Getting Started with UVM

Vanessa Cooper

Verification Consultant

Agenda

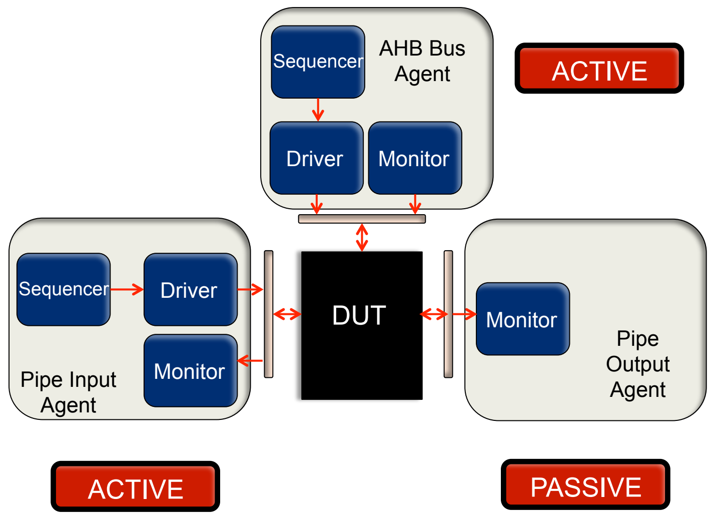
- Testbench Architecture
- Using the Configuration Database
- Connecting the Scoreboard
- Register Model: UVM Reg Predictor
- Register Model: Coverage

Agenda

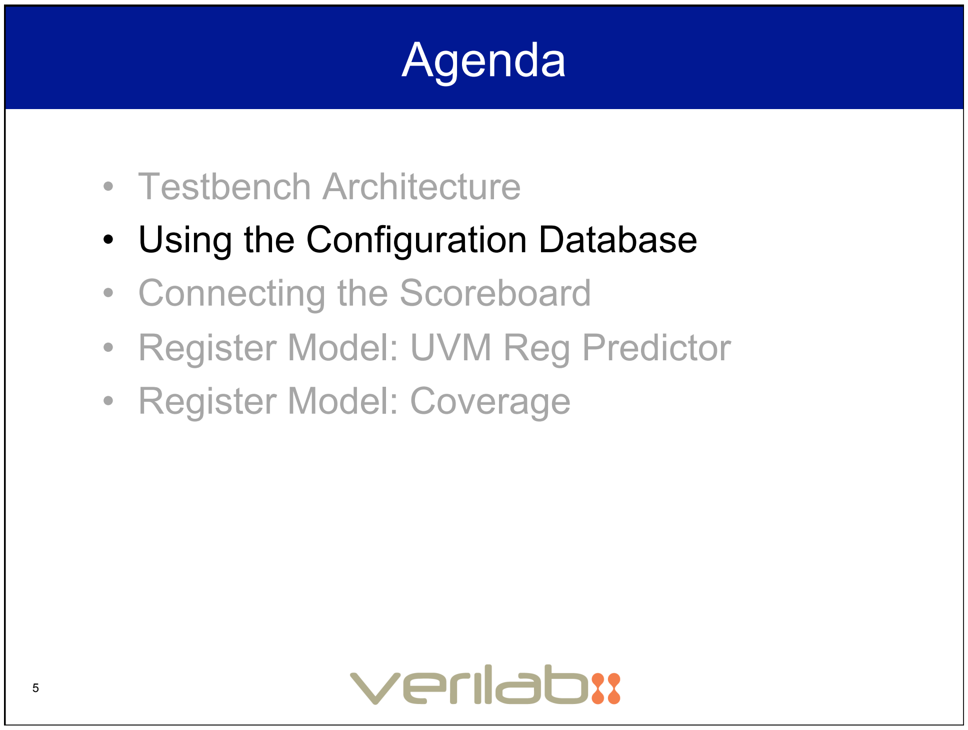
- Testbench Architecture
- Using the Configuration Database
- Connecting the Scoreboard
- Register Model: UVM Reg Predictor
- Register Model: Coverage

3

Testbench Architecture



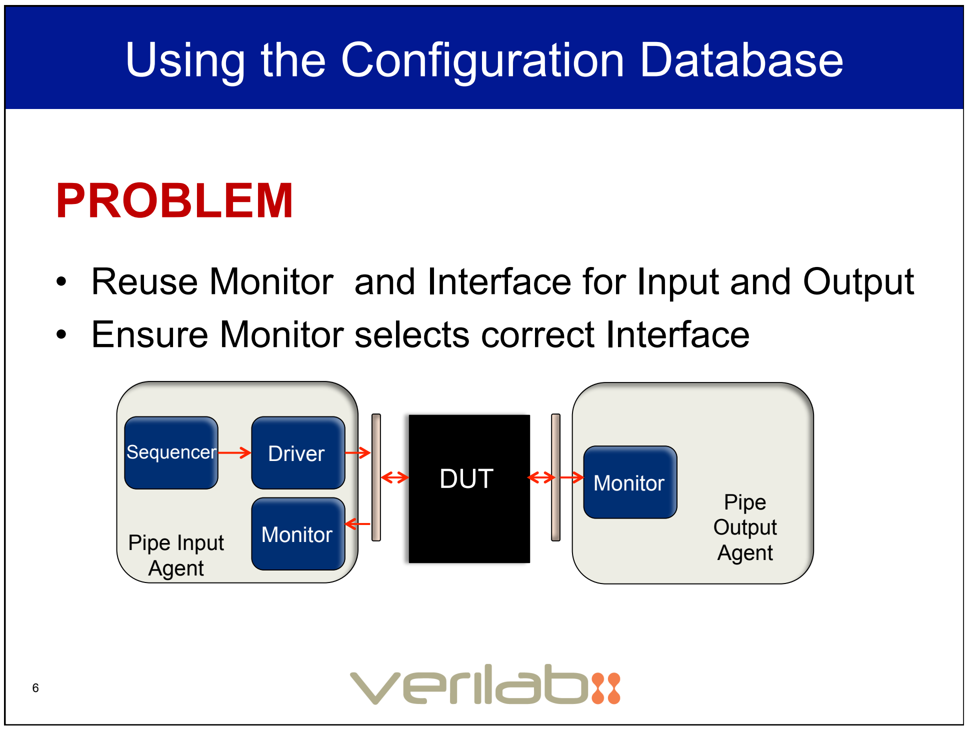
4



Agenda

- Testbench Architecture
- Using the Configuration Database
- Connecting the Scoreboard
- Register Model: UVM Reg Predictor
- Register Model: Coverage

