DSAGEN: Synthesizing Programmable Spatial Accelerators

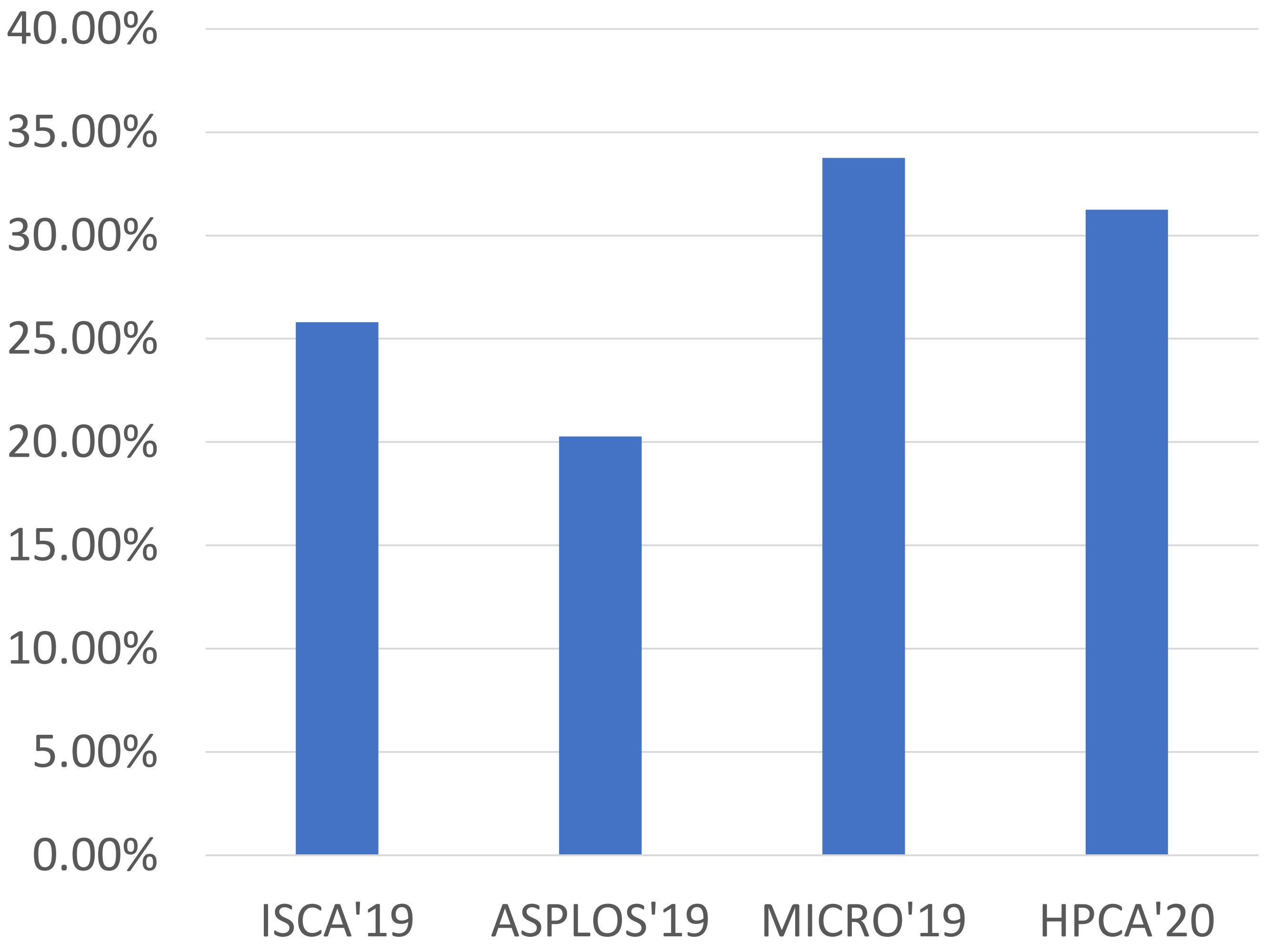
Jian Weng, Sihao Liu, Vidushi Dadu, Zhengrong Wang, Preyas Shah, Tony Nowatzki

University of California, Los Angeles

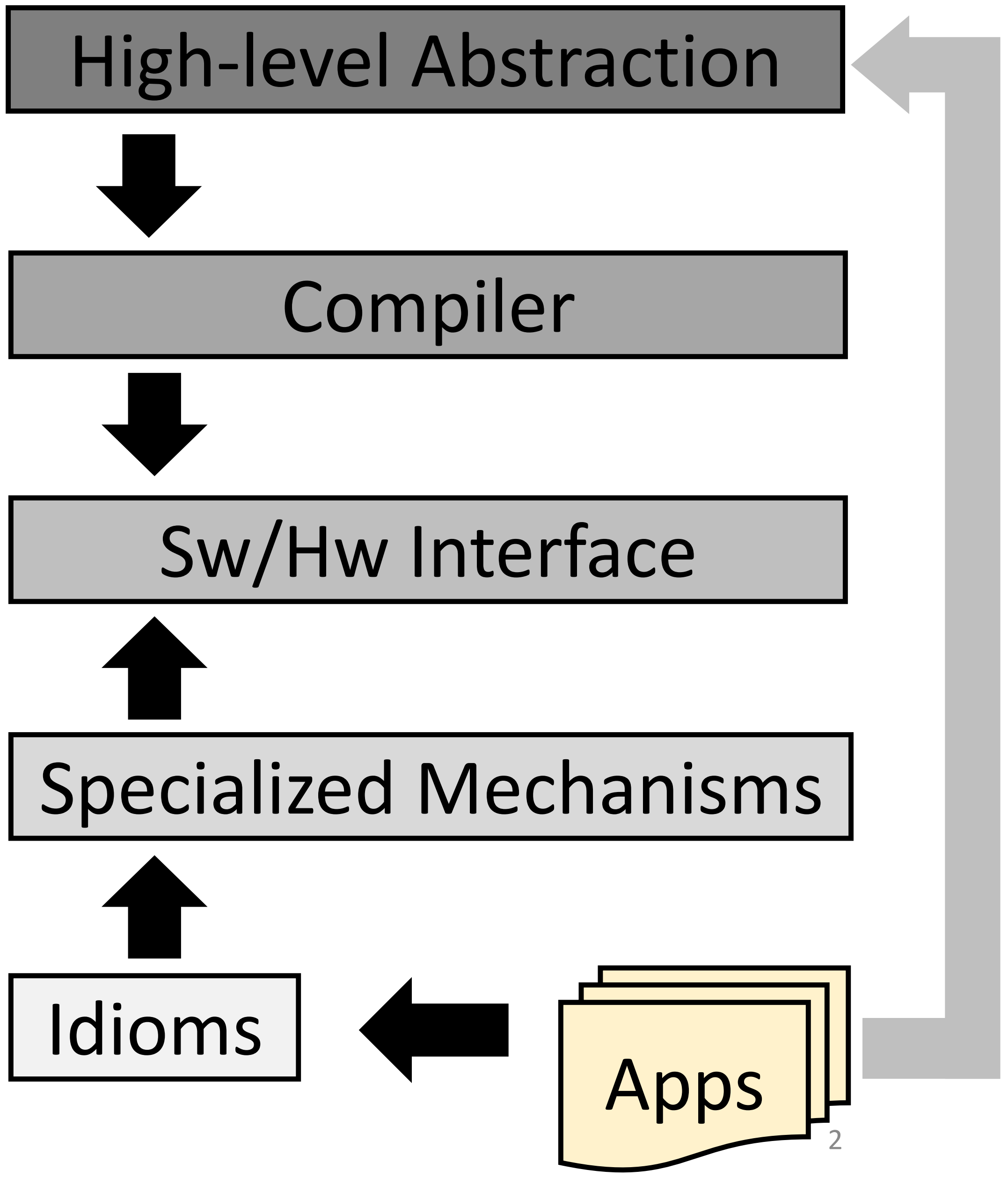
May 15th , 2020

Specialized Accelerators

Specialized architecture often occupies 1/5~1/3 of publications in top conferences.

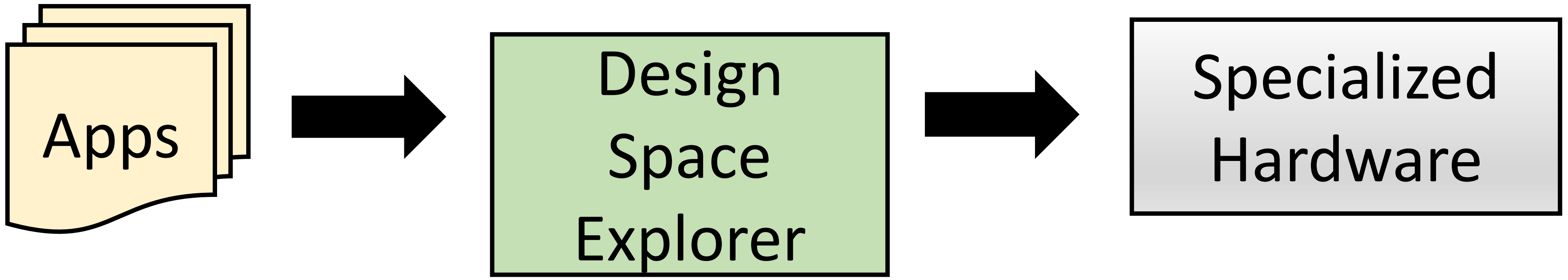


Existing Domain-Specific Approach:



