# Intro to Digital Design
# L6: Synchronous Timing Constraints

**Instructor:** Naomi Alterman

**Teaching Assistants:**

Derek de Leuw
Isabel Froelich
Kevin Hernandez
Aarjav Jain
Packard Stephenson

# Administrivia

Diagrams!
Start earlier

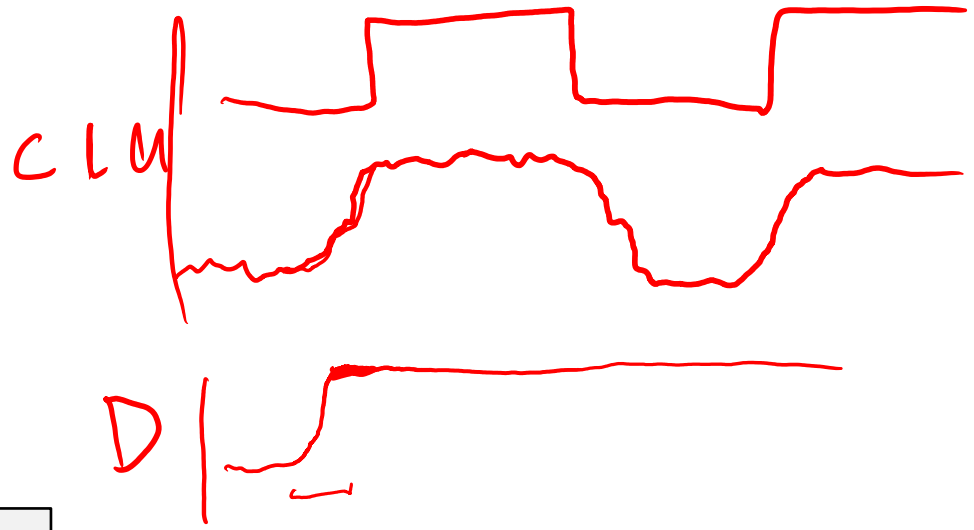
- Lab 6 out later today - Tug of War game
  - *Bigger* step up in difficulty from Lab 5
    - Putting together complex system built of multiple FSMs – interconnections!
    - Bonus points for smaller resource usage

# Outline

- **Synchronous Timing Constraints**

# Reminder: Flip Flops

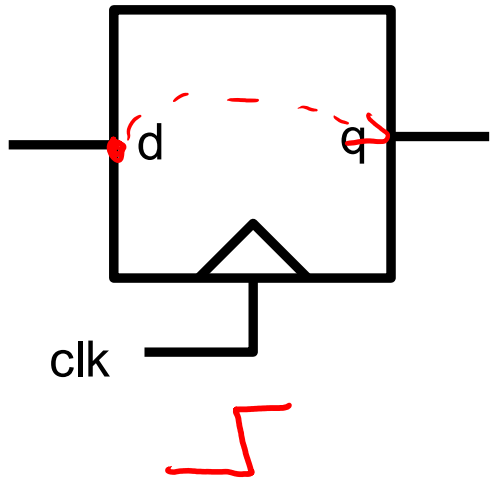
- A single bit of memory
- Copy d to q on the rising edge of the clock signal




```
module DFF (q, d, clk);
  output logic q;  // q is state-holding
  input  logic d, reset, clk;

  always_ff @(posedge clk) begin
     q <= d;
  end

endmodule
```



# Flip-Flop Timing Terminology (1/2)

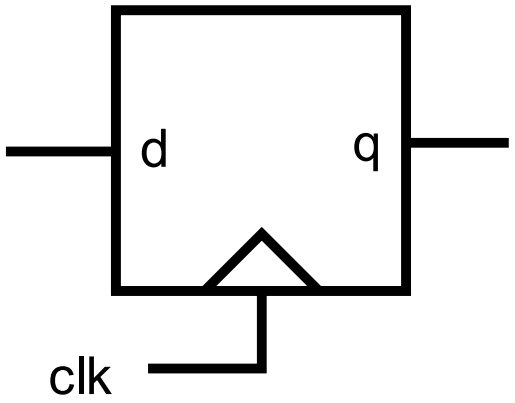
- Camera Analogy: motion blur in photos
  - *Don't move* while camera shutter is opening
  - *Don't move* while camera shutter is closing
  - *Check for blurriness* once image appears on the display

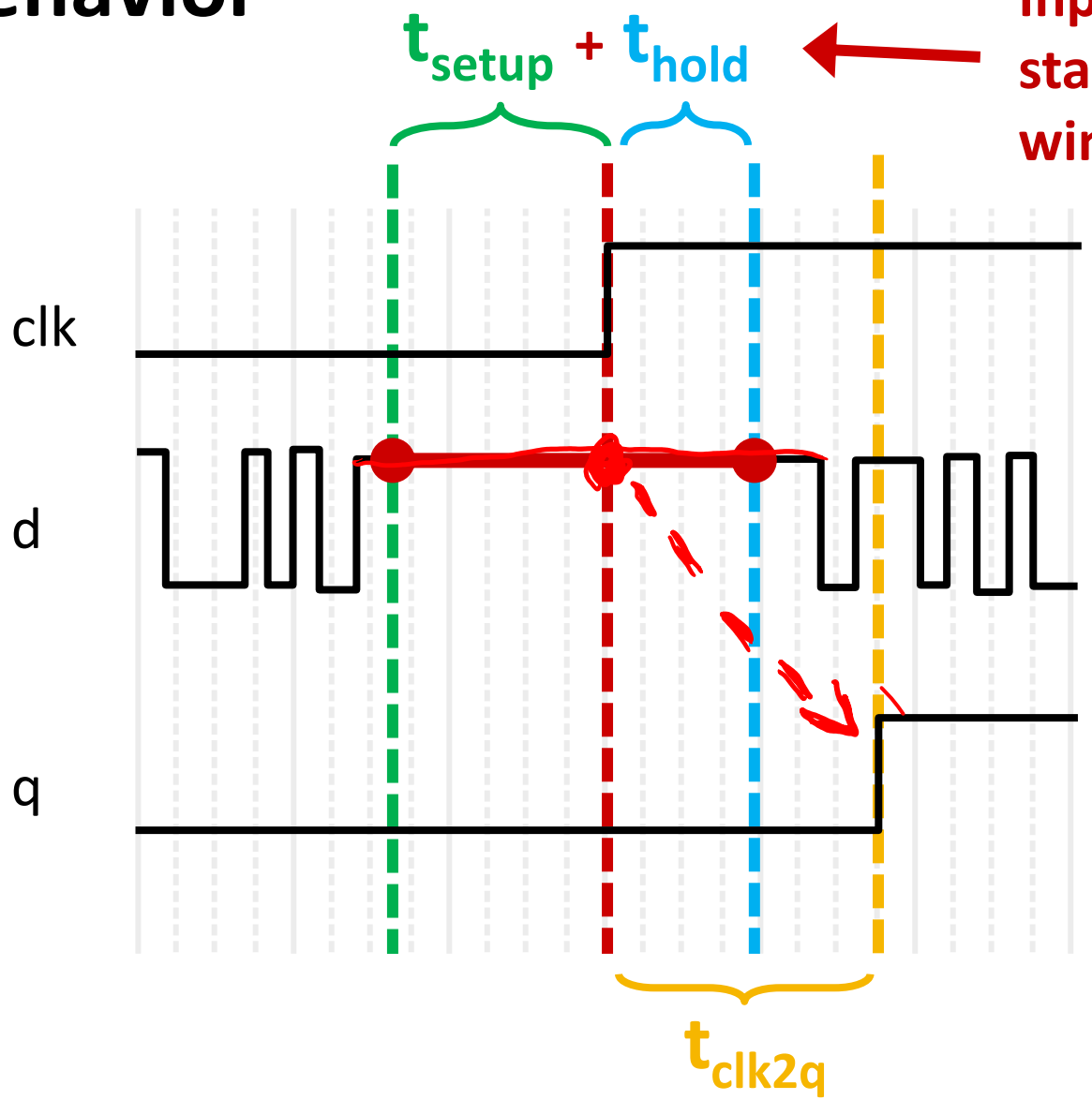

Flip flop sampling its input

# Flip-Flop Timing Terminology (2/2)

- Now applied to sequential logic elements:
  - *Setup Time:* how long the input must be stable *before* the CLK trigger for proper input read
  - *Hold Time:* how long the input must be stable *after* the CLK trigger for proper input read
    - *"CLK-to-Q" Delay:* how long it takes the output to change, measured from the CLK trigger

