Latches and Flip-Flops

Hanwool Jeong

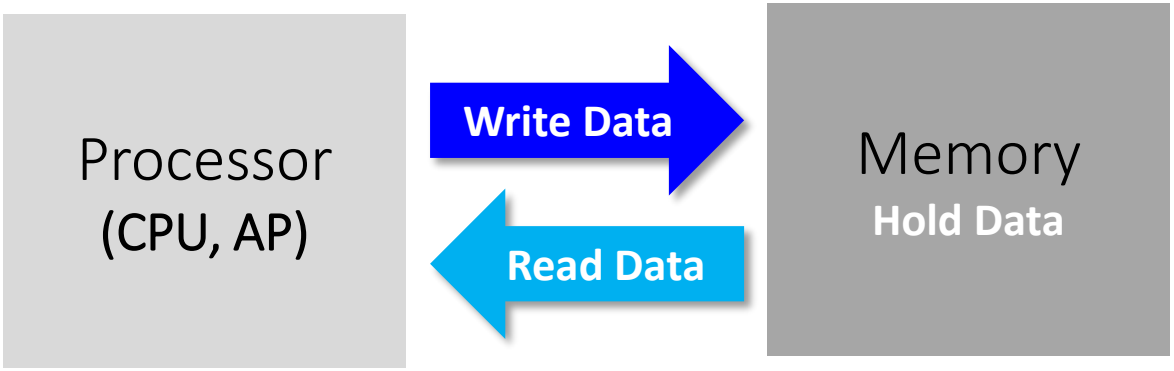
hwjeong@kw.ac.kr

Contents

- Brief introduction to memory
- Conventional latches and flip-flop circuits & issues
- Improved flip-flop circuits

Memory Operation

- Processor calculates
- Memory can **hold/read/write** data



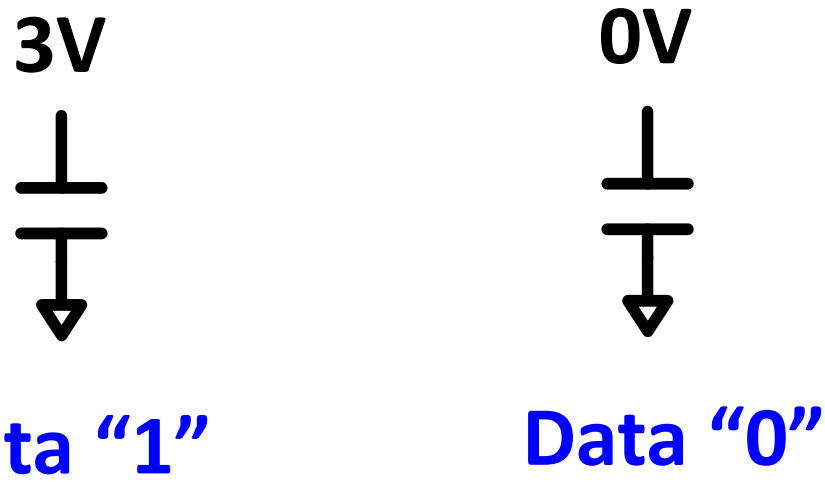
Form of Binary Data

- Memory element stores data in binary form : **“0” and “1”**
- Memory should be able to be in **two electrically separable states**.

➔ **Memory element should hold/write/read two possible states**

Let's Make A Memory Element
: Hold Two Possible States

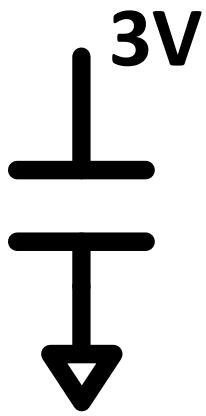
- Remember: Memory should **hold**/write/read **two states**
- How can we implement two electrically distinguishable states in the digital circuit?



➔ **Can we use a capacitor as a memory? How about writing data?**

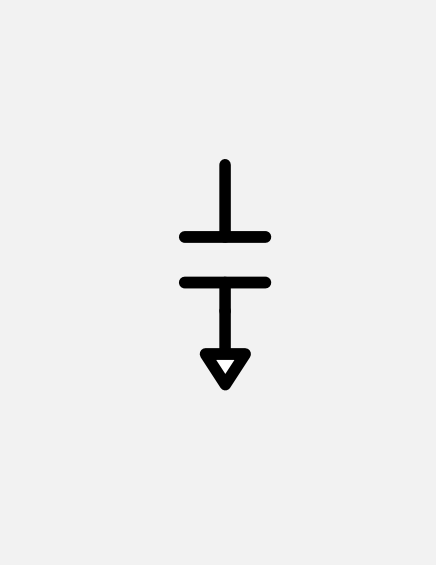
Problem of Capacitor Memory

- Can you change the following state into data "0"? ➔ **NO**

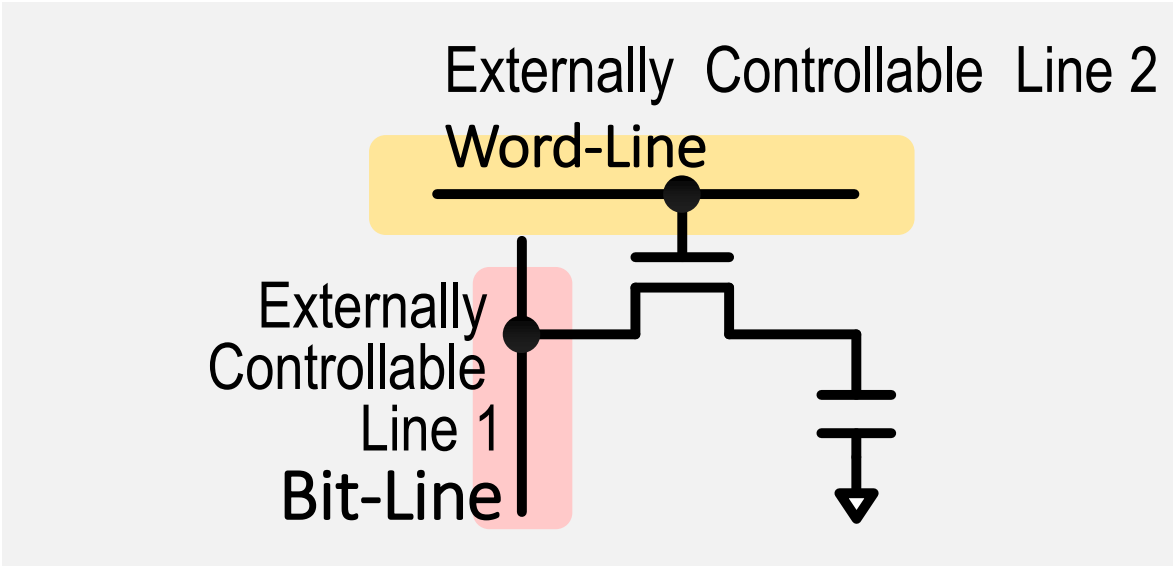


Data “1”

Addition of “Switch” for Read/Write



Capacitor

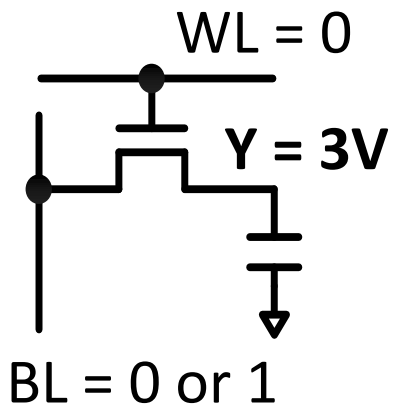
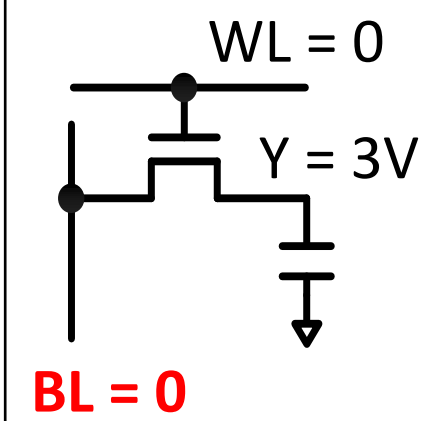
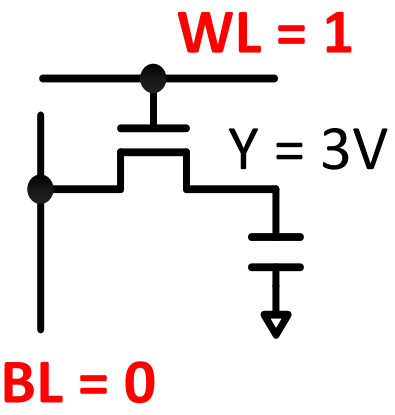
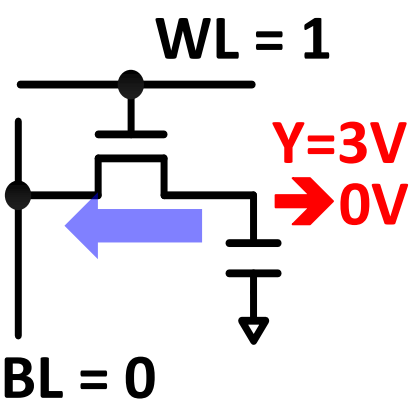
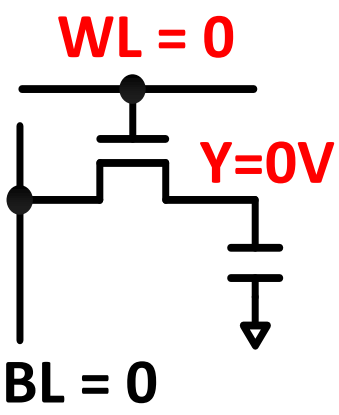


Capacitor

+ nMOSFET w/
Two externally controllable lines

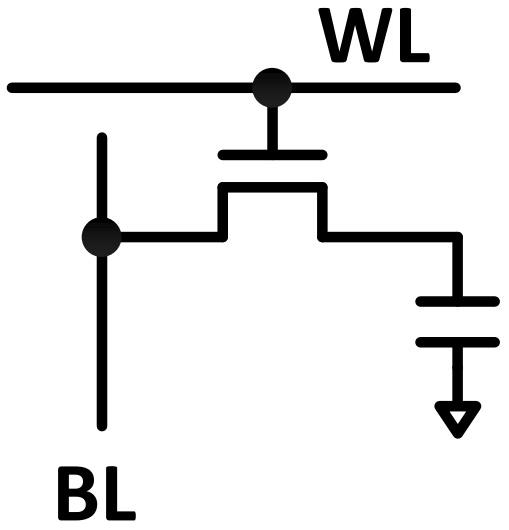
Let's Make A Memory Element
: Write Two Possible States

- Let's write data "0" to memory bitcell initially storing "1"

				
Hold data "1"	Driving BL to "0"	Driving WL to "1"	Y is discharged	**Hold data "0"**

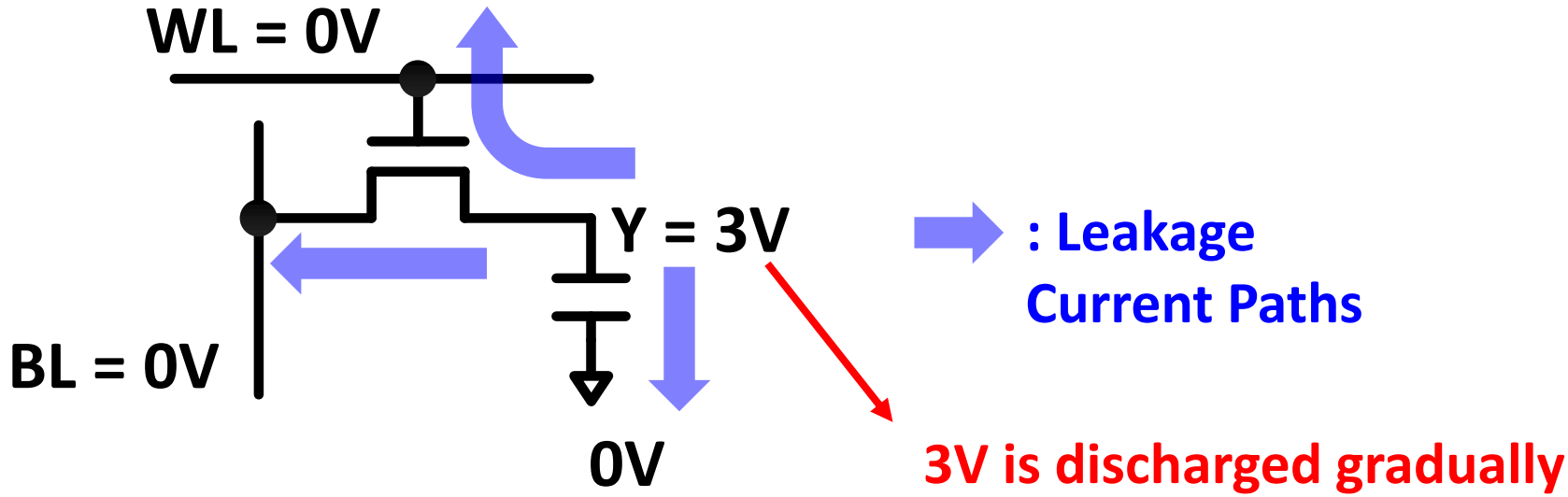
1T1C DRAM Bitcell

- Dynamic Random Access Memory (DRAM) Bitcell



Limitation of DRAM

- Suppose that data “1” is stored in the DRAM bitcell
- What if there is a leakage current or Noise?
  ➔ **Data can be “dynamically” changed**



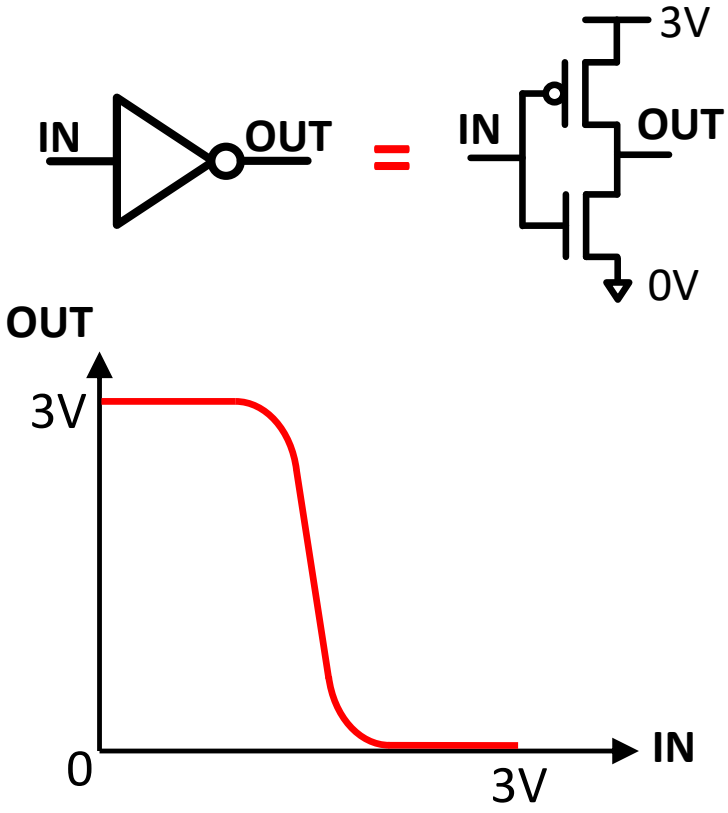
Can We Solve the Problem?

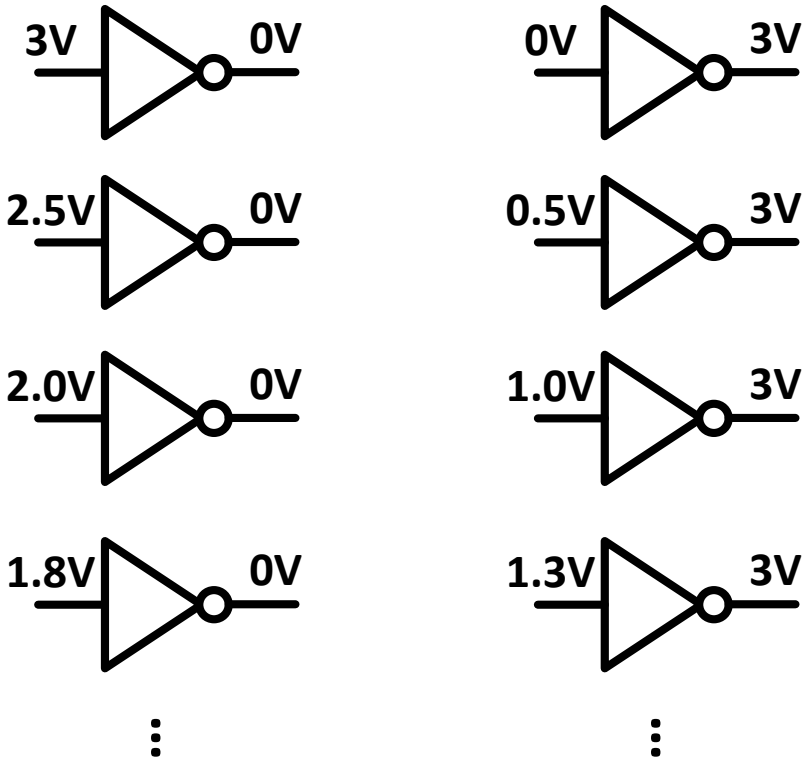
- How about using digital logic?
- It can yield “0” or “1” logic by 0V and 3V, respectively.
- But the input should be provided to realize it.

