

# Sequential Circuits

## Circuits with state

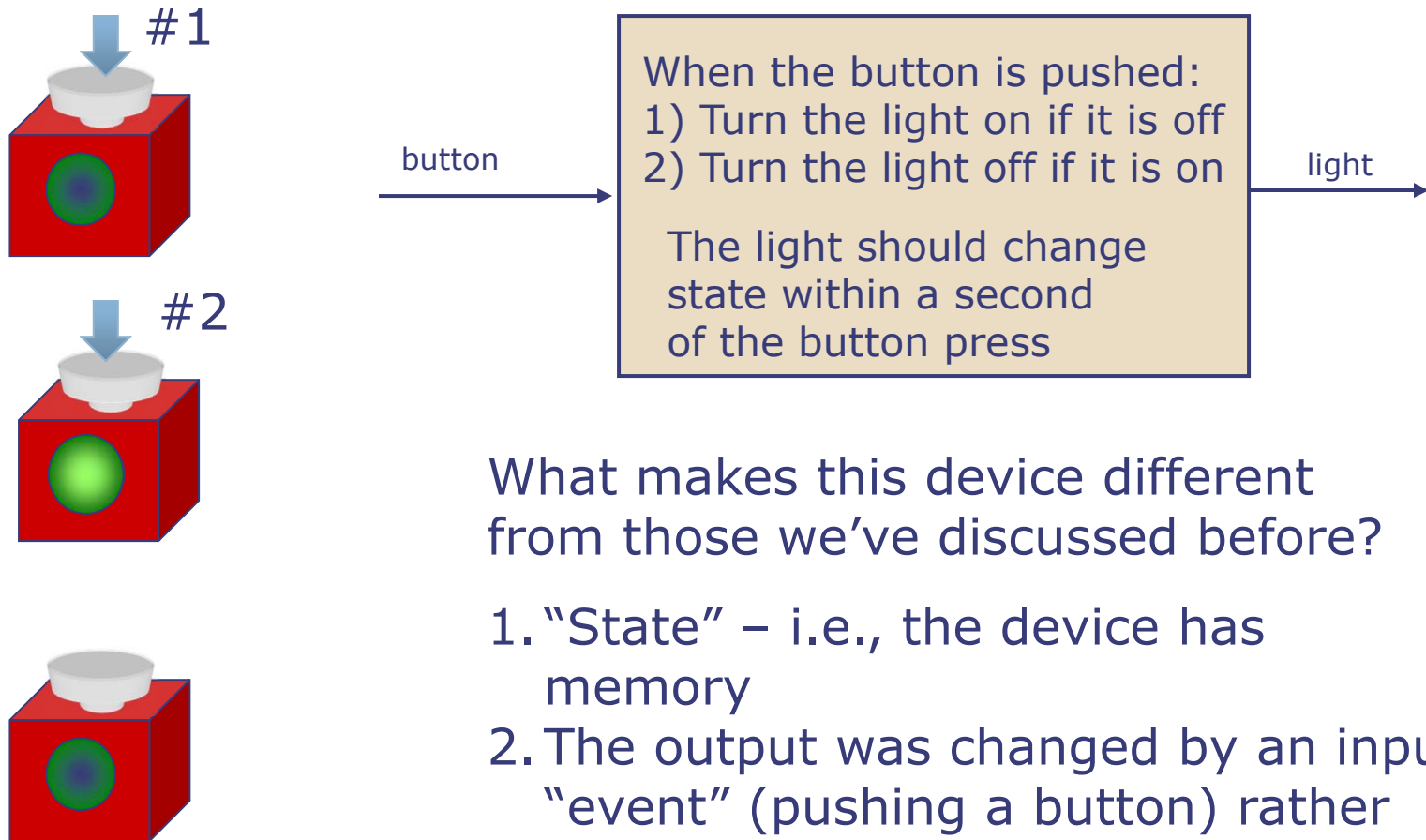
### Reminders:

- Quizzes returned in recitation tomorrow or pickup from Sally (room 32-G846) starting **Thursday**
- Lab 3 due Thursday

# Something We Can't Build (Yet)

---

What if you were given the following design specification:

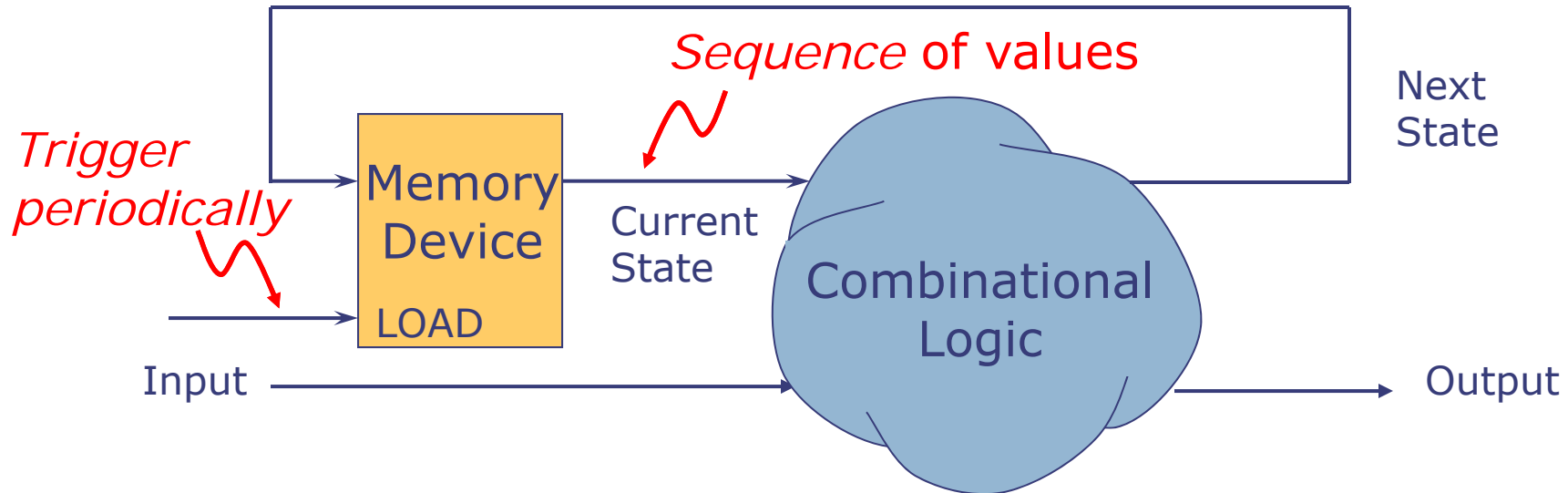


What makes this device different from those we've discussed before?