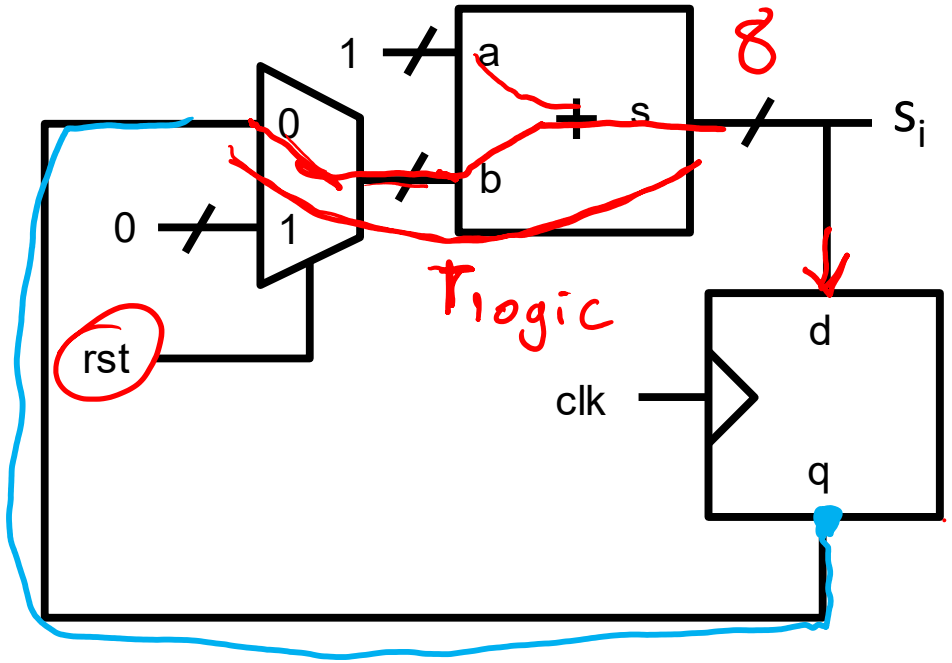
# Flip-Flop Timing Behavior

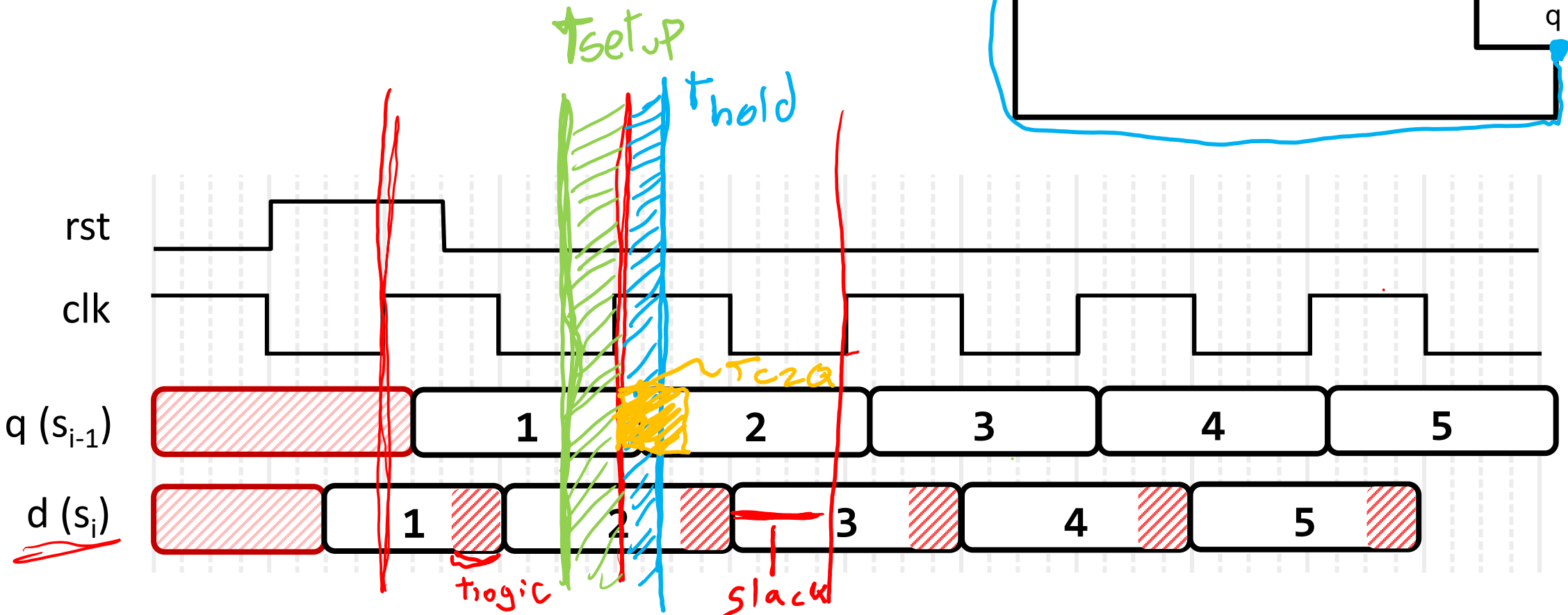


# Stopwatch: timing analysis

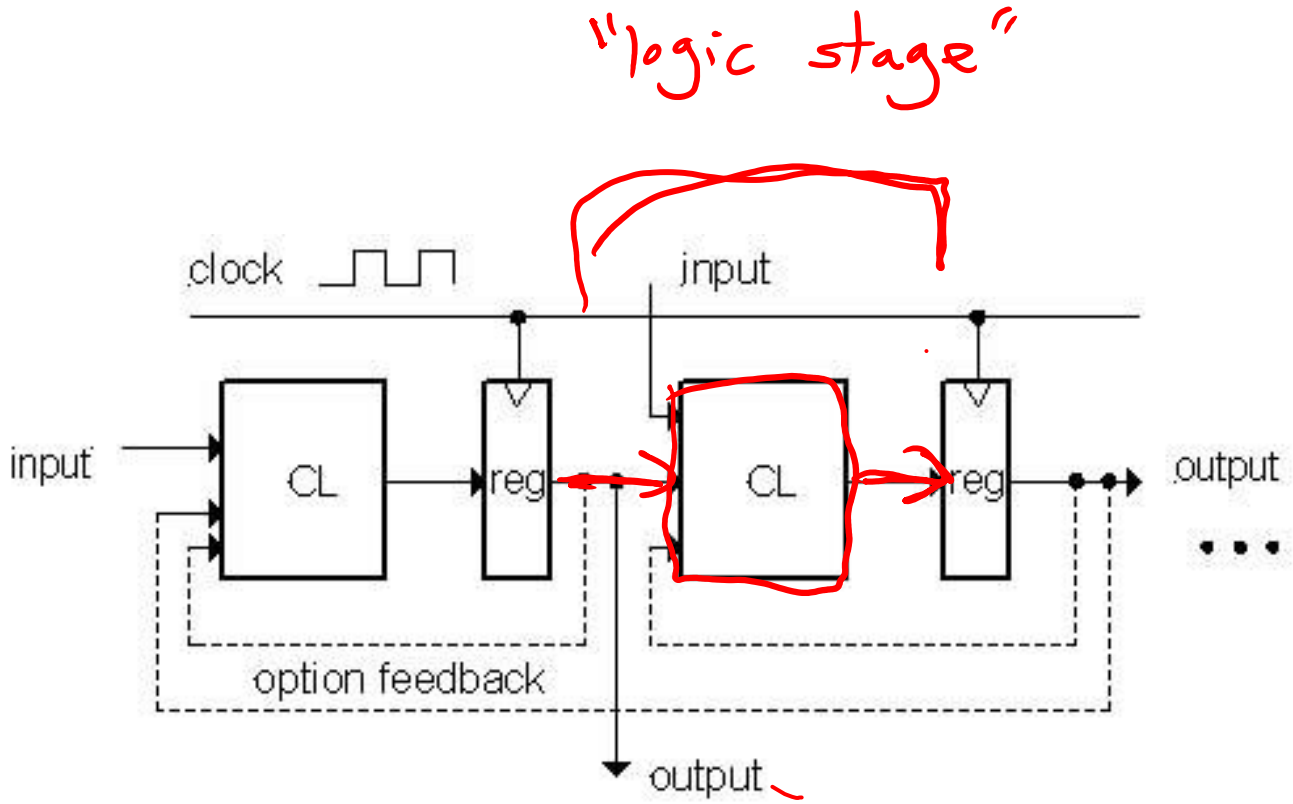
- As bits ripple through adder, $S_i$ is temporarily wrong!
- BUT! Register always captures correct value
- In good circuits, signal stays constant around the rising edge of CLK



