DEPARTMENT OF
INFORMATION SYSTEMS
AND COMPUTER SCIENCE

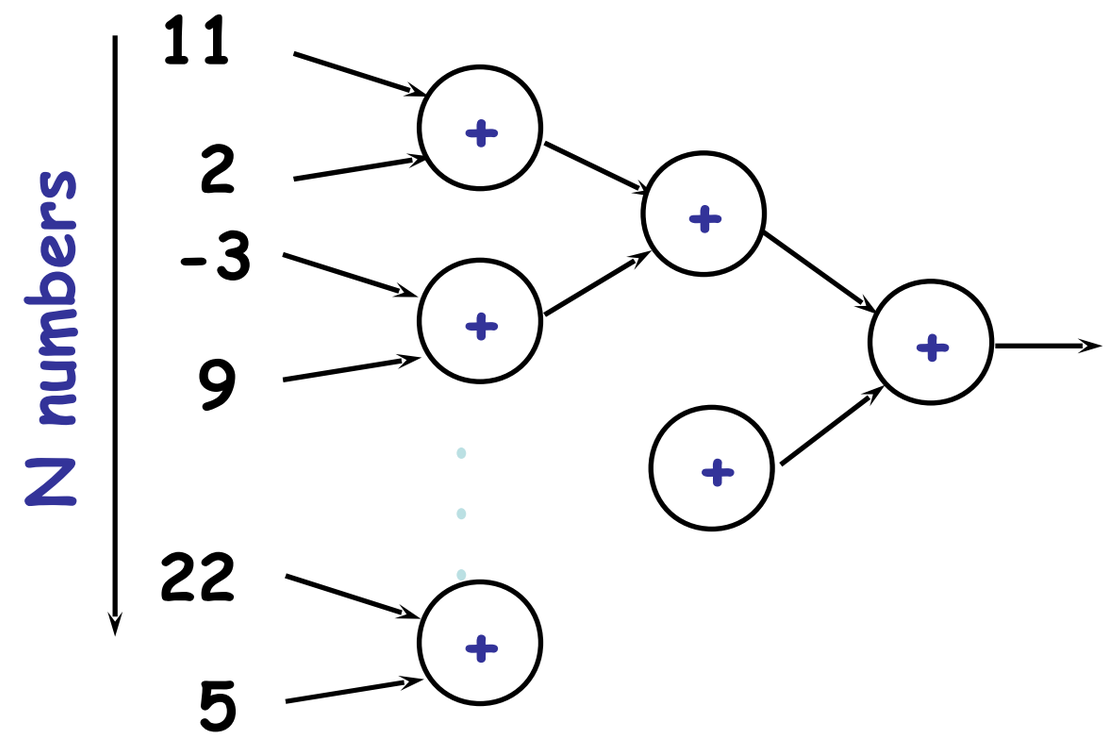
Feedback, State, and Memory

Cyclic Circuits – Can They Work?

Adding a column of N numbers

- Form tree of adders?
- Delay is $O(\log_2 N)$
- Size grows as $O(N)$
  - (N/2)+(N/4)+(N/8)... = N-1
  - or, 1+2+4+8+16+…+N/2 = N-1
- or, think of it as a tennis tournament: each player loses *exactly* once except for champion. Therefore, N-1 matches.

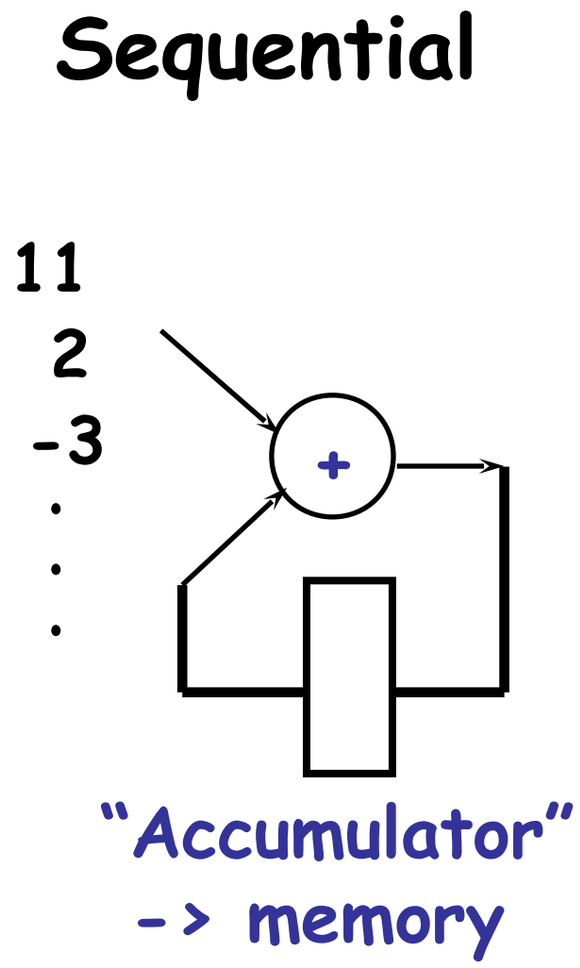
Combinational



- Not scalable to arbitrary N!
  - Need infinite hardware!
- Not practical for general CPU.

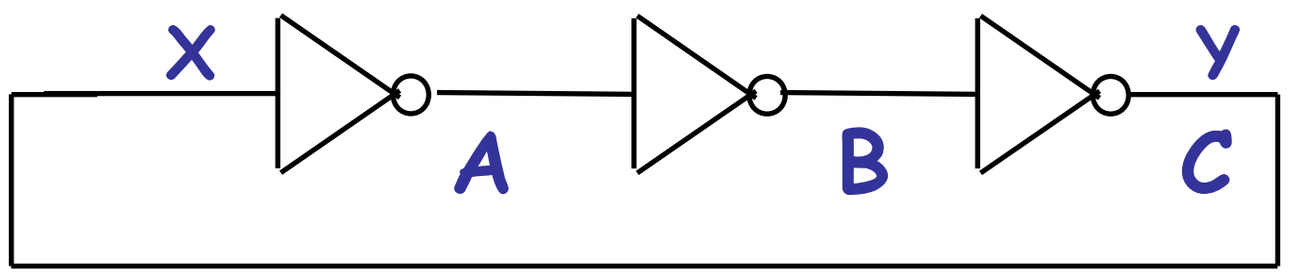
Adding a column of N numbers

- Use an "accumulator"?
  - *Remember* previous value and feed it to next input: NEED MEMORY
- Called "Sequential"
  - Because things happen "in sequence" over time.
- Size is constant!
- Delay is O(N) "cycles"
- A bit slower, but definitely more practical and scalable to arbitrary N.



Implementing Memory: Inverter Ring?

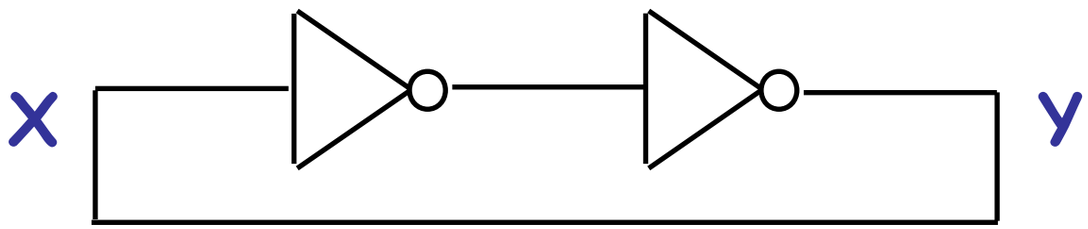
- Odd-number cycle --- unstable: values always change



t_{pd}

$V_y = \overline{\text{old } V_X} = \text{new } V_X$ ← oscillates!

- Even-number cycle --- stable (?): values don't change

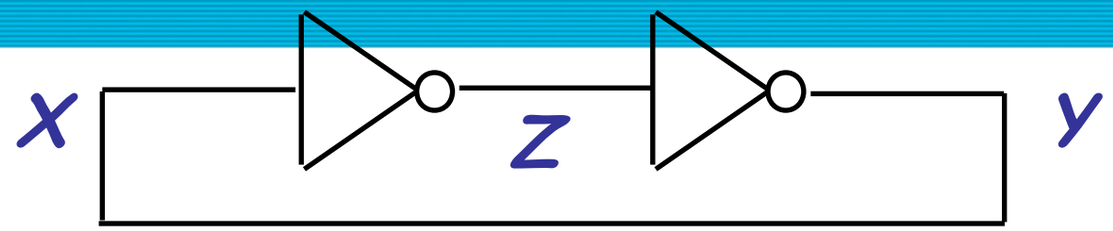


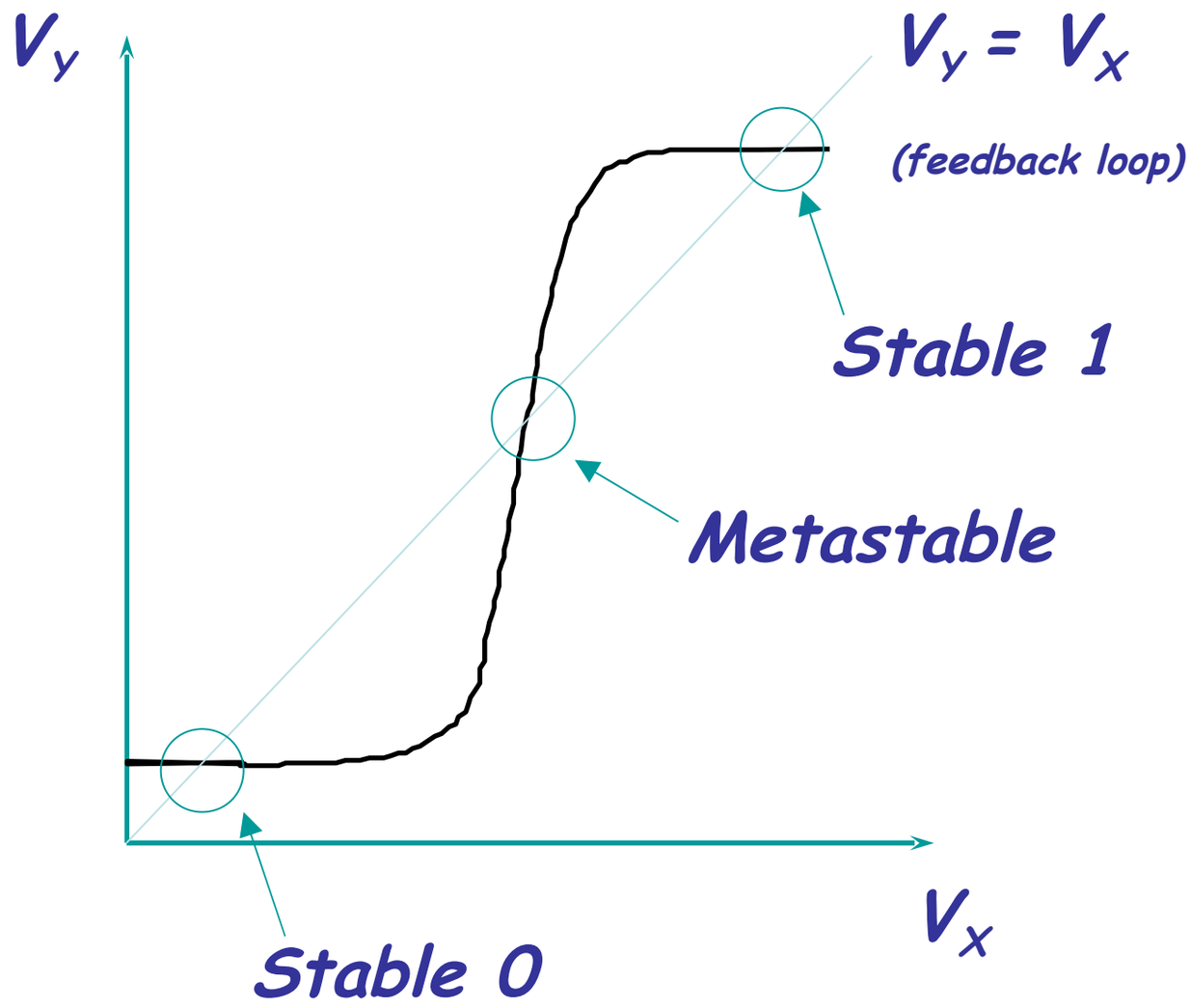
$V_y = \text{INVERT}(\text{INVERT}(V_X))$

