

DVClub Bristol 22-April-2009

The Verification Methodology Landscape

Jonathan Bromley, Doulos



The Verification Methodology Landscape



CONTENTS

The M-word

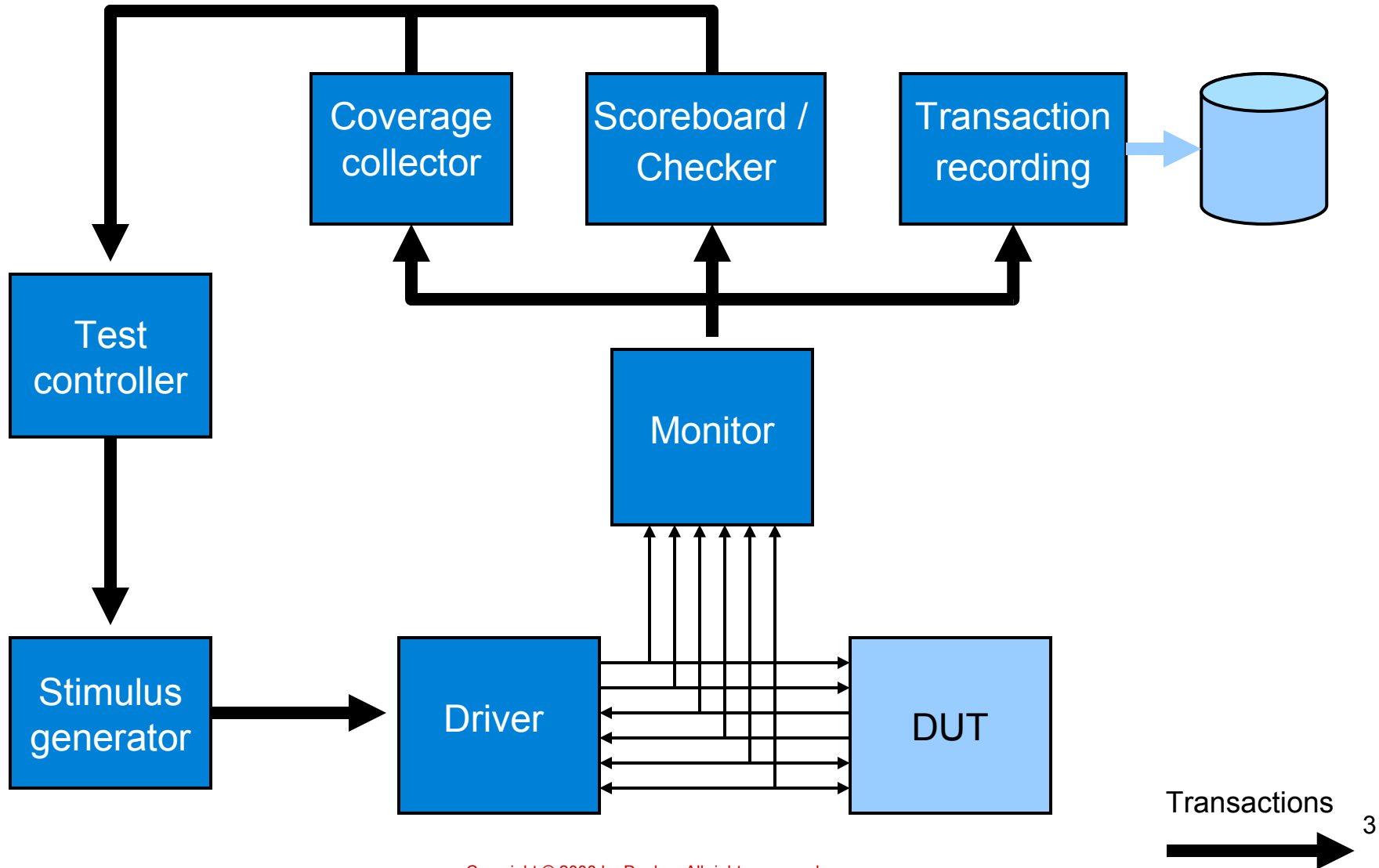
Languages, methodologies, tools and standards

