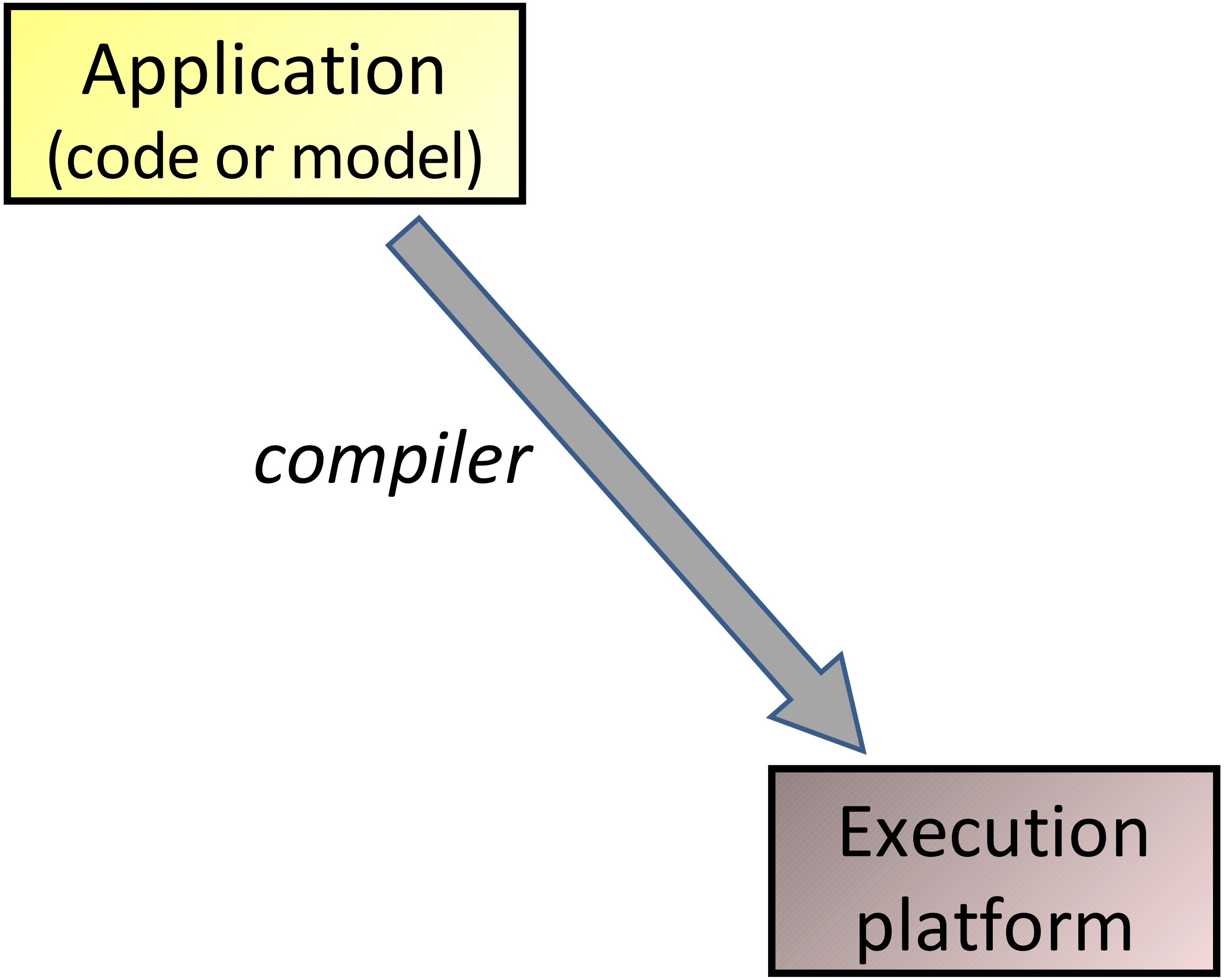
Formal model engineering for embedded applications and manycore architectures

Robert de Simone

INRIA Sophia Méditerranée
AOSTE team

Compilation (naive)

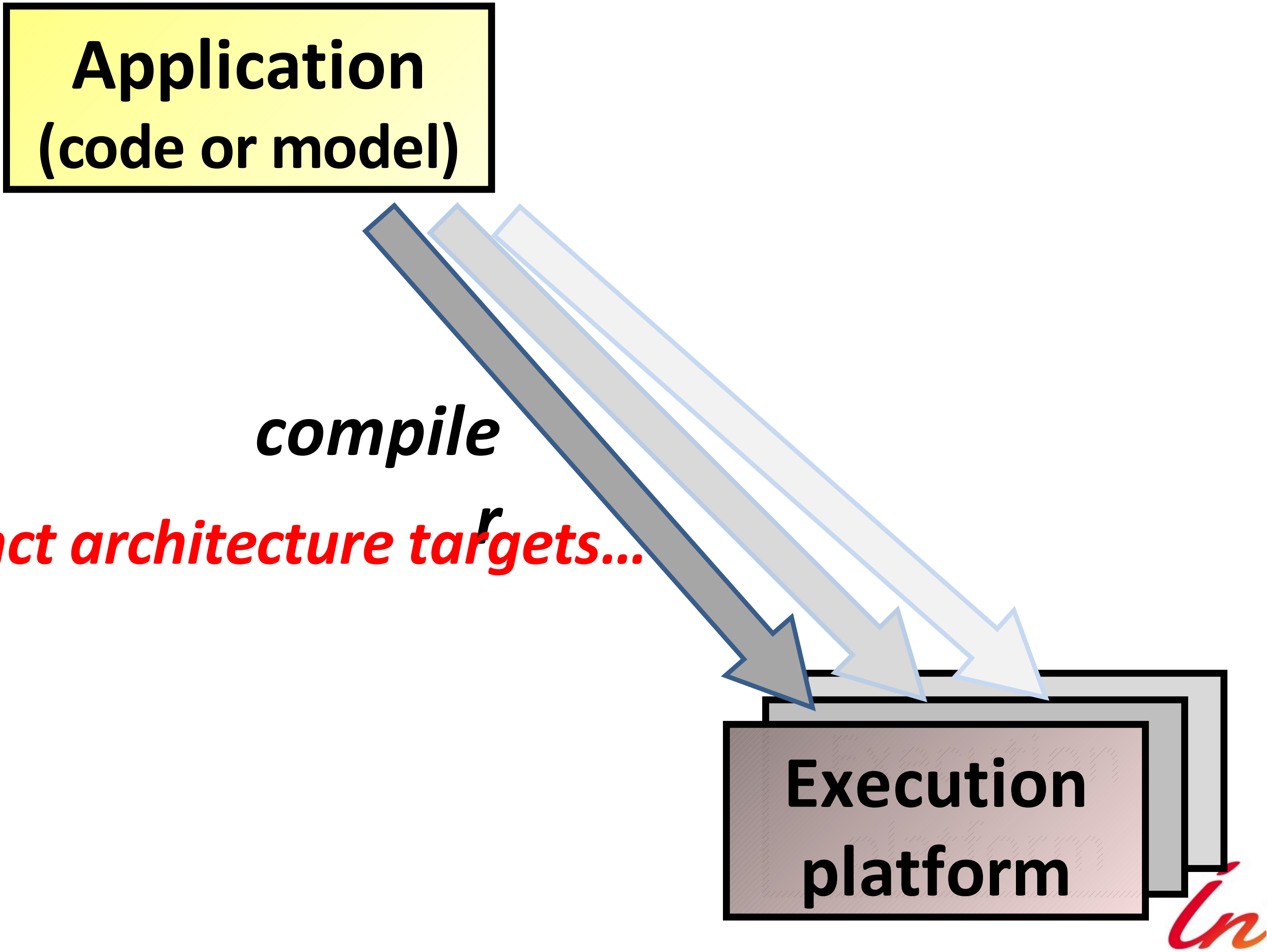


Application
(code or model)

compiler

Execution
platform

Compilation (less naive)



Application (code or model)