$V_y = V_X$ ← (potentially) stable

A Closer Look

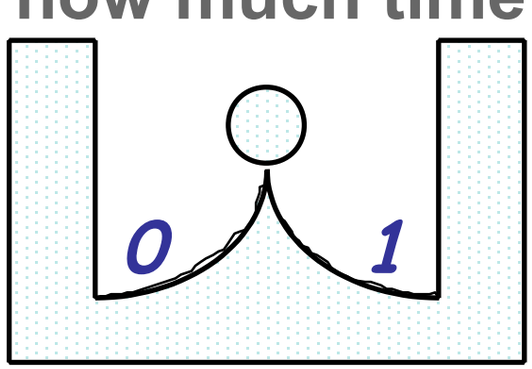
- 3 points on output curve where $V_Y=V_X$
- <u>Stable</u> 0 and 1 (2 points)
  - Slight deviation away from stable point will get pushed back by high-gain part.
    - Value will hold. (Slight deviations will correct themselves.)
- <u>Metastable</u> state (3rd point)
  - Slight deviation will push it quickly to either 0 or 1.

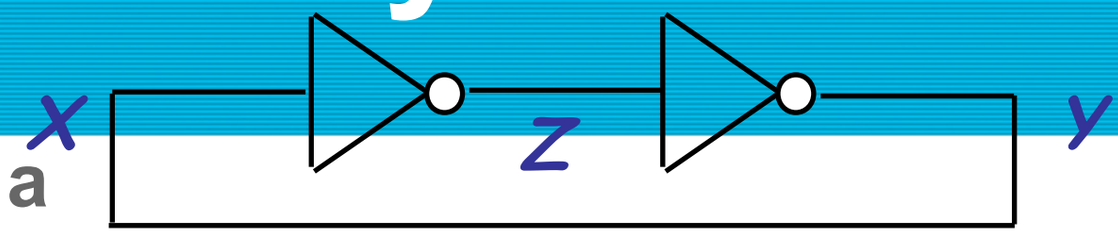


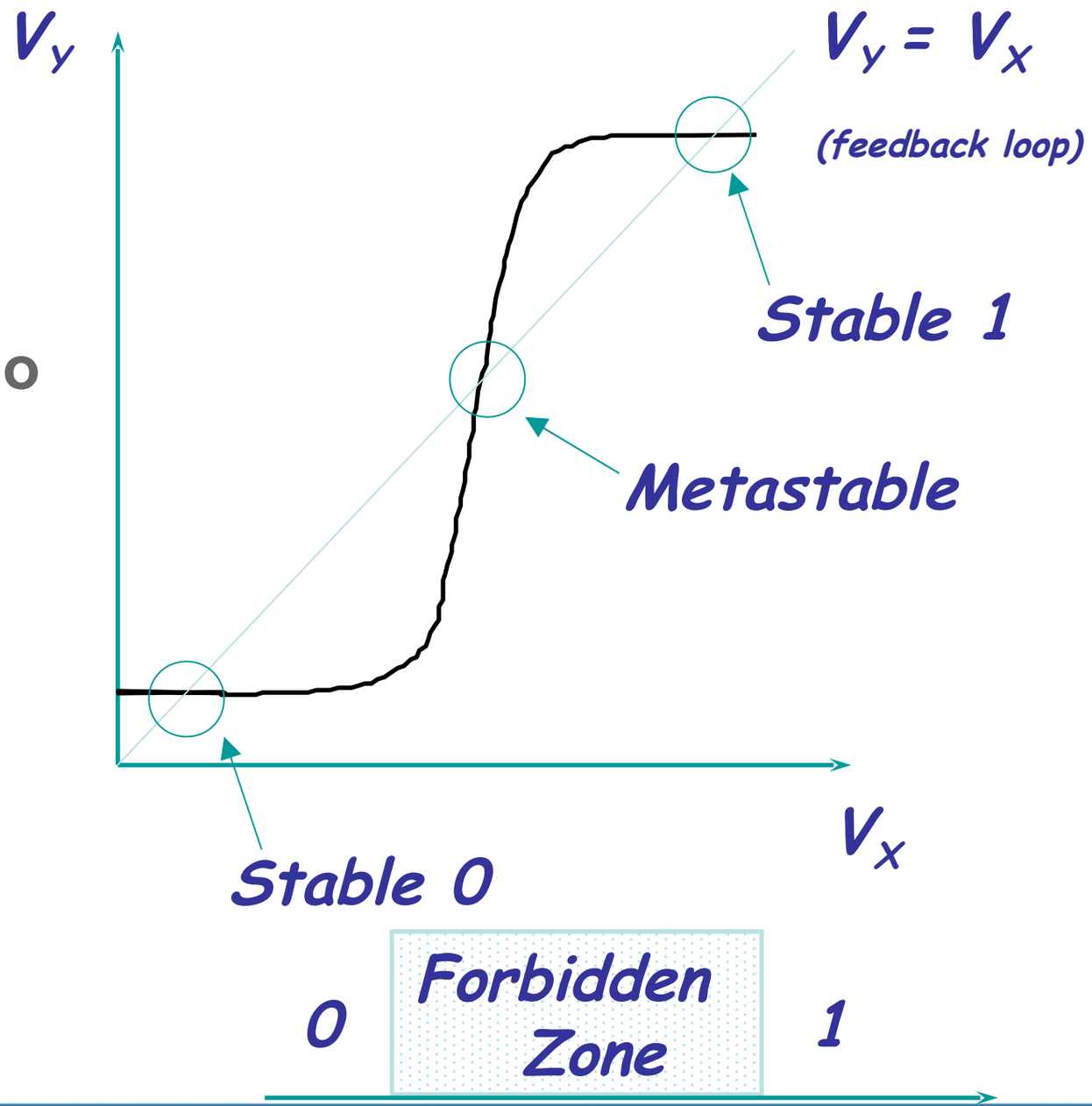


Metastability

- Like having a ball balanced on a hill:
  - Probability of staying metastable gets exponentially small in time, but always non-zero.
  - There is always a tiny chance that it will remain metastable no matter how much time passes.



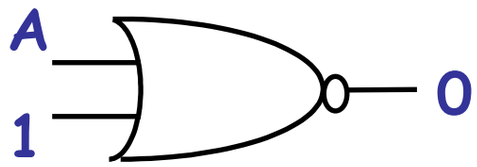




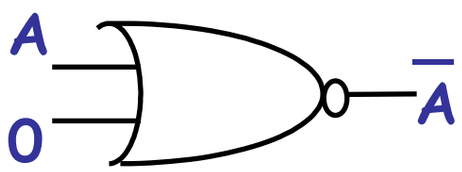
- Caused by feeding forbidden values into loop.
  - Therefore, avoid doing so.

Setting 0 or 1: the SR Latch

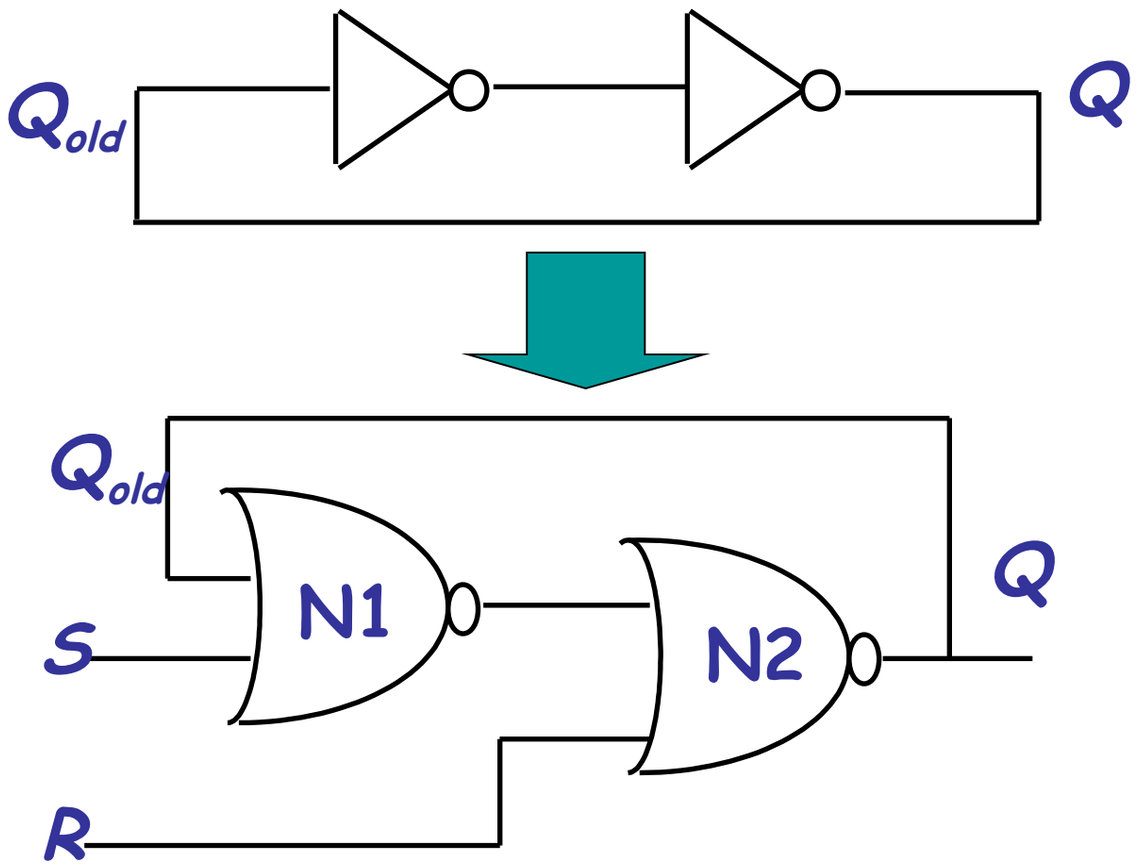
- Problem with Inverter Ring: No inputs!
- Even if it works:
  - Need “controllable” inverters!
- NOR
  - 1 -> always 0