➔ How can we handle this?

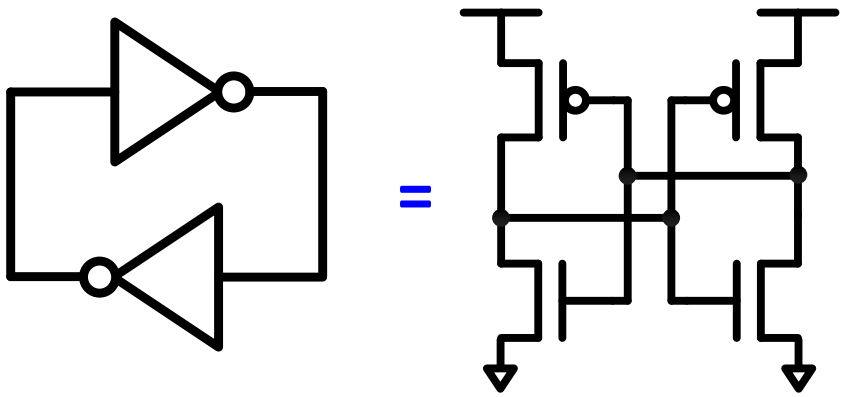
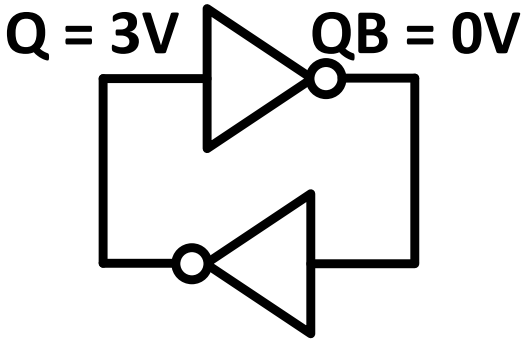
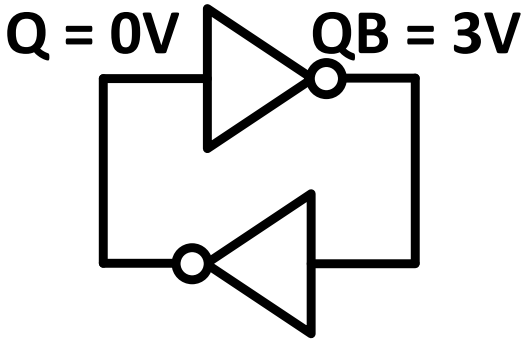
Revisit
Regeneration Feature of Inverter

- While the input can have arbitrary values, output can become 3V or 0V



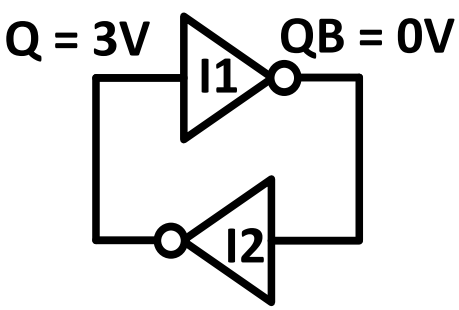
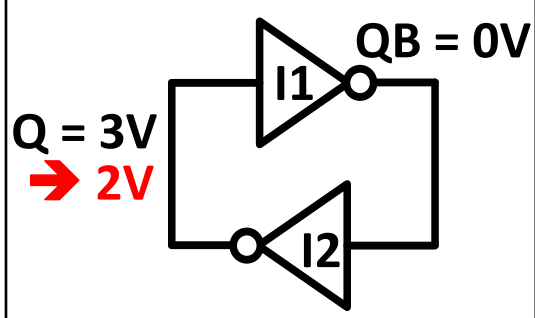
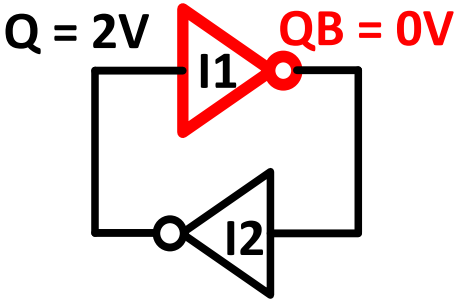
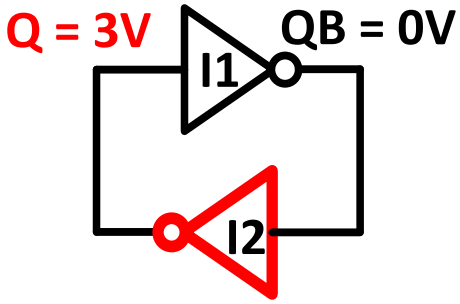


Cross-coupled Inverter as A Memory

Structure	Data States	
	 Data “1”	 Data “0”

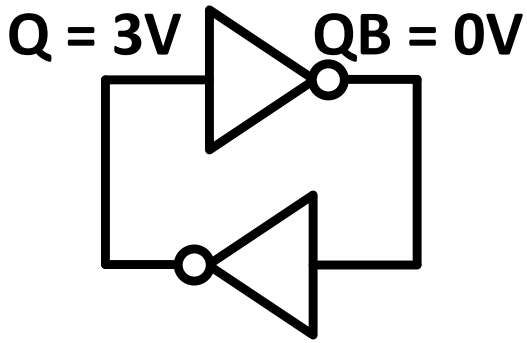
Robustness of Cross-Coupled Inverters

- Cross-coupled inverter is free from the problem which DRAM suffers from

			
Hold data “1”	Q is accidentally lowered to 2V	QB is kept 0V because of regeneration of I1	Q1 is back to 3V because of regeneration of I2

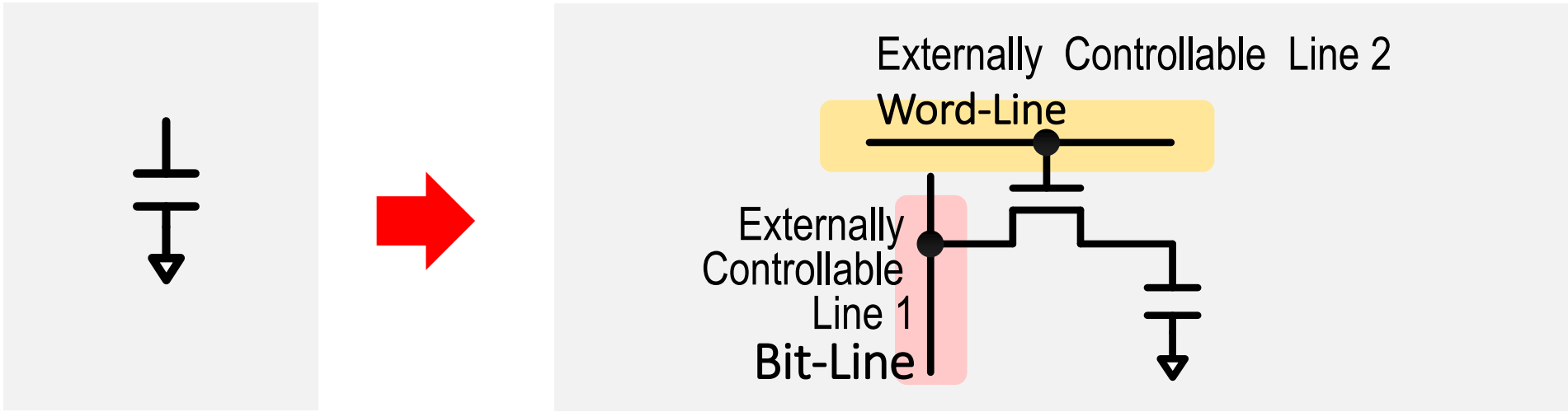
Cross-coupled Inverters as Memory?

- Can you change the following state into data "0"? ➔ **NO**



Data “1”

