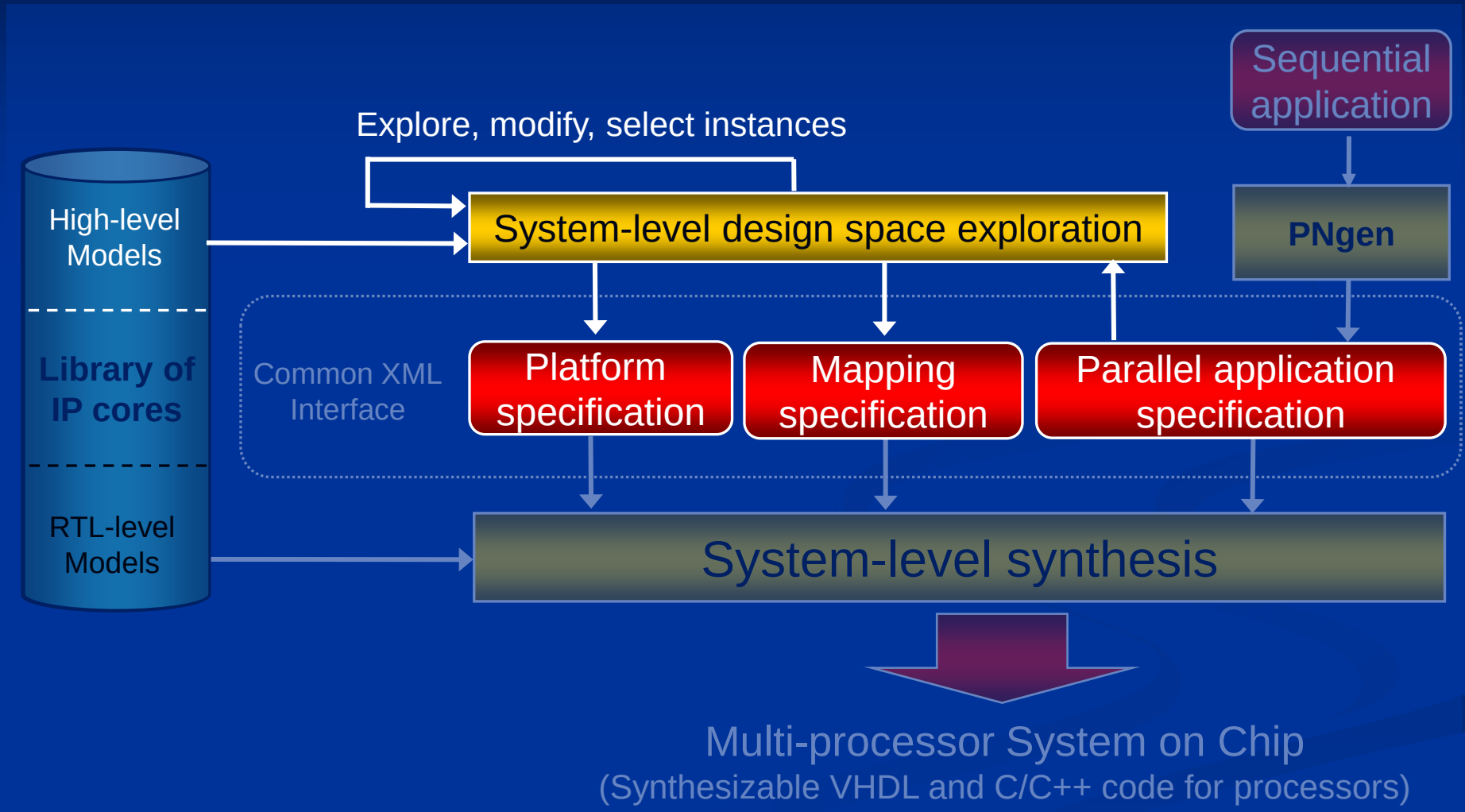
Design Space Exploration

Todor Stefanov

Leiden Embedded Research Center,
Leiden Institute of Advanced Computer Science
Leiden University, The Netherlands

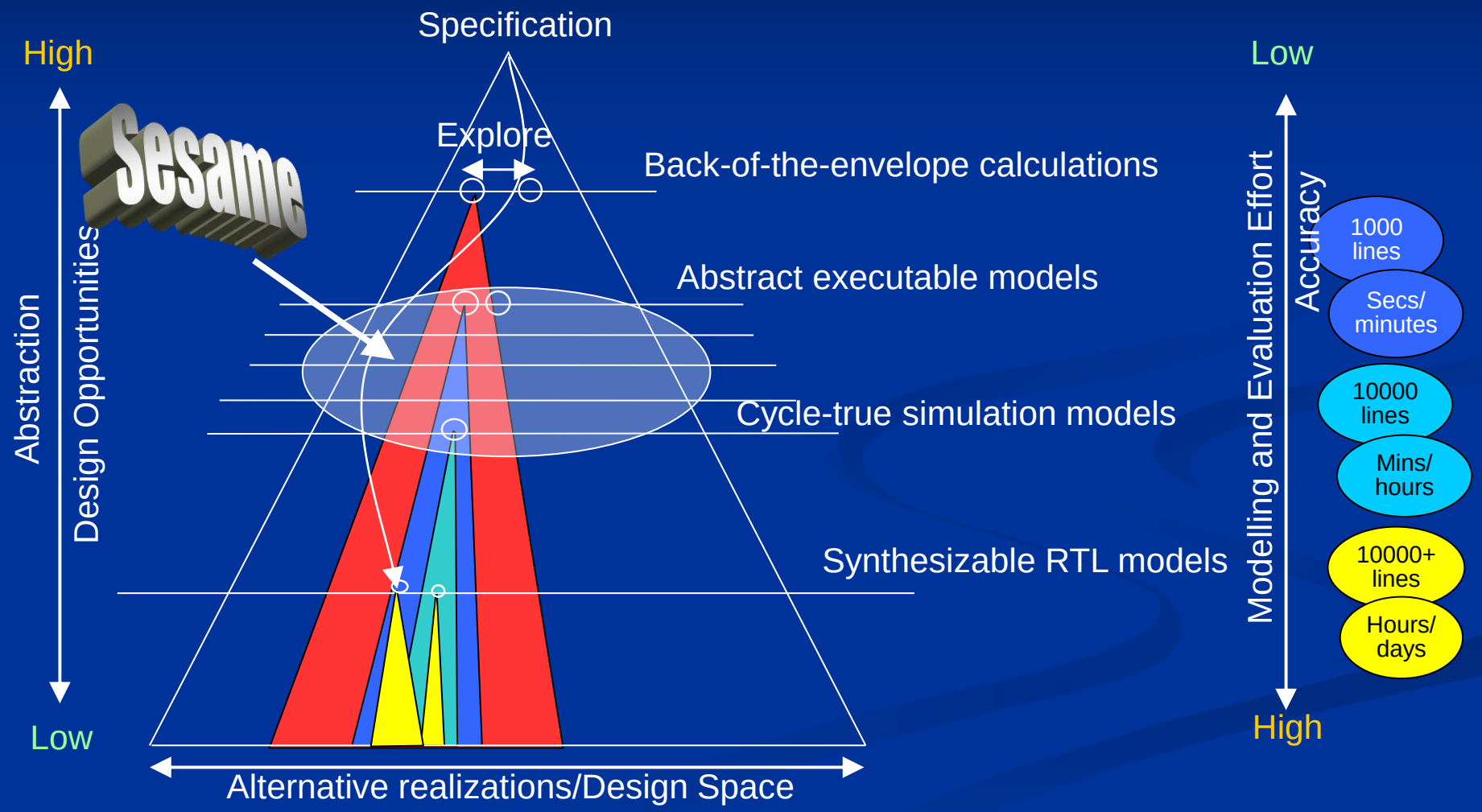
Design Space Exploration: SESAME tool



How to do DSE?

