EECS 470

R10K scheme, T_{clk}

Lecture 9 – Winter 2024

Slides developed in part by Profs. Austin, Brehob, Falsafi, Hill, Hoe, Lipasti, Martin, Roth, Shen, Smith, Sohi, Tyson, Vijaykumar, and Wenisch of Carnegie Mellon University, Purdue University, University of Michigan, University of Pennsylvania, and University of Wisconsin.

Warning: Crazy times coming

- P3 is due on Sunday (2/9)
  - It's a lot of work (20 hours?)
- Proposal is due on Tuesday (2/13)
  - It's not a lot of work (1 hour?) to do the write-up, but you'll need to meet with your group and discuss things.
    - Don't worry too much about getting this right. You'll be allowed to change. Just a line in the sand. Will discuss on Friday (2/16).
- HW3 is due on Wednesday (2/14)
  - It's a fair bit of work (4 hours?)
  - Answers will be posted right after it's due, no late assignments!

Warning: Crazy times coming

- P3 is due on Sunday (2/9)
  - It's a lot of work (20 hours?)
- Proposal is due on Tuesday (2/13)
  - It's not a lot of work (1 hour?) to do the write-up, but you'll need to meet with your group and discuss things.
    - Don't worry too much about getting this right. You'll be allowed to change. Just a line in the sand. Will discuss on Friday (2/16).
- HW3 is due on Wednesday (2/14)
  - It's a fair bit of work (4 hours?)
  - Answers will be posted right after it's due, no late assignments!
- Midterm is on Thursday 2/15, 7-9pm
  - Open book, open notes. No computers, phones, etc.
  - Exam Q&A, Monday 2/12 7-8:30pm, Tuesday 6-8:30pm
    - Print exams and bring them.
  - Q&A in class Thursday 2/15
  - Best way to study is look at old exams (on website)
- 15 minute group meetings on Friday 2/16 (rather than inlab)

Let’s lose the ARF!

- Why?
  - Currently have two structures that may hold values (ROB and ARF)
  - Need to write back to the ARF after every instruction!
- Other motivations?
  - ROB currently holds result (which needs to be accessible to all) as well as other data (PC, etc.) which does not.
    - So probably two separate structures anyways
  - Many ROB entry *result fields* are unused (stores, branches)

Physical Register file Version 1.0

- Keep a “Physical register file”
  - If you want to get the ARF back you need to use the RAT.
- But the RAT has speculative information in it!
  - We need to be able to undo the speculative work!
    - How?

How?

- Remove
  - The value field of the ROB
  - The whole ARF
- Add
  - A “Retirement RAT[1]” (RRAT) that is updated by retiring instructions the same way the RAT is by issuing instructions
- Actions:
  - When you finish execution, update the PRF as well as the ROB (ROB just gets “done” message now)
  - When you retire, update the RRAT
  - (Other stuff we need to think about goes here.)

[1] http://www.ecs.umass.edu/ece/koren/ece568/papers/Pentium4.pdf

Example

RAT

AR	PR
0	1
1	2
2	3
3	4
4	10

Dispatch:

Assembly
R1=R2*R3
R3=R1+R3

RRAT

AR	PR
0	1
1	2
2	3
3	4
4	10

Example

RAT

AR	PR
0	1
1	0
2	3
3	5
4	10

In-flight

Assembly
R1=R2*R3
R3=R1+R3

Renamed
P0=P3*P4
P5=P0+P4

RRAT

AR	PR
0	1
1	2
2	3
3	4
4	10

Resolving Branches

- RRAT
  - On mispredict at head of queue copy retirement RAT into RAT.
- Early resolution? (briefly)
  - BRAT
    - Keep a RAT copy for each branch *in a RS!*
      - If mispredict can recover RAT quickly.
      - ROB easy to fix, RS's a bit harder.

Freedom

- Freeing the PRF
  - How long must we keep each PRF entry?
    - Until we are *sure* no one else will read it before the corresponding AR is again written.
    - Once the instruction overwriting the Arch. Register commits we are certain safe.
  - So free the PR when the instruction which overwrites it commits.
    - In other words: when an instruction commits, Free the thing overwritten in the RRAT.
- We could do better
  - Freeing earlier would reduce the number of PRs needed.
  - But unclear how to do given speculation and everything else.

Sidebar

- One thing that must happen with the PRF as well as the RS is that a “free list” must exist letting the processor know which resources are available.
  - Maintaining these free lists can be a pain!
    - Let’s talk a bit about how one would do this.

R10K scheme

- What are we doing?
  - Removing the ARF
  - Removing the value field of the RoB.
  - Adding a Physical Register File (~sum ARF and RoB)
  - Adding a Retirement RAT (RRAT)

RAT

AR	Target
0	4
1	2
2	7
3	1

A: R1=MEM[R2+0]
B: R2=R1/R3
C: R3=R2+R0
D: Branch (R1==0)
E: R3=R1+R3
F: R3=R3+R0
G: R3=R3+19
H: R1=R7+R6

RRAT

AR	Target
0	
1	
2	
3	

RoB

PRF

0	1	2	3	4	5	6	7	8	9
3	2	44	55	3	66	7	11	8	20

Alternative option (v0.9?)

