technische universität
dortmund

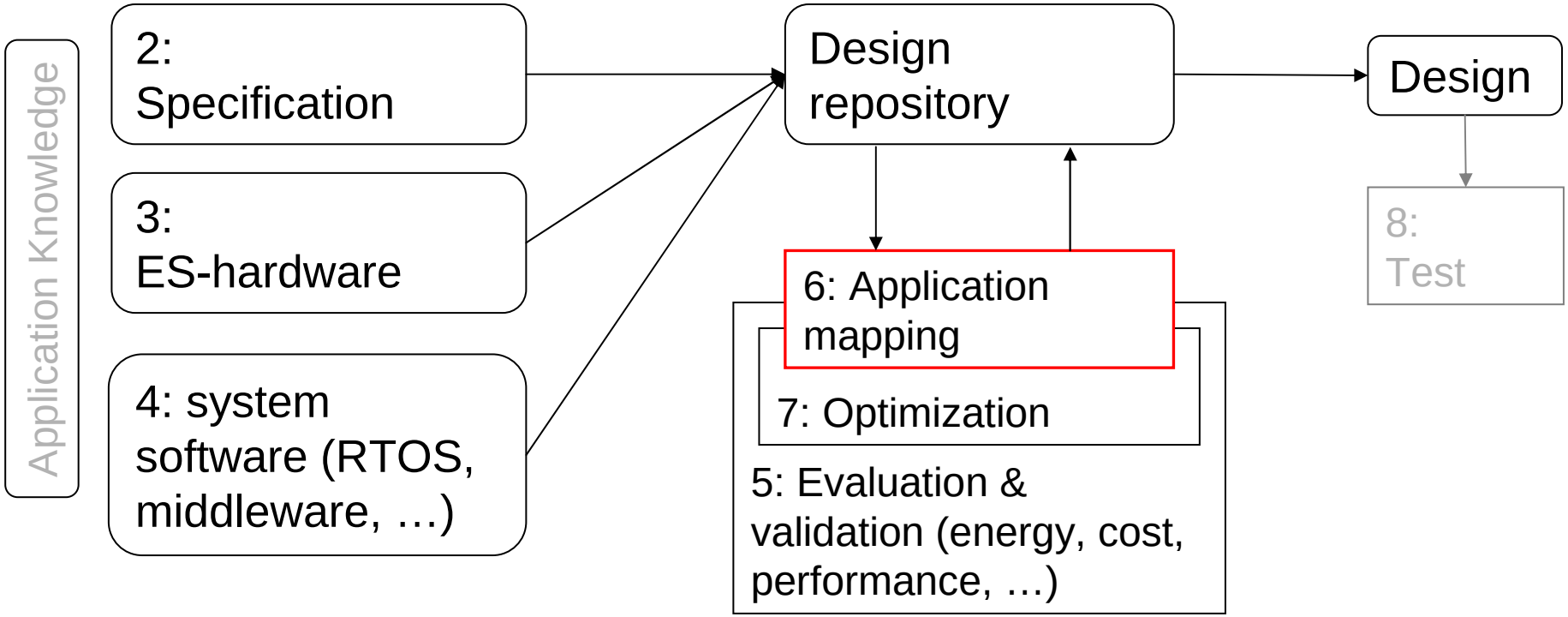
fakultät für informatik
informatik 12

Mapping of Applications to Multi-Processor Systems

Peter Marwedel
TU Dortmund, Informatik 12
Germany

2010/12/17

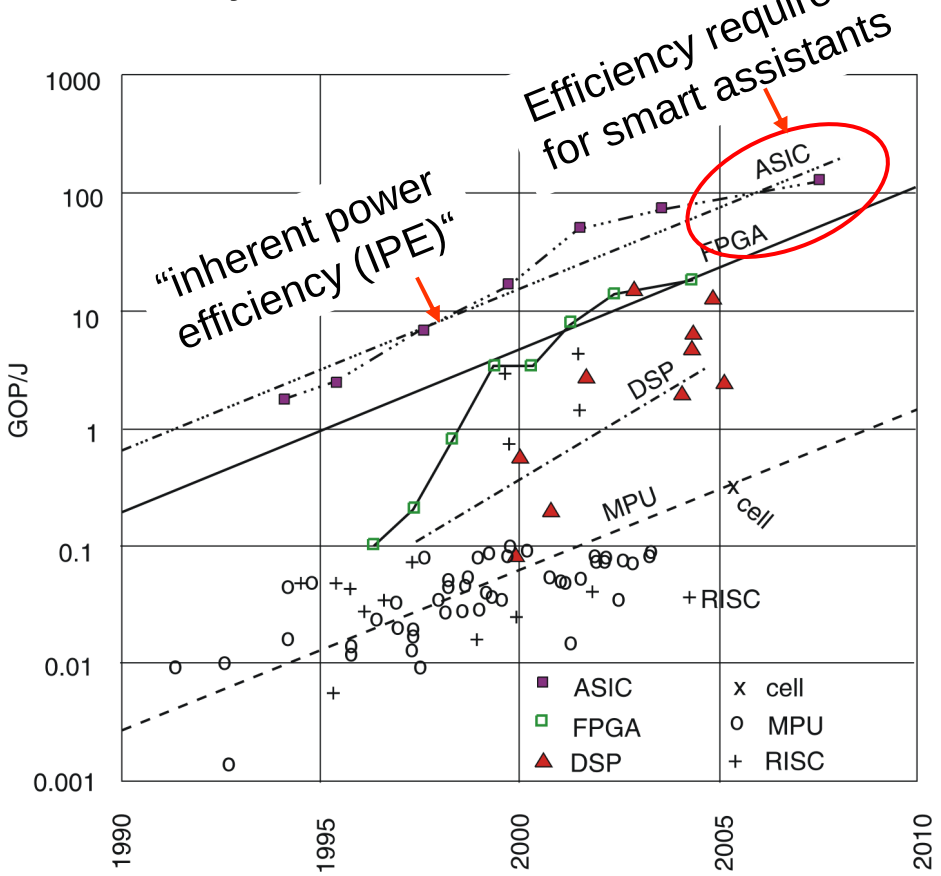
Structure of this course



Numbers denote sequence of chapters

The need to support heterogeneous architectures

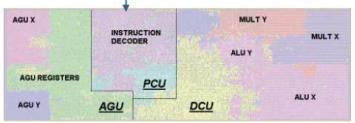
Energy efficiency a key constraint, e.g. for mobile systems



© Hugo De Man/Philips, 2007

Unconventional architectures close to IPE

Retargetable C compiler

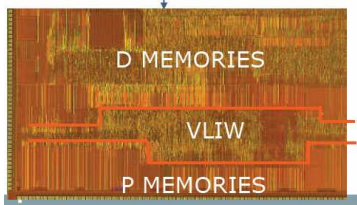


Courtesy: Philips-Target Compilers

Coolflux Audio ASIP

130 nm 0.9V 0.32mm² 24bit
2.0 mW MP3 incl. SRAMs
42 MOPS/mW (~1/4 IPE)

Retargetable C compiler

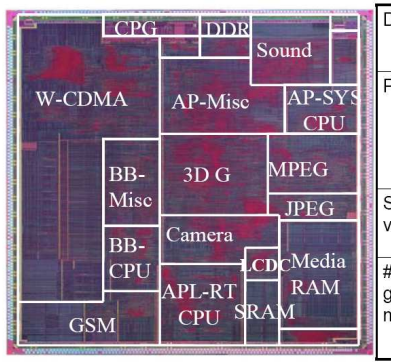


Courtesy: SiliconHive

41 Issue VLIW for SDR

130 nm 1.2V 6.5mm² 16 bit
30 operations / cycle (OFDM)
150 MHz 190mW (incl SRAMs)
24 GOPS/W (~ 1/5 IPE)

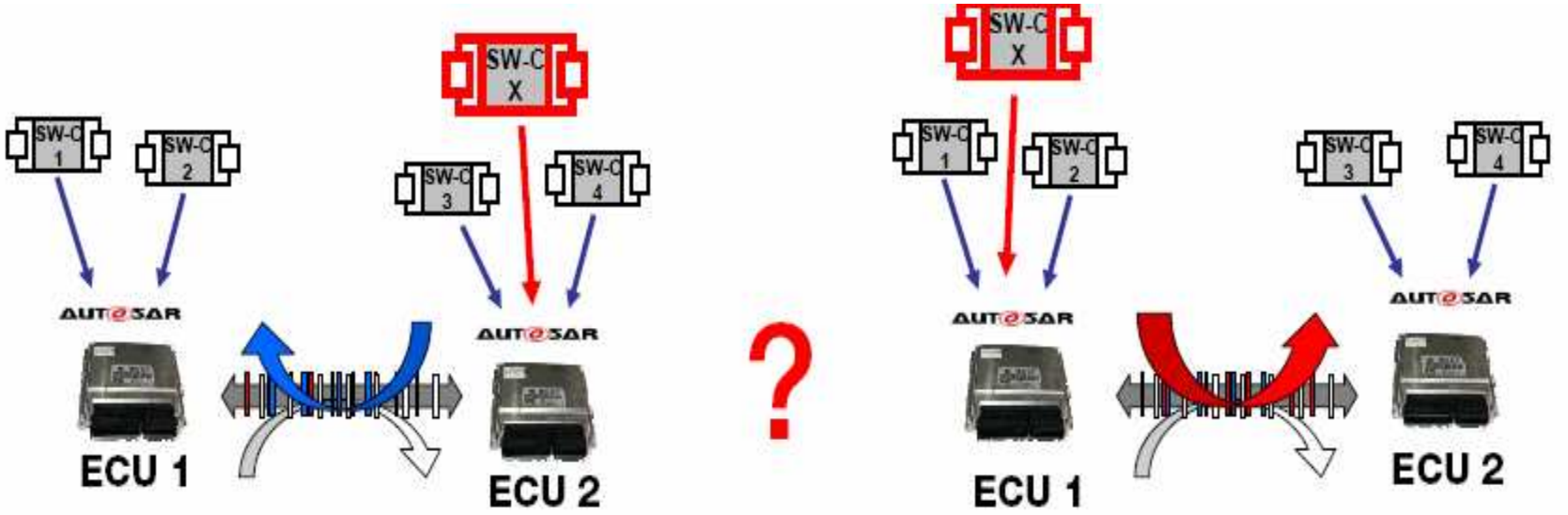
SH-MobileG1: Chip Overview



© Renesas, MPSoC'07

☞ How to map to these architectures ?

Practical problem in automotive design



- Evaluate alternatives („what if ?“)
  - Mapping
  - Scheduling
  - Communication

- **Early**
- **Quickly**
- **Cost-efficient**

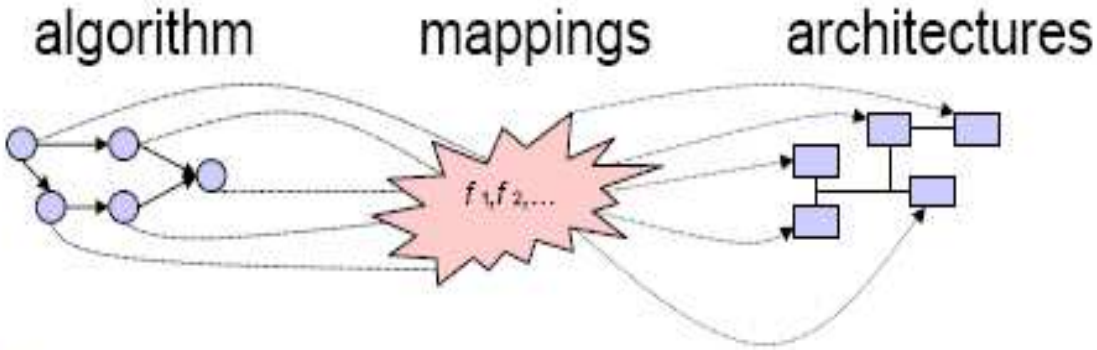
Which processor should run the software?