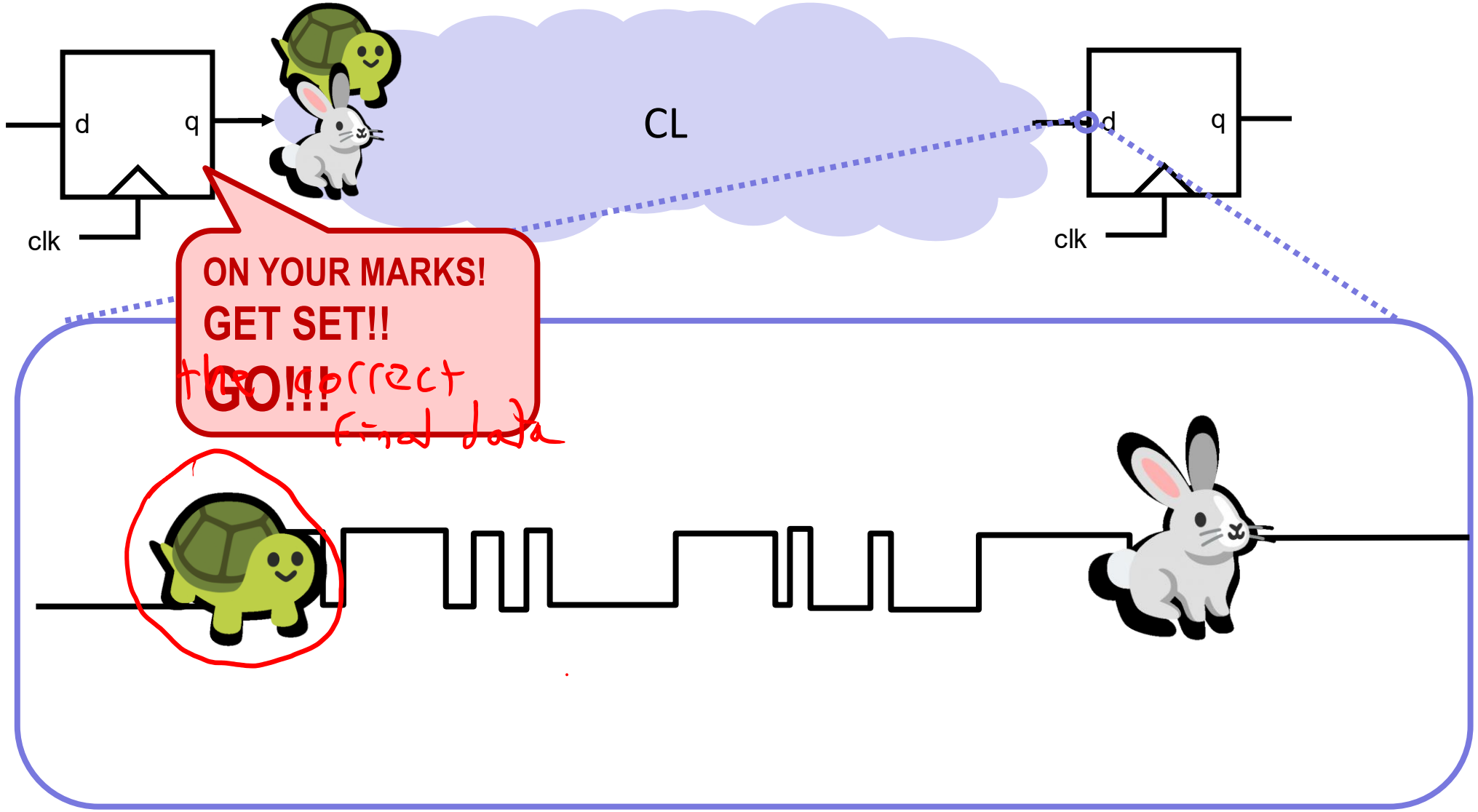



# Model for Synchronous Digital Systems

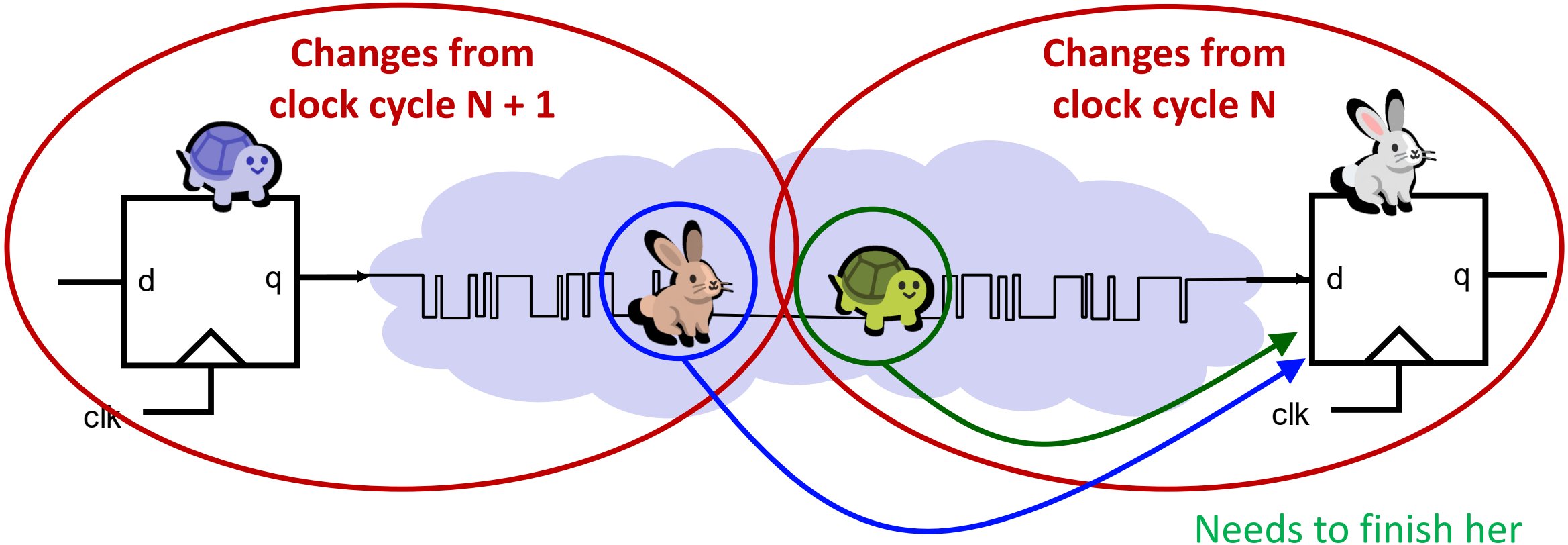
- Combinational logic blocks separated by registers
  - Clock signal connects only to sequential logic elements
  - Feedback is optional depending on application
- How do we know if things are going to break?!



