Hardware/software co-simulation for conformance testing of embedded networks

Finnish-Russian University Cooperation Program in Telecommunications (FRUCT) seminar

Tampere, Finland, Oct-08

Michel Gillet

michel.gillet@nokia.com

Nokia D R&D CT ASD PSD Connectivity SD

NOKIA

Outline

- Specification
- Virtual hardware
- Demo

NOKIA

Specification

Why writing a specification?

- For legal issues
  - Patents
  - Royalties
- But mainly for interoperability
- A specification is a good step towards interoperability, but …
- Specification always have ambiguities or various interpretations

NOKIA

Type of specifications (1/2)

- Informal specifications
  - Are written in an natural language, e.g. English, Russian, Finish, etc.
  - Prone to misinterpretations
- Formal specifications
  - Are written using a formal language, e.g. SDL
  - A natural language is only explain or describe the formal part of the specification
  - Since close to a mathematical description, much less prone to misinterpretations

NOKIA

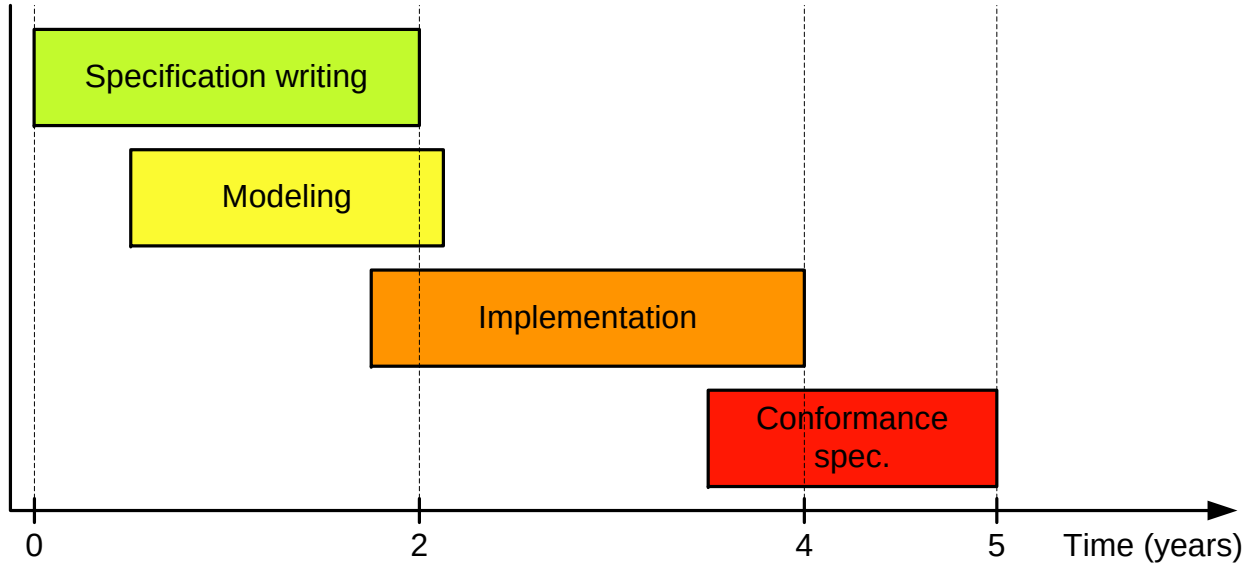
Type of specifications (2/2)

- Informal specifications
  - RFCs, USB, PCI-Express, SpaceWire, etc.
- Formal specifications
  - 3GPP, WiMAX, GSM, etc.

