EE382V: Principles in Computer Architecture
Parallelism and Locality
Fall 2008

Lecture 6 – HW and SW Parallelism

Mattan Erez

The University of Texas at Austin

Outline

- Parallel HW (multiple ALUs)
  - Analyze by shared resources
  - Analyze by synch/comm mechanisms
  - ILP, DLP, and TLP organizations
- Parallelism in SW
  - ILP/DLP/TLP?
- Parallel Programming
  - Design patterns

Parallel Execution

- Concurrency
  - what are the multiple resources?
- Communication
  - and storage
- Synchronization
- what is being ***shared*** ?
- What is being ***partitioned*** ?

Pipelining Summary

- Pipelining is using parallelism to hide latency
  - Do useful work while waiting for other work to finish
- Multiple parallel components, not multiple instances of same component
- Examples:
  - Execution pipeline
  - Memory pipelines
    - Issue multiple requests to memory without waiting for previous requests to complete
  - Software pipelines
    - Overlap different software blocks to hide latency: **computation/communication**

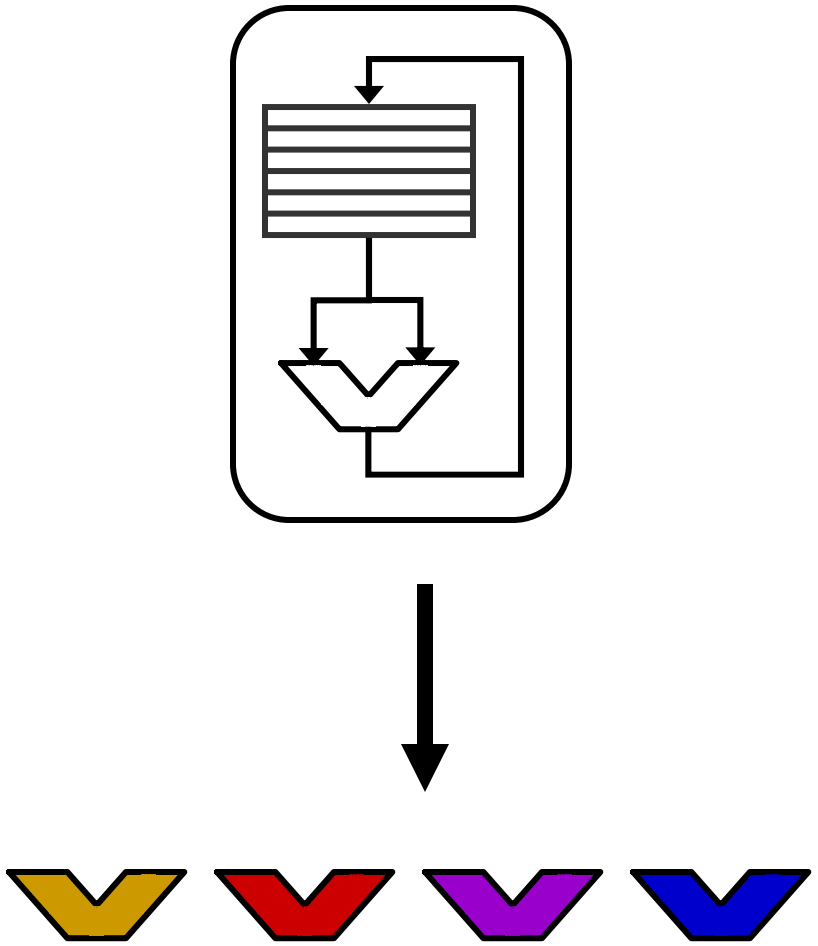
Resources in a parallel processor/system

- Execution
  - ALUs
  - Cores/processors
- Control
  - Sequencers
  - Instructions
  - OOO schedulers
- State
  - Registers
  - Memories
- Networks

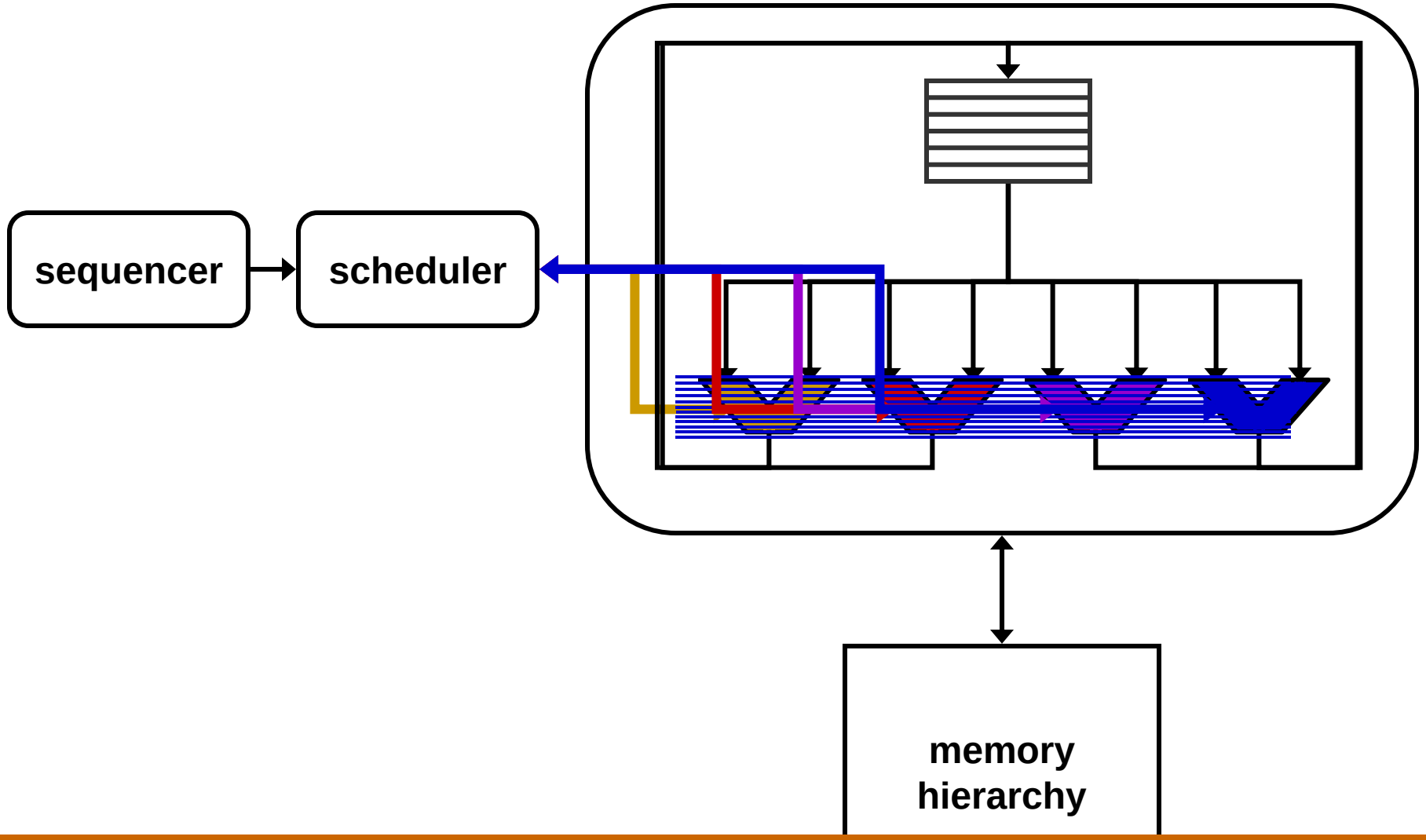
Communication and synchronization

- Synchronization
  - Clock – explicit compiler order
  - Explicit signals (e.g., dependences)
  - Implicit signals (e.g., flush/stall)
    - More for pipelining than multiple ALUs
- Communication
  - Bypass networks
  - Registers
  - Memory
  - Explicit (over some network)

Organizations for ILP (for multiple ALUs)

