Automated Design Space Exploration of CGRA Processing Element Architectures using Frequent Subgraph Analysis

Jackson Melchert, Kathleen Feng, Caleb Donovick, Ross Daly, Clark Barrett, Mark Horowitz, Pat Hanrahan, Priyanka Raina
Motivation

How can we generate an optimal PE architecture for a specific application domain?

1. Analyze application domain benchmarks to find possible optimizations
2. Quickly create PE designs that explore the design space
3. Automatically generate full compiler to run applications
Design Space Axes

Intraconnect

Number/Types of Operations

I/O to Interconnect

Operations

Intraconnect

I/O
Example Convolution Dataflow Graph

$$((((i_0 \times w_0) + (i_1 \times w_1)) + (i_2 \times w_2)) + (i_3 \times w_3)) + c$$
Frequent Subgraphs of a Convolution

Subgraph 1
Frequency: 4

Subgraph 2
Frequency: 4

Subgraph 3
Frequency: 4
Maximal Independent Set Analysis

For each subgraph:

1. Represent each occurrence of that subgraph as a node in a new graph
2. Add an edge between nodes if the subgraph occurrences overlap
3. Calculate the maximal independent set
Maximal Independent Set Analysis Example

Subgraph 3
Frequency 4

Maximal Independent Set
MIS Size = 2
Merging Subgraphs

- Allows for exploration of the design space by tuning how many subgraphs are merged
- Enables better coverage of application graphs
- Allows for more effectively analyzing multiple applications
- Intelligently explores the connectivity design space axis
Merging Subgraphs

Datapath graph merging:

1. Create a mapping between nodes of the same operation in both subgraphs
2. Create a “compatibility graph”
3. Find the maximum weight clique of this compatibility graph
4. Finally reconstruct the resulting merged graph
Merging Subgraphs

Subgraph 1
- \text{const } a0
- + a1

Subgraph 2
- \text{const } b0
- \times b1
- + b2
- + b3
Merging Subgraphs

(a) Subgraph 1
(b) Subgraph 2
(c) Potential Mergings
Merging Subgraphs

\[
\begin{array}{c}
\text{a2,a1} & \text{b3,b2} \\
\text{a2} & \text{b3} \\
\text{a1} & \text{b2} \\
\text{a0} & \text{b0}
\end{array}
\]

\[
\begin{array}{c}
\text{a2/b3} & \text{a1/b3} \\
\text{a1/b2} & \text{a0/b0} \\
\text{a2a1/b3b2} & \text{a2/b2}
\end{array}
\]

\[
\begin{array}{c}
w = 80 \\
w = 80 \\
w = 80
\end{array}
\]

\[
\begin{array}{c}
w = 12 \\
w = 30 \\
w = 80
\end{array}
\]
Merging Subgraphs

Maximun Weight Clique
Merging Subgraphs

Subgraph 1

Subgraph 2

Reconstructed Merged Graph
Design Space Exploration Framework

Application Dataflow Graph in CoreIR -> Application Frequent Subgraph Analysis

- Subgraph Mining
- Maximal Independent Set Analysis
- Ordered List of Frequent Subgraphs
- Subgraph Merging

Merged PE Graph -> PE Hardware and Mapper Generation

- PEak Specification Generation
- PEak Compiler

PE RTL Verilog
Design Space Exploration Framework

Application Frequent Subgraph Analysis

- Subgraph Mining
- Maximal Independent Set Analysis
- Ordered List of Frequent Subgraphs
- Subgraph Merging

PE Hardware and Mapper Generation

- PEak Specification Generation
- PEak Compiler
- PE RTL Verilog

CGRA Generation and Compilation Toolchain

- Halide Compiler
- Application Dataflow Graph in CoreIR
- Rewrite Rules
- Application Mapping
- Application Place & Route
- Configuration Bitstream Generation
- Application Simulation on CGRA
- CGRA RTL Generation
- PE/CGRA Synthesis