  - 0 -> inverter



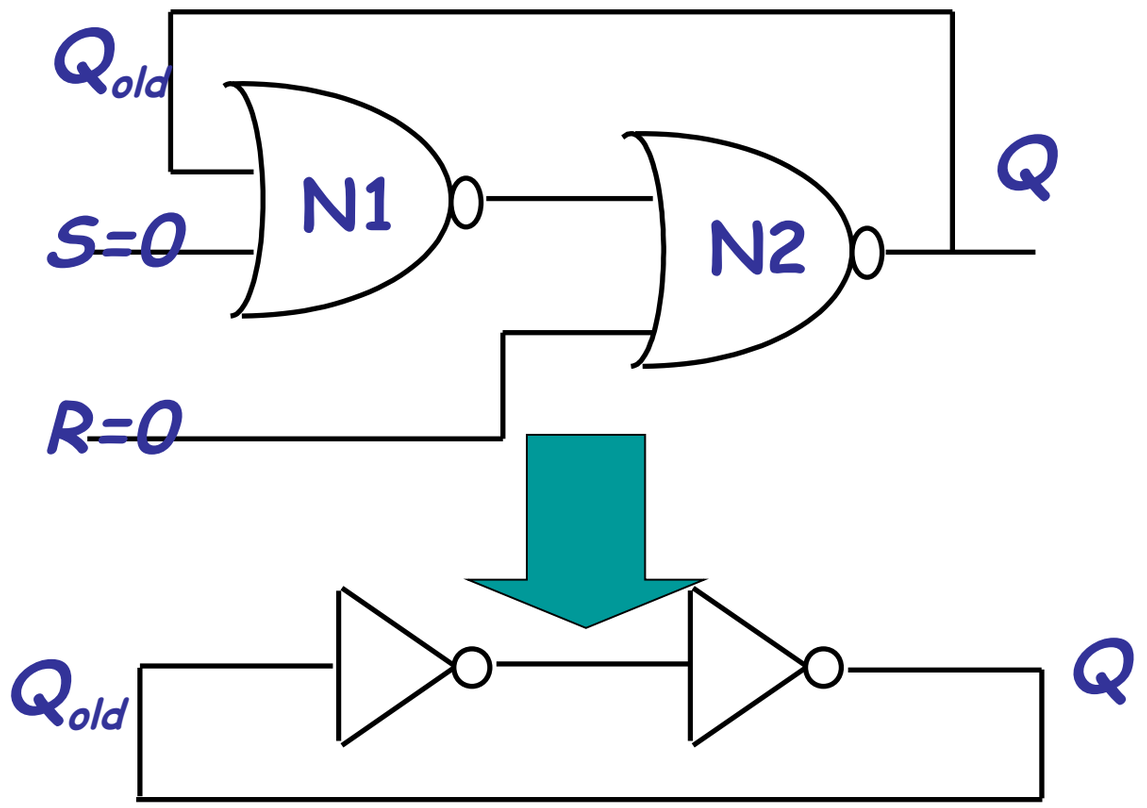
- Control configuration via S and R.



S	R	Q
0	0	Q_{old}
0	1	0
1	0	1
1	1	0

NOR-NOR SR Latch

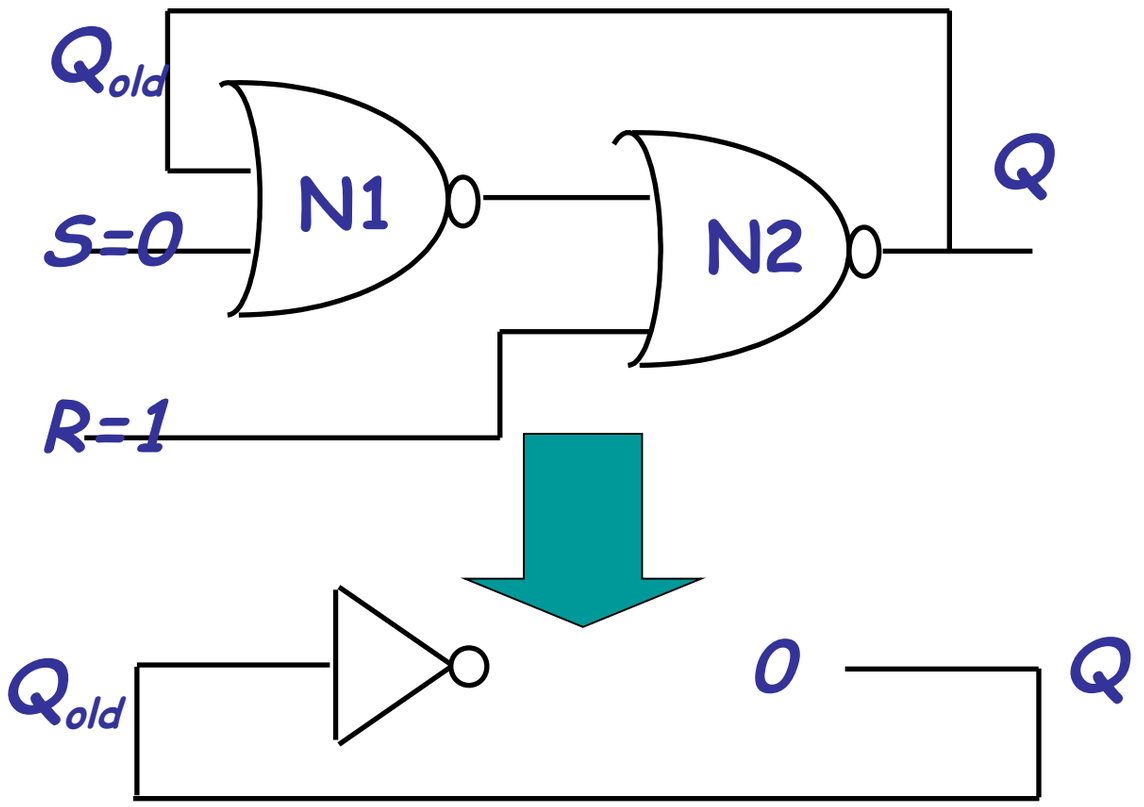
- S=0, R=0 -> “Latch”
  - Inverter loop
  - Q = Qold (keep old value)



S	R	Q
0	0	Q_{old}
0	1	0
1	0	1
1	1	0

NOR-NOR SR Latch

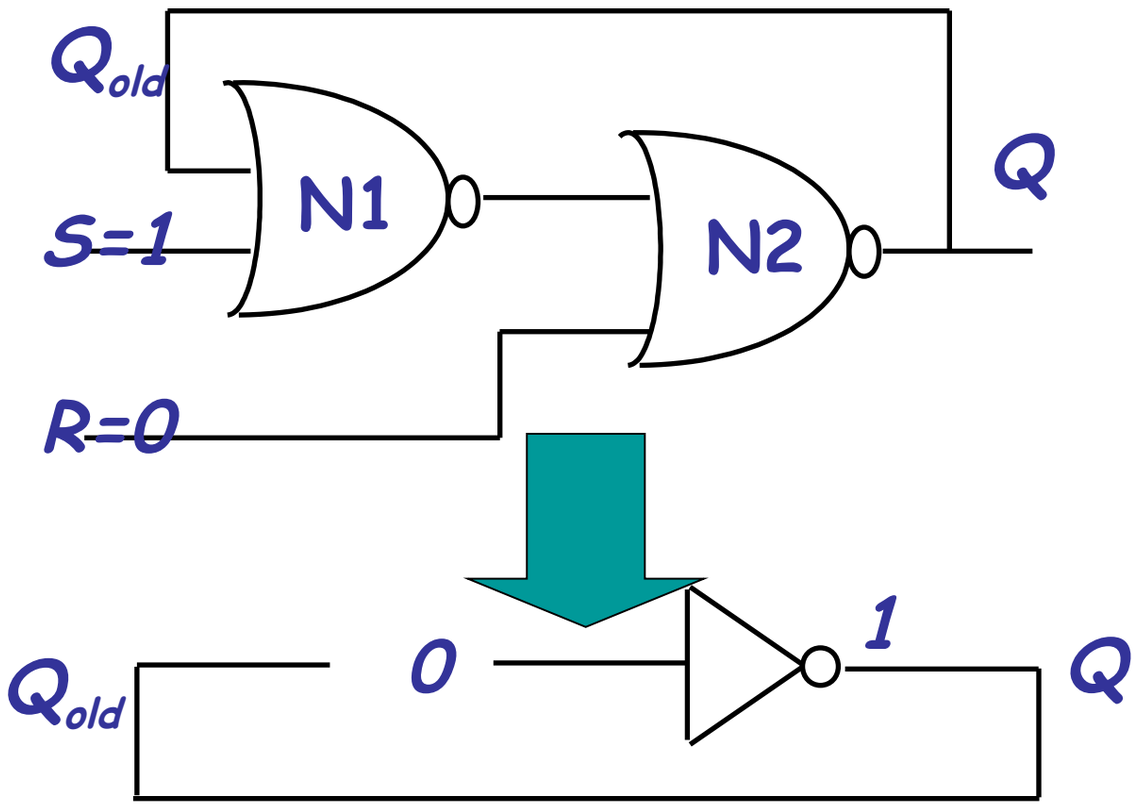
- S=0, R=1 -> “Reset”
  - N2 output always 0
  - Q = 0



S	R	Q
0	0	Q_{old}
0	1	0
1	0	1
1	1	0

NOR-NOR SR Latch

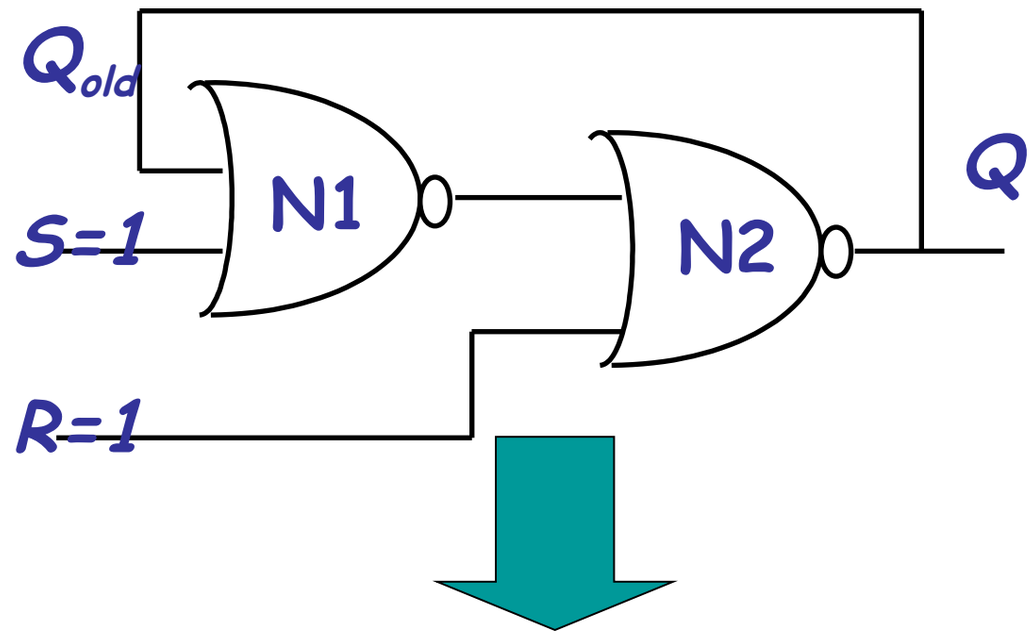
- S=1, R=0 -> "Set"
  - N1 output always 0, N2 inverts -> Q = 1