Revisit: What We Did for DRAM Bitcell

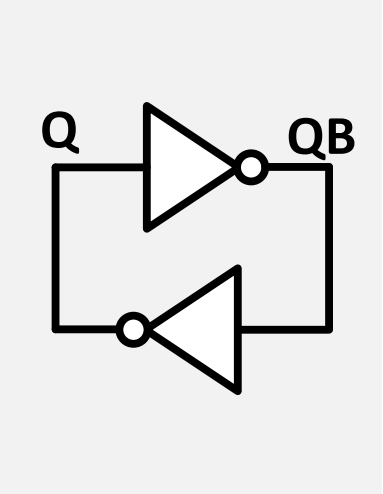


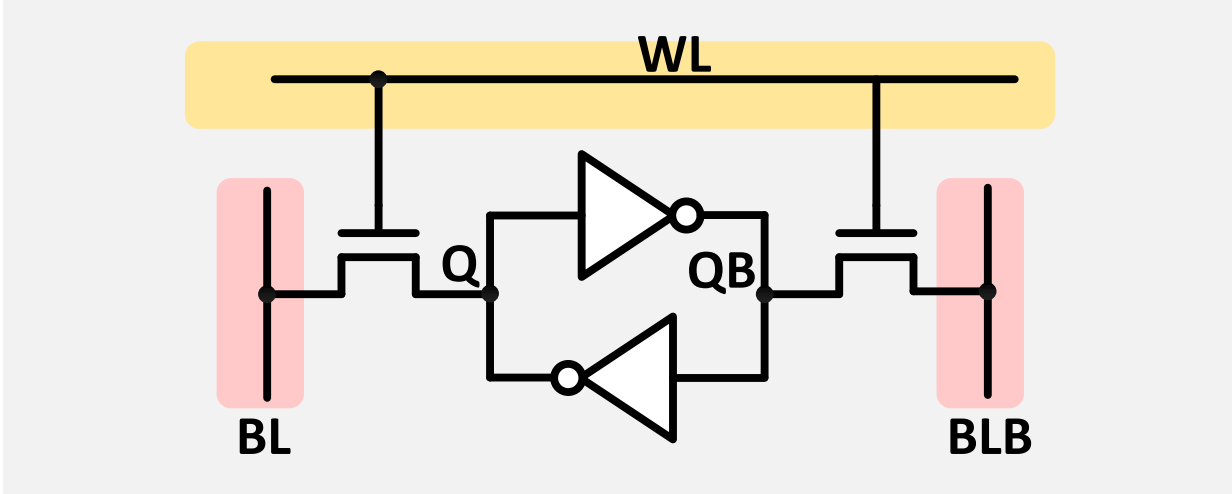
Capacitor

Capacitor

+ nMOSFET w/
Two externally controllable lines

Enabling Read/Write Operation to Cross-Coupled Inverters



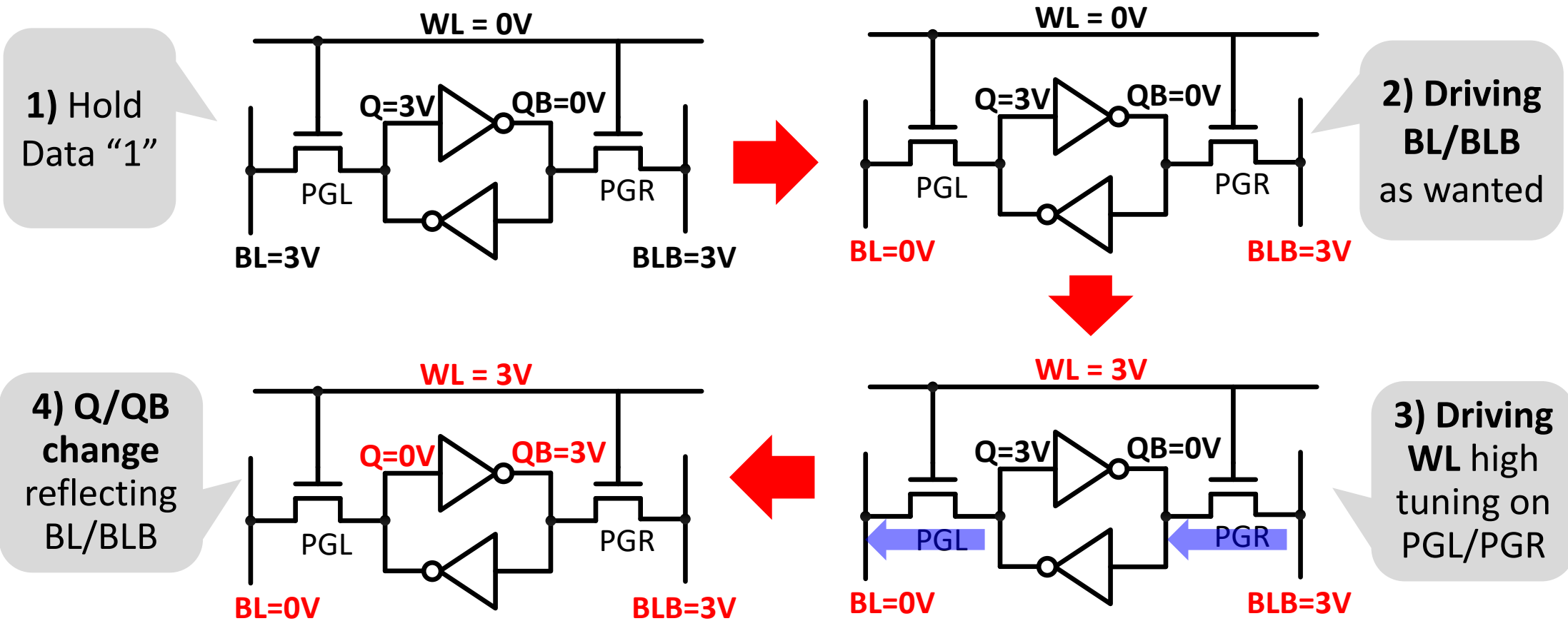


Cross-Coupled Inverter

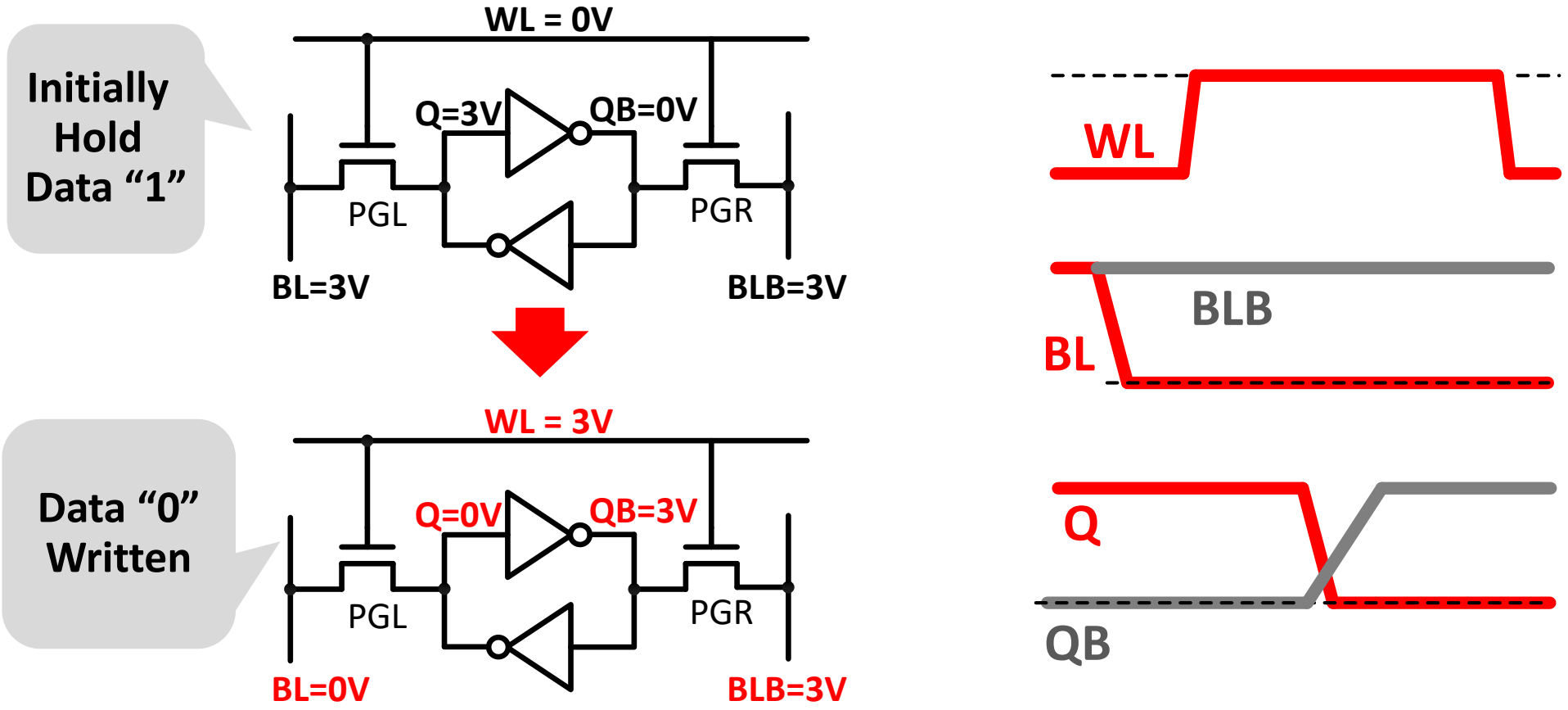
Cross-Coupled Inverter

+ Two nMOSFET w/
2+1 externally controllable lines

Write Operation

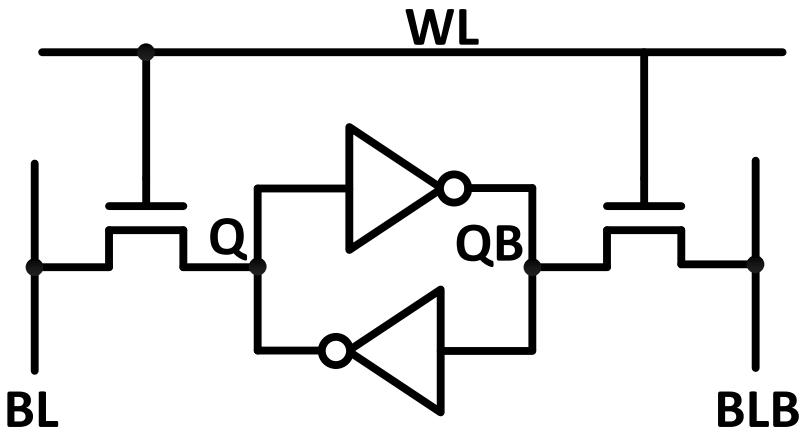


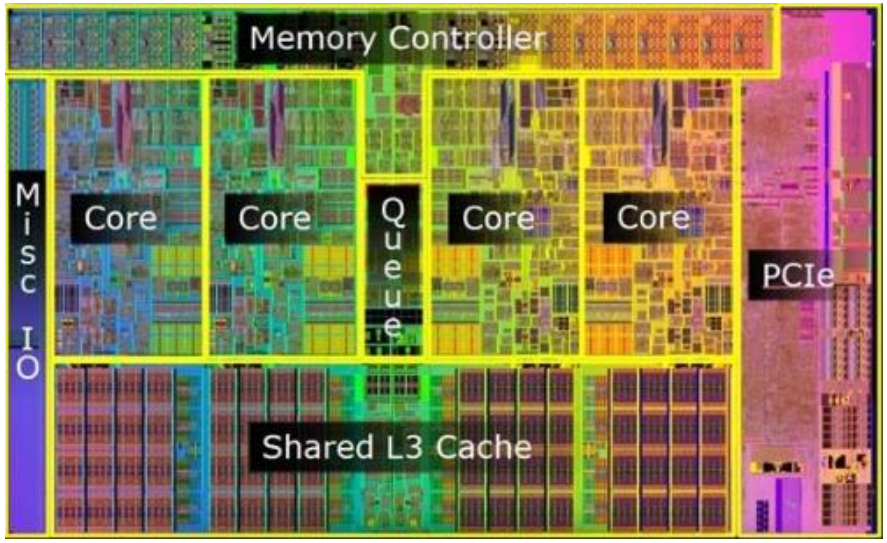
Write Operation Waveform



6T SRAM Bitcell

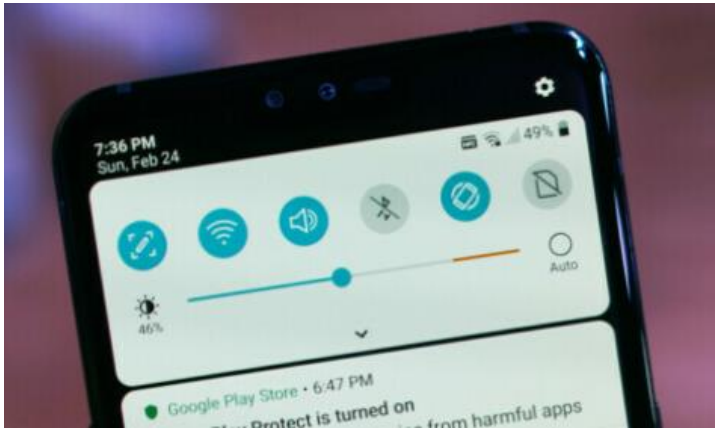
- Static Random Access Memory (SRAM) Bitcell
- “Static” means invariant memory storage operation over time

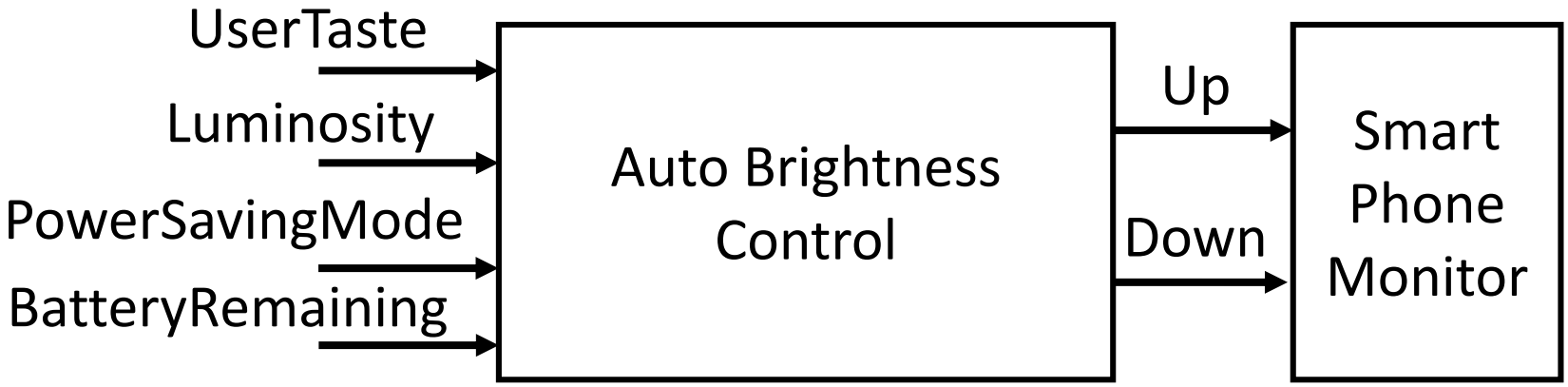




Need for Sequential Circuit & State; Smartphone Auto Brightness Control

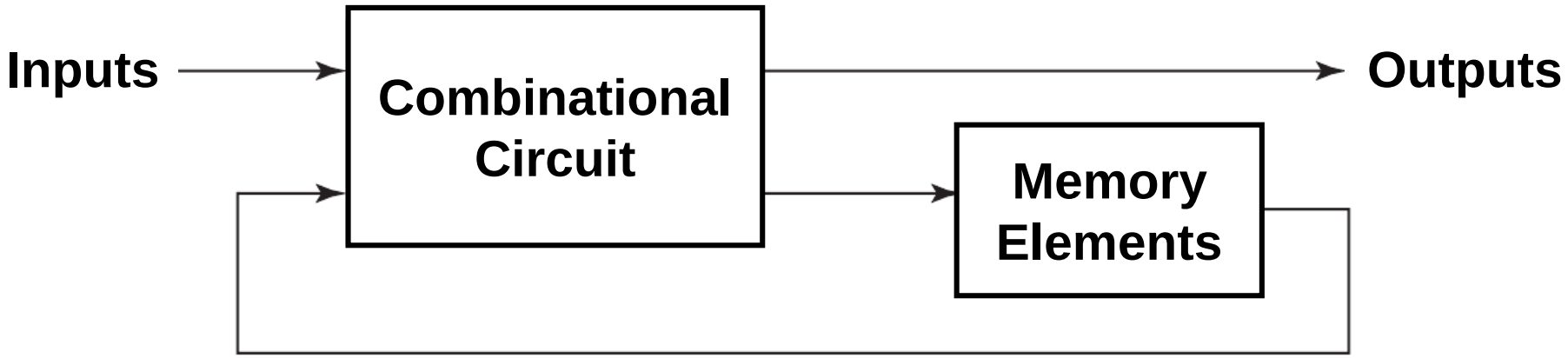
- We need clock (synchronous) and states (sequential).





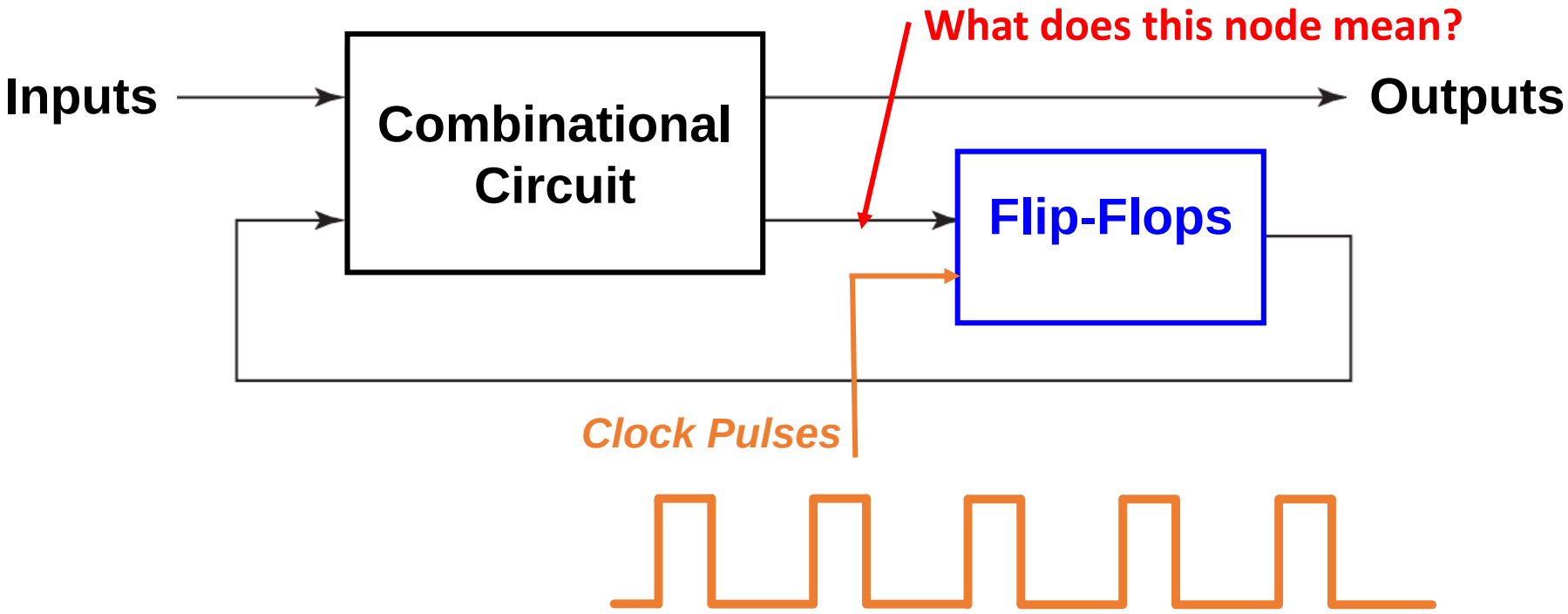
Sequential Circuit

- State is stored in memory, which is to be used at later time.
- Inputs/States/Outputs are all formed in binary.



Synchronous Sequential Circuit

- Storage elements in clocked sequential circuits are called flip-flops, which is a binary storage device storing one bit.
- Flip-flop updates the state every rising edge of clock. Then, the state is feedback to the combinational logic.

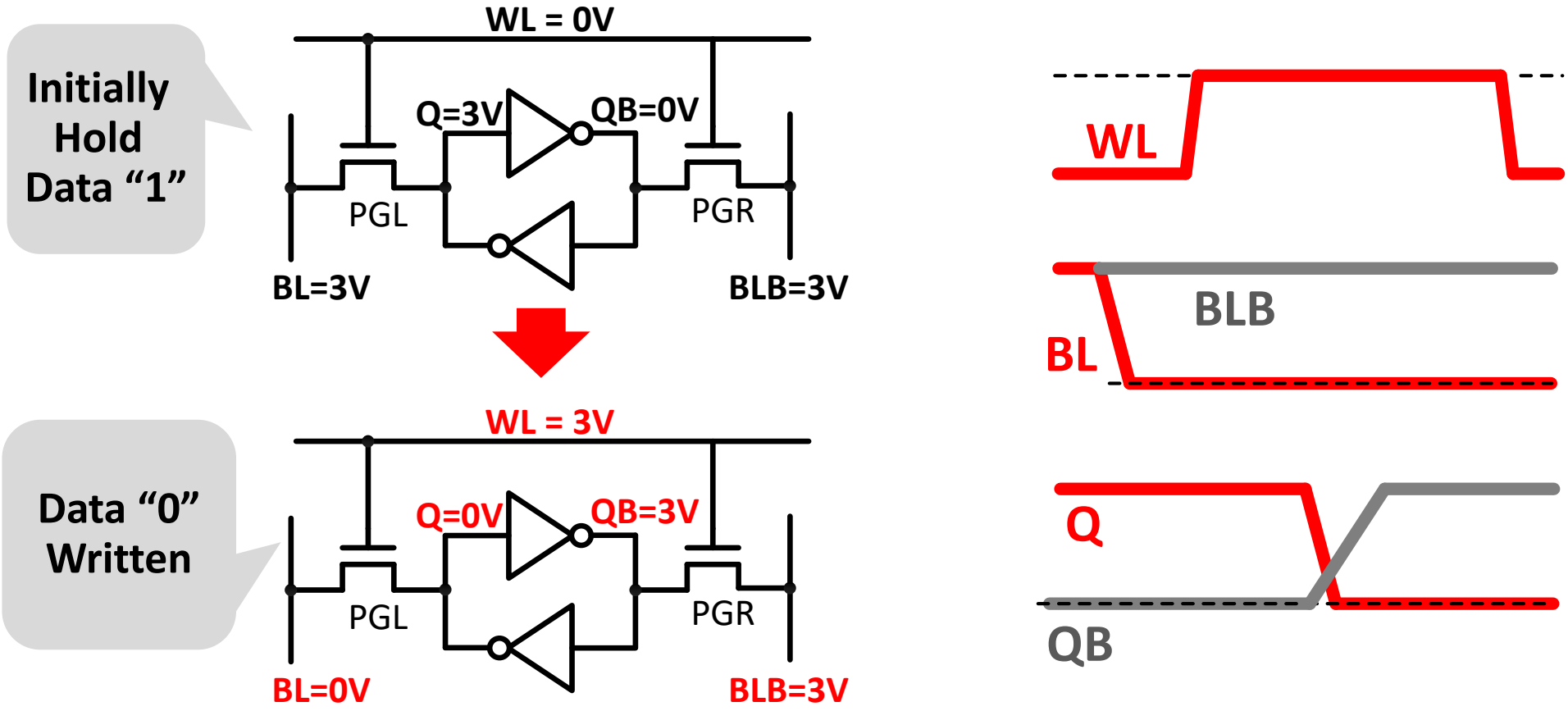


How Can We Store States?

