European SystemC Users Group

Nice 2007

# The SystemC TLM 2.0 Draft 1 Kit

**John Aynsley**

Technical Director, Doulos

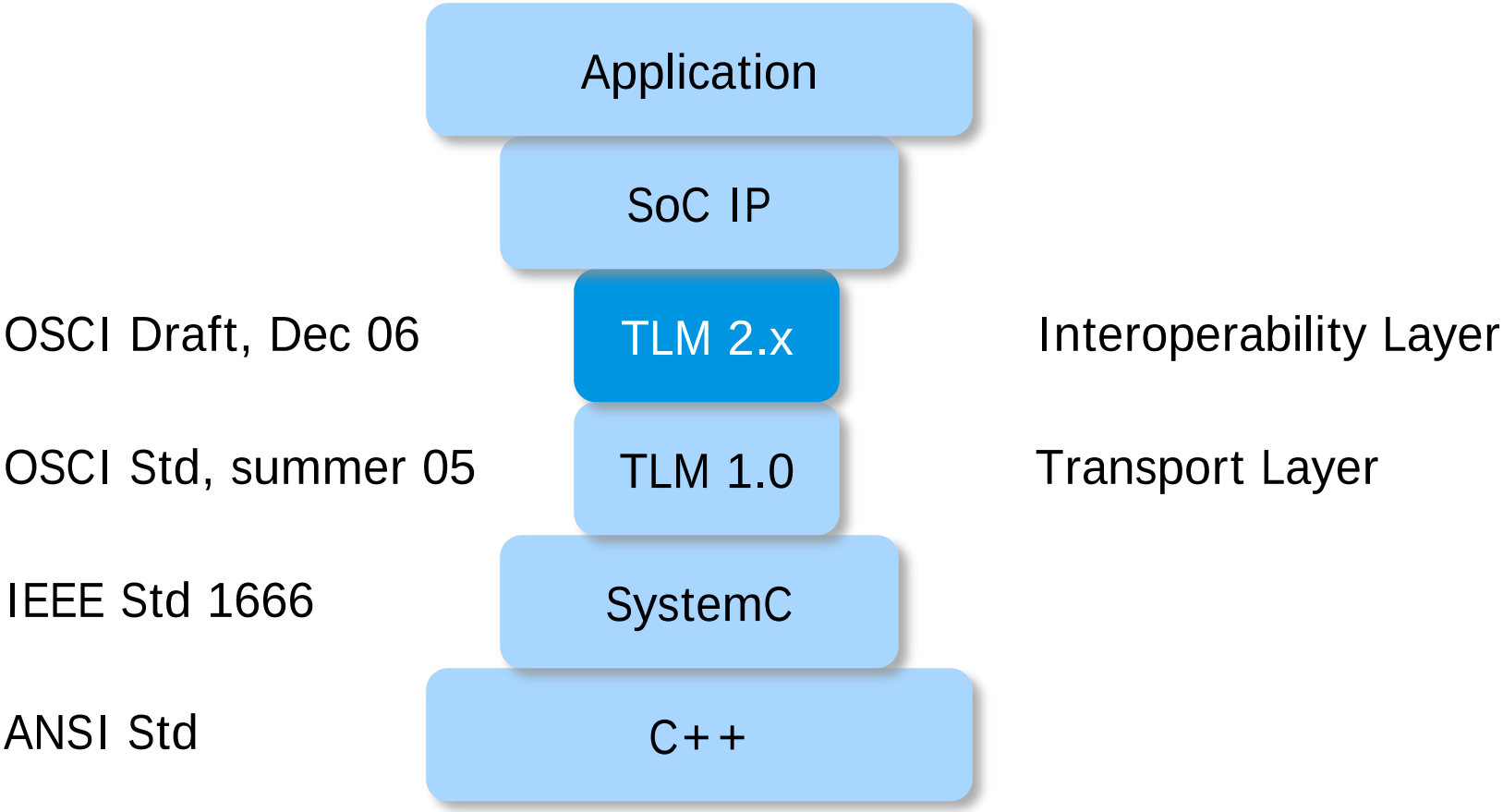
# The SystemC TLM 2.0 Draft 1 Kit

CONTENTS



