Scalable Software for MPSoC Platforms

Lothar Thiele, Iuliana Bacivarov, Wolfgang Haid, Kai Huang

ETH Zurich, Switzerland
Embedded Systems

Information processing system that is physically embedded within a larger system
A Sample HW Architecture (EU-SHAPES)

- Heterogeneous Tiles
- Regular Tile-Structure
- On-chip and off-chip networks
- SW Portability and Scalability
A Sample HW Architecture (EU-SHAPES)

- Heterogeneous Tiles
- Regular Tile-Structure
- On-chip and off-chip networks
- **SW Portability and Scalability**
Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- Optimization
- Scalability
- Calibration
Some Challenges in MPSoC Programming

• Design Process
• Programming Model
• Optimization
• Scalability
• Calibration

• DOL (Distributed Operation Layer)
Design Exploration

Application

Partitioning

Architecture
Template

Allocation

Task Graph

Binding

Architecture

Resource Sharing

Analysis
Optimization

Design Decisions

Optimization Feedback
DOL Design Flow
Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- Optimization
- Calibration
- Scalability

- DOL (Distributed Operation Layer)
- Process networks and explicit communication
Application Specification

Structure

- **Process Network**
  - Processes
  - SW channels (FIFO behavior)

- **Iterators**
  - Scalability for processes, SW channels, entire structures

Functional specification

- **Language**: C/C++
- **API**: DOL primitives

```
Algorithm 1 Process Model
1: procedure INIT(DOLProcess p)    ▶ initialization
2: initialize local data structures  
3: end procedure
4: procedure FIRE(DOLProcess p)    ▶ execution
5: DOL_read(INPUT, size, buf)      ▶ blocking read
6: manipulate                       
7: DOL_write(OUTPUT, size, buf)    ▶ blocking write
8: end procedure                   
```
Scalability at Specification Level

- Separation of instruction/thread level parallelism (inside processes) and process-level parallelism.

- Use of iterators in
  - architecture specification
  - application specification
  - mapping specification

\[ \{(i, j) : 1 \leq i \leq N \land i \leq j \leq N\} \]
Target Platform Abstraction (1)

- **Topology modeled by a graph**
  - two node types:
    - execution and comm. resources
    - storage resources
- **Execution resources**
  - RISCs, DSPs, ...
- **Communication resources**
  - buses, switches, links, I/Os
- **Storage resources**
  - RAMs, HW FIFOs, ...

Diagram:
- DSP core
- Int. Data Memory
- AHB Slave
- AHB Master DMA
- DXM interface
- NoC
- Multi-layer BUS
- DNP sub-system
- DMA interface
- buf mem. of switch
- DNP sub-system
Target Platform Abstraction (2)

- **RISC 0**
- **C0**
- **DSP 0**
- **MEM**
- **RISC 1**
- **C1**
- **DSP 1**

**Symbols:**
- Processor
- Memory
- Hardware channel
- Input / output / bidirectional port
- Connection

**Lines:**
- Connections between components

**Note:**
- Diagram illustrates the architecture connections between different components.
Target Platform Abstraction (2)

- <processor name="processor1" type="DSP">
  <port name="processor_port" type="duplex" />
  <configuration name="clock" value="100 MHz" />
</processor>

+ <processor name="processor2" type="RISC">
+ <memory name="sharedmemory" type="DXM">

- <hw_channel name="in_tile_link" type="bus">
  <port name="port1" type="duplex" />
  <port name="port2" type="duplex" />
  <port name="port3" type="duplex" />
  <configuration name="buswidth" value="32bit" />
</hw_channel>

- <connection name="processor1link">
  <origin name="processor1">
    <port name="processor_port" />
  </origin>
  <target name="in_tile_link">
    <port name="port1" />
  </target>
</connection>

+ <connection name="processor2link">
+ <connection name="memorylink"方才
Mapping Specification

- **Binding**
  - Processes to execution resources
  - SW channels to read/write paths

- **Scheduling**
  - Processors
  - Communication

- **Constraints**
  - To be considered during HdS generation
Mapping Specification

- `<binding name="generator_binding" type="computation">
  <origin name="generator"/>
  <target/>
  <resource name="processor2"/>
  <schedule type="roundrobin"/>
</binding>`

+ `<binding name="consumer_binding" type="computation">
  `<binding name="square_binding" type="computation">
    <origin name="square"/>
    <target/>
    <resource name="processor1"/>
    <schedule type="fixedpriority">
      <configuration name="priority" value="1"/>
    </schedule>
  </binding>`

