EE382V (17325): Principles in Computer Architecture
Parallelism and Locality
Fall 2007

Lecture 18 – Stream Processors (II)

Mattan Erez

The University of Texas at Austin

Outline

- **Summary of previous lecture:**
  - **Parallelism**
  - **Locality**
  - **Hierarchical control and scheduling**
  - **Throughput oriented I/O**
- Compute cluster
- SRF
- Stream memory system

- Many slides courtesy Jung Ho Ahn (HP Labs)

Hardware Efficiency → Greater Software Responsibility

- Hardware matches VLSI strengths
  - Throughput-oriented design
  - Parallelism, locality, and partitioning
  - Hierarchical control to simplify instruction sequencing
  - Minimalistic HW scheduling and allocation
- Software **given** more explicit control
  - Explicit hierarchical scheduling and latency hiding (***schedule***)
  - Explicit parallelism (***parallelize***)
  - Explicit locality management (***localize***)

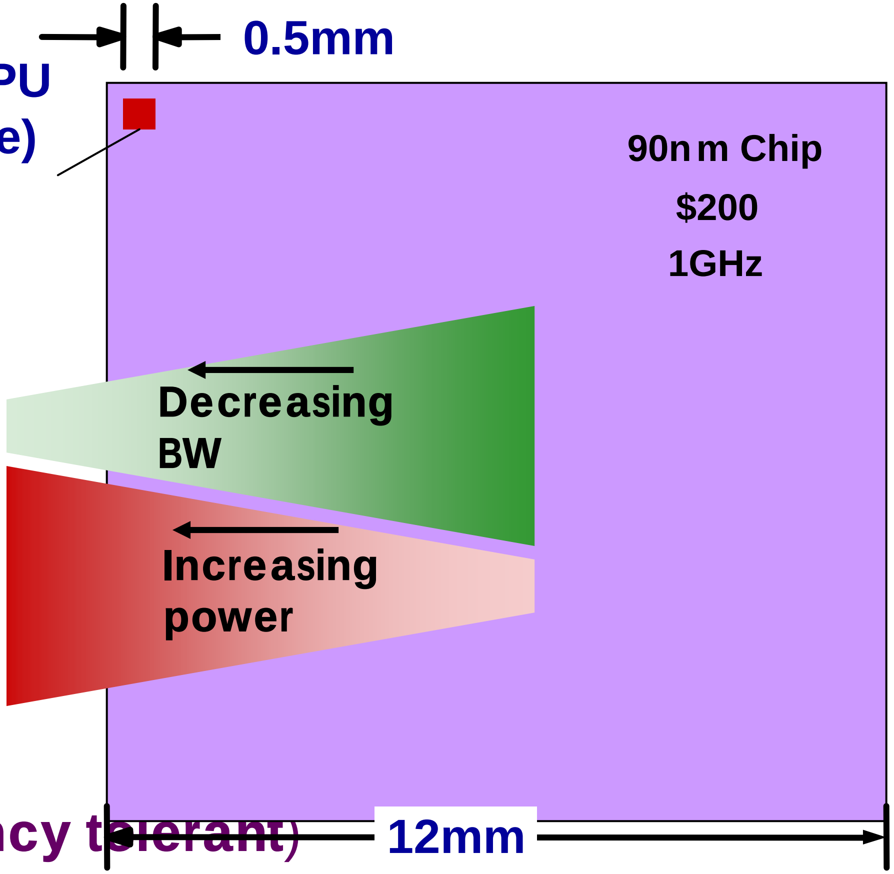
Must reduce HW "waste" but no free lunch

Outline

- Hardware strengths and the stream execution model
- Stream Processor hardware
  - Parallelism
  - Locality
  - Hierarchical control and scheduling
  - Throughput oriented I/O
- Implications on the software system
  - Current status
- HW and SW tradeoffs and tuning options
  - Locality, parallelism, and scheduling
- Petascale implications

Effective Performance on Modern VLSI

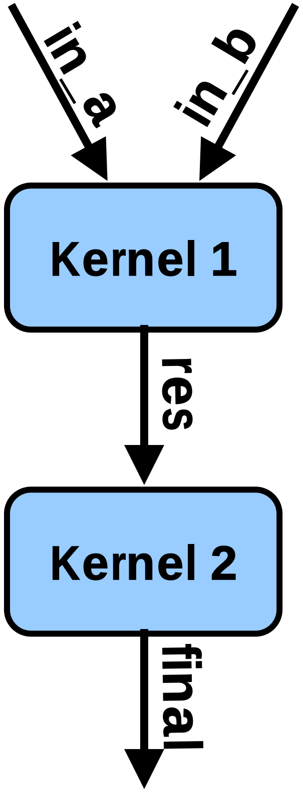
- Parallelism
  - 10s of FPUs per chip
  - Efficient control
- Locality
  - Reuse reduces global BW
  - Locality lowers power
- Bandwidth management
  - Maximize pin utilization
  - Throughput oriented I/O (**latency tolerant**)

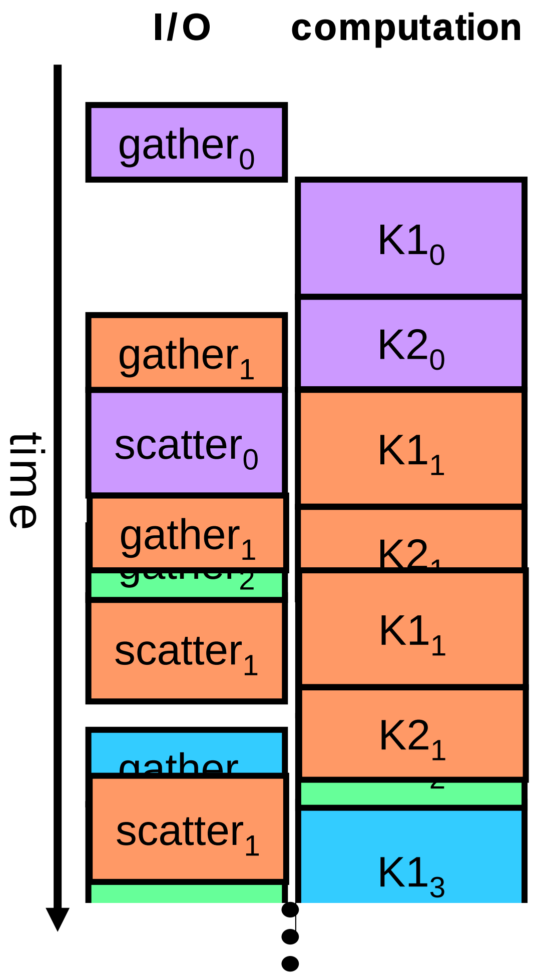


Parallelism, locality, bandwidth, and efficient control (and latency hiding)

Bulk Operations are Good for Hardware

- Parallelism
  - 10s of FPUs per chip
  - Efficient control
- Locality
  - Reuse reduces global BW
  - Locality lowers power
- Latency Tolerance
  - Throughput oriented I/O
  - Increasing on-/off-chip latencies
- Minimum control overhead





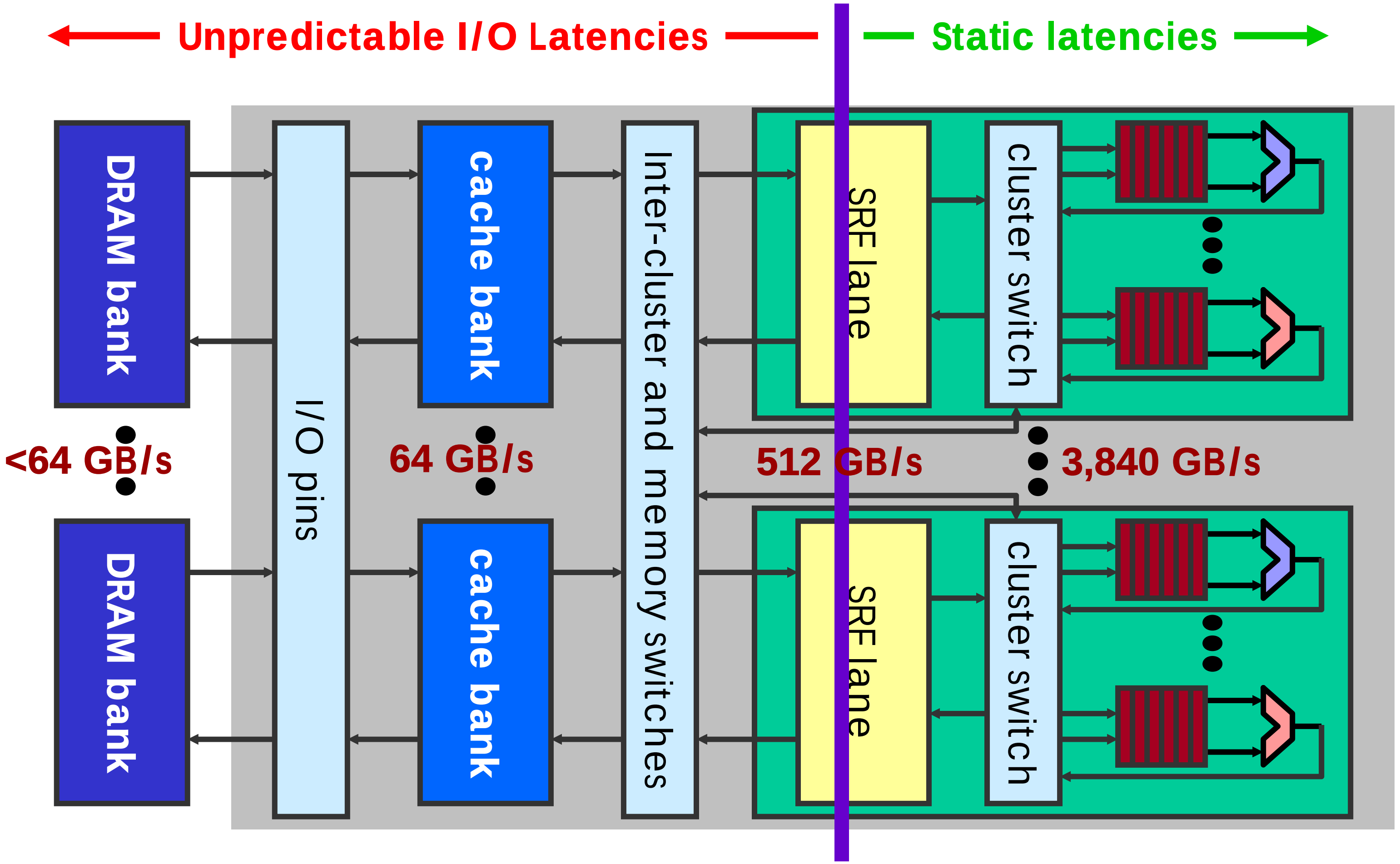
Hardware designed for throughput and not latency (memory BW, FLOPS, bulk exceptions, bulk coherency, ...)

