Design of Digital Systems II
Sequential Logic Design Principles (1)

Moslem Amiri, Václav Přenosil
Masaryk University

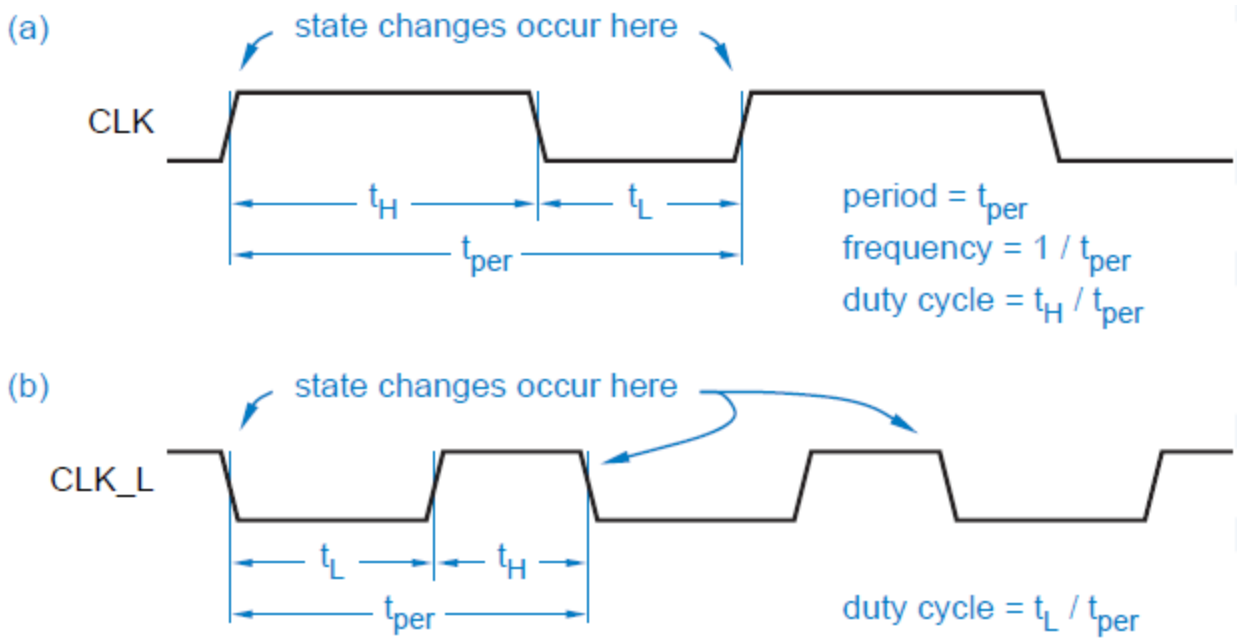
Resource: “Digital Design: Principles & Practices”
by John F. Wakerly

Introduction

- Logic circuits are classified into two types, "**combinational**" and "**sequential**".
- A **combinational** logic circuit is one whose outputs depend only on its current inputs.
- A **sequential** logic circuit is one whose outputs depend not only on its current inputs, but also on past sequence of inputs, possibly arbitrarily far back in time.
- So it is **inconvenient**, and often impossible, to describe behavior of a sequential circuit by means of a **table** that lists outputs as a function of input sequence that has been received up until current time.
- **State** of a sequential circuit is a collection of **state variables** whose values at any one time contain **all information about past necessary** to account for circuit's future behavior.
- In a digital logic circuit, **state variables are binary values**, corresponding to certain logic signals in circuit.
- A circuit with *n* binary state variables has 2^n possible states.

Introduction

- State changes of most sequential circuits occur at times specified by a free-running **clock** signal.
- Clock signals: (a) active high (b) active low:



- By convention, a clock signal is **active high** if state changes occur at **clock's rising edge** or when clock is **HIGH**, and active low in complementary case.
- First edge or pulse in a clock period or sometimes period itself is called a **clock tick**.

Introduction

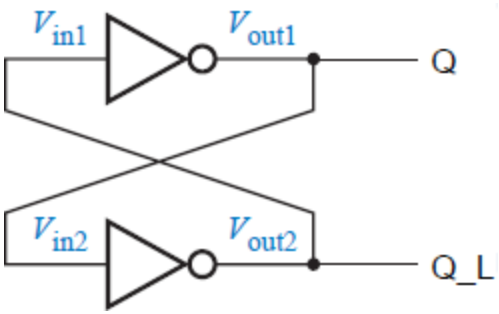
- There are generally **two types of sequential circuits**:

  1) **feedback sequential circuit** which uses ordinary gates and feedback loops to obtain memory in a logic circuit, thereby creating sequential circuit building blocks such as latches and flip-flops that are used in higher-level designs.

  2) **clocked synchronous state machine** which uses these building blocks, in particular edge-triggered D flip-flops, to create circuits whose inputs are examined and whose outputs change in accordance with a controlling clock signal.

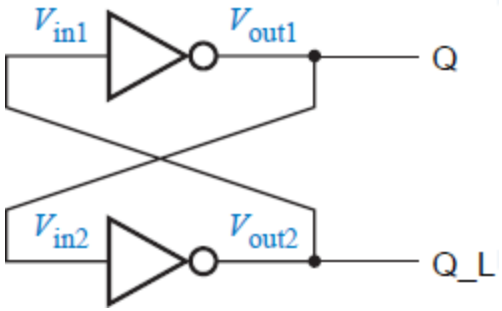
Bistable Elements: Digital Analysis

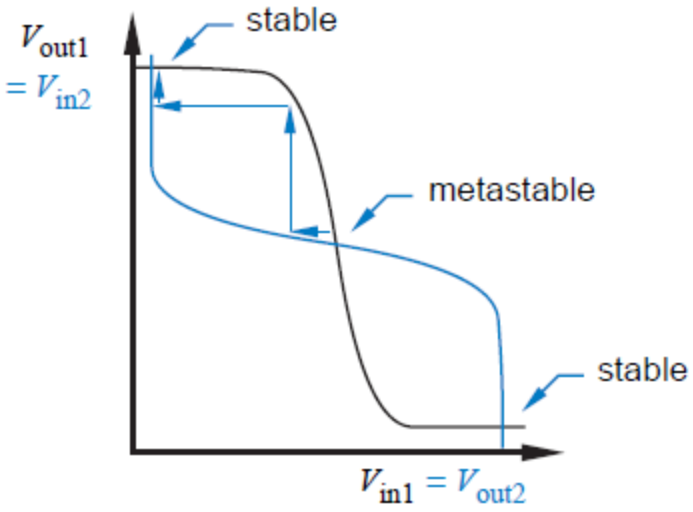
- Simplest sequential circuit (it has **no inputs**):



- This circuit is often called a **bistable**, since a strictly digital analysis shows that it has **two stable states**.
- If Q is HIGH, bottom inverter has a HIGH input and a LOW output, which forces the top inverter's output HIGH as we assumed in first place.
- If Q is LOW, bottom inverter has a LOW input and a HIGH output, which forces Q LOW, another stable situation.
- There are **two possible states**, Q = 0 and Q = 1.

Bistable Elements: Analog Analysis





Transfer function:

$V_{out1} = T(V_{in1})$

$V_{out2} = T(V_{in2})$

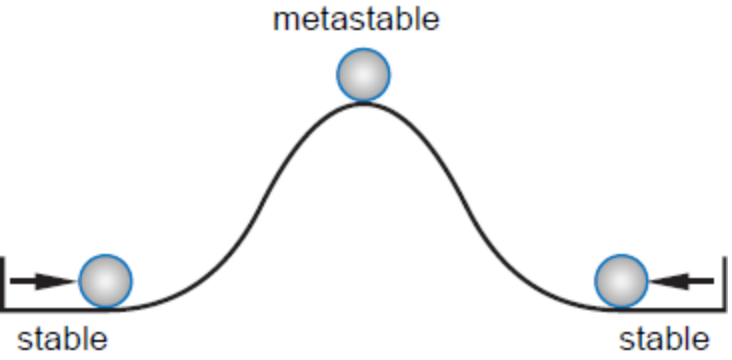
- Lines in figure above (right) show steady-state (**DC**) **transfer functions** T for inverters.
- Feedback loop is in **equilibrium** if **input and output** voltages of both inverters are **constant** DC values consistent with loop connection and inverters' DC transfer function. That is, we must have:

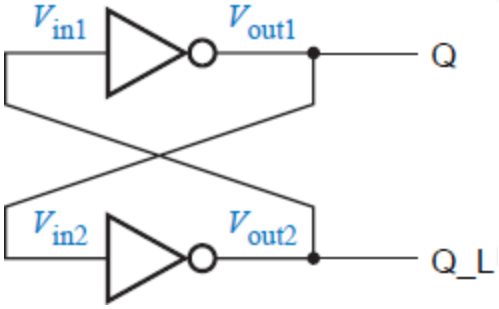
$$V_{in1} = V_{out2} = T(V_{in2}) = T(V_{out1}) = T(T(V_{in1}))$$

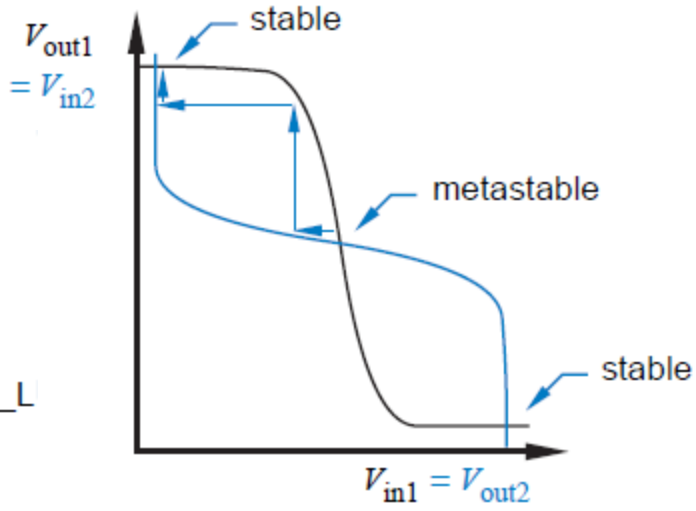
- Likewise we must have: $V_{in2} = T(T(V_{in2}))$

Bistable Elements: Analog Analysis

- These **equilibrium points** are ones in figure at which two transfer **curves meet**. There are **three** equilibrium **points**, two stable and one metastable.
- **Metastable point** occurs with V_{out1} and V_{out2} about **halfway** between a valid **logic 1** voltage and a valid **logic 0** voltage. *Q* and *Q_L* are **not valid logic signals** at this point. Yet loop equations are satisfied; if we can get circuit to operate at metastable point, it could **theoretically stay** there **indefinitely**.
- Metastable point is **not truly stable**, because **random noise** will tend to drive a circuit that is operating at metastable point toward one of stable operating points (consecutive arrows on right figure).







