HOW DOES THE DATAPATH SCALE WITH THE SUPERSCALAR DEGREE?

B. Goossens

DALI
Université de Perpignan Via Domitia
webdali.univ-perp.fr

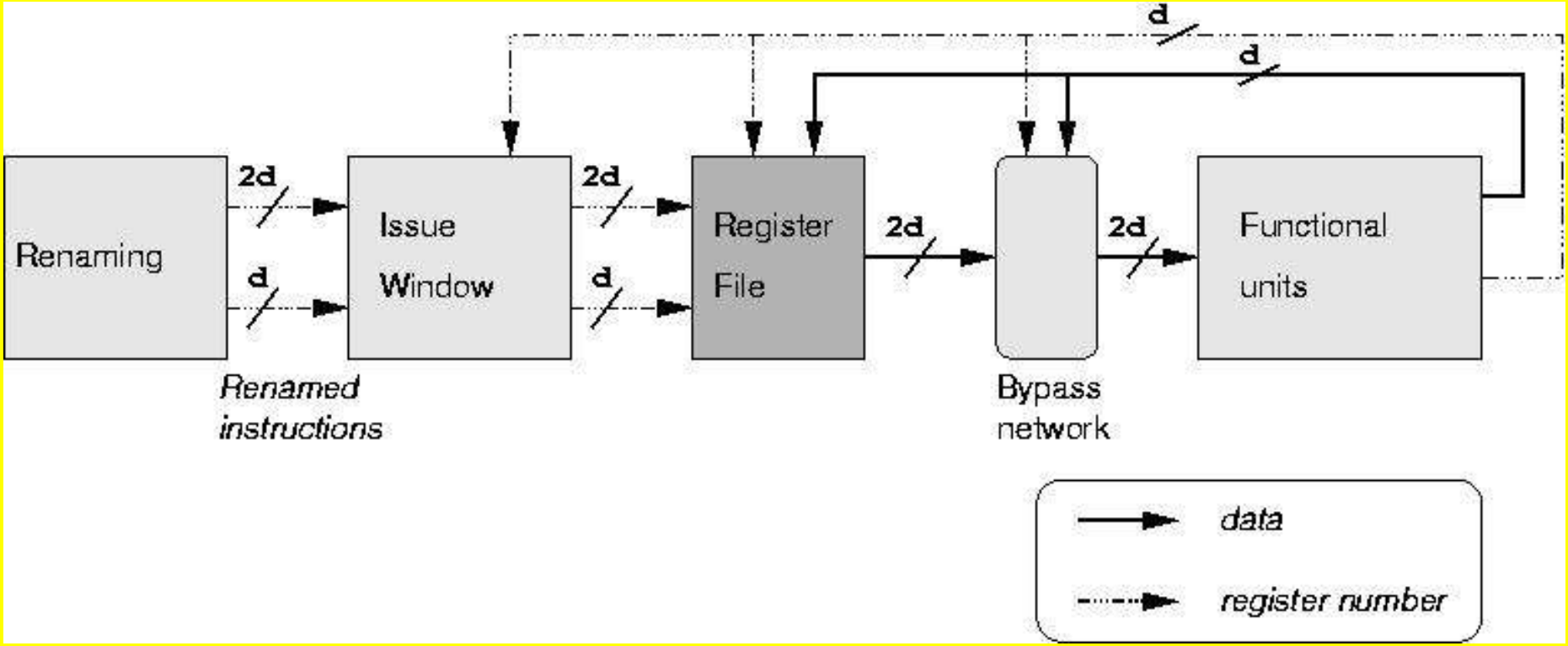
Russie, août 2006

UP VD

Outline

- Superscalar degree d datapath.
- Can we increase d?
- Experiment.
- How many ready sources at dispatch?
- How many last forwarded sources?
- How many first forwarded sources?
- Conclusion.

Superscalar degree d datapath



How does it scale with d?

