technische universität
dortmund

fakultät für informatik
informatik 12

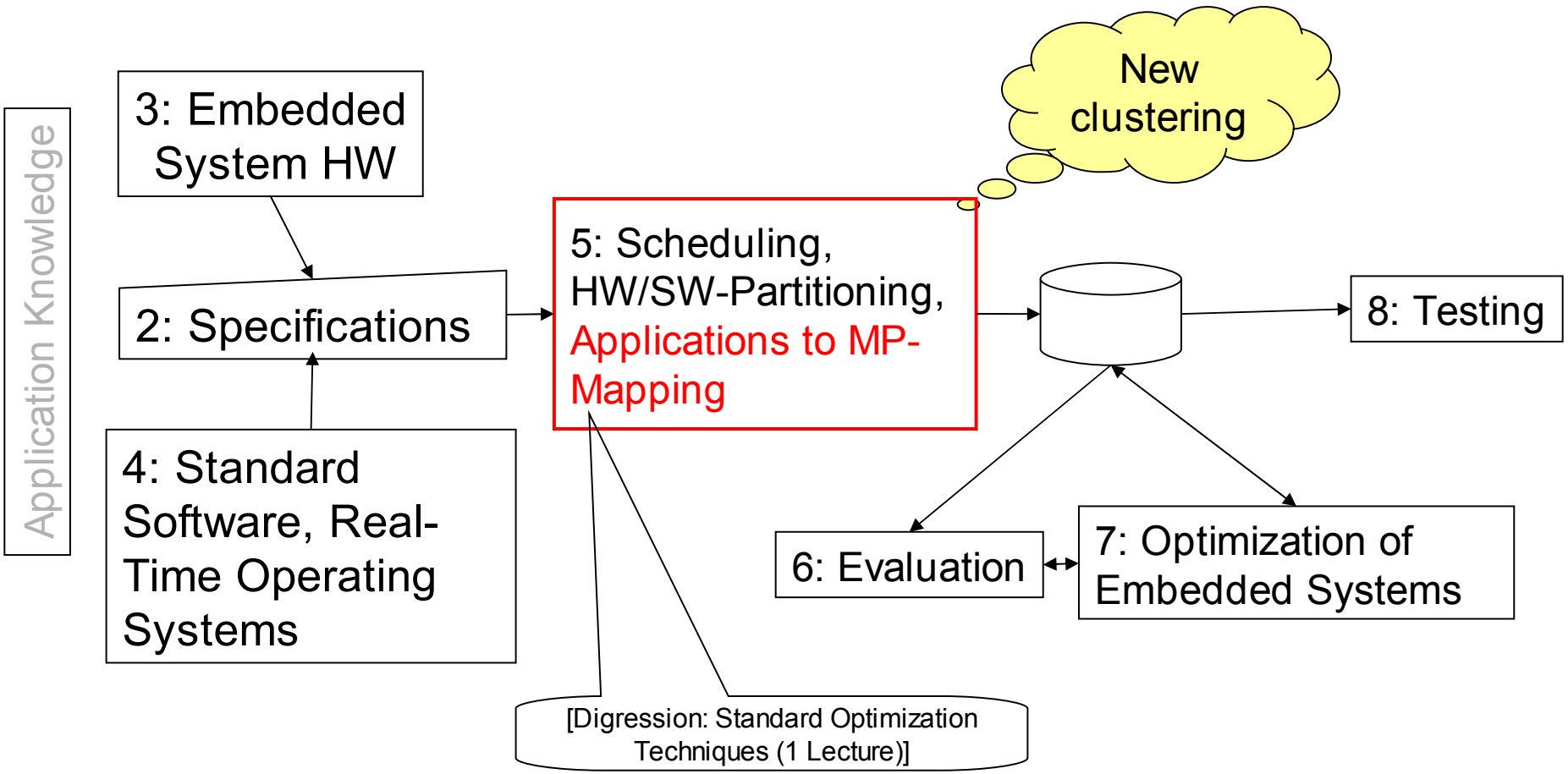
Mapping of Applications to Multi-Processor Systems

Peter Marwedel
Informatik 12
TU Dortmund
Germany

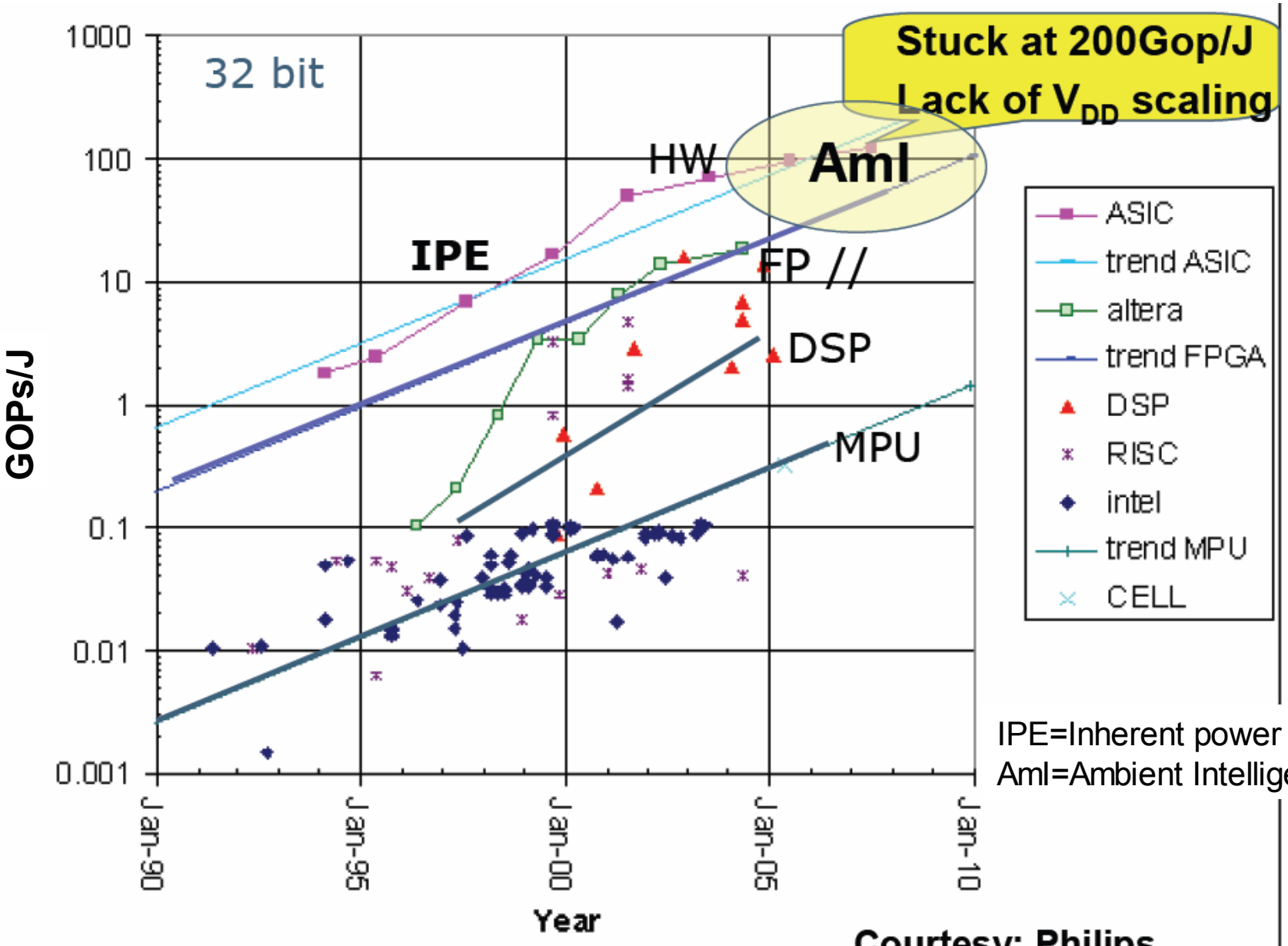
Graphics: © Alexandra Nolte, Gesine Marwedel, 2003

2008/12/14

Structure of this course



Energy Efficiency

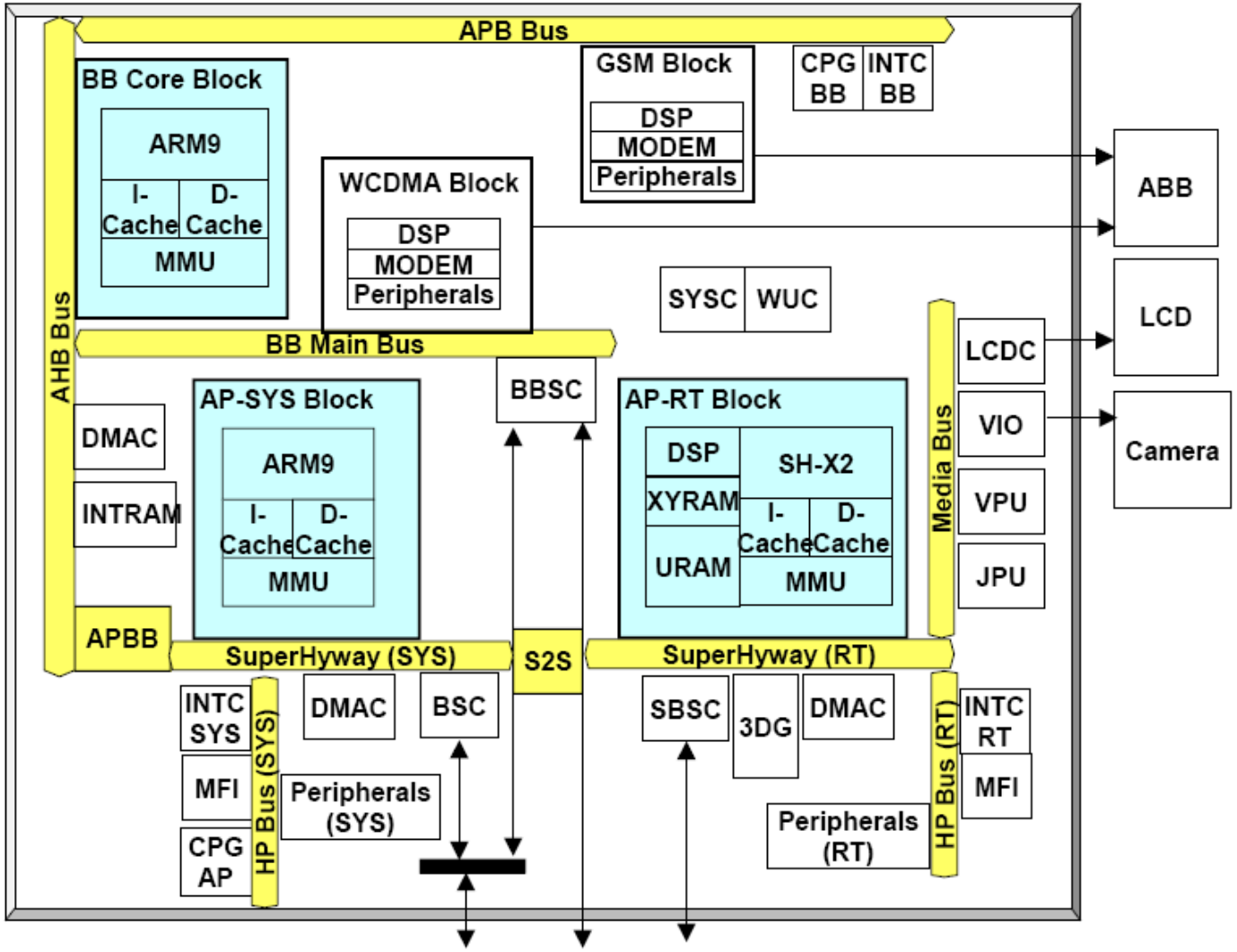


Courtesy: Philips
© Hugo De Man, IMEC, 2007

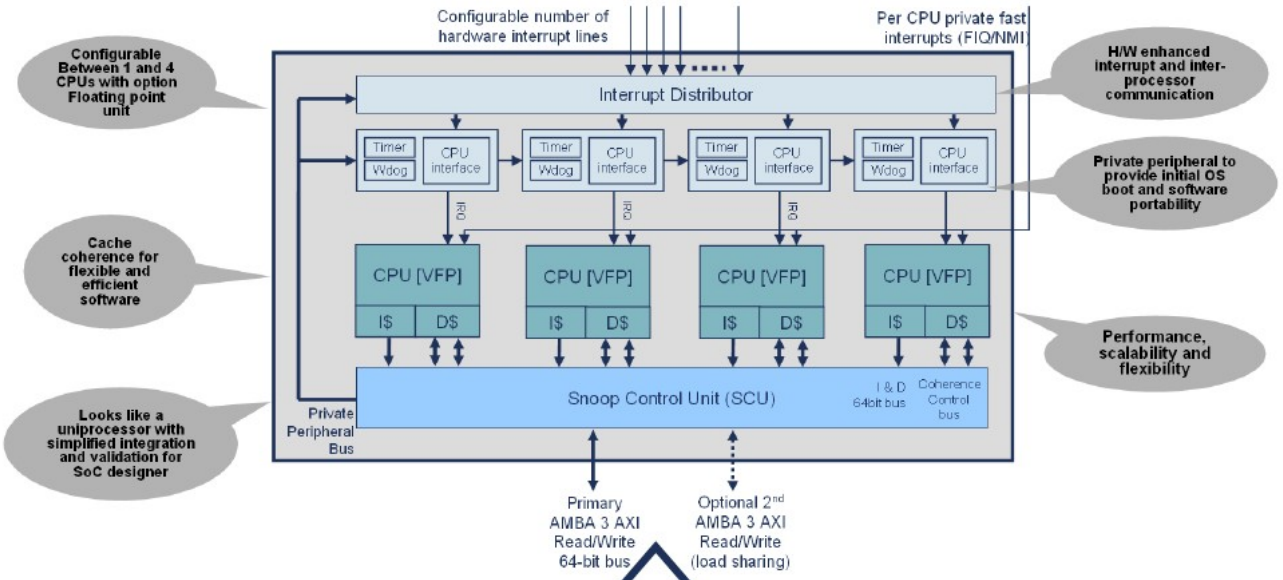
IPE=Inherent power efficiency
AmI=Ambient Intelligence

Heterogeneous Architectures

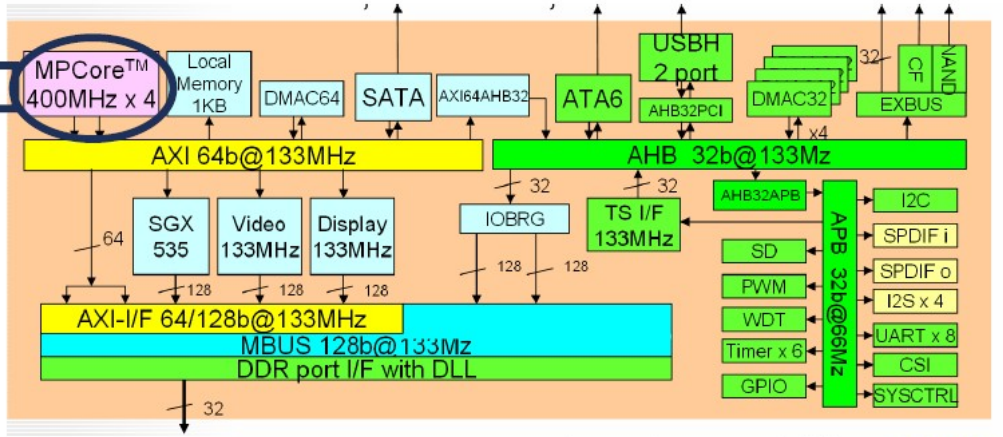
G1 Module Diagram



Homogeneous Architectures



- Utilizing the ARM11 MPCore multicore processor to extend the host processor node (application software domain) to execute across multiple homogeneous cores



NaviEngine®の特徴 NEC

Problem Description

Tools urgently needed!

Not many contributions yet!

Given

- A set of applications
- Scenarios on how these applications will be used
- A set of candidate architectures comprising
  - (Possibly heterogeneous) processors
  - (Possibly heterogeneous) communication architectures
  - Possible scheduling policies

Find

- A mapping of applications to processors
- Appropriate scheduling techniques (if not fixed)
- A target architecture (if DSE is included)

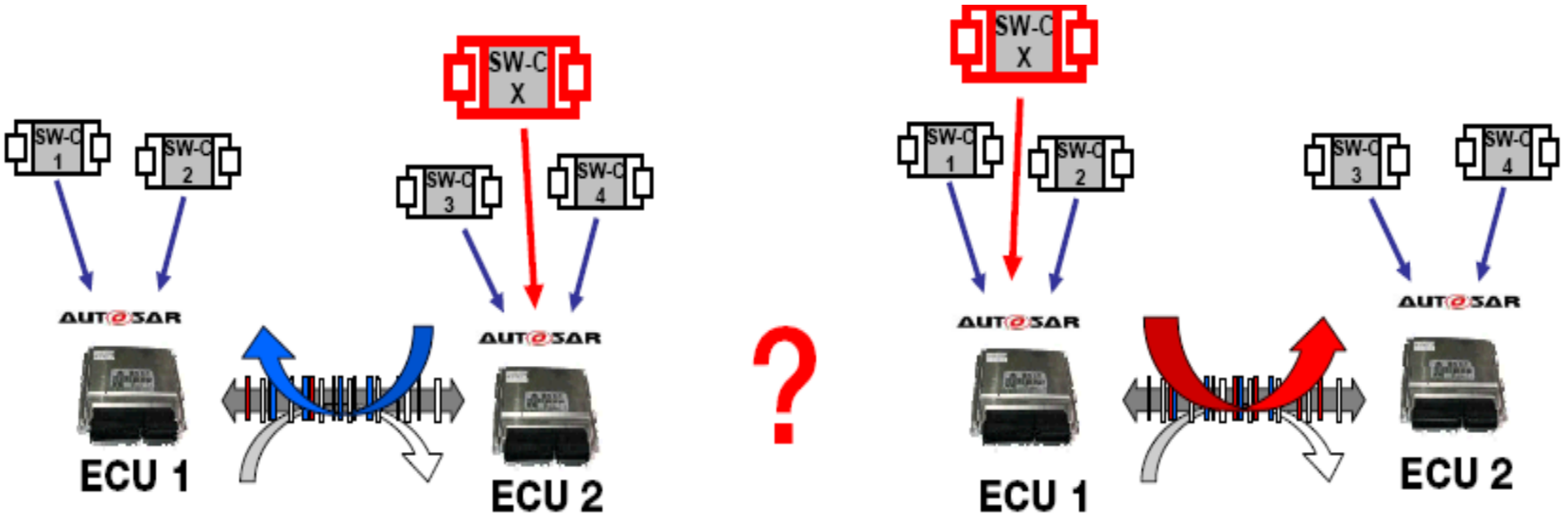
Objectives

- Keeping deadlines and/or maximizing performance
- Minimizing cost, energy consumption

Key distinctions between parallel processing for embedded applications and PC-like systems

	Embedded	PC-like
Architectures	Frequently heterogeneous, very compact	Mostly homogeneous, not compact (x86 etc)
x86 compatibility	Not relevant	Very relevant
Architecture fixed?	Sometimes not	Yes
MoCs	C+multiple MoCs (SDF, …)	Mostly von Neumann
Applications	Several concurrent apps.	Mostly single app.
Apps. known at design time	Most, if not all	Only some (WORD, ..)
Objectives	Multiple (energy, size, …)	Average performance dominating
Real-time relevant	Yes (!)	Hardly

Practical problem in automotive design



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

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

Which processor should run the software?

