Hybrid Optimization/Heuristic Instruction Scheduling for Programmable Accelerator Codesign

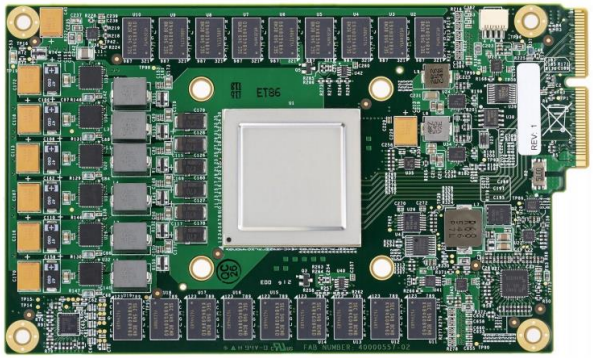
Tony Nowatzki[*] Newsha Ardalani[†]
Karthikeyan Sankaralingam[‡] Jian Weng[*]

[*] [†] [‡]

UCLA
Simple Machines
POLY ARCH
THE UNIVERSITY of WISCONSIN MADISON
VERTICAL

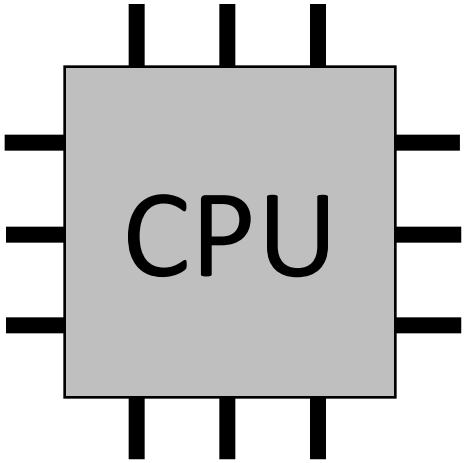
PACT 2018, Nov. 3rd

Google TPU



Coarse
Grain
Reconfigurable
Architectures

>



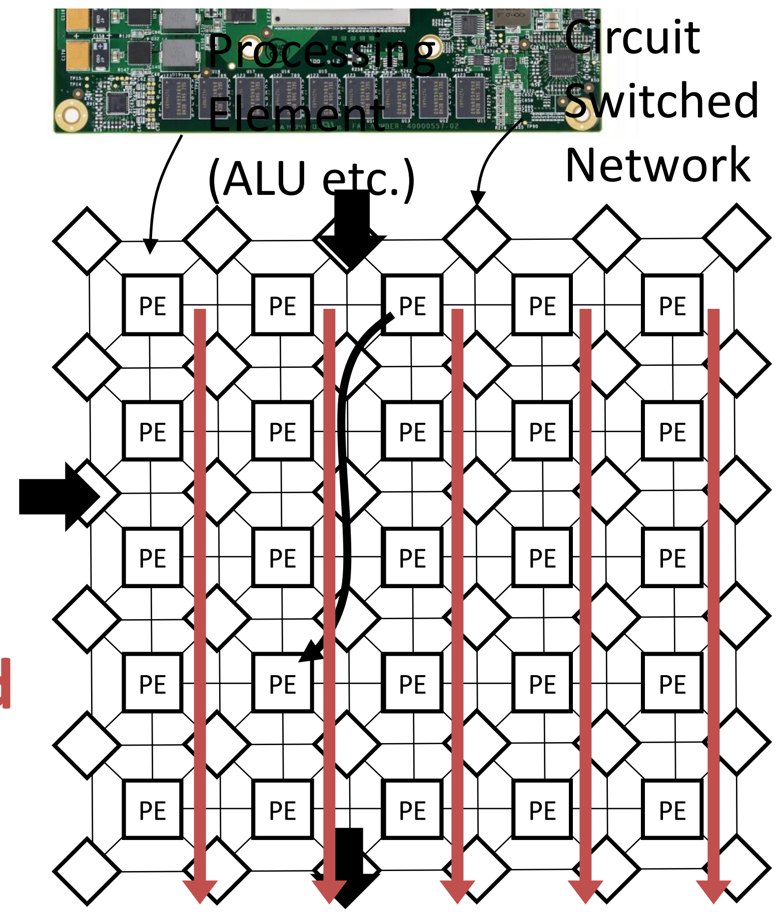
Problem: How can we map computation to this effectively?

Coarse
Grain
Reconfigurable
Architectures

What if
do dens
with la

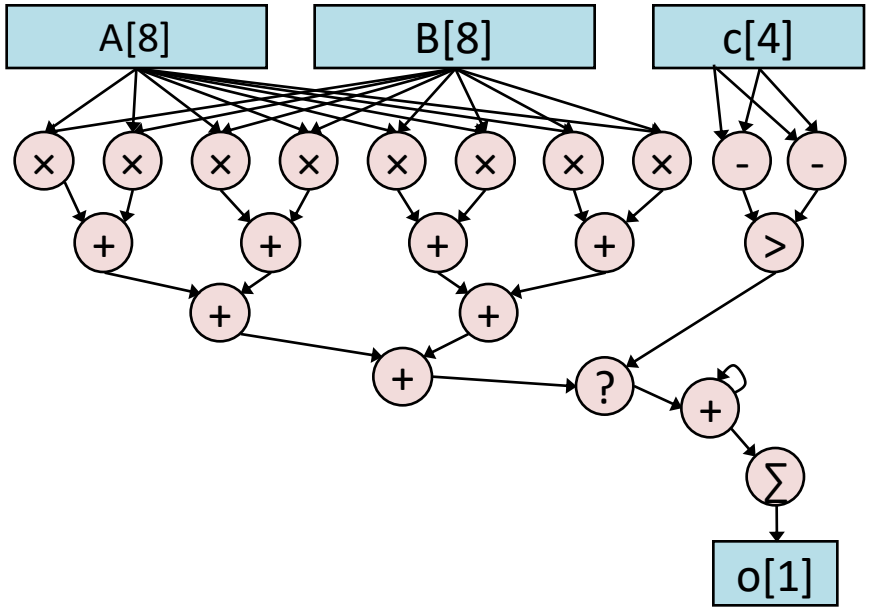
Examples: Charm, Camel, Stream-Dataflow, FPCA, DySER (almost), LSSD, Morphosys, etc …

Perfectly-pipelined execution (1 computation per cycle)

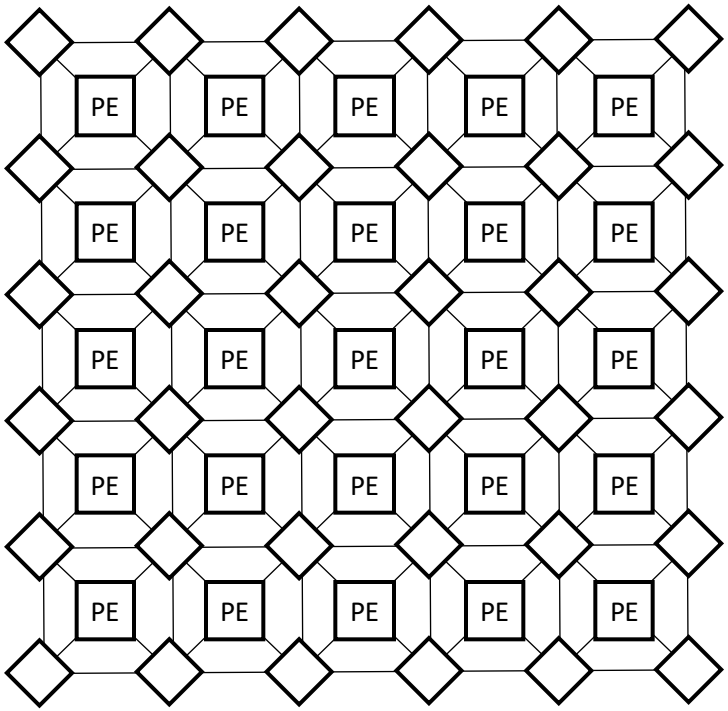


(dedicated PE CGRA)

