

# Digital Logic Design

4190.201

2014 Spring Semester

## 7. Sequential logic

Naehyuck Chang  
Dept. of CSE  
Seoul National University  
[naehyuck@snu.ac.kr](mailto:naehyuck@snu.ac.kr)

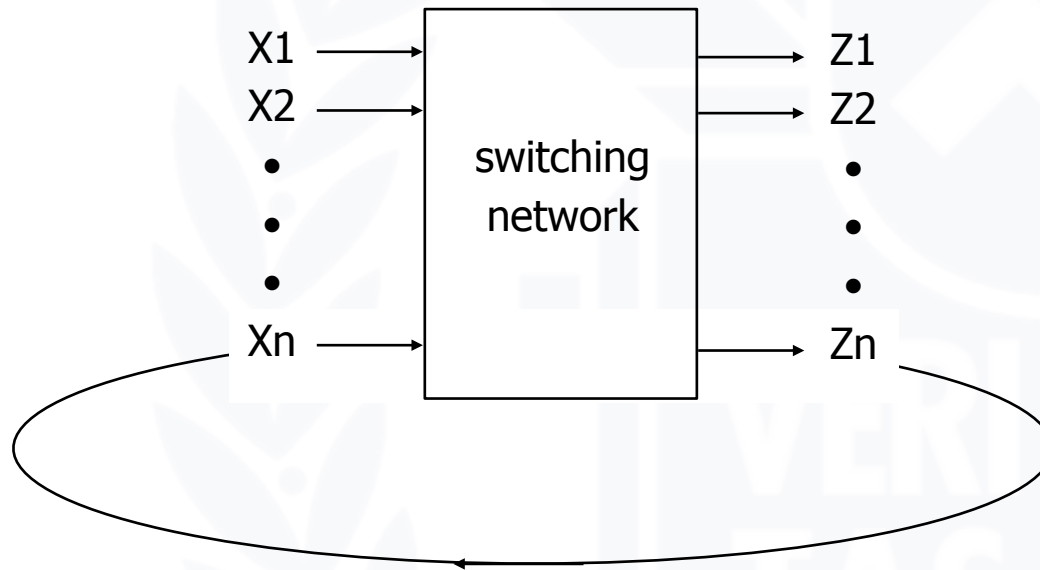


# Sequential logic

- Sequential circuits
  - Simple circuits with feedback
  - Latches
  - Edge-triggered flip-flops
- Timing methodologies
  - Cascading flip-flops for proper operation
  - Clock skew
- Asynchronous inputs
  - Metastability and synchronization
- Basic registers
  - Shift registers
  - Simple counters
- Hardware description languages and sequential logic

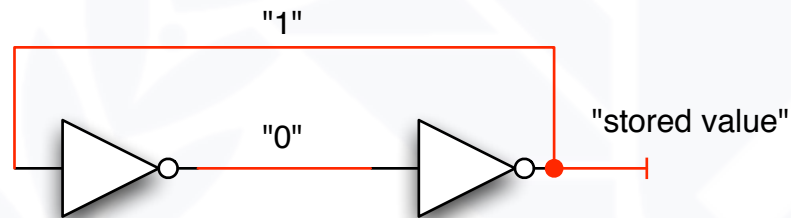
# Circuits with feedback

- How to control feedback?
  - What stops values from cycling around endlessly

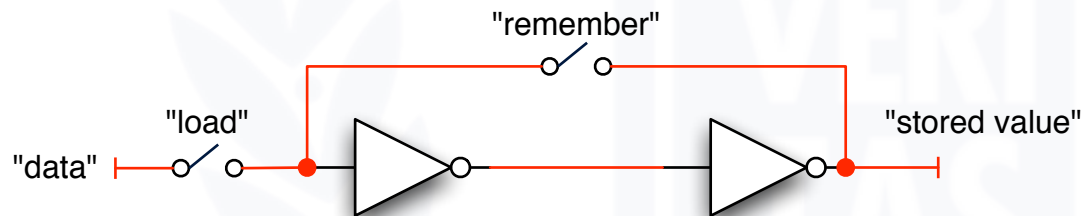


# Simplest circuits with feedback

- Two inverters form a static memory cell
  - Will hold value as long as it has power applied

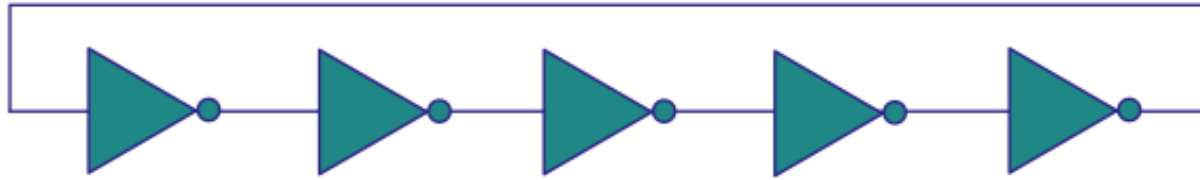


- How to get a new value into the memory cell?
  - Selectively break feedback path
  - Load new value into cell

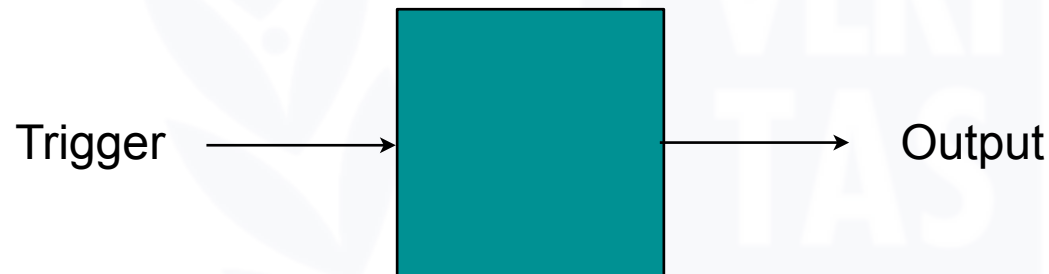


# Ring oscillator

- Odd number of inverter chain



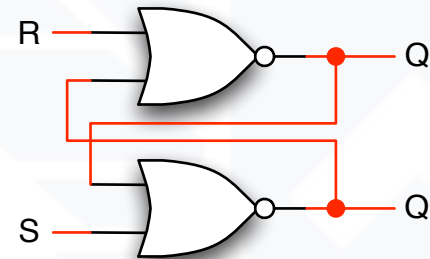
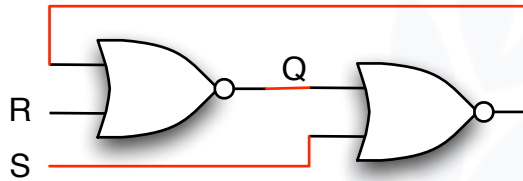
- Even numbered?
- Multivibrators
  - Astable
  - Bistable
  - Monostable



# Memory with cross-coupled gates

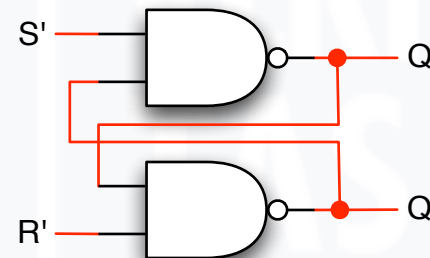
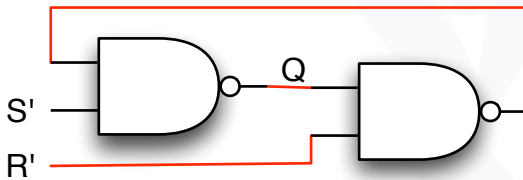
- Cross-coupled NOR gates

- Similar to inverter pair, with capability to force output to 0 (reset=1) or 1 (set=1)

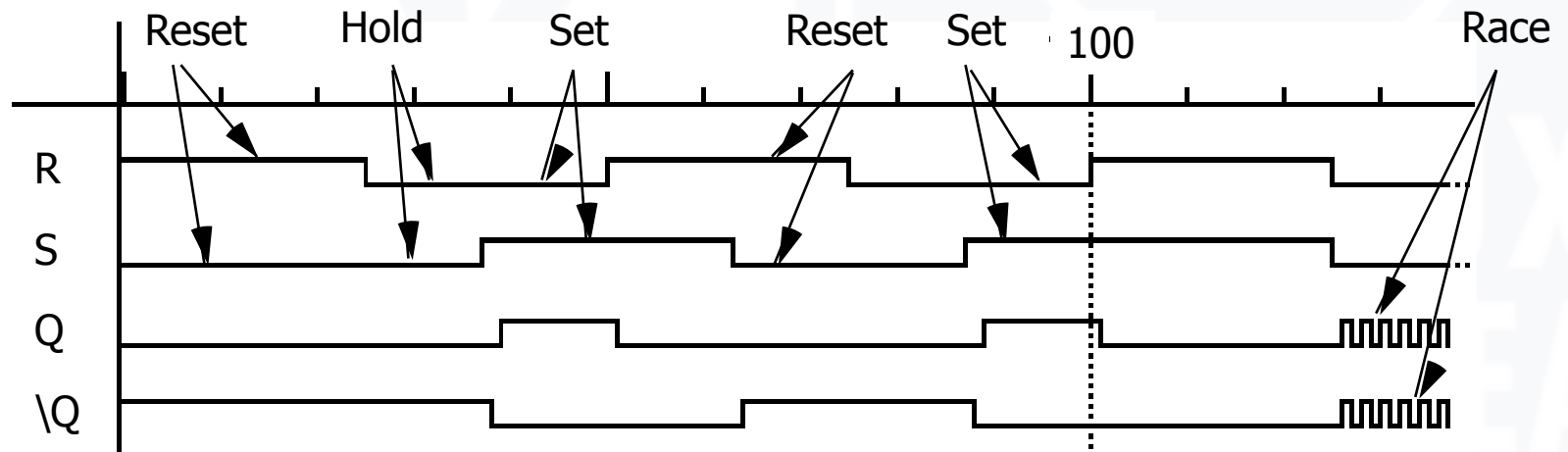
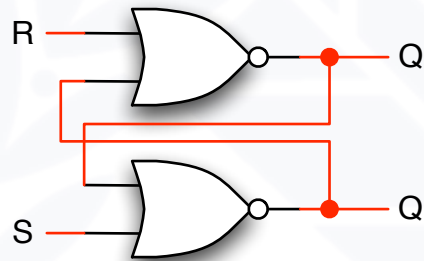


- Cross-coupled NAND gates

- Similar to inverter pair, with capability to force output to 0 (reset=0) or 1 (set=0)



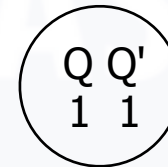
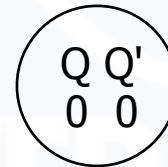
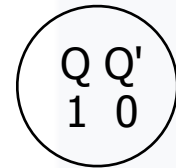
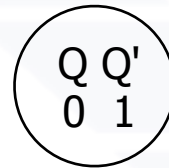
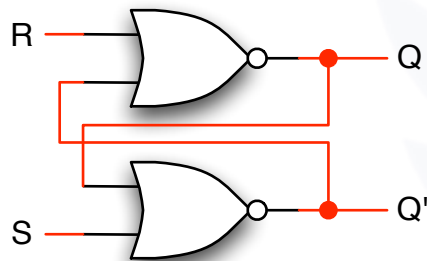
# Timing behavior



# State behavior or R-S latch

- Truth table of R-S latch behavior

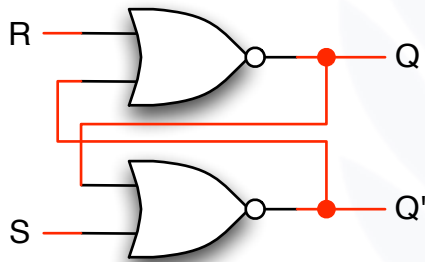
S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	unstable



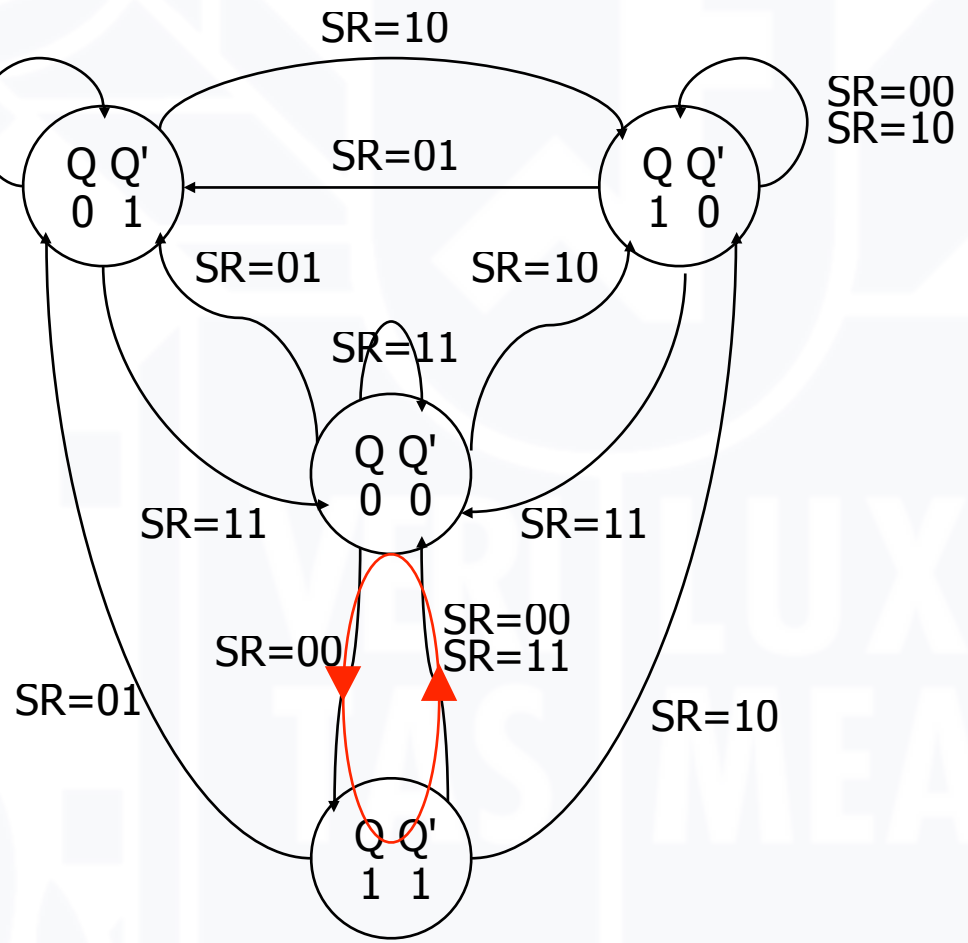


# Theoretical R-S latch behavior

- State diagram
- States: possible values
- Transitions: changes based on inputs

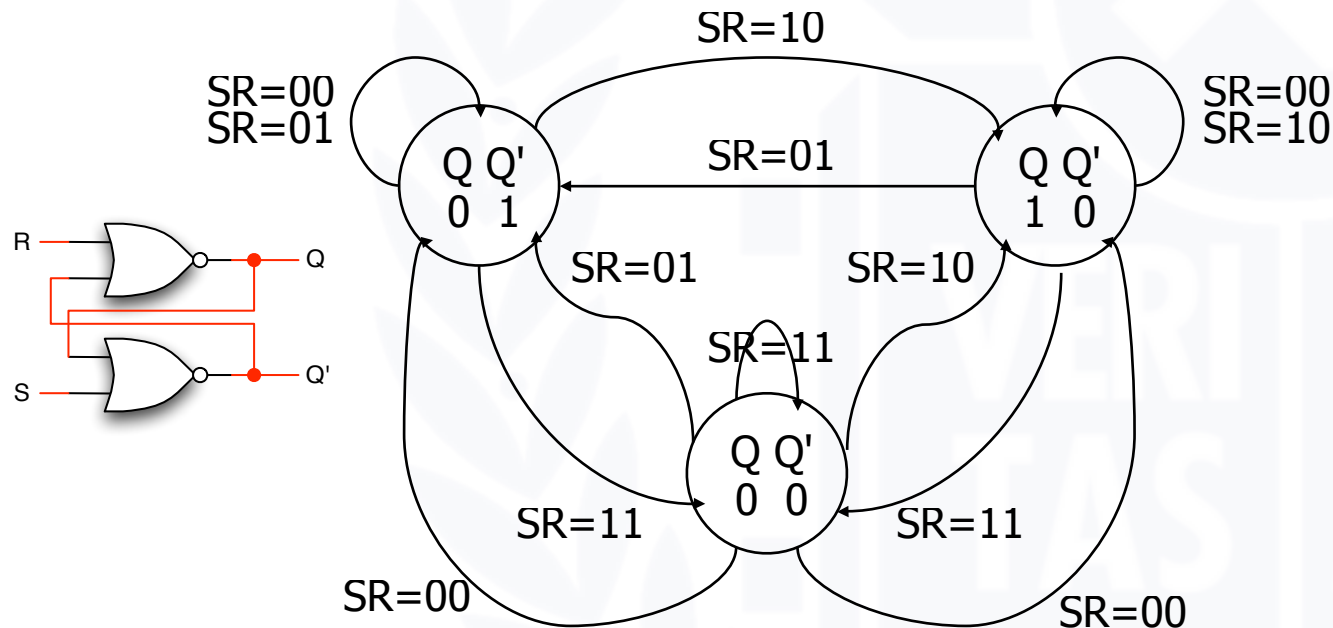


Possible oscillation  
between states 00 and 11



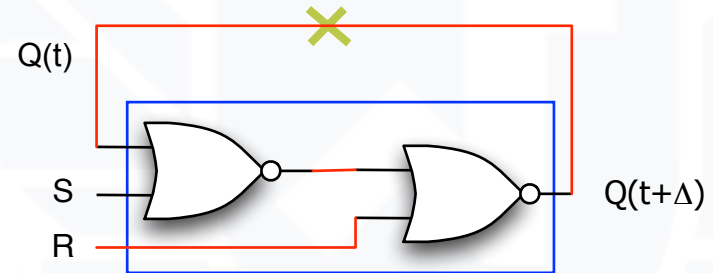
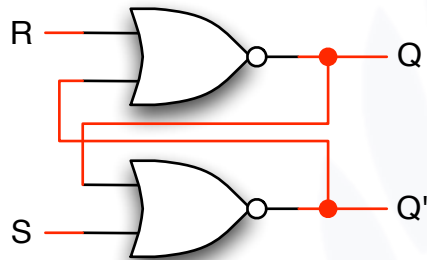
# Observed R-S latch behavior

