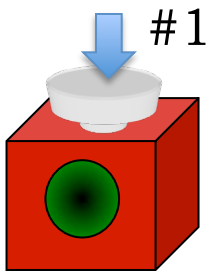
5. Sequential Logic

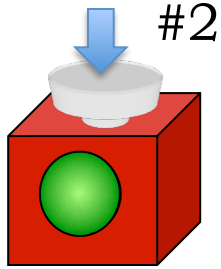
6.004x Computation Structures
Part 1 – Digital Circuits

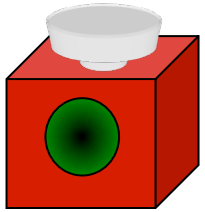
Copyright © 2015 MIT EECS

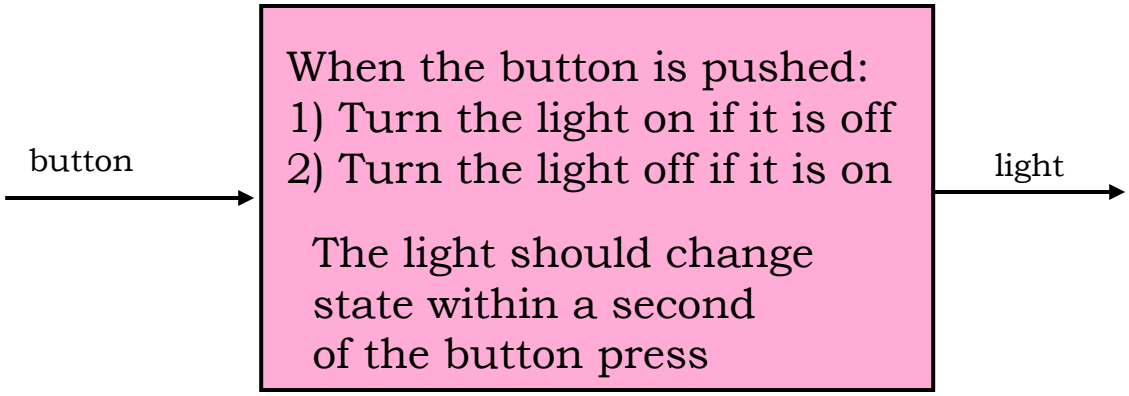
Something We Can't Build (Yet)

What if you were given the following design specification:





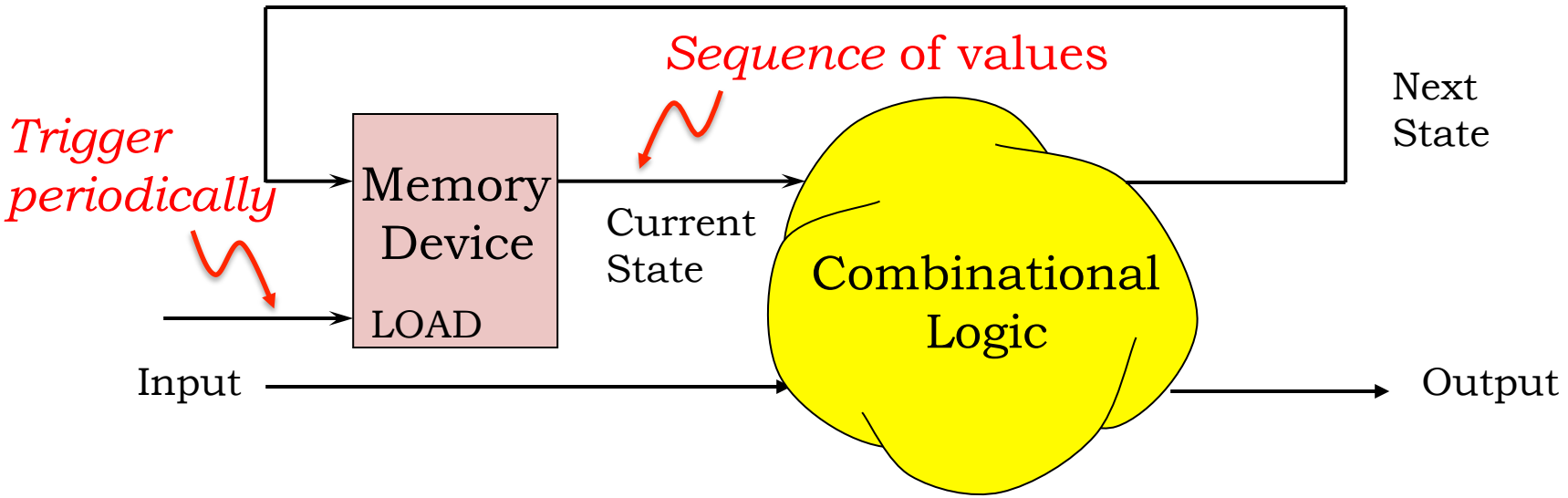




What makes this device different from those we've discussed before?

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

Digital State: What We'd Like to Build



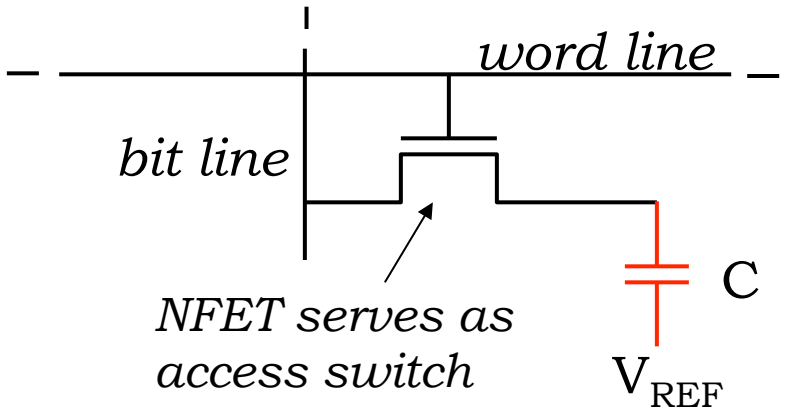
Plan: Build a Sequential Circuit with stored digital STATE –

- Memory stores CURRENT state, produced at output
- 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 Capacitors

We've chosen to encode information using voltages and we know from physics that we can "store" a voltage as charge on a capacitor:



Pros:

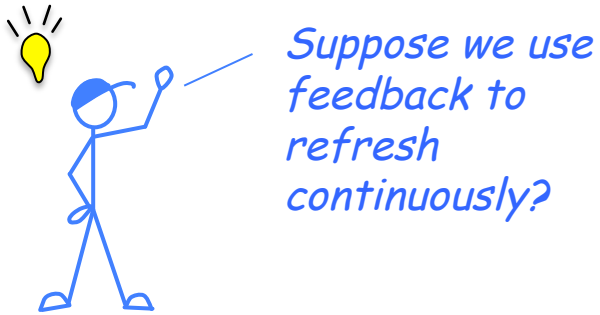
- compact – low cost/bit (on BIG memories)