Computation Graph

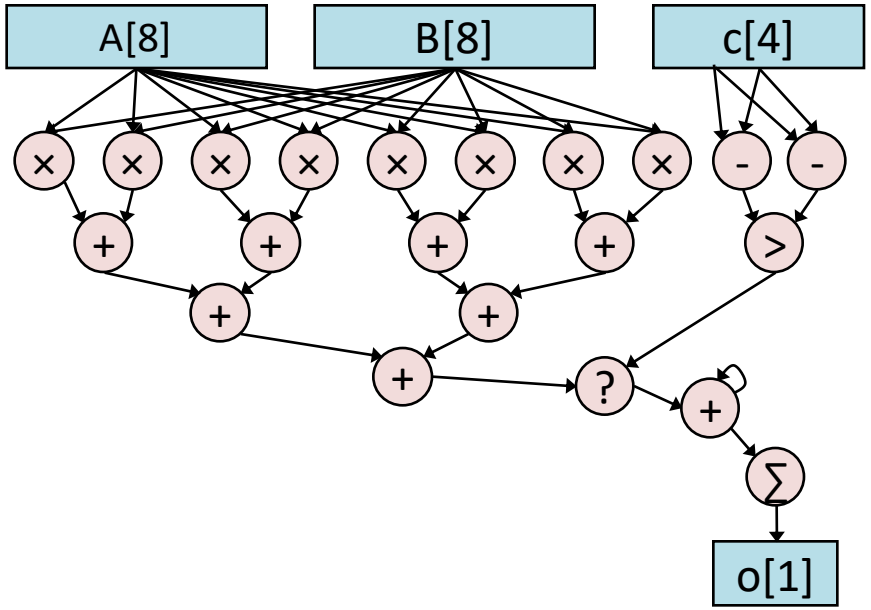


Hardware Graph

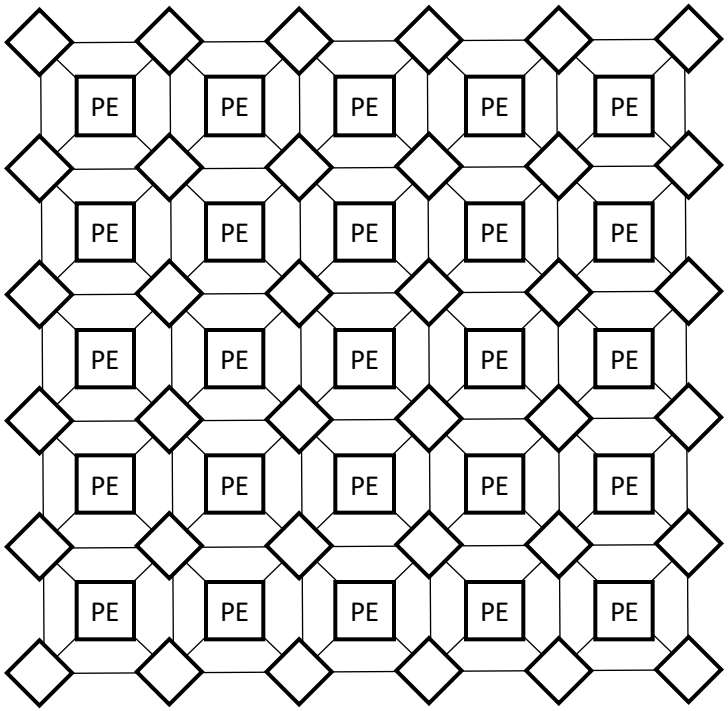


Mapping → **Routing** → **Timing**

Computation Graph

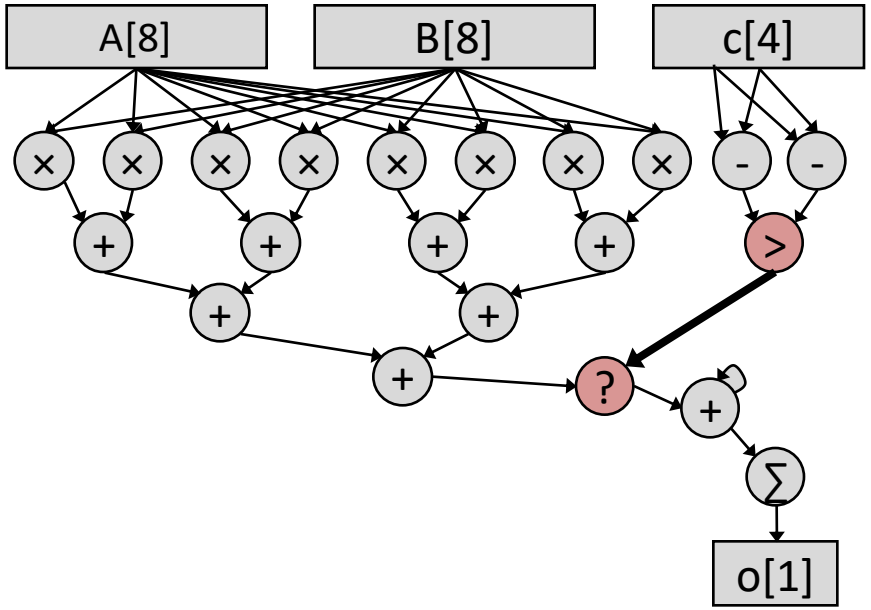


Hardware Graph

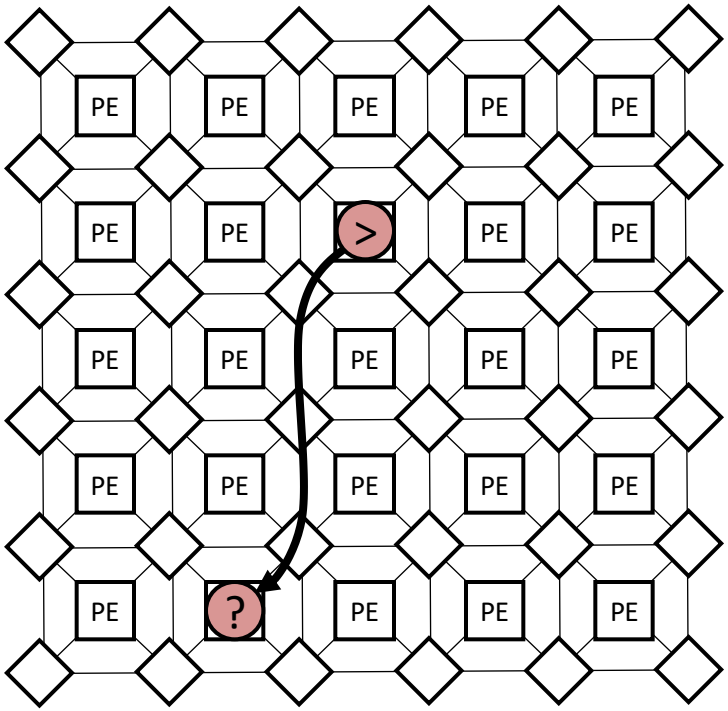


Mapping → **Routing** → **Timing**

Computation Graph

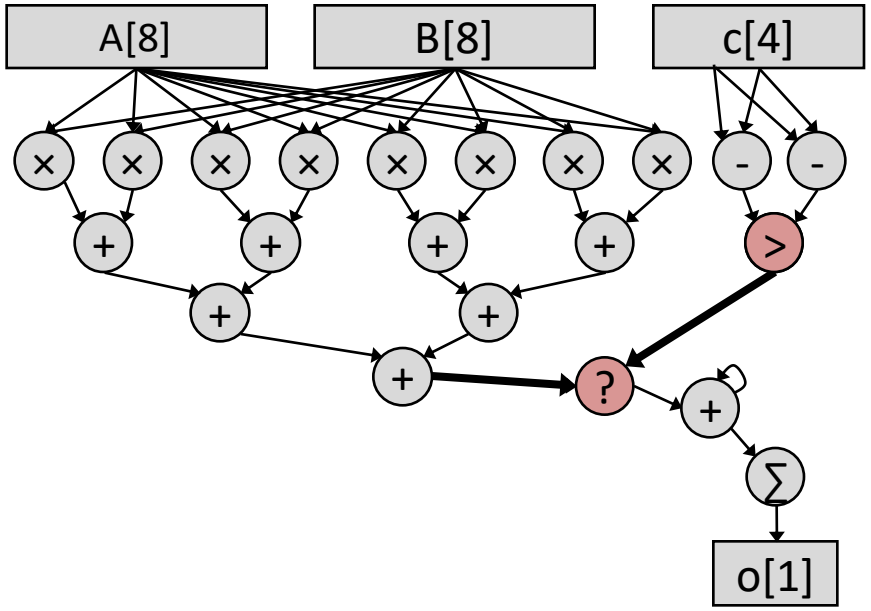


Hardware Graph

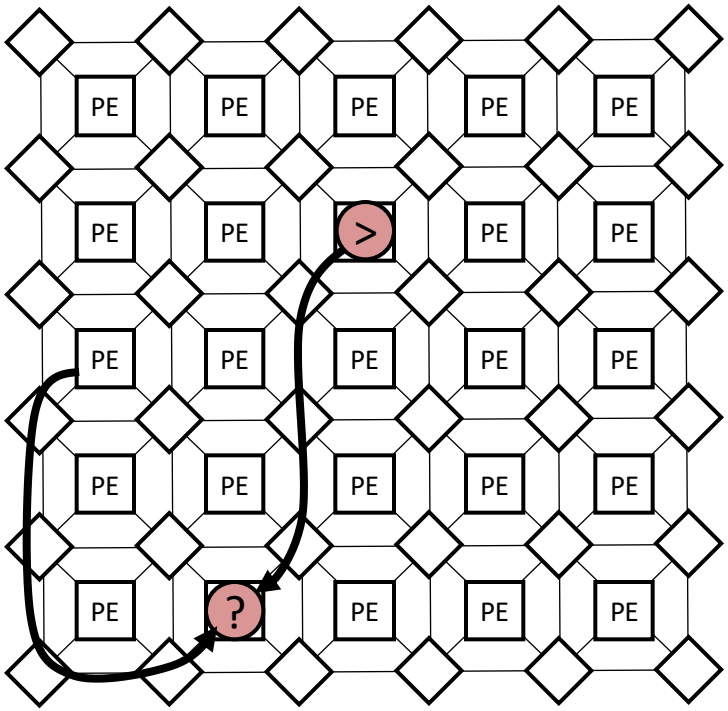


Mapping → **Routing** → Timing

Computation Graph



Hardware Graph



Mapping → Routing → **Timing**

Technique Overview

Joint Optimization

(days for solution)

Incremental Optimization

Joint Heuristic

(high area increase or integer factors performance loss)

