ECE/CS 552: Single Cycle Datapath

© Prof. Mikko Lipasti

Lecture notes based in part on slides created by Mark Hill, David Wood, Guri Sohi, John Shen and Jim Smith

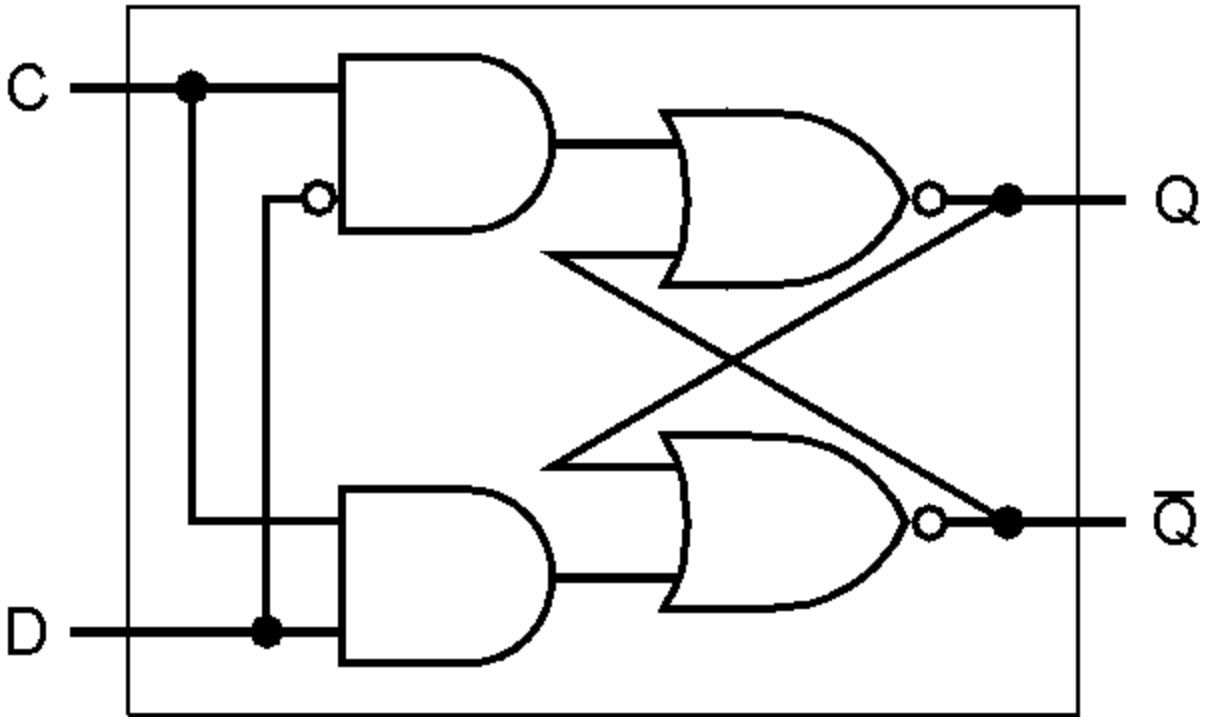
Processor Implementation

- Forecast – heart of 552 – key to project
  - Sequential logic design review (brief)
  - Clock methodology (FSD)
  - Datapath – 1 CPI
    - Single instruction, 2's complement, unsigned
- Next:
  - Control
  - Multiple cycle implementation (information only)
  - Microprogramming
  - Exceptions

