From Temporal Partitioning and Temporal Placement to Algorithmic Skeletons

Florian Dittmann, Franz J. Rammig
Heinz Nixdorf Institute
University of Paderborn, Germany
Motivation

- Making reconfigurable computing mature
  - Industrialization

- Capabilities
  - Processing in parallel
  - Runtime reconfiguration
  - Partial reconfiguration
  - Space and time
  - Etc.

- Abstraction
  - Layers
  - Beneficial methods
Overview

- Motivation ✓

- Partitioning methods and their application
  - “Lessons learned”

  - Layered approach
    - Specification Graph Approach
    - Reconfiguration Port Scheduling
    - Algorithmic Skeletons

- Cooperation

- Part-E

- Conclusion
Partitioning

- Applying the spectral method on coarse grained systems
  - Mesh-based nearest neighbor communication
  - 2D topology

- Mapping
  - Data flow graphs
  - Resource efficient
  - Communication optimized
Partitioning
Temporal + Spatial Partitioning/Placement

Basis
- ASAP scheduling
- Spectral placement

Combination
- Focusing on one level
- Location of the nodes in the spectral placement
- Placement of the extracted nodes on PE

Benefit
- ASAP: precedence constraints
- Spectral Method: overall closeness respected
Partitioning
Two Slot Model

- Execution Environment of exactly two slots
  - Alternating execution of tasks
  - Hiding of the reconfiguration overhead

- Exploiting partial run-time reconfiguration

- Challenges
  - Architecture demands for communication infrastructure
  - Partial bitstream generation
  - Task mapping
  - Partitioning
Two Slot Model

- Partitioning of the input algorithms

- Scheduling
  - Simple dispatching
  - Single server for two machines
Lessons Learned

- Partitioning as reasonable/fundamental step

- Challenges
  - Placement: fragmentation
  - Reconfiguration overhead
  - Etc.

- Valuable concept
  - Two phases
    - Reconfiguration phase \( RT \)
    - Execution phase \( EX \)

- Derivable concepts
  - Specification graph approach
    - Domain of platform-based design
  - Reconfiguration Port Scheduling
  - Algorithmic Skeletons
Layered Model for Design Methods

- EX and RT phase
  - EX varies
  - RT varies
→ Challenging design, scheduling, etc.

- Layer model/approach
  - Abstraction
  - Specification

Example:

<table>
<thead>
<tr>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partitioning</td>
</tr>
<tr>
<td>Dispatching</td>
</tr>
<tr>
<td>Execution Environment</td>
</tr>
<tr>
<td>FPGA</td>
</tr>
</tbody>
</table>
**Specification Graph Approach**

- **Tasks**
  - Problem graph
  - Integration of the RT phase

- **Processing units**
  - Architecture graph
  - Also heterogeneous

- **Mapping**
  - Links tasks, communication and reconfiguration with architectural resources
  - Hierarchical mapping edges

- **Synthesis**
  - Scheduling, allocation and binding
  - By evolutionary computing
Tasks → Problem Graph

- Task graph
  → G
- Add reconfiguration phases
  → G*
- Add Communication Vertexes
  → G_p
  (Problem Graph)

<table>
<thead>
<tr>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem Graph</td>
</tr>
<tr>
<td>Mapping</td>
</tr>
<tr>
<td>Architecture Graph</td>
</tr>
<tr>
<td>FPGA</td>
</tr>
<tr>
<td>CPU</td>
</tr>
<tr>
<td>FPGA</td>
</tr>
</tbody>
</table>
Specification Graph Approach
Mapping of Resources

problem and architecture graph
+ mapping edges
= specification graph

Diagram showing the relationships between resources and tasks, with delays indicated on the edges.
Specification Graph Approach

Extensions

- Multiple devices
- Multiple reconfiguration ports

- Platform-based design
- All within the domain of synthesis
**Reconfiguration Port Scheduling**

- Partially run-time reconfigurable FPGAs for real-time processing
  - Task set executed on FPGA

- Area assignment?
  - Prevent fragmentation
  - Offer communication

- Scheduling?
  - Execution time of tasks
  - Reconfiguration process
    - Overhead: time + single port
    - At a pace of the environment

→ Reconfiguration port scheduling
Reconfiguration Port Scheduling

- Slotted execution environment

- Real-time processing on slotted FPGA architecture
  - Guarantee meeting of deadlines
  - Constant reconfiguration phase
  - Deadline $d^*$
  - One reconfiguration port

