CS 7810 Lecture 2

Complexity-Effective Superscalar Processors

S. Palacharla, N.P. Jouppi, J.E. Smith
U. Wisconsin, WRL
ISCA ’97

Complexity-Effective

- Conflict between clock speed and parallelism
- Goals of the paper:
  - Characterize complexity as a function of issue width, window size, and feature size
  - Propose clustered microarchitecture that allows fast clocks with high parallelism

Current Trends (circa 1997)

- More functional units, large in-flight windows
- Impact on cycle-time critical structures
  - Register renaming
  - Instruction wake-up
  - Instruction selection
  - Result bypass
  - Register files
  - Caches

Wire Delay Trends

- Logic delays scale linearly with feature size
- Wire delay ~ RC = R_m x C_m x L^2
- $R_m = \rho$ / (width x thickness)
- C_m = 2 x ε x ε_0 x (thickness/width + width/thickness)
- R_m ~ S ; C_m ~ S ; L ~ 1/S (gate size scaled by 1/S)
- Hence, delay across 50K gates is constant in ps and is linear with S in terms of FO4

Update on Wire Delays

- "The Future of Wires", Ho, Mai, Horowitz, 2001
- C_m actually decreases with reduced feature widths
- Hence, wire delay across 50K gates (in FO4) increases only slightly and is not quite linear with S – uses repeaters
- Wire delays are still a problem (though, not as bad as Palacharla et al. claim) – also note, FO4s/clock is shrinking

Update on Wire Delays

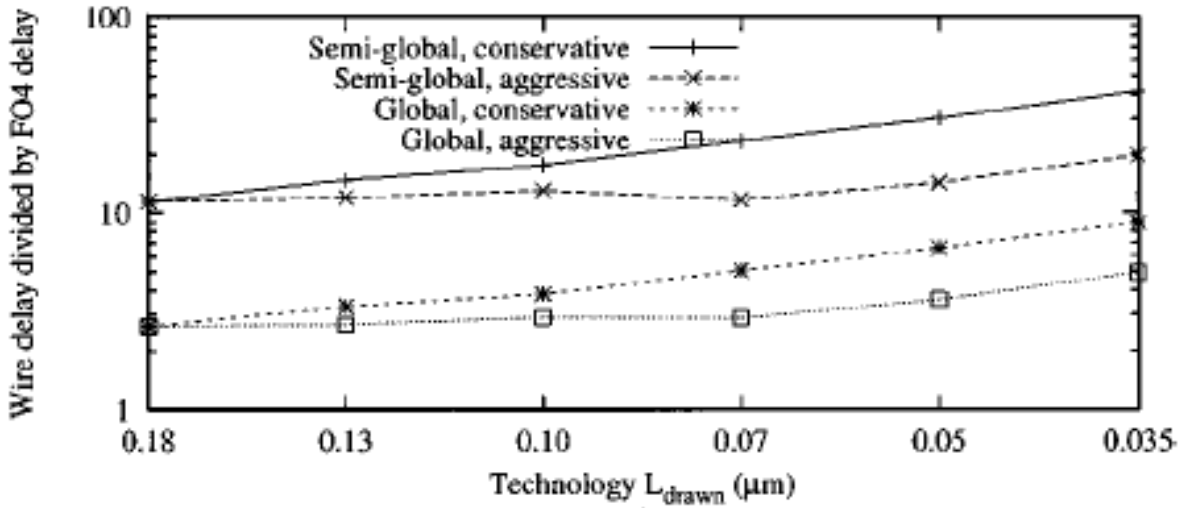
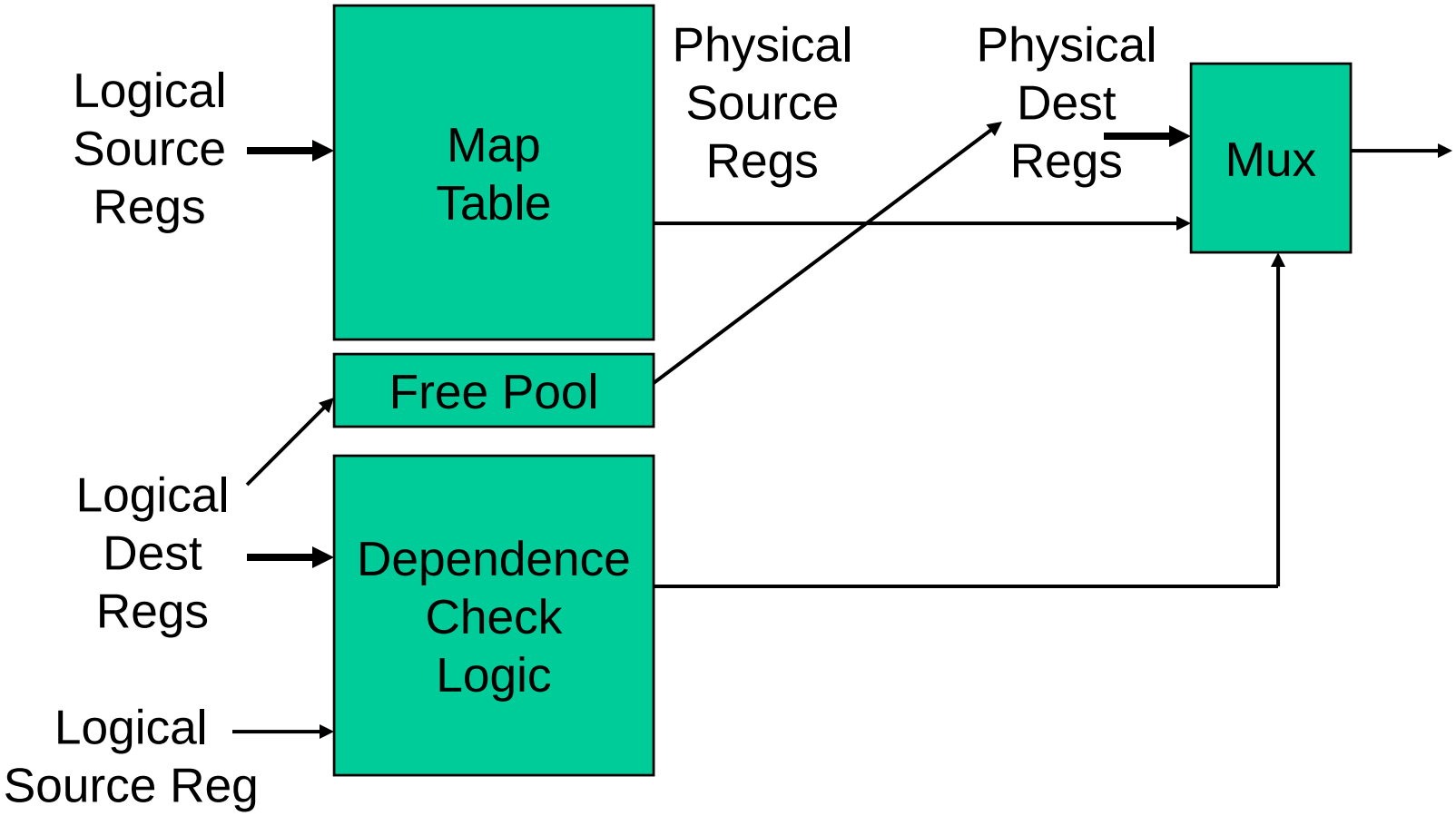


