A Hybrid Systolic-Dataflow Architecture for Inductive Matrix Algorithms

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

University of California, Los Angeles

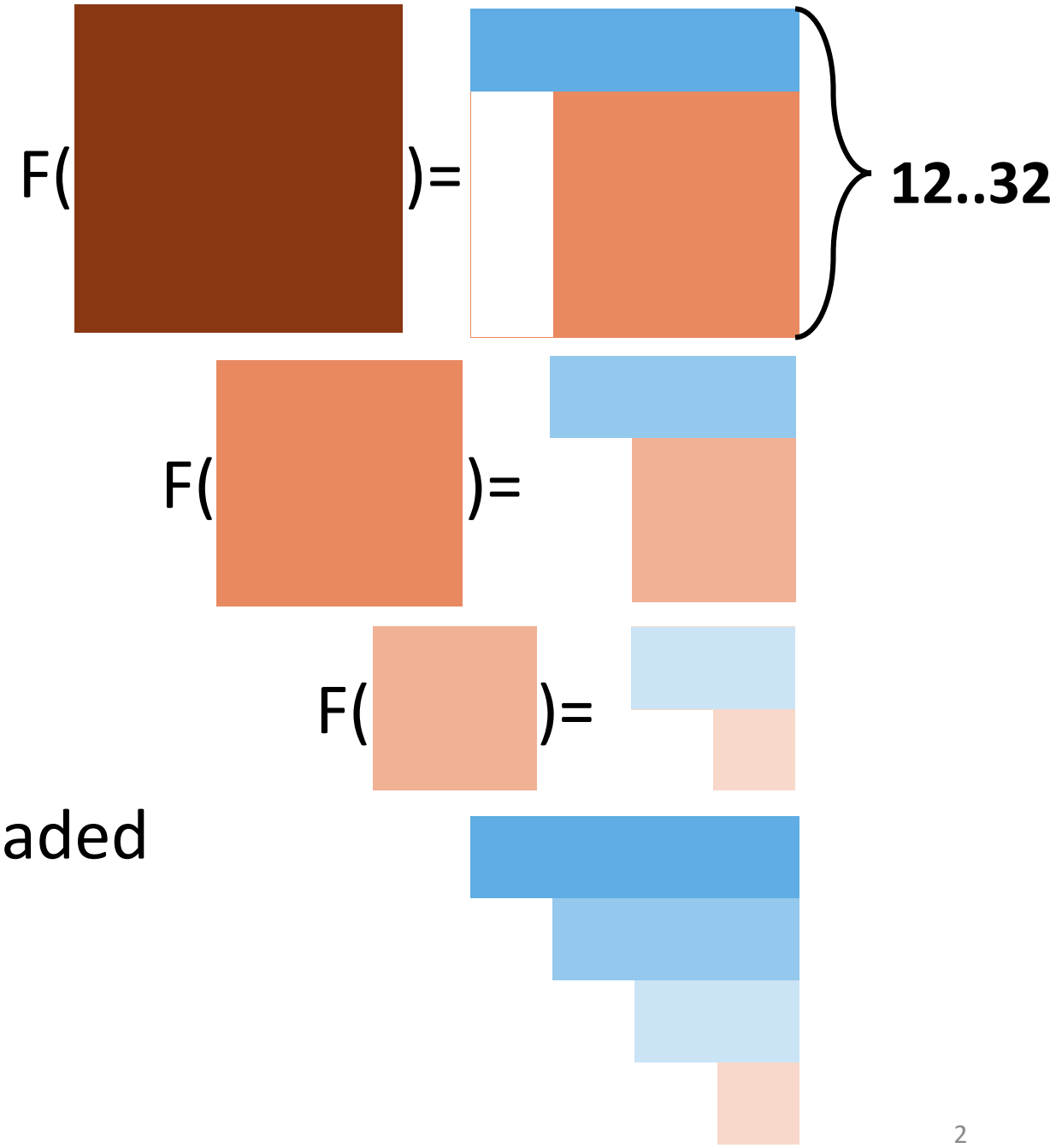
Feb 27th, 2020

POLY ARCH

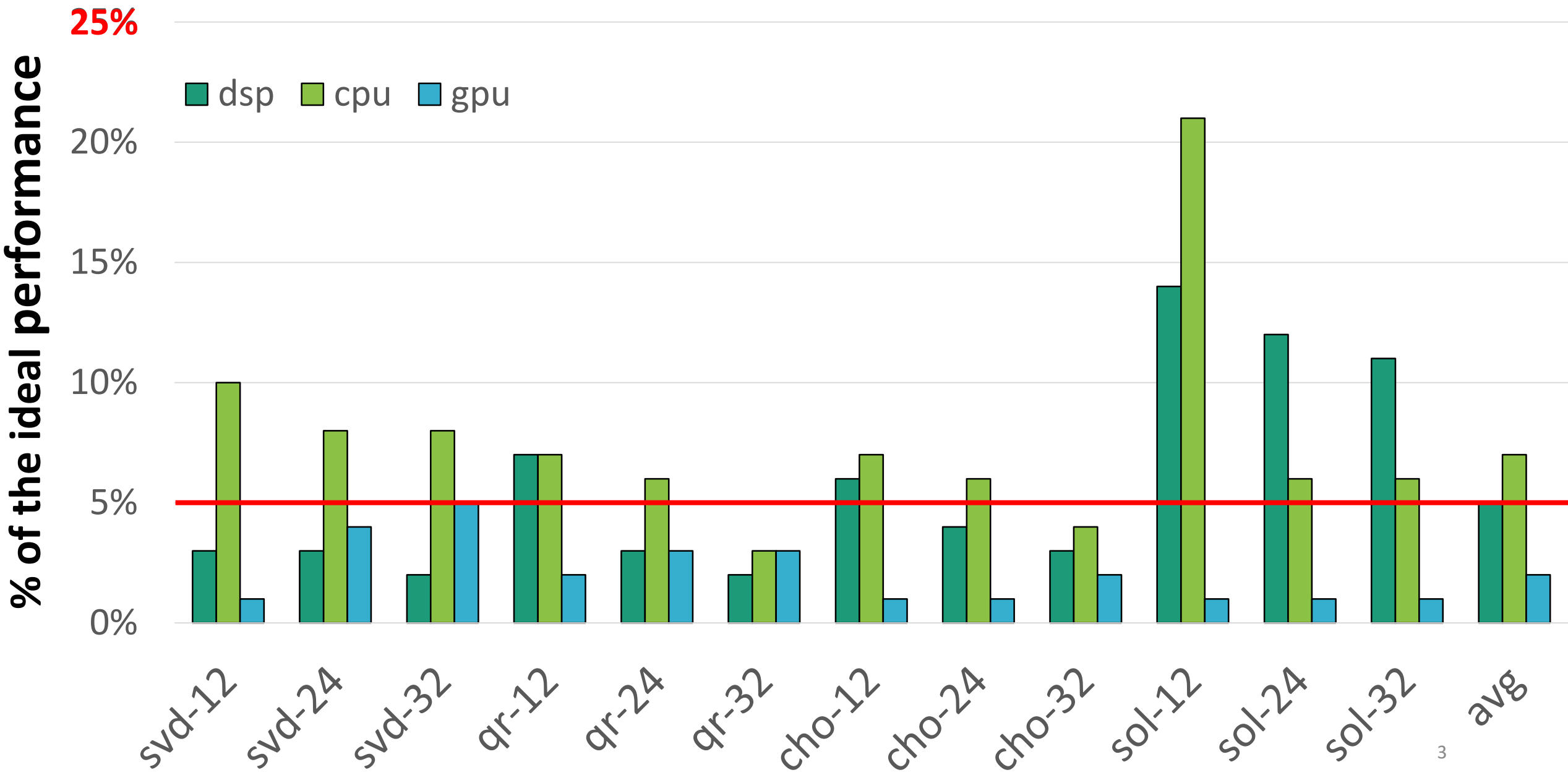
UCLA

Inductive Kernels

- Inductive Kernels
  - QR, SVD, Cholesky, Solver
- Challenges
  - Poor vectorization
  - Too small to be multi-threaded

