



New VMM Standard Library Features

Jason Sprott, CTO

DAC 2009, VMM Interoperability Breakfast



Agenda

- Introduction
- SystemC TLM 2.0 Support (NEW)
- Simulation Phases & Timelines (NEW)
- Test Concatenation (NEW)
- Class Factory (NEW)
- Parameterized VMM Generators (NEW)

About Verilab

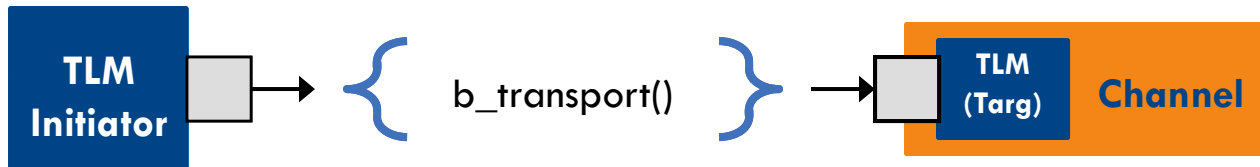
- Engineering Consultancy Specializing in Functional Verification
- Over 150 verification projects since 2000
- Worked with Synopsys on VMM 1.1 & 1.2
- Offices UK, Germany, USA

SystemC TLM 2.0 Support in VMM

- Allows for easier integration with TLM-based models
- VMM Implementation of TLM 2.0
 - TLM Core Interfaces (blocking & non-blocking)
 - Annotated timing
 - TLM Analysis Port
 - TLM Sockets
 - TLM Generic Payload
 - VMM Channel now has built-in TLM support
 - New classes and support Macros added

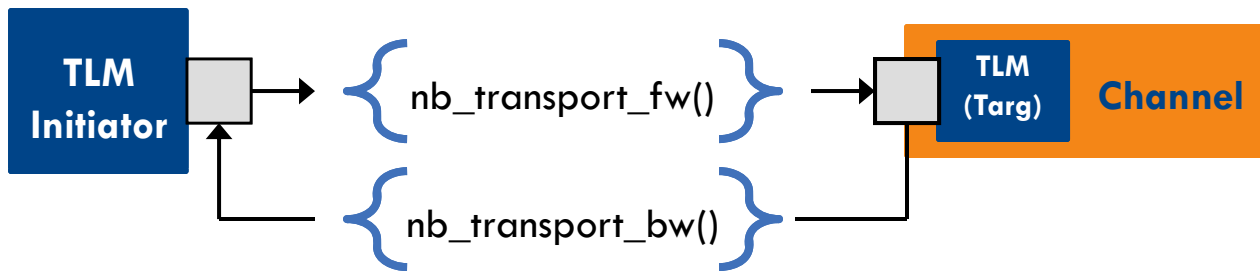
VMM Channel as TLM Target

Blocking Transport (Core Interface)

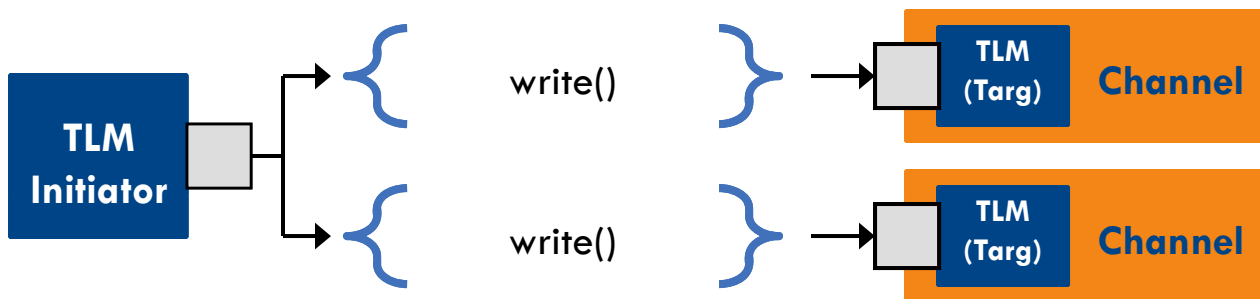


```
in_chan.tlm_bind(initiator.port, vmm_tlm::TLM_BLOCKING_EXPORT);
```

Non-blocking Transport (Core Interface)