Review Sequential Logic

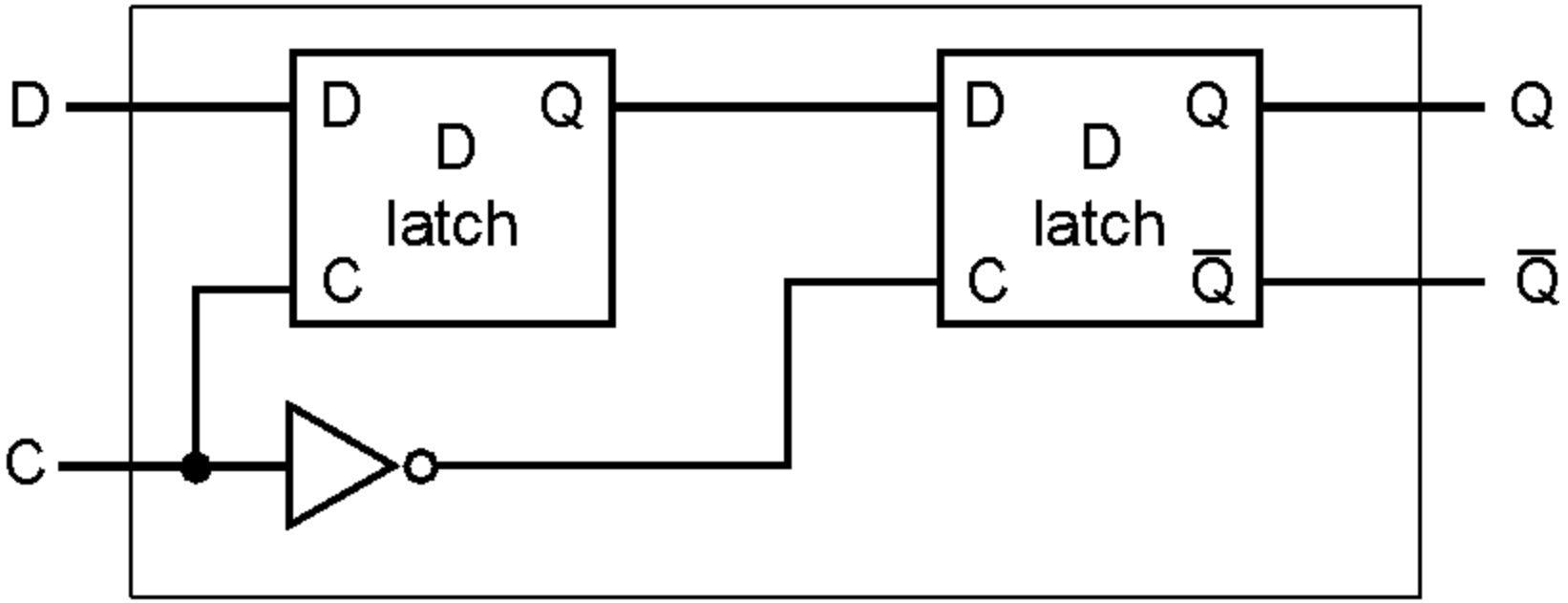
- Logic is combinational if output is solely function of inputs
  - E.g. ALU of previous lecture
- Logic is sequential or “has state” if output function of:
  - Past and current inputs
  - Past inputs remembered in “state”
  - Of course, no magic

Review Sequential Logic



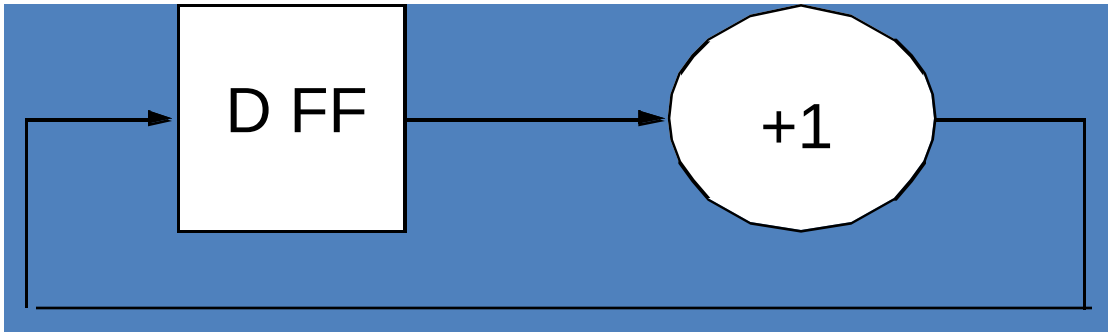
- Clock high, Q = D, ~Q = ~D after prop. Delay
- Clock low Q, ~Q remain unchanged
  - Level-sensitive latch

Review Sequential Logic



- Master/Slave D flip-flop
  - While clock high, Q_M follows D, but Q_S holds
  - At falling edge Q_M propagates to Q_S
  - *Opaque* except at falling (rising) clock edge

Review Sequential Logic



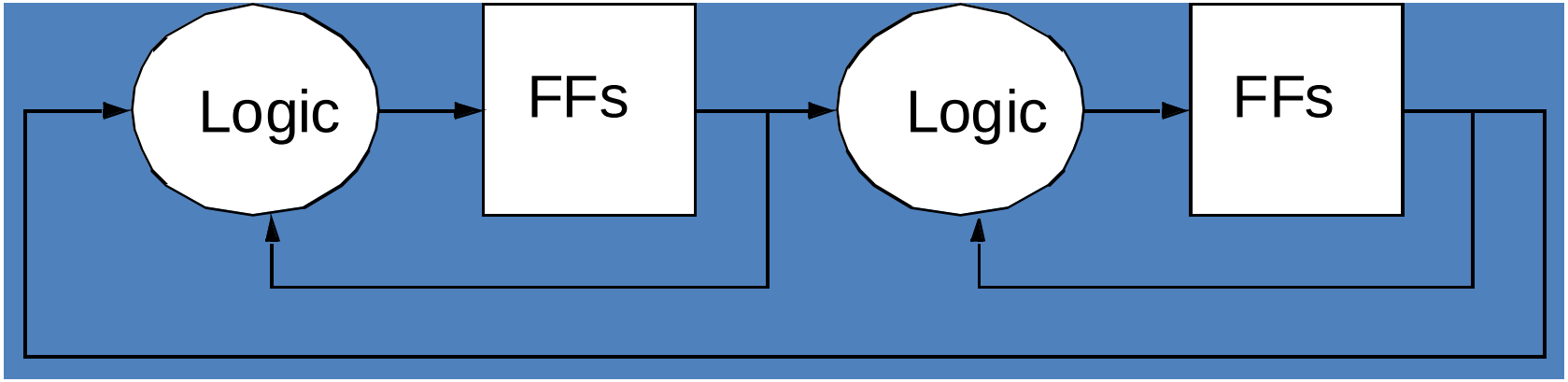
- Why can this fail for a latch?
  - Latch is transparent when clock is high (low)
  - Creates combinational loop
  - Increment evaluates unknown number of times

