Draft TLM 2.0 Overview

11/29/06

Agenda

- Roadmap
- TLM Generic principles
- Untimed TLM modeling (PV)
- Timed TLM modeling (PVT)
- TLM analysis
- Examples

Layered Standards

	User IP
Development Started Q305	TLM Interoperability Layer
OSCI : Summer 05 IEEE : 06	TLM 1.0 – Common Transport Mechanism
IEEE 1666	SystemC Core Language
ANSI C++	C++

Active OSCI TLM WG Members

- ARM
- BlueSpec
- Cadence
- ChipVision
- CoWare
- Doulos
- ESLX
- Forte
- Freescale
- GreenSocs
- Intel
- Jeda
- Mentor
- Philips
- SpringSoft
- ST
- Summit
- Synopsys
- Tuebingen University

- Other Organisations
  - OCP is very supportive of these efforts
    - There is significant X membership between the two organisations
    - Technical Chair of OCP is an OSCI member
  - There is also significant X membership with SPIRIT

What are we working on ?

How do we move transactions about ?

- Minor release of existing TLM 1.0 kit (bug fixes)
- Standard Bus Modeling APIs
  - Generic payload usable for PV and PVT modeling
  - Interrupt Modeling
  - Memory Map Services
  - Memory / Register Modeling

What transactions do we move about ?

- Standard Configuration and Control APIs
  - Configuration Interface
  - Debug Interface
  - Analysis Interface

How do we ***control*** and ***analyse*** the transactions moving through the TLM ?

TLM Roadmap

1.0.1

* Bug fixes

* Timed TLM core i/f
* Analysis interface
* Generic payload
* Examples

Draft 2.0
Public review

2.1
Candidate release

* Interrupt payload
* Debug interface
* Reg / mem objects
* Examples
* Users feedback

2.1 & LRM
Official release

* LRM
* Users feedback

2.2

* Configuration
* Profiling
* Memory map
* TLM IEEE

July Nov'06 DATE'07 DAC'07 TBD

Draft 2.0 release: Overview

- Untimed TLM modeling
  - Generic payload
    - Mostly for abstract modeling of transactions over on-chip bus
- Timed TLM modeling
  - Update of core interfaces for timed modeling
    - Mostly sc_time parameter to core interfaces
  - Generic tlm_bus payload
    - Mostly for performance modeling of communication over on-chip bus, taking into account pipelining
- Analysis interface
  - To monitor TLM ports

Agenda

- Roadmap
- Definitions and principles
- Untimed TLM modeling (PV)
- Timed TLM modeling (PVT)
- TLM analysis
- Examples

Definitions 1/3

- TLM terms
  Terms required to discuss/describe the TLM standard

  - *TLM core interface*: SystemC interface defined in the TLM kit
  - *TLM target port:* sc_export bound to a TLM core interface
  - *TLM initiator port:* sc_port bound to a TLM target port
  - *TLM target* : SC_MODULE instantiating at least one TLM target port
  - *TLM initiator* : SC_MODULE instantiating at least one TLM initiator port
  - *TLM transaction*: Data structures transferred by TLM core interface
  - *TLM Interoperability*: Ability to directly connect models together and have them exchange transactions with no code modification

Definitions 2/3

- System terms
  Terms required to discuss/describe system being modeled with the TLM standard
  - System initiator: definition depends on system being modeled… an example is a CPU issuing read/write requests
  - System target: definition depends on system being modeled… an example is a memory servicing read/write requests
  - System transaction: definition depends on system being modeled… an example is a read/write operation from a CPU to a memory

Definitions 3/3

- Corollary on definitions
  - A system initiator is always a TLM initiator
  - A system component might be both initiator and target
  - A SystemC module might be both a TLM initiator *and* target
  - A system transaction might be built from a sequence of several TLM transactions

- Notes
  - ‘Initiator’ is interchangeable with ‘master’
  - ‘Target’ is interchangeable with ‘slave’

TLM principles 1/2

- OSCI TLM standard intends to support efficient and safe exchange of transactions between SystemC modules

- TLM 1.0 defines core interfaces to transfer transactions between modules:
  - *Based on message-passing scheme*
  - *transport()* for bi-directional information exchange with one single IMC
  - *put()*, *get(), peek(), poke()* variants for unidirectional information transfer
  - Effective pass-by-value mechanism:
    - *transport() and put(): “const &” for interface parameters*
    - *transport() and get() : data sent back to caller as return value*