The big players: OVM, VMM, eRM

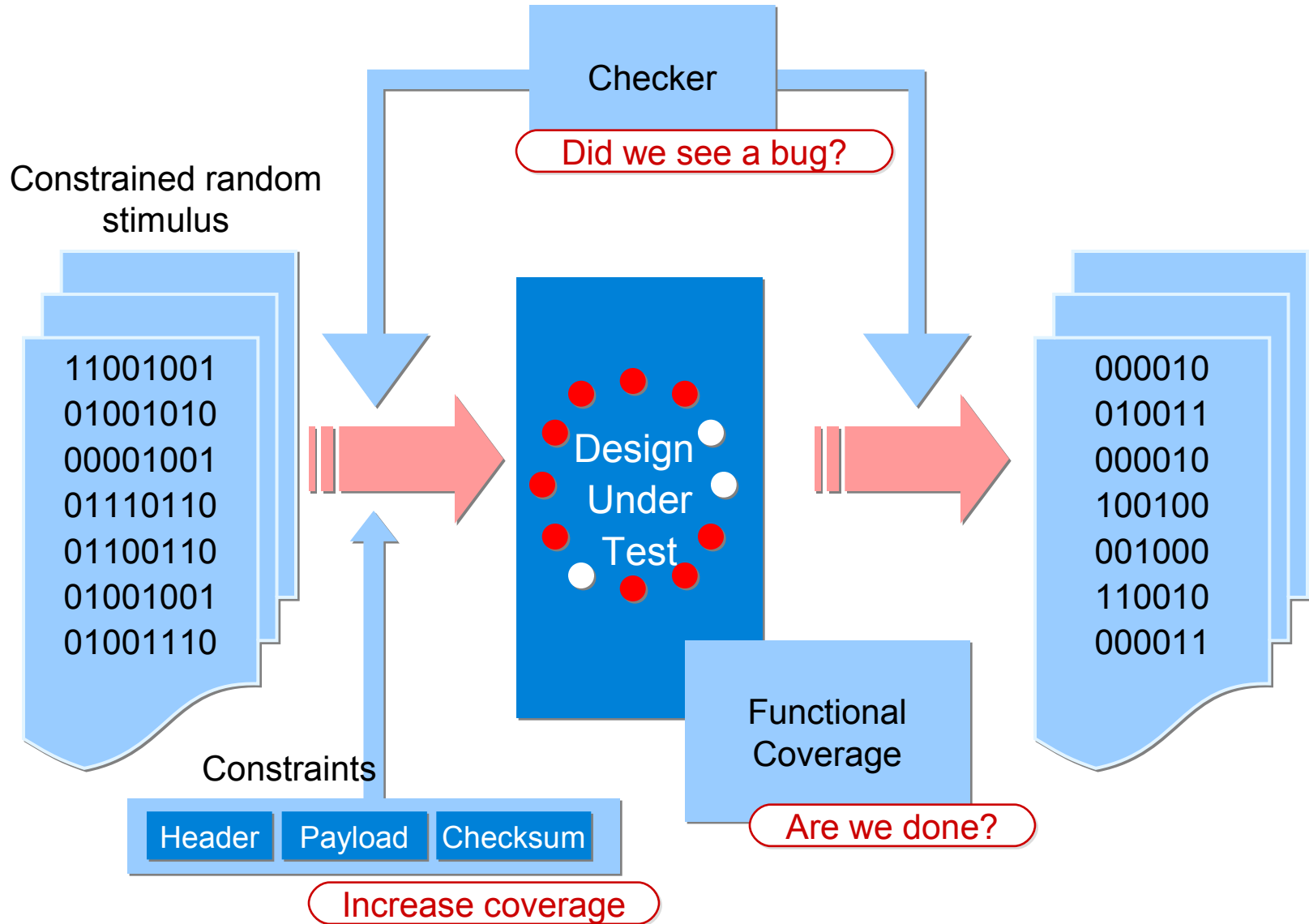
Interoperability and convergence

Conclusions?

