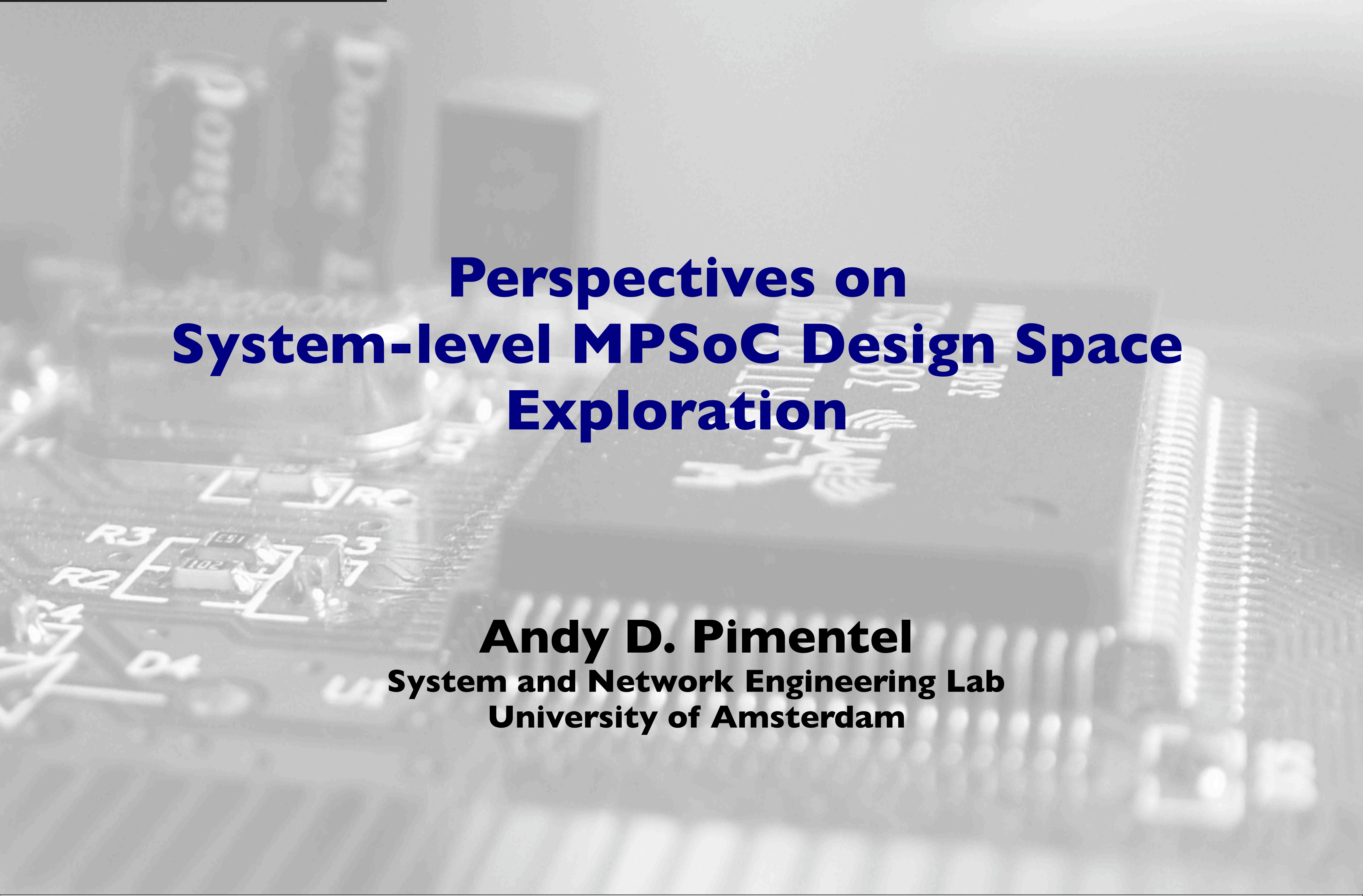


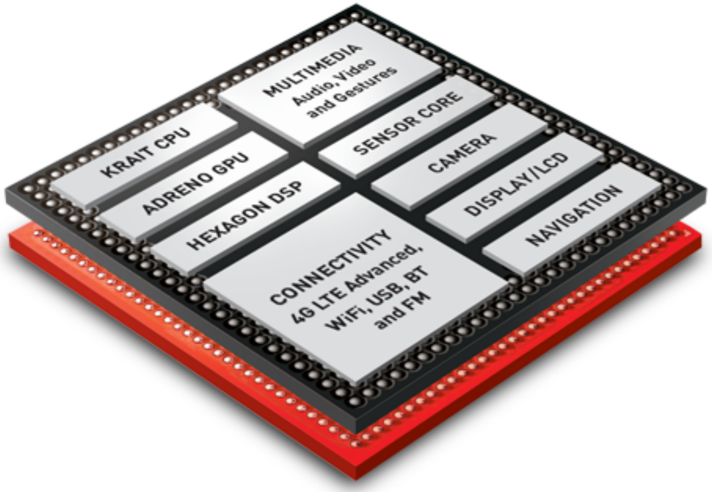
Perspectives on System-level MPSoC Design Space Exploration

Andy D. Pimentel

System and Network Engineering Lab

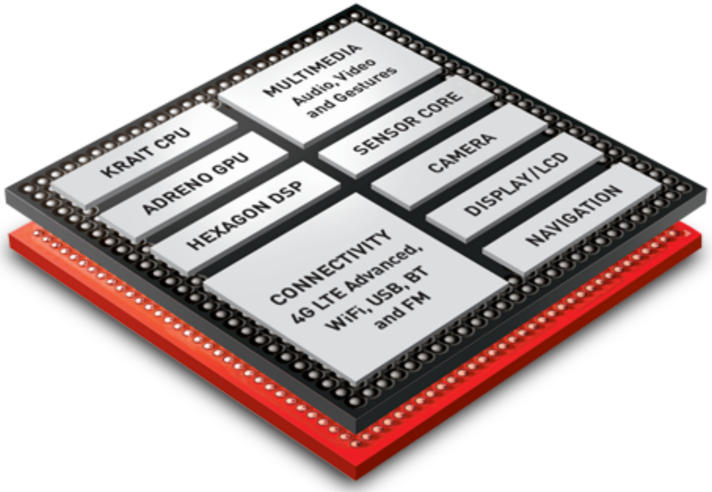
University of Amsterdam

Embedded Systems Design



- Design of embedded systems becomes increasingly complex
- Heterogeneous Multi-Processor System-on-Chip architectures
  - ✓ Different processor types, dedicated / reconfigurable hardware blocks, Network-on-Chip, etc.

Embedded Systems Design



- Design of embedded systems becomes increasingly complex
- Heterogeneous Multi-Processor System-on-Chip architectures
  - ✓ Different processor types, dedicated / reconfigurable hardware blocks, Network-on-Chip, etc.
- Many design requirements
  - ✓ High performance, low power, low cost, small form factor, high flexibility, high reliability, etc.
  - ✓ Typically conflicting requirements

Our Holy Grail…

[redacted]

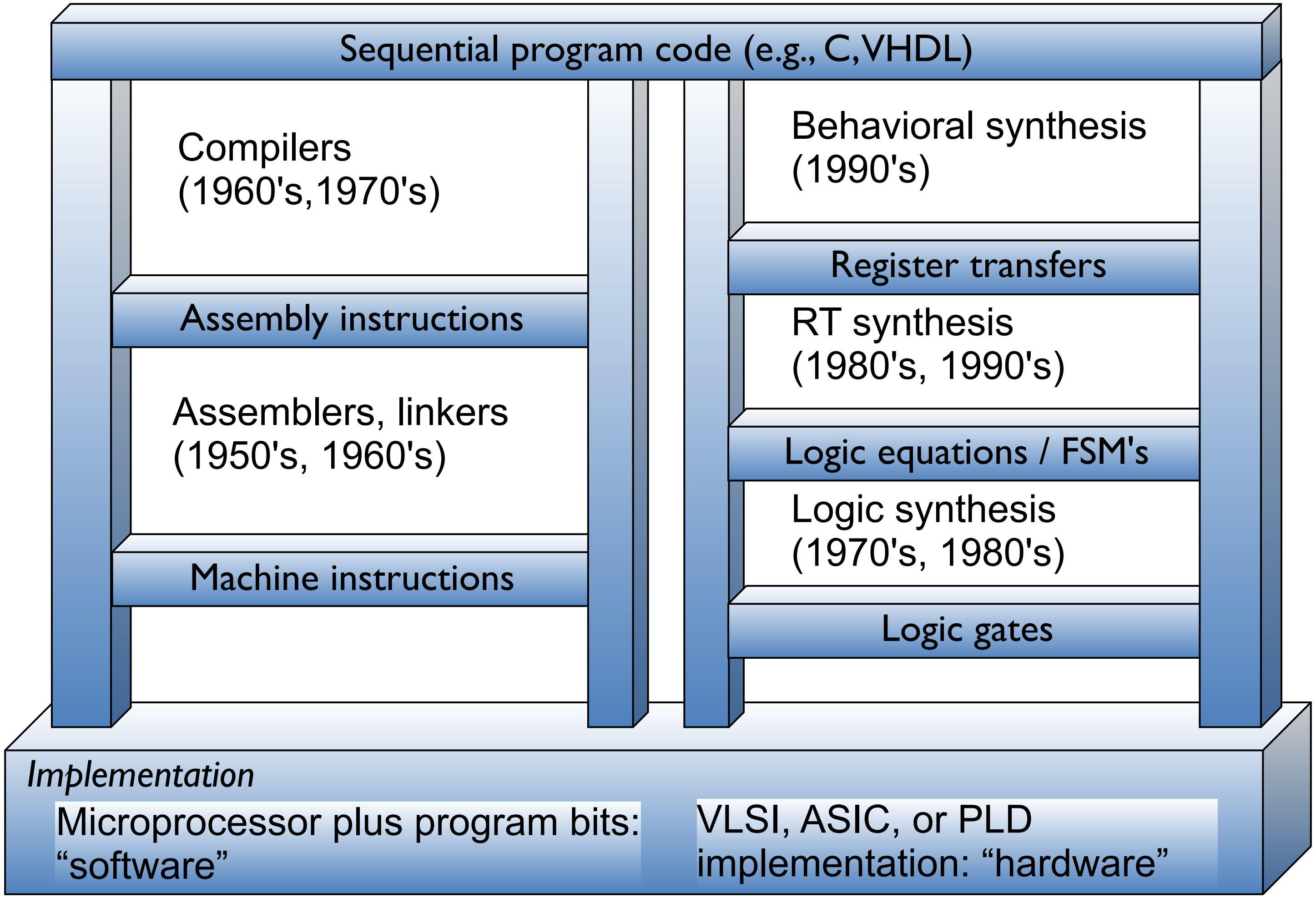
Our Holy Grail…

Our Holy Grail…

[redacted]

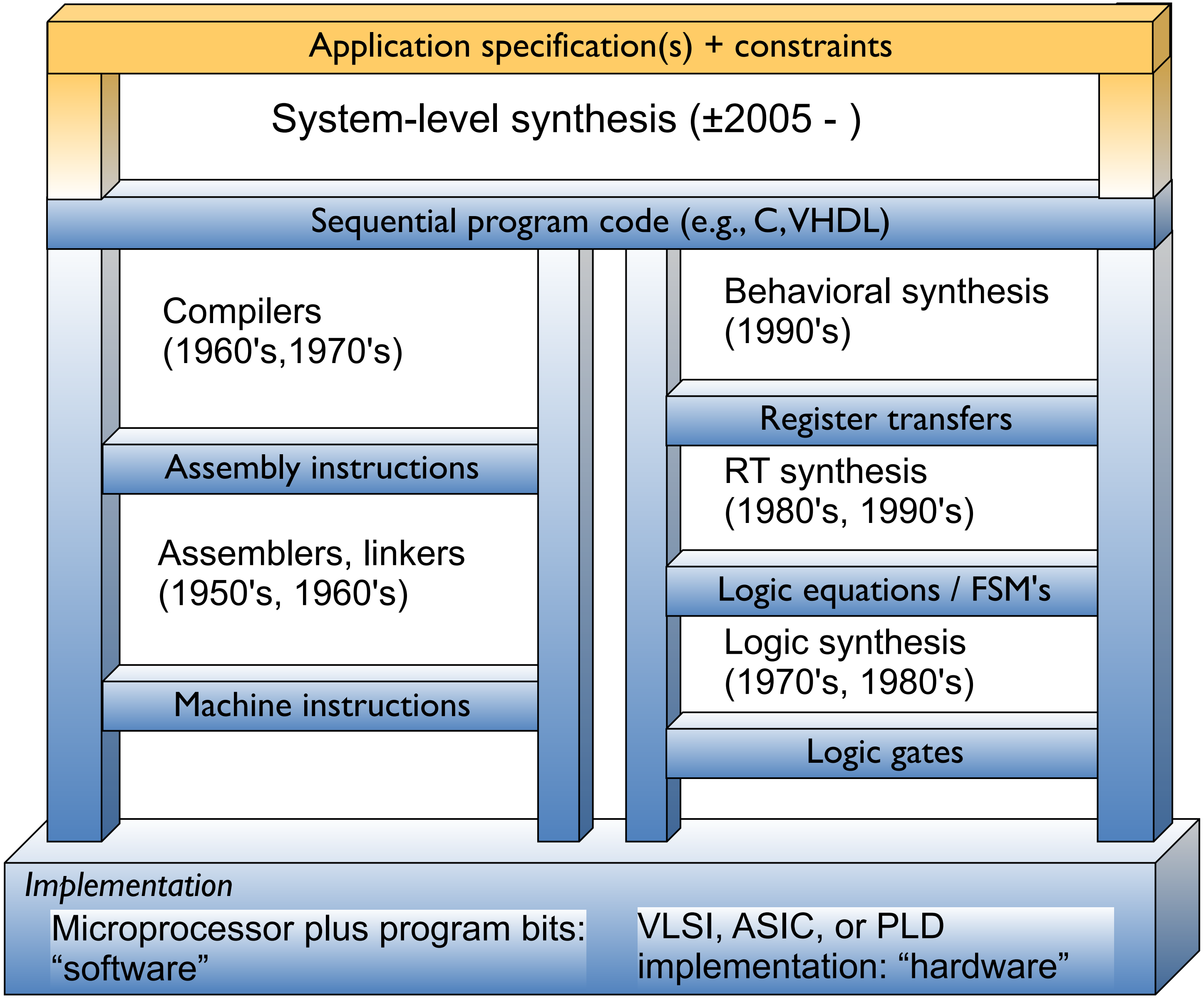
Climbing the abstraction ladder

Climbing the abstraction ladder


Sequential program code (e.g., C, VHDL)
Compilers (1960's,1970's)
Assembly instructions
Assemblers, linkers (1950's, 1960's)
Machine instructions
Behavioral synthesis (1990's)
Register transfers
RT synthesis (1980's, 1990's)
Logic equations / FSM's
Logic synthesis (1970's, 1980's)
Logic gates
Implementation
Microprocessor plus program bits: "software"
VLSI, ASIC, or PLD implementation: "hardware"

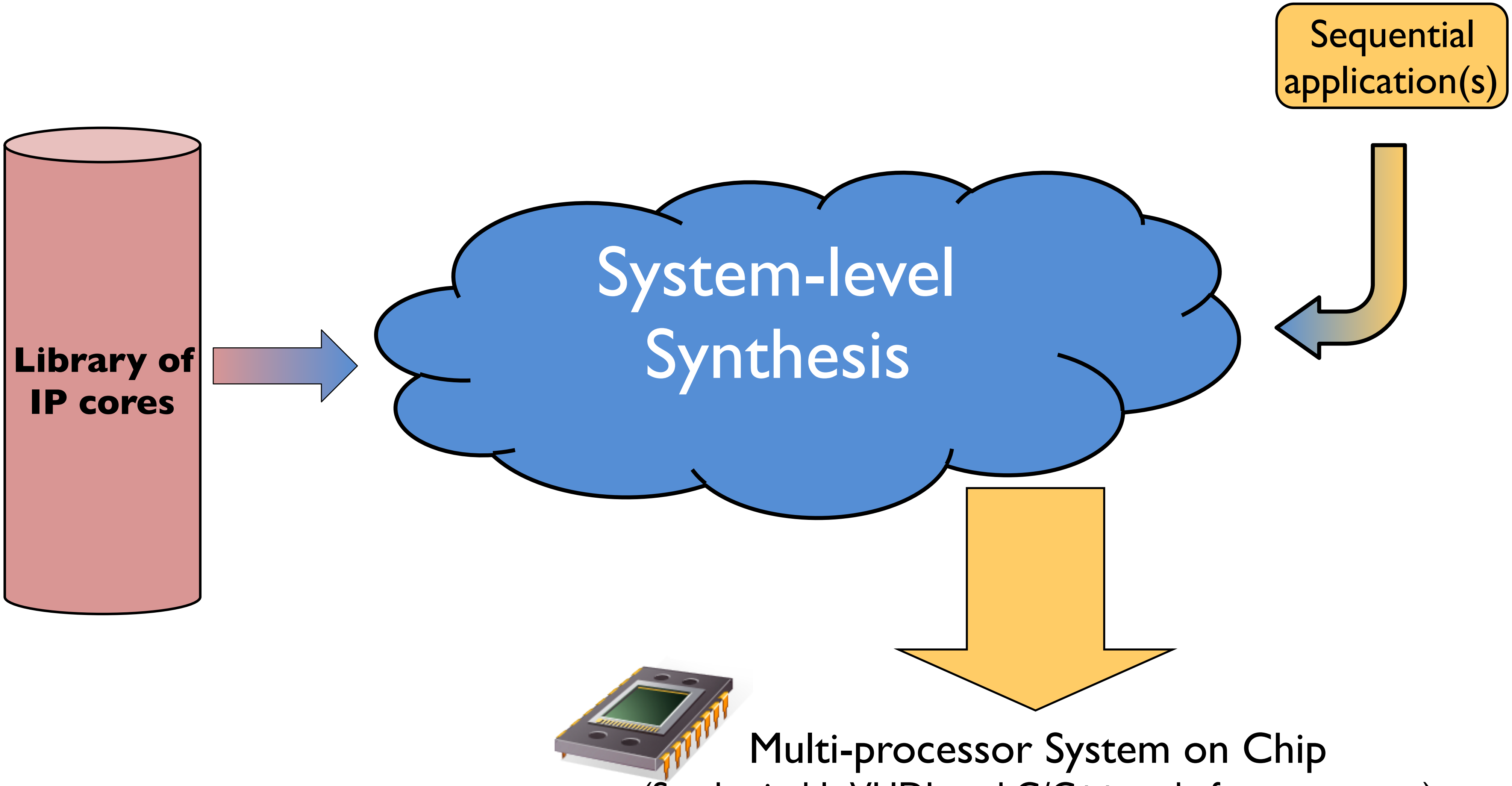
Source: Vahid/Givargis

Climbing the abstraction ladder



[IEEE TCAD'09]

Towards System-level Synthesis



Library of IP cores

Sequential application(s)

