



EEE 3101: Digital Logic and Circuits

Sequential Circuit: Latch & Flip-Flops

Course Teacher: Nafiz Ahmed Chisty

Head, Department of EEE

Associate Professor, Department of EEE & CoE

Faculty of Engineering

Room# D0105, D Building

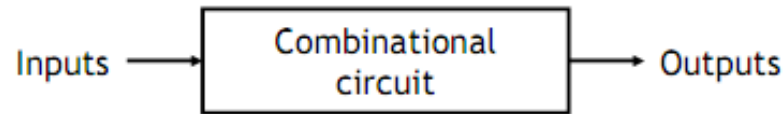
Email: chisty@aiub.edu

Website: <http://engg.aiub.edu/faculties/nafiz>

Website: www.nachisty.com

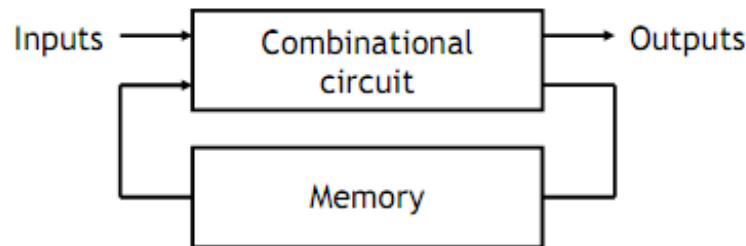


Combinational circuits



- So far we've only worked with **combinational circuits**, where applying the same inputs always produces the same outputs.
 - This corresponds to a mathematical function, where every input has a single, unique output.
 - In programming terminology, combinational circuits are similar to "functional programs" that do not contain variables and assignments.
- Such circuits are comparatively easy to design and analyze.

Sequential circuits



- In contrast, the outputs of a **sequential circuit** depend on not only the inputs, but also the **state**, or the current contents of some memory.
- This makes things more difficult to understand since the same inputs can yield *different* outputs, depending on what's stored in memory.
- The memory contents can also change as the circuit runs, so the *order* in which things occur makes a difference.

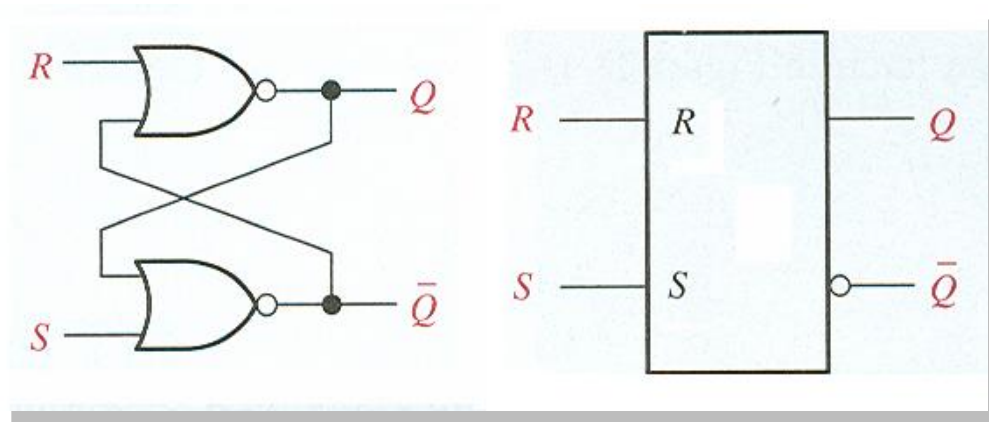


What exactly is memory?

- A memory should support at least three operations.
 1. It should be able to hold a value.
 2. You should be able to *read* the value that is saved.
 3. You should be able to *change* that value.
- We'll start with the simplest case, a one-bit memory.
 1. It should be able to hold a single bit, 0 or 1.
 2. You should be able to read the bit that is saved.
 3. You should be able to change the bit.
 - You can **set** the bit to 1
 - You can **reset** or **clear** the bit to 0.

Latches

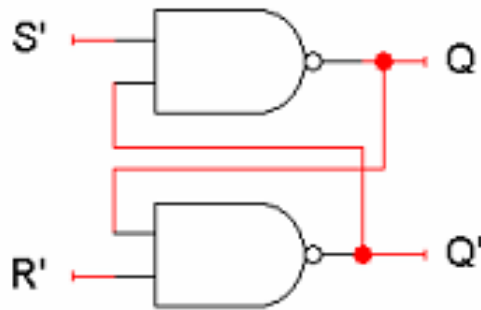
- S-R latch



INPUTS		OUTPUTS		COMMENTS
S	R	Q	\bar{Q}	
0	0	NC	NC	No change. Latch remains in present state.
0	1	0	1	Latch RESET.
1	0	1	0	Latch SET.
1	1	0	0	Invalid condition

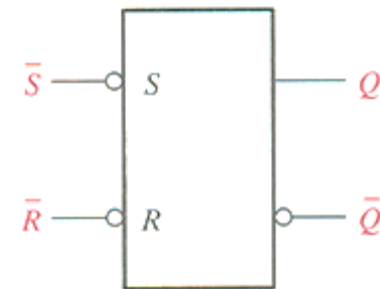
S'R' latch

- There are several other variations of the basic latch.
- You can use NAND instead of NOR gates to get a **S'R' latch**.

