



# Introduction to the SystemC TLM Standard

**Stuart Swan**

**Cadence Design Systems, Inc**

**June 2005**

# SystemC Transaction Level Modeling



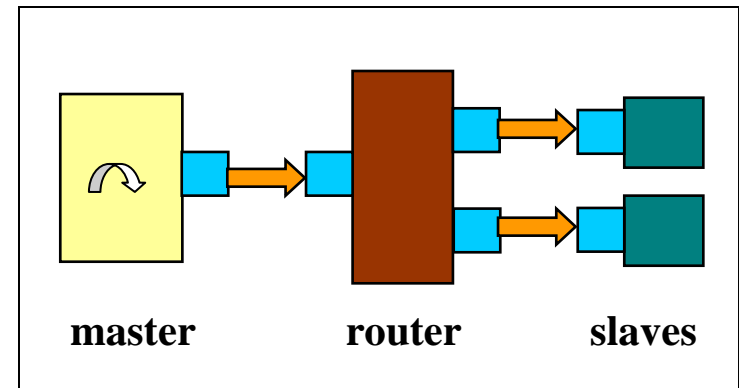
- *What is TLM?*

- Communication uses function calls

```
burst_read(char* buf, int addr, int len);
```

- *Why is TLM interesting?*

- Fast and compact
- Integrate HW and SW models
- Early platform for SW development, easy to distribute
- Early system exploration and verification
- Verification reuse



# SystemC Transaction Level Modeling



- *How is TLM being adopted?*
  - Widely used for verification
  - TLM for design is starting at major electronics companies
- *Is it really worth the effort?*
  - Yes, particularly for platform-based design and verification
- *What will help proliferate TLM?*
  - Standard TLM APIs and guidelines
  - Availability of TLM platform IP
  - Tool support

## ➤ SystemC TLM Standard

# May 2005: OSCI Releases SystemC TLM Standard



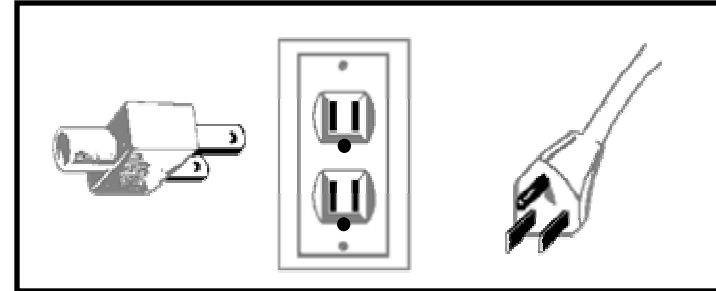
- Full press release available at [www.systemc.org](http://www.systemc.org)
- Companies endorsing TLM standard within press release:
  - Cadence, CoWare, Forte, Mentor, Philips, ST, Synopsys
  - Atrenta, Calypto, Celoxica, Chip Vision, ESLX, Summit, Synfora
  - OCP-IP
- TLM kit, whitepaper, and examples publicly available at [www.systemc.org](http://www.systemc.org)
- TLM standard is already in use in industry
- IEEE standardization process to begin soon

# TLM API Goals



- Common API foundation for transaction level modeling in SystemC
- Support design & verification IP reuse
- Provide common TLM recipe
- Ease of use, Safety, Speed
- Generality
  - Abstraction Levels
  - HW / SW
  - Different communication architectures (bus, packet, NOC, ...)
  - Different protocols

# Key Concepts



- Focus on SystemC interface classes
  - Define small set of generic, reusable TLM interfaces
  - Different components implement same interfaces
  - Same interface can be implemented
    - directly within a C/C++ function, or
    - via communication with other modules/channels in system
- Object passing semantics
  - Similar to `sc_fifo`, effectively pass-by-value
  - Avoids problems with raw C/C++ pointers
  - Leverage C++ smart pointers and containers where needed

