Lecture 8: Timing Elements (a.k.a Sequential Elements)

Based on material prepared by prof. Visvesh S. Sathe

Acknowledgements

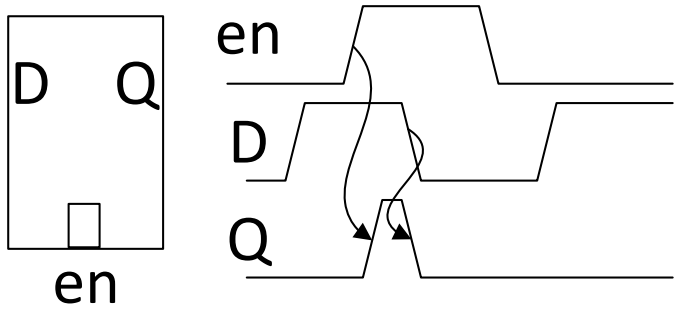
All class materials (lectures, assignments, etc.) based on material prepared by Prof. Visvesh S. Sathe, and reproduced with his permission

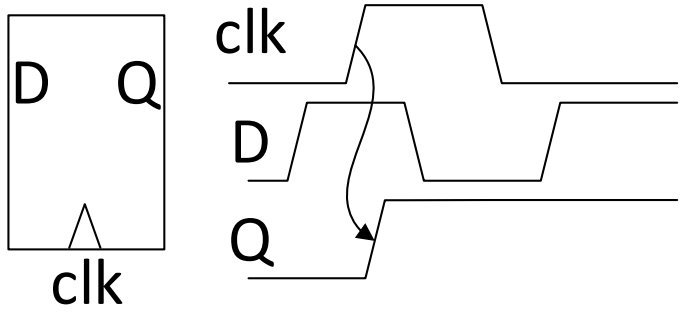
Visvesh S. Sathe
Associate Professor
Georgia Institute of Technology
https://psylab.ece.gatech.edu

UW (2013-2022)
GaTech (2022-present)

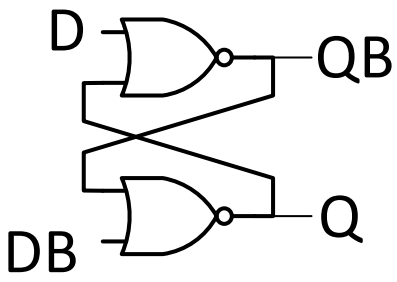
Coverage

- Latch
  - Basic structure and operation
  - Common nomenclature (A-Phase, B-Phase)
- Flip-Flops
  - Basic topology (Master-Slave)
  - Set/Reset, Scan capability
- Timing
  - Timing properties of sequential elements (Setup, Hold, clk-Q, D-Q)
  - Static timing example





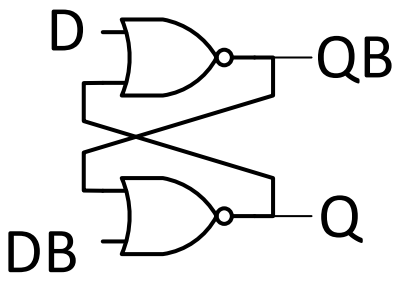
The Traditional Latch…



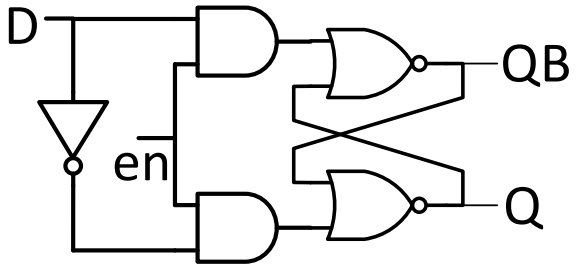
D	DB	Q[n]	QB[n]
0	0	Q[n-1]	QB[n-1]
0	1	0	1
1	0	1	0
1	1	0	0

- Basic objective: Robustly hold the logic state of a node
- Data processing synchronized by system clock → use clock to perform data-retention

The Traditional Latch…



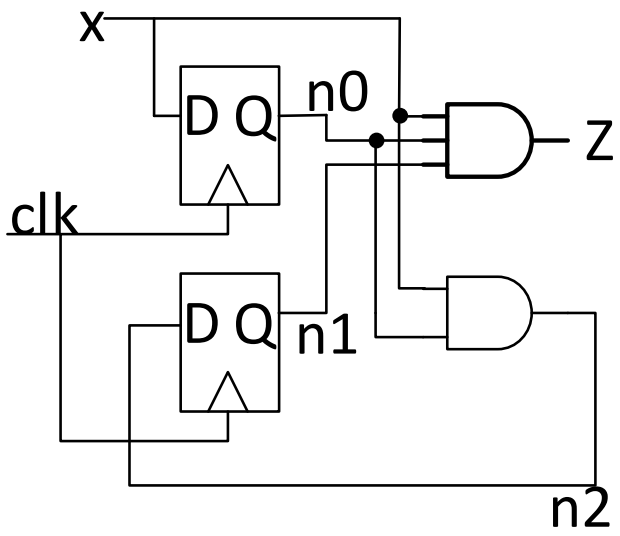
D	DB	Q[n]	QB[n]
0	0	Q[n-1]	QB[n-1]
0	1	0	1
1	0	1	0
1	1	0	0

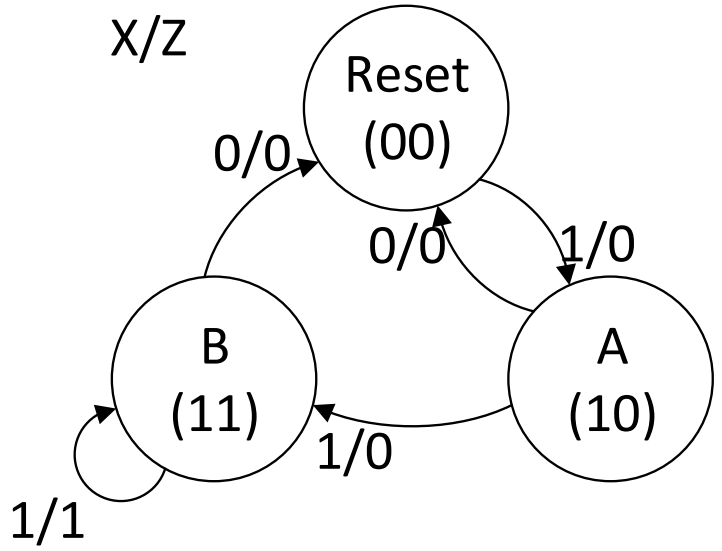


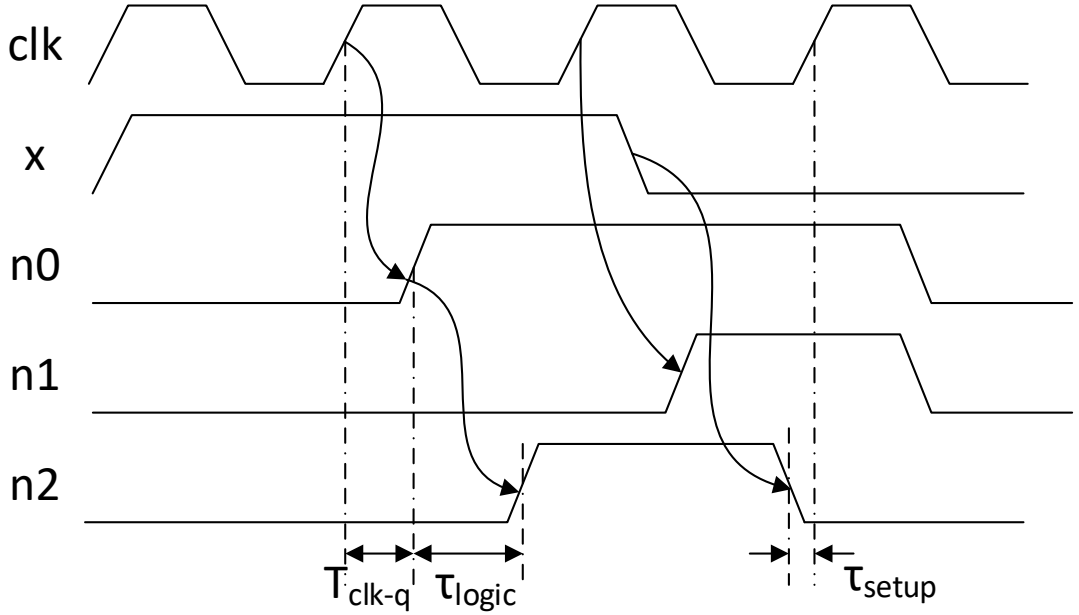
D	en	Q[n]	QB[n]
0	0	Q[n-1]	QB[n-1]
1	0	Q[n-1]	QB[n-1]
0	1	0	1
1	1	1	0

- Basic objective: Robustly hold the logic state of a node
- Data processing synchronized by system clock → use clock to perform data-retention

Motivation-1: State Machines

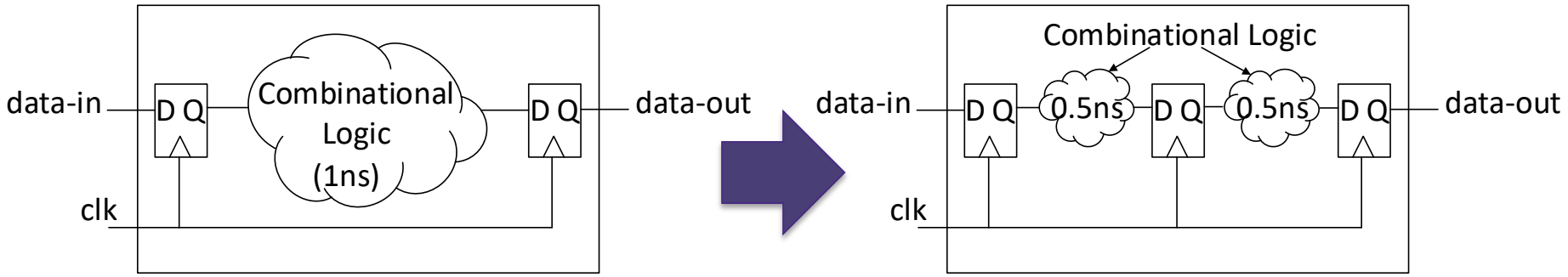






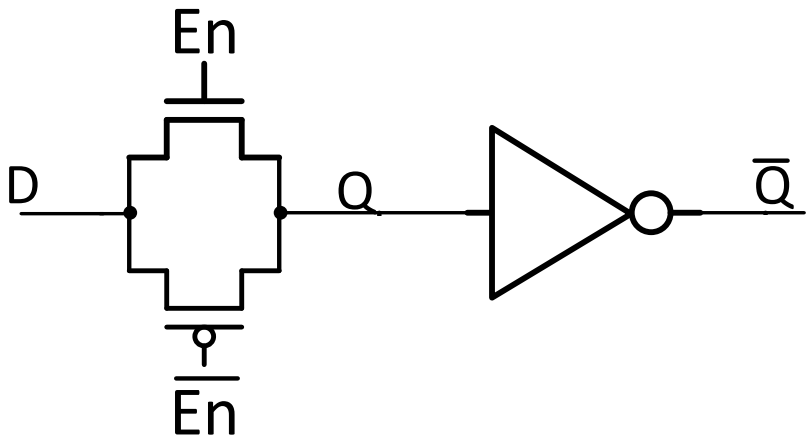
- *clk* samples *n2* and *x*
- Timing rules govern correct operation
  - How late can *n2* switch

