

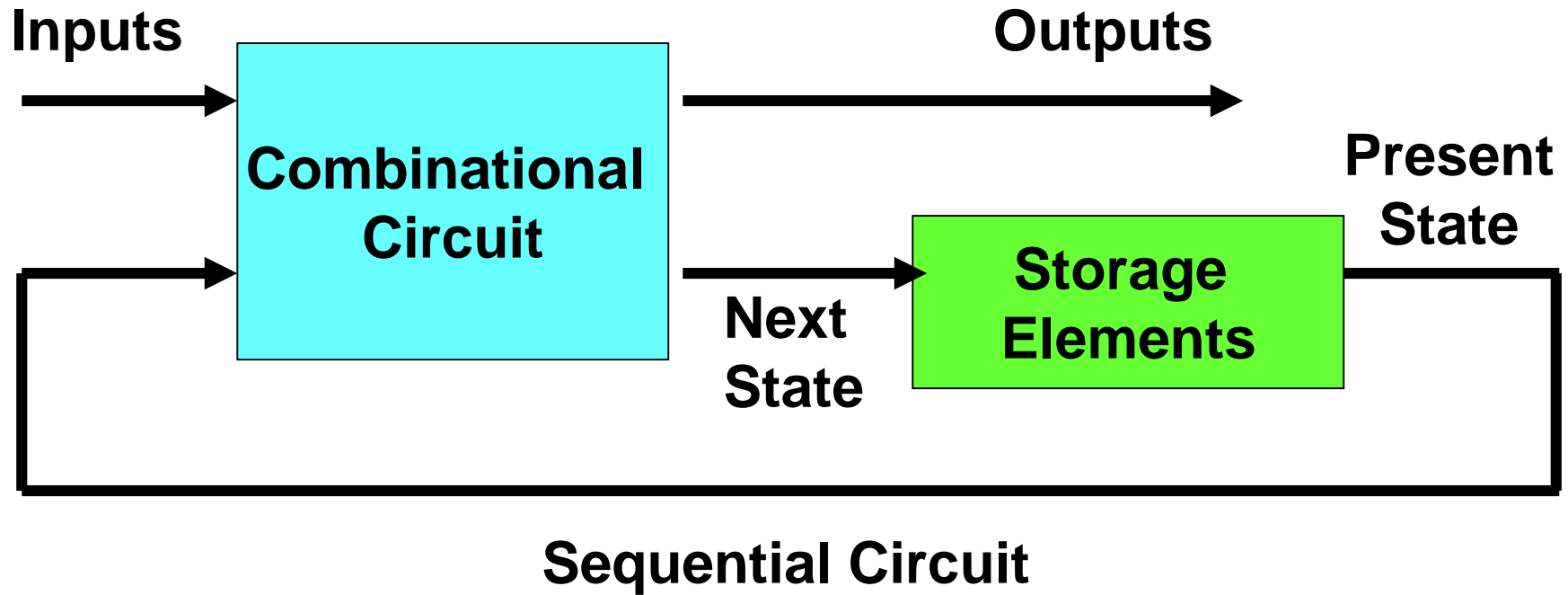
Overview: Sequential Logic Design Principles

- * Introduction
- * Bistable Elements
- * Latches & Flip-Flops
- * Analysis Procedure
- * Design Procedure




Chapter 7 – “Digital Design: Principles and Practices” book

Sequential Circuits (1/2)



Sequential Circuits (2/2)

- **Sequential systems** consist of *combinational circuits* and some form of *memory*.
- **Memory** can be either:
 - conventional memory devices; **or**
 - some type of feedback or delay network that serves in place of memory.
- **Examples** of **sequential circuits**:
 - Traffic Lights;
 - elevator controller.



Can you think of
other examples?

Synchronous *versus* Asynchronous

- *Sequential systems* come in two basic forms:
 - **Synchronous** → The system's behavior can be defined from the knowledge of its signals at *discrete moments in time*.
 - **Asynchronous** → The behavior depends on the order in which inputs change and can be affected at *any instant of time*.
1. **How** do you think synchronisation is achieved?
 2. **What** do you think may happen with asynchronous systems?

State

- **State**: The state or characteristics of a circuit are described by the values of its outputs.
 - Given a circuit with N outputs, with each taking on a binary value, the circuit has a total of 2^N possible states.
 - A state can be *stable* or *metastable*.
- For a sequential circuit, given the current state and input, we can predict the next state.

Sequential circuits: also called **FSMs** (Finite State Machines).

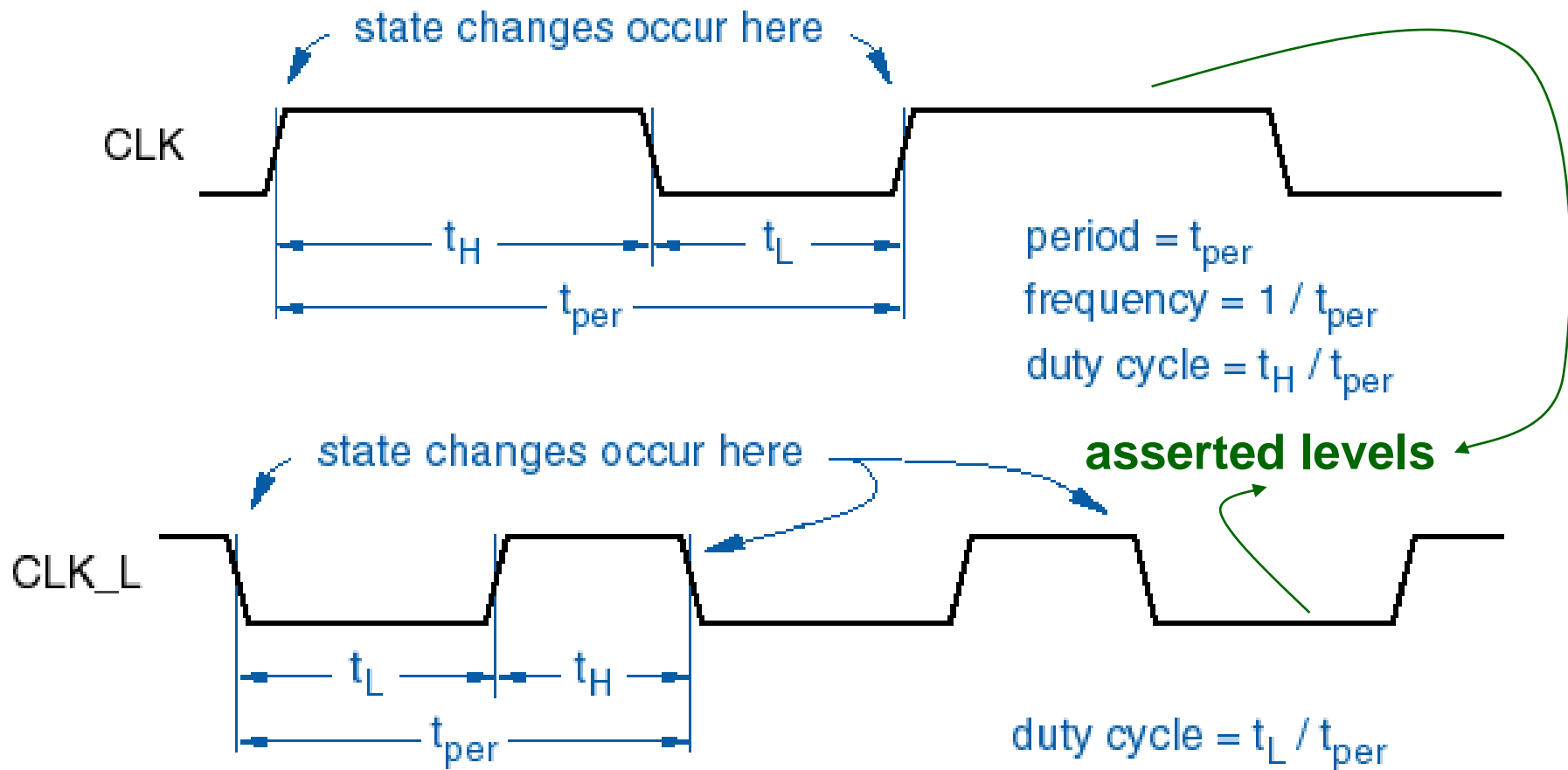
Clock Signals (1/2)

- **Characteristics of a clock signal:**
 - Most sequential circuits undergo a **state change** by a clock signal.
 - A clock signal may be *active high* or *active low*.
 - Its possible states include *high*, *low*, *rising edge*, *falling edge*.
 - Its parameters include *period*, *frequency*, and *duty cycle*.

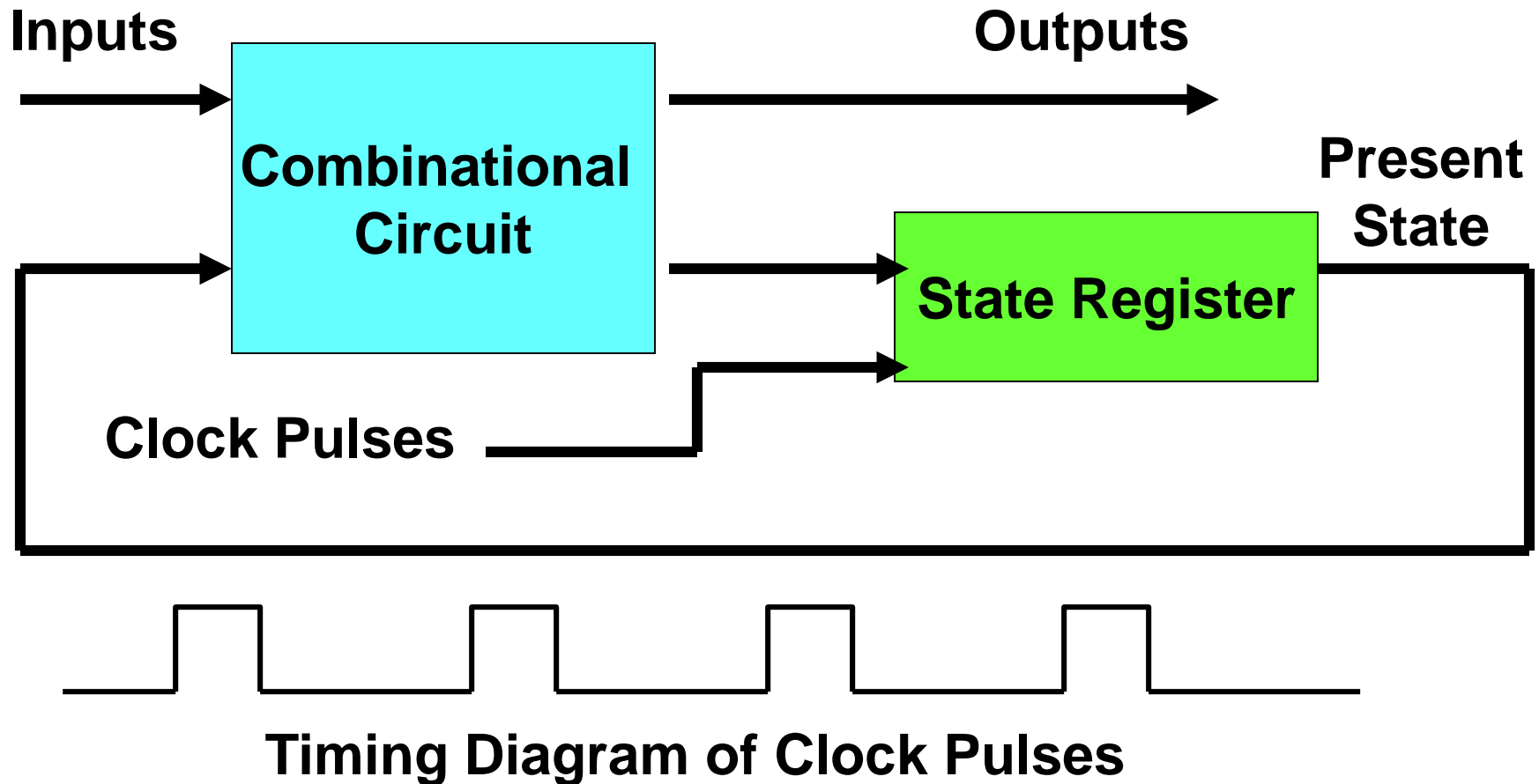
Period: time between successive transitions in the same direction.

Clock Signals (2/2)

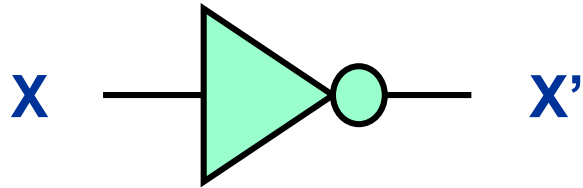
- Graphical illustration of a clock signal's operation:**



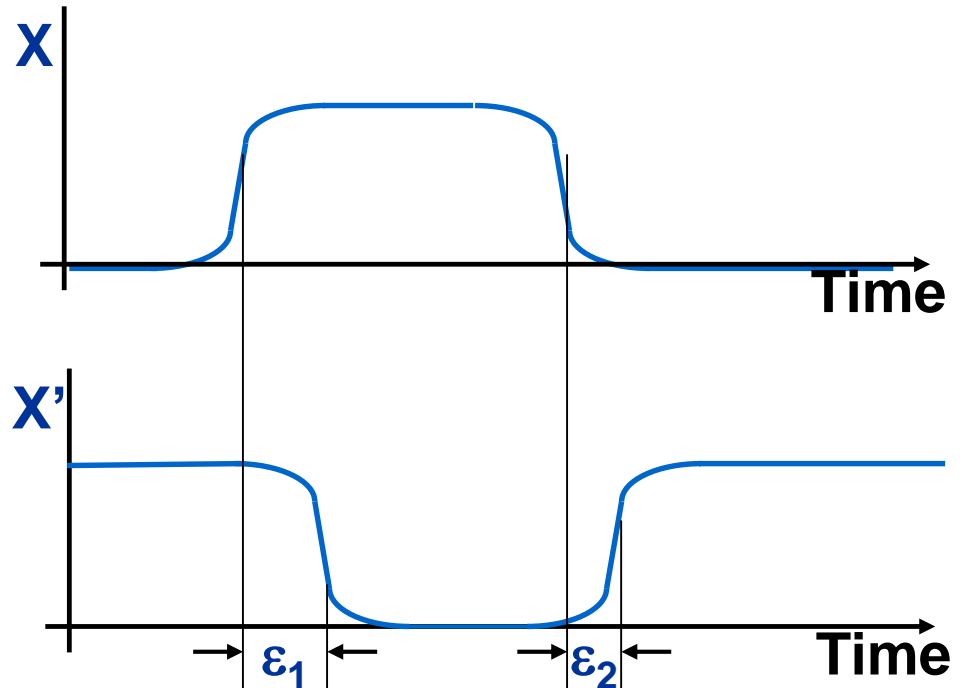
Synchronous Clocked Sequential Circuit



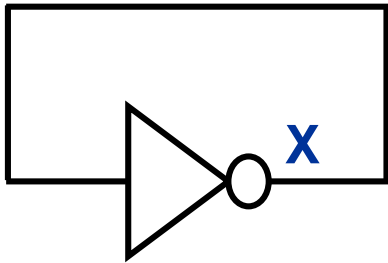
Gate Delays and Time Diagrams



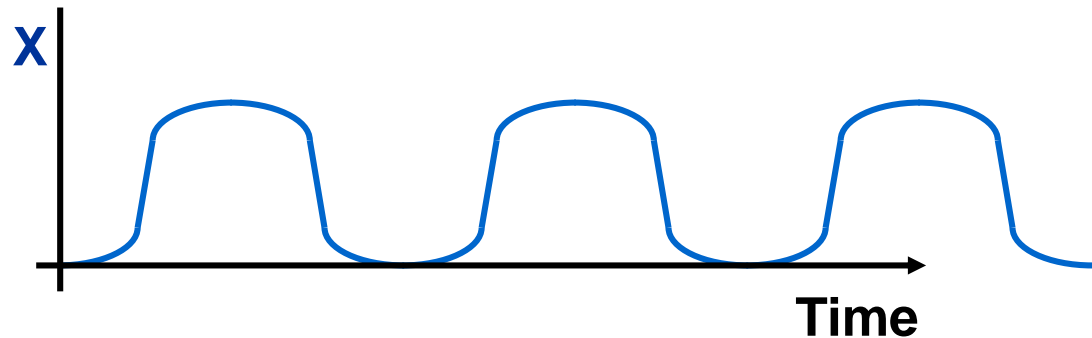
ϵ_1 may be different from ϵ_2 ,
why?