Bistable Elements

- **Bistable may stay** in **metastable** state for an **unpredictable length of time** before nondeterministically settling into one stable state or the other.
- If **simplest** sequential circuit is **susceptible to metastable** behavior, then **all sequential circuits are susceptible** and this behavior is not something that only occurs at power-up.
- Example: in **S-R flip-flops**, a pulse on *S* input forces flip-flop from 0 state to 1 state. A minimum pulse width is specified for *S* input.

  1) Apply a pulse of this width or longer, and flip-flop immediately goes to 1 state.

  2) Apply a very short pulse, and flip-flop stays in 0 state.

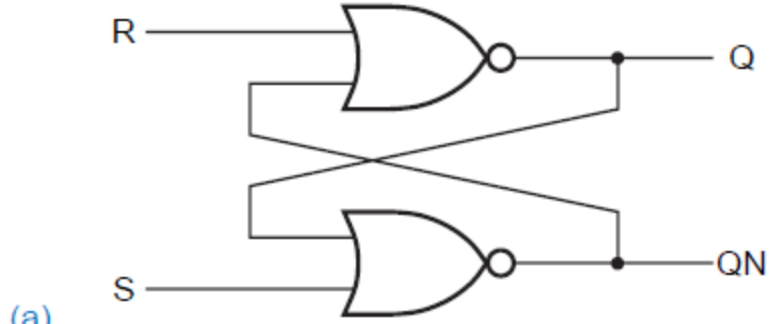
  3) Apply a pulse just under minimum width, and flip-flop may go into metastable state.
- Once flip-flop is in metastable state, its operation depends on "**shape of its hill**".

Latches and Flip-Flops

- Latches and flip-flops are basic building blocks of most sequential circuits.
- **Flip-flop** is a sequential device that normally **samples its inputs** and changes **its outputs** only at times determined by a **clocking signal**.
- **Latch** is a sequential device that **watches all of its inputs continuously** and **changes its outputs at any time**, independent of a clocking signal.

S-R Latch

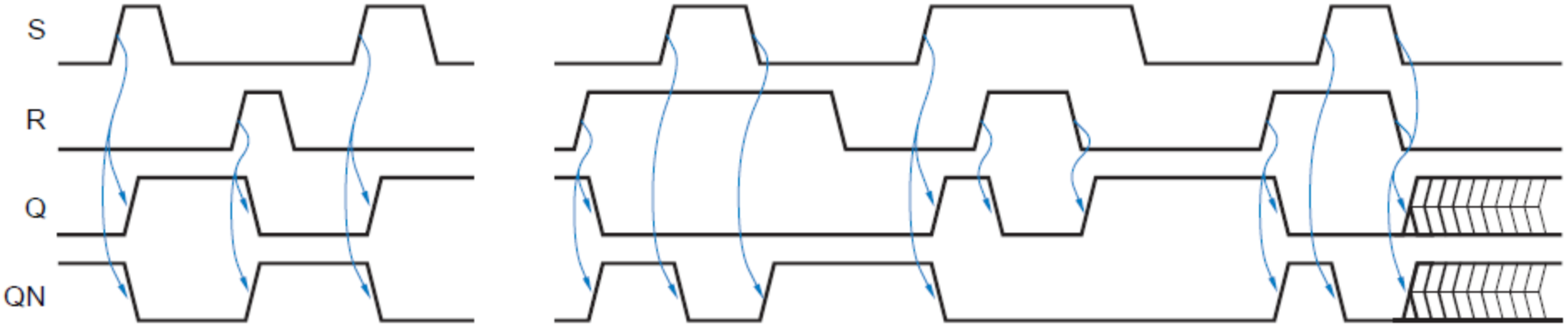
- An S-R (set-reset) latch based on NOR gates is shown here:



(a)

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

(b)



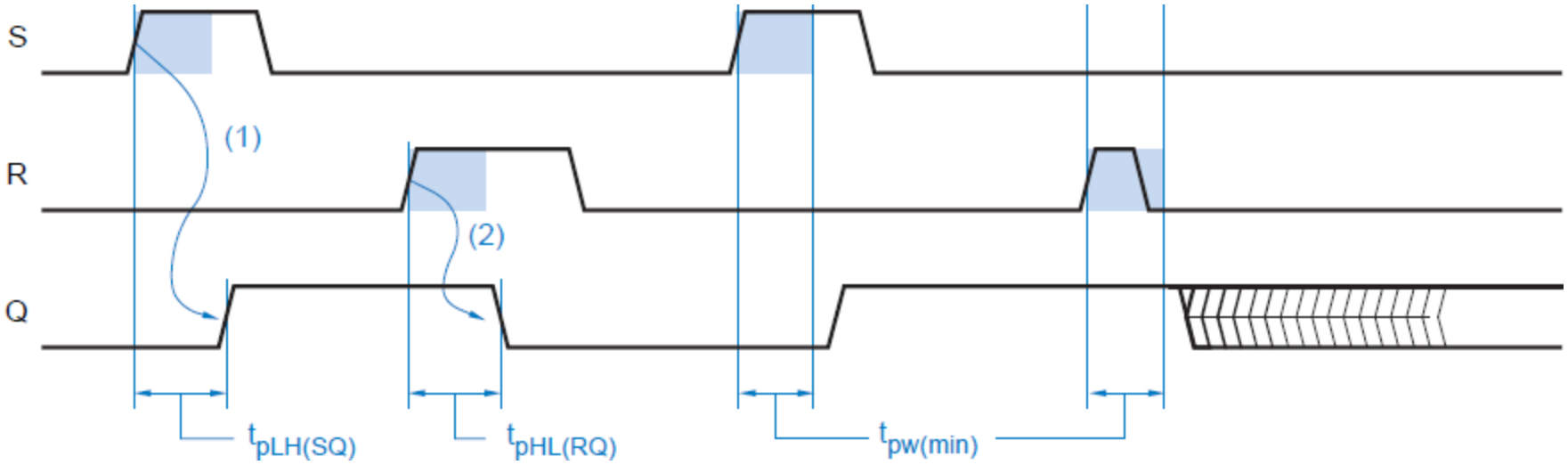
- In practice, we avoid applying **11** to a NOR latch because **outputs** of latch will **not be logical complements of each other** and because in a physical circuit a **race condition** occurs if inputs are changed from **11 to 00**.

S-R Latch

- **Propagation delay** is time it takes for a transition on an input signal to produce a transition on an output signal.
- A given latch or flip-flop may have **several different propagation delay specifications**, one for each pair of input and output signals.
- Also, propagation delay may be different depending on whether output makes a **LOW-to-HIGH** or **HIGH-to-LOW** transition.

S-R Latch

- **Timing parameters** for an S-R latch:



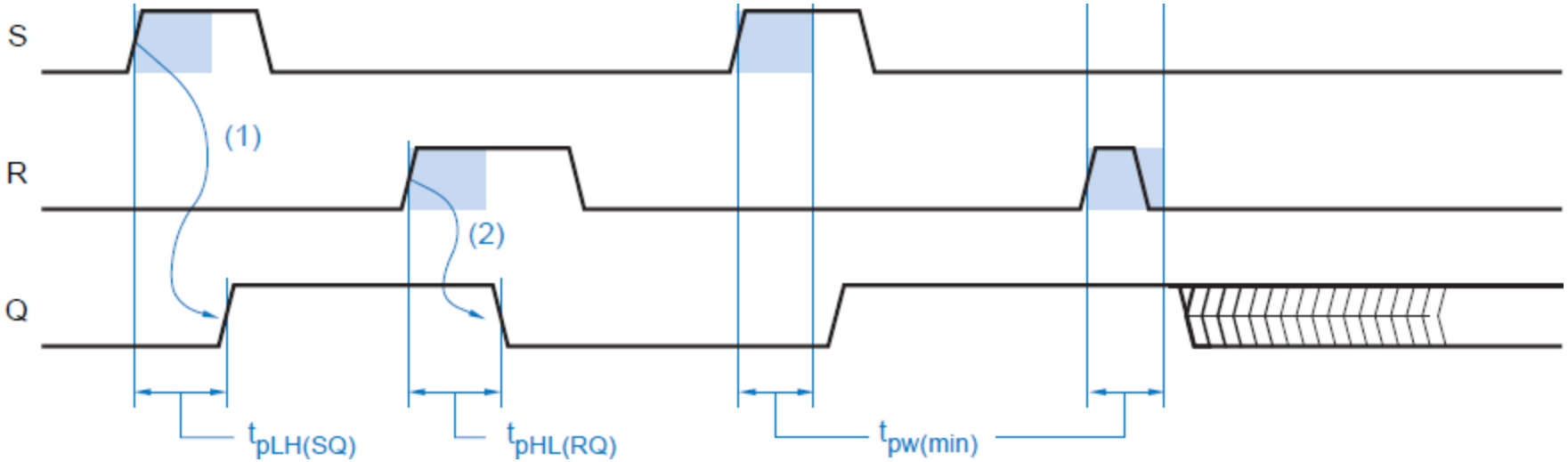
- In figure above:

  transition 1: a LOW-to-HIGH transition on S causing a LOW-to-HIGH transition on Q, so a propagation delay $t_{pLH(SQ)}$ occurs.

  transition 2: a LOW-to-HIGH transition on R causing a HIGH-to-LOW transition on Q, with propagation delay $t_{pHL(RQ)}$.

- Corresponding transitions on QN would have propagation delays $t_{pHL(SQN)}$ and $t_{pLH(RQN)}$.

S-R Latch

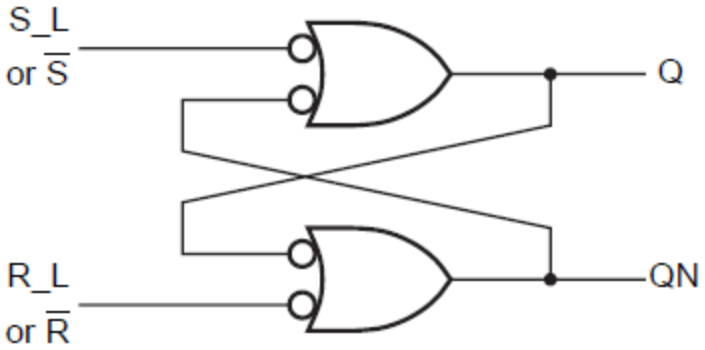


- **Minimum pulse width specifications** are usually given for S and R inputs.
- Latch may go into **metastable state** and remain there for a random length of time if a **pulse shorter than minimum width** $t_{pw(\min)}$ is applied to S or R.
- Latch can be **deterministically brought out of metastable state** only by applying a pulse to S or R that meets or exceeds minimum pulse width requirements.