Motivation-2: Pipelines



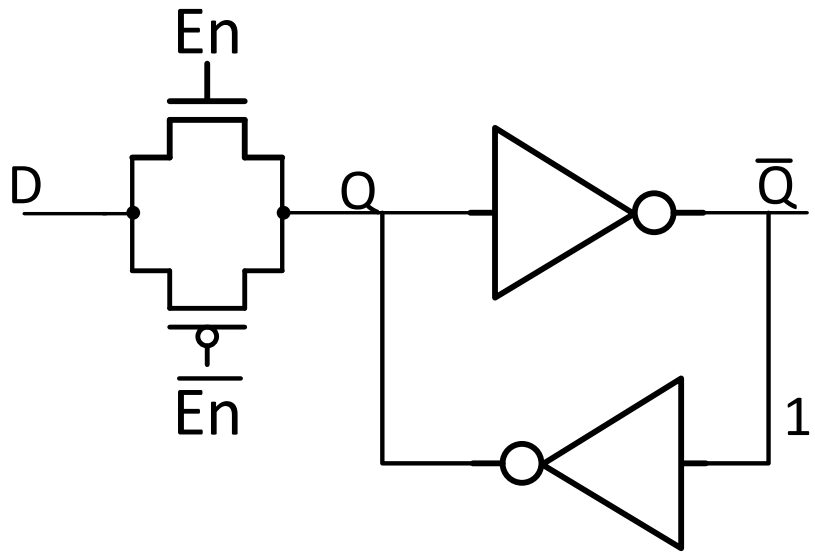
- Un-pipelined design: New data is computed after previous data computation is completed
- Pipelined design: ***Capture*** "in-flight" data being computed and ***Launch*** new data concurrently
  - Use a state-retention element (latch/flop) to hold the 2nd stage input steady while the first stage toggles due to the computation result of fresh inputs
  - Widely used performance/efficiency enhancement
- Timing properties of FF are very important in determining robustness/performance
  - What is the delay of the FF (clk→Q)
  - How soon must data arrive at the FF before its sampled
  - Does the latched data need to be held steady for some time? How much ?

Latch Construction - Recap



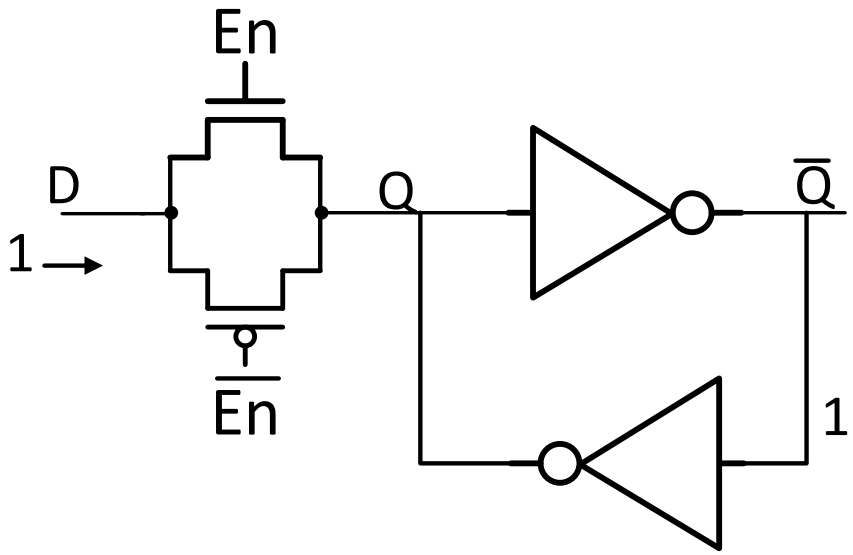
- Conditionally connect/disconnect transmission gate

Latch Construction - Recap



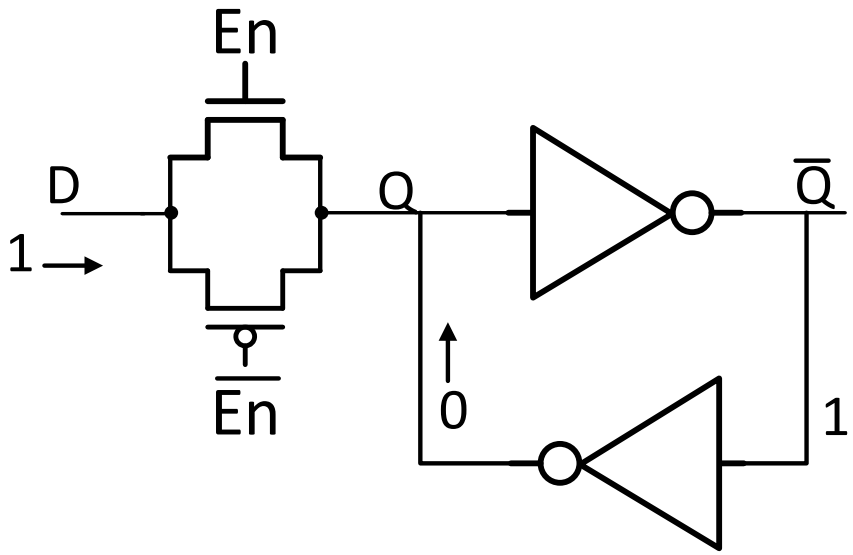
- Conditionally connect/disconnect transmission gate

Latch Construction - Recap



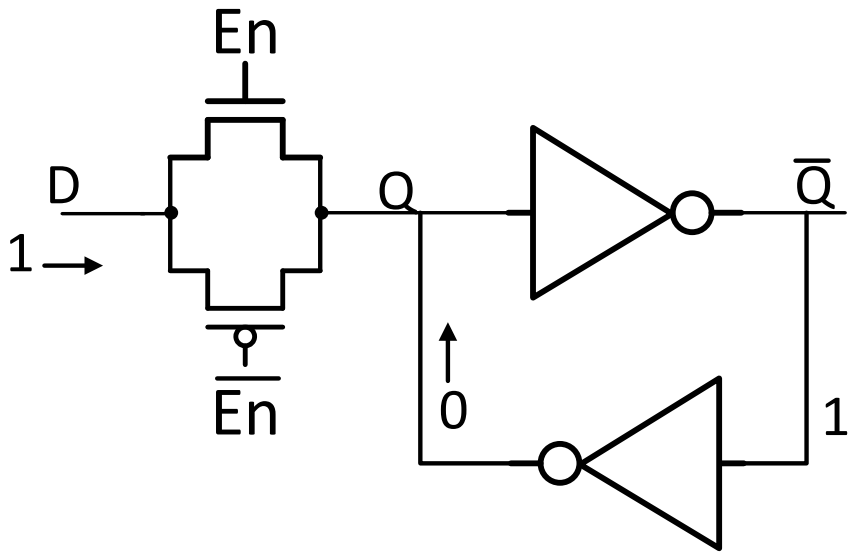
- Conditionally connect/disconnect transmission gate

Latch Construction - Recap



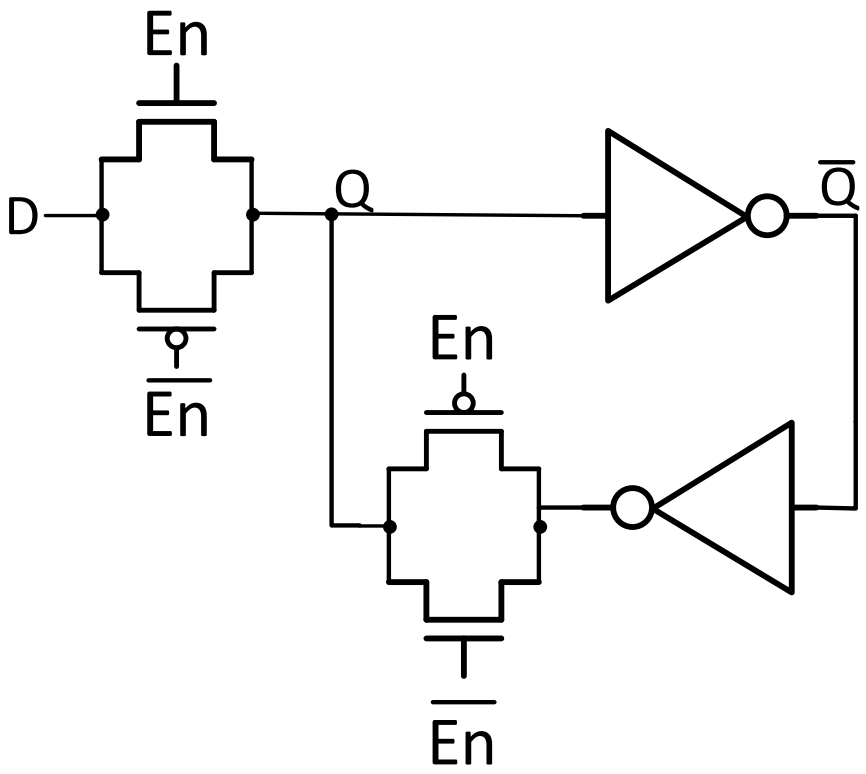
- Conditionally connect/disconnect transmission gate

Latch Construction - Recap



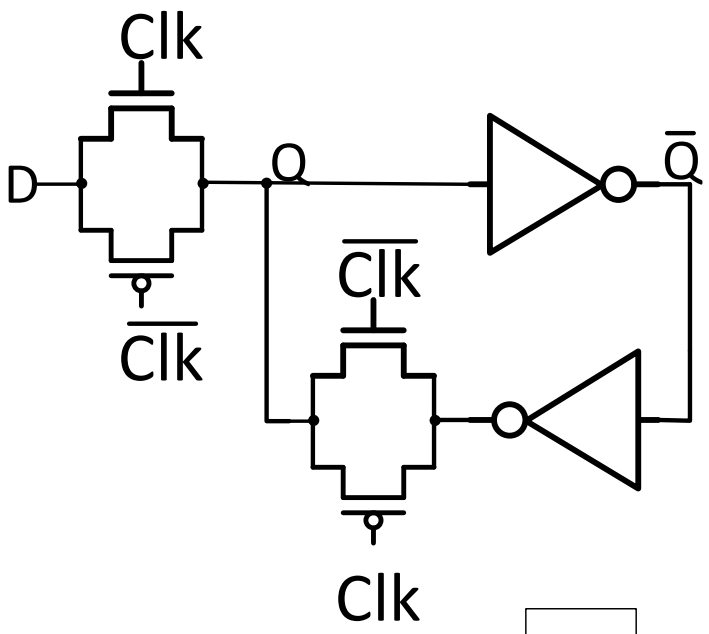
- Conditionally connect/disconnect transmission gate
- Robust operation requires Q to be actively held when En=0