Oscillating Circuit: Not Stable



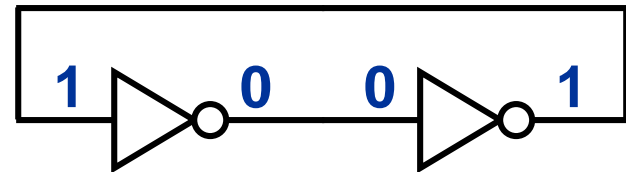
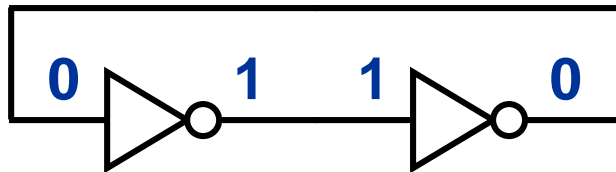
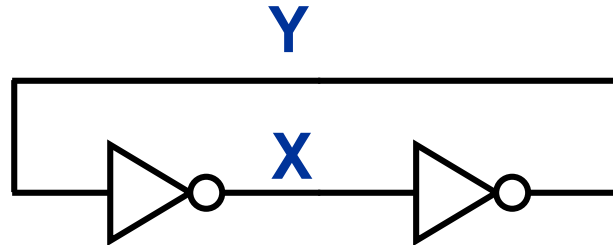
What does “**propagation delay**” depend on?



Circuit **never** reaches stability!!

Feedback Loop

Simplest bistable feedback circuit.



Overview: Sequential Logic Design Principles

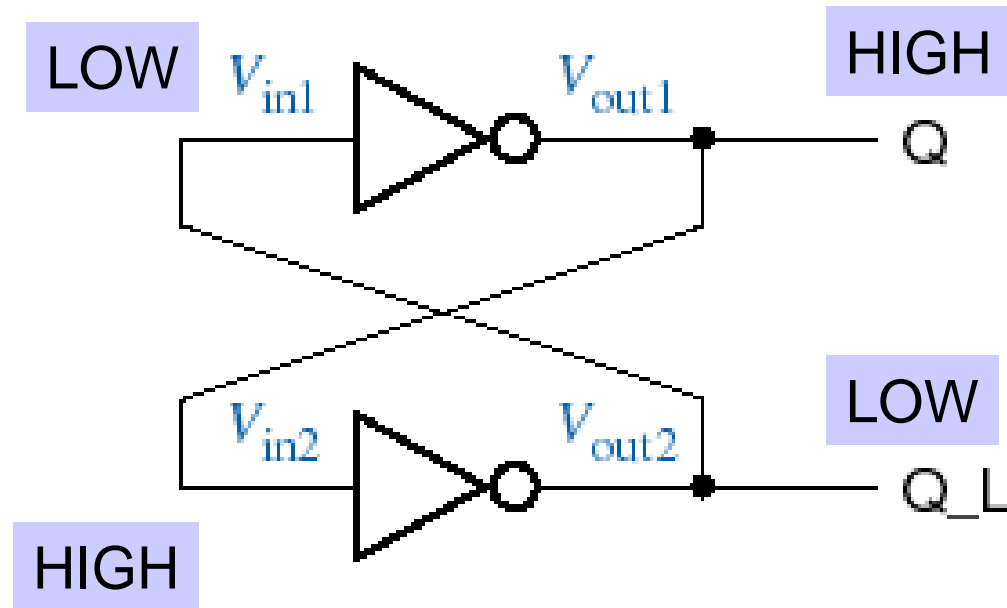
- * Introduction
- * Bistable Elements
- * Latches & Flip-Flops
- * Analysis Procedure
- * Design Procedure



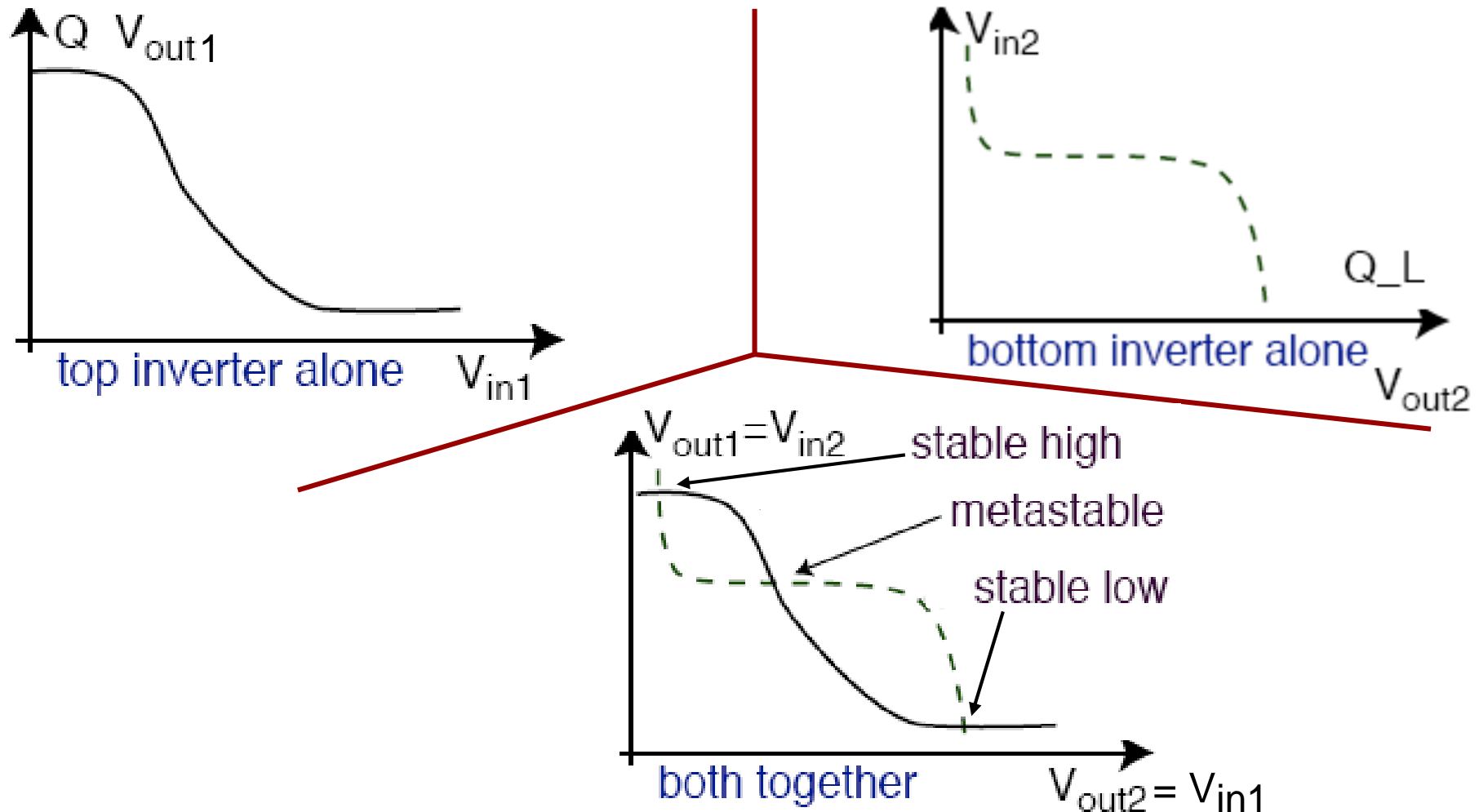
Chapter 7 – “Digital Design: Principles and Practices” book

Bistable Elements (1/2)

- **Bistable Element**: the simplest sequential circuit.
 - Consists of *2 inverters* and has *no inputs* and *2 outputs*.
 - Its *state* is characterised by the *values of its 2 outputs*.
 - It has only *2 states*: $(1,0)$ or $(0,1)$.
 - Its *output* depends only on its *previous input*, through a feedback loop.

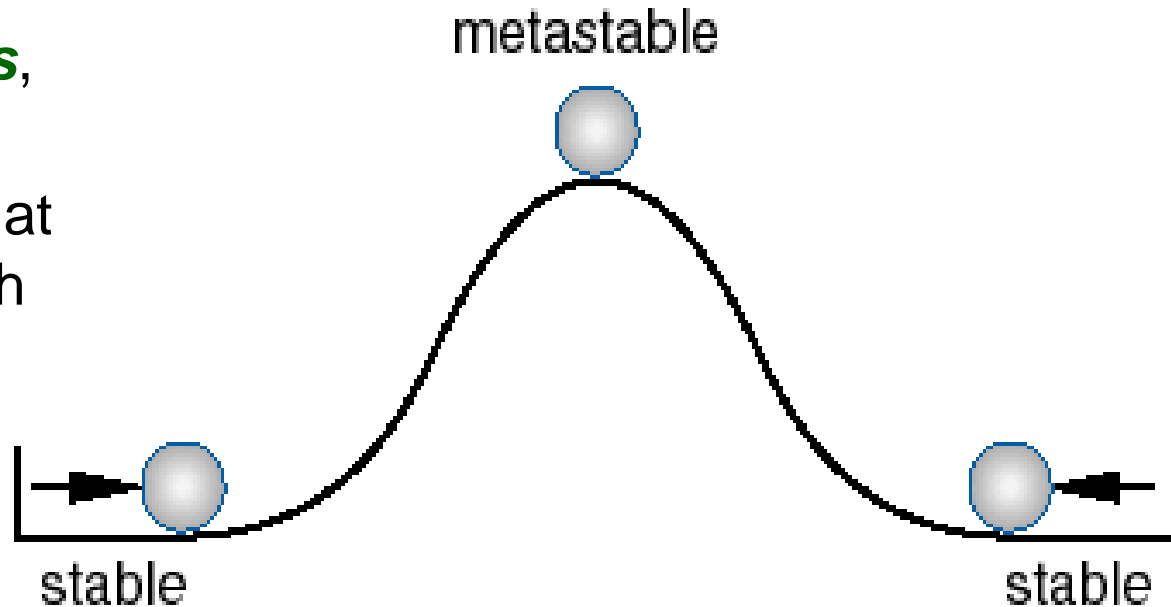


Bistable Elements (2/2)



Metastability & Stable States

- *There are not 2 states, there are 3!*
 - **Metastable** occurs at the point where both inputs are halfway between *0* and *1*.
 - Not a valid state!
 - Could stay in **metastable** forever if it wasn't for noise.



Metastability: Why is it important?

- **Metastability:**
 - All real systems are subject to it.
 - Problems are caused by “*asynchronous inputs*” that do not meet **flip-flop setup** and **hold times**.
 - Metastability is severe in high-speed systems since clock periods are so short, and “*metastability resolution time*” can be longer than one clock period.
 - Many digital designers, products, and companies have been “burned” by this phenomenon.

Overview: Sequential Logic Design Principles

- * Introduction
- * Bistable Elements
- * **Latches** & Flip-Flops
- * Analysis Procedure
- * Design Procedure



Chapter 7 – “Digital Design: Principles and Practices” book

Latches

- **Latch**: Basic storage element from which all *flip-flops* are constructed.
- *Latches* by themselves are not practical for use in synchronous sequential systems:
 - Simple *SR Latches* and *D Latches* do not have control methods to synchronise storage of data elements and therefore cannot be used in synchronous systems.

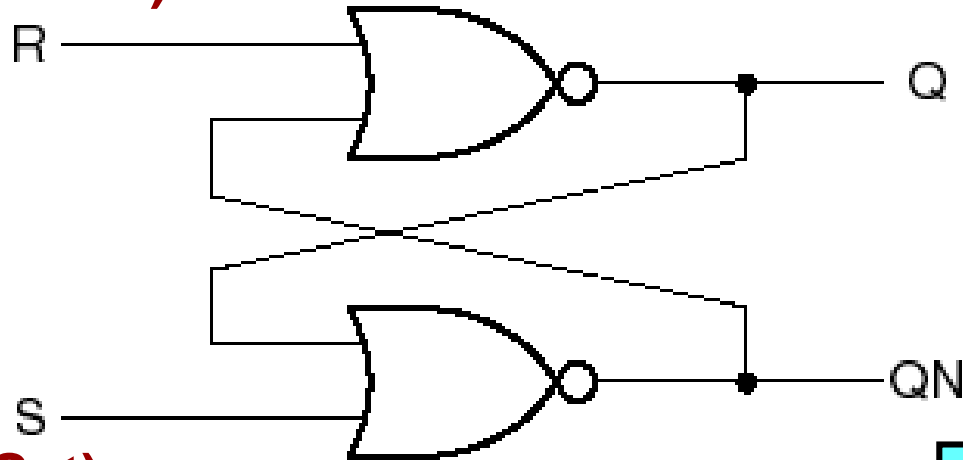
Set Reset (SR) Latch

- **SR Latch:**
 - **Undefined state** occurs when both **Set** and **Reset** inputs are **True** (this is “Problematic Design”).
 - Two types: **NOR Gate** and **NAND Gate**.
 - Basically operate in the same way, but use different logic levels to define **truth**.
 - 74 series logic chips: **74LS279**
 - However, **not very practical** for clocked synchronous circuits!

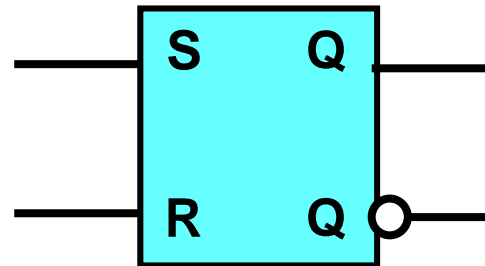
SR Latch with NOR Gates

SR Latch: (1) *S* sets or presets *Q* to 1;
(2) *R* resets or clears the *Q* output to 0.