A Simple Classification

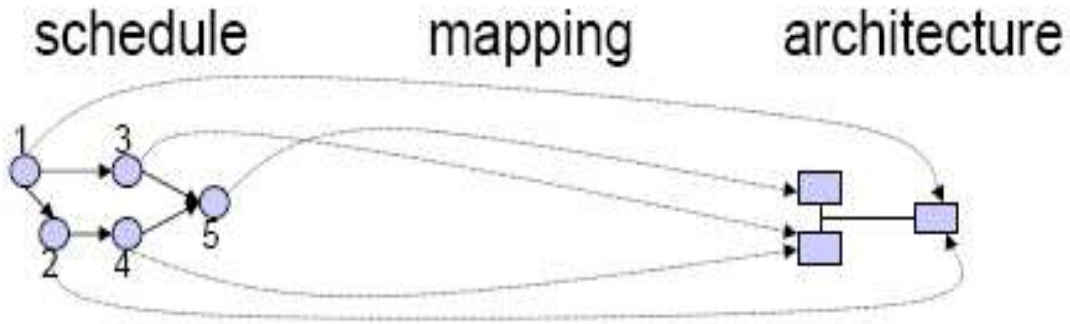
Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given task graph	Map to CELL, Hopes, Qiang XU (HK) Simunic (UCSD)	COOL codesign tool; EXPO/SPEA2 SystemCodesigner
Auto-parallelizing	Mnemee (Dortmund) Franke (Edinburgh) MAPS	Daedalus

Example: System Synthesis

Given:



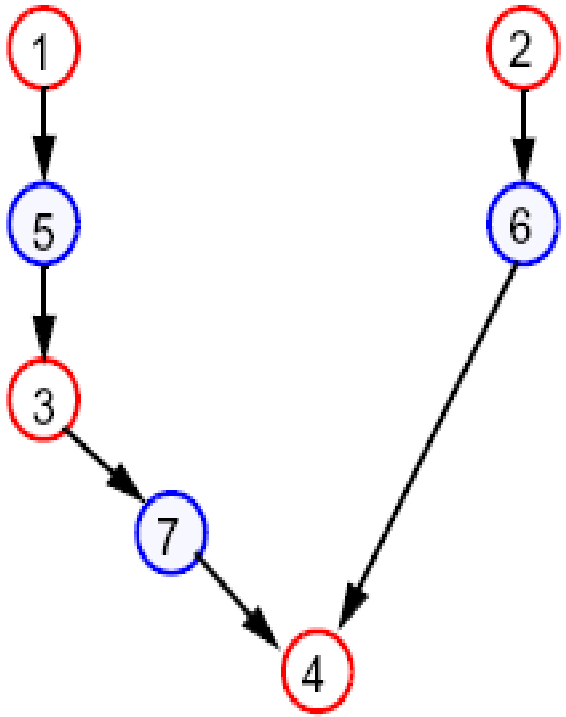
Goal:



Objectives: cost, latency, power consumption

Basic Model – Problem Graph

Problem graph $G_P(V_P,E_P)$:

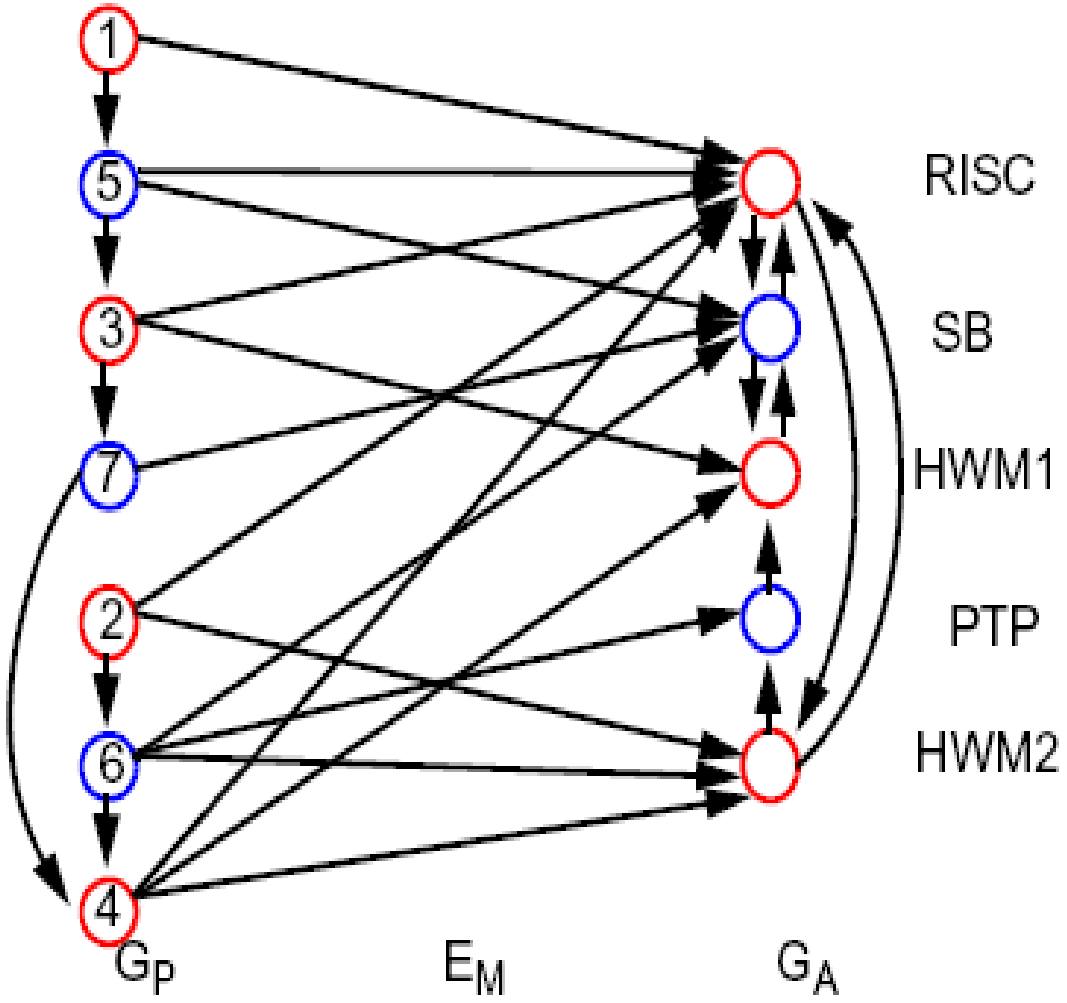


Interpretation:

- V_P consists of functional nodes V_P^f (task, procedure) and communication nodes V_P^c .
- E_P represent data dependencies

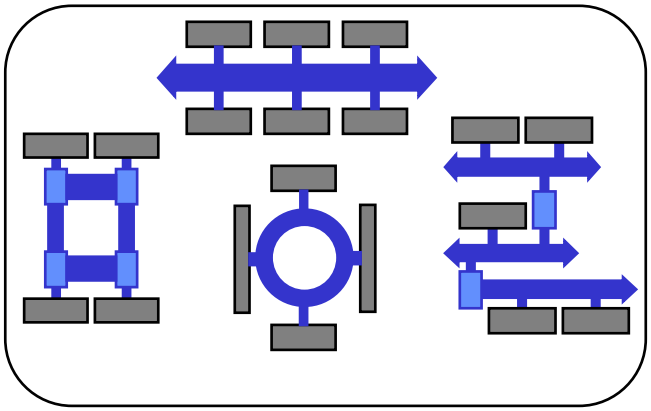
Basic Model: Specification Graph

Definition: A specification graph is a graph $G_S=(V_S,E_S)$ consisting of a problem graph G_P, an architecture graph G_A, and edges E_M. In particular, $V_S=V_P \cup V_A$, $E_S=E_P \cup E_A \cup E_M$

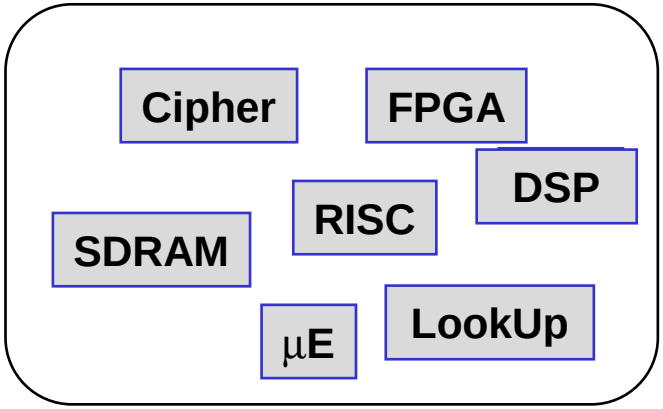


Design Space

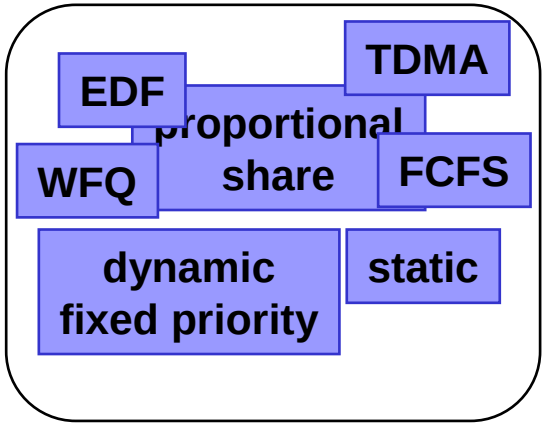
Communication Templates



Computation Templates

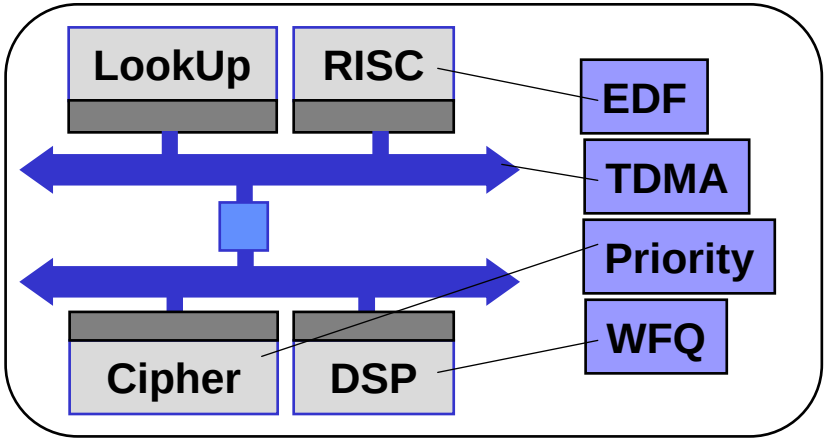


Scheduling/Arbitration

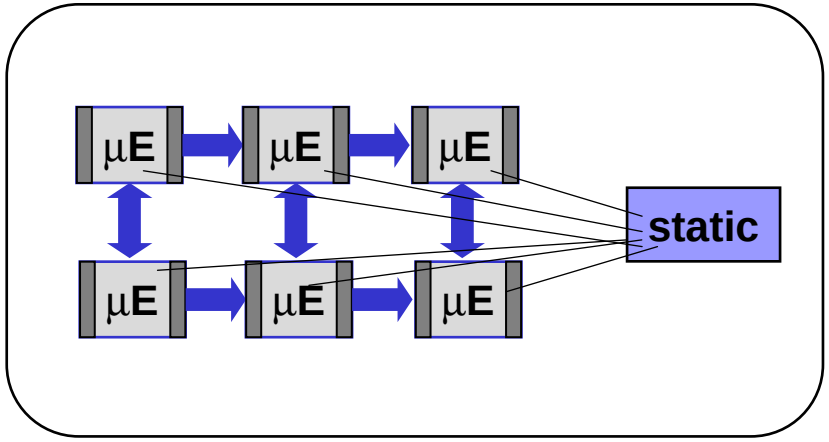


Which architecture is better suited for our application?

