OCP socket modelling with TLM-2.0

Announcement of OCP-IP's First Release

Hervé Alexanian, Sonics, inc

Introduction to OCP

- Open Standard
  - Owned by the OCP International Partnership
- OCP-IP provides much more than only a protocol
  - Functional verification specifications
  - Verification tools: BFMs and protocol checkers
  - Parameter capture formats
  - RTL timing classes
  - Analysis and debug tools
  - System-Level Design support
    - Standard interfaces for SystemC models of cores as well as RTL models of cores
- Enabling automation of core provision and SoC specification and assembly

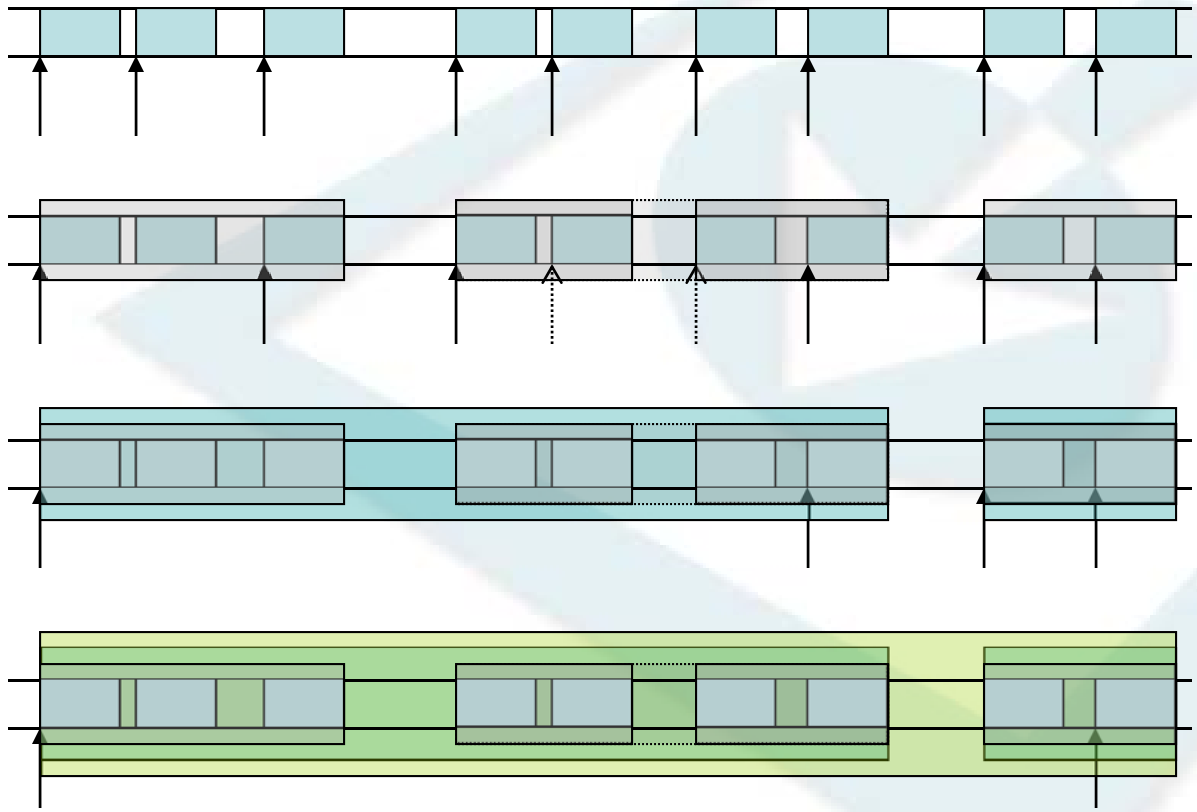
The *OCP Modelling Kit* Has Been Released

