PE Design Space Exploration

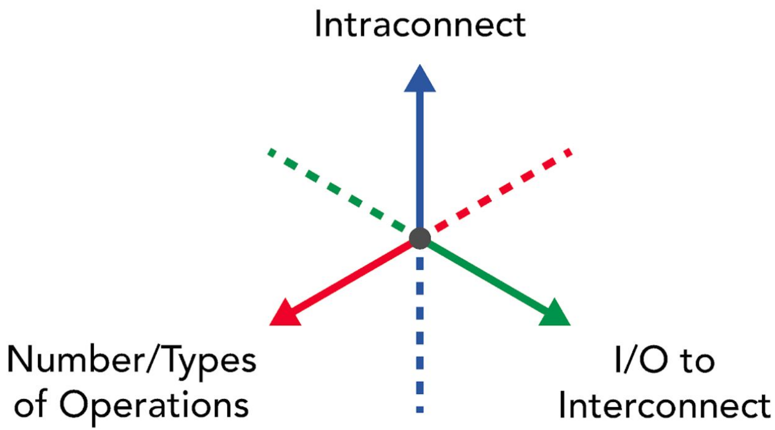
Jackson Melchert, Kathleen Feng, Caleb Donovick, Ross Daly

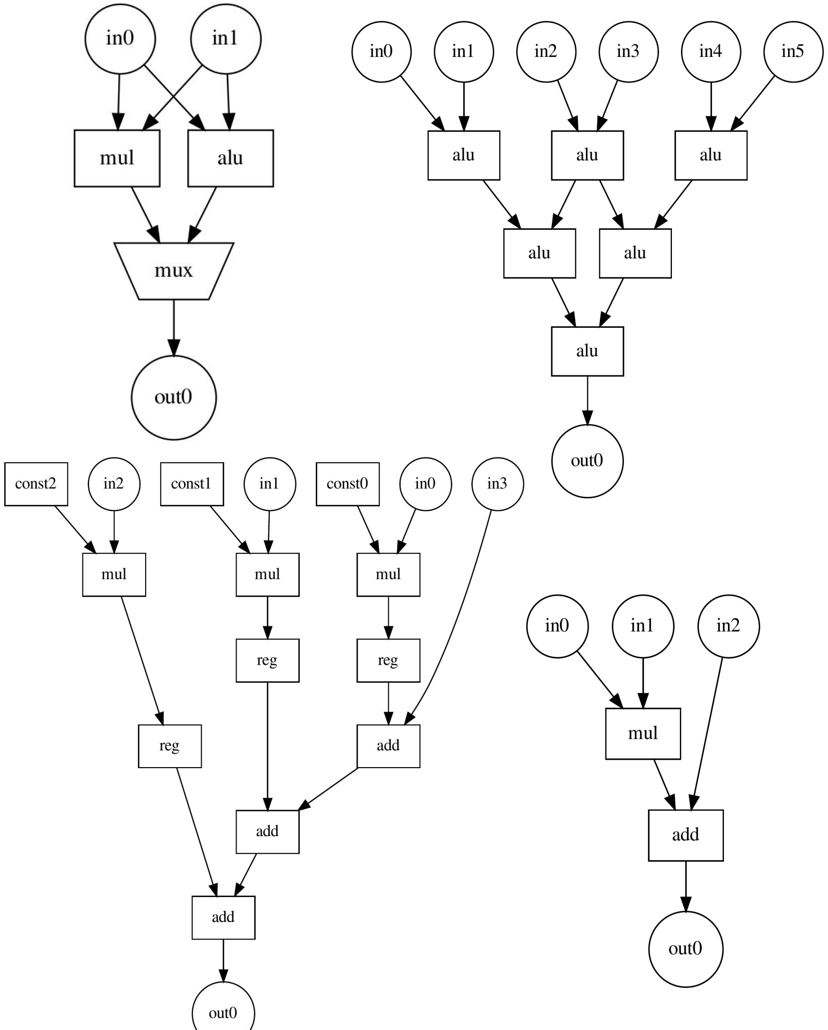
Motivation

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

1. Analyze application domain benchmarks to find possible optimizations
2. Explore PE design space utilizing PEak DSL
3. Automatically generate full compiler to run applications
4. Find optimal architecture given evaluation results

Design Space Axes



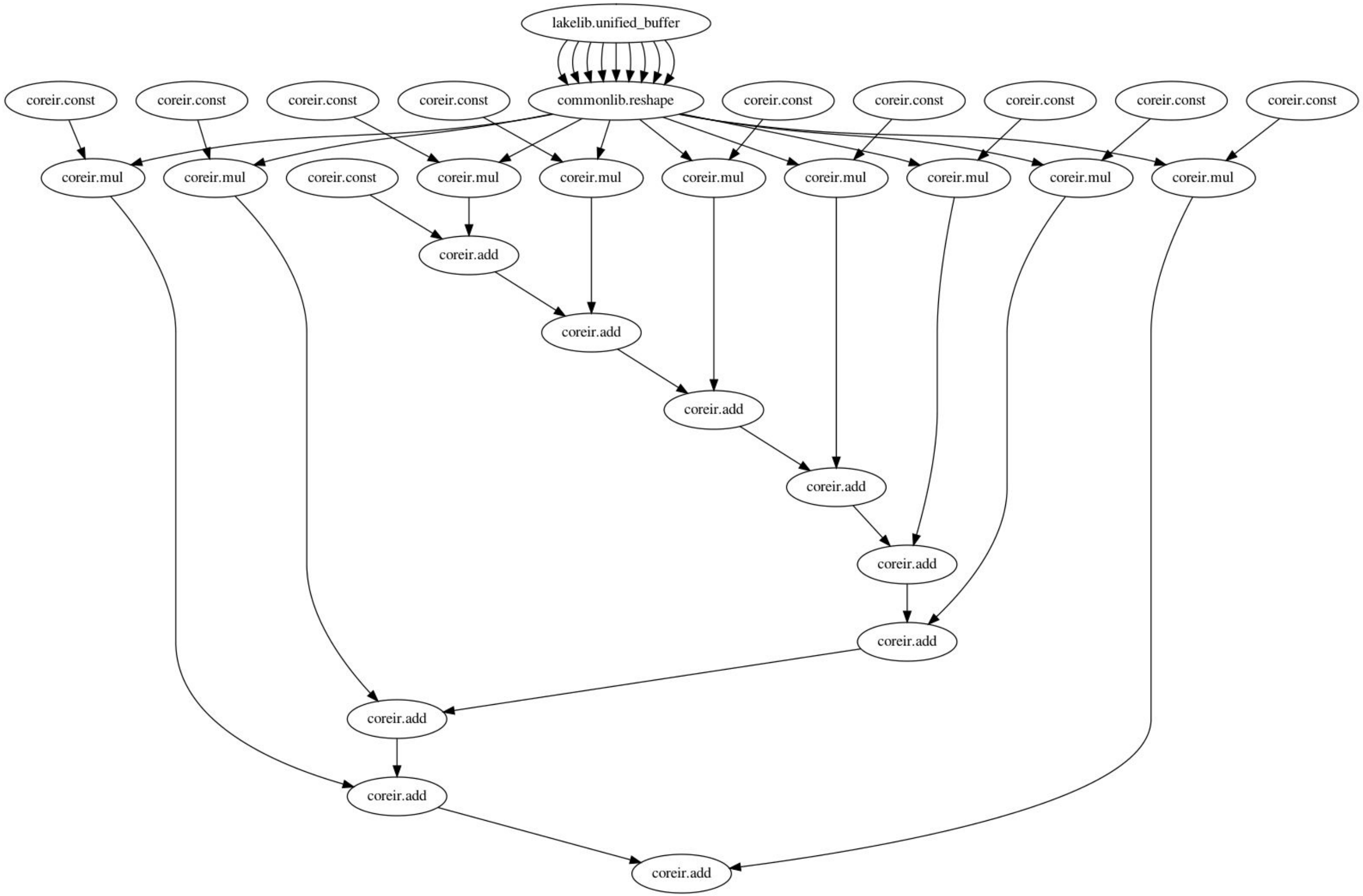


Application Analysis

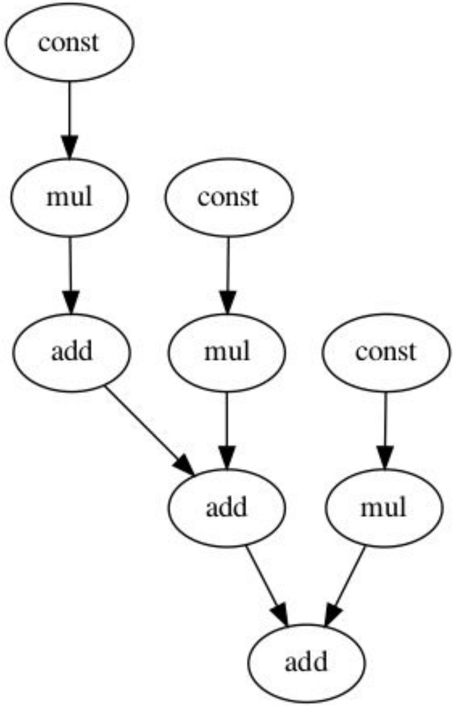
- How do we identify candidate PEs that explore the design space?

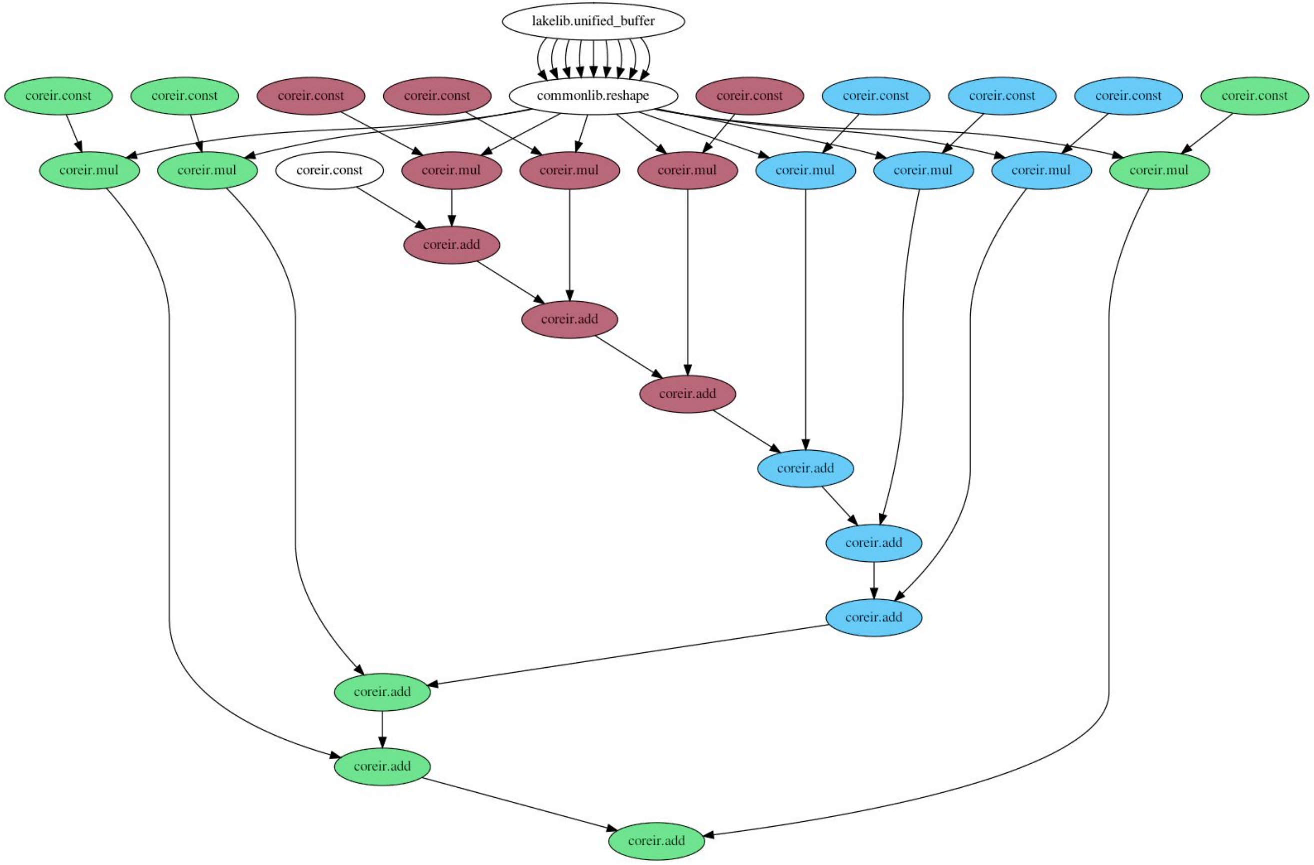
- Frequent subgraph analysis:
  1. Given a Halide application, generate the CoreIR graph
  2. Enumerate frequent subgraphs within the CoreIR graph

