# PRML: A Modeling Language for Rapid Design Space Exploration of Partially Reconfigurable FPGAs

CHREC NSF Center for High-Performance Reconfigurable Computing

Rohit Kumar
Dr. Ann Gordon-Ross

UF UNIVERSITY of FLORIDA

## 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

## 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
- PRML provides application-behavior-independent graph-theoretic techniques for application partitioning and analysis
  - Table 1 shows fundamental partitioning rules and rules' execution results when applied to an application's PRML model
  - Partitioning rules depends only on structural properties of graph

**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.

| Fundamental Partitioning Rules | Execution results |
|---|---|
| **1.** Eliminate hierarchy nodes and memory nodes inside the hierarchy nodes | Eliminates redundant memory nodes by flattening the PRML model. |
| **2.** Identify computation and iteration supernode(s) | Reduces the number of nodes by merging interdependent nodes. |
| **3.** Identify all execution paths/cycles except symbol paths/cycles and trivial paths (i.e., L1 paths) | Identifies all non-trivial input to output paths. |
| **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) | Identifies smaller data paths from the non-trivial input to output paths based on control choices. |
| **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) | Identifies all computation kernels. |
| **6.** Identify all sets of static module and PRMs based on L2 paths, L3 paths, and node's divergent attribute value | Identifies all possible path combinations considering paths generated by rules 3-5, divides these paths into the PRMs and the static module. |
| **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 | Calculates the number of PRRs required for each combination generated by rule 6 and creates all possible PRM to PRR assignments. |
| **8.** Create PR architectures. | Different PR architectures are created for each PRM variant and each PRM to PRR assignment. |

### 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

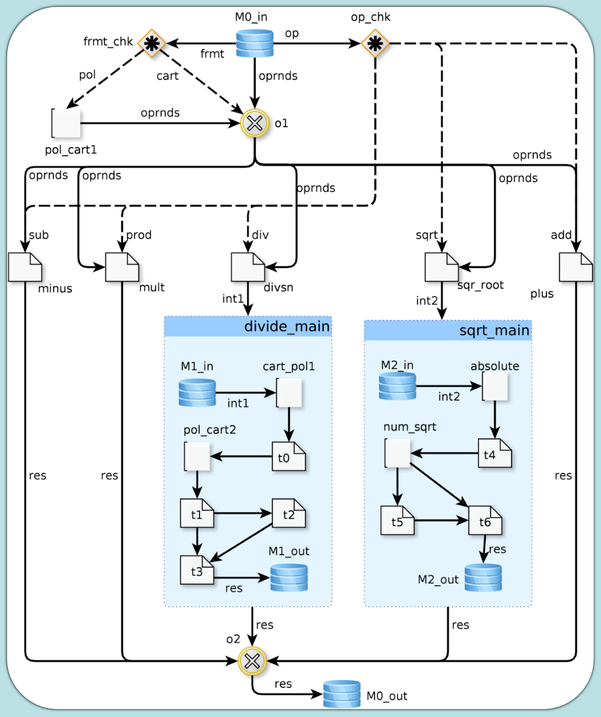


**Figure 1.** PRML model of a complex arithmetic core that performs a set of arithmetic operations (add, subtract, multiplication, division, square root) for complex number operands represented in polar or Cartesian format

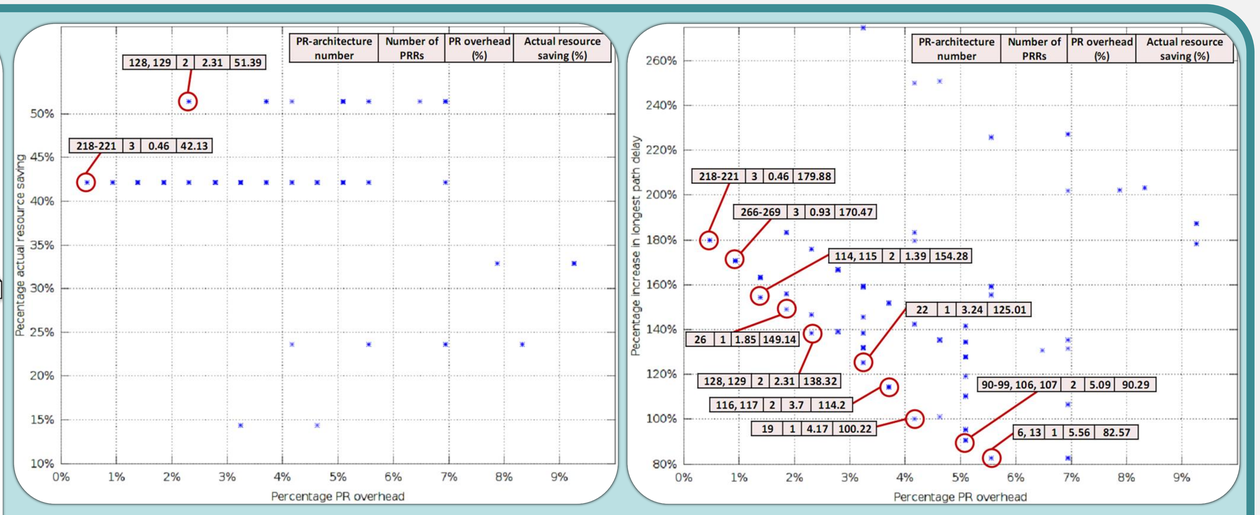


**Figure 2.** (left) Percentage of actual resource savings with respect to PR overhead for each PR-architecture; (right) Percentage increase in longest path delay with respect to PR overhead for each PR-architecture. Pareto optimal PR-architectures are circled. The boxes attached to circles show the Pareto optimal PR-architectures, number of PRRs, percentage PR overhead, and percentage actual resource saving (top) or percentage increase in longest path delay (right).

### 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**