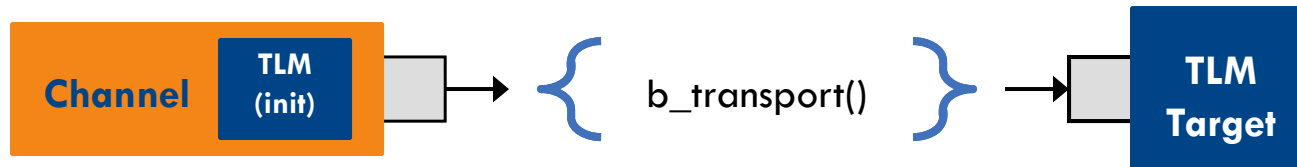


Analysis Port (Broadcast)

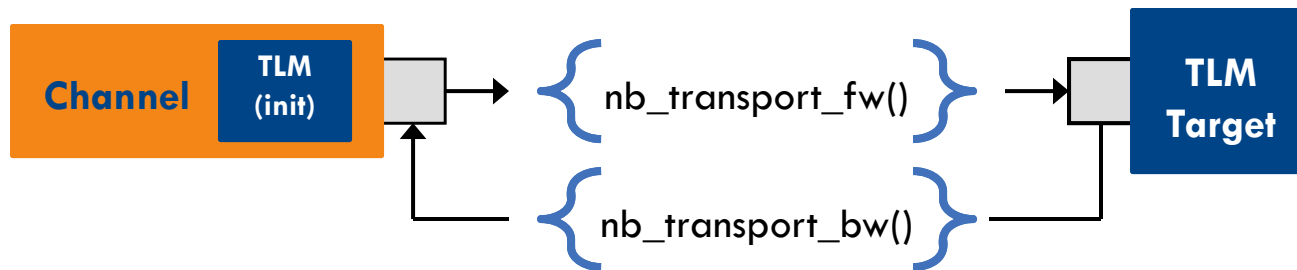


VMM Channel as TLM Initiator

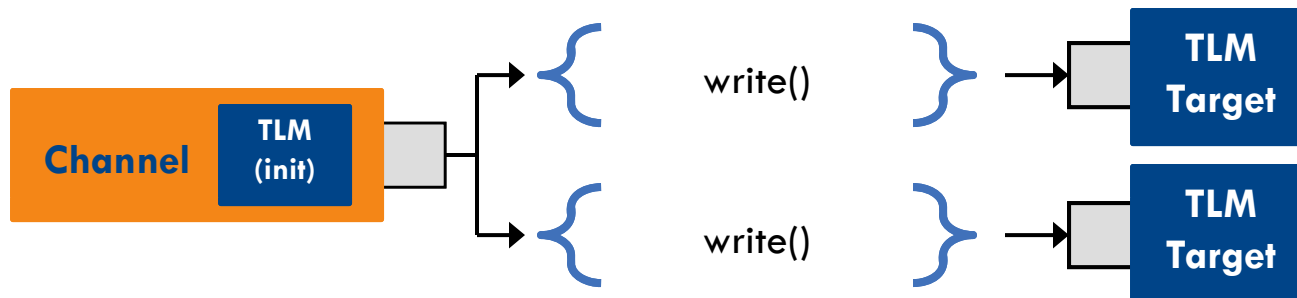
Blocking Transport (Core Interface)



Non-blocking Transport (Core Interface)