$\overline{S} - \overline{R}$ Latch

- An $\overline{S} - \overline{R}$ latch (S-bar-R-bar latch) with **active-low set and reset** inputs may be built from NAND gates:

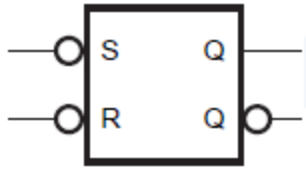
(a)



(b)

S_L	R_L	Q	QN
0	0	1	1
0	1	1	0
1	0	0	1
1	1	last Q	last QN

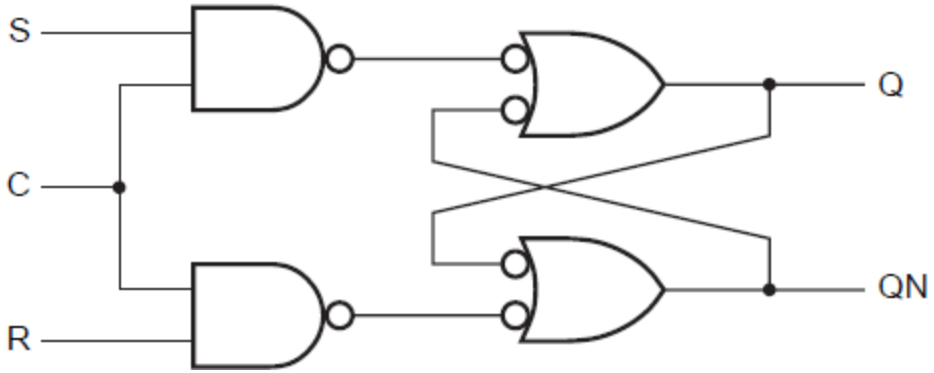
(c)



- In TTL and CMOS logic families, $\overline{S} - \overline{R}$ latches are used much more often than S-R latches because **NAND gates are preferred** over NOR gates.
- Regarding timing and metastability considerations, operation of the $\overline{S} - \overline{R}$ is the same as the S-R.

S-R Latch with Enable

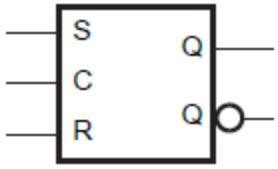
- An S-R or $\overline{S}-\overline{R}$ latch is **sensitive to its S and R inputs at all times.** However, it may easily be modified to create a device that is **sensitive** to these inputs **only when** an enabling input **C is asserted**. Such an **S-R** latch with enable is shown here:

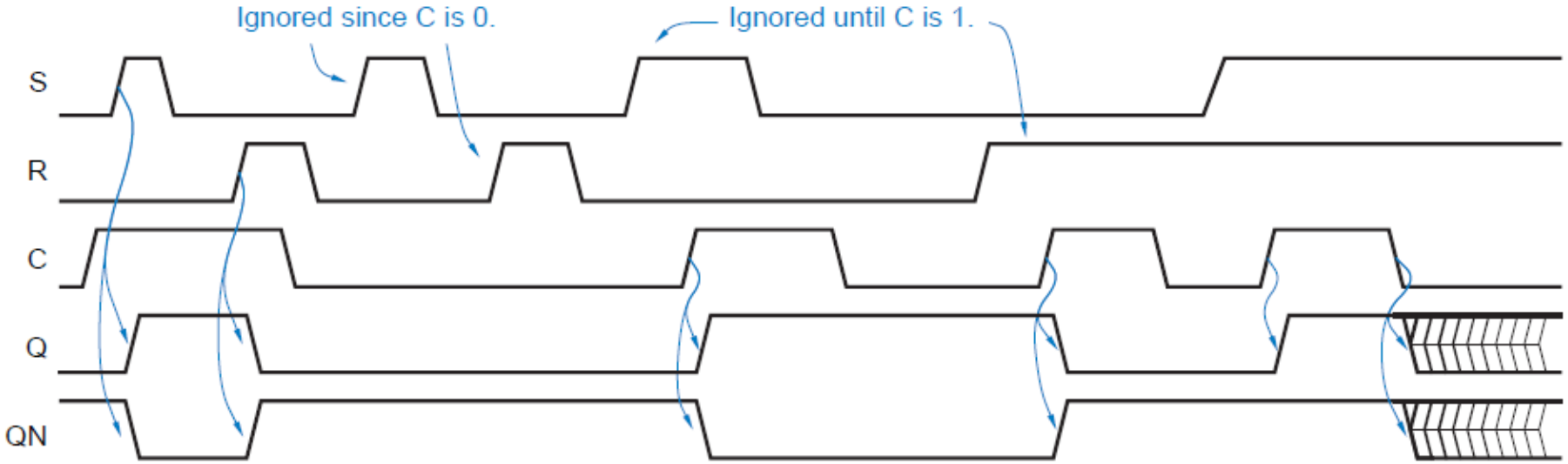


(b)

S	R	C	Q	QN
0	0	1	last Q	last QN
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1
x	x	0	last Q	last QN

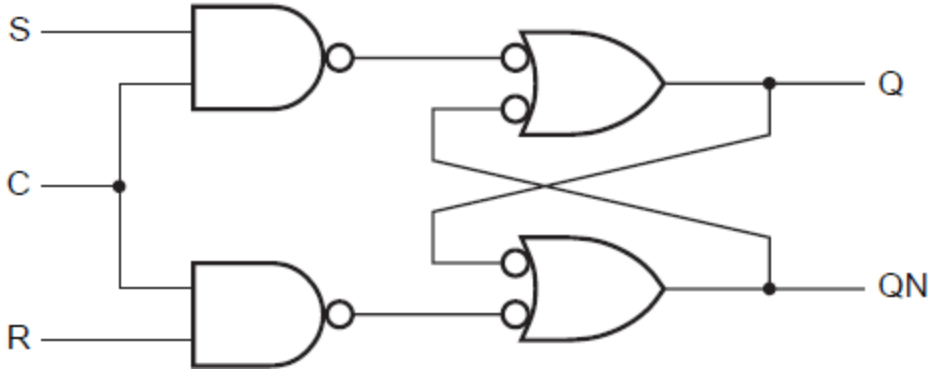
(c)





S-R Latch with Enable

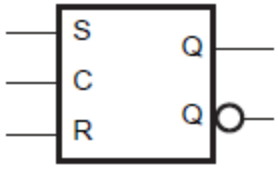
- If both S and R are 1 when C changes from 1 to 0, circuit behaves like an S-R latch in which S and R are negated simultaneously – next state is unpredictable and output may become **metastable**.

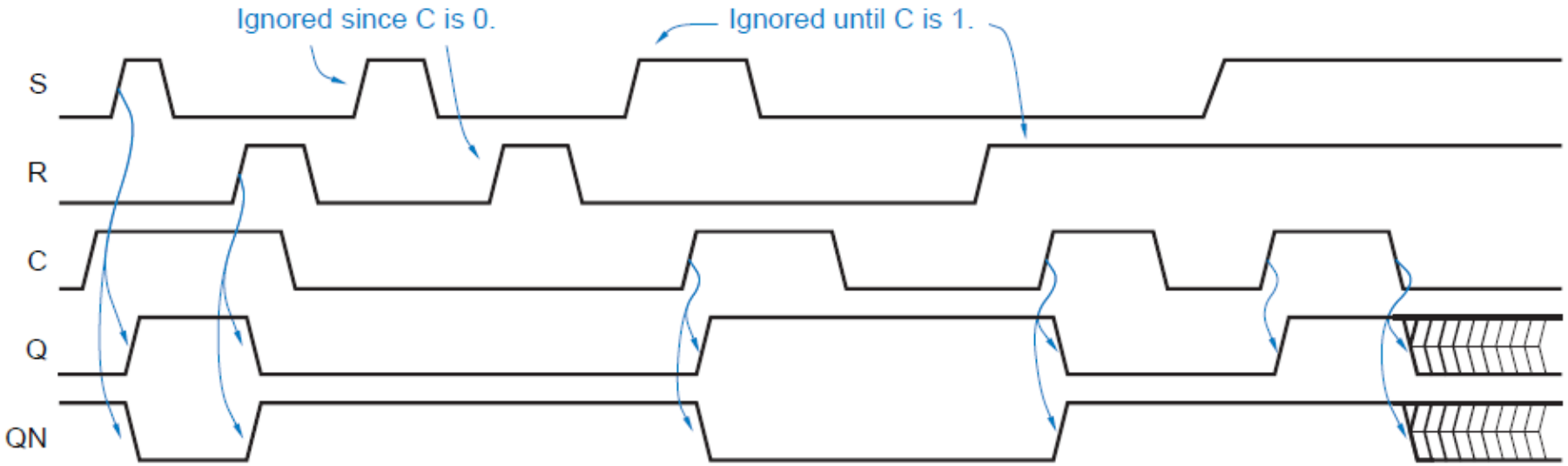


(b)

S	R	C	Q	QN
0	0	1	last Q	last QN
0	1	1	0	1
1	0	1	1	0
1	1	1	1	1
x	x	0	last Q	last QN

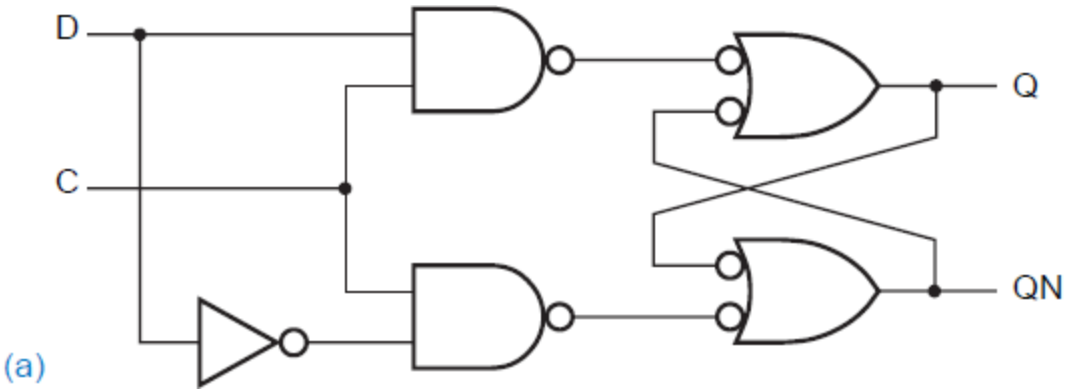
(c)





D Latch

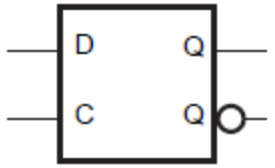
- **S-R latches** are useful in **control applications**, where we set a flag in response to some condition, and reset it when conditions change; so we control set and reset inputs independently.
- We often need latches simply to **store bits of information**. A **D latch** may be used in such an application:



(a)

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

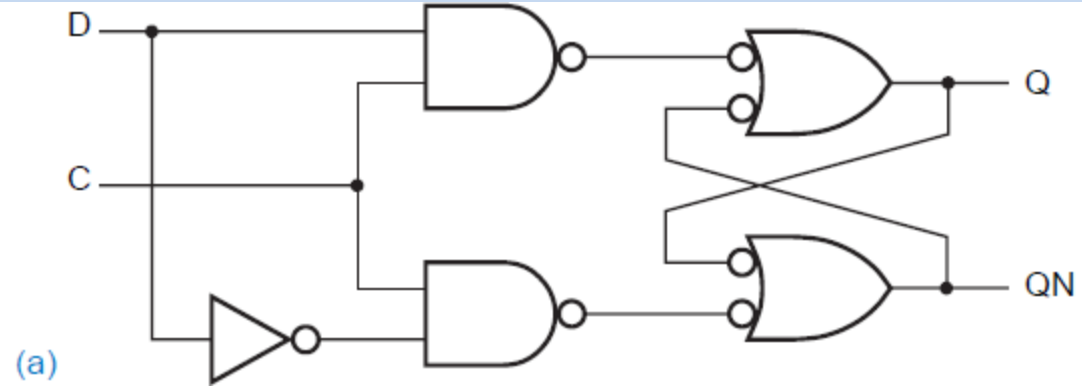
(b)



(c)

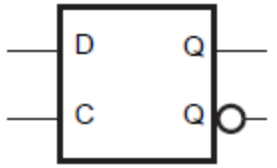
- **Inverter added eliminates troublesome situation** in S-R latches, where S and R may be asserted simultaneously.