NOKIA

Creation timeline 1

- Four main phase
  - Writing the specification itself
  - Modeling the specification
  - Implementing it
  - Conformance tests development



Conformance vs. compliance

- Conformance is “just” an assessment made on how well an implementation respects or follows a specification
- Compliance (sometimes called certification) is a legally binding contract
  - The winner gets a nice logo
- For this, often a standardization group creates
  - not only a specification
  - but also a testing/conformance/compliance document
- This document is a set of tests, which verify the behavior on an implementation against its specification

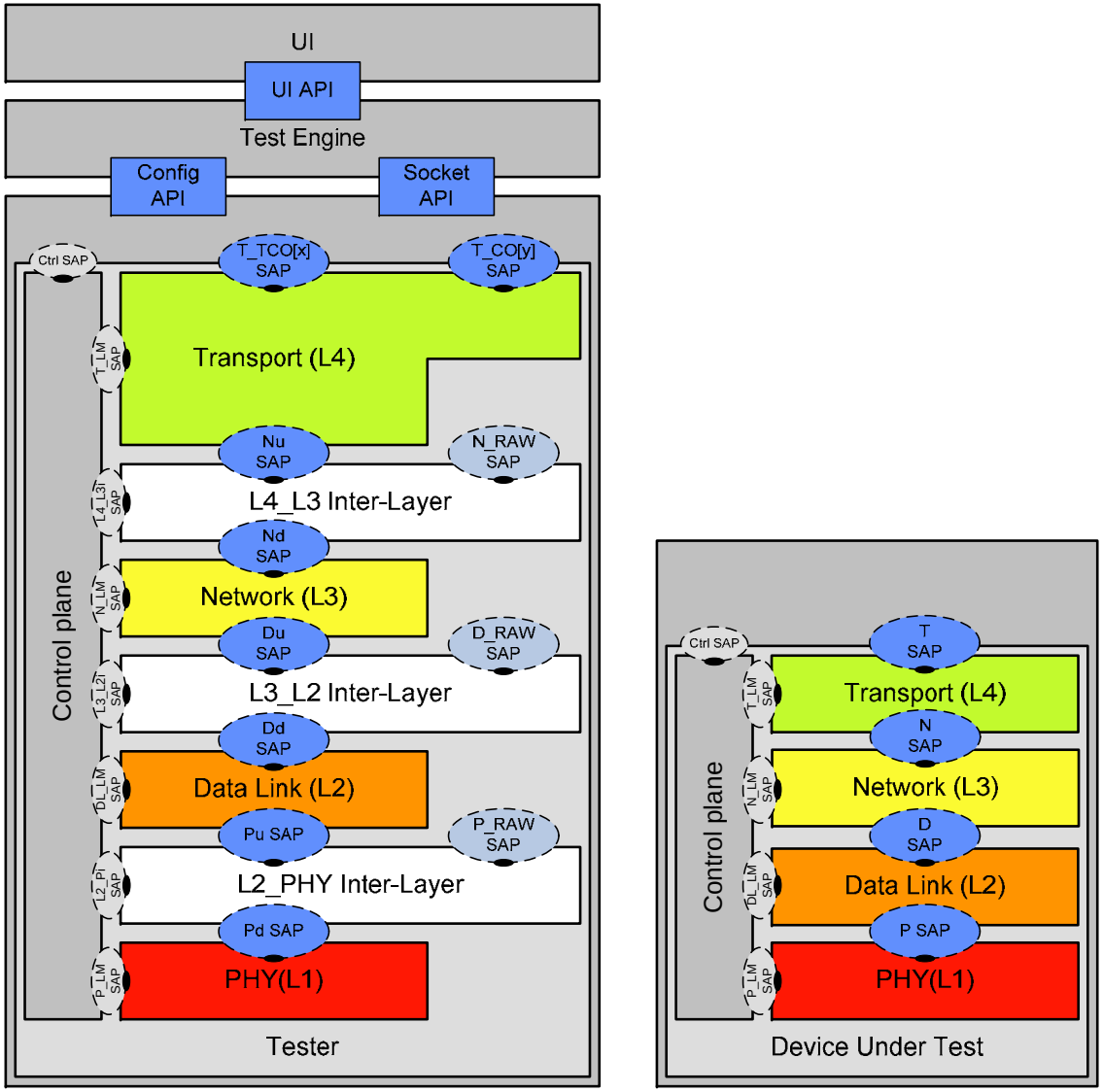
NOKIA

Type of testing document

- As for the specification itself, we have mainly 2 variants
- Informal type
  - Handwritten: “Shalls” → “assertions” → “tests”
- Formal type
  - One big part which is automated and generated by tools based on state space exploration of the formal specification
  - A smaller part which is hand written
- But in both cases, all the tests is effectively a software executed on PCs or dedicated embedded devices using a hardware implementation of the specification

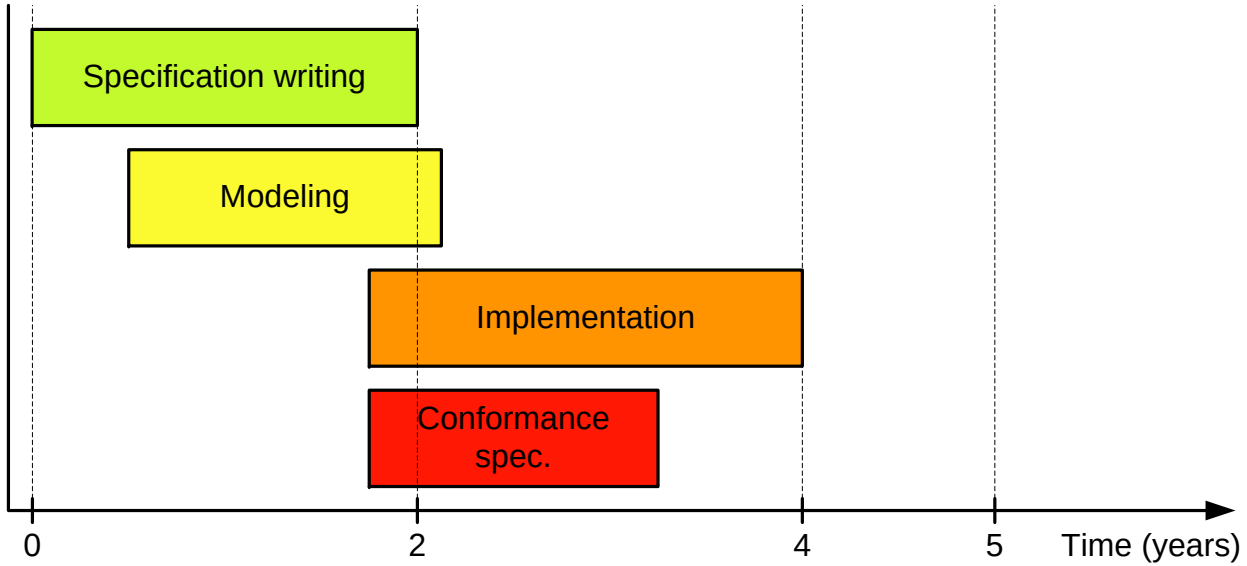
Tester and DUT

- The Device Under Test, DUT, is a implementation strictly following the specification
- The tester has additional behaviors, not part of the spec, which allows it to “violate” the spec
  - Wrong CRC, wrong format, etc.
- But still the spec behavior is not changed, we still have only one spec.



Creation timeline 2

- What if Conformance specification starts much earlier?
  - Tests can be used while implementations are being done
  - Verification can start at a very early stage
- better chances to have interoperable implementations
- ⇒ overall, improve the quality of the specification



Virtual hardware

NOKIA

Modeling

- For protocol in general, but even more so for embedded network, a significant part is implemented in hardware
- Developed models are very often not suitable for software/hardware co-simulation
- Mainly made for testing and verifying the protocols, not for virtual hardware
- Ex. SystemC is the de-facto standard for hardware modeling, but is also well known for having strong limitations in modeling software

SystemC limitation for modeling software

- No possibility to create dynamic processes after the elaboration phase is finished, at least version 2.0.1
  - Simulation starts only after the elaboration phase
  - SystemC 2.1 is a bit better, and 2.2 was still improved
- SystemC is based on “user threads”, which are scheduled in a collaborative manner by the SystemC simulator engine and not in preemptive way like os kernel for processes
- SystemC 3.0 is/was supposed to fix this, but already 2 or 3 years delayed

How to model software with SystemC?

