Digital Logic Circuit (SE273 – Fall 2020)

Lecture 7: Sequential Circuits

Jaesok Yu, Ph.D. (jaesok.yu@dgist.ac.kr)

Assistant Professor

Department of Robotics Engineering, DGIST

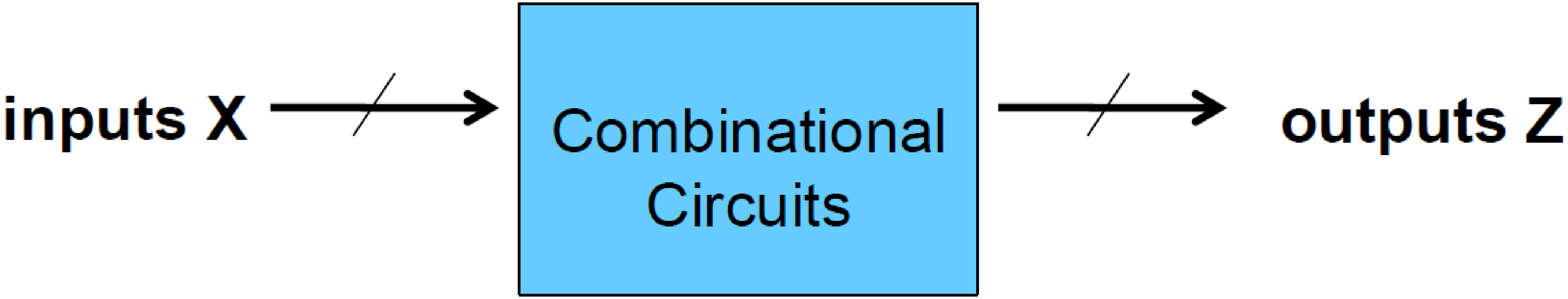
Goal

- Introduce the concept of sequential circuits
  - Sequential circuit definitions
  - Latches, flip-flops

- Different types of latches and flip-flops
  - SR latch, D latch
  - D flip-flop, SR flip-flop, and others

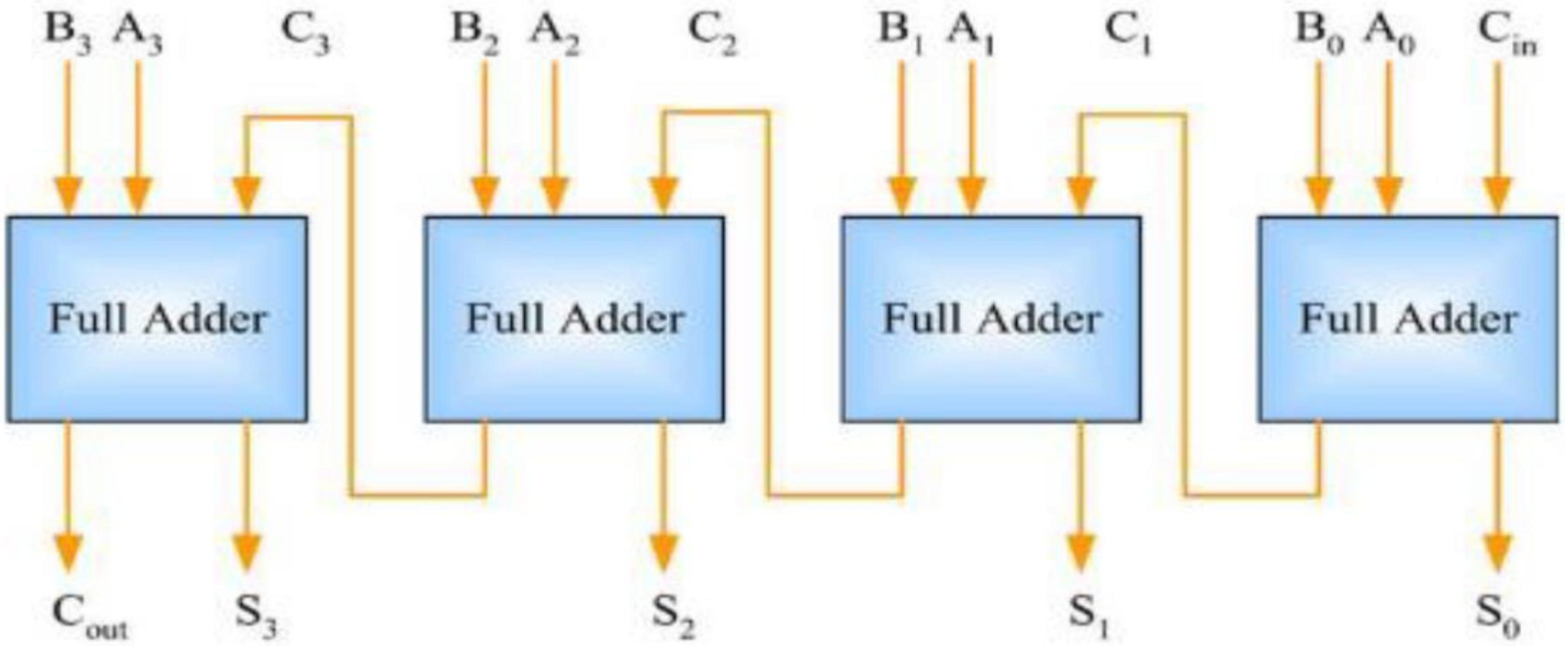
Combinational vs Sequential

- A combinational circuit:
  - At any time, **outputs depend only on inputs**
    - Changing inputs changes outputs
  - No regard for previous inputs
    - **No memory** (history)



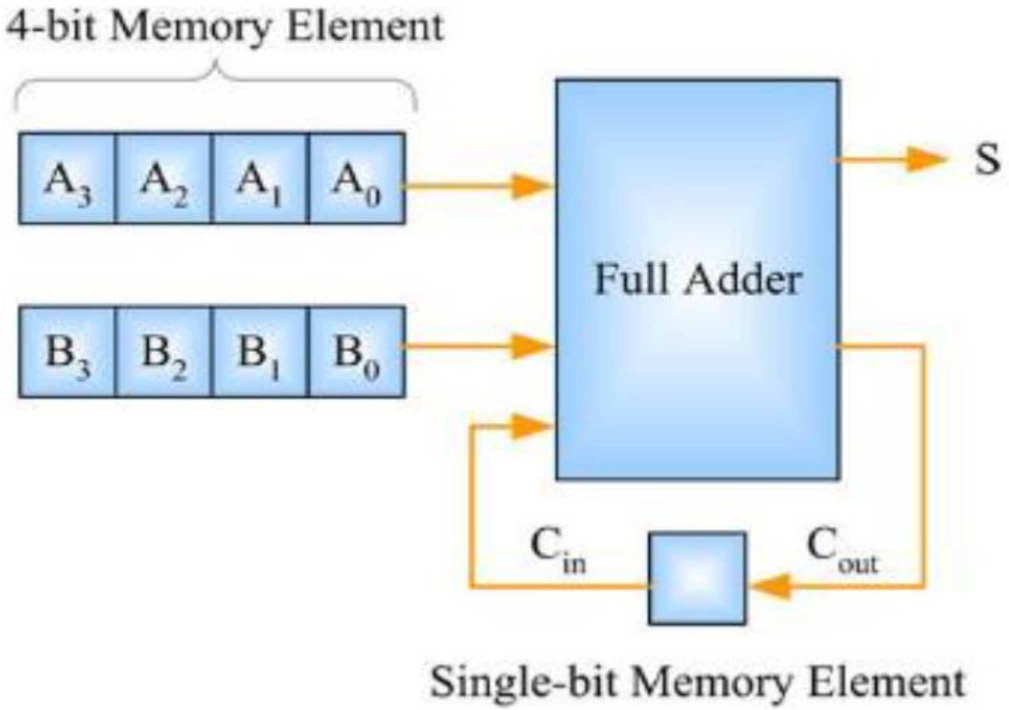
Combinational Adder

- 4-bit adder (ripple carry)
  - Notice how carry-out propagates
  - One adder is active at a time



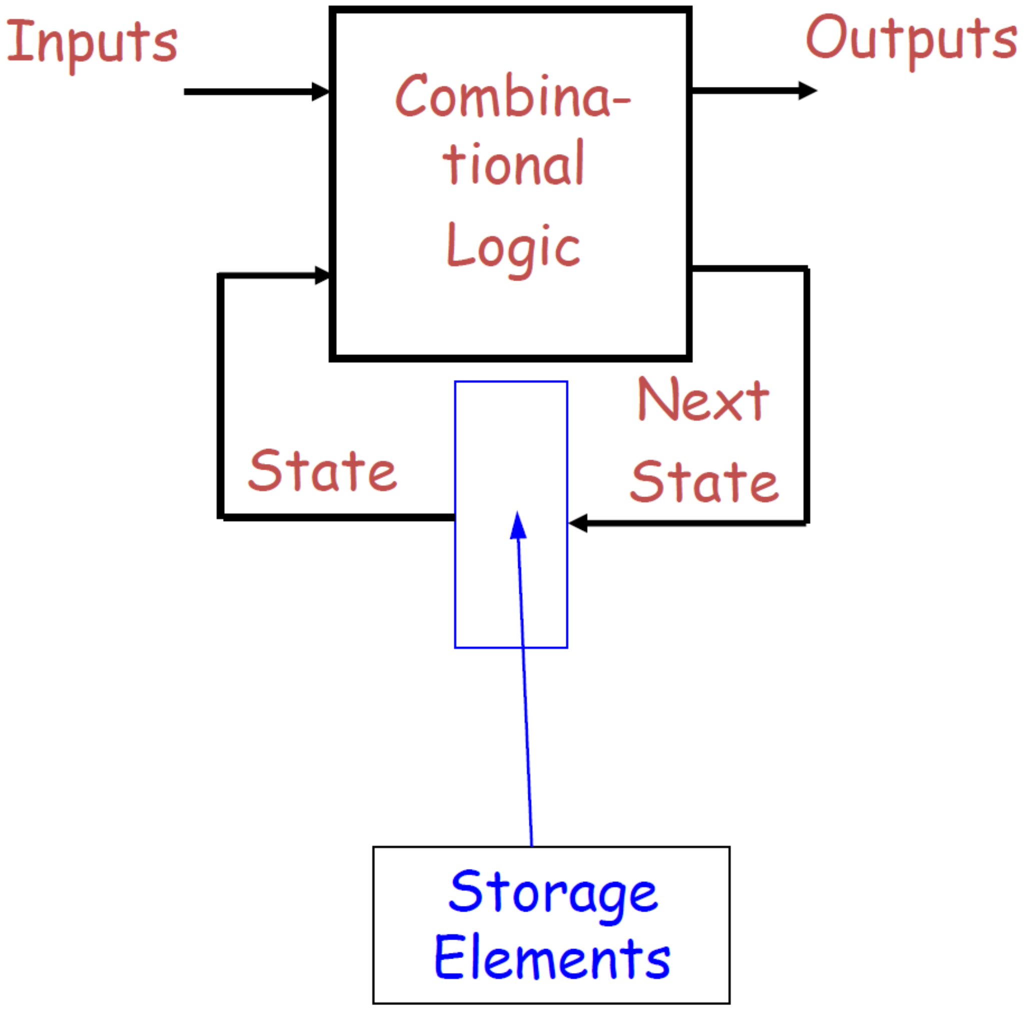
Sequential Adder

- 1-bit memory and two 4-bit memory
    - Only one full-adder!
    - N clocks (four in this case) to get the output
    - **The 1-bit memory defines the circuit state** (0 or 1)



Sequential Circuits

- A sequential circuit contains:
  - Storage elements: **latches** or **flip-flops**
  - Combinational logic that implements a multiple-output switching function:
    - Inputs are signals from the outside
    - Outputs are signals to the outside
    - Other inputs, State, are signals from storage elements
    - The remaining outputs, Next State, are inputs to storage elements



Moore vs Mealy machine

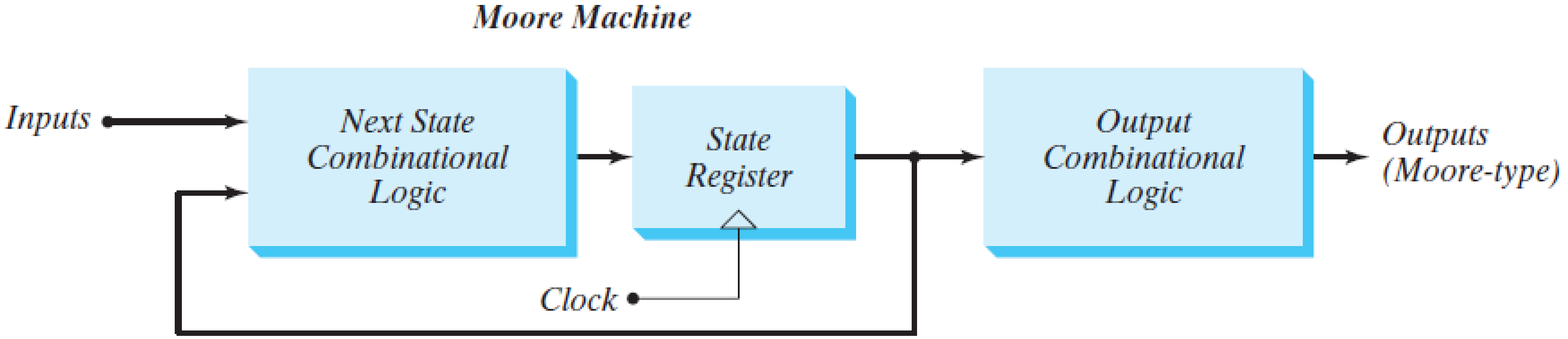
Combinatorial Logic

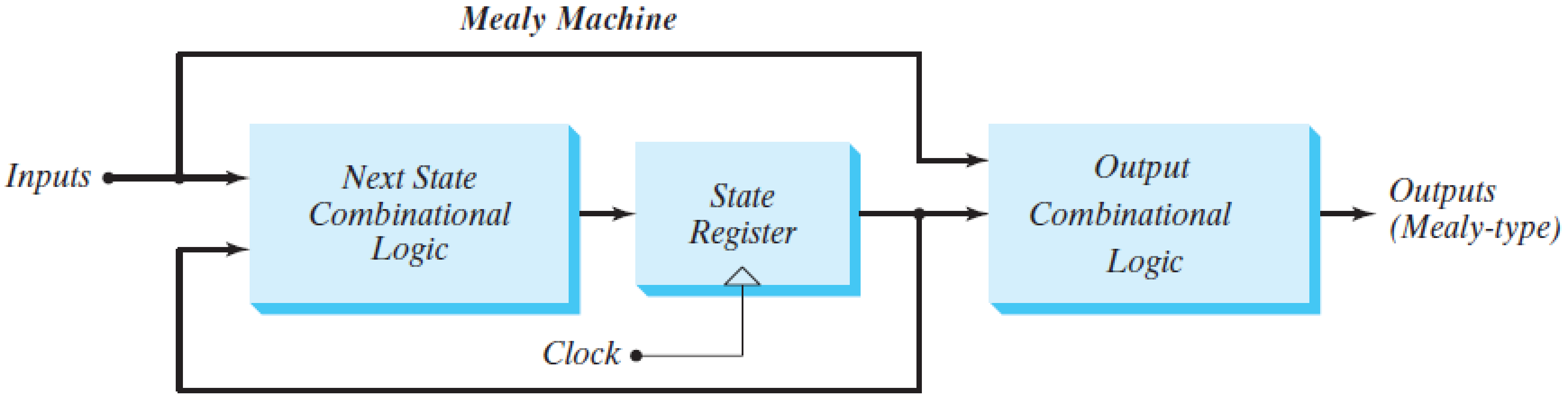
Next state function

Next State = f(Inputs, State)

