Introduction

Motivations
- Quantifying partial reconfiguration (PR) benefits is not straightforward
- Many possible PR architectures and layouts on device
- Manual and tedious characterization, analysis, and evaluation process
- Early design decision identification reduces design time effort
- Formulation-level analysis affords short design exploration time

Approach
- PR design space exploration in early design phases
  - Analyze application’s high-level source code to find Pareto optimal PR architectures
  - Evaluate target device feasibility of PR architectures
- PR modeling language (PRML) and partitioning
  - PRML models application’s algorithm based on the application components’ control and data dependence
  - Leverages advanced graph-theoretic techniques
  - Application-behavior-independent partitioning rules

Case Study Application
- Complex arithmetic core
  - Performs addition, subtraction, multiplication, division, and square root operation on complex number input
  - Figure 1 shows PRML model created with XML-based yED diagram editor tool

Results and Analysis
- Partitioning rules (Table 1) generated 390 PR architectures
  - PR architectures have up to three PR regions
  - Formulation-level PR design space exploration tool (published in FPT ’11) performs tradeoff analysis based on three metrics (Figure 2)
- Percentage change in longest path delay, actual resource savings, and PR overhead
- Tradeoff analysis enables designers to carefully select a PR architecture based on system goals
  - PR architectures with high actual resource savings (e.g., 128 and 129) can fit on a smaller device
  - PR architectures with low longest path delay (e.g., 6 and 13) afford high performance

PR Modeling Language (PRML)

- Early PR benefit evaluation reduces application’s PR design time efforts
  - Correct design decisions reduce designer effort and increase PR benefits
  - Algorithmic-level model evaluations require nominal effort

<table>
<thead>
<tr>
<th>Fundamental Partitioning Rules</th>
<th>Execution results</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Eliminate hierarchy nodes and memory nodes inside the hierarchy nodes</td>
<td>Eliminates redundant memory nodes by flattening the PRML model.</td>
</tr>
<tr>
<td>2. Identify computation and iteration supernode(s)</td>
<td>Reduces the number of nodes by merging interdependent nodes.</td>
</tr>
<tr>
<td>3. Identify all execution paths/cycles except symbol paths/cycles and trivial paths (i.e., L1 paths)</td>
<td>Identifies all non-trivial input to output paths.</td>
</tr>
<tr>
<td>4. Identify distinct smaller paths (i.e., L2 paths) from the L1 paths (sequentially break the L1 paths at choice and or-merge nodes but exclude symbol paths and trivial paths)</td>
<td>Identifies smaller data paths from the non-trivial input to output paths based on control choices.</td>
</tr>
<tr>
<td>5. Identify distinct smaller paths (i.e., L3 paths) from the L2 paths (break the L2 paths at iteration nodes and iteration supernodes but exclude trivial paths)</td>
<td>Identifies all computation kernels.</td>
</tr>
<tr>
<td>6. Identify all sets of static module and PRMs based on L2 paths, L3 paths, and node’s divergent attribute value</td>
<td>Identifies all possible path combinations considering paths generated by rules 3-5, divides these paths into the PRMs and the static module.</td>
</tr>
<tr>
<td>7. Assign PRMs to PRRs: (a) clone PRMs are assigned to the same PRR; (b) sibling PRMs are assigned to different PRRs; (c) cousin PRMs can be assigned to the same or different PRRs</td>
<td>Calculates the number of PRRs required for each combination generated by rule 6 and creates all possible PRM to PRR assignments.</td>
</tr>
<tr>
<td>8. Create PR architectures.</td>
<td>Different PR architectures are created for each PRM variant and each PRR to PRR assignment.</td>
</tr>
</tbody>
</table>

Table 1. Fundamental partitioning rules and brief description of the rules’ execution results after the rule is applied to the complex arithmetic core’s PRML model.

Case Study Application
- Complex arithmetic core
  - Performs addition, subtraction, multiplication, division, and square root operation on complex number input
  - Figure 1 shows PRML model created with XML-based yED diagram editor tool

Results and Analysis
- Partitioning rules (Table 1) generated 390 PR architectures
  - PR architectures have up to three PR regions
  - Formulation-level PR design space exploration tool (published in FPT ’11) performs tradeoff analysis based on three metrics (Figure 2)
- Percentage change in longest path delay, actual resource savings, and PR overhead
- Tradeoff analysis enables designers to carefully select a PR architecture based on system goals
  - PR architectures with high actual resource savings (e.g., 128 and 129) can fit on a smaller device
  - PR architectures with low longest path delay (e.g., 6 and 13) afford high performance

Future Work
Partitioning enhancements using an iterative process that incorporates feedback from tradeoff analysis and the PR application’s runtime performance throughput to repartition application