

Building a Parallel Simulation Kernel for Faster & Better Virtual Platforms

The VLAB™ 3 Hipersim Kernel

Jakob Engblom
engblom@cadence.com



SPONSORED BY





Dr. Jakob Engblom

Is: Product Marketing Director, Cadence

Was: Global Technical Marketing Manager, ASTC VLAB

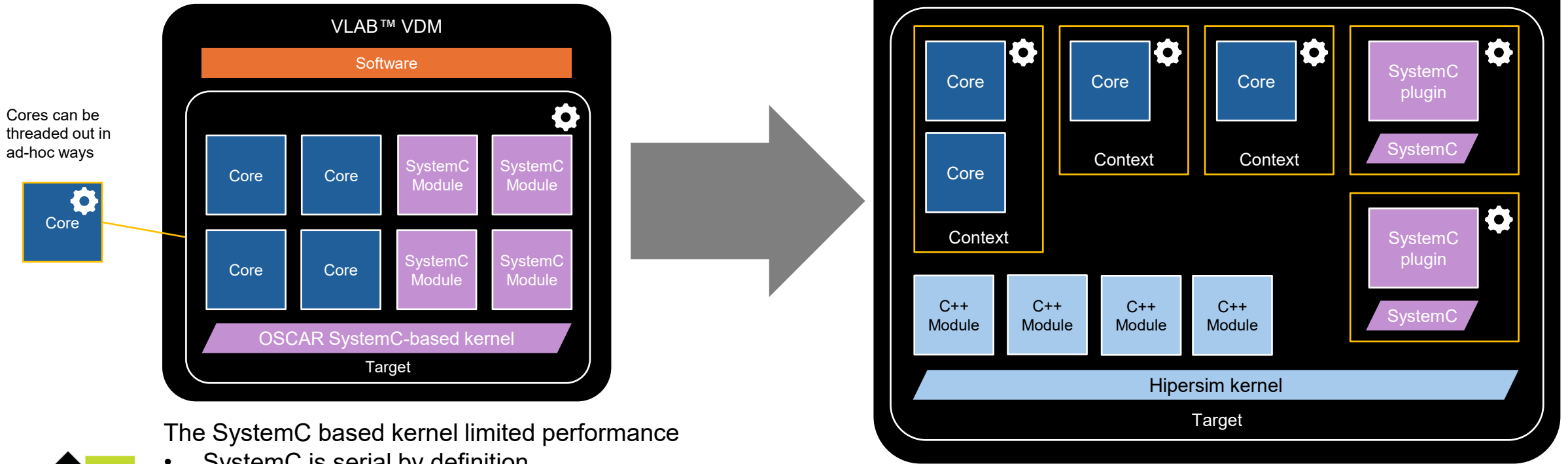
cādence[®]
VLAB | WORKS[™]



From Serial SystemC to Parallel C++

Requirement: faster simulation!

(bigger target systems, faster target cores, more target cores, real-time speed requested, ...)

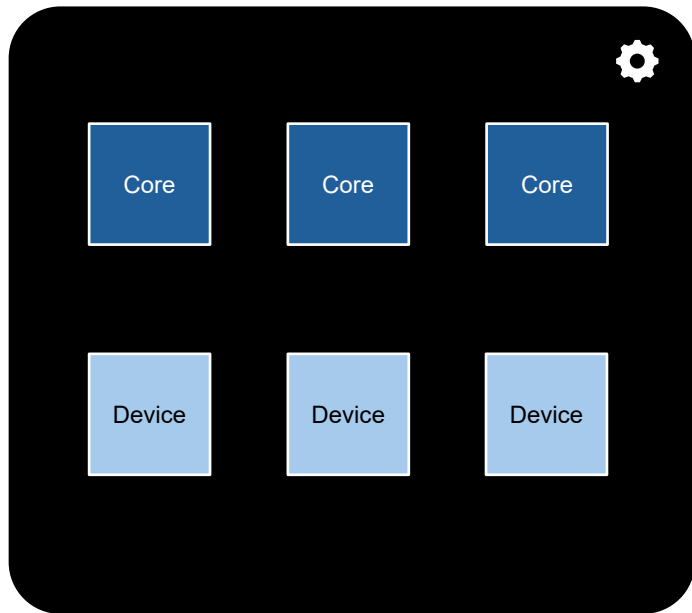


The SystemC based kernel limited performance

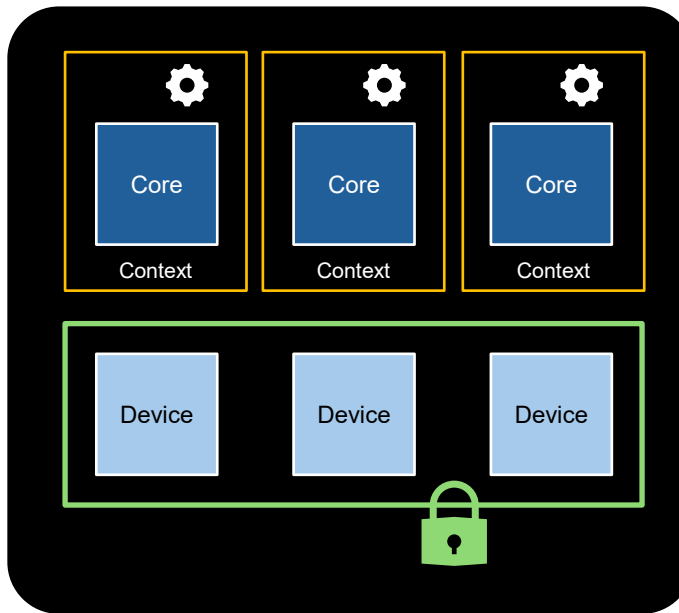
- SystemC is serial by definition
- SystemC simulation semantics carry a lot of overhead