Clocking Methology

- Motivation
  - Design data and control without considering clock
- Use Fully Synchronous Design (FSD)
  - Just a convention to simplify design process
  - Restricts design freedom
  - Eliminates complexity, can guarantee timing correctness
  - Not really feasible in real designs: off-chip I/O
  - Even in ECE 554 you will violate FSD

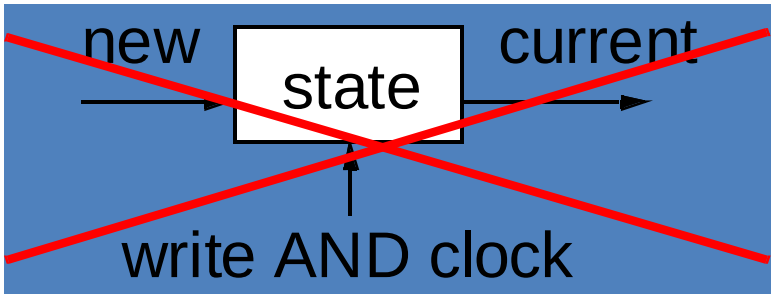
Our Methodology

- Only flip-flops
- All on the same edge (e.g. falling)
- All with same clock
  - No need to draw clock signals
- All logic finishes in one cycle

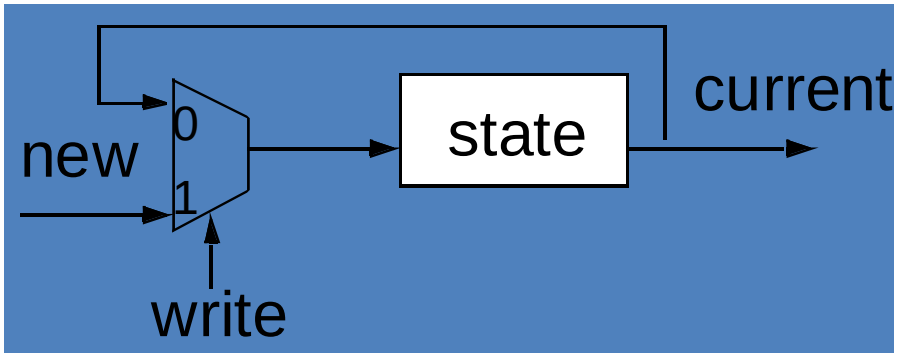


Our Methodology, cont'd

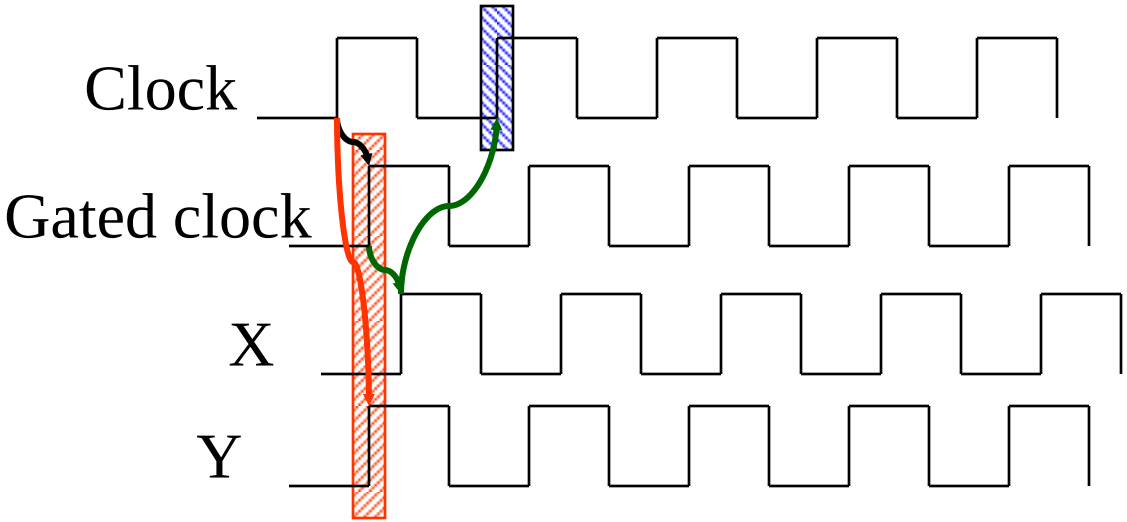
- No clock gating!
  - Book has bad examples

