TU WIEN

Connecting SystemC-AMS models and TLM 2.0 models using the loosely-timed coding style with temporal decoupling

Institut für Computertechnik

ICT

Institute of Computer Technology

Markus Damm
Institute of Computer Technology
Vienna University of Technology

Motivation / Overview

- ... assuming we have had sufficient motivation in the previous talk(s)...
- ⇒ SystemC-AMS is good, TLM2 is good, and using them together would be good.
- But how can we bring them to work together?
- **Goal:** General TLM2 ↔ SystemC-AMS converters with intuitive and obvious conversion semantics (if possible).
- **Focus here:** Interoperability between SystemC-AMS TDF and the TLM2 loosely timed coding style using temporal decoupling.

TLM 2.0 loosely timed coding style

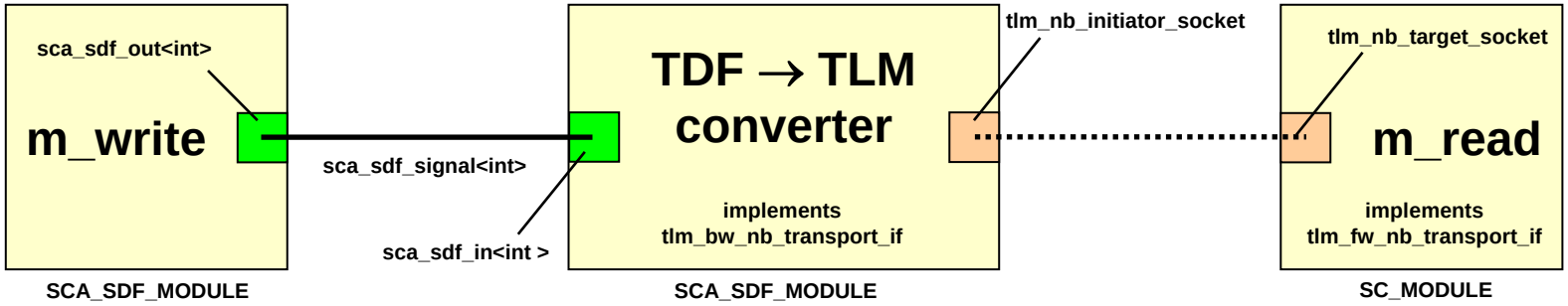
- In the loosely timed coding style, processes communicate via non-blocking method calls
- Transactions are sent by initiator processes with a **delay offset** with respect to the current simulation time
- Target processes can store these “future transactions” and process them at the right time
- Therefore, processes can **run ahead of simulation time** (“time warp”) until
  - a certain time limit is reached or
  - other processes request synchronization
- **Benefit** of this approach: Reducing context switches and therefore gaining simulation performance

SystemC-AMS TDF

- SystemC-AMS is basically *strictly* timed, so connecting with *loosely* timed models seems futile...
- ... **but** the Timed Synchronous Dataflow (TDF) Model of Computation in SystemC-AMS has also a kind of “time-warp”.
- When a source process has a output datarate > 1, it **also sends values “to the future”** with respect to the current SystemC (and SystemC-AMS!) simulation time.
- When a drain process has input datarate > 1, it **also receives “future values”** with respect to the current simulation time.
- In general, SystemC-AMS processes mostly run ahead SystemC simulation time, since the **SystemC-AMS time is always ≥ SystemC time**

⇒ There should be a way to bring these time warps together!

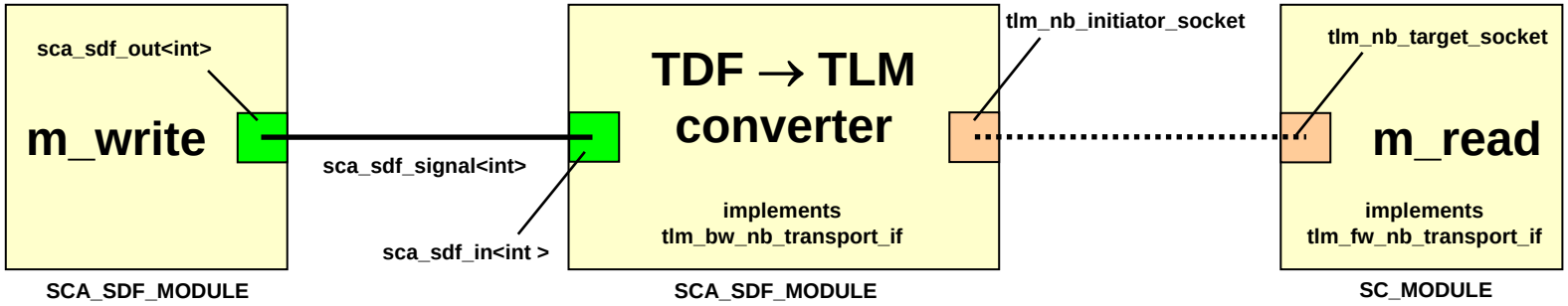
Conversion TDF → TLM



- **Basic idea:**
  - Collect the streaming TDF input, and pack junks of it into the data section of a transaction.
  - Every time its `sig_proc()` is called, the converter generates a transaction with a data array as long as its input data rate.
  - This transaction is then sent via `nb_transport` to a TLM target using as **time delay** the difference between the time of the last input token and the current SystemC time.