Register file expansion

Pentium 4: d = 6, reg file 18 ports (12r, 6w)

Area u

Access time: 2 cycles

If d = 8, reg file 24 ports (16r, 8w)

Area 1.77 u ($24^2/18^2$)

Access time: 4 cycles

Bypass network expansion

Pentium 4: d = 6, 4 ALU, 2 AGU

16 bypass paths (no bypass from/to AGU)

Area u

Routing time: 1 cycle

If d = 8, 5 ALU, 3 AGU

25 bypass paths

Area 1.56 u ($5^2/4^2$)

Routing time: 2 cycles

Can we increase the superscalar degree d?

In actual microarchitectures, it is not possible without either increasing the cycle or lengthening the pipeline.

But what resources does the datapath really need?

How many reads, how many writes, how many bypasses are performed each cycle?

Does it increase with d^2, 3d, 2d, d?

The simplescalar PISA simulator

It simulates a 5 stages out-of-order pipeline. (fetch, dispatch, issue, writeback, commit)

We have arranged the pipeline to maximize the captured ILP. (perfect control flow prediction, multiple basic blocks fetch up to d IPC, 1 cycle icache latency, no latency penalty to read sources or write and forward results)

The simulated datapath

Superscalar degree d varying 4, 8, 16.

Data cache 16KB L1, 1 cycle, 256KB L2, 6 cycles.

Unbounded issue/retire width.

A set of d functional units for every operation.

256 entries load/store queue.
512 entries instruction window.

The Mibench benchmark suite

9 benchmarks selected. (rawcaudio, basicmath, qsort, crc, fft, gsmencode, gsmdecode, ispell, patricia)

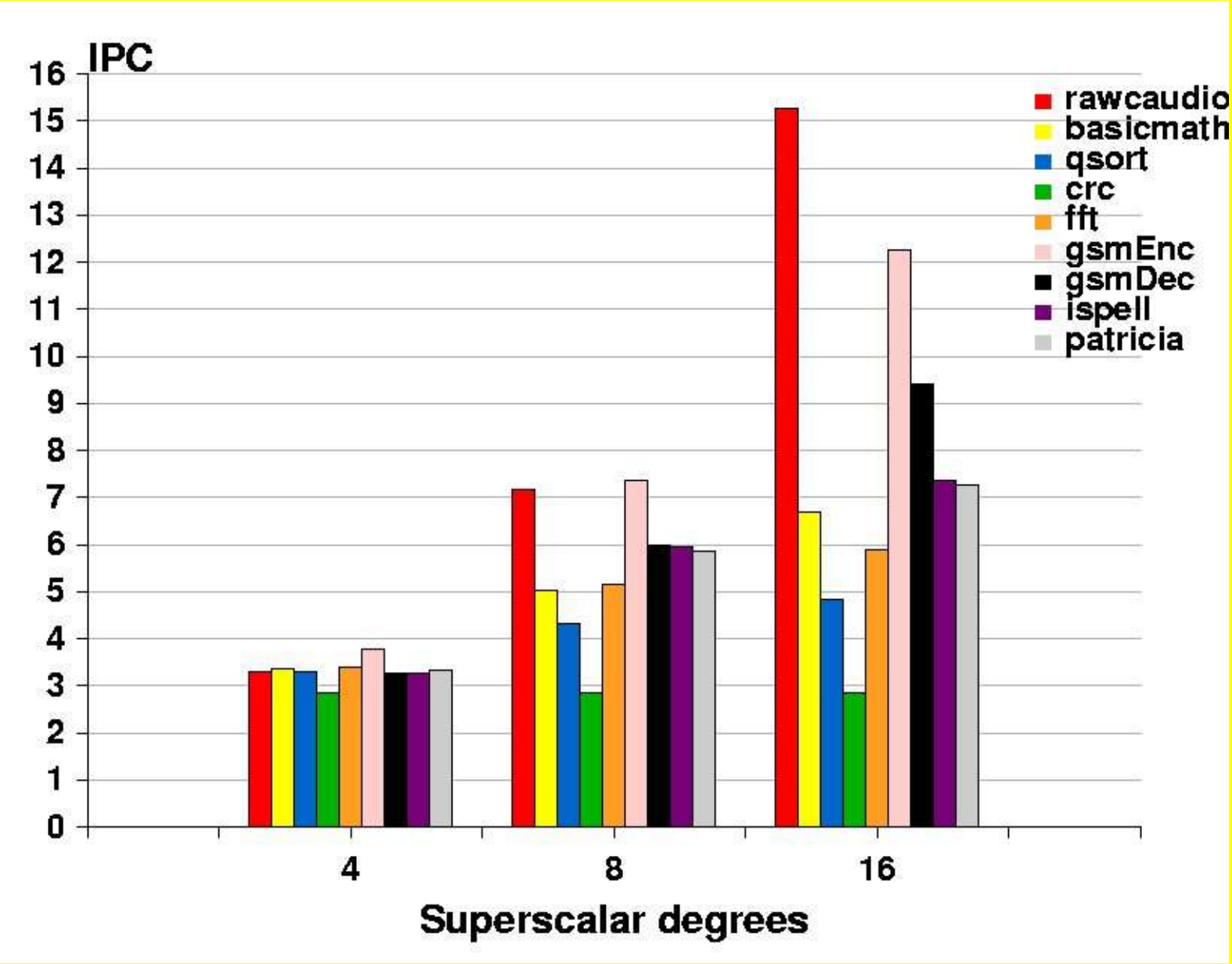
First 500M instructions run (no cache or predictor).