1. "State" – i.e., the device has memory
2. The output was changed by an input "event" (pushing a button) rather than an input "level"

# Digital State: What We'd Like to Build

---



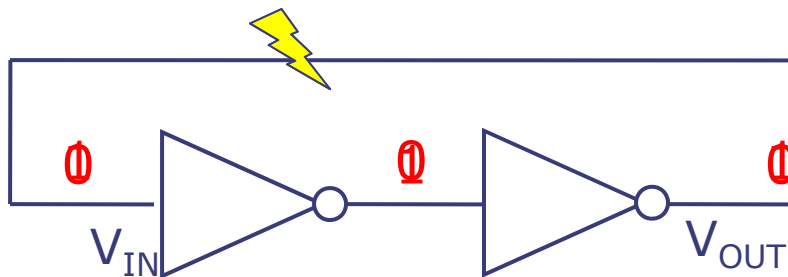
Plan: Build a Sequential Circuit with stored digital STATE –

- Memory stores CURRENT state
- Combinational Logic computes
  - NEXT state (from input, current state)
  - OUTPUT bits (from input, current state)
- State changes on LOAD control input

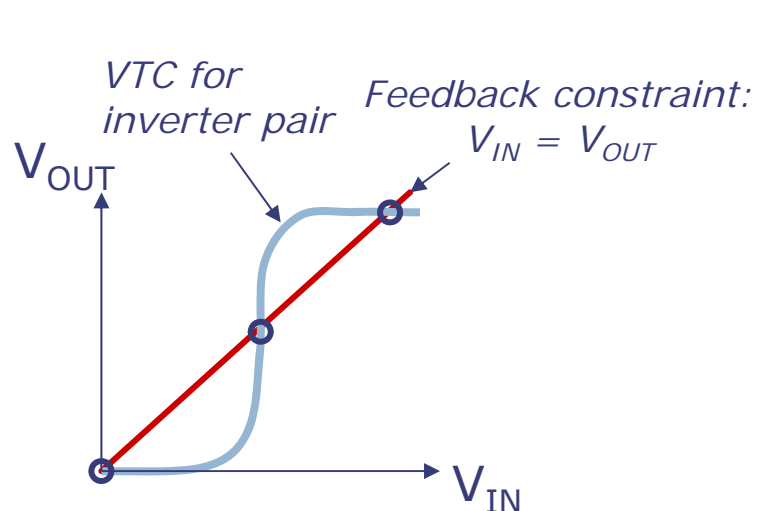
Needed:  
Loadable  
Memory

# Memory: Using Feedback

IDEA: use **feedback** to maintain storage indefinitely. Our logic gates are built to restore marginal signal levels, so noise shouldn't be a problem!



Result: a **bistable** storage element



Not affected by noise

Three solutions:

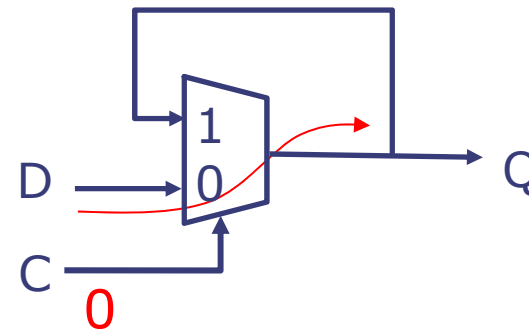
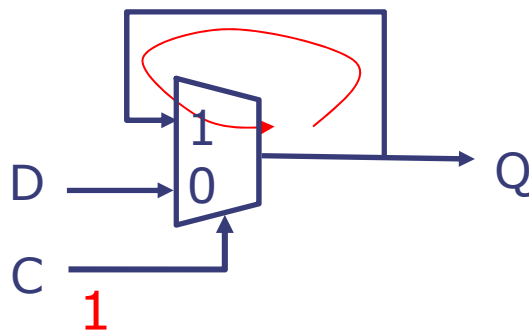
- two end-points are **stable**
- middle point is metastable

We'll get back to this!

# D Latch

A famous circuit that can hold state

---



if  $C=1$ , the value of  $Q$  *holds*

if  $C=0$ , the value of  $D$  *passes* to  $Q$

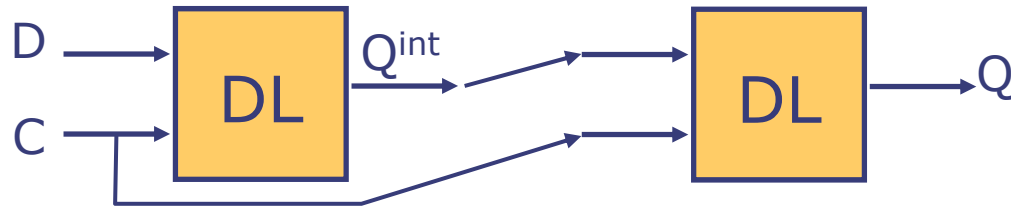


Let  $Q^{t-1}$  represent the value previously held in DL;  
 $Q^t$  represents the current value.