Output function, two types:

- Moore
  Outputs = h(State)
- Mealy
  Outputs = g(Inputs, State)





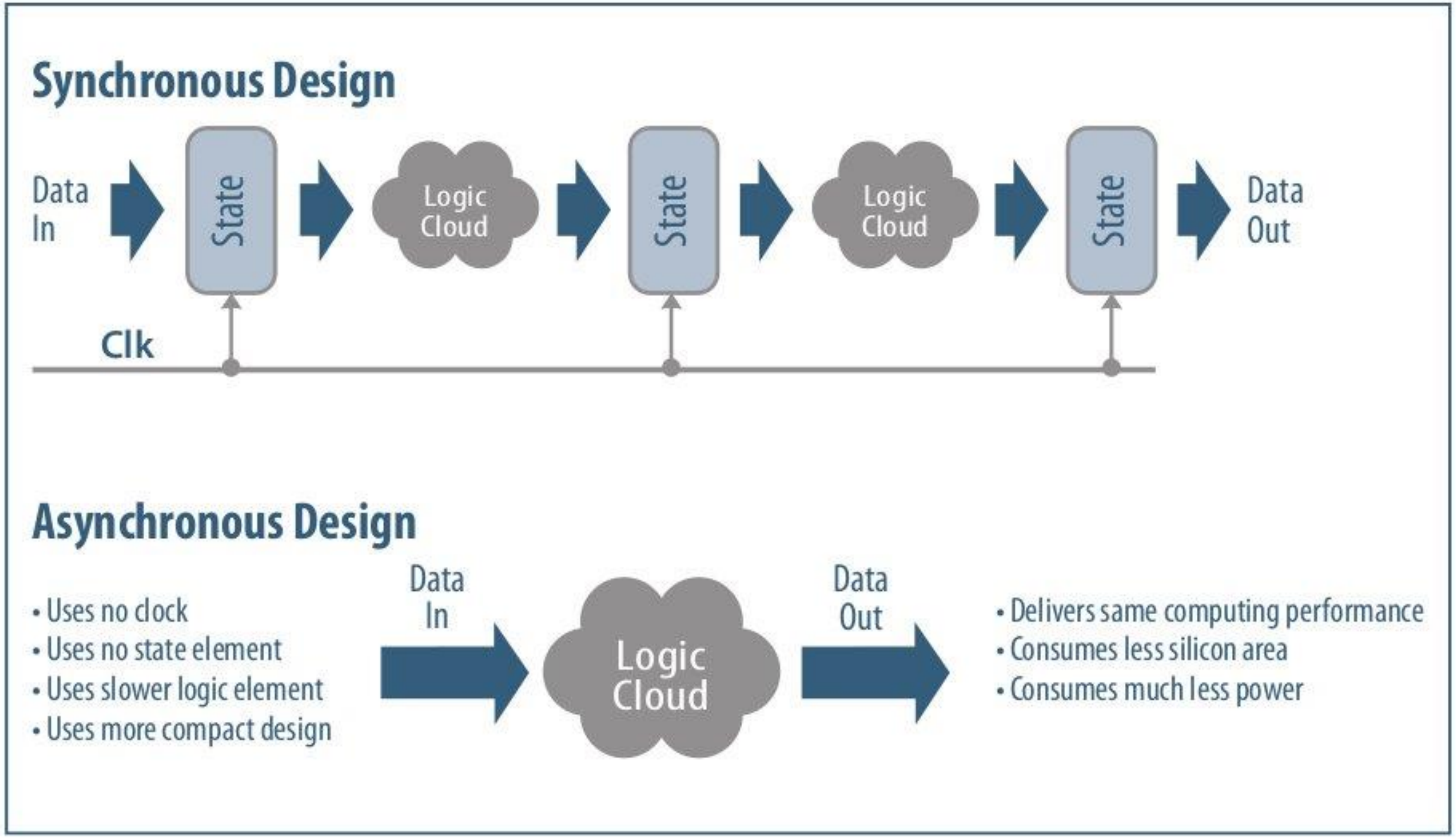
Output function type depends on specification and affects the design significantly

Types of Sequential Circuits

- Depends on the times at which:
  - Storage elements observe their inputs and change their state
- **Synchronous**
  - Behavior is defined from knowledge of signals **at discrete** instances of time
  - Storage elements observe inputs and can change state only in relation to a timing signal (**clock pulses from a clock**)

Synchronous vs Asynchronous Design

Typical Processor Circuity Implementation Comparison
-Design Methodology vs. Clockless/Asynchronous Design Methodology



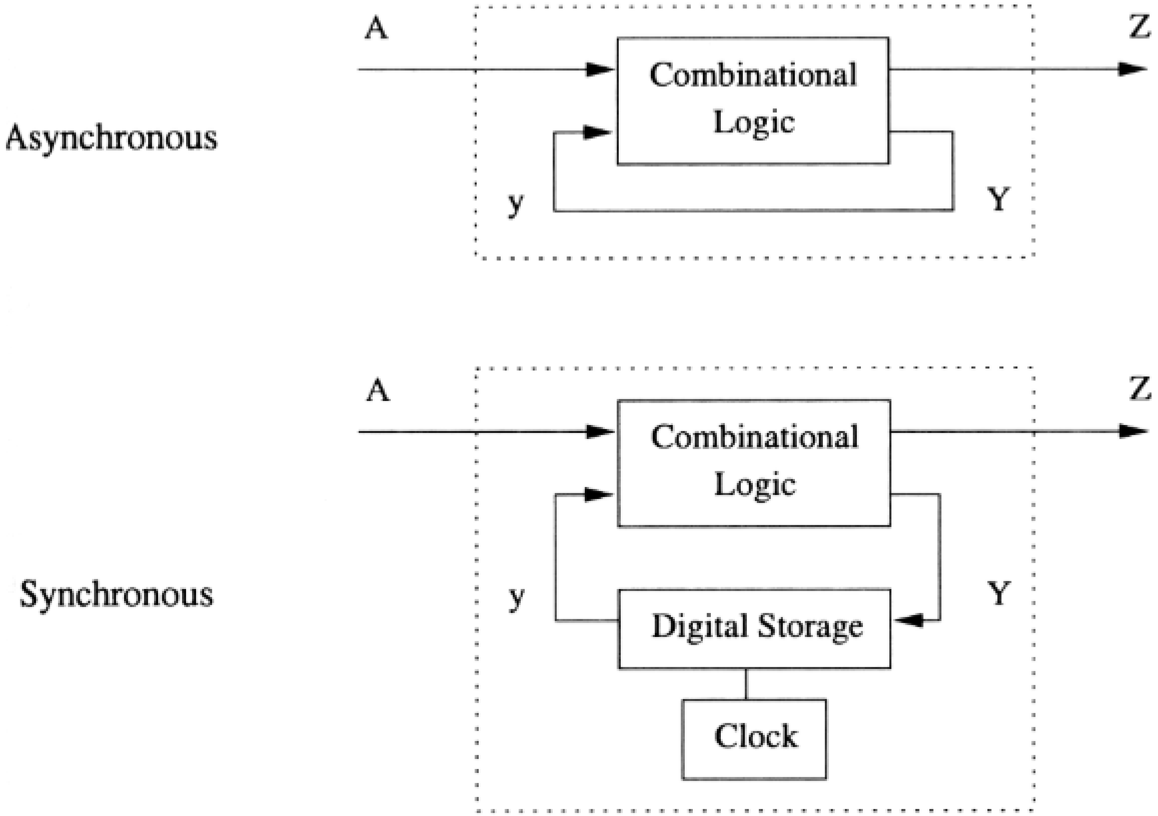
Synchronous Design

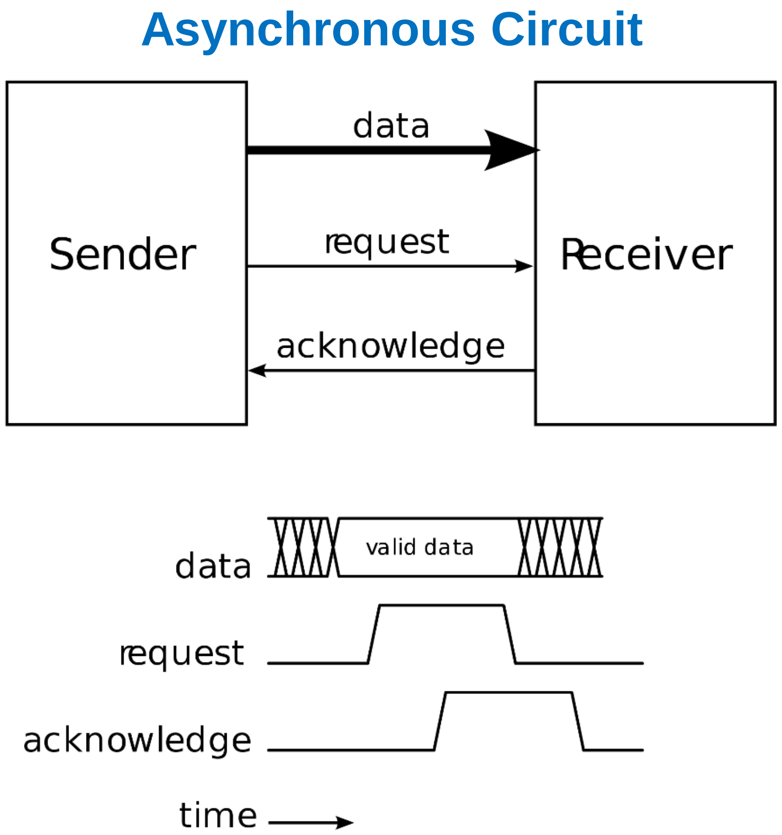
Asynchronous Design

- Uses no clock
- Uses no state element
- Uses slower logic element
- Uses more compact design

- Delivers same computing performance
- Consumes less silicon area
- Consumes much less power

Asynchronous vs Synchronous



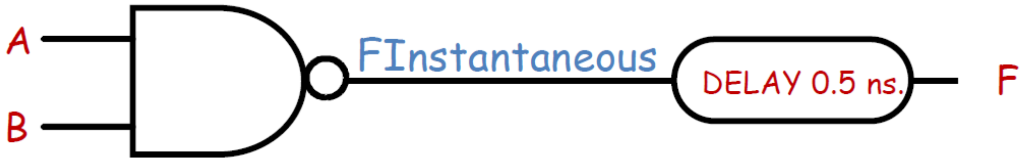


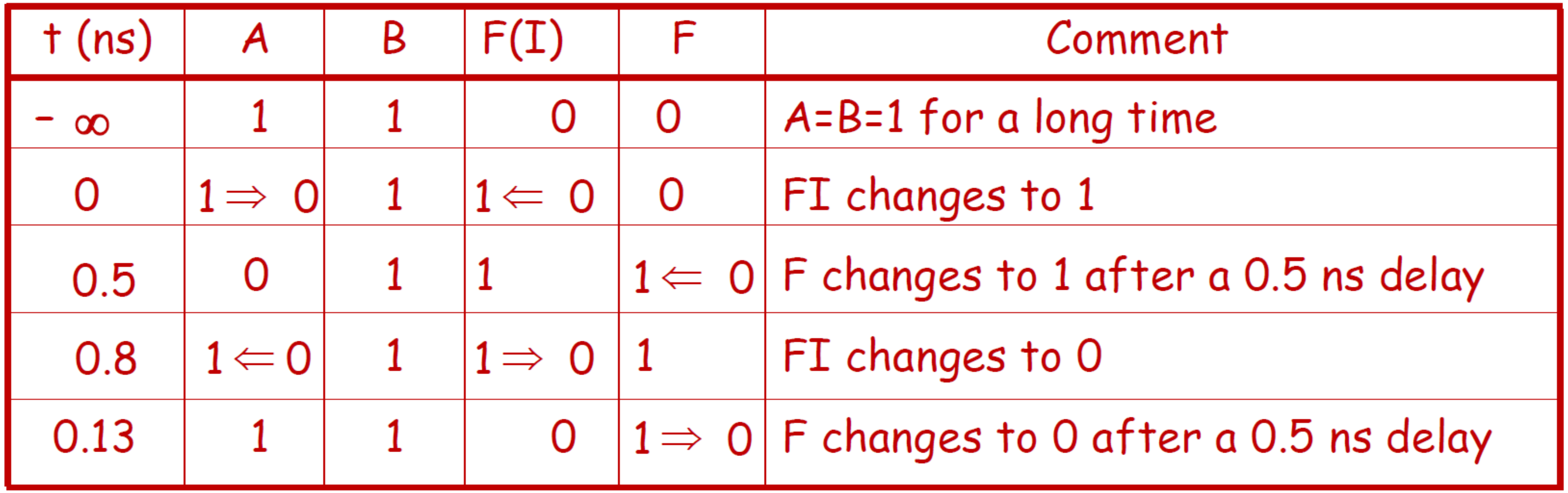
Discrete Event Simulation

- In order to understand the time behavior of a sequential circuit, we use **discrete event simulation**
- Rules:
  - Gates modeled by an ideal (instantaneous) function and a fixed gate delay
  - Any change in input values is evaluated to see if it causes a change in output value
  - Changes in output values are scheduled for the fixed gate delay after the input change
  - At the time for a scheduled output change, the output value is changed along with any inputs it derives

Simulated NAND Gate

- Example: a 2-input NAND gate with a 0.5 ns delay
  - Assume A and B have been 1 for a long time
  - At time t=0, A changes to 0 & at t=0.8ns, back to 1

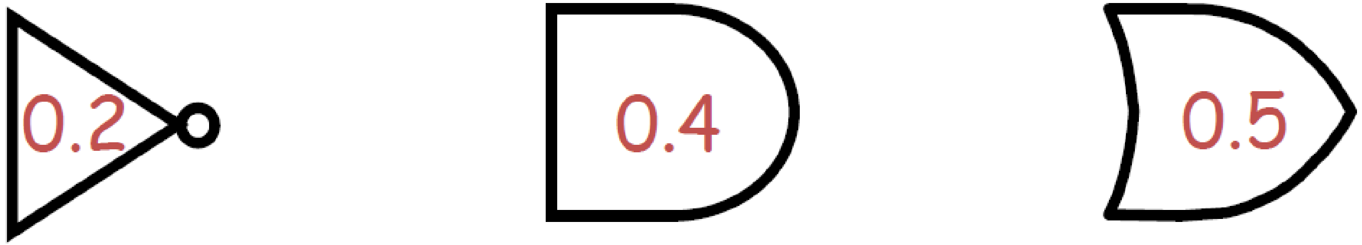




t (ns)	A	B	F(I)	F	Comment
- ∞	1	1	0	0	A=B=1 for a long time
0	1 ⇒ 0	1	1 ⇐ 0	0	FI changes to 1
0.5	0	1	1	1 ⇐ 0	F changes to 1 after a 0.5 ns delay
0.8	1 ⇐ 0	1	1 ⇒ 0	1	FI changes to 0
0.13	1	1	0	1 ⇒ 0	F changes to 0 after a 0.5 ns delay