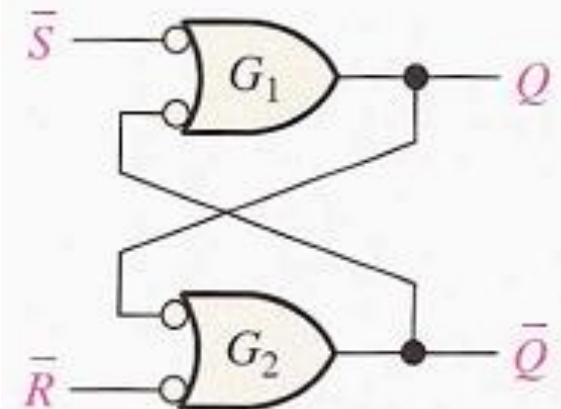
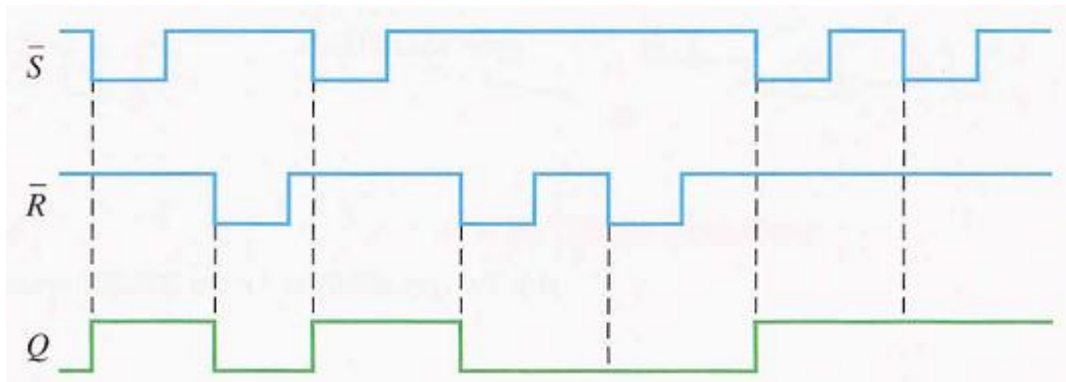


Active-LOW input \bar{S} - \bar{R} latch

S'	R'	Q
1	1	No change
1	0	0 (reset)
0	1	1 (set)
0	0	Avoid!

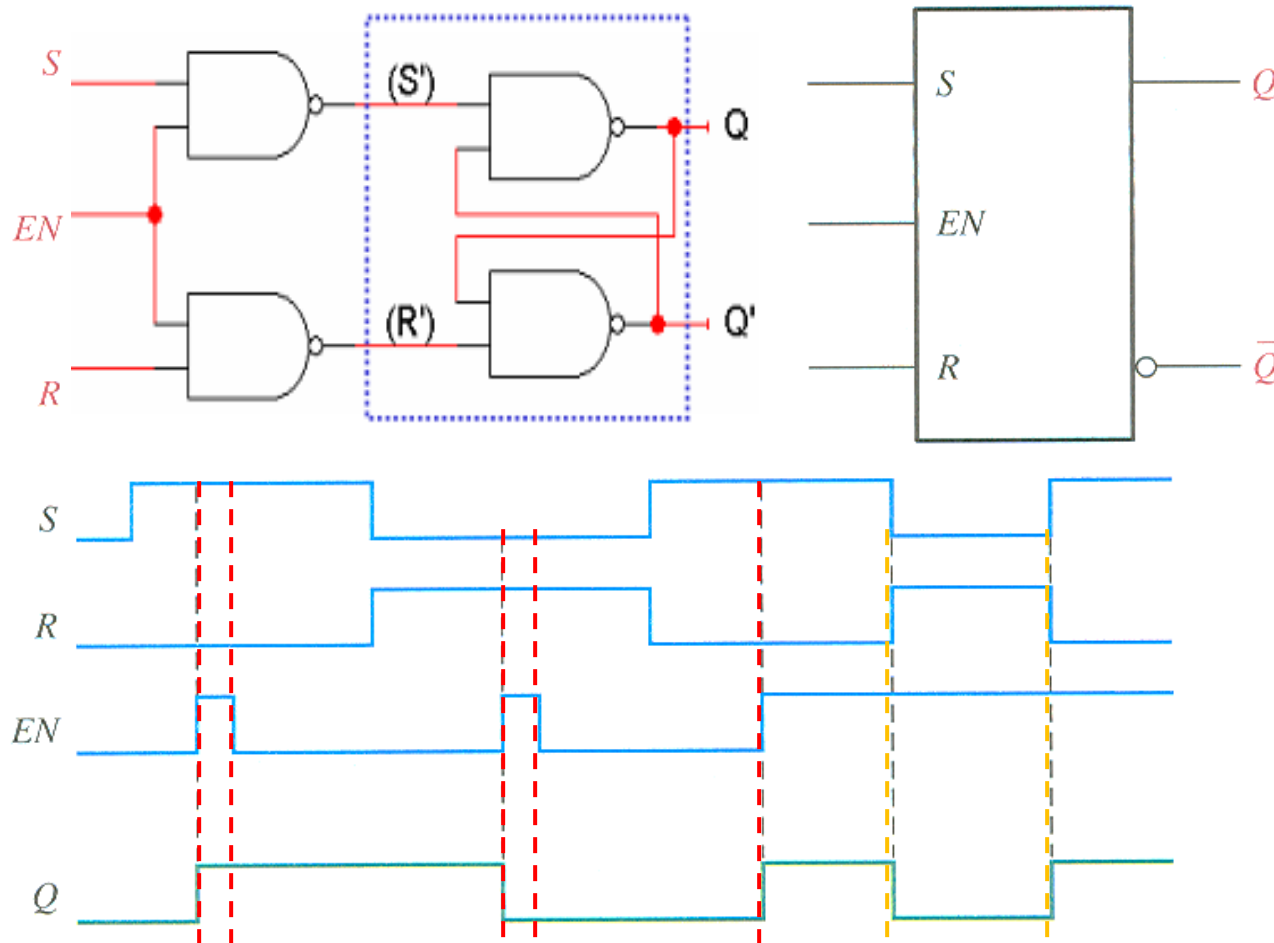


- This is just like an SR latch but with inverted inputs, as you can see from the table.