+ `<binding name="C1_binding" type="communication">
+ `<binding name="C2_binding" type="communication">

Iterated mapping possible
Mapping Constraints

- Certain processes must run on certain processors or processor types
- Some processes must be mapped to the same processor
- Some communication must be mapped onto specific paths
- Restrictions on resource sharing policies
Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- Optimization
- Scalability
- Calibration

- DOL (Distributed Operation Layer)
- Process networks and explicit communication
- Hybrid black-box methods
DOL Design Flow

- Performance Estimation
  - application
  - architecture
  - mapping

- Scheduling - Mapping Optimizer
  - architecture
  - application
  - mapping

- HdS / OS

- Compiler

- VP simulator
Optimization Criteria and Method

- **Correctness:**
  - avoid memory/buffer overflow / underflow
  - respect mapping constraints

- **Performance:**
  - end-to-end deadlines, throughput
  - jitter and burstiness
  - small sensitivity / large robustness

- **Optimization Method:**
  - Population-based multi-objective optimization.
  - Constraint handling embedded into optimizer
  - Exploration based on bottleneck and robustness information
Multiobjective Optimization

Maximize

\[ f \]

\( (x_1, x_2, \ldots, x_n) \rightarrow (y_1, y_2, \ldots, y_k) \)

decision space

objective space

Pareto optimal
not dominated
dominated
Multi-objective Optimization

Definition 1 (Dominance relation)
Let \( f, g \in \mathbb{R}^m \). Then \( f \) is said to dominate \( g \), denoted as \( f \succ g \), iff

1. \( \forall i \in \{1, \ldots, m\} : f_i \geq g_i \)
2. \( \exists j \in \{1, \ldots, m\} : f_j > g_j \)

Definition 2 (Pareto set)
Let \( F \subseteq \mathbb{R}^m \) be a set of vectors. Then the Pareto set \( F^* \subseteq F \) is defined as follows: \( F^* \) contains all vectors \( g \in F \) which are not dominated by any vector \( f \in F \), i.e.

\[
F^* := \{ g \in F \mid \forall f \in F : f \succ g \} \tag{1}
\]
Multiobjective Optimization

Maximize \((y_1, y_2, ..., y_k) = f(x_1, x_2, ..., x_n)\)

Pareto set = set of all Pareto-optimal solutions
A Generic Multiobjective EA

- Population
  - Evaluate sample vary
  - New population

- Archive
  - Update truncate
  - New archive
Evolutionary Algorithms for DSE

- **EA**
  1. selection
  2. recombination
  3. mutation

- **individual**
  - allocation
  - binding
  - scheduling

- **decode allocation**
- **decode binding**
- **scheduling**

- **design point**
  (implementation)

- **fitness evaluation**

**apply heuristics:** hybrid optimization

**fitness**
Evolutionary Algorithms for DSE

Evolutionary Algorithms (EA)

1. Selection
2. Recombination
3. Mutation

- Allocation
- Binding
- Decode Allocation
- Decode Binding
- Scheduling
- Fitness Evaluation

Design Point (Implementation)

Fitness
Design Space Exploration Framework

- **PISA&EXPO**
  - multi-objective optimization using evolutionary algorithms
  - [PISA] https://www.tik.ee.ethz.ch/pisa;
  - [EXPO] https://www.tik.ee.ethz.ch/expo
Design Space Exploration

Estimation

Analysis Method

45.4%

46.2%

27.1%

59.2%

90.8%

98.9%

83.6%

90.8%

16.9%

93.0%

EA

Evolutionary Algorithm

RISC 1

DSP 1

BUS 1

NoC

BUS 2

RISC 2

DSP 2

[PISA]

https://www.tik.ee.ethz.ch/pisa

[EXPO]

https://www.tik.ee.ethz.ch/expo
PISA Website

http://www.tik.ee.ethz.ch/pisa
Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- Optimization
- Scalability
- Calibration
- DOL (Distributed Operation Layer)
- Process networks and explicit communication
- Hybrid black-box methods
- Multi-level performance estimation
DOL Design Flow

Performance Estimation

application  architecture  mapping

Scheduling -Mapping Optimizer

Performance Estimation

architecture  application  mapping

HdS / OS

Compiler

binary  VP simulator
Analysis and Design

Embedded System = Computation + Communication + Resource Interaction

Analysis:
Infer system properties from subsystem properties.