Cons:

- complex interface
- stable? (noise, …)
- it leaks! ⇒ refresh

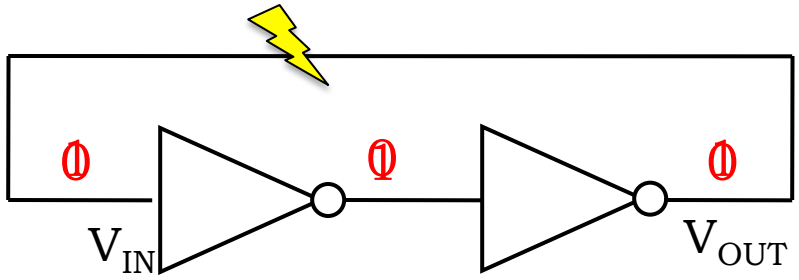
To write:
Drive bit line, turn on access fet, force storage cap to new voltage

To read:
precharge bit line, turn on access fet, detect (small) change in bit line voltage

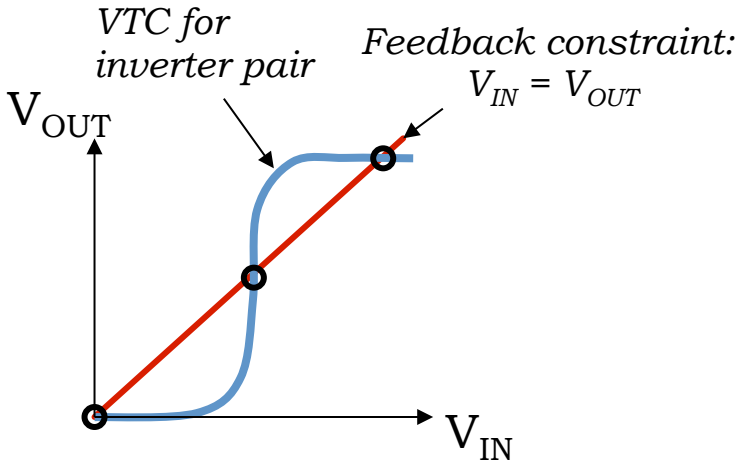


Memory: Using Feedback

IDEA: use positive 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



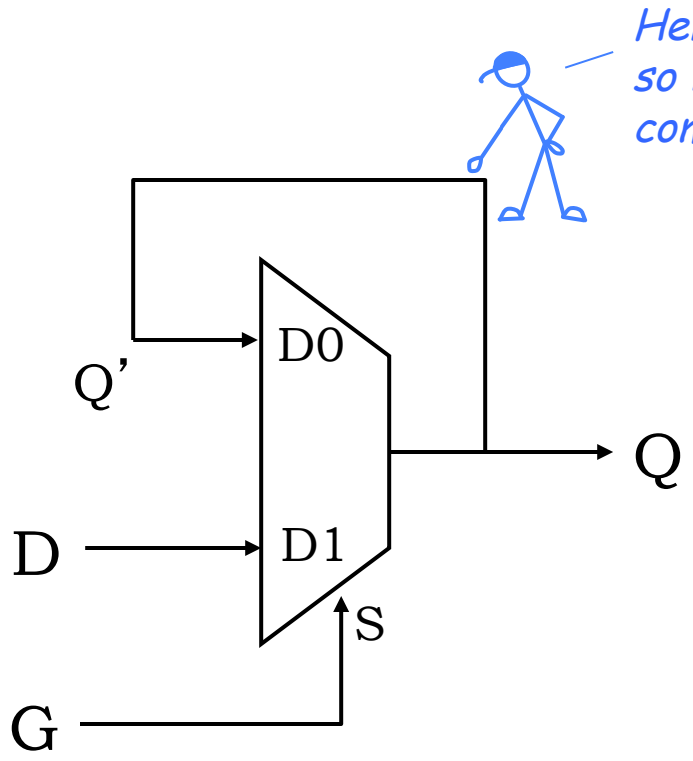
Not affected by noise

Three solutions:
- two end-points are stable
- middle point is metastable

We'll get back to this!

Settable Memory Element

It's easy to build a settable storage element (called a latch) using a *lenient* MUX:



Here's a feedback path, so it's no longer a combinational circuit.

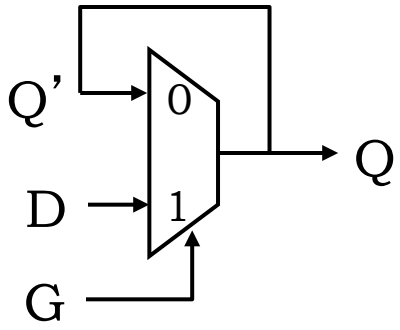
"state" signal appears as both input and output

G	D	Q'	Q	
0	--	0	0	Q stable
0	--	1	1	
1	0	--	0	Q follows D
1	1	--	1	

D: data input
G: gate input
Q: state output

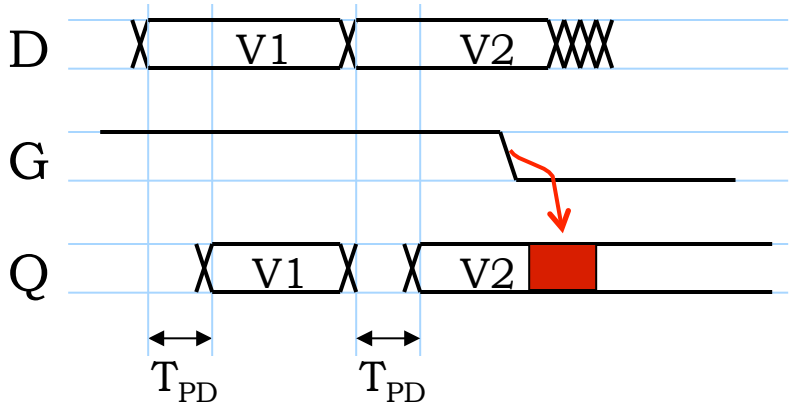
New Device: D Latch

Circuit:

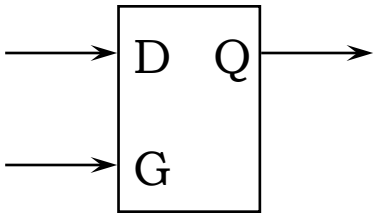


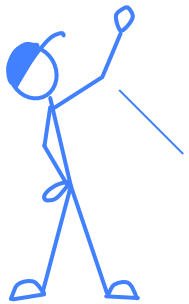
G=1:
Q follows D

G=0:
Q holds



Schematic Symbol:



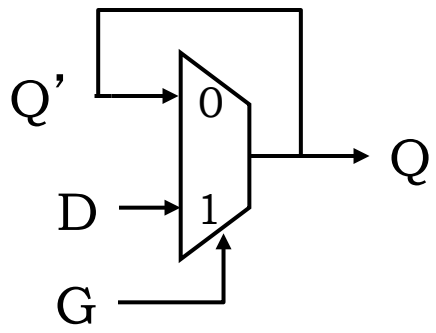


BUT… A change in D or G contaminates Q, hence Q' … how can this possibly work?

