Microprocessor Microarchitecture

Limits of Instruction-Level Parallelism

Lynn Choi

Dept. Of Computer and Electronics Engineering

Complexity of Superscalar Processors

- ***Baseline Superscalar Pipeline Model***

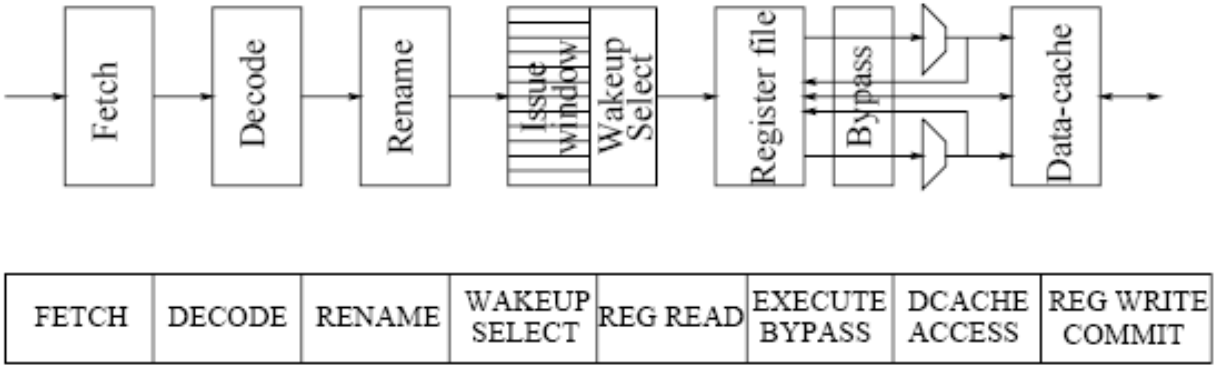


Figure 1: Baseline superscalar model.

- ***Each pipe stage has a structure whose complexity is a function of issue width***
  - Fetch (Branch Predictor Throughput, Taken Branches)
  - Decode/Rename (Renaming & Dependency Check)
  - Register Read (Wakeup and Select)
  - Execution (Bypass Logic)

Pipeline Complexity: Renaming

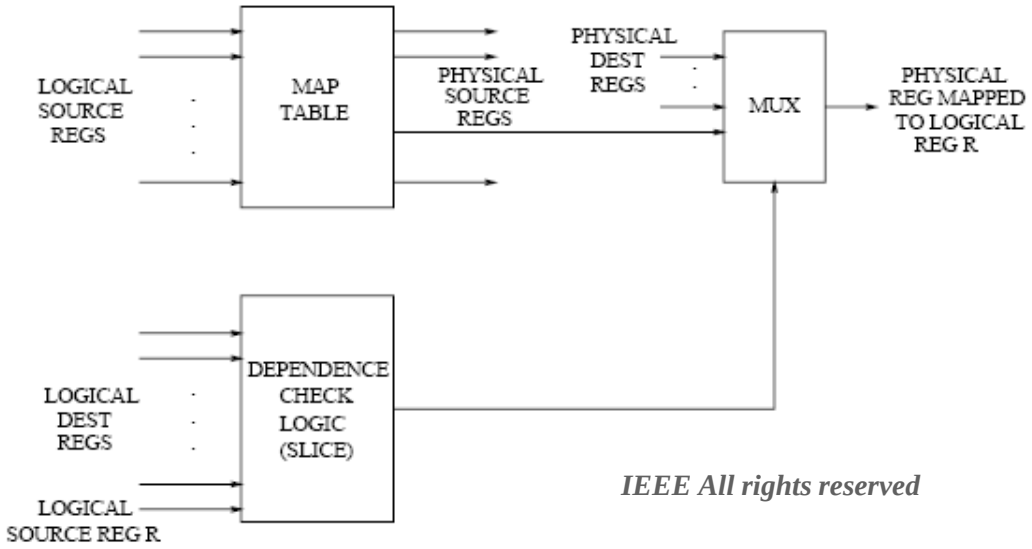


Figure 2: Register rename logic.

