Motivation

- Base algorithms
  - In order (low performance, low complexity)
  - Out-of-order (high performance, huge complexity)
    - Roadmap to high performance
      - Big instruction windows
      - Lots of functional units
      - Due to a broadcast and select mechanism
      - Instruction issue in the critical path
      - Long delays due to hardware size and complexity

- Is there something in between?
  - (high performance, low complexity)

Talk Outline

- Characteristics exploited
  - output register usage
  - latency of operation known

- First-Use Scheme
  - Block diagram
  - Presentation and evaluation of the scheme

- Distance Scheme
  - Block diagram
  - Presentation and evaluation of the scheme

Conclusions

Characteristics exploited

- First-Use scheme → Output register usage
  - 22% of the values produced by the SpecInt95 and
  - 25% of the values produced by the SpecFP95 are read
    more than once.
  - ⇒ Can avoid associative look-up if the non-ready instructions are
    kept in a table index by physical register.

- Distance scheme → Latency of operation known
  - Every FU latency is known (except for memory accesses)
  - ⇒ Output register availability time can be computed
  - ⇒ Instructions can be "tagged" with its issue cycle

Experimental Framework

- Simulator
  - based on SimpleScalar
  - + instruction queues
  - + physical registers
  - + issue mechanisms
  - 2 cluster architecture (INT & FP)

- Benchmarks
  - 8 Spec Int95

- Configuration
  - 8-way issue (4 int + 4 FP)
  - ROB of 64 entries

First-Use Scheme (I)

- Philosophy
  - Avoid an associative lookup by keeping the instructions with
    its producers.
  - Just one instruction is kept for physical register. When the
    register becomes available the instruction is released and it
    is sent to the ready queue.
  - Hopefully, most of the instructions will be ready when
    dispatching or (as the statistics say) the first use of its source
    operands.
If the availability time of the source registers is not known the
if
Similar to a "dynamic" VLIW.
When an instruction knows in which cycle it will be executed
if
else stall dispatch
if
ready queue
First-use table
else
- Submit and dispatch when all
- else stall dispatch
compute issue time and write
Schedule instructions according to cycle where the operands
I-buffer
else
- Submit to Wait-
3
- Submit to Wait-
if
3
Special case: memory loads
above.
if
detect that their operand(s) availability time is
When the instructions in the wait queue
then
and
- Submit to Wait-
and
then
then
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wait-
- Submit to Wa...
**Distance Scheme (iv)**

**First-Use vs. Distance (i)**

**First-Use vs. Distance (ii)**

**Conclusions**

- New alternatives for issue logic
  - Can eliminate associative search in the issue logic (20% IPC loss).
  - Similar IPC with one eighth of the associative search.
  - Significant reduction in the complexity of the issue logic, thus, possibly power consumption.
  - Significant performance improvement if cycle time factored in.

**Future Work**

- **First-Use Scheme**
  - Introduce associativity in the First-use table
    - keep >1 instruction per register

- **Distance Scheme**
  - Assume that memory accesses have a deterministic latency
    - Wait queue moves below the issue mechanism

- **More benchmarks**
  - Evaluate the SpecFP95 as well.