S	R	Q
0	0	Q_{old}
0	1	0
1	0	1
1	1	0

NOR-NOR SR Latch

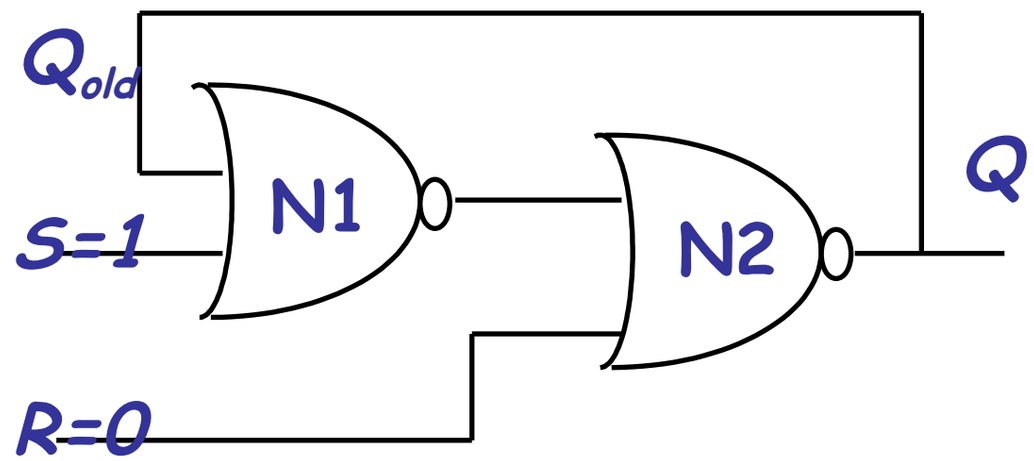
- S=1, R=1 (avoid)
  - results in Q =0, BUT, cannot *safely* switch back to S=0, R=0



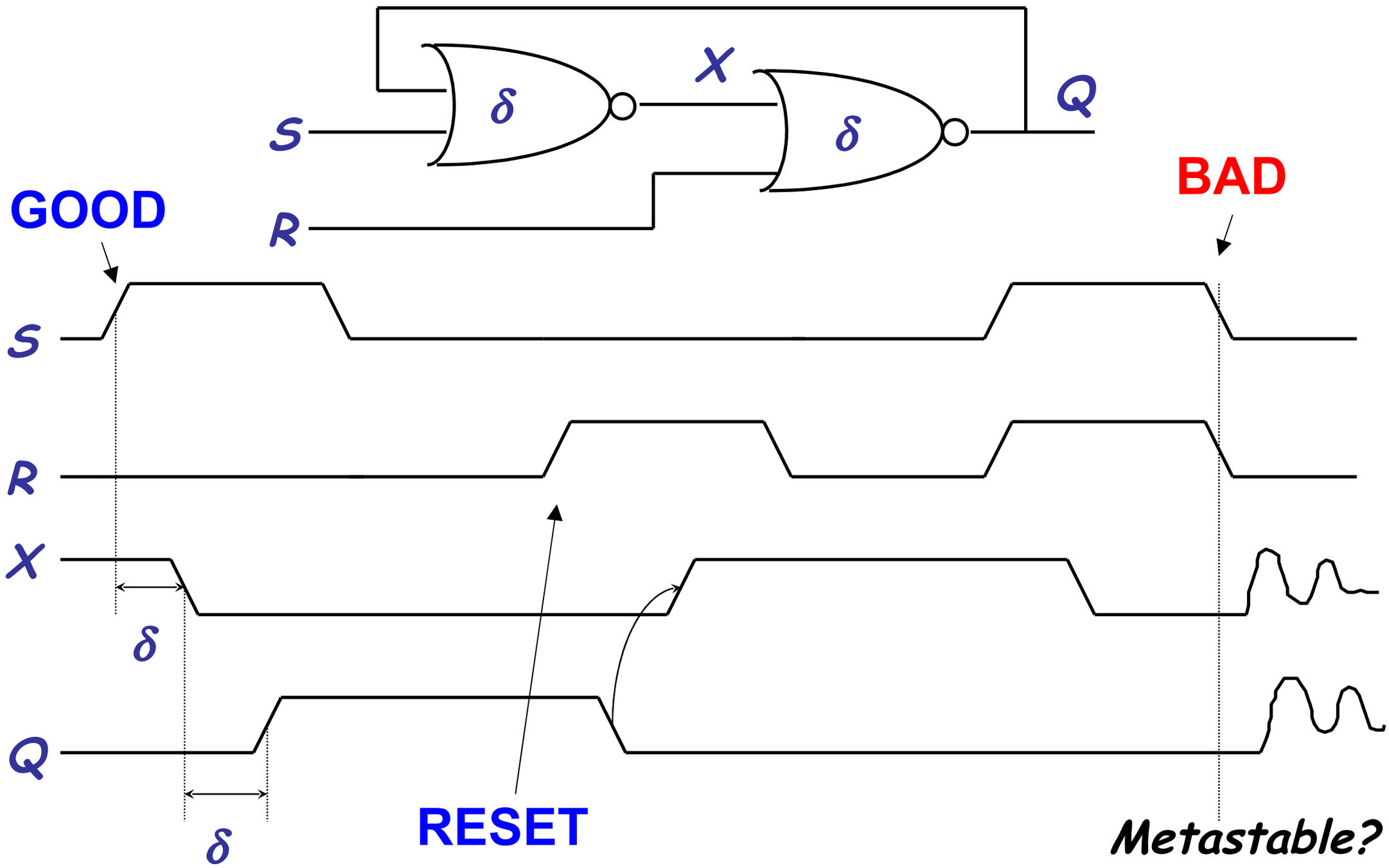
S	R	Q
0	0	Q_{old}
0	1	0
1	0	1
1	1	0

NOR-NOR SR Latch

- To use, either
  SET (S=1, R=0), or
  RESET (S=0, R=1).
- Then, hold.
  - Long enough for signals to propagate around loop.
- Then, **LATCH** by setting S=0, R=0.
- **NEVER** go from S=1, R=1 to S=0, R=0!
  - Both S and R are changing.
  - No way to know which one will get latched.
  - Possible metastable state if caught at forbidden values!



S-R Latch Timing

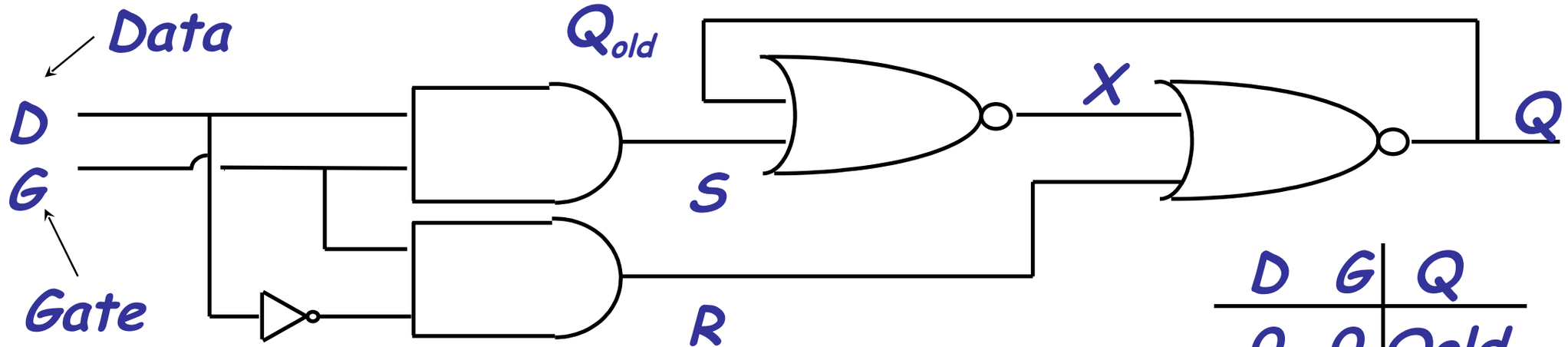


Timing Rules

- As much as possible, only one input can change “at the same time”.
  - Single input change or SIC rule.
- We also should
  allow input transitions only after state has settled!
  - We risk corrupting output if input changes while state/output is not yet stable.
  - This constraint will be quantified in terms of setup time and hold time.

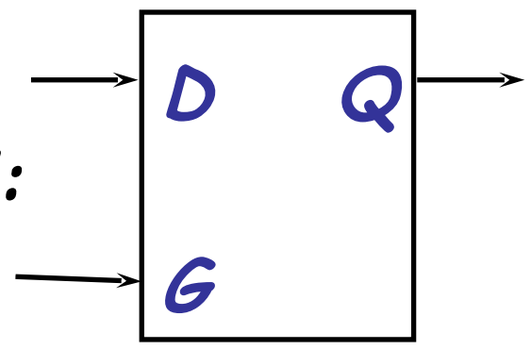
D-Latch or Transparent Latch

- If Gate is 0, then both S and R = 0.
- If Gate is 1, then S = D, R = ~D.
  - Also, Q = D, hence the name.