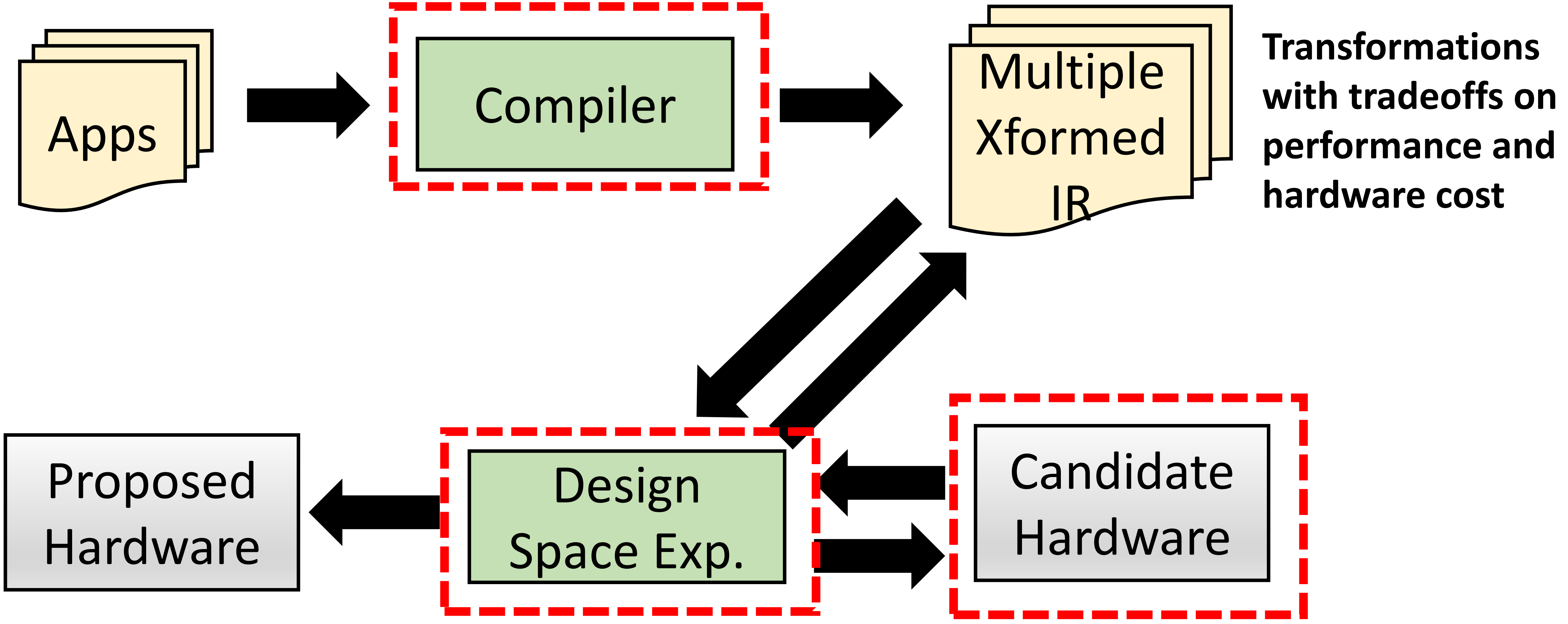
DSAGEN: Decoupled Spatial Accelerator Generator

Domain Specific



DSAGEN: Decoupled Spatial Accelerator Generator

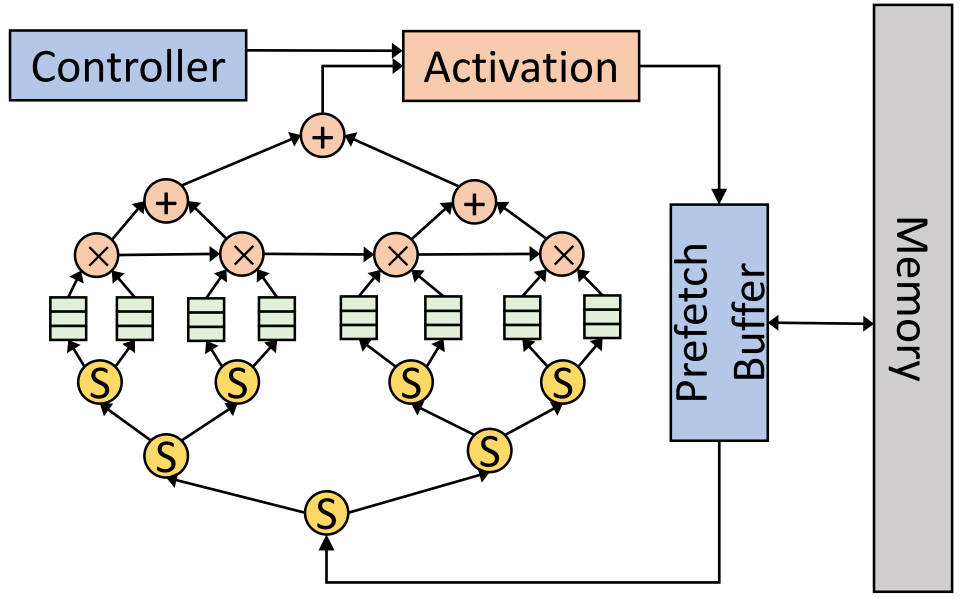
Domain Specific



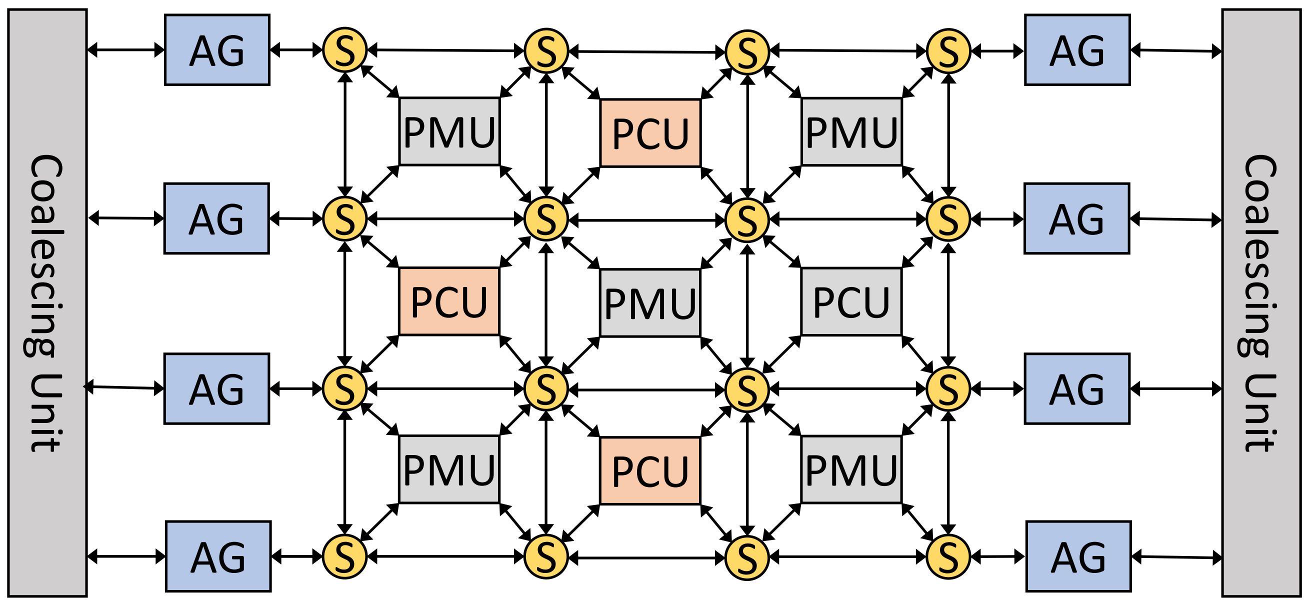
Transformations with tradeoffs on performance and hardware cost

Outline

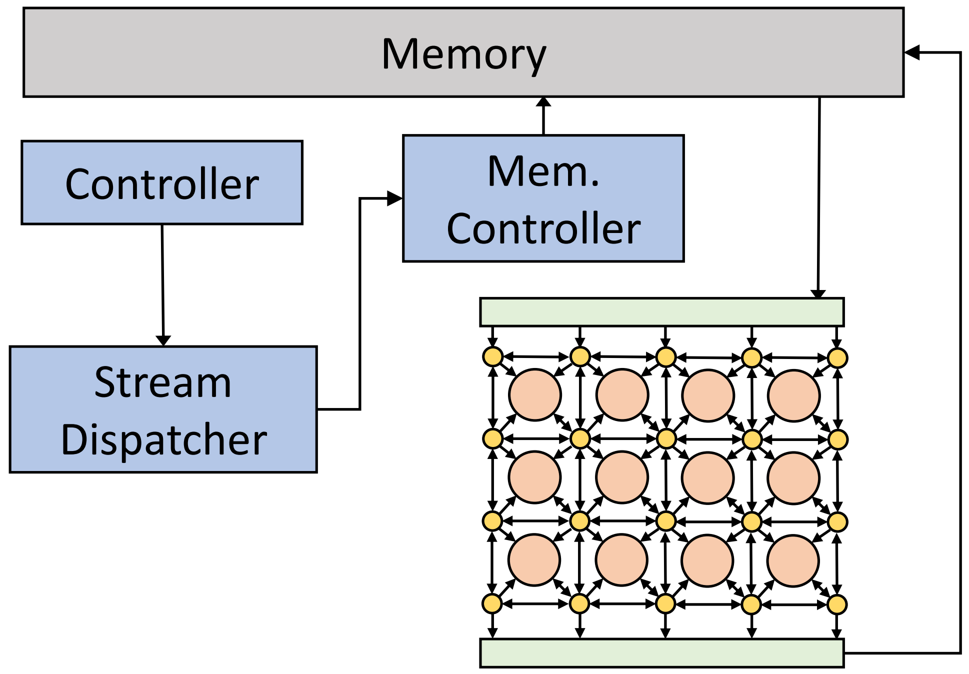
- **Design Space — Decoupled-Spatial Architecture**
  - **Insight from Prior Work**
  - **The Programming Paradigm**
  - **Design Space: Hardware Primitives (& Composition)**
- **Compilation**
- **Design Space Exploration**
- **Evaluation**



