

Using Programmer's View Timing Annotation for the Creation of Reusable TLM Models

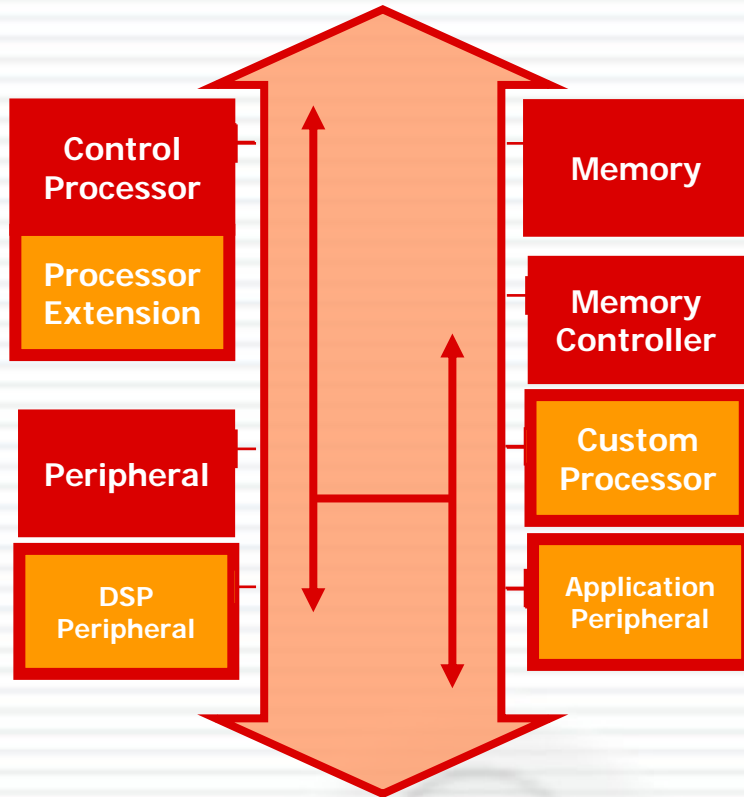
Charu Khosla
CoWare, Inc.

2/21/07

Outline

- Motivation
- How to Use Programmer's View Timing Annotation to Create Reusable TLM Models
- Case Study: Memory Controller

Platform Modeling using TLM



- ✓ Model the architecture and validate
- ✓ Model the functionality visible to SW for:
 - SW application development
 - SW performance analysis
- ✓ Model HW for optimization & verification