Frequent Subgraphs of Conv 3x3

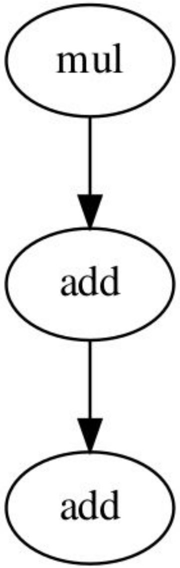


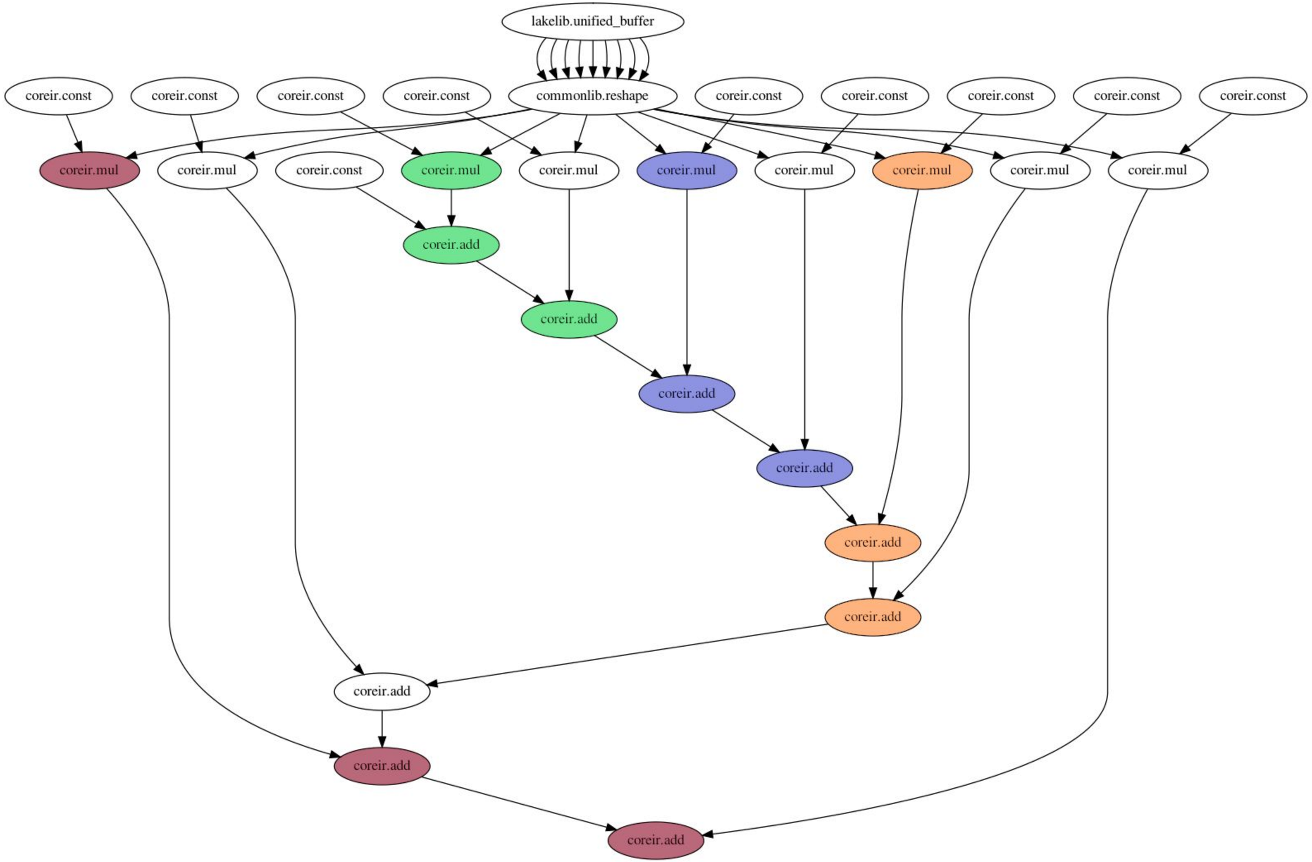
Frequent Subgraphs of Conv 3x3





Frequent Subgraphs of Conv 3x3



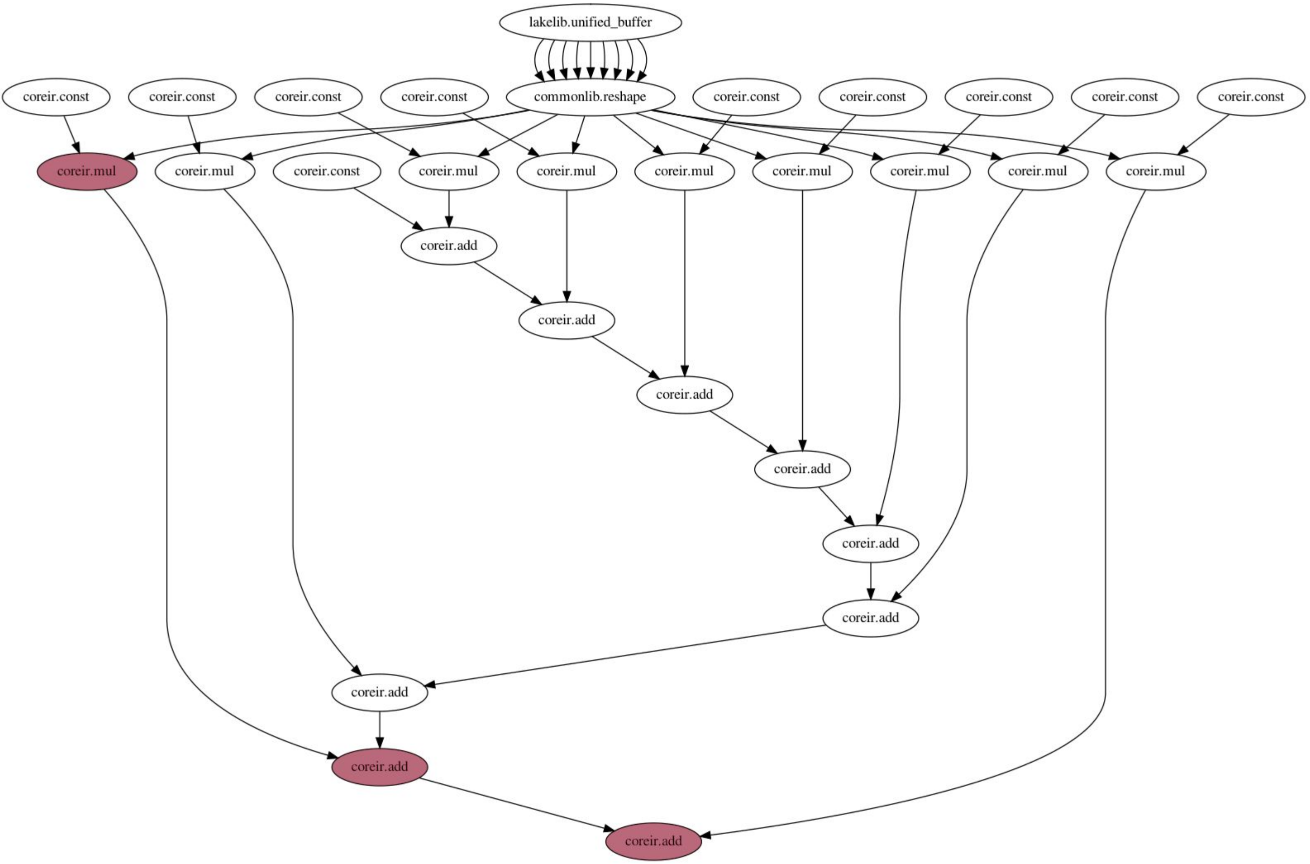


Maximal Independent Set Analysis

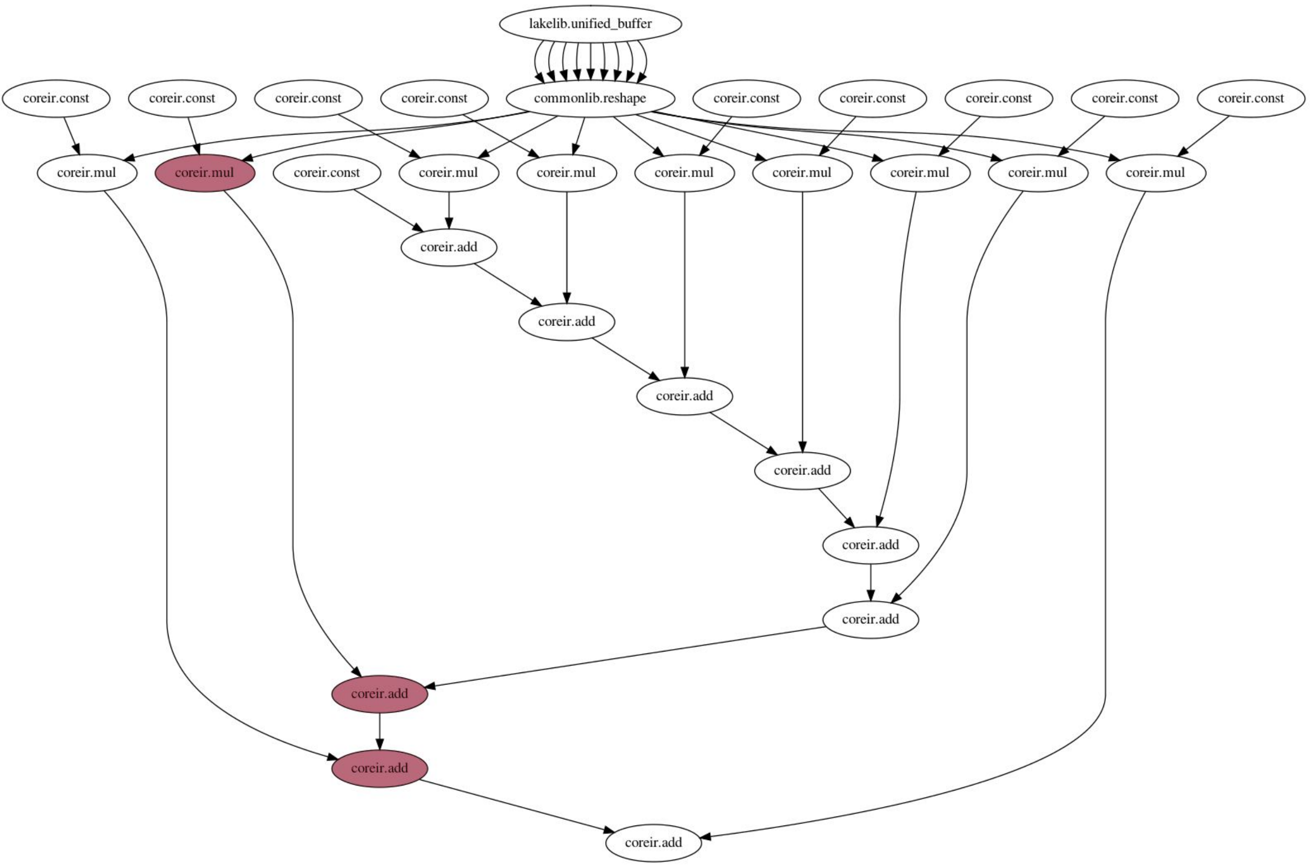
For each subgraph:

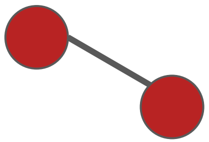
1. Represent each occurence of that subgraph as a node in a new graph
2. Add an edge between nodes if the subgraph occurrences overlap
3. Calculate the maximal independent set

