CSEE 3827: Fundamentals of Computer Systems, Spring 2011

4. Sequential Circuit Elements

Prof. Martha Kim (martha@cs.columbia.edu)
Web: http://www.cs.columbia.edu/~martha/courses/3827/sp11/

Outline (H&H 3.1-3.3, 3.5, 5.4)

- Sequential circuit elements
  - Latches
    - SR
    - D
  - Flip-Flops
    - D
    - Enabled
    - Resettable
  - Sync v. Async
  - Timing of Sequential Logic
  - Building blocks
    - Registers
    - Counters
    - Shift Registers

Introduction

- Outputs of sequential logic *depend on current and prior input values* – it has memory.

- Some definitions:

  - **State**: all the information about a circuit necessary to explain its future behavior

  - **Latches and flip-flops**: state elements that store one bit of state

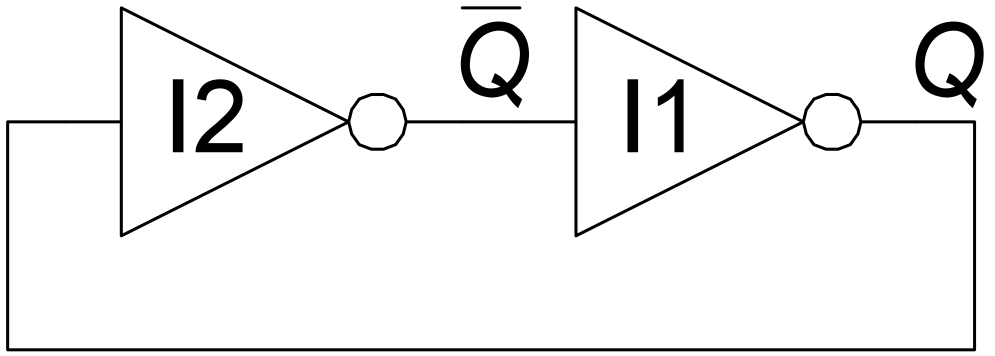
  - **Synchronous sequential circuits**: combinational logic followed by a bank of flip-flops

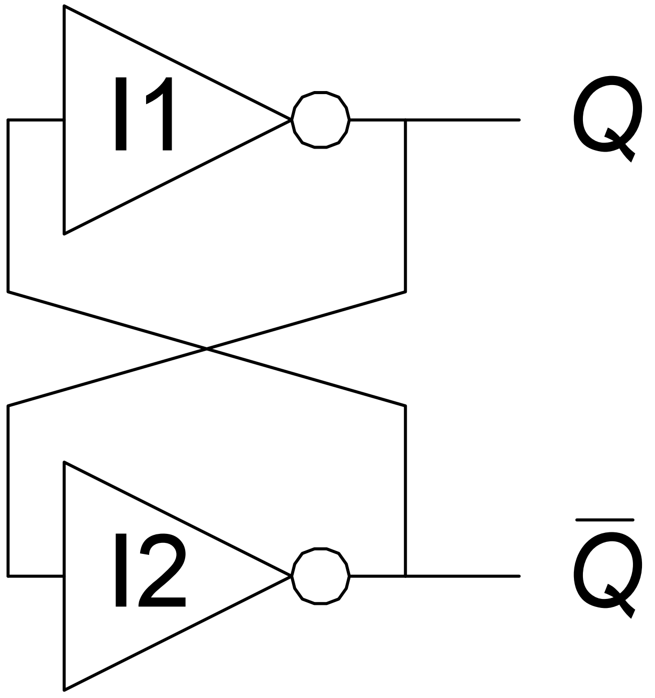
Sequential Circuits

- Give sequence to events
- Have memory (short-term)
- Use feedback from output to input to store information
- Circuit state is stored in **state elements**, e.g.,
    - Bistable circuit
    - SR Latch
    - D Latch
    - D Flip-flop

Bistable Circuit

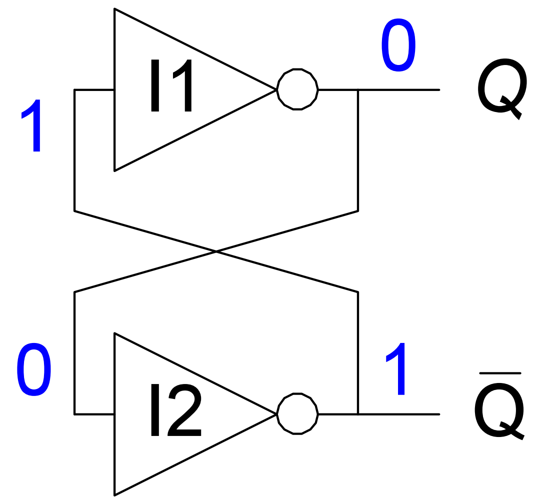
- Fundamental building block of other state elements
- Two outputs: Q, `Q
- No inputs



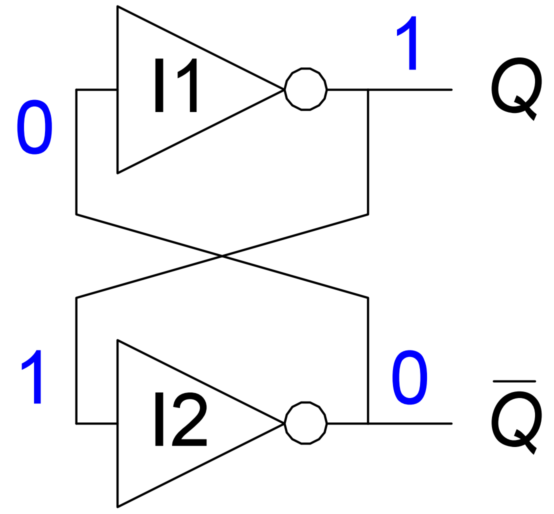


Bistable Circuit Analysis

- Consider the two possible cases:
  - Q = 0: then `Q = 1 and Q = 0 (consistent)



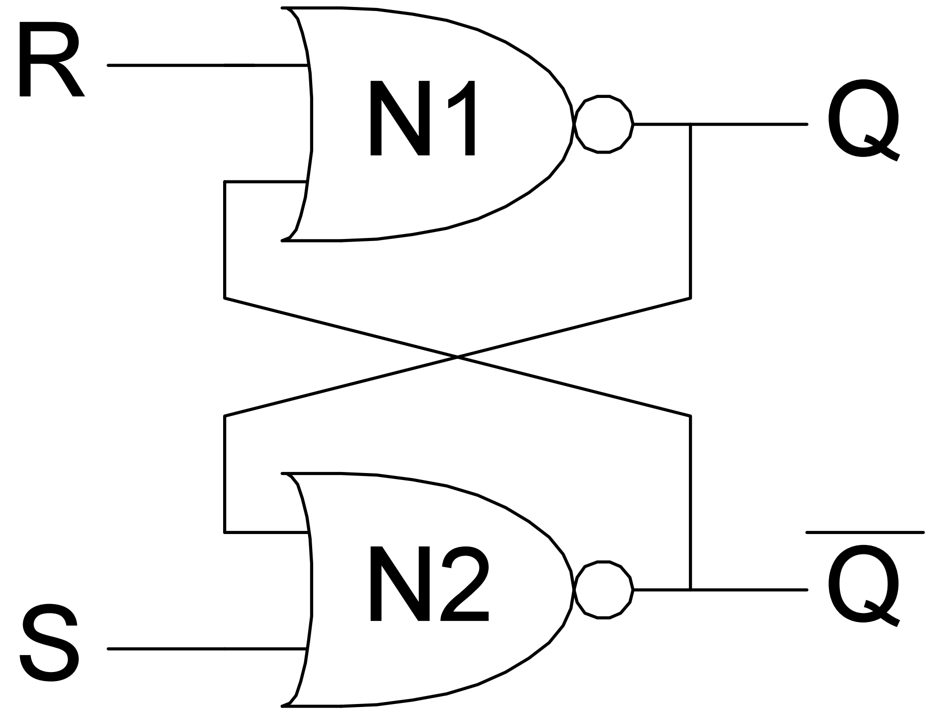
  - Q = 1: then `Q = 0 and Q = 1 (consistent)



- Bistable circuit stores 1 bit of state in the state variable, Q (or `Q)
- But there are ***no inputs to control the state***

SR (Set/Reset) Latch

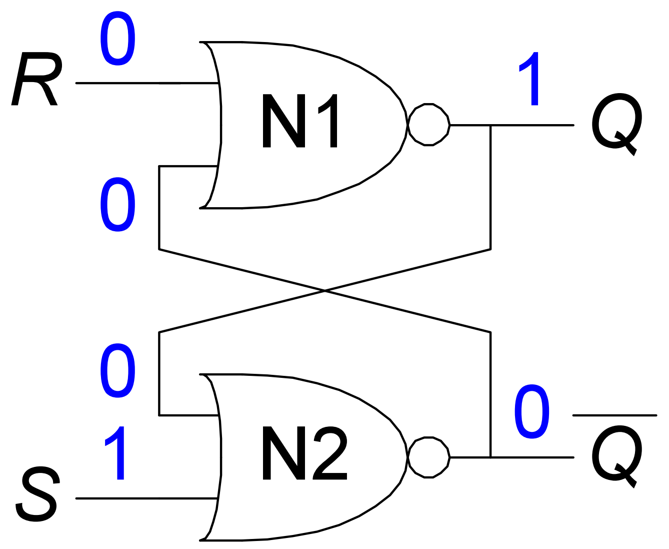
- SR Latch



- Consider the four possible cases:
  - S = 1, R = 0
  - S = 0, R = 1
  - S = 0, R = 0
  - S = 1, R = 1