(Reset)



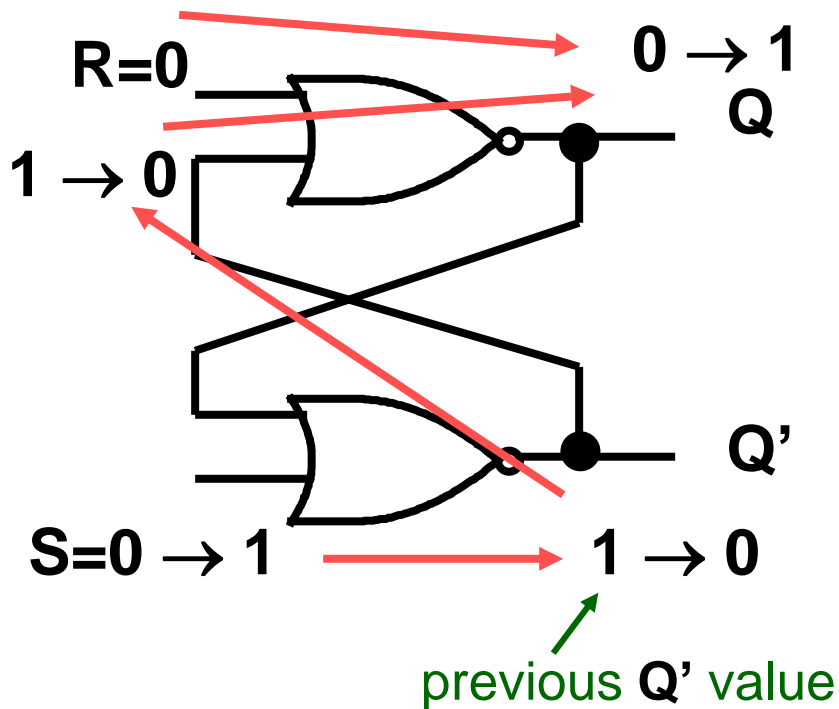
(Set)



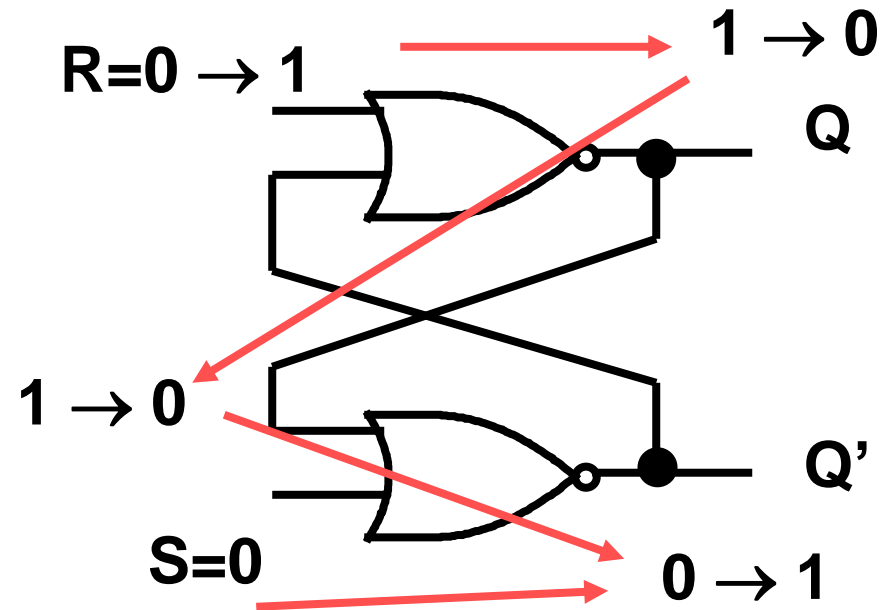
<i>Inputs</i>		<i>Outputs</i>	
S	R	Q	QN
0	0	last Q	last QN
0	1	0	1
1	0	1	0
1	1	0	0

Illegal (invalid)
operation when both
inputs are **True**.

SR Latch Operation (1/2)

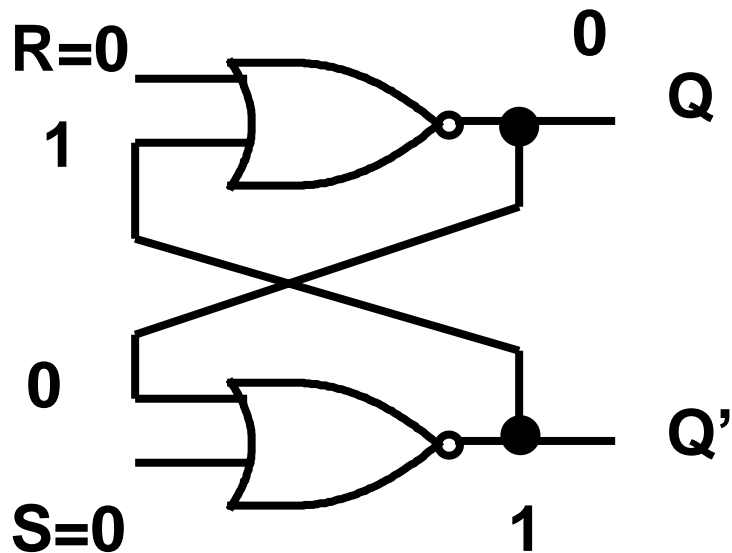


SET operation
(changing S to 1)



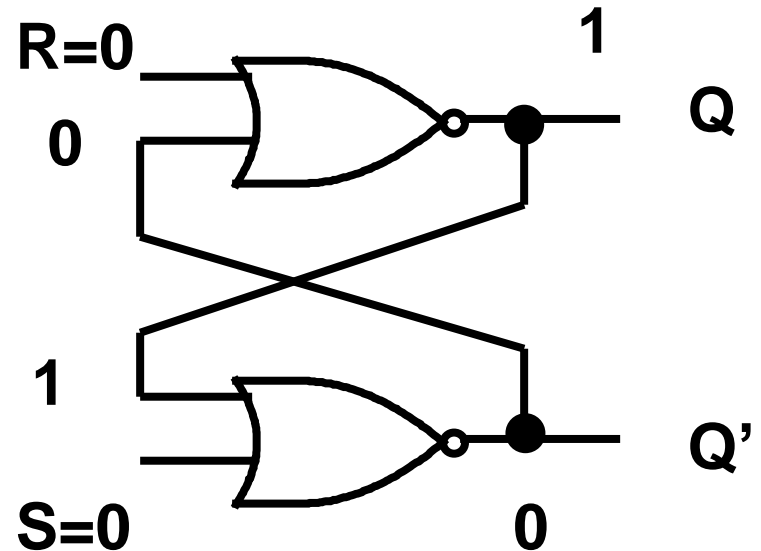
RESET operation
(changing R to 1)

SR Latch Operation (2/2)



Stable when:

$$S = R = 0, Q' = 1$$

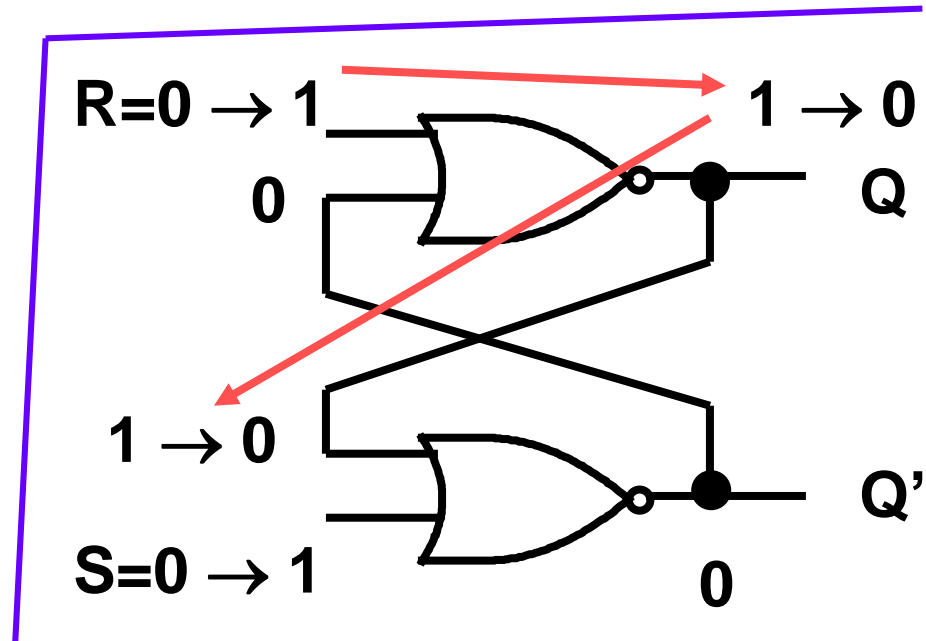


Stable when:

$$S = R = 0, Q = 1$$

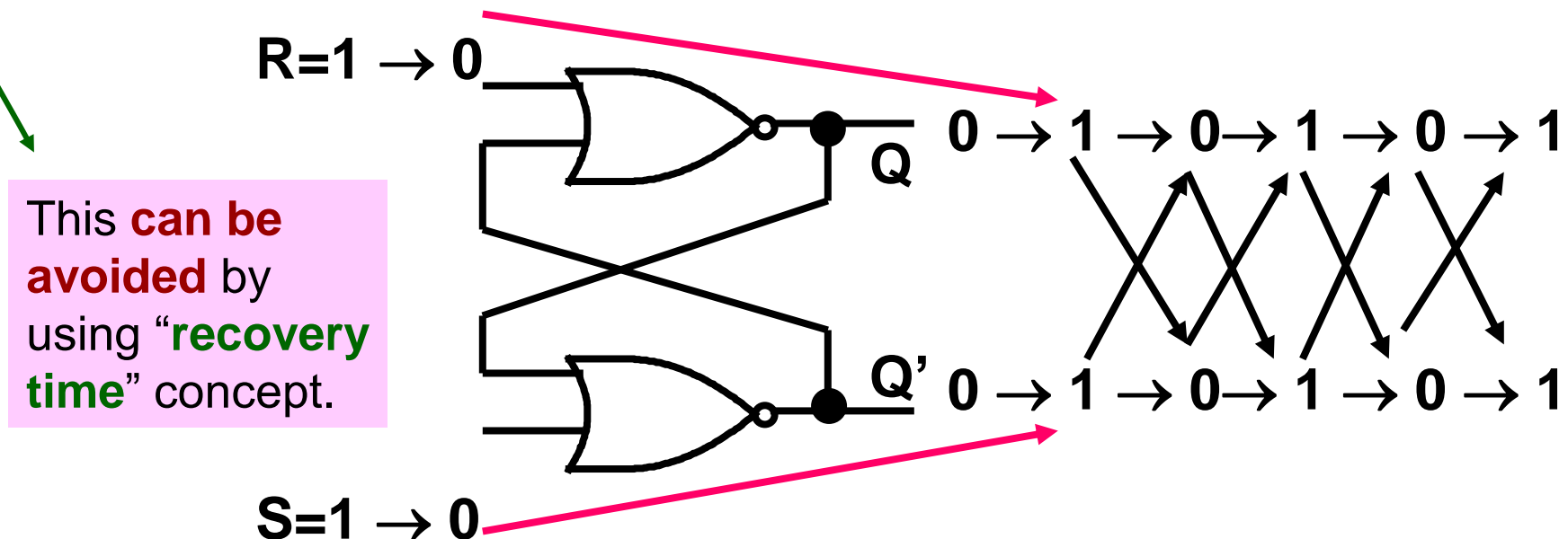
What about $S = R = 1$?

- What happens when both **Set** and **Reset** are **True**?
 - Assume that both S and R transition to 1 simultaneously.
 - Then, Q becomes '0', but Q' remains '0'! So **outputs are no longer complements of each other.**



What happens when S and R return to 0?

- **Oscillation** occurs if both S and R return to 0 simultaneously! But at some point, the system will settle into a stable condition.
- The bottom line is that $S = R = 1$ is an illegal input condition (or the SR latch should be designed, such that one input is *dominant*).



Recap. of the Last Lecture

- In the last lecture we introduced the concept of a sequential circuit
- We discussed the difference between a sequential circuit and a combinational circuit.
- We saw that there are 2 main forms of sequential circuit; asynchronous and synchronous.
- We saw that the output of an asynchronous circuit can be unreliable and unpredictable.
- We studied the first basic memory element; an SR flip-flop. We saw how it operates.
- We learned that a limitation of the SR flip-flop is that its output can enter an illegal state. A race condition is also possible.

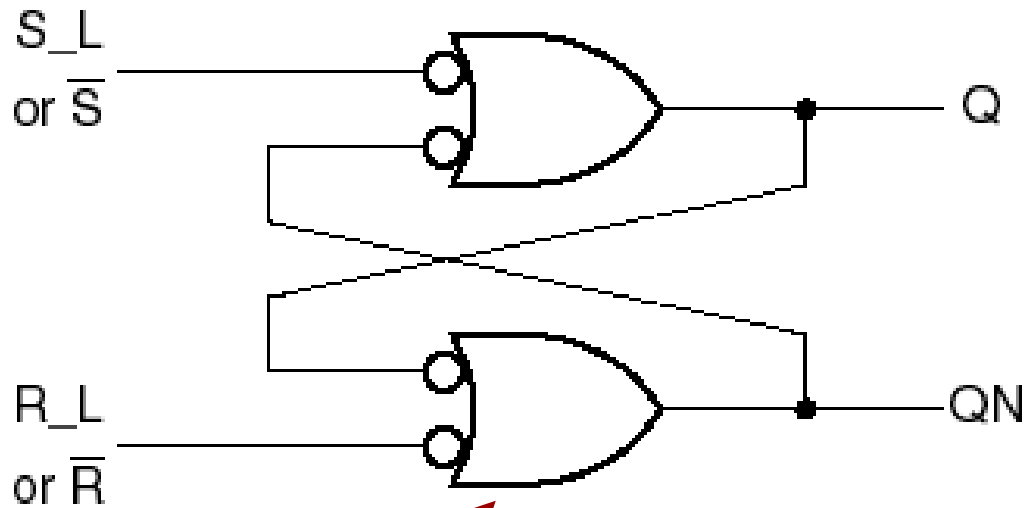
Learning Objectives for the Session

By the end of the session all students will be able to:

- 1) describe the operation of a: D-latch, a master slave flip-flop, and master slave JK flip-flop.
- 2) explain the problem with the SR latch that is solved by the master slave flip-flop.
- 3) explain the problem with the master slave flip-flop that is solved by the master slave JK flip-flop
- 4) differentiate between different types of flip-flop and make an appropriate selection for the application at hand

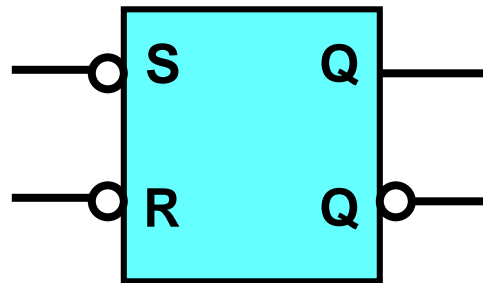
Some students will be able to explain the 1's/0's catching problem.

SR Latch (Built with NAND Gates)



Same function as
a **NAND** gate

$\bar{S}\bar{R}$ Latch



Inputs *Outputs*

S_L	R_L	Q	\bar{Q}
0	0	1	1
0	1	1	0
1	0	0	1
1	1	last Q	last \bar{Q}

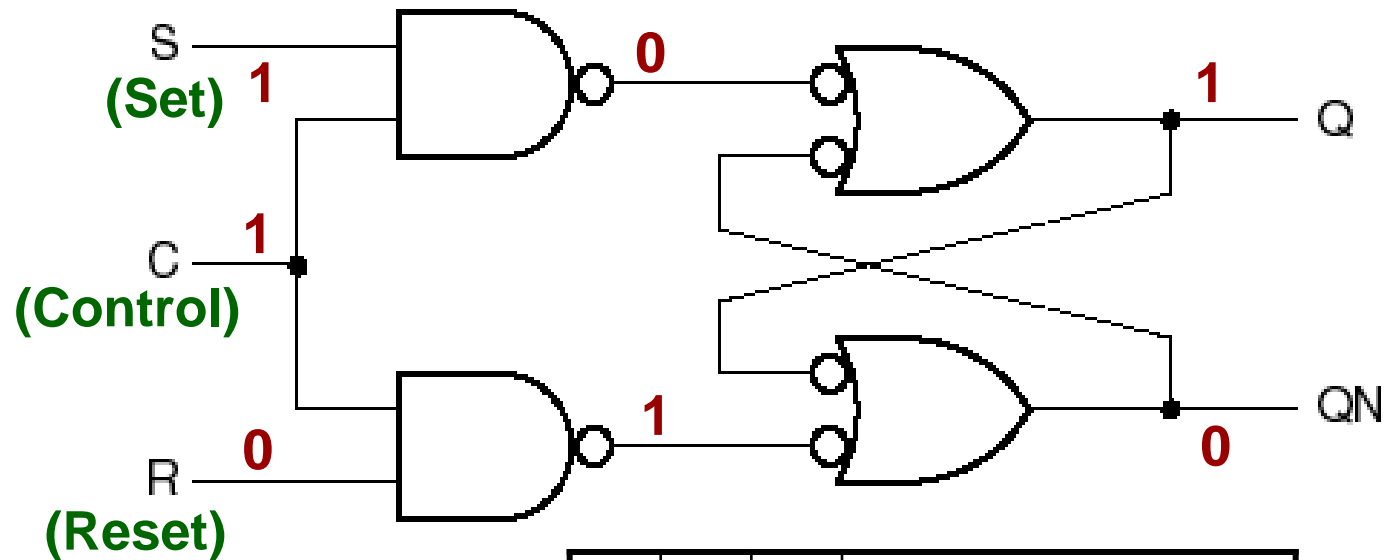
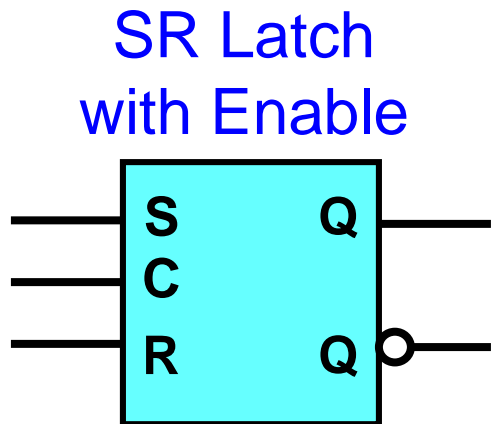
Illegal (invalid) operation
when both inputs are **True**.