Maximal Independent Set Analysis Example

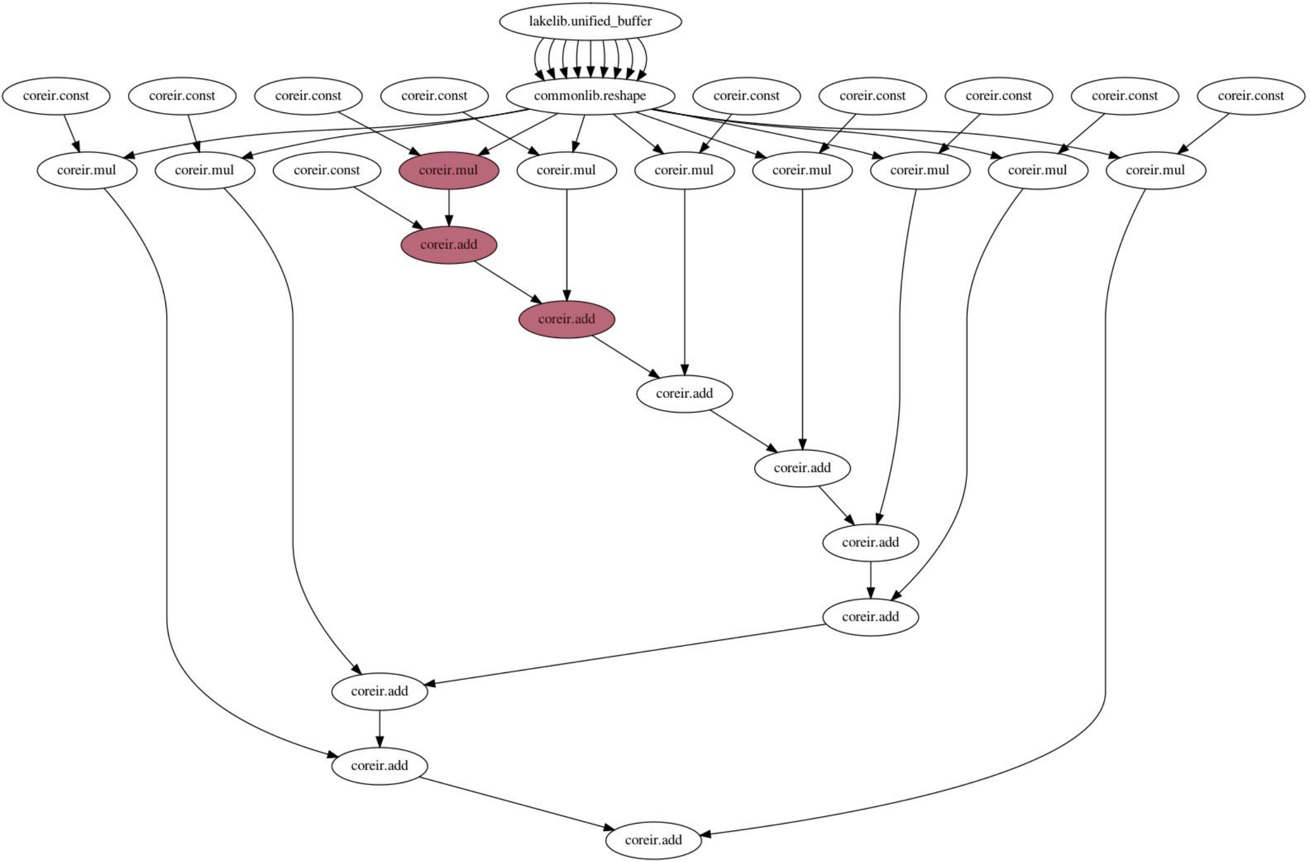


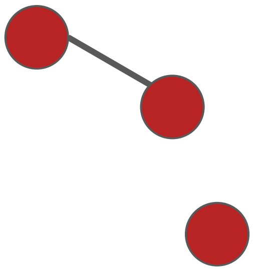
Maximal Independent Set Analysis Example



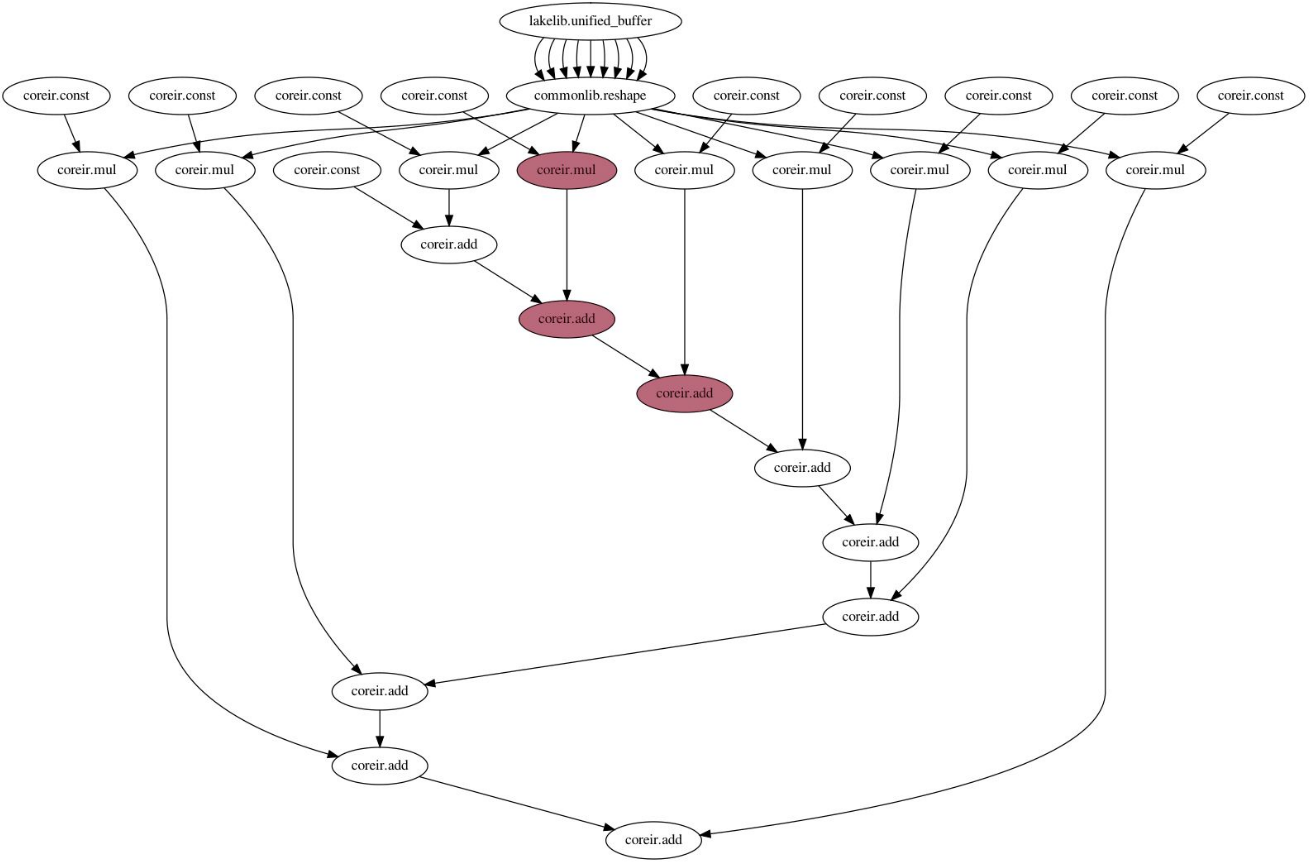


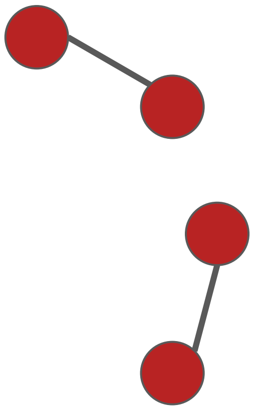
Maximal Independent Set Analysis Example



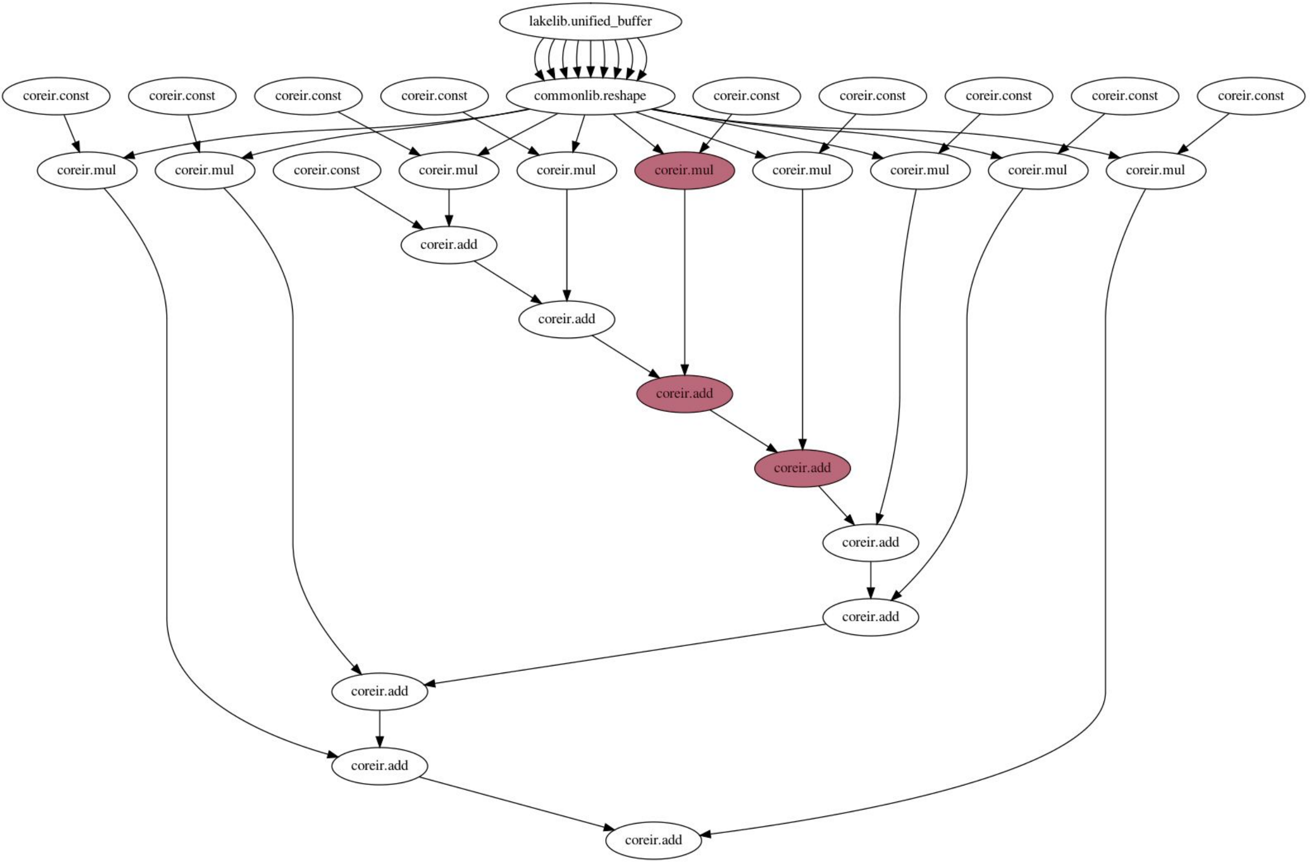


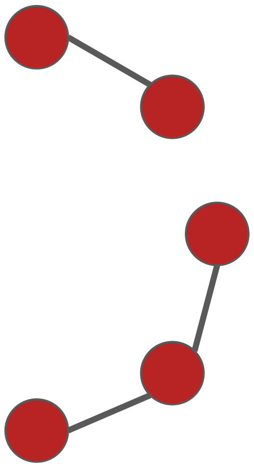
Maximal Independent Set Analysis Example



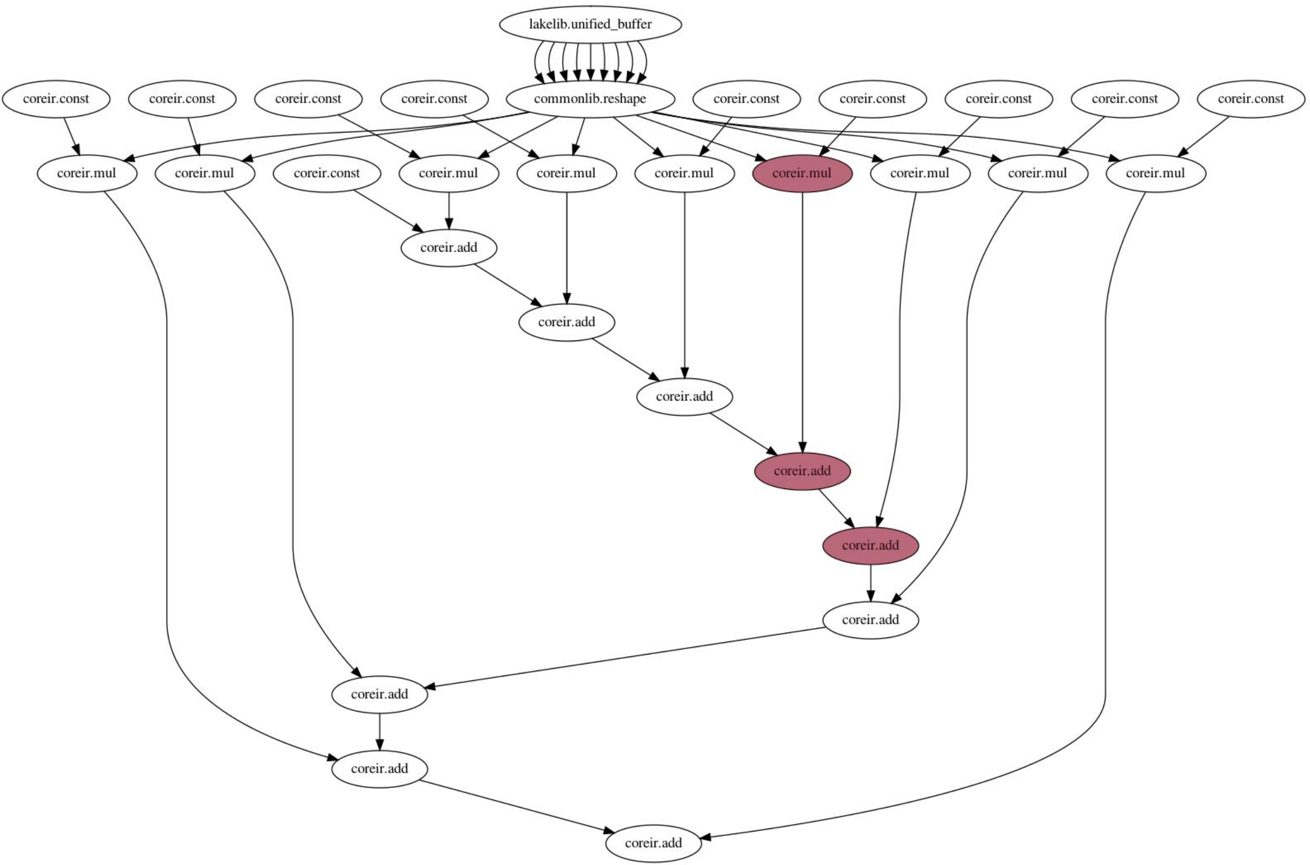


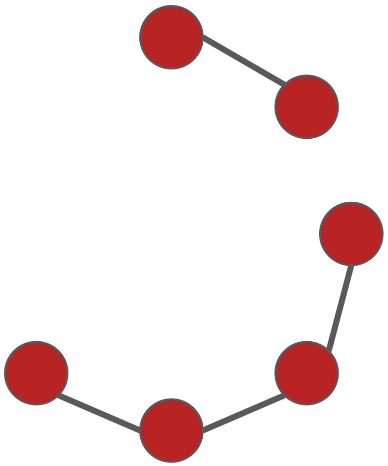
Maximal Independent Set Analysis Example