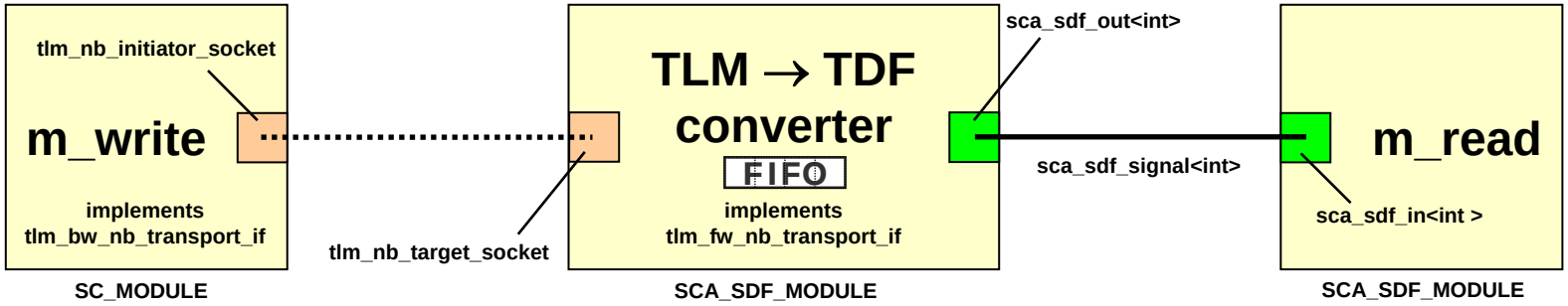
$$delay = time_{AMS} + (datarate - 1) \cdot sampletime - time_{SC}$$

Conversion TDF → TLM



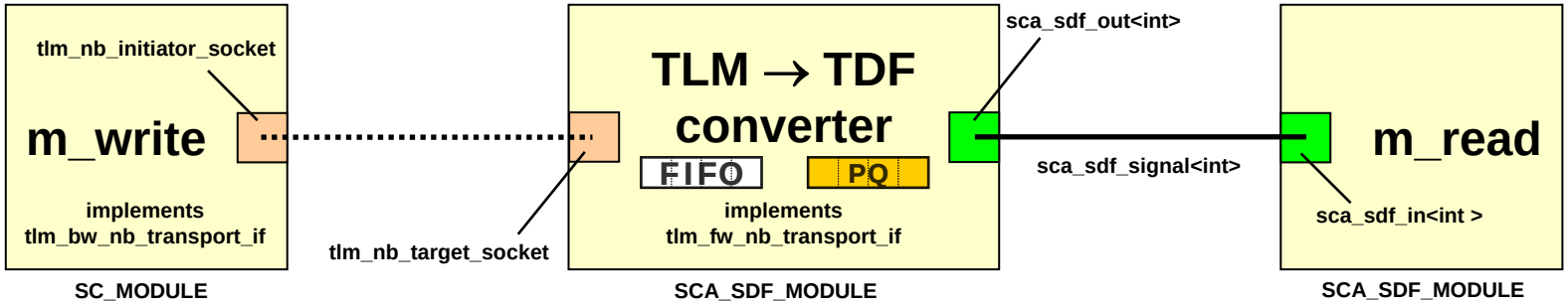
- The actual implementation allows also to set the size of the transaction data length independent from the input data rate
  ⇒ Basic approach remains the same
- **Demonstration...**
- **Problem** with this approach:
  - Only one (final) target possible for the transactions
  - Recipient address never changes
- We will revisit this conversion direction later on, after adressing the...

Conversion TLM → TDF



- Unfortunately, this is not that simple
- **Basic idea:**
  - Stream the data of the input transactions to a TDF signal
  - The transaction data are **buffered** in an internal FIFO
  - At every `sig_proc()` execution, `datarate` many values are taken from the buffer and written to the TDF output port
  - Empty buffer ⇒ send default values
  - If a transaction would cause a buffer overflow ⇒ error

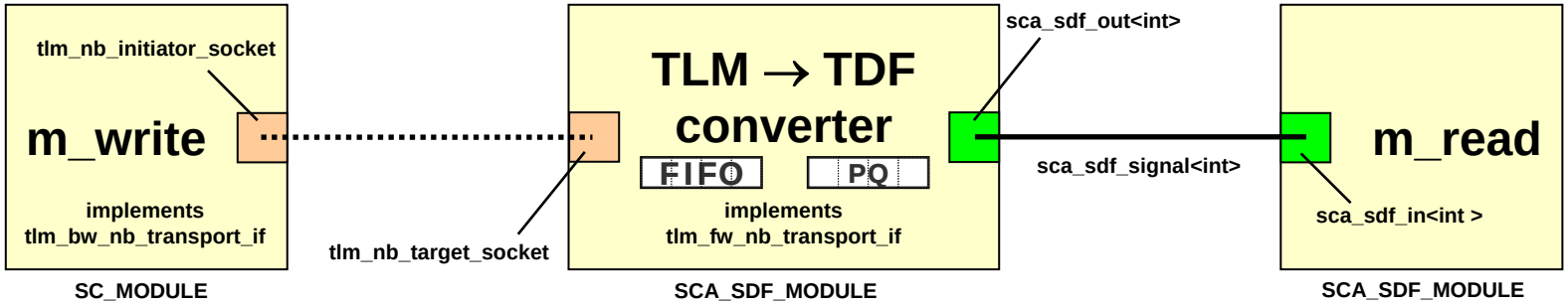
Conversion TLM → TDF



- Problems:
  - The transaction **delays may be twisted**, i.e. “later” transactions may arrive *before* “earlier” transactions

