ETH zürich

How Model-Based Design Simplifies the Debugging of Many-Core Systems

Iuliana Bacivarov

Computer Engineering and Networks Laboratory, ETH Zürich

1st International Workshop on Multicore Application Debugging (MAD) 2013, 14-15 November 2013, München, Germany

Institut für Technische Informatik
und Kommunikationsnetze
Computer Engineering and Networks Laboratory

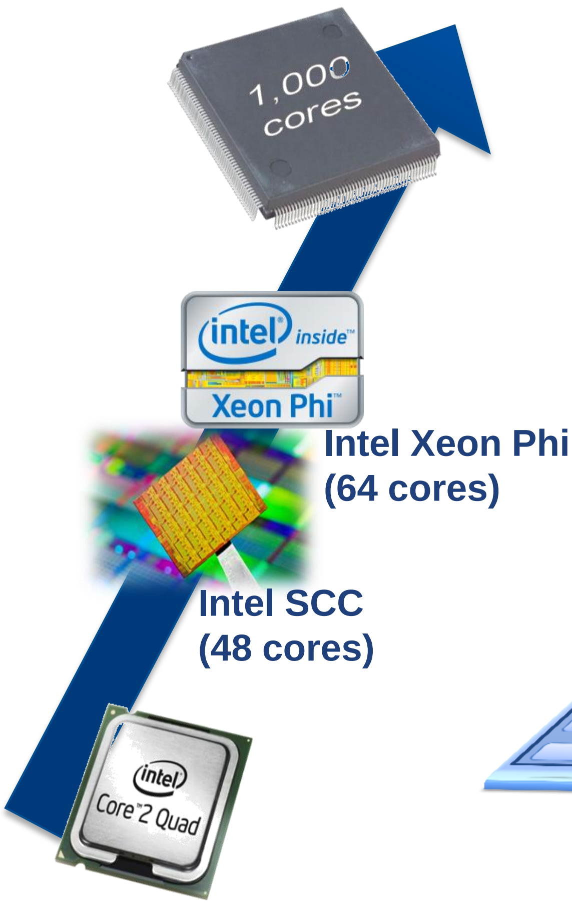
ETH zürich

Acknowledgements

- **team**
  Devesh Chokshi, Wolfgang Haid, Kai Huang, Shin-Haeng Kang, Pratyush Kumar, Devendra Rai, Lars Schor, Hoeseok Yang, Prof. Lothar Thiele

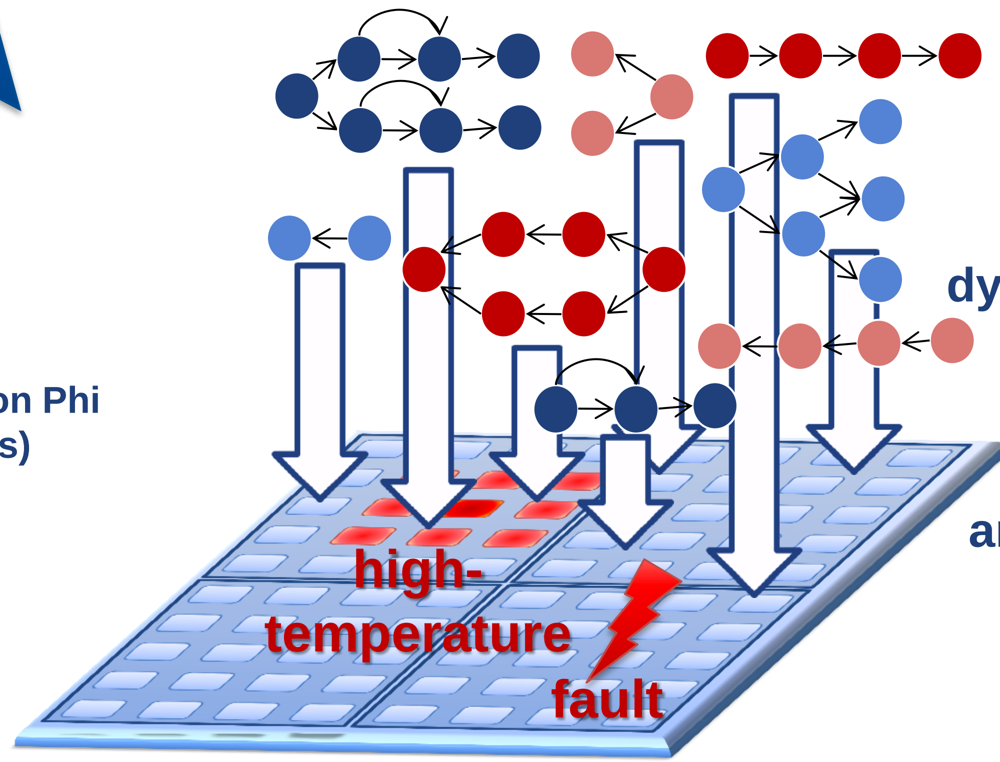
- **projects**
  EU-SHAPES, EU-PREDATOR, EU-COMBEST, EU-ARTISTDESIGN, EU-PRO3D, EU-EURETILE, nano-tera Extreme, nano-tera UltrasoundToGo

Current Embedded Systems are Complex



parallel applications

dynamic workloads

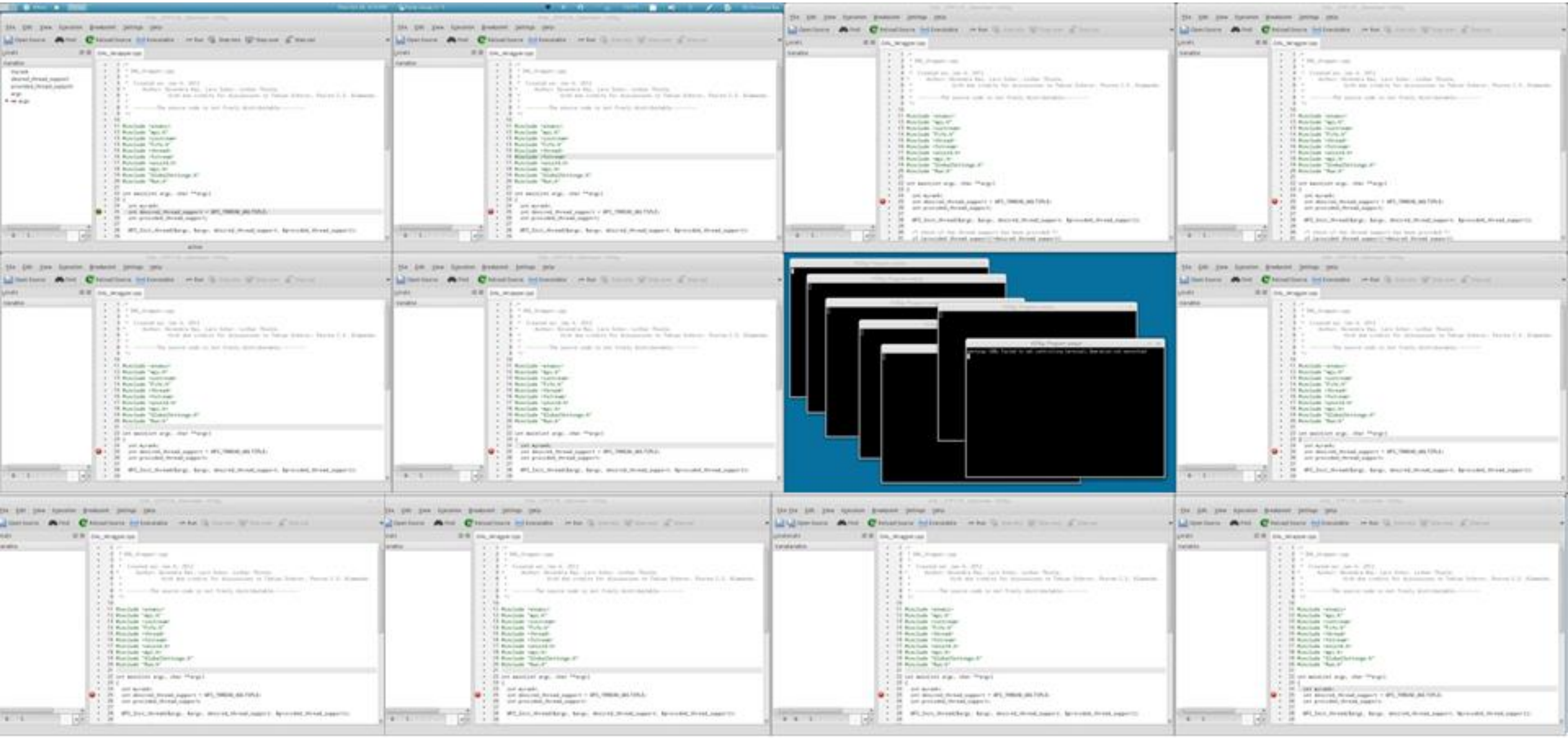


dynamic mapping

performance, real-time, power, and temperature

many-tile/many-core hardware

Debugging is Hard!