- States should be used mapped into binary numbers.
- If we have 50 possible states, then we should allocate 6 bits to represent a state. That is, we need six flip flops.
- Then, how does flip-flop or memory elements for storing state look like?
- It should be able to perform three main operations
  - The state should be able to be **stored (hold).**
  - The state should be able to be **changed** to what is intended **(write)**
  - The state should be able to be **detected (read).**

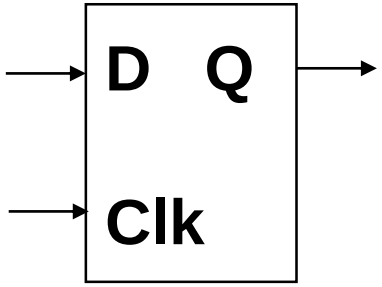
Revisit SRAM Write Operation

- It requires the differential data lines and suffers from contention



Initial Thought & Latch

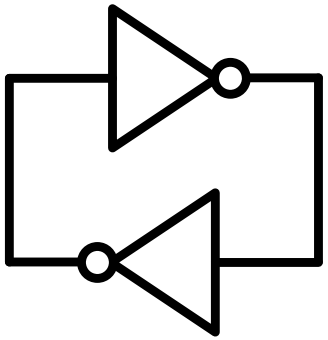
- Compared to SRAM which necessitates high density, a flip flop can be implemented with larger area.
- First, let's focus on implementing a memory element that is
  1) Free from the contention during data update.
  2) Writable only when clock is high.
- We will define three nodes for this memory element, and call it as a **latch**



1) Q is updated to D for Clk = 1
2) Q holds the old value for Clk = 0

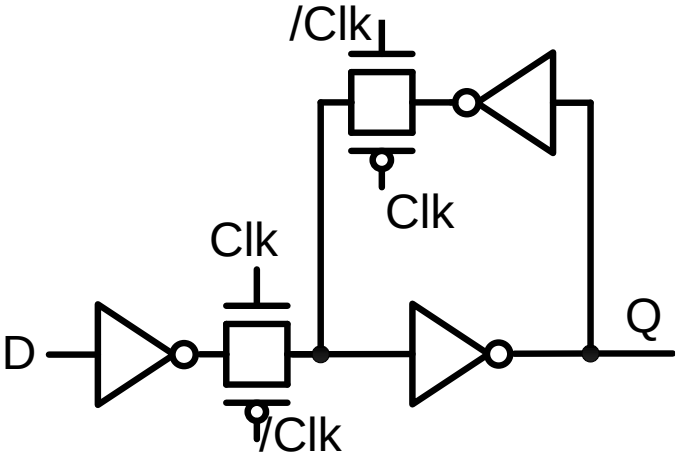
Let's Make A Latch

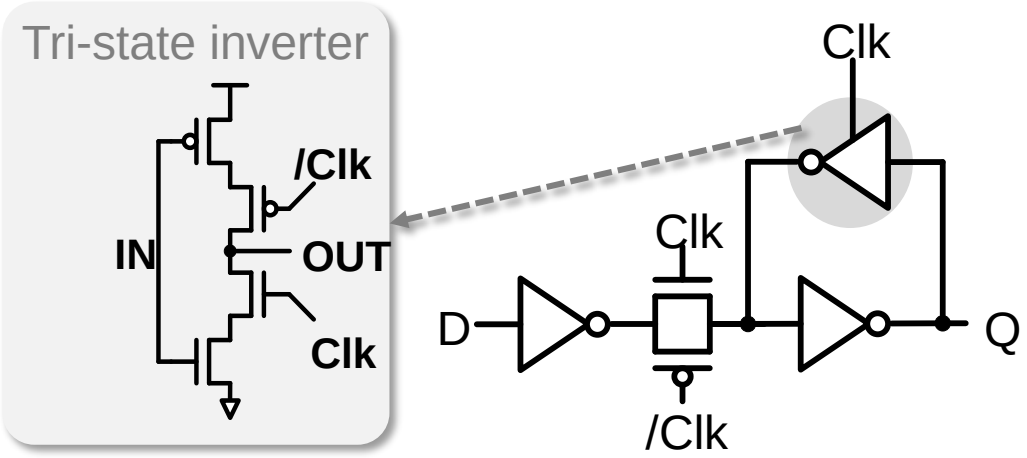
- How can we realize the following?
  1) Q is **updated** to D for Clk = 1
  2) Q **holds** the old value for Clk = 0

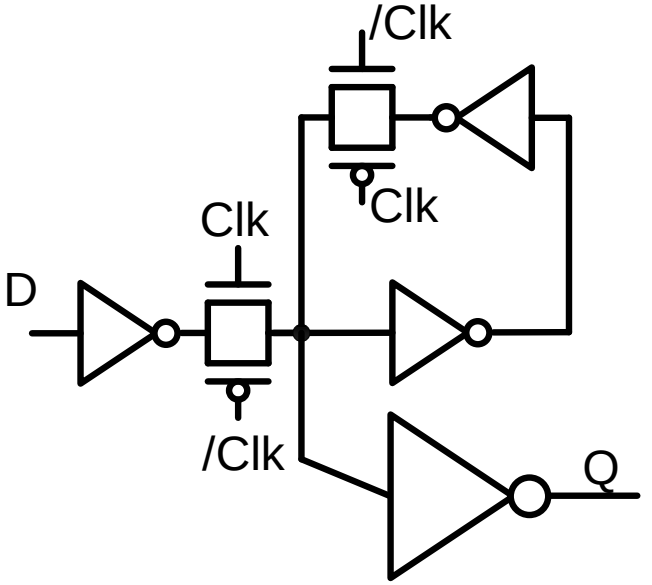


TG based D-Latch

- You can utilize tri-state inverter
- You can separately additionally output inverter to improve drivability.

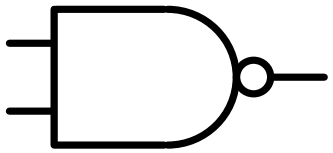


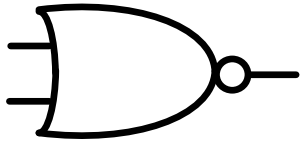




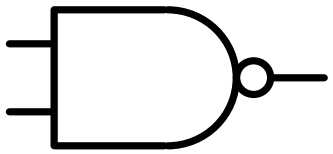
You Can Start From NAND2 or NOR2

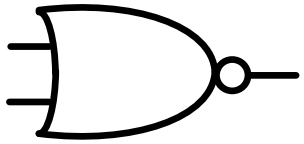
- NAND2 and NOR2 become inverter when





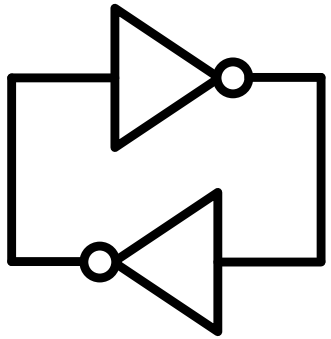
- NAND or NOR yield fixed output independently to other inputs when





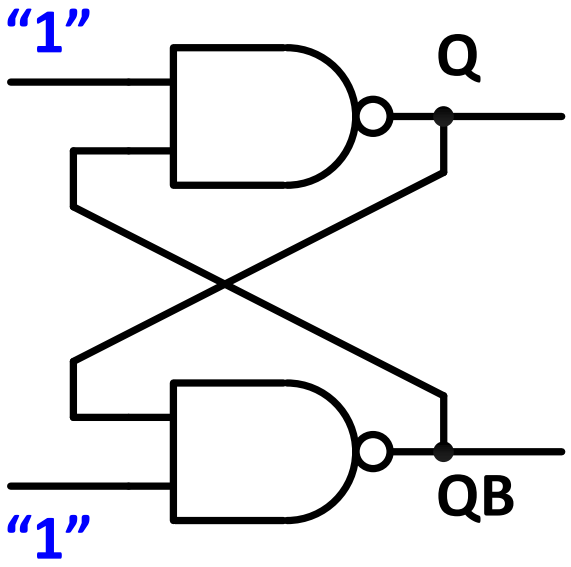
Cross-Coupled Inverters With NAND2 or NOR2

- For data hold, the cross-coupled inverters should be implemented.
- How can you configure the cross-coupled inverters using NAND2 or NOR2?

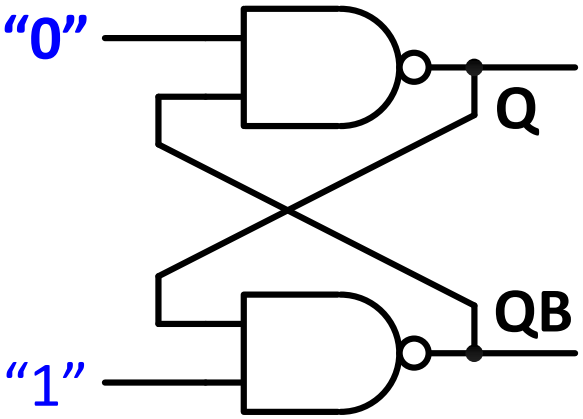


Data hold

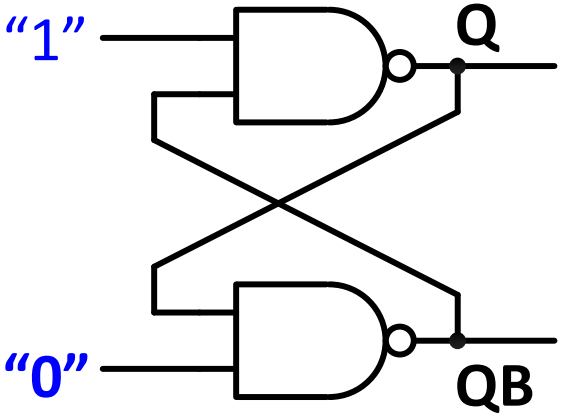
Holding & Updating Data



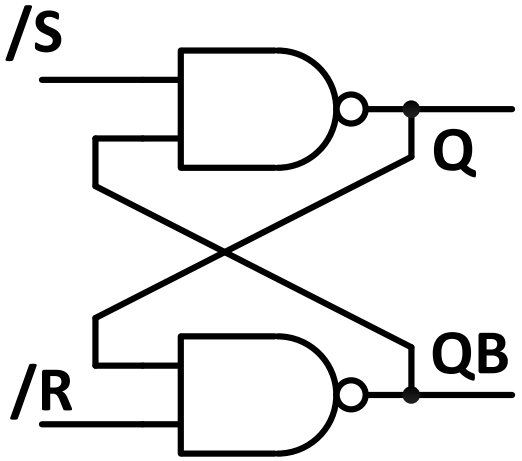
Holding Q & QB



Updating Q to 1



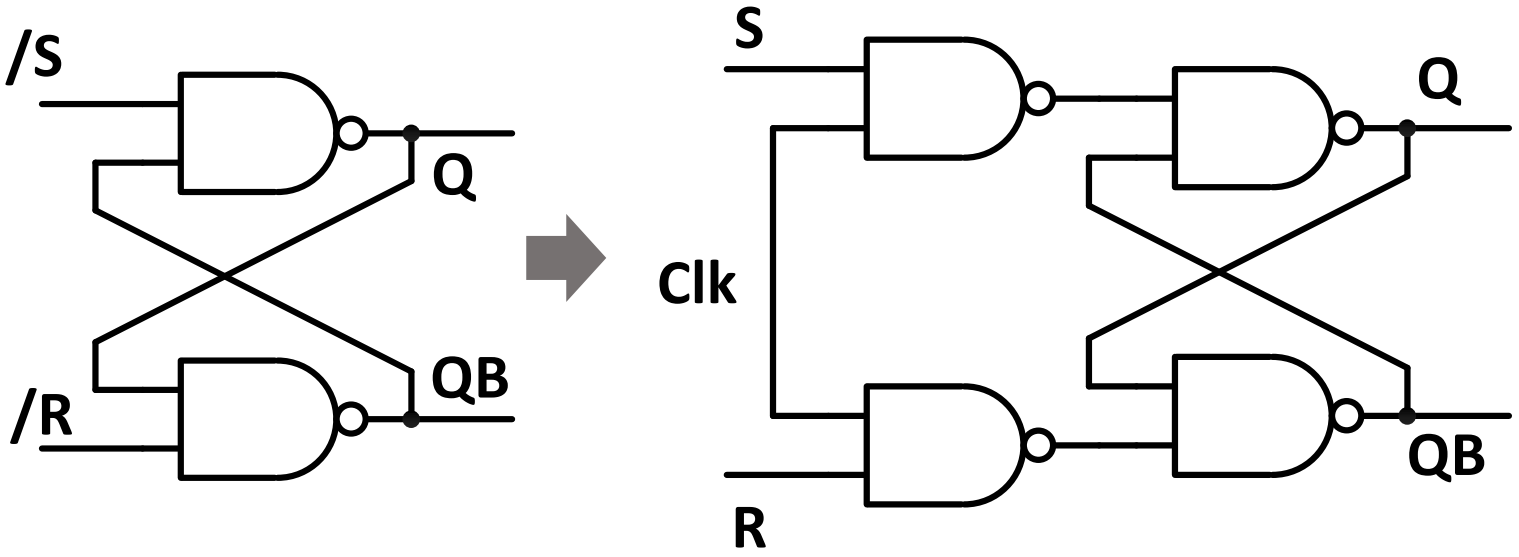
Updating Q to 0



/R	/S	
0	0	Forbidden
0	1	Q=0
1	0	Q=1
1	1	Hold

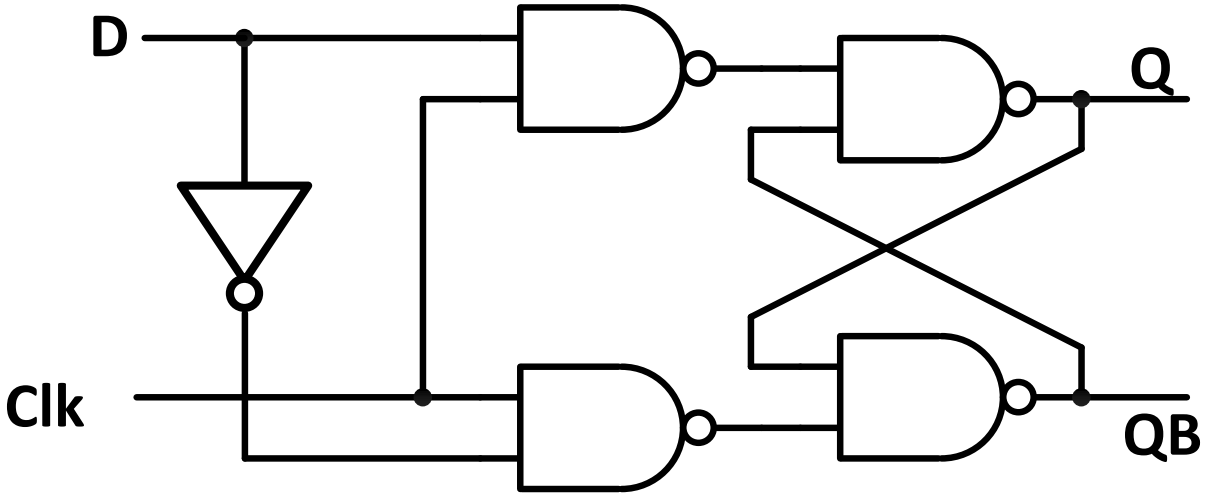
Updating Data Only When Clk = 1

- Focus on the target operation
  - Clk = 1 : Update (or we say this state as “transparent”)
  - Clk = 0 : Hold (or we say this state as “opaque”)



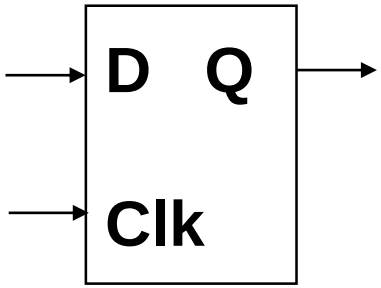
Clk	R	S	
1	0	0	Hold
1	0	1	Q=1
1	1	0	Q=0
1	1	1	Forbidden
0	X	X	Hold

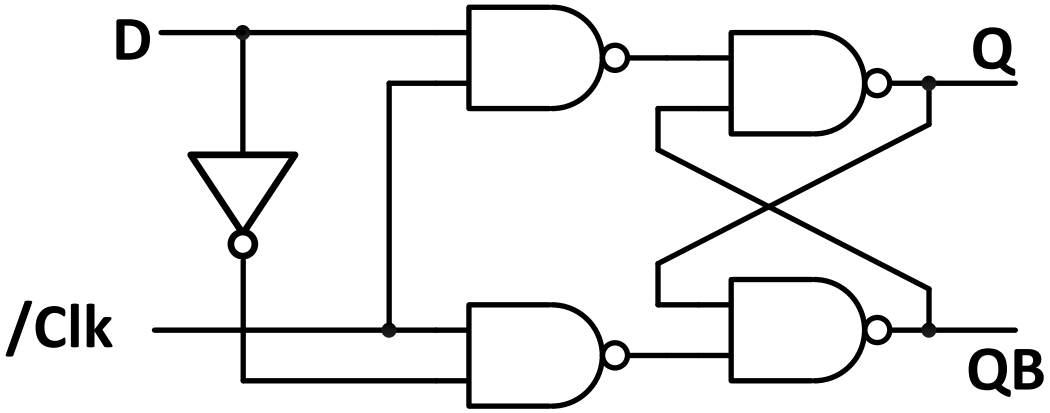
Preventing Forbidden State



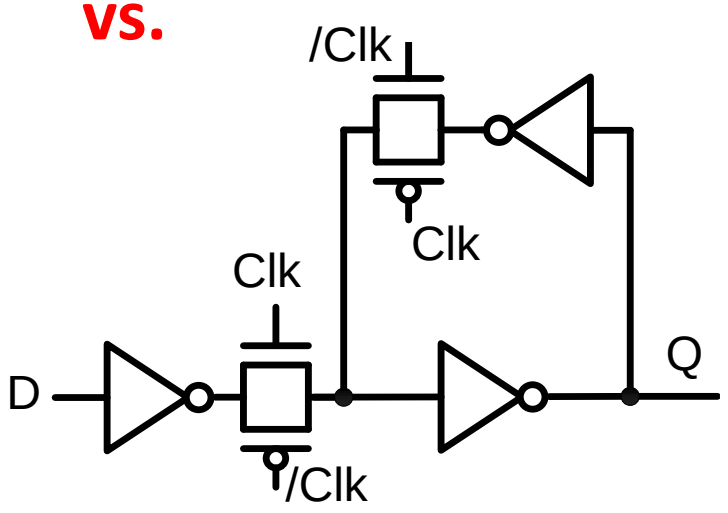
Clk	D	
0	0	Hold
0	1	Hold
1	0	Q=0
1	1	Q=1

Which Do You Prefer?