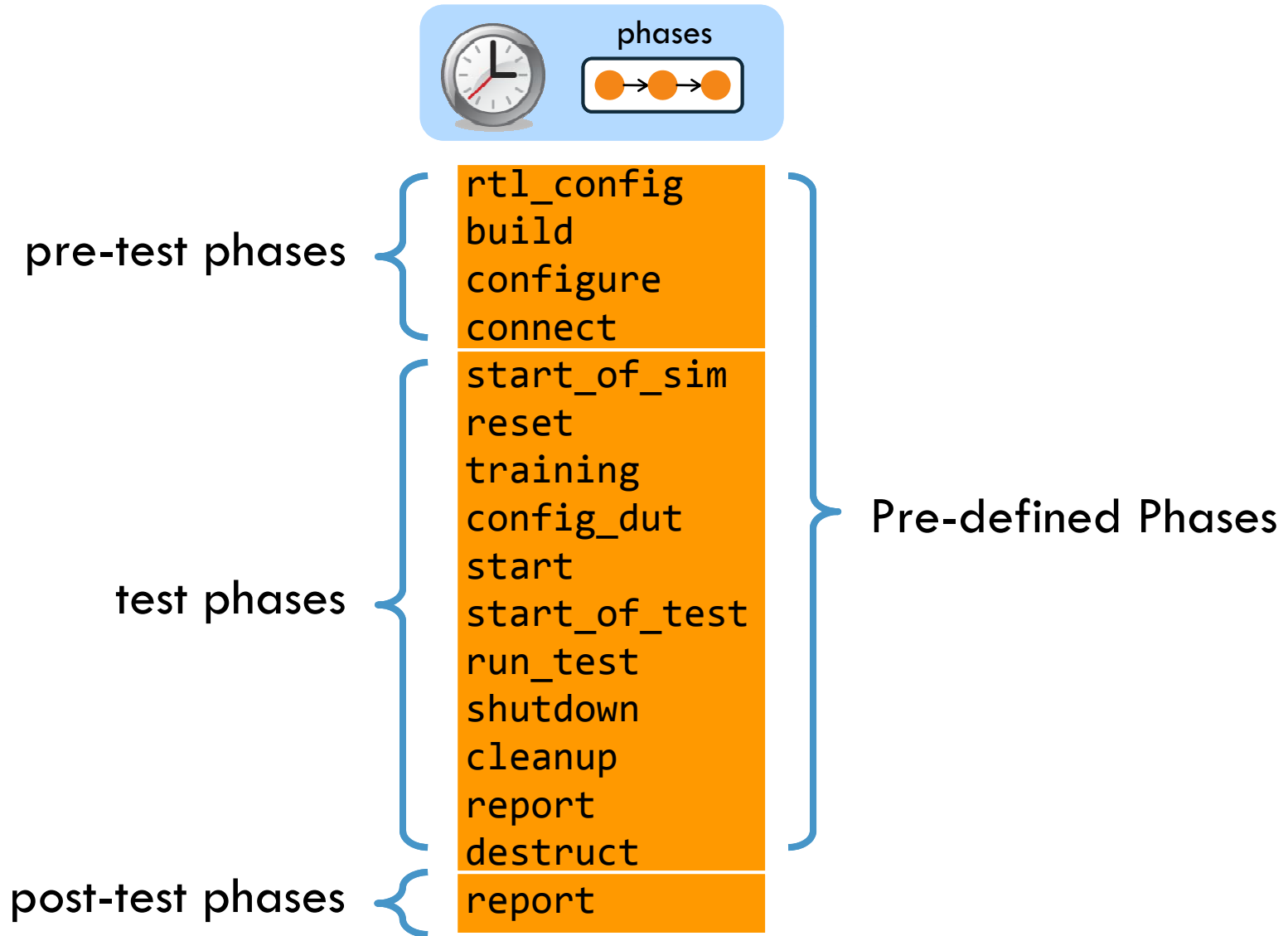
Analysis Port (Broadcast)