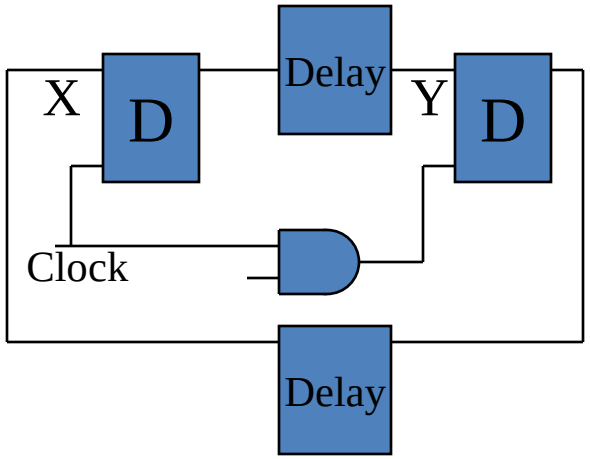


- Correct design:



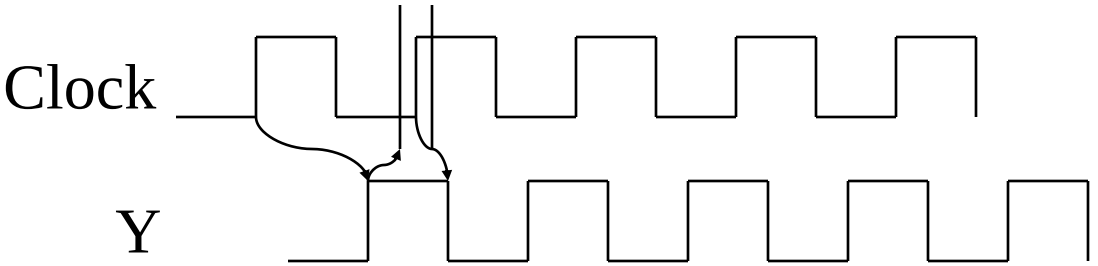
Delayed Clocks (Gating)

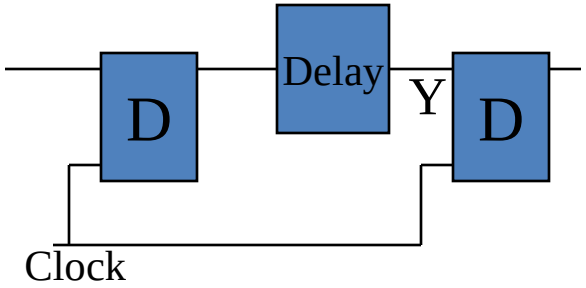




- Problem:
  - Some flip-flops receive gated clock late
  - Data signal may violate setup & hold req't

FSD Clocking Rules





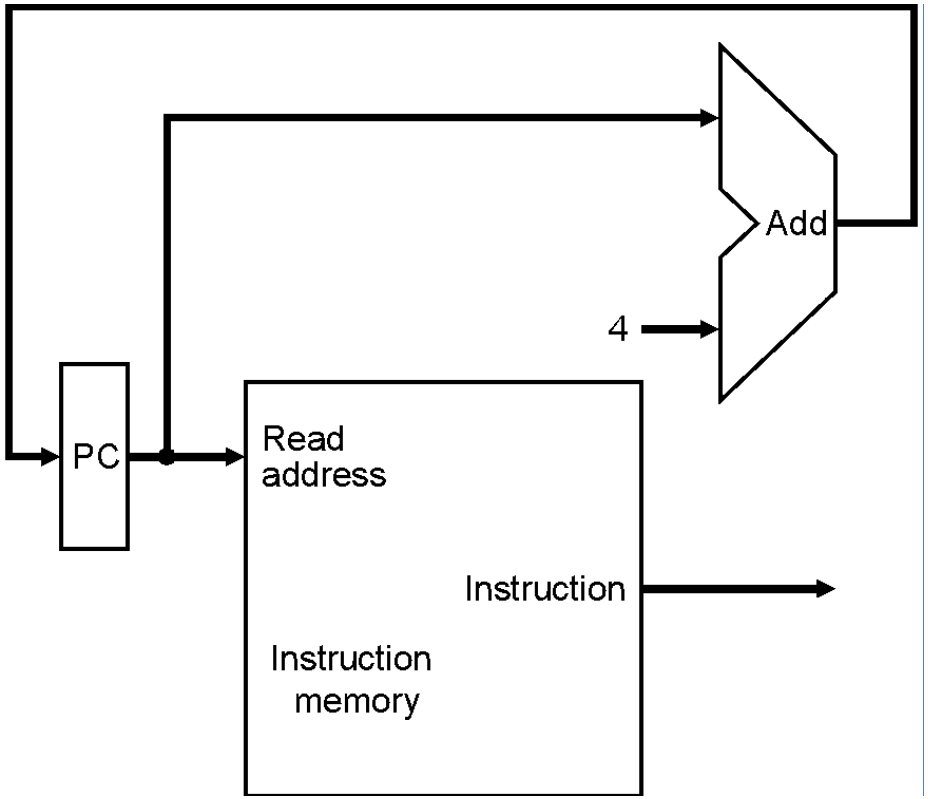
- T_{clock} = cycle time
- T_{setup} = FF setup time requirement
- T_{hold} = FF hold time requirement
- T_{FF} = FF combinational delay
- T_{comb} = Combinational delay
- FSD Rules:
  - $T_{clock} > T_{FF} + T_{combmax} + T_{setup}$
  - $T_{FF} + T_{combmin} > T_{hold}$