Low dcache miss rate.

Few long latency operations.

Benchmarks well suited to exhibit ILP.

Benchmarks IPC



1 ILP: supralinear

2, 3, 5 ILP: linear

6 – 9 ILP: infralinear

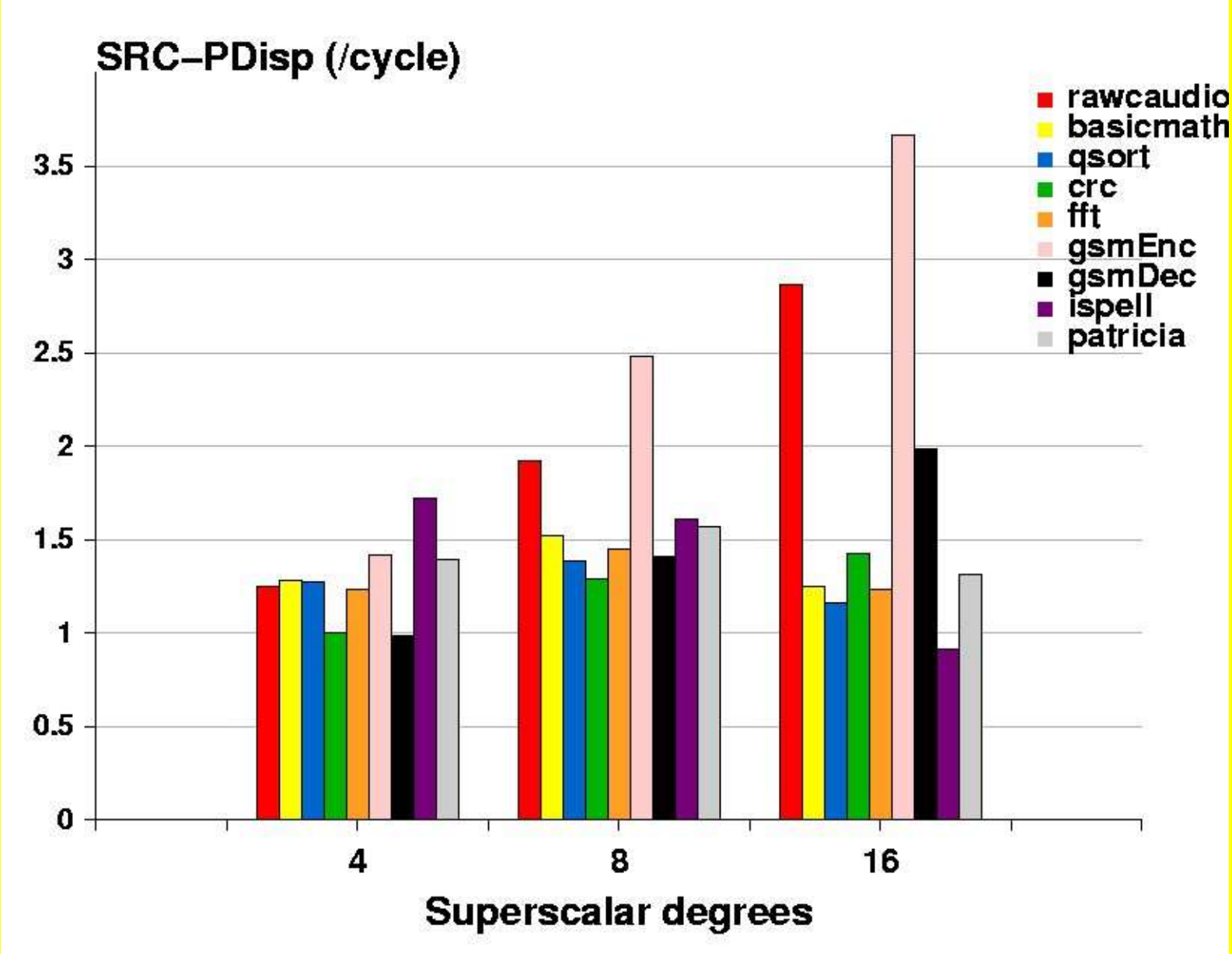
4 ILP: constant

IPC and ILP

For high ILP benchmarks, d: i and 2d: k*i, k>2
(increasing the degree helps the ILP capture)