→ mono processor scheduling algorithms
Performance of $d^*$

- For aperiodic task sets
  - $d^*$ outperforms $d$
  - Of 100 feasible task sets
    - $d^*$ finds approx. 90 and $d$ finds approx. 70

Performance depends on
- ratio $EX$ to $RT$
- # of slots

\[
\frac{t_{EX}}{t_{RT}} = l
\]
**Fixed Priority Example**

- **Periodic task set scheduling**
  - Static priorities
  - Preemption

![Diagram of fixed priority example with tasks scheduled over time slots](image)

- **Characteristics**
  - Deadlines ($D^*$) shorter than periods
  → Apply deadline monotonic scheduling (DM)
Fixed Priority Scheduling
Schedulability Analysis

- Parameters

- Response time analysis
  - DM with \( D^* \):
    \[
    \forall \tau_i : R_i < D_i^*
    \]
  - Schedulable if:
    \[
    R_i = t_{RT,i} + \sum_{j=1}^{i-1} \left[ \frac{R_i}{T_j} \right] \cdot t_{RT,j}
    \]
  - Critical instance: all tasks are released simultaneously

- Sufficient and necessary for DM with \( D \)
  - Challenging abnormalities for DM with \( D^* \)
Algorithmic Skeletons

Motivation

- **Tasks**
  - Programmability and portability
  - Structure and behavior of the tasks
  → Application level

- **Use of algorithmic skeletons**
  - Wrapping of tasks
  - Programming templates

- **Runtime environment**
  - Partial reconfiguration capabilities

- **Dispatcher**
Algorithmic Skeletons

Background

- Invented for the parallel processing domain
  - First discussed by Murray Cole in the mid 80ies

- Objectives
  - Separate structure of a computation from the computation itself
  - Free programmer from the implementation details of the structure
  - Implementation guideline for activities and their interactions

- Related: design patterns
  - Differences
    - Design level
    - Final implementation left to the freedom of the designer

- Algorithmic skeletons force the applications to be well-formed
  - Enable to extract valuable information
  - Design space exploration on a high level of abstraction
  - Static and dynamic optimization of implementations
Dynamic Reconfiguration

- **Multi-threading on an FPGA**
  - Hosting of more than one task
  - Share processing resource

- **Challenges**
  - Dispatch newly arriving tasks during run-time
    - Tasks are not known at design time
    - Architecture to facilitate dispatching must exist
  - Area assignment
  - Prevent fragmentation
  - Communication assignment

→ **Algorithmic Skeletons**

<table>
<thead>
<tr>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
</tr>
<tr>
<td>Algorithmic Skeletons</td>
</tr>
<tr>
<td>Skeleton Dispatching</td>
</tr>
<tr>
<td>Runtime Environment</td>
</tr>
<tr>
<td>FPGA</td>
</tr>
</tbody>
</table>
Dynamic Reconfiguration Example

- Combination of Pipeline and Farm Skeleton
  - Slotted architecture
  - Bus and direct communication

- Pipeline:

- Farm:
Cooperation

- **PadErOl (Erlangen [Prof. Teich] and Oldenburg [Prof. Nebel])**
  - Integrated design flow, see its journal

- **Braunschweig (Prof. Fekete)**
  - Reconfiguration phase scheduling/
    single server scheduling
  - Comparison of methods

- **Erlangen (Prof. Teich)**
  - Bus-based architecture for reconfiguration port scheduling

- **Paderborn (Prof. Platzner)**
  - Local cooperation
  - Information exchange

- **Erlangen Slot Machine**
Publications

2007

2006
- Götz, Marcelo; Dittmann, Florian; Pereira, Carlos E.: Deterministic Mechanism for Run-Time Reconfiguration Activities in an RTOS. In: Proceedings of the 4th International IEEE Conference on Industrial Informatics (INDIN 2006), Singapore, 2006
Part-E

- Eclipse based development environment for partial bitstream generation
  - Open-source
  - parte.sf.net
  - Tutorial available
  - Coffee break
  - Bitstreams in 2 min
Thank you for your attention.

Florian Dittmann, Franz Rammig
Heinz Nixdorf Institute
University of Paderborn
Fuerstenallee 11
33102 Paderborn
Germany

Phone: +49 (0) 52 51/60 64 92
Fax: +49 (0) 52 51/62 65 02
Email: roichen@upb.de
http://www.heinz-nixdorf-institut.de

Thanks to E. Weber, S. Frank, A Warkentin