- Many designers rely on detailed models and simulators for DSE
- Approach infeasible for early design stages
  - Effort to build detailed models is too high
    - systems are too complex
  - Low speeds of detailed simulators
    - seriously hamper system exploration
- Many design alternatives must be explored
  - modelling and evaluation effort must be low
  - accuracy must be sufficient to compare design points

Abstraction levels of DSE in Daedalus

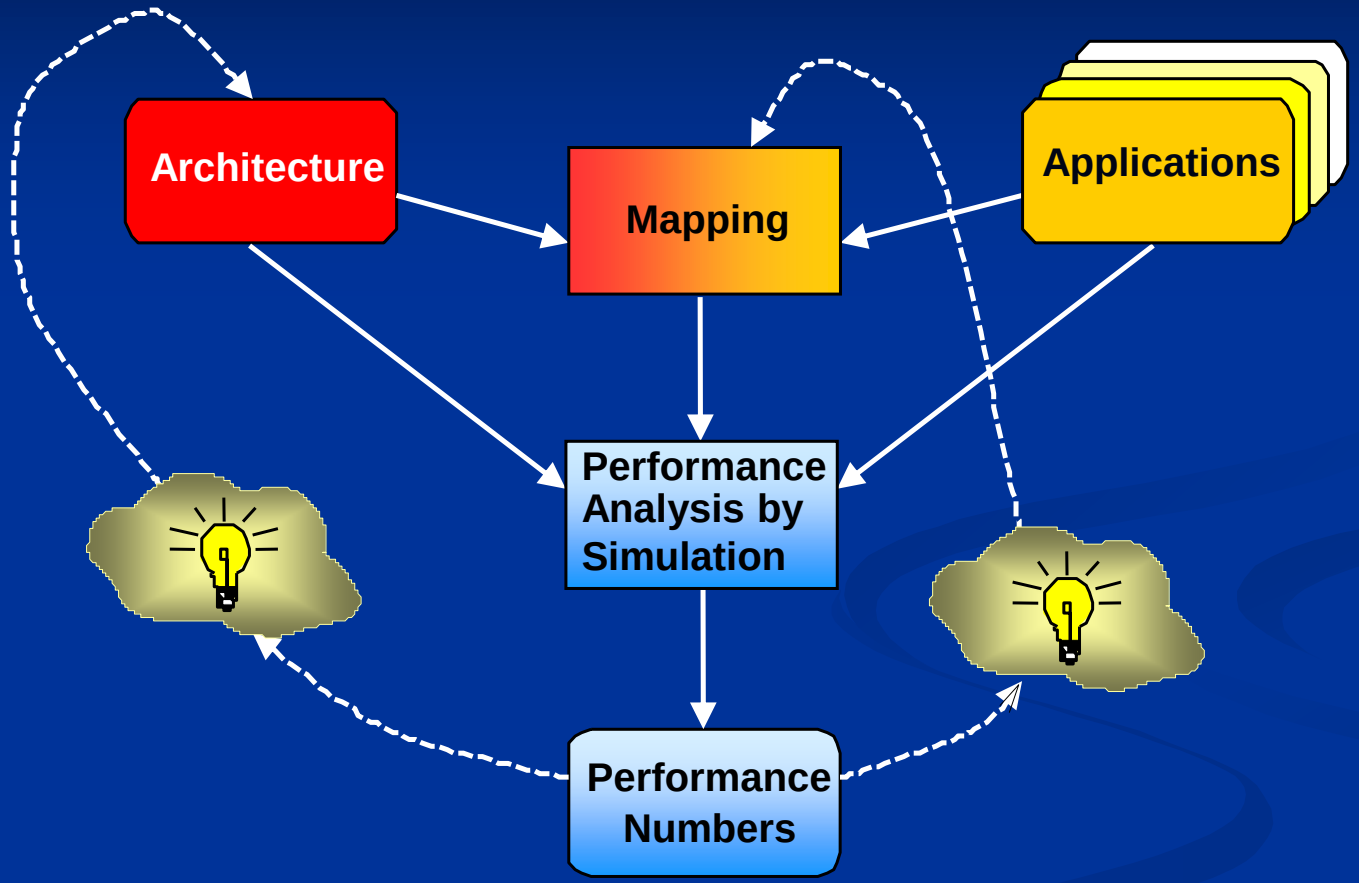


Sesame

Simulation of Embedded Systems Architectures for Multi-level Exploration

- Provides methods and tools for efficient DSE of heterogeneous MPSoCs
  - Different architectures, applications, and mappings
  - Different HW/SW partitionings
  - Mixed-level simulations
- Promotes reuse of models
- Targets streaming applications
  - Techniques and tools also applicable to other application domains

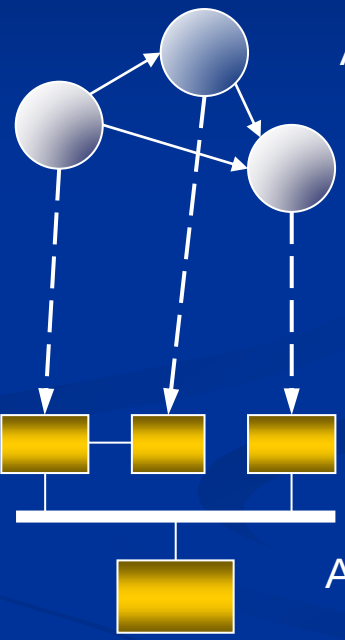
Sesame implements Y-chart Design Methodology



Use separate models for application and architecture behavior

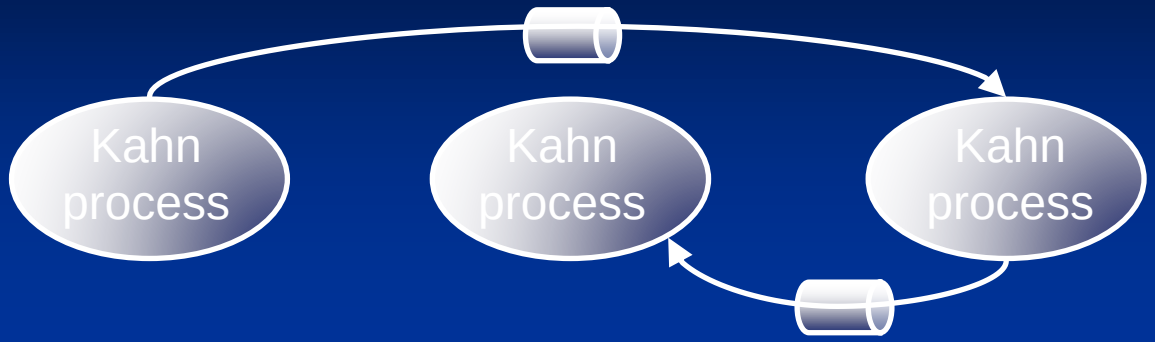
Modeling and simulation using Y-Chart methodology