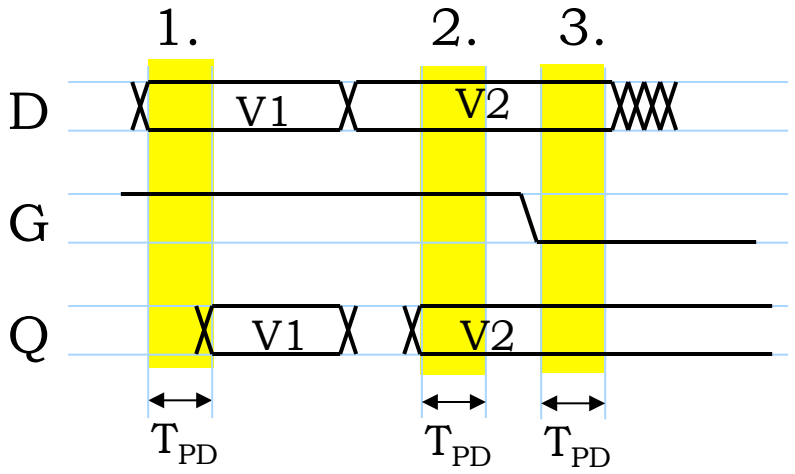
G=1: Q Follows D, *independently of Q'*

G=0: Q Holds stable Q', *independently of D*

A Plea for Lenience



G	D	Q'	Q
1	0	X	0
1	1	X	1
X	0	0	0
X	1	1	1
0	X	0	0
0	X	1	1

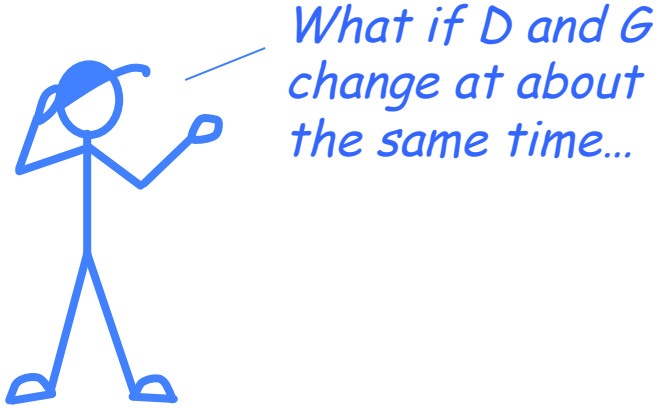


Assume LENIENT Mux, propagation delay of T_{PD}

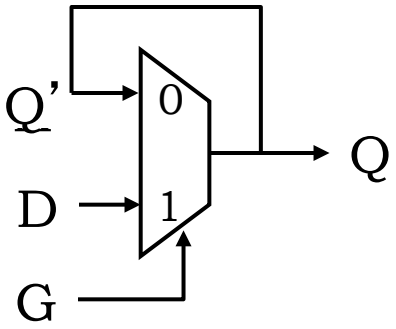
Then output valid when

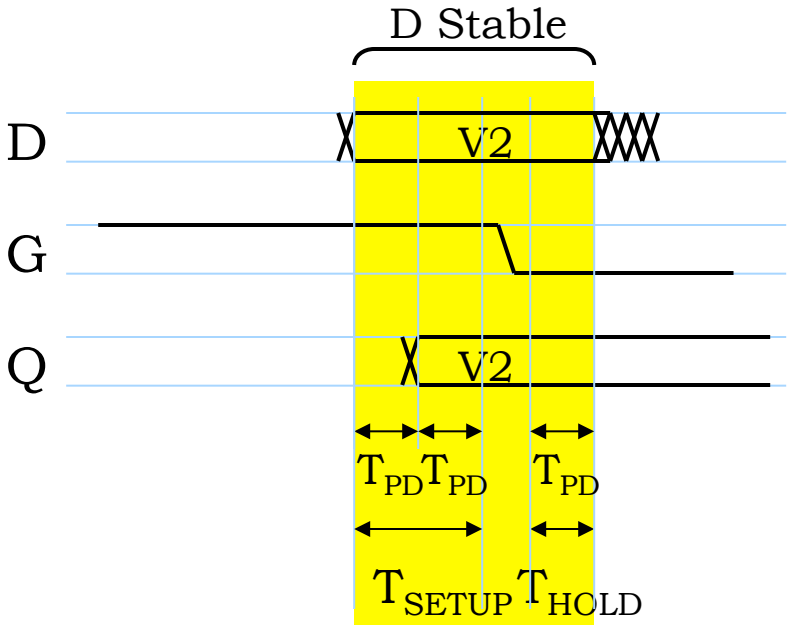
1. G=1, D stable for T_{PD}, *independently of Q'; or*
2. Q=D stable for T_{PD}, *independently of G; or*
3. G=0, Q stable for T_{PD}, *independently of D*

Does lenience *guarantee* a working latch?