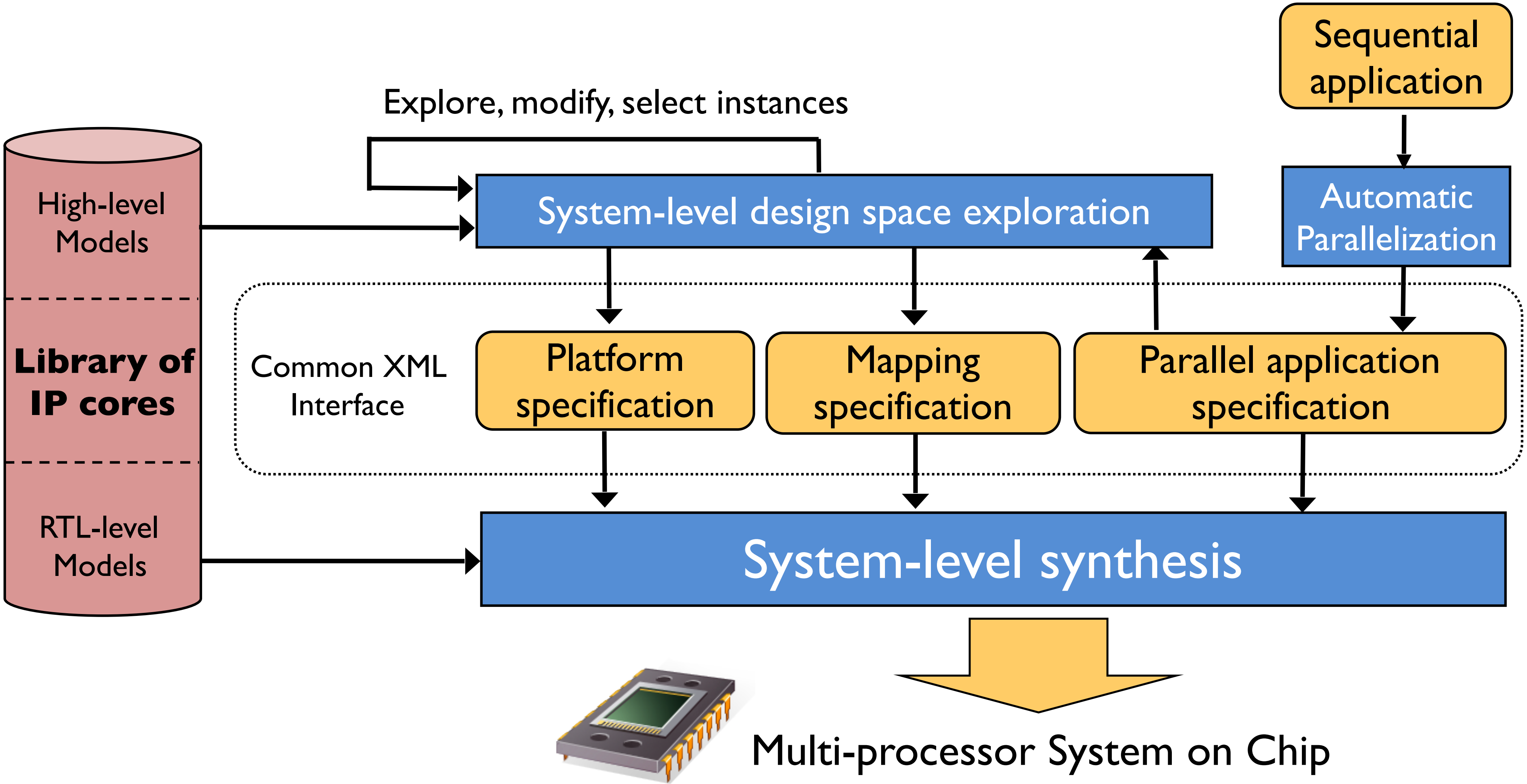
System-level Synthesis

Multi-processor System on Chip

(Synthesizable VHDL and C/C++ code for processors)

The context: Daedalus

A system-level synthesis framework



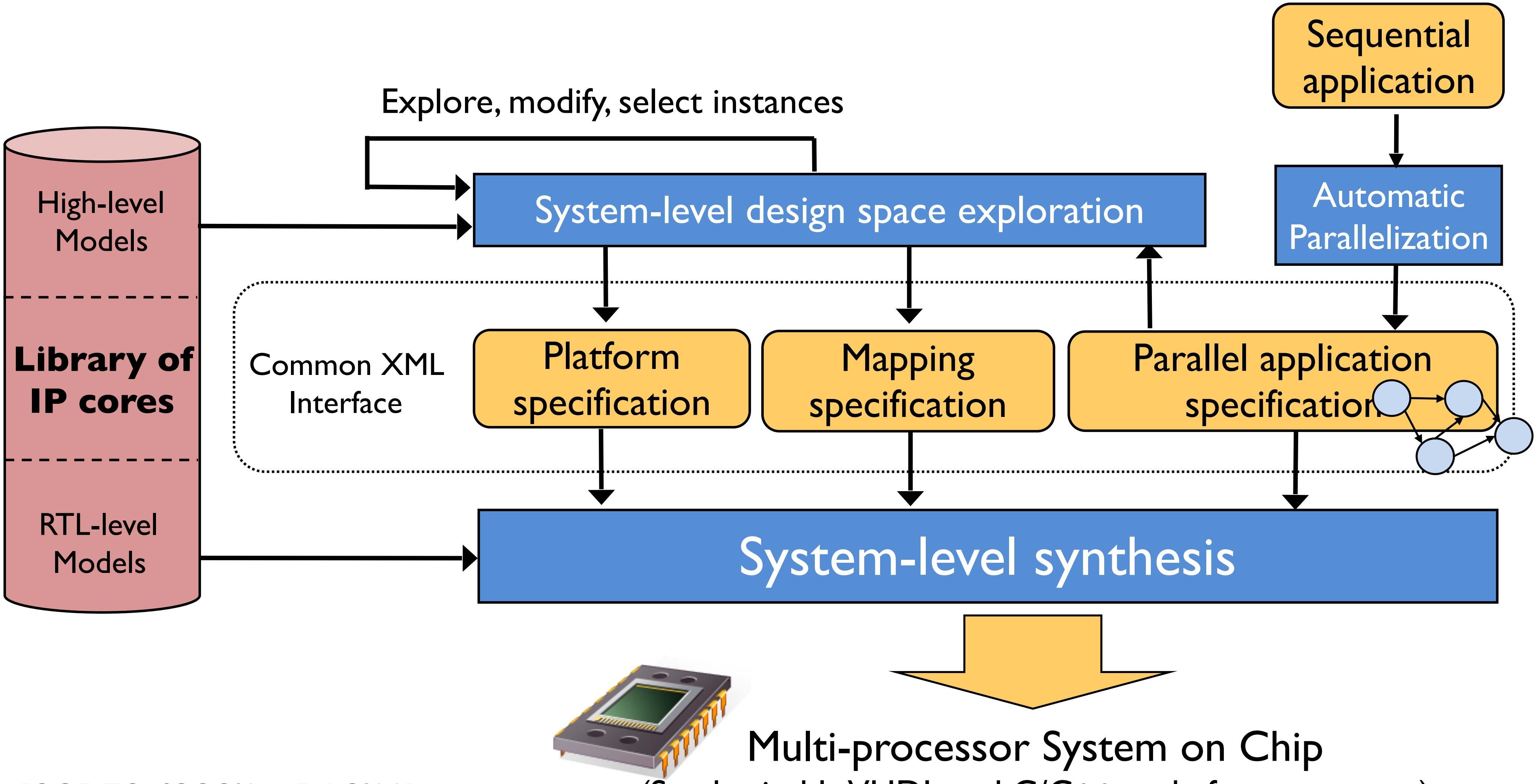
Multi-processor System on Chip

(Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS'07, DAC'08]

The context: Daedalus

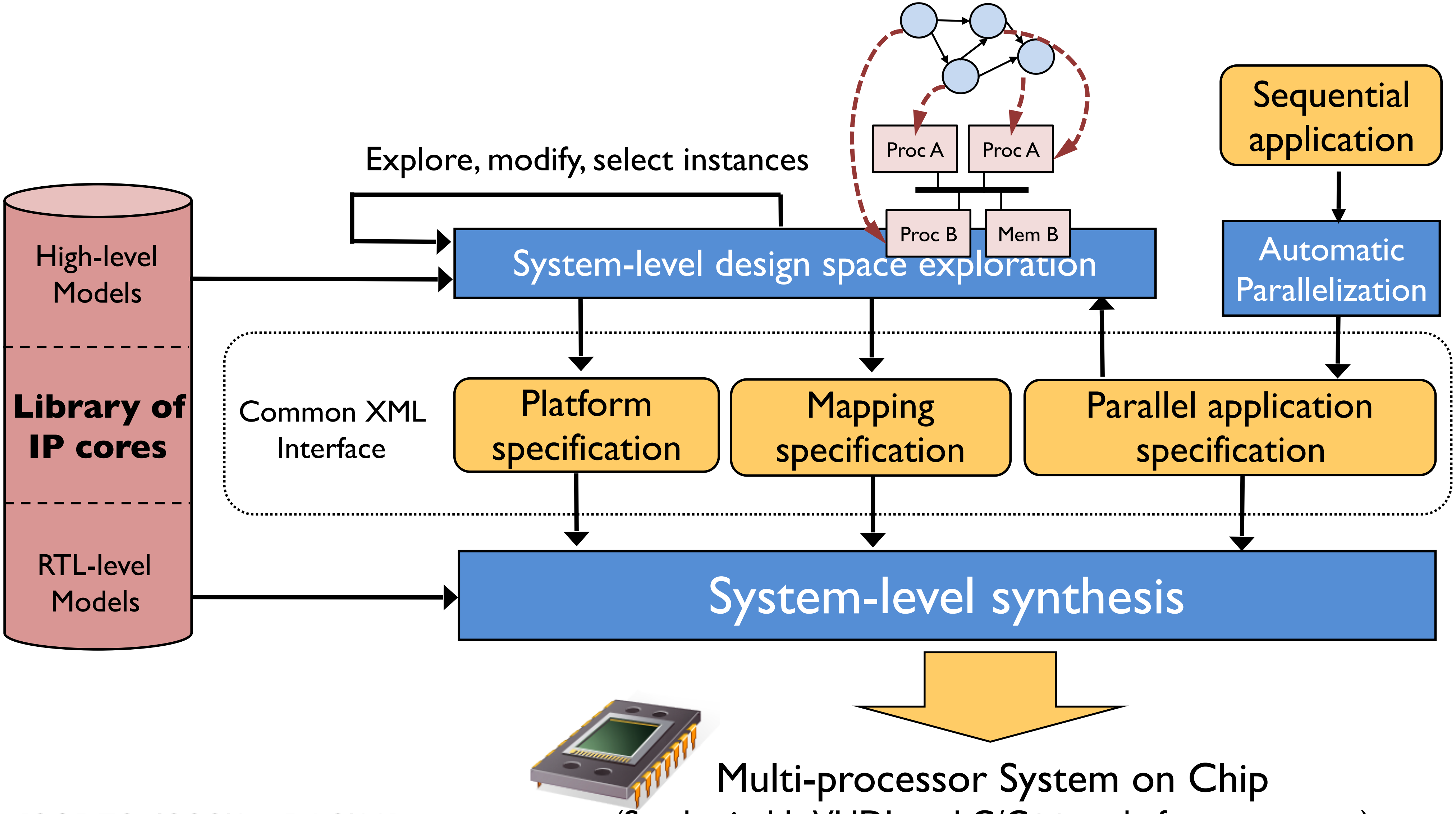
A system-level synthesis framework



Multi-processor System on Chip
(Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS'07, DAC'08]

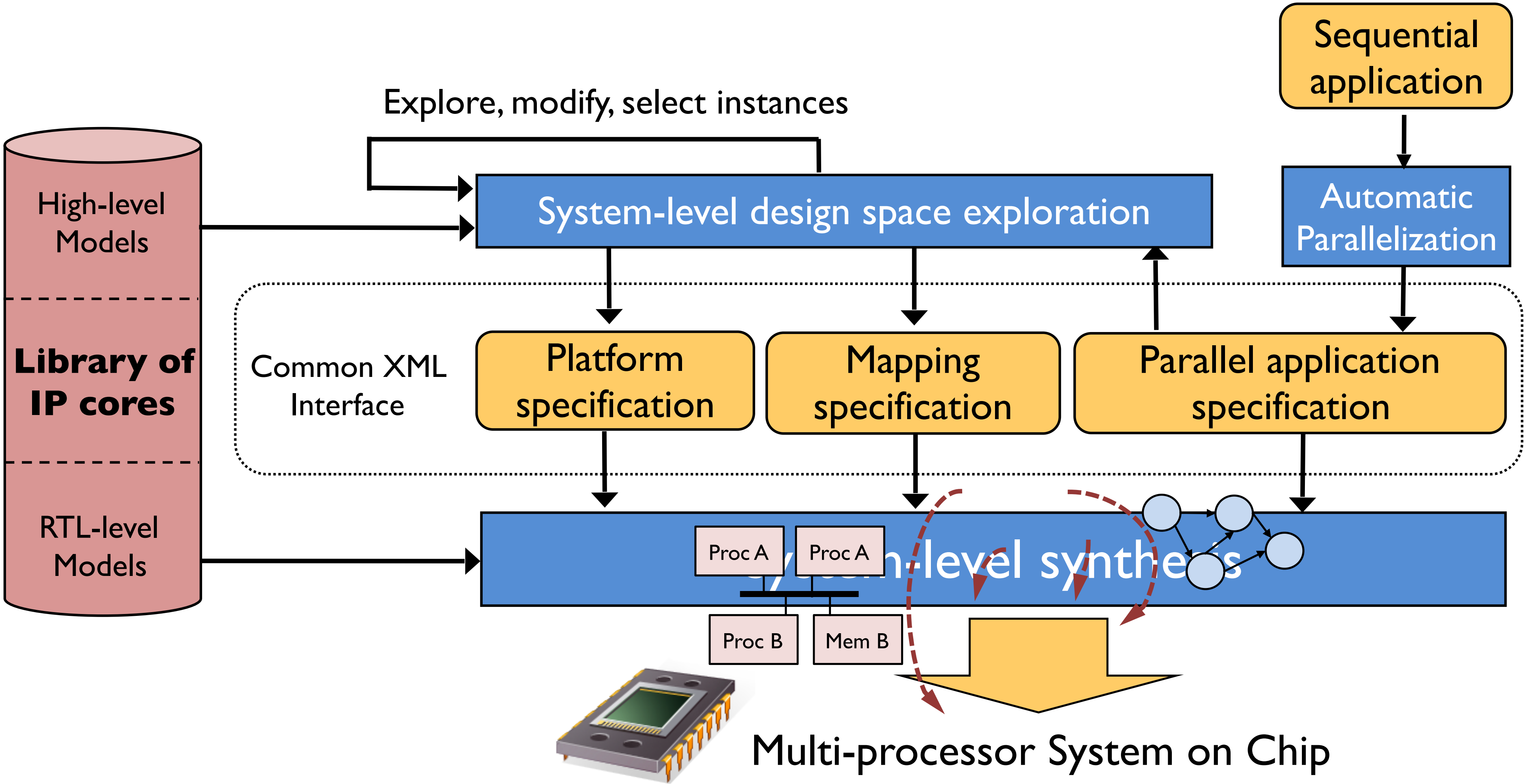
The context: Daedalus
A system-level synthesis framework



Multi-processor System on Chip
(Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS'07, DAC'08]

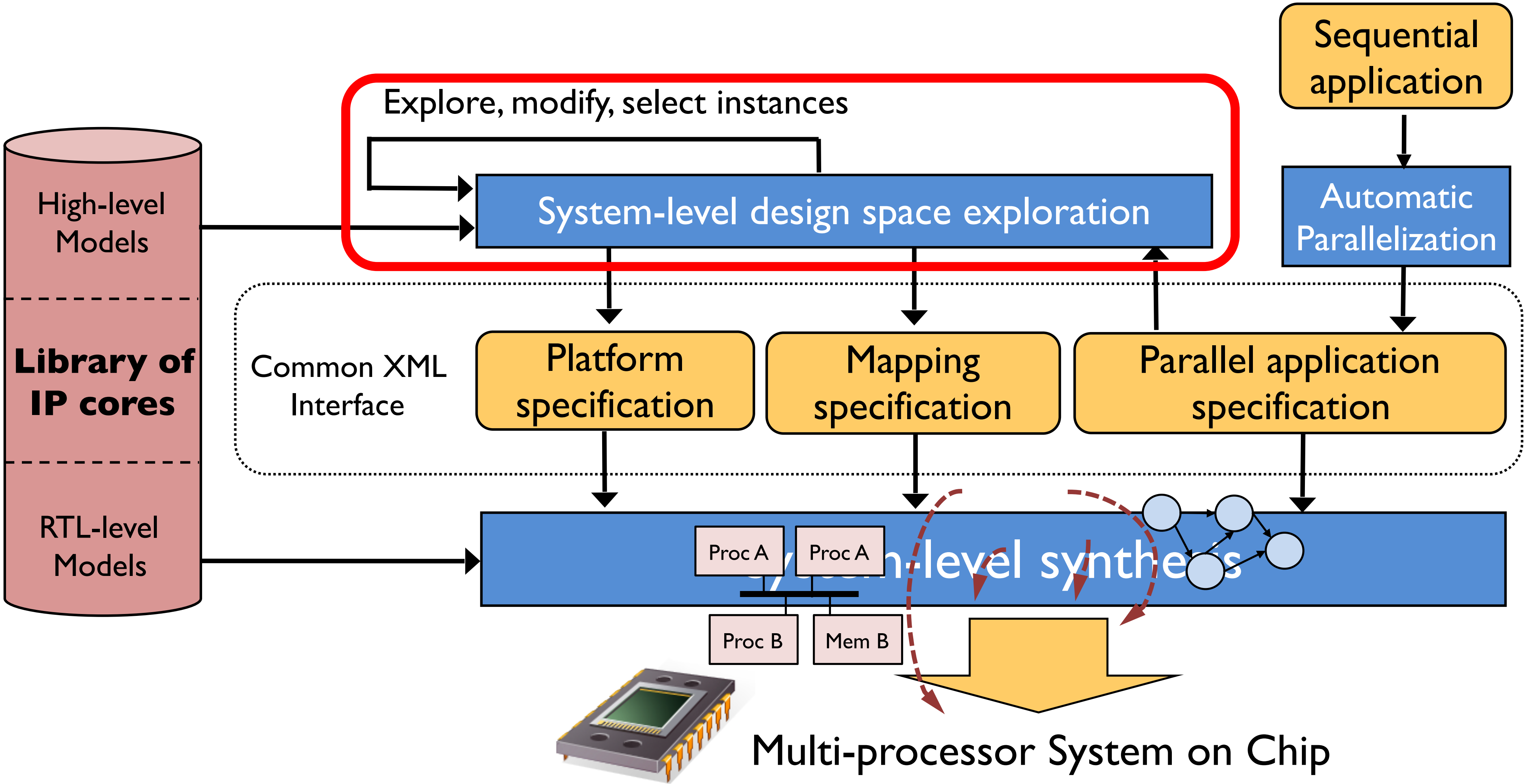
The context: Daedalus
A system-level synthesis framework



Multi-processor System on Chip
(Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS'07, DAC'08]

The context: Daedalus
A system-level synthesis framework



(Synthesizable VHDL and C/C++ code for processors)

[CODES+ISSS'07, DAC'08]

System-level Design
Space Exploration (DSE)

- We need to automatically
  - ✓ find the best decomposition of the (parallel) application(s)
  - ✓ decide what application task to perform in SW or accelerate using HW
  - ✓ choose the number and types of required processing elements in the (heterogeneous) system
  - ✓ decide on how to interconnect the processors
  - ✓ decide on how to map application tasks onto the selected processors
  - ✓ and so on…

System-level Design Space Exploration (DSE)

- We need to automatically
  - ✓ find the best decomposition of the (parallel) application(s)
  - ✓ decide what application task to perform in SW or accelerate using HW
  - ✓ choose the number and types of required processing elements in the (heterogeneous) system
  - ✓ decide on how to interconnect the processors
  - ✓ decide on how to map application tasks onto the selected processors
  - ✓ and so on…
- while simultaneously optimizing the system for cost, performance, energy consumption, reliability, etc.