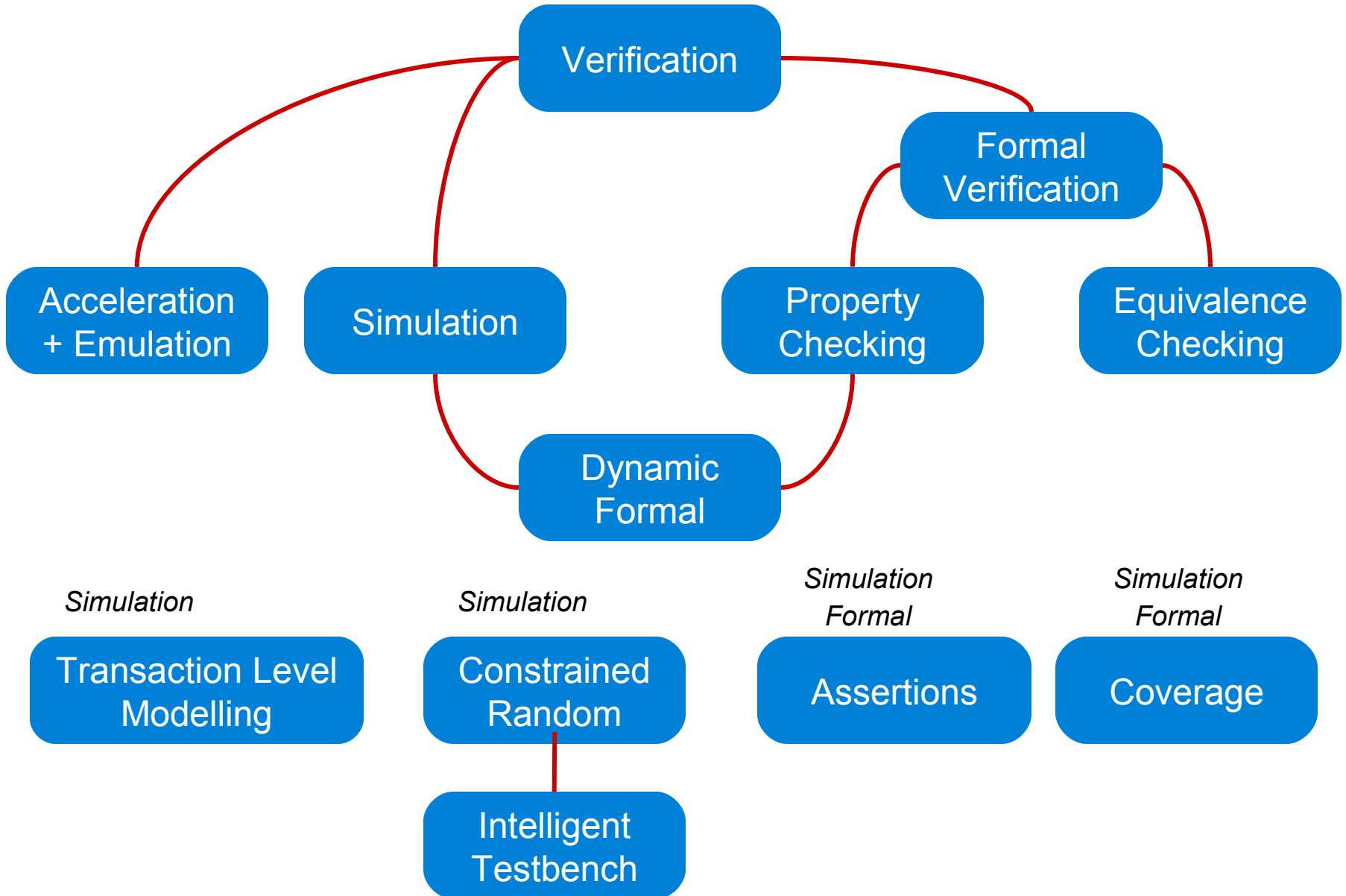
Verification Environment



Constrained Random Verification



The Verification Space



What Our Customers Want

- Ease of deployment
 - Customizable environment
 - but it must do something useful straight out of the box
 - Simple, uniform interface to any verification IP block
 - Gentle learning curve for the whole team
- Power
 - Complex testcases co-ordinated across the whole environment
 - Randomization
 - Sophisticated coverage analysis
- Interoperability
 - *Every customer we meet has legacy verification IP*

Methodology (or just a toolkit?)

- Tame the language monster
 - e, SystemVerilog, C++ are big and complicated
 - Many ways to solve a problem
 - Wheel reinvention is a hazard
- Toolkit
 - OOP encourages encapsulation of standard functionality in base classes
 - Ideal for component hierarchy, reporting, block-to-block communication, DUT connectivity
- Methodology
 - Published methodology encourages interoperability and re-use
 - Promote best practice

Mature Language Standards

List unchanged for 4 years...