Generalizing the Stream Model

- Medium granularity bulk operations
  - Kernels and stream-LD/ST
- Predictable sequence (of bulk operations)
  - Latency hiding, explicit communication
- Hierarchical control
  - Inter- and intra-bulk
- Throughput-oriented design
- Locality and parallelism
  - kernel locality + producer-consumer reuse
  - Parallelism within kernels

Generalized stream model matches VLSI requirements

SRF Decouples Execution from Memory



Decoupling enables efficient static architecture

Separate address spaces (MEM/SRF/LRF)

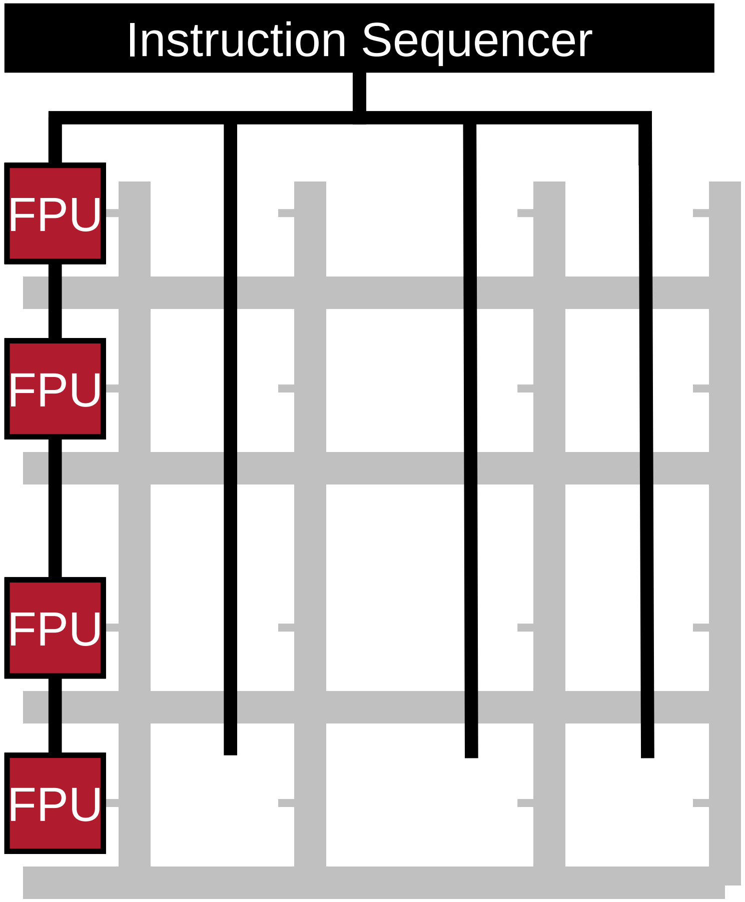
Outline

- Summary of previous lecture:
  - Parallelism
  - Locality
  - Hierarchical control and scheduling
  - Throughput oriented I/O
- **Compute cluster**
- SRF
- Stream memory system

Parallelism Tradeoffs and Tuning Opportunities

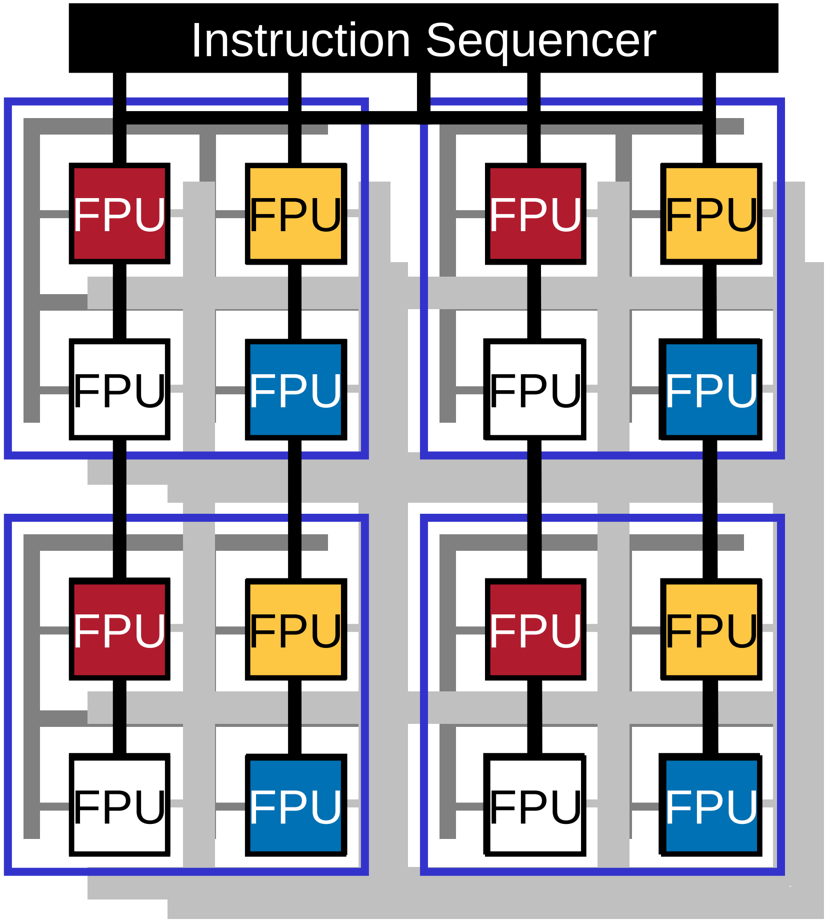
- 3 types of parallelism
  - Data Level Parallelism
  - Instruction Level Parallelism
  - Thread (Task) Level Parallelism

Data-Level Parallelism in Stream Processors



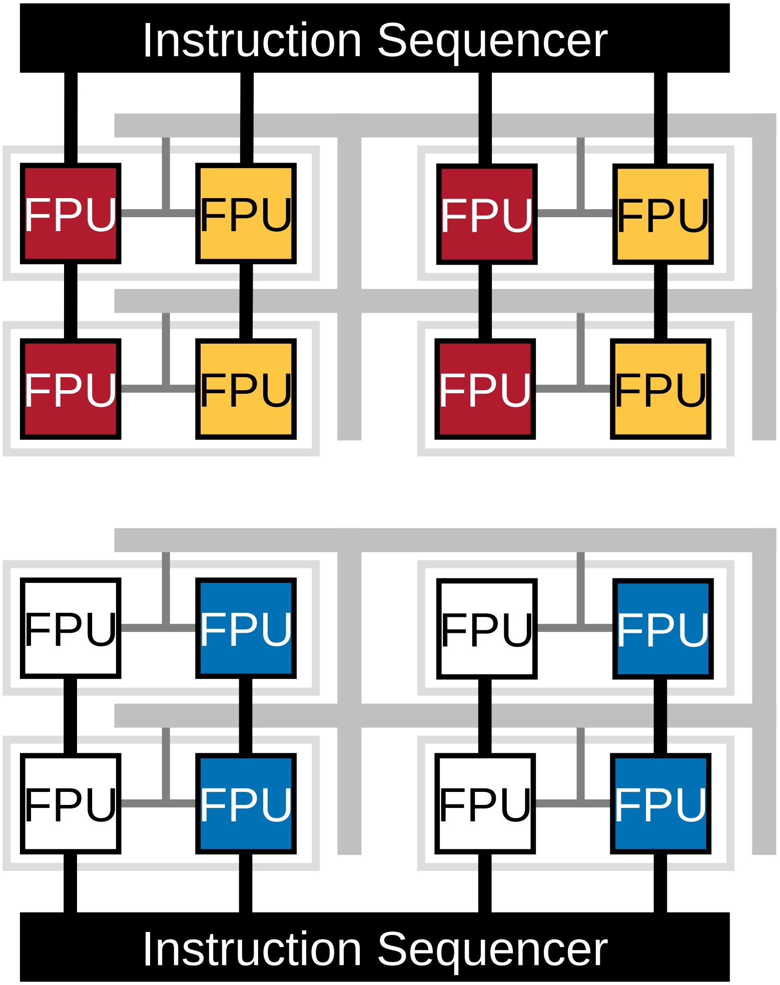
- SIMD
- Independent indexing per FPU
- Full crossbar between FPUs
- No sub-word operation