(a) ASPLOS18-MAERI



(b) ISCA17-Plasticine



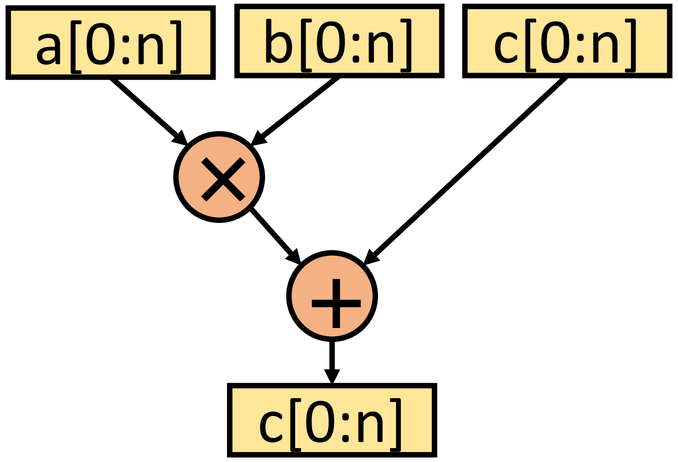
(c) ISCA17-Softbrain

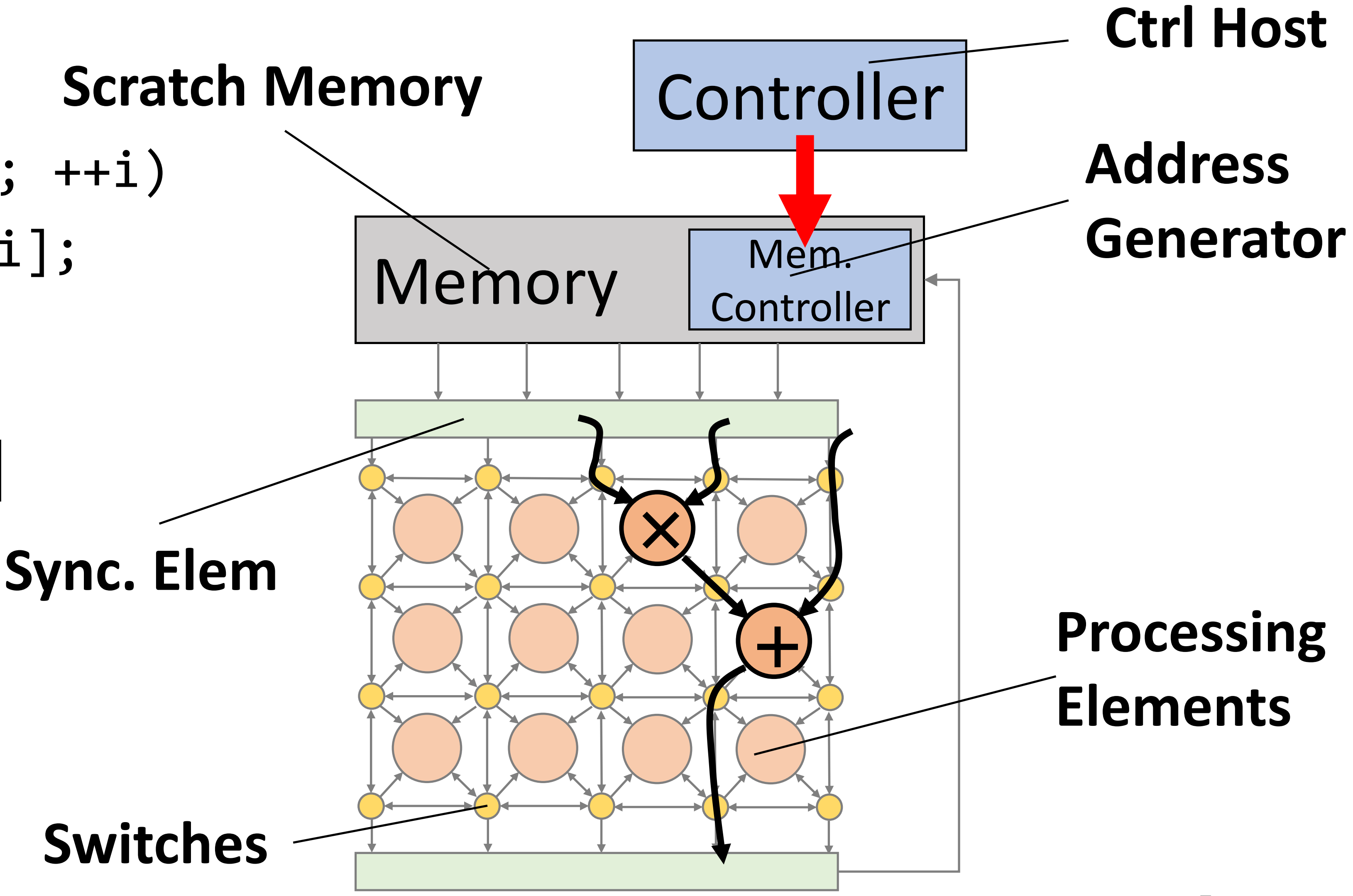
- Decoupled-Spatial Paradigm
  - Decoupled Compute/Memory
  - Spatially exposed resources
- Design Space
  - Composing hardware with simple primitives
  - Architecture Description Graph

Func. Unit	🟧
Memory	⬜
Switch	🟨
Control	🟦
Sync. Elem.	🟩

Background: Decoupled-Spatial Architecture

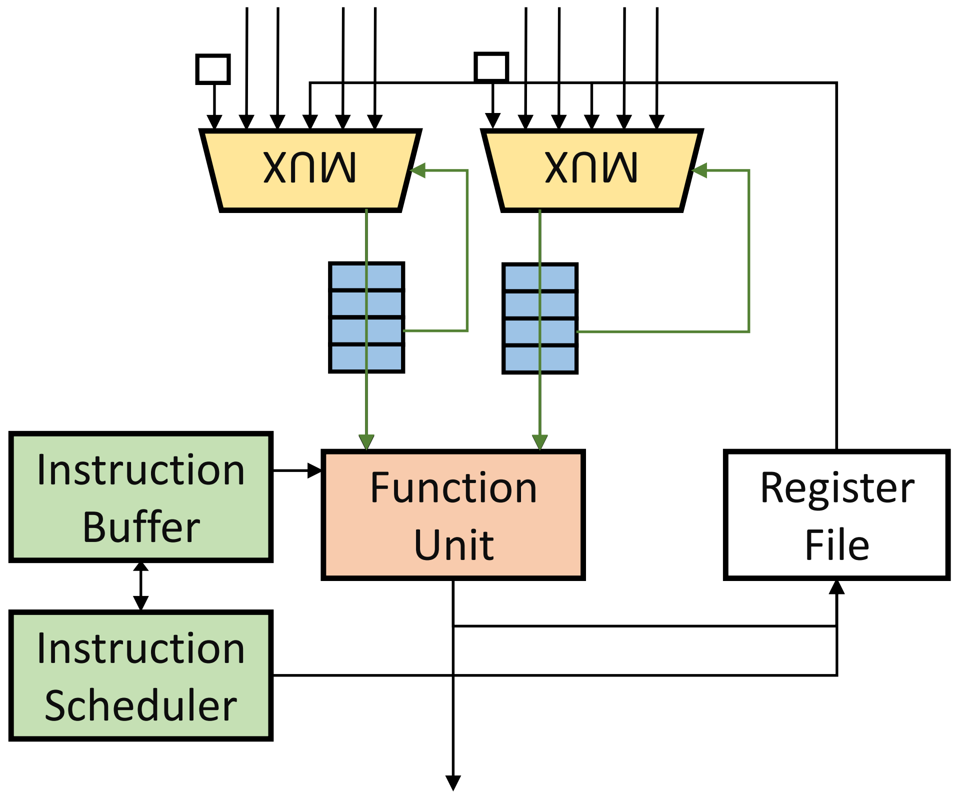
```
for (int i = 0; i < n; ++i)
    c[i] += a[i] * b[i];
```





