

18-742

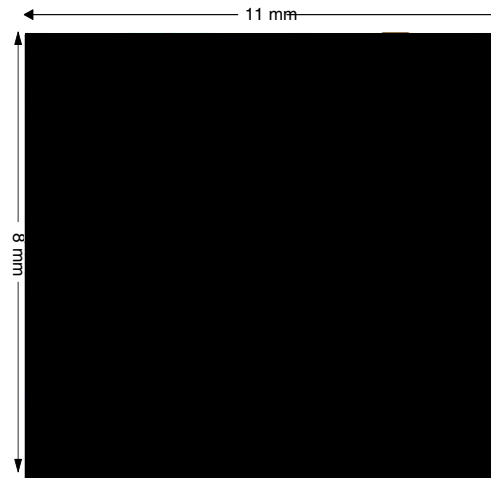
Lecture 23

**Speculative
Threading on
CMPs**

Spring 2005

Prof. Babak Falsafi

<http://www.ece.cmu.edu/~ece742>



Slides developed in part by Prof. Falsafi from Hill, Olukotun, Oplinger and Stets of Carnegie Mellon University, Google, Stanford University, and University of Wisconsin.

Readings

Papers and lecture notes only

Reader 7

- D. M. Tullsen, S. J. Eggers, J. S. Emer, H. M. Levy, J. L. Lo, and R. L. Stamm, *Exploiting Choice: Instruction Fetch and Issue on an Implementable Simultaneous Multithreading Processor*, ISCA 1996.
- J. Lo, L.A. Barroso, S. Eggers, K. Gharachorloo, H. Levy, and S. Parekh, *An Analysis of Database Workload Performance on Simultaneous Multithreaded Processors*, ISCA 1998.

Announcements

Monday:

- We meet at 2:30
- Comp. arch. talk at 4pm in HH 1112

Mike Taylor

RAW Group at MIT

*Scalar Operand Networks:
Enabling Scalable, Parallel Microprocessors*

Speculative Threading

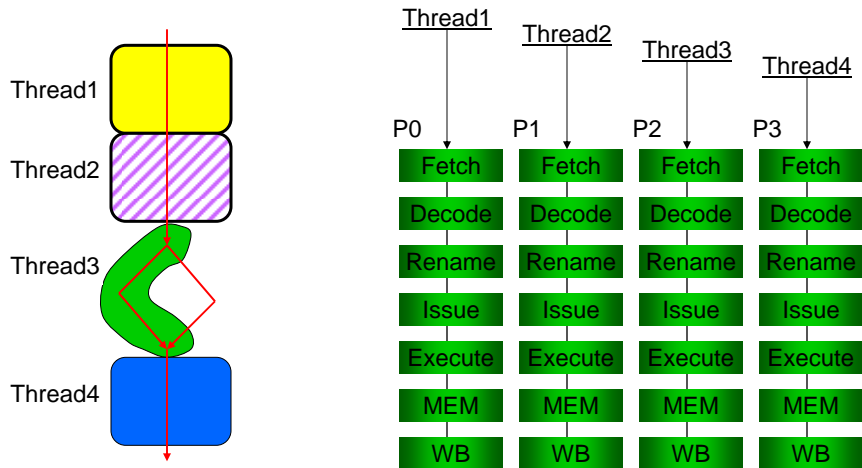
Also known as Speculative Multithreading or Thread-Level Speculation

- Take a sequential program
- Start with the dynamic instruction stream
- Peel off candidate basic blocks
- Execute them speculatively in parallel on different cores
 - Why speculatively?

Lots of academic projects (including Wisc., CMU, Stanford, Illinois)

Two industrial products/prototypes (Fujitsu, Sun)

Multiscalar Execution



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

5

What are good candidate points to peel?

