Logic and Computer Design Fundamentals

Chapter 5 – Sequential Circuits

Part 1 - Storage Elements

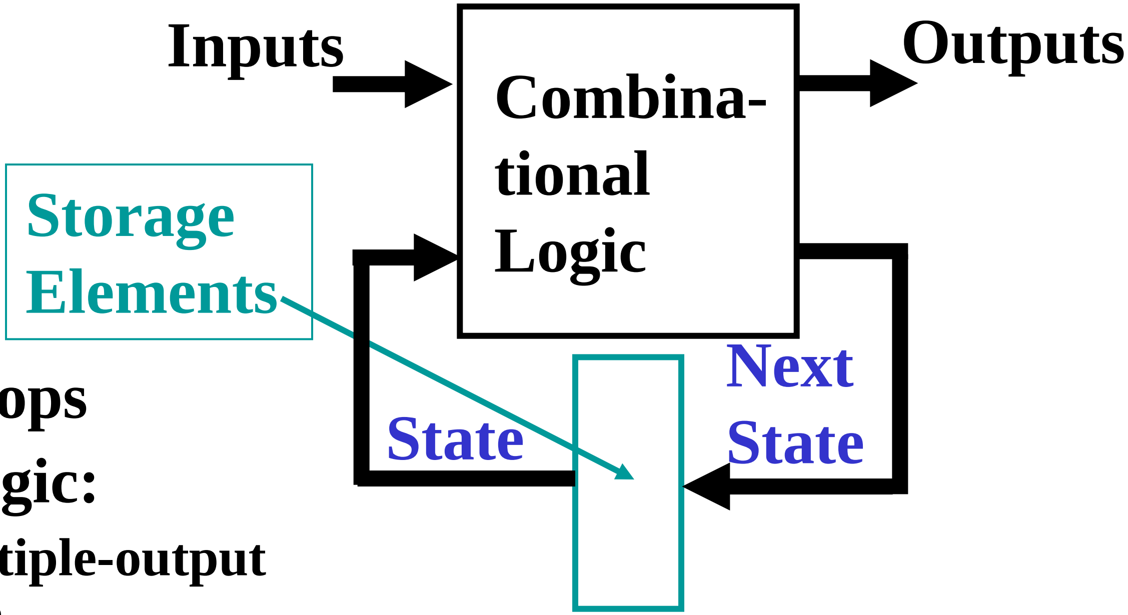
Charles Kime & Thomas Kaminski

(Hyperlinks are active in View Show mode)

Overview

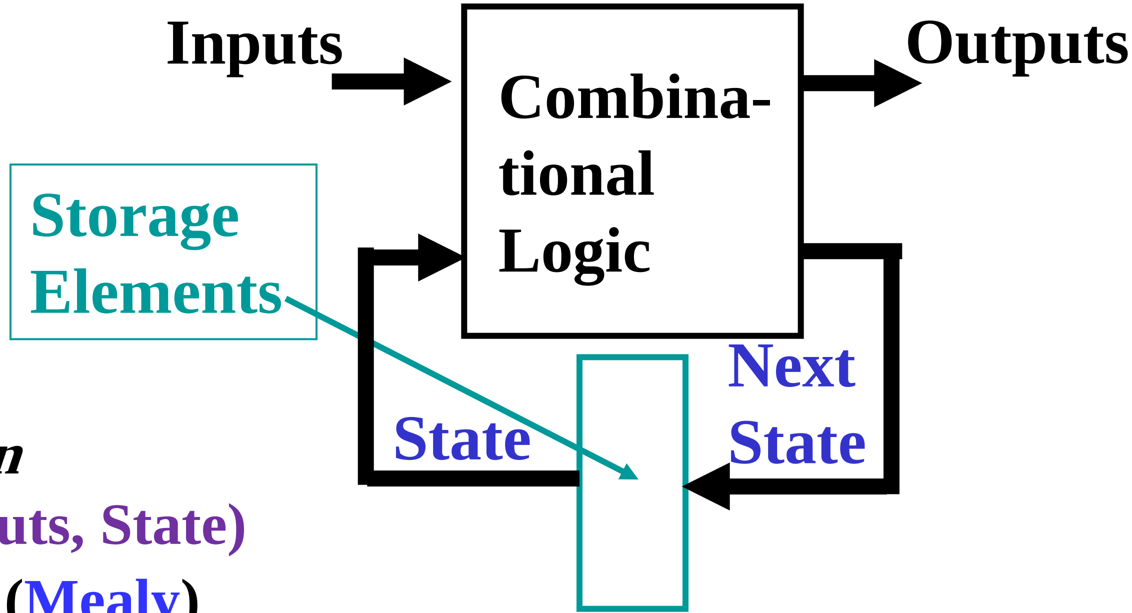
- **Part 1 - Storage Elements**
  - **Introduction to sequential circuits**
  - **Types of sequential circuits**
  - **Storage elements**
    - **Latches**
    - **Flip-flops**
- **Part 2 - Sequential Circuit Analysis**
- **Part 3 - Sequential Circuit Design**
- **Part 4 – State Machine Design**

Introduction to Sequential Circuits



- A Sequential circuit contains:
  - Storage elements: Latches or Flip-Flops
  - Combinational Logic:
    - Implements a multiple-output switching function
    - <u>Inputs</u> are signals from the outside.
    - <u>Outputs</u> are signals to the outside.
    - Other inputs, <u>State</u> or <u>Present State</u>, are signals from storage elements.
    - The remaining outputs, <u>Next State</u> are inputs to storage elements.

Introduction to Sequential Circuits



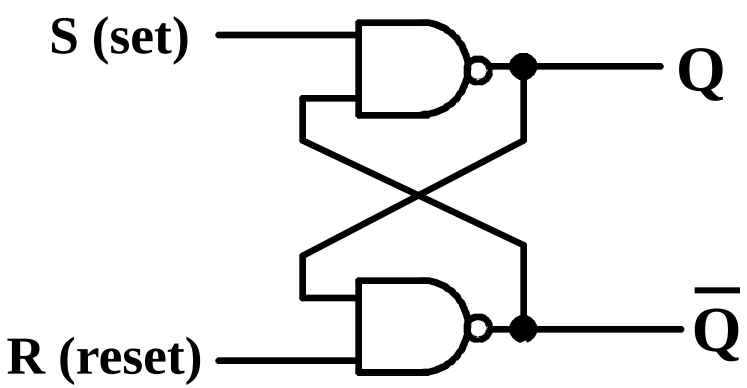
- Combinatorial Logic
  - *Next state function*
    Next State = f(Inputs, State)
  - *Output function* (Mealy)
    Outputs = g(Inputs, State)
  - *Output function* (Moore)
    Outputs = h(State)
- Output function type depends on specification and affects the design significantly

Types of Sequential Circuits

- **Depends on the <u>time</u>s at which:**
  - **storage elements observe their inputs, and**
  - **storage elements change their state**
- **<u>Synchronous</u>**
  - **Behavior defined from knowledge of its signals at <u>discrete</u> instances of time**
  - **Storage elements observe inputs and can change state only in relation to a timing signal (<u>clock pulses</u> from a <u>clock</u>)**
- **<u>Asynchronous</u>**
  - **Behavior defined from knowledge of inputs an any instant of time and the order in continuous time in which inputs change**
  - **Nevertheless, the synchronous abstraction makes complex designs tractable!**

Basic (NAND) $\overline{S}$ – $\overline{R}$ Latch

- "Cross-Coupling" two NAND gates gives the $\overline{S}$ -$\overline{R}$ Latch:
- Which has the time sequence behavior:



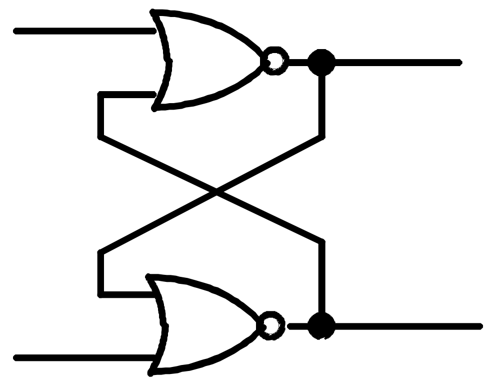
Time ↓

R	S	Q	$\overline{Q}$	Comment
1	1	?	?	Stored state unknown
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
1	1	?	?	Unstable!

- S = 0, R = 0 is <u>forbidden</u> as input pattern

Basic (NOR) S – R Latch

- **Cross-coupling two NOR gates gives the S – R Latch:**
- **Which has the time sequence behavior:**

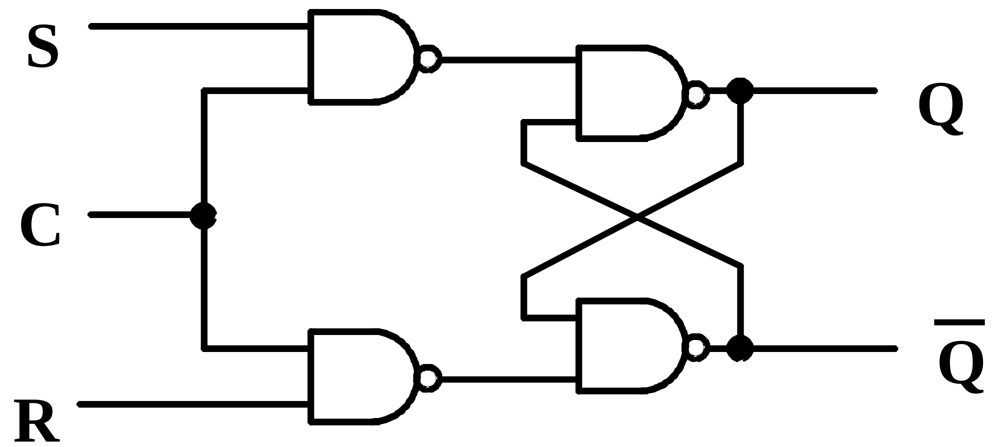


Time ↓

R	S	Q	$\overline{Q}$	Comment
0	0	?	?	Stored state unknown
0	1	1	0	"Set" Q to 1
0	0	1	0	Now Q "remembers" 1
1	0	0	1	"Reset" Q to 0
0	0	0	1	Now Q "remembers" 0
1	1	0	0	Both go low
0	0	?	?	Unstable!

Clocked S - R Latch

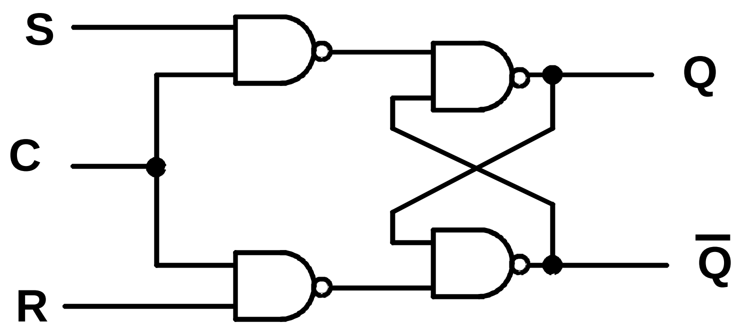
- **Adding two NAND gates to the basic $\overline{S}$ - $\overline{R}$ NAND latch gives the clocked S – R latch:**



- **Has a time sequence behavior similar to the basic S-R latch <u>except that</u> the S and R inputs are only observed when the line C is high.**
- **C means "control" or "clock".**

Clocked S - R Latch (continued)

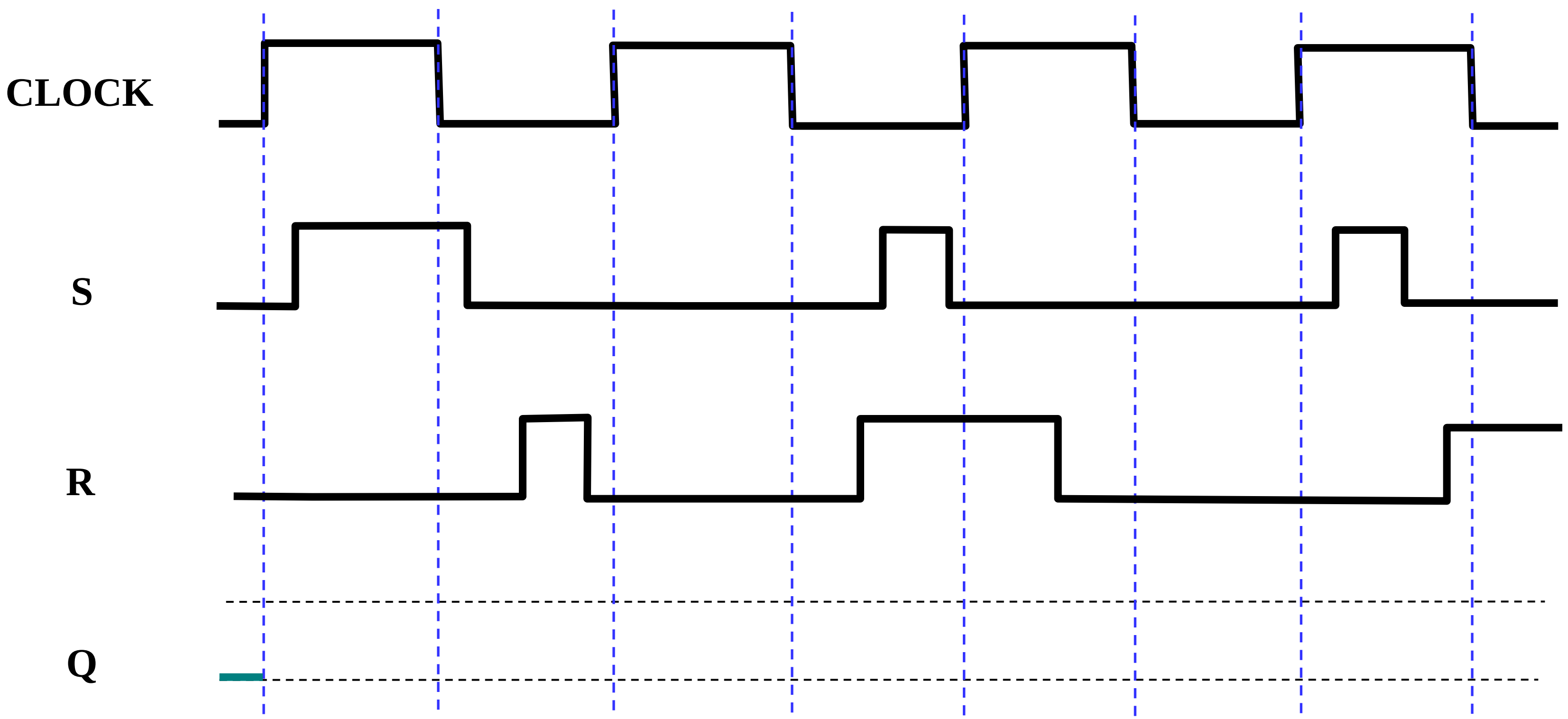
- The Clocked S-R Latch can be described by a table:



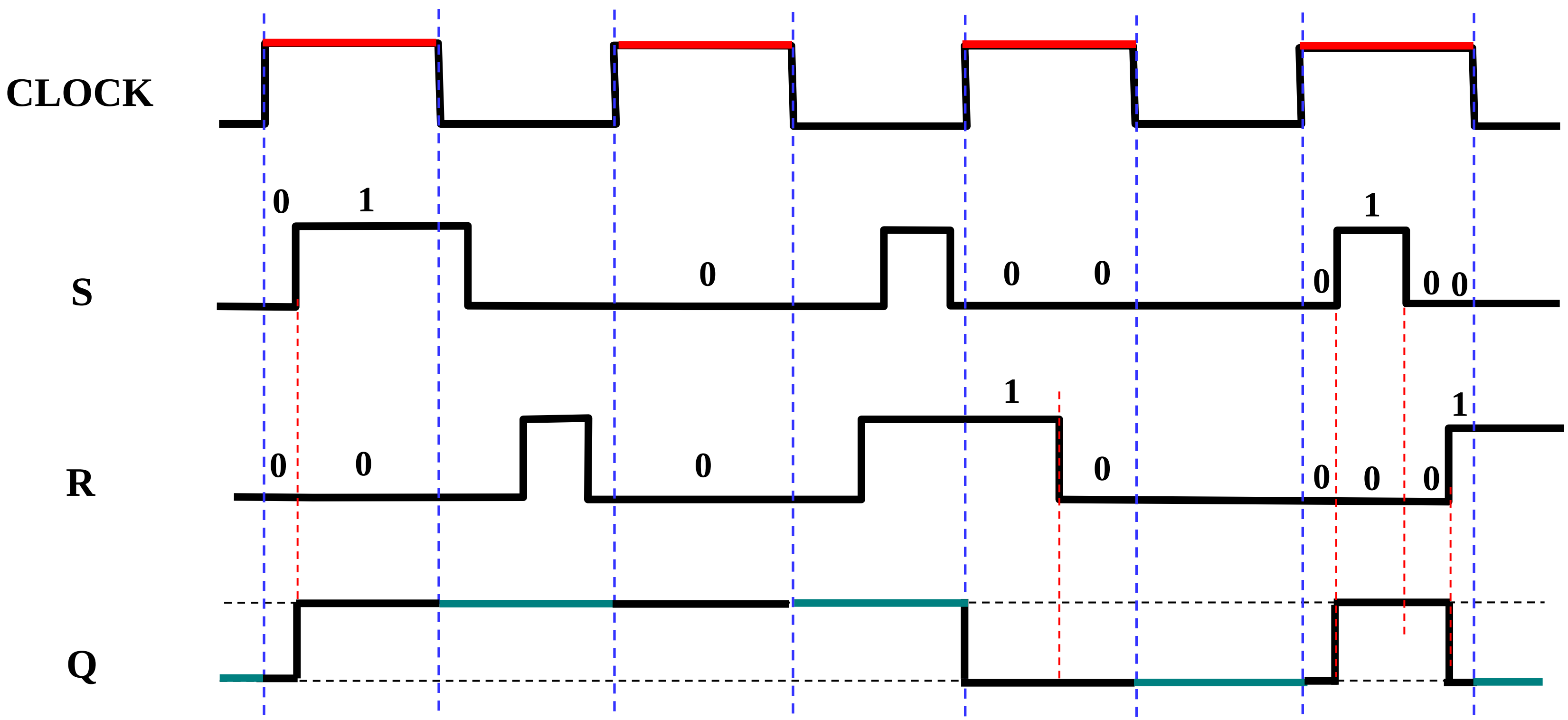
Q(t)	S	R	Q(t+1)	Comment
0	0	0	0	No change
0	0	1	0	Clear Q
0	1	0	1	Set Q
0	1	1	???	Indeterminate
1	0	0	1	No change
1	0	1	0	Clear Q
1	1	0	1	Set Q
1	1	1	???	Indeterminate

- The table describes what happens after the clock [at time (t+1)] based on:
  - current inputs (S,R) and
  - current state Q(t).

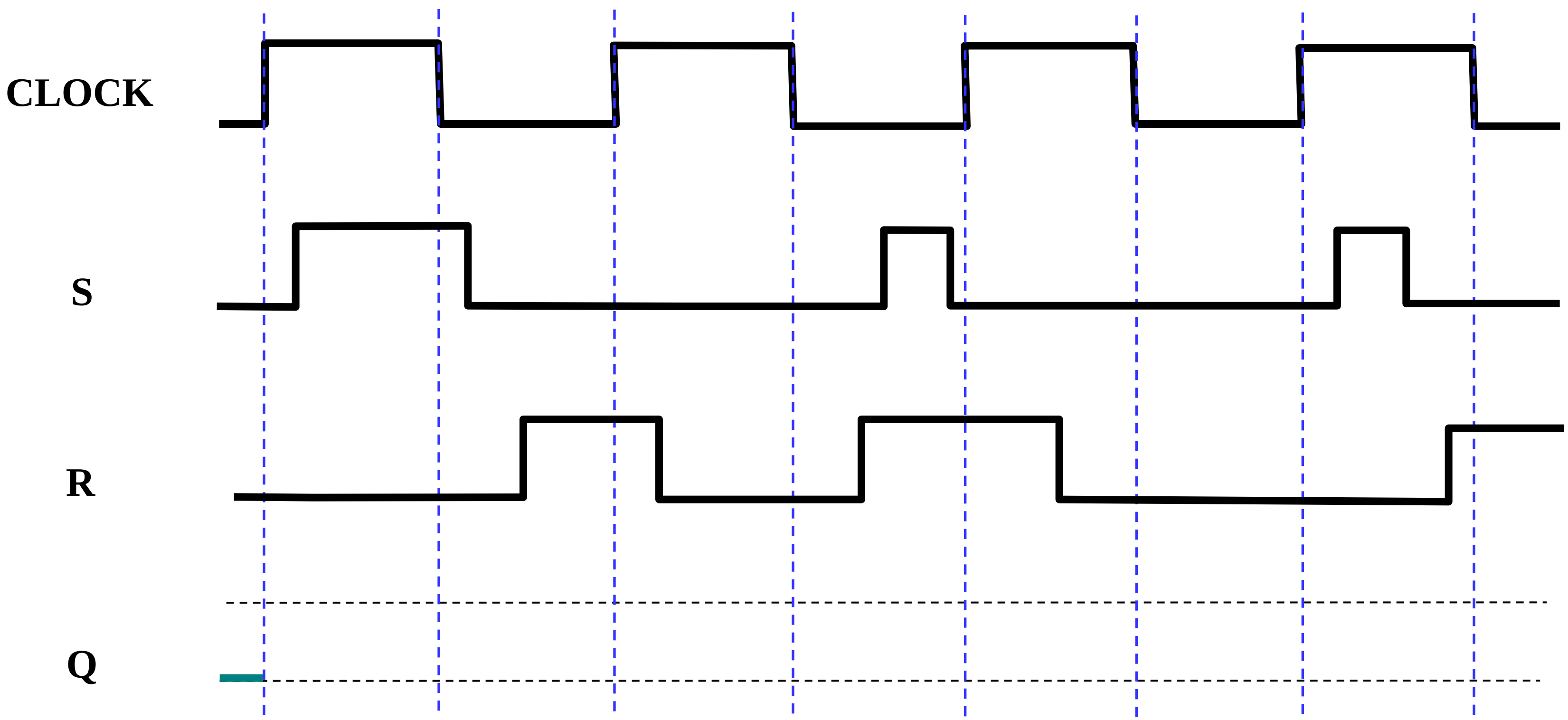
Clocked S - R Latch – Example 1



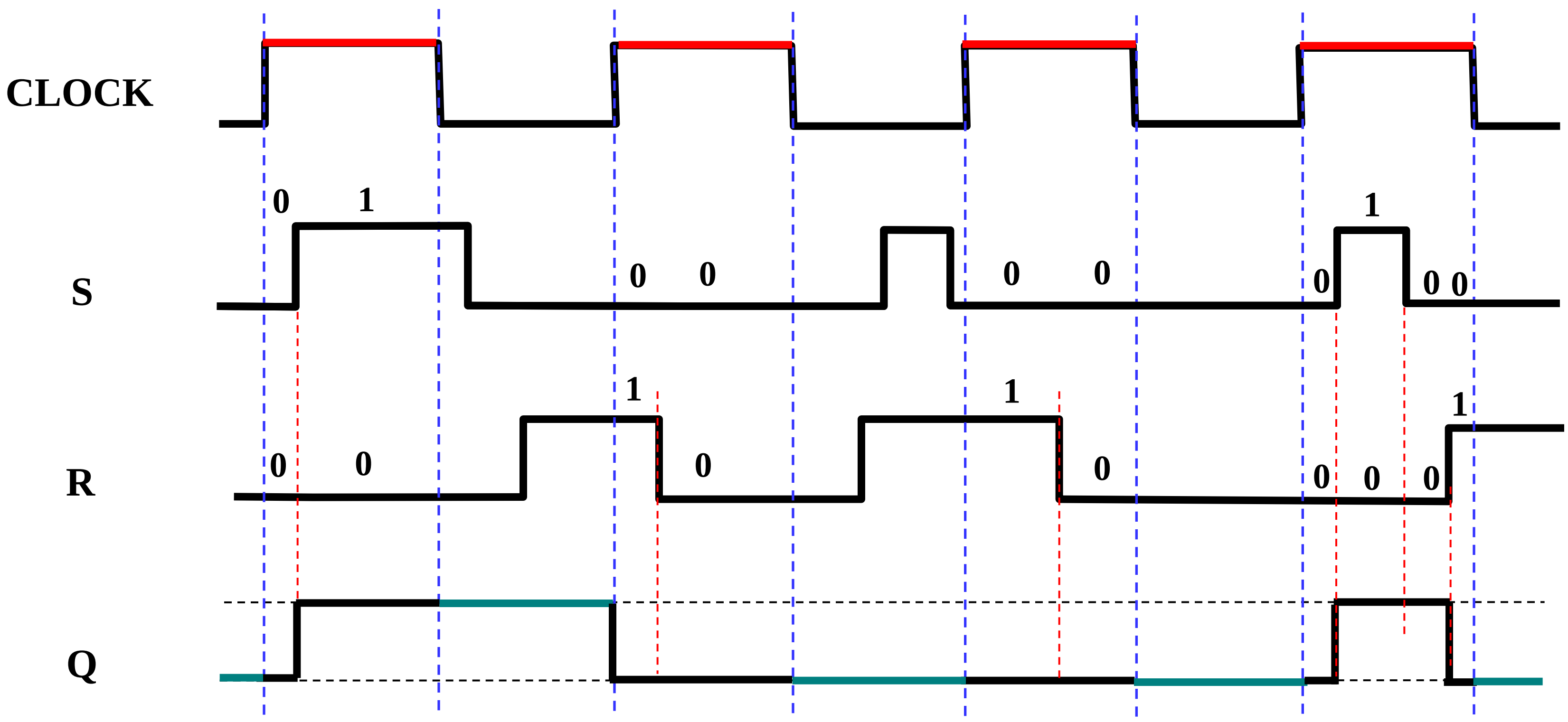
Clocked S - R Latch – Example 1



Clocked S - R Latch – Example 2

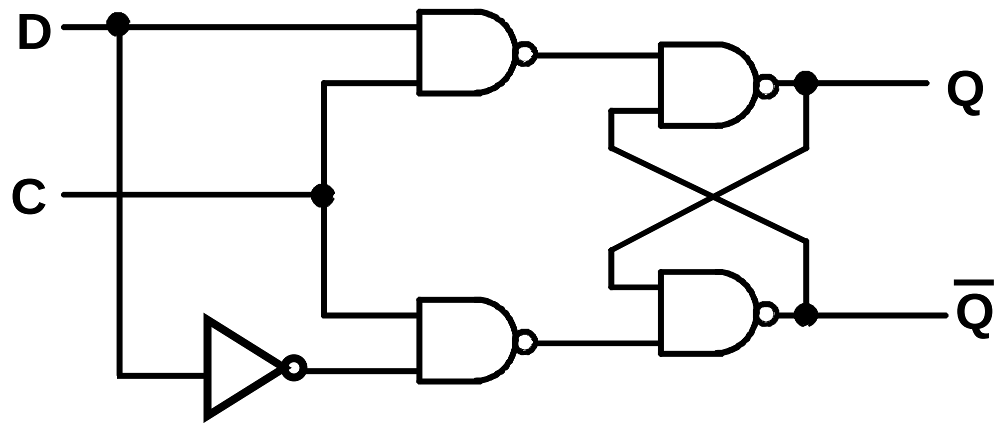


Clocked S - R Latch – Example 2



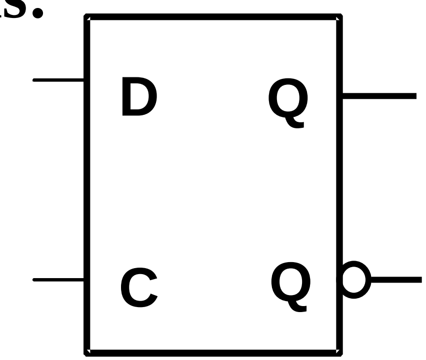
D Latch

- **Adding an inverter to the S-R Latch, gives the D Latch:**
- **Note that there are no "indeterminate" states!**

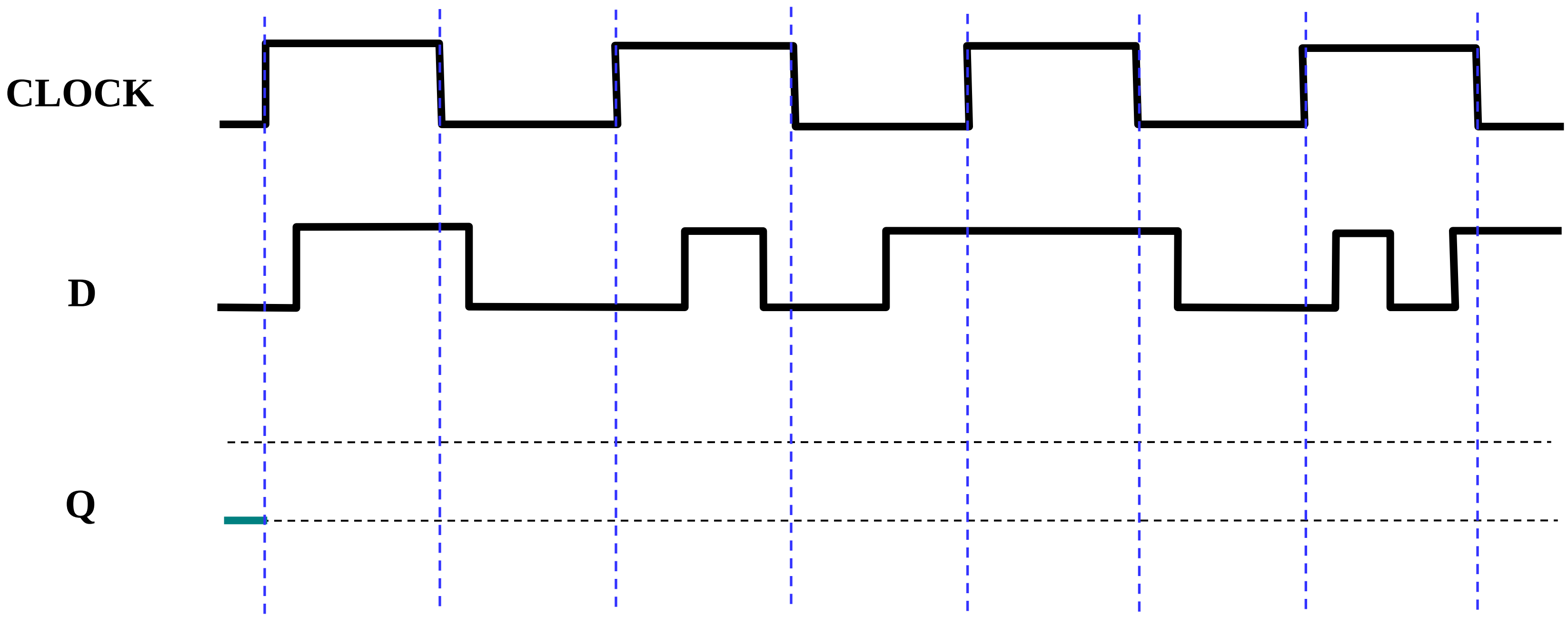


Q	D	Q(t+1)	Comment
0	0	0	No change
0	1	1	Set Q
1	0	0	Clear Q
1	1	1	No Change

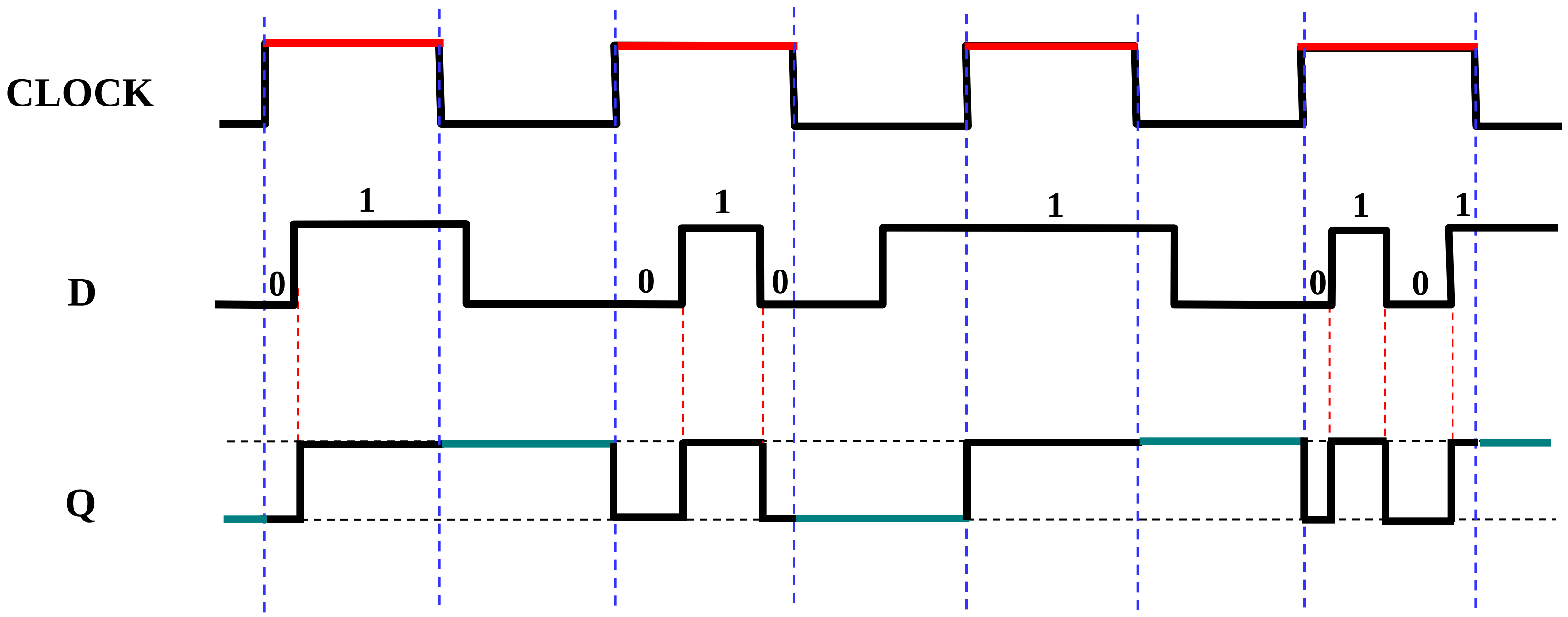
The graphic symbol for a D Latch is:



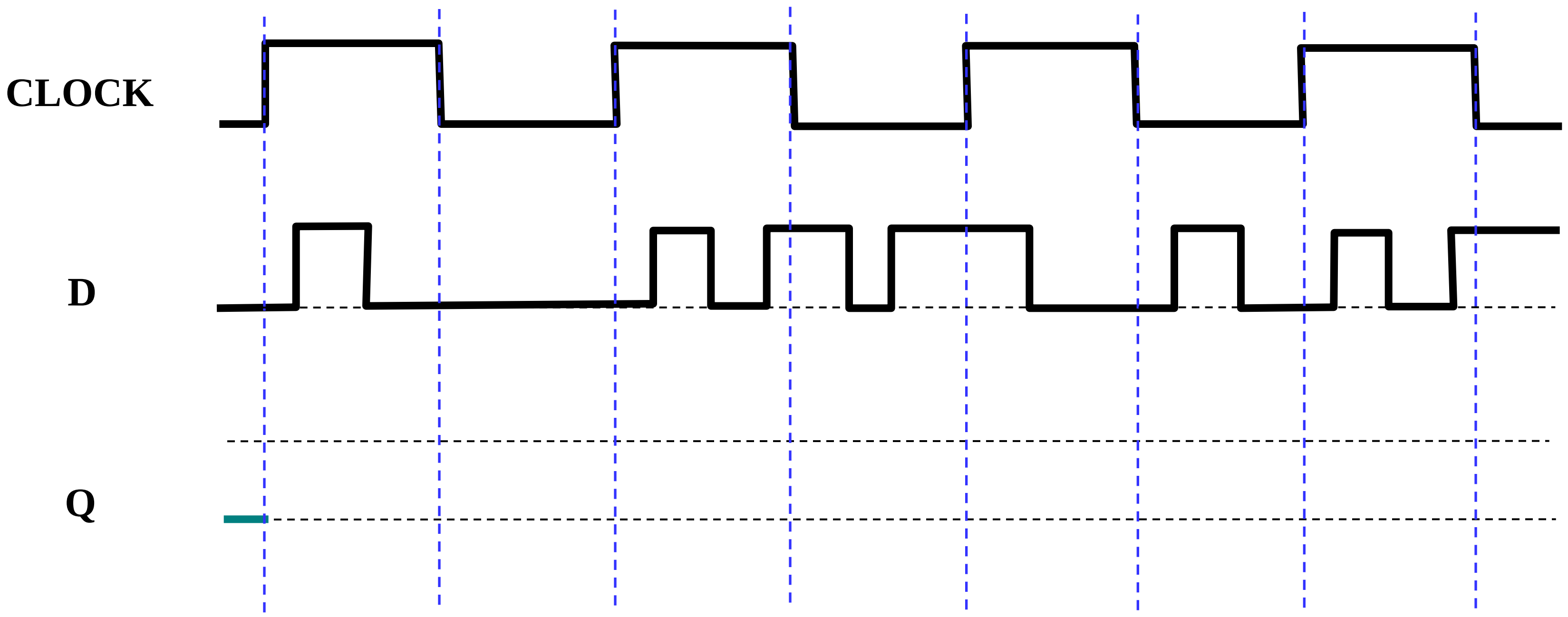
D Latch – Example 1



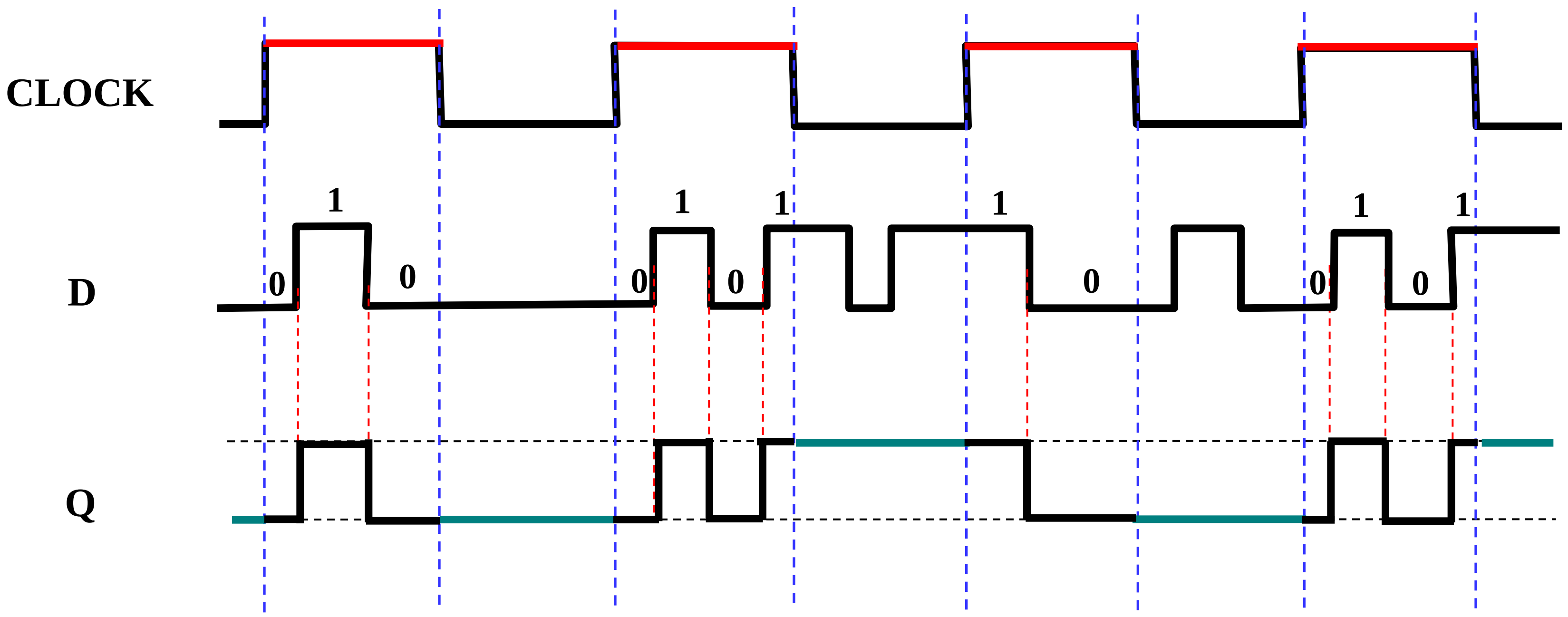
D Latch – Example 1



D Latch – Example 2



D Latch – Example 2



Flip-Flops

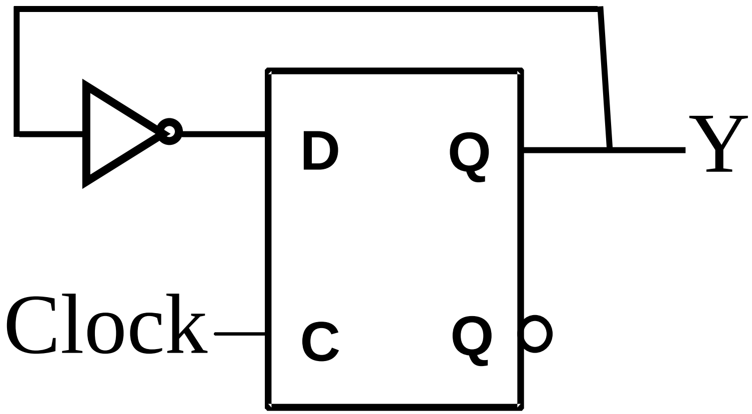
- The latch timing problem
- Master-slave flip-flop
- Edge-triggered flip-flop
- Standard symbols for storage elements
- Direct inputs to flip-flops

The Latch Timing Problem

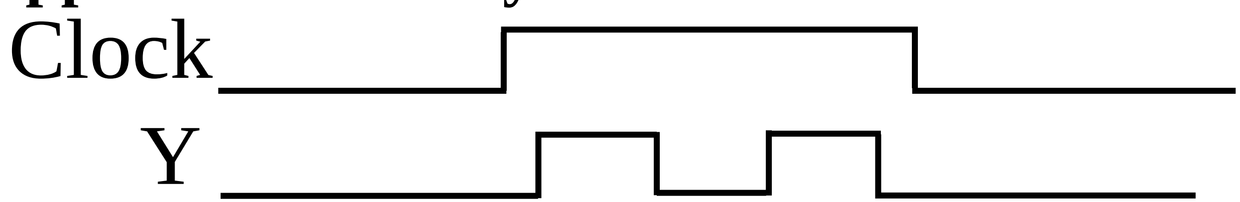
- In a sequential circuit, paths may exist through combinational logic:
  - From one storage element to another
  - From a storage element back to the same storage element
- The combinational logic between a latch output and a latch input may be as simple as an interconnect
- For a clocked D-latch, the output Q depends on the input D whenever the clock input C has value 1

The Latch Timing Problem (continued)

- **Consider the following circuit:**



- **Suppose that initially Y = 0.**



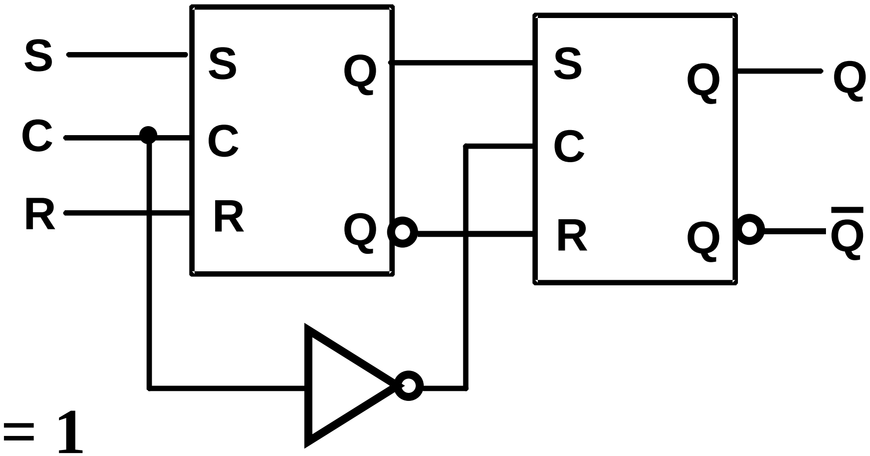
- **As long as C = 1, the value of Y continues to change!**
- **The changes are based on the delay present on the loop through the connection from Y back to Y.**
- **This behavior is clearly unacceptable.**
- **<u>Desired behavior</u>: Y changes <u>only once</u> per clock pulse**

Logic and Computer Design Fundamentals, 4e
PowerPoint® Slides

The Latch Timing Problem (continued)

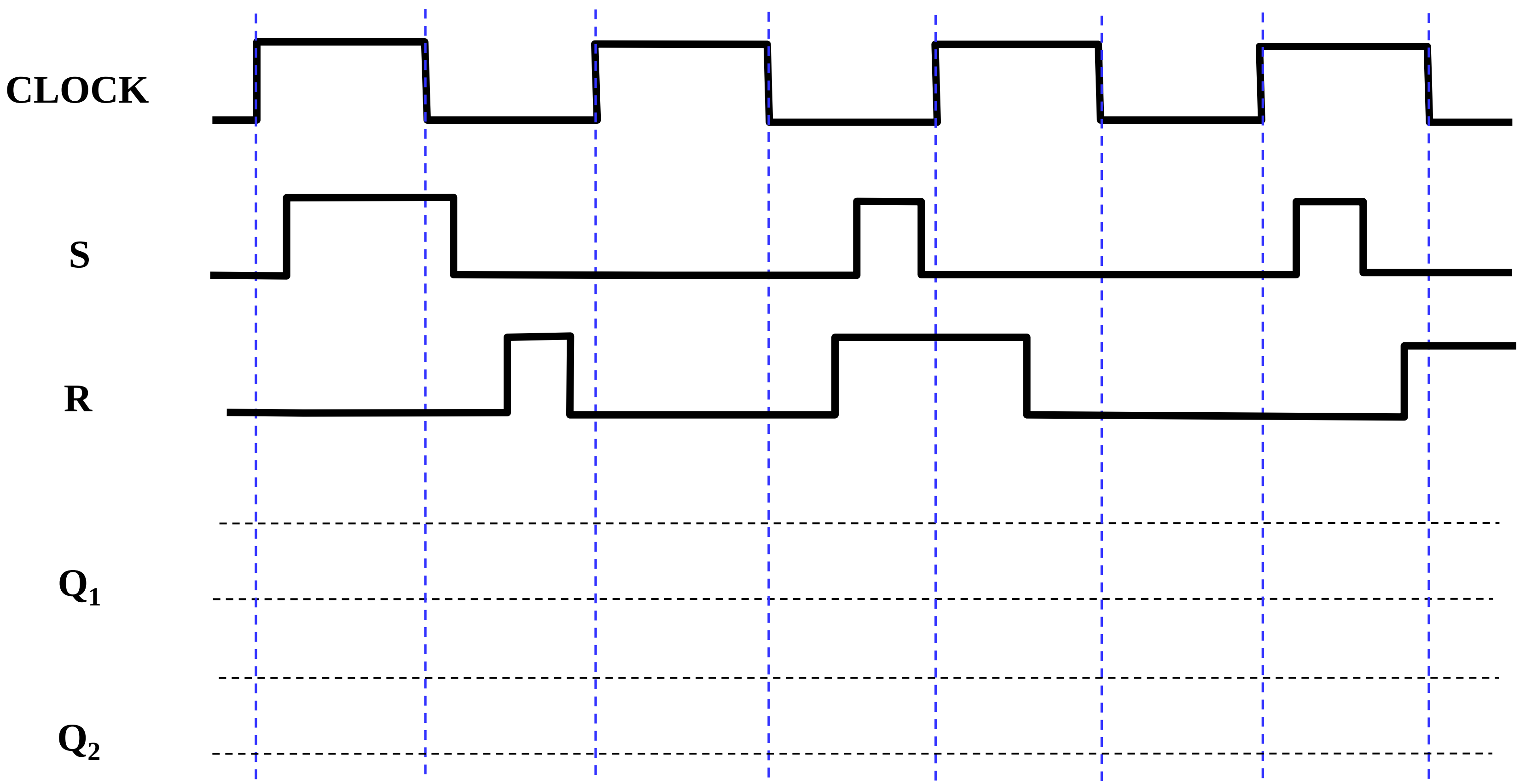
- **A solution to the latch timing problem is to <u>break</u> the closed path from Y to Y within the storage element**
- **The commonly-used, path-breaking solutions replace the clocked D-latch with:**
  - **a master-slave flip-flop**
  - **an edge-triggered flip-flop**

S-R Master-Slave Flip-Flop

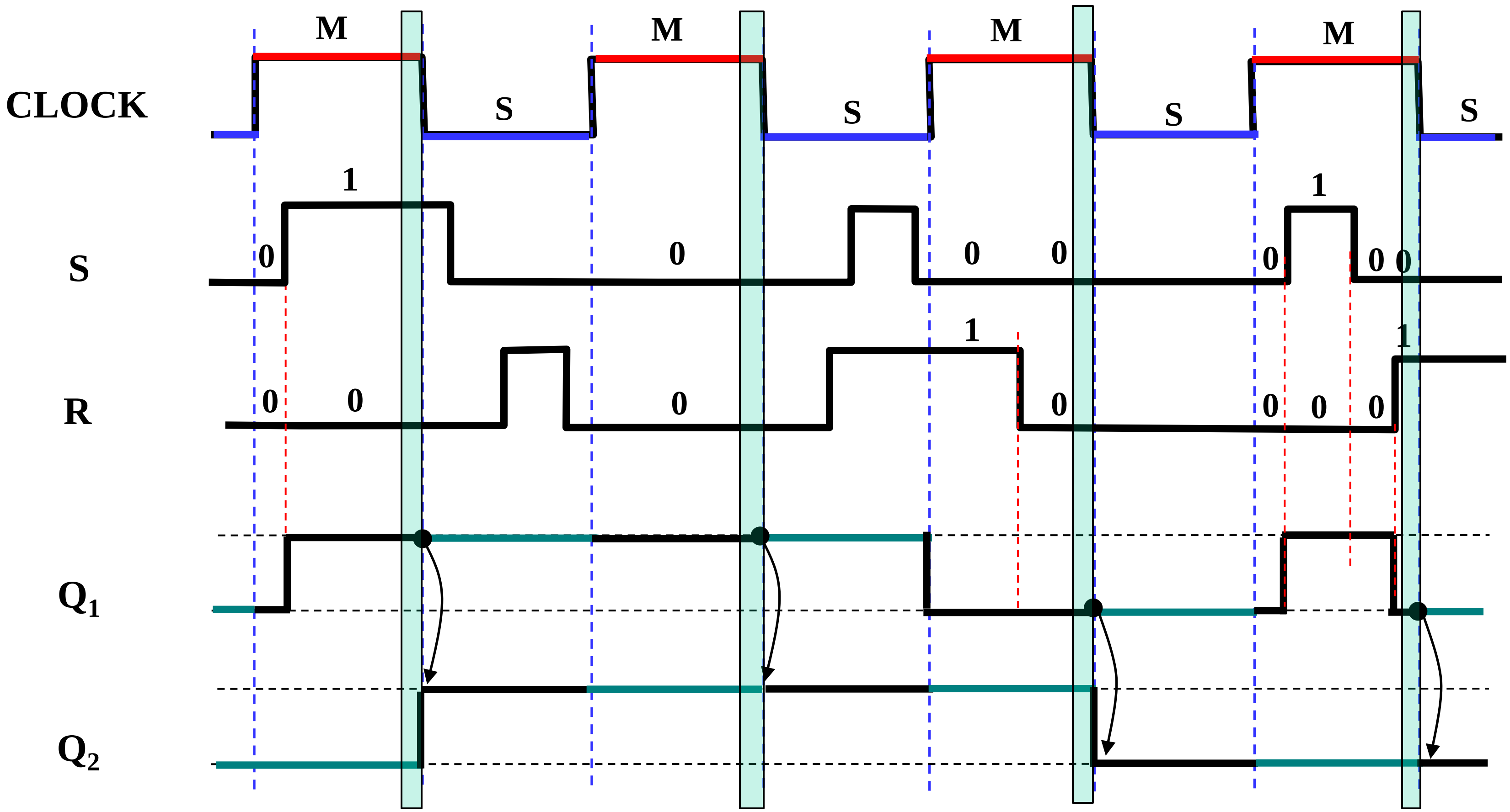


- **Consists of two clocked S-R latches in series with the clock on the second latch inverted**
- **The input is observed by the first latch with C = 1**
- **The output is changed by the second latch with C = 0**
- **The path from input to output is broken by the difference in clocking values (C = 1 and C = 0).**
- **The behavior demonstrated by the example with D driven by Y given previously is prevented since the clock must change from 1 to 0 before a change in Y based on D can occur.**

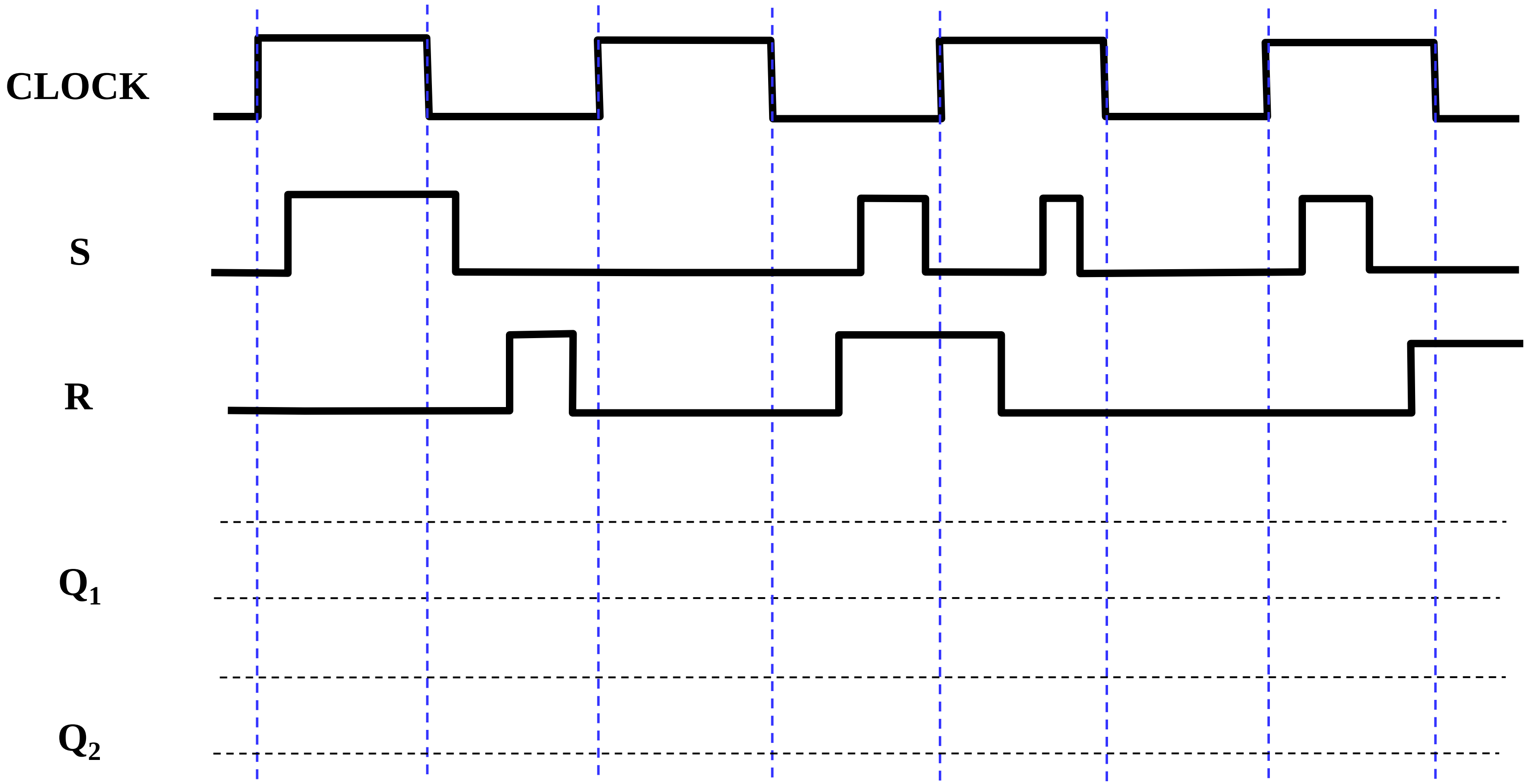
S-R Master-Slave Flip-Flop – Example 1



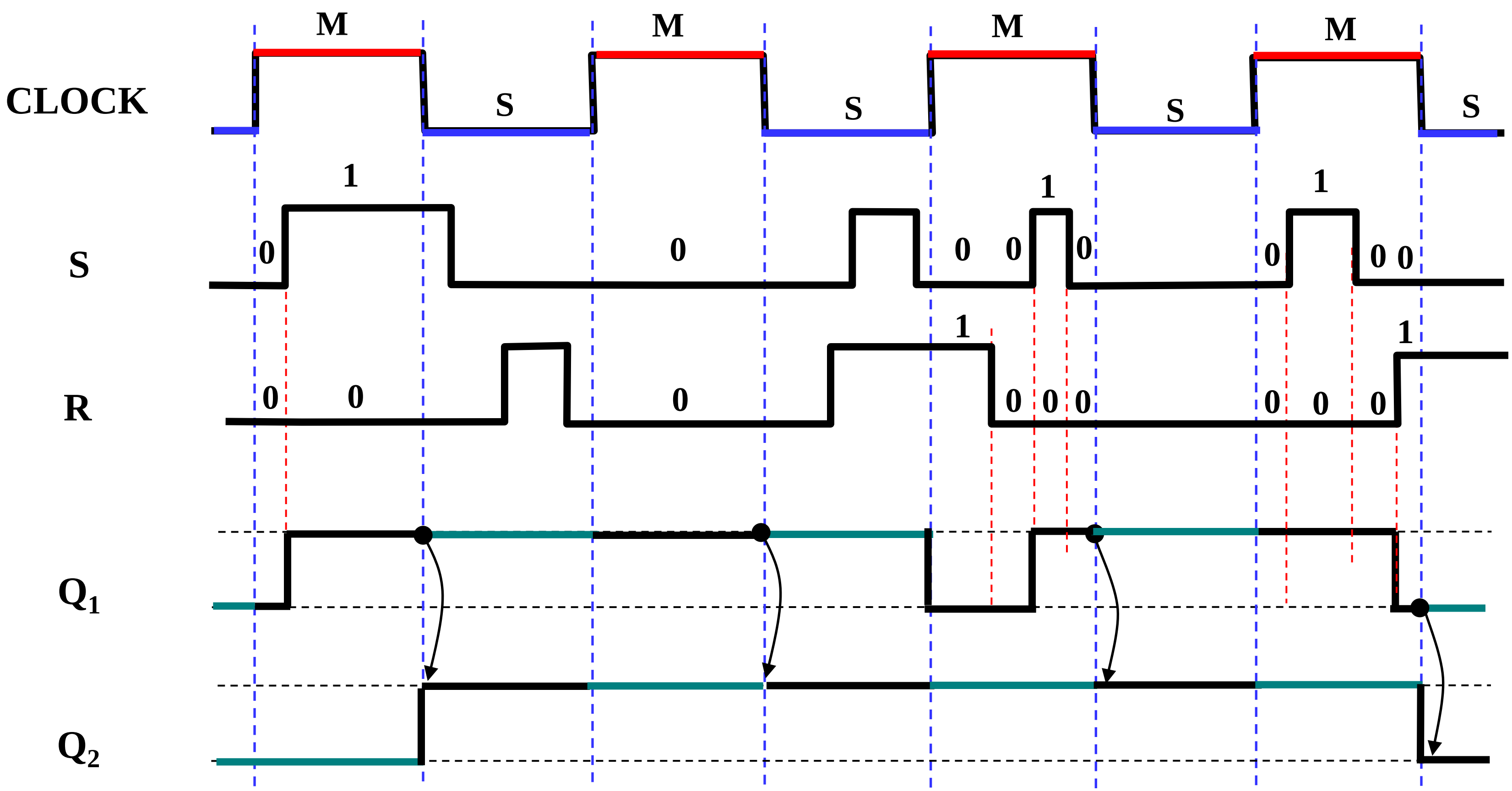
S-R Master-Slave Flip-Flop – Example 1



S-R Master-Slave Flip-Flop – Example 2



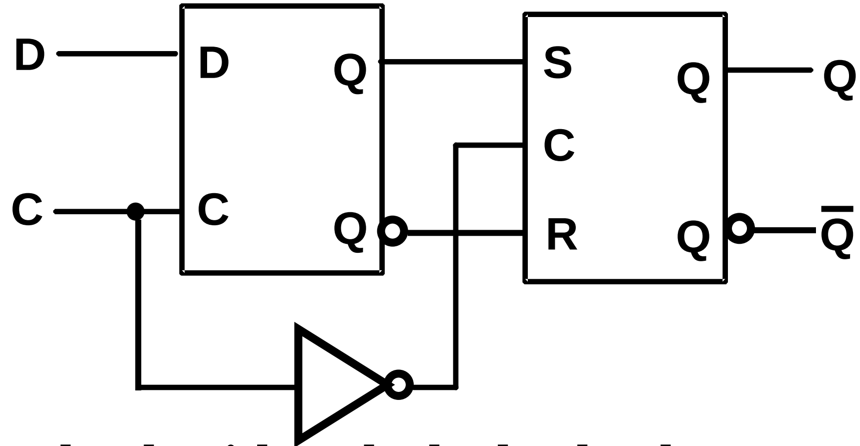
S-R Master-Slave Flip-Flop – Example 2



Flip-Flop Solution

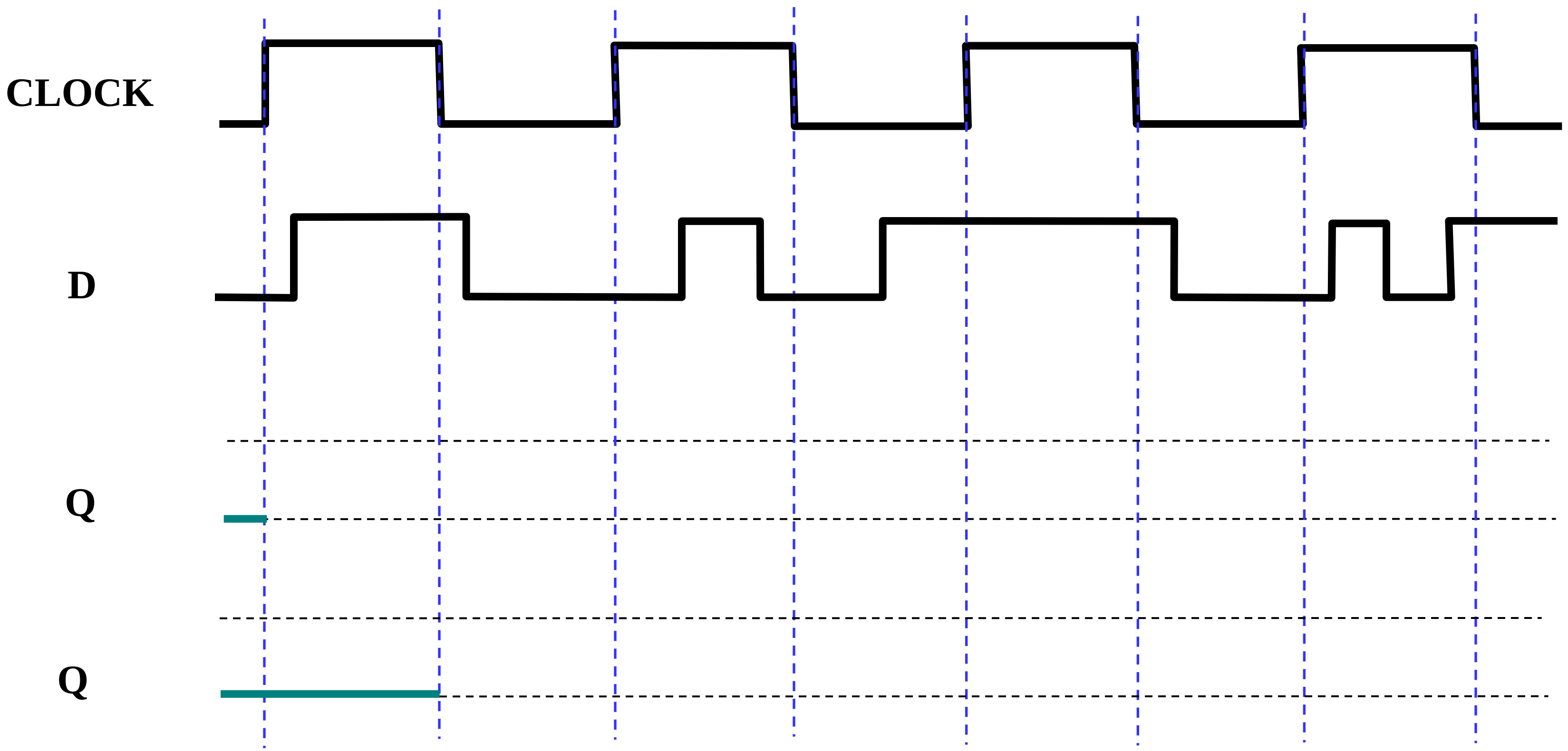
- Use edge-triggering instead of master-slave
- An *edge-triggered* flip-flop ignores the pulse while it is at a constant level and triggers only during a transition of the clock signal
- Edge-triggered flip-flops can be built directly at the electronic circuit level, or
- A master-slave D flip-flop which also exhibits edge-triggered behavior can be used.

Edge-Triggered D Flip-Flop

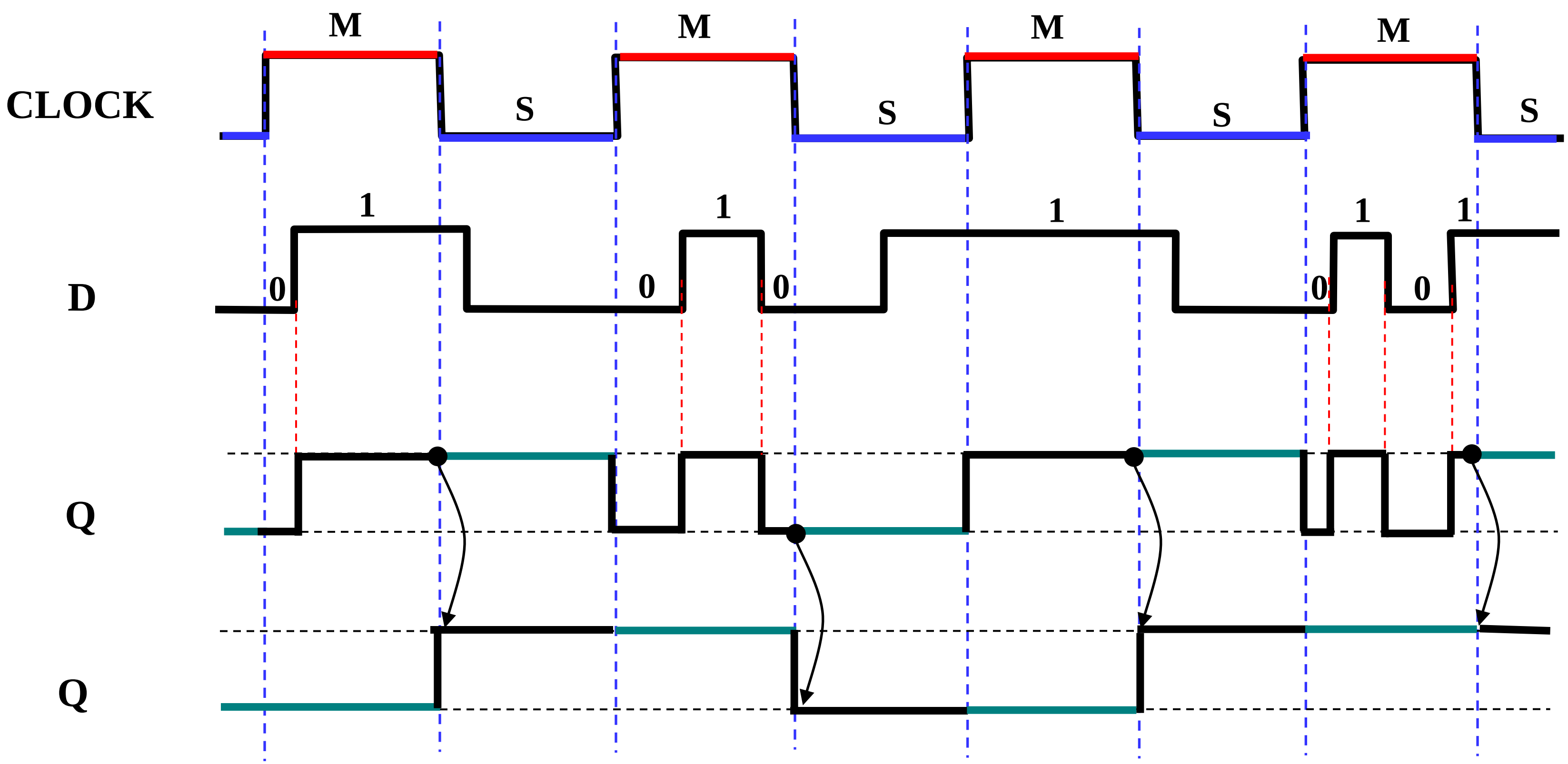


- The edge-triggered D flip-flop is the same as the master-slave D flip-flop
- It can be formed by:
  - Replacing the first clocked S-R latch with a clocked D latch or
  - Adding a D input and inverter to a master-slave S-R flip-flop
- The delay of the S-R master-slave flip-flop can be avoided since the 1s-catching behavior is not present with D replacing S and R inputs
- The change of the D flip-flop output is associated with the negative edge (falling edge) at the end of the pulse
- It is called a *negative-edge triggered* flip-flop

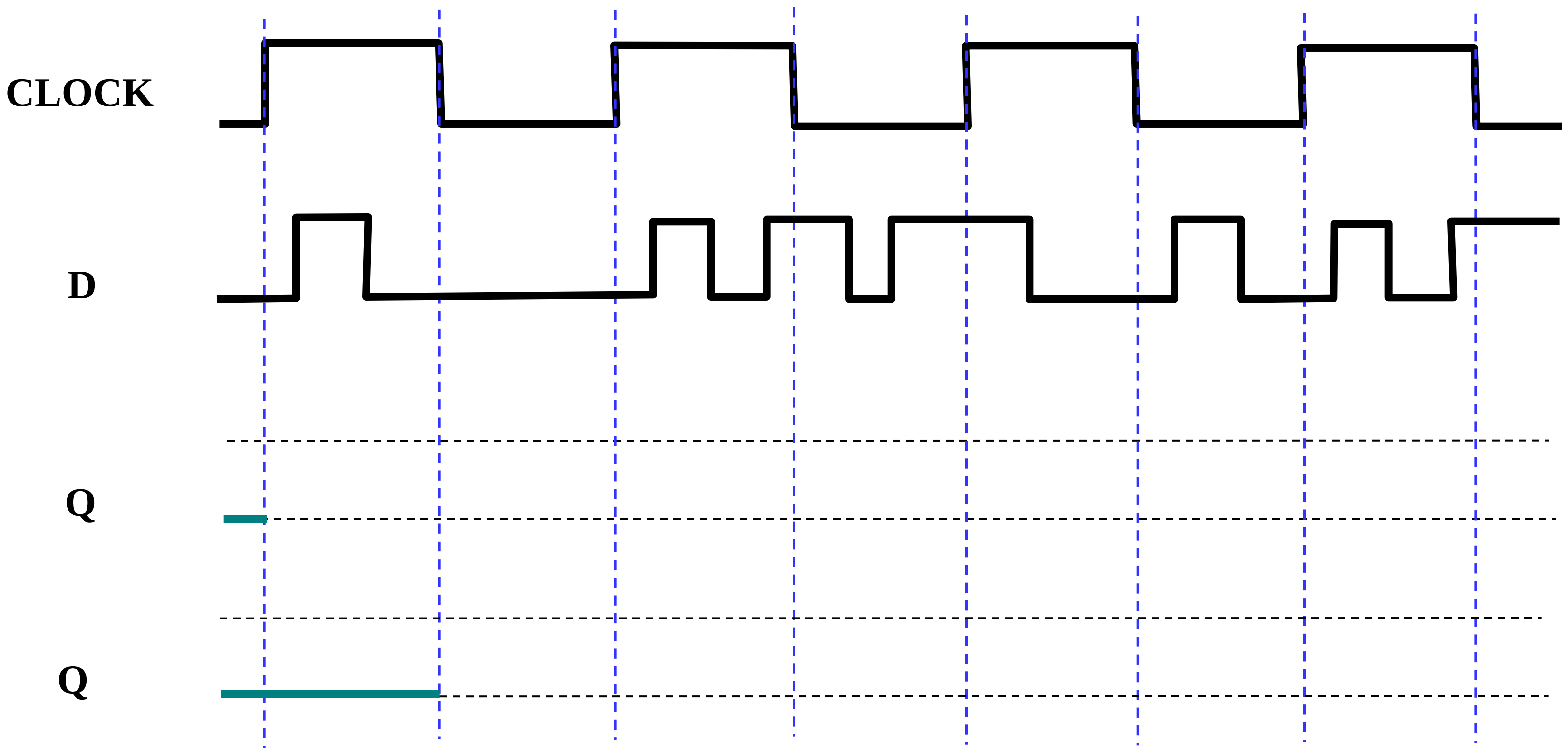
Edge-Triggered D Flip-Flop – Example 1



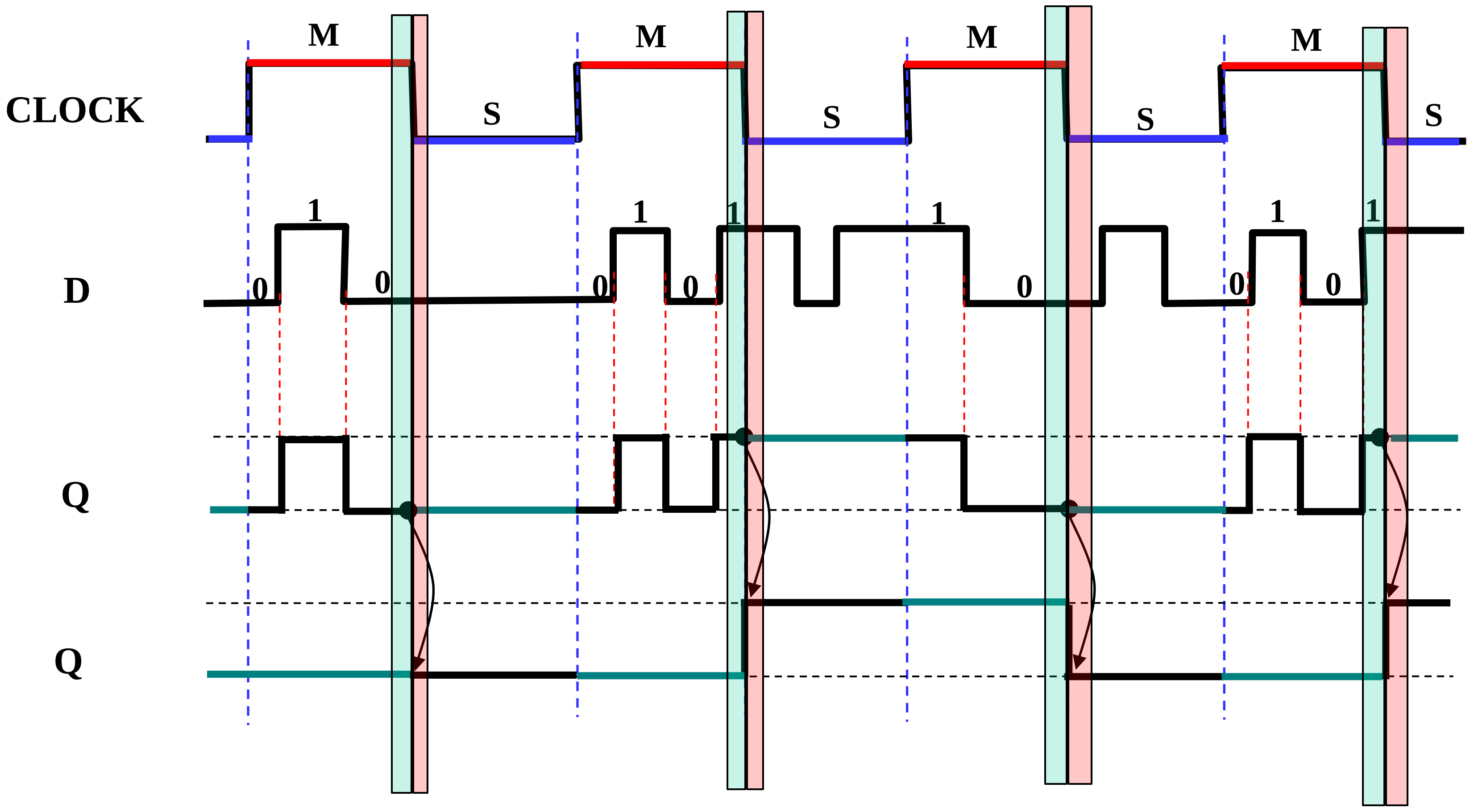
Edge-Triggered D Flip-Flop – Example 1



Edge-Triggered D Flip-Flop – Example 2

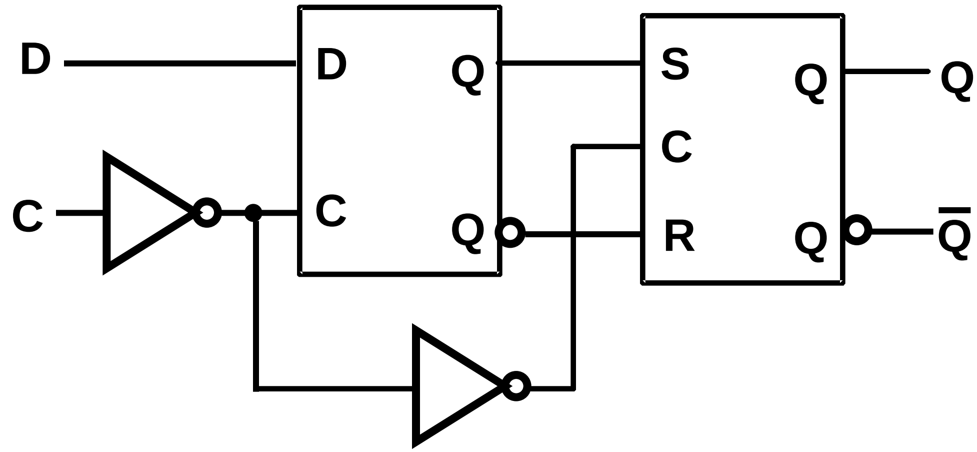


Edge-Triggered D Flip-Flop – Example 2



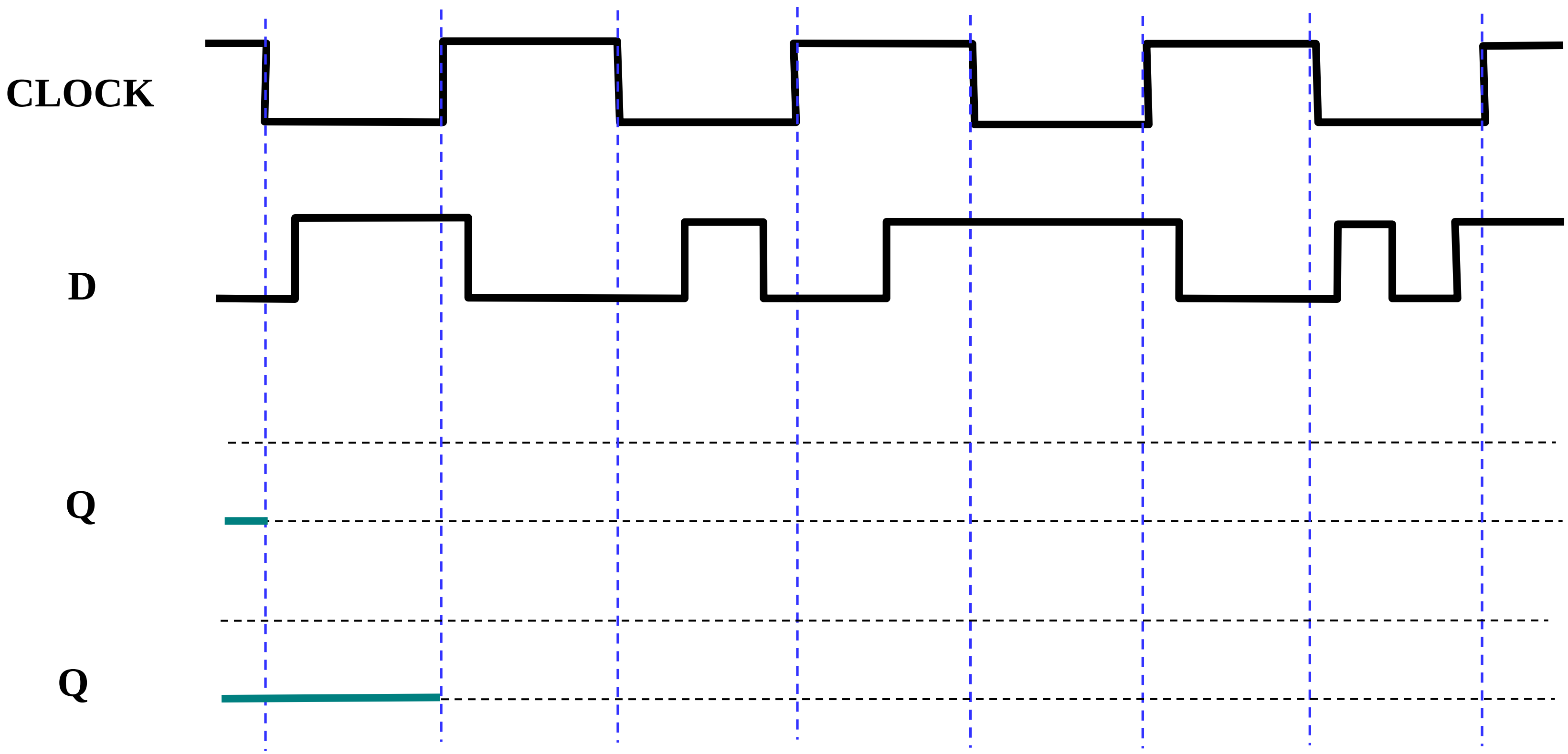
Positive-Edge Triggered D Flip-Flop

- Formed by adding inverter to clock input

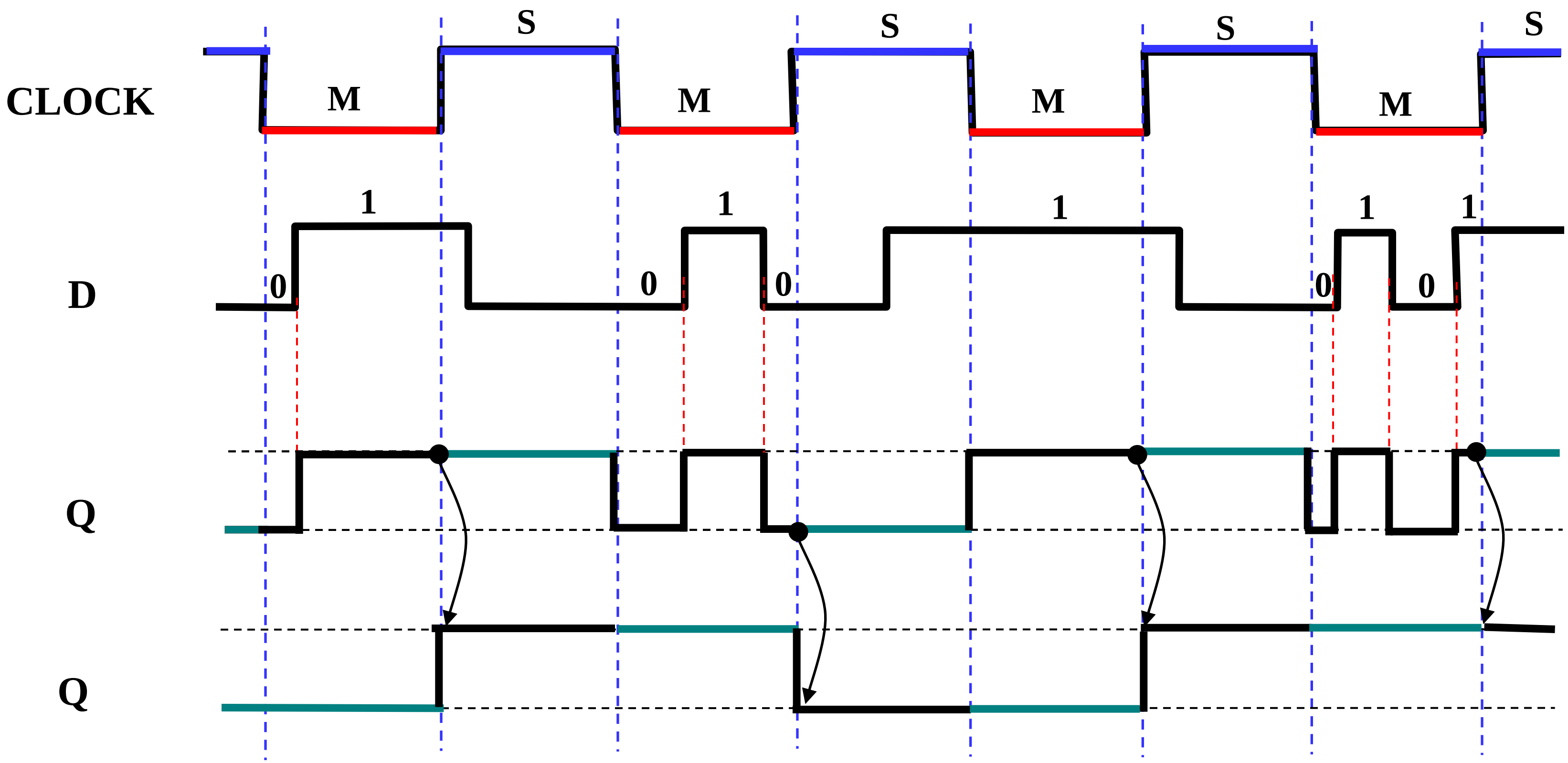


- Q changes to the value on D applied at the positive clock (rising edge) edge within timing constraints to be specified
- Our choice as the <u>standard flip-flop</u> for most sequential circuits

Positive-Edge Triggered D Flip-Flop – Example 1

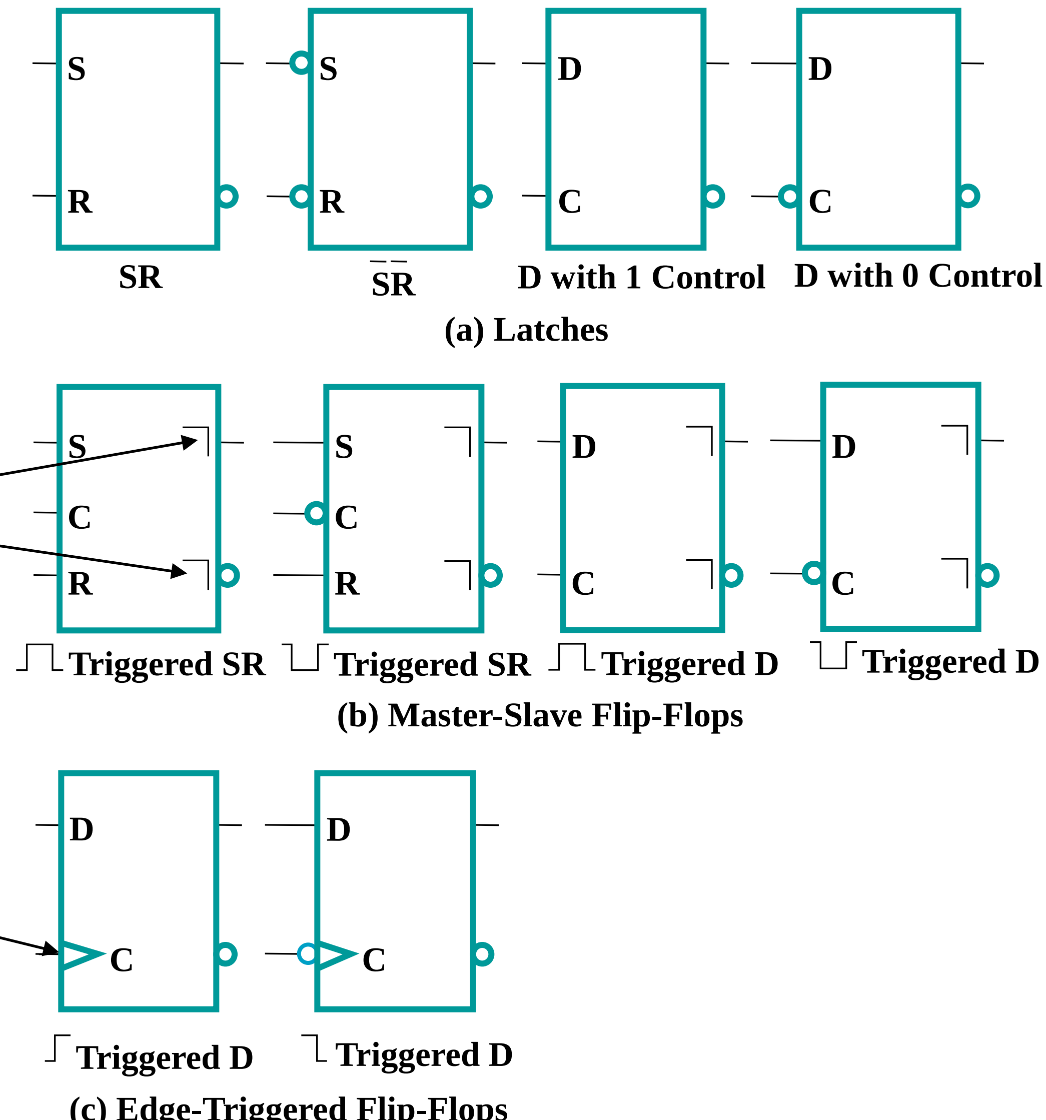


Positive-Edge Triggered D Flip-Flop – Example 1



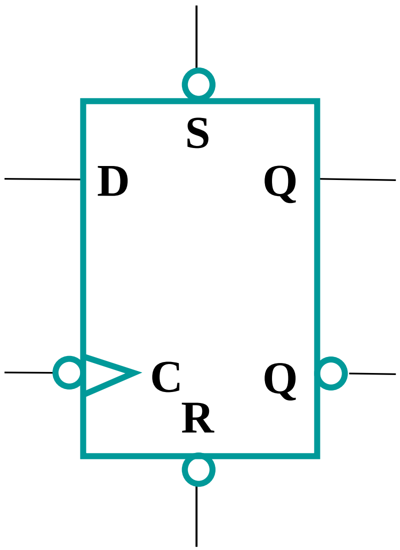
Standard Symbols for Storage Elements

- **Master-Slave: Postponed output indicators**
- **Edge-Triggered: Dynamic indicator**



Direct Inputs

- At power up or at reset, all or part of a sequential circuit usually is initialized to a known state before it begins operation
- This initialization is often done outside of the clocked behavior of the circuit, i.e., asynchronously.



- Direct R and/or S inputs that control the state of the latches within the flip-flops are used for this initialization.
- For the example flip-flop shown
  - 0 applied to $\overline{R}$ resets the flip-flop to the 0 state
  - 0 applied to $\overline{S}$ sets the flip-flop to the 1 state

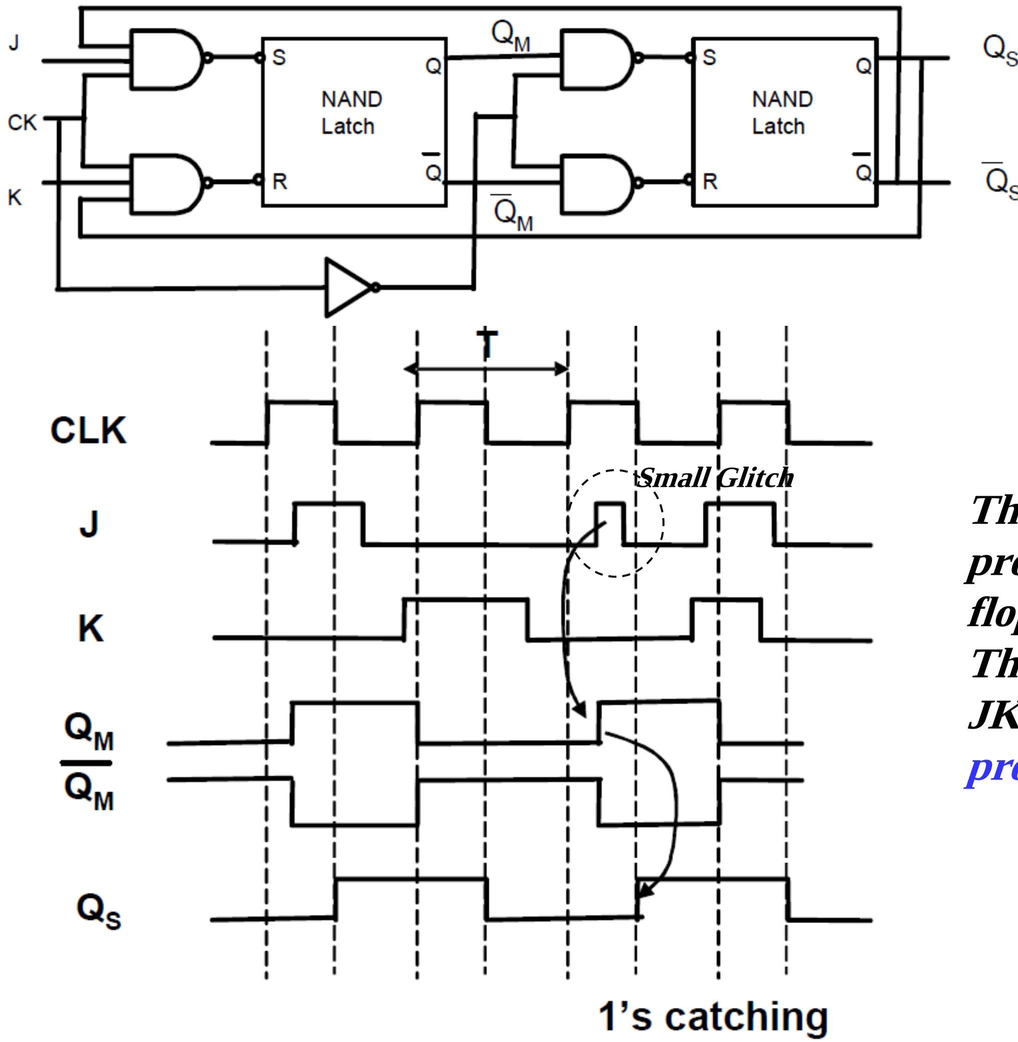
Other Flip-Flop Types

- **J-K and T flip-flops**
  - **Behavior**
  - **Implementation**
- **Basic descriptors for understanding and using different flip-flop types**
  - **Characteristic tables**
  - **Characteristic equations**
  - **Excitation tables**
- **For actual use, see Reading Supplement - Design and Analysis Using J-K and T Flip-Flops**

J-K Flip-flop

- **Behavior**
  - **Same as S-R flip-flop with J analogous to S and K analogous to R**
  - **Except that J = K = 1 is allowed, and**
  - **For J = K = 1, the flip-flop changes to the *opposite state***
  - **As a master-slave, has same "1s catching" behavior as S-R flip-flop**
  - **If the master changes to the wrong state, that state will be passed to the slave**
    - **E.g., if master falsely set by J = 1, K = 1 cannot reset it during the current clock cycle**

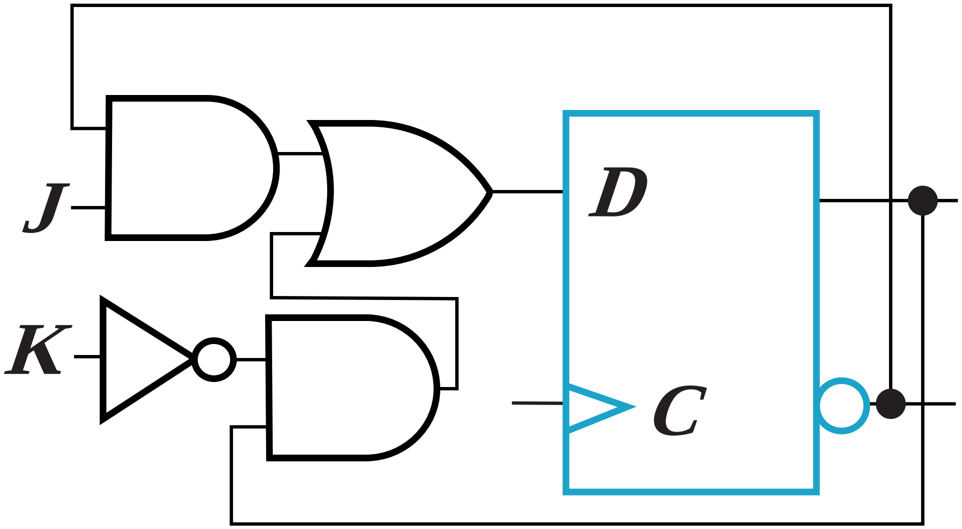
J-K Flip-flop (continued)



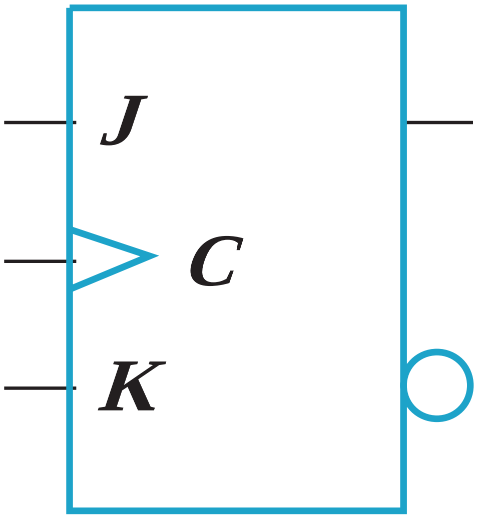
The small glitch in J propagates through the flip-flop even though it is small. This is due to the fact that the JK has the 1's catching problem.

J-K Flip-flop (continued)

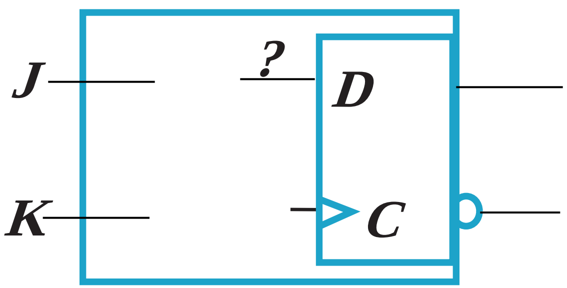
- Implementation
  - To avoid 1s catching behavior, one solution used is to use an edge-triggered D as the core of the flip-flop



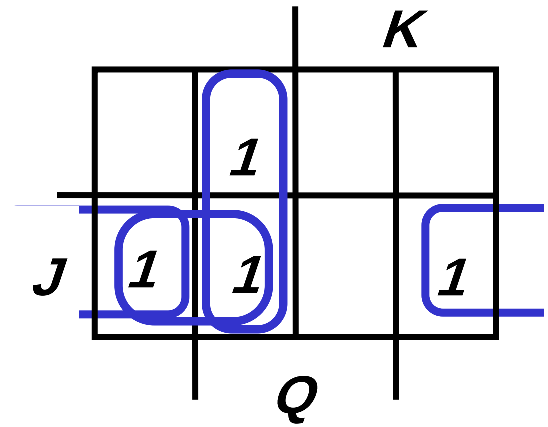
- *Symbol*



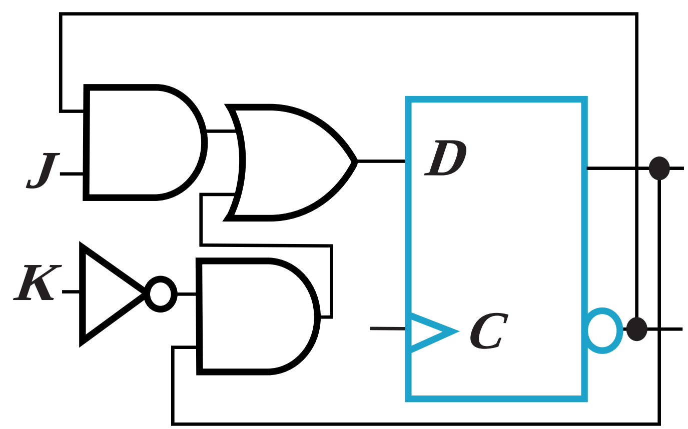
J-K Flip-flop (continued)



J	K	Q	Q(t+1)
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



$$D = J\,Q' + K'Q$$

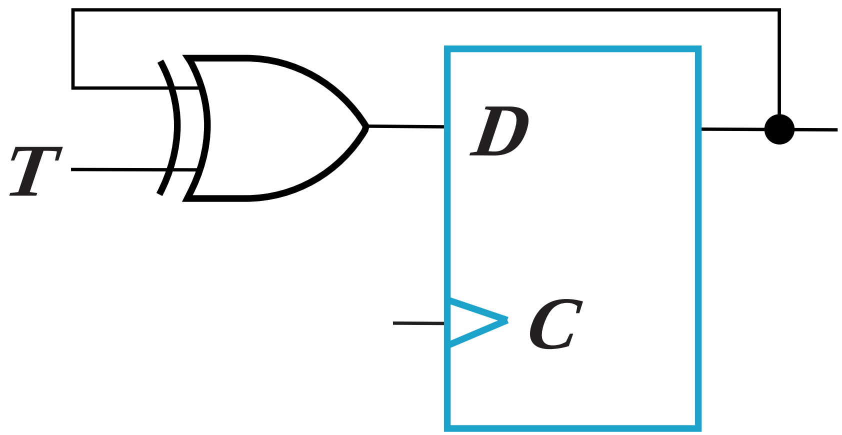


T Flip-flop

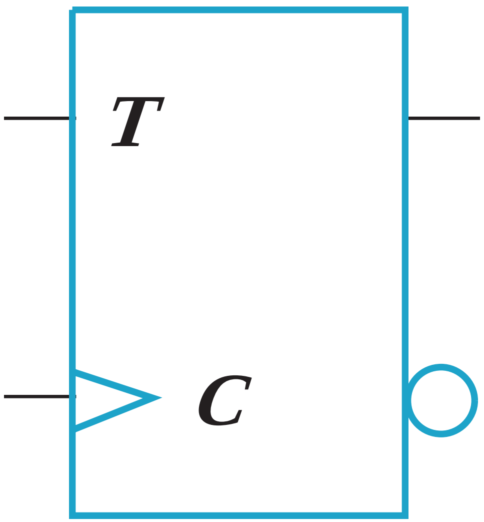
- **Behavior**
  - **Has a single input T**
    - **For T = 0, no change to state**
    - **For T = 1, changes to opposite state**
- **Same as a J-K flip-flop with J = K = T**
- **As a master-slave, has same "1s catching" behavior as J-K flip-flop**
- **Cannot be initialized to a known state using the T input**
  - **Reset (asynchronous or synchronous) essential**

T Flip-flop (continued)

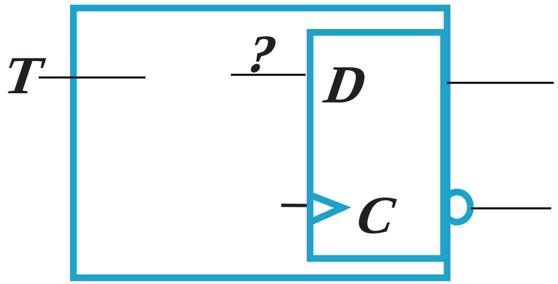
- **Implementation**
  - To avoid 1s catching behavior, one solution used is to use an edge-triggered D as the core of the flip-flop

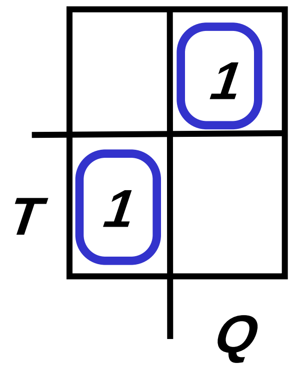


- ***Symbol***



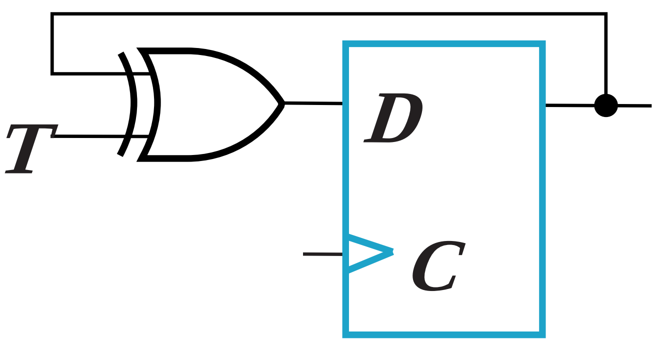
T Flip-flop (continued)





T	Q	Q(t+1)
0	0	0
0	1	1
1	0	1
1	1	0

$$D = T \oplus Q$$



Basic Flip-Flop Descriptors

- **Used in analysis**
  - ***Characteristic table*** **- defines the next state of the flip-flop in terms of flip-flop inputs and current state**
  - ***Characteristic equation*** **- defines the next state of the flip-flop as a Boolean function of the flip-flop inputs and the current state**
- **Used in design**
  - ***Excitation table*** **- defines the flip-flop input variable values as function of the current state and next state**

D Flip-Flop Descriptors

- **Characteristic Table**

D	*Q(t +1)*	*Operation*
0	*0*	*Reset*
1	*1*	*Set*

- **Characteristic Equation**

  Q(t+1) = D

- **Excitation Table**

Q(t +1)	*D*	*Operation*
0	*0*	*Reset*
1	*1*	*Set*

T Flip-Flop Descriptors

- **Characteristic Table**

T	*Q(t+1)*	*Operation*
0	*Q(t)*	*No change*
1	$\overline{Q}(t)$	*Complement*

- **Characteristic Equation**

$$Q(t+1) = T \oplus Q$$

- **Excitation Table**

Q(t+1)	*T*	*Operation*
Q(t)	*0*	*No change*
$\overline{Q}(t)$	*1*	*Complement*

S-R Flip-Flop Descriptors

- **Characteristic Table**

S	*R*	*Q(t+1)*	*Operation*
0	*0*	*Q(t)*	*No change*
0	*1*	*0*	*Reset*
1	*0*	*1*	*Set*
1	*1*	*?*	*Undefined*

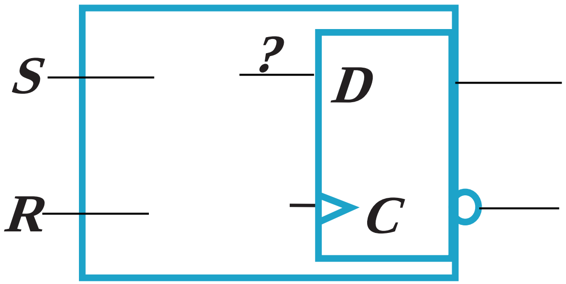
- **Characteristic Equation**

$$Q(t+1) = S + \overline{R}\,Q,\ S \cdot R = 0$$

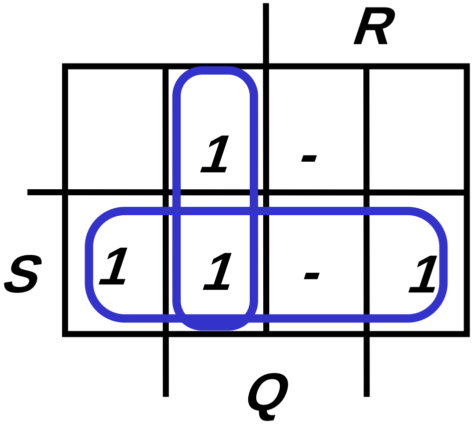
- **Excitation Table**

Q(t)	*Q(t+1)*	*S*	*R*	*Operation*
0	*0*	*0*	*X*	*No change*
0	*1*	*1*	*0*	*Set*
1	*0*	*0*	*1*	*Reset*
1	*1*	*X*	*0*	*No change*

S-R Flip-flop



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



SR=0

$$D = S + R'Q\,, \quad SR = 0$$

J-K Flip-Flop Descriptors

- **Characteristic Table**

J	K	Q(t+1)	Operation
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$\overline{Q}(t)$	Complement

- **Characteristic Equation**

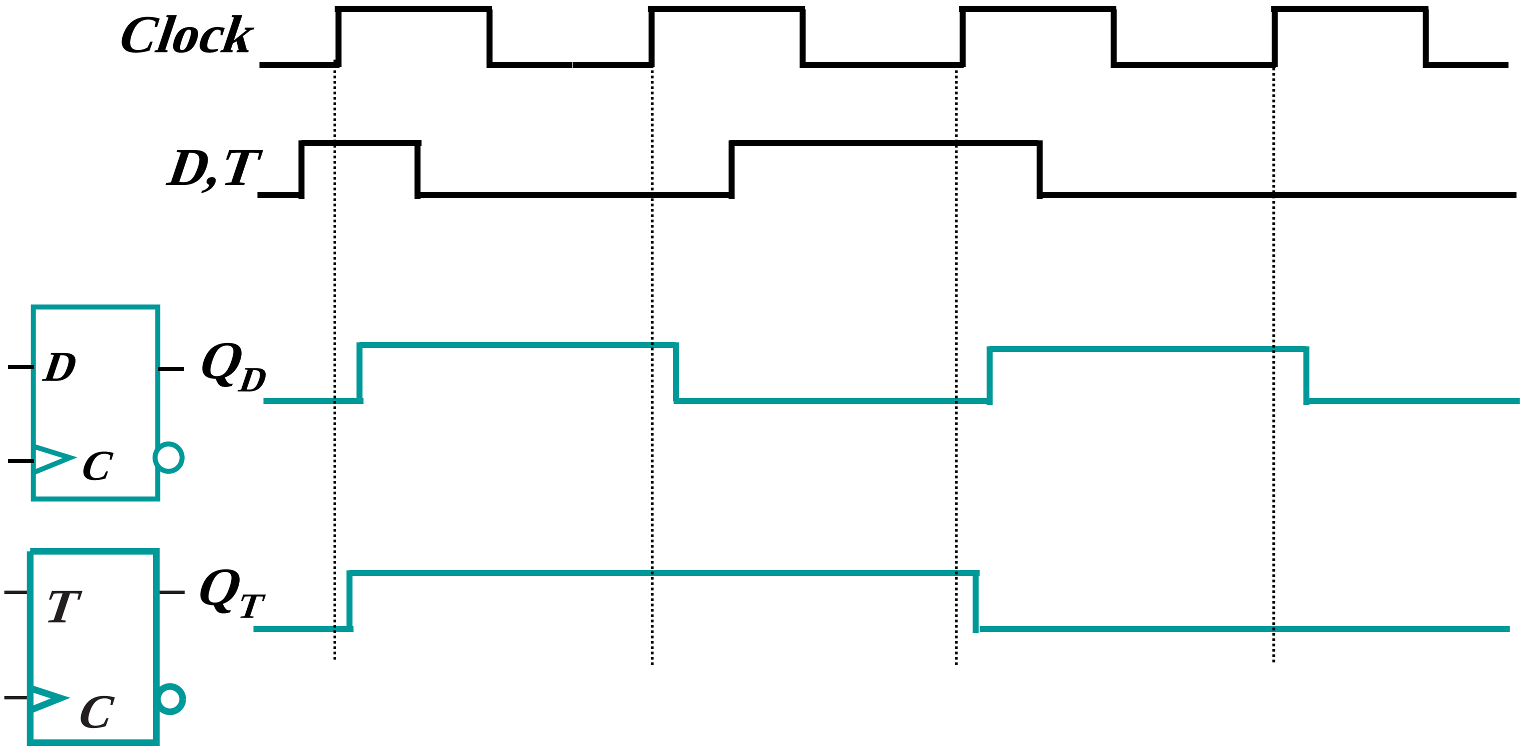
$$Q(t+1) = J\,\overline{Q} + \overline{K}\,Q$$

- **Excitation Table**

Q(t)	Q(t+1)	J	K	Operation
0	0	0	X	No change
0	1	1	X	Set
1	0	X	1	Reset
1	1	X	0	No Change

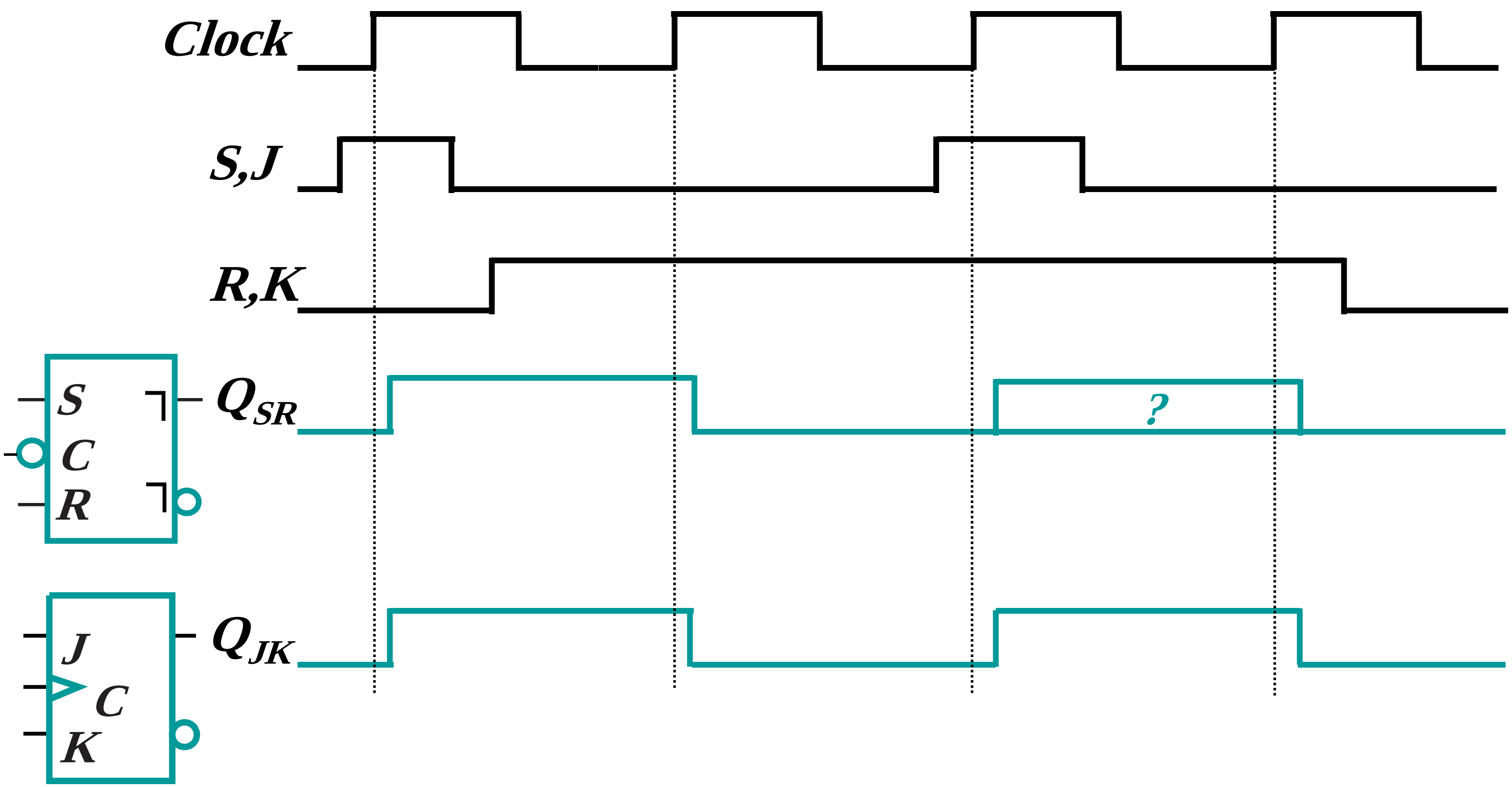
Example 1: Flip-flop Behavior

- **Use the characteristic tables to find the output waveforms for the flip-flops shown:**



Example 1: Flip-Flop Behavior (continued)

- **Use the characteristic tables to find the output waveforms for the flip-flops shown:**



Terms of Use

- **All (or portions) of this material © 2008 by Pearson Education, Inc.**
- **Permission is given to incorporate this material or adaptations thereof into classroom presentations and handouts to instructors in courses adopting the latest edition of Logic and Computer Design Fundamentals as the course textbook.**
- **These materials or adaptations thereof are not to be sold or otherwise offered for consideration.**
- **This Terms of Use slide or page is to be included within the original materials or any adaptations thereof.**