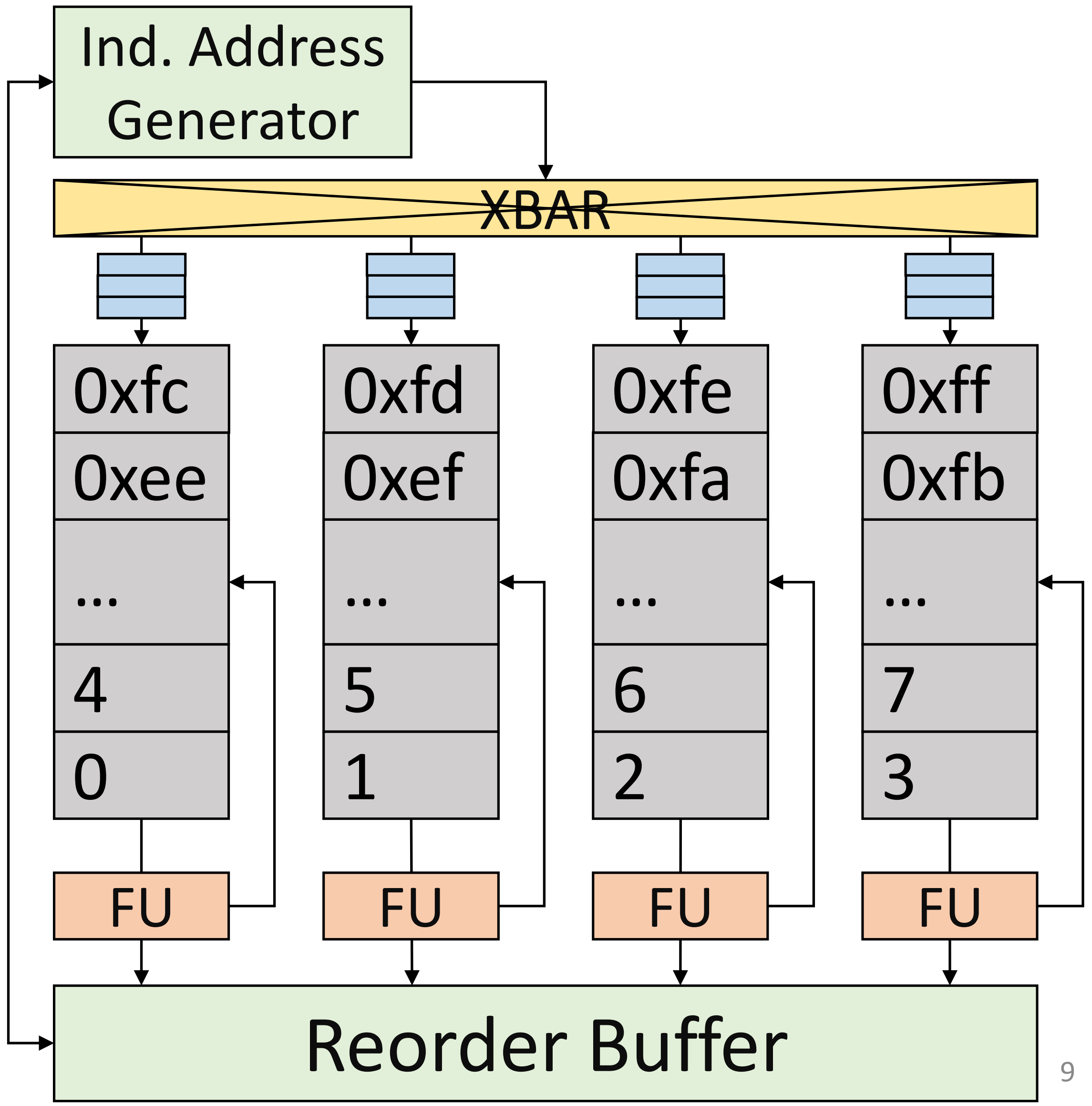
Hardware Primitives: Processing Element & Switch

Hardware Cost: Low	Dedicated (=1)	Shared (>1) High
Statically Scheduled	**“Systolic” 1x Area** + No contention - Harder to map - Higher power *Softbrain	**“CGRA” 2.6x Area** + Better resource utilization - Harder to map *Conventional CGRA
Dynamically Scheduled High	**“Ordered Dataflow” 2.1x Area** + Better flexibility *SPU	**“Tagged Dataflow” 5.8x Area** + Better flexibility + Better resource utilization *Triggered Instruction

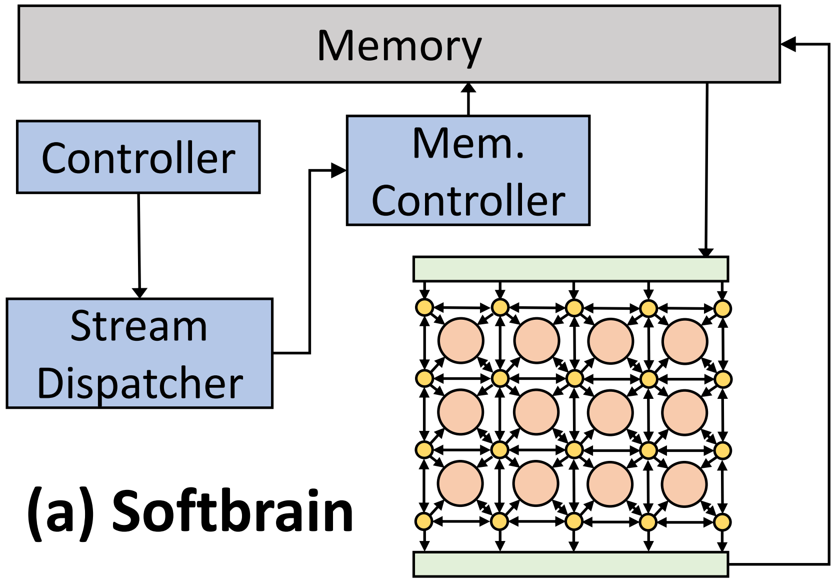


Hardware Primitives: Memory

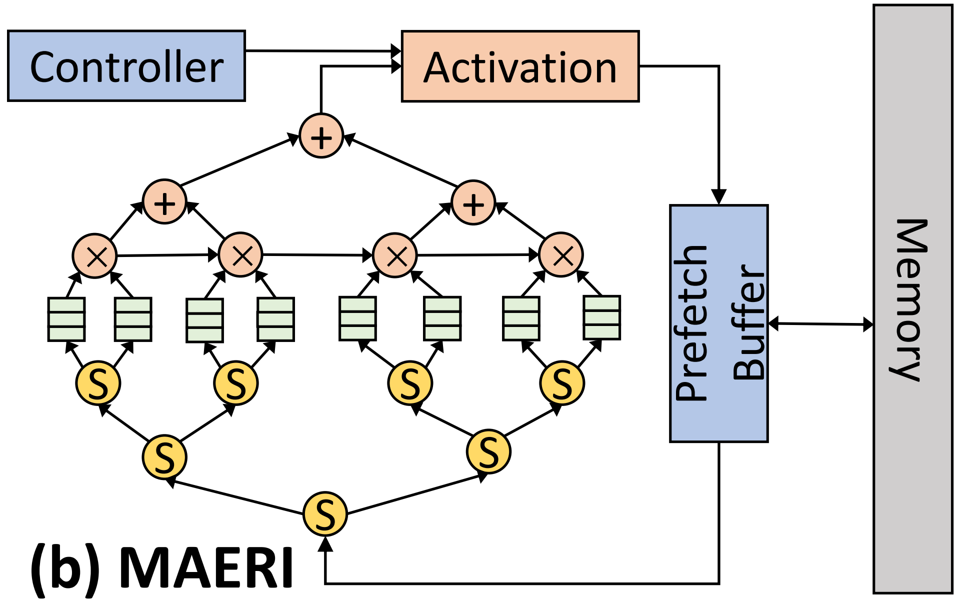
- Memory
  - Size
  - Bandwidth
  - Indirect Support
    - a[b[i]]
  - Atomic Update
    - a[b[i]] += 1