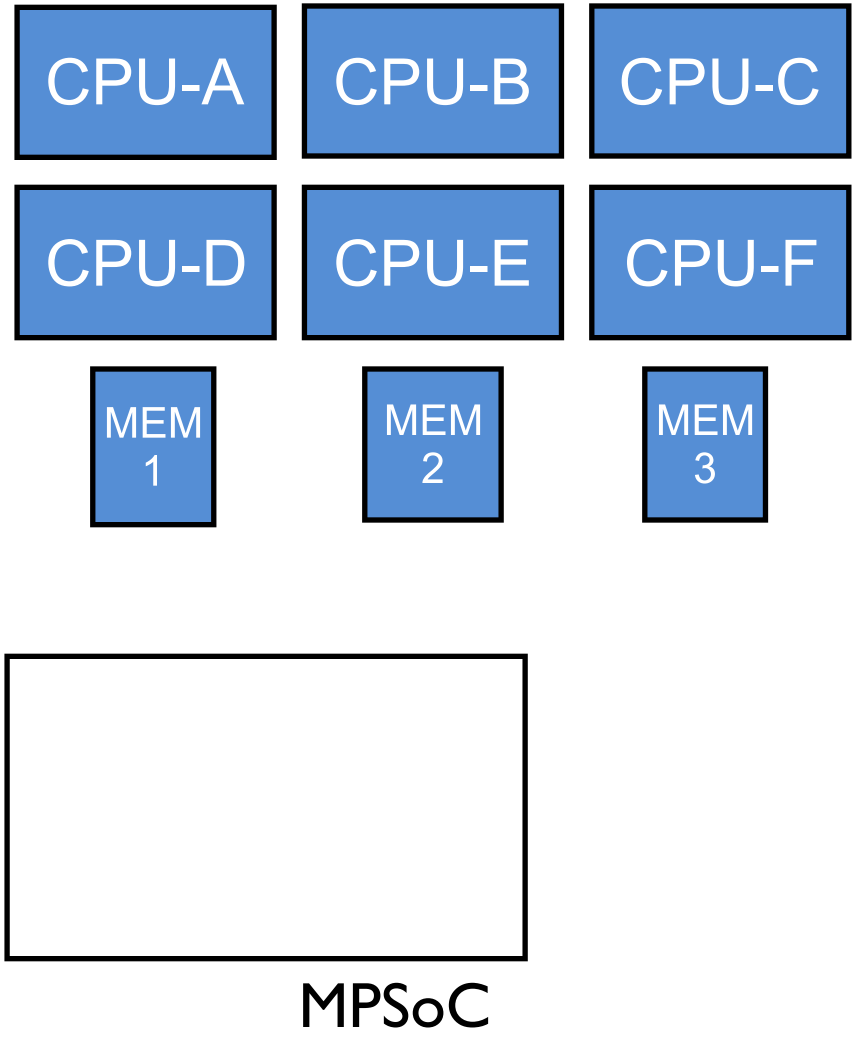
System-level Design
Space Exploration (DSE)

- We need to automatically
  - ✓ find the best dec ation(s)
  - ✓ dec
    ac
  - ✓ ch
  - ✓ d
  - ✓
- while si e system for cost, performance, energy on, reliability, etc.

Major challenge: develop DSE techniques that efficiently and effectively handle the vast design space, with sufficient accuracy

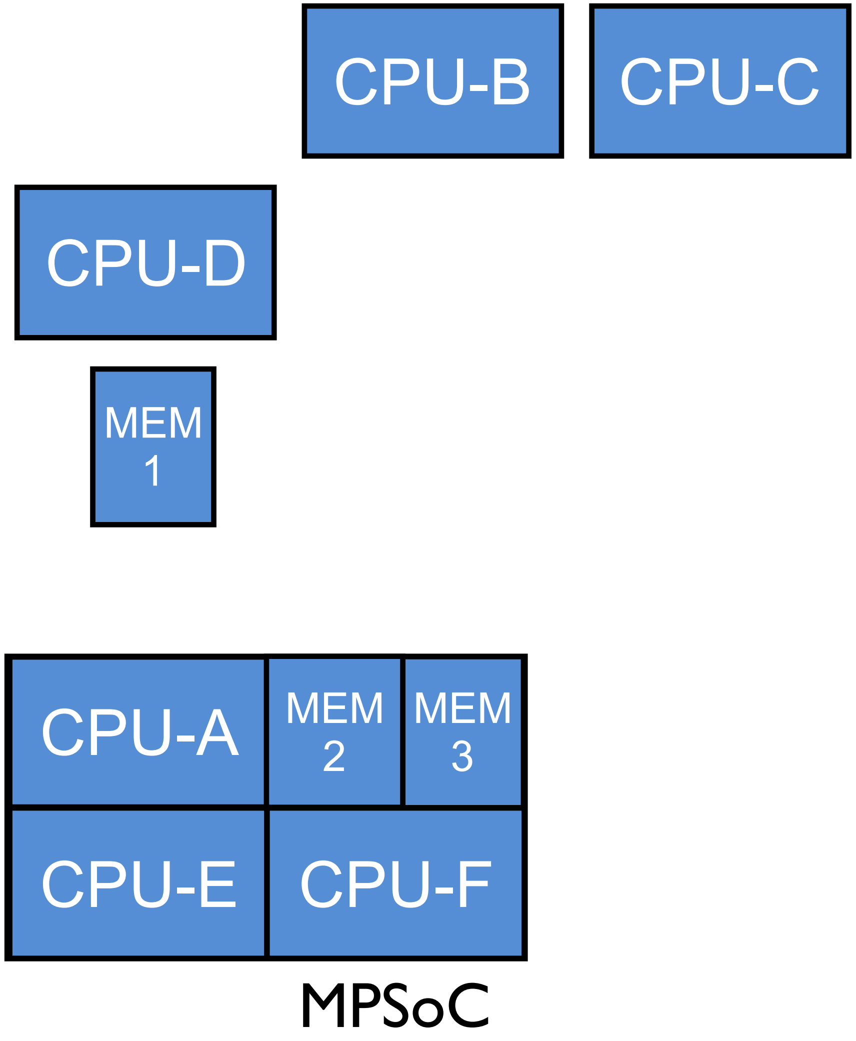
System-level Mapping DSE

- Exploring different
  - Resource allocations
    - ✓ Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)



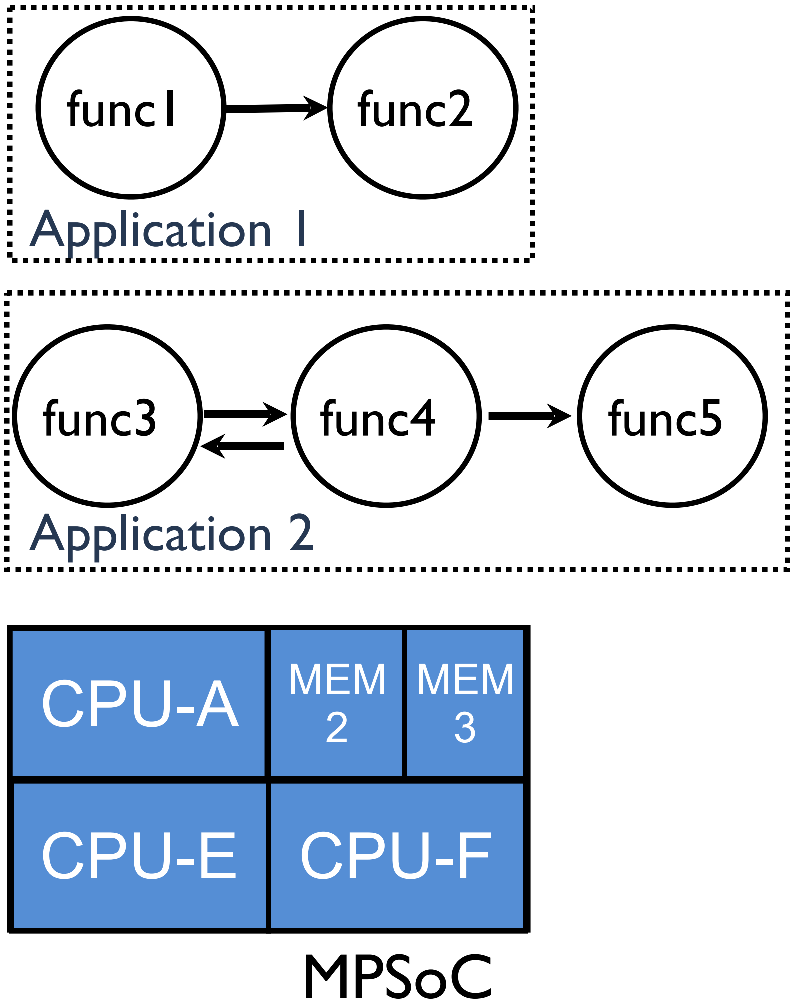
System-level Mapping DSE

- Exploring different
  - Resource allocations
    - ✓ Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)



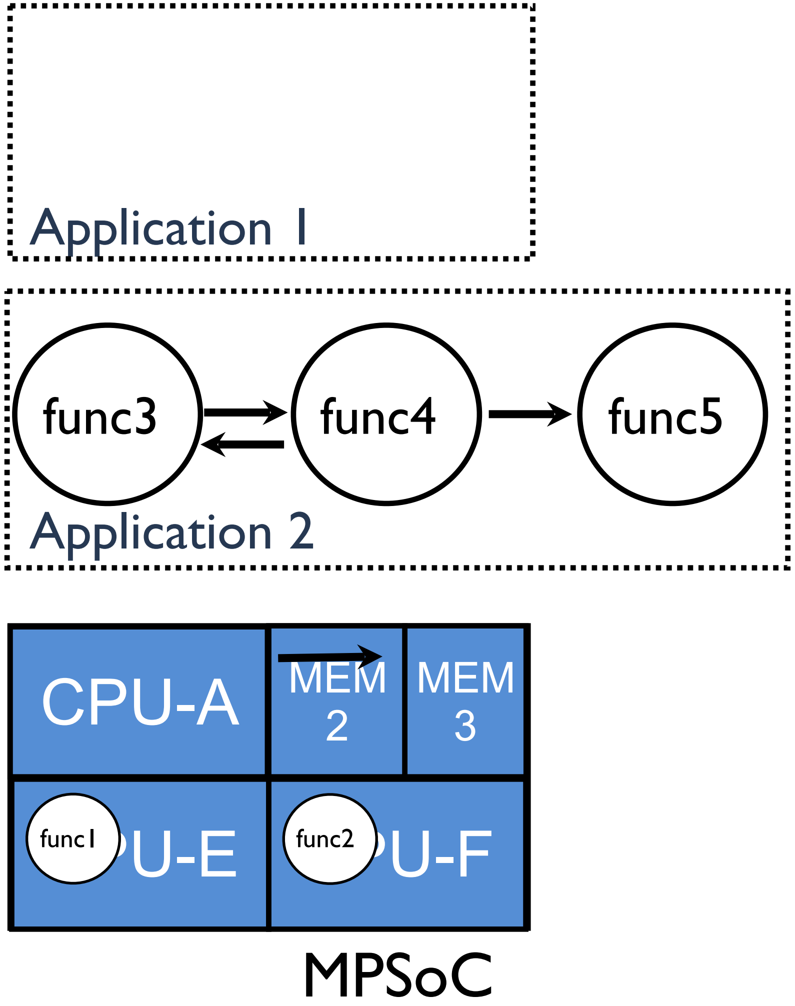
System-level Mapping DSE

- Exploring different
  - Resource allocations
    - ✓ Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)



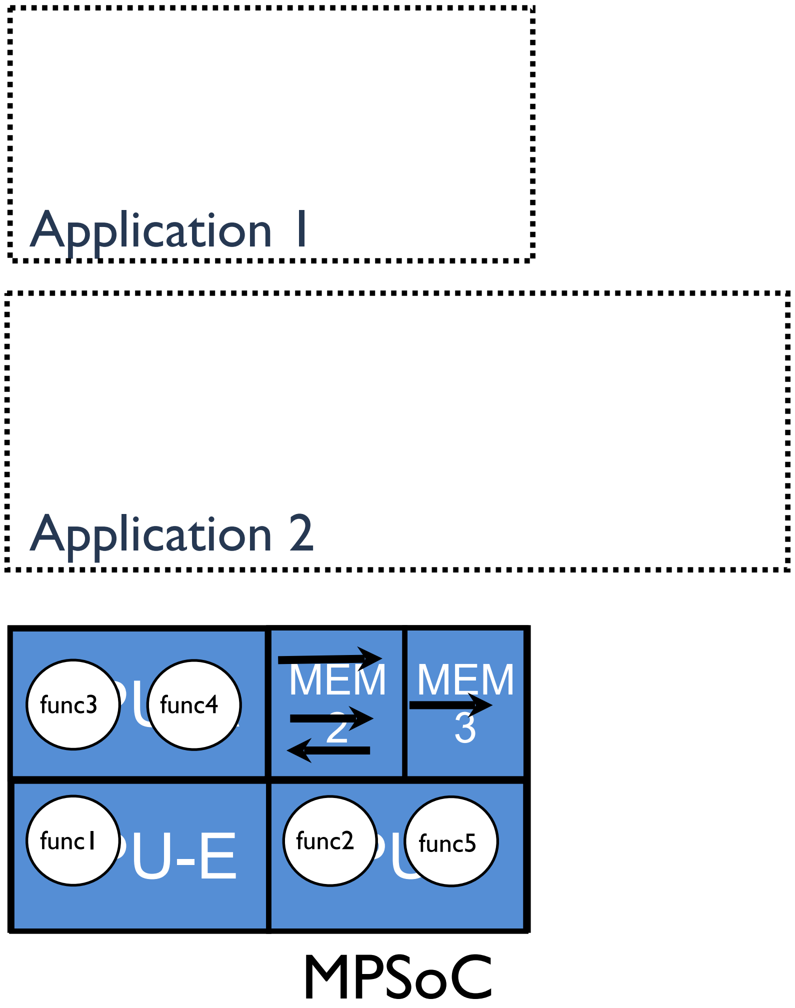
System-level Mapping DSE

- Exploring different
  - Resource allocations
    - ✓ Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)

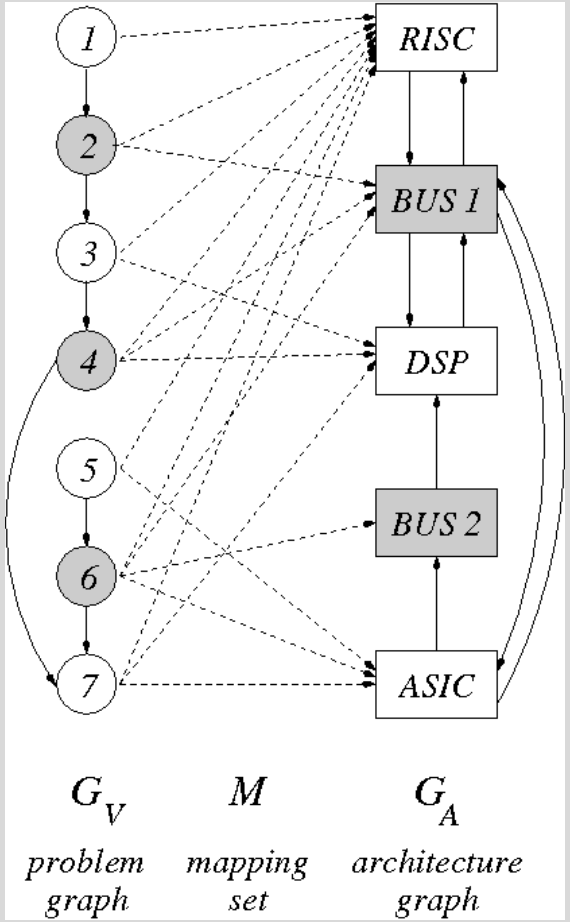


System-level Mapping DSE

- Exploring different
  - Resource allocations
    - ✓ Number and type of processors, memories, interconnect(s), etc.
  - Application to Resource bindings (spatial binding)
  - Task scheduling (temporal binding)



System-level DSE: two elements

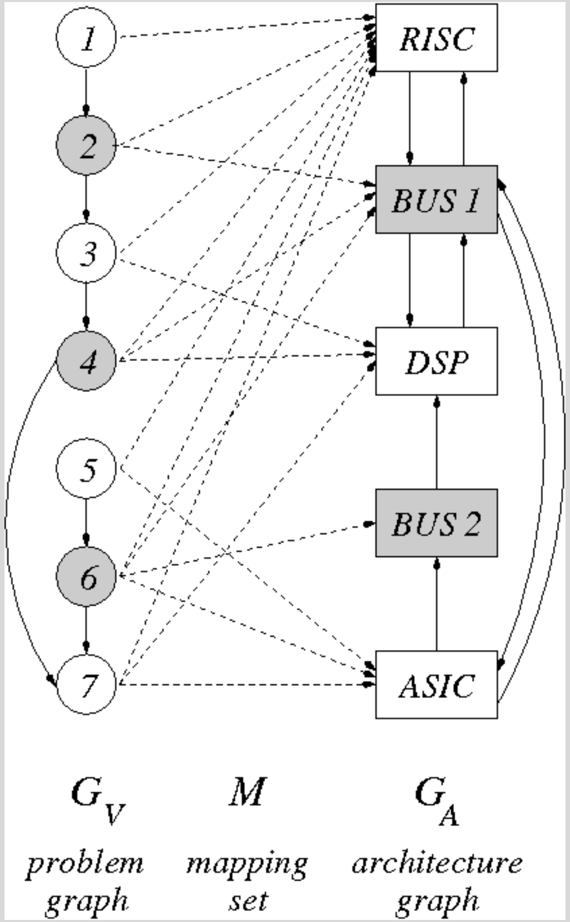


Source:
Teich et al.

Evaluating a design point

Searching the design space

System-level DSE: two elements

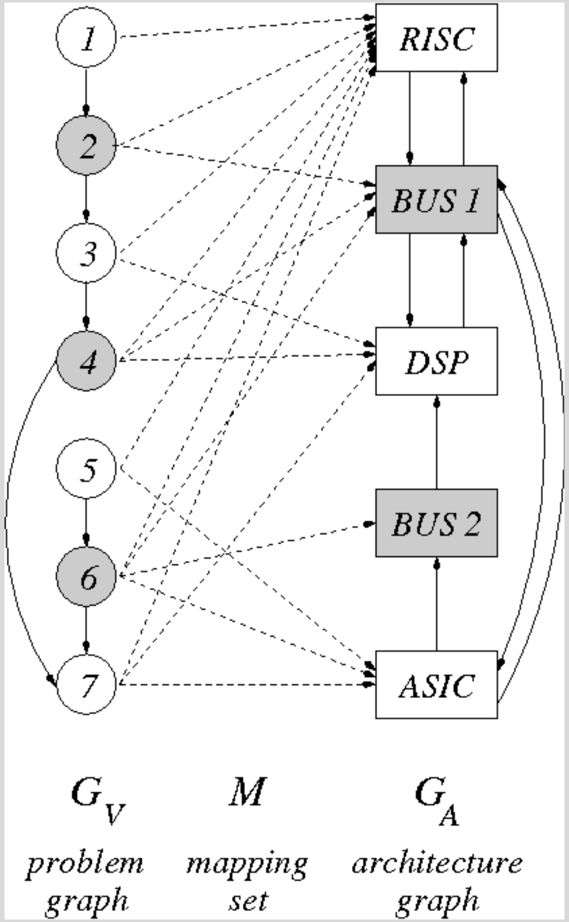


Source: Teich et al.