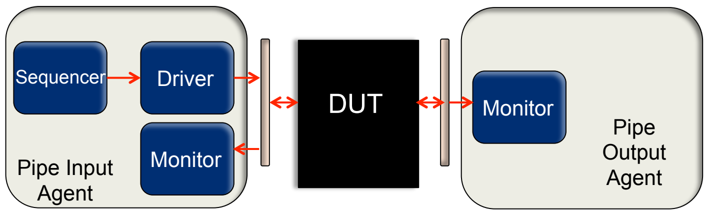
5



Using the Configuration Database

PROBLEM

- Reuse Monitor and Interface for Input and Output
- Ensure Monitor selects correct Interface



6

Using the Configuration Database

Top

```
dut_if vif(.clk(clk),.rst_n(rst_n));

uvm_config_db#(virtual dut_if)::set(uvm_root::get( ), "*",
                                    "dut_intf", vif);
```

```
static function void set(uvm_component cntxt,
                         string        inst_name,
                         string        field_name,
                         T             value)
```

7

verilab

Using the Configuration Database

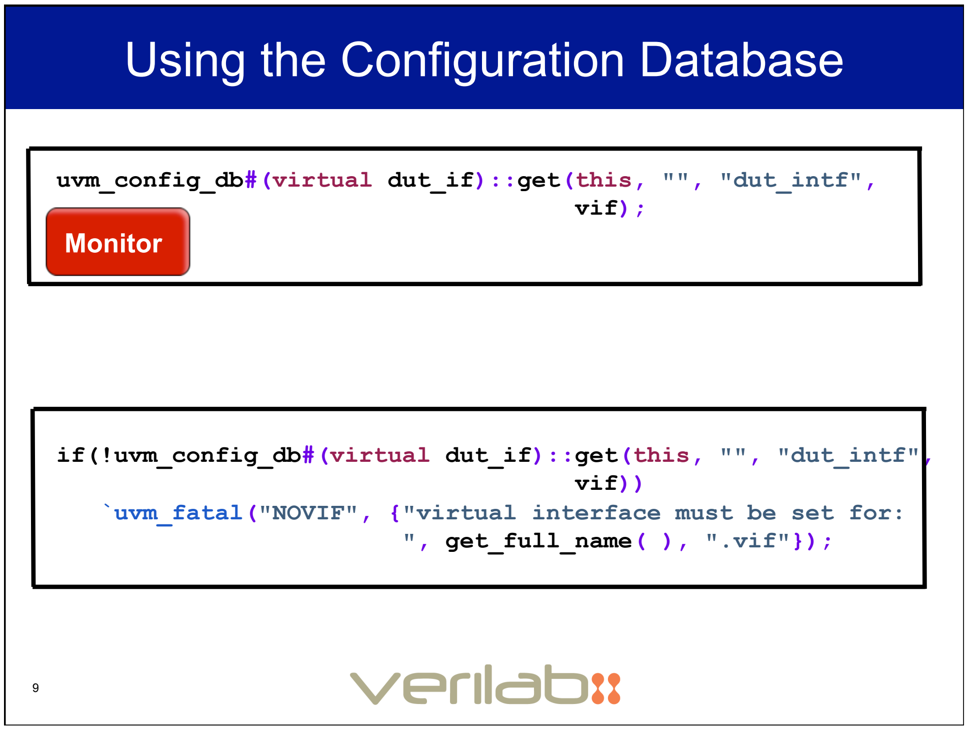
```
uvm_config_db#(virtual dut_if)::get(this, "", "dut_intf",
                                    vif);
```

Monitor

```
static function bit get(    uvm_component cntxt,
                            string        inst_name,
                            string        field_name,
                            ref T         value)
```

8

verilab



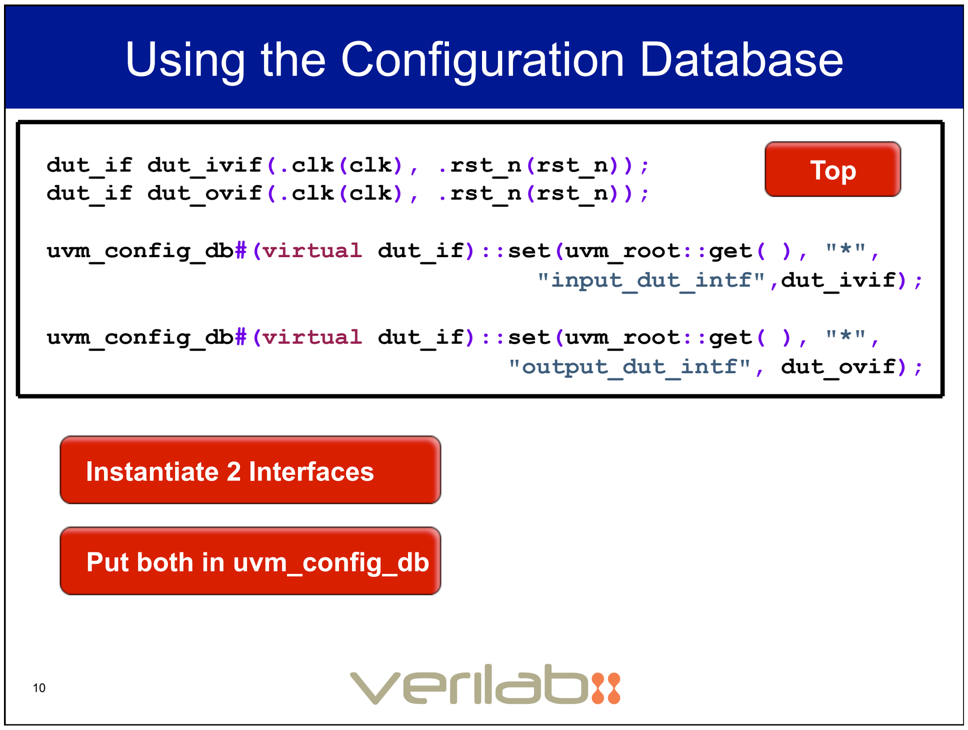
Using the Configuration Database

```
uvm_config_db#(virtual dut_if)::get(this, "", "dut_intf",
                                    vif);
```