Our Approach:
1. Phase Overlapping
2. Hybrid Scheduling

(full throughput & seconds to minutes & low area overhead)

Outline

- “Systolic” CGRAs
  - Throughput sensitivity and fractional initiation intervals.
  - Techniques for delay matching
- Scheduling Approach 1: Phasing-Overlapping
  - Tractable Optimization through Overlapping
- Scheduling Approach 2: Hybrid Scheduling
  - Heuristic to reduce search space
  - Optimization to deal with tricky cases
- Evaluation

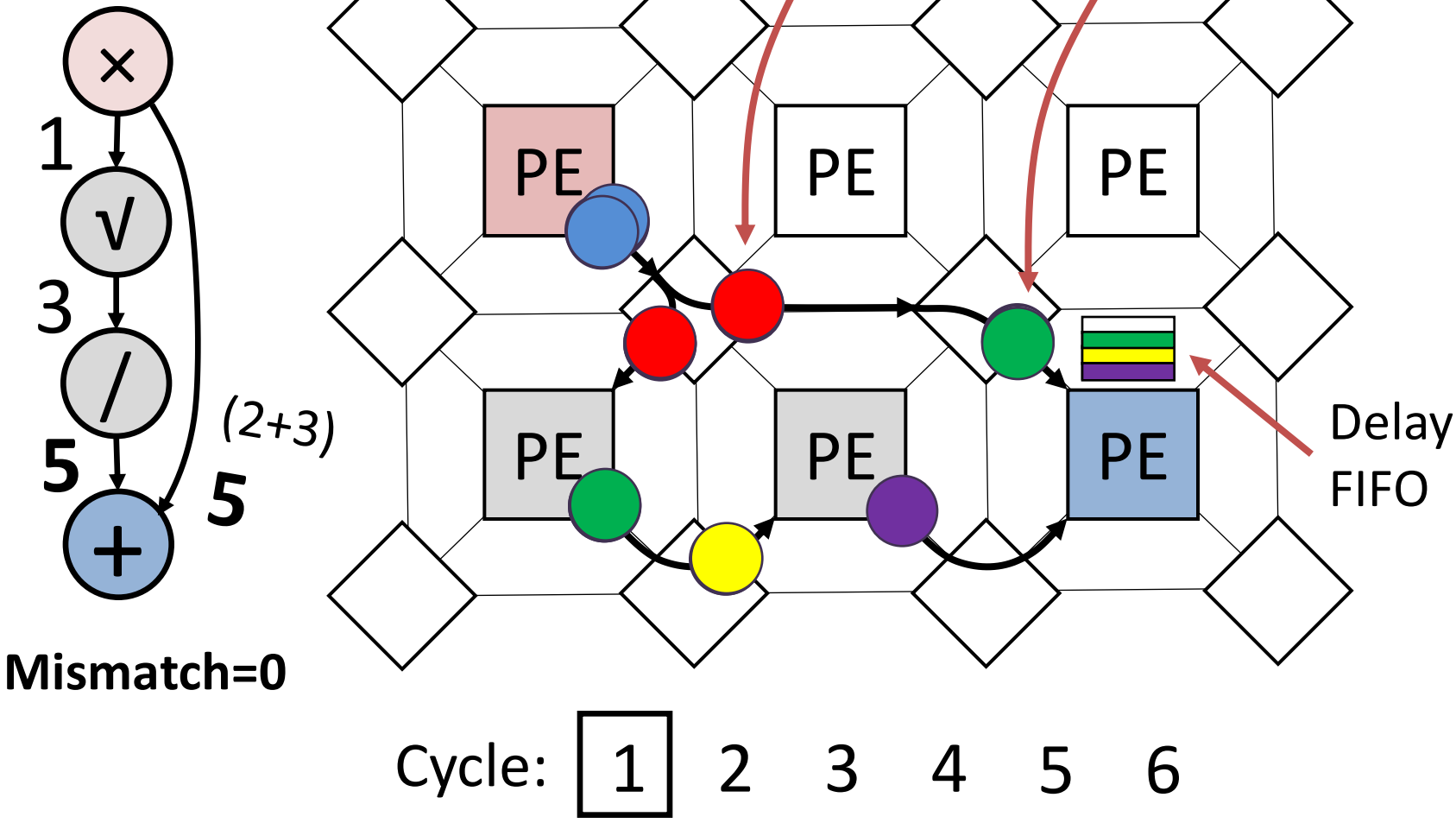
Requirements for Full-Pipelining

1. **Sufficient Data Bandwidth:** Data has to arrive at a bandwidth of sufficient for one set of inputs / cycle.