Evaluating a design point

Searching the design space

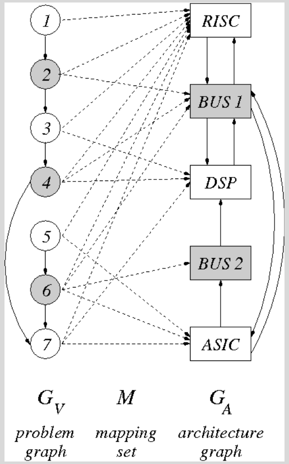
System-level DSE: two elements



Source: Teich et al.

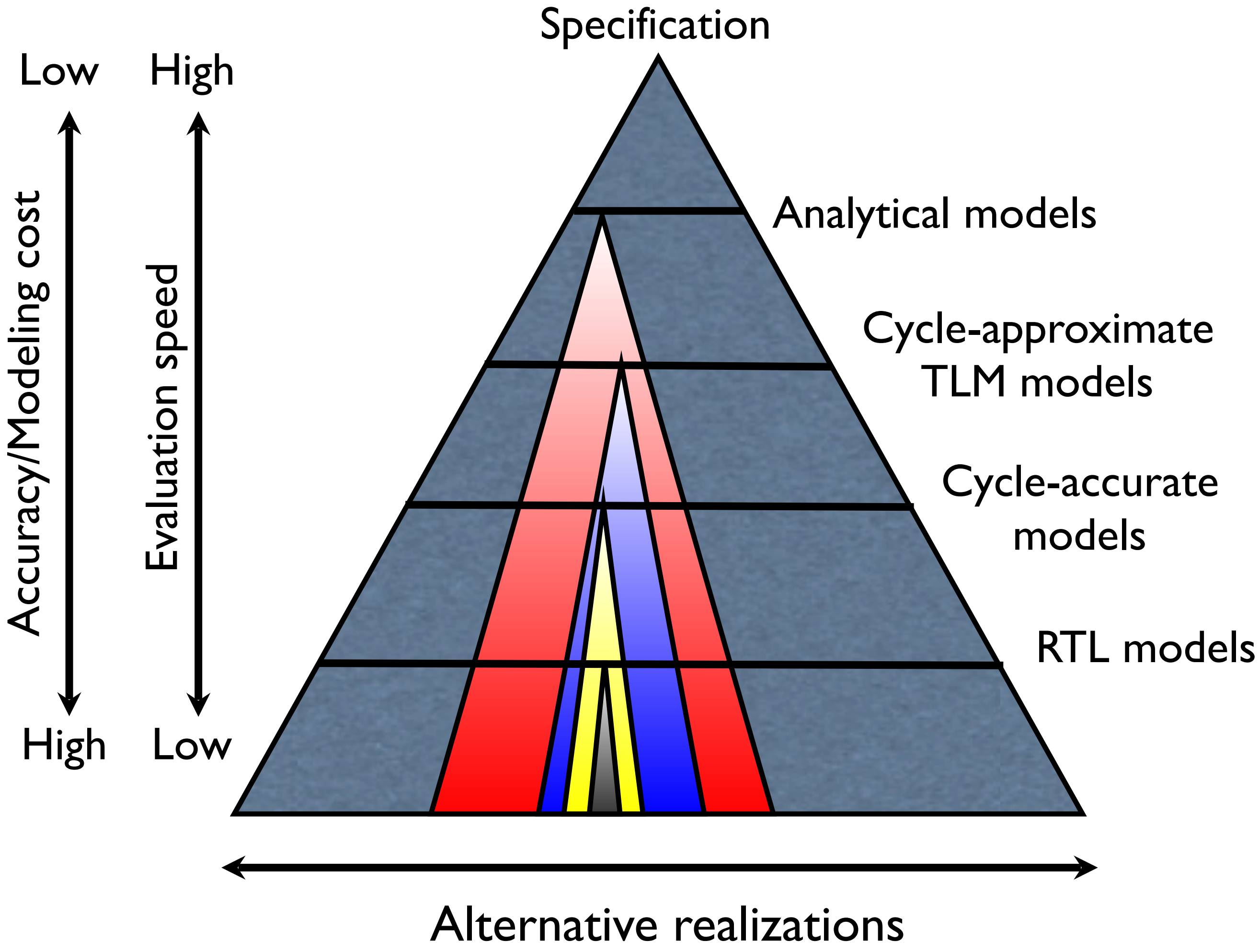
Evaluating a design point

System-level DSE: two elements



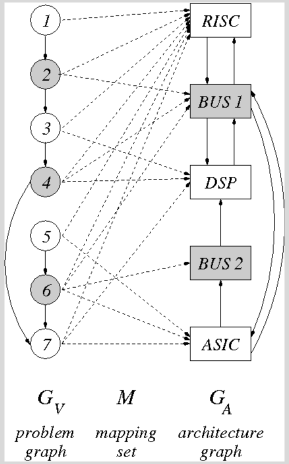
Source: Teich et al.

Evaluating a design point



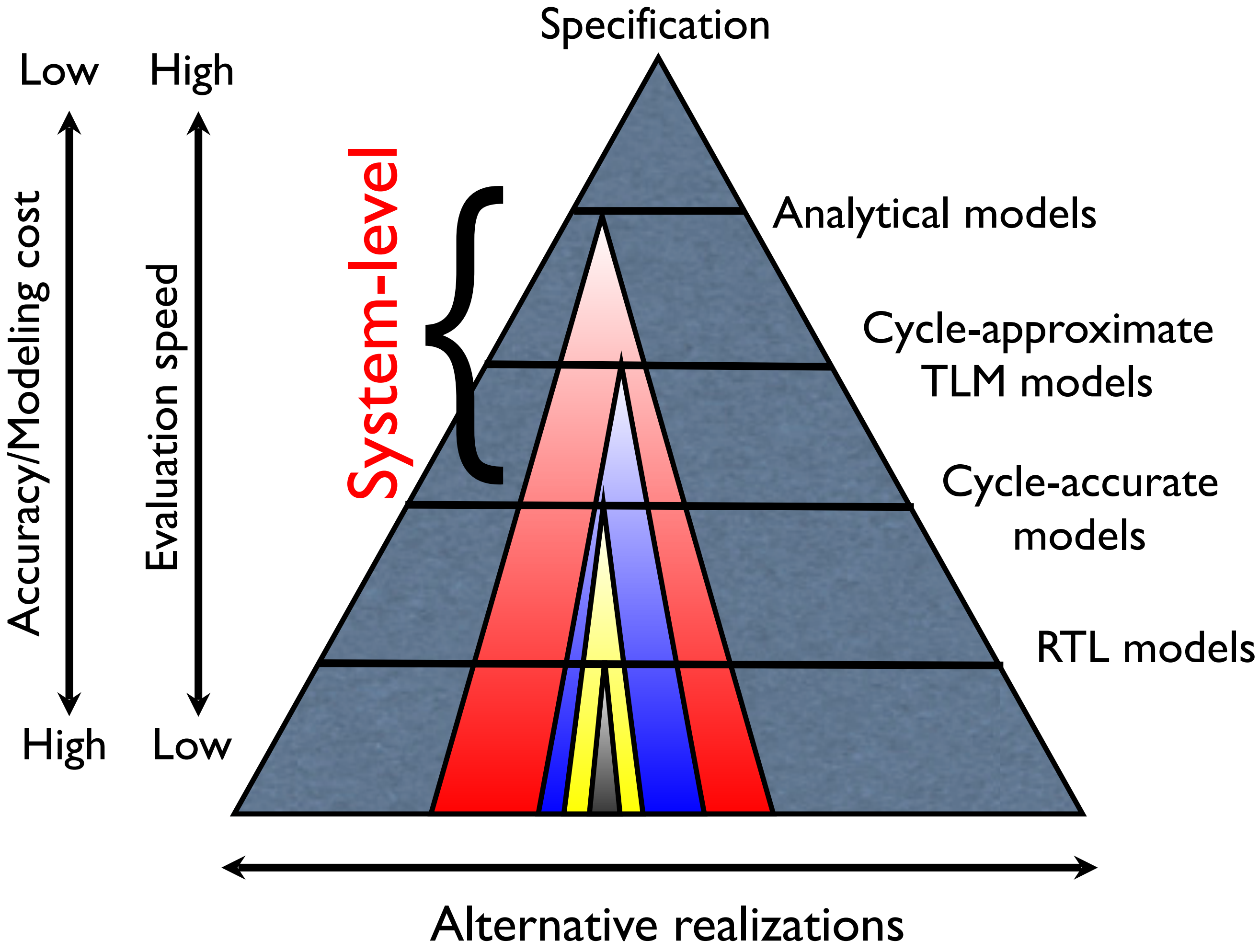
[IEEE Computer'01]

System-level DSE: two elements



Source: Teich et al.

Evaluating a design point



Specification

Analytical models

Cycle-approximate TLM models

Cycle-accurate models

RTL models

System-level

Low

High

Accuracy/Modeling cost

High

Low

Evaluation speed

Alternative realizations

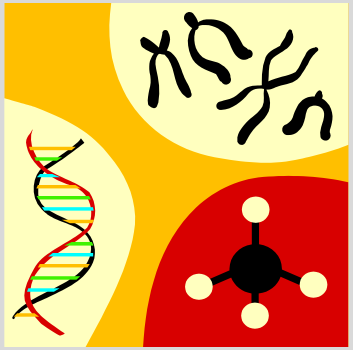
[IEEE Computer'01]

System-level DSE: two elements

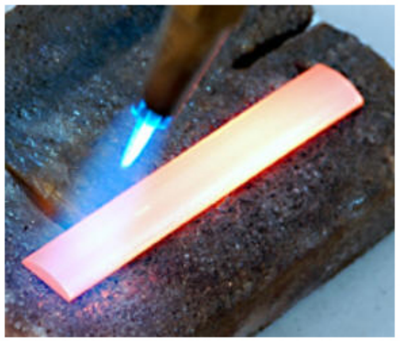
Searching the design space

- Exhaustive search usually is not feasible
- Typically, metaheuristics are used to search the design space
  - Only visit a relatively small number of design points
  - Single-objective or multi-objective optimization
  - Do not guarantee finding the global optimum

System-level DSE: two elements



Genetic Algorithm



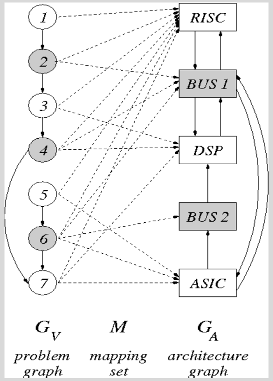
Simulated Annealing

Other metaheuristics

Ant Colony Optimization

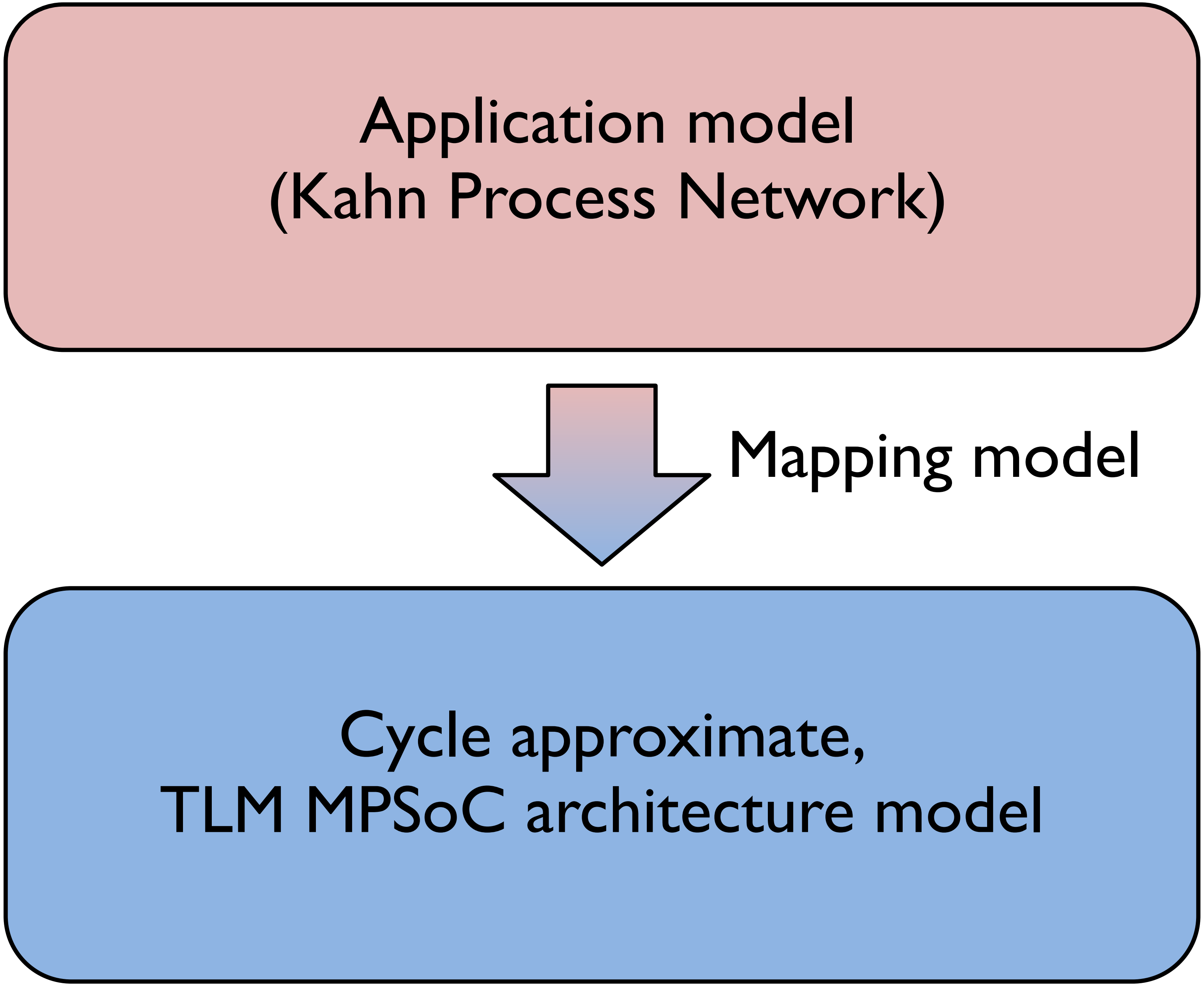
Searching the design space

- Exhaustive search usually is not feasible
- Typically, metaheuristics are used to search the design space
  - Only visit a relatively small number of design points
  - Single-objective or multi-objective optimization
  - Do not guarantee finding the global optimum



Evaluating a single design point

The Sesame simulation framework

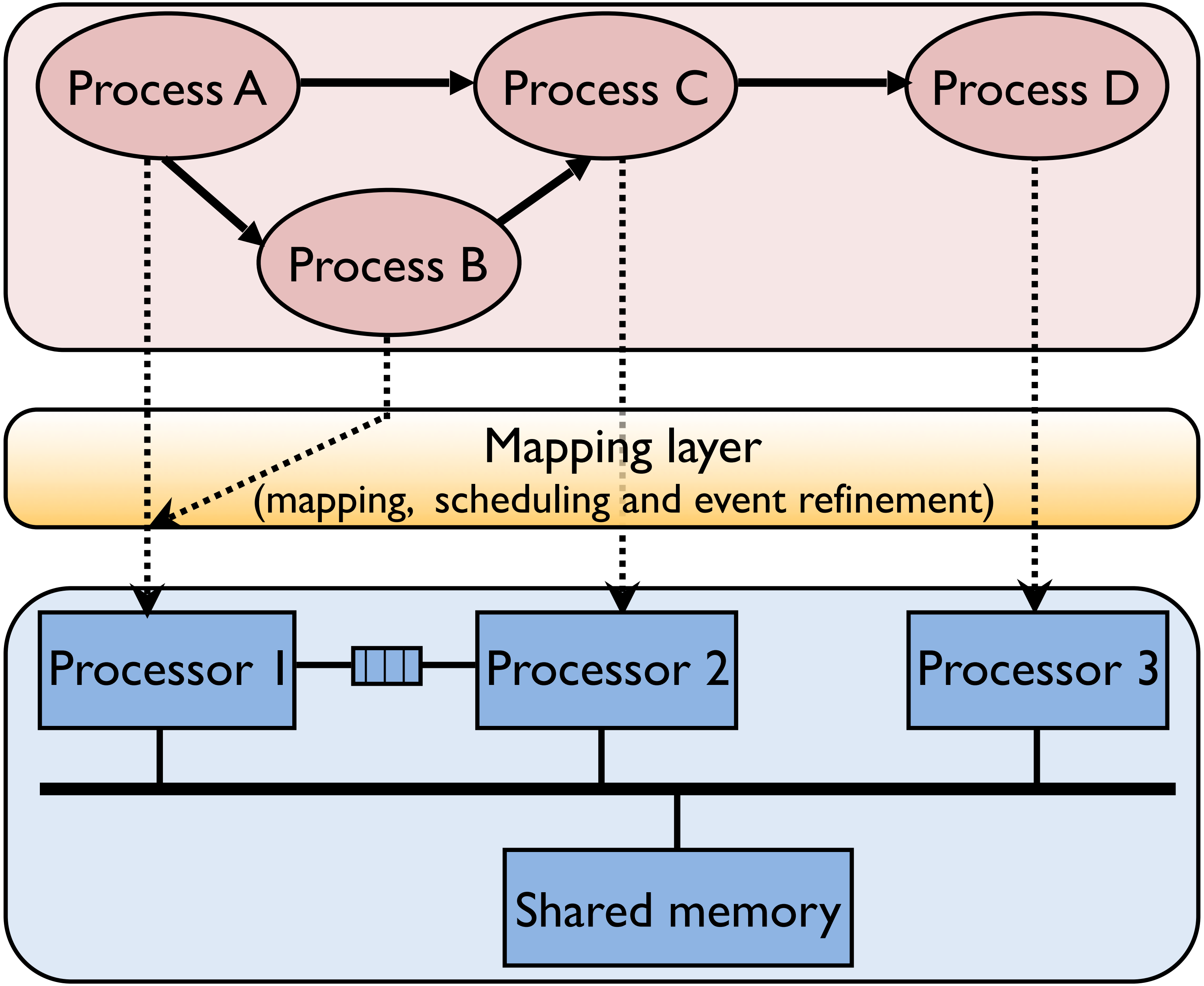


Application model
(Kahn Process Network)