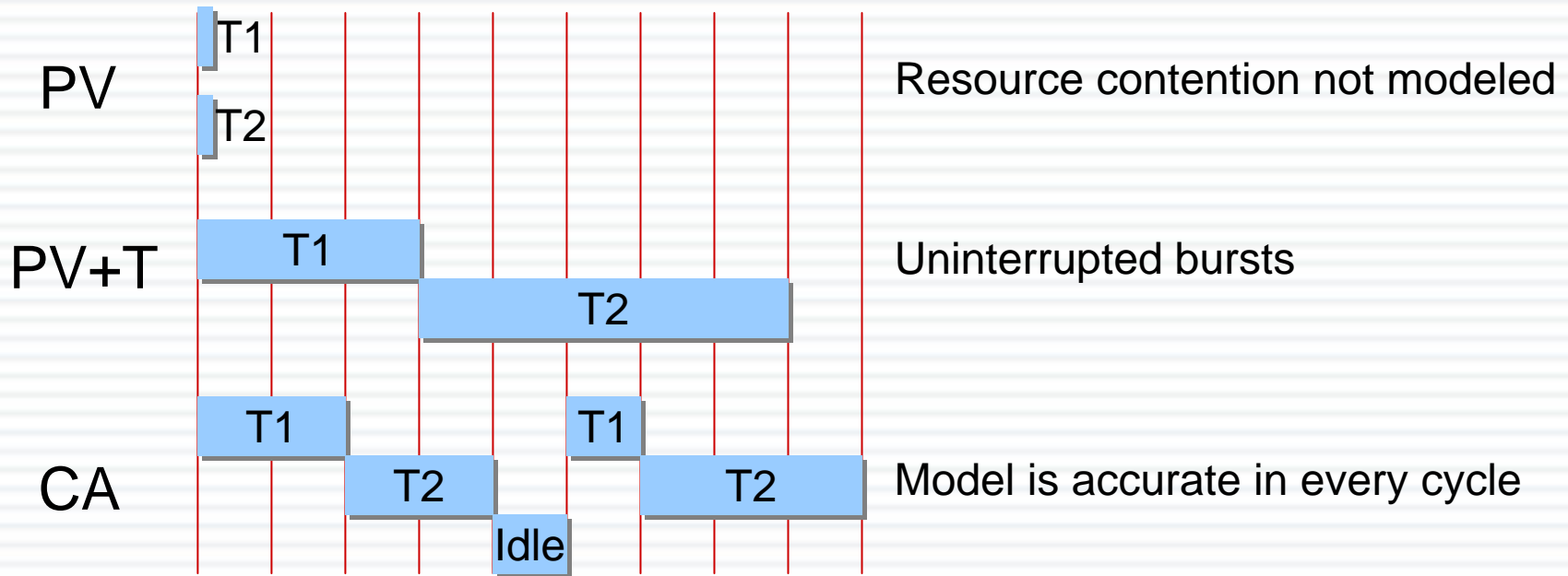


Standards-based transaction-level modeling and reuse

Modeling Requirements at TLM

TLM Use-model		Accuracy	Platform speed
Programmer's View (PV)	SW Application Development	Un-timed Register-accurate	50-200 MCPS
Programmer's View with Timing (PV+T)	SW Performance Analysis	> 90% Interval: 10-100 Kcycles	5 – 50 MCPS
	Architecture/Bus Exploration	> 90% Interval: 1-10 cycles	0.5 – 5 MCPS
Cycle Accurate (CA)	Verification	100%	0.1 - 0.5 MCPS

PV versus PV+T versus CA



PV: Un-Timed

Enables Application SW development

PV+T: Cycle Approximate

Enables SW performance analysis and Architecture/bus exploration

CA: Cycle Accurate

Enables HW-SW co-verification

*Is reuse possible
across design
tasks?* **YES**

SystemC Model Reuse Example

- Platform originally modeled at PV for Application SW Development
- Updated platform reused TLM peripheral models with PV+T timing annotation, enabling SW Performance Analysis of memory sub-system



3G Mobile Virtual Platform

- 55 unique models (95 instances)
 - 4 models within memory sub-system enabled with PV+T
- Runs the actual, unmodified software for the phone
 - PV+T enables SW performance analysis during OS boot, network registration, ...

SystemC Model Reuse Example

- Platform originally modeled at PV for Application SW Development
- Updated platform reused TLM peripheral models with PV+T timing annotation, enabling SW Performance Analysis of memory sub-system



3G Mobile Virtual Platform

	CoWare VP (w/ PV+T)	CoWare VP (at PV)	Real- Time
Phone OS Booted	31 sec	20 sec	2 sec
GSM Network Registration	476 sec	66 sec	8 sec
Idle execution	3.5x	3.5x	1x

Outline

- Motivation
- How to Use Programmer's View Timing Annotation to Create Reusable TLM Models
- Case Study: Memory Controller

Modeling Timing

Re-use goal:

- Speed is fast enough for Software Development use-case
- Timing is accurate enough for architecture/verification use-case

Strategies:

- “RTL style”: clocked threads/methods
 - Slow, but most accurate
 - Use “model gating”, i.e. disable clock when model is not active
 - Examples: synchronization of multiple ports, internal fifos
- “TLM style”: delayed event notification
 - Still relatively slow
 - Examples: Timer, DMA, ...
- “Timing Annotation”
 - Fast, but not always applicable
 - Example: Memory subsystem (MMU, cache, memory controller)