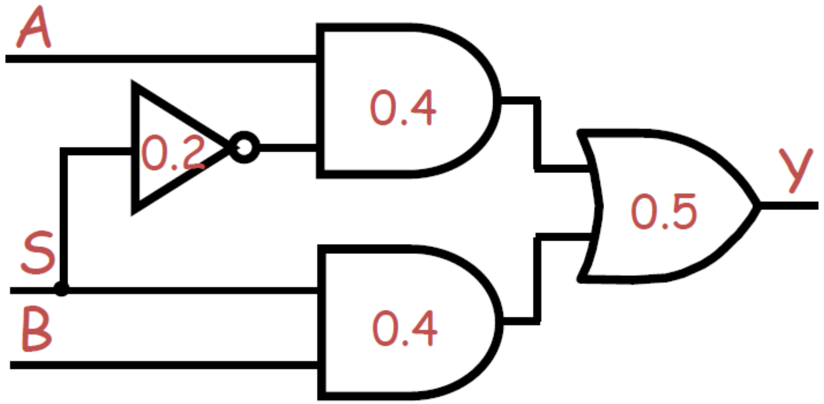
Gate Delay Models

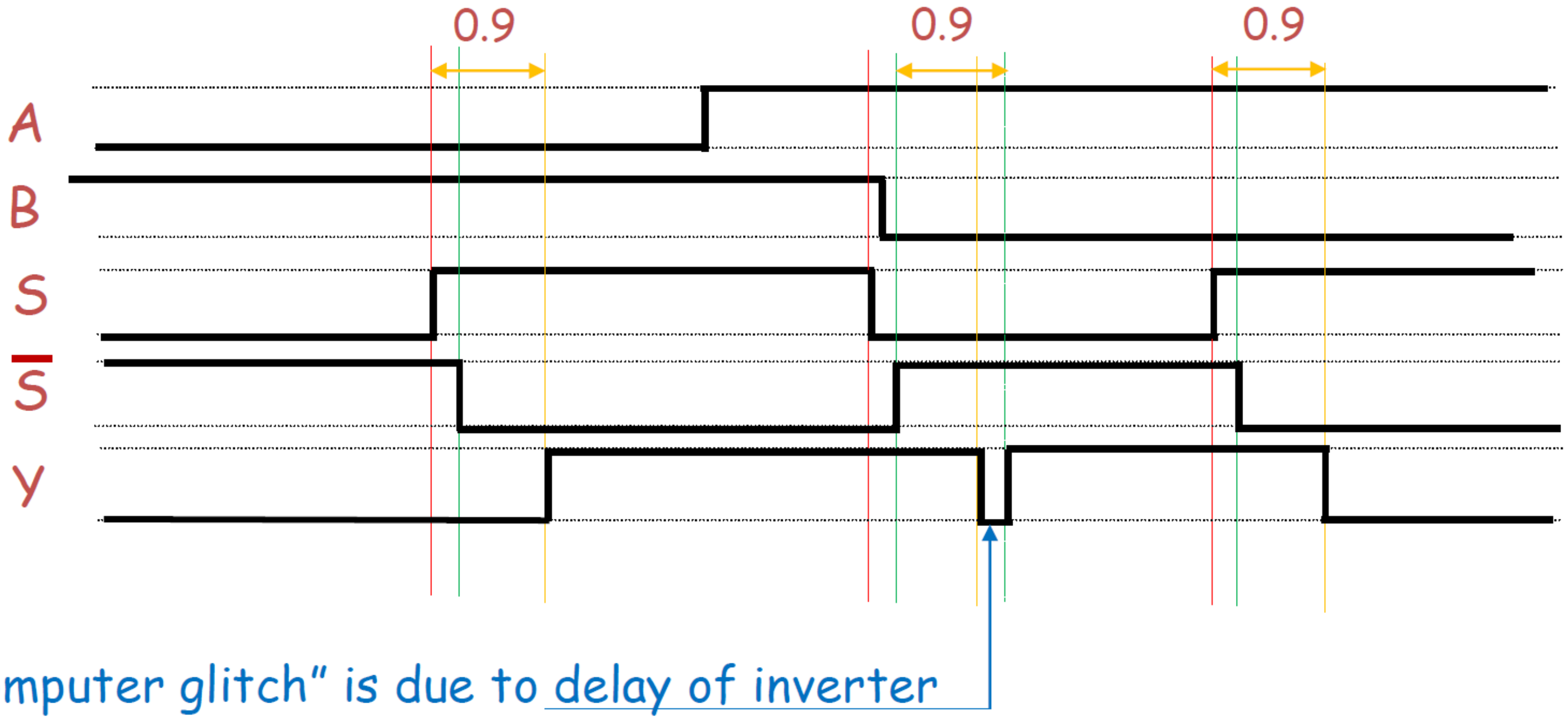
- Suppose gates with delay ‘k’ ns are represented for k = 0.2ns, k = 0.4ns, k = 0.5ns, repectively



Circuit Delay Models

- Consider a simple 2-input multiplexer:
- With function:
  - y = B for S = 1
  - y = A for S = 0





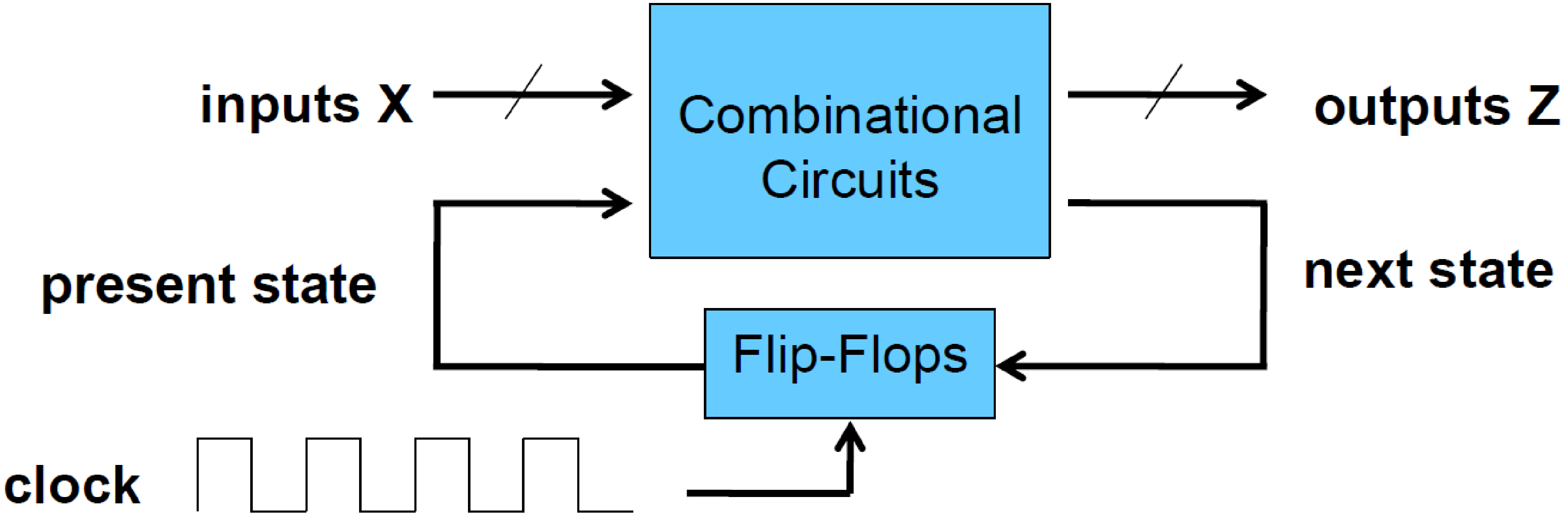
"computer glitch" is due to delay of inverter

Storage Elements (Memory)

- A storage element can maintain a binary state (0,1) indefinitely, until it is directed by an input signal to swith state
- Main difference btw storage elements:
  - # of inputs they have
  - How the inputs affect the binary state
- Two main types:
  - Latches (level-sensitive): useful in asynchronous circuits
  - Flip-flops (edge-sensitive): build w/ latches – synchronous circuits

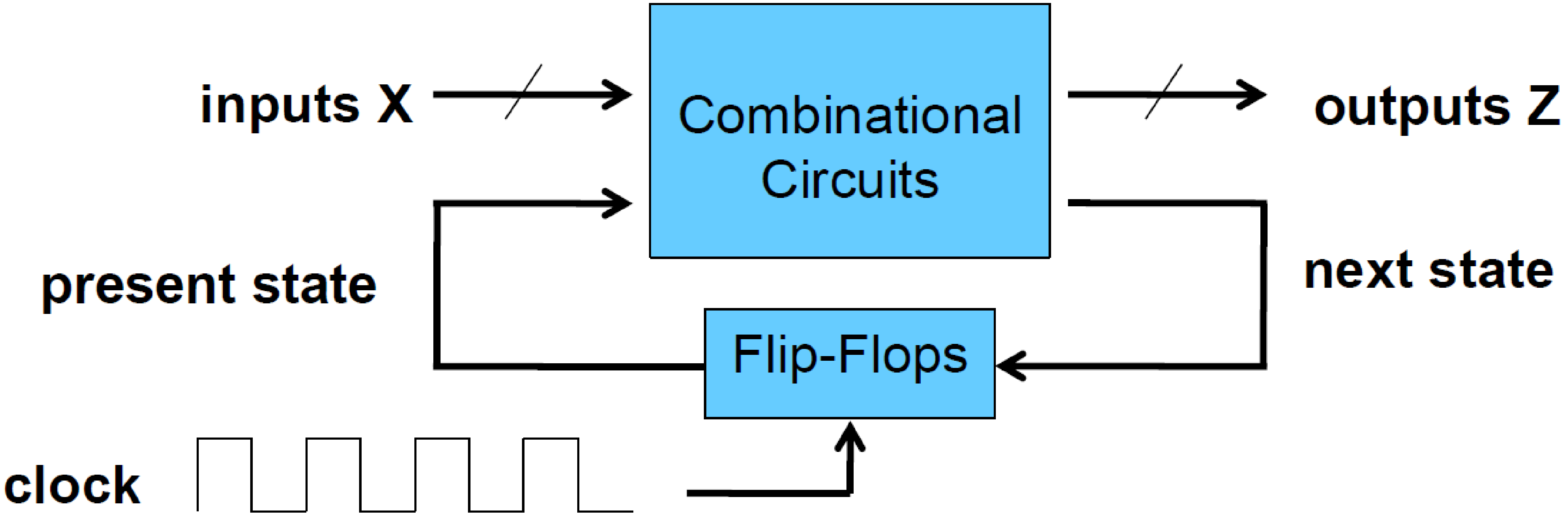
Synchronous Sequential Circuits

- Synchronous circuits employs a synchronizing signal called **clock**
- A clock determines **when** computational activities occur
- Other signals determine **what** changes will occur

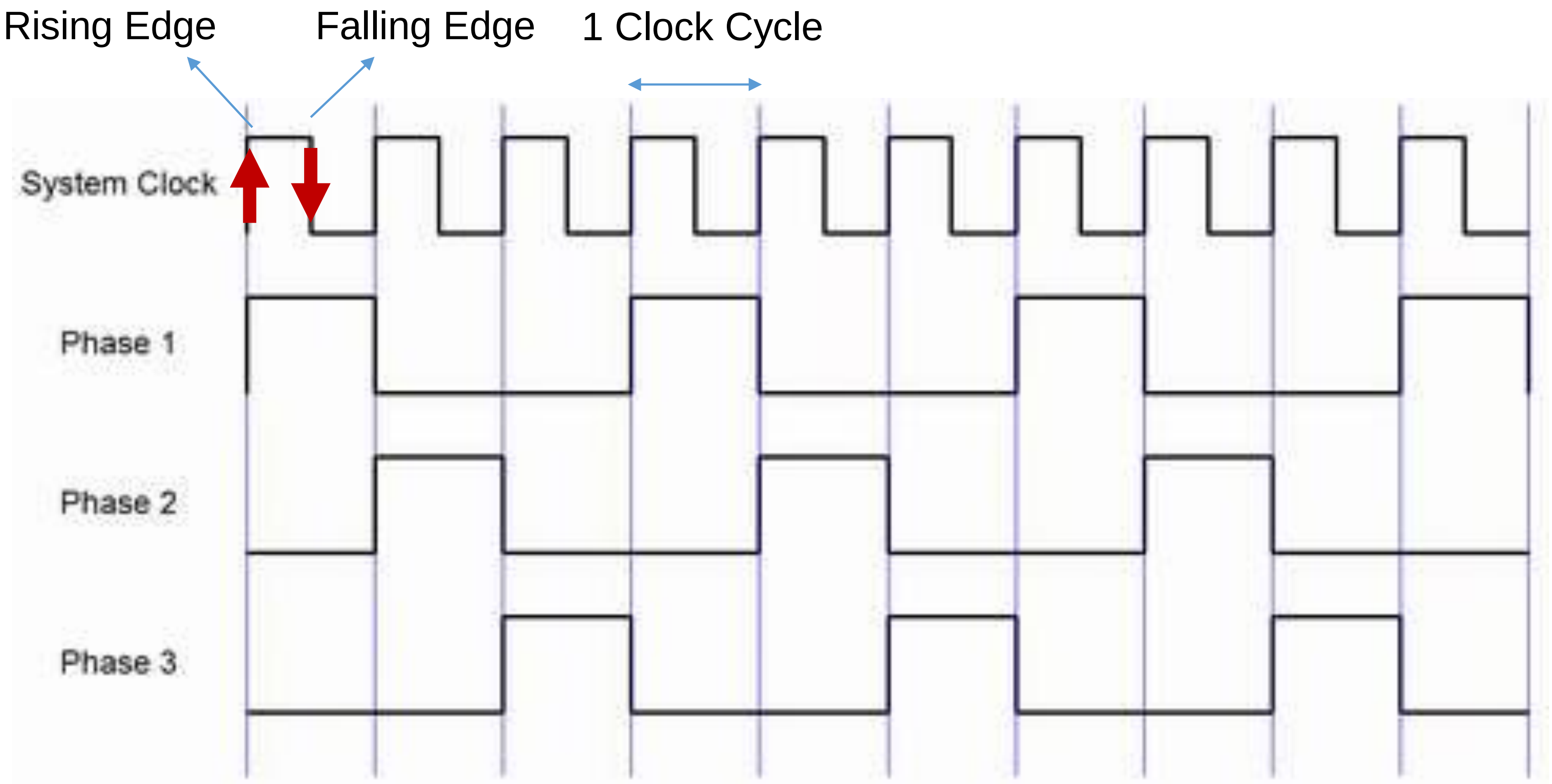


Synchronous Sequential Circuits

- The storage elements used in clocked sequential circuits are called **flip-flops (FFs)**
  - Each FF can store 1bit of information
  - A circuit may use many FFs: they define the circuit state



