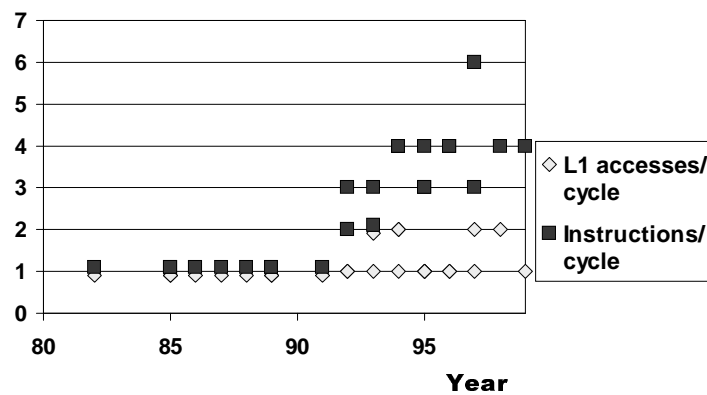


Lecture 24 ECE669
MLP (memory Level Parallelism) vs ILP
Maps: A Compiler-Managed Memory System for
Raw Machines

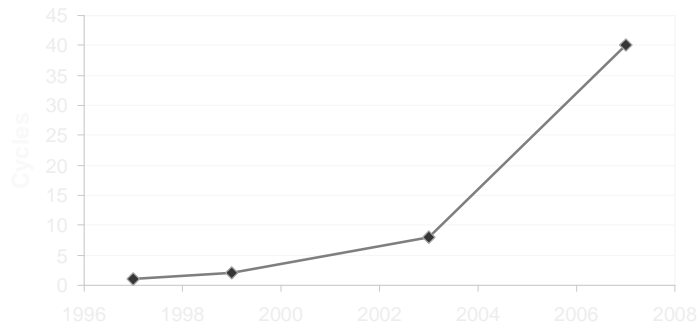
(Maps Courtesy of Rajeev Barua – University of Maryland)

ILP has improved, memory parallelism
has not



Multiple cache banks require arbitration logic

Cost of centralized memory

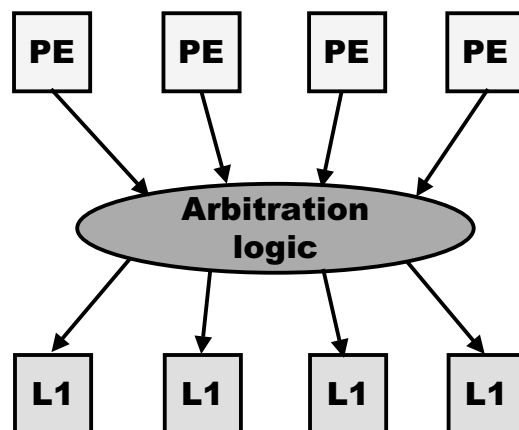


Cannot scale to larger caches retaining single cycle access.

- Runtime-resolved banks imply no locality on chip
 - May have to cross chip diameter

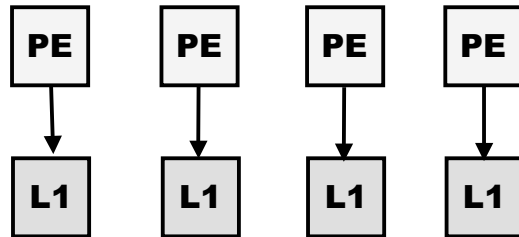
3

Hardware-managed memory banks



4

Hardware-managed memory banks
Software-exposed memory



- No arbitration logic
- Exploits locality on-chip
- Compiler-expressible memory parallelism

5

Software-exposed memory

Examples:

- DSP chips, iWarp, NuMesh, ... Raw

Providing a unified view of memory to the programmer

- MAPS Compiler

6

Enabling compiler technology

Maps compiler system performs two tasks:

- Memory distribution : Specifies how to map data objects in program to different memory banks.
- Bank disambiguation : Specifies which bank a load/store goes to assuming the above distribution.

7

Bank disambiguation

Requirement:

Memory reference must access the same compile-time predictable bank every time.

- Is this even possible ?

Goal:

Bank disambiguation while maximizing memory parallelism.