(explained later)

For low ILP benchmarks, d: i and 2d: i
(no ILP available in CRC: checksum computation)
(change the source code to exhibit ILP)
(for CRC: binary tree computation)

Sources counters



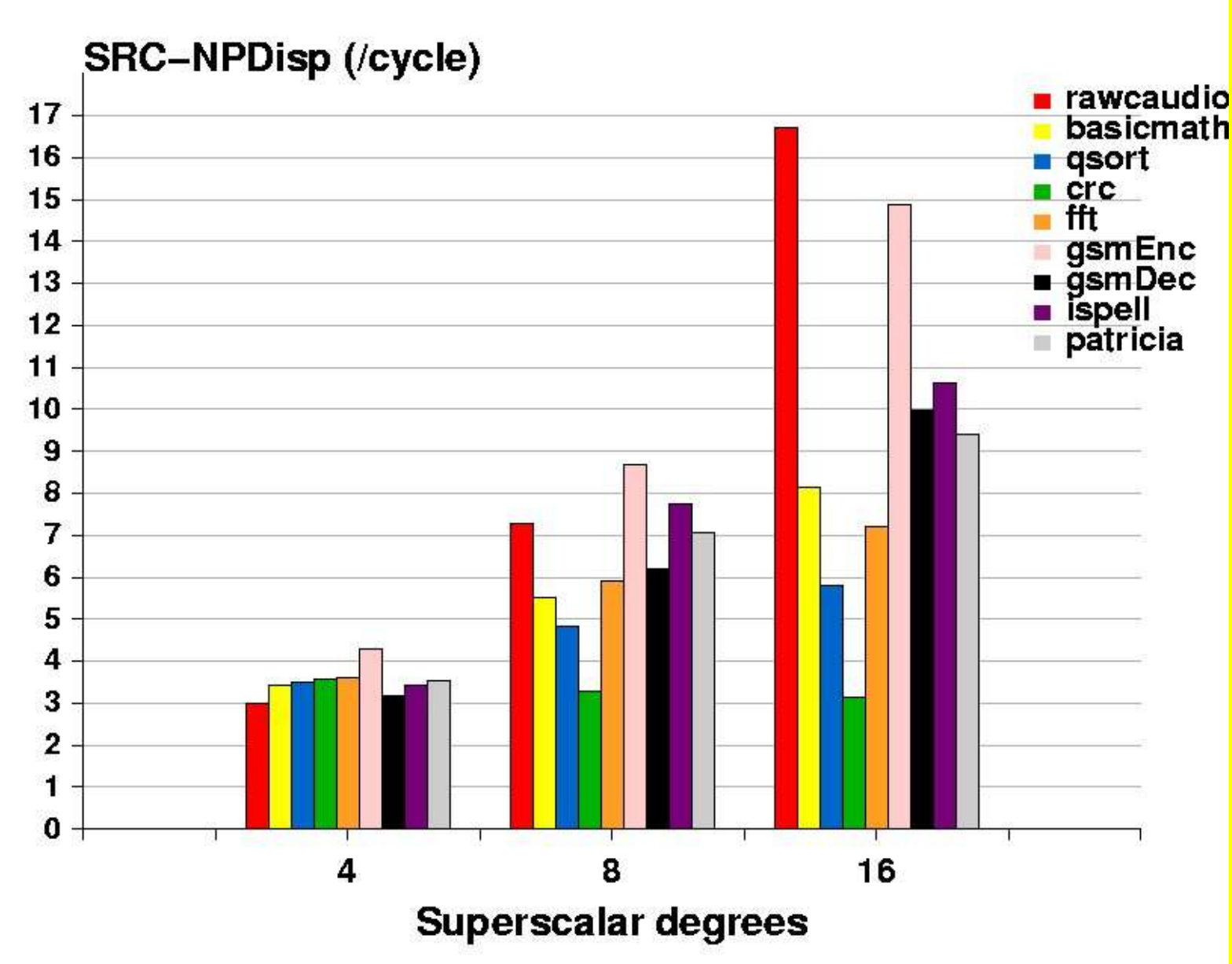
Ready sources after dispatch (readable from a front-end register file)