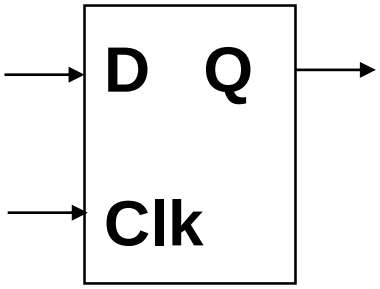


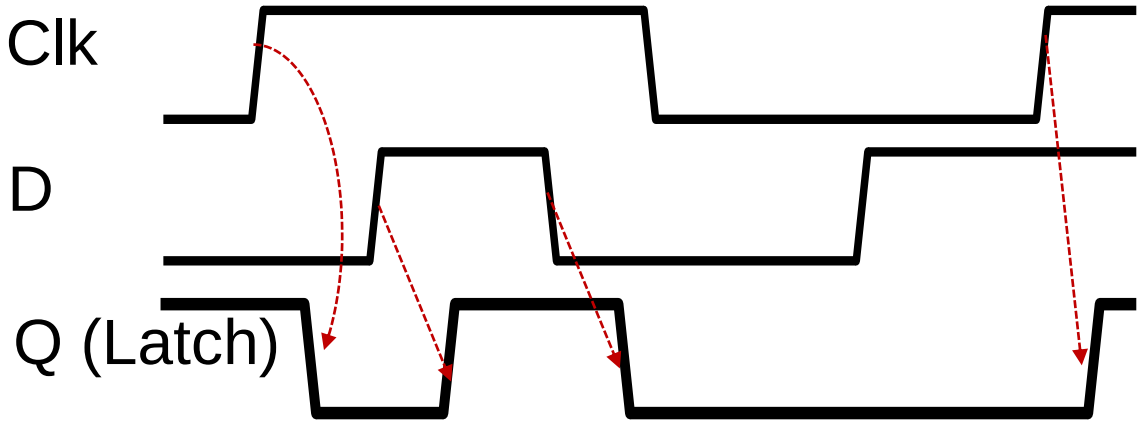


vs.



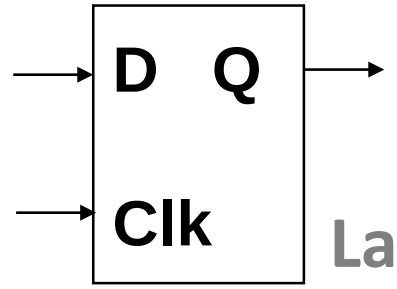
Operational Waveforms of Latch



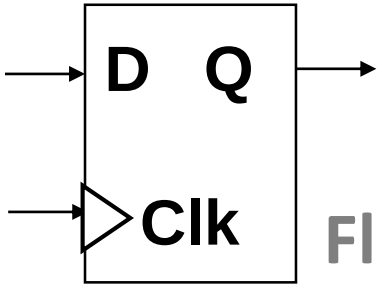


Latch to Flip-Flop

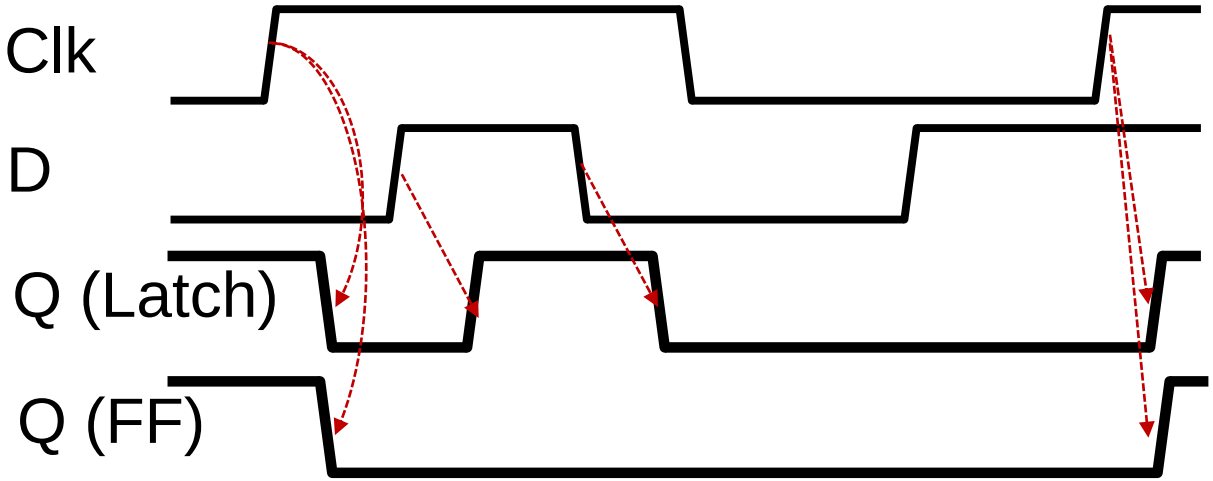
- Note that our target operation is to update D at the rising edge of clock, not Clk = 1.
- How can we implement this?



Latch



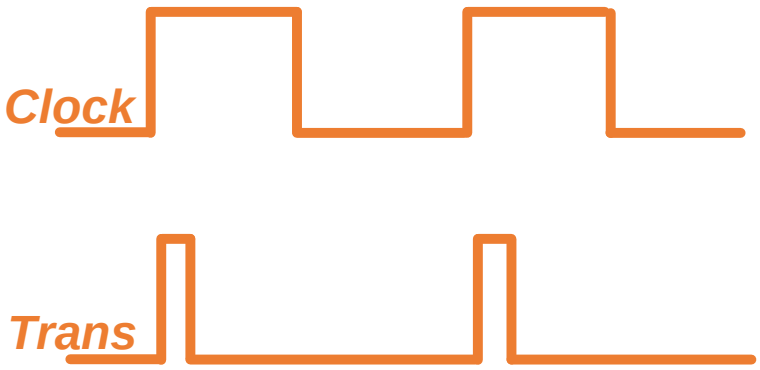
Flip-flop

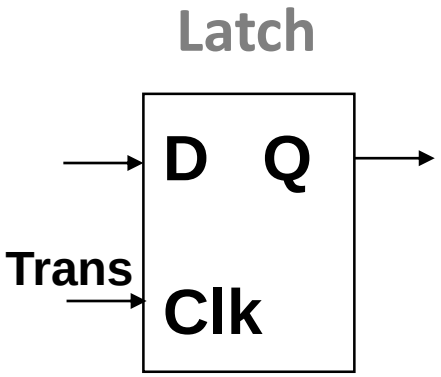


FF : D flows to Q only at clk rising edge

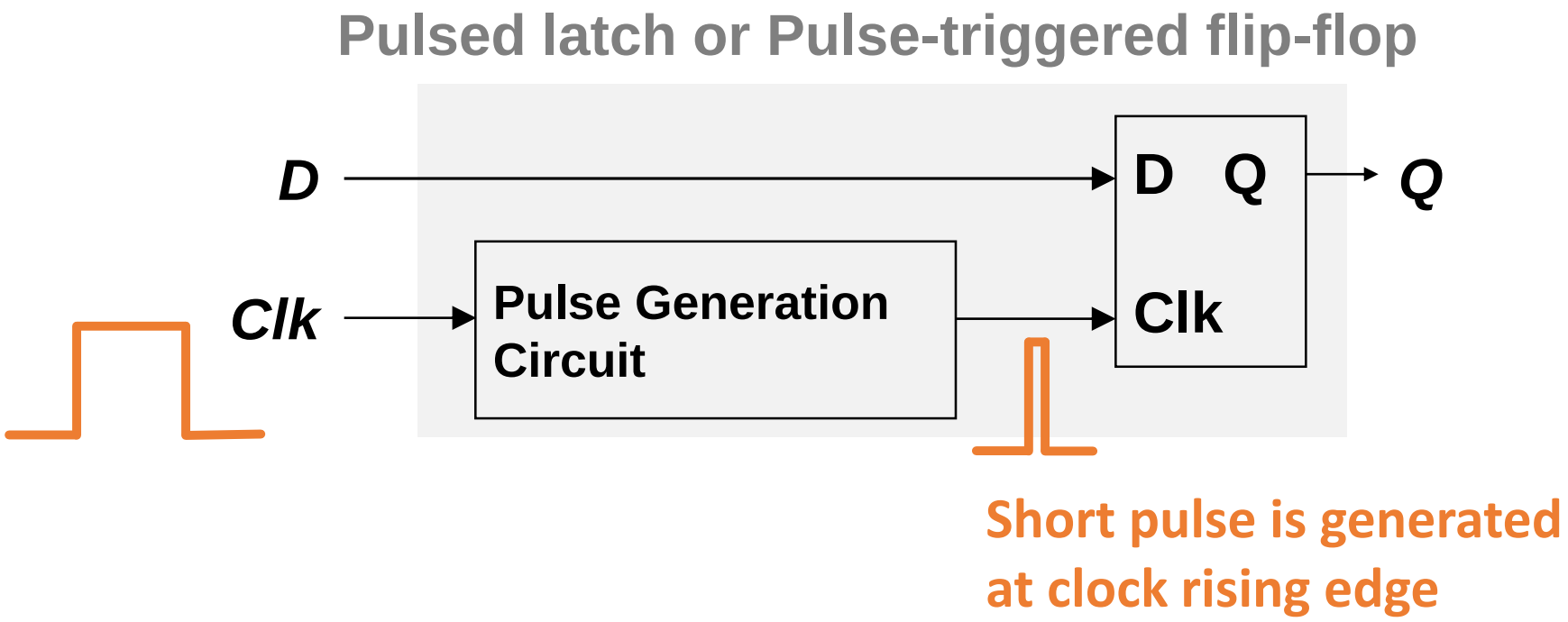
Ideation

- The latch reflects D into Q only when Clk = 1
- The aim of flip-flop is to reflect D into Q only at the rising edge of Clk.
- Then, how about shortly enable the latch at the rising edge of clock?
- That means,

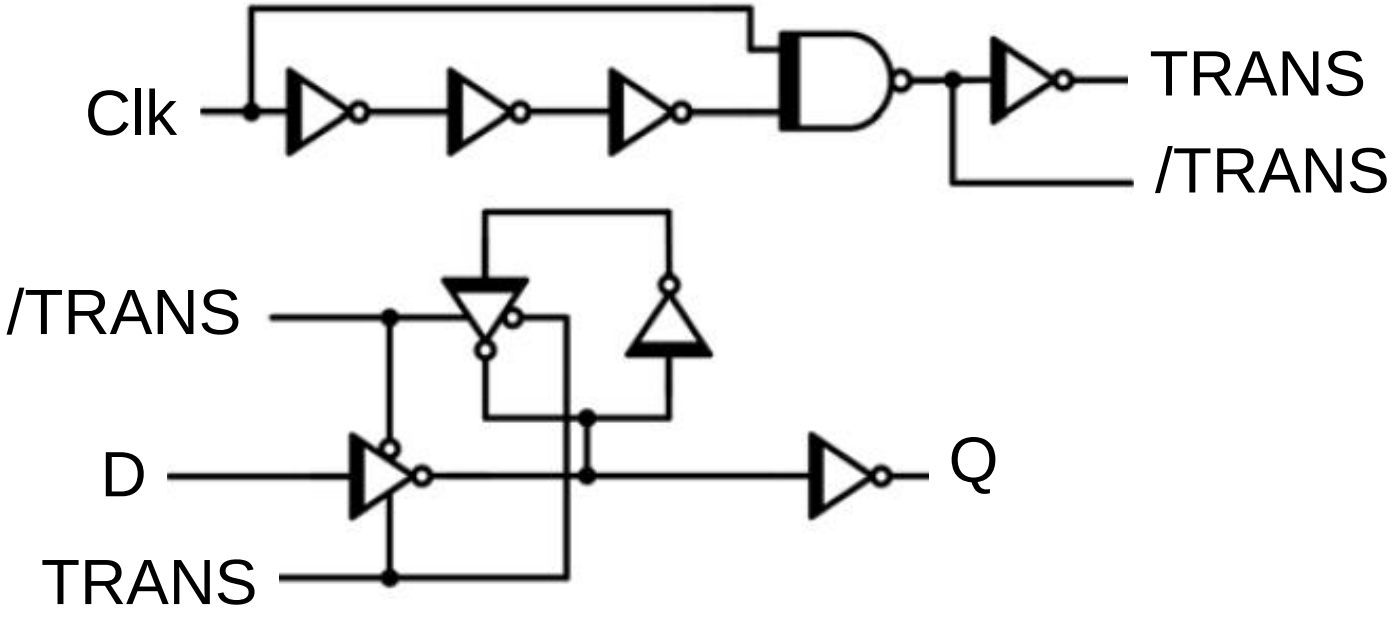




Pulsed Latch or Pulse-Triggered Flip Flop

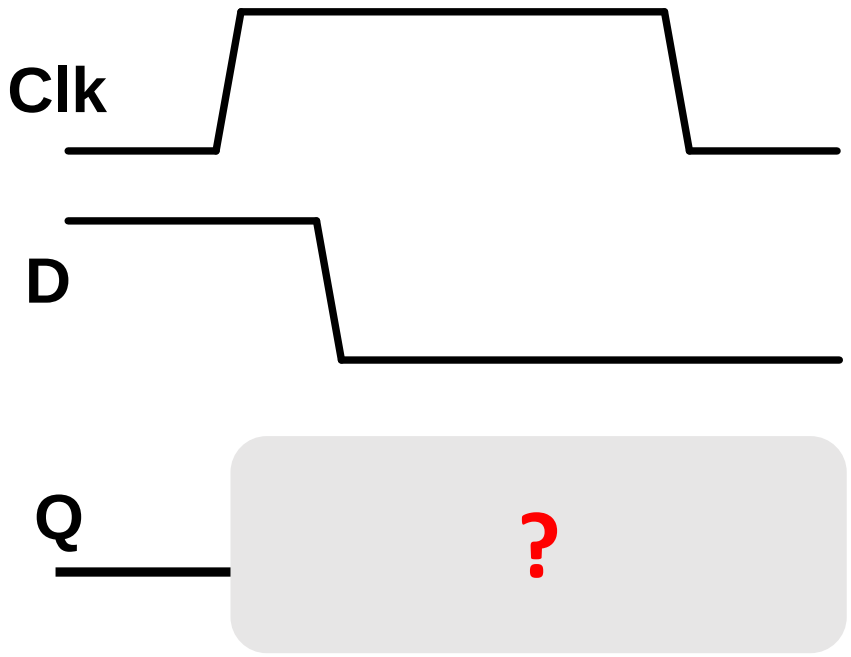


Conventional TG Pulsed Latch (TGPL)

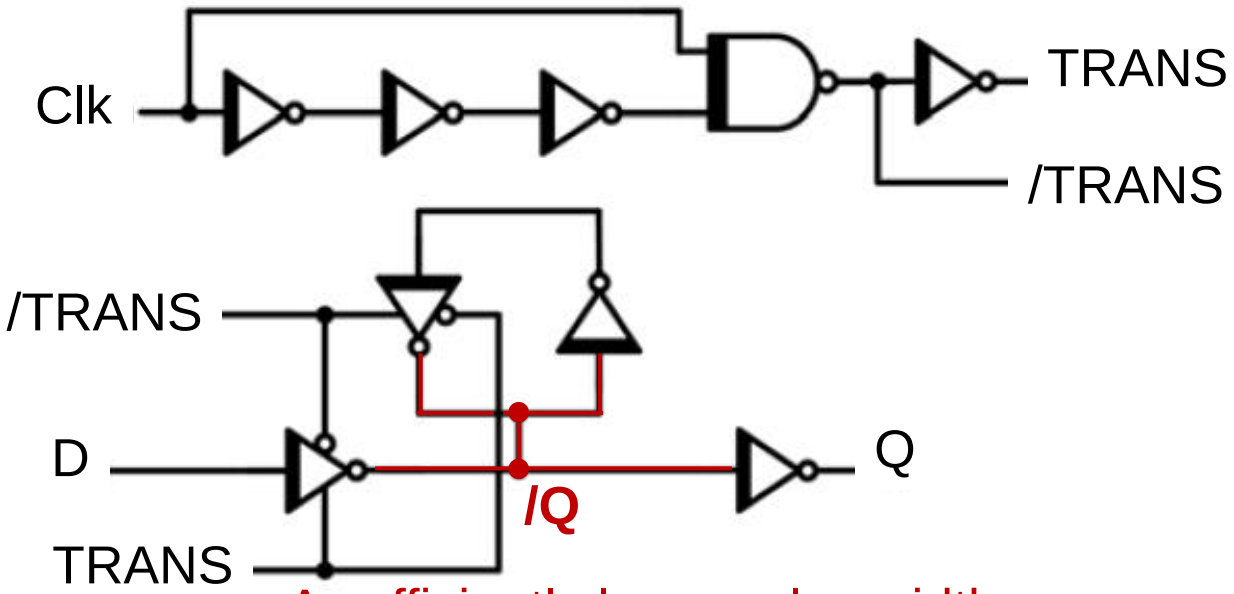


What if?

- Ideally, what should Q be?