8

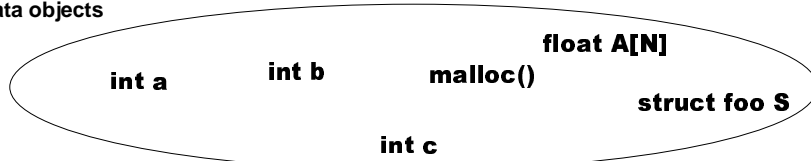
Bank disambiguation methods

- Equivalence class unification
: Baseline technique for all memory references
- Modulo unrolling
: Optimization for affine function array references

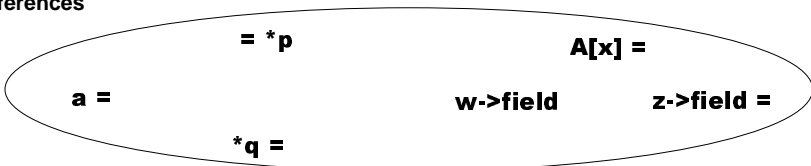
9

Equivalence class unification

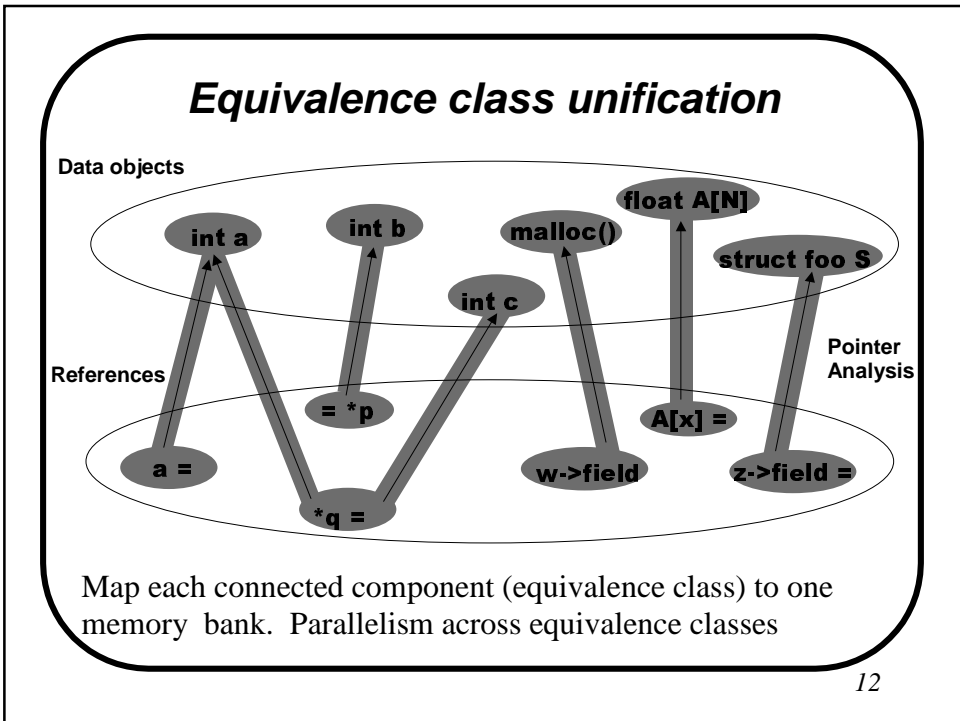
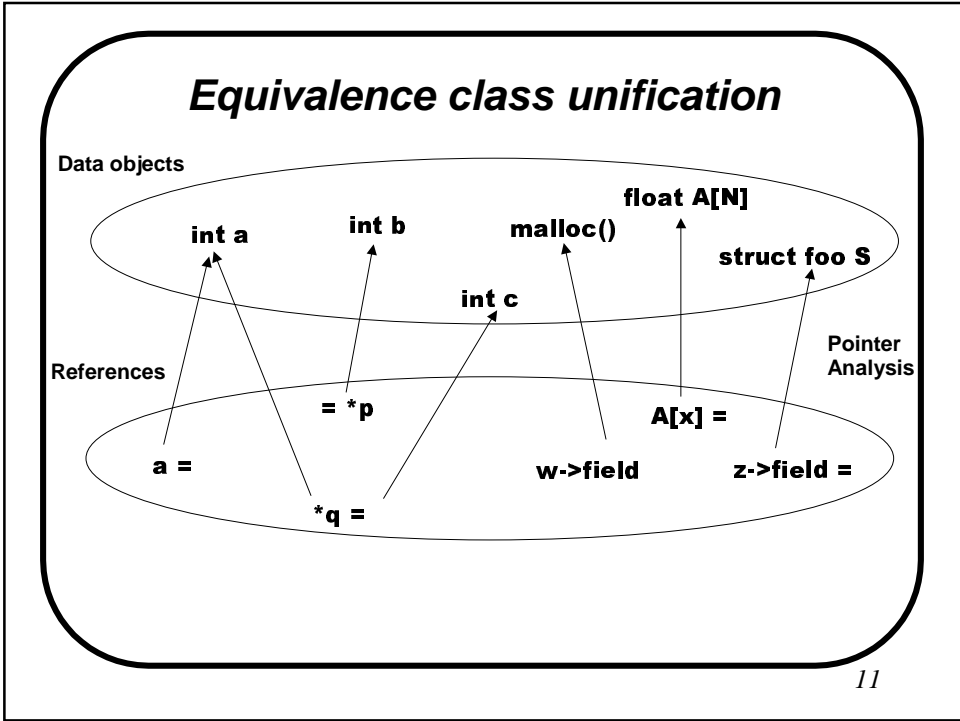
Data objects