*The focus of
this paper*

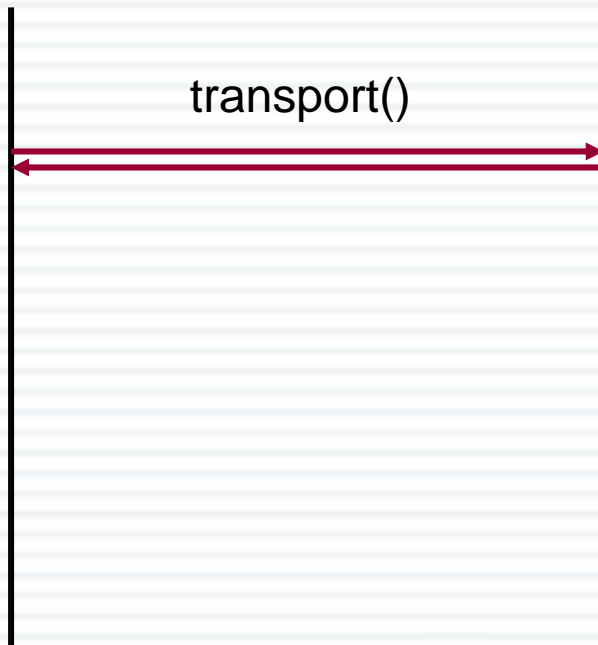
TLM2 Standard Interfaces

	Untimed	Timed
Bidirectional Blocking	<pre>void transport(const REQ&, RSP&)</pre>	<pre>void transport(const REQ &, RSP &, sc_time &)</pre>
Unidirectional Blocking	<pre>void put(const T&) void get(T&) void peek(T&)</pre>	
Unidirectional Non-Blocking	<pre>bool nb_put(const T&) bool nb_can_put() sc_event &ok_to_put() bool nb_get(T&) bool nb_can_get() sc_event &ok_to_get() bool nb_peek(T&) ... </pre>	<pre>bool nb_put(const T&, sc_time &) bool nb_can_put(sc_time &) bool nb_get(T&, sc_time &) bool nb_can_get(sc_time &) </pre>

Bidirectional Interface with Timing Annotation

PV Initiator

PV Target



PV target model

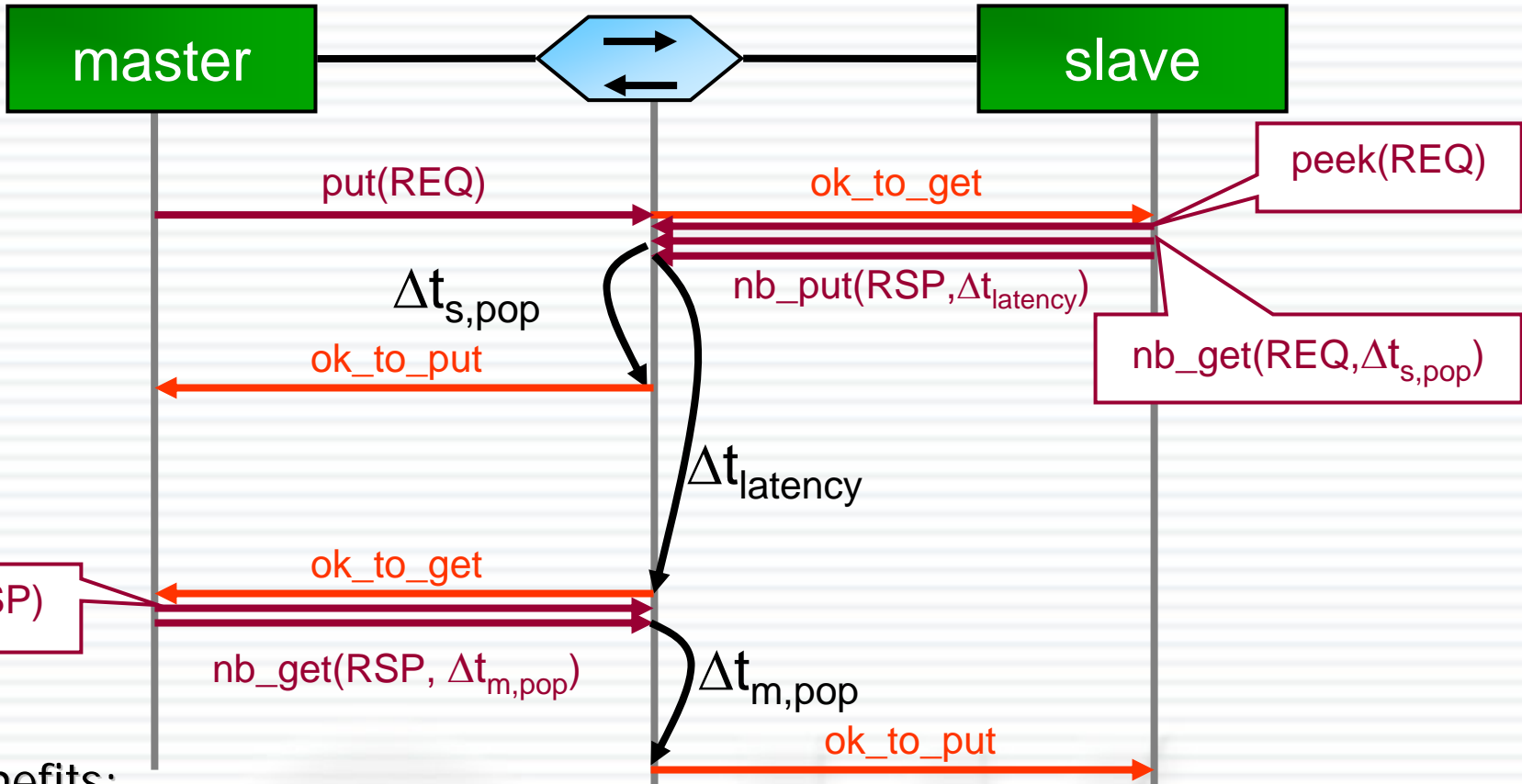
```
void transport(const REQ& rq, RSP& rp
               sc_time& latency)
{
    // do processing ...

    double lat = ...;
    latency = lat * clk_period;

    rp.get_status().set_ok();
}
```