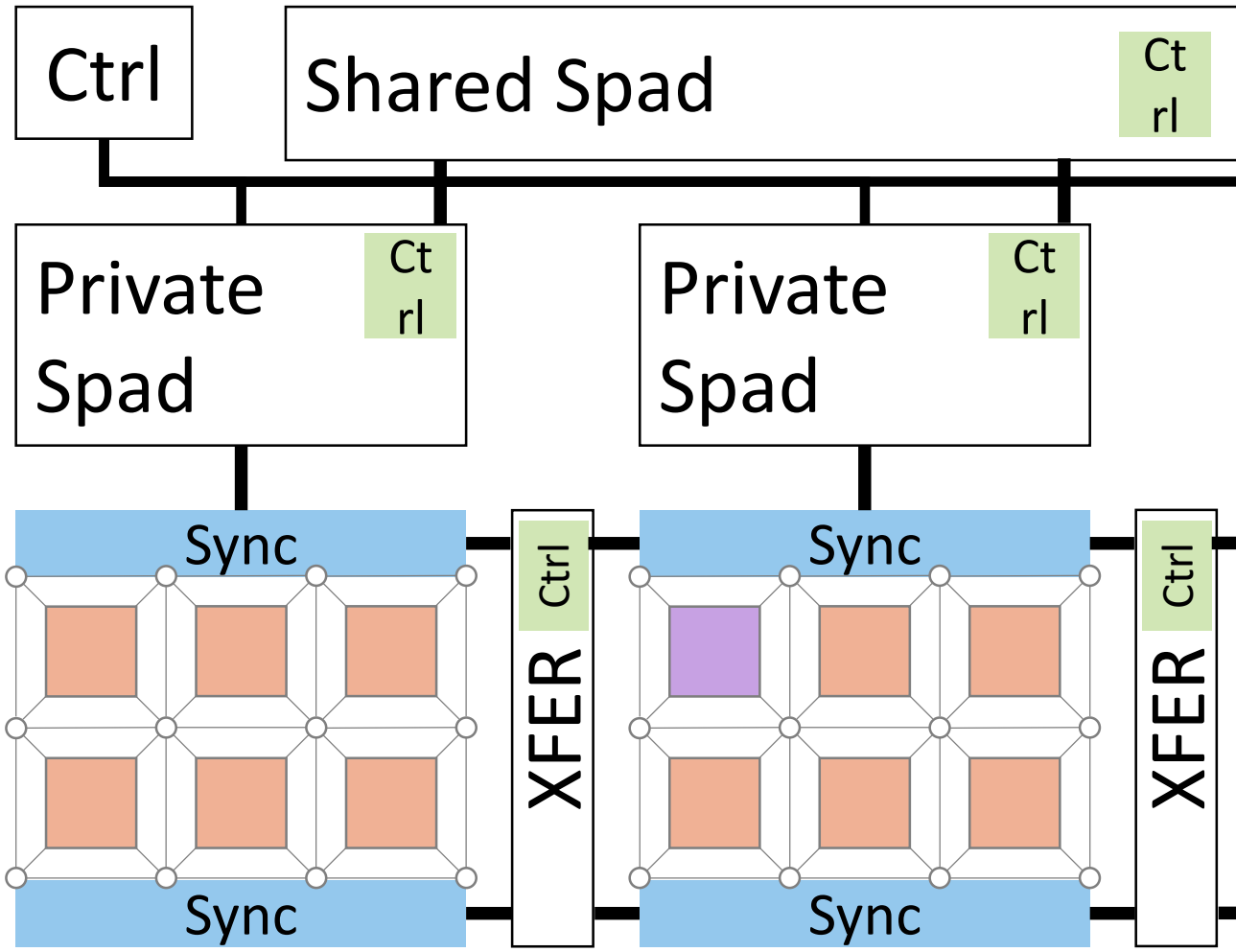


General Purpose Processor Performance



Our Goal: An Efficient, Flexible, and Specialized Architecture

- Base Design: Spatial Architecture
- Specialize Inductive Idioms
  - Hybridizing PEs, Inductive Control, and Implicit Vector Padding
  - Scale up with multiple lanes
- REVEL: **Re**configurable **Ve**ctor **L**anes
  - 3.4x, and 17x speedup over existing spatial architectures, and general purpose processors
  - 2x power and half area comparing with ASIC collection



Outline

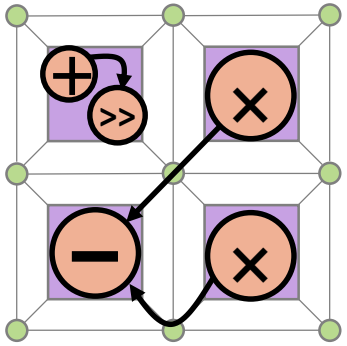
- **Background**
  - **“Dataflow” or “Systolic”? A tradeoff between cost and flexibility**
  - **Challenge 1: Synchronous Coordination**
  - **Challenge 2: Overwhelmed Processing Elements**
  - **Challenge 3: Overwhelmed Coordination**
    - **Inductive Access**
    - **Padding the Vectorization**
- REVEL: Reconfigurable Vector Lanes
- Evaluation

Spatial Architecture

- Architectures expose computing resource and on-chip network to programming interfaces

- Multiple instructions shared PE
- Dynamically scheduled execution

Tagged Dataflow

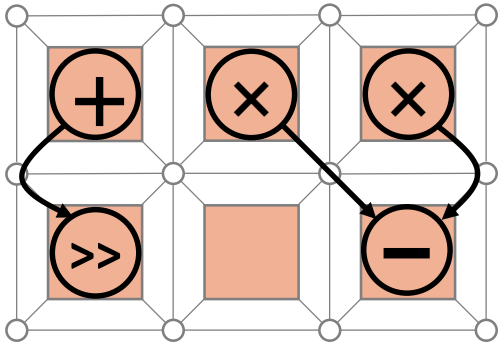


5.8x Area

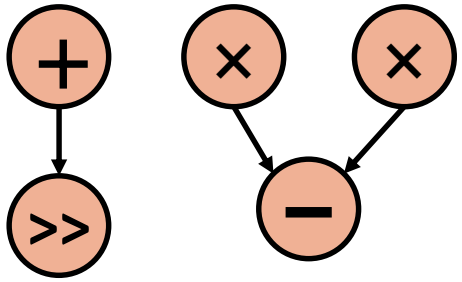
4.2x Power

- Each PE is dedicated to one instruction
- The timing of data arrival are determined when compilation