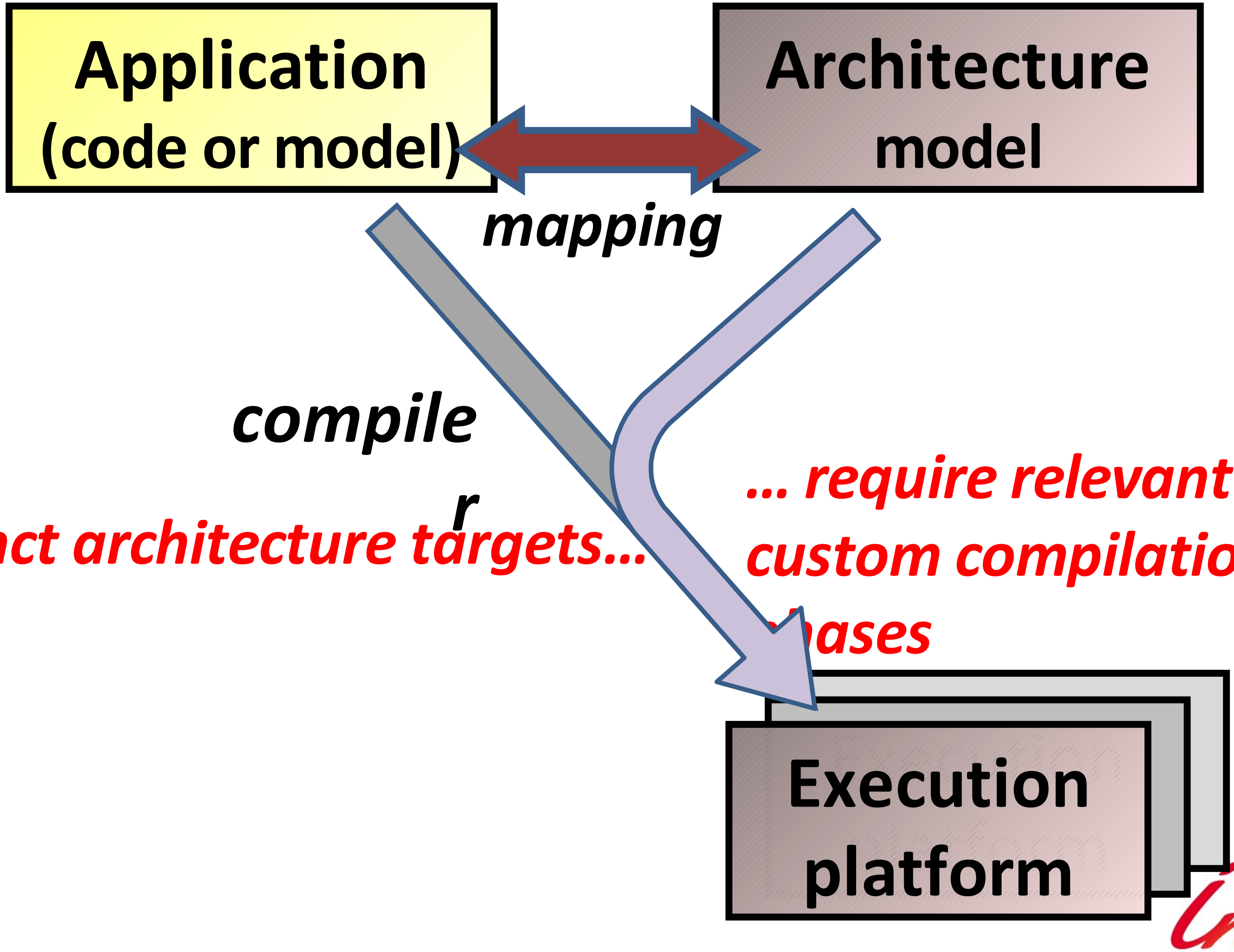
compile

r

Distinct architecture targets…

Execution platform

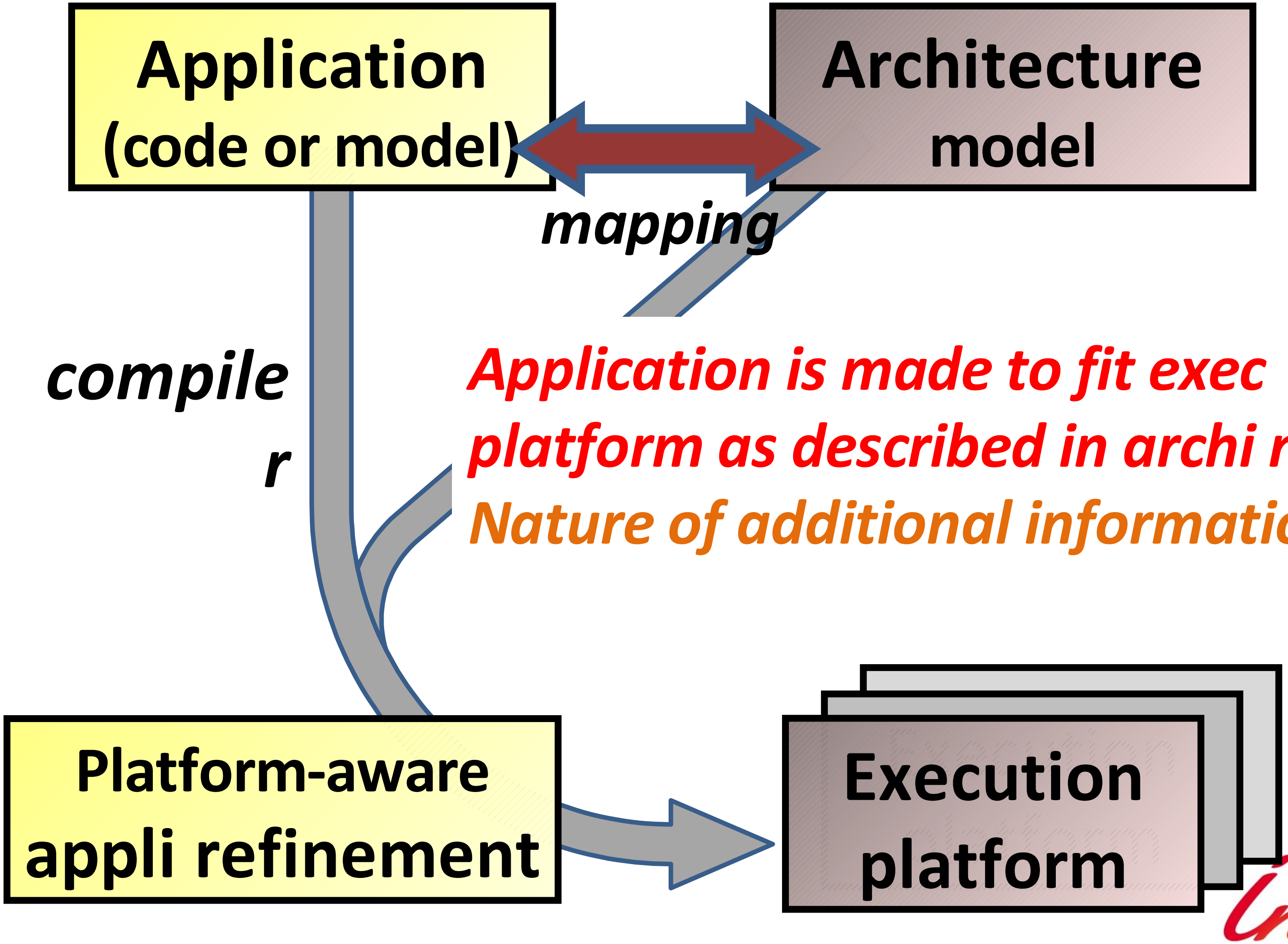
Compilation (less naive)



