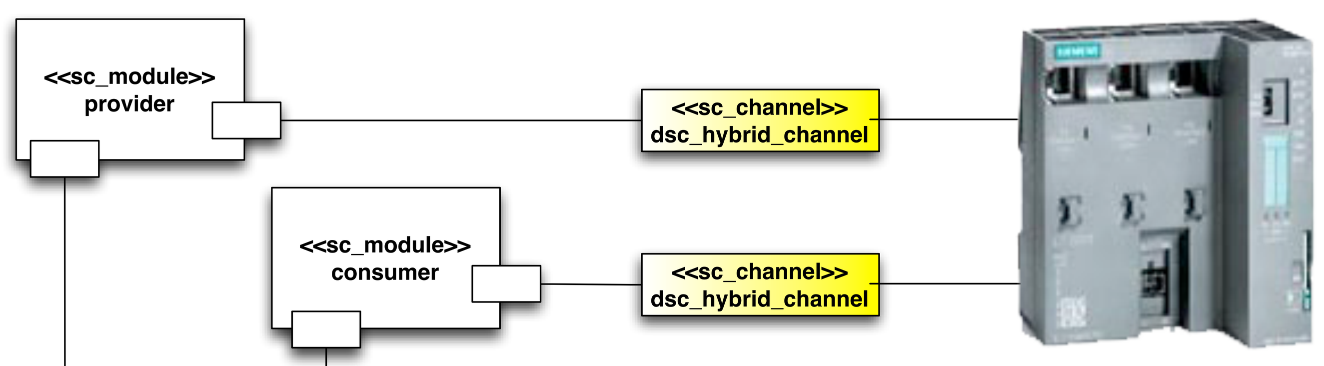
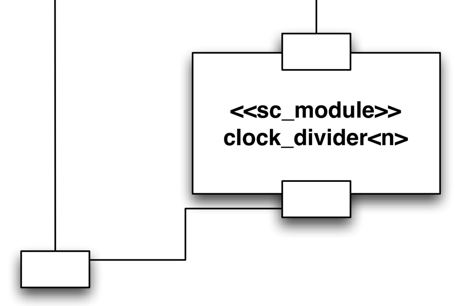
Hardware-in-the-loop for Hardware/Software Co-design of Real-time Embedded Systems

SIEMENS Doğan Fennibay[1, 2] Arda Yurdakul[2] Alper Şen[2]
1 Corporate Technology, Siemens AŞ, Kocaeli
2 Department of Computer Engineering, Boğaziçi University, İstanbul
dogan.fennibay@siemens.com, yurdakul@boun.edu.tr, alper.sen@boun.edu.tr

Connect your real (sub)system to your SystemC model!