References



10



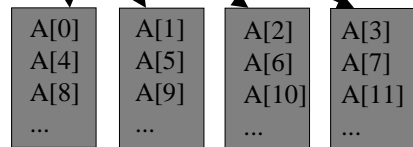
Memory parallelism from ECU

- Scalars ✓
- Structures ✓
- Arrays

13

Modulo Unrolling

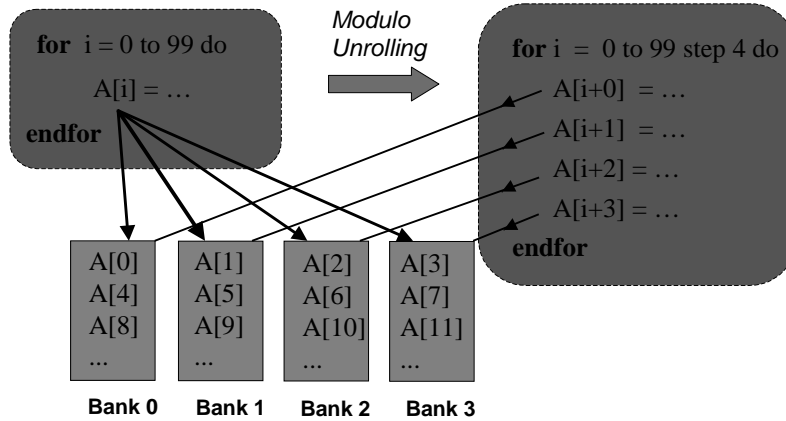
```
for i = 0 to 99 do  
  A[i] = ...  
endfor
```



Bank 0 Bank 1 Bank 2 Bank 3

14

Modulo Unrolling



15

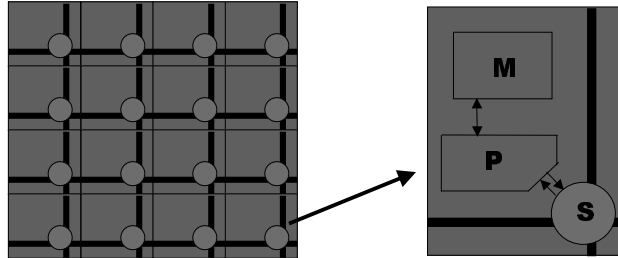
Fully automated method handling:

- Arbitrary affine functions
- Imperfectly nested loops
- Unknown loop bounds
- Is unaffected by non-affine accesses in the same loop

