ICS´01

Reducing the Complexity of the Issue Logic

Ramon Canal and Antonio González

Universitat Politècnica de Catalunya - Barcelona
{rcanal, antonio}@ac.upc.es

UPC

ICS´01

Motivation

- **Base alternatives**
  - In-order (limited performance, low complexity)
  - Out-of-order (high performance, huge complexity)
    - Roadmap to high performance
      - Large instruction windows
      - Lots of functional units
    - Associative lookups in issue
      - Due to a broadcast and select mechanism
    - Instruction issue in the critical path
    - Long delays due to hardware size and complexity
- **Can we have the best of both?**
  (high performance, low complexity)

UPC

ICS´01

Talk Outline

- **Characteristics exploited**
  - output register usage
  - latency of operation known
- **N-Use Scheme**
  - Block diagram
  - Presentation and evaluation of the scheme
- **Deterministic Latency Scheme**
  - Block diagram
  - Presentation and evaluation of the scheme
- **Conclusions**

UPC

ICS´01

Characteristics exploited

- **N-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.
- **Deterministic Latency scheme → Latency of operations 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
    ⇒ Assume memory accesses have a fixed latency

UPC

ICS´01

Experimental Framework

- **Simulator**
  - based on SimpleScalar
  - + instruction queues
  - + physical registers
  - + issue mechanisms
  - 2 cluster architecture (INT & FP)
- **Benchmarks**
  - Spec95 benchmark suite
- **Configuration**
  - 8-way issue (4 int + 4 FP)
  - ROB of 64 entries

UPC

ICS´01

N-Use Scheme (i)

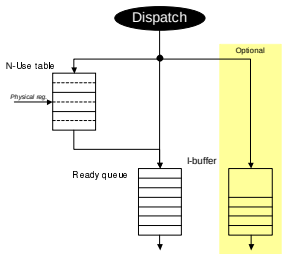
- **Philosophy**
  - Avoid an associative lookup by keeping the instructions with its producers.
  - Just N instructions are kept for physical register. When the register becomes available the instruction/s is/are sent to the ready queue.
  - Hopefully, most of the instructions will be ready when dispatching or (as the statistics say) within the first N uses of its source operands.

UPC

ICS'01

N-Use Scheme (ii)

- Procedure
  - If *operand(s) ready* then
    - ready queue
  - if *operand(s) not ready* and *first n-use (of all the operands)* then
    - N-use table
  - else
    - if *I-buffer* and *I-buffer not full* then
      - I-buffer
    - else stall dispatch



UPC

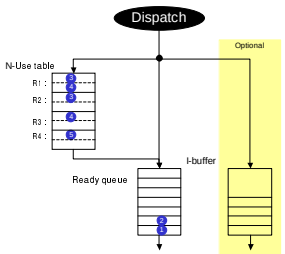
ICS'01

N-Use Scheme (iii)

- Example

1: LD R1, 0(R4)
2: LD R2, 0(R5)
3: ADD R3, R1, R2
4: MUL R4, R1, R3
5: ST0(R6), R4

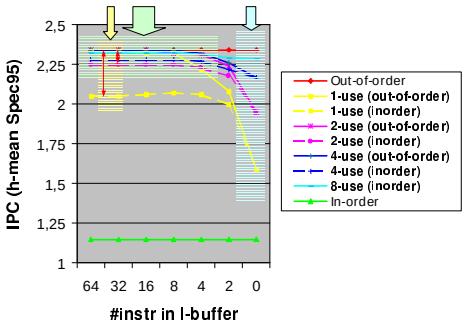
R4, R5 and R6 are assumed to be ready



UPC

ICS'01

N-Use Scheme (iv)



UPC

ICS'01

Deterministic Latency (i)

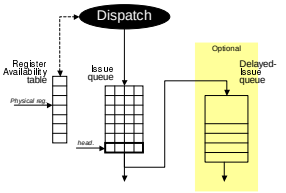
- Philosphy
  - Schedule instructions according to cycle where the operands will be available and when there is a free functional unit.
  - When an instruction knows in which cycle it will be executed it updates the availability cycle of its output register (if any).
  - Memory accesses are assumed to have a fixed latency (I.e. hit in L1 or hit L2)
  - If the instruction is meant to execute later the instruction is kept apart.
  - Similar to a "dynamic" VLIW
  - Can potentially overlap the issue time computation with the decode/renaming stage (not considered in this work)

UPC

ICS'01

Deterministic Latency (ii)

- Procedure
  - compute issue time and write output register availability time
  - when issuing a non-ready instruction
    - if the Delayed-Issue queue is not full then
      - send instruction to the Delayed-Issue queue
    - else stall issue



UPC

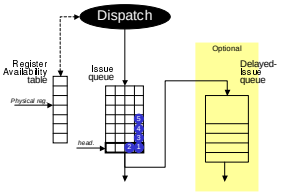
ICS'01

Deterministic Latency (iii)

- Example

1: LD R1, 0(R4)
2: LD R2, 0(R5)
3: ADD R3, R1, R2
4: MUL R4 R1, R3
5: ST0(R6), R4

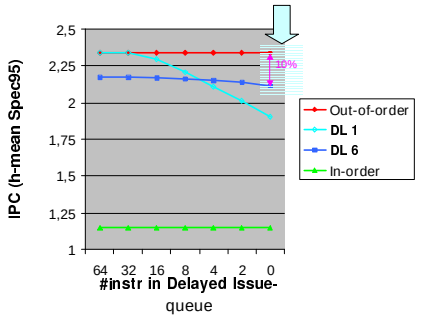
R4, R5 and R6 are assumed to be ready



UPC

ICS'01

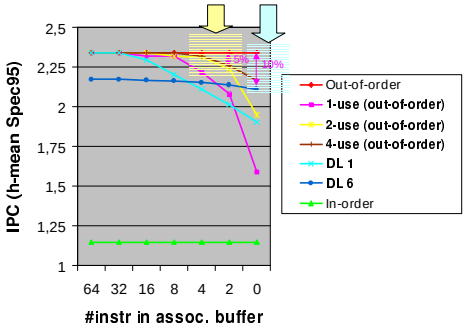
Deterministic Latency (iv)



UPC

ICS'01

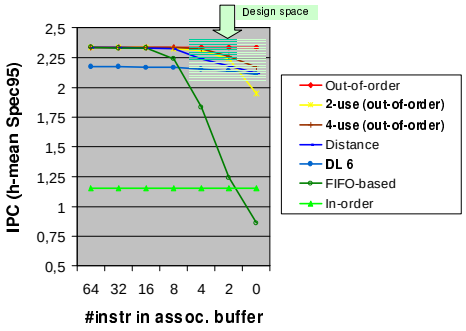
N-Use vs. DL



UPC

ICS'01

N-Use vs. DL vs. Other



UPC

ICS'01

Conclusions

- **New alternatives for issue logic**
  - Can eliminate associative search in the issue logic (10% IPC loss).
  - Similar IPC with one sixteenth 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.

UPC

ICS'01

Q & A

Get corrected version at:

http://people.ac.upc.es/rcanal

UPC