Distinct architecture targets…

… require relevant info for custom compilation phases

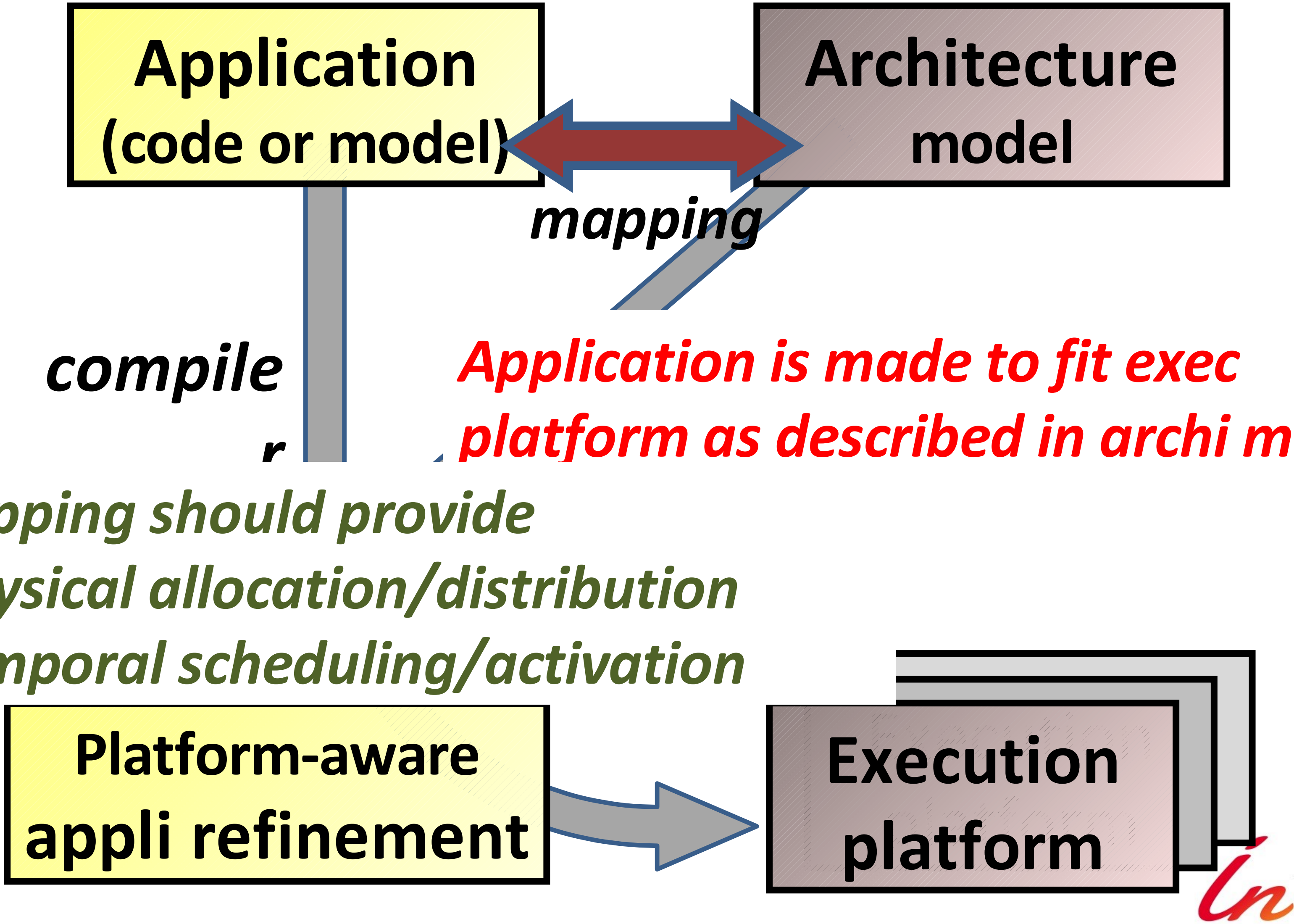
Compilation (less naive)



Application is made to fit exec platform as described in archi model:

Nature of additional information ?

Compilation (less naive)



Application is made to fit exec platform as described in archi model:

mapping should provide

- *physical allocation/distribution*
- *temporal scheduling/activation*

The case of embedded systems

- **Pioneering multicore parallel architectures** (homogeneous or heterogeneous)
- **Predictable, intensive data/signal processing applications** (streaming)
- **Co-design issue** (architectural design space exploration)
- **Statically co-mapping several applications onto joint architecture** (big issue in automotive/avionic domains)
- **Data transfers and communications become prominent , must be made time-predictable and efficient** (overlaps with on-chip networks topologies)

→ **Currently, compilation far from automatic** (user directives)