- Use “back pointers” instead of RRAT.
  - Record, in the ROB, which value in the RAT you overwrote.
    - On commit, free that value (it will be the same as the one you would have overwritten in the RAT!)
    - On mispredict, “undo” each step in reverse order (from tail to head).
  - This gives same functionality as RRAT.
    - Slower to handle mispredict ***that is at the head of the RoB.***
      - But could in theory handle mispredict as

Optimizing CPU Performance

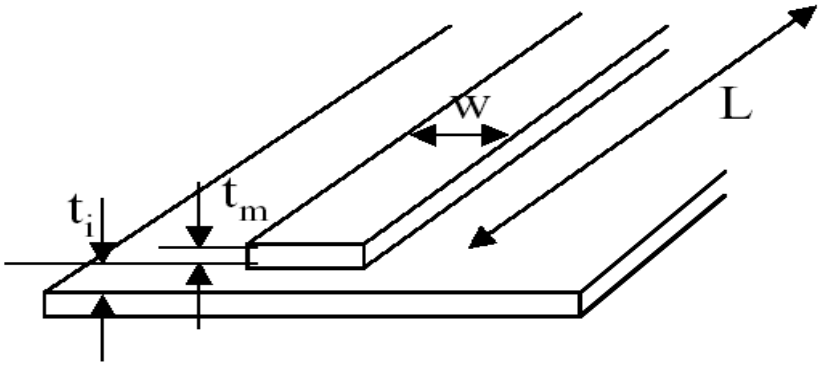
- Golden Rule: $t_{CPU} = N_{inst} * CPI * t_{CLK}$
- Given this, what are our options
  - Reduce the number of instructions executed
  - Reduce the cycles to execute an instruction
  - Reduce the clock period
- Our first focus: Reducing CPI
  - Approach: *Instruction Level Parallelism* (ILP)


Tclock

t_{CLK}

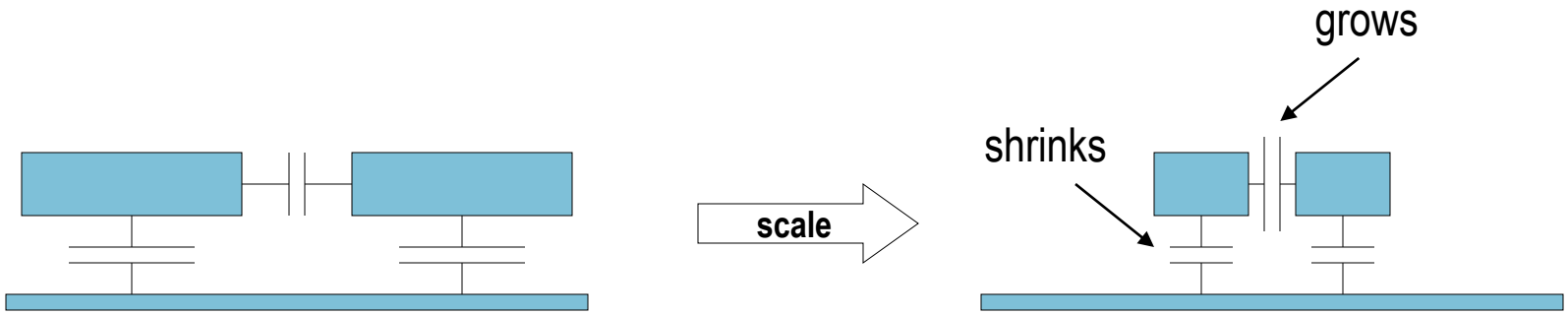
- Recall: $t_{CPU} = N_{inst}*CPI*t_{CLK}$
- What defines t_{CLK}?
  - Critical path latency (= logic + wire latency)
  - Latch latency
  - Clock skew
  - Clock period design margins
- In current and future generation designs
  - Wire latency becoming dominant latency of critical path
  - Due to growing side-wall capacitance
  - Brings a spatial dimension to architecture optimization
    - E.g., How long are the wires that will connect these two devices?

Determining the Latency of a Wire



Wire delay ~ RC

$$RC = \frac{\rho L}{t_m w}\frac{\varepsilon w L}{t_i} = \rho\varepsilon\frac{L^2}{t_m t_i}$$