Monitor

```
if(!uvm_config_db#(virtual dut_if)::get(this, "", "dut_intf",
                                    vif))
   `uvm_fatal("NOVIF", {"virtual interface must be set for:
                       ", get_full_name( ), ".vif"});
```

9



Using the Configuration Database

```
dut_if dut_ivif(.clk(clk), .rst_n(rst_n));
dut_if dut_ovif(.clk(clk), .rst_n(rst_n));

uvm_config_db#(virtual dut_if)::set(uvm_root::get( ), "*",
                                  "input_dut_intf",dut_ivif);

uvm_config_db#(virtual dut_if)::set(uvm_root::get( ), "*",
                                "output_dut_intf", dut_ovif);
```

Top

Instantiate 2 Interfaces

Put both in uvm_config_db

10

Using the Configuration Database

```
class dut_monitor extends uvm_monitor;
   virtual dut_if vif;
   string monitor_intf;
   ...
endclass: dut_monitor
```

Monitor

```
uvm_config_db#(string)::set(this,"input_env.agent.monitor",
                            "monitor_intf", "input_dut_intf");
```

ENV

```
uvm_config_db#(string)::set(this, "output_env.agent.monitor",
                           "monitor_intf",  "output_dut_intf");
```

11

verilab

Using the Configuration Database

Monitor

```
class dut_monitor extends uvm_monitor;
   virtual dut_if vif;
   string monitor_intf;

   uvm_config_db#(string)::get(this, "", "monitor_intf",
                                           monitor_intf);

   uvm_config_db#(virtual dut_if)::get(this, "",
                                        monitor_intf, vif);
   ...
endclass: dut_monitor
```

12

verilab

Using the Configuration Database

```
uvm_config_db#(virtual dut_if)::set(uvm_root::get( ), "*",
                                "input_dut_intf",dut_ivif);
```

Top

```
uvm_config_db#(virtual dut_if)::set(uvm_root::get( ),
         "*.dut_agent.monitor", "input_dut_intf",dut_ivif);
```

13

Using the Resource Database

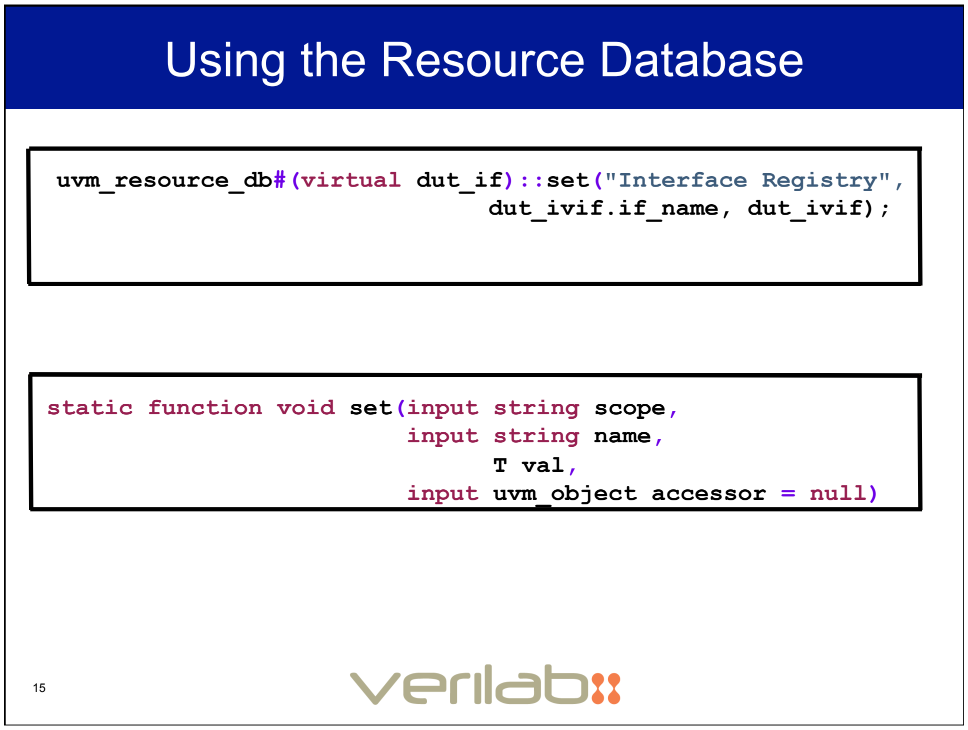
- Let the interface name itself

```
interface dut_if(input clk, rst_n);
   string if_name = $sformatf("%m");
endinterface
```

- Put the interfaces in the Interface Registry

```
uvm_resource_db#(virtual dut_if)::set("Interface Registry",
                               dut_ivif.if_name, dut_ivif);

uvm_resource_db#(virtual dut_if)::set("Interface Registry",
                               dut_ovif.if_name, dut_ovif);
```