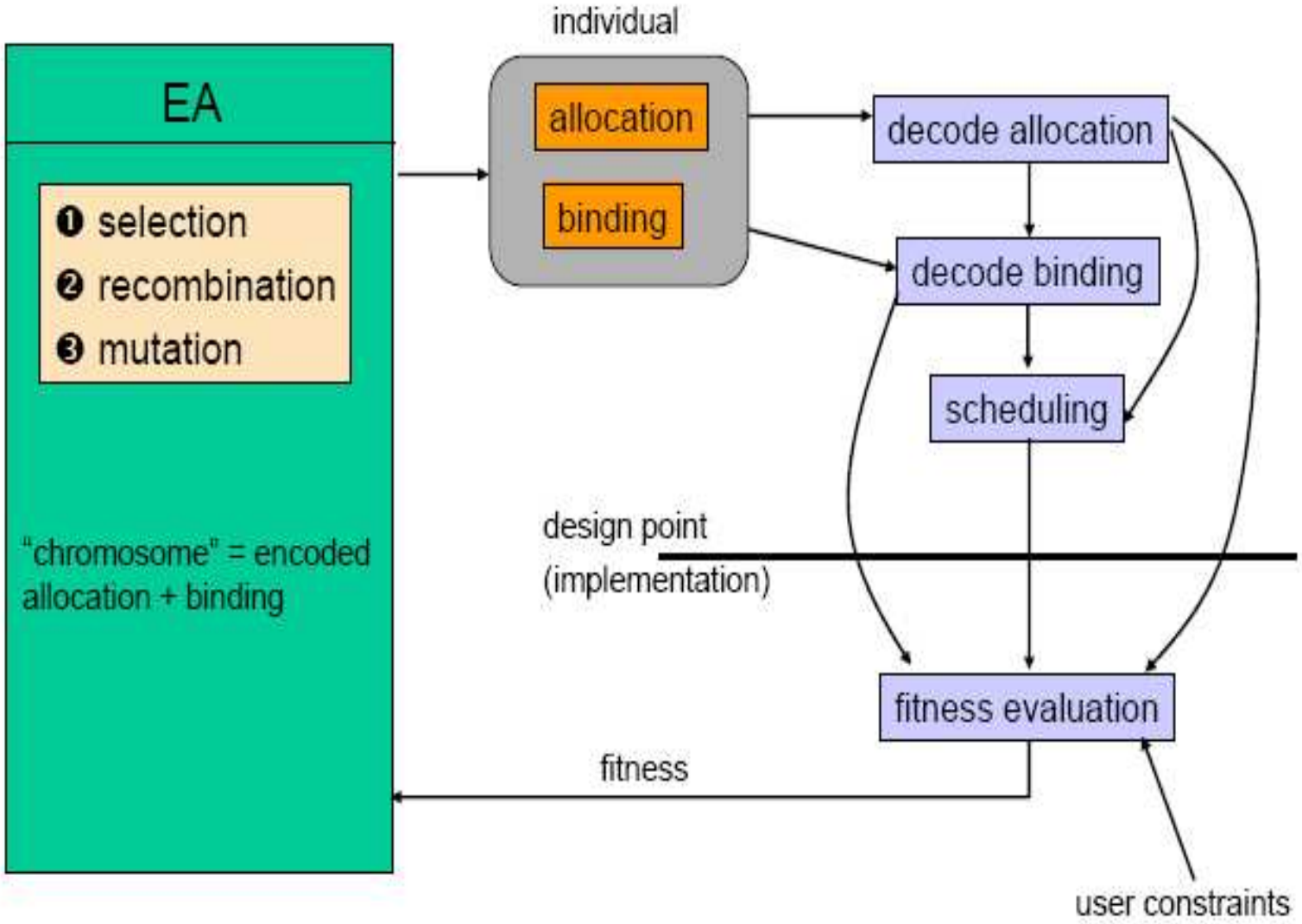
Architecture # 1



Architecture # 2



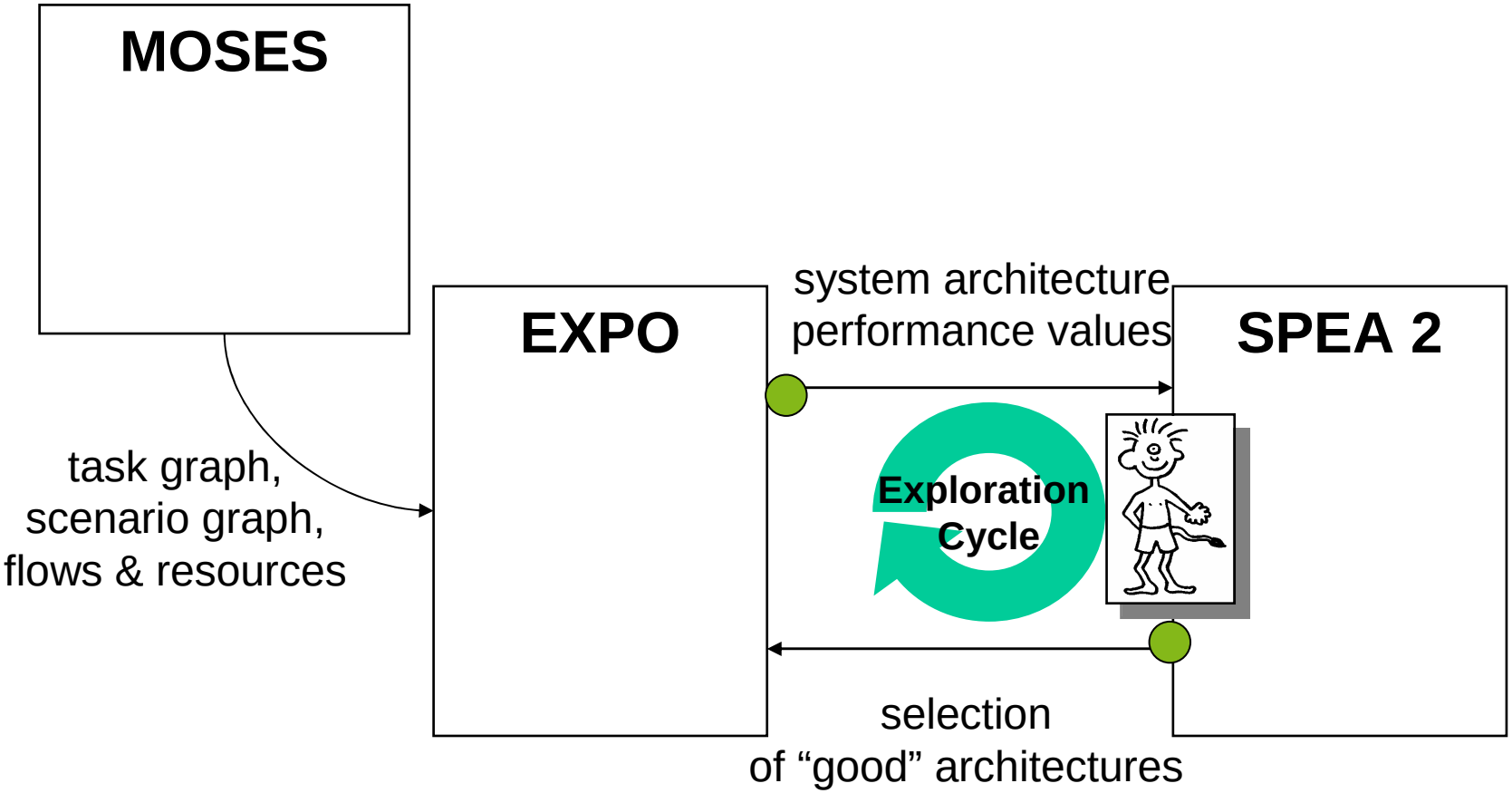
Evolutionary Algorithms for Design Space Exploration (DSE)



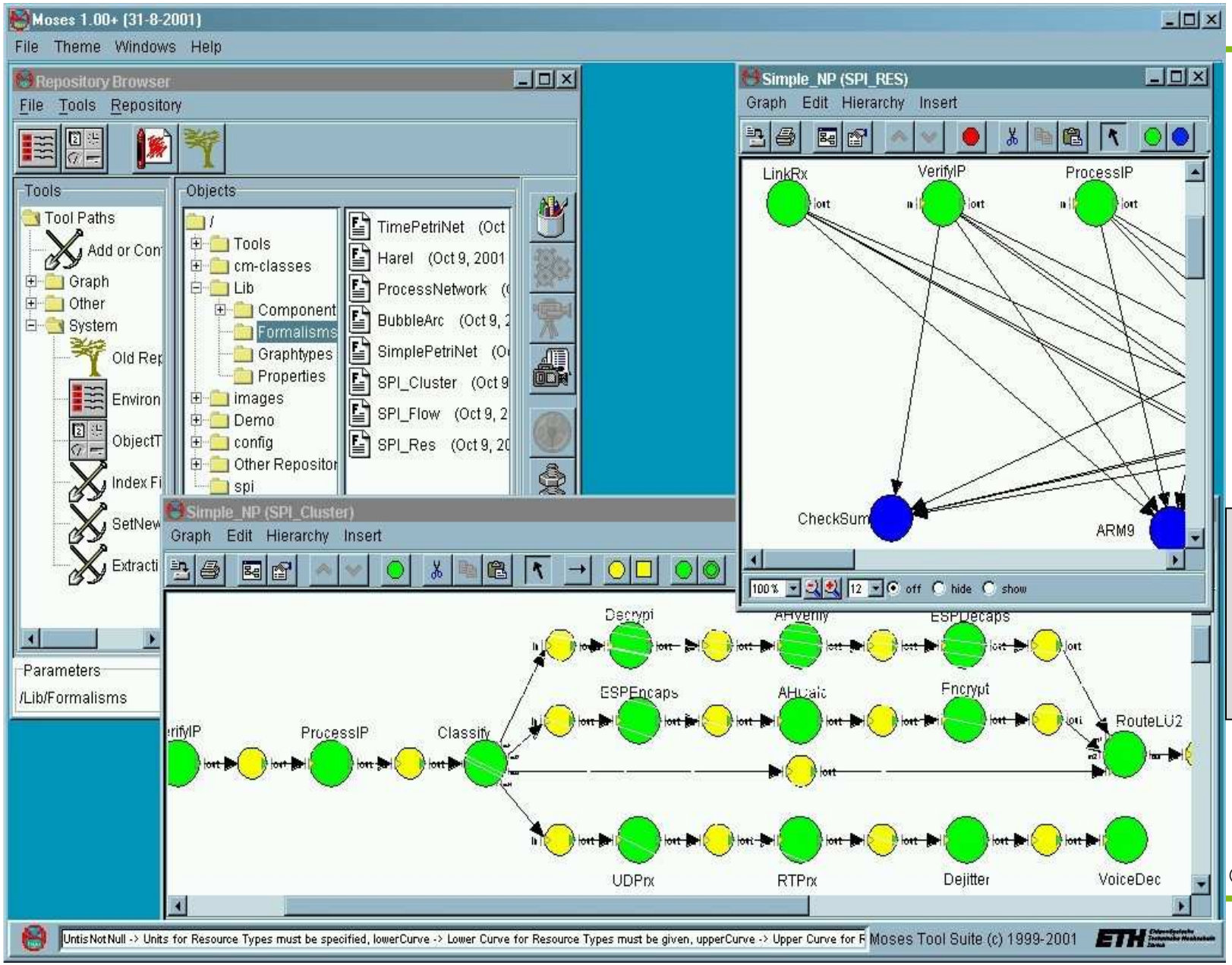
Challenges

- Encoding of (allocation+binding)
  - simple encoding
    - eg. one bit per resource, one variable per binding
    - easy to implement
    - many infeasible partitionings
  - encoding + repair
    - eg. simple encoding and modify such that for each $v_p \in V_P$ there exists at least one $v_a \in V_A$ with a $\beta(v_p) = v_a$
    - reduces number of infeasible partitionings
- Generation of the initial population, mutation
- Recombination

EXPO – Tool architecture (1)

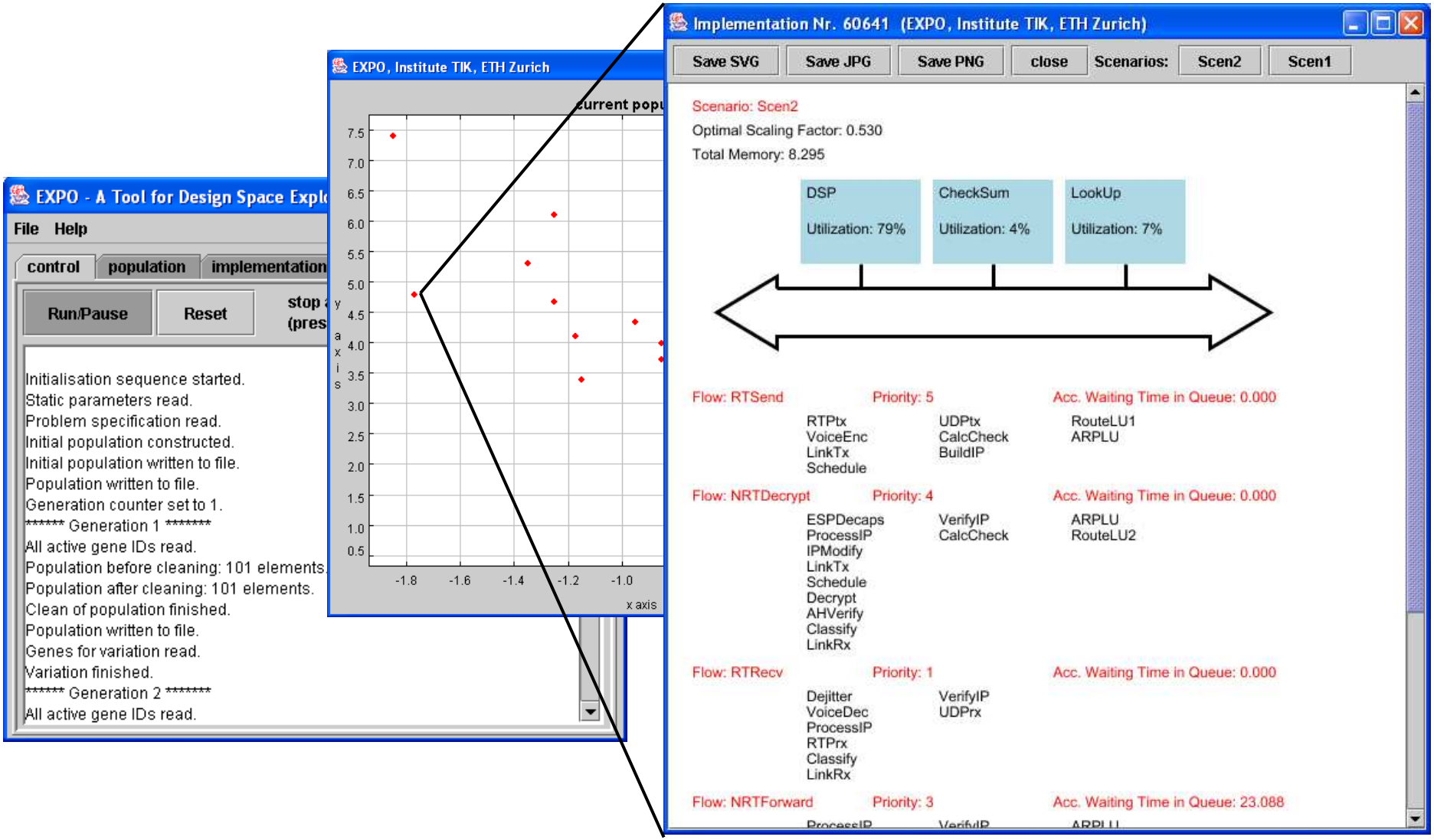


EXPO – Tool architecture (2)