SR Latch with Control Input (1/2)

- What happens if either **Set** or **Reset** are changed in an SR Latch?
 - Must come up with **control method** (**Enable**) to control when SR Latch can be changed.
 - Leaving **Control Line** at logic '0' prevents latch from being written with new value.
 - But **still have** a problem with **indeterminate outputs**!

SR latch with control input: **only sensitive to S, R** inputs when control line is asserted.

SR Latch with Control Input (2/2)

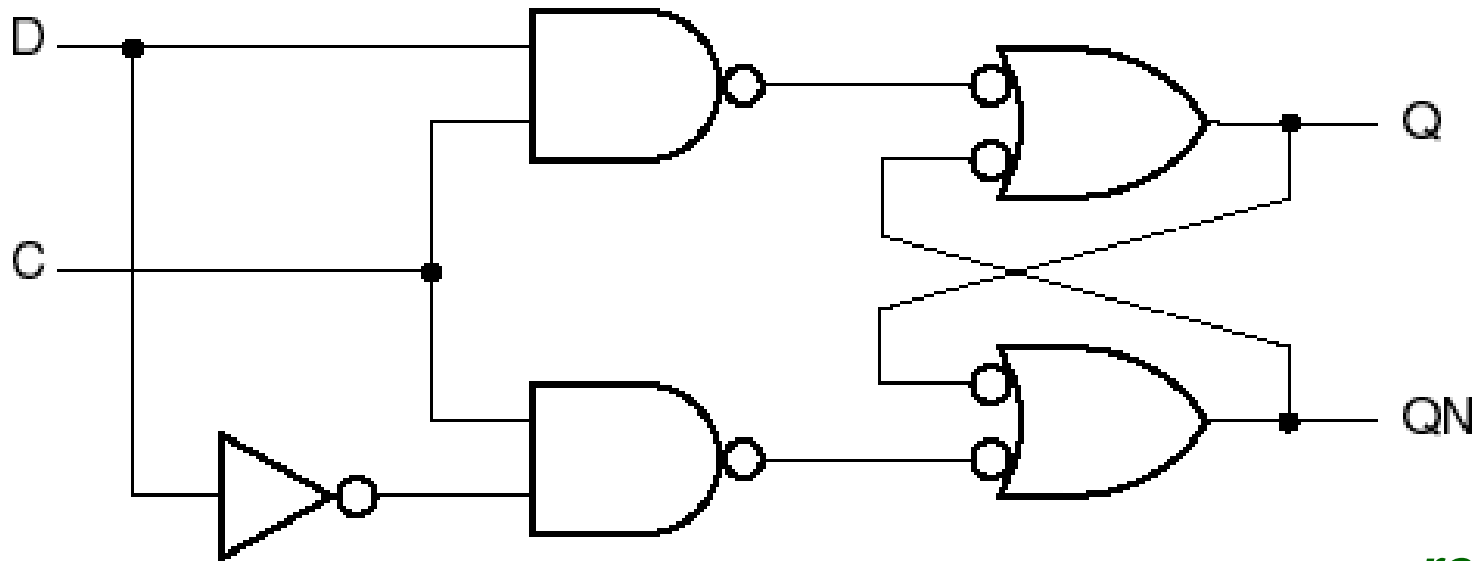


S	R	C	Q*
0	0	1	
0	1	1	
1	0	1	
1	1	1	
x	x	0	

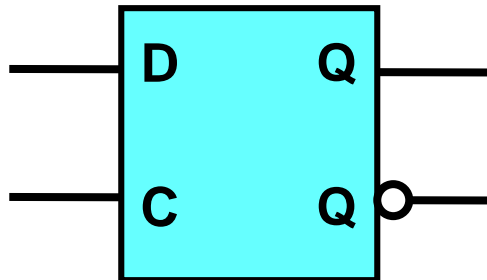
D Latch (1/2)

- So far, **we can say**:
 - Don't ever want to design a system where the next state is indeterminate.
 - Can eliminate the **indeterminate state** by ensuring that both **Set** and **Reset** inputs are never equal to '1' at the same time!
- **D Latch**:
 - Has only *two inputs*: **Data** (D) and **Control** (C).
 - *D* input to *Set* input and the complement of *D* input to the *Reset* input always.
 - No longer have an indeterminate state.

D Latch (2/2)



D Latch

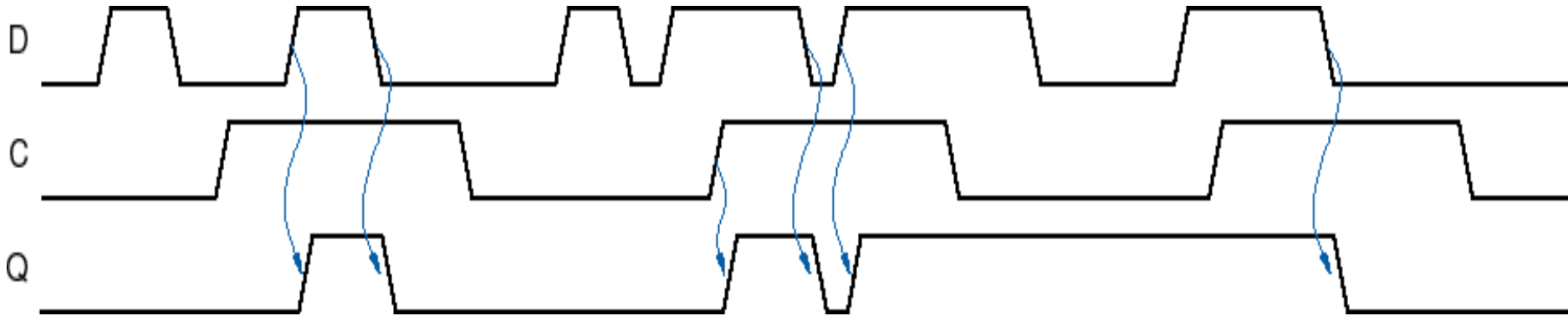


C	D	Q	QN
1	0	0	1
1	1	1	0
0	x	last Q	last QN

reset state

set state

D-Latch Operation



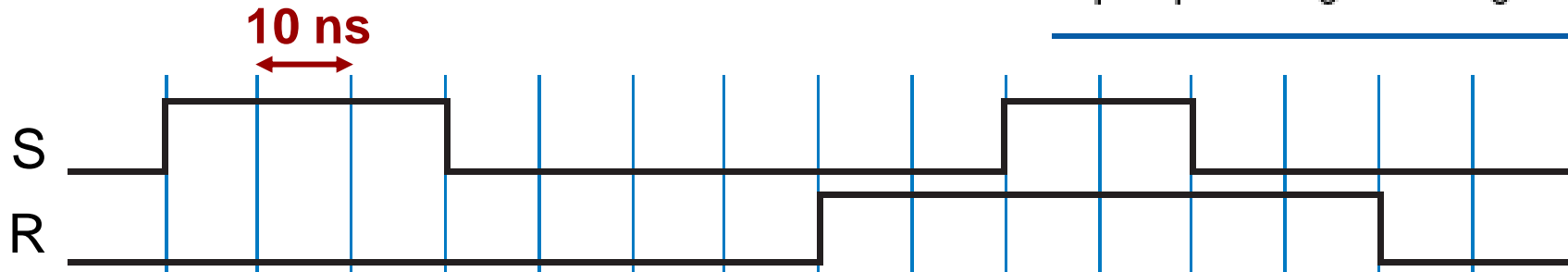
There is a difference between the behaviour of an ideal circuit and that of a practical implementation. In the later propagation delays are present.

Task 1

- Draw the outputs of an SR latch for the input waveforms shown below.
Assume that the propagation delay of a NOR gate is 10ns.

$$Q^* = S + R'Q$$

S	R	Q	QN
0	0	last Q	last QN
0	1	0	1
1	0	1	0
1	1	0	0



To be completed in class ...



Overview: Sequential Logic Design Principles

- * Introduction
- * Bistable Elements
- * Latches & Flip-Flops
- * Analysis Procedure
- * Design Procedure



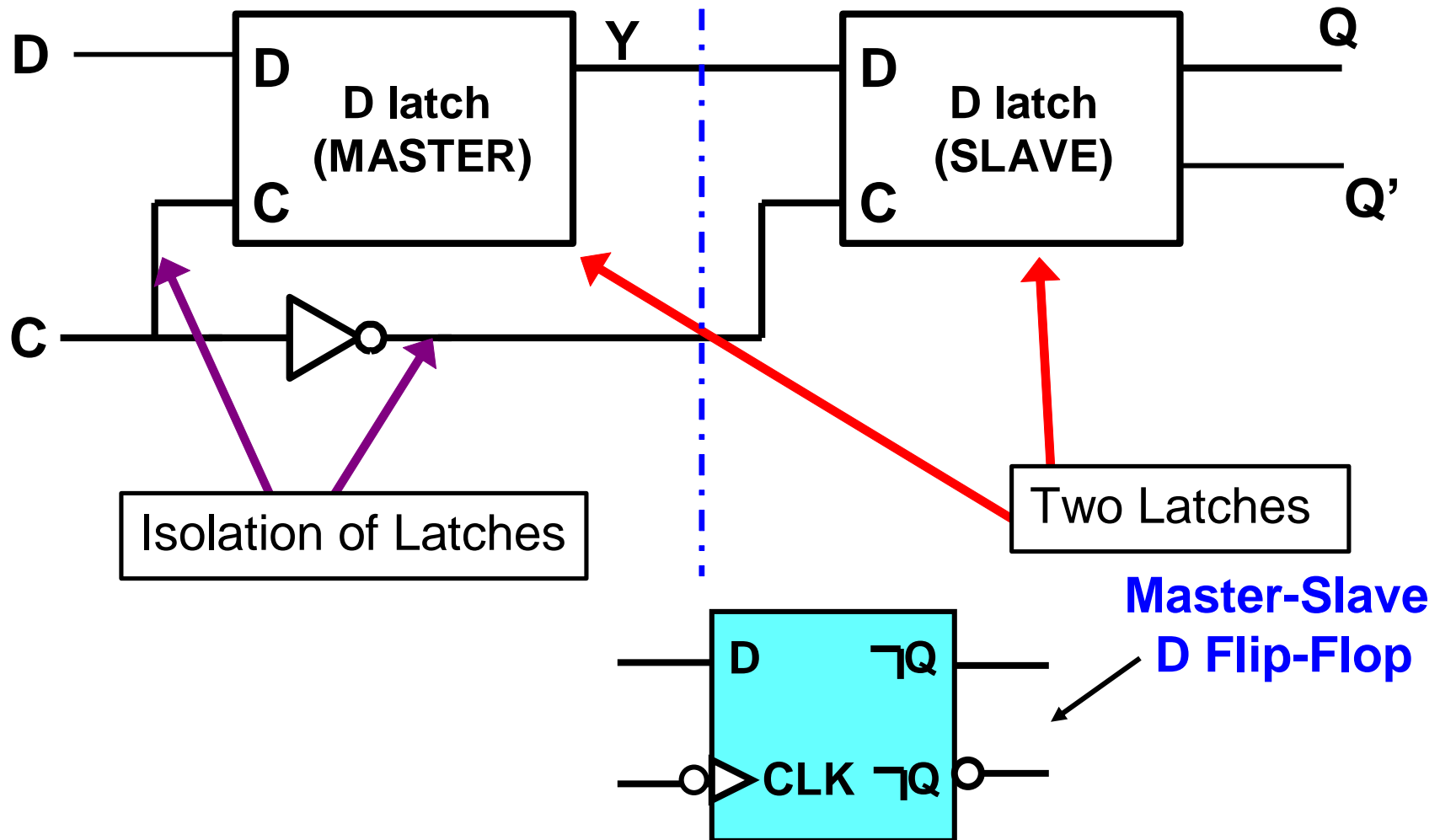
Chapter 7 – “Digital Design: Principles and Practices” book

Flip-Flops

- **Level sensitive latches** are inadequate, because:
 - Output continually changes if input changes.
 - **Race conditions** can occur.
- Need a better approach to storing data!
 - We will consider both **D and JK-type flip-flops**.
- You can check the following URL for **animated simulation of the various flip-flop types**:

<http://tams-www.informatik.uni-hamburg.de/applets/hades/webdemos/16-flipflops/10-srff/chapter.html>

D Type Master-Slave Flip-Flop (1/5)



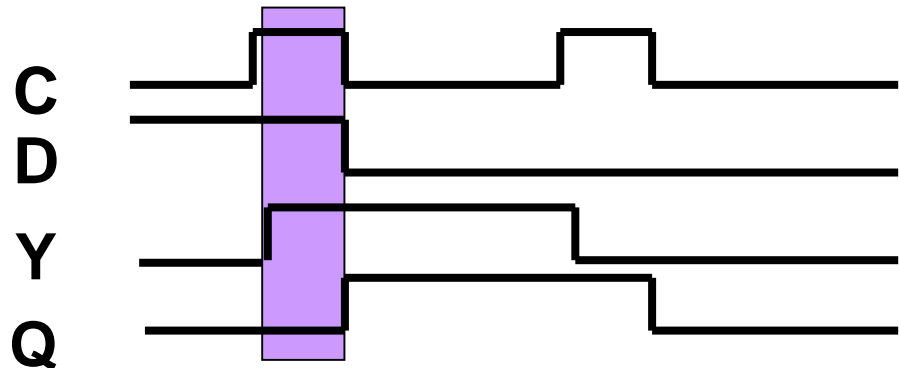
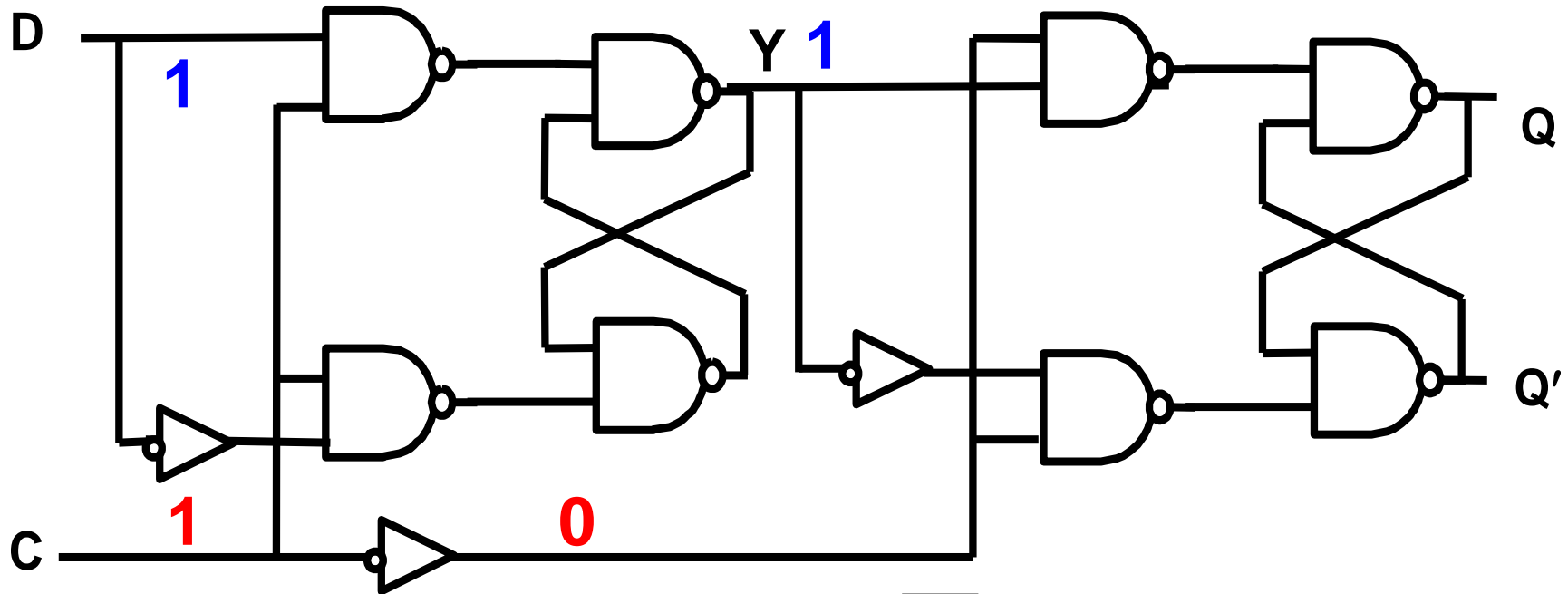
Master-Slave Flip-Flops

- **How Master-Slave Flip-Flops work:**
 - Employ *two latches*.
 - Isolate output from input:
 - *Output* cannot change continuously with *input*.
 - Input must have a *stable setup time*.