C	D	$Q^{t-1}$	$Q^t$	
0	0	X	0	} pass
0	1	X	1	
1	X	0	0	} hold
1	X	1	1	

# Building circuits with D latches

---



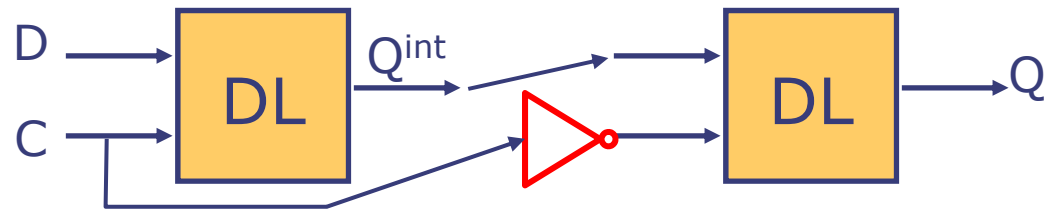
- If two latches are driven by the same C signal, they *pass* signals at the same time and *hold* signals at the same time.
- The composed latches look just like a single D latch (assuming signals aren't changing too fast)



# Building circuits with D latches

*continued*

---

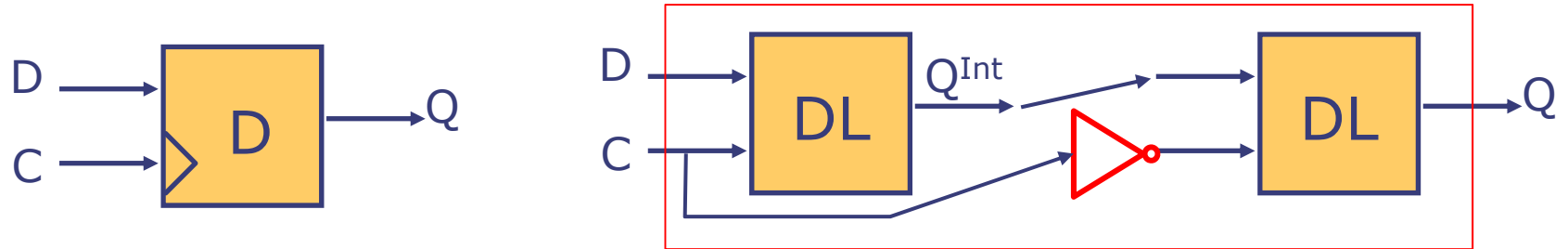


- If latches are driven by inverted C signals, one is always holding, and one is always passing
- How does this circuit behave?
  - $C = 0$ :  $Q^{\text{int}}$  follows the input D, but Q holds its old value
  - $C = 1$ :  $Q^{\text{int}}$  holds its old value, but Q follows  $Q^{\text{int}}$
  - Q doesn't change when  $C=0$  or  $C=1$ , it changes when **C transitions from 0 to 1 (a *rising-edge* of C)**

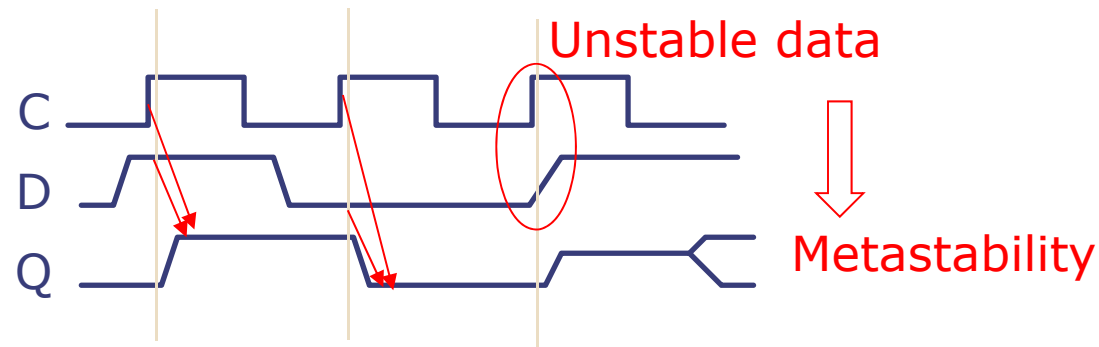
*What happens on a falling edge (C: 1  $\rightarrow$  0)?*

# Edge-Triggered D Flip flop

A basic storage element



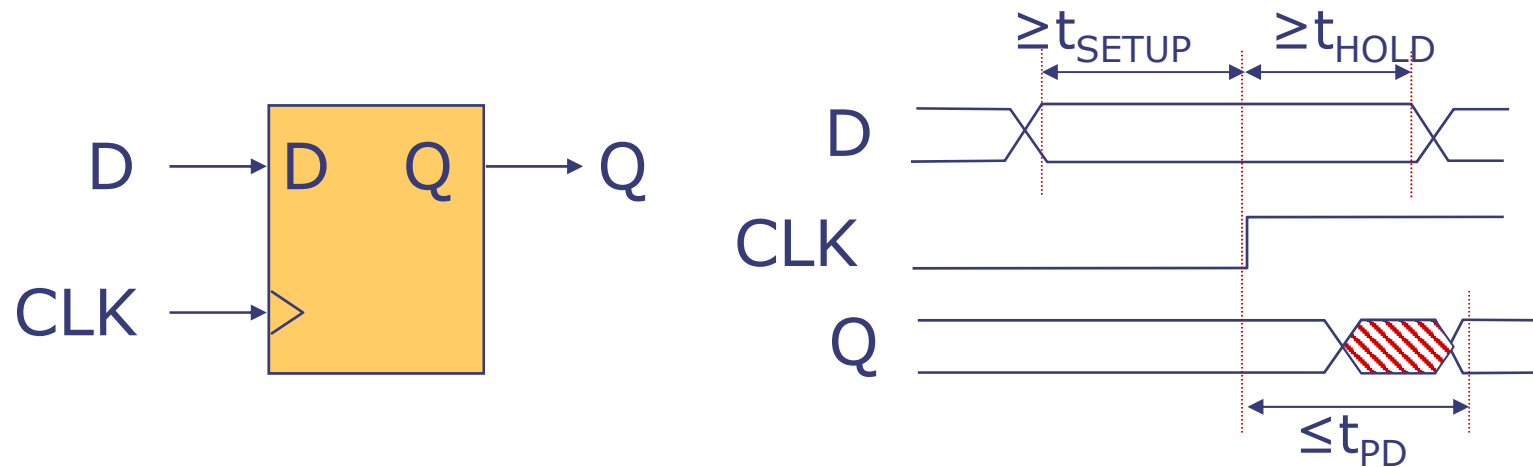
Suppose C changes periodically (called a *Clock* signal)



*Data is sampled at the rising edge of the clock and must be stable at that time*