Superscalar's big bottleneck:

- Branch predictability
- Predicts all branches the same way

Key observation:

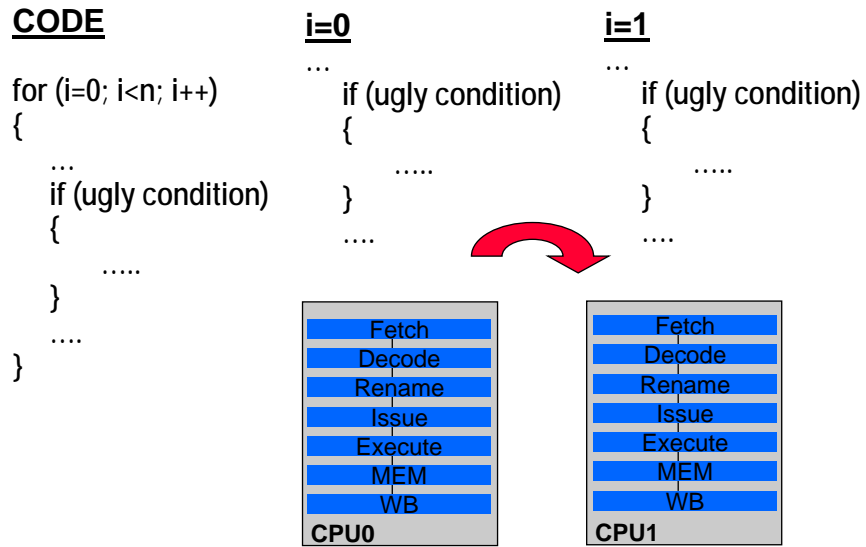
- Loop branches are more predictable than if-then-else branches
- Peel off code at loop branch boundaries
 - If-then-else branches remain within thread
 - If they mispredict locally, they do not affect thread-level control-flow
- Peel off code at function call boundaries
- Compiler annotates candidate thread spawn points

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

6

Example: Loop



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

7

Sequential Execution Semantics (SES)

Do not confuse with sequential consistency

SES states:

- Program outcome should be as if it was run on one processor
- There are SES control-flow and data-flow dependences between threads on multiple cores

But, conventional multiprocessors have

- Independent control flow
- Independent data flow

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

8

Example Data-Flow Dependence

```
for i = 1 to 5
{
    ...
    ... = x
    ...
    x = ...
    ...
}
```

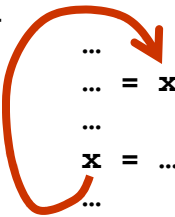
(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

9

Example: Loop-Carried Dependence

```
for i = 1 to 5
{
    ...
    ... = x
    ...
    x = ...
    ...
}
```



(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

10

Example ctd.

|
for

(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

11

Example: Fork

|
for
Iteration 1 | 2 | 3 | 4 |

(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

12

Example ctd.

```
      |  
      for  
Iteration 1 | 2 | 3 | 4 |  
            | | | |  
            ... = x ... = x ... = x ... = x  
            | | | |
```

(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

13

Example ctd.

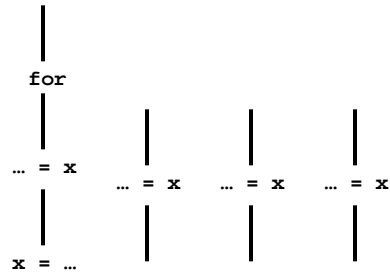
```
      |  
      for  
Iteration 1 | 2 | 3 | 4 |  
            | | | |  
            ... = x ... = x ... = x ... = x  
            | | | |
```

(C) 2005 Babak Falsafi from Adve, Falsafi,
Hill, Lebeck, Reinhardt, Smith & Singh

18-742

14

Example ctd.