- Very difficult to observe R-S latch in the 1-1 state
  - One of R or S usually changes first
- Ambiguously returns to state 0-1 or 1-0
  - A so-called "race condition"
  - Or non-deterministic transition



# R-S latch analysis

- Break feedback path



S	R	Q(t)	Q(t+Δ)	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	not allowed
1	1	1	X	

		S	
		0	1
Q(t)	0	X	1
	1	X	1
		R	

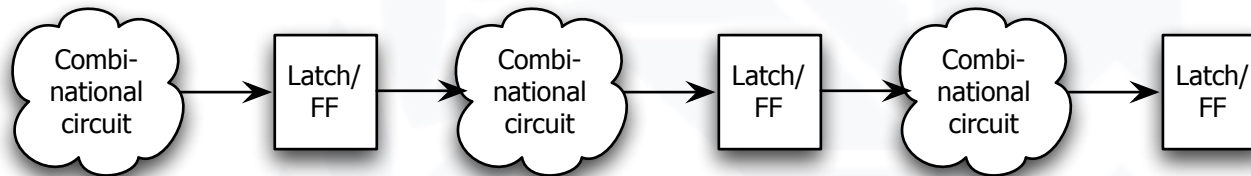
Characteristic equation  
 $Q(t+\Delta) = S + R' Q(t)$

# Activity: R-S latch using NAND gates

# R-S latch

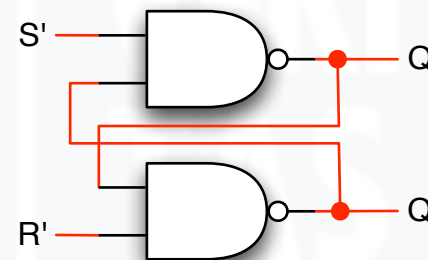
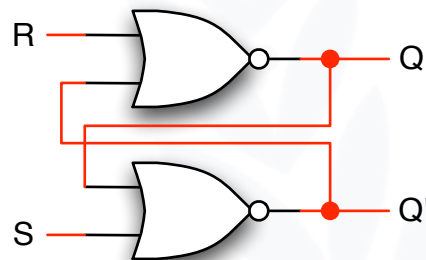
- Summary

- Usage of latches and FFs



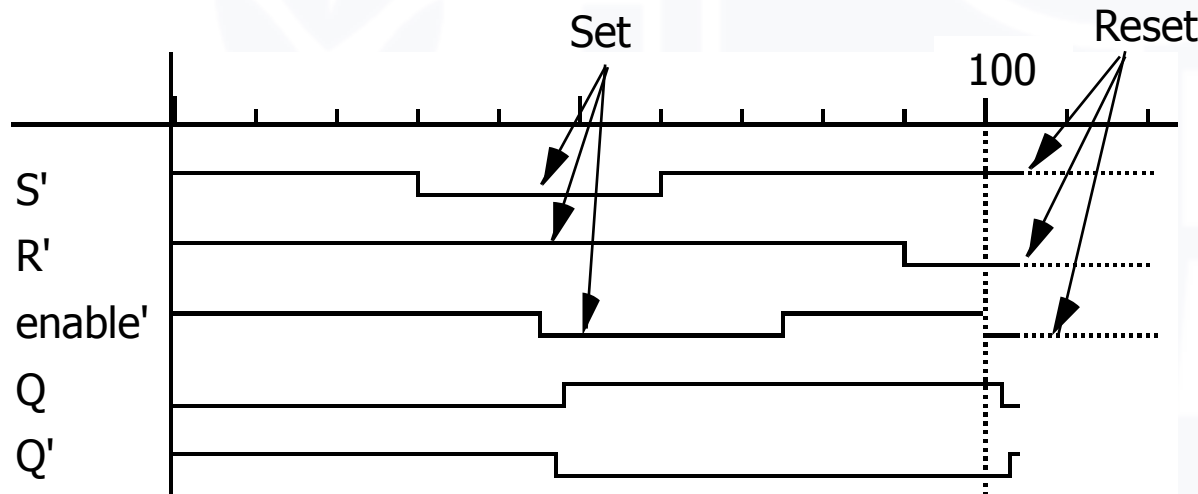
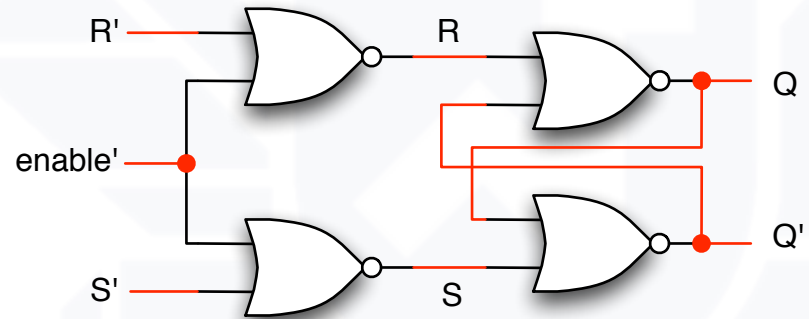
- Shortcoming of RS latch

- Should be hazard free
- Design consideration of forbidden input states



# Gated R-S latch

- Control when R and S inputs matter
  - Otherwise, the slightest glitch on R or S while enable is low could cause change in value stored



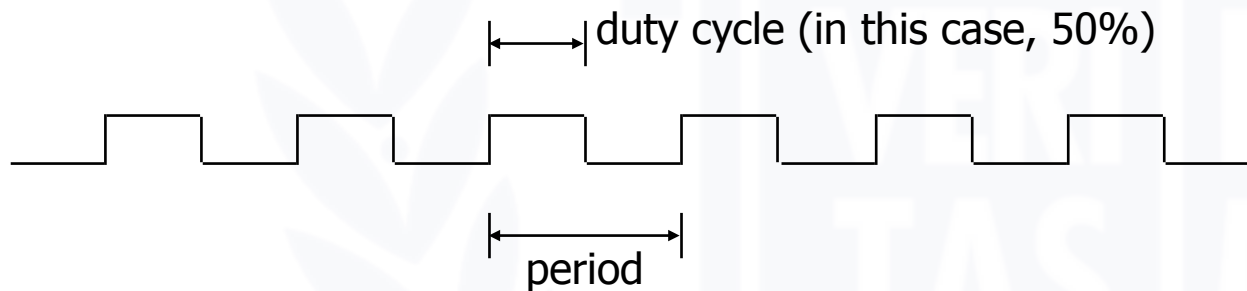
# Overview of flip flops and latches

- Flip flop
  - Edge sensitive
  - Positive (rising) or negative (falling) edges
- Latch
  - Level sensitive
  - Positive or negative
  - Transparent when enable is active
- Function
  - Preserve previous output state
- Types
  - RS: set or reset regardless of current output
  - Toggle: toggle current output (only for FF)
  - D: sample current input
  - JK: multifunction



# Clocks

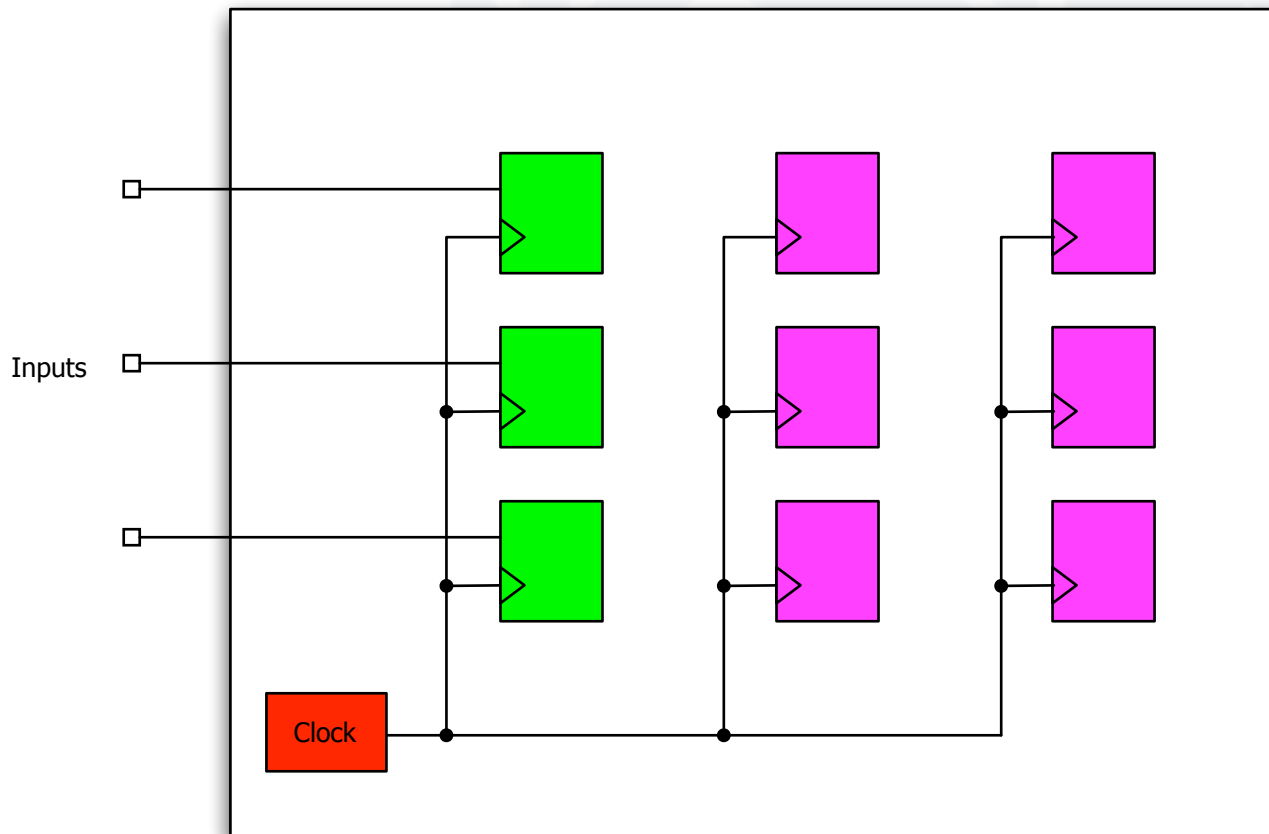
- Used to keep time
  - Wait long enough for inputs (R' and S') to settle
  - Then allow to have effect on value stored
- Clocks are regular periodic signals
  - Period (time between ticks)
  - Duty-cycle (time clock is high between ticks - expressed as % of period)





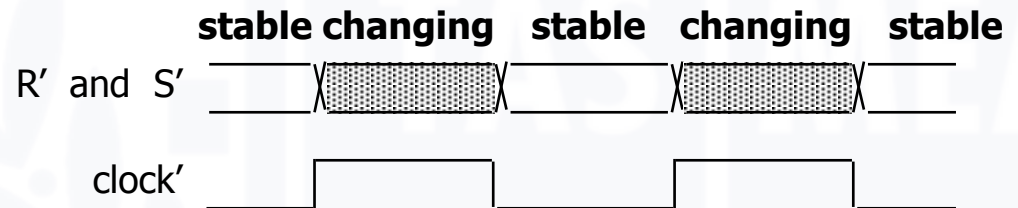
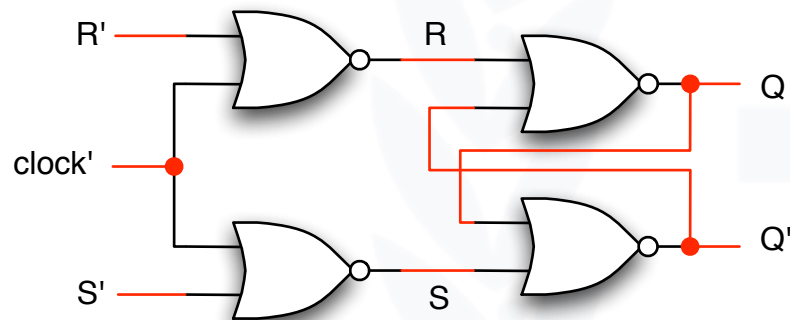
# Clocks

- Clocks for synchronous sequential circuits
  - All the FFs in a synchronous state machine should be connected to a single clock
  - All the external inputs should be synchronized to the clock



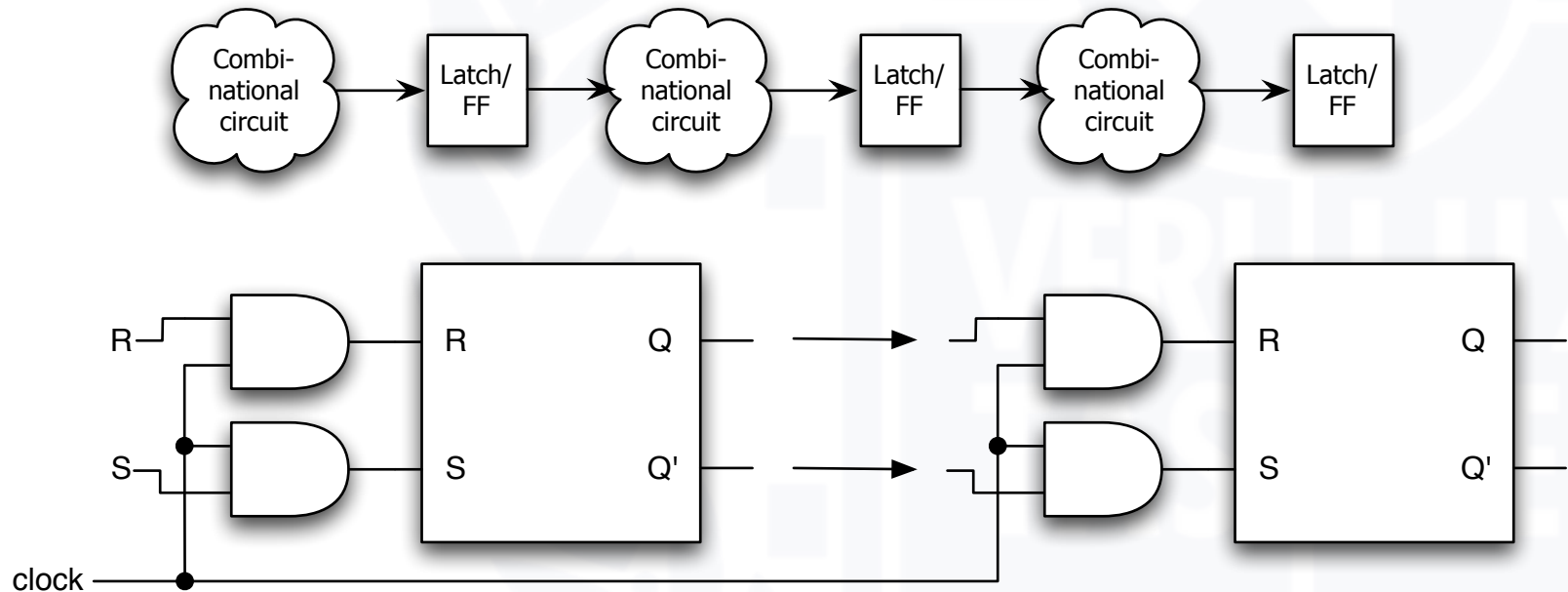
# Clocks (cont'd)