Latch Construction - Recap

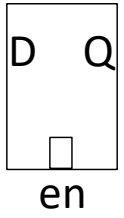


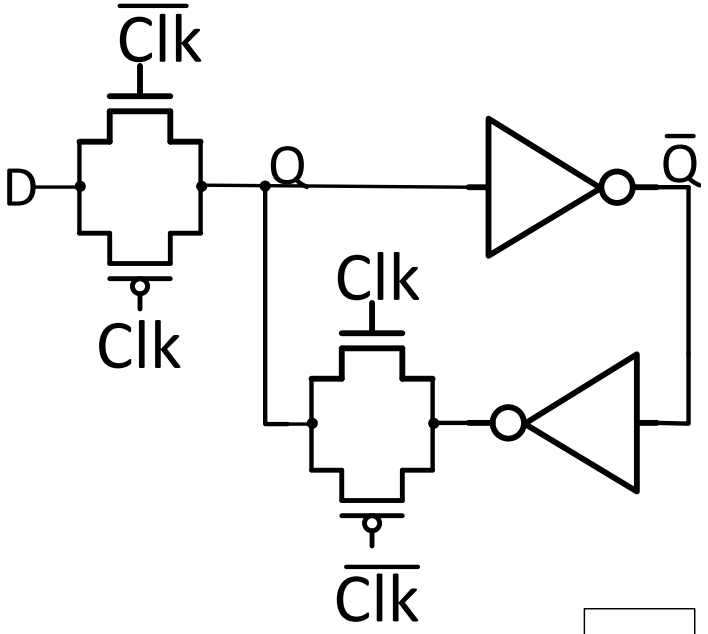
- Avoid contention with use of complementary transmission gate operation
  - When En='1': Latch is ***transparent***. Feedback path is open (turned off)
  - When En='0': Latch is ***opaque***. Feedback path holds state

Latch Operation (Timing)

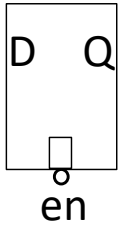


A-Phase Latch





B-Phase Latch

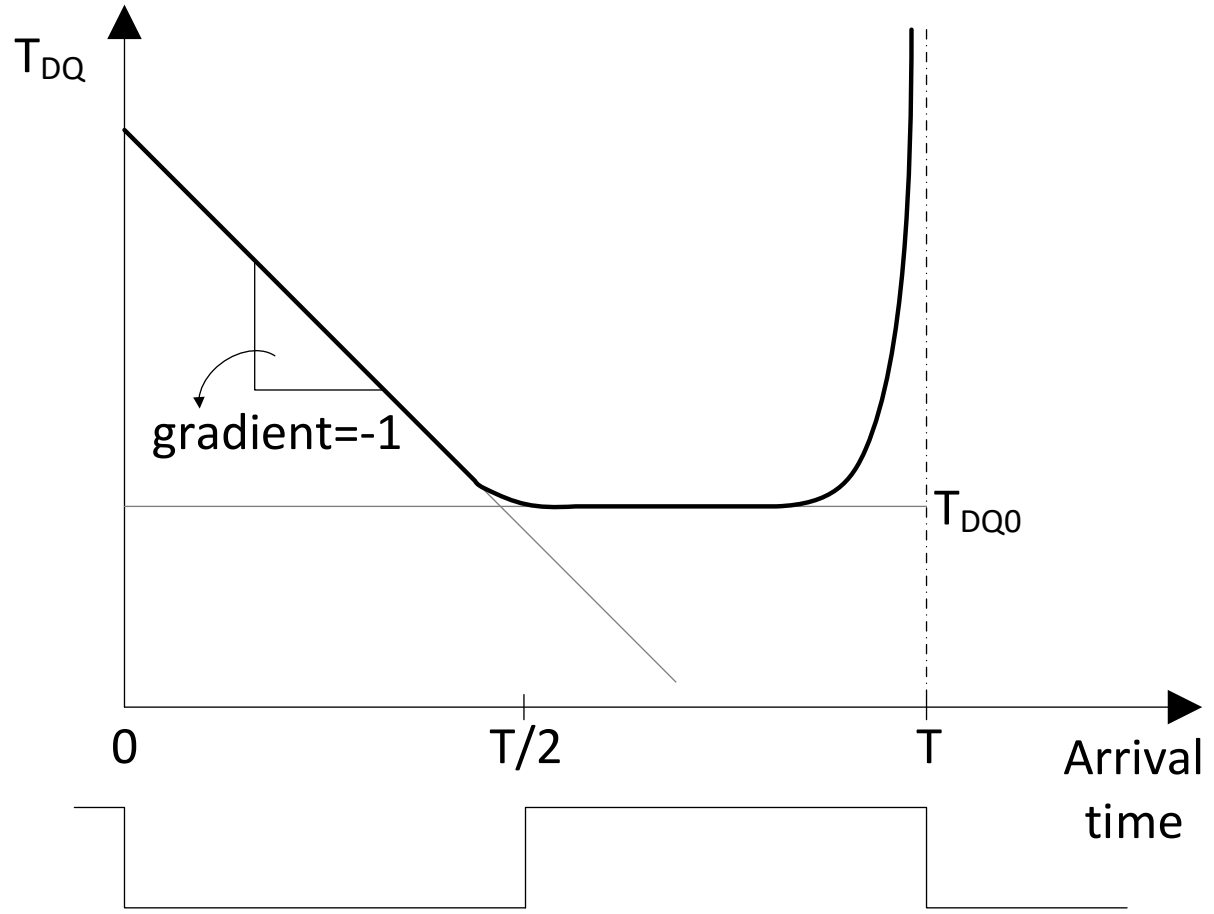


- Referred to as “Level-Senstive” timing elements
  - Transparent Phase : Allow data arriving at D to pass to Q
  - Opaque Phase : Hold data at Q steady regardless of values at D
- Latch transparent in the A-phase is called an A-phase latch
- Latch transparent in the B-phase is called a B-phase latch

Timing Characteristics : Nomenclature

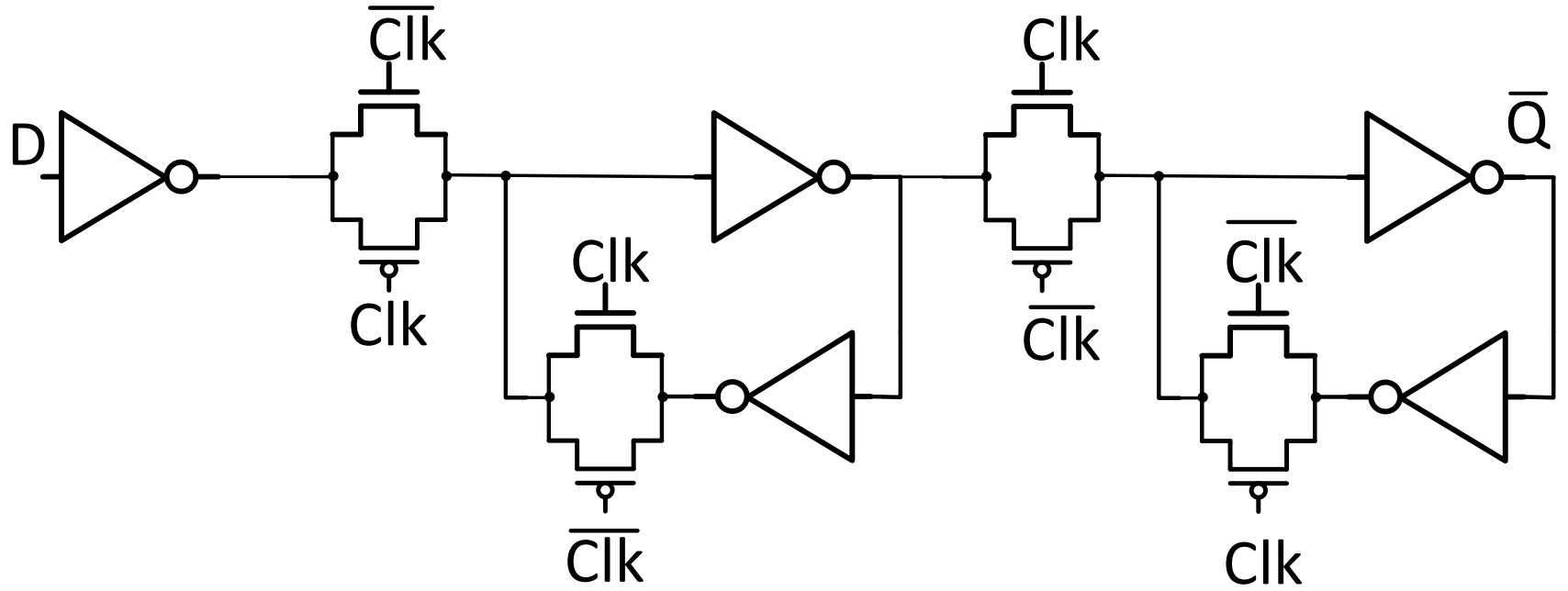
- Data is said to ***arrive*** at the latch input D at time: ***arrival_time***
- Data is said to ***depart*** latch output Q at time: ***departure_time***
- T_{CQ} is the delay between the clock transparency edge and the departure time
- T_{DQ} is the delay between departure_time and arrival_time
- T_{setup} is the delay **before** the latch becomes **opaque** that data must arrive for a "good-quality" capture
- T_{hold} is the delay **after** the latch becomes opaque that data must be held steady for a "good-quality" capture
- For both setup and hold times, "good-quality" capture is required. Correctly latching the value is not good enough

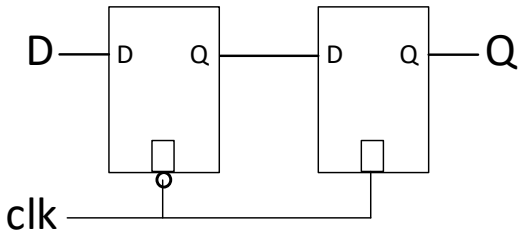
Latch Timing Characteristics : A quick overview



- Latch T_{CQ} is relevant when data arrives as latch is opaque
- T_{DQ} is the predominantly used metric for delay through the latch when data arrives during transparent operation