# Key TLM Terms



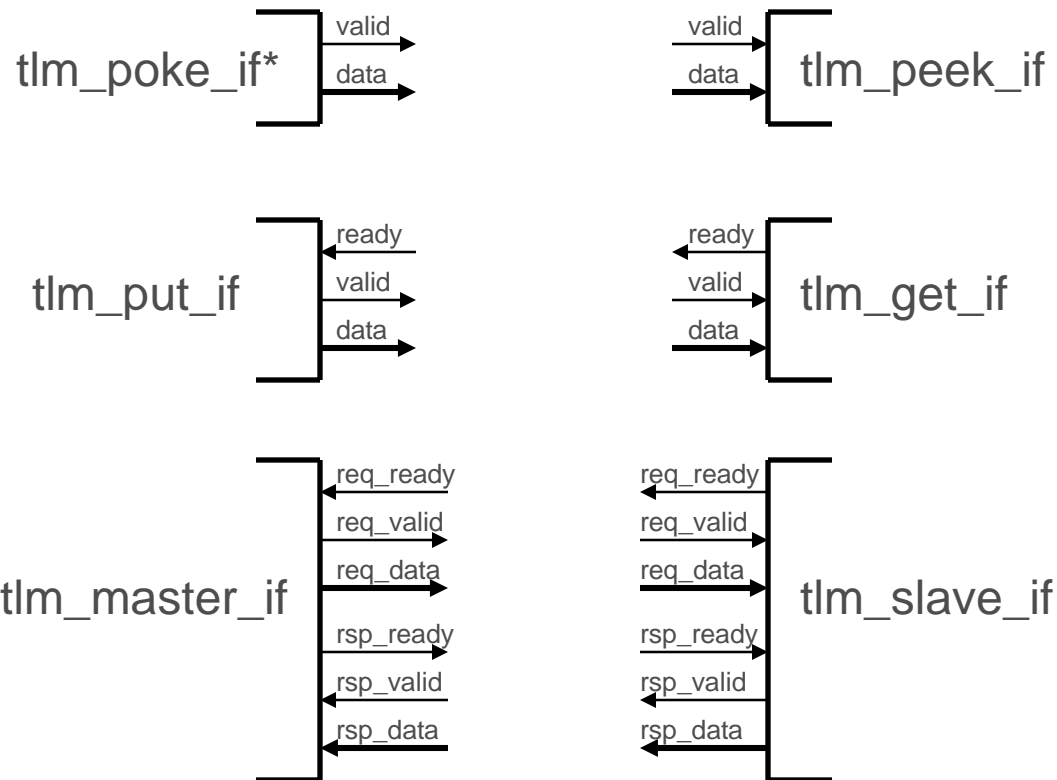
- **Nonblocking:** Means function implementations *can never* call wait().
- **Blocking:** Means function implementations *might* call wait().
- **Unidirectional:** data transferred in *one* direction
- **Bidirectional:** data transferred in *two* directions
- **Poke/Peek:** Poke overwrites data and can never block. Peek reads most recent valid value. Poke/Peek are similar to write/read to a variable or signal.
- **Put/Get:** Put queues data. Get consumes data. Put/Get are similar to writing/reading from a FIFO.
- **Pop:** A pop is equivalent to a get in which the data returned is simply ignored.
- **Master/Slave:** A master initiates activity by issuing a *request*. A slave passively waits for requests and returns a *response*.

# Primary TLM Interfaces



- Primary Unidirectional Interfaces
  - tlm\_poke\_if<T> / tlm\_peek\_if<T>
  - tlm\_put\_if<T> / tlm\_get\_if<T>
- Primary Bidirectional Interfaces
  - tlm\_master\_if<REQ, RSP>
  - tlm\_slave\_if<REQ, RSP>
  - tlm\_transport\_if<REQ, RSP>
- *Note: tlm\_poke\_if not yet in OSCI TLM kit, should be added soon*

# Hardware Implied by TLM Interfaces



The TLM interfaces can be easily mapped to HW. Understanding this mapping helps you to understand how to use the TLM interfaces.

Note that the TLM interfaces are also useful in non-HW parts of your system (e.g. testbenches, SW modeling).