…With a Little Discipline





To <u>reliably latch</u> V2:

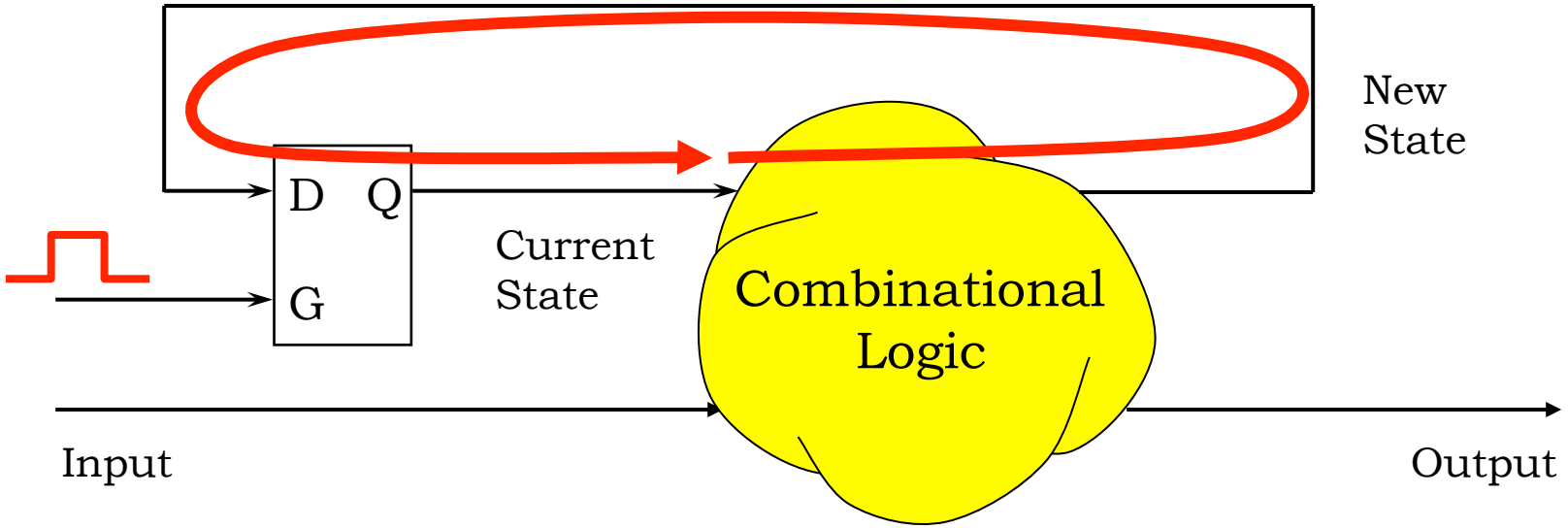
- Apply V2 to D, holding G=1
- After T_{PD}, V2 appears at Q=Q'
- After another T_{PD}, Q' & D both valid for T_{PD}; *will hold Q=V2 independently of G*
- Set G=0, while Q' & D hold Q=D
- After another T_{PD}, G=0 and Q' are sufficient to hold Q=V2 *independently of D*

Dynamic Discipline for our latch:

$T_{SETUP} = 2T_{PD}$: interval *prior to* G transition for which D must be stable & valid

$T_{HOLD} = T_{PD}$: interval *following* G transition for which D must be stable & valid

Let's Try It Out!



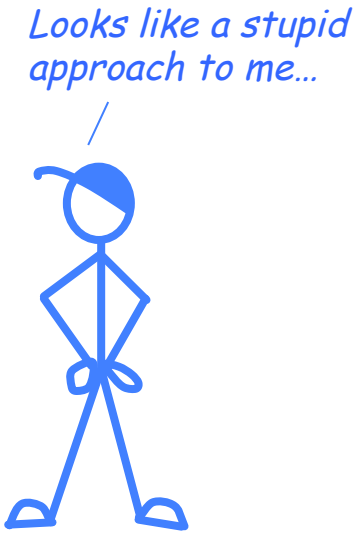
When G=1, latch is *Transparent…*

… provides a combinational path from D to Q.

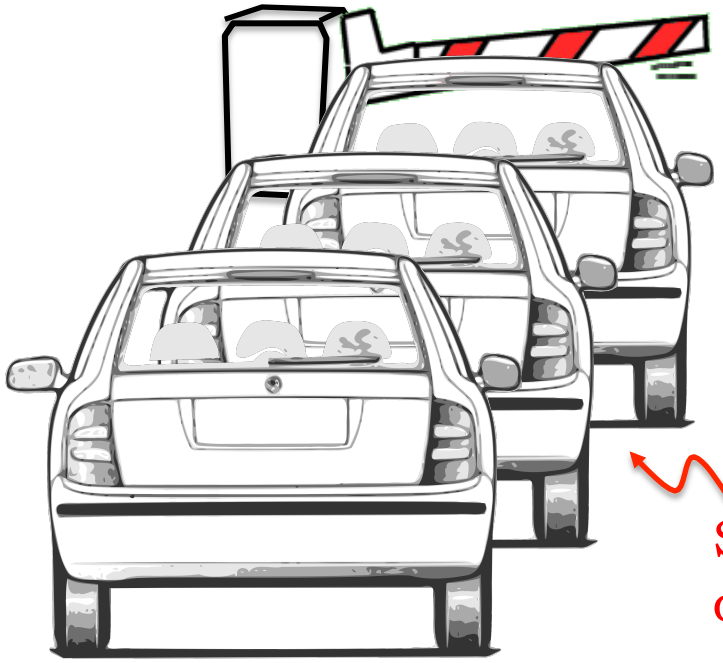
Can't work without tricky timing constraints on G=1 pulse:

- Must fit within contamination delay of logic
- Must accommodate latch setup, hold times