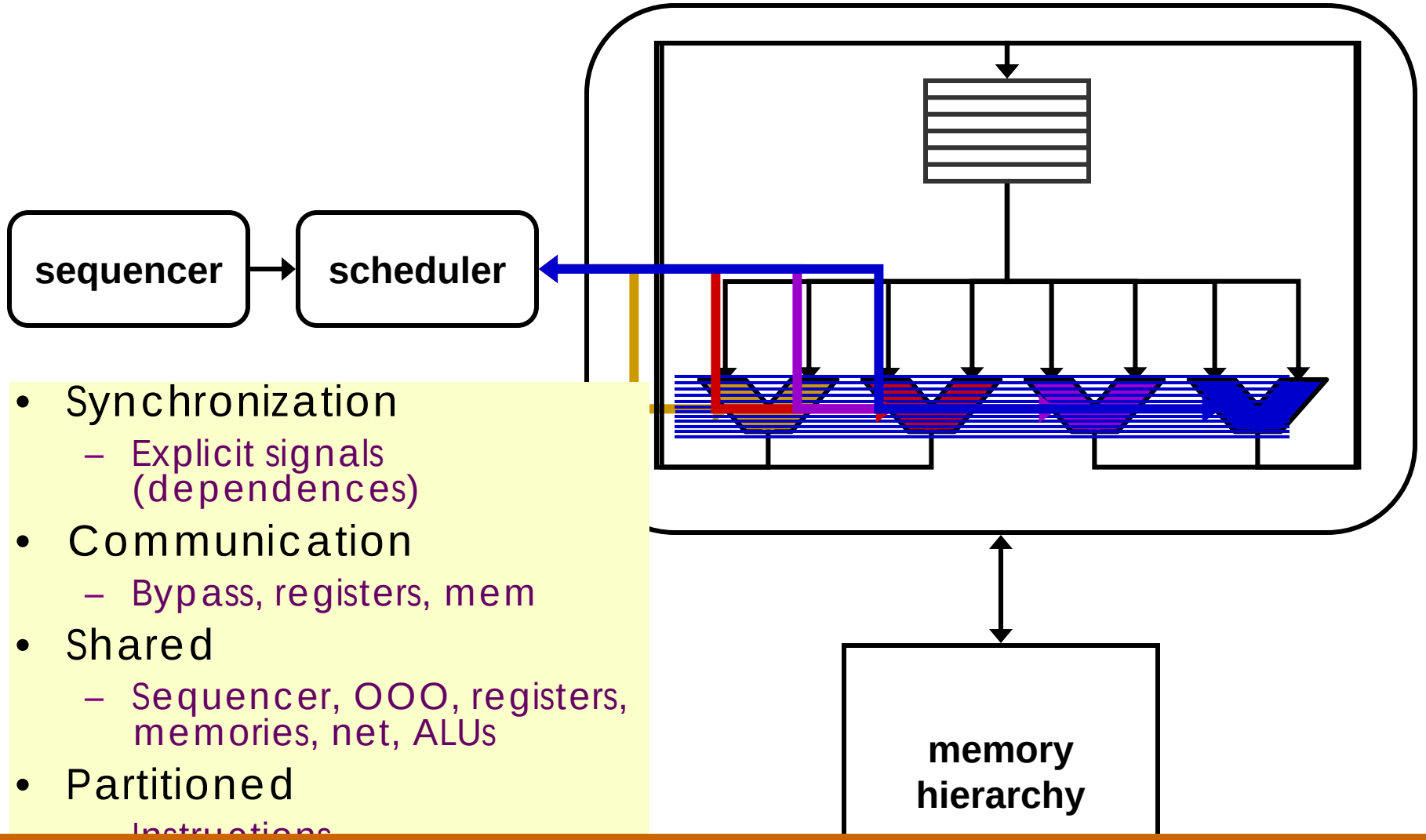


Superscalar (ILP for multiple ALUs)



How many ALUs?

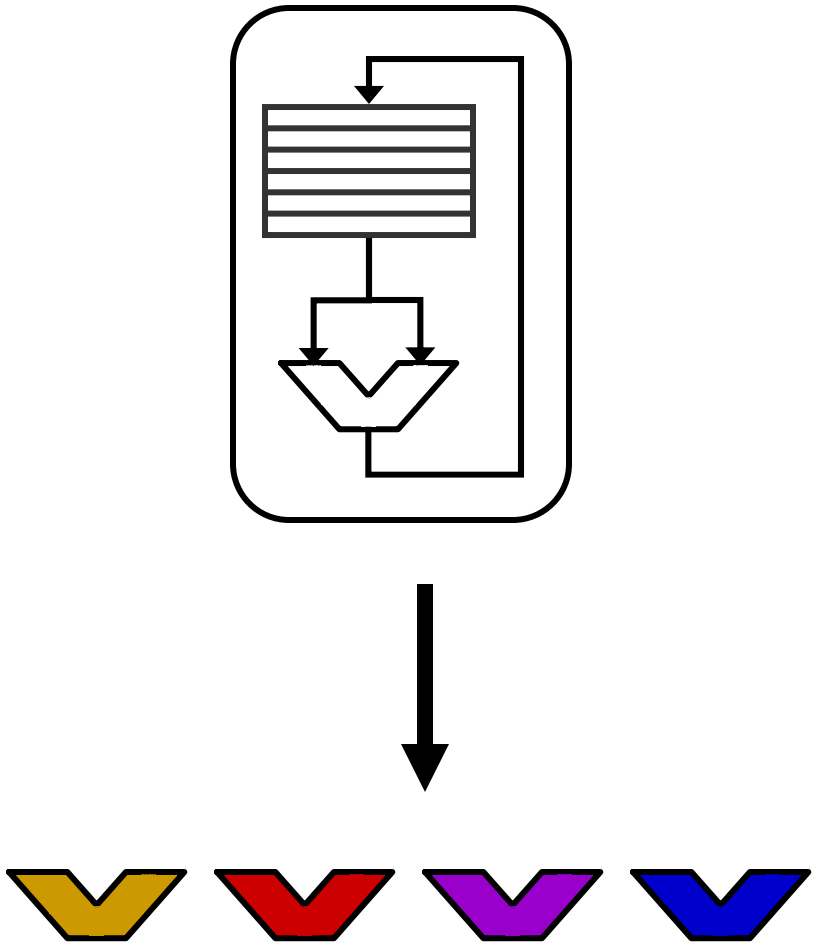
Superscalar (ILP for multiple ALUs)



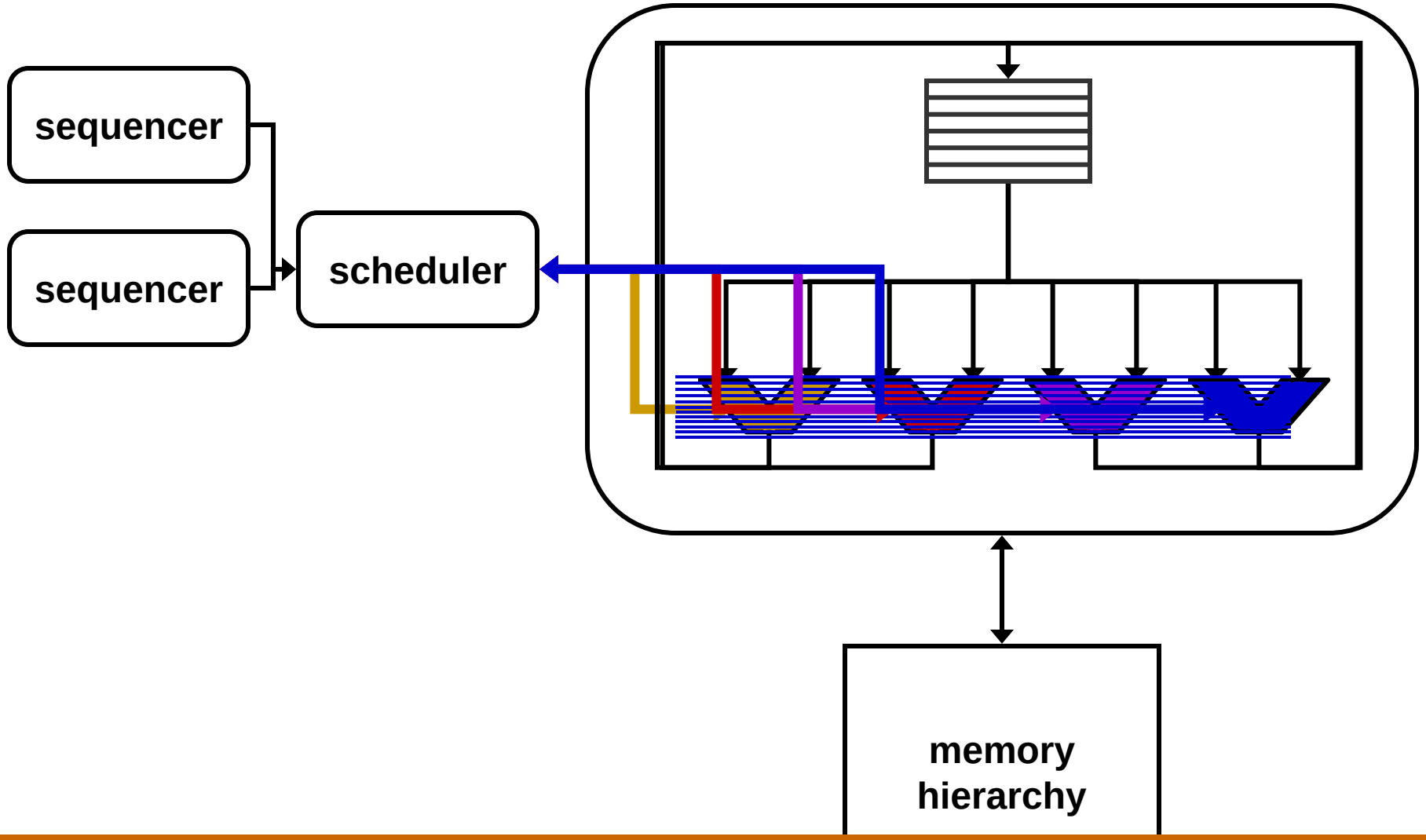
- Synchronization
  - Explicit signals (dependences)
- Communication
  - Bypass, registers, mem
- Shared
  - Sequencer, OOO, registers, memories, net, ALUs
- Partitioned
  - Instructions

How many ALUs?