Design:
Build a system from subsystems while meeting requirements.
Challenge

Make Analysis and Synthesis Compositional

Analysis:
Infer system properties from subsystem properties.

Design:
Build a system from subsystems while meeting requirements.
Why Is Evaluation Difficult?

- **Non-determinism:**
  - uncertain system environment, e.g. load scenarios
  - (non-deterministic) computations in processing nodes

- **Interference:**
  - *sharing* exclusive resources (scheduling and arbitration)
  - *resource feedback:* internal data streams interact on exclusive resources which in turn change stream characteristics
  - interaction between *resource types:* exclusive (computation, communication) and shared (energy)
  - long range *dependencies*
Difficulties

Task Communication
Task Scheduling
Complex Input:
- Timing (jitter, bursts, ...)
- Different Event Types

Variable Resource Availability
Variable Execution Demand
- Input (different event types)
- Internal State (Program, Cache, ...)
System-Level Evaluation Methods

e.g. delay, power, throughput

Worst-Case

Best-Case

Real System Measurement Simulation Probabilistic Analysis

Worst Case Analysis
Run-time Comparison

• **Modular Performance Analysis**
  – Generation: 0.5s; Analysis: 2s

• **Functional Simulation**
  – Generation: 35s; Simulation: 3s

• **Virtual Platform Simulation**
  – Generation: 170s; Simulation: 1300s
Simple Analytic Model: Additive Model

1. Processor $c$ with worst total runtime

$$\text{obj}_1 = \max_{c \in C} \left\{ \sum_{\forall p \text{ mapped to } c} n(p) \cdot r(p, c) \right\}$$

2. Communication link with worst load

$$\text{obj}_2 = \max_{g \in \mathcal{G}} \left\{ \sum_{\forall s \text{ mapped onto } g} \frac{b(s)}{t(g)} \right\}$$

- $n(p)$: Number of firings of task $p$
- $r(p, c)$: Runtime of task $p$ on processor $c$
- $b(s)$: Bandwidth of communication link $g$
- $t(g)$: Total time of communication link $g$
Classification of Analytic Methods

• **Some classical approaches**
  – Combine Binding and Resource Allocation:
    • Multiprocessor scheduling: Extension of uni-processor scheduling theory to multiple processors.
  – Holistic Analysis
    • [Tindell et al.]: Based on response time analysis and fixed point calculations
  – Component-Based Analysis
    • Symta/S [Ernst et. al.]: Concatenation of classical results from uni-processor real-time analysis
    • *Network calculus [Cruz et. al.]: Generalized modeling of streams and resources based on arrival and service curves*
Modular Performance Analysis (MPA)

http://www.mpa.ethz.ch/
Load Model (Environment)

Event Stream

number of events in in t=[0 .. 2.5] ms

Arrival Curve $\alpha$

maximum / minimum arriving demand in any interval of length 2.5 ms
Load Model - Examples

- periodic
- periodic w/ jitter
- periodic w/ burst
- complex
Service Model (Resources)

Resource Availability

available service in $t=[0 \ldots 2.5]$ ms

Service Curves [$\beta^l$, $\beta^u$]

maximum/minimum available service in any interval of length 2.5 ms
Service Model - Examples

- **full resource**: 
  - \( \beta^u \)
  - \( \beta^l \)

- **bounded delay**: 
  - \( \beta^u \)
  - \( \beta^l \)

- **TDMA resource**: 
  - \( \beta^u \)
  - \( \beta^l \)

- **periodic resource**: 
  - \( \beta^u \)
  - \( \beta^l \)
Processing Model (HW/SW)

HW/SW Components
- Processing semantics and functionality of HW/SW tasks

Abstract Components
- $\alpha'(\Delta) = f(\alpha, \beta)$

Service Model 
Load Model 
Processing Model

HW/SW Task

Predicate $\Psi$

RT$C$

$\alpha$

$\Delta$

$\beta$

$t$
System Module

Linear System Theory [Baccelli, Cohen, Olsder, Quadrat 1992]
Calculus for Networks [Le Boudec 1998, 2001], [Cruz 1991]
Adversarial Queuing Theory [Andrews, Borodin, Kleinberg, Leighton, … 1996]
System Composition

How to interconnect service?