- Public Standard TLM interfaces
  - Based on and compatible with OSCI TLM 2.0
- 4 Levels of abstraction supported: fully cycle-accurate to fully untimed
- OCP configuration management
  - May be hard-coded or supplied to a generic component model at run-time
  - Run-time resolution of master and slave OCP configurations
- OCP master and slave sockets, providing
  - Memory management for extensions and payload objects
  - Payload event queues for timing annotation support or clock cycle synchronization
  - Convenience API for user code
  - Direct bind to OSCI TLM 2.0 sockets where functionally possible
- Performance and trace monitors
- Legacy adapters
- RTL adapters
- Documentation
- Examples

OCP-IP SystemC Next Generation Interface Standards

	OCP-IP SystemC Interface	OSCI TLM compatibility
TL0	Not specified by OCP-IP separately for SystemC from other HDLs	None, this is the RTL level
TL1	OCP-IP TL1	Uses TLM-2 generic payload, sometimes with extensions. Uses different protocol phases and rules from OSCI TLM-2.0 BP. Uses nb_transport()
TL2	OCP-IP TL2	Uses TLM-2 generic payload, sometimes with extensions. Extensions are a subset of the extensions used at OCP-IP-TL1. Uses different protocol phases and rules from OSCI TLM-2.0 BP and from OCP-IP-TL1. Uses nb_transport()
TL3 TL4	OCP-IP TL3/TL4	Uses TLM-2 generic payload, sometimes with extensions. Extensions are a subset of the extensions used at OCP-IP-TL2. Uses the same protocol phases and rules as OSCI-TLM-2.0 BP. Extensions may be ignorable in which case OCP-IP-TL3 is directly interoperable with OSCI-TLM-2.0-BP. Uses nb_transport() and b_transport()

Abstraction levels



Timing points accurate to the cycle. Combinatorial paths handled

Meta data used For dynamic calculation to increase accuracy

Static calculation for Request/Response

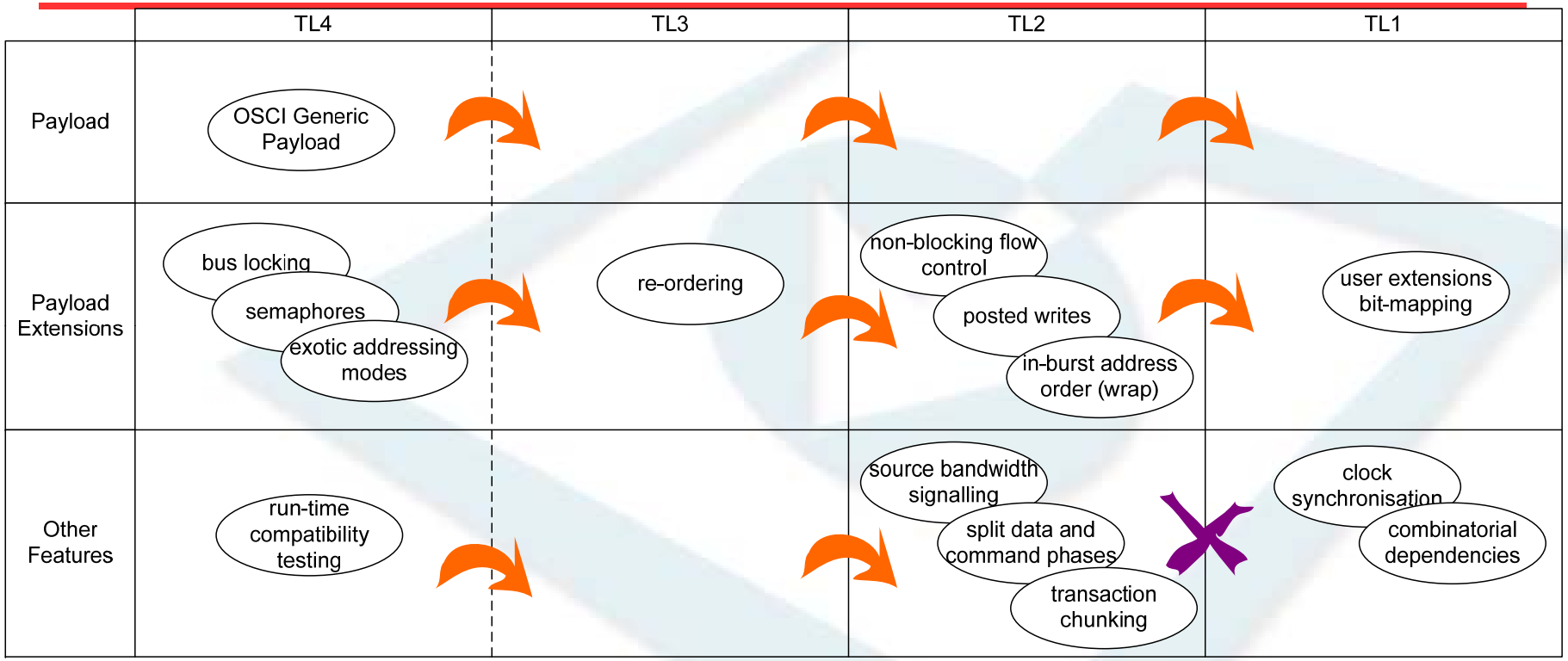
Static calculation for entire transaction SystemC time advances in quanta.