14



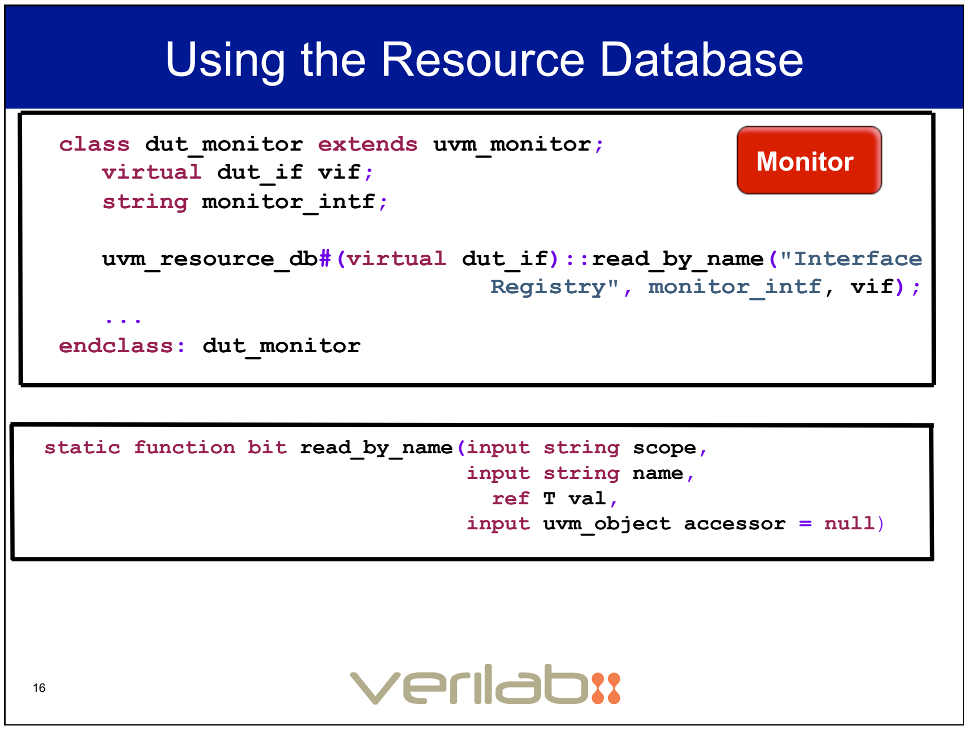
Using the Resource Database

```
uvm_resource_db#(virtual dut_if)::set("Interface Registry",
                                       dut_ivif.if_name, dut_ivif);
```

```
static function void set(input string scope,
                         input string name,
                               T val,
                         input uvm_object accessor = null)
```

15

verilab



Using the Resource Database

```
class dut_monitor extends uvm_monitor;
   virtual dut_if vif;
   string monitor_intf;

   uvm_resource_db#(virtual dut_if)::read_by_name("Interface
                                      Registry", monitor_intf, vif);
   ...
endclass: dut_monitor
```

Monitor

```
static function bit read_by_name(input string scope,
                                 input string name,
                                   ref T val,
                                 input uvm_object accessor = null)
```

16

verilab

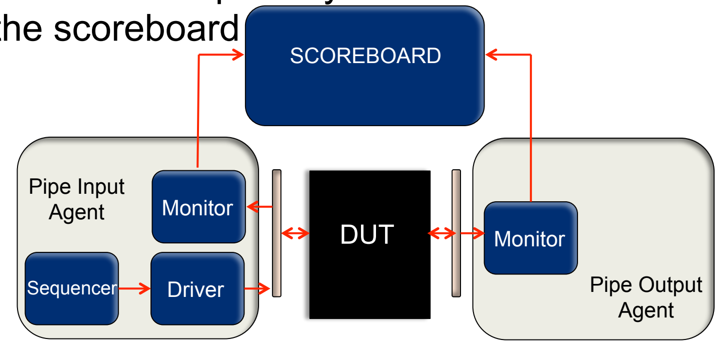
Agenda

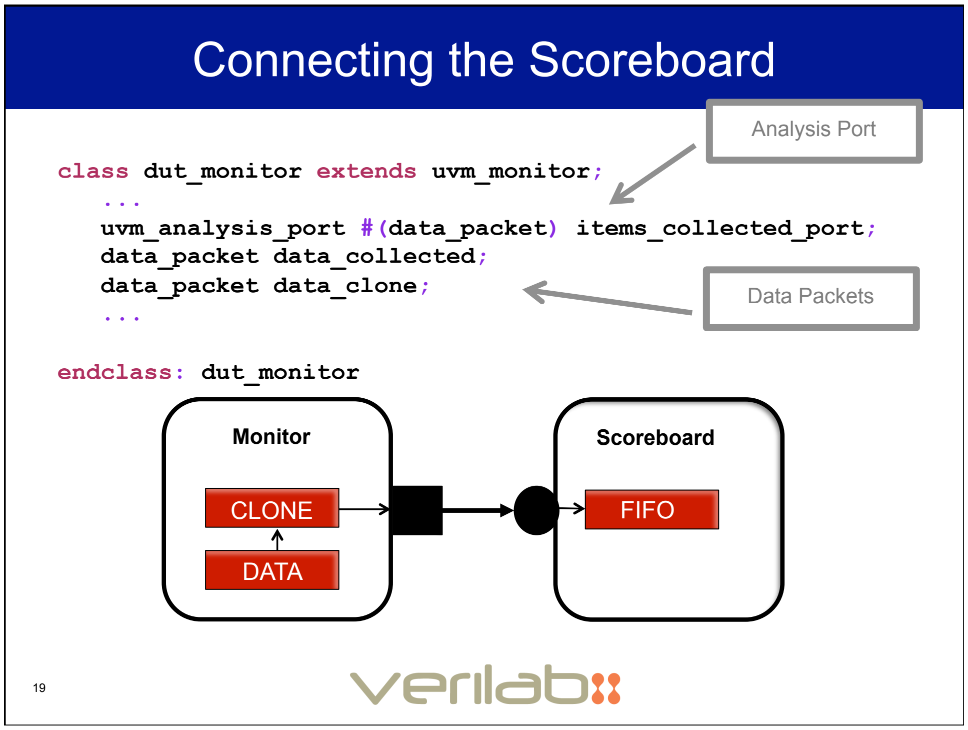
- Testbench Architecture
- Using the Configuration Database
- Connecting the Scoreboard
- Register Model: UVM Reg Predictor
- Register Model: Coverage