Scheduling!

\[ \beta_{CPU} \]

\[ \beta_{BUS} \]

\[ \beta_{DSP} \]

\[ \alpha \]

\[ \alpha' \]
Scheduling and Arbitration

FP/RM \( \beta \)

\( \alpha_A \rightarrow \alpha'_A \)

\( \alpha_B \rightarrow \alpha'_B \)

GPC

GPC

GPS \( \beta' \)

EDF \( \beta \)

\( \alpha_A \rightarrow \alpha'_A \)

\( \alpha_B \rightarrow \alpha'_B \)

EDF

TDMA \( \beta' \)

\( \alpha_A \rightarrow \alpha'_A \)

\( \alpha_B \rightarrow \alpha'_B \)

RR \( \beta \)

\( \alpha_A \rightarrow \alpha'_A \)

\( \alpha_B \rightarrow \alpha'_B \)

RR

Share

sum

GPC

GPC

GPC

GPC

\( \beta' \)

\( \beta' \)

\( \beta' \)

\( \beta' \)

Service Model

Load Model

Processing Model

Swiss Federal Institute of Technology

Computer Engineering and Networks Laboratory
Complete System Composition

\[ \beta_{CPU} \rightarrow \alpha \rightarrow \gamma \rightarrow \beta_{BUS} \rightarrow \beta_{DSP} \]

- CPU
- BUS
- DSP
- GPC
- TDMA
- RM
- GSC

Service Model 
Load Model 
Processing Model
Delay and Backlog

\[ \text{maximum delay } D \]

\[ \text{maximum backlog } B \]
Celebrated Result on Delay and Backlog

\[
\beta_{\text{eff}}^l = \bigotimes_{i=1}^{n} \beta_i^l
\]

maximum end-to-end delay \( D \)

accumulated maximum backlog \( B \)

end-to-end delay \( D \)

\[\Delta\]

\[\alpha^u\]

\[\beta_1^l\]

\[\beta_2^l\]

\[\beta_n^l\]

\[\alpha^u\]

\[\text{GPC}_1\]

\[\text{GPC}_2\]

\[\text{GPC}_n\]
RTC Toolbox (www.mpa.ethz.ch/rtctoolbox)

<table>
<thead>
<tr>
<th>Matlab Command Line</th>
<th>Simulink</th>
</tr>
</thead>
<tbody>
<tr>
<td>MPA Library</td>
<td>RTI Library</td>
</tr>
<tr>
<td>Min-Plus/Max-Plus Algebra Library</td>
<td></td>
</tr>
<tr>
<td>Matlab / Java Interface</td>
<td></td>
</tr>
<tr>
<td>Java API</td>
<td></td>
</tr>
<tr>
<td>Min-Plus/Max-Plus Algebra, Utilities</td>
<td></td>
</tr>
<tr>
<td>Efficient Curve Representation</td>
<td></td>
</tr>
</tbody>
</table>
Integration

- Calibration data
  - | tile | BCET | WCET |
  - | t₁  | 4700 | 5000 |
  - | t₂  | 8000 | 9000 |
- Application specification
- Mapping specification
- Architecture specification
- interconnect

- Framework-independent analysis meta-model
- MPA Matlab toolbox script
- SymTA/S system description
- Other formats
Integration / Application Modeling

Algorithm 1 Example of a process with multiple inputs and outputs.

1: function FIRE(DOLProcess *p)
2:    DOL_read(input[1], buffer_in[1], N_in[1]);
3:    DOL_read(input[2], buffer_in[2], N_in[2]);
4:    DOL_read(input[3], buffer_in[3], N_in[3]);
5:    execute;
6:    DOL_write(output[1], buffer_out[1], N_out[1]);
7:    DOL_write(output[2], buffer_out[2], N_out[2]);
8: end function
Integration / Architecture Template
Integration / Communication Modeling

intra-processor communication

inter-processor communication
Some Challenges in MPSoC Programming

- Design Process
- Programming Model
- Optimization
- Calibration
- Scalability
- DOL (Distributed Operation Layer)
- Process networks and explicit communication
- Hybrid black-box methods
- Reference points
DOL Design Flow

Performance Estimation

applicationarchitecturemapping

Scheduling - Mapping Optimizer

Performance Estimation

architectureapplicationmapping

HdS / OS

Compiler

binary

VP simulator

calib. data
Reference Points

• Data sheets
  – basic platform characteristics, bounds on delay, throughput, …

• Functional Simulation
  – communication volume, number of task invocations, …

• Low Level Simulation
  – Benchmark mappings: map benchmark applications to platform for profiling the OS, communication services, network, …
  – Sample Mappings: map the application to the platform in a limited number of settings; estimations on execution and communication times
Integration