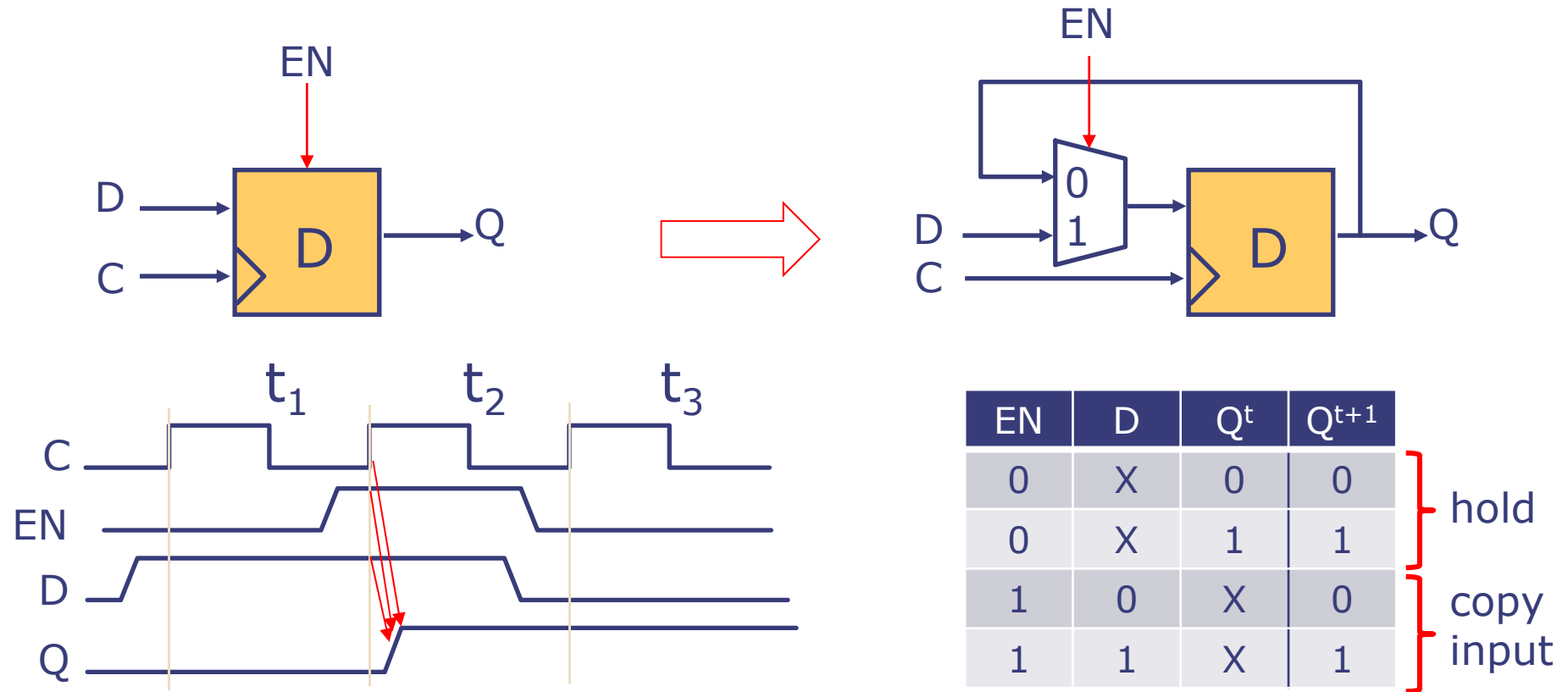


# D Flip-Flop Timing



- Flip-flop input D should not change around the rising edge of the clock to avoid *metastability*
- Formally, D should be a stable and valid digital value:
  - For at least  $t_{\text{SETUP}}$  before the rising edge of the clock
  - For at least  $t_{\text{HOLD}}$  after the rising edge of the clock
- Flip-flop propagation delay  $t_{\text{PD}}$  is measured from rising edge of the clock to valid output (CLK $\rightarrow$ Q)

# D Flip-Flop with Write Enable

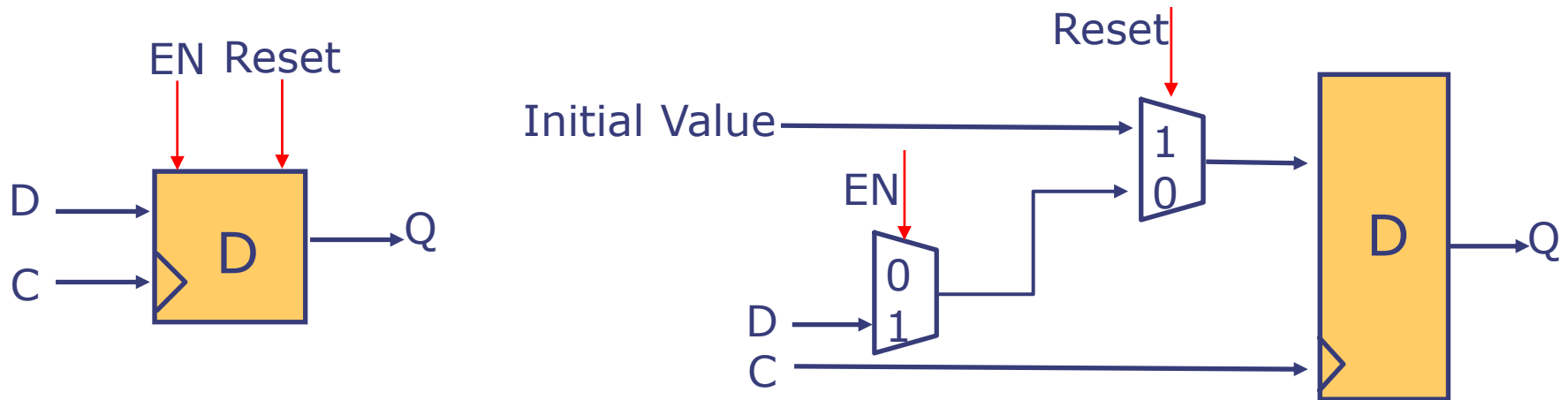


Data is captured only if  
EN is on

Transitions happen at rising  
edge of the clock  
No need to specify the  
clock explicitly