- IEEE 1076 VHDL
- IEEE 1850™ PSL

- IEEE 1364 Verilog
- IEEE 1800™ SystemVerilog

- IEEE 1647™ e

- ISO/IEC 14882 C++
- IEEE 1666™ SystemC

- Tcl/Tk, Perl

Crude Caricature

FPGA, RTL, Europe, Mil-Aero

ASIC, RTL, USA/RoW

Hardware verification

Hardware verification

Modelling, verification

Virtual hardware prototypes for S/W dev

Scripting

New Standards Activity

- Verilog and SystemVerilog unified
 - LRM this year, currently in ballot feedback
 - Major enhancements to assertions
- Verification methodology
 - OVM 2.0, OVM-SC
 - VMM open-source
 - eRM3 - e / SV interoperability (Cadence)
 - OVM / VMM interoperability (Accellera)
- SystemC TLM-2.0

The Big Methodology Players



www.ovmworld.org



www.vmmcentral.org

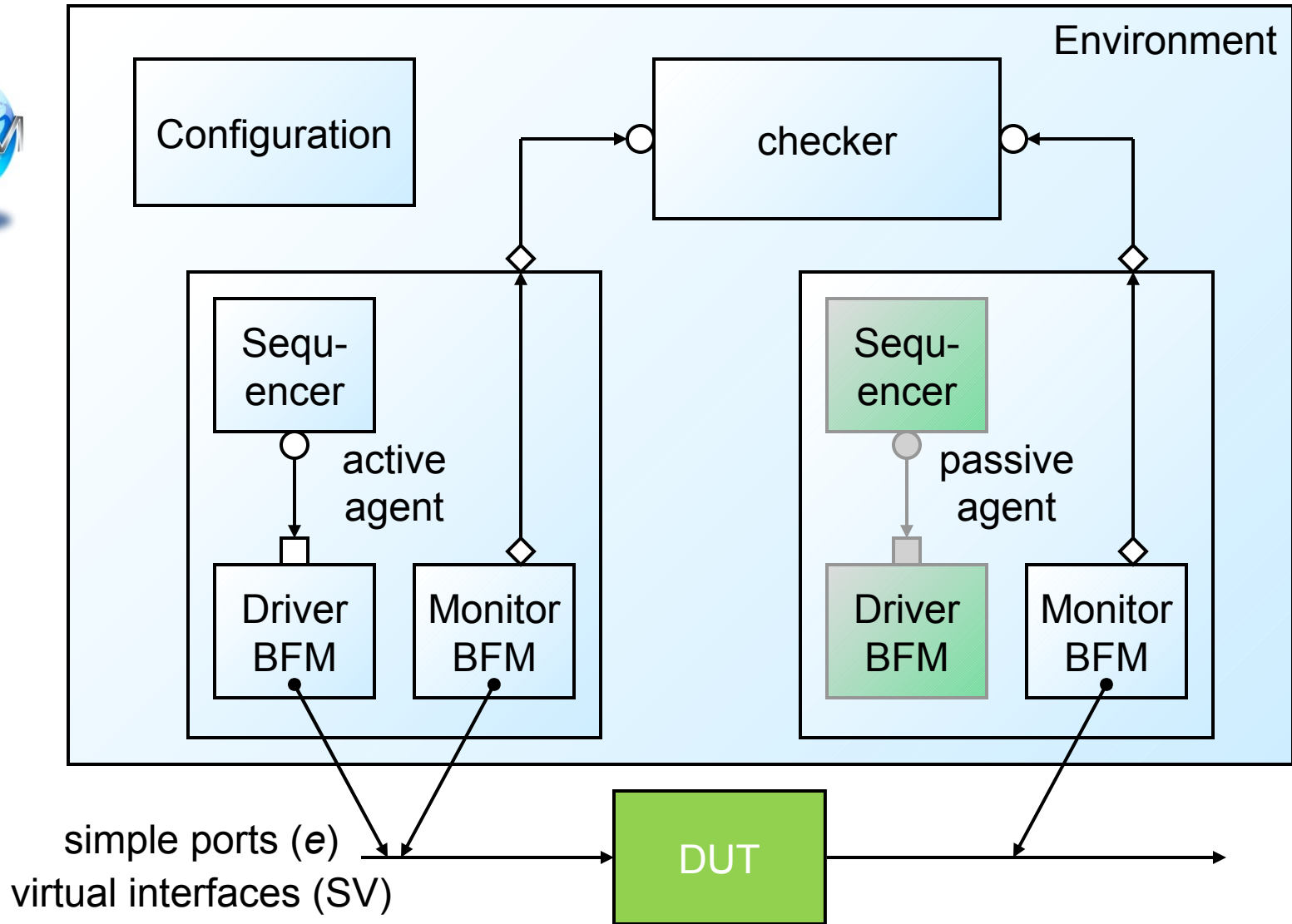
eRM

www.cadence.com

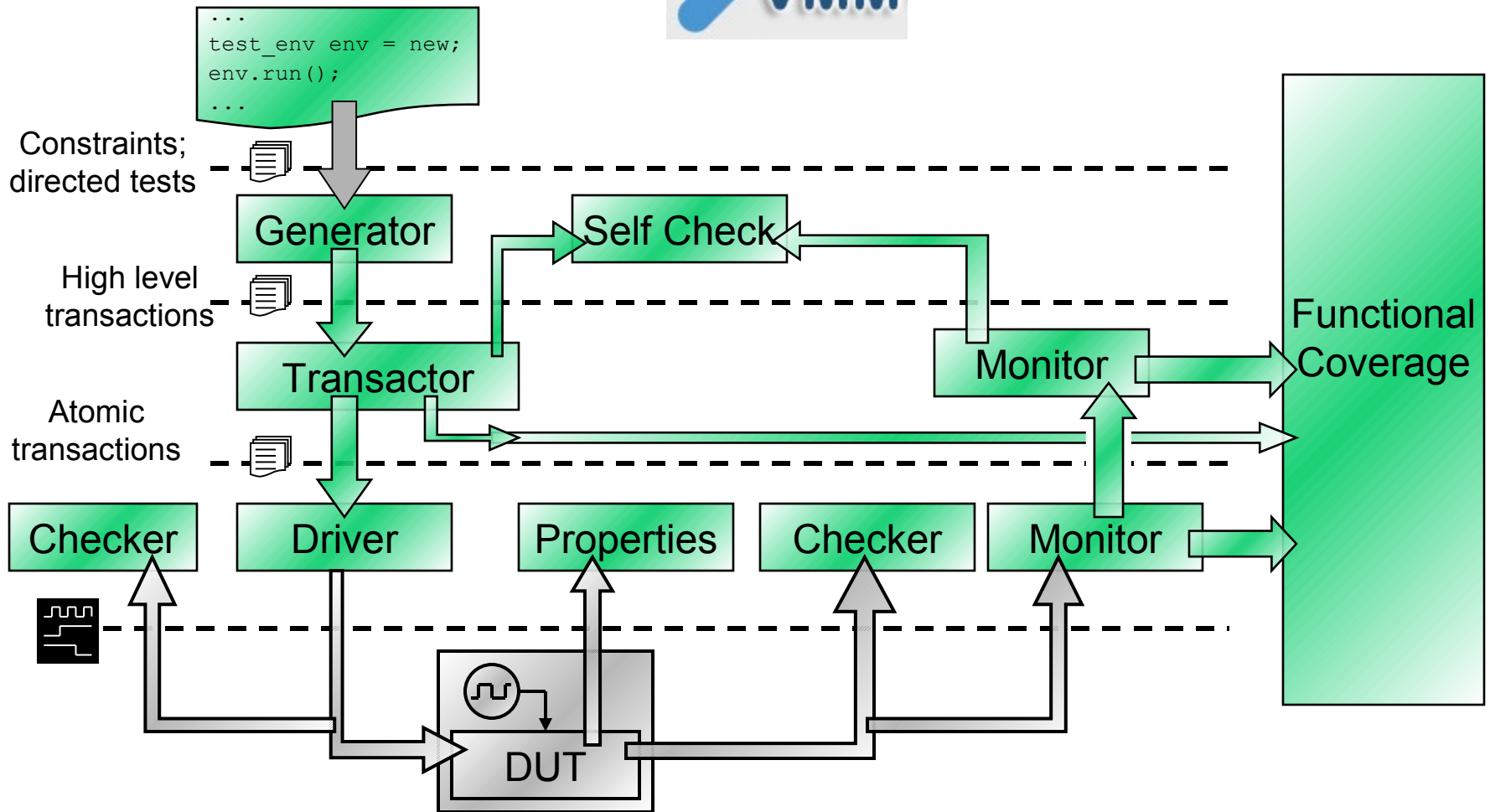
Static View of Testbench



eRM



Static View of Testbench (VMM)