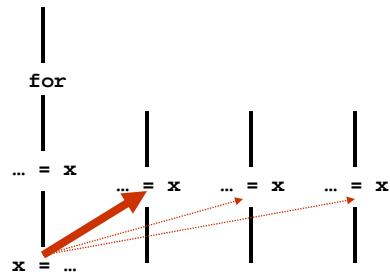


(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

15

Example: Data Hazard

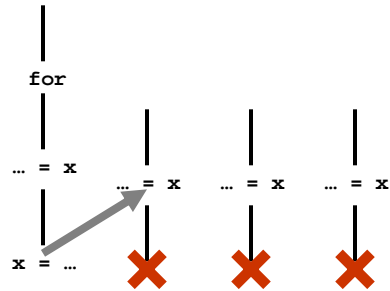


(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

16

Example: Squash

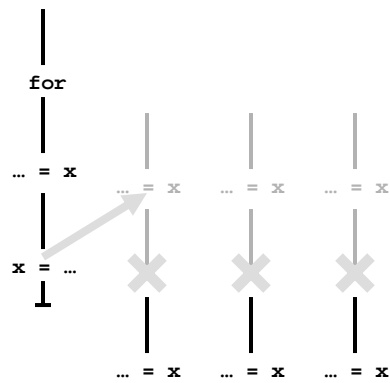


(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

17

Example: Restart / Rollback

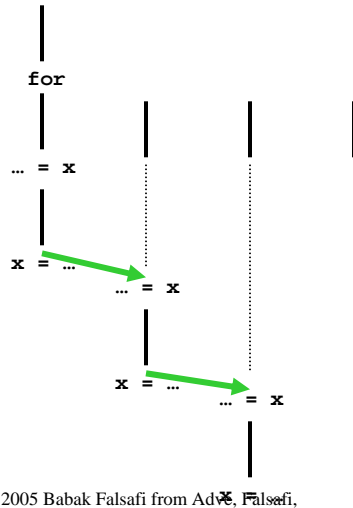


(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

18

Example: Synchronizing the Accesses in HW



X Is either:

- Register allocated
- In memory

Register-allocated dependences:

- Known at compile time
- Compiler passes the info to HW
- Need HW for communication values and synchronizing the register accesses

In-memory dependences:

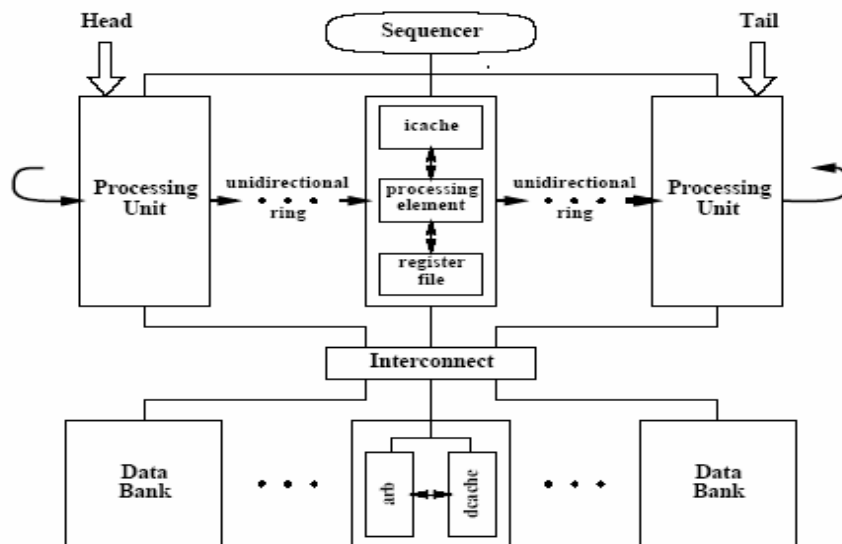
- May not be known at compiler time
- Not clear where the sources/sinks are
- Need more sophisticated HW

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

19

Multiscalar Anatomy



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

20

Register Flow

Reminder: compiler annotates thread spawn points

- **Embedded thread “header” information in binary**
- **Register “def” and “use” masks for each thread**
- **Explicit register release instructions after last “def”**
- **Register values travel on the “register ring” between SES-ordered threads**
- **Upon thread invocation, first “use” of register blocks until a value arrives from a prior thread**

Memory Flow

Much more complicated

Compiler knows nothing

Each core does independent loads/stores out of a shared L1 cache

- **What is wrong with this?**

A special structure called “ARB” tracks all loads/stores from cores

- **Guarantees SES order**

Unlike register def/use synchronization

- **By default, memory operations go speculatively**
- **A misspeculation tagged by ARB rolls back thread**
- **A Memory Dependence Predictor (remember 741?) synchronizes predictable load/store communications to avoid overhead**

Distributed Memory Disambiguation

Conventional cores:

- Have cache hierarchies
- Disambiguate memory in a centralized structure (LSQ)

Multiscalar cores:

- Share L1
- Disambiguate memory in a centralized structure (ARB)

Big deviation from a CMP

- Multiscalar solution not scalable

Subsequent proposals (Wisc, CMU, Stanford, Illinois):

- Per-core cache hierarchy
- Distributed structure for memory disambiguation

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

23

What is so hard about memory disambiguation across cores?