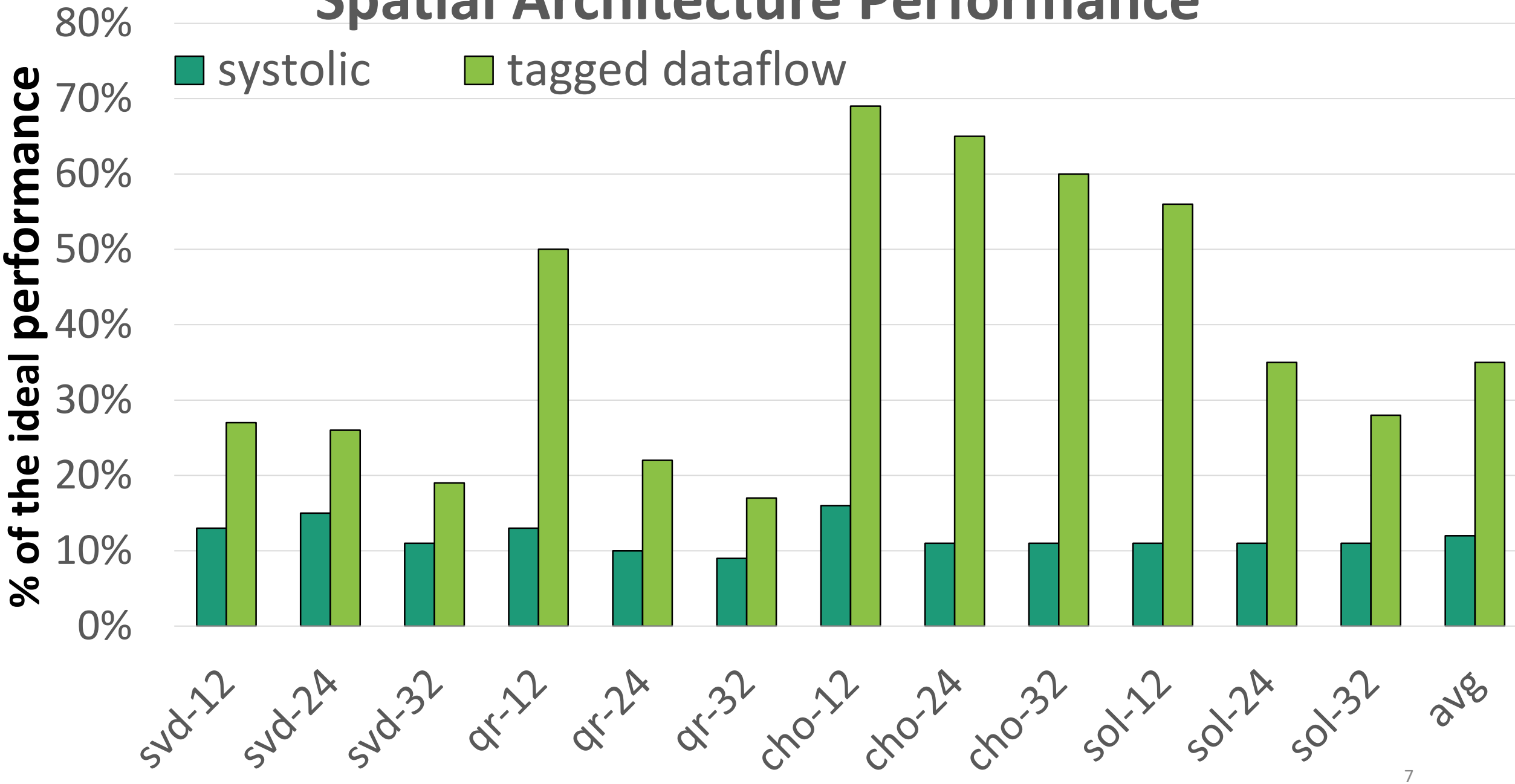
Systolic



Dependence Graph

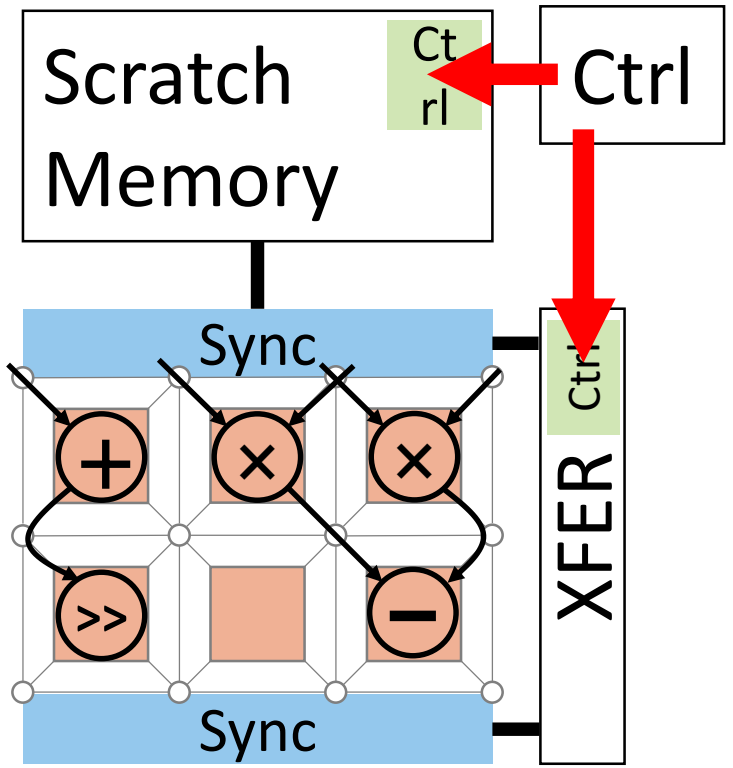


Spatial Architecture Performance



Base Design: Systolic Architecture with Decoupled Data Access

- Arithmetic operations are offloaded onto spatial architecture
- Data access are decoupled and coordinated by the controller
- Synchronization buffers serve as interfaces between dynamic/static timing

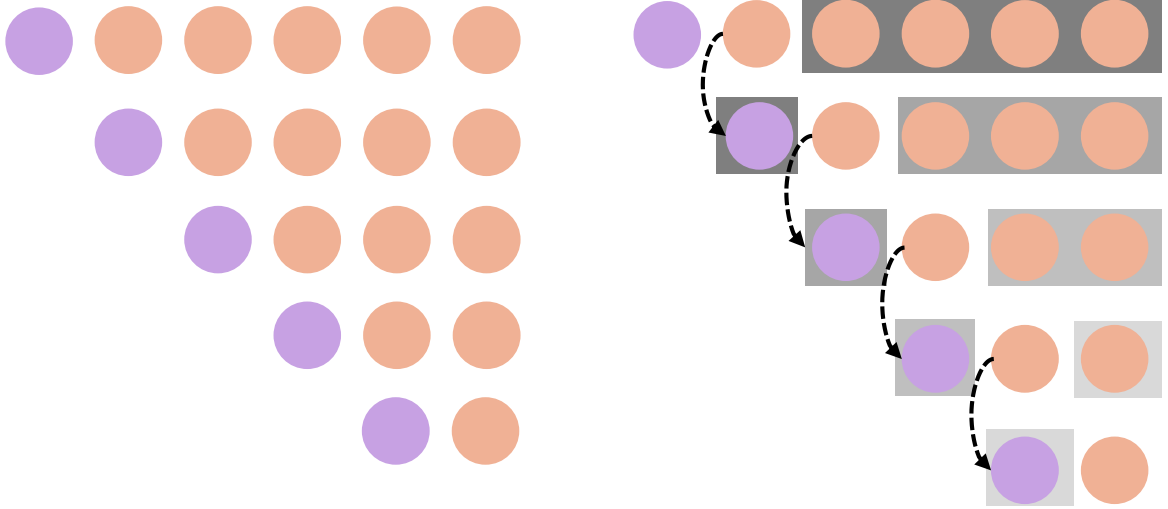


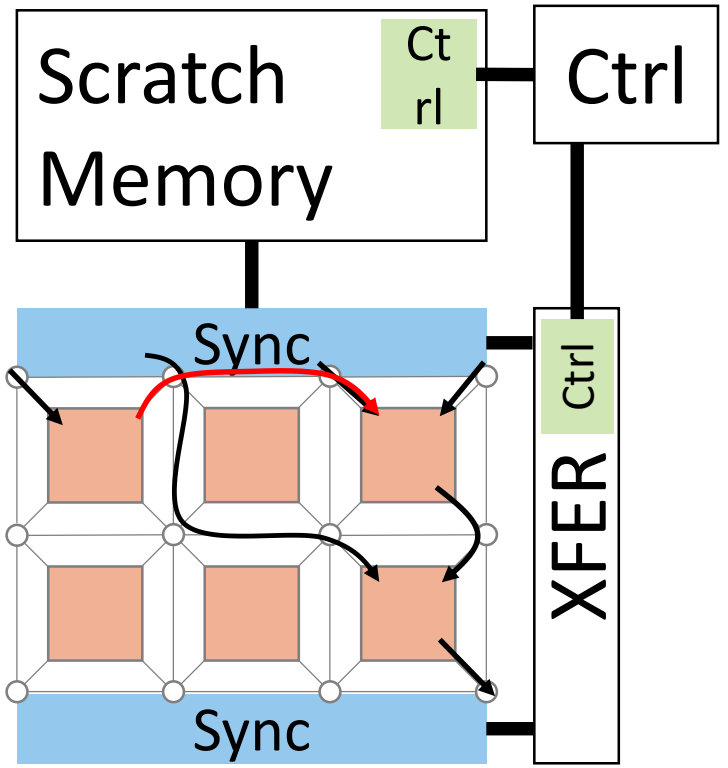
Challenge 1: Non-uniform Produce/Consume Rate

produce|consume

1|(n-j-1)

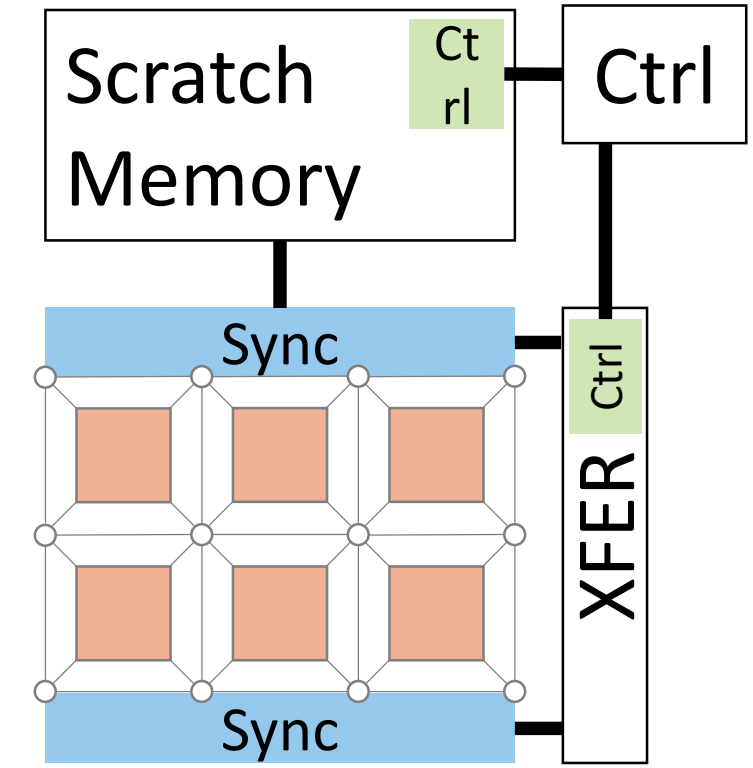
```
for (j=0; j<n; ++j) {
  x[j] = x[j] ÷ a[j,j];
  for (i=j+1; i<n; ++i)
    x[i] − x[j] × [j,i];
}
```





