Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture

Ramadass Nagarajan

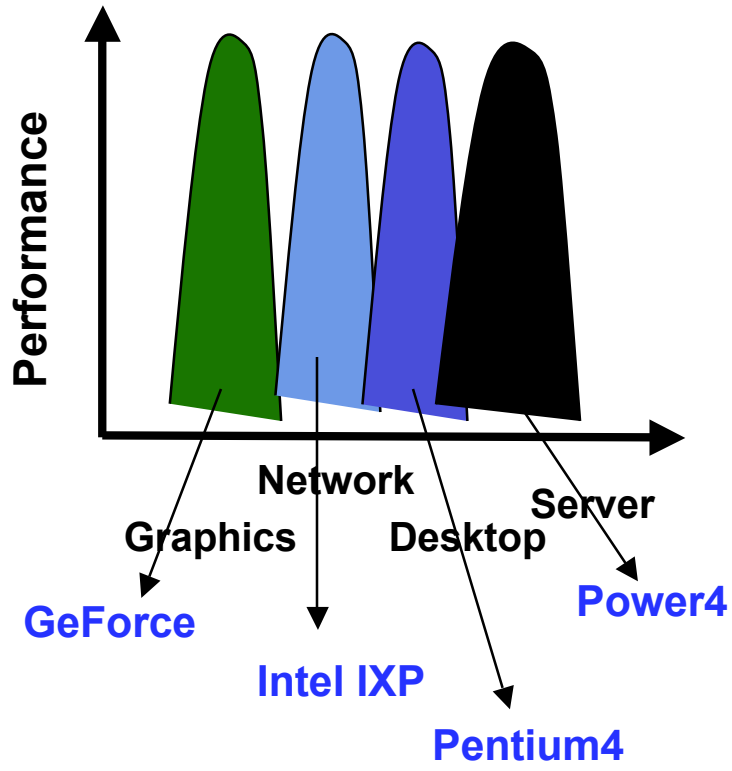
Karthikeyan Sankaralingam
Haiming Liu
Changkyu Kim
Jaehyuk Huh

Doug Burger
Stephen W. Keckler
Charles R. Moore

Computer Architecture and Technology Laboratory
Department of Computer Sciences
The University of Texas at Austin

Trends in Programmable Processors

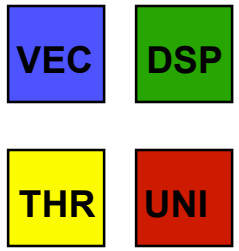
- Workloads are becoming diverse
  - Increased specialization among processors
- Benefits of specialization
  - Performance, power, area
- Problems of specialization
  - Poor performance outside intended domain
  - Little design re-use



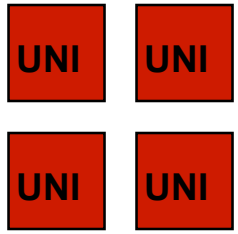
Courtesy: Bob Graybill, DARPA

Homogeneity versus Heterogeneity

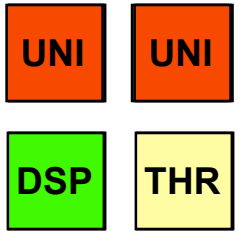
- Heterogeneous - multiple different types of processors
  (Eg: Tarantula [Espasa et al, ISCA 2002])
  + Performance advantages
  – Load balancing inefficiencies
  – Higher design complexity



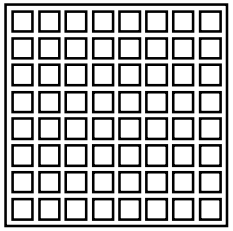
- Homogeneous - single or multiple of same processor
  + Flexible/general purpose
  + Ample design reuse
  – Processor mismatch inefficiencies



- Approach: **Hardware Polymorphism**
  – Start with high performance homogeneous substrate
  – Add coarse-grained reconfigurability to micro-architectural elements
  – Manage different elements appropriately for different applications



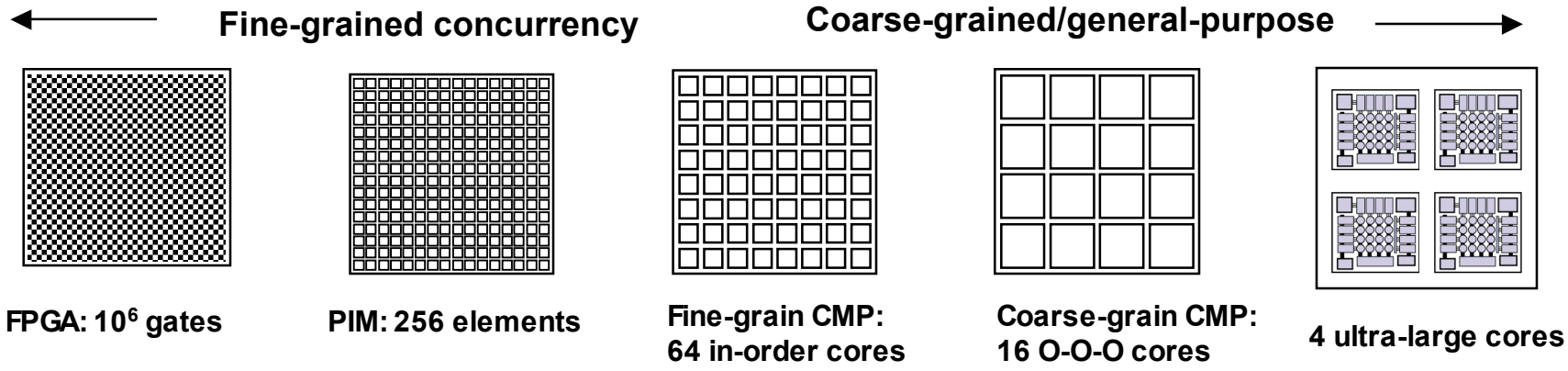
Challenges for Homogenous Systems



Fine-grain CMP:
64 in-order cores

- High degree of partitioning
  - Necessary for fine-grained concurrency
- High computational density (ALUs/mm^2)
  - Necessary for data parallel applications
- Keep communication localized
  - Permits technology scalability
- Minimize specialized hardware
  - Reduces design complexity

What is the Right Granularity of Processing?