Think private writeback L1s and a shared L2

- Need coherence among L1s

But, also need SES memory dependence order

How do we change the coherence protocol to implement SES dependence order?

Requirements (changes to CMP):

- Common case of cache hit should go fast
- Cache misses should not take much longer than in a CMP
- No sequential searches in caches upon thread invocation, completion or rollback
- Coherence protocol needs tens of states (see SVC by Gopal et al.)

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

24

Stanford Hydra

Made the problem much simpler

- At a performance cost?

No need for register communication

- Communicate register values through memory
- As in conventional register load/spills

Make the L1 caches writethrough

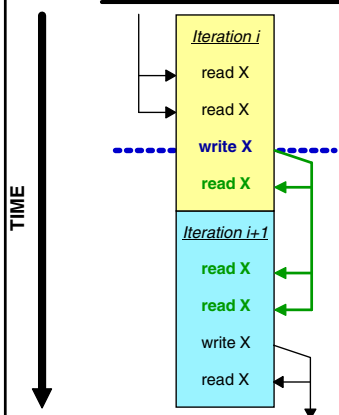
(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

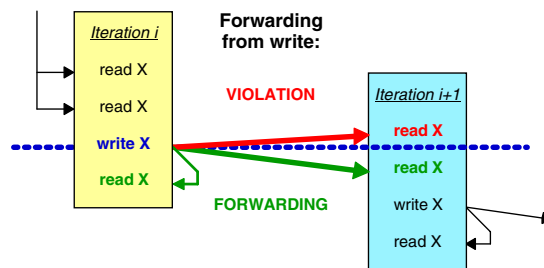
25

Data Speculation in Hydra: Requirements (I)

Original Sequential Loop



Speculatively Parallelized Loop



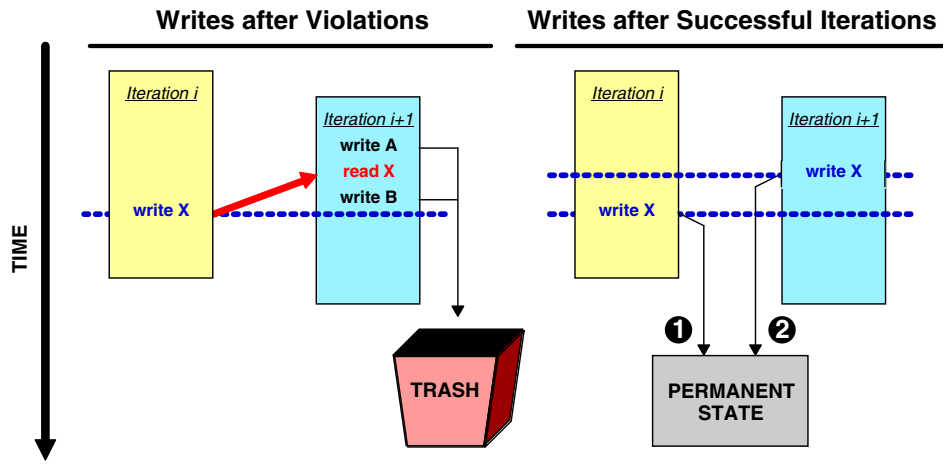
- Forward data between parallel threads
- Detect violations when reads occur too early

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

26

Data Speculation in Hydra: Requirements (II)



- Safely discard bad state after violation
- Correctly retire speculative state

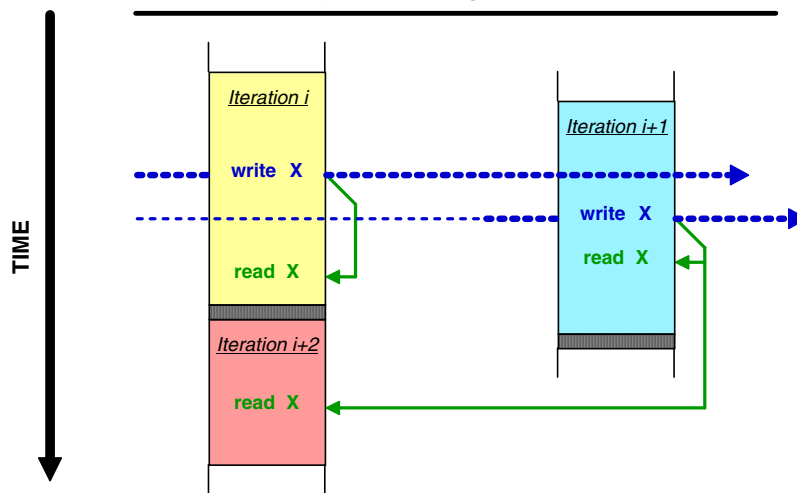
(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

