

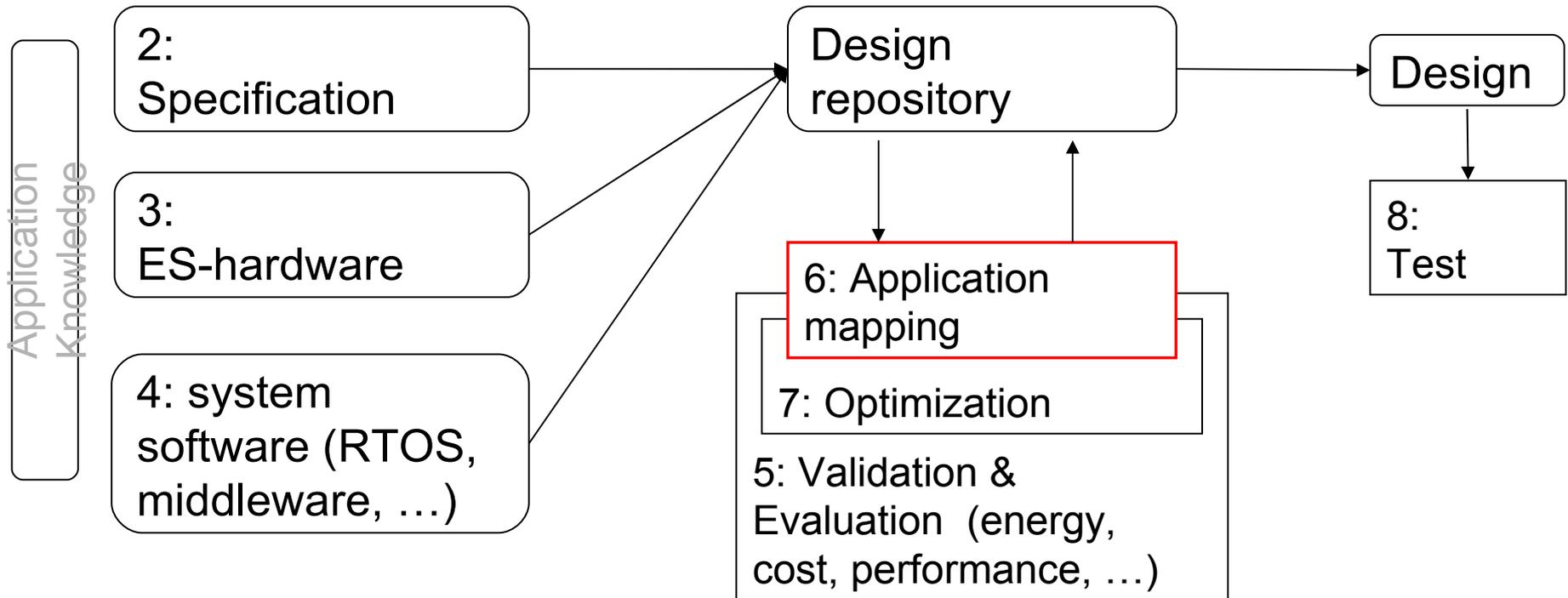
# Mapping of Applications to Multi-Processor Systems

Peter Marwedel  
Informatik 12  
TU Dortmund  
Germany

2009/12/17



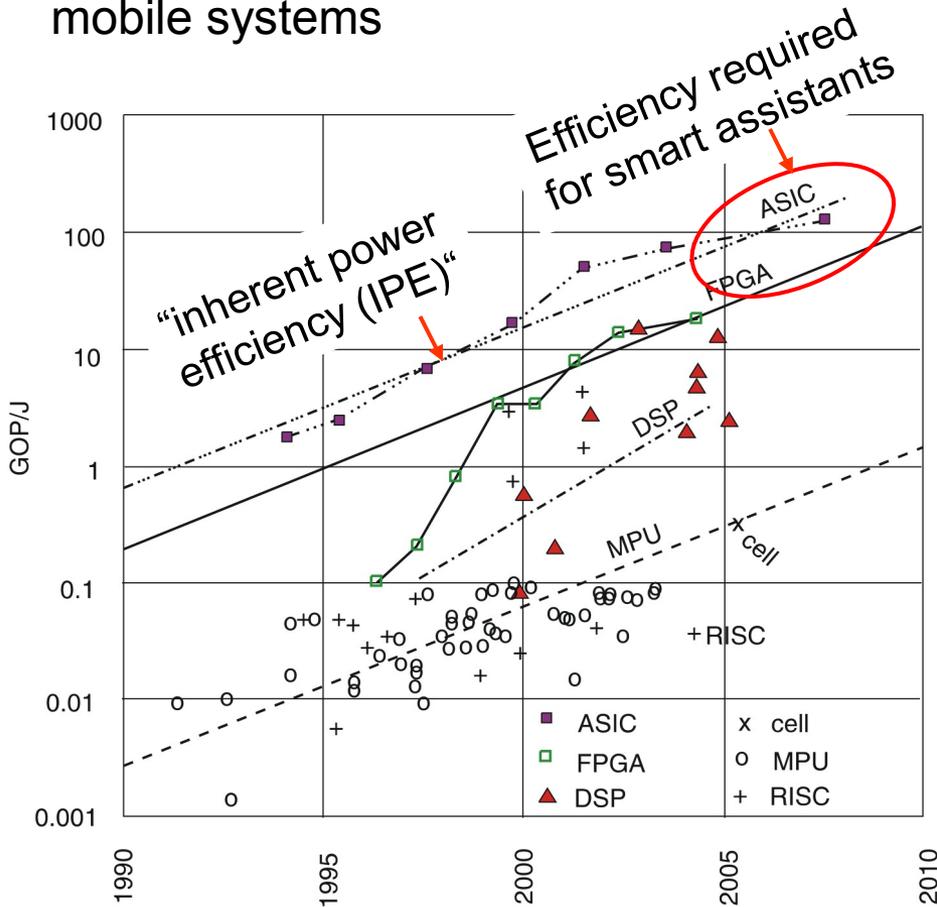
# Structure of this course



Numbers denote sequence of chapters

# The need to support heterogeneous architectures

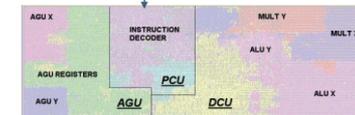
Energy efficiency a key constraint, e.g. for mobile systems



© Hugo De Man/Philips, 2007

Unconventional architectures close to IPE

Retargetable C compiler

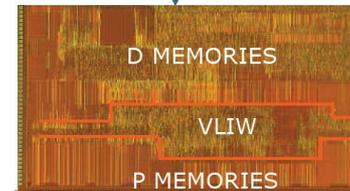


## Coolflux Audio ASIP

130 nm 0.9V 0.32mm<sup>2</sup> 24bit  
2.0 mW MP3 incl. SRAMs  
**42 MOPS/mW (~1/4 IPE)**

Courtesy: Philips-Target Compilers

Retargetable C compiler

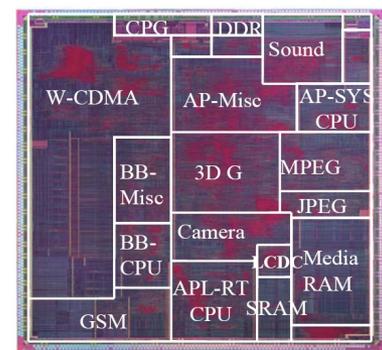


## 41 Issue VLIW for SDR

130 nm 1.2V 6.5mm<sup>2</sup> 16 bit  
30 operations / cycle (OFDM)  
150 MHz 190mW (incl SRAMs)  
**24 GOPS/W (~ 1/5 IPE)**

imec Courtesy: SiliconHive

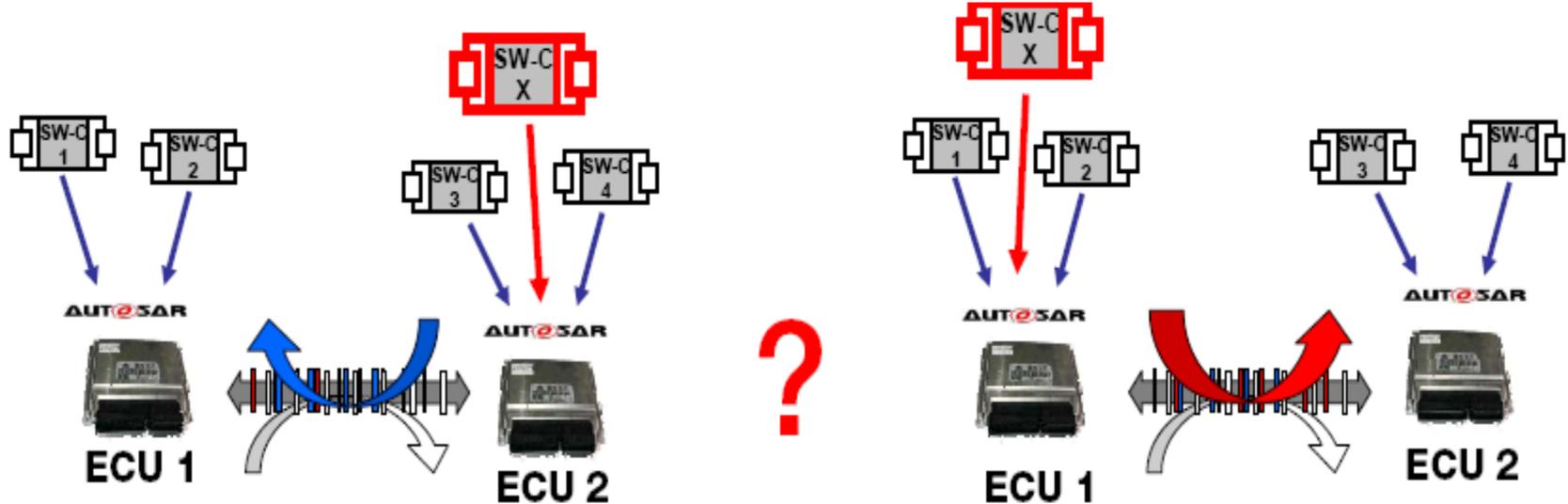
## SH-MobileG1: Chip Overview



© 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

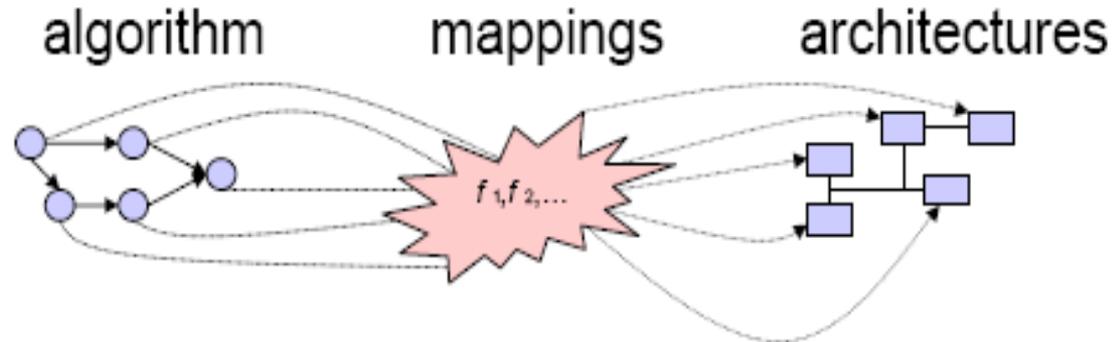
*Which processor should run the software?*

# 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; <b>EXPO/SPEA2</b> SystemCodesigner
Auto-parallelizing	Mneme (Dortmund) Franke (Edinburgh)  MAPS	Daedalus

# Example: System Synthesis

Given:



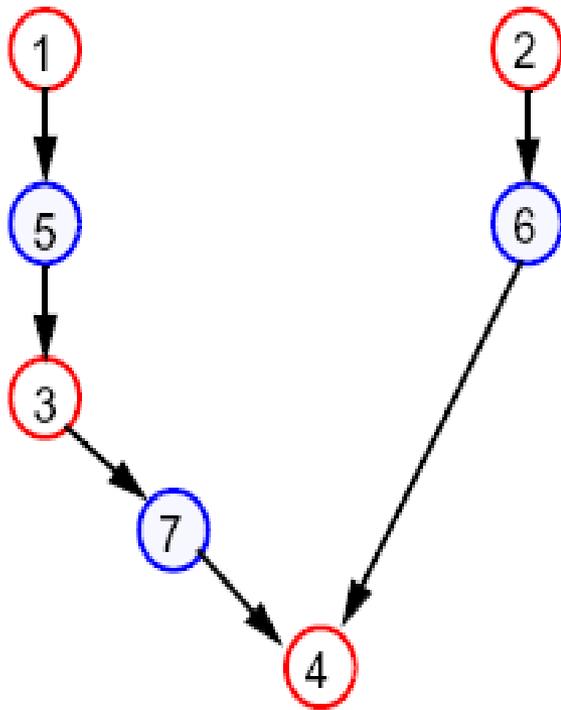
Goal:



Objectives: cost, latency, power consumption

# Basic Model – Problem Graph

Problem graph  $G_P(V_P, E_P)$ :

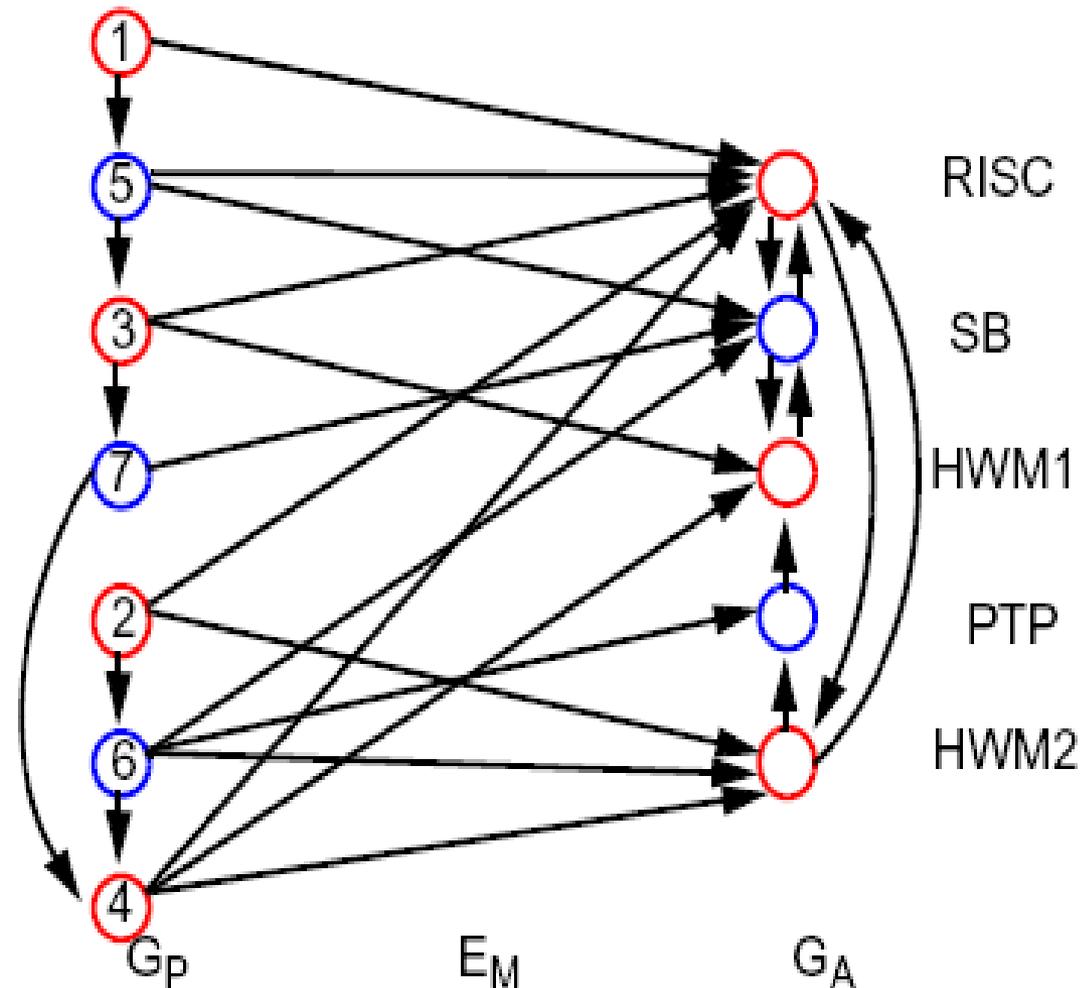


Interpretation:

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

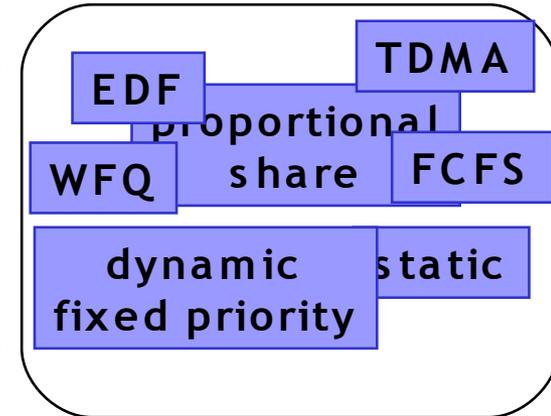
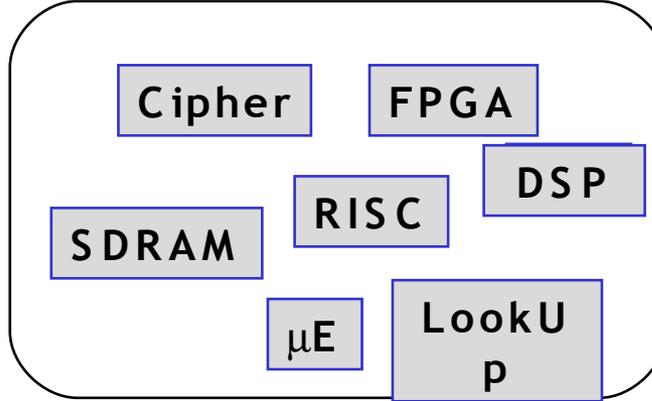
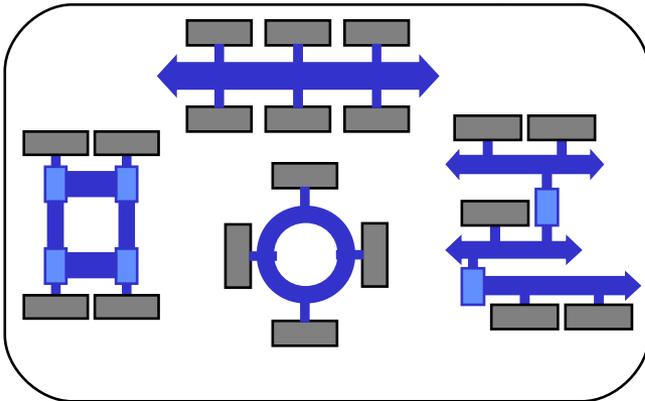
# Basic Model: Specification Graph

**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,  $V_S = V_P \cup V_A$ ,  $E_S = E_P \cup E_A \cup E_M$



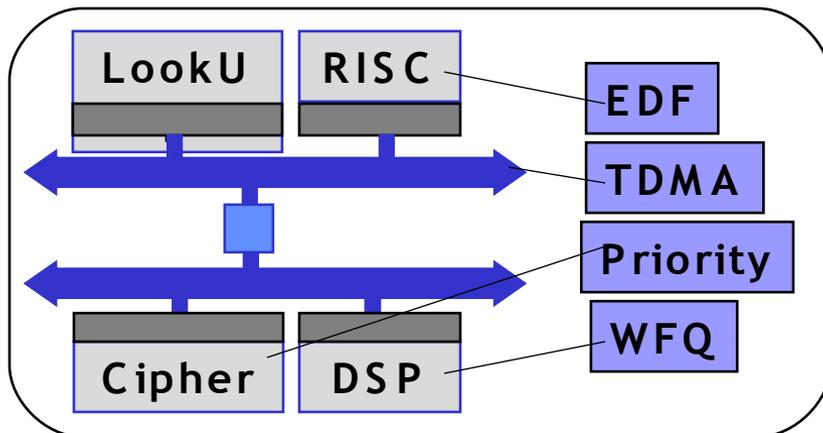
# Design Space

Communication Templates Computation Templates Scheduling/Arbitration

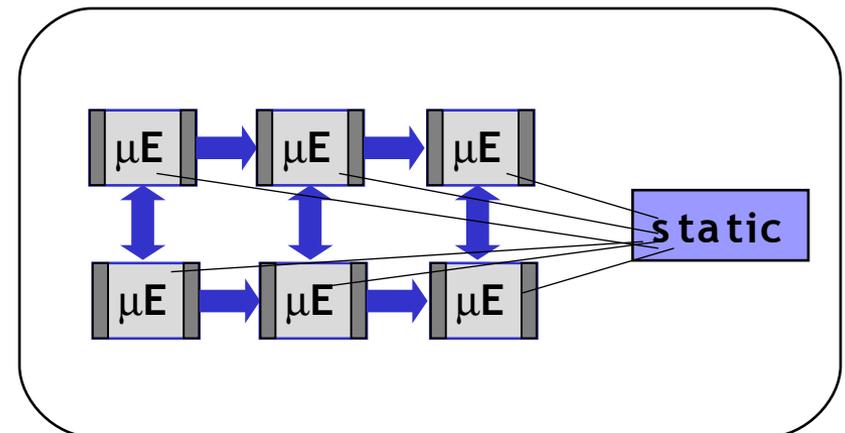


Which architecture is better suited for our application?