- *Structure*
  - Map table
    - RAM scheme (MIPS R10000)
      - Logical register # is used as an index to access the physical register entry
    - CAM scheme (DEC 21264)
      - The number of entries in the CAM is equal to the number of physical registers
      - CAM is less scalable than RAM since # physical registers increases as the issue width
  - Dependency check logic
    - The number of comparisons required for the dependency check increases quadratically as the issue width increases
- *Delay*
  - $T_{decode}, T_{wordline}, T_{bitline} = c0 + c1 \times IW + c2 \times IW^2$

Pipeline Complexity: Wakeup

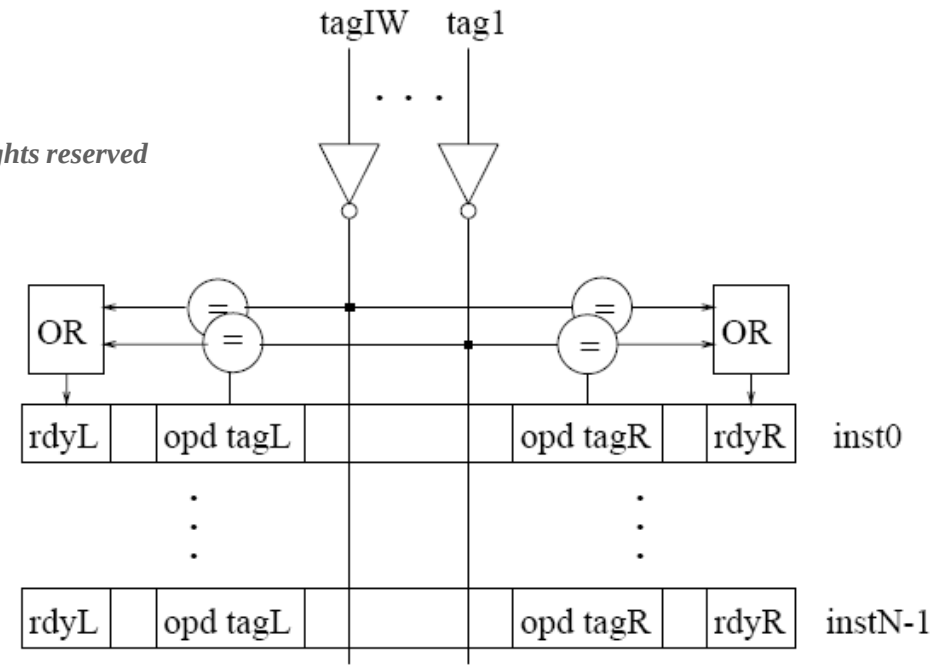


Figure 4: Wakeup logic.

- *Structure*
  - Issue Window is a CAM array
  - Each entry of the CAM has *2 x IW* (issue width) comparators
- *Delay* = $T_{tagdrive} + T_{tagmatch} + T_{matchOR}$
  - The time taken to drive the tags depends on the length of the tag lines and the number of comparators on the tag lines
  - $T_{tagdrive} = c0 + (c1 + c2 \times IW) \times WINSIZE + (c3 + c4 \times IW + c5 \times IW^2) \times WINSIZE^2$
  - $T_{tagmatch}$, $T_{matchOR} = c0 + c1 \times IW + c2 \times IW^2$