Poke/peek have overwrite semantics similar to writing to a variable or signal

Put/get have queuing semantics similar to writing to a FIFO

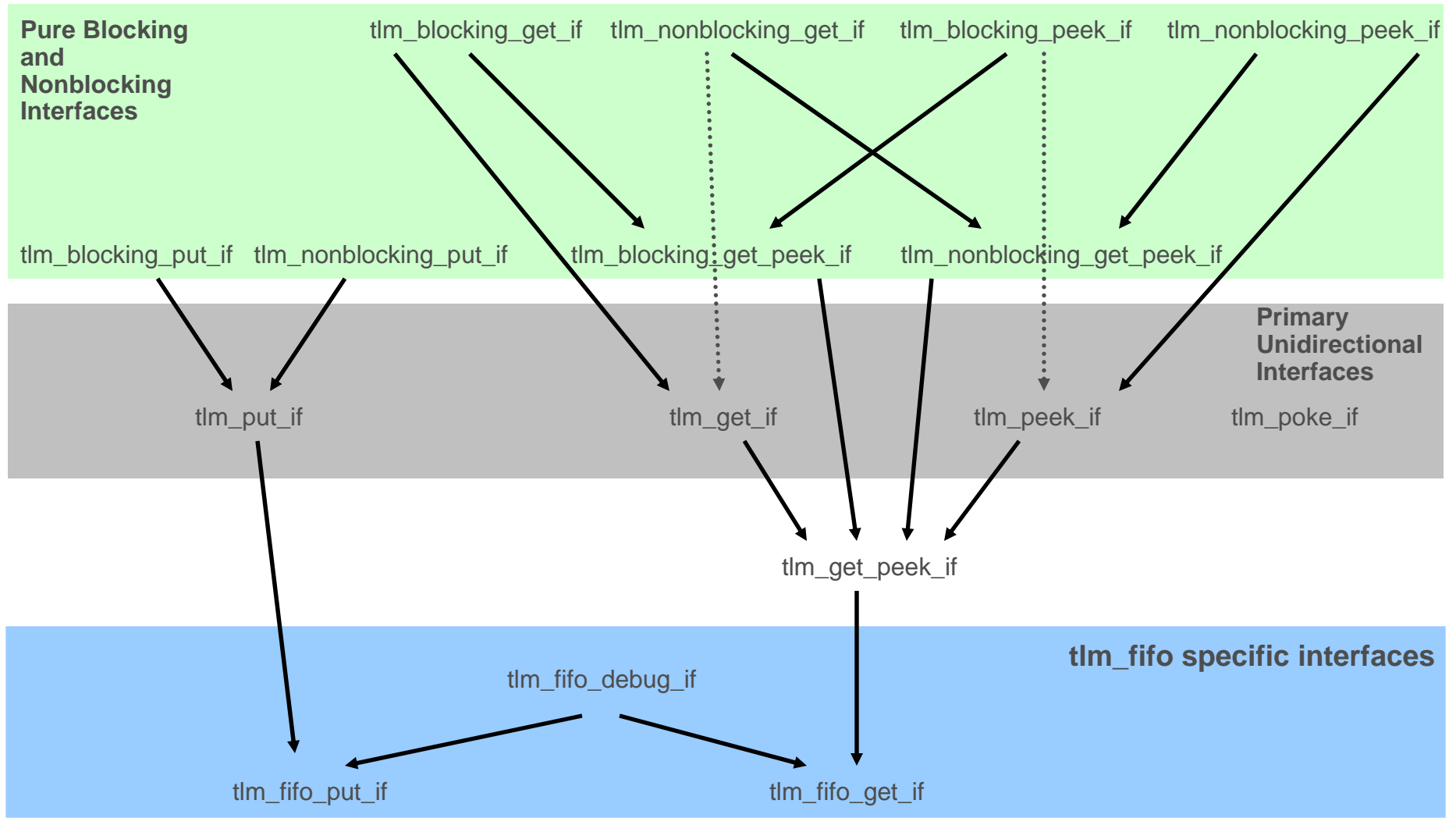
When values propagate asynchronously, combinational logic is implied.

When values are held across clock edges, hardware registers are implied.