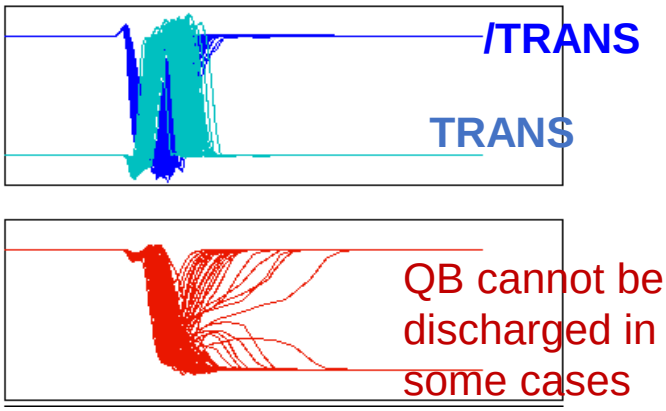


Large Pulse Width Constraint for Variation Aware Design

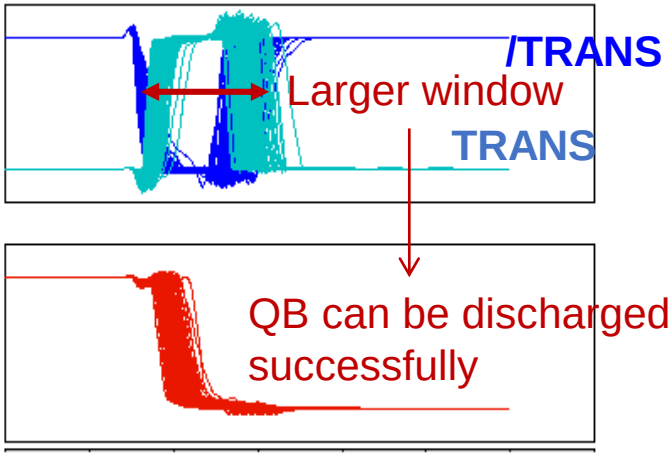


A sufficiently large pulse width is required for stable write of */Q*

Short pulse width, D=1



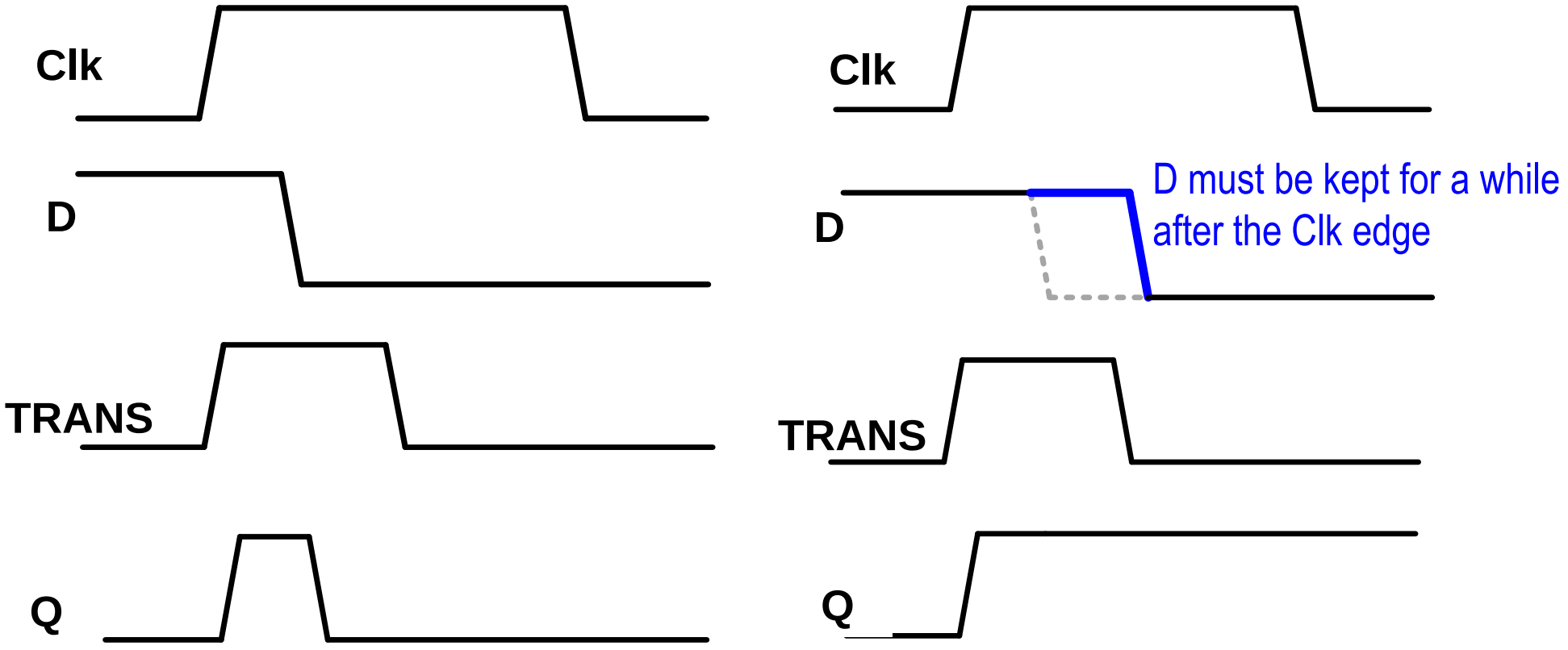
Long pulse width, D=1



S. D. Naffziger, et al., "The implementation of the Itanium 2 microprocessor," in *IEEE Journal of Solid-State Circuits*, vol. 37, no. 11, pp. 1448-1460, Nov. 2002.

Hold Time Issue in Pulsed Latch (Race Condition)

- How can we handle this? **We need hold time constraint for D!**



- **Hold time** is the minimum amount of **time** the data input must be unchanged after the clock edge for stable sampling

Ideation

Key: D = Q(t) while D ← Q(t+1) only at the rising edge

- Okay, let's accept Q(t+1) and store it **before clock becomes high** inside the flip-flop.

  ➔Q(t+1) must be disconnected to Q (∵Q should be kept with Q(t)).
  ➔Q(t+1) must be connected to D to accept and store it

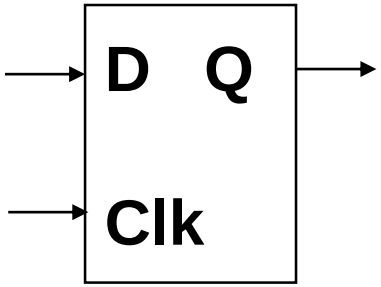
- **At the rising edge of clock**, stop accepting D and transfer Q(t+1) to the output Q.

  ➔Q(t+1) must be connected to Q.
  ➔Q(t+1) must be disconnected to D.

- Then, until the next rising edge of clock,

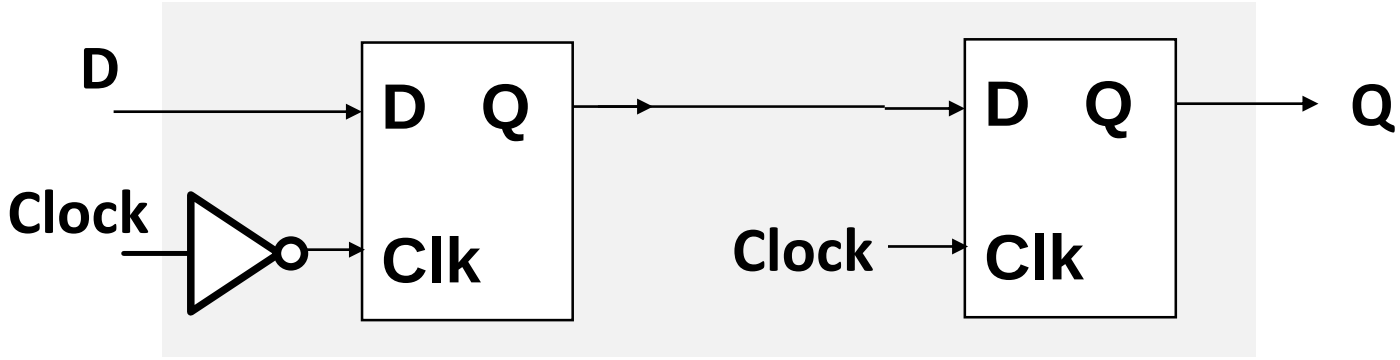
  ➔Q(t+1) must be disconnected to Q and Q(t+1) must be connected to D to accept and store it.
  ➔**Q(t) must be kept and stored**

Circuit Design for Flip Flop



Clk =0 : Q is kept
Clk=1 : Q ← D

- When Clk = 1, that is, just before clock rises
  - Q(t+1) is dynamically changed according to D (D acceptation) while **it is disconnected Q**.
  - Q(t) is stably & unchanged stored and **connected to Q.**
- Thus, we need two latches

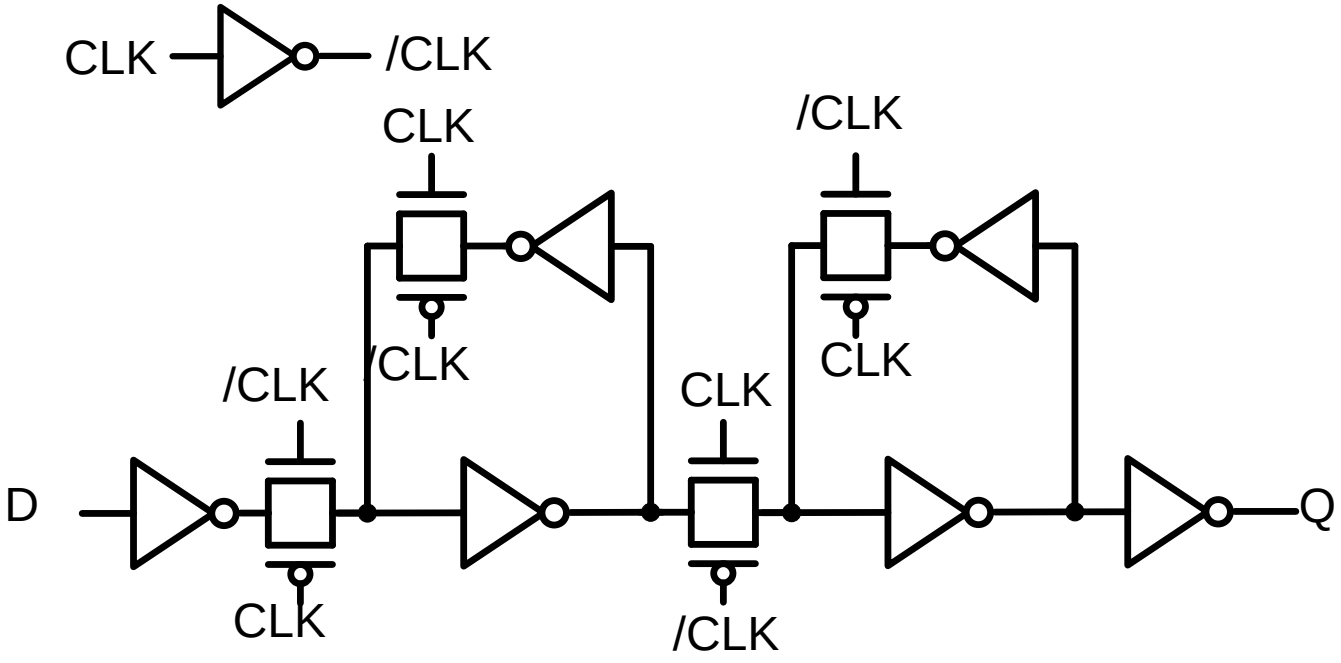


When is this latch supposed to accept D?

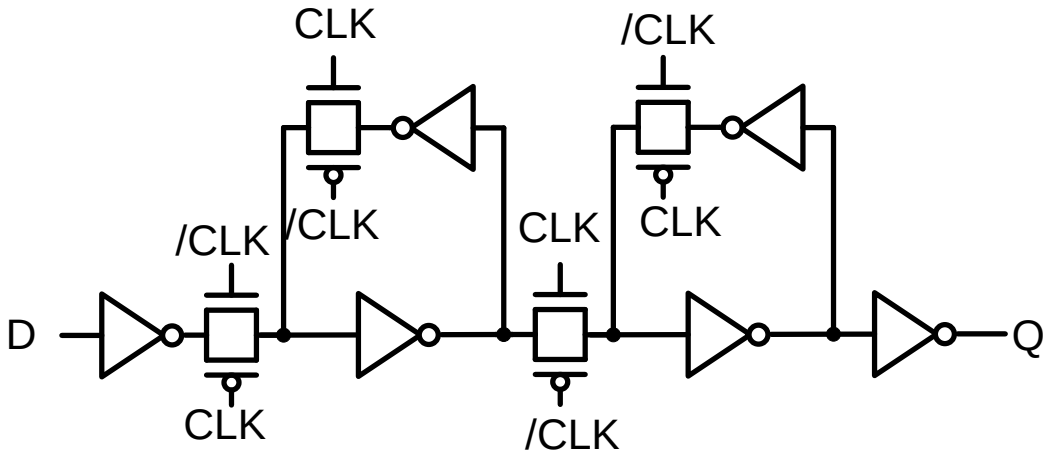
When is this latch supposed to update Q? (To what?)

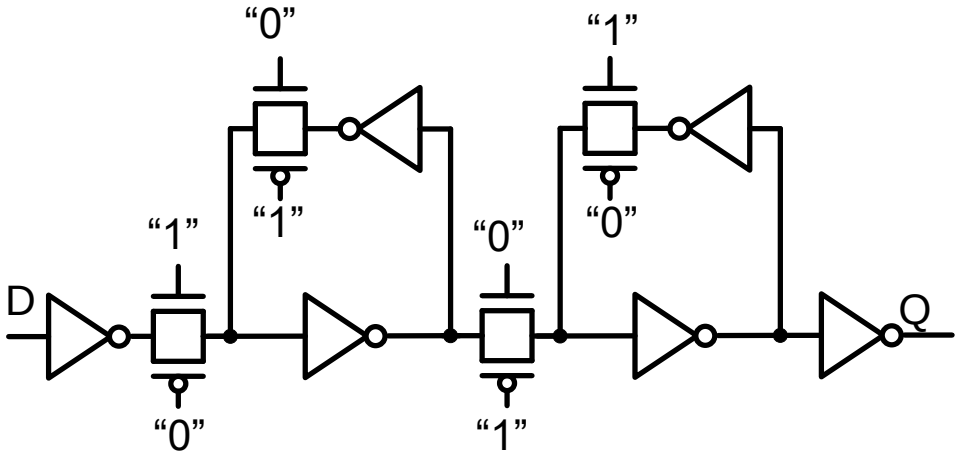
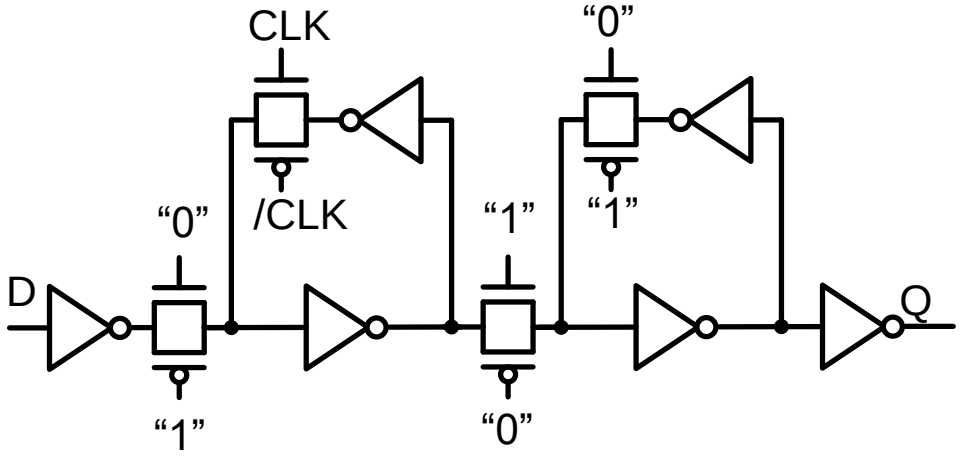
TR Level Schematic of Conventional TG Flip Flop (TGFF)

- We also call this type of structures as **master-slave** latch based Flip-Flop



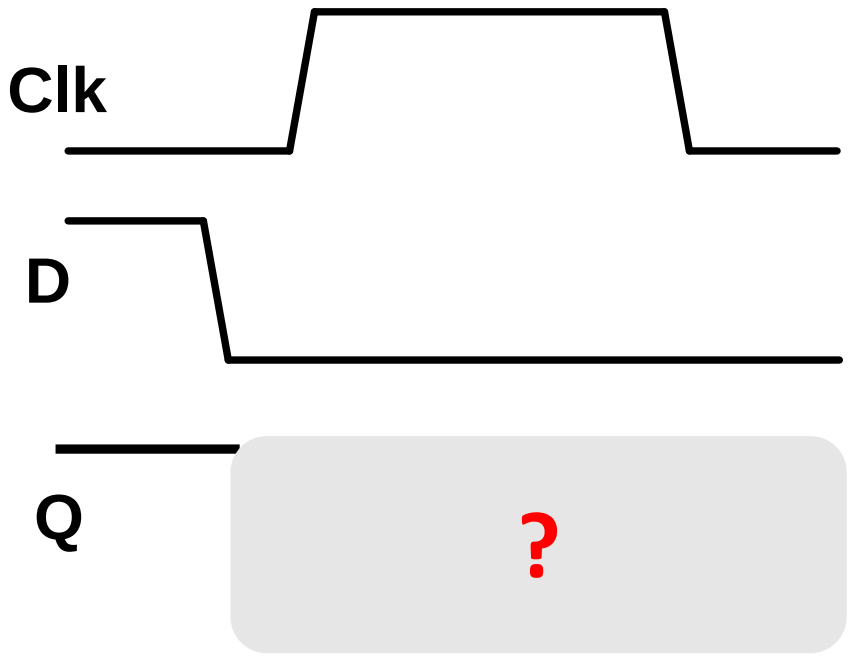
Operation of TGFF



Clk = low	Clk = high
	

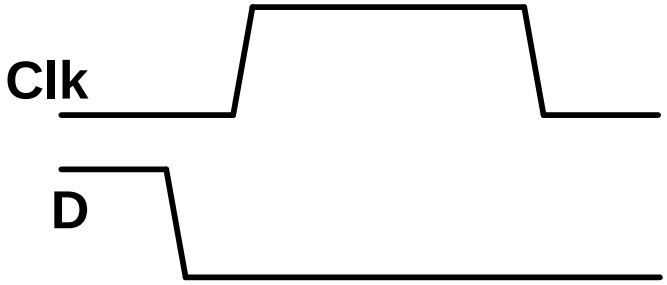
What if?

