OCP Sockets for TLM-2.0
Using OSCI-TLM-2.0 to Model a Real Bus Protocol at Multiple Levels of Abstraction

Hervé Alexanian, Sonics
Mark Burton, Robert Günzel, Greensocs
James Aldis, Texas Instruments
Work Context

- OCP open point to point configurable protocol for SOC integration
- **OCP-IP** promotes and supports OCP
  - Functional verification specifications
  - Verification tools: BFMss and protocol checkers
  - Parameter capture formats
  - RTL timing classes
  - Analysis and debug tools
- **System-Level Design Group**
  - Standard interfaces for SystemC modeling
  - Most involved: GreenSocs/Texas Instruments/Sonics
OCP-IP’s Existing SystemC Infrastructure

- SystemC infrastructure available from OCP since 2003
- Widely used: 100s of downloads, many users
- Maintained up-to-date with latest OCP protocol version
- Largely based on SystemC-2.0 technology
- Supported by EDA vendors
OCP-IP does not provide a SOC interconnect model

- Strictly point-to-point
- Different layers of abstraction
- Network-on-chip vendors provide compatible models of their IPs
Next Generation OCP-IP SystemC Infrastructure

- OSCI TLM-2.0 Adoption
  - The OCP-IP technology has some disadvantages
    - Difficult to bridge to other bus technologies
    - Costly to maintain because of complete infrastructure
  - OCP-IP owns the abstraction level definitions
  - OCP can draw great benefits from:
    - a generic memory-mapped bus payload
    - definitions of abstraction levels
  - OCP should build on top of TLM-2.0
    - Using extension mechanism
    - should be faster, cleaner, easier to bridge

- Existing OCP-IP SystemC technology is supported through backward-compatibility adapters
# OCP-IP SystemC Next Generation Interface Standards

<table>
<thead>
<tr>
<th>OCP-IP SystemC Interface</th>
<th>OSCI TLM compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>TL0</td>
<td>Not specified by OCP-IP separately for SystemC from other HDLs</td>
</tr>
<tr>
<td>TL1</td>
<td>OCP-IP TL1</td>
</tr>
<tr>
<td>TL2</td>
<td>OCP-IP TL2</td>
</tr>
<tr>
<td>TL3/TL4</td>
<td>OCP-IP TL3/TL4</td>
</tr>
</tbody>
</table>

TL4
## OCP-IP Abstraction Levels

<table>
<thead>
<tr>
<th>Timing Accuracy</th>
<th>Abstractions</th>
<th>OSCI equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TL0</strong> Cycle accurate</td>
<td>None, this is the RTL level</td>
<td>SystemC synthesise-able subset</td>
</tr>
<tr>
<td><strong>TL1</strong> Can be fully cycle-accurate, requiring clock synchronisation between bus master and bus slave, and respecting the OCP protocol. All beats of a burst are modelled.</td>
<td>Wires and signals are not modelled</td>
<td>none so far</td>
</tr>
<tr>
<td><strong>TL2</strong> User selectable number of timing points per bus burst</td>
<td>No clock synchronisation therefore some non-determinism. Optional averaging of bus occupancy over bursts or parts of bursts. Flow control not modelled explicitly.</td>
<td>none so far</td>
</tr>
<tr>
<td><strong>TL3</strong> 4 timing points per bus burst, bus occupancy determined only by ‘data receiver’</td>
<td>No modelling of independent write data phases, no ability to model intra-burst timing effects, no distinction between address order within a burst (eg wrapping and incrementing bursts are equivalent)</td>
<td>Approx-timed (AT) nb_transport()</td>
</tr>
<tr>
<td><strong>TL4</strong> Minimum necessary to run software on a virtual platform</td>
<td>“Pure functional” representation of memory-mapped bus. No flow control or ordering effects are modelled.</td>
<td>Loosely-timed (LT) b_transport() and nb_transport()</td>
</tr>
</tbody>
</table>
Why Isn’t OSCI TLM-2.0 Enough?

- Generic Payload and Base Protocol provide a memory-mapped bus API with 100% interoperability……but
  - It is functionally limited
    - simple addressing modes
    - no semaphores or bus locking
    - etc
  - It only offers two levels of abstraction
    - loosely-timed (~ OCP-IP TL4)
    - approximately-timed (~ OCP-IP TL3)
- Nevertheless
  - For many bus interfaces, TLM-2.0 *is* enough
    - the majority of IP cores have simple bus interfaces
  - At higher levels of abstraction, distinctions between OCPs or OCP and other protocols disappear
  - **OCP-IP interfaces will be compatible with TLM-2.0 Base Protocol wherever possible**
OCP-IP Socket...

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

  Protocol negotiation to cover all OCP’s

  Memory management

  Safe handling of time

  Same concepts as used in “GreenSocket”
### Layered Structure of the Interfaces

<table>
<thead>
<tr>
<th>Payload</th>
<th>TL4</th>
<th>TL3</th>
<th>TL2</th>
<th>TL1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>OSCI Generic Payload</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extensions</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>bus locking</td>
<td>re-ordering</td>
<td>non-blocking flow</td>
<td>user extensions</td>
</tr>
<tr>
<td></td>
<td>semaphores</td>
<td></td>
<td>control</td>
<td>bit-mapping</td>
</tr>
<tr>
<td></td>
<td>exotic addressing</td>
<td></td>
<td>posted writes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>modes</td>
<td></td>
<td>in-burst address</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>order (wrap)</td>
<td></td>
</tr>
<tr>
<td>Other</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Features</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>run-time compatibility</td>
<td></td>
<td>source bandwidth</td>
<td>clock synchronisation</td>
</tr>
<tr>
<td></td>
<td>testing</td>
<td></td>
<td>signalling</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>split data and command phases</td>
<td>combinational</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>transaction chunking</td>
<td>dependencies</td>
</tr>
</tbody>
</table>