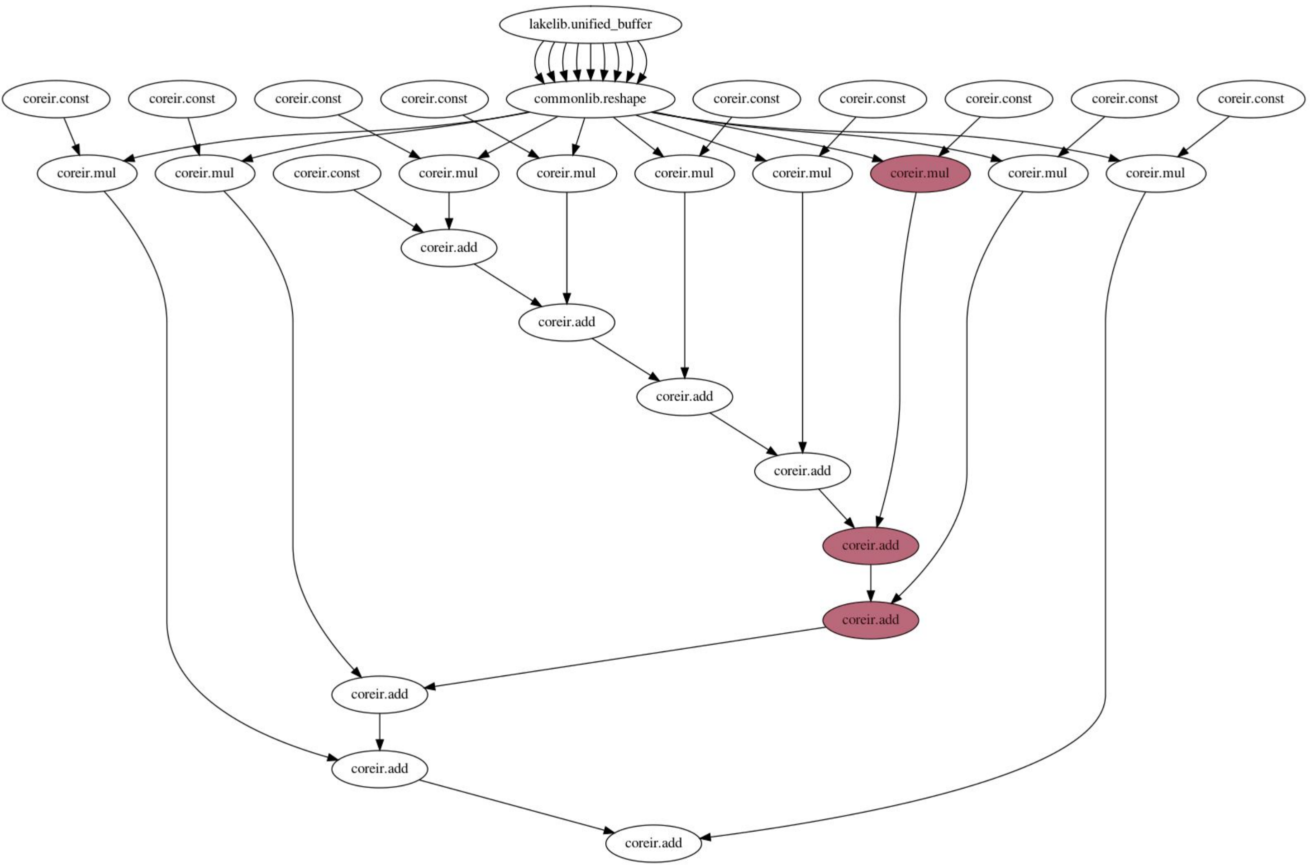


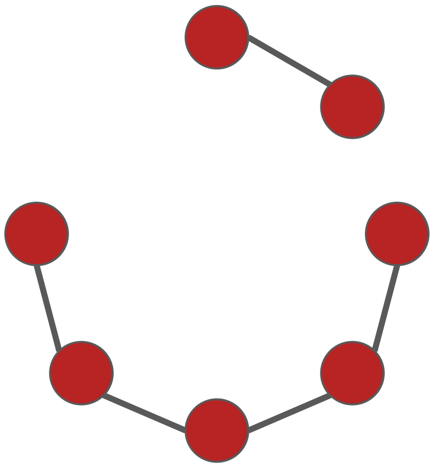
Maximal Independent Set Analysis Example



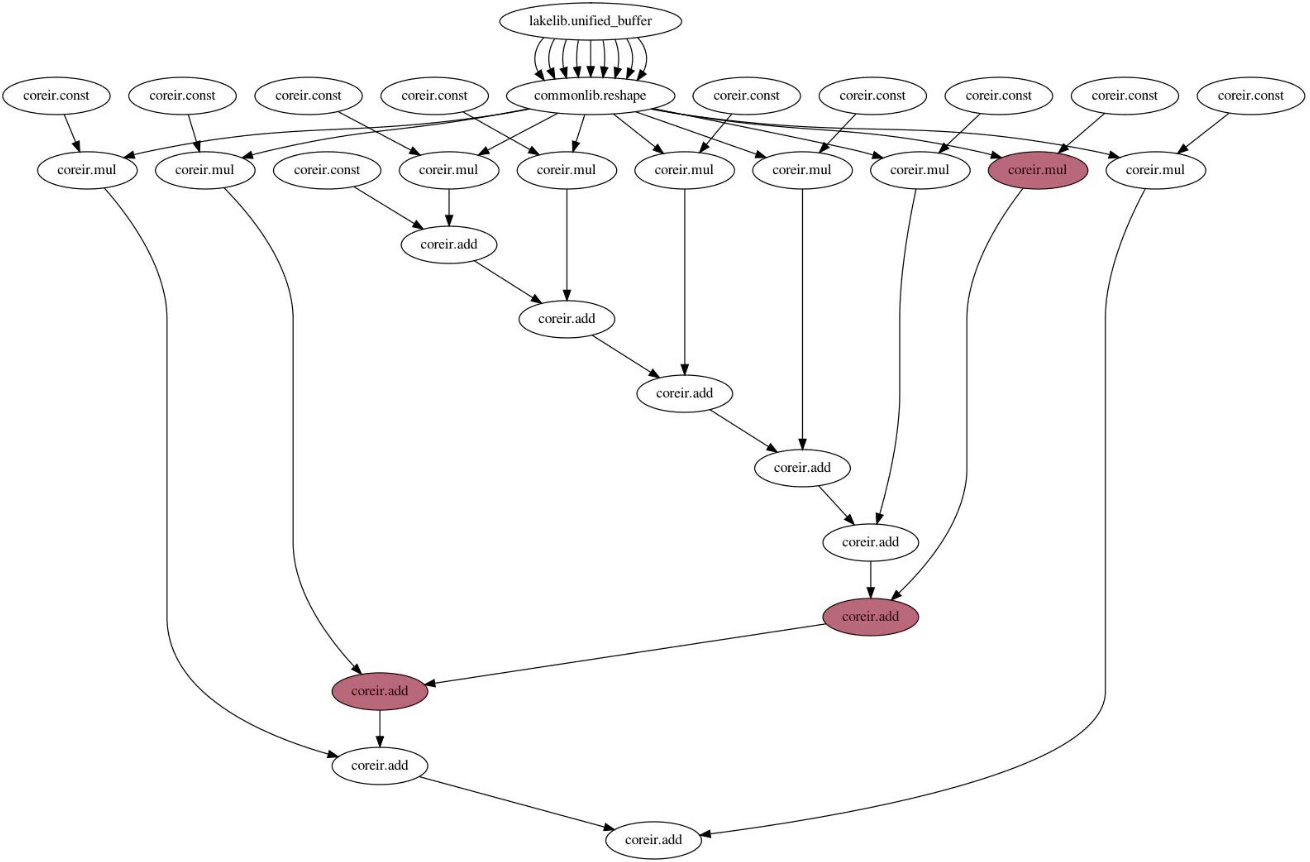


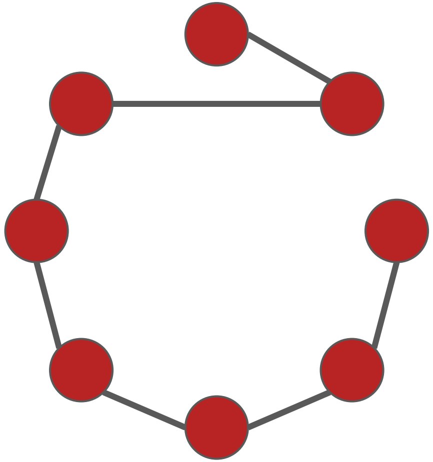
Maximal Independent Set Analysis Example



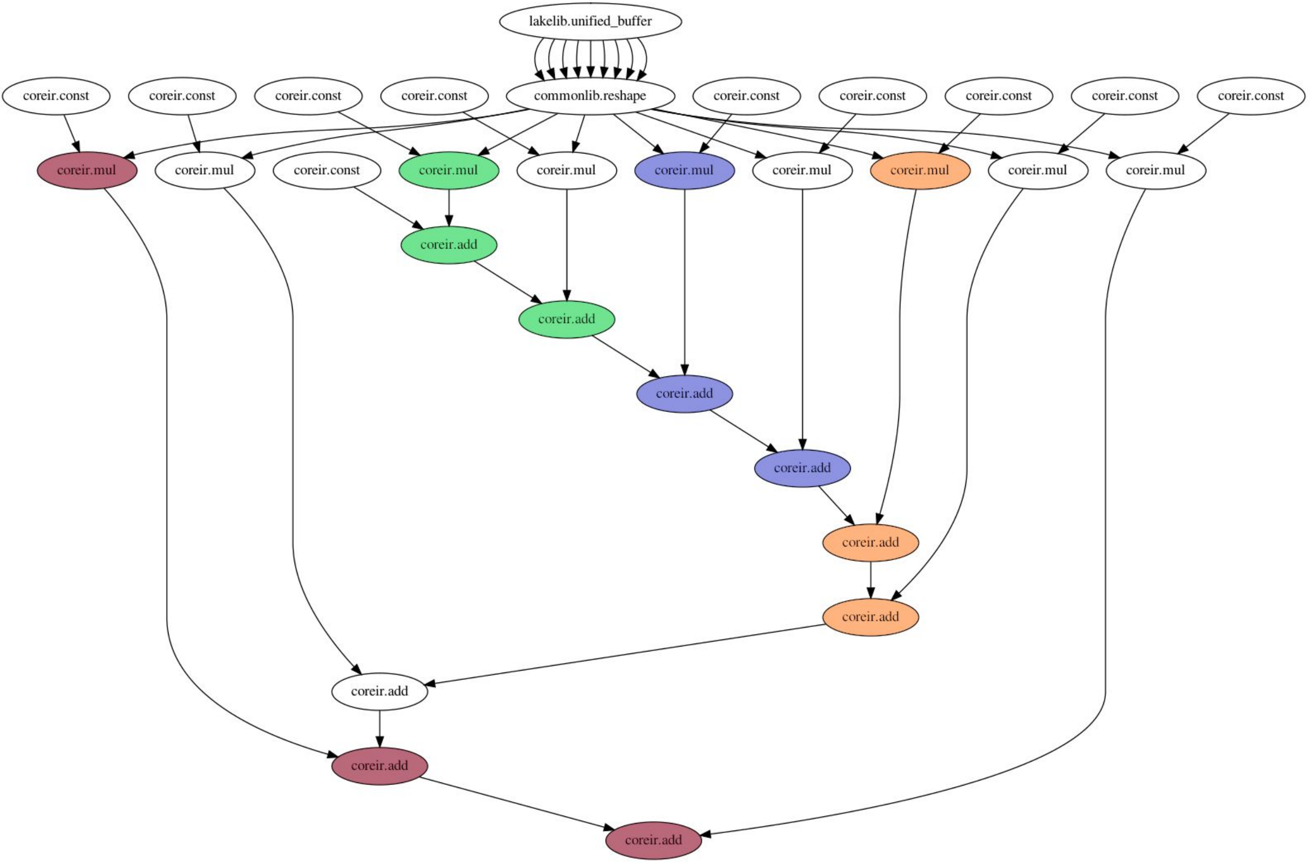


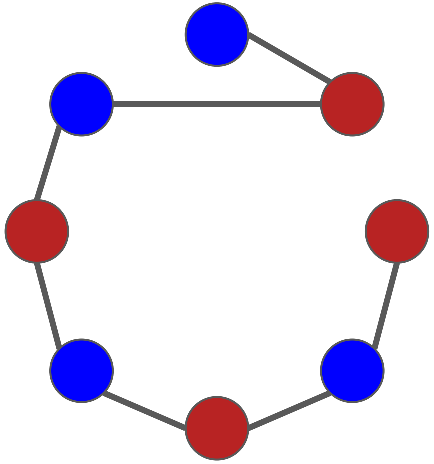
Maximal Independent Set Analysis Example