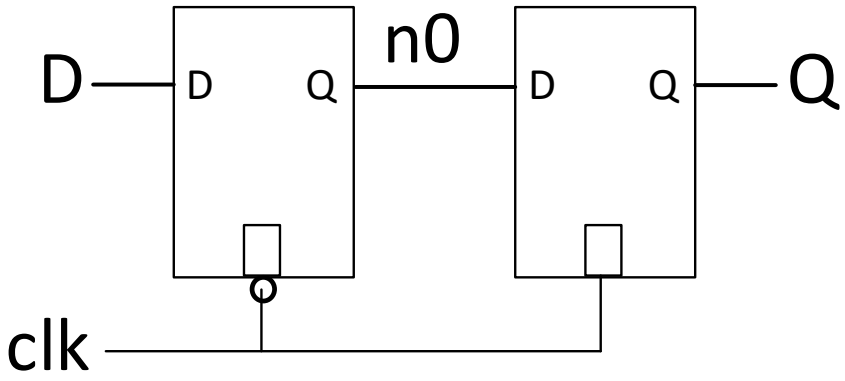
Flip Flop Structure

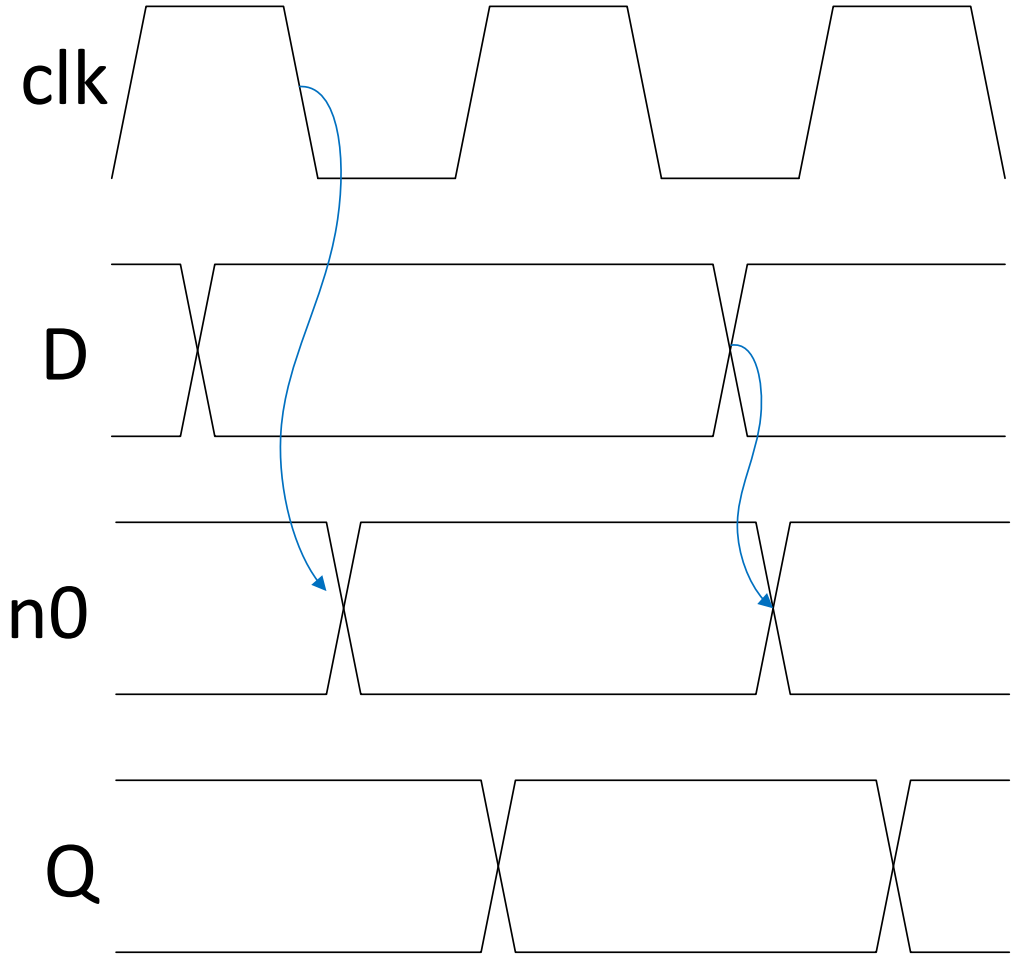




- Flip-flops: A latch-pair
- **For the rest of this class: FF → Master-Slave Flip-Flop**

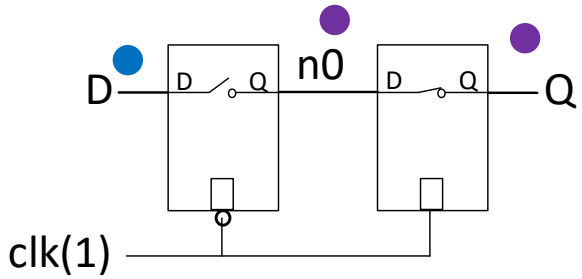
Flip Flop: Basic Operation Walkthrough

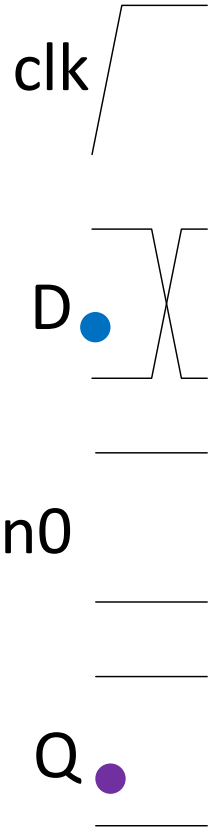




- Using B-A latch combination enables FF to sample at a clock edge, and preserve state at all other times

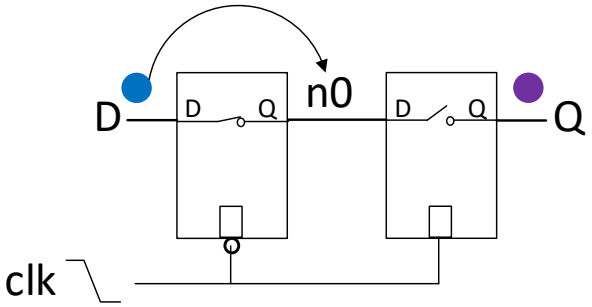
Flip Flop: Basic Operation Walkthrough

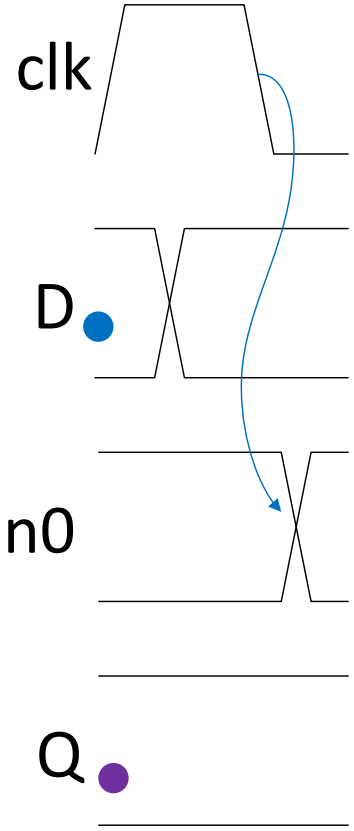




- clk='1'
  - B-latch opaque
  - Input data does not transfer to n0
  - FF holds previous state

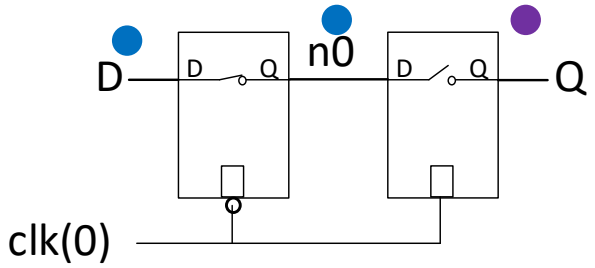
Flip Flop: Basic Operation Walkthrough

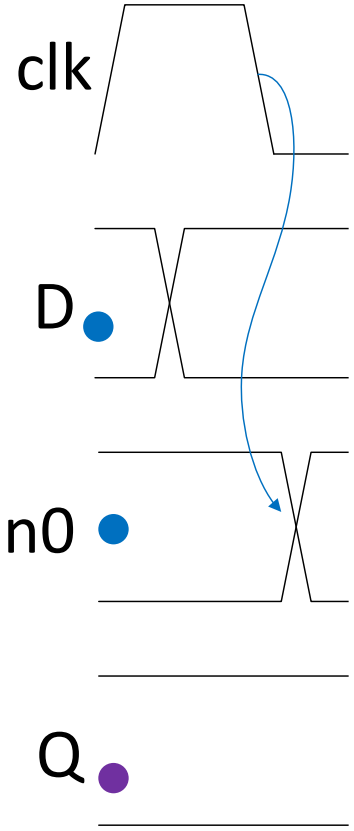




- clk= 1→0
  - B-latch now transparent
  - Data flows to n0
  - A-latch becomes opaque
  - Blocks flow of data from n0 to Q

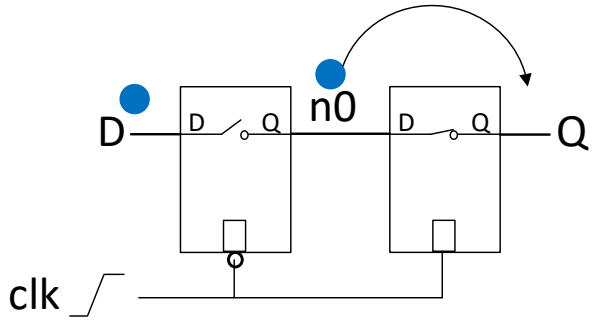
Flip Flop: Basic Operation Walkthrough

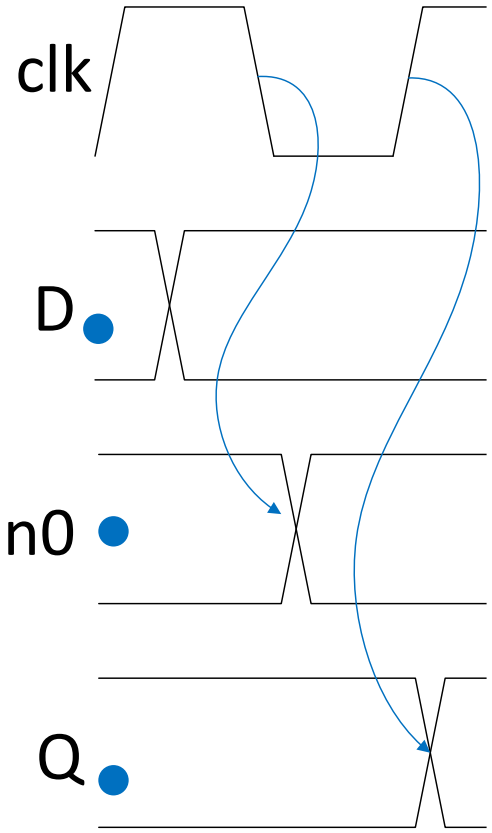




- clk= 1→0
  - B-latch now transparent
  - Data flows to n0
  - A-latch becomes opaque
  - Blocks flow of data from n0 to Q

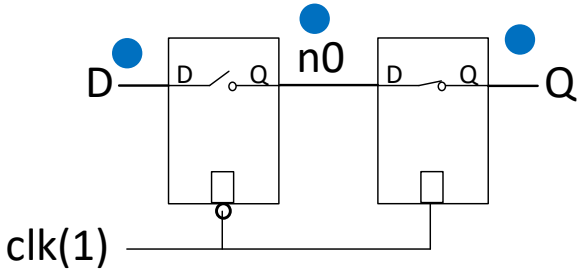
Flip Flop: Basic Operation Walkthrough

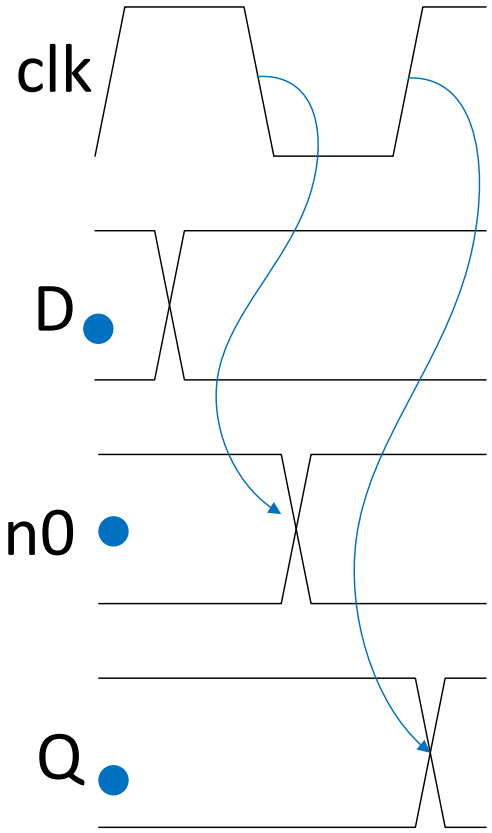




- clk= 0→1
  - A-latch becomes transparent
  - n0 data flows to output
  - B-latch becomes opaque
  - Simultaneous transparent-opaque action creates edge-based sampling and update