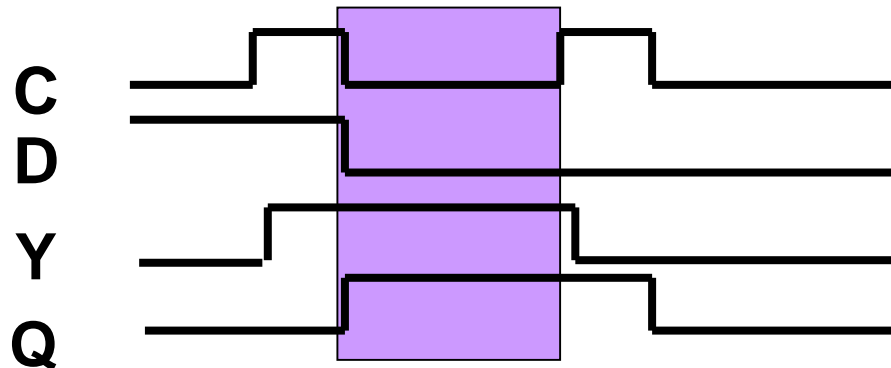
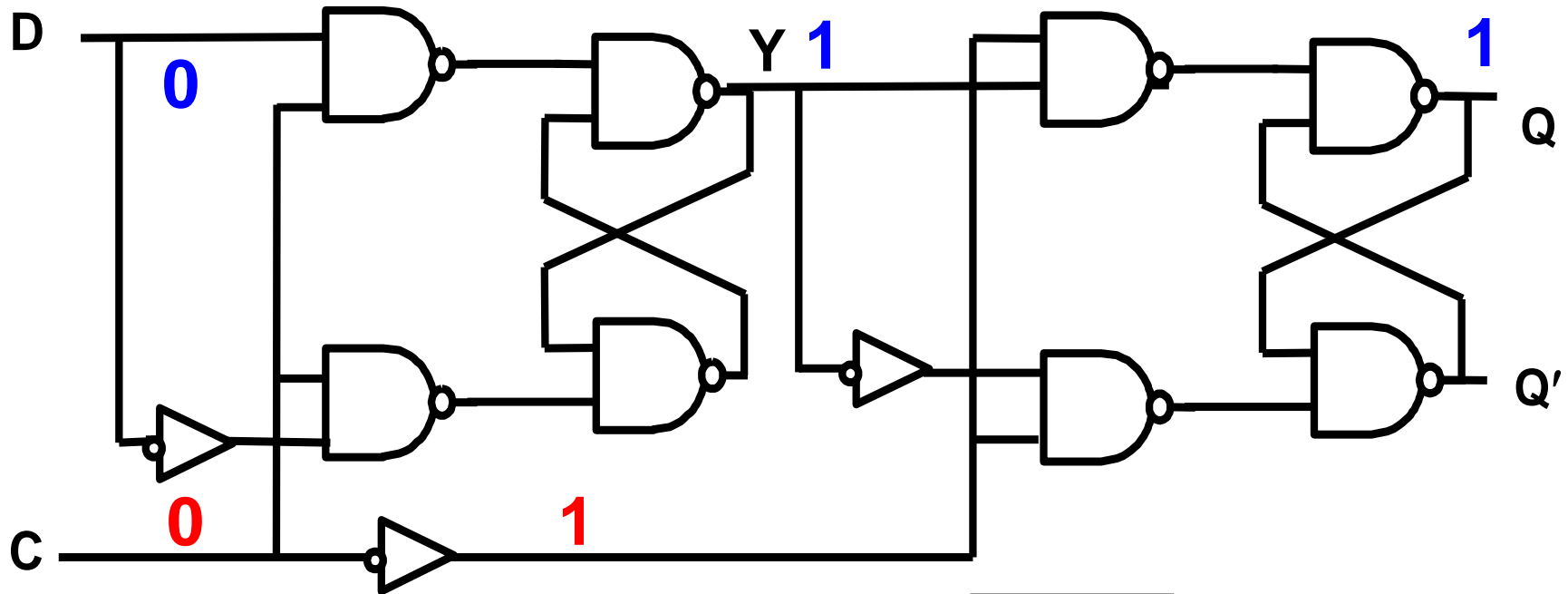
Master-Slave D Flip-Flop

- Very **popular design tool** (usually present in PLDs).
- Used extensively in designing **registers**.
- Uses **2 D Latches**:
 - *Master D Latch* to latch input.
 - *Slave D Latch* to latch output.
- Latches never operate together.
- Uses **narrow pulse clocking** approach.
- Eliminates racing conditions.

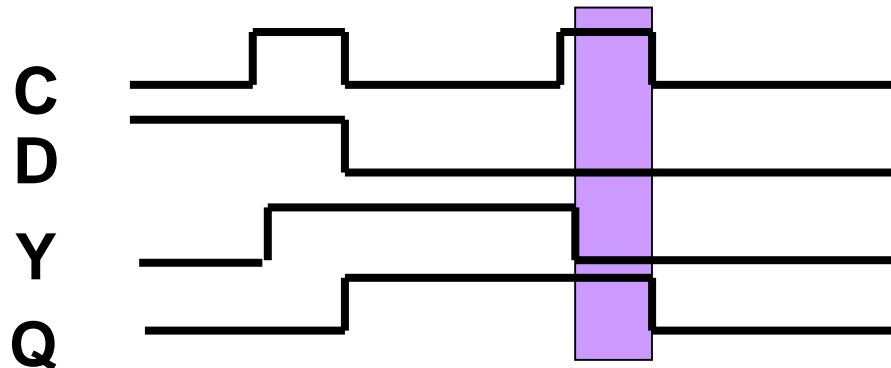
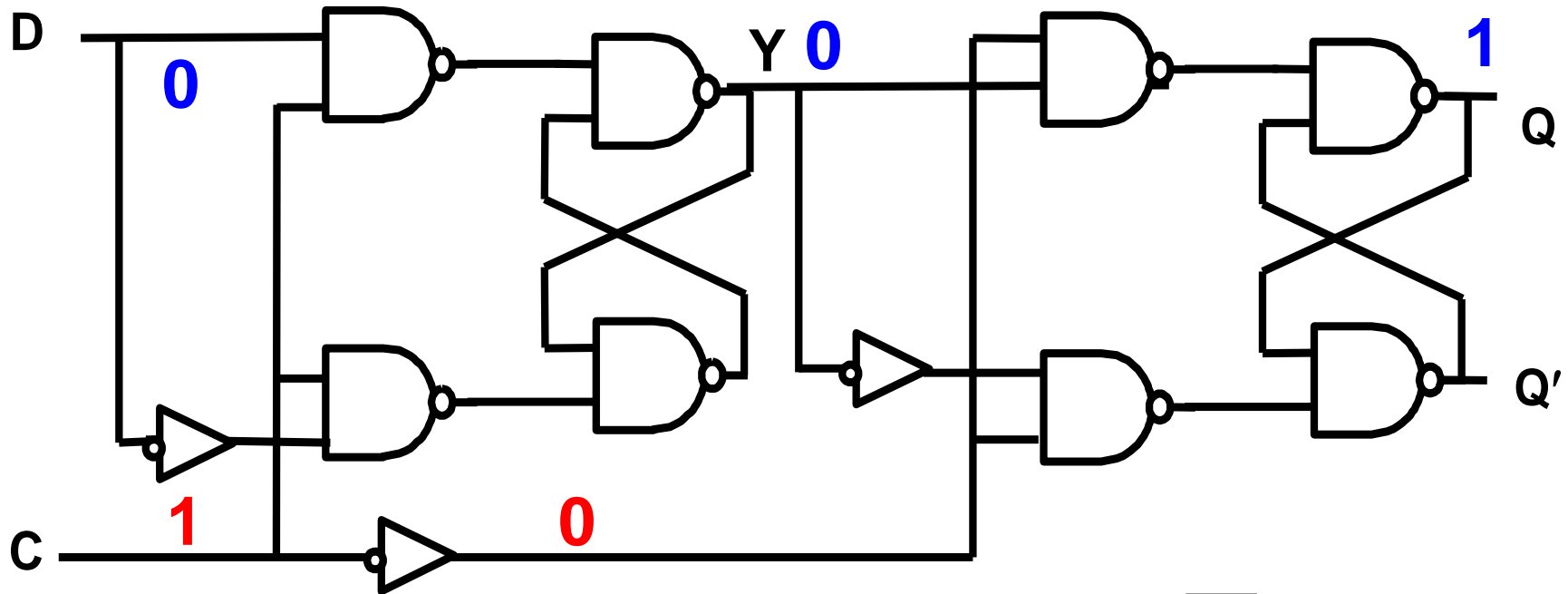
D Type Master-Slave Flip-Flop (2/5)



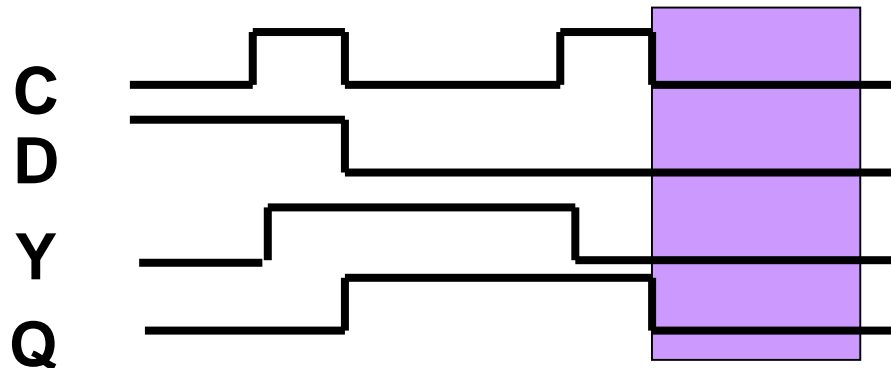
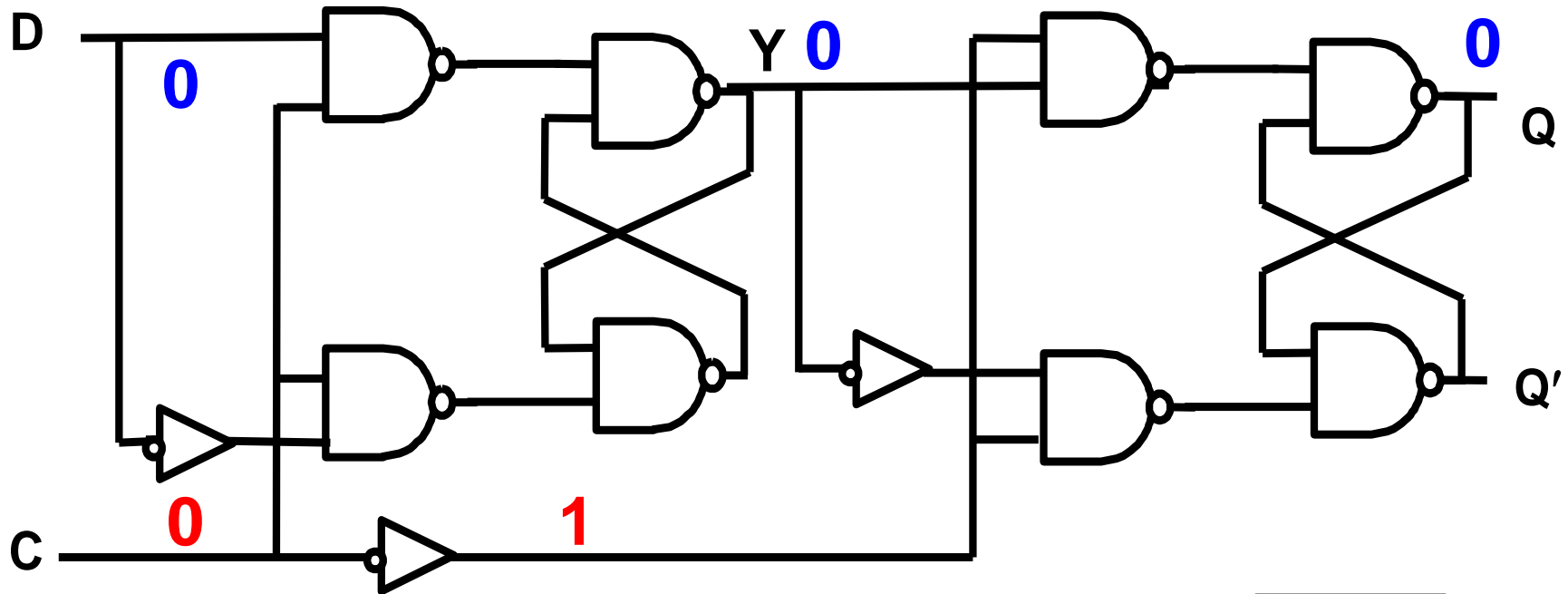
D Type Master-Slave Flip-Flop (3/5)



D Type Master-Slave Flip-Flop (4/5)



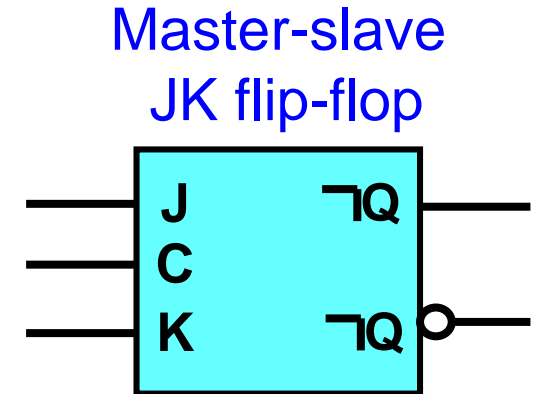
D Type Master-Slave Flip-Flop (5/5)