Pipeline Complexity: Select

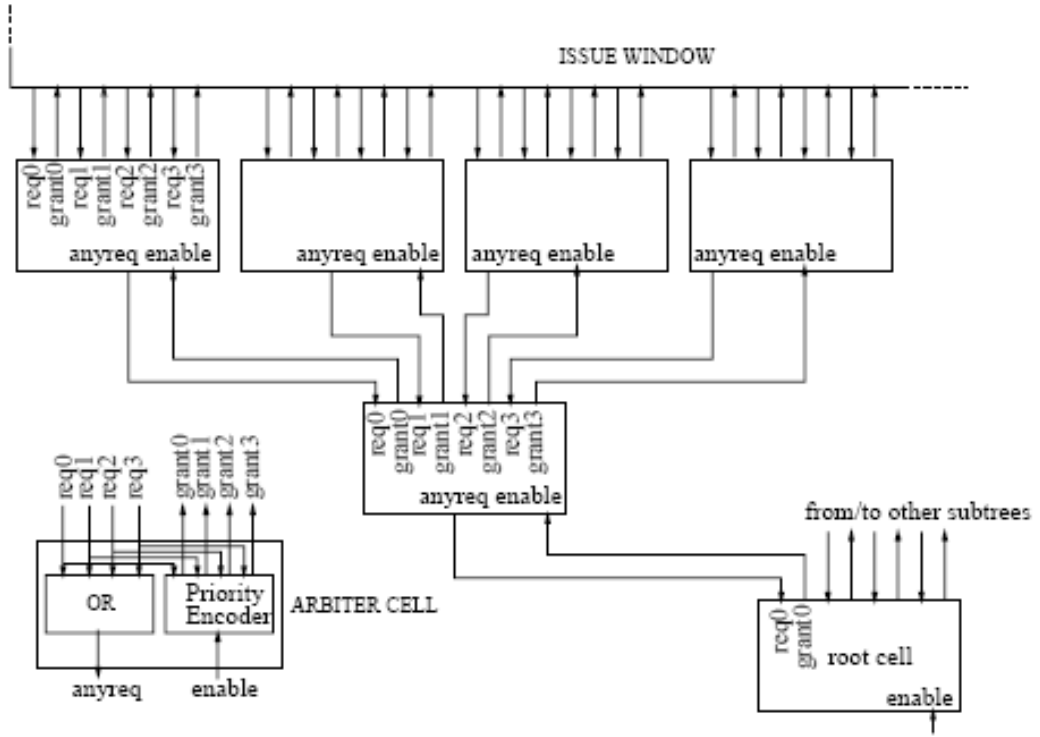


Figure 7: Selection logic.

- *Structure*
  - Selection logic consists of a tree of arbiters. request-grant mechanism.
  - Selection policy : location based (left most entry have the highest priority)
- *Delay* = $T_{selection} = c0 + c1 \times log4(WINSIZE)$
  - Delay depends on the height of the arbitration tree = log4(WINSIZE)
  - Delay increases logarithmically with window size