Domain: Industrial Applications

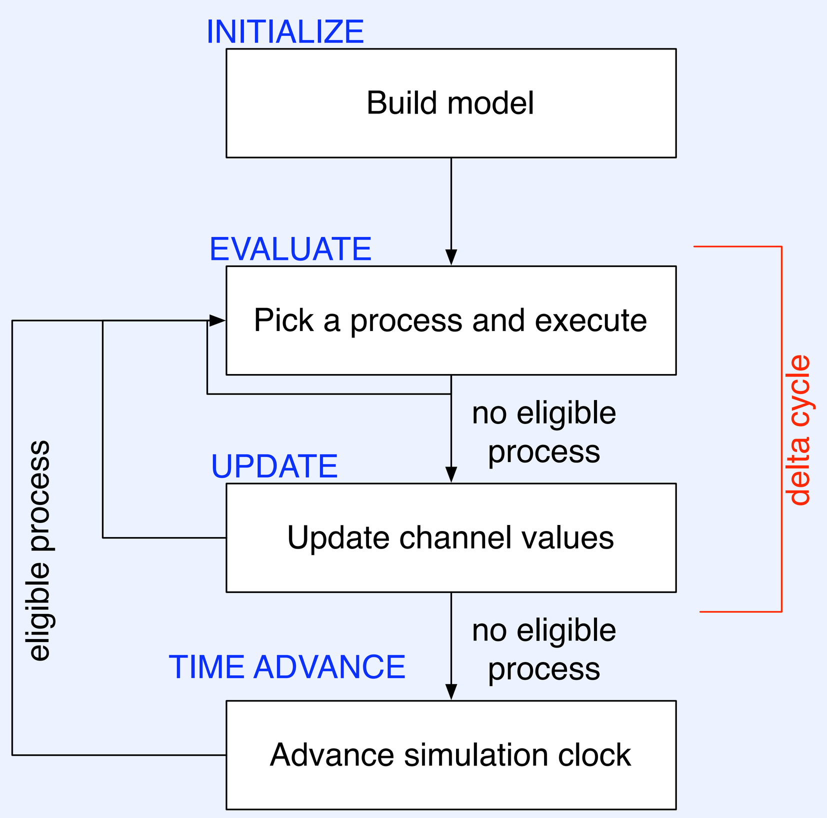
- Strict hard real-time constraints
- Data exchange rate 10 KHz achievable with current modeling platforms
- System-level modeling is the new design trend

Why would I want to do that?

- To avoid modeling complex systems when physical implementations of such systems are available.
- To obtain a higher modeling accuracy with less modeling effort.
- To test virtual subsystems with a real testbed.

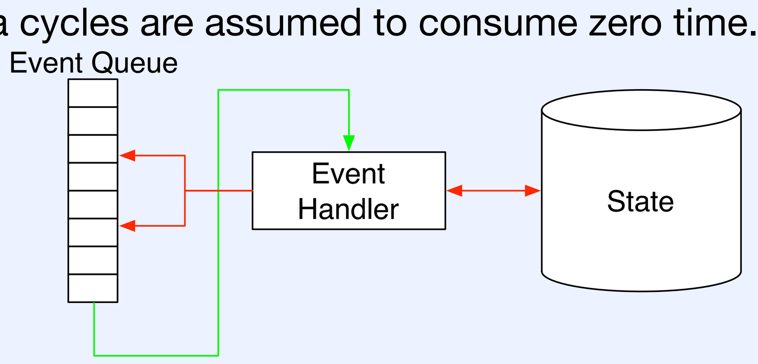
1 Patch SystemC kernel for real-time execution

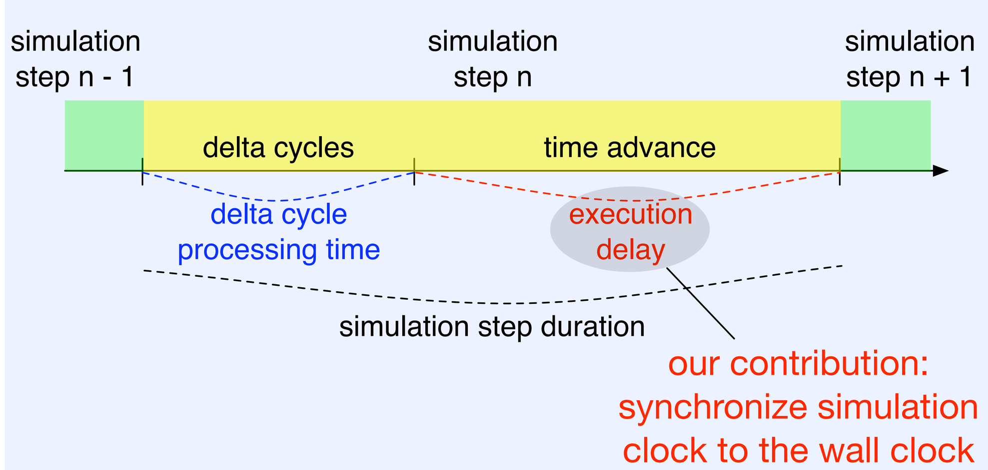
Why?
Virtual subsystems need to behave according to the real time clock as the real subsystems do.