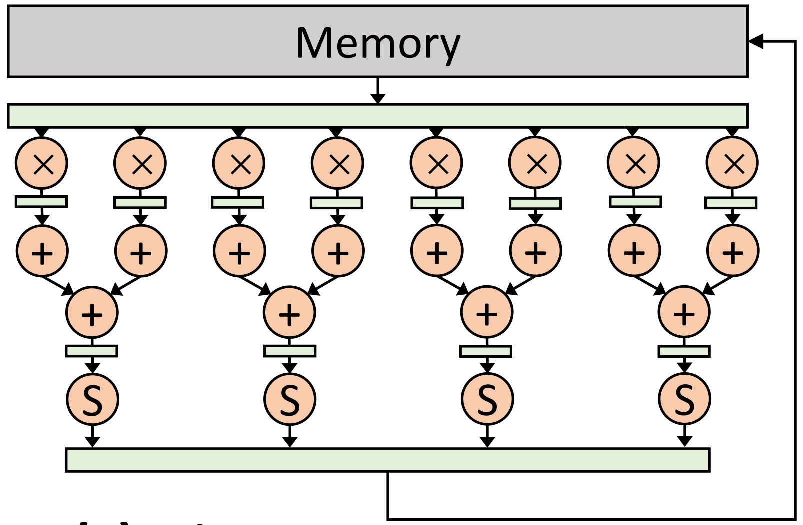
Examples of ADG



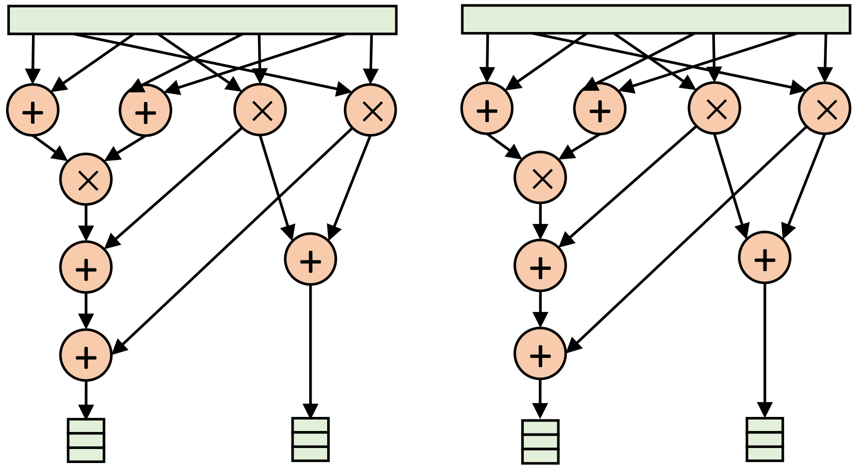
(a) Softbrain



(b) MAERI



(c) Diannao

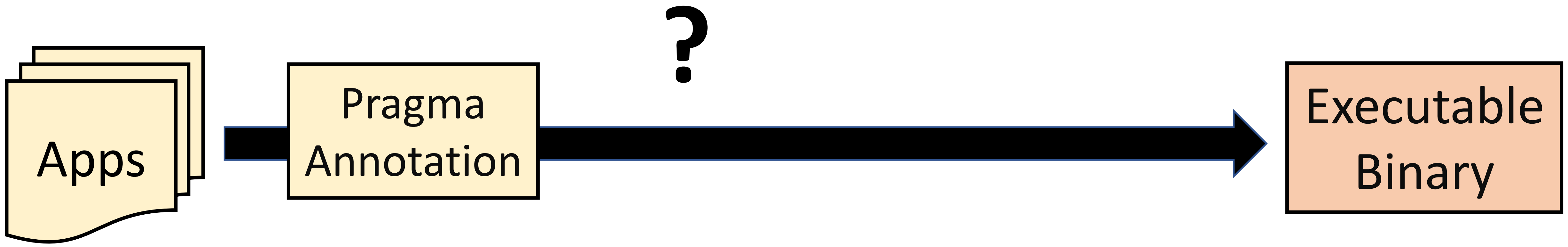


(d) Data Path of Complex Mul.

Outline

- Decoupled-Spatial Architecture
- **Compilation**
  - **High-Level Abstraction**
  - **Hardware-Aware Modular Compilation**
- Design Space Exploration
- Evaluation

Compiling High-Level Lang. to Decoupled Spatial

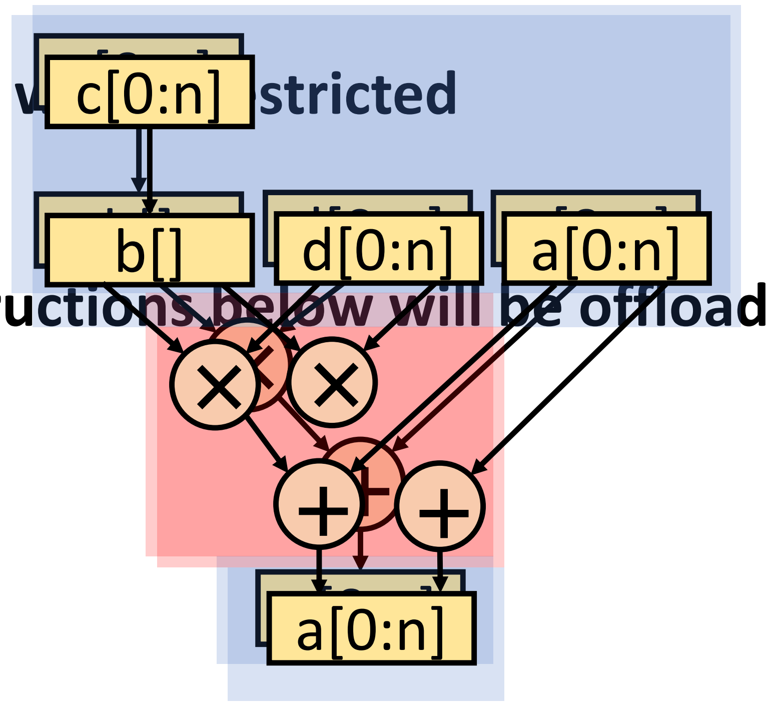


How to abstract diverse underlying features with a unified high-level interface?

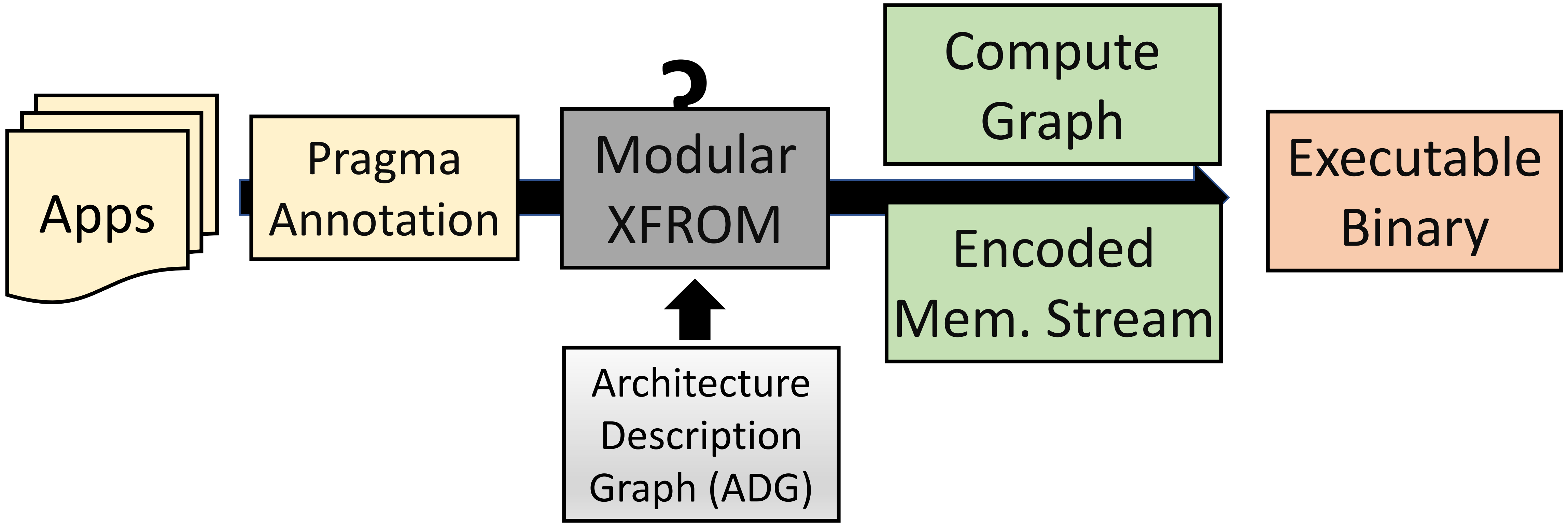
- Programmer Hints
  - Which code regions are offloaded onto the spatial accelerator.
  - Which memory accesses can be decoupled intrinsics.
  - Which offloaded regions should be concurrent.

An example of pragma annotation

```
#pragma config ← The offloaded region in this compound body are concurrent
{
  #pragma stream ← The memory accesses below w stricted
  for (i=0; i<n; ++i)
    #pragma offload ← The computational instructions below will be offloaded
    for (j=0; j<n; ++j)
      a[i*n+j] += b[c[j]] * d[i*n+j];
}
```



Compiling High-Level Lang. to Decoupled Spatial

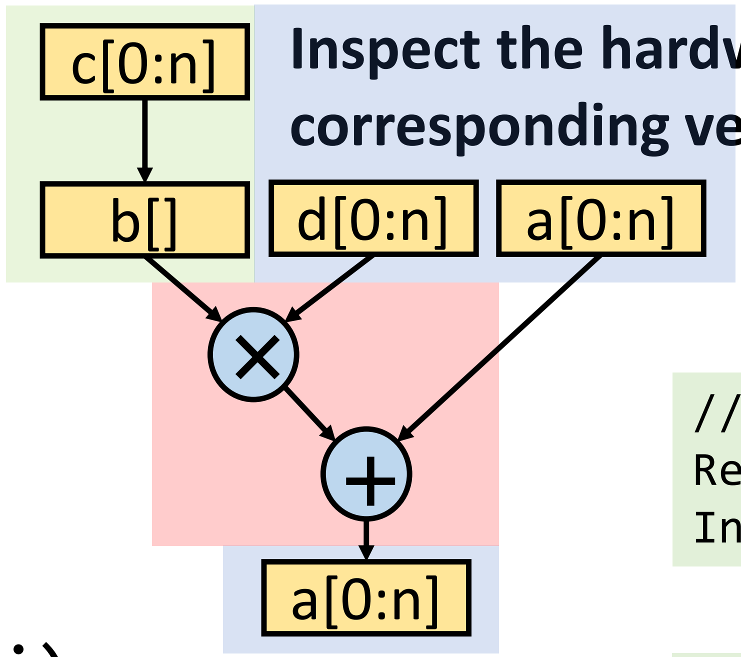


How to hide the diversity of underlying hardware?