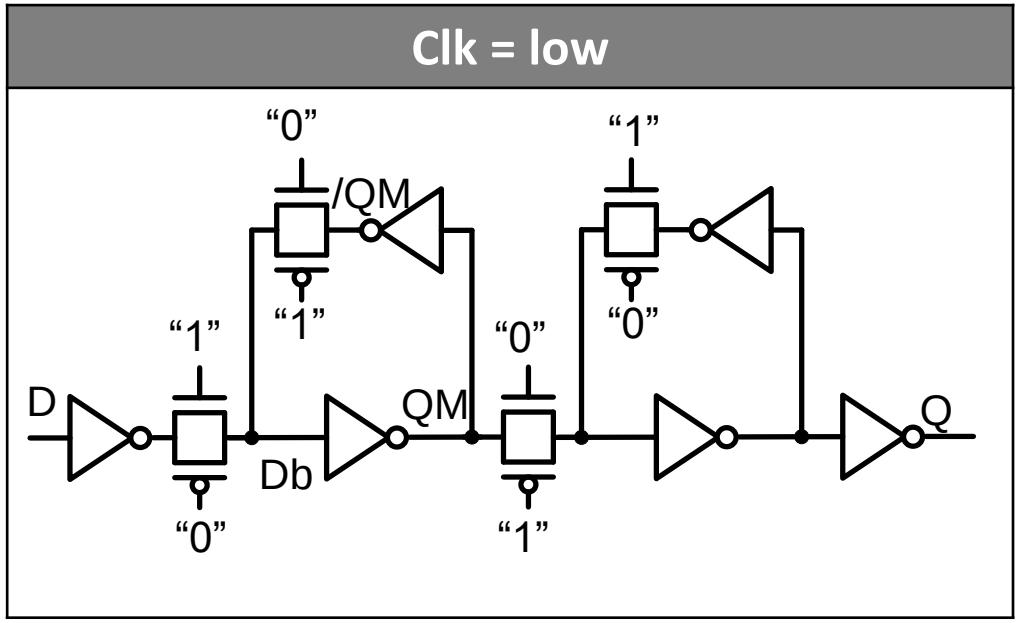
- Ideally, what should Q be?



Failing to Capture D @ Clk Rising

- During Clk=0, the master latch should capture low D.
- If Clk rises too quickly after D falls, D may not be able to reflected to the master latch.

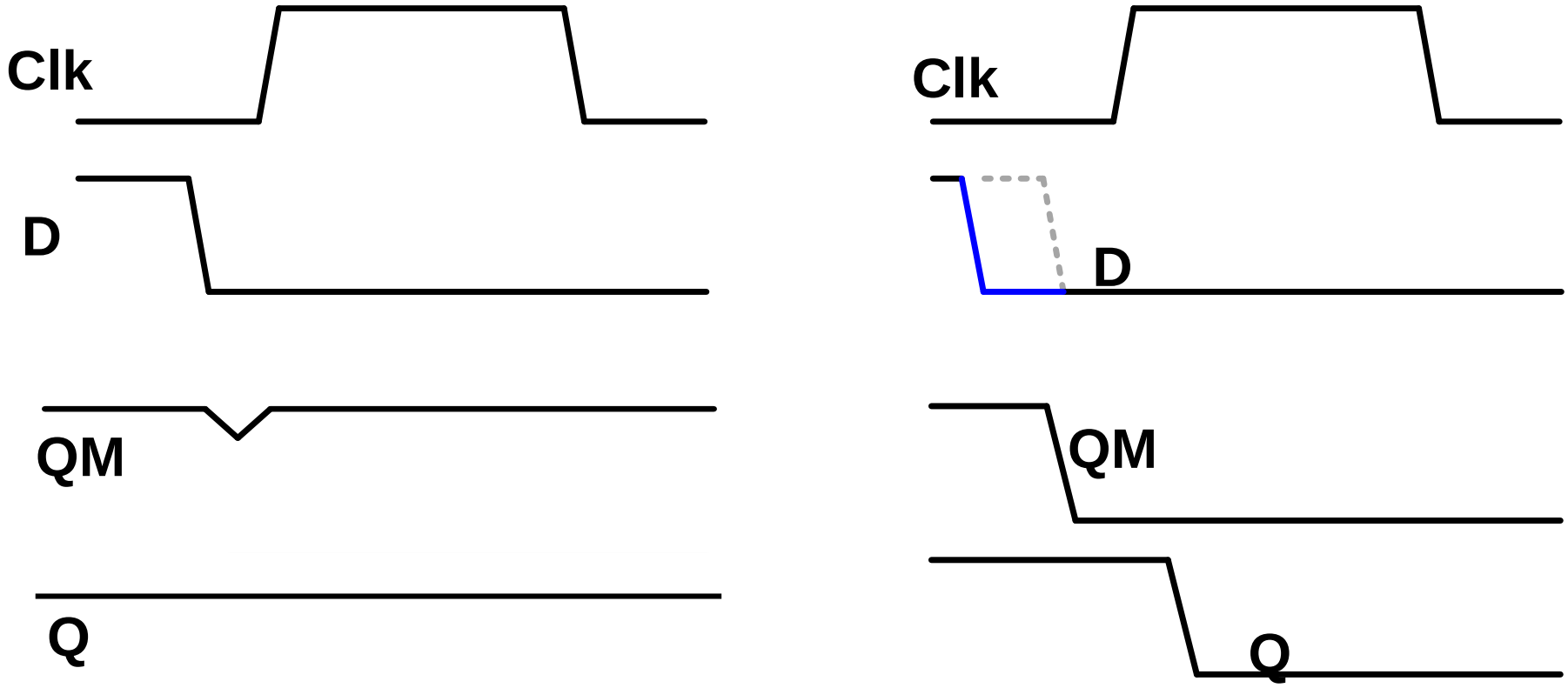




➔ **How can we handle this?**

Setup Time Issue in TGFF

- **We need a setup time constraint for flip-flop.**



- **Setup time** is the minimum amount of **time** the data input must be stable before the clock edge for stable latching.

Summary of Issues in Conventional FF (TGPL and TGFF)

- In TGPL, the generated pulse should have sufficiently large width to guarantee the appropriate development of internal nodes of the latch.
- Enlarged pulse width may reflect D change after Clk edge inside the latch, which incurs the malfunction.
- Thus, D should be kept unchanged after the Clk rising edge, meaning the TGPL have a finite positive hold time constraint.
- In TGFF, the master latch a finite time to capture D before Clk edge because it takes times for the internal node to change.
- Thus, if D changes right before Clk edge may not be reflected to the master latch, meaning TGFF requires a finite positive setup time constraint.

Any Way We Can Mitigate Conventional FF Issues?

- There may be two approaches
  1) Reducing hold time for TGPL.
  2) Reducing setup time TGFF.
- Let’s see how 1) can be implemented thru a circuit design.

Ideation

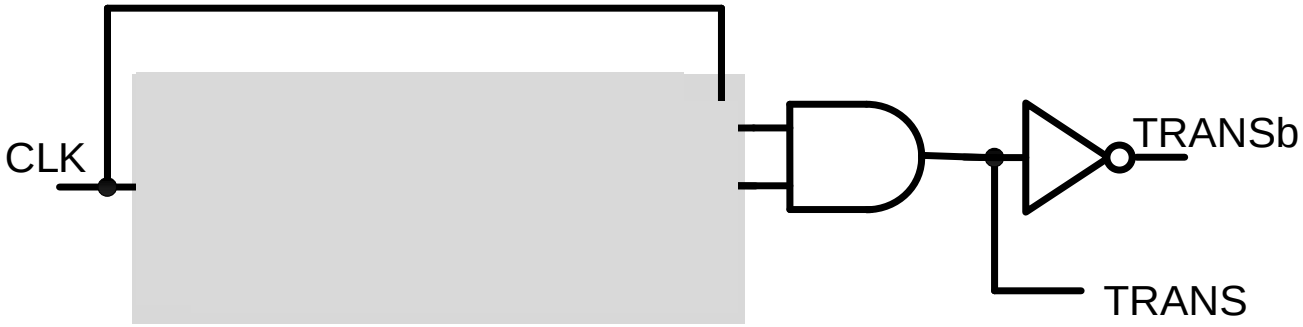
- The cause of large hold time in TGPL is pulse width may unnecessarily large in many TGPLs, although it is inevitable to cover the slow TGPL.
- We can reduce the hold time if we can generate the pulse width as the exact time it is required for a TGPL.
- What is the meaning of the exact time?

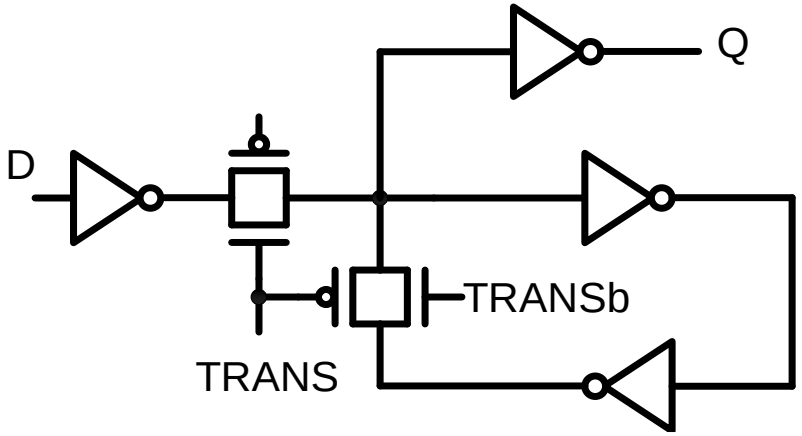
➔ That is the time required for Q to be captured properly.

➔ As soon as Q is well developed, we disconnect D from the latch.

- Is there any way we can disconnect D from the internal nodes of the latch, **right after** the capture is performed properly?

Fitted Delay Generation is Key!



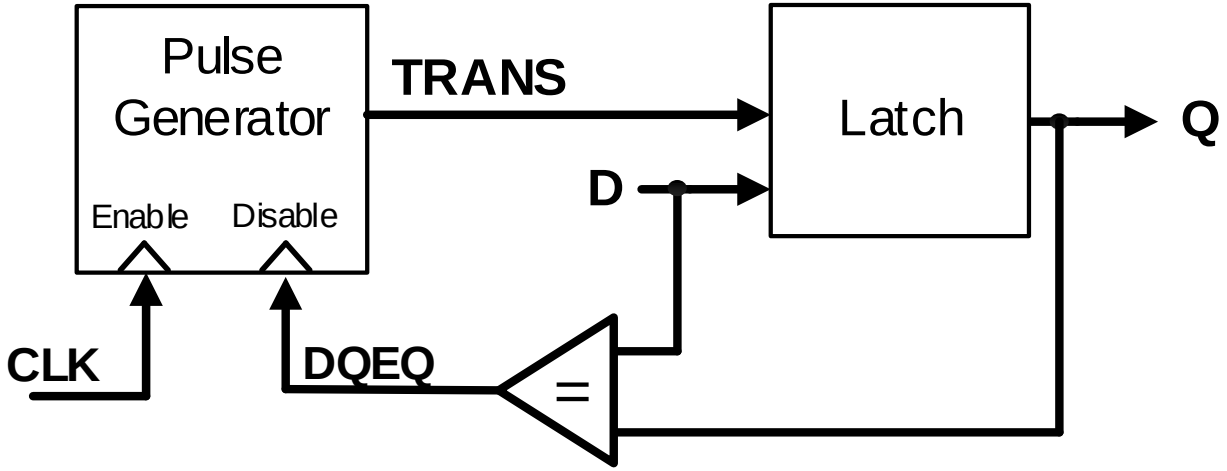


S. Luo, et al, "An Adaptive Pulse-Triggered Flip-Flop for a High-Speed and Voltage-Scalable Standard Cell Library," in *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 60, no. 10, pp. 677-681, Oct. 2013.

➔ **Local variation can still exist!**

Let's Sense Whether the Q is Captured

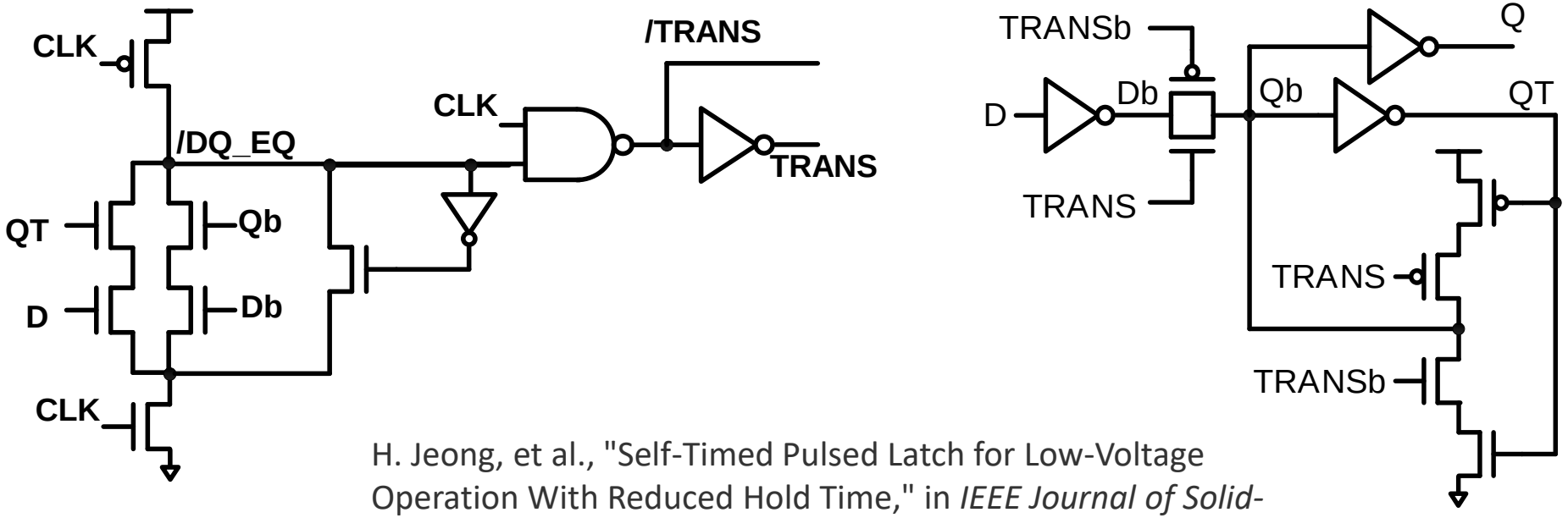
- How can we sense whether Q is properly developed?
- We can sense whether Q becomes same with D.



TRANS Generation Through XNOR

- We saw various XNOR and XOR. Static XNOR/XOR is bulky.
- What we have to do is compare D & Q after the rising edge.

➔ Thus, we can utilize dynamic XNOR

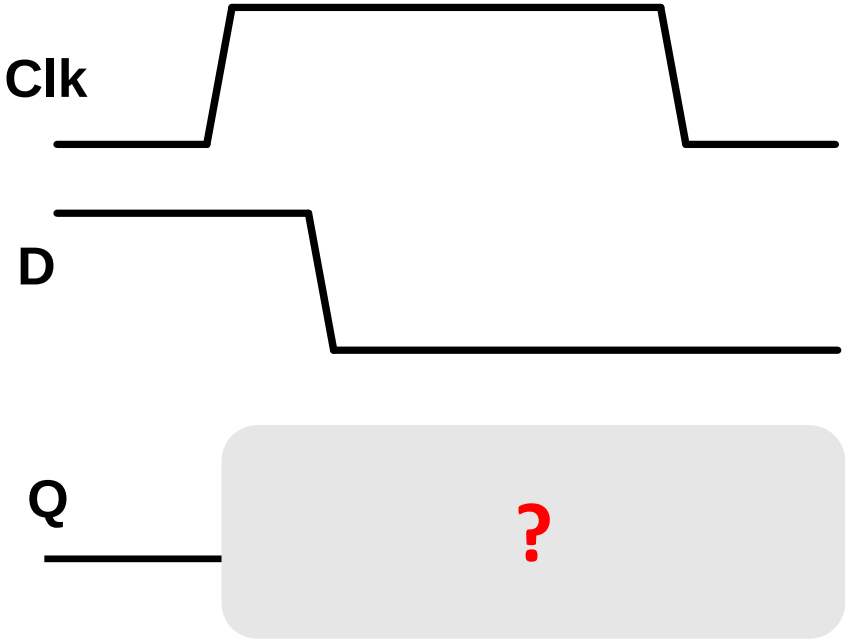


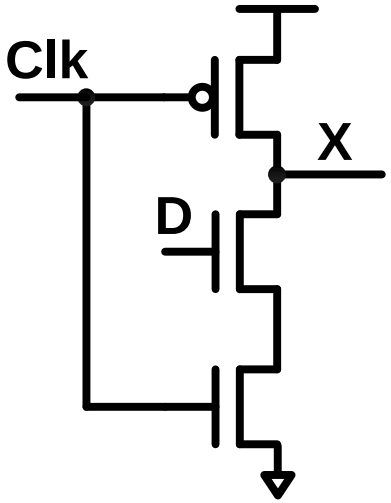
H. Jeong, et al., "Self-Timed Pulsed Latch for Low-Voltage Operation With Reduced Hold Time," in *IEEE Journal of Solid-State Circuits*, vol. 54, no. 8, pp. 2304-2315, Aug. 2019

Hold Time Issue Arises When..

