Cascade: An Application Pipelining Toolkit for Coarse-Grained Reconfigurable Arrays

Jackson Melchert, Yuchen Mei, Kalhan Koul, Qiaoyi Liu, Mark Horowitz, Priyanka Raina
Motivation

- CGRA have large performance and energy efficiency benefits over FPGAs
  - To achieve commercial viability, CGRAs must demonstrate performance and EDP that approaches ASICs

- Current CGRA compilers either do not pipeline resulting in low performance or pipeline exhaustively resulting in high power
  - They cannot pipeline applications effectively, we need a new compiler with a focus on pipelining
Background - CGRA Architecture

• Several classes of CGRA interconnects exist:
  1. Exhaustively pipelined interconnects
     • Expensive
  2. Non-pipelined interconnects
     • Cannot run at high frequencies
  3. Interconnects with configurable registers
     • Need pipelining techniques to run fast
Our Approach

• We adapt FPGA and ASIC-like pipelining and register retiming techniques to CGRAs, and propose a technique for absorbing registers into memory tiles.

• We propose a post-PnR pipelining technique that iteratively identifies the critical path in an application, breaks it, and performs rescheduling while avoiding cyclic rescheduling and PnR.

• An end-to-end CGRA compiler, called Cascade, which has an automatic CGRA timing model generator, a static timing analysis tool for CGRA applications, and a large set of existing and our proposed pipelining techniques.
Post Place and Route Pipelining

- After place and route, we know locations of tiles and nets
- Iteratively identity the critical path, break it, reschedule the application, and repeat
- We can only add registers to existing routes, so eventually we run out of pipelining opportunities
Post Place and Route Pipelining Example
Post Place and Route Pipelining Example

Stanford University
Post Place and Route Pipelining Example

Stanford University
Rescheduling CGRA Applications

Statically scheduled applications need to be updated after inserting registers

1. Statically schedule the application without registers
2. Place and route the application
3. Pipeline
4. Analyze the application and compute new latencies
5. Reschedule the application before bitstream generation

Avoids cyclic rescheduling and pipelining
Optimizing Register Resource Usage

- Registers that are not on the critical path can be removed to save energy
- Static schedules of memory tiles can be adjusted to absorb the delays
Cascade

- End-to-end application compiler that achieves high performance through many pipelining techniques
- Cascade includes:
  - Automatic CGRA timing model generator
  - Static timing analysis tool for CGRA applications
  - A large set of existing and our proposed pipelining techniques
Methodology for Generating Timing Model

Interconnect Specification

Post P&R Netlist

Timing Queries (report_timing)

Application STA Tool

All Timing Paths

Worst-Case Timing Model

IN to PE: 240ps

PE to OUT: 200ps

PE Tile

IN to PE:

PE to OUT:
Static Timing Analysis Model

Critical Path
Results - STA Model Evaluation

- STA model predicts the actual clock period accurately
  - Above 500 MHz, the average error is 13%
- All clock period errors are positive
  - Our model is pessimistic, it will provide a lower bound for the clock frequency
Results - Software Pipelining Dense Applications

![Graph showing runtime improvements for different applications with various optimization techniques.](image)
Results - Software Pipelining Dense Applications

![Chart showing EDP (μJ·s/frame) for Gaussian, Unsharp, Camera, Harris, and ResNet applications comparing unpipelined and all software pipelining. The chart shows percentage improvements: -86%, -94%, -98%, -99%, and -96%, respectively.]

Stanford University
Results - Software Pipelining Sparse Applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Compute pipelining</th>
<th>+ Placement optimization</th>
<th>+ Post-PnR pipelining</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector Add</td>
<td>-51%</td>
<td>-50%</td>
<td>-57%</td>
</tr>
<tr>
<td>Matrix Mul</td>
<td>-23%</td>
<td>-57%</td>
<td>-47%</td>
</tr>
<tr>
<td>Tensor MTTKRP</td>
<td>-60%</td>
<td>-57%</td>
<td>-57%</td>
</tr>
</tbody>
</table>

Stanford University
Results - Software Pipelining Sparse Applications

-72% for Vector Element Add
-60% for Matrix Element Mul
-71% for Tensor MTTKRP
-81% for Tensor TTV

Stanford University
Conclusion

• Cascade is an open-source end-to-end CGRA compiler that achieves high performance and low EDP

• We adapt prior work on pipelining to CGRAs, introduce novel register absorption and post-PnR pipelining techniques, and put them all together in a pipelining compiler

• 8-34x shorter critical paths and 7-190x lower EDP for dense apps
  3-5.2x shorter critical paths and 2.5-5.2x lower EDP for sparse apps