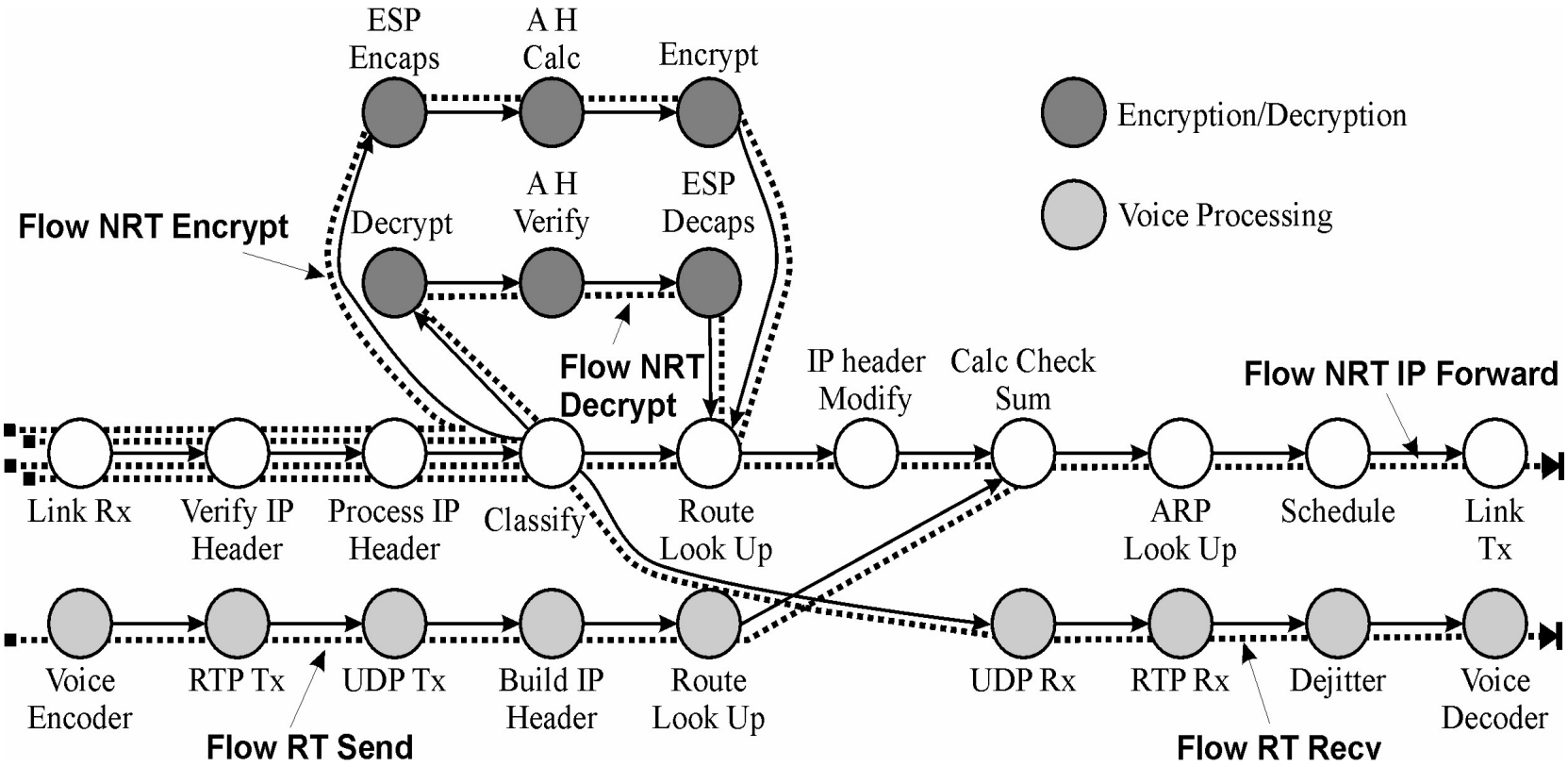
Tool available online: http://www.tik.ee.ethz.ch/expo/expo.html

EXPO – Tool (3)


EXPO - A Tool for Design Space Expl
File Help
control population implementation
Run/Pause Reset
Initialisation sequence started.
Static parameters read.
Problem specification read.
Initial population constructed.
Initial population written to file.
Population written to file.
Generation counter set to 1.
****** Generation 1 *******
All active gene IDs read.
Population before cleaning: 101 elements.
Population after cleaning: 101 elements.
Clean of population finished.
Population written to file.
Genes for variation read.
Variation finished.
****** Generation 2 *******
All active gene IDs read.
EXPO, Institute TIK, ETH Zurich
x axis
Implementation Nr. 60641 (EXPO, Institute TIK, ETH Zurich)
Save SVG Save JPG Save PNG close Scenarios: Scen2 Scen1
Scenario: Scen2
Optimal Scaling Factor: 0.530
Total Memory: 8.295
DSP
Utilization: 79%
CheckSum
Utilization: 4%
LookUp
Utilization: 7%
Flow: RTSend Priority: 5 Acc. Waiting Time in Queue: 0.000
RTPtx VoiceEnc LinkTx Schedule
UDPtx CalcCheck BuildIP
RouteLU1 ARPLU
Flow: NRTDecrypt Priority: 4 Acc. Waiting Time in Queue: 0.000
ESPDecaps ProcessIP IPModify LinkTx Schedule Decrypt AHVerify Classify LinkRx
VerifyIP CalcCheck
ARPLU RouteLU2
Flow: RTRecv Priority: 1 Acc. Waiting Time in Queue: 0.000
Dejitter VoiceDec ProcessIP RTPrx Classify LinkRx
VerifyIP UDPrx
Flow: NRTForward Priority: 3 Acc. Waiting Time in Queue: 23.088

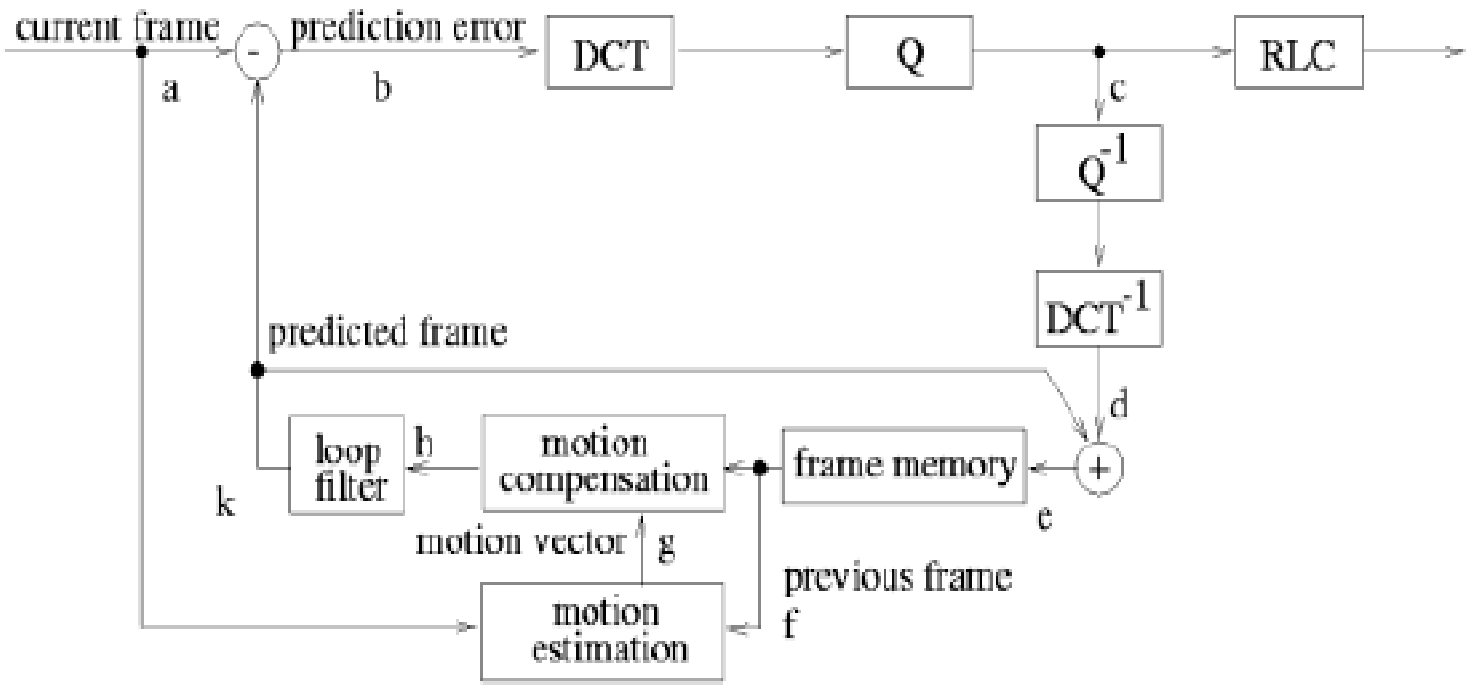
Application Model

Example of a simple stream processing task structure:



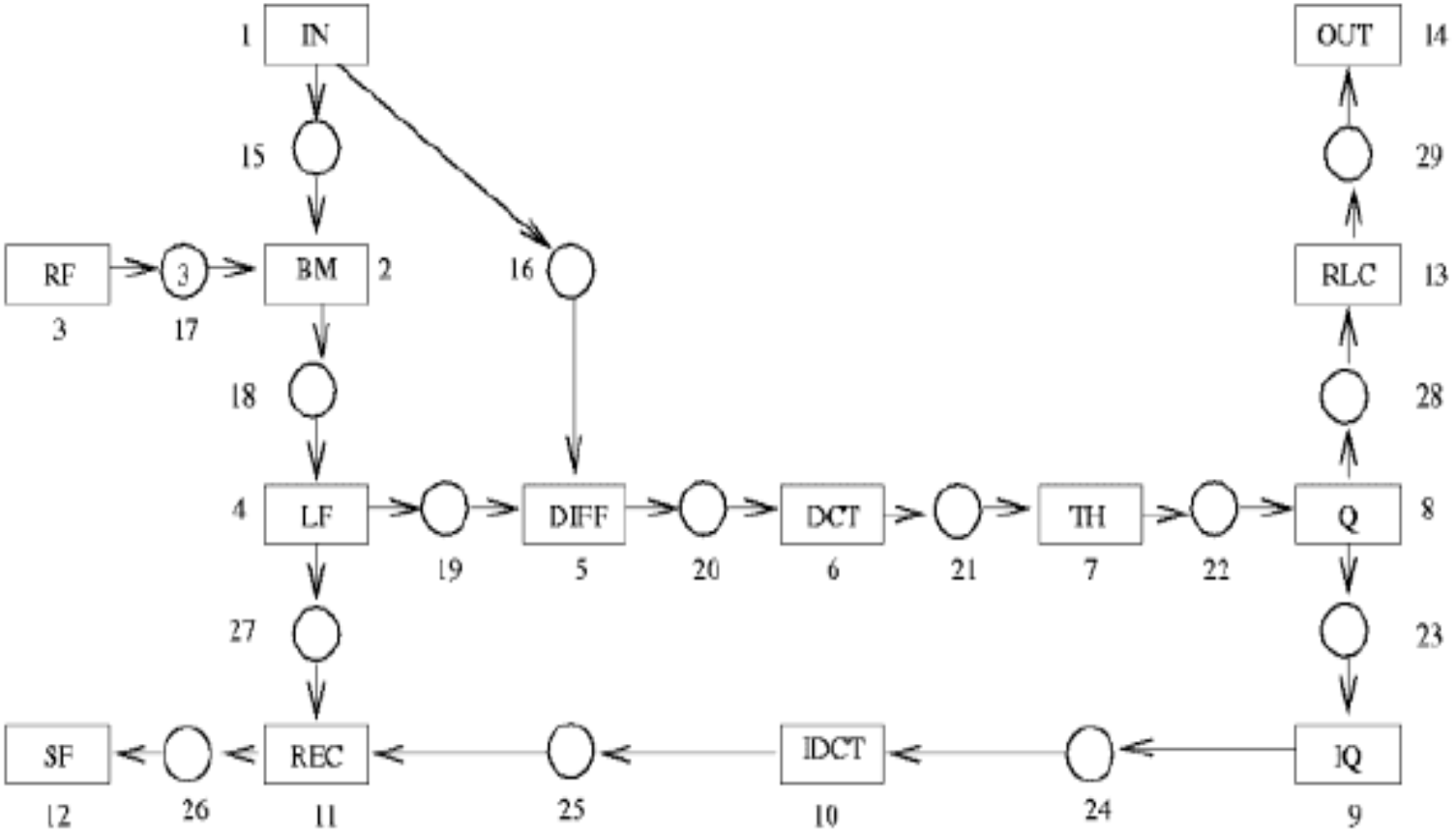
Exploration – Case Study (1)

behavioral specification of a video codec for video compression

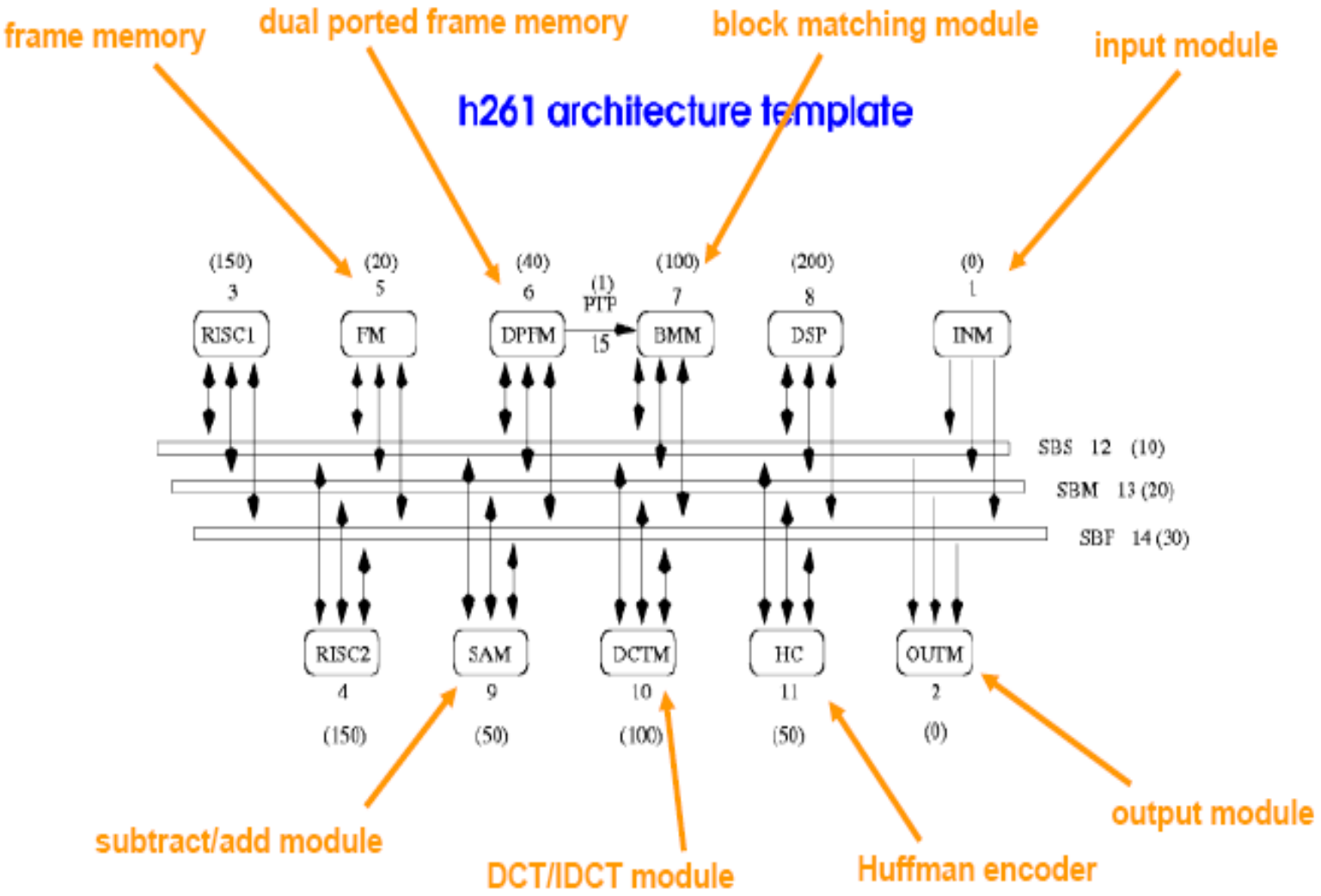


Exploration – Case Study (2)