Pipeline Complexity: Bypass Logic

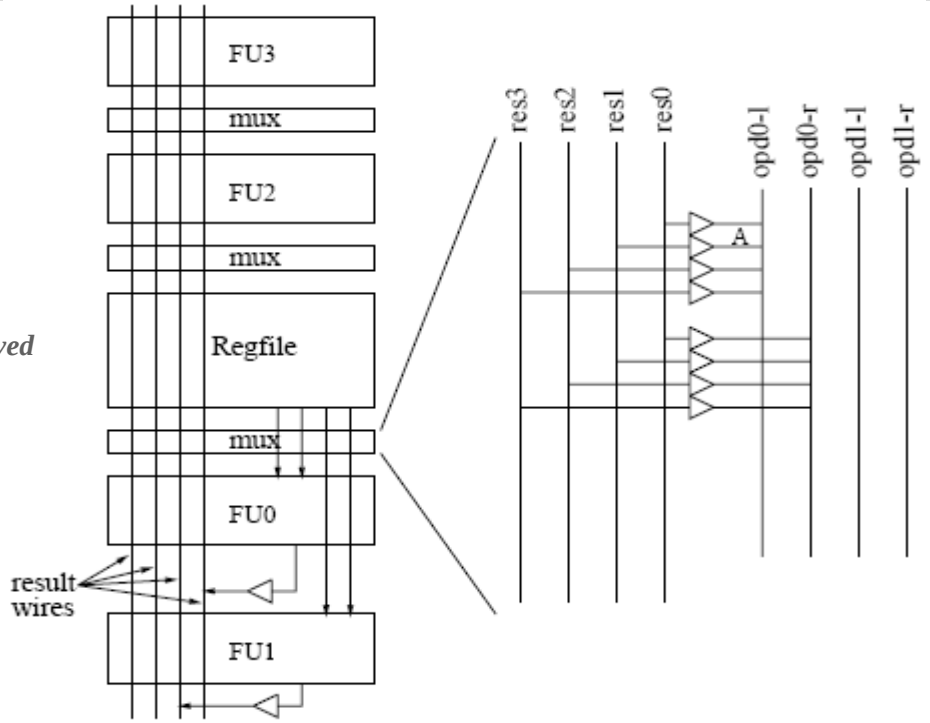


Figure 9: Bypass logic.

- *Structure*
  - The number of bypass paths is determined by the depth of the pipeline and issue width
  - Nbypass_paths = 2 x IW^2 x S : (S = # pipeline stages after the 1st result stage)
  - Datapath : buffers are used to drive the bypass values
  - Control : controlling the operand MUXes
- *Delay:*
  - T_{bypass} = 0.5 x Rmetal x Cmetal x L^2 *where* L *is the length of the result wires.*
  - Increasing issue width increases the length of the result wires.