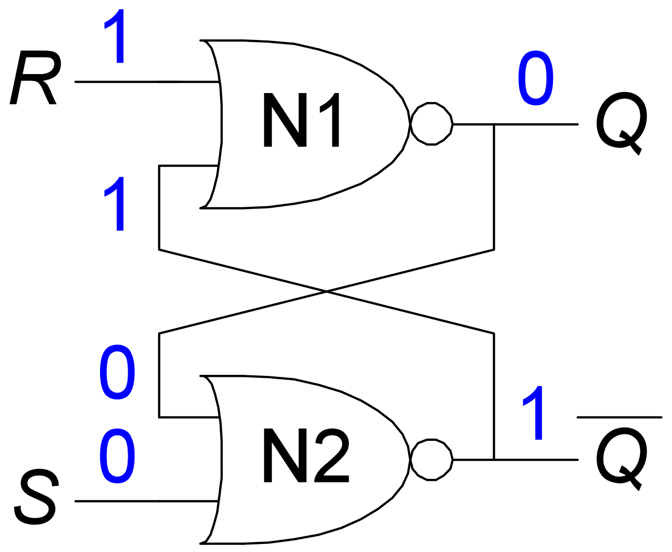
SR Latch Analysis

- S = 1, R = 0: then Q = 1



set

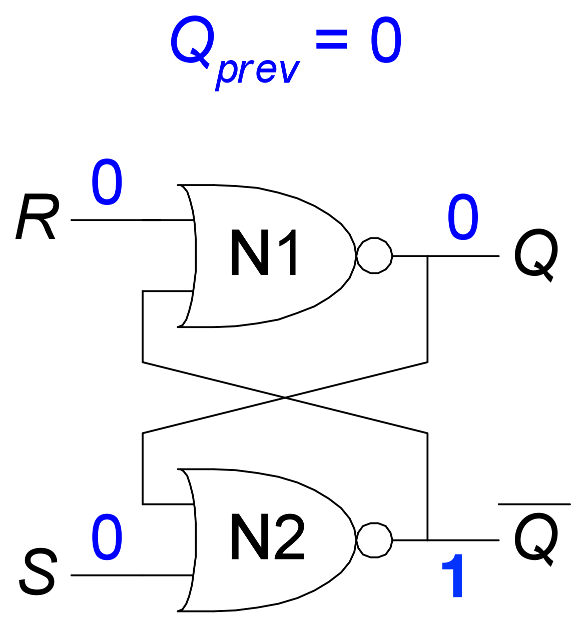
- S = 0, R = 1: then Q = 0

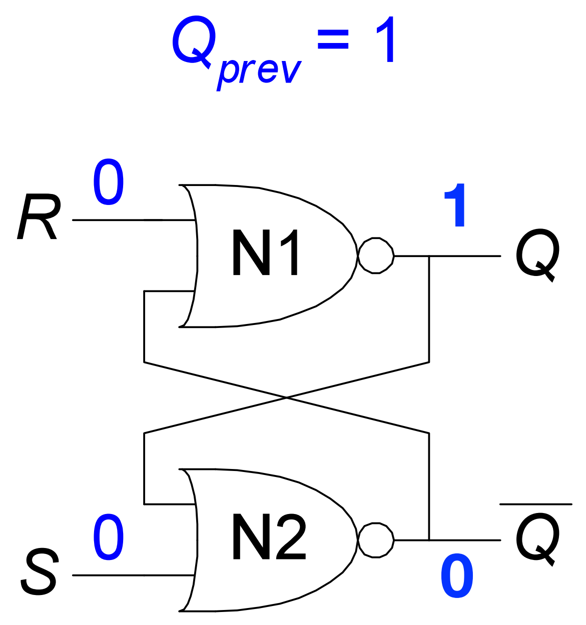


reset

SR Latch Analysis

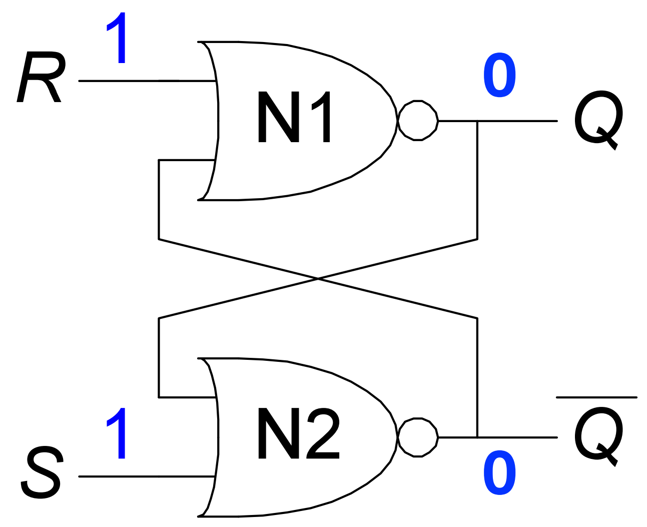
- S = 0, R = 0: then Q = Qprev





memory!

- S = 1, R = 1: then Q = 0 and `Q = 0

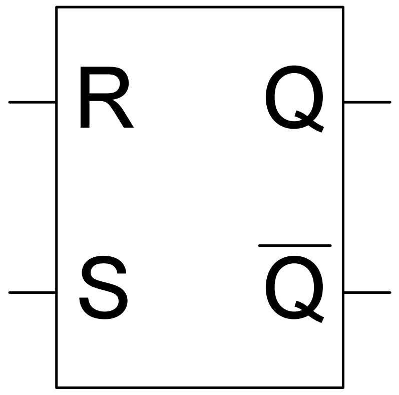


Q=`Q
Invalid state

SR Latch Symbol

- SR stands for Set/Reset Latch, storing one bit of state (Q)
- Control the value being stored with S, R inputs
  - Set (“make the output 1”): S=1,R=0,Q=1
  - Reset (“make the output 0”): S=0,R=1,Q=0

SR Latch Symbol

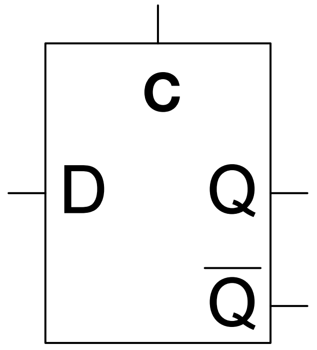


- Must do something to **avoid invalid state** (S = R = 1)

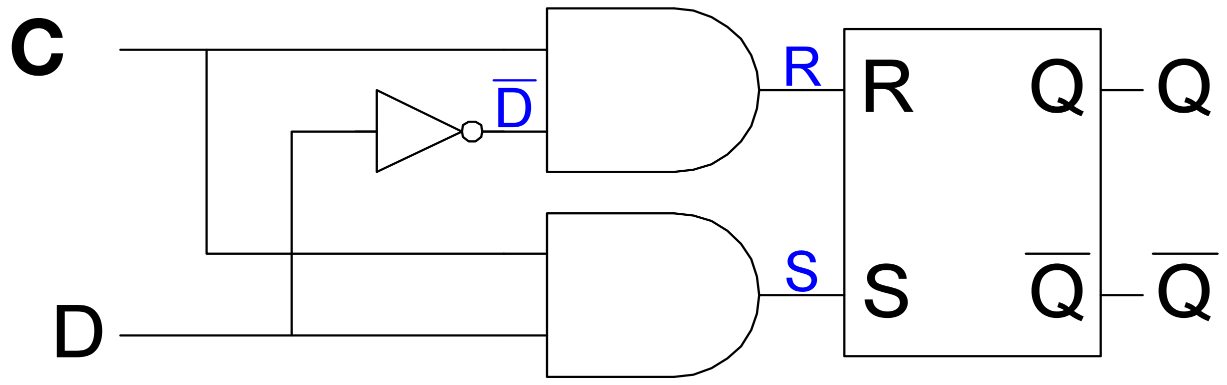
D Latch

- Two inputs: C, D
  - **C**: controls *when* the output changes
  - **D (the data input)**: controls *what* the output changes to
- Function
  - When C = 1, D passes through to Q (the latch is transparent)
  - When C = 0, Q holds its previous value (the latch is opaque)
- Avoids invalid case when Q ≠ NOT `Q

D Latch Symbol



D Latch Internal Circuit



C	D	`D	S	R	Q	`Q
0	X					
1	0					
1	1					

Where we are, where we are headed

Latches are circuits that can store “state”

- set the latch to a value (0 or 1)
- put the latch in a “same value” mode to hold the value

To do complicated computations

- intermediate “state” must be maintained
- various steps of the computation must be coordinated

Q: How to coordinate computations and the changing of state values across lots of different parts of a circuit

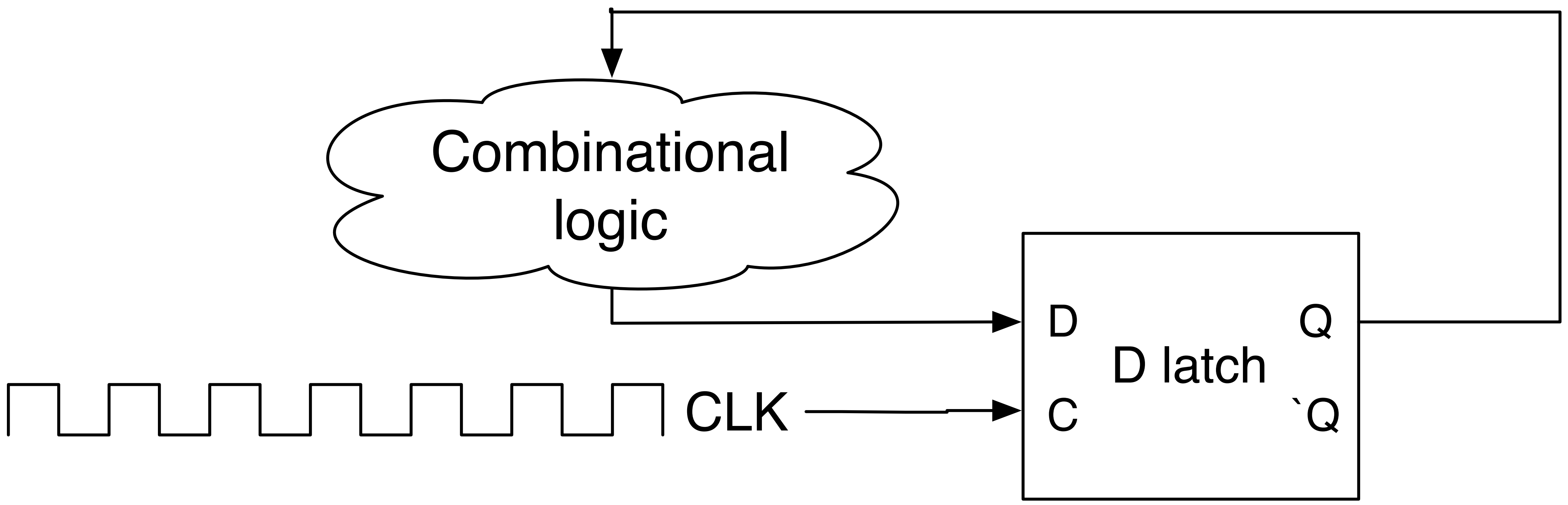
A: Introduce a **clocking** mechanism

- each clock pulse, combinational computations can be performed, results stored (in latches)

Q: How to introduce clocks into latches?

flip flops: latches on a clock

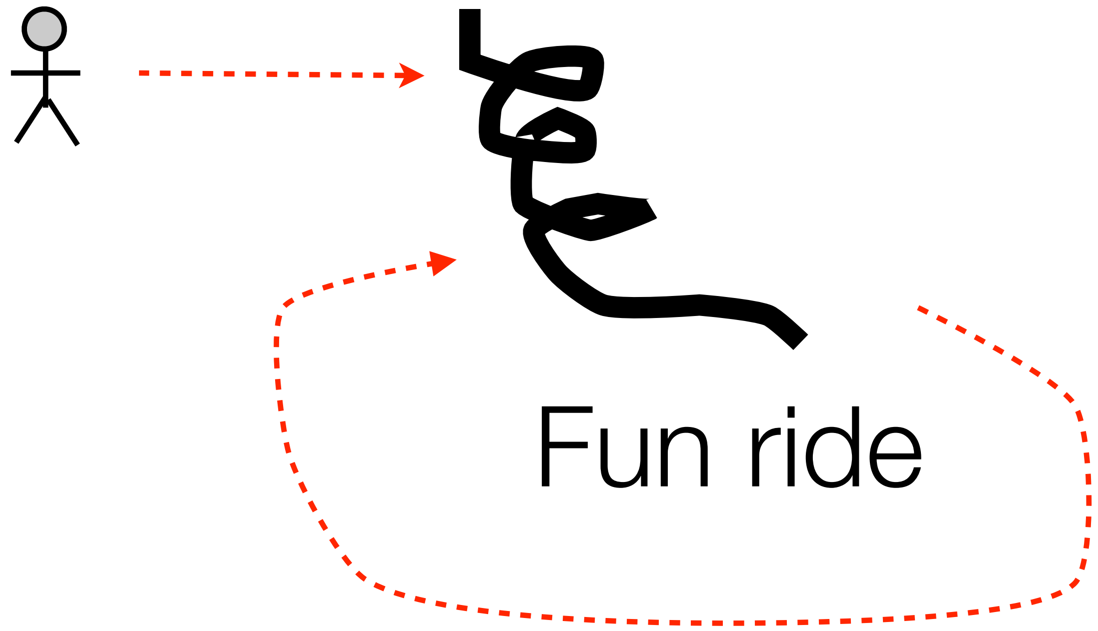
- A straightforward latch is not safely (i.e., predictably) synchronous



- The problem is transparency of latches: as soon as the input changes, at some time later the output will change