Focus of the ArtistDesign Network

1st Workshop on Mapping Applications
To MPSoCs, Rheinfels castle, June, 2008

- Future architectures of MPSoCs (John Goodacre, ARM)
- SPEA2 (Lothar Thiele, ETHZ)
- Car-Entertainment Applications -> MP Systems (Marco Bekooij, NXP)
- MAPS (Rainer Leupers, Aachen)
- Overview over parallelization techniques (C. Lengauer, U. Passau)
- DSE of Heterogeneous MPSoCs (Ristau, Fettweis, TU Dresden)
- Daedalus (Ed Deprettere, U. Leiden)
- Mapping to the CELL processor (U. Bologna and others)
- Timing analysis issues (various)

Programme and slides: http://www.artist-embedded.org/artist/-Mapping-of-Applications-to-MPSoCs-.html

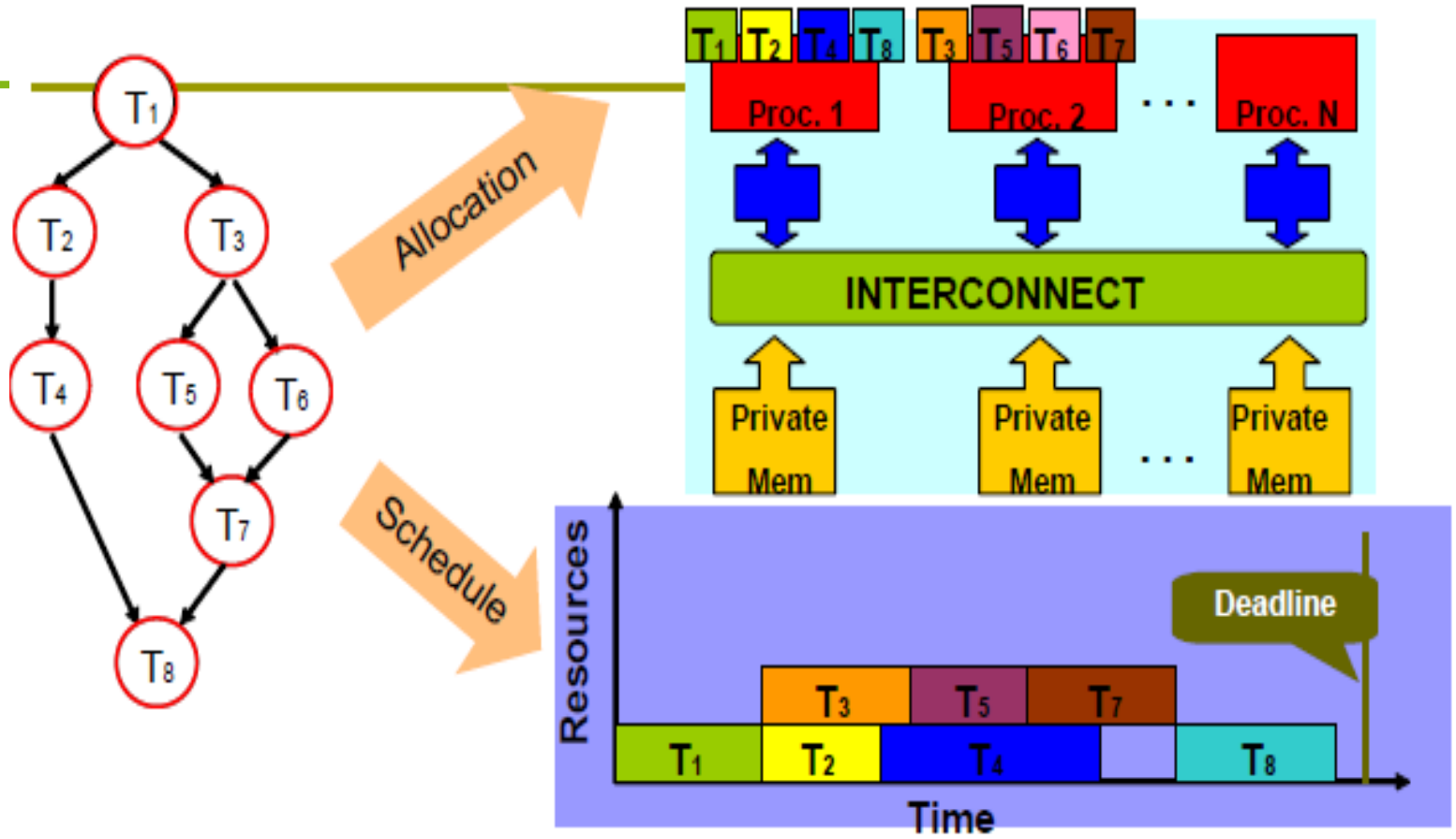
Related Work

- Scheduling theory:
  Provides insight for the mapping *task* $\rightarrow$ *start times*
- Hardware/software partitioning:
  Can be applied if it supports multiple processors
- High performance computing (HPC)
  Automatic parallelization, but only for
  - single applications,
  - fixed architectures,
  - no support for scheduling,
  - memory and communication model usually different
- High-level synthesis
  Provides useful terms like scheduling, allocation, assignment
- Optimization theory

A Simple Classification

Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given model	Map to CELL, HOPES, ETHAM	COOL codesign tool; EXPO/SPEA2
Auto-parallelizing	Franke, O'Boyle et al., Mnemee MAPS	Daedalus

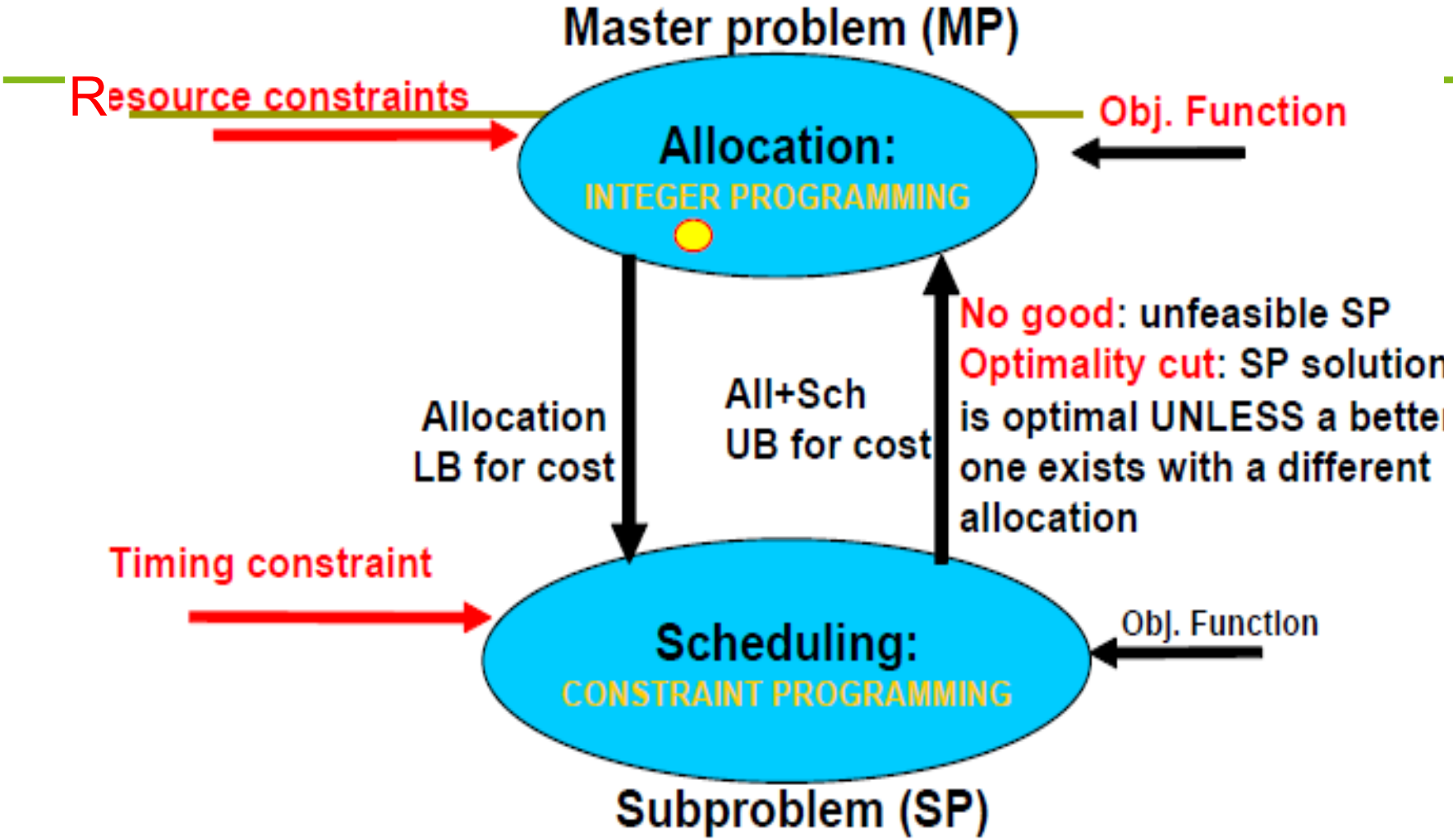
A fixed architecture approach: Map → CELL



- 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

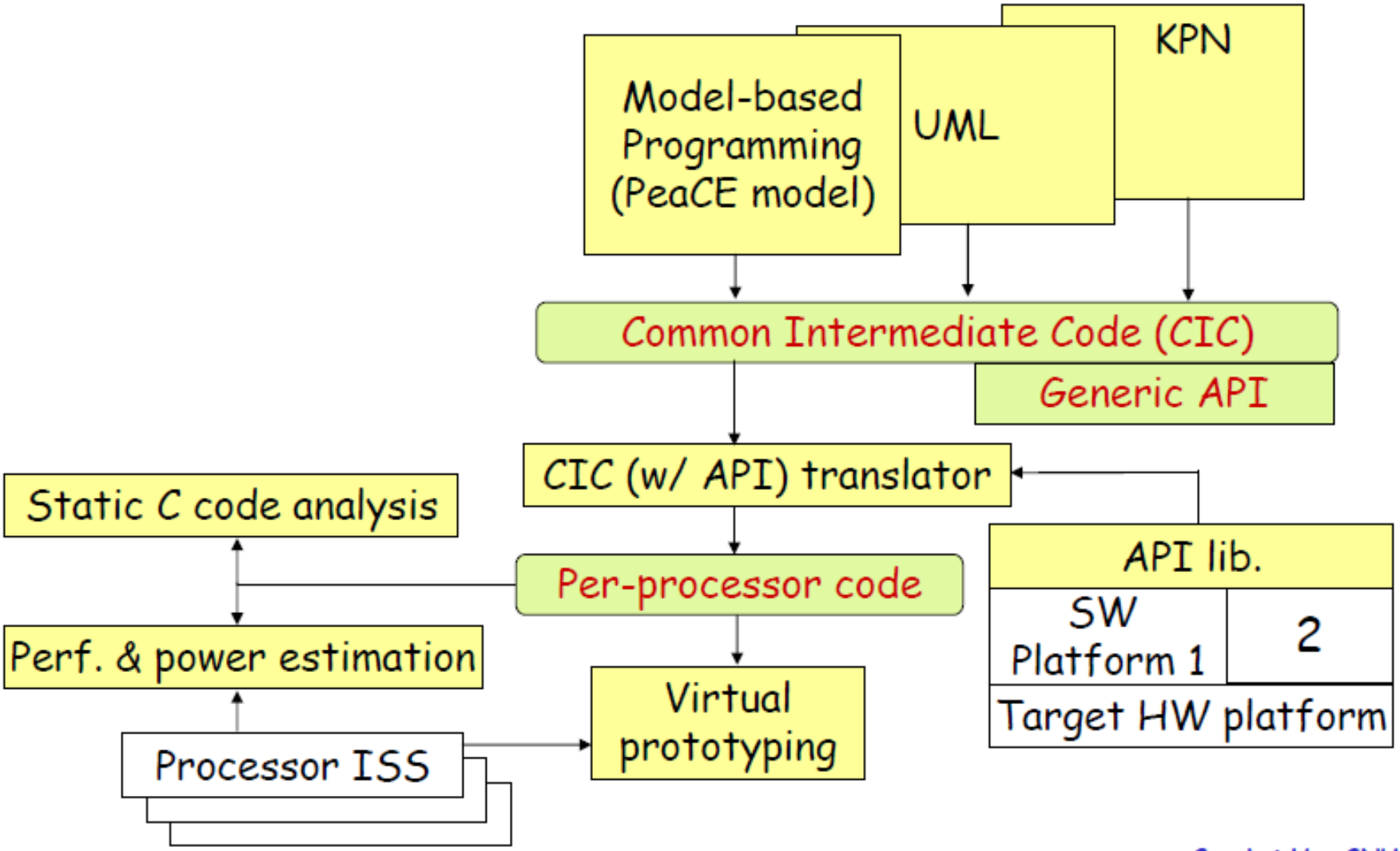
Partitioning into Allocation and Scheduling



Iterations stop when MP becomes unfeasible!

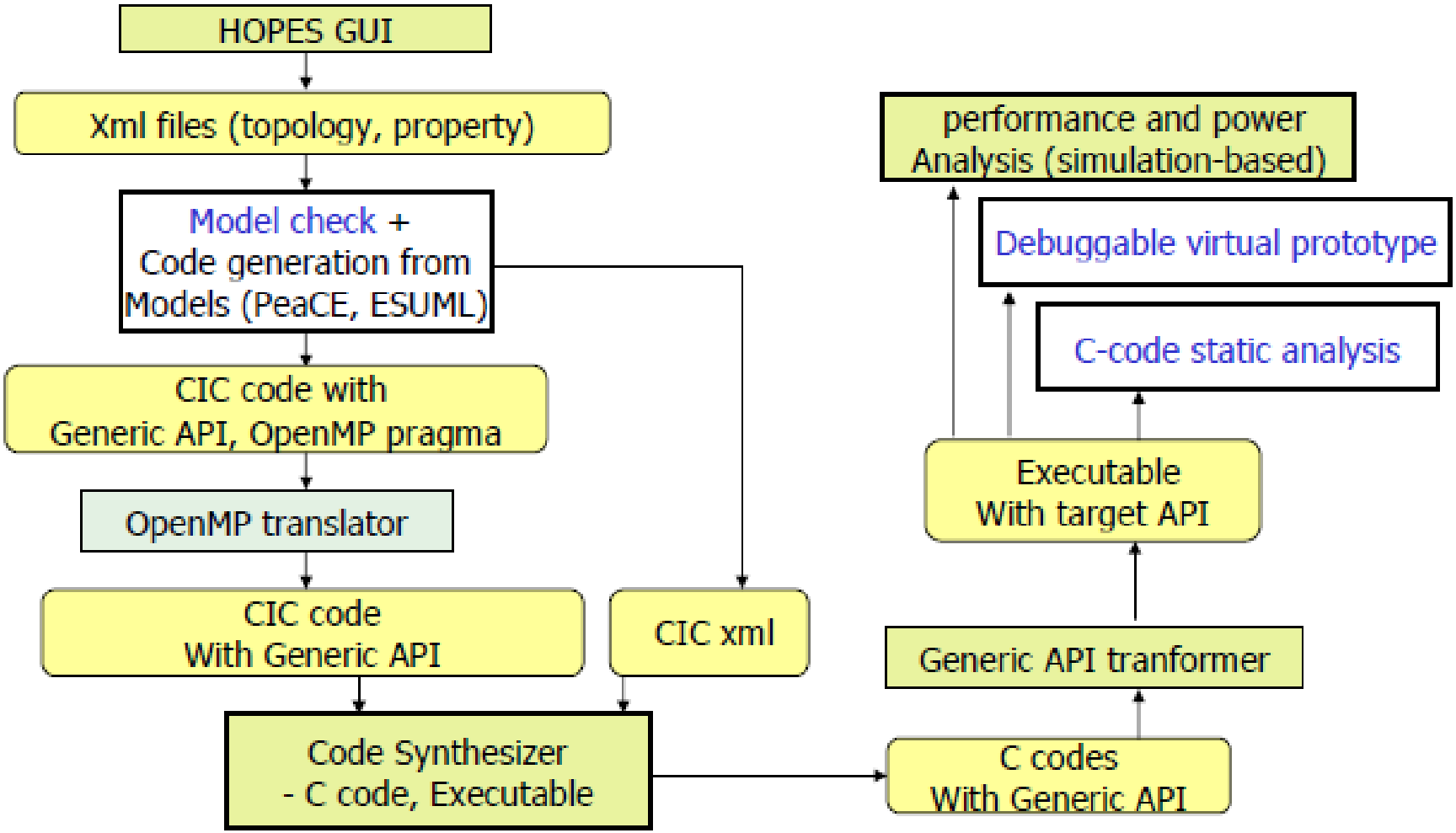
HOPES Proposal

HOPES



Design Flow

HOPES



Proposed design flow

HOPES

- Model-based programming
  - PeaCE model (dataflow + FSM + task model)
  - ESUML (embedded system UML) model
- CIC (Common Intermediate Code)
  - OpenMP pragma + generic API
- Static Analysis
  - Buffer overrun, memory leak, null dereference, stack size
- Virtual prototyping
  - Performance and power estimation
  - with source-level debugging capability