Maximal Independent Set Analysis Example





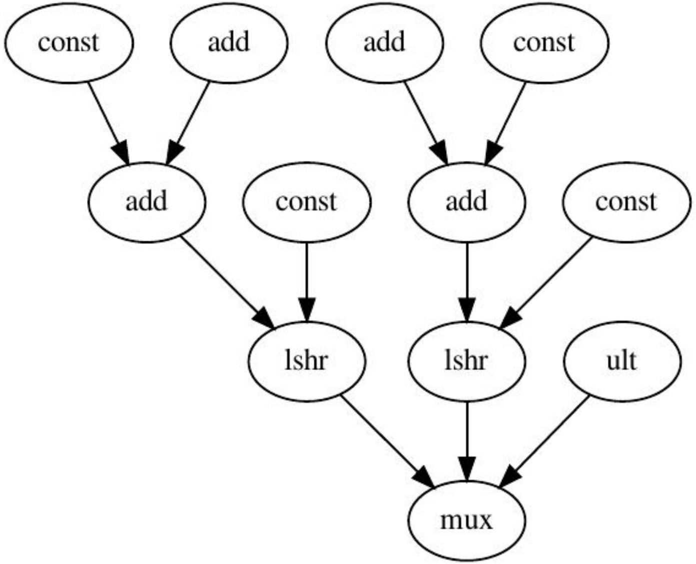
What next?

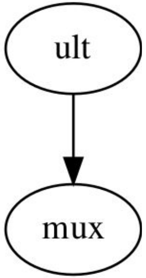
- Do transformation passes on subgraphs to explore design space
  - Add muxes
  - Remove inputs using constant registers
  - Route more operation outputs to PE outputs
  - Add operations that don’t exist in subgraph

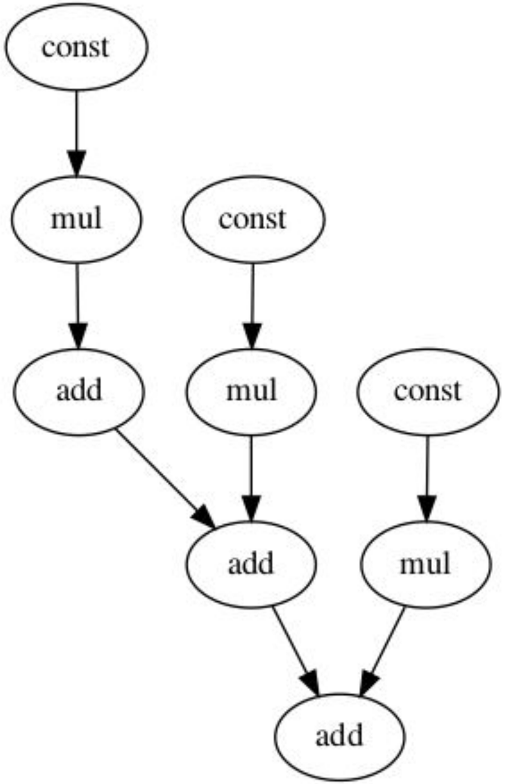
What next?

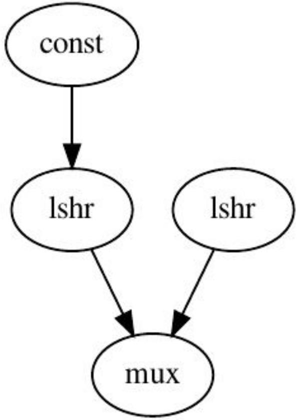
- Do transformation passes on subgraphs to explore design space
  - Add muxes
  - Remove inputs using constant registers
  - Route more operation outputs to PE outputs
  - Add operations that don’t exist in subgraph
- Merge many interesting subgraphs
  - Enables better coverage of application graphs
  - Intelligently explores the connectivity design space axis
  - Allows for more effectively analyzing multiple applications

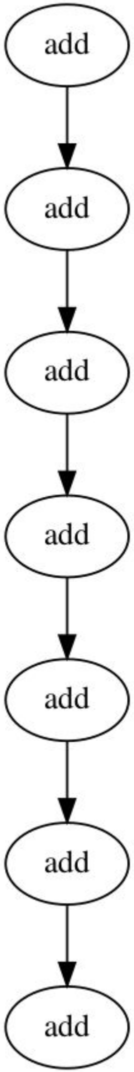
What do frequent subgraphs look like?