- Application model in Sesame
  - Describes only functional behavior of application
  - Independent from architecture, HW/SW partitioning and timing characteristics
  - Generates application events representing the workload imposed on the architecture
- Architecture model in Sesame
  - Specifies parameterized timing behaviour of architecture components
  - Models timing consequences of application events

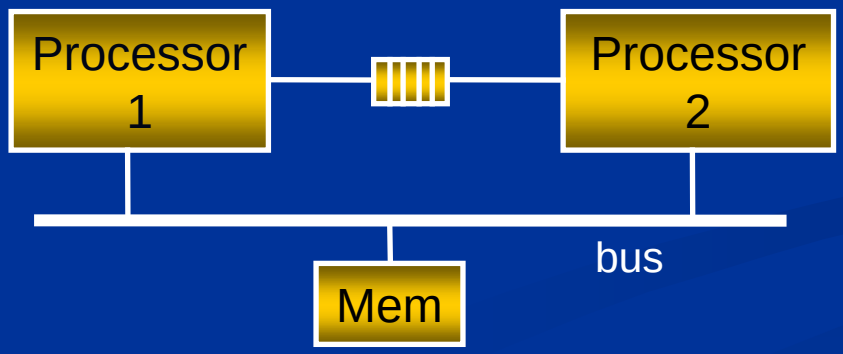


- Explicit mapping of application and architecture models
  - Trace-driven co-simulation
  - Easy reuse of both application and architecture models!

The Sesame framework layers

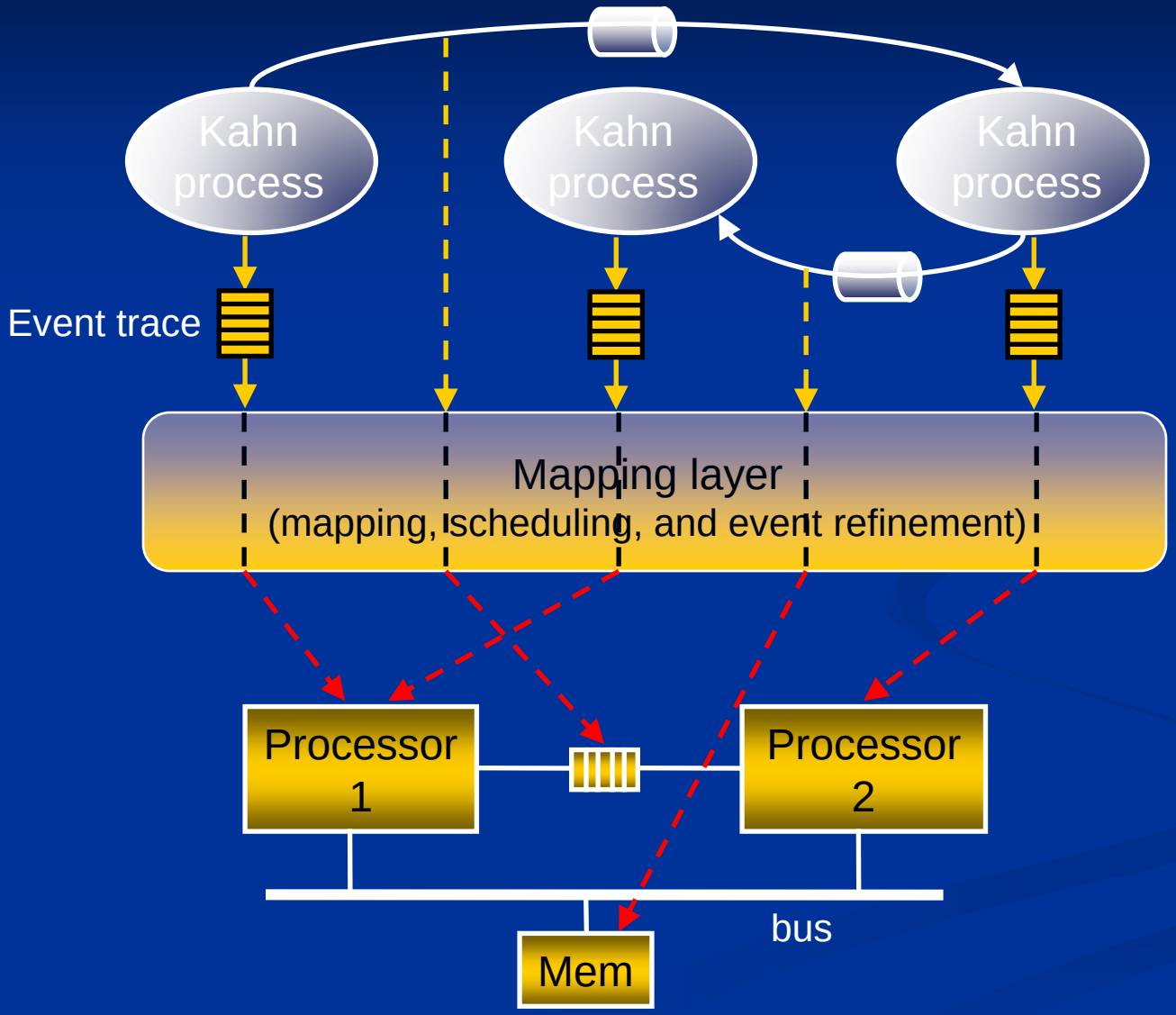


Application layer
(Kahn Process Network, functional behavior)



Architecture layer
(discrete event simulation, performance behavior)