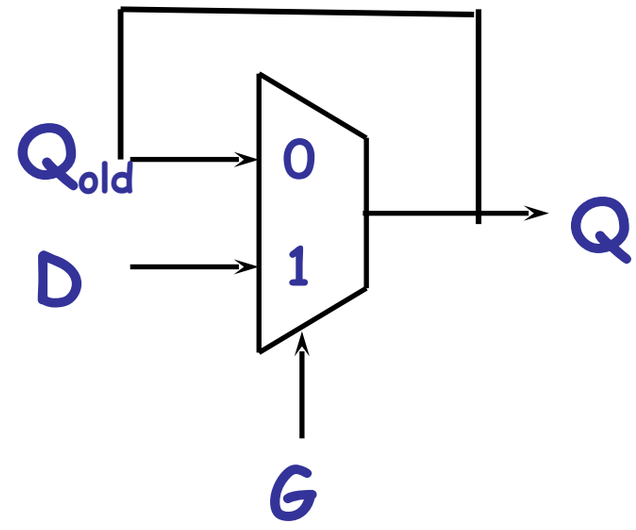
D	G	Q
0	0	Qold
0	1	0
1	0	Qold
1	1	1

Symbol:

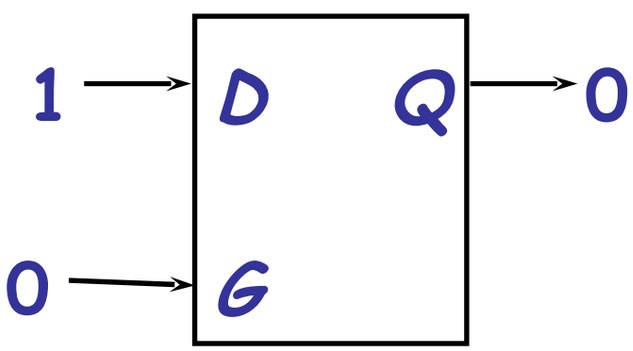


Another implementation of D-Latch

- Can be seen as a mux loop:
  - Q = D, if G=1 “transparent”.
  - Q = Q_{old} if G=0 “latched”.
- Or, can be seen as a “gate”:
  - G=1 “open”
  - G=0 “closed”



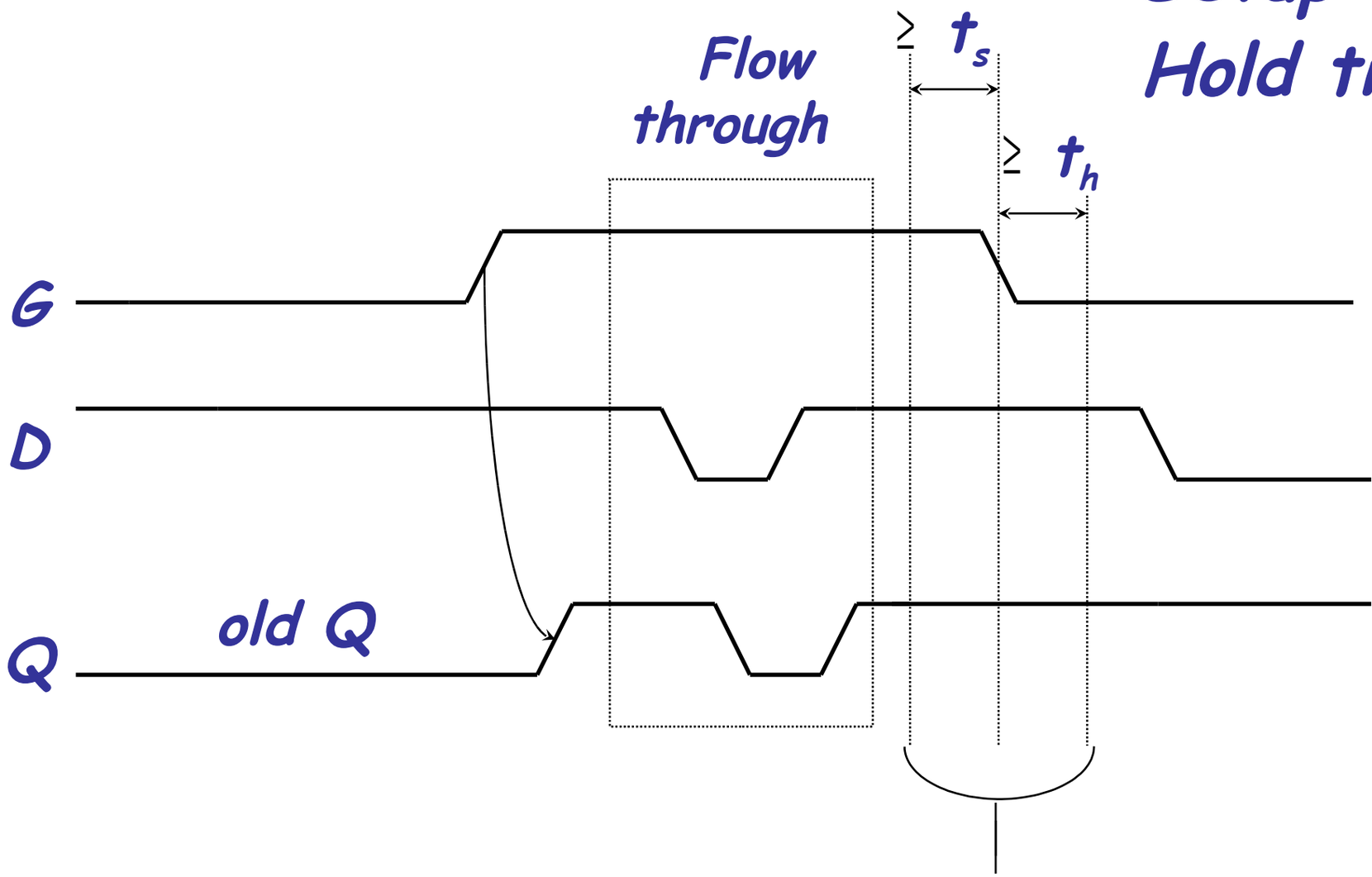
- Different from CMOS transmission gate because it has memory.
  - Transmission gate outputs hi-Z (disconnected) when closed .
  - D-latch outputs old value.



Timing of D-Latch

Setup time = t_s

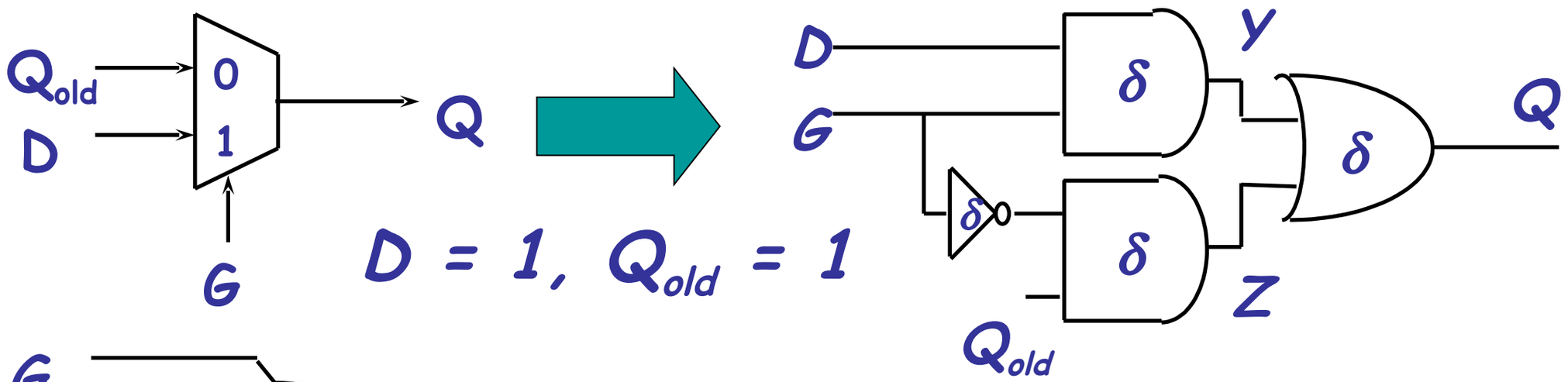
Hold time = t_h

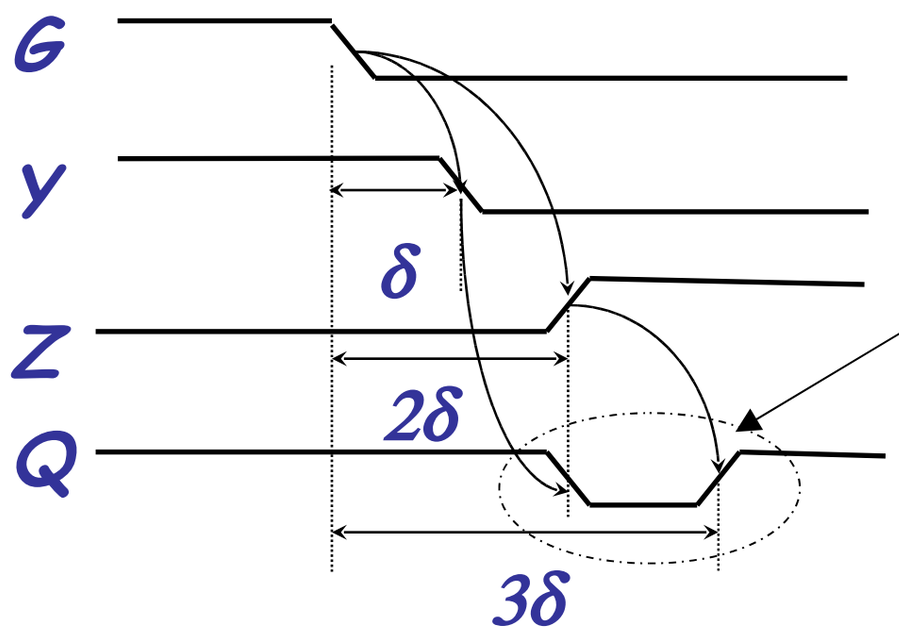


D must not change in this region

Reality Bites: Hazards

- In combinational logic, we don't care what happens between t_{cd} and t_{pd}.

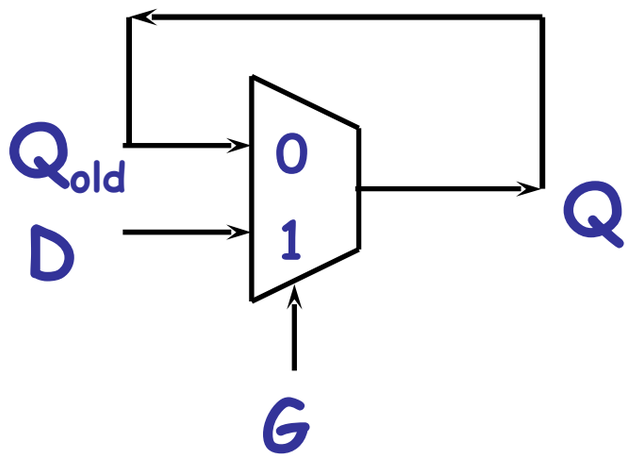


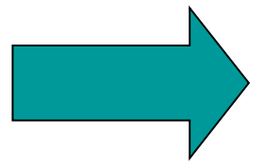


- There is a "**glitch**" in Q, but technically, this is OK, as long as we define t_{pd} to be at least 3δ.
- Remember, tpd = time from new valid in to new valid out.