D Latch

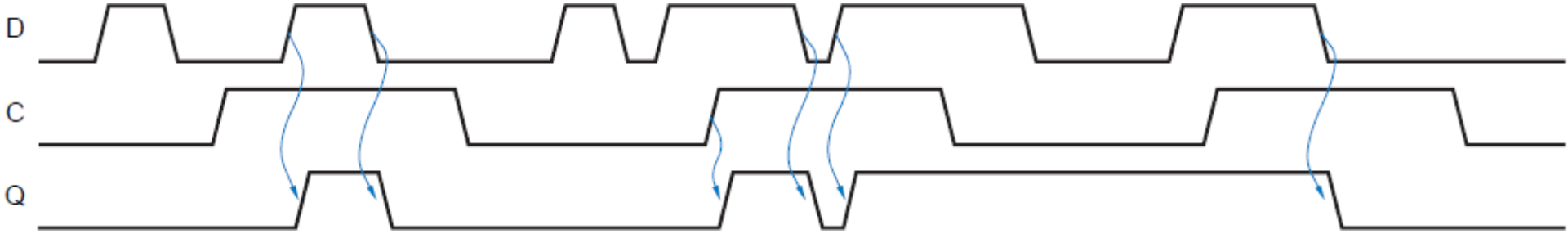


C	D	Q	QN
1	0	0	1
1	1	1	0
0	x	last Q	last QN

(b)

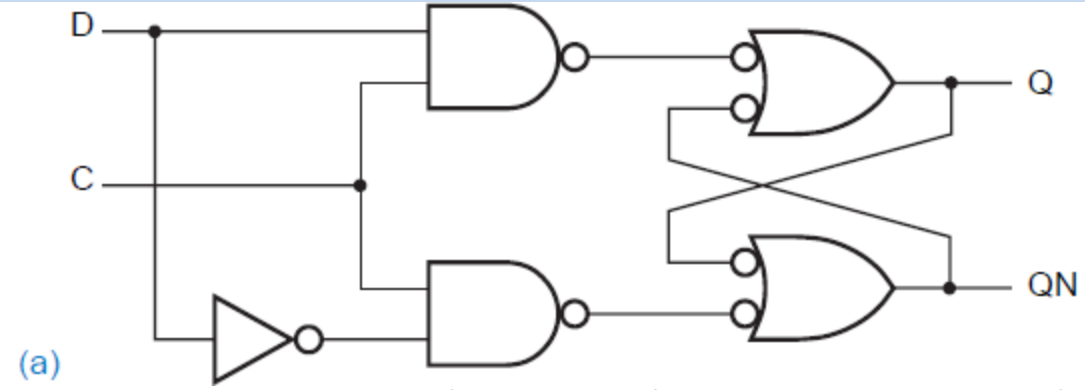


- Functional behavior of a D latch for various inputs:

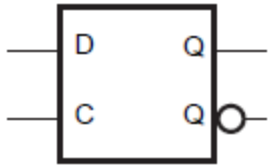


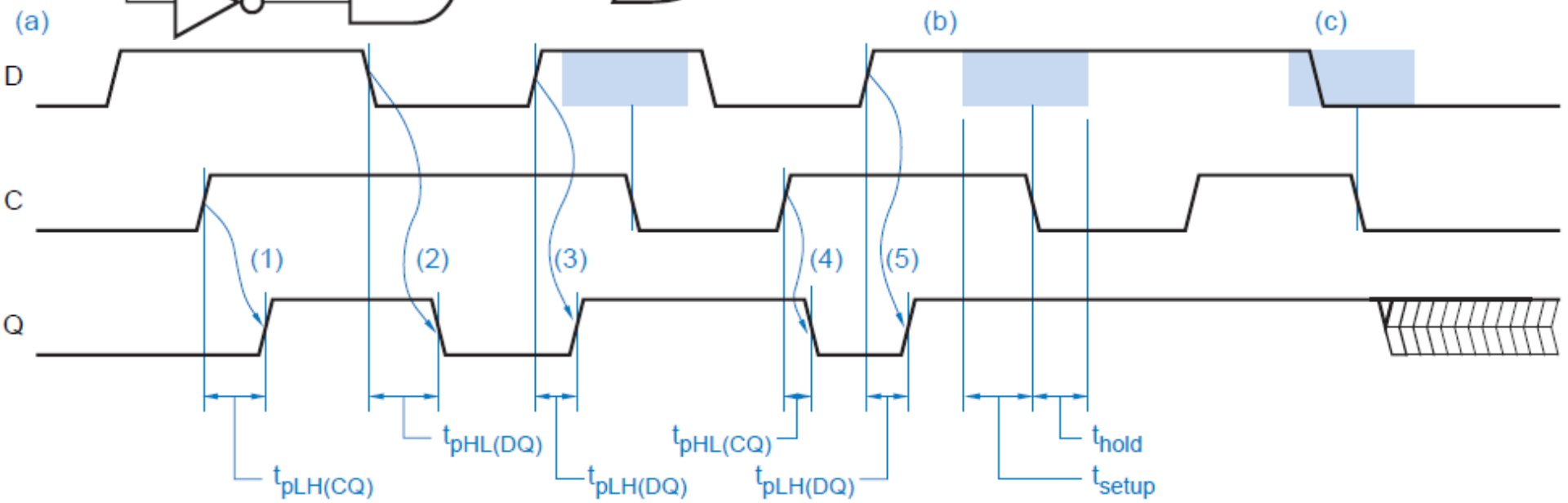
- When **C input is asserted**, Q output follows D input. In this situation, latch is said to be "**open**" and path from D input to Q output is "**transparent**"; circuit is called a "**transparent latch**" for this reason.
- When **C input is negated**, latch "**closes**"; Q output retains its **last value** and no longer changes in response to D, as long as C remains negated.

D Latch: Timing Parameters



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



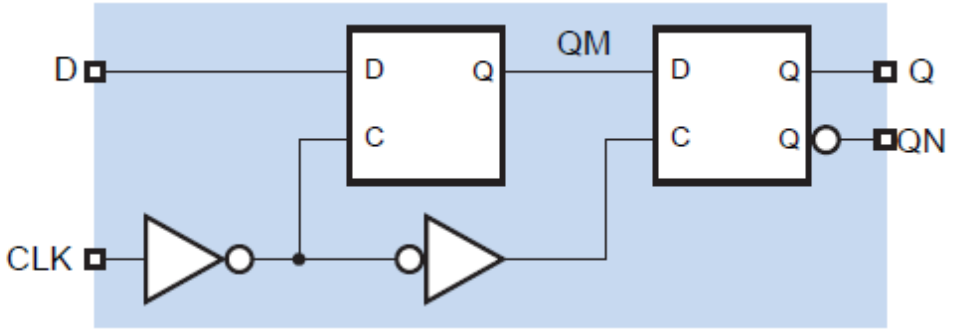


- **Setup time** (**Hold time**) is minimum amount of time the data signal should be held steady **before** (**after**) clock event so that data are reliably sampled by clock. Although D latch eliminates S = R = 1 problem of S-R latch, it does not eliminate metastability problem (unpredictable or metastable).

Edge-Triggered D Flip-Flop

- A **positive-edge-triggered D flip-flop** combines **a pair of D latches** to create a circuit that samples its D input and changes its Q and QN outputs only at rising edge of a controlling CLK signal:

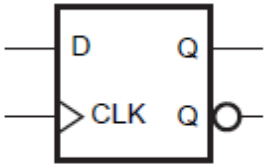
(a)



(b)

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

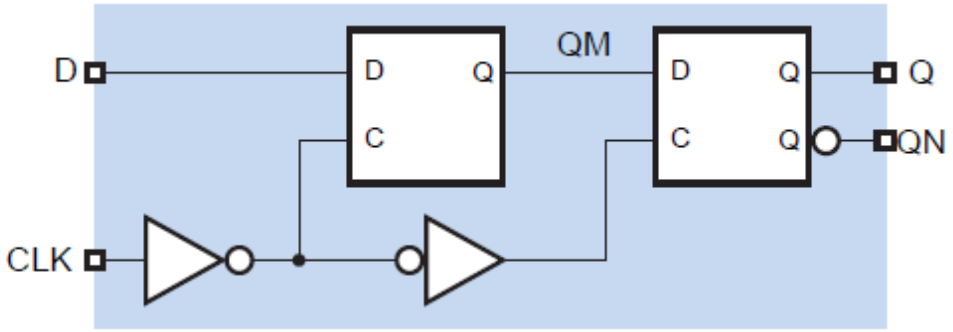
(c)



- First latch is called **master**; it is **open** and follows input **when CLK is 0**. When **CLK goes to 1**, **master** latch is **closed** and its output is transferred to second latch, called **slave**.
- **Slave** latch is **open** all the while that **CLK is 1**, but **changes only at beginning of this interval**, because master is closed and unchanging during rest of interval.
- **Triangle** on D flip-flop's CLK input indicates **edge-triggered** behavior, and is called a "dynamic-input indicator".