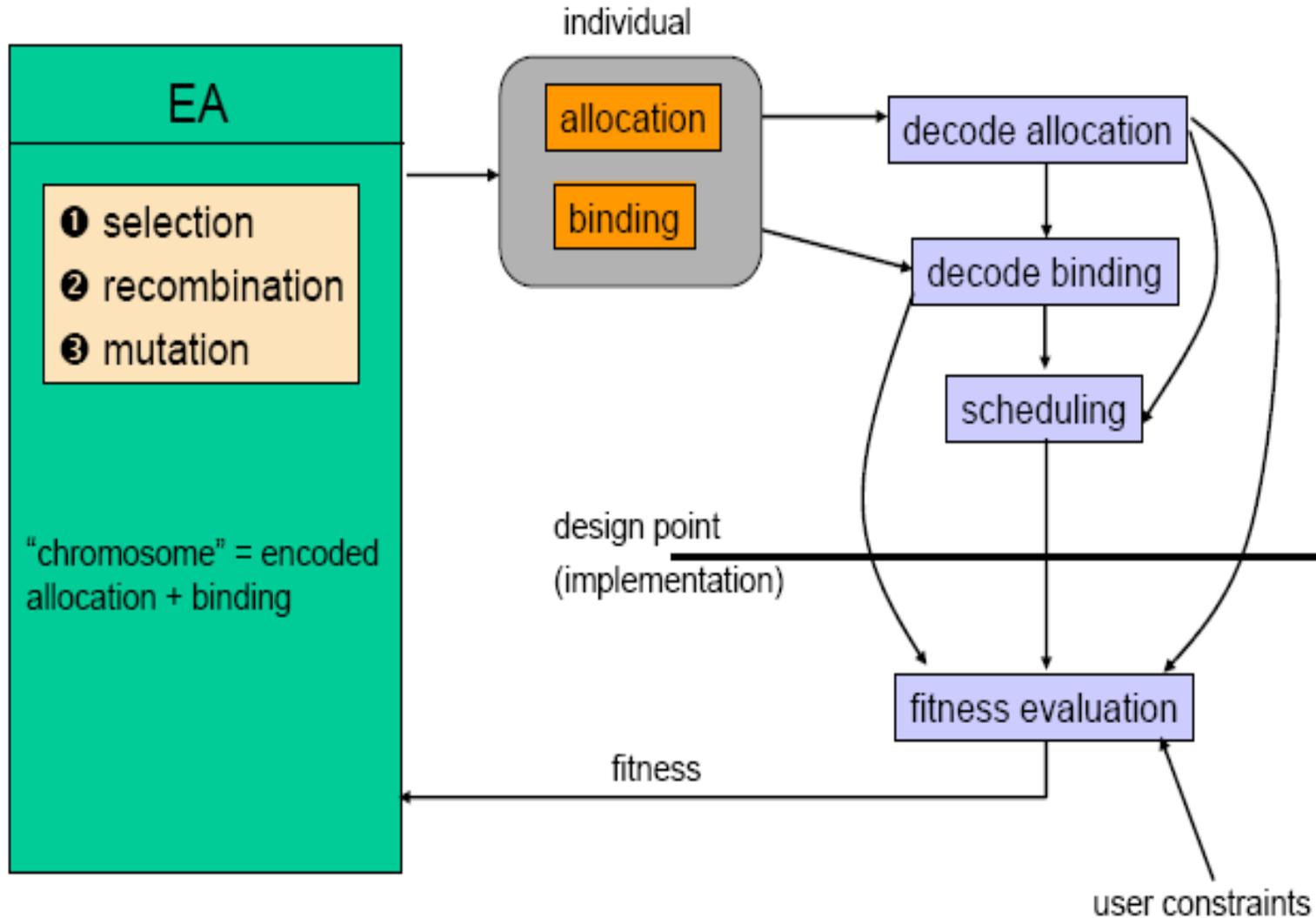
Architecture # 1



Architecture # 2



# Evolutionary Algorithms for Design Space Exploration (DSE)

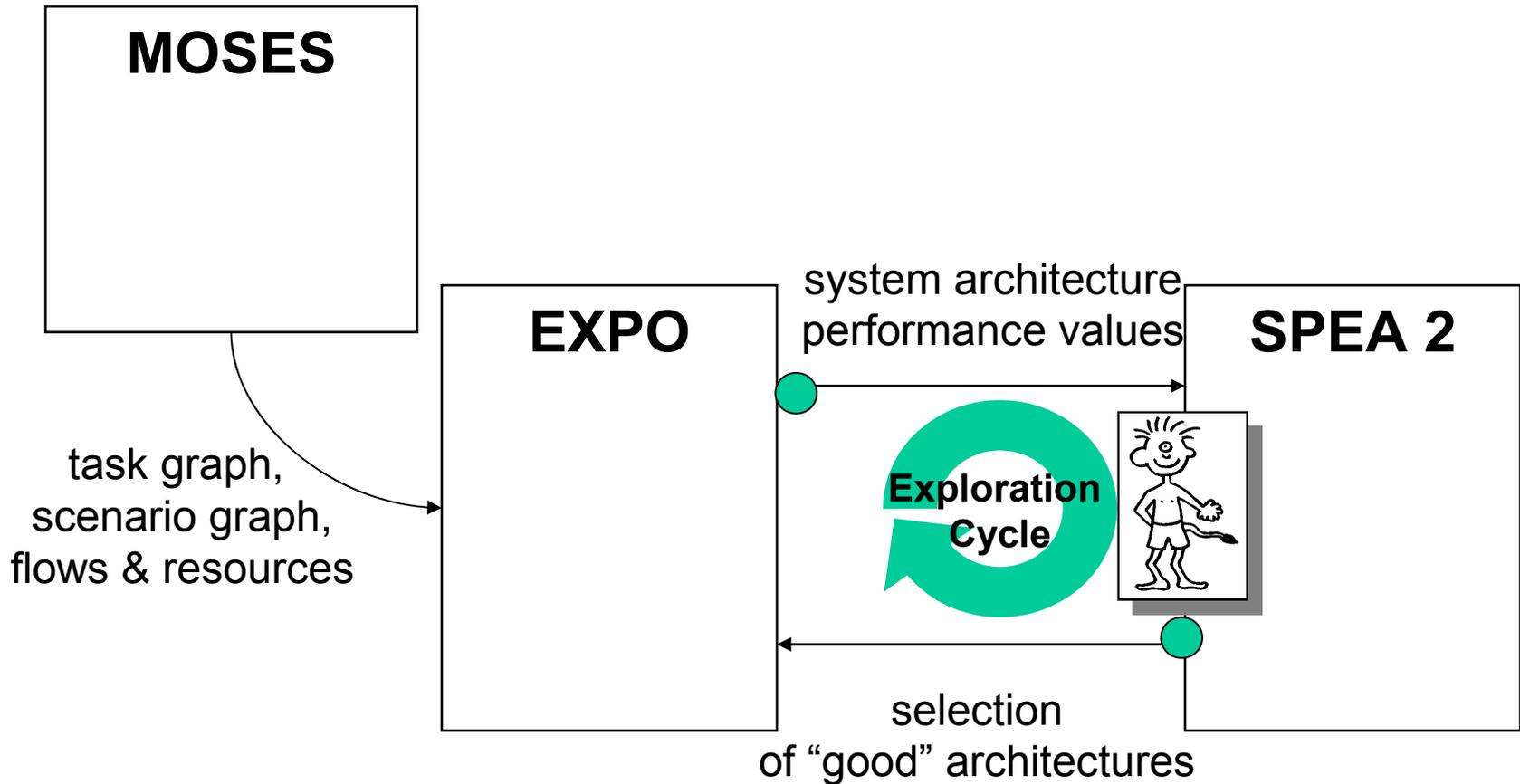


# Challenges

---

- 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)

Moses 1.00+ [31-8-2001]

File Theme Windows Help

Repository Browser

File Tools Repository

Tools

Tool Paths

- Add or Con
- Graph
- Other
- System

Old Rep

Environ

ObjectT

Index F

SetNew

Extract

Objects

- /
- Tools
- cm-classes
- Lib
- Component
- Formalisms
- Graphtypes
- Properties
- images
- Demo
- config
- Other Repositor
- spi

- TimePetriNet (Oct
- Harel (Oct 9, 2001
- ProcessNetwork (O
- BubbleArc (Oct 9, 2
- SimplePetriNet (O
- SPI\_Cluster (Oct 9
- SPI\_Flow (Oct 9, 2
- SPI\_Res (Oct 9, 20

Simple NP (SPI\_RES)

Graph Edit Hierarchy Insert

LinkRx VerifyIP ProcessIP

CheckSum ARM9

Simple NP (SPI\_Cluster)

Graph Edit Hierarchy Insert

Decrypt AHVerify ESPDecaps

ESPEncaps AHUaic Fncrypt

RouteLU2

VerifyIP ProcessIP Classify

UDPrx RTPrx Dejitter VoiceDec

UnitsNotNull -> Units for Resource Types must be specified, lowerCurve -> Lower Curve for Resource Types must be given, upperCurve -> Upper Curve for F Moses Tool Suite (c) 1999-2001 ETH

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

© L. Thiele, ETHZ

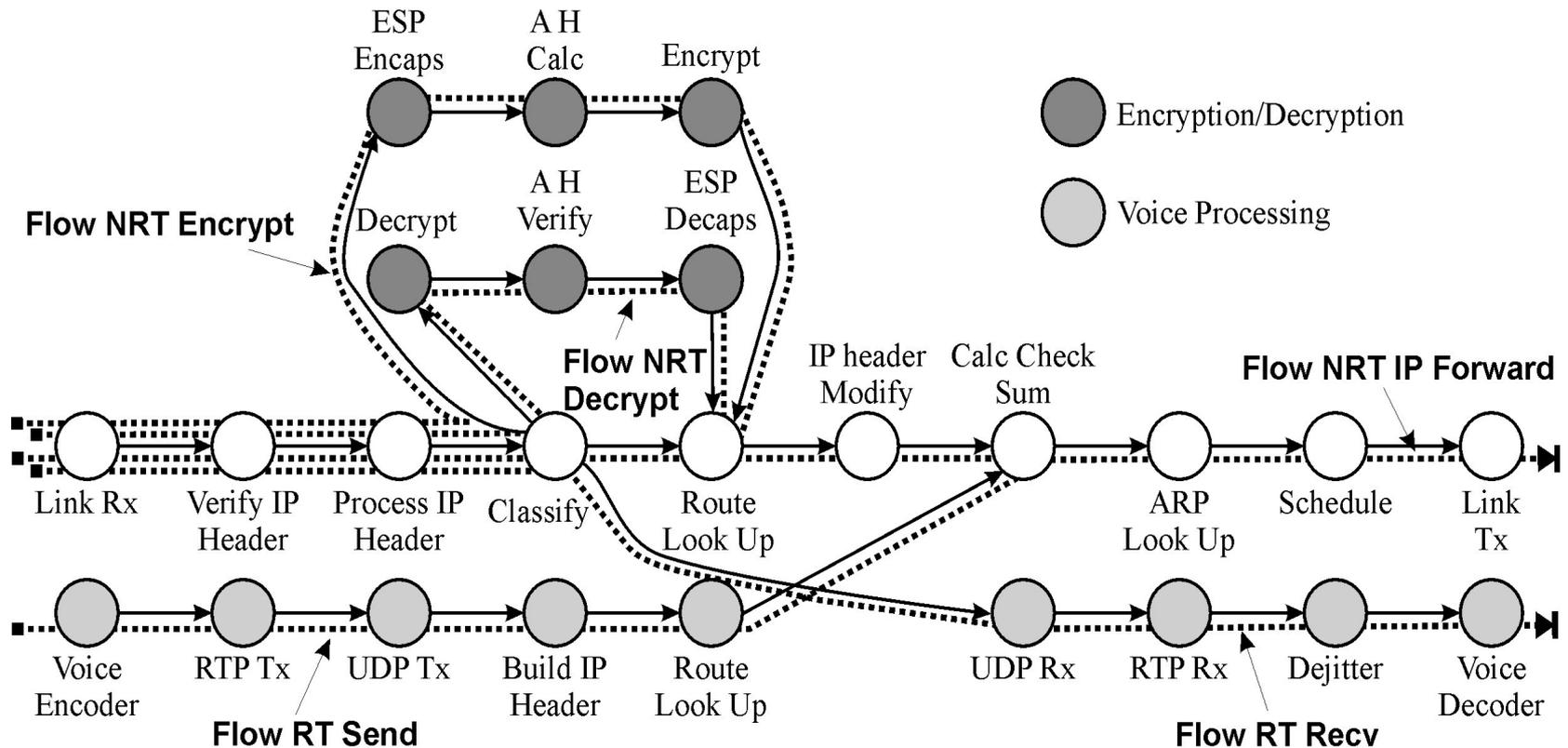
# EXPO – Tool (3)

The screenshot displays the EXPO tool interface with three main components:

- Log Window (Left):** Shows the execution progress, including initialization, population construction, and the start of two generations. The log text includes:
  - 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.
- Scatter Plot (Middle):** A plot titled "EXPO, Institute TIK, ETH Zurich" showing "current population" data points. The x-axis is labeled "x axis" and ranges from -1.8 to -1.0. The y-axis is labeled "y axis" and ranges from 0.5 to 7.5. A red diamond marker is at approximately (-1.8, 7.5).
- Implementation Window (Right):** Titled "Implementation Nr. 60641 (EXPO, Institute TIK, ETH Zurich)", it shows:
  - Scenario: Scen2
  - Optimal Scaling Factor: 0.530
  - Total Memory: 8.295
  - Resource Utilization: DSP (79%), CheckSum (4%), LookUp (7%).
  - A network flow diagram with three flows:
    - Flow: RTSend (Priority: 5):** RTPtx, VoiceEnc, LinkTx, Schedule, UDPtx, CalcCheck, BuildIP, RouteLU1, ARPLU. Acc. Waiting Time in Queue: 0.000.
    - Flow: NRTDecrypt (Priority: 4):** ESPDecaps, ProcessIP, IPModify, LinkTx, Schedule, Decrypt, AHVerify, Classify, LinkRx, VerifyIP, CalcCheck, ARPLU, RouteLU2. Acc. Waiting Time in Queue: 0.000.
    - Flow: RTRecv (Priority: 1):** Dejitter, VoiceDec, ProcessIP, RTPrx, Classify, LinkRx, VerifyIP, UDPrx. Acc. Waiting Time in Queue: 0.000.
    - Flow: NRTForward (Priority: 3):** ProcessIP, VerifyIP, ARPLU. Acc. Waiting Time in Queue: 23.088.