<table>
<thead>
<tr>
<th>tile</th>
<th>BCET</th>
<th>WCET</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>4700</td>
<td>5000</td>
</tr>
<tr>
<td>t₂</td>
<td>8000</td>
<td>9000</td>
</tr>
</tbody>
</table>

MPA Matlab toolbox script
SymTA/S system description
other formats

framework-independent analysis meta-model

intercon.  p₁  p₂
          p₃  p₂₃
          p₄

t₁  p₁
    p₃
    p₄

interconnect

t₂  p₂
    p₃
    p₄
Functional Simulation

- Process network (with iterators)
- Flattened process network (w/o iterators)
- Visualization
- XML Flattener
- Simulation Generator
- SystemC Simulation
- Processes behavior

Diagram showing the process network (with iterators) and flattened process network (w/o iterators) connected to XML Flattener, Simulation Generator, and SystemC Simulation.
Example 1: MJPEG Process Network
Example 1: MJPEG Functional Simulation

<table>
<thead>
<tr>
<th>number of activations</th>
<th>1</th>
<th>2</th>
<th>297000</th>
<th>891000</th>
<th>297000</th>
<th>594000</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>transferred data (in MByte)</td>
<td>15.0</td>
<td>519.4</td>
<td>421.4</td>
<td>421.4</td>
<td>975.6</td>
<td>228.2</td>
<td></td>
</tr>
</tbody>
</table>
Example 1: Workload Extraction

functional simulation  workload bounds

produced data amount for each communication event

\[
\begin{align*}
29469 \\
: \\
171800 \\
199954 \\
228124 \\
256287 \\
285747 \\
315241 \\
344750 \\
374228 \\
403744 \\
433213 \\
462725 \\
: \\
2911282
\end{align*}
\]

e = 3

accumulated workload

\[
\gamma(e)
\]

periodic extension

\[
\gamma^\mu(e)
\]

safe bounds from trace

\[
\gamma(e)
\]

number of consecutive events

\[
L = 4
\]
Example 1: Platform
Example 1: Reference Points

- Platform benchmarks
  - communication bandwidth of network components
- Individual task simulations

<table>
<thead>
<tr>
<th>Process</th>
<th>Runtime on RISC</th>
<th>Runtime on DSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>dispatch gop</td>
<td>0.13</td>
<td>0.20</td>
</tr>
<tr>
<td>dispatch macroblock</td>
<td>6.68</td>
<td>8.52</td>
</tr>
<tr>
<td>dispatch block</td>
<td>0.06</td>
<td>0.04</td>
</tr>
<tr>
<td>transform block</td>
<td>2.00</td>
<td>1.25</td>
</tr>
<tr>
<td>collect block</td>
<td>0.05</td>
<td>0.04</td>
</tr>
<tr>
<td>collect macroblock</td>
<td>12.33</td>
<td>8.51</td>
</tr>
<tr>
<td>collect gop</td>
<td>0.18</td>
<td>0.30</td>
</tr>
</tbody>
</table>
# Example 1: Calibration Times

<table>
<thead>
<tr>
<th>Step</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>model calibration</strong> (one-time effort)</td>
<td></td>
</tr>
<tr>
<td>functional simulation generation</td>
<td>35 s</td>
</tr>
<tr>
<td>functional simulation</td>
<td>3 s</td>
</tr>
<tr>
<td>synthesis (generation of binary)</td>
<td>176 s</td>
</tr>
<tr>
<td>simulation on virtual platform</td>
<td>1300 s</td>
</tr>
<tr>
<td>log-file analysis and back-annotation</td>
<td>50 s</td>
</tr>
<tr>
<td>model generation</td>
<td>0.5 s</td>
</tr>
<tr>
<td>performance analysis based on generated model</td>
<td>2 s</td>
</tr>
</tbody>
</table>

*To be done for every exploration cycle*
Example 1: Mapping Optimization

- Exploration under two criteria:
  1. load balancing for the computation
  2. load balancing for the communication
Example 2: Wave Field Synthesis
Example 2: Platform

- ARM tile 1
  - Instruction and data memory
  - Scratchpad memory
  - DMA controller
  - ARM core

- ARM tile N
  - Instruction and data memory
  - Scratchpad memory
  - DMA controller
  - ARM core

