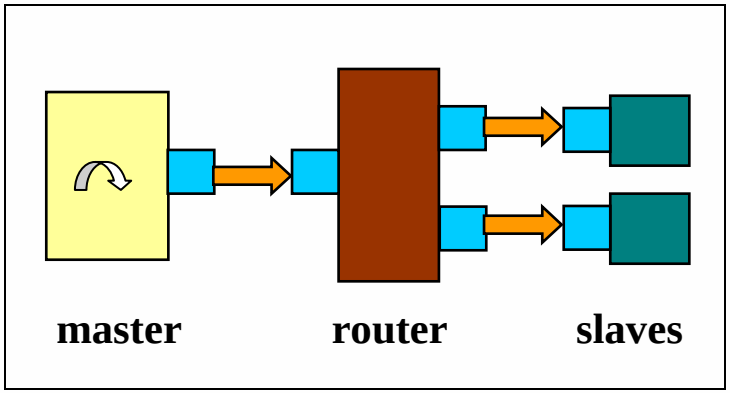
Towards a SystemC Transaction Level Modeling Standard

Stuart Swan
Adam Rose
John Pierce
June 2004

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
  - Early system exploration and verification
  - Verification reuse



SYSTEMC™

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

SystemC TLM Standards Efforts

- OSCI TLM WG
- OCP-IP
- June 2004: OSCI / OCP-IP TLM Standardization Alliance
  - Agreement to build on a common TLM API foundation
- TLM API proposal from Cadence distributed to OSCI and OCP-IP
  - Proposal intended as common foundation for OSCI and OCP-IP
  - Allows protocol-specific APIs (e.g. AMBA, OCP)
  - Wide range of abstraction levels

Endorsements of Current TLM Proposal

"We are excited about the TLM API proposal that is currently being reviewed by the OSCI TLM working group. This proposal satisfies the technical requirements of the TLM-API WG. We believe it can provide the standard foundation that enables transaction level SystemC IP to be developed and reused quickly and efficiently."

- Adam Donlin, Xilinx
- Frank Ghenassia, ST Microelectronics, Chairman of OSCI TLM WG
- Serge Goossens, CoWare
- Anssi Haverinen, Nokia, Chairman of OCP-IP TLM Working Group
- Mike Meredith, Forte Design Systems
- Stuart Swan, Cadence Design Systems

TLM API Goals

- Support design & verification IP reuse
- Provide common TLM recipe
- Usability
- 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 Concepts (cont.)

- Unidirectional vs. bidirectional dataflow
  - Unidirectional interfaces are similar to sc_fifo
  - Bidirectional can be easily and cleanly layered on unidirectional
  - Separates requests from responses
- Blocking vs. nonblocking
- Use sc_port & sc_export