Edge-Triggered D Flip-Flop

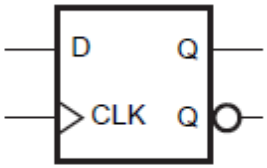
(a)



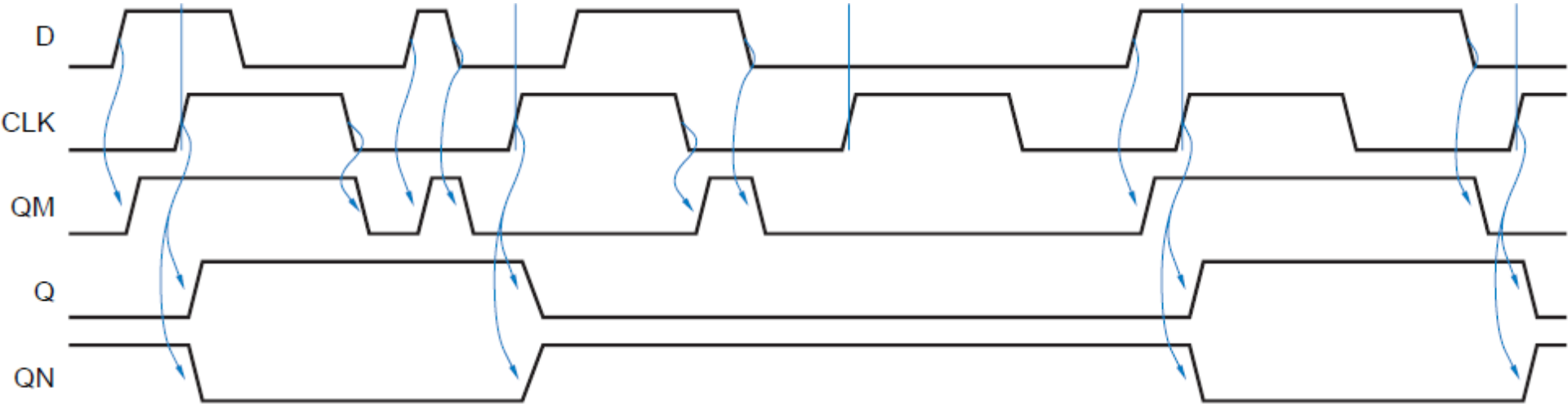
(b)

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

(c)

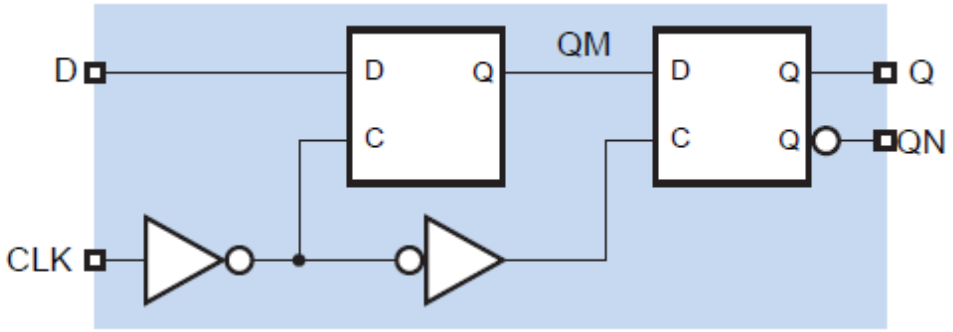


- Functional behavior of a positive-edge-triggered D flip-flop:



Edge-Triggered D Flip-Flop

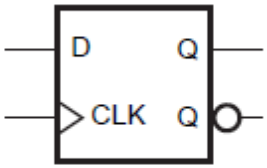
(a)



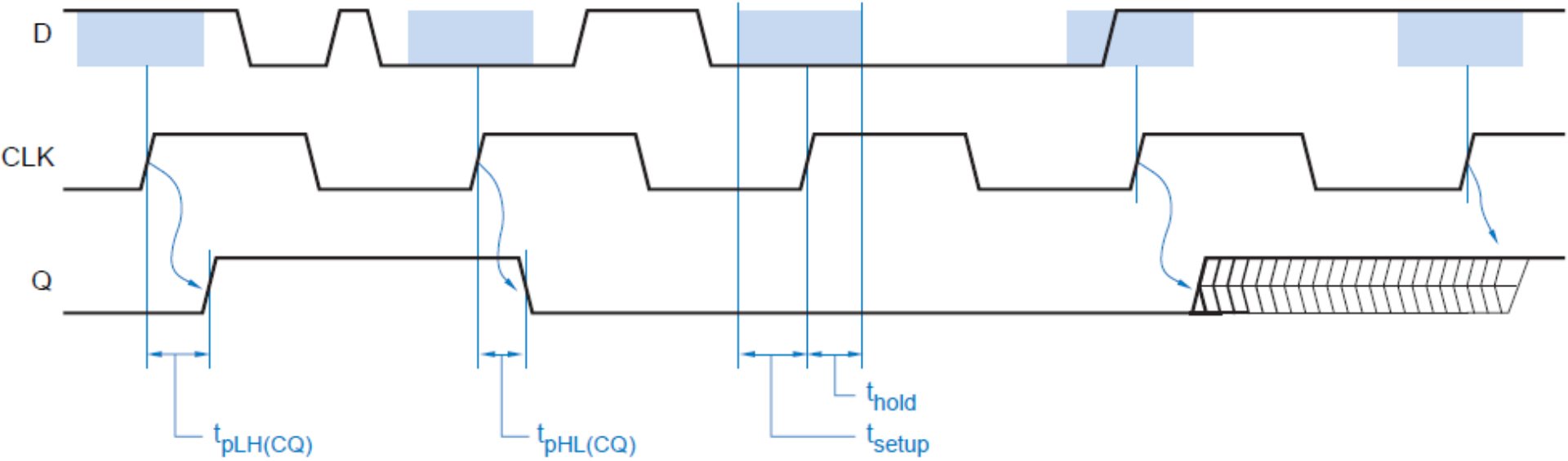
(b)

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

(c)



- Timing behavior of a positive-edge-triggered D flip-flop:



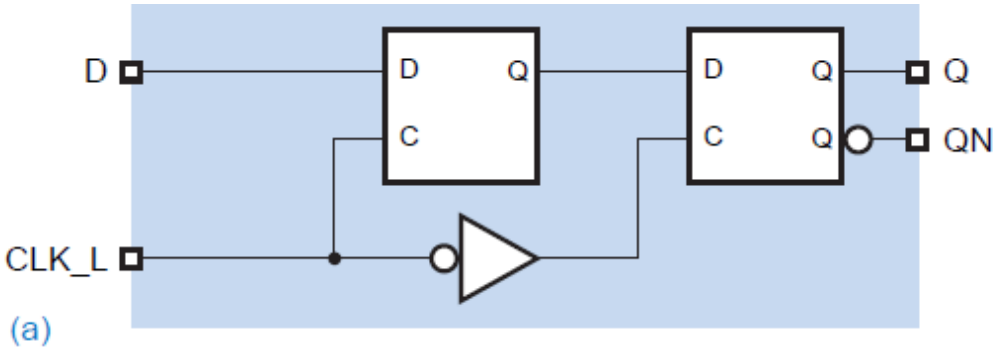
- **Different delays** may be specified for **LOW-to-HIGH** and **HIGH-to-LOW** output changes.

Edge-Triggered D Flip-Flop

- Like a D latch, edge-triggered D flip-flop has a **setup and hold time window** during which D inputs must not change.
- This window occurs around triggering edge of clock.
- If **setup and hold times are not met**, flip-flop output will usually go to a **stable**, though **unpredictable, 0 or 1 state**.
- In some cases, output will **oscillate** or go to a **metastable** state halfway between 0 and 1.
- If flip-flop goes into **metastable state**, it will **return to a stable** state on its own only after a **probabilistic delay**.
- It can also be **forced into a stable state** by **applying another triggering clock edge** with a D input that **meets setup- and hold-time** requirements.

Edge-Triggered D Flip-Flop

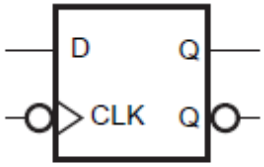
- A **negative-edge-triggered D flip-flop** simply inverts clock input, so that all action takes place on falling edge of CLK_L; by convention a falling-edge trigger is considered to be active low:



(a)

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

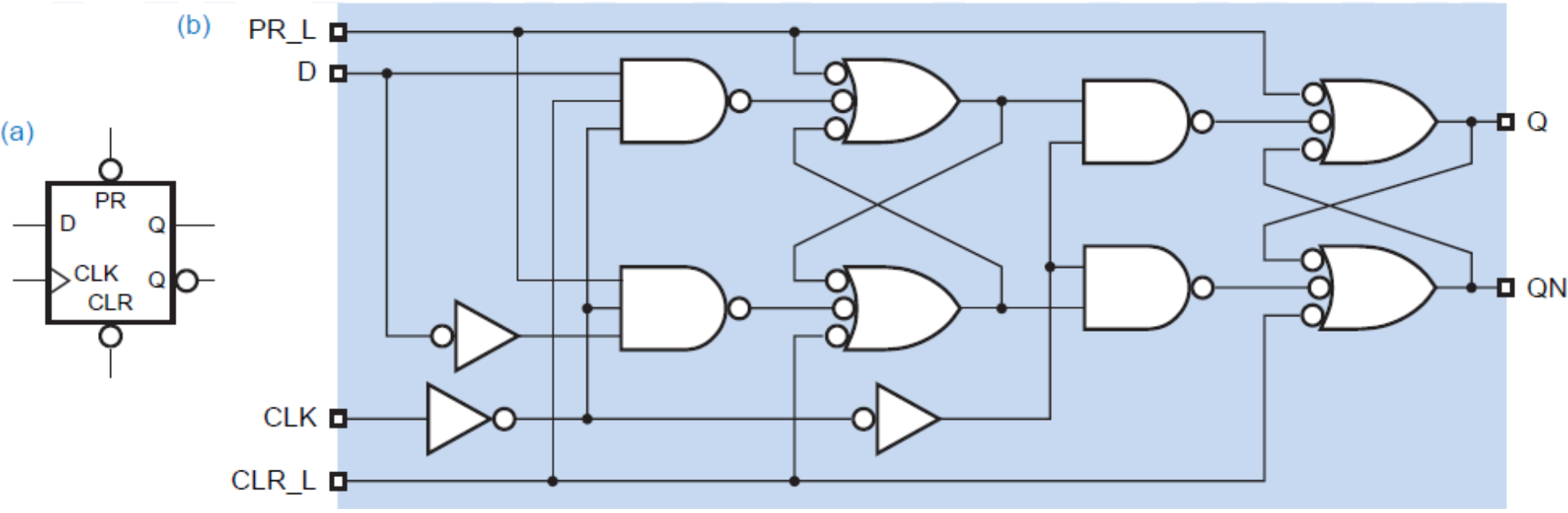
(b)



(c)

Edge-Triggered D Flip-Flop

- Some D flip-flops have **asynchronous inputs** that may be used to force flip-flop to a particular state independent of CLK and D inputs.
- These inputs, typically labeled **PR** (**preset**) and **CLR** (**clear**), behave like set and reset inputs on an S-R latch.
- Positive-edge-triggered D flip-flop with preset and clear:

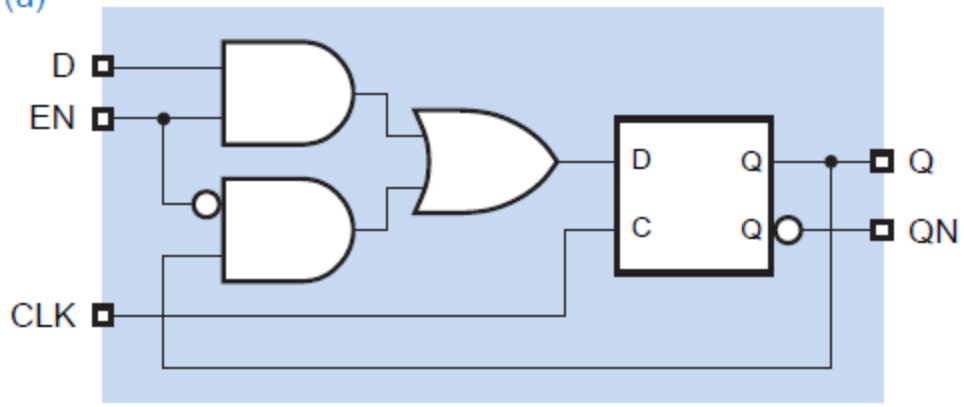


- Asynchronous inputs are usually used for **initialization** and **testing purposes**, to force a sequential circuit into a known starting state.