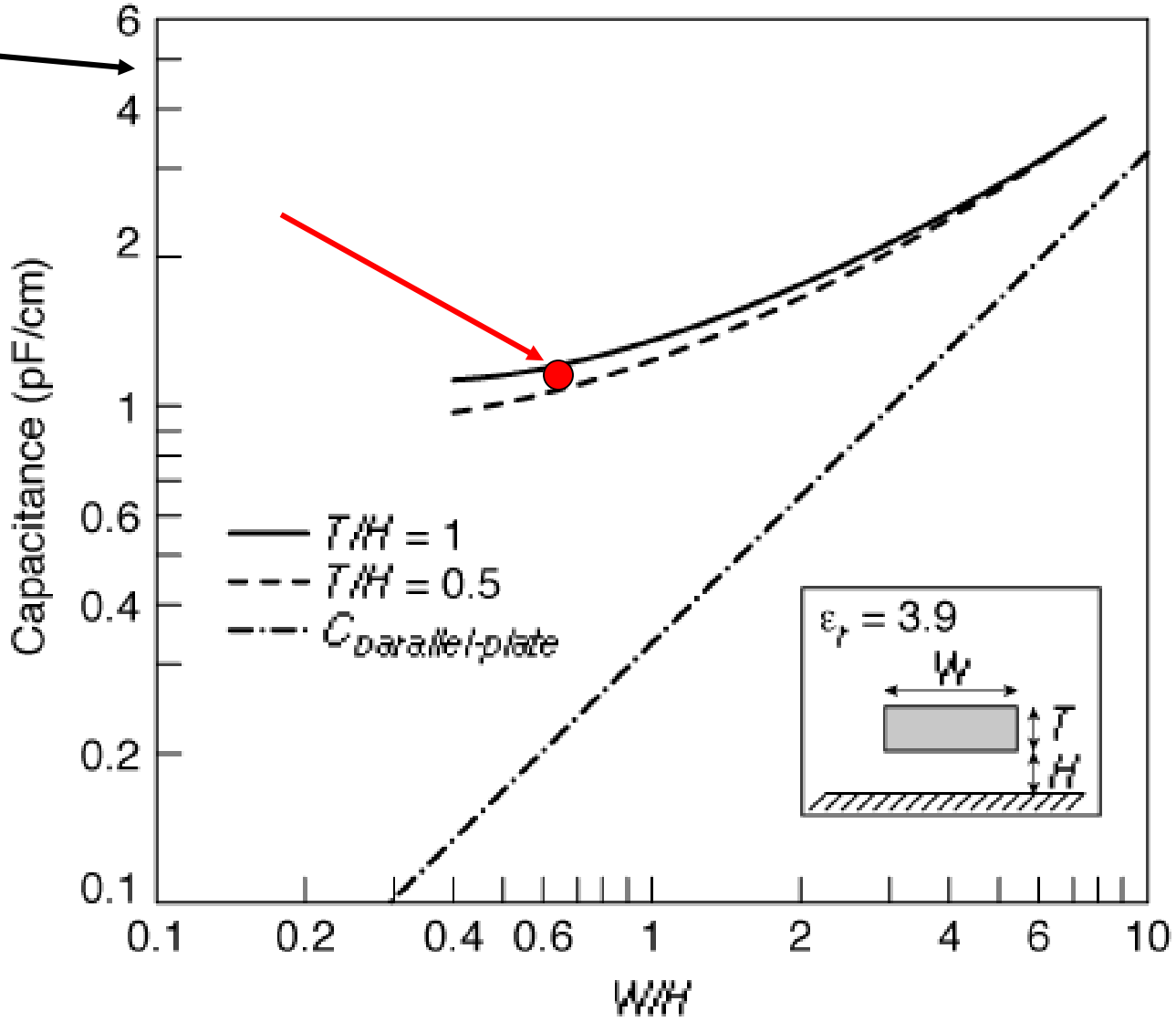


But reality is worse…. (Fringe)

(from [Bakoglu89])

For Intel 0.25u process

- W~=0.64
- T~=0.48
- H is around 0.9.



Moral of the “t_{CLK}" story

- As we shrink wire delay starts to dominate
  - Agarwal et. al. Clock Rate versus IPC: the End of the Road for Conventional Microarchitectures, ISCA 2000
  - d

And reducing the number of instructions executed…

- Sorry, wrong class.
  - Compilers can help with this (a lot in some cases)
  - So can ISA design, but usually not too much.
  - Making instructions too complex hurts ILP and t_{CLK}
- So on the whole reducing # of instructions doesn't look to be viable.
  - So ILP would seem to be "where it's at"

Optimizing CPU Performance

- Golden Rule: $t_{CPU} = N_{inst} * CPI * t_{CLK}$
- Given this, what are our options
  - Reduce the number of instructions executed
  - Reduce the cycles to execute an instruction
  - Reduce the clock period
- Our first focus: Reducing CPI
  - Approach: *Instruction Level Parallelism* (ILP)


Early Branch
Resulution

BRAT

- Simple idea:
  - When we mispredict we need to recover things to the state when the branch finished issuing.
    - RAT:
      - Just make a copy
      - Free list is non-trivial
    - RS
      - Can we just make a copy?
    - RoB
      - Easy money.

Note: the literature usually calls this “map table checkpoints” or “branch stack” or some such. I find that unwieldy so BRAT or BMAP will be used here. See “Checkpoint Processing and Recovery: Towards Scalable Large Instruction Window Processors” for a nice overview of options on branch misprediction recovery.

RAT

AR	Target
0	0
1	1
2	2
3	3

RS1	
RS2	
RS3	
RS4	

BRAT

AR	Target
0	
1	
2	
3	

RoB

PRF freelist:

Group formation