Datapath – 1 CPI

- Assumption: get whole instruction done in one long cycle
- Instructions:
  - and, lw, sw, & beq
- To do
  - For each instruction type
  - Putting it all together

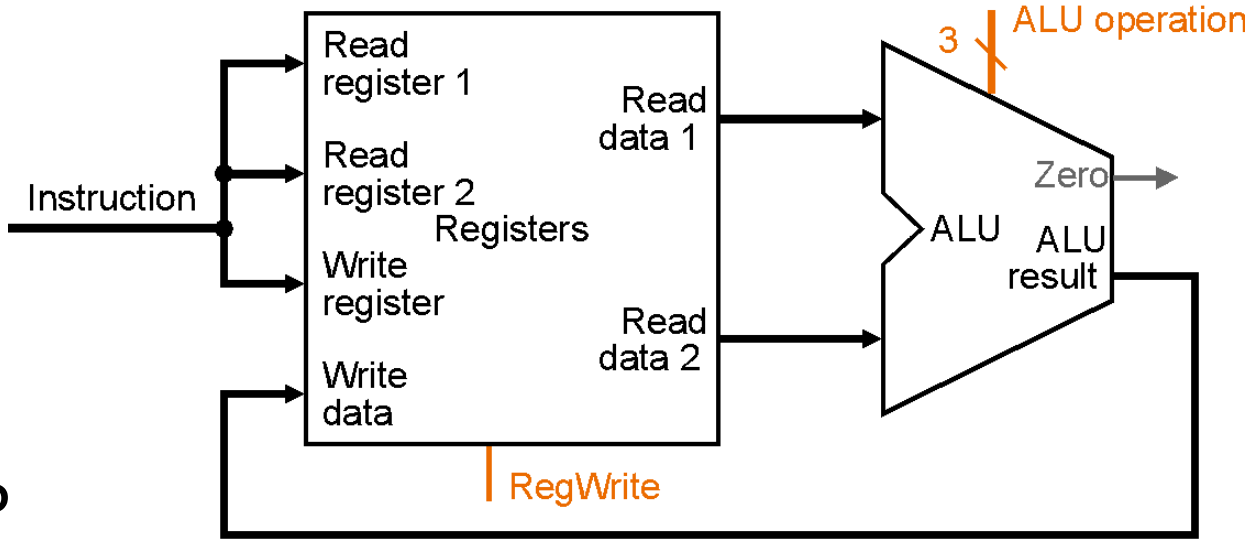
Fetch Instructions

- Fetch instruction, then increment PC
  - Same for all types
- Assumes
  - PC updated every cycle
  - No branches or jumps
- After this instruction fetch next one