Edge-Triggered D Flip-Flop with Enable

- A commonly desired function in D flip-flops is **ability to hold last value stored**, rather than load a new value, at clock edge.
- This is accomplished by adding an ***enable*** input.
- Positive-edge-triggered D flip-flop with enable:

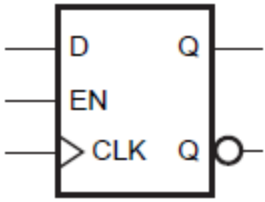
(a)



(b)

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

(c)



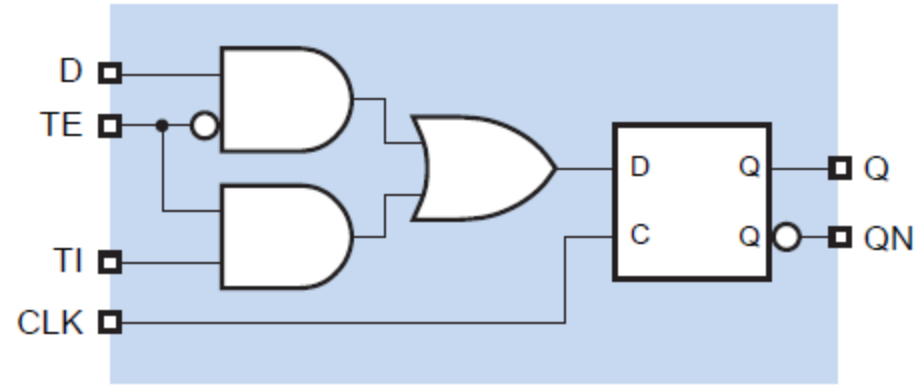
- A **2-input multiplexer** controls value applied to internal flip-flop's D input.

Scan Flip-Flop

- An important flip-flop function for **ASIC testing** is **scan capability**.
- Idea is to be able to **drive** flip-flop's **D input with an alternate source of data** during device testing.
- When all of flip-flops are put into testing mode, a **test pattern** can be "**scanned in**" to ASIC using flip-flops' alternate data inputs.
- After test pattern is loaded, flip-flops are put **back into "normal" mode**, and all of flip-flops are clocked normally.
- After one or more clock ticks, flip-flops are put back into test mode, and **test results are "scanned out"**.

Scan Flip-Flop

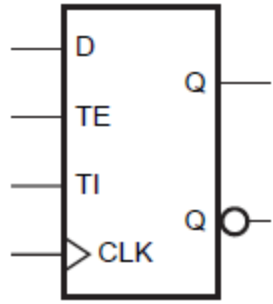
- Positive-edge-triggered D flip-flop with scan:



(a)

TE	TI	D	CLK	Q	QN
0	x	0	↑	0	1
0	x	1	↑	1	0
1	0	x	↑	0	1
1	1	x	↑	1	0
x	x	x	0	last Q	last QN
x	x	x	1	last Q	last QN

(b)



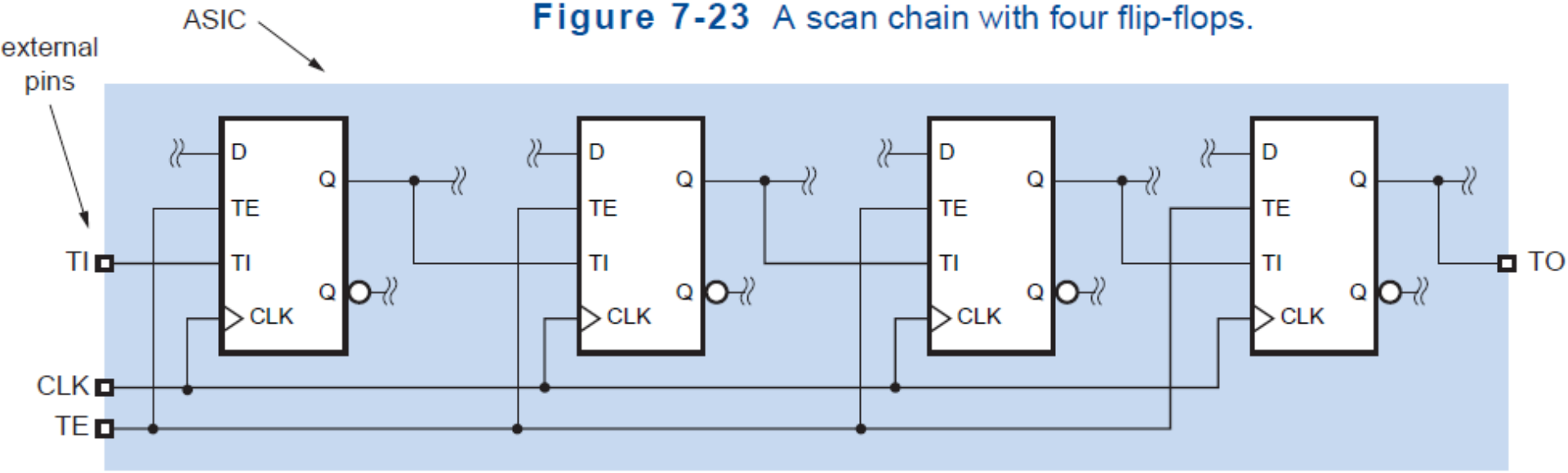
(c)

- It is a D flip-flop with a **2-input multiplexer** on the D input.
- When TE (test enable) input is negated, circuit behaves like an ordinary D flip-flop. When TE is asserted, it takes its data from TI (test input) instead of from D.

Scan Flip-Flop

- Extra inputs are used to connect all of an ASIC's flip-flops in a **scan chain** for testing purposes.

Figure 7-23 A scan chain with four flip-flops.



- TE inputs of all flip-flops are connected to a **global TE** input, while each flip-flop's Q output is connected to another's TI input in serial fashion.
- **TI, TE, and TO** (test output) connections are strictly for **testing purposes**.
- To test circuit, global TE input is asserted while *n* clock ticks occur and ***n* test-vector bits** are applied to global TI input and are thereby scanned (shifted) into *n* flip-flops (*n* = 4 in figure above).

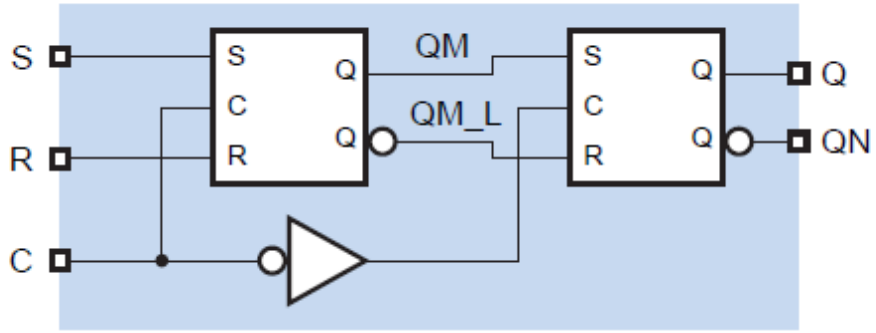
Scan Flip-Flop

- Then TE is negated, and circuit is allowed to run for **one or more additional clock ticks**.
- New state of circuit, represented by new values in *n* flip-flops, can be observed (**scanned out)** at TO by asserting TE while ***n* more clock ticks** occur.
- To make testing process more efficient, another test vector can be scanned in while previous result is being scanned out.
- Scan capability can also be added to other flip-flop types.

Master/Slave S-R Flip-Flop

- S-R latches are useful in "**control**" applications, where we may have independent conditions for setting and resetting a control bit.
- If control bit is supposed to be changed only at **certain times with respect to a clock signal** (certain edge), then we need an S-R flip-flop.
- If we substitute S-R latches for D latches in a negative-edge-triggered D flip-flop, we get a master/slave S-R flip-flop:

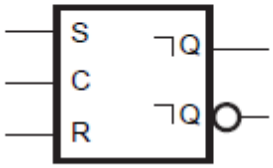
(a)



(b)

S	R	C	Q	QN
x	x	0	last Q	last QN
0	0	⊓	last Q	last QN
0	1	⊓	0	1
1	0	⊓	1	0
1	1	⊓	undef.	undef.

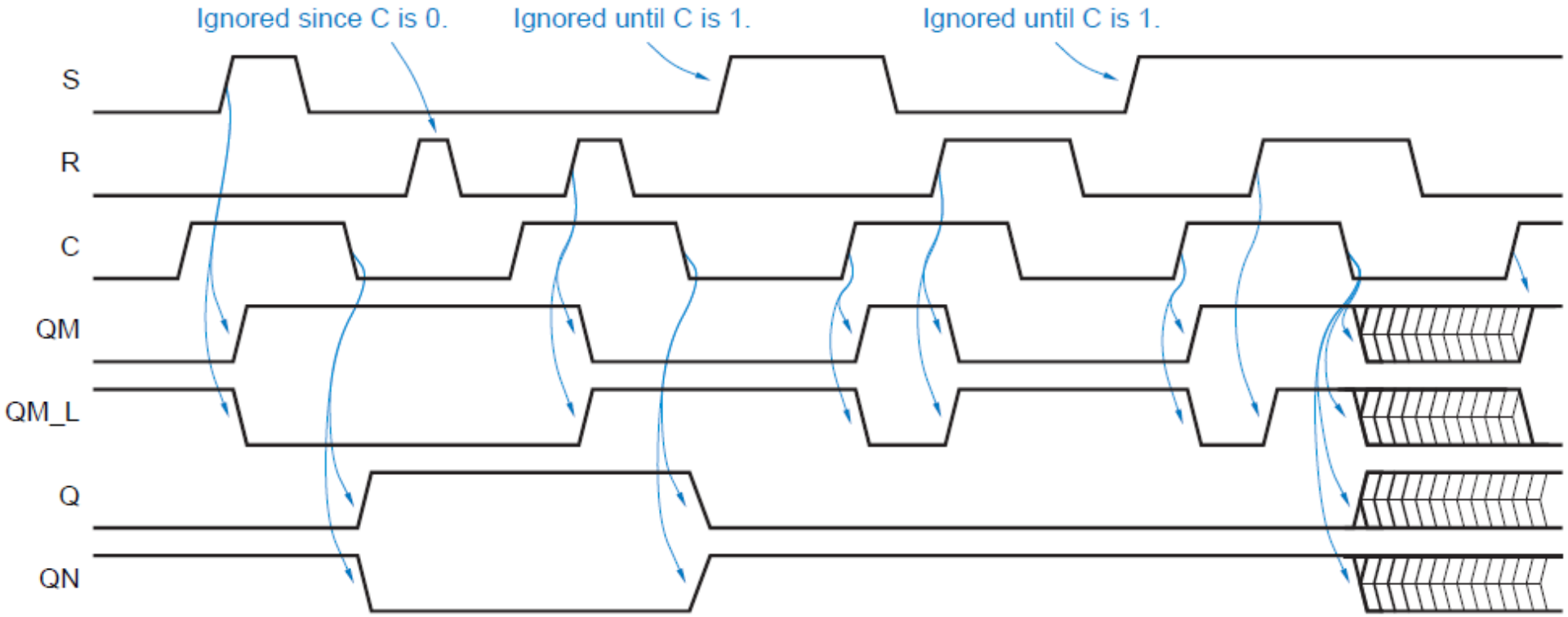
(c)



- S-R flip-flop changes its outputs only at falling edge of a control signal C.
- However, new output value depends on input values not just at falling edge, but during **entire interval in which C is 1 prior to falling edge**.