→***Need for explicit detailed architecture model***

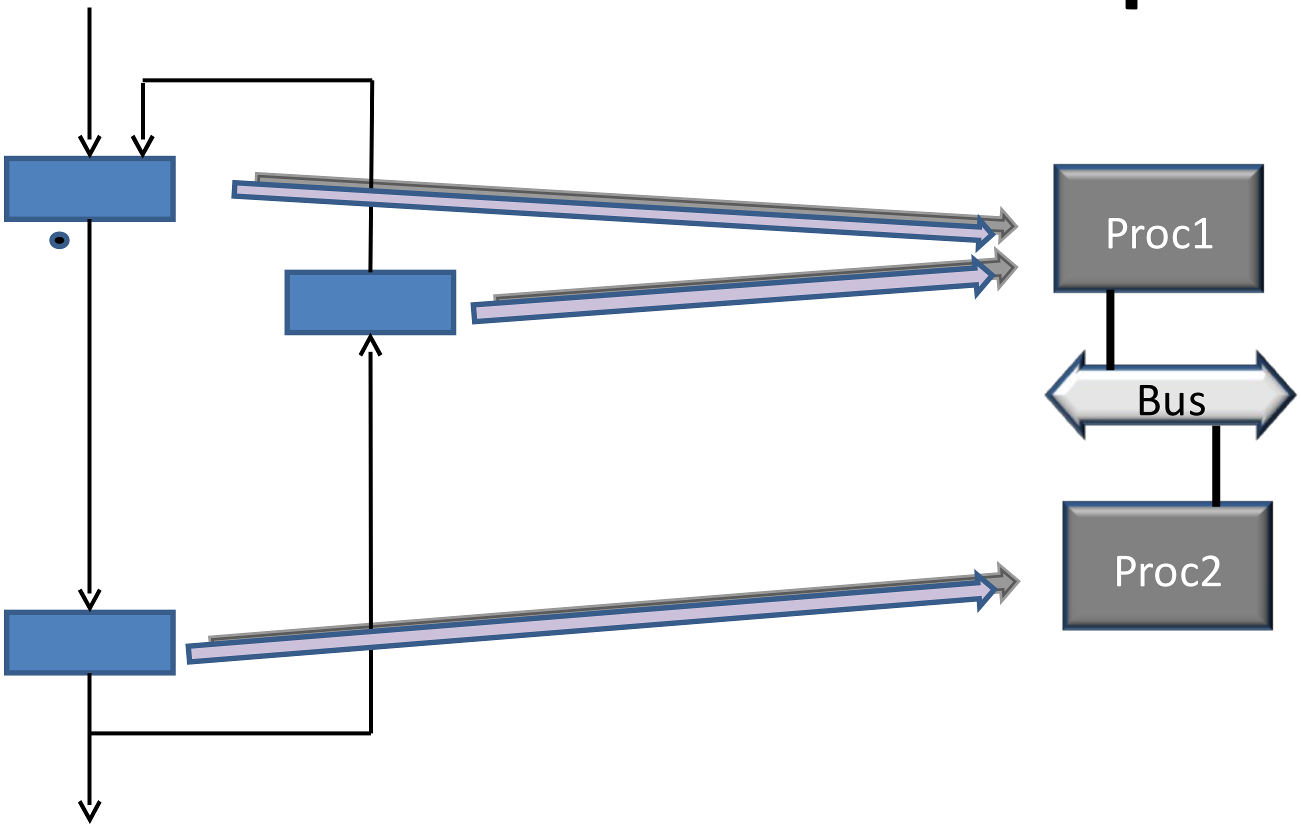
Main objective of current talk

- **Our goal:**

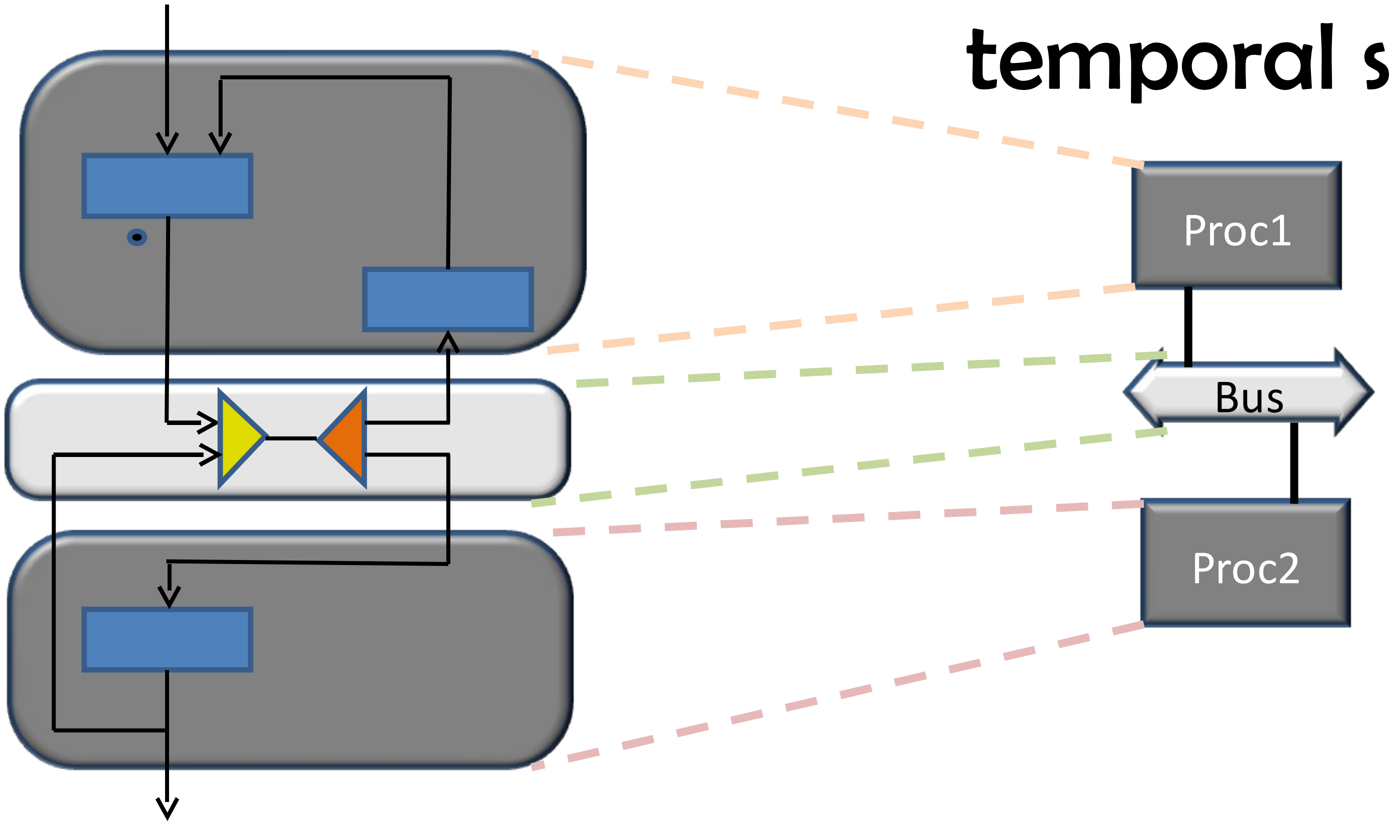
study how existing and emerging **formal models and methods** from theoretical computer science can be used to

- **Represent** the models
- **Specify**, and in some cases even **solve** the **optimized mapping** construction issue

Draft example: Mapping



Mapping = spatial distribution/routing + temporal scheduling



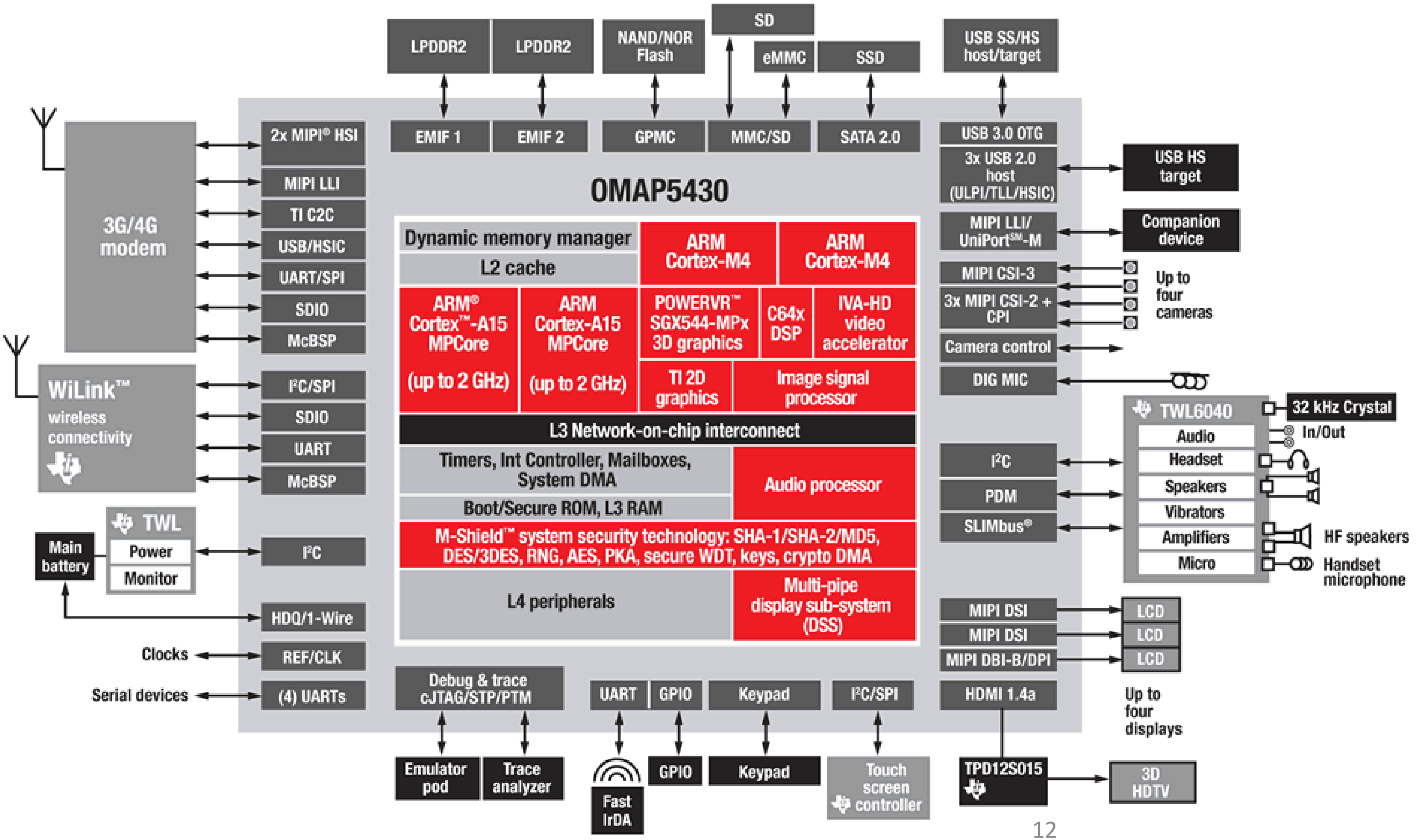
Typical embedded architecture

- Multi-, or even many-core in the future.
- Mix or generic CPUs and specific processors (GPUs, FPGAs, DSPs)
- Local memory (sometimes L1 caches) and global one.
- Power management for energy saving (battery life)
- Extra functional requirements
- Predicatble architecture ? (accurate WCET approx.)
- Growing importance of communication costs over computation ones

