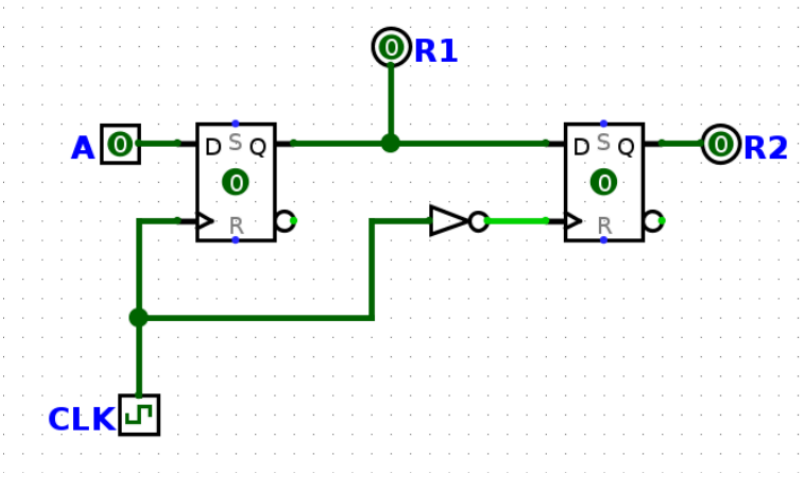
CS61C Spring 2026

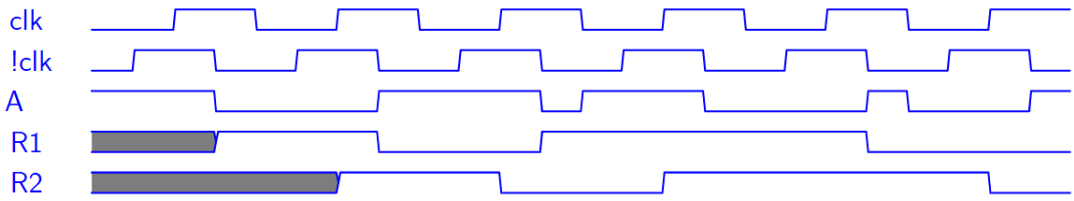
Single-Cycle Datapath

Discussion 8

1 SDS Intro

1.1 Fill out the timing diagram. The clock period (rising edge to rising edge) is 8ps. For every register, clk-to-q delay is 2ps, setup time is 4ps, and hold time is 2ps. NOT gates have a 2ps propagation delay, which is already accounted for in the !clk signal given.

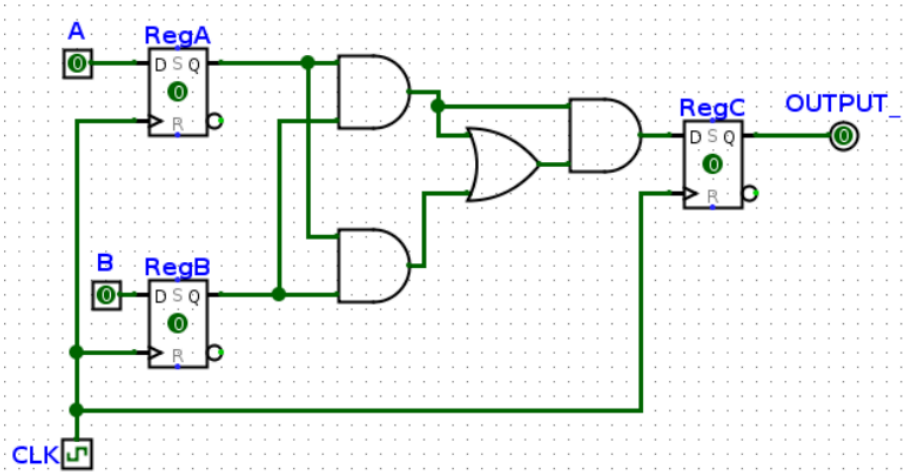




1.2 In the circuit below:

- RegA and RegB have setup, hold, and clk-to-q times of 4ns,
- All logic gates have a delay of 5ns
- RegC has a setup time of 6ns.

What is the maximum allowable hold time for RegC? What is the minimum acceptable clock cycle time for this circuit, and clock frequency does it correspond to?



The maximum allowable hold time for RegC is how long it takes for RegC's input to change, so `(clk-to-q of A or B) + shortest CL time = 4 + (5 + 5) = 14 ns`.

The minimum acceptable clock cycle time is `clk-to-q + longest CL time + setup time = 4 + (5 + 5 + 5) + 6 = 25 ns`.

25 ns corresponds to a clock frequency of $\left(\frac{1}{25*10^{-9}}\right)s^{-1} = 40$ MHz

2 Single-Cycle CPU

For this worksheet, we will be working with the single-cycle CPU datapath provided on the last page.