- Modular Transformation
  - Specialized Hardware features often dictate the code transformation
  - A fallback is required when the hardware feature is not available

Modular Transformation

```
#pragma config
{
  #pragma stream
  for (i=0; i<n; ++i)
    #pragma offload
    for (j=0; j<n; ++j)
      a[i*n+j] += b[c[j]] * d[i*n+j];
}
```

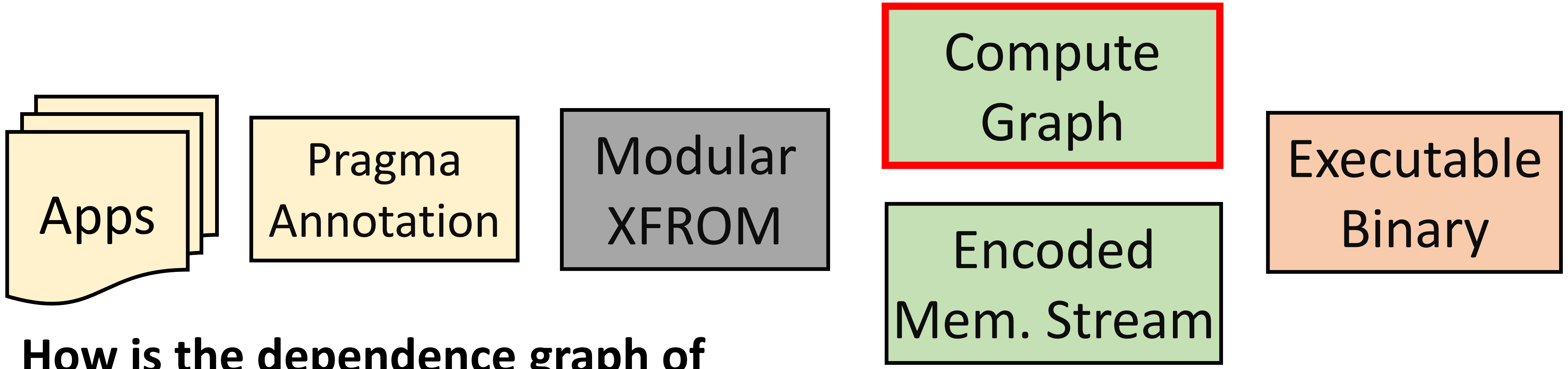


Inspect the hardware features to generate corresponding version of indirect memory

```
// With indirect support
Read c[0:n], stream0
Indirect b, stream0, stream1
```

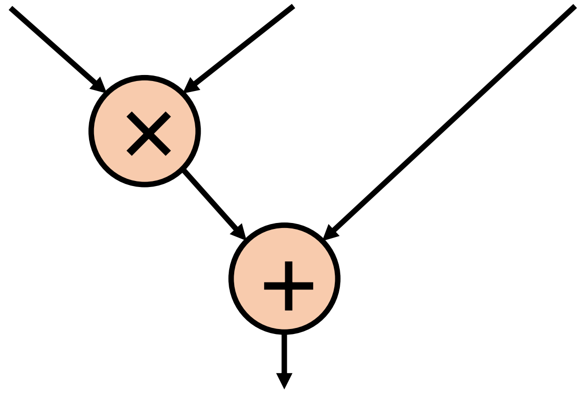
```
// Without indirect support
for (j=0; j<n; ++j)
   Scalar b[c[j]], stream0
```

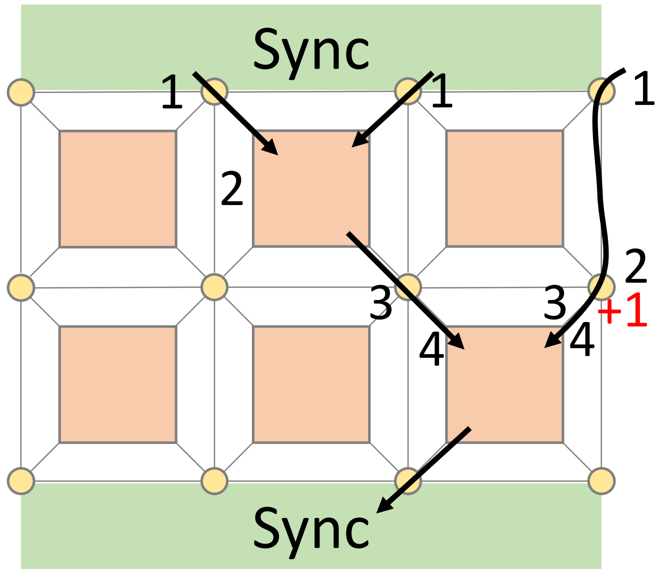
Compiling High-Level Lang. to Decoupled Spatial



How is the dependence graph of computational instructions mapped?

Spatial Mapping





How is the dependence graph of computational instructions mapped?

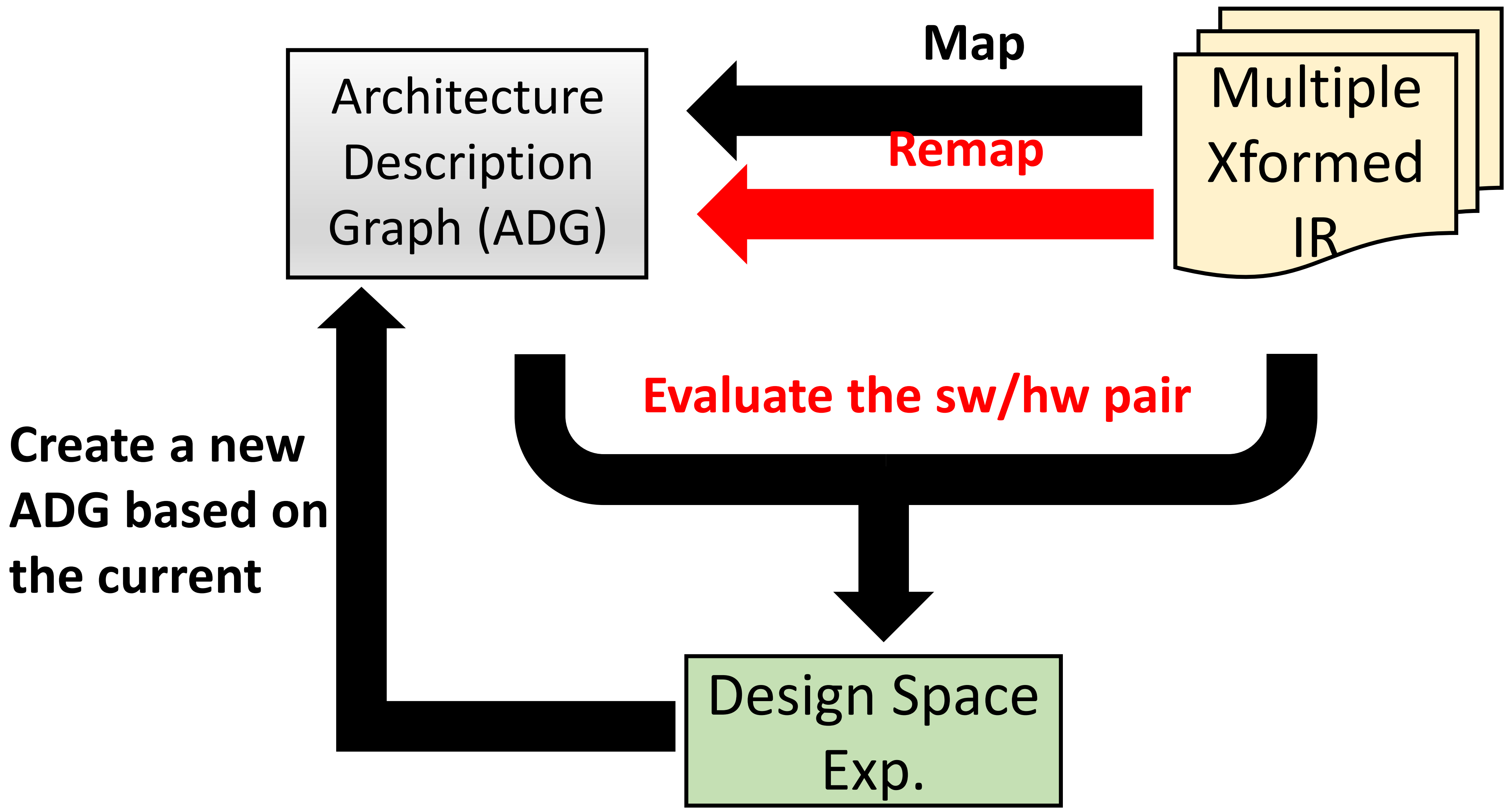
1. Placement: Map instruction to PE's with corresponding capability.
2. Routing: Routing the dependence edges thru the spatial network.
3. Timing: If necessary, balance the timing of data arrival

- If one of 1-3 is not successful, revert some nodes and repeat 123

Outline

- Decoupled-Spatial Architecture
- Compilation
- **Design Space Exploration**
  - **Drive the Search**
  - **Evaluating Design Points**
  - **Repairing the Mapping**
- Evaluation

Design Space Exploration


Map
Architecture Description Graph (ADG)
Remap
Multiple Xformed IR
Evaluate the sw/hw pair
Create a new ADG based on the current
Design Space Exp.