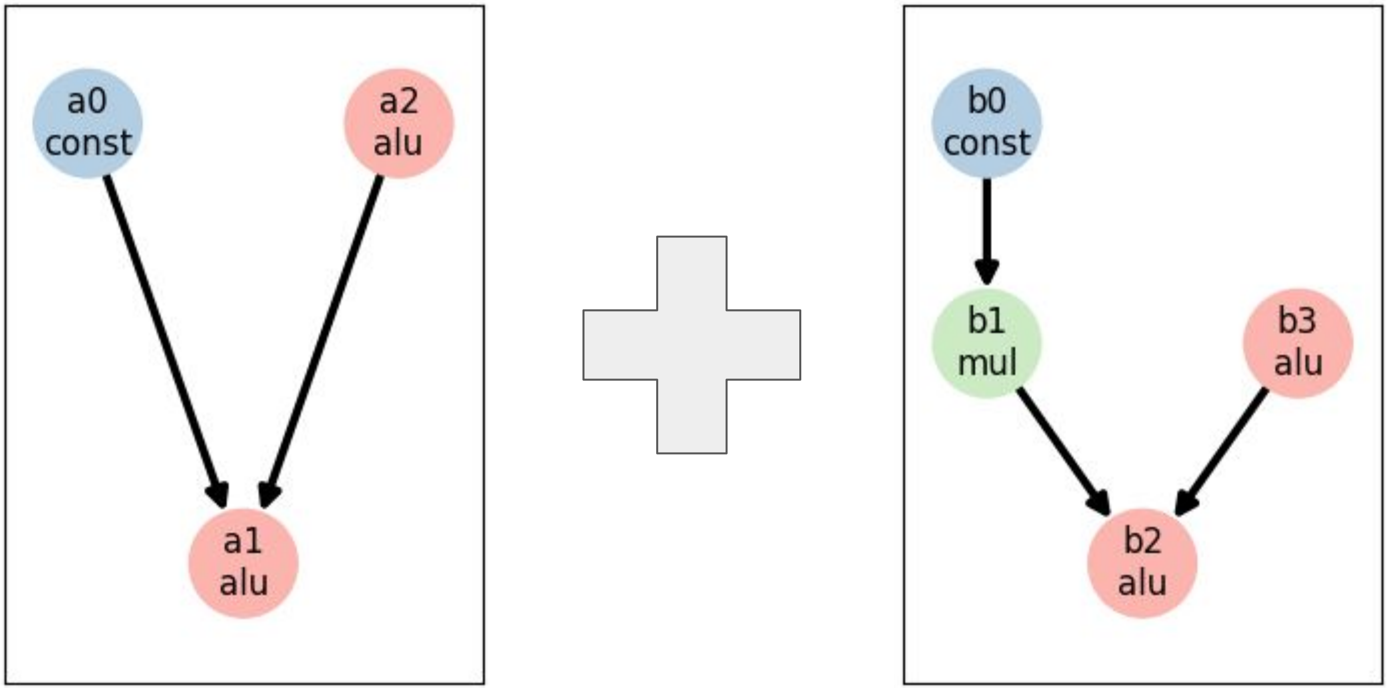




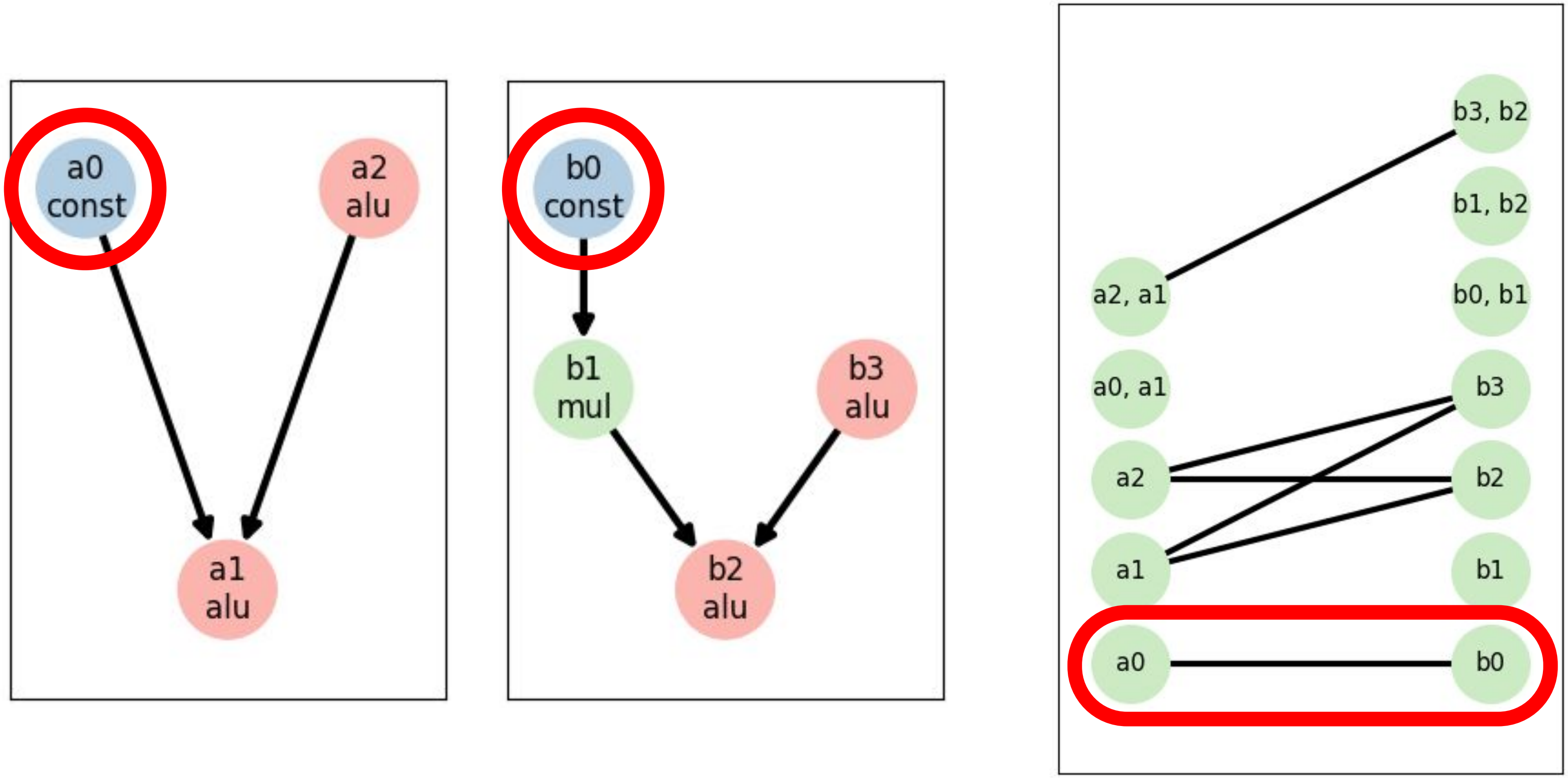
Merging Subgraphs

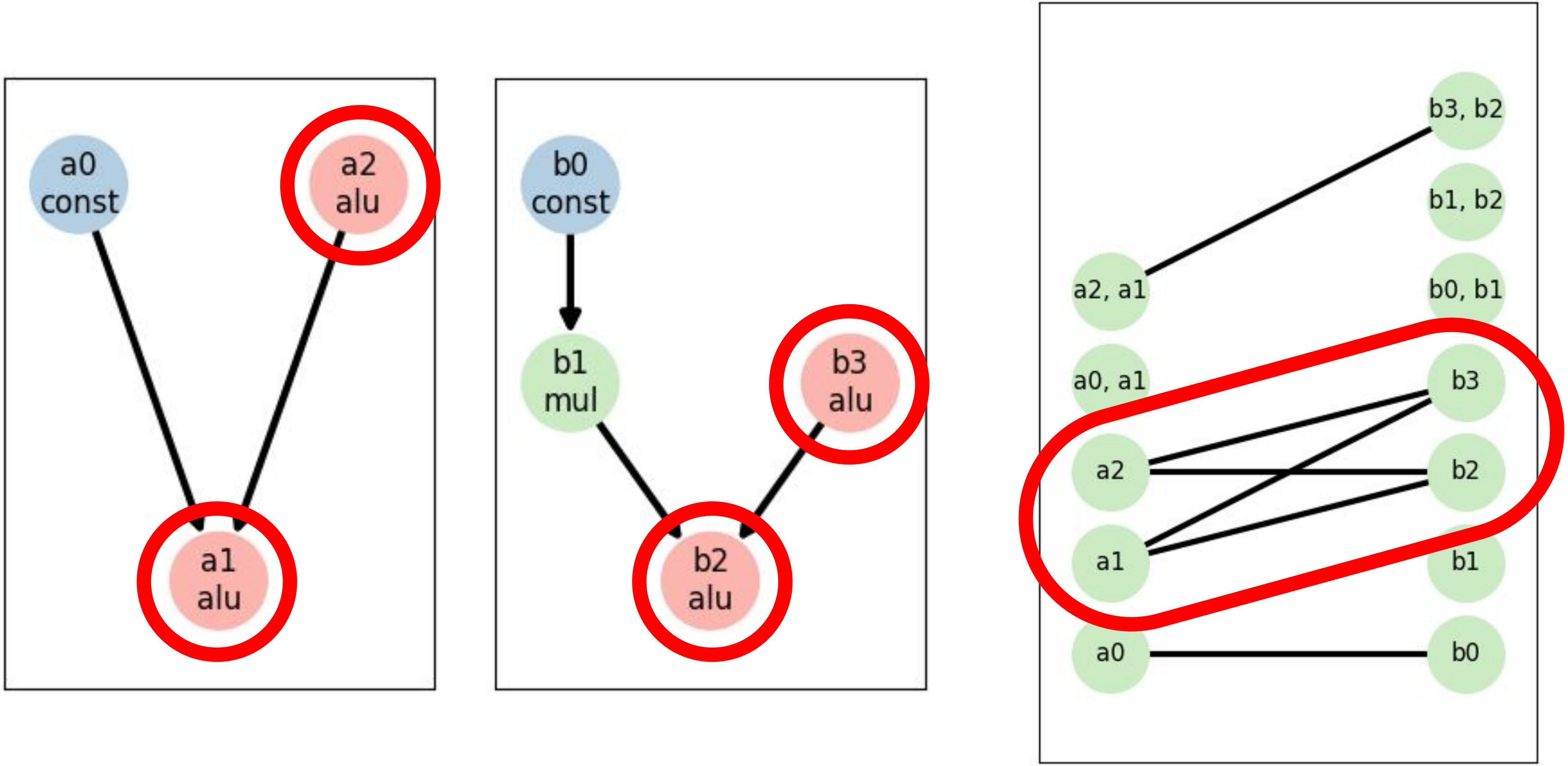
Dataflow graph merging:

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

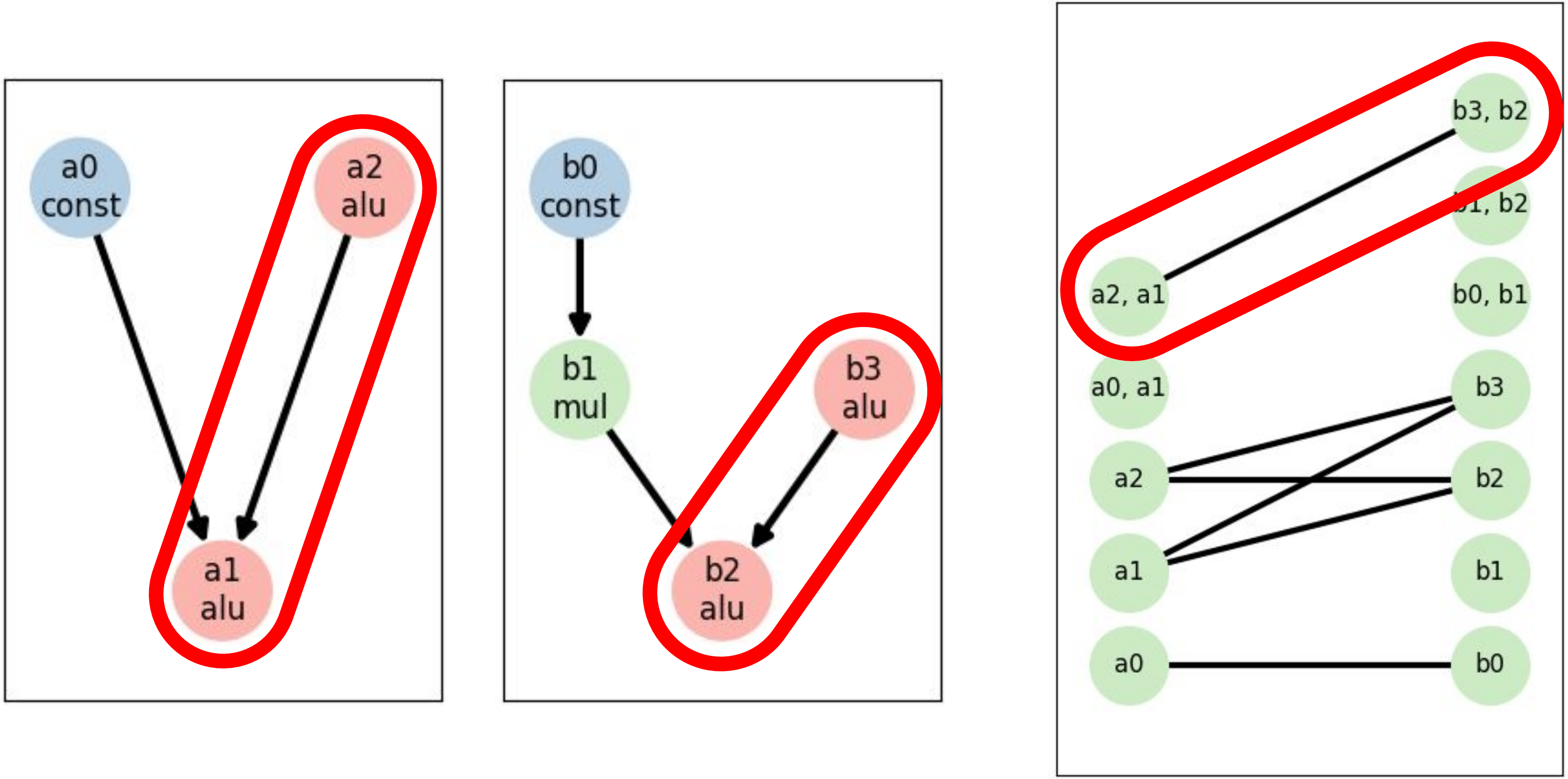


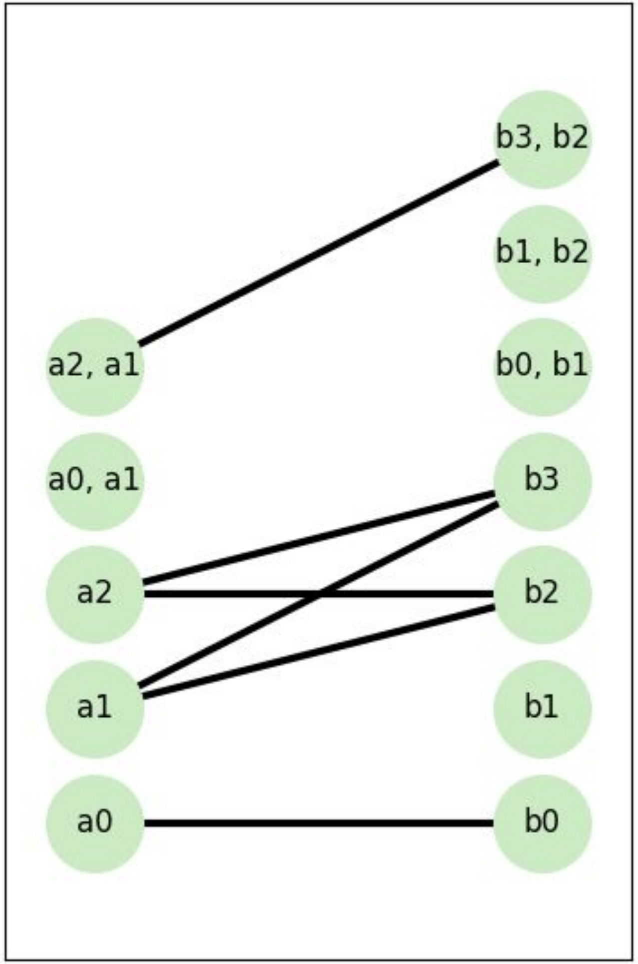
a0
const
a2
alu
a1
alu
b0
const
b1
mul
b3
alu
b2
alu

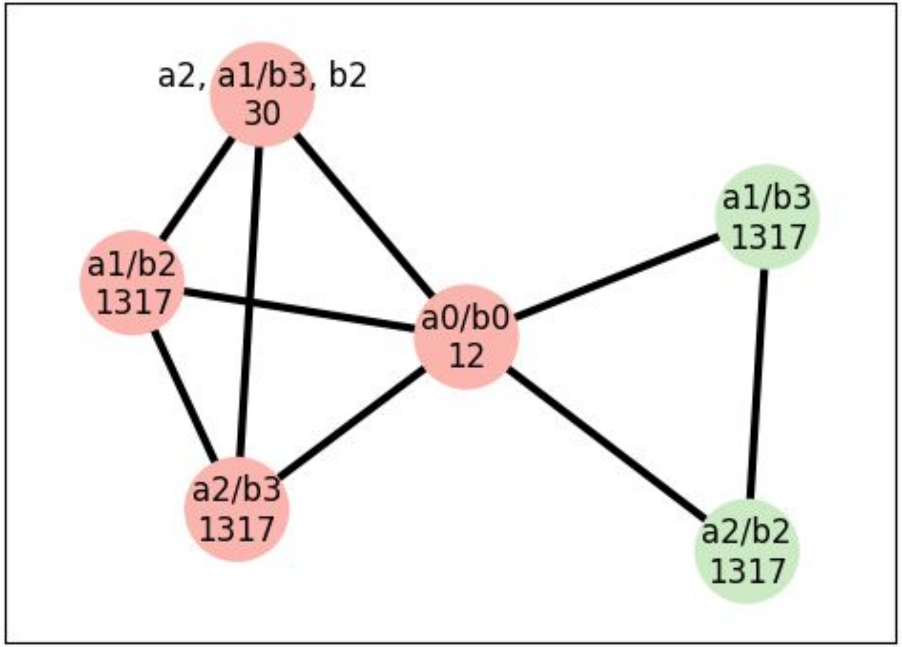

a0
const
a2
alu
a1
alu
b0
const
b1
mul
b3
alu
b2
alu
b3, b2
b1, b2
a2, a1
b0, b1
a0, a1
b3
a2
b2
a1
b1
a0
b0

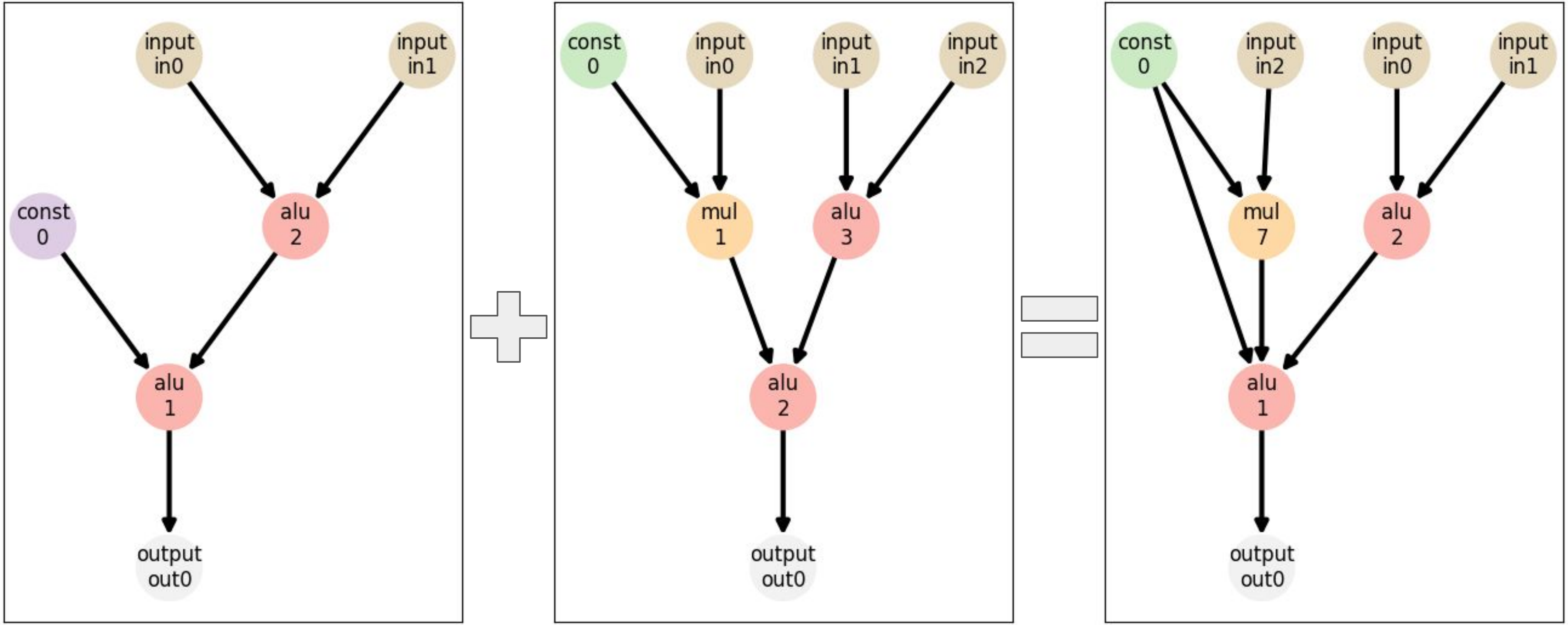


a0
const
a2
alu
a1
alu
b0
const
b1
mul
b3
alu
b2
alu
b3, b2
b1, b2
a2, a1
b0, b1
a0, a1
b3
a2
b2
a1
b1
a0
b0