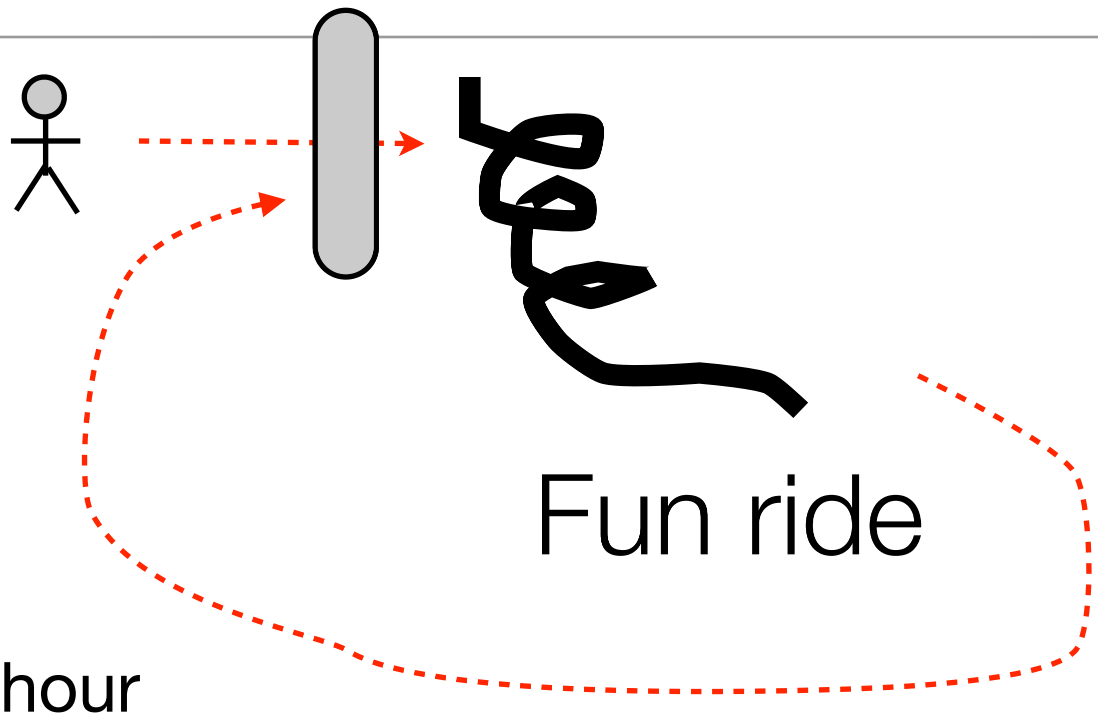
- Flip flops are designed so that outputs will **not** change within a single clock pulse

Implementing the 1-ride-per-hour



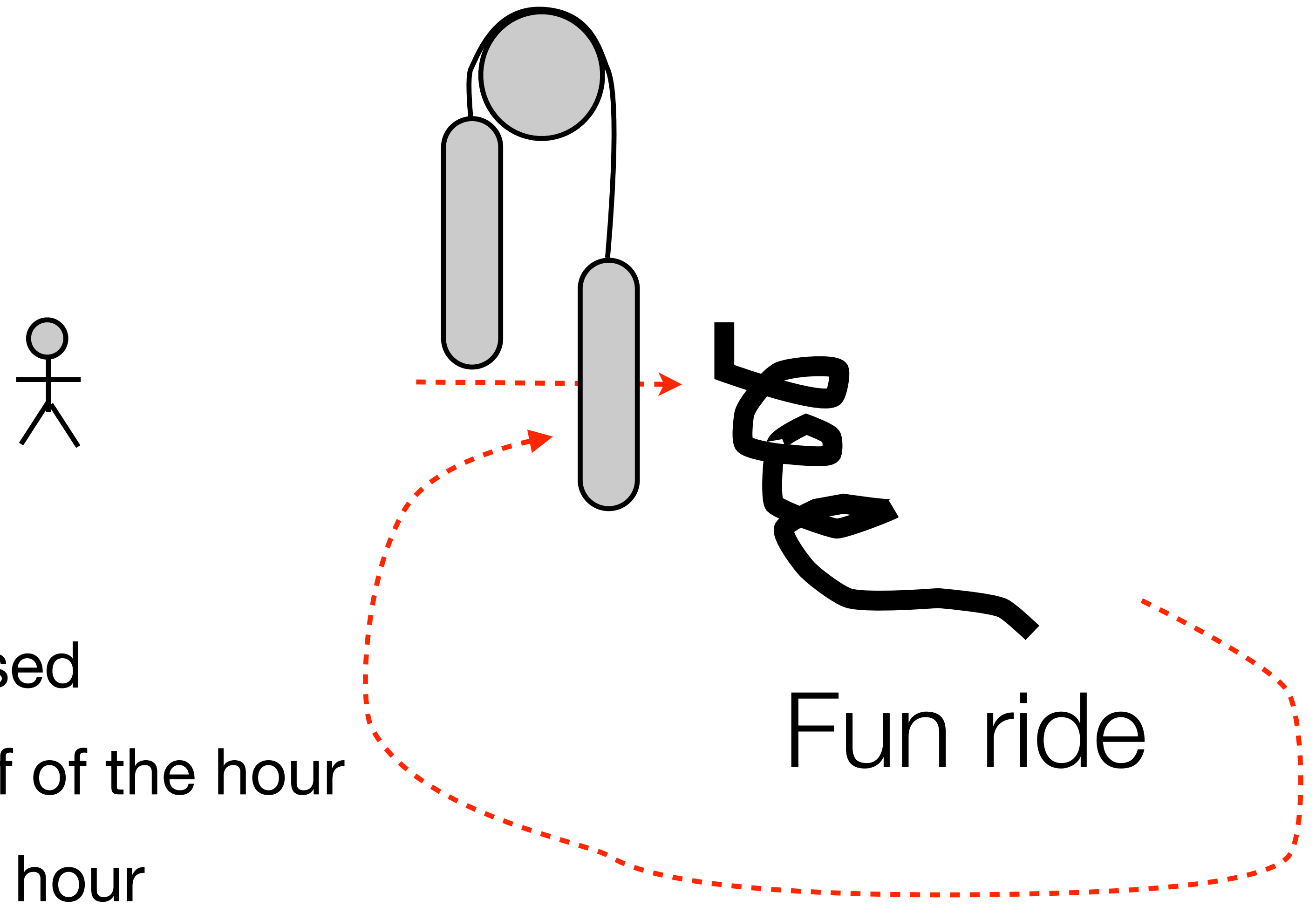
- Suppose there is
  - Fun ride
  - People should only ride at most once per hour
  - How to stop someone from riding too often?

Solution #1: Build a gate



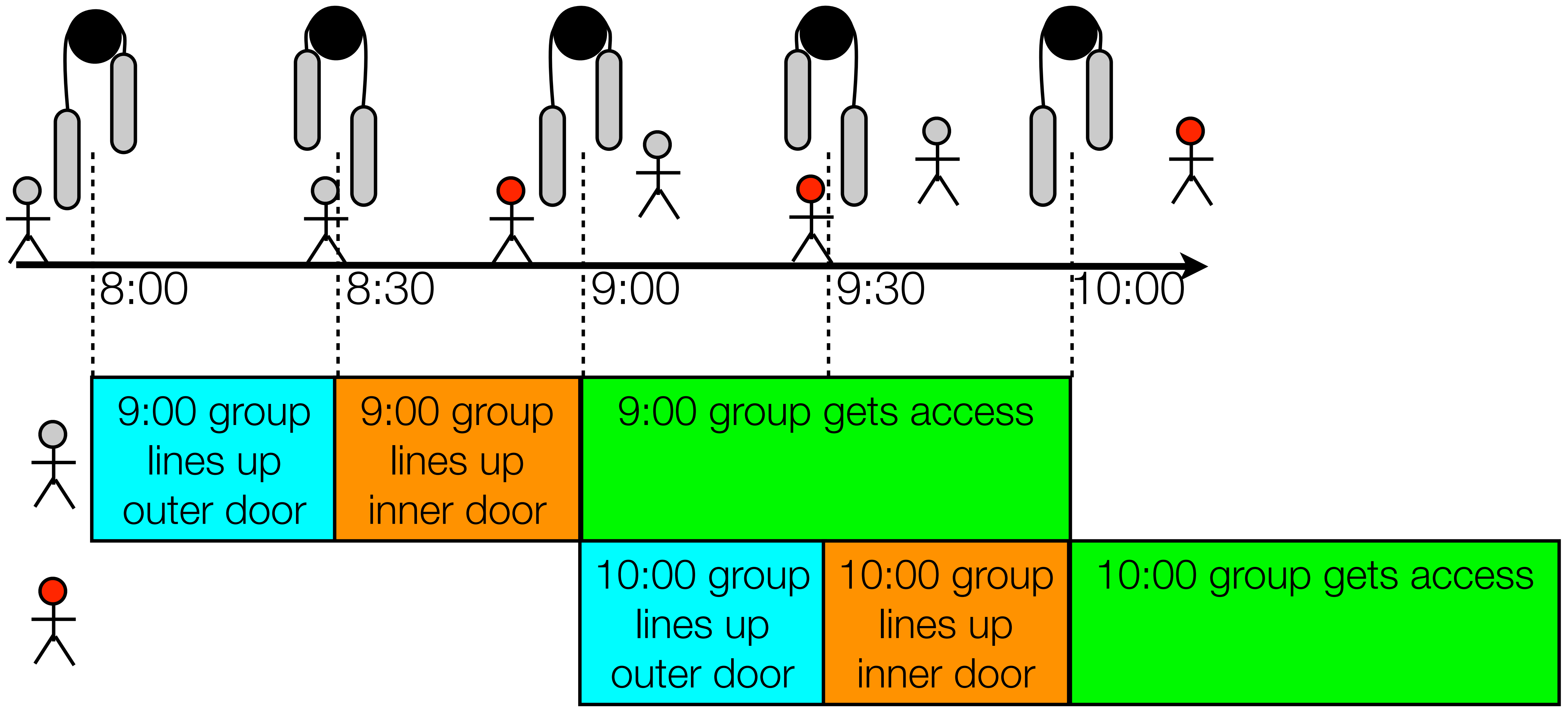
- Gate opens once per hour
    - Problem: how long to leave gate open?
    - Too short: not everyone might make it through in time (limits rideability)
    - Too long: “fast” person can go through, ride, and get through gate again

Solution #2: Pair of alternating gates

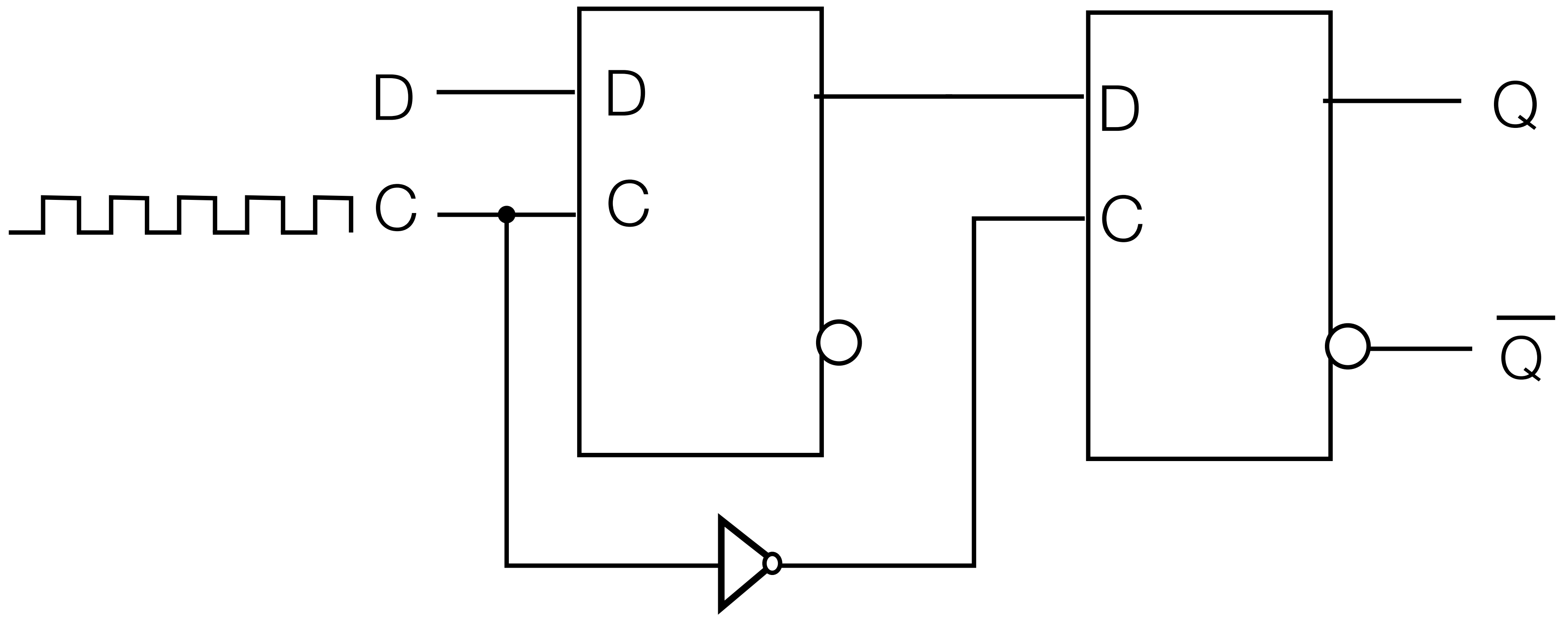


- Gates alternate being open and closed
  - 1st gate: open on the bottom half of the hour
  - 2nd gate: open on top half of the hour
  - Anyone lined up from X:00 to X:59 can ride the ride once from (X+1):00 to (X+1):59
    - X:00 - 1st gate closes, people can start waiting in front for ride
    - X:30 - 1st gate opens allowing people into middle region
    - (X+1):00 - anyone who showed up between X:00-X:59 gets through 2nd gate