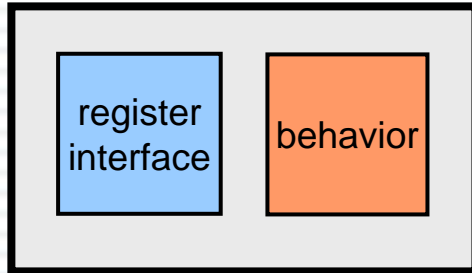
- Benefits:
 - Preserve potential for high simulation speed (not calling wait)
 - Defers realization of timing to initiator

Unidirectional Interfaces with Timing Annotation

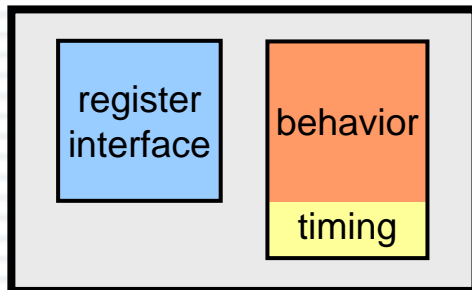
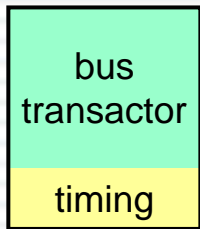


- Benefits:
 - Fine grained Timing Annotation
 - Both Accept Latency and Response Latency annotated

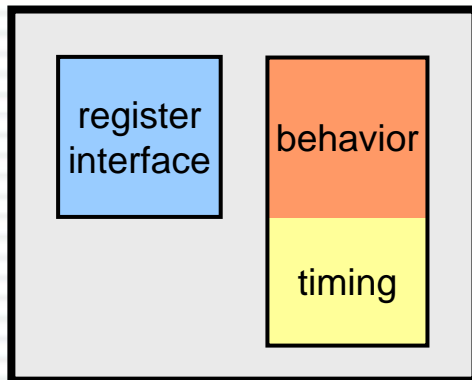
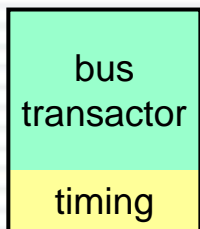
Modeling for Re-use