Negative-OR equivalent of the NAND gate \bar{S} - \bar{R} latch in Figure

Gated S-R latch



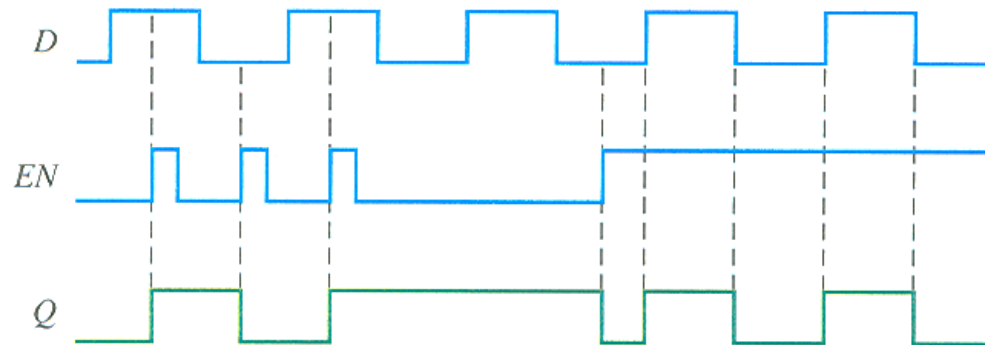
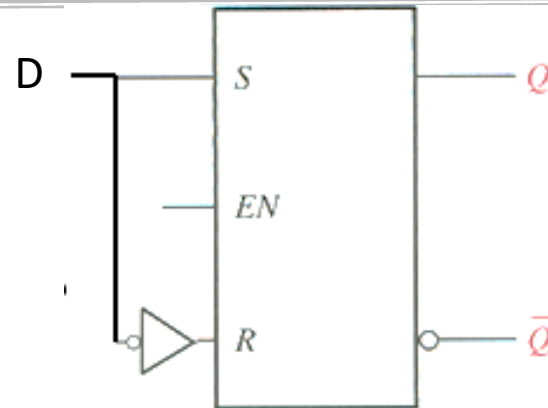
EN	S	R	S'	R'	Q
0	x	x	1	1	No change
1	0	0	1	1	No change
1	0	1	1	0	0 (reset)
1	1	0	0	1	1 (set)
1	1	1	0	0	Evil!

- Notice the hierarchical design!
 - The dotted blue box contains the S'R' latch from the previous slide.
 - The additional NAND gates are simply used to generate appropriate inputs for the S'R' latch.

Gated D latch

EN	D	Q
0	x	No change
1	0	0
1	1	1

Q only changes when E is high



A **D latch** is also based on an S'R' latch. The additional gates generate the S' and R' signals, based on inputs D ("data") and EN ("Enable")

- When $EN=0$, S' and R' are both 1, so Q does not change.
- When $EN=1$, the latch output Q will equal the input D.
- There are two main advantages of a D latch.
 - No more messing with one input for set and another input for reset!
 - This latch has no "bad" input combinations to avoid. Any of the four possible assignments to EN & D are valid.

Flip-Flops

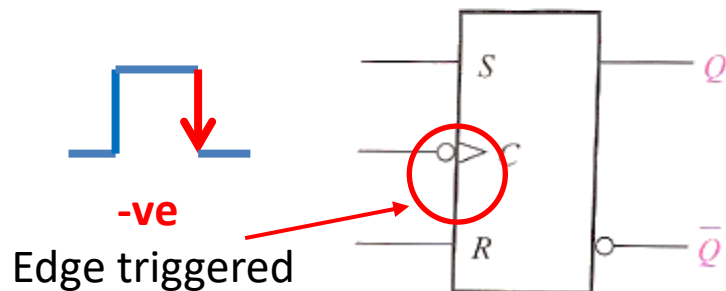
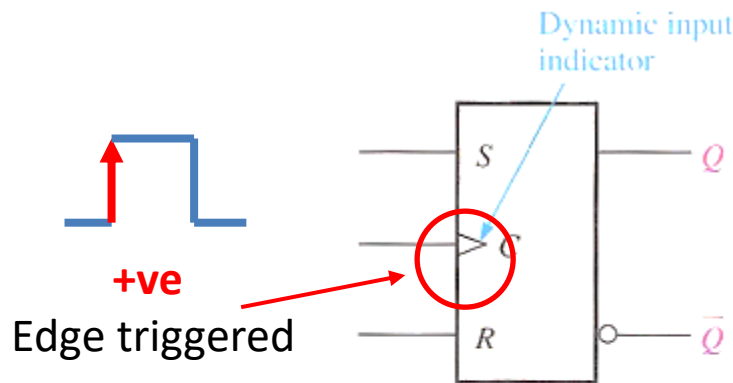
Flip-flops are synchronous bistable devices, also known as bistable multivibrators.

In this case, the term synchronous means that the output changes state only at a specified point on the triggering input called the clock (CLK), which is designated as a control input, C; that is, changes in the output occur in synchronization with the clock.

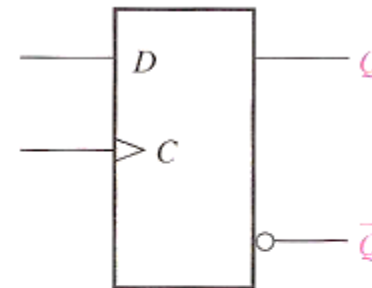
An edge-triggered flip-flop changes state either at the positive edge (rising edge) or at the negative edge (falling edge) of the clock pulse and is sensitive to its inputs only at this transition of the clock.