Debugging

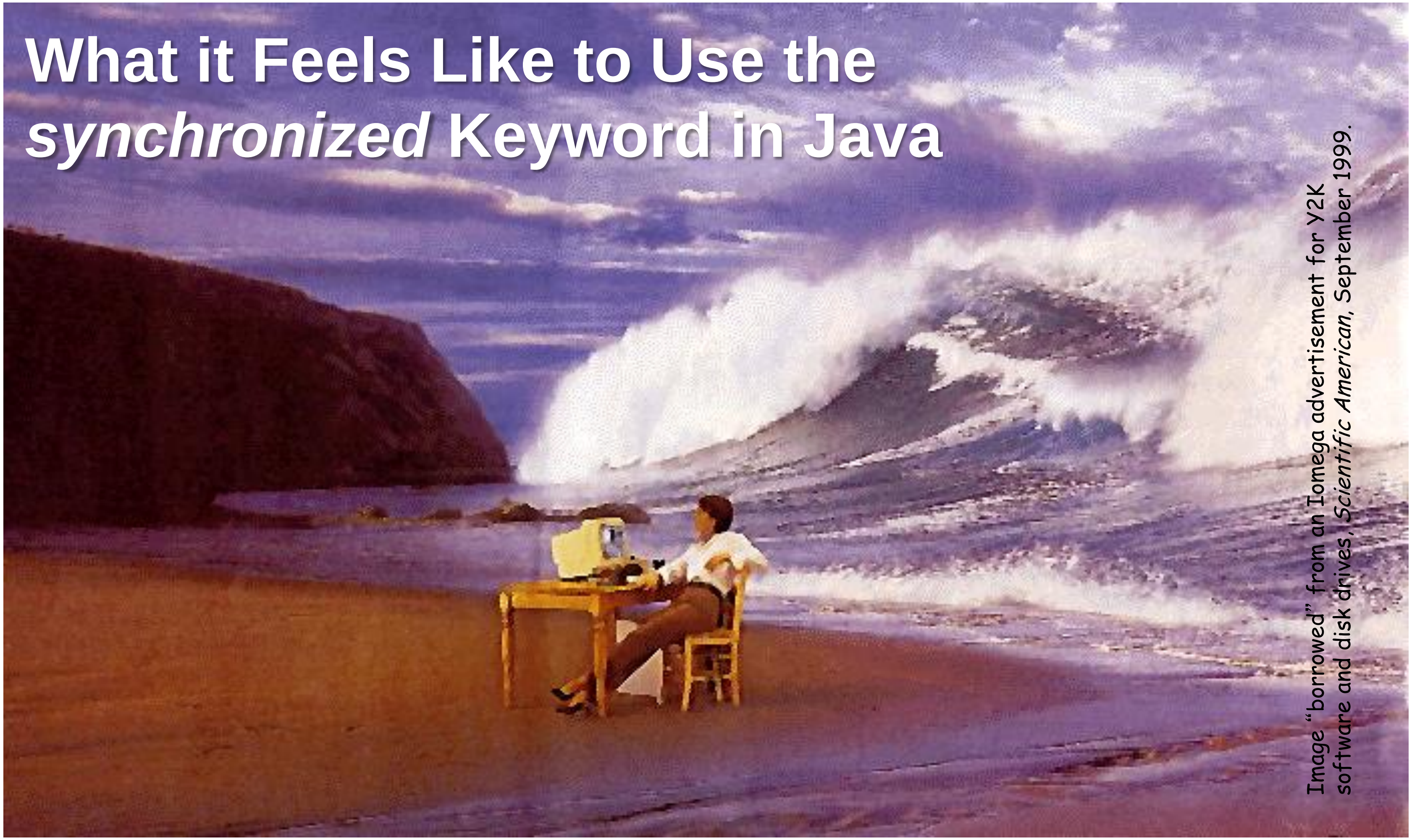
- "Debugging is a ***methodical process*** of finding and reducing the number of bugs, or defects, in a computer program or a piece of electronic hardware, thus ***making it behave as expected***."

  ---- Wikipedia

- "Debugging tends to be harder when various subsystems are ***tightly coupled***, as changes in one may cause bugs to emerge in another."

  ---- Wikipedia

Problems with Parallel Programming



Ed Lee, The Future of Embedded Software, 2006
http://ptolemy.eecs.berkeley.edu/presentations/06/

Problems with Parallel Programming

- Threads are wildly nondeterministic
- The programmer's job is to prune away the non-determinism by imposing constraints on execution order (e.g., mutexes)

> "Humans are quickly overwhelmed by concurrency and find it much more difficult to reason about concurrent than sequential code. Even careful people miss possible interleavings among even simple collections of partially ordered operations."
>
> ***H. Sutter and J. Larus. Software and the concurrency revolution. ACM Queue, 3(7), 2005.***

- *Nontrivial software written with threads, semaphores, and mutexes is incomprehensible to humans*
- *… and doesn't deliver a rigorous, analyzable, and understandable model of concurrency.*

Ed Lee, The Future of Embedded Software, 2006
http://ptolemy.eecs.berkeley.edu/presentations/06/

Key Concepts in Model-Based Design

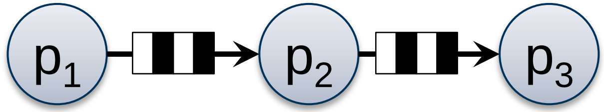
- Models are composed to form designs.
- Models evolve during design.
- Specifications are executable models.
- Deployed code is generated from models.
- Modeling languages have formal semantics.
- Modeling languages themselves are modeled.

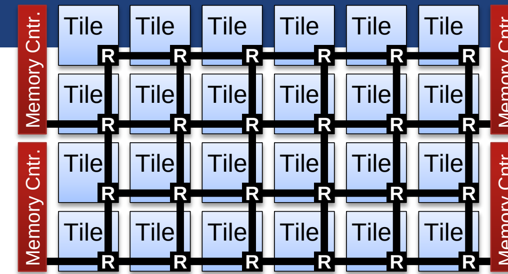
- For general-purpose software, this is about
  - Object-oriented design

- For embedded systems, this is about
  - Time
  - Concurrency

Ed Lee, The Future of Embedded Software, 2006
http://ptolemy.eecs.berkeley.edu/presentations/06/