Parallelism: Locking Strategy

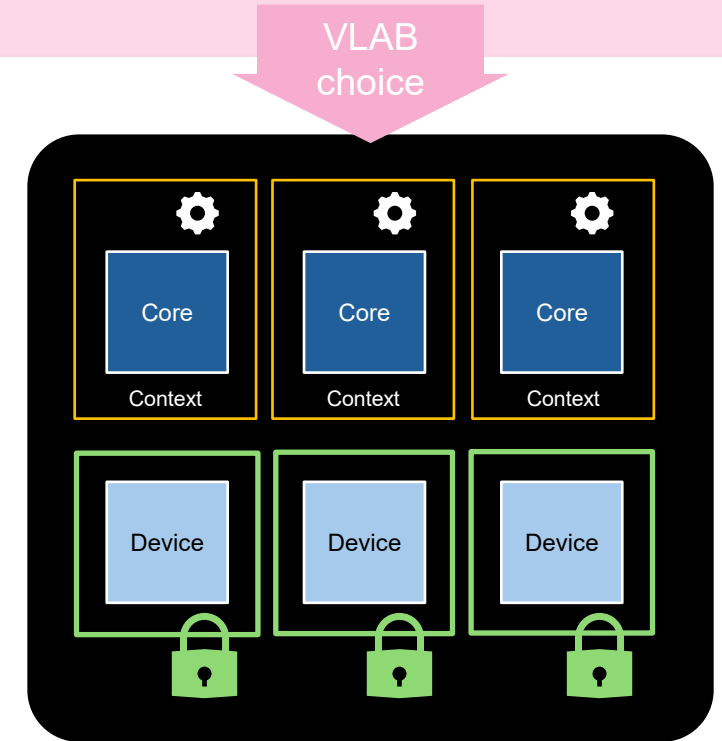


Serial execution – no locking needed



All devices covered by one big lock

Attempt to maintain serial semantics
Works fine – as long as software does not access hardware too often. Not ideal for embedded and automotive targets.



Fine-grained locking

Multiple cores can access devices simultaneously.
Framework locking and unlocking avoids deadlocks
Semantics are not too different from serial (!)

It Works!

Your mileage might vary

Limited by actual parallelism in the workload

Booting is usually serial

Many workloads have a single core on the critical path

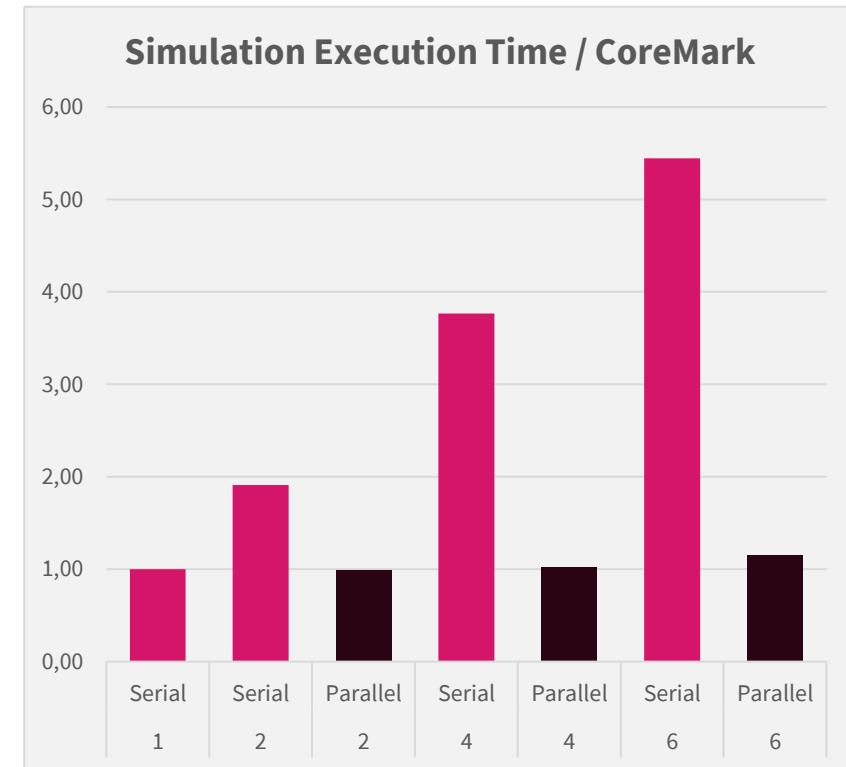
Watch out for thread spamming!
“Moar threads” do not necessarily help

Use configurations with a few cores per thread (context) to reduce wasted time

Serial simulation performance is vital for overall efficiency



N independent programs running in parallel



This is the easy case!

To Summarize: Hipersim for Speed!

We need a faster simulator!

SystemC is semantically serial
(parallelism requires change)

SystemC has a significant kernel overhead
(due to event semantics)

Implement new Hipersim kernel
(built for performance)

Streamlined C++ API for modeling and simulation
(avoid SystemC overhead)

Parallelism designed-in
(for performance scaling)

Faster base simulator
(parallelism is a "turbo" on top of a strong core engine)

Fine-grained locking strategy
(maximize parallelism)

SystemC models still work
(running on their own threads)





AI



Security



Systems



EDA



Design



**THE CHIPS
TO SYSTEMS
CONFERENCE**

SPONSORED BY

