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

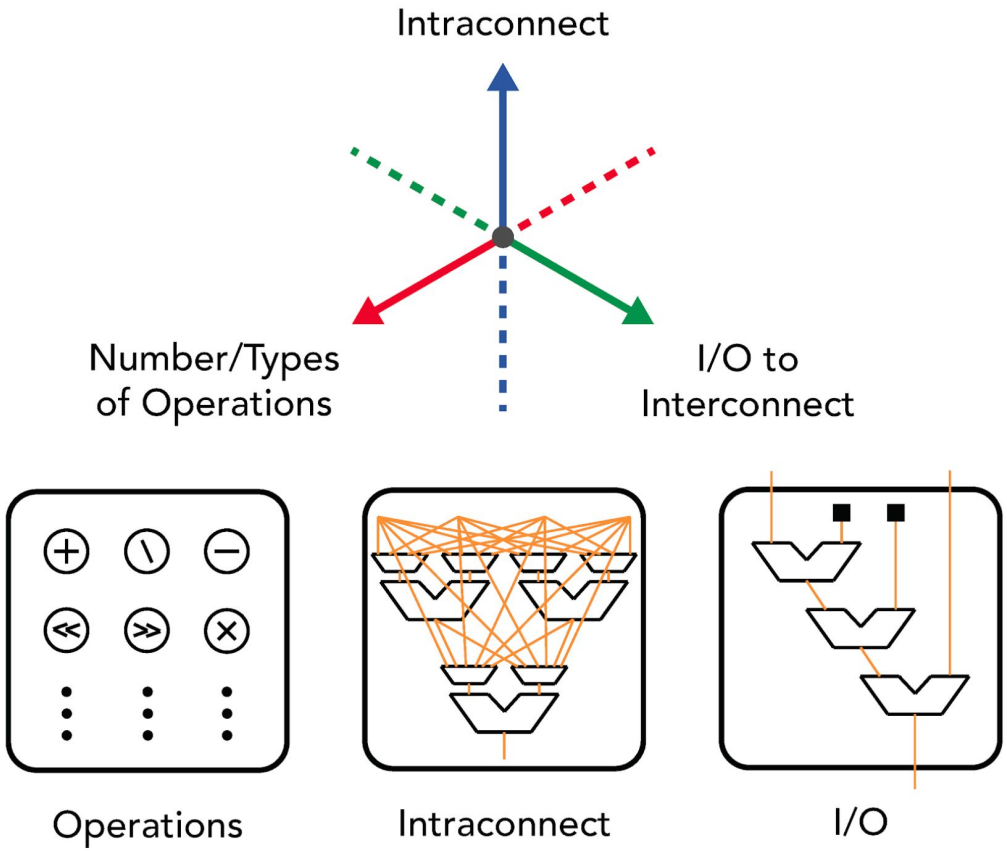
Jackson Melchert, Kathleen Feng, Ross Daly, Caleb Donovick, Khushal Sethi

Motivation

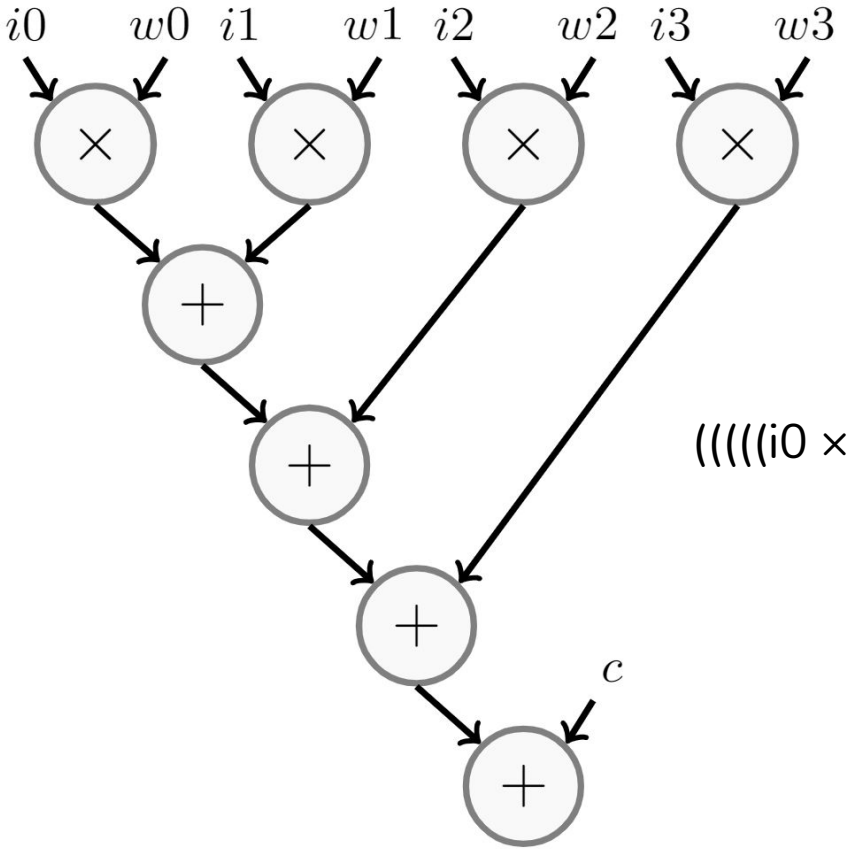
How can we generate an optimal PE architecture for a specific application domain?

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

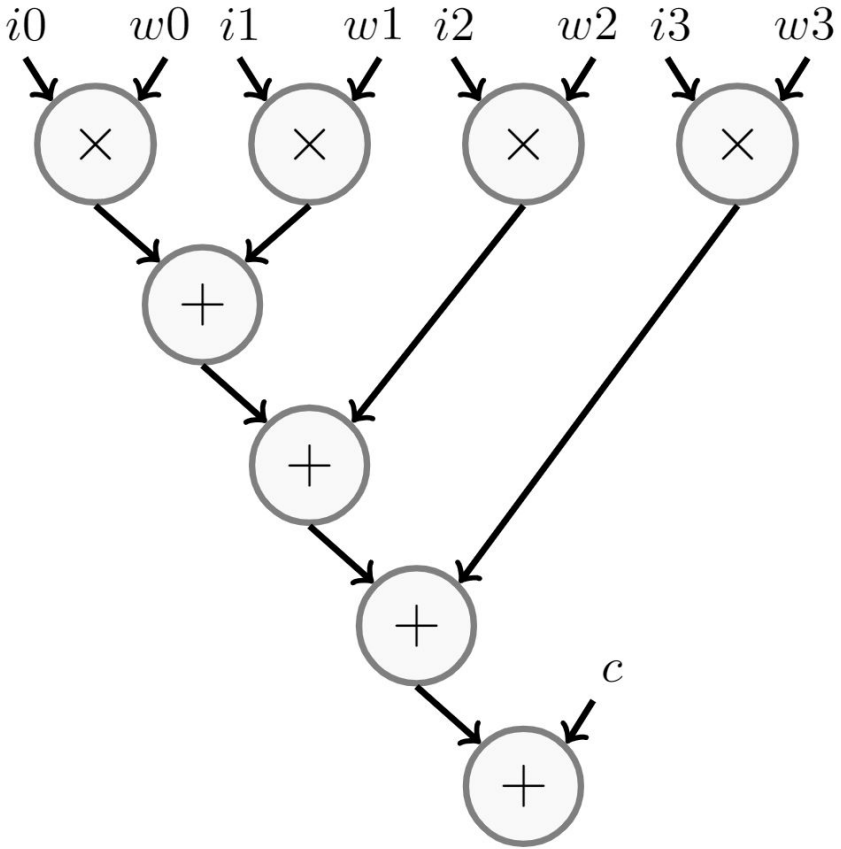


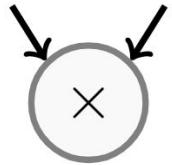
Example Convolution Dataflow Graph



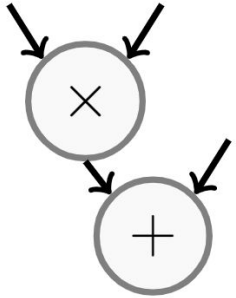
(((((i0 × w0)+ (i1 × w1)) + (i2 × w2)) + (i3 × w3)) + c)