SMT/TLS (ILP for multiple ALUs)

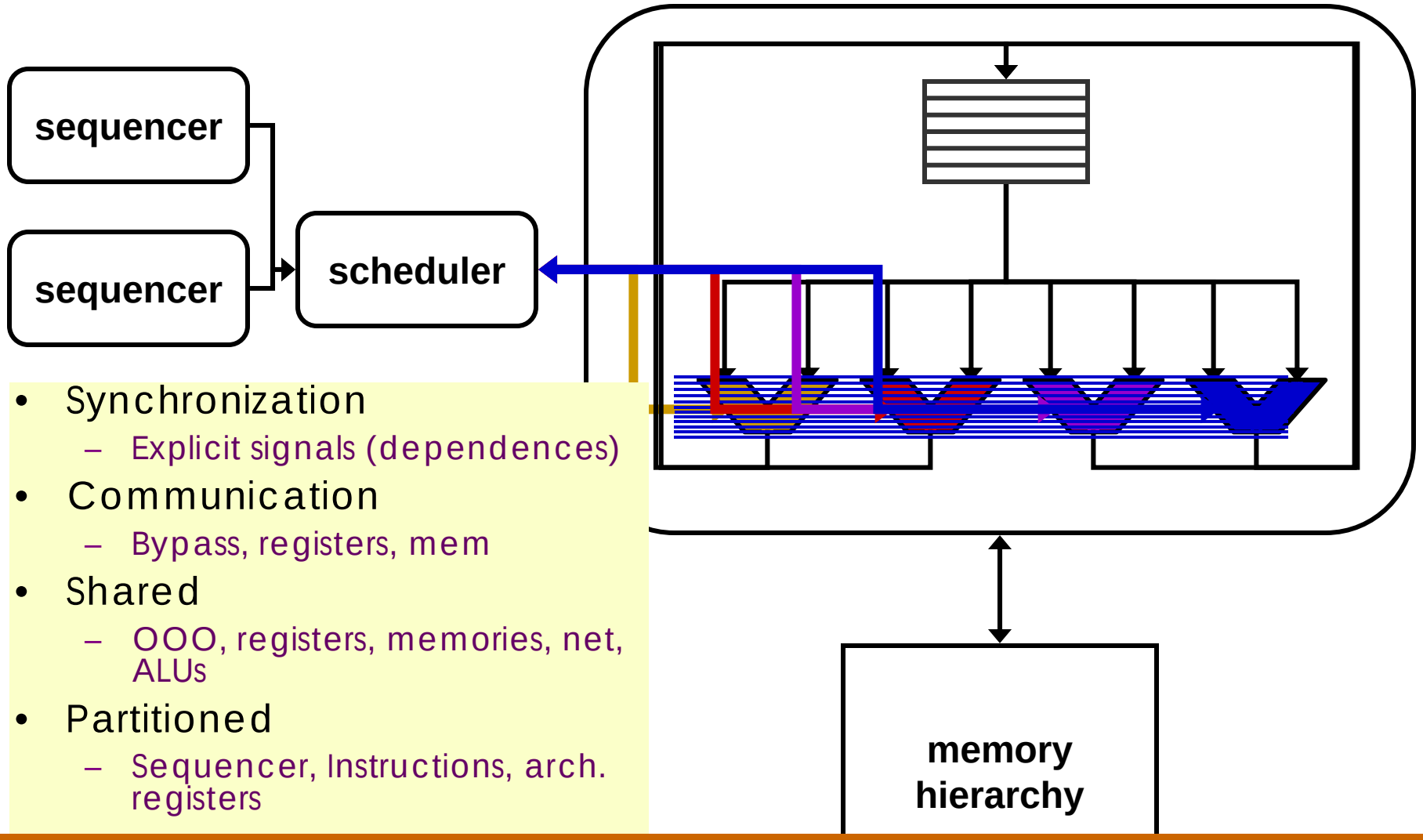


SMT/TLS (ILP for multiple ALUs)



Why is this ILP? How many threads?

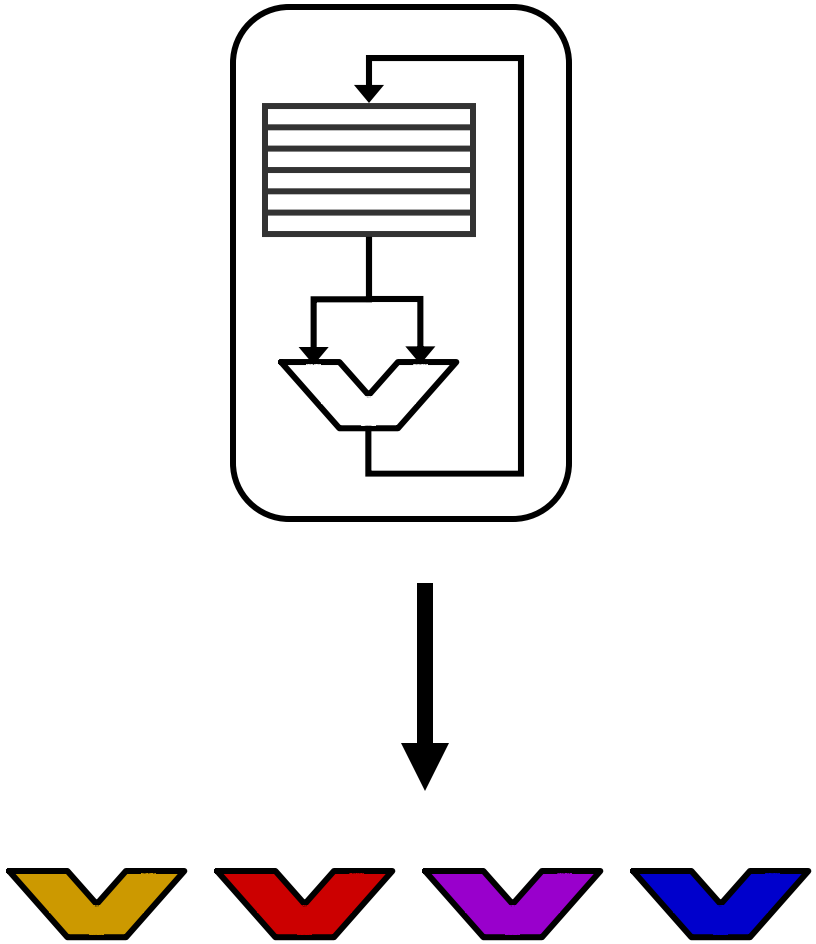
SMT/TLS (ILP for multiple ALUs)



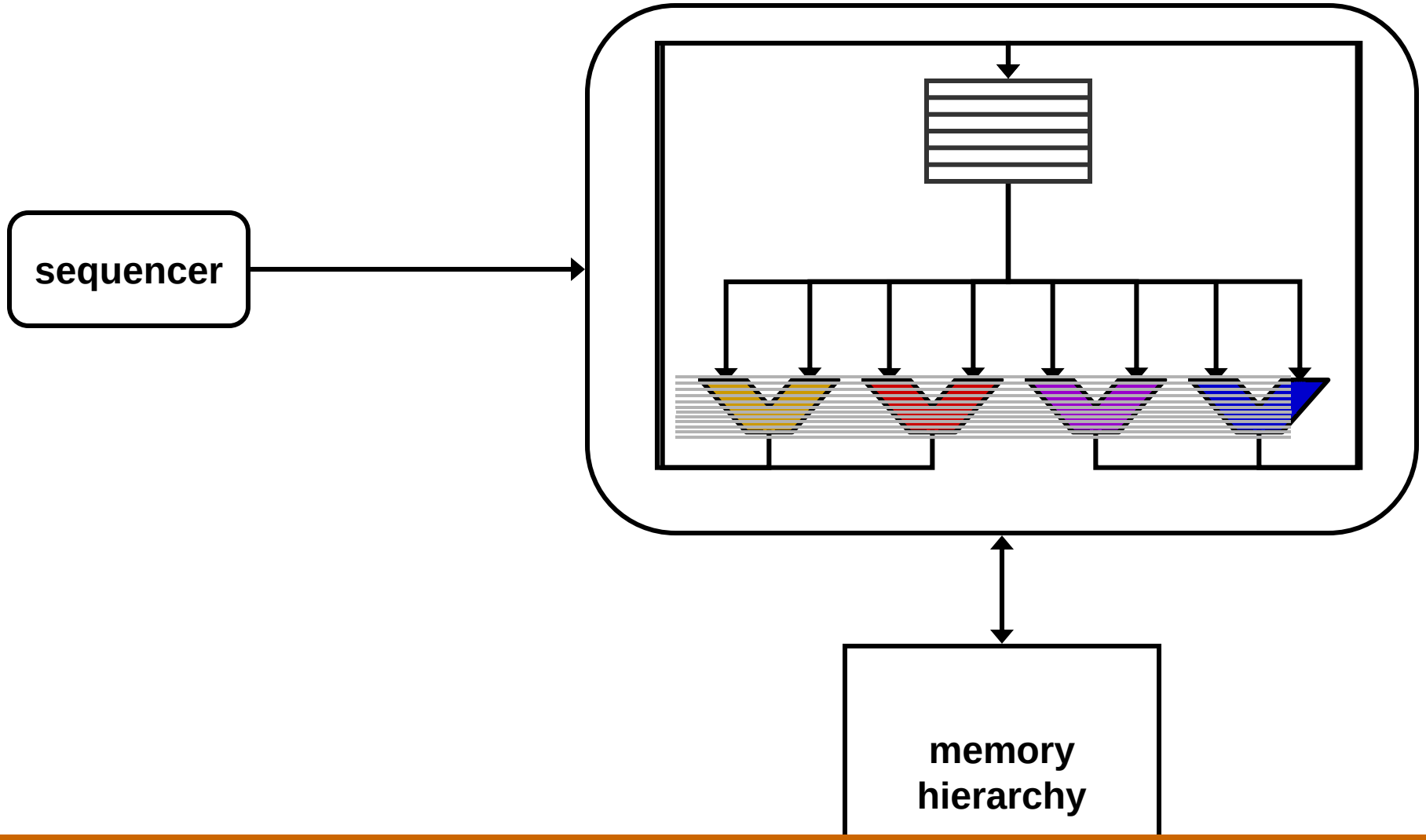
- Synchronization
  - Explicit signals (dependences)
- Communication
  - Bypass, registers, mem
- Shared
  - OOO, registers, memories, net, ALUs
- Partitioned
  - Sequencer, Instructions, arch. registers

Why is this ILP? How many threads?