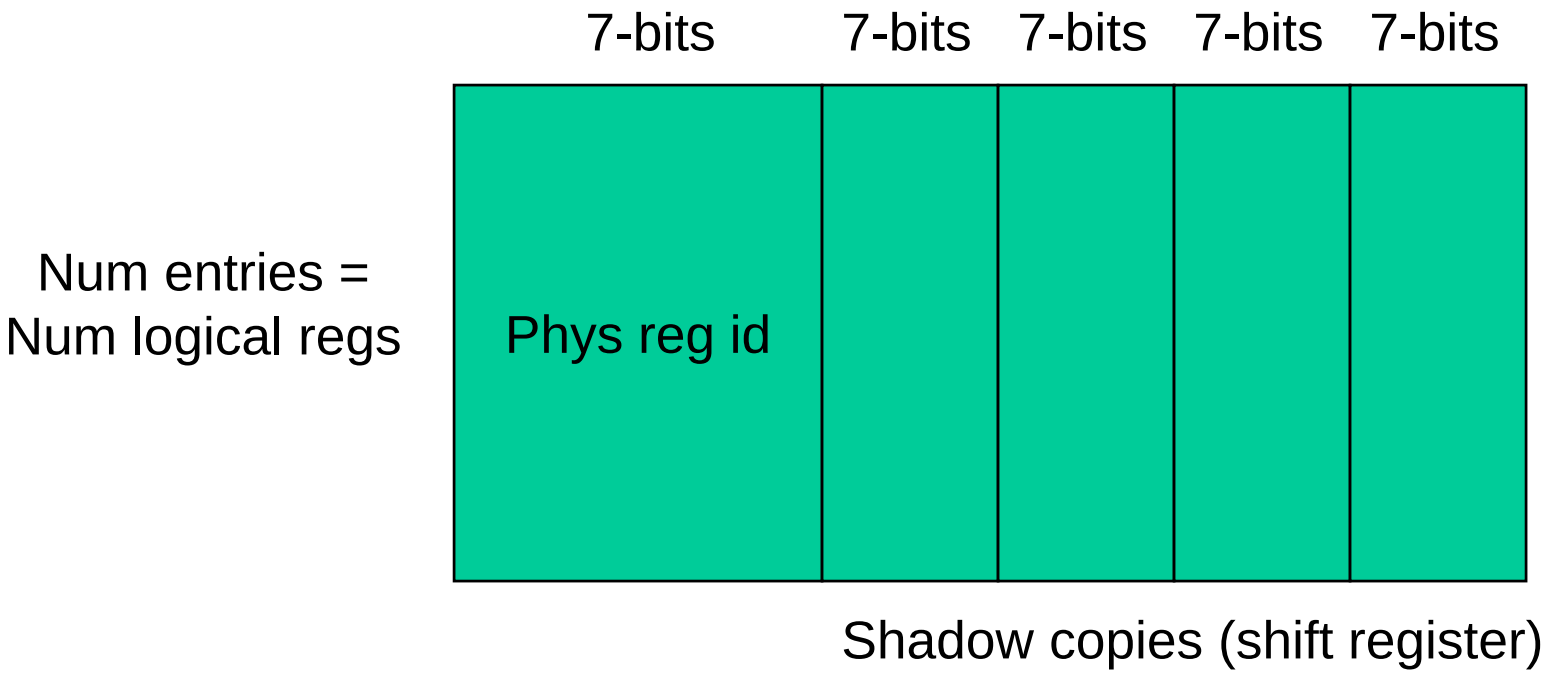
Fig. 17. Wire delays (in FO4s) for scaled-length wires spanning 50 K gates.