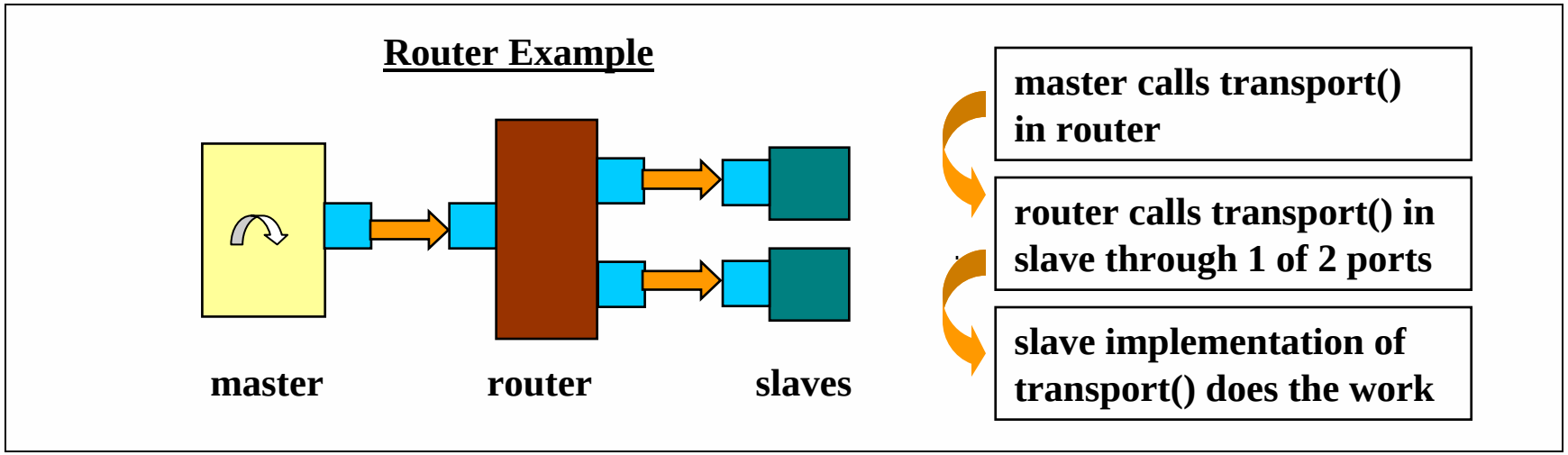
Layered TLM API Architecture

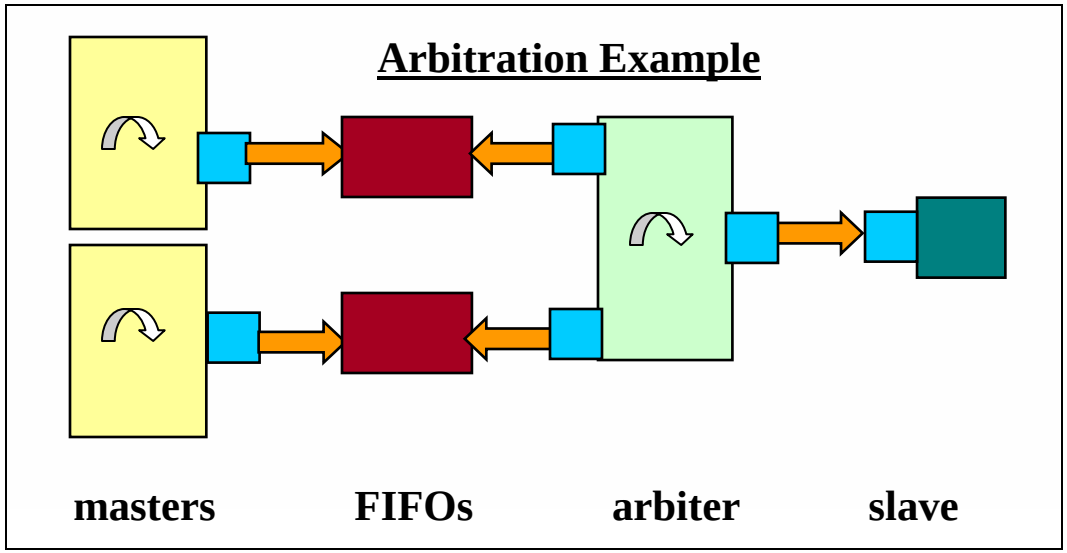
User Layer Protocol-specific "convenience" API Targeted for embedded SW engineer Typically defined and supplied by IP vendors	amba_bus->burst_read(buf, adr, n);

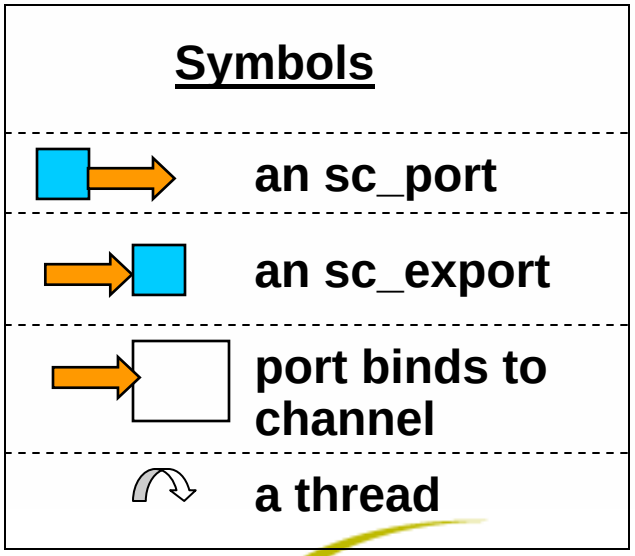
Protocol Layer
Protocol-specific code
Adapts between user layer and transport layer
Typically defined and supplied by IP vendors
	req.addr = adr; req.num = n;
	rsp = transport(req);
	return rsp.buf;

