Automatic Extraction of Multi-Objective Aware Parallelism for Heterogeneous MPSoCs

Daniel Cordes, Michael Engel, Olaf Neugebauer, Peter Marwedel

TU Dortmund University
Department of Computer Science 12
Dortmund
Germany
Introduction

- Complexity of embedded software increasing
  - Multiple cores available in MPSoCs
  - Manual parallelization error-prone & time-consuming

- Heterogeneous MPSoCs can be more efficient than homogeneous
  - Cores behave differently on same parts of application
  - Efficient balancing of tasks very difficult

- Restrictions of embedded architectures
  - Low computational power, small memories, battery-driven
  - Good trade-off between different objectives must be found

→ Multi-objective aware approach based on Genetic Algorithms to extract parallelism for heterogeneous MPSoCs presented here
Outline

1. Motivating Example
2. Parallelization Methodology
3. GA-based Parallelization Approach
4. Results
5. Conclusion & Future Work
big.LITTLE architecture (ARM)

- Performance vs. power of ARM’s big.LITTLE architecture
- Many trade-offs possible
Example – Homogeneous architecture

- Timing example of parallelized edge-detect application (UTDSP benchmark suite)
- Execution on homogeneous architecture
- Well-balanced solution
Example – Heterogeneous architecture

- Same solution on heterogeneous architecture
- Faster cores not well utilized
- Unbalanced solution with a lot of wasted performance
Example – Heterogeneous architecture

- Well-balanced solution
- Increased performance of cores utilized
- But: Other solutions might also be better for other objectives!
Question...

How can we extract multi-objective aware parallelism for heterogeneous MPSoCs automatically?

- Input: Sequential C-Code
- Output: Parallelized application

- Used techniques:
  - Annotated Hierarchical Task Graph (AHTG)
  - Genetic Algorithms
  - High-Level Evaluation Models

- Goal:
  - Extract Task and Pipeline Parallelism
Parallelization Methodology

1. Extract ICD-C IR and Augmented Hierarchical Task Graph (AHTG)

Extraction techniques for homogeneous MPSoCs already exist:

- **DATE’12**: Multi-Objective Aware Extraction of Task-Level Parallelism Using Genetic Algorithms.
- **CODES’12**: Automatic extraction of multi-objective aware pipeline parallelism using genetic algorithms.

2. Process nodes bottom-up

3. Apply parallelization approaches

4. Create a front of Pareto-optimal solution candidates with different approaches

5. Attach solution candidates and continue with other nodes

6. Select (best) solution candidate as final result

7. Annotate Source Code

8. Implement parallelism
GA-Approach: Task-Level Chromosome

Node-to-Task Mapping

<table>
<thead>
<tr>
<th>$T_1$</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$T_3$</th>
<th>...</th>
<th>$T_4$</th>
</tr>
</thead>
</table>

Node $1..n$ → Node-to-Task Mapping

Hierarchical Parallel Solution

<table>
<thead>
<tr>
<th>$S_{1,4}$</th>
<th>$S_{2,3}$</th>
<th>$S_{3,2}$</th>
<th>$S_{4,8}$</th>
<th>...</th>
<th>$S_{n,4}$</th>
</tr>
</thead>
</table>

Node $1..n$ → Hierarchical Parallel Solution

Task-to-Processor-Class Mapping

<table>
<thead>
<tr>
<th>$P_1$</th>
<th>$P_2$</th>
<th>...</th>
<th>$P_1$</th>
</tr>
</thead>
</table>

Task $1..i$ → Task-to-Processor-Class Mapping

Chromosome Representation

Task Graph

Node $1..n$ → Node-to-Task Mapping
GA-Approach: Task-Level Chromosome

Node-to-Task Mapping

Task 1 - T1  T1  T2  T3  ...  T4

Hierarchical Parallel Solution

S1,4  S2,3  S3,2  S4,8  ...  Sn,4

Task-to-Processor-Class Mapping

P1  P2  ...  P1

Node-to-Task Mapping

Node1..n

Hierarchical Parallel Solution

Node1..n

Task-to-Processor-Class Mapping

Task1..i
GA-Approach: Task-Level Chromosome
### GA-Approach: Pipeline Chromosome

#### Node-to-Pipeline Mapping

<table>
<thead>
<tr>
<th>T₁</th>
<th>T₁</th>
<th>T₂</th>
<th>T₃</th>
<th>...</th>
<th>T₄</th>
</tr>
</thead>
</table>

← Stage₁..ₕ x Subtasks₁..ₙ →

#### Sub-Tasks Used

<table>
<thead>
<tr>
<th>U₁,₁</th>
<th>U₁,₂</th>
<th>...</th>
<th>Uₕ,t</th>
</tr>
</thead>
</table>

