TLM-2.0 in Action: An Example-based Approach to Transaction-level Modeling and the New World of Model Interoperability

John Aynsley, Doulos
The LRM, or Hardcore TLM-2.0

CONTENTS

• Memory management
• Base protocol phases
• Exclusion rules and early completion
• Timing annotation
• Ignorable phases
Generic Payload

-Do not construct/delete generic payload objects repeatedly
-Do use a transaction pool
-A generic payload memory manager is a good way to do this

```cpp
tlm::tlm_generic_payload* trans;
sc_time delay;
...
trans = m_mm.allocate();
trans->acquire();
...
init_socket->b_transport( *trans, delay );
...
trans->release();
```
Non-blocking Transport

```
enum tlm_sync_enum { TLM_ACCEPTED, TLM_UPDATED, TLM_COMPLETED };
```

```
template < typename TRANS = tlm_generic_payload,
            typename PHASE = tlm_phase >

class tlm_fw_nonblocking_transport_if : public virtual sc_core::sc_interface {
  public:
    virtual tlm_sync_enum nb_transport_fw( TRANS& trans, PHASE& phase, sc_time& t ) = 0;
};
```

- `nb_transport_fw` called from initiator to target
- `nb_transport_bw` called from target to initiator
- Trans, phase and time arguments set by caller and modified by callee
- Phase is local to each socket-to-socket hop
Base Protocol - Phases

- Request and response exclusion rules

Initiator: BEGIN_REQ → END_REQ → BEGIN_RESP → END_RESP

Target: Request accept delay
Latency of target
Response accept delay

Copyright © 2008-2009 by Doulos. All rights reserved.
Causality

Initiator sets attributes

Initiator Target

BEGIN_REQ

Interconnect Interconnect

END_REQ

BEGIN_RESP

Target

Target modifies attributes

END_RESP

BEGIN_REQ

Interconnect

b_transport

return

TARGET

return

END_RESP

Initiator checks response

Initiator sets attributes

Copyright © 2008-2009 by Doulos. All rights reserved.
• Only target can set response status attribute
• Initiator / interconnect / target roles are dynamic
• Address attribute only valid on first downstream call path
• Address may by overwritten by each interconnect
• Requests must reach given target in-order
• Responses may return to given initiator out-of-order
• Transport and DMI transaction routing must be the same
• DMI pointer by-passes interconnects
Early Completion

- END_RESP or TLM_COMPLETED ends a transaction over a hop
- TLM_COMPLETED not mandatory
- TLM_COMPLETED from target implies BEGIN_RESP
- BEGIN_RESP implies END_REQ
- These implicit phases play in the exclusion rules

- Transaction completes at different times on different hops
- tlm_mm_interface::free() called on final release()
Timing Annotation Argument

Temporally decoupled initiators

Initiator

+80ns +70ns +60ns

Interconnect

+40ns +30ns +20ns +90ns +80ns +70ns

Initiator

+30ns +20ns +10ns

Delay arguments added to sc_time_stamp()
Transaction Ordering

- Current time = sc_time_stamp() + delay_argument

Pseudo-code

```plaintext
wait 100ns
nb_transport_fw T1, BEGIN_REQ, 50ns
nb_transport_fw T2, END_RESP, 10ns
wait 100ns
nb_transport_fw T3, BEGIN_REQ, 20ns
```

- Current time is strictly non-decreasing for a given transaction
- Current time is normally non-decreasing from a given initiator
- Merged transaction streams can be out-of-order
- Out-of-order transactions may be executed in *any* order
- LT: execute immediately and pass on timing annotation
- AT: schedule for execution at proper time
- Exclusion rules based on call order alone
Ignorable Phases

- Extra timing points
- Only after BEGIN_REQ
- Only before END_RESP
- Cannot demand a reaction
- Sender must allow for no reaction
- Most phases are not ignorable
  - so use a new protocol traits class
• IEEE 1666
  standards.ieee.org/getieee/1666/index.html

• OSCI SystemC 2.2 and TLM-2.0
  www.systemc.org

• Tutorial introduction to TLM-2.0 and Free *TLM-2.0 Protocol Checker*
  www.doulos.com/knowhow/systemc/tlm2