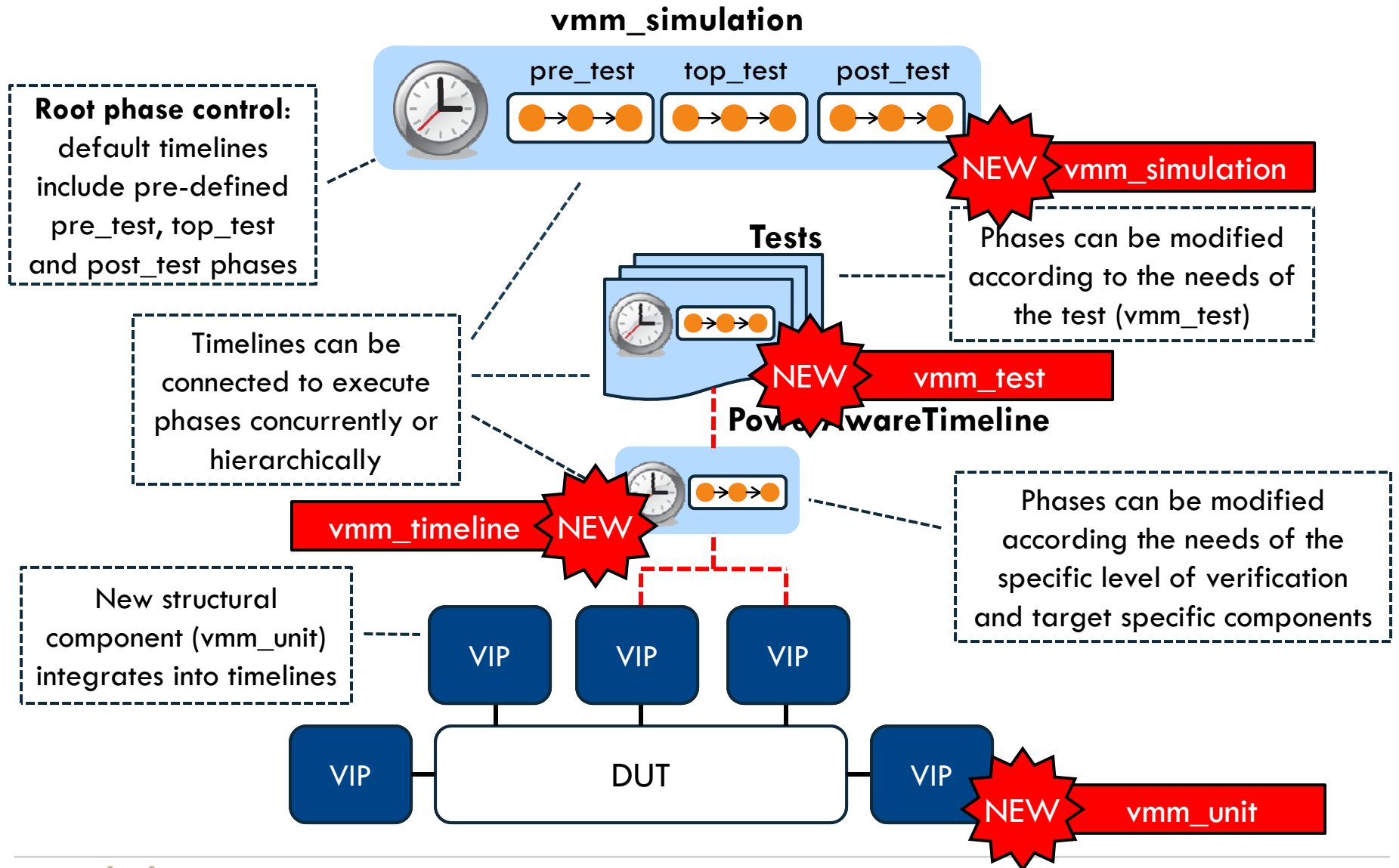
New Phases & Timelines

- Greater flexibility and extensibility
 - Accommodate foreign VIP requirements
 - Handle changing requirements, e.g. Block to Chip, Low Power
 - Roll-back of phases, e.g. test concatenation
- Encapsulate phases in a “timeline”
- Easy to add, delete, override phases
- Control components individually or as a group