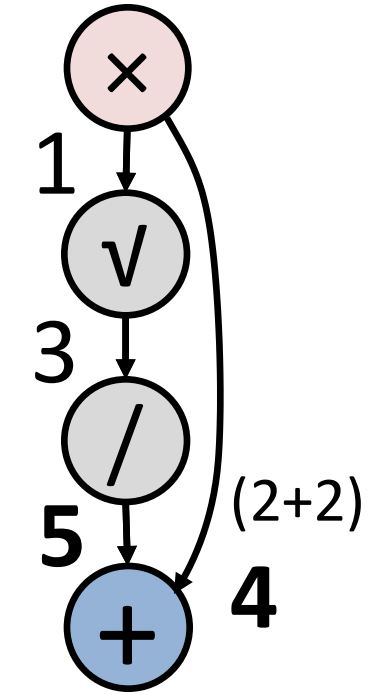
2. **No Compute Contention:** Each instruction gets a dedicated compute unit.
3. **No Routing Contention:** Each dependence gets a dedicated routing path.

4. **No Storage Contention:** Each operand is guaranteed a free storage location at each step.

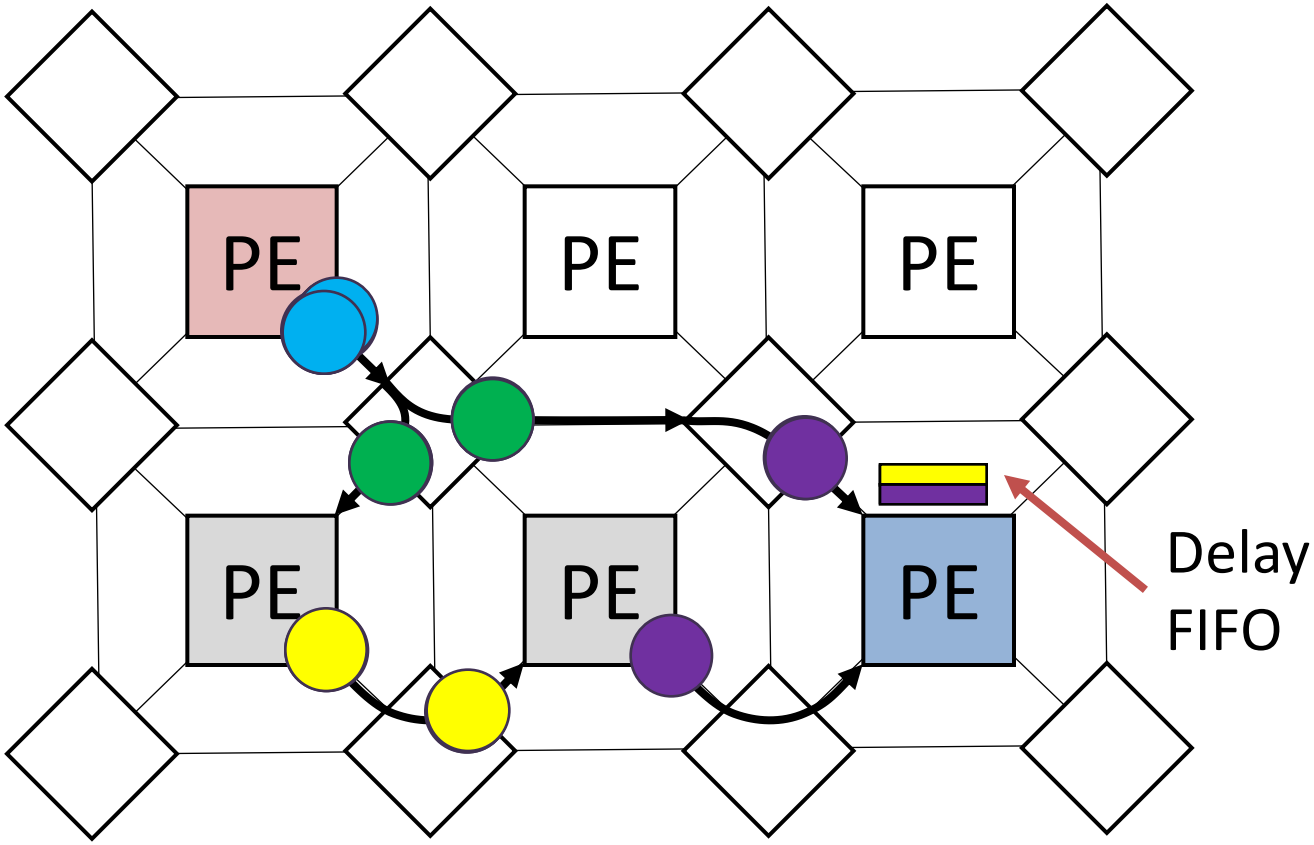
Fully-pipelined Systolic CGRA



Fully-pipelined Systolic CGRA



Mismatch=1



Cycle: 1 2 ~~3~~ 4 5 ~~6~~

Throughput Formula

$$\text{Throughput} = \frac{\text{FIFO Length}}{\text{Max. Mismatch + FIFO Length}}$$