Ready sources after dispatch

There are less than 4 ready sources per cycle after dispatch. It is useless to have more than 4 read ports on a register file accessed during the dispatch stage.

The number of ready sources may decrease when d is increased (dependence length of more than 4 instructions and less than 8).

Sources counters



Non ready sources after dispatch (obtained from the forwarding network)

Non ready sources after dispatch

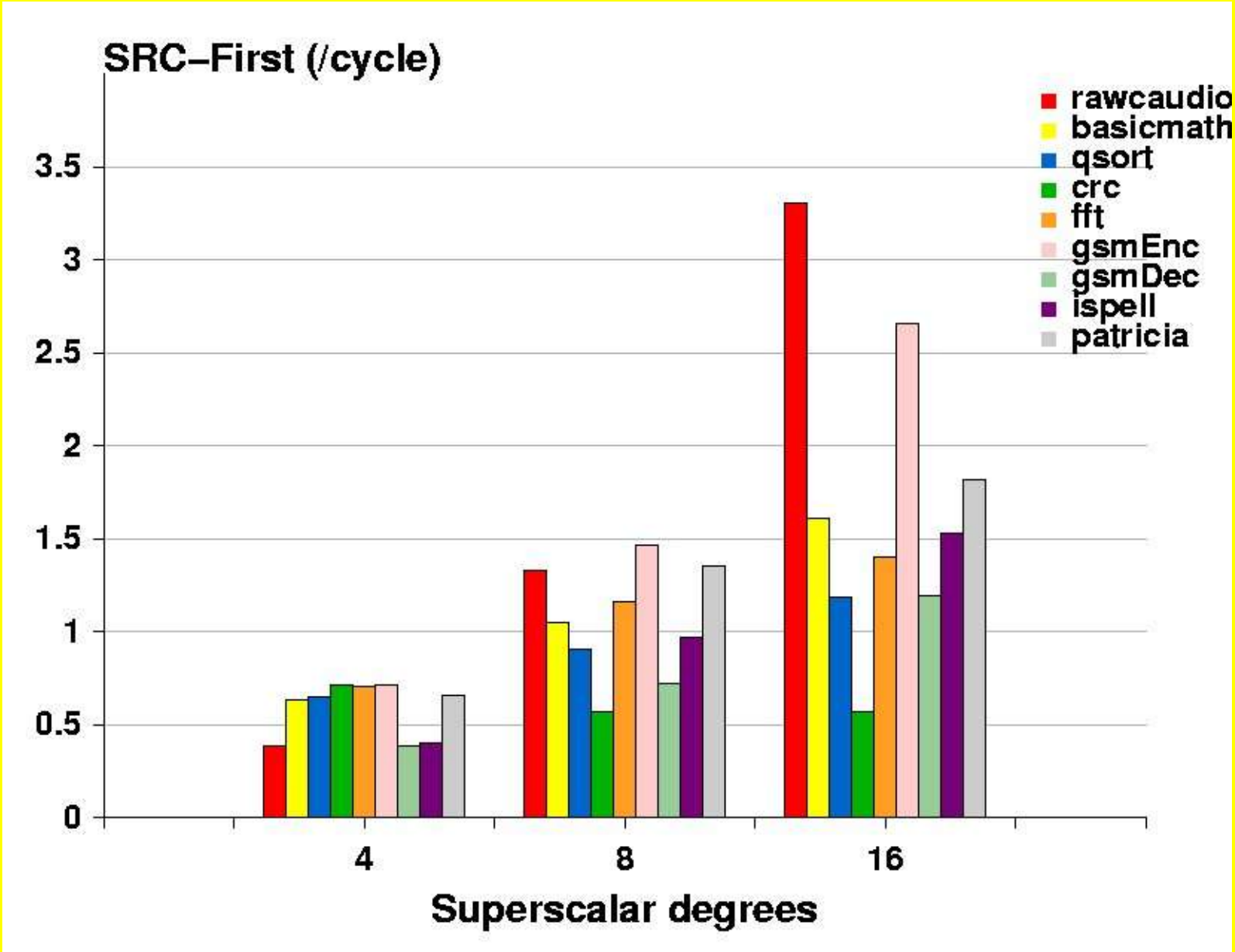
Up to 17 non ready sources per cycle for d=16.