From “Future of Wires”, Ho, Mai, Horowitz

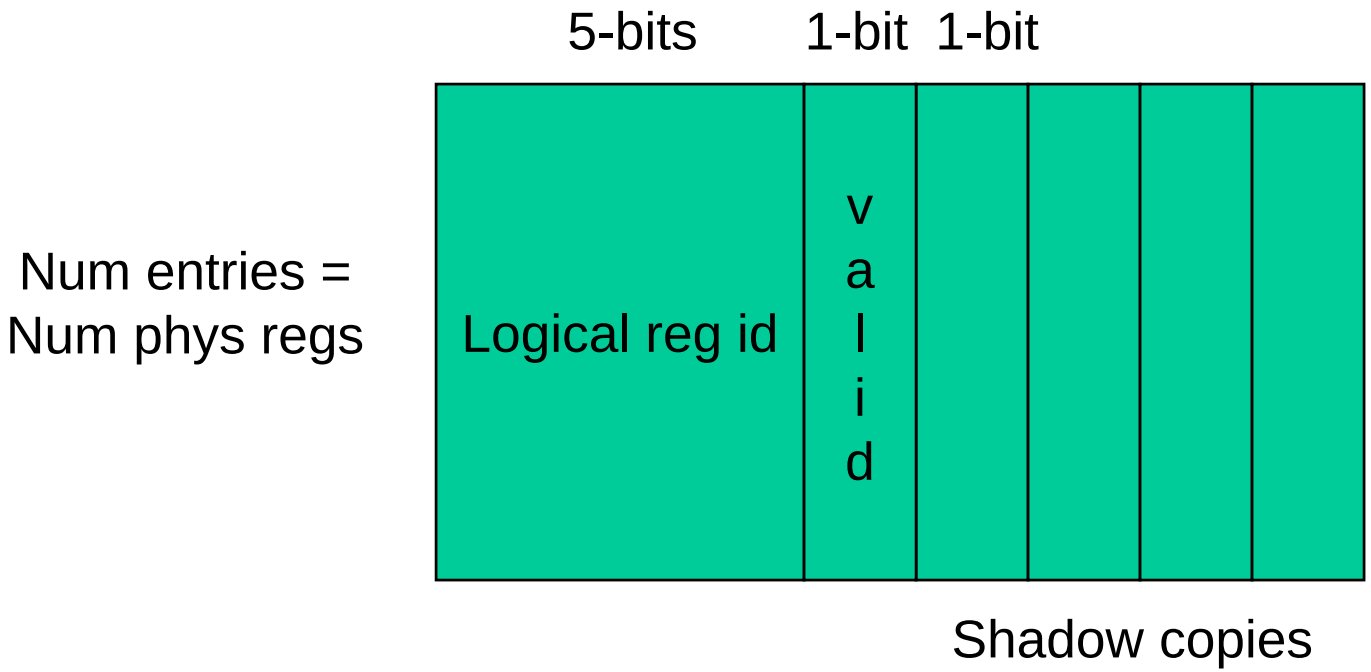
Register Rename Logic



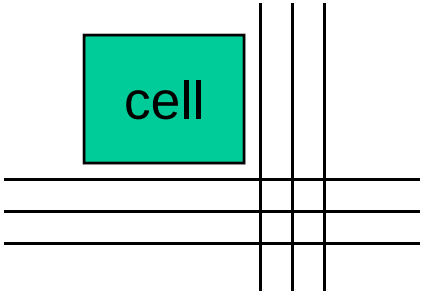
Map Table – RAM



Map Table – CAM



Delay Model



Wire length = C + 3 x IW

Delay = RC

$= c_0 + c_1 \times IW + c_2 \times IW^2$

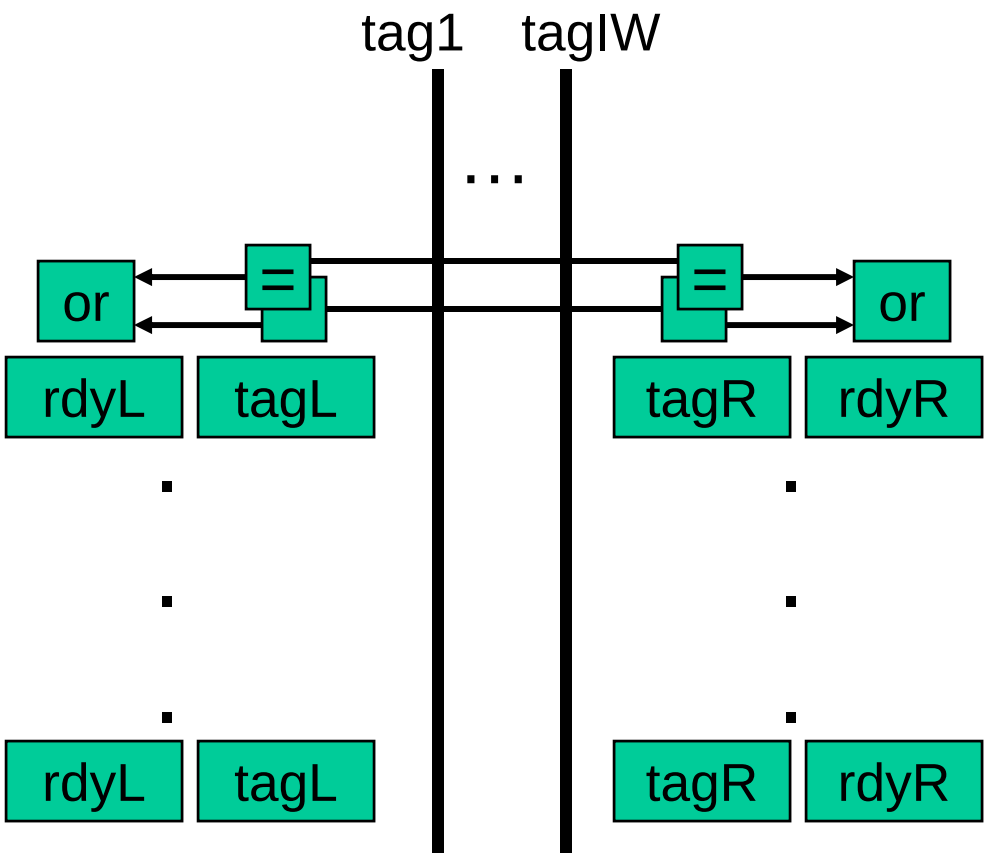
Rename delay ~ IW

The wire delay component increases as we shrink to 0.18μ

Problems:

- They assume that wire delay/λ (in ns) remains constant.
- No window size?

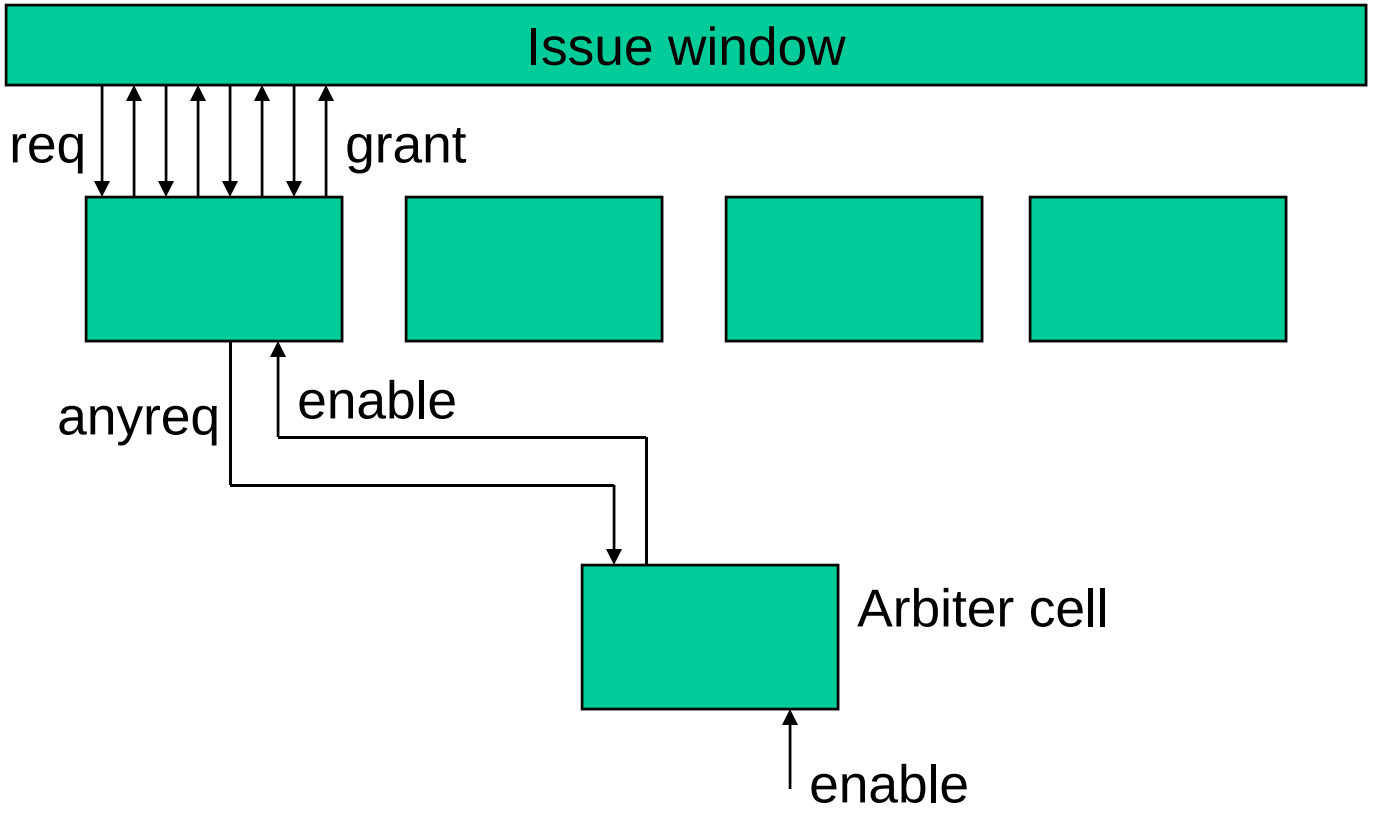
Wakeup Logic



Wakeup Logic

- CAM array wire length ~ issue width x winsize
- Capacitive load ~ winsize
- Matchline length ~ issue width
- Issue width has a greater impact on delay as it influences tagdrive and tagmatch (the quadratic components are not very dominant)
- For smaller features, the wire delays dominate

Selection Logic



Selection Logic

- Multiple FUs are handled by having more stages in series – further increases selection logic delay
- Delay ~ log(WINSIZE)
- Wire lengths ~ WINSIZE, but are ignored – hence, delay scales very well with feature size

Bypass Delay

- The number of bypass paths equals $2xIW^2xS$ (S is the number of pipeline stages)
- Wire length ~ IW, hence, delay ~ IW^2
- The layout and pipeline depth (capacitive load) also matter