Connecting the Scoreboard

PROBLEM

- What is a simple way to connect the monitors to the scoreboard



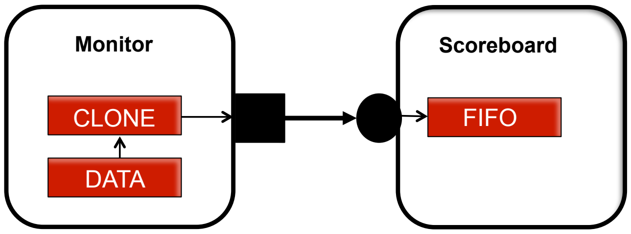


Connecting the Scoreboard

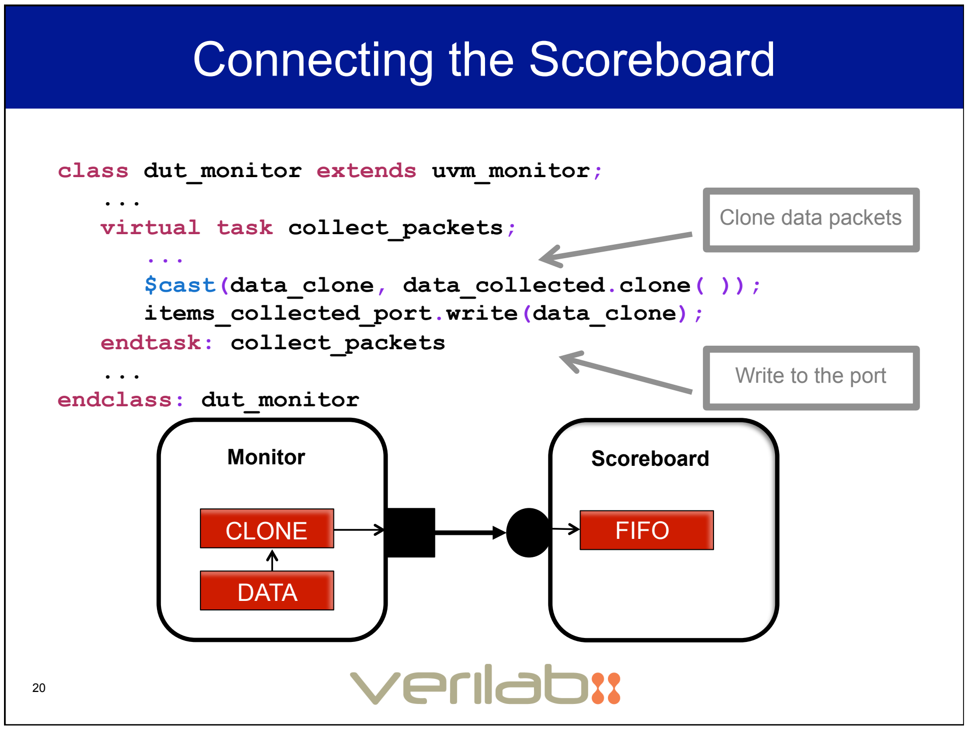
```
class dut_monitor extends uvm_monitor;
   ...
   uvm_analysis_port #(data_packet) items_collected_port;
   data_packet data_collected;
   data_packet data_clone;
   ...
endclass: dut_monitor
```

Analysis Port

Data Packets



19

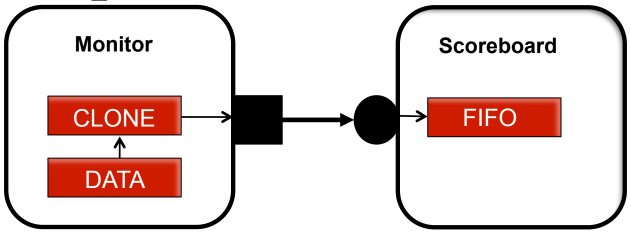


Connecting the Scoreboard

```
class dut_monitor extends uvm_monitor;
   ...
   virtual task collect_packets;
      ...
      $cast(data_clone, data_collected.clone( ));
      items_collected_port.write(data_clone);
   endtask: collect_packets
   ...
endclass: dut_monitor
```

Clone data packets

Write to the port



20

Connecting the Scoreboard

```
class dut_scoreboard extends uvm_scoreboard;
   ...
   uvm_tlm_analysis_fifo #(data_packet) input_packets_collected;
   uvm_tlm_analysis_fifo #(data_packet) output_packets_collected;
   ...

   virtual task watcher( );
      forever begin
         @(posedge top.clk);
         if(input_packets_collected.used( ) != 0) begin
            ...
         end
      end
   endtask: watcher
endclass: dut_scoreboard
```

TLM Analysis Ports

.used() not .size()

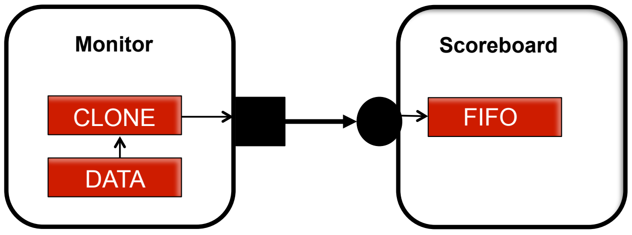
21

Connecting the Scoreboard

```
input_env.agent.monitor.items_collected_port.connect
      (scoreboard.input_packets_collected.analysis_export);
```

ENV

```
 output_env.agent.monitor.items_collected_port.connect
      (scoreboard.output_packets_collected.analysis_export);
```



22

Connecting the Scoreboard

```
virtual task watcher( );
   forever begin
      @(posedge top.clk);
      if(input_packets_collected.used( ) != 0) begin
         ...
      end
   end
endtask: watcher
```

```
virtual task watcher( );
   forever begin
      input_packets_collected.get(input_packets);
         ...
      end
   end
endtask: watcher
```

23

verilab

Agenda

- Testbench Architecture
- Using the Configuration Database
- Connecting the Scoreboard
- Register Model: UVM Reg Predictor
- Register Model: Coverage