JK Flip-Flops

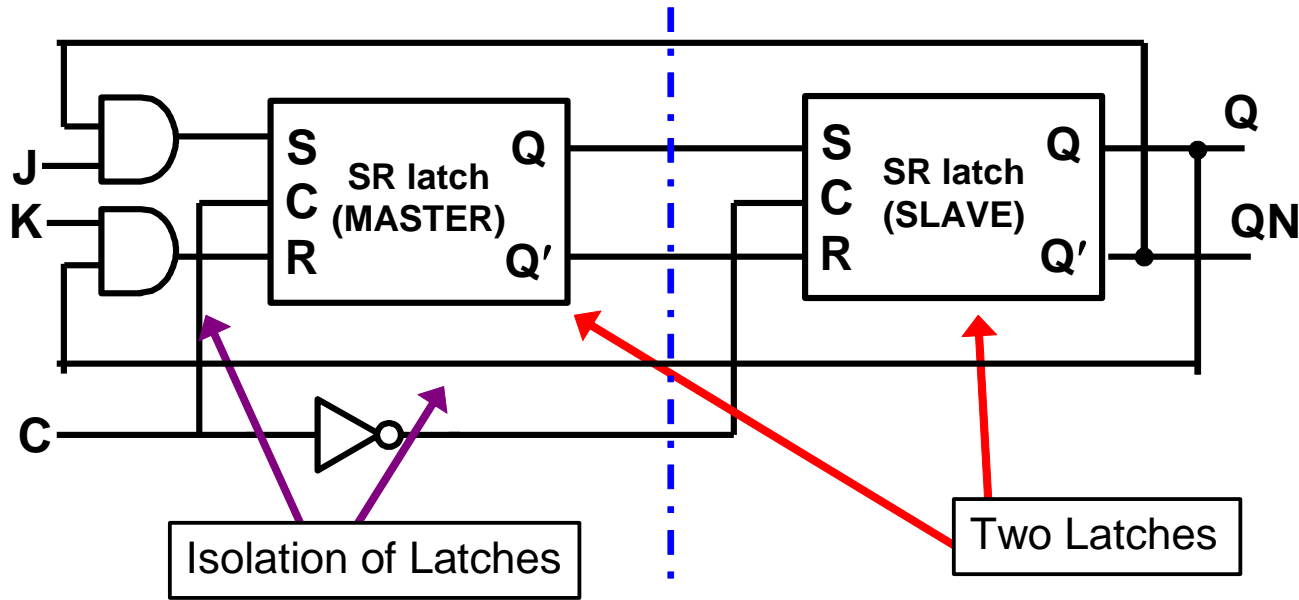
- **JK Flip-Flop:**

- Uses 2 SR latches with control inputs.
 - The SR latch still **has undesirable indeterminate next state** (when **S** and **R** are both **1**).
- Adds **AND** gates to **S** and **R** inputs with feedback from the SR slave latch output.
 - Now when **S** and **R** inputs are **1**, the output of the flip-flop merely complements.
- Uses **narrow pulse clocking** approach.

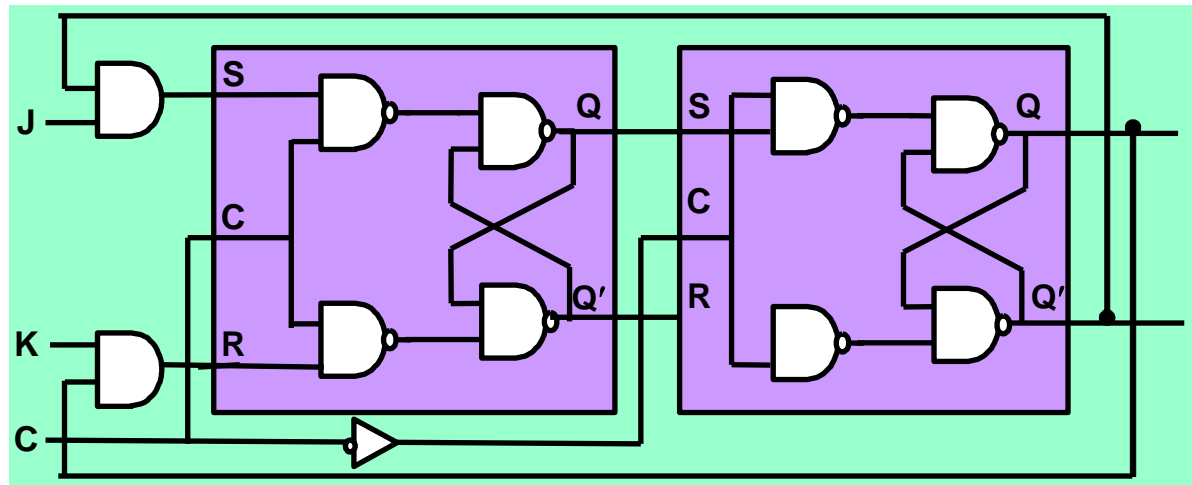


Minimum pulse width for inputs, to avoid going into a metastable state.

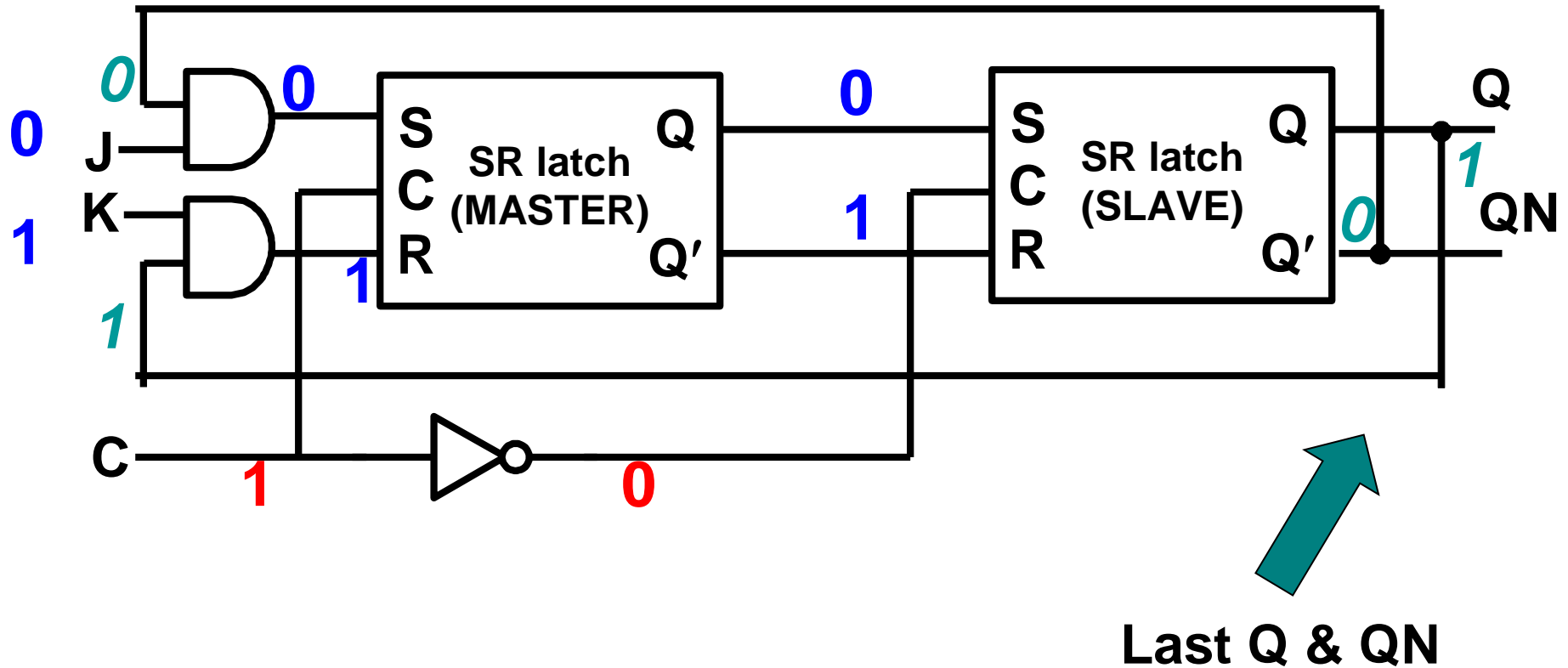
JK Type Master-Slave Flip-Flop



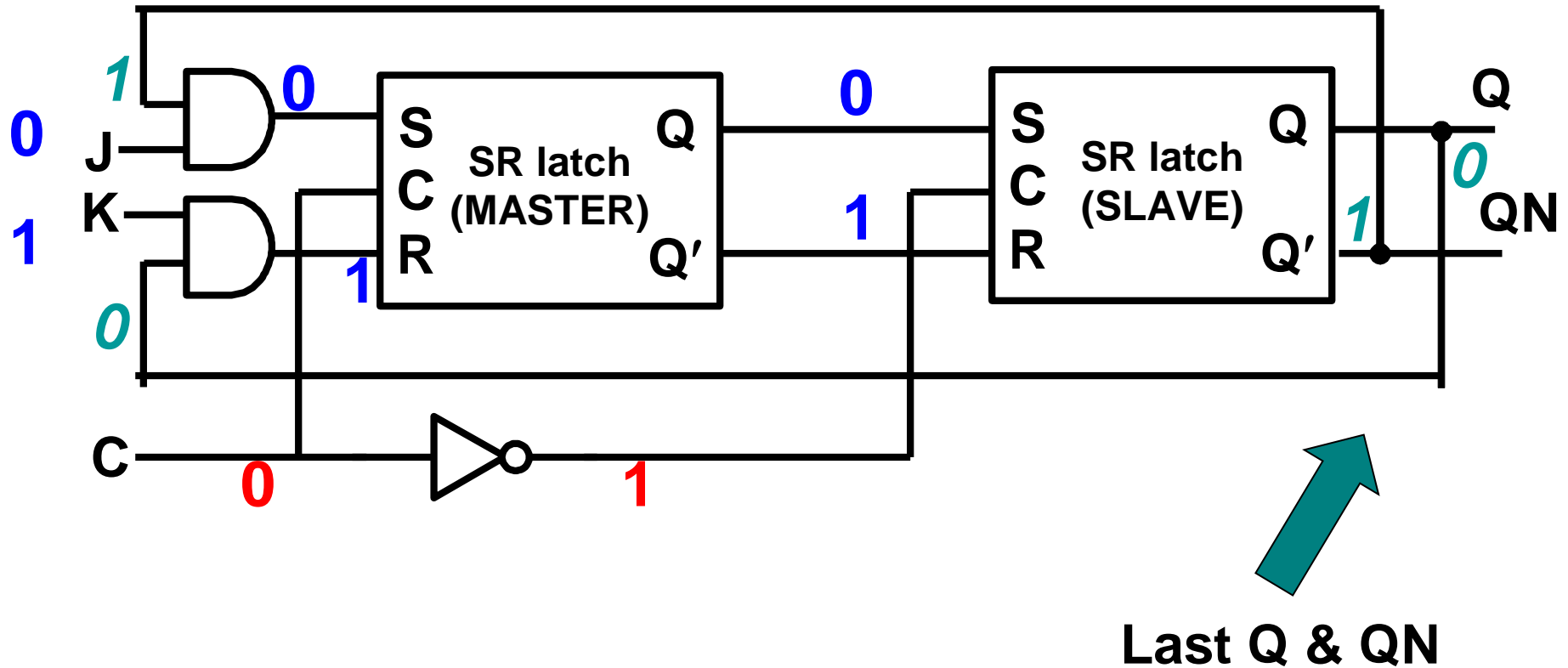
Question for you: What do we need to do to set this flip-flop?



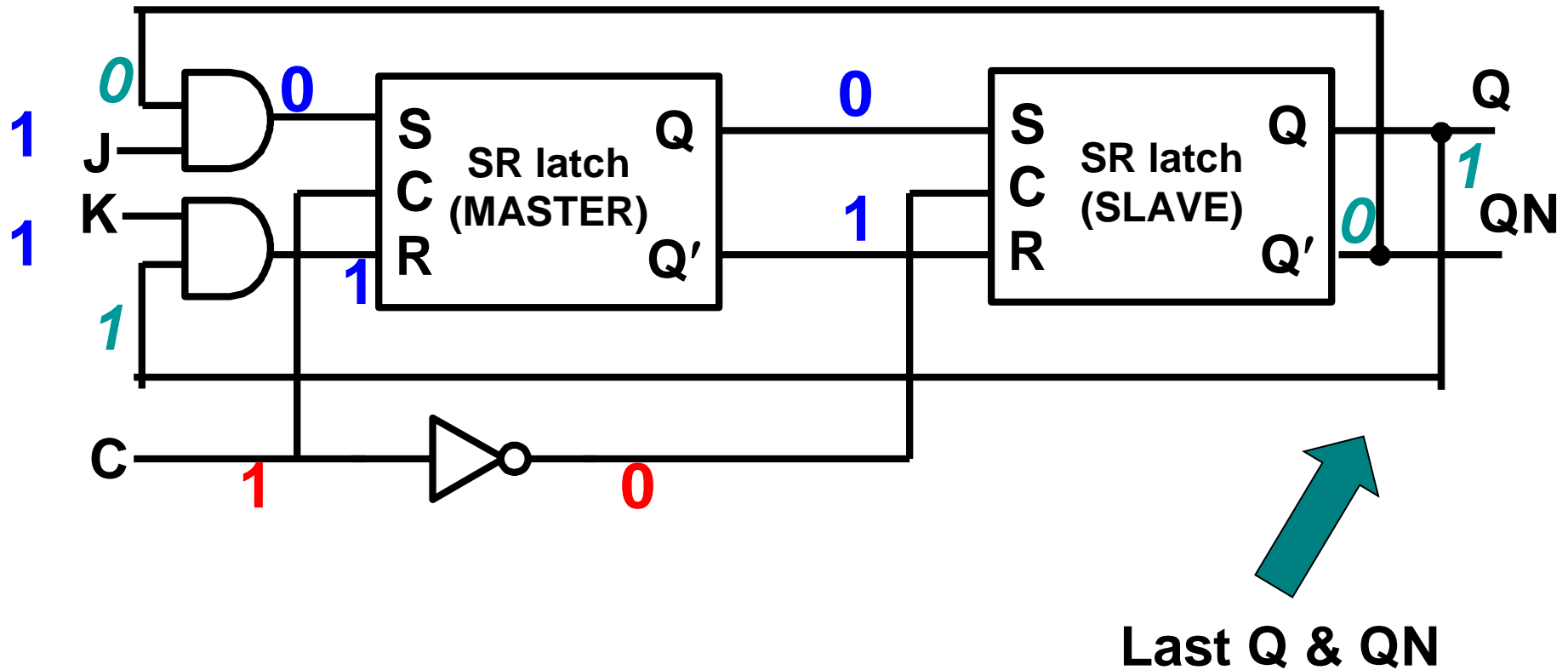
Master-Slave JK Flip-Flop: $J=0$, $K=1$ (1/2)



