VERIFICATION ACADEMY

Basic OVM

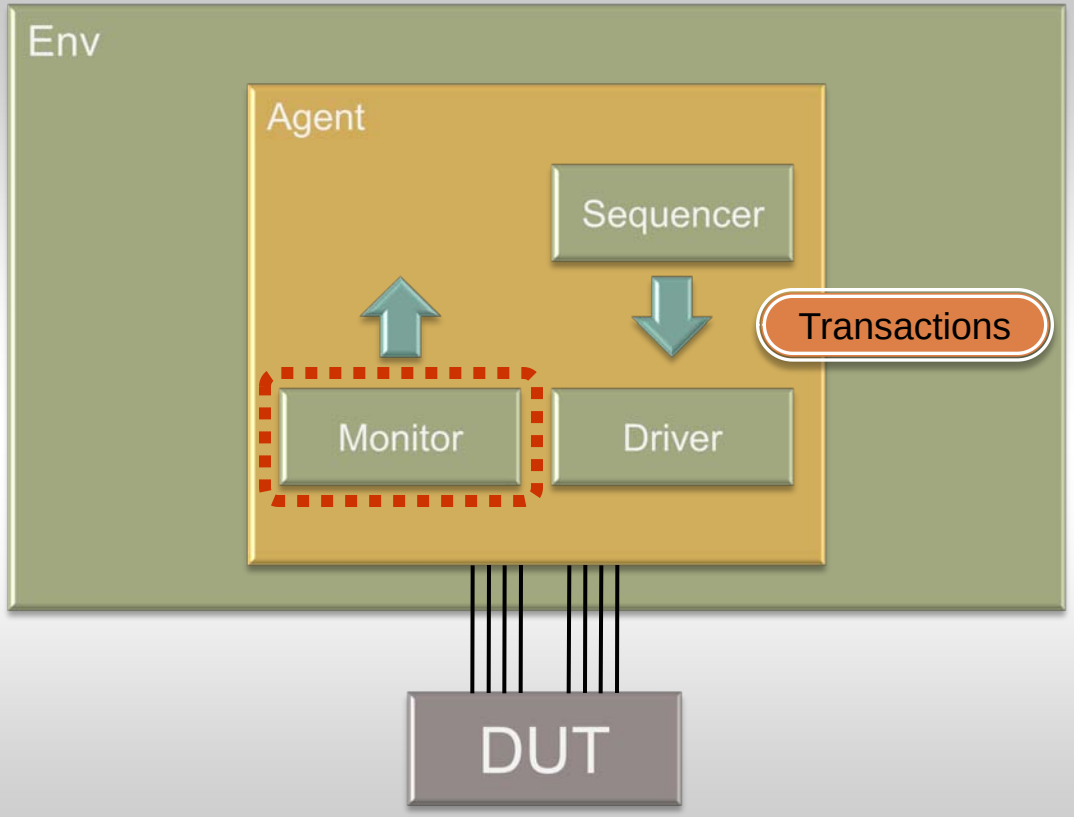
Monitors and Subscribers

John Aynsley
CTO, Doulos

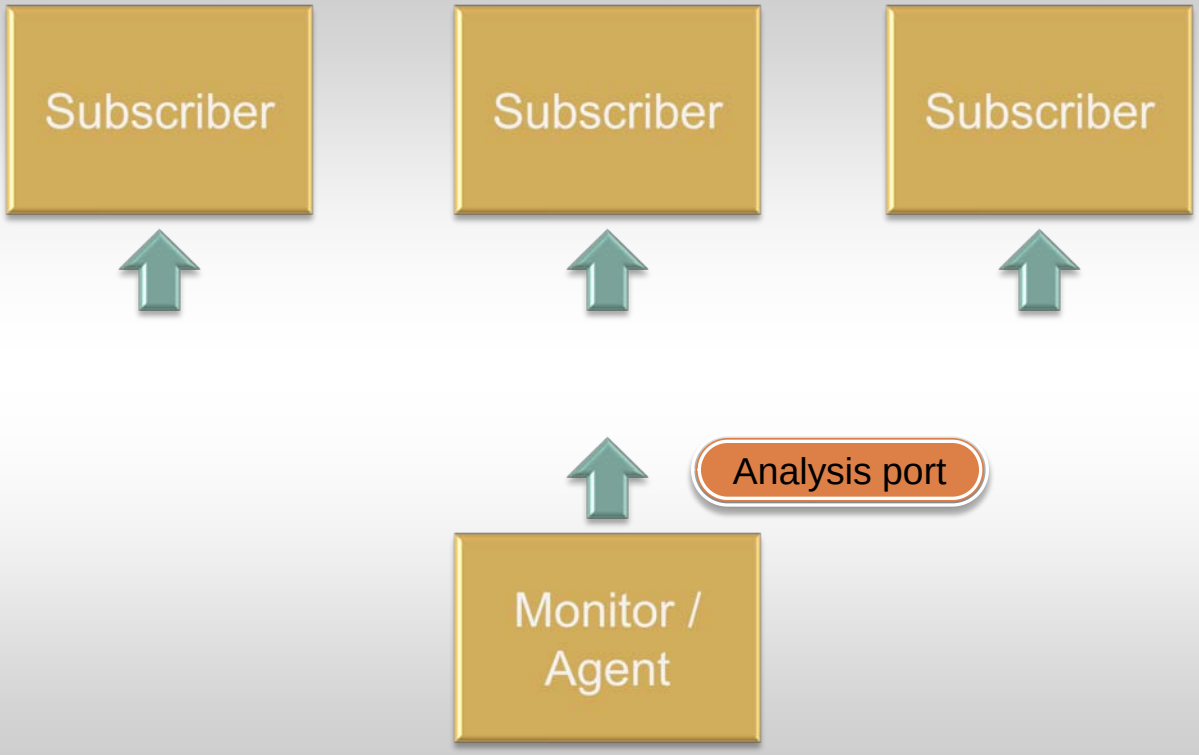
academy@mentor.com
www.verificationacademy.com

Mentor Graphics®

Monitor



Analysis Ports



Monitor

```
class my_monitor extends ovm_monitor;
```

Monitor

```
class my_monitor extends ovm_monitor;

  `ovm_component_utils(my_monitor)

  ovm_analysis_port #(my_transaction) aport;
```

Monitor

```
class my_monitor extends ovm_monitor;

  `ovm_component_utils(my_monitor)

  ovm_analysis_port #(my_transaction) aport;

  virtual dut_if dut_vi;

  function new ...

  function void build;
    super.build();