ALU Instructions

- and $1, $2, $3 # $1 <= $2 & $3



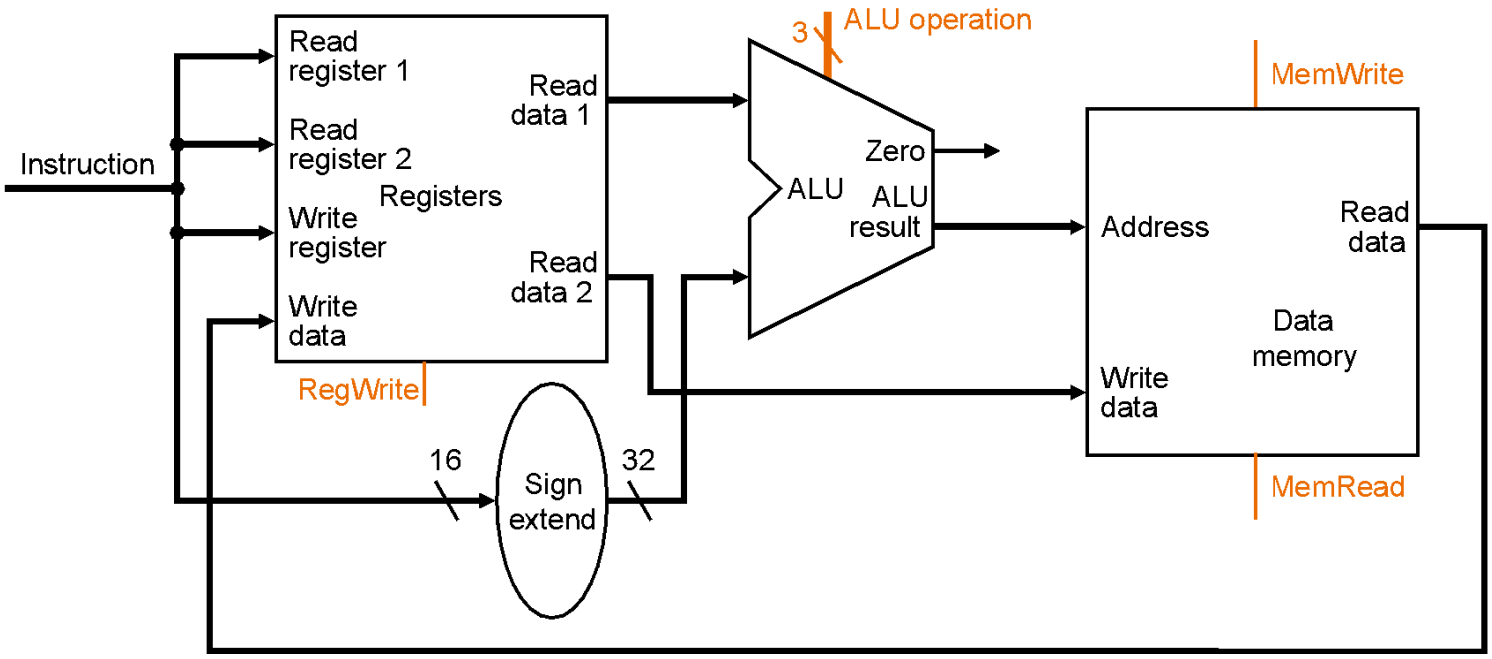
- E.g. MIP

Opcode	rs	rt	rd	shamt	function
6	5	5	5	5	6

Load/Store Instructions

- lw $1, immed($2) # $1 <= M[SE(immed)+$2]
- E.g. MIPS I-format:

Opcode	rt	rt	immed
6	5	5	16



Branch Instructions

- beq $1, $2, addr # if ($1==$2) PC = PC + addr<<2
- Actually

  newPC = PC + 4

  target = newPC + addr << 2 # in MIPS offset from newPC

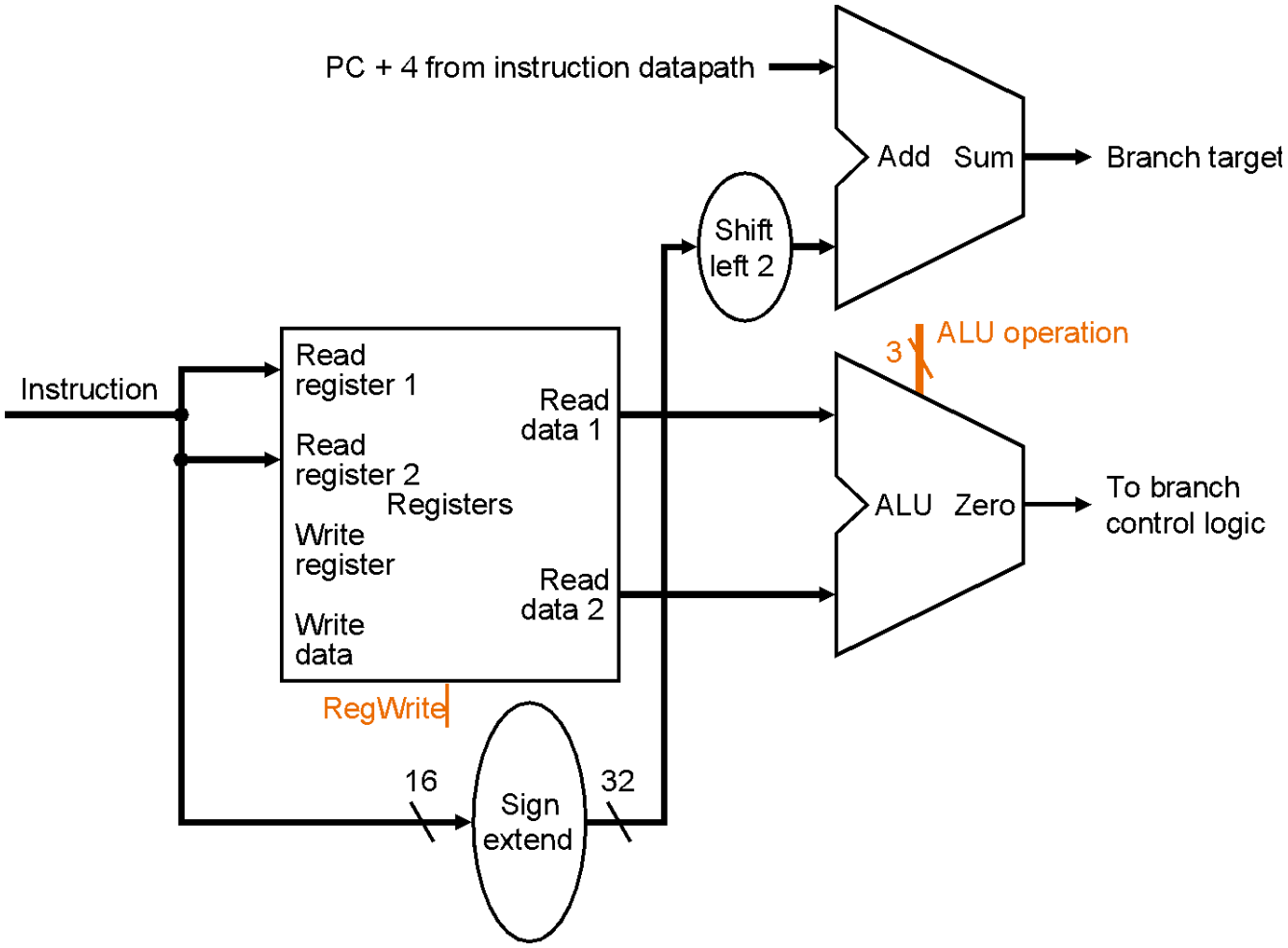
  if (($1 - $2) == 0)