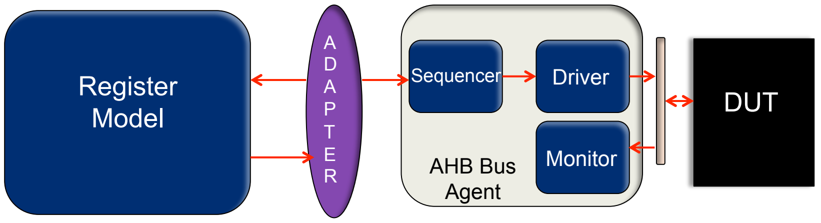
24

verilab

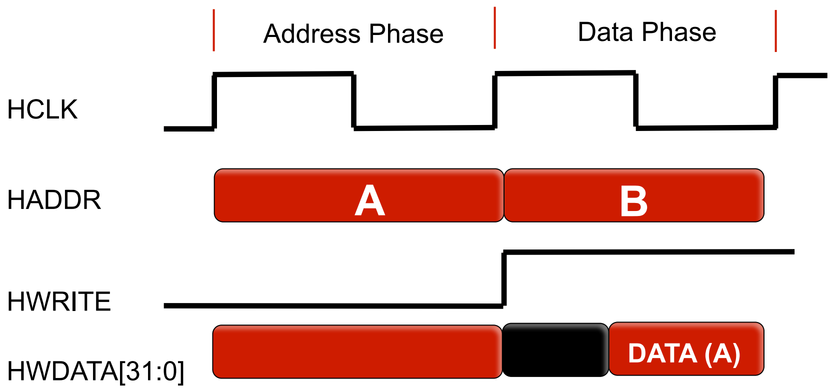
Register Model: UVM Reg Predictor

PROBLEM

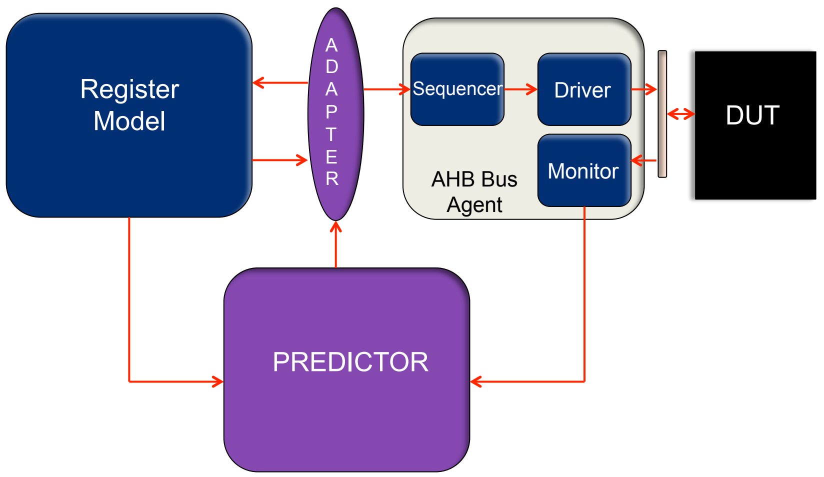
- Use the Register Model with the pipeline AHB bus
- Capture read data accurately



Register Model: UVM Reg Predictor



Register Model: UVM Reg Predictor



27

verilab

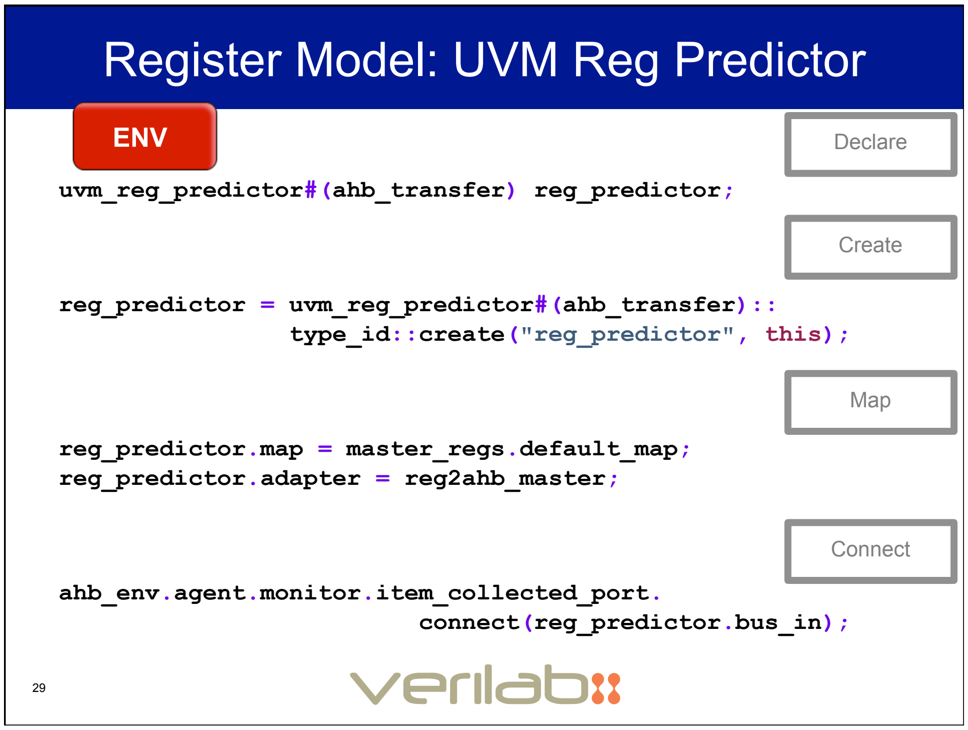
Register Model: UVM Reg Predictor

- build_phase
  - Create the predictor with the bus uvm_sequence_item parameter in your env

- connect_phase
  - Set the predictor map to the register model map
  - Set the predictor adapter to the register adapter
  - Connect the predictor to the monitor