PeaCE Project

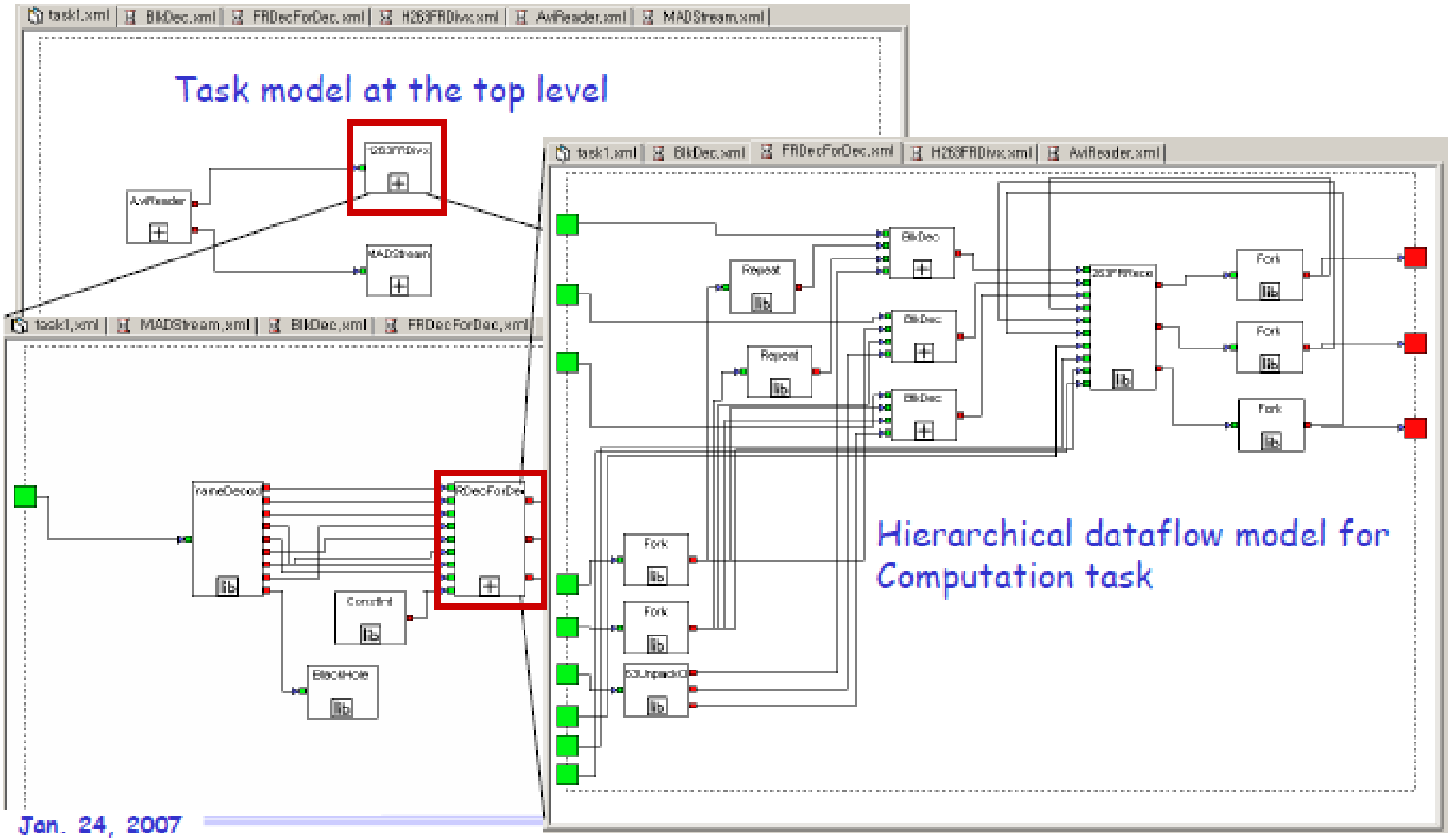
HOPES

- PeaCE
  - **P**tolemy **e**xtension **a**s a **C**odesign **E**nvironment based on Ptolemy classic
  - open-source research platform
  - Officially released in DAC 2005. (version 1.0)

- PeaCE home page
  - http://peace.snu.ac.kr/research/peace

Divx Player: Top Model

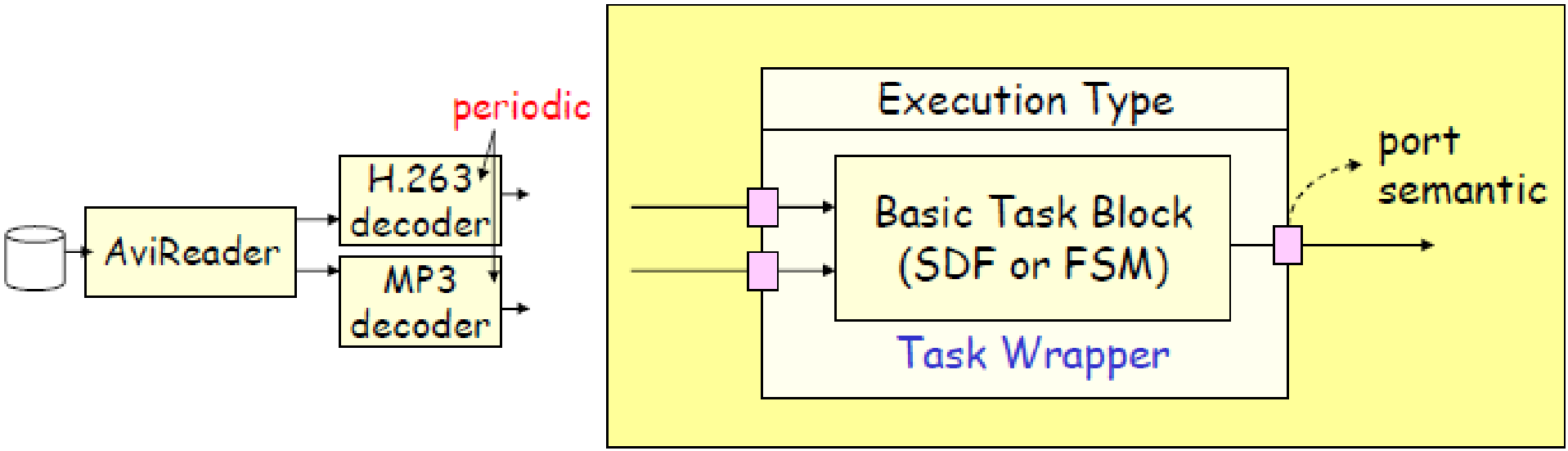
HOPES



Task Model

HOPES

- Task execution semantics
  - periodic, sporadic, functional
- Port properties
  - data rate : static or dynamic
  - data size : static or variable
  - port type : queue or buffer



CIC Code Generation

HOPES

- Application tasks
  - Task in task-level specification model → task_name.cic
- Generic API
- Partitioning considering data parallelism
  - openMP programming

```
void h263decoder_go (void) {
        ...
         l = MQ_RECEIVE("mq0", (char *)(ld_106->rdbfr), 2048);
         ...
         # pragma omp parallel for
         for(i=0; i<99; i++) {
               //thread_main()
            ....
         }
          // display the decode frame
            dither(frame);
}
```

DivX player CIC xml (Hardware)

HOPES

```
<?xml version="1.0" ?>
 <CIC_XML>
  <hardware>
   <processor name="arm926ej-s0">
    <index>0</index>
    <localMem name="lmap0">
     <addr>0x0</addr>
     <size>0x10000</size>  // 64KB
    </localMem>
    <sharedMem name="shmap0">
     <addr>0x10000</size>
     <size>0x40000</size>  // 256KB
     <sharedWith>1</sharedWith>
    </sharedMem>
    <OS>
     <support>TRUE</support>
    </OS>
   </processor>
```

```
   <processor name="arm926ej-s1">
    <index>0</index>
    <localMem name="lmap1">
     <addr>0x0</addr>
     <size>0x20000</size>  // 128KB
    </localMem>
    <sharedMem name="shmap1">
     <addr>0x20000</size>
     <size>0x40000</size>  // 256KB
     <sharedWith>0</sharedWith>
    </sharedMem>
    <OS>
     <support>TRUE</support>
    </OS>
   </processor>
  </hardware>
```

DivX player CIC xml (Constraint)

HOPES

```
<constraints>
 <memory>16MB</memory>
 <power>50mWatt</power>
  <mode name="default">
  <task name="AviReaderI0">
   <period>120000</period>
   <deadline>120000</deadline>
   <priority>0</priority>
    <subtask name="arm926ej-s0">
    <execTime>186</execTime>
    </subtask>
   </task>
   <task name="H263FRDivxI3">
   <period>120000</period>
   <deadline>120000</deadline>
   <priority>0</priority>
    <subtask name="arm926ej-s0">
    <execTime>5035</execTime>
    </subtask>
   </task>
   <task name="MADStreamI5">
   <period>120000</period>
   <deadline>120000</deadline>
   <priority>0</priority>
    <subtask name="arm926ej-s0">
    <execTime>13</execTime>
    </subtask>
  </task>
 </mode>
</constraints>
```

Related work

ETHAM: An Energy-Aware Task Allocation Algorithm for Heterogeneous Multiprocessor (Chang et al. – DAC'08)

- Input: Directed Acyclic Task Graph (DAG)
- Combines mapping, scheduling + DVS utilization in 1 step
- Builds tree with all possible solutions
- Uses Ant Colony Optimizations (ACO) on this tree

Temperature Aware Task Scheduling in MPSoCs (Coskun et al. – DATE'07)

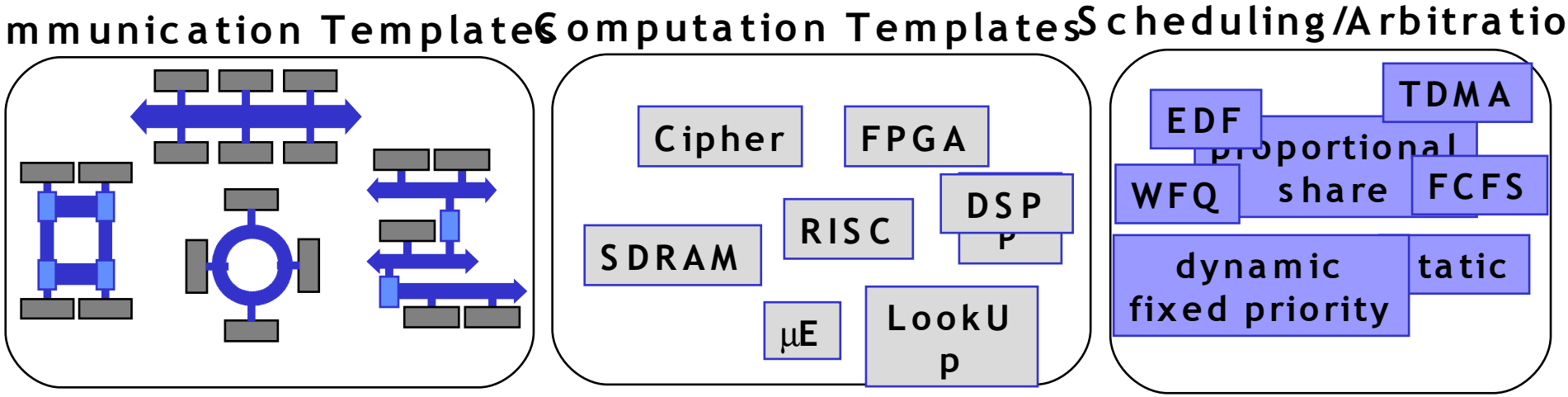
- Higher reliability through temperature optimized dynamic OS scheduling
- Strategy is called "Adaptive Random"
- Considers history of core's temperate for scheduling

Session on Task Mapping @ DATE 2009

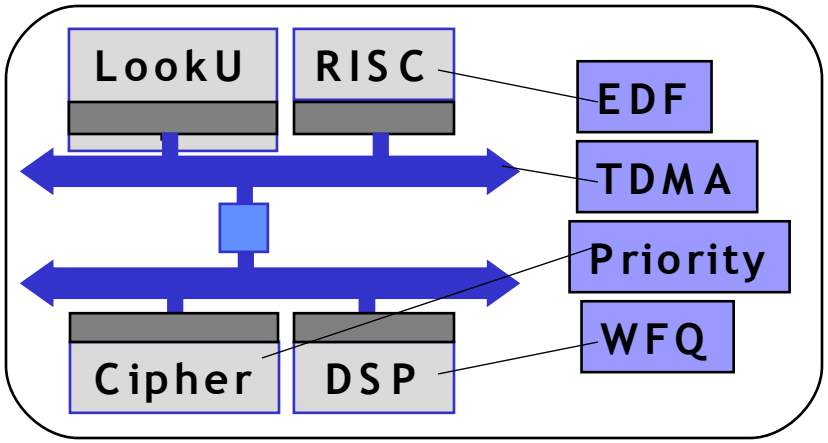
A Simple Classification

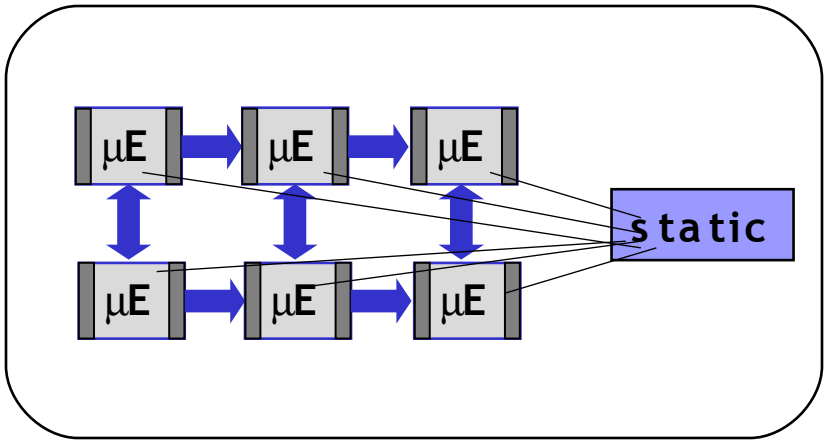
Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given model	Map to CELL, HOPES, ETHAM	COOL codesign tool; EXPO/SPEA2
Auto-parallelizing	Franke, O'Boyle et al. Mnemee MAPS	Daedalus

Design Space



Which architecture is better suited for our application?





Target Platform

Communication

- *micro-network* on chip for synchronization and data exchange consisting of busses, routers, drivers
- some critical *issues*: topology, switching strategies (packet, circuit), routing strategies (static – reconfigurable – dynamic), arbitration policies (dynamic, TDM, CDMA, fixed priority)
- *challenges*: heterogeneous components and requirements, compose network that matches the traffic characteristics of a given application (domain)

Mapping Scenario: Overview

Given

1. specification of the task structure (**task model**) = for each flow the corresponding tasks to be executed
2. different usage scenarios (**flow model**)

Sought

processor implementation (**resource model**) =
architecture* + task mapping + scheduling

Objectives:

1. maximize performance
2. minimize cost

Subject to:

1. memory constraints
2. delay constraints

} (**performance model**)

***: 2 cases:**

1. fixed architecture
2. architecture to be designed

Example: System Synthesis

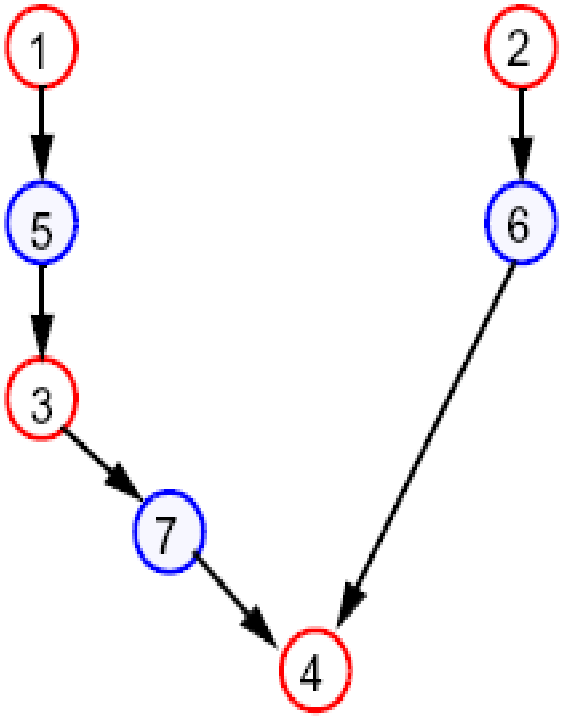
Given:

Goal:

Objectives: cost, latency, power consumption

Basic Model – Problem Graph

Problem graph $G_P(V_P,E_P)$:

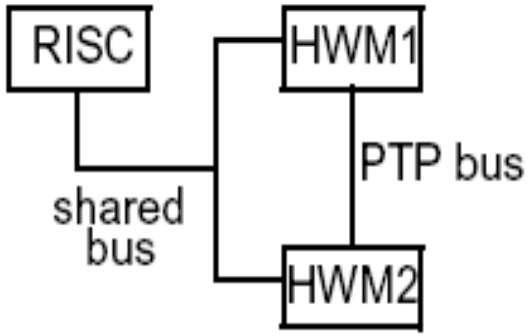


Interpretation:

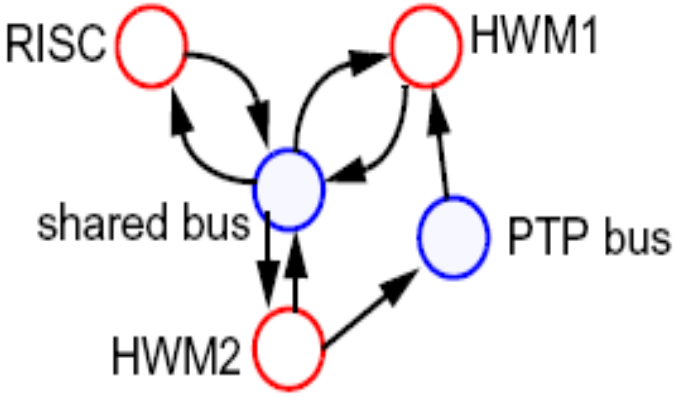
- V_P consists of functional nodes V_P^f (task, procedure) and communication nodes V_P^c .
- E_P represent data dependencies

Basic Model – architecture graph

Architecture graph $G_A(V_A,E_A)$:



Architecture

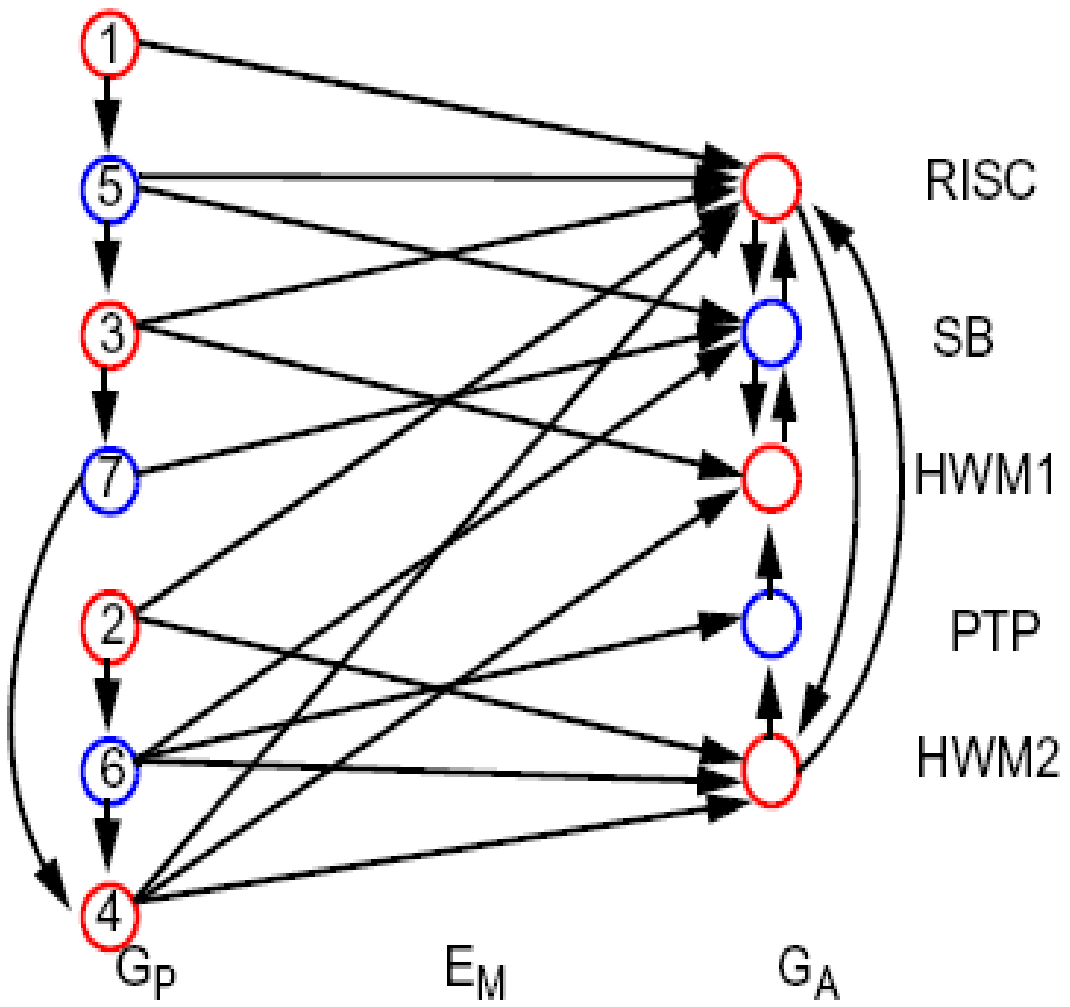


Architecture graph

- V_A consists of functional resources V_A^f (RISC, ASIC) and bus resources V_A^c. These components are potentially allocatable.
- E_A model directed communication.