Frequent Subgraphs of a Convolution

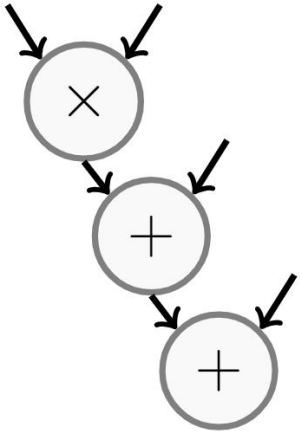




Subgraph 1
Frequency: 4



Subgraph 2
Frequency: 4



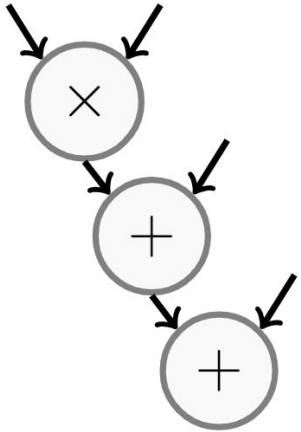
Subgraph 3
Frequency: 4

Maximal Independent Set Analysis

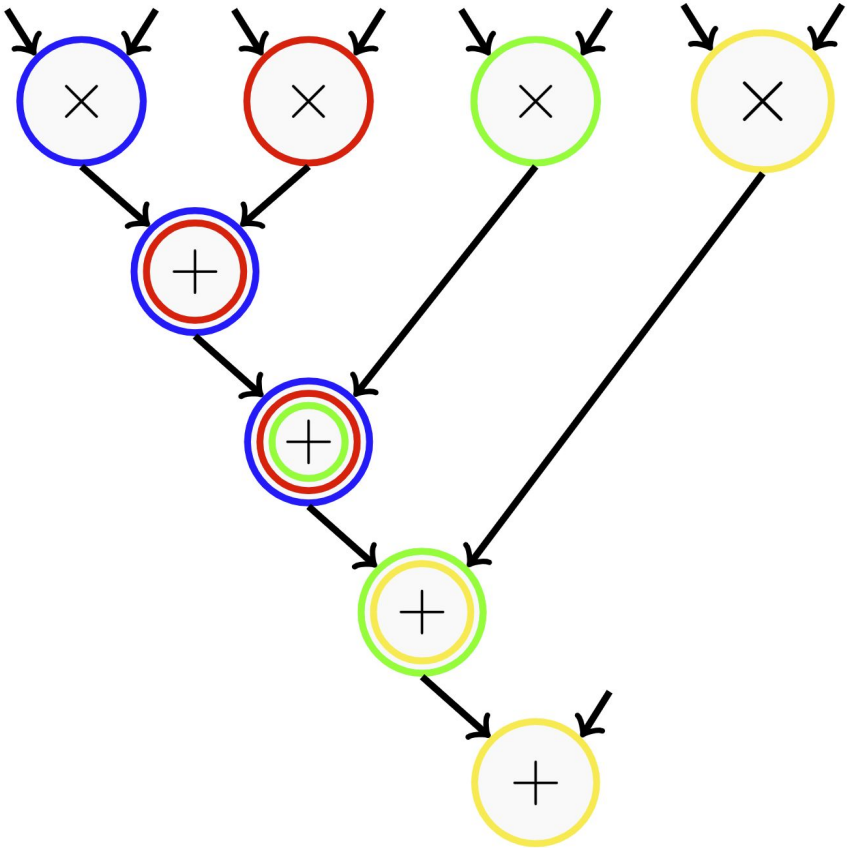
For each subgraph:

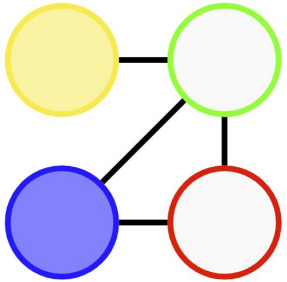
1. Represent each occurence of that subgraph as a node in a new graph
2. Add an edge between nodes if the subgraph occurrences overlap
3. Calculate the maximal independent set

Maximal Independent Set Analysis Example



Subgraph 3
Frequency 4





Maximal Independent Set
MIS Size = 2

Merging Subgraphs

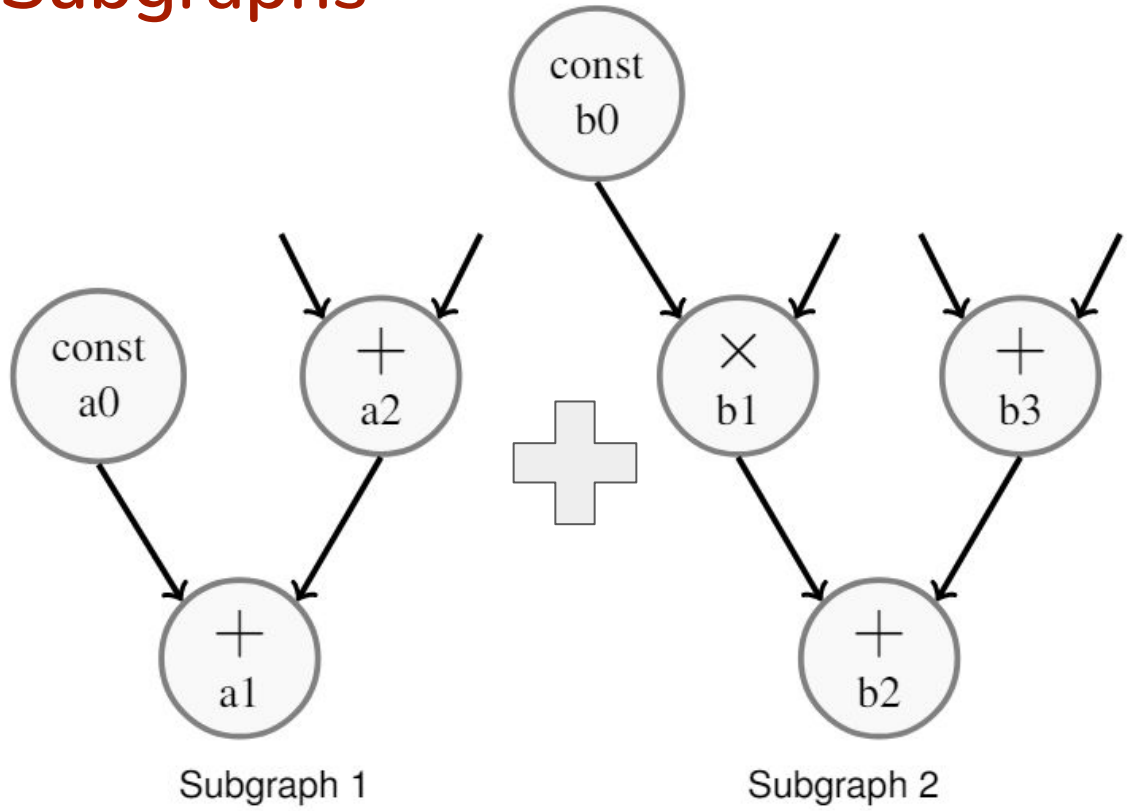
- Allows for exploration of the design space by tuning how many subgraphs are merged
- Enables better coverage of application graphs
- Allows for more effectively analyzing multiple applications
- Intelligently explores the connectivity design space axis

Merging Subgraphs

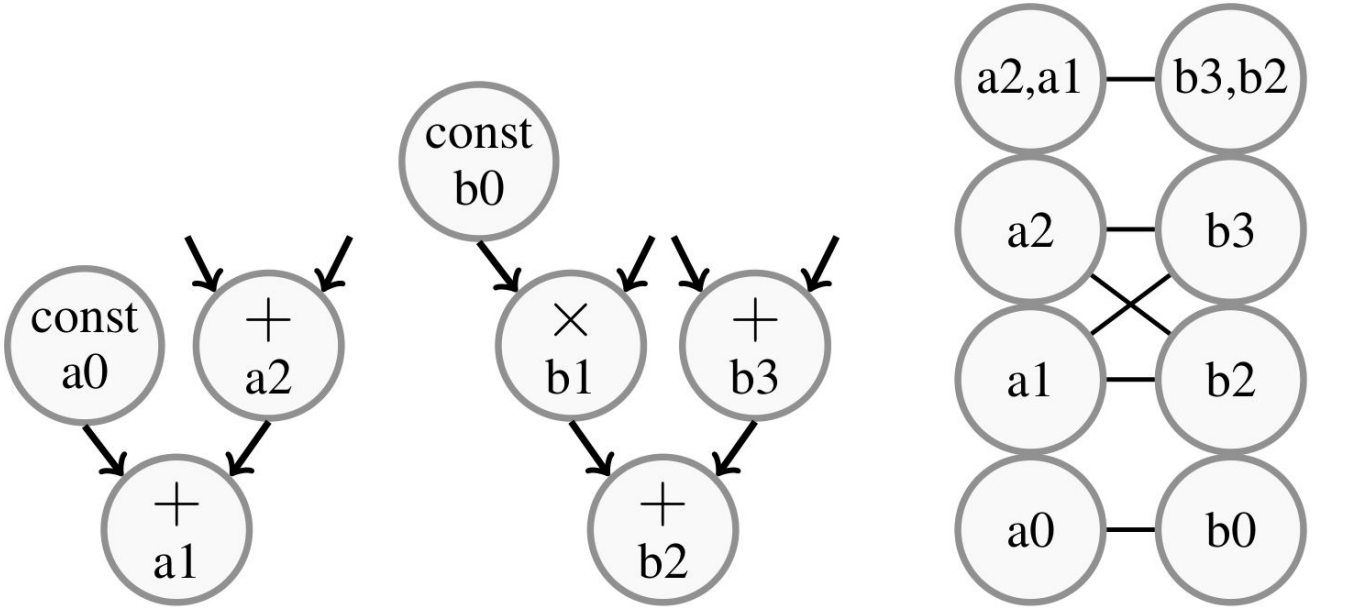
Datapath graph merging:

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

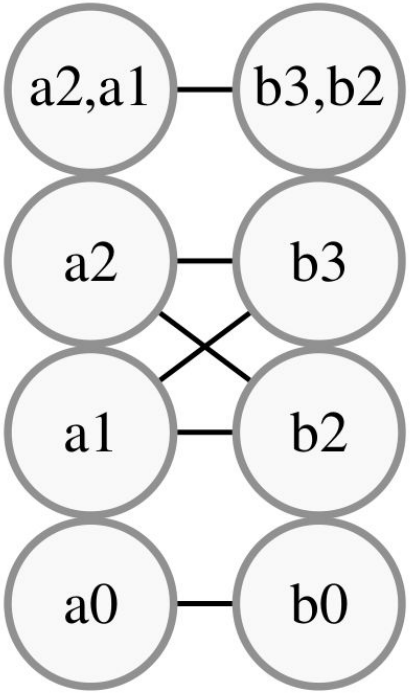


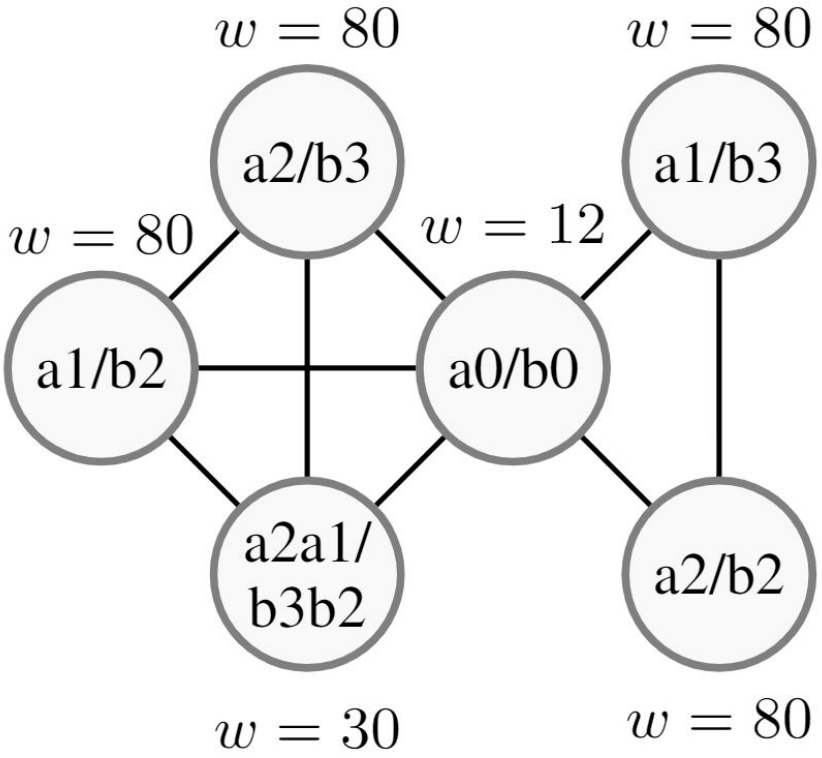
Merging Subgraphs



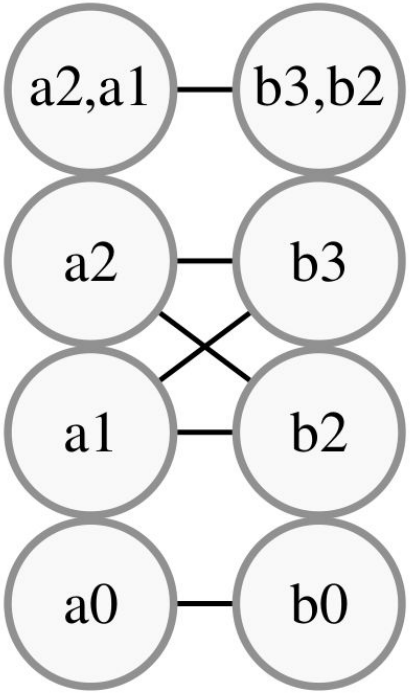
(a) Subgraph 1 (b) Subgraph 2 (c) Potential Mergings