← Stage₁..ₕ x Subtasks₁..ₙ →

#### Chunk Sizes of Sub-Tasks

<table>
<thead>
<tr>
<th>C₁,₁</th>
<th>C₁,₂</th>
<th>...</th>
<th>Cₕ,t</th>
</tr>
</thead>
</table>

← Stage₁..ₕ x Subtasks₁..ₙ →

#### Sub-Task to Processor Class

<table>
<thead>
<tr>
<th>P₁</th>
<th>P₂</th>
<th>...</th>
<th>Pₕ</th>
<th>PT</th>
</tr>
</thead>
</table>

← Stage₁..ₕ x Subtasks₁..ₙ →

← 1 →

#### Chromosome Representation

- **Node-to-Pipeline**:
  - T₁
  - T₂

- **Task Graph**:
  - T₁
  - N₁
  - N₂
  - T₂
  - N₃

2 Nodes in Stage 1, 1 Node in Stage 2
GA-Approach: Pipeline Chromosome

Node-to-Pipeline Mapping

Sub-Tasks Used

Chunk Sizes of Sub-Tasks

Sub-Task to Processor Class

Scheduling

Node-to-Pipeline Mapping

Sub-Tasks Used

Chunk Sizes of Sub-Tasks

Sub-Task to Processor Class

Scheduling

Chromosome Representation

Sub-Tasks Used

Task Graph

3 Subtasks of Pipeline 1, 1 Subtask of Pipeline 2

Unused
## GA-Approach: Pipeline Chromosome

<table>
<thead>
<tr>
<th>Node-to-Pipeline Mapping</th>
<th>Sub-Tasks Used</th>
<th>Chunk Sizes of Sub-Tasks</th>
<th>Sub-Task to Processor Class</th>
<th>Scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁ T₁ T₂ T₃ ... T₄</td>
<td>U₁,₁ U₁,₂ ... Uₚ,t</td>
<td>C₁,₁ C₁,₂ ... Cₛ,t</td>
<td>P₁ P₂ ... Pₛ</td>
<td>PT</td>
</tr>
</tbody>
</table>

- **Node-to-Pipeline Mapping**: Node₁..n
- **Sub-Tasks Used**: Stage₁..s x Subtasks₁..t
- **Chunk Sizes of Sub-Tasks**: Stage₁..s x Subtasks₁..t
- **Sub-Task to Processor Class**: Stage₁..s x Subtasks₁..t
- **Scheduling**: Stage₁..s x Subtasks₁..t

### Chromosome Representation

- **Chunk Sizes of Sub-Tasks**
  - C₁,₁: 6
  - C₁,₂: 4
  - C₁,₃: 2
  - C₂,₁: 12
  - C₂,₂: 2
  - C₂,₃: 6

- ** Chunk Sizes**
  - T₁,₁
  - T₁,₂
  - T₁,₃
  - T₂,₁
  - T₂,₂
  - T₂,₃

- **Unused**: unused sub-tasks
GA-Approach: Pipeline Chromosome

Node-to-Pipeline Mapping

<table>
<thead>
<tr>
<th>Node 1</th>
<th>T₁</th>
<th>T₂</th>
<th>T₃</th>
<th>...</th>
<th>T₄</th>
</tr>
</thead>
</table>

Sub-Tasks Used

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>Subtasks 1..t</th>
<th>Stage 2</th>
<th>Subtasks 1..t</th>
<th>Stage 3</th>
<th>Subtasks 1..t</th>
<th>Stage 4</th>
<th>Subtasks 1..t</th>
</tr>
</thead>
<tbody>
<tr>
<td>U₁,₁</td>
<td></td>
<td>C₁,₁</td>
<td></td>
<td>P₁</td>
<td></td>
<td>PT</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C₁,₂</td>
<td></td>
<td>P₂</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C₅,₁</td>
<td></td>
<td>P₂</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C₅,₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>C₅,t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Chunk Sizes of Sub-Tasks

<table>
<thead>
<tr>
<th>Stage 1</th>
<th>Subtasks 1..t</th>
<th>Stage 2</th>
<th>Subtasks 1..t</th>
<th>Stage 3</th>
<th>Subtasks 1..t</th>
<th>Stage 4</th>
<th>Subtasks 1..t</th>
</tr>
</thead>
<tbody>
<tr>
<td>P₁</td>
<td></td>
<td>P₂</td>
<td></td>
<td>P₂</td>
<td></td>
<td>P₂</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sub-Task to Processor Class Mapping