OVM Key Features

- Constrained random, coverage-driven verification
- Separation of tests from verification environment
- Configuration of verification environment
 - through a table
- Verification IP reuse (canonical structure and guidelines)

- TLM communication
- Automation (where missing from SystemVerilog language)
- Hierarchical sequential stimulus (sequences)
- Standardized messaging

eRM Key Features

- Constrained random, coverage-driven verification
- Separation of tests from verification environment
- Configuration of verification environment
 - through AOP extension and pre-run constraints
- Verification IP reuse (rigorously standardized rules)

- Communication via ports
- Automation (using e language's macro features)
- Hierarchical sequential stimulus (sequences)
- Standardized messaging

VMM Key Features

- Constrained random, coverage-driven verification
- Configuration of verification environment
 - through configuration objects passed to verification components
- Verification IP reuse (conventions)
- Communication via channels, callbacks, notifications
- Automation (scripts and macros)
- Hierarchical sequential stimulus (scenarios)
- Standardized messaging
- Strongly influenced by RVM (Synopsys Vera)

Structure of an OVM Component

```
class my_driver extends ovm_driver #(my_transaction);
```

Base class

```
// ovm_seq_item_pull_port #(..) seq_item_port;
```

TLM port (inherited)

```
my_dut_if_wrapper m_dut_if;
```

Connection to DUT

```
function new(string name, ovm_component parent);  
    super.new(name, parent);  
endfunction: new
```

Constructor

```
function void build;  
    super.build();  
endfunction: build
```

Build phase
callback

```
virtual task run;  
    forever begin  
        ...  
    end  
endtask: run
```

Run phase
callback

```
endclass: my_driver
```

Phase Methods (OVM)

build

Call factory

connect

Make TLM connections

end_of_elaboration

After connections hardened

start_of_simulation

Get ready to run

run

Task (executed concurrently)