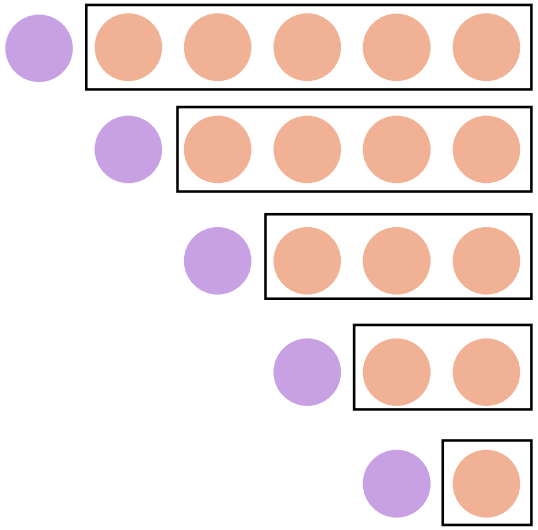
Challenge 2: Overwhelmed Processing Elements

```
for (k=0; k<n; ++k) {
  inv = 1.0 ÷ a[k,k];
  invsqrt = 1.0 ÷ sqrt(a[k,k]);
  for (j=k; j<n; ++j)
    l[j,k] = a[k,j] × invsqrt;
  for (j=k+1; j<n; ++j)
    for (i=j; i<n; ++i)
      a[j,i] = a[k,i] × a[k,j] × inv;
}
```



Challenge 3.1: Overwhelmed Coordination

```
for (j=0; j<n; ++j) {
   x[j] = x[j]/a[j,j];
   for (i=j+1; i<n; i+=2) {
     x[i] -= x[j]*a[j,i];
}}
```



Rectangular Slicing: a[n:m,p:q]

Triangular Slicing:

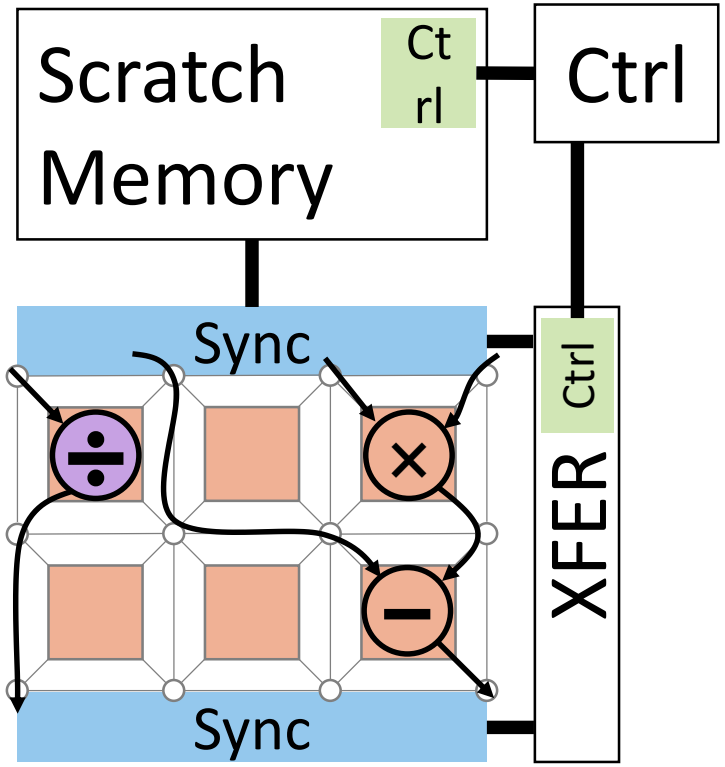
a[j+1:n]

a[j+2:n]

a[j+3:n]

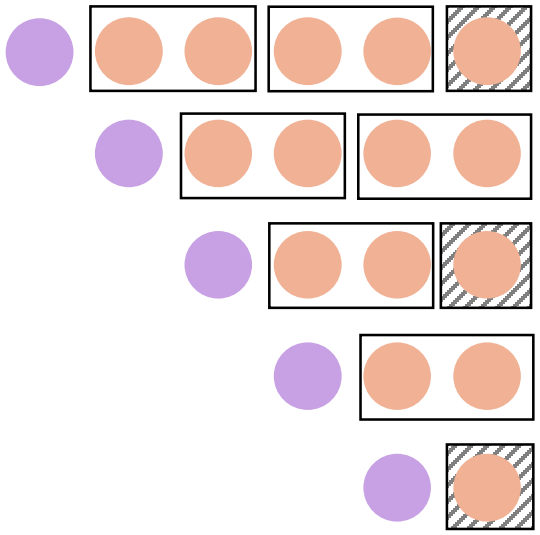
…

a[n-2:n]



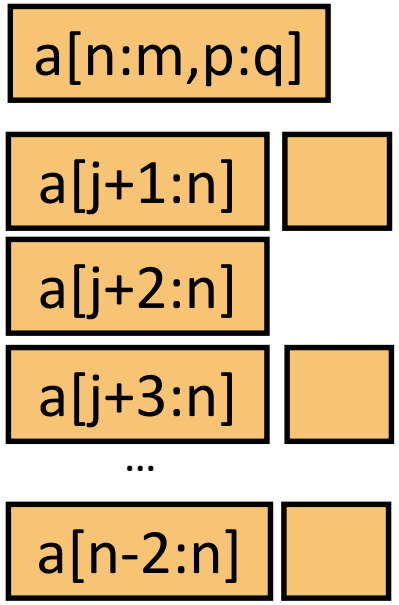
Challenge 3.2: Imperfect Loop Tiling

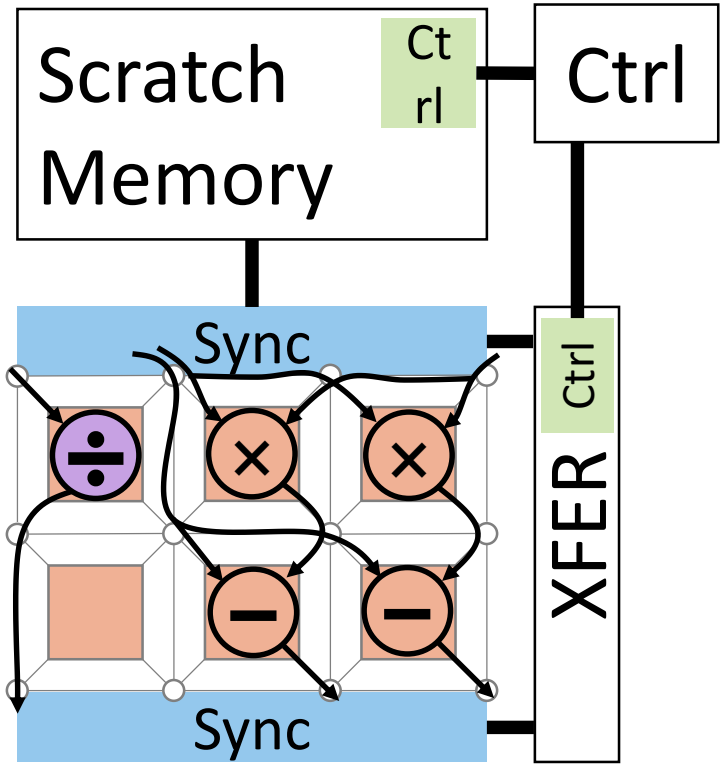
```
for (j=0; j<n; ++j) {
  x[j] = x[j]/a[j,j];
  for (i=j+1; i<n; i+=2) {
    x[i] -= x[j]*a[j,i];
    if (i+1<n) x[i+1] -= x[j]*a[j,i+1];
}}
```



Rectangular Slicing:

Triangular Slicing:



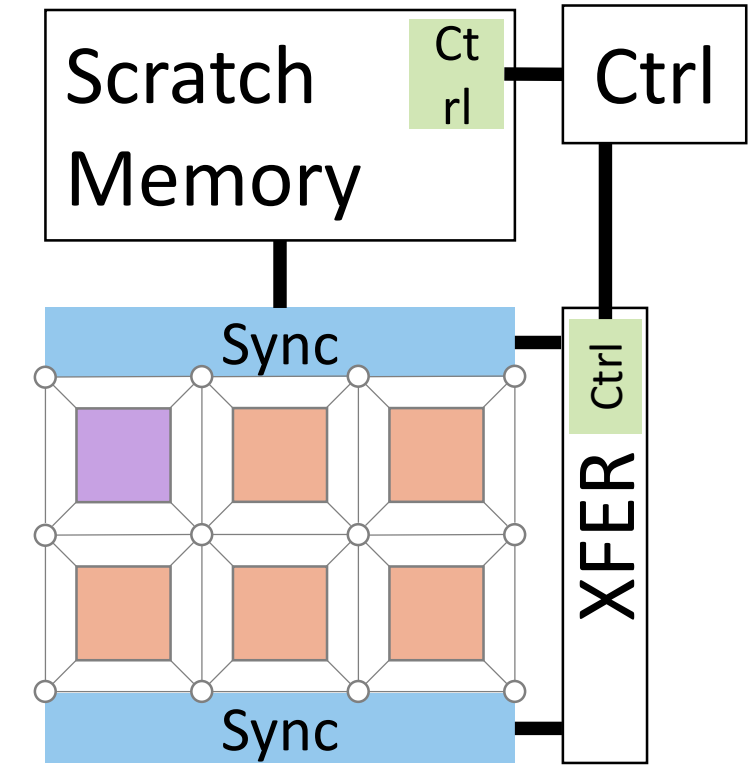


Outline

- Spatial Architecture
- **REVEL: Reconfigurable Vector Lane**
  - **Specialization 1: Hybridizing processing elements**
  - **Specialization 2: Coordinating non-uniform dependences**
  - **Specialization 3: Inductive Access Intrinsics**
  - **Specialization 4: Implicit vectorization predication**
  - **Scalability: Larger Spatial or Multiple Lanes?**
- Evaluation

Specialization 1: Hybridizing Systolic and Dataflow

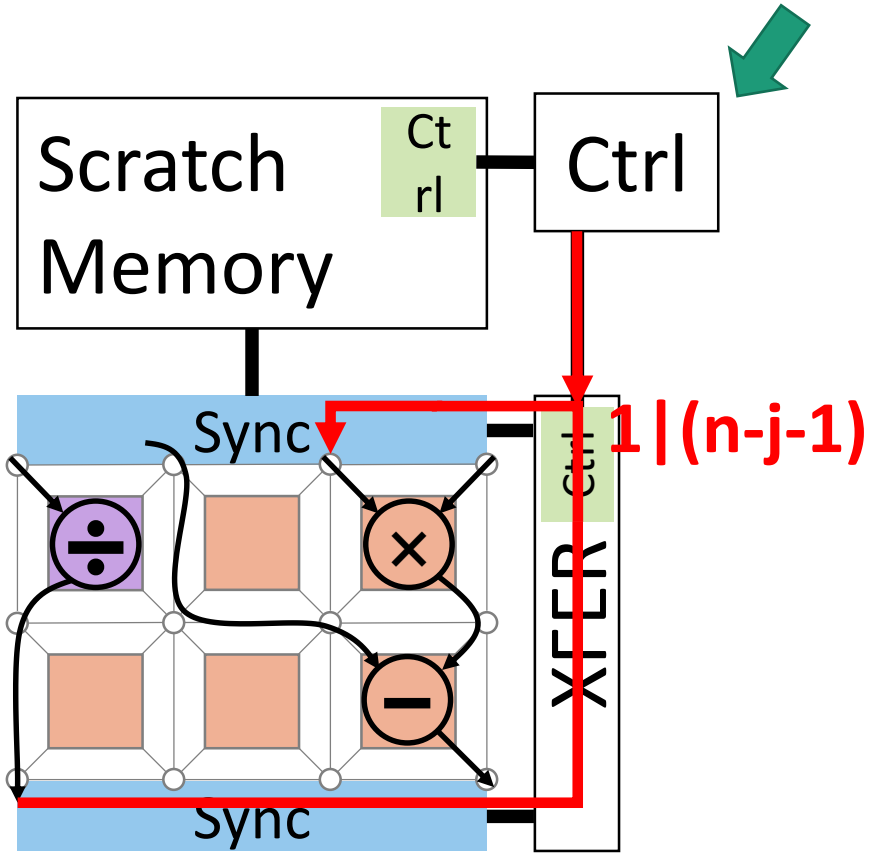
```
for (k=0; k<n; ++k) {
  inv = 1. / a[k,k];                    // O(1)
  invsqrt = 1. / sqrt(a[k,k]);          // O(1)
  for (j=k; j<n; ++j)                   // O(n)
    l[j,k] = a[k,j] * invsqrt;
  for (j=k+1; j<n; ++j)                 // O(n²)
    for (i=j; i<n; ++i)
      a[j,i] = a[k,i] - a[k,j] * inv;
}
```



Specialization 2: Coordinating Non-uniform Dependences

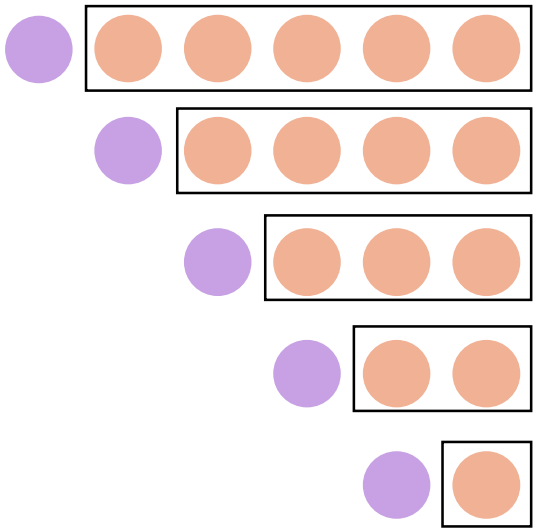
```
for (j=0; j<n; ++j) {
  x[j] = x[j]/a[j,j];
  for (i=j+1; i<n; ++i)
    x[i] -= x[j]*a[j,i];
}
```

1|(n-j-1)



Specialization 3: Inductive Memory Access

```
for (j=0; j<n; ++j) {
  x[j] = x[j]/a[j,j];
  for (i=j+1; i<n; i+=2) {
    x[i] -= x[j]*a[j,i];
    if (i+1<n) x[i+1] -= x[j]*a[j,i+1];
}}
```



Rectangular Slicing: a[n:m,p:q]

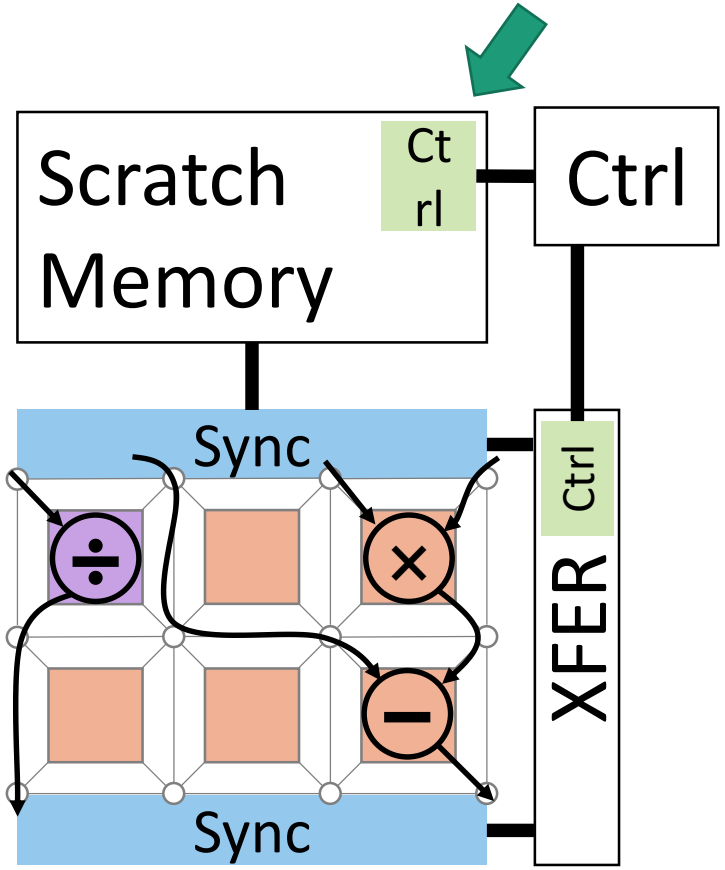
Triangular Slicing: j_{k+1}^{n} a[k,j:n]

a[j+2:n]

a[j+3:n]