Layered Structure of the Interfaces

- Prior Kit had Flaws
  - Incomplete documentation
  - Custom data structures
  - Each layer had variations in data representation
  - Forced custom layer adapters

- Using the TLM generic payload allows
  - More reuse at different layers
  - Each layer to focus on its timing specificities
  - Much easier for layer adapters

Layered Structure of the Interfaces



- The orange arrows show where technology from a high level of abstraction is re-used at a lower level
- Thus TL2 is a superset of TL3 which is a superset of OSCI BP
- TL1 is not quite a superset of TL2 but is a superset of TL3
  - TL1 and TL2 technology for modelling timing is different

OCP-IP Socket…

- TLM-2.0 defines the concept of ‘sockets’
- **OCP-IP provides an OCP specific socket with a number of important features.**

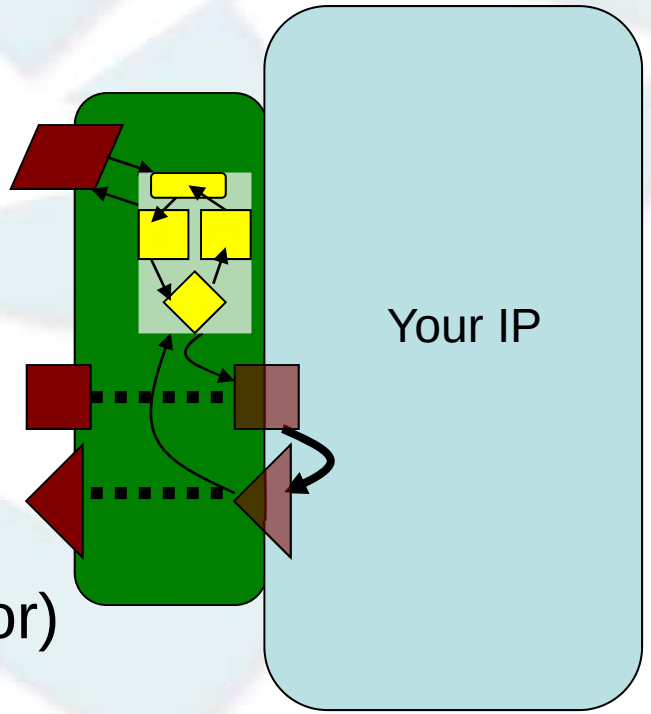
Protocol negotiation to cover all OCP’s

Memory management

Safe handling of time

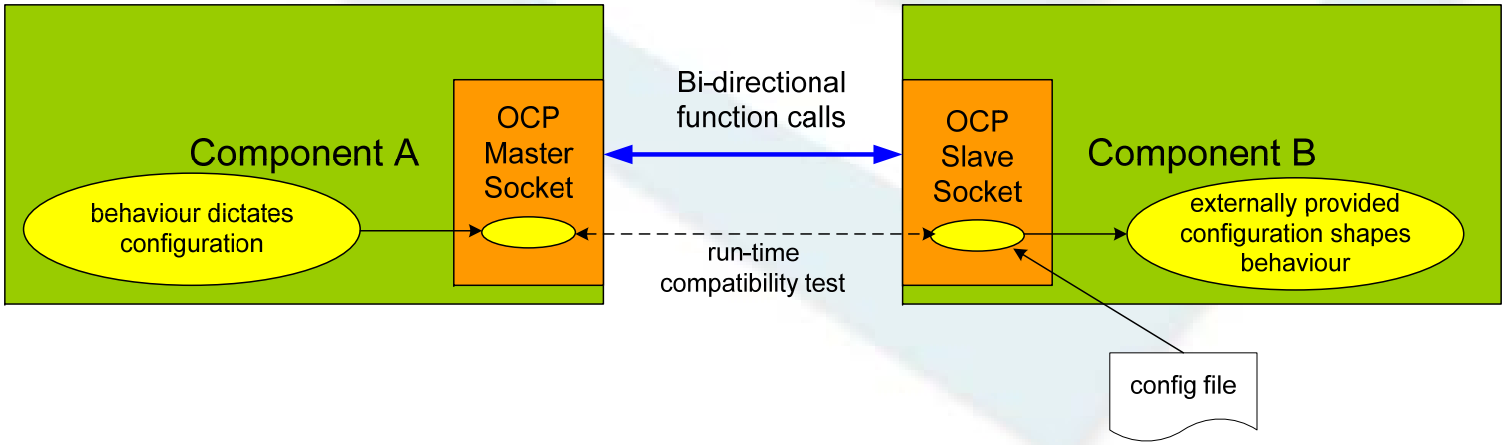
Standard transport observer (monitor)

Same concepts as used in “GreenSocket”



Socket Bindability

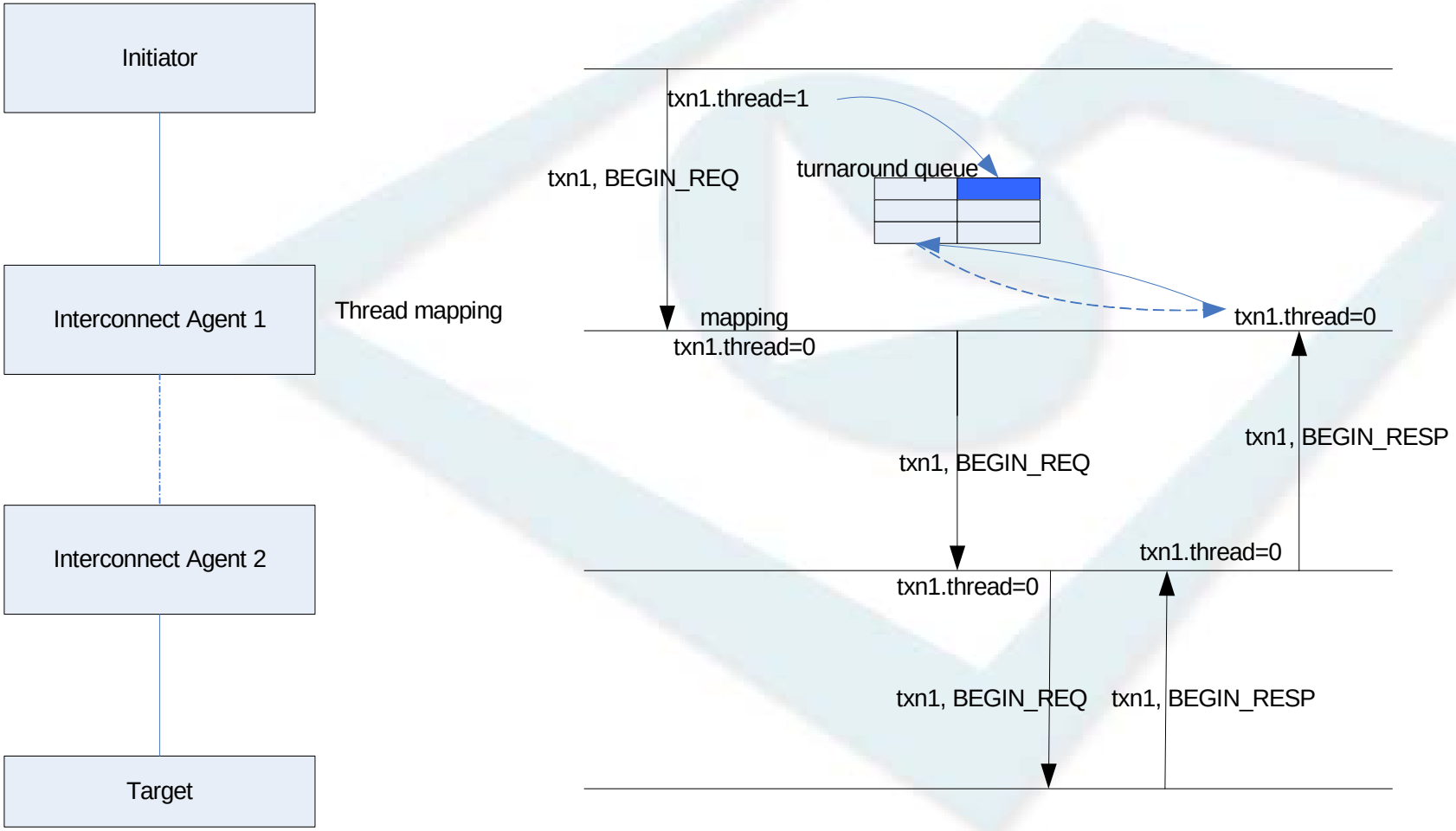
- OCP TLM Sockets test bindability at elaboration time
  - OCP configuration parameters for master and slave are compared
    - binding is rejected for incompatible components
    - a generic slave may inherit its configuration from the master (or v-v)
  - No direct binding between abstraction levels
  - Direct binding from OCP-IP TL3/4 to OSCI TLM 2.0 Base Protocol
    - provided OCP configuration does not exceed BP functionality