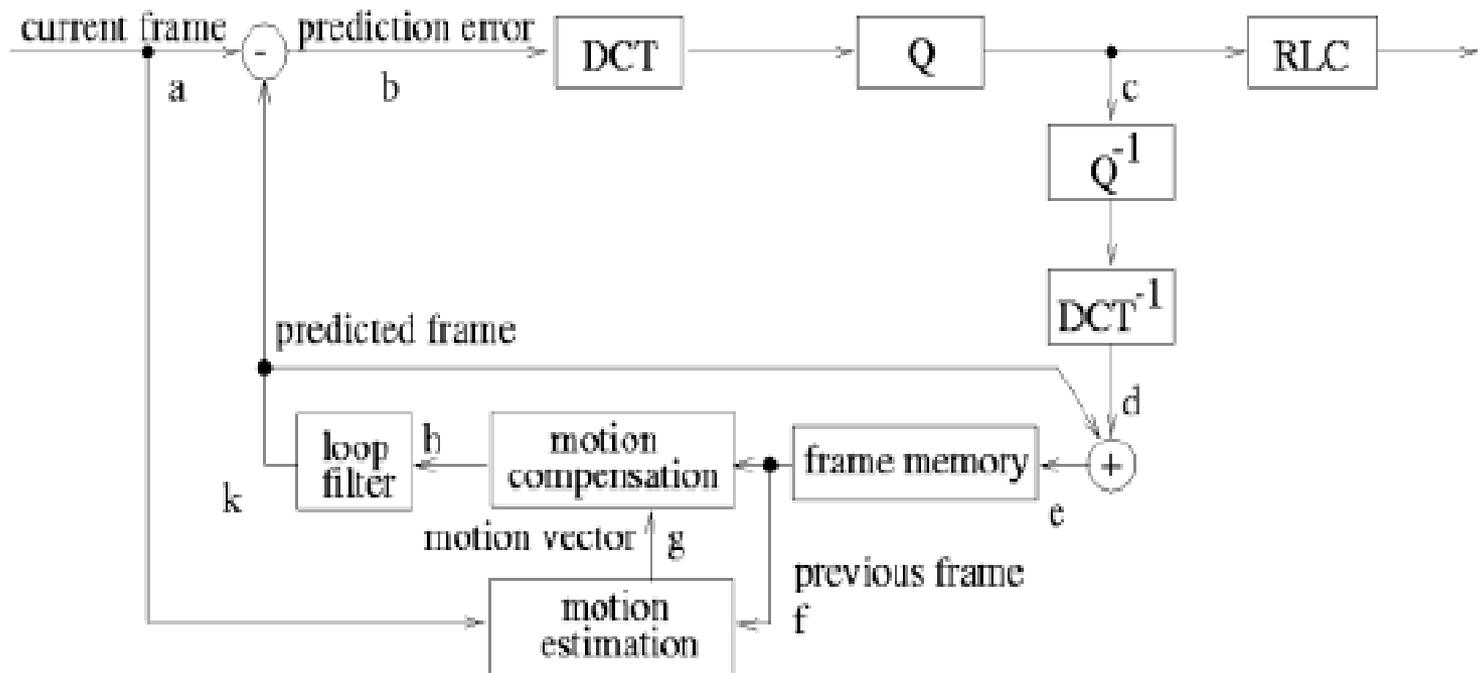
# Application Model

Example of a simple stream processing task structure:



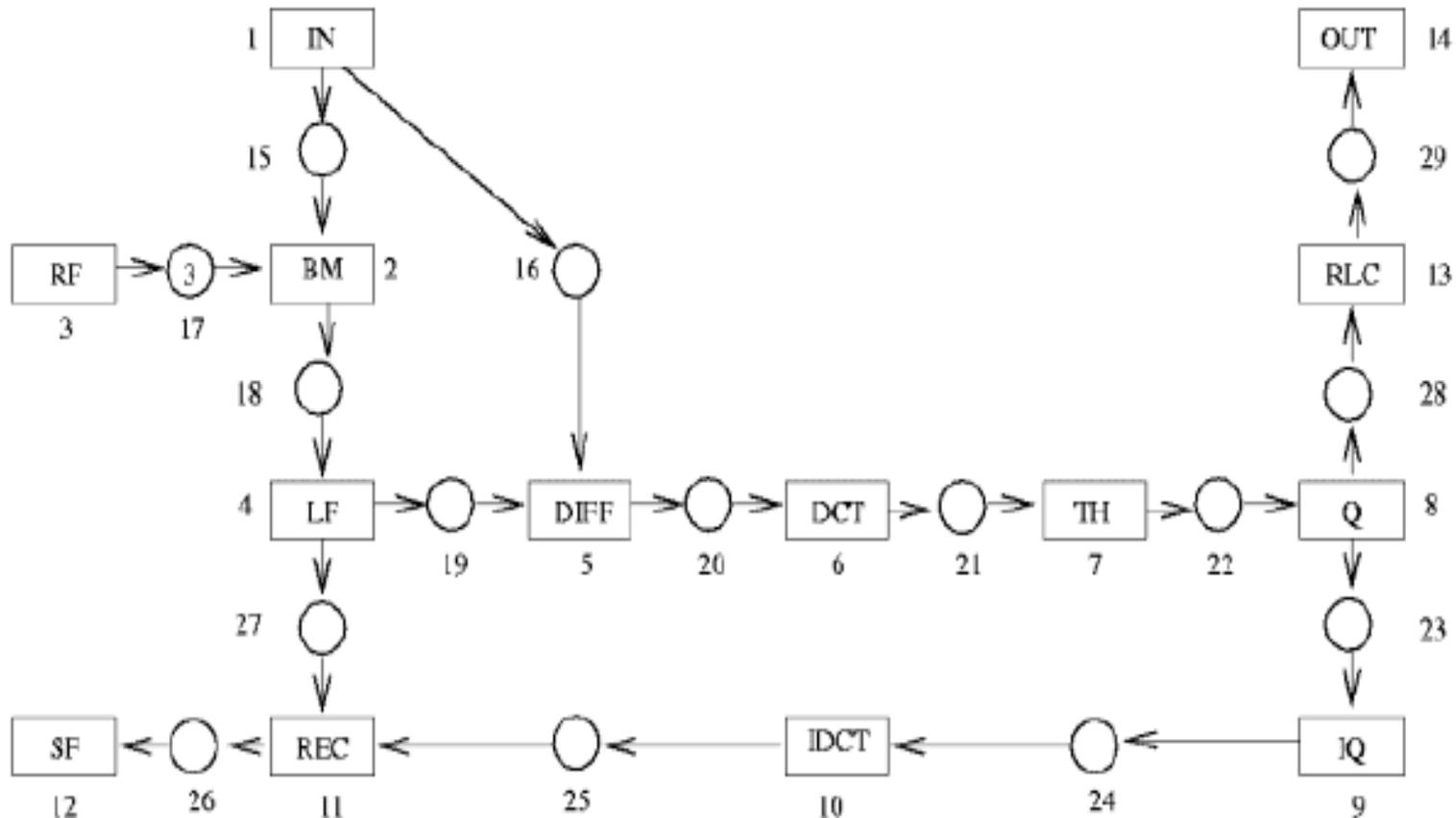
# Exploration – Case Study (1)