SystemC uses a discrete event simulator

- Simulation clock is advanced in discrete intervals.
- Delta cycles are assumed to consume zero time.



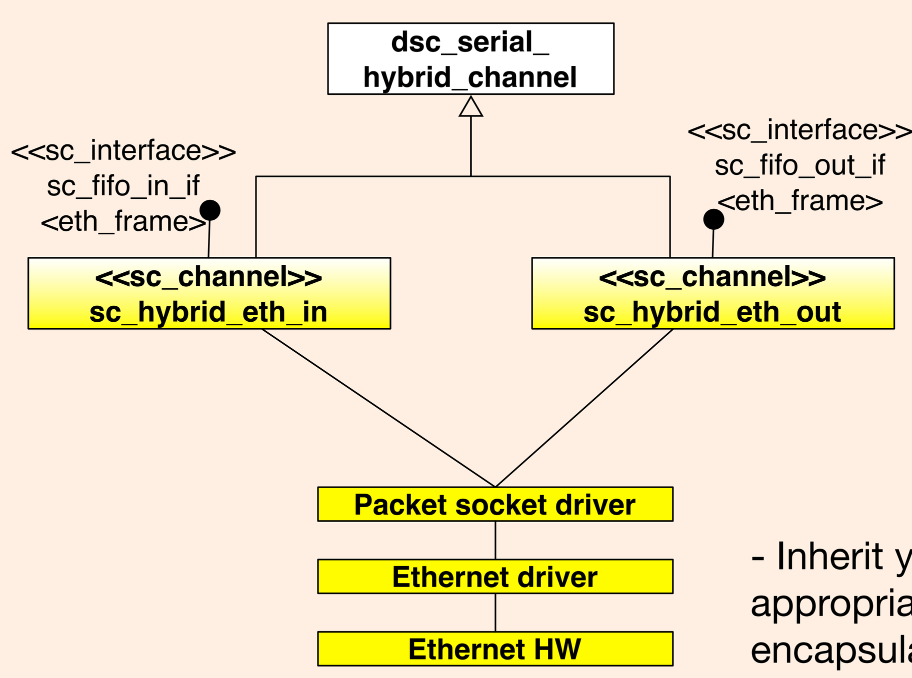


The patch also includes update_real functionality.

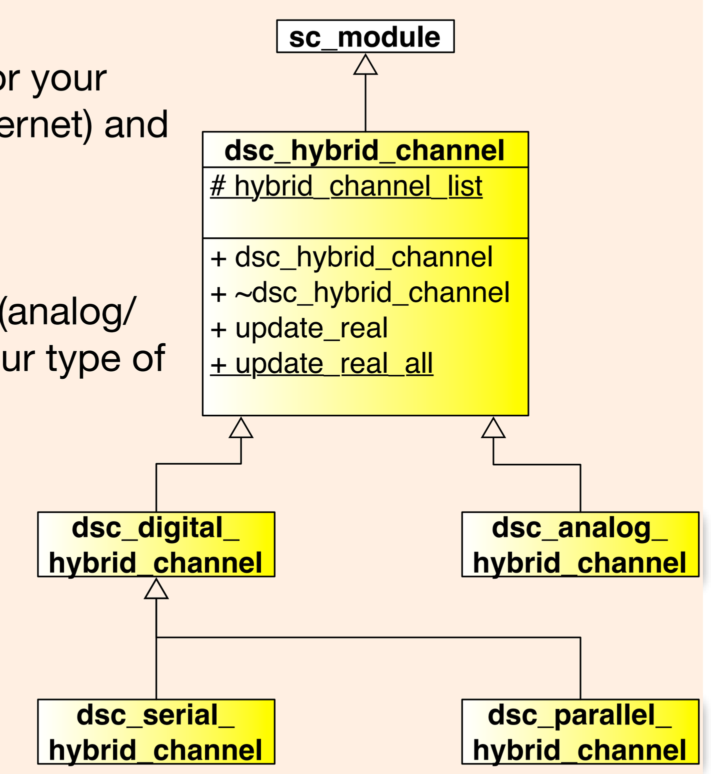
2 Build a hybrid channel

Why?
We have to establish communication between real and virtual subsystems.