DOULOS

# Layered Standards



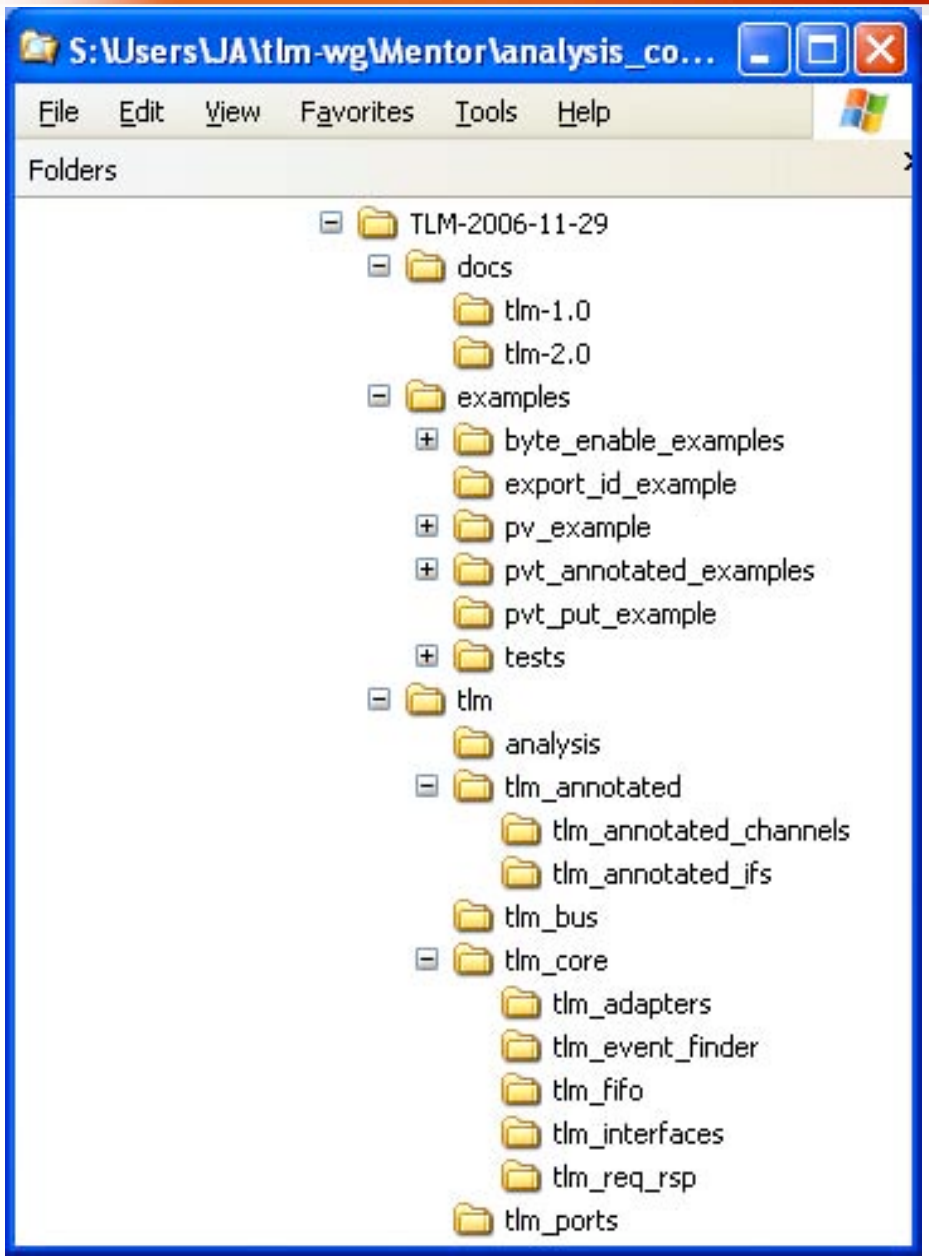
# OSCI TLM 2.x Agenda

- Timing annotation on core interfaces
- Generic payloads for memory-mapped busses
- Analysis ports