⇒ We use a **payload queue** (PQ) to store the transactions *before* writing their data to the buffer, which we only do when necessary (i.e. when the buffer has less data than the datarate).

Conversion TLM → TDF

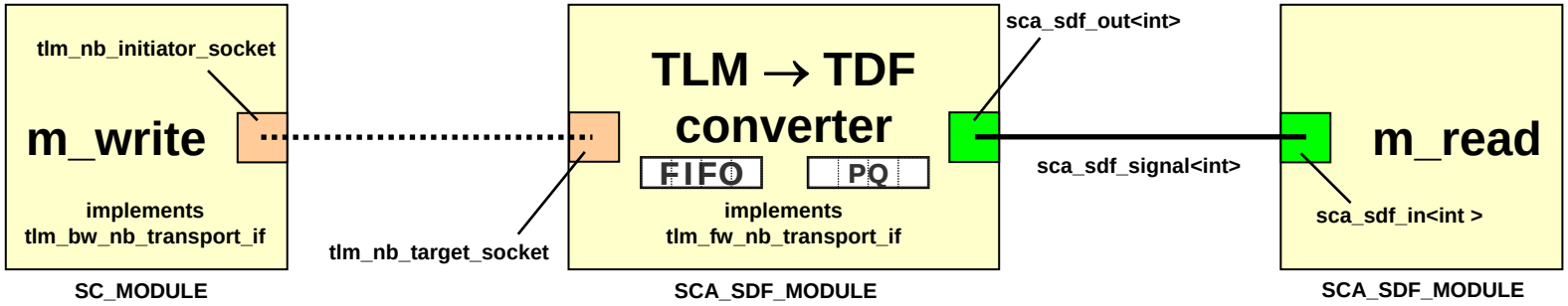


- ...more problems:
  - The **converter may run ahead** so far that the initiators might not had the chance to produce sufficient transactions to fill the buffer (even though they would).

⇒ We **call `wait(t)`** if buffer and PQ are empty with **`t`** the difference between current (local) SystemC-AMS time and SystemC-time.

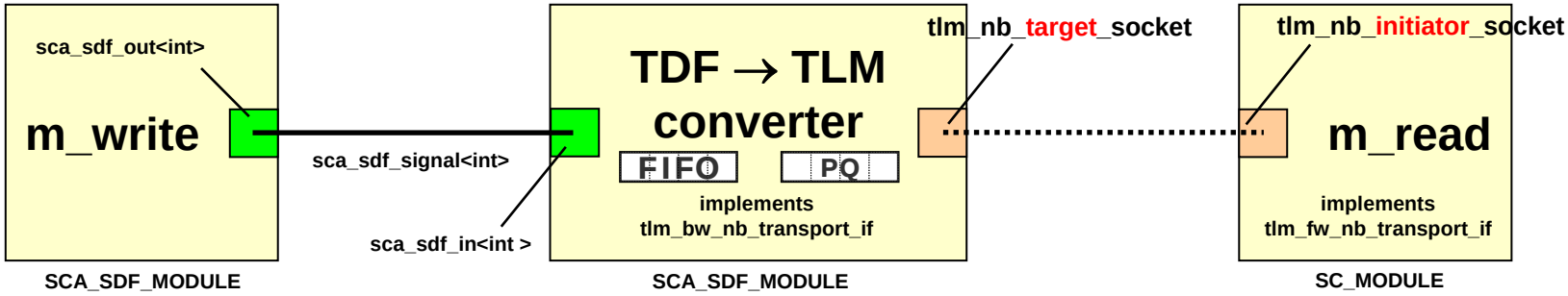
  - Works with the prototype, **but:** will the SystemC-AMS standard support this???

Conversion TLM → TDF



- ...apart from that, everything works fine!
- **Demonstration...**

Conversion TDF → TLM revisited



- As mentioned before, the proposed TDF → TLM2 converter can address only one target
- More realistic approach: The TLM side sends **`read`** transactions (requests) to the converter
- We use an internal buffer and PQ again
- The converter copies the requested data into the transaction and returns it
- Buffer overflow causes data loss

conclusion

- Connecting TLM2 and SystemC-AMS models is not only possible, but also fruitful.
- TLM2 temporal decoupling processes can work well together with SystemC-AMS statically scheduled TDF processes.
- Uncertainties remain regarding transaction acceptance / rejection

Thank you for your attention!

Your:

- questions
- comments
- ideas
- objections