<table>
<thead>
<tr>
<th>Processor Class Mapping</th>
<th>ProcClass C₁</th>
<th>ProcClass C₂</th>
<th>ProcClass C₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>T₁,₁</td>
<td>N₁</td>
<td>N₁</td>
<td>T₁,3</td>
</tr>
<tr>
<td>T₁,₂</td>
<td>N₂</td>
<td>N₂</td>
<td>N₂</td>
</tr>
<tr>
<td>T₂,₁</td>
<td>N₃</td>
<td>N₁</td>
<td>N₁</td>
</tr>
</tbody>
</table>

Iteration Scheduling:

0 = Chunk-Based, 1 = Interleaved, 2 = Fully Interleaved
Results – JPEG encoder

![Graph showing performance metrics for JPEG encoder with Pareto-optimal and dominated pipelines.](Image)

- Domated Pipeline
- Domated Task-Level
- Domated MIXED
- Pareto-Optimal Pipeline
- Pareto-Optimal Task-Level
- Pareto-Optimal MIXED

**Axes:**
- Communication factor [bytes]
- Energy [J]
- Speedup of Exec. Time [cycles]
<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Time*</th>
<th>#Nodes</th>
<th>#Populations</th>
<th>#Individuals</th>
<th>#Mutations</th>
<th>#Crossover</th>
<th>#Solutions</th>
</tr>
</thead>
<tbody>
<tr>
<td>adpcm enc.</td>
<td>01:31</td>
<td>36</td>
<td>1,520</td>
<td>153,453</td>
<td>30,729</td>
<td>104,962</td>
<td>63</td>
</tr>
<tr>
<td>bound. value</td>
<td>01:17</td>
<td>12</td>
<td>644</td>
<td>83,973</td>
<td>17,900</td>
<td>54,964</td>
<td>34</td>
</tr>
<tr>
<td>compress</td>
<td>10:59</td>
<td>289</td>
<td>8,936</td>
<td>706,266</td>
<td>141,170</td>
<td>464,112</td>
<td>30</td>
</tr>
<tr>
<td>edge detect</td>
<td>02:43</td>
<td>105</td>
<td>2,872</td>
<td>200,371</td>
<td>40,002</td>
<td>133,096</td>
<td>118</td>
</tr>
<tr>
<td>filterbank</td>
<td>03:24</td>
<td>7</td>
<td>412</td>
<td>50,779</td>
<td>12,229</td>
<td>44,164</td>
<td>47</td>
</tr>
<tr>
<td>fir 256</td>
<td>00:30</td>
<td>13</td>
<td>388</td>
<td>29,889</td>
<td>6,629</td>
<td>21,515</td>
<td>44</td>
</tr>
<tr>
<td>iir 4</td>
<td>03:02</td>
<td>13</td>
<td>852</td>
<td>105,224</td>
<td>22,631</td>
<td>79,206</td>
<td>63</td>
</tr>
<tr>
<td>jpeg 2000</td>
<td>05:13</td>
<td>62</td>
<td>2,868</td>
<td>312,242</td>
<td>68,142</td>
<td>246,427</td>
<td>333</td>
</tr>
<tr>
<td>latnrm 32</td>
<td>01:11</td>
<td>17</td>
<td>636</td>
<td>53,642</td>
<td>11,462</td>
<td>39,831</td>
<td>54</td>
</tr>
<tr>
<td>mult 10</td>
<td>01:01</td>
<td>36</td>
<td>1,060</td>
<td>70,855</td>
<td>14,635</td>
<td>57,226</td>
<td>90</td>
</tr>
<tr>
<td>spectral</td>
<td>02:25</td>
<td>51</td>
<td>2,260</td>
<td>213,230</td>
<td>44,696</td>
<td>158,624</td>
<td>114</td>
</tr>
<tr>
<td>average</td>
<td>03:01</td>
<td>58</td>
<td>2,041</td>
<td>179,993</td>
<td>37,293</td>
<td>127,648</td>
<td>90</td>
</tr>
</tbody>
</table>

* AMD Opteron @ 2.4 GHz

Most benchmarks processed in 1-2 minutes

213k created and evaluated individuals

Number of solutions range from 43 - 333

Overall 2,5 minutes: 0.6 milli sec. / individual
Conclusions & Future Work

Conclusion

- New multi-objective aware parallelization approaches for heterogeneous embedded systems presented
- Huge optimization potential
  - High speedups with pipeline technique
  - Less energy consumption with task-level technique
  - Good trace-offs with combination

Future Work

- Integrate DFS/DVS in models
- Integrate other objectives, like, e.g., memory consumption
Thank you for your attention!

Questions?