System Clock

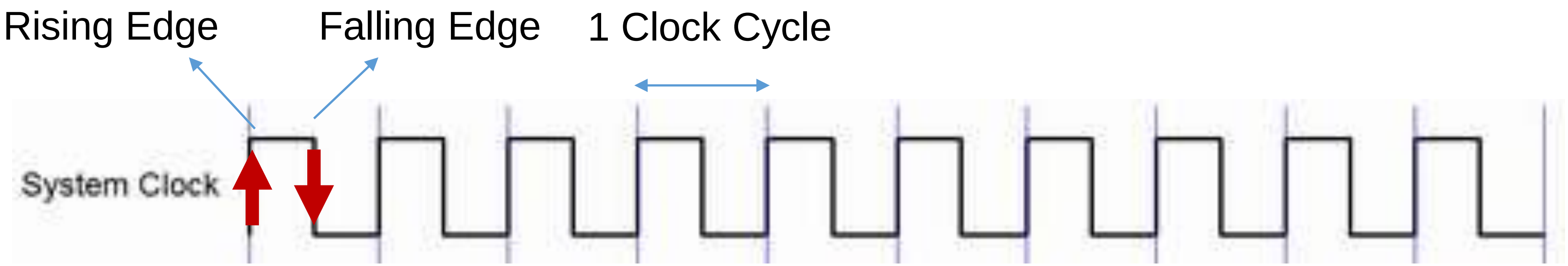


Rising Edge

Falling Edge

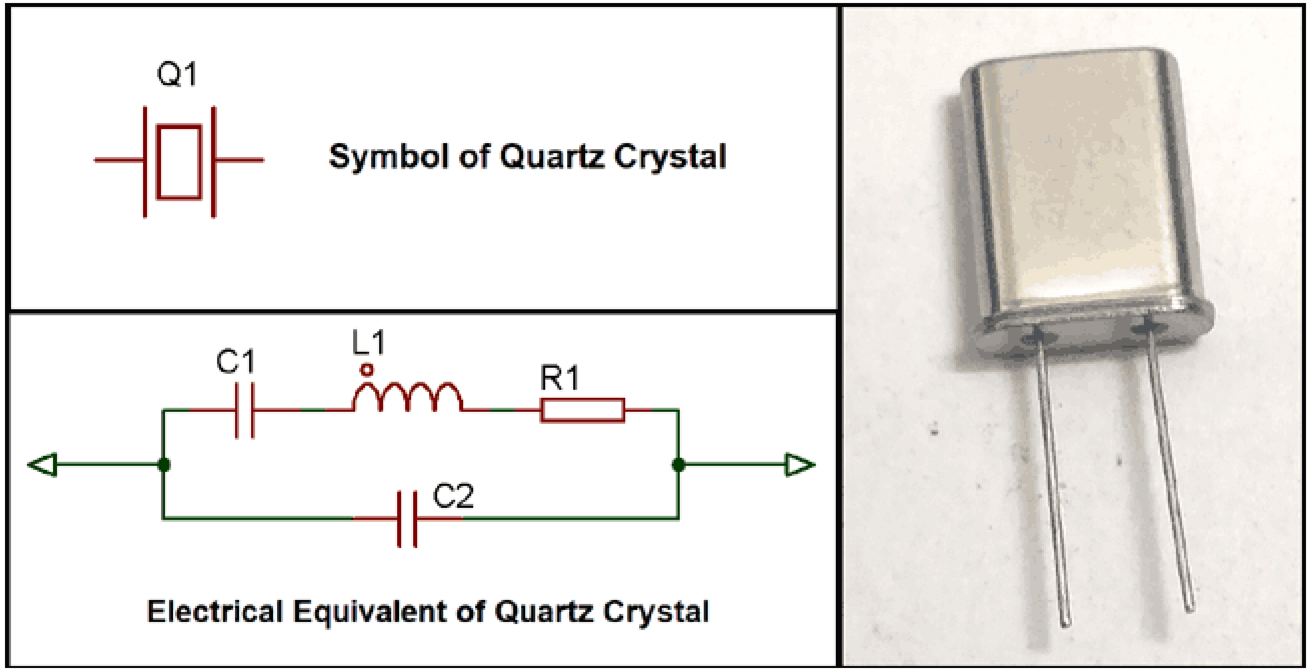
1 Clock Cycle

System Clock

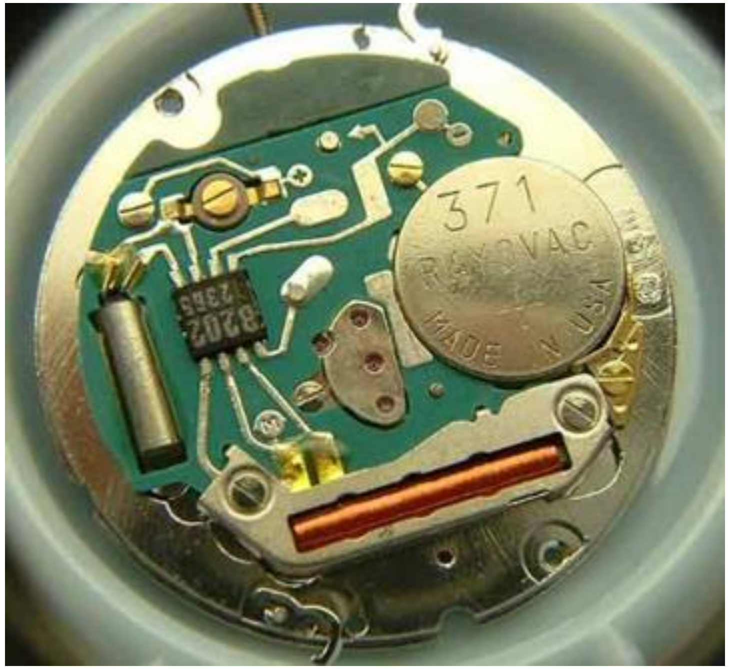


- Clock period (normally measured in micro- or nanoseconds) is the time between successive transitions in the same direction.
- Clock frequency (measured in MHz or GHz) is the reciprocal of clock period
- Clock width is the time interval during which clock is equal to 1
- Duty cycle is the ratio of the clock width and clock period
- Clock signal is ACTIVE HIGH if the changes occur at the rising edge or during the clock width. Otherwise, it is active low.

System Clock



Quartz frequency = 32,768 Hz (=2^{15})
> 15bit binary counter will count 1 sec.



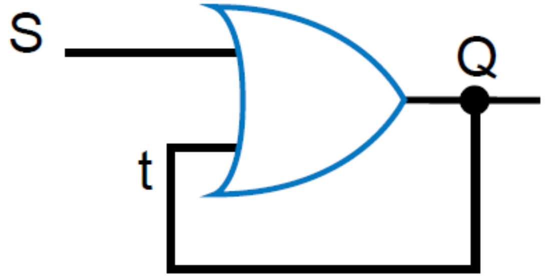
Latches

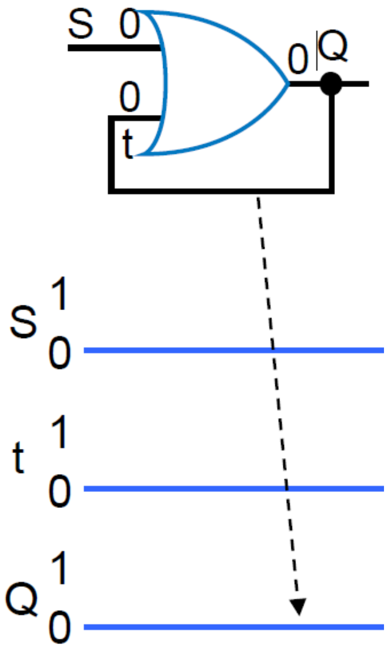
What are Latches?

- A latch is a binary storage element
  - Can store 0 or 1
- The most basic memory component
- Easy to build
  - Build with primitive gates (NORs, NANDs, INV)

First Attempt for Bit Storage

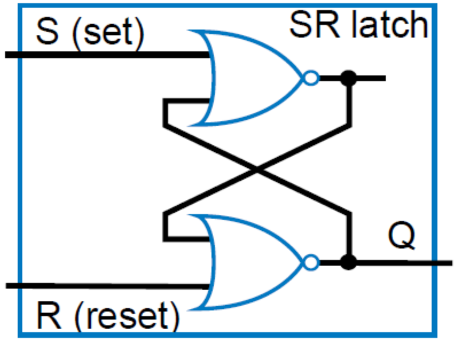
- We need some sort of feedback
  - Does a circuit on the right do what we want?

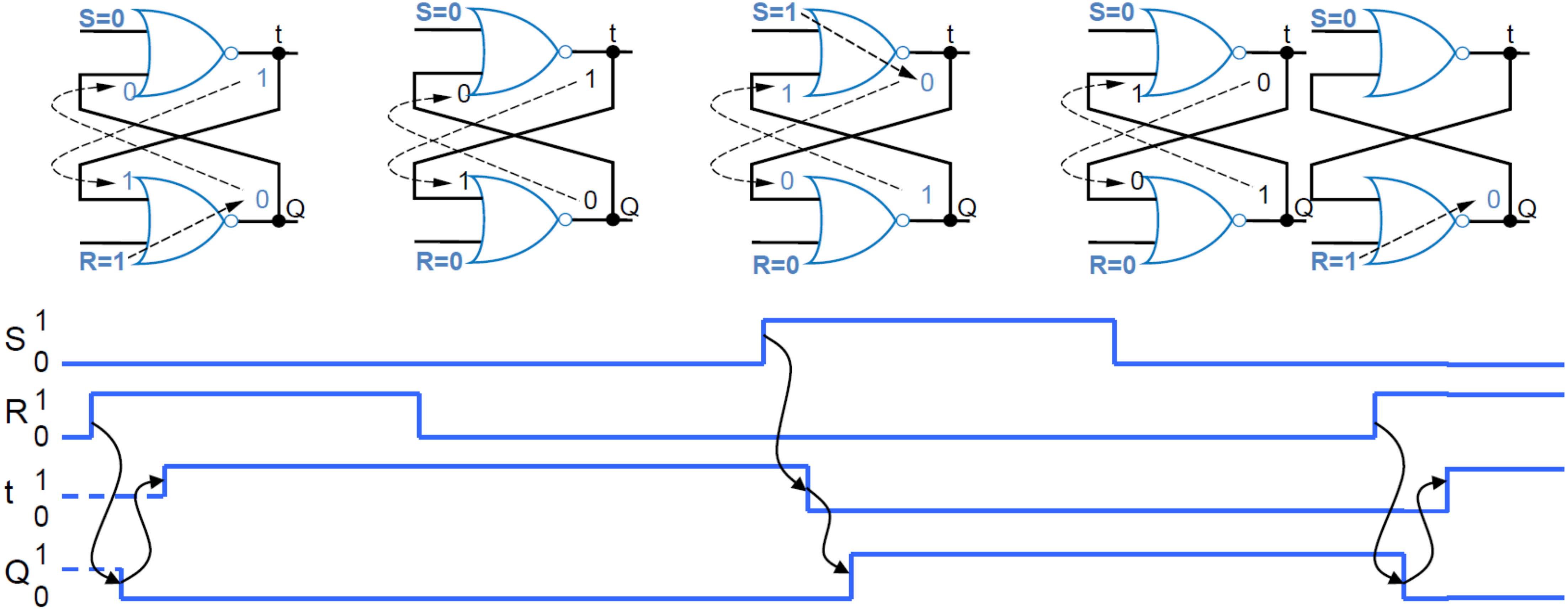




Bit Storage Using SR Latch

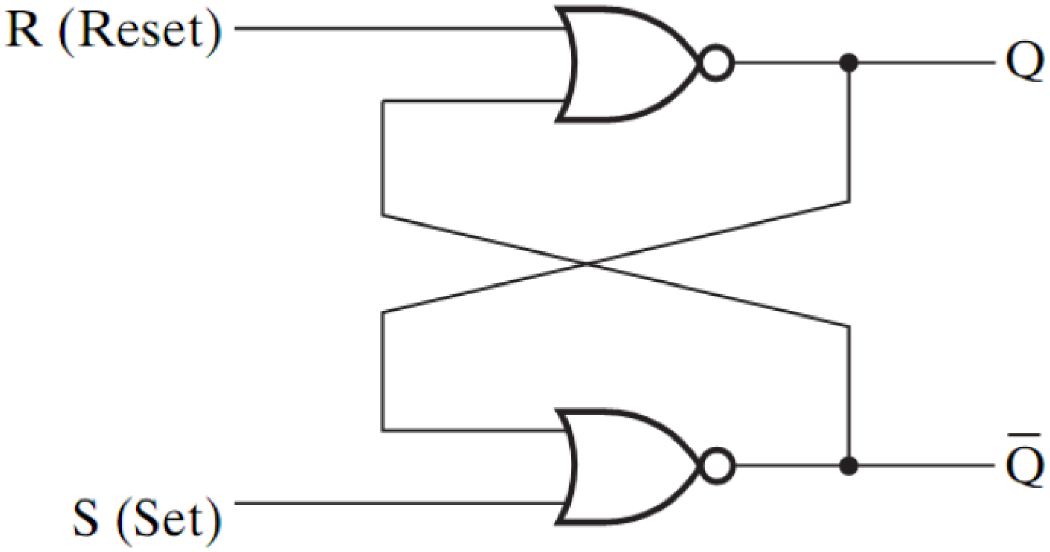
- Does the circuit to the right, with cross-coupled NOR gates, do what we want?