Want to signal an INSTANT, not an INTERVAL…

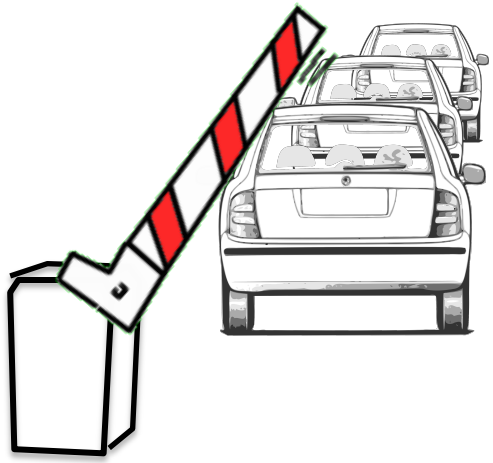


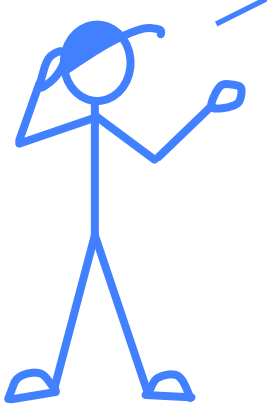
Flakey Control Systems



Sequence of values

Gate closed

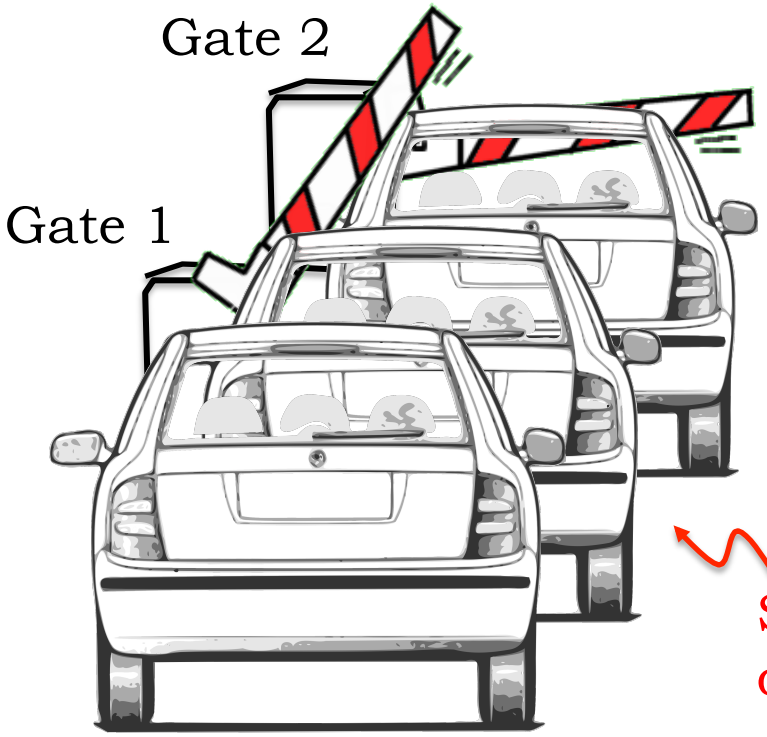




How do we ensure only one car gets through?

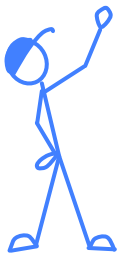
Gate open

Solution: Escapement Strategy (2 gates)

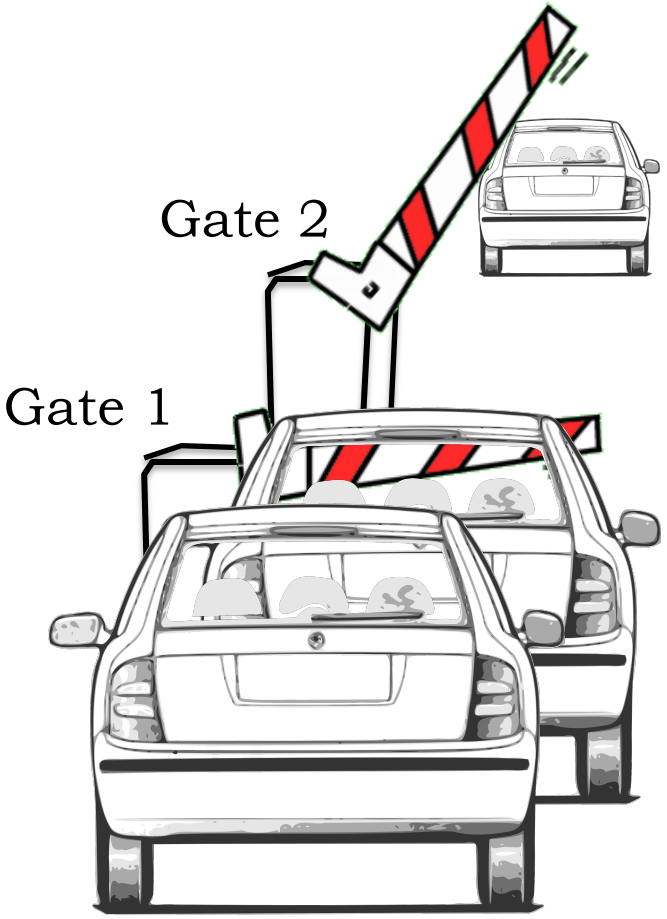


Gate 1: open
Gate 2: closed

Sequence of values

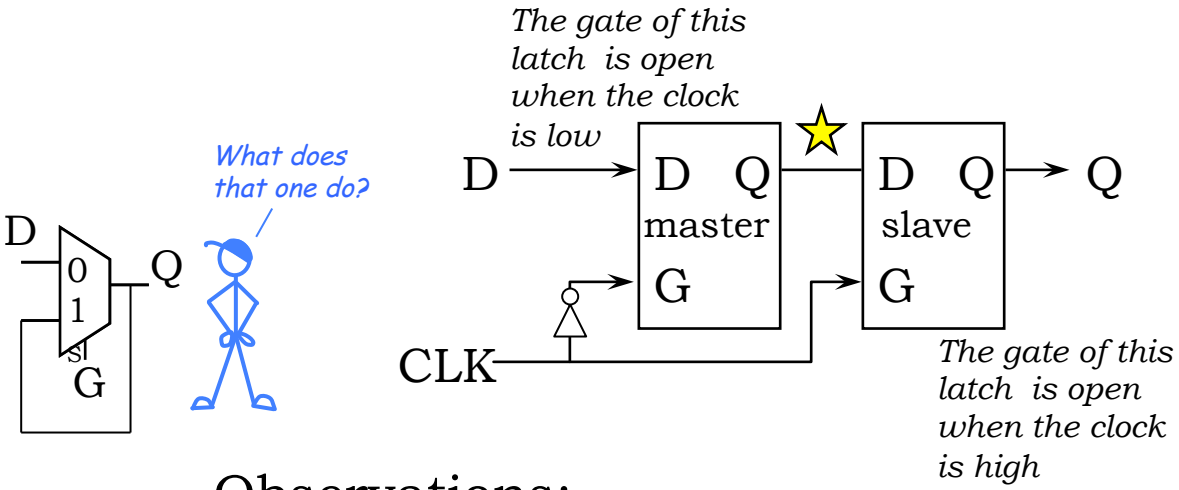


Key: at no time is there a path through both gates



Gate 1: closed
Gate 2: open

(Edge-Triggered) D Register



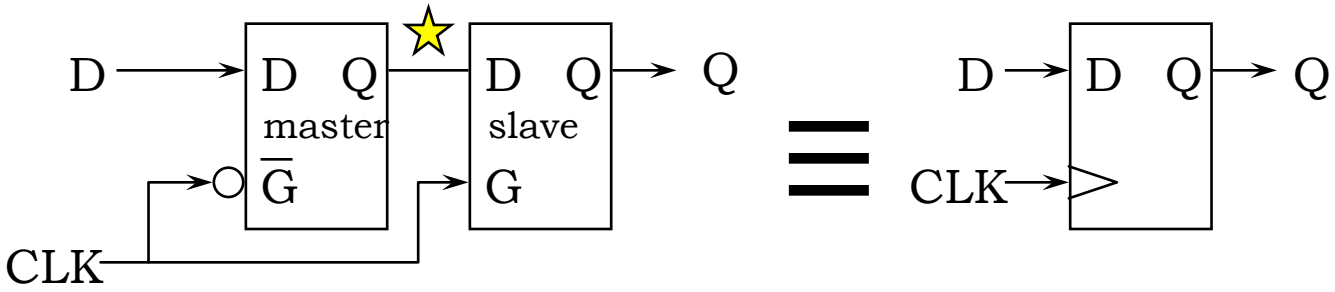
Observations:

