Virtual Platforms and Timing

Brian Bailey Consulting

Email: brian_bailey@acm.org
Tel: 503 632 7448
Cell: 503 753 6040
Web: brianbailey.us

Time is an illusion

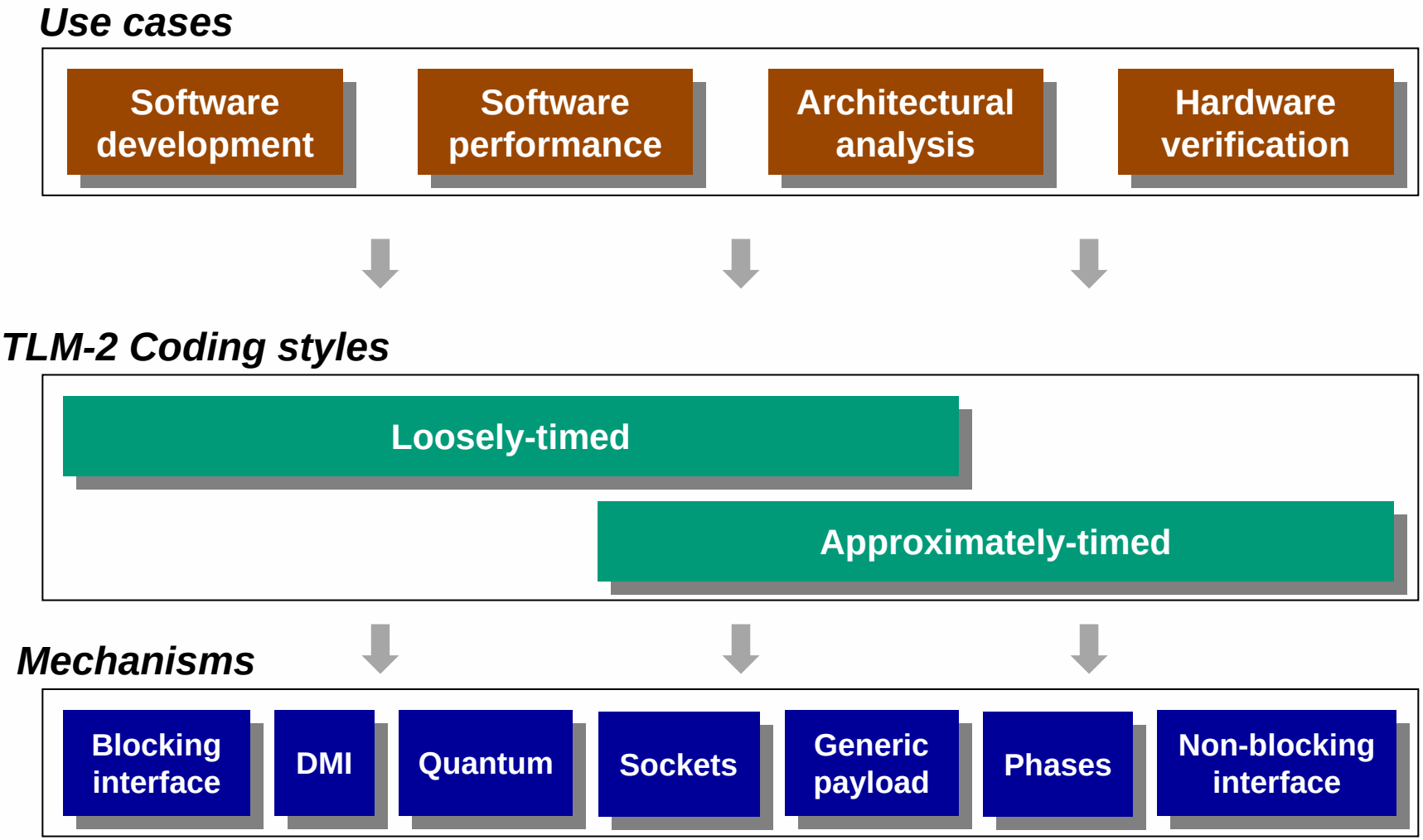
- Time comes from architectural elements
- Time is the most complex aspect of a simulation environment
- Time slows everything down

- Use time wisely
  - A tradeoff between performance and accuracy
  - Performance enables
    - Real life scenarios
    - Statistical analysis