PV Peripheral = behavior
+ register accurate



PV+T Peripheral = PV Peripheral
+ approximate timing
+ optional PV+T bus transactor



VV Peripheral = PV Peripheral
+ cycle-accurate timing
+ cycle-accurate bus transactor

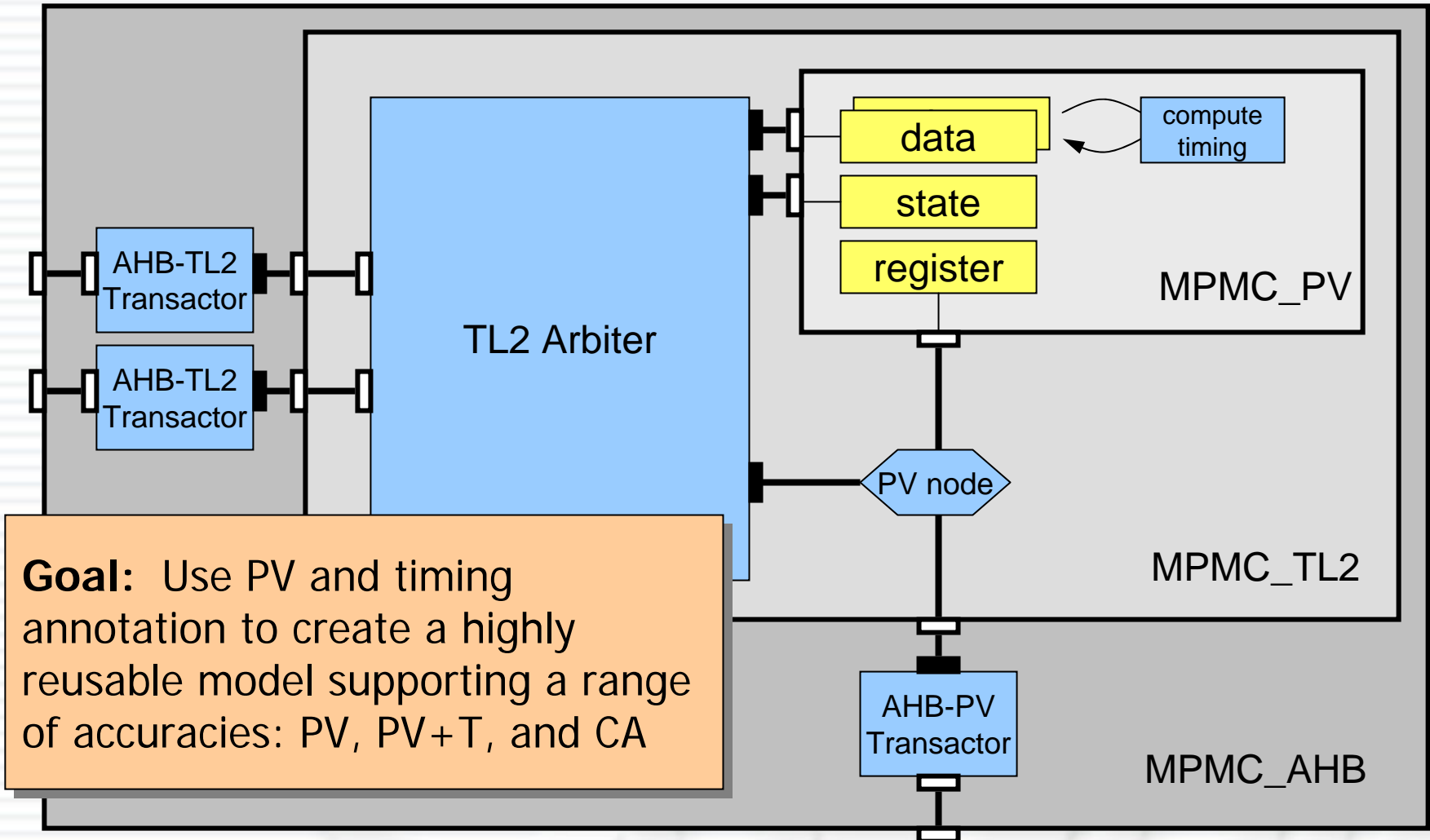
Ways to Compute Delay

- Static
 - Constructor argument or model property (fixed timing at the construction)
 - Low accuracy, only suitable for simplistic timing models
 - Example: register read/write, fixed latency blocks
- Stochastic
 - Configurable distribution and moments
 - Medium accuracy, suitable for high level architecture exploration
 - Example: Cache miss probability
- Dynamic
 - Delay is computed for every individual transaction, based on transaction attributes and state
 - Highest accuracy
 - Highest modeling effort
 - Example: memory controller