## behavioral specification of a video codec for video compression

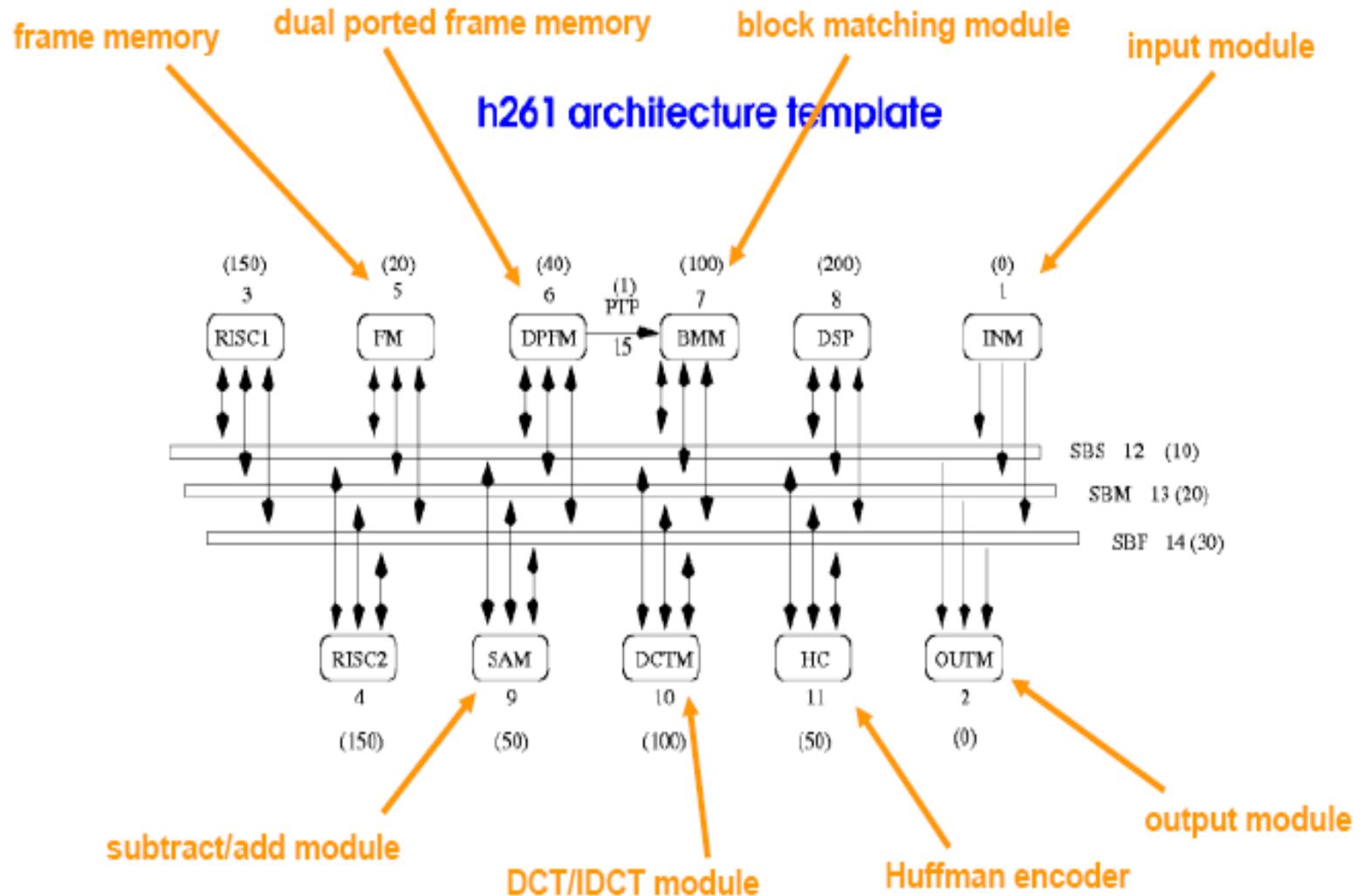


# Exploration – Case Study (2)

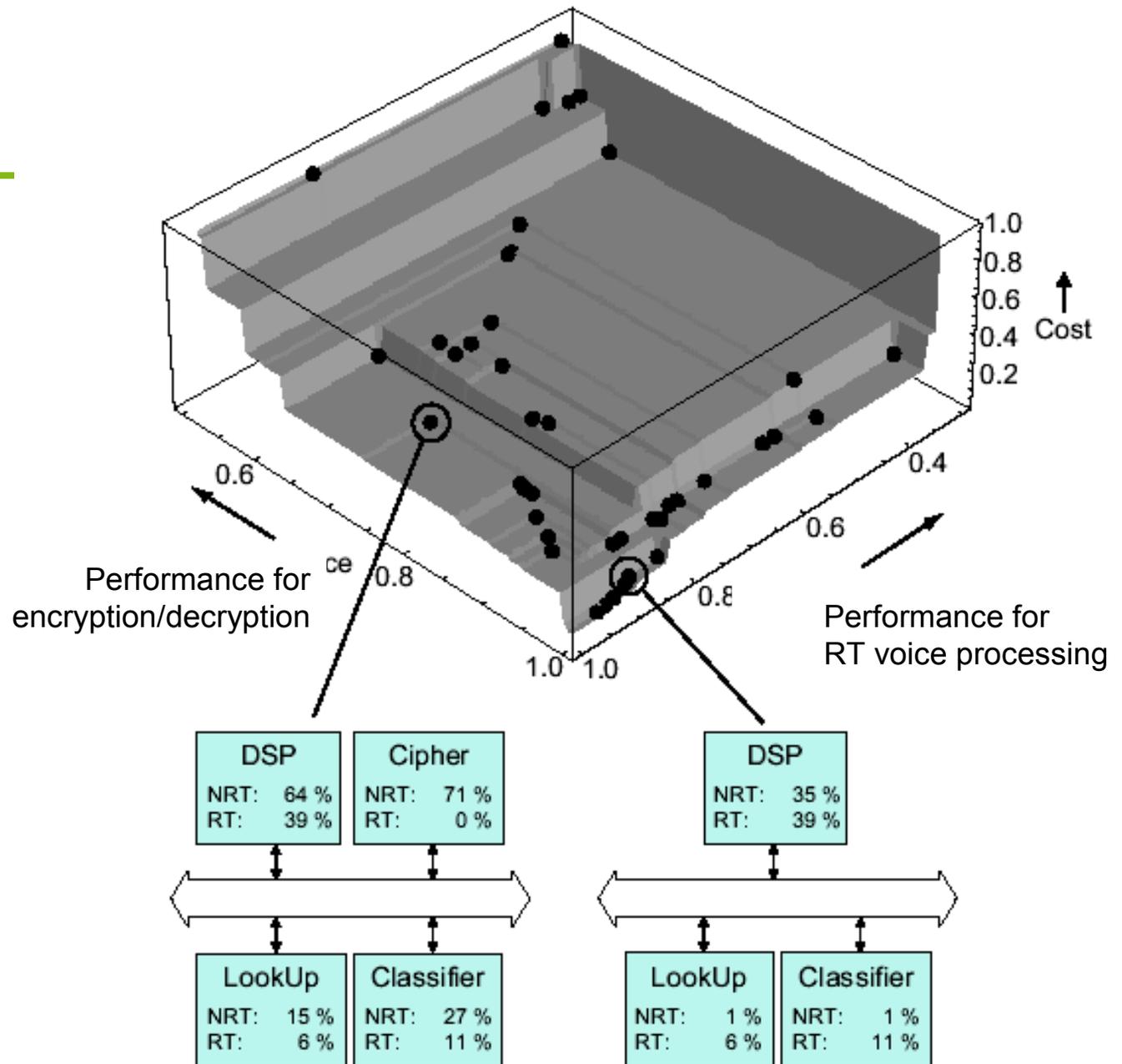
## problem graph of the video coder



# Exploration – Case Study (3)



# More Results



# Design Space Exploration with SystemCoDesigner

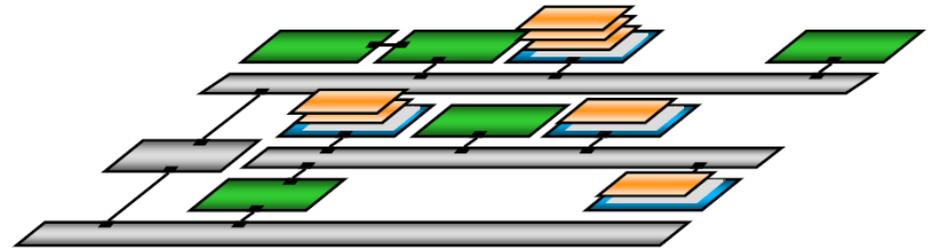
(Teich, Erlangen)

- 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

## System Synthesis (Actor Binding)



- $A$  denotes the set of actors
- Actor binding activation  $\alpha: A \times R \rightarrow \{0, 1\}$
- $\alpha(a, r) = 1$  binds actor  $a$  onto resource  $r$
- $\forall a \in A: \sum \alpha(a, r) = 1$  (Each actor is bound exactly once)

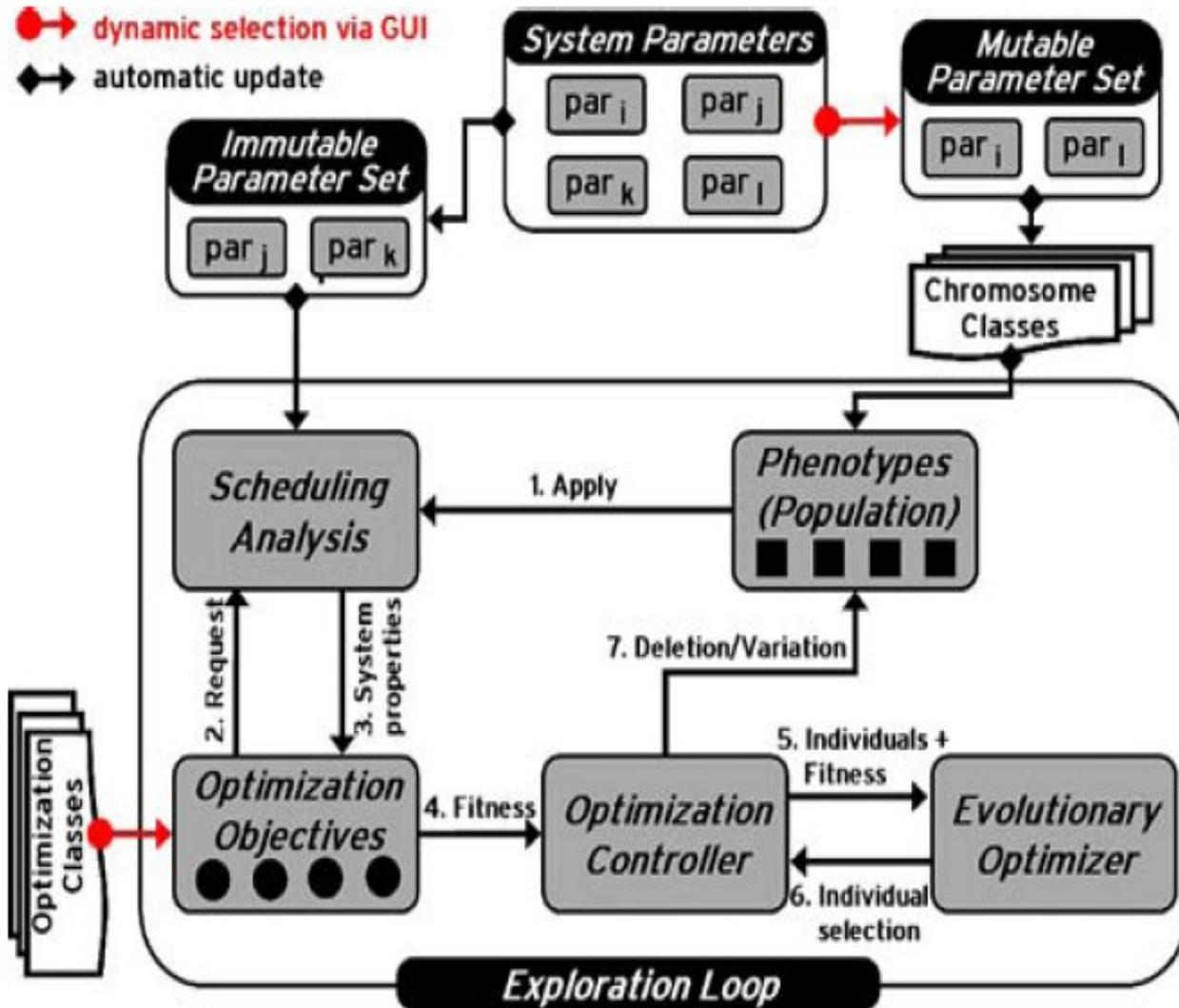


© University of Erlangen-Nuremberg  
Hardware-Software-Co-Design

8

© J. Teich, U. Erlangen-Nürnberg

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

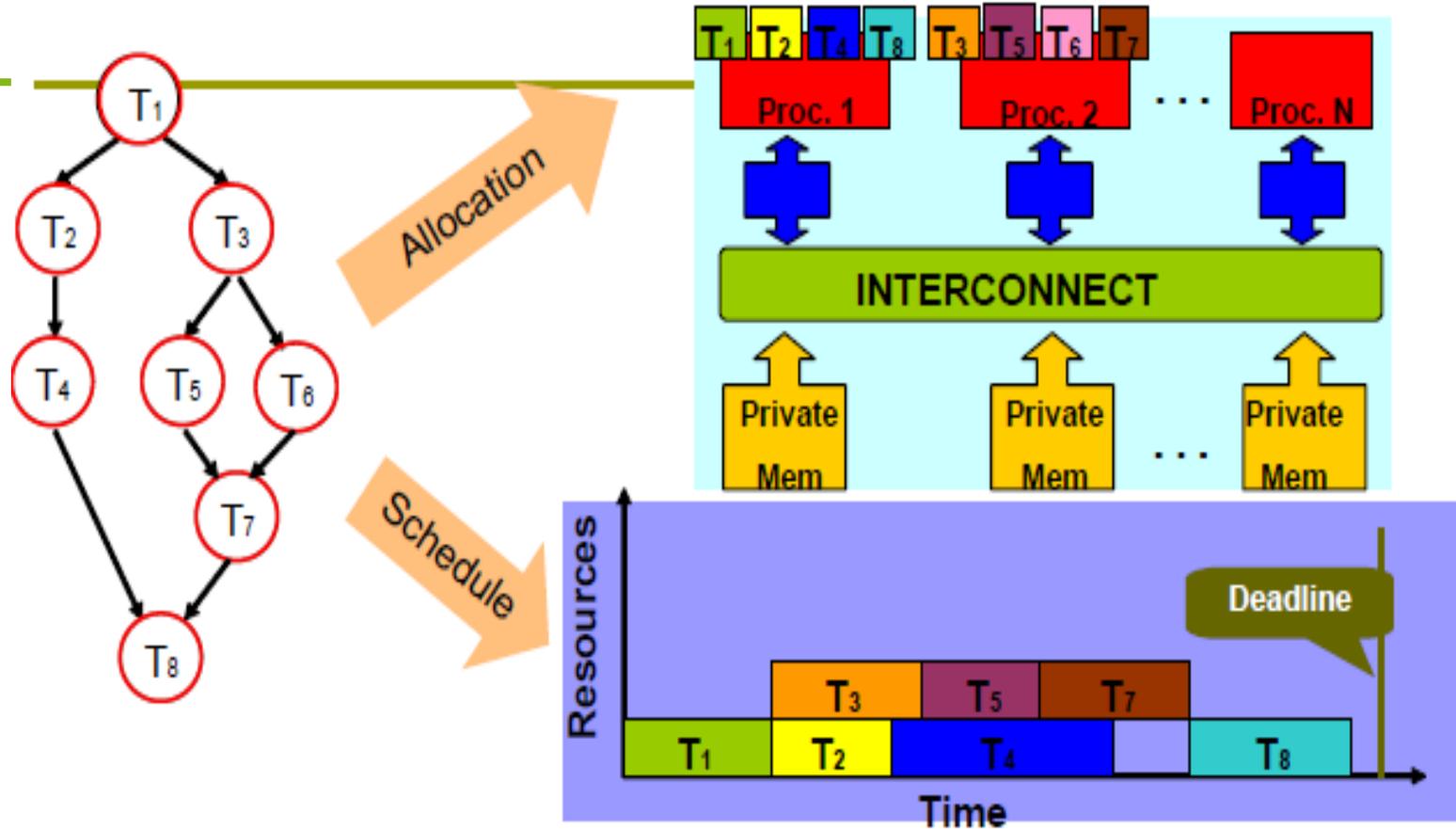


[R. Ernst et al.: A framework for modular analysis and exploration of heterogeneous embedded systems, *Real-time Systems*, 2006, p. 124]