- Configuring granularity through polymorphism
  - Synthesis: Emulate coarse-grained cores using fine-grained PEs
  - Partitioning: Partition coarse-grained cores into fine-grained PEs
- Synthesizing fine-grained PEs difficult at best

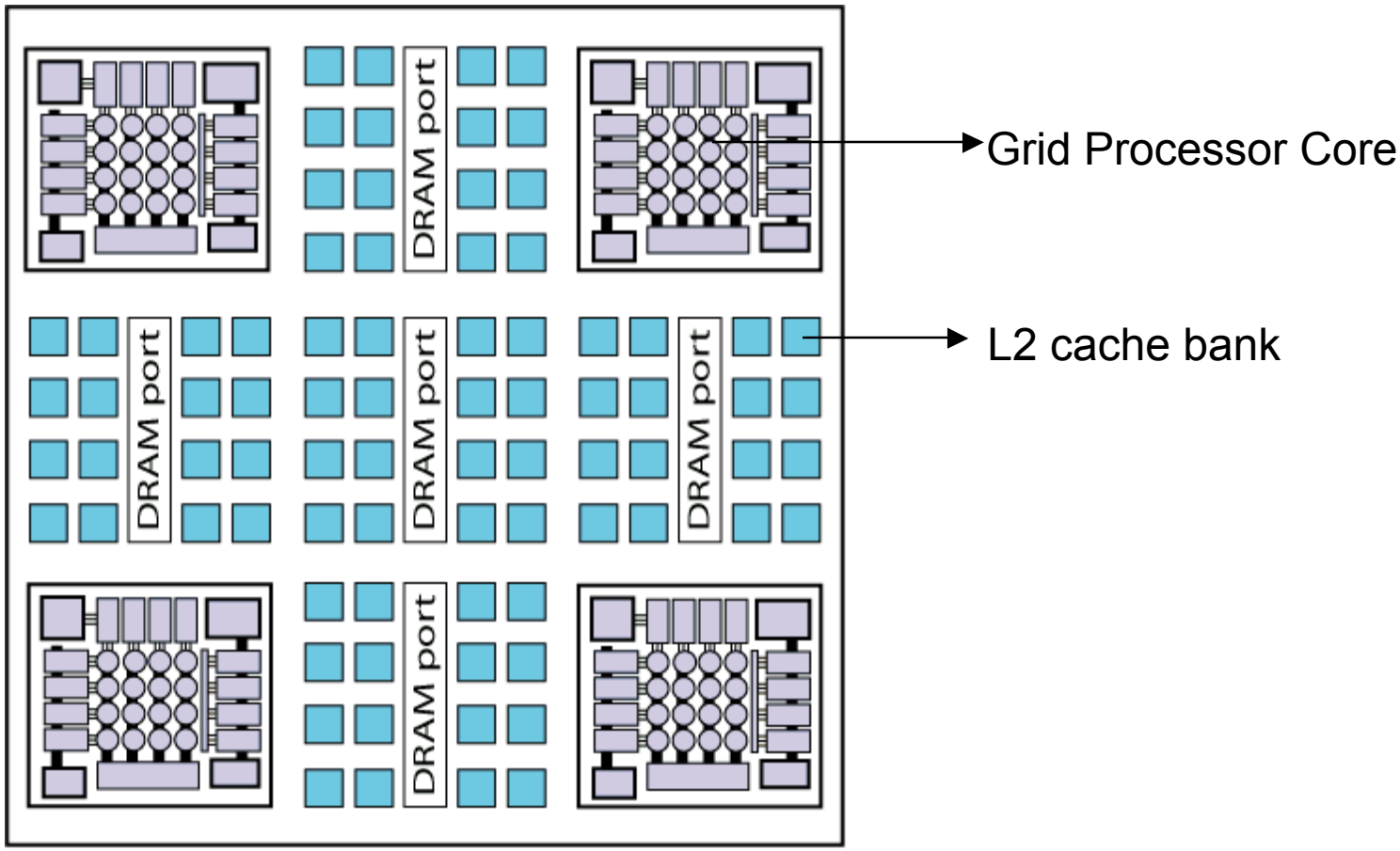
- Partitioning ultra-large cores
  - Maximize resources for single-threaded performance
  - Sub-divide for finer granularity
  - Configure micro-architectural elements for different levels of parallelism