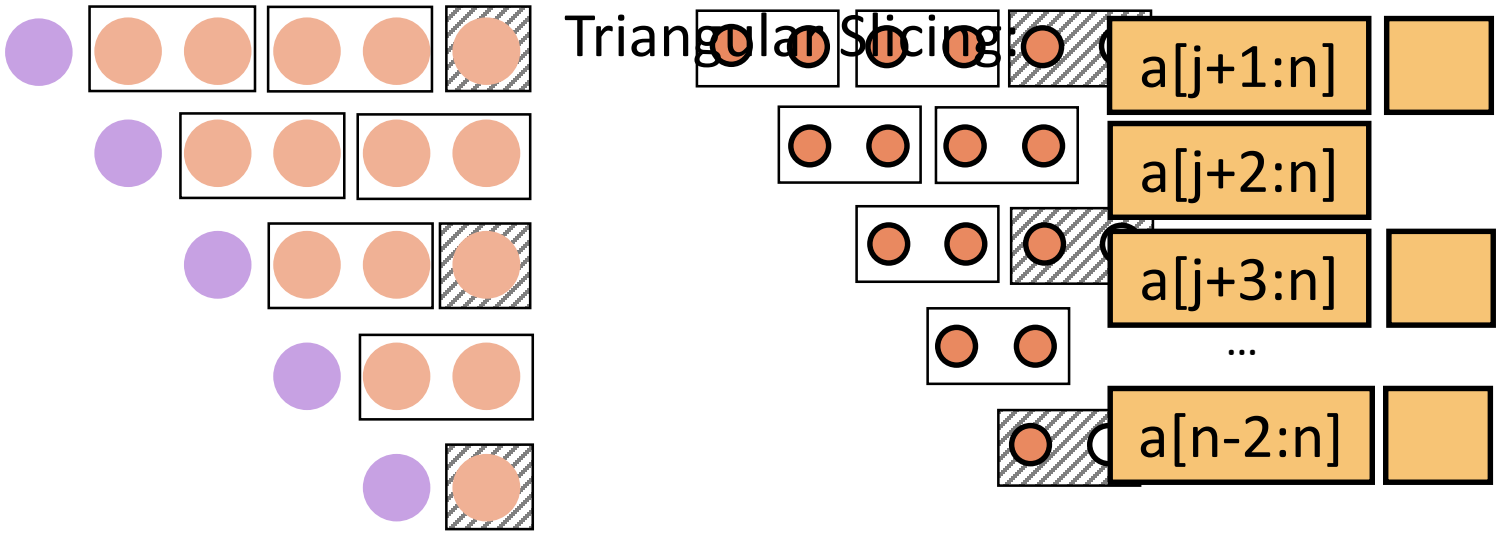
…

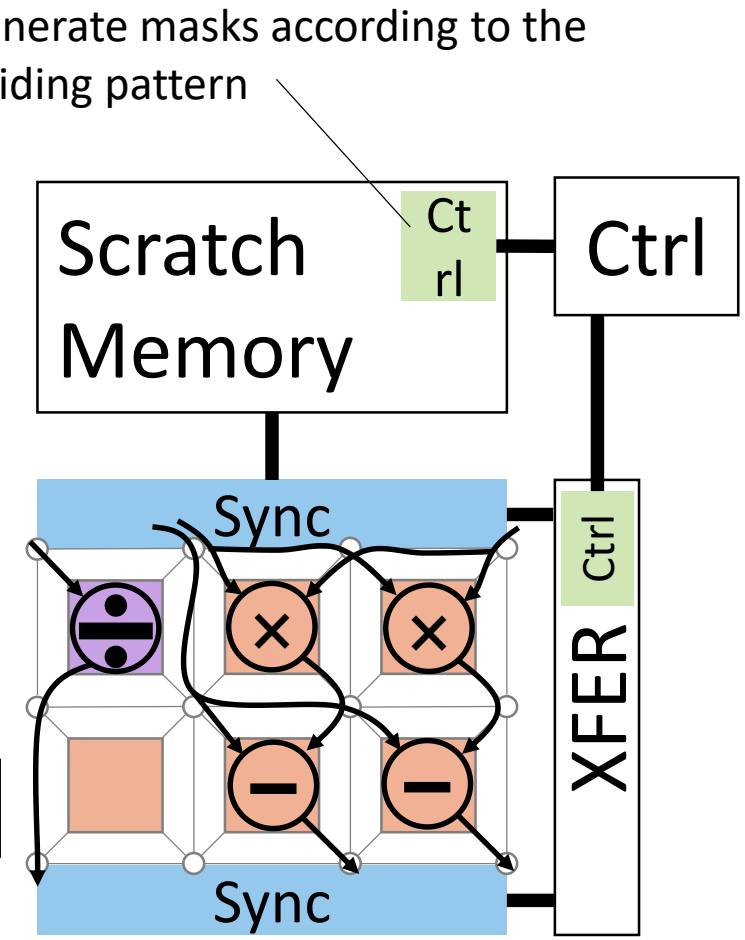
a[n-2:n]



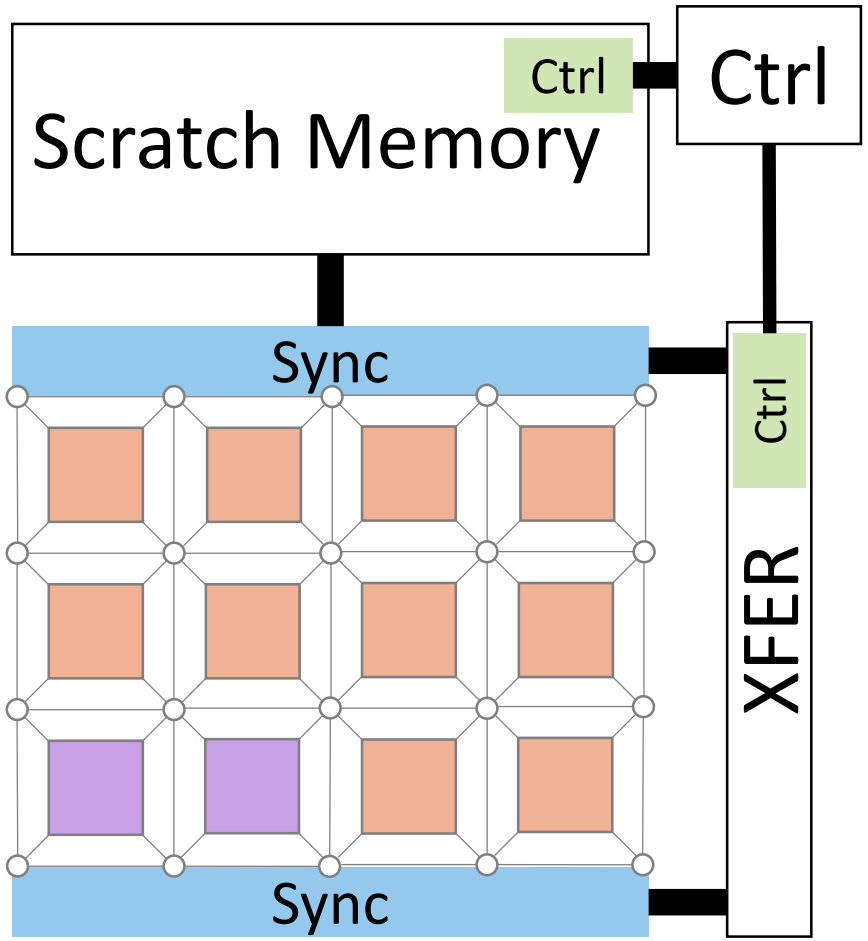
Specialization 4: Implicit Vectorization Predication