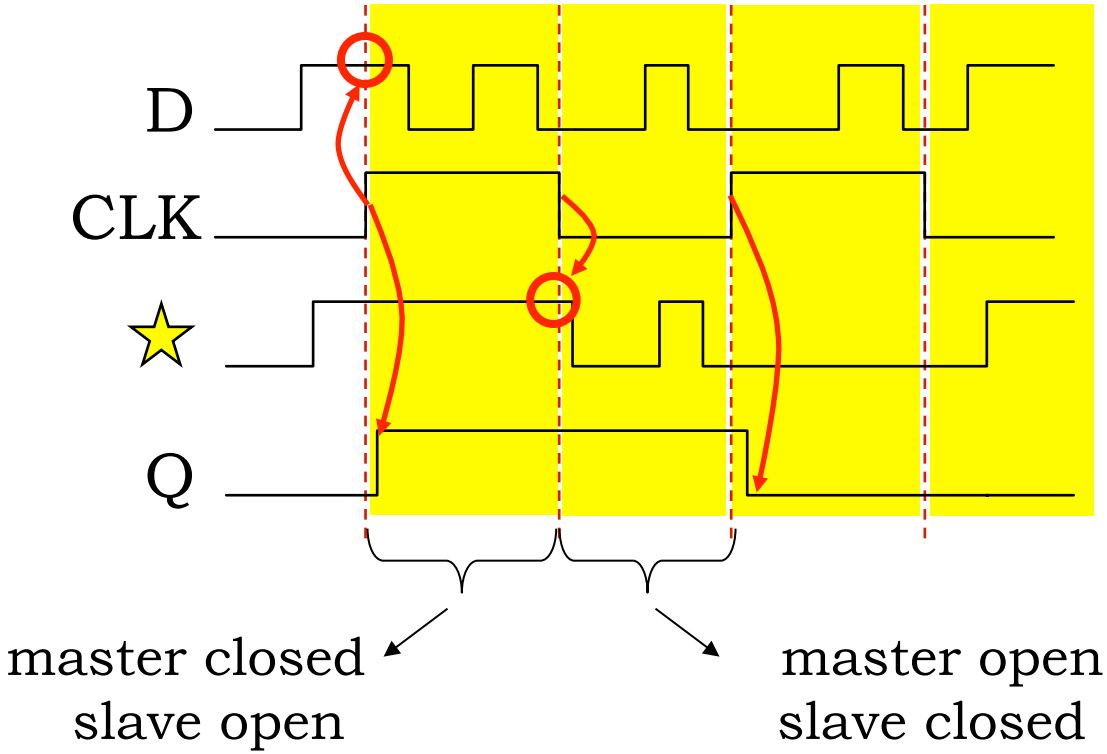
- only one latch "transparent" at any time:
  - master closed when slave is open
  - slave closed when master is open

  ⇒ no combinational path through register

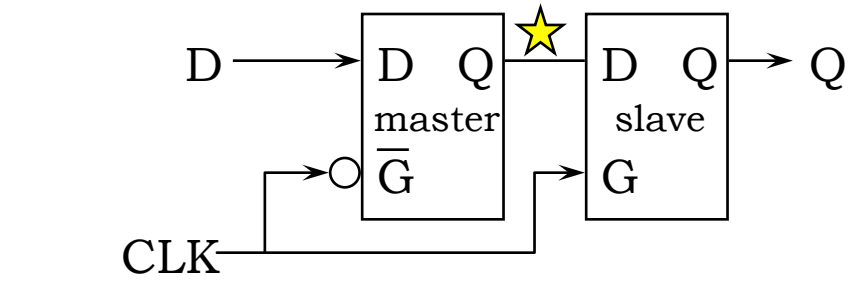
  (the feedback path in one of the master or slave latches is always active)