Outline

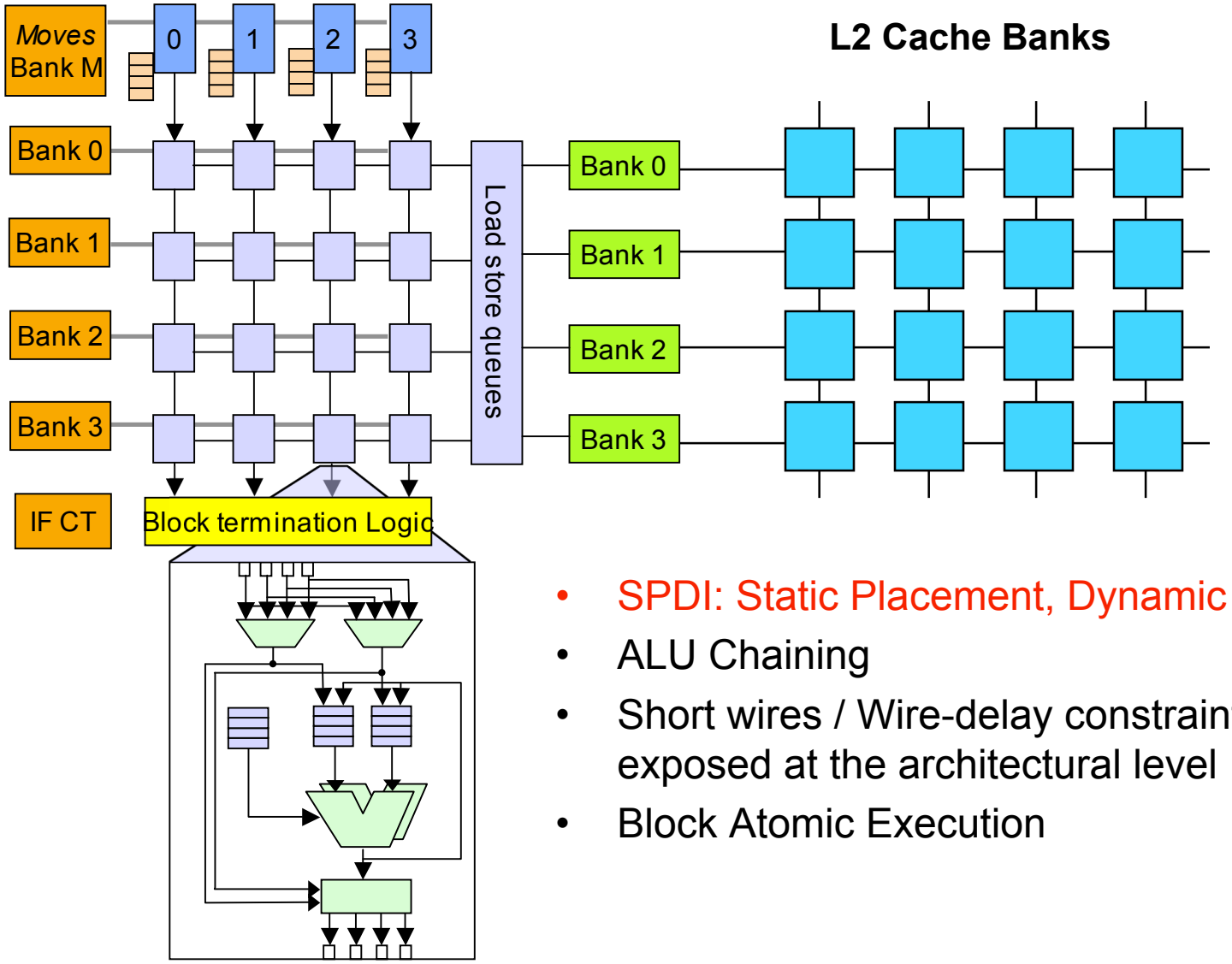
- Introduction to polymorphous systems
- TRIPS architectural overview
- Polymorphous components
- Supporting different granularities of parallelism
  - Instruction-level parallelism (ILP)
  - Thread-level parallelism (TLP)
  - Data-level parallelism (DLP)
- Conclusions

TRIPS Overview

CMP with large Grid Processor cores and L2 cache banks



TRIPS Overview

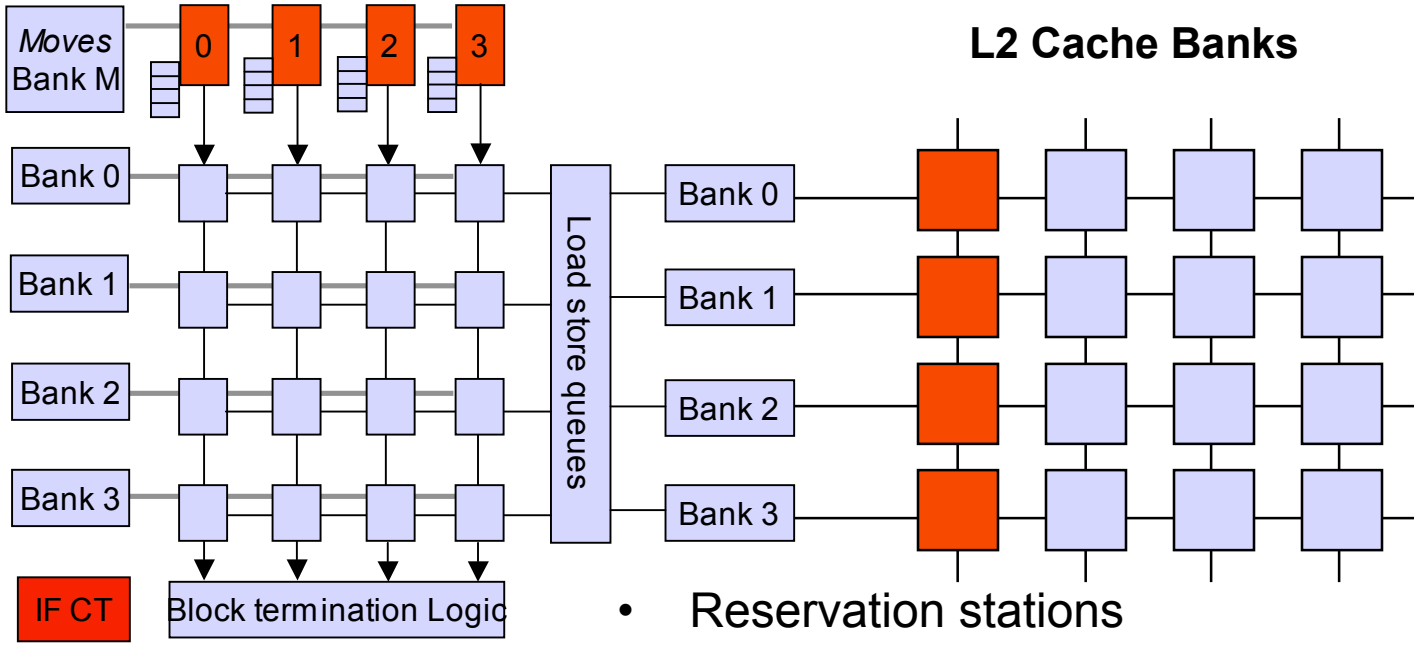


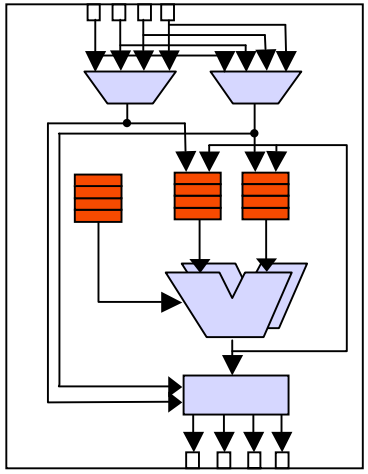
- SPDI: Static Placement, Dynamic Issue
- ALU Chaining
- Short wires / Wire-delay constraints exposed at the architectural level
- Block Atomic Execution