Mathematical formulas in paper

16

Raw architecture



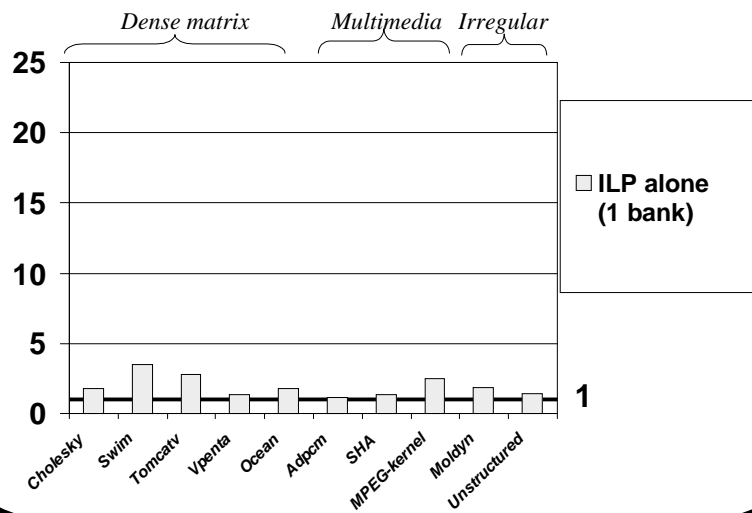
Provides :

- Multiple instruction streams
- Software-exposed distributed memory
- Fast communication using static network (3 cycle load)
 - Traditional dynamic network much slower (28 cycles)

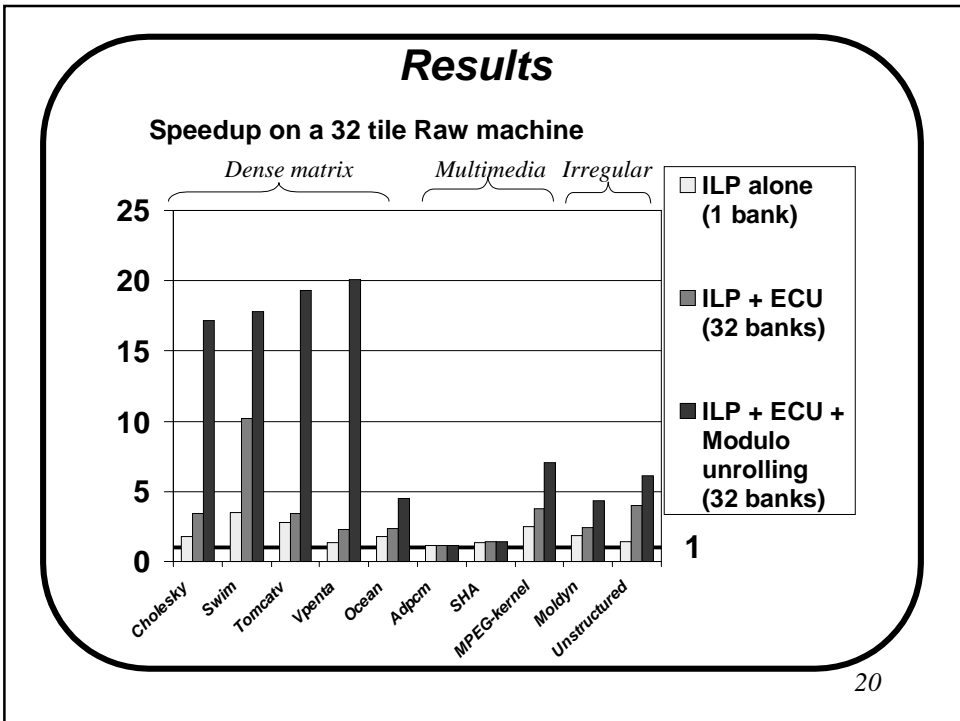
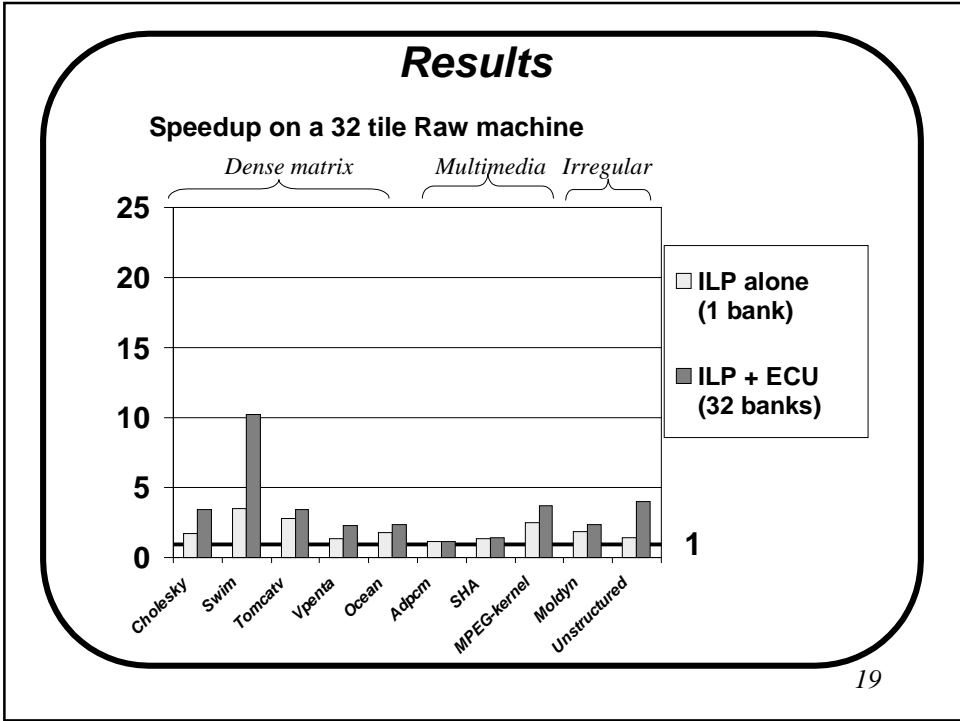
17