The number of non ready sources is correlated to the ILP (compare to IPC diagram).

It is crucial for ILP capture to have a correctly sized forwarding network.

We must count separately the forwardings that launch their instructions and the ones that don't.

Sources counters



Forwarded first ready source

UP VD

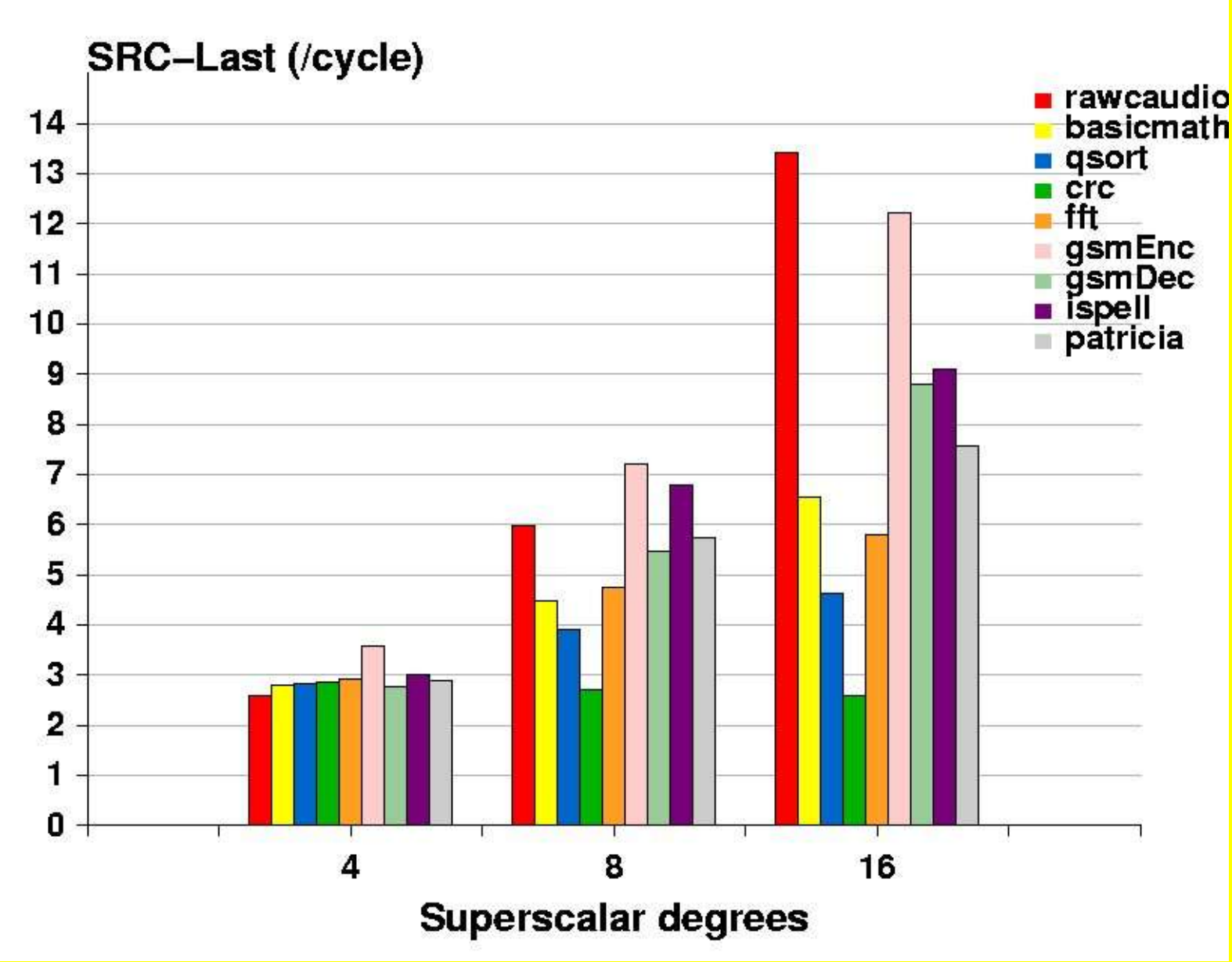
Forwarded first ready sources

Some of the forwarded sources concern instructions waiting for another not yet ready source.

These sources need not be bypassed to the issued instruction but can rather be forwarded via a less critical path.

The ready sources after dispatch and the first ready sources can both be read from the register file (background read request; 7 read ports are enough for d=16).

Sources counters



Forwarded last ready source

UP VD

Forwarded last ready sources

Some of the forwarded sources concern instructions waiting for no other source (single source or ready).

These sources are critical to the performance. They may be bypassed to the issued instruction (issuing is scheduled according to the last source latency).

Last ready sources are correlated to the ILP.

The diagram shows that when we increase d, we help to capture the remaining ILP.