- As said earlier, 2 main problems
- Must deal with dynamic creation of SystemC processes
  - SystemC 2.2 has pretty much done that
- But how to model a preemptive process scheduler with SystemC?
  - Option 1: port different OS kernels to SystemC: hard and complex
  - Option 2: create alternative SystemC scheduler with a more “software” behavior: but then it’s very hard to keep hardware modeling efficient or easy

“Conclusion”

- We have the choice between
  - A very hard and complex solution
  - A more manageable solution to simulate software, but we loose the possibility to simulate hardware
- Both solutions are not suitable
- Any proposal is welcomed to solve this problem

NOKIA

What do we have?

- A SystemC model of a hardware implementation of a protocol stack
- The need to write software on top of the protocol stack, which is the conformance tests
- We need to simulate both
- A SystemC simulation is simply a C++ software running in an OS on a computer …

- The OS has a process/thread scheduler … why not reuse it??

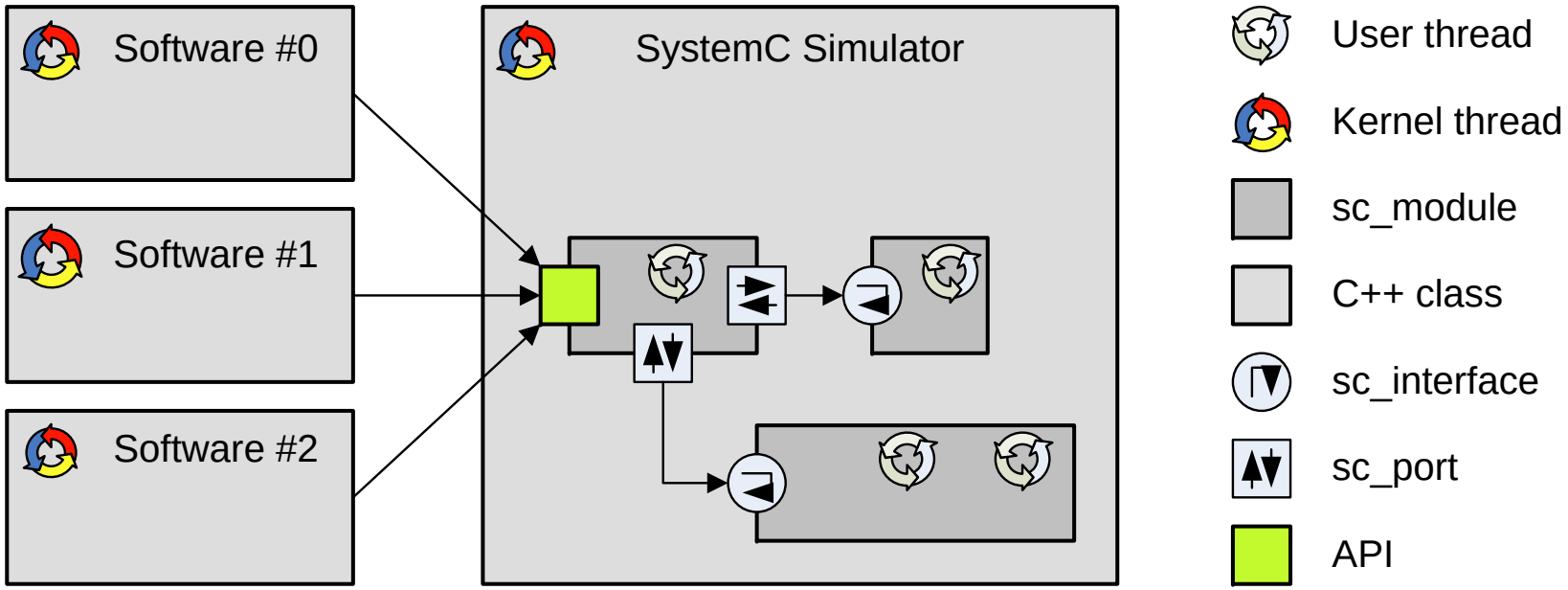
SystemC simulator

- SystemC models hardware by using a cooperative sets of independent state machines
- Each state machine is model by a SystemC “process”