Draft 1

- Cycle accurate modeling
- Debug interface
- Configuration interface
- Interrupt modeling
- Memory map and register modeling

# TLM 2.0 Draft 1 Kit




+ TLM_2.0_Overview.pdf

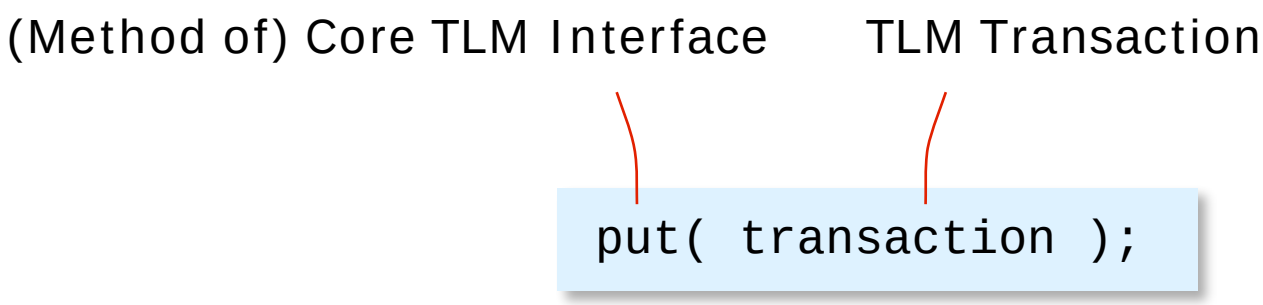
History
Empty

TLM2.0 examples
Includes Powerpoint

TLM2.0
TLM2.0
TLM2.0
TLM2.0

TLM1.0
TLM1.0
TLM1.0
TLM1.0 +
TLM1.0
TLM2.0

# Fundamental Principle of TLM 2.0 draft 1




- TLM transaction only valid for duration of function call

  (system transaction sliced-and-diced into independent chunks) …

  … except for pass-by-pointer mode

# The SystemC TLM 2.0 Draft 1 Kit

CONTENTS



# Payloads for Memory-Mapped Busses

- Generic payloads tlm_request and tlm_response
- Use for generic PV and PVT modeling whenever possible
- Use standard interpretation for fields whenever possible
- Use custom extension mechanism only when necessary
- Use as starting point for specific protocol implementation