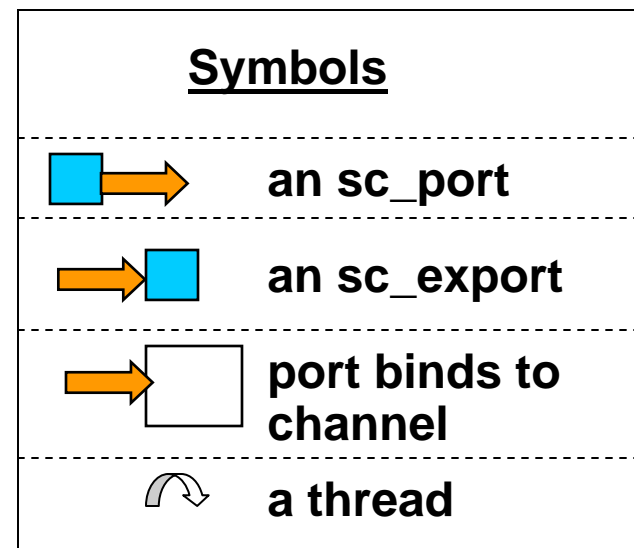
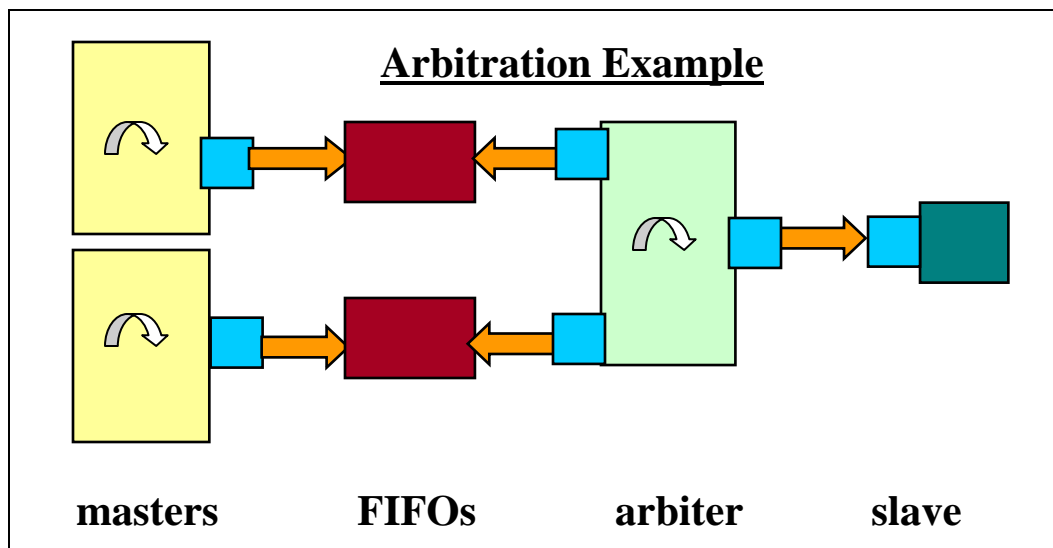
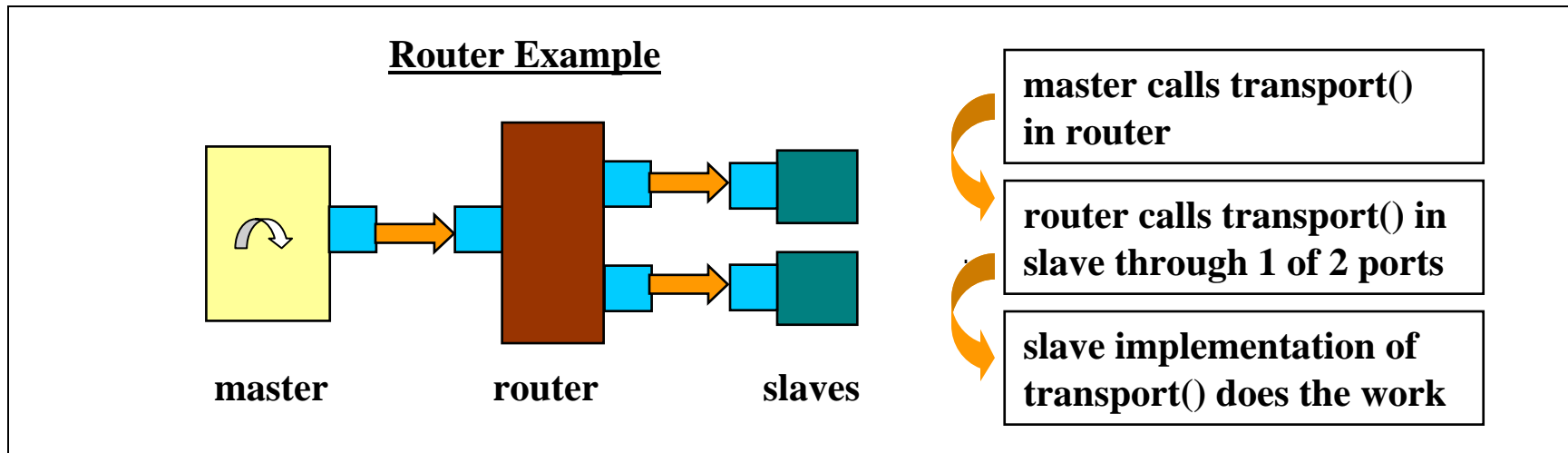
tlm\_transport\_if implies same HW as tlm\_master\_if, but also requests and responses are tightly coupled.

tlm\_poke\_if is not yet in OSCI TLM standard, should be added soon.