27

Data Speculation in Hydra: Requirements (III)

Multiple Memory "Views"



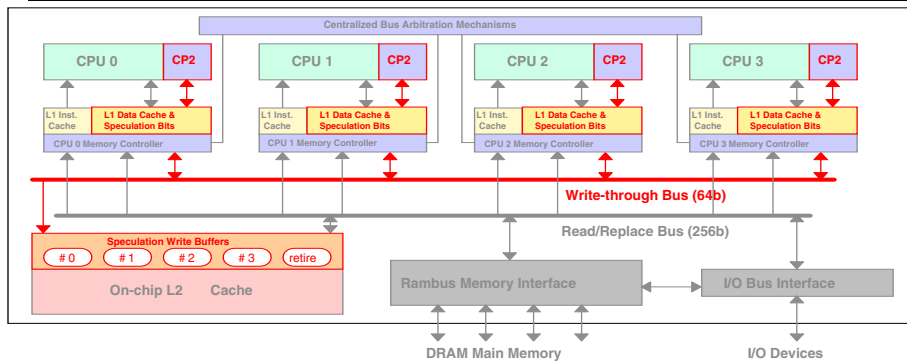
- Maintain multiple "views" of memory

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

28

Hydra Speculation Support



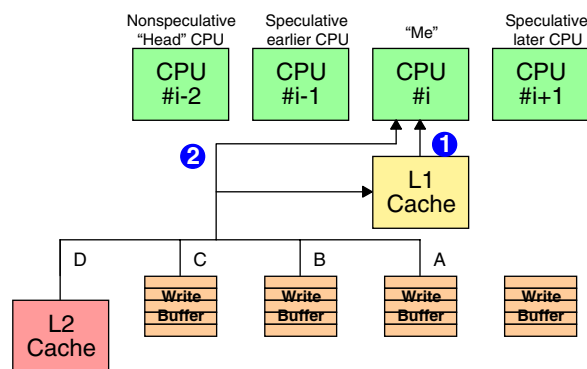
- ❑ Write bus & L2 buffers forward
- ❑ "Read" L1 tags for violations, "Dirty" L1 tags and wbuff provide backup
- ❑ Wbuff reorder & retire spec. state
- ❑ Speculation coprocessors to control threads

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

29

Speculative Reads



L1 hit

Read bits are set

L1 miss

L2 and write buffers are checked in parallel

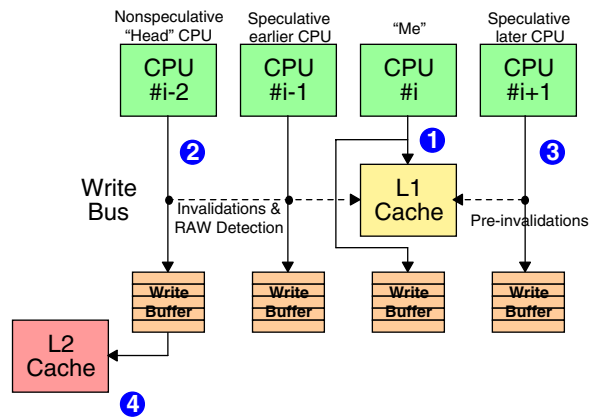
Latest written values from a cache block are pulled in by priority encoders on each byte (priority A-D)

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

30

Speculative Writes



- A CPU writes to its L1 cache & write buffer
- "Earlier" CPUs invalidate our L1 & cause RAW hazard checks
- "Later" CPUs just pre-invalidate our L1
- Non-speculative write buffer drains out into the L2