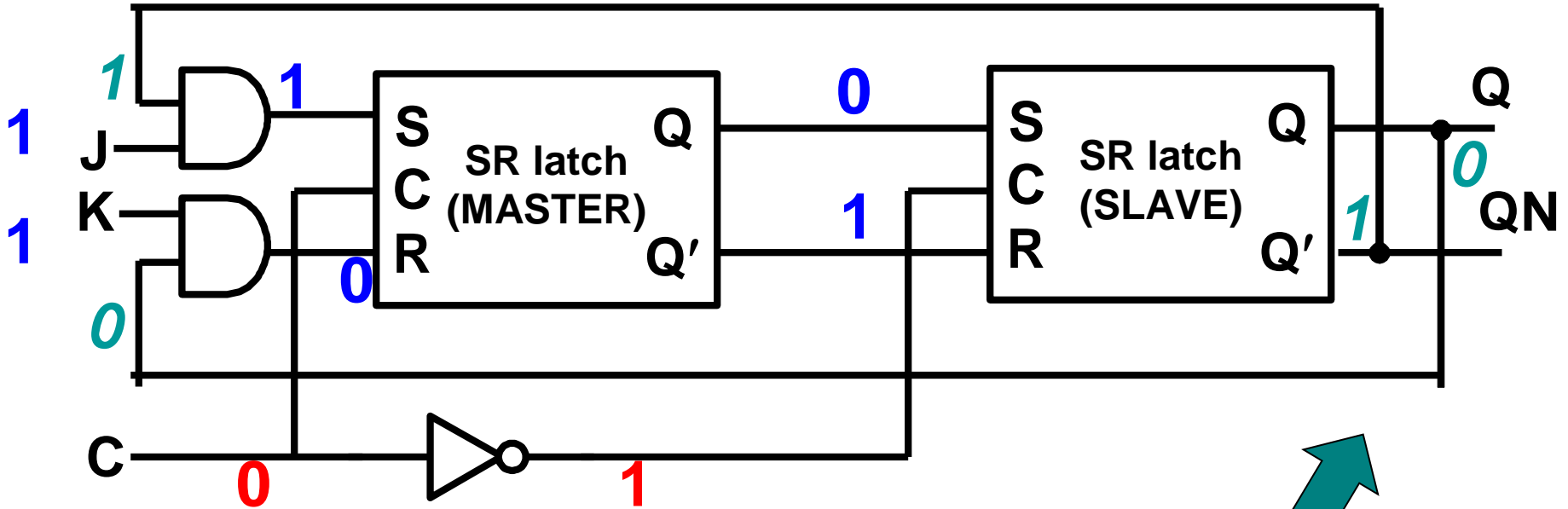
Master-Slave JK Flip-Flop: $J=0, K=1$ (2/2)



Master-Slave JK Flip-Flop: $J=1, K=1$ (1/2)



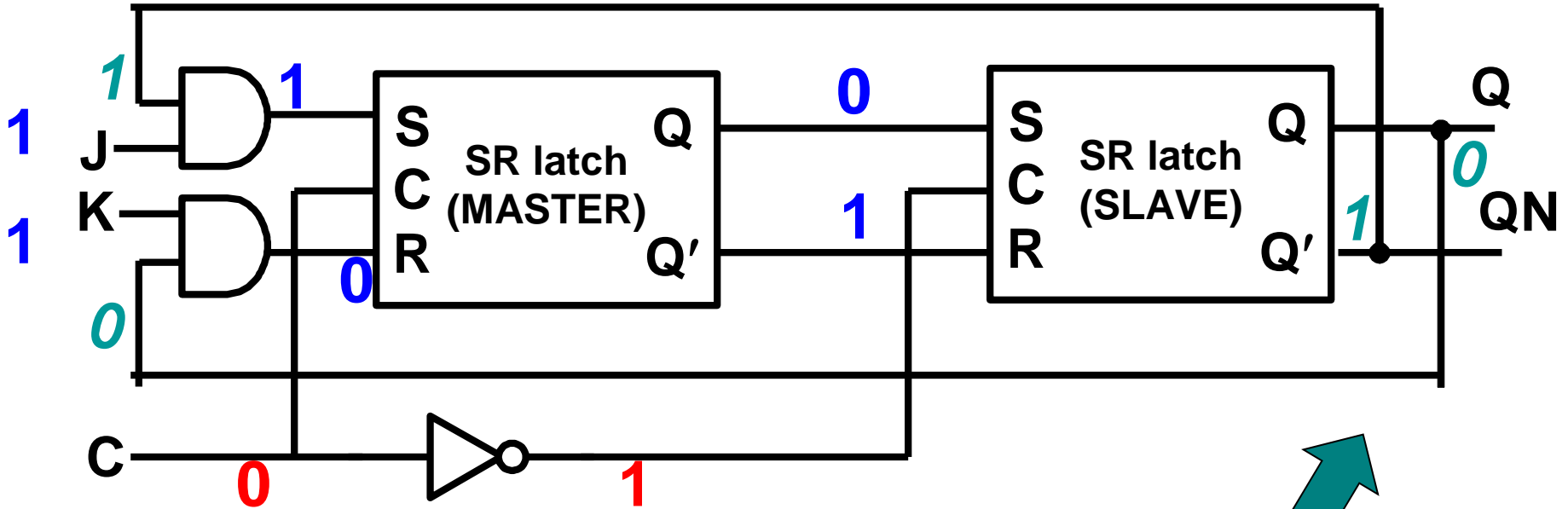
Master-Slave JK Flip-Flop: $J=1, K=1$ (2/2)



J	K	C	Q	QN
x	x	0		
0	0	0		
0	1	0		
1	0	0		
1	1	0		

Last Q & QN

Master-Slave JK Flip-Flop: $J=1, K=1$ (2/2)

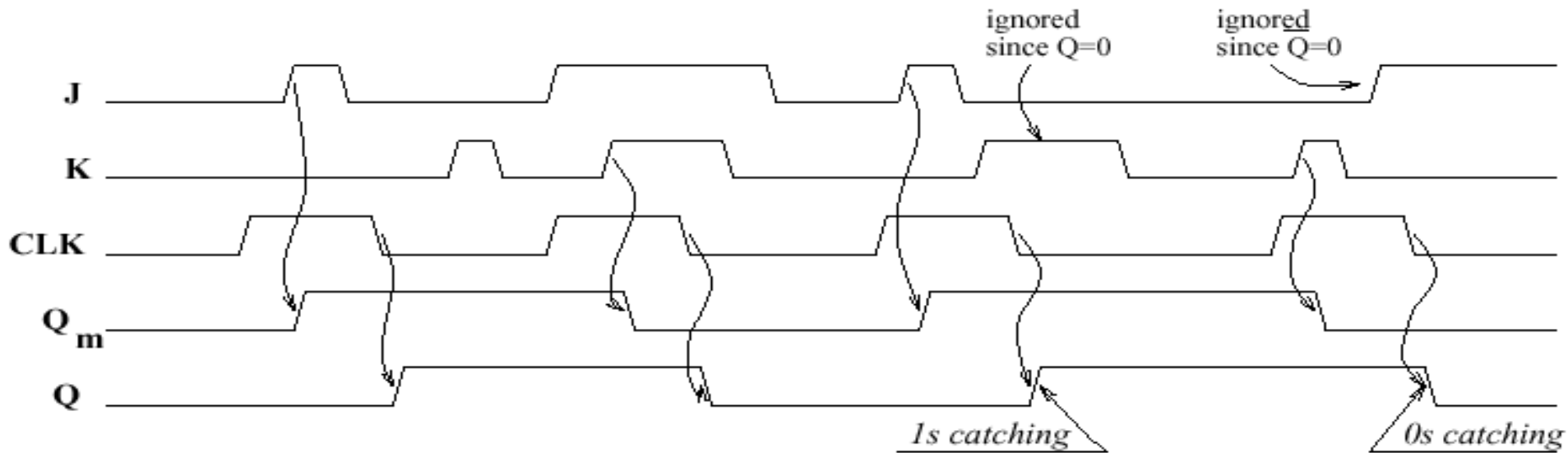


J	K	C	Q	QN
x	x	0	last Q	last QN
0	0	0	last Q	last QN
0	1	0	0	1
1	0	0	1	0
1	1	0	last Q	last QN

Last Q & QN

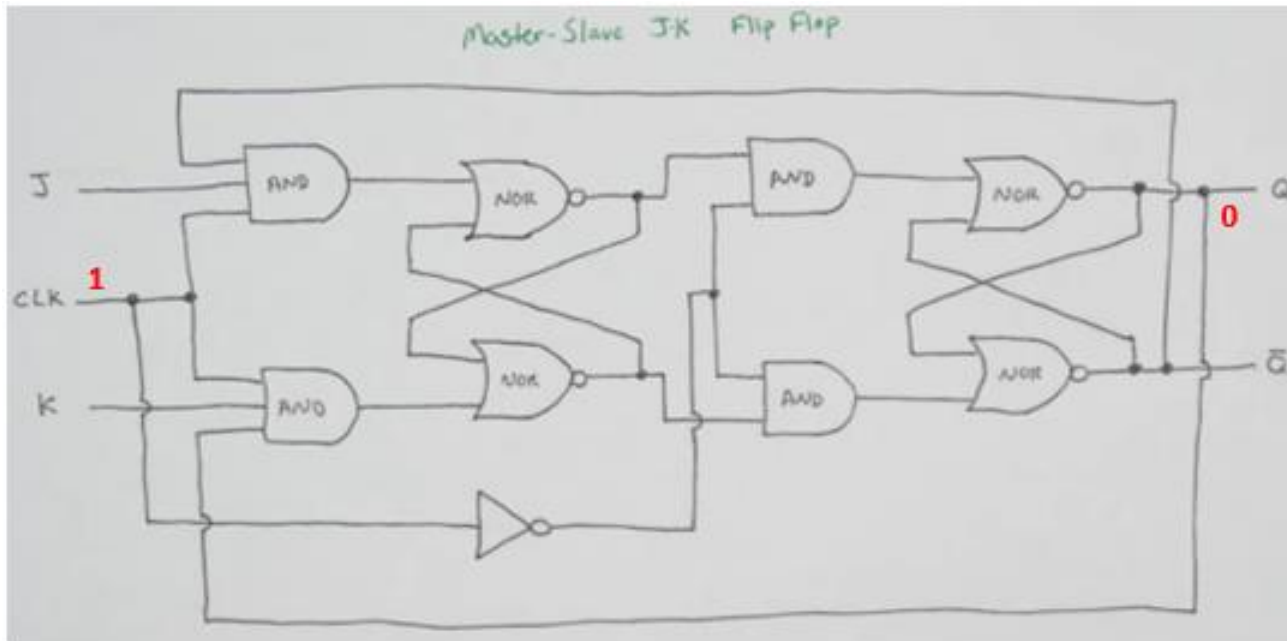
Master-Slave JK Flip-Flop: Problems

- If inputs **J** and **K** are not held valid during the entire period when CLK is active for the master, the above flip-flop exhibits **1's** and **0's catching behaviour**.
 - **1's Catching**: Output changes to **1** even though **K** and not **J** is asserted at the end of the triggering pulse.
 - **0's Catching**: Output changes to **0** even though **J** and not **K** is asserted at the end of the triggering pulse.



Master-Slave JK Flip-Flop: Problems

- **HOMEWORK:** Consider the circuit shown below. Assume that J and K are both low. Analyse the circuit to determine the output of the Master and Slave flip-flop.
- Now consider that J goes high briefly due to a glitch on this line. Following this the clock goes low. Analyse the circuit once again to understand the effect.



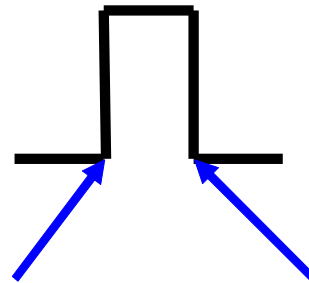
Edge-Triggered Flip-Flops

- **Edge-triggered flip-flops:**

- **Ignore inputs** while the clock pulse is at a constant level.
- Only **set outputs** on clock pulse transitions.

- *Negative or positive edges:*

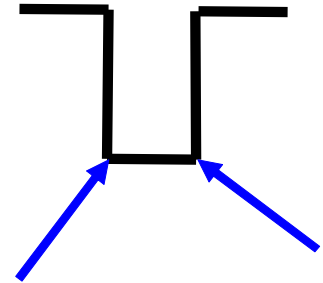
Positive Pulse



Positive Edge

Negative Edge

Negative Pulse





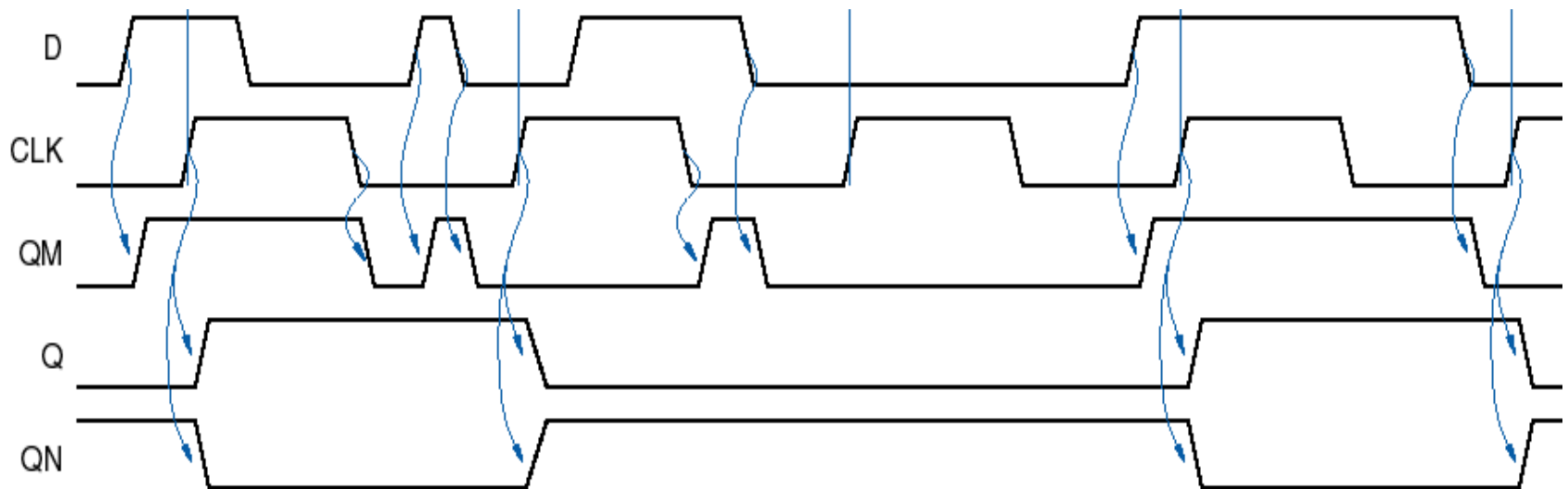
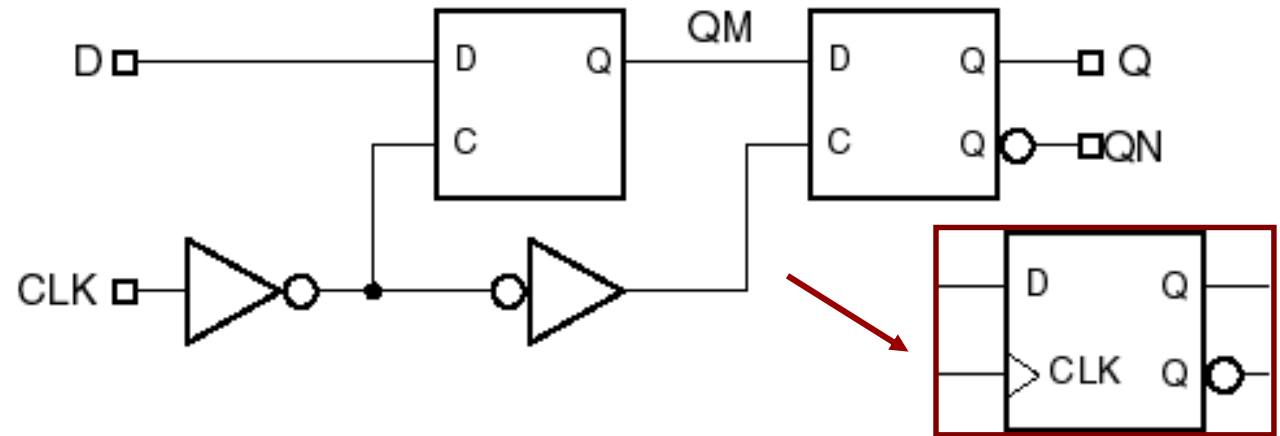
Negative Edge

Positive Edge

- For **positive triggered flip-flops**, the value at the **D** input is transferred to **Q** on a positive transition.

