Managing Interconnect Delay with Architectural and Compiler Techniques

Walt Fish
Chris Flesher
David Suksumrit
Allen Wan

ELEC 525
20 April 2004

Introduction

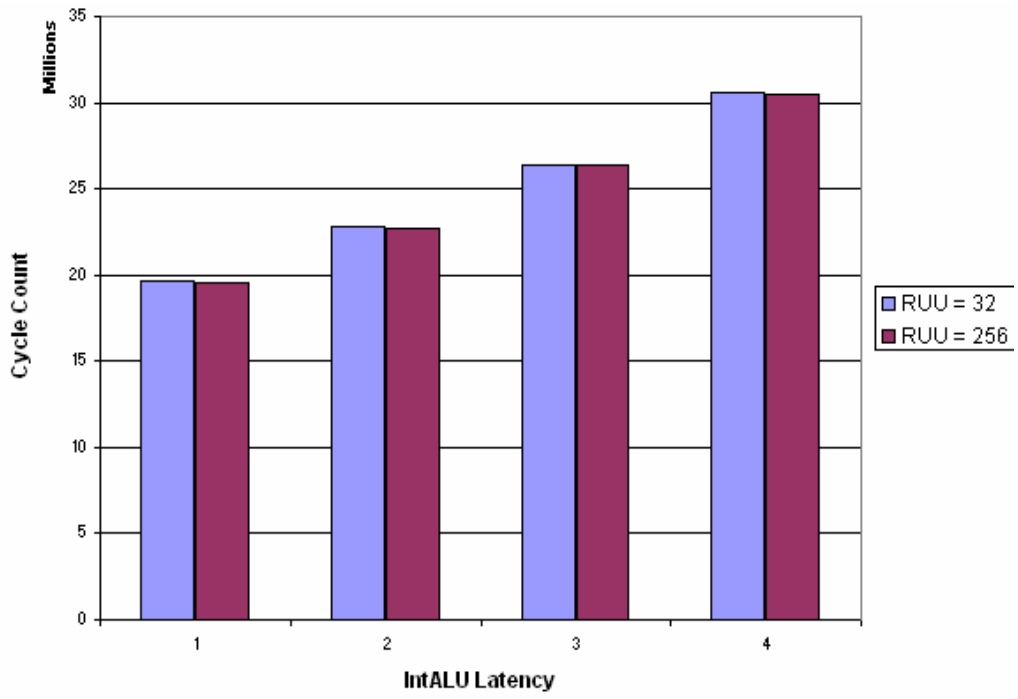
- ITRS 2002 update “Design and layout solutions are needed” to address increasing interconnect delay
- Delay between Register/ALU combinations no longer equal
- By exposing interconnect delay to ISA, the compiler can acknowledge that delay and optimize for it

Hypothesis

- We believe we can address the issue of increasingly dominant interconnect delay by statically scheduling instructions in FU’s that are physically closer to their source and destination registers

Motivation - Redux

- Increasing total execution latency
- Linear increase in delay results in linear increase in execution time
- RUU not filling
- Few cache misses

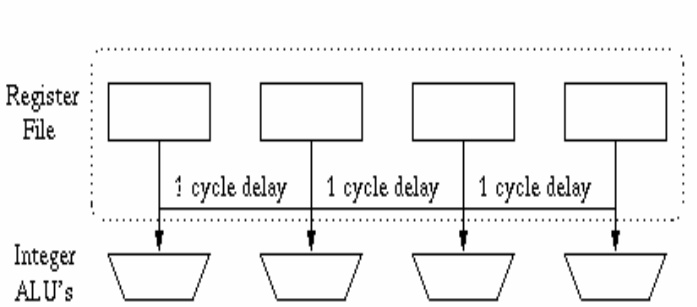


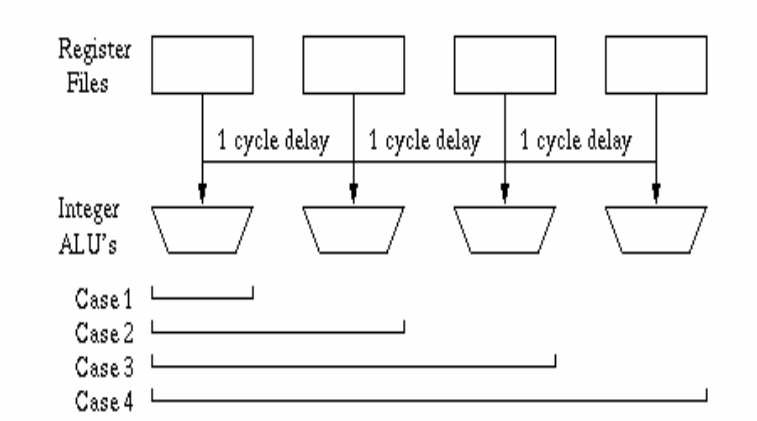
Methodology Overview

- Hack SimpleScalar to execute instructions with different latencies based on interconnect delays between FU's and registers
- Increase ALU/register proximity via banking
  - Manually rename registers
  - Hand reorder assembly