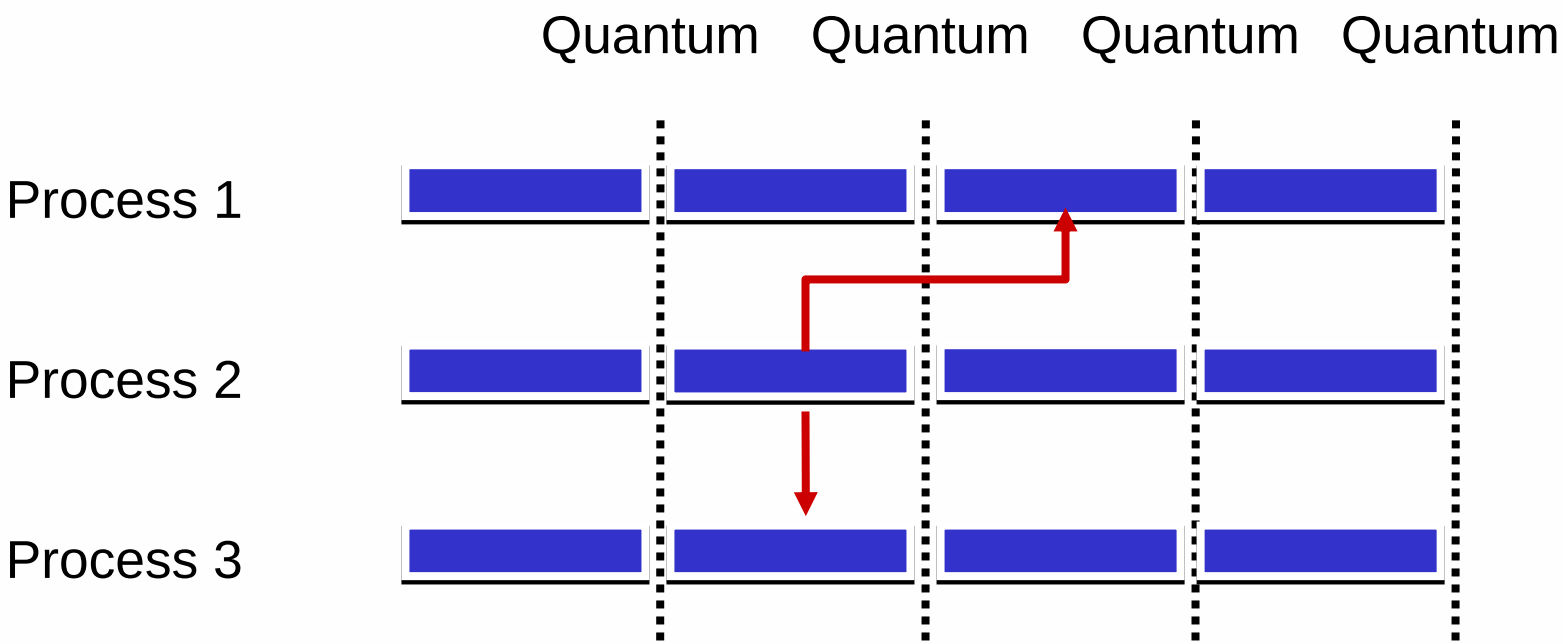
Platform Convergence

- **Virtual platforms should have been the killer app of ESL**
  - **But they were proprietary**
  - **There was a lack of models**
  - **No interoperability**
- **Along came SystemC and TLM 2.0**
  - **Far from perfect**
  - **Enough to solve interoperability**
  - **Enough to bring about convergence**
  - **Attempts to relieve performance bottlenecks**

