Overview

In accelerating dynamic multi-core systems with Coarse Grained Reconfigurable Arrays (CGRAs), offloading and mapping multiple kernels to a CGRA at run time can help to achieve higher performance. In this work, we introduce a novel CGRA programming methodology to configure the CGRA at run time in a multi-core system to exploit opportunities for acceleration.

Introduction

- Embedded systems are seeing increased computation demands.
- Multi-core CPUs can help to achieve energy-efficient performance.
- CGRAs provide reconfigurability with a much larger potential throughput and energy efficiency.

- Limitations of traditional CGRAs
  - Typically paired to a single processor.
  - Can have a long compilation time (seconds to hours).

  - Our approach: Tatami – a flexible methodology for near-optimal floorplanning
    - Upfront Processing: Tatami blocks in multiple sizes.
    - Run time Processing: Floorplanning by composing Tatami blocks.

Results

- 99% of the performance of the fully-flexible system.
- 13% higher performance over private CGRA accelerators.
- Fast run time configuration. Specifically, when 4 kernels are mapped among 12 kernels, the best performing floorplan is found in 0.5 microseconds.

Methodology

The goal of this work is to accommodate long CGRA compilation times with upfront analysis and low-impact kernel restrictions. Specifically, the time-consuming CGRA compilation is evaluated upfront and the light-weight, high-performance optimizer generates floorplans to maximize throughput.

Upfront processing

- C source code → CGRA compiler → Tatami blocks
- The blocks’ width are W/2 or W (4 or 8 in Figure 2).
- ①: 2 memory ports at each row
- ②: 1 memory port at each row on its left
- Performing compilation at run time is too time consuming; instead, we focus on the easier tasks that uses predefined compilation targets, called Tatami blocks.

Run time processing

- Floorplanning is completed in microseconds by composing Tatami blocks.
- ③: on the right half, to map a half-width Tatami block, the block using memory ports on the left (②) is rotated.
- Maximizing throughput ($\text{max} = \sum \frac{\# \text{instr.}}{T}$)

Results

- 99% of the performance of the fully-flexible system.
- 13% higher performance over private CGRA accelerators.
- Fast run time configuration. Specifically, when 4 kernels are mapped among 12 kernels, the best performing floorplan is found in 0.5 microseconds.

Storage savings over upfront compilation

In traditional upfront CGRA scheduling, for $m$-combinations of all $n$ kernels, there exist $\binom{n}{m}$ optimal floorplans for each case. Then, the total number of floorplans to be stored is $\sum_{m=0}^{n} \binom{n}{m} = 2^n$.

Throughput gain over dedicated CGRAs

We achieve higher performance as we can fully utilize the existing resources while the dedicated CGRAs are underutilizing them.