# tlm_request

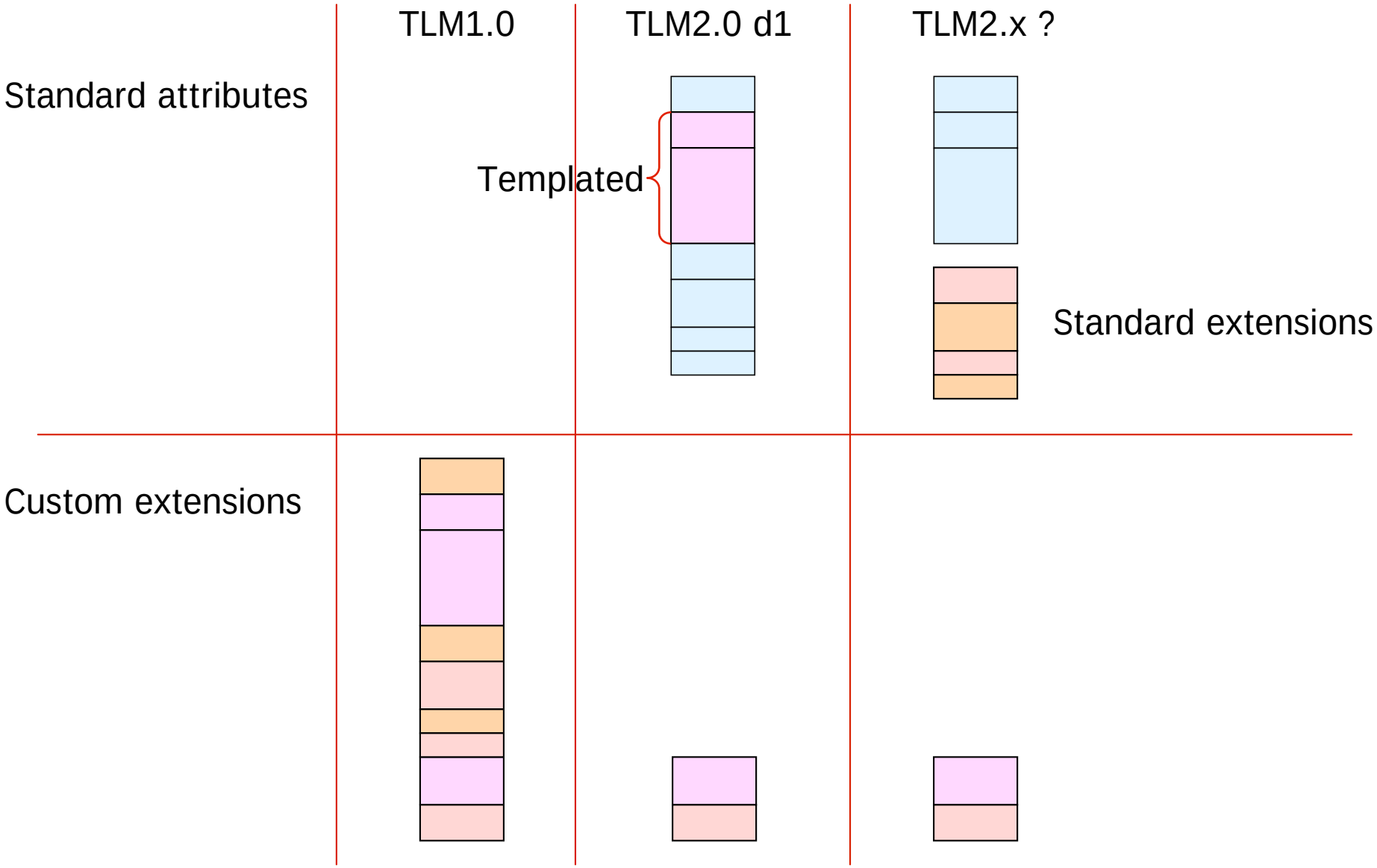
```
template<typename ADDRESS, typename DATA, ...>
class tlm_request { ...
  tlm_command          m_command;
  tlm_mode             m_mode;
  ADDRESS              m_address;
  DATA                *m_data;                  // deep copy
  unsigned int         m_block_size;            // size of data array
  const unsigned int *m_byte_enable;            // may be null
  unsigned int         m_byte_enable_period;    // size of byte en array
  tlm_block_mode       m_block_mode;
  unsigned int         m_block_address_incr;    // bytes-per-word
  unsigned int         m_priority;
  unsigned int         m_master_thread_id;
  unsigned int         m_transaction_id;
  unsigned int         m_tlm_export_id;         // identifies target export
  std::vector<tlm_custom_base*>
                      *m_custom_vector_ptr; };  // custom extensions
```

# tlm_response

```
template<typename DATA, ...>
class tlm_response { ...
protected:
  DATA              *m_data;
  unsigned int       m_block_size;
  tlm_status         m_status;
  unsigned int       m_priority;
  unsigned int       m_master_thread_id;
  unsigned int       m_transaction_id;
  unsigned int       m_tlm_export_id;
  std::vector<tlm_custom_base *>
                     *m_custom_vector_ptr;
};
```