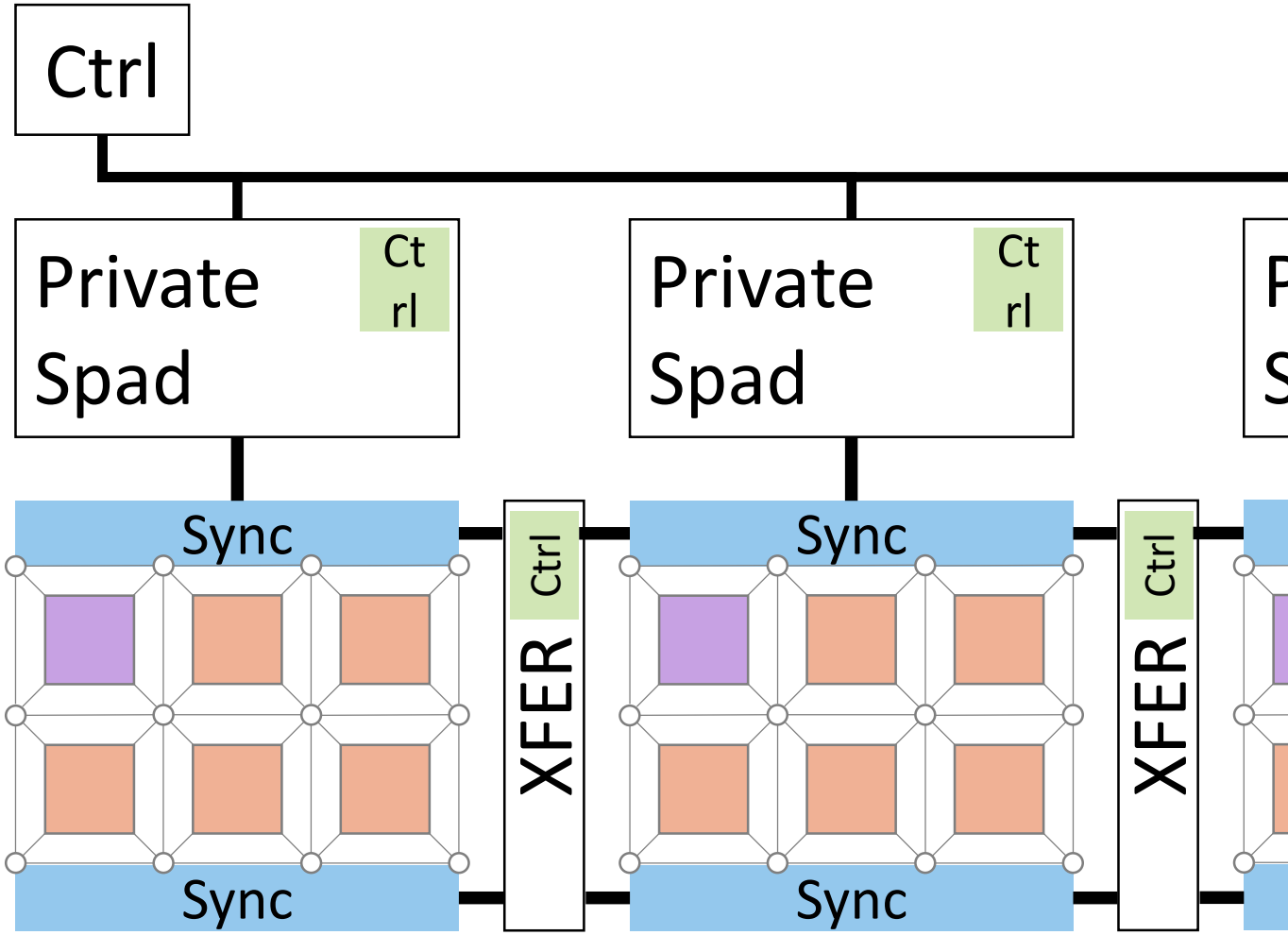
```
for (j=0; j<n; ++j) {
  x[j] = x[j]/a[j,j];
  for (i=j+1; i<n; i+=2) {
    x[i] -= x[j]*a[j,i];
    if (i+1<n) x[i+1] -= x[j]*a[j,i+1];
}}
```





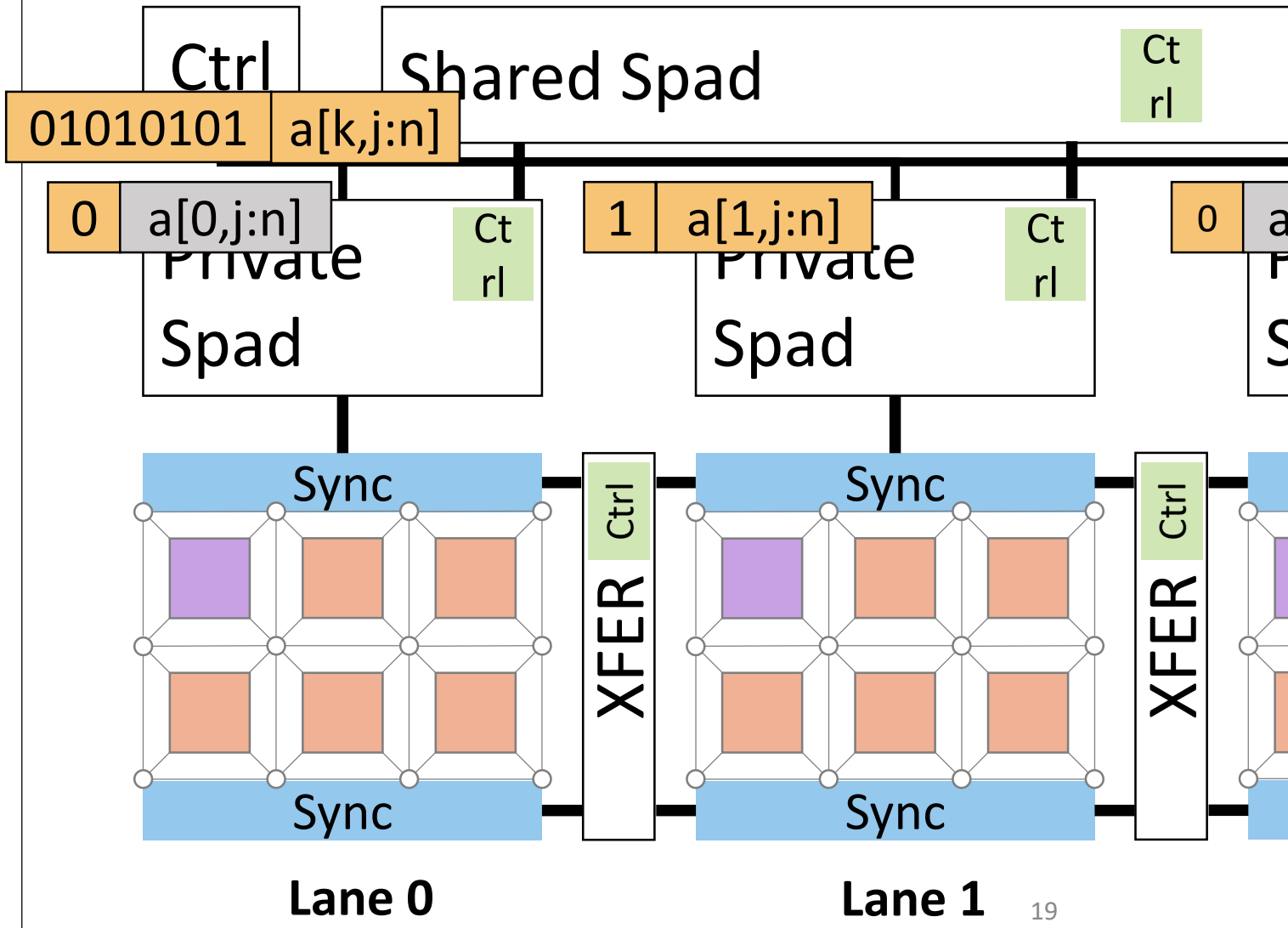
Scalability: A larger mesh or multiple lanes?





REVEL: **Re**configurable **Ve**ctor **L**anes

- A centralized control core commands multiple lanes
  - Predication based broadcast
  - SIMT-like control commands
- Each lane is independent to execute



Outline

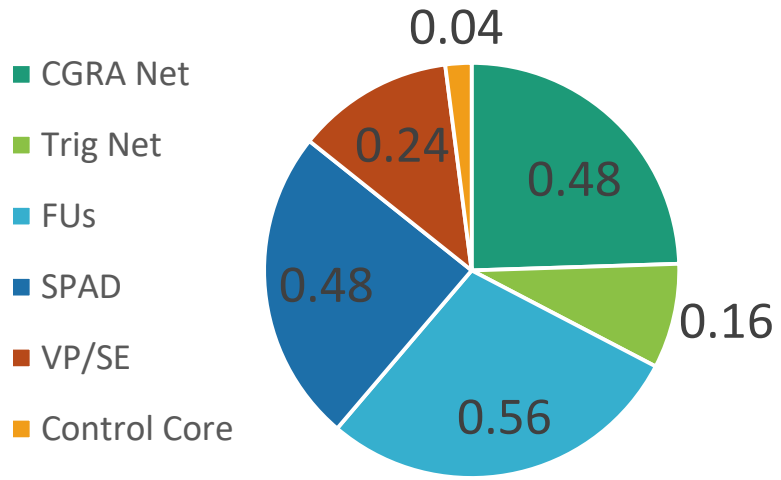
- Spatial Architecture
- REVEL: Reconfigurable Vector Lanes
- **Evaluation**
  - **Methodology**
  - **Speedup**