Transport Layer Uses generic data transport APIs and models Facilitates interoperability of models Key focus of TLM standard May use generic fifos, arbiters, routers, xbars, pipelines, etc.	sc_port<tlm_transport_if<REQ, RSP> > p;

Transaction Level Modeling with the TLM API

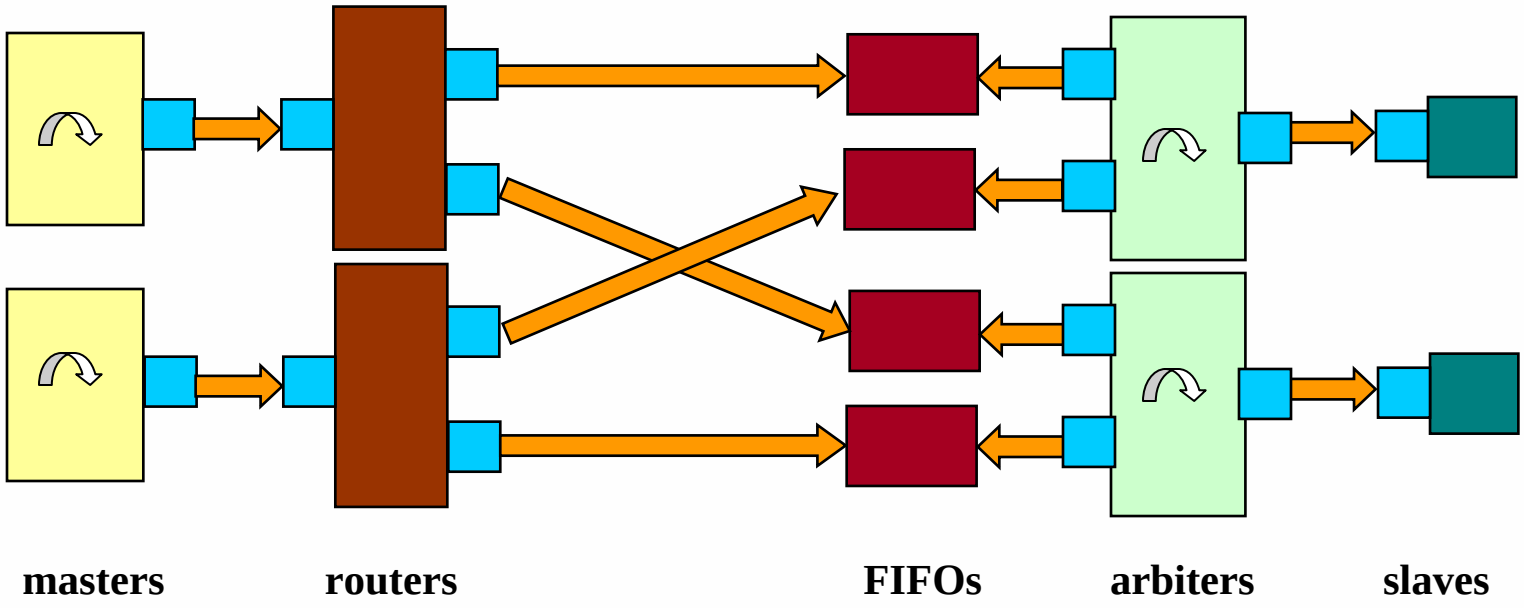






Transaction Level Modeling – Cross Bar

- Uses the same components on the previous slide connected in different ways



Cross Bar Switch

Getting More Information

- Join OSCI and the TLM WG
  - www.systemc.org
- Contact me - Stuart Swan
  - stuart@cadence.com
- Contact Chairman of OSCI TLM WG – Frank Ghenassia
  - frank.ghenassia@st.com
- Contact Chairman of OCP-IP TLM WG – Anssi Haverinen
  - anssi.haverinen@nokia.com
- Any Questions?