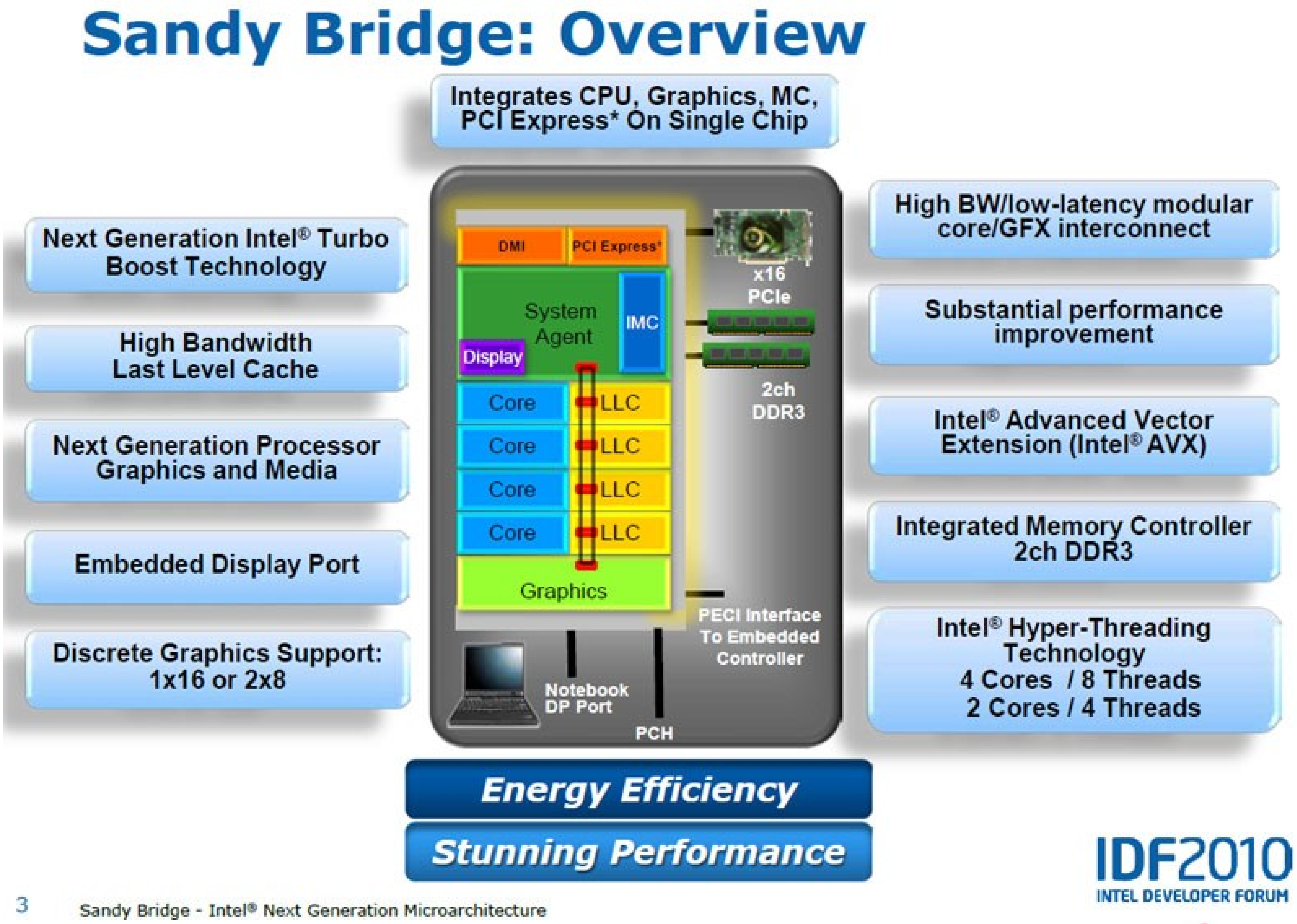
Currently such systems too often compute fast, then wait for so long the next available data, wasting power.

Target architecture (example)

TI OMAP5430 SoC



Target architecture (example)



Sandy Bridge: Overview

Integrates CPU, Graphics, MC, PCI Express* On Single Chip

Next Generation Intel® Turbo Boost Technology

High Bandwidth Last Level Cache

Next Generation Processor Graphics and Media

Embedded Display Port

Discrete Graphics Support: 1x16 or 2x8

High BW/low-latency modular core/GFX interconnect

Substantial performance improvement

Intel® Advanced Vector Extension (Intel® AVX)

Integrated Memory Controller 2ch DDR3

Intel® Hyper-Threading Technology
4 Cores / 8 Threads
2 Cores / 4 Threads

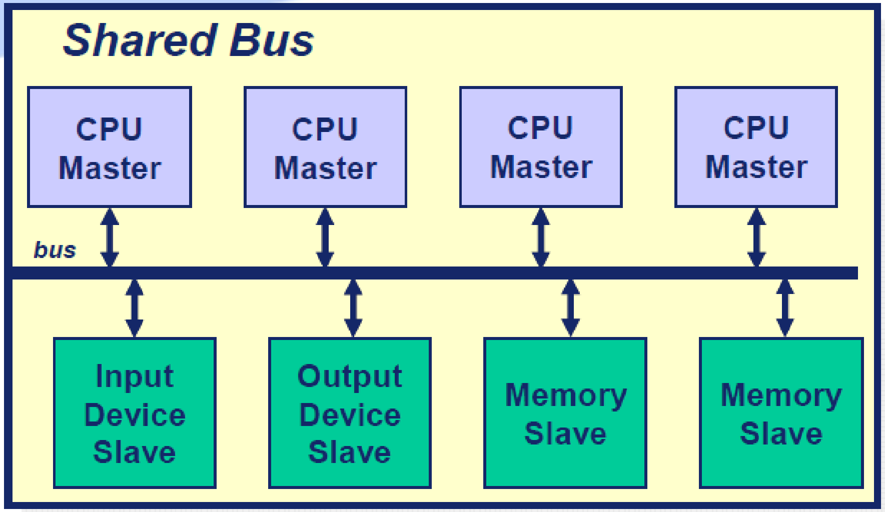
Energy Efficiency

Stunning Performance

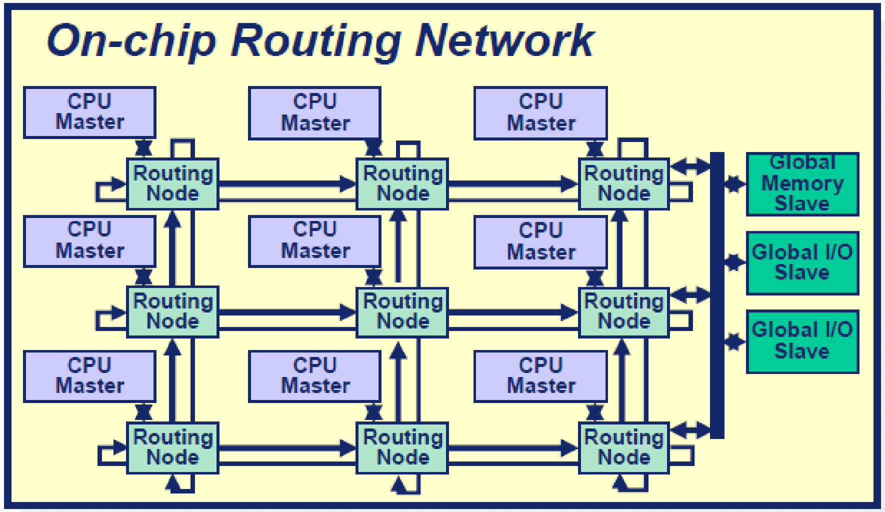
3 Sandy Bridge - Intel® Next Generation Microarchitecture