extract

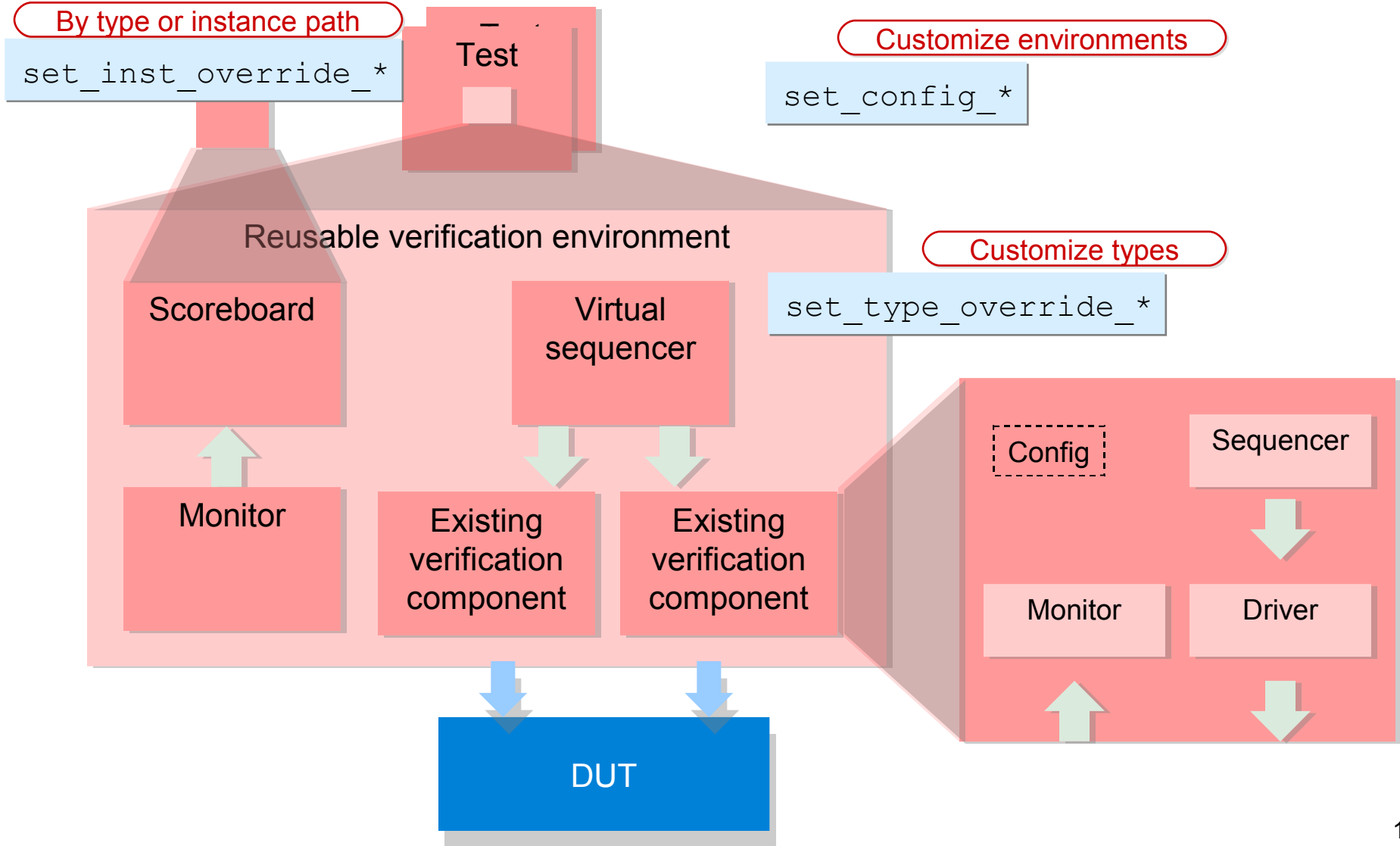
check

report

Post-processing

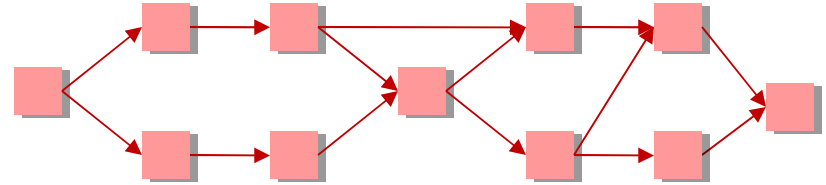
Similar phase arrangements in VMM, eRM

Reconfigurable Environment (OVM)

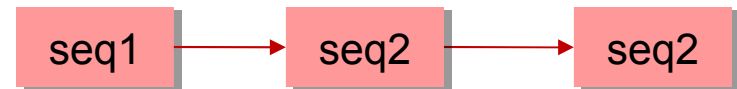


Layered Sequential Stimulus

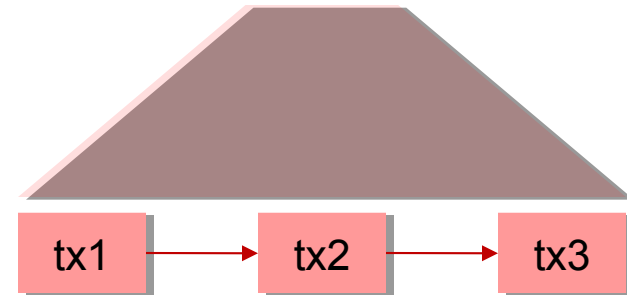
Tests enumerate possible top-level sequences