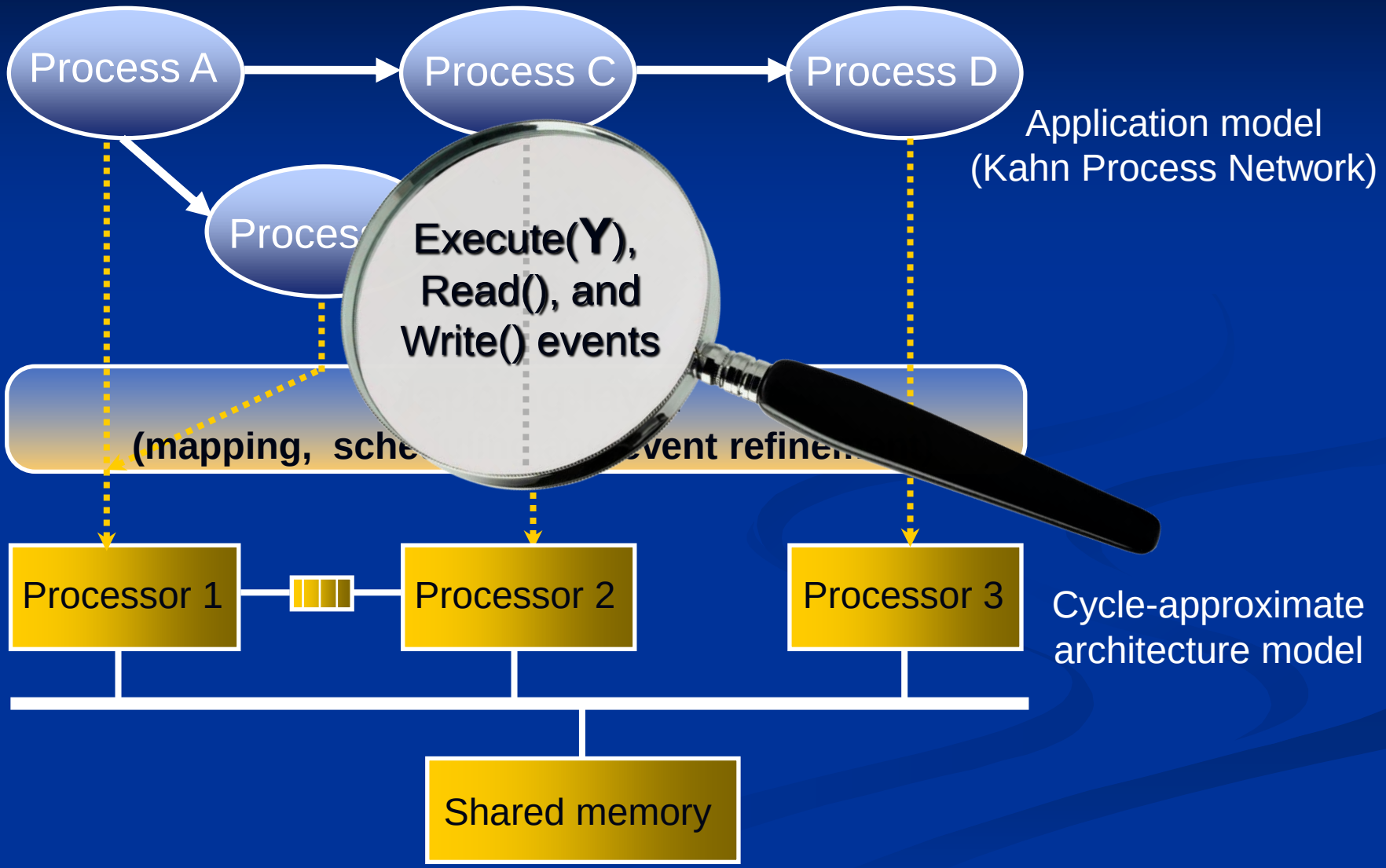
The Sesame framework layers



Application layer
(Kahn Process Network, functional behavior)

Architecture layer
(discrete event simulation, performance behavior)