The Good News

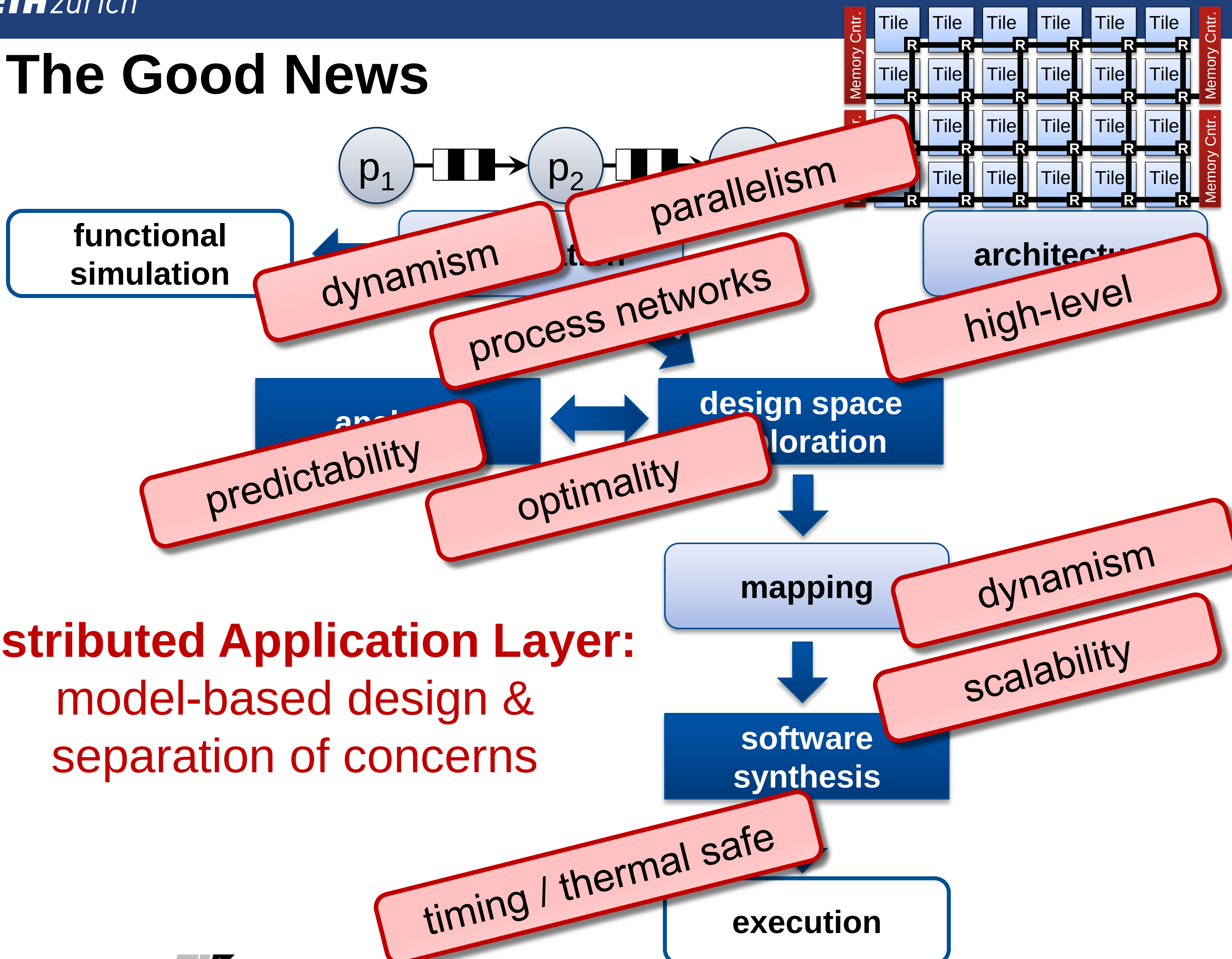




Model-Based Design enables a 'correct by design' execution

execution

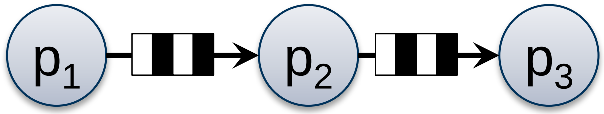
The Good News



Distributed Application Layer:

model-based design &
separation of concerns

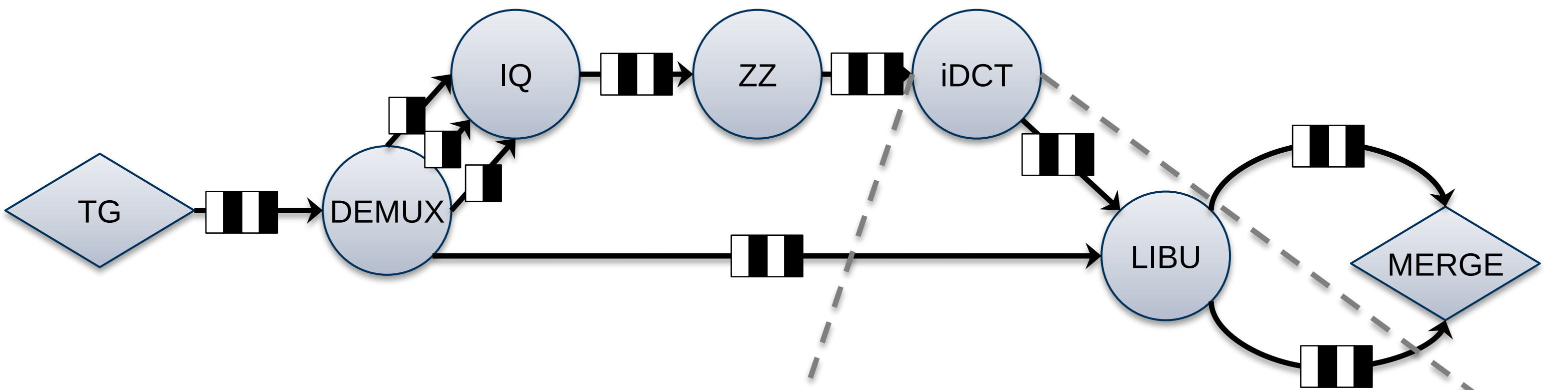
Application Specification: Kahn Process Network



- Proposed by Kahn in 1974 as a general-purpose scheme for parallel programming
  - READ: destructive and blocking
  - WRITE: non-blocking
  - FIFO: infinite size
- Unique attribute: **determinate**

- **Deterministic** model of computation
  - Focus on causality, not order (implementation independent)
  - Functional behavior is independent of timing (execution time, communication time, scheduling)
  - Data-driven scheduling: processes run whenever they are ready

Application Specification: MPEG2 KPN



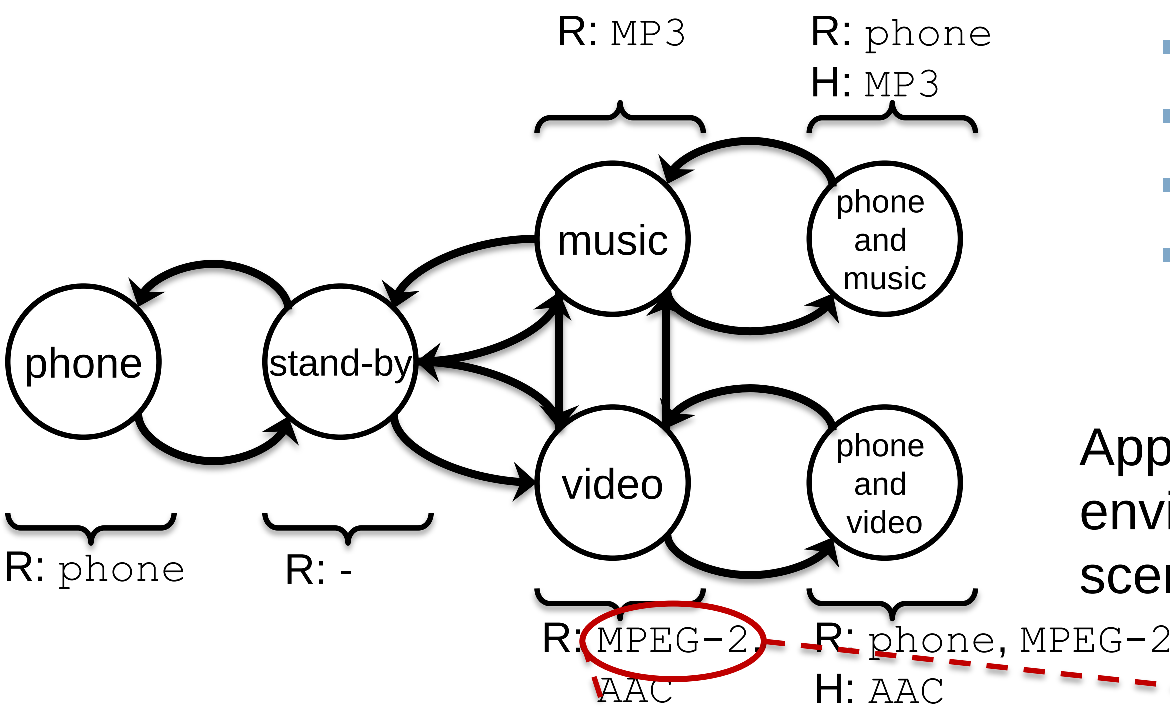

```
procedure INIT(ProcessData *p)
  initialize();
end procedure

procedure FIRE(ProcessData *p)
  fifo->READ(buf, size);
  manipulate();
  fifo->WRITE(buf, size);
end procedure

procedure FINISH(ProcessData *p)
  cleanup();
end procedure
```