Challenges for Different Levels of Parallelism

- Instruction-level parallelism[Nagarajan et al, Micro01]
  - Populate large instruction window with useful instructions
  - Schedule instructions to optimize communication and concurrency
- Thread-level parallelism
  - Partition instruction window among different threads
  - Reduce contentions for instruction and data supply
- Data-level parallelism
  - Provide high density of computational elements
  - Provide high bandwidth to/from data memory

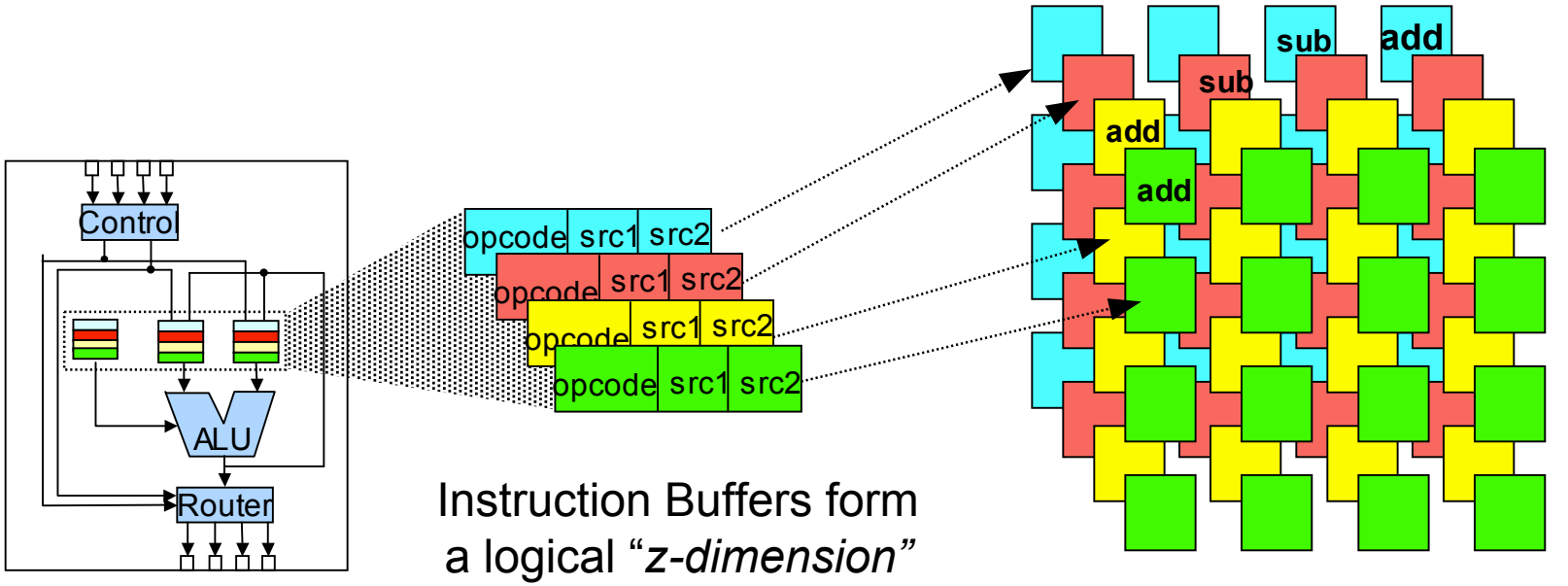
TRIPS Configurable Resources





- Reservation stations
  - Instruction window management
- Instruction fetch control
  - Speculation/non-speculation, multiple threads, mapping re-use.
- Register files
  - Speculative vs. non-speculative data storage
- L2 cache banks
  - tag lookup, replacement, b/w to near banks

Aggregating Reservation Stations: Frames



Execution Node

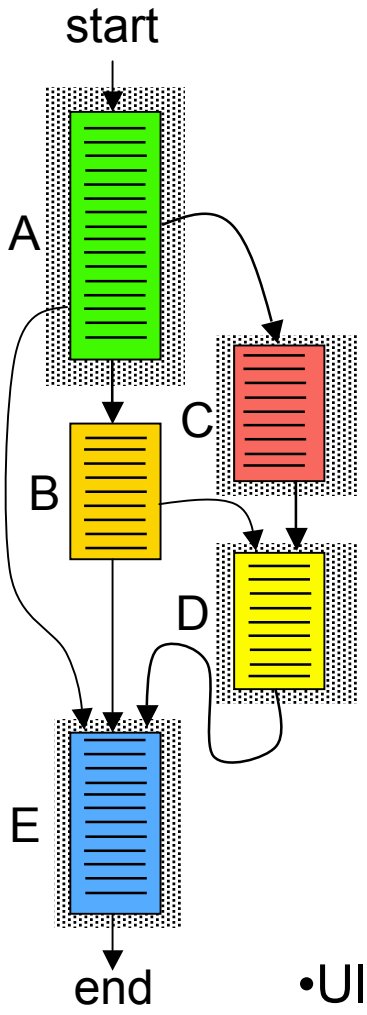
Instruction Buffers form a logical "*z-dimension*" in each node

4 logical *frames* each with 16 instruction slots

- Instruction buffers add depth to the execution array
  - 2D array of ALUs; 3D volume of instructions