- Controlling an R-S latch with a clock (Clocked RS-FF)
  - Can't let R and S change while clock is active (allowing R and S to pass)
  - Only have half of clock period for signal changes to propagate
  - Signals must be stable for the other half of clock period
- Can be free from hazard
  - If we design all the combinational logic propagation during the enable is inactive



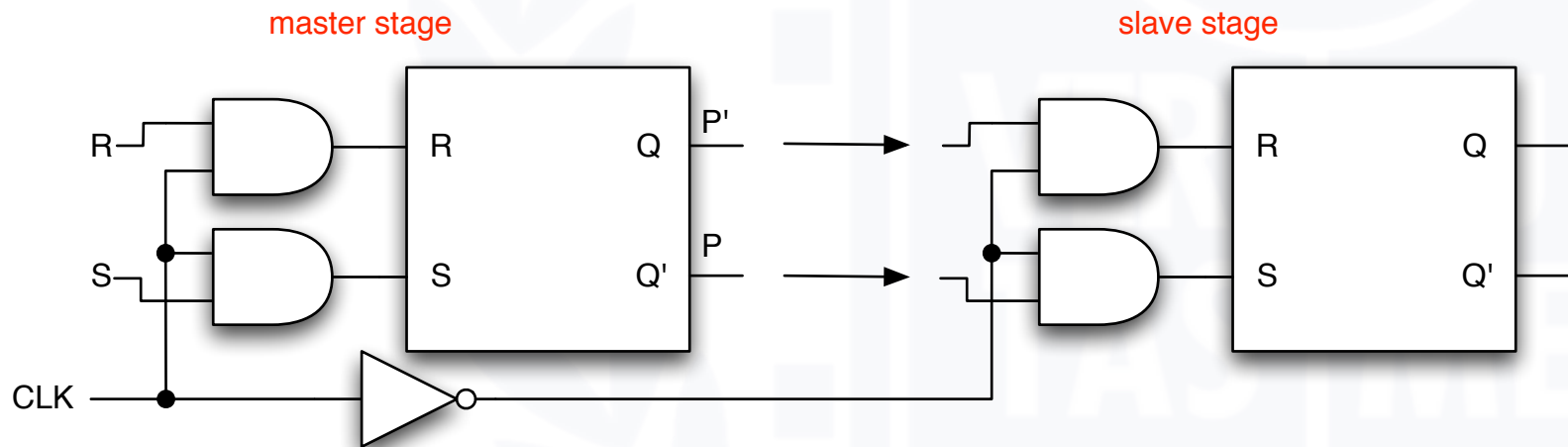
# Cascading latches

- Connect output of one latch to input of another
- How to stop changes from racing through chain?
  - Need to be able to control flow of data from one latch to the next
  - Move one latch per clock period
  - Have to worry about logic between latches (arrows) that is too fast



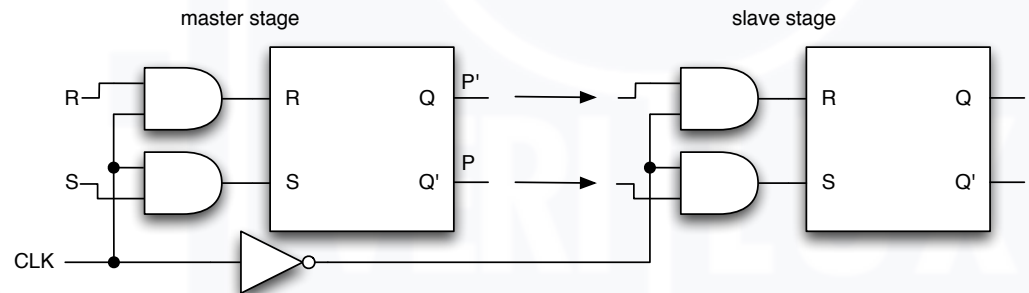
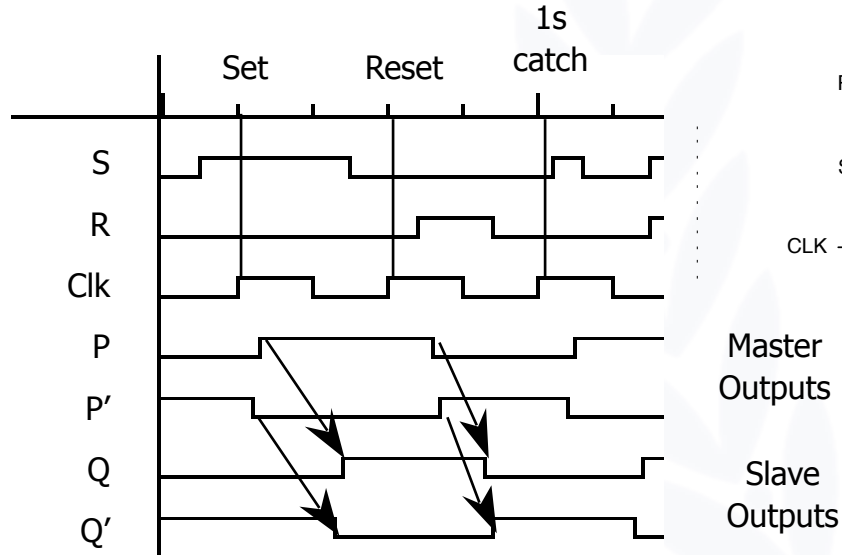
# Master-slave structure

- Break flow by alternating clocks (like an air-lock)
  - Use positive clock to latch inputs into one R-S latch
  - Use negative clock to change outputs with another R-S latch
- View pair as one basic unit
  - Master-slave flip-flop
  - Twice as much logic
  - Output changes a few gate delays after the falling edge of clock but does not affect any cascaded flip-flops



# The 1s catching problem

- In first R-S stage of master-slave FF
  - 0-1-0 glitch on R or S while clock is high is "caught" by master stage
    - Either  $\text{Clk}=0$  or  $R/S=0$  results in the same effect
    - Static 0 hazards on R during  $\text{Clk}=1$  makes the master stage reset
    - Static 0 hazard on S during  $\text{Clk}=1$  makes the master stage set
  - Once the master stage is set/reset, the slave stage will be set/reset when  $\text{Clk}=\downarrow$
  - Leads to constraints on logic to be hazard-free
    - Loss of advantage of synchronous design (free from hazard consideration)



# J-K flip flop

- Truth table

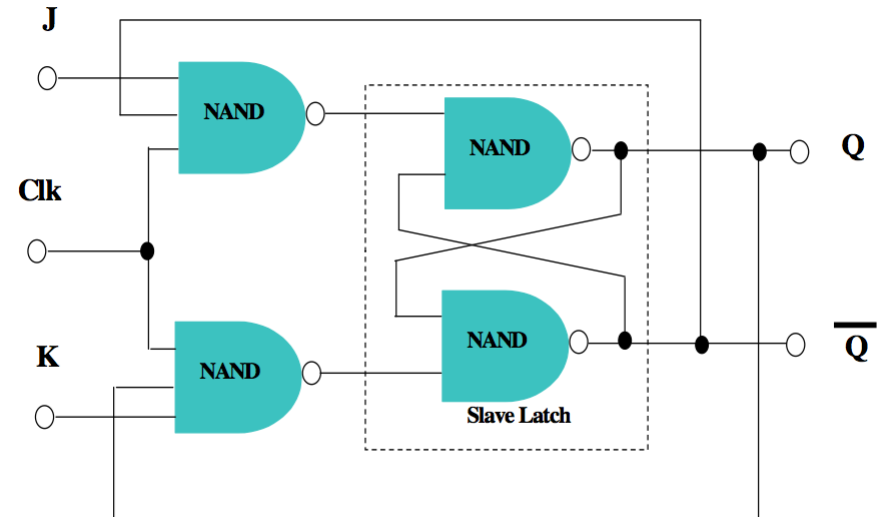
- Basic

J	K	Q
0	0	no change
1	0	1
0	1	0
1	1	toggle

- Clocked

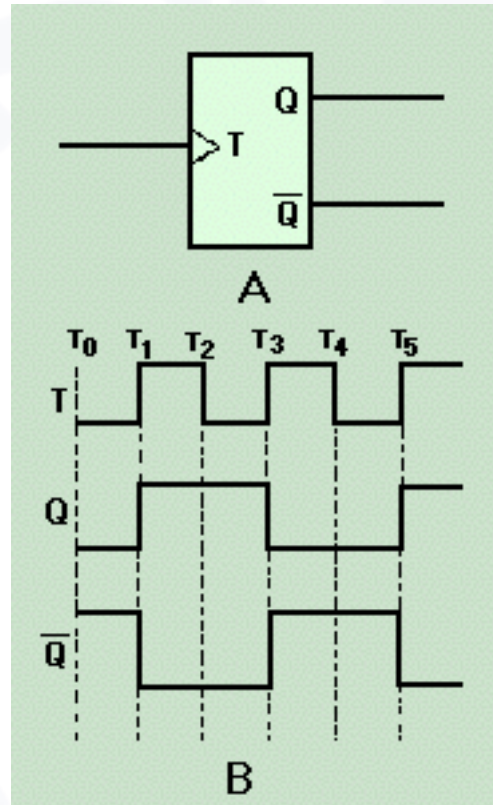
J	K	Clk	Q	Q_bar
0	0	Pos-edge	No change	
0	1	Pos-edge	0	1
1	0	Pos-edge	1	0
1	1	Pos-edge	Toggle	

- Both R and S cannot be 1 at the same time



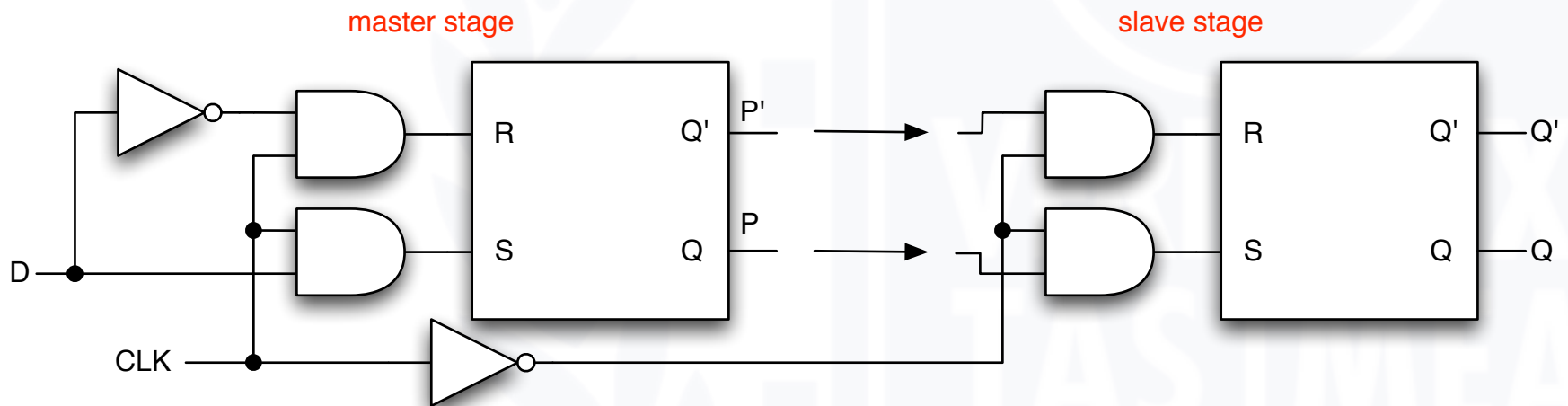
# Toggle flip flop

- Divide by 2
- 50% duty output



# D flip-flop

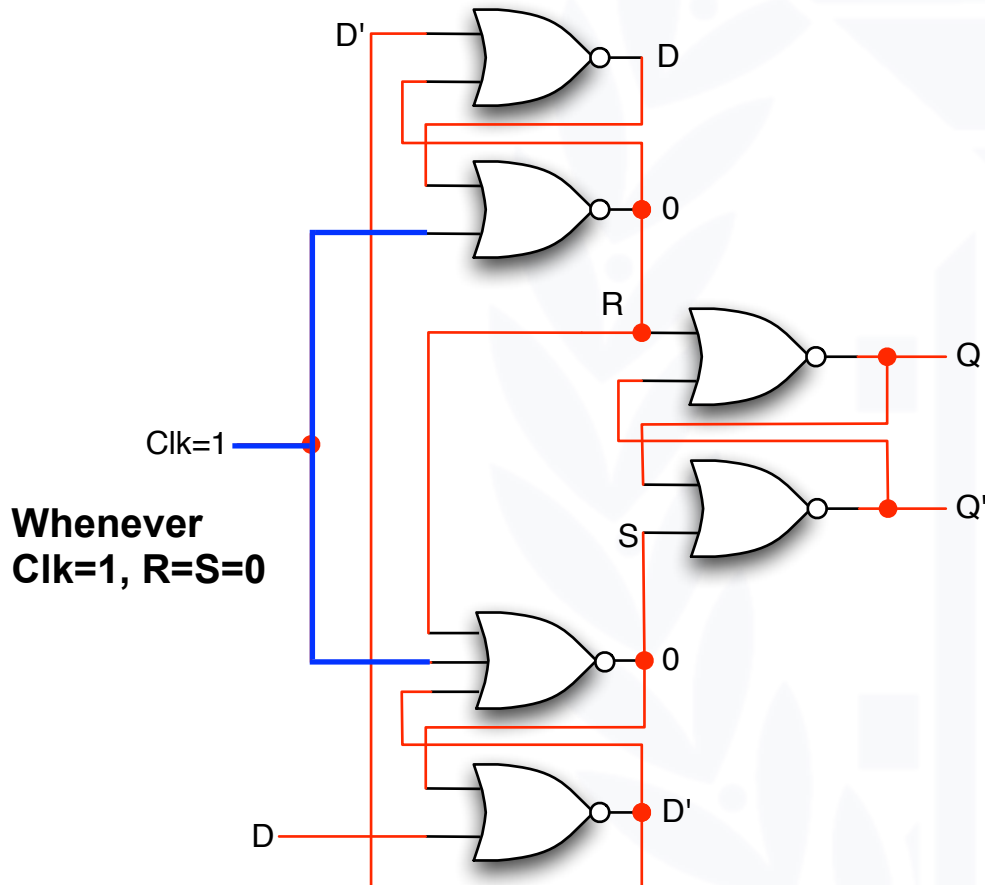
- Make S and R complements of each other
  - Eliminates 1s catching problem
  - Can't just hold previous value (must have new value ready every clock period)
  - Value of D just before clock goes low is what is stored in flip-flop
  - Can make R-S flip-flop by adding logic to make  $D = S + R' Q$





## Edge-triggered flip-flops

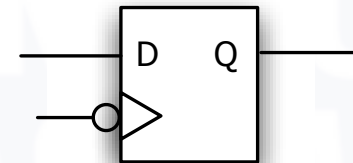
- More efficient solution: only 6 gates
- Sensitive to inputs only near edge of clock signal (not while high)