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



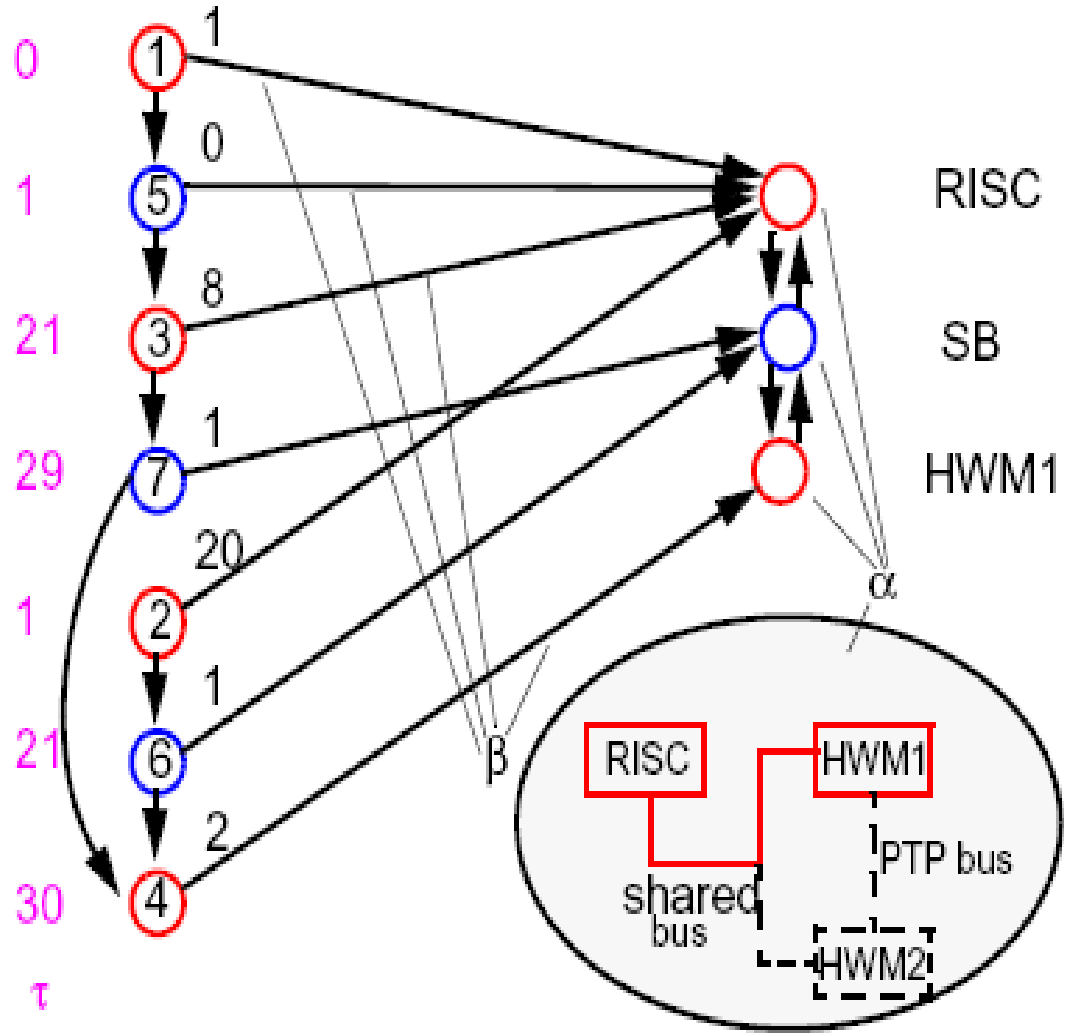
Basic model - synthesis

Three main tasks of synthesis:

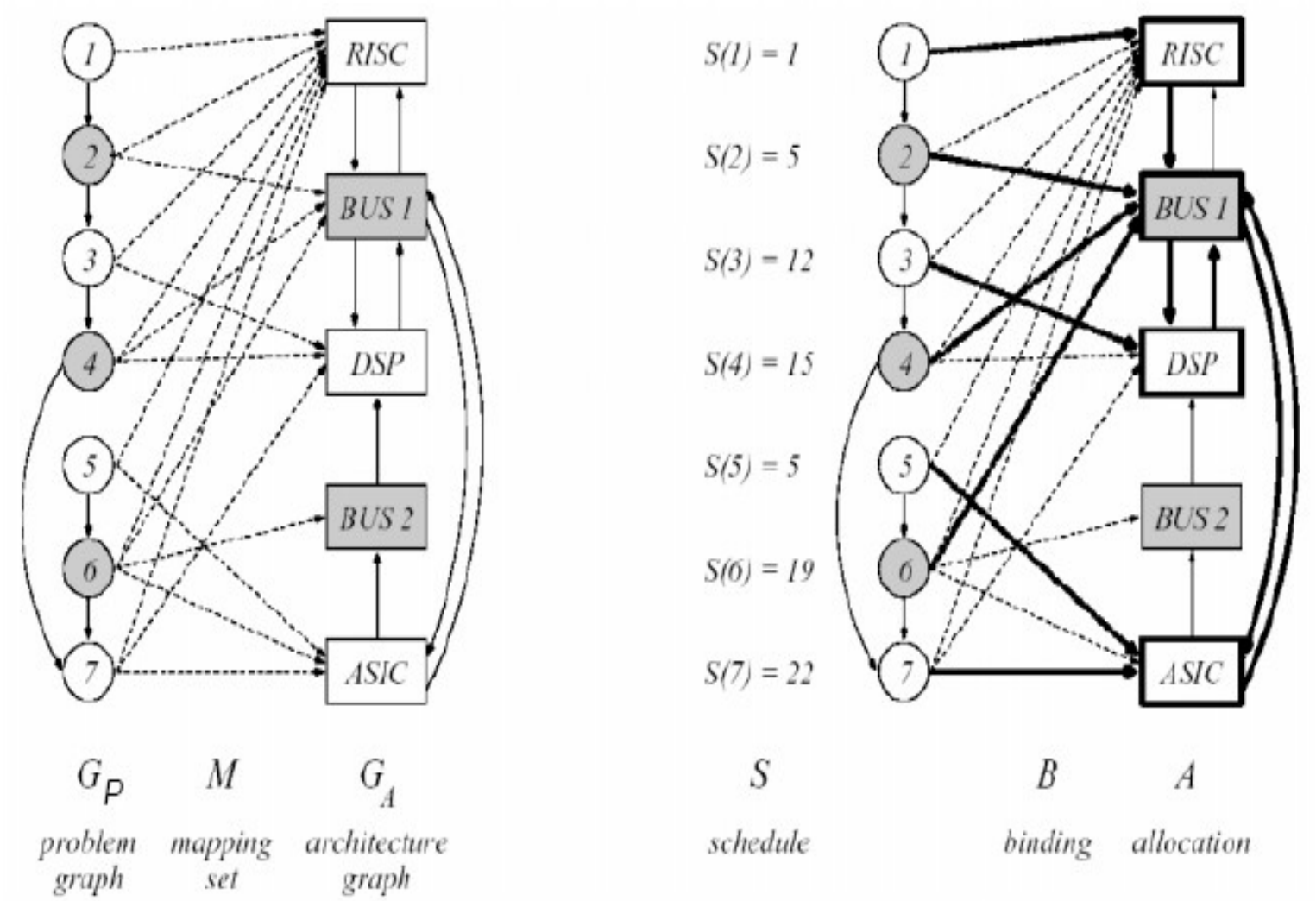
- Allocation α is a subset of V_A.
- Binding β is a subset of E_M, i.e., a mapping of functional nodes of V_P onto resource nodes of V_A.
- Schedule τ is a function that assigns a number (start time) to each functional node.

Basic model - implementation

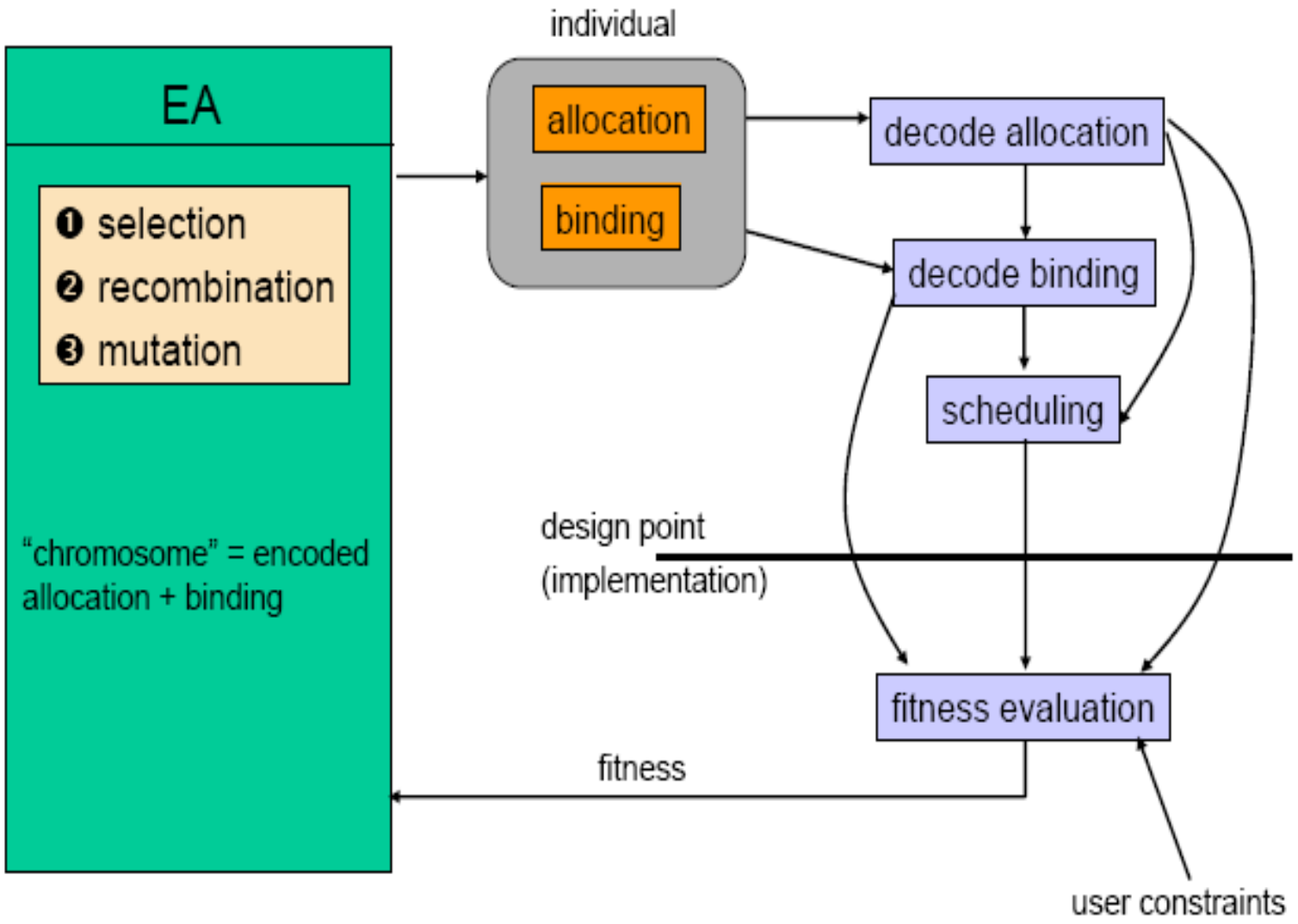
Definition: Given a specification graph G_S an implementation is a triple (α,β,τ), where α is a feasible allocation, β is a feasible binding, and τ is a schedule.



Example


1
2
3
4
5
6
7
RISC
BUS 1
DSP
BUS 2
ASIC
S(1) = 1
S(2) = 5
S(3) = 12
S(4) = 15
S(5) = 5
S(6) = 19
S(7) = 22
G_P
M
G_A
problem graph
mapping set
architecture graph
S
schedule
B
binding
A
allocation

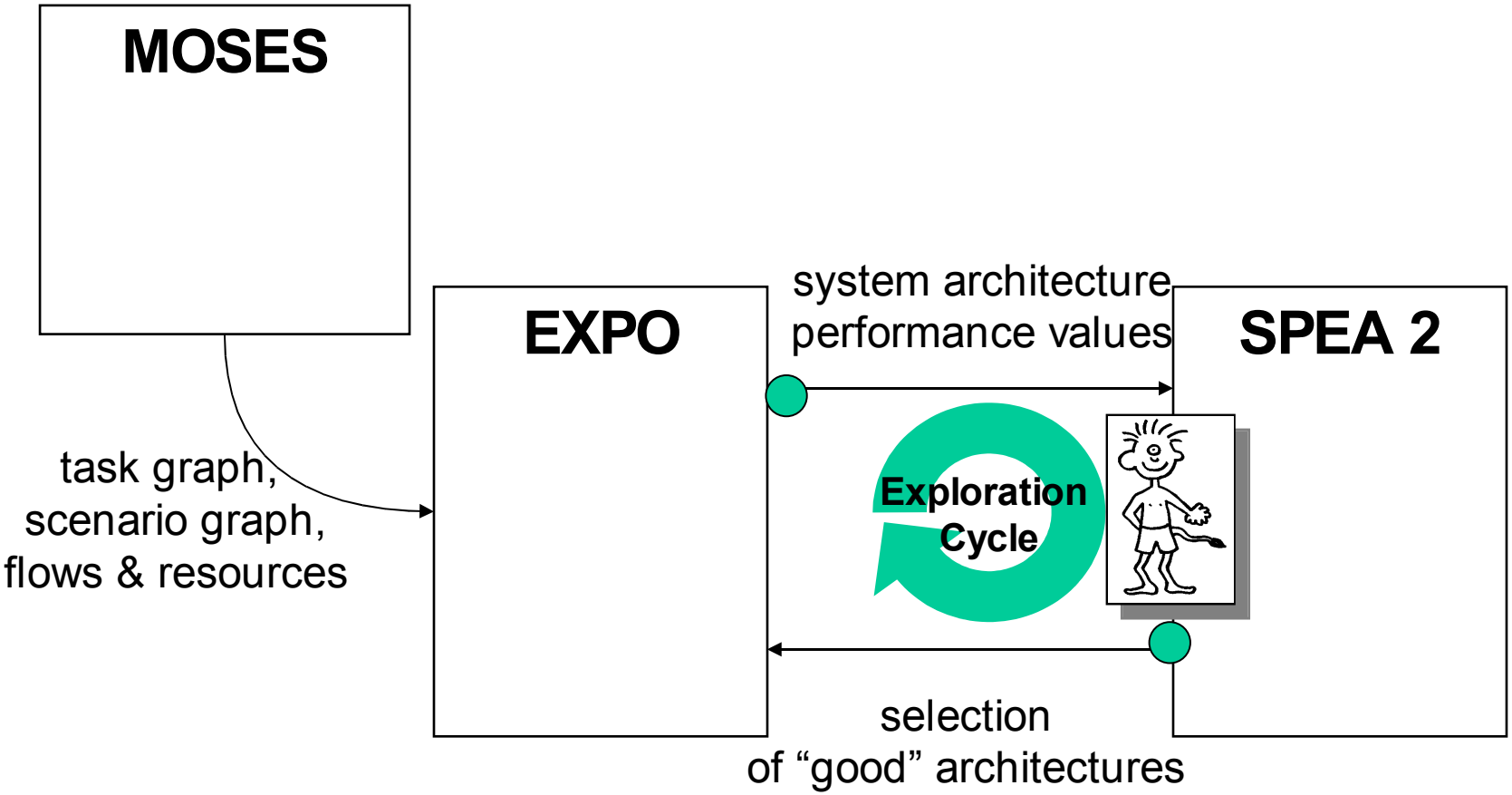
Evolutionary Algorithms for Design Space Exploration (DSE)