Extracting ILP: Frames for Speculation

16-wide OOO issue



Execute A

↓

Predict C
Execute C

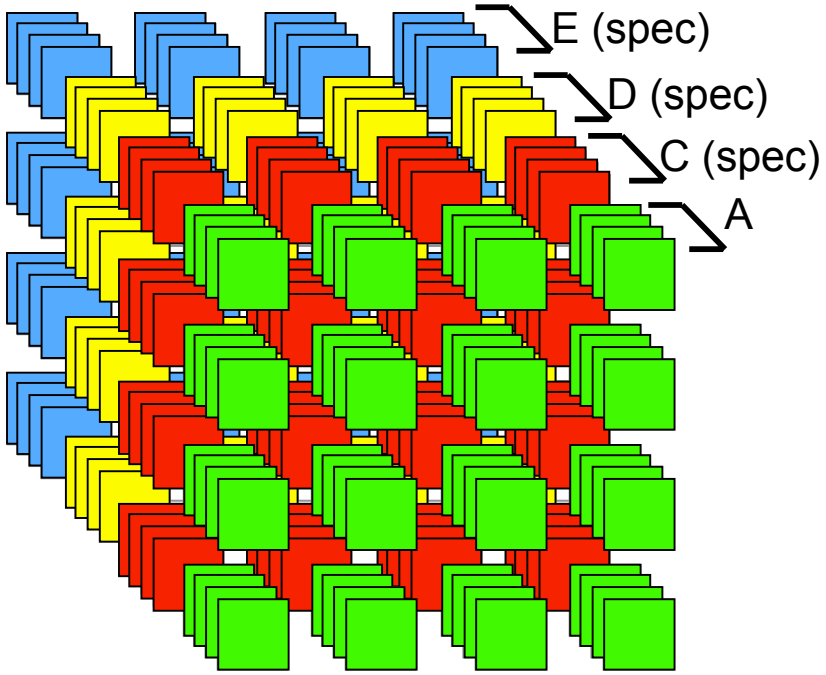
↓

Predict D
Execute D

↓

Predict E
Execute E

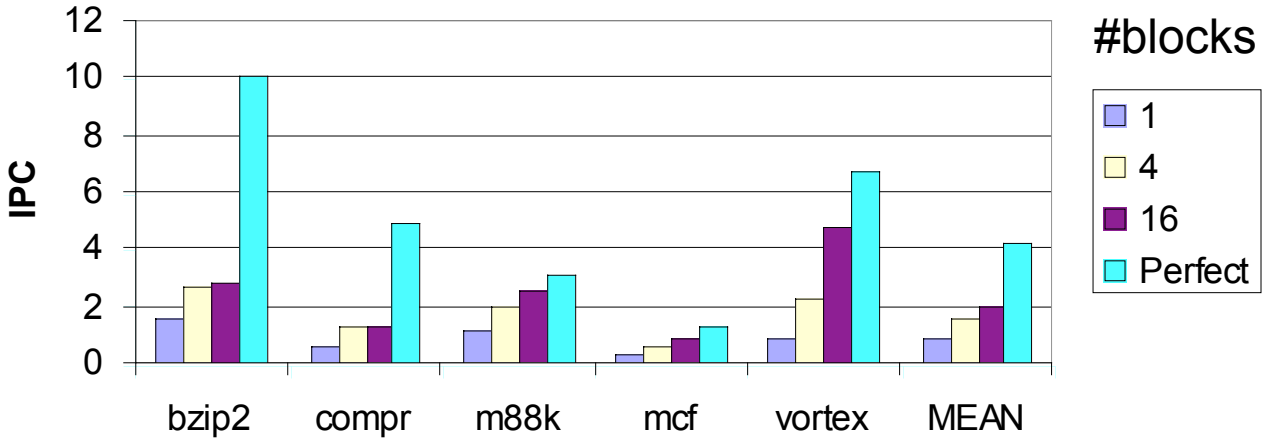
16 total frames (4 sets of 4)



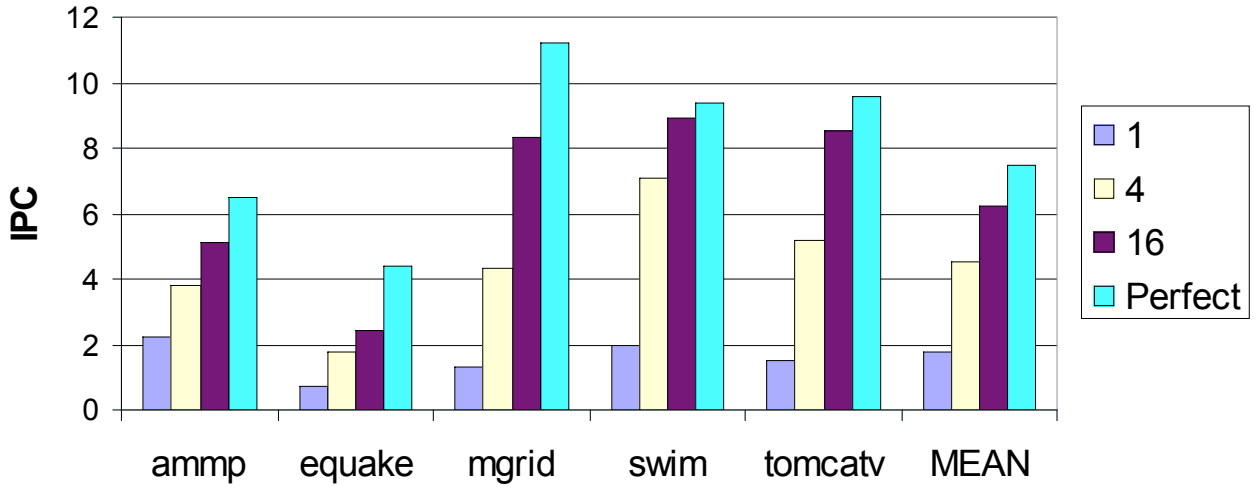
- Ultra-wide issue from a large distributed instruction window