SR Latch

- **Two states**: ‘set’ (Q = 1) and ‘reset’ (Q = 0)
  - When S = R = 0, Q remains the same (**S = R = 1 is not allowed!**)
  - State of the circuit depends not only on the current inputs, but also on the recent history of the inputs



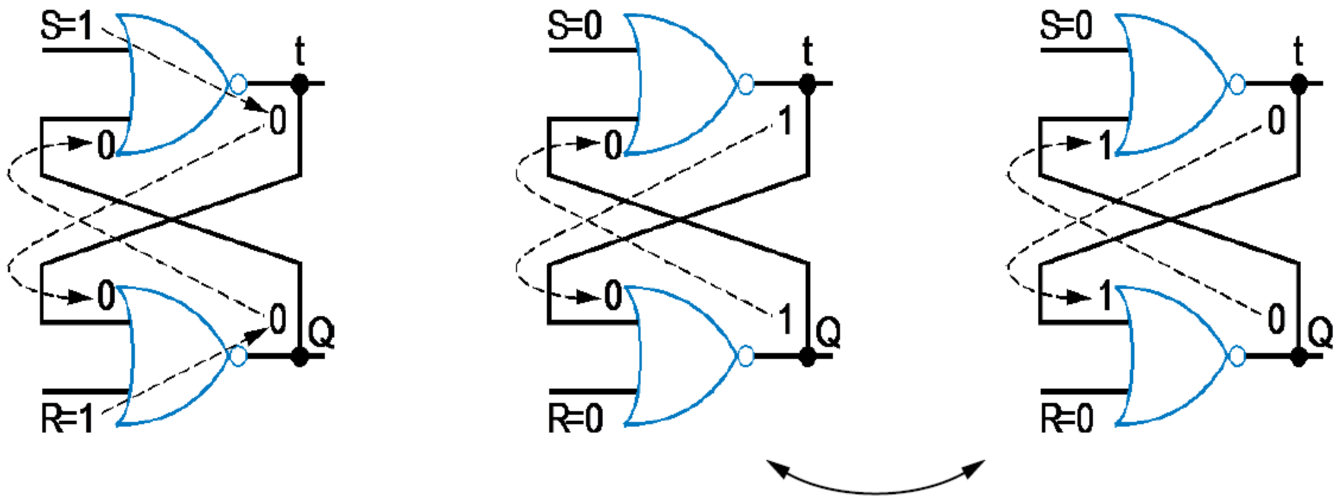
Logic diagram

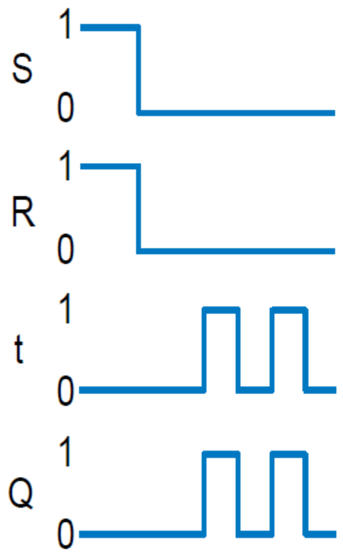
S	R	Q	$\bar{Q}$	
1	0	1	0	Set state
0	0	1	0	
0	1	0	1	Reset state
0	0	0	1	
1	1	0	0	Undefined

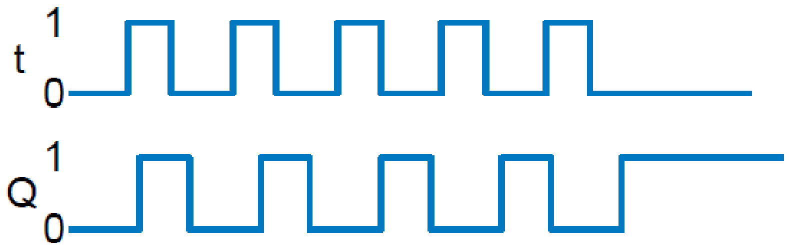
Function table

Problem with SR Latch

- If S = R = 1 simultaneously, we don’t know what value Q will take



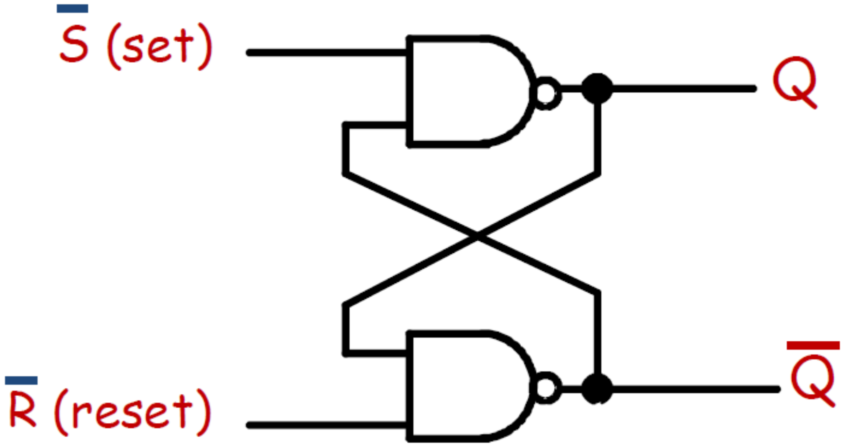




Q may oscillate. Then, because one path will be slightly longer than the other, Q will eventually settle to 1 or 0 – but we don't know which.

Basic (NAND) S’R’ Latch

- “Cross-coupling” two NAND gates gives the S`-R` latch



- Which has the time sequence behavior of:

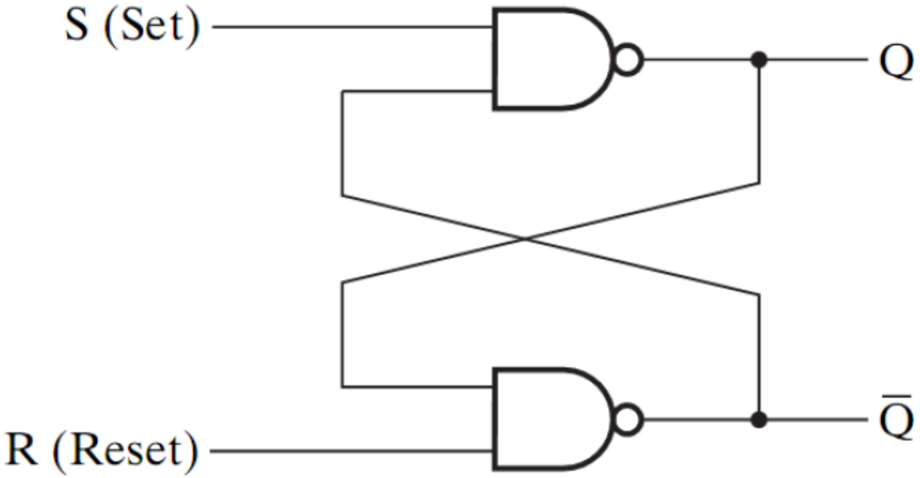
Time

$\overline{R}$	$\overline{S}$	Q	$\overline{Q}$	Comment
1	1	?	?	Stored state unknown: 01/10
1	0	1	0	“Set” Q to 1
1	1	1	0	Now Q “remembers” 1
0	1	0	1	“Reset” Q to 0
1	1	0	1	Now Q “remembers” 0
0	0	1	1	Both go high Unstable!

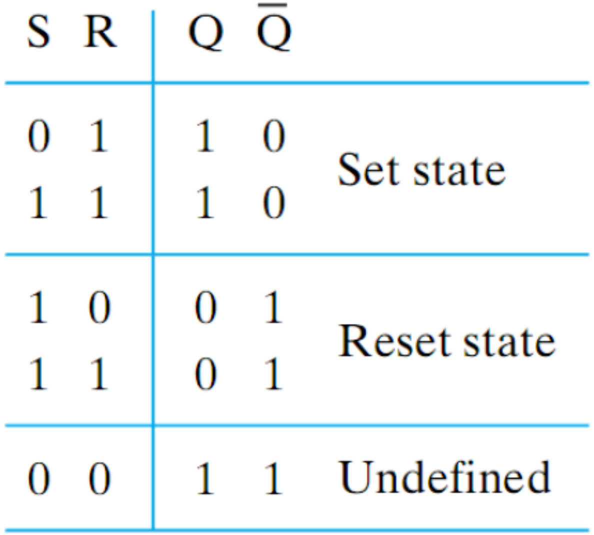
forbidden

S'R' Latch

- Similar to SR latch (complemented)
  - Two states: 'set' (Q = 0) and 'reset' (Q = 1)
  - When S = R = 1, Q remains the same
  - **S = R = 0 is not allowed!**



Logic diagram

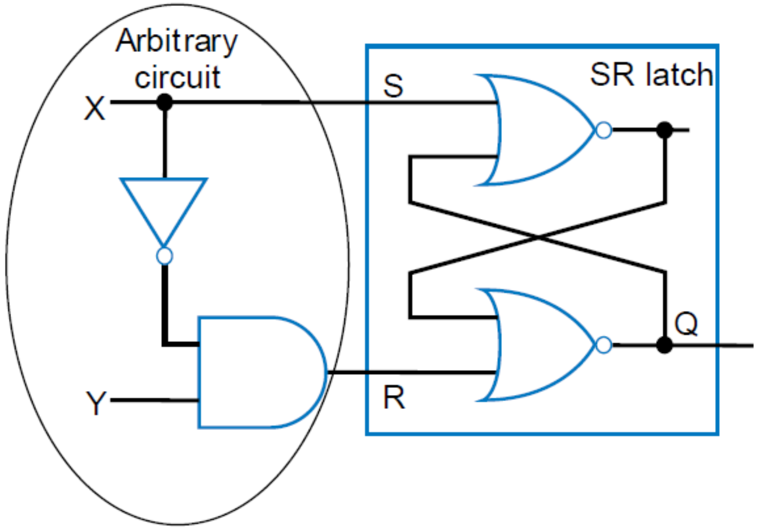


S	R	Q	$\bar{Q}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

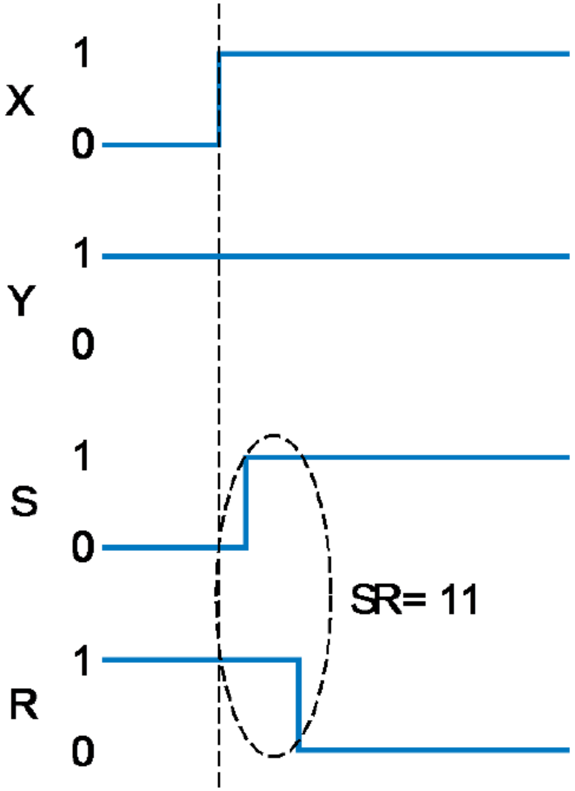
Function table

More Problems with S'R' Latch

- The problem occurs even if SR inputs come from a circuit that supposedly never set S = 1 and R = 1 at the same time
  - But it may happen, due to different delays of different paths

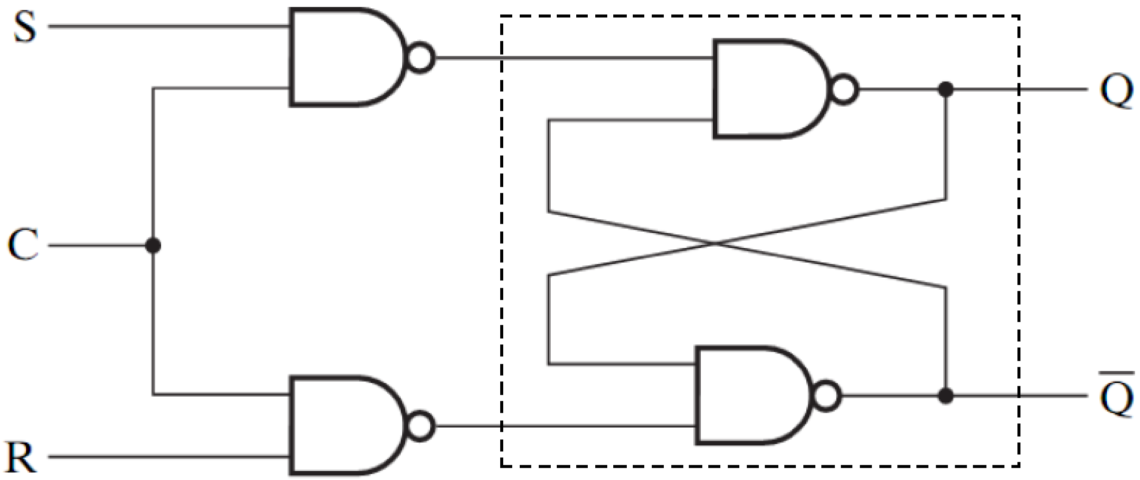


The longer path from X to R than to S causes SR=11 for short time - could be long enough to cause oscillation

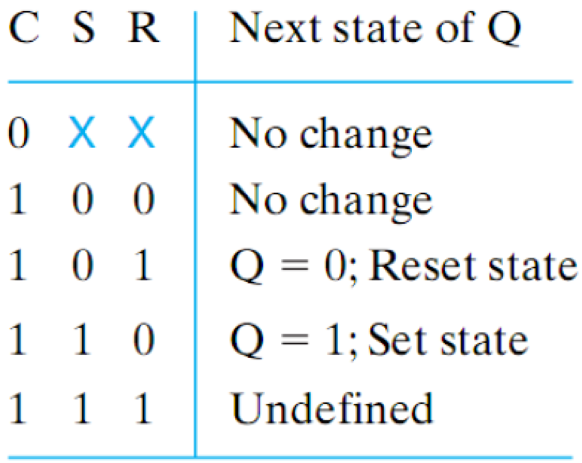


S’R’ Latch with Clock

- An SR latch can be modified to control when it changes an additional input signal ‘Clock (C)’
  - When C = 0, the S and R inputs have no effect on the latch
  - When C = 1, the inputs affect the state of the latch and possibly the output



Logic diagram

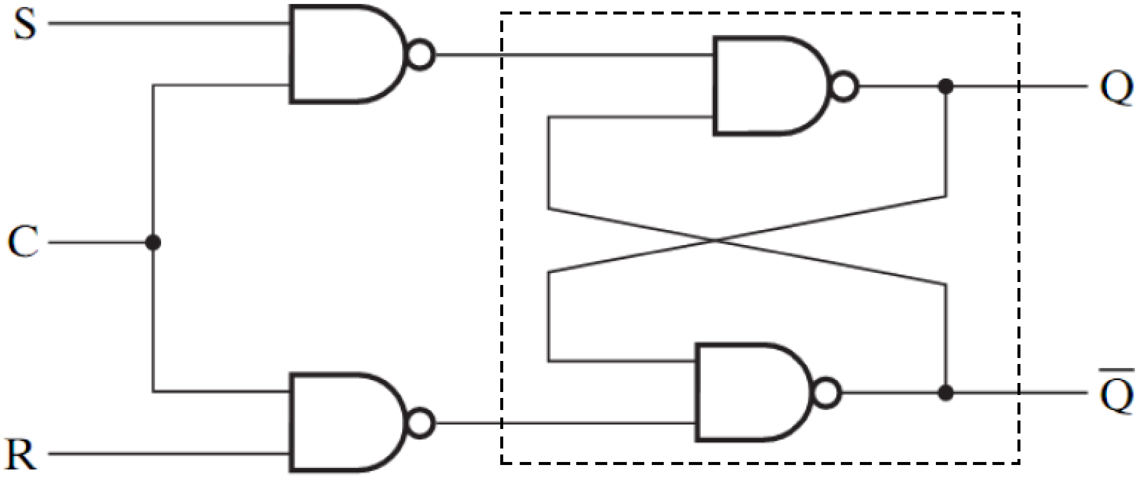


C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

Function table

Dealing with the Undefined State (S'R' Latch)

- How can we eliminate the undefined state?