2 Door system concluded

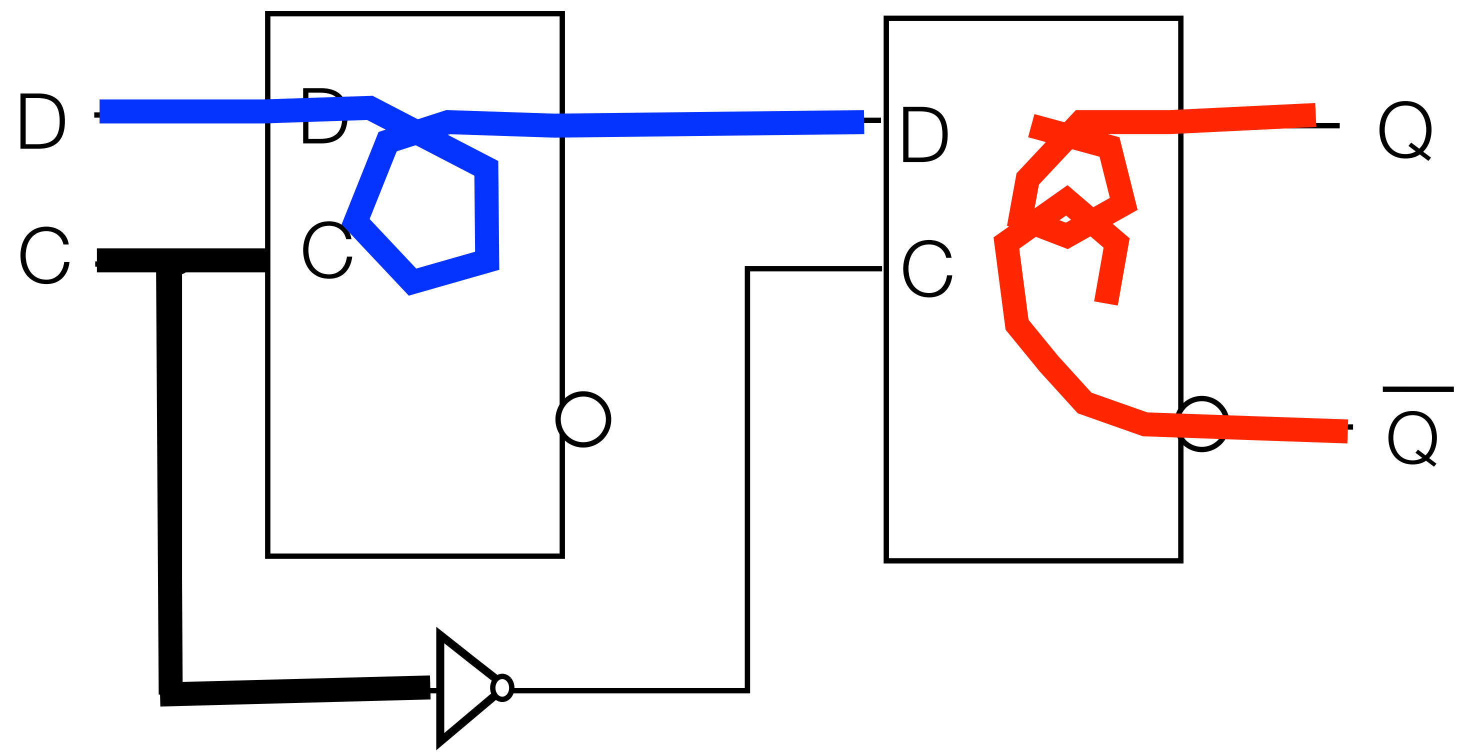

8:00
8:30
9:00
9:30
10:00
9:00 group lines up outer door
9:00 group lines up inner door
9:00 group gets access
10:00 group lines up outer door
10:00 group lines up inner door
10:00 group gets access

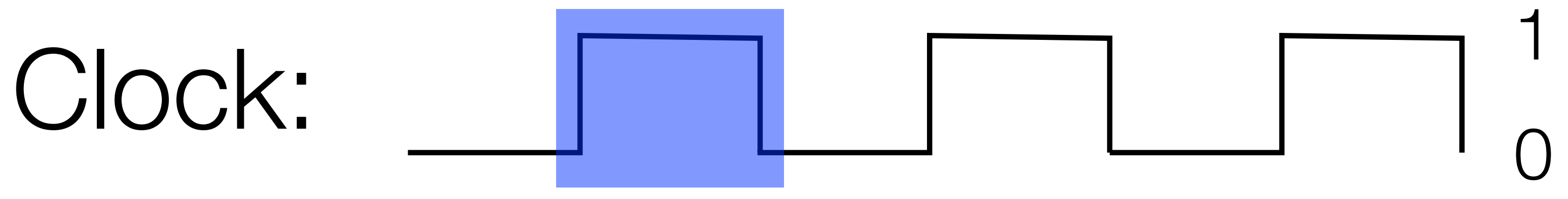
D Flip-Flop Summary



- C (Control) is fed a clock pulse (alternates between 0 and 1 with fixed period)
  - C=1: Master latch "on", Slave latch "off"
    - New D input read into master
    - Previous Q values still emitted (not affected by new D inputs)
  - C=0: Master latch "off", Slave latch "on"
    - Changing D inputs has no effect on Master (or Slave) latch
    - D inputs from last time C=1 stored safely in Master and transferred into Slave and reflected on output Q

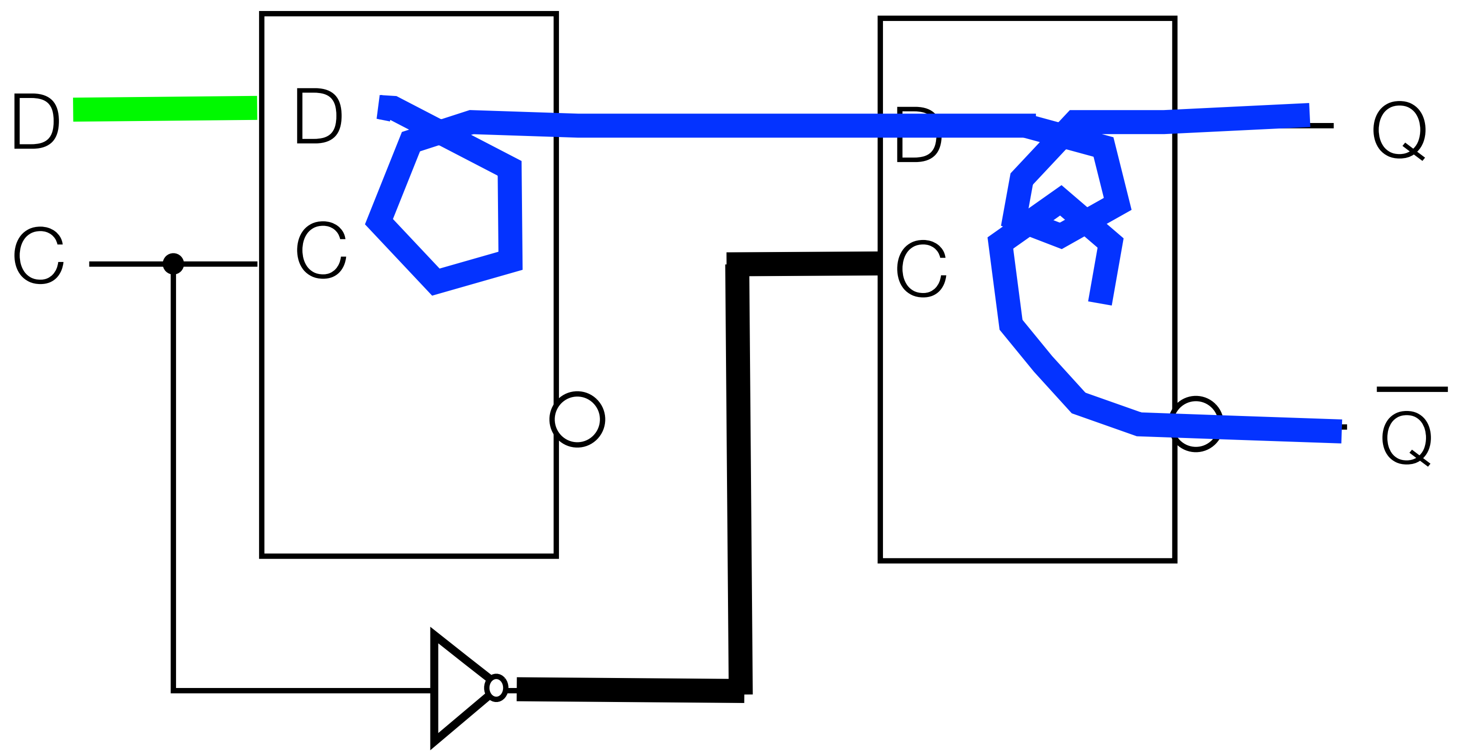
D Flip-Flop Analysis

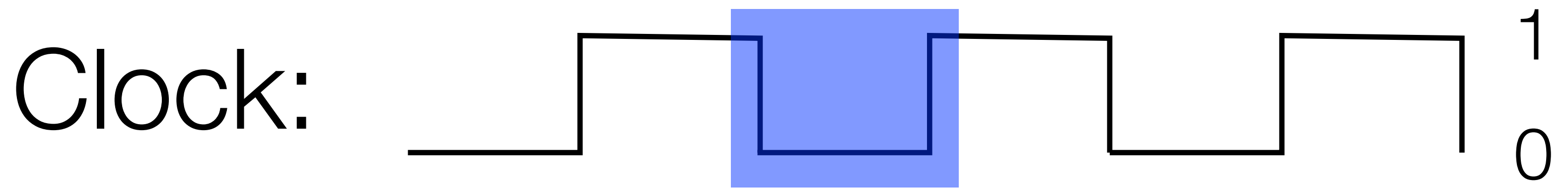




- C=1: Master latch “on”, Slave latch “off”
  - New S & R inputs read into master
  - Previous Q values still emitted (not affected by new S&R inputs)