## Negative edge-triggered D flip-flop (D-FF)

4-5 gate delays

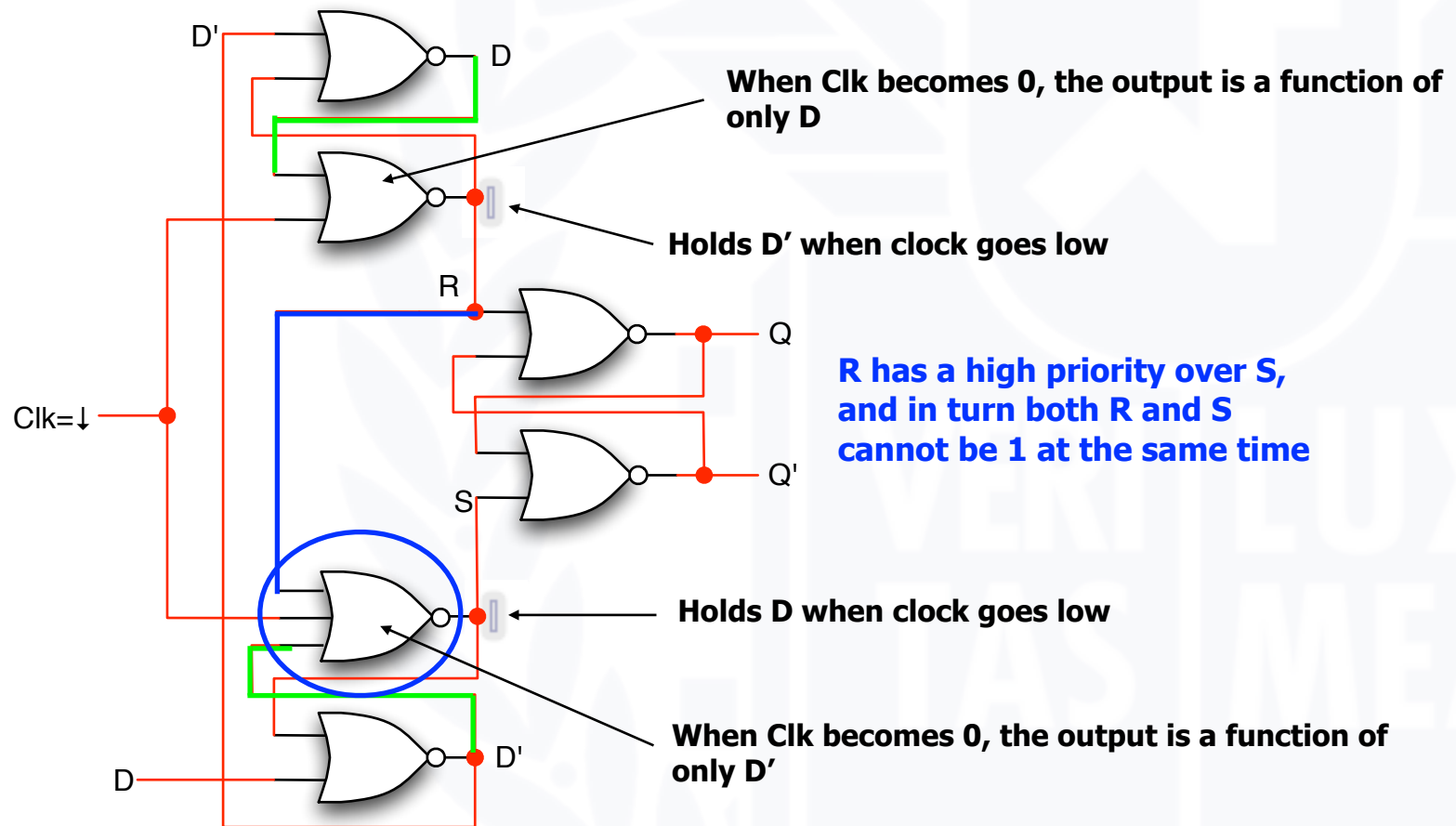
Must respect setup and hold time constraints to successfully capture input



characteristic equation  
 $Q(t+1) = D$

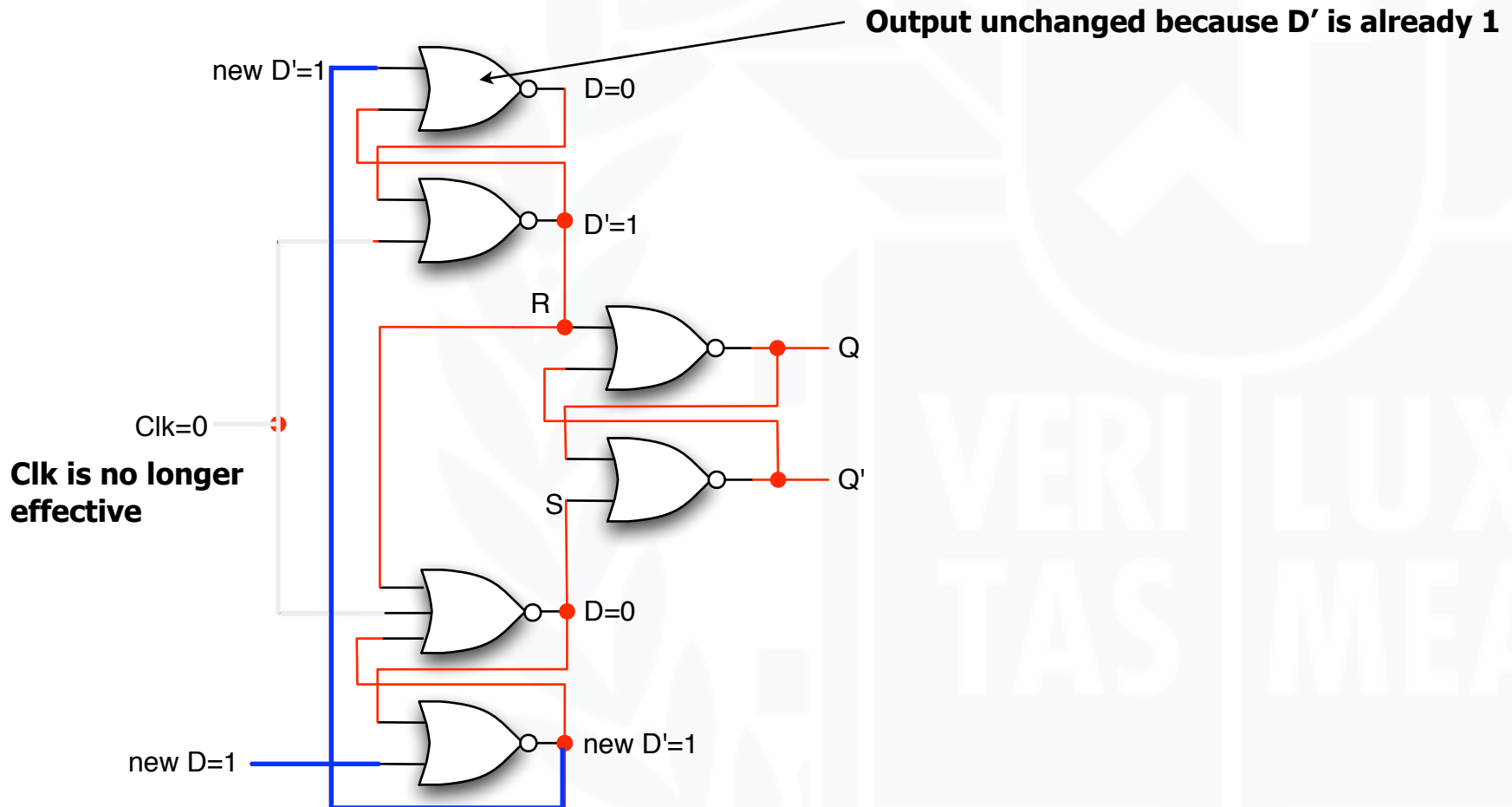
# Edge-triggered flip-flops

- When Clk goes from high to low ( $\text{Clk}=\downarrow$ )



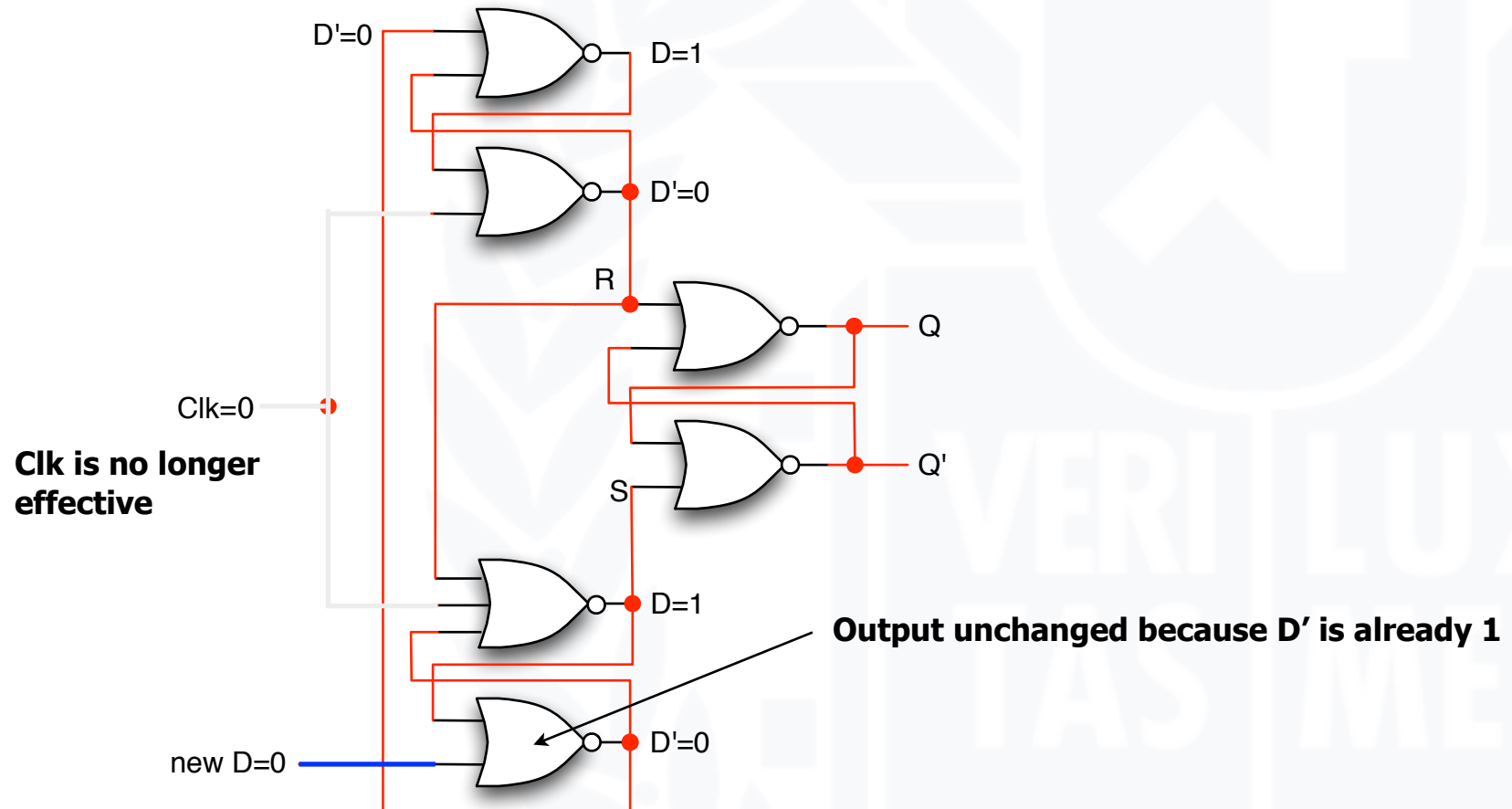
# Edge-triggered flip-flops (cont'd)

- When  $\text{Clk}=\text{low}$ ,  $D=0$  and new  $D=1$ 
  - $D$  is held



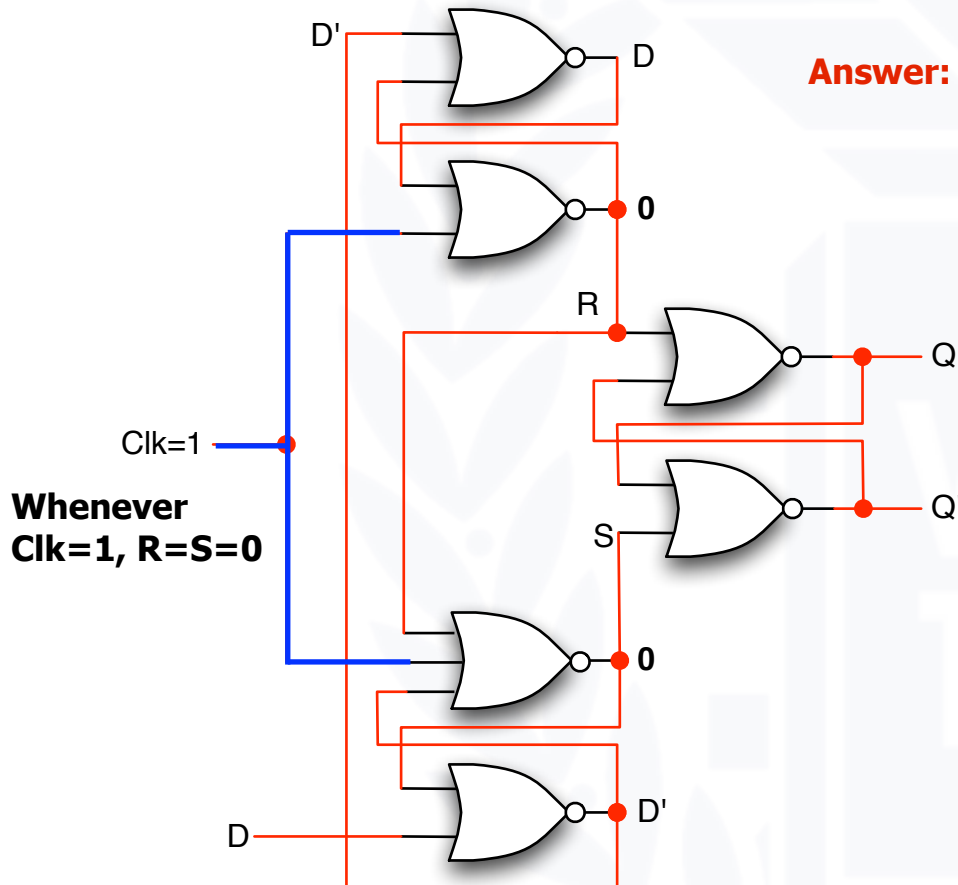
# Edge-triggered flip-flops (cont'd)

- When  $\text{Clk}=\text{low}$ ,  $D=1$  and new  $D=0$ 
  - $D$  is held



# Edge-triggered flip-flops

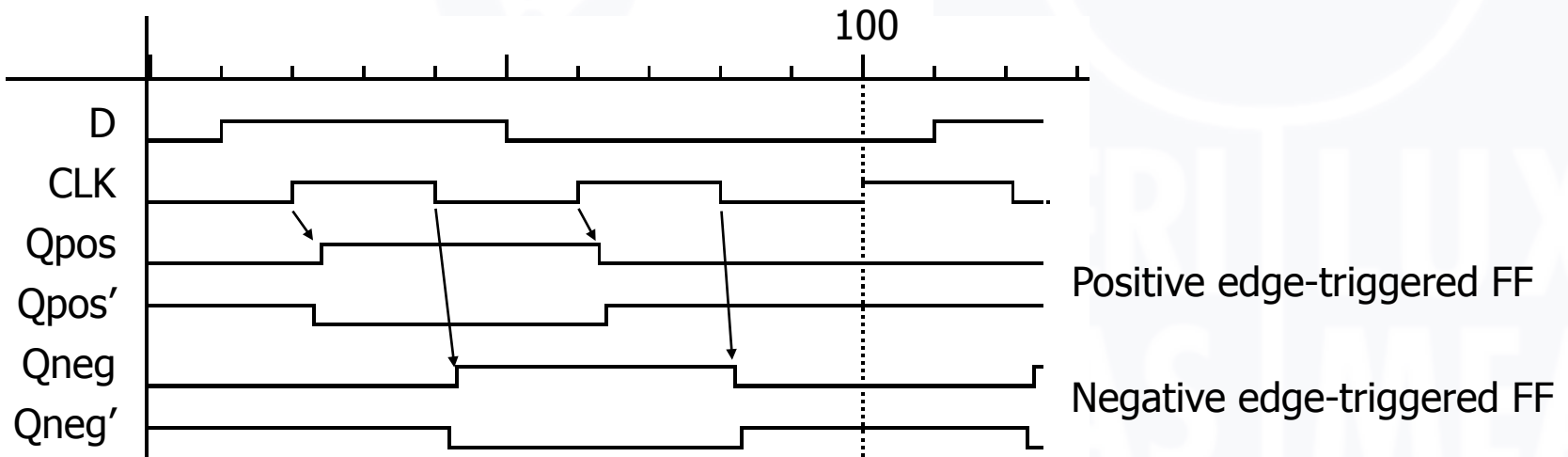
- Wrap up
  - Then when the FF becomes ready for a new sampling?