Master/Slave S-R Flip-Flop

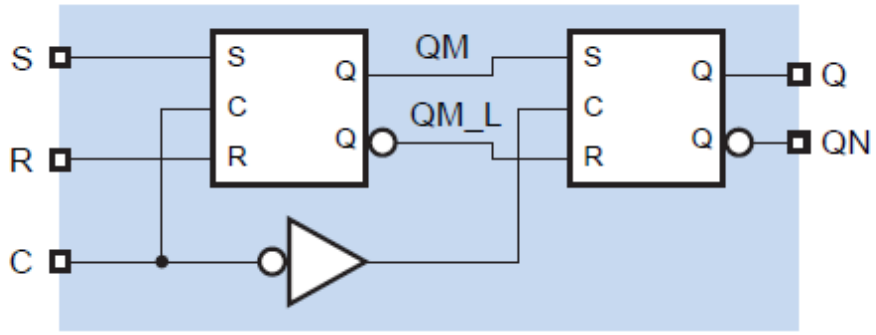
- Internal and functional behavior of a master/slave S-R flip-flop:



- A short pulse on S any time during the interval in which C is 1, can set master latch; likewise a pulse on R can reset it.
- Value transferred to flip-flop output on falling edge of C depends on whether master latch was last set or cleared while C was 1.

Master/Slave S-R Flip-Flop

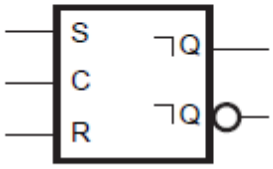
(a)



(b)

S	R	C	Q	QN
x	x	0	last Q	last QN
0	0	⎍	last Q	last QN
0	1	⎍	0	1
1	0	⎍	1	0
1	1	⎍	undef.	undef.

(c)

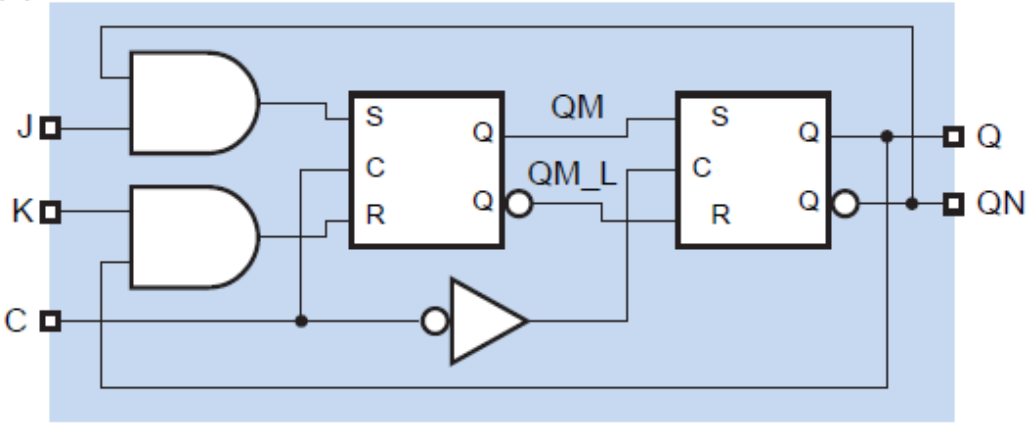


- Logic symbol for master/slave S-R flip-flop (figure (c) above) **does not use a dynamic-input indicator**, because flip-flop is not truly edge-triggered.
- It is more like a latch that follows its input during entire interval that C is 1, but that changes its output to reflect **final latched value** only when C goes to 0.
- In symbol, a **postponed-output indicator** indicates that output signal does not change until enable input C is negated.
- Flip-flops with this kind of behavior are sometimes called **pulse-triggered flip-flops**.
- Operation of master/slave S-R flip-flop is **unpredictable** (and may even become **metastable**) if both S and R are asserted at falling edge of C.

Master/Slave J-K Flip-Flop

- **Problem** of what to do **when S and R are asserted simultaneously** is solved in a master/slave J-K flip-flop.
- J and K inputs are analogous to S and R. Master/slave J-K flip-flop:

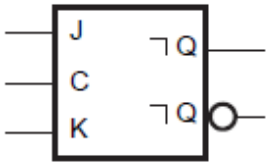
(a)



(b)

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

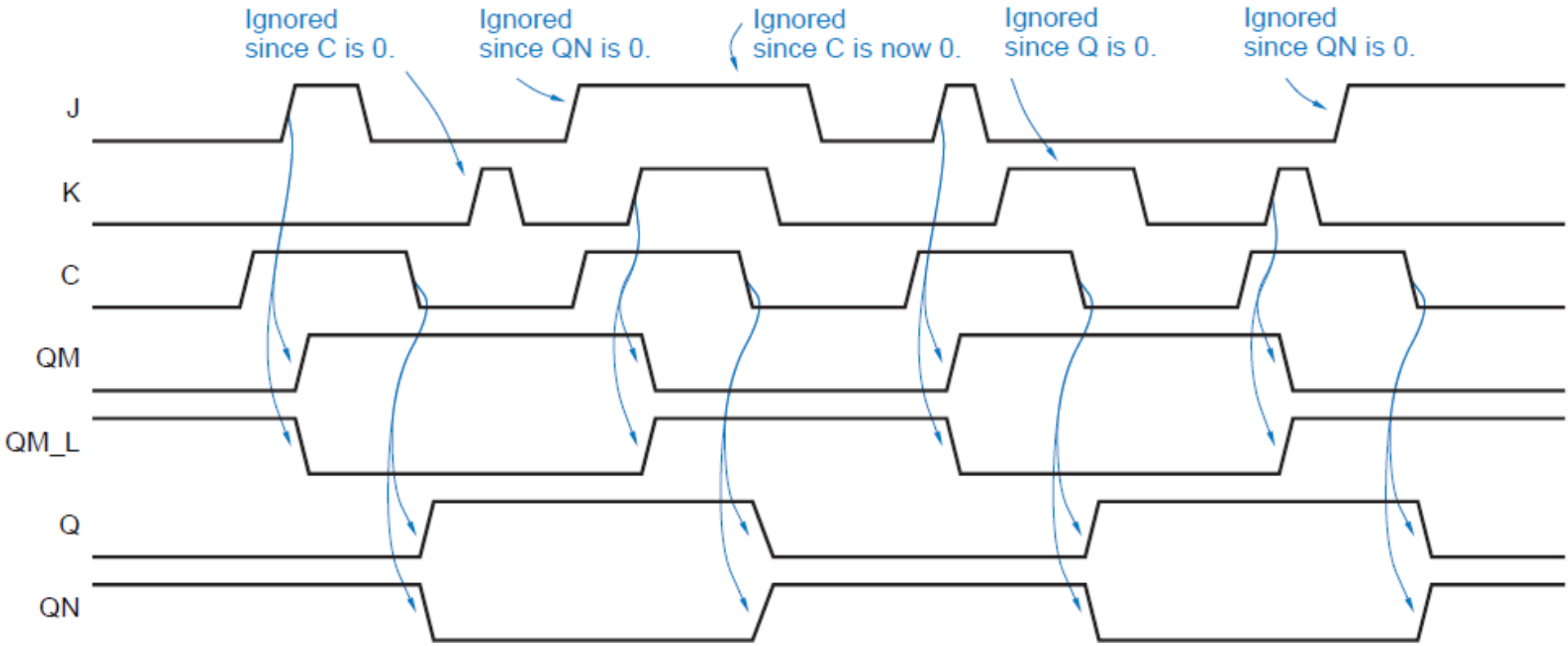
(c)



- Asserting J asserts master's S input only if flip-flop's QN output is currently 1 (i.e., Q is 0), and asserting K asserts master's R input only if Q is currently 1.
- Thus if J and K are asserted simultaneously, flip-flop goes to opposite of its current state.

Master/Slave J-K Flip-Flop

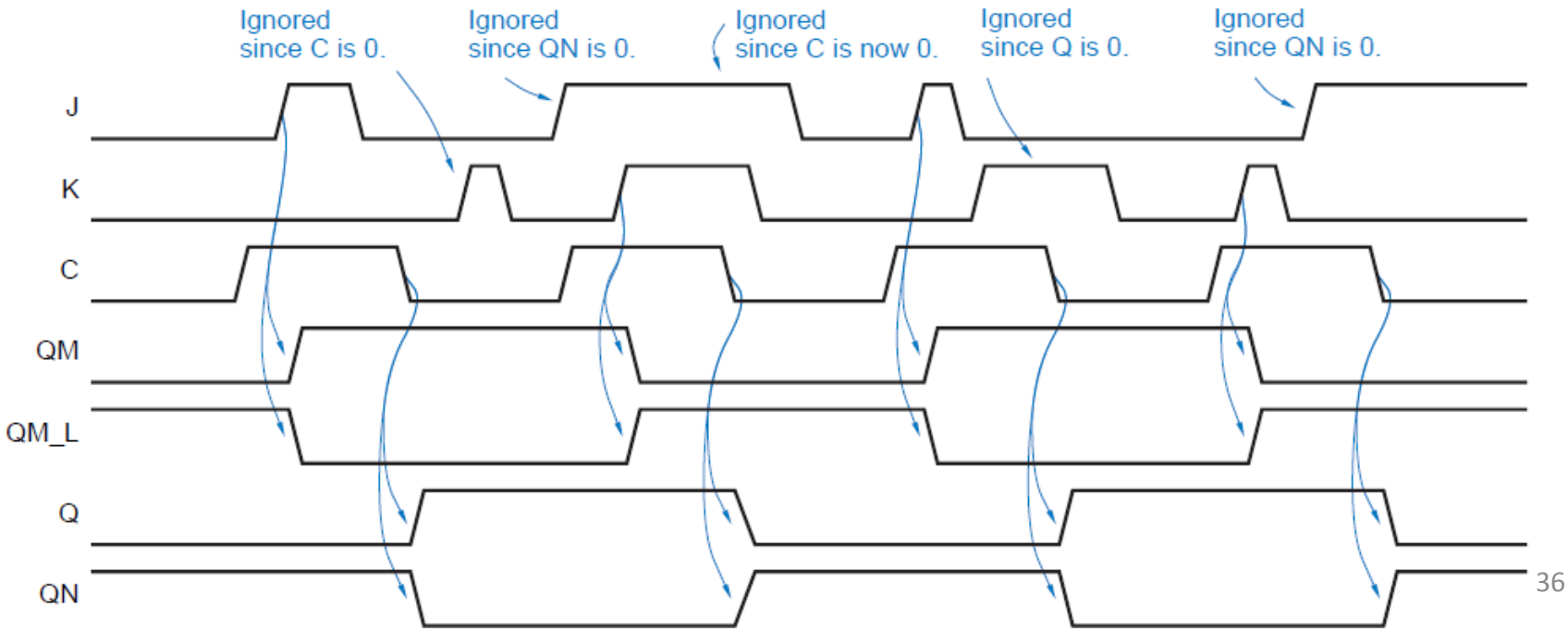
- Internal and functional behavior of a master/slave J-K flip-flop:



- J and K inputs need not be asserted at end of triggering pulse for flip-flop output to change at that time.