a0
const
a2
alu
a1
alu
b0
const
b1
mul
b3
alu
b2
alu
b3, b2
b1, b2
a2, a1
b0, b1
a0, a1
b3
a2
b2
a1
b1
a0
b0


b3, b2
b1, b2
a2, a1
b0, b1
a0, a1
b3
a2
b2
a1
b1
a0
b0


a2, a1/b3, b2
30
a1/b3
1317
a1/b2
1317
a0/b0
12
a2/b3
1317
a2/b2
1317



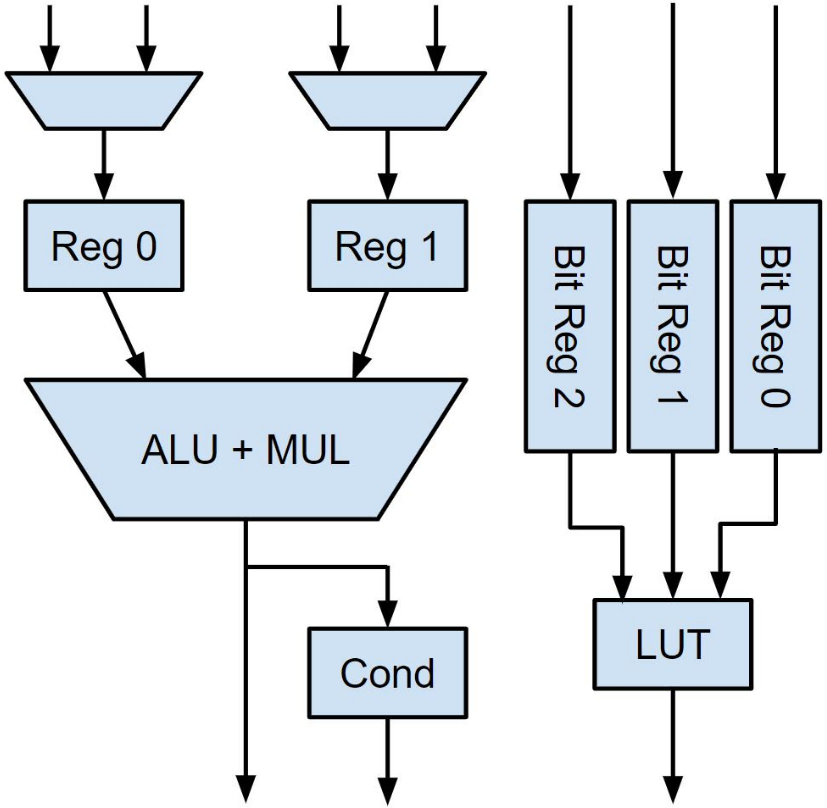
input in0
input in1
const 0
alu 2
alu 1
output out0
const 0
input in0
input in1
input in2
mul 1
alu 3
alu 2
output out0
const 0
input in2
input in0
input in1
mul 7
alu 2
alu 1
output out0

Subgraph Merging Benefits

- Allows for exploration of the design space by tuning how many subgraphs are merged
- Can efficiently map to multiple applications at once
- Scales well to large number of subgraphs
  - Much faster than maximal independent set analysis
- Can also output rewrite rules for mapping purposes

Experimental Results - Baseline

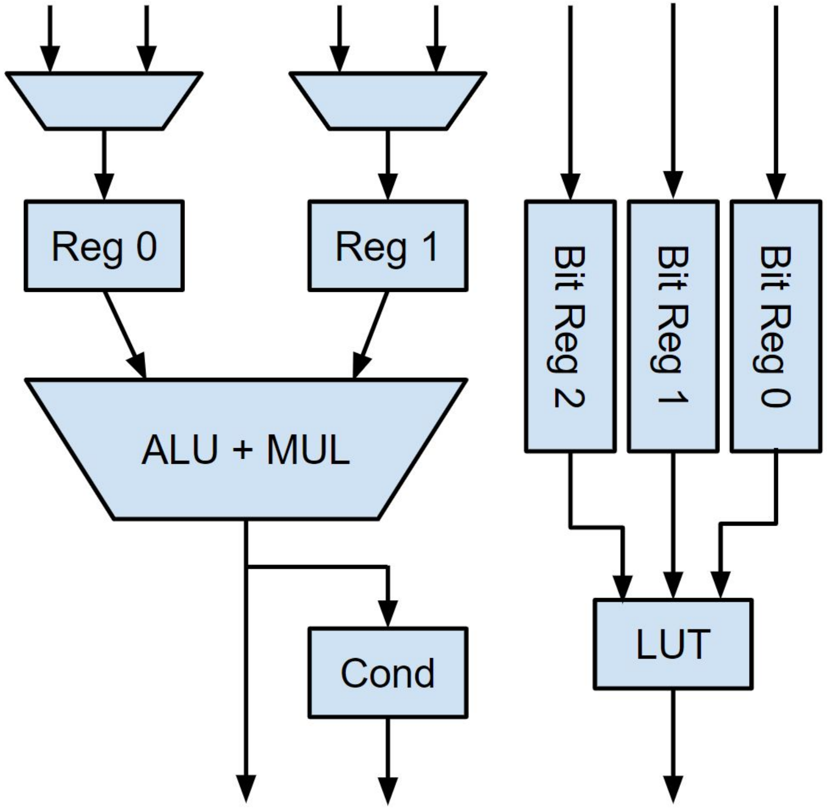
- Can do every primitive operation
- Only 1 operation per PE
- Has expensive multiplier regardless of application



Experimental Results - Baseline

- Energy per operation
  - 0.6ns clock period
  - Post-synthesis

Const	11.76 fJ
Add	1.44 pJ
Multiply	1.43 pJ

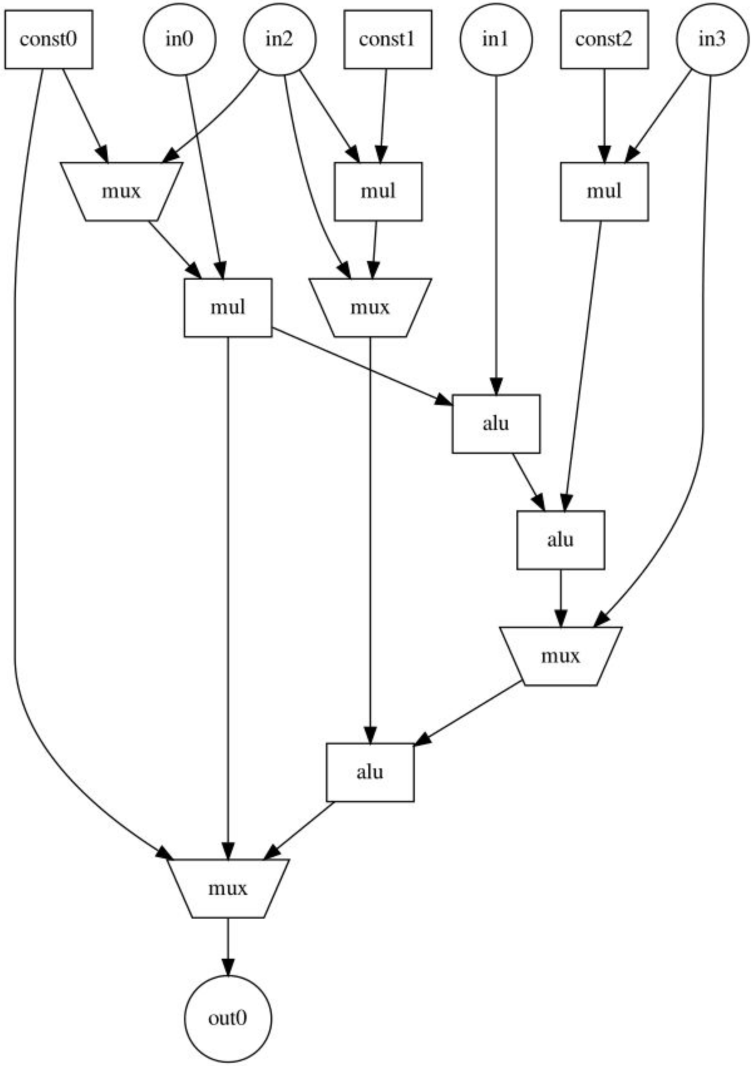


Experimental Results

- Specialized for 3x3 convolution:

in1 + (in0 * const0) + (in2 * const1) + (in3 * const2)

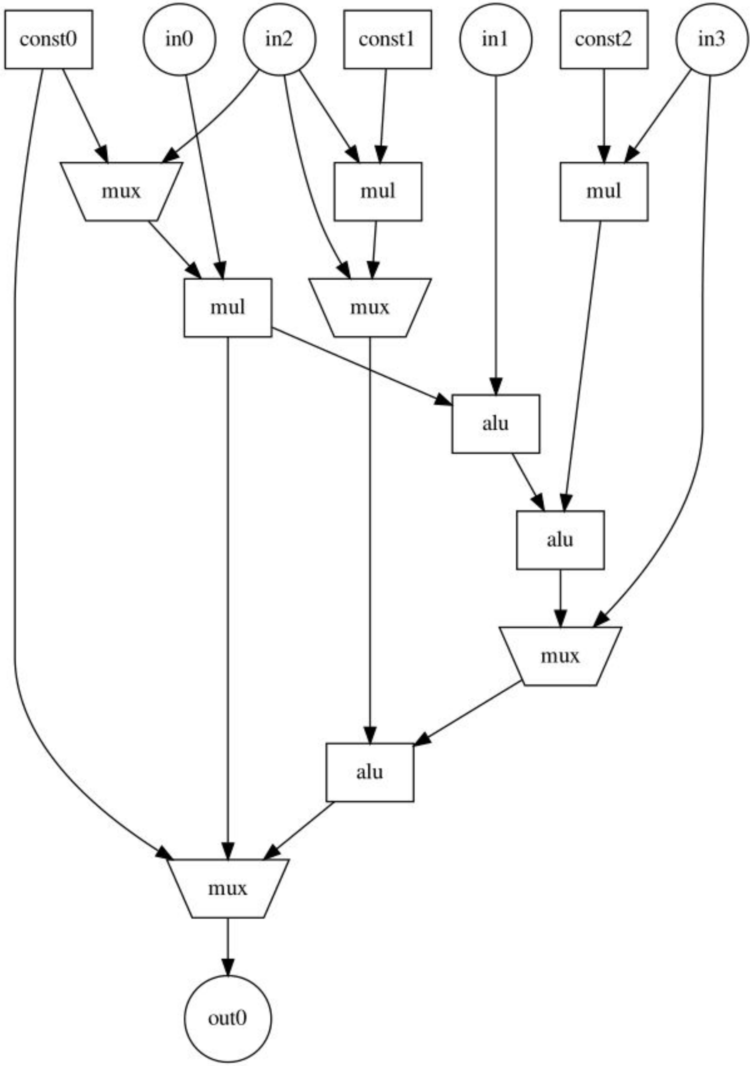
- Also implements every baseline PE op
- Can replace each ALU with just an adder for further specialization