VLIW (ILP for multiple ALUs)

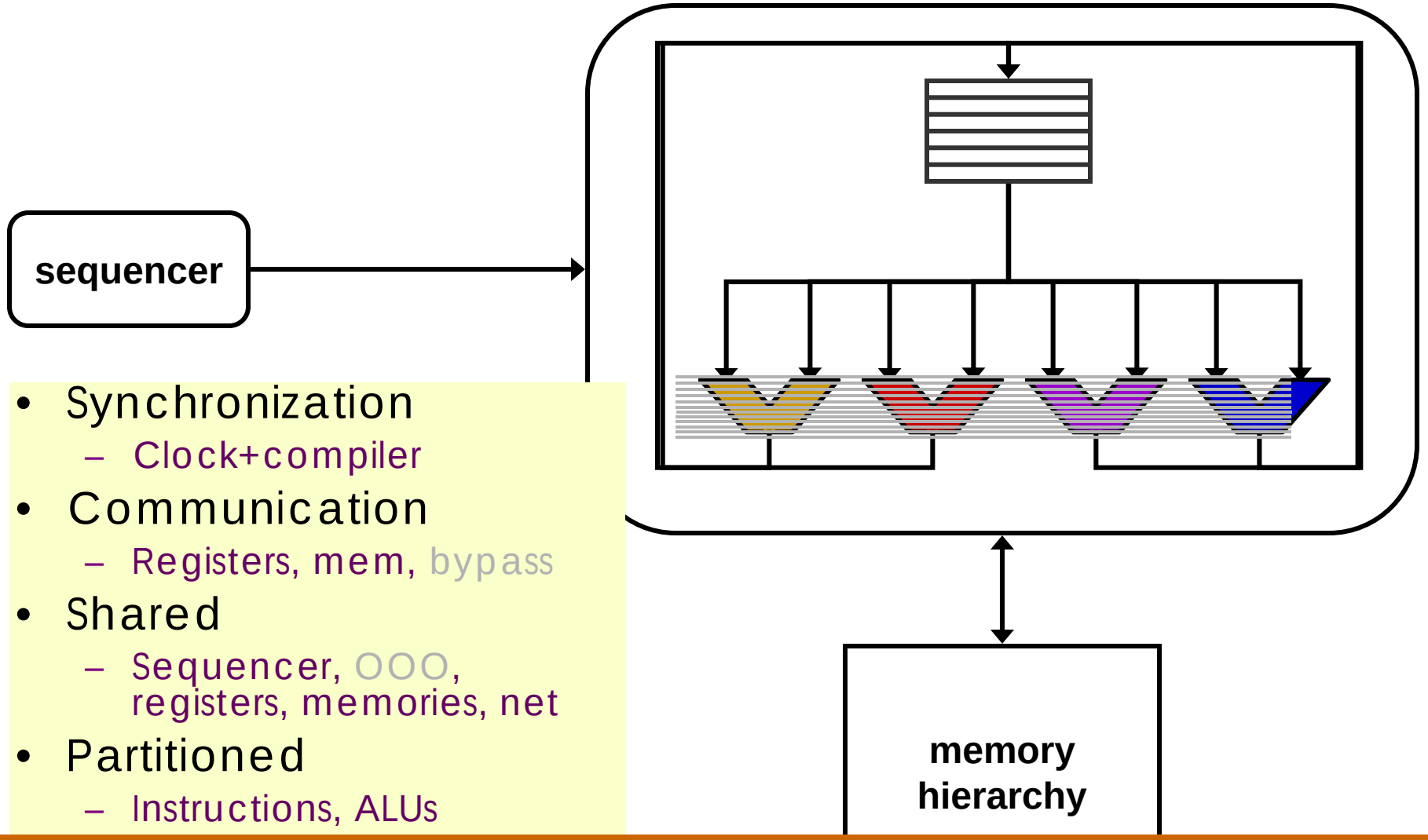


VLIW (ILP for multiple ALUs)



How many ALUs?

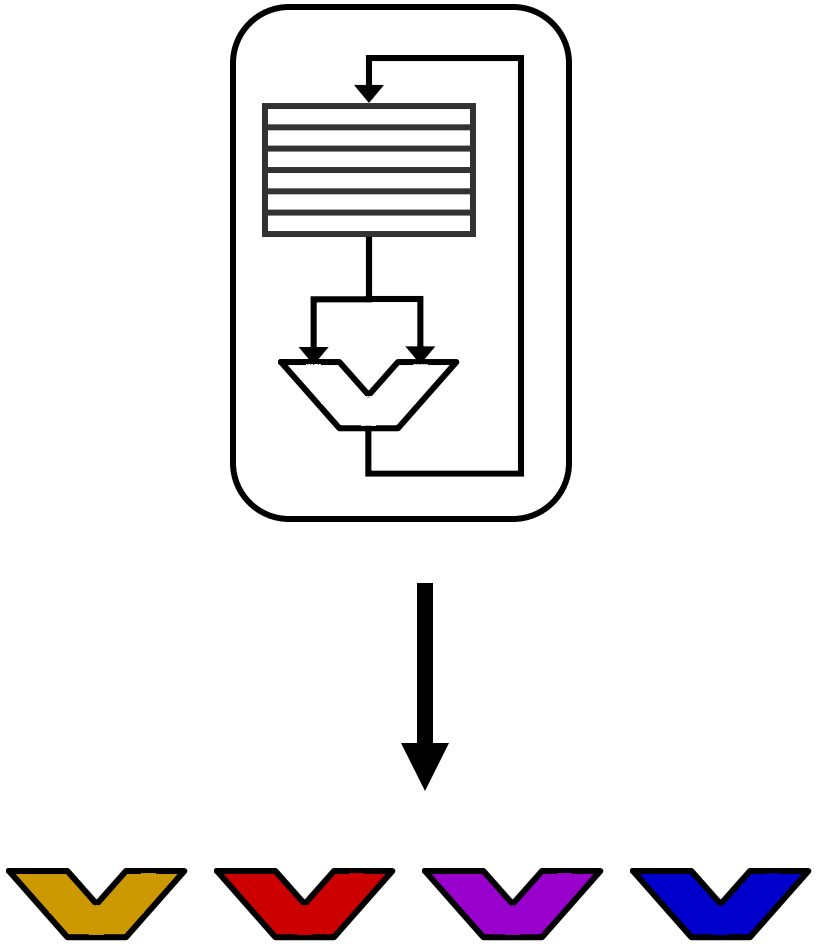
VLIW (ILP for multiple ALUs)



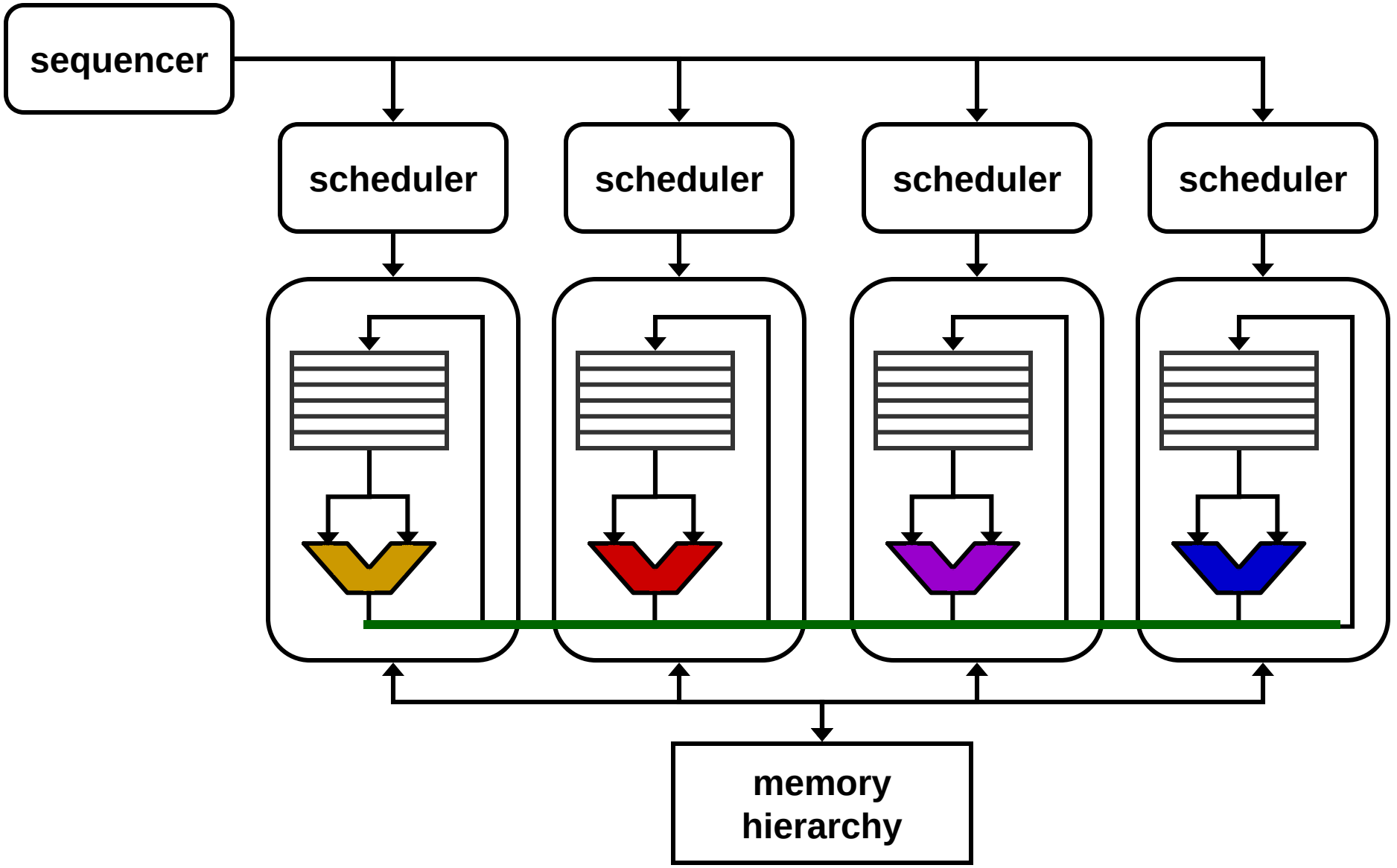
- Synchronization
  - Clock+compiler
- Communication
  - Registers, mem, bypass
- Shared
  - Sequencer, OOO, registers, memories, net
- Partitioned
  - Instructions, ALUs

How many ALUs?