Virtual or layered sequences

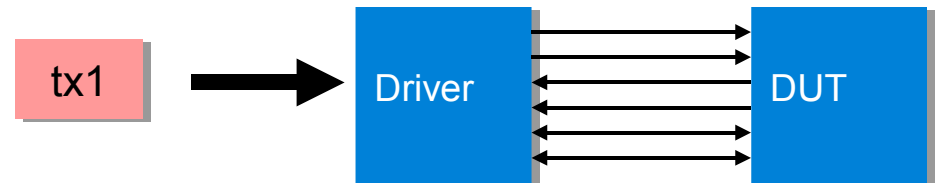


Constrained random sequence of transactions



Randomized transactions are not enough

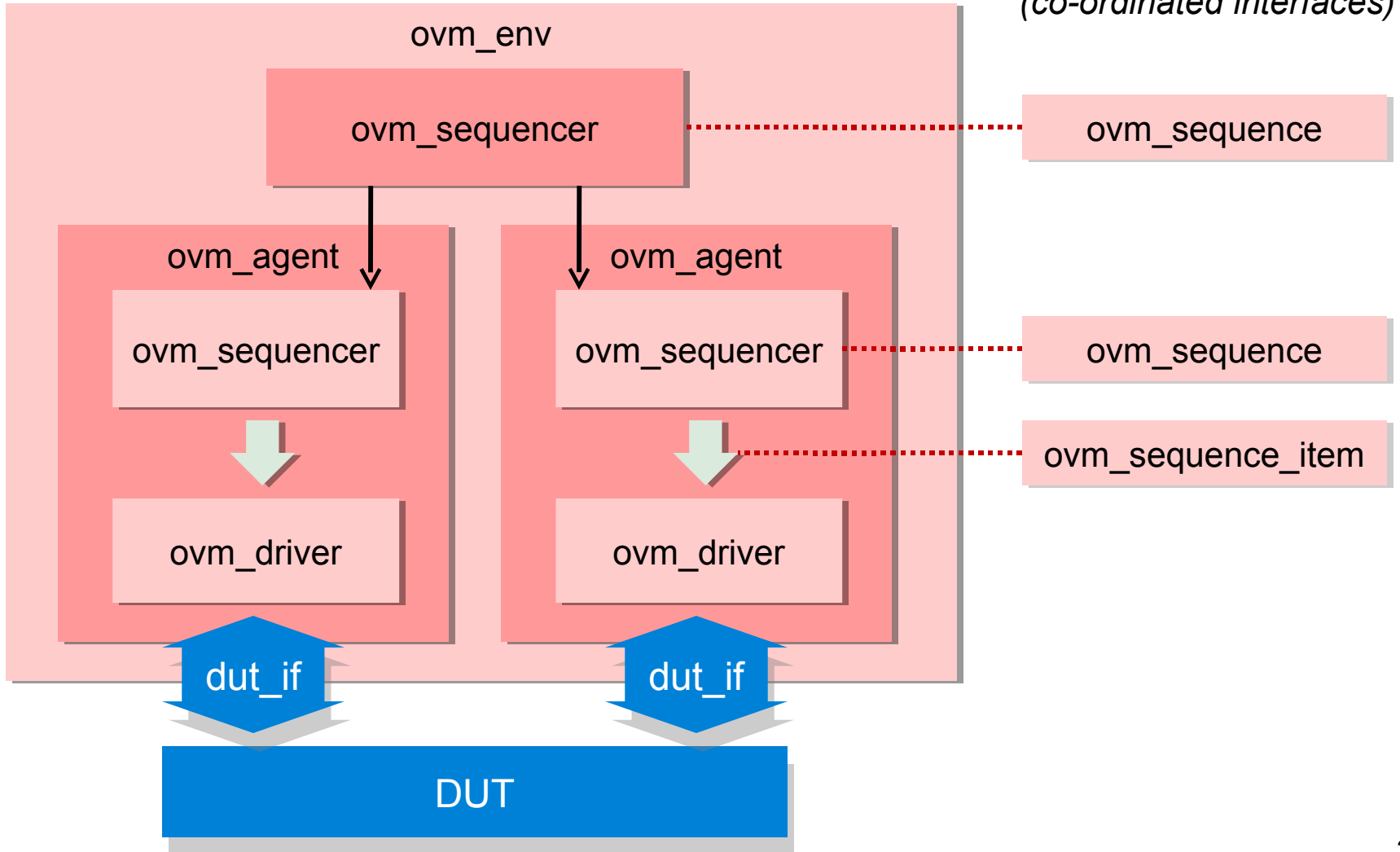
Drive transactions into DUT



Virtual Sequences

Component hierarchy

*Stimulus hierarchy
(co-ordinated interfaces)*



Scenario Generator (VMM)

Verification environment

Scenario generator

select_scenario

select

scenario_set

[0] atomic

[1] burst

[2] RMW

burst

items

copies of items

generator's output channel

Downstream transactor

Now and Next

- VMM: rapidly growing collection of "applications"
 - register abstraction layer, hierarchy, ...
- OVM/VMM interoperability toolkits/standards
- OVM/eRM mixed-language tools
- OVM-SC
- Increasing availability of verification IP

Conclusion

- Interesting times
 - standards don't always keep up with user needs
- Challenges for users choosing a new approach:
 - tools?
 - methodology?
 - decisions are not yet completely decoupled
- Training is important:
 - VMM, OVM, eRM are not difficult ...
 - ... but jump-starting your efforts pays dividends

System Design

SystemC

ARM • C++

Verification Methodology

e • **PSL • SCV**

SystemVerilog

Hardware Design

VHDL • Verilog

Altera • Xilinx

Perl • Tcl/Tk