# How Are Flip-Flops Initialized?

---

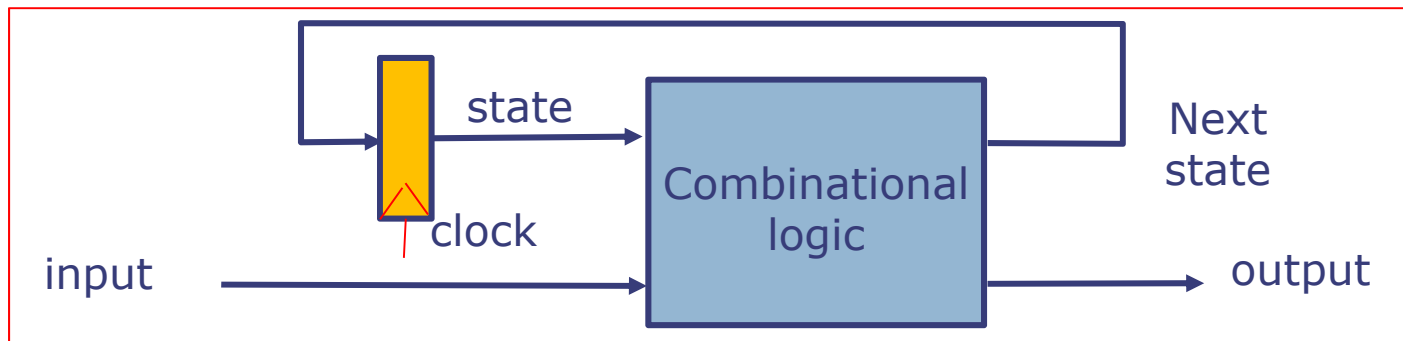


- When  $\text{Reset} = 1$ , flip flop is set to initial value regardless of value of  $\text{EN}$
- When  $\text{Reset} = 0$ , then it behaves like a D flip-flop with enable

# Synchronous Sequential Circuits

---

- State is maintained in registers that all share the same periodic clock signal.
- Registers update their contents simultaneously, at the rising edge of the clock.
- This allows discretizing time into cycles and abstracting sequential circuits as **finite state machines (FSMs)**.



# A Simple Sequential Circuit...

---

Lets make a digital binary *Combination Lock*:

Specification:

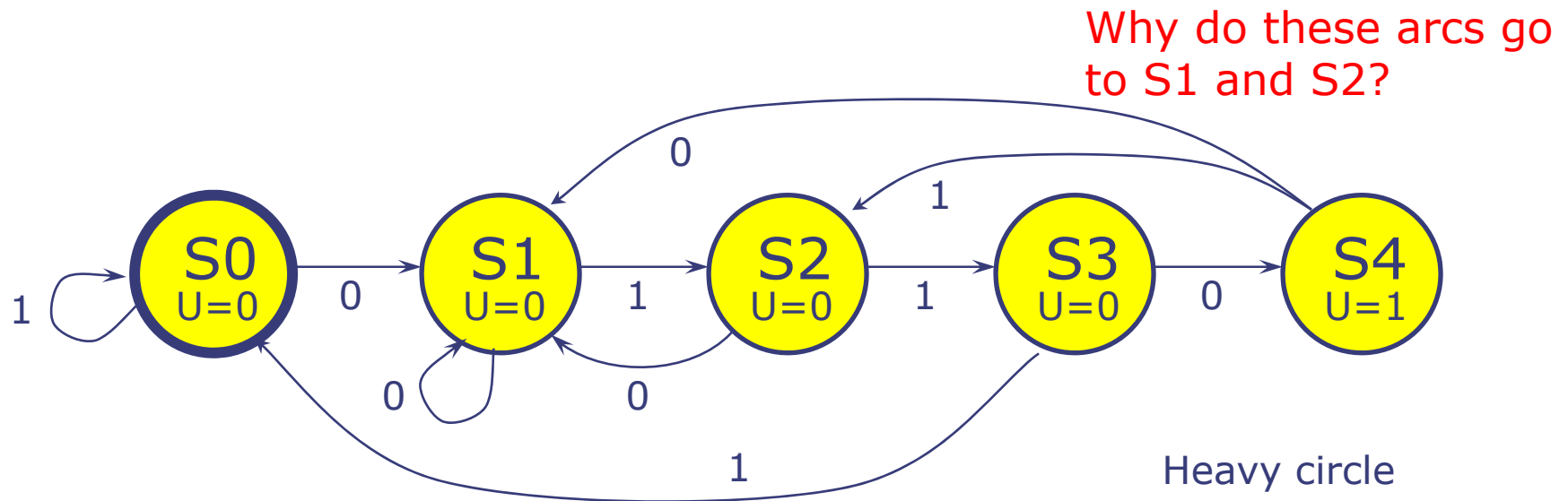


- One input ( “0” or “1” )
- One output ( “Unlock” signal )
- UNLOCK is 1 if and only if:

How many states  
do we need?

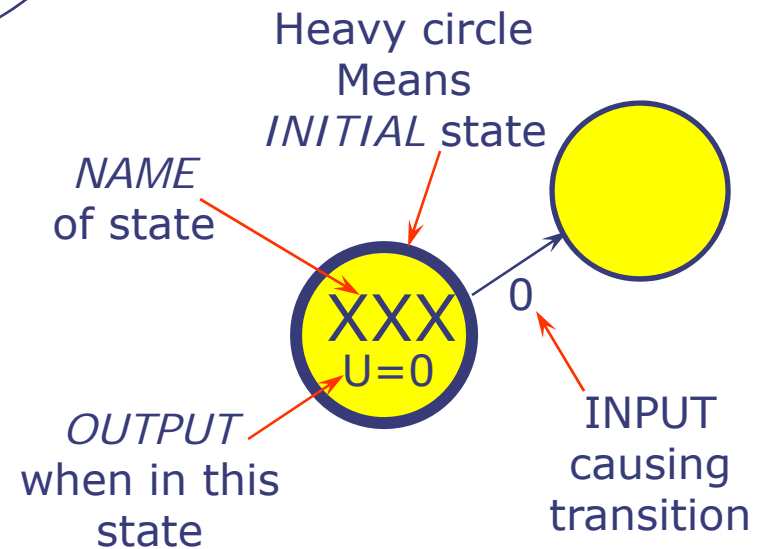
Last 4 inputs were the  
“combination”: 0110

# State Transition Diagram



## Designing our lock ...

- Need an initial state; call it S0.
- Must have a separate state for each step of the proper entry sequence
- Must handle other (erroneous) entries



