

Synopsys Interoperability Conference 2009

*Get a Grip!*

Get VMM enabled with Doulos

Doug Smith, Doulos



# Getting VMM enabled with Doulos

## CONTENTS

---

- ***Introducing Doulos***
- *What's "hip" with VMM 1.2*
- *VMM Offers from Doulos*





# Introducing Doulos

- Training specialists
- Focus on EDA language standards
- Established 18 years
- Trained over 17,000 engineers across 36 countries
- Offices in the US and Europe

## System Design

**SystemC**

**ARM • C++**

## Verification Methodology

*e* • **PSL • SCV**

**SystemVerilog**

## Hardware Design

**VHDL • Verilog**

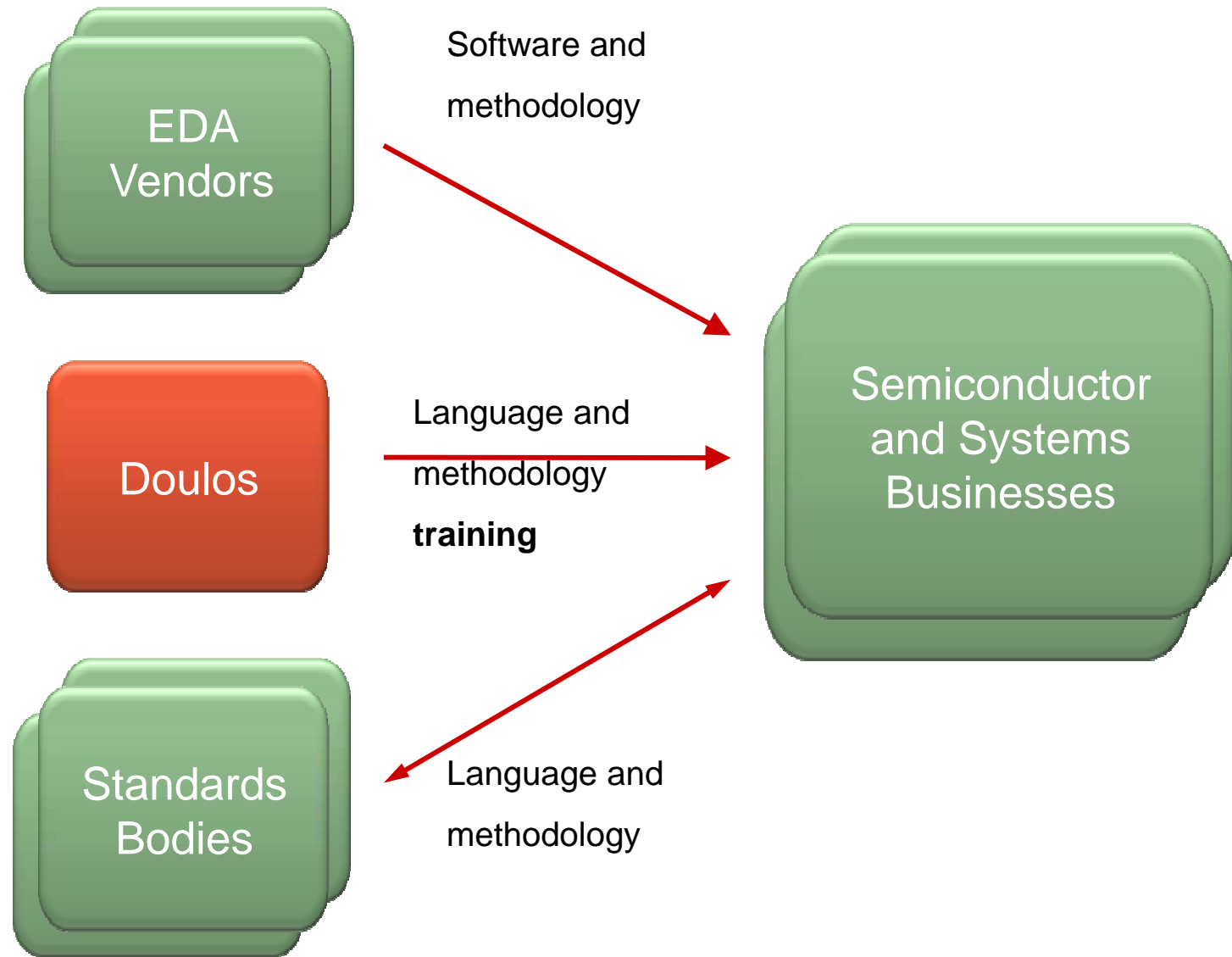
**Altera • Xilinx**

**Perl • Tcl/Tk**





# Doulos in the EcoSystem



# Getting VMM enabled with Doulos

## CONTENTS

---

- *Introducing Doulos*
- ***What's "hip" with VMM 1.2***
- *VMM Offers from Doulos*





# New “hip” features of VMM 1.2

- Path matching utility class
- Implicit phasing
- Automatic end-of-test consensus
- Multiple test execution
- Test timelines
- Class factory
- Hierarchical option specification
- RTL configuration
- Transaction Level Modeling (TLM 2.0)
- New structural components



