CSE372
Digital Systems Organization and Design Lab

Prof. Milo Martin

Unit 5: P37X Pipelined

Agenda for Today

- Discuss Lab 2
  - Due next week
- Discuss Lab 3
  - Part 1: Pipelined design
  - Part 2: Pipelined, superscalar design
- The evils of clock gating
  - Slides from UC-Berkeley
- Discuss video device and standards
  - Slides from UNC

Lab 2: Non-Pipelined Processor Lab

- Any comments or problems?
- Any problems getting it to work on the board?
  - No? Restricted Verilog has been successful!
- Reminder: final report due Friday

Final Lab: A More Advanced Datapath

- Result of lab 2: fully-functional P37x processor
  - CSE240 level of sophistication
    - Single cycle
  - We gave you the basic datapath "design"
- Goal of lab 3:
  - CSE372 level of sophistication (use what you have learned)
    - Pipelined processor (5 stage, fully-bypassed)
    - Branch prediction (simple BTB)
    - Superscalar execution (dual issue)
  - Exact design is up to you
    - Minimum baseline pipeline specified (part 1)
    - Up to you to decide how to improve design (part 2)

Final Lab, Part 1: Pipelined Processor

- Familiar 5-stage pipeline
  - Fetch, Decode, Execute, Memory, Writeback
  - Fully bypassed
- One-cycle "load use" penalty
  - A dependent instruction right after the load
- Branch handling
  - Resolved in "execute" stage, two-cycle penalty
  - Simple branch predictor to efficiently execute branches
- Performance counters
  - Cycle counter
  - Instruction counter, branch stall counter, load stall counters

Final Lab, Part 2: Improve IPC

- Part 2: use what you have learned
  - Analysis and design
- We're giving you some simple "toy" benchmarks
  - Goal: improve the IPC of these benchmarks
    - Keeping 5-stage pipeline, 1-cycle load-use delay
  - Task: look at programs, uses ideas from CSE372 to improve them
- More opportunity for creativity
  - longer rope

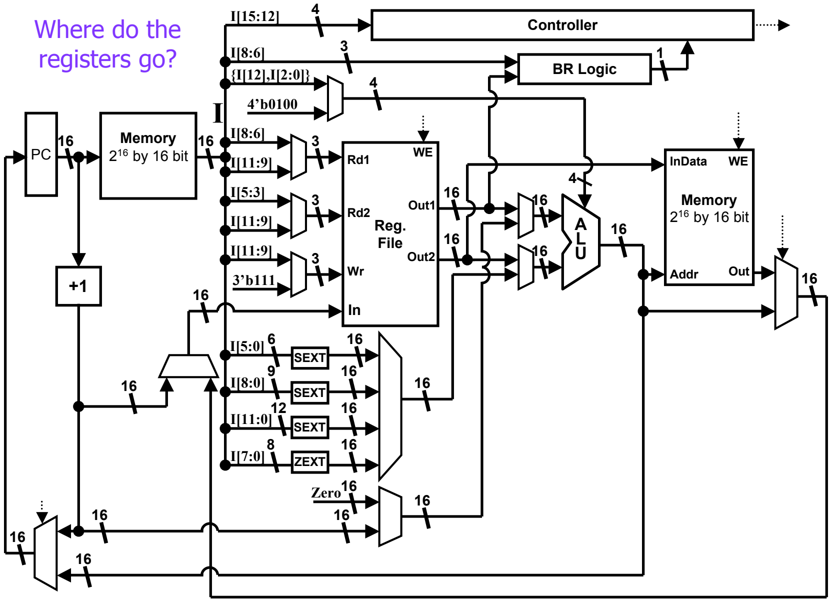
Final Lab: Tiered Designs & Grading

- Last year: honors points
- "A" level: IPC > 1 on a few benchmarks
  - Dual issue superscalar
- "A-" level: significantly reduce lost IPC due to branches
  - Extend branch predictor
- "B+" level: build pipeline specified in assignment
  - 5-stage pipeline
- "B" level: nothing!
  - You're done after lab 2

Important First Step: Design Document

- The pipelined datapath design is up to you
  - As is how to test it
  - As is how your choose to improve IPC
- Design document - Due Friday, March 30th
  - Describe the datapath
    - Where are the pipeline registers?
    - List or table of which signals are latched in which stages
  - Describe control
    - Bypassing logic, any tricky bypassing cases
  - Include a diagram (or diagrams) of the datapath
  - Design/schematics of any new components (e.g., branch predictor)
  - **Include testing strategy**
  - How are you going to improve IPC?
    - Separate superscalar design issues from single-issue design?

Where do the registers go?