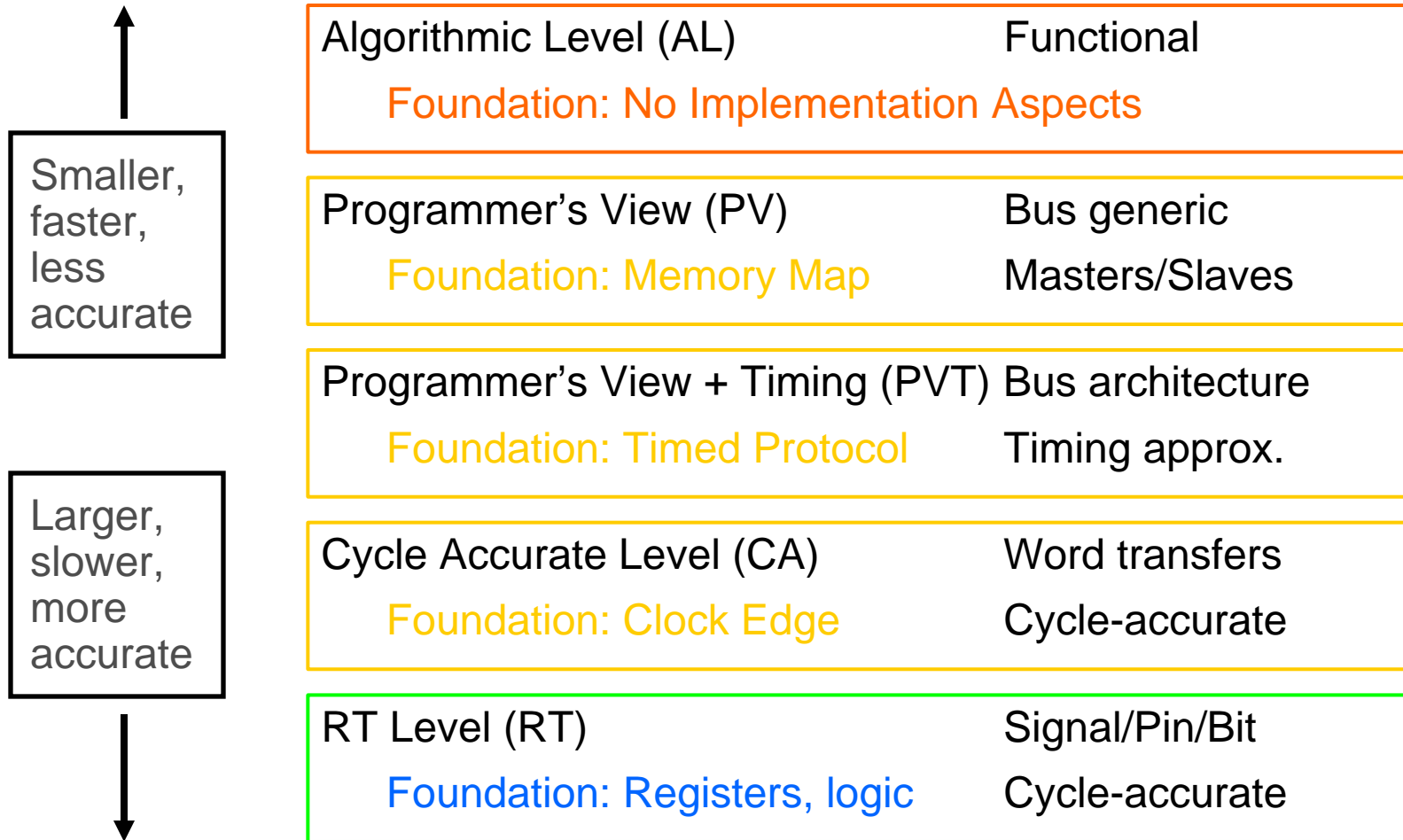
# TLM Unidirectional Interfaces Inheritance Diagram