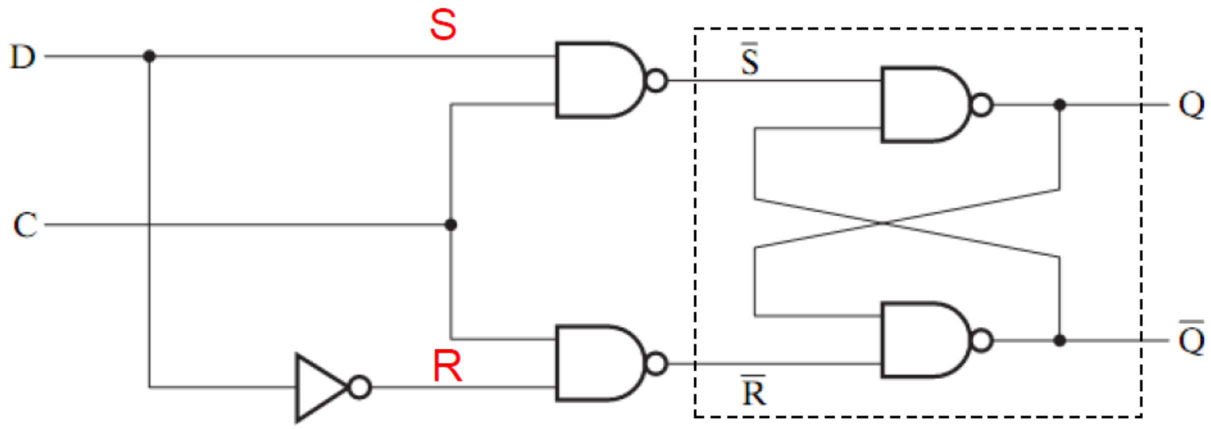
Logic diagram

C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; Set state
1	1	1	Undefined

forbidden

Function table

D Latch



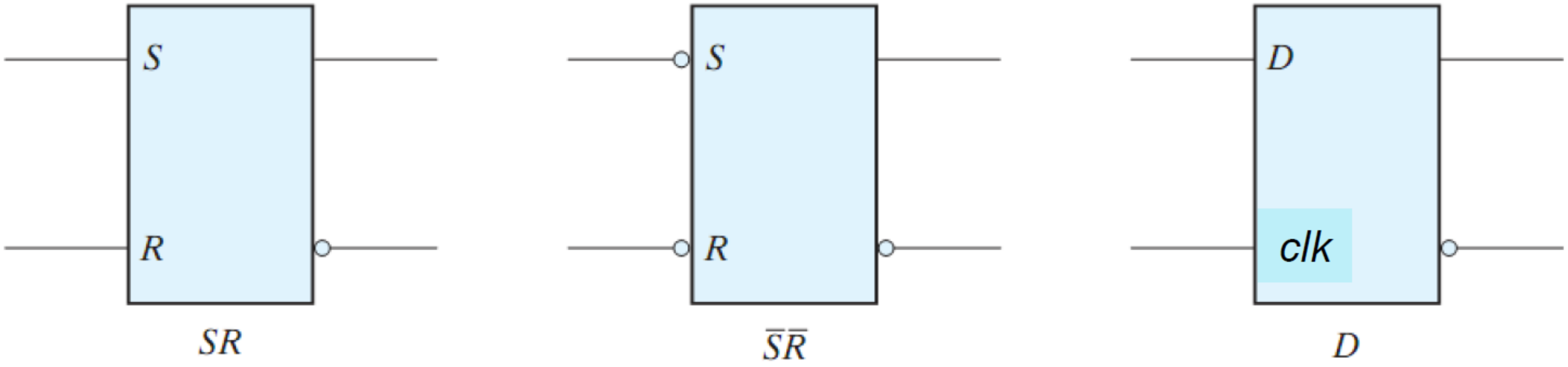
Logic diagram

C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

Function table

- Ensure S and R are never equal to 1 at the same time
  - **Add inverter**
- Only one input (D)
  - D stands for ‘data’
- Output follows input when C = 1
  - Transparent
- When C = 0, Q remains the same

Graphical Symbols for Latches

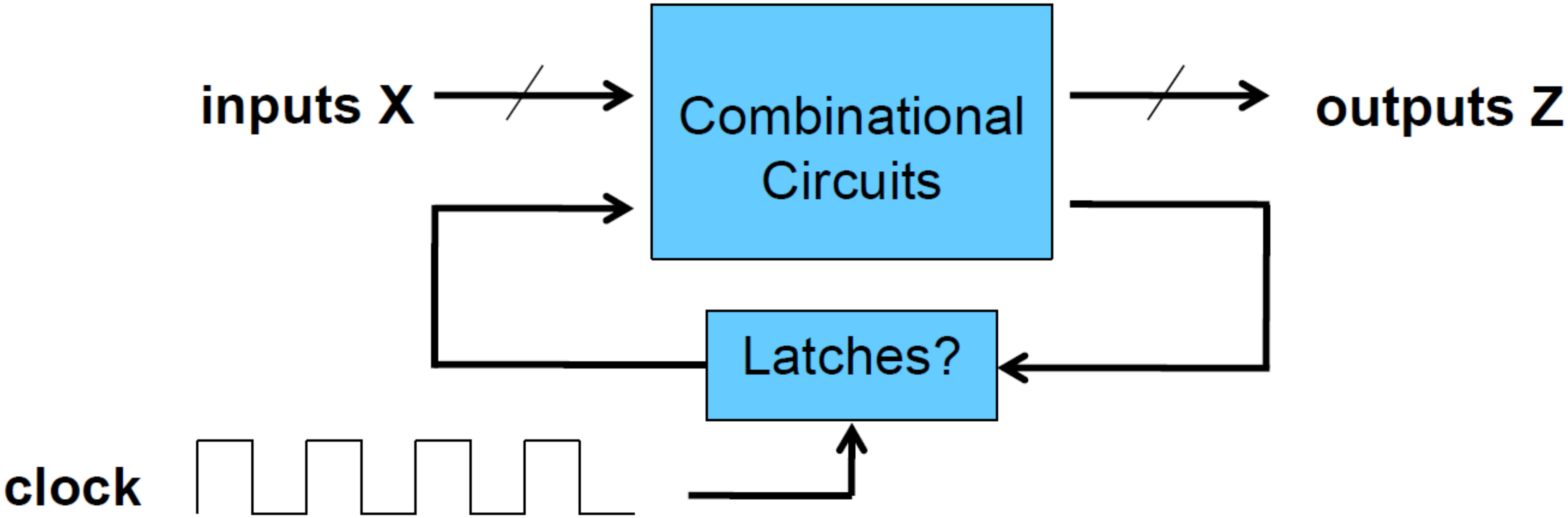


- A latch is designated by a rectangular block with inputs on the left and outputs on the right
- For S`R` latch (w/ NANDs), bubbles are added to the input

Flip-Flops

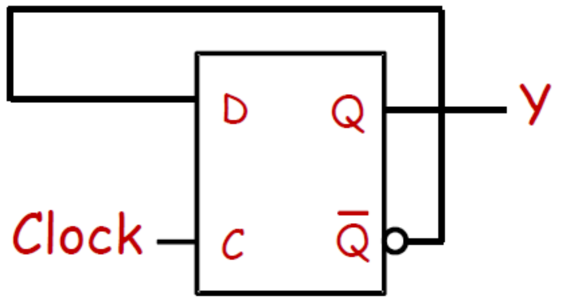
Issues with Latches

- What happens if Clock = 1?
  - A latch is **transparent**: state keeps changing as long as clock remains active
  - Due to this uncertainty, latches cannot be **reliably** used as storage elements

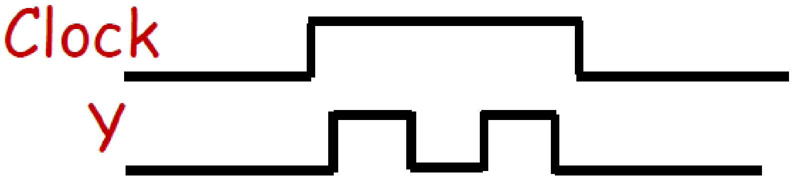


Timing Problem in Latches

- Consider the following circuit



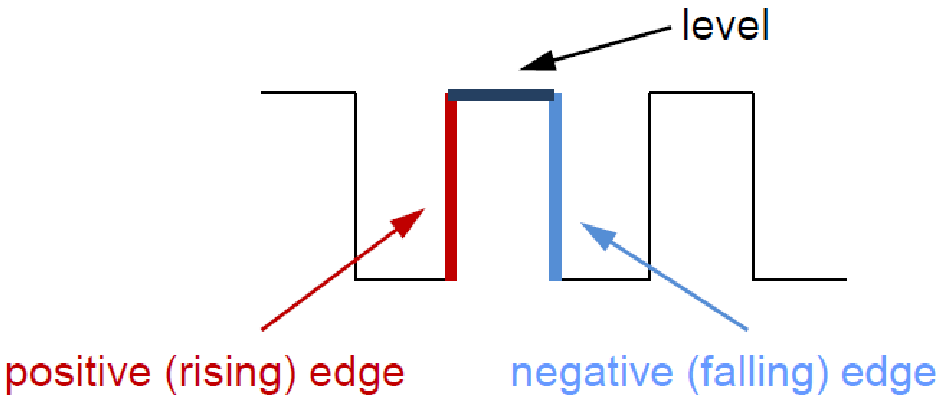
- Suppose that initially y = 0



✓ As long as C = 1, the value of y continues to change!
✓ The changes are based on the delay present on the loop
✓ This behavior is clearly unacceptable
✓ **Desired behavior**: y changes **only once** per clock pulse

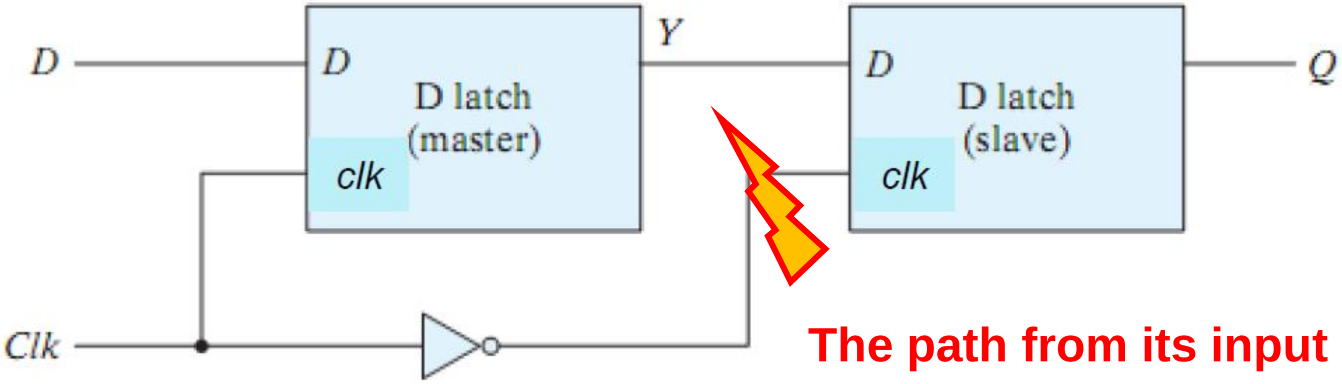
Flip-Flops (FFs)

- A flip-flop is an 1bit memory similar to latches
- Solves the issue of latch transparency
  - Note that latches are level-sensitive
    - Active when the clock = 1 (whole duration)
- FFs are edge-triggered or edge-sensitive memory
  - Active only at transitions, i.e. either from 0 → 1 or 1 → 0



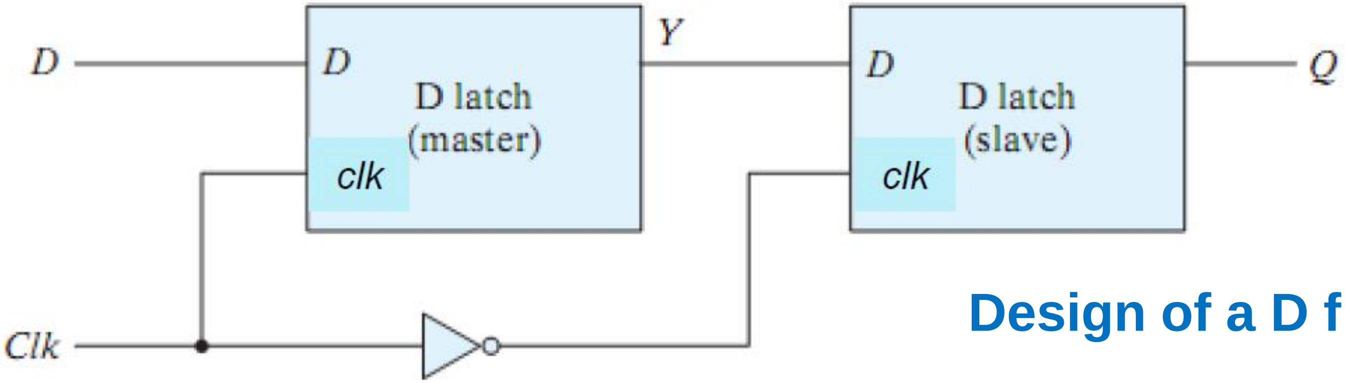
Design of a Flip-Flop

- FF can be built using two latches in a **master-slave** configuration
  - A master latch receives external inputs
  - A slave latch receives inputs from the master latch
- Depending on the clock signal, only one latch is active
  - If clk = 1, the master latch is enables and inputs are latched
  - If clk = 0, the master is disabled and the slave is activated to generate outputs



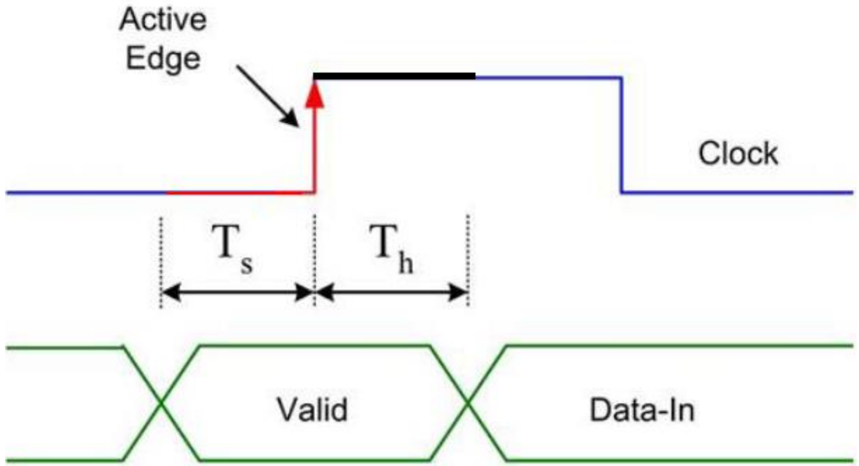
The path from its input to its outputs is broken

Property of a Flip-Flop



Design of a D flip-flop

- Important timing considerations
  - **Setup Time (T_s):** The minimum time during which D input must be maintained before the clock transition occurs.
  - **Hold Time (T_h):** The minimum time during which D input must not be changed after the clock transition occurs.