Use Cases, Coding Styles and Mechanisms



Loosely-timed

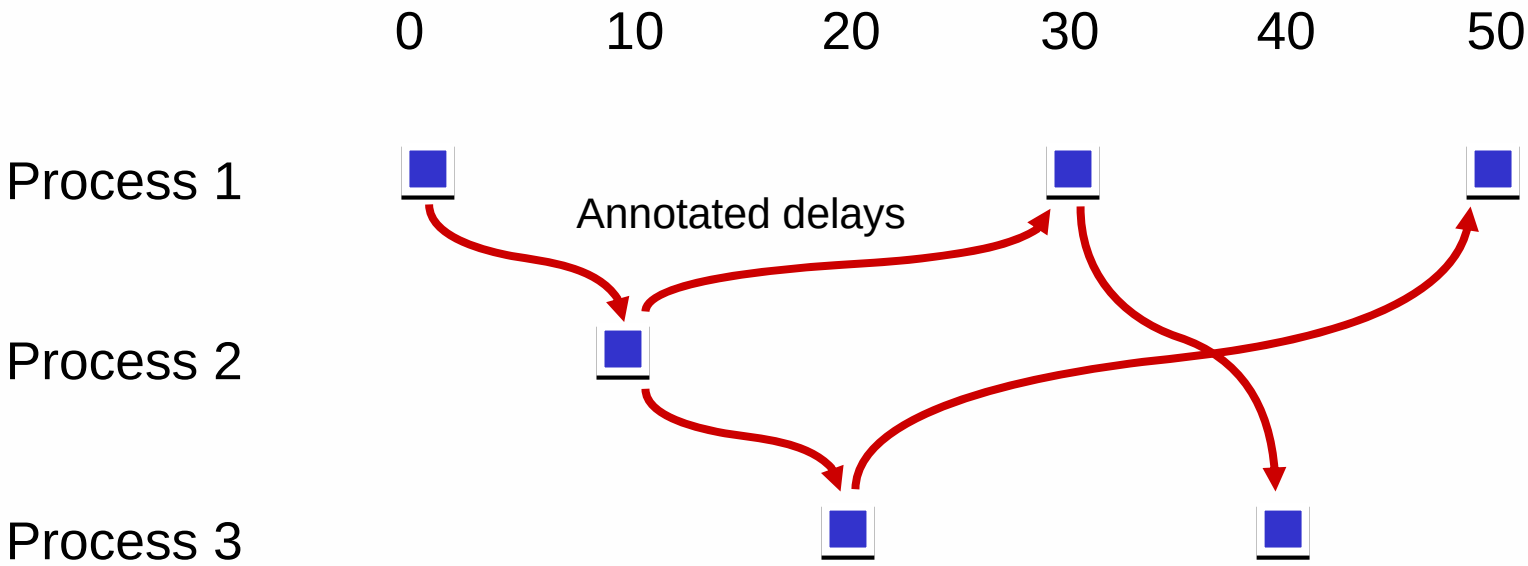


Each process runs ahead up to quantum boundary

sc_time_stamp() advances in multiples of the quantum

Deterministic communication requires explicit synchronization

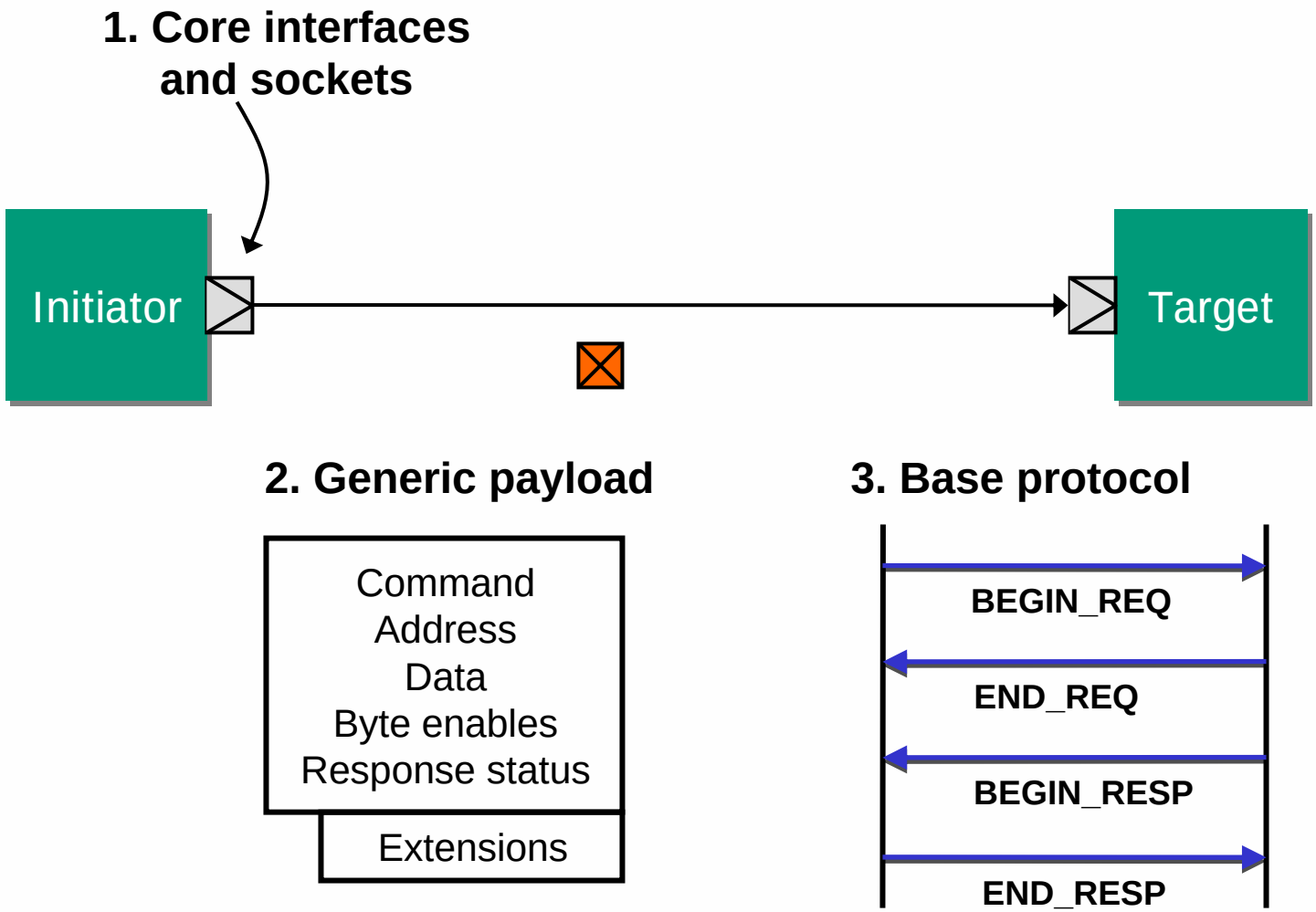
Approximately-timed



Each process is synchronized with SystemC scheduler

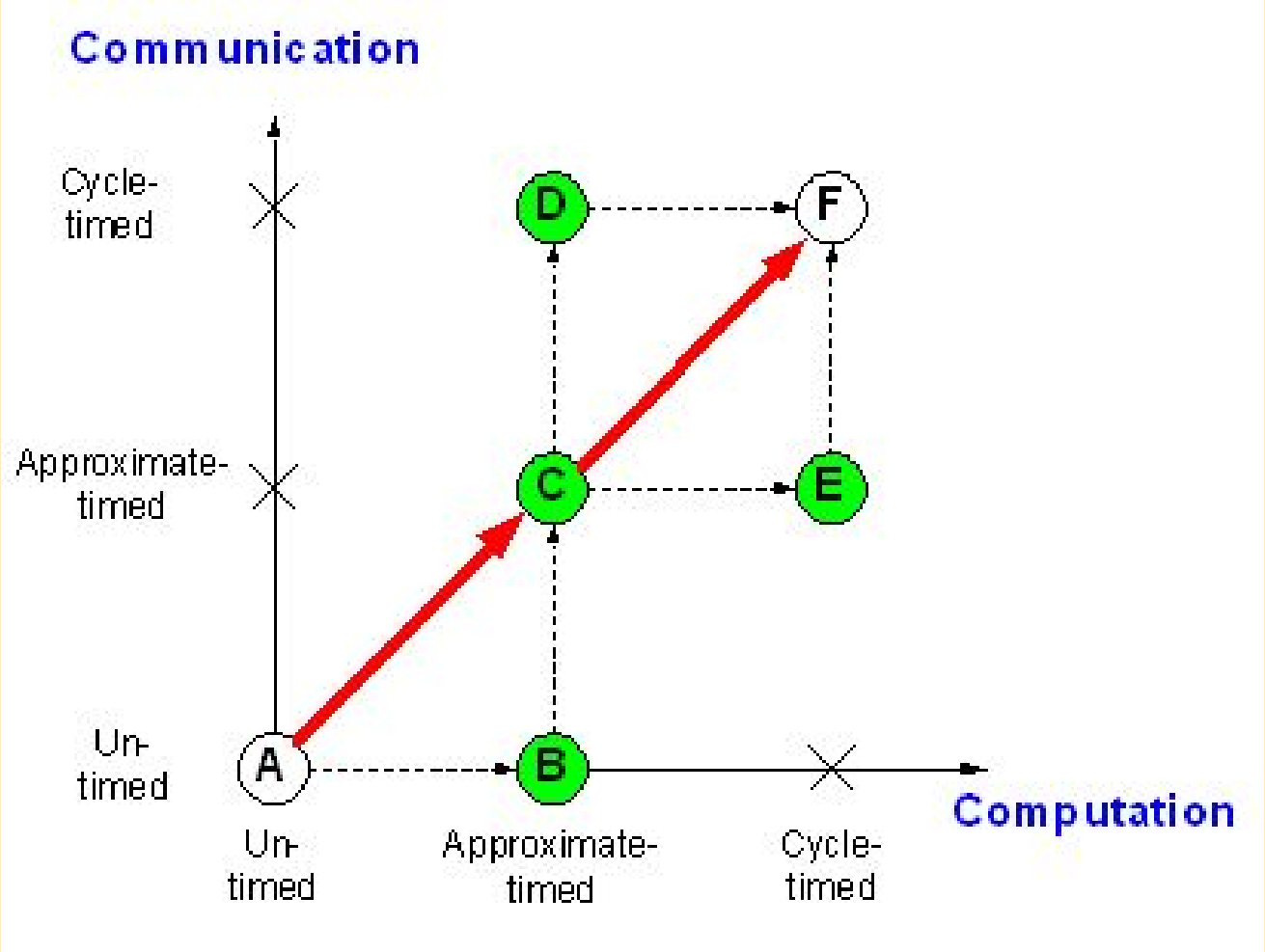
Delays can be accurate or approximate

Interoperability Layer

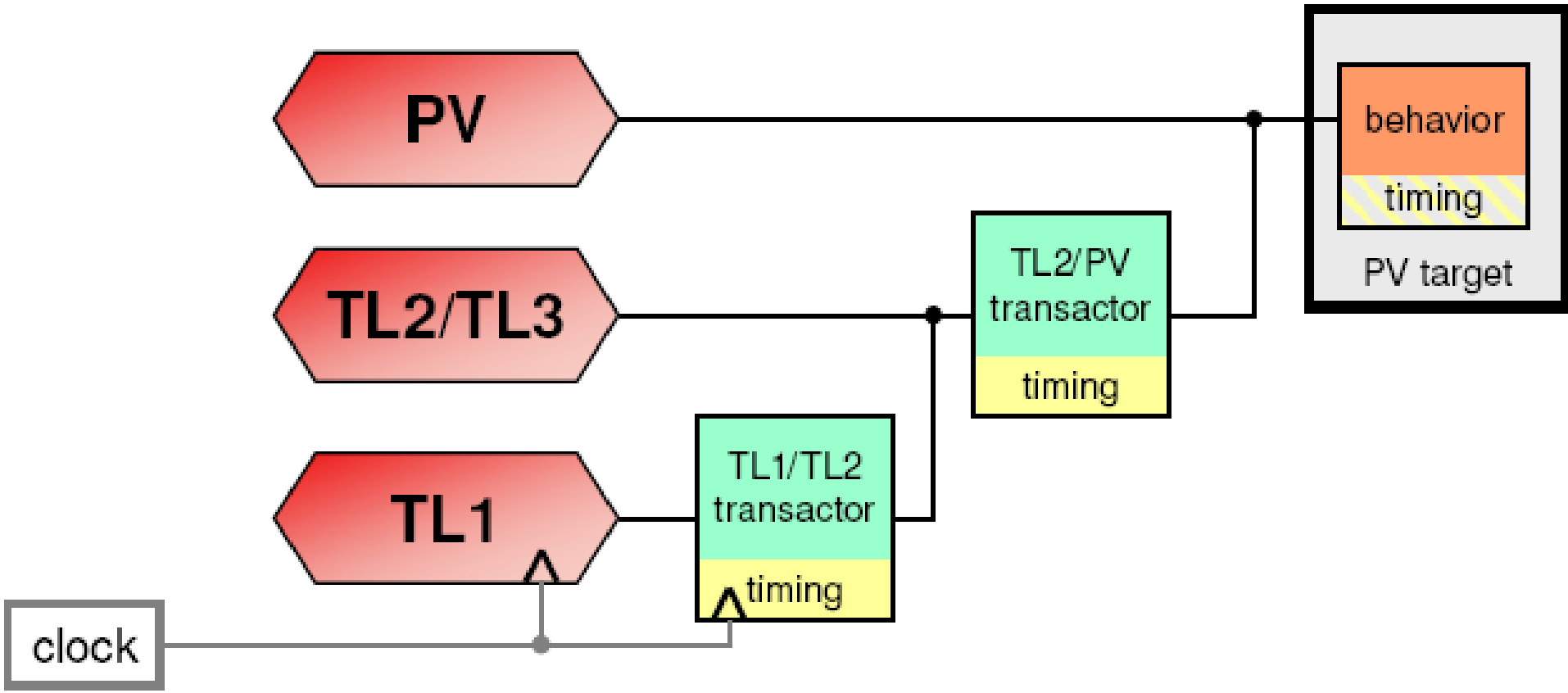