Edge-Triggered Flip-Flops

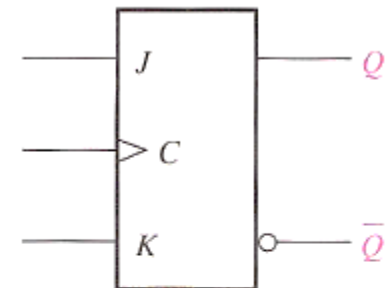
- Edge-triggered S-R flip-flop
- Edge-triggered D flip-flop
- Edge-triggered J-K flip-flop
- Edge-triggered T flip-flop



(a) S-R

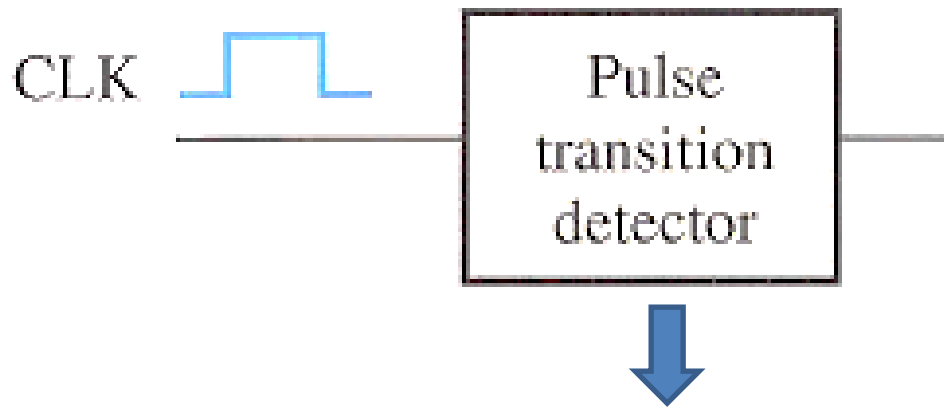


(b) D

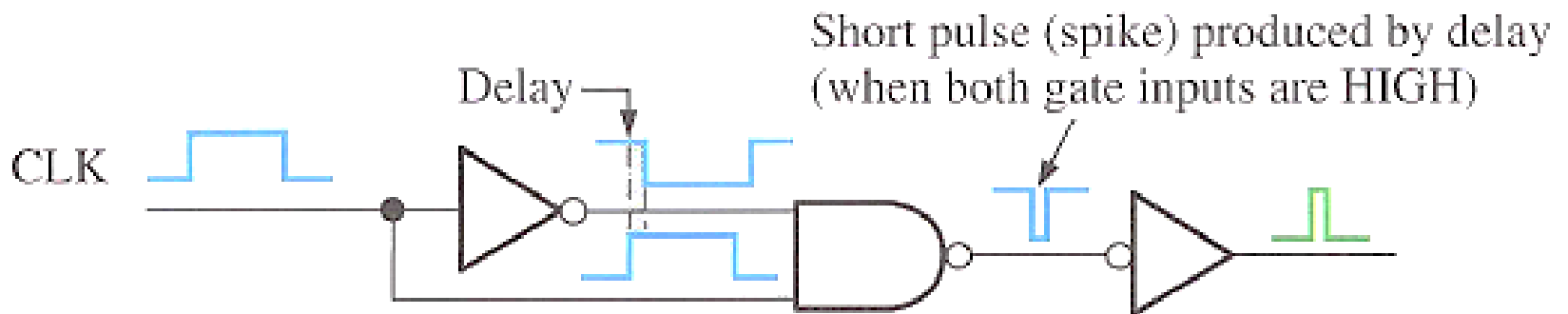


(c) J-K

A Method of Edge-Triggering

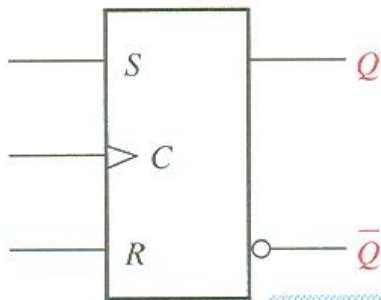


A type of pulse transition detector



Edge-triggered S-R flip-flop

The S and R inputs of the **S-R flip-flop** are called **synchronous** inputs because data on these inputs are transferred to the flip-flop's output only on the triggering edge of the clock pulse.



INPUTS			OUTPUTS		COMMENTS
S	R	CLK	Q	\bar{Q}	
0	0	X	Q_0	\bar{Q}_0	No change
0	1	\uparrow	0	1	RESET
1	0	\uparrow	1	0	SET
1	1	\uparrow	?	?	Invalid