    aport = new("aport", this);
    ...
```

Monitor

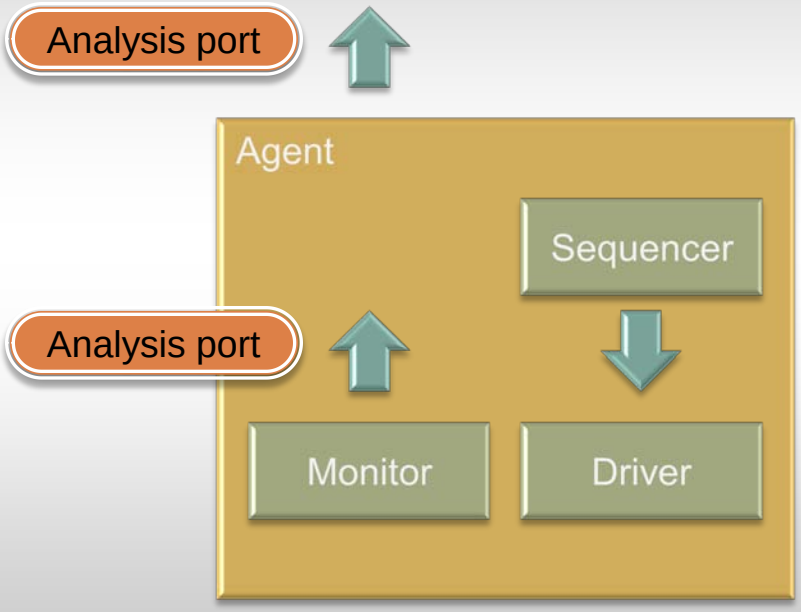
```
class my_monitor extends ovm_monitor;
  ...
  task run;
    forever
    begin
      my_transaction tx;

      @(posedge dut_vi.clock);
      tx = my_transaction::type_id::create("tx");
      tx.cmd  = dut_vi.cmd;
      tx.addr = dut_vi.addr;
      tx.data = dut_vi.data;
```

Monitor

```
class my_monitor extends ovm_monitor;
  ...
  task run;
    forever
    begin
      my_transaction tx;

      @(posedge dut_vi.clock);
      tx = my_transaction::type_id::create("tx");
      tx.cmd  = dut_vi.cmd;
      tx.addr = dut_vi.addr;
      tx.data = dut_vi.data;

      aport.write(tx);
```