Mapping model

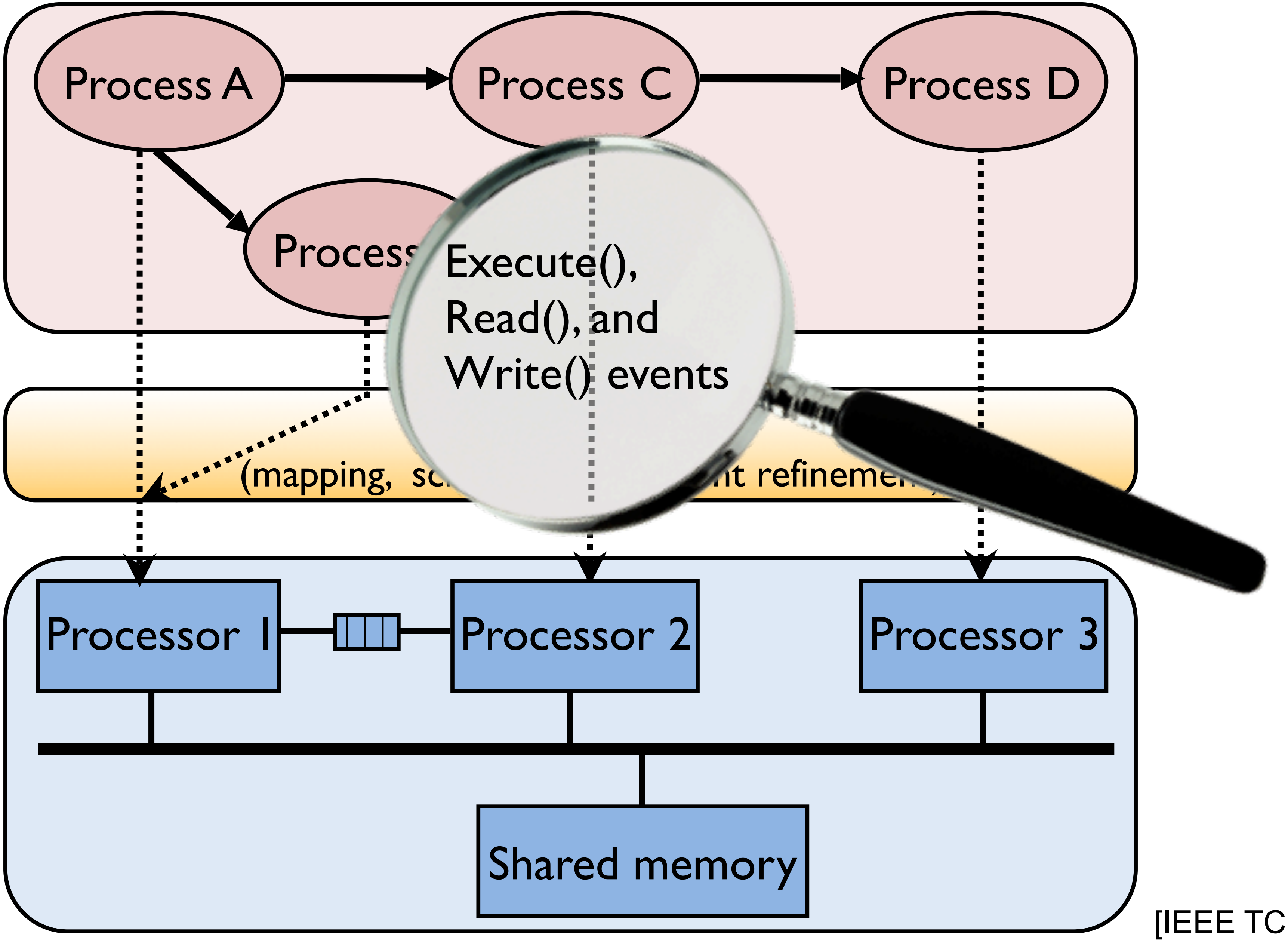
Cycle approximate,
TLM MPSoC architecture model

The Sesame simulation framework



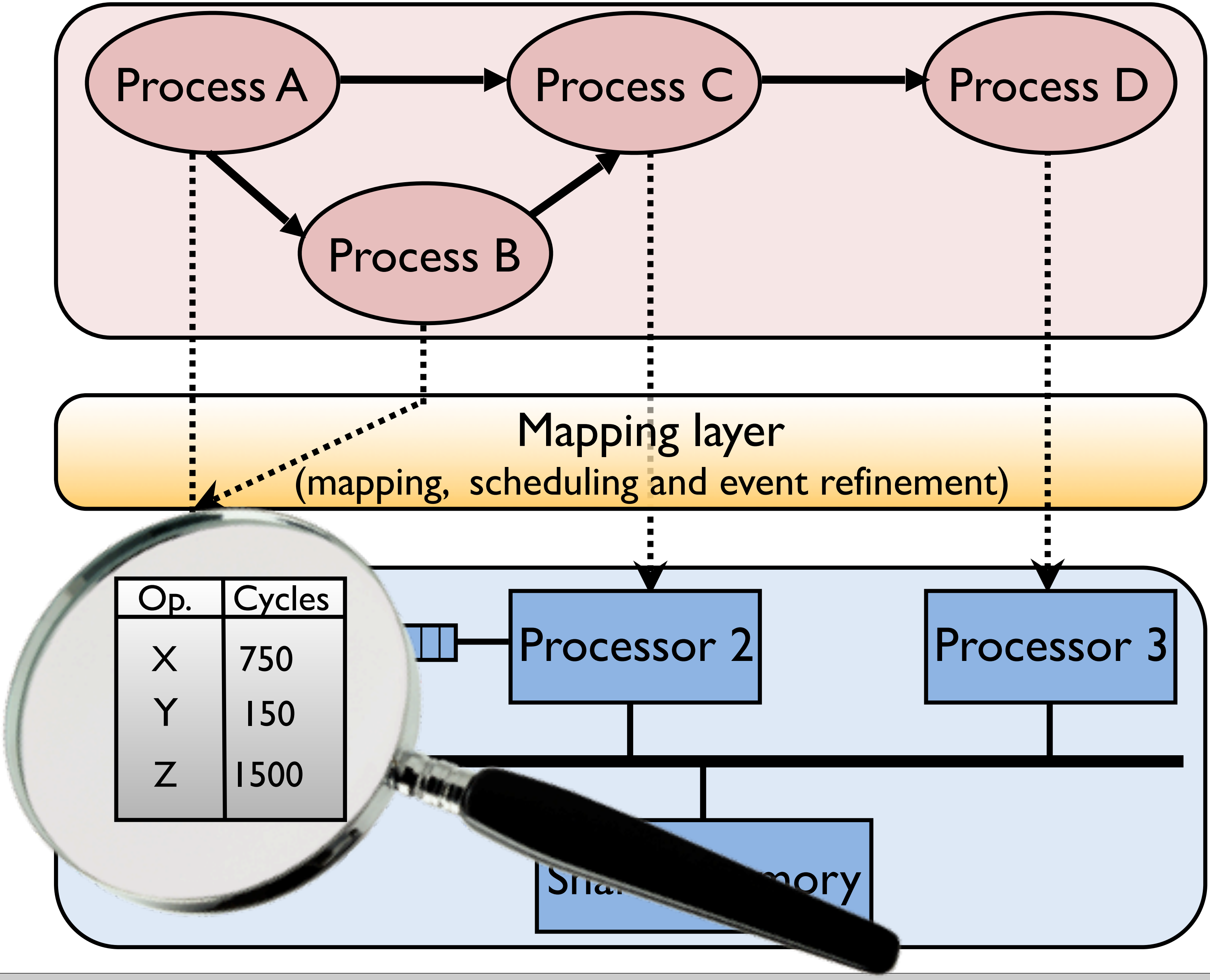
[IEEE TC'06]

The Sesame simulation framework



[IEEE TC'06]

The Sesame simulation framework

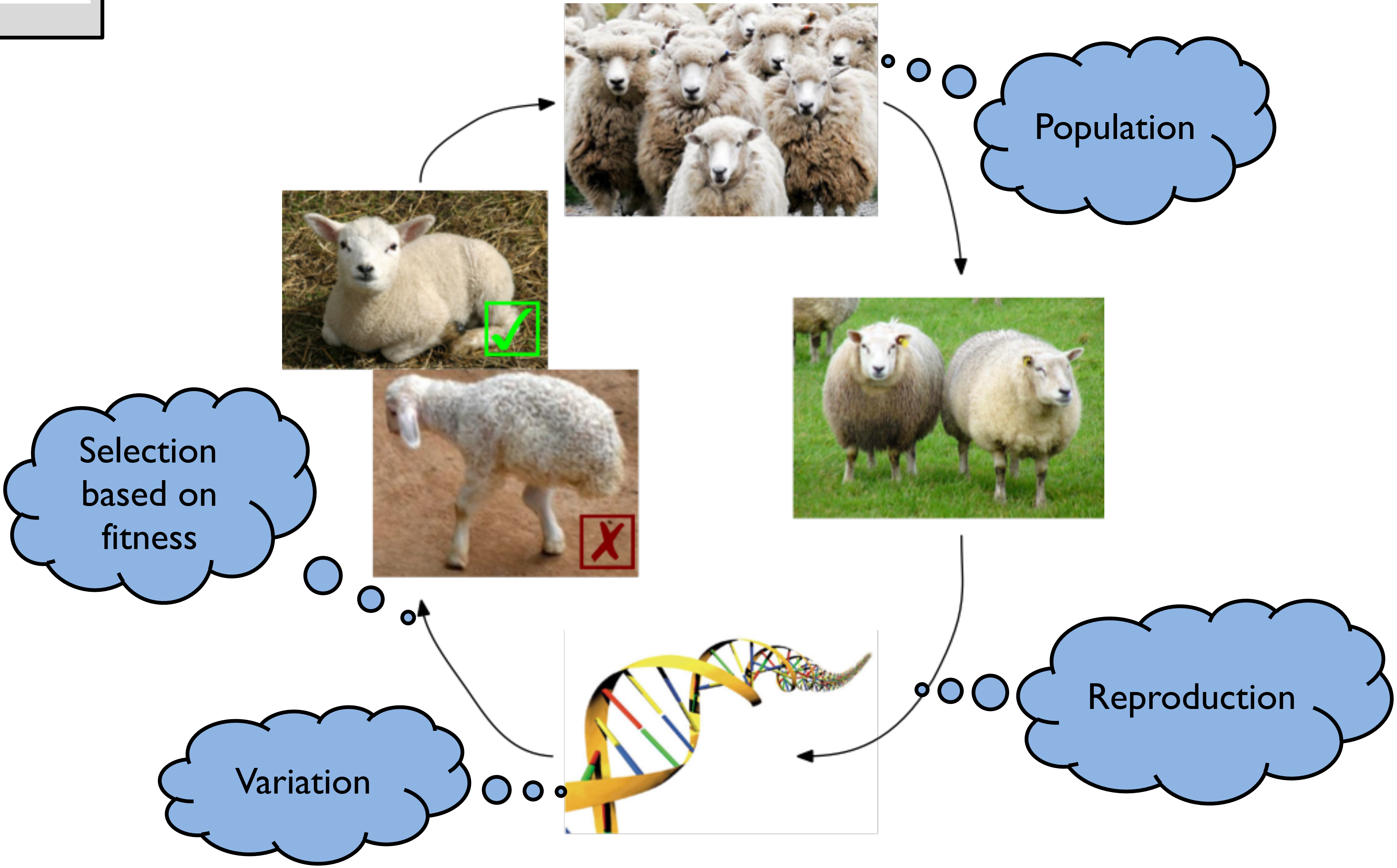


Op.	Cycles
X	750
Y	150
Z	1500

[IEEE TC'06]

Exploring the design space

Exploring the design space: GAs



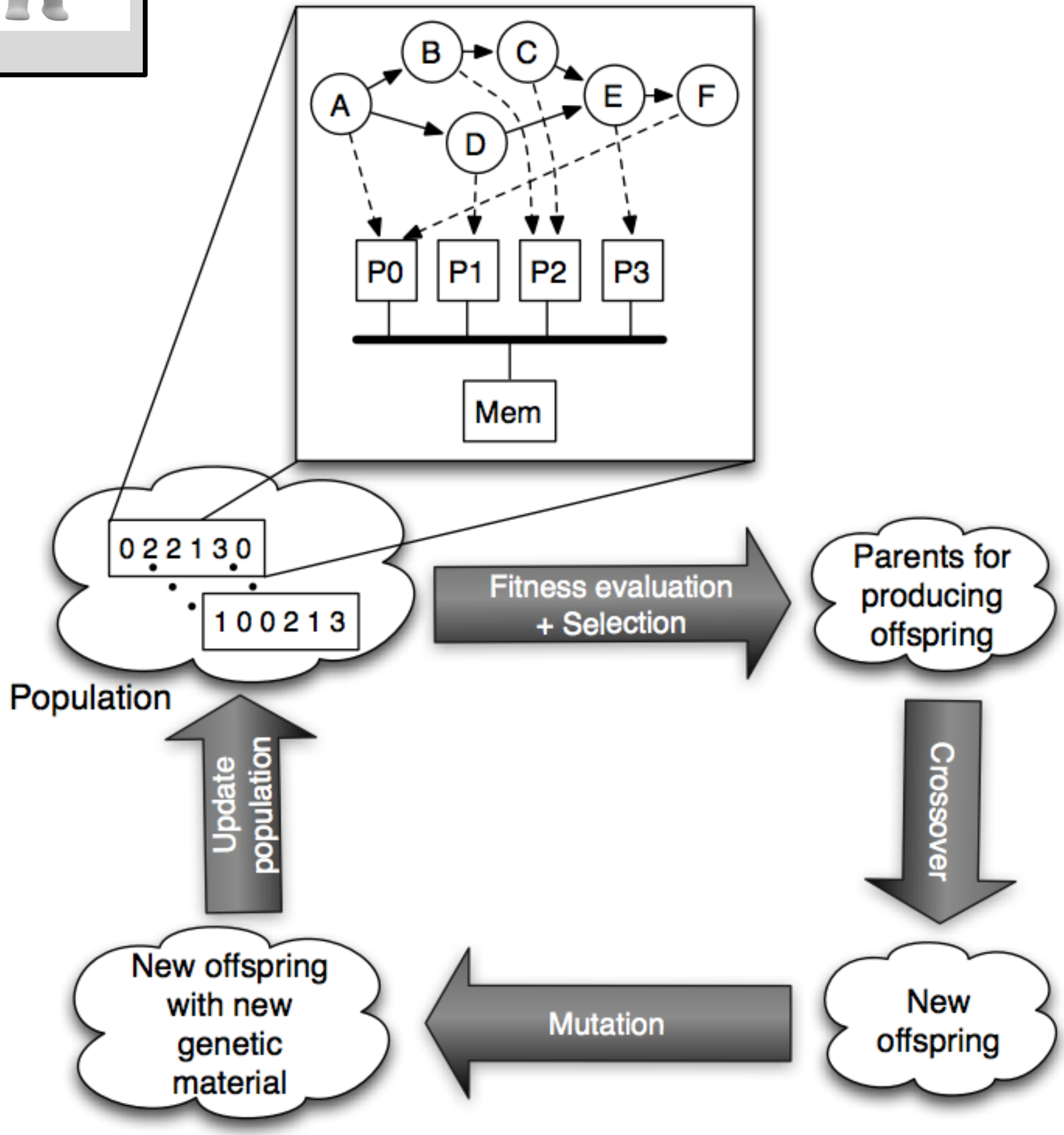
Population

Reproduction

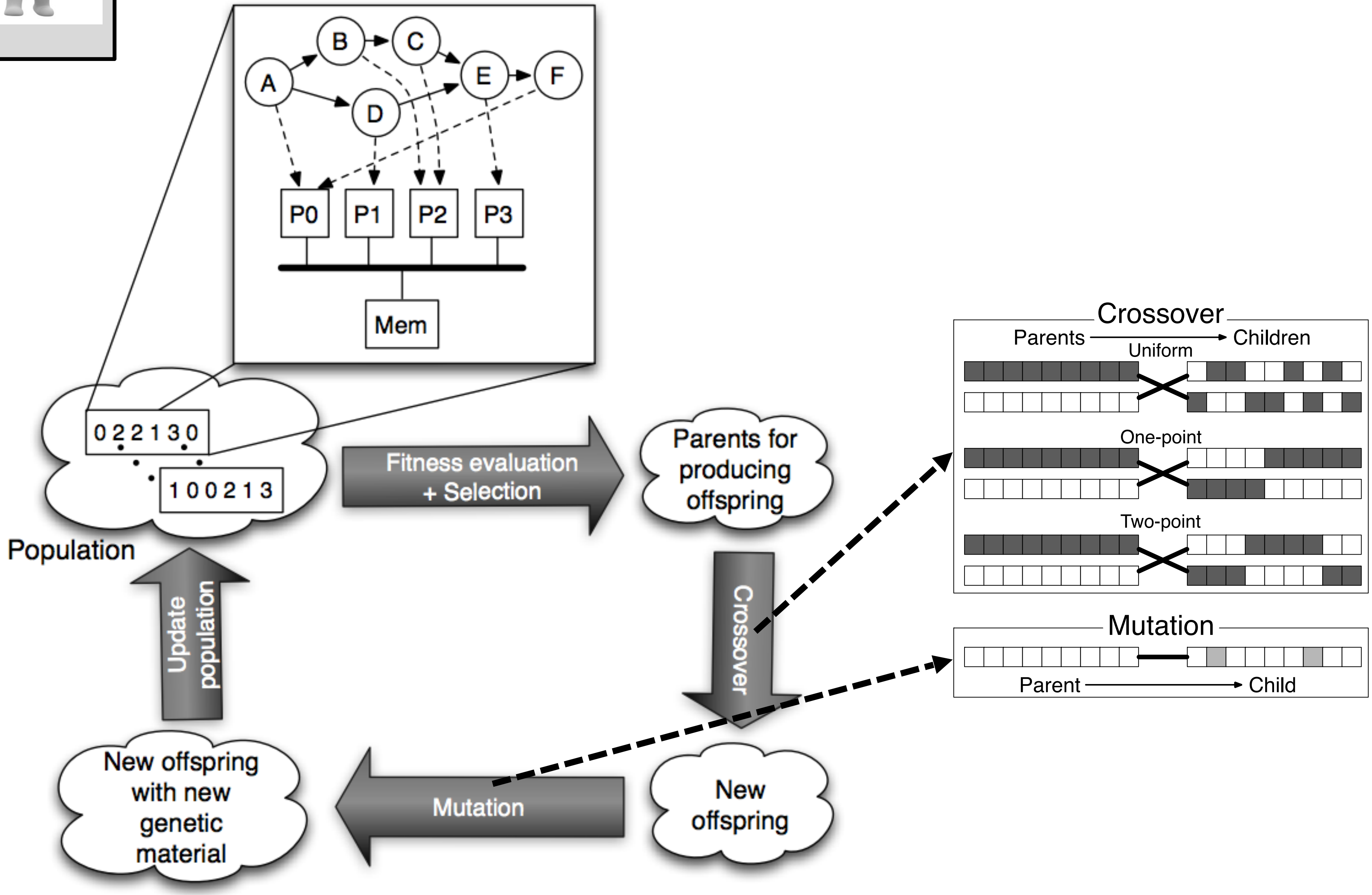
Variation

Selection based on fitness

Exploring the design space: GAs

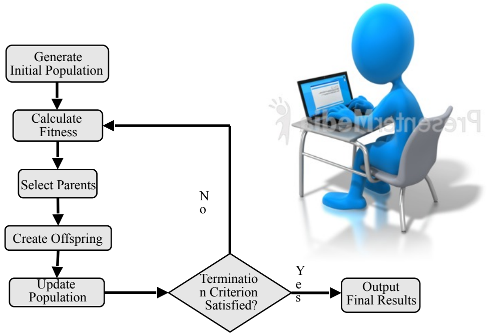

B
C
A
E
F
D
P0
P1
P2
P3
Mem
0 2 2 1 3 0
1 0 0 2 1 3
Population
Fitness evaluation + Selection
Parents for producing offspring
Crossover
New offspring
Mutation
New offspring with new genetic material
Update population