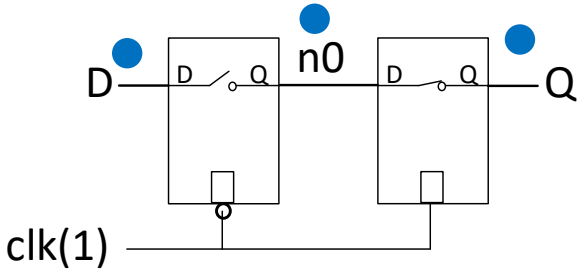
Flip Flop: Basic Operation Walkthrough

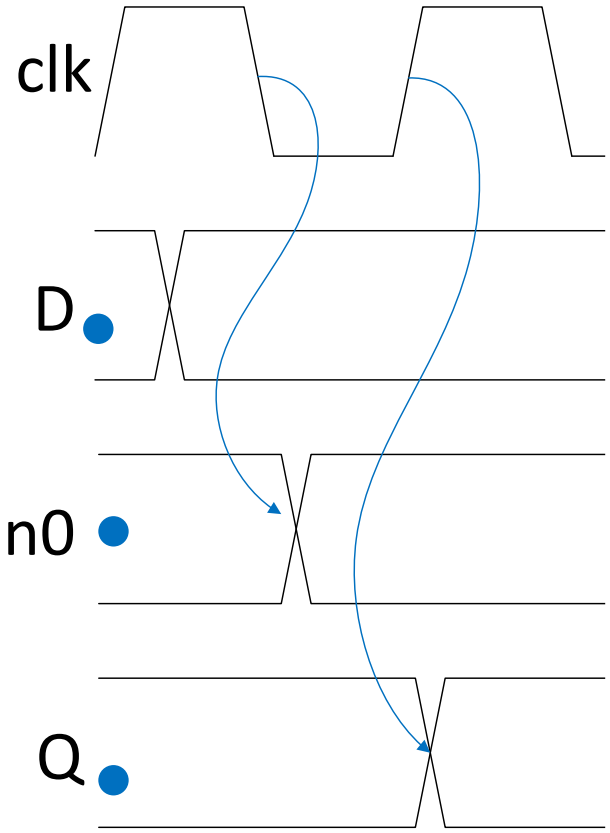




- clk= 0→1
  - A-latch becomes transparent
  - n0 data flows to output
  - B-latch becomes opaque
  - Simultaneous transparent-opaque action creates edge-based sampling and update

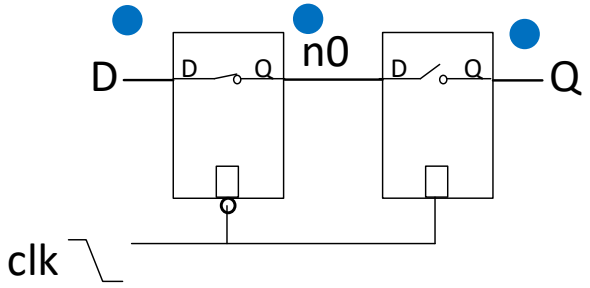
Flip Flop: Basic Operation Walkthrough

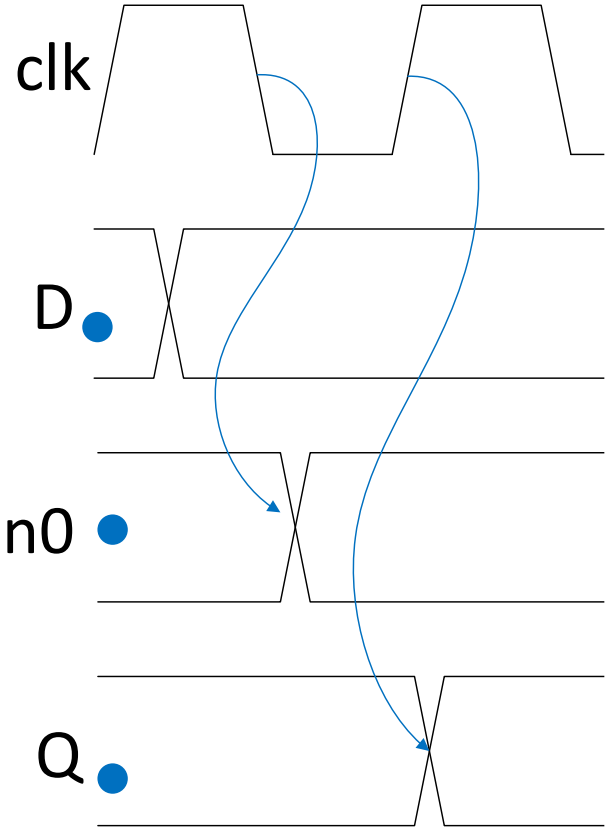




- clk= 1→0
  - A-latch opaque
  - B-latch transparent
  - Simultaneous transparent-opaque action creates edge-based sampling and update

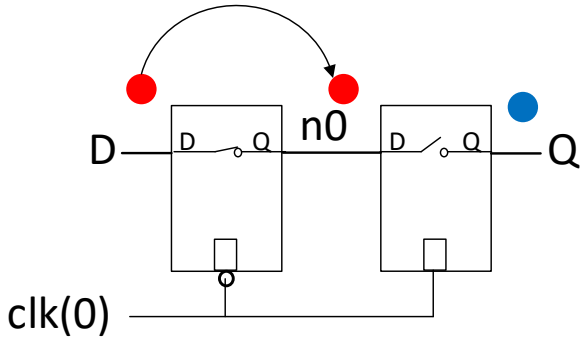
Flip Flop: Basic Operation Walkthrough

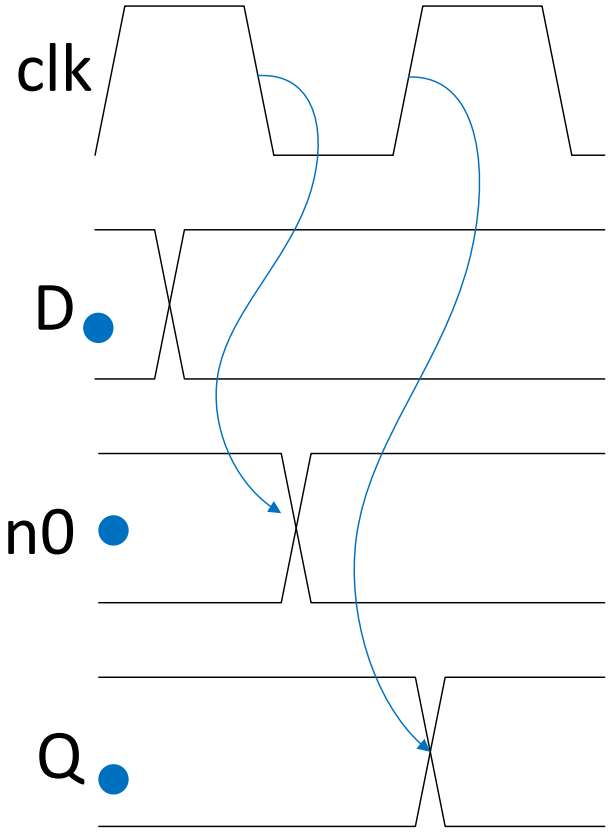




- clk= 1→0
  - A-latch opaque
  - B-latch transparent
  - Simultaneous transparent-opaque action creates edge-based sampling and update

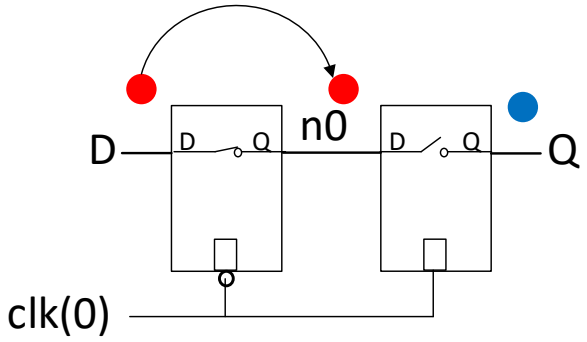
Flip Flop: Basic Operation Walkthrough

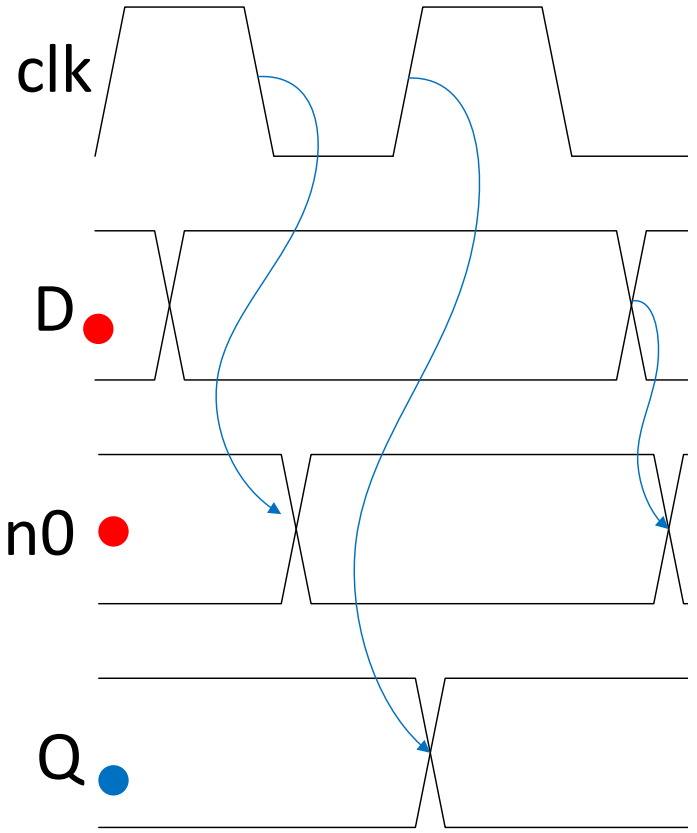




- D toggles while clk=0
  - B-latch transparent
  - Data flows to n0
  - A-latch opaque
  - Q holds previous value

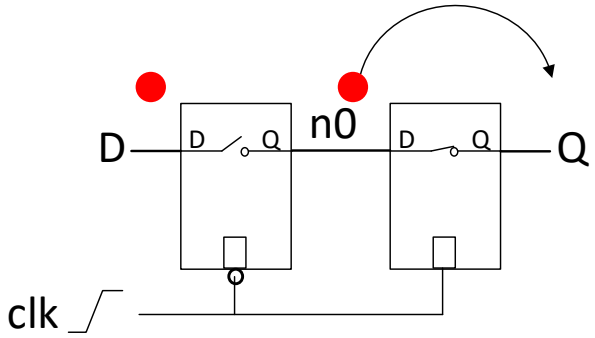
Flip Flop: Basic Operation Walkthrough

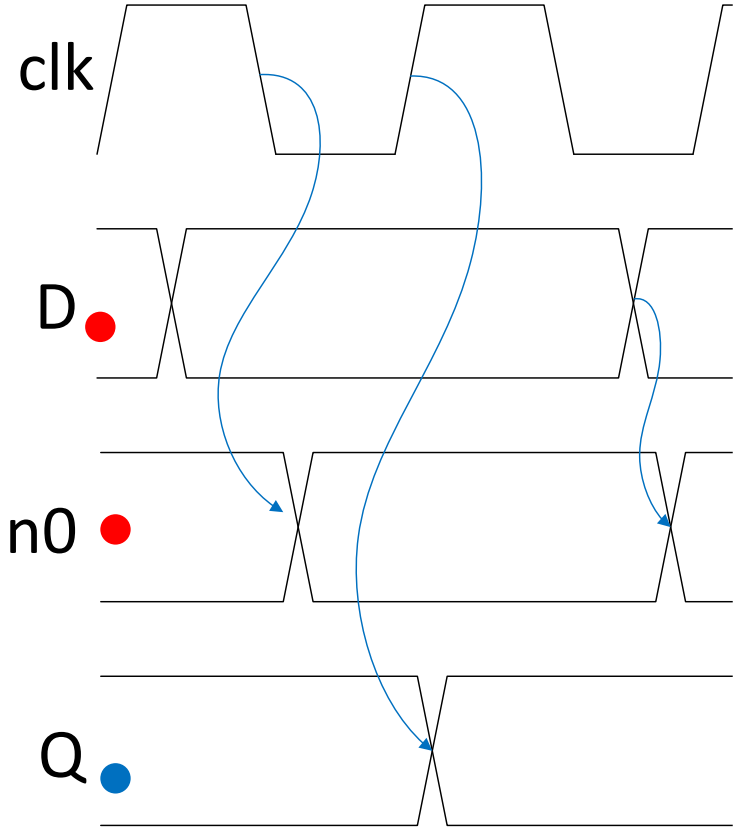




- D toggles while clk=0
  - B-latch transparent
  - Data flows to n0
  - A-latch opaque
  - Q holds previous value