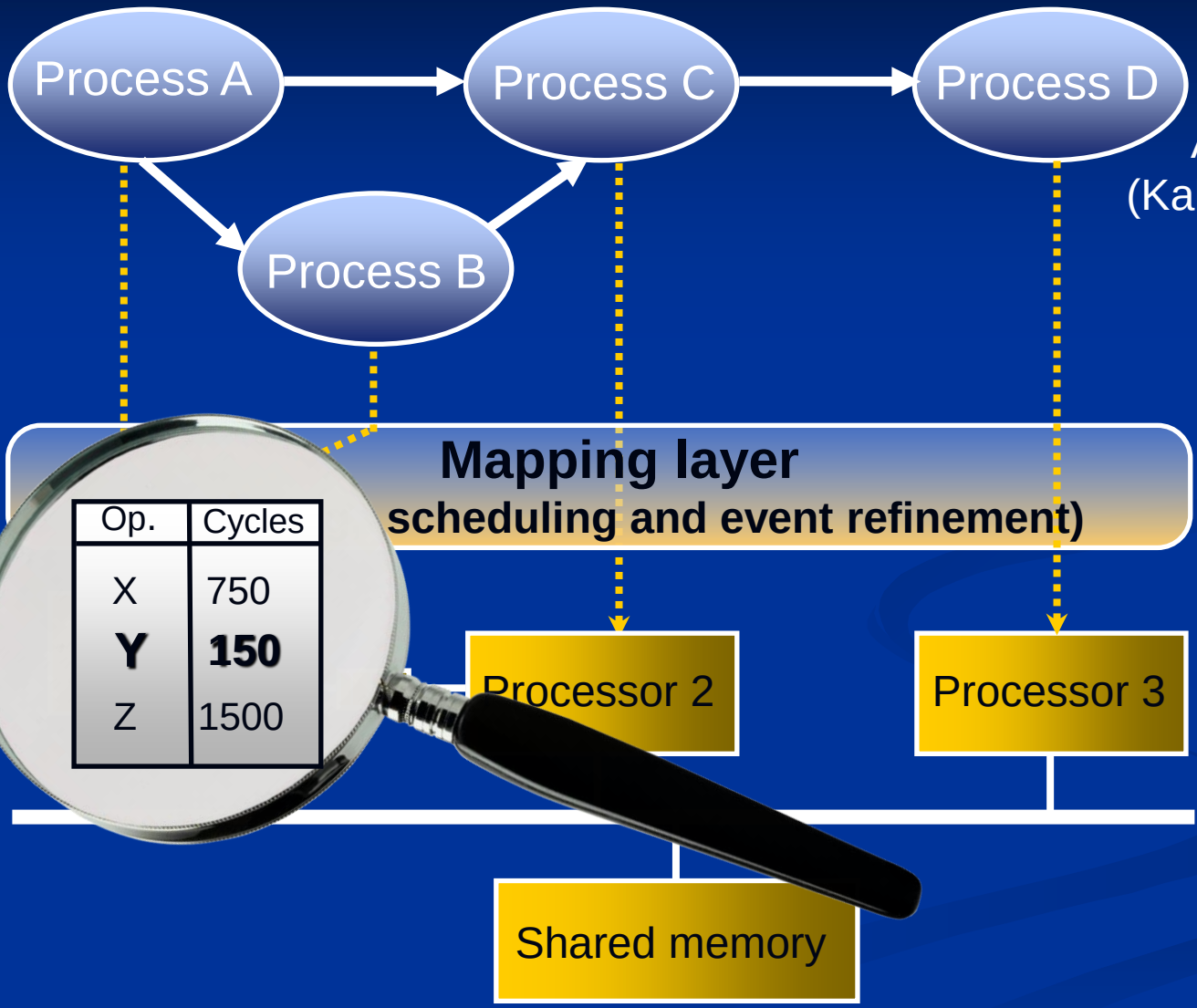
Sesame trace events



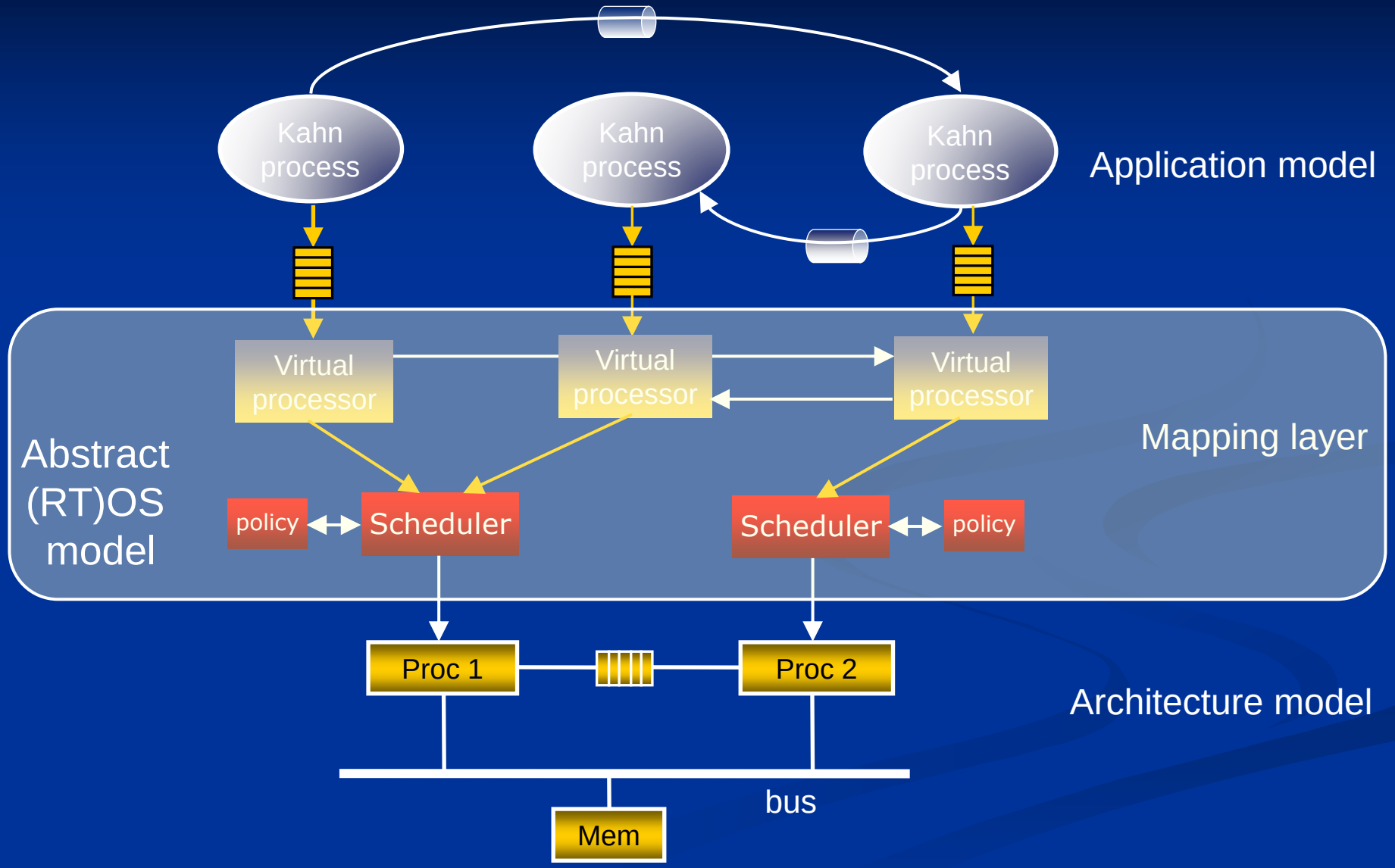
Application model (Kahn Process Network)

Cycle-approximate architecture model

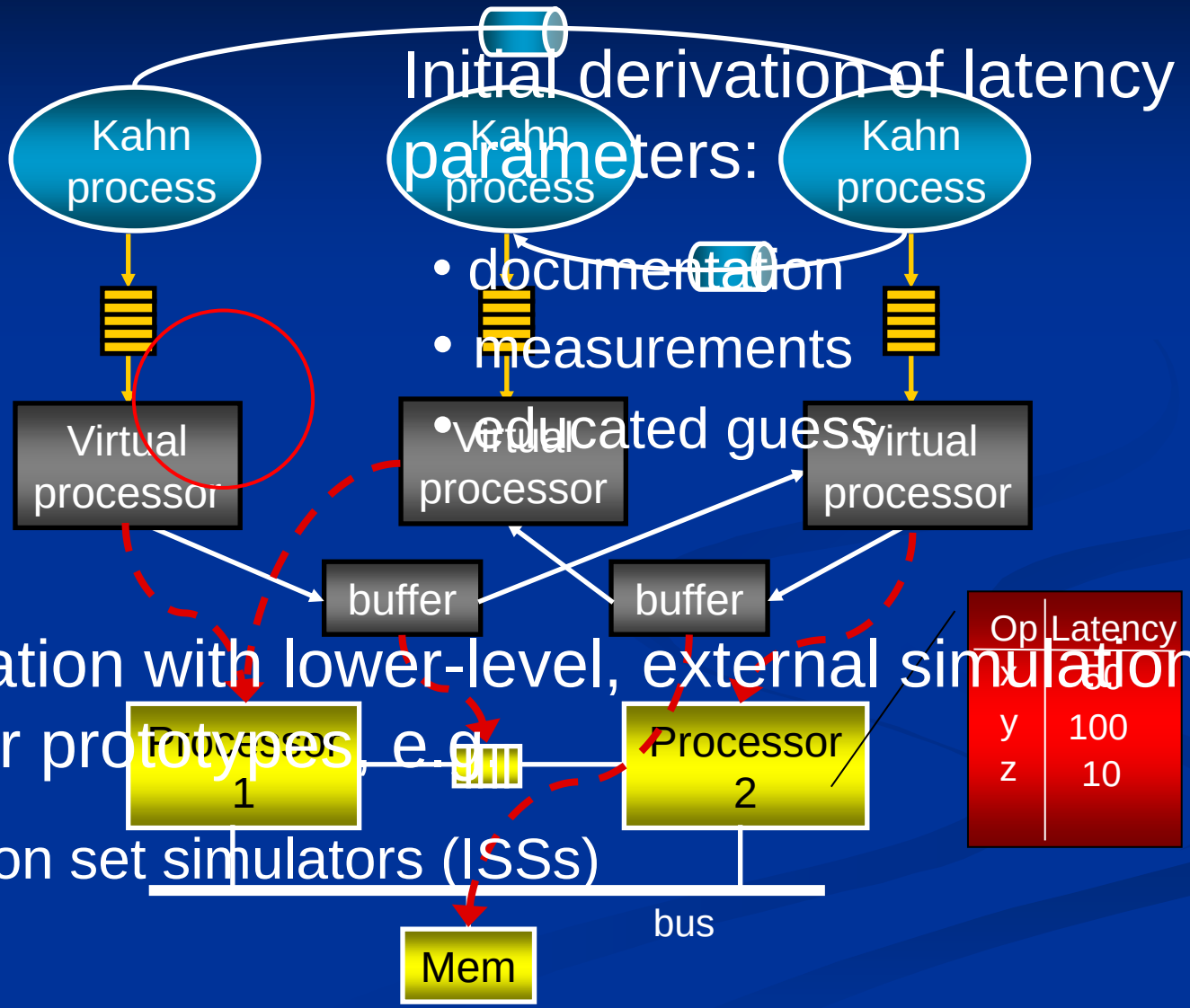
Sesame time tables



Sesame's mapping layer



Architecture model calibration



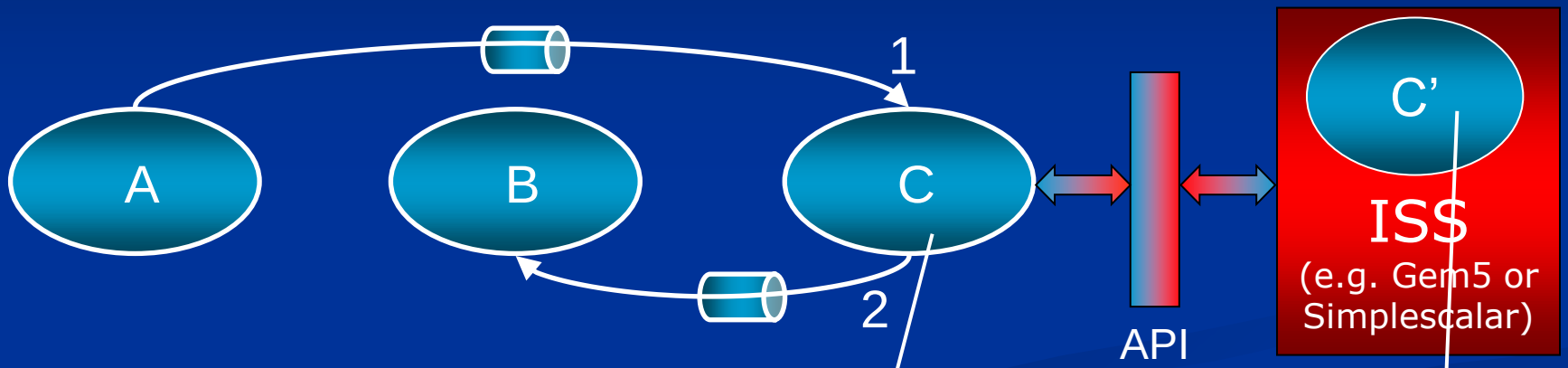
Initial derivation of latency parameters:

- documentation
- measurements
- educated guess

Or calibration with lower-level, external simulation models or prototypes, e.g.