# Minimum and Maximum Delay



# Race around the clock

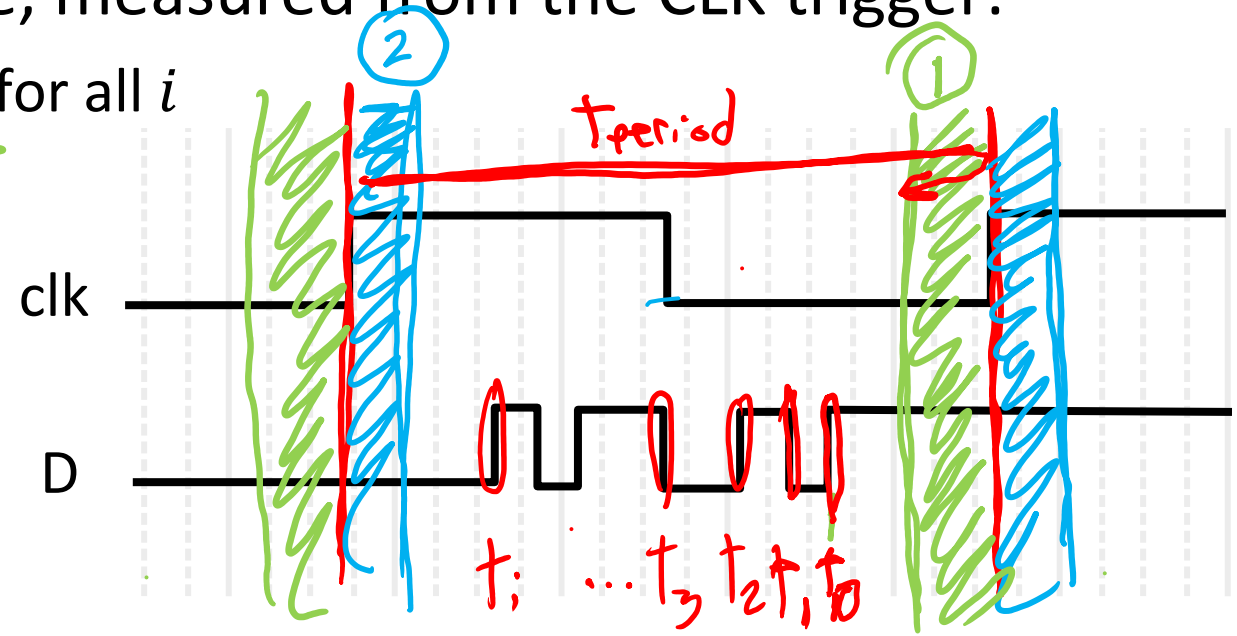



Needs to finish her race **after hold time** for cycle N

Needs to finish her race **before setup time** for cycle N

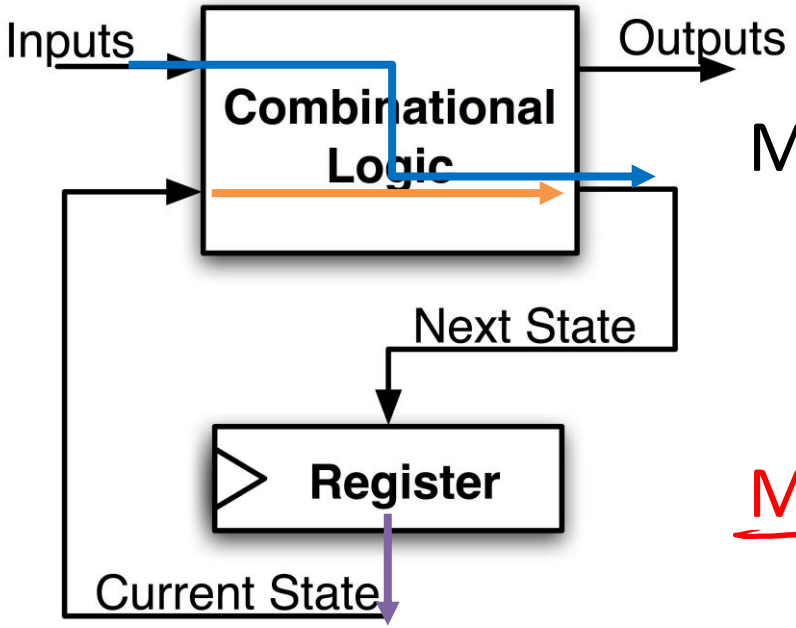
# Timing equations

- Within a given clock period ($t_{period}$), the short paths 🐰 shouldn't violate hold time ($t_{hold}$) and the long paths 🐢 shouldn't violate setup time ($t_{setup}$)
- Let $t_{input,i}$ be the time it takes for the input of a register to change for the $i^{th}$ time in a single clock cycle, measured from the CLK trigger:
  - $t_{hold} \leq t_{input,i} \leq t_{period} - t_{setup}$ for all $i$
  - Two separate constraints!



# Minimum Delay

- If shortest path to register input is too short, might violate hold time constraint
  - Input could change before state is "locked in"
  - Particularly problematic with *asynchronous* signals



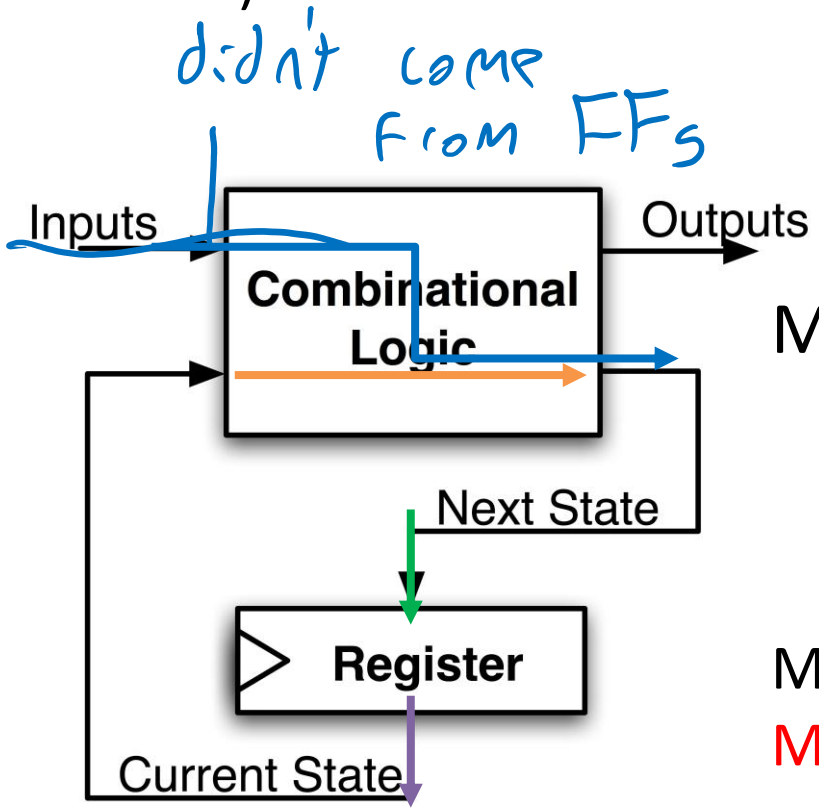

"worst case"

Min Delay = min( CLK-to-Q Delay + Min CL Delay, Min CL Delay)

Min Delay ≥ Hold Time

# Maximum Clock Frequency

- What is the max frequency of this circuit?
  - Limited by how much time needed to get correct Next State to Register ($t_{setup}$ constraint)



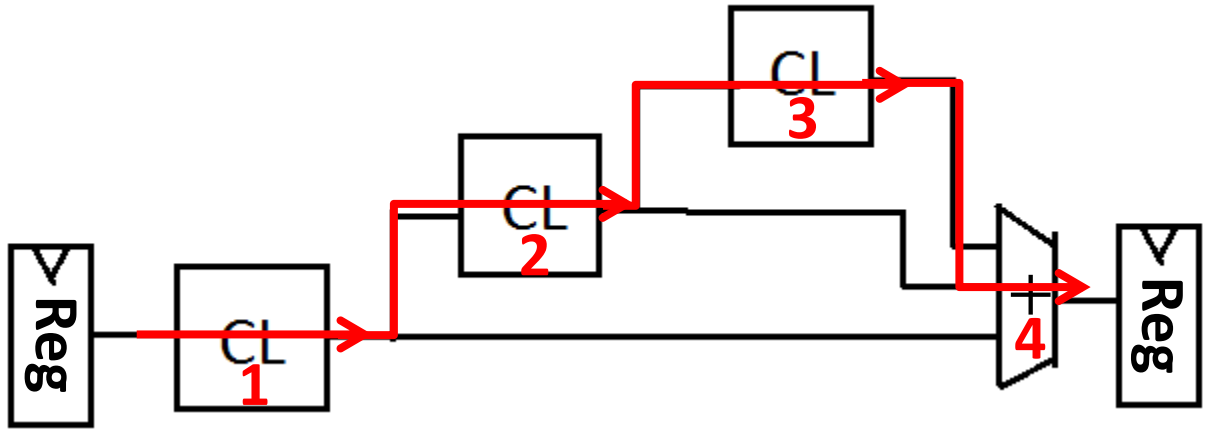

Max Delay= max(CLK-to-Q Delay + Max CL Delay, Max CL Delay)