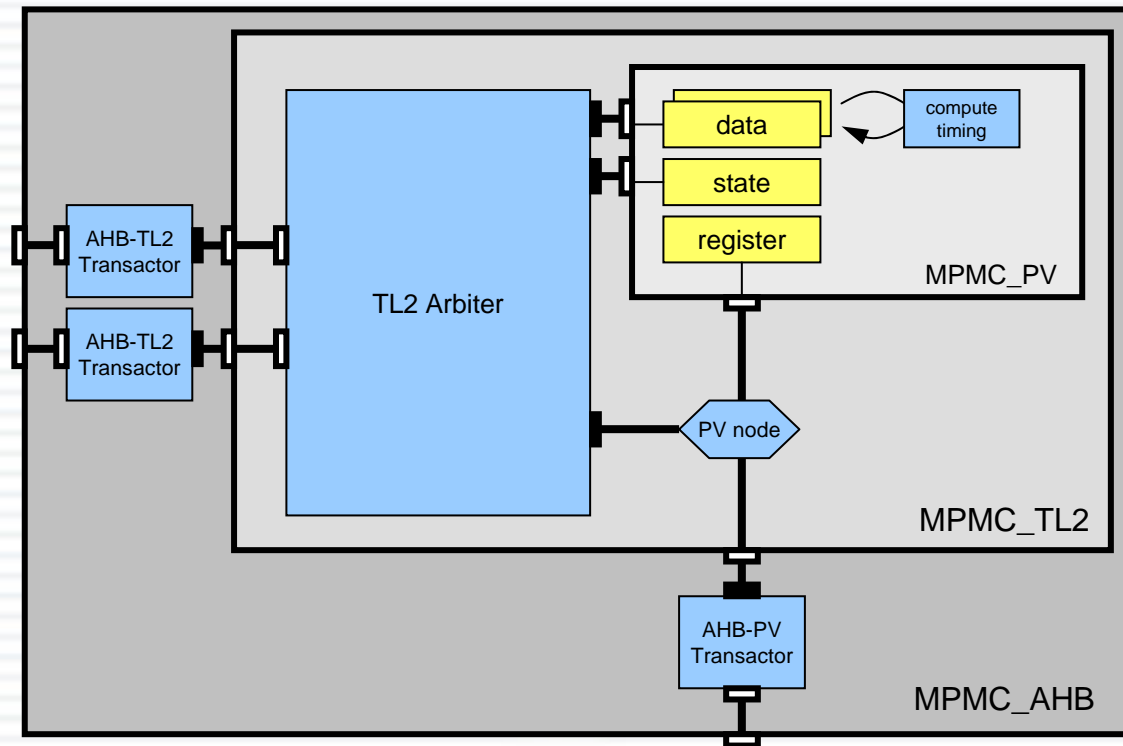
Outline

- Motivation
- How to Use Programmer's View Timing Annotation to Create Reusable TLM Models
- Case Study: Memory Controller

Multi-Port Memory Controller Case Study



Multi-Port Memory Controller Case Study



Success!

- Lessons learned:
 - PV timing annotation not enough for modeling arbitration
 - Partitioned model into 2 parts, still enabling high reuse
- Verified against cycle accurate reference
 - Stopped timing refinement at 98% accuracy
- Modeling effort vs. RTL
 - Estimate 1/10 the effort for PV model alone
 - Estimate 1/3 the effort to develop the PV model, timing model, and arbiter

Thank You!

Charu Khosla
CoWare, Inc.

2/21/07