- Instruction set simulators (ISSs)
- ESPAM

Calibration using an ISS

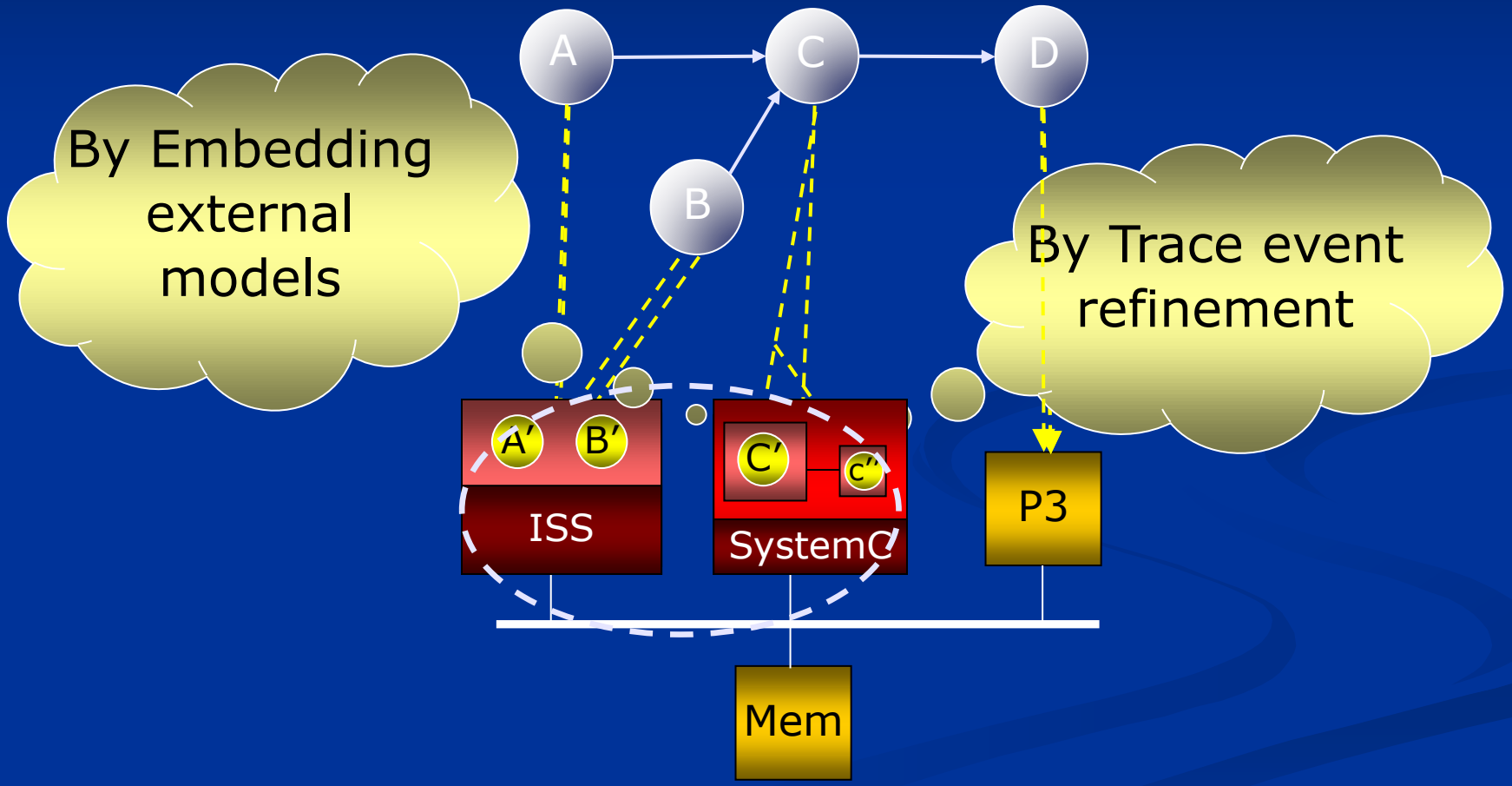


ISS measures cycle times of annotated code fragments

Mixed-level system simulation

- "Zoom in" on interesting system components in architecture model
  - Simulate these components at a lower level
- Retain high abstraction level for other components
  - Saves modelling effort
  - May save simulation overhead
- Integration of external simulation models
  - ISSs, SystemC models, etc.
- BUT…Mixed-level simulation can be complex!
  - multiple time domains and time grain sizes (synchronization)
  - functional and non-functional architecture model components