Experimental Results

- Energy per operation
  - 0.9 ns max path delay
  - Post-synthesis

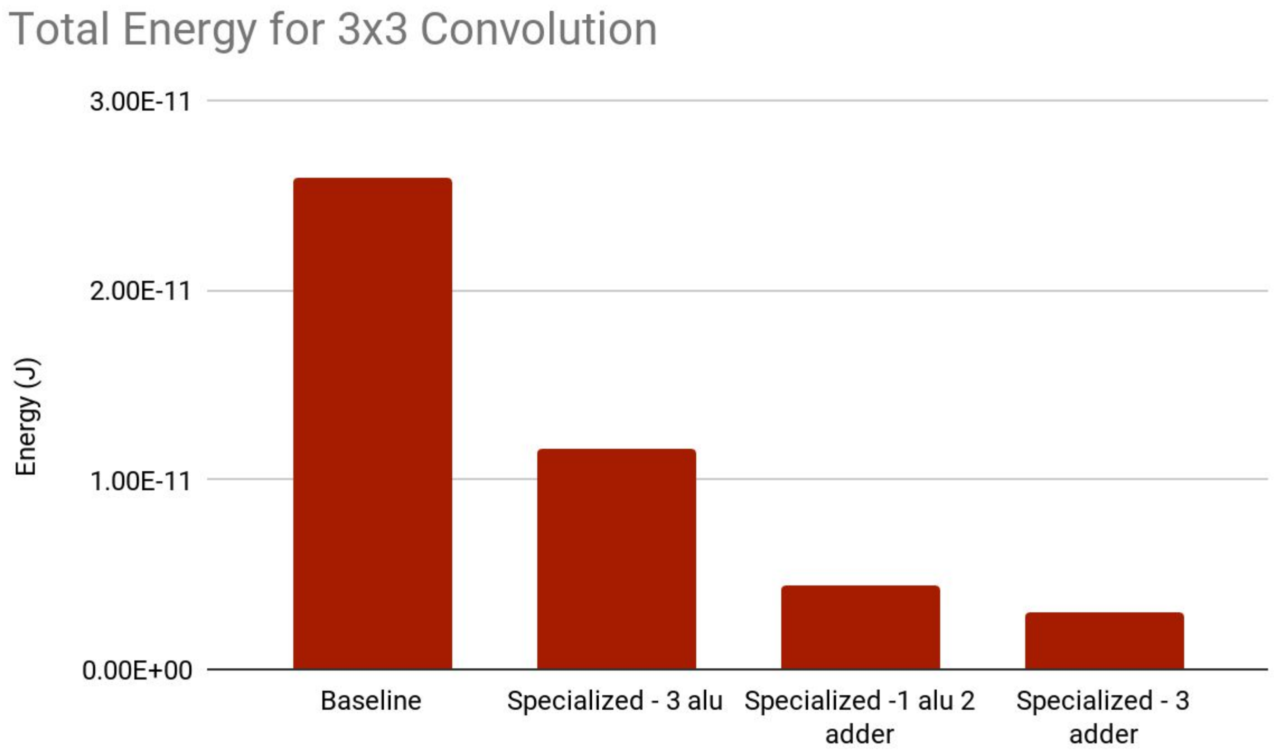
	3 ALU	1 ALU 2 adders	3 adders
Const	0.626 fJ	0.192 fJ	0.121 fJ
3 x Multiply-Add	3.88 pJ	1.48 pJ	1.01 pJ



Experimental Results

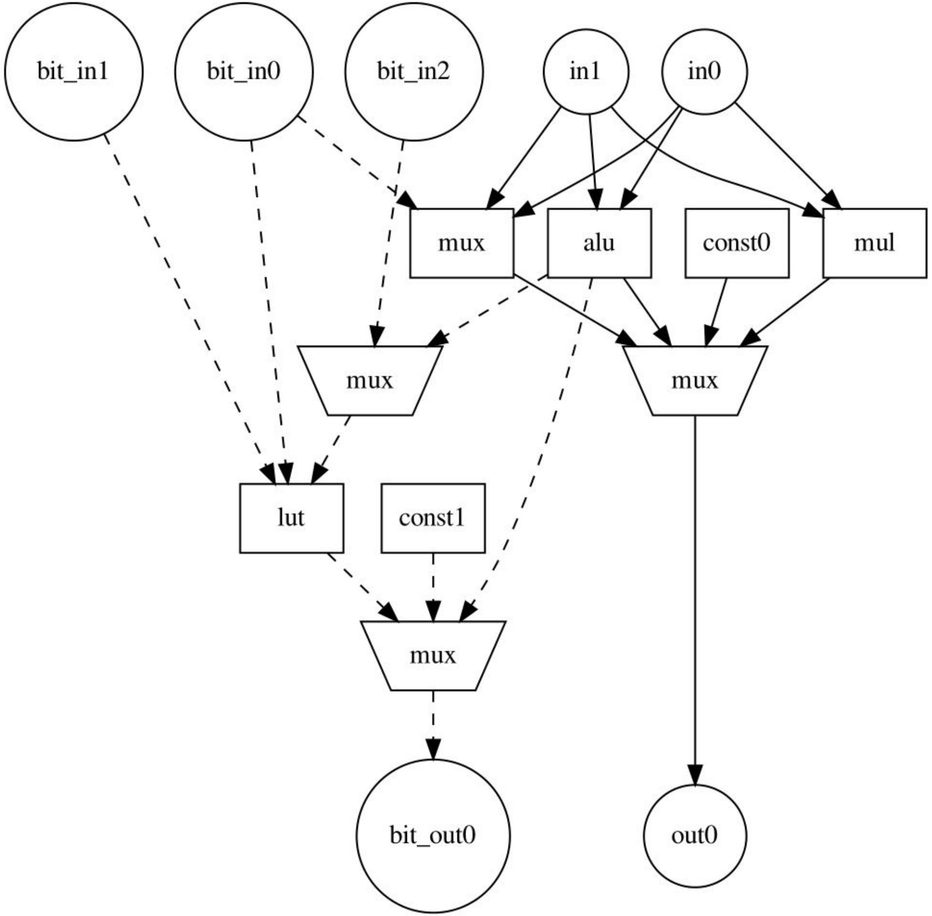
PE	Frequency	Area (μm^2)	# PEs	Net Area
Baseline	1.66 GHz	956.97	18	17225.46
Specialized - 3 alu	1.11 GHz	3479.96	4	13919.84
Specialized - 1 alu 2 adders	1.11 GHz	1553.48	4	6213.92
Specialized - 3 adder	1.11GHz	1220.26	4	4881.04

Experimental Results - Energy per Convolution



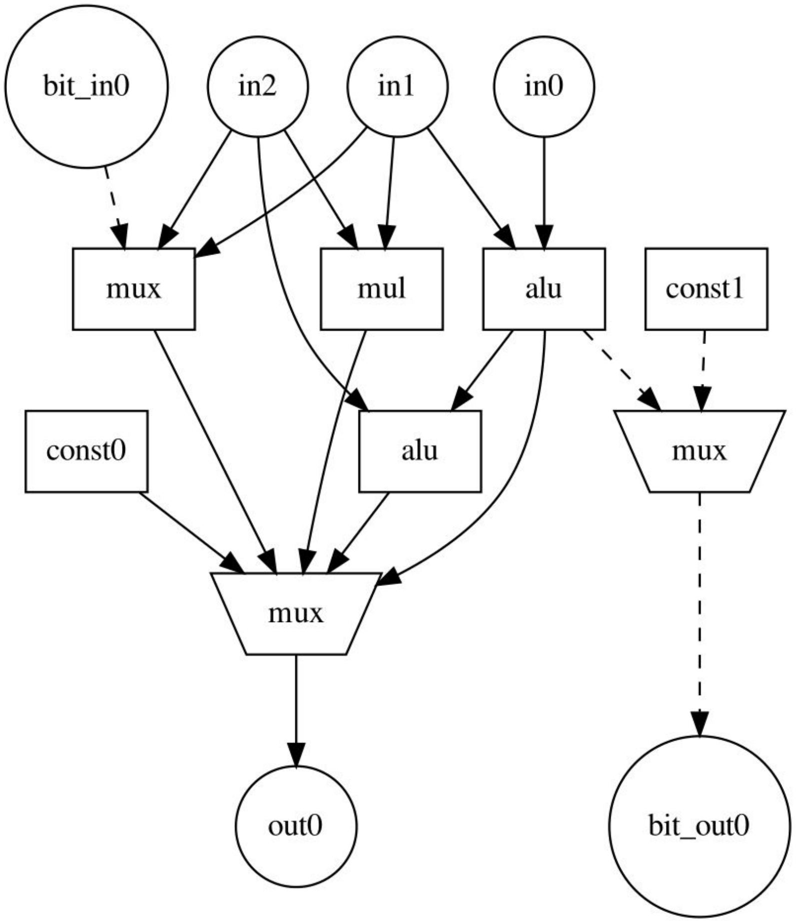
More Specialized PEs - Harris Corner Detection

- Bit operations implemented on 3 input LUT
- PE inputs can be selected using the dynamic mux mux



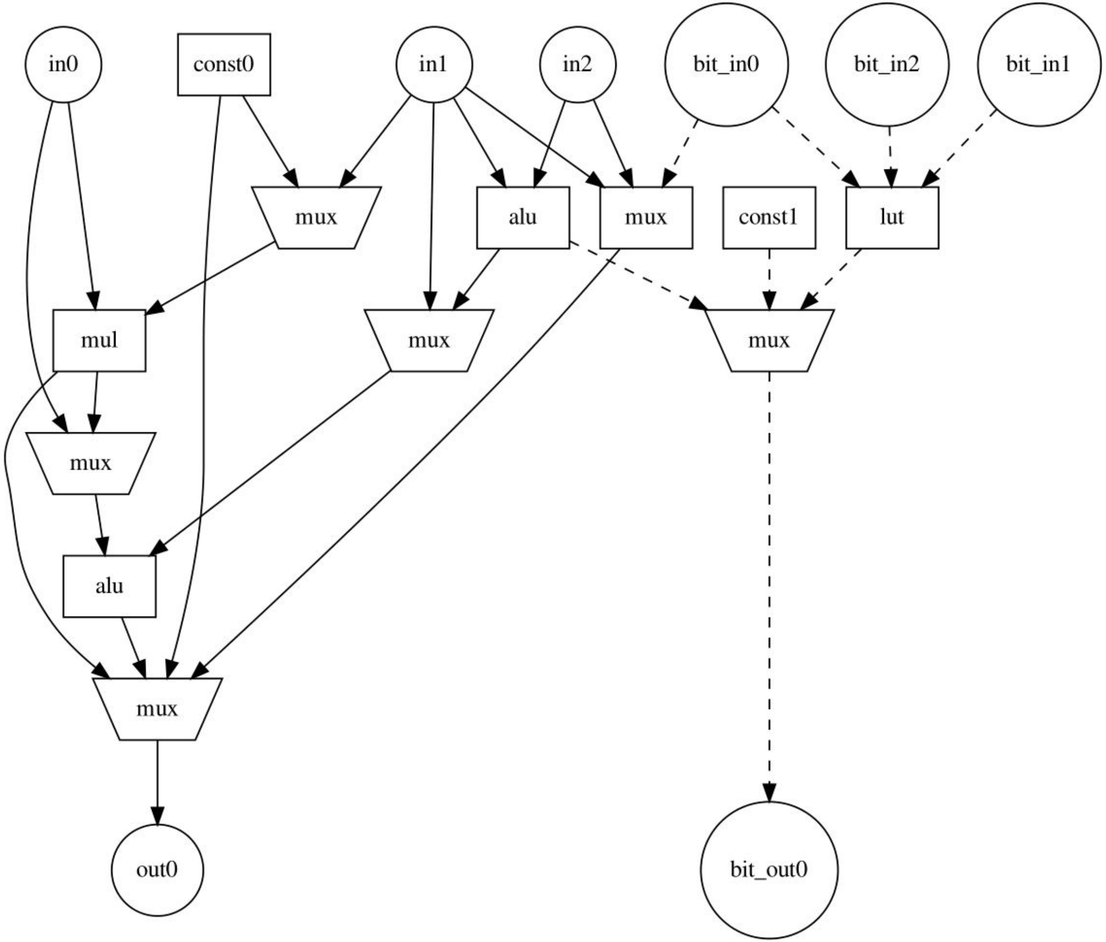
More Specialized PEs - Camera Pipeline

- Camera pipeline has 17 unique operations
- Also implements any 3 input chained ALU operation



More Specialized PEs

- Specialized for:
  - Camera Pipeline
  - Harris
  - 3x3 Convolution
- Implements multiply-ALU operations
- Also can do 3 input chained ALU operations



Summary

- Goal: Generate optimized PE architectures for an application domain
- Generated candidate PEs by analyzing applications using:
    - Frequent subgraph mining
    - Maximal independent set analysis
    - Subgraph merging
- Demonstrated energy, area, and performance benefits of specialized PEs vs baseline simple PE

Results (0.9ns), pJ/cycle, post DC

3mul2alu1add:
3mul4add:1.64e-03*0.9ns=1.476pJ
add:1.12e-03*0.9ns=1.008pJ
const:2.13e-07*0.9ns=0.1917fJ (this is all leakage energy, same for the other PEs below)
mul:1.83e-03*0.9ns=1.647pJ

3mul3alu:
3mul4add:4.31e-03*0.9ns=3.879pJ
add:2.26e-03*0.9ns=2.034pJ
const:6.95e-07*0.9ns=0.6255fJ
mul:4.55e-03*0.9ns=4.095pJ

3mul4add:
3mul4add:1.12e-03*0.9ns=1.008pJ
add:7.22e-04*0.9ns=0.6498pJ
const:1.34e-07*0.9ns=0.1206fJ
mul:1.29e-03*0.9ns=1.161pJ

lassen-no-fp:
add:2.40e-03*0.6ns=1.44pJ
const:1.96e-05*0.6ns=11.76fJ
mul:2.38e-03*0.6ns=1.43pJ