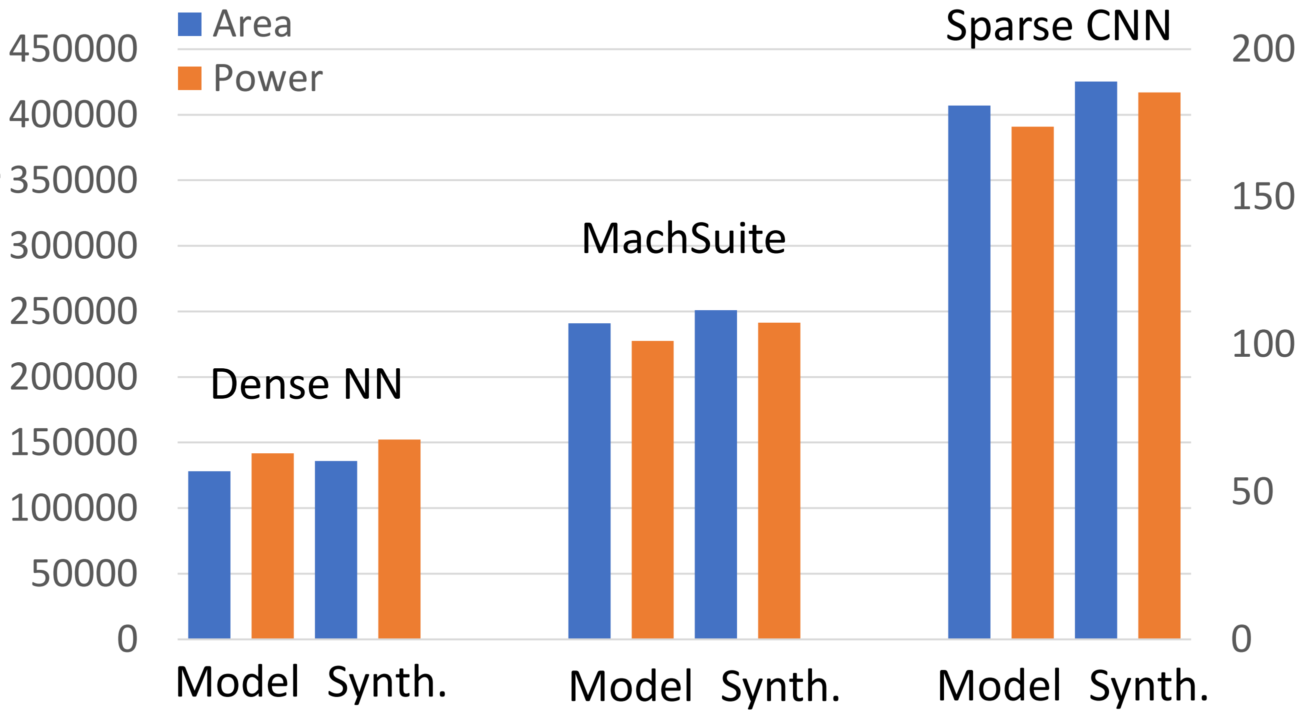
Estimation Model

- Performance
    - Spatial architecture essentially enables hardware specialized sw-pipelining
    - The ratio of data availability determines the performance
    - Perf=#Inst * (Activity Ratio)

The model has mean performance error of 7%, and with maximum error 30%.

- Power/Area
    - Synthesis can be time consuming
    - A regression model can predict the trend of hardware cost

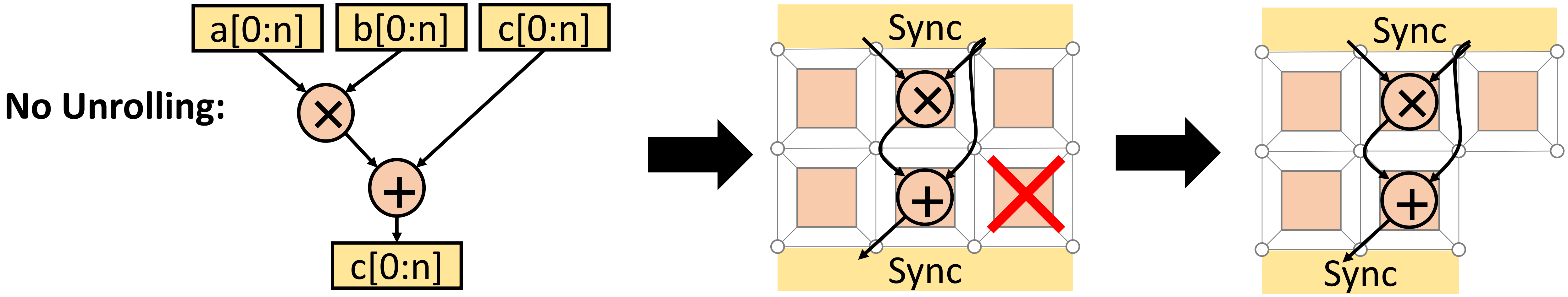
Model Validation



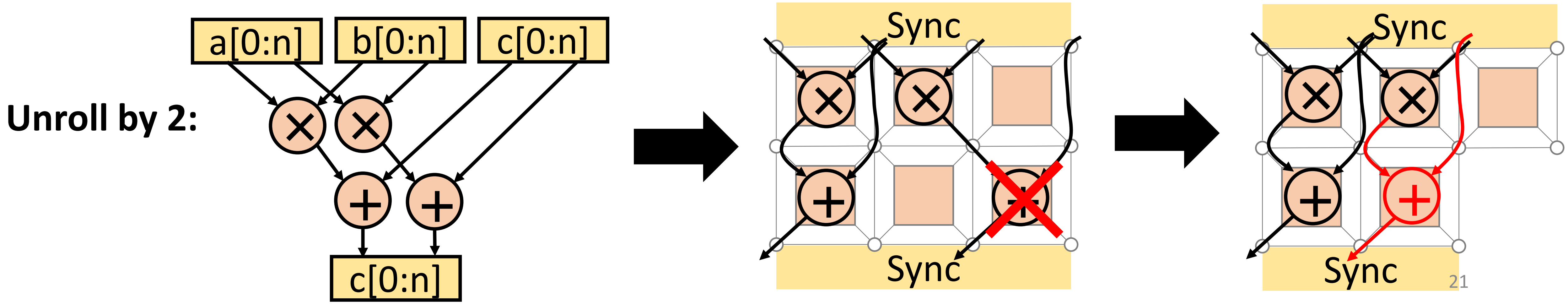
Repairing the Spatial Mapping

```
// Original Code
for (i=0; i<n; ++i)
  c[i]+=a[i]*b[i];
```

No Unrolling:



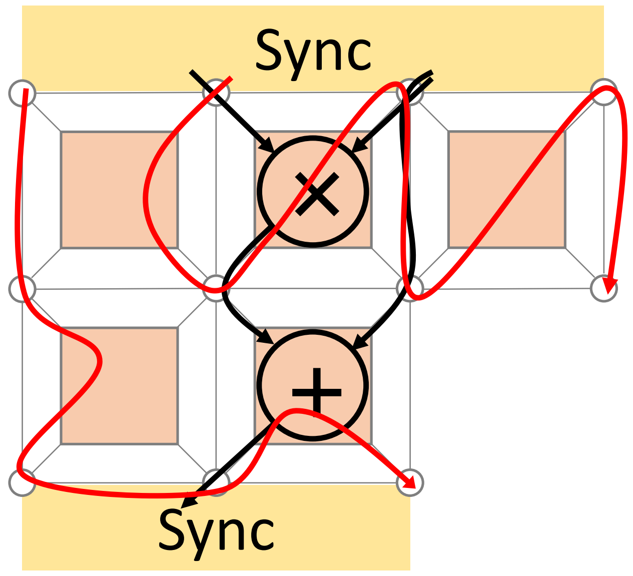
Unroll by 2:



Hardware/Software Interface Generation

- How to configure accelerator with arbitrary topology?
  - Reuse the data path for configuration
  - Find path(s) that cover(s) all the components
  - A heuristic based heuristic algorithm to minimize the longest path of configuration

- For a graph with m nodes covered by n paths, the longest path cannot be shorter than $\lceil \frac{m}{n} \rceil$.
- We only introduces 40% overhead over the ideal bound.



Outline

- **Decoupled-Spatial Architecture**
- **Compilation**
- **Design Space Exploration**
- **Evaluation**
  - **Methodology**
  - **Compiler**
  - **Design Space Exploration**