Maximal interoperability for memory-mapped bus models

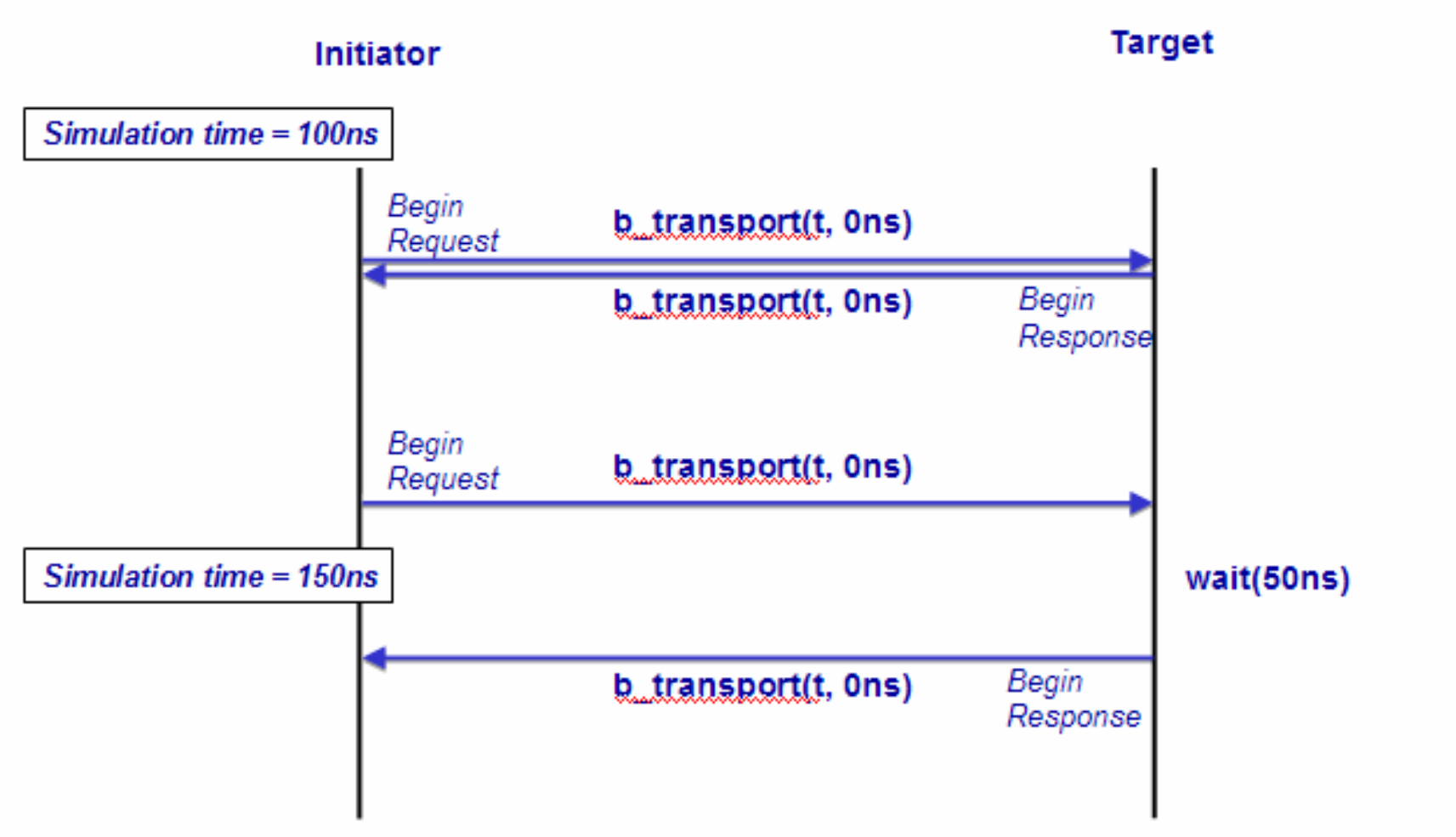
Platform refinement



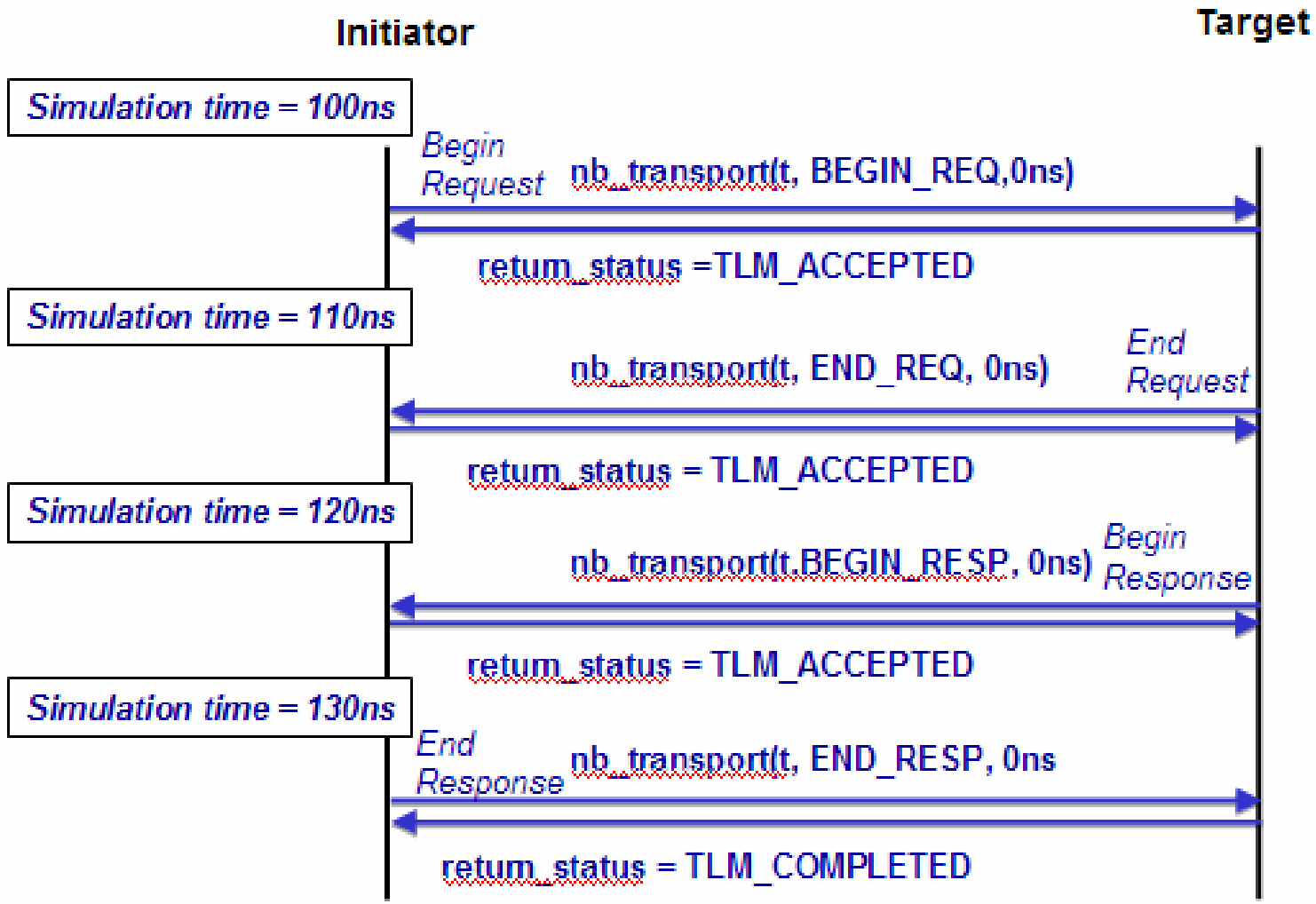
Gajski System Modeling Graph



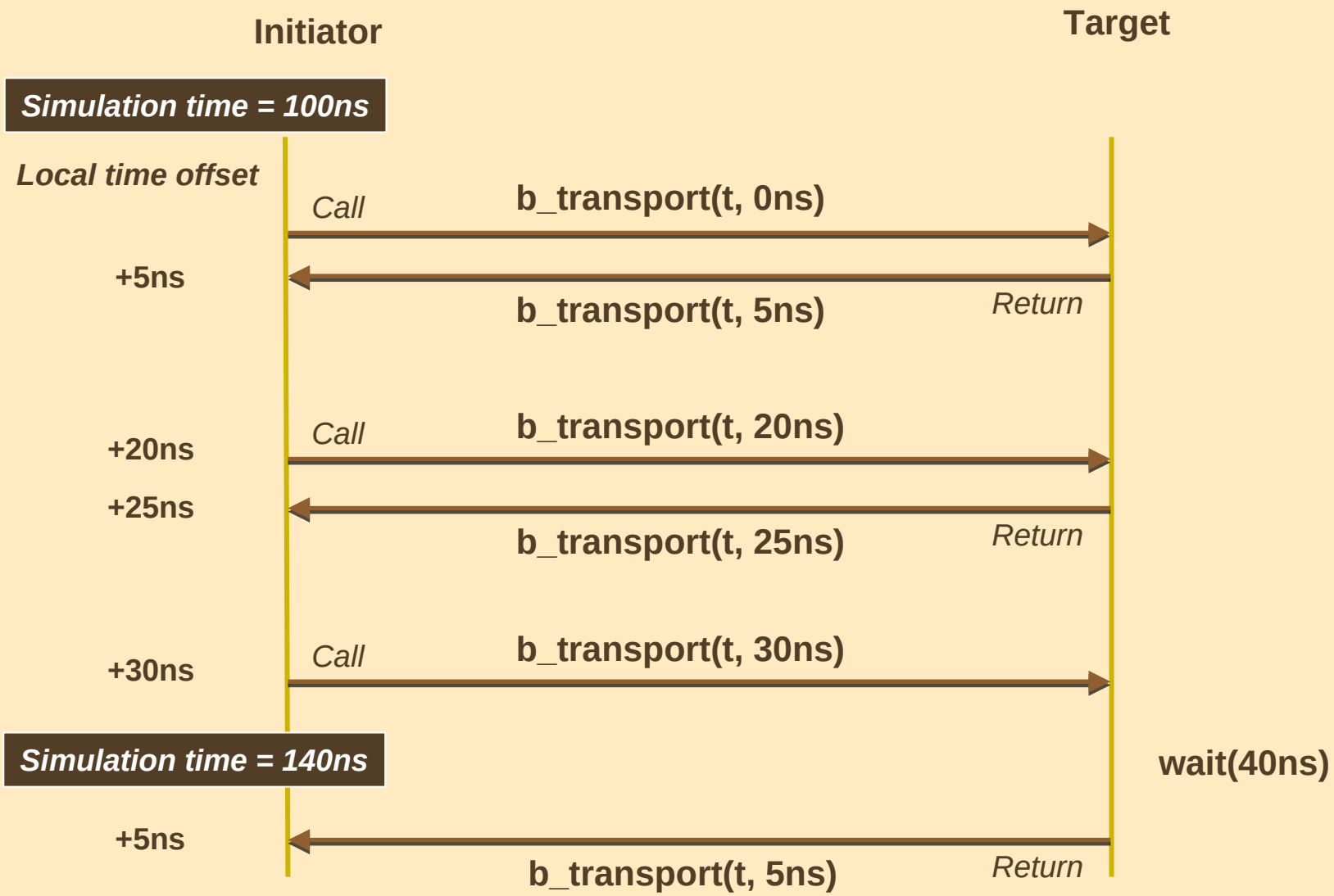
Timing annotation in TLM - LT



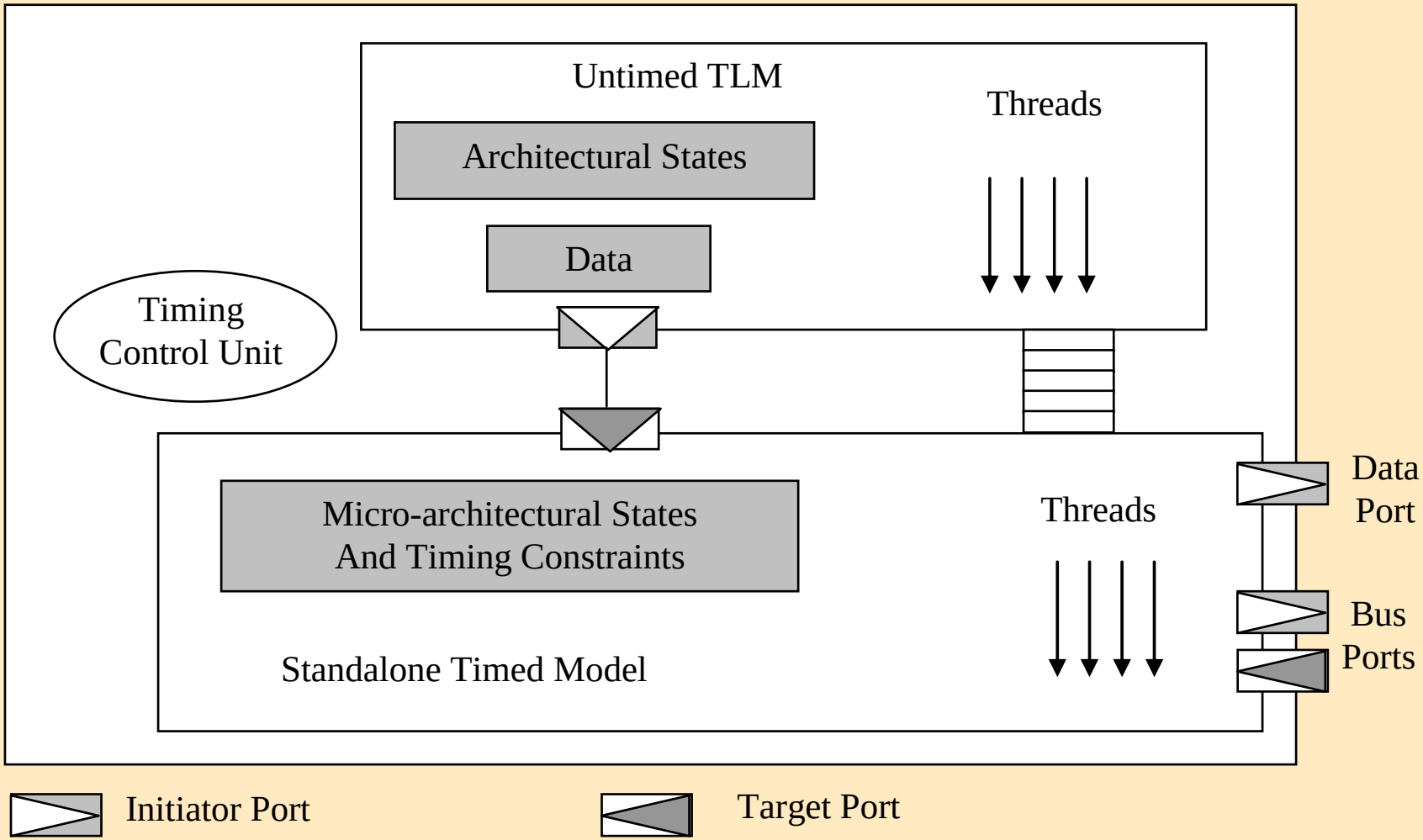
Timing Annotation in TLM - AT



Timing Annotation in TLM

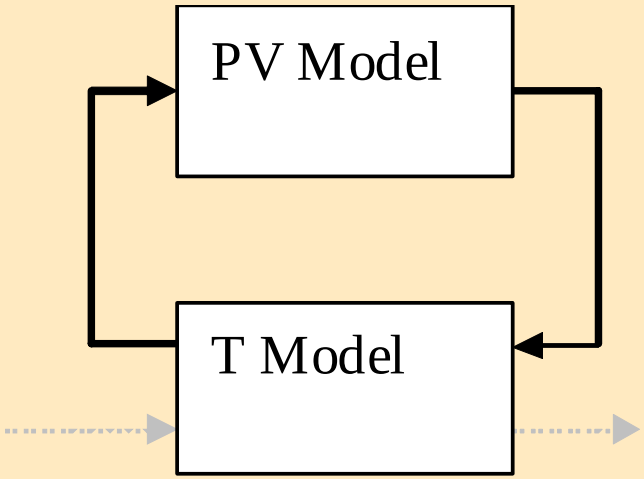


Model Separation

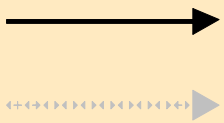