Data- and Instruction-Level Parallelism in Stream Processors



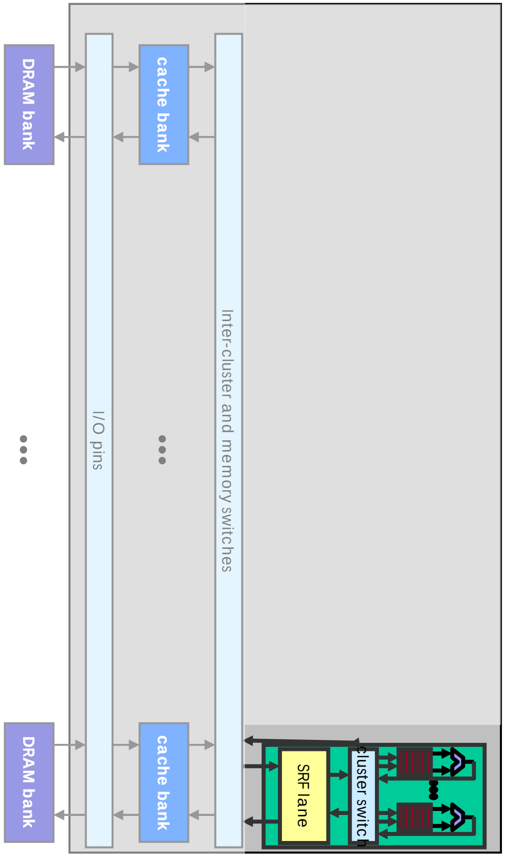
- A group of FPUs = A Processing Element (PE) = A Cluster
- VLIW
- Hierarchical switch provides area efficiency

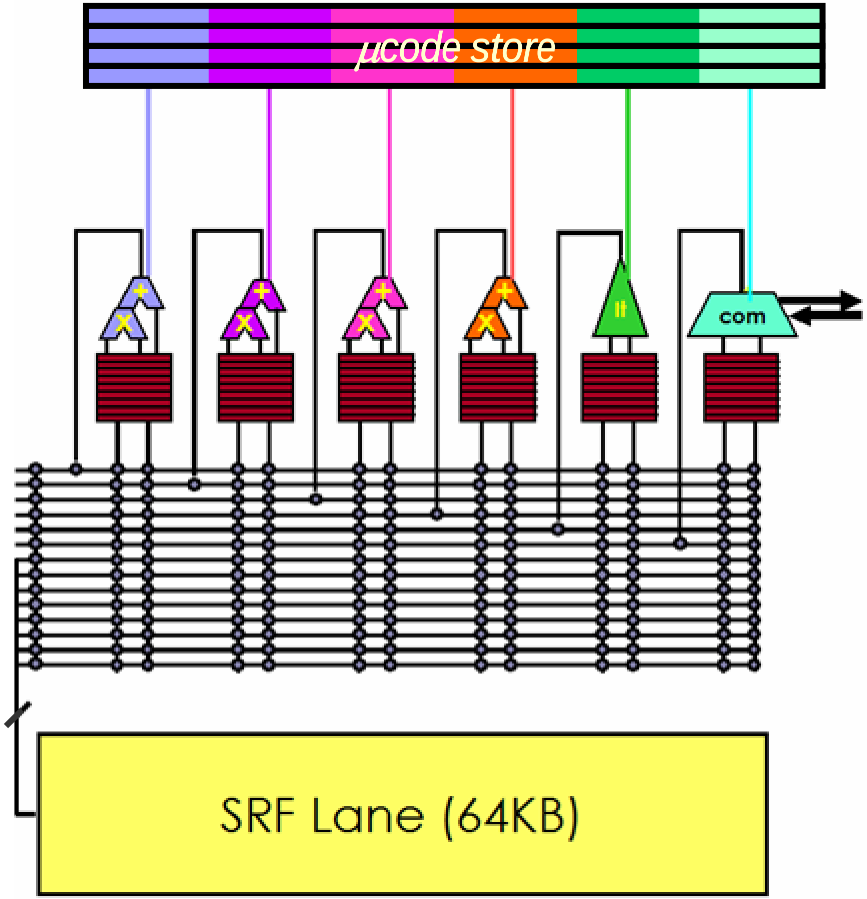
Data-, Instruction- and Thread-Level Parallelism in Stream Processors



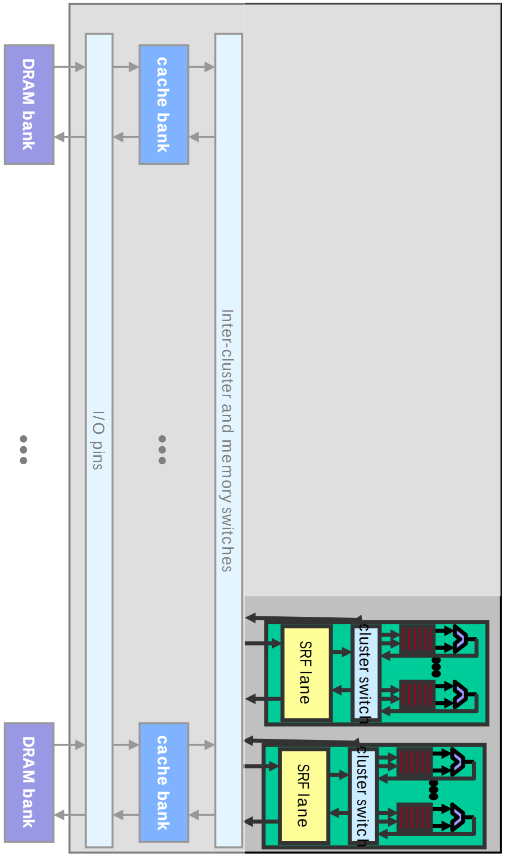
- Sequencer group
  - Each instruction sequencer runs different kernels

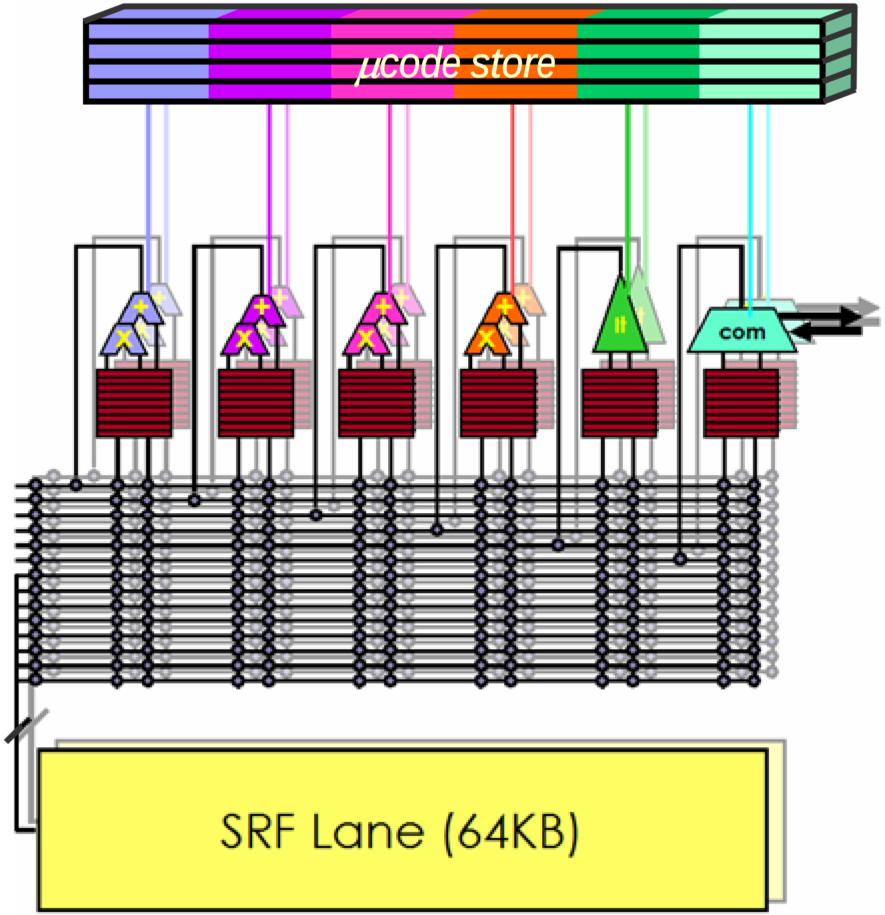
Clusters Enable Locality, Use ILP and DLP for Efficient Instruction Supply