Public get/set methods

# Approaches to Interoperability



# The SystemC TLM 2.0 Draft 1 Kit

CONTENTS

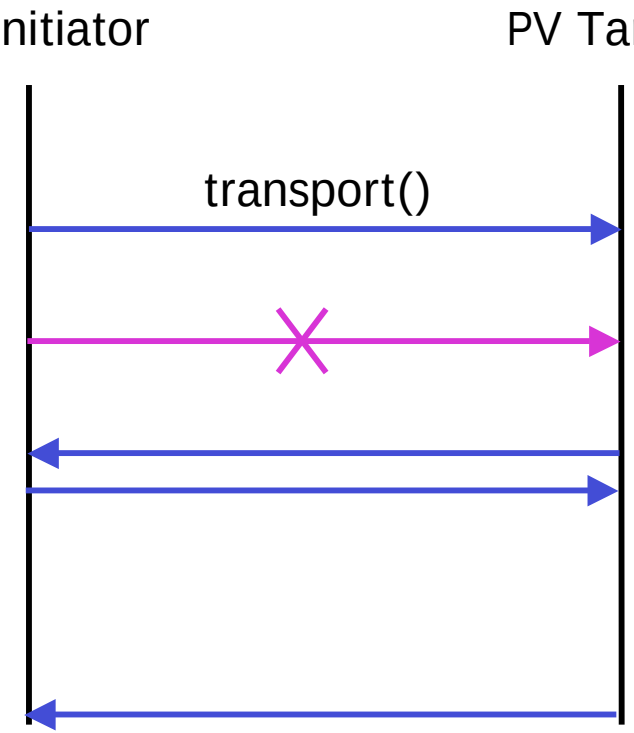


DOULOS

# TLM 2.0 Core Interfaces

| | Untimed | Timed |
|---|---|---|
| Bidirectional Blocking | void transport(<br>const REQ&, RSP&); | void transport(<br>const REQ&, RSP&, sc_time&); |
| Unidirectional Blocking | void put(const T&);<br>void get(T&);<br>void peek(T&); | void nb_transport( const REQ&, RSP&);<br>*rsp includes delay or event* |
| Unidirectional Non-Blocking | bool nb_put(const T&);<br>bool nb_can_put();<br>sc_event &ok_to_put();<br><br>bool nb_get(T&);<br>bool nb_can_get();<br>sc_event &ok_to_get();<br><br>bool nb_peek(T&);<br>... | bool nb_put(const T&, const sc_time&);<br>bool nb_can_put(const sc_time&);<br><br>bool nb_get(T&, const sc_time&);<br>bool nb_can_get(const sc_time&); |