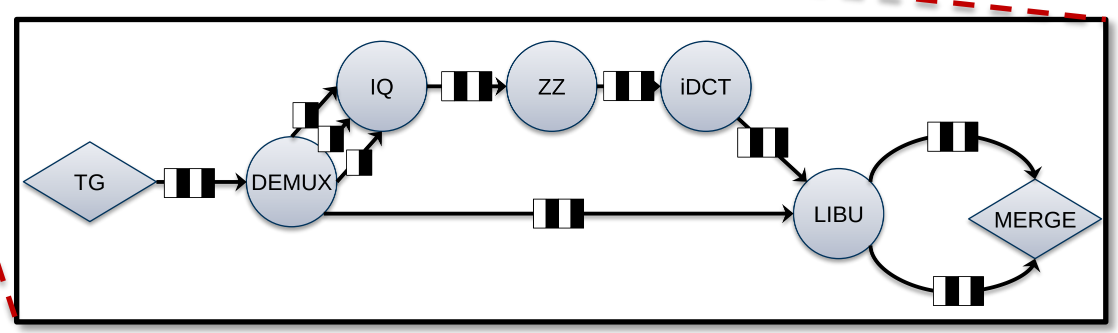
- Kahn process network
- Unique attribute: **determinate**

Execution Scenarios Specification



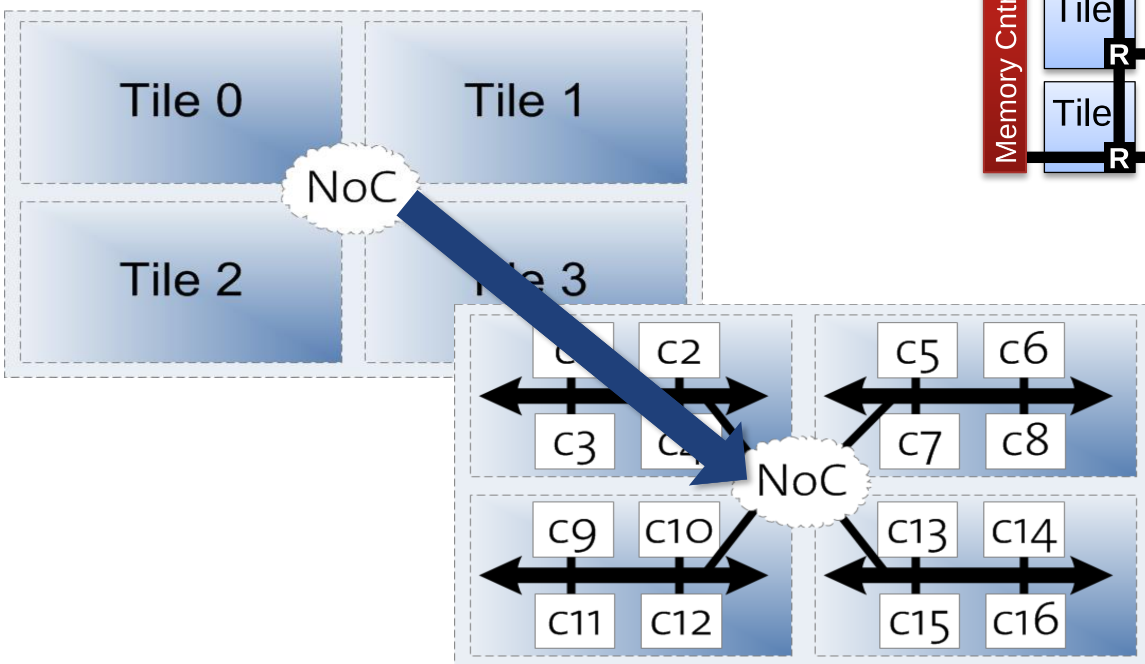
- Each application can:
  - START
  - STOP
  - PAUSE
  - RESUME

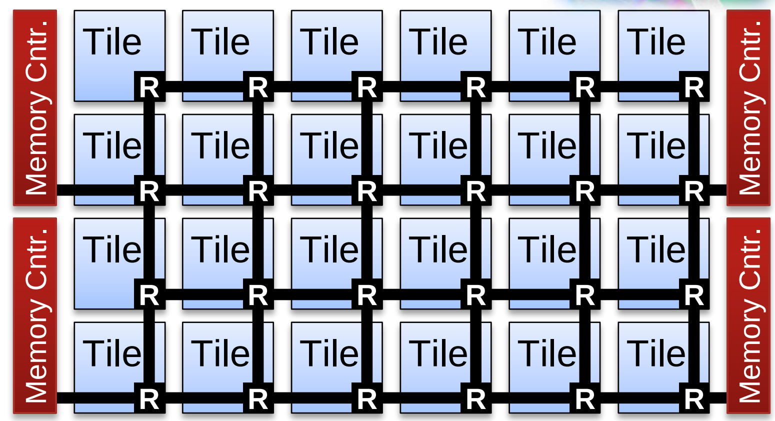
Application / run-time environment can request a scenario change