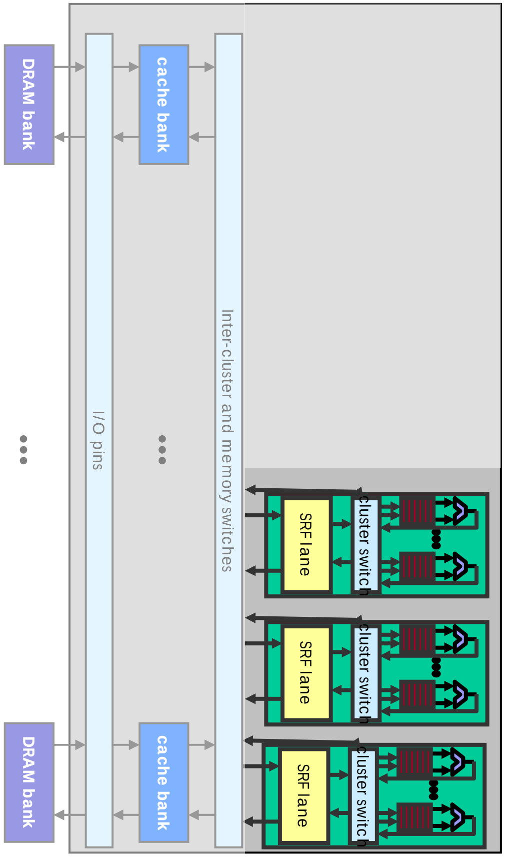


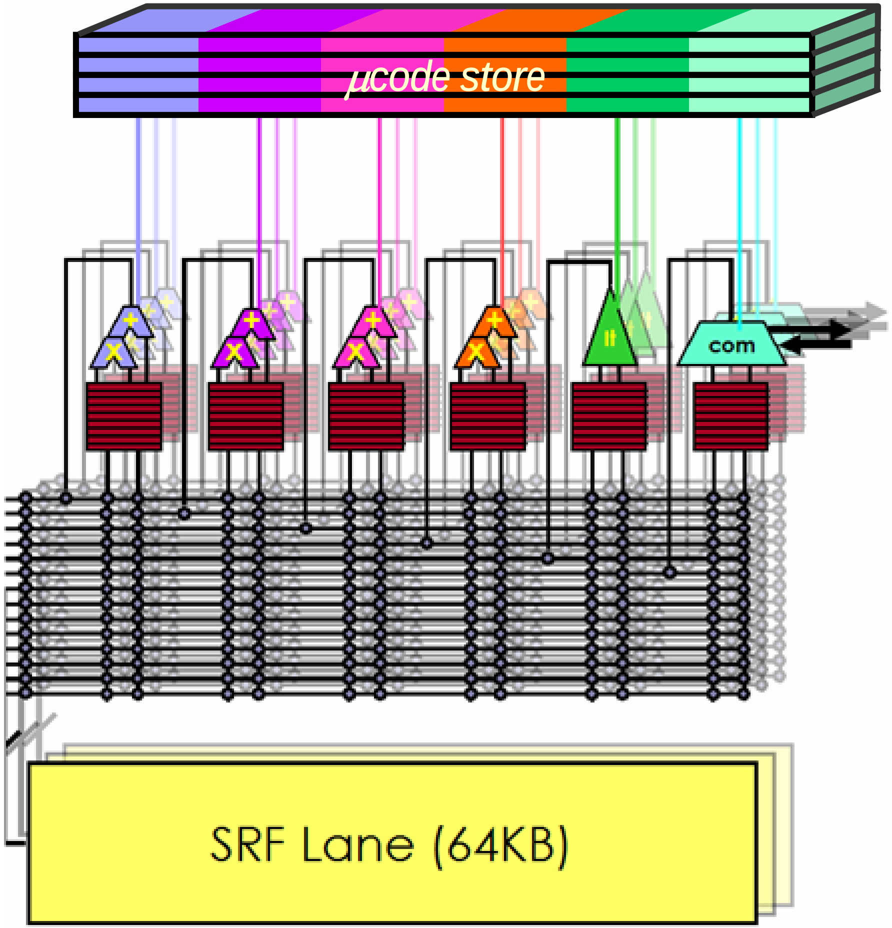
Stream Architecture Exploits ILP and DLP for Efficient Instruction Supply



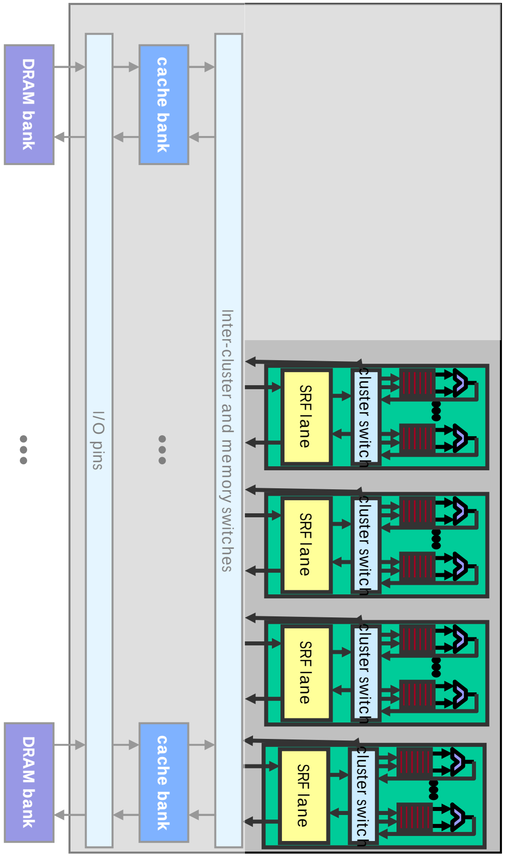


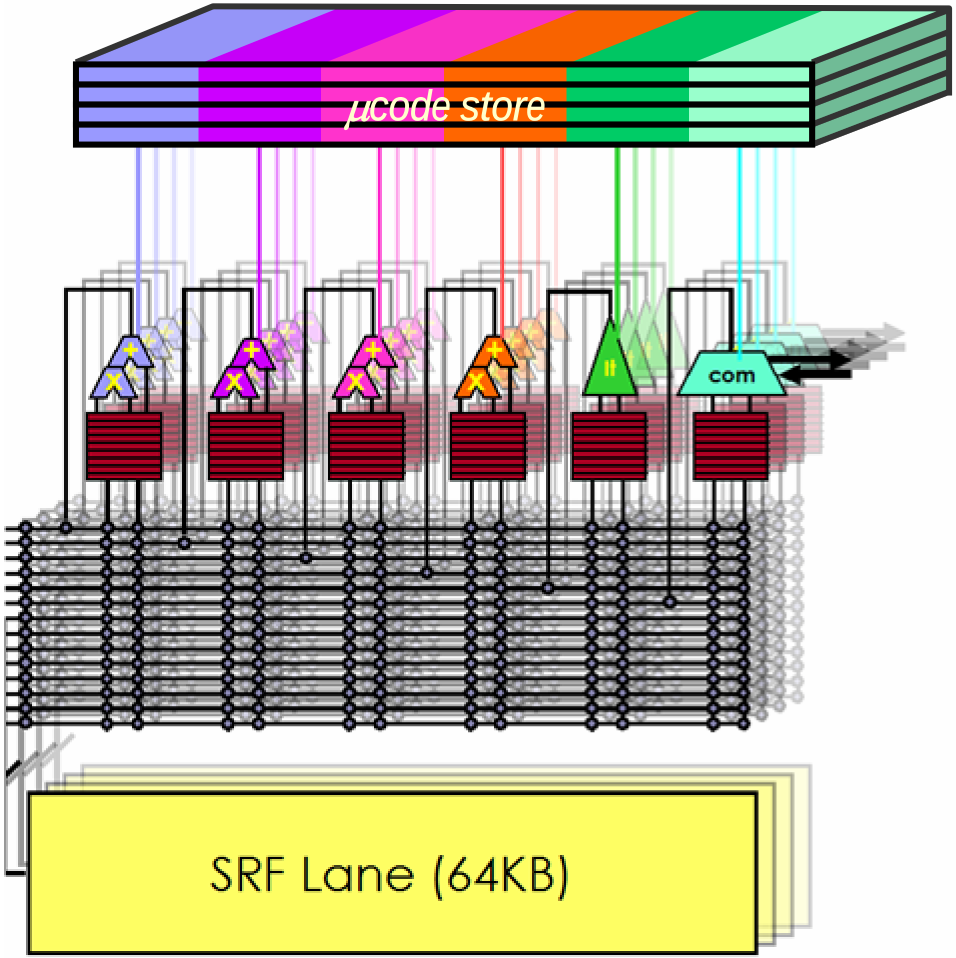
Stream Architecture Exploits ILP and DLP for Efficient Instruction Supply