- Bus
Example 2: Analysis Models

M-JPEG

WFS
## Example 2: Compilation Times

<table>
<thead>
<tr>
<th>step</th>
<th>duration</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>P-C</td>
</tr>
<tr>
<td>model calibration (one-time effort)</td>
<td></td>
</tr>
<tr>
<td>functional simulation generation</td>
<td>22 s</td>
</tr>
<tr>
<td>functional simulation</td>
<td>0.2 s</td>
</tr>
<tr>
<td>synthesis (generation of binary)</td>
<td>2 s</td>
</tr>
<tr>
<td>simulation on MPARM</td>
<td>23 s</td>
</tr>
<tr>
<td>log-file analysis and back-annotation</td>
<td>1 s</td>
</tr>
<tr>
<td>model generation</td>
<td>1 s</td>
</tr>
<tr>
<td>performance analysis based on generated model</td>
<td>0.2 s</td>
</tr>
</tbody>
</table>
### Example 2: Accuracy

<table>
<thead>
<tr>
<th>process</th>
<th>proc.</th>
<th>pr.</th>
<th>delay</th>
<th>backlog</th>
<th>pr.</th>
<th>delay</th>
<th>backlog</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-c.p1</td>
<td>1</td>
<td>1</td>
<td>209 (≤ 223)</td>
<td>5 (≤ 6)</td>
<td>2</td>
<td>357 (≤ 401)</td>
<td>6 (≤ 8)</td>
</tr>
<tr>
<td>p-c.p3</td>
<td>1</td>
<td>2</td>
<td>329 (≤ 371)</td>
<td>7 (≤ 9)</td>
<td>1</td>
<td>37 (≤ 43)</td>
<td>1 (≤ 2)</td>
</tr>
<tr>
<td>p-c.p2</td>
<td>2</td>
<td>1</td>
<td>29 (≤ 38)</td>
<td>1 (≤ 2)</td>
<td>1</td>
<td>30 (≤ 35)</td>
<td>1 (≤ 2)</td>
</tr>
<tr>
<td>mjpeg.ss</td>
<td>1</td>
<td>1</td>
<td>203 (≤ 240)</td>
<td>4 (≤ 6)</td>
<td>2</td>
<td>321 (≤ 441)</td>
<td>3 (≤ 5)</td>
</tr>
<tr>
<td>mjpeg.ms</td>
<td>2</td>
<td>1</td>
<td>694 (≤ 781)</td>
<td>1 (≤ 3)</td>
<td>1</td>
<td>133 (≤ 190)</td>
<td>1 (≤ 1)</td>
</tr>
<tr>
<td>mjpeg.sf</td>
<td>2</td>
<td>1</td>
<td>2591 (≤ 3014)</td>
<td>5 (≤ 6)</td>
<td>2</td>
<td>3226 (≤ 4315)</td>
<td>6 (≤ 6)</td>
</tr>
<tr>
<td>mjpeg.mf</td>
<td>2</td>
<td>2</td>
<td>1881 (≤ 2143)</td>
<td>2 (≤ 4)</td>
<td>1</td>
<td>307 (≤ 340)</td>
<td>1 (≤ 2)</td>
</tr>
<tr>
<td>mjpeg.zii</td>
<td>3</td>
<td>1</td>
<td>6164 (≤ 6762)</td>
<td>4 (≤ 6)</td>
<td>1</td>
<td>5971 (≤ 6663)</td>
<td>4 (≤ 6)</td>
</tr>
<tr>
<td>wfs.ctrl</td>
<td>1</td>
<td>1</td>
<td>202 (≤ 235)</td>
<td>3 (≤ 5)</td>
<td>3</td>
<td>405 (≤ 795)</td>
<td>5 (≤ 7)</td>
</tr>
<tr>
<td>wfs.src</td>
<td>1</td>
<td>2</td>
<td>292 (≤ 387)</td>
<td>4 (≤ 5)</td>
<td>2</td>
<td>228 (≤ 357)</td>
<td>3 (≤ 5)</td>
</tr>
<tr>
<td>wfs.ls</td>
<td>1</td>
<td>3</td>
<td>4931 (≤ 5402)</td>
<td>8 (≤ 12)</td>
<td>1</td>
<td>4996 (≤ 5512)</td>
<td>9 (≤ 14)</td>
</tr>
<tr>
<td>wfs.comp1</td>
<td>2</td>
<td>1</td>
<td>1606 (≤ 1919)</td>
<td>12 (≤ 15)</td>
<td>2</td>
<td>6157 (≤ 7720)</td>
<td>26 (≤ 30)</td>
</tr>
<tr>
<td>wfs.comp2</td>
<td>2</td>
<td>2</td>
<td>5960 (≤ 6838)</td>
<td>25 (≤ 26)</td>
<td>1</td>
<td>1940 (≤ 2156)</td>
<td>15 (≤ 20)</td>
</tr>
</tbody>
</table>
Open Issues
Compositional Analysis