Master/Slave J-K Flip-Flop

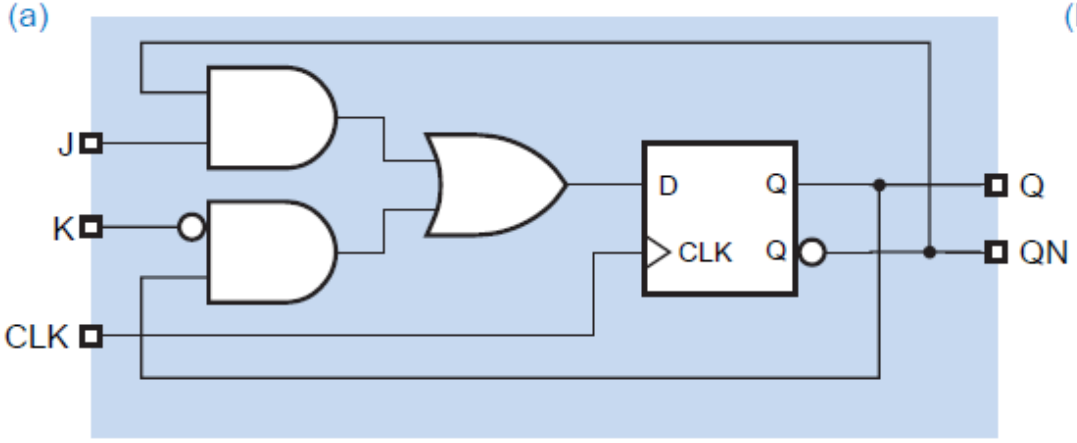
- Because of gating on master latch's S and R inputs, it is possible for flip-flop **output to change to 1** even though **K and not J is asserted** at end of triggering pulse. This behavior, known as ***1s catching***, is illustrated in second-to-last triggering pulse in figure.
- An analogous behavior known as ***0s catching*** is illustrated in last triggering pulse.
- Because of this behavior, **J and K inputs** of a J-K master/slave flip-flop **must be held valid during entire interval that C is 1**.



Edge-Triggered J-K Flip-Flop

- **Problem of 1s and 0s catching** is solved in an edge-triggered J-K flip-flop:

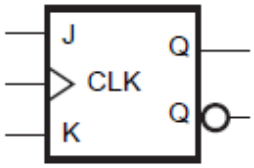
(a)



(b)

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

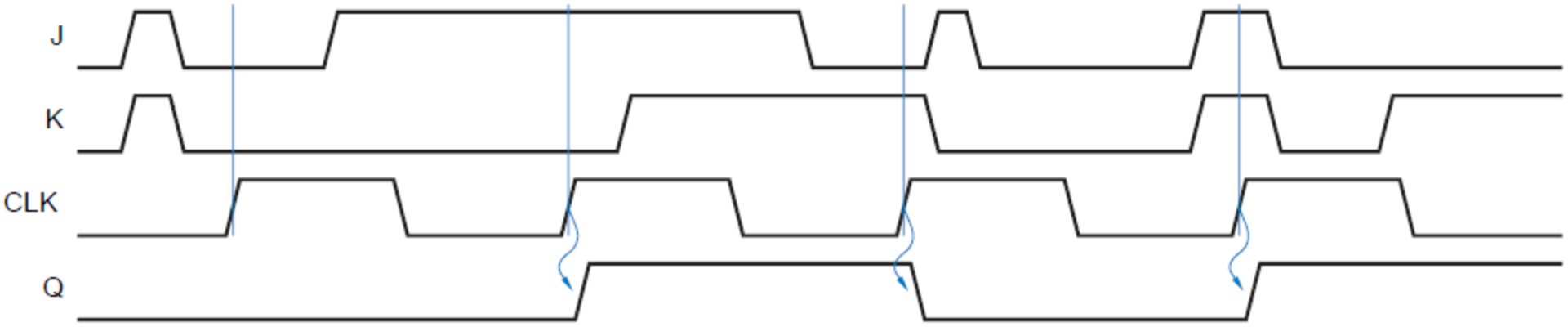
(c)



- Using an edge-triggered D flip-flop internally, edge-triggered J-K flip-flop samples its inputs at rising edge of clock and produces its next output according to "characteristic equation" $Q^* = J.Q' + K'.Q$

Edge-Triggered J-K Flip-Flop

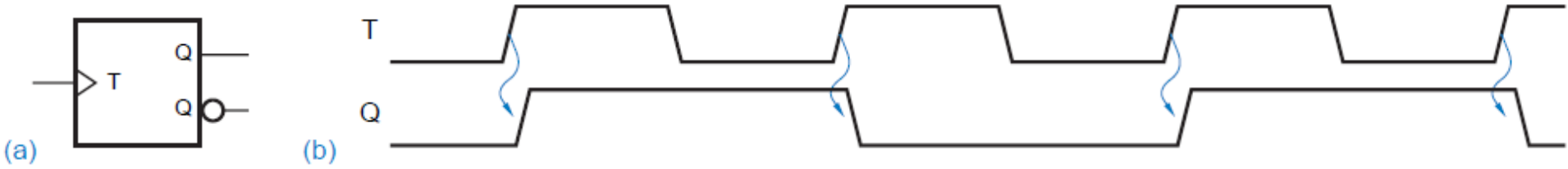
- Functional behavior of a positive-edge-triggered J-K flip-flop:



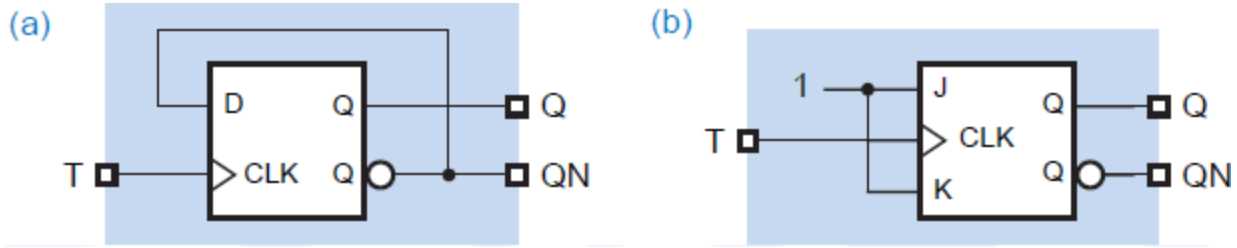
- Like D input of an edge-triggered D flip-flop, J and K inputs of a J-K flip-flop **must meet** published **setup- and hold-time** specifications with respect to triggering clock edge for proper operation.
- Most common application of J-K flip-flops is in clocked synchronous state machines.

T Flip-Flop

- A T (toggle) flip-flop **changes state on every tick** of clock.
- Positive-edge-triggered T flip-flop and its functional behavior:



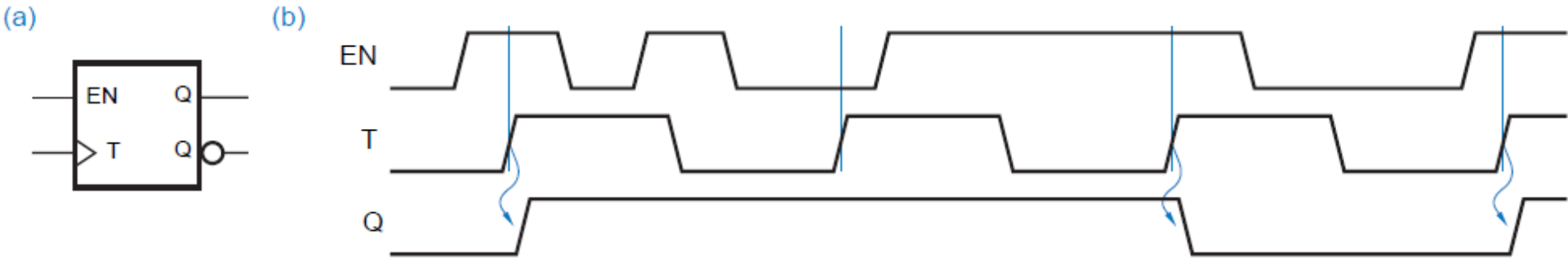
- Signal on flip-flop's Q output has precisely **half frequency** of T input.
- Following figure shows how to obtain a T flip-flop from a D or J-K flip-flop:



- T flip-flops are most often used in **counters** and **frequency dividers**.

T Flip-Flop

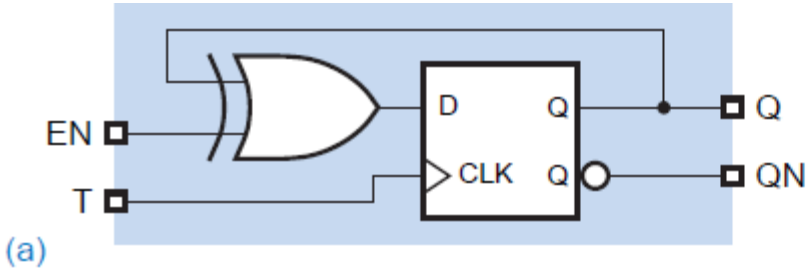
- In many applications of T flip-flops, **flip-flop need not be toggled** on every clock tick. Such applications can use a **T flip-flop with enable**.
- Positive-edge-triggered T flip-flop with enable:



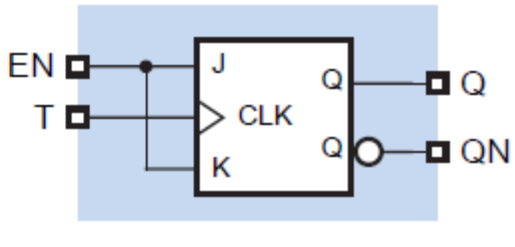
- Flip-flop changes state at triggering edge of clock only if enable signal EN is asserted.
- Like D, J, and K inputs on other edge-triggered flip-flops, **EN input must meet specified setup and hold times** with respect to triggering clock edge.

T Flip-Flop

- Possible circuits for a T flip-flop with enable:



(a)



(b)