      PC = target

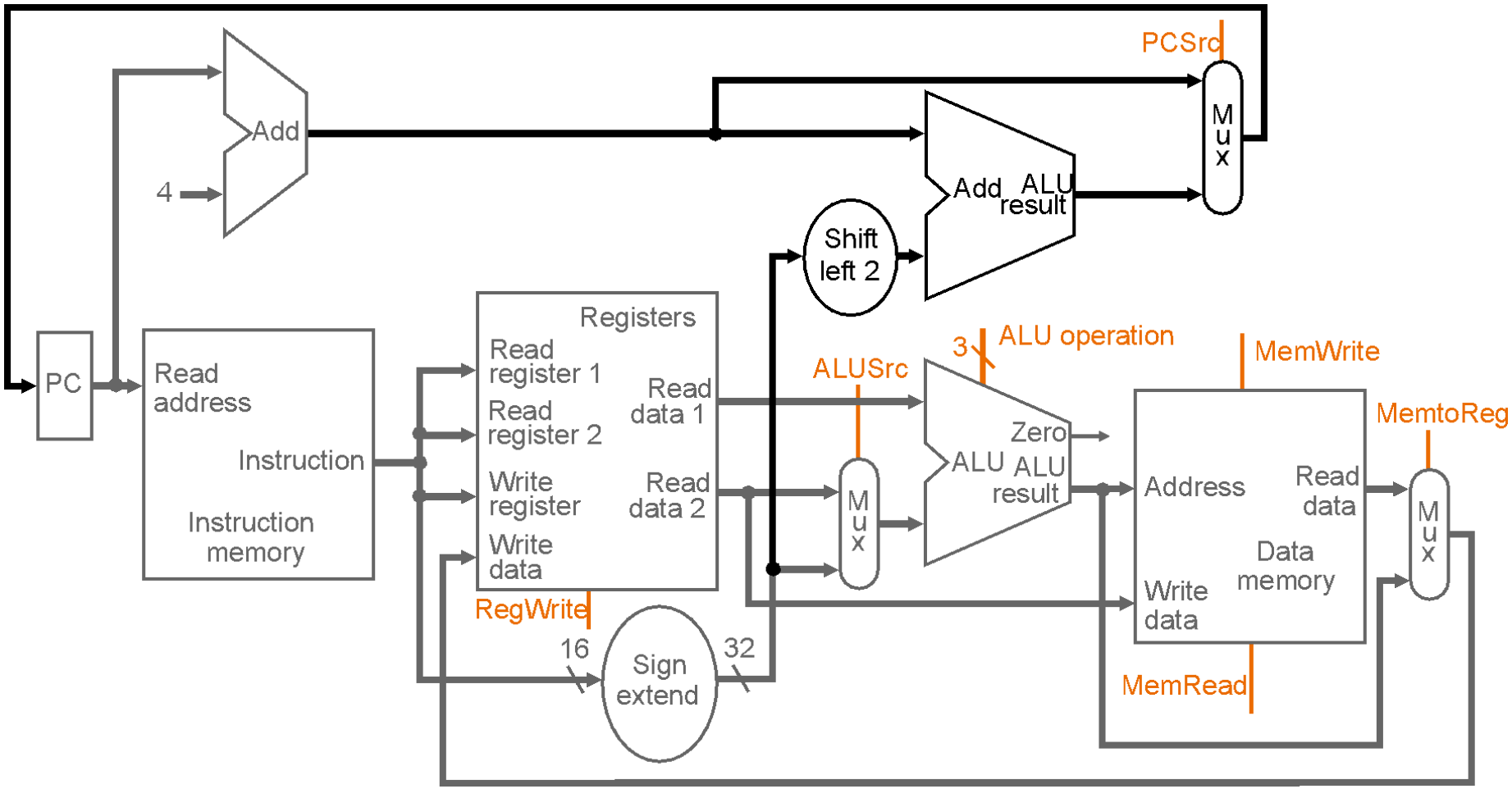
  else

      PC = newPC

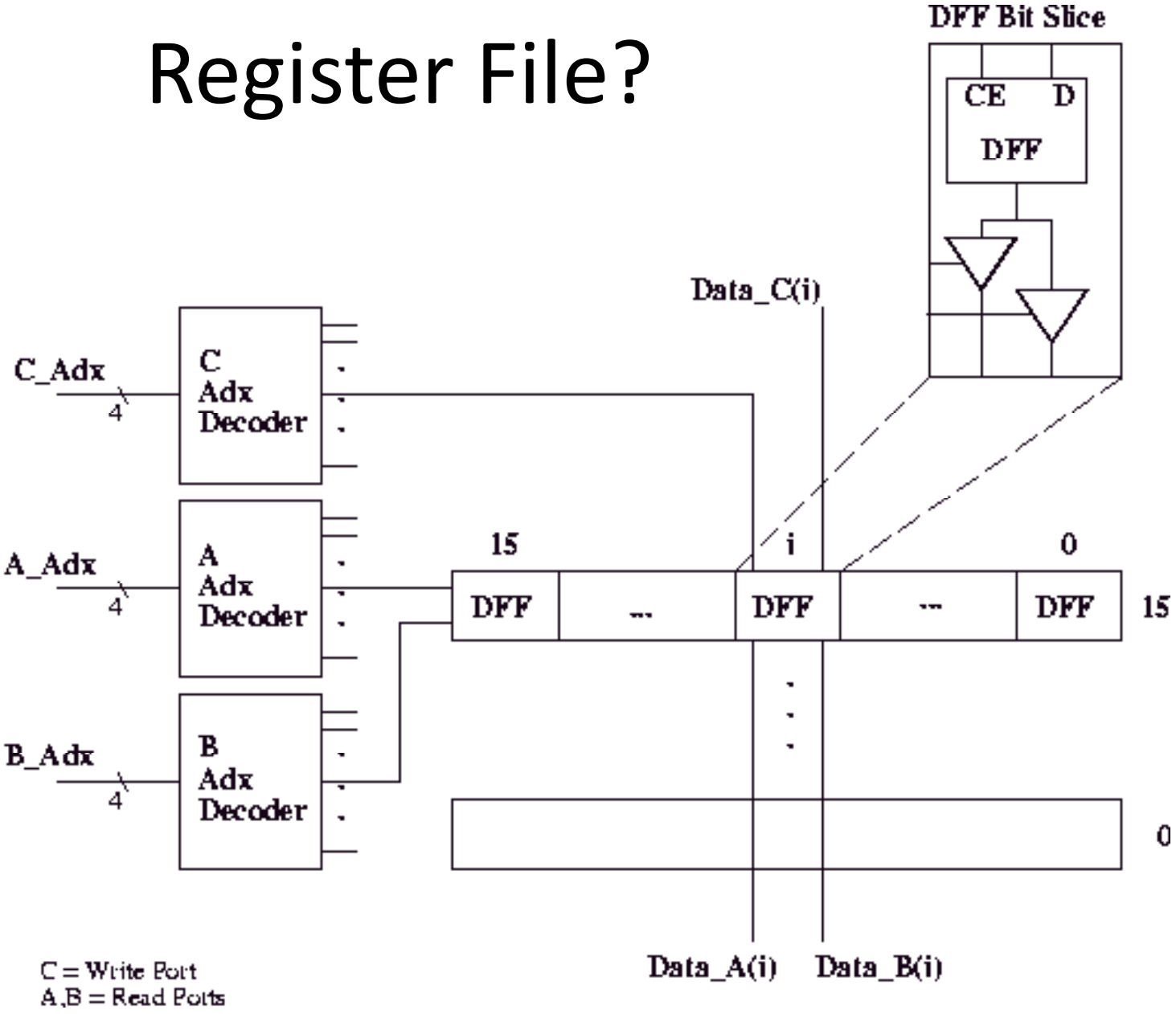
Branch Instructions



All Together



Register File?



Summary

THE UNIVERSITY of WISCONSIN MADISON

- Sequential logic design review (brief)
- Clock methodology (FSD)
- Datapath – 1 CPI
  - ALU, lw, sw, beq instructions