# 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; <b>EXPO/SPEA2</b> SystemCodesigner
Auto-parallelizing	Mneme (Dortmund) Franke (Edinburgh)  MAPS	Daedalus

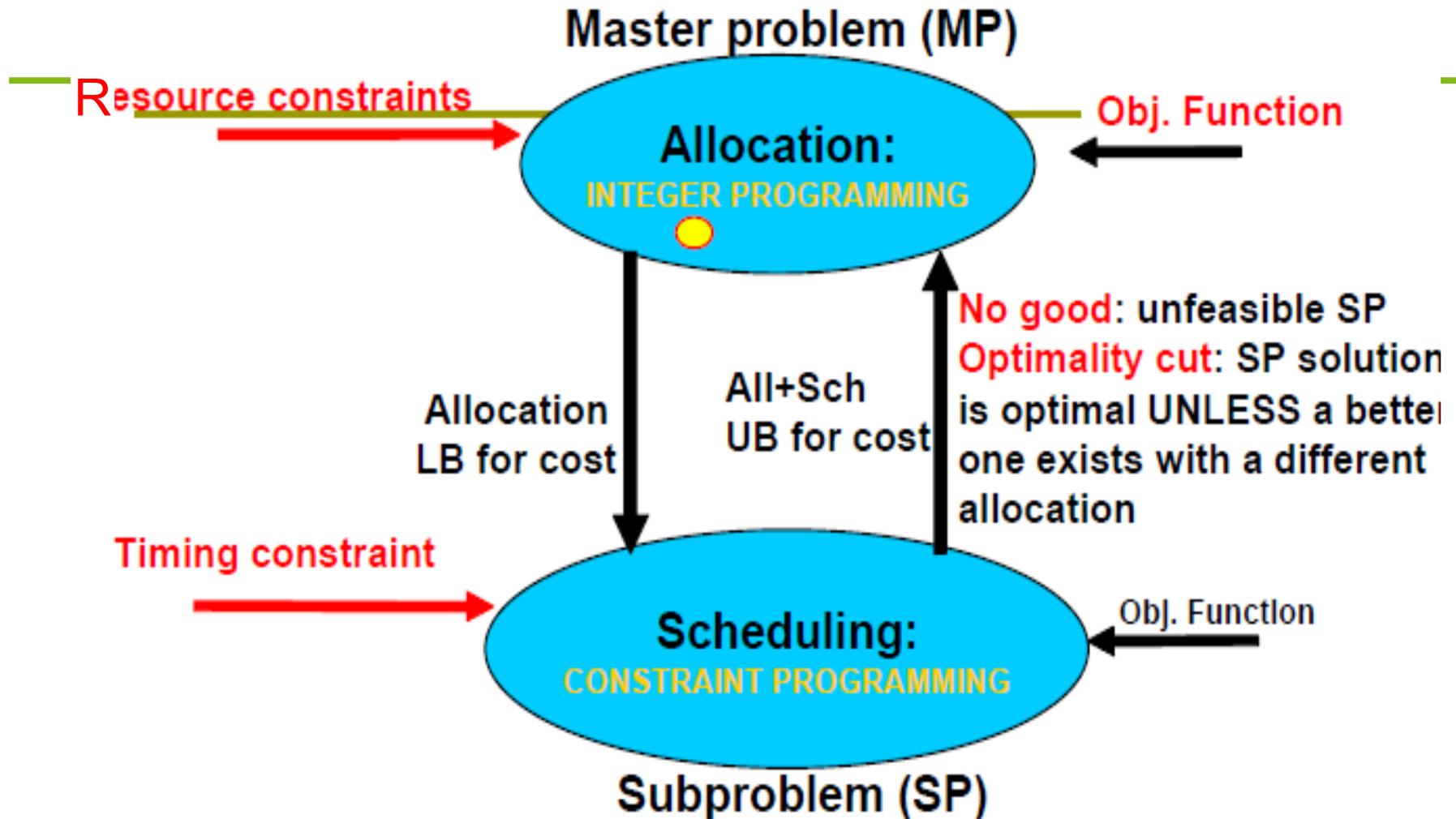
# A fixed architecture approach: Map $\rightarrow$ CELL



- 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

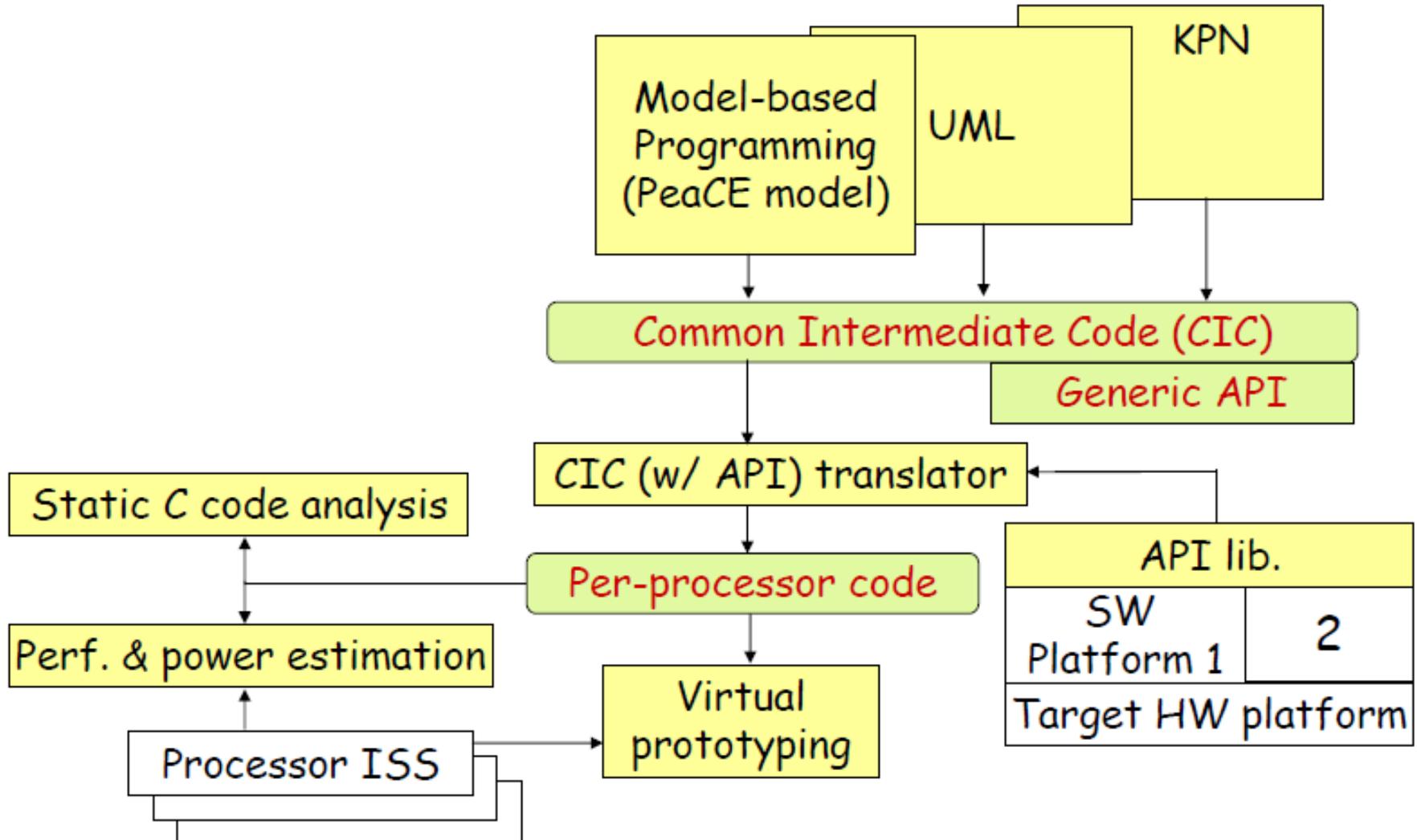


Iterations stop when MP becomes unfeasible!



# HOPES Proposal

HOPES



Jan. 24, 2007

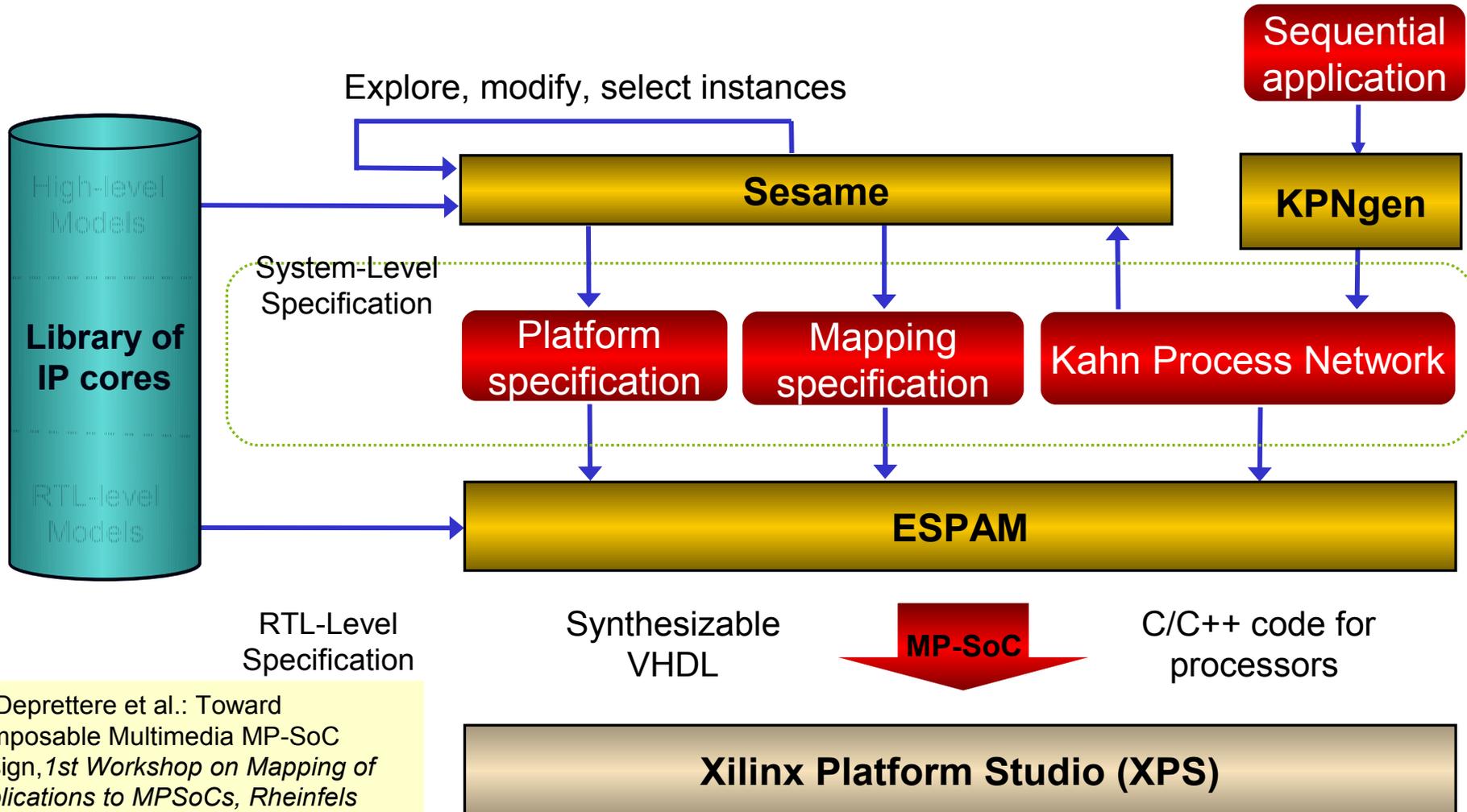
Soonhoi Ha, SNU

9

# 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; <b>EXPO/SPEA2</b> SystemCodesigner
Auto-parallelizing	Mneme (Dortmund) Franke (Edinburgh)  MAPS	<b>Daedalus</b>