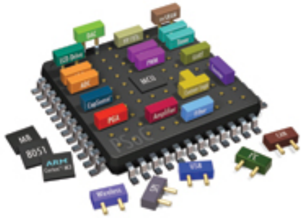
Exploring the design space: GAs



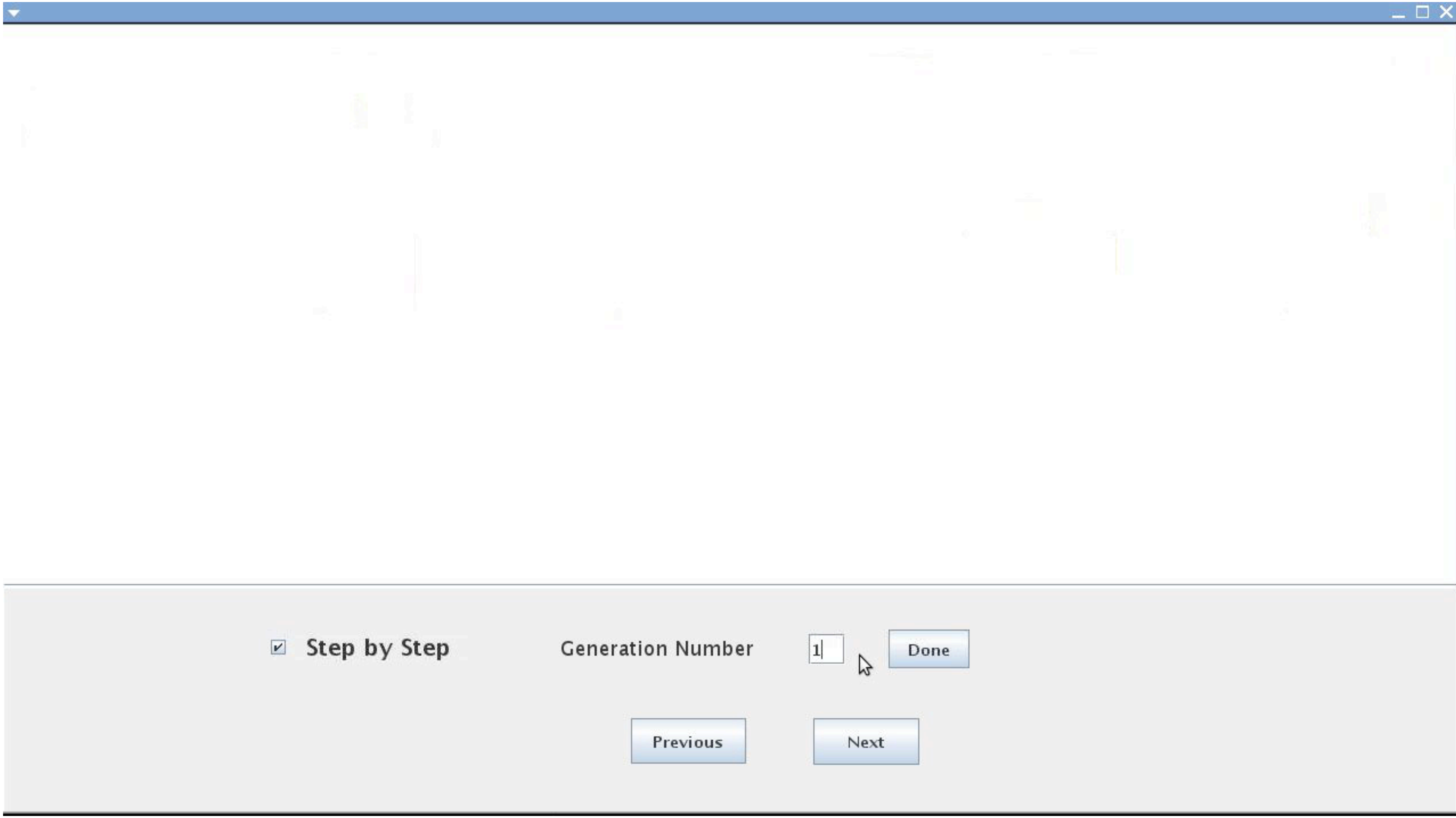
Analyzing the DSE process and its results

- Visualization support for three aspects:
  - Help algorithm developers to find the best optimization algorithm for their specific problem
  - Help designers to analysis the DSE results
  - Help decision makers to choose the most preferred solution





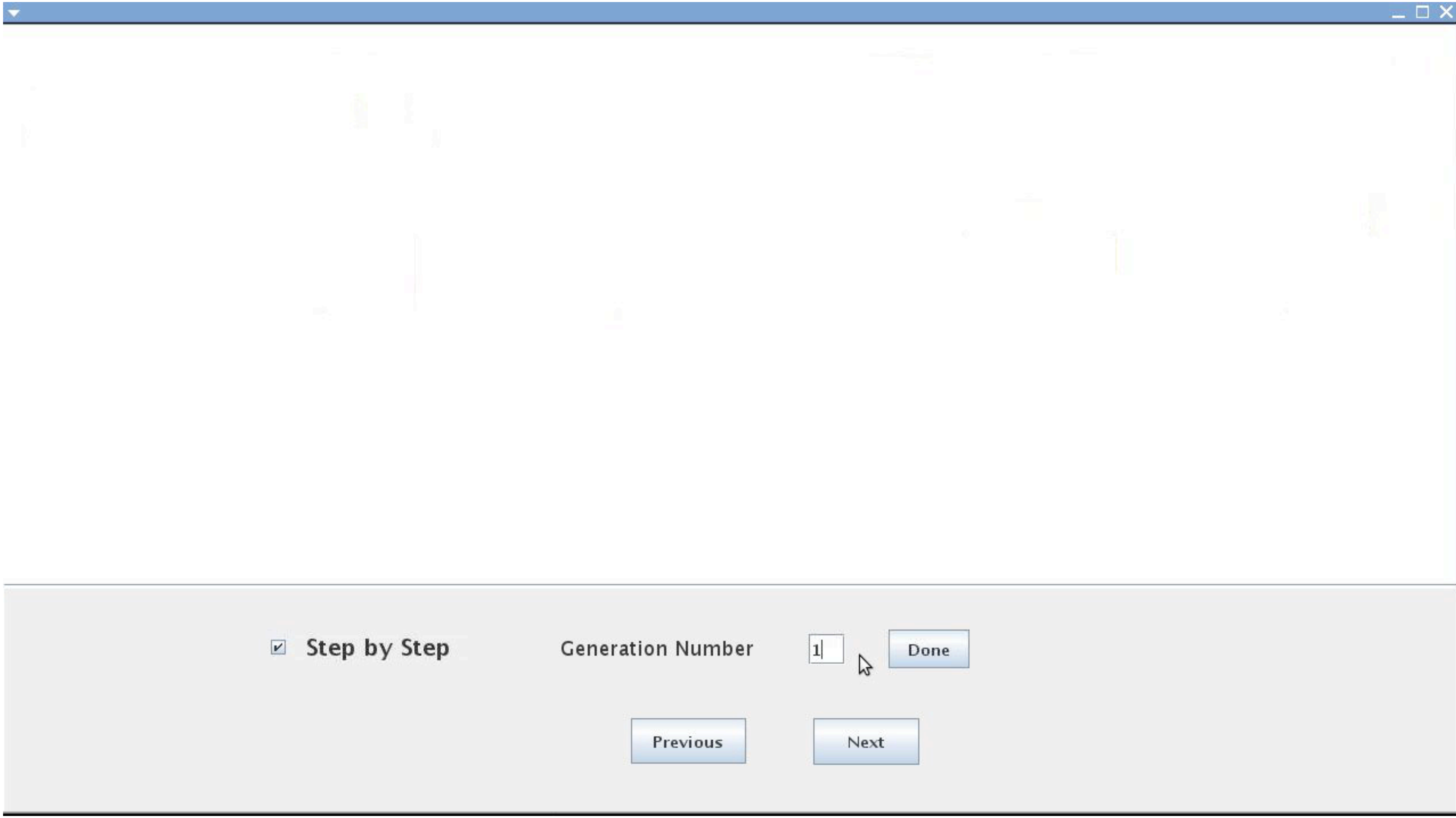
Analyzing the DSE process and its results (cont'd)



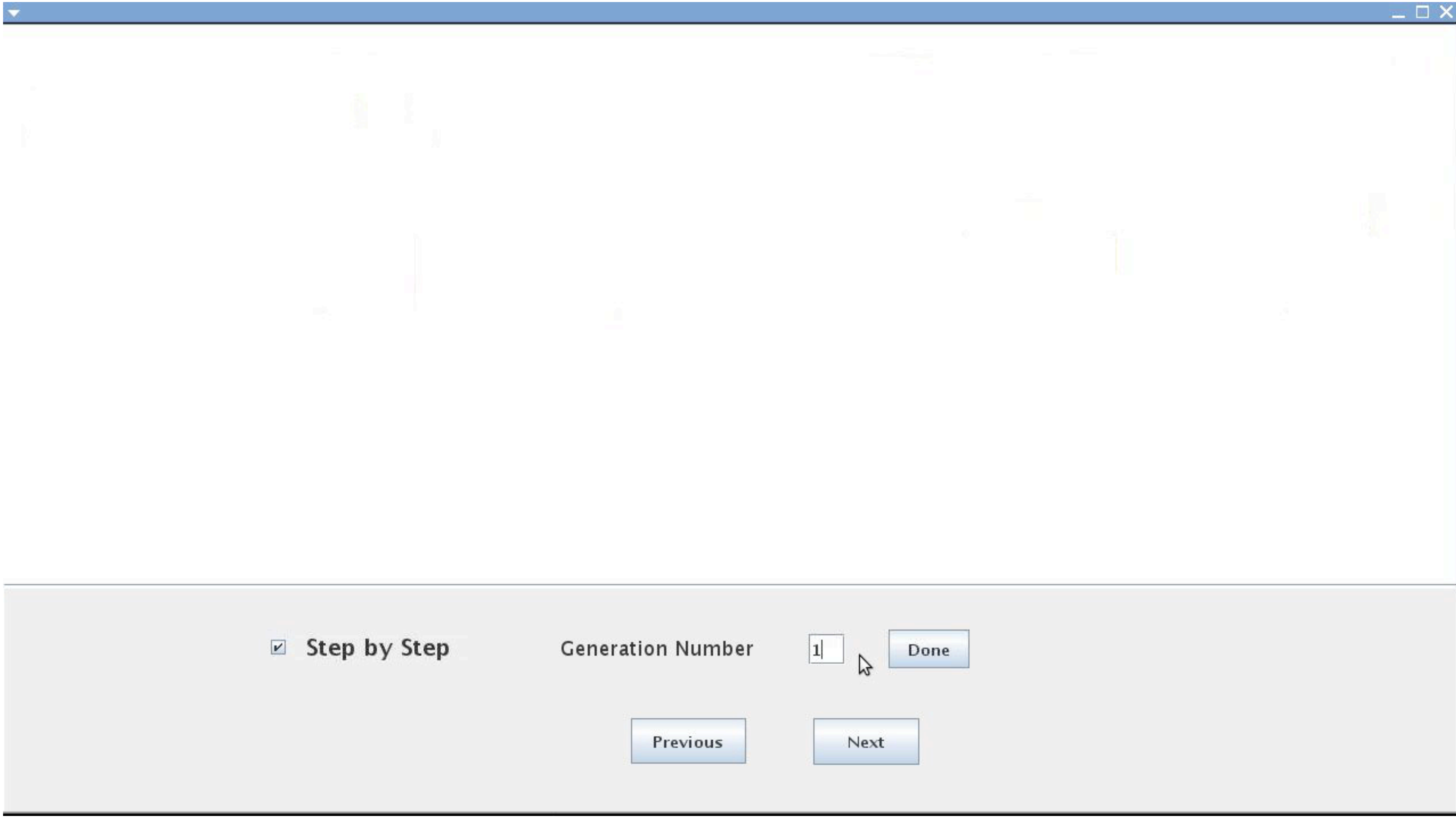
☑ Step by Step

Generation Number: 1

Analyzing the DSE process and its results (cont'd)

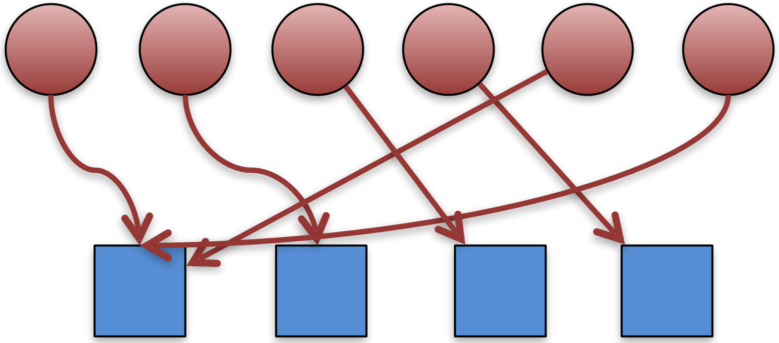


Analyzing the DSE process and its results (cont'd)



Exploiting domain knowledge

- For example, making the search process aware of "mapping symmetries"
  - GA encoding: [0,1,2,3,0,0]

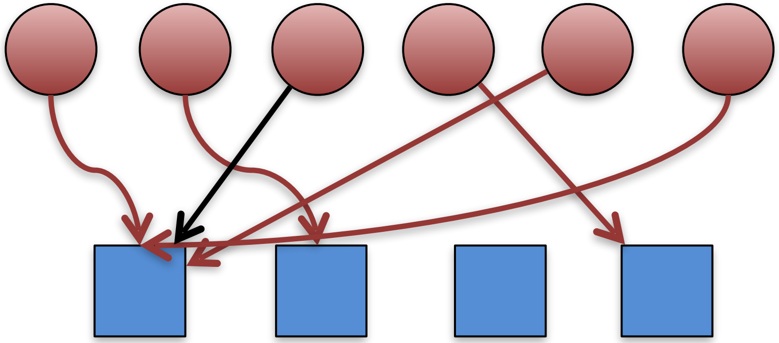


- A "Mapping distance" (δ) metric to maintain diversity and prevent evaluating duplicates
  - δ(a,a) = 0 (equality)
  - δ(a,b) = #transformations needed to achieve equality
  - δ([0,1,2,3,0,0] , [0,1,0,0,2,3]) = 4

Exploiting domain knowledge

- For example, making the search process aware of "mapping symmetries"
  - GA encoding: [0,1,2,3,0,0]

$\delta = 1$

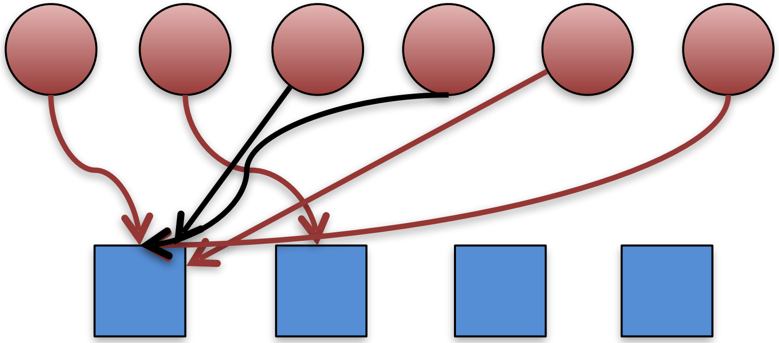


- A "Mapping distance" (δ) metric to maintain diversity and prevent evaluating duplicates
  - δ(a,a) = 0 (equality)
  - δ(a,b) = #transformations needed to achieve equality
  - δ([0,1,2,3,0,0] , [0,1,0,0,2,3]) = 4

Exploiting domain knowledge

- For example, making the search process aware of "mapping symmetries"
  - GA encoding: [0,1,2,3,0,0]

δ = 2

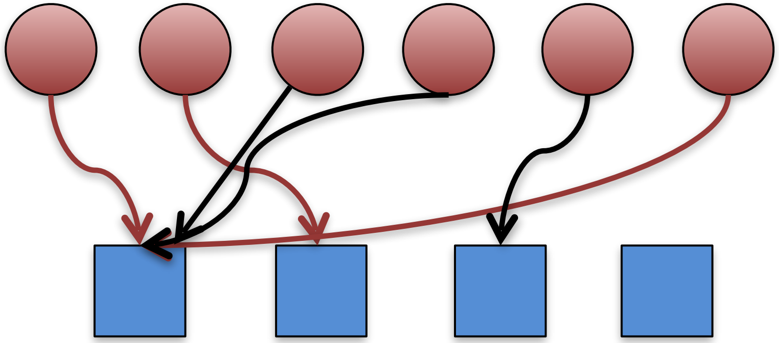


- A "Mapping distance" (δ) metric to maintain diversity and prevent evaluating duplicates
  - δ(a,a) = 0 (equality)
  - δ(a,b) = #transformations needed to achieve equality
  - δ([0,1,2,3,0,0] , [0,1,0,0,2,3]) = 4

Exploiting domain knowledge

- For example, making the search process aware of "mapping symmetries"
  - GA encoding: [0,1,2,3,0,0]

δ = 3

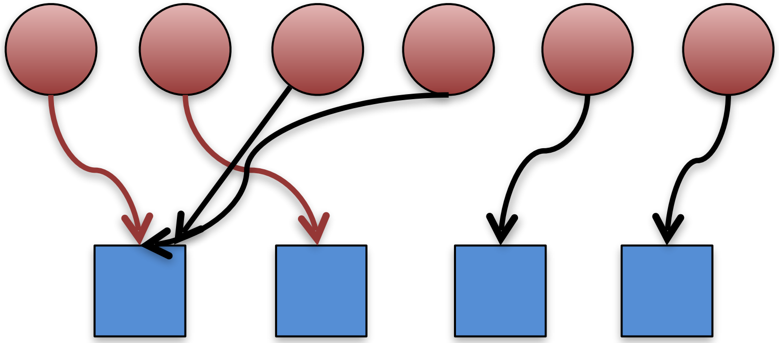


- A "Mapping distance" (δ) metric to maintain diversity and prevent evaluating duplicates
  - δ(a,a) = 0 (equality)
  - δ(a,b) = #transformations needed to achieve equality
  - δ([0,1,2,3,0,0] , [0,1,0,0,2,3]) = 4

Exploiting domain knowledge

- For example, making the search process aware of “mapping symmetries”
  - GA encoding: [0,1,2,3,0,0]