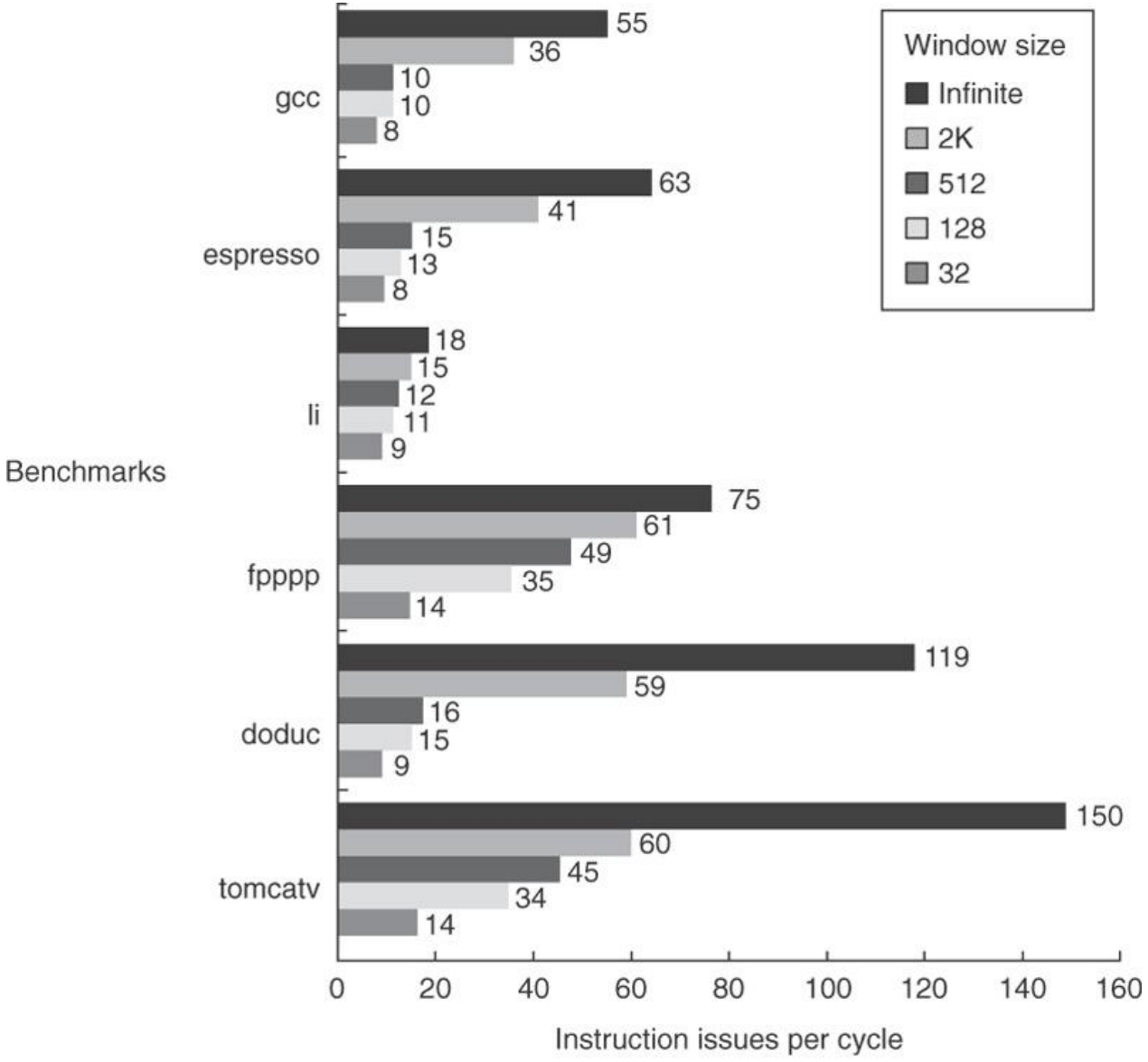
Pipeline Complexity: Summary

- The window logic and the bypasses seem to pose the largest problems
- 4-way superscalar
  - Wakeup & selection logic has the greatest delay among all the structures considered
  - Wakeup and select together constitute what appears to be an atomic operation.
  - If they are divided into multiple pipeline stages, dependent instructions cannot be issued in consecutive cycles.
- 8-way superscalar
  - The bypass delay grows by a factor of over 5, and is now worse than the (wakeup + select) delay
  - ***Bypass delay can easily become a bottleneck for wider issue widths.***