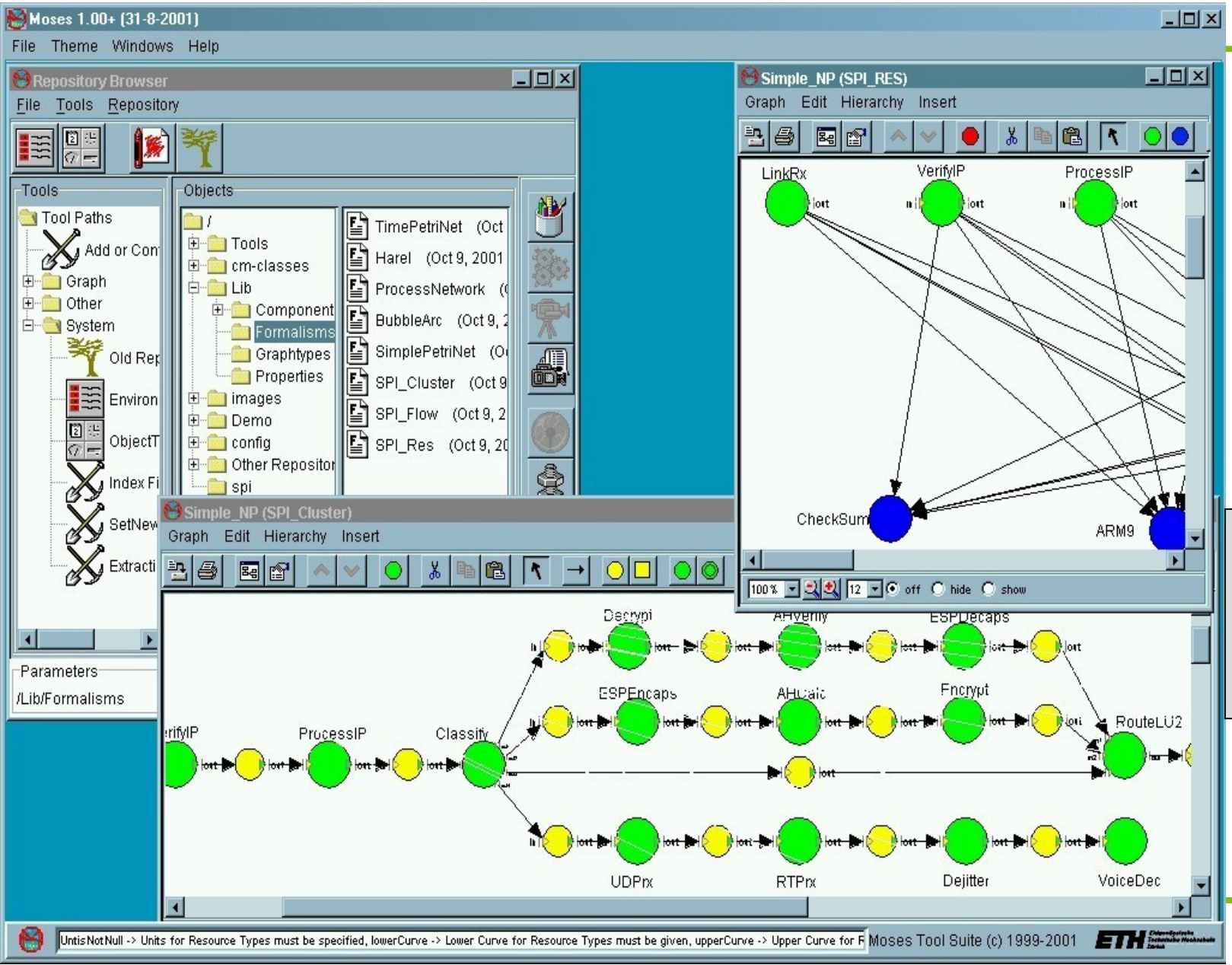
Challenges

- 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)

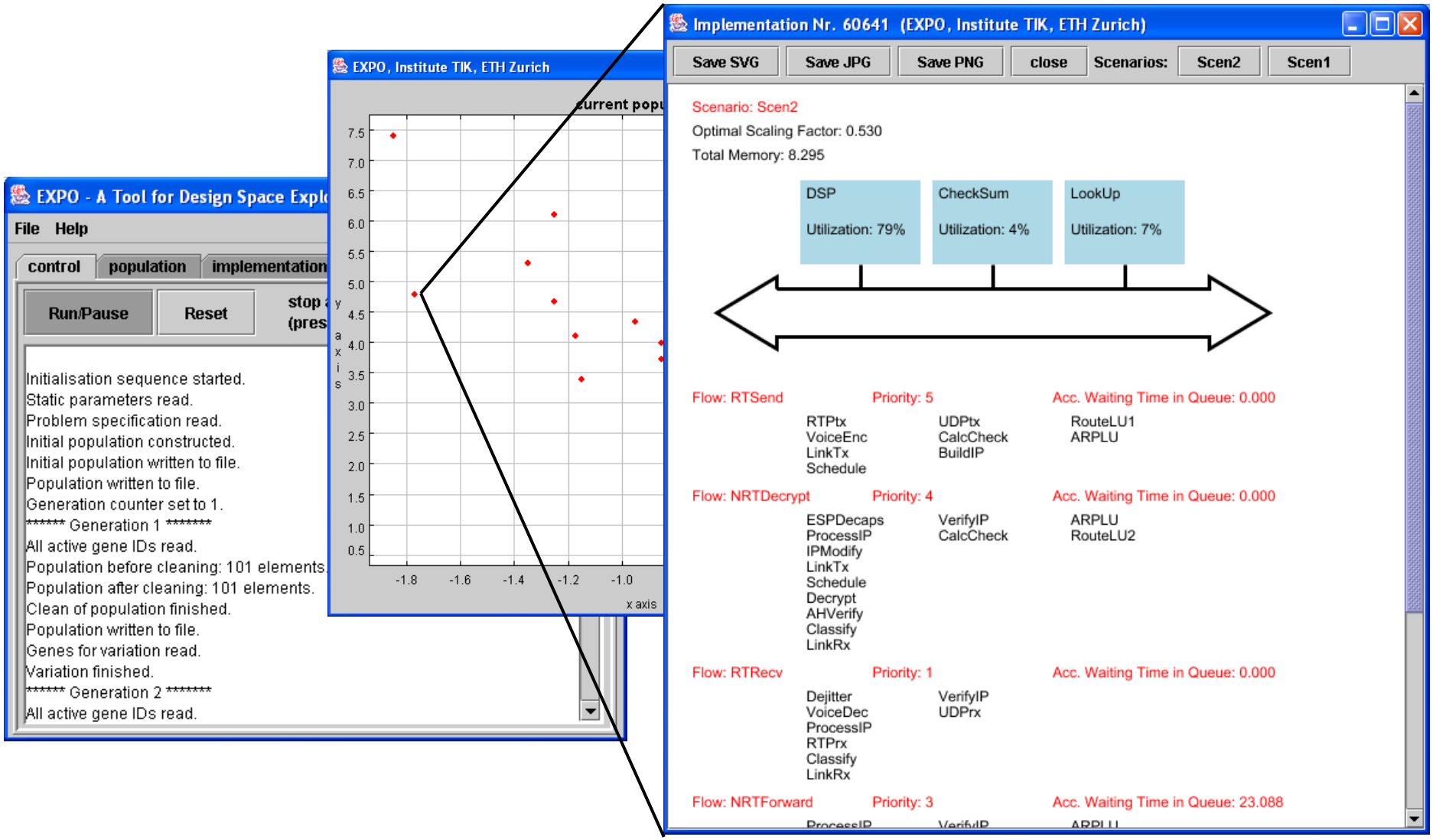


EXPO – Tool architecture (2)



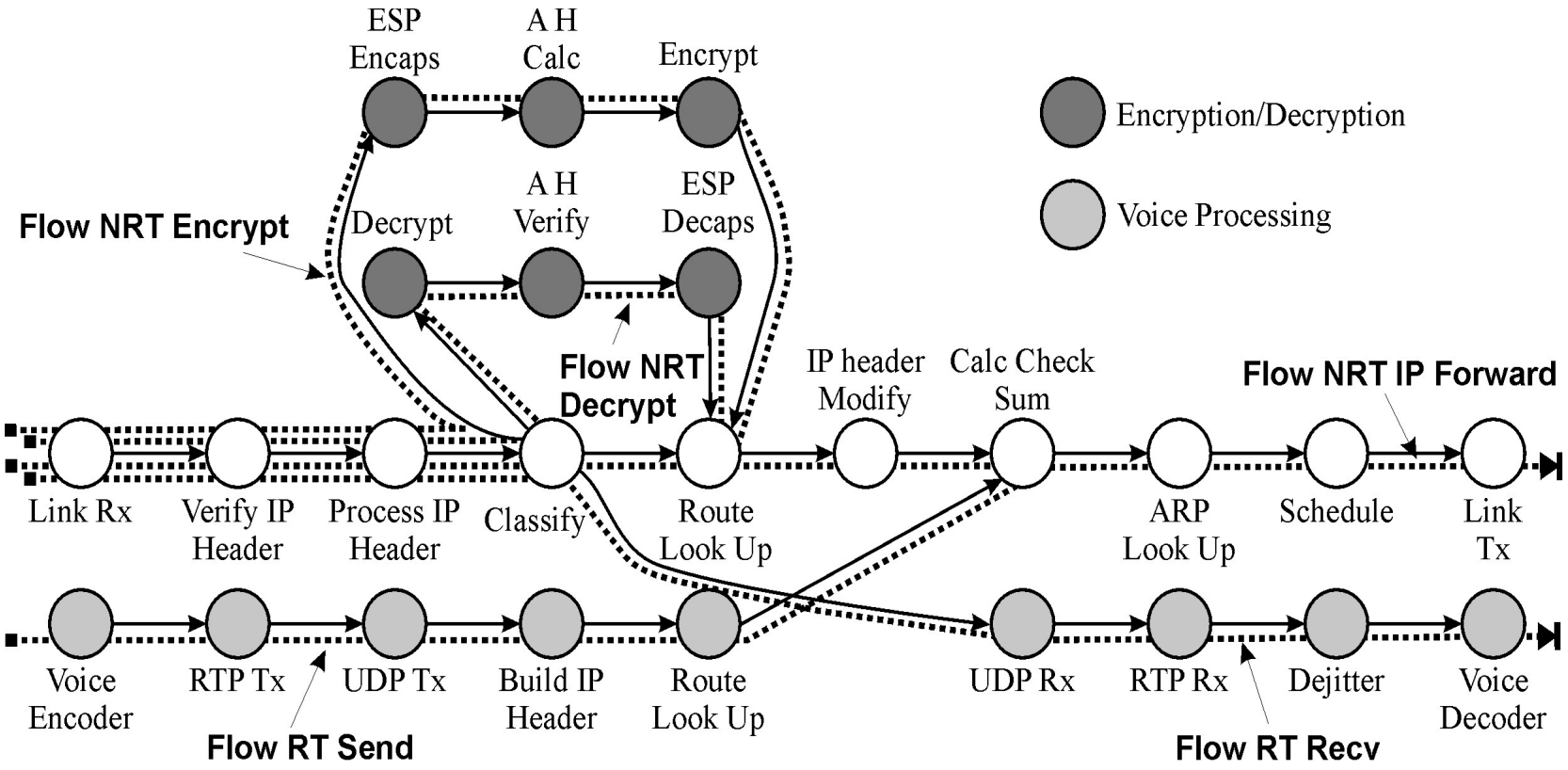
Tool available online: http://www.tik.ee.ethz.ch/expo/expo.html

EXPO – Tool (3)


EXPO - A Tool for Design Space Explo
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
y axis
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

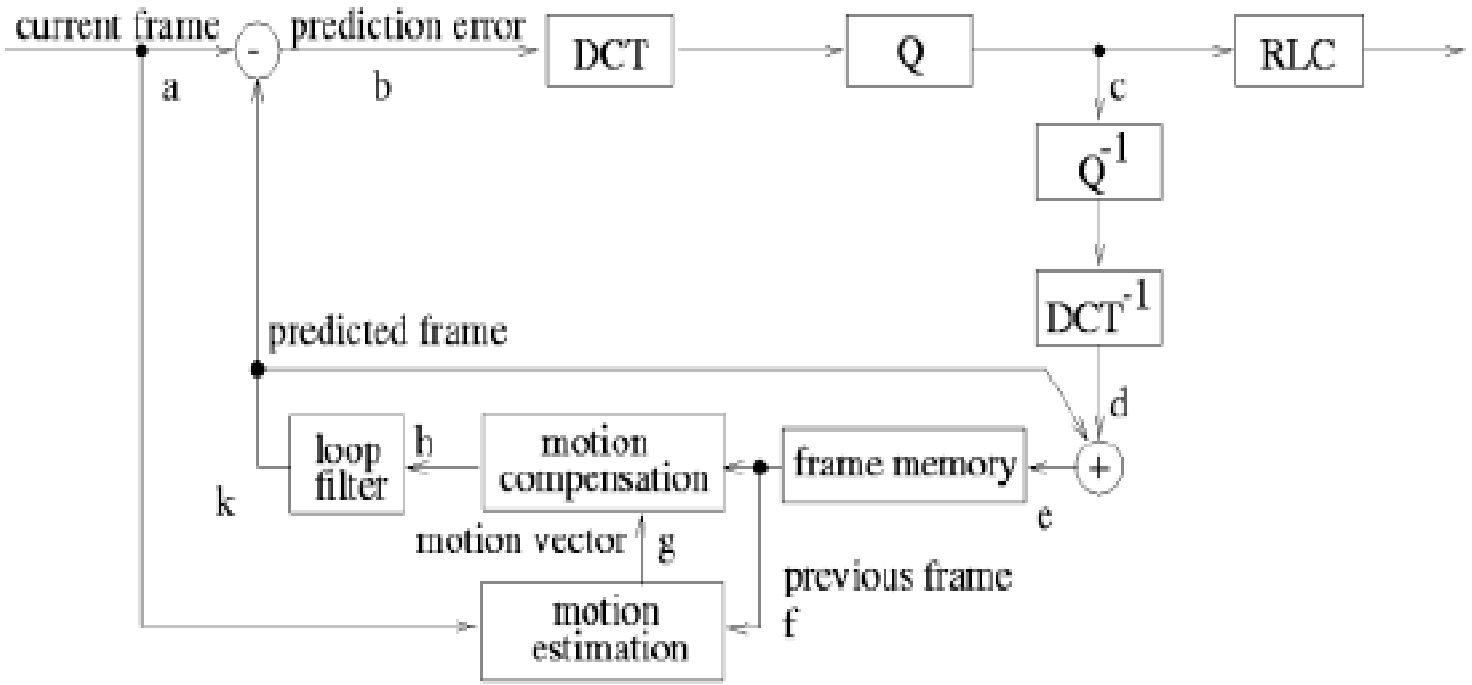
Application Model

Example of a simple stream processing task structure:



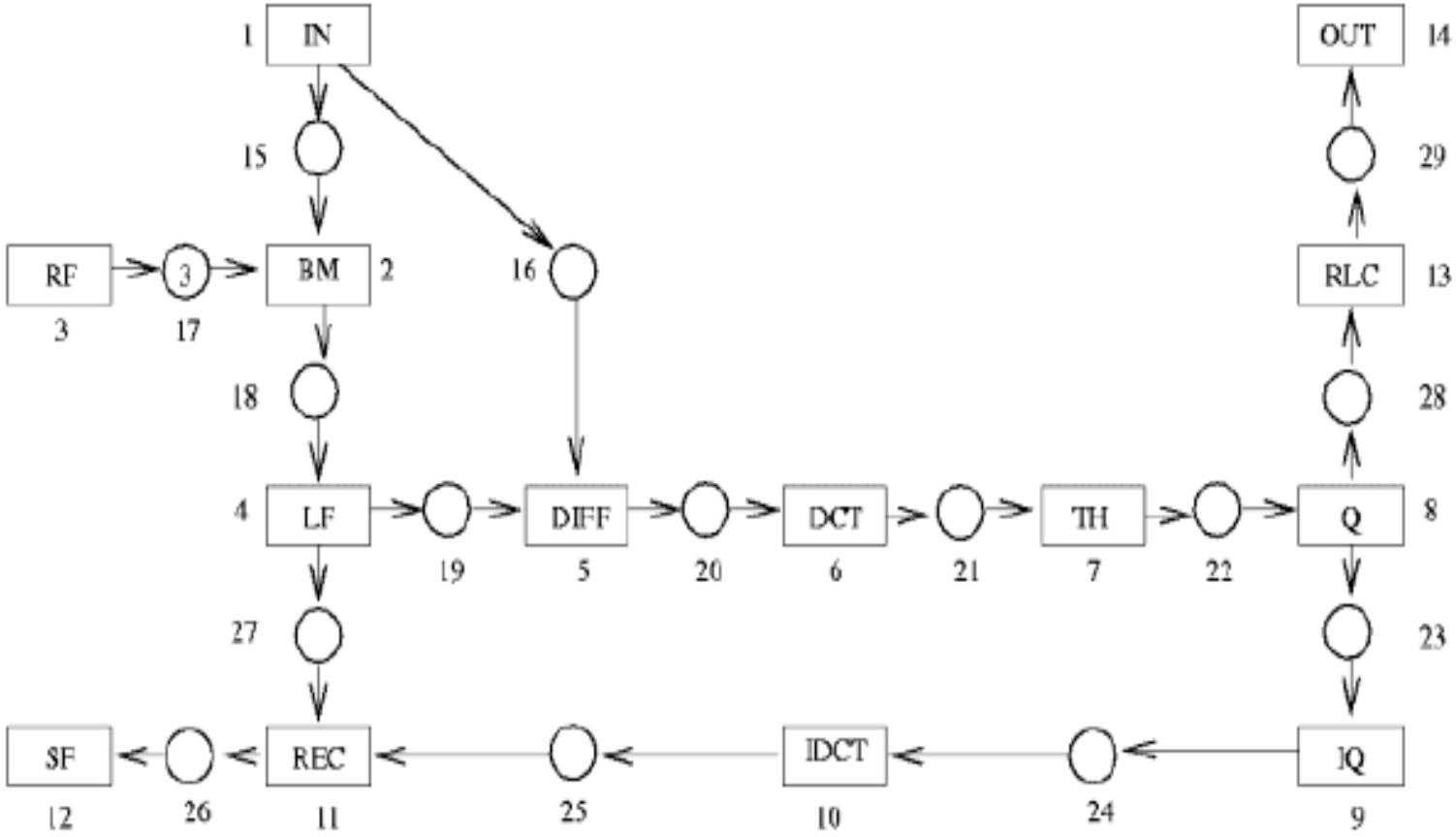
Exploration – Case Study (1)

behavioral specification of a video codec for video compression

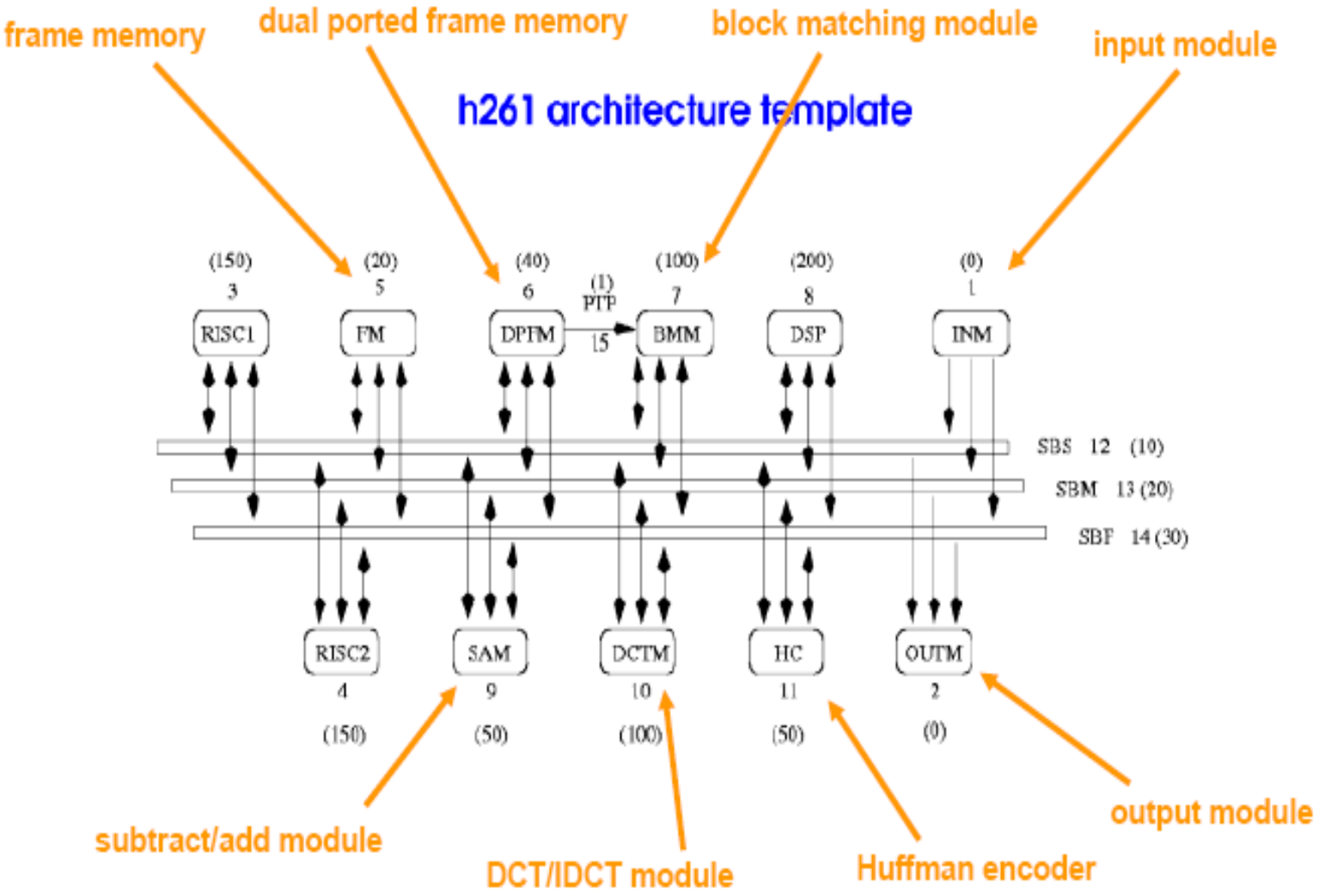


Exploration – Case Study (2)