Results

Speedup on a 32 tile Raw machine



18



Related Work

- Relative memory disambiguation
 - Machines with runtime-resolved banks
Eg: Multiflow Trace VLIW *Lowney et. al. 93*
- Bank disambiguation
 - Machines with software-addressable banks
Eg: ELI-512 *Fisher83*
 - Unrolling observation

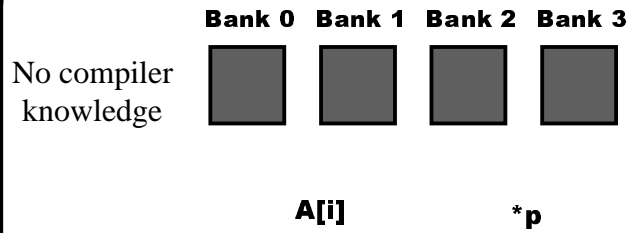
21

Summary

- Showed a class of architectures that can exploit memory parallelism,
- Presented enabling *Maps* compiler techniques for this class,
- Demonstrated a factor of 3 to 5 improvement using these techniques.

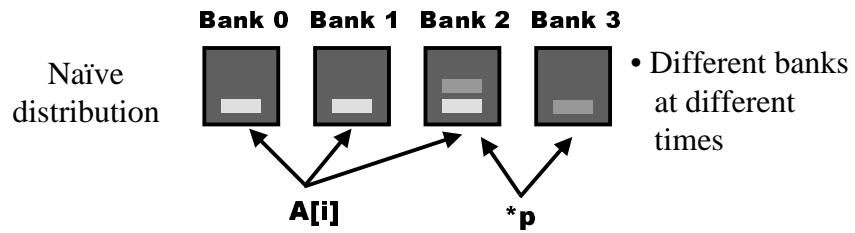
22

Understanding bank disambiguation



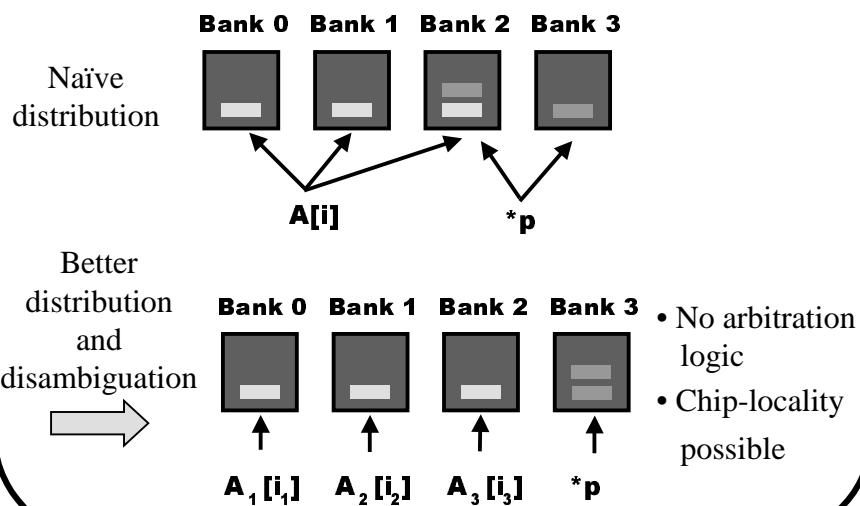
23

Understanding bank disambiguation



24

Understanding bank disambiguation



25

Difficulties with scaling on-chip memory bandwidth

For example:

	Max. number of L1 accesses /cycle
Alpha 21364	2
Pentium-II	2
MIPS R12000	1
UltraSparc-3	1

Multiple banks require arbitration logic

26

Overview

- ✓ • Understanding bank disambiguation
- ✓ • *Maps* bank disambiguation methods
 - Raw architecture
 - Results

27

Example:

```
for x = 2 to vmax do           // unroll by 2
  A[1][2x] = ...
  for y = 5 to x do           // unroll by 4
    ...
    A[x + 2y][100 - y] += y - x
  endfor
  sum += p → data - ...
  p = p → next
  while (...)
    ...
  endwhile
endfor
```

28

General method in [HiPC-98 Barua et. al.]

Unroll factor (U_j) :

$$D_j = N / \gcd \left(N, \sum_{i=1}^d c_{i,j} \prod_{l=i+1}^d MAX_l \right)$$
$$U_j = \text{lcm}(D_j, s_j) / s_j$$

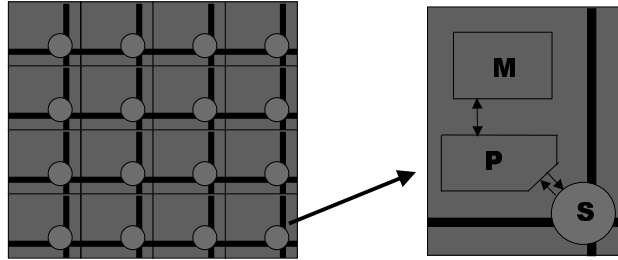
29

Overview

- Understanding bank disambiguation
- *Maps* bank disambiguation methods
- Raw architecture
- Results

30

Raw Architectures

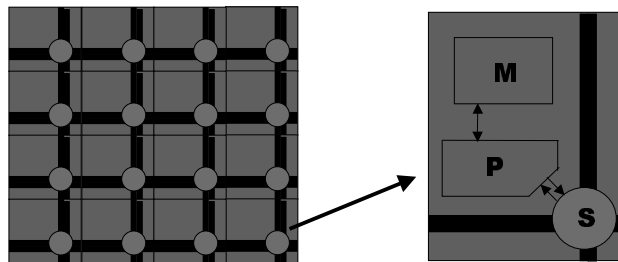


Features :

- Simple, replicated design
- Short wires ⇔ fast clock
- Software-exposed memory

31

Communication on Raw



Primary network is a very fast static network

- Compiler routed and scheduled.
- End-to-end cost of load is 3 cycles.
 - Traditional (dynamic) network is much slower.
End-to-end cost of load is 28 cycles

32

Maps: memory disambiguation

Maps, an enabling compiler technology for architectures with software-exposed memory.

- Do deep program analysis to perform memory distribution and disambiguation
- Resulting information and transformations may be useful on a wide class of machines.

[ISCA-99 Barua et. al.]