Summary of Results

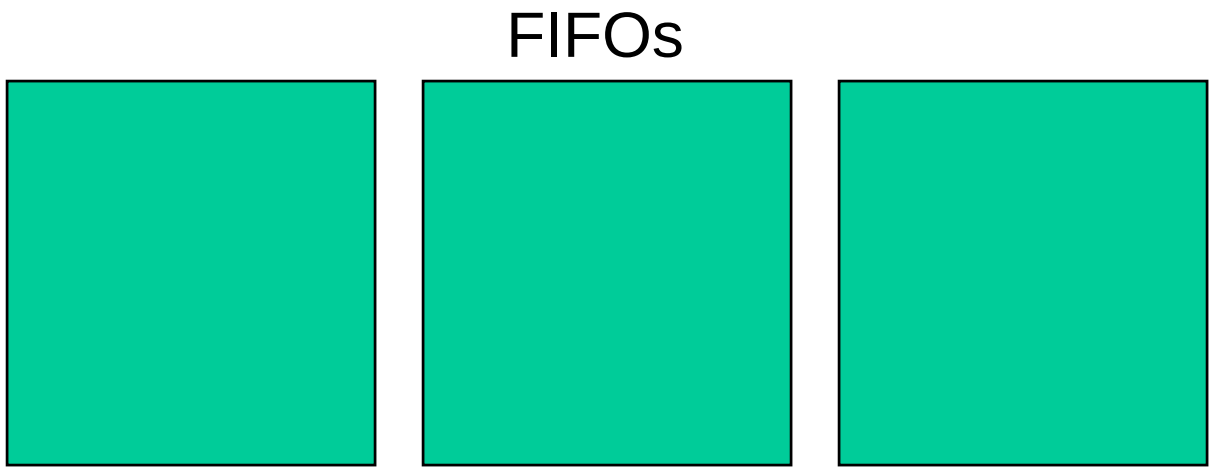
Issue Width	Window Size	Rename Delay (ps)	Wakeup + Select (ps)	Bypass Delay (ps)
0.8μm technology				
4	32	1577.9	2903.7	184.9
8	64	1710.5	3369.4	1056.4
0.35μm technology				
4	32	627.2	1248.4	184.9
8	64	726.6	1484.8	1056.4
0.18μm technology				
4	32	351.0	578.0	184.9
8	64	427.9	724.0	1056.4

Bottlenecks

- Wakeup+Select and Bypass have the longest delays and represent atomic operations
- Pipelining will prevent back-to-back operations
- Increased issue width / window size / wire delays exacerbate the problem (also for the register file and cache)

Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2

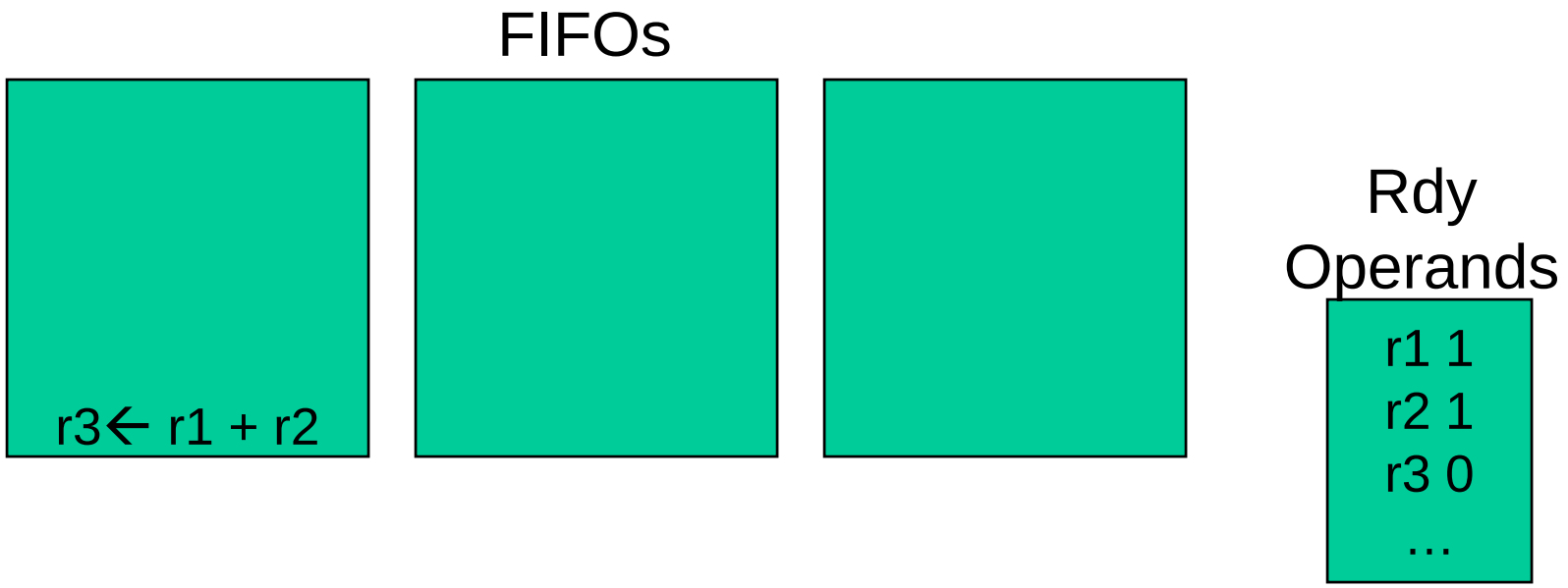


Rdy Operands

r1 1
r2 1
r3 0
…

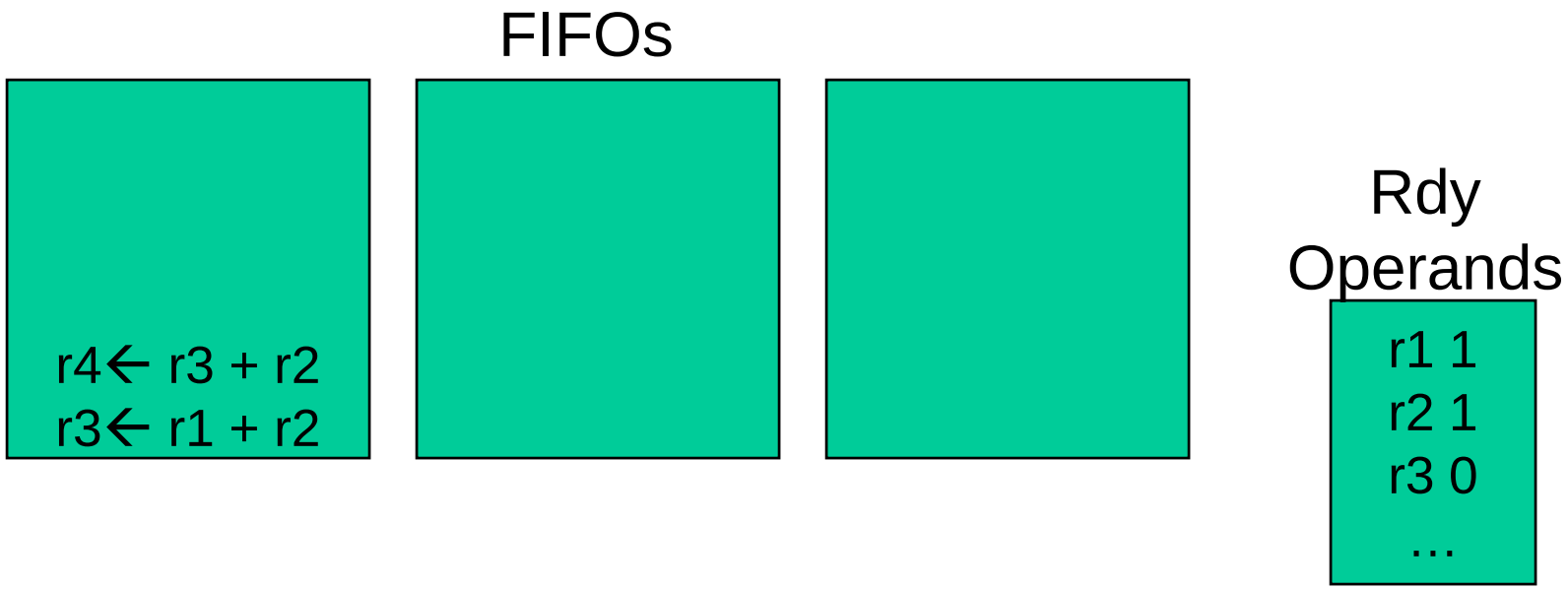
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