- Each SystemC “process” is a user thread
- The whole SystemC simulator is a single kernel thread scheduling the user threads
- What if we simulate a software process by a kernel thread?

NOKIA

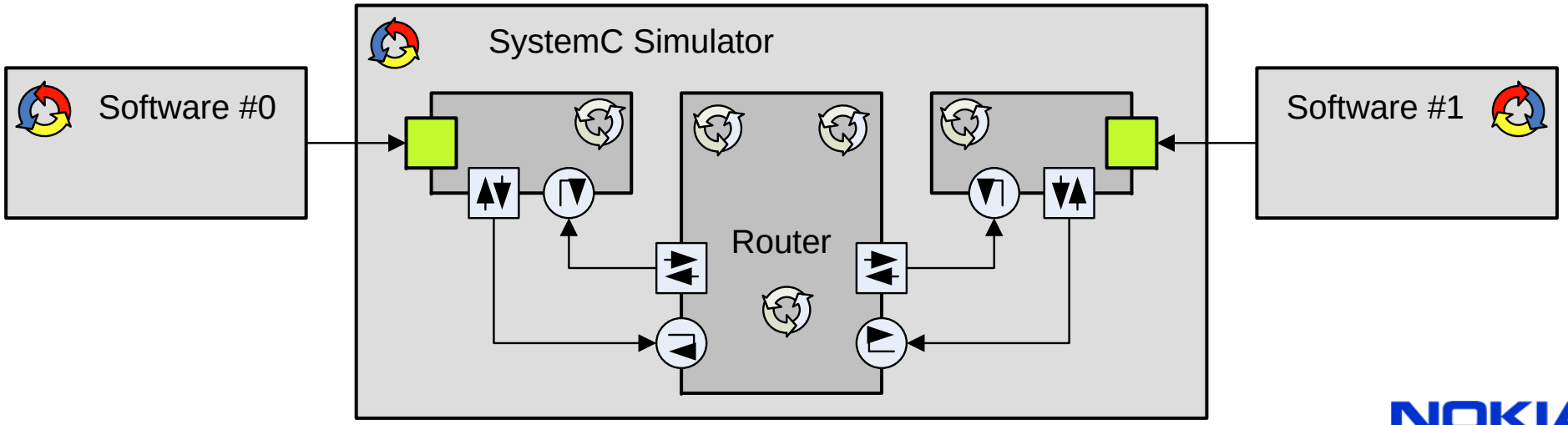
SW/HW co-simulation

- One kernel thread for the SystemC simulator
- One kernel thread for each software process
- The linkage between SystemC and pure C++ is done through APIs, which are independent of SystemC



Limitations

- This example models
  - 1 router
  - 2 endpoints, each with a CPU and OS, running each a software
- The real system has then 2 independent OS kernels
- In our model, we have only OS kernel modeled
- We can’t model deterministically and/or accurately the case where the CPUs used be soft#0 and soft#1 have different speed



Problem: SystemC is not thread safe (1/2)

- One must know
  - SystemC is an event-driven simulator
  - that the core of a SystemC simulator is in one member function of the class sc_simcontext
    - void sc_simcontext::simulate(const sc_time& duration)
- This member function handles a list of events, sc_event
- We must be sure that the handling of the list of sc_events is thread safe
- We need to introduce a mutex (or critical section, etc.)