Application Frequent Subgraph Analysis: Subgraph Mining → Maximal Independent Set Analysis → Ordered List of Frequent Subgraphs → Subgraph Merging

PE Hardware and Mapper Generation: PEak Specification Generation → PEak Compiler → PE RTL Verilog

CGRA Generation and Compilation Toolchain: Halide Compiler → Application Dataflow Graph in CoreIR → Rewrite Rules → Application Mapping → Application Place & Route → Configuration Bitstream Generation → Application Simulation on CGRA → CGRA RTL Generation → PE/CGRA Synthesis

PE/CGRA Power, Performance, Area Estimation
Automatic PE Mapping

1. Rewrite Rule Generation:
   ● Generating a set of rewrite rules from a PEak program

2. Instruction Selection:
   ● Transforming a graph of CoreIR operations to a graph of PEs using the rewrite rules
Rewrite Rule Synthesis

Rewrite Rule Generator

C = A + B

In0 ↔ A
In1 ↔ B
ALU ↔ +
Out ↔ C

Formal Model of PE
Rewrite Rule
Formal Model of Operation
Instruction Selection
Evaluation - Baseline PE

- One ALU
- One multiplier
- Two registers for integer operands
- Bit registers and LUT for bitwise operations
Camera Pipeline Results

<table>
<thead>
<tr>
<th>PE Variation</th>
<th>Num PEs</th>
<th>PE Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>348</td>
<td>750.4</td>
</tr>
<tr>
<td>1</td>
<td>318</td>
<td>585.1</td>
</tr>
<tr>
<td>2</td>
<td>283</td>
<td>585.9</td>
</tr>
<tr>
<td>3</td>
<td>187</td>
<td>620.5</td>
</tr>
<tr>
<td>4</td>
<td>148</td>
<td>758.6</td>
</tr>
<tr>
<td>5</td>
<td>140</td>
<td>727.3</td>
</tr>
</tbody>
</table>
Camera Pipeline Results

- Energy/Op (pJ) vs. Frequency (GHz): 8.3x lower
- Area (μm²) vs. Frequency (GHz): 3.4x lower
Image Processing Results

Camera Pipeline

Harris

Laplacian Pyramid

Gaussian
Image Processing Results

Gaussian

Laplacian

Harris

Camera

Area (normalized)
Image Processing Results

- **Gaussian**
  - Base: [Bar Graph]
  - IP: [Bar Graph]
  - Spec: [Bar Graph]

- **Laplacian**
  - Base: [Bar Graph]
  - IP: [Bar Graph]
  - Spec: [Bar Graph]

- **Harris**
  - Base: [Bar Graph]
  - IP: [Bar Graph]
  - Spec: [Bar Graph]

- **Camera**
  - Base: [Bar Graph]
  - IP: [Bar Graph]
  - Spec: [Bar Graph]
Image Processing Results

Gaussian

Laplacian

Harris

Camera

Energy (normalized)

Base | IP | Spec
---|---|---
Base | IP | Spec
Base | IP | Spec
Base | IP | Spec
Base | IP | Spec
Image Processing Results

- **Gaussian**
- **Laplacian**
- **Harris**
- **Camera**

**Energy (normalized)**

- **Base**
- **Ip**
- **Spec**
Resnet Convolutional Layer Results

Area (normalized)

- Base
- ML0
- ML1
- ML2
- ML3

Energy (normalized)

- Base
- ML0
- ML1
- ML2
- ML3

Legend:
- PE
- SB
- CB
- MEM
- Other
Conclusion

- Developed an automated framework for design space exploration of CGRA PEs
  - Used subgraph mining techniques to analyze applications
  - Used maximal independent set analysis to pick interesting subgraphs
  - Merged interesting subgraphs together to form a PE
  - Automatically generated a compiler for the customized CGRA
  - Demonstrated energy and area benefits of specialization