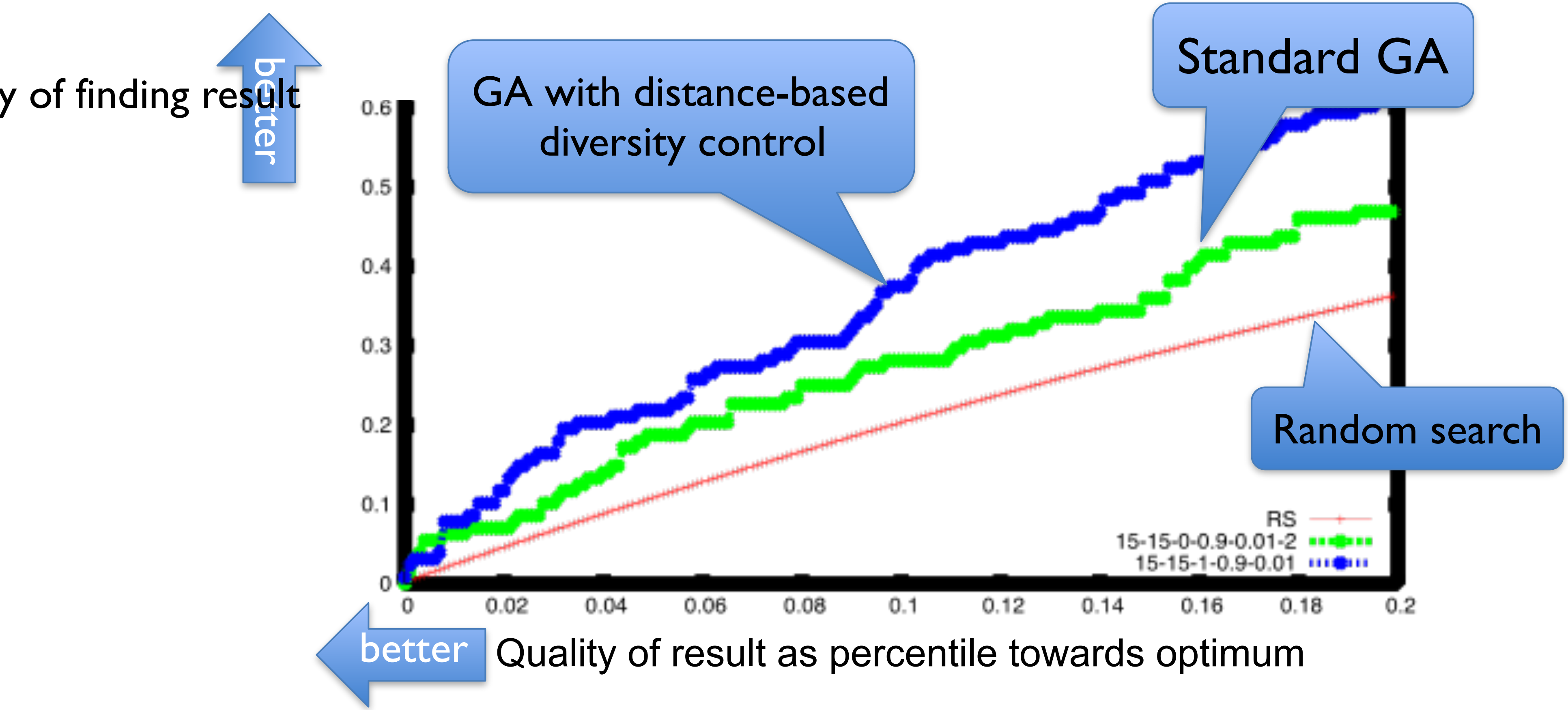
$\delta = 4$



- A “Mapping distance” (δ) metric to maintain diversity and prevent evaluating duplicates
  - δ(a,a) = 0 (equality)
  - δ(a,b) = #transformations needed to achieve equality
  - δ([0,1,2,3,0,0] , [0,1,0,0,2,3]) = 4

A small example

- 11-process application, 4-processor crossbar architecture
- Design space: 4^{11} = 4M design points (175275 unique)
- Summary of results of repeated GA experiments (dominating lines show better GA performance)



y of finding result

better

GA with distance-based diversity control

Standard GA

Random search

better

Quality of result as percentile towards optimum

Multi-functional embedded systems

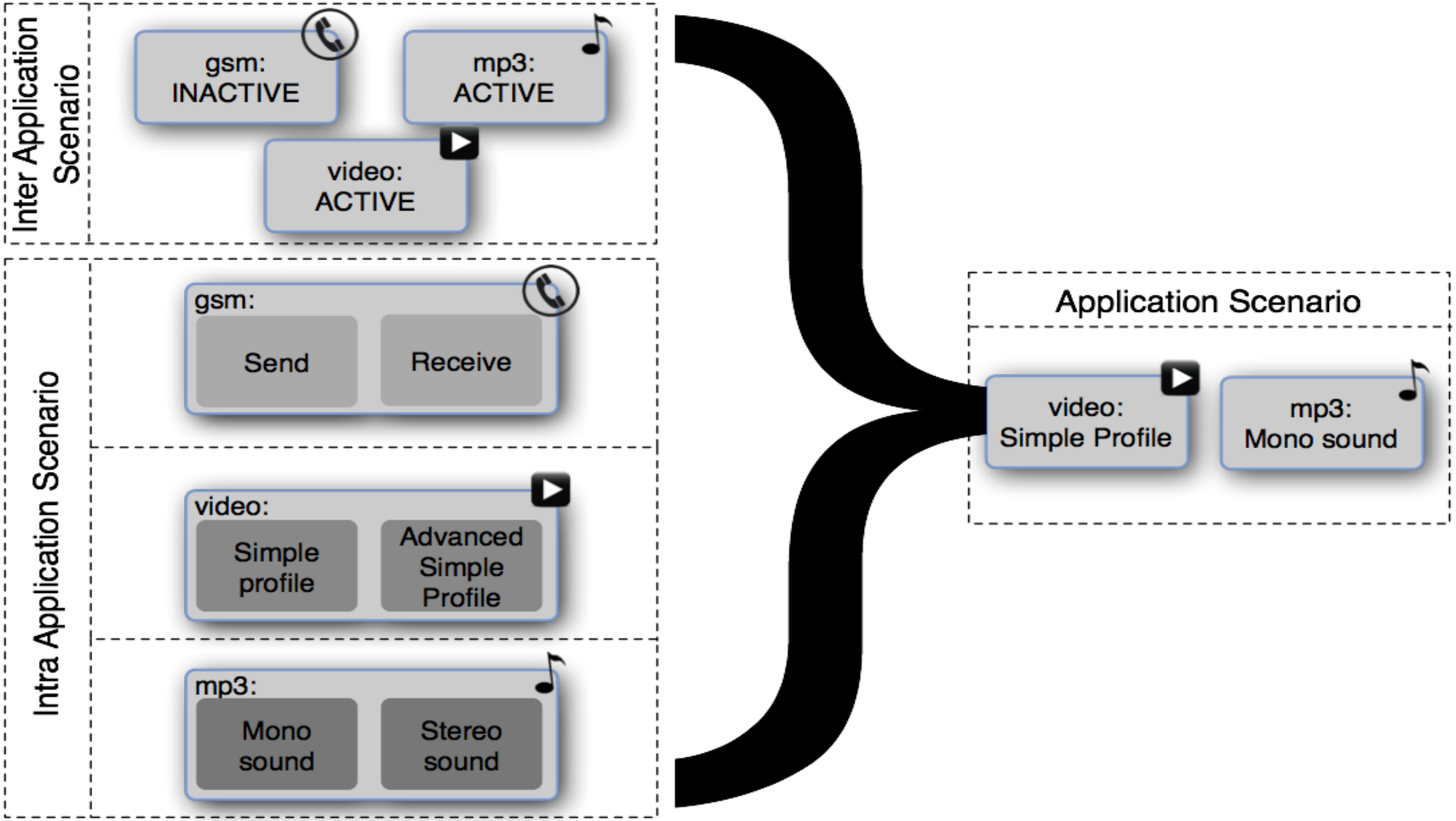
- Modern embedded systems need to support multiple applications and standards
- Multiple applications can be active simultaneously, contending for system resources
- Application workload may change over time
  - ✓ System demands change over time

Multi-functional embedded systems

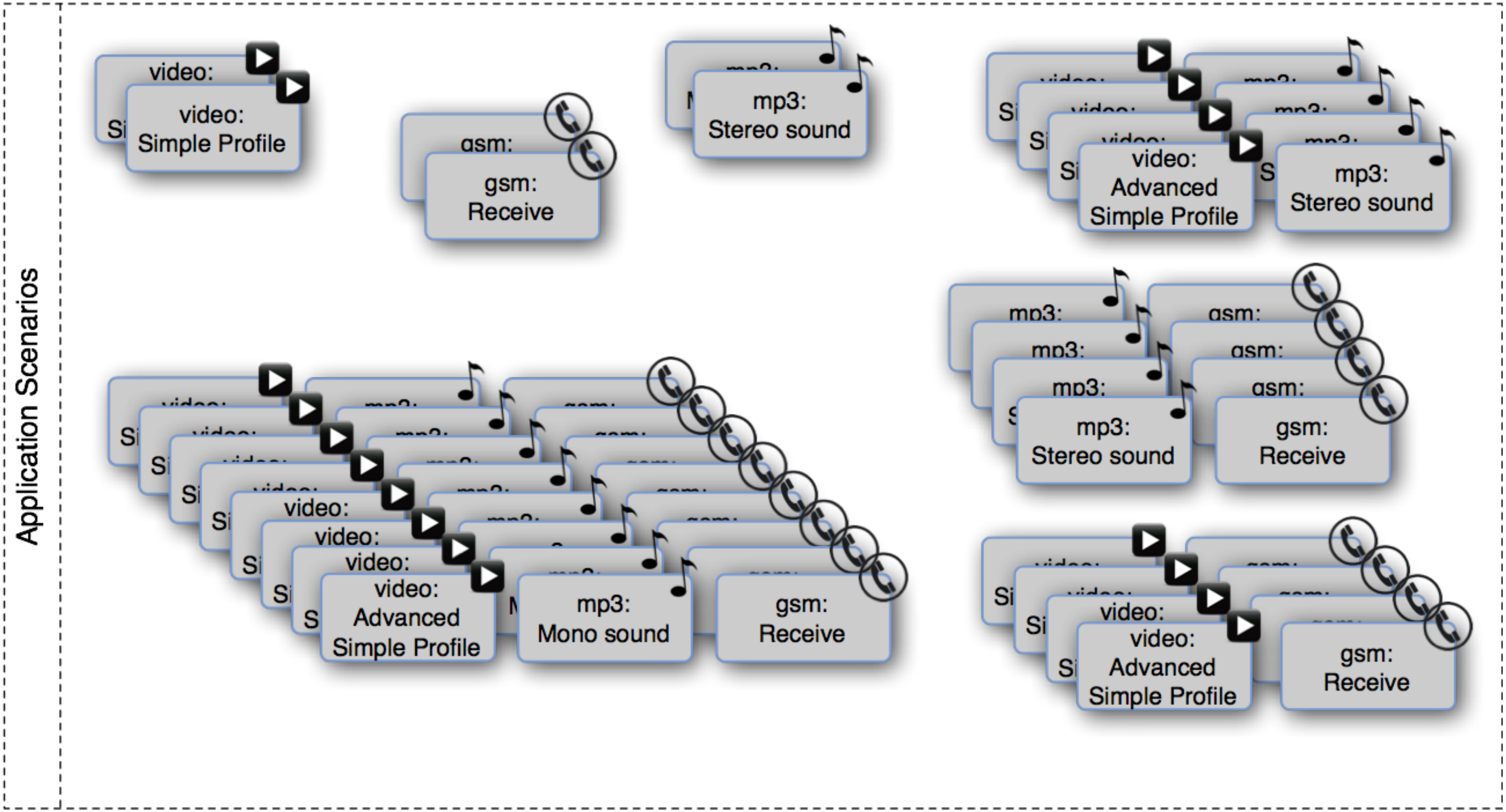
- Modern embedded systems need to support multiple applications and standards
- Multiple applications can be active simultaneously, contending for system resources
- Application workload may change over time
  - ✓ System demands change over time

How to perform DSE for multi-application workloads?
How to deal with dynamic workload behavior?

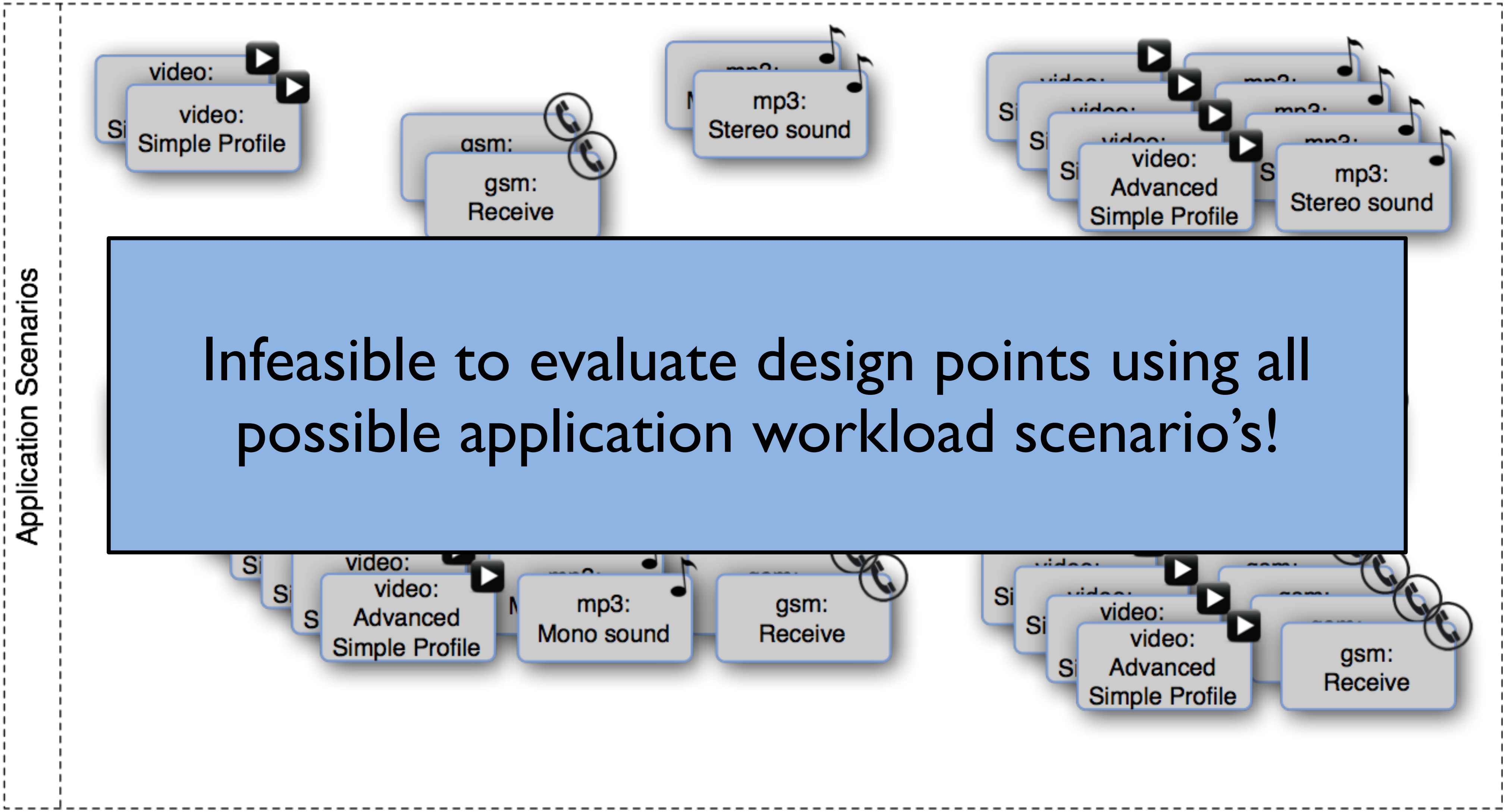
DSE for multi-application systems: scenario-based DSE