ILP Results with Speculation

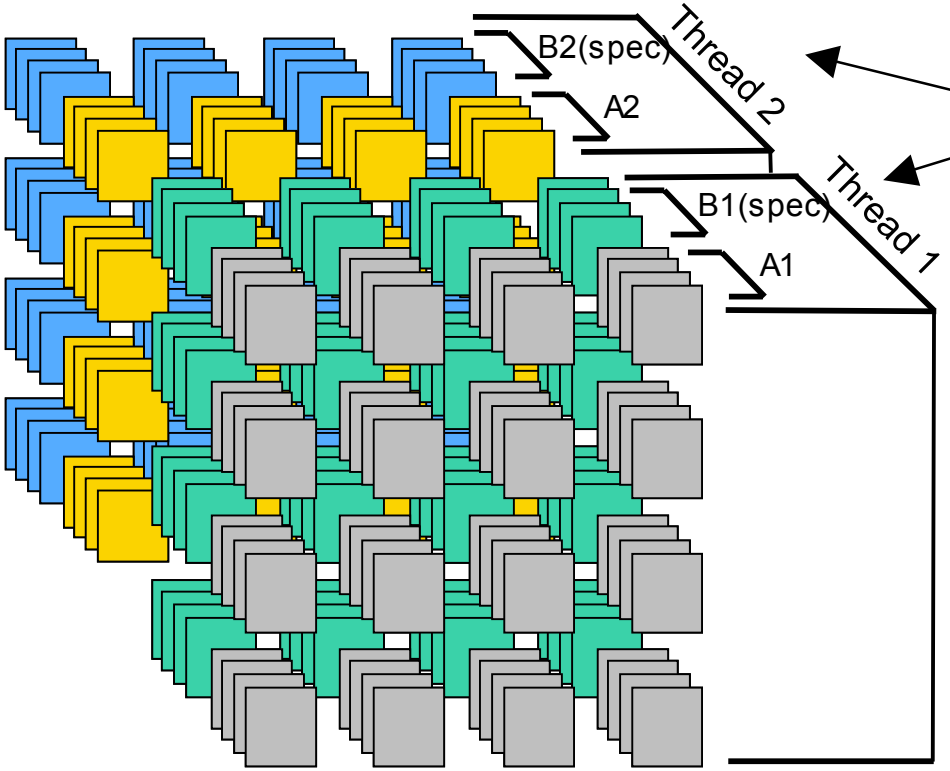
SPEC Int Programs



SPEC FP programs



Configuring Frames for TLP



Divide frame space among threads

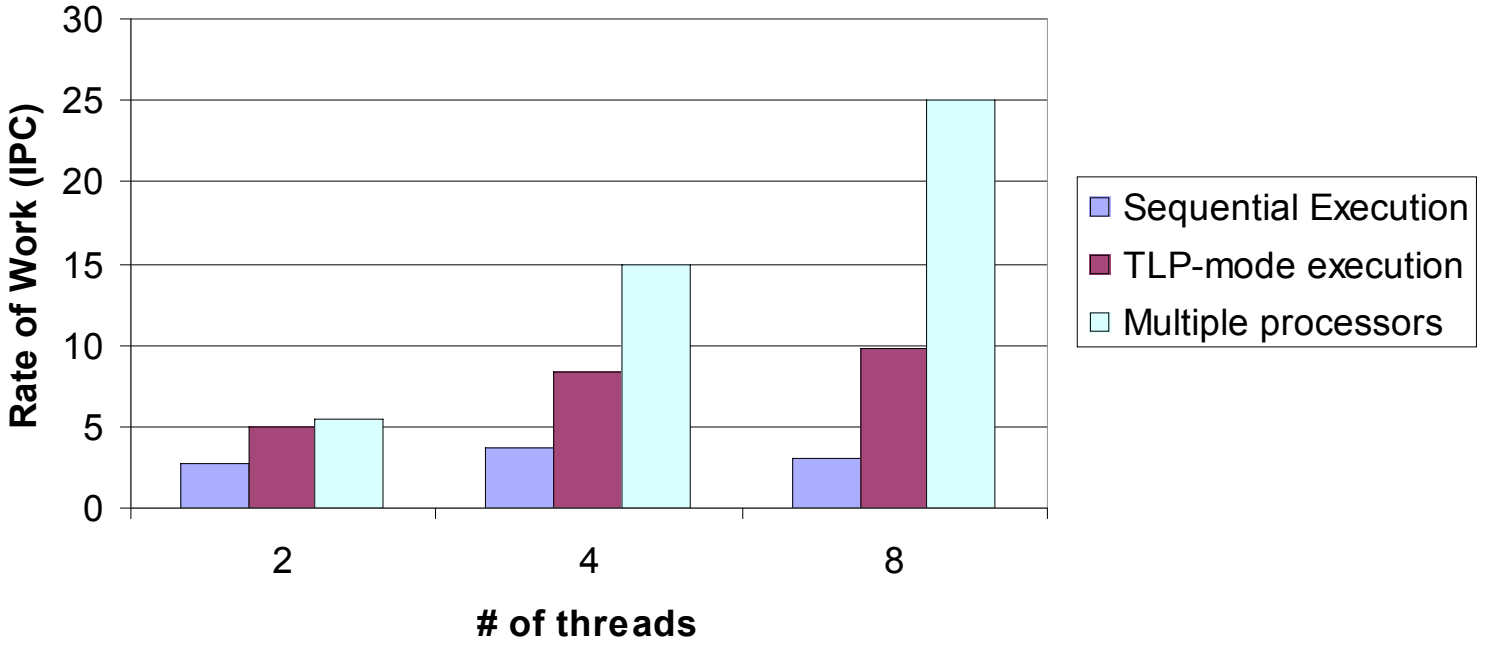
- Each can be further sub-divided to enable some degree of speculation