Min Period = Max Delay + Setup Time

Max Freq = 1/Min Period

# The Critical Path

- The *critical path* is the longest delay between *any* two registers in a circuit
- The clock period must be *longer* than this critical path, or the signal will not propagate properly to that next register

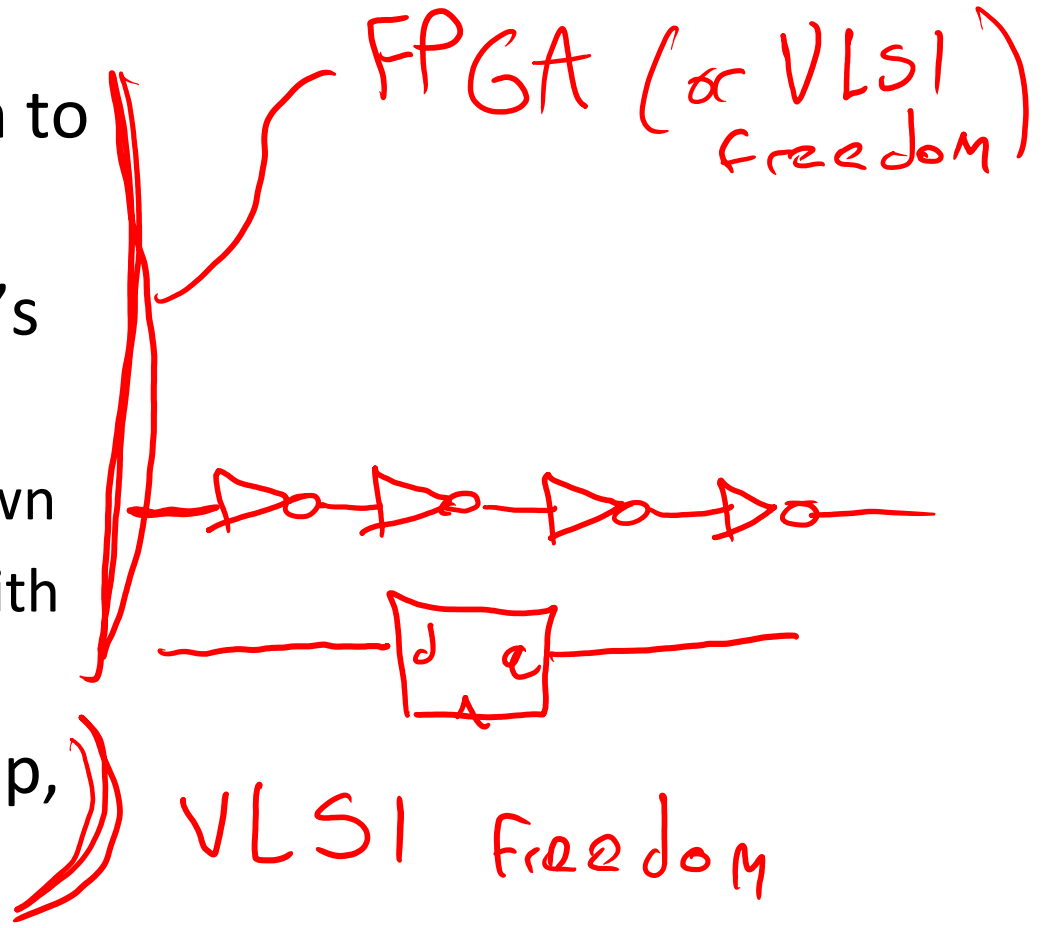



**Critical Path =**

CLK-to-Q Delay
+ CL Delay 1
+ CL Delay 2
+ CL Delay 3
+ Adder Delay
+ Setup Time

# Well, what can we do then?

- **Choose** a **clock period** that's long enough to allow for the critical path
- **Design** our **combinational logic** so that it's neither too fast nor too slow
  - Too fast: put **delay elements** in to slow it down
  - Too slow: **split up** the logic into **two stages** with a **register in the middle**
- **Design** our **flip-flops** to have smaller setup, hold and clock-to-Q times



# 1 Practice Question

$t_{hold} \le t_{CLmin}$

$t_{CLmax} \le t_{period} - t_{setup}$

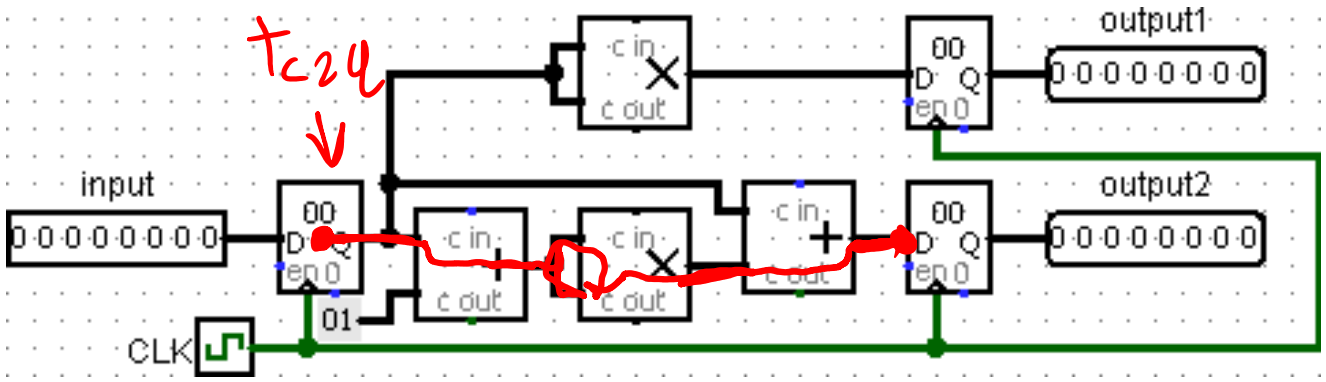
$\frac{1}{1\,GHz} = \frac{1}{1\times10^9} = 1000\,ps$

We want to run our CPU at 1 GHz. Assume:

$t_{add}$ = 100 ps, $t_{mult}$ = 200 ps, $t_{setup}$ = $t_{hold}$ = 50 ps.

What is the maximum $t_{clk\text{-}to\text{-}q}$ we can allow?

$t_{CLMax} \le 950$




$t_{c2q} + t_{add} + t_{mult} + t_{add} \le 950$

$t_{c2q} + 400 \le 950$

$t_{c2q} \le 550\,ps$

(A) 550 ps (B) 750 ps

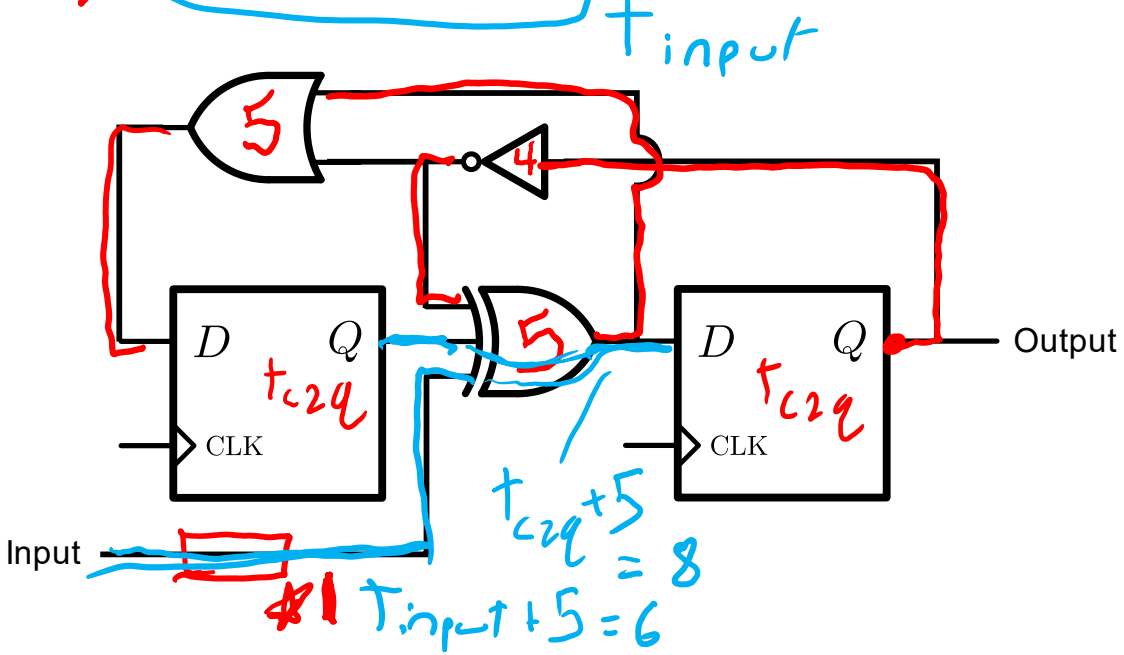
(C) 500 ps (D) 700 ps

# 2 Another practice question

$t_{hold} \leq t_{CLmin}$

$t_{CLmax} \leq t_{period} - t_{setup}$ (20 − 2)