- How about disabling effect of falling D at the rising edge of Clk?

➔ Dynamic circuit has the monotonicity!

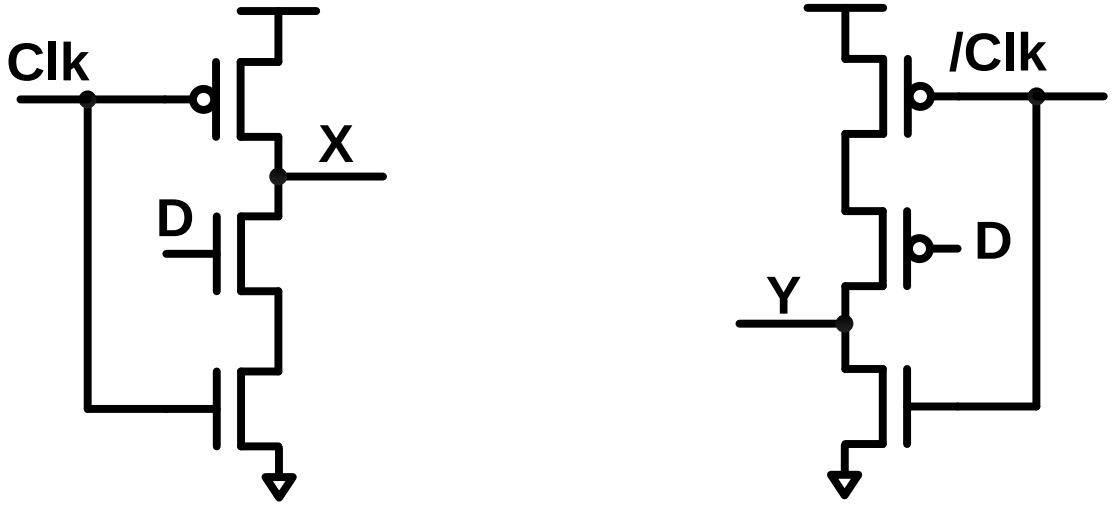




Ideation (cont'd)

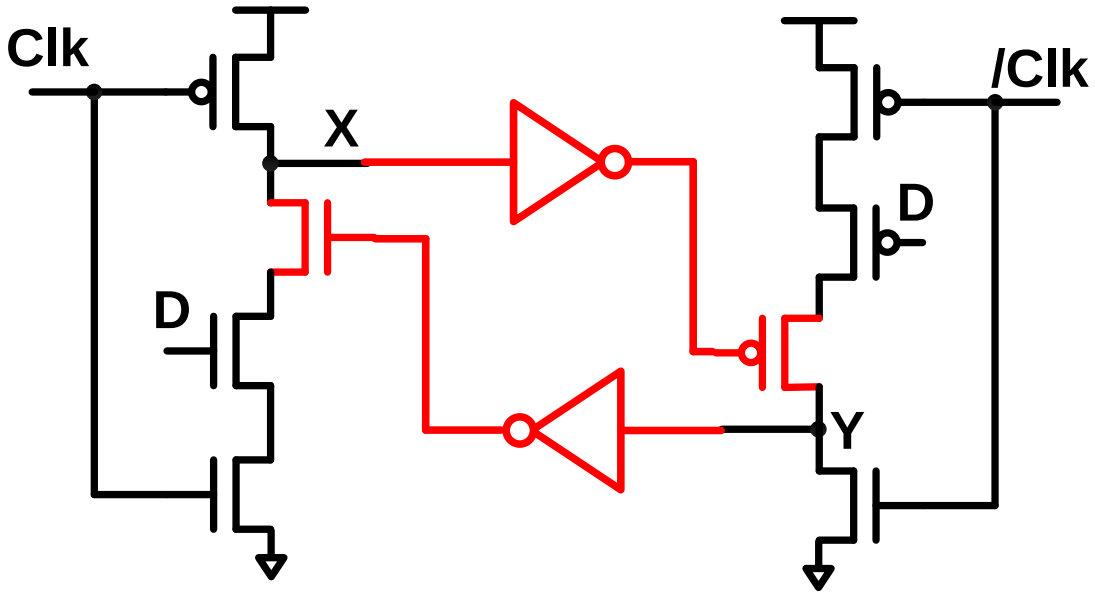
- Once X falls, D=1 should be captured.
- How about if D=0? We can use a dual dynamic circuit!
- Once Y rises, D=0 should be captured.
- As soon as D is sensed, the input D should be blocked/disconnected from the latch. **How?**

➔ Y rise blocks D to X path, X fall blocks D to Y path (Why?)



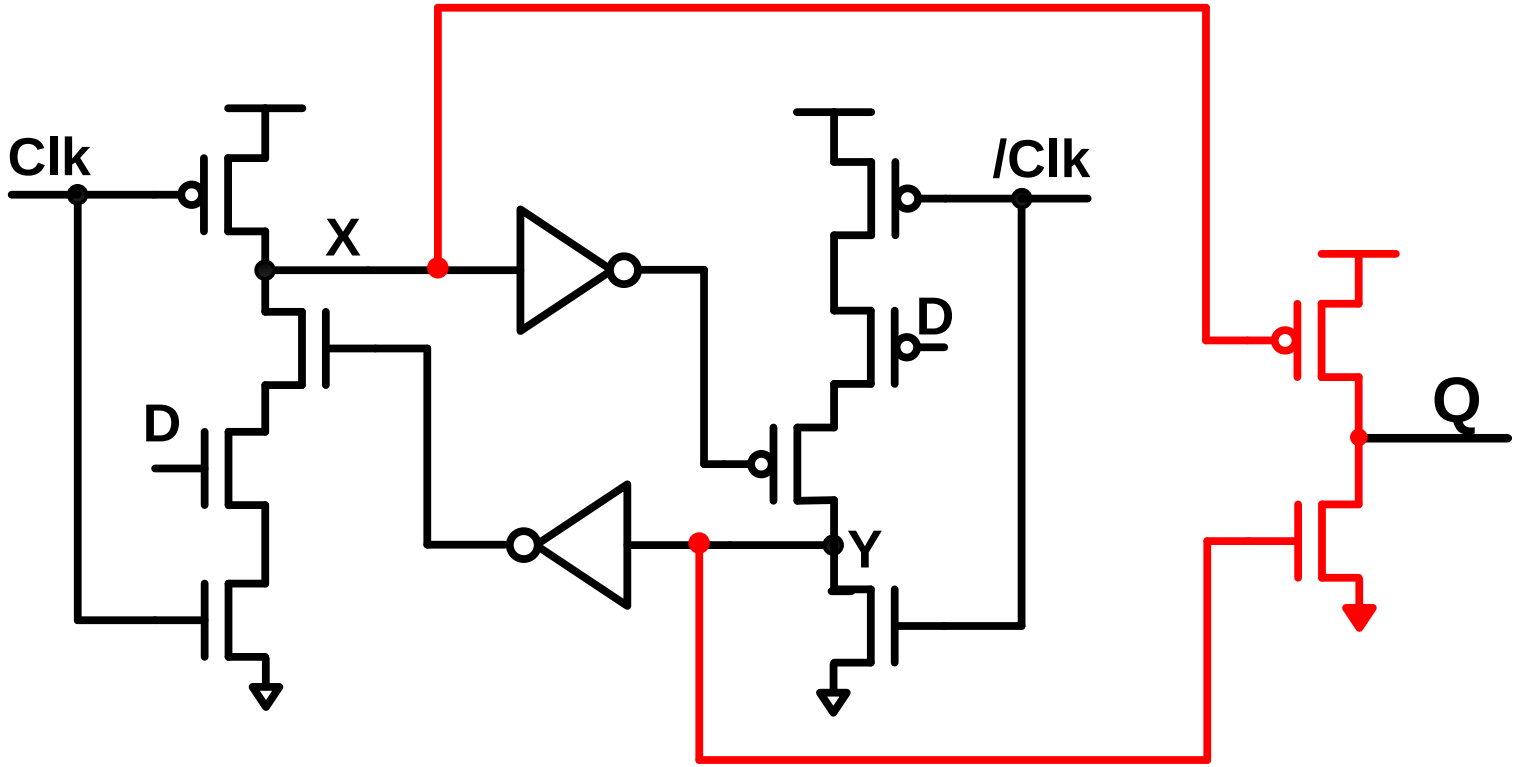
Ideation

- Y rise blocks D to X path, X fall blocks D to Y path



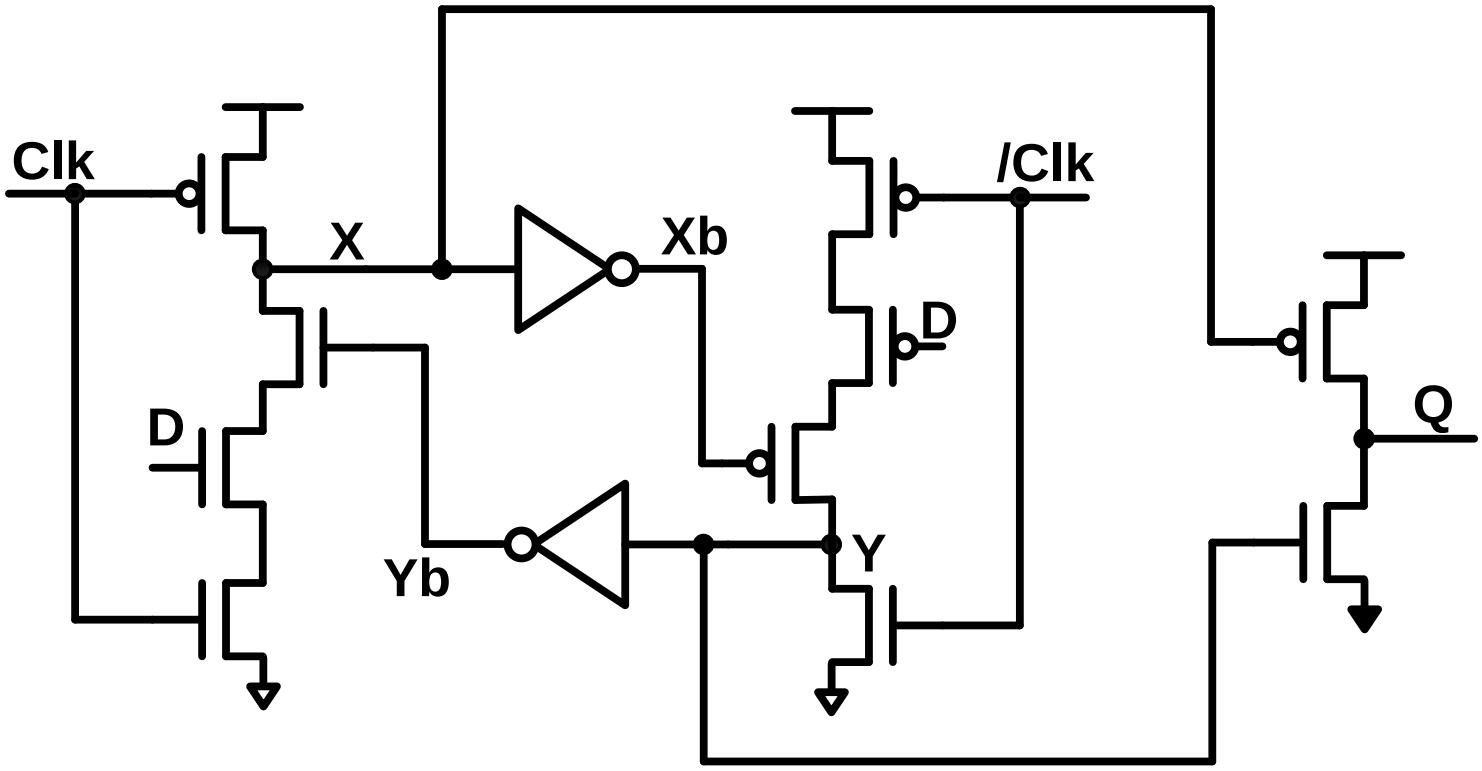
Circuit Completion; Q Driving

- X = 0 means D = 1 at the Clk edge, thus Q should be 1.
- Y = 1 means D = 0 at the Clk edge, thus Q should be 0.

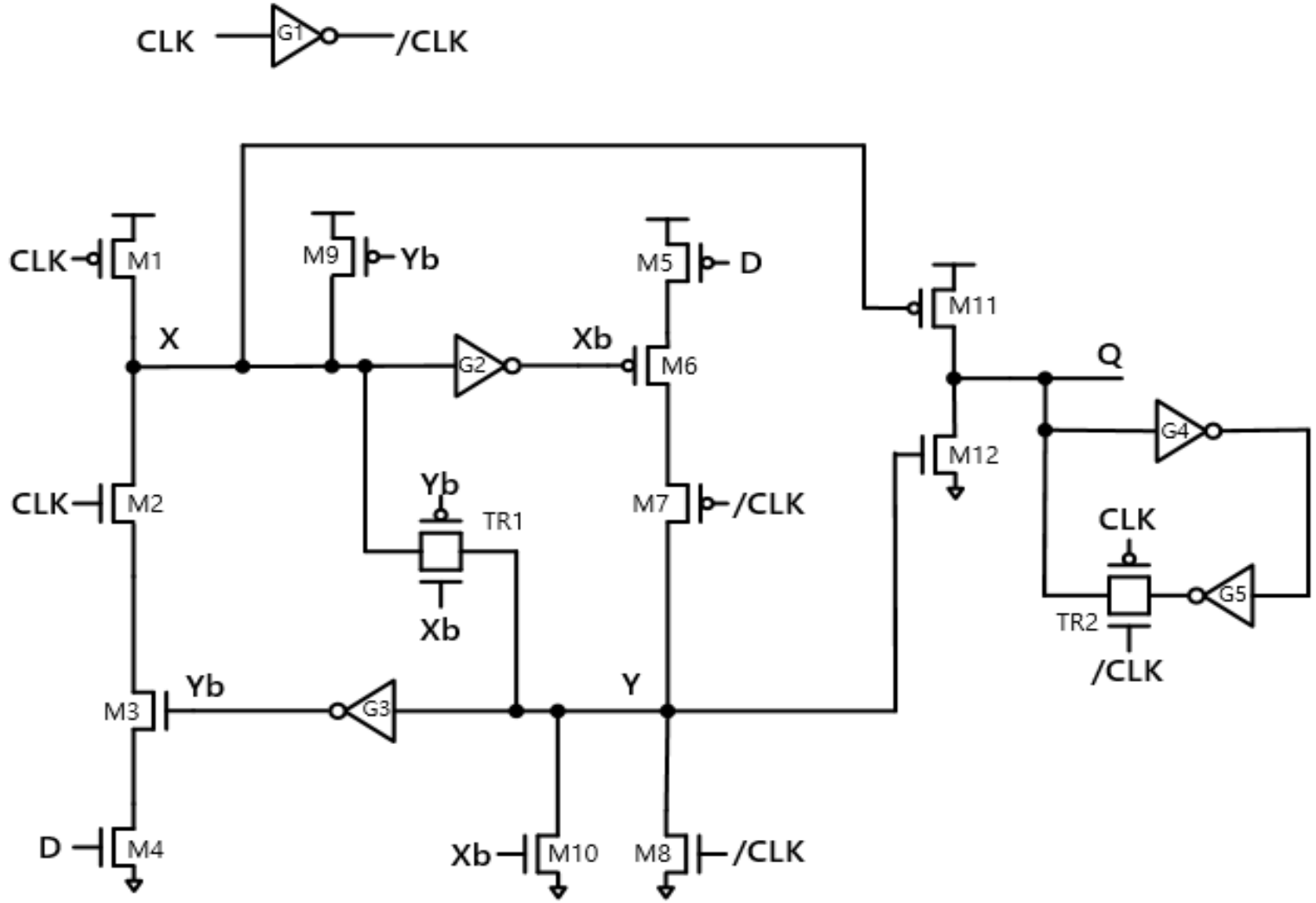


Circuit Completion; Latching

- Q should be latched for low Clk.
- X or Y should be latched for high Clk.



Complete Design for Self Shut-off Pulsed Latch



Summary

- TGPL has the limitation of large hold time constraint due to inevitable large pulse width.
- To reduce the hold time in TGPL, the adaptive pulse triggering techniques have been proposed.