# Transaction Level Modeling with the TLM API



# TLM Abstraction Levels

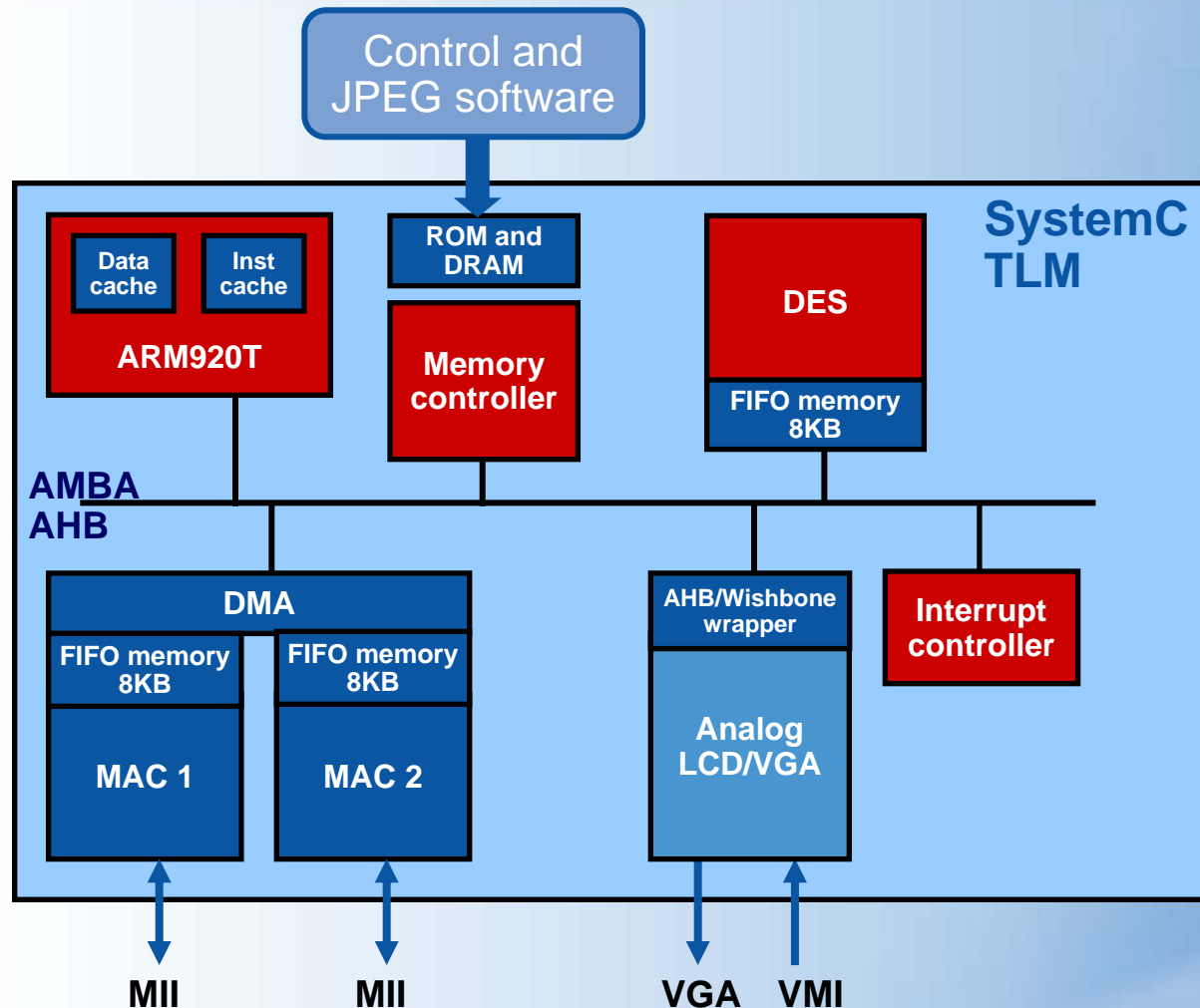


*Model at a few levels that target the "pain" and risk in your D&V flow*

# Example TLM Application #1



## Wireless Picture Frame based on ARM920T

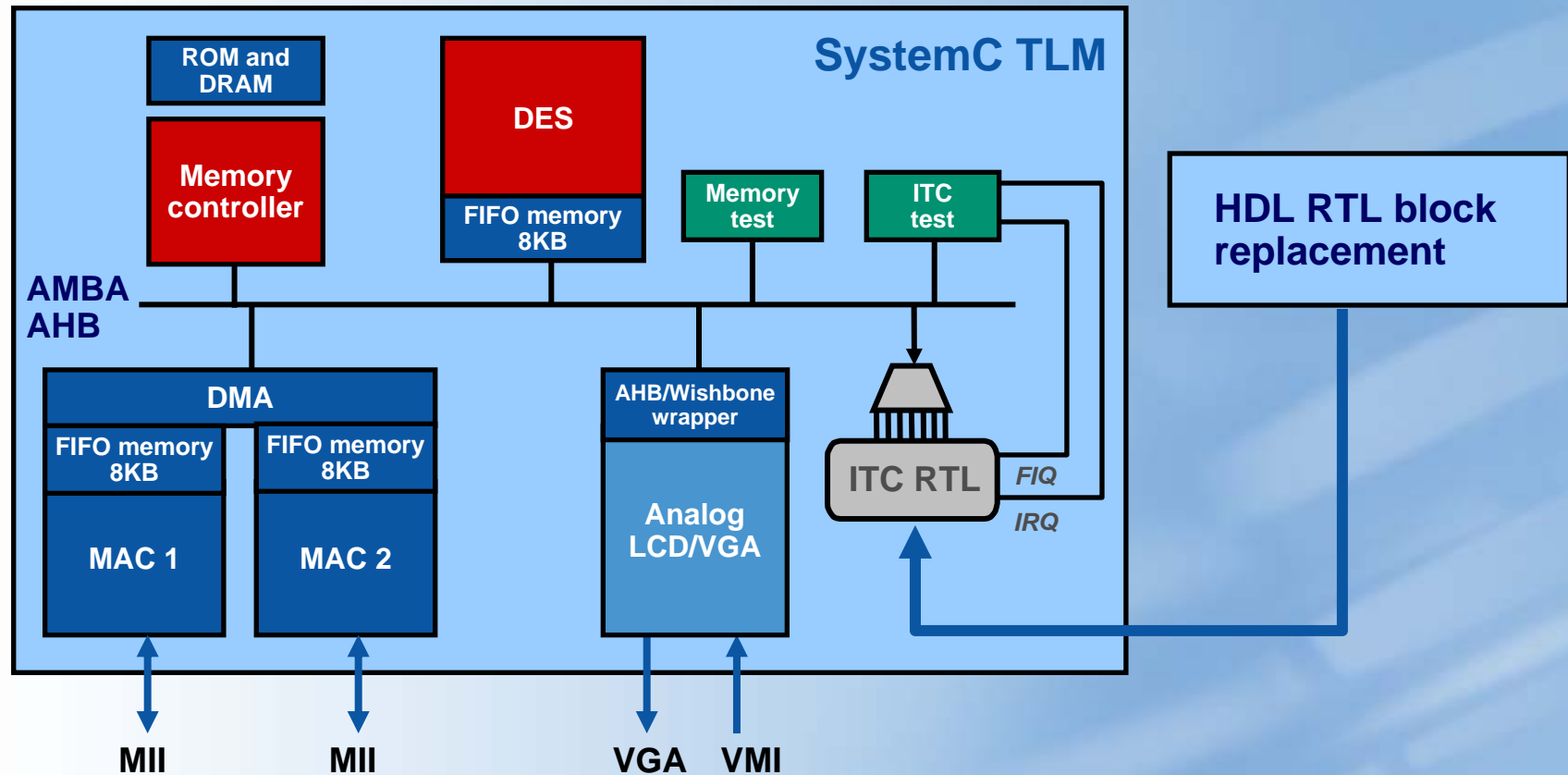


### SystemC TLM Model

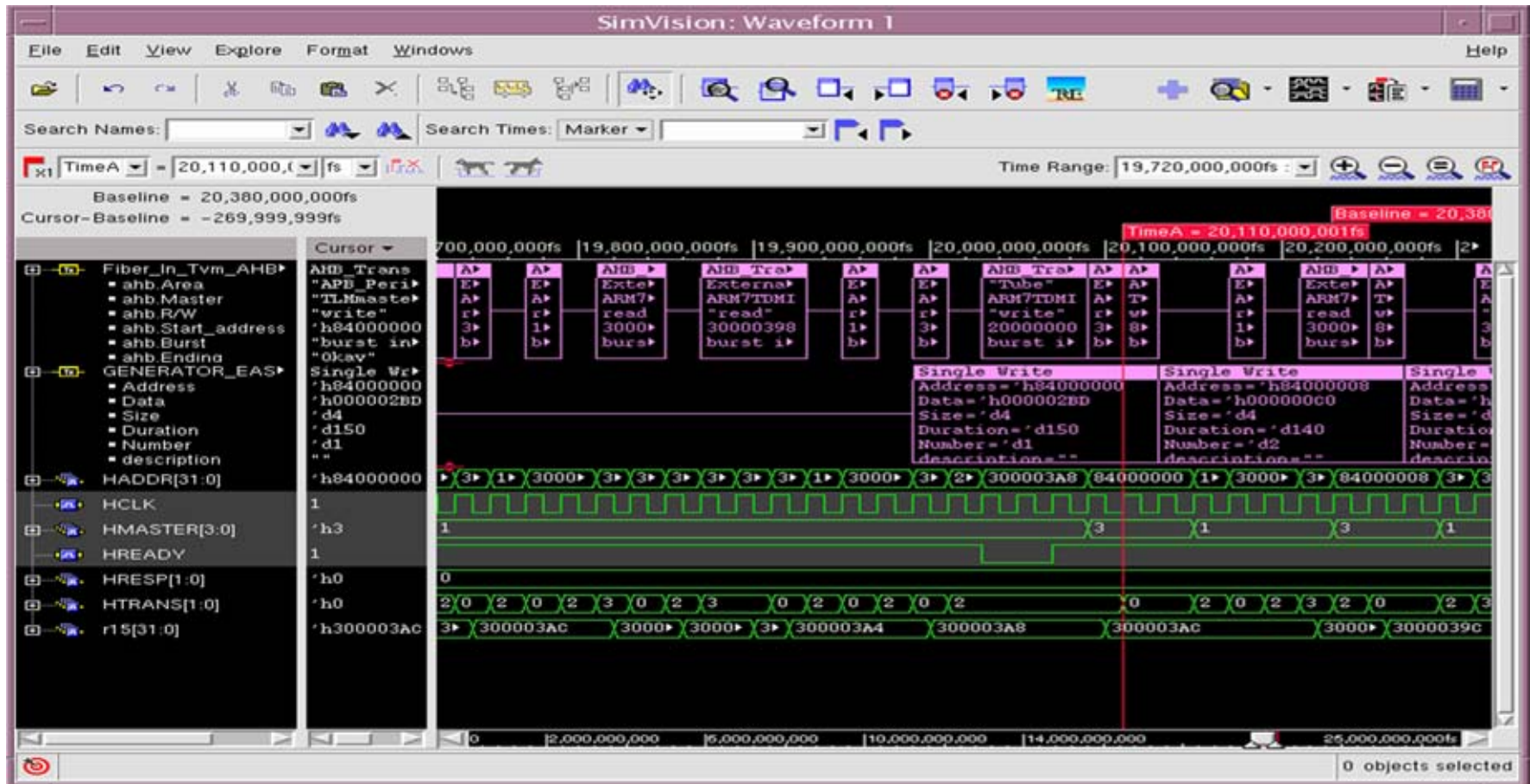
- Full address map
- Models all AHB transactions
- ARM920 ISS master
- Multiple slaves
- Complete SW on ISS
  
- PV provides ~1M cps
- CA provides ~50kcps

# Example TLM Application #2

*PV TLM Model reused to drive RTL block*



# Incisive Unified Transaction and Signal Viewing



Ability to visualize transactions alongside signals makes debugging much easier

# Conclusions



- Benefits of TLM:
  - Fast and compact
  - Integrate HW and SW models
  - Early platform for SW development, easy to distribute
  - Early system exploration and verification
  - Verification reuse
- TLM is the next level of design and verification abstraction in EDA, and the shift is now starting.
- The OSCI TLM standard is available now and is already in use, and should foster the development of a TLM IP ecosystem.