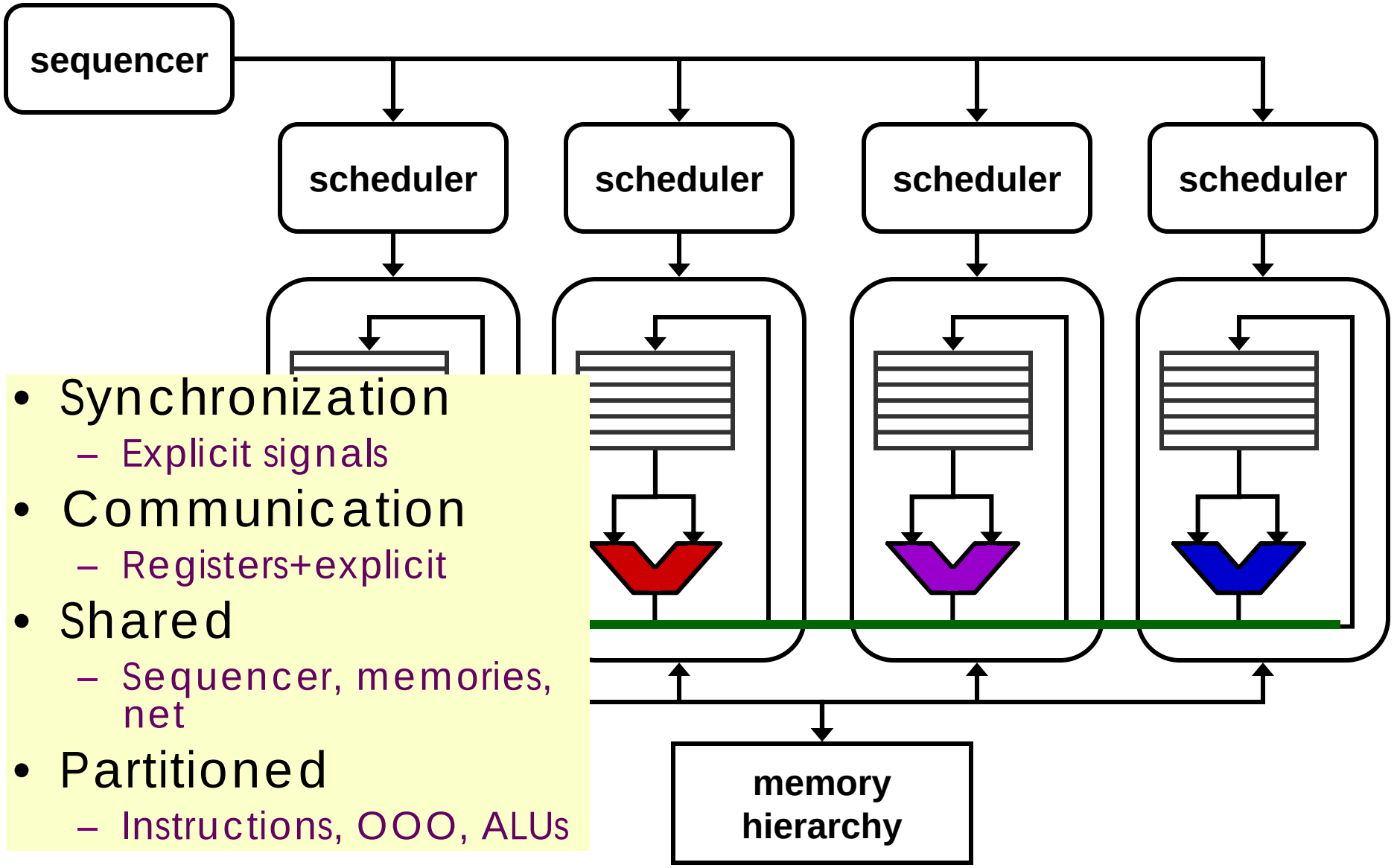
Explicit Dataflow (ILP for multiple ALUs)



Explicit Dataflow (ILP for multiple ALUs)

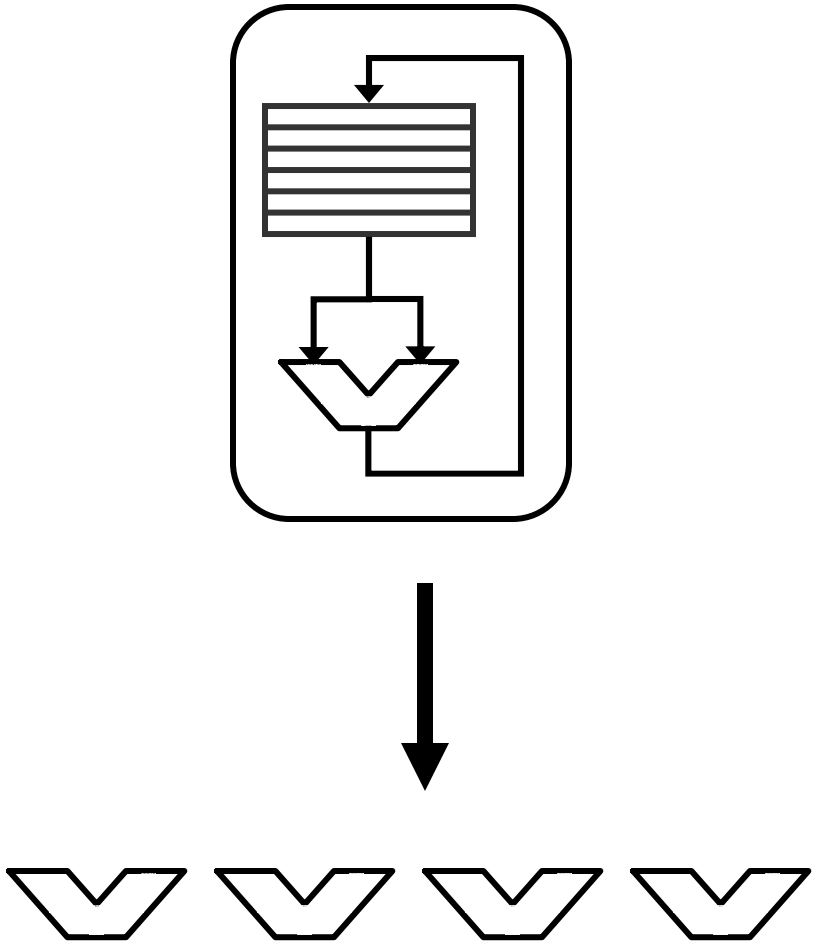


Explicit Dataflow (ILP for multiple ALUs)



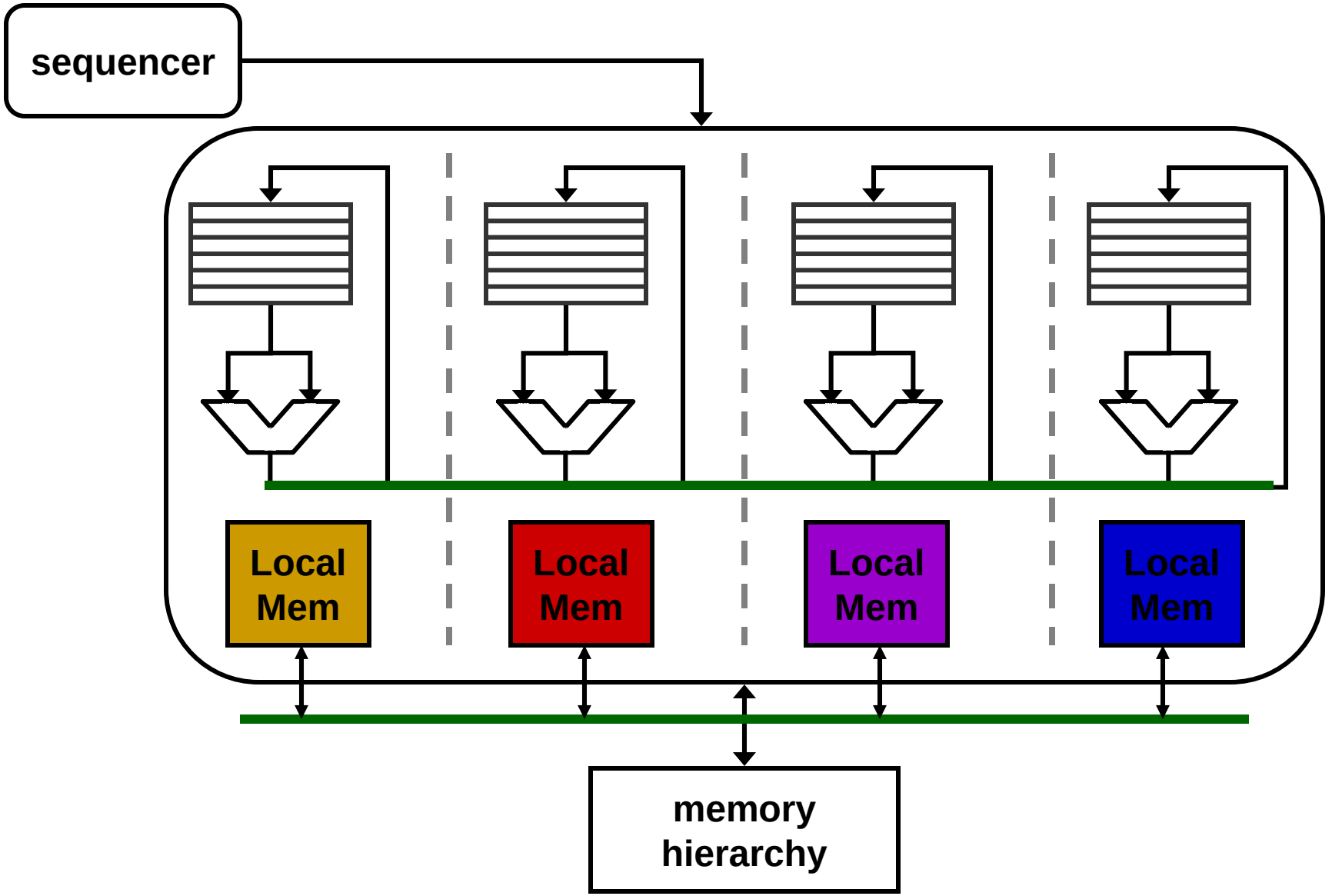
- Synchronization
  - Explicit signals
- Communication
  - Registers+explicit
- Shared
  - Sequencer, memories, net
- Partitioned
  - Instructions, OOO, ALUs