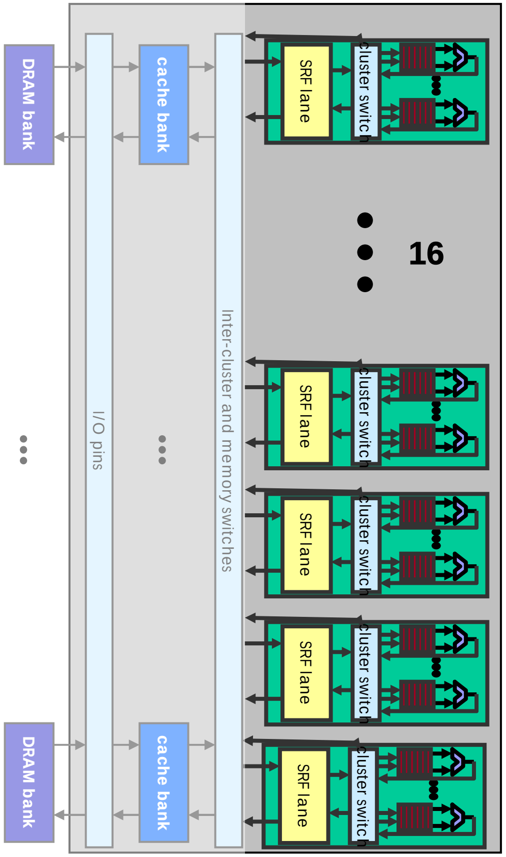


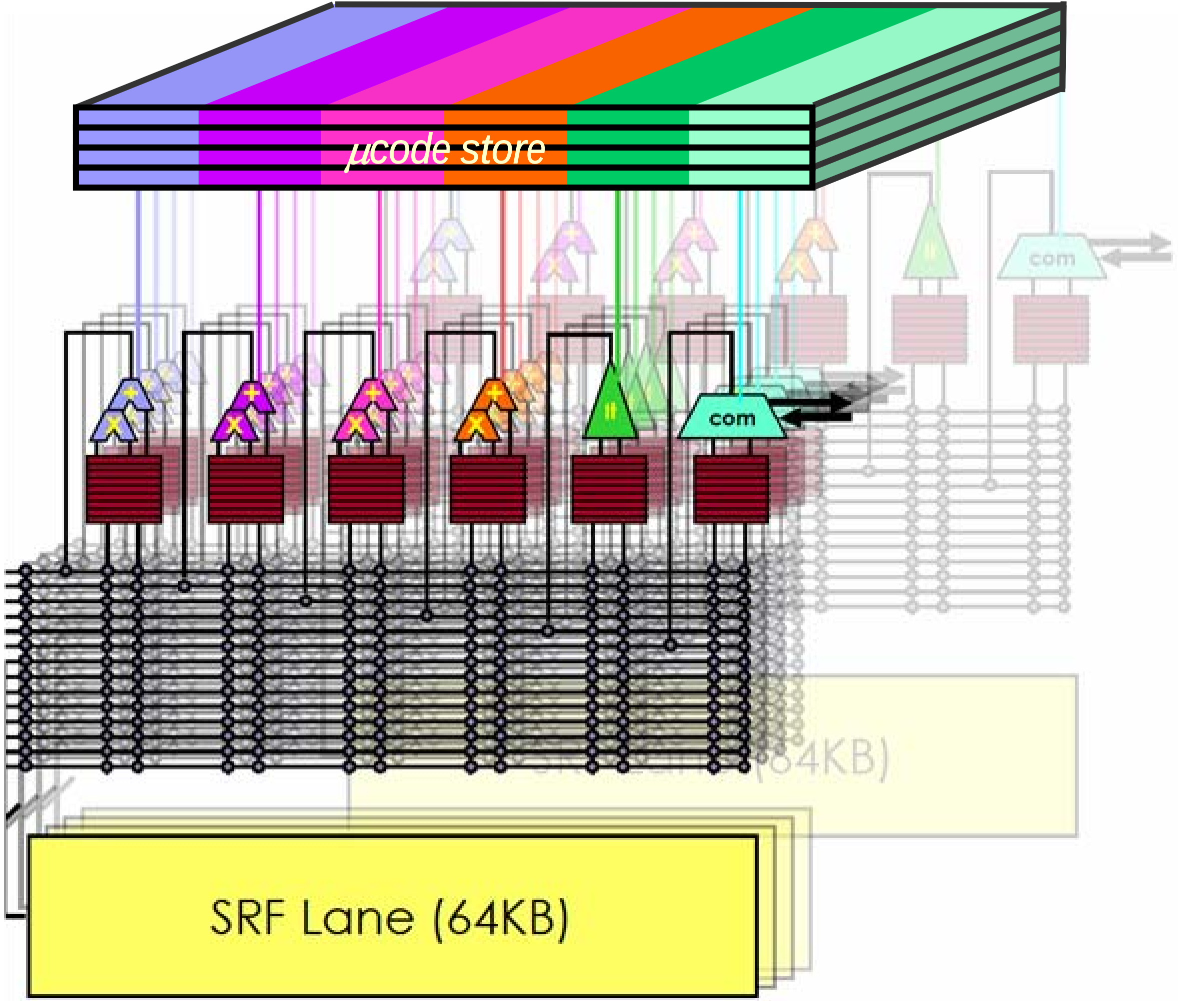
Stream Architecture Exploits ILP and DLP for Efficient Instruction Supply



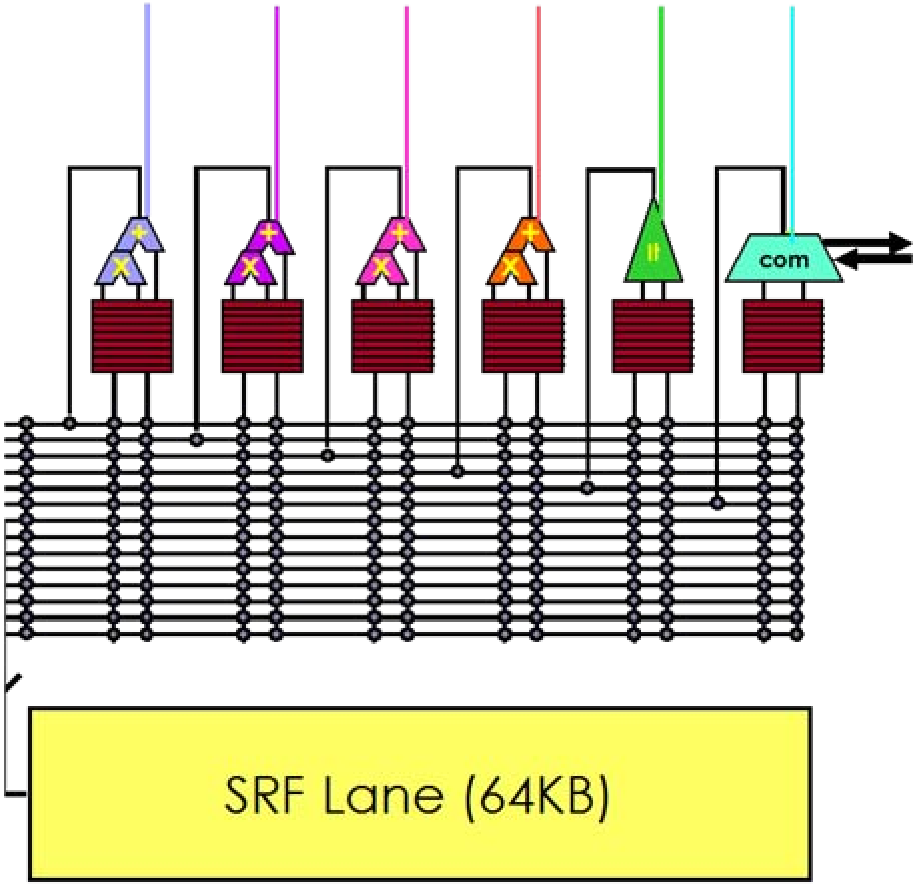


Stream Architecture Exploits ILP and DLP for Efficient Instruction Supply

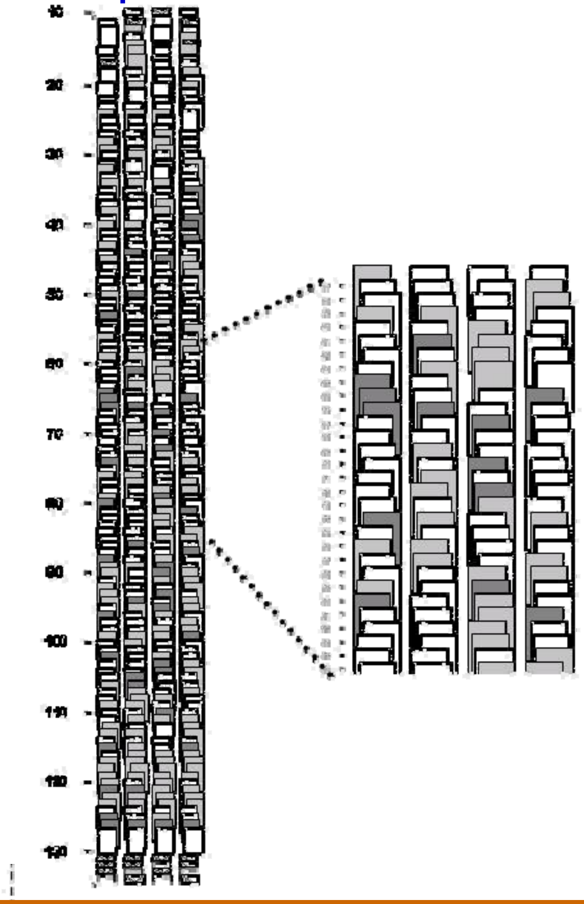




Compiler Optimizes VLIW Kernel Scheduling



Optimized schedule



Merrimac decouples memory and execution enables static optimization and reduces hardware

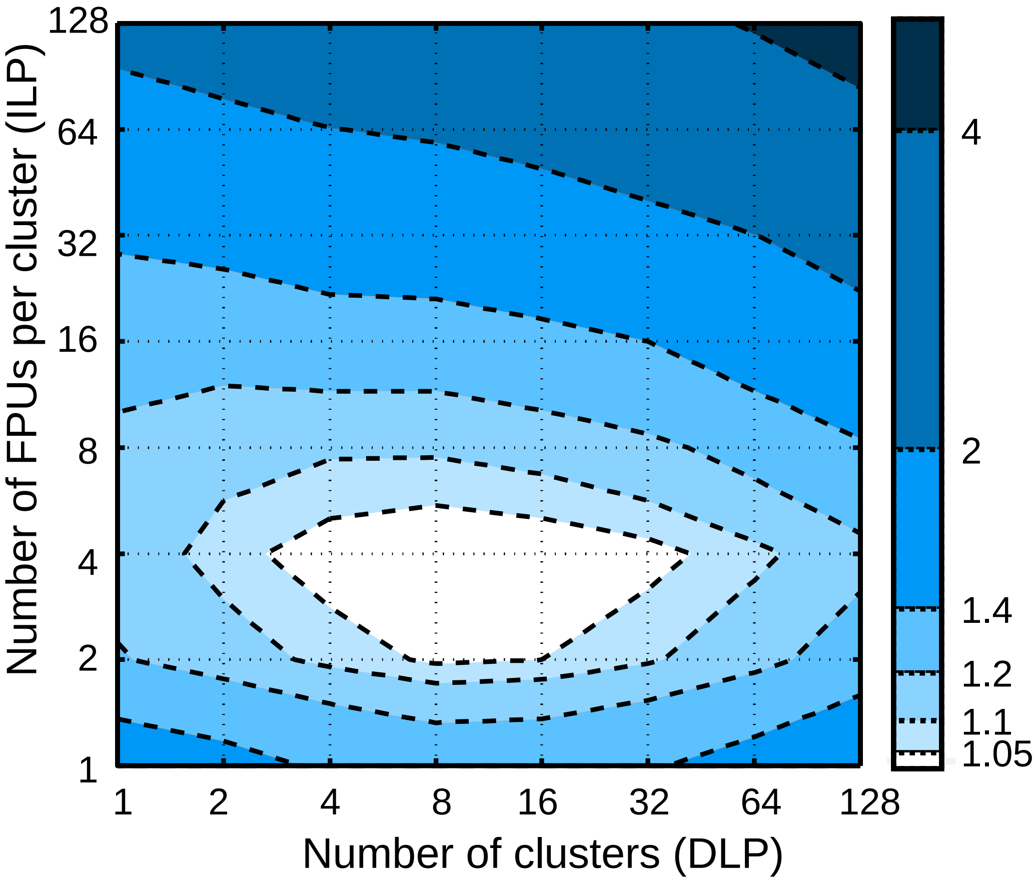
Parallelism Tradeoffs and Tuning Opportunities