NOKIA

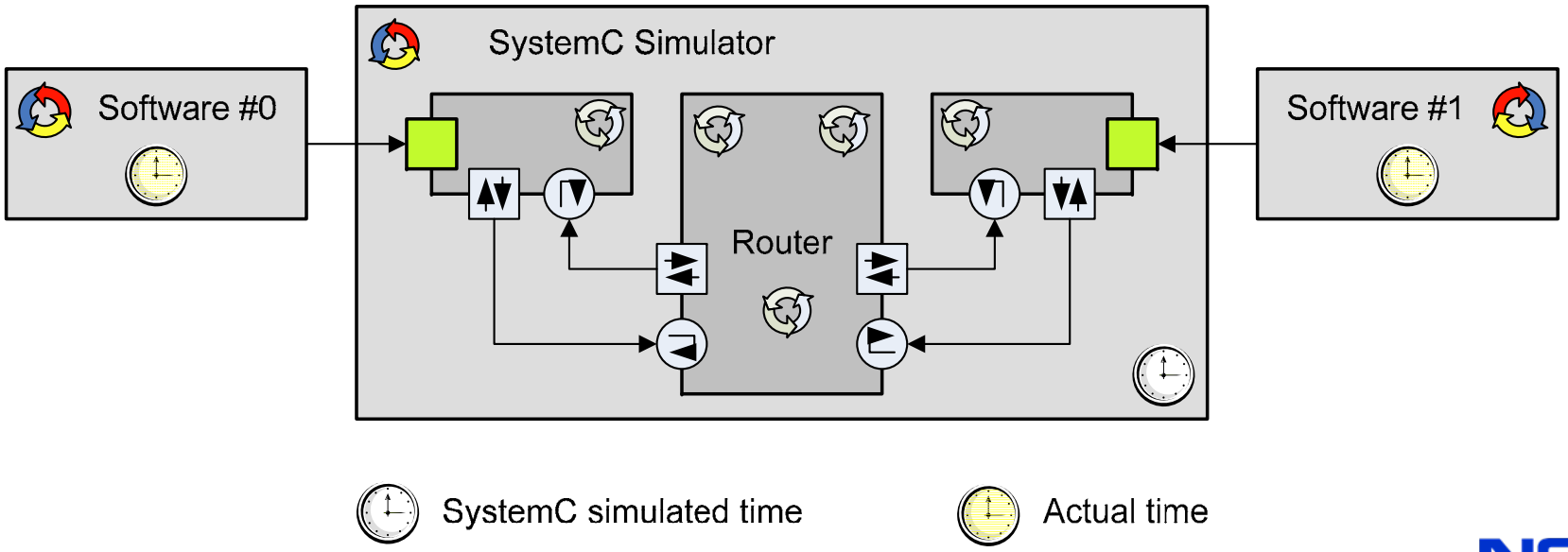
Problem: SystemC is not thread safe (2/2)

- We could have a mutex
  - At every place where a sc_event is added or removed from the event list, but it basically implies a re-writing the SystemC library
  - Only in the simulate member function around one simulation cycle
- The latter gives a synchronization point between kernel threads, where the list of sc_event can safely be modified