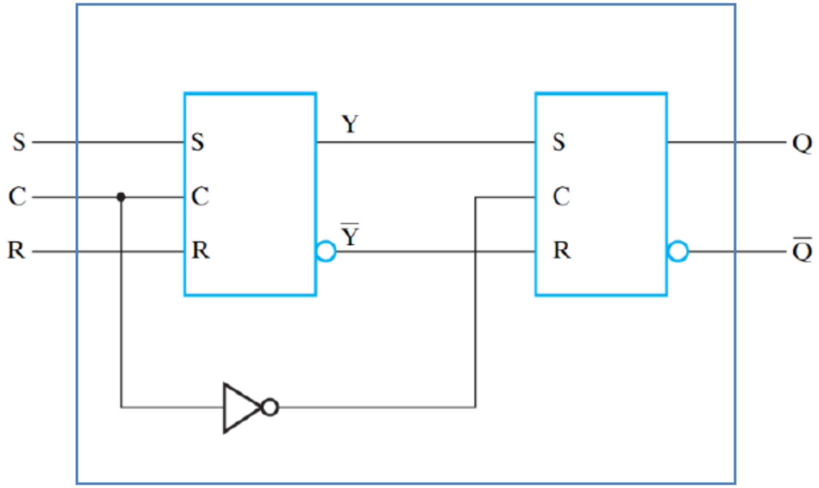


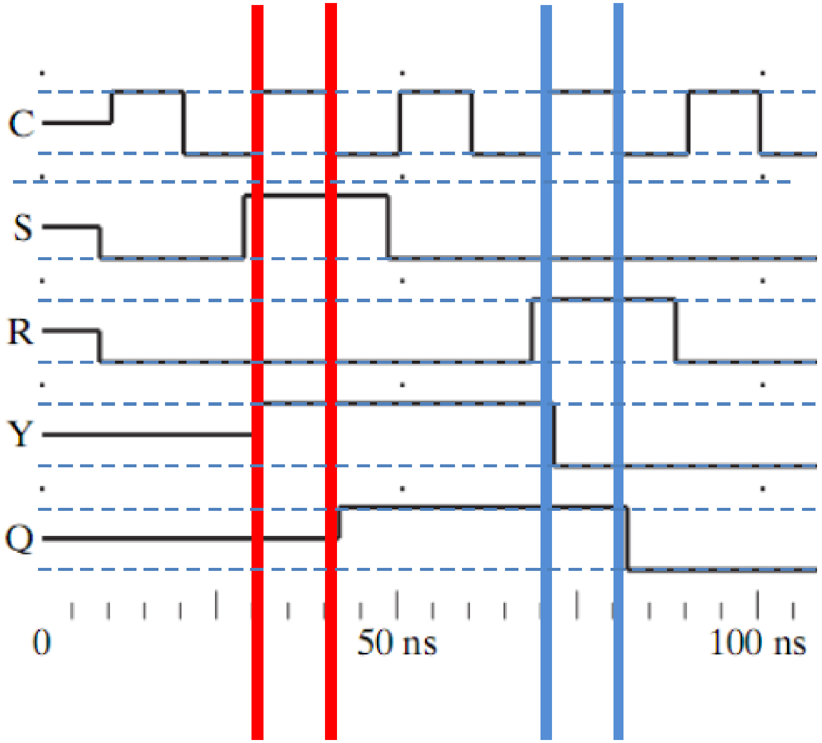
SR Flip-Flop

- Built using two SR latches (master and slave)

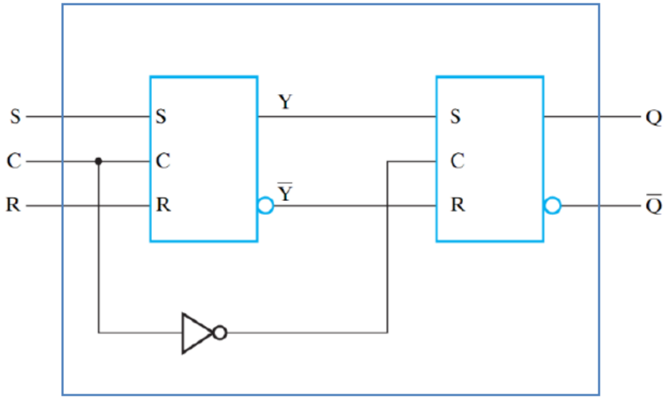
- Data is entered on the rising edge of the clock pulse
  - But, the output does not reflect the change until the falling edge

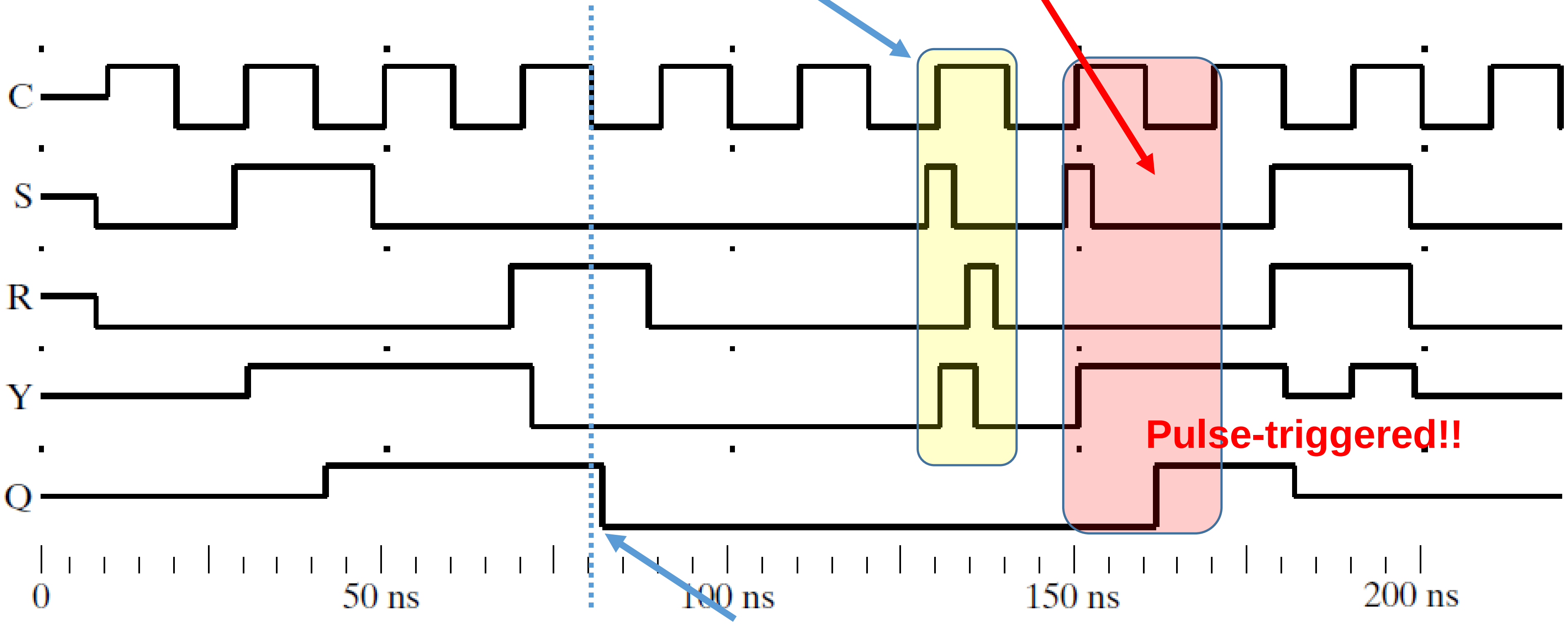
- Q is sampled at the falling edge





More on Simulation of SR Flip-Flops





1s catching behavior

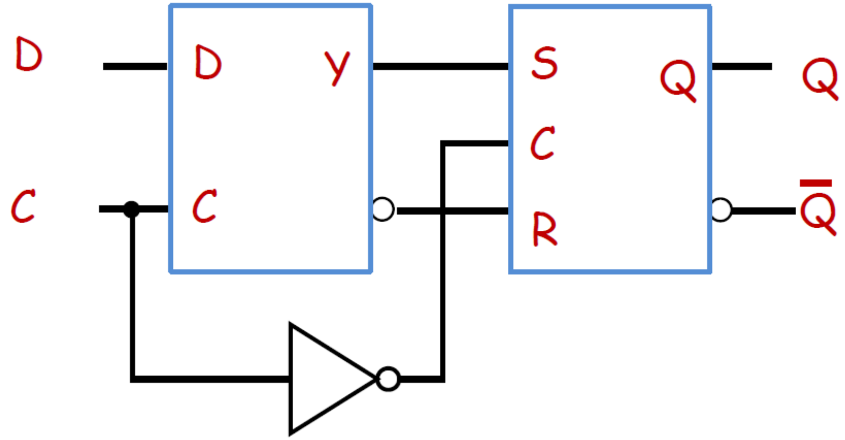
Q should be kept 0
(as S = R = 0)

Pulse-triggered!!

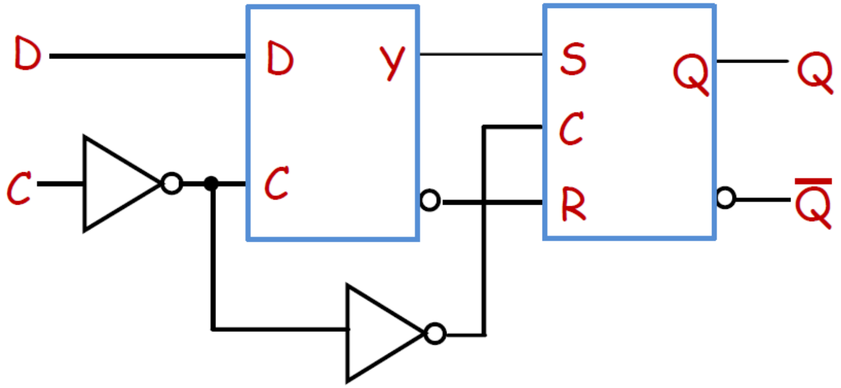
Gate delay

Edge-Triggered D Flip-Flop

- The change of D flip-flop output is associated w/ negative edge at the end of the pulse (**negative-edge triggered FF**)



- **Positive-edge triggered** D flip-flop is formed by adding an inverter to ‘C’ input



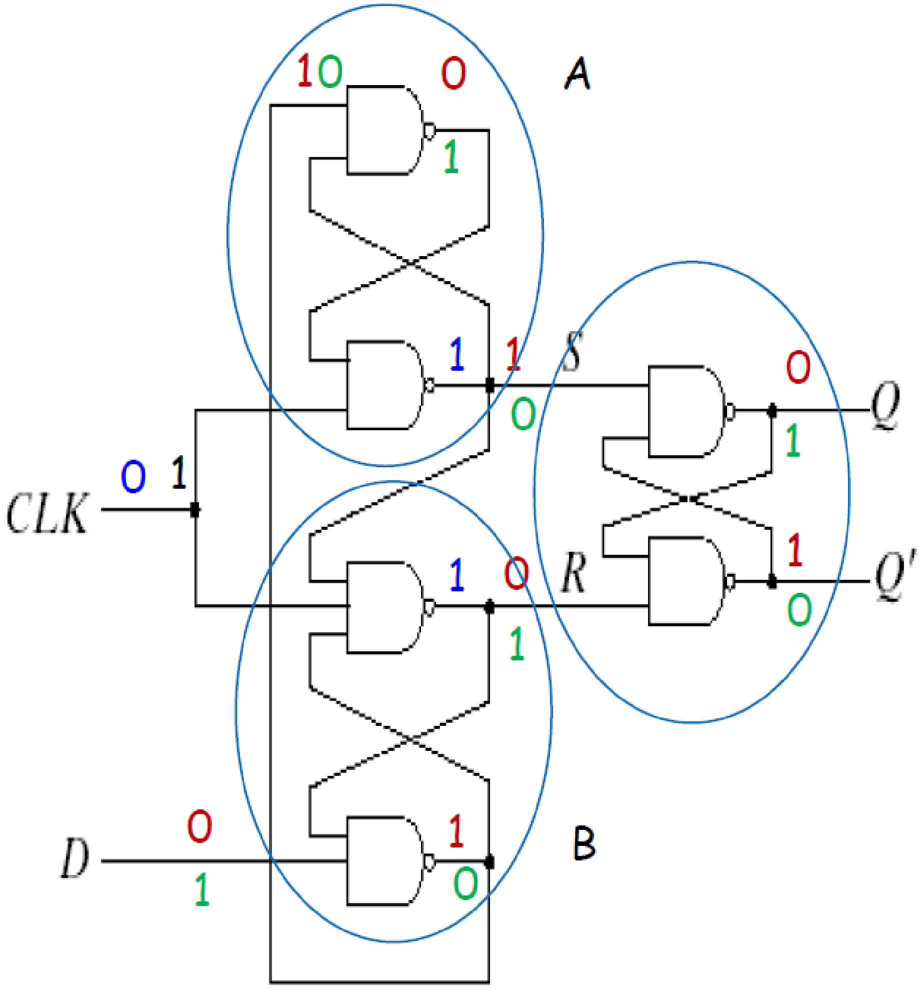
C(0→1): copies the state of the master latch

D-Type Positive Edge-Triggered Flip-Flop

- An efficient construction of an edge-triggered D FF uses three SR latches
  - Two latches respond to D and CLK
  - The third latch provides Q

✓ If CLK = 0, both output signals of input stage are 1 regardless of D

✓ The output latch is unaffected and it stored the <u>previous state</u>



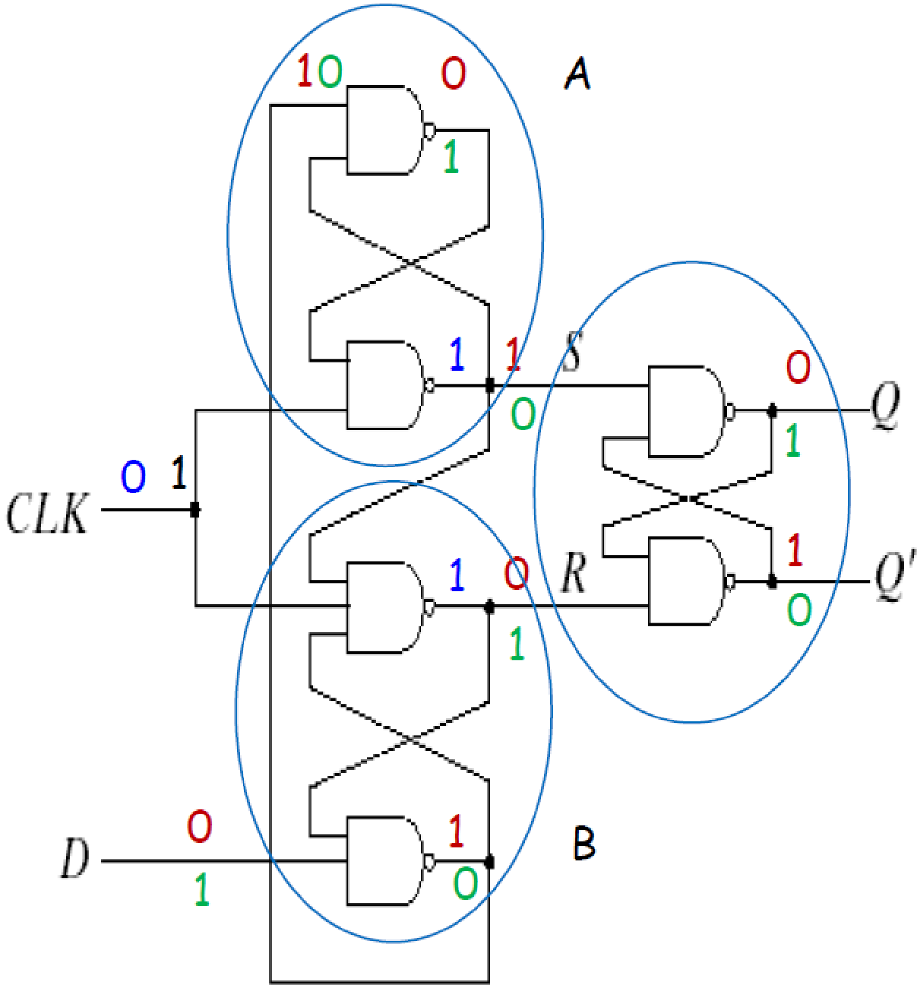
D-Type Positive Edge-Triggered Flip-Flop

- An efficient construction of an edge-triggered D FF uses three SR latches

✓ When CLK goes from 0 to 1, only one of the output voltages goes to 0 & sets/resets the output latch

✓ Consider D = CLK = 0 with {R, S} = 2’b11. What happens with CLK 0→1 ?