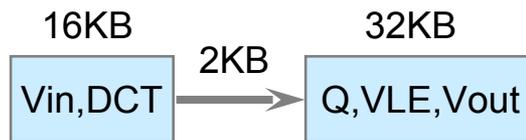
# Daedalus Design-flow



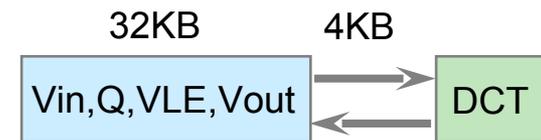
Ed Deprettere et al.: Toward Composable Multimedia MP-SoC Design, *1st Workshop on Mapping of Applications to MPSoCs, Rheinfels Castle, 2008*

# JPEG/JPEG2000 case study

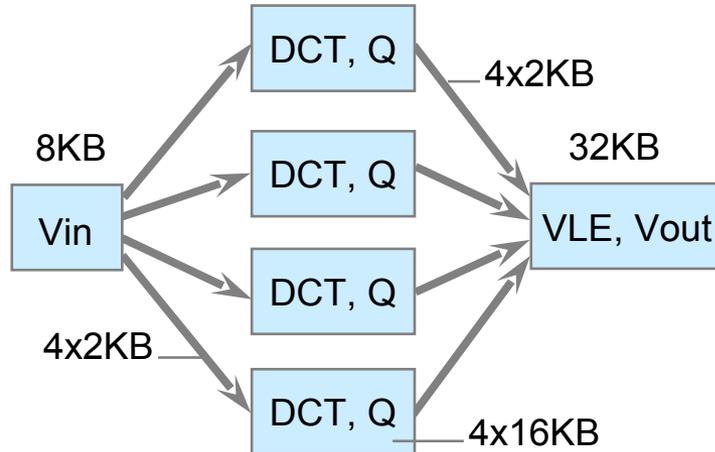
## Example architecture instances for a single-tile JPEG encoder:



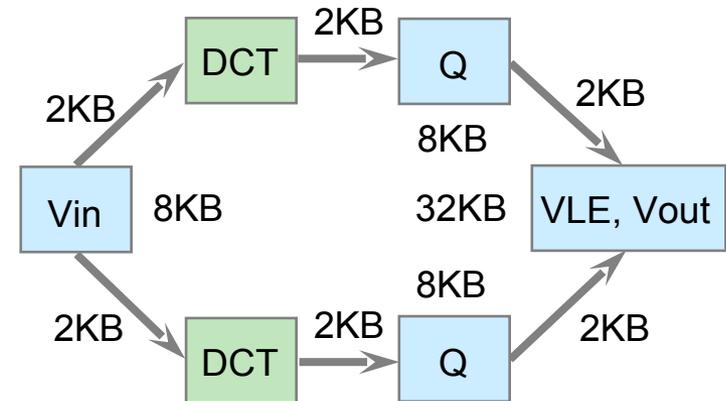
2 MicroBlaze processors (50KB)



1 MicroBlaze, 1HW DCT (36KB)



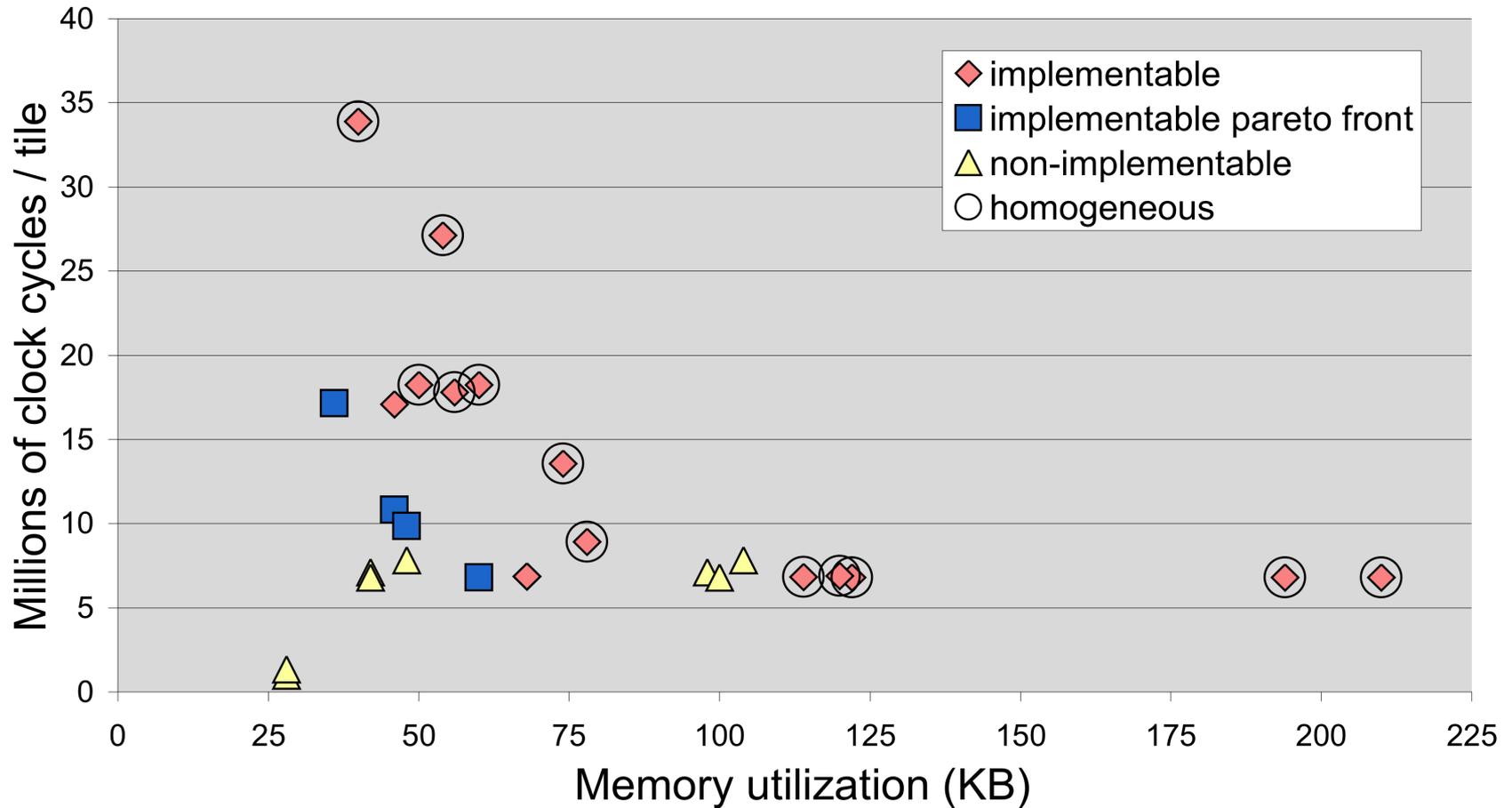
6 MicroBlaze processors (120KB)



4 MicroBlaze, 2HW DCT (68KB)

# Sesame DSE results: Single JPEG encoder DSE

## Performance-memory trade-off 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; <b>EXPO/SPEA2</b> SystemCodesigner
Auto-parallelizing	Mneme (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!***

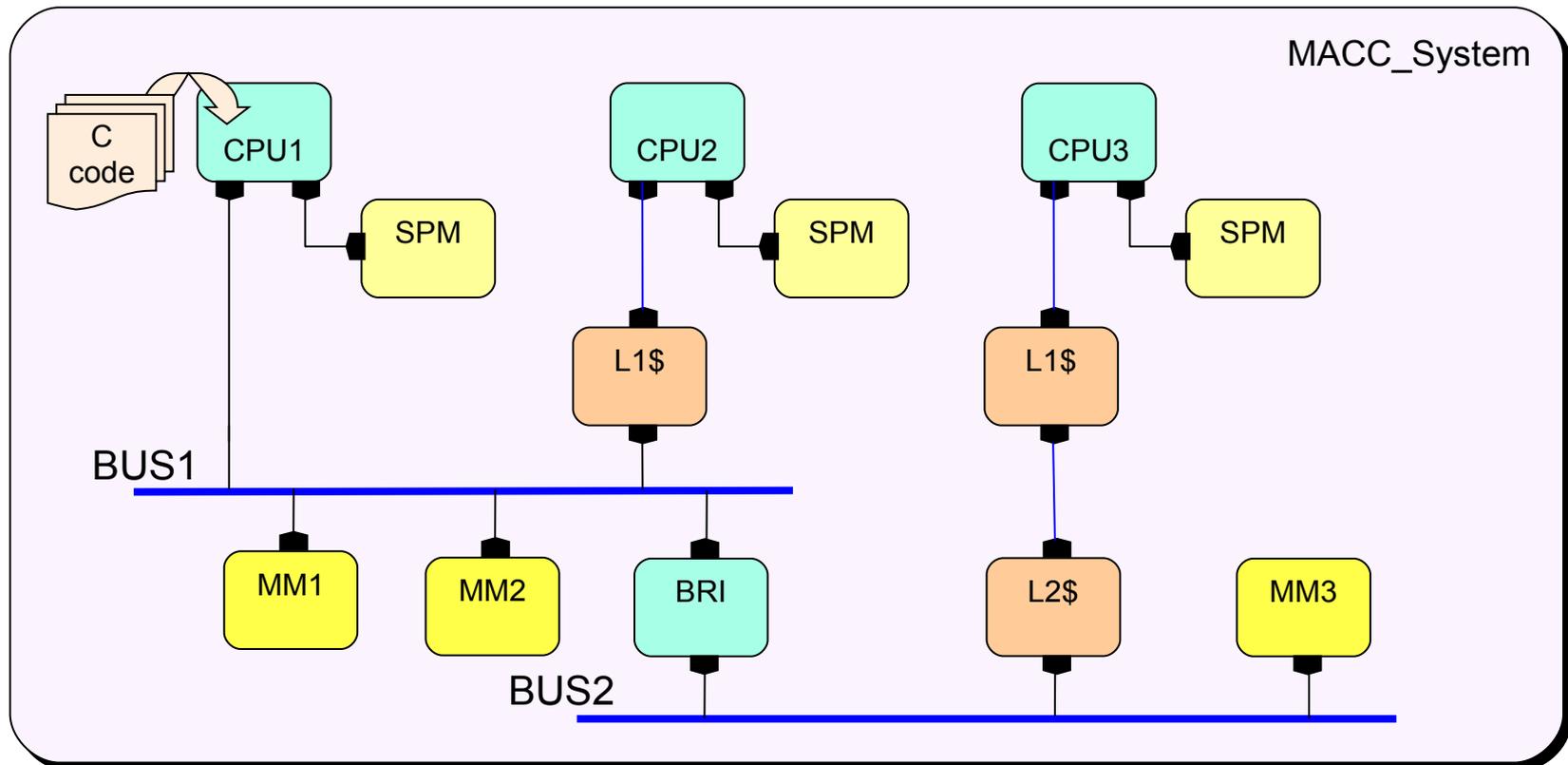
☞ Work of Franke, O’Boyle (Edinburgh)

