Hardware/Software Partitioning of Digital Systems

F. Dufour
Advisor: M. Radetzki

ITI
Institut für Technische Informatik
Universität Stuttgart

Department of Technical Computer Science
University of Stuttgart

Seminar “Embedded Systems”
1. Partitioning and digital co-design
   - Co-design
   - Definition of the partitioning problem
   - Partitioning in Co-Design

2. Algorithms for H/S Partitioning
   - Constructive algorithms: hierarchical clustering
   - Iterative algorithms: simulated annealing
   - The partitioning algorithm in the COSYMA system
### Generic processors

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>cheap, standard</td>
</tr>
<tr>
<td>fast and easy development of systems</td>
</tr>
<tr>
<td>often not optimized enough to meet time constraints</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>programmable microcontrollers, microprocessors, ...</td>
</tr>
</tbody>
</table>
Application-specific hardware components

**Properties**
- synthesized using CAD tools
- optimized for a specific task
- but expensive and long to design

**Example**
ASIC, most DSPs, ...
Hardware/Software systems

An ideal hw/sw system would...

- combine the advantages of hardware and software
- meet the time constraints
- be designed in a fast, inexpensive way
Process of Hardware-Software Co-design

1. Functional specification
2. Profiling, simulation, scheduling
3. Partitioning
   - HW synthesis
   - SW synthesis
4. Integration

---

F. Dufour  Advisor: M. Radetzki  Hardware/Software Partitioning of Digital Systems
A Formal Definition of Partitioning

Definition

Given a set \( O = o_1, o_2, \ldots, o_n \) of objects, a partition \( P = p_1, p_2, \ldots, p_m \) is sought, that satisfies:

- \( p_1 \cup p_2 \cup \ldots \cup p_m = O \),
- \( p_i \cap p_j = \{ \} \forall i, j, \ i \neq j \), and
- the cost \( f(P) \) is minimal.
The cost function $f$

Properties

- Evaluates the global partitioning
- Expresses the factors to minimize
  - Execution time
  - Chip area
  - Needed memory
  - Hardware cost
- All costs have different weights (linear, exponential, ...)
- $f$ should be incrementally computable to allow optimization schemes
Aim of partitioning schemes

to assign every **functional object** of the specification to a **system component** of the target architecture

The yielded partition must:
- meet the time constraints
- optimize the costs

⇒ minimize the **cost** function
Difficulty of partitioning

**Problems**

- completely determines the following stages (hw and sw synthesis)
- **NP-hard** problem

⇒ mostly done “by hand”, relying on experience
Outline

1. Partitioning and digital co-design
   - Co-design
   - Definition of the partitioning problem
   - Partitioning in Co-Design

2. Algorithms for H/S Partitioning
   - Constructive algorithms: hierarchical clustering
   - Iterative algorithms: simulated annealing
   - The partitioning algorithm in the COSYMA system
Constructive algorithms

<table>
<thead>
<tr>
<th>Properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>bottom-up</td>
</tr>
<tr>
<td>build a partition</td>
</tr>
<tr>
<td>no cost function</td>
</tr>
<tr>
<td>use <strong>closeness</strong> metrics: communication costs, performance...</td>
</tr>
<tr>
<td>a function of pairs of objects, not of the global system</td>
</tr>
</tbody>
</table>

Constructive algorithms: hierarchical clustering
Iterative algorithms: simulated annealing

The partitioning algorithm in the COSYMA system
Hierarchical clustering: how it works

- at the beginning, each functional object is a partition object
- then, compute closeness between each pair of partition objects
- merge the two p.o. that have the highest closeness
Hierarchical clustering: an example

**Figure:** Hierarchical clustering example graph with nodes 1, 2, 3, and 4 connected by weighted edges.

- Edge weights: 30, 25, 15, 10, 10, 10
Hierarchical clustering : an example
Hierarchical clustering: an example

![Hierarchical clustering diagram](image-url)
Hierarchical clustering: an example
Hierarchical clustering: an example
Hierarchical clustering: observations

- builds a tree of partition objects
- the (horizontal) cutlines are partition candidates
- the result is often used as start for iterative algorithms
Iterative algorithms