Architecture Specification

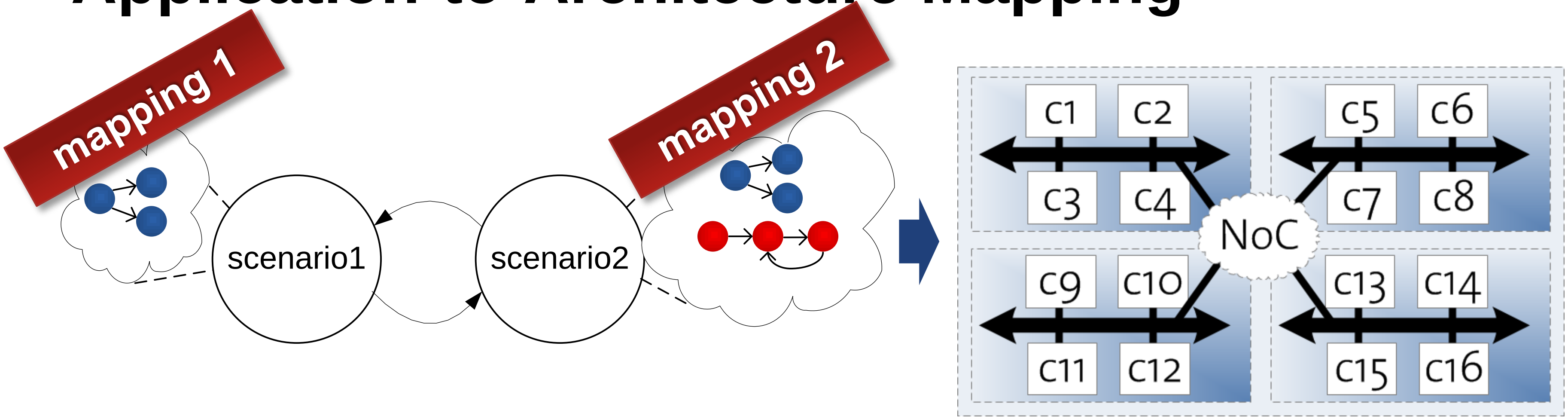
Hierarchical architecture

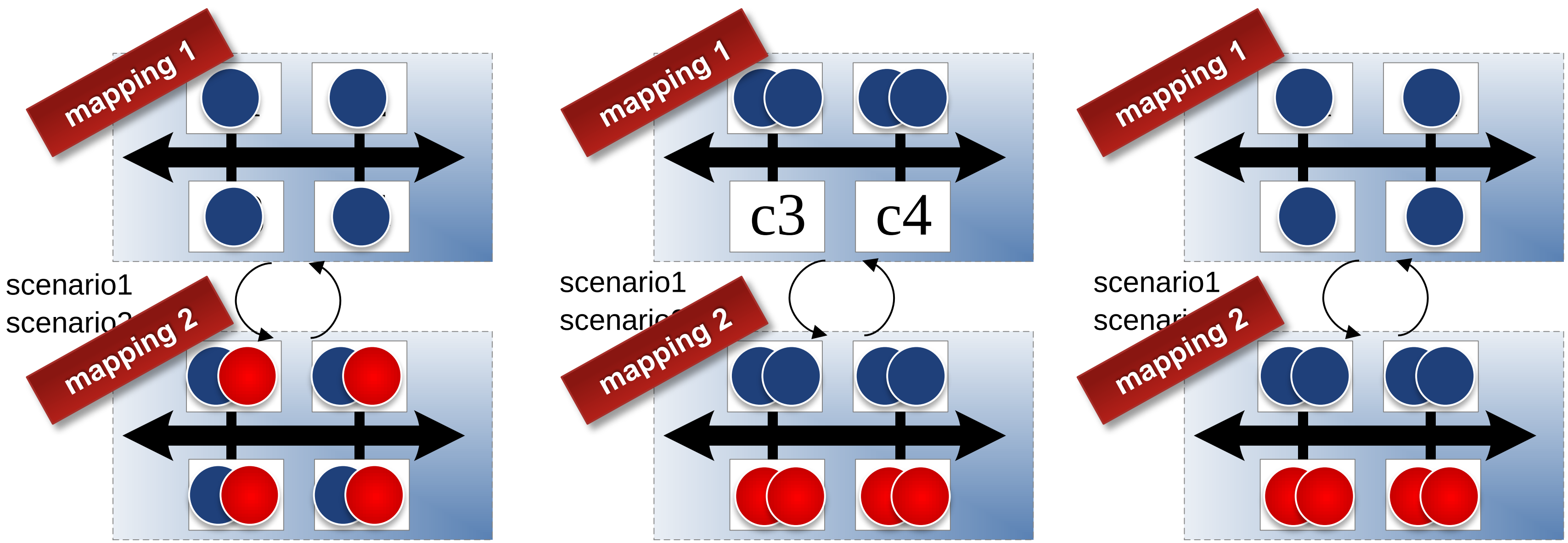




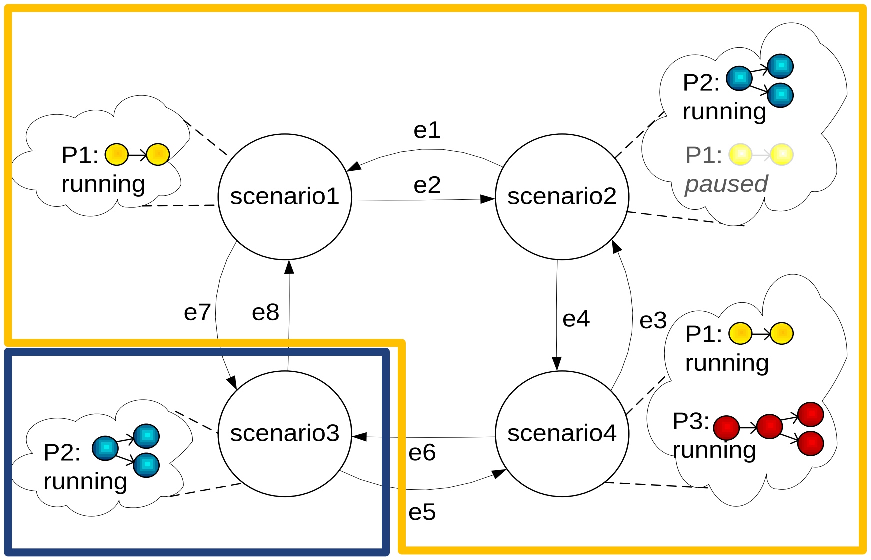
e.g., Intel SCC

Application-to-Architecture Mapping



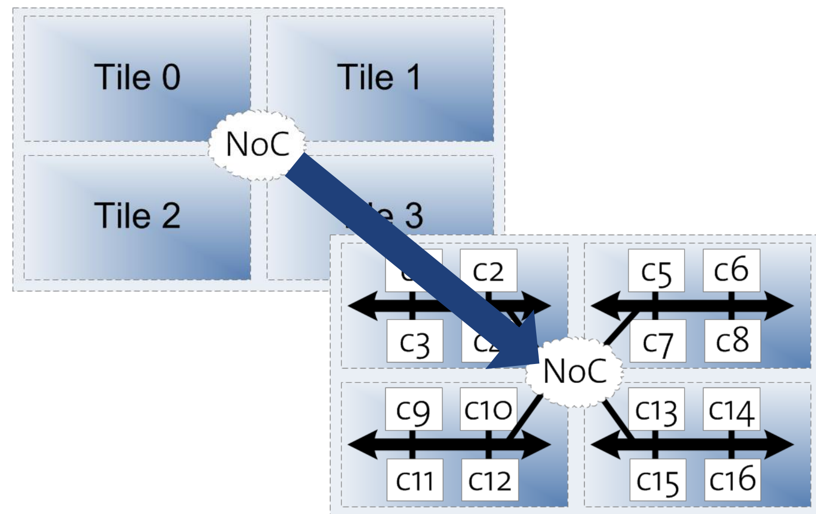


Hierarchical Mapping Optimization – via Problem Decomposition



state-based decomposition

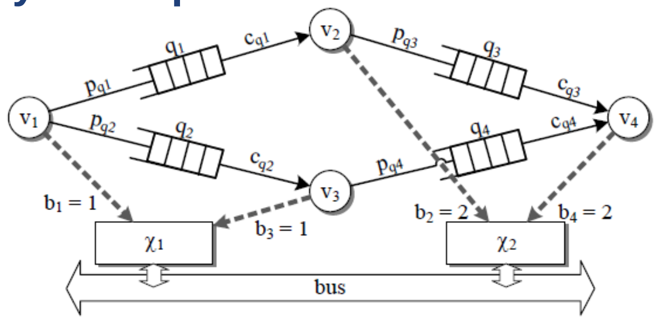
architecture-based decomposition

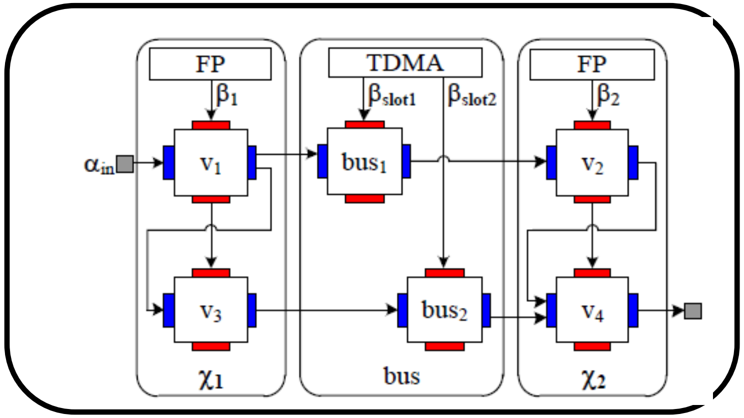


*[ref] S. Kang, H. Yang, L. Schor, I. Bacivarov, S. Ha and L. Thiele, **Multi-Objective Mapping Optimization via Problem Decomposition for Many-Core Systems**, ESTIMedia, Tampere, Finland, Oct. 2012*
*[ref] L. Schor, I. Bacivarov, D. Rai, H. Yang, S. Kang and L. Thiele, **Scenario-Based Design Flow for Mapping Streaming Applications onto On-Chip Many-Core Systems**, CASES, Tampere, Finland, Oct. 2012*