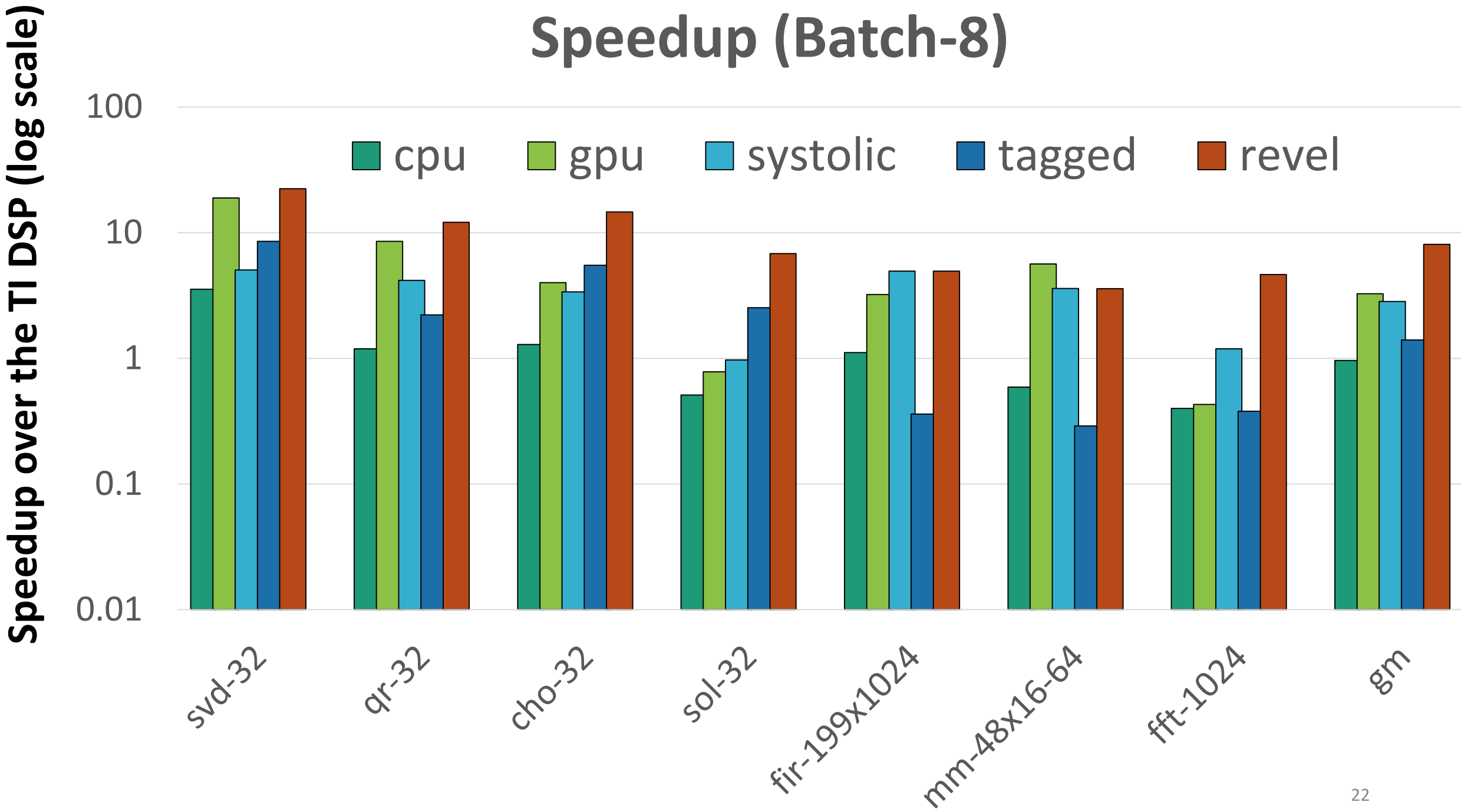
Evaluation Methodology

- Performance
  - Gem5 RISCV in-order core integrated with a cycle-accurate spatial architecture simulator
    - Extending the stream-dataflow ISA
  - Baselines:
    - Intel(R) Xeon(R) Silver 4116 @2.10GHz (Intel MKL)
    - TI6678 DSP @1.25GHz (TI DSPLIB)
    - NVIDIA Titan (cuBlas)

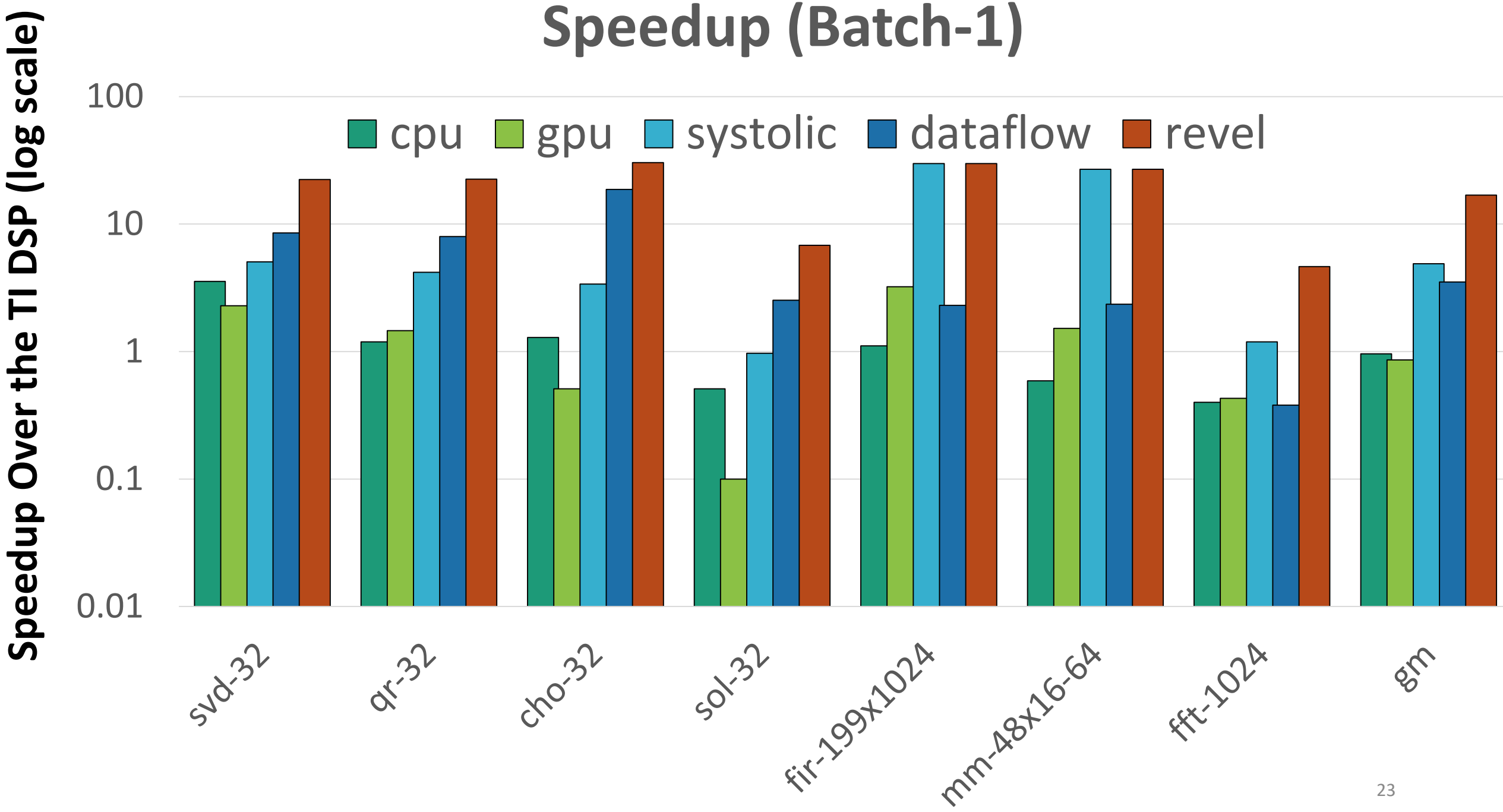
} Same peak performance

- Power/Area
  - Spatial Architecture implemented in Chisel
  - Synthesized in Synopsys DC 28nm @1.25GHz
  - SRAM power/area are estimated by CACTI.




Speedup (Batch-8)
Speedup over the TI DSP (log scale)
100
10
1
0.1
0.01
cpu
gpu
systolic
tagged
revel
svd-32
qr-32
cho-32
sol-32
fir-199x1024
mm-48x16-64
fft-1024
gm

Speedup (Batch-1)



Conclusion

- According to our results, REVEL and its hybrid architecture is a promising next-generation digital signal processing architecture.
- More broadly, our work demonstrates the importance of considering multiple execution models.