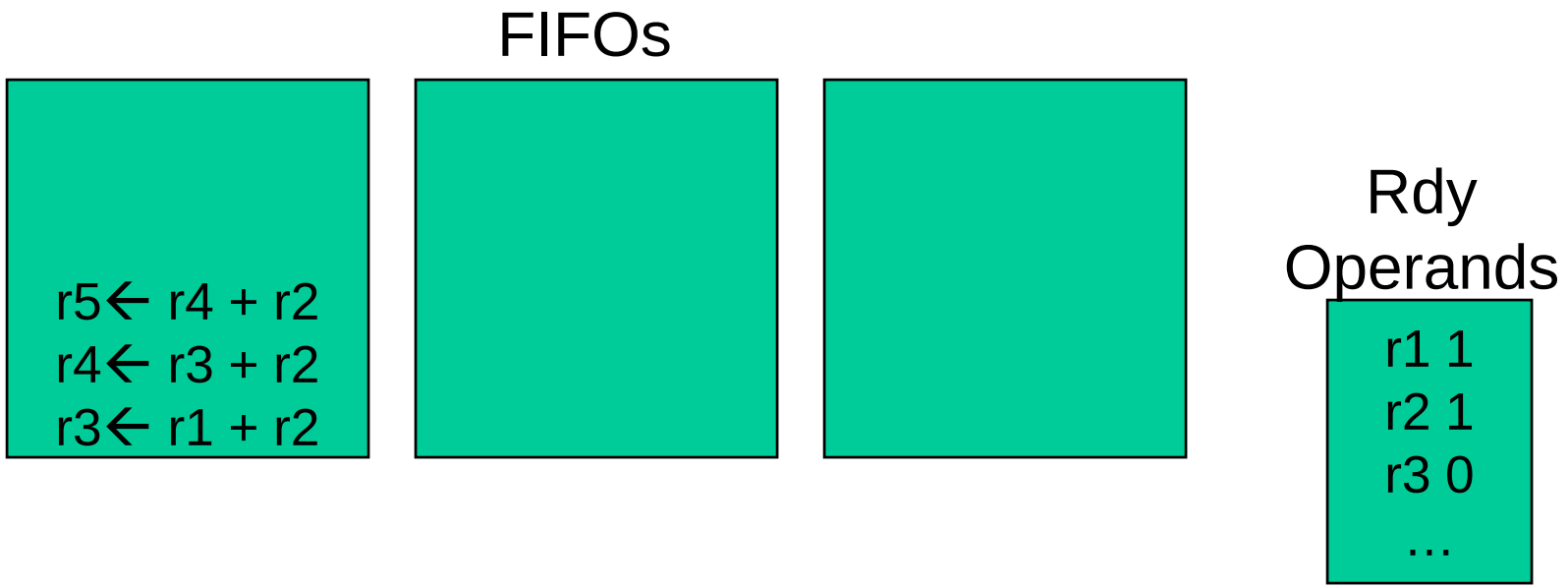
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



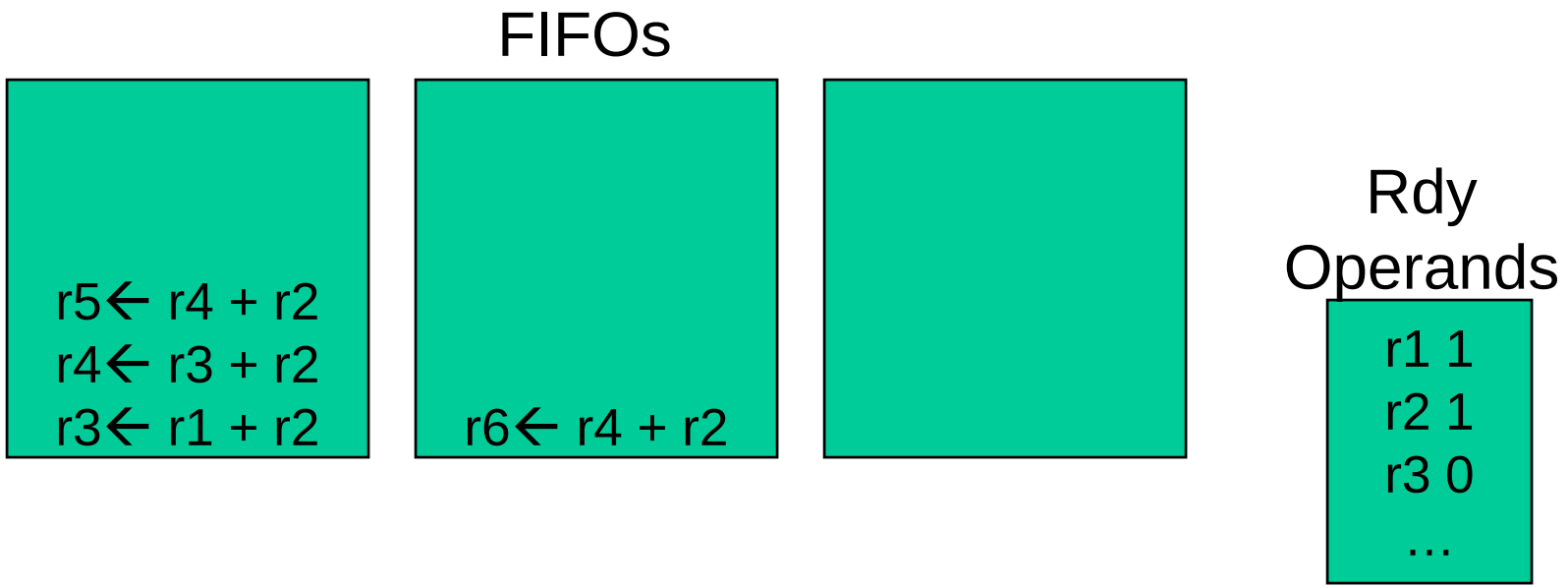
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



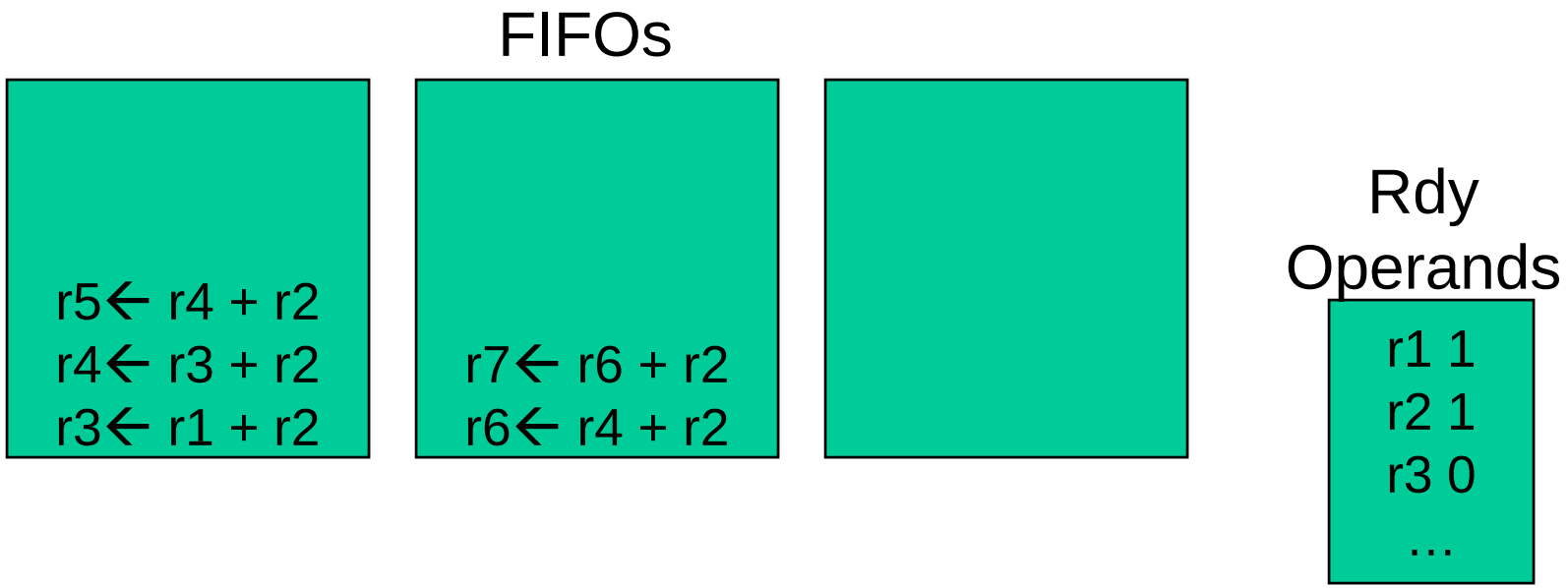
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