# Valid State Diagrams

---

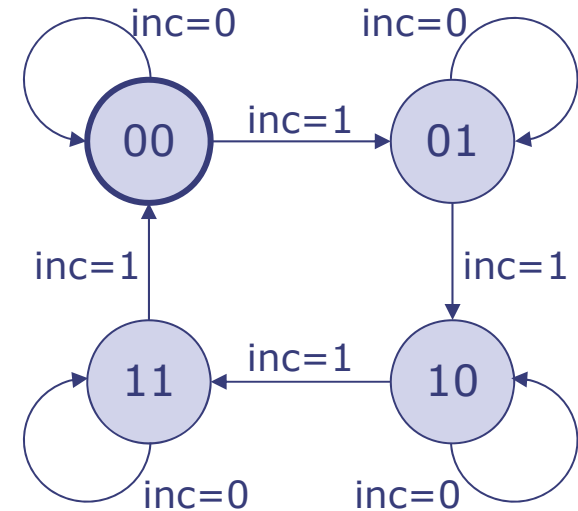
- Arcs leaving a state must be:
  - (1) **mutually exclusive**
    - *can't have two choices for a given input value*
  - (2) **collectively exhaustive**
    - *every state must specify what happens for each possible input combination. "Nothing happens" means arc back to itself.*

# Modulo-4 Counter

## An Example

---

Prev State	NextState	
q1q0	inc = 0	inc = 1
00	00	01
01	01	10
10	10	11
11	11	00



State Transition Diagram

$$q0^{t+1} = \sim inc \cdot q0^t + inc \cdot \sim q0^t$$

$$= inc \oplus q0^t$$

$$q1^{t+1} = \sim inc \cdot q1^t + inc \cdot \sim q1^t \cdot q0^t + inc \cdot q1^t \cdot \sim q0^t$$

$$= \sim inc \cdot q1^t + inc \cdot (q1^t \oplus q0^t)$$

$$\text{Output} = \text{current state} = q1q0$$



# Modulo-4 Counter Circuit

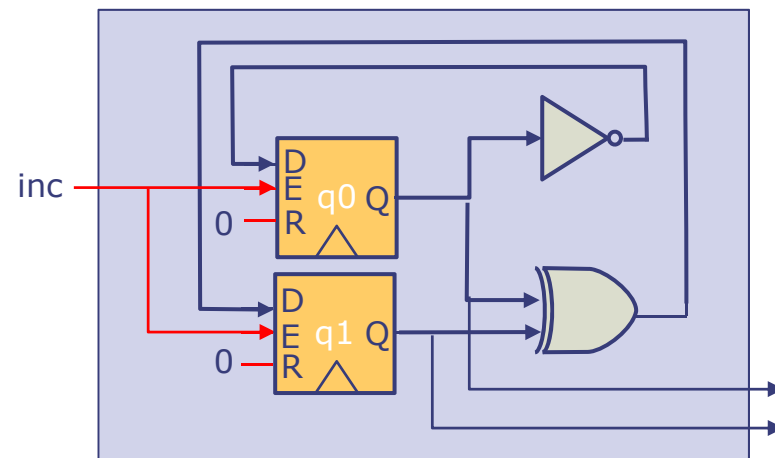
## Using D flip-flops

$$q0^{t+1} = \sim inc \cdot q0^t + inc \cdot \sim q0^t$$

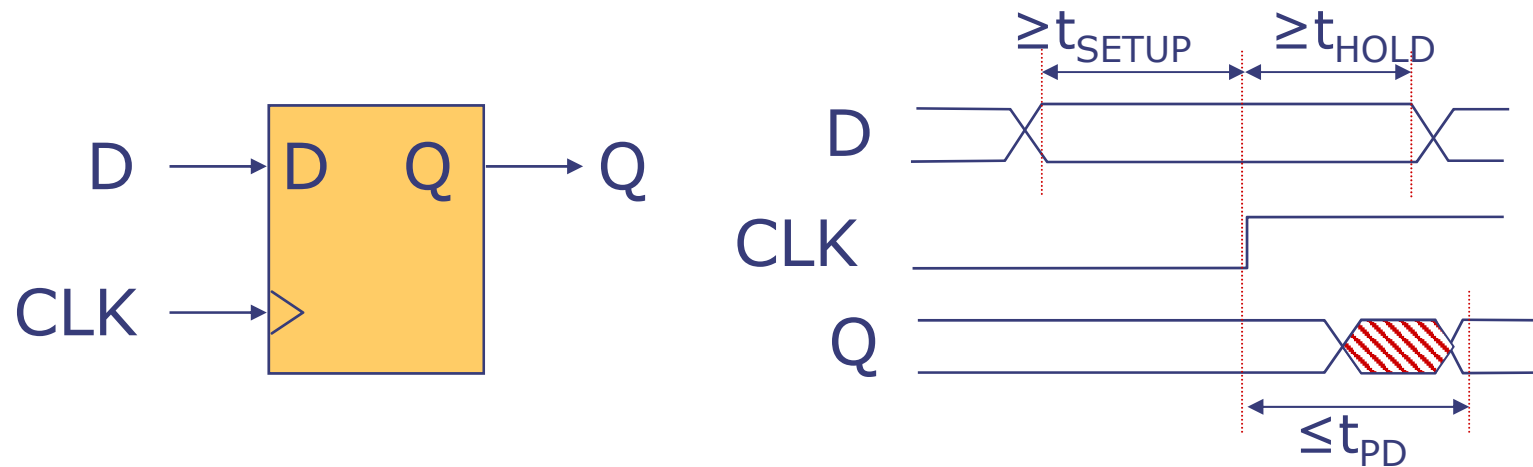
$$q1^{t+1} = \sim inc \cdot q1^t + inc \cdot (q1^t \oplus q0^t)$$

- Use two D flip-flops with *reset* and *enable* to store  $q0$  and  $q1$
- Notice, the state only changes when  $inc$  is 1. Let's connect  $inc$  to *enable* and simplify the equations
- Output is just  $q1q0$
- Reset both flip-flops to 0 at startup to start in state 00

$$\left. \begin{aligned} q0^{t+1} &= \sim q0^t \\ q1^{t+1} &= q1^t \oplus q0^t \end{aligned} \right\} inc=1$$