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

31

Creating Speculative Threads

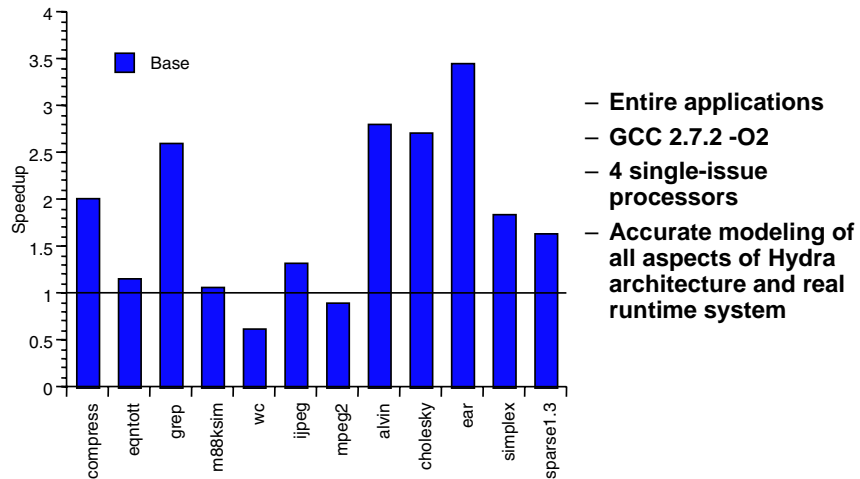
- **Speculative loops**
 - `for` and `while` loop iterations
 - Typically one speculative thread per iteration
- **Speculative procedures**
 - Execute code after procedure speculatively
 - Procedure calls generate a speculative thread
- **Compiler support**
 - C source to source translator
 - `Pfor`, `pwhile`
 - Analyze loop body and globalize any local variables that could cause loop-carried dependencies

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

32

Base Speculative Thread Performance



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

33

Improving Speculative Runtime System

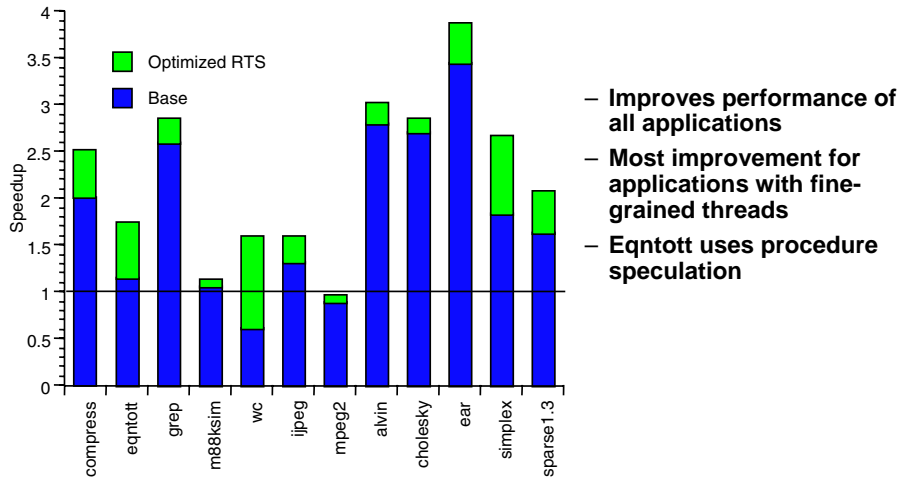
- **Procedure support adds overhead to loops**
 - Threads are not created sequentially
 - Dynamic thread scheduling necessary
 - Start and end of loop: 75 cycles
 - End of iteration: 80 cycles
- **Performance**
 - Best performing speculative applications use loops
 - Procedure speculation often lowers performance
 - Need to optimize RTS for common case
- **Lower speculative overheads**
 - Start and end of loop: 25 cycles
 - End of iteration: 12 cycles (almost a factor of 7)
 - Limit procedure speculation to specific procedures

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

34

Improved Speculative Performance



- Improves performance of all applications
- Most improvement for applications with fine-grained threads
- Eqntott uses procedure speculation

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

35

Feedback and Code Transformations

- **Feedback tool**
 - Collects violation statistics (PCs, frequency, work lost)
 - Correlates read and write PC values with source code
- **Synchronization**
 - Synchronize frequently occurring violations
 - Use non-violating loads
- **Code Motion**
 - Find dependent load-stores
 - Move loads down in thread
 - Move stores up in thread