2.1 List all possible values that each control signal may take on for the single cycle datapath, then briefly describe what each value means for each signal.

(a) PCSel

- 0: PC = OldPC + 4
- 1: PC = ALU output (for branches, jumps, etc...)

(b) RegWEn

- 0: WB value not written to register in RegFile
- 1: WB value is written

(c) ImmSel

- 0-4: used for I, B, S, J, U-type immediates.
- 5-7: unused

(d) BrEq

- 0: set to 0 when `rdata1` is not equal to `rdata2`
- 1: set to 1 when `rdata1` is equal to `rdata2`

(e) BrLt

- 0: set to 0 when `rdata1` is not less than `rdata2`
- 1: set to 1 when `rdata1` is less than `rdata2`

(f) ALUSel

Note: this uses the same design reference 61C does; may differ based on CPU design. You do NOT have to memorize all of these!

- 0: `add`
- 1: `sll`
- 2: `slt`
- 3: unused
- 4: `xor`
- 5: `srl`
- 6: `or`
- 7: `and`
- 8: `mul`
- 9: `mulh`
- 10: unused
- 11: `mulhu`
- 12: `sub`
- 13: `sra`
- 14: unused
- 15: `bsel`

(g) MemRW

- 0: disables writing to memory
- 1: enables writing to memory

(h) WBSel

- 0: DMEM read output
- 1: ALU output
- 2: PC + 4
- 3: unused

2.2 Fill out the following table with the control signals for each instruction based on the datapath on the last page.

- If the value of the signal does not affect the execution of an instruction, use the * (don't care) symbol to indicate this.
- If the value of the signal does affect the execution, but can be different depending on the program, list all possible values (for example, for a signal that may output 0 and 1, write 0/1).
- For ALUSel, write the ALU operation (`add`, `or`, `sll`, ...)

The first row has been filled out for you.

	BrEq	BrLT	PCSel	ImmSel	BrUn	ASel	BSel	ALUSel	MemRW	RegWEn	WBSel
add	*	*	0 (PC + 4)	*	*	0 (Reg)	0 (Reg)	add	0	1	1 (ALU)
ori	*	*	0	I	*	0 (Reg)	1 (Imm)	or	0	1	1 (ALU)
lw	*	*	0	I	*	0 (Reg)	1 (Imm)	add	0	1	0 (MEM)
sw	*	*	0	S	*	0 (Reg)	1 (Imm)	add	1	0	*
beq	0/1	*	0/1	B	*	1 (PC)	1 (Imm)	add	0	0	*
jal	*	*	1 (ALU)	J	*	1 (PC)	1 (Imm)	add	0	1	2 (PC + 4)
blt	*	0/1	0/1	B	0	1 (PC)	1 (Imm)	add	0	0	*

3 Timing the Datapath

Clocking review:

- A **state element** is an element connected to the clock (denoted by a triangle at the bottom). The **input signal** to each state element must stabilize before each **rising edge**. For example, registers are state elements.
- The **critical path is the longest delay path between any two state elements in the circuit (with no other state elements along that path)**. The circuit cannot be clocked faster than this, since anything faster would mean that the correct value is not guaranteed to reach the state element in the alloted time. We can shorten the critical path by placing registers along it, thus reducing the amount of logic between state elements (i.e. registers);.

For this exercise, the times for each circuit element is given as follows:

Clk-to-Q	RegFile Read	PC/RegFile Setup	Mux	Adder
5ns	35ns	20ns	15ns	20ns

ALU	Branch Comp	Imm Gen	MEM Read	DMEM Setup
100ns	50ns	45ns	300ns	200ns

3.1 Mark an X for the datapath stages used by each instruction

	IF	ID	EX	MEM	WB
`add`	X	X	X		X
`ori`	X	X	X		X
`lw`	X	X	X	X	X
`sw`	X	X	X	X	
`beq`	X	X	X		
`jal`	X	X	X		X

3.2 Ignoring the length of a clock cycle, how long does it take to execute each instruction? Assume that the setup times to the RegFile and the PC are the same.

Hint: For each instruction, first identify which elements of the datapath are being used. What is the longest path between two state elements?