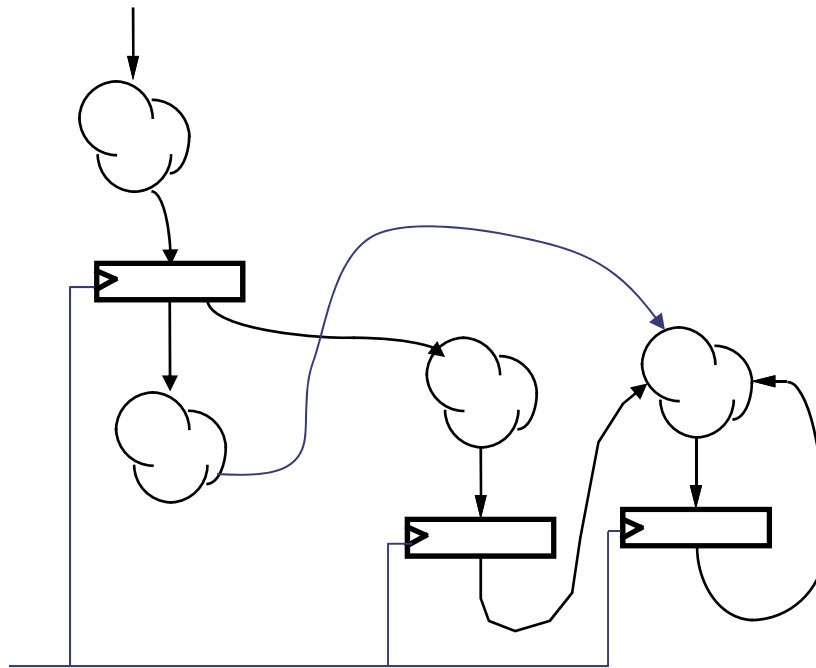
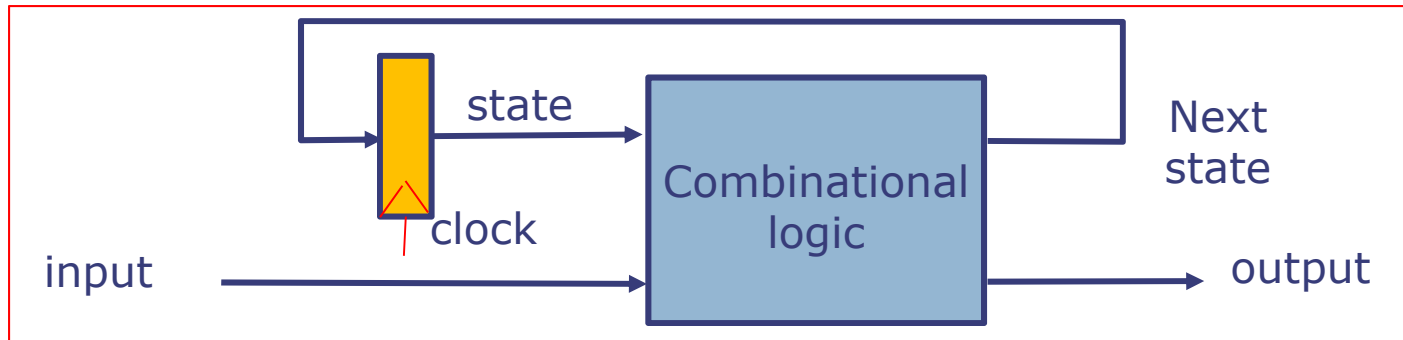
# Recap: D Flip-Flop Timing



- Flip-flop input D should not change around the rising edge of the clock to avoid *metastability*
- Formally, D should be a stable and valid digital value:
  - For at least  $t_{\text{SETUP}}$  before the rising edge of the clock
  - For at least  $t_{\text{HOLD}}$  after the rising edge of the clock
- Flip-flop propagation delay  $t_{\text{PD}}$  is measured from rising edge of the clock to valid output (CLK $\rightarrow$ Q)