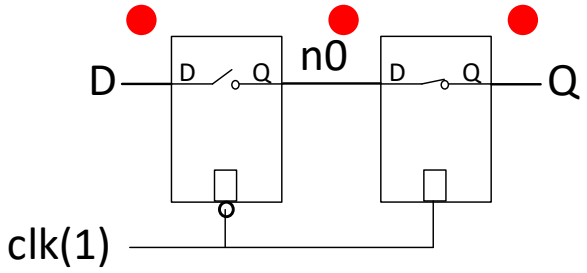
Flip Flop: Basic Operation Walkthrough

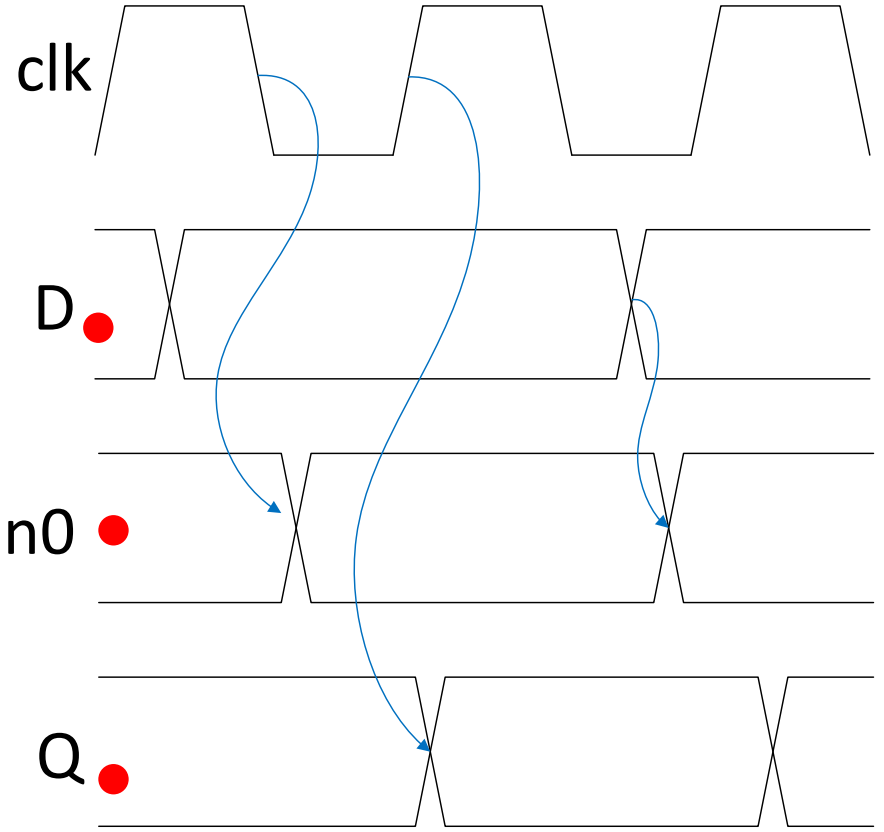




- clk 0→1
  - B-latch opaque
  - A-latch transparent
  - Data at n0 flows to Q

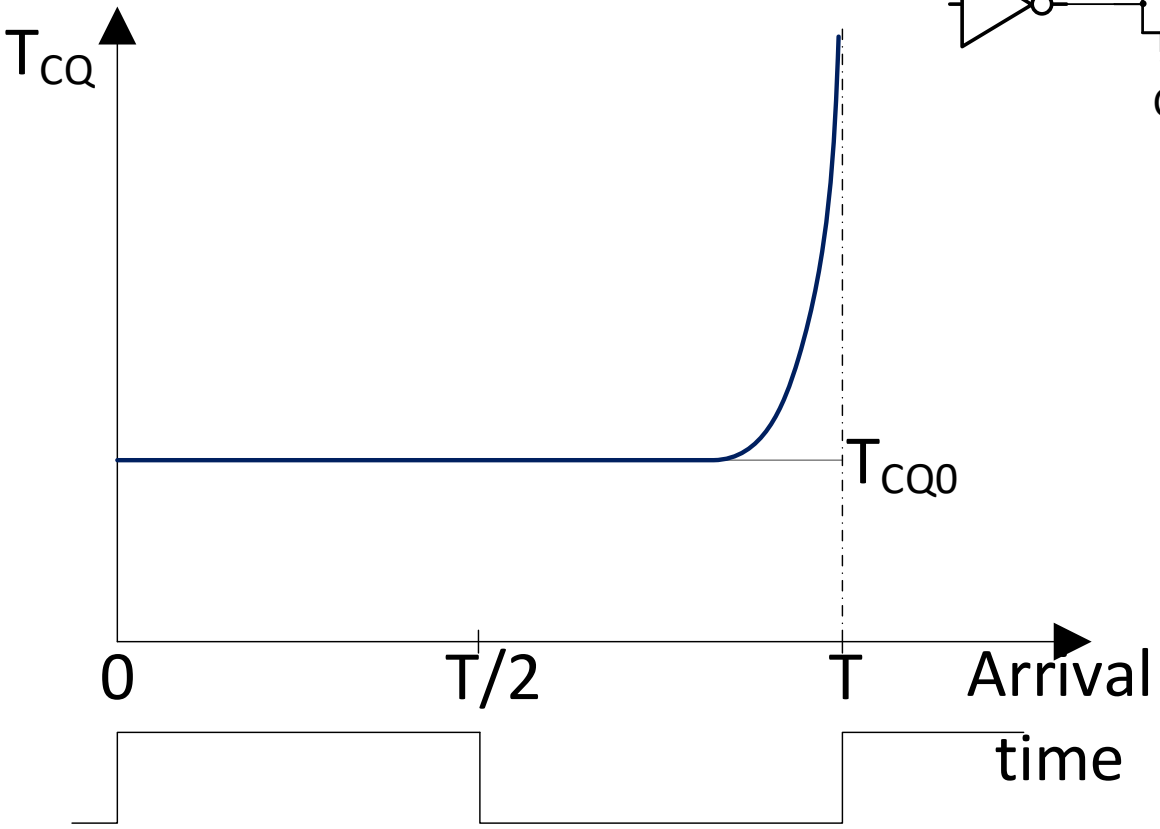
Flip Flop: Basic Operation Walkthrough

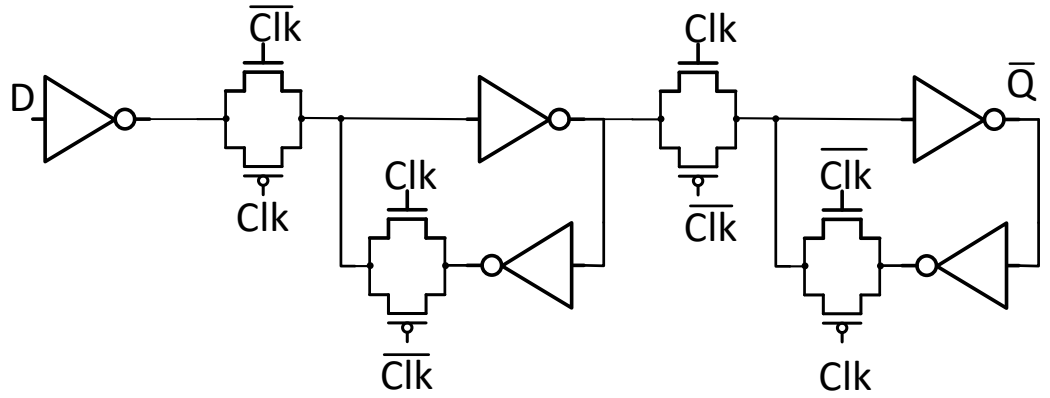




- clk
  - B-latch transparent
  - Data flows to n0
  - A-latch opaque
  - Q holds previous value