28

verilab



Register Model: UVM Reg Predictor

ENV

Declare

```
uvm_reg_predictor#(ahb_transfer) reg_predictor;
```

Create

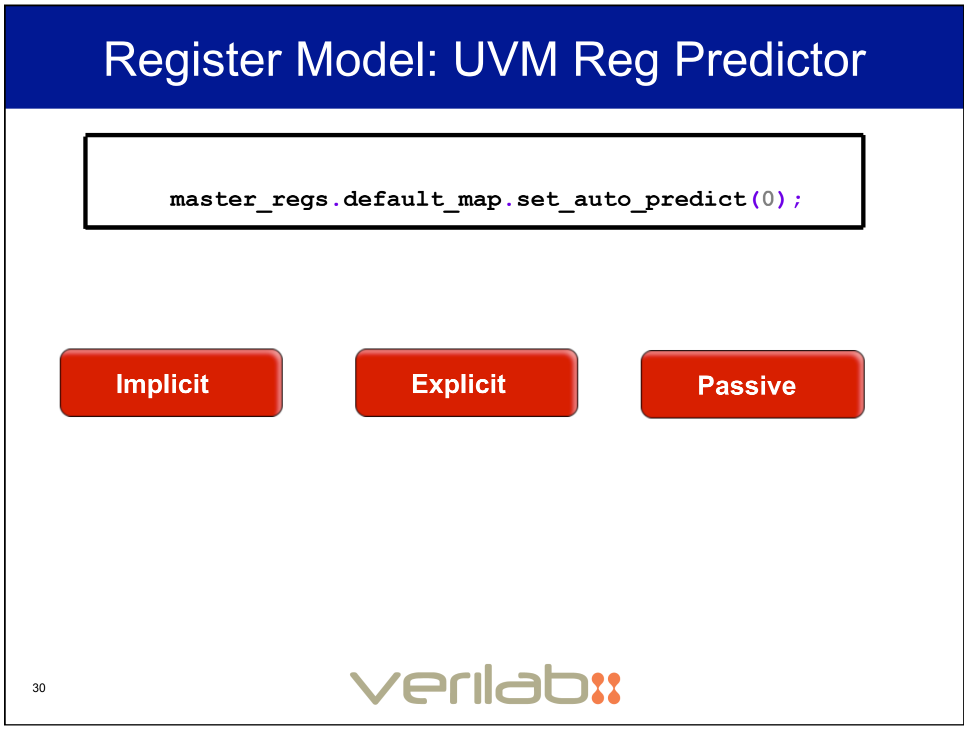
```
reg_predictor = uvm_reg_predictor#(ahb_transfer)::
                    type_id::create("reg_predictor", this);
```

Map

```
reg_predictor.map = master_regs.default_map;
reg_predictor.adapter = reg2ahb_master;
```

Connect

```
ahb_env.agent.monitor.item_collected_port.
                              connect(reg_predictor.bus_in);
```



Register Model: UVM Reg Predictor

```
master_regs.default_map.set_auto_predict(0);
```

Implicit

Explicit

Passive

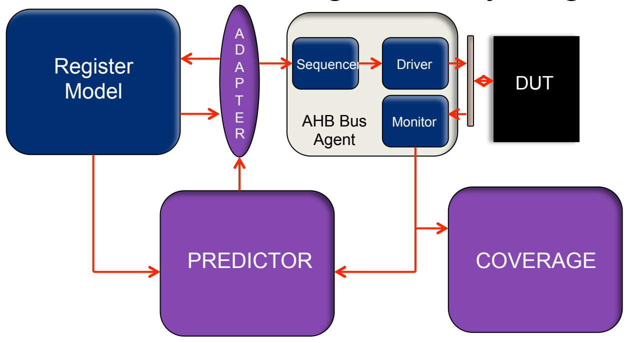
Agenda

- Testbench Architecture
- Using the Configuration Database
- Connecting the Scoreboard
- Register Model: UVM Reg Predictor
- Register Model: Coverage

Register Model: Coverage

PROBLEM

- How do I enable coverage with my Register Model



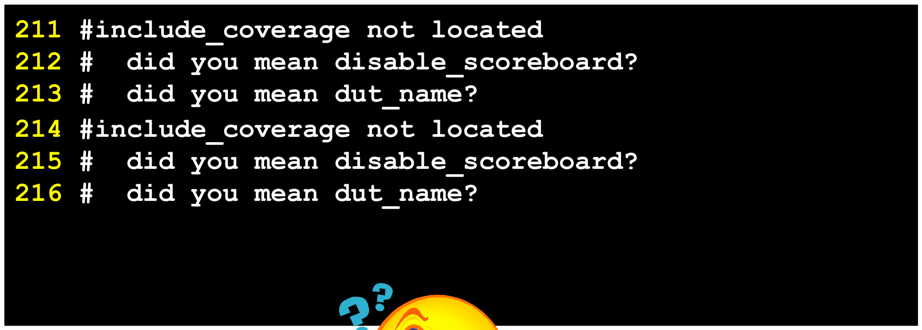
Register Model: Coverage

```
class regs_control_reg extends uvm_reg;

   rand uvm_reg_field control;

   function new(string name = "regs_control_reg");
       super.new(name, 32, build_coverage(UVM_CVR_ALL));
    endfunction: new

   virtual function void build( );
       ...
   endfunction: build

   `uvm_object_utils(regs_control_reg)

endclass: regs_control_reg
```

Specify Coverage

33

Register Model: Coverage



```
211 #include_coverage not located
212 #   did you mean disable_scoreboard?
213 #   did you mean dut_name?
214 #include_coverage not located
215 #   did you mean disable_scoreboard?
216 #   did you mean dut_name?
```

34

Register Model: Coverage

```
class base_test extends uvm_test;
   ...

   `uvm_component_utils(base_test)

   function new(string name, uvm_component parent);
      super.new(name, parent);
      uvm_reg::include_coverage("*", UVM_CVR_ALL);
   endfunction: new

   ...
endclass: base_test
```