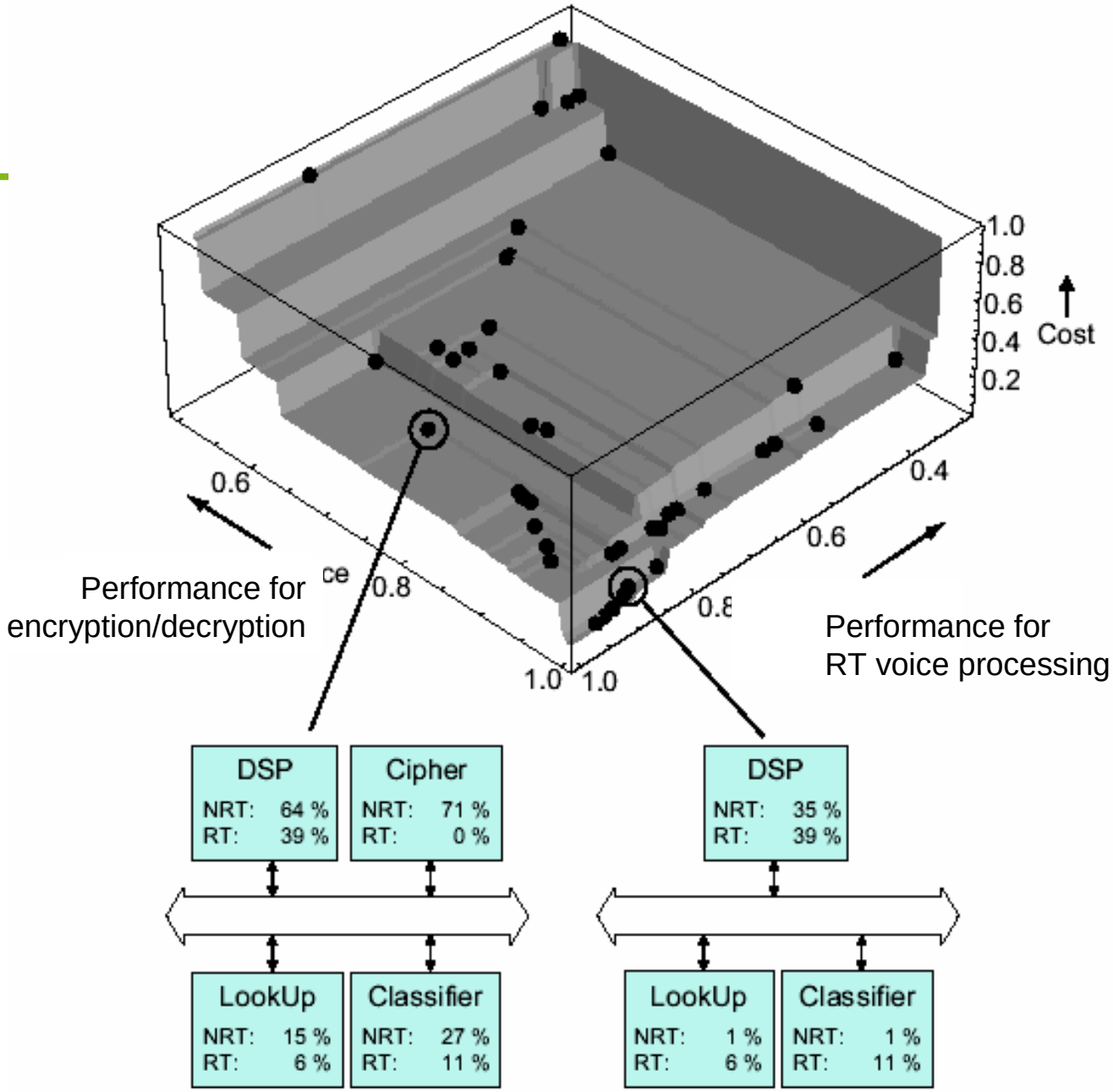
problem graph of the video coder



Exploration – Case Study (3)

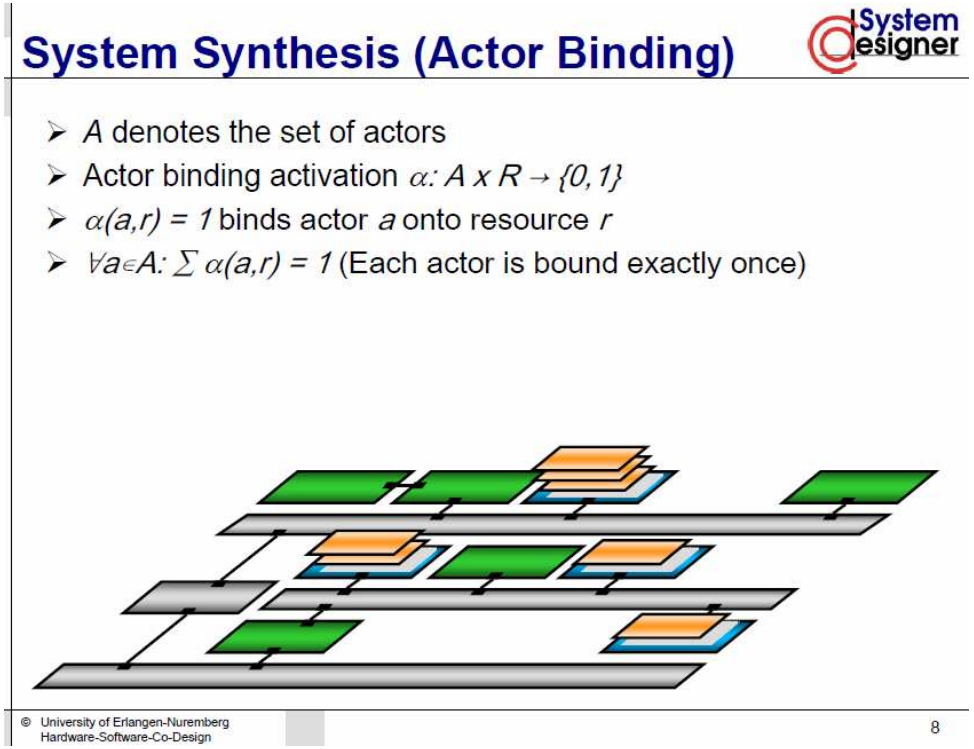


More Results



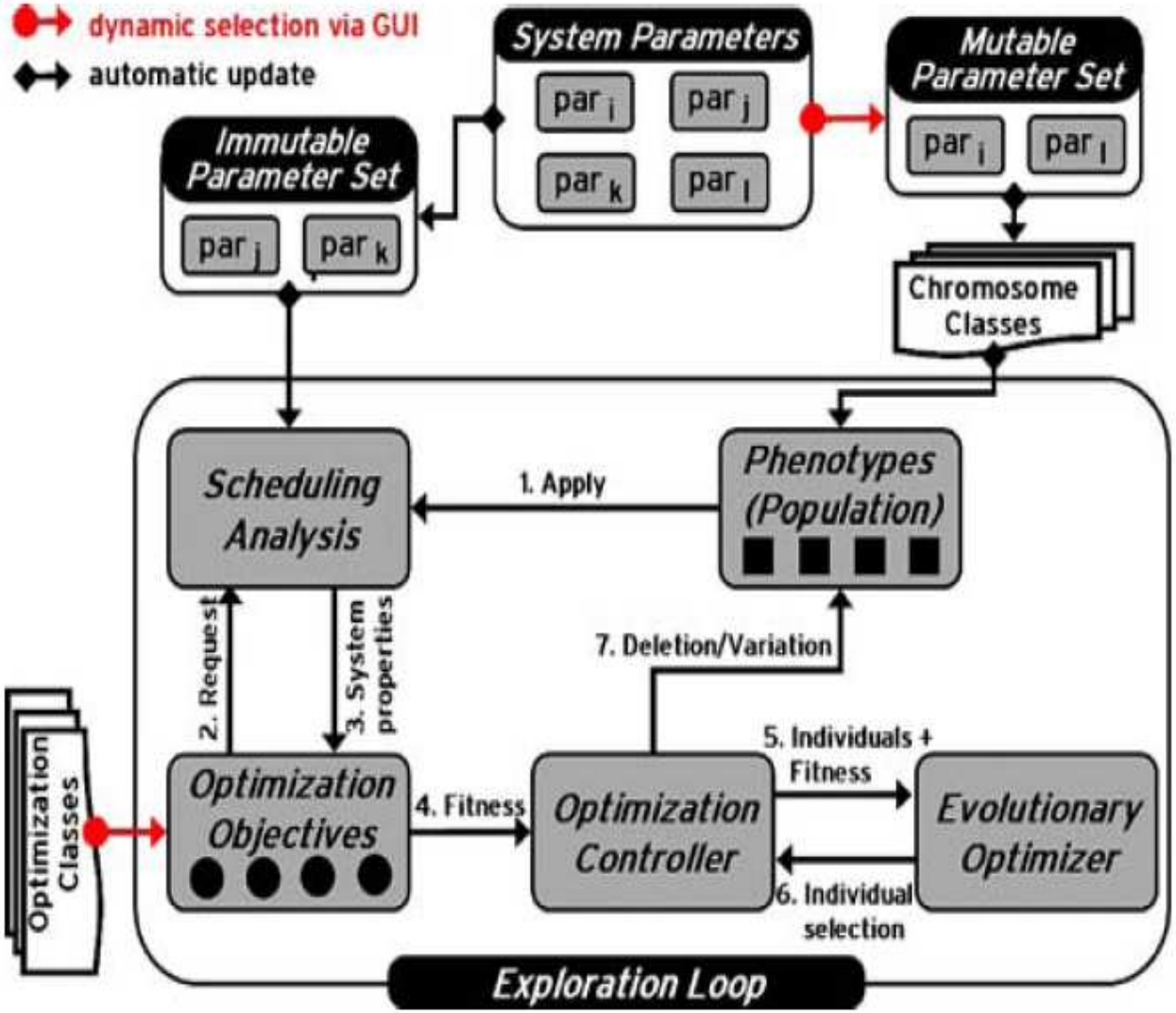
Design Space Exploration with SystemCoDesigner
(Teich, Erlangen)

- System Synthesis comprises:
  - Resource allocation
  - Actor binding
  - Channel mapping
  - Transaction modeling
- Idea:
  - Formulate synthesis problem as 0-1 ILP
  - Use Pseudo-Boolean (PB) solver to find feasible solution
  - Use multi-objective Evolutionary algorithm (MOEA) to optimize Decision Strategy of the PB solver



© J. Teich, U. Erlangen-Nürnberg

A 3rd approach based on evolutionary algorithms: SYMTA/S:

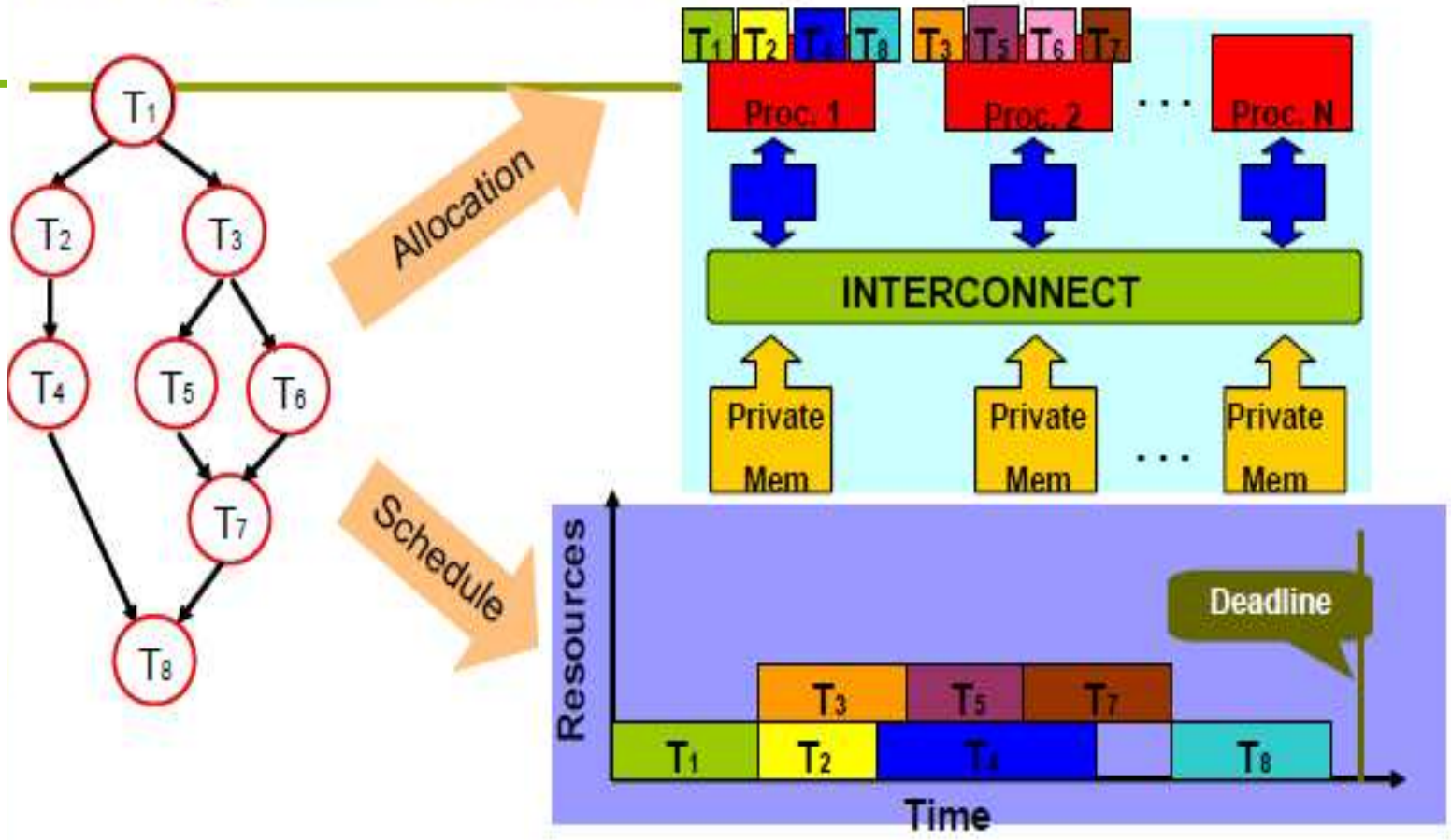


[R. Ernst et al.: A framework for modular analysis and exploration of heteterogenous embedded systems, *Real-time Systems*, 2006, p. 124]

A Simple Classification

Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given task graph	Map to CELL, Hopes Qiang XU (HK) Simunic (UCSD)	COOL codesign tool; EXPO/SPEA2 SystemCodesigner
Auto-parallelizing	Mnemee (Dortmund) Franke (Edinburgh) MAPS	Daedalus

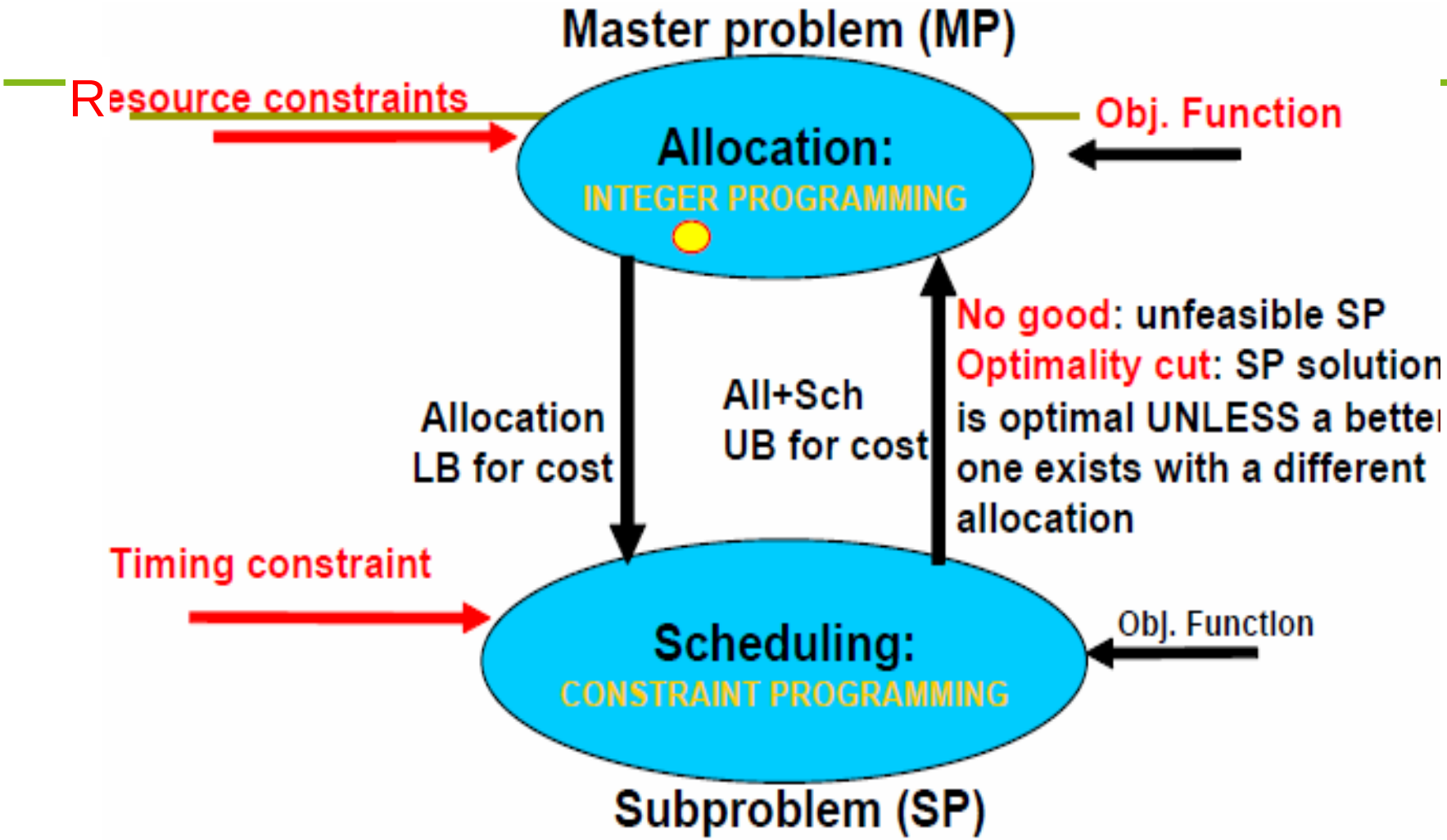
A fixed architecture approach: Map → CELL



- The problem of allocating and scheduling task graphs on processors in a distributed real-time system is NP-hard.

Martino Ruggiero, Luca Benini: Mapping task graphs to the CELL BE processor, *1st Workshop on Mapping of Applications to MPSoCs, Rheinfels Castle*, 2008

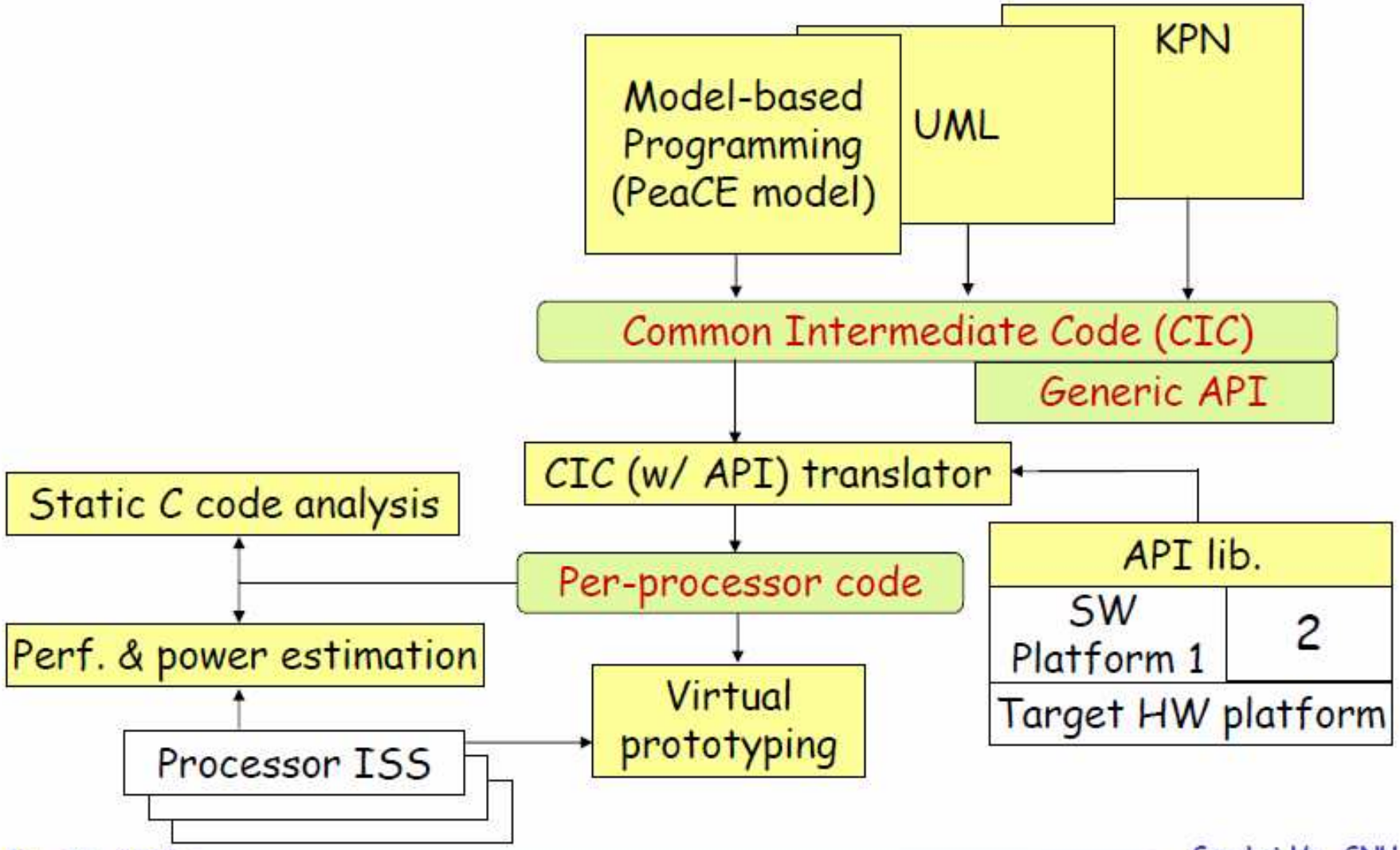
Partitioning into Allocation and Scheduling



Iterations stop when MP becomes unfeasible!

HOPES Proposal

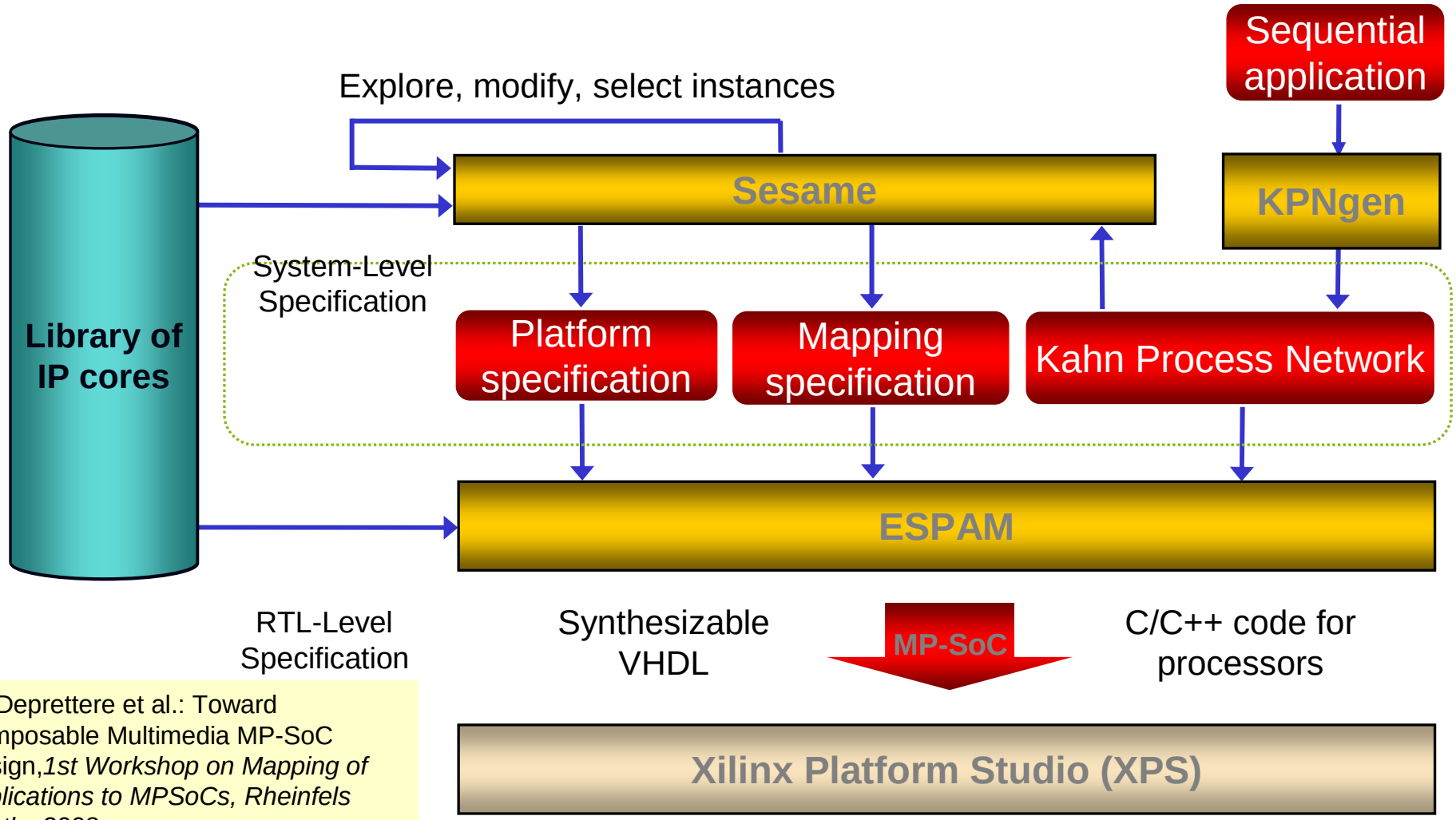
HOPES



A Simple Classification

Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given task graph	Map to CELL, Hopes, Qiang XU (HK) Simunic (UCSD)	COOL codesign tool; EXPO/SPEA2 SystemCodesigner
Auto-parallelizing	Mnemee (Dortmund) Franke (Edinburgh) MAPS	Daedalus