Architecture

- Partition register file
  - Move registers closer to FU's
  - Decrease interconnect delay



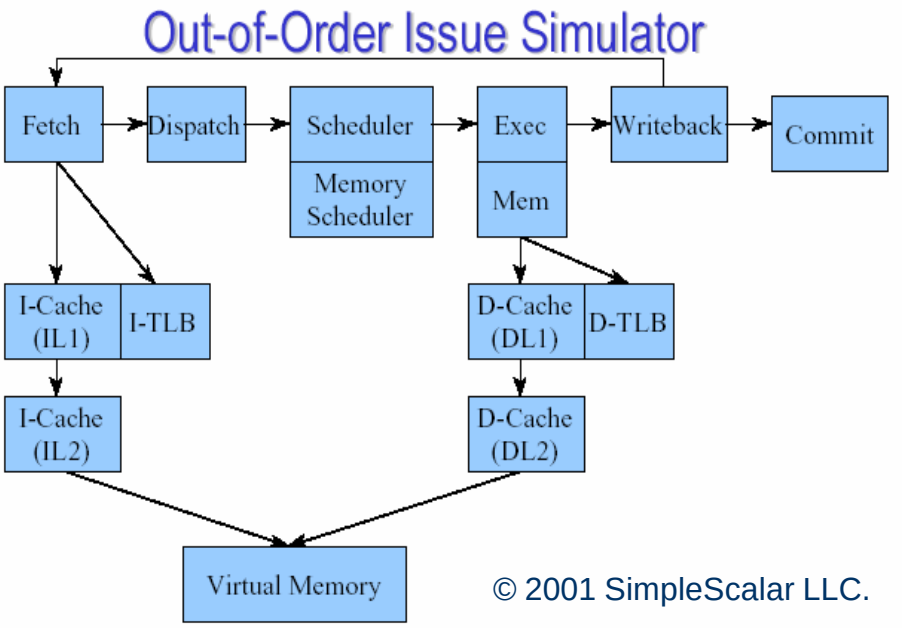


Configuration File

- Configuration file
  - 4 wide issue, 4 commit
  - 128 entry reorder buffer
  - Perfect branch prediction
  - 16KB L1 D-cache, 16KB L1 I-cache, 256KB L2 unified cache

Hacking SimpleScalar

- Modify SimpleScalar dispatch functions
  - Add FU classes with differing latencies
  - Change class of instruction according to annotated assembly