33

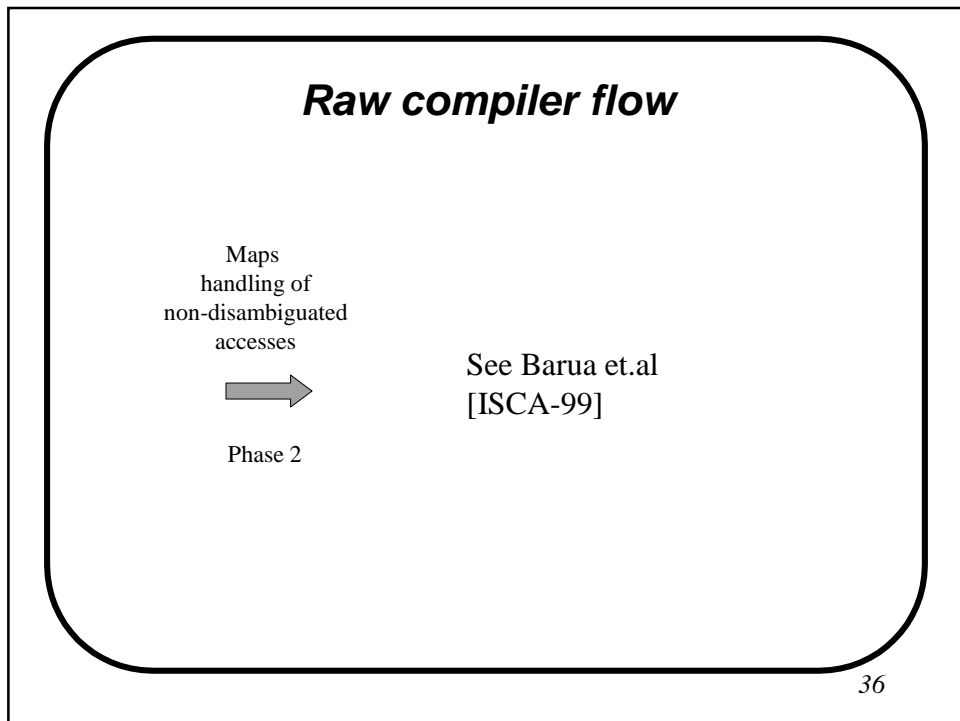
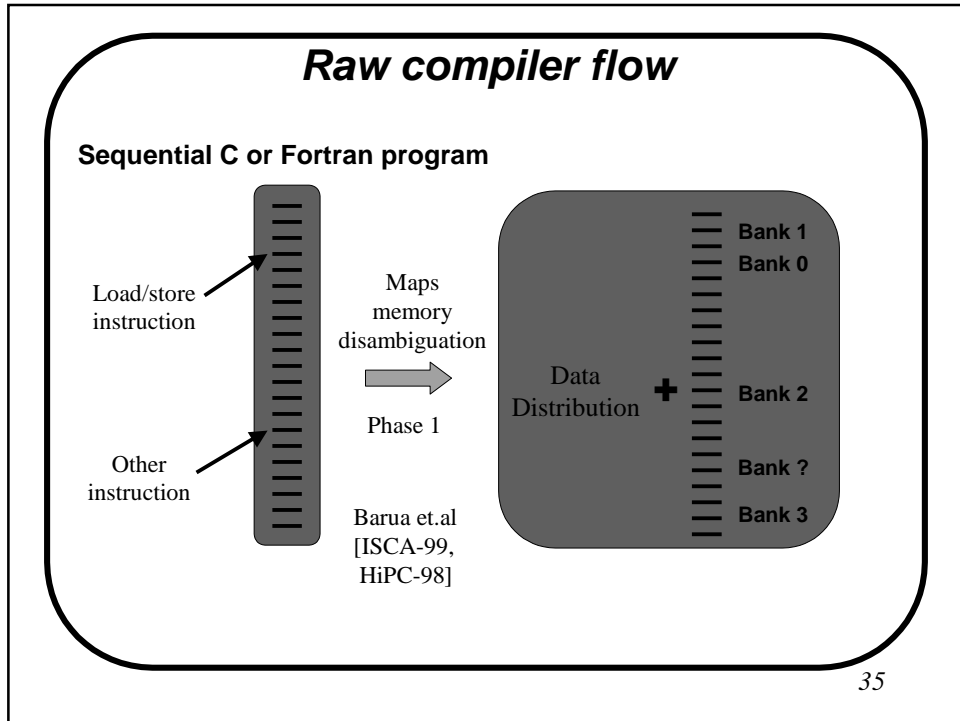
Preview of results