- Applications
  - **Throughput oriented** vs. real-time constraint
  - Strong vs. **weak** scaling
  - **Regular vs. irregular**
  - Dynamic / **(practically-)static** datasets
- Hardware
  - DLP: **SIMD**, short vectors
  - ILP: **VLIW / execution pipeline**, OoO
  - TLP: **MIMD**, SMT (style)
  - Communication options
    - **Partial switches, direct sequencer-sequencer switch**

Hardware models for some options, active research on other options and performance models

Heat-map (Area per FPU) – 64 bit

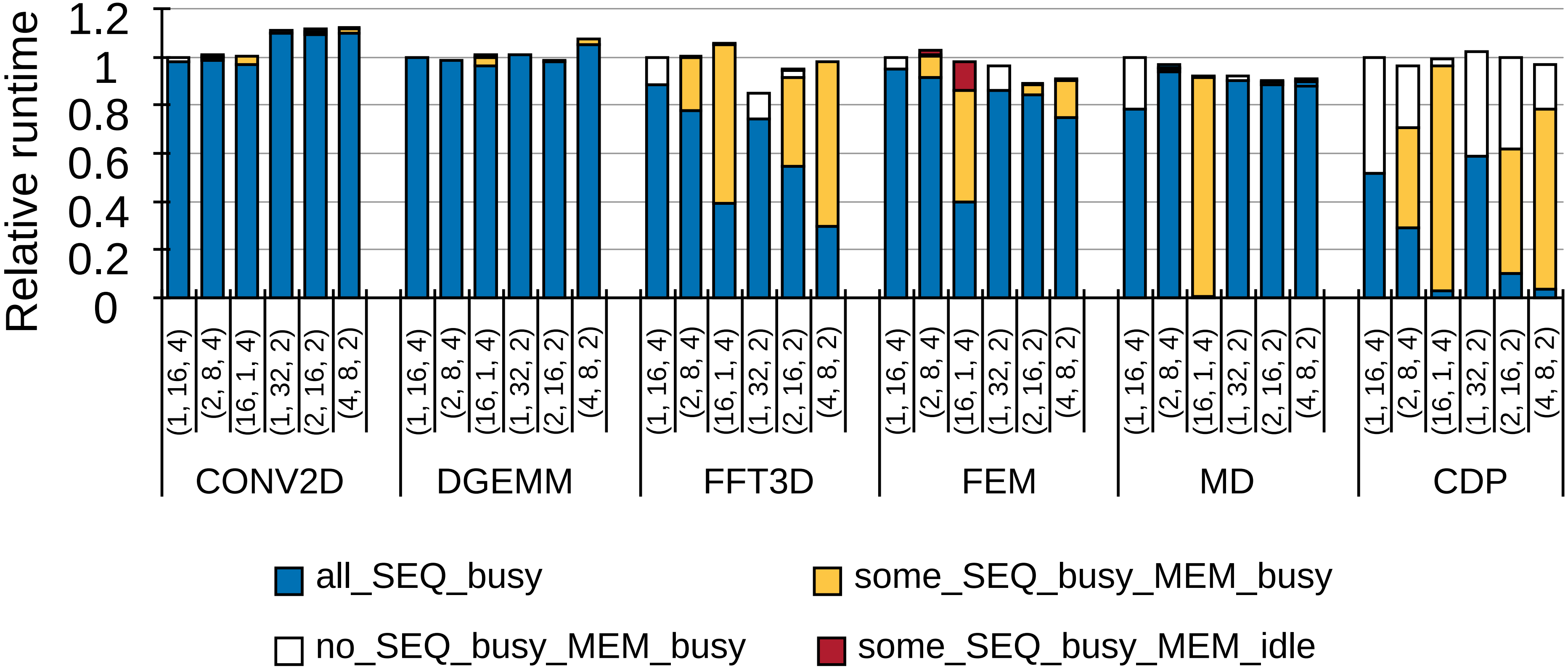
Area overhead of intra-cluster switches



Area overhead of an

Many reasonable hardware options for 64-bit

Application Performance

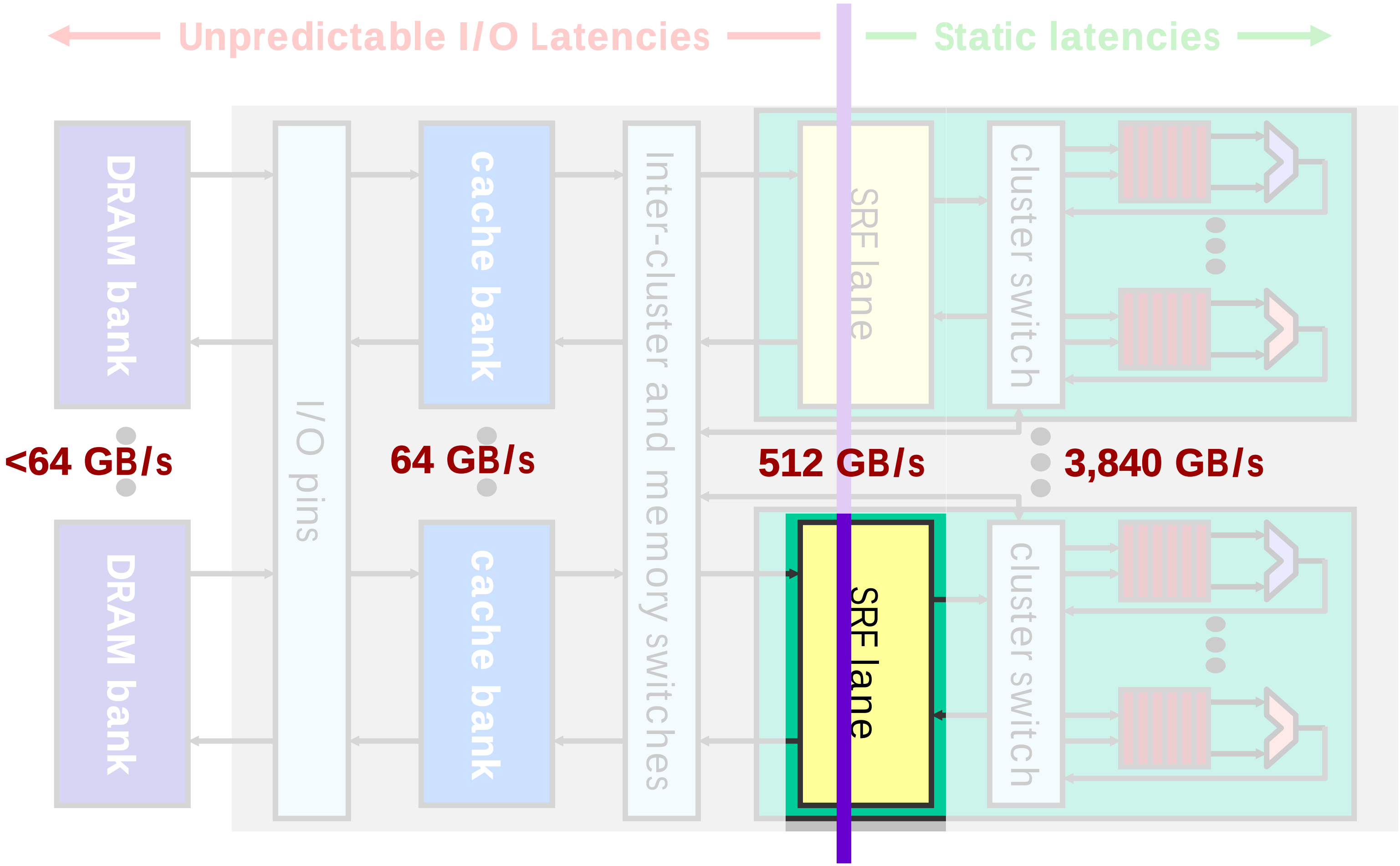


Small performance differences for "good streaming" applications

Outline

- Summary of previous lecture:
  - Parallelism
  - Locality
  - Hierarchical control and scheduling
  - Throughput oriented I/O
- Compute cluster
- **SRF**
- Stream memory system