From Specification to Analysis and Simulations

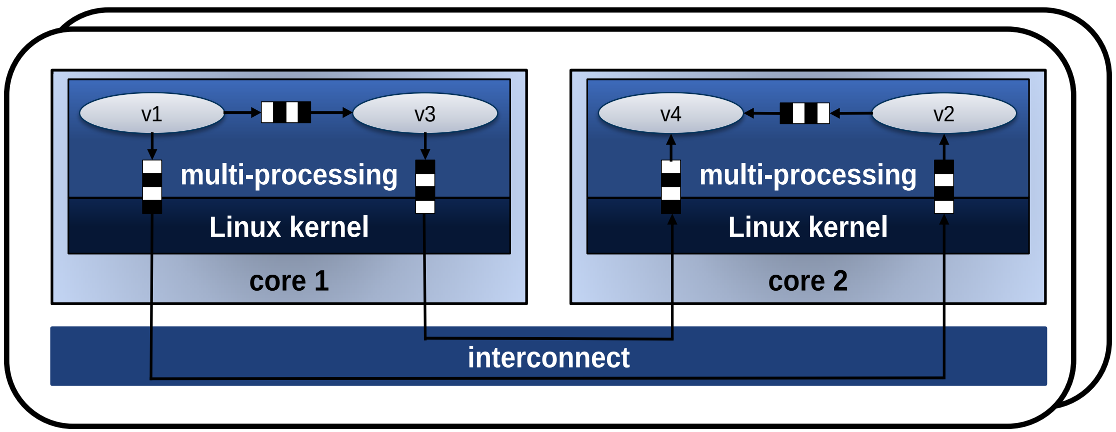
system specification





MPA analysis model

functional simulation **simulation/execution**

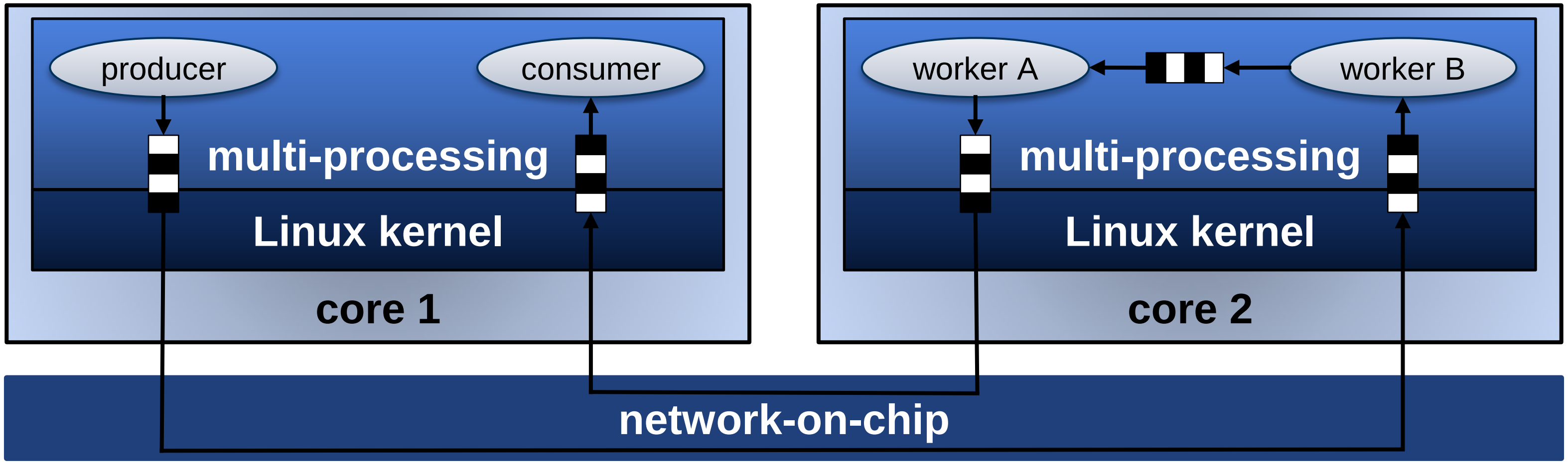


- automatic generation of different system 'views'
  - analysis
  - functional simulation
  - cycle-/instruction-accurate simulation
  - execution on hardware

[ref] K. Huang, W. Haid, I. Bacivarov, M. Keller, and L. Thiele. ***Embedding Formal Performance Analysis into the Design Cycle of MPSoCs for Real-time Multimedia Applications.*** *ACM TECS, Vol. 11, No. 1, pages 8:1-8:23, March, 2012.*
[ref] L. Schor, I. Bacivarov, D. Rai, H. Yang, S. Kang and L. Thiele, ***Scenario-Based Design Flow for Mapping Streaming Applications onto On-Chip Many-Core Systems****, CASES, Tampere, Finland, Oct. 2012*

Runtime System

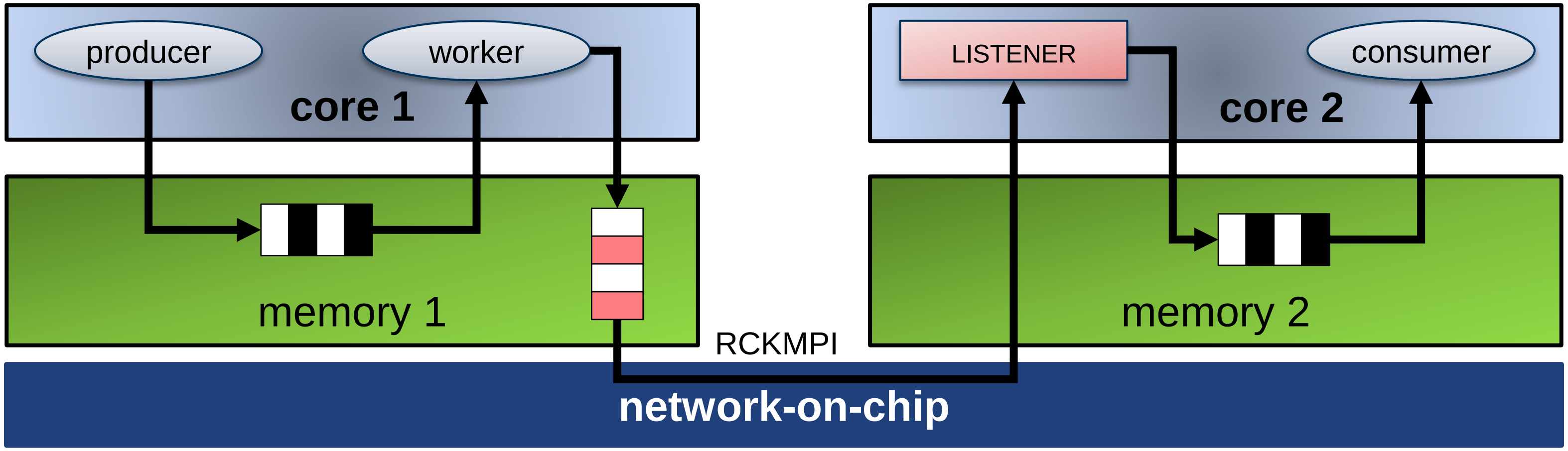
- provides an implementation of the programming interface
  - inter-process communication (distributed memory)
  - multi-processing mechanisms
  - services to manage processes and channels at runtime



[ref] L. Schor, D. Rai, H. Yang, I. Bacivarov, and L. Thiele, ***Reliable and Efficient Execution of Multiple Streaming Applications on Intel's SCC Processor.*** *Runtime and Operating Systems for the Many-core Era (ROME) August 2013.*