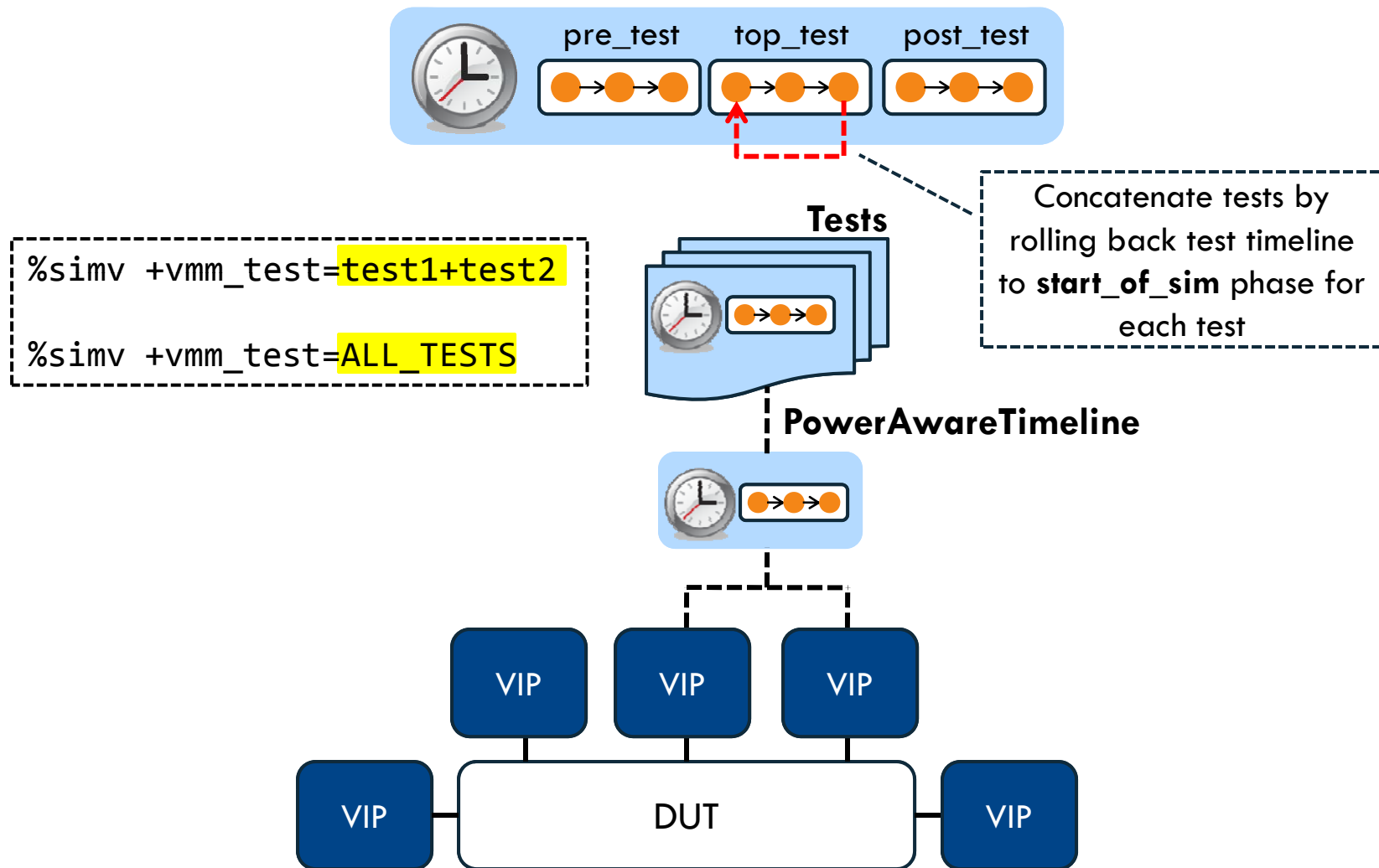
Timelines – Pre-defined Phases



Controlling Phases With Timelines



Test Concatenation



Class Factory

- Enables re-use of existing environments
- Easily replace any object in a testbench with a derived or compatible type
- ``vmm_class_factory` macro makes it easy
- Instantiating objects using the factory

```
Foo my_foo = new( ... );
```

```
c = Foo::create_instance( ... );
```

- We can tell the factory to generate something different without changing any architecture

```
Foo::override_with_new(...); // or
```

```
Foo::override_with_copy(...);
```

Using Parameterized Generators

VMM 1.1 implementation with macros

```
`vmm_channel(my_trans)
`vmm_atomic_gen(my_trans, "my_trans Atomic Generator")

my_trans_channel      ch = new(...); // type created by macro
my_trans_atomic_gen  gen0 = new(...); // type created by macro

my_callback0 extends my_trans_atomic_gen_callbacks;
endclass
```

VMM 1.2 implementation with NEW parameterized classes

```
vmm_channel_typed #(my_trans) ch = new(...); // from VMM 1.1
vmm_atomic_gen #(my_trans) gen0 = new(...); // NEW

my_callback0 extends vmm_atomic_gen_callbacks #(my_trans);
endclass
```

Summary

- TLM 2.0 in VMM Channels
- More control of simulation phases using timelines
- Phase rollback makes test concatenation possible
- Class Factory makes code more reusable
- Parameterized generators instead of macros

And there's lots more ...

Jason Sprott, CTO

www.verilab.com