Create the hybrid channel



- Get the hardware interface for your communication type (here Ethernet) and its IO driver.
- Pick a communication class (analog/ digital, parallel/serial) fitting your type of communication.
- Inherit your SystemC channel from the appropriate hybrid channel class and encapsulate the integration to the I/O driver inside.



Manage output timing

Decide in which phase to transmit the output values to real subsystems:

Why?
Different types of communication need different timing of output operations.

Phase	Advantages	Disadvantages	Example
Evaluate e.g. *write*	Signal does not wait	Signal value has to remain the same in later delta cycles. (e.g. sc_fifo) High number of output operations.	An Ethernet channel
Update *update*	Final value from concurrent processes will be used (e.g. sc_signal)	Processing time of update phase increases. Concurrent outputs w.r.t. the simulation clock distributed in real-time.	A digital I/O channel
Time advance *update_real*	Fewer total output operations. Concurrent outputs from delta cycles gathered together.	Glitches occurring at the end of delta cycles not transmitted to the real subsystems.	A PWM motor driver

Incorporate external events (inputs)

Why?
Discrete event simulators advance the simulation clock according to the next element in the event queue. If an external event occurs between the current time and the next internal event, it will not be processed until the next internal event.

- Use an asynchronous OS thread to get the external event which sets a fast-to-check flag for SystemC. (optional)
- Specify a polling rate.
- Poll the external event at the specified rate from a dedicated SystemC thread and set a SystemC event on detection.
- Rest of the model can use that event.

3 Improve determinism

Why?
Non-determinism in OS reduces predictability of virtual subsystems and therefore decreases accuracy of the whole model.

RTOS vs. GPOS

- GPOS: More facilities, I/O interfaces
- RTOS: Deterministic behavior.

Proposed solution: GPOS with real-time improvements (Linux with RT PREEMPT)

Improvements in OS

- Patch the Linux kernel with RT PREEMPT.
  - Interrupt handlers moved to thread context
  - Spinlocks made preemptible
  - Priority inheritance implemented for semaphores and spinlocks
  - Linux timers replaced with high resolution counterparts
- Turn off power management
- Disable swap memory

Application settings

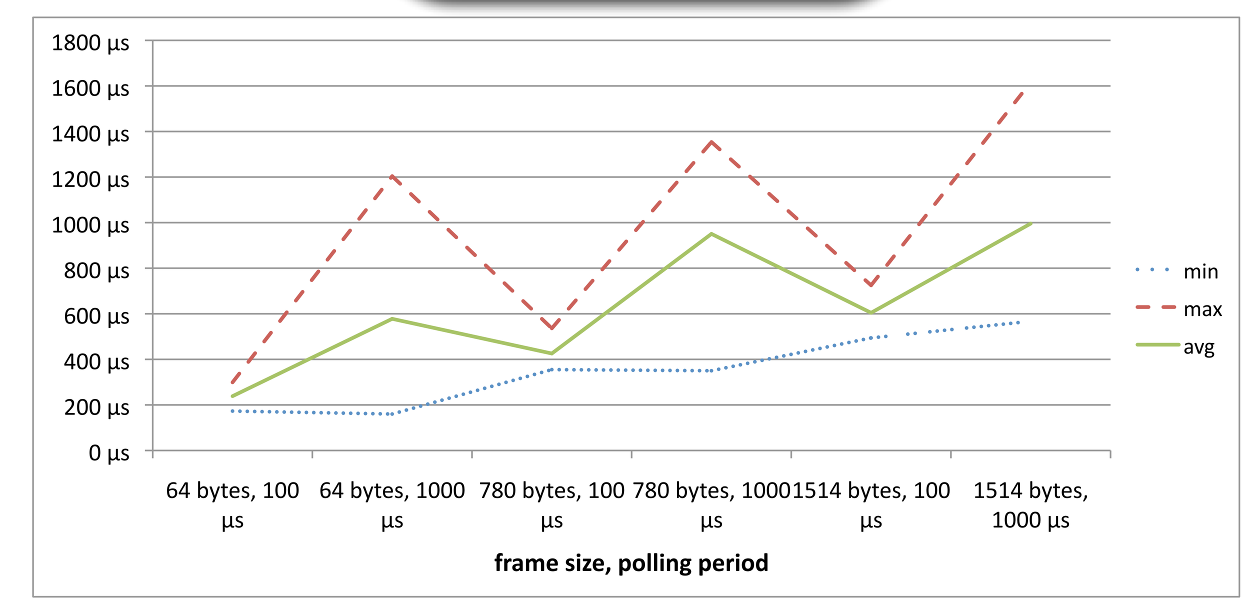
- Set application threads to real-time scheduling and set priorities.
- Extend thread stacks at initialization to avoid page faults during execution.