Frank Ghenassia (Ed.) Transaction-Level Modeling with SystemC. Springer 2005

A different approach

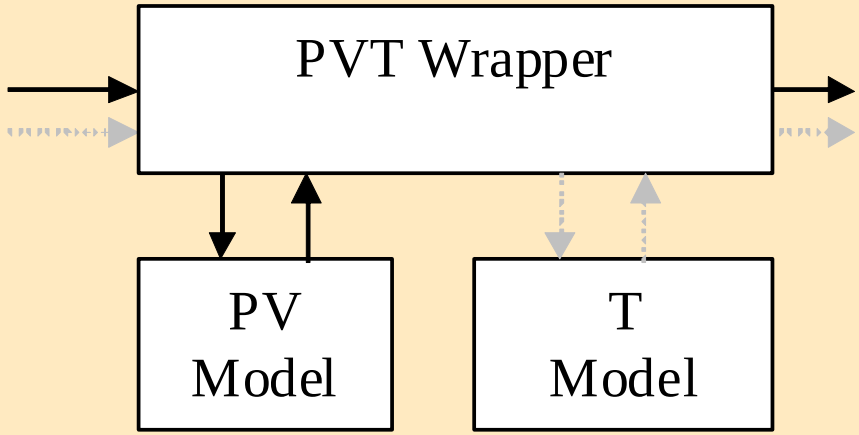


Intuitive structure



PV

T



A better structure

Getting it right

- There appears to be universal agreement that getting timing right is:
  - Time consuming
  - Difficult
  - Non-intuitive
- Most approaches have difficulty with
  - Modeling pipelining
  - Quickly changing micro-architecture

Declarative Timing

- **Being used by Mentor's Vista product**

```
data_delay_policy ON slave WAIT STATES 2