TLM principles 2/2

- Transaction lifetime corresponds to the duration of the execution of the core TLM interface
  - Unless TLM protocol guarantees extended validity

- Content of TLM transaction is only guaranteed to remain valid during the lifetime of the transaction
  - Unless TLM protocol guarantees extended validity

Generic payload

- Provide generic support for bus-based communication (ie. memory-mapped systems)
  - Built-in extension mechanism for bus-specific features

- Message-passing “philosophy”
  - Default implementation with effective pass-by-value mechanism
  - Required for transaction storage for usage after transaction completes

- Speed optimization for IP modeling
  - Requires explicit activation of pointer-mode
  - Pointer-mode requires compliance to rules described in slide 15

Generic payload: pointer-mode rules 1/2

- Usage when request and response structures used in matching pairs, typically for memory-mapped busses
- Allocate and Deallocation
  - if you allocate, you must deallocate
  - if you can't or won't deallocate, use pass-by-val
  - people may call you a master, system initiator or whatever if you allocate
- Read / Write
  - if you're a master and the transaction is a write, you have write permission on arrays in the request
  - if you're a master and the transaction is a read, you have read permission on the arrays in the response
  - if you're a slave and the transaction is a write, you have read permission on the data in the request
  - if you're a slave and the transaction is a read, you have write permission on the data in the response
  - No other read or write permissions are granted on the arrays other than those specified above.
- Request / Response matching
  - the slave must copy the pointers to the arrays from request to response.
  - the slave should copy common fields (transaction_id, master_thread_id, block_size, export_id, priority) from request to response.

Generic payload: pointer-mode rules 2/2

- Data structures must be allocated and owned by the TLM initiator
- To ensure thread safety, the tlm_* data structures should not be member of the TLM initiator class, but local variables of the function initiating the transaction (sc_thread)
- On the system target side,
  - *Write*: the conveyed data needs to be locally copied
  - *Read*: the TLM system target copies the locally-stored data to the initiator-owned data structure

tlm_request data structure 1/2

m_command	read, write– enumerated type
m_mode	REGULAR, DEBUG, CONTROL – enumerated type
m_address	Templatized data type. *TLM data types recommended, but not compulsory*
m_block_mode	INCREMENT, STREAMING, WRAP, -Enumerated type
m_block address incr	Unsigned int
m_master_thread_id	Unsigned int
m_transaction_id	Unsigned int
m_custom_vector_ptr *	Pointer to a vector of request extensions *By-value mode: points to extension local copy* *By-pointer mode: points to user extension*
m_data *	Templatized data type. *TLM data types recommended, but not compulsory* *By-value mode: points to request local copy of data content* *By-pointer mode: points to user copy of data content*
m_block size	Unsigned int: can be 1.N. Determines how many chunks of data are transferred. Block_size =1 for single transfer

tlm_request data structure 2/2

m_byte enable *	Array of byte enable information. *By-value mode: points to request local copy of byte enable content* *By-pointer mode: points to user copy of data content*
m_byte_enable_period	Byte enable information for blocks
m_priority	Unsigned int, specifying priority of current transaction
m_export_id	Unsigned int – used to identify the incoming TLM target port used to transfer the transaction in current module