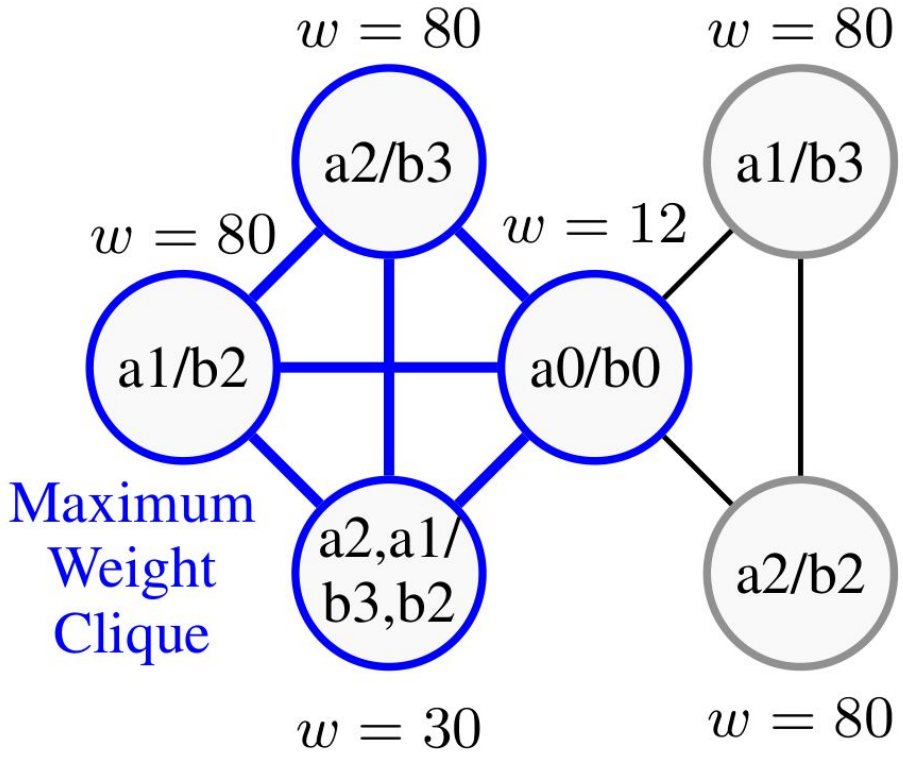
Merging Subgraphs



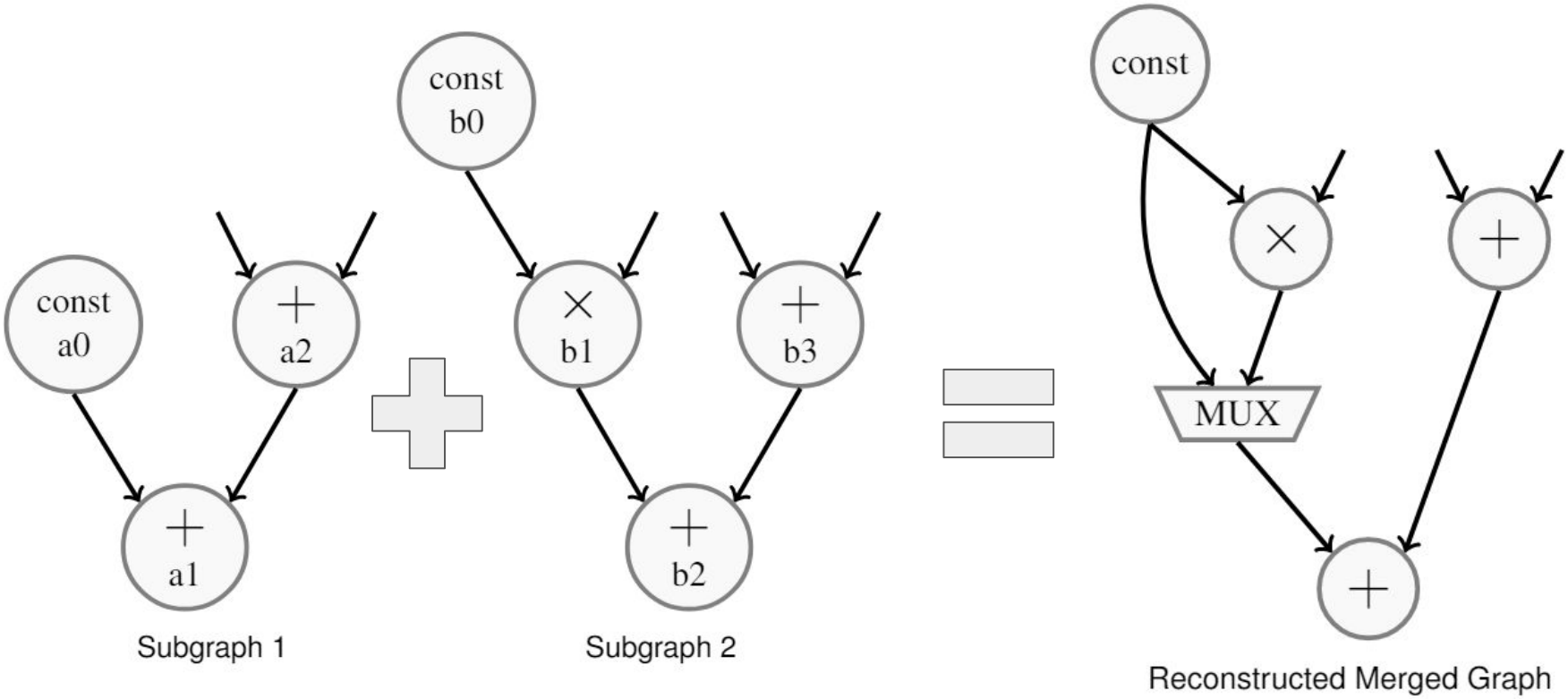


Merging Subgraphs

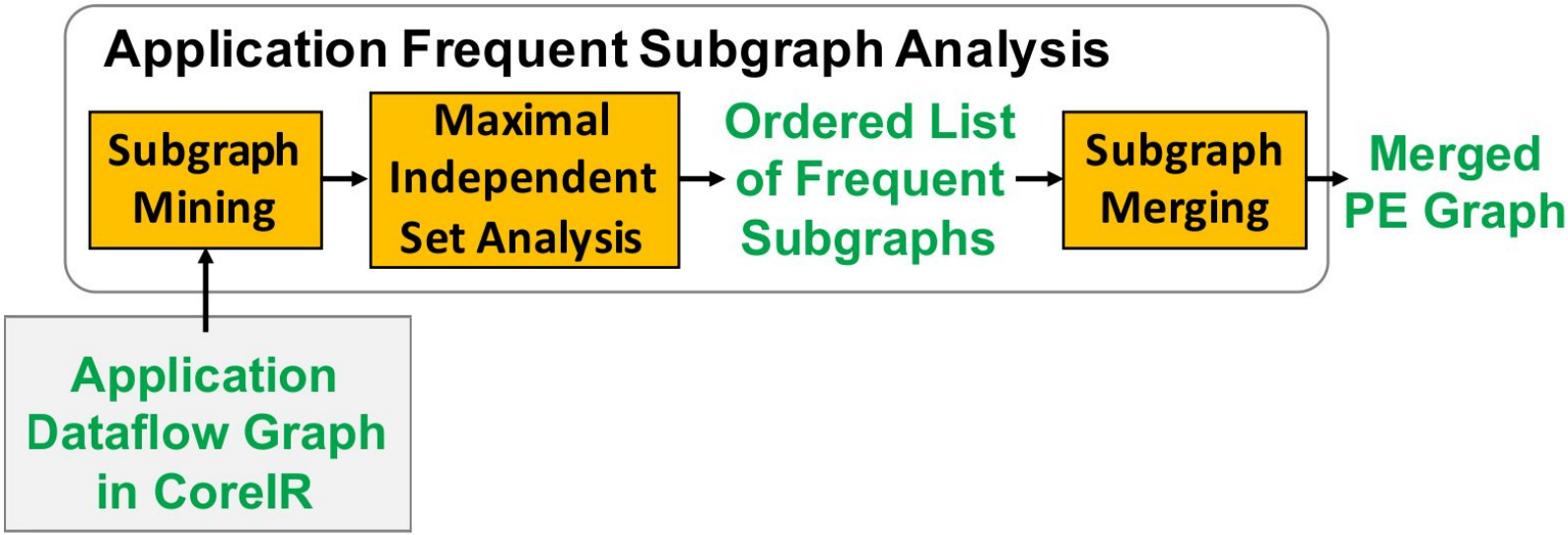




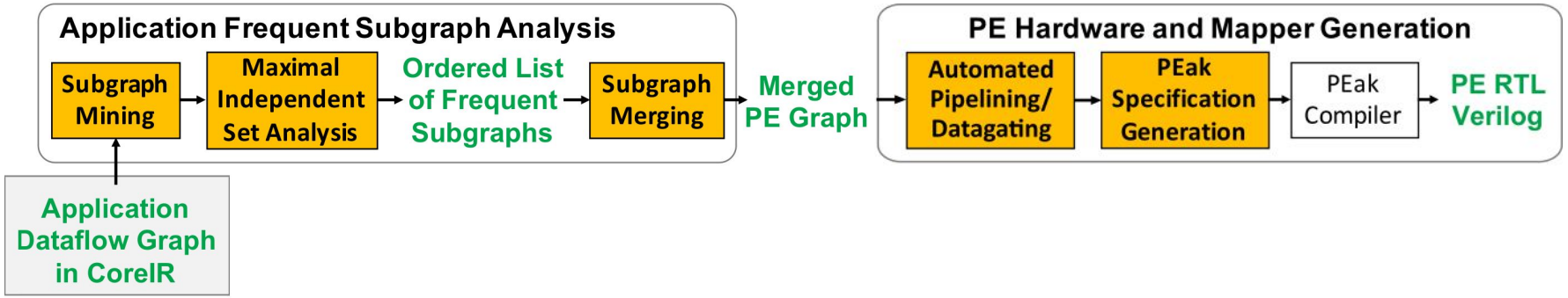
Merging Subgraphs



Design Space Exploration Framework



Design Space Exploration Framework

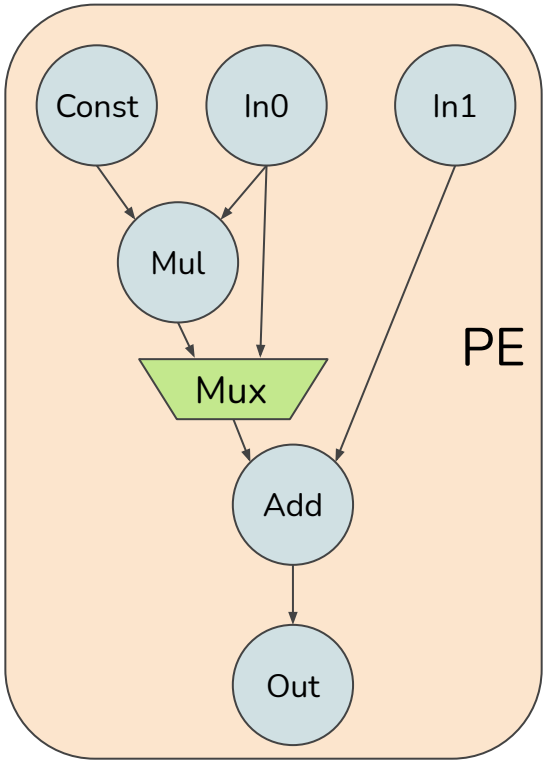


Automated Pipelining

- We need to pipeline PEs to avoid timing issues with complex PE operations
- 2 stages in automated PE pipelining:
    1. Determine the number of pipelining stages
        a. Determined using static timing analysis
    2. Retime those registers into their optimal positions
        a. Done using a retiming algorithm that minimizes the critical path through the PE