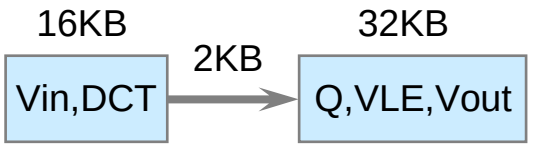
Daedalus Design-flow



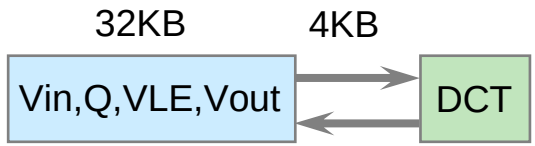
Ed Deprettere et al.: Toward Composable Multimedia MP-SoC Design,*1st Workshop on Mapping of Applications to MPSoCs, Rheinfels Castle*, 2008

JPEG/JPEG2000 case study

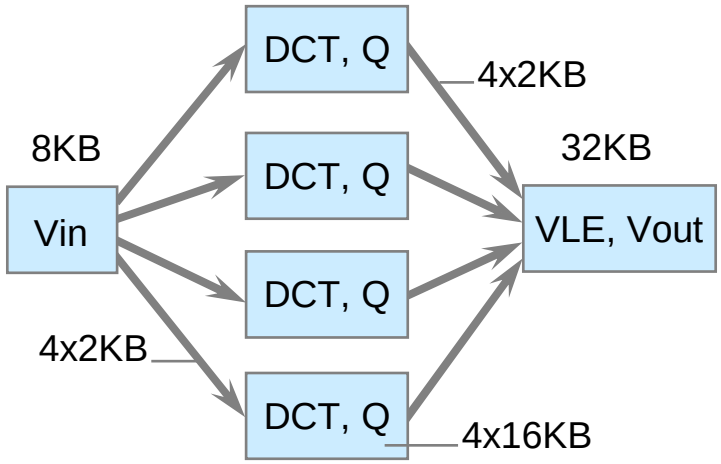
Example architecture instances for a single-tile JPEG encoder:



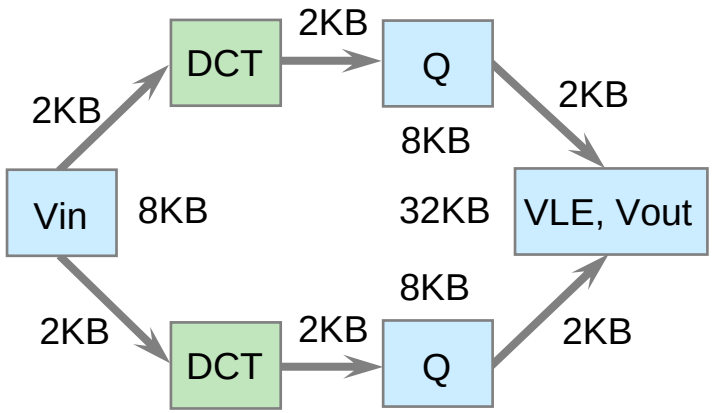
2 MicroBlaze processors (50KB)



1 MicroBlaze, 1HW DCT (36KB)

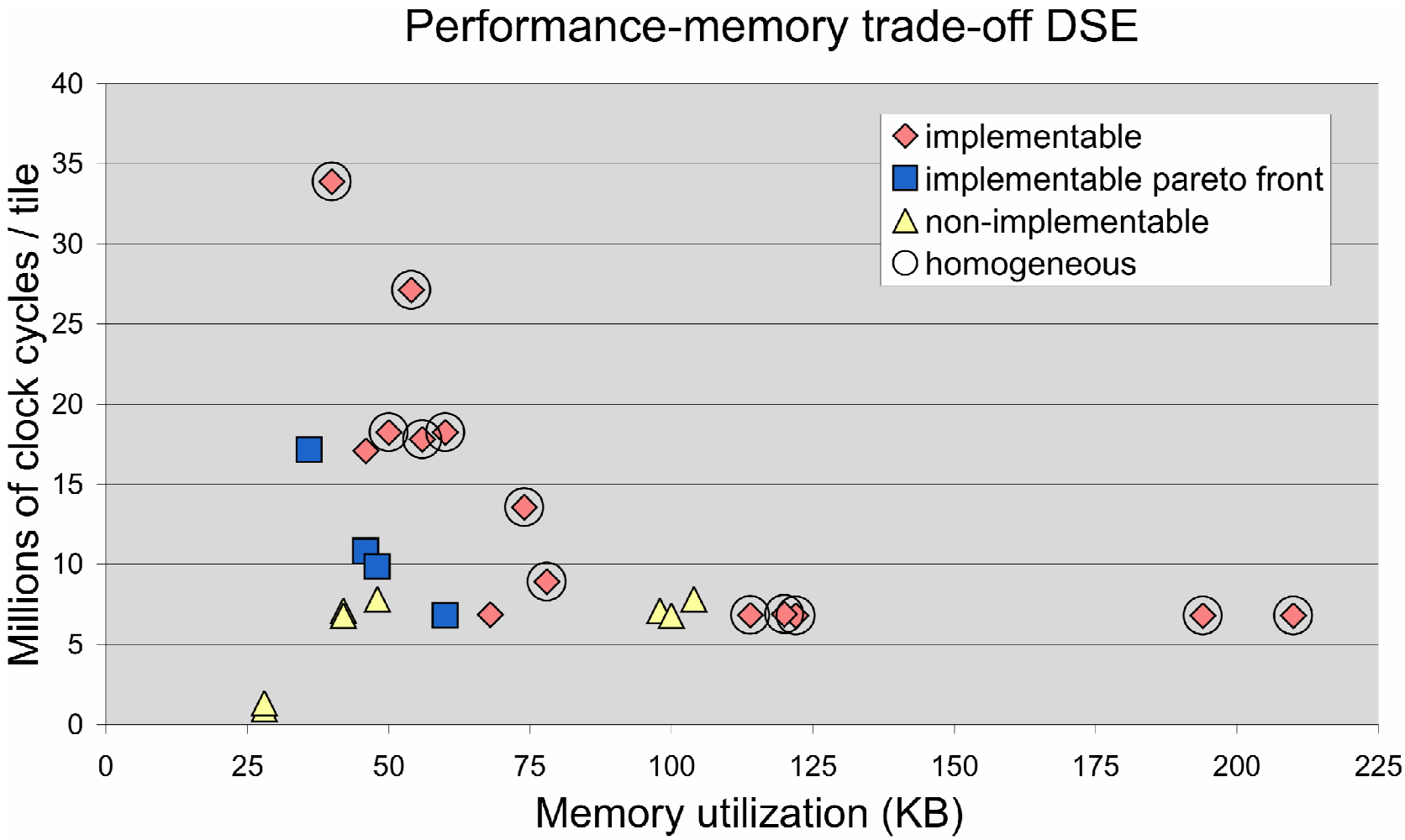


6 MicroBlaze processors (120KB)



4 MicroBlaze, 2HW DCT (68KB)

Sesame DSE results:
Single JPEG encoder DSE



A Simple Classification

Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given task graph	Map to CELL, Hopes, Qiang XU (HK) Simunic (UCSD)	COOL codesign tool; EXPO/SPEA2 SystemCodesigner
Auto-parallelizing	Mnemee (Dortmund) Franke (Edinburgh) MAPS	Daedalus

Auto-Parallelizing Compilers

Discipline “High Performance Computing”:

- Research on vectorizing compilers for more than 25 years.
- Traditionally: Fortran compilers.
- Such vectorizing compilers usually inappropriate for Multi-DSPs, since assumptions on memory model unrealistic:
  - Communication between processors via *shared memory*
  - Memory has only *one single* common *address space*

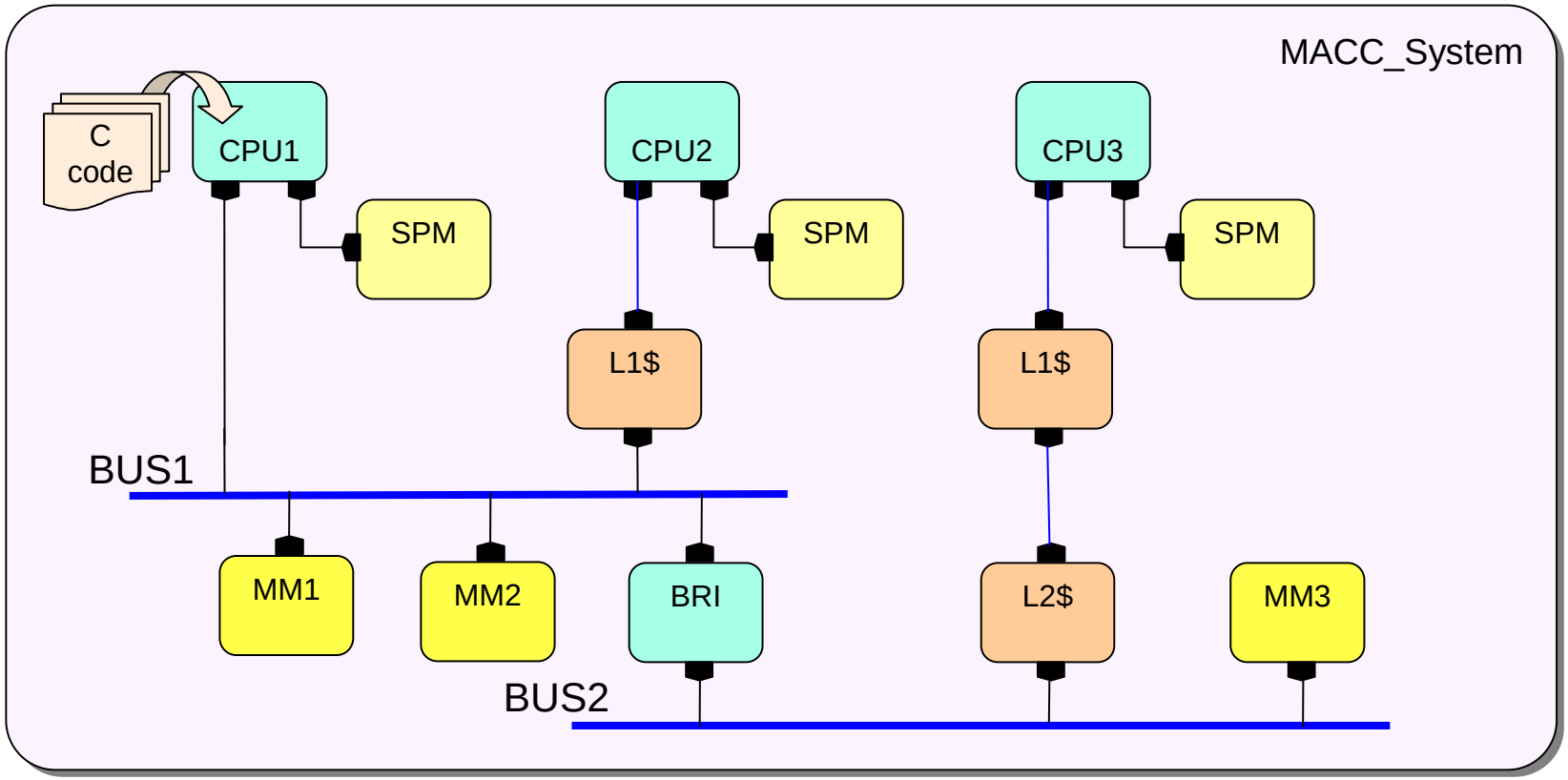
☞ ***De Facto no auto-parallelizing compiler for Multi-DSPs!***

☞ Work of Franke, O’Boyle (Edinburgh)