# Transport Interface with Explicit Timing



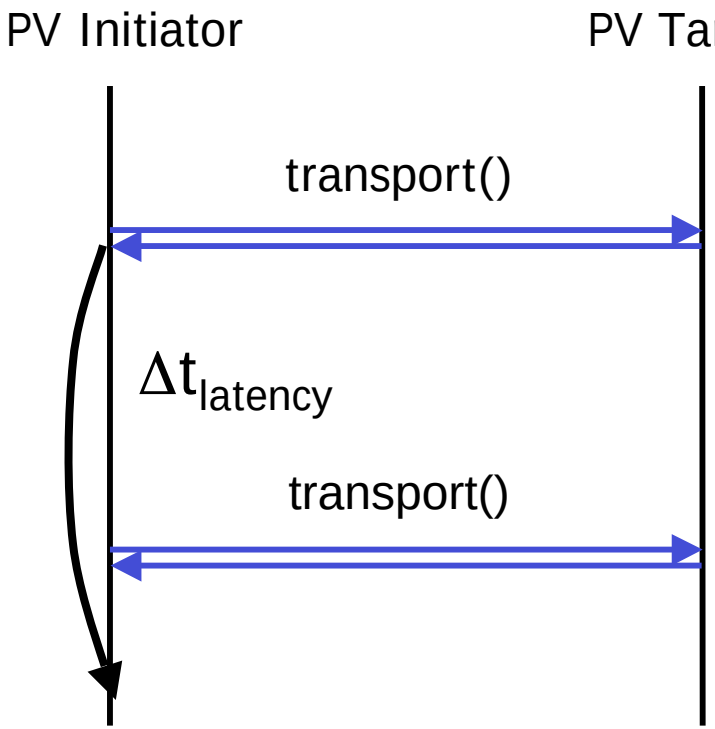

PV target model

```
void transport(const REQ& rq, RSP& rp)
{
  // do processing
  // ...
  unsigned int latency = ...;
  wait(latency * clk_period);
  rp.get_status().set_ok();
}
```

- Initiation interval ≥ latency

*Slide derived from TLM2.0 kit*

# Transport Interface with Timing Annotation




PV target model

```
void transport(const REQ& rq, RSP& rp
               sc_time& latency)
{
  // do processing ...

  double lat = ...;
  latency = lat * clk_period;

  rp.get_status().set_ok();
}
```

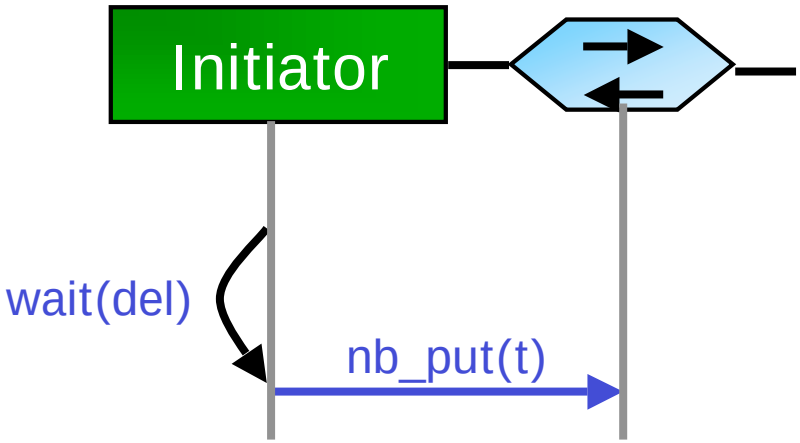
- Benefits:
  - Not calling wait => fast
  - Flexible - initiation interval < latency
  - Defers realization of timing

