Introduction

In CGRAs (Coarse-Grained Reconfigurable Architectures), its configuration time is sometimes an obstacle for efficient computing. In this work, we propose a novel configuration data compression technique based on a multicast configuration scheme called RoMultic. In addition, two types of scheduling algorithms for the proposed technique are also proposed. Experimental results show that the proposed method achieves power reduction as well as area reduction.

Configuration Multicasting

- **CMA (Cool Mega Array)** [1]
- **PE (Processing Element)**
  - Composed of
    1. Simple ALU
    2. Switching Element (SE)
- **PE Array**
  - 12 cols × 8 rows of PEs
- For a scalable system
  - Multiple CMA chips can be connected with a host CPU
  - Each component for each chip is mapped to same address space as the host CPU
  - Configuration data is transferred via a general purpose data bus
  - Configuration time is a concern

Multicasting Scheme

- **RoMultic** [2]
  - Multicasting data to PEs which have the same config.
  - Using two bit-maps for PE rows & columns
- **Overwriting Scheduling**
  - Possible to reduce more data
  - High priority to multicasting widely distributed configs.
- **Configuration data format (20bit)**
  - 2 types of multicasting in the original CMA
    1. ALU-by-ALU
    2. SE-by-SE
  - Because of not enough data space (12bit)
  - Unused 2 bits space
  - Multicastd ALUs or SEs must have a completely same config.
  - Less possibility of data reduction for complex config.

Proposed Method

**Fine-grain Multicasting**

- Any combination of configuration fields available
  - Need for additional flag bits
  - Mapped to global address space
- **Scheduling Algorithms**
  - Greedy Scheduling
    - In order of largest multicasted configuration data
  - How to find configuration patterns
    1. FGM-E: Espresso-based Fine-Grain Multicasting
      - Not always optimal but fast algorithm
      - Using a heuristic logic minimization algorithm: Espresso
      - Supporting Don't Care X as well as 1 and 0
      - For each unfixed configuration, making a truth table
      - Logic 0: already fixed parts (to prevent them from being overwritten)
      - Logic 1: same config. parts
      - Don't care X: others
      - Target configuration (As are fixed)

**Evaluation**

- **Reduction Ratio**
  - For randomly generated configs.
  - FGM-I is better for large configs.
  - FGM-I achieves around 20 % reduction regardless of the size
  - For config. of image process apps.
    - FGM-I: 11.7% , FGM-I: 19.14% (Avg.)
- **Execution time**
  - FGM-I takes quite a long time
  - FGM-E is finished within 1 min for any size of the configuration
- **Area Overhead and Power consumption**
  - The proposed method can be implemented with a negligible area overhead
  - It can reduce the dynamic power thanks to minimum overwriting