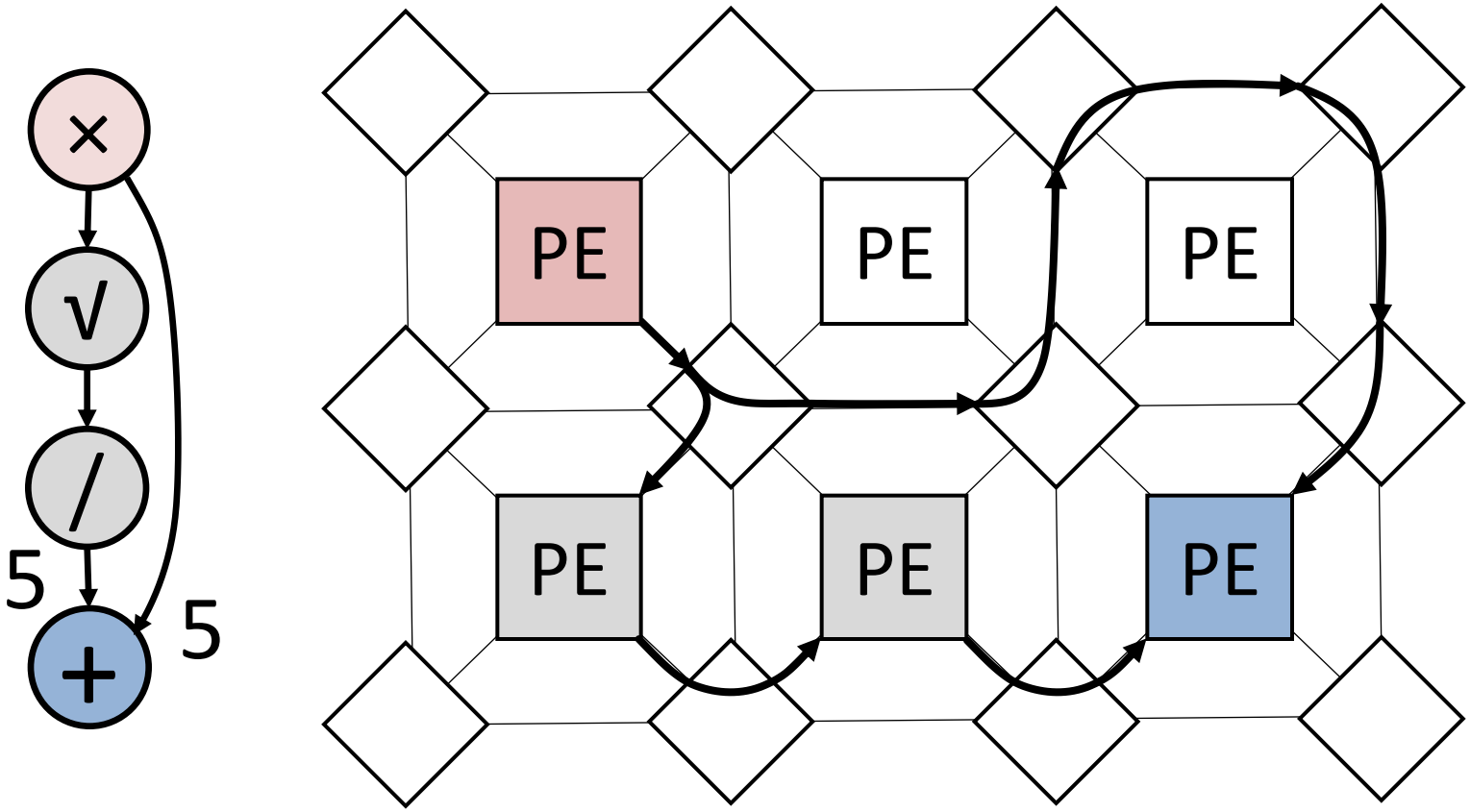
Delay-FIFOs Help Reduce Latency Mismatch

Provide buffer space which increases throughput

But at the cost of area…

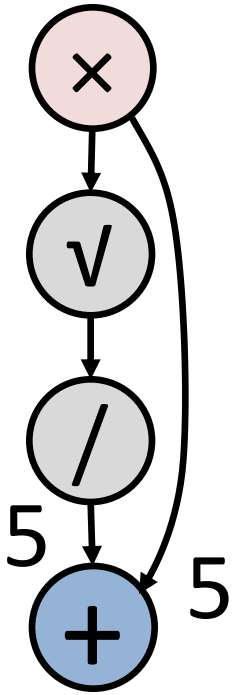
- Inverse of initiation interval in compiler terminology.

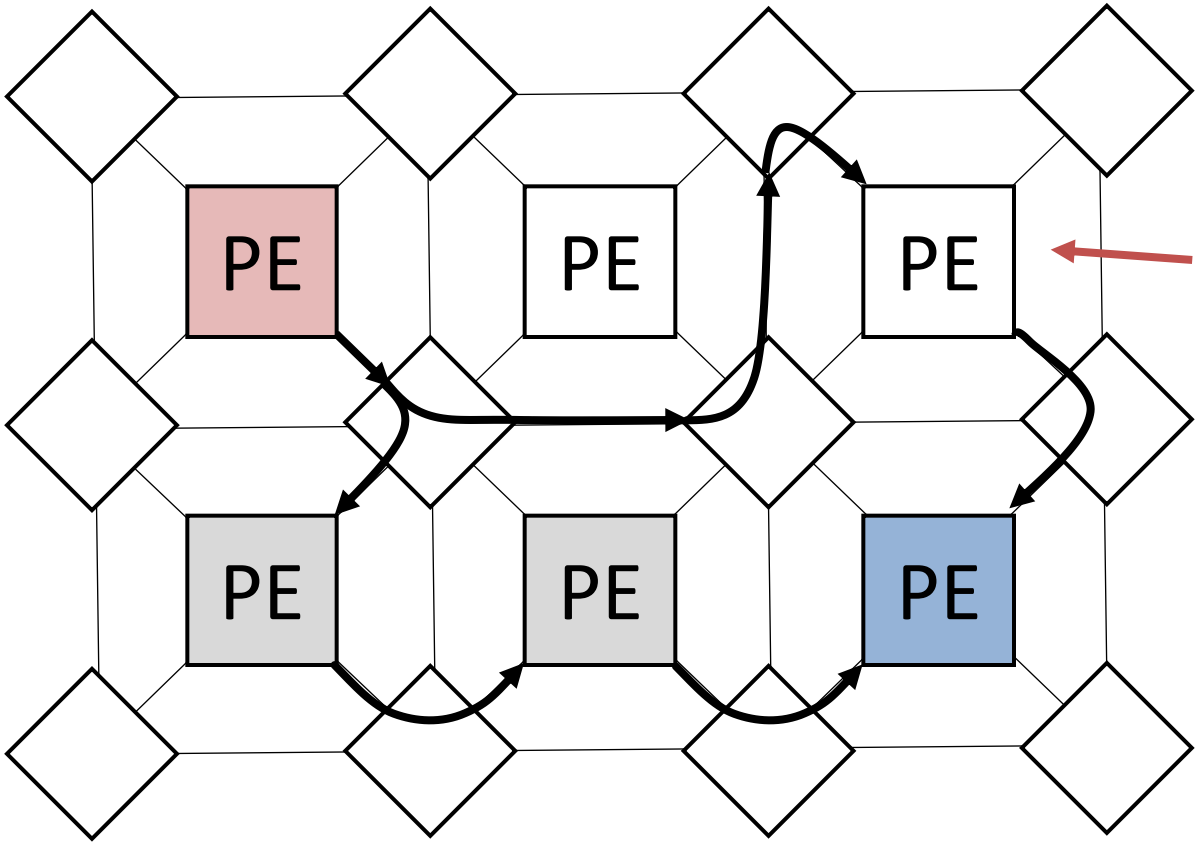
Alternatives to Delay-FIFOs (1)



Use a longer route: Costs **Routing** Resources

Alternatives to Delay-FIFOs (2)





This is also just good for improving routing bandwidth …

Pass through a PE: Costs **Mapping** Resources

“Systolic” CGRA Challenge Summary

- Systolic CGRA throughput is **very** sensitive to timing constraints
- Several ways to balance delays:
  - Pass-through **(affects mapping)**
  - Long route **(affects routing)**
  - Delay FIFOs **(affects timing)**
- Mapping / Routing / Timing responsibilities are highly interdependent
- Codesign: Either more delay FIFOs or more advanced scheduler

Outline

- Challenges for pipelining “Systolic” CGRAs
  - Throughput sensitivity and fractional initiation intervals.
  - Techniques for delay matching
- **Scheduling Approach 1: Phasing-Overlapping**
  - **Tractable Optimization through Overlapping**
- Scheduling Approach 2: Hybrid Scheduling
  - Heuristic to reduce search space
  - Optimization to deal with tricky cases
- Evaluation

Optimization Background

- Prior work demonstrates optimization-based spatial scheduling: Integer Linear Programming [PLDI 2013], SMT [TOPLAS 2014]
- Basic Integer Linear Programming Approach:
  - Decision Variables:
    - Assigning instructions to PEs (binary)
    - Assigning dependences to a set of Routes (binary)
    - Assigning delays to each PE (integer)
  - Linear Constraints:
    - Limit schedule to be legal
- We added three forms of delay-matching.

$$\forall_{ve \in G, n \in N} \quad \Sigma_{nl \in H} M_{el} = M_{vn} + P_{en}$$
$$\forall_{ev \in G, n \in N} \quad \Sigma_{ln \in H} M_{el} = M_{vn} + P_{en}$$

$$\forall_{e \in E} \quad X_e \leq \text{FIFO\_LEN}(1 + \Sigma_{n \in N} P_{en})$$

$$\forall_{e \in E, ev \in G} \quad mismatch >= T_v - T_e$$

Optimization Phases

- Joint Optimization: (50% Failure)
  - Timeout -- Very restricted hardware of systolic CGRA

- Separate Phases: (85% Failure)
  - Fast, but doesn't capture inter-dependence!