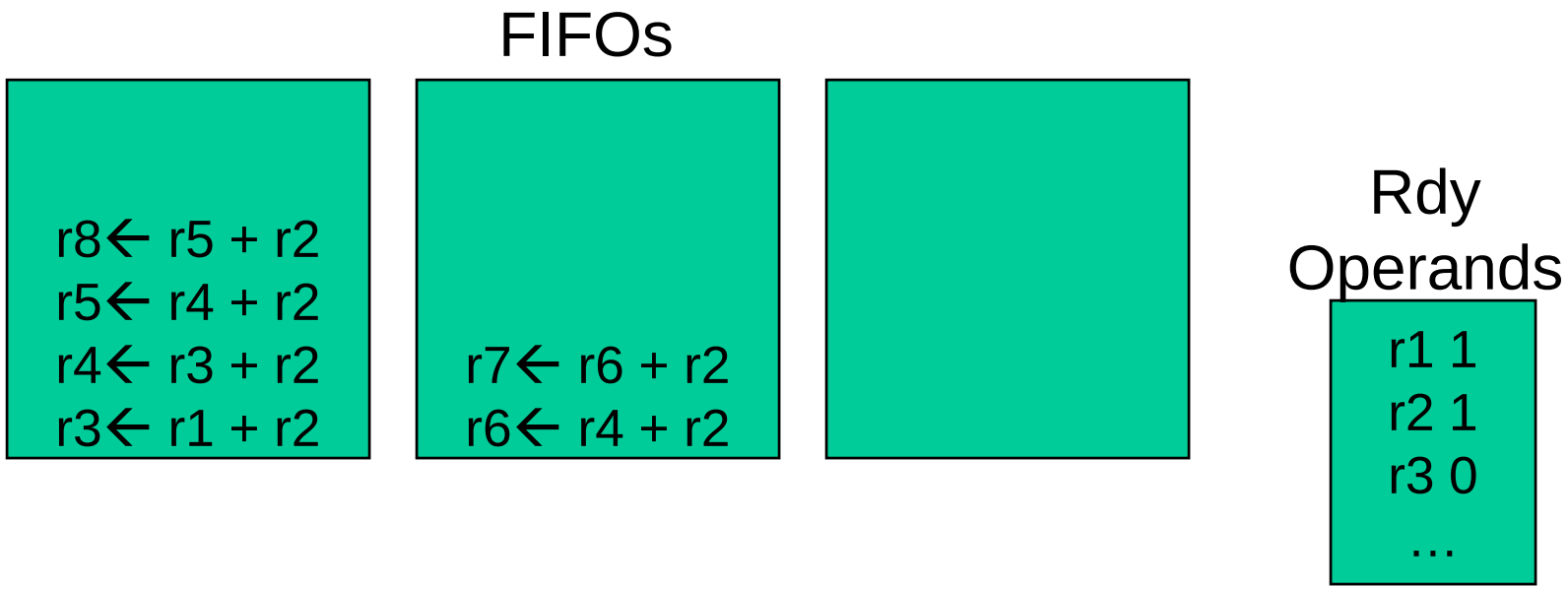
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



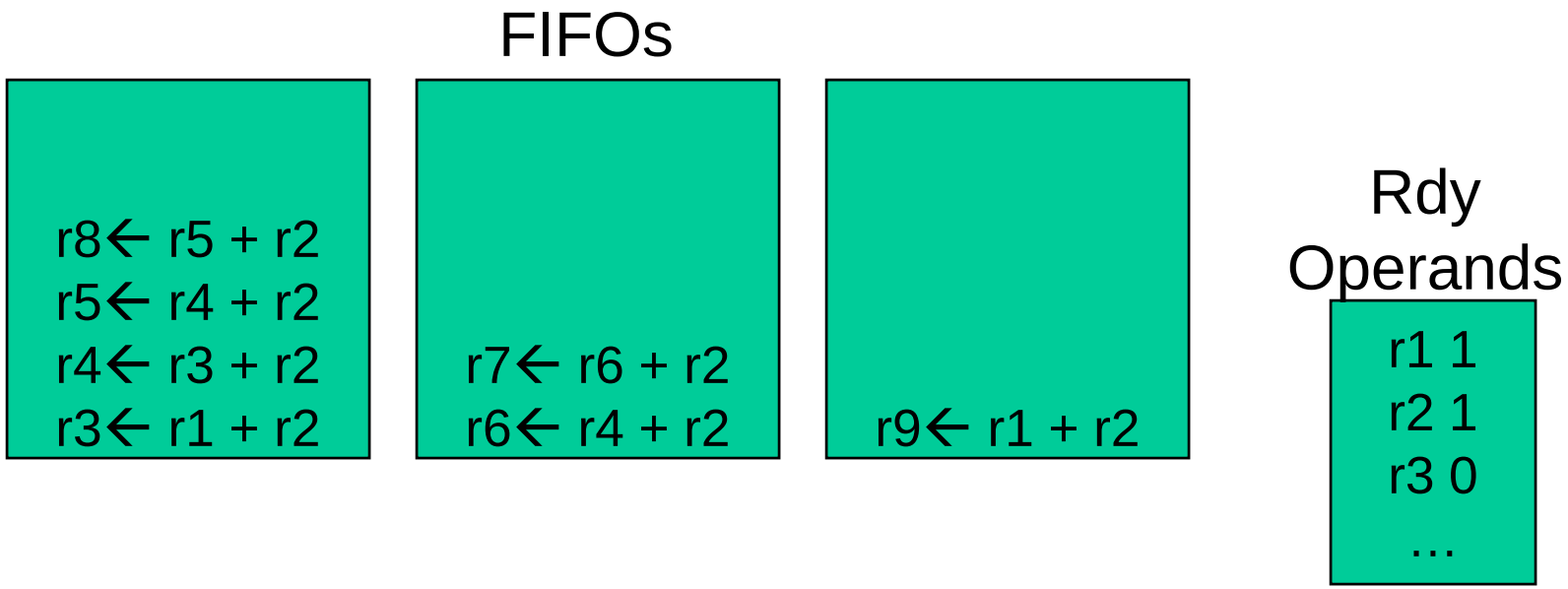
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