SRF Decouples Execution from Memory



<64 GB/s

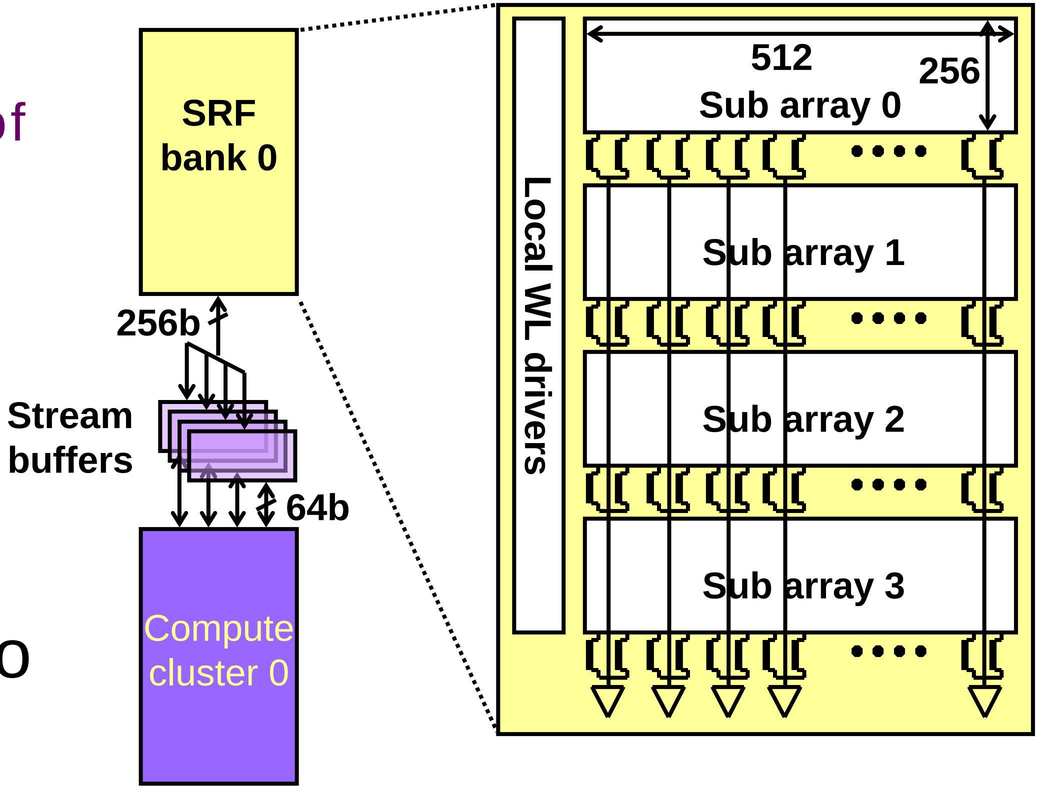
64 GB/s

512 GB/s

3,840 GB/s

SRF Sequential Access

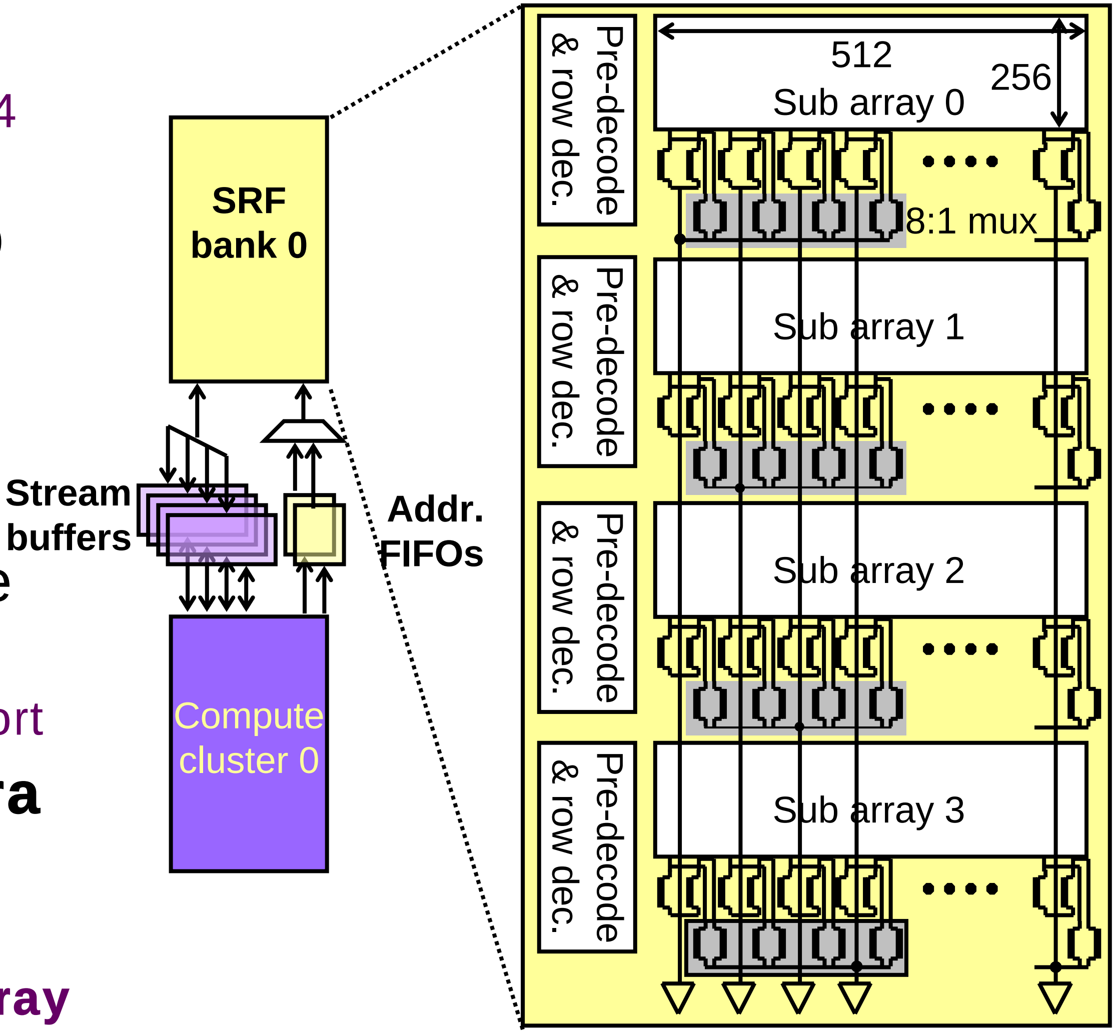
- Single ported memory
  - Efficient wide access of 4 contiguous words
- Implemented using sub arrays
  - Reduced access time
  - Reduced power
- Stream-buffers match bandwidth to compute needs
  - Time multiplex the SRF port



Wide single port time multiplexed by stream buffers

In-lane Indexing Almost Free

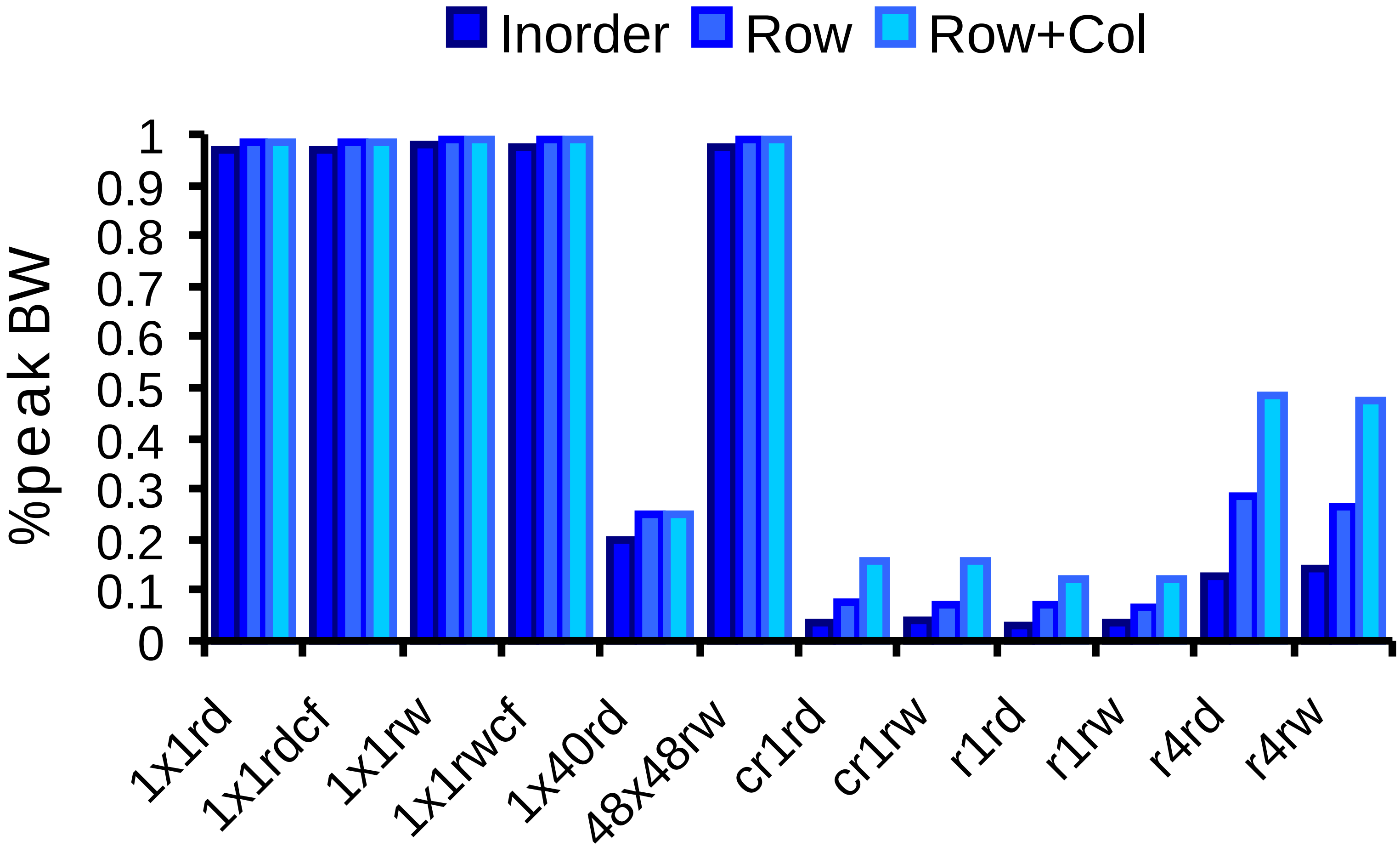
- Single ported memory
  - Efficient wide access of 4 contiguous words
- Implemented using sub arrays
  - Reduced access time
  - Reduced power
- Stream-buffers match bandwidth to compute needs
  - Time multiplex the SRF port
- **Indexed SRF at low extra cost**
  - **8:1 MUX in sub-arrays**
  - **Row decoder per sub-array**