Automated Datagating

- Unused multiplier in PEs dissipate a lot of unnecessary energy

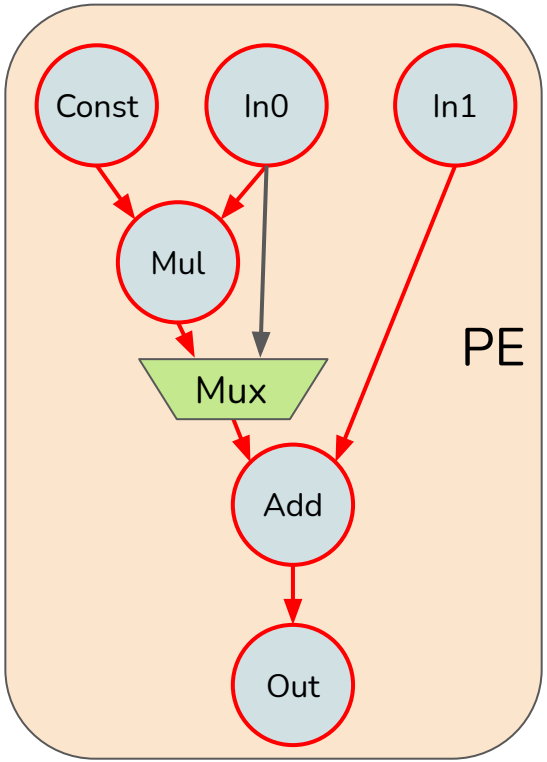


Automated Datagating

- Unused multiplier in PEs dissipate a lot of unnecessary energy
- Operation 1

  Out = Const x In0 + In1

  - Leave the operation unconstrained



Automated Datagating

- Unused multiplier in PEs dissipate a lot of unnecessary energy
- Operation 1

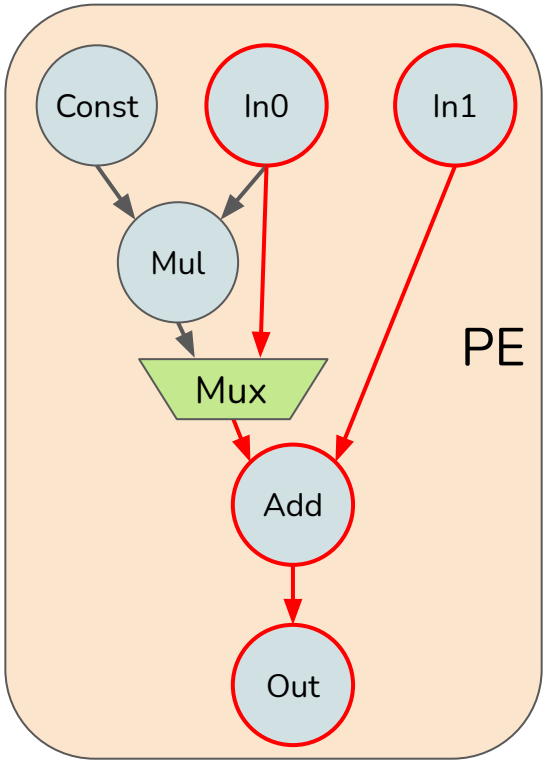
  Out = Const x In0 + In1

  - Leave the operation unconstrained

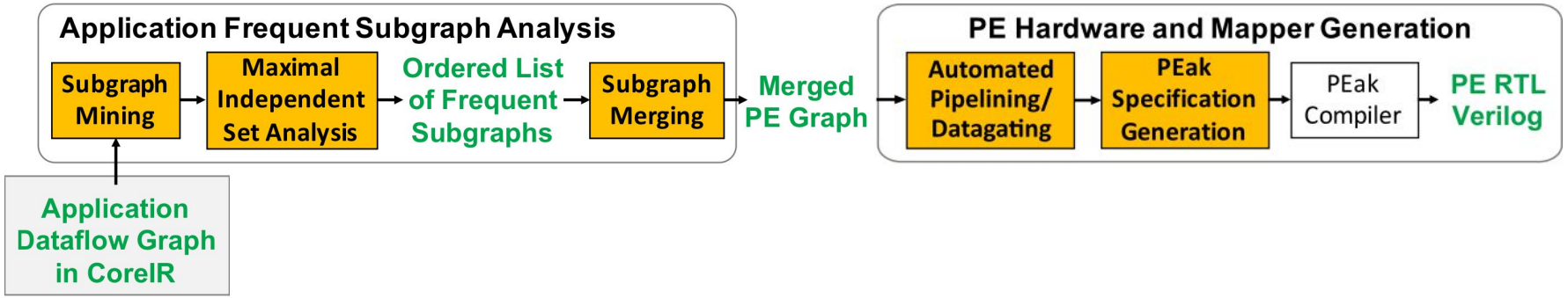
- Operation 2

  Out = In0 + In1

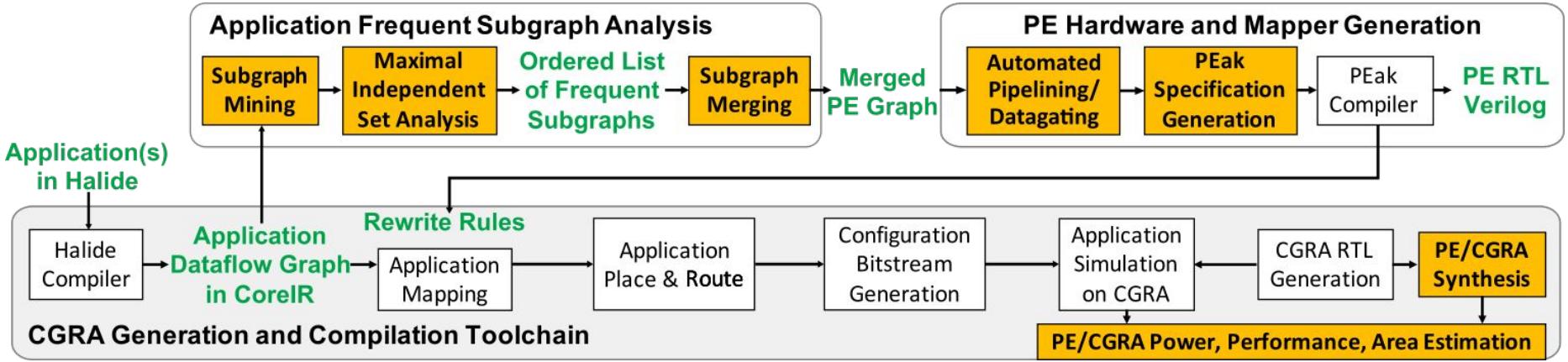
  - Constrain instruction to Mul so that it replaces Const and In0 with 0's



Design Space Exploration Framework



Design Space Exploration Framework

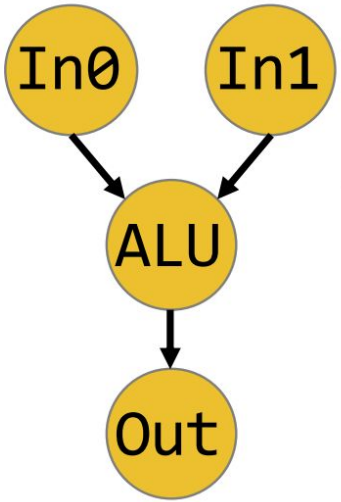


Automatic PE Mapping

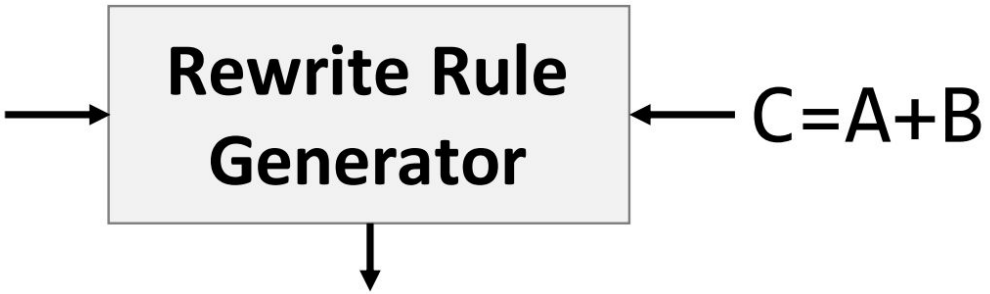
1. Rewrite Rule Generation:
    - Generating a set of rewrite rules from a PEak program