- Fighting the Abstraction Loss -
Compositional Methods

• ... suffer from abstraction loss:
  – For example, we are not able to properly model timing correlations between streams.
  – Analysis results may be overly pessimistic
  – We need new models that are able to talk about timing correlations between event streams.
Performance Analysis Methods

**Analytic Real-Time Analysis**
- Solution of closed form expressions
- Examples: RTC, SymTA/S, MAST

  + Good scalability
  + Fast analysis

  - Limited to few specific measures
  - Systems restricted to specific models
  - Overly conservative results

**State-based Real-Time Analysis**
- Model checking of properties
- Examples: Timed Automata (TA), FSM

  - Poor scalability
  - Slow verification

  + Verification of functional and non-functional properties
  + Modeling power
  + Exact results

- State space explosion
Hybrid Analysis

Interfaces

Analytic

State-based

Analytic

Analytic

Analytic

Analytic
Interfacing

\[
\alpha' = f(\alpha, \beta)
\]

RTC

\[
\alpha' = f(\alpha, \beta)
\]

RTC

\[
\alpha' = f(\alpha, \beta)
\]

RTC

\[
\alpha' = f(\alpha, \beta)
\]
Do we use the right abstractions?
WCET

\[ x = a + b; \]

LOAD r2, _a
LOAD r1, _b
ADD r3,r2,r1

PPC 755

© Reinhard Wilhelm
(Timing) Predictability

response time

year

WCRT
Bound

WCRT

BCRT
Bound
Application and Architecture

Architecture

fixed cycle CPU
single processor
multiple cores
distributed

Application

single task
static tasks
dynamic tasks

Predictability
Classification of Predictability Loss

- **Analysis Loss:**
  - Construct system that can be easily analyzed
  - Use appropriate abstractions (models and methods)

- **System Design Loss:**
  - Decrease interference, long-range dependencies
  - Increase robustness of components
  - Use appropriate interfaces
Application and Architecture

Architecture

Application

- fixed cycle CPU
- single processor
- multiple cores
- distributed

Predictability

- single task
- static tasks
- dynamic tasks
Classification of Predictability Loss

- **Analysis Loss:**
  - Construct system that can be easily analyzed
  - Use appropriate abstractions (models and methods)

- **System Design Loss:**
  - Decrease interference, long-range dependencies
  - Increase robustness of components
  - Use appropriate interfaces
Application and Architecture

Architecture

fixed cycle CPU

single processor

multiple cores

distributed

Application

single task

static tasks

dynamic tasks
Classification of Predictability Loss

- **Analysis Loss:**
  - Construct system that can be easily analyzed
  - Use appropriate abstractions (models and methods)

- **System Design Loss:**
  - Decrease interference, long-range dependencies
  - Increase robustness of components
  - Use appropriate interfaces
Application and Architecture

Architecture

fixed cycle CPU
single processor
multiple cores
distributed

Application

single task
static tasks
dynamic tasks
A task is (classically) characterized by its WCET.
- May be useful in case of simple processors, but we have long-range state-dependent uni-processor behavior (pipelines, caches, speculation).
- In case of multi-processors, we have additional interferences on the communication system which heavily influences WCET. We also may have intra-task parallelism.
- WCET can no longer be considered as a useful interface between these abstraction layers.

What about the other interfaces?
- Is the classical ISA (using instructions that abstract away time) still appropriate?
Acknowledgement

• **Co-workers:**
  Iuliana Bacivarov, Wolfgang Haid, Kai Huang, Kai Lampka, Clemens Moser, Ernesto Wandeler, Simon Perathoner, Nikolay Stoimenov, Jan Beutel, Jian-Jia Chen, …

• **Funding:**
  EU-SHAPES, EU-PREDATOR, EU-COMBEST, EU-ARTIST, IBM, Siemens, NCCR-MICS, KTI