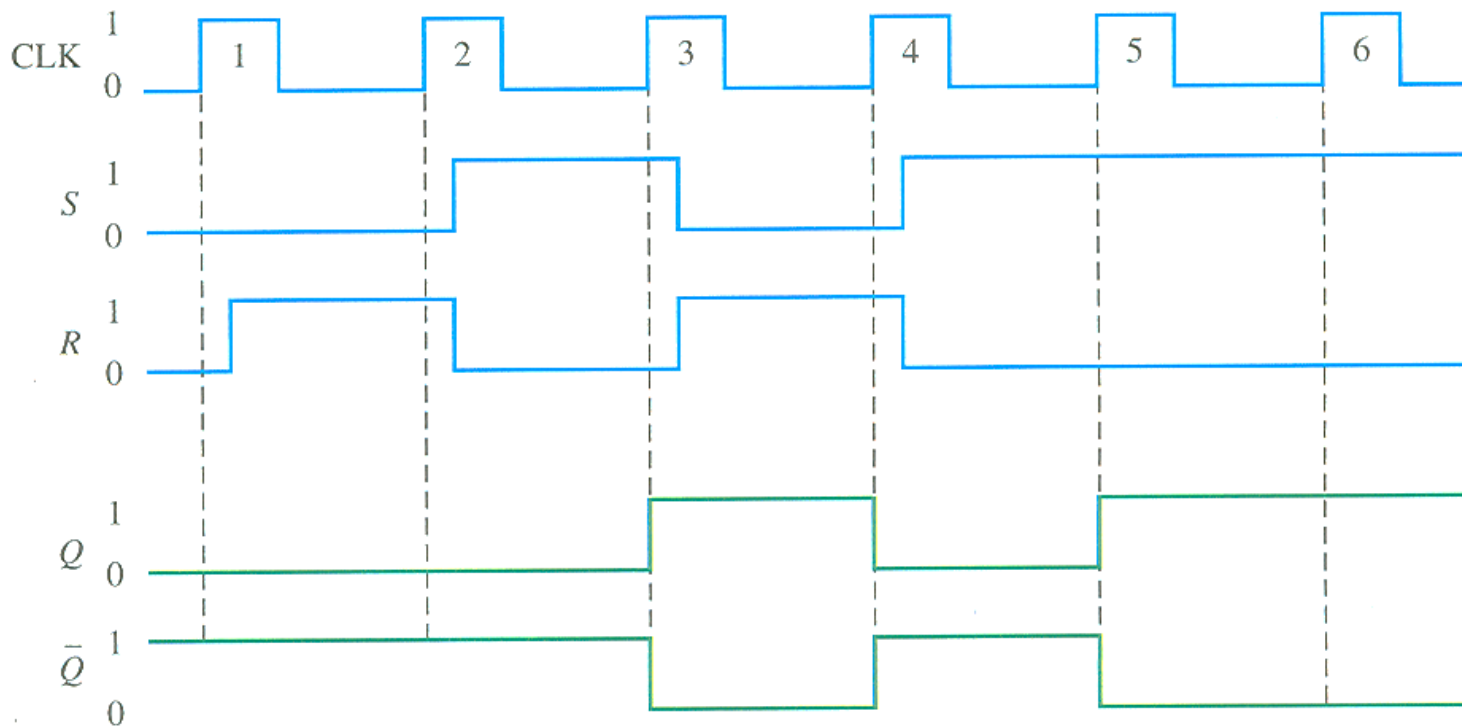
\uparrow = clock transition LOW to HIGH

X = irrelevant ("don't care")

Q_0 = output level prior to clock transition

Edge-triggered S-R flip-flop

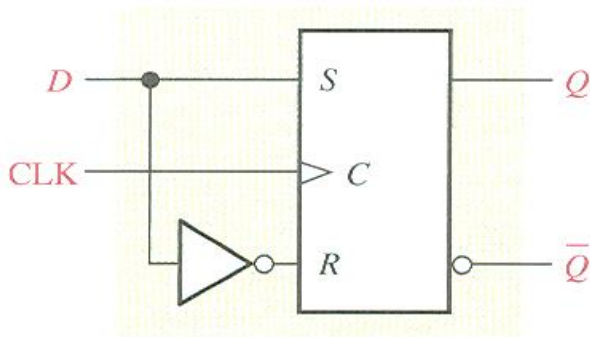
INPUTS			OUTPUTS		COMMENTS
S	R	CLK	Q	\bar{Q}	
0	0	X	Q_0	\bar{Q}_0	No change
0	1	\uparrow	0	1	RESET
1	0	\uparrow	1	0	SET
1	1	\uparrow	?	?	Invalid



Remember, *the flip-flop cannot change state except on the triggering edge of a clock pulse*. The S and R inputs can be changed at any time when the clock input is LOW or HIGH (except for a very short interval around the triggering transition of the clock) without affecting the output.

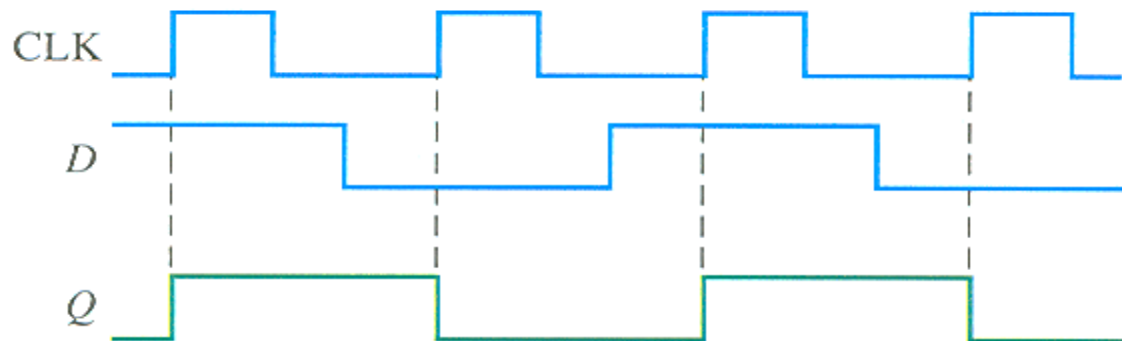
• Edge-triggered D flip-flop

The **D flip-flop** is useful when a single data bit (1 or 0) is to be stored.



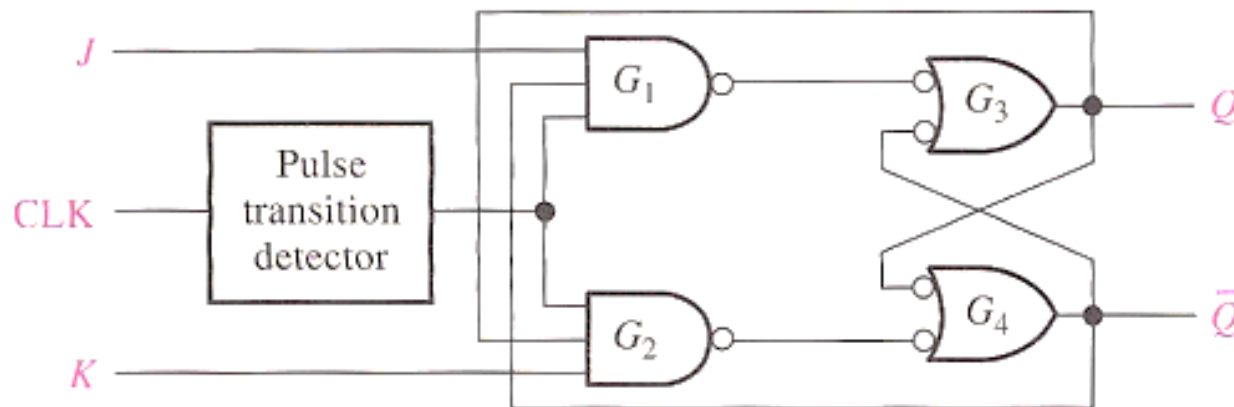
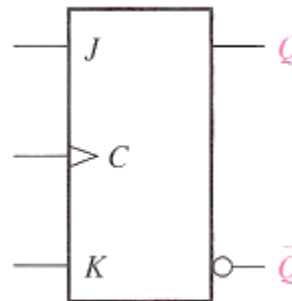
INPUTS		OUTPUTS		COMMENTS
D	CLK	Q	\bar{Q}	
1	↑	1	0	SET (stores a 1)
0	↑	0	1	RESET (stores a 0)