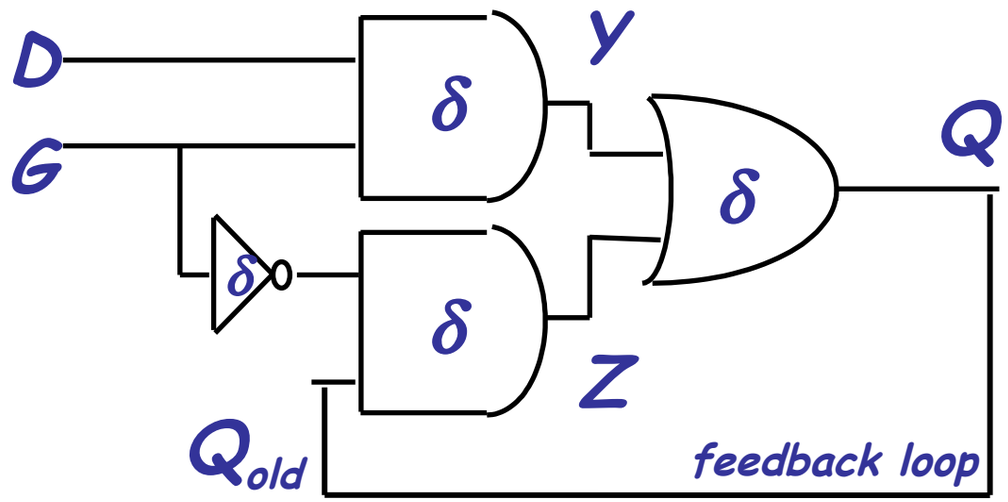
Reality Bites: Hazards

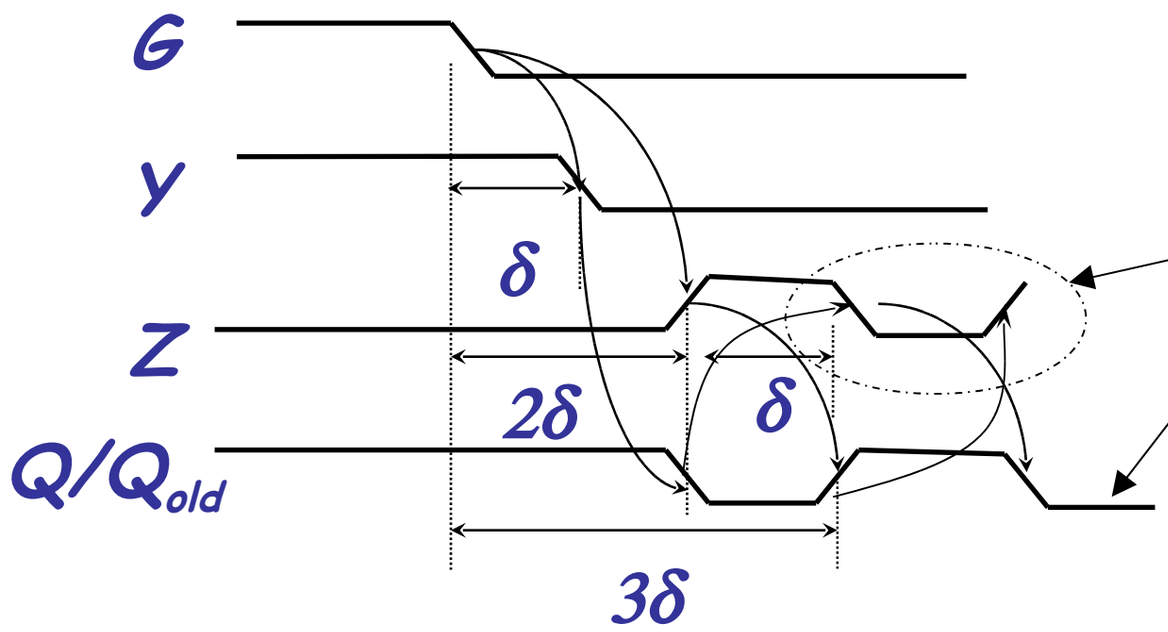
- But what happens when we add a feedback loop? (e.g., mux loop D-latch)





D = 1

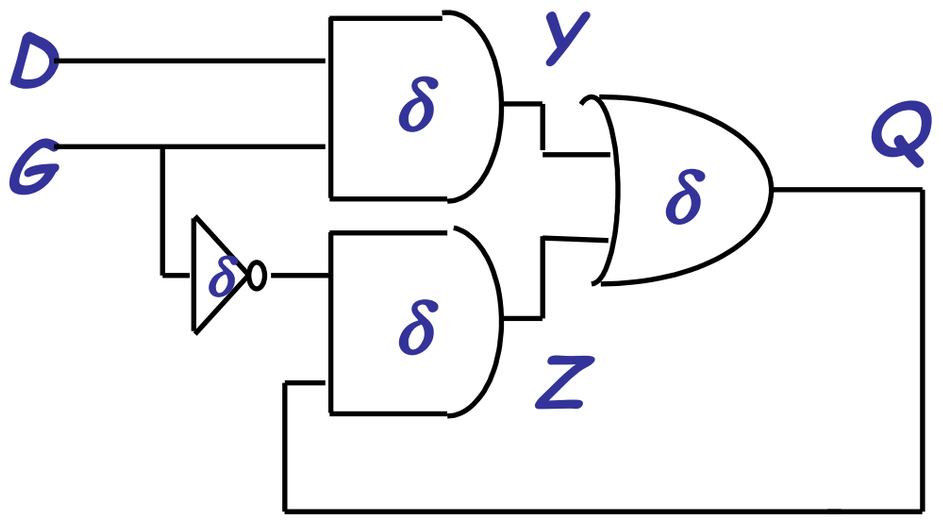




- "Glitch" gets into feedback.
  - Glitch in Q causes glitch in Z, then Q, etc. etc.
- Can cause oscillations, metastability, or wrong values.
  - We want Q to stay 1, but it doesn't!
- Note: Hazards happen even when timing rules are followed!

Fixing Hazards

- Hazard due to input moving from DG to GQ.
  - Inputs to OR *momentarily* both 0, due to delays.
    - "Glitch" in OR output.
- Think of patches as "umbrellas".
  - We "get wet" crossing from patch to patch.
- Solution: Add patches to cover gaps!



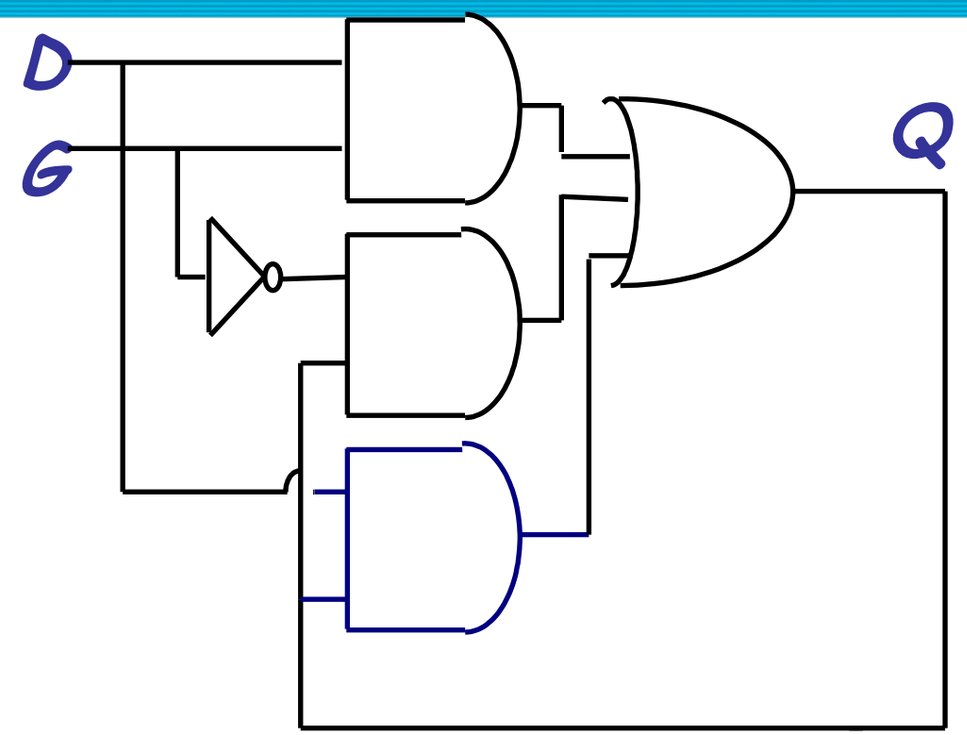
Karnaugh Map:

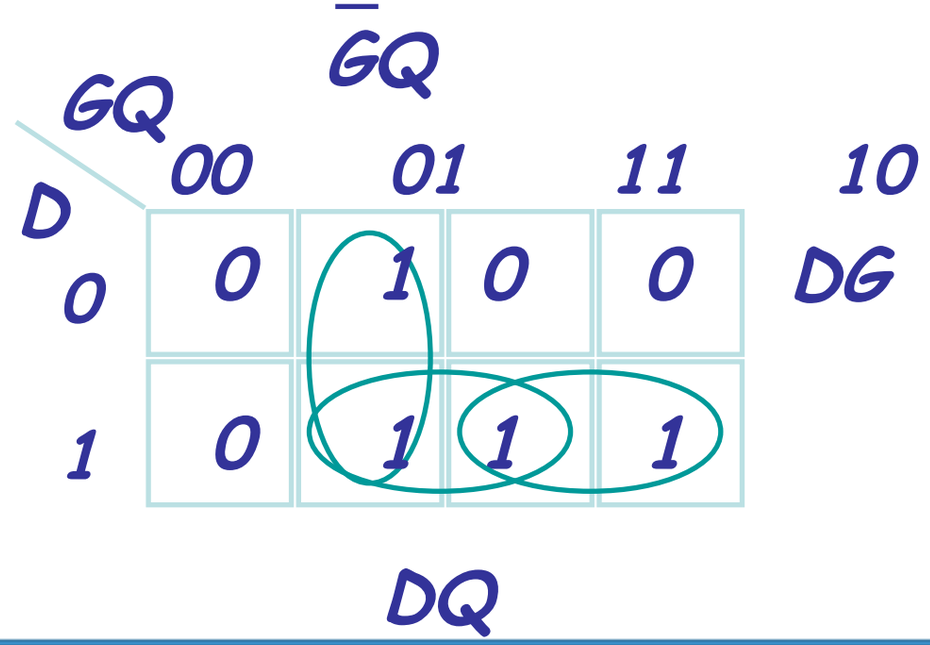
D \ $\overline{G}Q$	GQ	00	01	11	10
0		0	1	0	0
1		0	1	1	1

DG

Fixing Hazards

- Add patches to cover gaps in K-map.
  - Now, OR gate output is held high by DQ when crossing from DG to GQ.
- Assumes *hazard-free (glitch-free) gates.*
  - No glitches due to irrelevant inputs.
- No glitches due to SIC.
  - e.g., OR output stable at 1, as long as at least 1 input is 1 – regardless of how the other 2 inputs change.