- In-between Joint: (75% Failure, poor throughput)

Our Approach: Phase Overlapping

- Insight: Phase Interdependence Matters
  - But relationship between adjacent phases is more relevant…

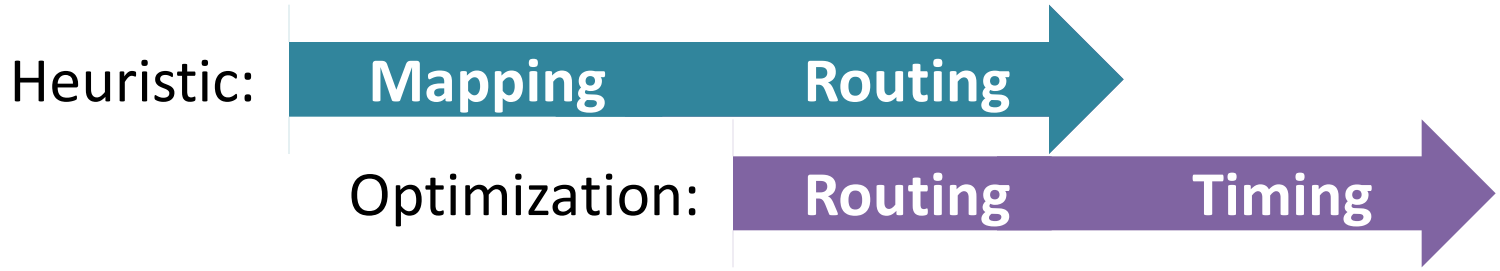
- Phase Overlapping:
  - Perform Mapping and Routing together
  - Discard Routing (keep Mapping)
  - Perform Routing and Timing together
- Good: Only 15% fail, mostly fully-pipelined
- Remaining Problems:
  - Mapping/routing phase takes time (90% or more)
  - Sometimes, we get an unlucky mapping/routing (looks like high potential, but cannot be fixed for timing)

Overview

- “Systolic” CGRAs
  - Throughput sensitivity and fractional initiation intervals.
  - Techniques for delay matching
- Scheduling Approach 1: Phasing-Overlapping
  - Tractable Optimization through Overlapping
- **Scheduling Approach 2: Hybrid Scheduling**
  - **Heuristic to reduce search space**
  - **Optimization to deal with tricky cases**
- Evaluation

Hybrid Scheduling: Integrate Heuristic

- Insight: Mitigate the problems overlapped scheduling by trying things repeatedly if we got an unlucky mapping.
- But … Optimization won't work for mapping/routing
  - Still too slow (and do we really need it?)
  - ILP solver's don't tend to generate unique solutions

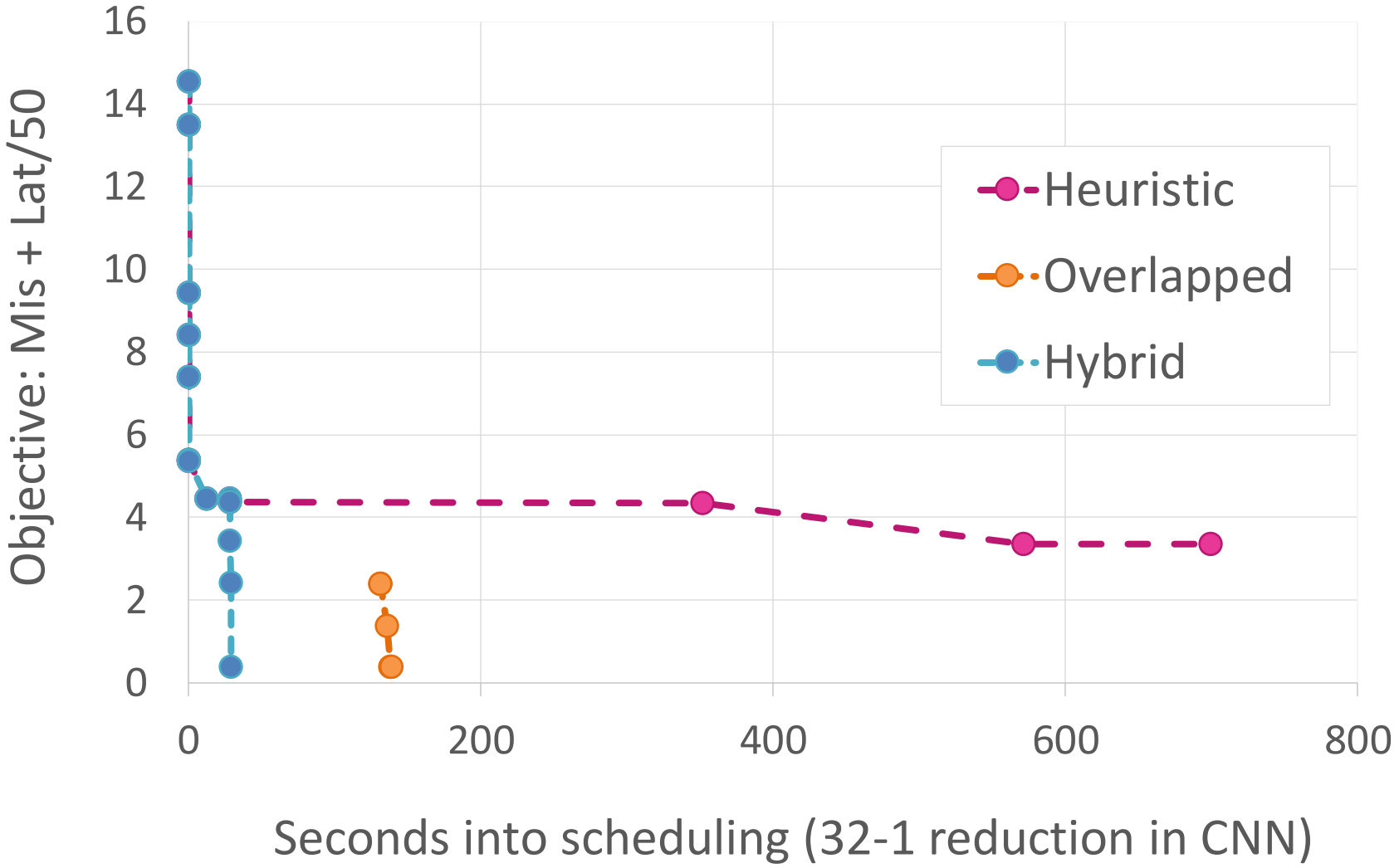


- Hybrid Approach: Use a heuristic for the mapping/routing phase, to quickly get a plausible mapping, then optimize the routing and timing together.