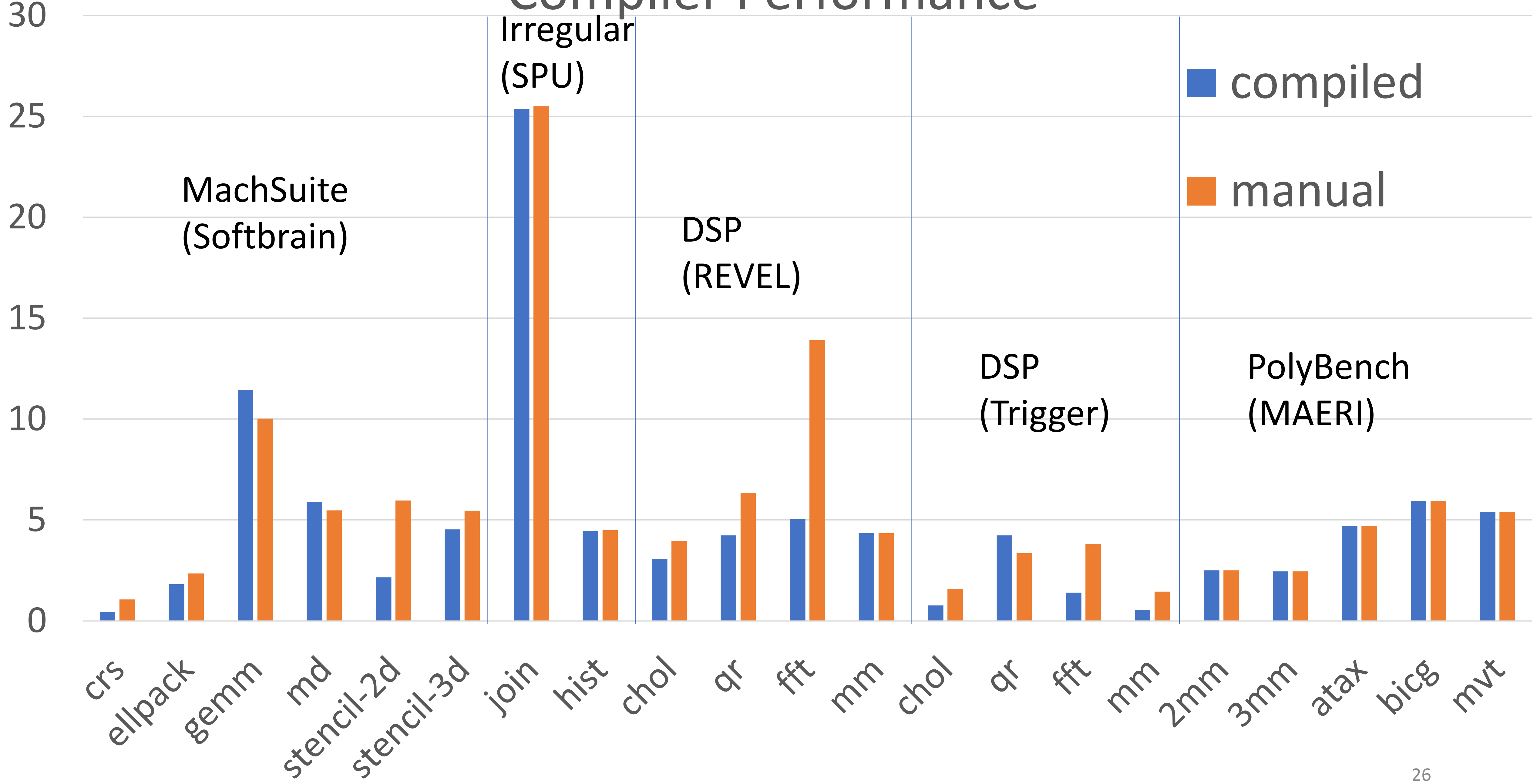
Methodology

- Performance
  - Gem5 RISCV in-order core integrated with a cycle-accurate spatial accelerator simulator
    - The in-order core is extended with stream decoupled ISA
- Power/Area
  - All the components are implemented in Chisel RTL
  - Synthesized in Synopsys DC 28nm @1.00GHz
  - SRAM power/area are estimated by CACTI 7.0

Compiler Performance

- Softbrain — MachSuite
  - Versatile accelerator can handle moderate irregularity
- SPU — Histogram, and Key Join
  - Accelerator specialized for irregular workloads
- REVEL and Trigger — DSP
  - Accelerator specialized for imperfect loop body
- MAERI — PolyBench
  - Accelerator for neural network

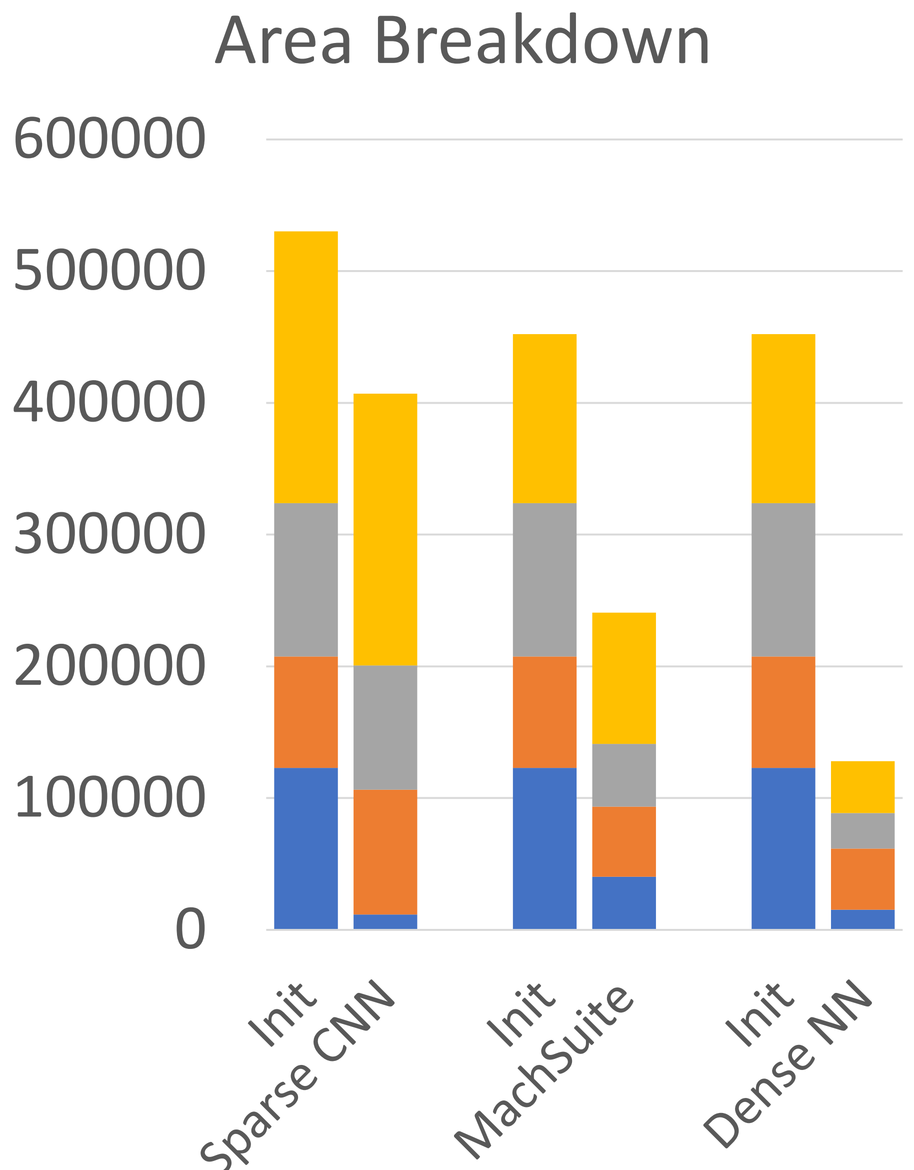
Compiler Performance

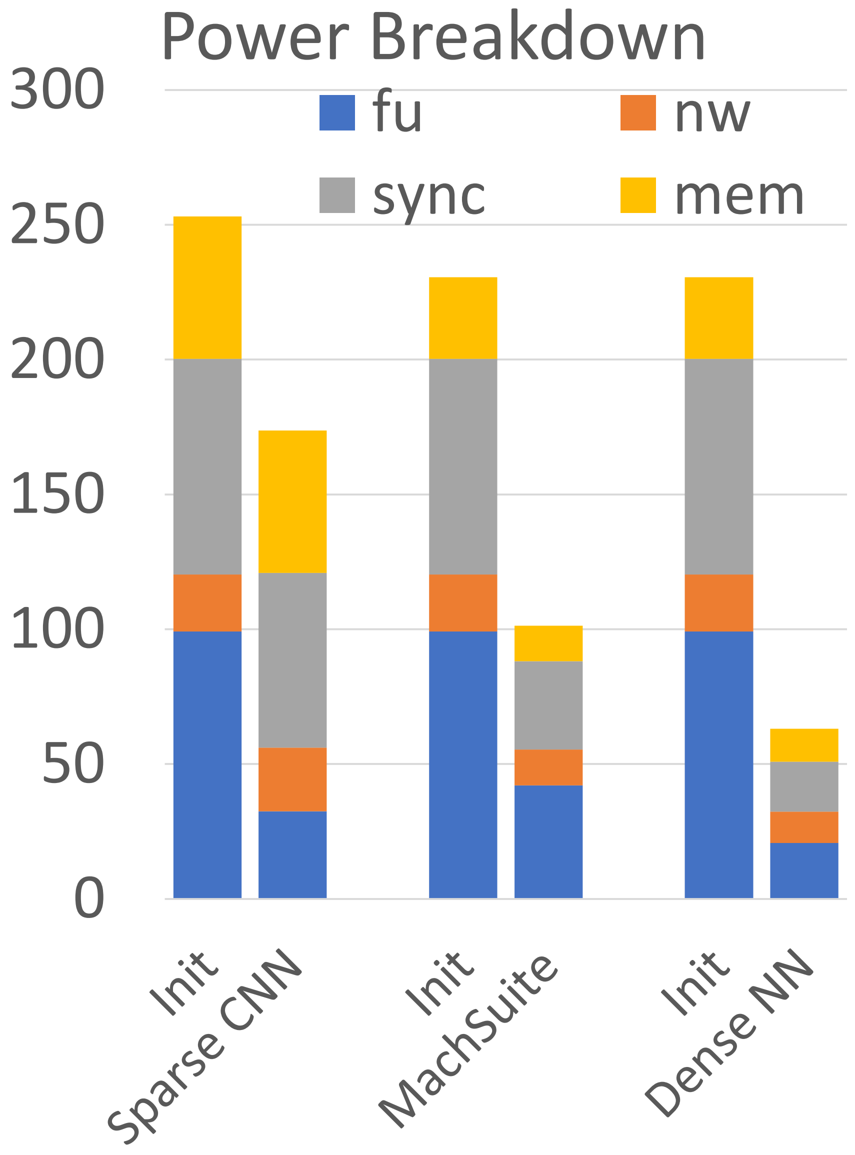


Design Space Explorer

- Workloads
  - Dense Neural Network
  - MachSuite
  - Sparse Convolutional Neural Network
- Initial Design
  - A 5x5 mesh with all capability (arithmetic, control, and indirect)
- Objective: $perf^2/mm^2$

Design Space Explorer





Sparse CNN: 24h
MachSuite: 19.2h
Dense NN: 16h

Conclusion

	HLS	**Manual**	**DSAGEN**
Frontend	C+Pragma	DSL/Intrinsics, etc.	C+Pragma
Design Flow	Nearly Automated	Manual	Nearly Automated
Input	A **Single** Application	Multiple Target Applications	**Multiple** Target Applications
Output	**Application-Specific** Accel.	ASIC/Programmable Accel.	A **Programmable** Accelerator
Design Space	**Limited**	**Rich**	**Rich**

Q&A

- Our framework is working in progress at: https://github.com/PolyArch/dsa-framework
- All the questions and comments are welcomed