A Stake in Methodology

- TLM is a shift, especially with multi-phased modelling
- GreenSocket + GP extensions + TLM extended phases
  - Comprehensive documentation
  - Allow to bridge TLM to OCP terminology
  - Follow Base Protocol
- “Convenience API”
  - Functionality developed from assembling concrete platforms
    - Examples, monitors, layer adapters
    - Member contributions
      - Sonics has interest and experience in TL1
  - Becomes Methodology Layer

Methodology example: Attribute Mutability



A Stake in Methodology: Practical Additions

- Transaction invariant
  - Applicable at TL1/TL2/TL3
  - Captures all OCP extensions representing transaction data
  - Instance specific extension
  - Added at first sight in nb_transport
- Transaction tracking
  - Applicable at TL1 and TL2
  - Track a phase as soon as it is received (nb_transport)
- Timing guards
  - Applicable at TL1
  - Ensures correct evaluation times for all protocol timing arcs

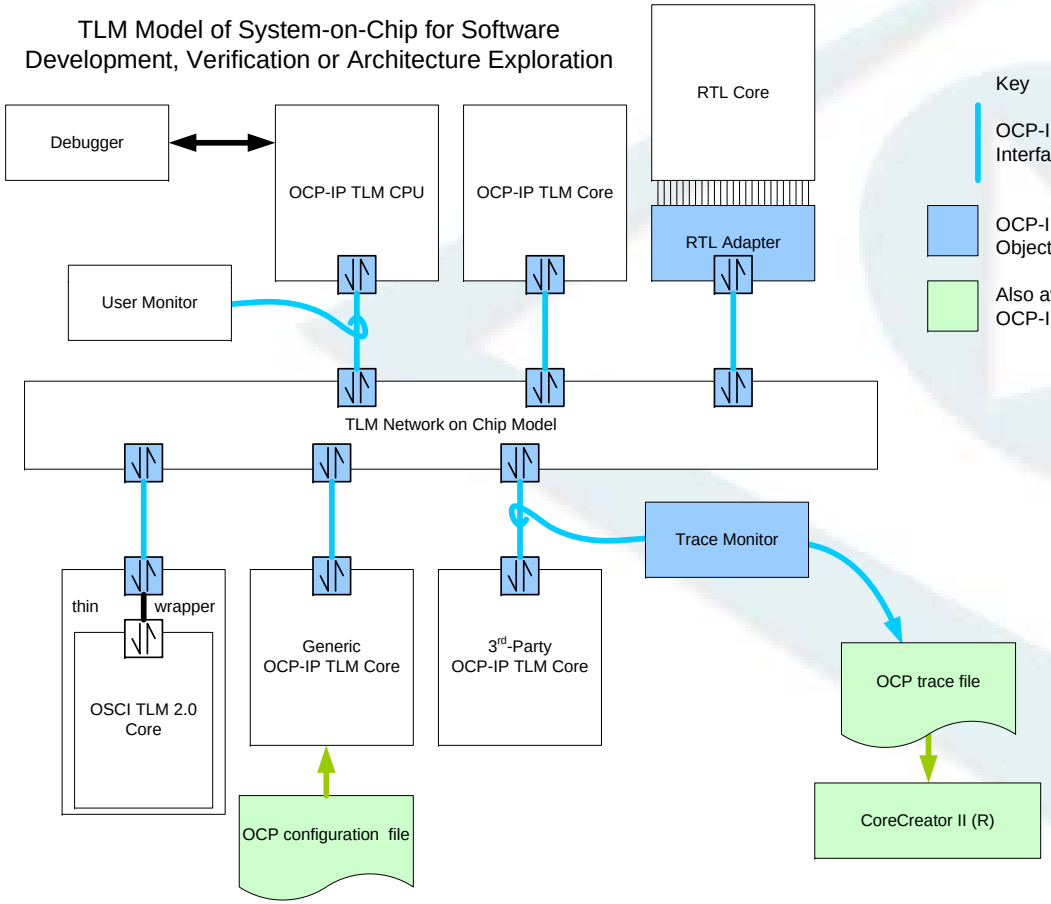
Transaction Invariant & Tracking

```
tlm::tlm_sync_enum
my_module::nb_transport( tlm::tlm_generic_payload& txn, tlm::tlm_phase& ph,
                         sc_core::sc_time& tim) {
    if ( ph == tlm::BEGIN_REQ ) {
        ocpip::ocp_txn_burst_invariant* p_inv = ocpip::check_ispec_extension