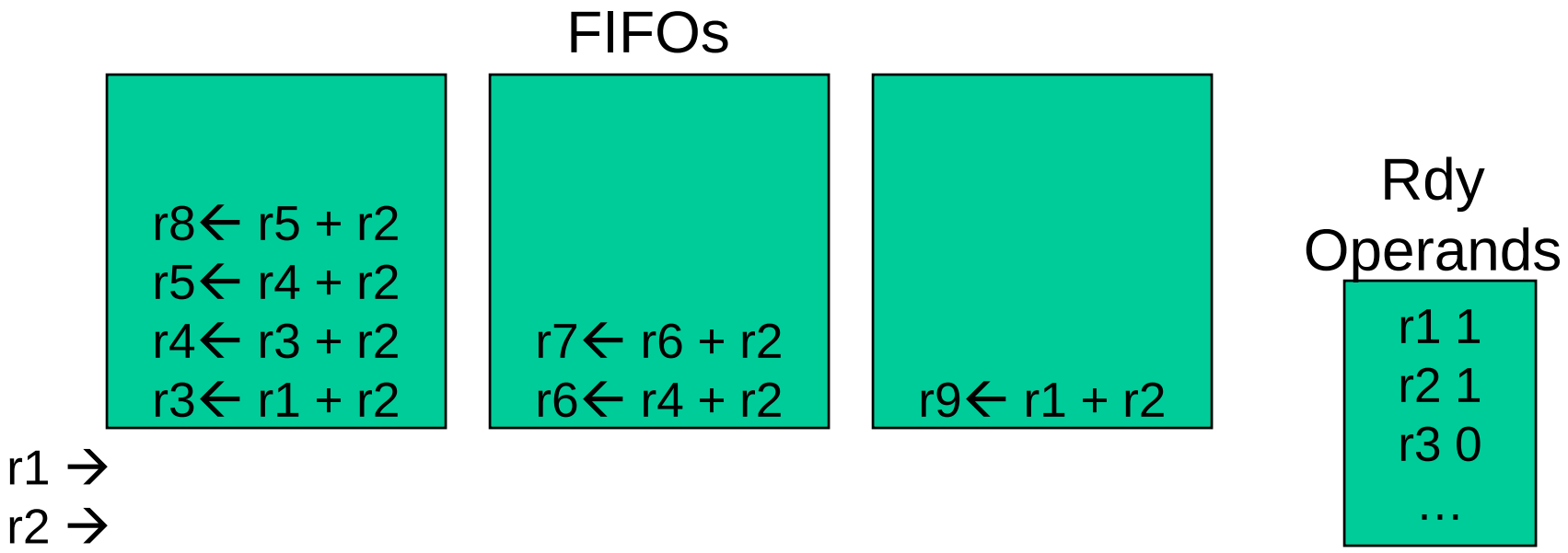
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



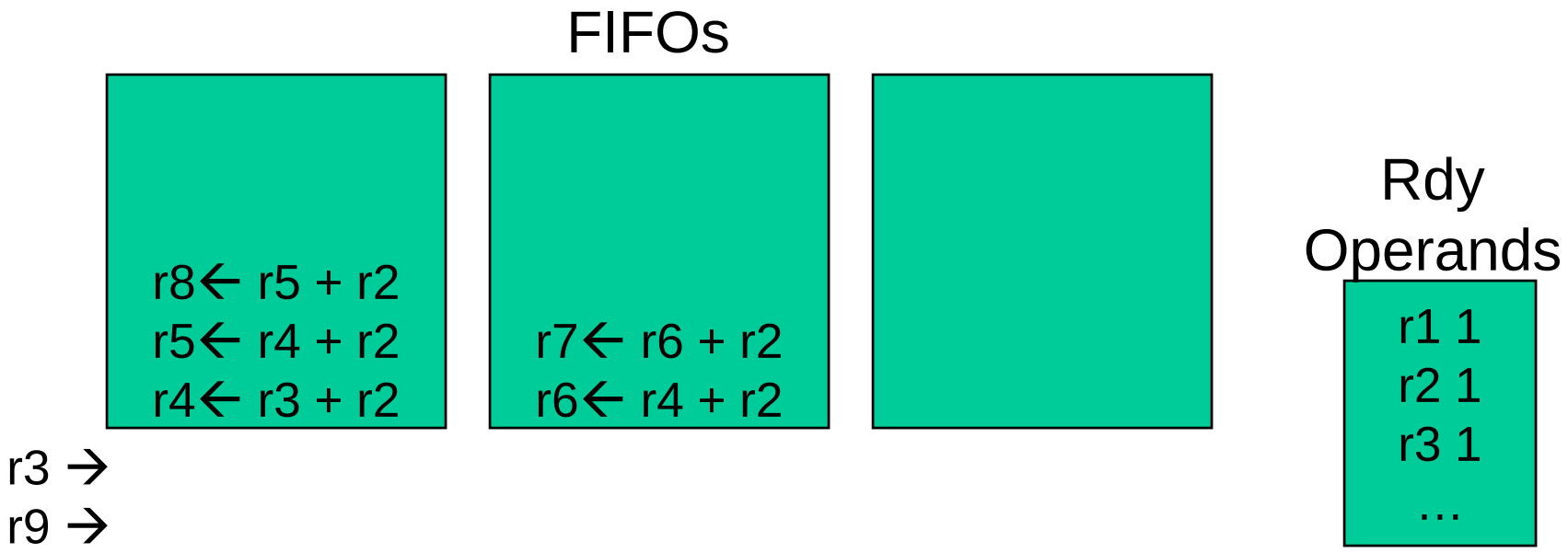
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



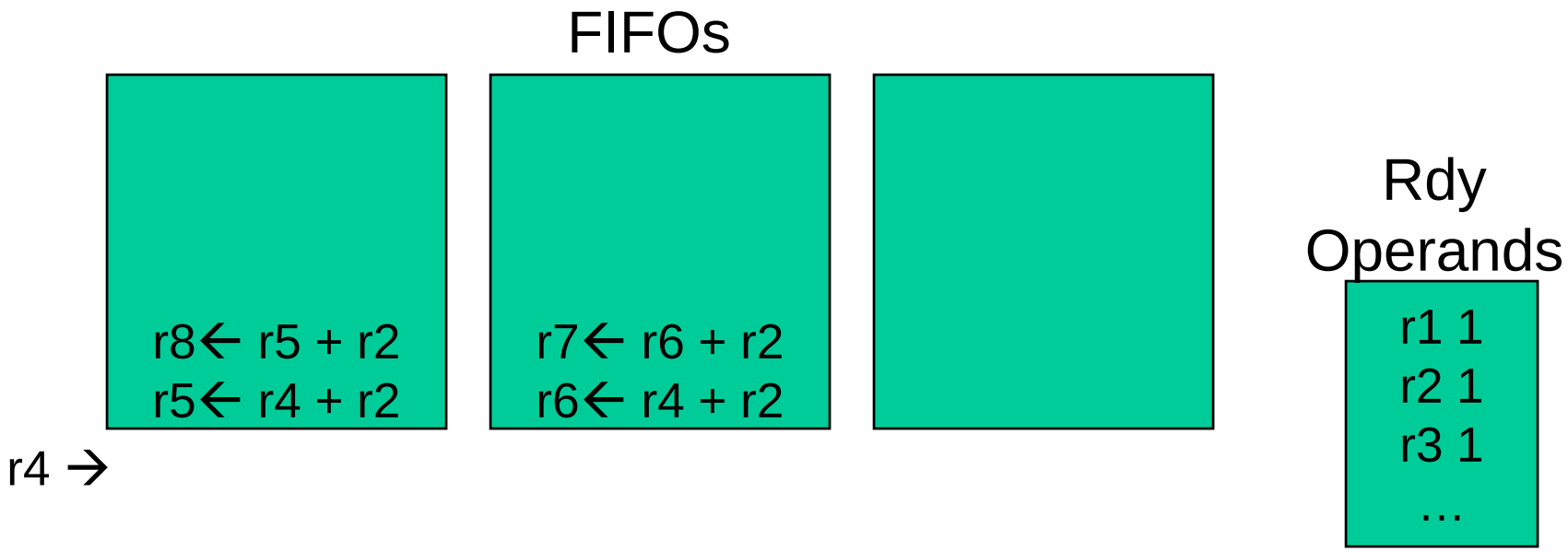
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