**Answer: when  $\text{Clk}=1$ , and in turn  $\text{R}=\text{S}=1$**

# Edge-triggered flip-flops (cont'd)

- Positive edge-triggered
  - Inputs sampled on rising edge; outputs change after rising edge
- Negative edge-triggered flip-flops
  - Inputs sampled on falling edge; outputs change after falling edge



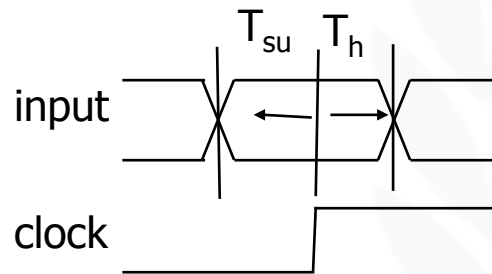
# Timing methodologies

- Rules for interconnecting components and clocks
  - Guarantee proper operation of system when strictly followed
- Approach depends on building blocks used for memory elements
  - We'll focus on systems with edge-triggered flip-flops
    - Found in programmable logic devices
  - Many custom integrated circuits focus on level-sensitive latches
- Basic rules for correct timing:
  - Correct inputs, with respect to time, are provided to the flip-flops
  - No flip-flop changes state more than once per clocking event

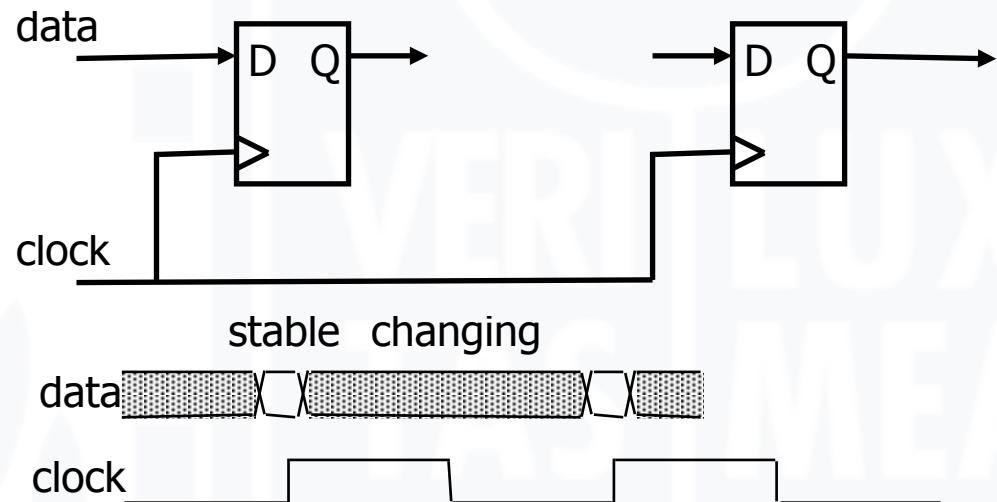
# Timing methodologies (cont'd)

## Definition of terms

- Clock: periodic event, causes state of memory element to change  
can be rising edge or falling edge or high level or low level
- Setup time: minimum time before the clocking event by which the input must be stable ( $T_{su}$ )
- Hold time: minimum time after the clocking event until which the input must remain stable ( $T_h$ )

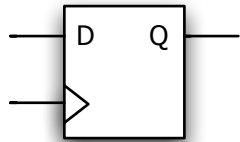


There is a timing "window" around the clocking event during which the input must remain stable and unchanged in order to be recognized

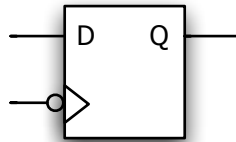




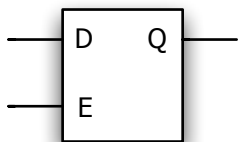
# Comparison of latches and flip-flops



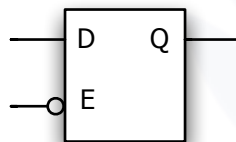
Positive edge D-FF



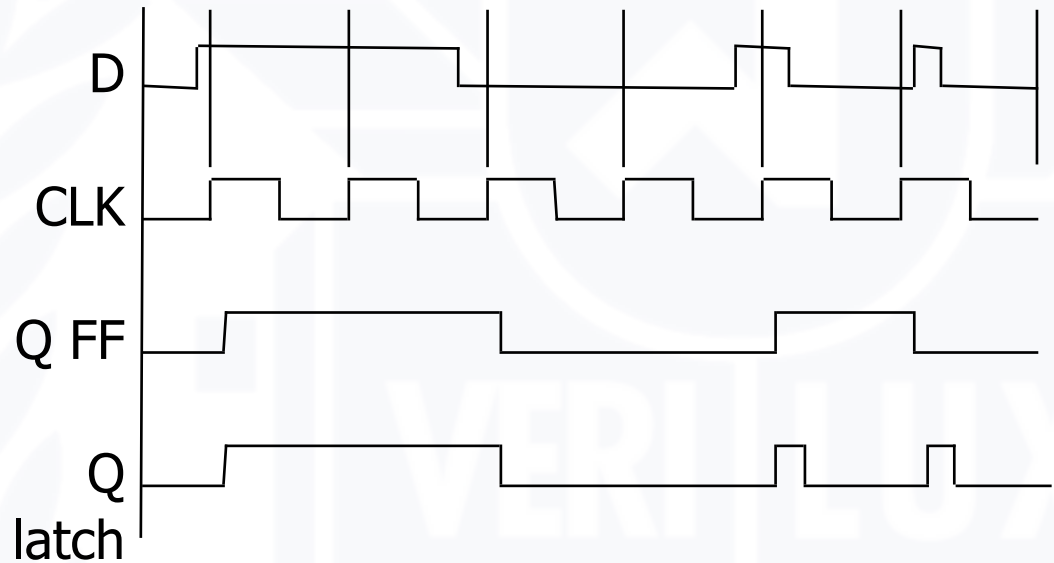
Negative edge D-FF



Positive edge D-latch



Negative edge D-latch



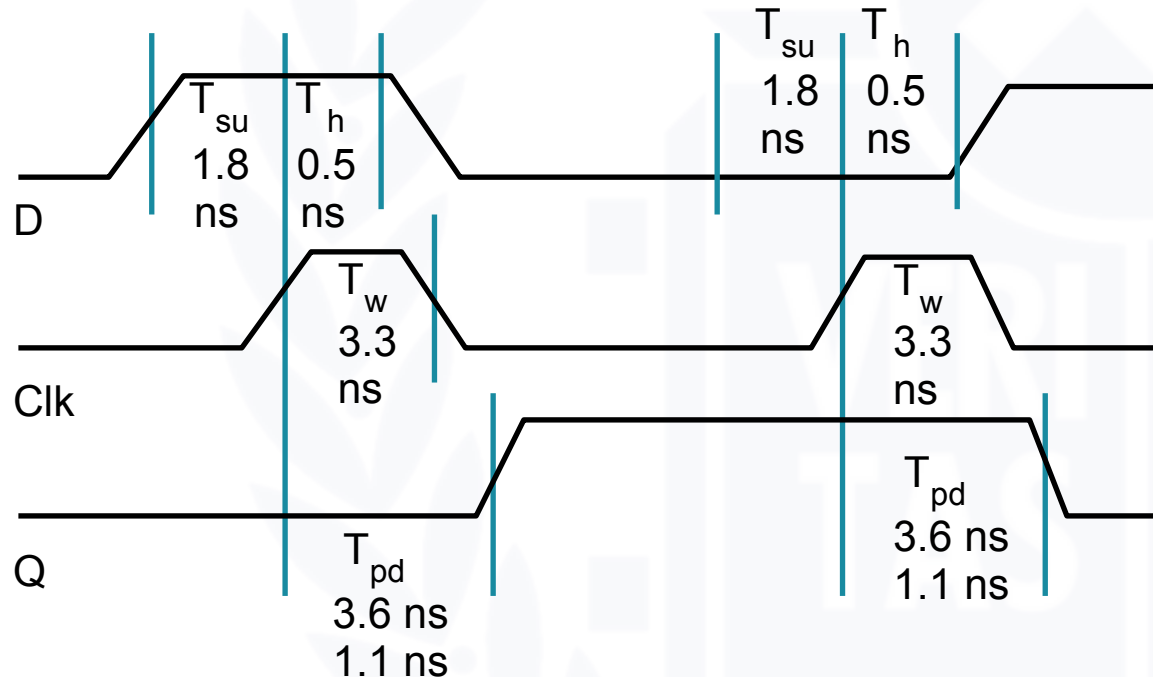
Behavior is the same unless input changes while the clock is high

# Comparison of latches and flip-flops (cont'd)

<u>Type</u>	<u>When inputs are sampled</u>	<u>When output is valid</u>
unclocked latch	always	propagation delay from input change
level-sensitive latch	clock high ( $T_{su}/T_h$ around falling edge of clock)	propagation delay from input change or clock edge (whichever is later)
master-slave flip-flop	clock high ( $T_{su}/T_h$ around falling edge of clock)	propagation delay from falling edge of clock
negative edge-triggered flip-flop	clock hi-to-lo transition ( $T_{su}/T_h$ around falling edge of clock)	propagation delay from falling edge of clock

# Typical timing specifications

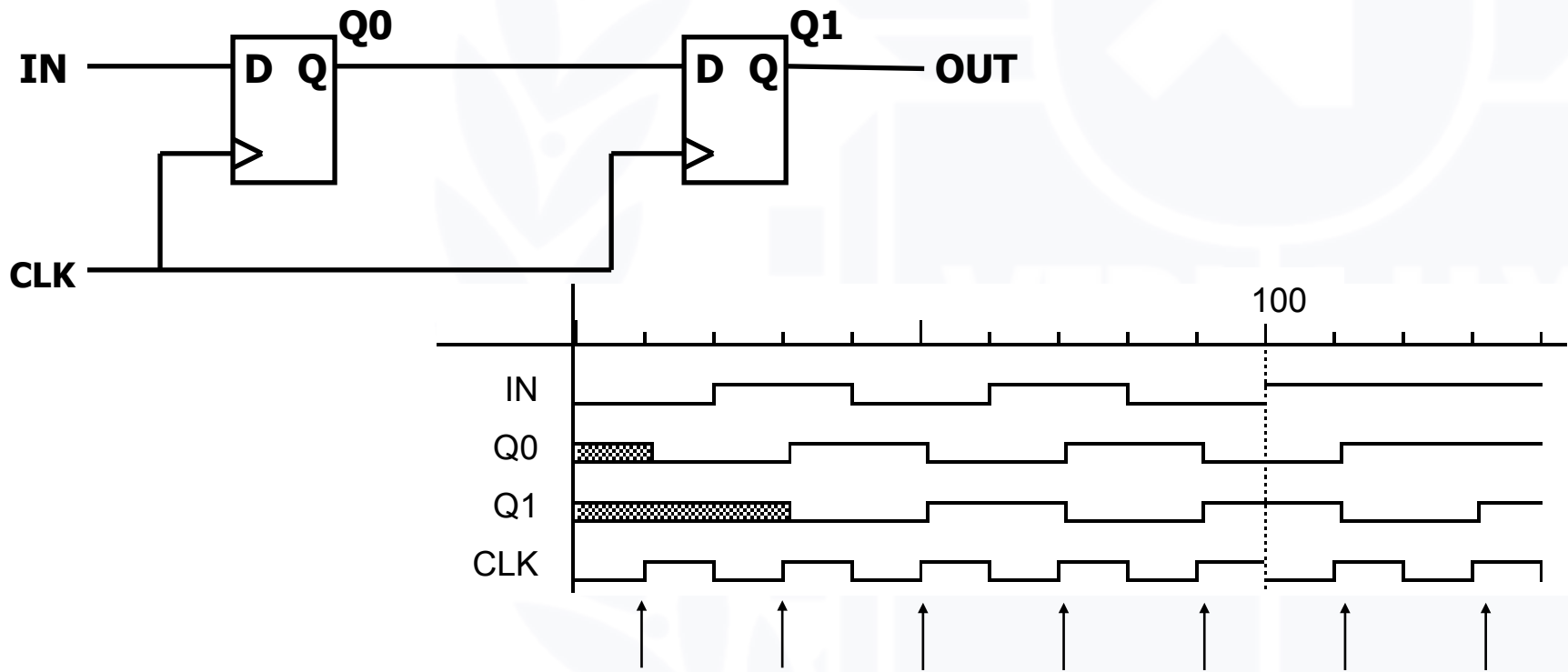
- Positive edge-triggered D flip-flop
  - Setup and hold times
  - Minimum clock width
  - Propagation delays (low to high, high to low, max and typical)



All measurements are made from the clocking event (the rising edge of the clock)

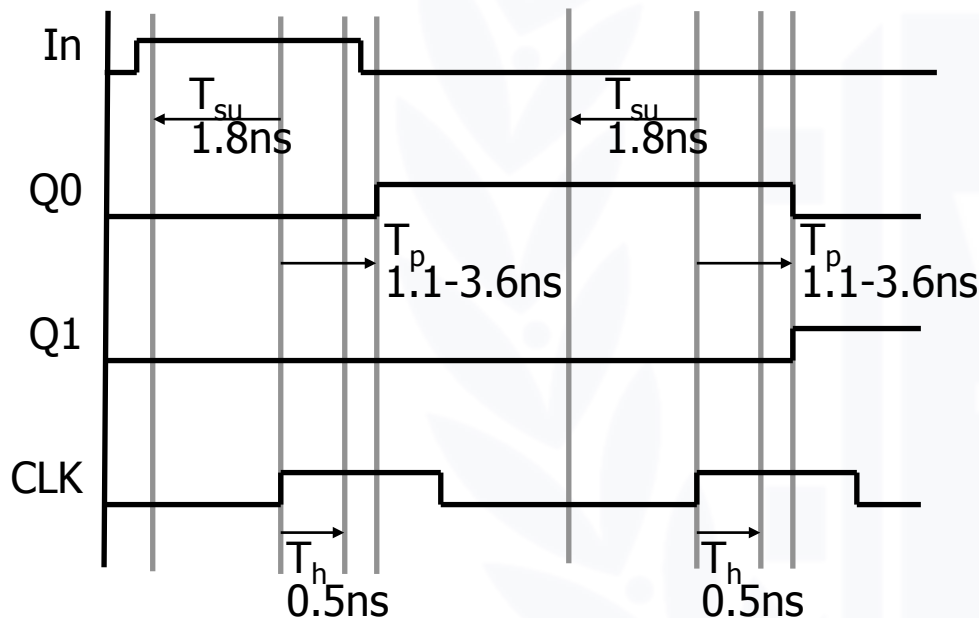
# Cascading edge-triggered flip-flops

- Shift register
  - New value goes into first stage
  - While previous value of first stage goes into second stage
  - Consider setup/hold/propagation delays (prop must be  $>$  hold)



# Cascading edge-triggered flip-flops (cont'd)

- Why this works
  - Propagation delays exceed hold times
  - Clock width constraint exceeds setup time
  - This guarantees following stage will latch current value before it changes to new value

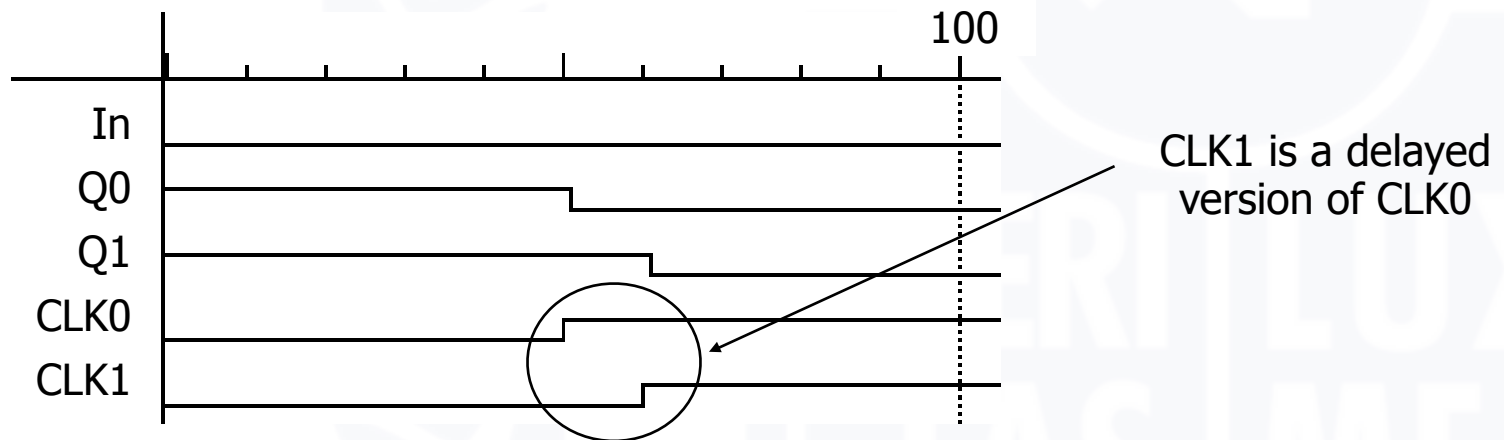


Timing constraints guarantee proper operation of cascaded components

Assumes infinitely fast distribution of the clock

# Clock skew

- The problem
  - Correct behavior assumes next state of all storage elements determined by all storage elements at the same time
  - This is difficult in high-performance systems because time for clock to arrive at flip-flop is comparable to delays through logic
  - Effect of skew on cascaded flip-flops:



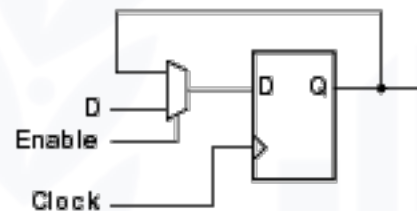
original state:  $IN = 0, Q0 = 1, Q1 = 1$

due to skew, next state becomes:  $Q0 = 0, Q1 = 0$ , and not  $Q0 = 0, Q1 = 1$

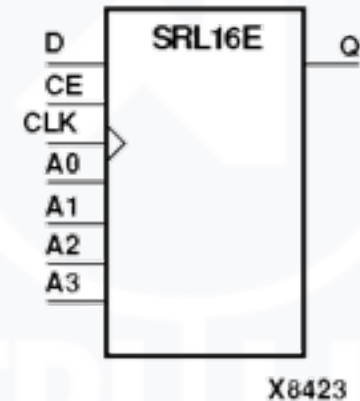
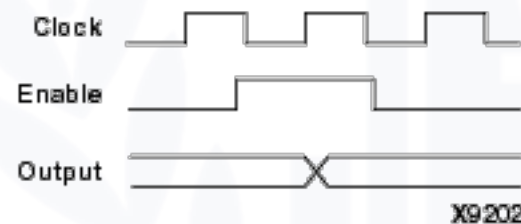
# Clock gating and clock enable

- Clock enable
  - Functionally (logically) disable the clock but not physically
  - No clock path change, no delay or skew change

a) Using a Feedback Path

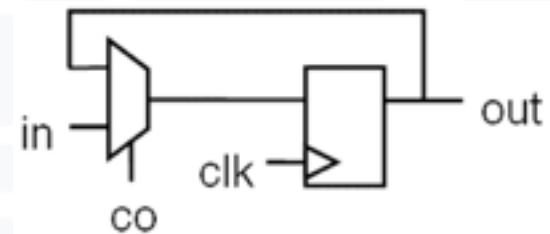
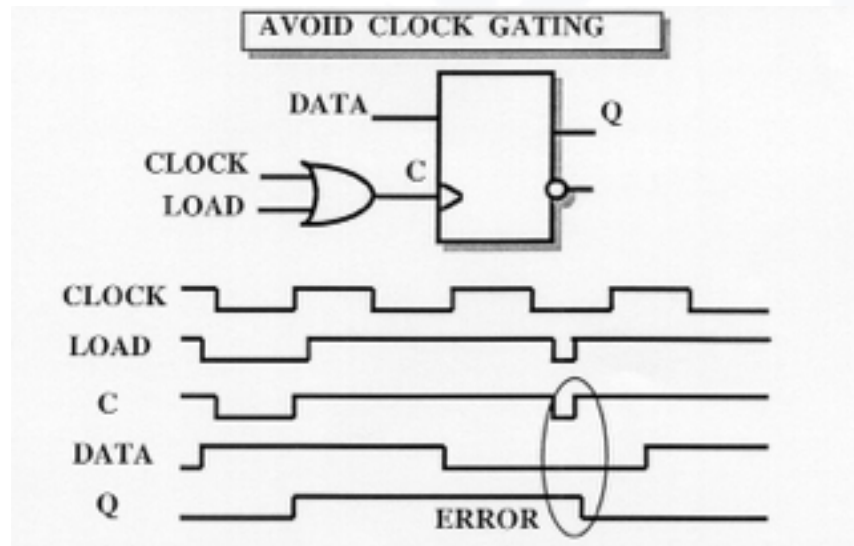


b) Corresponding Timing Diagram

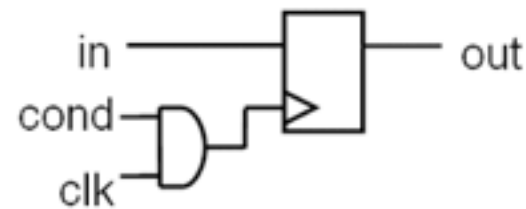


# Clock gating and clock enable

- Clock gating
  - Physically block the clock
  - Serious timing change
  - Should be used with low-level physical design support to prevent from clock skew



Transformation  
does not change  
the state of the  
flops and registers





# Flip-flop features

- Reset (set state to 0) – R
  - Synchronous:  $D_{\text{new}} = R' \cdot D_{\text{old}}$  (when next clock edge arrives)
  - Asynchronous: doesn't wait for clock, quick but dangerous
- Preset or set (set state to 1) – S (or sometimes P)
  - Synchronous:  $D_{\text{new}} = D_{\text{old}} + S$  (when next clock edge arrives)
  - Asynchronous: doesn't wait for clock, quick but dangerous
- Both reset and preset
  - $D_{\text{new}} = R' \cdot D_{\text{old}} + S$  (set-dominant)
  - $D_{\text{new}} = R' \cdot D_{\text{old}} + R'S$  (reset-dominant)
- Selective input capability (input enable or load) – LD or EN
  - Multiplexor at input:  $D_{\text{new}} = LD' \cdot Q + LD \cdot D_{\text{old}}$
  - Load may or may not override reset/set (usually R/S have priority)
- Complementary outputs – Q and Q'

# Summary of latches and flip-flops

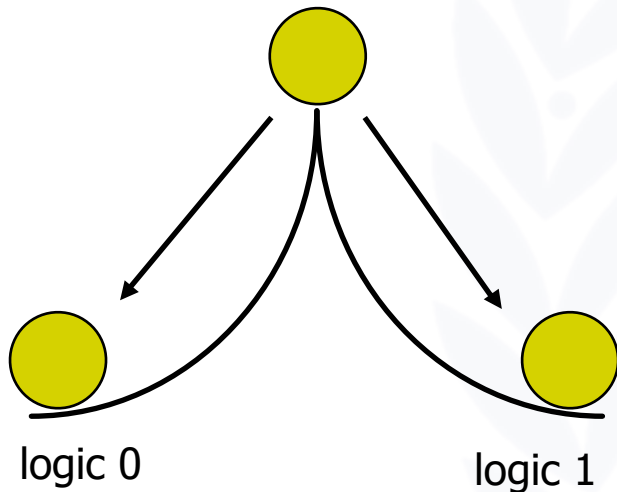
- Development of D-FF
  - Level-sensitive used in custom integrated circuits
    - Can be made with 4 switches
  - Edge-triggered used in programmable logic devices
  - Good choice for data storage register
- Historically J-K FF was popular but now never used
  - Similar to R-S but with 1-1 being used to toggle output (complement state)
  - Good in days of TTL/SSI (more complex input function:  $D = J Q' + K' Q$ )
  - Not a good choice for PALs/PLAs as it requires 2 inputs
  - Can always be implemented using D-FF
- Preset and clear inputs are highly desirable on flip-flops
  - Used at start-up or to reset system to a known state

# Metastability and asynchronous inputs

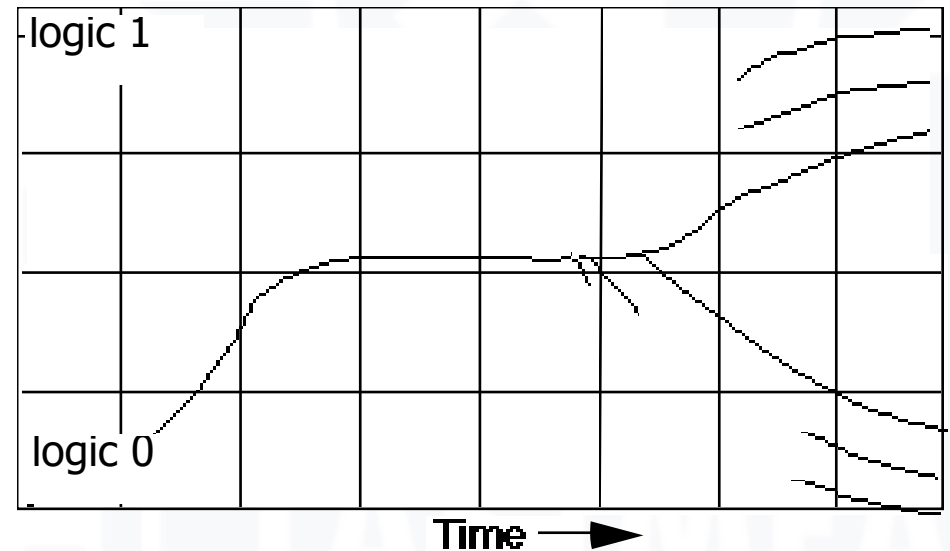
- Clocked synchronous circuits
  - Inputs, state, and outputs sampled or changed in relation to a common reference signal (called the clock)
  - e.g., master/slave, edge-triggered
- Asynchronous circuits
  - Inputs, state, and outputs sampled or changed independently of a common reference signal (glitches/hazards a major concern)
  - e.g., R-S latch
- Asynchronous inputs to synchronous circuits
  - Inputs can change at any time, will not meet setup/hold times
  - Dangerous, synchronous inputs are greatly preferred
  - Cannot be avoided (e.g., reset signal, memory wait, user input)

# Synchronization failure

- Occurs when FF input changes close to clock edge
  - The FF may enter a metastable state – neither a logic 0 nor 1 –
  - It may stay in this state an indefinite amount of time
  - This is not likely in practice but has some probability



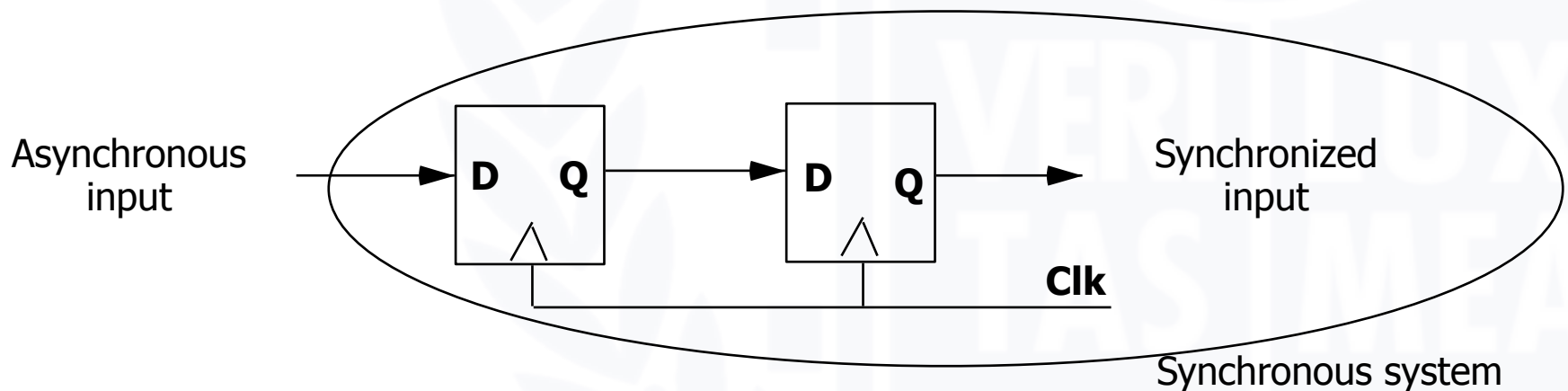
Small, but non-zero probability that the FF output will get stuck in an in-between state



Oscilloscope traces demonstrating synchronizer failure and eventual decay to steady state

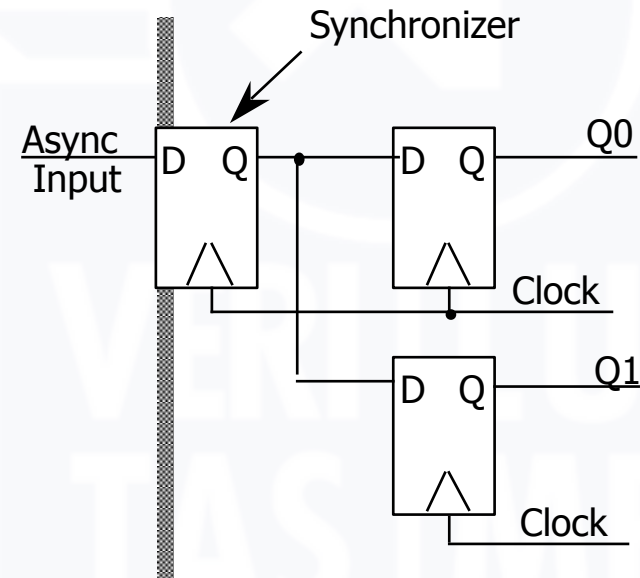
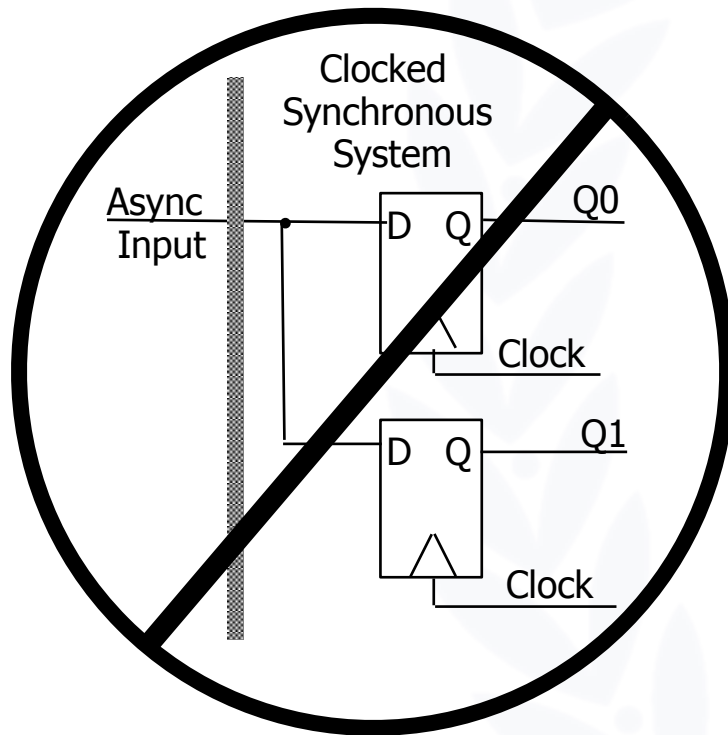
# Dealing with synchronization failure

- Probability of failure can never be reduced to 0, but it can be reduced
  - Slow down the system clock  
this gives the synchronizer more time to decay into a steady state;  
synchronizer failure becomes a big problem for very high speed systems
  - Use fastest possible logic technology in the synchronizer  
this makes for a very sharp "peak" upon which to balance
  - Cascade two synchronizers  
this effectively synchronizes twice (both would have to fail)



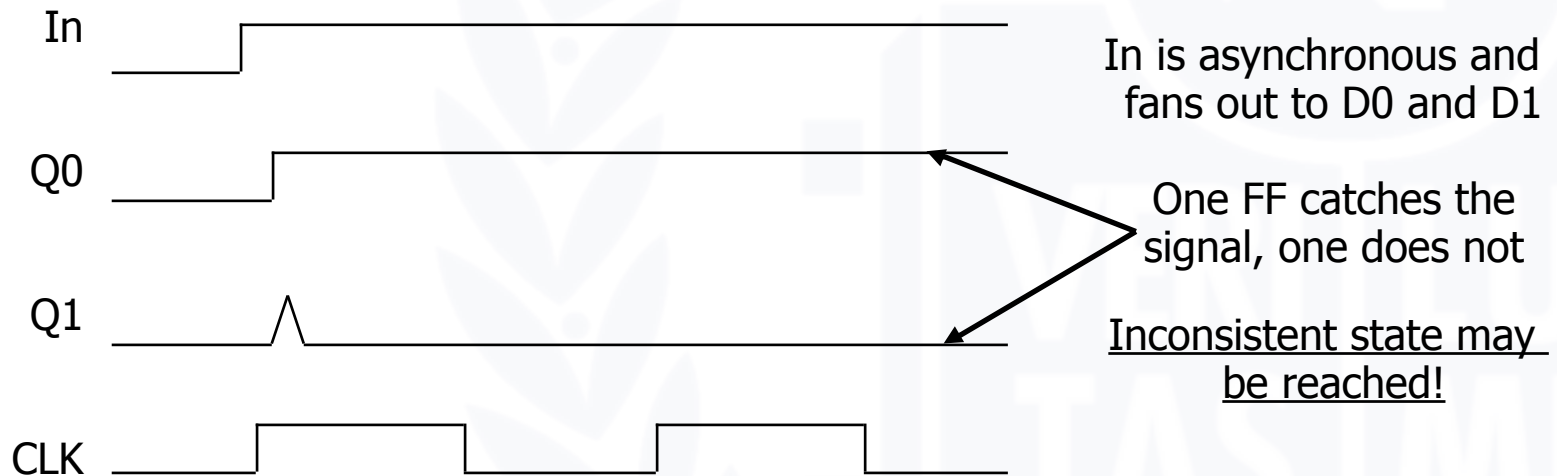
# Handling asynchronous inputs

- Never allow asynchronous inputs to fan-out to more than one flip-flop
  - Synchronize as soon as possible and then treat as synchronous signal