2. Instruction Selection:
    - Transforming a graph of CoreIR operations to a graph of PEs using the rewrite rules

Rewrite Rule Synthesis



Formal Model of PE

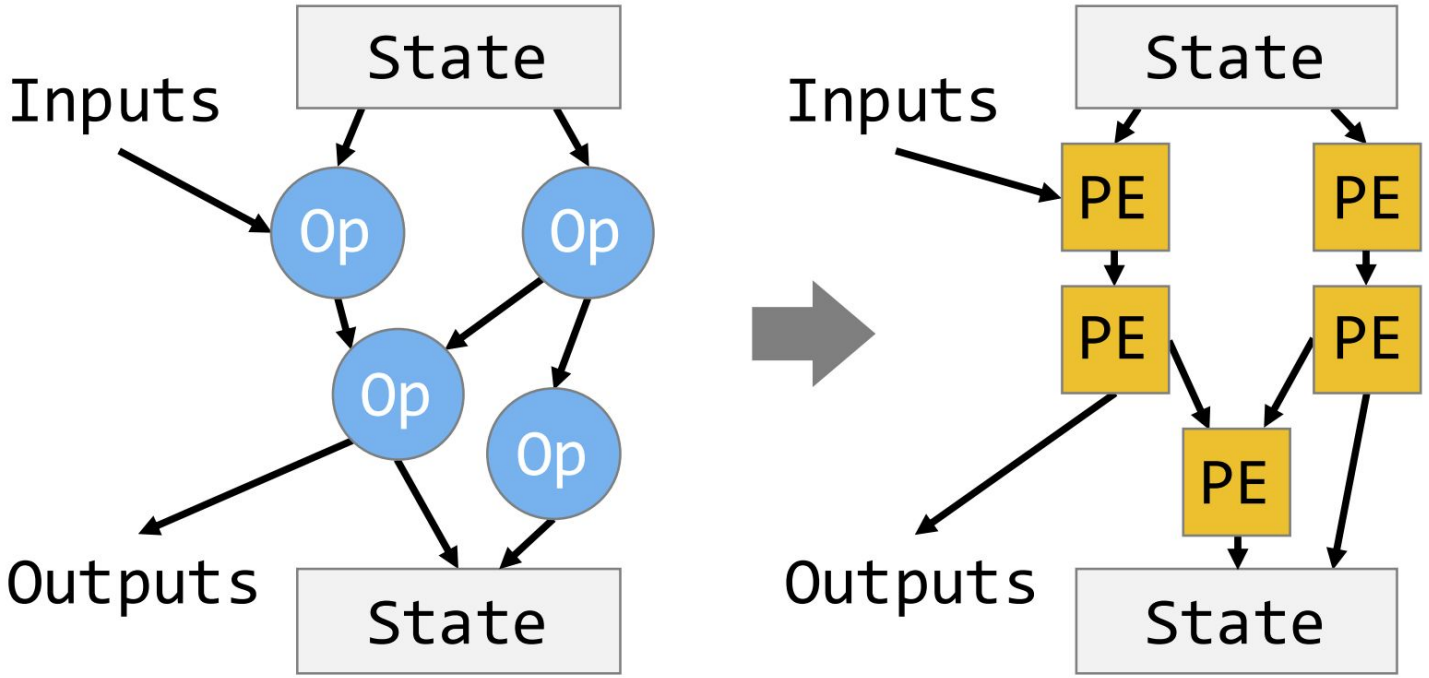


In0 ⇔ A
In1 ⇔ B
ALU ⇔ +
Out ⇔ C

Rewrite Rule

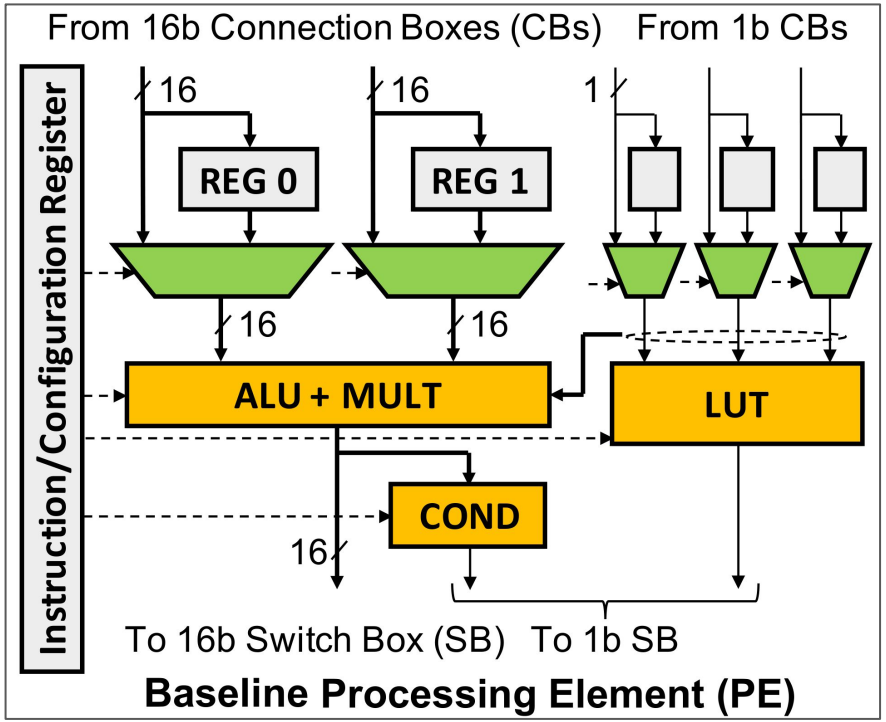
Formal Model of Operation

Instruction Selection

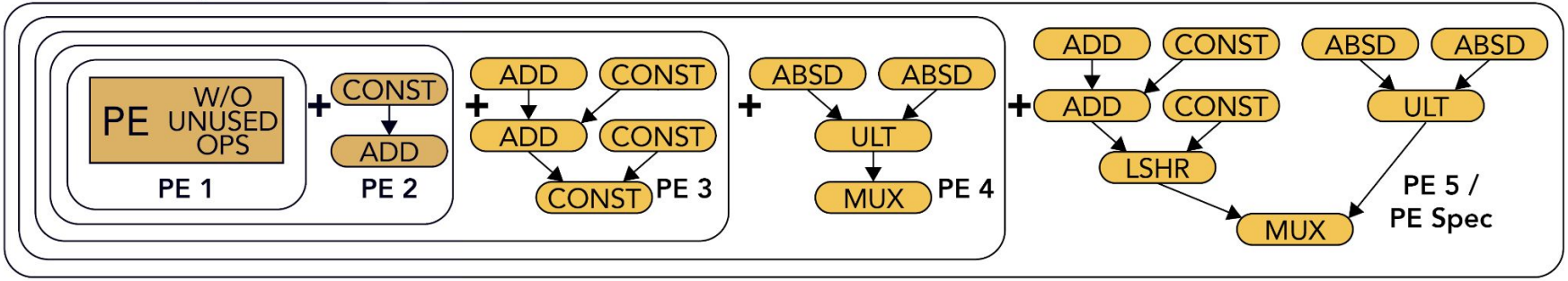


Evaluation - Baseline PE

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

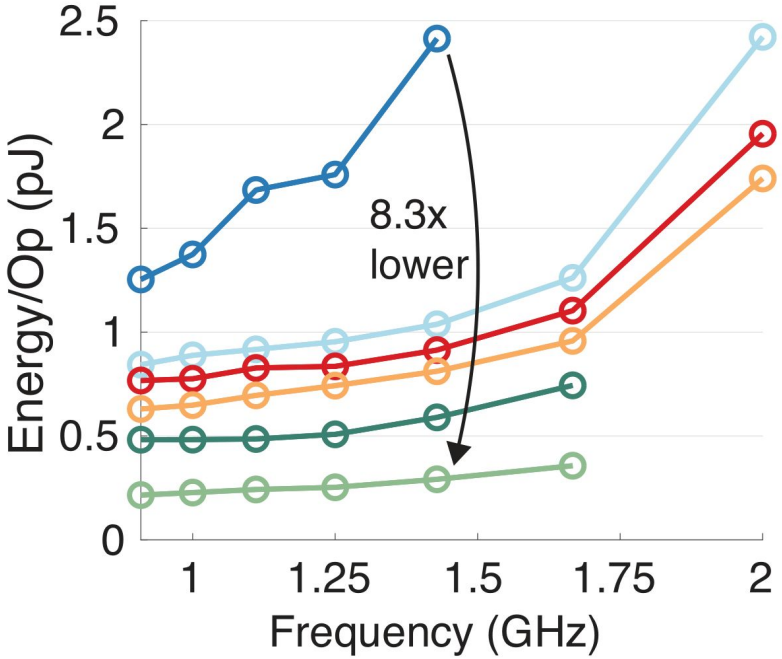


Camera Pipeline Results



PE Variation	Num PEs	PE Area
Baseline	348	750.4
1	318	585.1
2	283	585.9
3	187	620.5
4	148	758.6
5	140	727.3