DLP for multiple ALUs

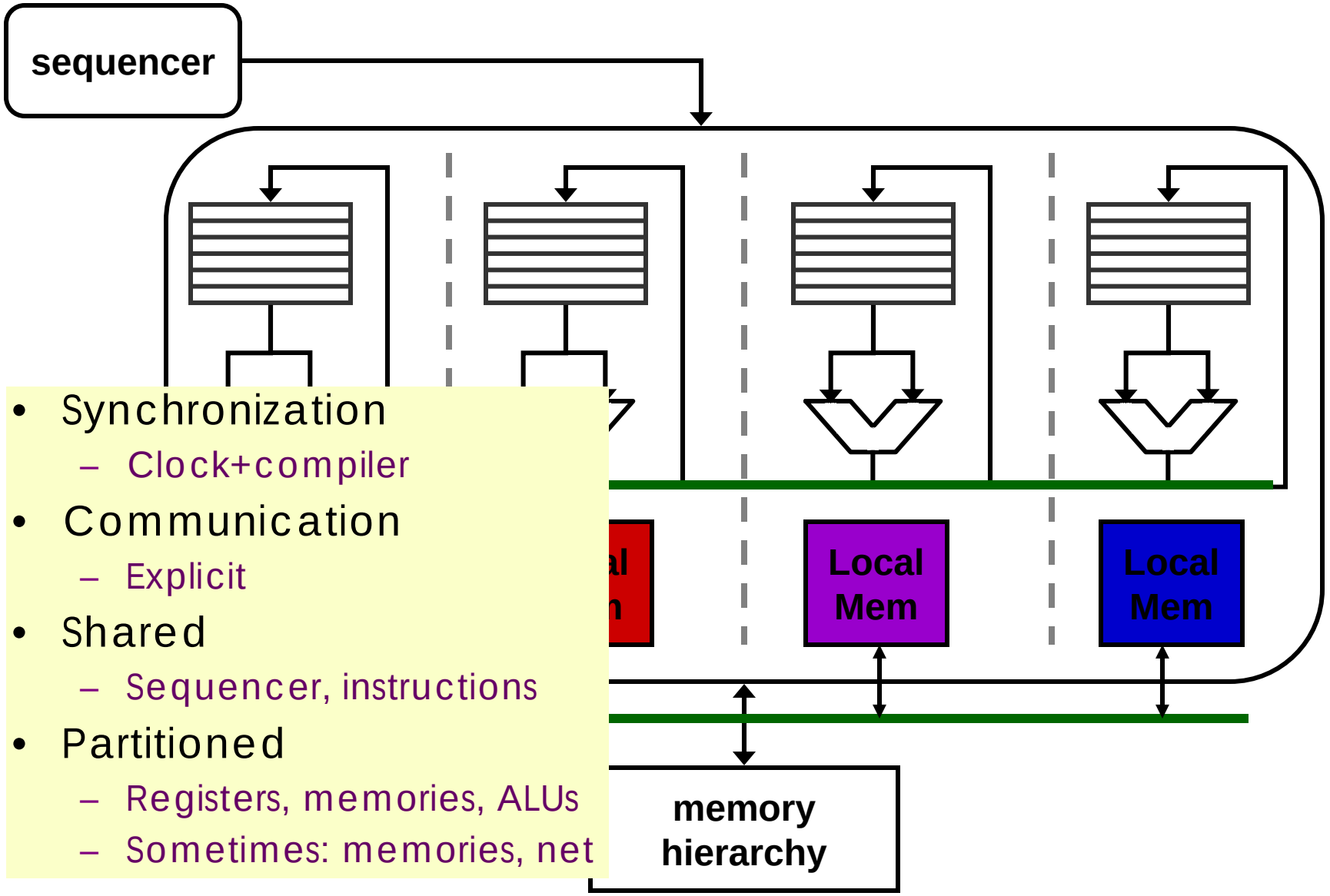


From HW – this is SIMD

SIMD (DLP for multiple ALUs)

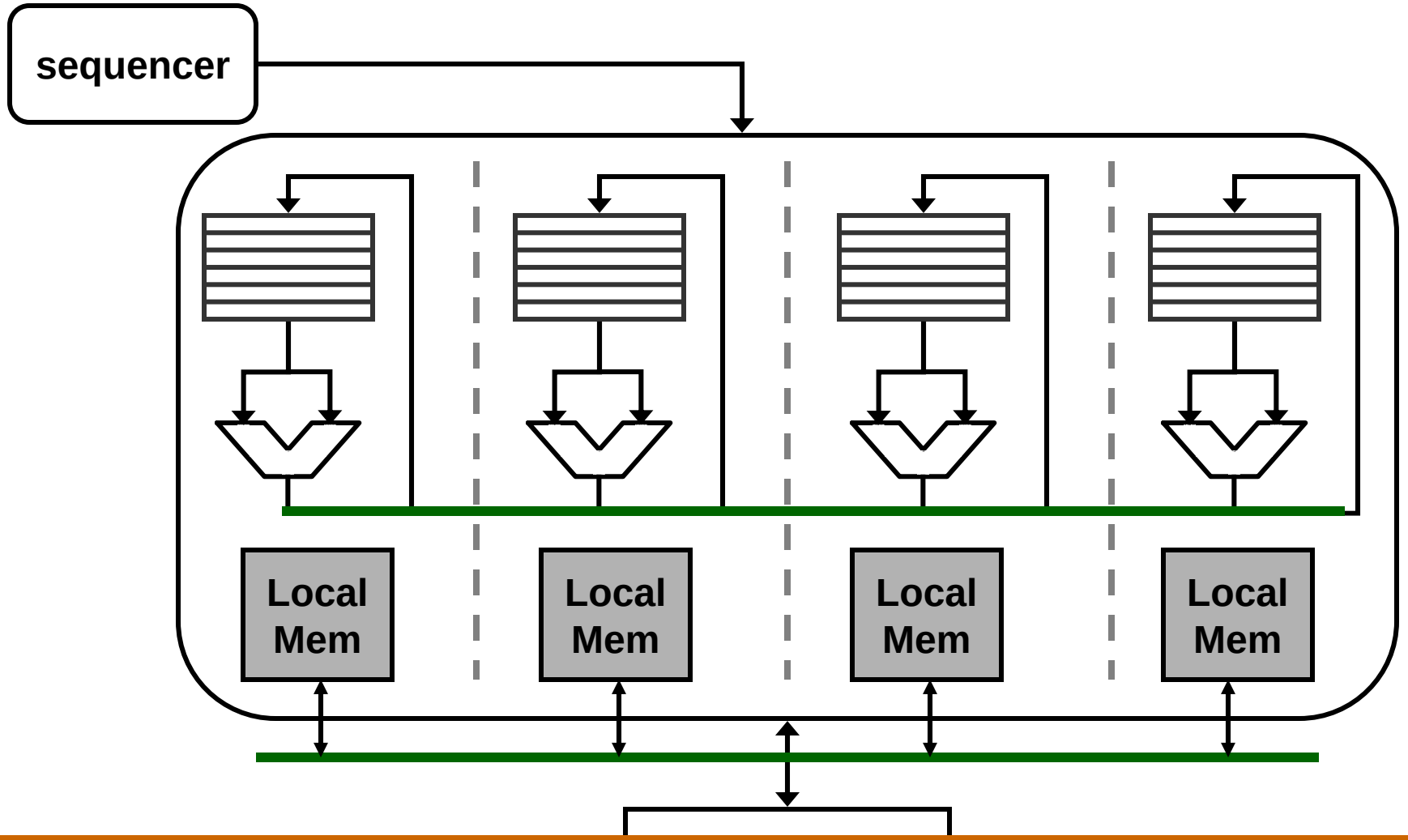


SIMD (DLP for multiple ALUs)