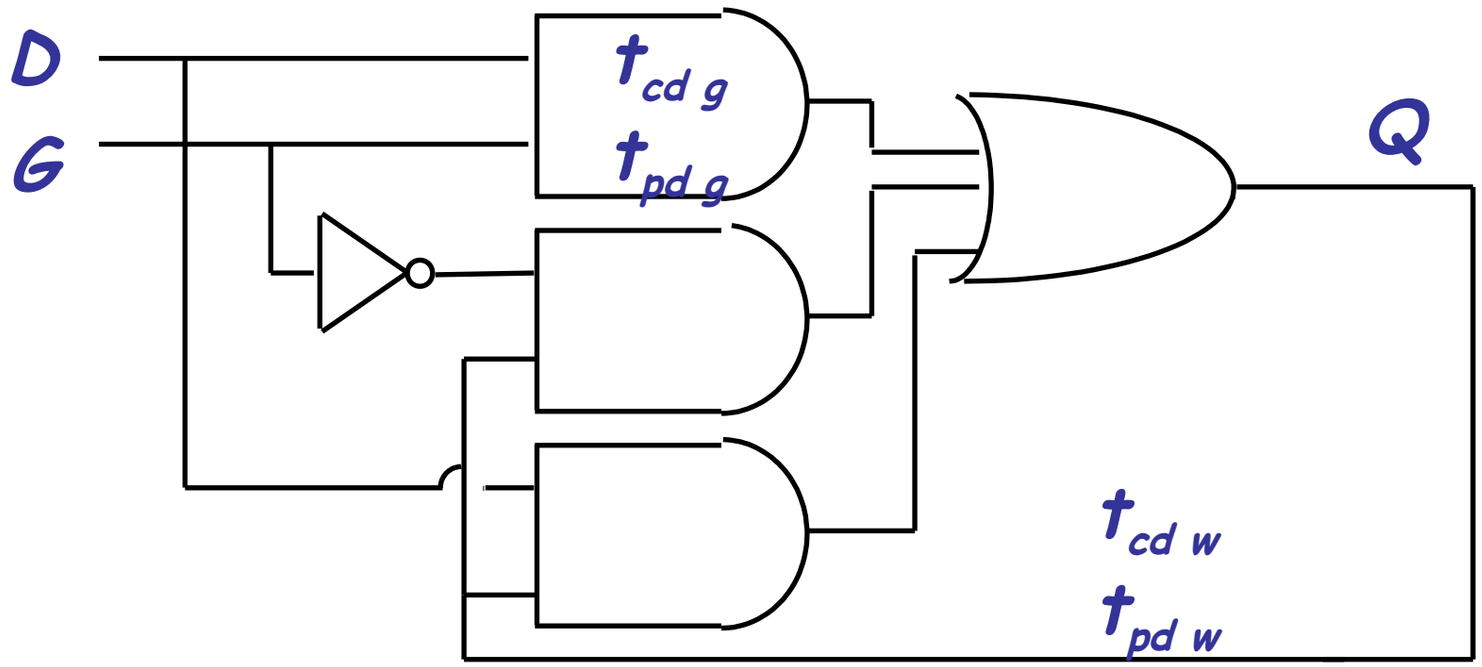




Hazards Synopsis

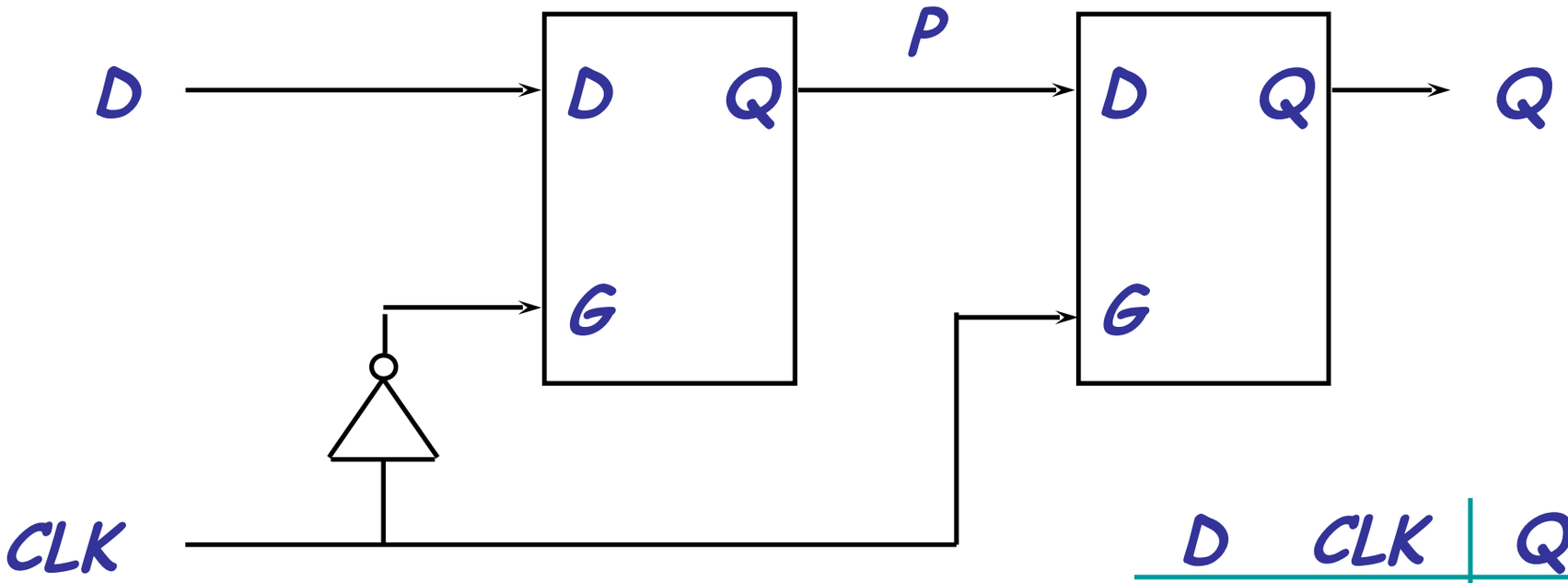
- **Hazards:** "Glitches" in output behavior *even if we follow SIC rule*.
  - If SIC is not followed, then glitches are hard to avoid because of race conditions (think of the inputs as starting positions).
  - Hazards are glitches that occur *even if only one input is changed*.
- **Problem:** If glitches are part of feedback loop, then bad values propagate!
  - Results include oscillations, metastability, wrong values, etc.
- **Solution:** Cover gaps in K-map.
  - Add gates to connect all patches .
  - Protects against SIC hazards only.

Reality Bites (Again)



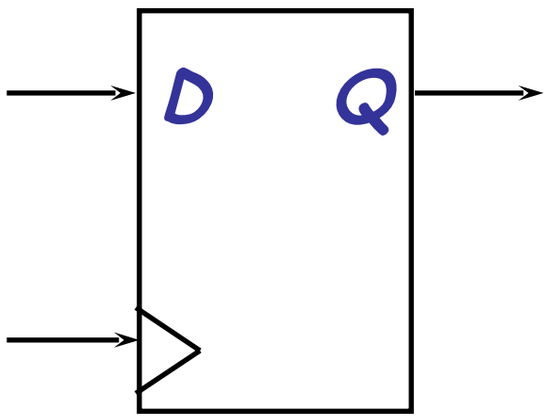
- In reality, setup and hold times are not only dependent on contamination and propagation delays of gates, but also of wires.
- Exhaustive transistor-level simulation is required to determine setup and hold times accurately – But we won't worry about that.

Edge-Triggered Flip-Flop



D	CLK	Q
{0, 1}	0	Q_{old}
{0, 1}	1	Q_{old}
D	↑	D
{0, 1}	↓	Q_{old}

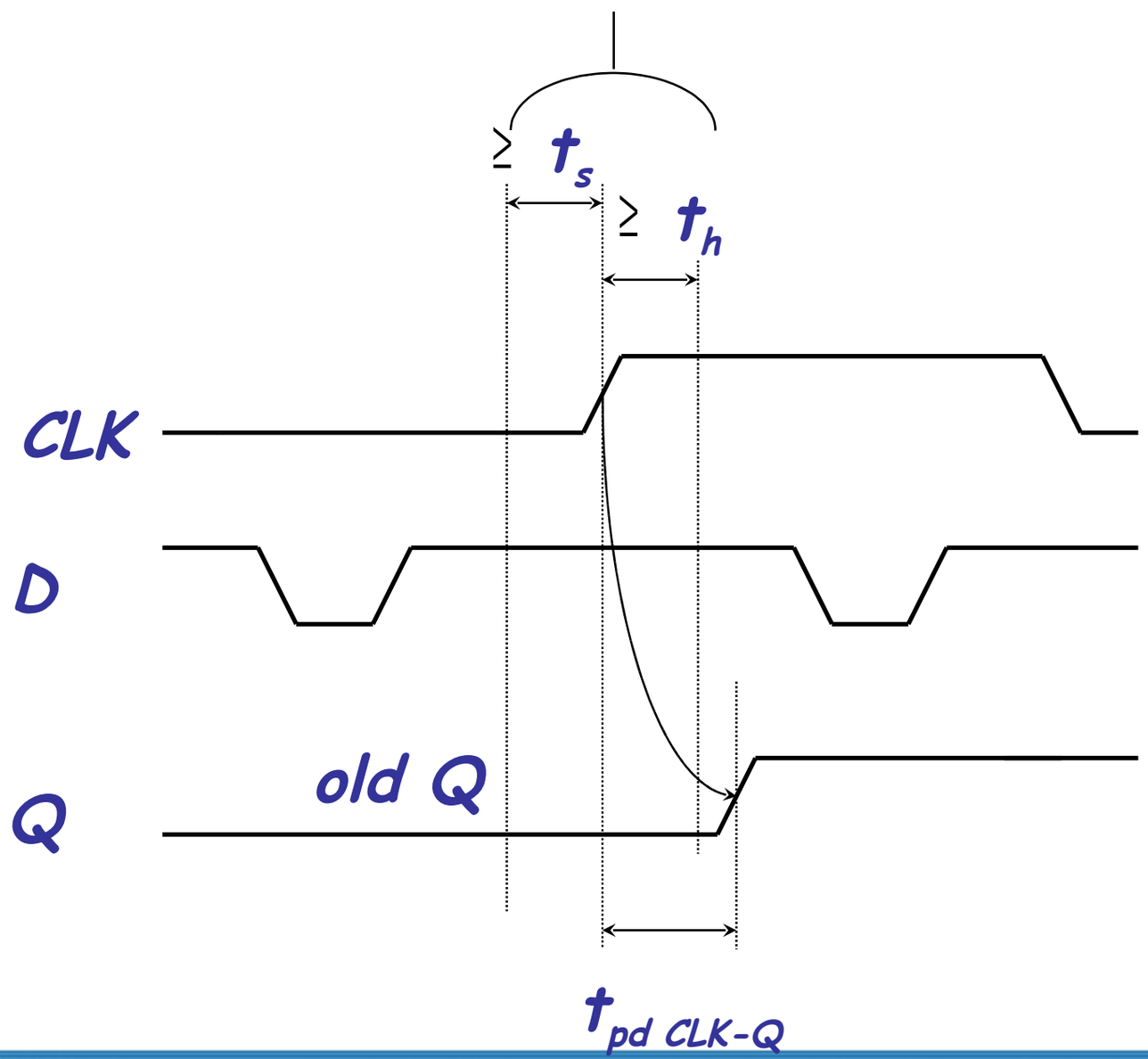
Symbol:



aka register or "Positive-edge-triggered FF"