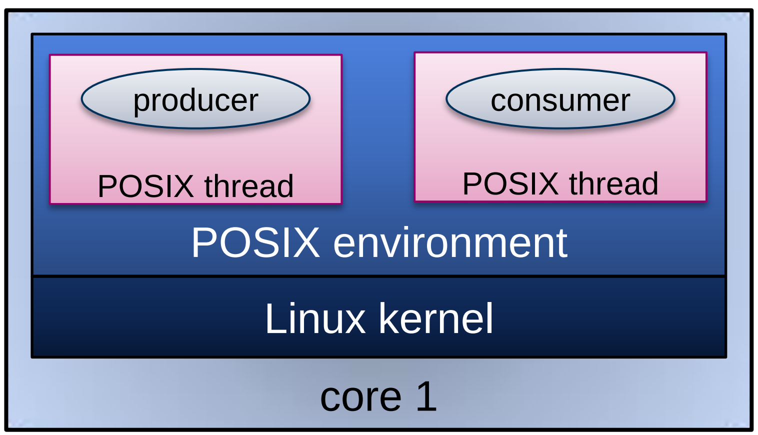
Inter-Process Communication

- shared vs. distributed memory
- on Intel SCC, RCKMPI lib. for inter-core communication
- one listener thread per core for all incoming traffic
- virtual buffer at sender to limit traffic



Multi Processing

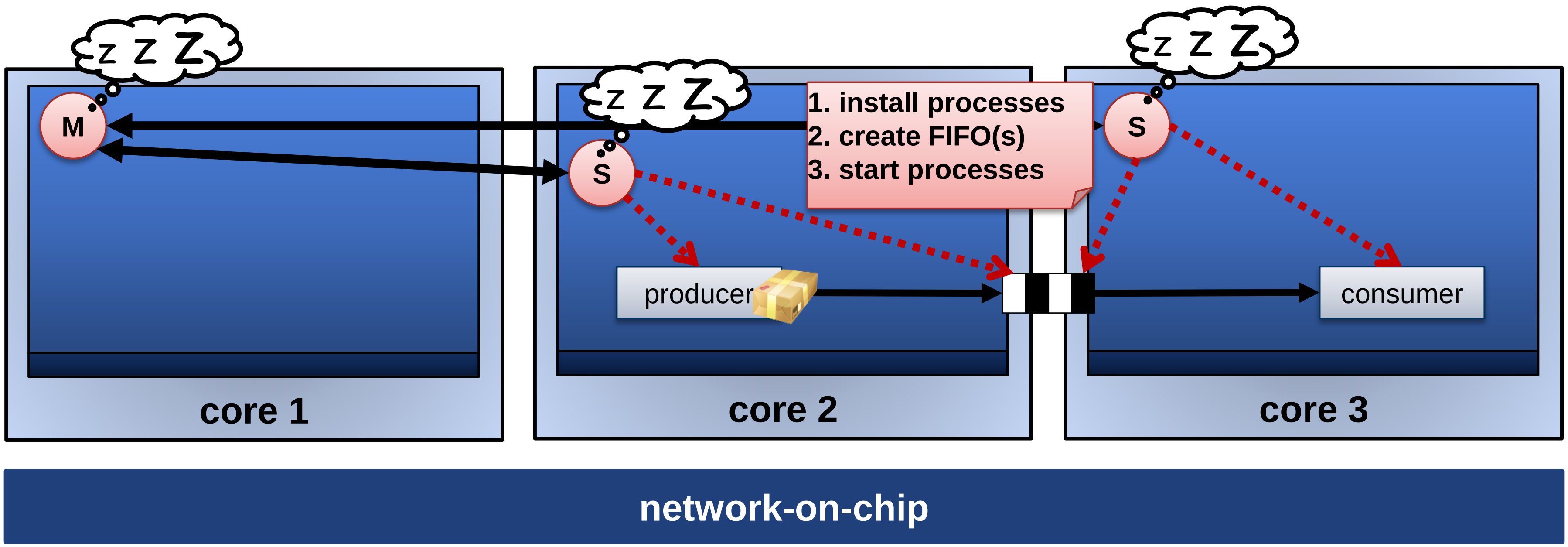
- on top of Linux kernel – processes mapped onto POSIX threads
- data-driven execution – no global scheduler required



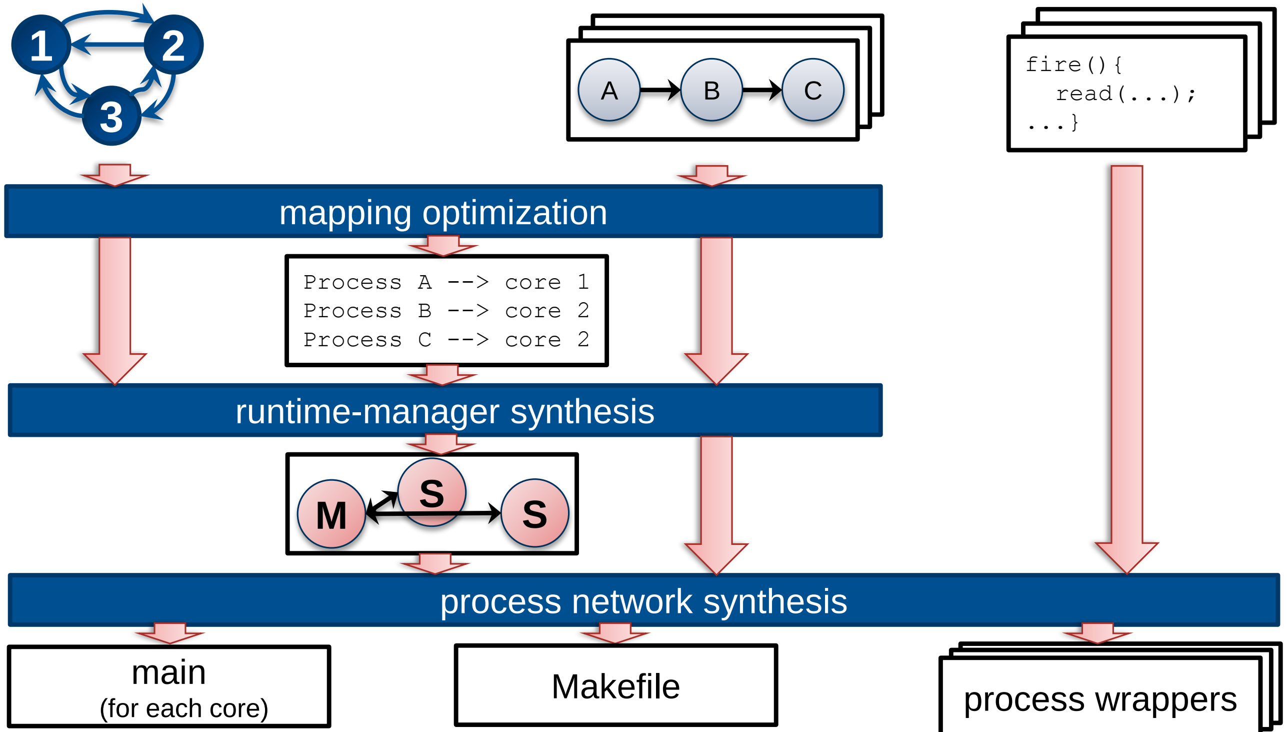

```
void *producer_thread
        (void *arg) {
    Process *p = (Process*) arg;
    while (!p->stopped) {
         p->fire();
    }
}
```

Runtime Manager

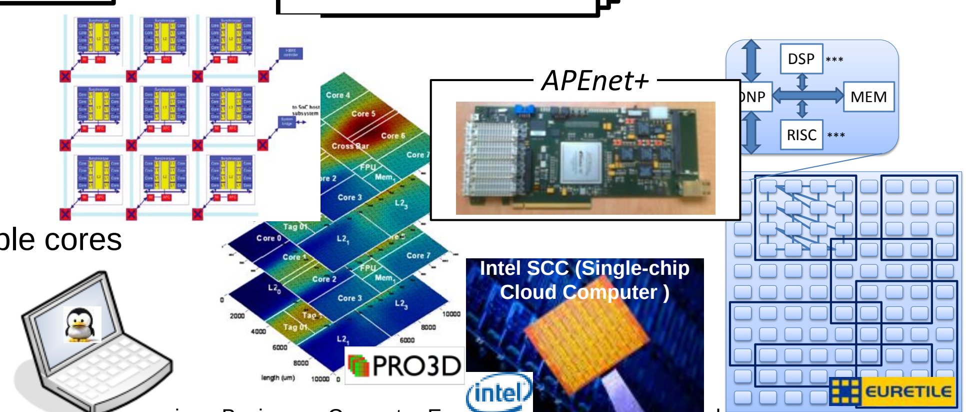
- specified as a process network
  - **one master process**: manages dynamic execution
  - **one slave process per core**: manage processes and channels