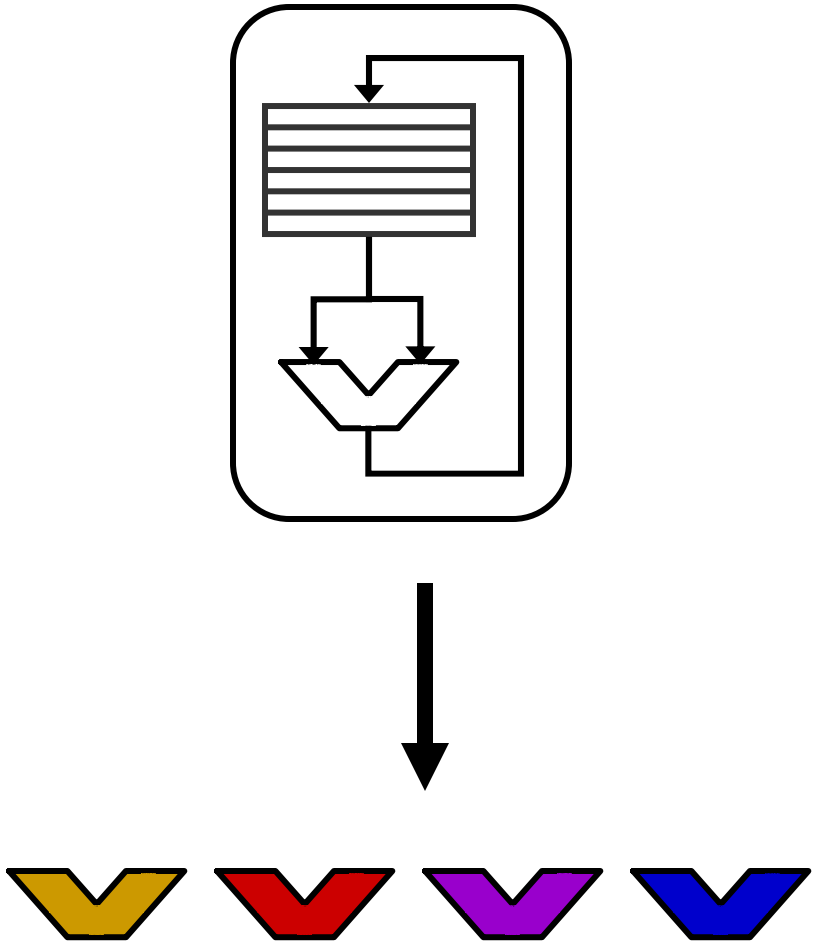
- Synchronization
  - Clock+compiler
- Communication
  - Explicit
- Shared
  - Sequencer, instructions
- Partitioned
  - Registers, memories, ALUs
  - Sometimes: memories, net

Vectors (DLP for multiple ALUs)

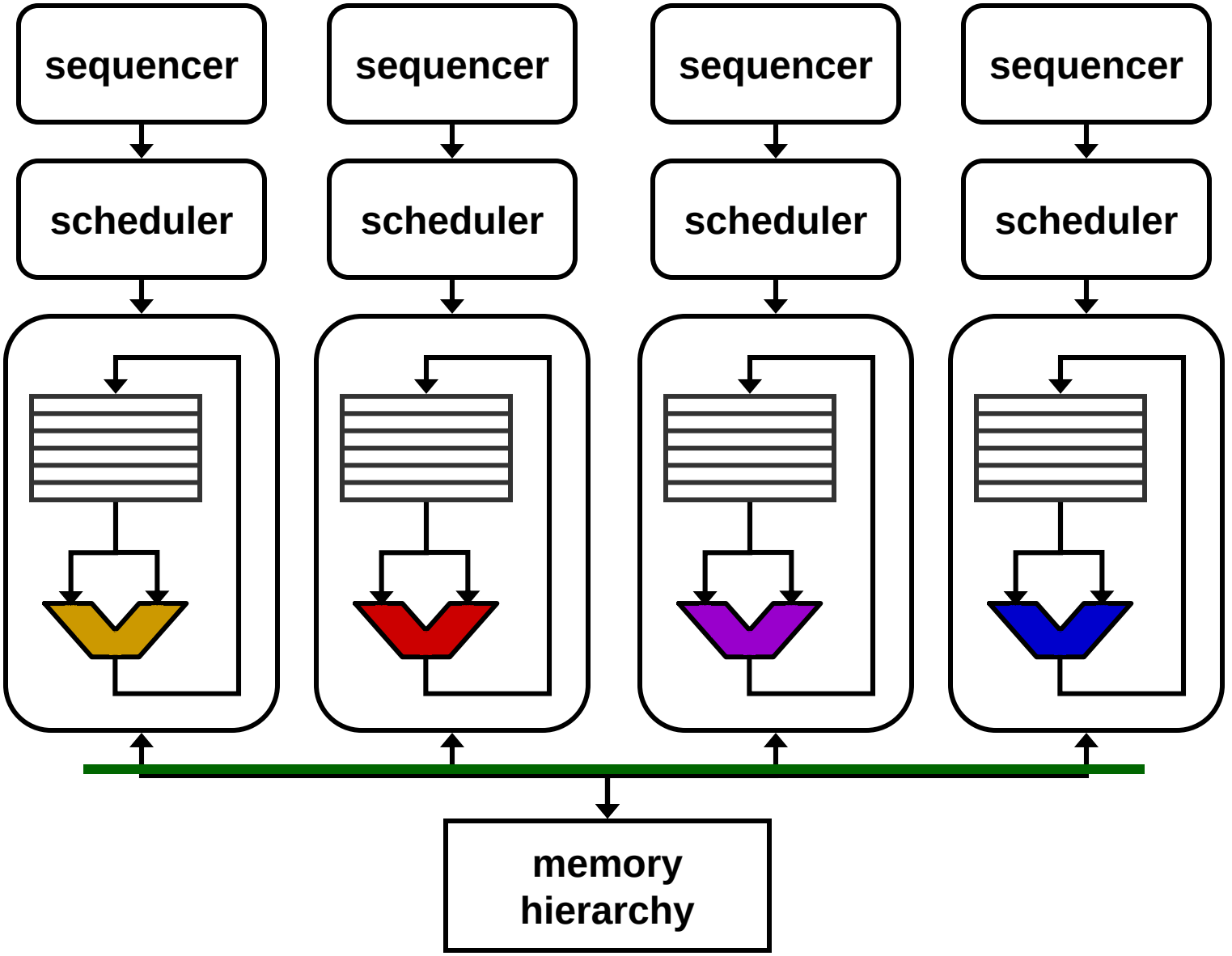


Vectors: memory addresses are part of single-instruction and not part of multiple-data

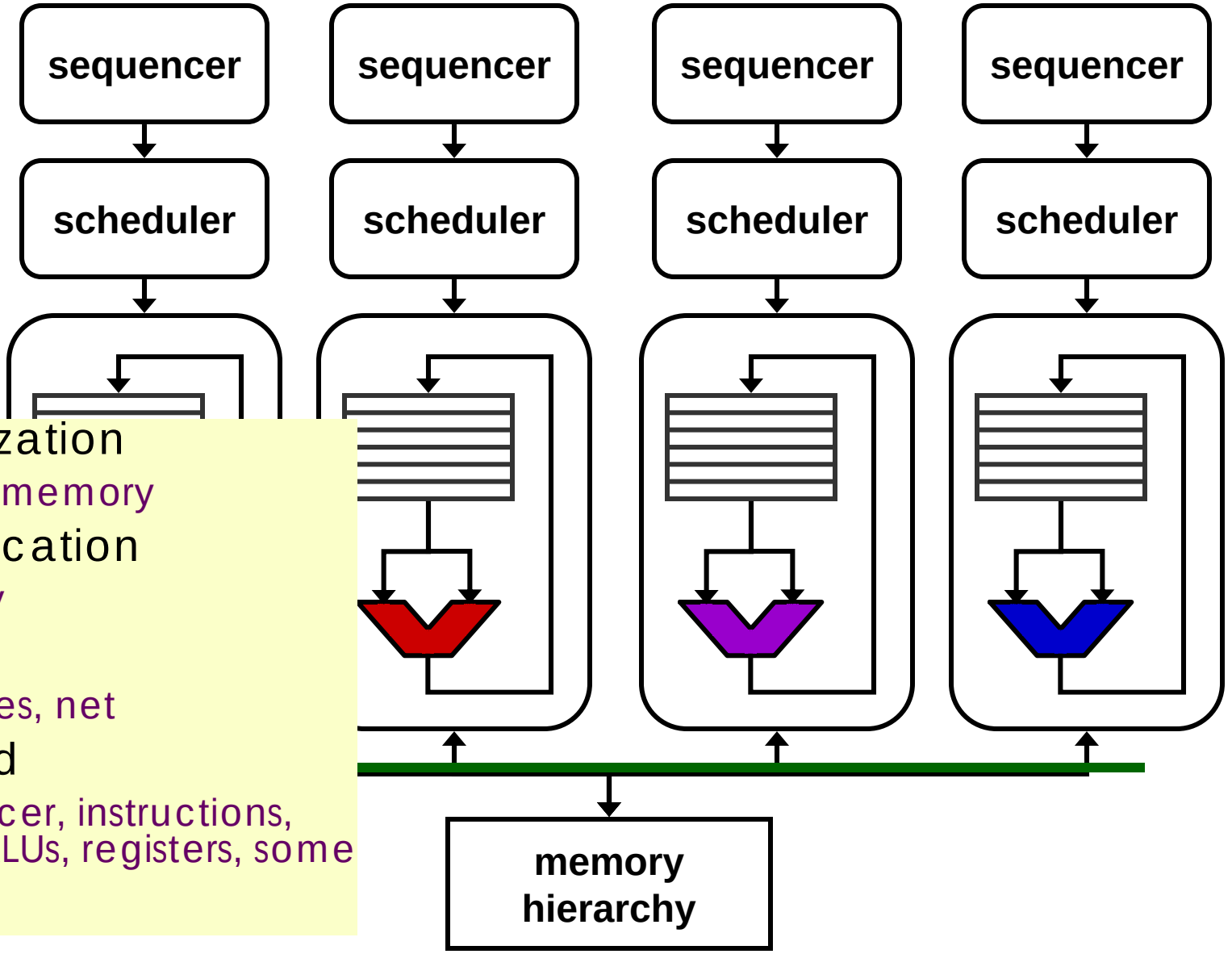
TLP for multiple ALUs



MIMD – shared memory (TLP for multiple ALUs)