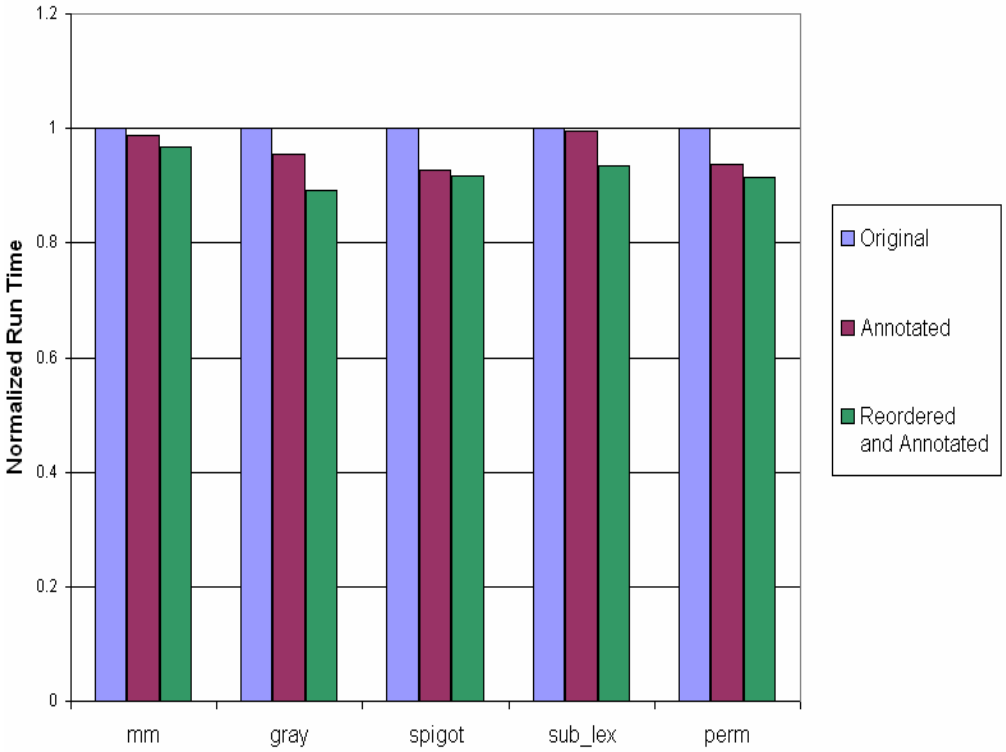
Annotated Instructions

- Annotate integer micro-benchmarks
  - Matrix manipulation
  - Gray code
  - Pi
  - Lexicographic order
  - Perm.c – “Short and bewildering recursive method”

```
sll/b    $9,$15,2
lw/a     $15,32($fp)
addu/b   $9,$9,$15
lw/a     $9,0($9)
lw/a     $16,16($fp)
sw/b     $9,0($2)
sll/b    $8,$16,2
lw/a     $15,32($fp)
lw/a     $16,16($fp)
addu/b   $8,$8,$15
addu/a   $15,$16,1
sw/b     $15,0($8)
j        $L6
```

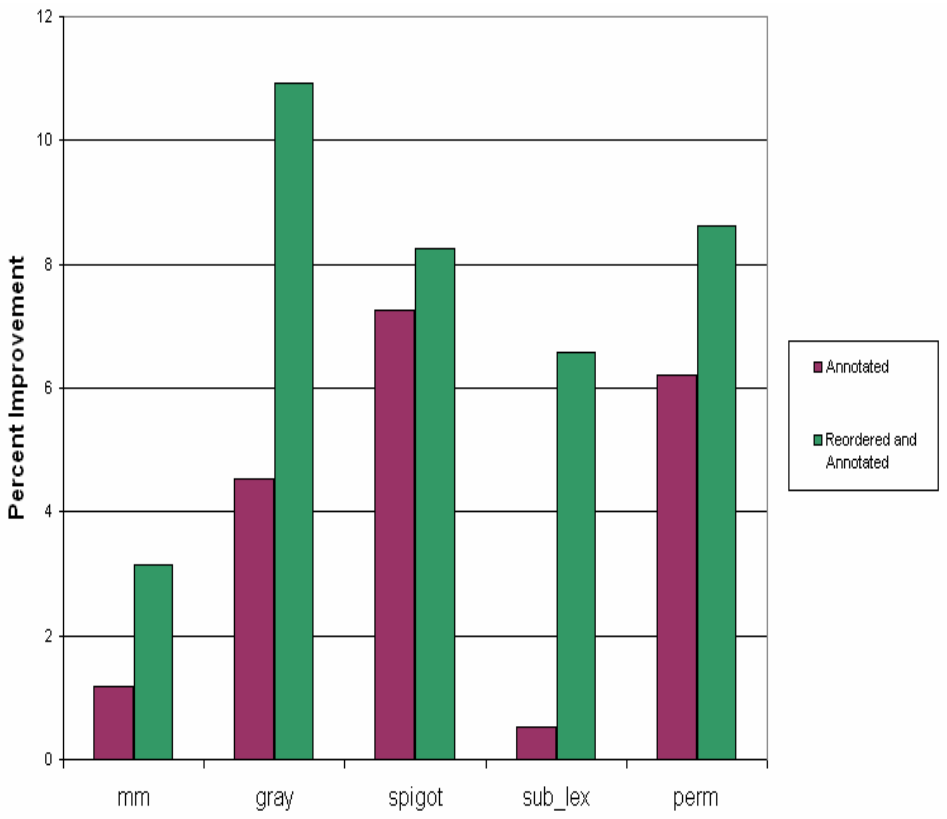
Results

- 4% average improvement through annotating only
- 7.5% average improvement through annotating, reordering, renaming



Results

- Annotations similar to dynamic execution
- Annotations with renaming and reordering analogous to static scheduling



Conclusion

- Issues raised in hypothesis valid
- Proposed solution demonstrates value of addressing interconnect delay
- We need architecture and compilers to take advantage of exposed interconnect delays
  - Expose delay to ISA
  - RUU’s can not always hide latency
- Hand-reordering code is not fun