Sends tx through analysis port

Agent


Analysis port
Agent
Sequencer
Analysis port
Monitor
Driver

Agent

```
class my_agent extends ovm_agent;
  ...
  ovm_analysis_port #(my_transaction) aport;
```

Agent

```
class my_agent extends ovm_agent;
  ...
  ovm_analysis_port #(my_transaction) aport;
  ...
  function void build;
    super.build();
    aport = new("aport", this);
    my_sequencer_h = my_sequencer::type_id::create ...
    my_driver_h    = my_driver   ::type_id::create ...
    my_monitor_h   = my_monitor  ::type_id::create ...
```

Agent

```
class my_agent extends ovm_agent;
  ...
  ovm_analysis_port #(my_transaction) aport;
  ...
  function void build;
    super.build();
    aport = new("aport", this);
    my_sequencer_h = my_sequencer::type_id::create ...
    my_driver_h    = my_driver   ::type_id::create ...
    my_monitor_h   = my_monitor  ::type_id::create ...
  endfunction: build

  function void connect;
    ...
    my_monitor_h.aport.connect( aport );
```

Connecting a Subscriber

```
class my_env extends ovm_env;
  ...
  my_agent      my_agent_h;
  my_subscriber my_subscriber_h;
  ...
  function void build;
    super.build();
    my_agent_h      = my_agent     ::type_id::create ...
    my_subscriber_h = my_subscriber::type_id::create ...
  endfunction: build

  function void connect;
    my_agent_h.aport.connect(
                          my_subscriber_h.analysis_export );
  endfunction: connect
```

Subscriber

```
class my_subscriber extends ovm_subscriber
                            #(my_transaction);
  `ovm_component_utils(my_subscriber)



  ...



  function void write(my_transaction t);
```

Coverage and Checking

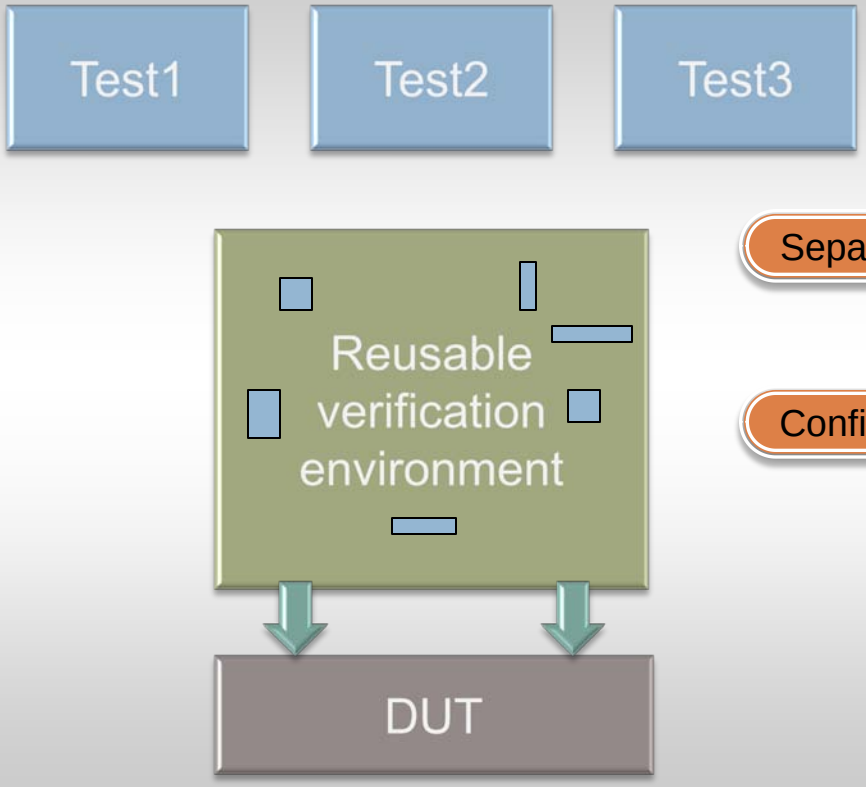
Coverage registers

```
bit cmd;
int addr;
int data;

covergroup cover_bus;
  coverpoint cmd;
  coverpoint addr { bins a[16] = {[0:255]}; }
  coverpoint data { bins d[16] = {[0:255]}; }
endgroup: cover_bus

function void write(my_transaction t);
  ...
  cmd  = t.cmd;
  addr = t.addr;
  data = t.data;
  cover_bus.sample();
```