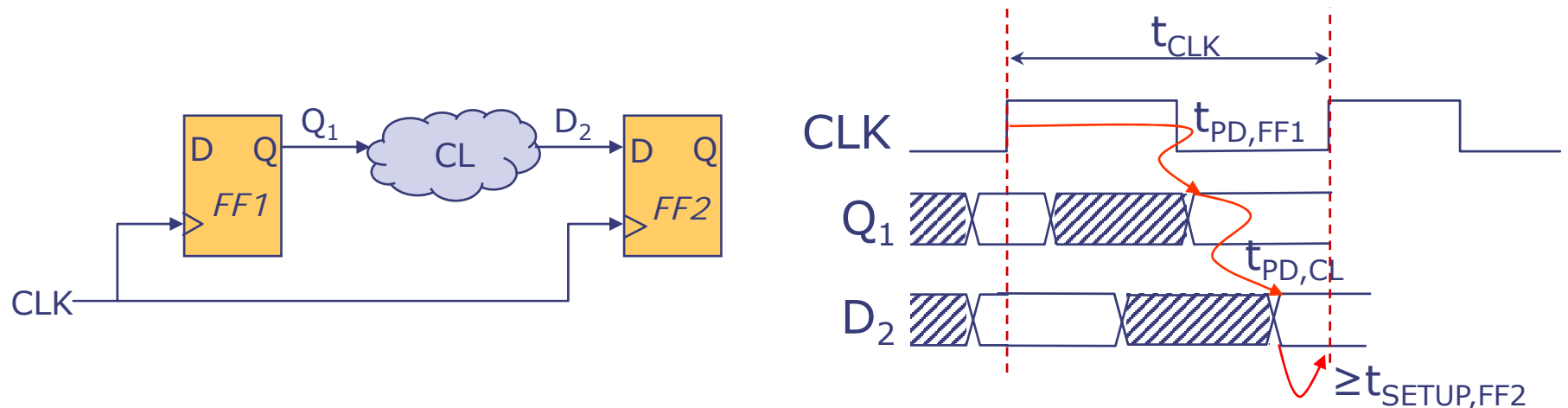
# Single-clock Synchronous Circuits

---



Need to analyze  
timing of each  
register to register  
path

# Meeting the Setup-Time Constraint

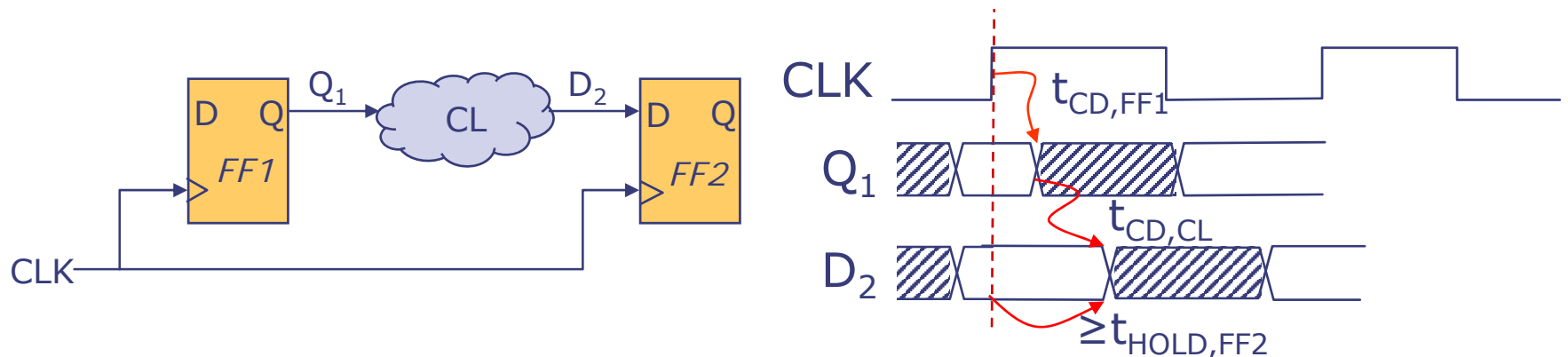


- To meet FF2's setup time,

$$t_{CLK} \geq t_{PD,FF1} + t_{PD,CL} + t_{SETUP,FF2}$$

- The slowest register-to-register path in the system determines the clock; Equivalently, a given register technology and clock limit the amount of combinational logic between registers

# Meeting the Hold-Time Constraint



- Hold time ( $t_{\text{HOLD}}$ ) constraint of FF2 may be violated if  $D_2$  changes too quickly
- Propagation delay ( $t_{\text{PD}}$ ), the upper bound on time from valid inputs to valid outputs, does not help us analyze hold time!
- *Contamination delay* ( $t_{\text{CD}}$ ) is the lower bound on time from input transition to output transition
- To meet FF2's hold-time constraint

$$t_{\text{CD},\text{FF1}} + t_{\text{CD},\text{CL}} \geq t_{\text{HOLD},\text{FF2}}$$

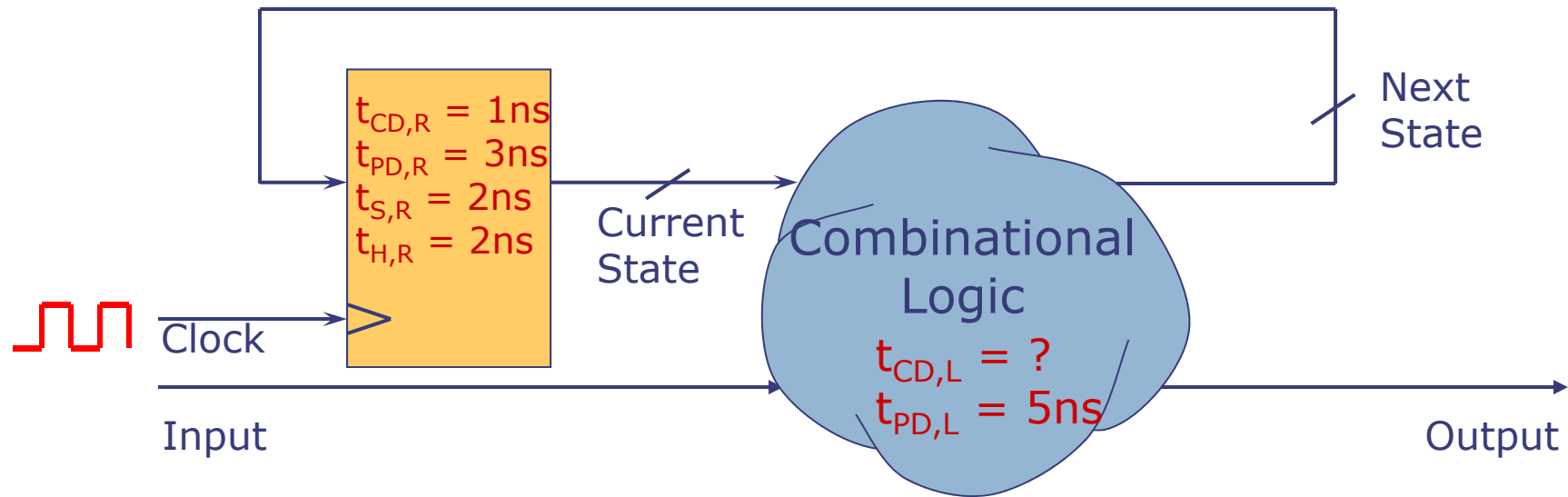
Tools may need to add logic to fast paths to meet  $t_{\text{HOLD}}$

# Timing Summary

---

- For a sequential circuit to work properly, we must guarantee that the **setup time** and **hold time** constraints of every register will always be satisfied.
- The **setup time** constraint is affected by both the logic in the circuit *and* the clock period. To fix violations, either:
  - Change the logic to be faster (lower  $t_{PD}$ )
  - Change the clock to be slower (higher  $t_{CLK}$ )
  -
- The **hold time** constraint is affected *only* by the logic in the circuit. Changing the clock period will not fix violations.
  - Sum of contamination delays must be greater than the register hold time, otherwise the circuit won't work.
- If hold time is satisfied, then circuit can be clocked at a clock period that is at least as long as the **maximum sum of the propagation delays plus setup time** across all register to register paths.

# Sequential Circuit Timing Example



## Questions:

- Constraints on  $t_{CD}$  for the logic?
- Minimum clock period?

$$t_{CD,R} (1 \text{ ns}) + t_{CD,L} (?) \geq t_{H,R} (2 \text{ ns})$$

$$t_{CD,L} \geq 1 \text{ ns}$$

$$t_{CLK} \geq t_{PD,R} + t_{PD,L} + t_{S,R} = 10 \text{ ns}$$

# Thank you!

Next lecture:  
Sequential logic in Minispec