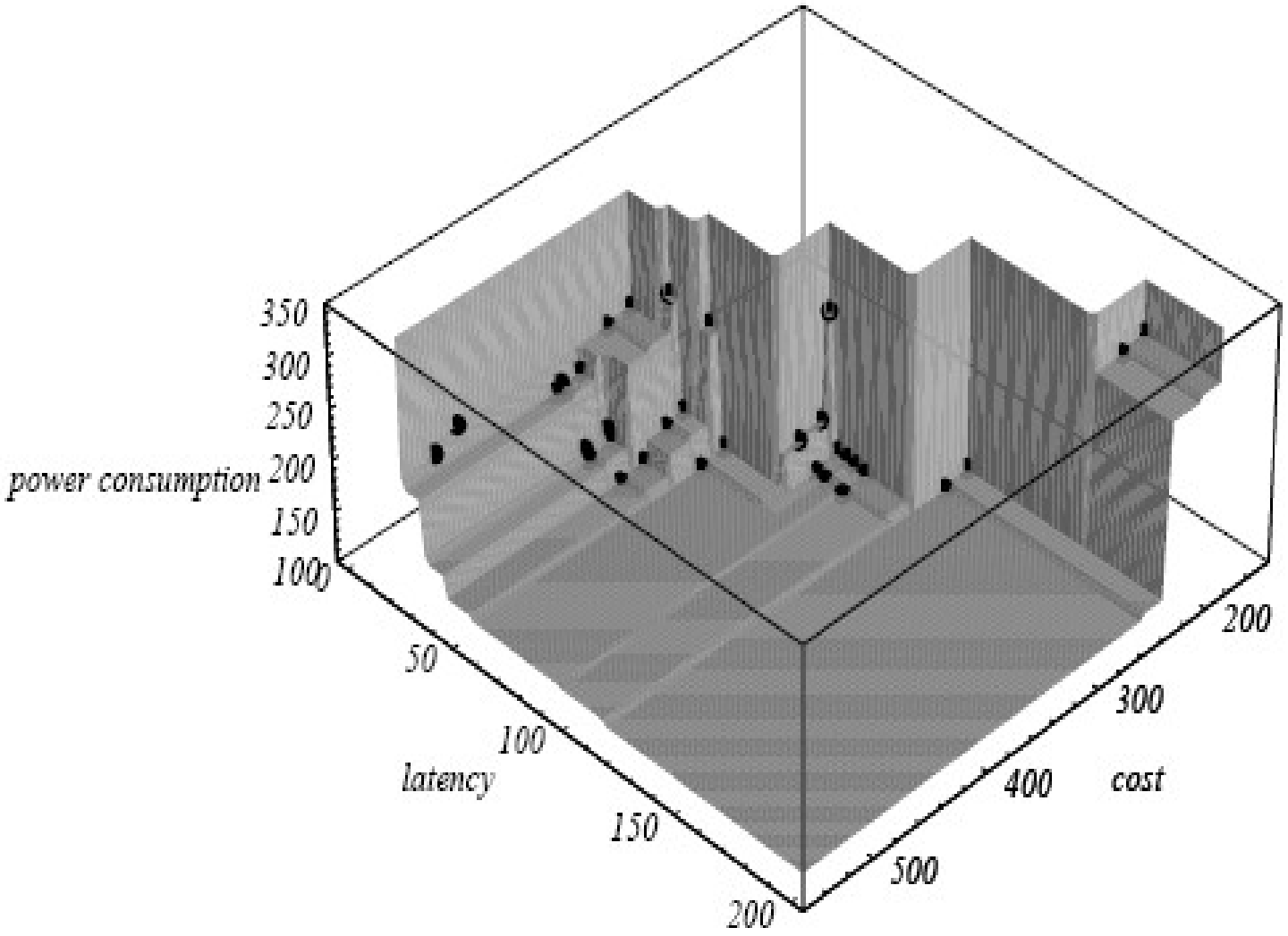
problem graph of the video coder



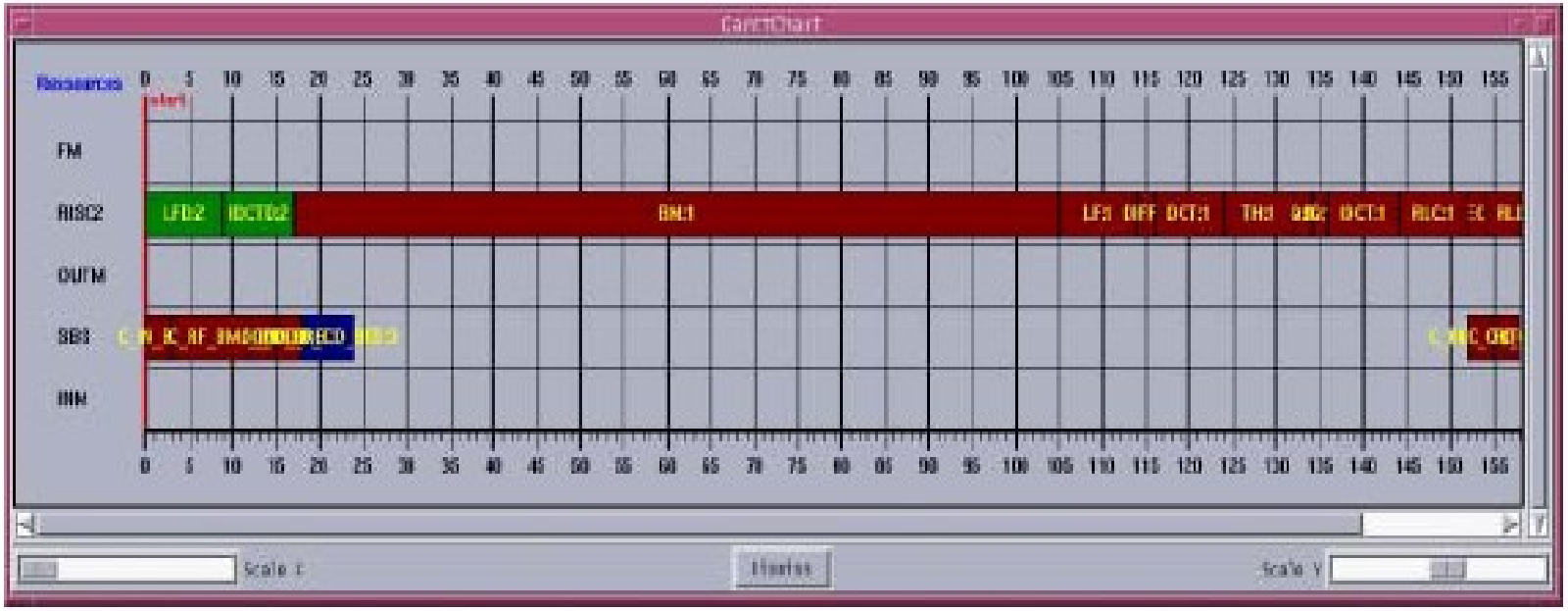
Exploration – Case Study (3)