- The circuit below has the following timing parameters
  - $t_{period} = 20\ ns, t_{setup} = 2\ ns$
  - $t_{XOR} = t_{OR} = 5\ ns, t_{NOT} = 4\ ns$
  - Input changes 1 ns after clock trigger




1. What is the max allowable $t_{C2Q}$?

$t_{max} \leq 18$

$t_{c2q} + t_{not} + t_{xor} + t_{or} \leq 18$

$t_{c2q} + 14 \leq 18 \Rightarrow$ **$t_{c2q} \leq 4$**

2. If $t_{C2Q} = 3\ ns$, what is the max allowable $t_{hold}$?

$t_{hold} \leq t_{input} + t_{xor}$

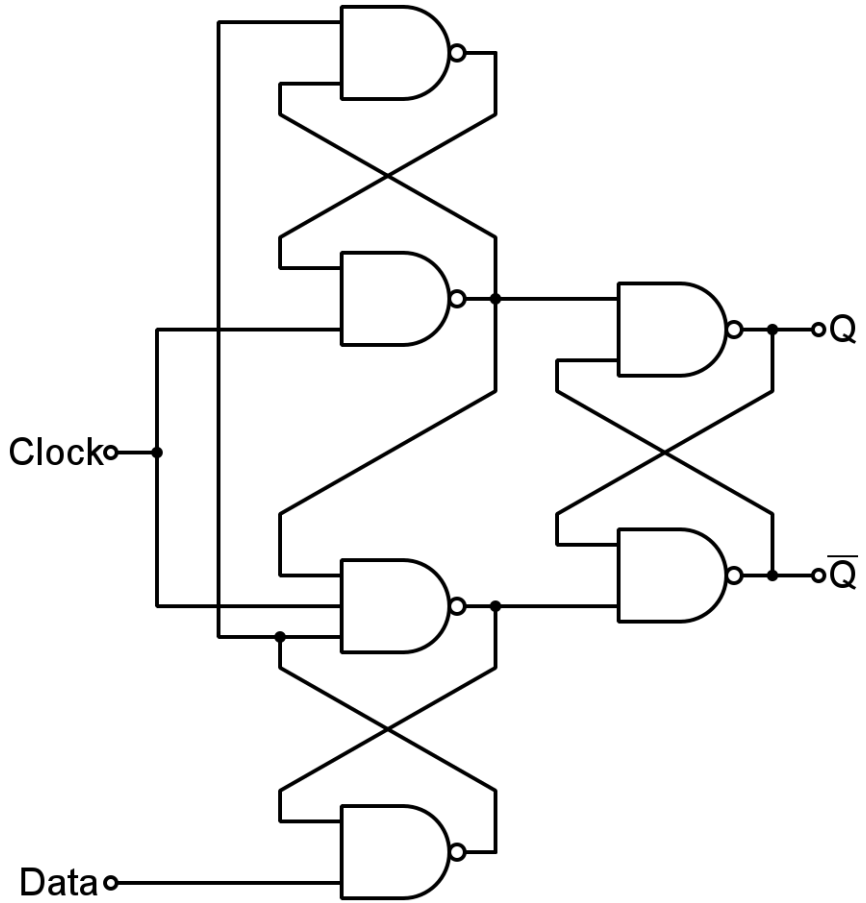
$\leq 1 + 5$

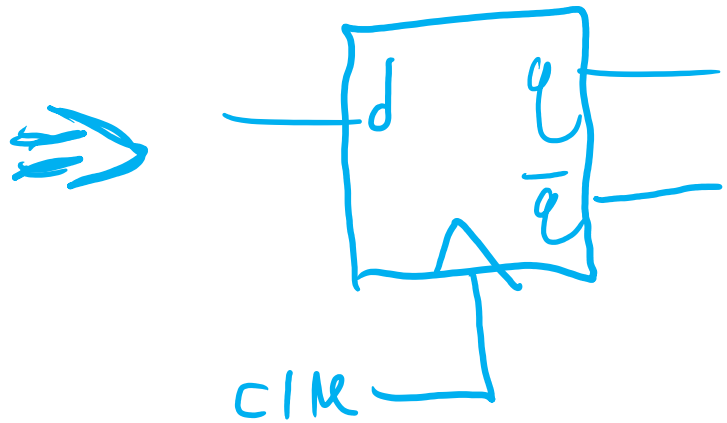
**$t_{hold} \leq 6$**

# Miso Moment

# Where Do Timing Constraints Come From?

Edge-triggered
D flip-flop:

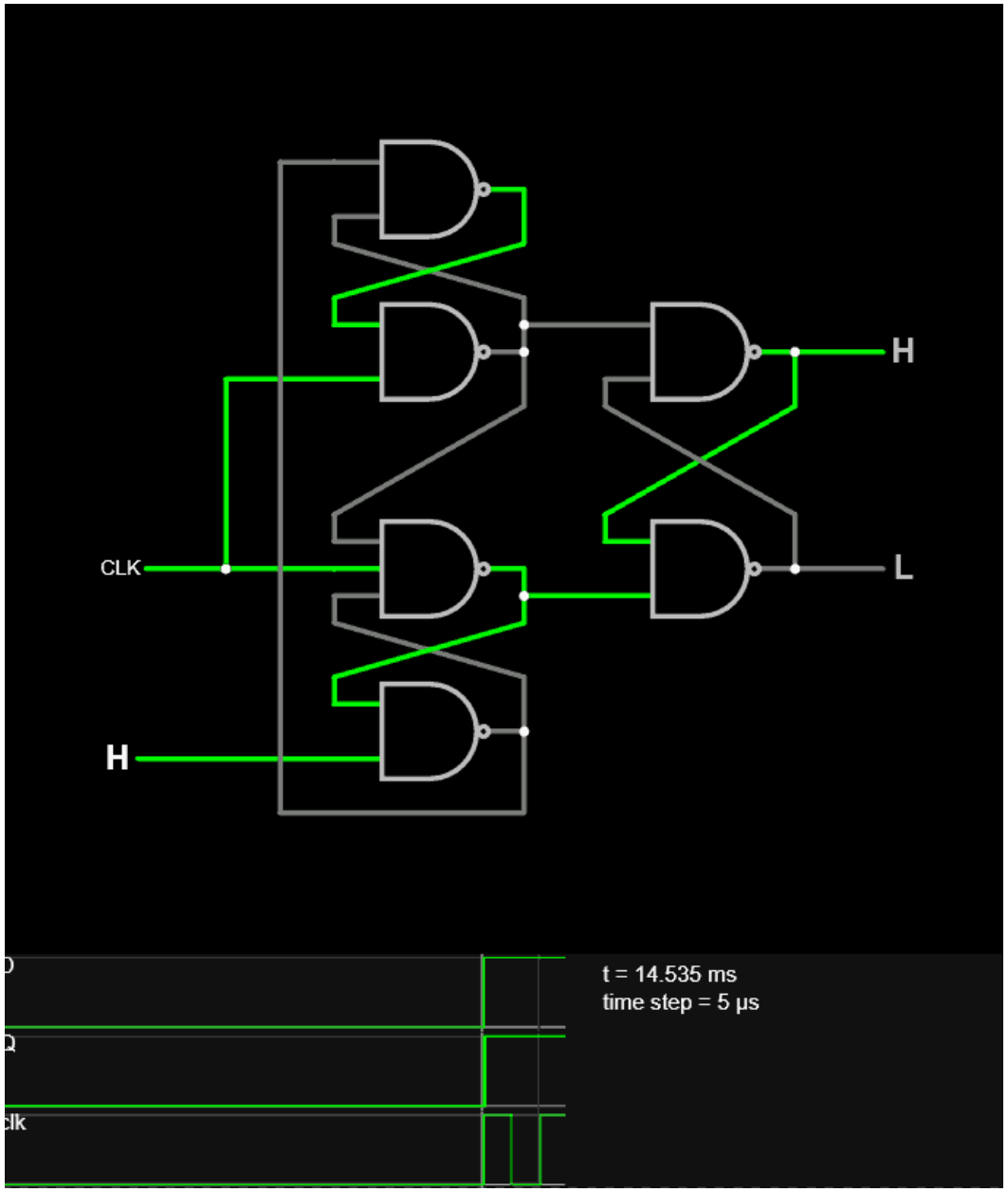



By Nolanjshettle at English Wikipedia, CC BY-SA 3.0,
https://commons.wikimedia.org/w/index.php?curid=40852354