- Shown: *2 threads, each with 1 speculative block*

- Alternate configuration might provide 4 threads

•Multiple partitioned instruction window for different threads

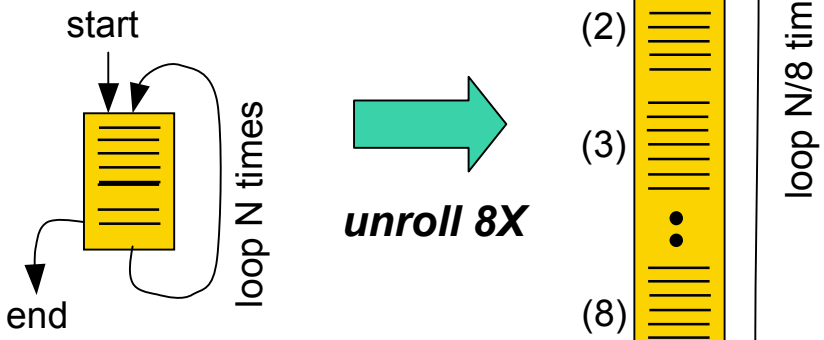
TLP Results

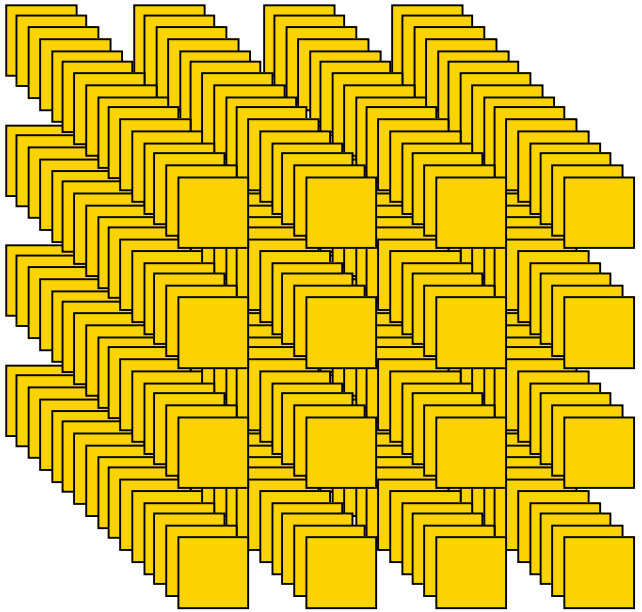


- Speedup: 1.8x to 2.9x
- Reasons for performance losses
  - Contention for resources (principally in instruction and data supply)
  - Reduced instruction window size

Using Frames for DLP

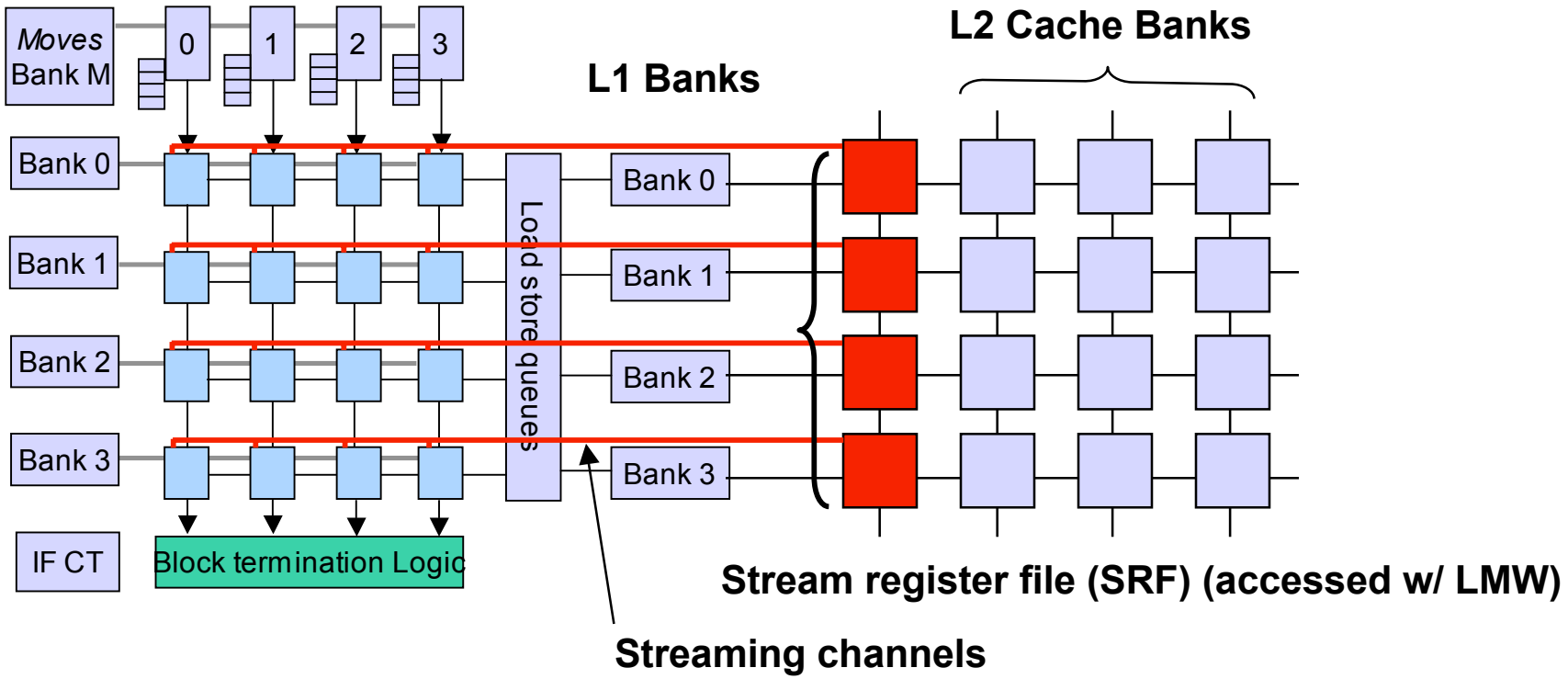
Streaming Kernel:
- read input stream element
- process element
- write output stream element