How to do mixed-level simulation

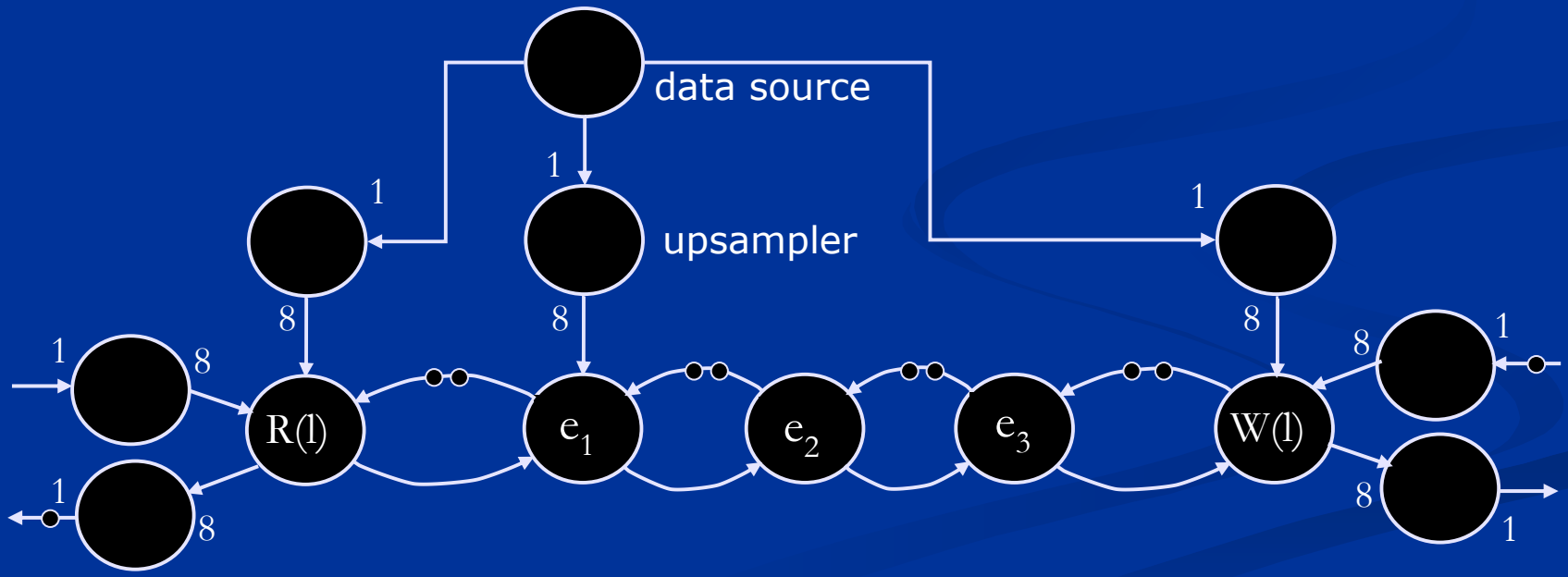


Trace Event Refinement: Mapping problem

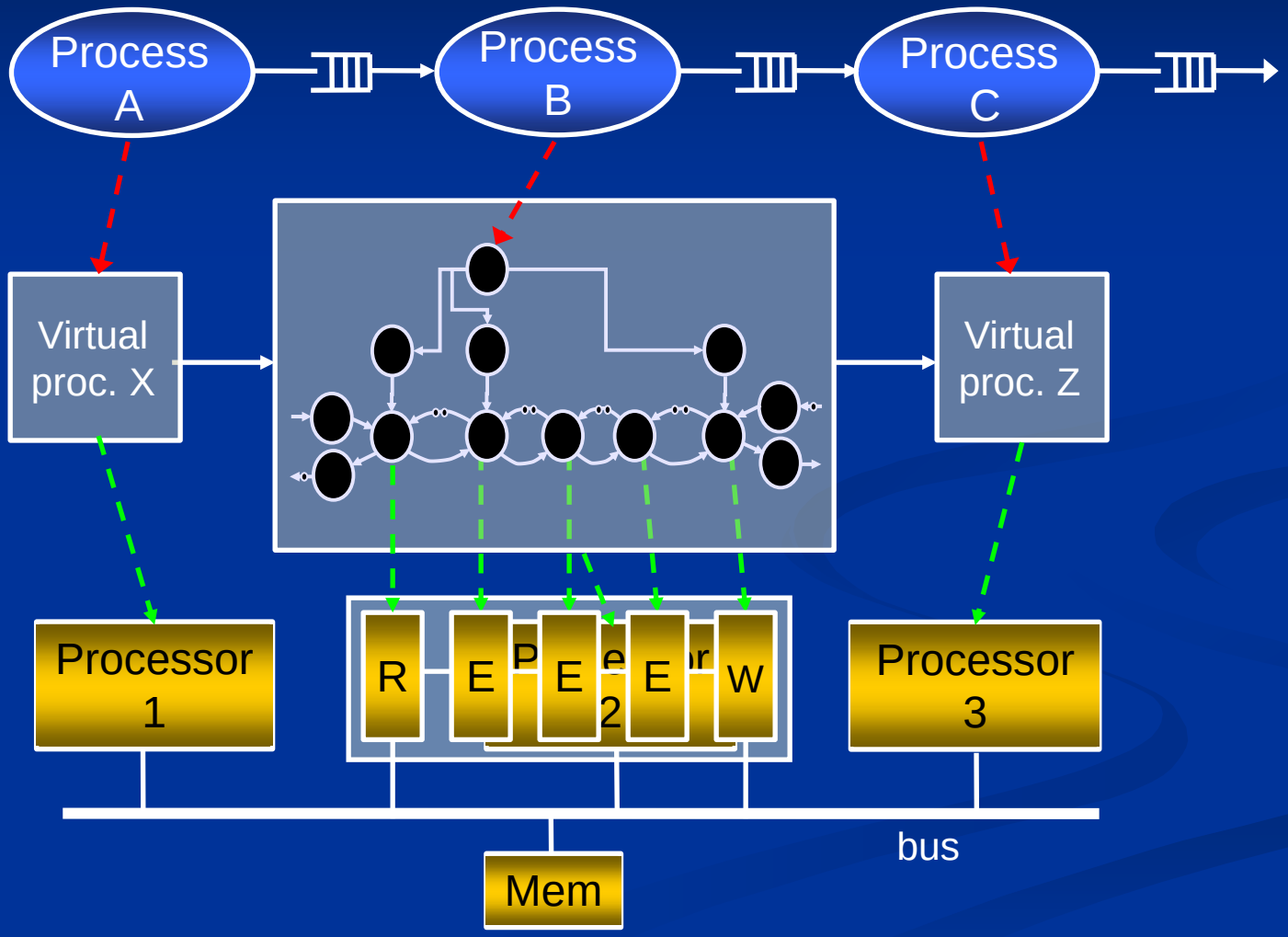
- Application events: Read, Write and Execute
- Typical mismatch between application events and architecture primitives, examples:
  - Architecture primitives operating on different data granularities
  - Architecture primitives more refined than application events
- Trace events from the application layer need to be refined
- How?
  - ~~Refine the application model~~
  - A transformation mechanism between the application and architecture models