(a) jal

$$\begin{aligned}\text{jal} &= \text{clk-to-Q} + \text{Mem-Read} + \text{Imm-Gen} + \text{Mux(BSel)} + \text{ALU} + \text{Mux(PCSel)} + \text{PCSetup} \\ &= 5\ \text{ns} + 300\ \text{ns} + 45\ \text{ns} + 15\ \text{ns} + 100\ \text{ns} + 15\ \text{ns} + 20\ \text{ns} \\ &= 500\ \text{ns}\end{aligned}$$

(b) lw

$$\begin{aligned}\text{lw} &= \text{clk-to-Q} + \text{Mem-Read} + \max(\text{RegFileRead} + \text{Mux(ASel)}, \text{Imm-Gen} + \text{Mux(BSel)}) \\ &\quad + \text{ALU} + \text{Mem-Read} + \text{Mux(WBSel)} + \text{RegFileSetup} \\ &= 5\text{ ns} + 300\text{ ns} + 60\text{ ns} + 100\text{ ns} + 300\text{ ns} + 15\text{ ns} + 20\text{ ns} \\ &= 800\text{ ns}\end{aligned}$$

(c) sw

$$\begin{aligned}\text{sw} &= \text{clk-to-Q} + \text{Mem-Read} + \max(\text{RegFileRead} + \text{Mux(ASel)}, \text{Imm-Gen} + \text{Mux(BSel)}) \\ &\quad + \text{ALU} + \text{DMEM Setup} \\ &= 5\text{ ns} + 300\text{ ns} + 60\text{ ns} + 100\text{ ns} + 200\text{ ns} \\ &= 665\text{ ns}\end{aligned}$$

3.3 Which instruction(s) are responsible for the critical path?

Load instructions use all 5 datapath stages and take the longest to execute (`lw` calculated above takes 800 ns) .

3.4 What is the highest clock frequency for this single cycle datapath?

$$\begin{aligned}\text{Maximum Frequency} &= \frac{1}{\text{critical path delay}} \\ &= \frac{1}{800\text{ nanoseconds}} \\ &= \frac{1}{800 * 10^{-9}\text{ seconds}} \\ &= 1,250,000\text{ s}^{-1} \\ &= 1.25\text{ MHz}\end{aligned}$$

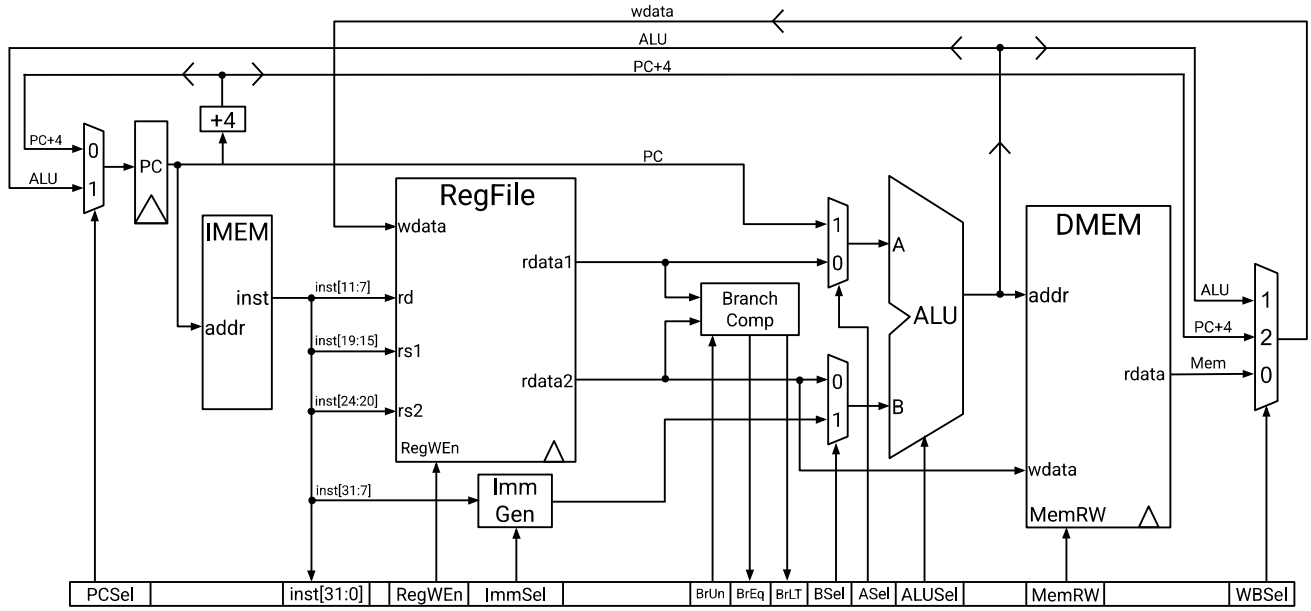
3.5 Why is the single-cycle datapath inefficient?

At any given time, most of the parts of the single cycle datapath are not being used. Even though not every instruction exercises the critical path, the datapath can only be clocked as fast as the slowest instruction.

3.6 How can you improve its performance? What is the purpose of pipelining?

Performance can be improved with pipelining, or putting registers between stages so that the amount of combinational logic between registers is reduced. This shortens the critical path, allowing for a faster clock time.

Single-Cycle Datapath Diagram



5-Stage Datapath Diagram