NOKIA

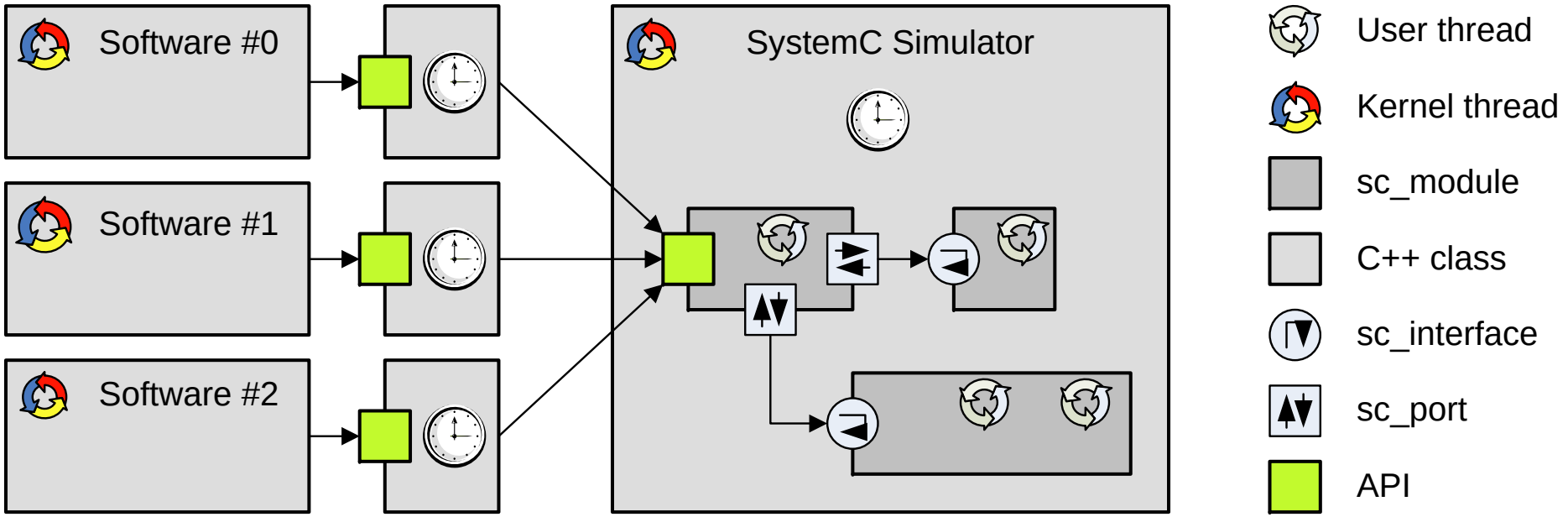
Problem: time consistency (1/2)

- SystemC simulates the notion of time
- While the Software running directly in the OS uses the physical time
- If Soft#0 use a function to sleep 1 s, in SystemC the time may have advanced by 1 ps, 1 s or 1 hour



Problem: time consistency (2/2)

- Solution: align the time reference of the Software model with the SystemC simulated time
  - Simply use one semaphore per Software: first trigger an expiring timed event in the SystemC simulator, wait on the semaphore; when the event “expires”, signal the semaphore



NOKIA

API used

- For an embedded network, the APIs used are
  - BSD Socket API for the data paths
  - Some specific APIs for control
- But all functions of the APIs between SystemC and the Software models must be “time aligned”
  - For each of this functions an execution time is defined
- The APIs are independent of SystemC
  - $\Rightarrow$ the same tests can be reuse with a SDL, VHDL or real hardware “backend”

Demo

- The demo comes form the conformance work done in UniPro, which is an embedded network defined in the MIPI standardization
- 2 nodes; frames have sequence numbers because of a ack-nack mechanism to re-transmit in case of errors; the number of available sequence number is limited
- The test does the following
  - Send enough frames to use up all available sequence number
  - And verify that a timer defined in the specification to avoid deadlocks behaves properly

NOKIA

Conclusion

- Hardware/software co-simulation is easily achievable with SystemC 2.0.1, no need to wait for SystemC 3.0
- A SystemC framework can be used to build virtual hardware/software model, making possible software development when the hardware is not yet available
- Give the opportunity
  - to develop conformance tests in parallel with the hardware implementation of a specification $\Rightarrow$ better specification
  - Build/verify the software platform using the embedded network technology

NOKIA

Thank you