Computational refinement

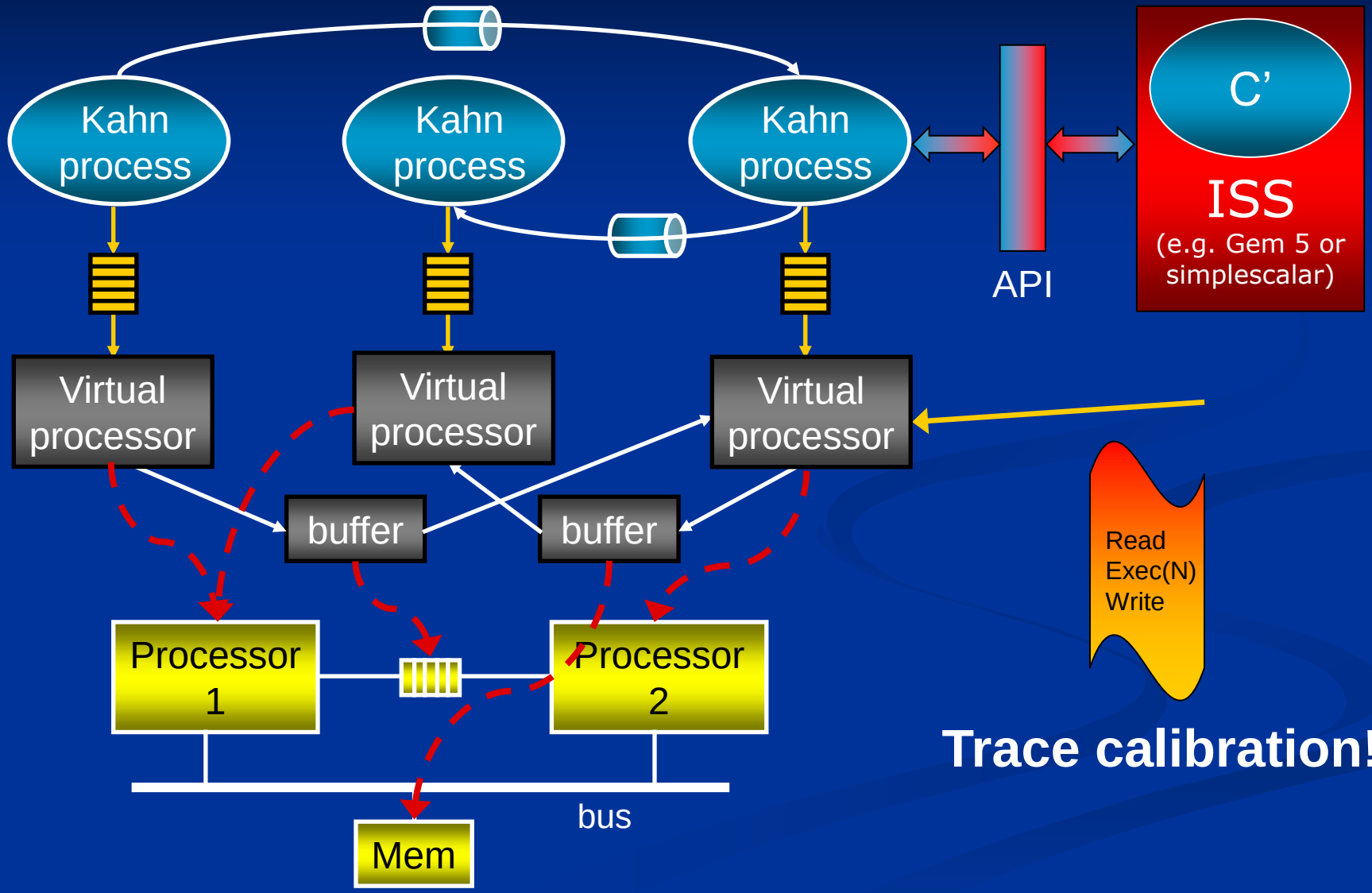
- Example trace transformation rules
  - R(block) $\Rightarrow$ R(line) $\rightarrow . . . \rightarrow$ R(line) (1)
  - W(block) $\Rightarrow$ W(line) $\rightarrow . . . \rightarrow$ W(line) (2)
  - E(block) $\Rightarrow$ E(line) $\rightarrow . . . \rightarrow$ E(line) (3)
  - E(line) $\Rightarrow$ $e_1 \rightarrow . . . \rightarrow e_n$ (4)



Computational refinement (2)



Mixed-level co-simulation by Integrating External Simulators

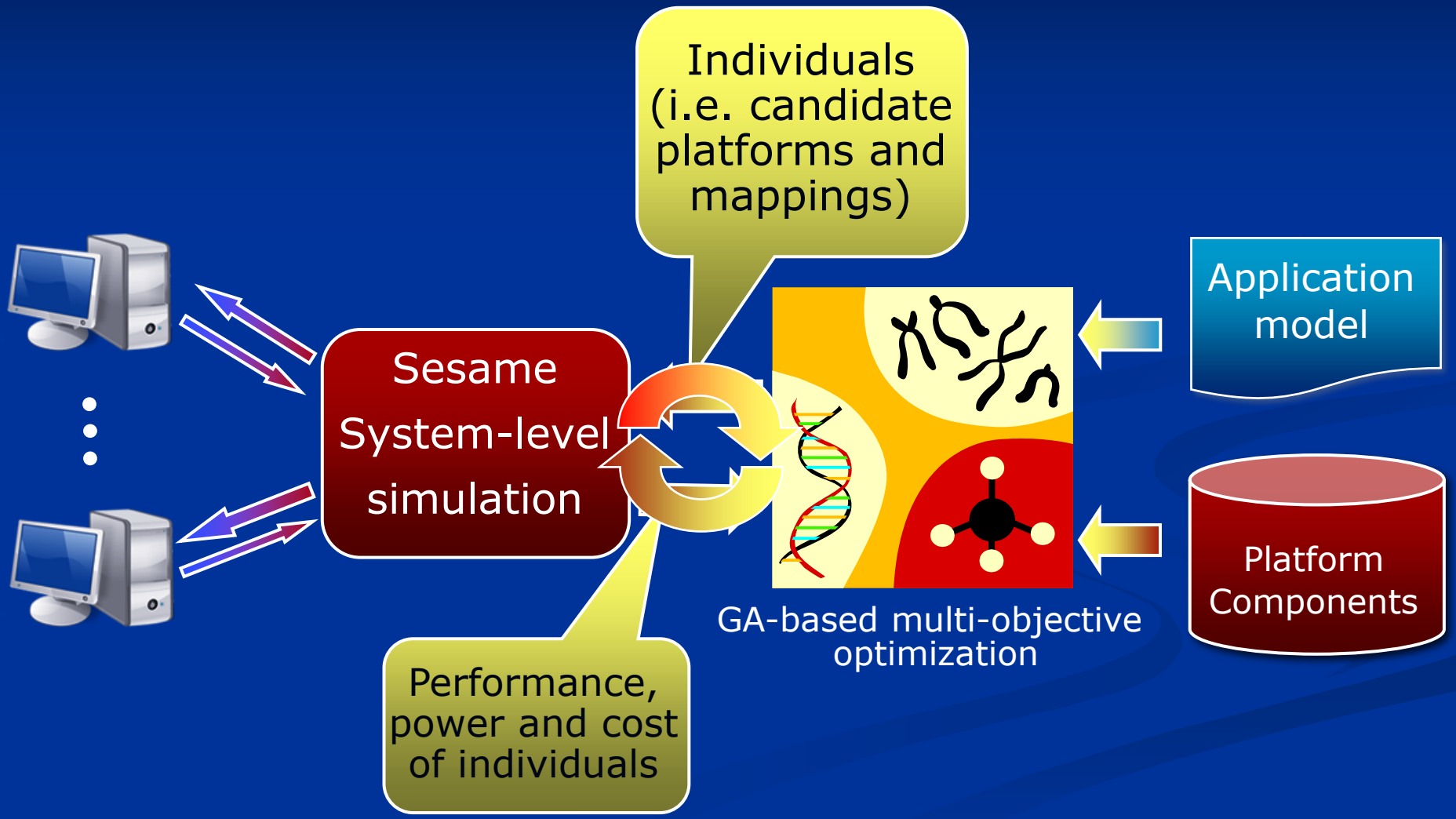


Trace calibration!

Towards real DSE with Sesame

- Sesame supplies basic methods & tools for evaluating application, architecture, and mapping combinations
  - Simulating the entire design space is not an option
- More is needed to explore large design spaces
  - How to avoid exhaustive search in the design space?
  - How to travers efficiently the design space?
- The answer is:
  - Use genetic/evolutionary algorithms to travers/search
  - Use Sesame simulations as fitness function
  - Find Pareto-optimal designs using multi-objective optimization

Design space exploration (DSE) using Genetic Algorithms



Sesame Summary

- Targets efficient evaluation of different
  - Application-to-architectures mappings
  - Hardware/Software partitionings
  - MP-SoC architectures
    - Different type and number of processing cores, interconnects (NoCs), scheduling policies, etc.
- Provides approximations/insight on
  - Cycle times, system utilization, bottlenecks/resource contention
- Low modeling effort and high simulation speed
  - Modeling in a matter of hours/days
  - Typically, a full system-level MP-SoC simulation takes less than 1 second on an average laptop