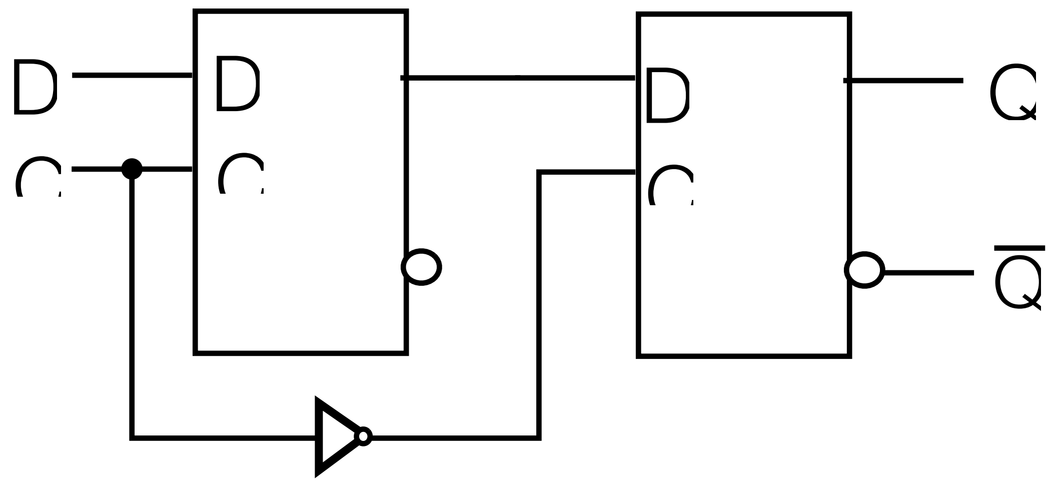
D Flip-Flop Analysis

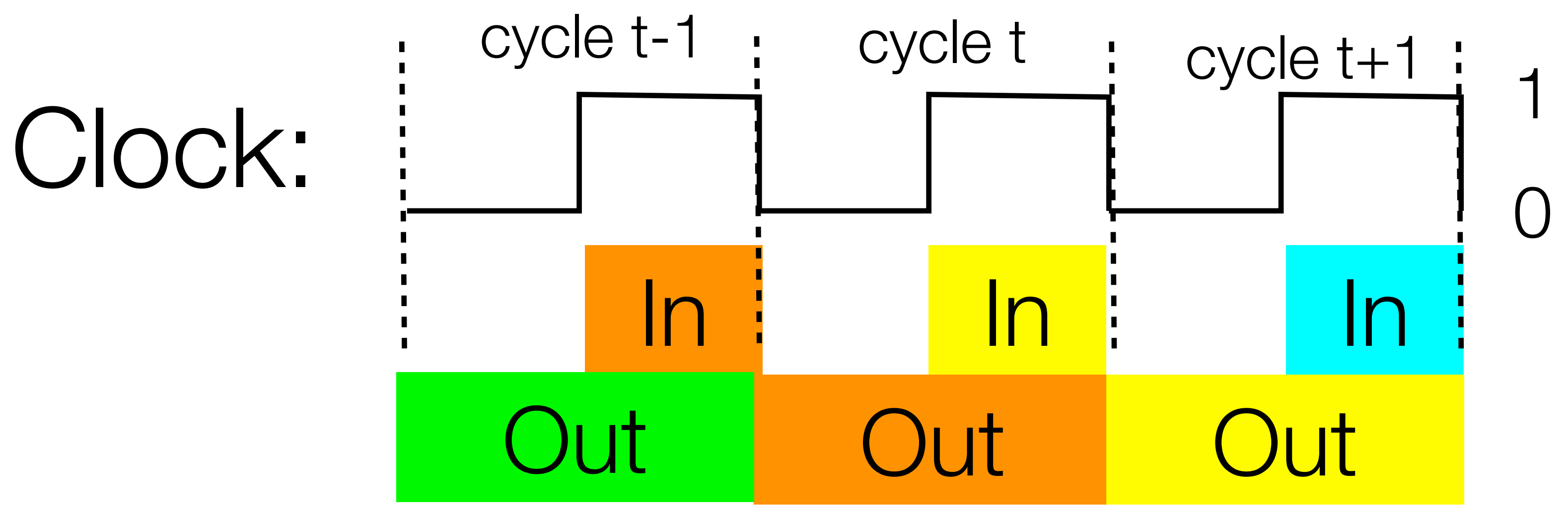




- C=0: Master latch “off”, Slave latch “on”
  - Changing D inputs has no effect on Master (or Slave) latch
  - D inputs from last time C=1 stored safely in Master and transferred into Slave and reflected on output Q

Flip-Flop Activation over Time



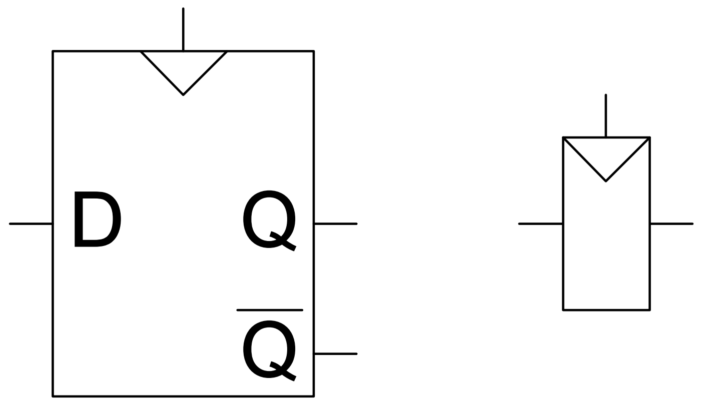


- Q(t): value output by Flip-Flop during the tth clock cycle (clock =0, then 1 during a full cycle)
- Depends on input during end of t-1st cycle

D Flip-Flop Summary

- Two inputs: CLK, D
- Function
  - The flip-flop “samples” D on the rising edge of CLK
    - When CLK rises from 0 to 1, D passes through to Q
    - Otherwise, Q holds its previous value
  - Q changes only on the rising edge of CLK
- A flip-flop is called an edge-triggered device because it is activated on the clock edge

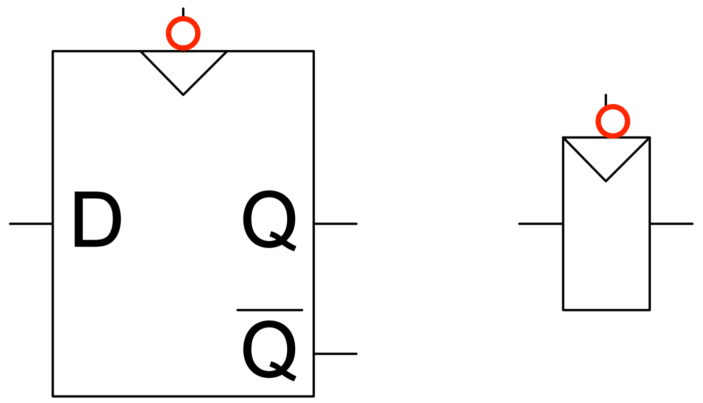
D Flip-Flop Symbols



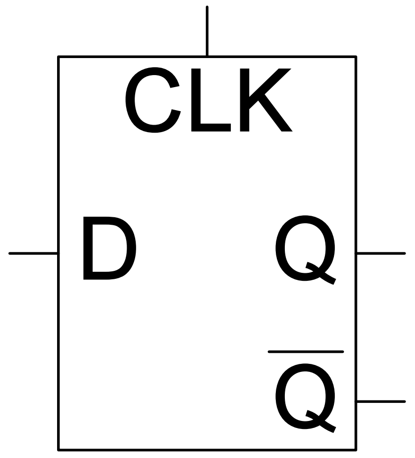
D Flip-Flop Summary

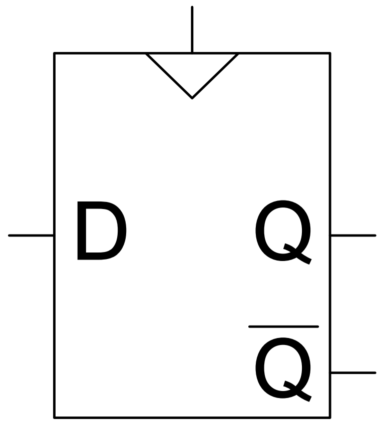
- Two inputs: CLK, D
- Function
  - The flip-flop "samples" D on the falling edge ~~rising edge~~ of CLK
    - When CLK falls from 1 to 0 ~~rises from 0 to 1~~, D passes through to Q
    - Otherwise, Q holds its previous value
  - Q changes only on the falling edge ~~rising edge~~ of CLK
- A flip-flop is called an edge-triggered device because it is activated on the clock edge