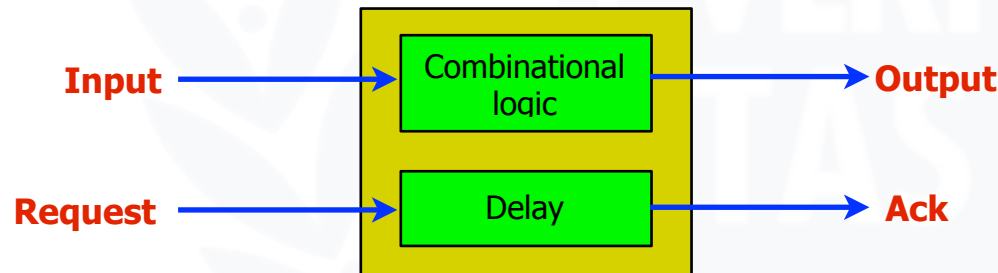
# Handling asynchronous inputs (cont'd)

- What can go wrong?
  - Input changes too close to clock edge (violating setup time constraint)



# Self-timed and speed-independent circuits

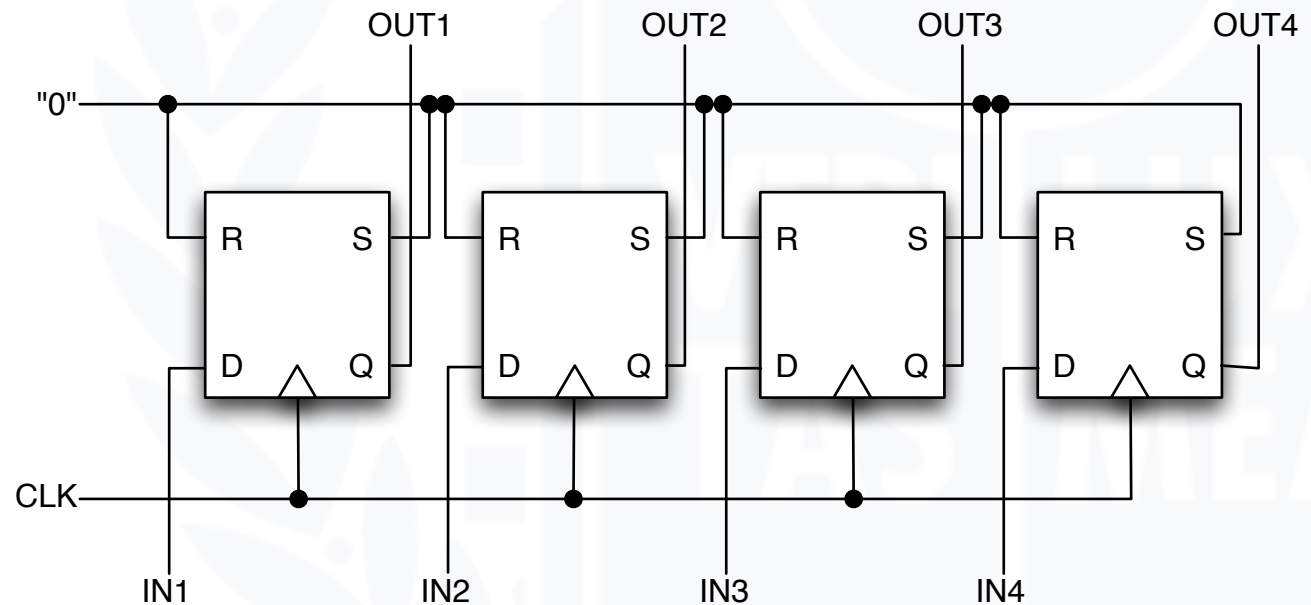
- Large physical size is hard to distribute a clock w/o intolerable clock skew
  - GALS (globally asynchronous and locally synchronous)
  - Independently clocked subsystems
- Delay-insensitive signaling
  - Between two independently clocked subsystems
    - Request/acknowledgement signaling
    - Synchronous request/acknowledgement signaling
    - Synchronous request with wait signaling
    - Four-cycle asynchronous signaling
    - Two-cycle asynchronous signaling
- Self-timed circuit
  - Locally asynchronous circuit





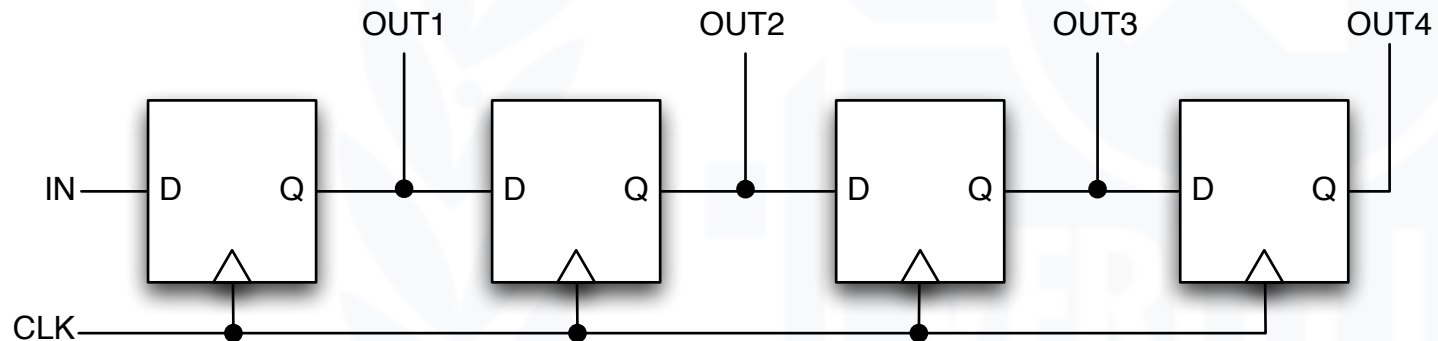
# Registers

- Collections of flip-flops with similar controls and logic
  - Stored values somehow related (for example, form binary value)
  - Share clock, reset, and set lines
  - Similar logic at each stage
- Examples
  - Shift registers
  - Counters



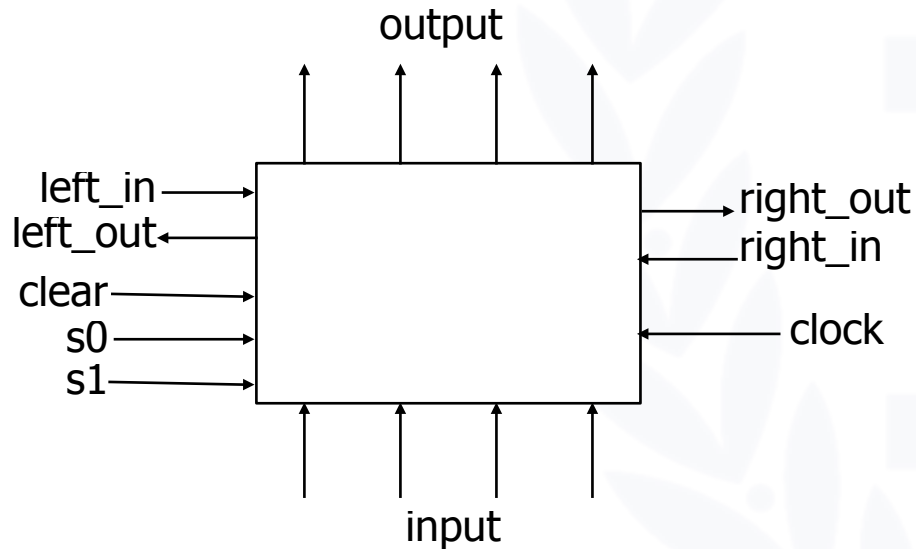
# Shift register

- Holds samples of input
  - Store last 4 input values in sequence
  - 4-bit shift register:



# Universal shift register

- Holds 4 values
  - Serial or parallel inputs
  - Serial or parallel outputs
  - Permits shift left or right
  - Shift in new values from left or right



Clear sets the register contents and output to 0

s1 and s0 determine the shift function

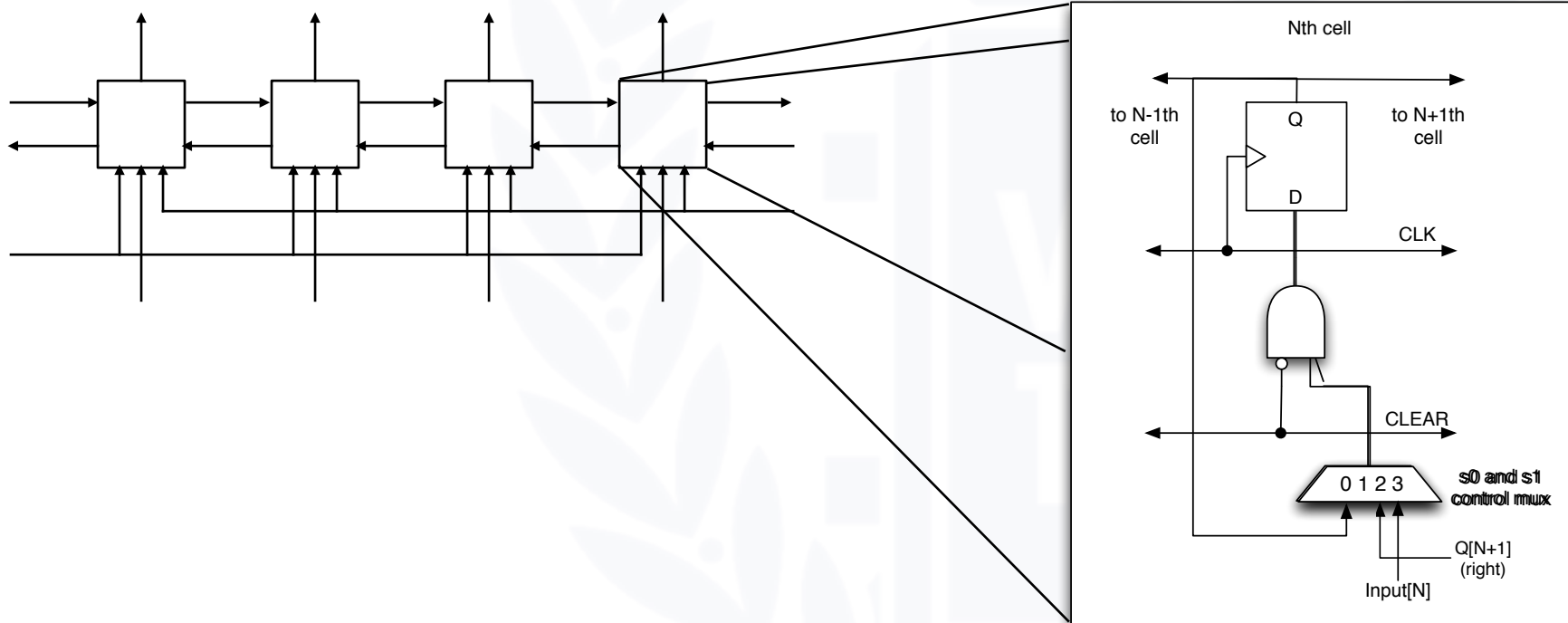
s0	s1	function
0	0	hold state
0	1	shift right
1	0	shift left
1	1	load new input

# Design of universal shift register

- Consider one of the four flip-flops

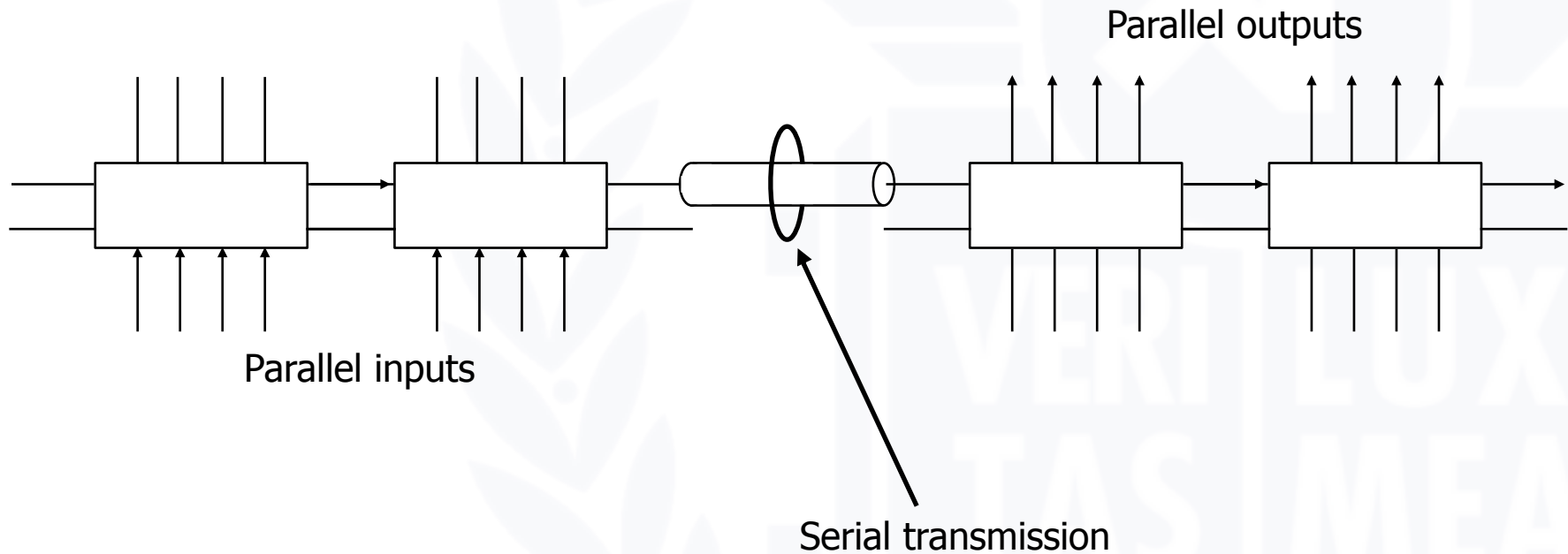
- New value at next clock cycle:

clear	s0	s1	new value
1	—	—	0
0	0	0	output
0	0	1	output value of FF to left (shift right)
0	1	0	output value of FF to right (shift left)
0	1	1	input