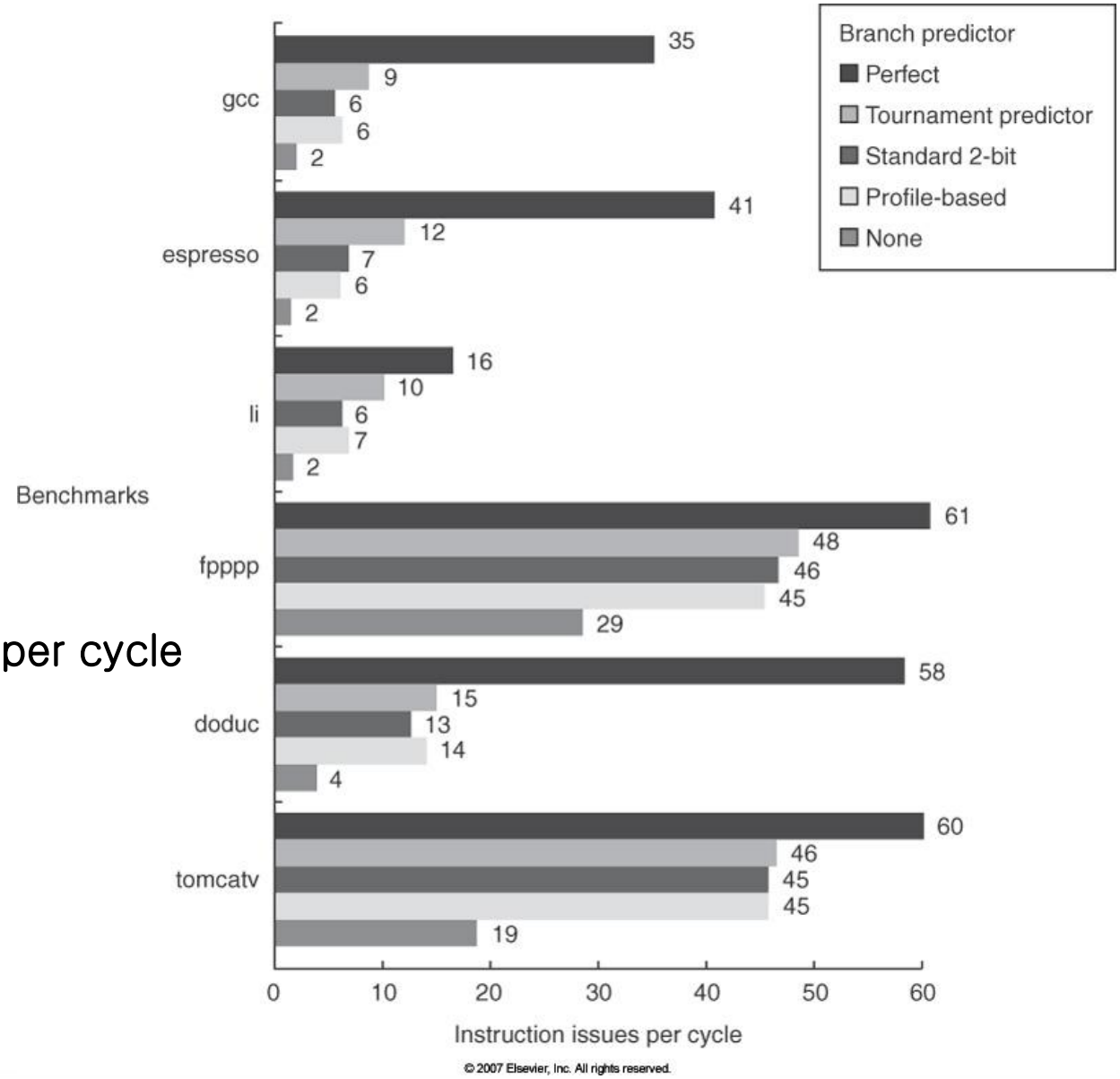
Limitations of ILP

- ***Assume an ideal processor model***
  - Register renaming
    - Infinite number of physical registers
    - All WAW and WAR hazards can be removed
  - Branch prediction
    - Perfect branch prediction
  - Jump prediction
    - All jumps are perfectly predicted
  - Memory address alias analysis
    - All memory addresses are known exactly
  - Perfect caches
    - All memory accesses take 1 clock cycle