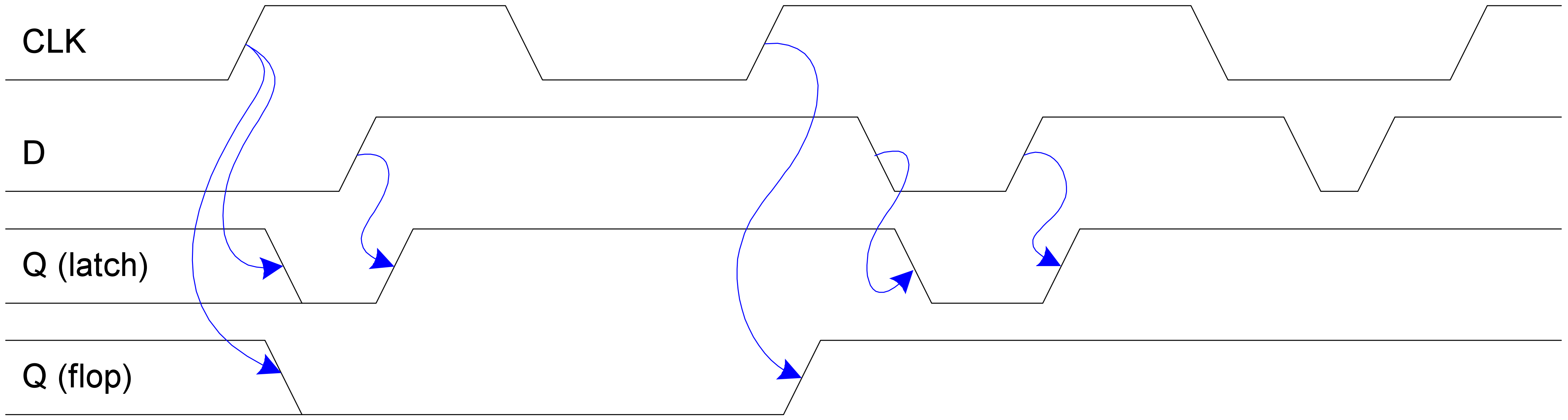
D Flip-Flop Symbols



D Flip-Flop vs. D Latch

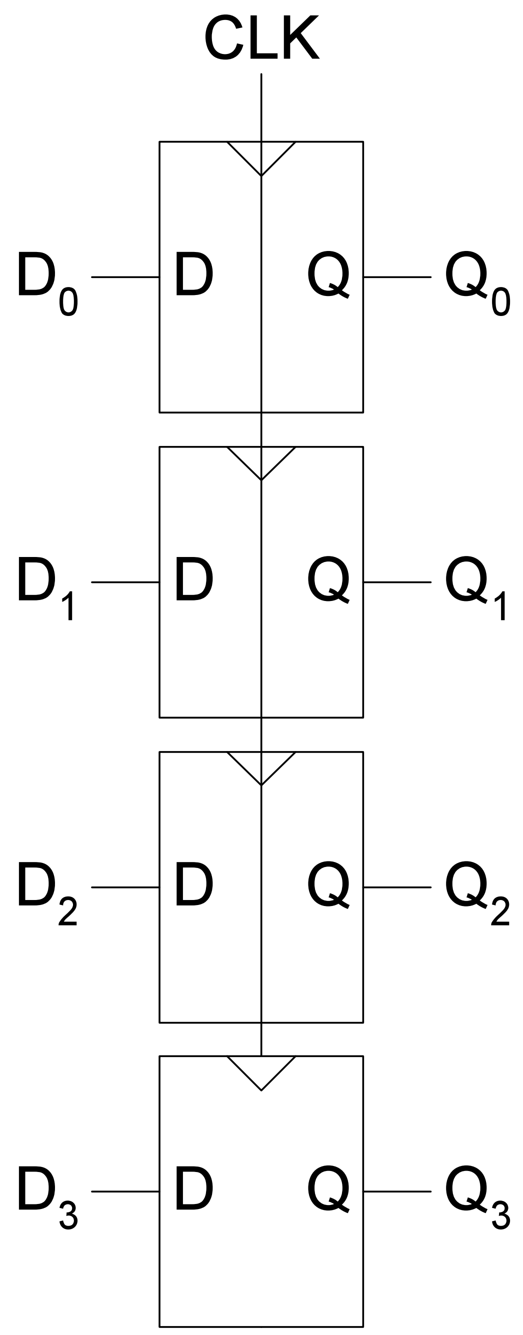


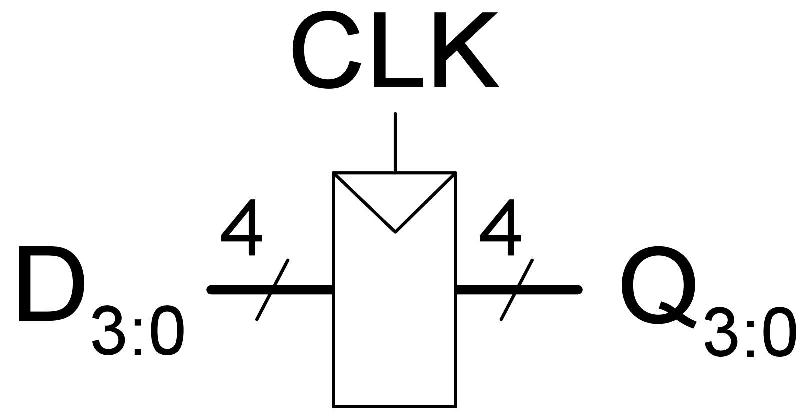




Latch outputs change at any time, flip-flops only during clock transitions

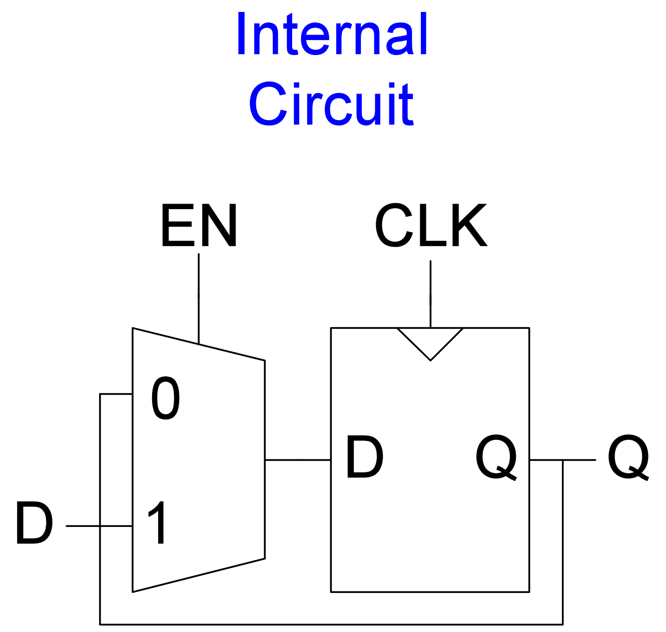
Registers

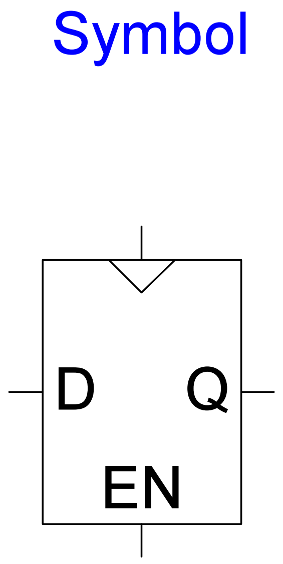




Enabled Flip-Flops

- Inputs: CLK, D, EN
    - The enable input (EN) controls when new data (D) is stored
- Function
    - EN = 1: D passes through to Q on the clock edge
    - EN = 0: the flip-flop retains its previous state

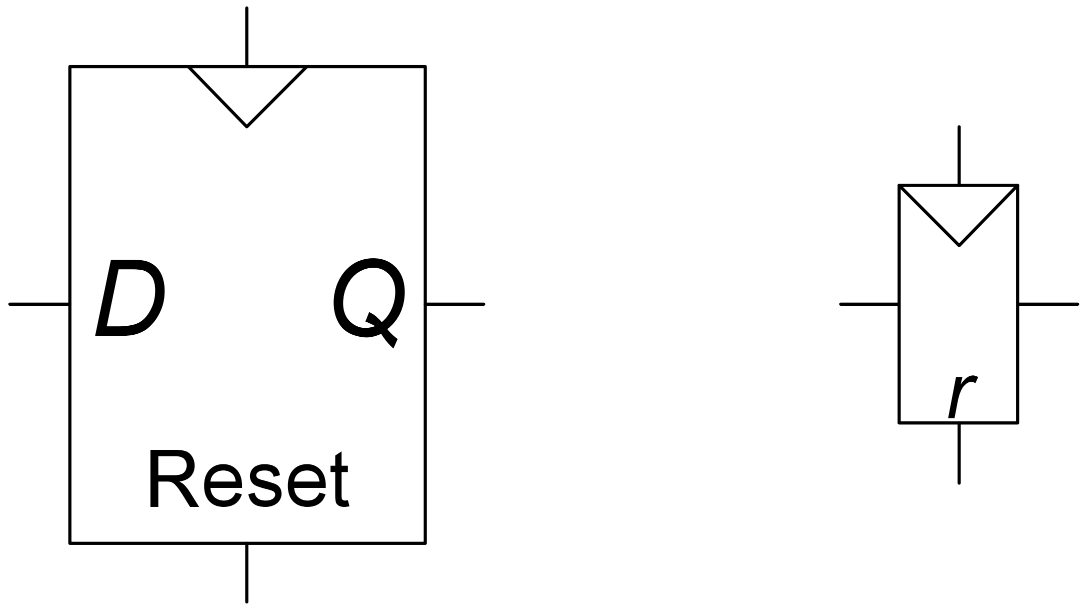




Resettable Flip Flops

- Inputs: CLK, D, Reset
- Function:
  - Reset = 1: Q is forced to 0
  - Reset = 0: the flip-flop behaves like an ordinary D flip-flop

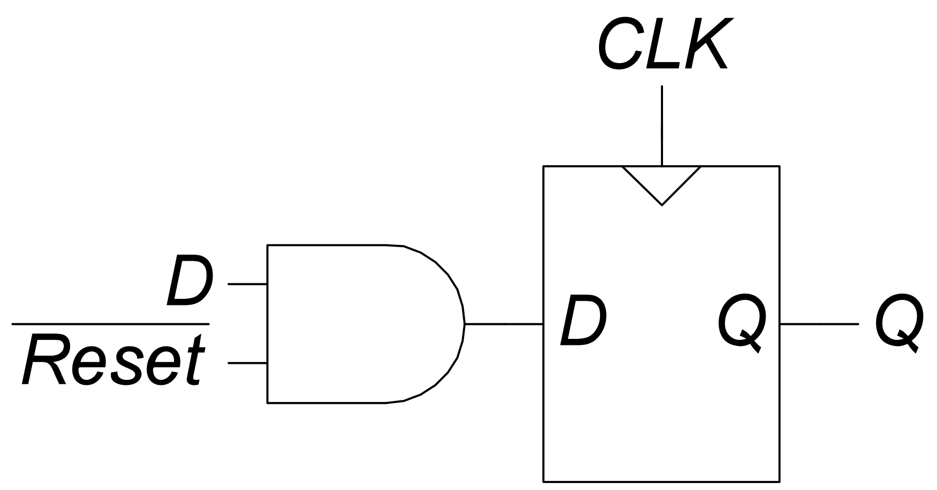
Symbols



Resettable Flip-Flops (2)

- Two types:
  - Synchronous: resets at the clock edge only
  - Asynchronous: resets immediately when Reset = 1
- Asynchronously resettable flip-flop requires changing the internal circuitry of the flip-flop (see Exercise 3.10)
- Synchronously resettable flip-flop?

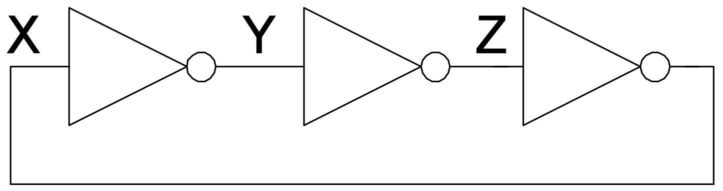
Internal Circuit

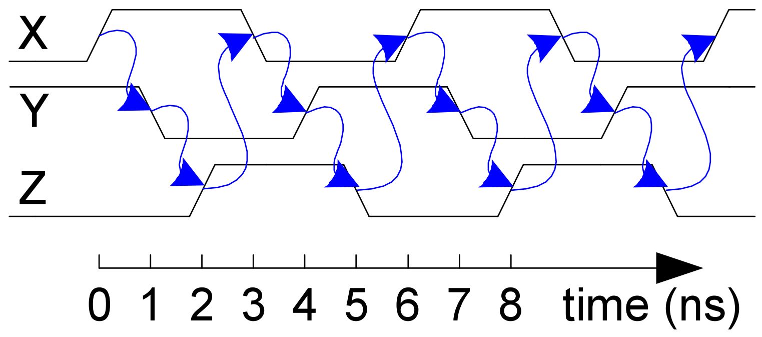


Sequential Logic

- Sequential circuits: all circuits that aren't combinational

- A problematic circuit:





- This circuit has no inputs and 1-3 outputs

- It is an astable circuit that oscillates

- Its period depends on the delay of the inverters – which depends on the manufacturing process, temperature, etc

- The circuit has a **cyclic path**: output fed back to input

Synchronous Sequential Logic Design

- Breaks cyclic paths by inserting registers
- These registers contain the state of the system
- The state changes at the clock edge, so we say the system is **synchronized** to the clock

- Rules of synchronous sequential circuit composition:
  - Every circuit element is either a register or a combinational circuit
  - At least one circuit element is a register
  - All registers receive the same clock signal
  - Every cyclic path contains at least one register
- Two common synchronous sequential circuits
  - Finite State Machines (FSMs)
  - Pipelines

Timing Issues

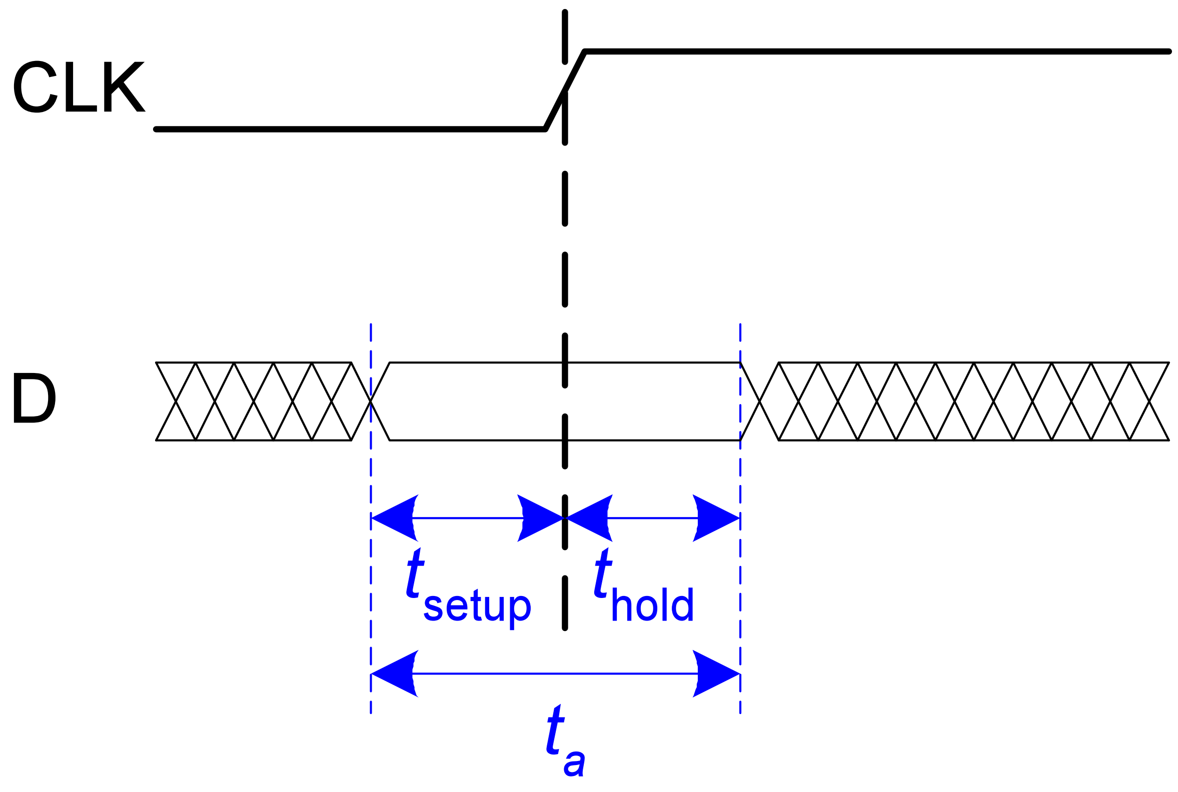
- Flip-flop samples D at clock edge
- D must be stable when it is sampled
- Similar to a photograph, D must be stable around the clock edge
- If D is changing when it is sampled, metastability can occur