✓ Consider CLK = 0, D = 1 with {R, S} = 2’b11. What happens with CLK 0→1 ?



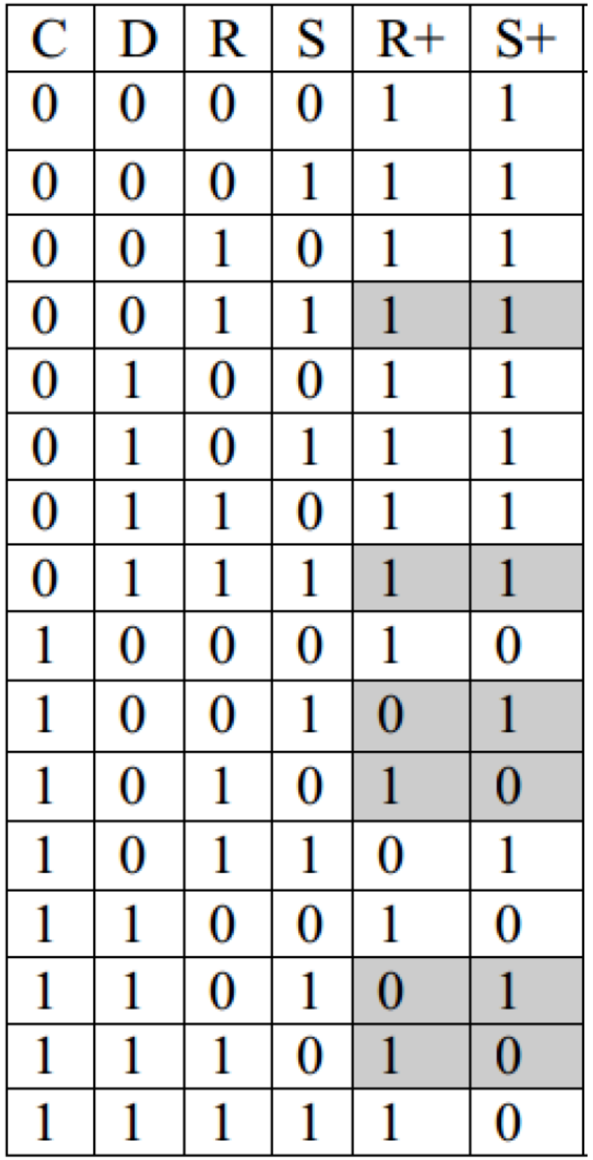
State Transition for Previous D Flip-Flop

- To analyze the previous FF design, the next state equations are:

$$S_+ = (\overline{C \cdot A}) = \overline{C} + \overline{A} = \overline{C} + B \cdot S =$$
$$= \overline{C} + (\overline{D} + \overline{R}) \cdot S$$

$$R_+ = \overline{S} + \overline{C} + \overline{B} = \overline{S} + \overline{C} + R \cdot D$$

- These equations lead to the following flow table where input variables are: C (CLK), D, R, and S

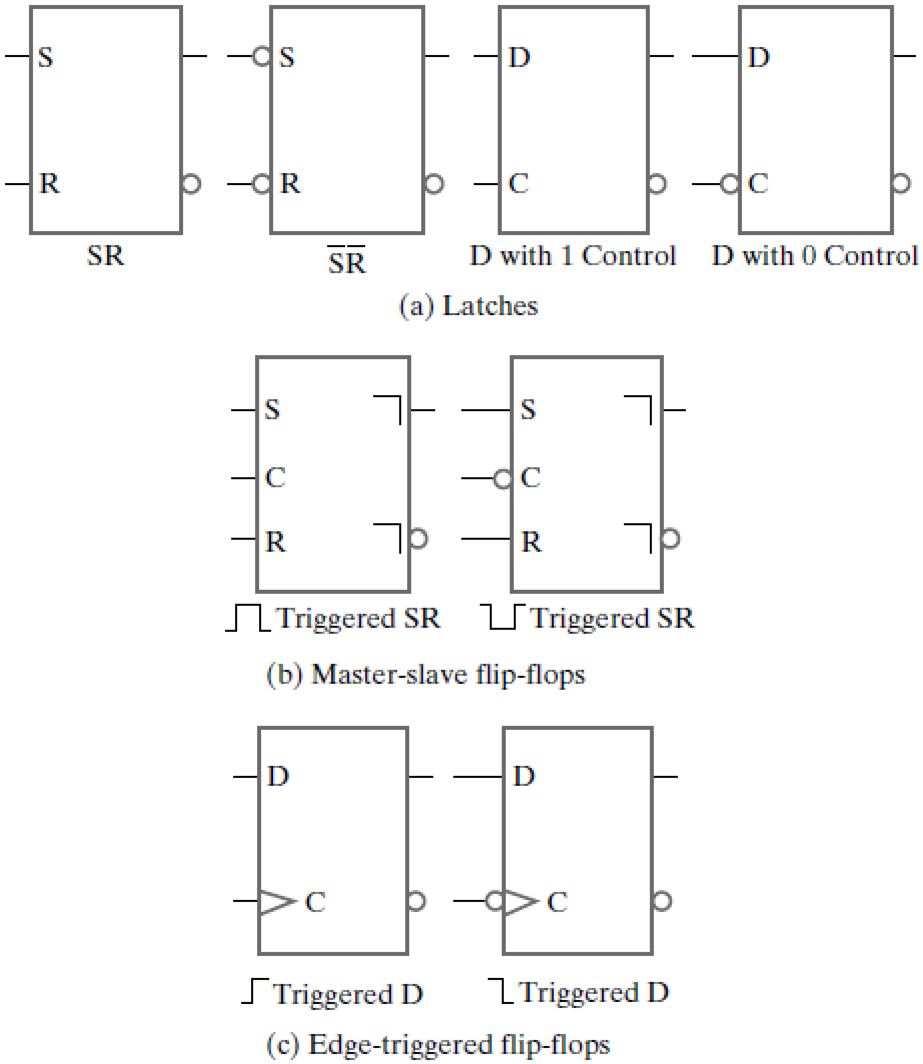


C	D	R	S	R+	S+
0	0	0	0	1	1
0	0	0	1	1	1
0	0	1	0	1	1
0	0	1	1	1	1
0	1	0	0	1	1
0	1	0	1	1	1
0	1	1	0	1	1
0	1	1	1	1	1
1	0	0	0	1	0
1	0	0	1	0	1
1	0	1	0	1	0
1	0	1	1	0	1
1	1	0	0	1	0
1	1	0	1	0	1
1	1	1	0	1	0
1	1	1	1	1	0

Standard Graphics Symbols

- A rectangular block with inputs (left) and outputs (right)
  - Bubble represents the complement
  - Ex) S`R` latch

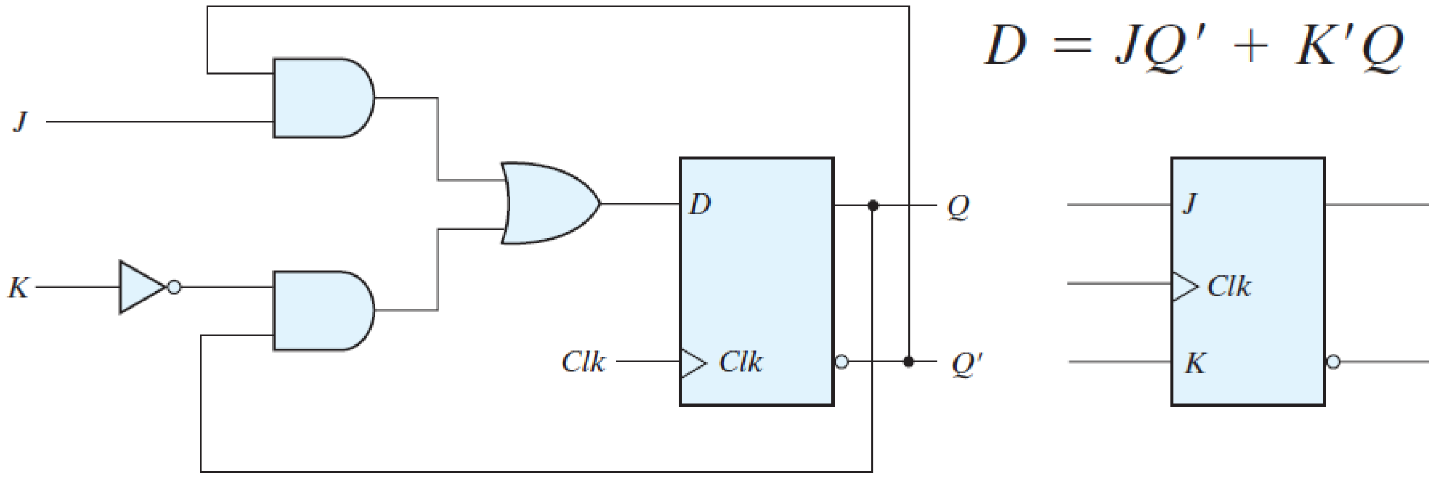
- The inverters unfortunately cause the clock signal to these FFs to be delayed
  - Careful design of **clock tree** is required



Other Flip-Flops

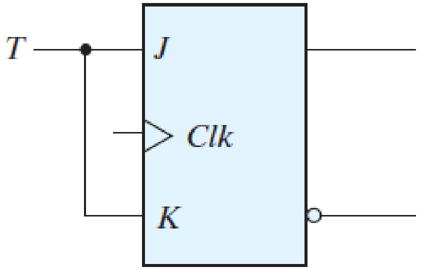
- D flip-flop is the most economical and efficient (why?)
  - Still, there are other types of FFs that are less widely used
  - **JK** and T flip-flops ✓Set to 1, reset to 0, or complement output

$$D = JQ' + K'Q$$

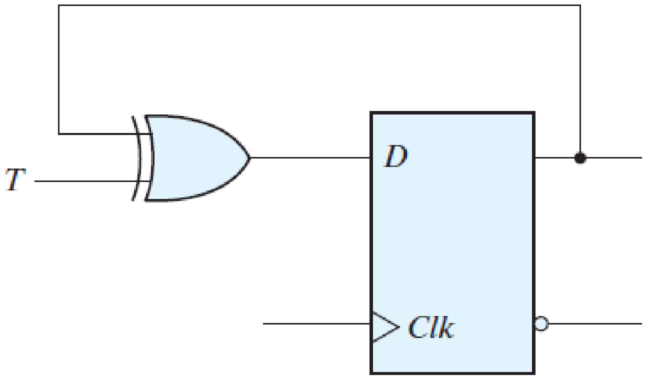


Other Flip-Flops

- D flip-flop is the most economical and efficient (why?)
  - Still, there are other types of FFs that are less widely used
  - JK and T flip-flops (toggle FF)

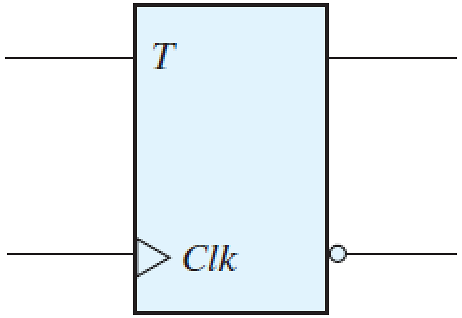


(a) From *JK* flip-flop



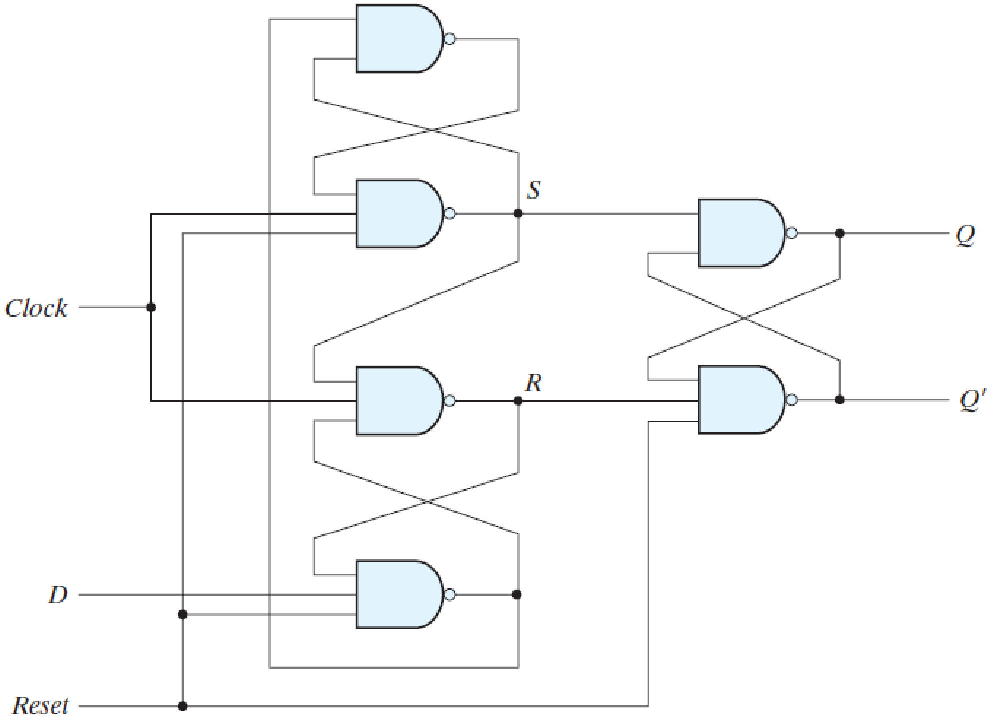
(b) From *D* flip-flop

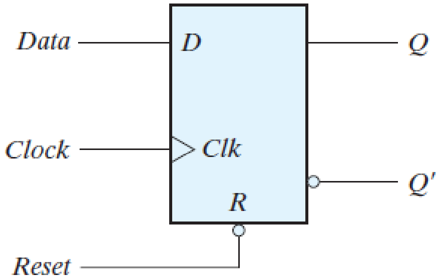
$$D = T \oplus Q = TQ' + T'Q$$



Direct Inputs

- Some FFs have asynchronous inputs to force FF to a particular state
  - Direct set (preset): sets FF to 1
  - Direct reset (clear): sets FF to 0
  - Good to bring all FFs to a known starting state after power-on





Graphic symbol

R	*Clk*	*D*	*Q*	*Q'*
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

Function table