© 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

Macc - MaccTest/macedExample1.maced - Eclipse SDK

File Edit Navigate Search Project Run Window Help

50%

Navigator

- MaccTest
  - .project
  - macedExample1.maced
  - mparm3.xml

Outline

Container	Config-Value
boolcfg	false
intcfg	0
AccessAspectHandler	
AddrSpace	
GuiContent	
Port	

Properties

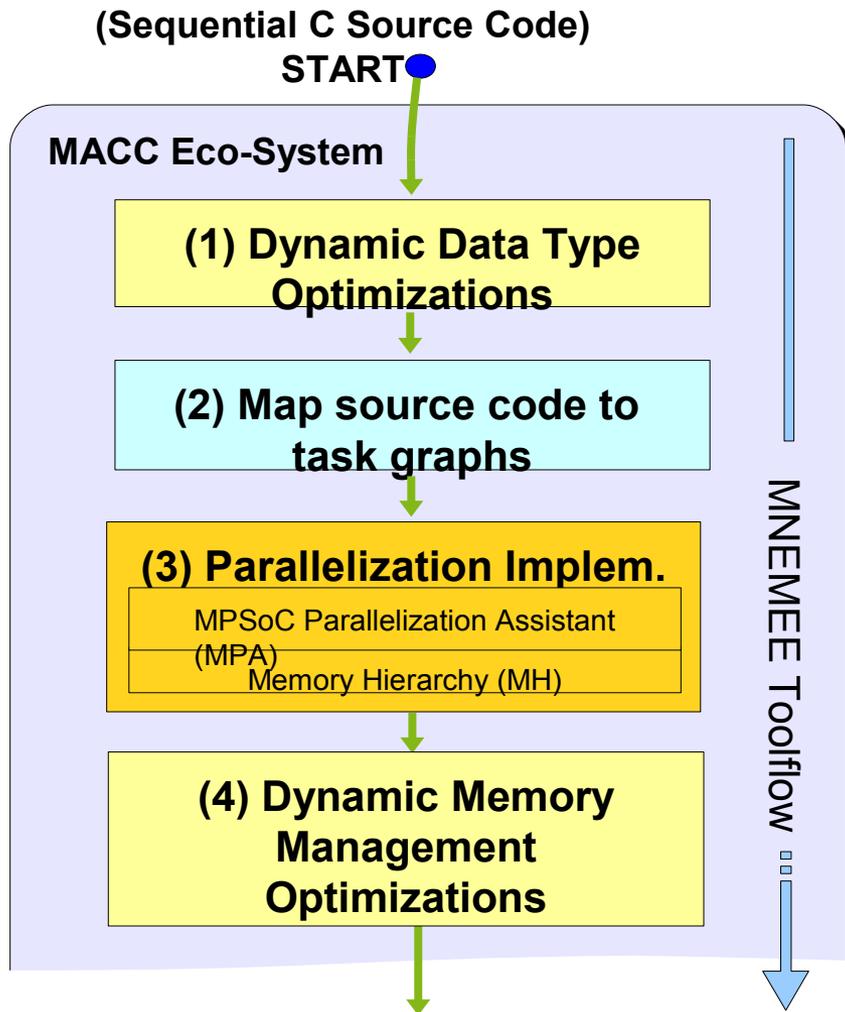
Name	Adressraumbeginn	Adressraumende
ProcessorAddrSpace	0x0	0xffffffff

Diagram Description: The diagram shows a system architecture with two ARM processors at the top, each connected to a LOCAL bus. Below each LOCAL bus is a Cache, a Busbridge, and a Scratchpad. These are connected to a central AHB bus, which is further connected to Private and Shared memory blocks.

Palette

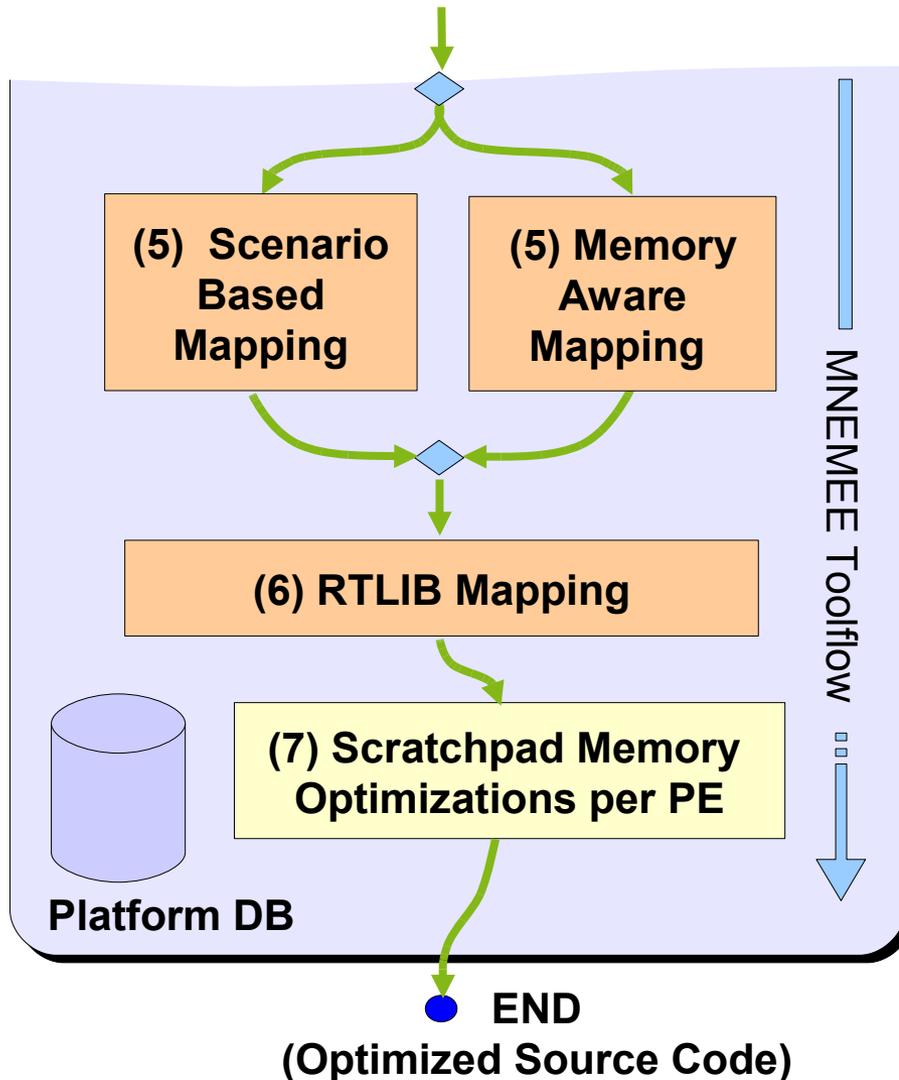
- Selektieren
- Markieren
- Bus - Konnektor
- D - Konnektor
- Komponent...
- Prozessor
- Speicher
- Cache
- Komponente
- Busse
- Bus

# Toolflow Detailed View



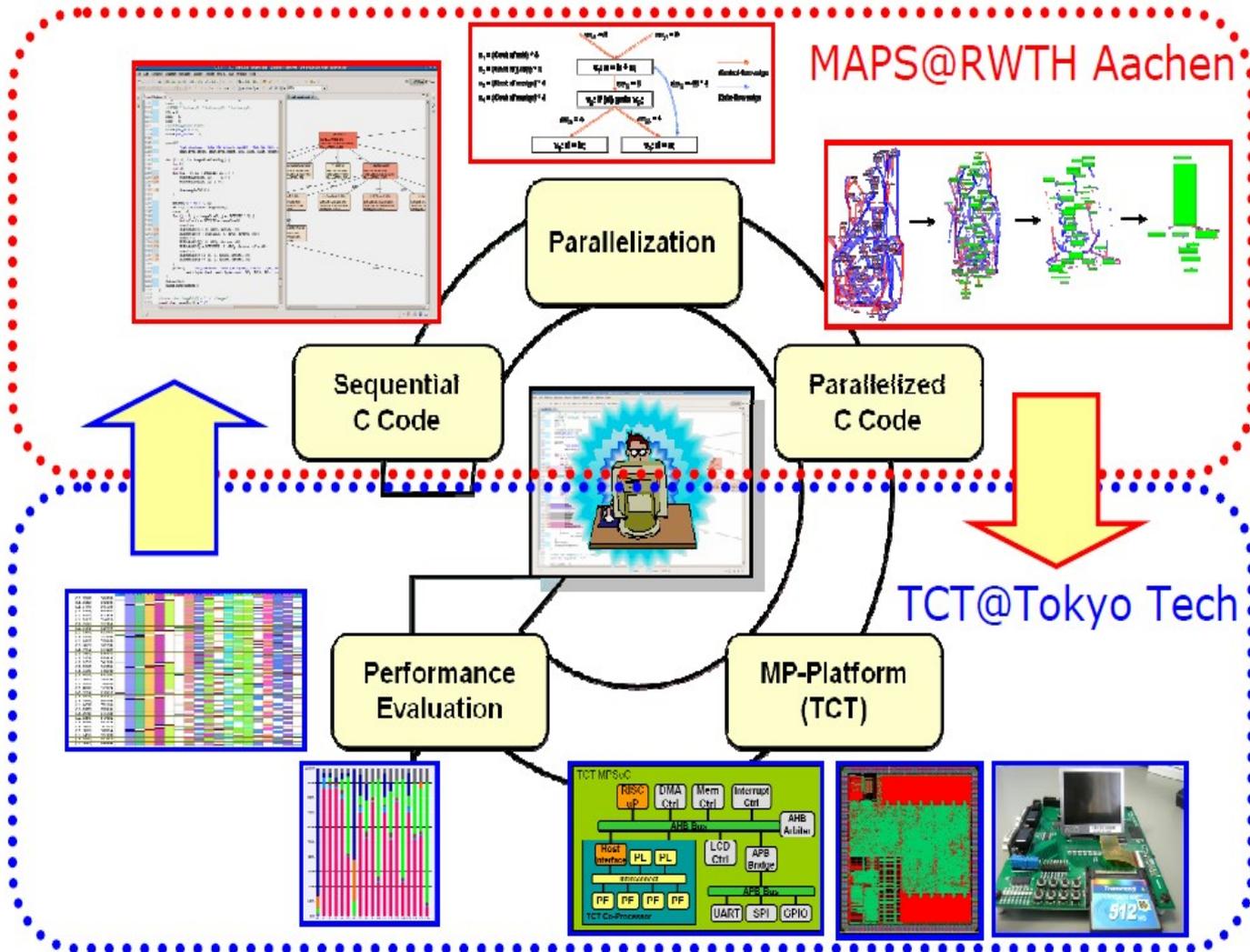
1. Optimization of dynamic data structures
2. Extraction of potential parallelism
3. Implementation of parallelism; placement of static data
4. Placement of dynamic data

# Toolflow Detailed View



5. Perform mapping to processing elements
  - Scenario based
  - Memory aware
5. Transform the code to implement the mapping
6. 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