Impact of Window Size

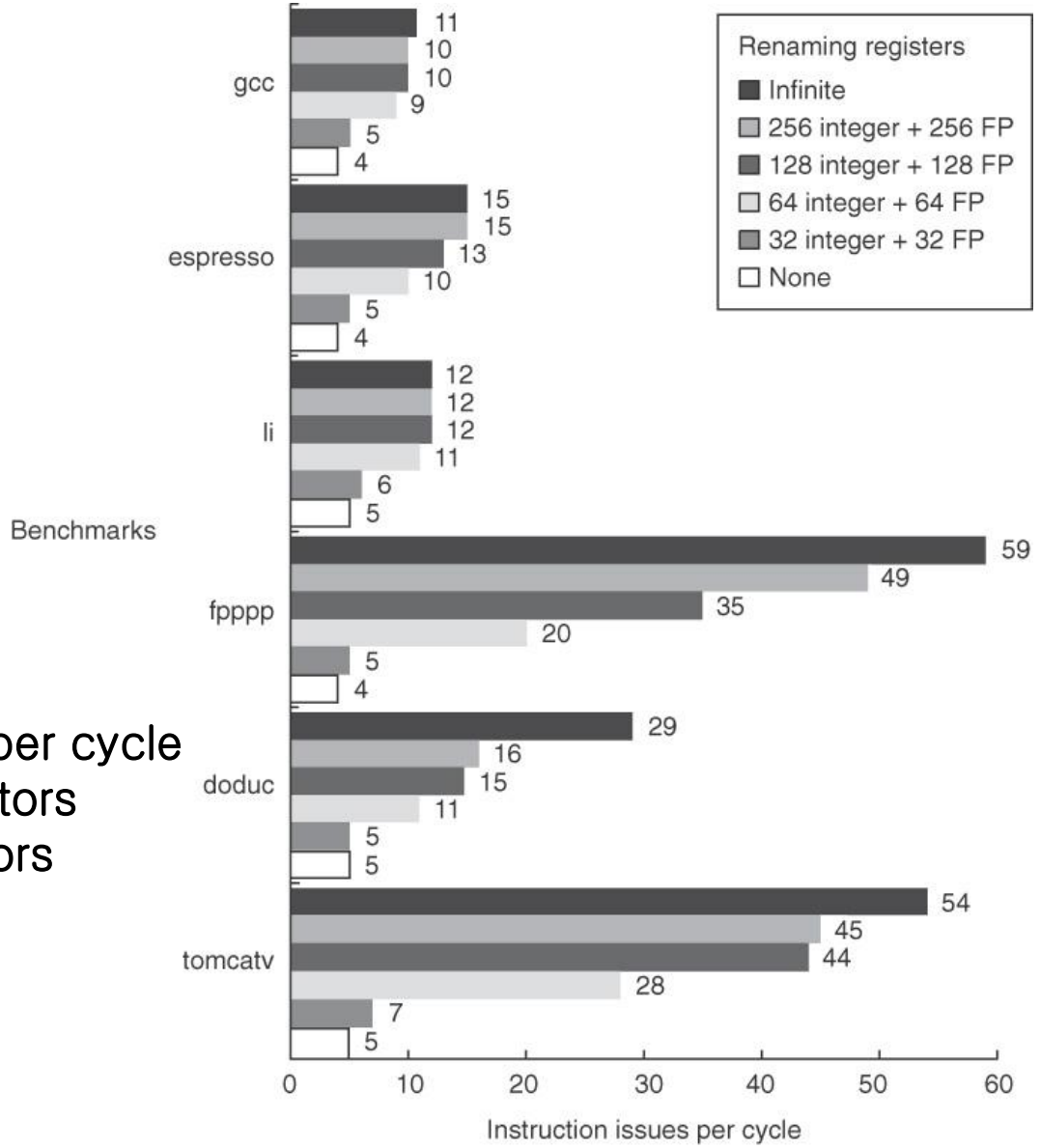


Impact of Branch Prediction



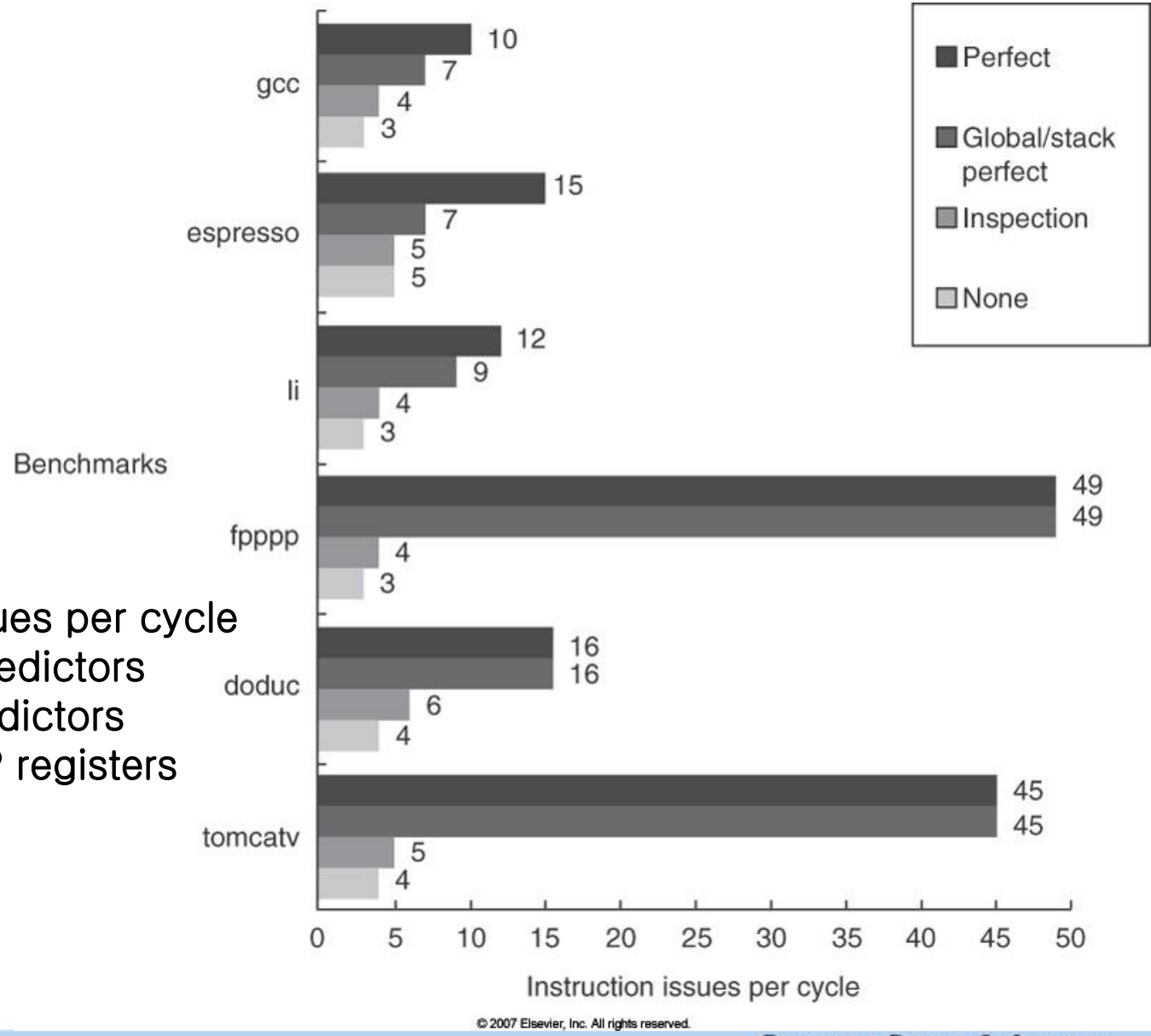
2K window
Max. 64 instruction issues per cycle

Impact of Finite Registers



2K window
Max. 64 instruction issues per cycle
8K entry tournament predictors
2K jump and return predictors

Impact of Imperfect Alias Analysis



2K window
Max. 64 instruction issues per cycle
8K entry tournament predictors
2K jump and return predictors
256 integer and 256 FP registers

Limits of Multiple-Issue Processors: Revisited

- ***Doubling the issue rate above the current 3-6 issue, i.e. 6-12 issue requires***
  - Issue 3-4 data memory accesses per cycle
  - Resolve two or three branches per cycle
  - Rename and access more than 20 registers per cycle
  - Fetch 12-24 instructions per cycle
  - ➔ The complexity of implementing these capabilities would sacrifice the maximum clock rate
- ***Another issue is power!***
  - Modern microprocessors are primarily power limited.
    - *Static power grows proportionally to the transistor count*
    - *Dynamic power is proportional to the product of the number of switching transistors and the switching rate*
  - Microprocessors trying to achieve both a high IPC and a high CR must switch more transistors and switch them faster!

Limits of Multiple-Issue Processors

- ***Energy inefficiency of multiple-issue processors***
  - Multiple instruction issue incurs overhead in logic that grows faster than the issue rate increase
    - Dependence checking, register renaming, wakeup and select, etc.
    - Without voltage reduction, higher IPC will lead to lower performance/watt!
  - Growing gap between peak issue rate and sustained performance
    - The number of transistor switching is proportional to the peak issue rate
    - The performance is proportional to the sustained rate
    - Thus, growing gap translates to increasing energy per unit performance!
- ***For example, speculation is inherently inefficient***
  - It can never be perfect, thus there is inherently waste in executing computations before we know that whether the path is taken or not
- ***Increasing clock rate is also not energy efficient***
  - Increasing clock rate will increase transistor switching frequency
  - Faster clock rate need deeper pipeline, but it will increase the overhead of pipelining