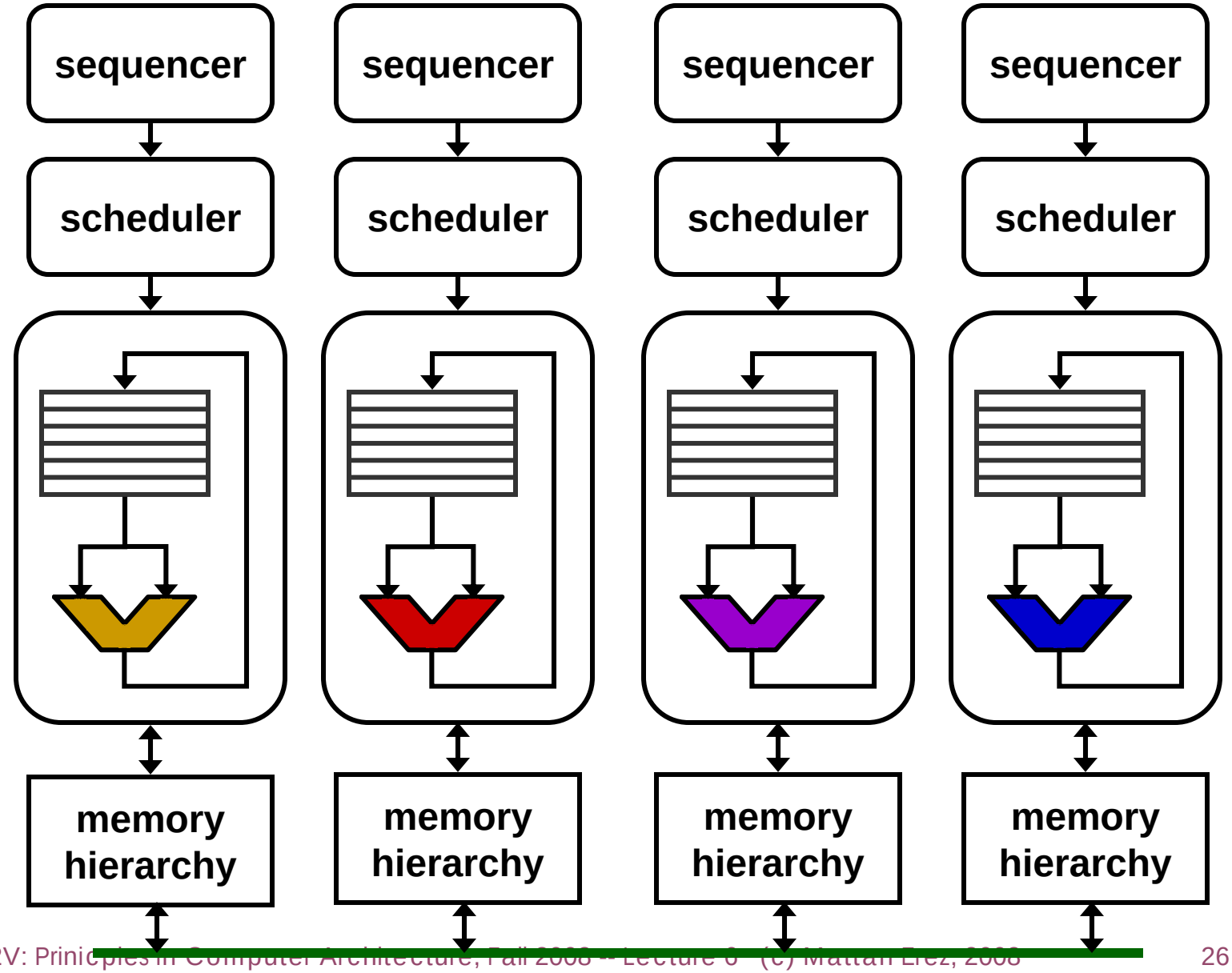


MIMD – shared memory (TLP for multiple ALUs)

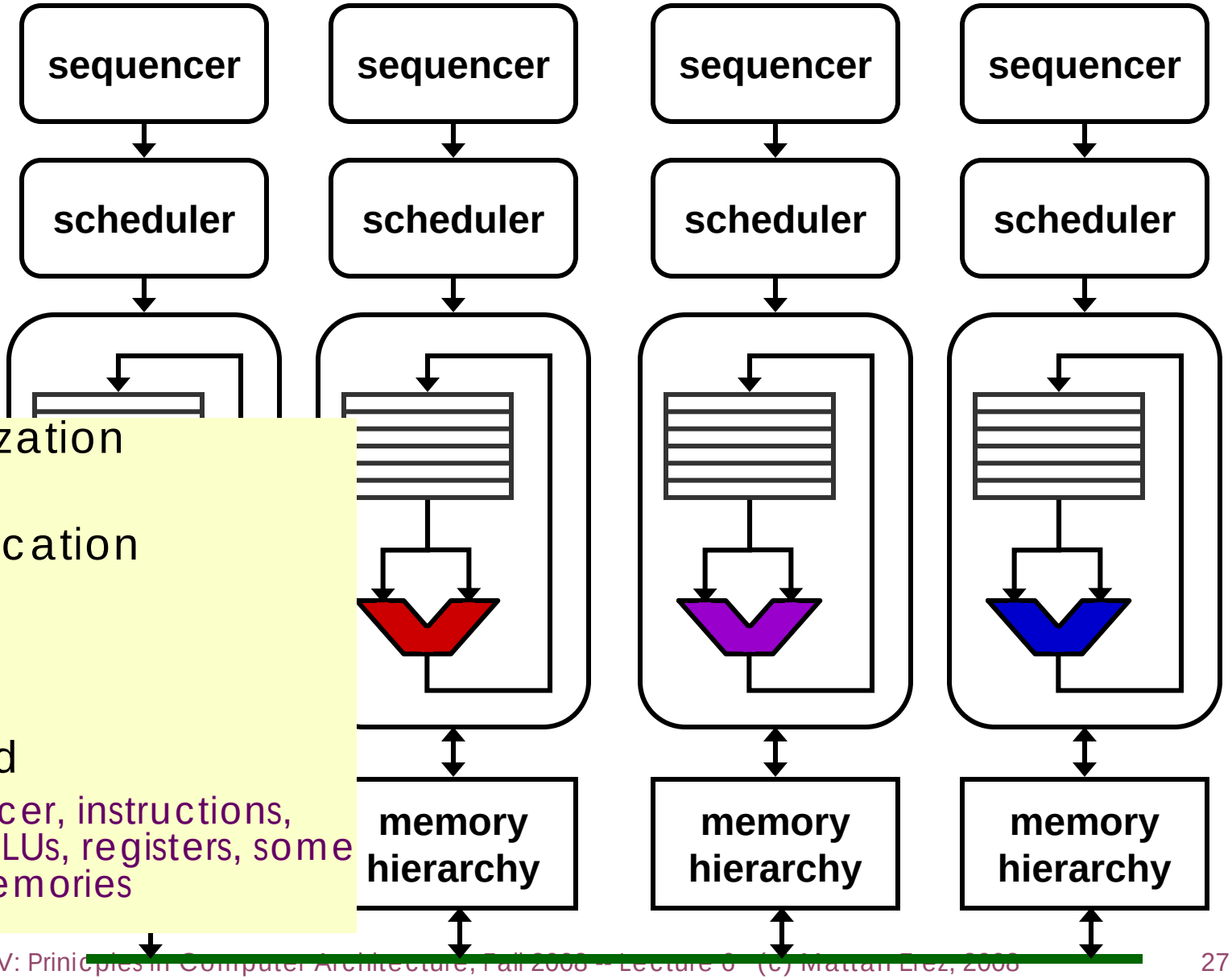


- Synchronization
  - Explicit, memory
- Communication
  - Memory
- Shared
  - Memories, net
- Partitioned
  - Sequencer, instructions, OOO, ALUs, registers, some nets

MIMD – distributed memory



MIMD – distributed memory



- Synchronization
  - Explicit
- Communication
  - Explicit
- Shared
  - Net
- Partitioned
  - Sequencer, instructions, OOO, ALUs, registers, some nets, memories

Summary of communication and synchronization

Style	Synchronization	Communication
Superscalar	explicit signals (RS)	registers+bypass
VLIW	clock+compiler	registers (bypass?)
Dataflow	explicit signals	registers+explicit
SIMD	clock+compiler	explicit
MIMD	explicit signals	memory+explicit

Summary of communication and synchronization

Style	Synchronization	Communication
Superscalar		
VLIW		
Dataflow		
SIMD		
MIMD		

Summary of sharing in ILP HW

Style	Seq	Inst	OOO	Regs	Mem	ALUs	Net
Superscalar	S	P	S	S	S	S	S
SMT/TLS	P	P	S	S	S	S	S
VLIW	S	P	N/A	S	S	**P**	S
Dataflow	B	P	P	B	S	**P**	S

Summary of sharing in ILP HW

Style	Seq	Inst	OOO	Regs	Mem	ALUs	Net
Superscalar							
SMT/TLS							
VLIW							
Dataflow							

Summary of sharing in DLP and TLP

Style	Seq	Inst	OOO	Regs	Mem	ALUs	Net
Vector							
SIMD							
MIMD							

Summary of sharing in DLP and TLP

Style	Seq	Inst	OOO	Regs	Mem	ALUs	Net
Vector	S	S	N/A	P	**s**	P	B
SIMD	S	S	N/A	P	**p**	P	B
MIMD	P	P	P	P	S/P	P	B