// The latency of read and write transactions on port slave is 2 clocks
split_policy ON master BURST SIZE bufferSize BUS ahb LATENCY 3
// The traffic on the master bus (read and write) is done through AHB bus policy
//    with maximal bursts of size bufferSize (a parameter of the model)
//    and 3 clocks delay between bursts
sequential_policy CAUSE slave.write EFFECT master.read DELAY 20
// The master read starts after the end of a register write in 20 clocks
sequential_policy CAUSE slave.write EFFECT end_write_int.write
    LATENCY 0
// The end-write interrupt occurs after a register write
pipeline_policy CAUSE master.read EFFECT master.write DATA DELAY
    pipelineSize LATENCY 0
// The master writes are pipelined to the master reads after a pipelineSize data
// delay and zero clocks delay (pipelineSize is a parameter of the model)
```

Fir Filter example

- Functional description takes less than a page
- Timing declared declaratively – just a few lines
- Generated TLM 2.0 code
  - 1000 lines of header
  - 2500 lines of code
  - Does not take into account the routines inserted to handle bus protocols

Conclusions

- Timing is tough
  - Need accuracy
    - But not with large simulation overhead
    - Needs to be intuitive
  - Need timing to be modeled separately
    - Should not have to replicate micro-architecture
    - Allow exploration
    - Facilitate Re-use
- TLM 2.0 is a “computer” language
  - Not very good as a modeling language
  - Need solutions layered on top of it

Thank You

brian_bailey@acm.org

http://brianbailey.us