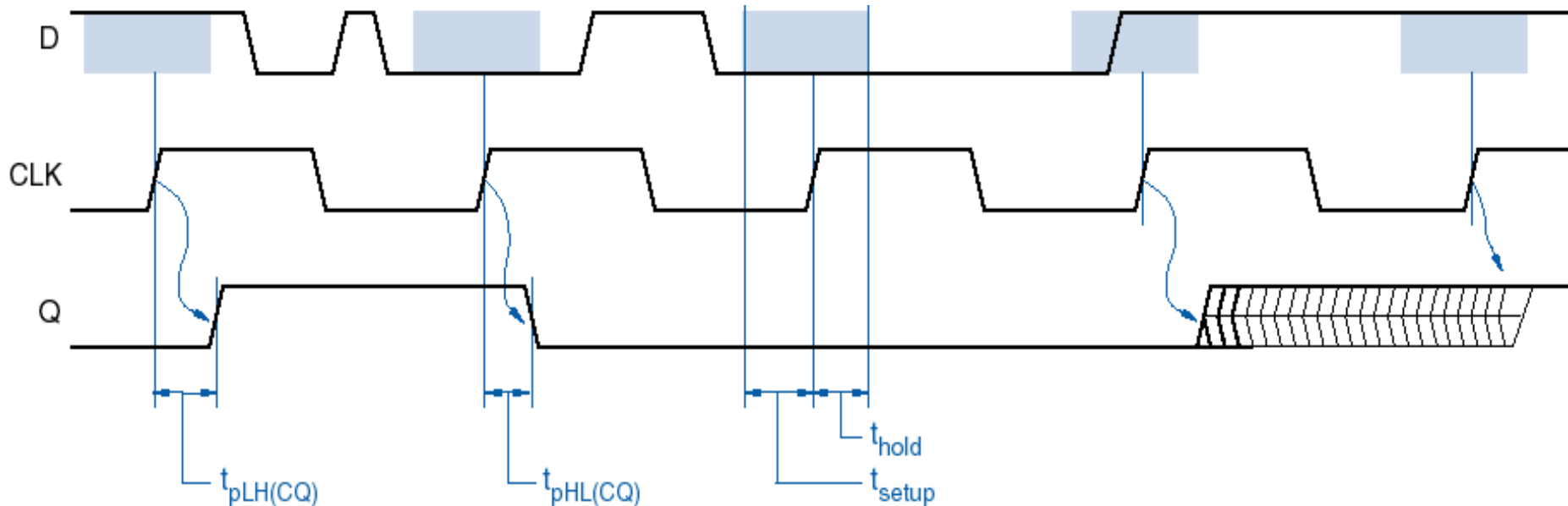
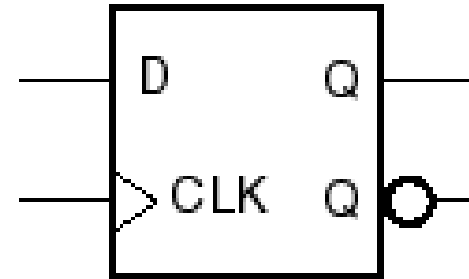
Positive Edge-Triggered D Flip-Flop

D	CLK	Q	QN
0		0	1
1		1	0
x	0	last Q	last QN
x	1	last Q	last QN





D Flip-Flop: Timing Parameters

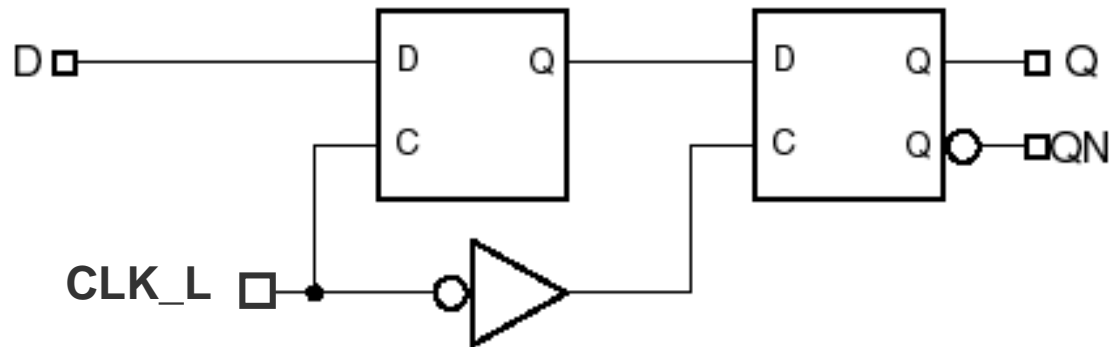
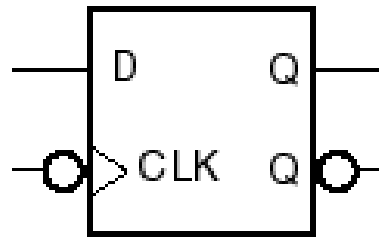
- Timing parameters:
 - **Propagation delay** (from CLK).
 - **Setup time** (D before CLK).
 - **Hold time** (D after CLK).



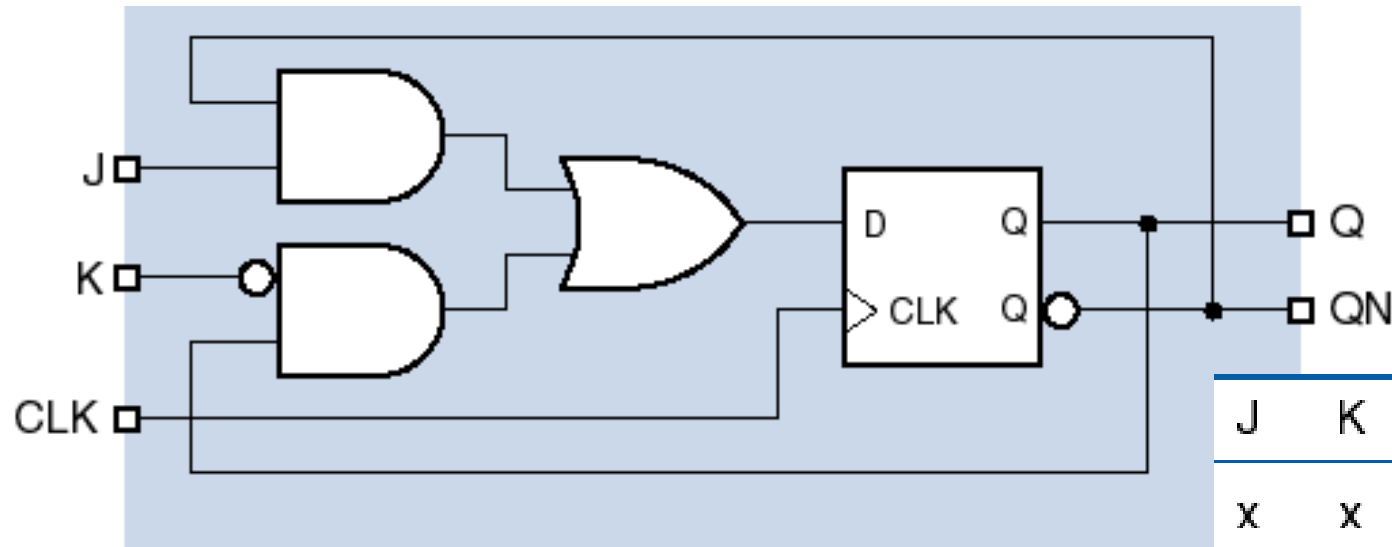
Other D Flip-Flop Variations

- **Negative-edge triggered:** By **inverting the clock input**, all the action takes place on the falling edge of CLK_L (active low).
- It is the same as the first *master-slave flip-flop* we looked at!

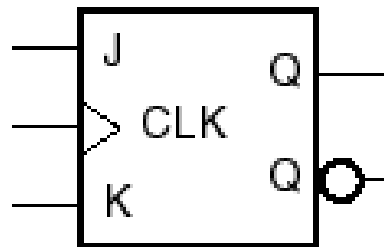
D	CLK_L	Q	QN
0		0	1
1		1	0
x	0	last Q	last QN
x	1	last Q	last QN



Edge Triggered JK Flip-Flop

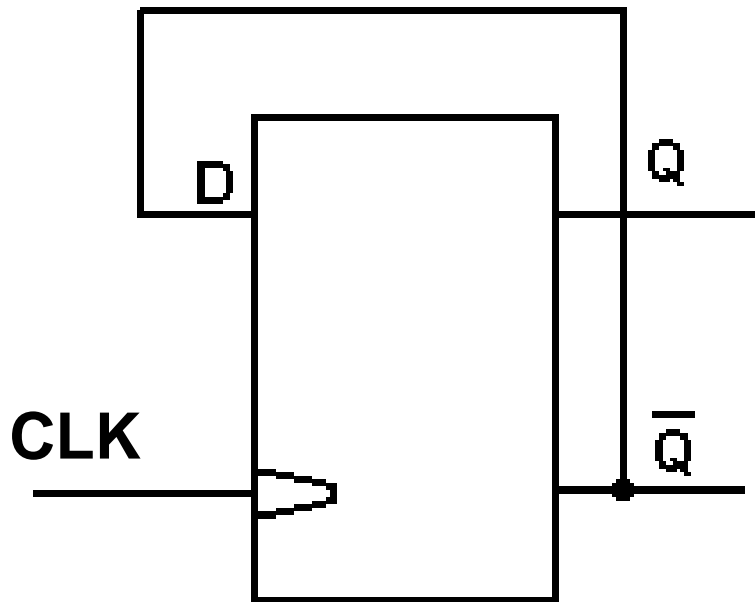


Solves the 1's and 0's catching problem of the master-slave!



J	K	CLK	Q	QN
x	x	0	last Q	last QN
x	x	1	last Q	last QN
0	0		last Q	last QN
0	1		0	1
1	0		1	0
1	1		last QN	last Q

Toggle flip-flop (T-Flip-Flop)

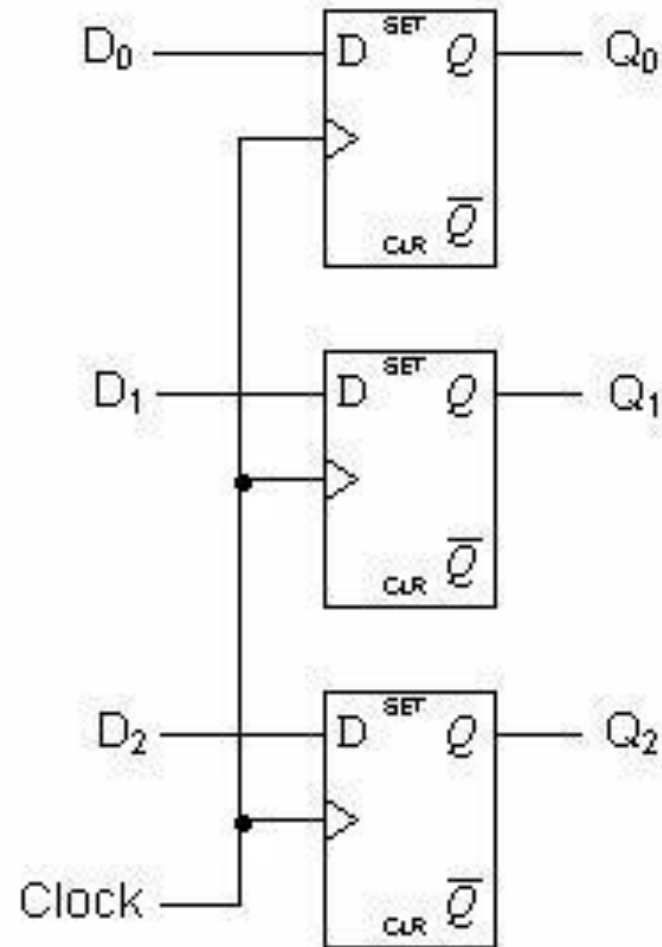


- Using **T-Flip-Flops** the next state output is always *toggled*.
- When the **D-type flip-flop** is triggered by the clock the **next Q** output will be the ***complement of the current Q***.

Applications of Flip-Flops (1/2)

- **Parallel Data Storage:**

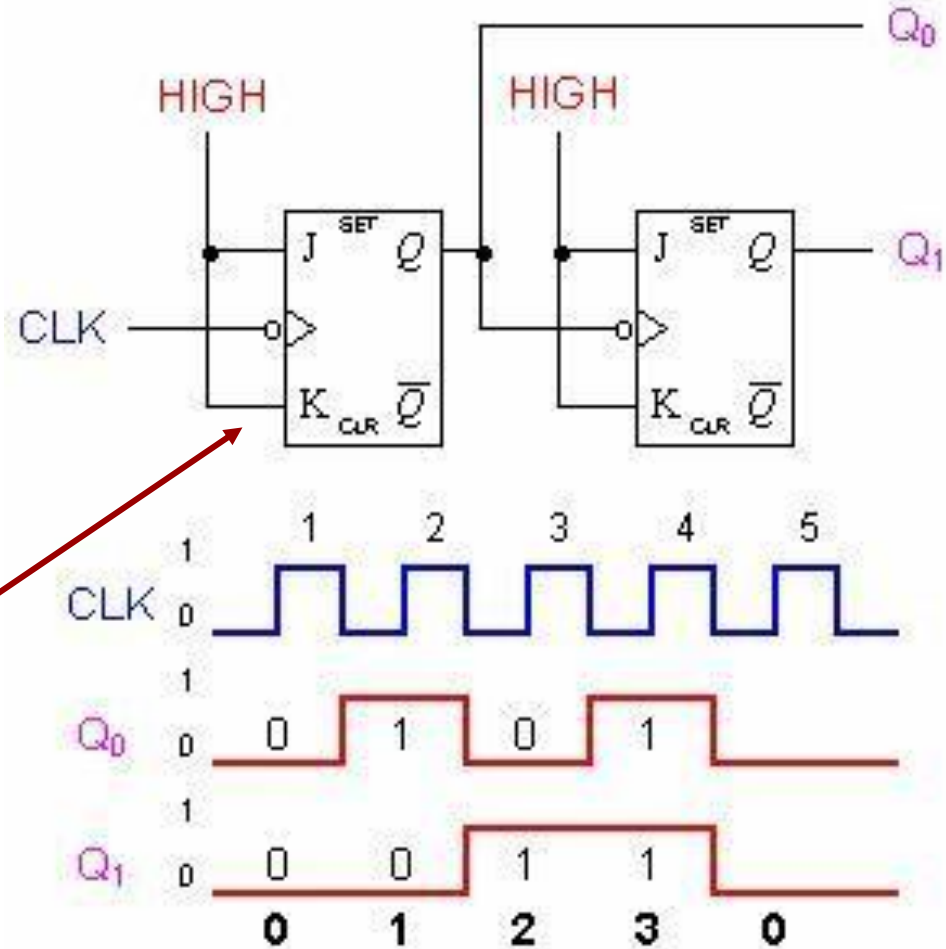
- In digital systems, data is normally stored in groups of bits to represent e.g., numbers or codes.
- We can take several bits of data on parallel lines and store them simultaneously in a group of flip-flops (**Register**).



Applications of Flip-Flops (2/2)

- **Digital Counters:**

- The 2-bit binary sequence is repeated every 4 clock pulses.
- When it reaches the value 3, it resets back to 0 and the sequence begins again.



You can use **Toggle Flip-Flop** instead