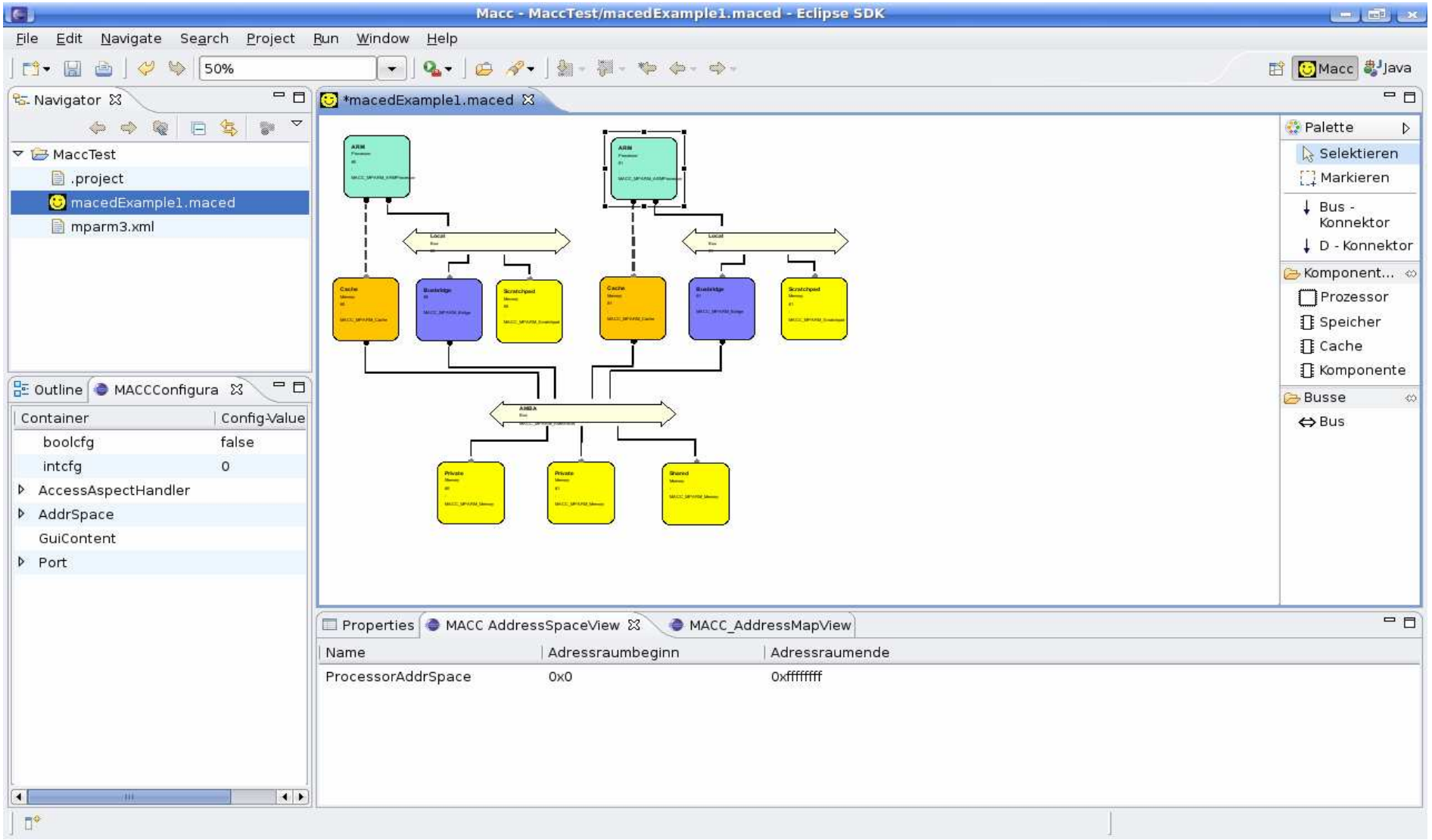
© Falk

Introduction of Memory Architecture-Aware Optimization
The MACC PMS (Processor/Memory/Switch) Model

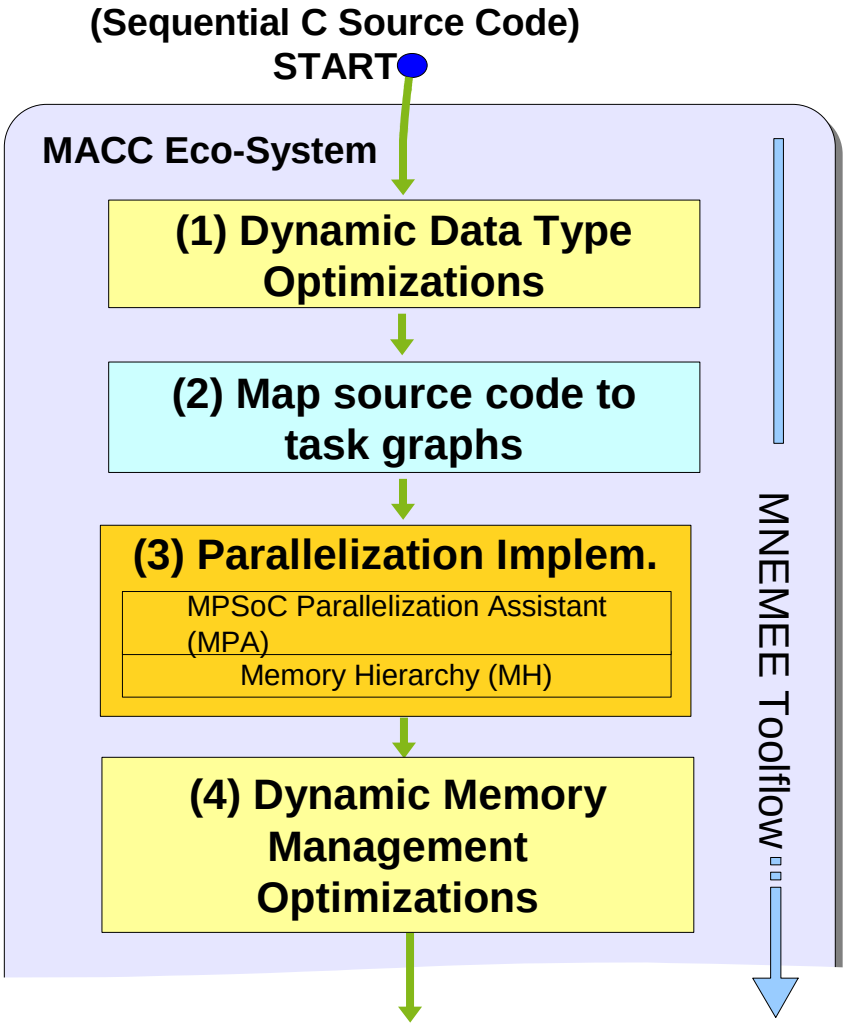
- Explicit memory architecture
- API provides access to memory information



MaCC Modeling Example via GUI

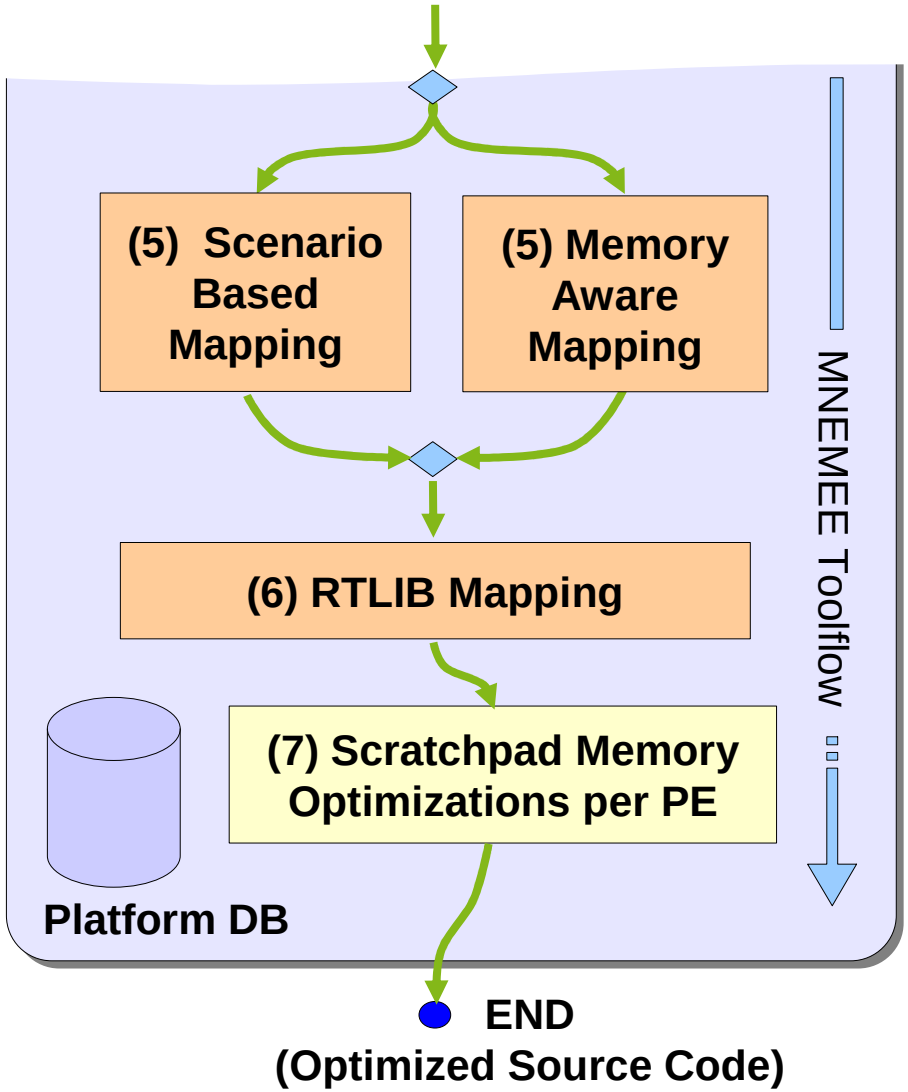


Toolflow Detailed View



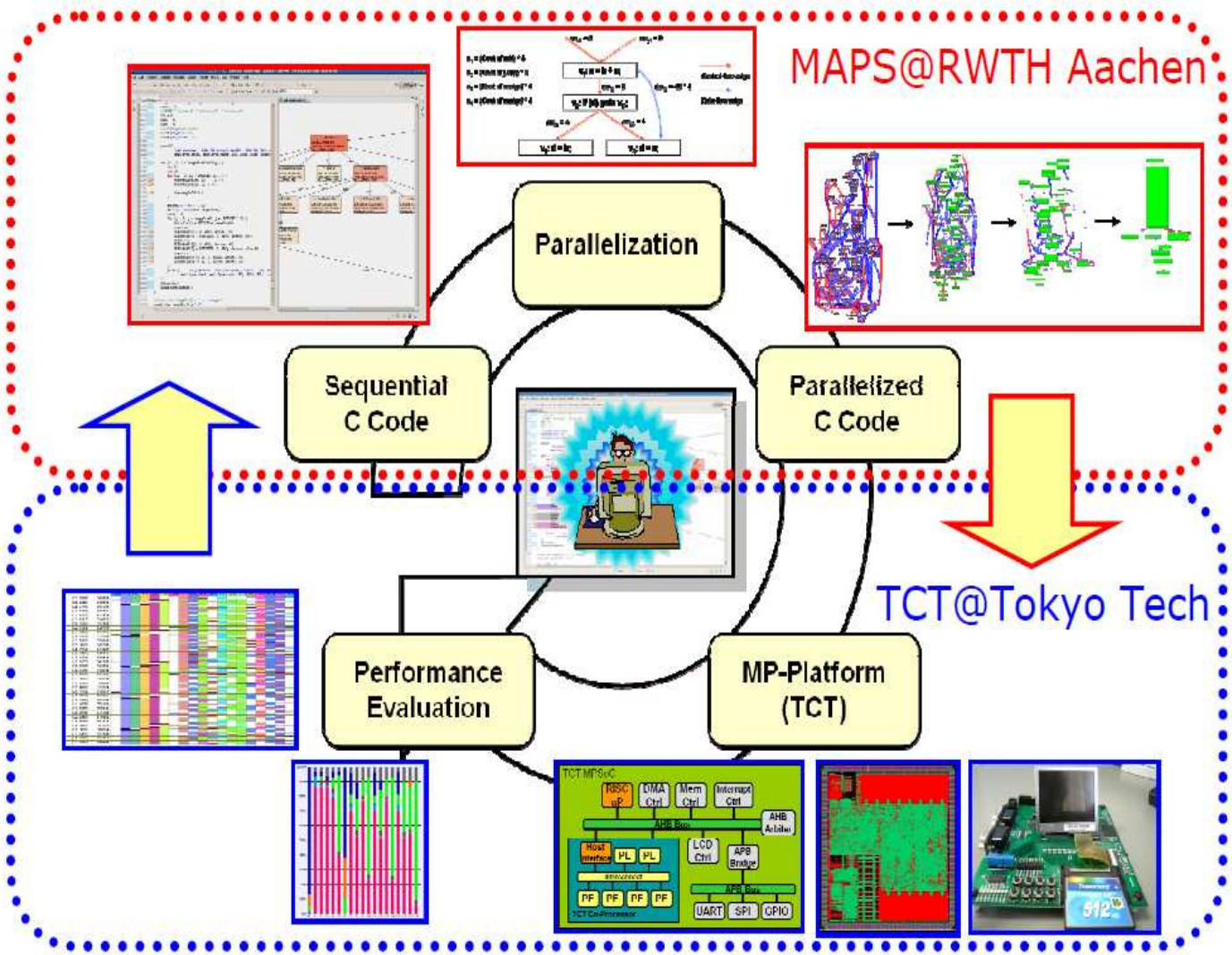
1. Optimization of dynamic data structures
2. Extraction of potential parallelism
3. Implementation of parallelism;
   placement of static data
4. Placement of dynamic data

Toolflow Detailed View



5. Perform mapping to processing elements
   - Scenario based
   - Memory aware
6. Transform the code to implement the mapping
7. Perform scratchpad memory optimizations for each processing element

MAPS-TCT Framework



© Leupers, Sheng, 2008

Rainer Leupers, Weihua Sheng: MAPS: An Integrated Framework for MPSoC Application Parallelization, *1st Workshop on Mapping of Applications to MPSoCs, Rheinfels Castle*, 2008

Summary

- Clear trend toward multi-processor systems for embedded systems, there exists a large design space
- Using architecture **crucially** depends on **mapping tools**
- Mapping applications onto heterogeneous MP systems needs allocation (if hardware is not fixed), binding of tasks to resources, scheduling
- Two criteria for classification
  - Fixed / flexible architecture
  - Auto parallelizing / non-parallelizing
- Introduction to proposed Mnemee tool chain

Evolutionary algorithms currently the best choice