            <ocpip::ocp_txn_burst_invariant>( txn, m_acc );
        if ( p_inv == NULL ) {
            first_sight( txn );
            txn.acquire();
        }
        ocp_txn_position* p_pos = ocpip::require_ispec_extension<ocp_txn_position>( txn, m_acc );
        p_pos->req_position = m_burst_tracker[p_inv->threadid].track_phase( txn, tlm::BEGIN_REQ );
    }
    ...
}

void my_module::first_sight( tlm::tlm_generic_payload& txn ) {
    ocpip::ocp_txn_burst_invariant* p_inv = m_invariant_ext_pool.create();
    ocpip::ocp_txn_position*        p_pos = m_position_ext_pool.create();

    *p_inv = ocpip::ocp_txn_burst_invariant::init_from( txn, m_ocp_params );
    m_acc(txn).set_extension( p_inv );
    m_acc(txn).set_extension( p_pos );
}
```

Wrap-up



- OCP Modelling Kit exploits all of TLM-2.0
  - Generic Payload
  - Extension Mechanism
  - Timing Annotation
  - Base Protocol
- OCP has added to TLM-2.0
  - Extensions
  - Run-time compatibility testing
  - Technology for increased timing accuracy
- Available NOW

More information:

Technical Article: www.chipdesignmag.com

OCP: www.ocpip.org
Datasheet: http://www.ocpip.org/socket/datasheets/OCP_TLM_Datasheet.pdf
Code Download: http://www.ocpip.org/systemc_download

GreenSocs: www.greensocs.org