A Heuristic for Hybrid Scheduling

- Need a heuristic that **quickly** generates **unique** schedules!
- Basic Approach: Stochastic Scheduling
  - Iteratively build schedule in topological order **(fast)**
  - Normally be rational: Choose a location for each instruction which minimizing routing resources and timing mismatch.
  - Occasionally be irrational: Choose a purposely bad mapping, hoping that it might help later on **(unique)**
  - Repeat a few times to find a good schedule **(good)**
- Objective function: Minimize total mismatch (to make routing/timing scheduling easier)

Intuition for Solution Quality

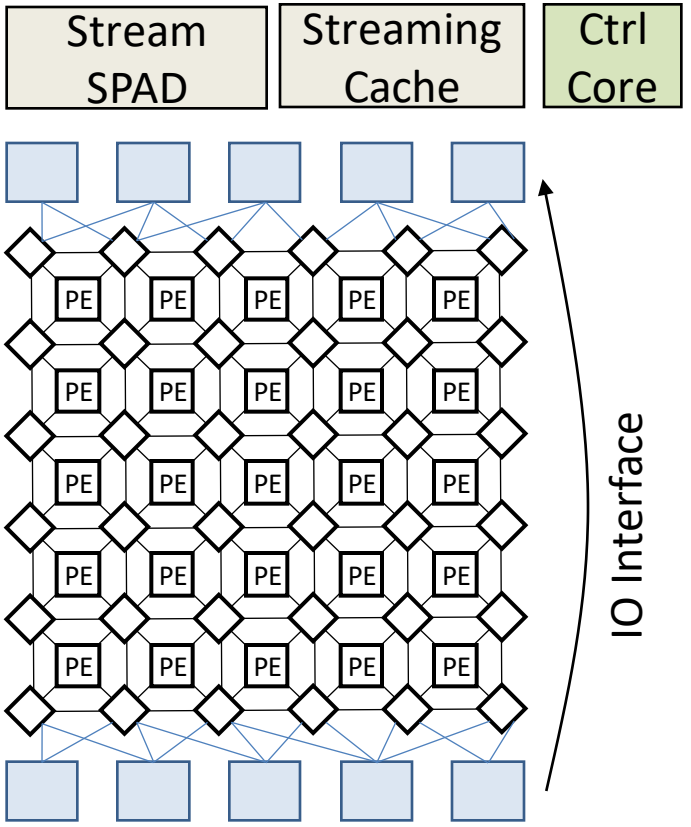


Outline

- “Systolic” CGRAs
  - Throughput sensitivity and fractional initiation intervals.
  - Techniques for delay matching
- Scheduling Approach 1: Phasing-Overlapping
  - Tractable Optimization through Overlapping
- Scheduling Approach 2: Hybrid Scheduling
  - Heuristic to reduce search space
  - Optimization to deal with tricky cases
- **Evaluation**

Methodology – Accelerator Modeling

- Softbrain Accelerator
  - 5x5 Processing Elem (PE)
  - 64-bit datapath (subword)
  - Heterogeneous Grid
  - All FUs are fully pipelined
- Simulation: All blocks simulated at cycle-level in gem5, single “core”
- Area Analysis
  - Synthesize Chisel-based design in 55nm tech library.
  - Assumes Control core (single-issue inorder 16KB I$/D$) + 4KB SPAD
- Workloads (accelerator centric)
  - MachSuite, CNN/DNN, Dense Linear Algebra QR/Cholesky/FFT)



Softbrain [ISCA 2017]

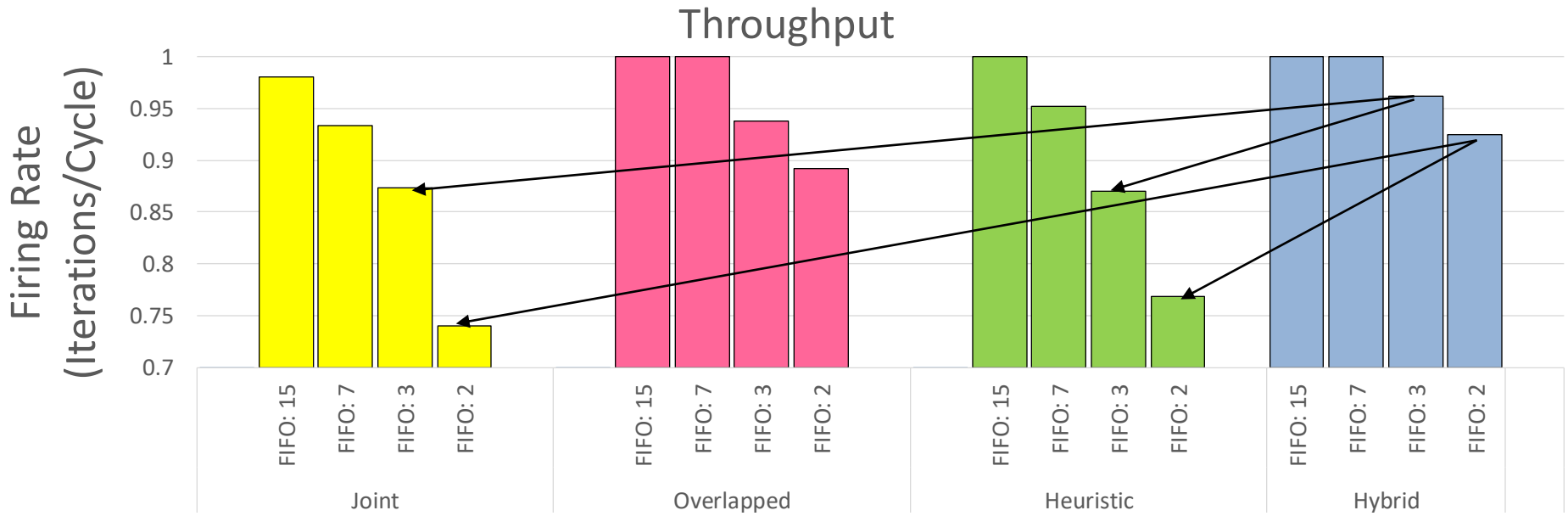
Does delay FIFO area matter?

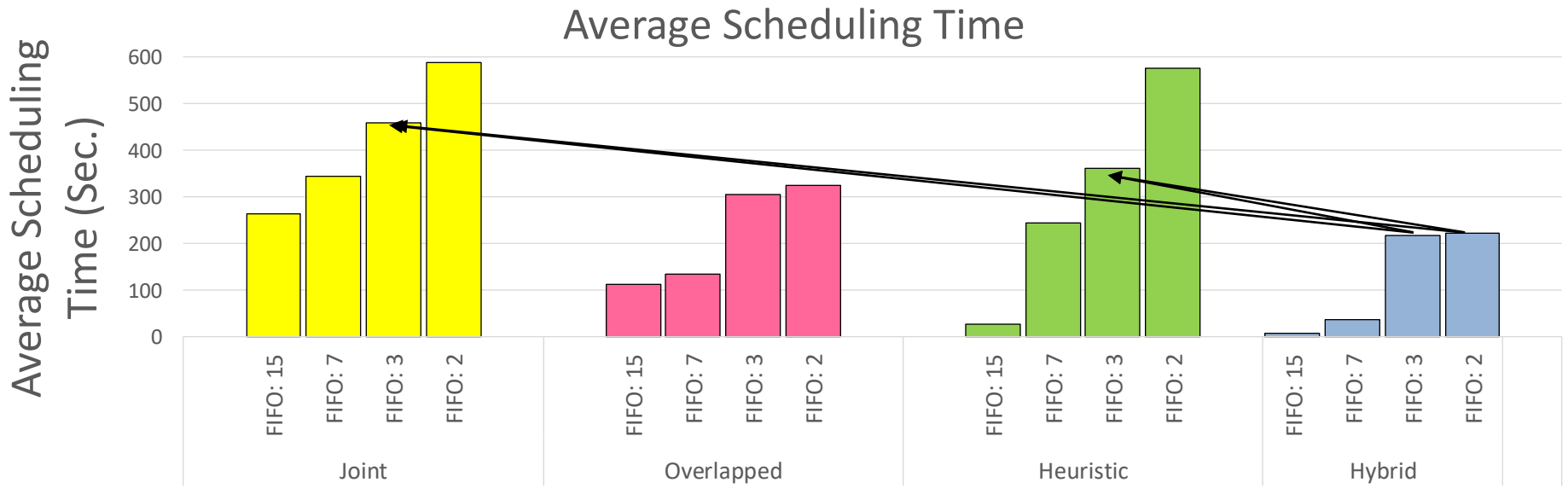
FIFO Length	Scheduling Difficulty	Area (mm^2)	Overhead
2	Very Hard	0.528	9.67%
3	Hard	0.543	12.90%
7	Medium	0.606	25.85%
15	Easy	0.736	52.86%