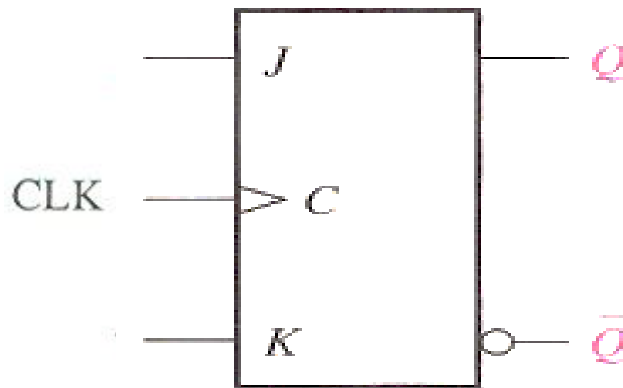
↑ = clock transition LOW to HIGH



• Edge-triggered J-K flip-flop

The **J-K flip-flop** is versatile and is a widely used type of flip-flop. The functioning of the J-K flip-flop is identical to that of the S-R flip-flop in the SET, RESET, and no-change conditions of operation. The difference is that the J-K flip-flop has no invalid state as does the S-R flip-flop.



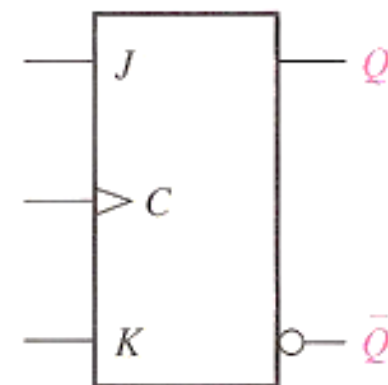
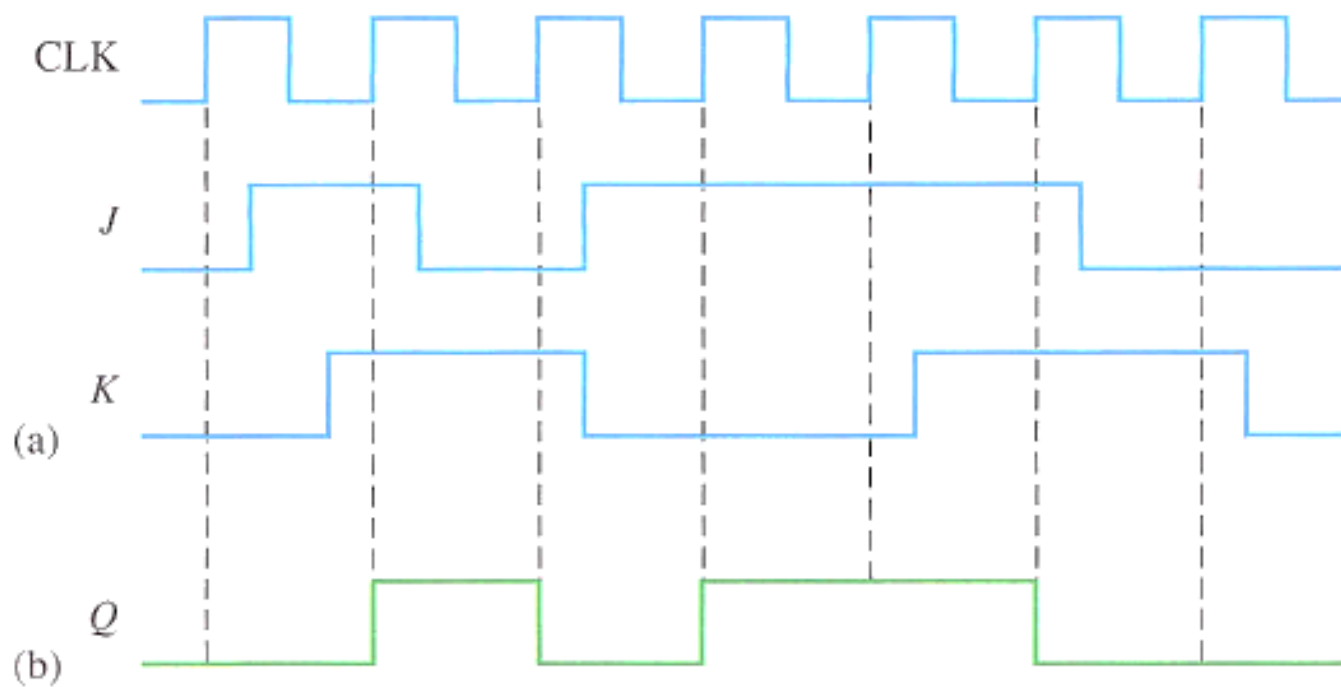


INPUTS			OUTPUTS		COMMENTS
J	K	CLK	Q	\bar{Q}	
0	0	\uparrow	Q_0	\bar{Q}_0	No change
0	1	\uparrow	0	1	RESET
1	0	\uparrow	1	0	SET
1	1	\uparrow	\bar{Q}_0	Q_0	Toggle