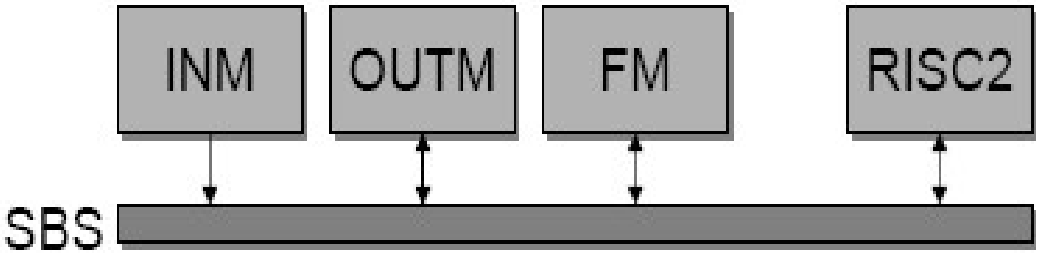


EA Case Study – Design Space

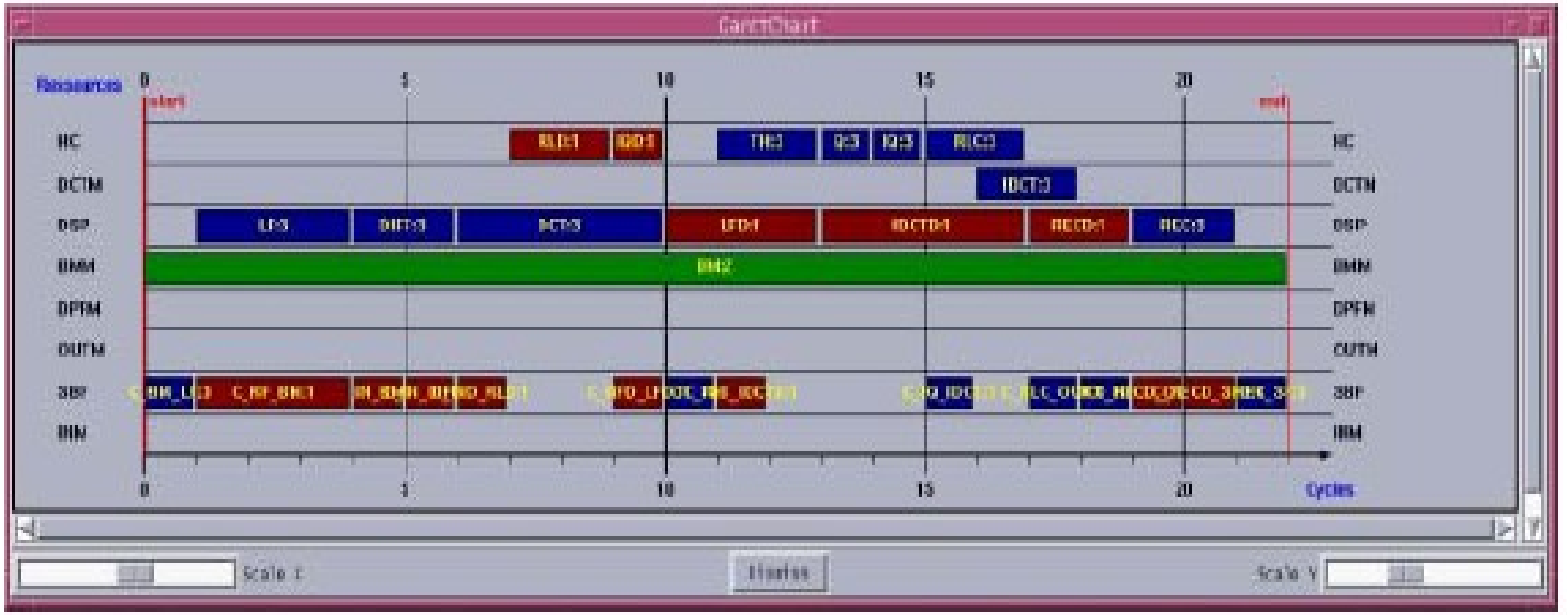


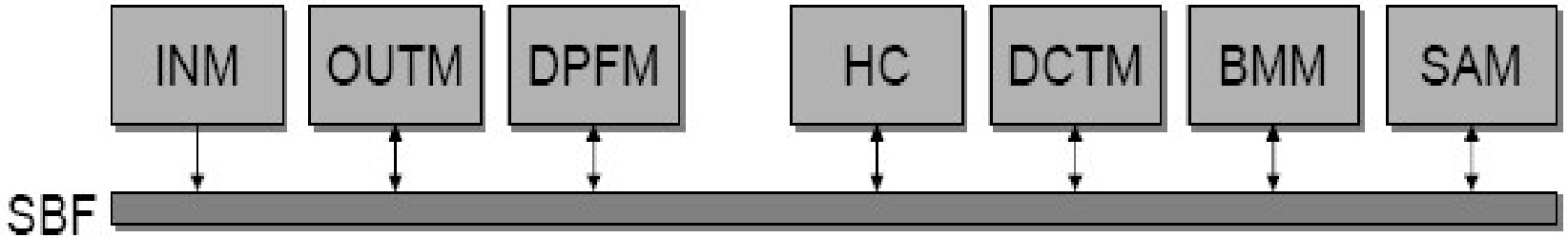
Exploration Case Study – Solution 1



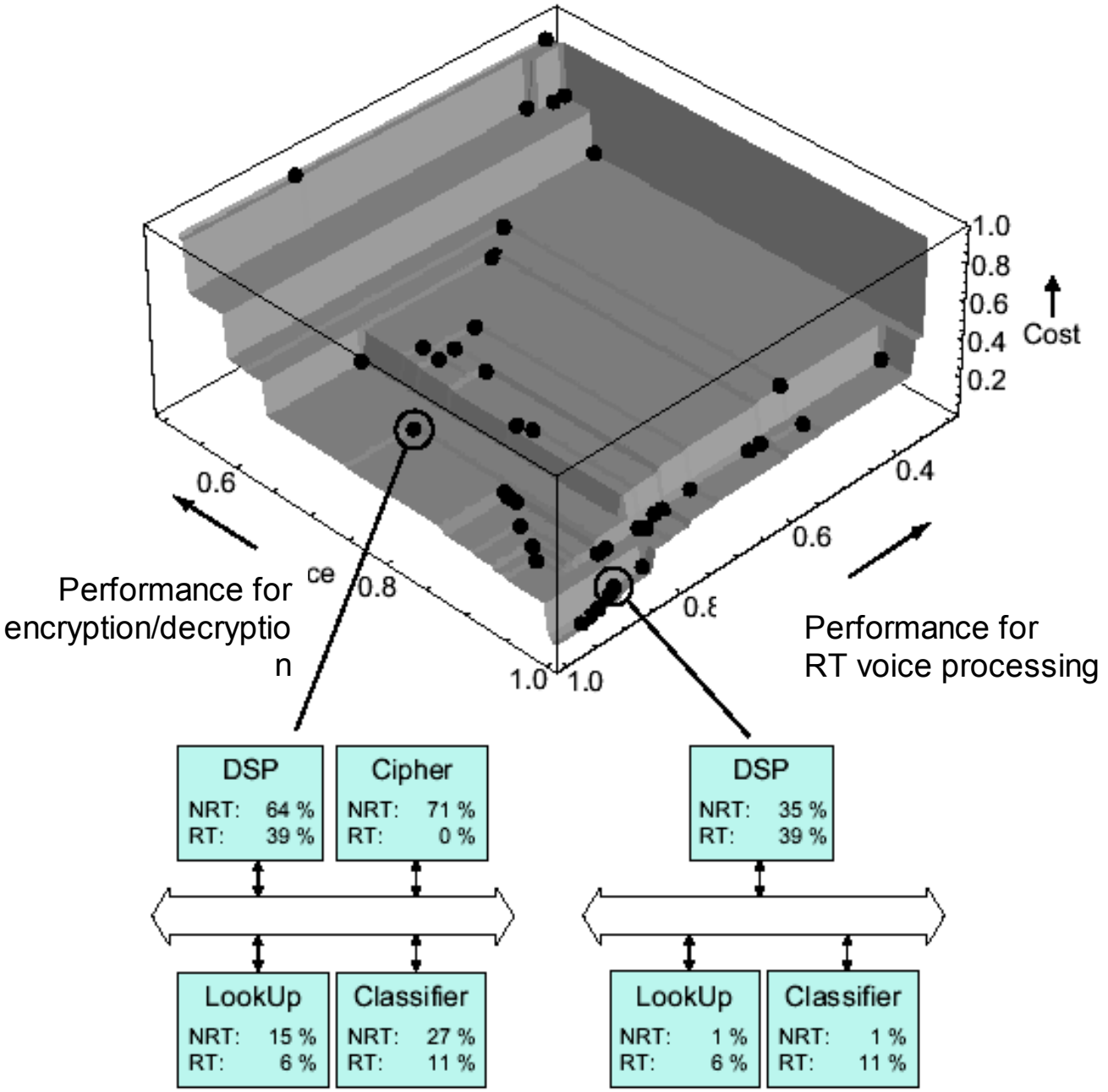


Exploration Case Study – Solution 2

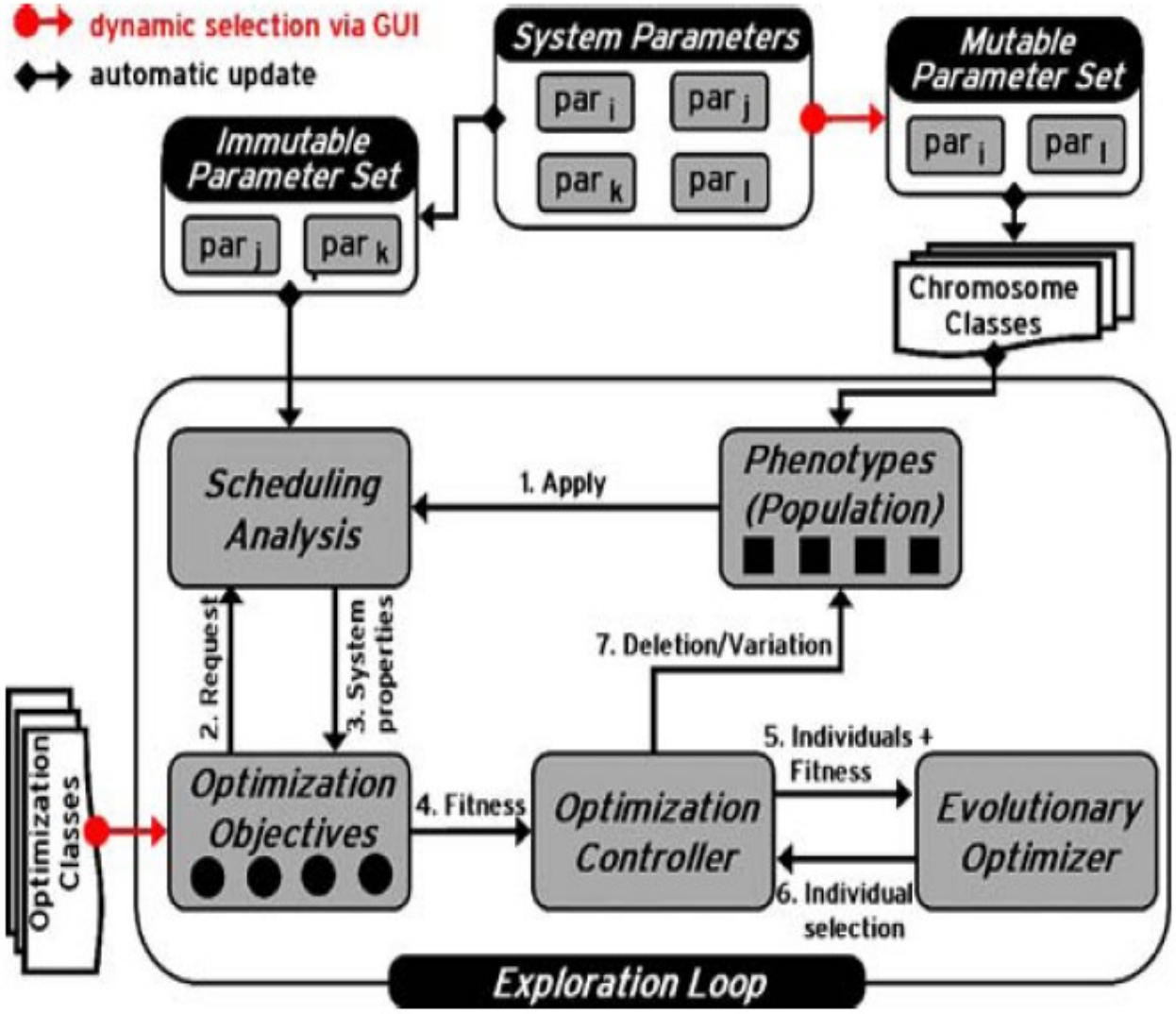




More Results



A 2nd 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]

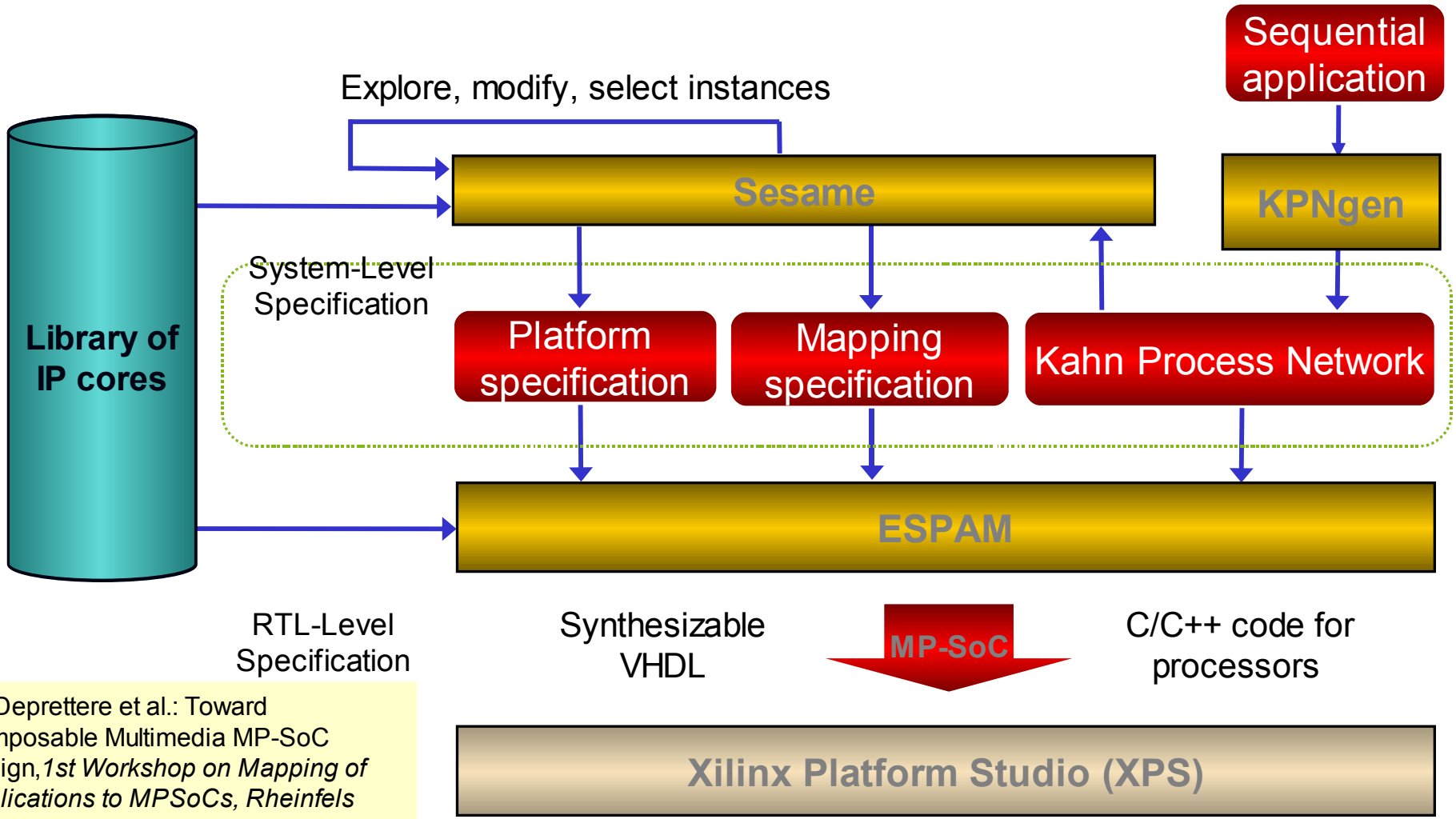
Additional Contributions

- SystemCoDesigner (Teich et al.)
- Compaan (B. Kienhuis et al.)
- Milan (J. David et al.)
- Charmed (S. Bhattacharyya et al.)
- Metropolis (A. Sangiovanni-Vincentelli et al.)

A Simple Classification

Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given model	Map to CELL, HOPES, ETHAM	COOL codesign tool; EXPO/SPEA2
Auto-parallelizing	Franke, O'Boyle et al. Mnemee MAPS	Daedalus

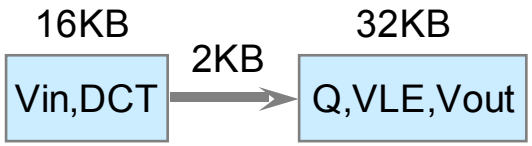
Daedalus Design-flow



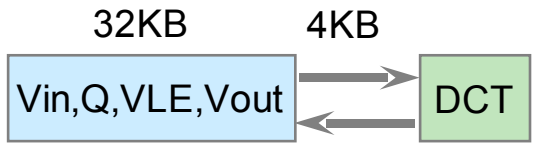
Ed Deprettere et al.: Toward Composable Multimedia MP-SoC Design, *1st Workshop on Mapping of Applications to MPSoCs, Rheinfels Castle*, 2008

JPEG/JPEG200 case study

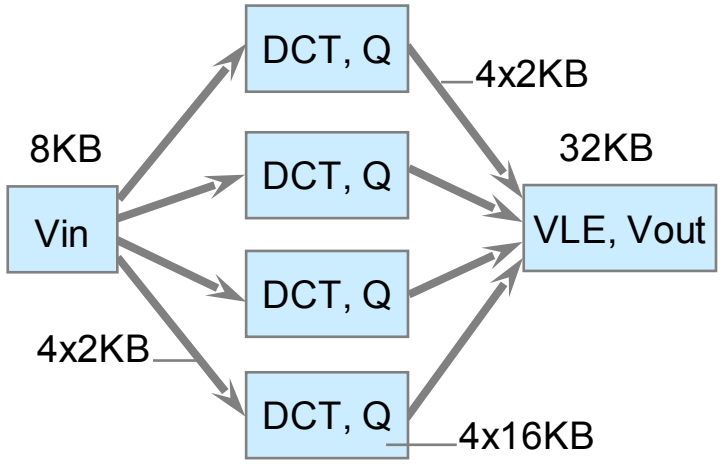
Example architecture instances for a single-tile JPEG encoder:



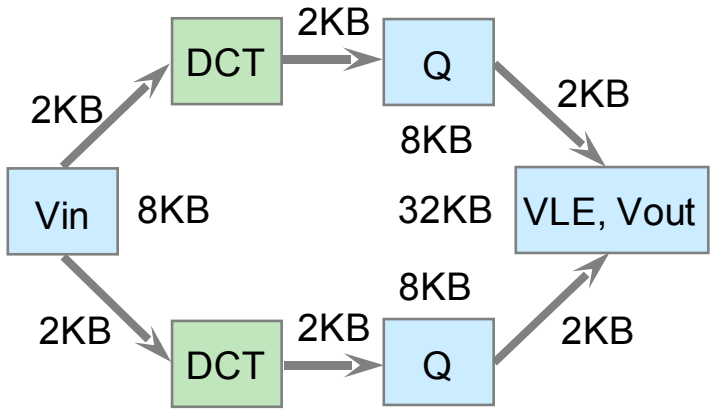
2 MicroBlaze processors (50KB)



1 MicroBlaze, 1HW DCT (36KB)

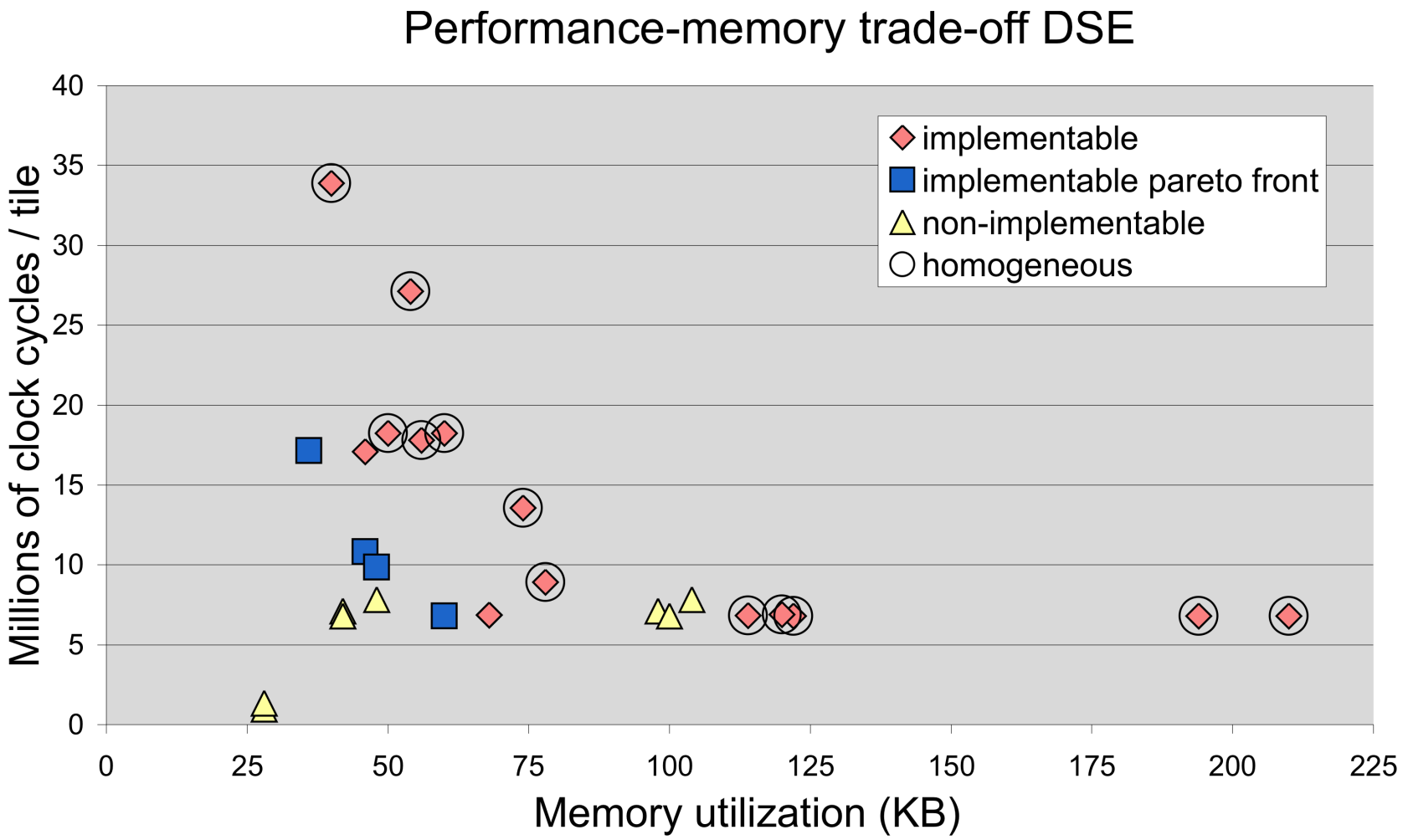


6 MicroBlaze processors (120KB)



4 MicroBlaze, 2HW DCT (68KB)

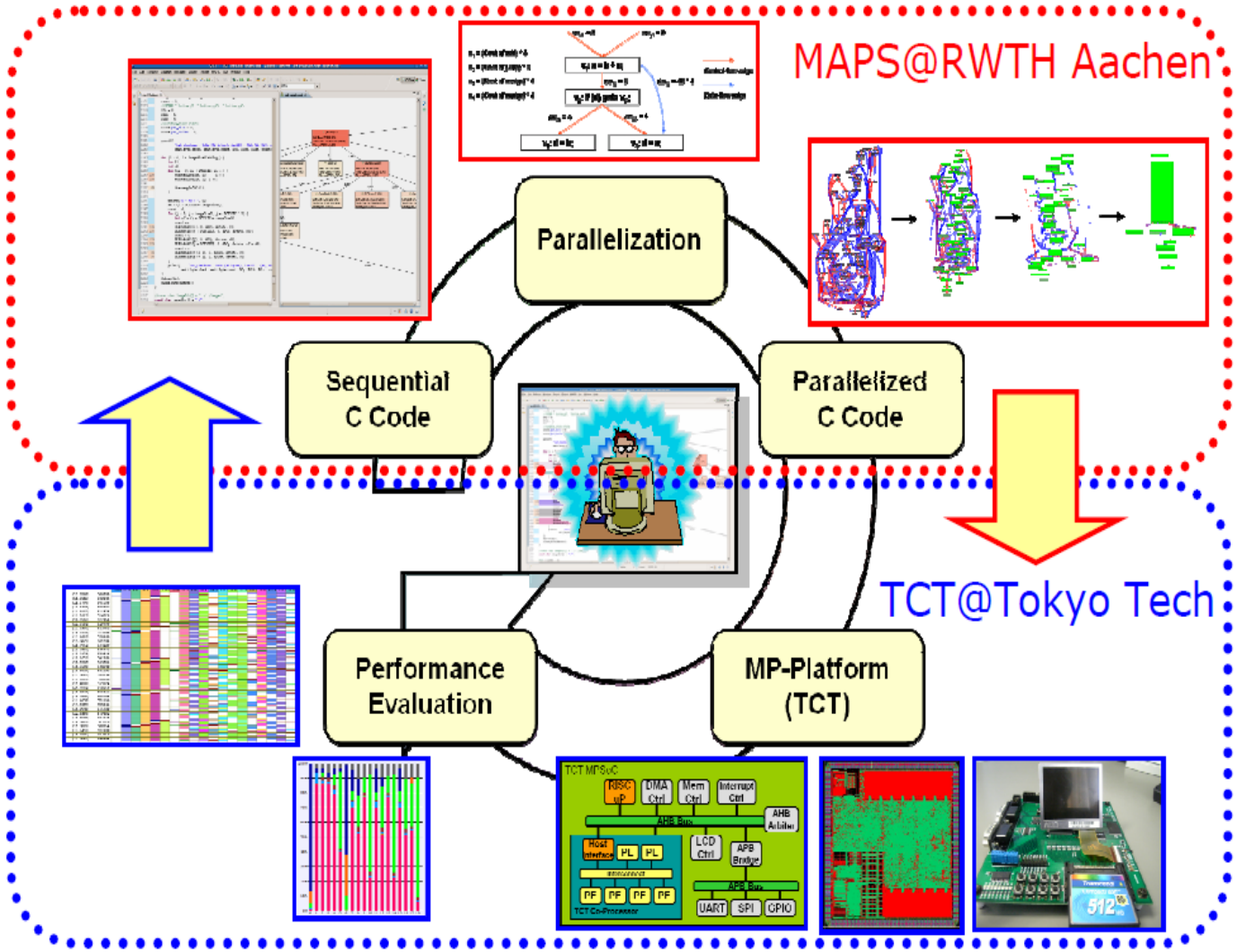
Sesame DSE results:
Single JPEG encoder DSE



A Simple Classification

Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given model	Map to CELL, HOPES, ETHAM	COOL codesign tool; EXPO/SPEA2
Auto-parallelizing	Franke, O'Boyle et al. Mnemee MAPS	Daedalus

MAPS-TCT Framework



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

MAPS: MPSoC Application Programming Studio

- ***A practical MPSoC software development tool suite***
  - Sequential C (input) → "threaded C" (output)
  - Powerful analysis tools for providing rich feedback to the programmers
    - *Static dependence analysis*
    - *Dynamic profiling*
  - Powerful clustering method for extracting coarse-grain parallelism
    - *Weighted Statement Control Data Flow Graph (WSCDFG):* annotates dynamic profiling information on CDFG
    - *Coupled Block (CB):* subgraph of WSCDFG that is schedulable and tightly coupled by data dependence
    - *Constrained Agglomerative Hierachical Clustering (CAHC):* iterative clustering for building coarser graphs