# Implicit Phasing

Pre-Test Timeline	<code>rtl_config_ph</code>	Create or read RTL configuration
	<code>build_ph</code>	Build testbench
	<code>configure_ph</code>	Configure options
	<code>connect_ph</code>	Make TLM connections
Top-Test Timeline	<code>start_of_sim_ph</code>	Start of simulation
	<code>reset_ph</code>	Reset the design
	<code>training_ph</code>	Physical interface training
	<code>config_dut_ph</code>	Configure the design
	<code>start_ph</code>	Logical start of test
	<code>start_of_test_ph</code>	Physical start of test
	<code>run_test_ph</code>	Body of test
	<code>shutdown_ph</code>	Stop stimulus
	<code>cleanup_ph</code>	Read final design state
Post-Test Timeline	<code>report_ph</code>	Report pass or fail
	<code>final_ph</code>	Final actions



# Phasing example

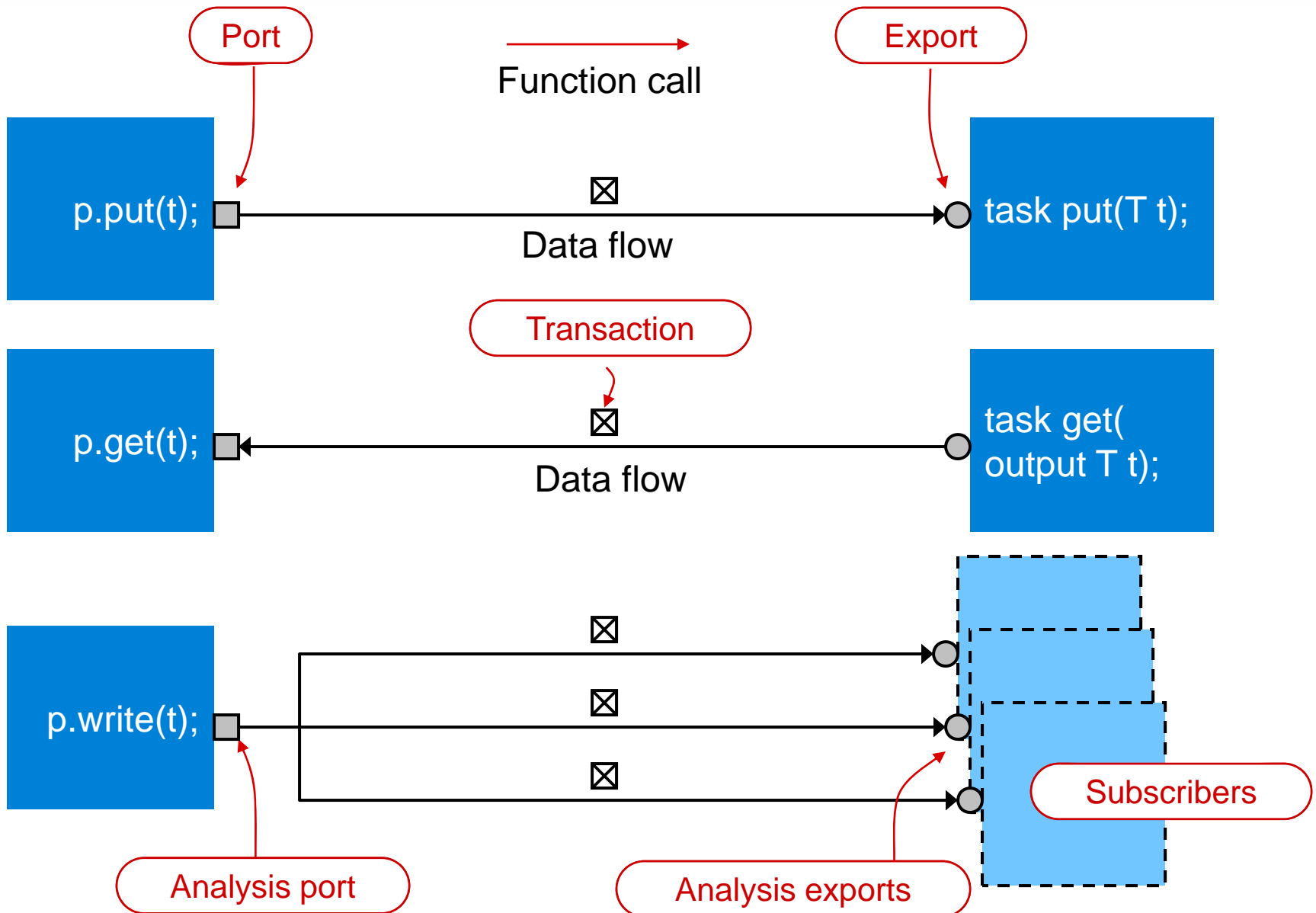
```
class my_env extends vmm_group;
  virtual dut_intf      dut_if;
  ...
  function void build_ph();
    m_sub  = new( "m_sub", dut_if );
    m_mon  = new( this, "m_mon", dut_if );
    m_sb   = new( "m_sb" );
  endfunction : build_ph
  function void connect_ph();
    m_sub.wb_mon.sb_ap.tlm_bind( m_sb.exp_ap );
    m_mon.sb_ap.tlm_bind( m_sb.inp_ap );
  endfunction : connect_ph
  task reset_ph();
    dut_if.wb_adr_i = {AWIDTH{1'bx}}; // Initialize
    ...
    dut_if.wb_rst_i = 0; // Reset
    #20;
  endtask: reset_ph
endclass : my_env
```