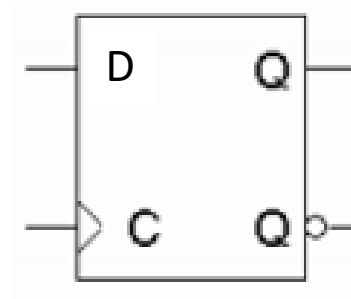
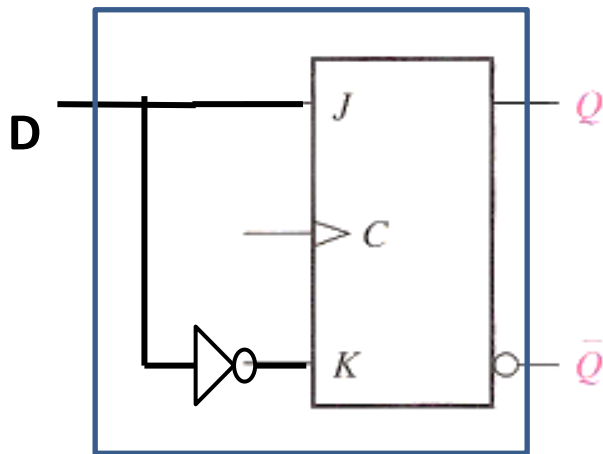
\uparrow = clock transition LOW to HIGH

Q_0 = output level prior to clock transition

INPUTS			OUTPUTS		COMMENTS
J	K	CLK	Q	\bar{Q}	
0	0	\uparrow	Q_0	\bar{Q}_0	No change
0	1	\uparrow	0	1	RESET
1	0	\uparrow	1	0	SET
1	1	\uparrow	\bar{Q}_0	Q_0	Toggle



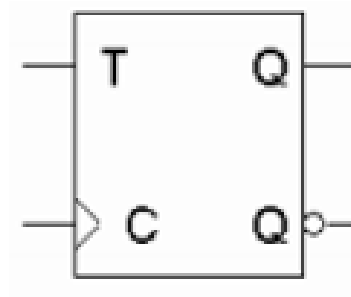
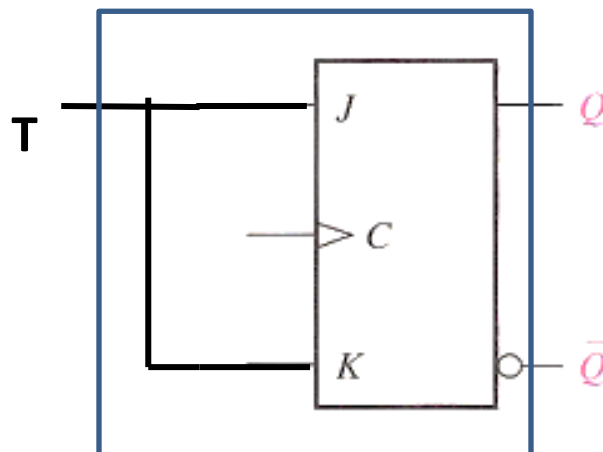
D Flip-flop:



C	D	Q
0	x	No change
1	0	0
1	1	1

T Flip-flop:

A T flip-flop can only maintain or complement its current state.

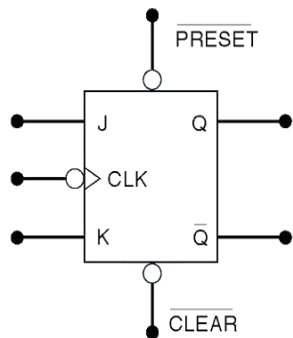


C	T	Q_{next}
0	x	No change
1	0	No change
1	1	Q'_{current}

Asynchronous Preset and Clear Inputs

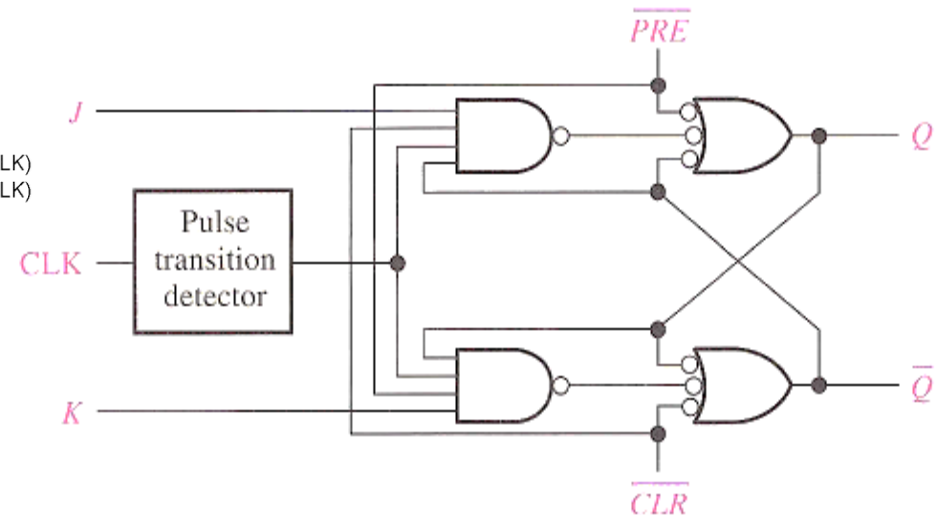
For the flip-flops just discussed, the S - R , D , and J - K inputs are called *synchronous inputs* because data on these inputs are transferred to the flip-flop's output only on the triggering edge of the clock pulse; that is, the data are transferred synchronously with the clock.

Most integrated circuit flip-flops also have **asynchronous** inputs. These are inputs that affect the state of the flip-flop independent of the clock. They are normally labeled **preset** (PRE) and **clear** (CLR), or *direct set* (S_D) and *direct reset* (R_D) by some manufacturers. An active level on the preset input will set the flip-flop, and an active level on the clear input will reset it.