tlm_response data structure

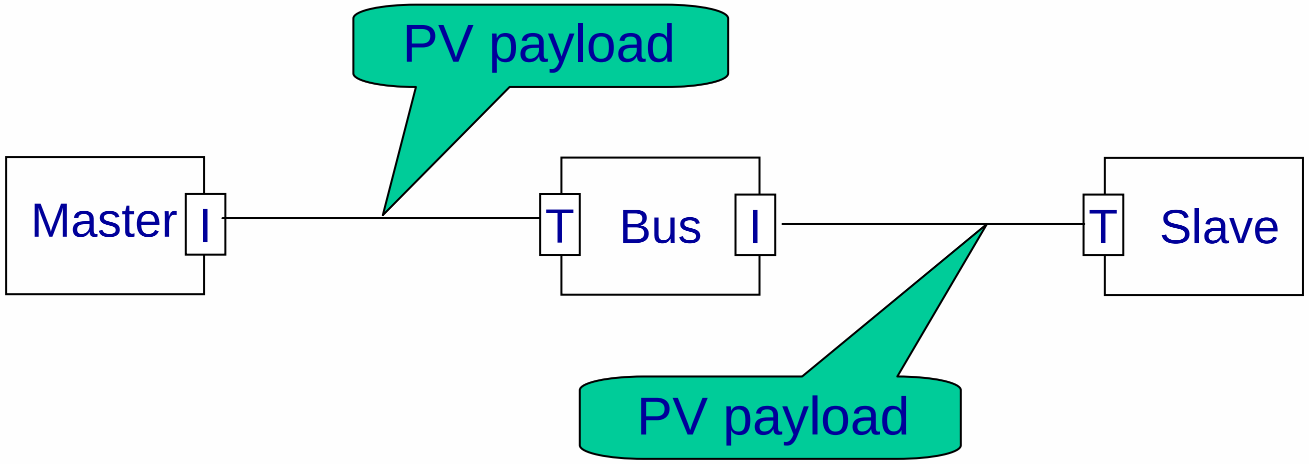
m_status	tlm_bus_status class: OK, ERROR, NO_RESPONSE
m_transaction_id	Unsigned int
m_master_thread_id	Unsigned int
m_data *	Templatized data type. *TLM data types recommended, but not compulsory* *By-value mode: points to response local copy of data content* *By-pointer mode: points to user copy of data content*
m_block_size	Unsigned int: can be 1.N. Determines how many chunks of data are transferred.
m_export_id	Unsigned int – used to identify the incoming TLM target port used to transfer the transaction in current module
m_priority	Unsigned int, specifying priority of current transaction
m_custom_vector_ptr*	Pointer to vector of response extensions *By-value mode: points to extension local copy* *By-pointer mode: points to user extension*

Agenda

- Roadmap
- TLM principles
- Modeling styles with Generic payload
- TLM analysis
- Examples

Untimed TLM modeling

- Based on *transport()* core TLM interface
- Defines transaction content and associated programming rules
- All models using the PV payload and transport core interface can be connected and simulated together

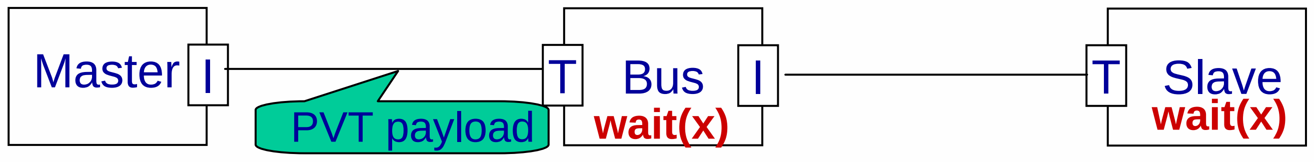


Timed TLM modeling 1/2

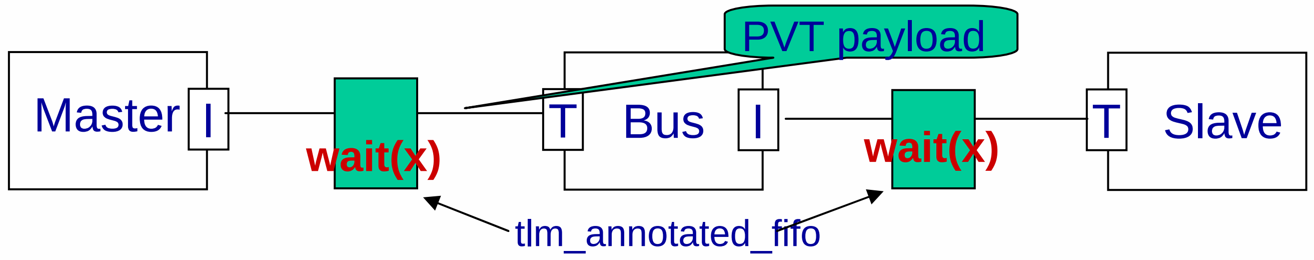
- Enable timing annotations of event-based simulations
- With PVT payload, based on unidirectional TLM interfaces
- All models using the PVT payload and same core interfaces can be connected and simulated together

Timed TLM modeling 2/2

- 2 structural approaches supported:
  - 1) Insert delays in models



  - 2) Rely on annotated fifo between components to take timing delays into account, using timed TLM core interfaces