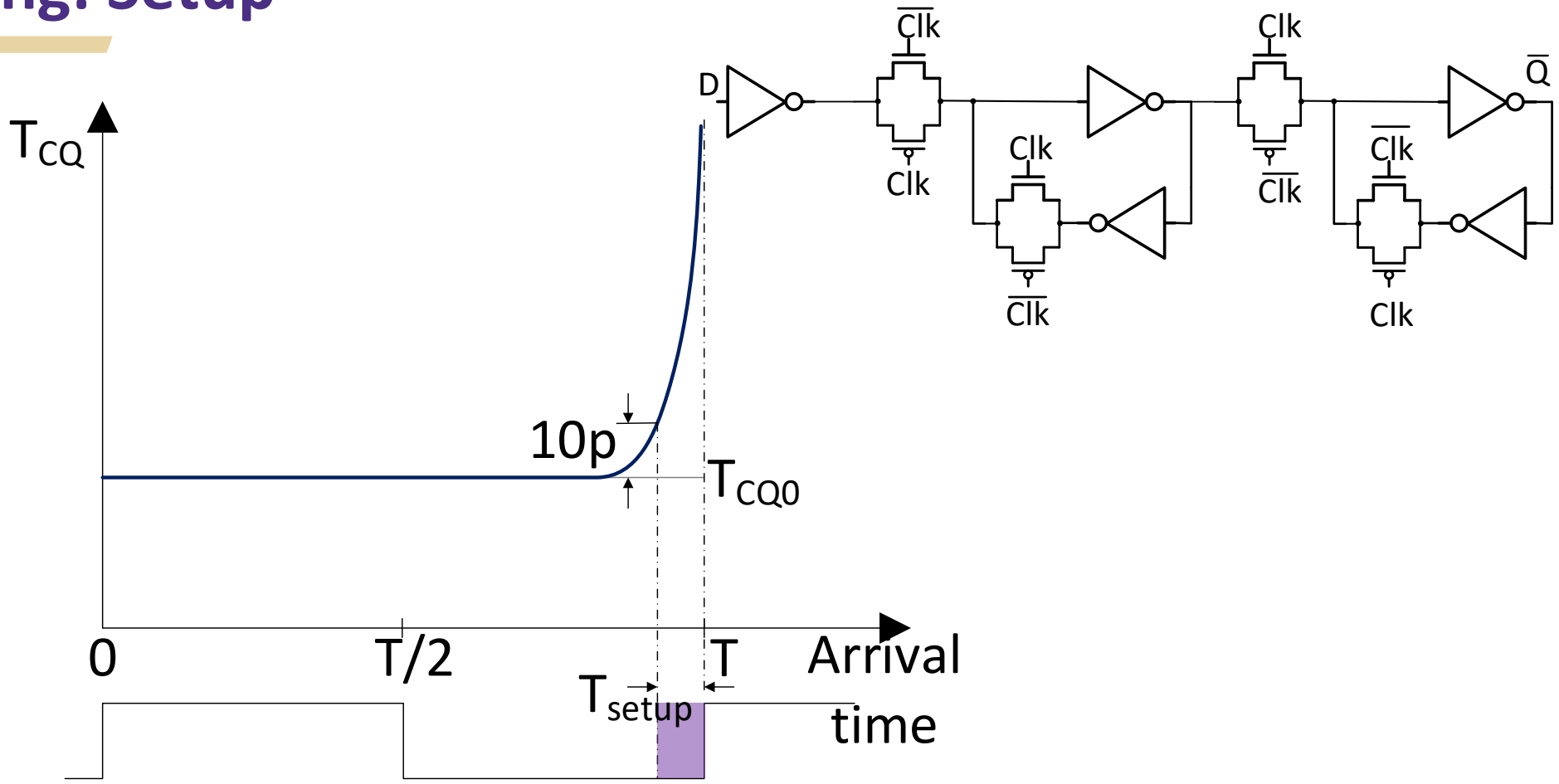
Timing: Setup





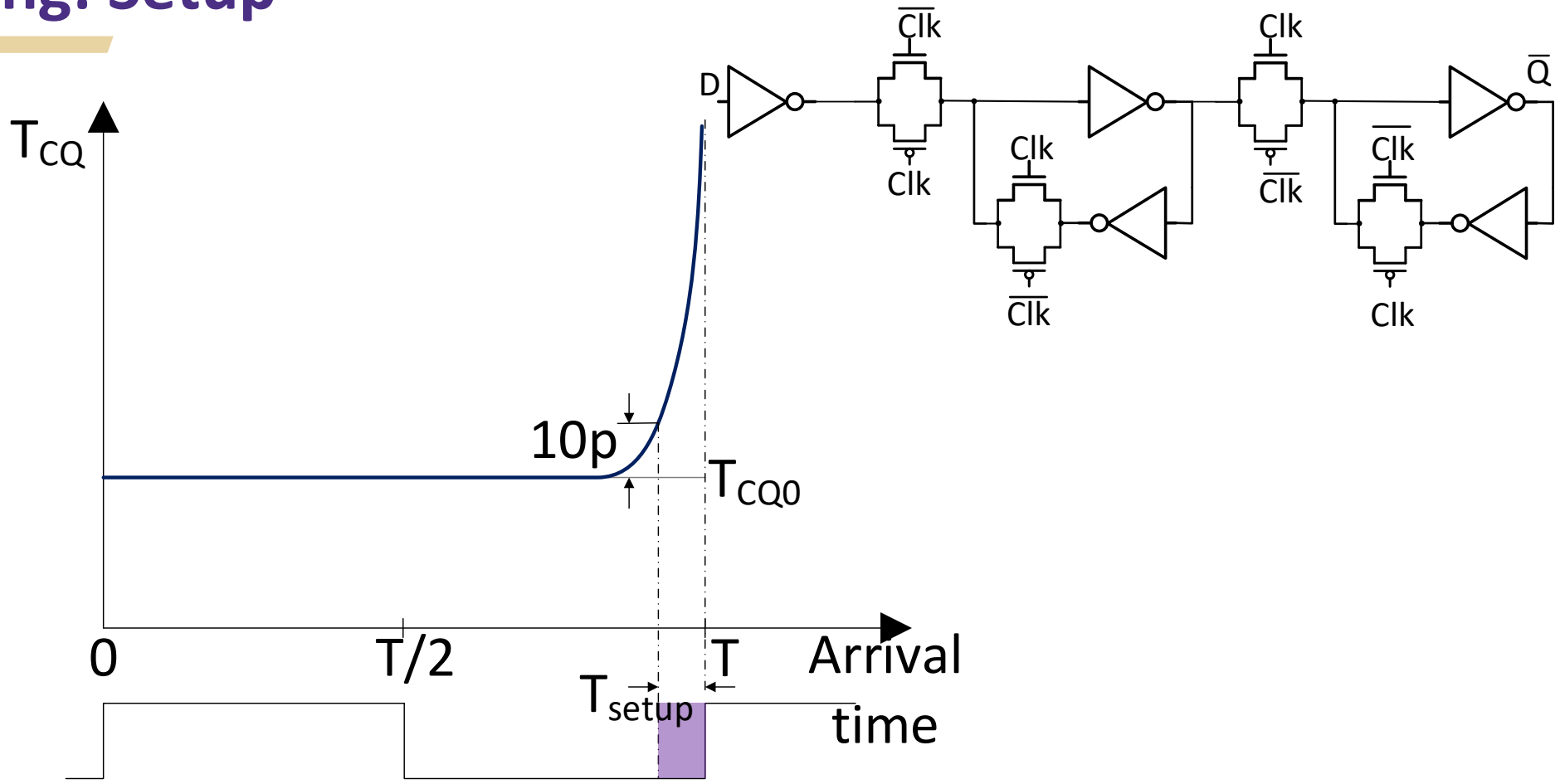
- Data must arrive at FF input T_{setup} before latching edge
  - Allow Data to flow through B-Latch, arrive at opaque A-latch before clk →1
  - Later data arrival causes reduced drive through slave latch, ↑n3, Q_bar delay
  - Even more delay.. Clk→1 while(or before) n0 transitions. Feedback restores old state

Timing: Setup



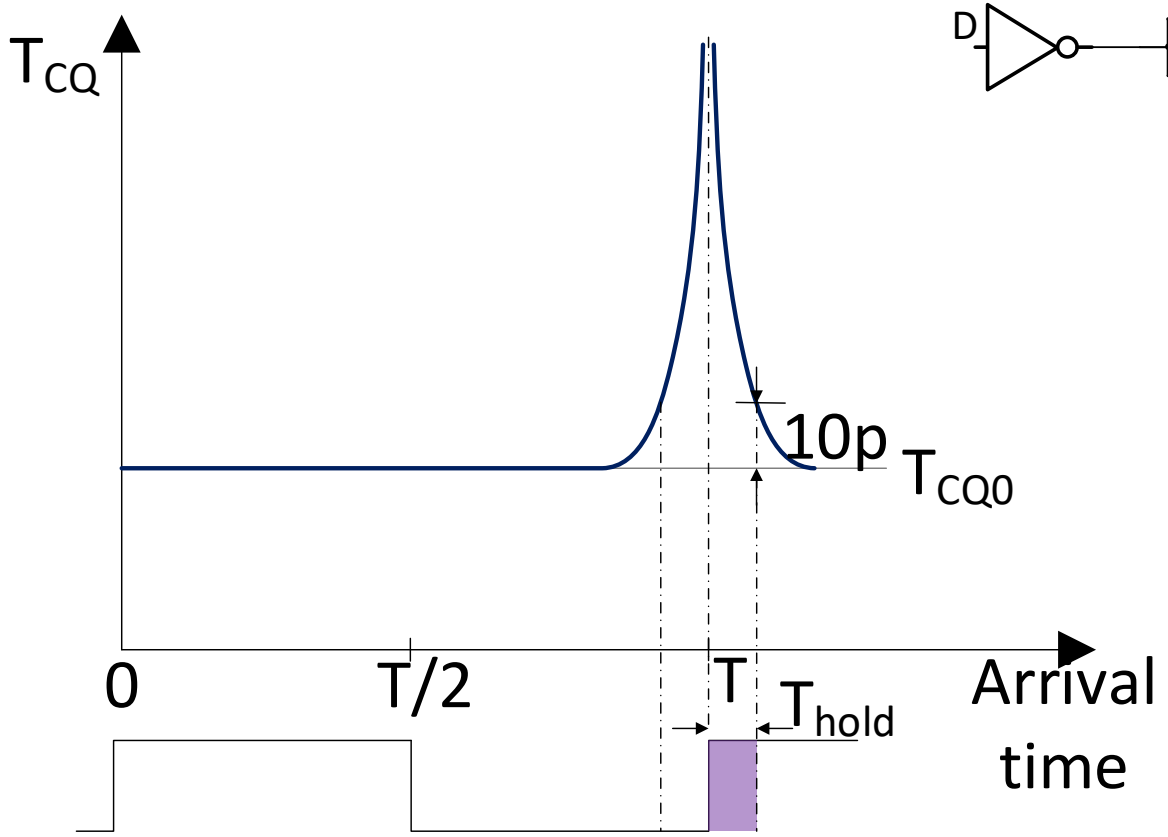
- Requirement for data arrival before clock edge varies
  - In this class, setup is violated if T_{CQ} "pushout" exceeds 10ps

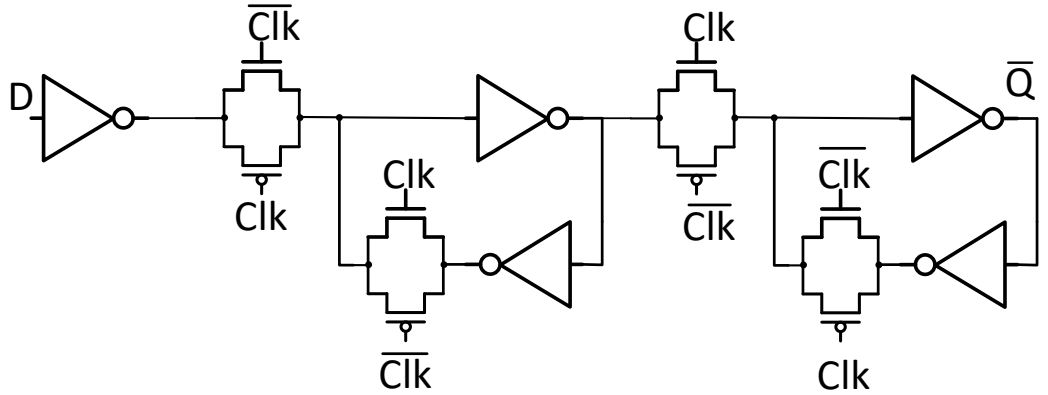
Timing: Setup

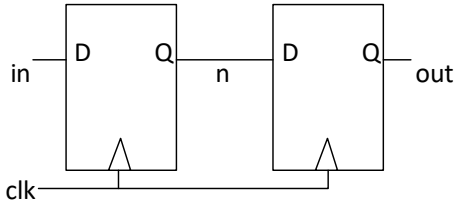


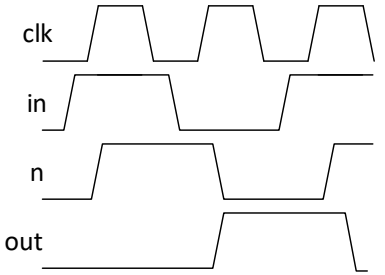
- Requirement for data arrival before clock edge varies
  - In this class, setup is violated if T_{CQ} "pushout" exceeds 10ps
  - Why not define T_{setup} based on correctly latched data?