IDF2010
INTEL DEVELOPER FORUM

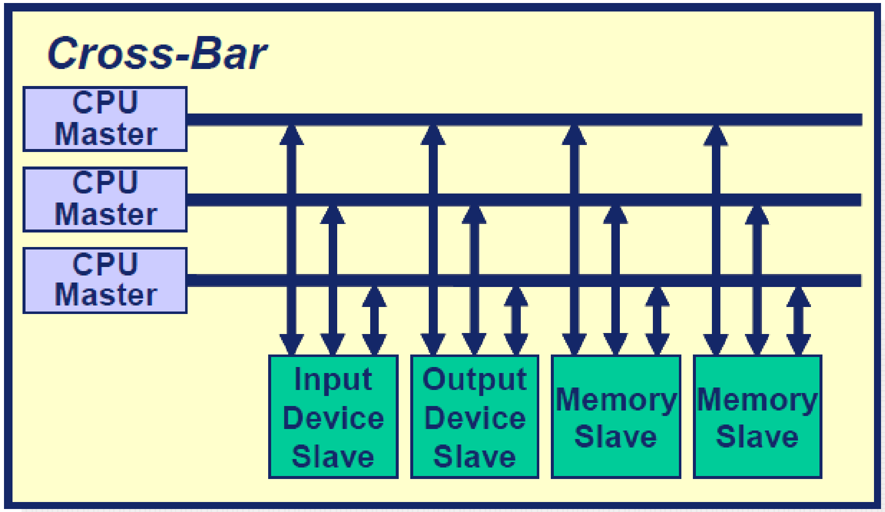
Don’t Forget the On-Chip Communications Architecture



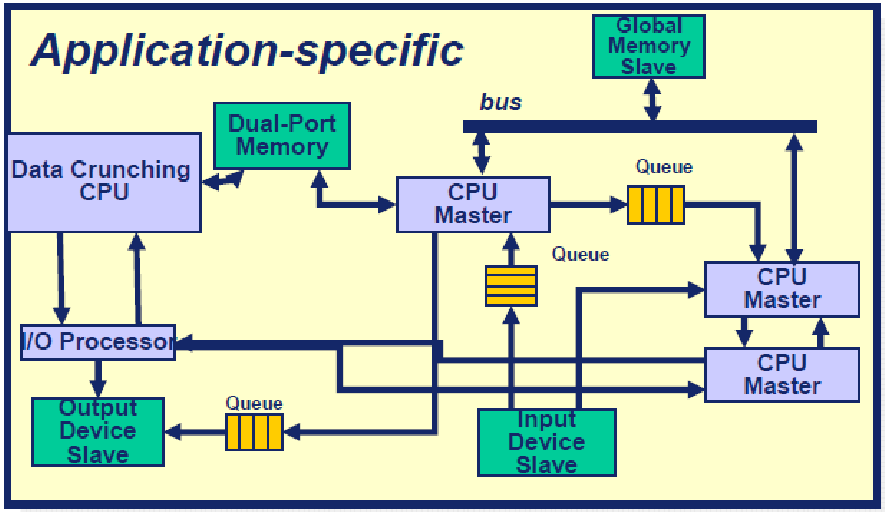
Shared Bus



On-chip Routing Network



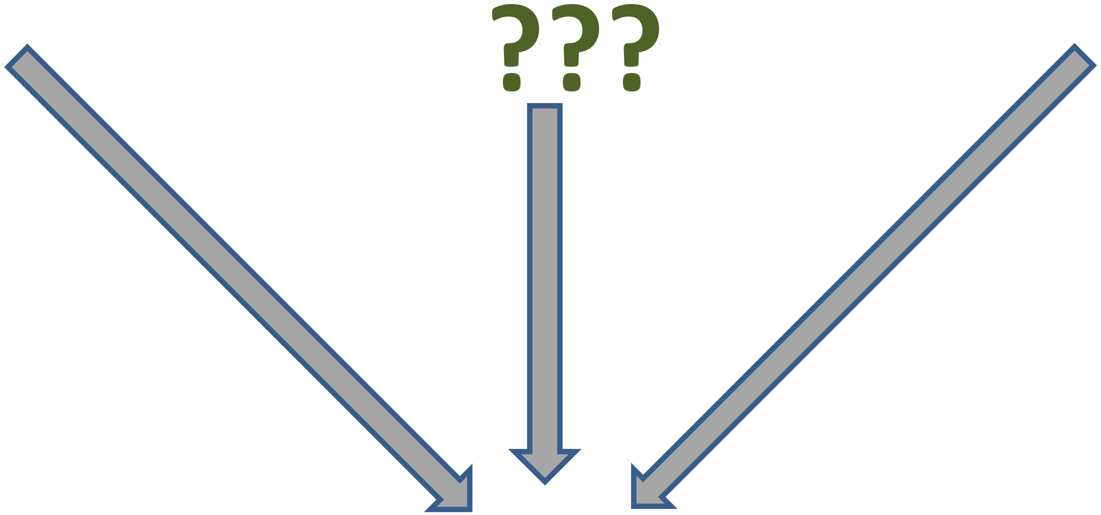
Cross-Bar



Application-specific

Our case: multicore (homo/heterogeneous) execution platforms

What kind(s) of Application models



???

Architecture = Network of Processors

Typical embedded application

- Individual components:
  - Intensive data/signal processing algorithms (FFT, JPEG, MPEG, GSM, LTE,…), filters

→ Nested loop programs (with fixed bounds)

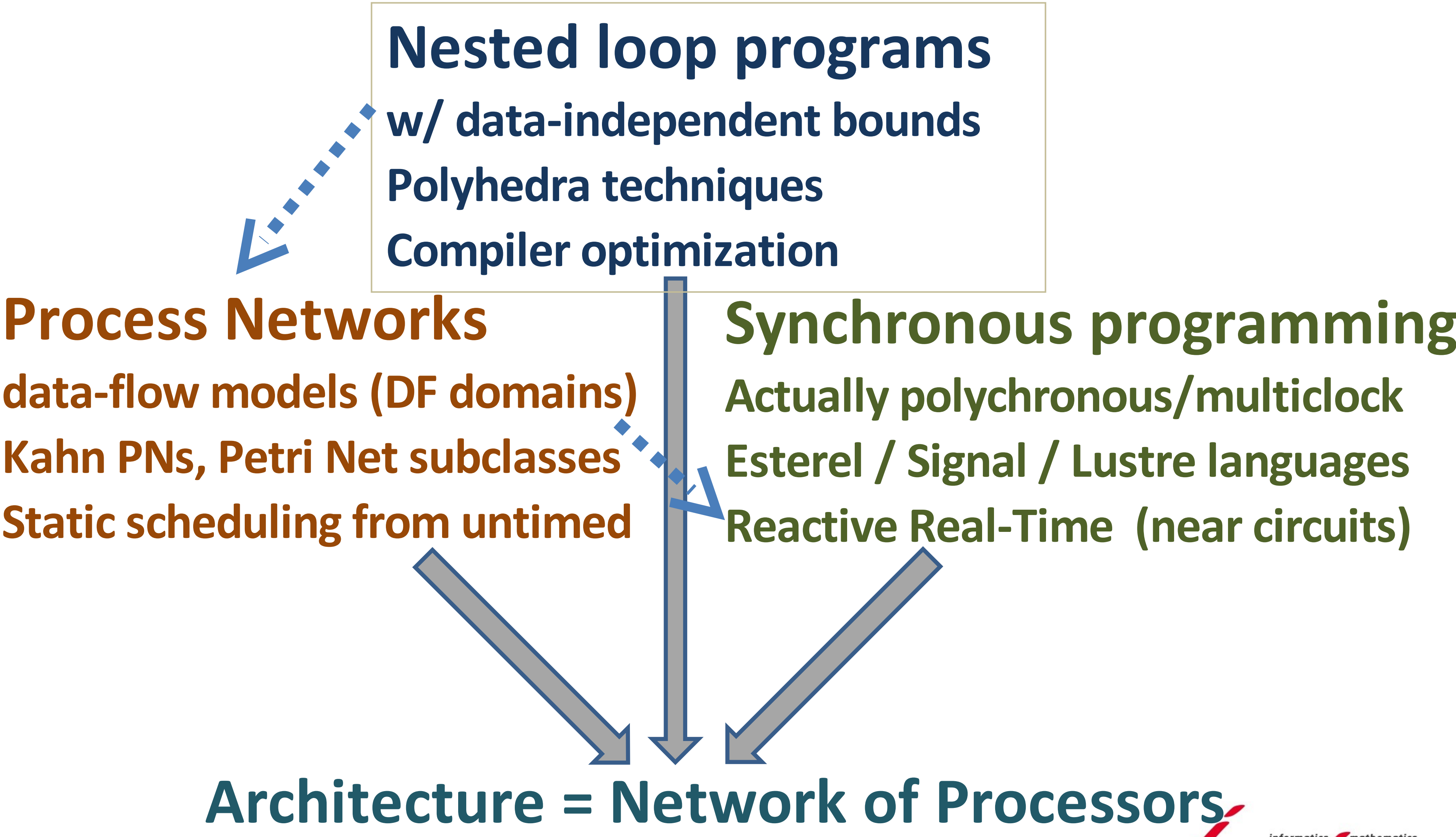
- Composition:
  - Streaming, pipelining, concurrent