Latency hunting

- Eliminate further sources of latency (corrupt driver, hardware causing unnecessary interrupts etc.)
- Use ftrace facility of Linux for diagnosis.

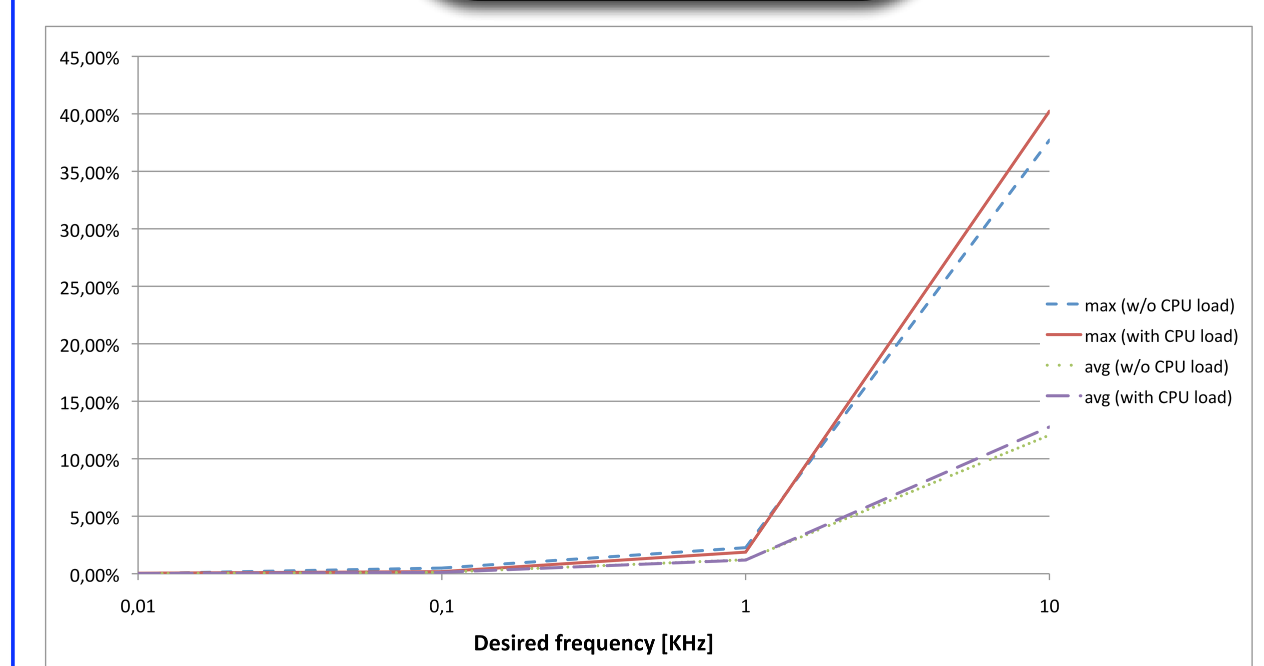
How does it perform?