Timing: Hold 1



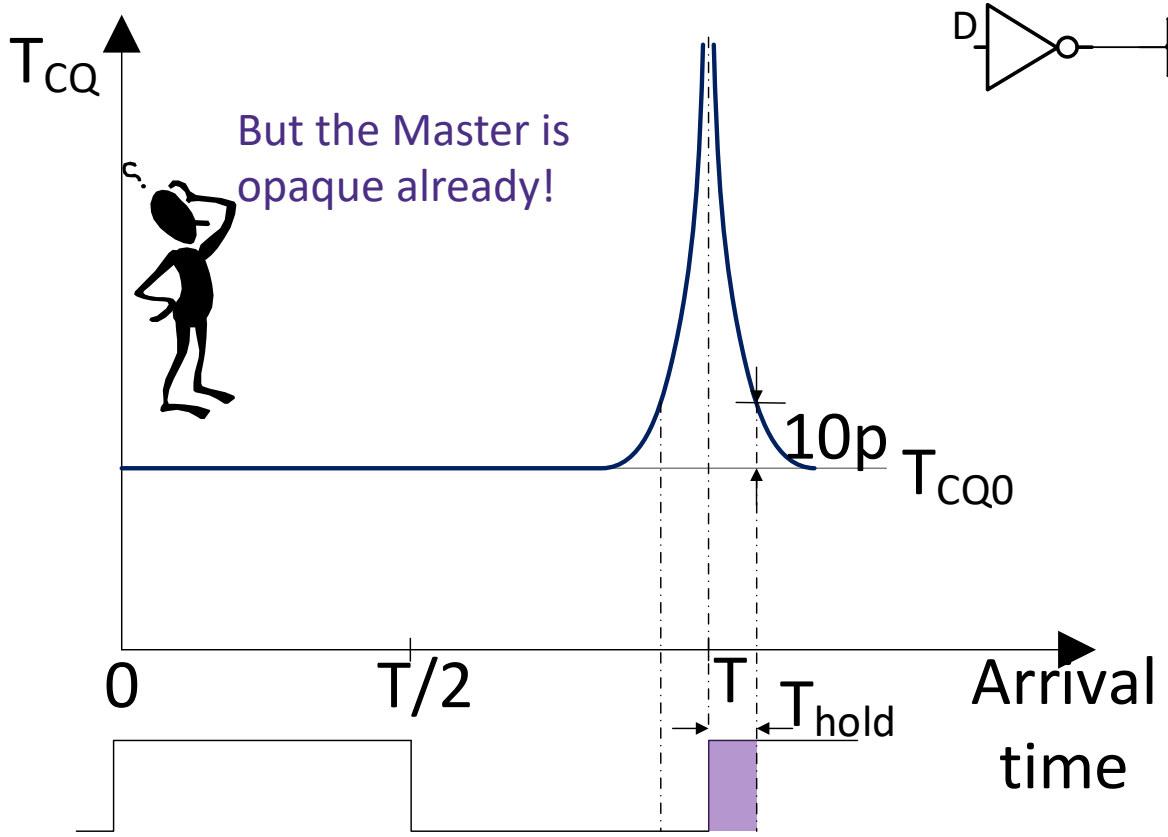


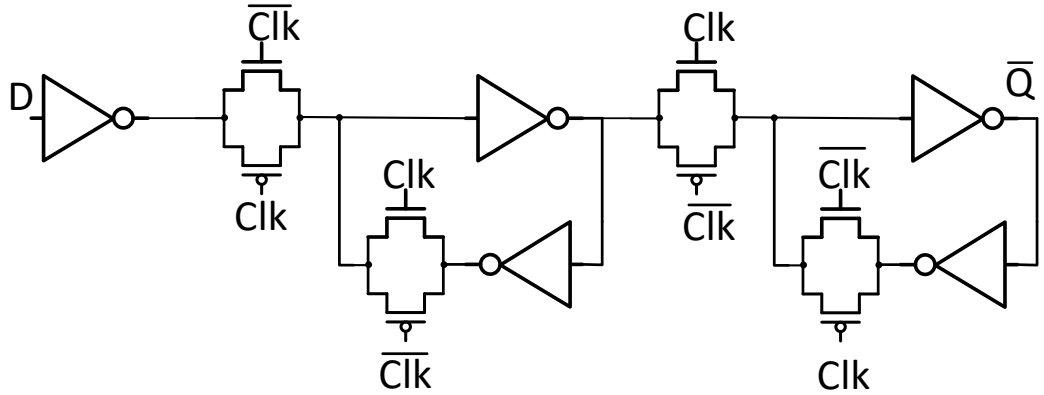


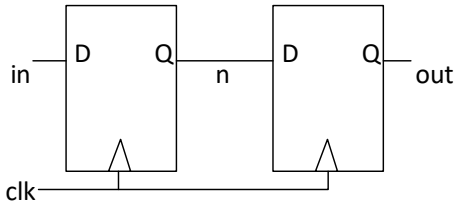


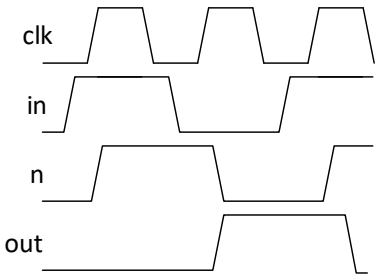
- Data must remain stable T_{hold} after latching edge
- If fresh data arrives at the flop input, before the hold time
  - Successively earlier data arrival causes reduced slave latch drive, ↑n3, Q_bar delay
  - Still earlier data makes input appear to be a glitch, and not correctly latched
  - Hold time violations also referred to as a race condition (fresh data races through, overwriting correct data)

Timing: Hold 1



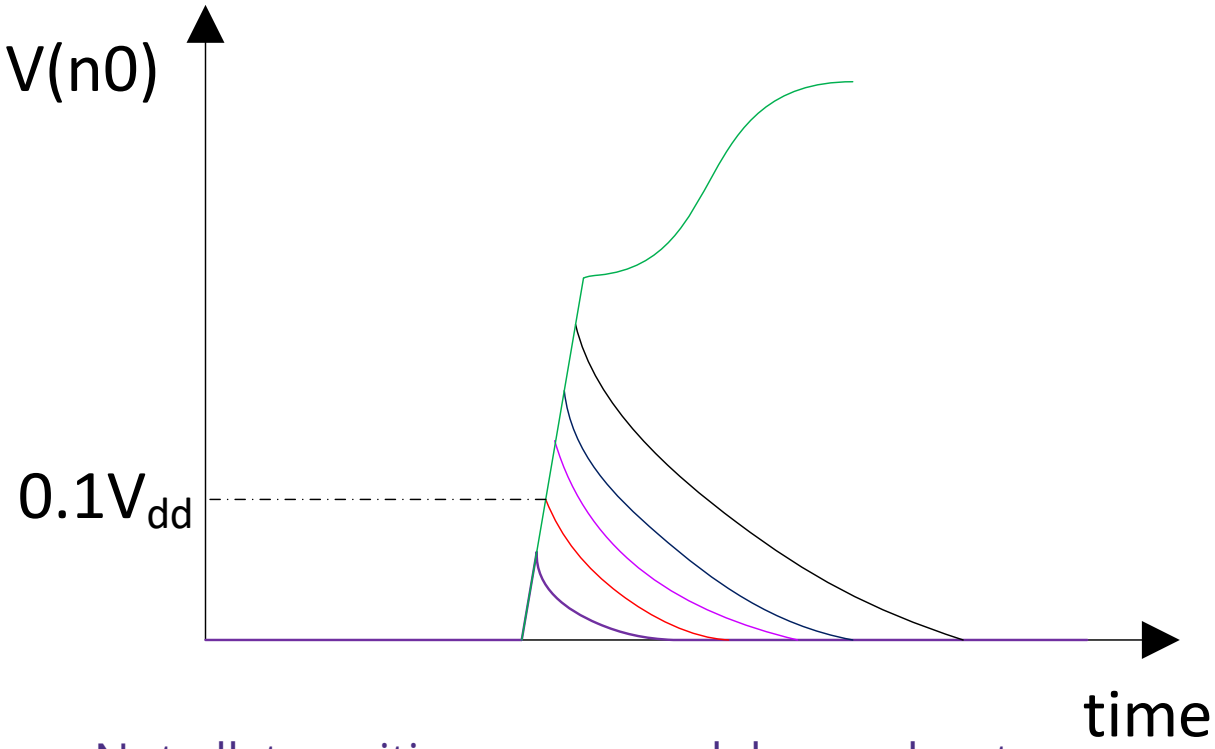


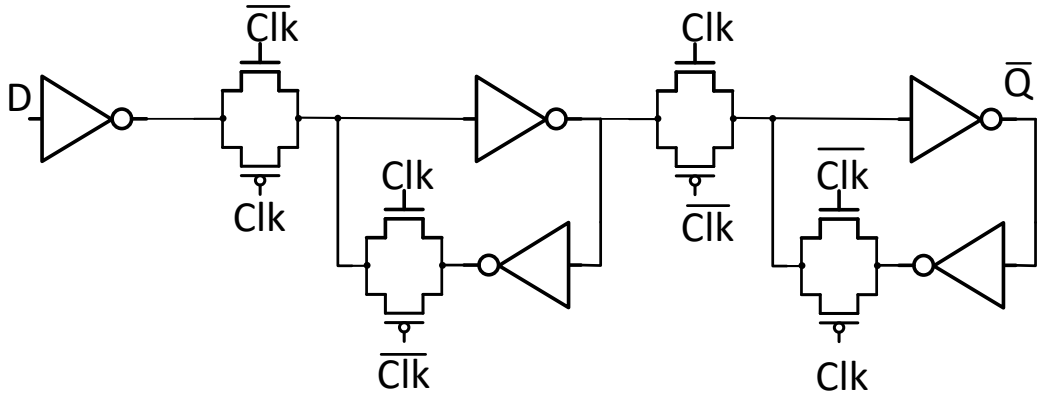


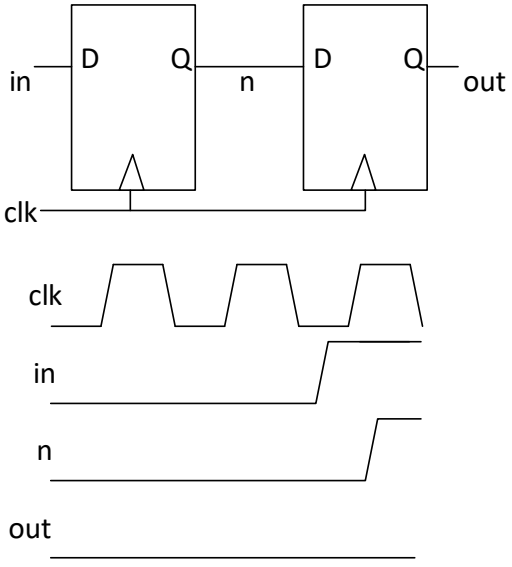


- Data must remain stable T_{hold} after latching edge
- If fresh data arrives at the flop input, before the hold time
  - Successively earlier data arrival causes reduced slave latch drive, ↑n3, Q_bar delay
  - Still earlier data makes input appear to be a glitch, and not correctly latched
  - Hold time violations also referred to as a race condition (fresh data races through, overwriting correct data)

Timing: Hold 2







- Not all transitions cause a delay pushout
- Consider data-sequence : 0→0→1
  - If "1" begins to violate hold time and arrives successively earlier, before tx-gate is off
  - n0 begins to charge up but only until sampling master tx-gate is still on
  - Once tx-gate is off, feedback restores value of n0 (if n0 is still early in the charge-up state)
- 10% glitch amplitude on state retention node (n0 here) considered glitching limit

T_{setup} and T_{hold} review

- What happens when data arrives after T_{setup} and before T_{hold}?
  A. Data is undefined
  B. Complementary data is latched
  C. T_{cq} = infinite

- T_{setup} : Time before sampling edge that data must arrive (in a 50% sense)
- T_{hold} : Time after sampling edge that **latched** data must remain stable (in a 50% sense)
- T_{setup}, T_{hold} are not always constrained to be positive
  - In a master-slave flop, T_{hold} is often negative

Analyze This…

- Push out the slave clock relative to the master clock
  - What are the benefits
  - What are the risks

Analyze This…

- Push out the slave clock relative to the master clock
  - What are the benefits
  - What are the risks
- Push out the master clock relative to the slave clock
  - What are the benefits
  - What are the risks

Reading/Thinking assignment

- Think about how you can cause a hold violation in a Master-slave latch…what actually goes on???What implication does it have on the hold-times of M-S latches?