A Simple Classification

Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given model	Map to CELL, HOPES, ETHAM	COOL codesign tool; EXPO/SPEA2
Auto-parallelizing	Franke, O'Boyle et al. Mnemee 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!***

Auto-Parallelizing Compilers (2)

- LooPo (C. Lengauer et al.)
- Various commercial compilers
- …

Mostly focusing on HPC

Workflow of Auto-Parallelization for Multi-DSPs (B. Franke, M. O'Boyle)

Program Recovery
- Removal of undesired low-level constructs in IR
- Replacement by equivalent high-level constructs

Parallelism Detection
- Identification of parallelizable loops

Partitioning and Mapping of Data
- Minimization of communication overhead between DSPs

Memory Access Localization
- Minimization of accesses to remote memories

Data Transfer Optimization
- Exploitation of DMA for burst transfers

☞Beyond Loop Parallelization and Static Analysis

Problems with Real Codes

- "Hand-optimized" for specific platform
  - E.g. pointer based array traversals for efficient address calculation
- Unusual idioms to implement frequently used operations
  - E.g. modulo operations in array index expressions for circular buffers
- Defeat auto-parallelization!
  - "Plain C" is best for parallelization
  - Need to recover canonical program representation

Program Recovery
- Eliminate Pointer Conversion -

Affine array index expressions are critical!

```
int A[N],B[N],C[N];
int *ptr_a = &A[0];
int *ptr_b = &B[0];
int *ptr_c = &C[N-1];
for (i = 0; i < N; i++)
{
   *ptr_a++ = *ptr_b++ * *ptr_c--;
}
```

```
int A[N],B[N],C[N];

for (i = 0; i < N; i++)
{
  A[i] = B[i] * C[N-i-1];
}
```

Affine Index Expressions

Program Recovery
- Modulo Elimination -

Affine array index expressions are critical!

```
int A[N], B[M];

for (i = 0; i < N; i++)
{
   X = A[i] * B[i%M];
}
```

```
int A[N], B[M];

for (i = 0; i < N/M; i++)
{
   for (j = 0; j < M; j++)
   {
      X = A[i*M+j] * B[j];
   }
}
```

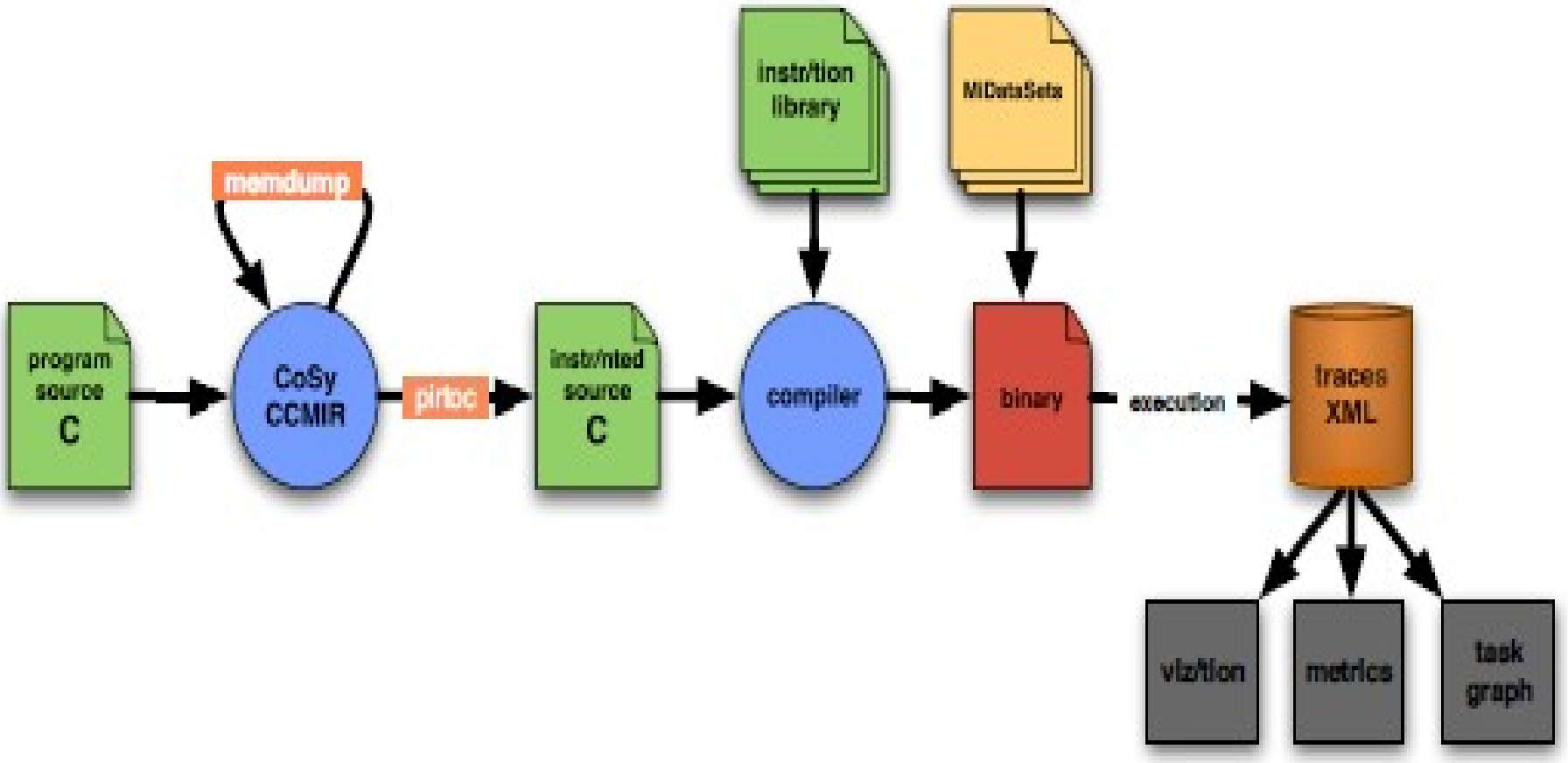
Affine Index Expressions

Rank Modifying Code and Data Transformations

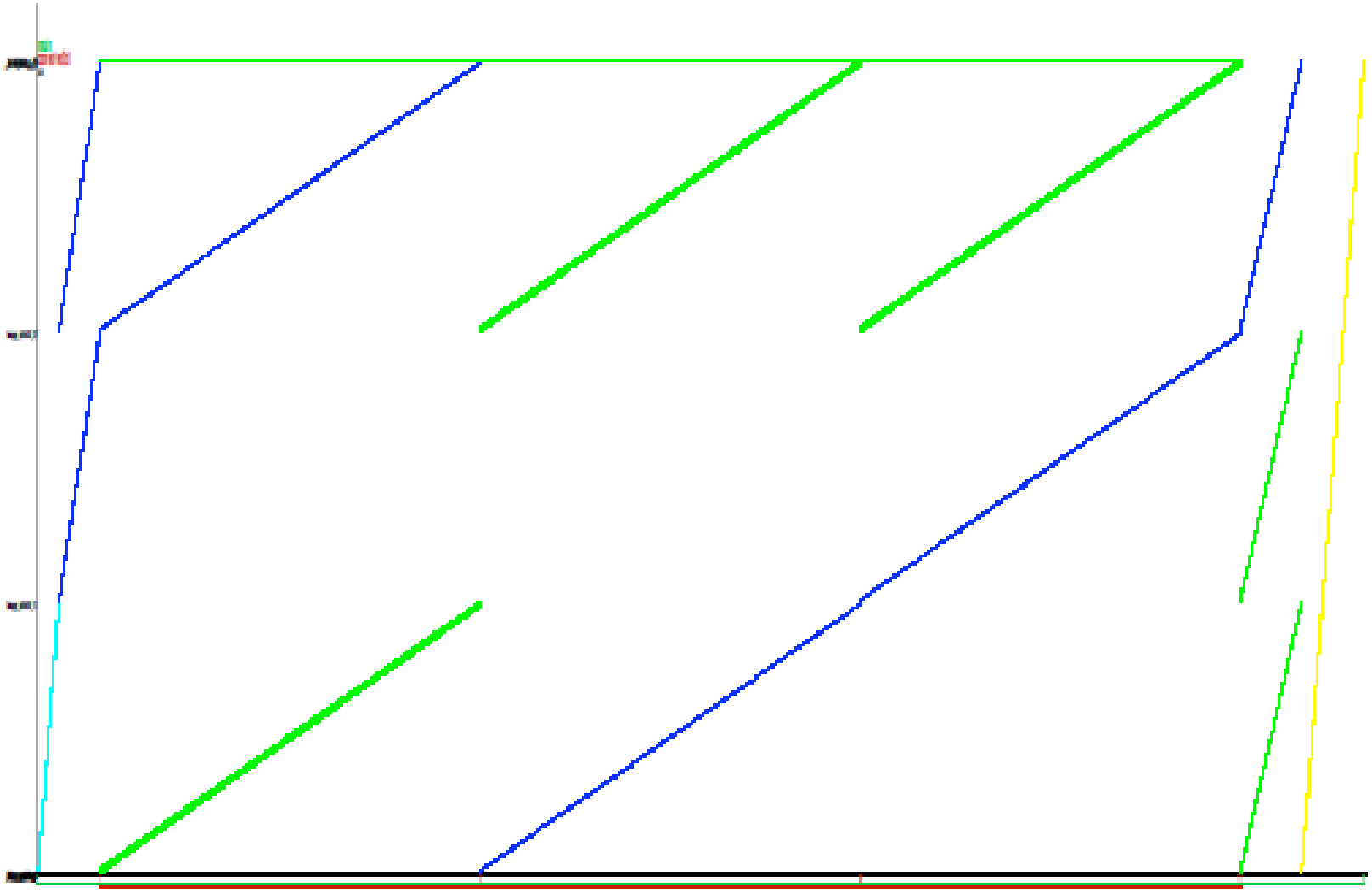
- Unified Linear Algebra Framework for Code and Data Transformations
- Extensions to Unimodular Transformations
  - Elements of transformation matrix are functions with div & mod
- Representation of additional transformations
  - Array dimensionality transformations
  - Strip-mining and linearization of loops

Coarse Grain Parallelism & Task Graph Extraction

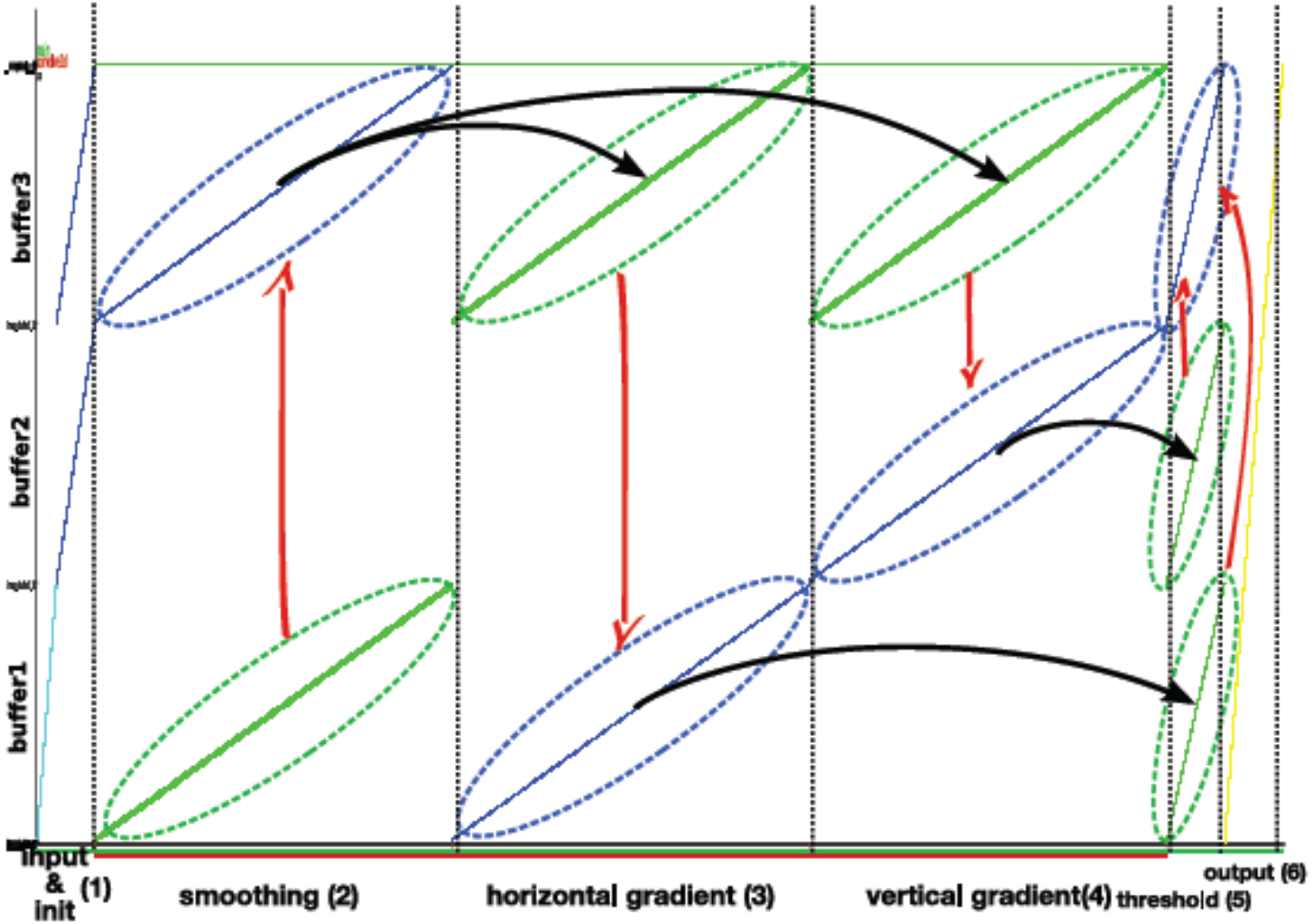
Profiling Infrastructure



Coarse Grain Parallelism & Task Graph Extraction



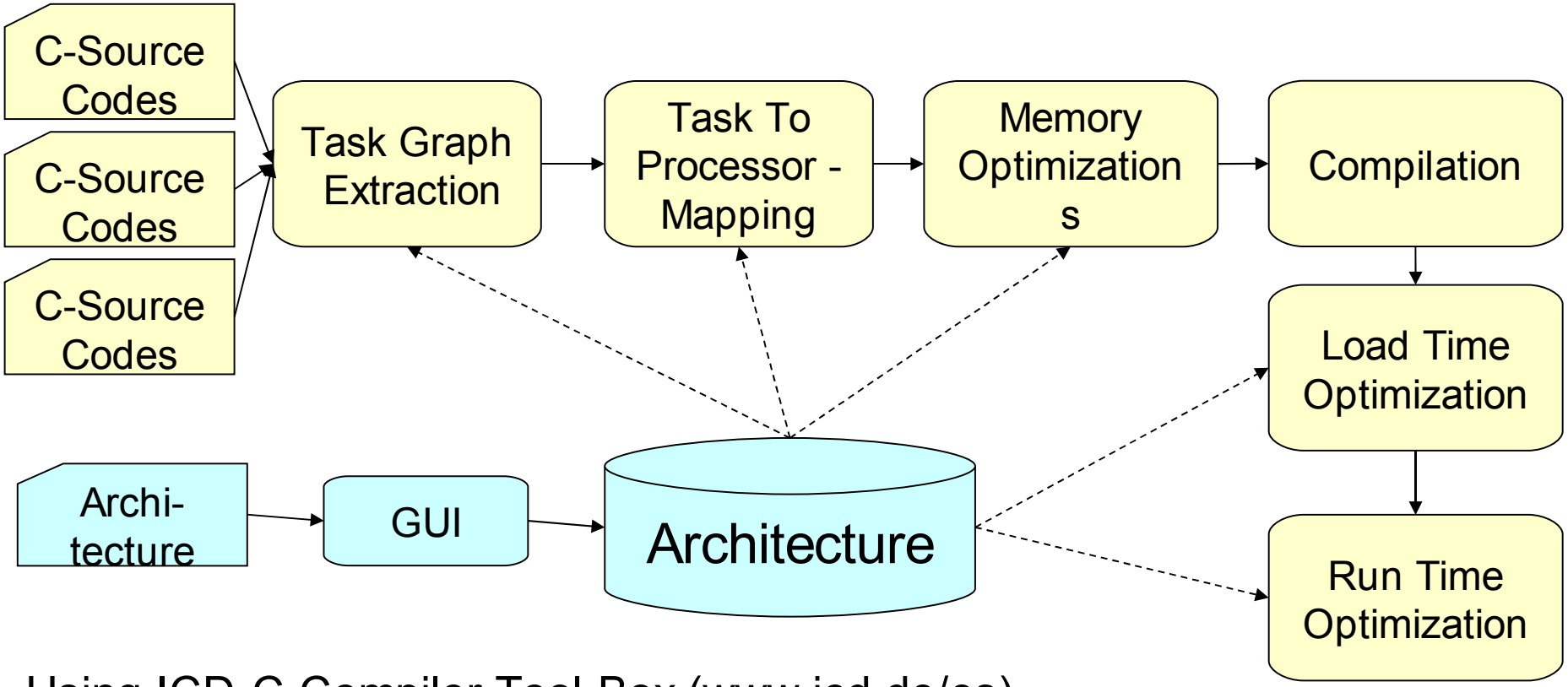
Coarse Grain Parallelism & Task Graph Extraction



Memory architecture aware compilation: Proposed Mnemee Tool Flow (Simplified)

Mnemee=*M*emory ma*n*ag*eme*nt technology for adaptive and efficient design of *e*mbedded systems,

(European Project: IMEC, Dortmund, Eindhoven, U. Athens, Intracom, Thales)



Using ICD-C Compiler Tool-Box (www.icd.de/es)

Future Work

- Implementation of the Mnemee tool flow
- 2nd Workshop on Mapping of Applications to MPSoCs
  - To be held June 29-30, 2009, at Rheinfels Castle
  - Information: http://www.artist-embedded.org/artist/Mapping-Applications-to-MPSoCs.html
- Work by other researchers in the area

Summary (1)

Mapping applications onto heterogeneous MP systems needs

- allocation (if hardware is not fixed)
- binding of tasks to resources
- scheduling

There exists a large design space

Evolutionary algorithms currently are the only algorithms known to solve the existing multi-objective optimization problems.

Details on the encoding, optimization strategies etc. are available in papers by L. Thiele et al. (ETH Zürich) and R. Ernst (TU Braunschweig)

Summary (2)

- Clear trend toward multi-processor systems for embedded systems
- Using architecture **crucially** depends on **mapping tools**
- ArtistDesign network cluster focusing on mapping
- Providing an overview of available techniques
- Two criteria for classification
  - Fixed / flexible architecture
  - Auto parallelizing / non-parallelizing
- Introduction to proposed Mnemee tool chain
- Future work
- Summary