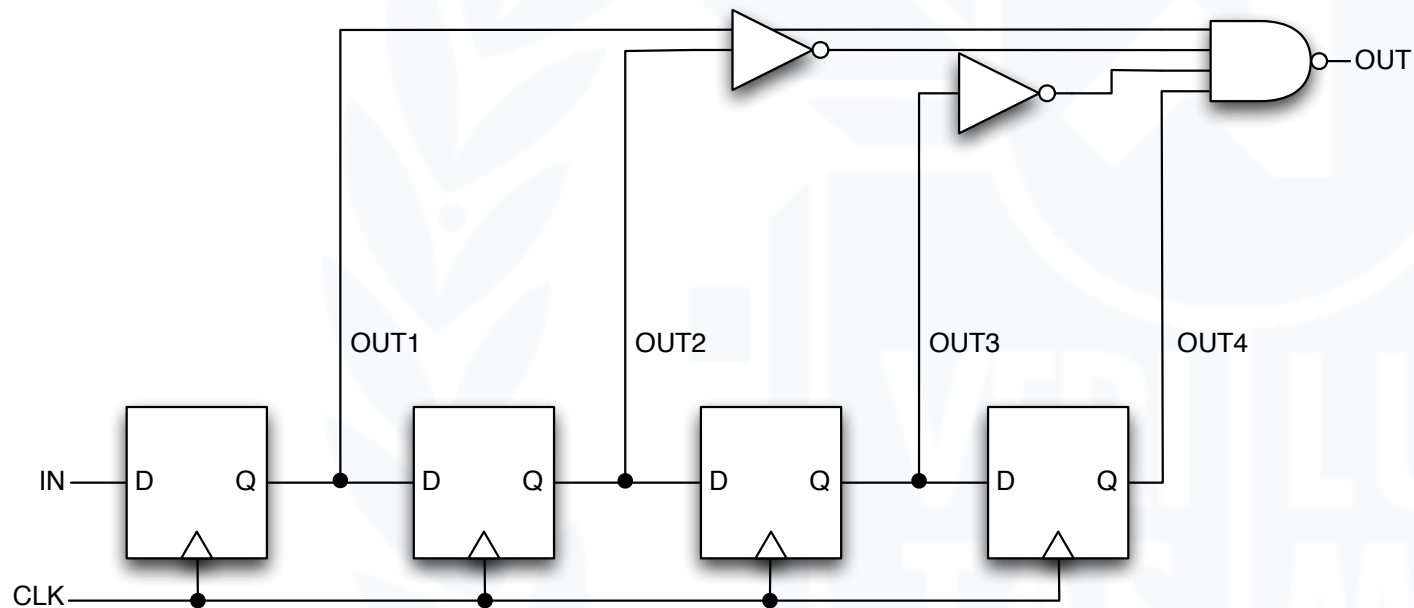
# Shift register application

- Parallel-to-serial conversion for serial transmission



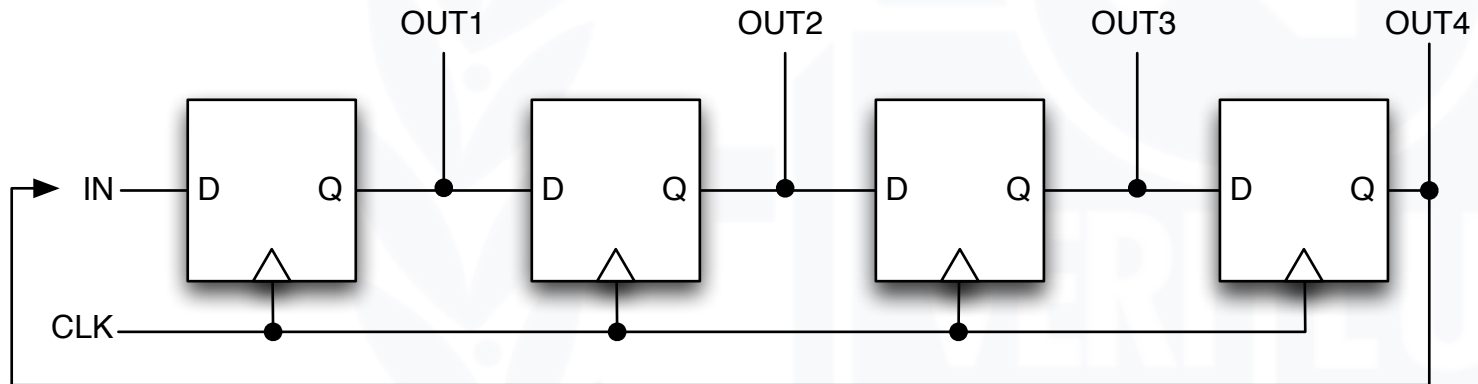
# Pattern recognizer

- Combinational function of input samples
  - In this case, recognizing the pattern 1001 on the single input signal



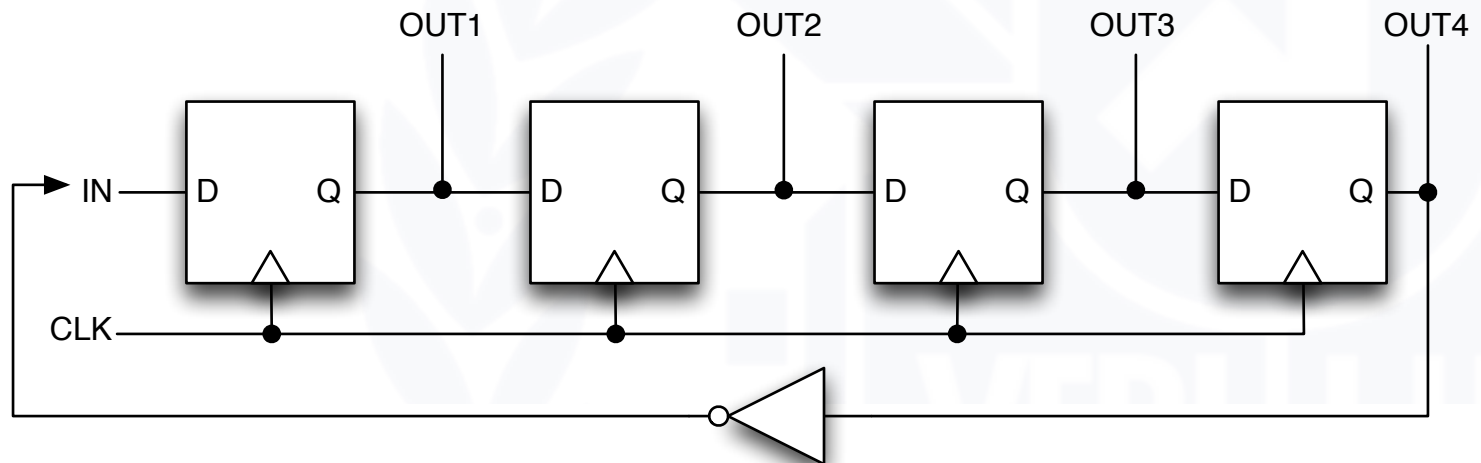
# Counters

- Sequences through a fixed set of patterns
  - In this case, 1000, 0100, 0010, 0001
  - If one of the patterns is its initial state (by loading or set/reset)



# Activity

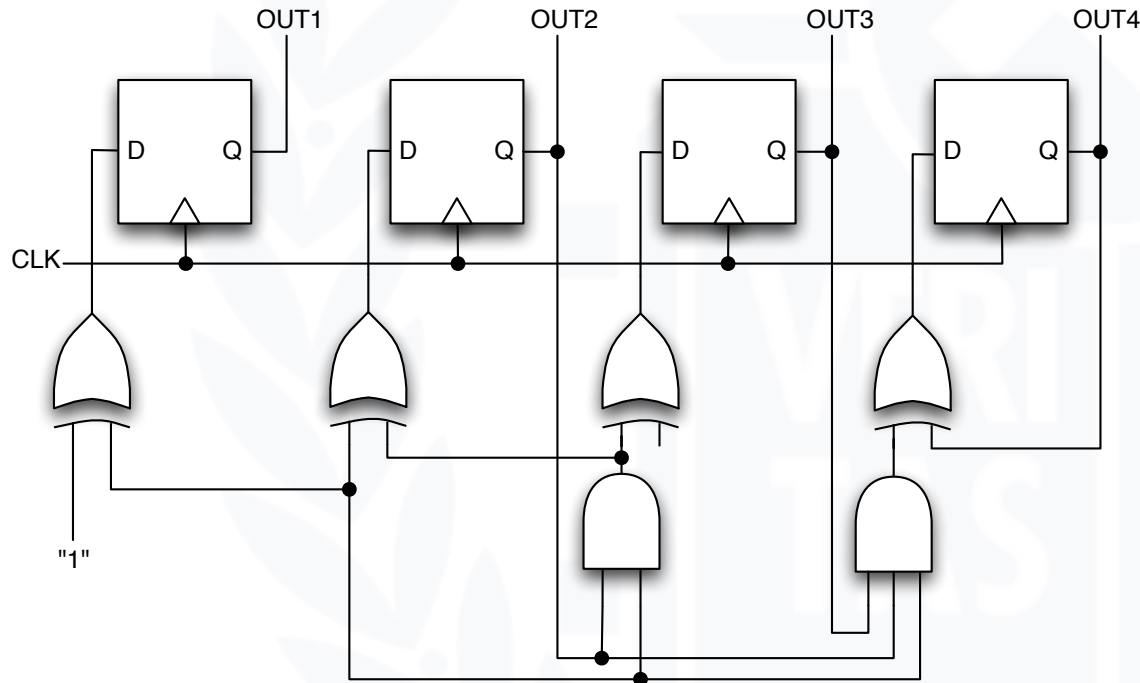
- How does this counter work?





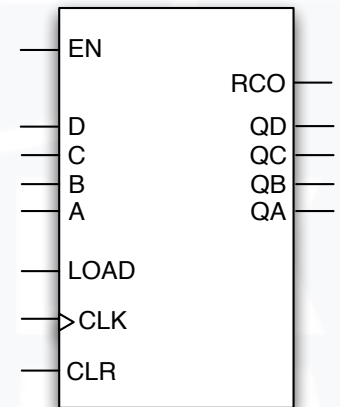
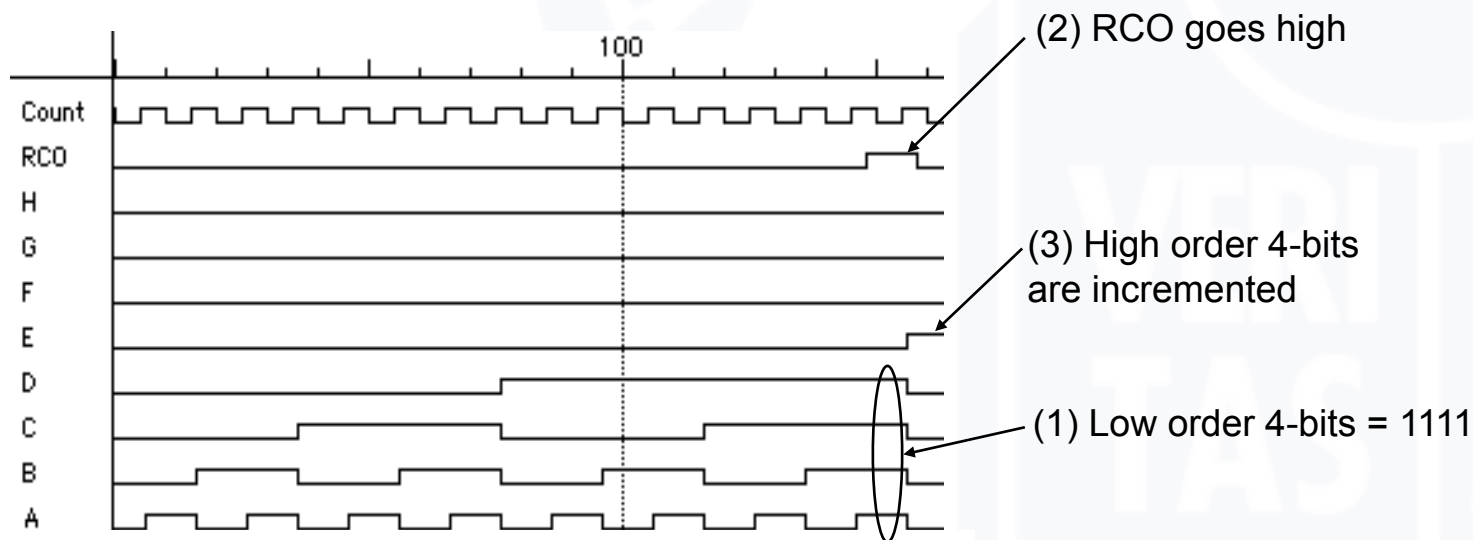
# Binary counter

- Logic between registers (not just multiplexer)
  - XOR decides when bit should be toggled
  - Always for low-order bit, only when first bit is true for second bit, and so on



# Four-bit binary synchronous up-counter

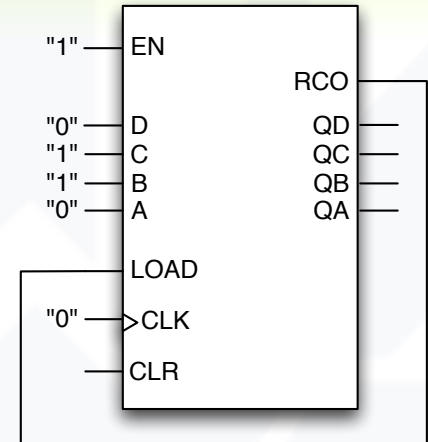
- Standard component with many applications
  - Positive edge-triggered FFs w/ synchronous load and clear inputs
  - Parallel load data from D, C, B, A
  - Enable inputs: must be asserted to enable counting
  - RCO: ripple-carry out used for cascading counters
    - High when counter is in its highest state 1111
    - Implemented using an AND gate



# Offset counters

- Starting offset counters – use of synchronous load

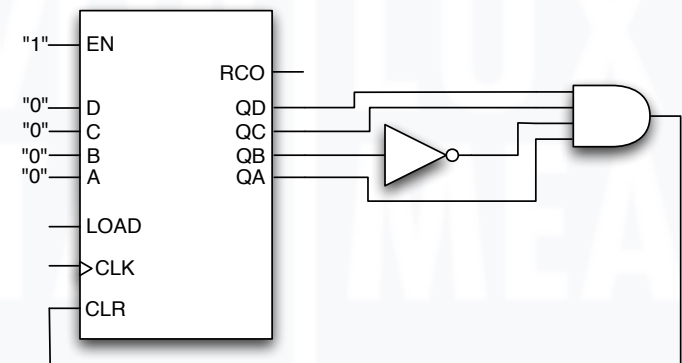
- e.g., 0110, 0111, 1000, 1001,  
1010, 1011, 1100, 1101, 1111, 0110, ...



- Ending offset counter – comparator for ending value

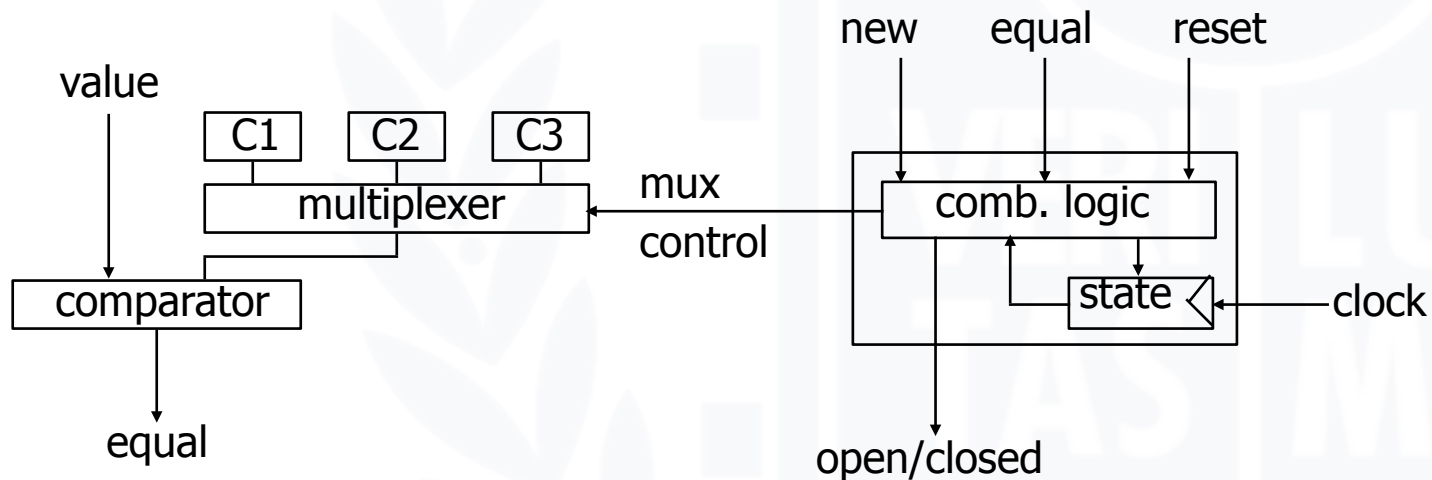
- e.g., 0000, 0001, 0010, ..., 1100, 1101, 0000

- Combinations of the above (start and stop value)



# Sequential circuits

- Circuits with feedback
  - Outputs =  $f(\text{inputs, past inputs, past outputs})$
  - Basis for building "memory" into logic circuits
  - Door combination lock is an example of a sequential circuit
    - State is memory
    - State is an "output" and an "input" to combinational logic
    - Combination storage elements are also memory



# Sequential logic summary

- Fundamental building block of circuits with state
  - Latch and flip-flop
  - R-S latch, R-S master/slave, D master/slave, edge-triggered D flip-flop
- Timing methodologies
  - Use of clocks
  - Cascaded FFs work because propagation delays exceed hold times
  - Beware of clock skew
- Asynchronous inputs and their dangers
  - Synchronizer failure: what it is and how to minimize its impact
- Basic registers
  - Shift registers
  - Counters