Camera Pipeline Results



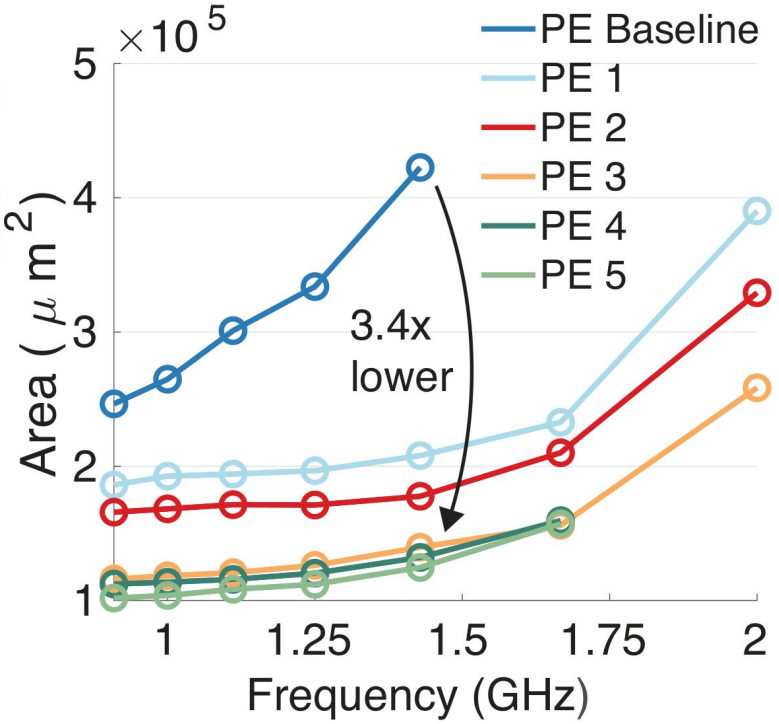


Image Processing Results

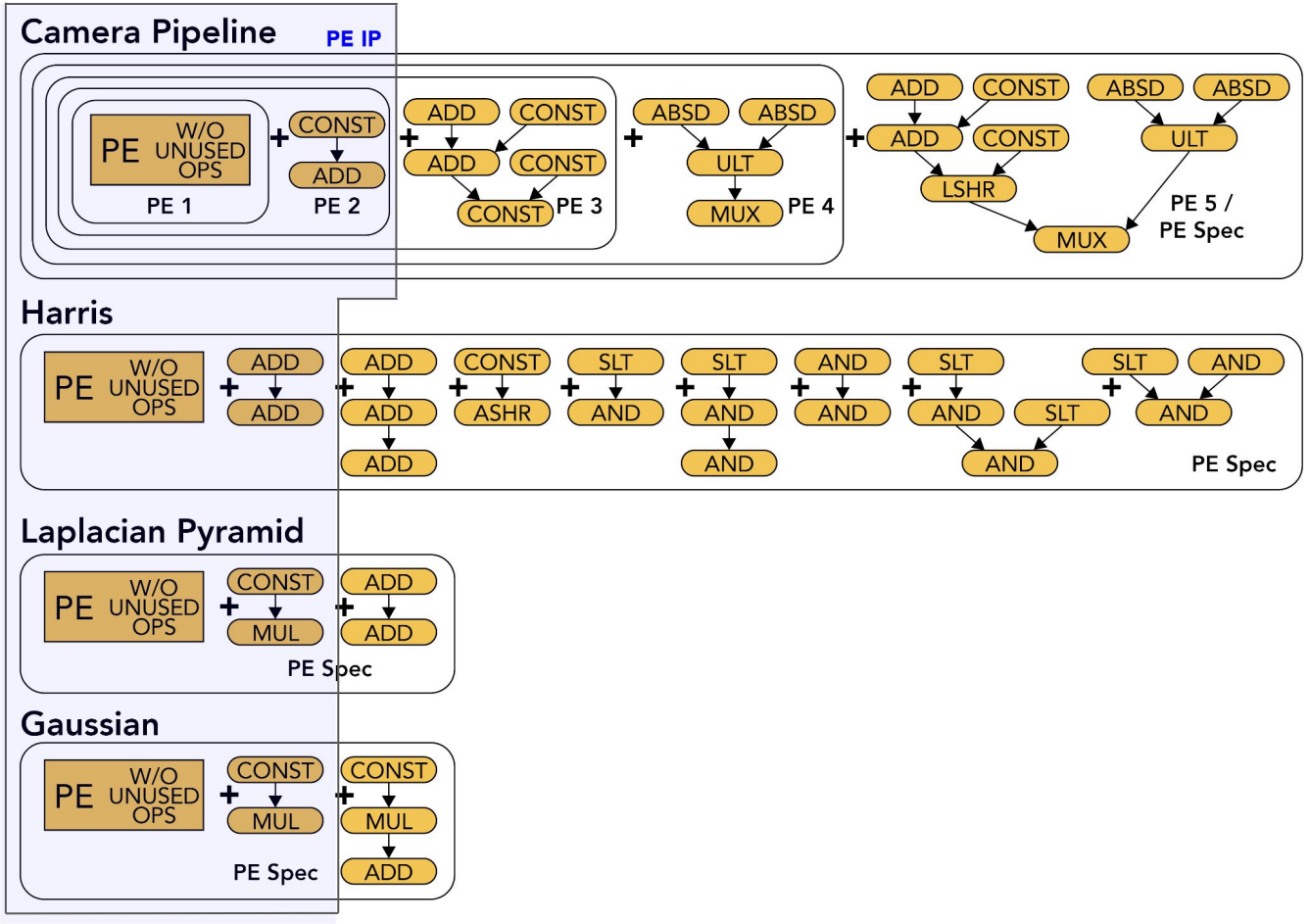


Image Processing Results

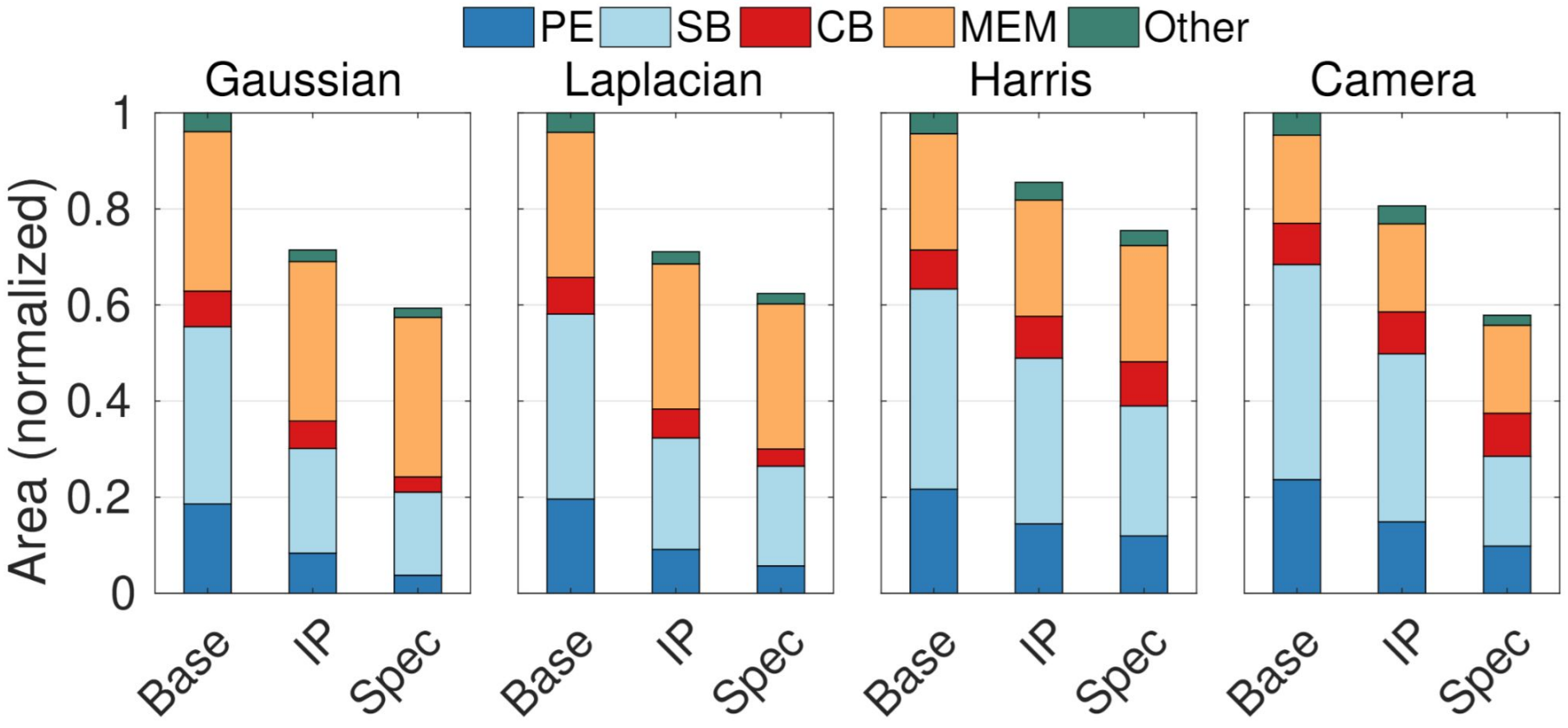


Image Processing Results

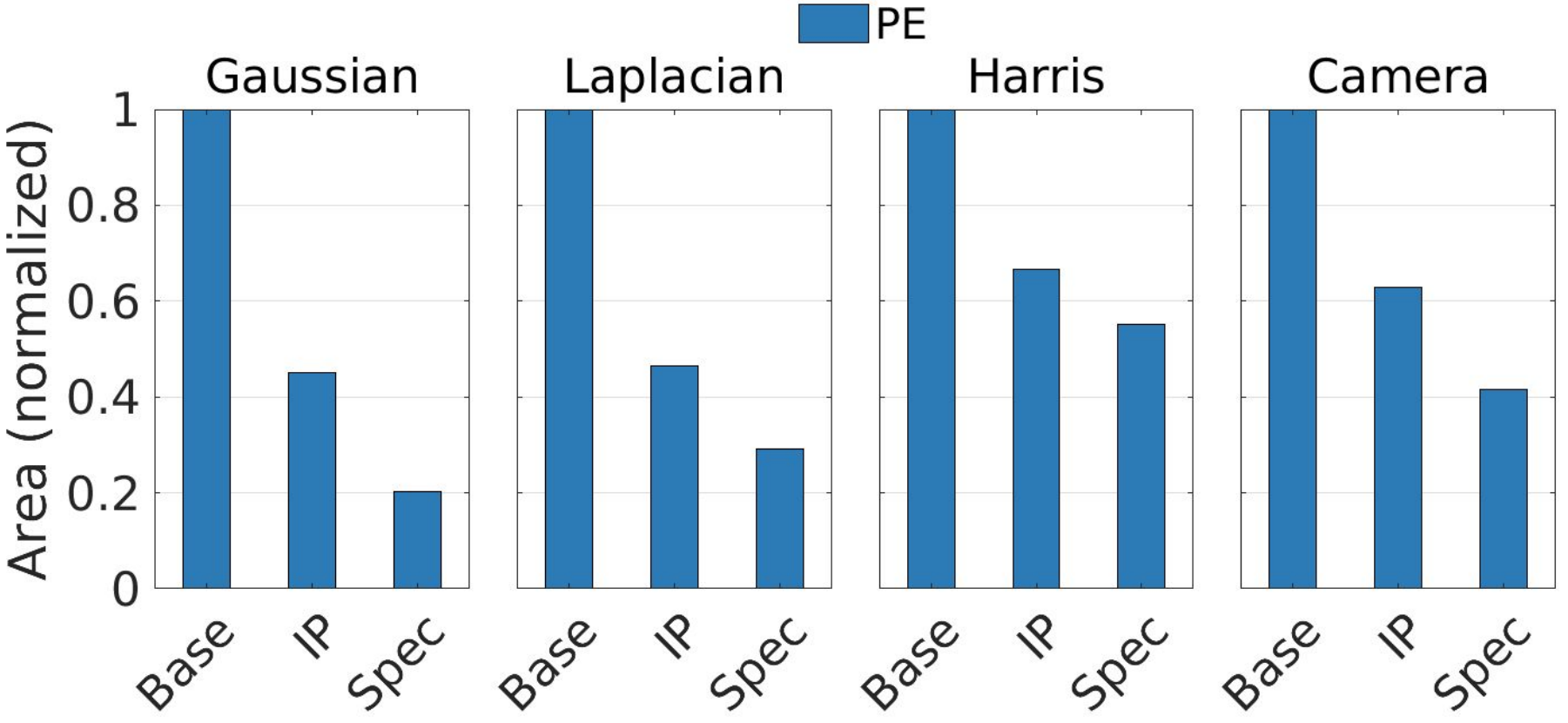


Image Processing Results