# DFF feedback loop simulator

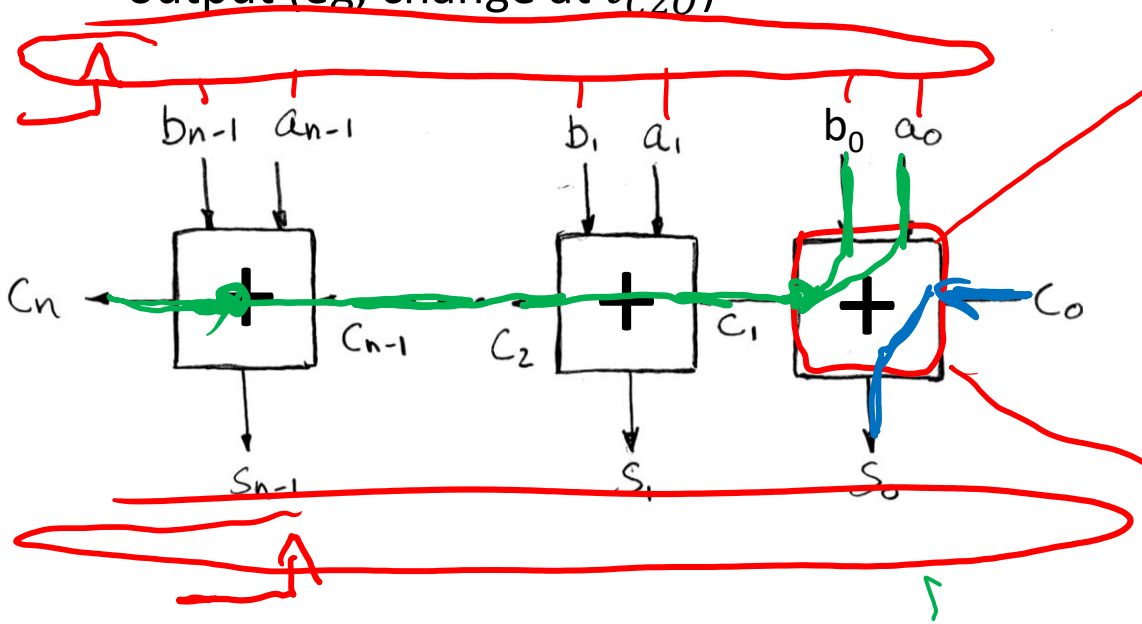
- https://www.falstad.com/circuit/e-edgedff.html
- This is ***way*** outside the scope of this class. Only worry about this if you're curious




clk →

# 3 More timing Practice

- For an $n$-bit ripple-carry adder, what is the **earliest** and **latest** time that the output will change after clock edge?
  - Assume gate delay of 1 ns
  - Assume A, B, $c_0$ straight from register output (eg, change at $t_{C2Q}$)



