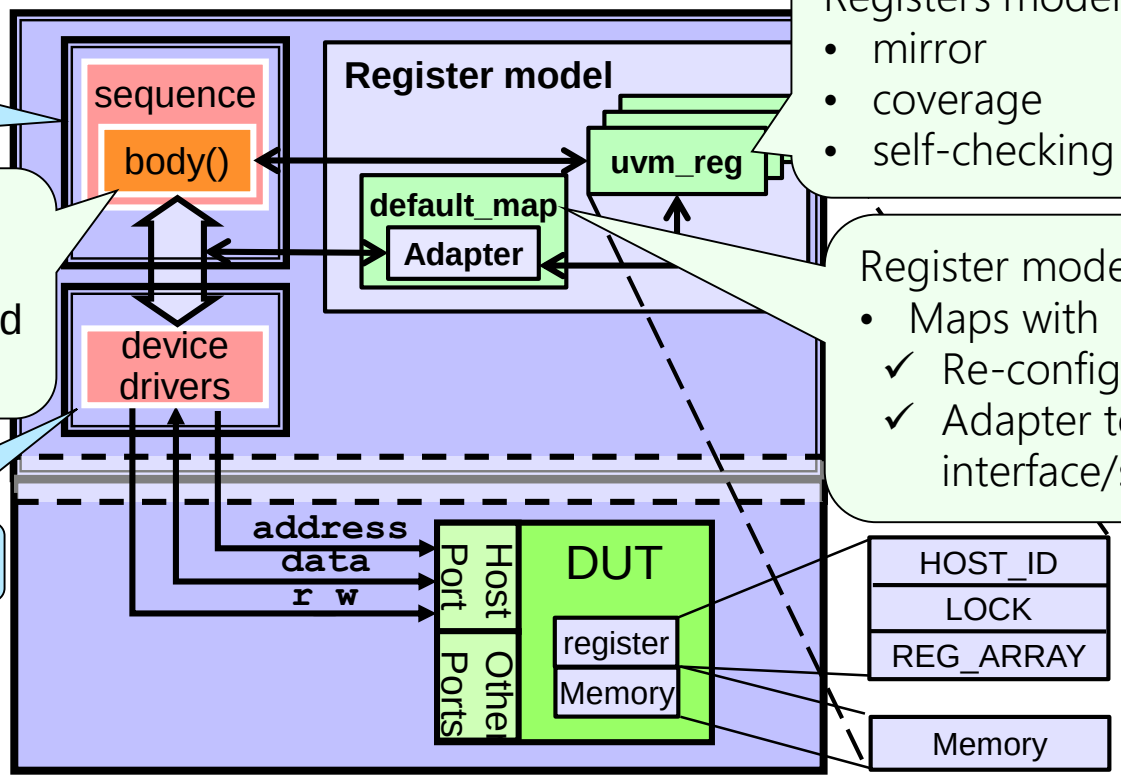
Sequencer unchanged

Registers modeled with built-in:
- mirror
- coverage
- self-checking

New sequence uses register models' abstraction to read and write DUT fields

Register model contains:
- Maps with
  - ✓ Re-configurable address
  - ✓ Adapter to work with any interface/sequencer

Driver unchanged




```
class host_sequence extends uvm_reg_sequence#(uvm_sequence#(host_data));
  virtual task body(); uvm_status_e status; uvm_reg_data_t data;
    regmodel.HOST_ID.read(status, data, UVM_BACKDOOR, .parent(this));
    regmodel.RAM.write(status, 25, '1, UVM_FRONTDOOR, .parent(this));
  endtask
en
```

Access via reg/mem names
Status of execution returned

memory access specifies offset address

Access via front or back door

For debuggin

UVM Command Line Options (1/2)

+UVM_TESTNAME=<class name>

+UVM_VERBOSITY=<verbosity>

+UVM_TIMEOUT=<timeout>,<overridable>

+UVM_MAX_QUIT_COUNT=<count>,<overridable>

+UVM_PHASE_TRACE

+UVM_OBJECTION_TRACE

+UVM_CB_TRACE_ON

+UVM_CONFIG_DB_TRACE

+UVM_RESOURCE_DB_TRACE

+UVM_DUMP_CMDLINE_ARGS

UVM Command Line Options (2/2)

+uvm_set_verbosity=<comp>,<id>,<verbosity>,<phase>

+uvm_set_verbosity=<comp>,<id>,<verbosity>,time,<time>

+uvm_set_action=<comp>,<id>,<severity>,<action>

+uvm_set_severity=<comp>,<id>,<current severity>,<new severity>

+uvm_set_inst_override=<req_type>,<override_type>,<full_inst_path>

+uvm_set_type_override=<req_type>,<override_type>[,<replace>]

+uvm_set_config_int=<comp>,<field>,<value>

+uvm_set_config_string=<comp>,<field>,<value>

+uvm_set_default_sequence=<seqr>,<phase>,<type> // UVM-1.2 Only

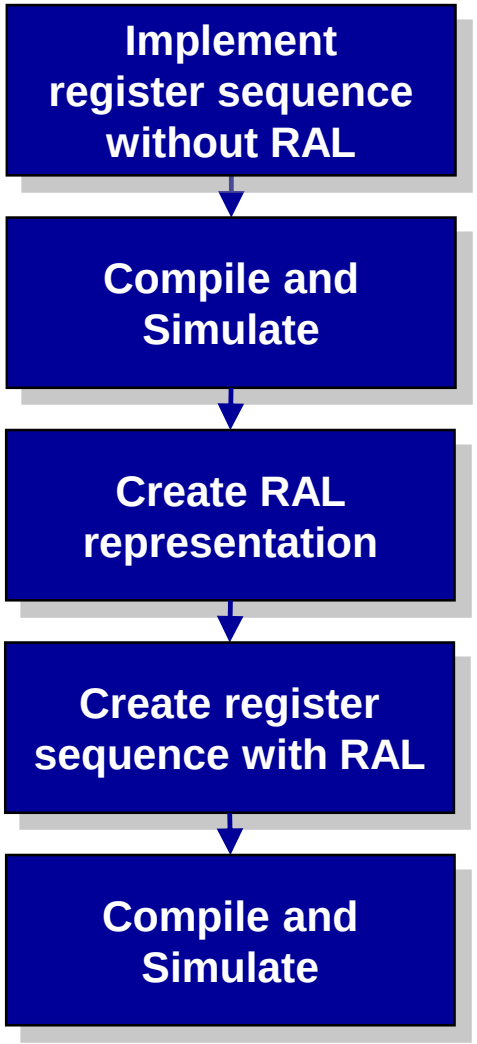
Getting Help

- **Code examples:**
  - $VCS_HOME/doc/examples/uvm
- **Solvnet:**
  - www.solvnet.synopsys.com
- **VCS support:**
  - vcs_support@synopsys.com
- **Synopsys verification video & SNUG:**
  - http://www.synopsys.com/Support/Training/Pages/ces-training-videos-2016.aspx
  - https://www.youtube.com/user/synopsys
  - www.snug-universal.org

Lab 6 Introduction

60 min

Implement RAL



That’s all Folks!