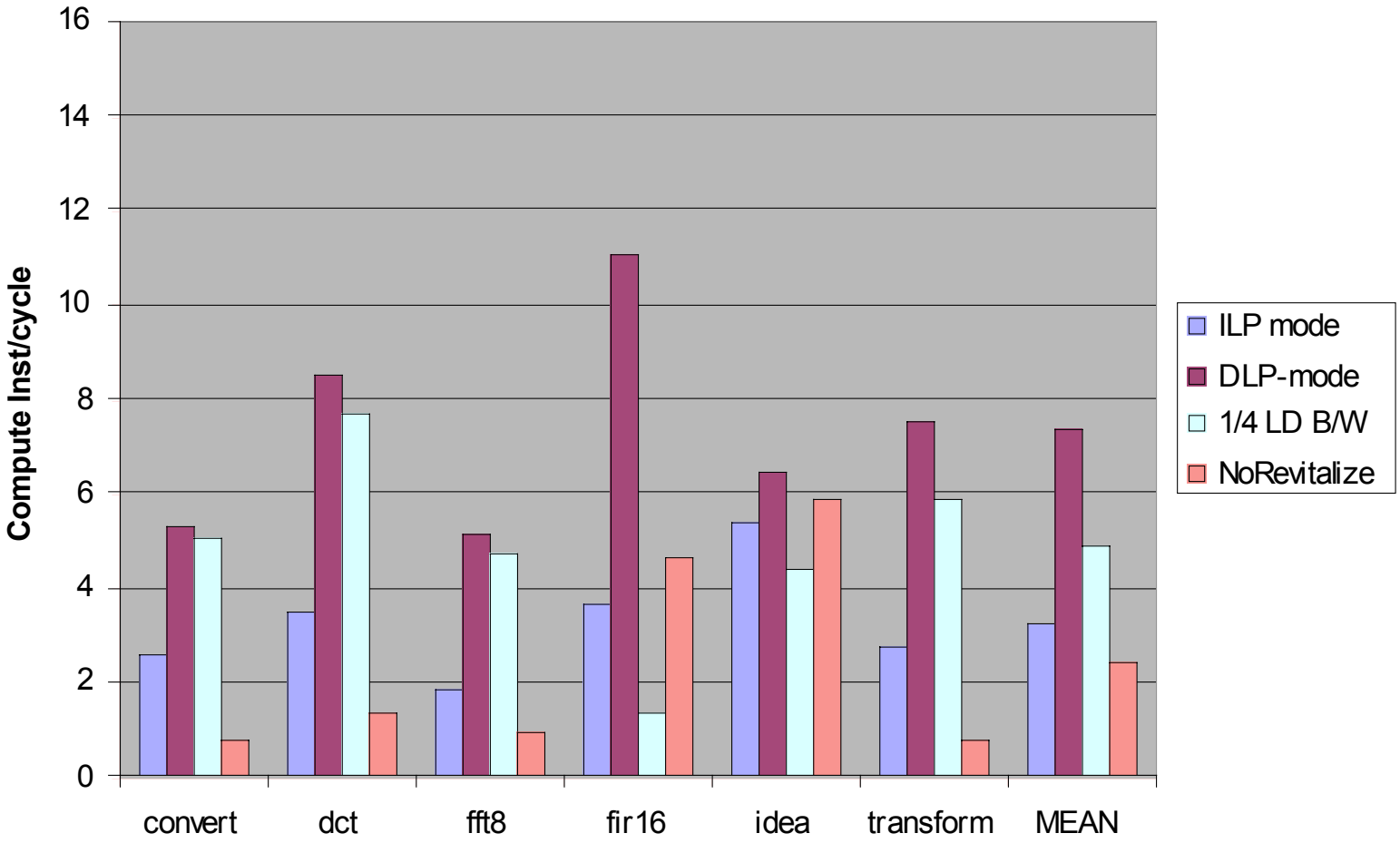
- Map very large unrolled kernels to window
  - Turn-off speculation
  - Keep communication localized
- Mapping re-use: Fetch/map loop body once, re-use many times
  - Re-vitalization initiates successive iterations

Configuring Data Memory for DLP



- Regular data accesses
  - Subset of L2 cache banks configured as SRF
  - High bandwidth data channels to SRF
  - Reduced address communication
- Constants saved in reservation stations

DLP Results (4x4 GPA)



- **Performance metric omits overhead, LD/ST instructions**

Results: Summary

- ILP: instruction window occupancy
  - Peak: 4x4x128 array $\Rightarrow$ 2048 instructions
  - Sustained: 493 for Spec Int, 1412 for Spec FP
  - Bottleneck: branch prediction
- TLP: instruction and data supply
  - Peak: 100% efficiency
  - Sustained: 87% for two threads, 61% for four threads
- DLP: data supply bandwidth
  - Peak: 16 ops/cycle
  - Sustained: 6.9 ops/cycle

Related Work

- Polymorphous homogeneous
  - SmartMemories: Modular reconfigurable architecture[Mai, ISCA '01]
- Fine-grained homogeneous
  - RAW: Baring it all to software [Waingold, IEEE Computer '00]
- Ultra-fine grained homogeneous
  - Piperench reconfigurable architecture and compiler [Goldstein, IEEE Computer '00]
- Heterogeneous
  - Tarantula Vector Extensions to the EV8 [Espasa, ISCA '02]

Conclusions

- TRIPS: Coarse-grained homogeneous approach with polymorphism.
  - Sub-divide a powerful uniprocessor
  - ILP: Well-partitioned powerful uniprocessor (GPA)
  - TLP: Divide instruction window among different threads
  - DLP: Mapping reuse of instructions and constants in grid
- Future work
  - Demonstrate viability with HW/SW prototype
  - Design software interfaces to exploit configurable hardware
- How well homogeneous approaches compare with specialized cores?
- How large should these cores scale?