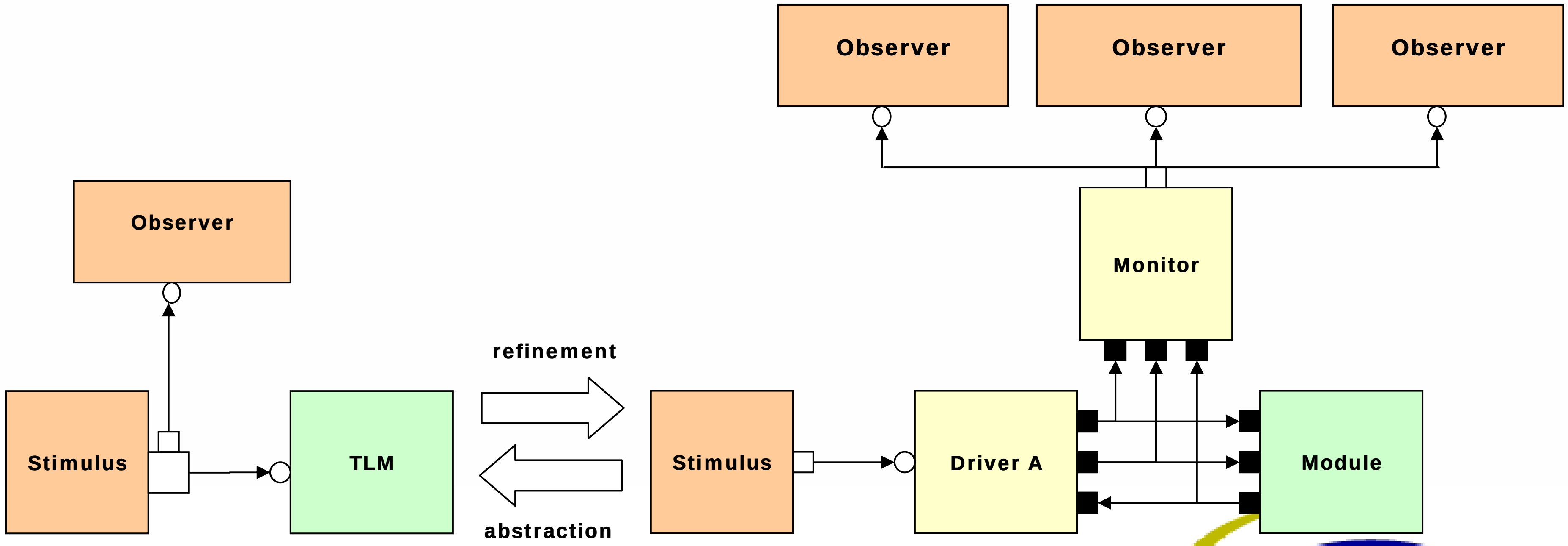


Agenda

- Roadmap
- TLM principles
- Untimed TLM modeling (PV)
- Timed TLM modeling (PVT)
- TLM analysis
- Examples

Analysis Ports

- Non-intrusive monitoring of transactions going through TLM ports
  - Essentially, this is a SystemC implementation of the observer pattern
- Main features
  - Possibility to connect zero, one or many observers to a single analysis port
  - Non blocking interface
  - Possibility to use the same port and interface for RTL monitors and TLM level communication



Examples provided with TLM 2.0 1/2

- PV example
  - Illustrate usage of transport and generic payload for untimed TLM modeling

- PVT annotated examples
  - Illustrate usage of timed annotated interfaces for timed TLM modeling
  - tlm_annotated_fifo used to connect initiator and target ports
  - Timing of transactions managed in annotated fifo

- PVT put example
  - Illustrate usage of standard unidirectional interfaces for timed TLM modeling
  - Direct binding between initiator and target ports
  - Timing of transactions modeled managed in IPs

Examples provided with TLM 2.0 2/2

- Export_id
  - Illustrate usage of export_id feature to identify incoming target port of the transaction
- Byte enable
  - Illustrate semantic and available modes to support partially valid data contained in transaction (also referred as byte lanes)

Conclusion

- TLM 1.0 provided foundation for TLM modeling
- TLM 2.0 specifies PV and PVT interfaces, enabling model interoperability for untimed and timed TLM models. Kit also provides mechanism for analysis of transactions
- TLM WG will address remaining required features defined in roadmap
- Evaluate and send feedback !
  - TLM 2.0 user feedback will be considered for enhancements of future TLM kits
  - Send feedback and questions to tlm-group@systemc.org