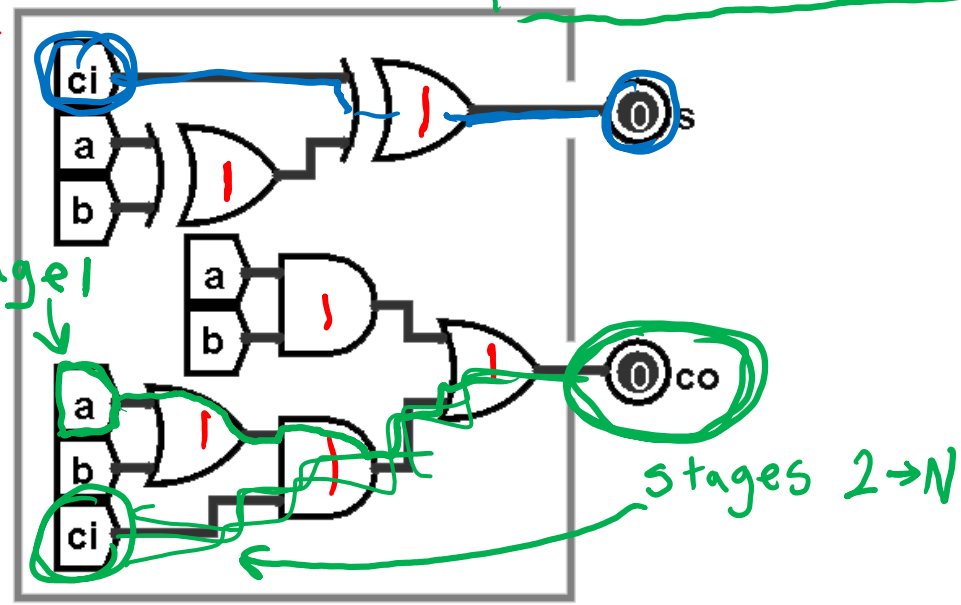

$t_{early} = t_{c2q} + t_{xor}$

$= t_{c2q} + 1$ ← Not the "right" answer for that bit just the earliest change

$t_{late} = t_{c2q} + t_{OR} + t_{and} + t_{OR} + (N-1)(t_{and} + t_{OR})$

stage 1 | stages 2→N

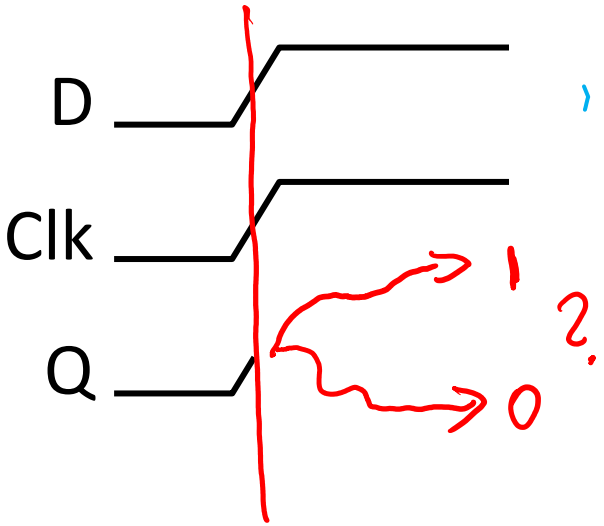
$= t_{c2q} + 3 + (N-1)(2)$



# Flip-Flop Realities: External Inputs

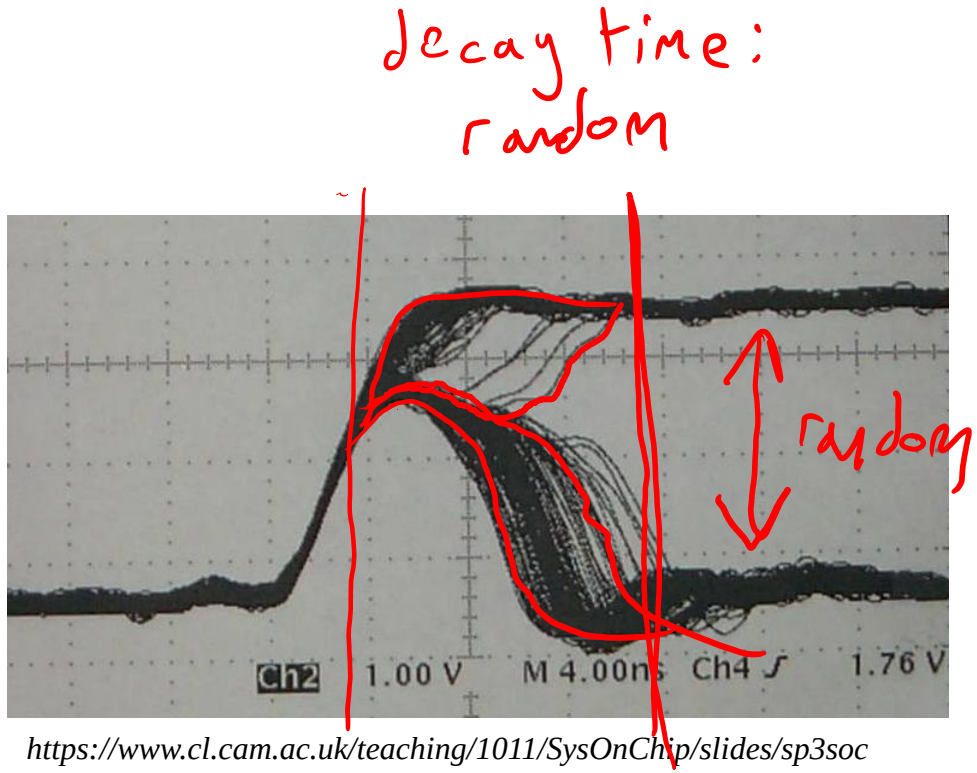
- External inputs aren't synchronized to the clock
  - If not careful, can violate timing constraints

- What happens if input changes around clock trigger?



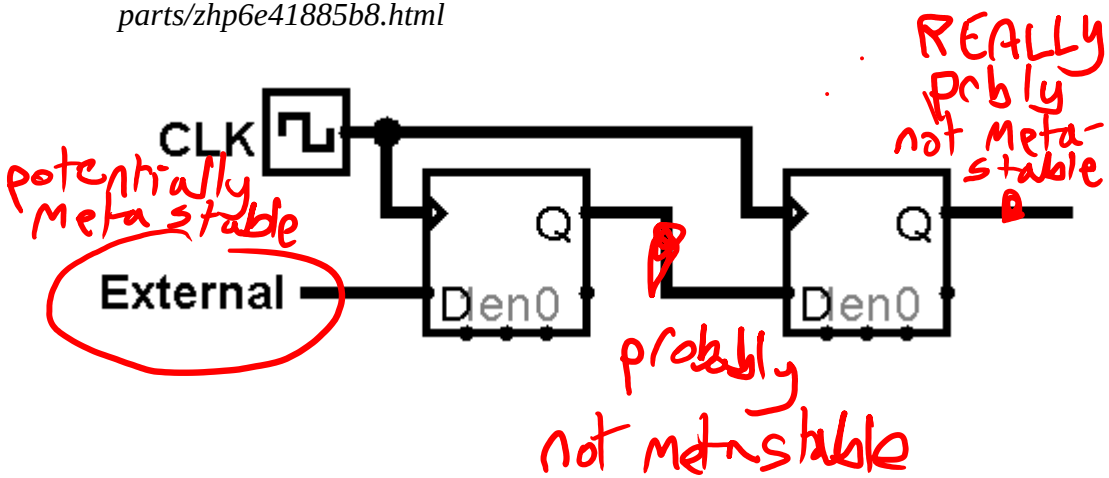
# Flip-Flop Realities: Metastability

- **Metastability** occurs when a digital feedback loop settles into an **unstable equilibrium** storing a **non-binary** voltage
  - Can last for a potentially unbounded amount of time
  - Will randomly decay to a '0' or a '1'….probably



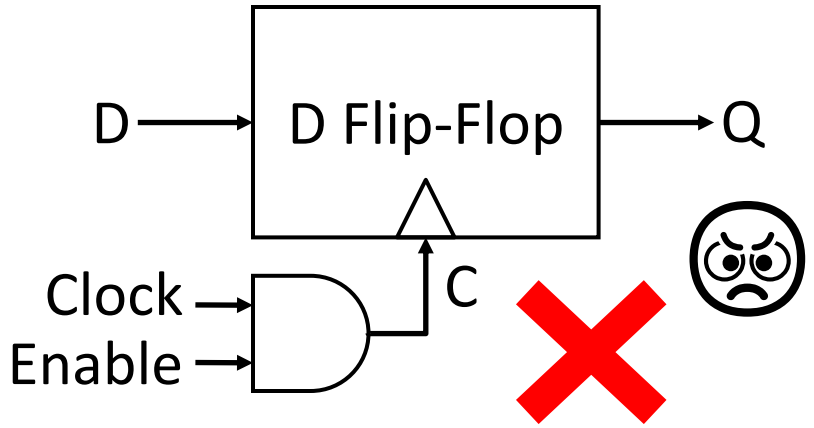

*https://www.cl.cam.ac.uk/teaching/1011/SysOnChip/slides/sp3socparts/zhp6e41885b8.html*

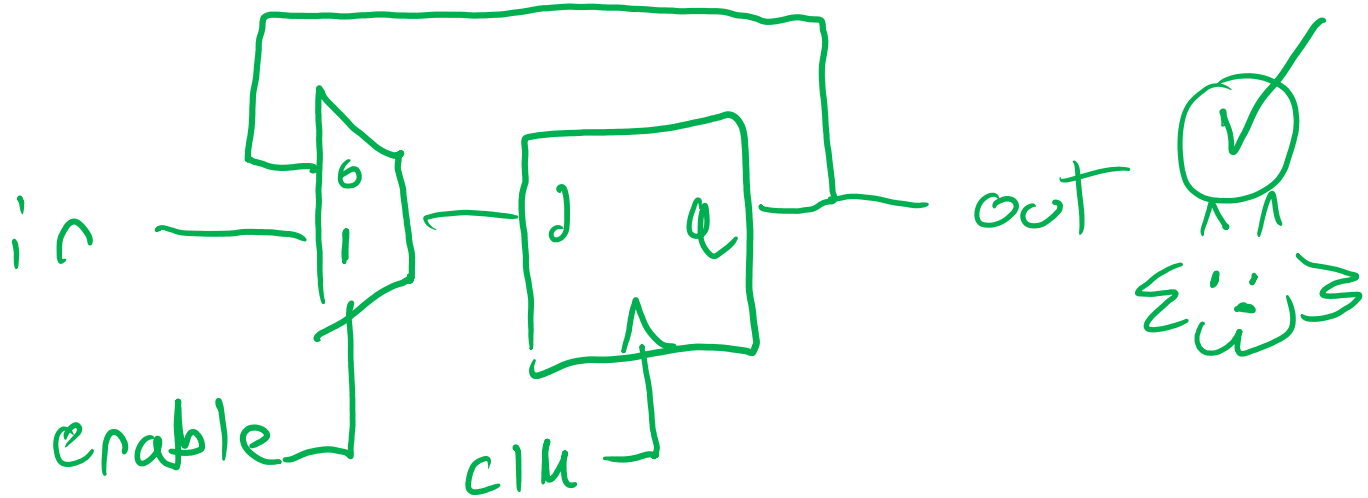
- Flip flops help reject transients
  - Longer chains = higher chance of filtering out all metastabilitiy, but longer signal delay



# Flip-Flop Realities: NEVER GATE THE CLOCK!!!

- Don't do it
- Introduces unnecessary **clock skew** into design
- Makes timing equations ***exceptionally*** hard to model



# Summary of Timing Terms

- Clock: steady square wave that synchronizes system

- Flip-flop: one bit of state that samples every rising edge of CLK (positive edge-triggered)
- Register: several bits of state that samples on rising edge of CLK (positive edge-triggered); often has a RESET

- Setup Time: when input must be stable *before* CLK trigger
- Hold Time: when input must be stable *after* CLK trigger
- CLK-to-Q Delay: how long it takes output to change from CLK trigger

# Extra: Unrelated to SDS, just more FSM practice

- Recognize the string `101` with the following behavior
  - Input: `1 0 0 1 0 1 0 1 1 0 0 1 0`
  - Output: `0 0 0 0 0 1 0 1 0 0 0 0 0`
- State diagram to implementation: