

# **Machine Structure 02**

## **Chapter 04**

### **The sequential circuits**

# chapter plan

- 1. Introduction**
- 2. Flip-flops**
- 3. Counters**
- 4. Registers**

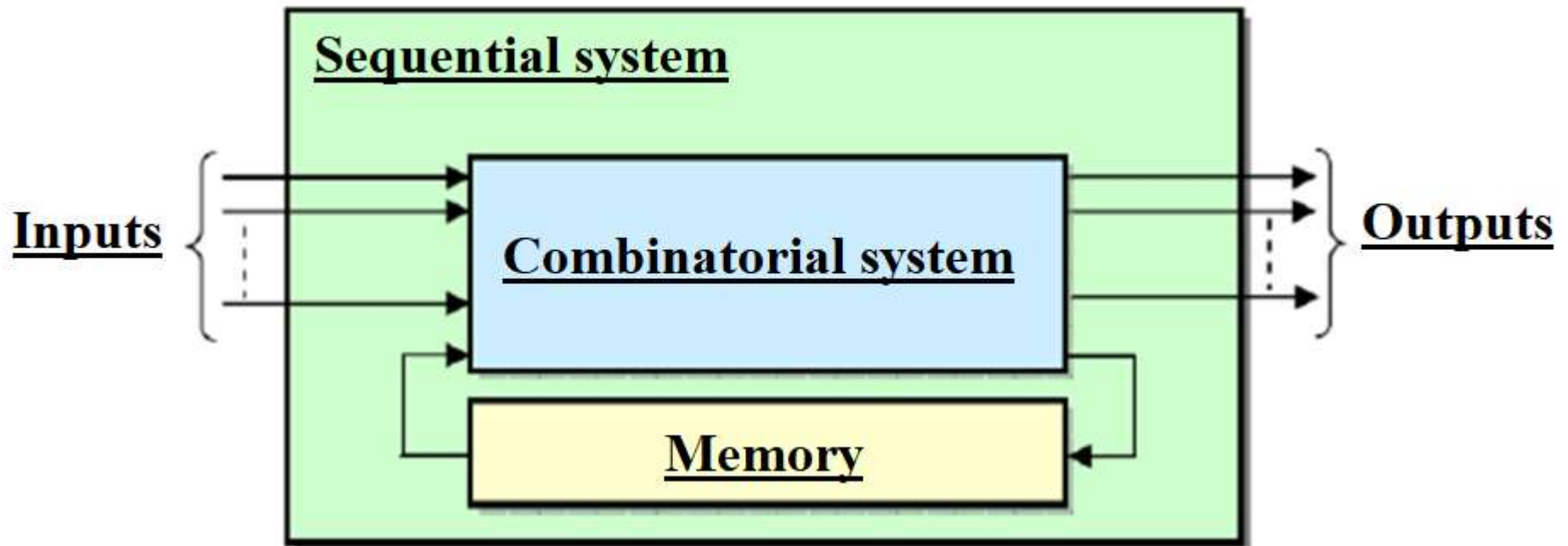
## Introduction:

Several circuits used in everyday life need memory. Thus, A sequential system is a logical system whose state of the output variables depends **on the state of the input variables + the previous state of the output variables.**

The system remembers the past by recording the previous states of its outputs, calling for this, **internal variables, or memories.**

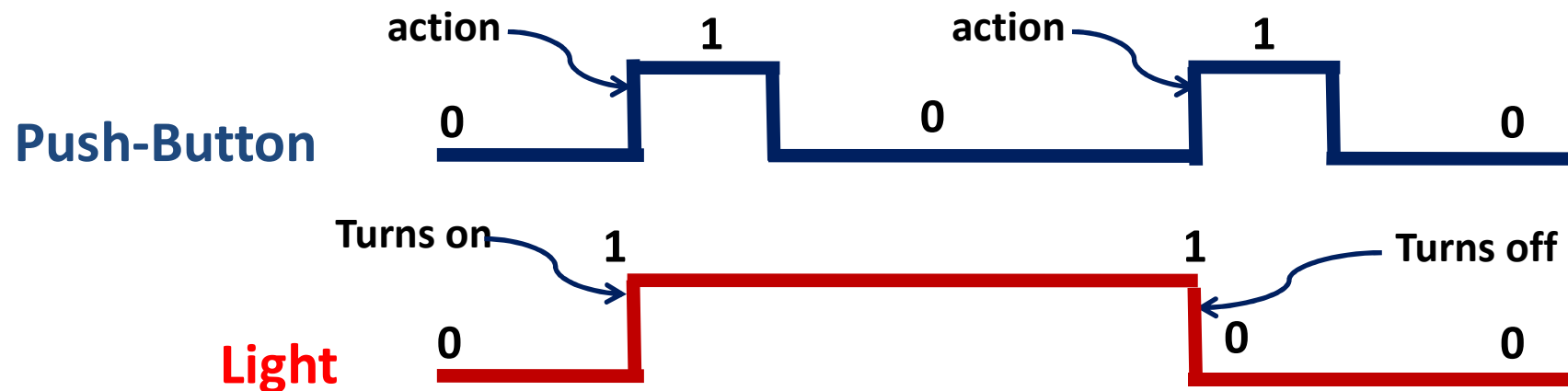
# Introduction:

(A sequential circuit is a digital (logic) circuit whose state at **time t+1** is a function of the inputs at the same **time t+1** and of the previous state of the system (**time t**).



## Concept of state:

By way of illustration, let us take the example of an electric lamp controlled by a **push button** in such a way that an action on the button turns on the lamp, and that a successive action turns it off, according to the example of the following timing diagram :



## Concept of state

The position (actuated or not), is not easily identifiable by the operator; an indicator light is often added to them, indicating the state of the circuit.

It should be noted that the **evolution of the system** depends not only on the position of the push button at a given moment, but also on whether the lamp is **on** or **not**.

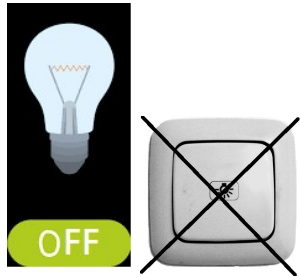
**So:** The system therefore depends on the **previous state** because it retains the memory of the previous action; this is the **essential** characteristic of a sequential system.

# Concept of state

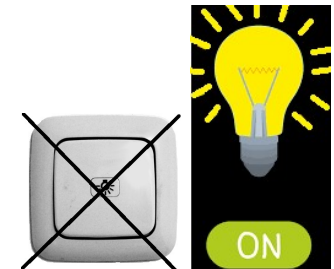
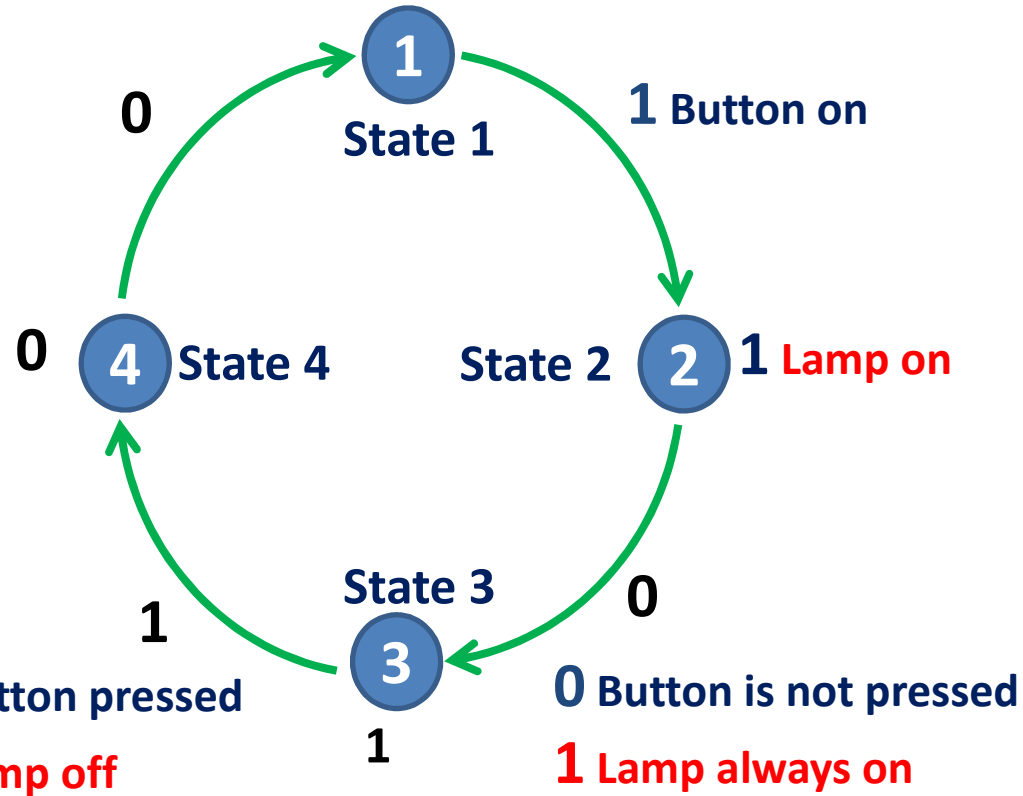
## Phases diagram

0 Button is not pressed

1 Lamp always off



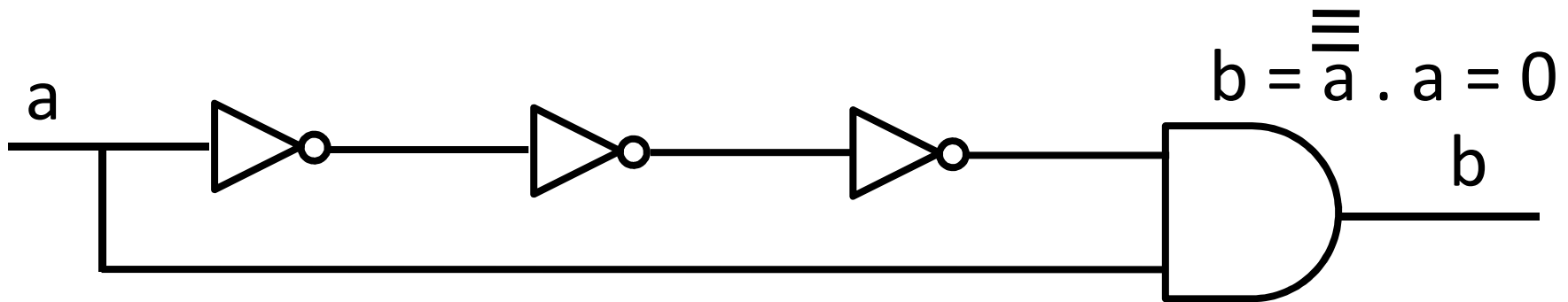
1 Button pressed  
0 Lamp off



## Signal propagation delay :

The design of sequential logic circuits takes into account a property that we ignored in combinatorial circuits. This is the propagation delay of the electrical signals through the logic gates.

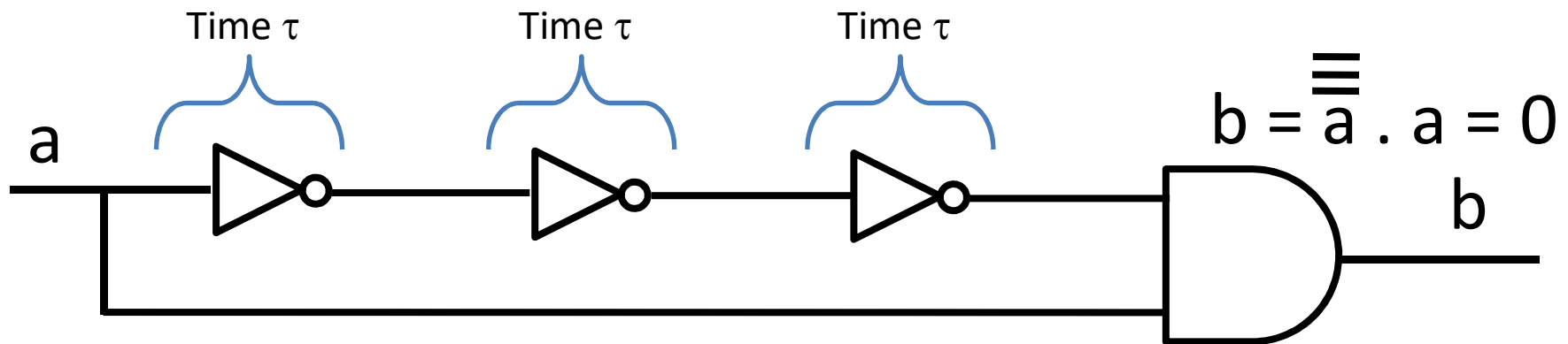
Example :





## Signal propagation delay :

Indeed, we assumed that as soon as a change occurs in the inputs of a combinatorial circuit, **immediately** the outputs are affected. In reality, each **logic gate** will always take a **certain time** (called **propagation time = Time  $\tau$** ) to pass an electrical signal from its input to its output.



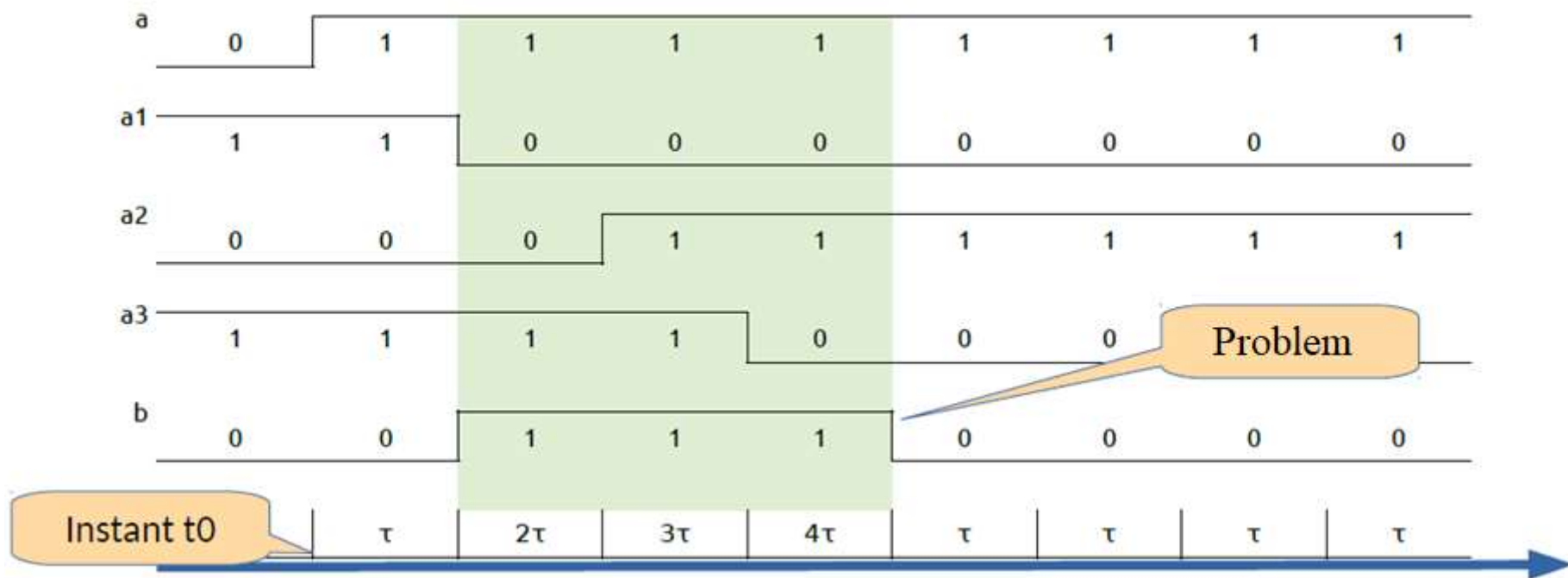
In principle, based on the combinatorial logic circuits, the output of this circuit should always be equal to **0**.

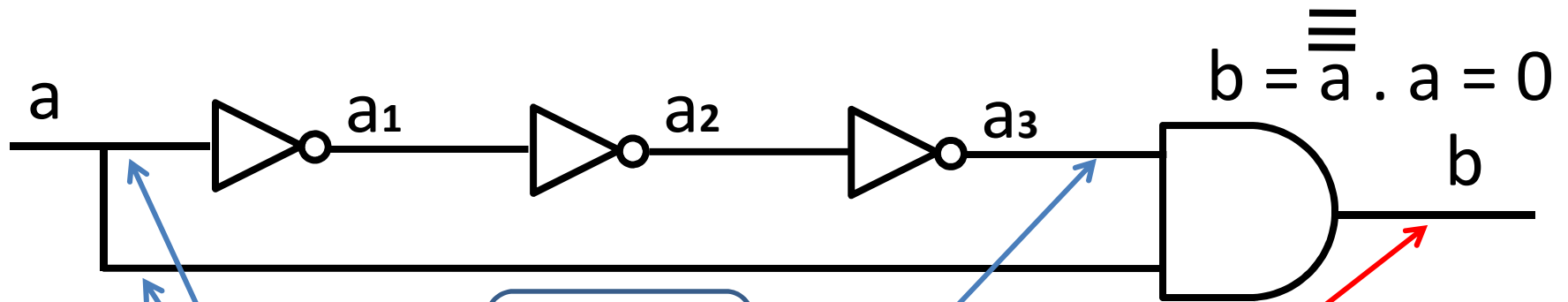
You will see, in reality that there is a small **randomness** (problem) that occurs when we change the value of the input "a".

Suppose that each of the gates (here "NOT" and "AND") takes a time equal to  $\tau$ .

Let's plot in a temporal graph (which we call **chronogram**) the evolution of our signals: (a, a1, a2, a3 and b).

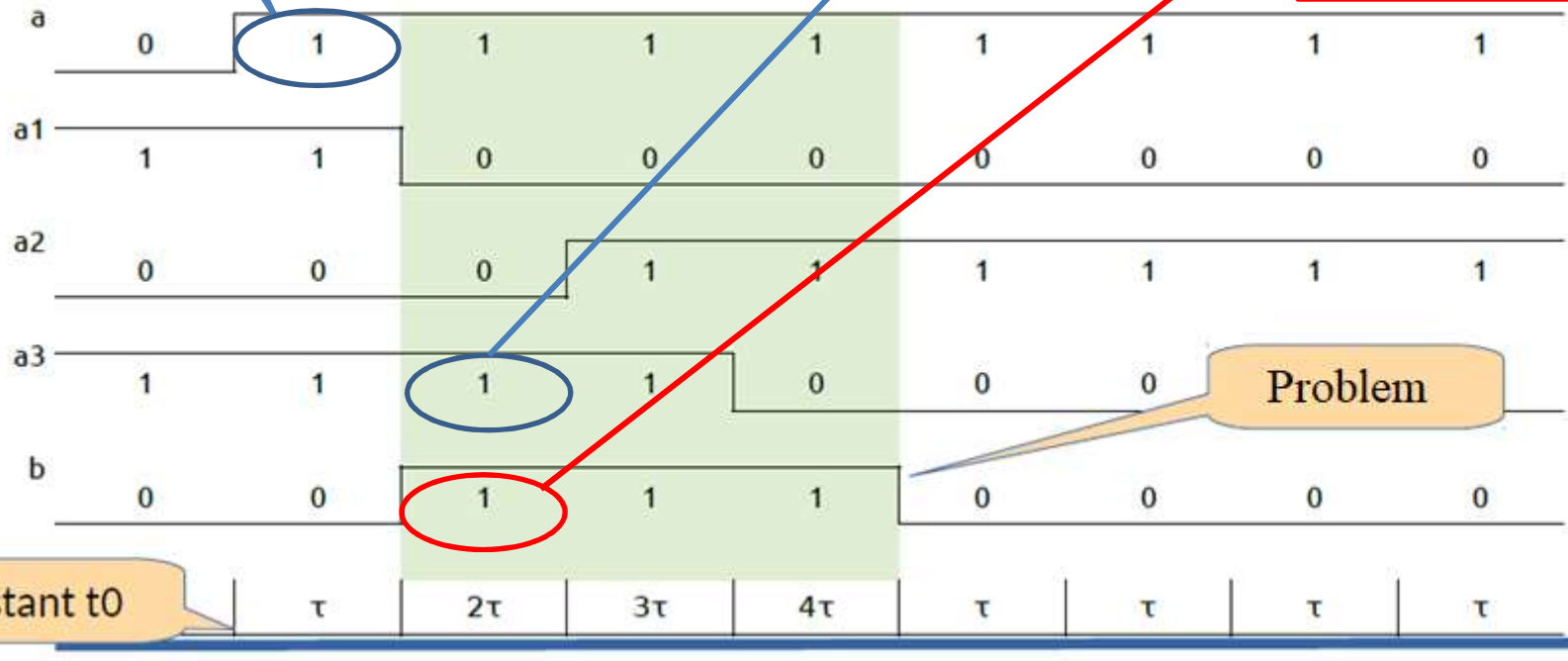
**Note:** The time diagram indicated below is called "**chronogram**". It is often used to show the functions of sequential logic circuits.



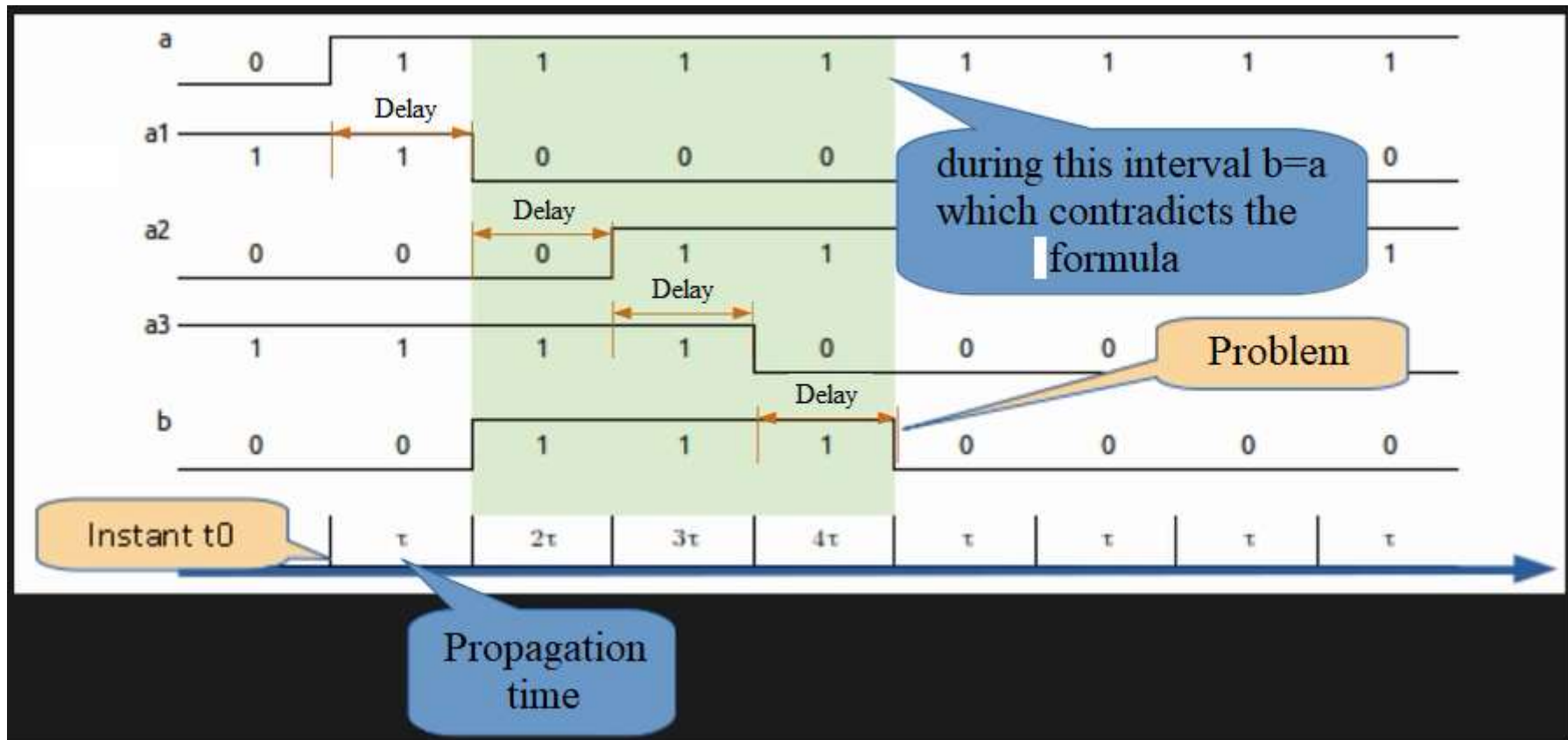


always retain the previous value 1

$b = 1 \cdot 1 = 1$



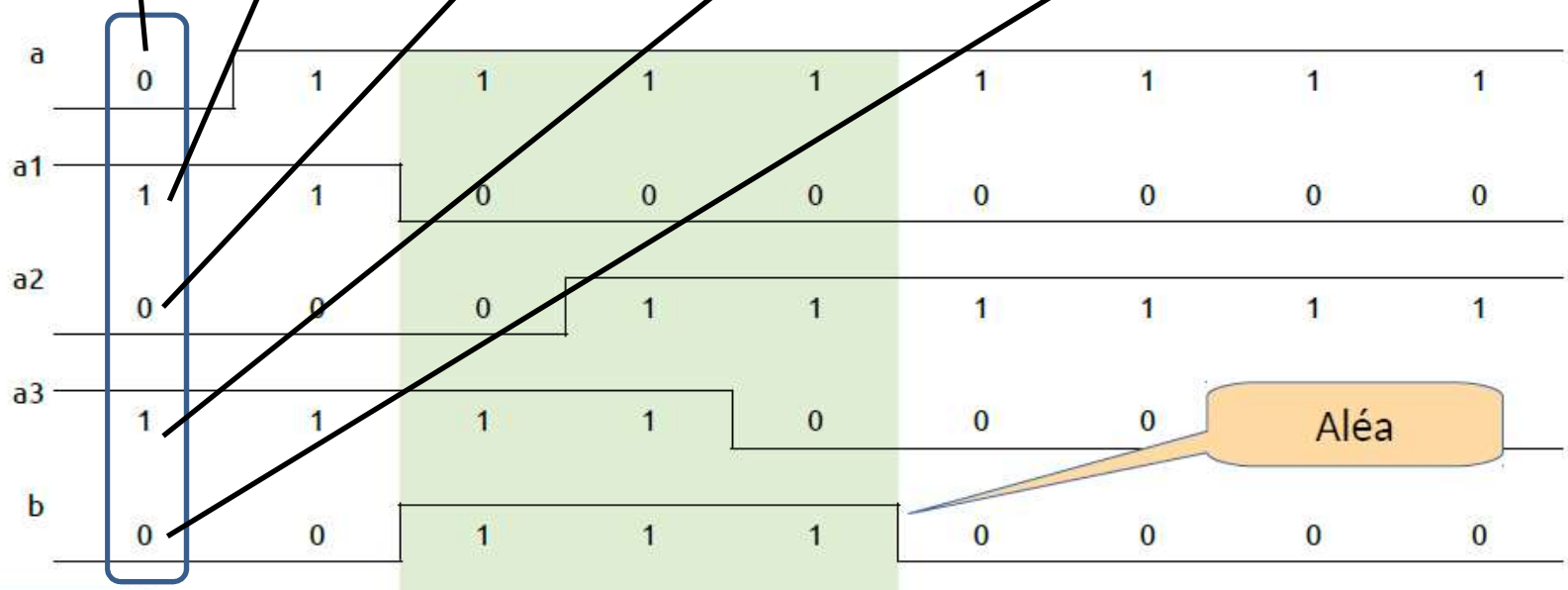
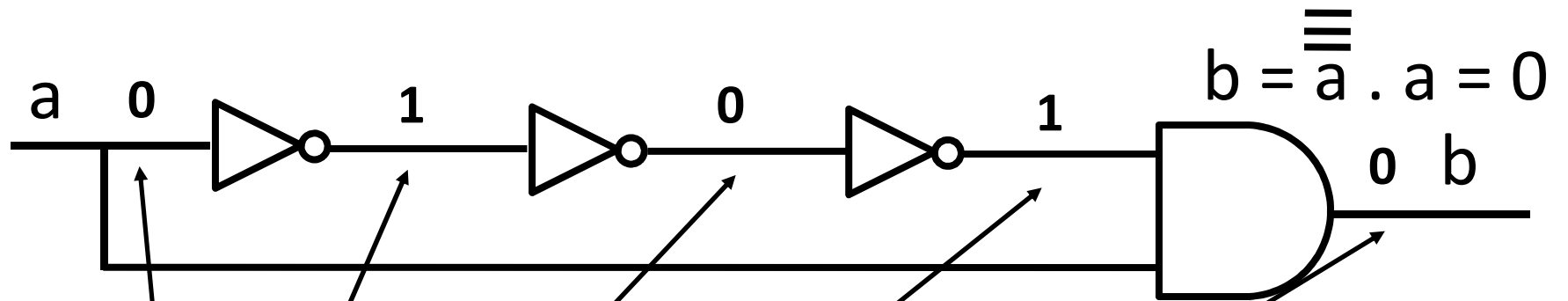
Problem

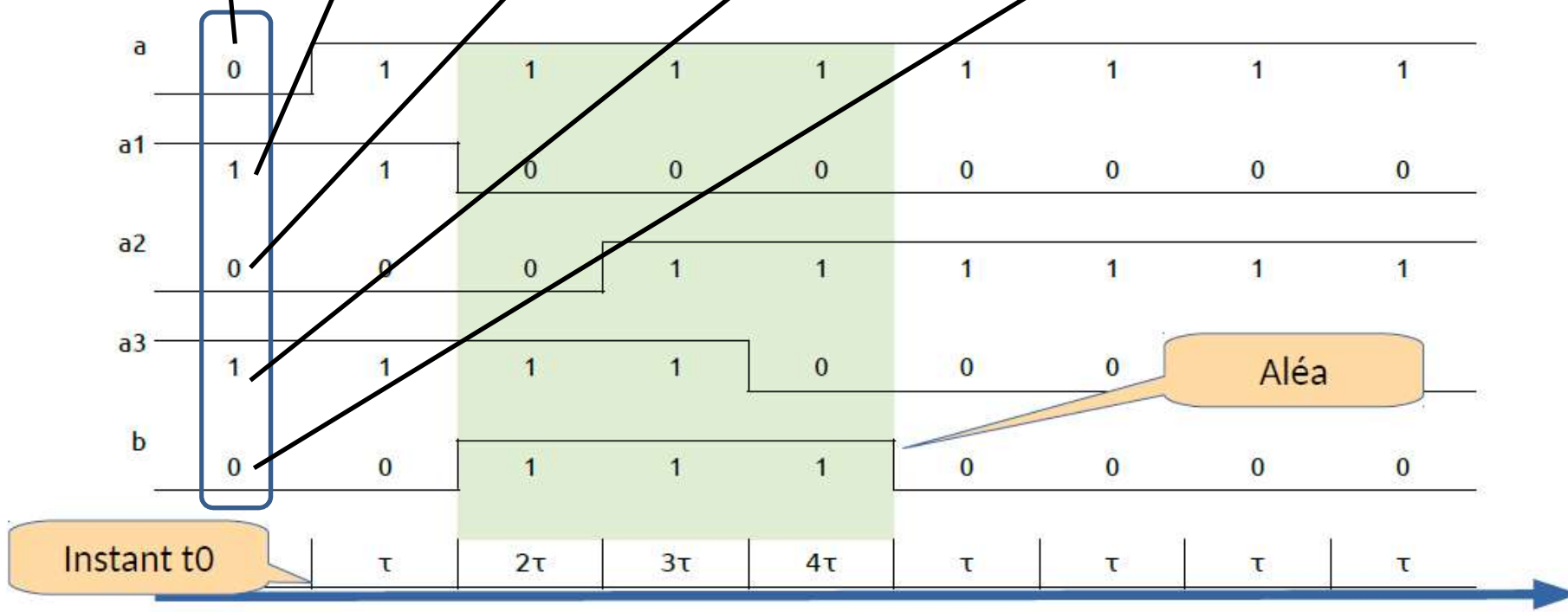
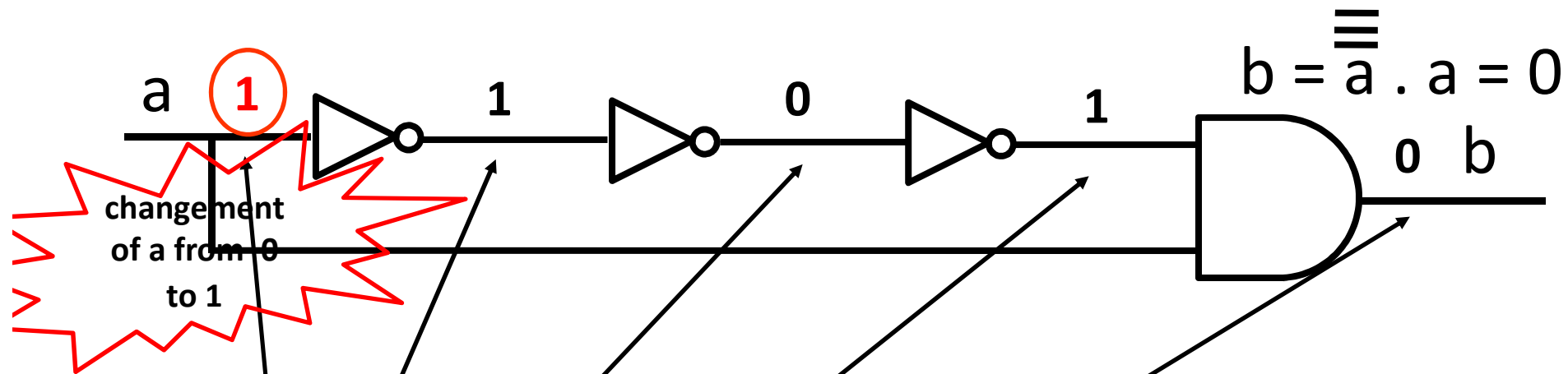


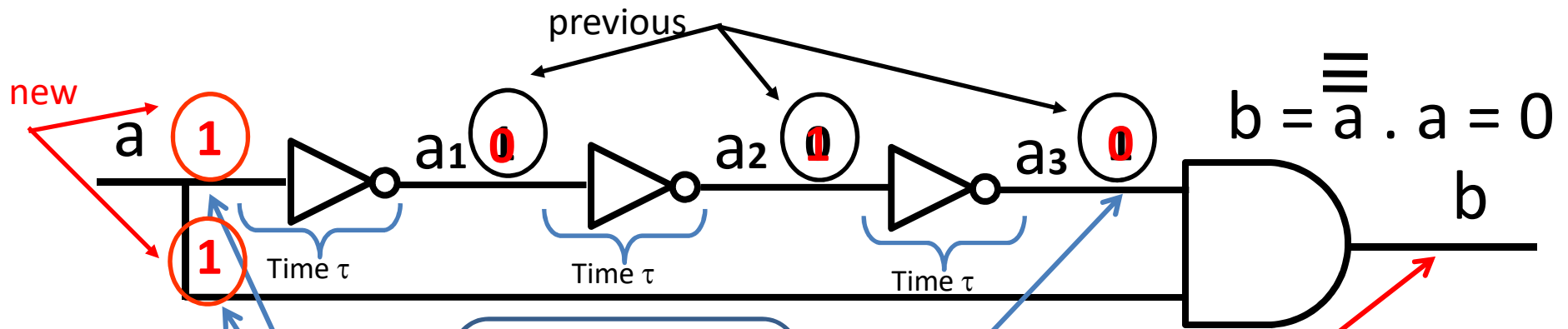
Let's assume that the value of 'a' switches from:

'0' to '1' at time  $t_0$ .

- At time  $(t_0 + \tau)$ , 'a1' reverts to 0 (thus 'a1' is the inverse of 'a').

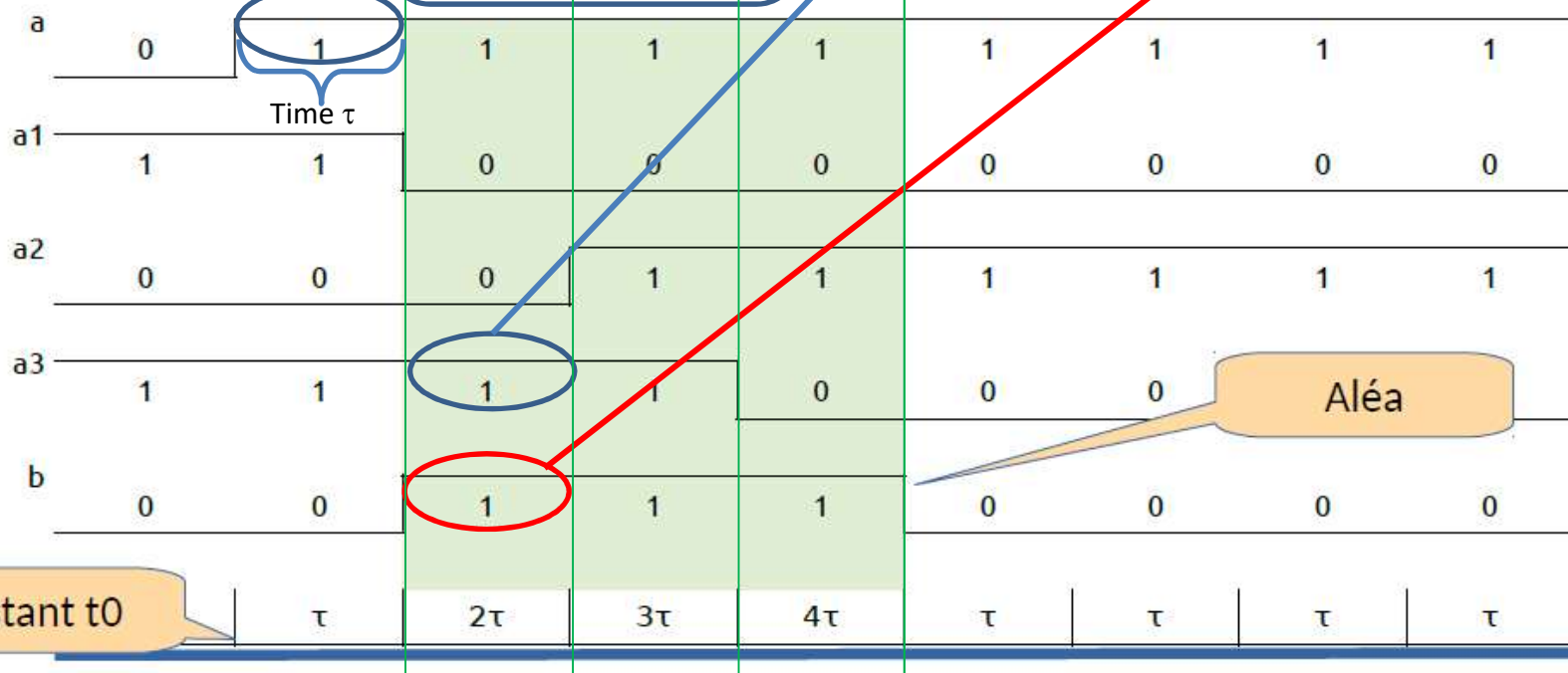






Before the end of the Time ' $\tau$ ', the other values a1, a2, a3, and b always retain their previous values.

**b = 1.1 = 1**





- At time  $(t_0+2\tau)$ , 'a2' transitions to 1 (then 'a2' is the inverse of 'a1').
- At time  $(t_0+3\tau)$ , 'a3' transitions to 0 (then 'a3' is the inverse of 'a2').
- It should be noted that between  $(t_0)$  and  $(t_0+3\tau)$ , 'a' and 'a3' are both at '1'
- simultaneously, which creates a particular situation for 'b' being equal to '1'.

## Types of sequential circuits:

There are two types of sequential circuits:

**Synchronous:** where the behavior is determined by signals at discrete moments (measured, moderated) of time (clocked at the rhythm of a signal called '**clock**' or **CLK**);

**Asynchronous:** where the behavior is determined by signals at any moment, and the order in which inputs change. Thus, in:

**Asynchronous circuits:** the output is modified as soon as there is a change in the state of the inputs.

**In synchronous circuits:** the output changes only after a clock signal.

## Synchronous System (Concept of Clock)

Synchronization is achieved using a clock, which provides a **periodic signal**.

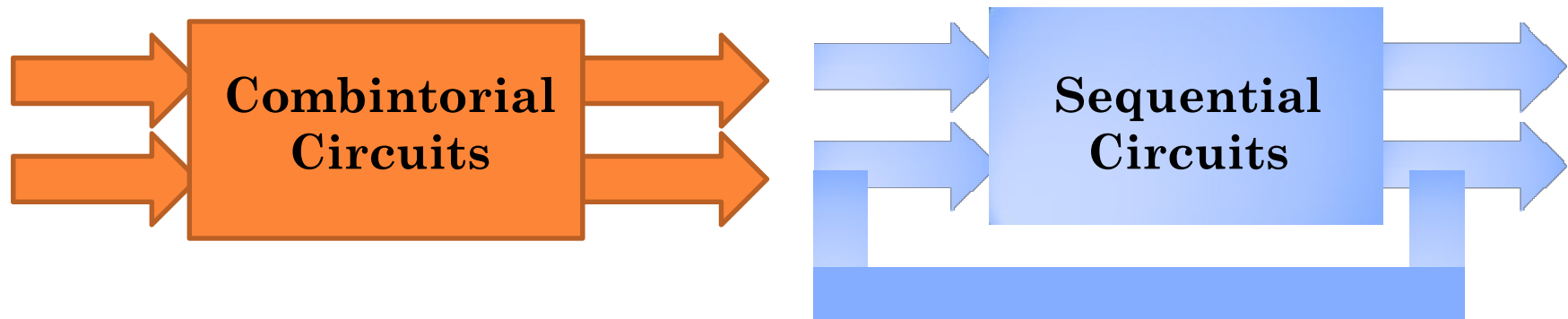
This periodic signal is distributed throughout the system so that **storage elements** are only affected with the **arrival of a clock pulse**.

The clock determines when there is activity in the circuit, and the other signals determine what (i.e., what changes) in the circuit.

The storage elements controlled by the clock transition are called **flip-flops**

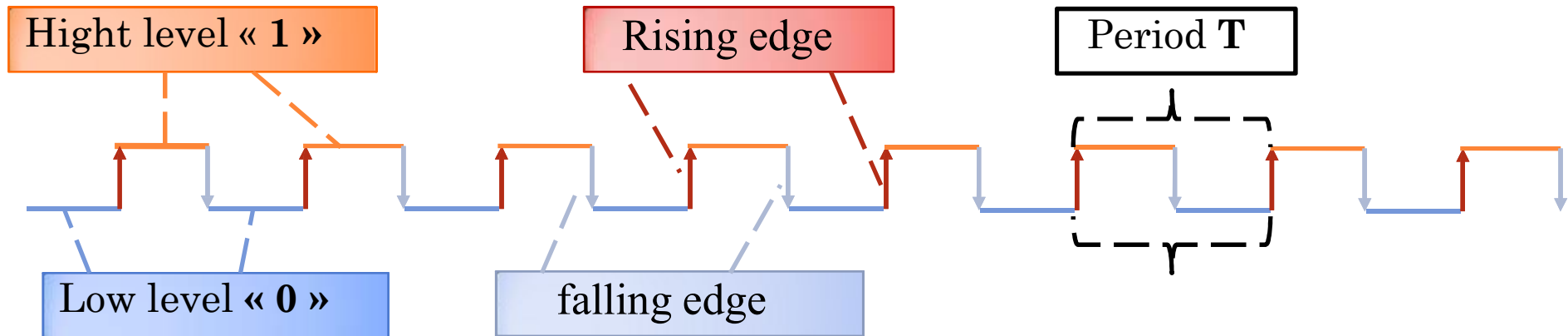
- The **flip-flop** is the basic element allowing the construction of sequential circuits (**Memory, Register, Counter**, etc.).
- A **clock** is a **logical variable** that successively transitions from **0 to 1** and from **1 to 0** periodically.
- This variable is often used as an input in sequential circuits (the clock is denoted by **h** or **ck** (clock)).

- **Introduction to Sequential Circuits:**
- **Combinational Circuits:** Output functions are expressed based on logical expressions of **only** the input variables.
- **Sequential Circuits:** Output functions depend not only on the current state of the input variables but also on the **previous** (past) state of certain output variables (memory properties).



## The concept of Clock

A clock, noted by H or ck (clock), is a logical variable that successively transitions from 0 to 1 and from 1 to 0 periodically:



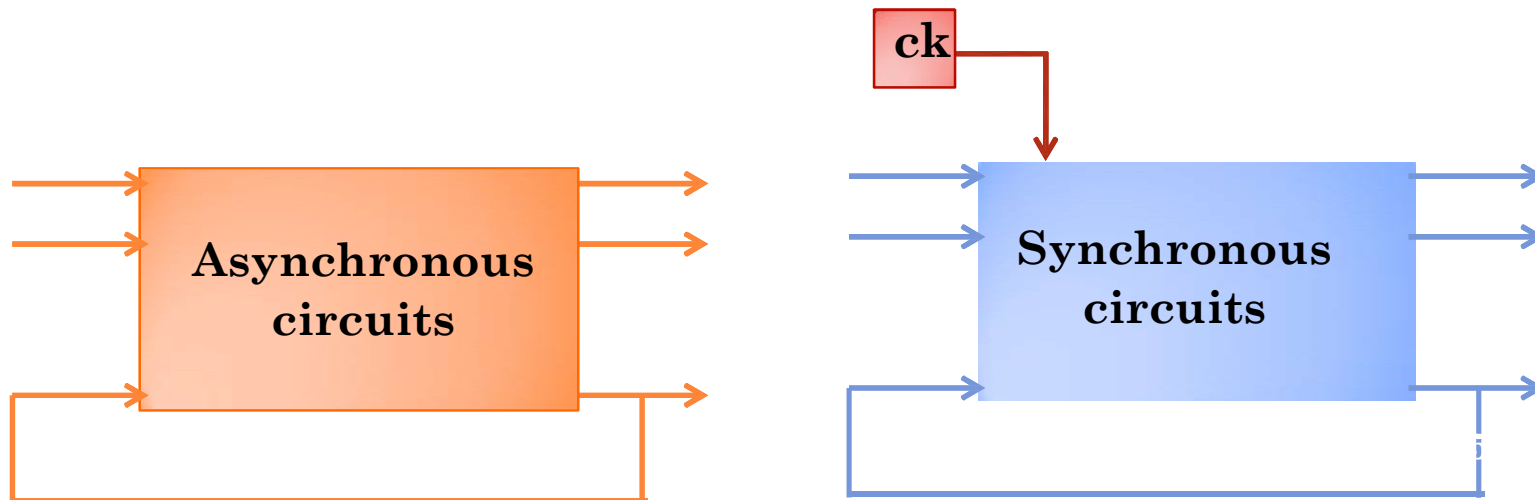
**Frequency =  $1/T$**   
**= number of changes**  
**per second in hertz Hz**

Clock	Period
1 Hz	1 second
1 Mega Hz	1 milli-second
1 Giga Hz	1 nano-second

# Introduction

**Asynchronous Circuits:** System variables evolve freely over time.

**Synchronous Circuits:** The evolution of variables depends on a clock pulse as one of the input signals.



# Part1: Flip-Flops



6



## Flip-Flops

A flip-flop is a logical circuit capable, under certain circumstances, of maintaining the values of its outputs despite changes in input values, that is, of memorizing its state or 'memory.' It is the memory element of sequential logic.

Two main categories of flip-flops are distinguished:

- Asynchronous flip-flops, also called **latches**
- Synchronous flip-flops (dependent on a clock signal), simply called **flip-flops**.

## Flip-flop:

### A Flip-Flop:

A flip-flop is a basic sequential circuit element that allows for the storage of a binary (bit) information. It can be synchronous or asynchronous and has two complementary outputs,  $Q$  and  $\bar{Q}$ .

$$Q^- = Q(t)$$

$$Q^+ = Q(t+1) = F(E_i, Q^-)$$

### Examples include:

RS flip-flop

D flip-flop

JK flip-flop

T flip-flop, etc.



# BASCULE RS



R	S	Q+	
0	0	Q-	Memory State
0	1	1	Set to 1
1	0	0	Reset to 0
1	1	X	Not allowed State

## **RS Flip-Flop Definition:**

In the name of this flip-flop, you notice two letters: 'R' and 'S'.

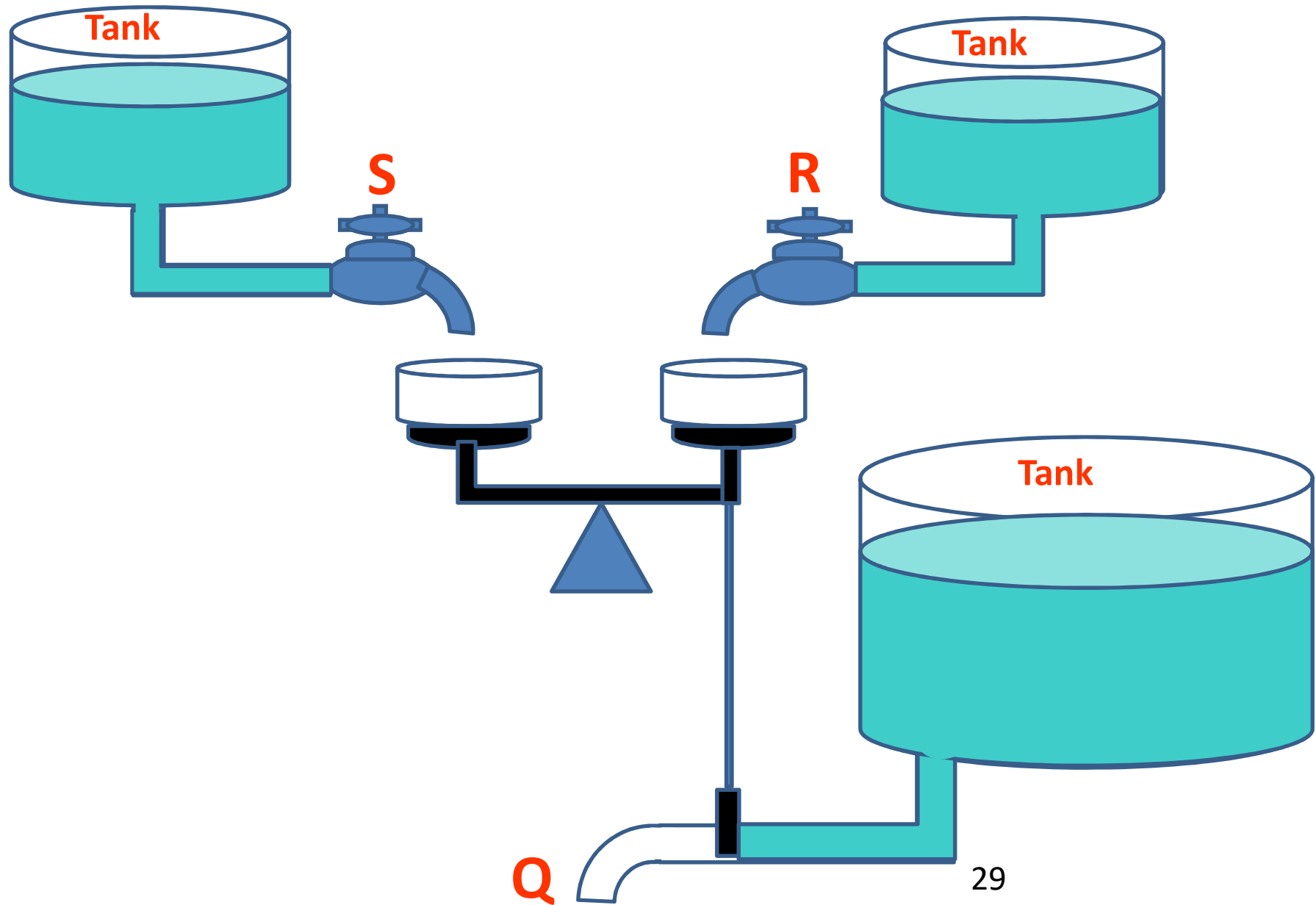
These are the inputs of this flip-flop:

- ✓ 'R' for reset or setting to zero '0' and
- ✓ 'S' for set or setting to one '1'.

Of course, this flip-flop will have an output that we will call Q.

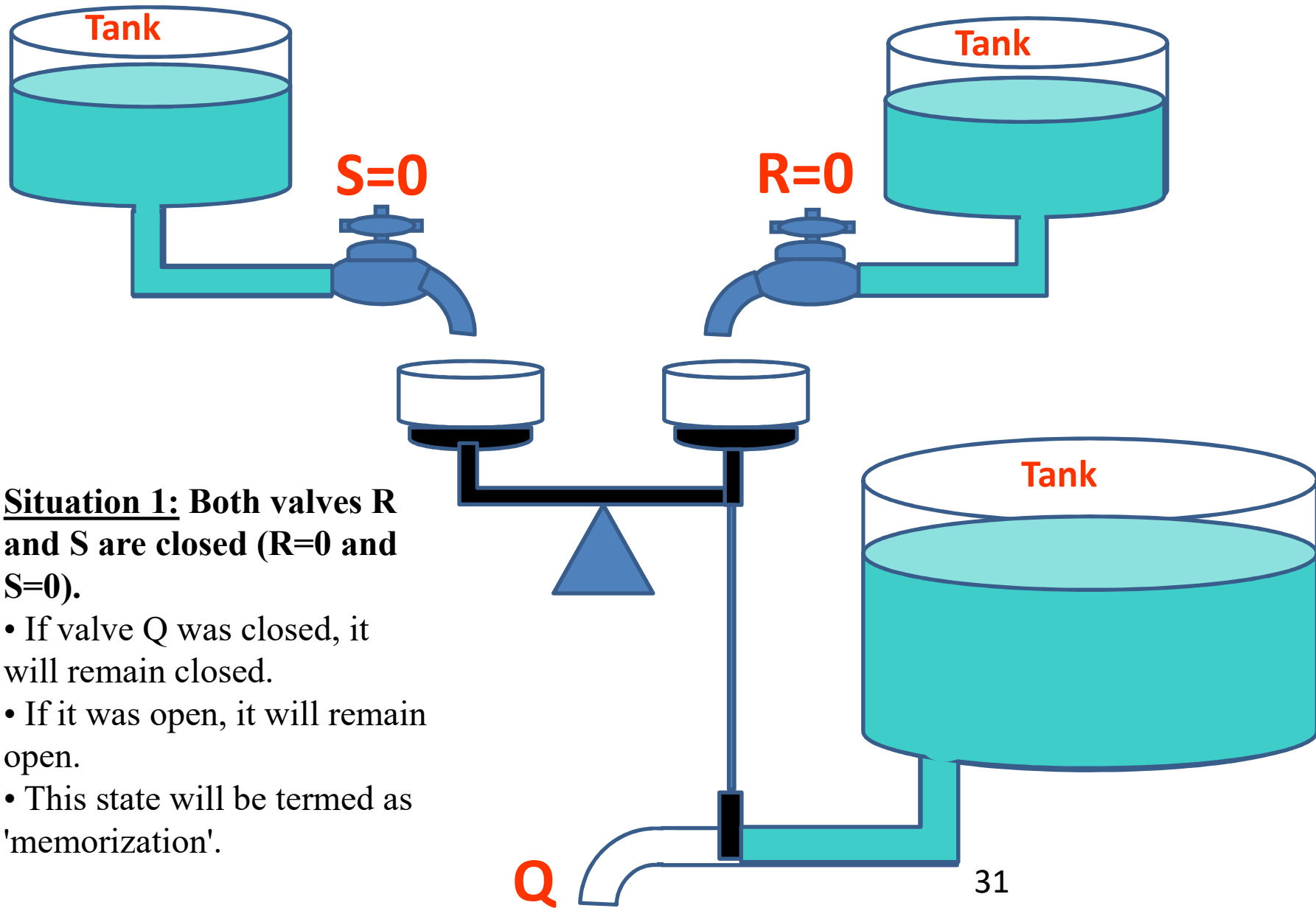
In reality, we will always have two outputs: Q and Q-bar: Q and  $\bar{Q}$ .

# Example of RS Flip-Flop Operation:



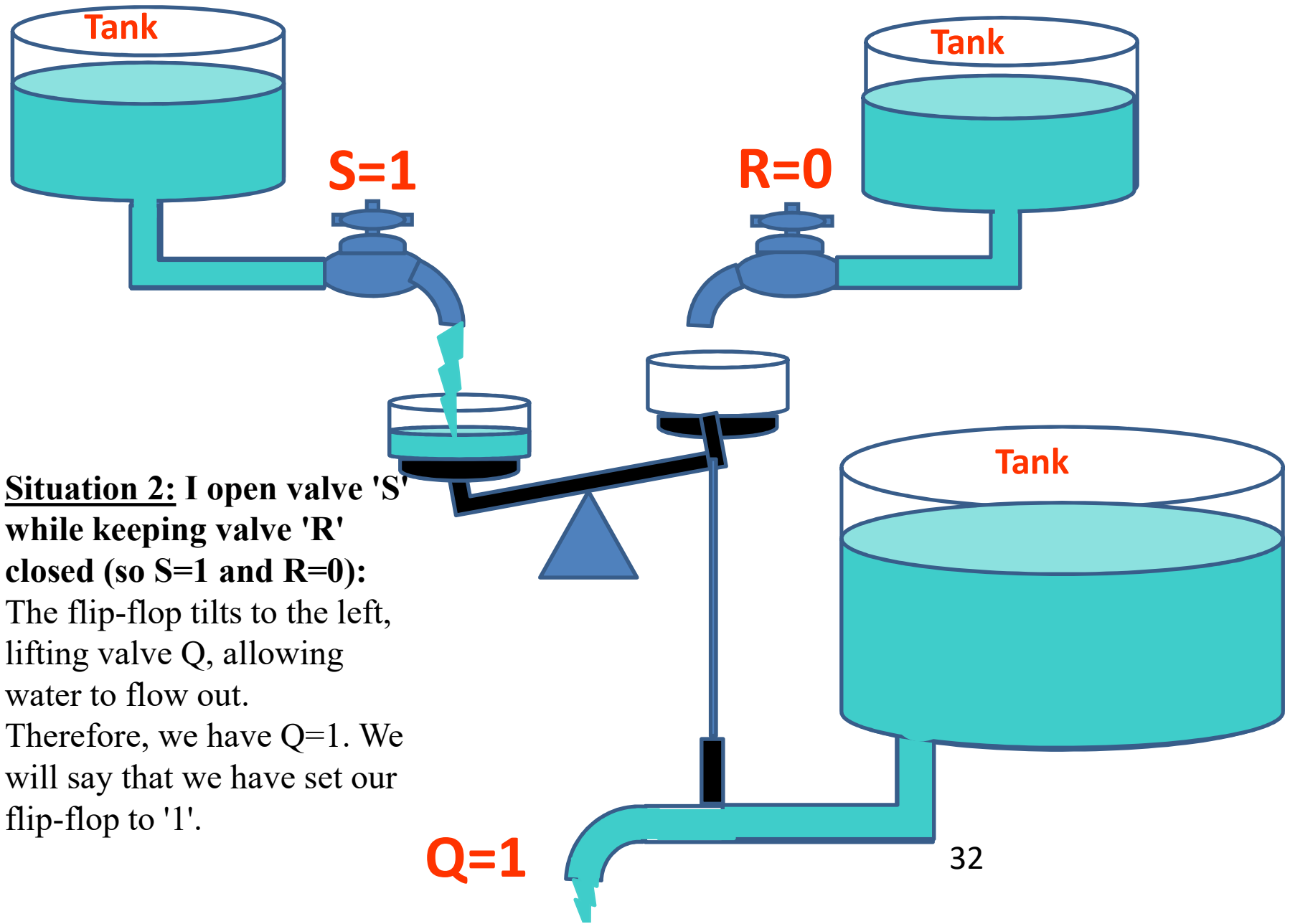
## Example of RS Flip-Flop Operation:

- Two valves R and S
- One flip-flop
- One output Q, which is a pipe
- $R = 1$  means that valve R is open (same for S)
- $R = 0$  means that valve R is closed (same for S)
- Output Q (output pipe) at '1' means water is being discharged
- Output Q (output pipe) at '0' means water is not being discharged
- The right part of the flip-flop is connected to a rod that operates a valve to open or close the output pipe Q.



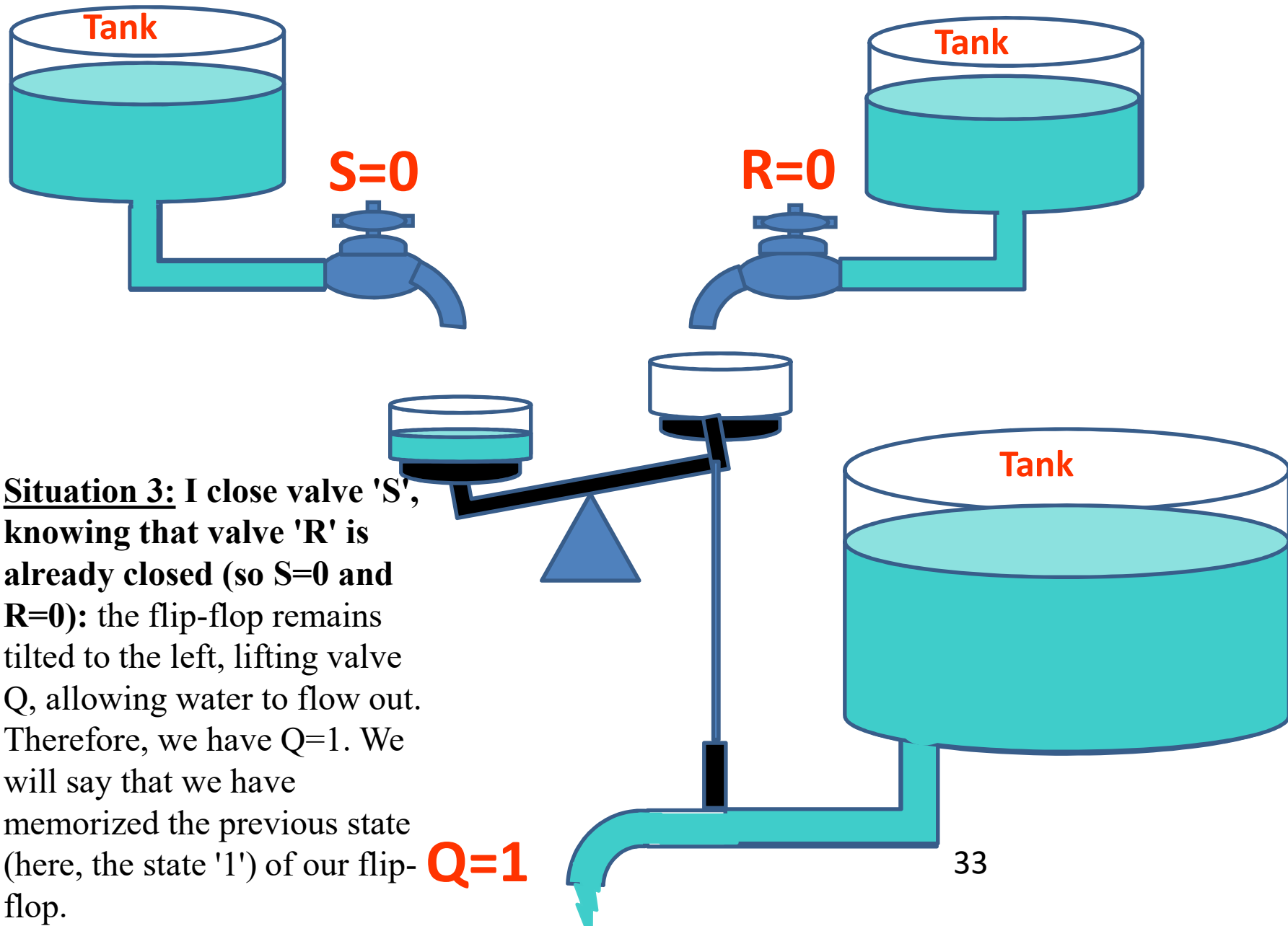
**Situation 1: Both valves R and S are closed ( $R=0$  and  $S=0$ ).**

- If valve Q was closed, it will remain closed.
- If it was open, it will remain open.
- This state will be termed as 'memorization'.

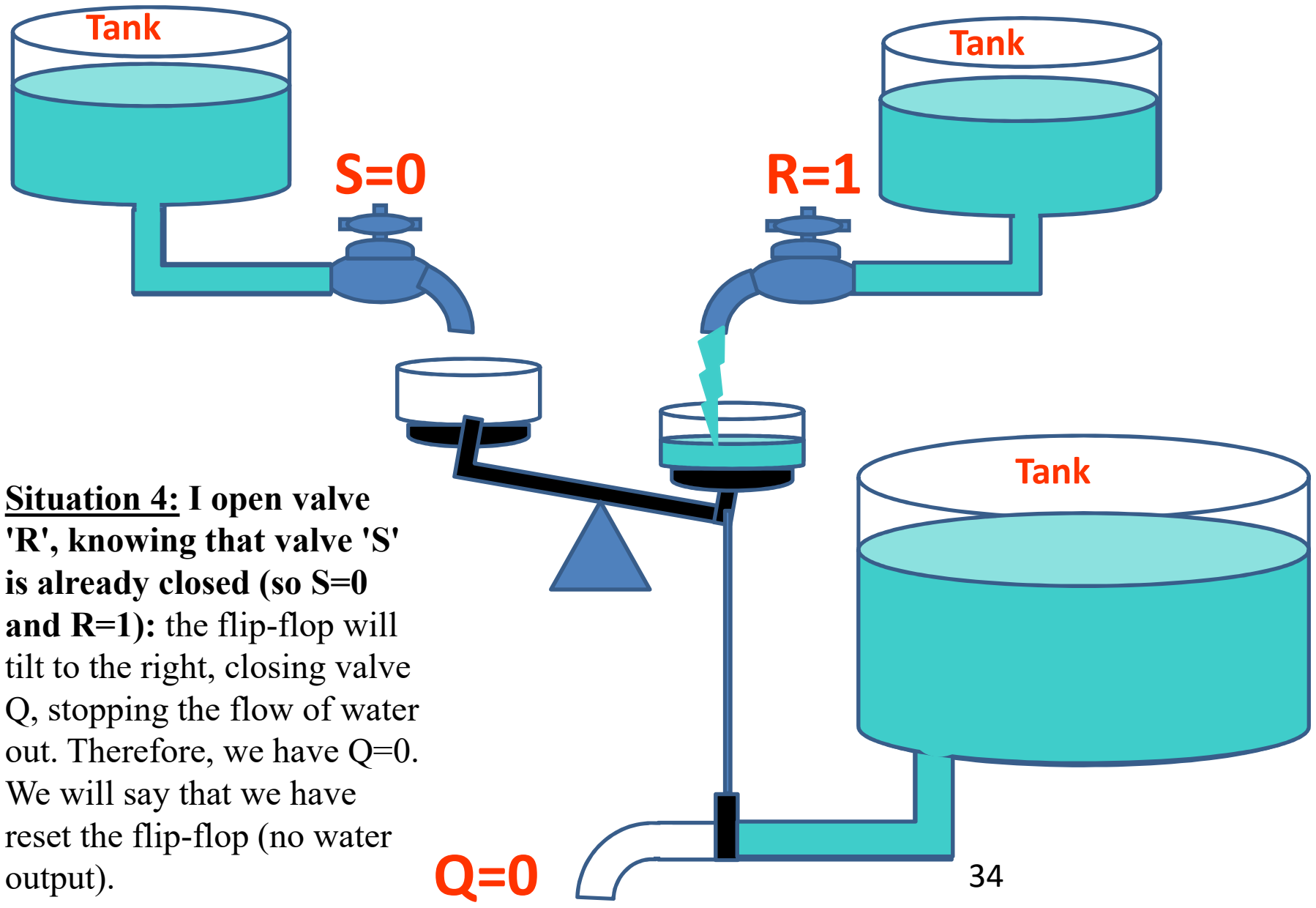


**Situation 2:** I open valve 'S' while keeping valve 'R' closed (so  $S=1$  and  $R=0$ ): The flip-flop tilts to the left, lifting valve Q, allowing water to flow out. Therefore, we have  $Q=1$ . We will say that we have set our flip-flop to '1'.



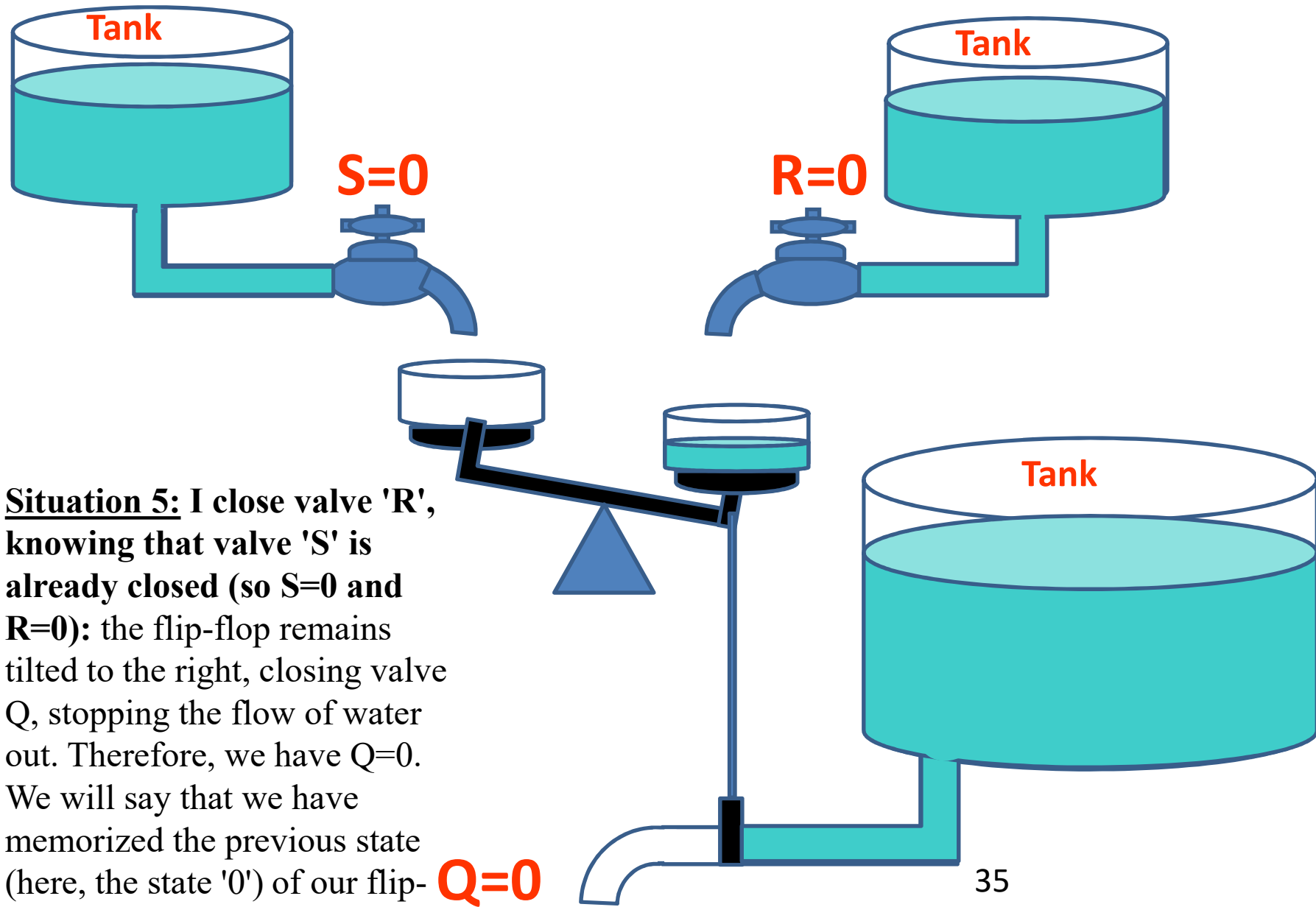


**Situation 3:** I close valve 'S', knowing that valve 'R' is already closed (so  $S=0$  and  $R=0$ ): the flip-flop remains tilted to the left, lifting valve Q, allowing water to flow out. Therefore, we have  $Q=1$ . We will say that we have memorized the previous state (here, the state '1') of our flip-flop.

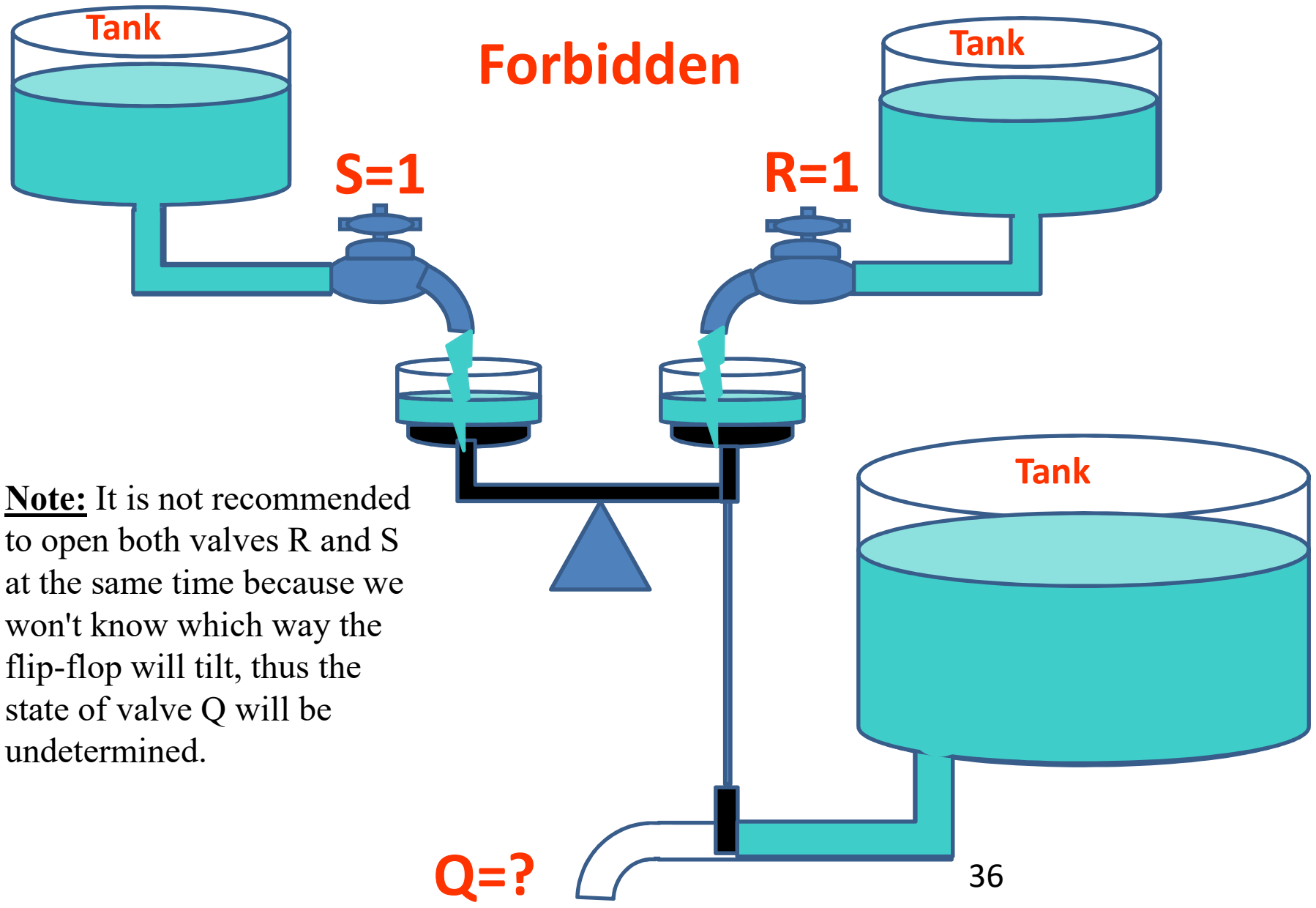


**Situation 4:** I open valve 'R', knowing that valve 'S' is already closed (so  $S=0$  and  $R=1$ ): the flip-flop will tilt to the right, closing valve Q, stopping the flow of water out. Therefore, we have  $Q=0$ . We will say that we have reset the flip-flop (no water output).

$Q=0$



**Situation 5:** I close valve 'R', knowing that valve 'S' is already closed (so  $S=0$  and  $R=0$ ): the flip-flop remains tilted to the right, closing valve Q, stopping the flow of water out. Therefore, we have  $Q=0$ . We will say that we have memorized the previous state (here, the state '0') of our flip-flop.

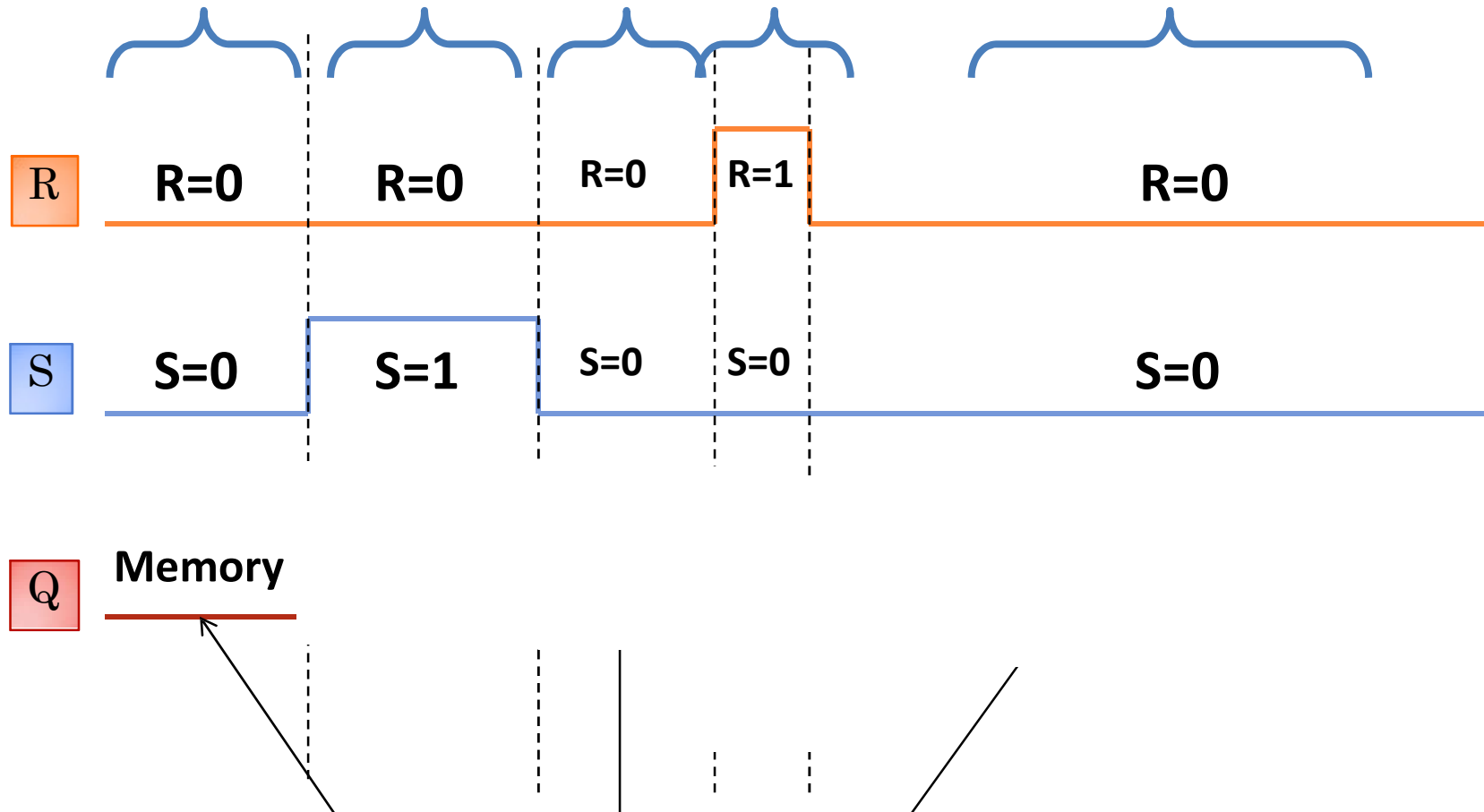


**Note:** It is not recommended to open both valves R and S at the same time because we won't know which way the flip-flop will tilt, thus the state of valve Q will be undetermined.

Q=?

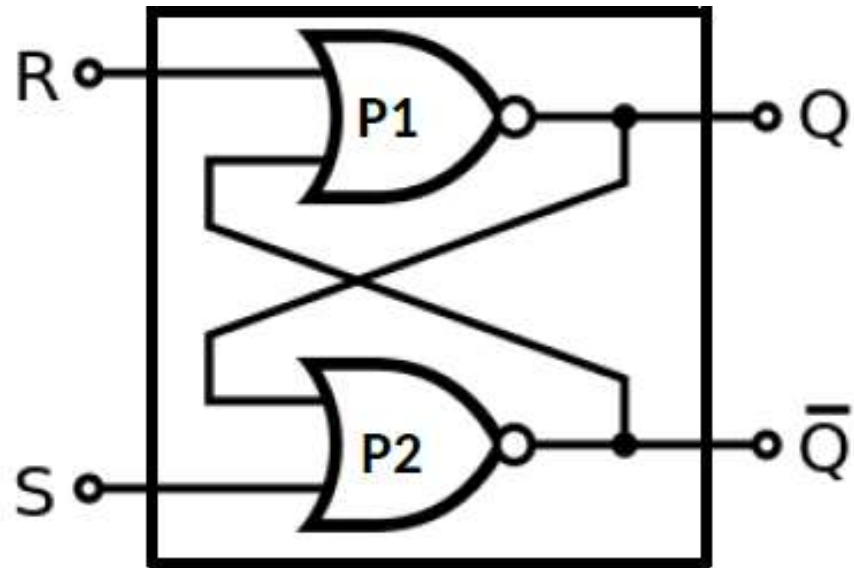
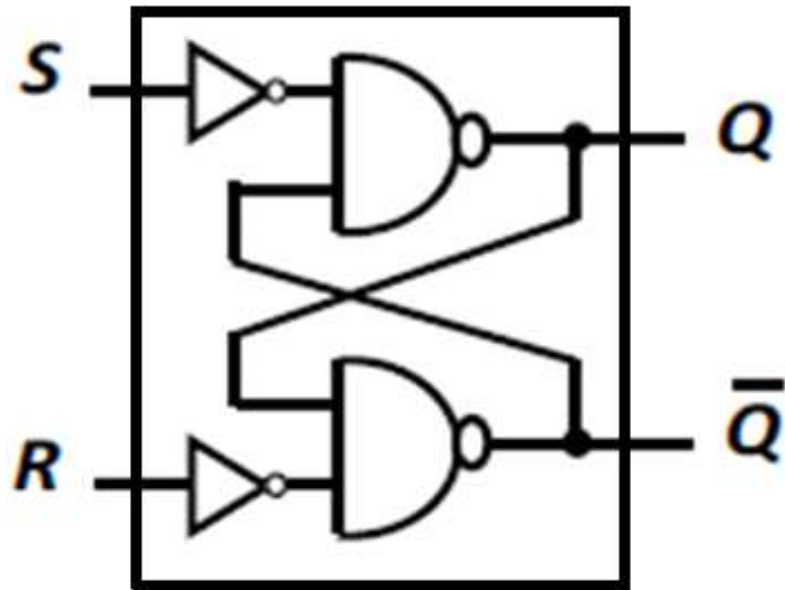
# RS Flip-flop

Timing diagram:



# RS Flip-flop

Internal structure



# RS Flip-flop

## Internal structure

Remember that the purpose of a flip-flop is to fulfill the memory function.

We will use the two inputs R and S to set the flip-flop to '1' or '0'.

When both R and S are '0', the flip-flop should remember its previous state.

You will see that we will avoid setting both R and S to '1' at the same time.

# RS Flip-flop

## Internal structure

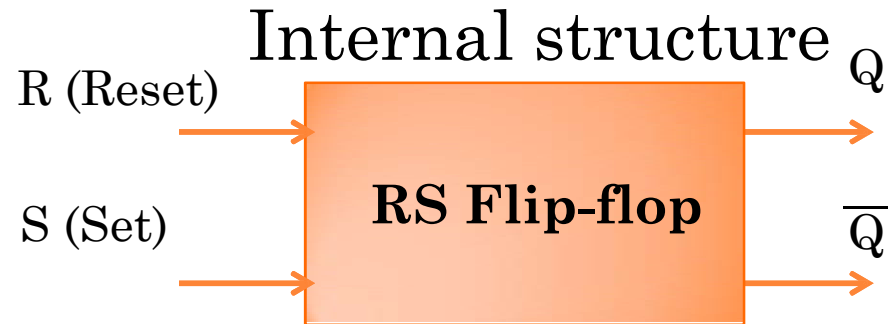
Let's verify this operation by analyzing the RS flip-flop circuit based on NOR gates:

We can represent the function  $Q$  as:

- two states  $Q^+$  and  $Q$ .
- $Q^+$  indicating the future state of the flip-flop and  $Q$  indicating its present state.
- Thus, we can write  $Q^+ = f(R, S, Q)$ .



# RS Flip-flop



R	S	Q+	
0	0	Q-	Memory state
0	1	1	(Set) to 1
1	0	0	(Reset) to 0
1	1	X	Forbidden state

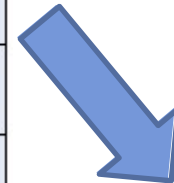
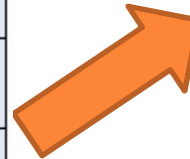


R	S	Q-	Q+
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	X
1	1	1	X

# RS Flip-flop

## Internal structure

R	S	Q-	$\bar{Q}$ -	Q+	$\bar{Q}$ +
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	1	1	0
0	1	1	0	1	0
1	0	0	1	0	1
1	0	1	0	0	1
1	1	0	1	X	X
1	1	1	0	X	X



RS	00	01	11	10
Q-				
0	0	1	X	0
1	1	1	X	0

$$Q_+ = S + \bar{R} Q_-$$

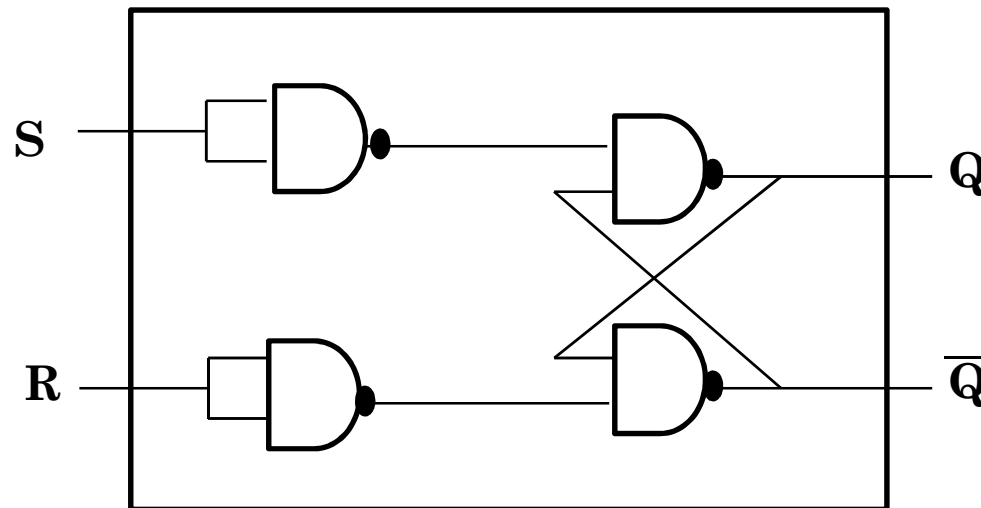
RS	00	01	11	10
$\bar{Q}$ -				
0	0	0	X	1
1	1	0	X	1

$$\bar{Q}_+ = R + \bar{S} \bar{Q}_-$$

# RS Flip-flop

## Internal structure

Exercise 1: Implement the RS flip-flop using only NAND gates.







# SYNCHRONOUS FLIP-FLOPS

The change of state of an output in a synchronous sequential system depends on the **state of the control inputs** and that of the active synchronization signal called the **clock signal**.

## Clock Signal Modes of Operation:

There are four modes of clock signal operation known by the following symbols (see figure):

Symbol	Designation
	High level
	Low level
	Rising edge
	Falling edge

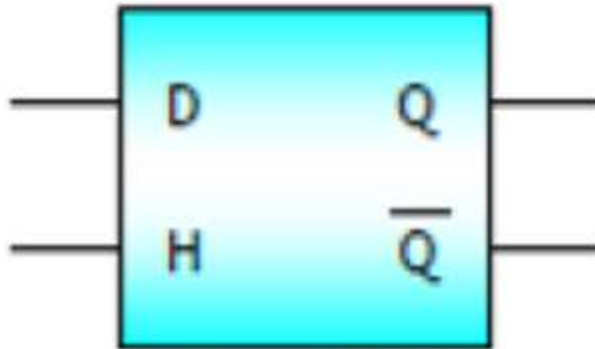
# SYNCHRONOUS FLIP-FLOPS

According to the clock signal mode of operation, we distinguish two families of flip-flops:

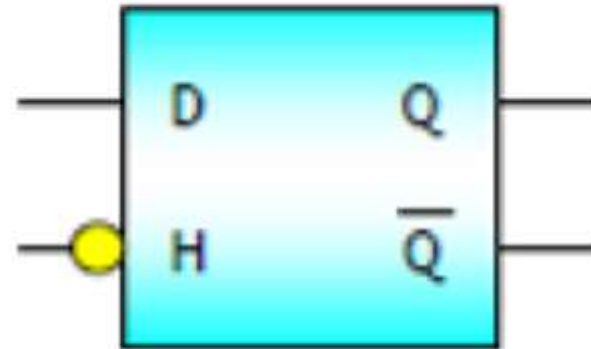
- **Level-triggered flip-flops (high-level or low-level control):** The flip-flop is said to be controlled statically (active on high-level or active on low-level).
- **Edge-triggered flip-flops (rising edge or falling edge control):** The flip-flop is said to be controlled dynamically (active on rising edge or active on falling edge).

# SYNCHRONOUS FLIP-FLOPS

Symbol:

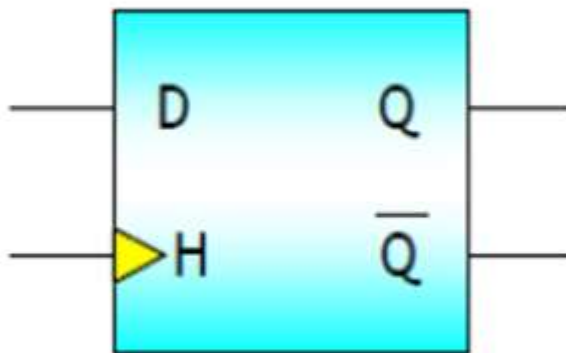


Active on the high level of the clock signal Ck

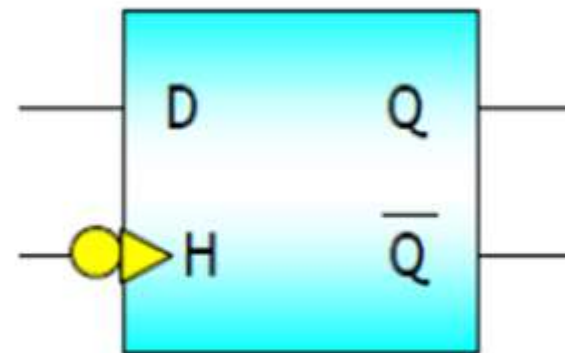


Active on the low level of the clock signal Ck

Symbol:



Active on the  $\uparrow$  of Ck



Active on the  $\downarrow$  of Ck

## SYNCHRONOUS FLIP-FLOPS RSH

### Synchronous RSH or RST Flip-Flop (RS with enable)

We can easily modify the RS flip-flop to make it synchronous, meaning that changes are only allowed when the clock signal is high (or low).

The RSH flip-flop is a flip-flop where the inputs **S** and **R** are only taken into account in **coincidence** with a **control signal**.

This signal can be provided by a clock, thus creating a synchronous flip-flop.

Therefore, the RSH flip-flop is an RS flip-flop whose Set and Reset commands **only change the state** of the output **Q** after the authorization of a clock signal **H (Clock CK)**.

## RSH Flip-Flop: RS flip-flop with a clock signal



The RSH flip-flop is a: synchronized RS flip-flop with a clock signal H.

H	R	S	Q+	
0	X	X	Q-	Memory
1	0	0	Q-	
1	0	1	1	RS Flip-Flop
1	1	0	0	
1	1	1	X	



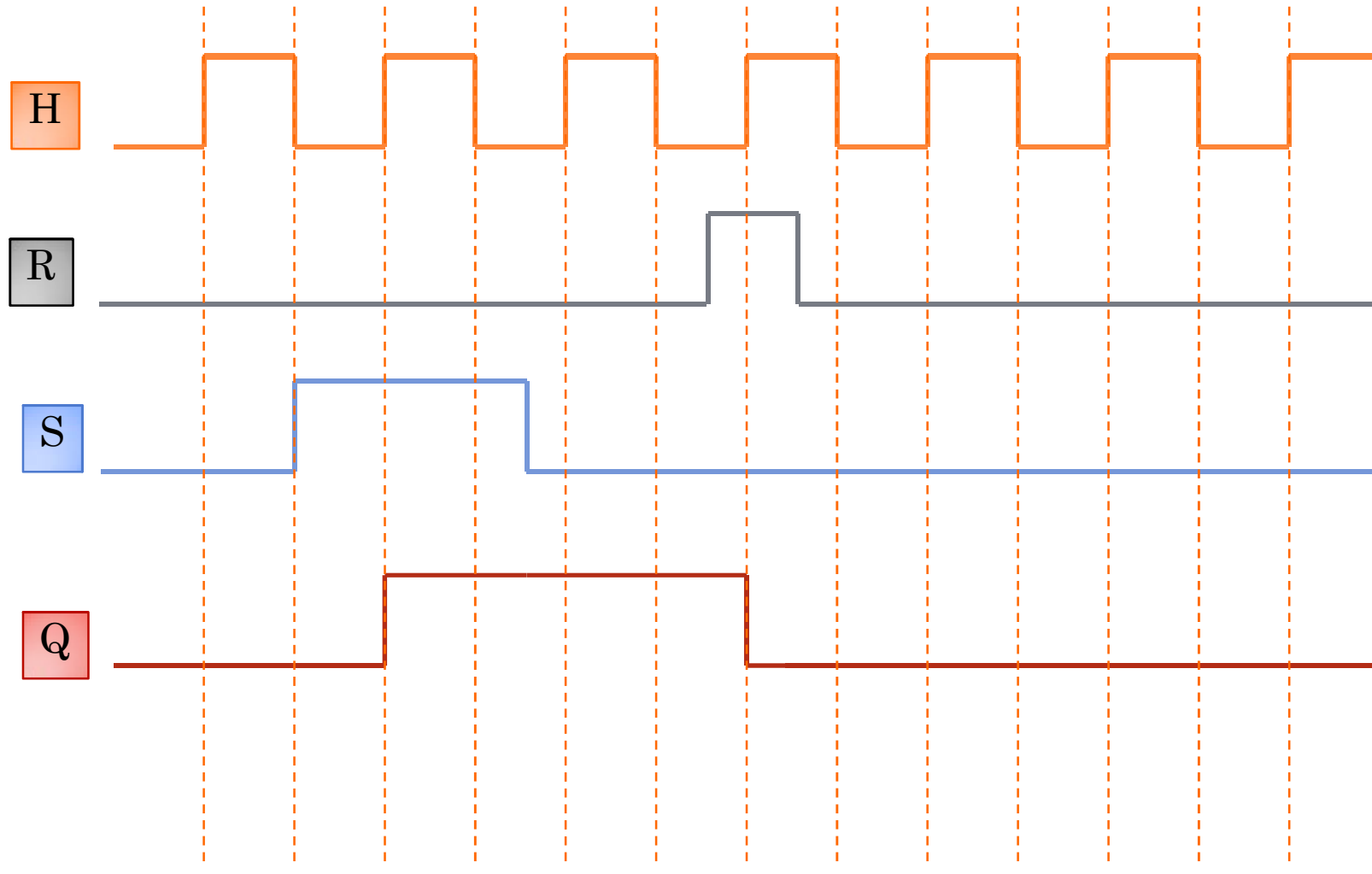
# Types de synchronisations

Several ways to consider changes in the state of flip-flops:

- **High-level clock:** we consider the flip-flop inputs when the clock level is high (at '1').
  - **Low-level clock:** we consider the flip-flop inputs when the clock level is low (at '0').
- 
- Rising clock edge.
  - Falling clock edge.

# RSH flip-flop (Synchronous RS)

## TIMING DIAGRAM



# RSH flip-flop (Synchronous RS) TIMING DIAGRAM

H 0

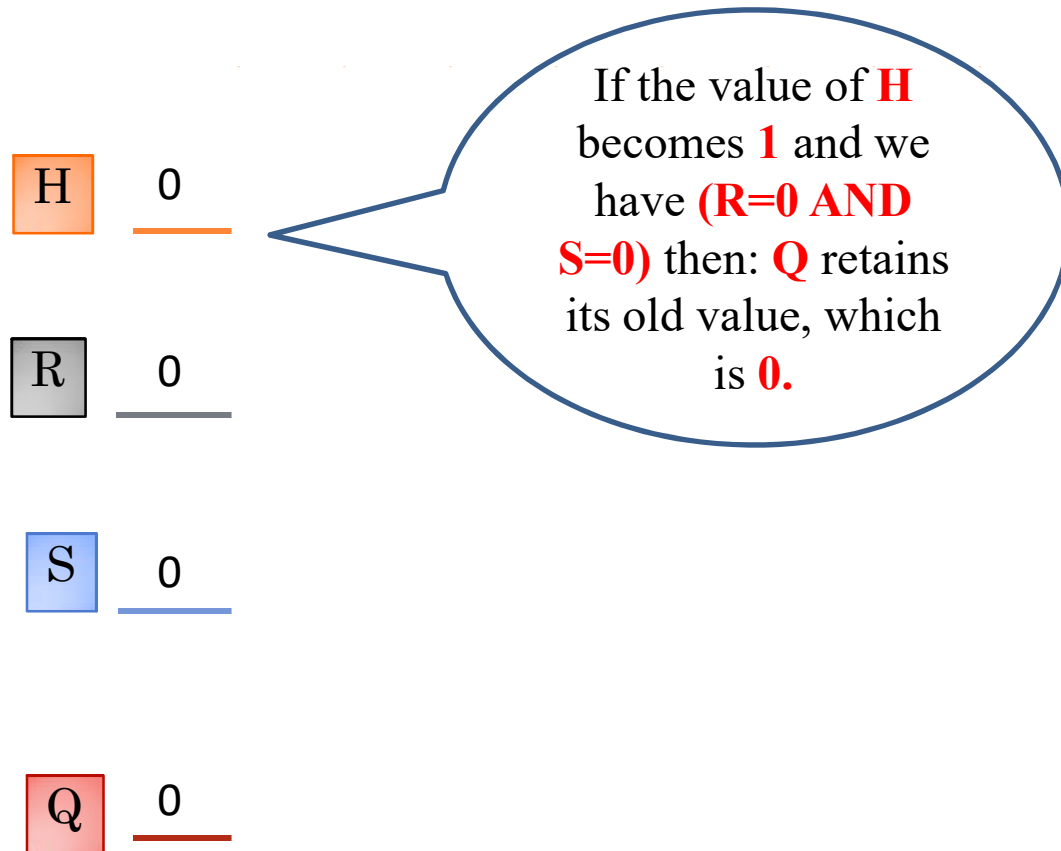
R 0

S 0

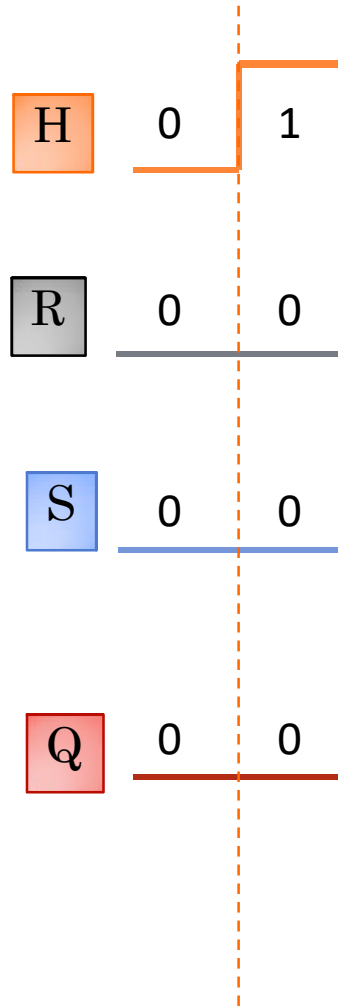
Q 0

# RSH flip-flop (Synchronous RS)

## TIMING DIAGRAM

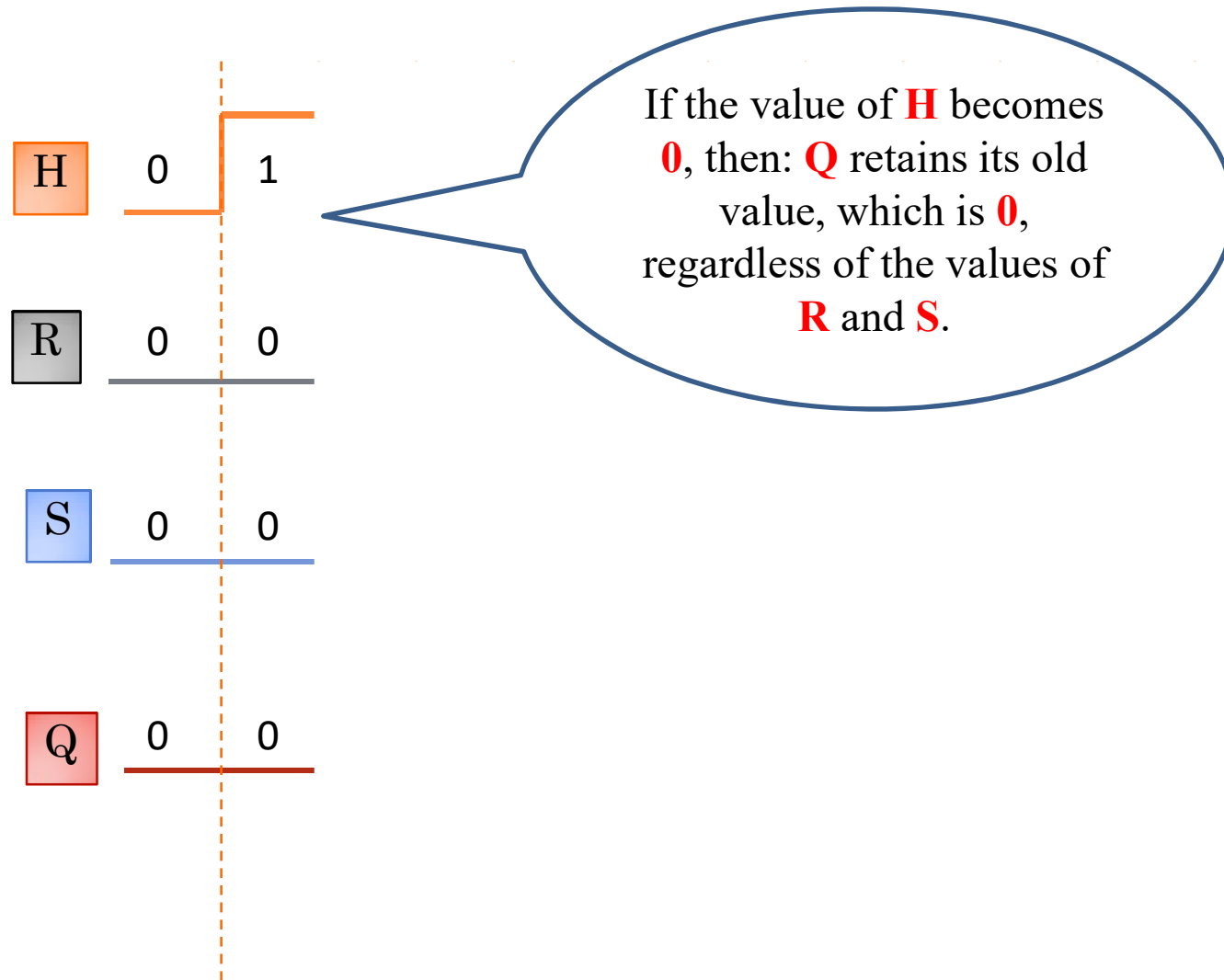


# RSH flip-flop (Synchronous RS) TIMING DIAGRAM

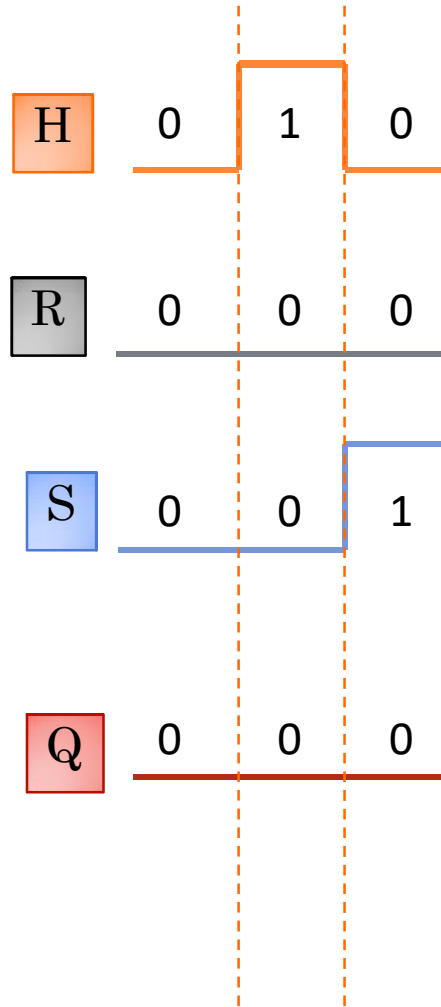


# RSH flip-flop (Synchronous RS)

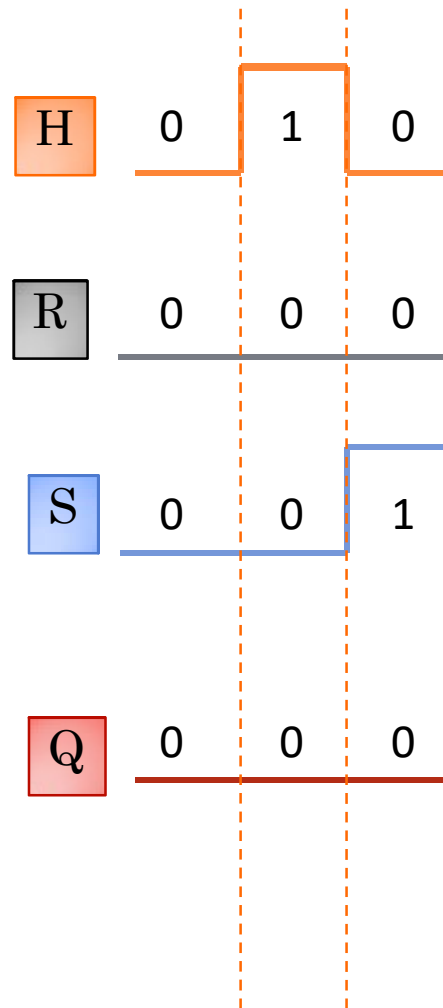
## TIMING DIAGRAM



# RSH flip-flop (Synchronous RS) TIMING DIAGRAM



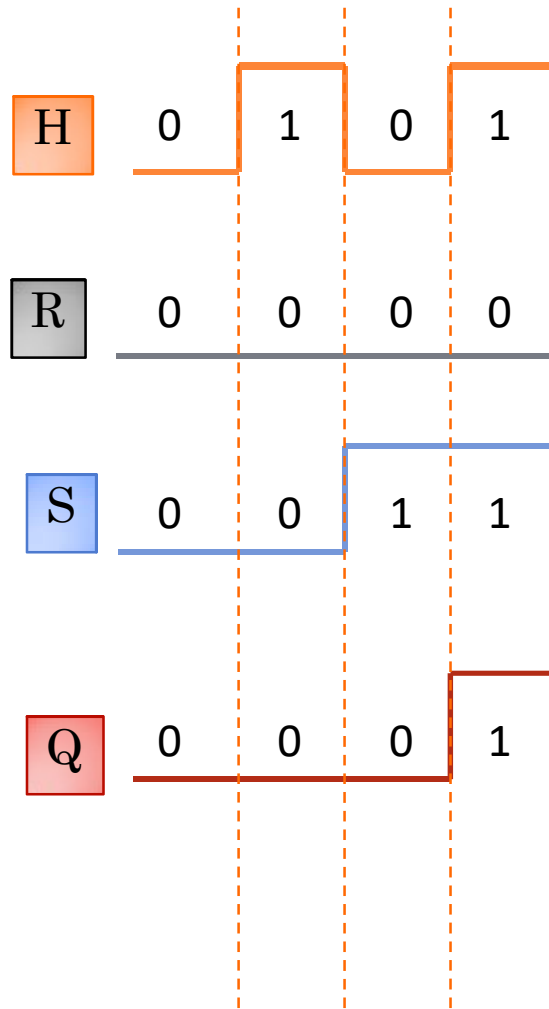
# RSH flip-flop (Synchronous RS) TIMING DIAGRAM



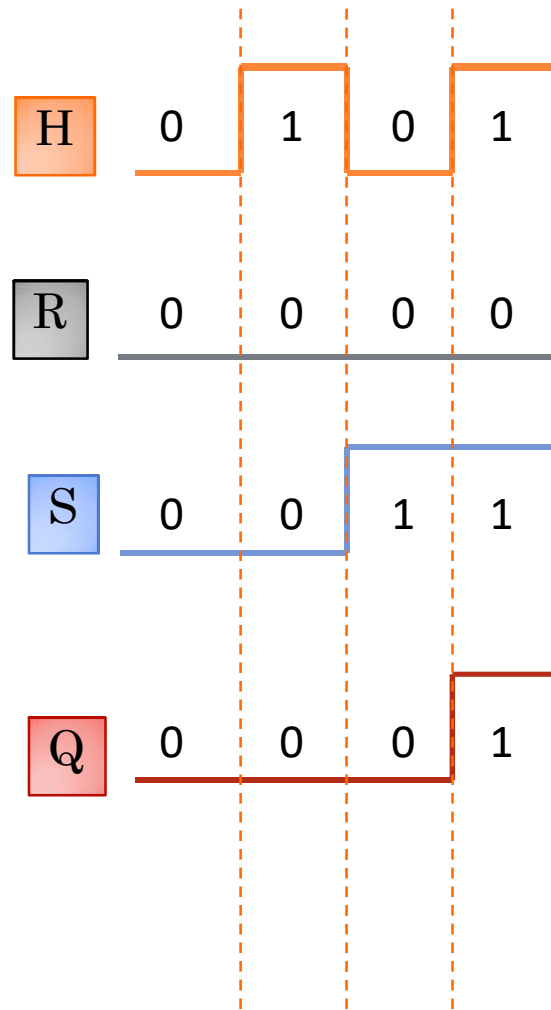
If the value of **H** switches back to **1**, then: **Q** changes its state according to **R** and **S**, which is (**R=0 AND S=1**), so: **Q=1**.



# RSH flip-flop (Synchronous RS) TIMING DIAGRAM

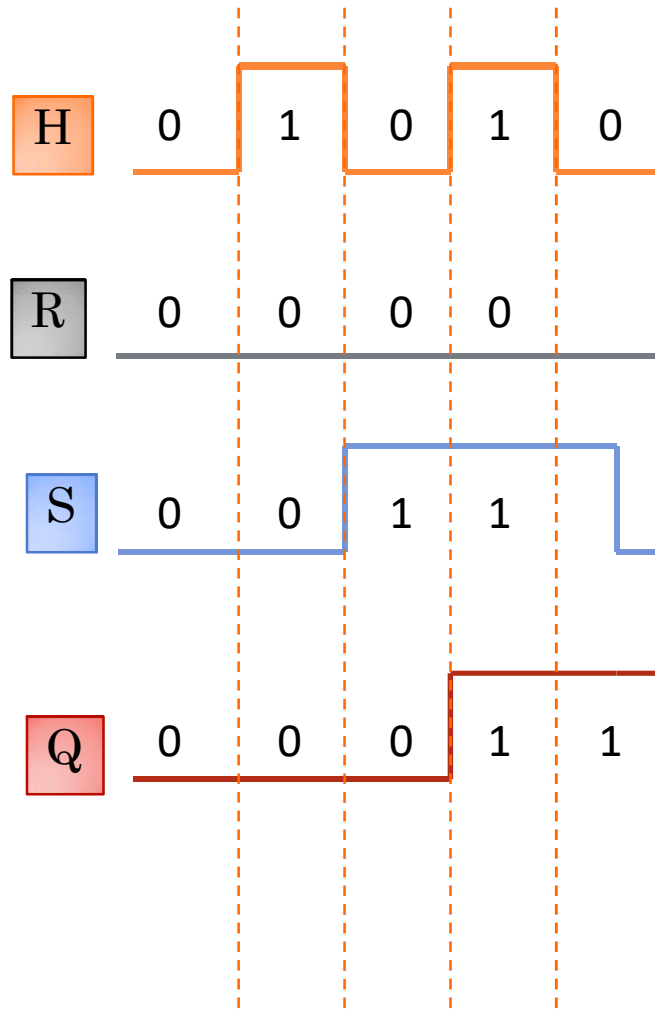


# RSH flip-flop (Synchronous RS) TIMING DIAGRAM



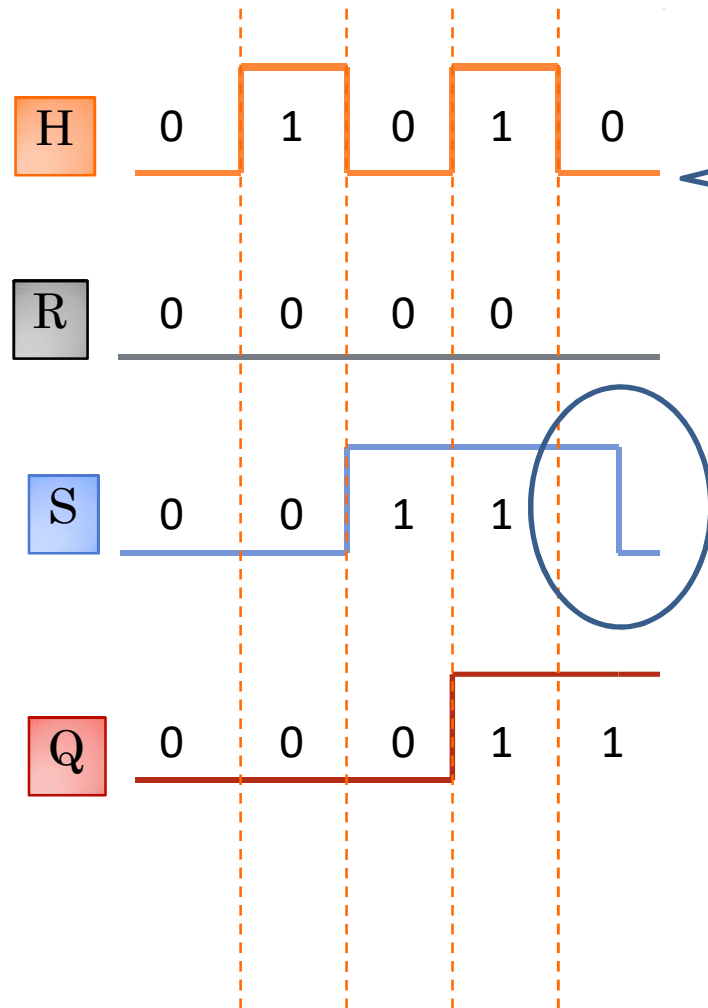
If the value of **H** becomes **0**, then: **Q** retains its old value, which is **1**, regardless of the values of **R** and **S**.

# RSH flip-flop (Synchronous RS) TIMING DIAGRAM



# RSH flip-flop (Synchronous RS)

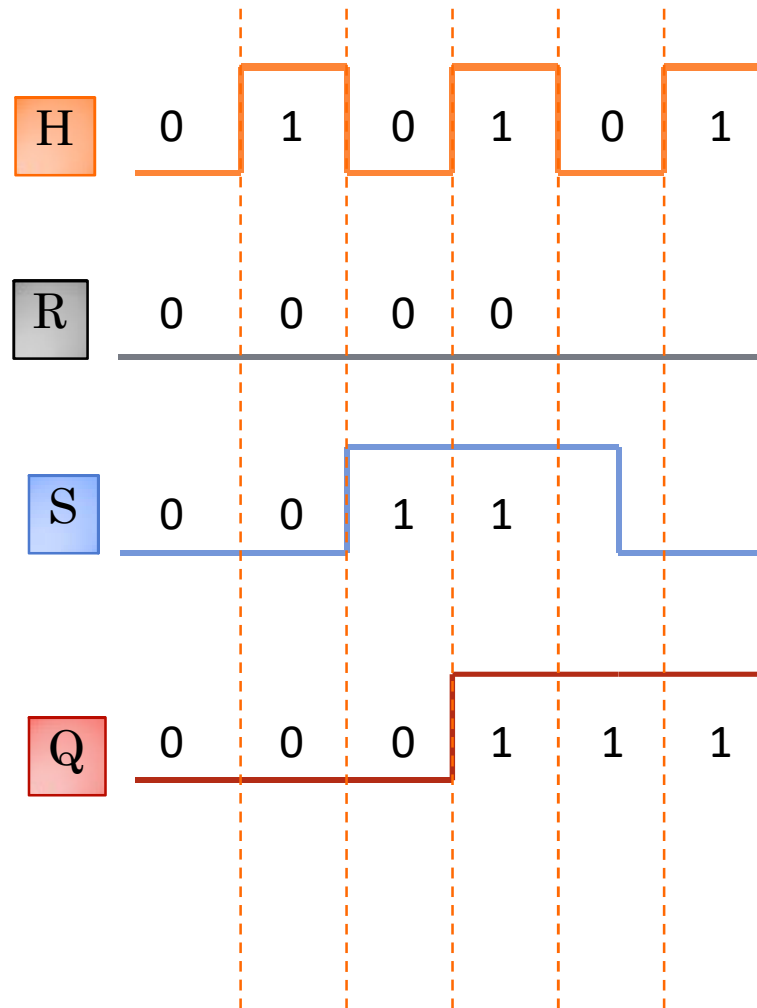
## TIMING DIAGRAM



If the value of **H** switches back to **1**, then: **Q** changes its state according to **R** and **S**, which is (**R=0 AND S=0**), so: **Q** retains its old value, which is **1**.

# RSH flip-flop (Synchronous RS)

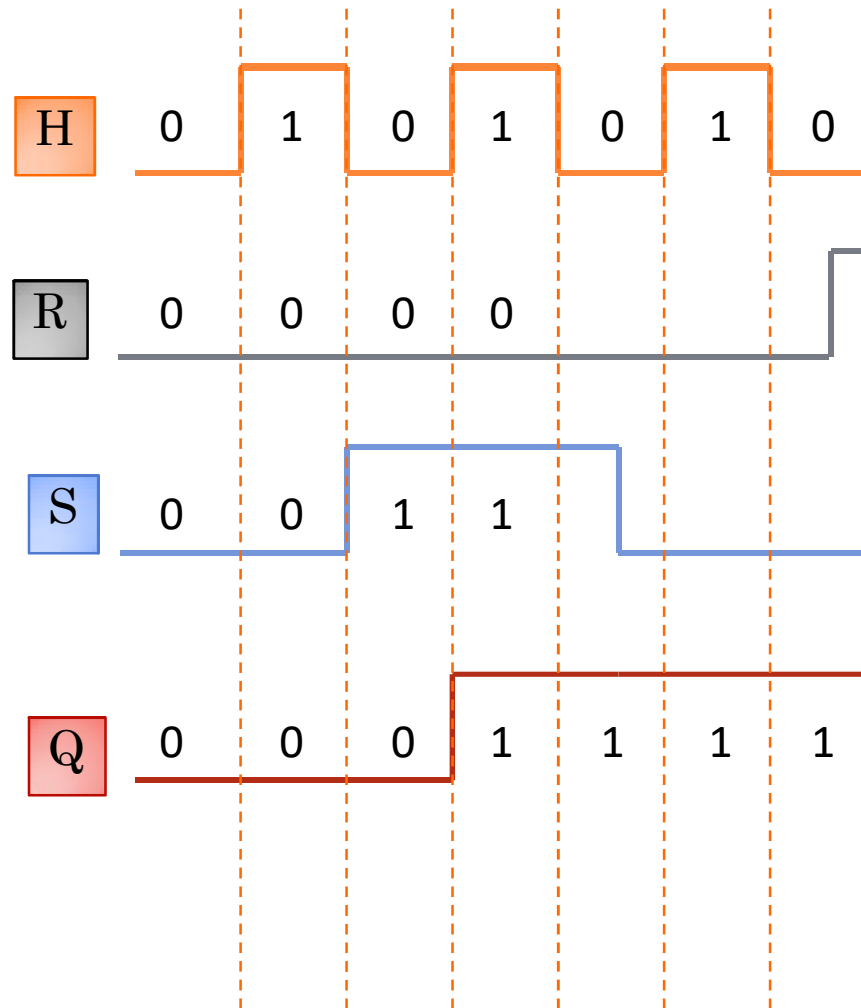
## TIMING DIAGRAM



If the value of **H** becomes **0**, then: **Q** retains its old value, which is **1**, regardless of the values of **R** and **S**.

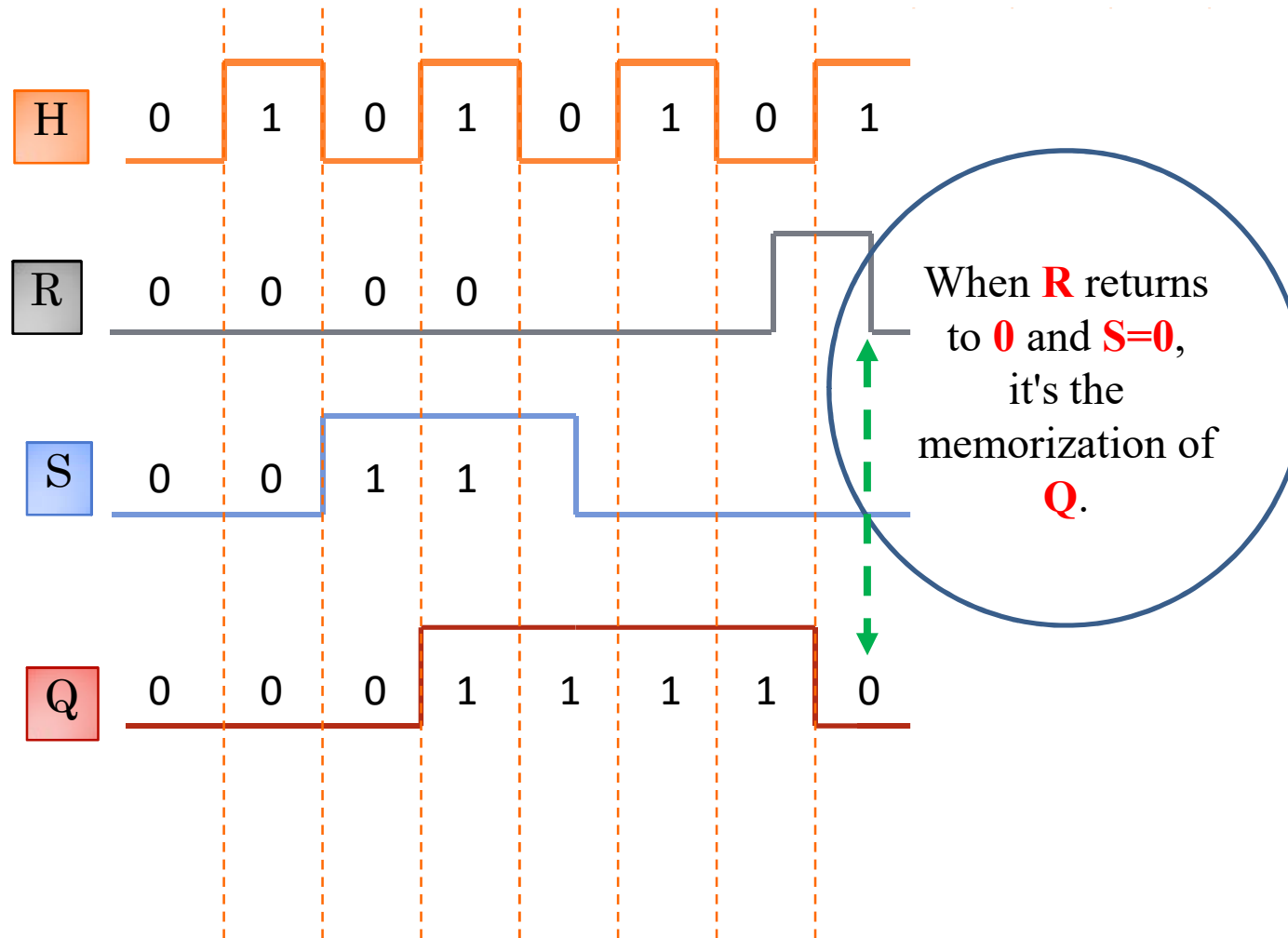
# RSH flip-flop (Synchronous RS)

## TIMING DIAGRAM

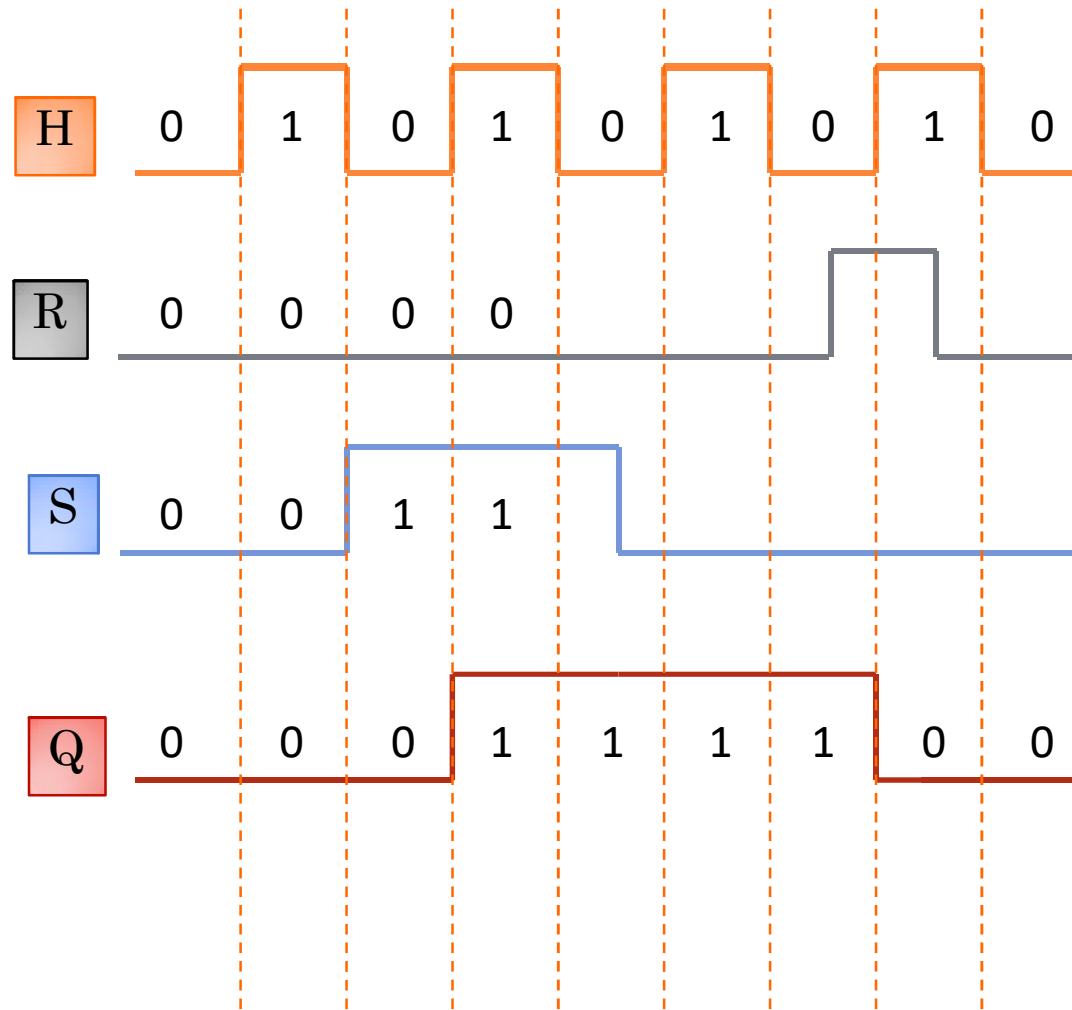


If the value of **H** switches back to **1**, then: **Q** changes its state according to **R** and **S**, which is (**R=1 AND S=0**), so: **Q=0**.

# RSH flip-flop (Synchronous RS) TIMING DIAGRAM

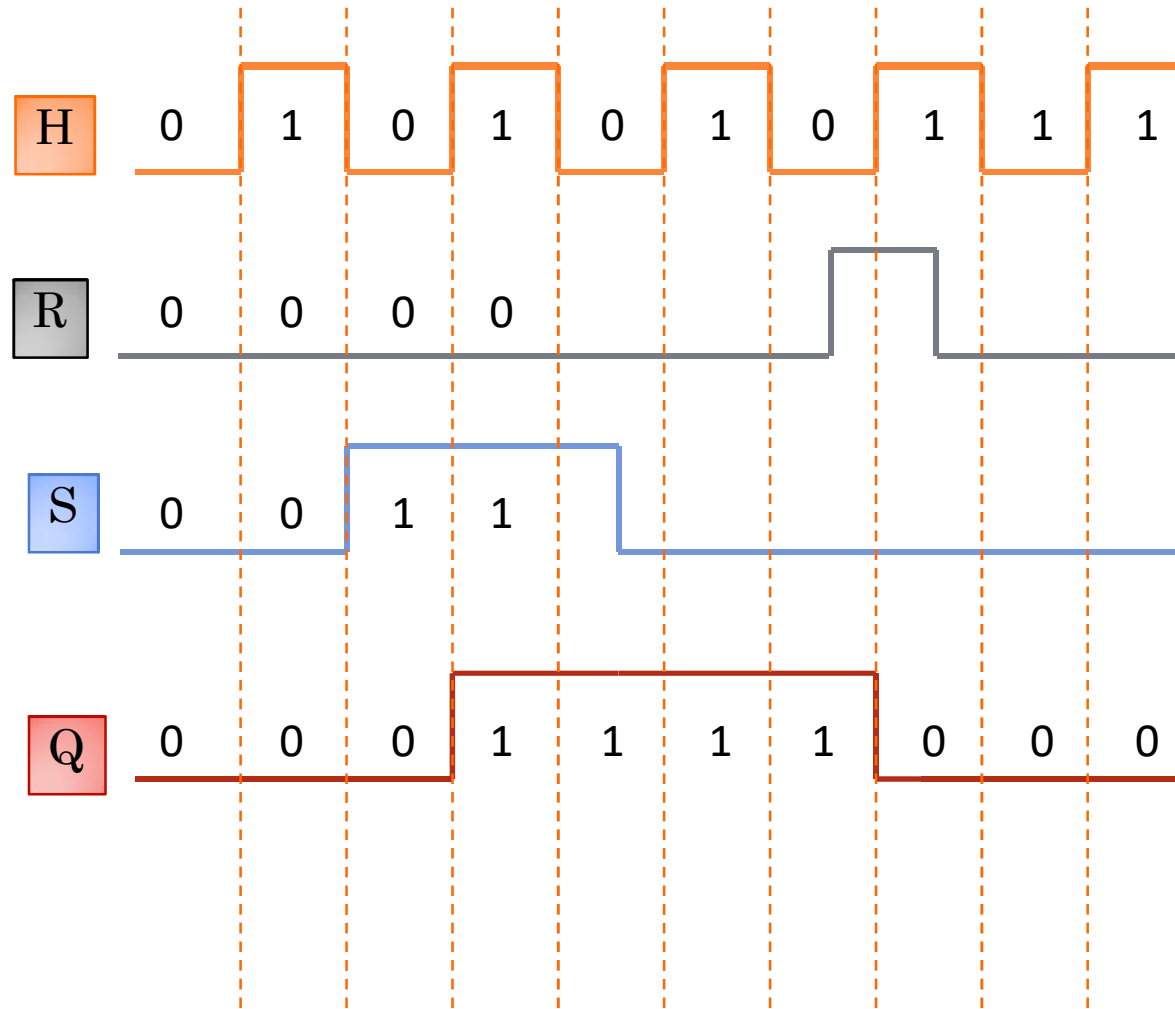


# RSH flip-flop (Synchronous RS) TIMING DIAGRAM





# RSH flip-flop (Synchronous RS) TIMING DIAGRAM



# RSH flip-flop (Synchronous RS)

## EXAMPLE: TIMING DIAGRAM



# RSH flip-flop (Synchronous RS)

## EXAMPLE: TIMING DIAGRAM



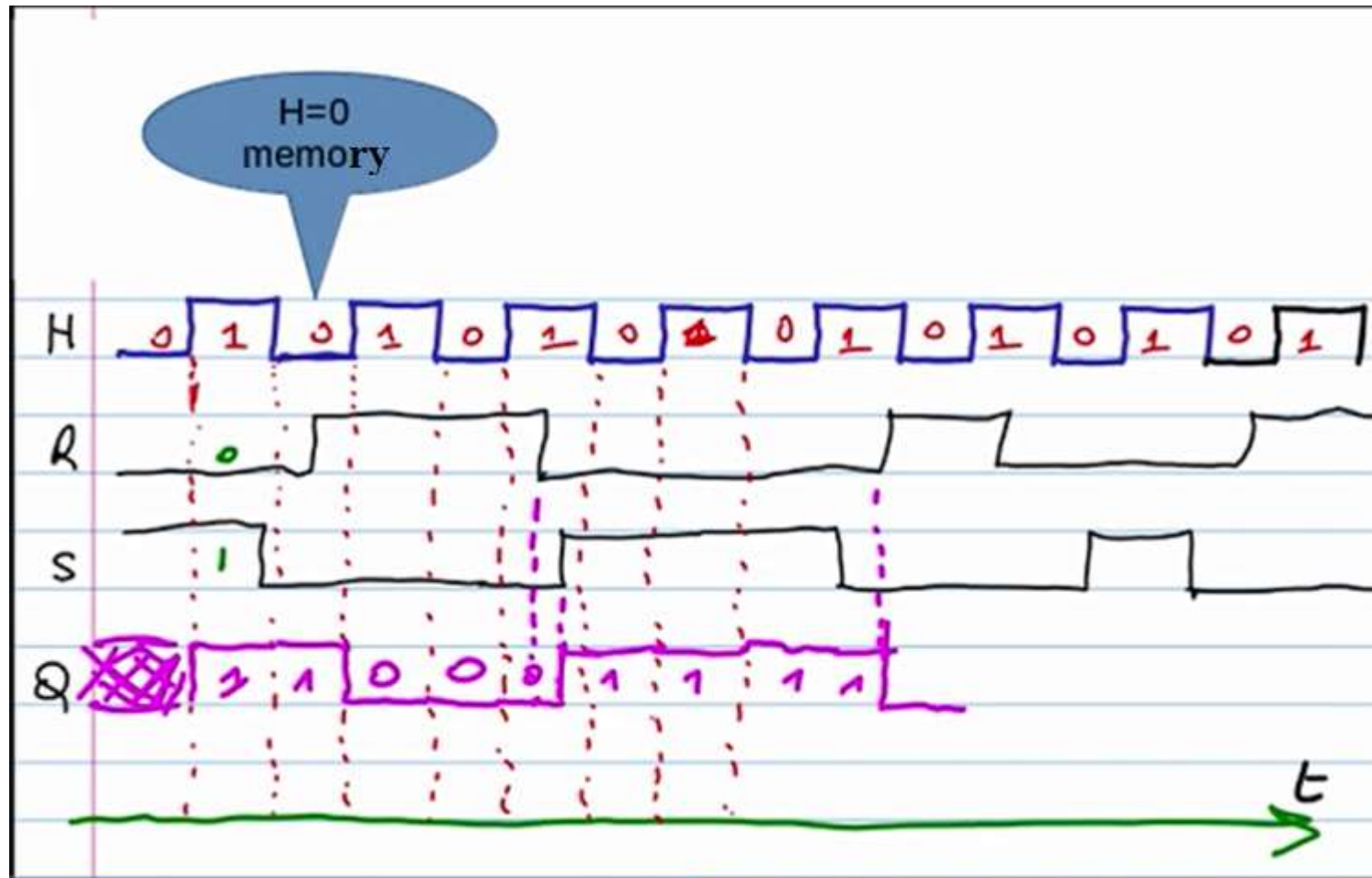
# RSH flip-flop (Synchronous RS)

## EXAMPLE: TIMING DIAGRAM



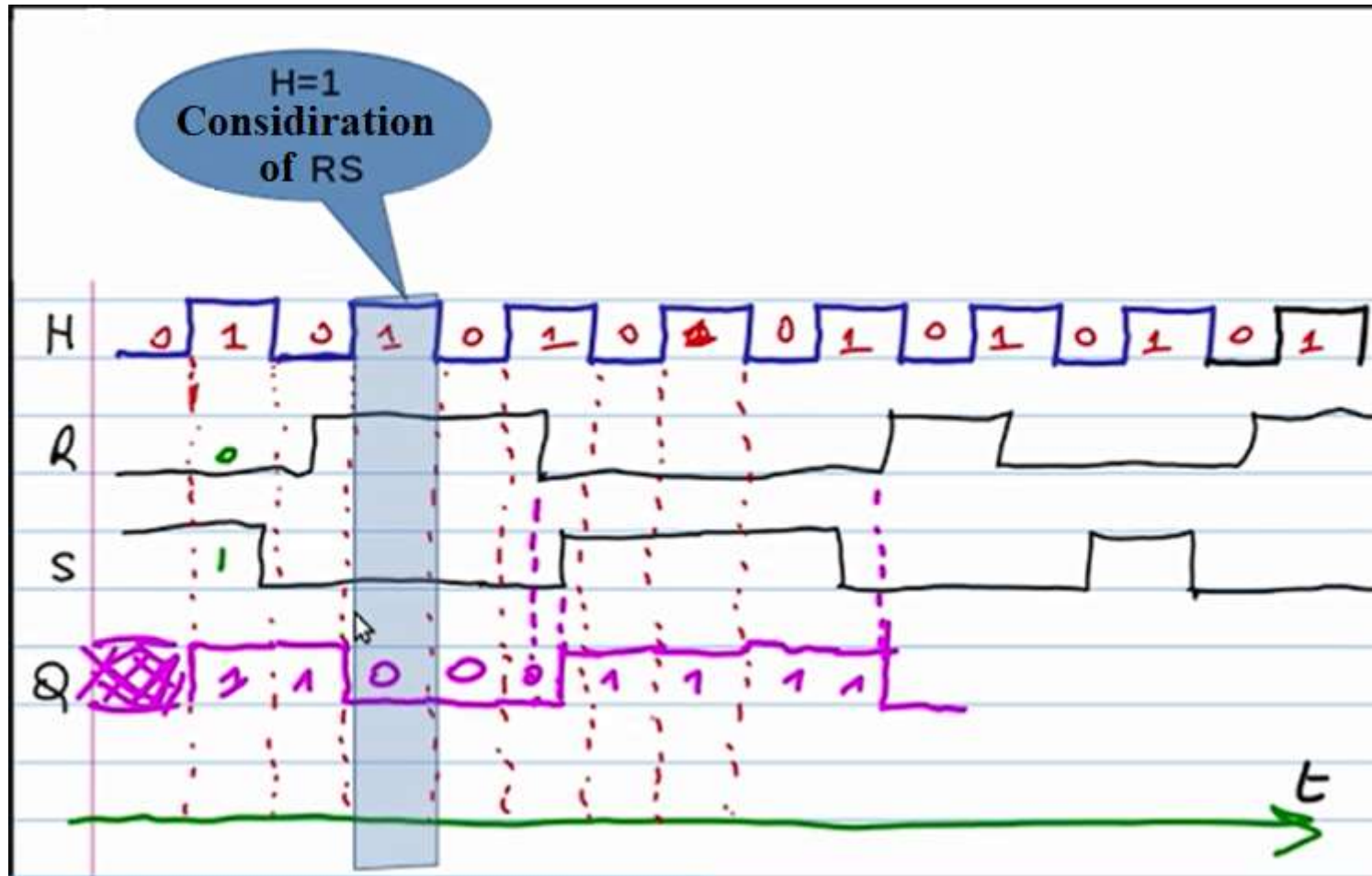
# RSH flip-flop (Synchronous RS)

## EXAMPLE: TIMING DIAGRAM



# RSH flip-flop (Synchronous RS)

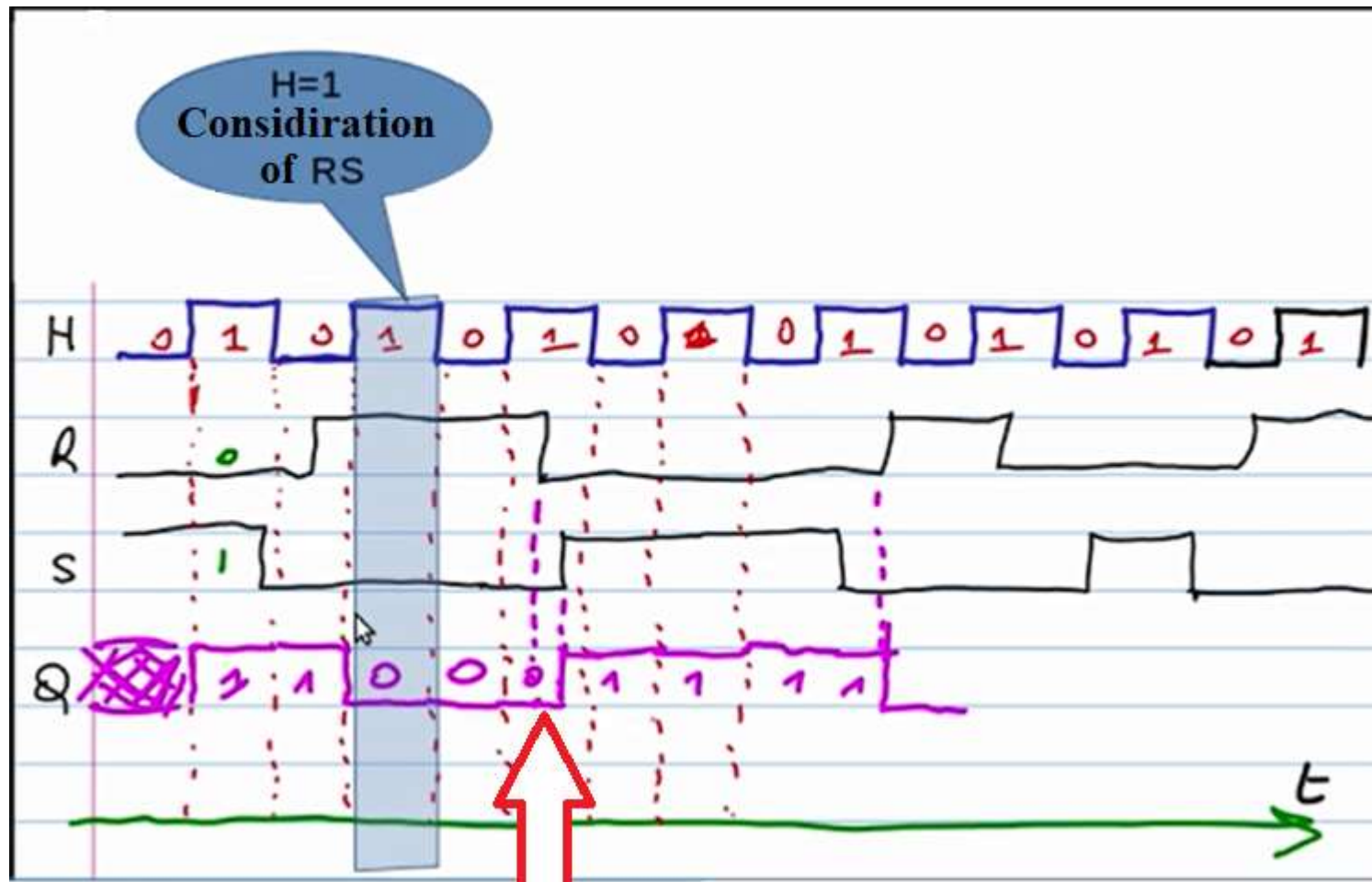
## EXAMPLE: TIMING DIAGRAM





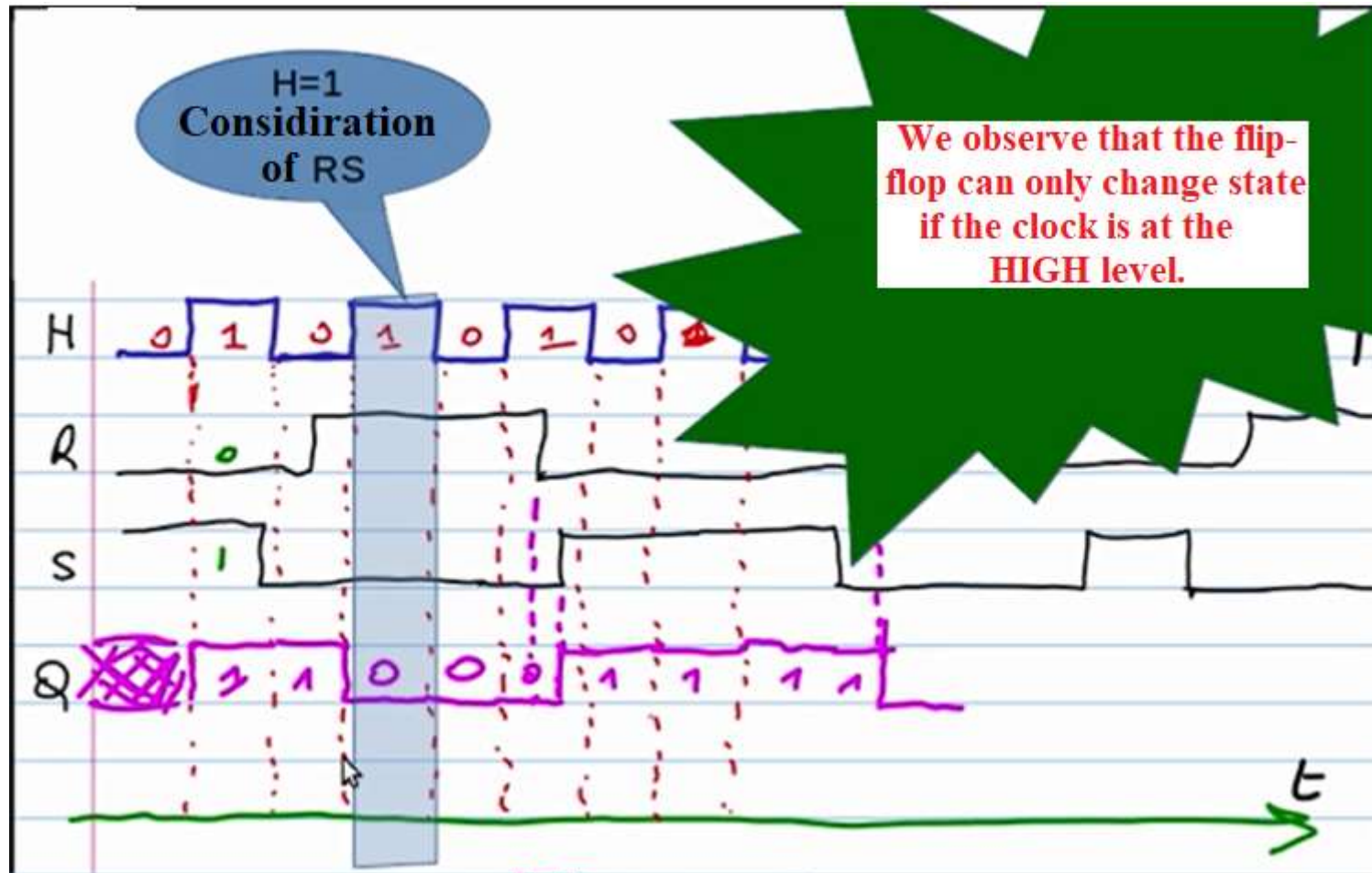
# RSH flip-flop (Synchronous RS)

## EXAMPLE: TIMING DIAGRAM



# RSH flip-flop (Synchronous RS)

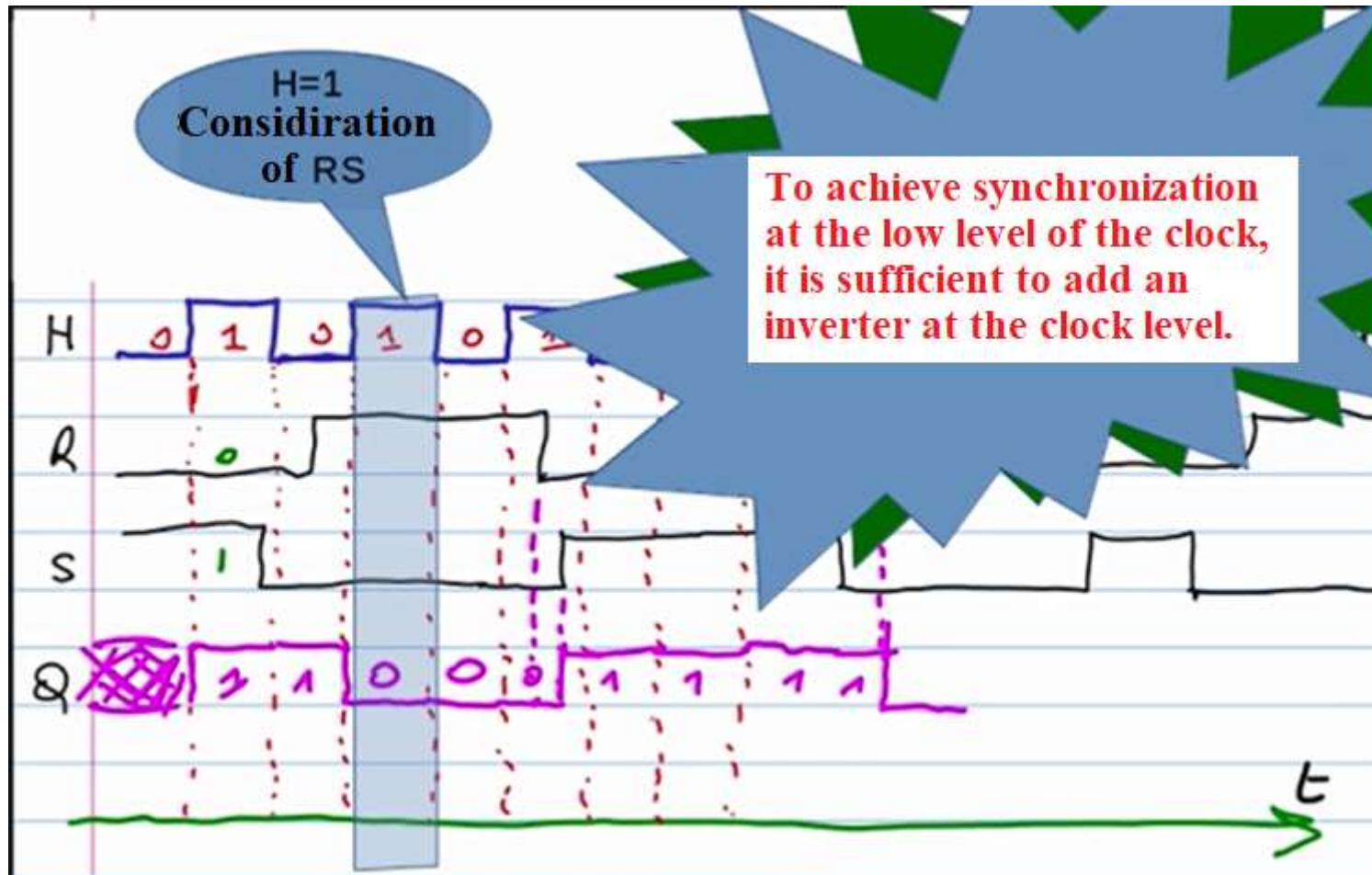
## EXAMPLE: TIMING DIAGRAM





# RSH flip-flop (Synchronous RS)

## EXAMPLE: TIMING DIAGRAM

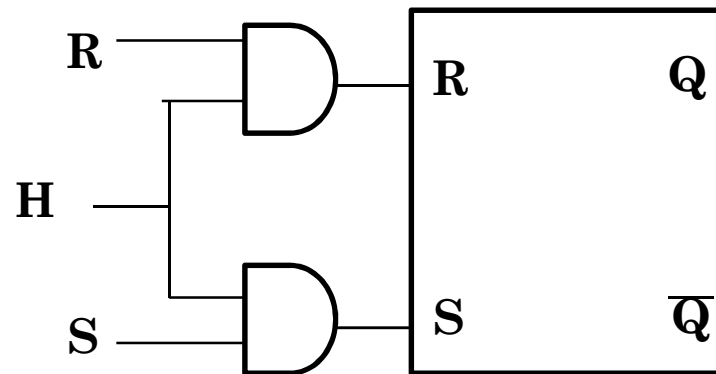


## RSH flip-flop (Synchronous RS)



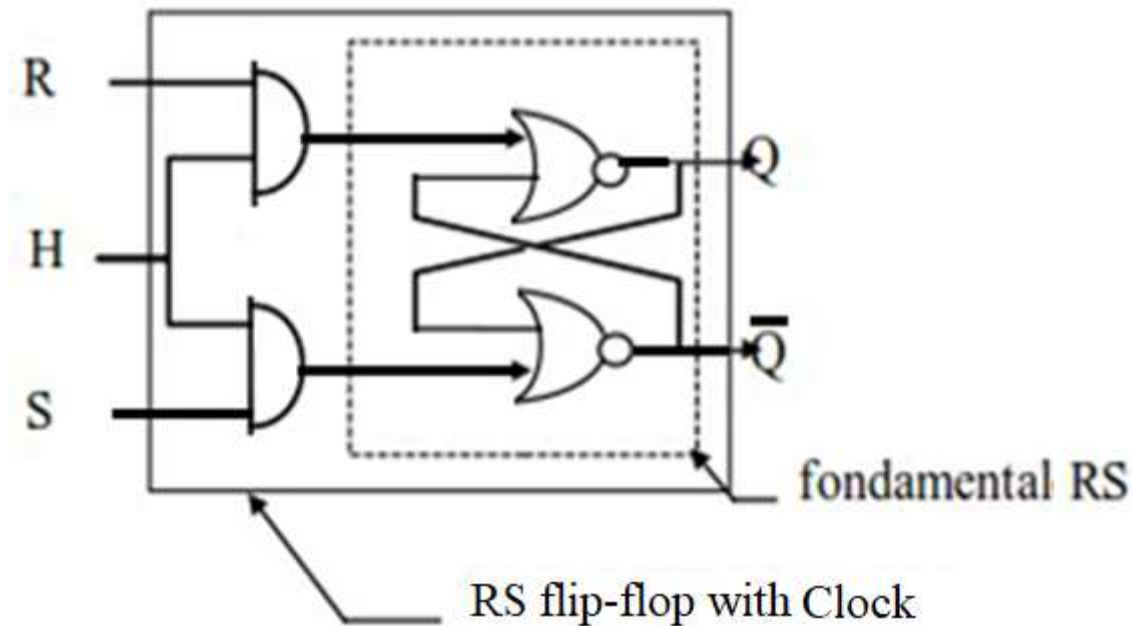
**Exercise 2:** Provide the circuit of this RSH flip-flop using the RS flip-flop.

**Solution:** Simply create a logical AND between its inputs and the clock signal, to ensure that the consideration of its inputs is synchronized by the clock signal.

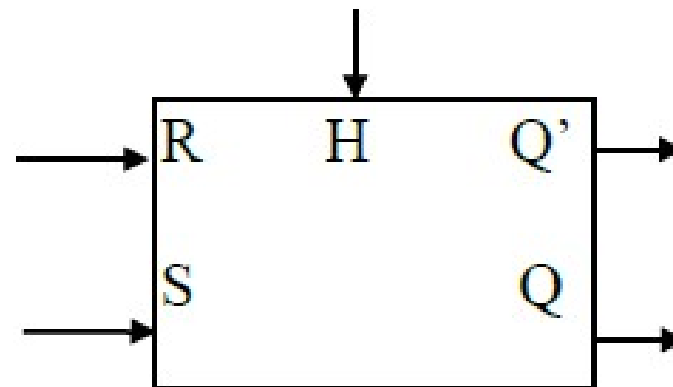


## RSH flip-flop (Synchronous RS)

**Exercise 2:** Provide the circuit of this RSH flip-flop using the RS flip-flop.



Schematically, the RS flip-flop is represented as follows:



# D flip-flop

**D Flip-Flop**: That's a synchronous latch triggered by either the high or low level.



On the high level « 1 »

if  $H = 1$  then  $Q^+ = D$



On the low level « 0 »

if  $H = 0$  then  $Q^+ = D$

$H/\bar{H}$	$Q^+$
0	$Q^-$
1	D



$H/\bar{H}$	D	$Q^+$
0	0	$Q^-$
0	1	$Q^-$
1	0	0
1	1	1

# D Flip-Flop

The D flip-flop (Data or Datum) is derived from the RSH flip-flop by adding an inverter gate between the **Set and Reset** inputs to have only one input to determine the logical level to be memorized.

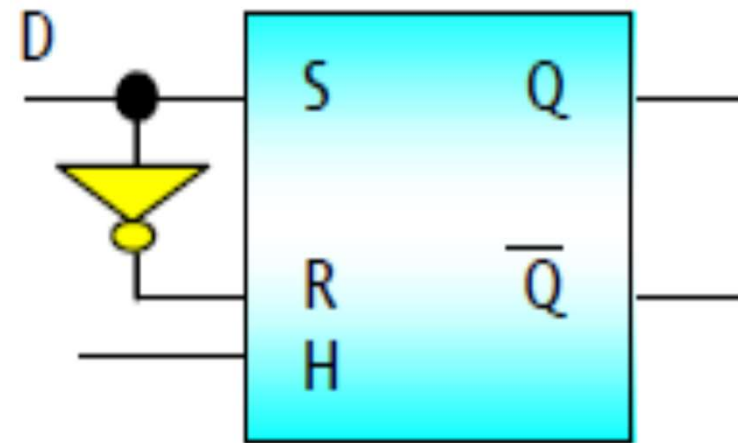
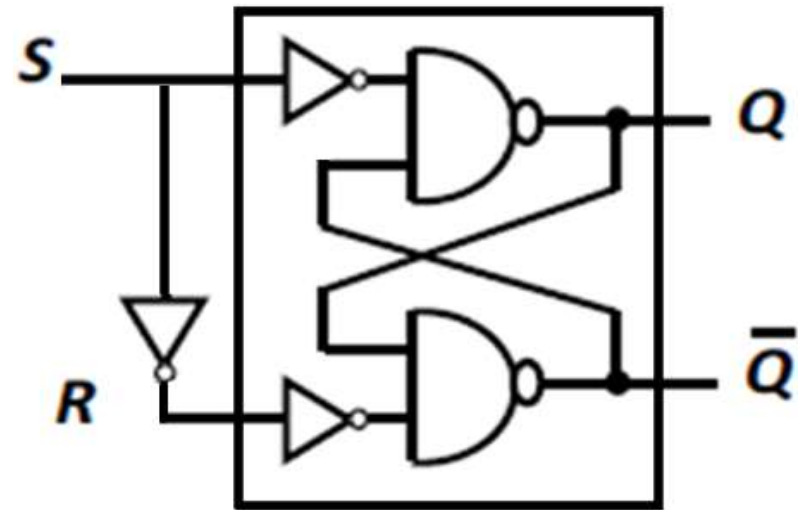
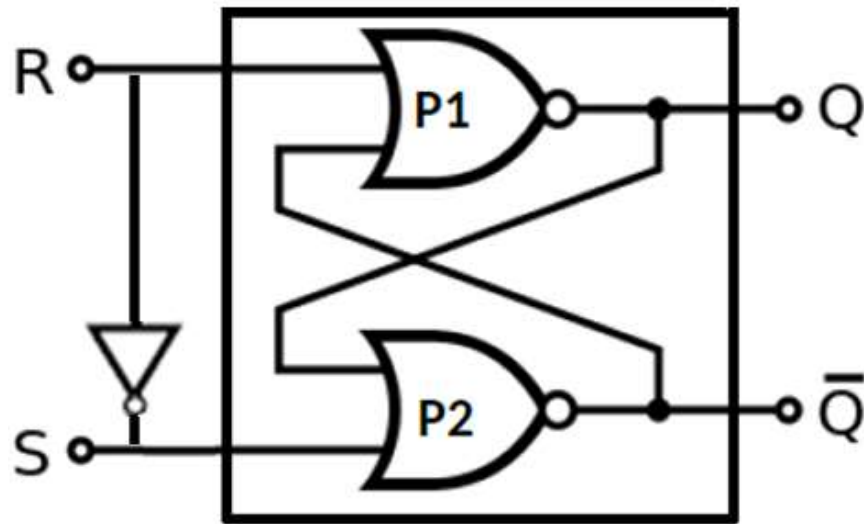
With this flip-flop, there is no longer an **invalid combination of  $S=R=1$**  (Under these conditions, the state  $R=S=1$ , which led to ambiguous operation, can no longer exist).

Two types of D flip-flops are distinguished:

Level-triggered D flip-flop (Static).

Edge-triggered D flip-flop (Dynamic),

# D Flip-Flop



## D latch or Latch D Flip-Flop (Latch: Lock)

The D flip-flop is an RSH flip-flop whose inputs R and S are complemented in order to eliminate the forbidden state ( $R=S=1$ ) and minimize the number of memory states.

- When H is active and  $D=1$ :  
the output Q will be set to 1 (set).
- When H is active and  $D=0$ :  
the output Q will be reset to 0 (reset).
- When H is not active:  
the outputs remain in the previous state (memory).

## D latch or D flip-flop with lock (Latch: Lock)

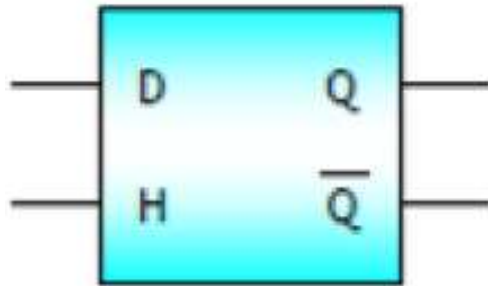
It's a synchronous static flip-flop triggered by the clock signal level, operating as follows:

- The flip-flop is transparent as long as the clock signal is at the high level (or low level).  
The output **Q** follows all variations of the input **D**.  
The latch is said to be **transparent**.
- The state of the output **Q** is locked (memorized) as long as the clock signal is at the low level (or high level).  
The output **Q** retains its logical state.  
The latch is said to be **latched**.

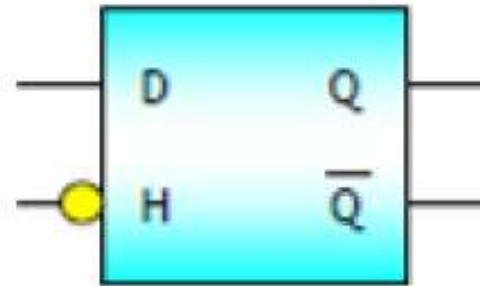


# D latch or D flip-flop with lock (Latch: Lock)

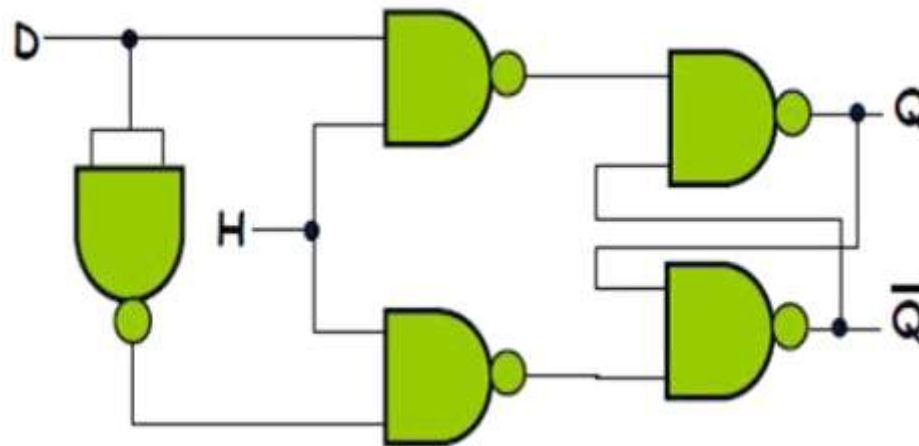
Symbol:



Active on **high**  
level of H



Active on **low**  
level of H.



## D latch or D flip-flop with lock (Latch: Lock)

<b>Truth table</b>			
<b>H</b>	<b>D</b>	<b>Q</b>	<b>Comment</b>
<b>0</b>	<b>x</b>	<b>q</b>	<b>Memorization</b>
<b>1</b>	<b>0</b>	<b>0</b>	<b>the latch copies the value of D to Q.</b>
<b>1</b>	<b>1</b>	<b>1</b>	

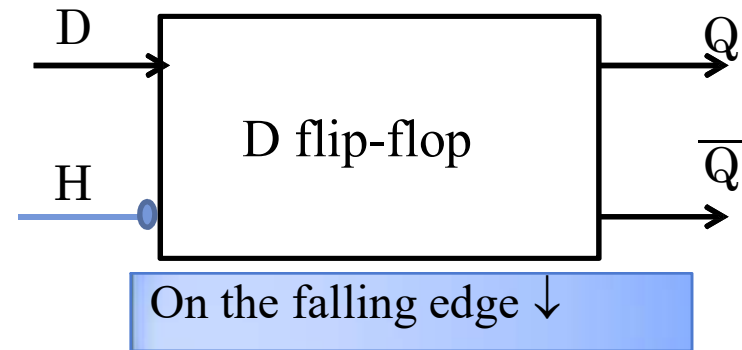
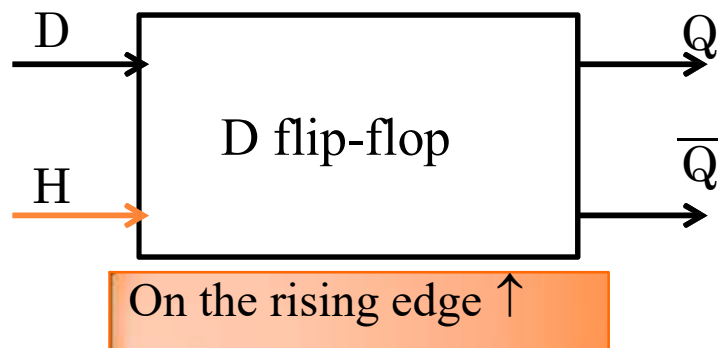
## D latch or D flip-flop with lock (Latch: Lock)

- When  $H = 1$ , the output  $Q$  is identical to  $D$  ( $Q=D$ ); this is referred to as the latch being transparent.
- When  $H = 0$ , the output  $Q$  remains at the last value of  $D$  it had before  $H$  transitioned to 0. In other words, the output is 'latched' to  $D$  and does not change as long as  $CLK$  remains at the low level, even if  $D$  changes value.

The operation of the D latch is summarized in the following truth table.

# D flip-flop

- It's a flip-flop synchronized on the rising edge or the falling edge.

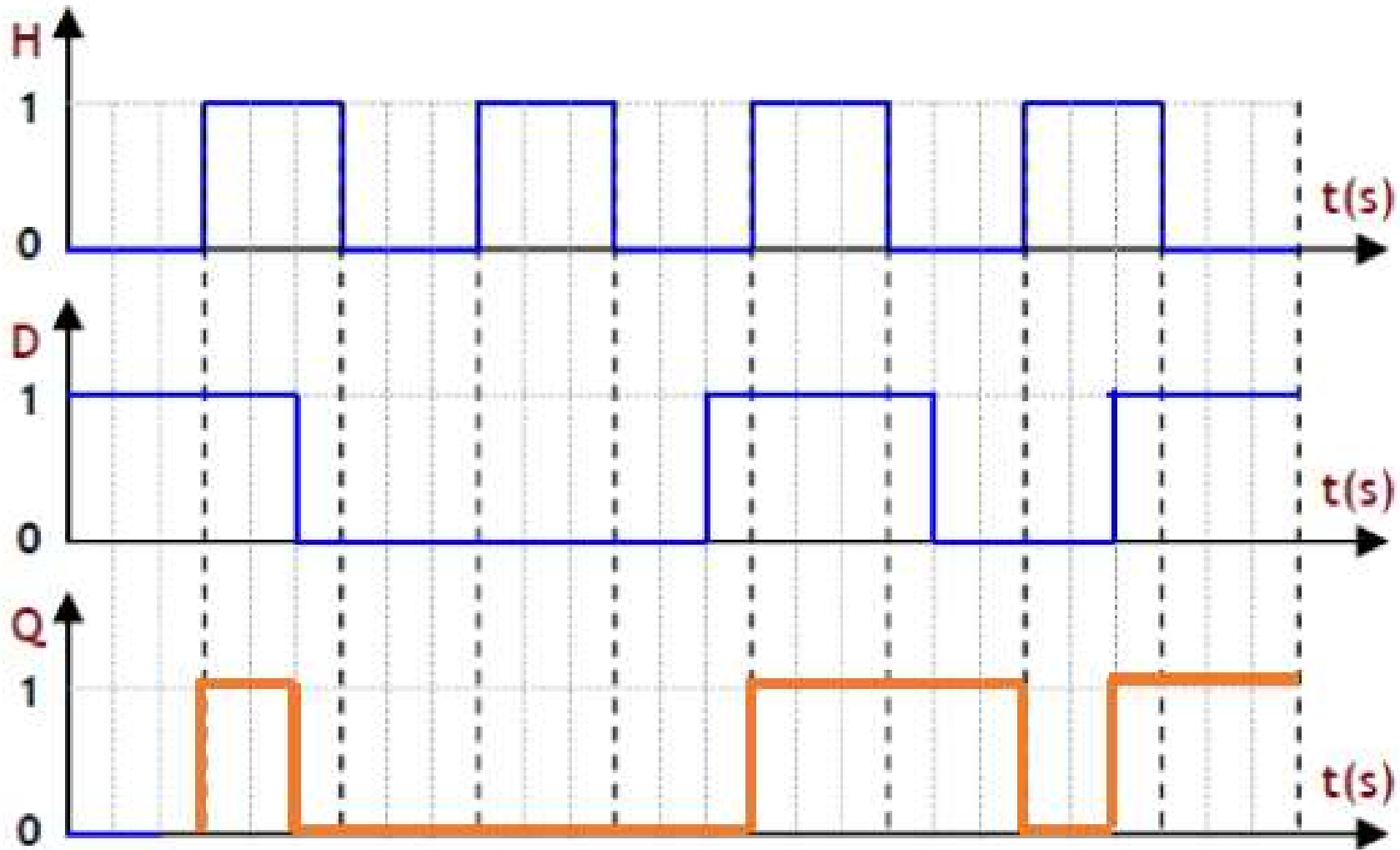


H	Q+
0/1/↓	Q-
↑	D

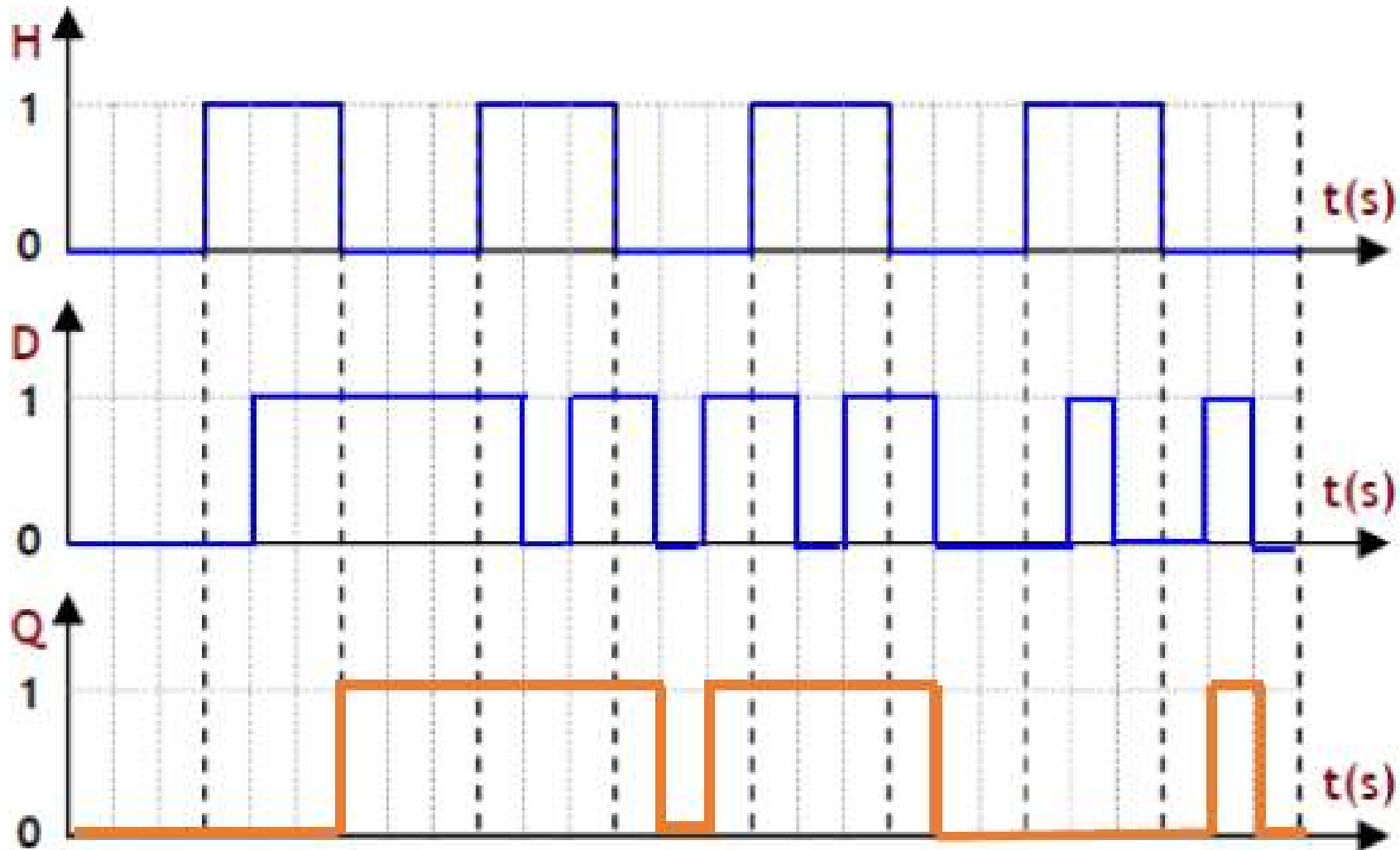


H	D	Q+
0/1/↓	0	Q-
0/1/↓	1	Q-
↑	0	0
↑	1	1

## D Latch Timing Diagram (H Active High)



## D Latch Timing Diagram (H Active Low)





# D flip-flop

Exercise 3: Transform an RSH flip-flop to act as a D latch (active high)

H	D	Q+
0	0	Q-
0	1	Q-
1	0	0
1	1	1

H	R	S	Q+
0	X	X	Q-
1	0	0	Q-
1	0	1	1
1	1	0	0
1	1	1	X

$$H_D = H_{RSH}, R = \bar{D}; S = D$$



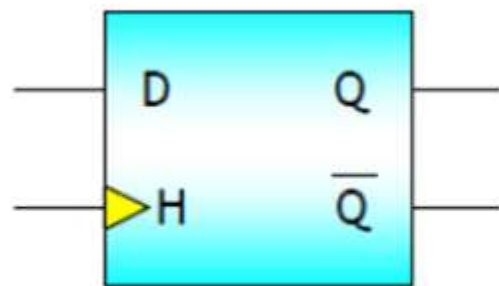
## A D flip-flop or a flip-flop with edge-triggered control.

This is a synchronous **dynamic** flip-flop on the clock edge, operating as follows:

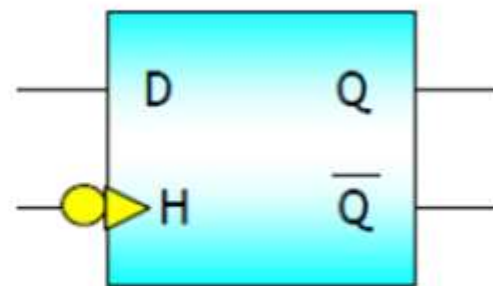
In the presence of the active clock edge, the flip-flop copies the logical state of input **D** to output **Q**.

In the absence of the active clock edge, the flip-flop retains its logical state at output **Q**.

Symbol :



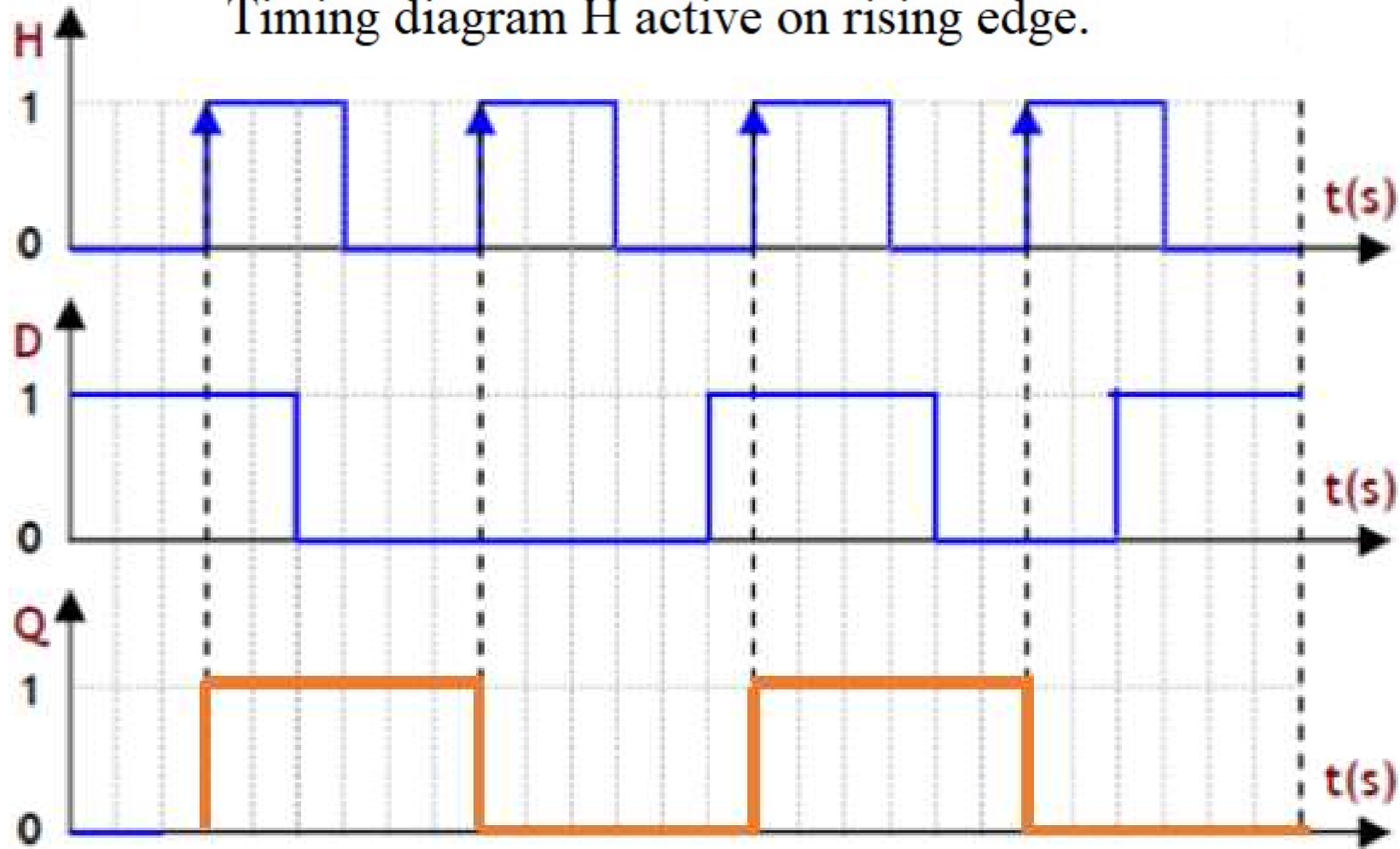
Active on  $\uparrow$  of H



Active on  $\downarrow$  of H

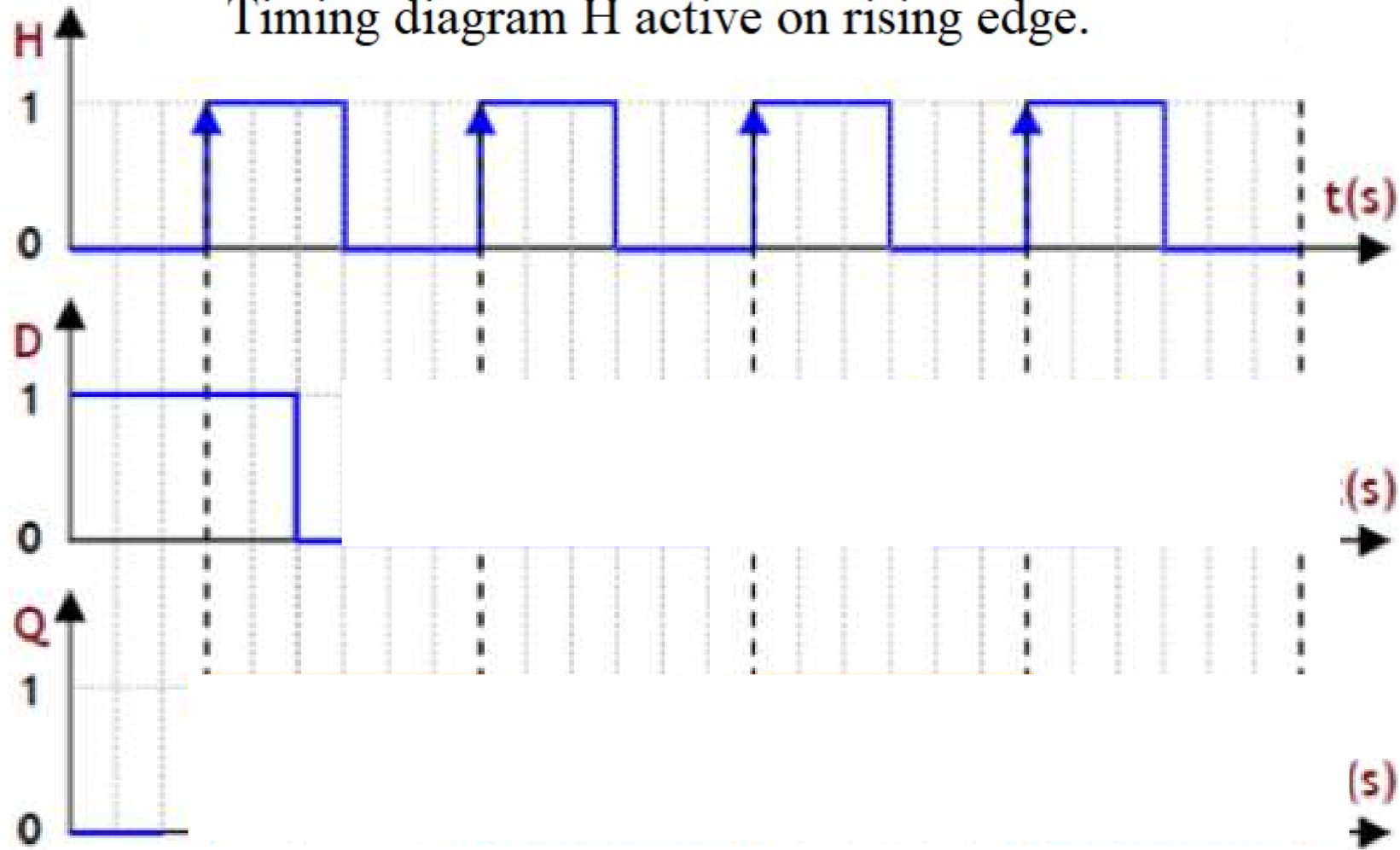
# A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.



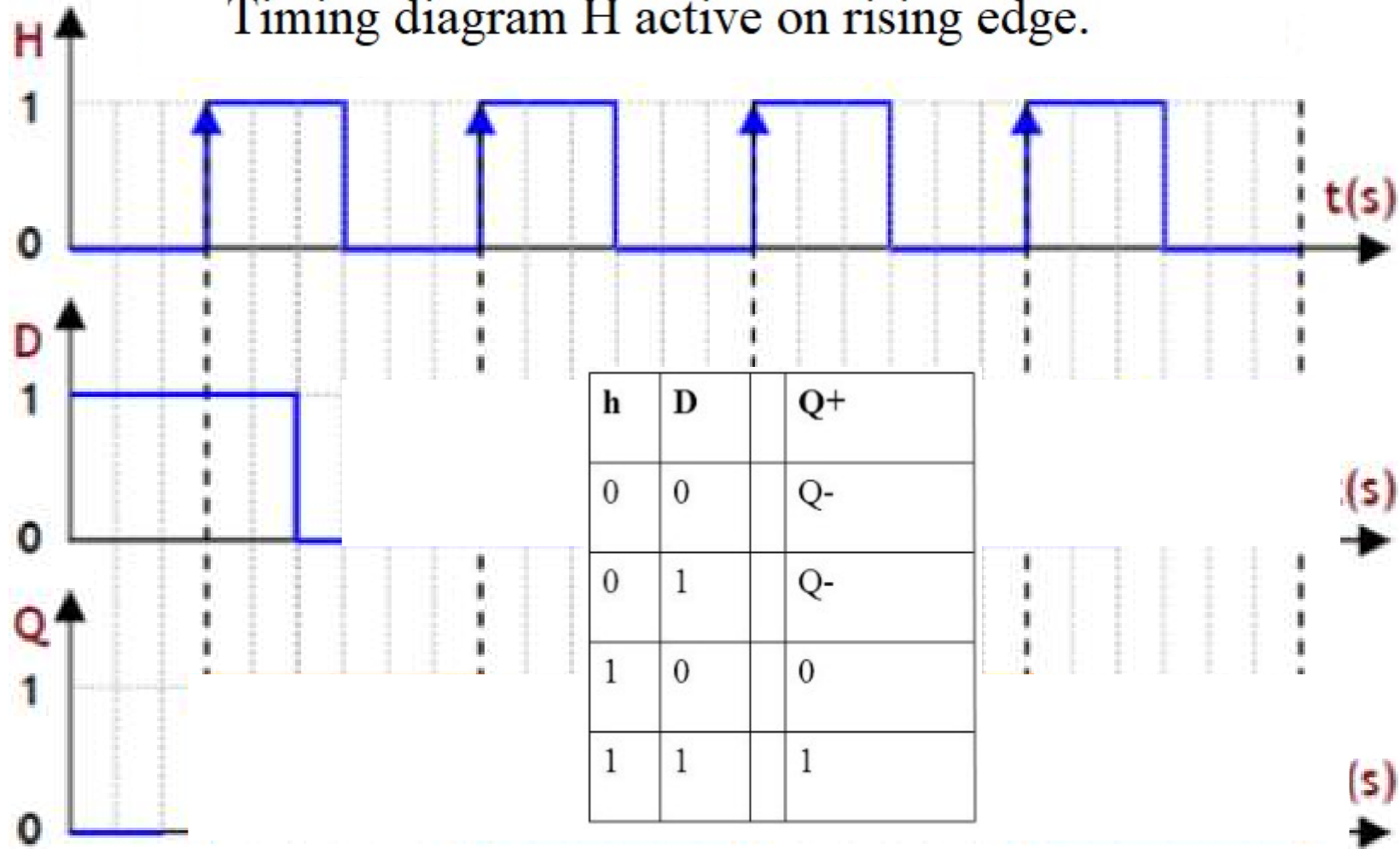
A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.



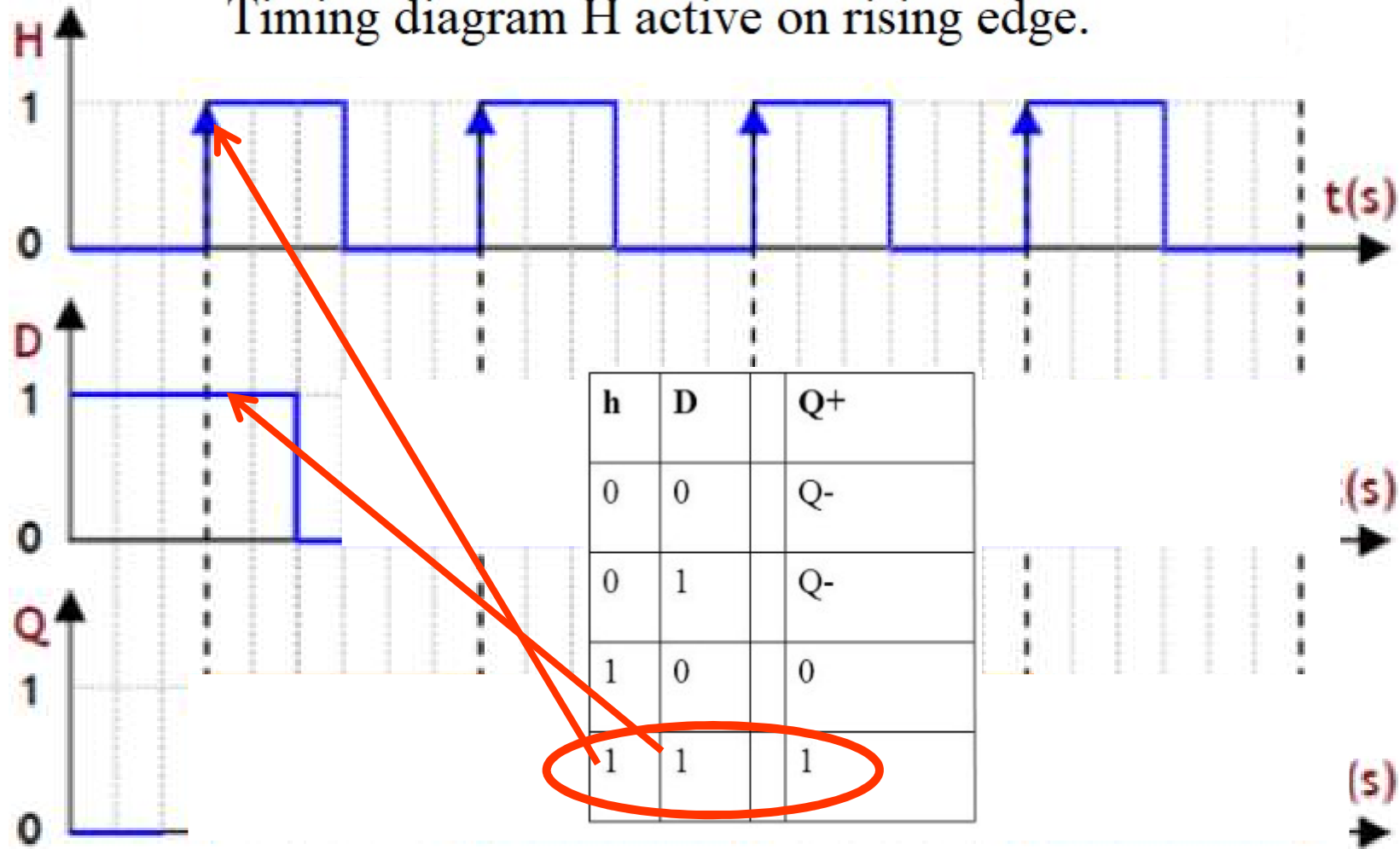
# A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.



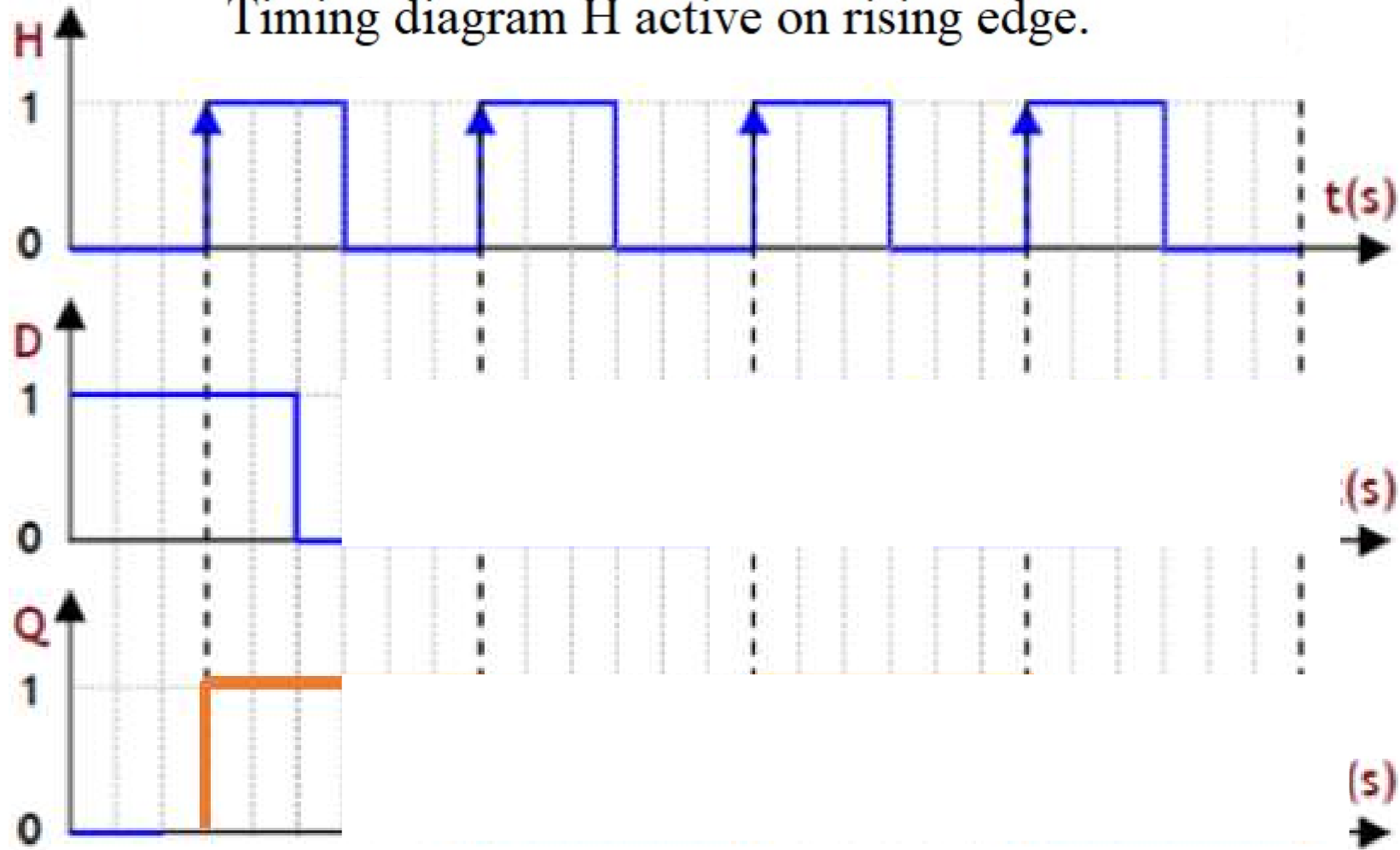
# A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.



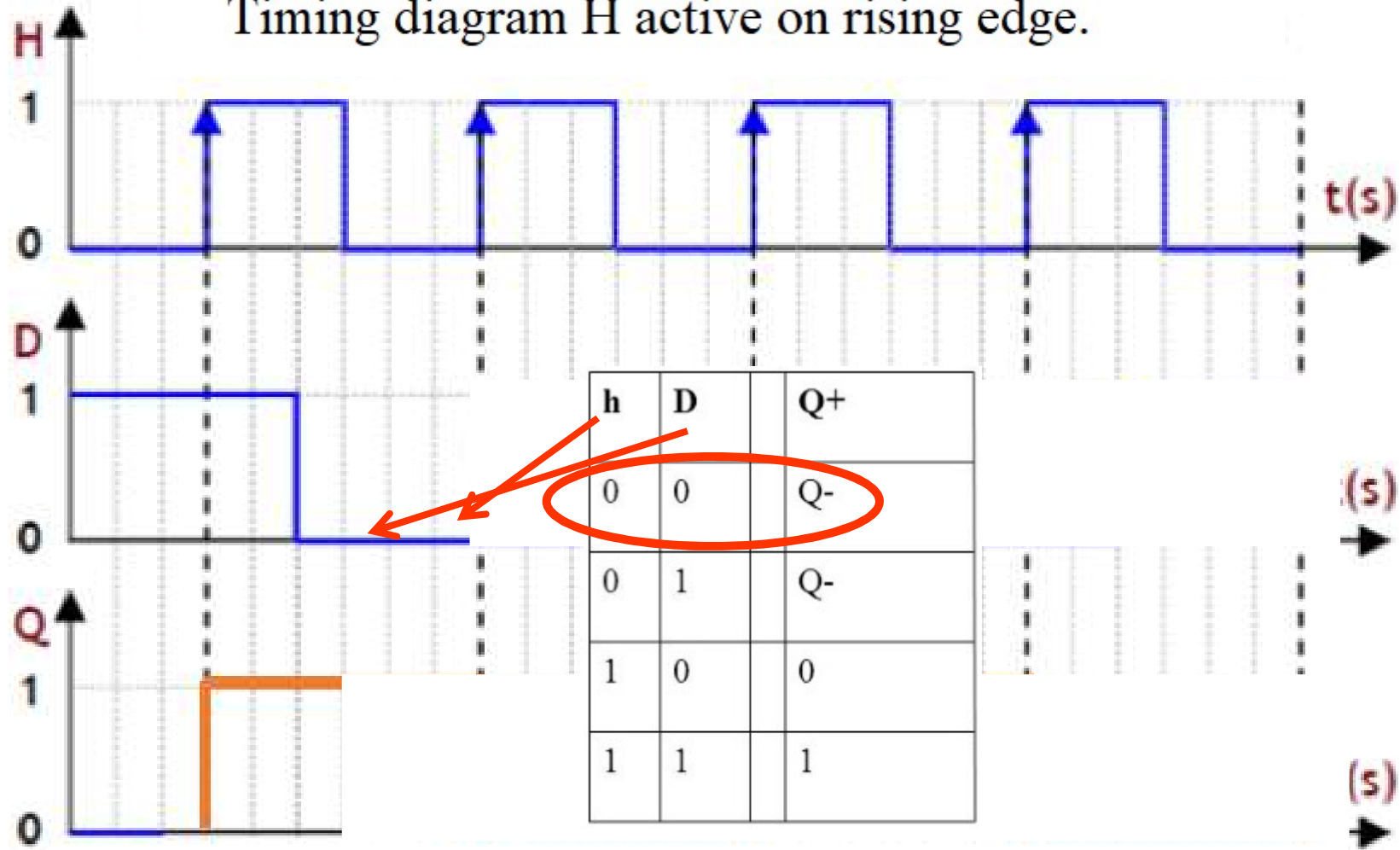
# A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.



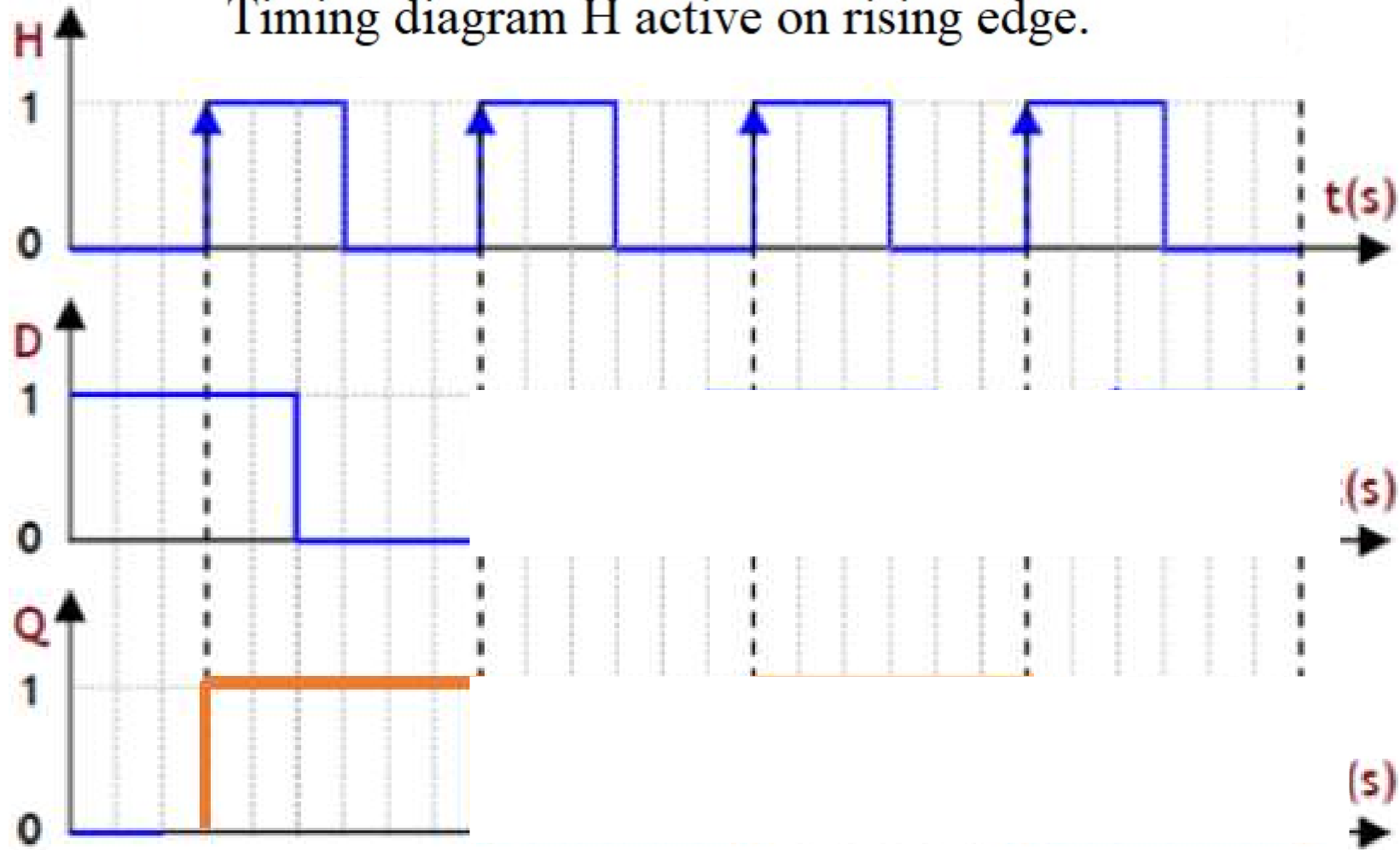
# A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.



# A D flip-flop or a flip-flop with edge-triggered control.

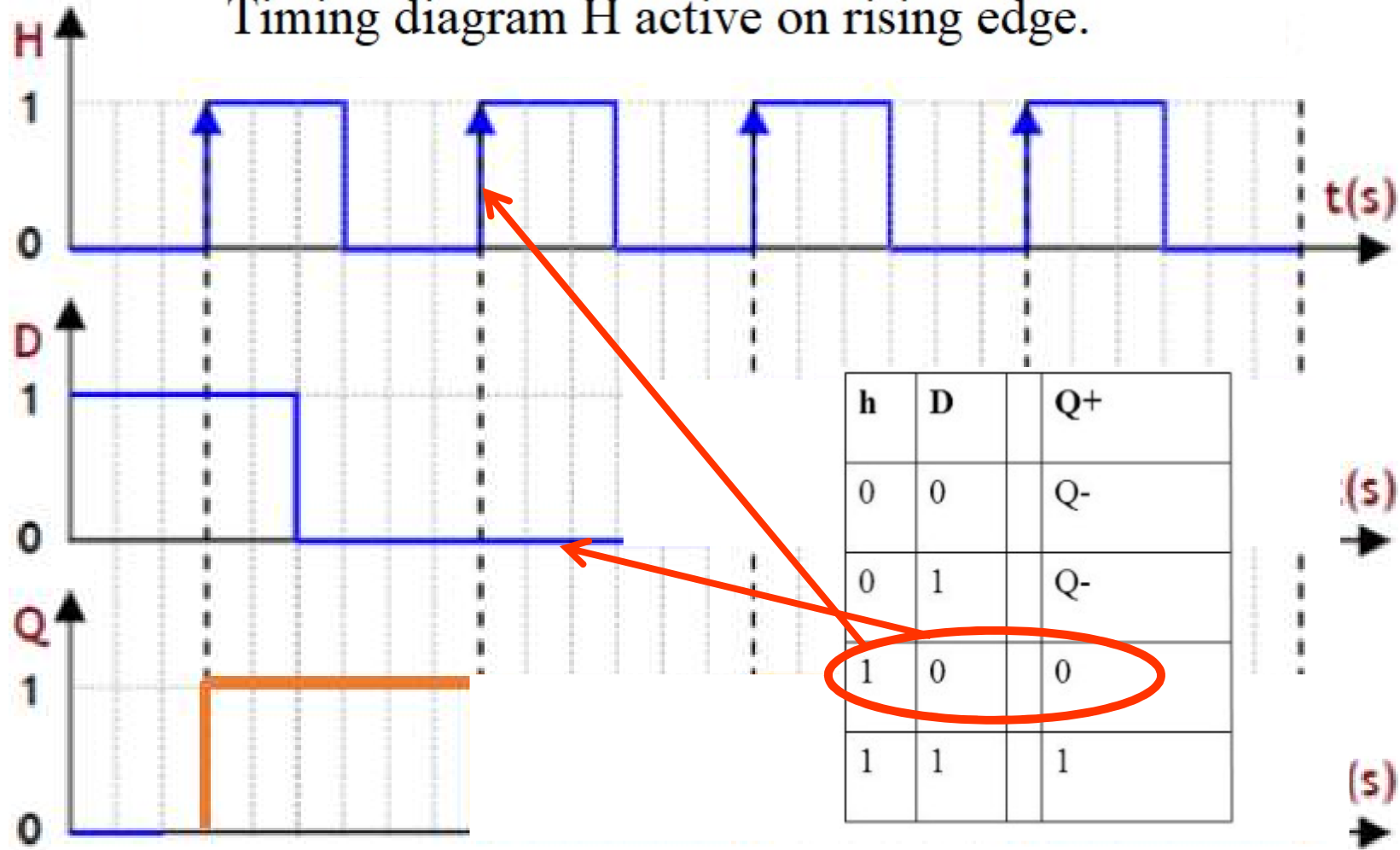
Timing diagram H active on rising edge.





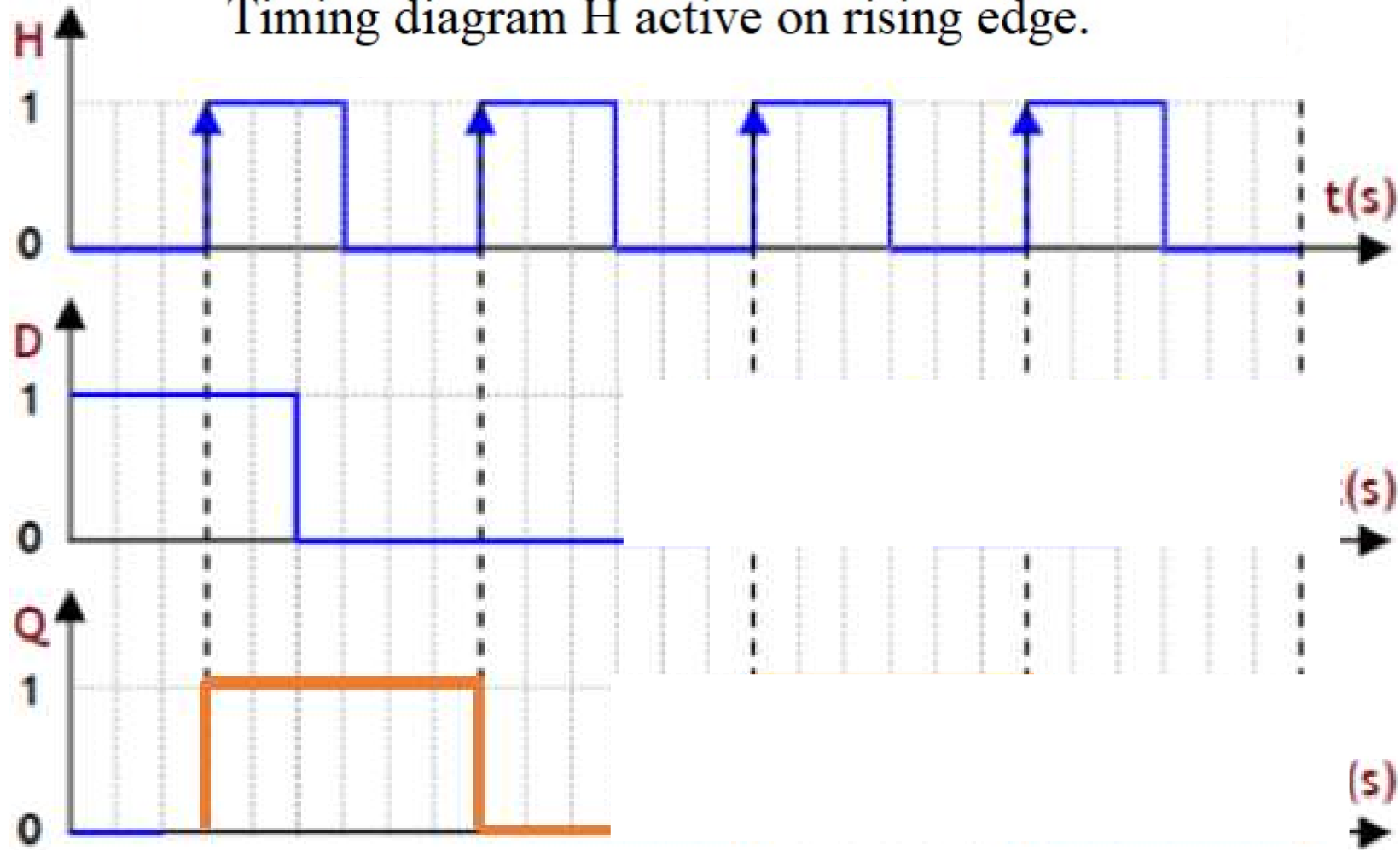
# A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.



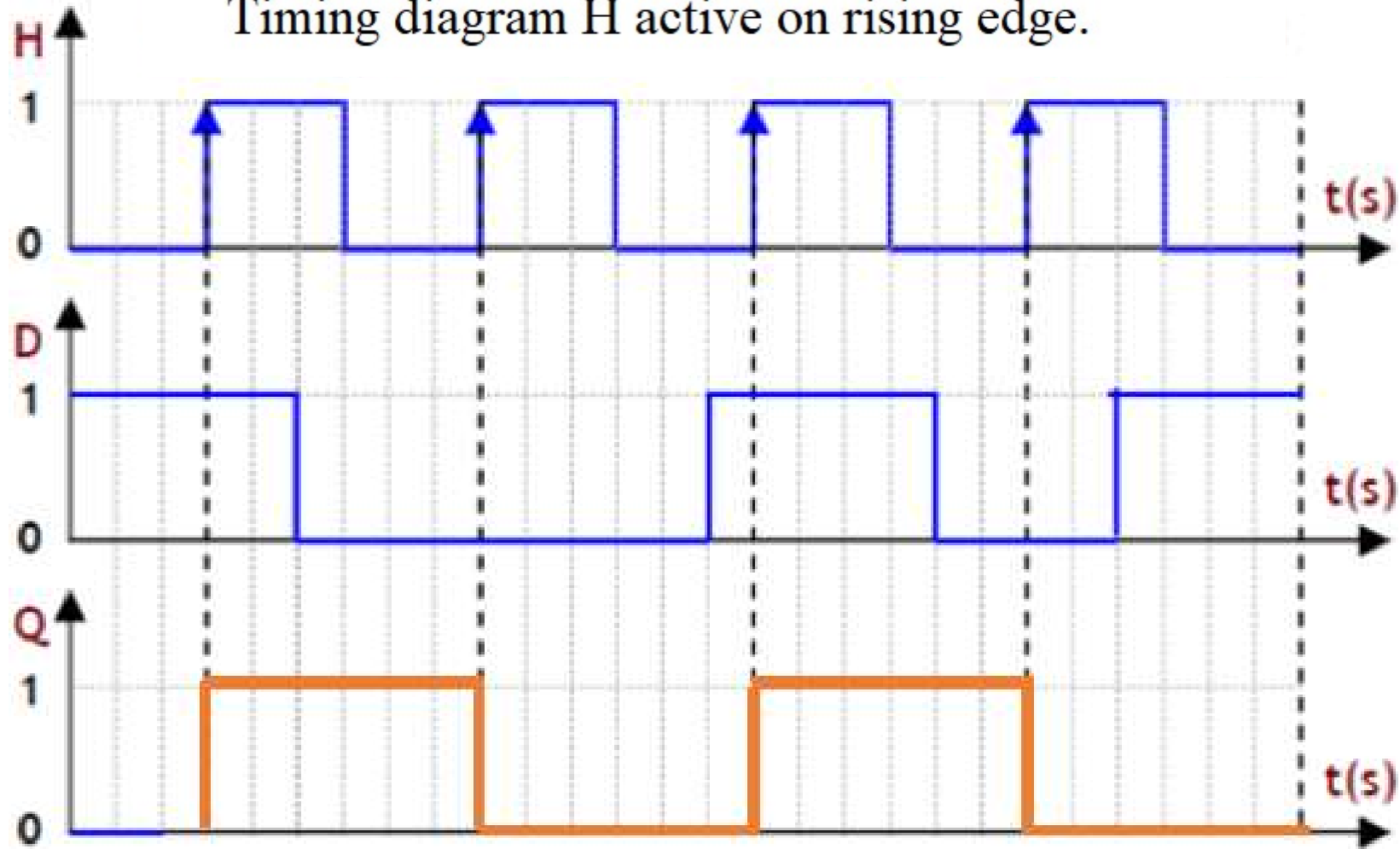
A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.



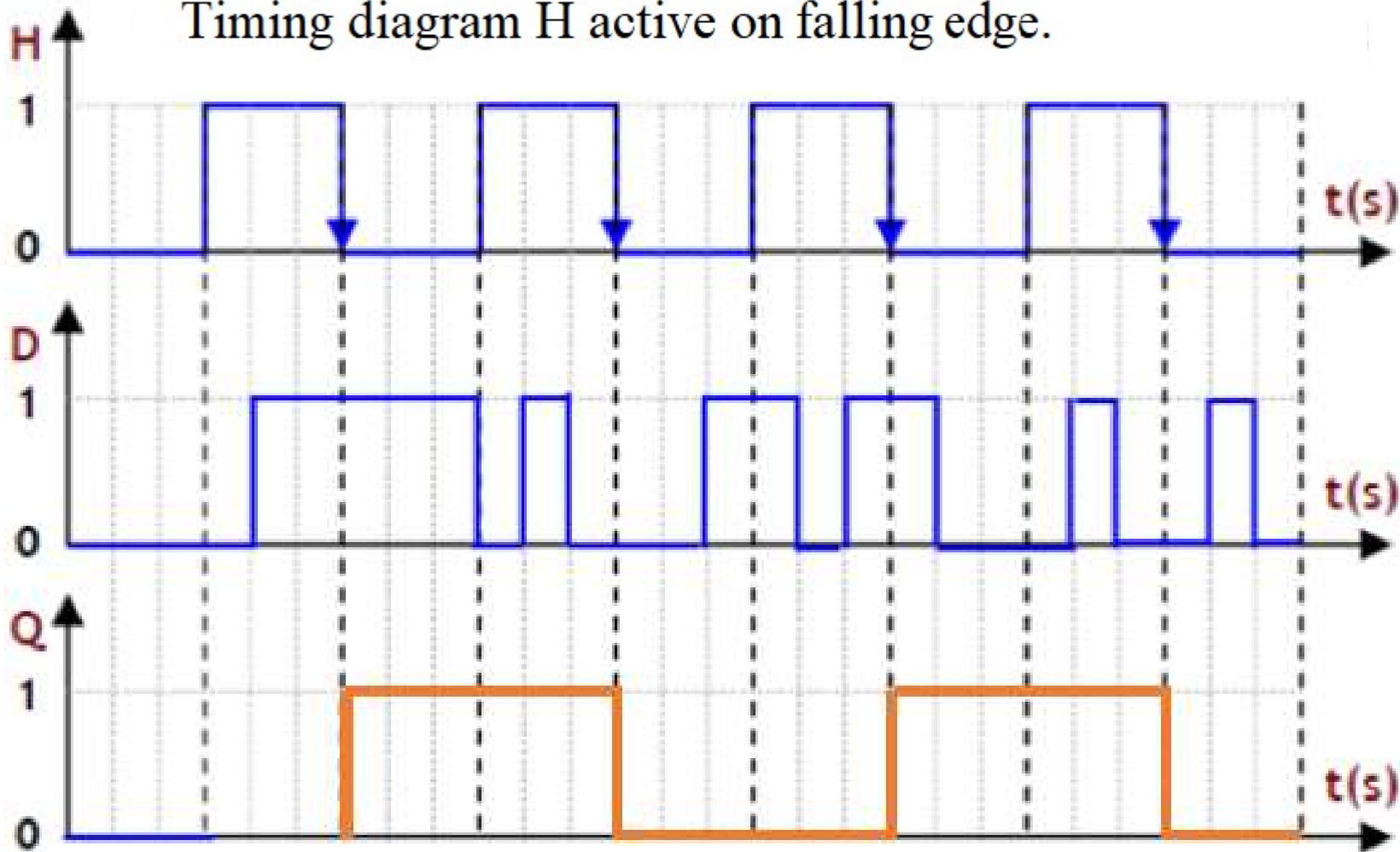
# A D flip-flop or a flip-flop with edge-triggered control.

Timing diagram H active on rising edge.

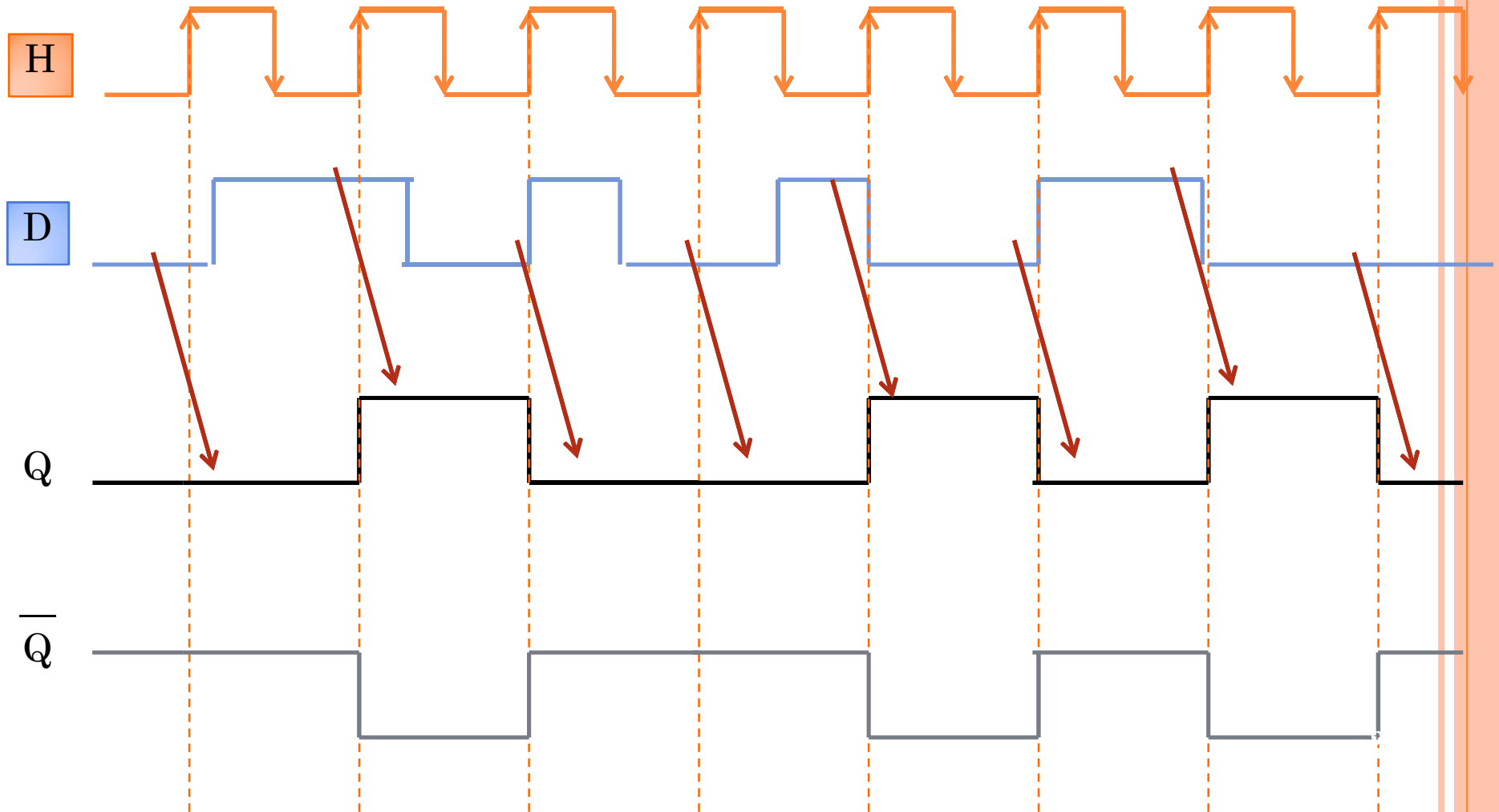


# A D flip-flop or an edge-triggered flip-flop with falling edge control.

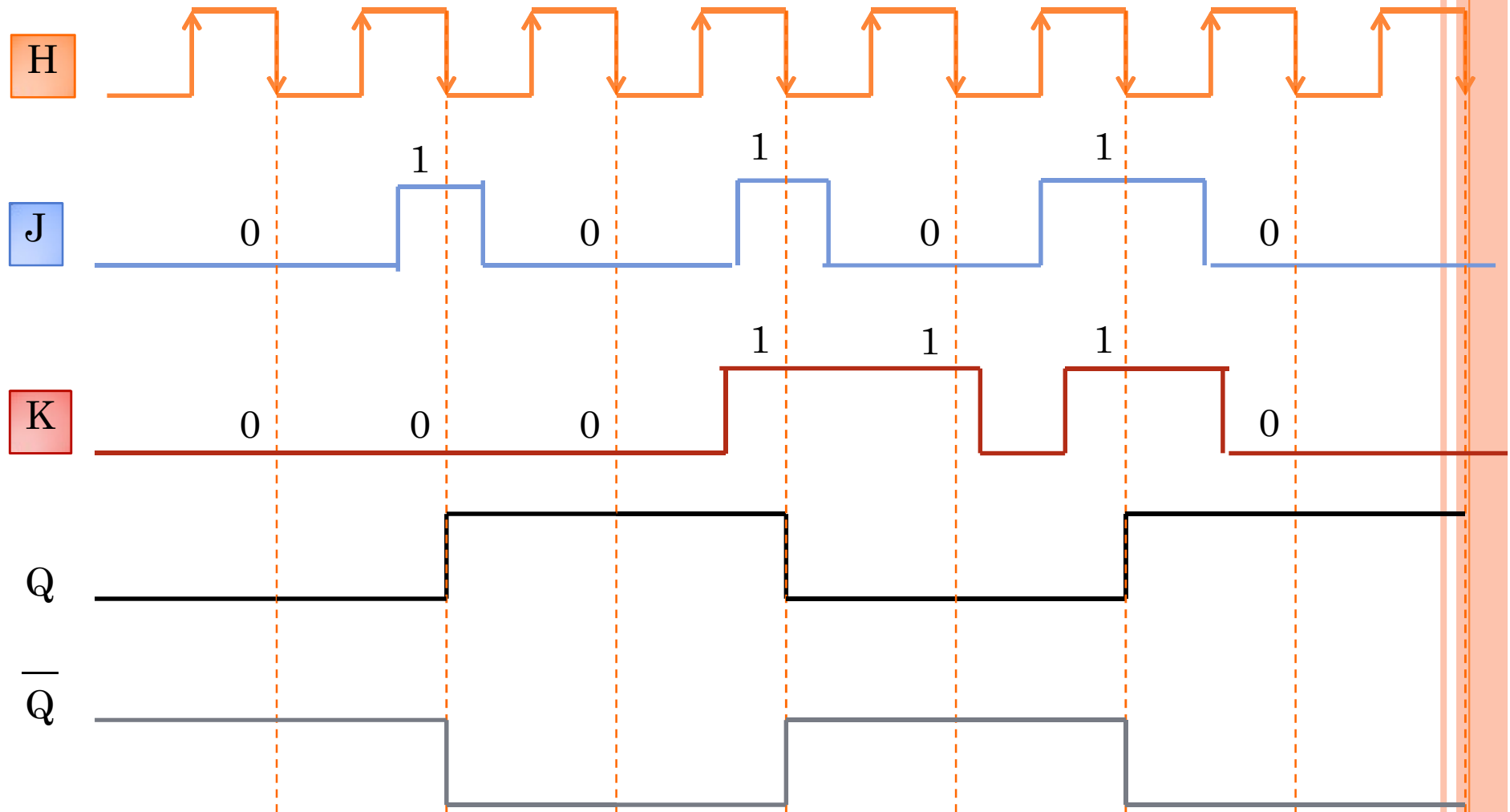
Timing diagram H active on falling edge.



# D flip-flop edge-triggered with Rising edge control

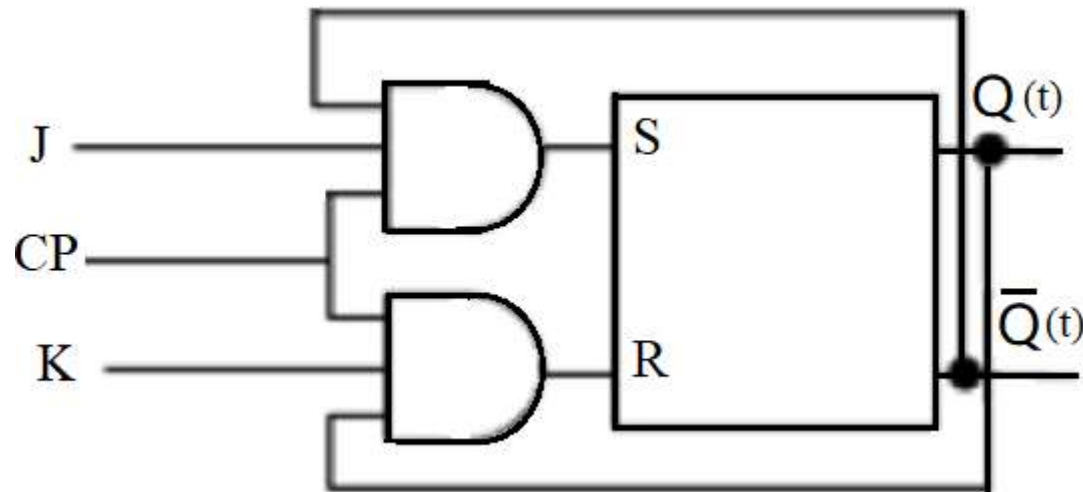


# JK flip-flop edge-triggered with falling edge control



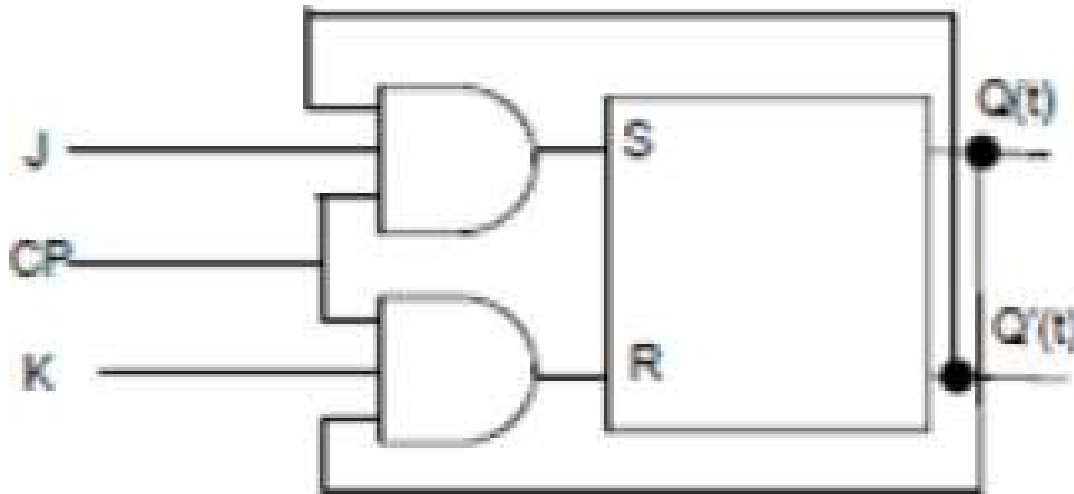
## JK flip-flop

The JK flip-flop is an improvement over the RS flip-flop. The undetermined case of the latter (S and R both equal to 1) is defined in the JK flip-flop by inverting the states of the outputs by connecting the inputs R and S to the outputs.  $S=J\bar{q}$  et  $R=Kq$



## JK flip-flop

The JK flip-flop thus eliminates the ambiguity that exists for the combination  $S=R=1$  in the RSH flip-flop.



Output equation

Q \ JK	00	01	11	10
0	0	0	1	1
1	1	0	0	1

$$Q(t+1) = J\bar{Q} + \bar{K}Q$$



## JK flip-flop

The JK flip-flop thus eliminates the ambiguity that exists for the combination  $S=R=1$  in the RSH flip-flop.

J	K	Q(t)	S	R	Q(t+1)	Comment	Comment
0	0	0	0	0	Q(t)	Q(t)	No change,
0	0	1	0	0	Q(t)		
0	1	0	0	0	Q(t)=0	0	Reset to 0
0	1	1	0	1	0		
1	0	0	1	0	1	1	Set to 1
1	0	1	0	0	Q(t)=1		
1	1	0	1	0	1	Q'(t)	Inverted state
1	1	1	0	1	0		

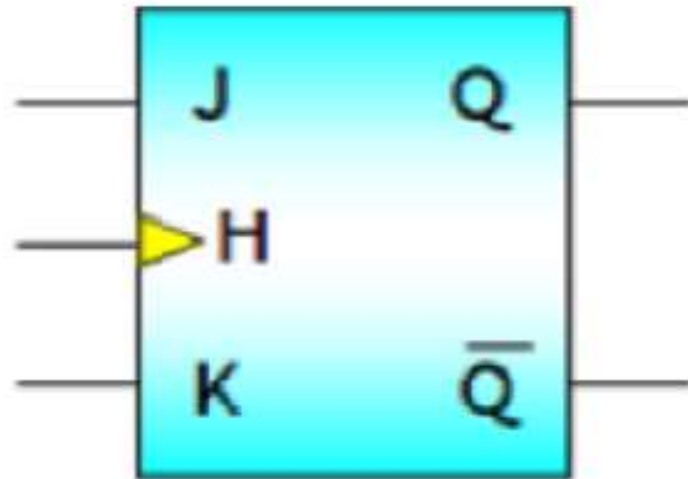
## JK Flip-flop

The JK flip-flop is a dynamic synchronous flip-flop with two inputs controlling the state of the flip-flop, and a synchronization input H:

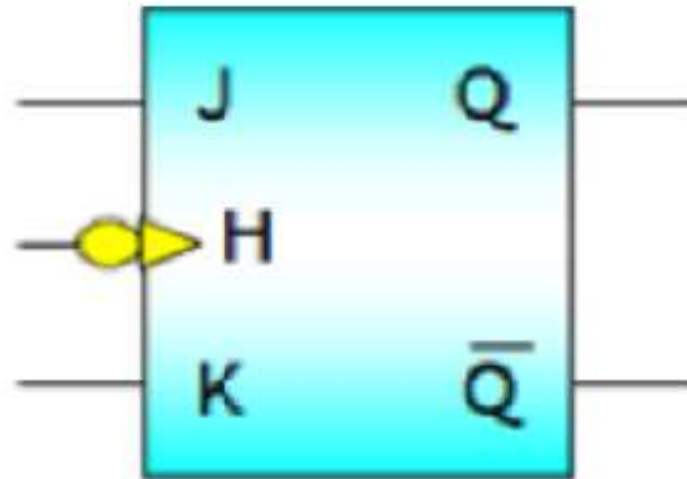
- The **J** input (Jump, set to one) acts as the **S** input of the RSH flip-flop.
- The **K** input (Kill, reset to zero) acts as the **R** input of the RSH flip-flop.
- In the absence of the clock signal, the flip-flop retains the previous state of the output **Q** (memory).
- For the combination  $J=K=0$ , the flip-flop remembers the state of the output **Q** at each active clock edge (rising or falling edge).

# JK Flip-flop

Symbol :



Active on  $\uparrow$  of H



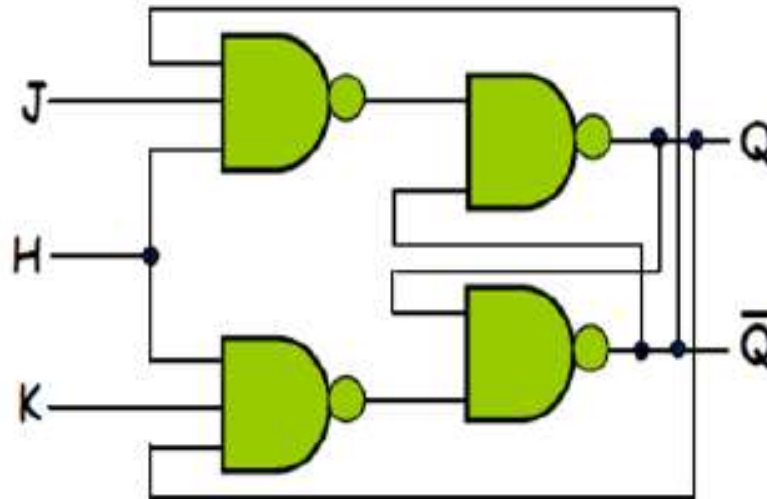
Active on  $\downarrow$  of H

## JK Flip-flop

When  $J=\bar{K}$ , the output  $Q$  copies the state of the  $J$  input at each active clock edge.

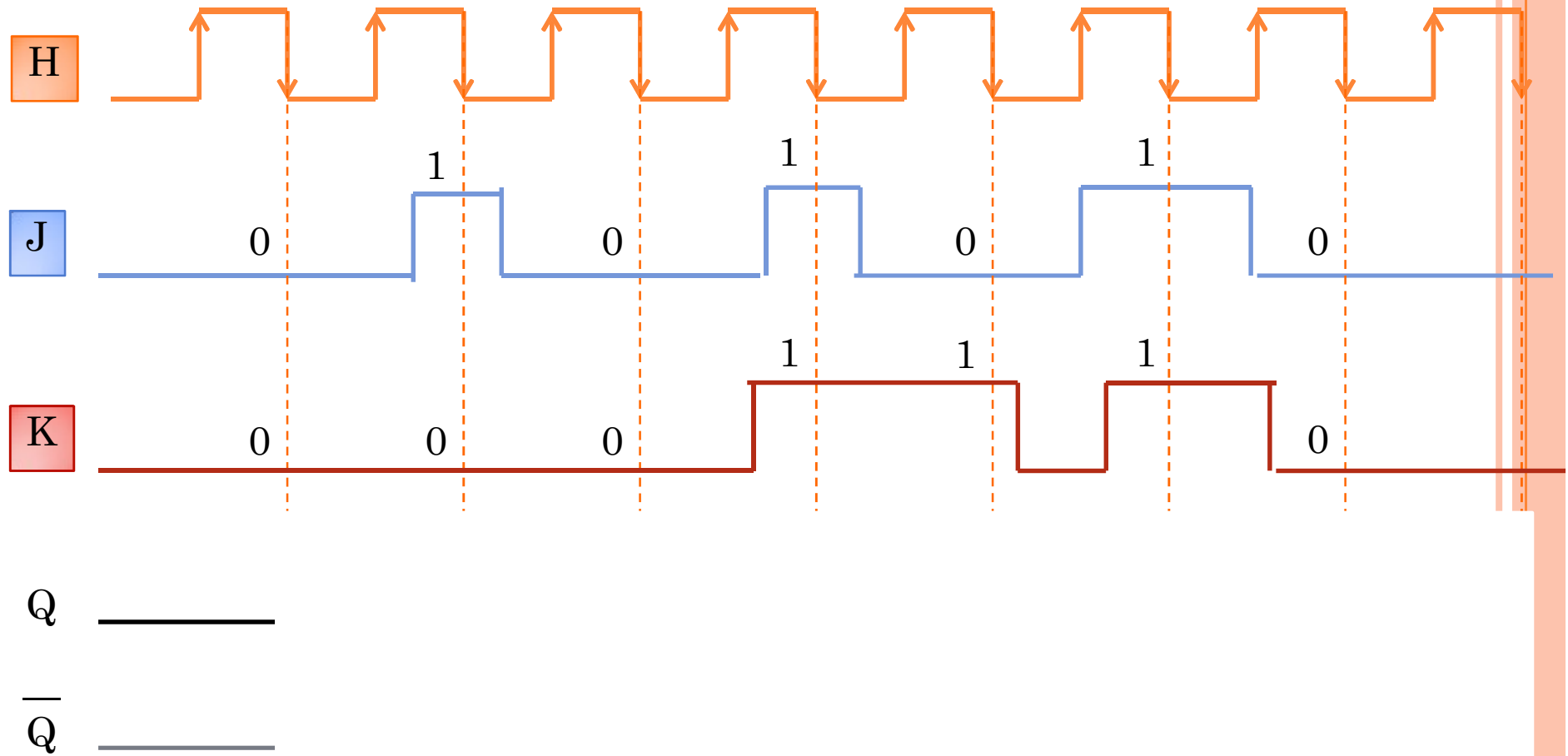
- For the combination  $JK=10$ , the output  $Q$  is set to  $1$  at each active clock edge.
- For the combination  $JK=01$ , the output  $Q$  is set to  $0$  at each active clock edge.
- With simultaneous action on  $J$  and  $K$  ( $J=K=1$ ), the flip-flop changes state at each active clock edge, which is the **toggle** mode.

# Synchronous JK flip-flop with rising clock edge

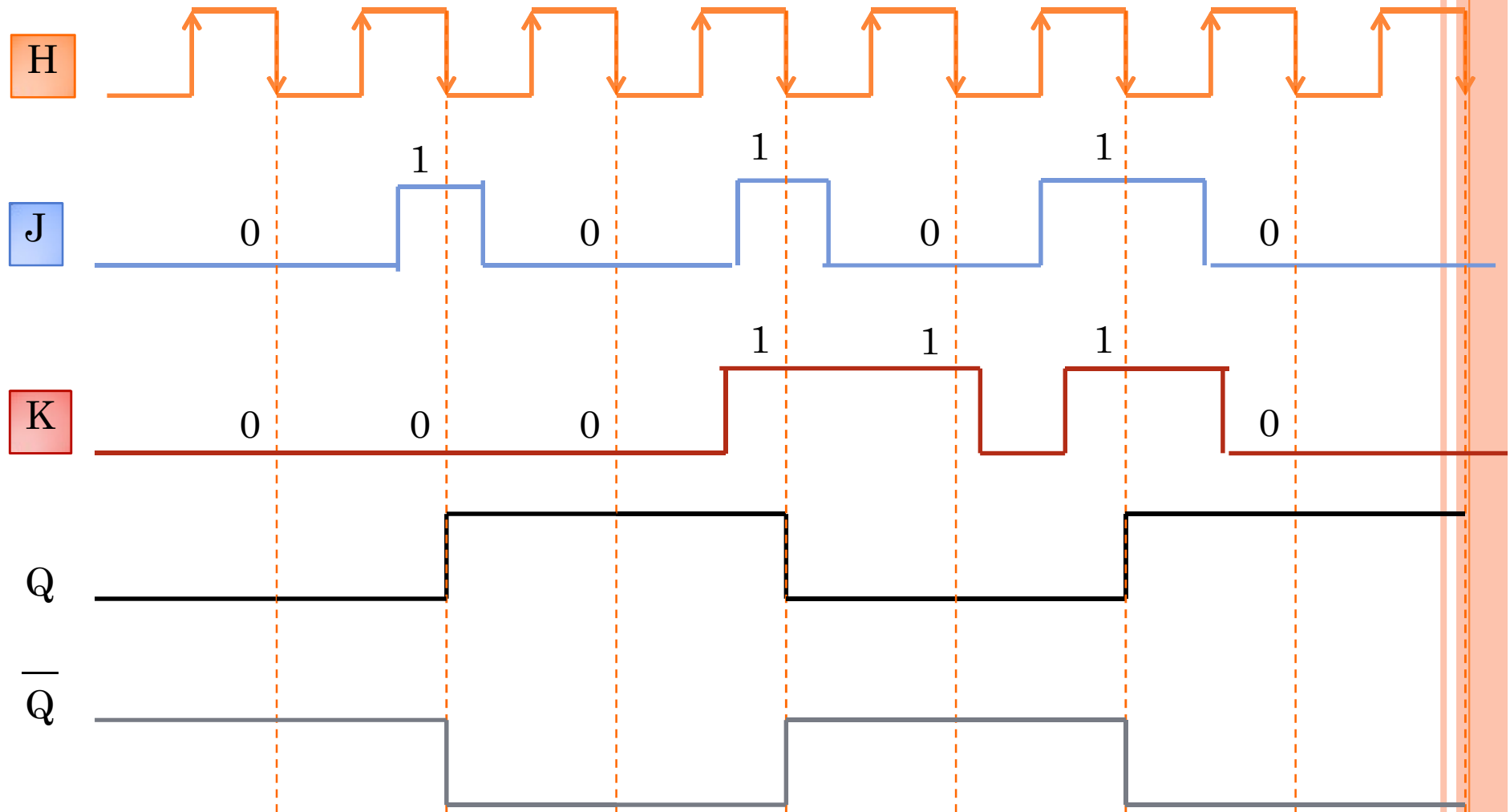


The truth table				
H	J	K	Q	Comment
↑	0	0	q	Memory
↑	0	1	0	Resetting the output Q
↑	1	0	1	Setting the output Q to 1
↑	1	1	$\bar{q}$	Toggling the output state

# JK flip-flop edge-triggered with falling edge control



# JK flip-flop edge-triggered with falling edge control



# JK flip-flop

## ASYNCHRONOUS

It's a variant flip-flop of RS where it takes into account the case  $R=S=1$ .



J	K	Q+	
0	0	Q-	Memory state
0	1	<b>0</b>	Reset
1	0	<b>1</b>	Set
1	1	$\overline{Q}$ -	Switching



# JK flip-flop SYNCHRONOUS

It's a flip-flop with two inputs J and K, and a Clock (rising or falling edge).



On Rising edge  $\uparrow$

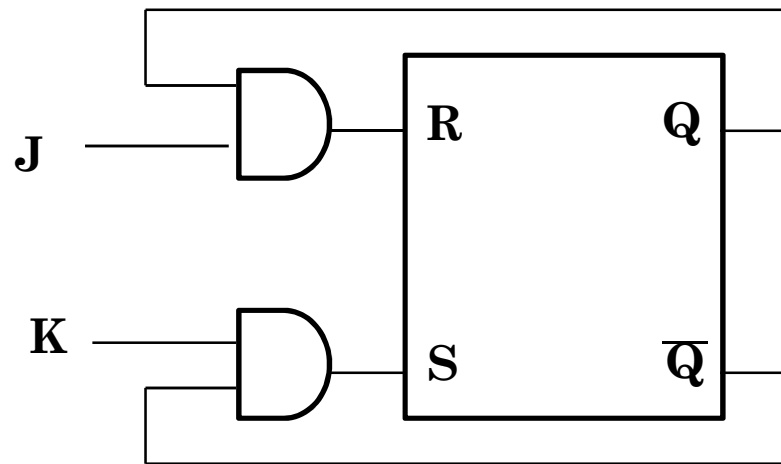


On Falling edge  $\downarrow$

H	J	K	Q+
0/1, $\uparrow$	X	X	Q-
$\downarrow$	0	0	Q-
$\downarrow$	0	1	0
$\downarrow$	1	0	1
$\downarrow$	1	1	$\overline{Q}$ -

# JK flip-flop ASYNCHRONOUS

**Exercise 05:** Implementing an asynchronous JK flip-flop using an RS flip-flop.



# JK flip-flop

## ASYNCHRONOUS

**Exercise 05:** Implementing an asynchronous JK flip-flop using an RS flip-flop.

R	S	Q-	Q+
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
<b>1</b>	<b>1</b>	<b>0</b>	<b>X</b>
<b>1</b>	<b>1</b>	<b>1</b>	<b>X</b>

J	K	Q-	Q+
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
<b>1</b>	<b>1</b>	<b>0</b>	<b>1</b>
<b>1</b>	<b>1</b>	<b>1</b>	<b>0</b>

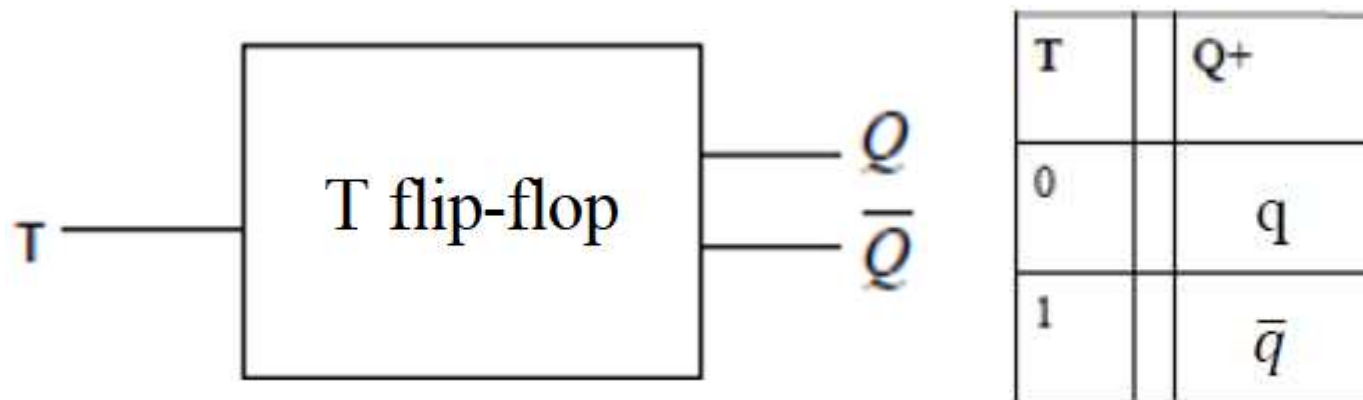
## T Flip-flop

The T (toggle, Trigger) flip-flop operates on a clock edge.

It allows to maintain the previous output value or to invert it (complement or not the current state).

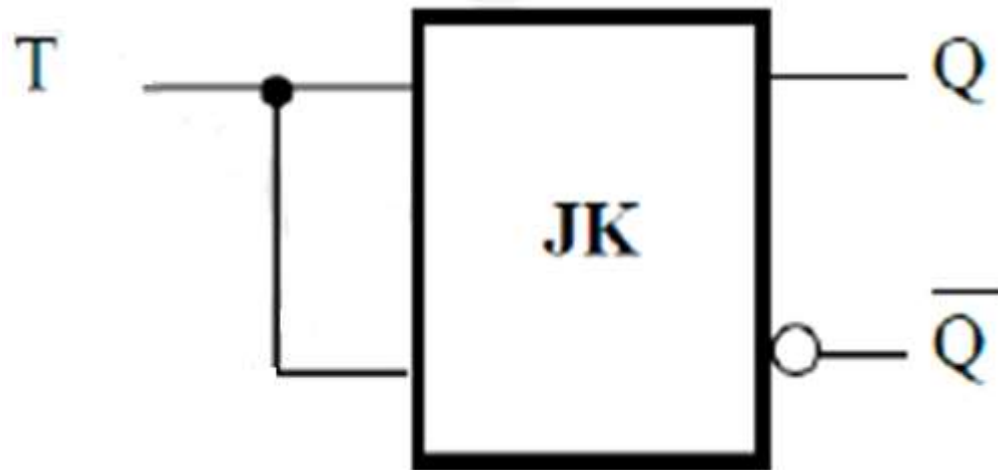
This type of flip-flop is particularly interesting for implementing counters.

The T flip-flop can be realized from a D flip-flop.



## T Flip-flop

For a J-K flip-flop, we observe that if  $J=K=1$ , the state of the output is inverted at **each clock cycle**. Thus, the **T flip-flop** has a single input called 'Trigger', which can be obtained by connecting the **J and K** inputs to the **same** source. It is therefore sometimes called the **complement flip-flop**.



## T Flip-flop

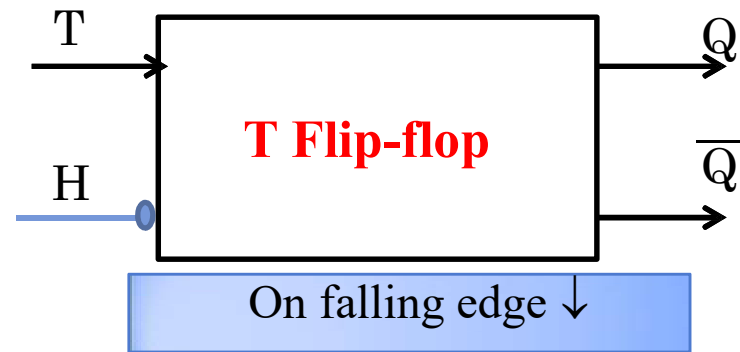
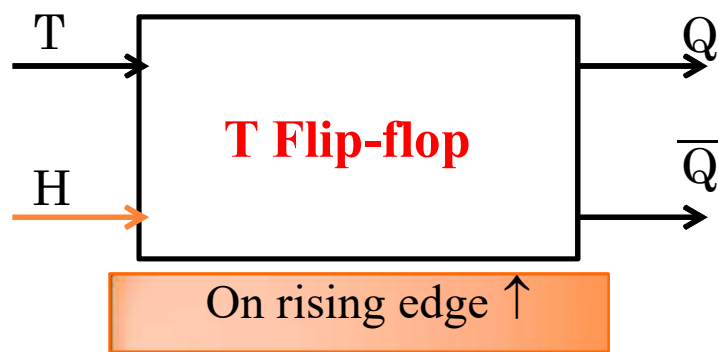
The synchronous **T flip-flop** is activated by the clock signal **H**.

The single input **T** (Trigger) controls the state of the flip-flop.

The output **Q** changes state each time the input **T** transitions to logic state **1** and retains its state the rest of the time.

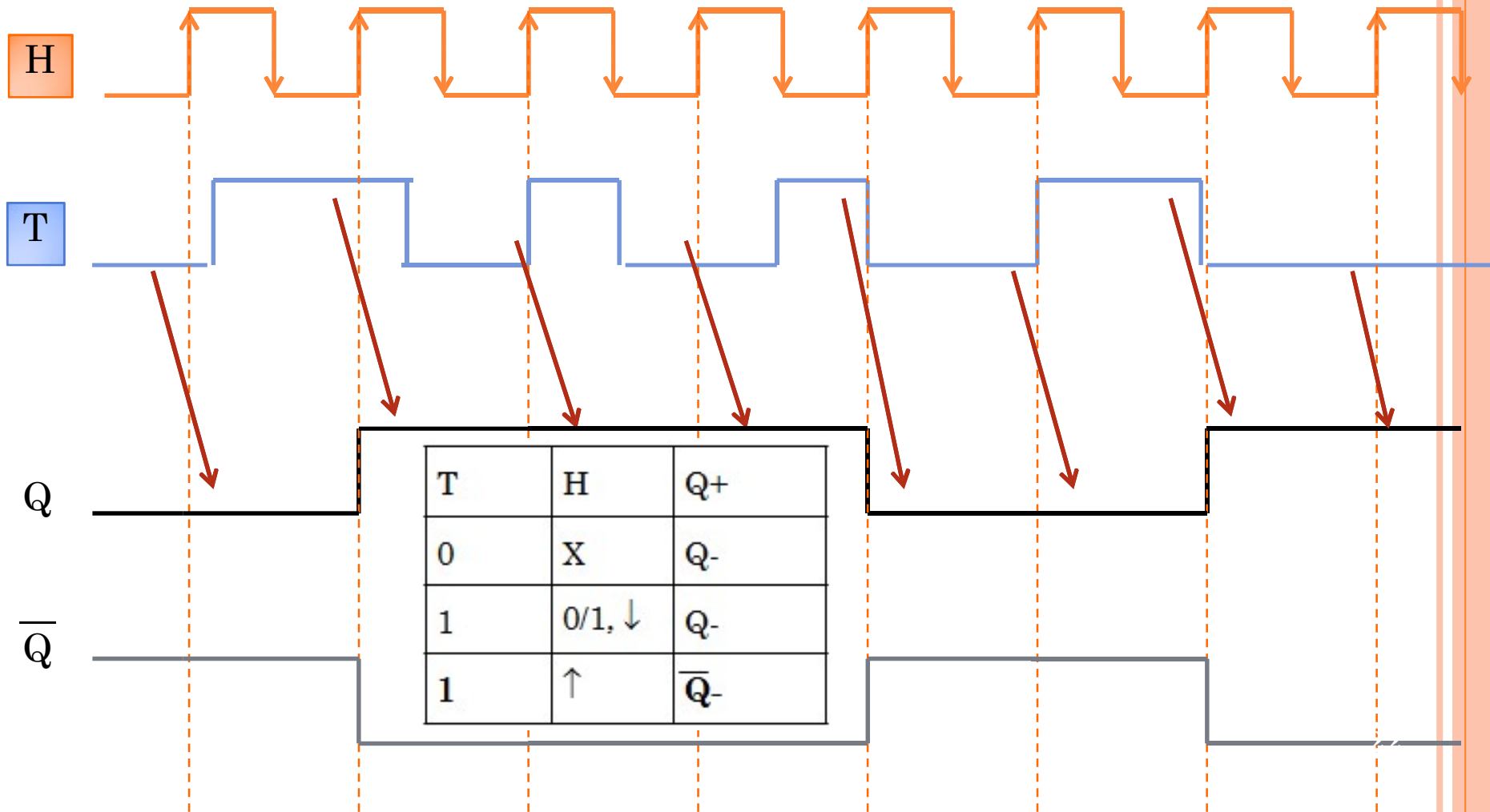
# T Flip-flop

The T (Toggle) flip-flop toggles with each clock pulse (rising or falling edge) when its input T is active.



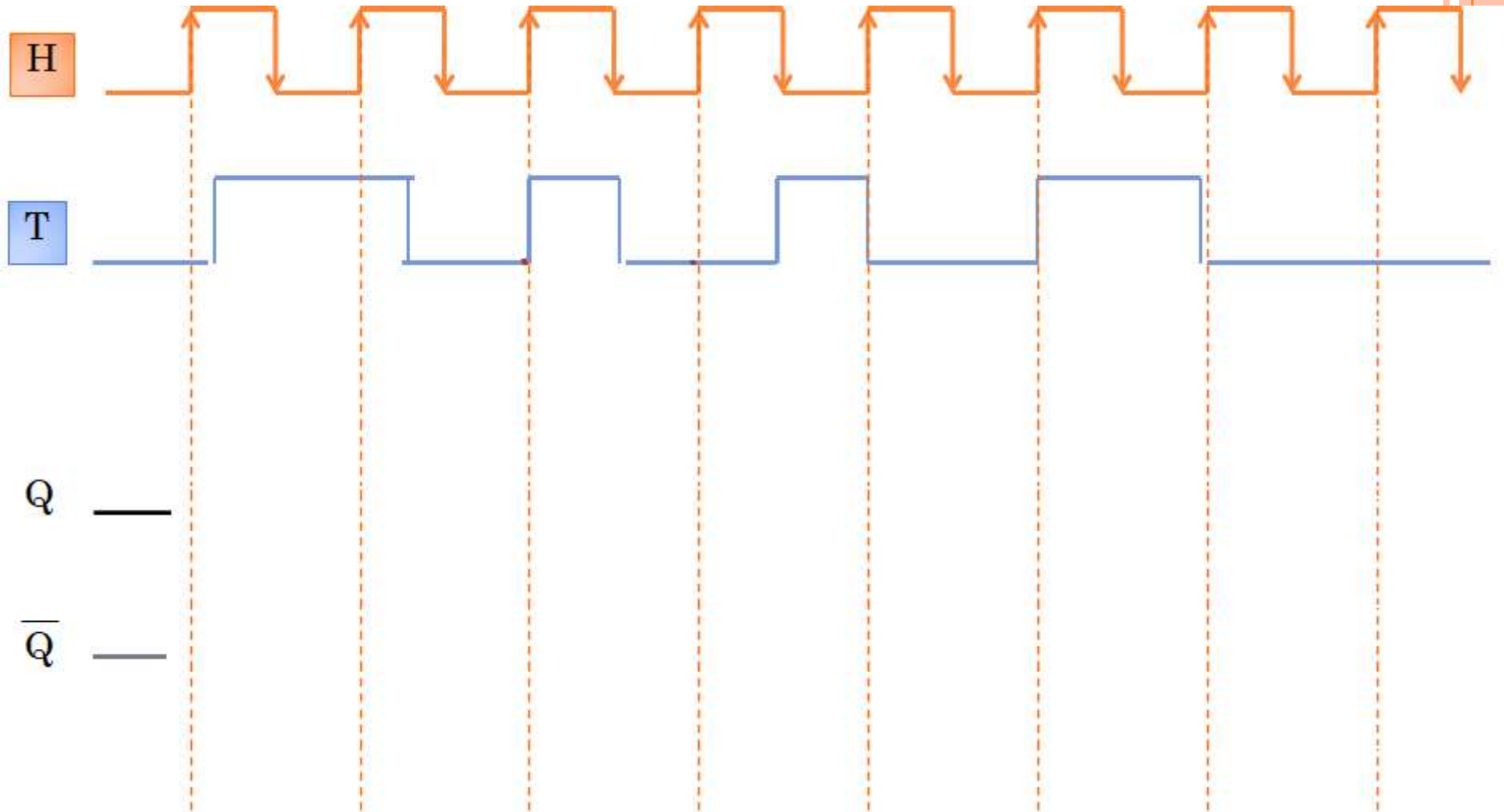
T	H	Q+
0	X	Q-
1	0/1, ↓	Q-
1	↑	$\overline{Q}$ -

# T flip-flop edge-triggered with Rising edge control

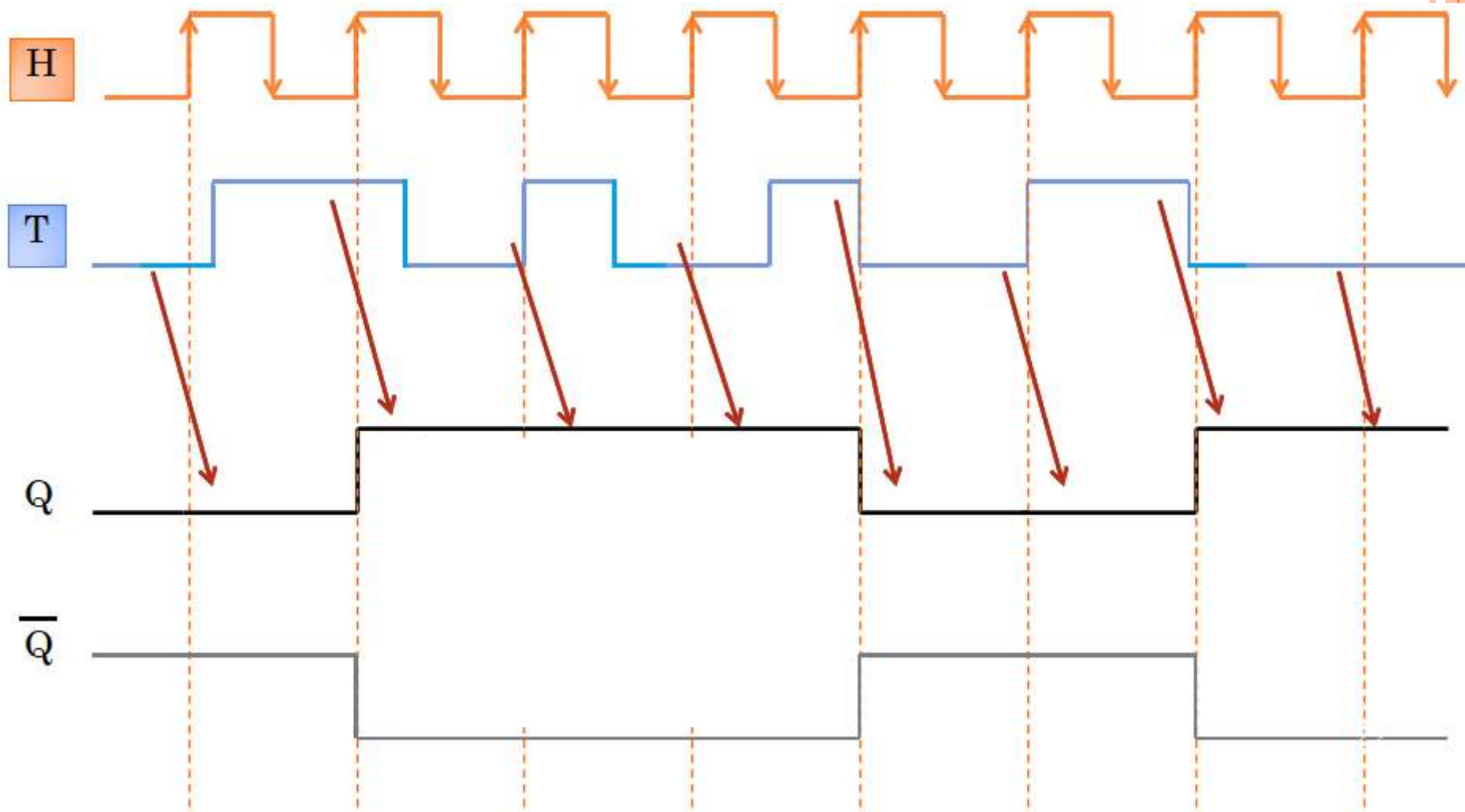




# T flip-flop edge-triggered with Rising edge control



# T flip-flop edge-triggered with Rising edge control

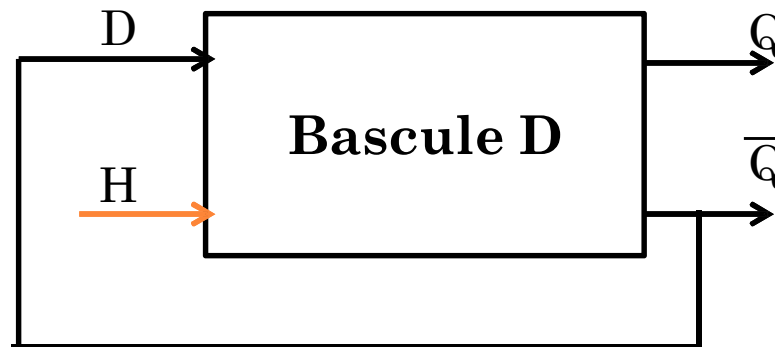


# T flip-flop

**Exercise 05:** Modify a D flip-flop to act as a T flip-flop (rising edge).

H	$Q^+$
0/1/↓	$Q^-$
↑	D

T	H	$Q^+$
0	X	$Q^-$
1	0/1, ↓	$Q^-$
1	↑	$\overline{Q^-}$



# JK flip-flop

**Exercise 06:** Convert a synchronous JK flip-flop into a D flip-flop.

H	D	Q+
0/1	0	Q-
0/1	1	Q-
↑	0	0
↑	1	1

H	J	K	Q+
0/1	X	X	Q-
↓	0	0	Q-
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q-}$

$$H_{JK} = \overline{H}_D, J = D ; K = \overline{D}$$

# JK flip-flop

**Exercise 07:** Transform a JK flip-flop to act as a T flip-flop (falling edge).

T	H	Q+
0	X	Q-
1	0/1, ↑	Q-
1	↓	$\overline{Q-}$

H	J	K	Q+
0/1, ↑	X	X	Q-
↓	0	0	Q-
↓	0	1	0
↓	1	0	1
↓	1	1	$\overline{Q-}$

$$H_T = H_{JK}, J = K = T$$

## FLIP-FLOP EXCITATION TABLE

Characteristic tables of flip-flops specify the next state when the inputs and the current state are known.

However, during the design of a process (logic circuit), transition states (current state and next state) are often known, and it is the inputs that need to be determined to ensure the desired transition. For this reason, we need a table that provides the necessary inputs for a given state transition. This table is called the flip-flop (or combinational circuit) excitation table. It consists of two columns,  $Q_t$  and  $Q_{t+1}$ , and one column for each input. There are four transitions (4 rows) to go from  $Q_t$  to  $Q_{t+1}$ .

## FLIP-FLOP EXCITATION TABLE

For example, for the J-K flip-flop, to achieve the transition from 0 to 1, the J input must be in the state 1, regardless of the state of the K input.

$J = K = 1$  flips the state of the flip-flop, or  $J = 1$  and  $K = 0$  loads 1 into the flip-flop. A cross indicates that the state of the considered input is irrelevant: 0 or 1.

$Q_t$	$Q_{t+1}$	RS Flip-Flop		D Flip-Flop	JK Flip-Flop		T Flip-Flop
		S	R	D	J	K	T
0	0	0	X	0	0	X	0
0	1	1	0	1	1	X	1
1	0	0	1	0	X	1	1
1	1	X	0	1	X	0	0

## FLIP-FLOP INITIALIZATION

The RSH, D, T, and JK flip-flops operate synchronously with respect to a clock signal (H).

For the operation of a system, it is often necessary that these flip-flops be initialized, meaning that their output Q is set to 1 or 0 independently of the clock signal.

This is the role of two additional inputs:

- Preset: sets the output Q to 1.
- Clear: sets the output Q to 0.



## FLIP-FLOP INITIALIZATION

Clear (Cl) and Preset (Pr) are two asynchronous inputs that:

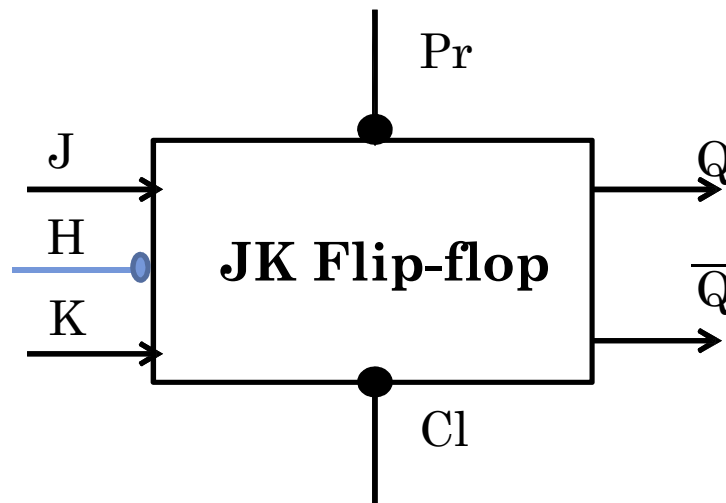
- operate with negative logic
- have higher priority than the clock

Clear	Preset	H	Q
0	0	X	Forbidden state
0	1	X	1
1	0	X	0
1	1	↑	Flip-flop

# FLIP-FLOP INITIALIZATION

## JK Flip-flop

**Exercise 8:** Provide the truth table of the JK flip-flop with the inputs Clear and Preset.



On the falling edge ↓

Thank you  
for your attention