Input Timing Constraints

- Setup time: t_{setup} = time before the clock edge that data must be stable (i.e. not changing)

- Hold time: t_{hold} = time after the clock edge that data must be stable

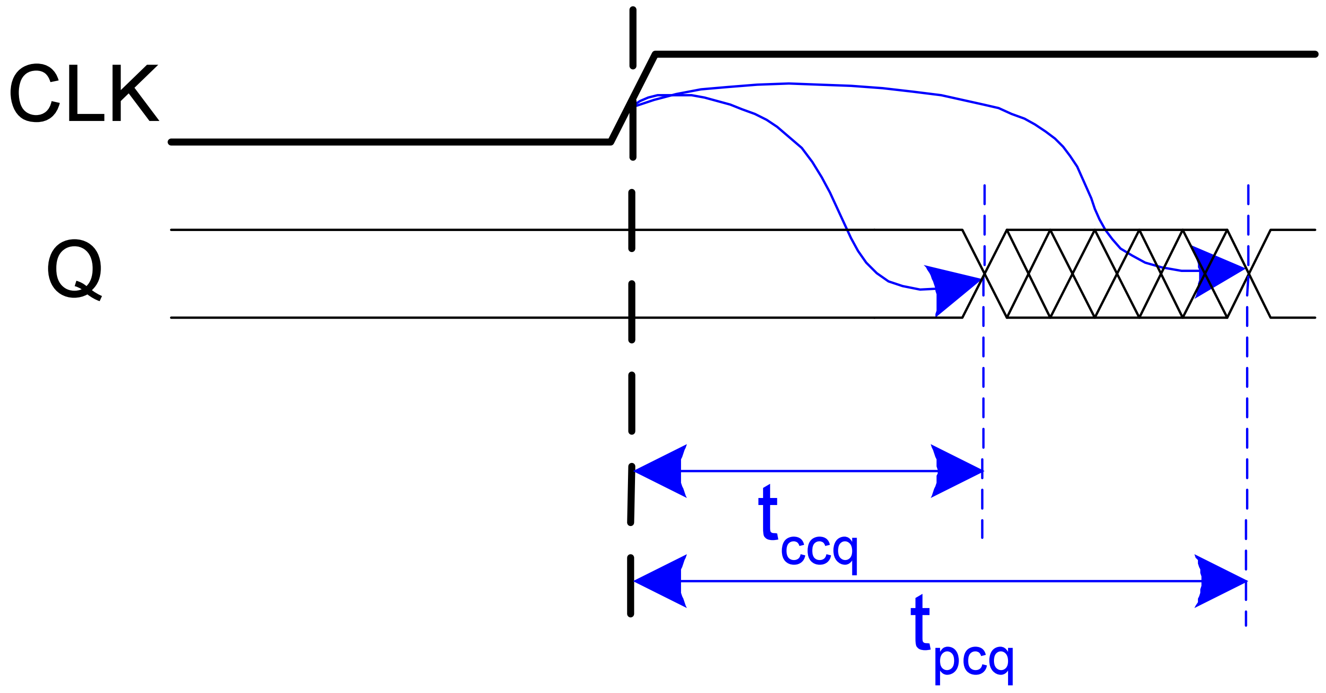
- Aperture time: t_a = time around clock edge that data must be stable ($t_a = t_{setup} + t_{hold}$)



Output Timing Constraints

- Propagation delay: t_{pcq} = time after clock edge that the output Q is guaranteed to be stable (i.e., to stop changing)

- Contamination delay: t_{ccq} = time after clock edge that Q might be unstable (i.e., start changing)

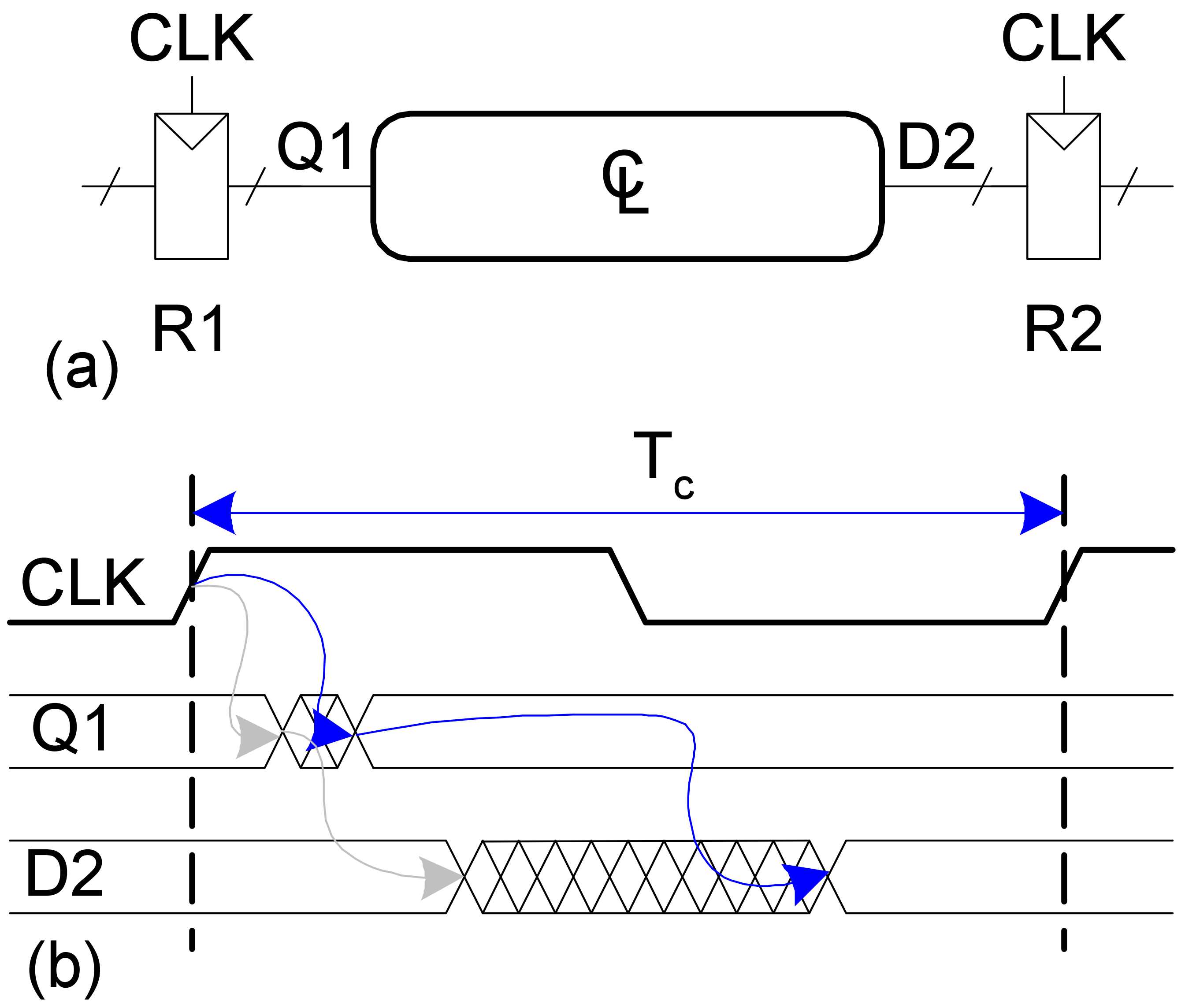


Dynamic Discipline

- The input to a synchronous sequential circuit must be stable during the aperture (setup and hold) time around the clock edge.
- Specifically, the input must be stable
  - at least t_{setup} before the clock edge
  - at least until t_{hold} after the clock edge

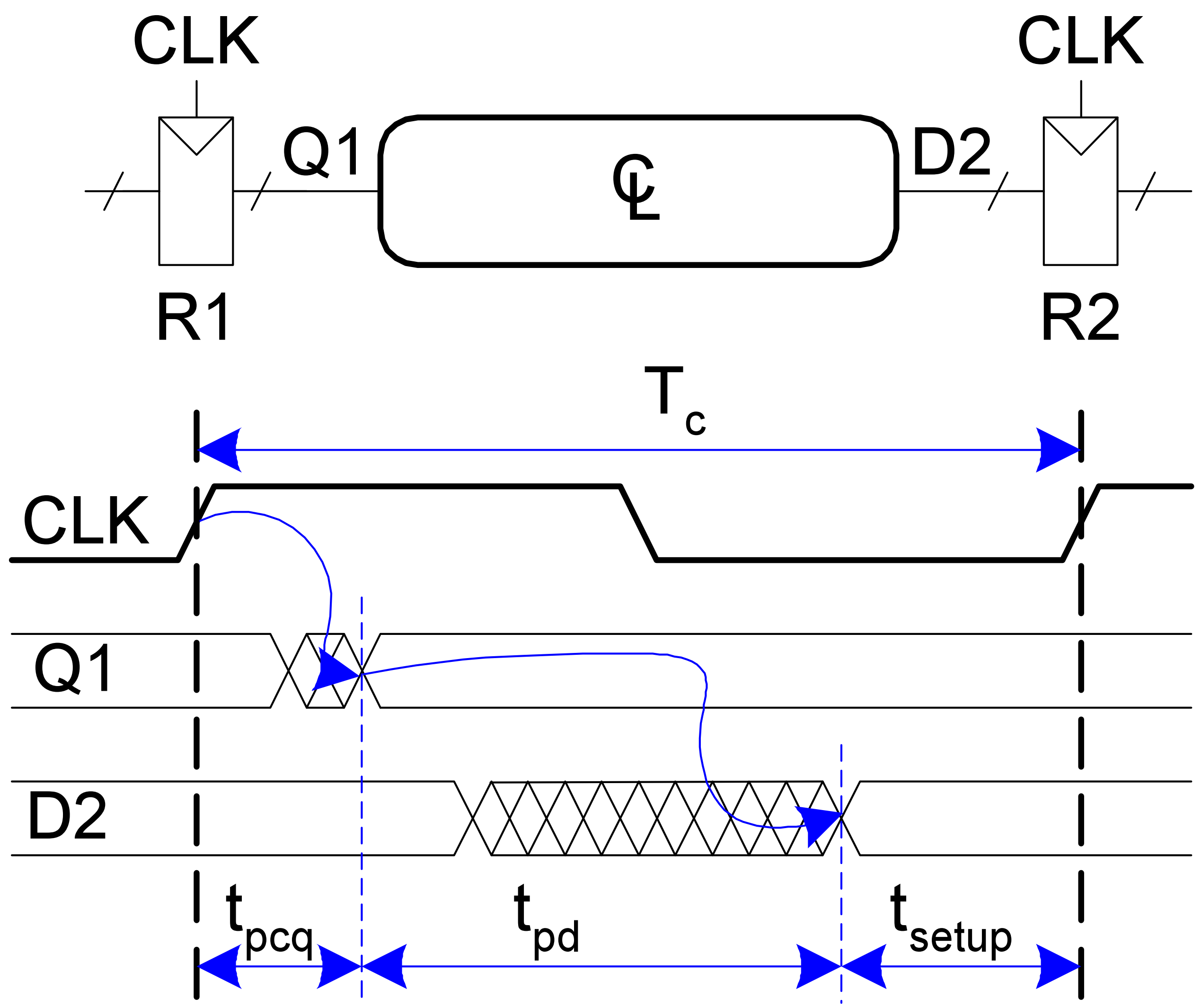
Dynamic Discipline

- The delay between registers has a **minimum** and **maximum** delay, dependent on the delays of the circuit elements



Setup Time Constraint

- The setup time constraint depends on the **maximum** delay from register R1 through the combinational logic.
- The input to register R2 must be stable at least t_{setup} before the clock edge.