Outline

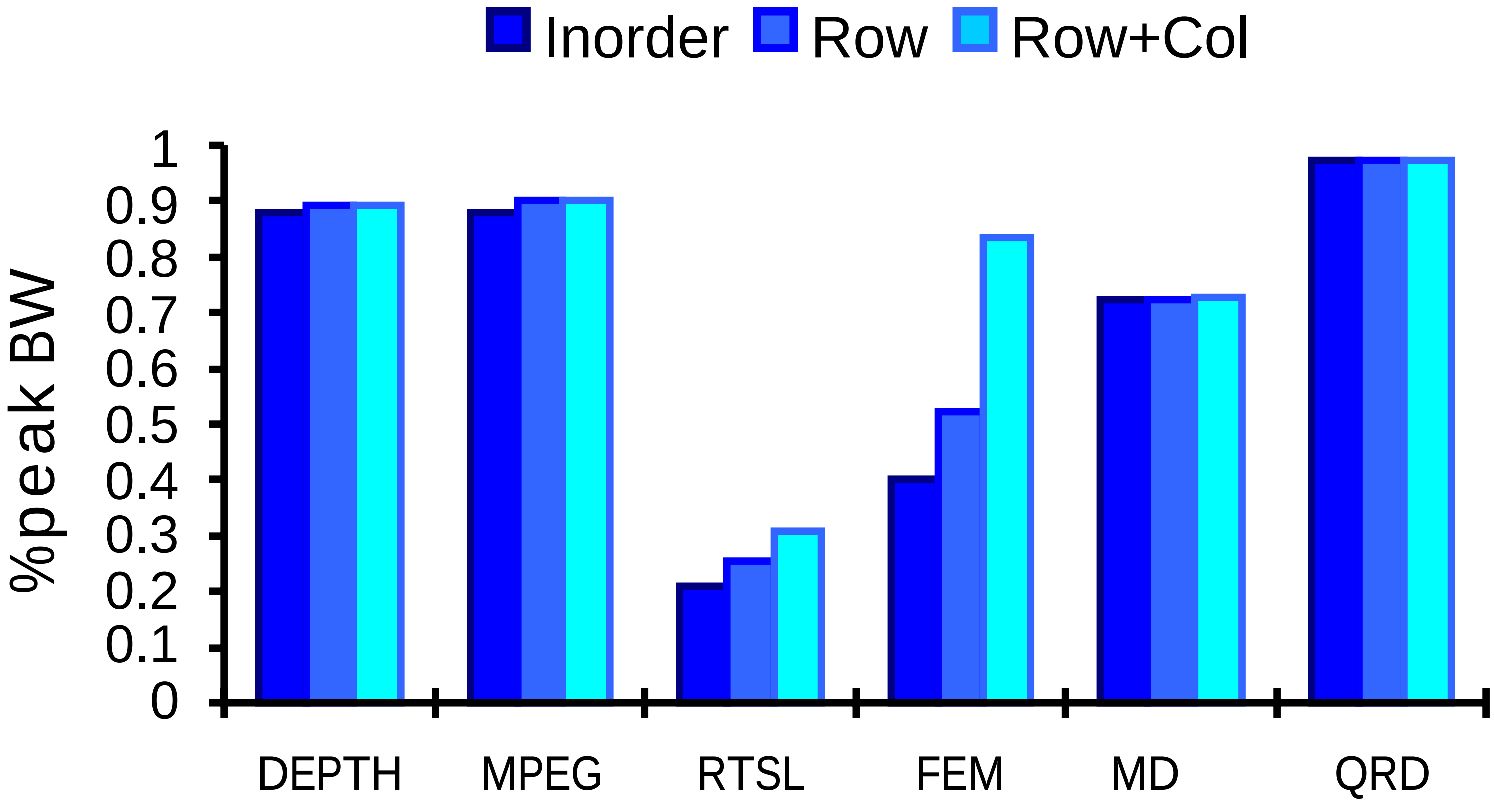
- Summary of previous lecture:
  - Parallelism
  - Locality
  - Hierarchical control and scheduling
  - Throughput oriented I/O
- Compute cluster
- SRF
- **Stream memory system**

Streaming Memory Systems



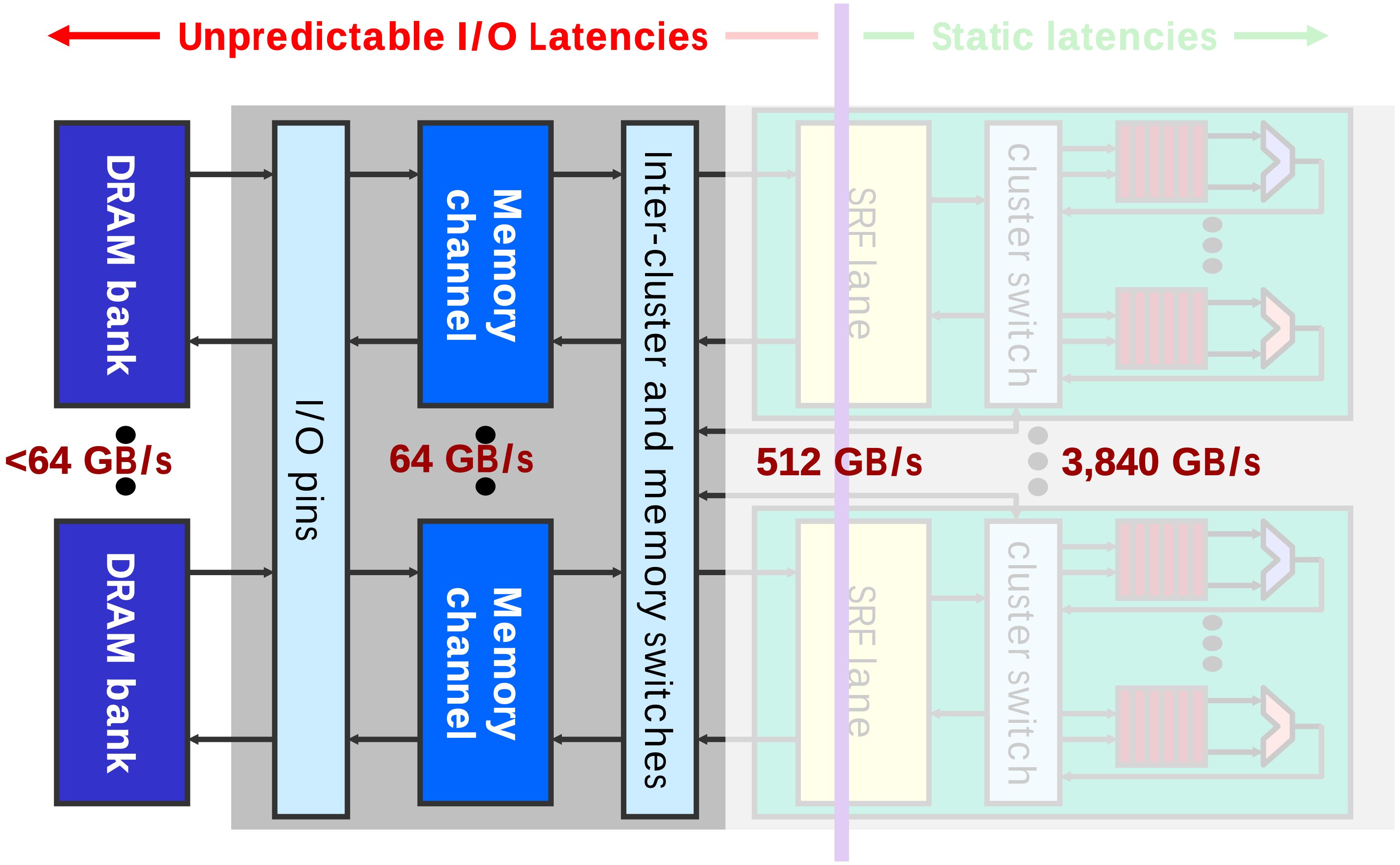
DRAM systems are very sensitive to access pattern, Throughput-oriented memory system helps

Streaming Memory Systems Help



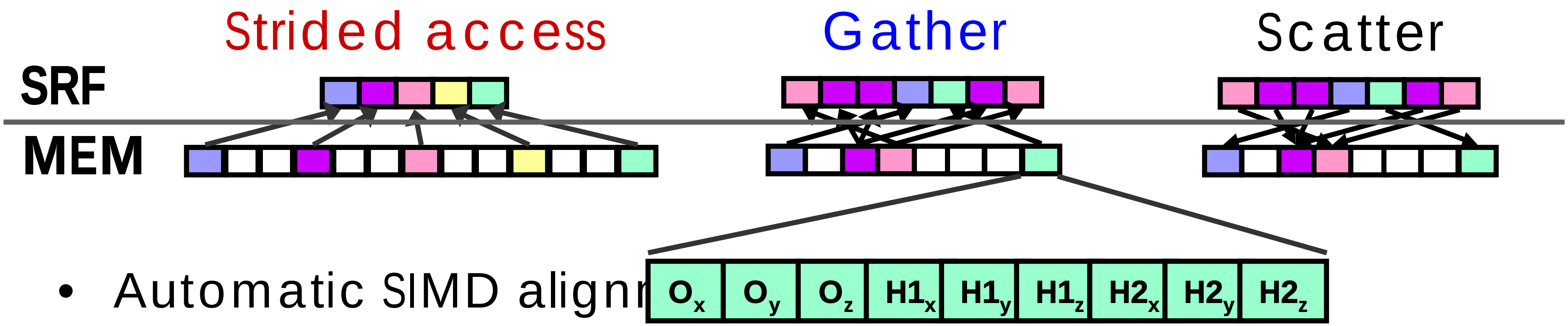
Capable memory system even more important for applications

SRF Decouples Execution from Memory



Streaming Memory Systems

- Bulk stream loads and stores
  - Hierarchical control
- Expressive and effective addressing modes
  - Can’t afford to waste memory bandwidth
  - Use hardware when performance is non-deterministic



- Automatic SIMD alignr
  - Makes SIMD trivial (SIMD ≠ short-vector)

Stream memory system helps the programmer and maximizes I/O throughput

- Rest of memory system details delayed for another lecture

More Hardware Features

- Take advantage of bulk execution model
  - Cheap OOO scheduling of kernels and stream memory operations (more when we talk about software system)
  - High-throughput exception handling
  - Low-cost fault-tolerance