Include Coverage

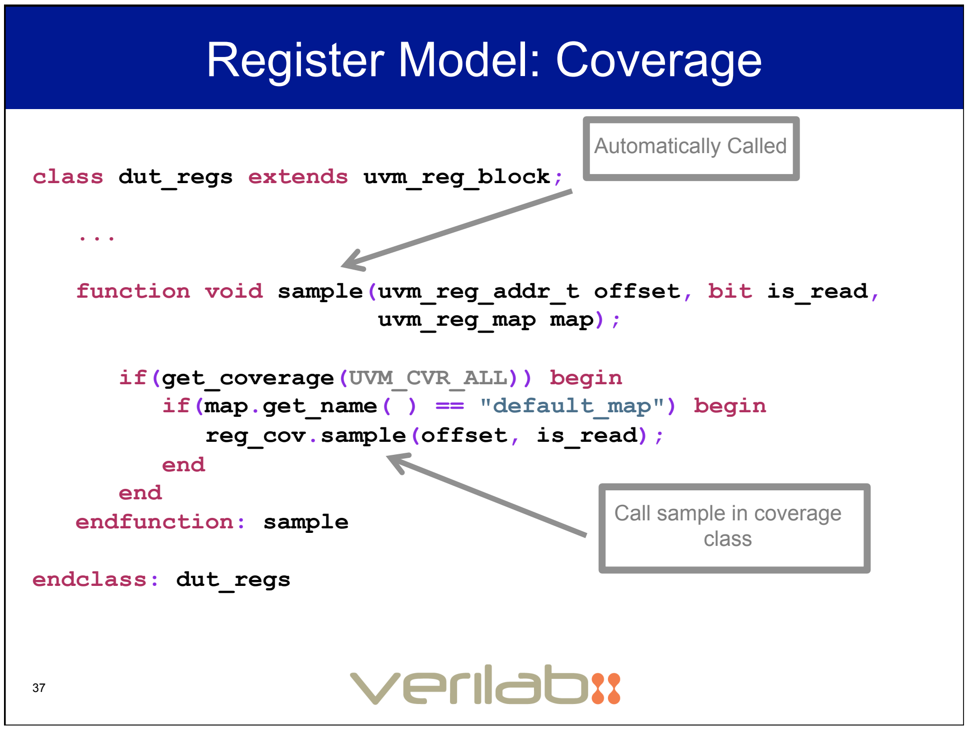
35

Register Model: Coverage

```
class dut_regs extends uvm_reg_block;
   ...
   reg_coverage reg_cov;

   virtual function void build( );
      if(has_coverage(UVM_CVR_ALL)) begin
         reg_cov = reg_coverage::type_id::create("reg_cov");
         set_coverage(UVM_CVR_ALL);
      end
      ...
   endfunction: build

   `uvm_object_utils(dut_regs)
endclass: dut_regs
```

Instantiate coverage class

36



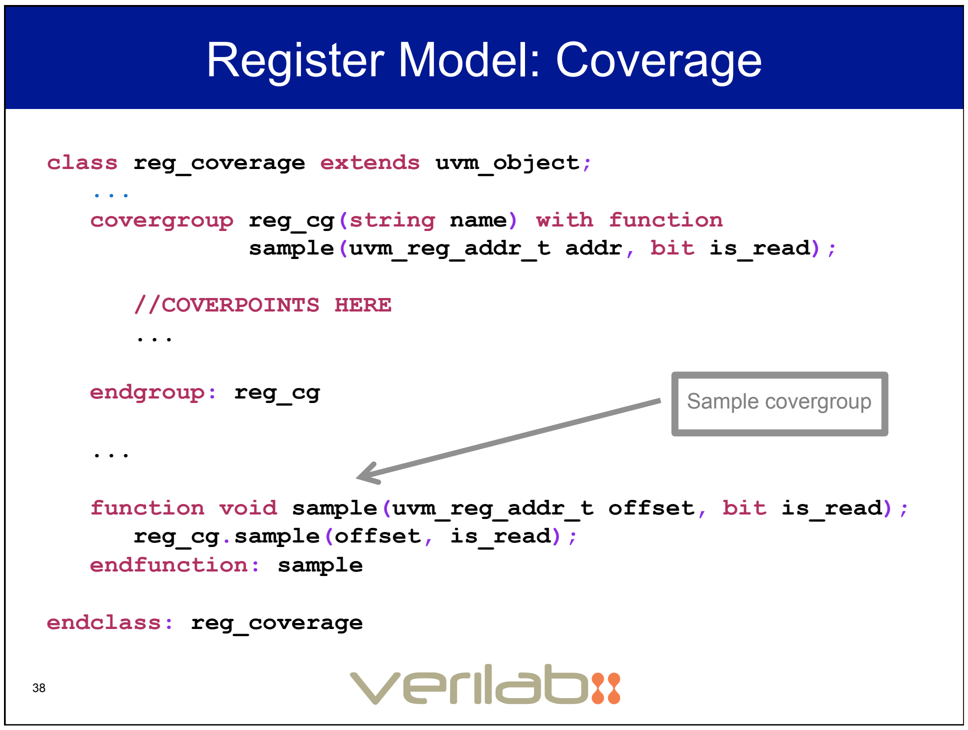
Register Model: Coverage

Automatically Called

```
class dut_regs extends uvm_reg_block;

   ...

   function void sample(uvm_reg_addr_t offset, bit is_read,
                        uvm_reg_map map);

      if(get_coverage(UVM_CVR_ALL)) begin
         if(map.get_name( ) == "default_map") begin
            reg_cov.sample(offset, is_read);
         end
      end
   endfunction: sample

endclass: dut_regs
```

Call sample in coverage class

37



Register Model: Coverage

```
class reg_coverage extends uvm_object;
   ...
   covergroup reg_cg(string name) with function
              sample(uvm_reg_addr_t addr, bit is_read);

      //COVERPOINTS HERE
      ...

   endgroup: reg_cg

   ...

   function void sample(uvm_reg_addr_t offset, bit is_read);
      reg_cg.sample(offset, is_read);
   endfunction: sample

endclass: reg_coverage
```

Sample covergroup

38

Register Model: Coverage

Covergroup

```
ADDR: coverpoint addr {
   bins mode = {'h00};
   bins cfg1 = {'h04};
   bins cfg2 = {'h08};
   bins cfg3 = {'h0C};
   }

RW: coverpoint is_read {
   bins RD = {1};
   bins WR = {0};
   }

ACCESS: cross ADDR, RW;
```

39

Questions

- Testbench Architecture
- Using the Configuration Database
- Register Model: UVM Reg Predictor
- Register Model: Coverage
- Connecting the Scoreboard

40