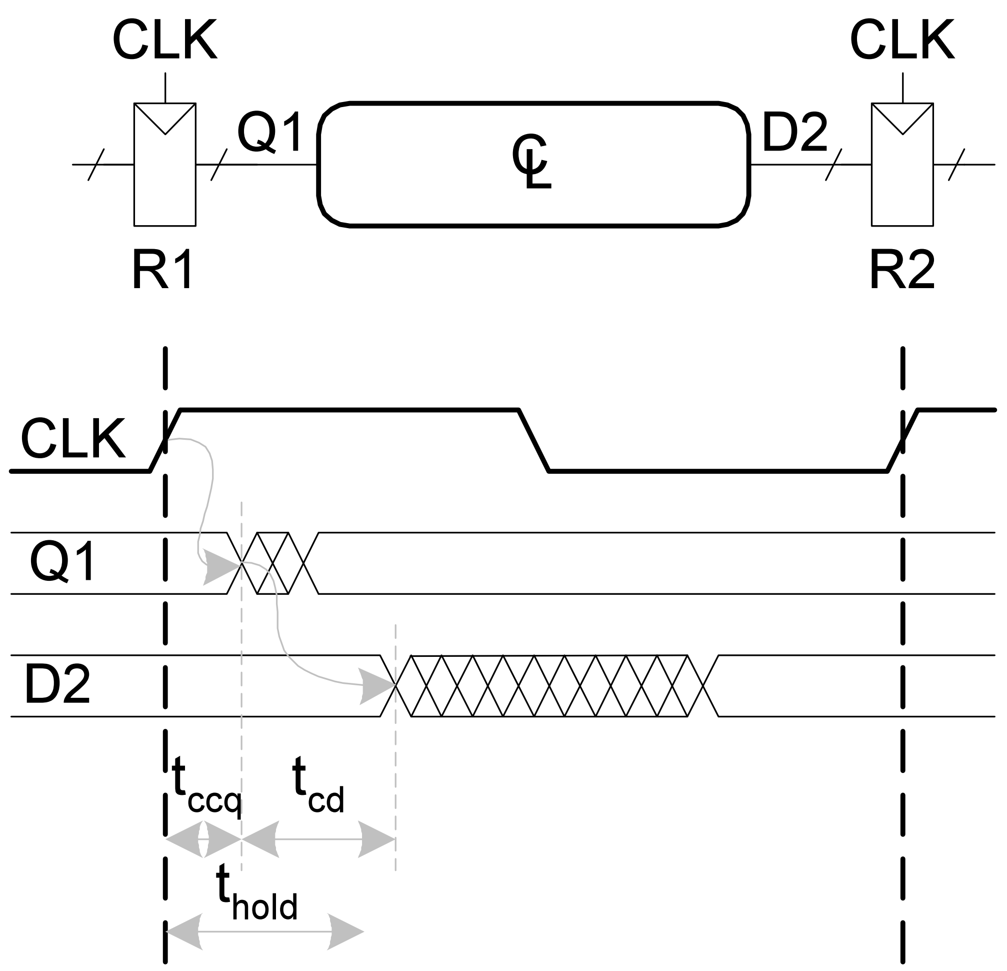


$$T_c \geq t_{pcq} + t_{pd} + t_{setup}$$
$$t_{pd} \leq T_c - (t_{pcq} + t_{setup})$$

Hold Time Constraint

- The hold time constraint depends on the **minimum** delay from register R1 through the combinational logic.

- The input to register R2 must be stable for at least t_{hold} after the clock edge.



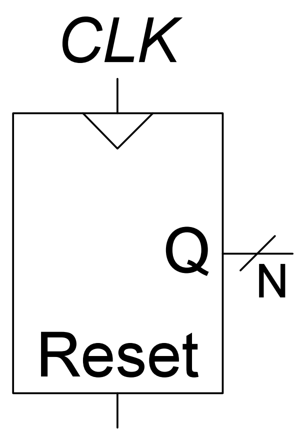
$$t_{hold} < t_{ccq} + t_{cd}$$

$$t_{cd} > t_{hold} - t_{ccq}$$

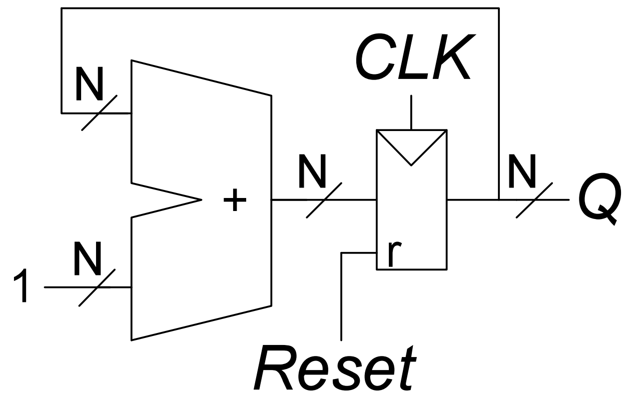
Counters

- Increments on each clock edge.
- Used to cycle through numbers. For example,
  - 000, 001, 010, 011, 100, 101, 110, 111, 000, 001…
- Example uses:
  - Digital clock displays
  - Program counter: keeps track of current instruction executing

Symbol



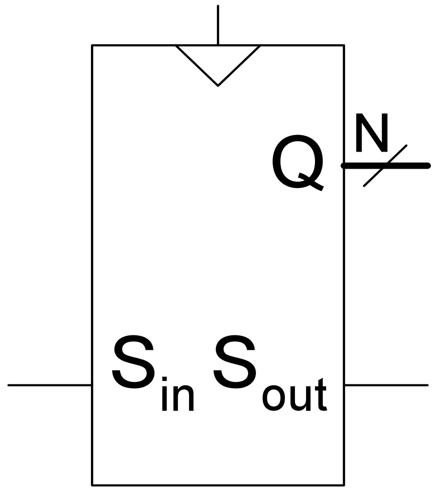
Implementation



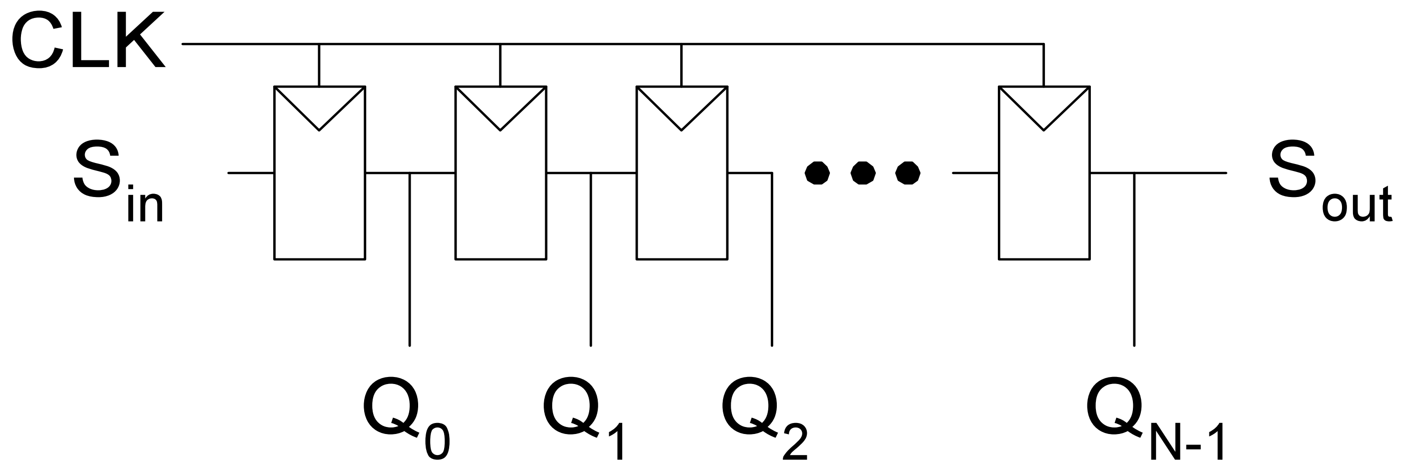
Shift Register

- Shift a new value in on each clock edge
- Shift a value out on each clock edge
- Serial-to-parallel converter: converts serial input (S_{in}) to parallel output ($Q_{0:N-1}$)

Symbol:

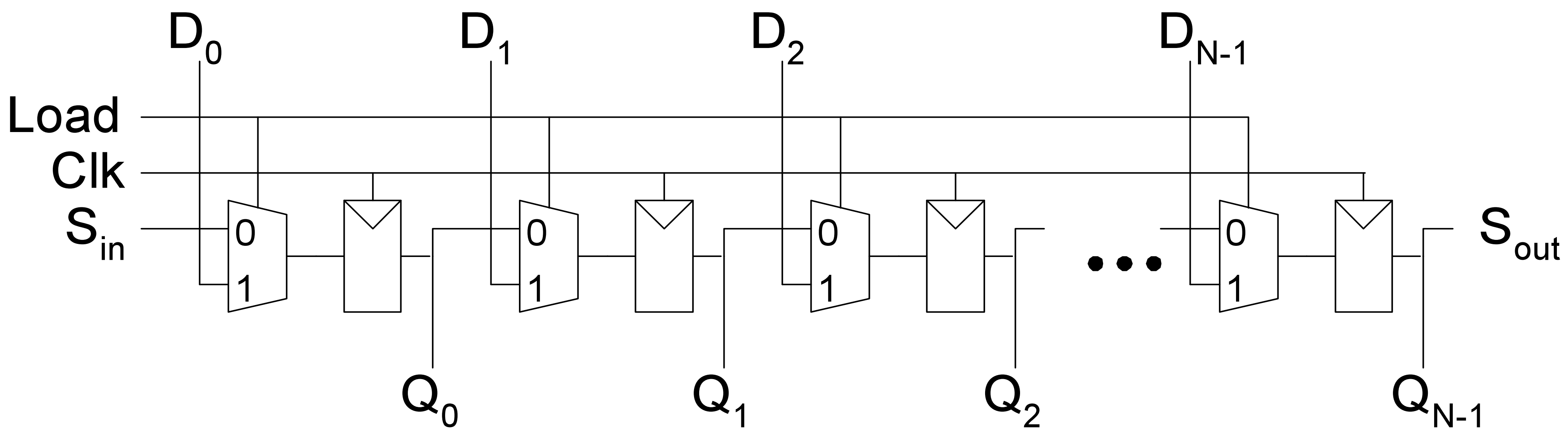


Implementation:



Shift Register with Parallel Load

- When *Load = 1*, acts as a normal N-bit register
- When *Load = 0*, acts as a shift register
- Now can act as a *serial-to-parallel converter* (S_{in} to $Q_{0:N-1}$) or a parallel-to-serial converter ($D_{0:N-1}$ to S_{out})



D0
D1
D2
DN-1
Load
Clk
Sin
0
1
Sout
Q0
Q1
Q2
QN-1