Build

TLM Connections

Reset DUT

# Transaction Level Modeling





# TLM example

```
class my_sbd extends vmm_sb_ds_typed#( my_trans );
  `vmm_tlm_analysis_export( _inp )
  `vmm_tlm_analysis_export( _exp )

vmm_tlm_analysis_export_inp#(my_sbd, my_trans) inp_ap = new(...);
vmm_tlm_analysis_export_exp#(my_sbd, my_trans) exp_ap = new(...);

// Provide an implementation for the TLM analysis ports
function void write_inp( int id = -1, my_trans trans );
  // Insert the transaction into the queue
  this.inp_insert( trans );
endfunction

function void write_exp( int id = -1, my_trans trans );
  // Expect this transaction
  this.expect_in_order( trans );
endfunction

endclass : my_sbd
```

Declare analysis exports

Define export implementation

# Class factory

- Factory creates the requested object
- Factory objects can be overridden
- Replace any object, transaction, scenario, or transactor
- Test cases can use factory to replace components and objects
- Each class can have its own factory
- Simple macro to implement factory





# Class factory example

```
class my_trans#( type T = int ) extends vmm_data;  
  `vmm_data_member_begin( my_trans#(T) )  
  ...  
  `vmm_data_member_end( my_trans#(T) )  
  
  `vmm_class_factory( my_trans#(T) )  
endclass
```

Creates new(), copy(),  
and allocate()

Creates class factory

```
class my_gen #( type T = int ) extends vmm_xactor;  
  function new( string inst, vmm_unit parent = null );  
    super.new( "my_gen", inst, , parent );  
    tr = my_trans#(T)::create_instance( this, "MY TRANS");  
  endfunction  
  ...  
endclass
```

Factory creates object

```
class test extends vmm_test;  
  function void start_of_sim_ph();  
    // Replace factory in env0.gen  
    my_trans::override_with_new( "@env0:gen:randomized obj",  
my_trans::this_type, log, `__FILE__, `__LINE__);  
  endfunction  
endclass
```

Replace class factory

# Getting VMM enabled with Doulos

## CONTENTS

---

- *Introducing Doulos*
- *What's "hip" with VMM 1.2*
- ***VMM Offers from Doulos***





# Unique Training Offer

- Tutors are subject-matter experts
- Teach project-ready skills
- Work closely with EDA and technology vendors
- Customized classes and curricula to meet **YOUR** needs



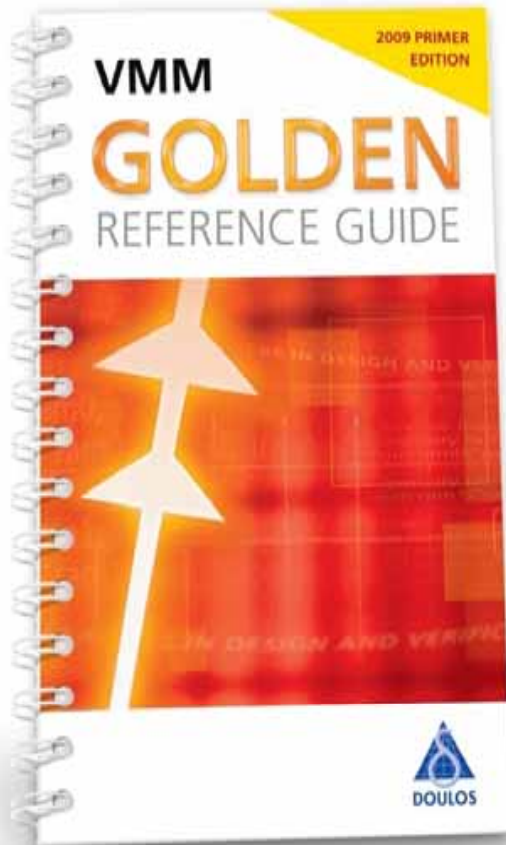
# Professional Training Classes



- VMM Adopter Class
- SystemVerilog training customized for VMM
  - See *SystemVerilog Out of the Box* – [www.doulos.com/systemverilog](http://www.doulos.com/systemverilog)
- Public open-enrollment or private on-site classes



# VMM Golden Reference Guide



- Reference guide to VMM
- Focuses on the most important issues
- Practical hints-and-tips

*Get your free copy here*





## For More Information

- For more information on Doulos training  
call **1-888-GO-DOULOS**  
or visit **[www.doulos.com](http://www.doulos.com)**

System Design

**SystemC**

**ARM • C++**

Verification Methodology

*e* • **PSL • SCV**

**SystemVerilog**

Hardware Design

**VHDL • Verilog**

**Altera • Xilinx**

**Perl • Tcl/Tk**