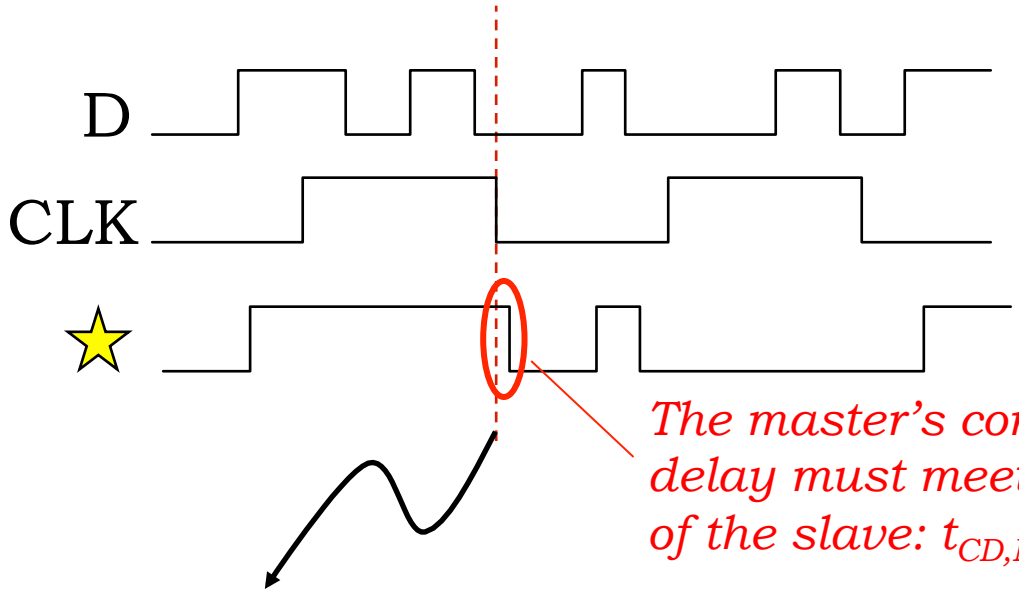
D-Register Waveforms





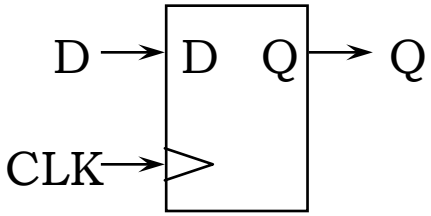
Um, about that hold time…

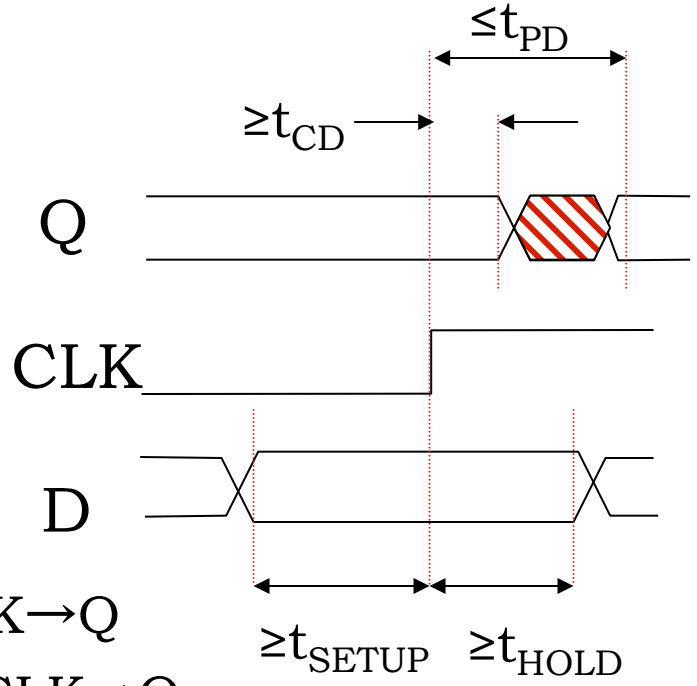




Slave latch is closing ⇒ ☆ must meet setup/hold times
but master latch is opening so ☆ may change

D-Register Timing 1





t_{PD}: maximum propagation delay, CLK→Q

t_{CD}: minimum contamination delay, CLK→Q

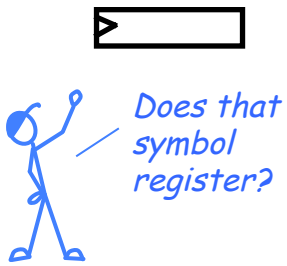
t_{SETUP}: setup time

guarantee that D has propagated through feedback path before master closes

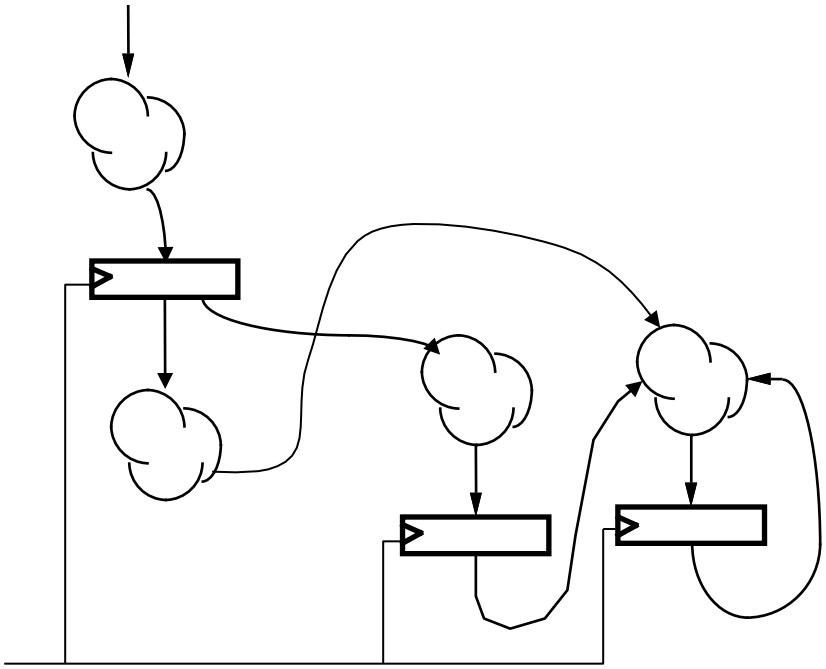
t_{HOLD}: hold time

guarantee master is closed and data is stable before allowing D to change

Single-clock Synchronous Circuits



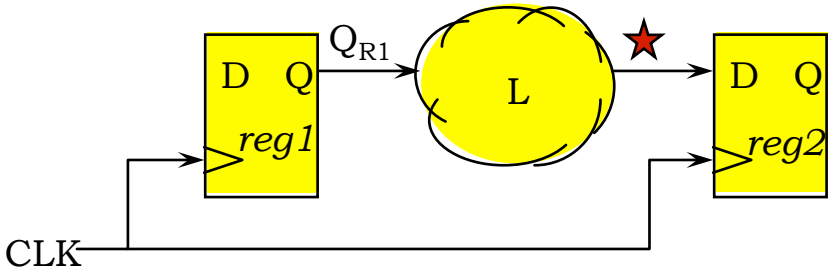
We'll use registers in a highly constrained way to build digital systems:

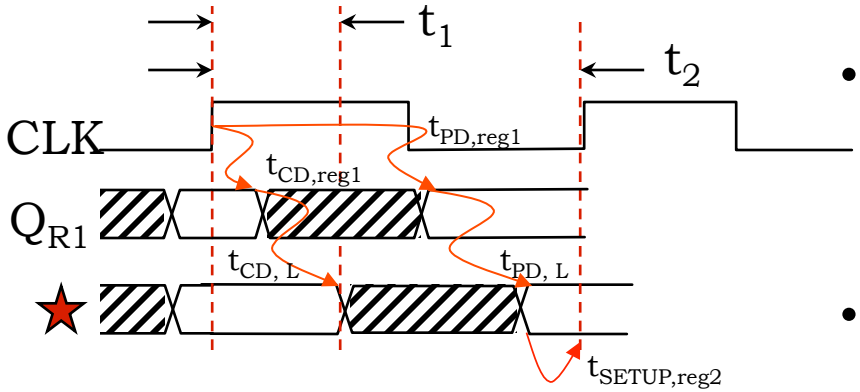


Single-clock Synchronous Discipline

- No combinational cycles
- Single periodic clock signal shared among all clocked devices
- Only care about value of register data inputs just before rising edge of clock
- Period greater than every combinational delay + setup time
- Change saved state after noise-inducing logic transitions have stopped!