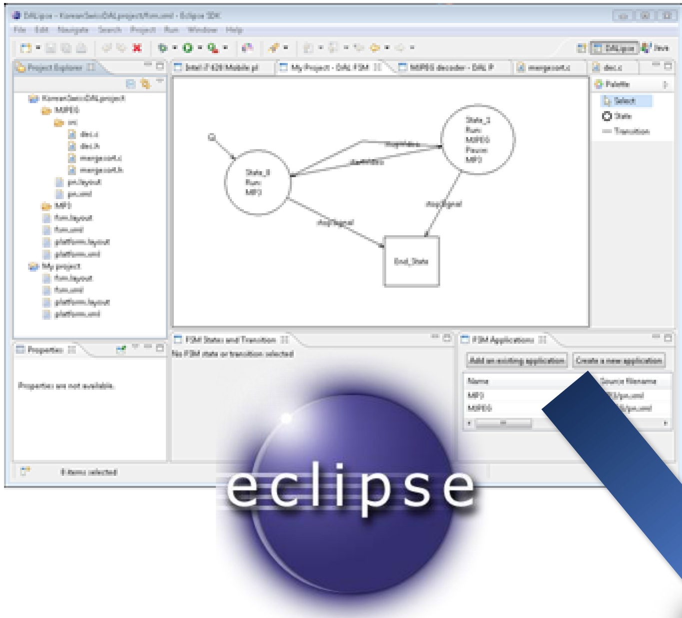
Synthesis Backend



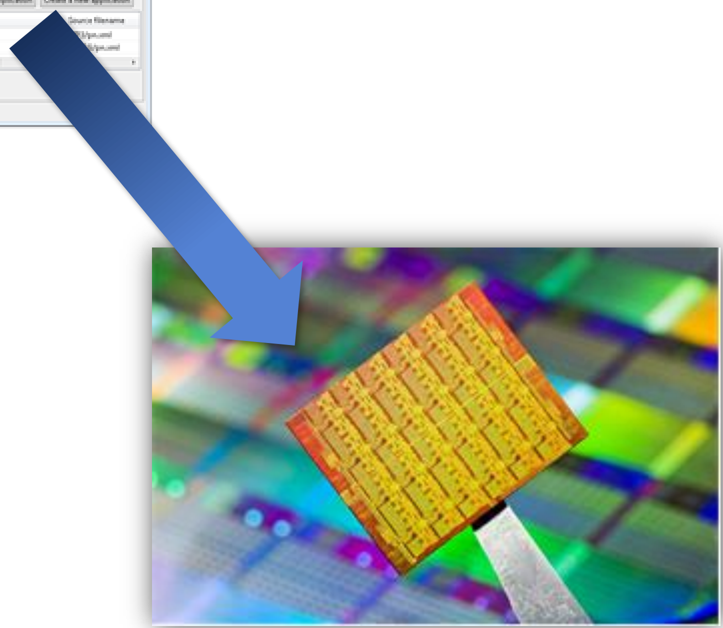
- **target platforms**
  - functional simulation on Linux
  - multi-cluster system:
    - each Linux server forms one cluster with multiple cores
    - Inter-cluster communication with MPI
  - Intel SCC
  - QUonG platform (INFN)

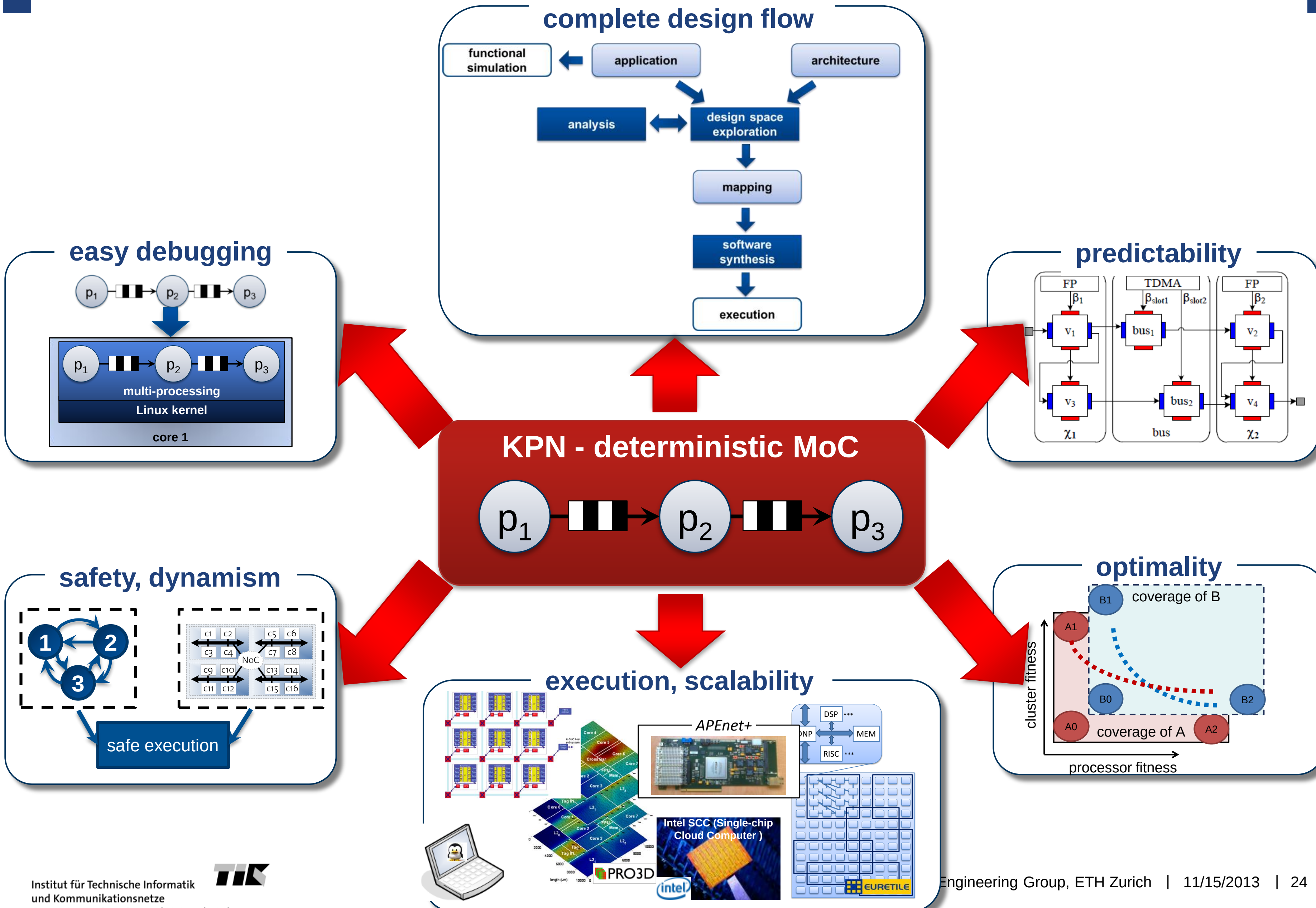


Deployment



DAL is available:
www.tik.ee.ethz.ch/~euretile/dal.php





complete design flow
functional simulation
application
architecture
analysis
design space exploration
mapping
software synthesis
execution
easy debugging
p1
p2
p3
multi-processing
Linux kernel
core 1
predictability
FP
TDMA
β1
βslot1
βslot2
β2
v1
bus1
v2
v3
bus2
v4
χ1
bus
χ2
KPN - deterministic MoC
p1
p2
p3
safety, dynamism
1
2
3
c1 c2 c5 c6
c3 c4 c7 c8
NoC
c9 c10 c13 c14
c11 c12 c15 c16
safe execution
optimality
coverage of B
B1
A1
B0
B2
A0
A2
coverage of A
cluster fitness
processor fitness
execution, scalability
APEnet+
DSP
DNP
MEM
RISC
Intel SCC (Single-chip Cloud Computer)
intel
PRO3D
EURETILE