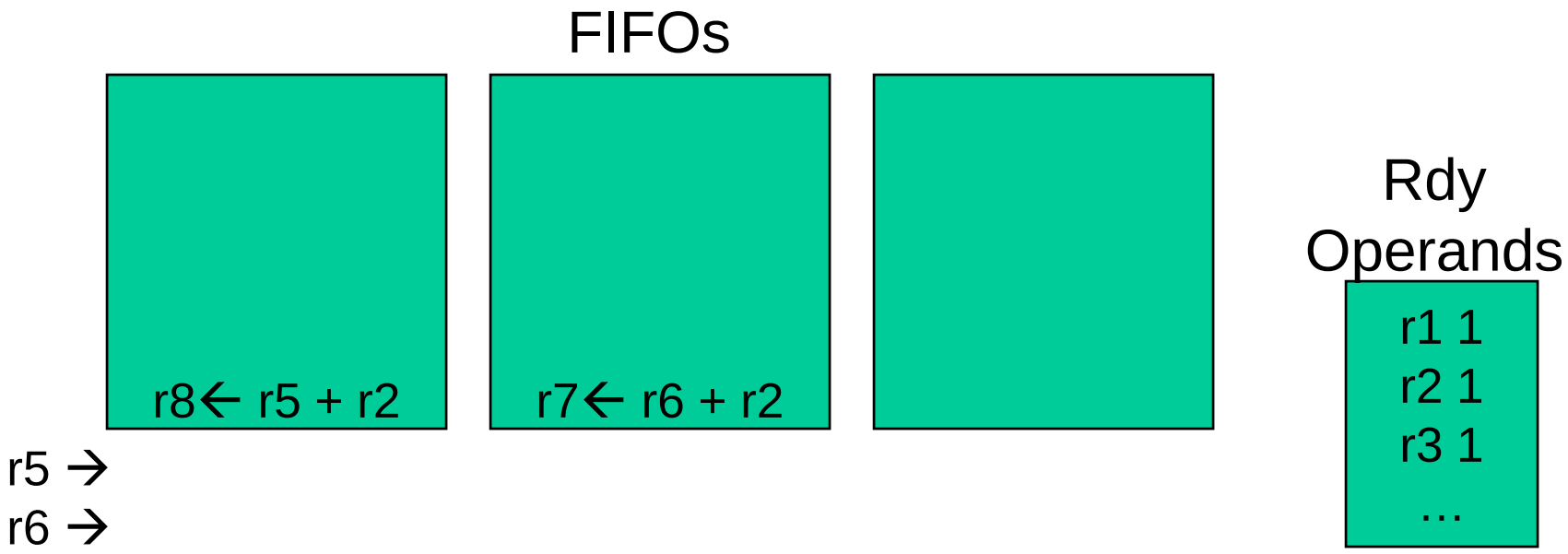
Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



Dependence-Based Microarchitecture

r3 ← r1 + r2
r4 ← r3 + r2
r5 ← r4 + r2
r6 ← r4 + r2
r7 ← r6 + r2
r8 ← r5 + r2
r9 ← r1 + r2



Pros and Cons

- Wakeup and select over a subset of issue queue entries (only FIFO heads)
- Under-utilization as FIFOs do not get filled (causes about 5% IPC loss) – but it is not hard to increase their sizes
- You still need an operand-rdy table

Clustered Microarchitectures

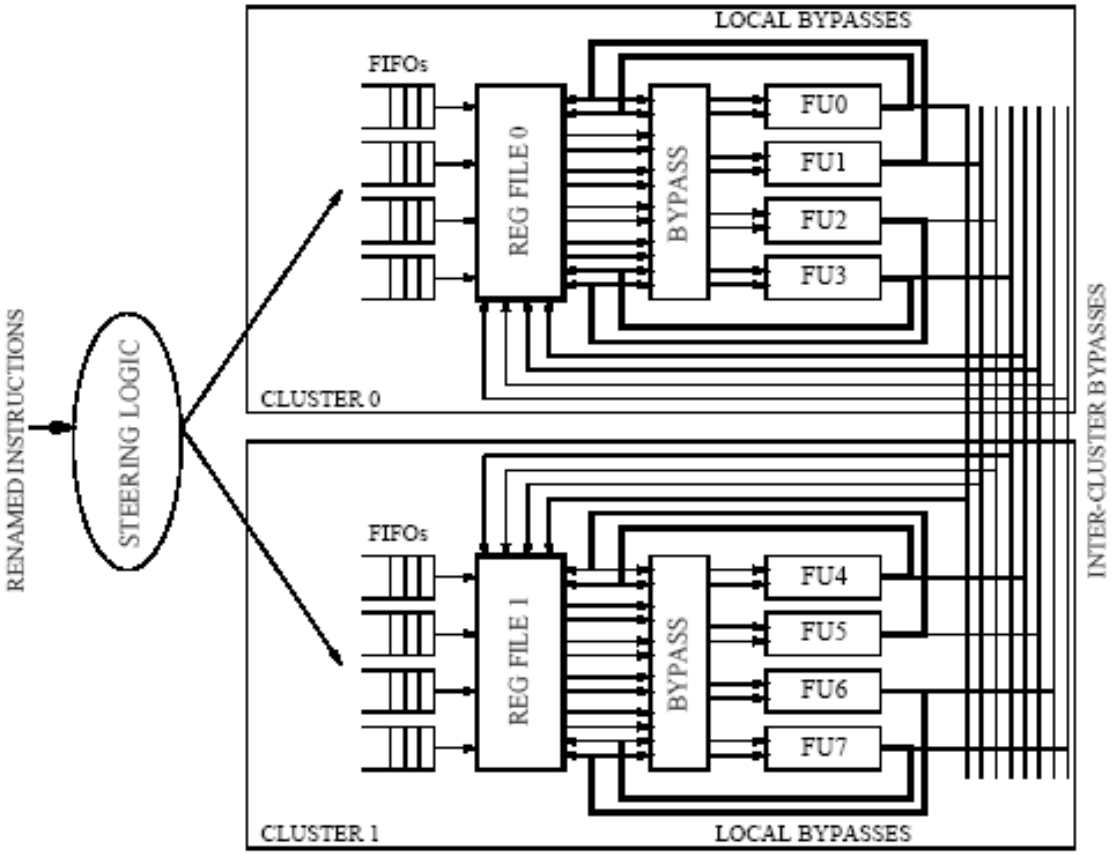


Figure 14: Clustering the dependence-based microarchitecture: 8-way machine organized as two 4-way clusters (2 X 4-way).

Clustered Microarchitectures

- Simplifies wakeup+select and bypassing
- Dependence-based, hence most communication is local
- Low porting requirements on register file, issue queue
- IPC loss of 6.3%, but a clock speed improvement

Conclusions

- As issue width and window size increase, the delays of most structures go up dramatically
- Dominant wire delays exacerbate the problem
- Hence, to support large widths, build smaller cores that communicate with each other
- With dependence information, it is possible to minimize communication costs

Next Class' Paper

- "Clock Rate vs. IPC: The End of the Road for Conventional Microarchitectures", ISCA'00

- Do not get bogged down in details & methodology