Reporting

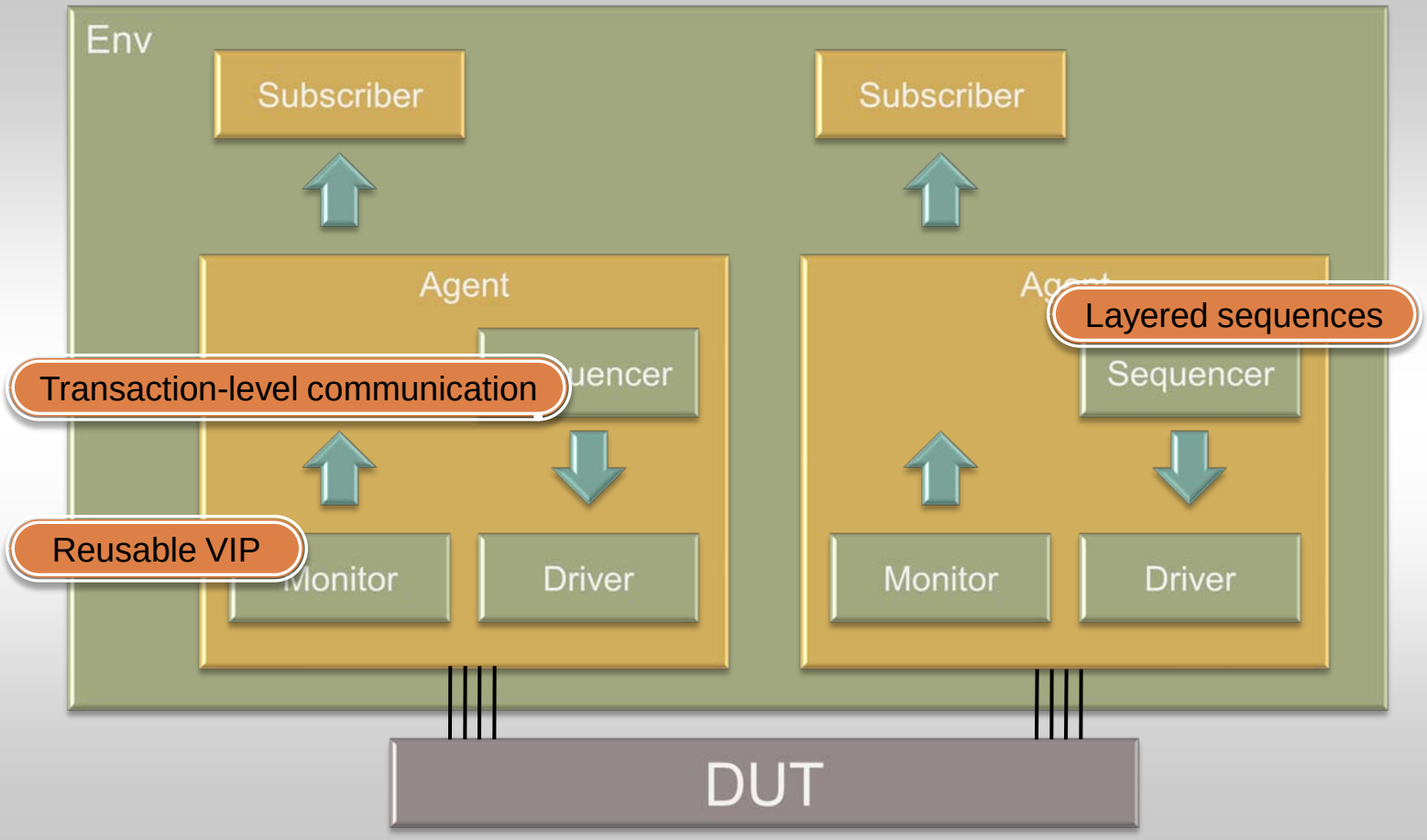
```
function void write(my_transaction t);
  ...

  ovm_report_info("ja", "Transaction received");
```

Summary 1



- Separating tests from test bench
- Configurable test bench

Summary 2


Env
Subscriber
Subscriber
Agent
Sequencer
Monitor
Driver
Monitor
Driver
DUT
Layered sequences
Transaction-level communication
Reusable VIP

VERIFICATION ACADEMY

Basic OVM

Monitors and Subscribers

John Aynsley
CTO, Doulos

academy@mentor.com
www.verificationacademy.com

Mentor Graphics®