- 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.
Generic Payload Extensions for OCP

- All levels of abstraction (pure functional)
  - read exclusive/Read linked-write conditional
  - addressing modes (block bursts, wrapping, xor)
- TL3 and below
  - threads/tags (out-of-order responses for pipelined transactions)
- TL2 and below
  - writes with early or no response
  - non-blocking flow control
  - data handshake phases
- TL1
  - timing specification for combinatorial paths
- Approximately 15 extensions are derived from the OSCI extension base class for OCP
  - more will be required in the future as OCP grows
  - for any given OCP configuration only a subset are required
  - in many cases none are required
OCP Payload Extension Examples

tlm::tlm_generic_payload* txn = master.get_transaction();
// fill generic payload
...
...

// single request, non posted write
master.validate_extension<nonposted>( *txn );
master.validate_extension<srmd>( *txn );

// on thread 3
master.get_extension<threadid>( *txn )->value = 3;
Increased Timing Accuracy

- TL1 and TL2 are more accurate than can be supported by TLM-2.0 Base Protocol
  - OCP-IP will provide some technology for interoperability with higher timing accuracy than offered by TLM-2.0
  - TL1 is fully cycle-accurate
    - timing points for every beat of a burst for master and slave
    - clock synchronisation rules
    - timing information exchange for managing combinatorial dependencies
  - TL2 provides a user-selectable level of accuracy
    - transactions may be broken into smaller “chunks”
    - data “creation rate” may be specified dynamically without needing to model every beat of a burst
    - no requirement to be clock-synchronised so some inevitable limit to attainable accuracy
OCP TL1: Phase Extensions

Master Socket allocate txn

N1 clock cycles

N3 clock cycles

Slave Socket

nb_transport_fw( txn, BEGIN_REQ, SC_ZERO_TIME)

TLM_UPDATED(END_REQ, SC_ZERO_TIME)

nb_transport_fw( txn, BEGIN_DATA, SC_ZERO_TIME)

nb_transport_bw( txn, END_DATA, SC_ZERO_TIME)

nb_transport_fw( txn, BEGIN_DATA, SC_ZERO_TIME)

TLM_UPDATED(END_DATA, SCZERO_TIME)

nb_transport_bw( txn, BEGIN_RESP, SC_ZERO_TIME)

nb_transport_bw( txn, END_RESP, SC_ZERO_TIME)

N2 clock cycles

N4 clock cycles

OCP Burst Write: MBurstLength=2, MBurstSingleReq=1
OCP TL1: Phase Extensions

- OCP specifies phase ordering rules
- Does not match OSCI BP
  - SRMD Read Burst:
    - 1 BEGIN_REQ/END_REQ phase
    - N BEGIN_RESP/END_RESP phases
- Data handshake phase: BEGIN_DATA/END_DATA
- Thread busy phase
  - Singleton transaction maintained by socket
  - nb_transport THREAD_BUSY_CHANGE phase
- Additional rules
  - Example: Thread busy flow control must end phase with TLM_UPDATED return
OCP TL2: Intra Burst Timing

Master Socket

allocate txn

word count=4

N1 * clockperiod

word count=4

Slave Socket

nb_transport_fw( ms.timing, TL2_TIMING_CHANGE)

TLM_ACCEPTED

nb_transport_fw( txn, BEGIN_REQ, SC_ZERO_TIME)

TLM_UPDATED (END_REQ, 10 ns)

nb_transport_fw( txn, BEGIN_REQ, SC_ZERO_TIME)

TLM_UPDATED (END_REQ, 10 ns)

nb_transport_bw( txn, BEGIN_RESP, SC_ZERO_TIME)

TLM_UPDATED( END_RESP, 20 ns)

N2 * clockperiod

word count=8

OCP Burst Read: MBurstLength=8, MBurstSingleReq=0
OCP TL2: Intra Burst Timing

- Burst Granularity
  - Custom “word_count” extension
  - Rules are derived from OCP phase ordering
    - For example, a slave may not send a response with a
      word_count greater than the aggregate request
      word_count received
- Burst Timing Approximation
  - Per-beat timing
    - If master sends request with word_count=4 and per-beat
      delay is 10 ns, slave should process in 4*10 ns.
    - Timing may change with singleton transaction and
      TL2_TIMING_CHANGE phase
Socket Bindability

- OSCI TLM-2.0 proposes a compile-time mechanism for testing compatibility
  - OCP needs something more sophisticated
    - there are too many OCPs (1000s) to have a traits class for every one
    - inter-OCP compatibility rules are too sophisticated: can not divide OCPs into disjoint sets of mutually compatible
    - direct binding to OSCI Base Protocol ought to be possible for TL3 with appropriate configuration
    - future plans include OCP sockets that can adapt to the abstraction level at bind time
  - Therefore OCP-IP will provide an elaboration-time compatibility check
    - based on exchange of OCP configuration information during binding
    - permits the SystemC models to fall back to a common configuration
    - permits creation of “generic” components that adapt their behavior to the core they are bound to
Wrap-up

- This is not only going to work
- it is going to work efficiently
- OCP can exploit all of TLM-2.0
  - Generic Payload
  - Extension Mechanism
  - Timing Annotation
  - Base Protocol
- OCP needs to add
  - Extensions
  - Run-time compatibility testing
  - Technology for increased timing accuracy
More information:

OCP: www.ocpip.org

GreenSocs: www.greensocs.org