Timing in a Single-clock System





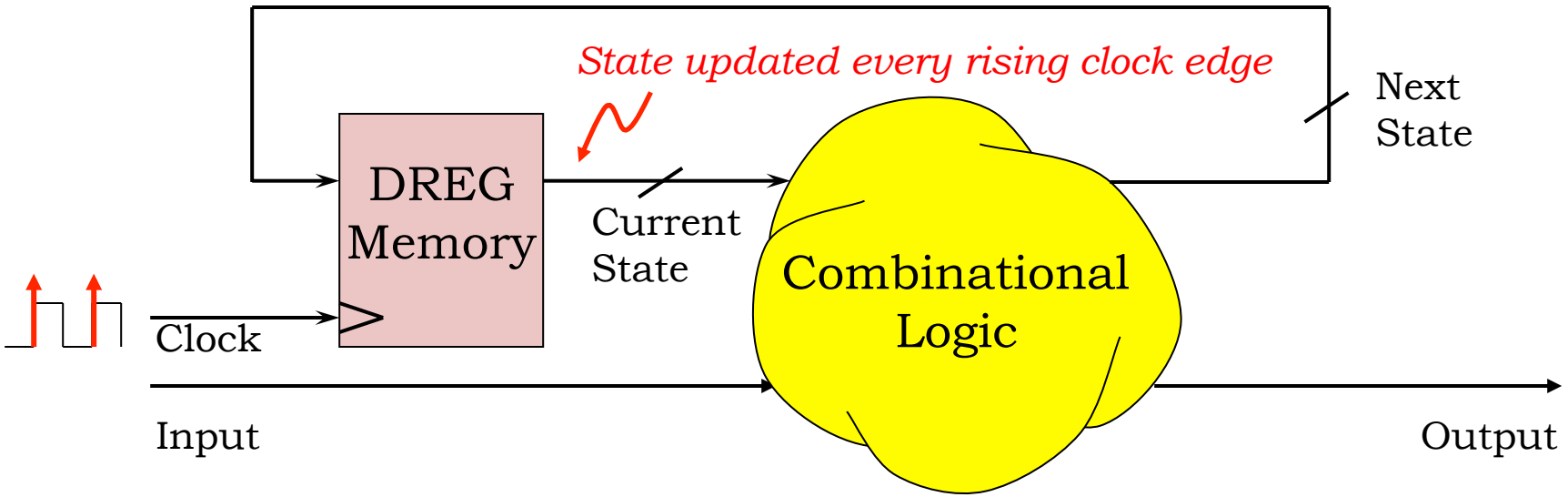
$t_1 = t_{CD,reg1} + t_{CD,L} \geq t_{HOLD,reg2}$

$t_2 = t_{PD,reg1} + t_{PD,L} + t_{SETUP,reg2} \leq t_{CLK}$

Questions for register-based designs:

- how much time for useful work (i.e. for combinational logic delay)?
- what happens if there's no combinational logic between two registers?
- what happens if CLK signal doesn't arrive at the two registers at exactly the same time (a phenomenon known as "clock skew")?

Model: Discrete Time

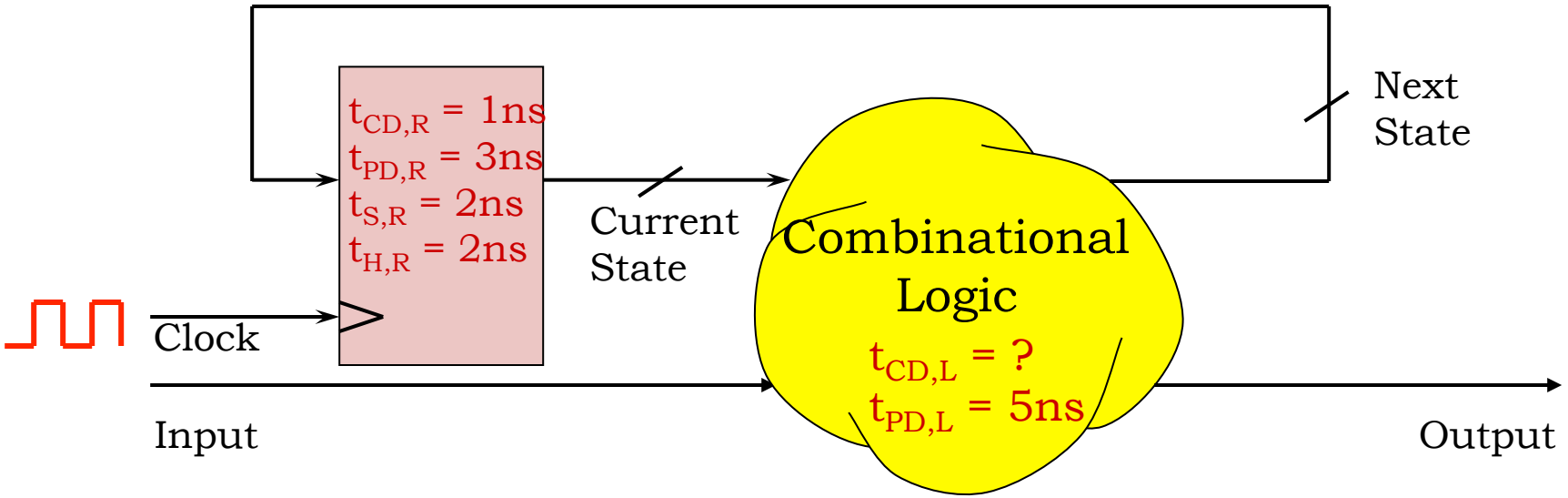


Active Clock Edges punctuate time ---

- Discrete Clock periods
- Sequences of states
- Simple rules – eg truth tables – relating outputs to inputs and the current state)

- ABSTRACTION: Finite State Machines (next lecture!)

Sequential Circuit Timing



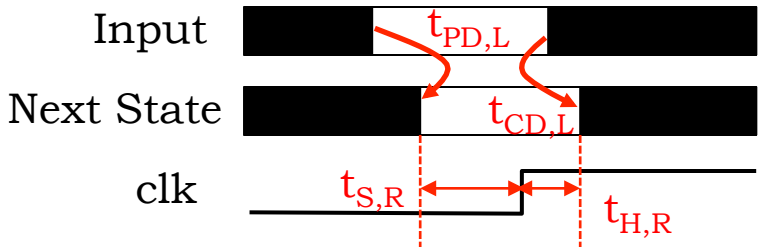
Questions:

- Constraints on t_{CD} for the logic?
- Minimum clock period?
- Setup, Hold times for Inputs?

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

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

$t_{S,INPUT} = t_{PD,L} + t_{S,R}$ = 7 nS
$t_{H,INPUT} = t_{H,R} - t_{CD,L}$ = 1 nS



Summary

Basic memory elements:

- Feedback, detailed analysis => basic level-sensitive devices (eg, latch)
- 2 Latches => Register
- Dynamic Discipline: constraints on input timing

Synchronous 1-clock logic:

- Simple rules for sequential circuits
- Yields clocked circuit with T_S, T_H constraints on input timing

Finite State Machines

Next Lecture Topic!