Scenarios: they are exponential

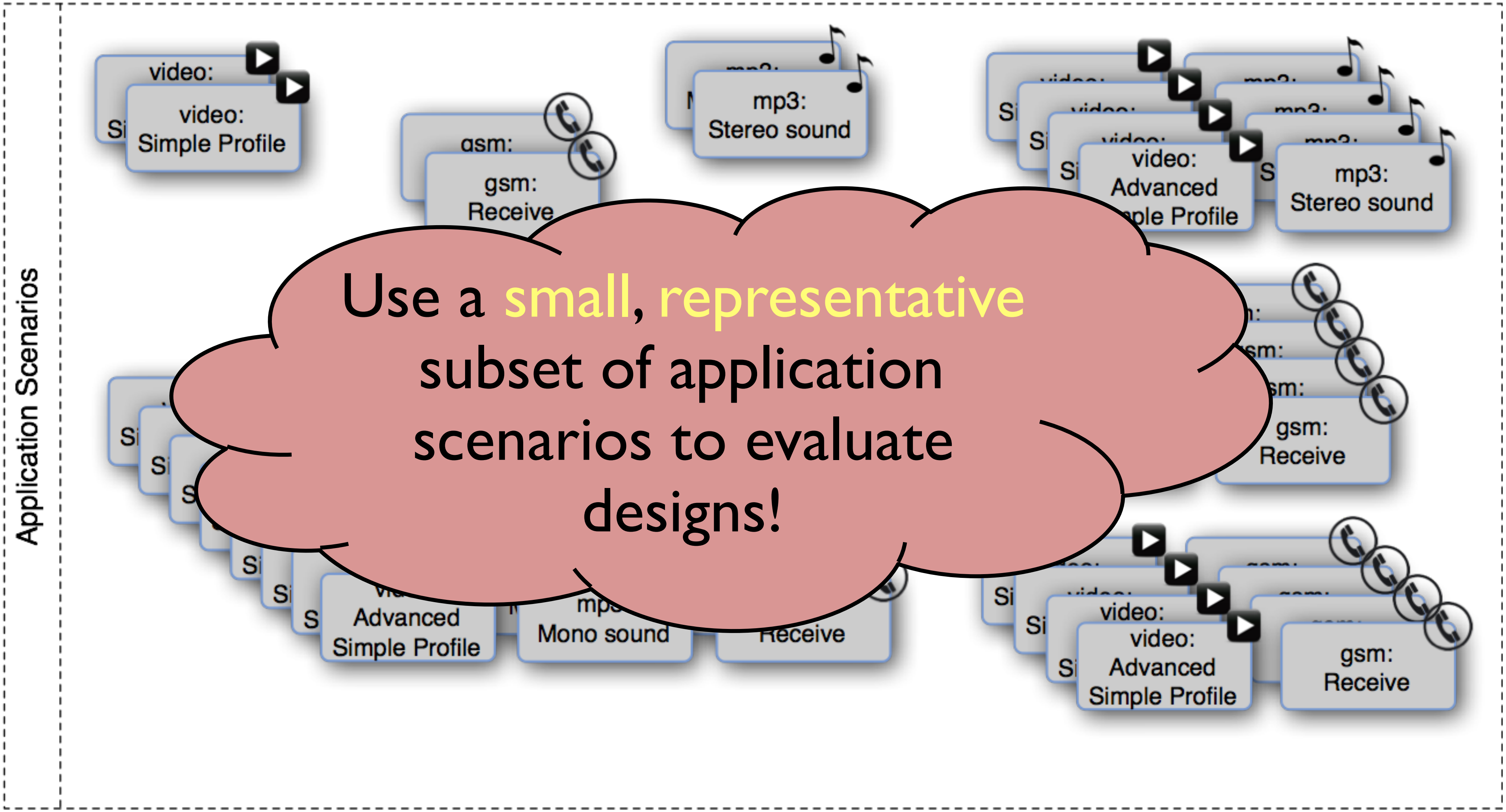


Scenarios: they are exponential



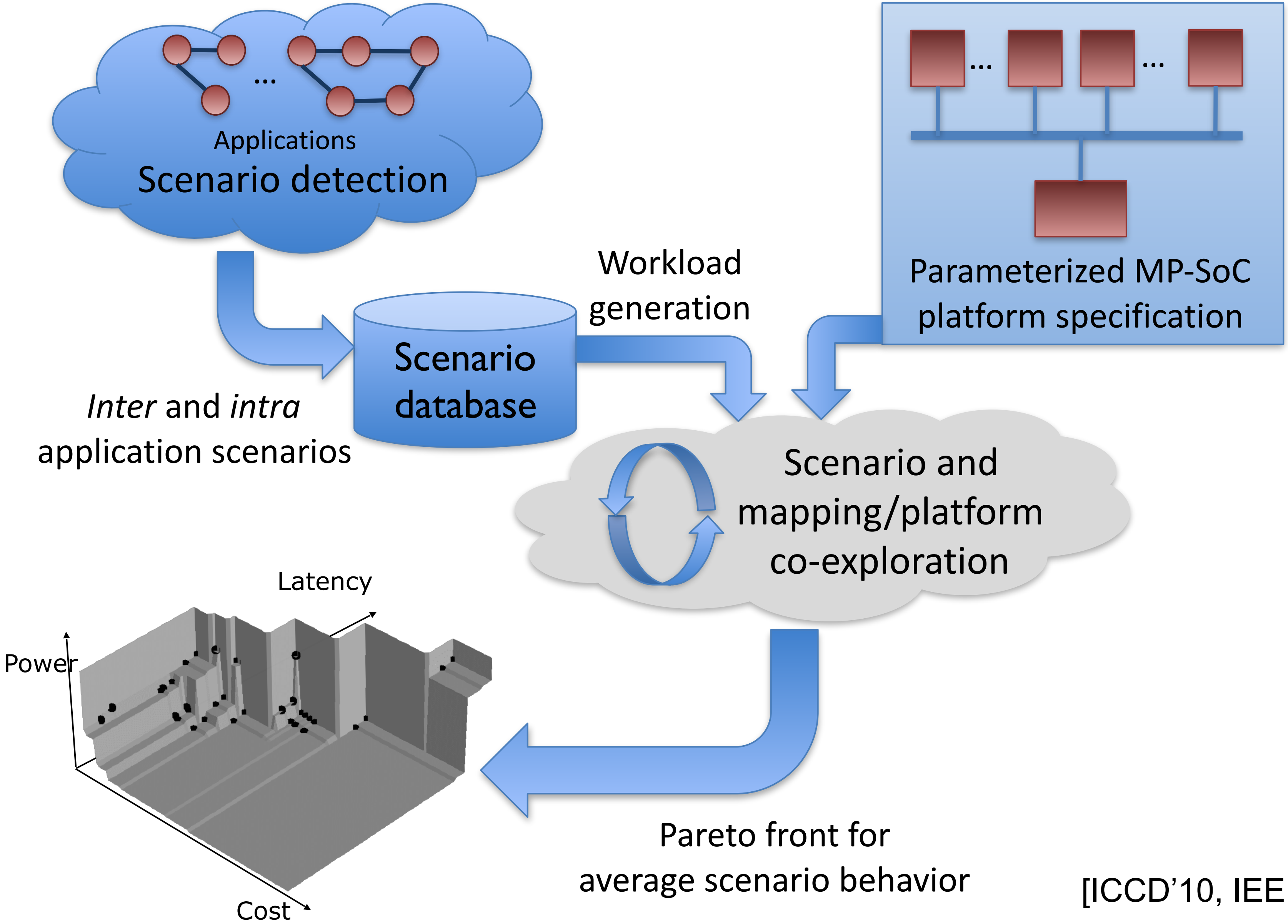
Infeasible to evaluate design points using all possible application workload scenario's!

Scenarios: they are exponential



Use a small, representative subset of application scenarios to evaluate designs!

Scenario-based DSE



Applications
Scenario detection
…
Scenario database
Workload generation
…
…
Parameterized MP-SoC platform specification
Inter and *intra* application scenarios
Scenario and mapping/platform co-exploration
Latency
Power
Cost
Pareto front for average scenario behavior

[ICCD'10, IEEE TCAD'13]

The need for system adaptivity

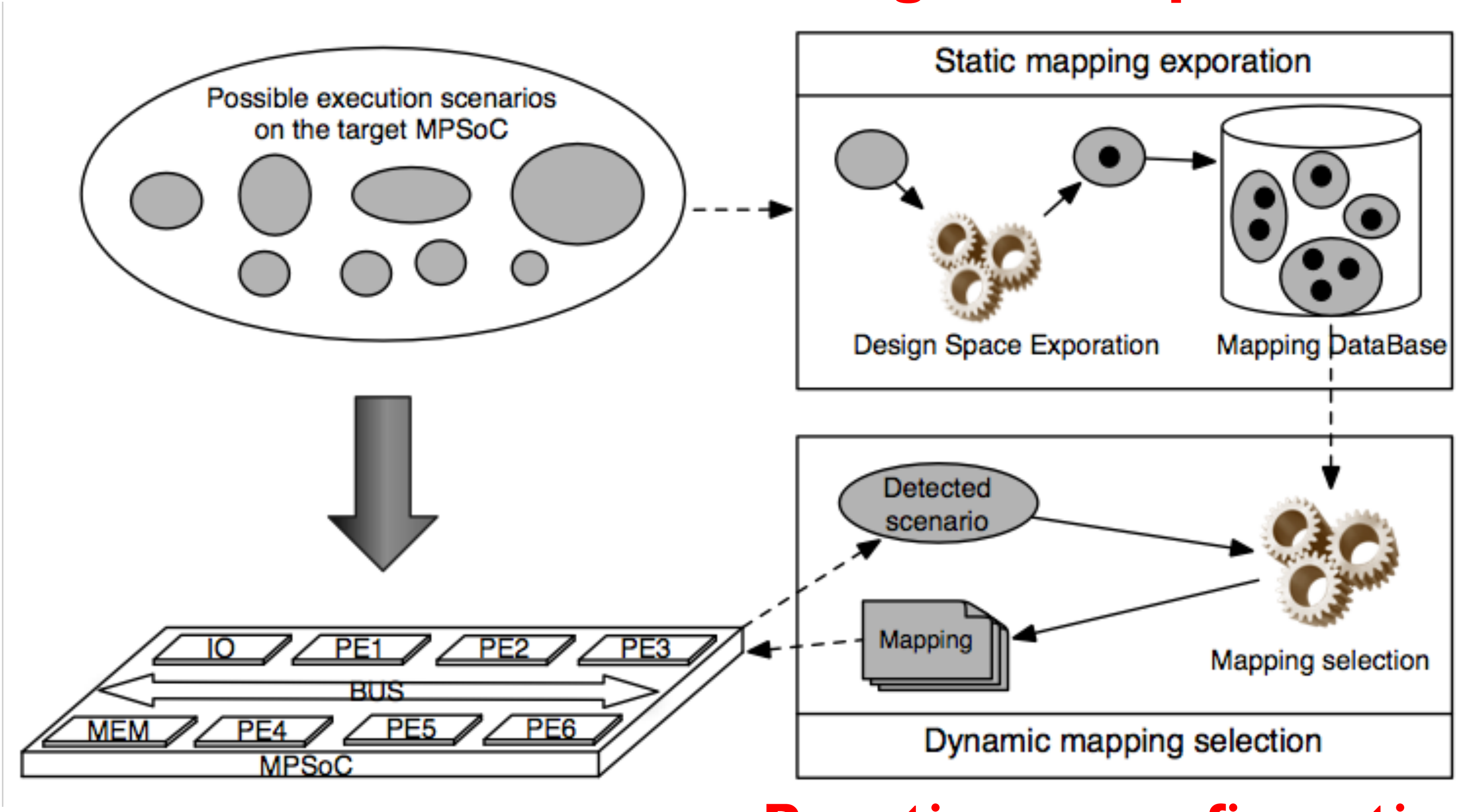
- Cope with changing (demands of) application workloads
- Dynamic QoS management allowing to trade off different system qualities like performance, precision and power consumption
- Cope with transient and/or permanent system faults

The need for system adaptivity

- Cope with changing (demands of) application workloads
- Dynamic QoS management allowing to trade off different system qualities like performance, precision and power consumption
- Cope with transient and/or permanent system faults
- Types of adaptivity:
  - Component reconfiguration (e.g., DVFS, reconfigurable HW, reconfigurable network, etc.)
  - Run-time (re-)mapping of application tasks

Run-time adaptive systems

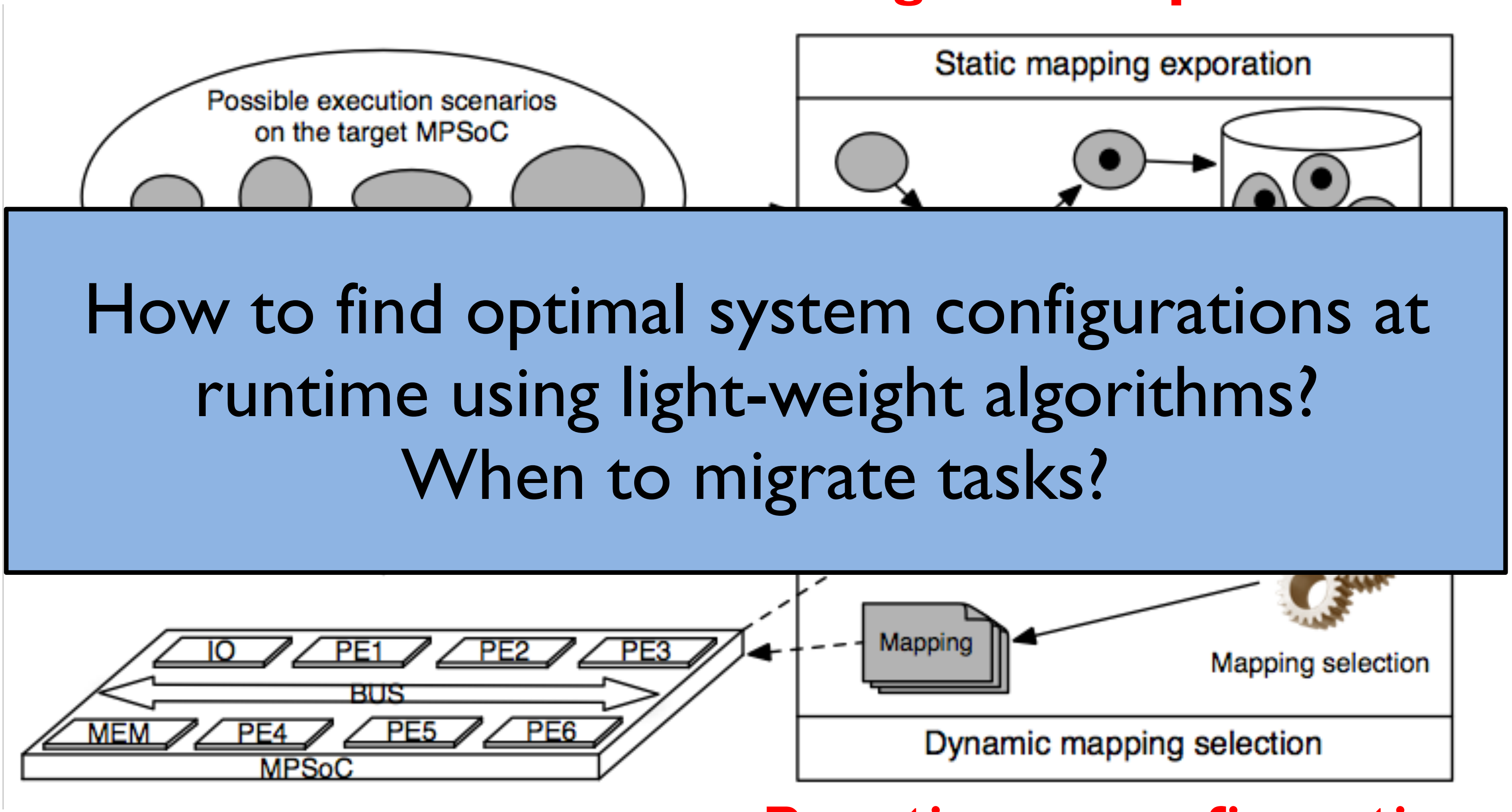
Design-time optimization



Run-time reconfiguration

Run-time adaptive systems

Design-time optimization


Static mapping exporation
Possible execution scenarios on the target MPSoC
Mapping
Mapping selection
Dynamic mapping selection
IO
PE1
PE2
PE3
BUS
MEM
PE4
PE5
PE6
MPSoC

How to find optimal system configurations at runtime using light-weight algorithms?
When to migrate tasks?

Run-time reconfiguration

Adaptive MPSoCs

- Re-mapping (migration) of tasks not always beneficial!
  - Dependent on workload scenario duration

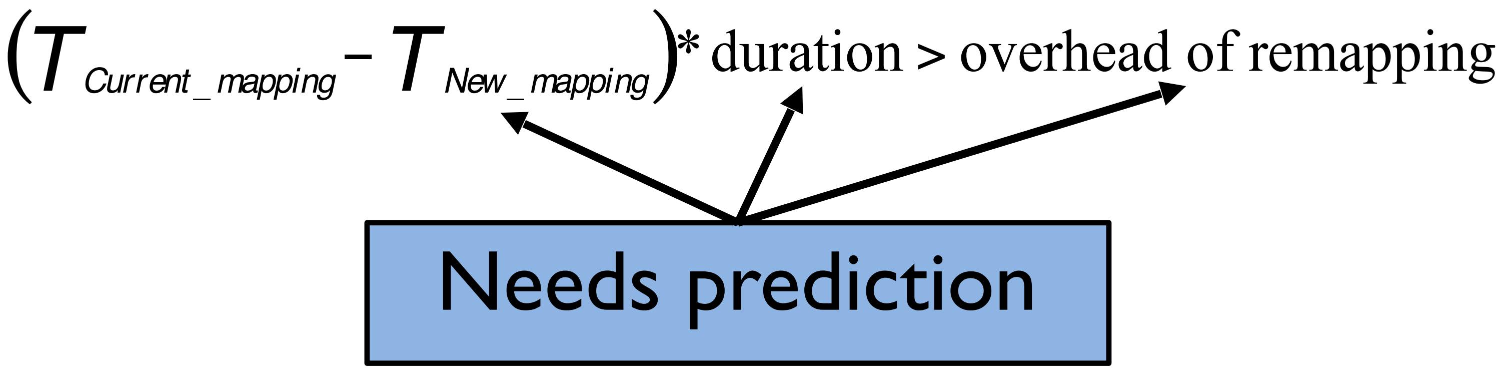
Adaptive MPSoCs

- Re-mapping (migration) of tasks not always beneficial!
  - Dependent on workload scenario duration
- This leads to a need for adaptivity throttling
  - Predict whether or not it is beneficial to re-map

$$(T_{Current\_mapping} - T_{New\_mapping}) * \text{duration} > \text{overhead of remapping}$$

Adaptive MPSoCs

- Re-mapping (migration) of tasks not always beneficial!
  - Dependent on workload scenario duration
- This leads to a need for adaptivity throttling
  - Predict whether or not it is beneficial to re-map



$$\left(T_{Current\_mapping} - T_{New\_mapping}\right) * \text{duration} > \text{overhead of remapping}$$

Needs prediction

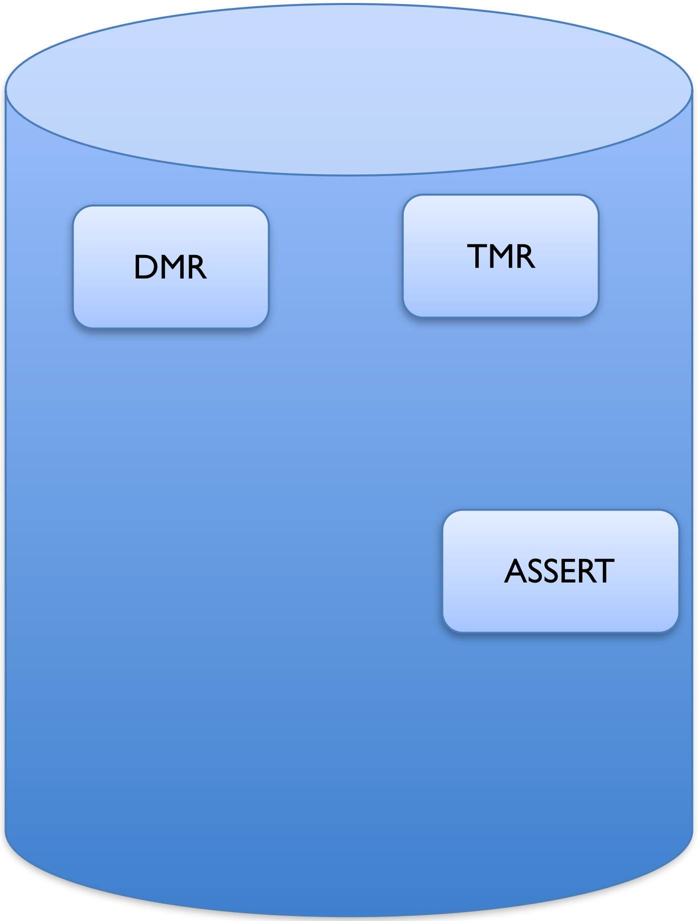
Incorporating additional optimization objectives

Reliability-aware DSE

Incorporating fault-tolerance as design objective

Reliability-aware DSE

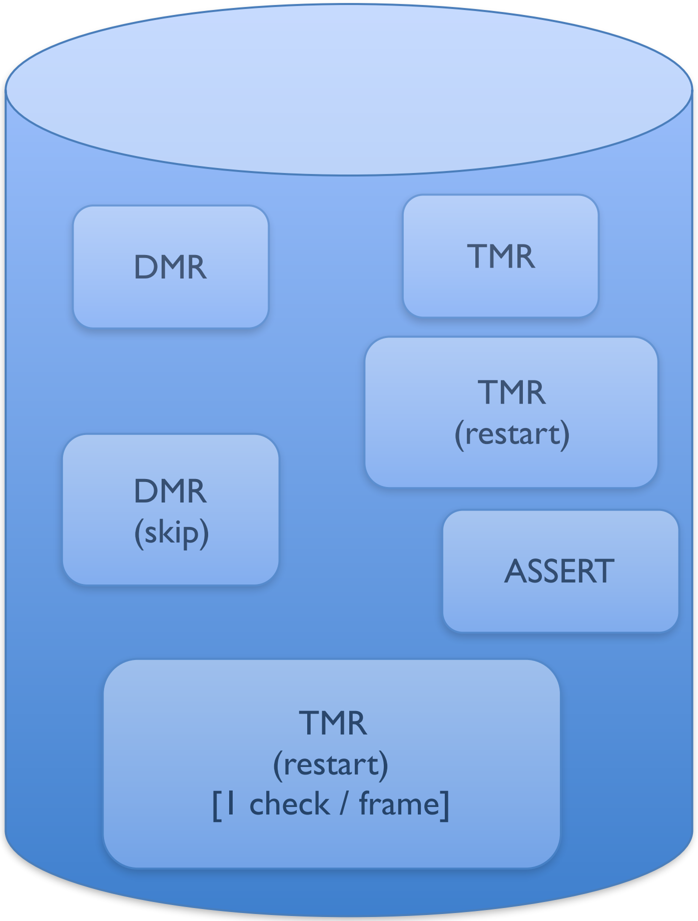
Incorporating fault-tolerance as design objective



Reliability-aware DSE

Incorporating fault-tolerance as design objective

- Detection
- Recovery
  - E.g. trade-off checkpoint overhead / restart overhead
- Design options
  - Different effects on reliability
  - Affects other objectives (like performance, power and costs)



[CODES+ISSS'12]

Security-aware DSE?

- Increasing ubiquity and connectivity of embedded systems → security!
- At this moment, security mostly an afterthought in the design process
- Security must be an objective in early DSE!
  - Security mechanisms affect other design objectives

Security-aware DSE?

- Increasing ubiquity and connectivity of embedded systems → security!
- At this moment, security mostly an afterthought in the design process
- Security must be an objective in early DSE!
  - Security mechanisms affect other design objectives

BIG CHALLENGE:
how do you quantify the level of security?