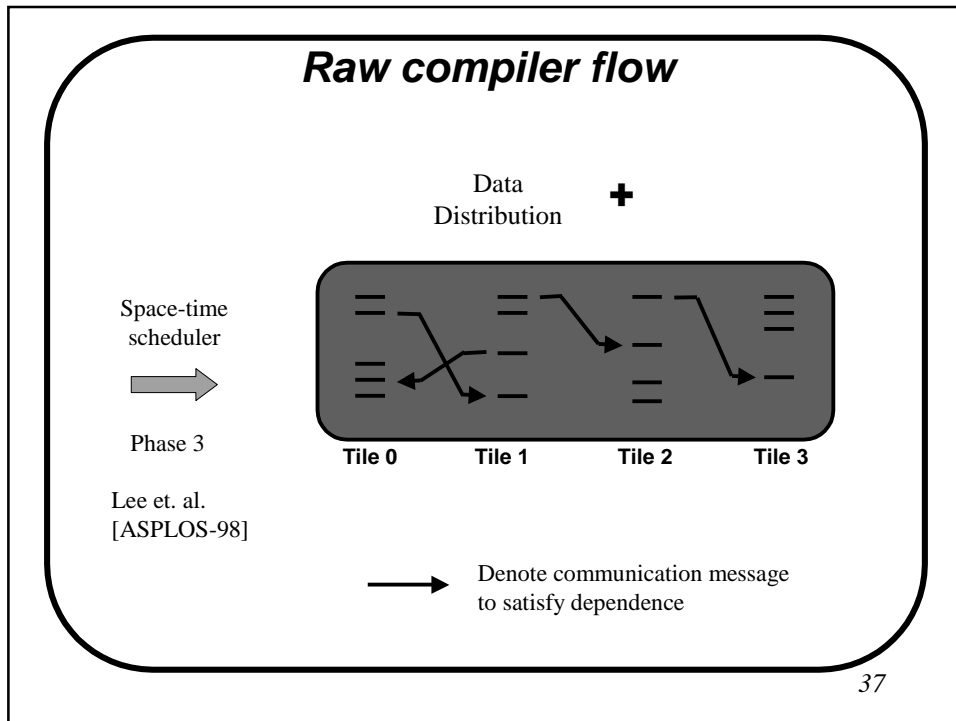
Speedup for Mpeg-encode on Raw:

(1-issue, 1 memory bank)	1.00
(32-issue, 1 memory bank)	2.47
(32-issue, 32 memory banks)	7.07

Results later will show similar improvements using Maps for a wide range of programs.

34





- ### **Dynamic network costs**
- Message composition costs
 - Destination, handler address or type.
 - Expensive receipt mechanisms
 - Disambiguation of message type
 - Possible saving away of state
 - Congestion costs
 - May spill to memory
 - Synchronization costs
 - Ensuring atomic updates may require locks.
- 38

May not be a big problem currently

But:

- Larger issue rate will become possible
- Wire delay will be much more important in the future
- Average required memory bandwidth \neq Peak

39

What about non-disambiguated accesses ?

Equivalence-class unification can eliminate all non-disambiguated accesses.

However allowing some non-disambiguated accesses enables certain optimizations. [ISCA-99 Barua et. al.]

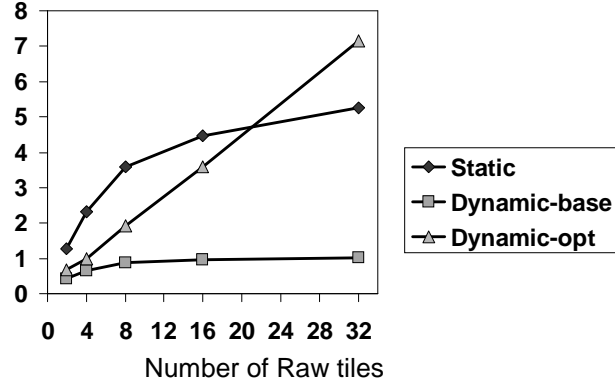
Non-disambiguated accesses done on dynamic network using Software Serial Ordering technique.

40

Why dynamic references ?

Might allow more memory parallelism than static in the best case.

Speedup for Unstructured



41

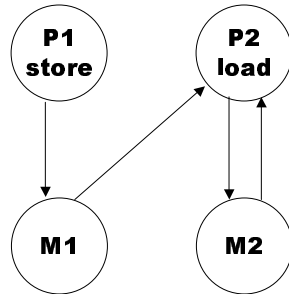
Two kinds of memory-exposed architectures

- Multiple instruction streams, one per bank
 - Raw approach
- Single instruction stream with bank annotations
 - May allow more conventional machines to benefit from Maps.

42

Software Serial Ordering

Method to enforce dependencies between dynamic accesses efficiently.



43

Uses for dynamic accesses

May supplement static accesses. Example use:

- For making “Bad” pointer accesses

44