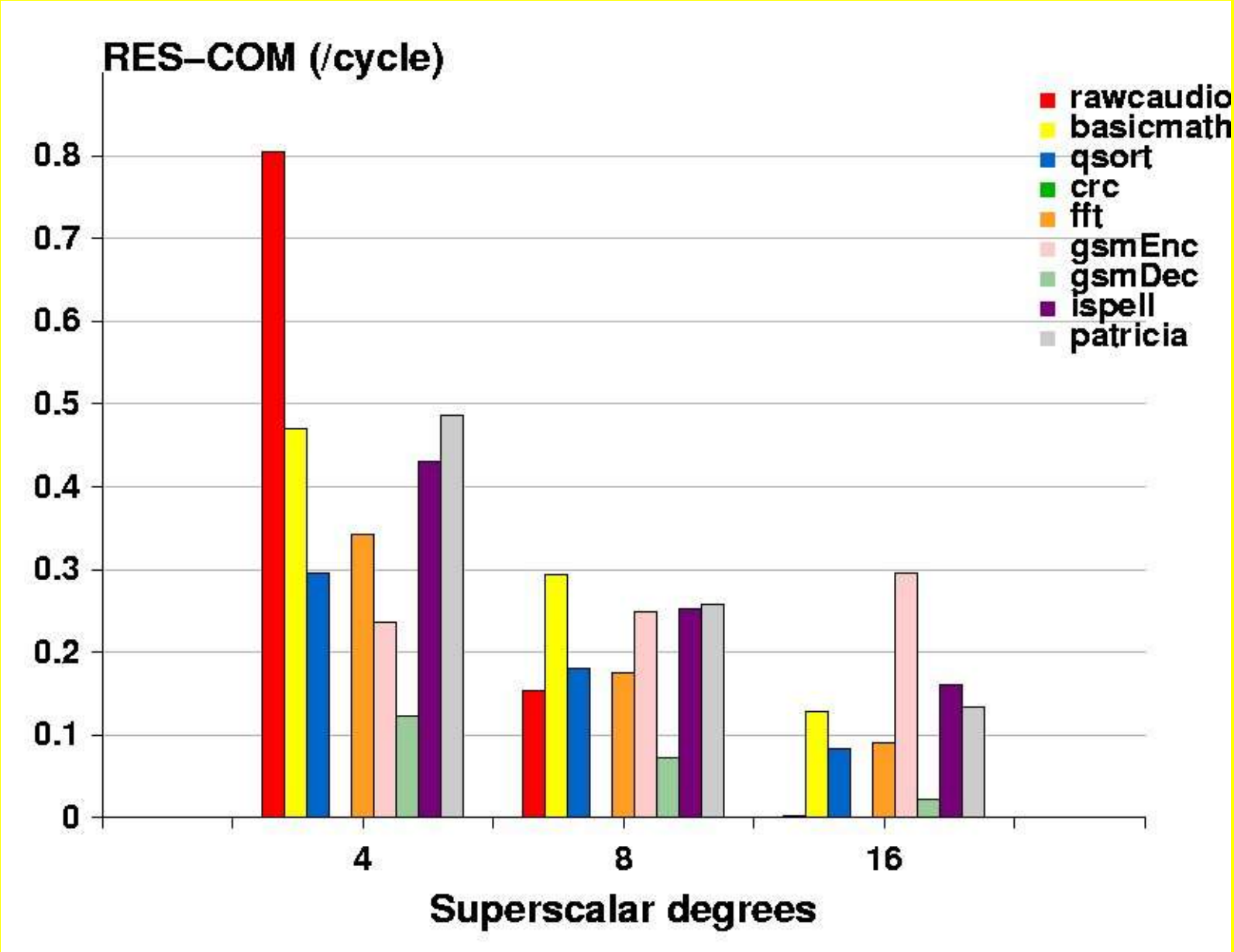
High d helps ILP capture

Assume that each instruction i lastly depends on i-8.

When d=4, no instruction is launched with a zero cycle delay. The low superscalar degree adds a one cycle delay from result to source (insts in WB match with insts in DISPATCH).

When d>=8, every instruction can be launched with a zero cycle delay from last result (insts in WB or COMMIT match with insts in ISSUE).

Results counters



Results committed to Ri renamed RRj and RAT[i]=j (Ri still renamed RRj at commit)

Results to be committed to register

Less than 1 result has to be committed per cycle.

It decreases when d is increased (less than 0.3 for d=16).

Committed results can be filtered among the committed instructions the same way destinations are renamed. Looking at 2d committed instructions removes quite all the results commitments.

Reducing the number of write ports

Experiments to measure IPC when write ports on the register file are bounded from d/2 to d.

For d=4, IPC is degraded as soon as 3 WP.
For d=8, IPC is degraded as soon as 6 WP.
For d=16, IPC is degraded as soon as 12 WP.

Write ports scale with d unless the write policy is changed.

Reducing the number of read ports

Experiments to measure IPC when write ports are bounded to the minimum fixed by the preceding experiment and read ports are varied from d to 2d.

For d=4, IPC is degraded as soon as 4 RP.
For d=8, IPC is degraded as soon as 9 RP.
For d=16, IPC is degraded as soon as 19 RP.

Read ports scale with d unless the read policy is changed.

Conclusion

Only 1 write port if filtered commitments.

Only d read ports for background read requests concerning ready or first forwarded sources.

Only d bypass links to directly send the results from functional units output to issued instructions.

Only d/4 forward links to send results to waiting instructions.

Conclusion

Datapath scales with d not d^2.

Scaling register file access ports with d without changing its place in the datapath degrades IPC. Scaling it with 2d does not degrade IPC.

What should the data path look like? Where should the register file be placed? How should the bypass and forwarding networks be organized?

Part of the answer in the next talk.