Edge-Triggered Flip-Flop

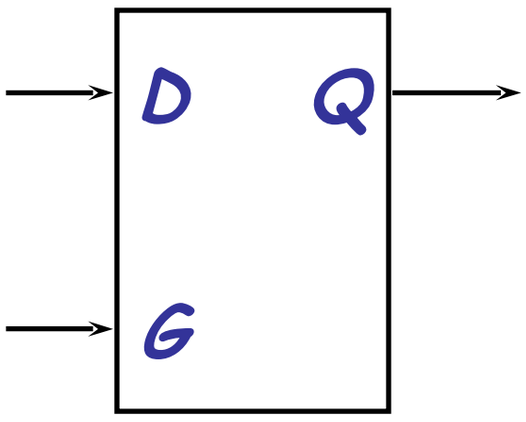
D must not change in this region



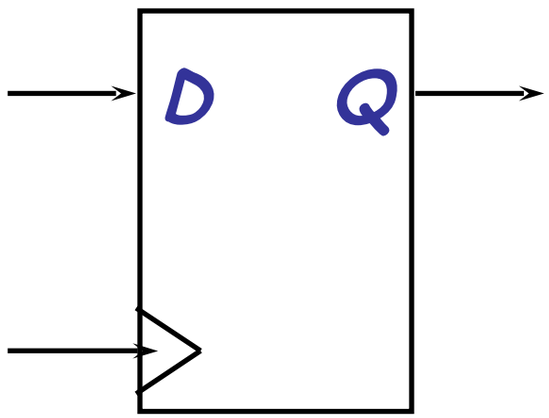
Setup time = t_s

Hold time = t_h

Latches and Flip-Flops

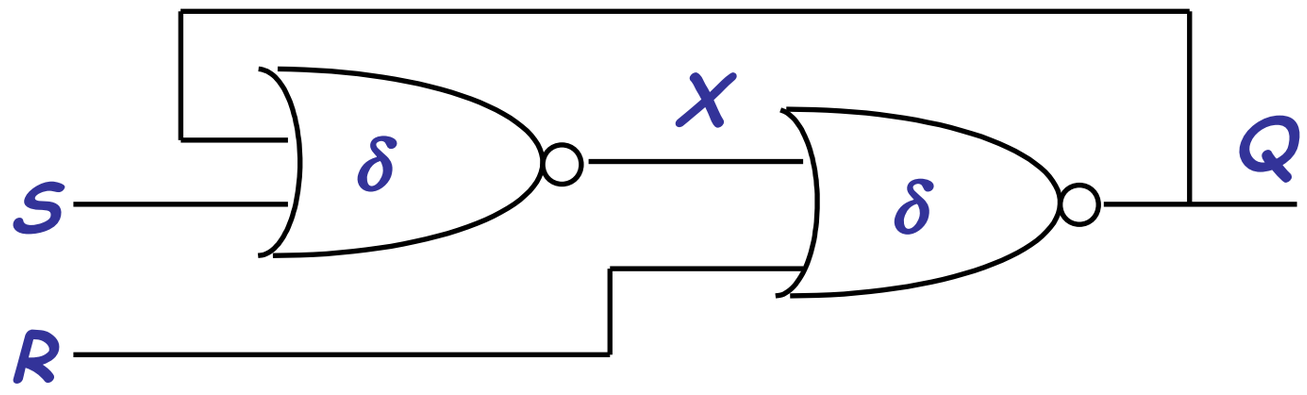


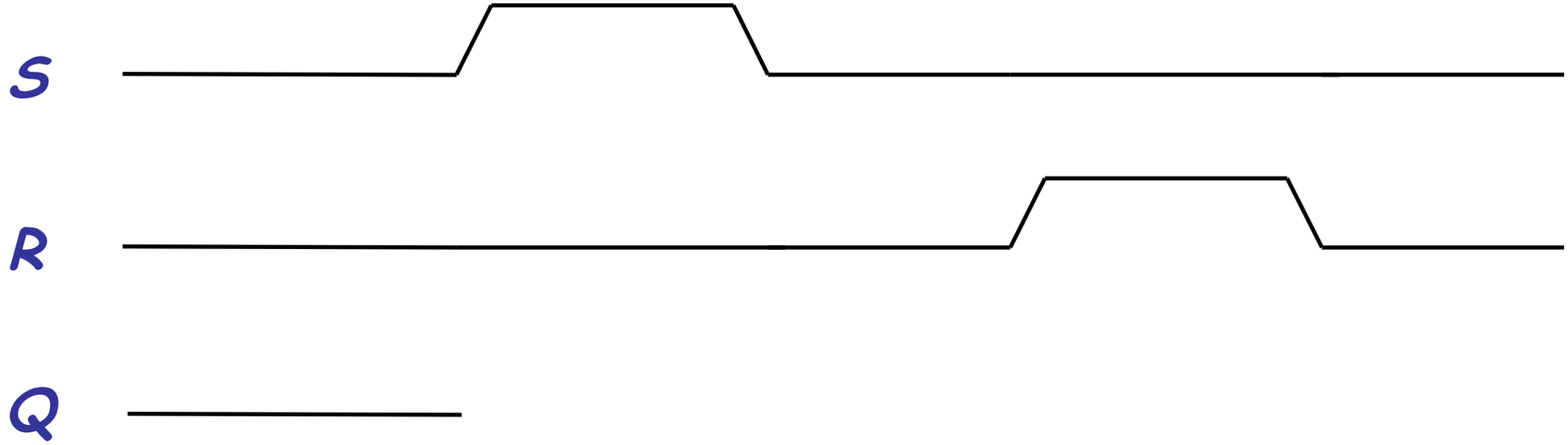
D-Latch: like a toll-gate, transparent when G=1.



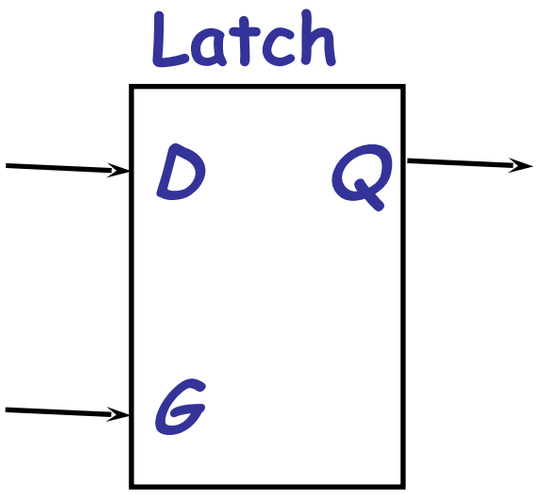
FF: like an air-lock, never transparent except at clock edge.

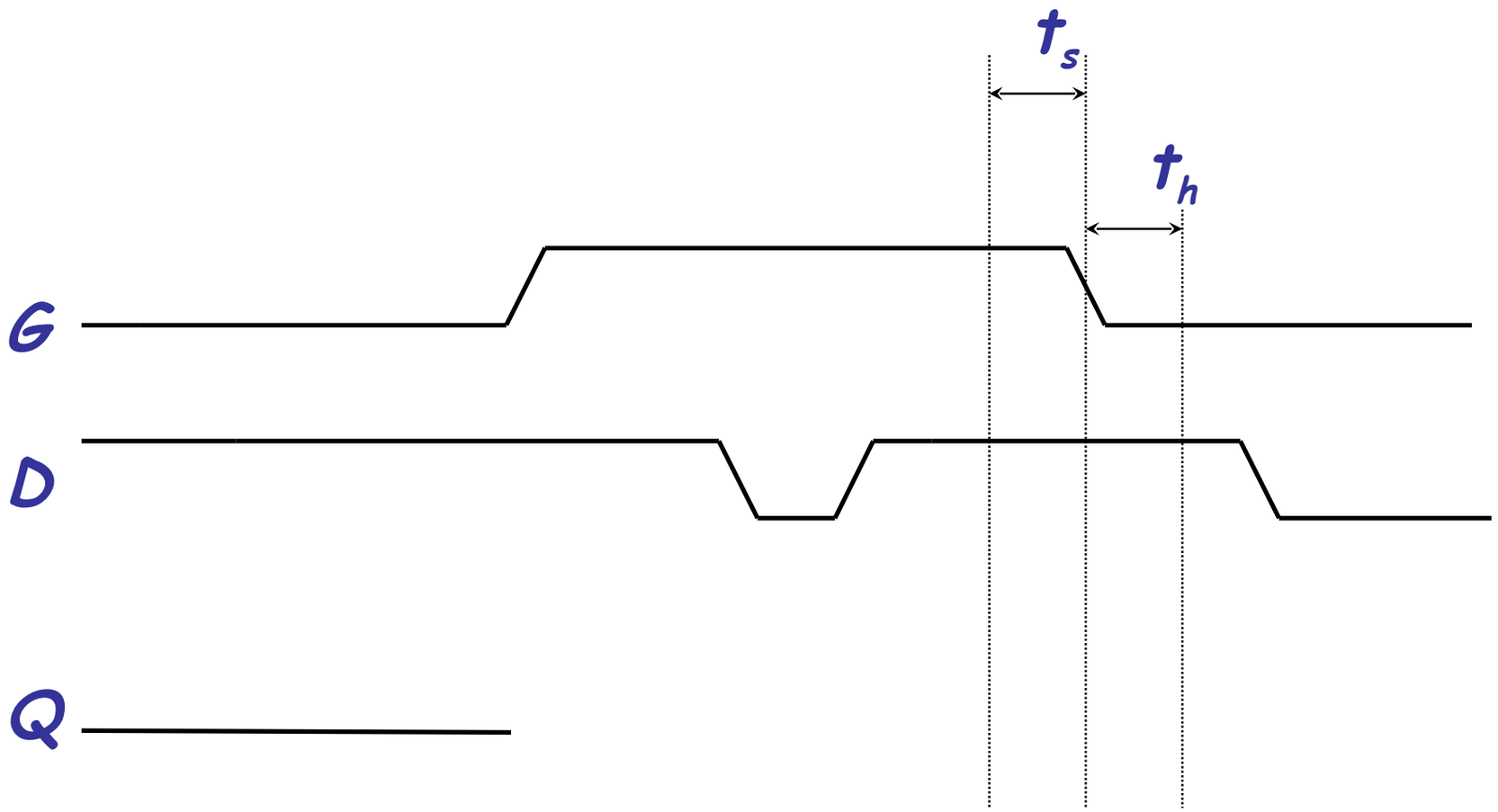
S-R Latch



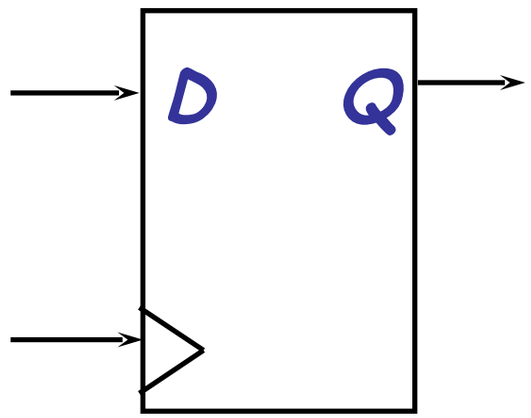


D-Latch





Edge-Triggered Flip-Flop



Setup time = t_s

Hold time = t_h

Clk-Q prop. = t_{pdC-Q}

Clk-Q cont. = t_{cdC-Q}

Assume that setup and hold times are satisfied!