→ Dataflow process networks

- Operational modes
  - Finite-state control-flow
  - Interrupts, context switches and logical time ordering as programming constructs

→ Synchronous languages

based on strong theories, developed partly at INRIA for the last thirty years



Nested loop programs

w/ data-independent bounds

Polyhedra techniques

Compiler optimization

Process Networks

data-flow models (DF domains)

Kahn PNs, Petri Net subclasses

Static scheduling from untimed

Synchronous programming

Actually polychronous/multiclock

Esterel / Signal / Lustre languages

Reactive Real-Time (near circuits)

Architecture = Network of Processors

Meeting points

- Nested loops programs implicitly assume a shared memory, loop transformation allow to extract parallelism (dopar/forall loops instead of doseq/for loops)
- Transformations to process networks need to optimize (distributed) memory transfert
- Process networks a genuine place of study for static regular scheduling and routing
- Explicit schedules are part of synchronous language modeling (also: nested loop programs refer to multidimensional logical time)
- Note: all these theories produce best results in the (theorerical) regular/rational domain:
  - Fixed (affine bounds for nested loops)
  - Static schedules and regular graph shape for process networks
  - Finite-state control and clock calculus for synchronous languages

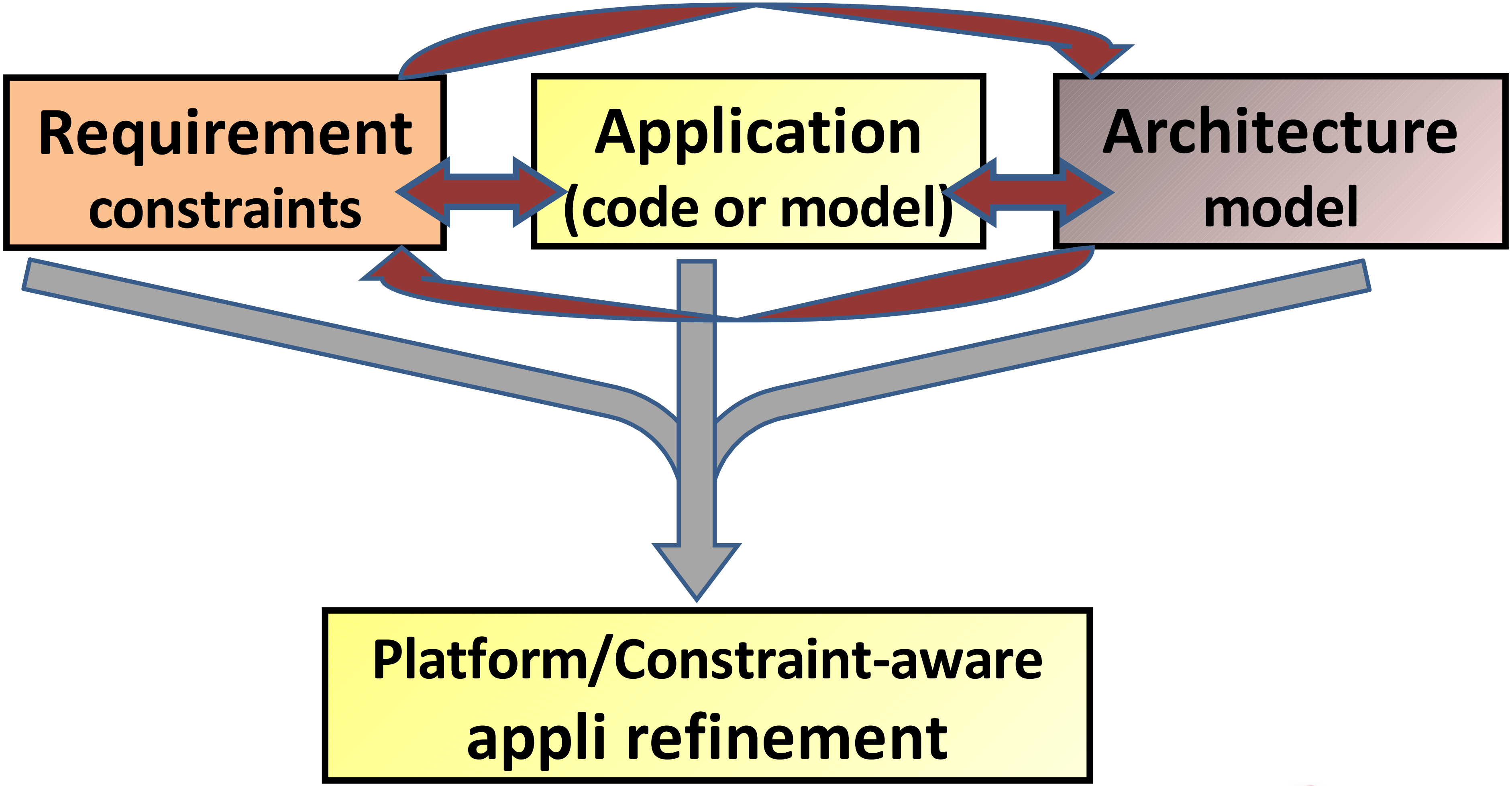
Bibliographic references (partial)

- **Nested loops: transformations**
  - A. Darte, Y. Robert, F. Vivien, loop parallelization algorithms, Compiler Optimizations for Scalable PS, chapter 6, LNCS 1808, 2001
- **Nested loops: scheduling**
  - Paul Feautrier. Some efficient solutions to the affine scheduling problem, I, one dimensional time. Int. I. of Parallel Programming, 21(5):October 1992.
  - Paul Feautrier. Some efficient solutions to the affine scheduling problem, II, multidimensional time. Int. I. of Parallel Programming, 21(6) December 1992.
- **Process networks**
  - Kahn, G. The semantics of a simple language for parallel programming, Proceedings of IFIP Congress 74,1974,
  - F. Commoner, Anatol W. Holt, Shimon Even, Amir Pnueli: Marked Directed Graphs. J. Comput. Syst. Sci. 5(5) (1971)
  - Lee, E. and Park, T. Dataflow Process Networks. In Proceedings of the IEEE(1995).
- **Process network scheduling**
  - J. Carlier et Ph. Chrétienne, Problèmes d'ordonnancement : modélisation, complexité, algorithmes, Masson, Paris, 1988
  - A. Cohen, M. Duranton, Ch. Eisenbeis, C. Pagetti, F. Plateau, and M. Pouzet, N-synchronous Kahn networks: a relaxed model of synchrony for real-time systems. (POPL '06)
  - Julien Boucaron, Anthony Coadou, Robert de Simone. Formal Modeling of Embedded Systems with Explicit Schedules and Routes. In Synthesis of Embedded Software: Frameworks and Methodologies for Correctness by Construction, Chapter 2, Springer Science+Business Media, LLC 2010, July 2010.

- **Synchronous languages, scheduling**
  - Irina M. Smarandache, Paul Le Guernic: Affine Transformations in SIGNAL and Their Application in the Specification and Validation of Real-Time Systems. ARTS 1997
  - Charles André, Julien DeAntoni, Frédéric Mallet, Robert de Simone. The Time Model of Logical Clocks available in the OMG MARTE profile. In Synthesis of Embedded Software: Frameworks and Methodologies for Correctness by Construction, Chap. 7, Springer Science+Business Media, LLC 2010, July 2010.
  - Dumitru Potop-Butucaru, Robert de Simone, Yves Sorel. From Synchronous Specifications to Statically-Scheduled Hard Real-Time Implementations. In Synthesis of Embedded Software: Frameworks and Methodologies for Correctness by Construction, Chapter 8, Springer Science+Business Media, LLC 2010, July 2010.