Properties

- start with a complete partition
- modify it iteratively
- always have a complete partition
- can evaluate the cost function
Simulated annealing: analogy with crystals

Formation of a crystal

- melting the metal lets the atoms wander randomly
- as the temperature decreases, the moves become smaller
- slower cooling helps find states of lower internal energy: crystals form
Simulated annealing: how it works

**Iterate**
- change the solution by one random "move" (moving one object from hw to sw or vice-versa)
- "downhill" moves are always allowed
- "uphill" moves are allowed or rejected, depending on the temperature

**Until**
condition on temperature or cost has been reached
Simulated annealing: an example run

\[ f \]

possible moves

design space

T
Simulated annealing: an example run

possible moves vs design space
Simulated annealing: an example run

\[ f \]

possible moves  design space

T
Simulated annealing: an example run

possible moves  design space
Simulated annealing: transition probabilities

Definition: transition probability

The probability $P(E, E', T)$ of making the transition from the current state $s$ with energy $E$ to a candidate new state $s'$ with energy $E'$ is defined as follows:

$$P(E, E', T) = \begin{cases} 1 & \text{if } E' < E \text{ (downhill moves allowed)} \\ e^{\frac{E - E'}{T}} & \text{otherwise} \end{cases}$$

The probability of allowing an “uphill” move decreases with the temperature, and as the energy gap rises.
Simulated annealing: observations

- to allow “uphill” movements prevents getting stuck in local minima
- as temperature decreases, a minimum will be reached
- theoretically: an absolute minimum (if T decreases “slowly enough”)
- in practice: yields a near-optimal solution in a few iterations
COSYMA : overview

**CO-SYnthesis for eMbedded micro Architectures**

- CAD tool for h/s systems
- research project mainly developed at the University of Braunschweig
- covers the special case of bi-partitioning: one software component and one hardware component
Main idea: dynamic granularity

Problem with too coarse granularity

More hardware implemented as really needed
⇒ non-optimal solution as to the costs because we leave potential solutions aside.
Main idea: dynamic granularity

Problem with too fine granularity

Too many possible combinations.
⇒ inefficient exploration of the design space: it is impossible to explore all the possibilities!
First step: base granularity

- build a functional graph of the application at the instruction level (finest possible)
- add information about nesting level
Base granularity: an example

```
a=... for

b=...

if

endif

endfor
```
Base granularity: an example
Second step: partitioning objects

- build consistent partitioning objects
- they will regroup whole control constructs
- this will eliminate "unrealistic" solutions
Partitioning objects: an example

- Constructive algorithms: hierarchical clustering
- Iterative algorithms: simulated annealing
- The partitioning algorithm in the COSYMA system
Partitioning objects: an example
Partitioning objects: an example
Third step: macro instructions

- change the current solution by moving a p.o.
- moves are generated using simulated annealing
Macro instructions: an example
Macro instructions: an example
Observations on the algorithm

- cost function computed incrementally
- nesting level $\sim$ closeness in clustering algorithm
- dynamic granularity speeds up computation without leaving aside potentially good solutions
- does not take parallel execution into account
Summary

- Definition of hardware/software partitioning
- Constructive algorithms: hierarchical clustering
- Transformative algorithms: simulated annealing
- A case study: COSYMA
Conclusions

H/S co-design...
- is a valuable and well-spread technique
- needs reliable CAD tools
- most stages are now well handled
- but there remain some difficult problems

H/S partitioning...
- can help reduce costs if efficiently done
- is a complex problem
- but efficient techniques are emerging
**Conclusions**

<table>
<thead>
<tr>
<th>H/S co-design...</th>
<th>H/S partitioning...</th>
</tr>
</thead>
<tbody>
<tr>
<td>- is a valuable and well-spread technique</td>
<td>- can help reduce costs if efficiently done</td>
</tr>
<tr>
<td>- needs reliable CAD tools</td>
<td>- is a complex problem</td>
</tr>
<tr>
<td>- most stages are now well handled</td>
<td>- but efficient techniques are emerging</td>
</tr>
<tr>
<td>- but there remain some difficult problems</td>
<td></td>
</tr>
</tbody>
</table>
The end

Thank you for your attention!
Questions?