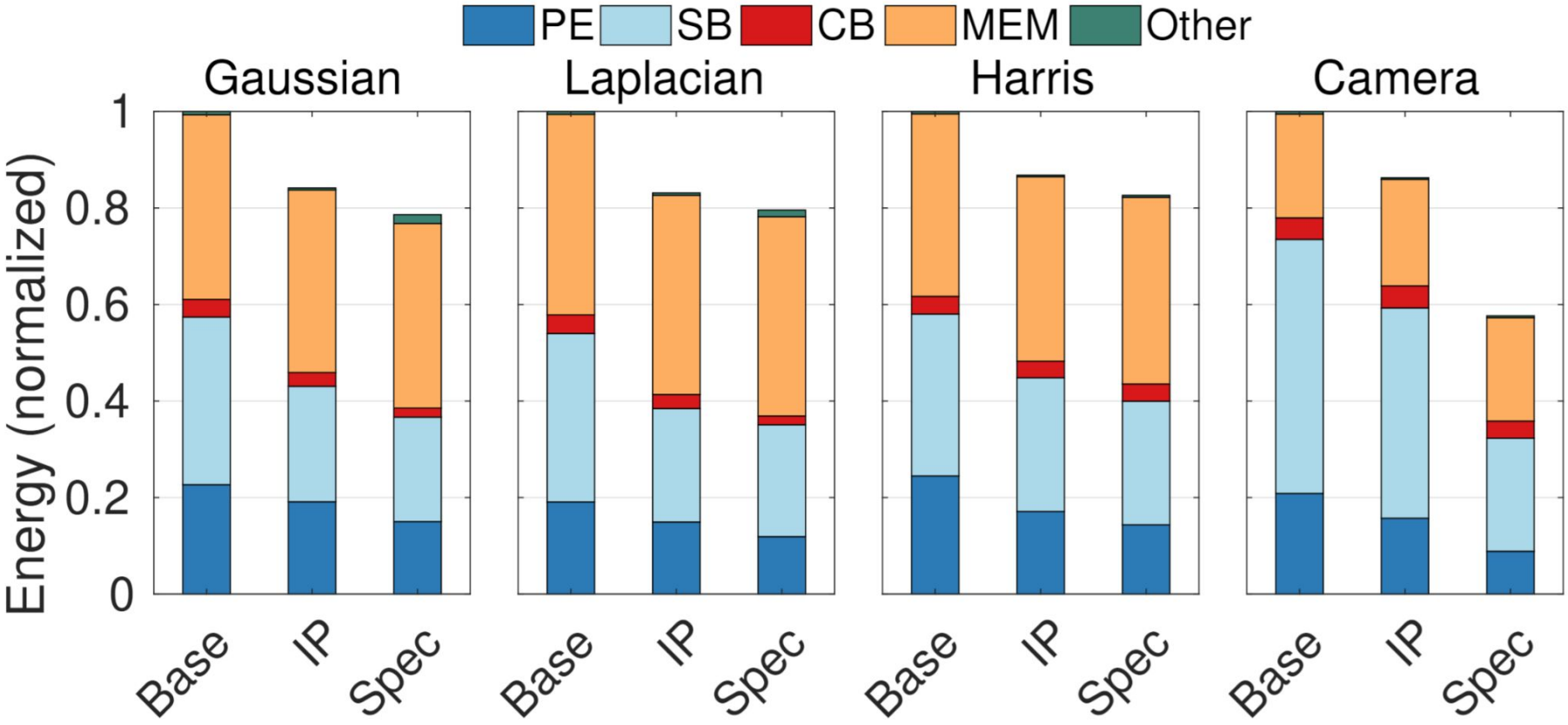
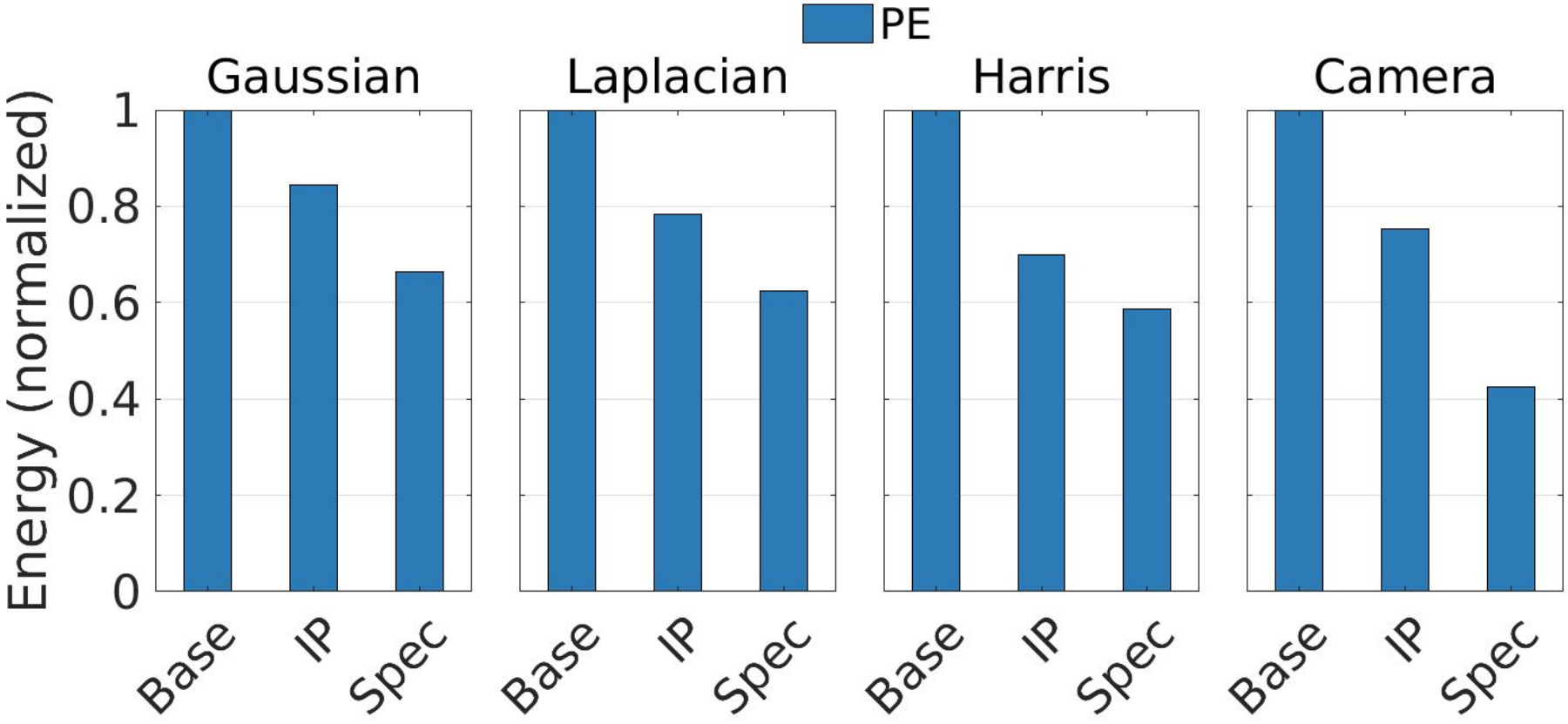
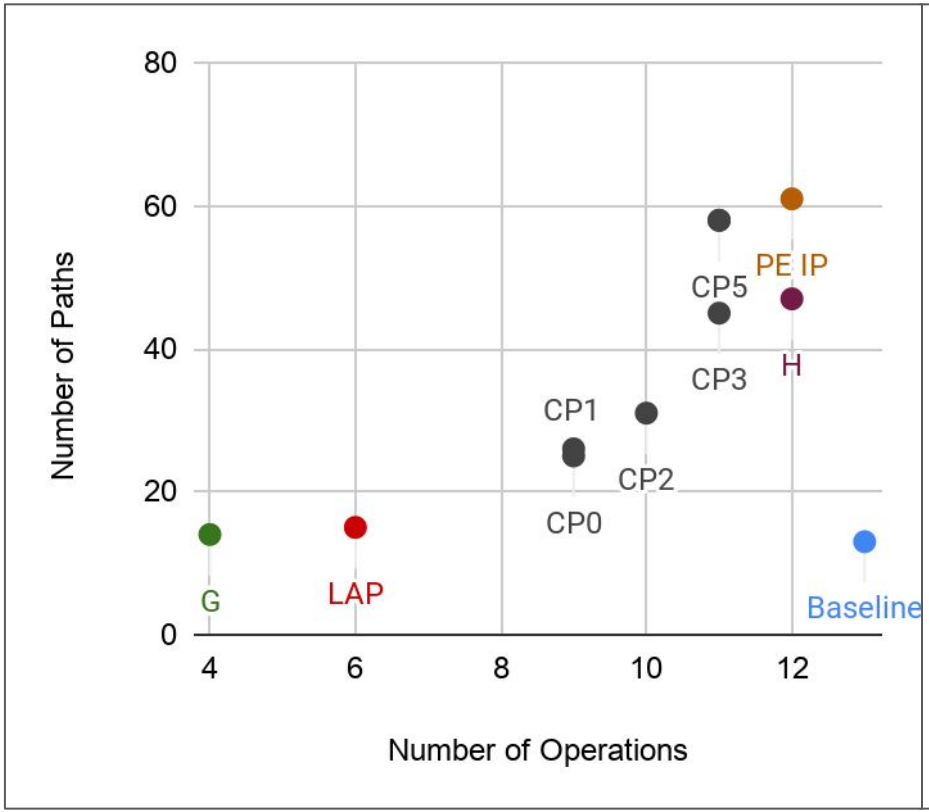
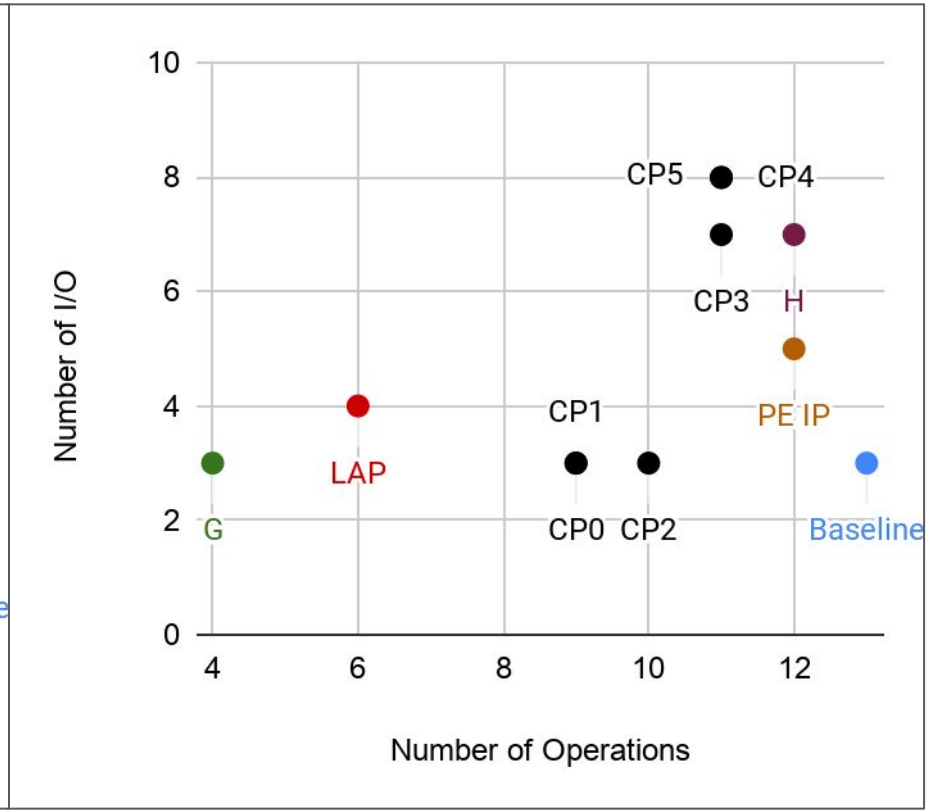


Image Processing Results



Design Space of Image Processing PEs





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