Methodology – Experimental Setup

- Scheduler Designs:
  - **Joint** Optimization
  - **Overlapped** Optimization
  - **Heuristic** Only
  - **Hybrid** (overlap/part heuristic)
- Scheduling Timeout: 20 Minutes
- Problem: How to compare schedulers that across a set of workloads that may fail?
  - We don't! Instead, we seed all schedulers with an initial solution from the heuristic.
  - Joint and Overlapped are also hybrid schedulers in the evaluation, just using a more traditional approach.

Performance vs Time vs Complexity



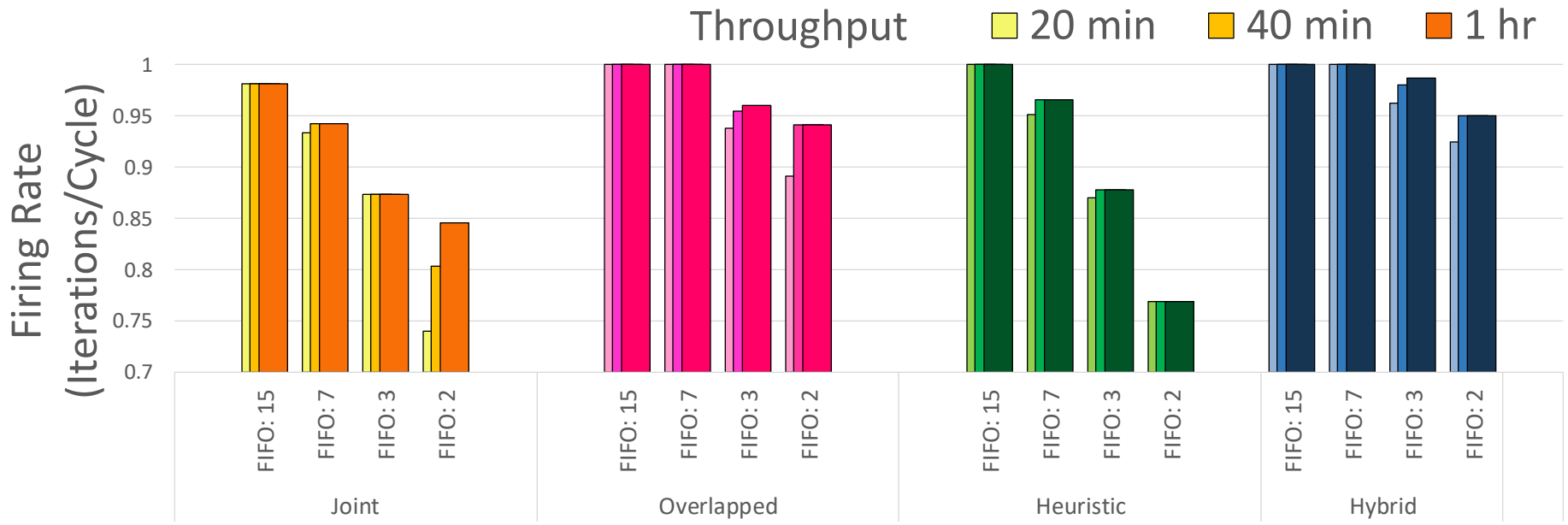


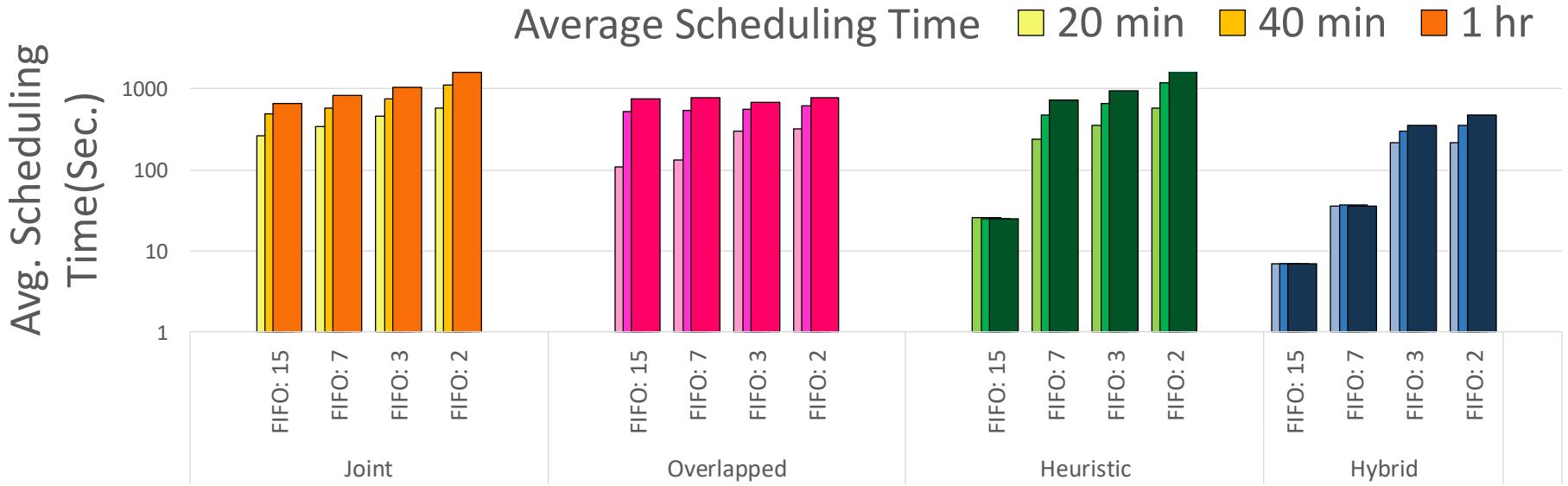
Conclusions

- Systolic CGRAs gain efficiency by simplifying the execution model.
- Consequence is tradeoff between easy scheduling low hardware overhead, and high performance:
  - Minimizing delay-FIFO size is key factor in reducing area and increasing scheduler difficulty in finding minimum II.
- Two novel techniques:
  - Phase Overlapping (capture the phase interdependence)
  - Hybrid Scheduling (using heuristic to generate solutions)
- Broadly, codesign is the key to success of future reconfigurable architectures.

Thank you

Scheduling-time Sensitivity





Modeled vs *Measured* Throughput