*Slide derived from TLM2.0 kit*

# Explicit versus Implicit Timing

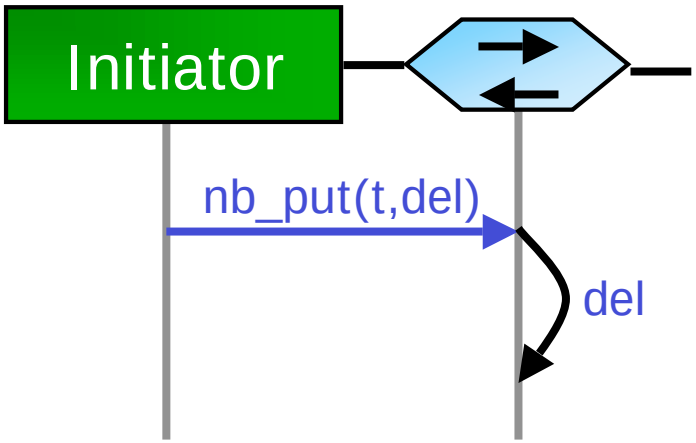
Compare

```
wait(del);
nb_put(t);
```



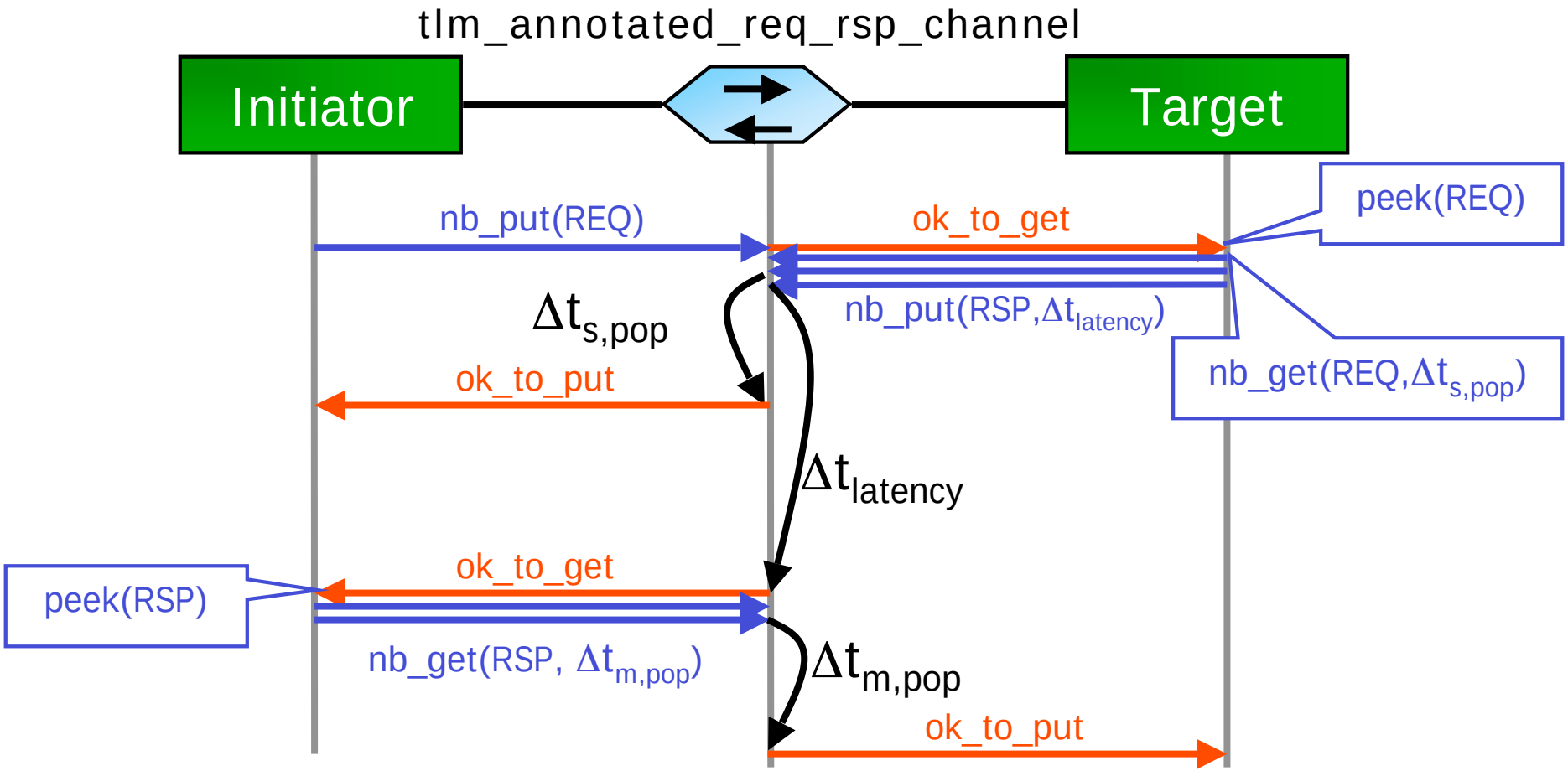

Conceptually,
a delay in the initiator

```
nb_put(t,del);
```




No context switch!

# Put/Get Interface with Timing Annotation




*Slide derived from TLM2.0 kit*

# The SystemC TLM 2.0 Draft 1 Kit

Introduction

Generic Payloads

Pass-by-Pointer

Timing Annotati

TLM 2.0 Draft 1 for Public Review

Download from www.systemc.org

DOULOS