I/O Performance



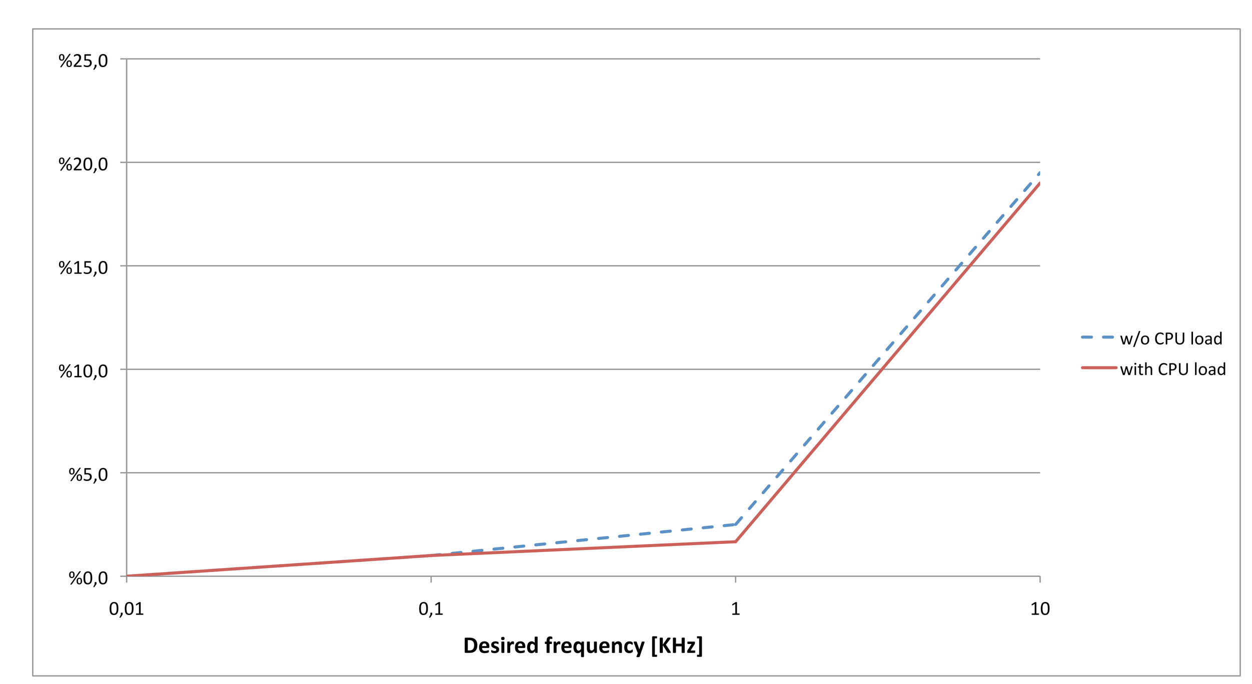
Round-trip time measurement over Ethernet
- Polling time is the main factor.
- Frame size has a linear, mild effect.
- 1 ms communication cycle feasible with low polling period.

Simulation Performance



Square wave generation: delta cycle proc. time / simulation step
- Average load <15% at 50 microseconds signal change time with a simple model.
- There is room for higher performance or more complex model, but jitter is the problem.

Determinism



Square wave generation: maximum jitter / desired period
- Frequency sustained up to 10 KHz, unstable at 100 KHz
- CPU load has minimal effect, OS scheduler working fine
- Measurements coincide with internal measurements above, main factor is the simulation performance rather than the I/O performance.