Model-based design flow

alternative to v-cycle


Requirement constraints
Application (code or model)
Architecture model
Platform/Constraint-aware appli refinement

What models ?

Engineering models
Design flow, specs

Formal models
Formal methods

Programming models
Truly: coding style

What models ?

Engineering models

Design flow, specs

Integration + translations
Complete design flow
Requirements → Specs → Integration → tests&verif
Generic aspects

Formal models

Formal methods

Specific aspects
Dedicated analysis
Efficient algorithmic issues

Programming models

Truly: coding style

Implementation or simulation (distinct aims)

Models or code ?

From models to code :

High-level modeling, synthesis, simulation as executable specifications, inclusion of non-functional checks "at runtime" Ex: SystemC

From code to models :

Low-level modeling (intermediate compilation formats), operational semantics, interpretation of specific features Ex: OpenMP, OpenCL

Formal models
Formal methods

Programming models
Truly: coding style

Formal models methods, and tools

Process algebras
(rather state-based, action-labeled)

- CCS, CSP, Lotos, …, Promela

Process networks
(rather activity-based, data-flows)

- Petri, Kahn PNs, SDF,…

Model-checking + exhaustive verification

Set/Logic theory
(temporal behav encoding)

- B method, TLA+, COQ, …

Property-checking + theorem proving

Timed models

- Timed, Hybrid automata

Stochastic/probabilistic models

- Markov chains, Queuing theories…

Combining modern disciplines from Theoretical Computer Science

- Model-Based Engineering
- Concurrency Theory
- Scheduling Theory
- High-Performance Computing
- Compiler Optimization
- Formal verification, validation, certification
- Performance and Timing Analysis
- Requirement engineering and Non-Functional aspects (temperature, power, autonomy, security)

Parallelizing compilers on polyhedral data spaces for High-Performance

AFFINE BOUNDS NESTED LOOPS

Loop transformations

```
DO i=1,n
   DO j=1,n
       a(i,j)=a(i-1,j-1) + a(i,j-1)
   END
END
```

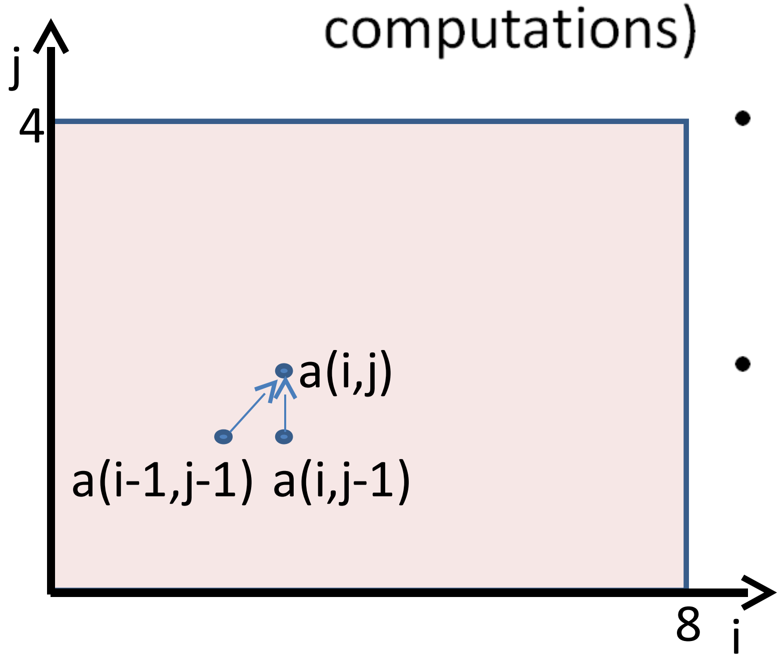
```
DOSEQ j=1,n
   DOPAR i=1,n
       a(i,j)=a(i-1,j-1) + a(i,j-1)
   END
END
```

- Many kinds of transformations studied to exhibit as much parallelism as possible
- Usually rely on Reduced Dependency Graphs with proper distance labeling
- In the end one tries to get increasing control variables (**i,j**,…), ranging in polyhedral spaces, so that statements executed simultaneously can be defined as slices in hyperplanes orthogonal to the progressions (rough sketchy idea)
- A scheduling in an integer assignement of a logical date to any operation (the corresponding assignment then receives a multidimensional schedule, because of possible simultaneity)

From A. Darte, Y. Robert, F. Vivien'reference article

Dream (or nightmare?)

- How can one produce Process Network descriptions from nested loops (or better said, SUREs or RDGs)
- Of course limited, but very often already polyhedra (for bounds) separated from dependency graphs (for computations)

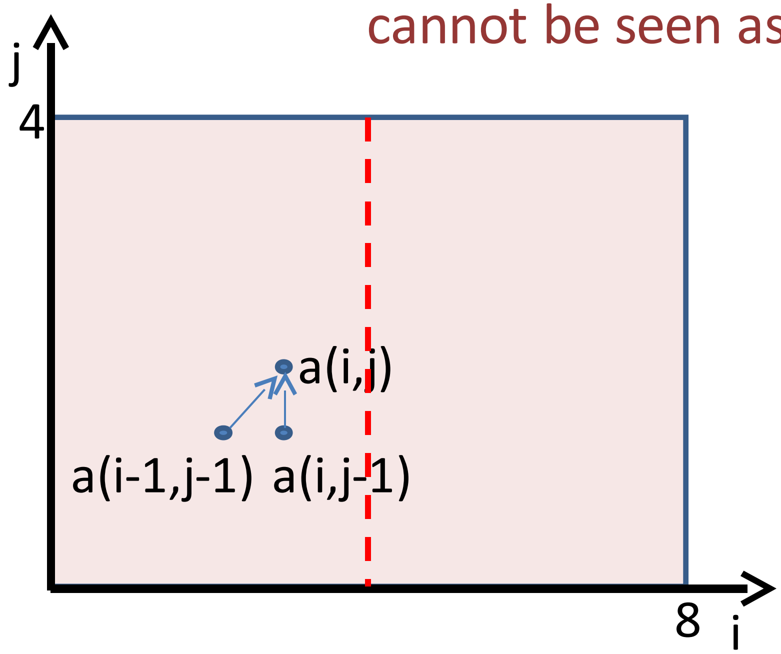


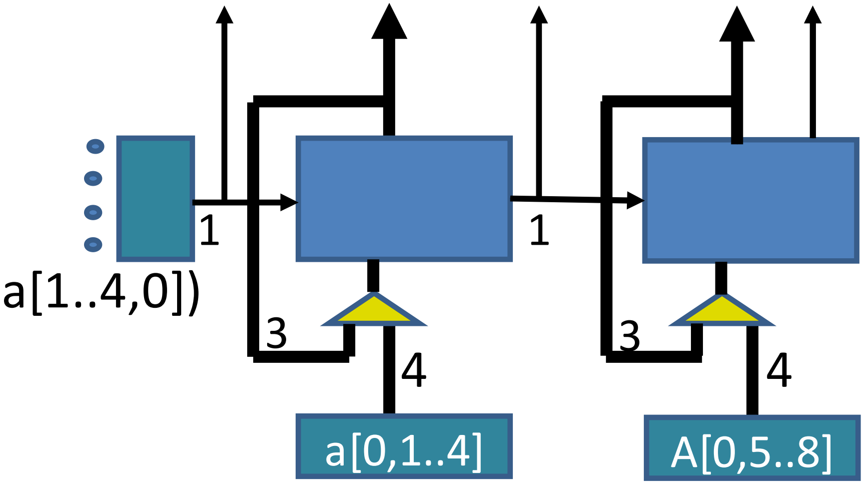
- Needs to split clearly between (sequential) iterations and parallel ones (currently expanded)
- Issue of potential (application) vs real (architecture) concurrency/parallelism

Dream (or nightmare?)

- Greenish nodes provide constants (in parallel at the bottom, sequentially on the left)
- Blue nodes each compute 4 values in parallel, shifts the last right across the border

Lack of generality certainly when values to be sent across cannot be seen as simply buffered.





Thank you

Questions ?