Temporal Partitioning and Temporal Placement Respecting Reconfiguration Times

Florian Dittmann, Franz Rammig
Heinz Nixdorf Institute
University of Paderborn, Germany
Context

- Basis: data flow graphs, task graphs
  - To be executed on reconfigurable devices
    - Realistic
    - Optimized
    - respecting the specific needs of reconfigurable systems/fabrics

- Reconfigurable Fabric
  - As resources for computation
  - Usually embedded into a more complex system

- Goal: Design methodology
  - Algorithms for partitioning, placement, execution, etc.
  - Not yet another platform
**Temporal + Spatial Partitioning/Placement**

**The Spectral Method Enhanced**

- **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
Temporal + Spatial
The Spectral Method Enhanced

- Coarse Grain Systems
- Static method
**Execution on Fine Grain Devices**

- Using FPGAs as resources for processing
  - Within a system
  - Reconfiguration time

- Fundamental model used
  - Execution time $EX$
  - Reconfiguration time $RT$

- Need for Methods to dynamically and partially reconfigure FPGAs
**Execution on Fine Grain Devices**

- **Using FPGAs as resources for processing**
  - Within a system
  - Reconfiguration time

- **Fundamental model used**
  - Execution time $EX$
  - Reconfiguration time $RT$

- **Need for Methods to dynamically and partially reconfigure FPGAs**
Overview

- **Context**
  - Spectral Method
  - Execution on Fine-Grain Devices

- **Two Slot Model**
  - Frequency adaptation

- **Real-Time Scheduling of the Reconfiguration Phase**
  - Respecting the single reconfiguration port

- **Modeling**
  - Model-Driven Generation of Partial Bitstreams

- **Cooperation**

- **Lesson Learned/Conclusion**
**Two Slot Model**

**Run Time Environment**

- **Hiding of the reconfiguration time**
  - Ideally: Processing on an FPGA becomes transparent for the designer
  - Seamless
  - Optimization of the overall response time
    - Exploit reconfiguration behavior
Two Slot Model
Partitioning

- Adapt the algorithms to match the environment
  - Data stream algorithms
  - Partitioning needed

- Criteria
  - Two slots that execute sequentially
  - Area is fixed

- Aim
  - Sequence of modules/blocks
  - Blocks of closely connected processing operations

- Partitioning investigated
Two Slot Model
Partitioning

- Adapt the algorithms to match the environment
  - Data stream algorithms
  - Partitioning needed

- Criteria
  - Tow slots that execute sequentially
  - Area is fixed

- Aim
  - Sequence of modules/blocks
  - Blocks of closely connected processing operations

- Partitioning investigated
Two Slot Model
Partitioning

- Adapt the algorithms to match the environment
  - Data stream algorithms
  - Partitioning needed

- Criteria
  - Two slots that execute sequentially
  - Area is fixed

- Aim
  - Sequence of modules/blocks
  - Blocks of closely connected processing operations

- Partitioning investigated
Two Slot Model
Partitioning

- Partitions of equal area requirements
  - No circular edges
  - Sequential ordering

- Open parameter: execution time of the partitions
  - Adapt the execution time
  - Homogeneous partitions ease to apply the pipeline behavior

- Preserve behavior
  - Overall execution time stays the same
  - Reaction time of the overall system is not affected
**Two Slot Model**

**Partitioning**

- **Partitions of equal area requirements**
  - No circular edges
  - Sequential ordering

- **Open parameter: execution time of the partitions**
  - Adapt the execution time
  - Homogeneous partitions ease to apply the pipeline behavior

- **Preserve behavior**
  - Overall execution time stays the same
  - Reaction time of the overall system is not affected
Two Slot Model
Clock Frequency Variation

- Open parameter: execution time

Slot A

Slot B

- Adaptation of execution time
  - Manipulating the execution frequency/clock
  - Preserve the overall behavior

- Two approaches
  - Multiple clock regions for adapting the frequencies
    - DCMs, etc.
    - Explicit set of frequencies needed
  - Reconfigure the clock itself
Two Slot Model
Clock Frequency Variation

- Clustering of Tasks with Different Ex. Times
  - Algorithm of organic computing area
  - Division of ant labor

- Potential for low power
Overview

- Context
  - Spectral Method
  - Execution on Fine-Grain Devices

- Two Slot Model
  - Frequency adaptation

- Real-Time Scheduling of the Reconfiguration Phase
  - Respecting the single reconfiguration port

- Modeling
  - Model-Driven Generation of Partial Bitstreams

- Cooperation

- Lesson Learned/Conclusion
Scheduling
Motivation

- **Reasonable platform for execution on FPGAs**
  - No external fragmentation
  - Placement abstracted by homogeneous slots
  - Communication via a bus

- **Dynamic loading of tasks**
  - Execution and reconfiguration time
**Scheduling Background**

- Set of $n$ independent tasks, $m$ slots, $m < n$

- How to schedule the tasks to be executed on the fabric?