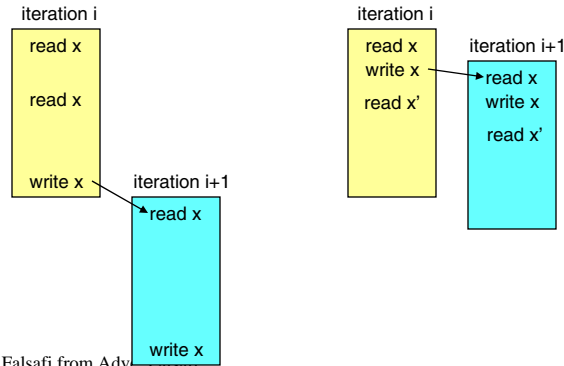
(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

36

Code Motion

- Rearrange reads and writes to increase parallelism
- Delay reads and advance writes
- Create local copies to allow earlier data forwarding

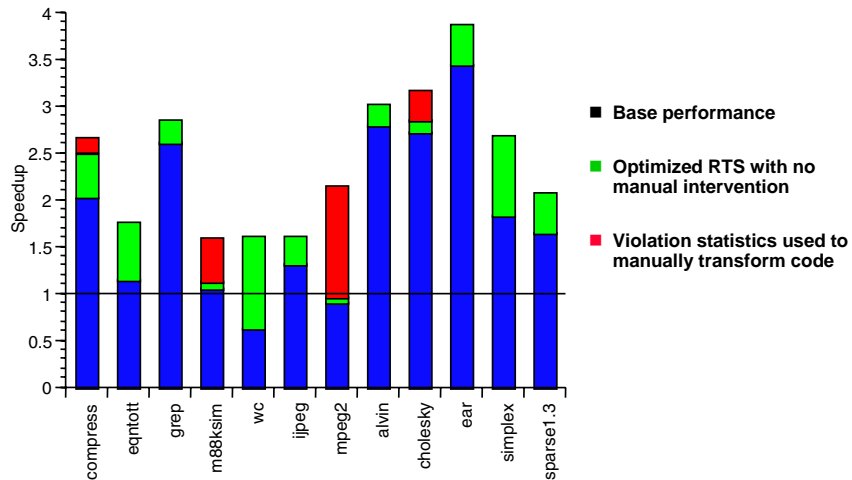


(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

37

Optimized Speculative Performance



(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

38

Size of Speculative Write State

- Max size determines size of write buffer for max performance
- Non-head processor stalls when write buffer fills up
- Small write buffers (< 64 lines) will achieve good performance

Max no. lines of write state

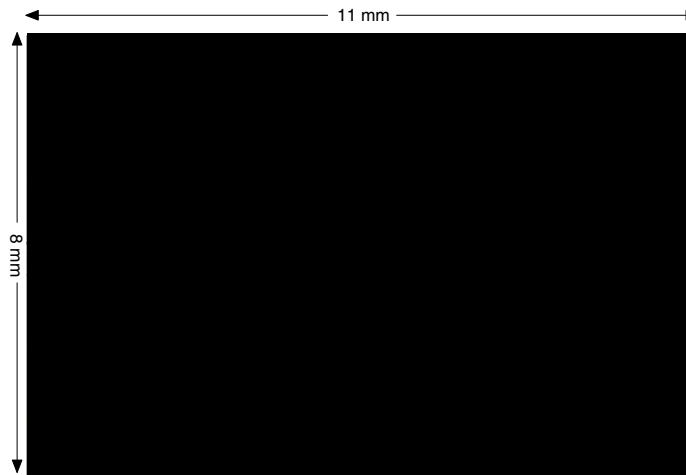
compress	24
eqntott	40
grep	11
m88ksim	28
wc	8
jpeg	32
mpeg	56
alvin	158
cholesky	4
ear	82
simplex	14

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742 32 byte cache lines

39

Hydra Prototype



- Design based on Integrated Device Technology (IDT) RC32364
- 88 mm² in 0.25 μ m with 8 KB I, D and 128 KB L2

(C) 2005 Babak Falsafi from Adve, Falsafi, Hill, Lebeck, Reinhardt, Smith & Singh

18-742

40