Bypassing, Stalling, and Flushing

- Bypassing
  - Which value to use in a given stage?
  - Control logic looks at "recent past"
  - Look at instruction in later stage
- Stalling
  - Dependent instruction after load
  - How?
- Flushing
  - Speculatively execute instructions after branch
  - Using the prediction
  - If wrong, need to cancel two instructions
    - Fetch and Decode
  - Again, how?

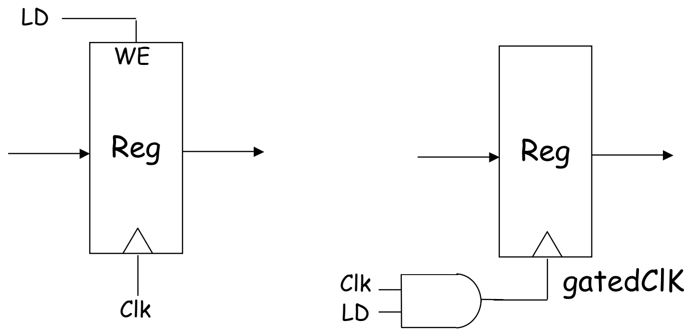
Simple Branch Predictor

- Predict the "next PC" for an instruction
  - Predict during Fetch: PC in, next-PC prediction out
  - Train at Execute: PC in, actual next-PC in, write enable in
- Simplest: Two registers
  - "Tag" register
  - "Next PC" register
  - If the tag matches fetch PC, return value of "Next PC" register
    - Else, return PC+1
- Detect misprediction via comparing PC
  - If wrong, train predictor
  - Write PC into tag register, write "actual next PC" into "next PC"

Other Thoughts on Lab 3

- Good news: gain some experience with design
  - More opportunities for creativity
  - More than 4 weeks to complete project, groups of three
- Much harder than it looks
  - Last year, students were shocked at how much harder it was
    - "Thrown into the deep end"
  - All through CSE240 & CSE372 thus far, we've given you design
    - Good designs dramatically simplify implementation effort
    - Example: how much longer would Snake/Breakout have taken without using our design?
- In retrospect, students skimped on design (& document)

Why Gating of Clocks is Bad!

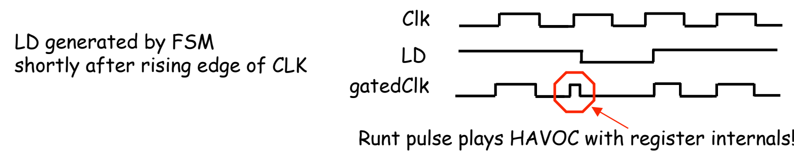


GOOD

BAD

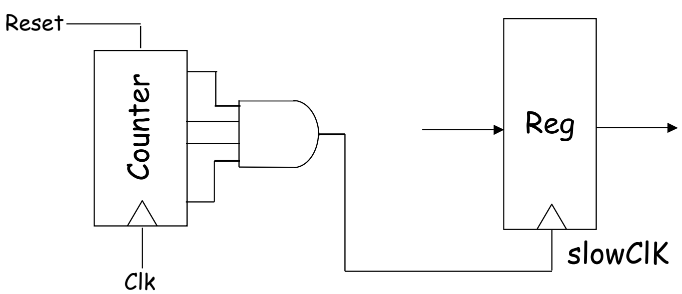
Do NOT Mess With Clock Signals!

Why Gating of Clocks is Bad!



Do NOT Mess With Clock Signals!

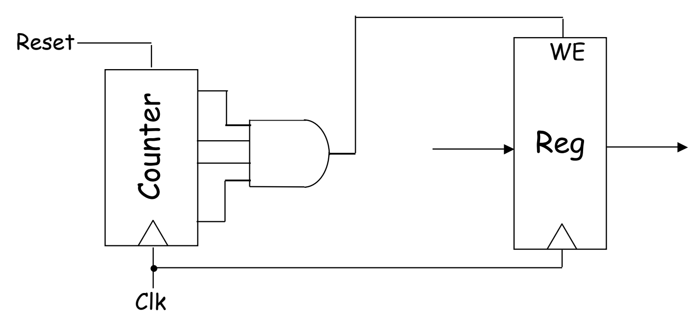
Gating of Clocks: Bad



BAD

Do NOT Mess With Clock Signals!

Non-Gating of Clocks: Good



Good!

Do NOT Mess With Clock Signals!

Video Device

- How does our video device actually work?

- Frame buffer
  - RAM that holds the current image on the screen
  - Hardware walks over frame buffer to generate analog signal
  - In LC-3 and P37x, we puts this frame buffer right in memory
    - Most systems today have dedicated frame buffers

- In some classes, they make you build the video circuit, too
  - Switch to slides from UNC