  - **Reconfiguration port**
    - Mutual exclusive

  - **Known mutual exclusive device/problem:**
    - Single processor

- Similarity: Serial ‘activation’

- Explore known scheduling strategies of the single processor domain
  $\rightarrow$ EDD and EDF
### Scheduling Metric

Set of \( n \) independent tasks, \( m \) slots, \( m < n \), \textit{a-periodic}

<table>
<thead>
<tr>
<th>( d_i )</th>
<th>execution deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_{EX,i} )</td>
<td>computation or execution time</td>
</tr>
<tr>
<td>( t_{RT} )</td>
<td>reconfiguration time</td>
</tr>
<tr>
<td>( L_{max} )</td>
<td>maximum lateness</td>
</tr>
<tr>
<td>( d_i^* )</td>
<td>reconfiguration deadline</td>
</tr>
</tbody>
</table>

→ Schedule the reconfiguration deadline \( d^* \)
Scheduling
Earliest Due Date

- **Synchronous arrival times**
  - aperiodic, no dependencies

- **Jackson‘s algorithm**
  - Execute in the order of non-decreasing deadlines
  - Use deadline $d^*$

```
Slot 1  RT  EX
Slot 2  RT  EX  RT  EX
Slot 3  RT  EX  RT  EX
```
Scheduling
Earliest Due Date

- Jackson’s algorithm
  - Can miss to produce feasible schedule
  - $d = \text{deadline of the task (RT + EX)}$
  - $d^* = \text{deadline of the reconfiguration time (d – EX)}$

Diagram showing scheduling with EDD algorithm.
Scheduling
Earliest Due Date

- **Guarantee test**
  - Can be calculated using a vector

- **Feasibility analysis**
  - Overall execution time
  - Verified during guarantee test

\[
\begin{pmatrix}
  z_1 \\
  z_2 \\
  z_3
\end{pmatrix}
: \begin{pmatrix}
  0 \\
  0 \\
  0
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  t_{RT,1} + t_{EX,1} \\
  t_{RT,1}
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  t_{RT,1} + t_{EX,1} \\
  t_{RT,1} + t_{RT,2} + t_{EX,2} \\
  t_{RT,1} + t_{RT,2}
\end{pmatrix}
\rightarrow
\begin{pmatrix}
  t_{RT,1} + t_{RT,2} + t_{EX,2} + t_{RT,4} \\
  t_{RT,1} + t_{RT,2} + t_{EX,2} + t_{RT,4} + t_{EX,4} \\
  t_{RT,1} + t_{RT,2} + t_{EX,2} + t_{RT,4}
\end{pmatrix}
\]
Scheduling
Earliest Deadline First

- Asynchronous arrival times
  - Non-periodic, no dependencies

- EDF
  - Earliest deadline first

- Preemption or NP-complete
  - Preemption of executing tasks is challenging
    - Save states
    - Consumes approx. twice the reconfiguration time
  - Preemption of the reconfiguration phase
    - No states
    - Eases scheduling
    - Challenging to realize in practice
    - Additional overhead is acceptable
Scheduling
Earliest Deadline First

- **EDF example**
  - \( r \) = release time
  - \( d \) = deadline of the task (RT + EX)
  - \( d^* \) = deadline of the reconfiguration time (d – EX)

```
Slot 1
  RT₁  EX₁

Slot 2
  R   T  EX₂

Slot 3
  RT₃  EX₃
```
**Scheduling**

*Earliest Deadline First*

- **EDF example 2**
  - On preemption
    - Restart of whole reconfiguration phase is possible
    - Affects the behavior of the system

![Diagram showing scheduling example](image)
Scheduling Summary

- Proximity to single processor scheduling
  - Mutual exclusive reconfiguration port

- Preemption
  - RT phase

- Currently
  - Period
  - Dependability
  - Etc.
Overview

- Context
  - Spectral Method
  - Execution on Fine-Grain Devices

- Two Slot Model
  - Frequency adaptation

- Real-Time Scheduling of the Reconfiguration Phase
  - Respecting the single reconfiguration port

- Modeling
  - Model-Driven Generation of Partial Bitstreams

- Cooperation

- Lesson Learned/Conclusion
Modeling
UML class diagram

- Three basic classes
- Follows and extends XAPP 290
Modeling
UML class diagram

- Three basic classes
- Follows and extends XAPP 290
**Modeling**
**Tool: PART-E**

- **XAPP290 → PART-Y → PART-E**
Overview

- Context
  - Spectral Method
  - Execution on Fine-Grain Devices

- Two Slot Model
  - Frequency adaptation

- Real-Time Scheduling of the Reconfiguration Phase
  - Respecting the single reconfiguration port

- Modeling
  - Model-Driven Generation of Partial Bitstreams

- Cooperation

- Lesson Learned/Conclusion
PadErOl-Cooperation

Ziel: Ein durchgängiger Entwurfsfluss für dynamisch rekonfigurierbare Systeme

Momentan: Evaluierung anhand eines dynamisch rekonfigurierbaren Kryptographie-Systems
Overview

- **Context**
  - Spectral Method
  - Execution on Fine-Grain Devices

- **Two Slot Model**
  - Frequency adaptation

- **Real-Time Scheduling of the Reconfiguration Phase**
  - Respecting the single reconfiguration port

- **Modeling**
  - Model-Driven Generation of Partial Bitstreams

- **Cooperation**

- **Lesson Learned/Conclusion**
Lesson Learned

- Processing on reconfigurable devices
  - Partitioning
    - Temporal/spatial using the spectral method
  - Execution
    - Reconfiguration phase
    - Execution phase
  - Frequency adaptation
  - Scheduling of the reconfiguration phase

- Modeling/Realization
  - Model driven
  - Eclipse based tool

- Cooperation
Meaningful Publications


Thank you for your attention.