\overline{PRESET}	\overline{CLEAR}	FF response
1	1	Clocked operation*
0	1	$Q = 1$ (regardless of CLK)
1	0	$Q = 0$ (regardless of CLK)
0	0	Not used

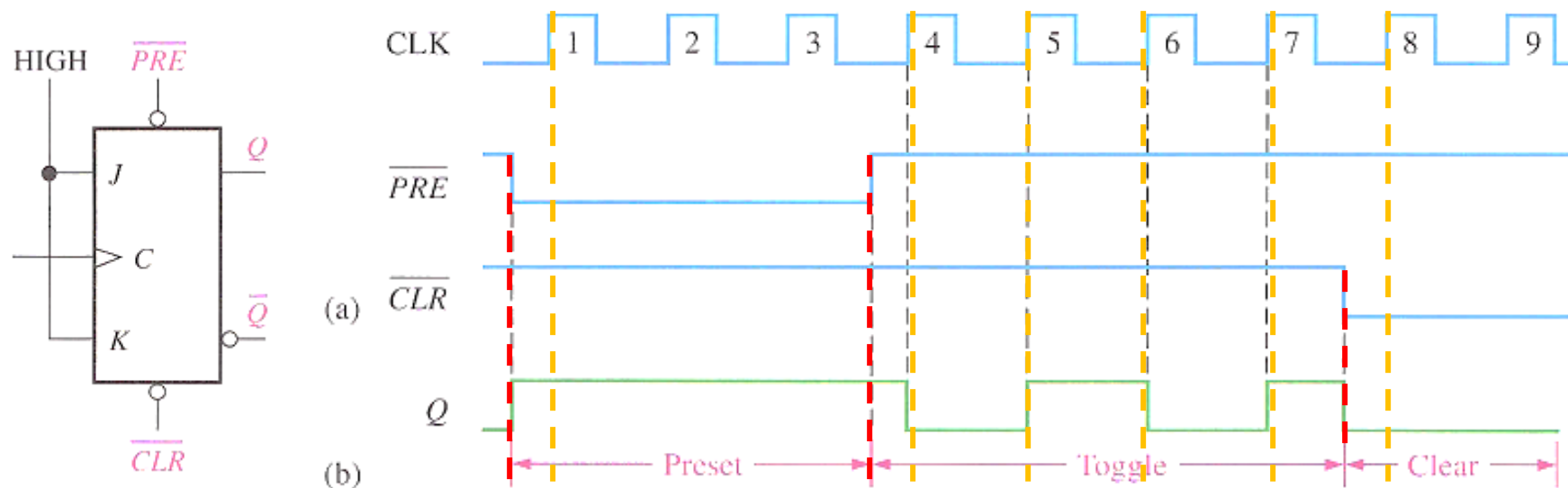
* Q will respond to J , K , and CLK



puts work. As you can see, they are connected so that they override the effect of the synchronous inputs, J , K , and the clock.

EXAMPLE 7-8

For the positive edge-triggered J-K flip-flop with preset and clear inputs in Figure 7-28, determine the Q output for the inputs shown in the timing diagram in part (a) if Q is initially LOW.



1. During clock pulses 1, 2, and 3, the preset (\overline{PRE}) is LOW, keeping the flip-flop SET regardless of the synchronous J and K inputs.
2. For clock pulses 4, 5, 6, and 7, toggle operation occurs because J is HIGH, K is HIGH, and both \overline{PRE} and \overline{CLR} are HIGH.
3. For clock pulses 8 and 9, the clear (\overline{CLR}) input is LOW, keeping the flip-flop RESET regardless of the synchronous inputs.

FLIP-FLOP APPLICATIONS

three general applications of flip-flops are

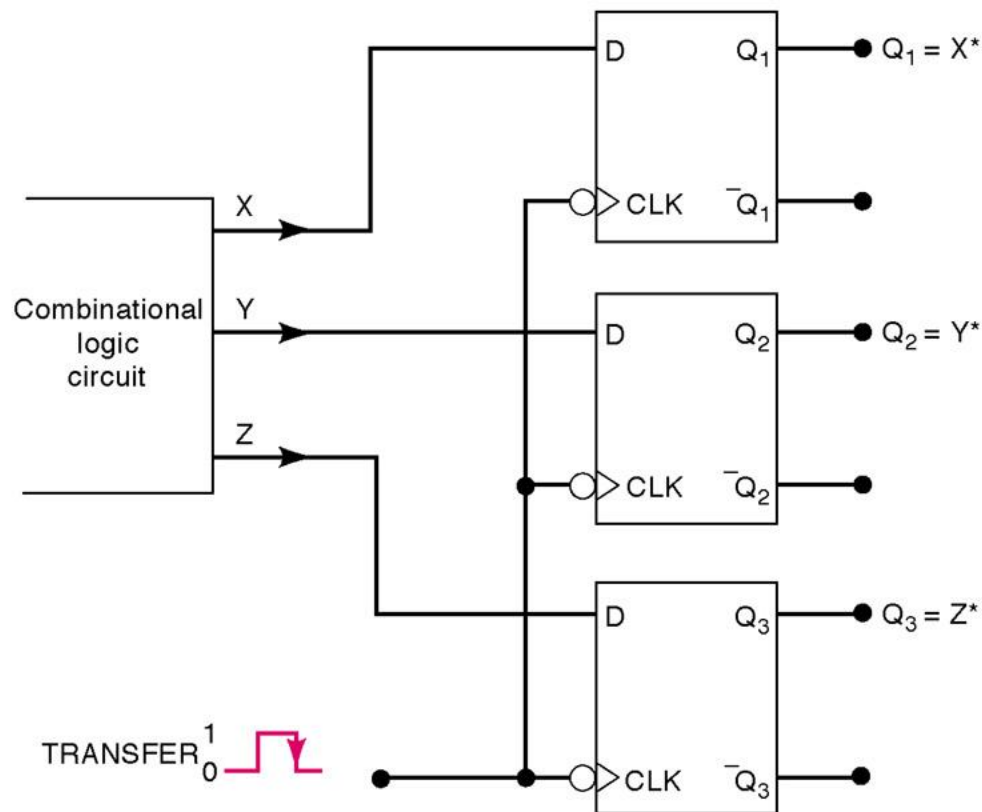
Parallel Data Storage

Frequency Division

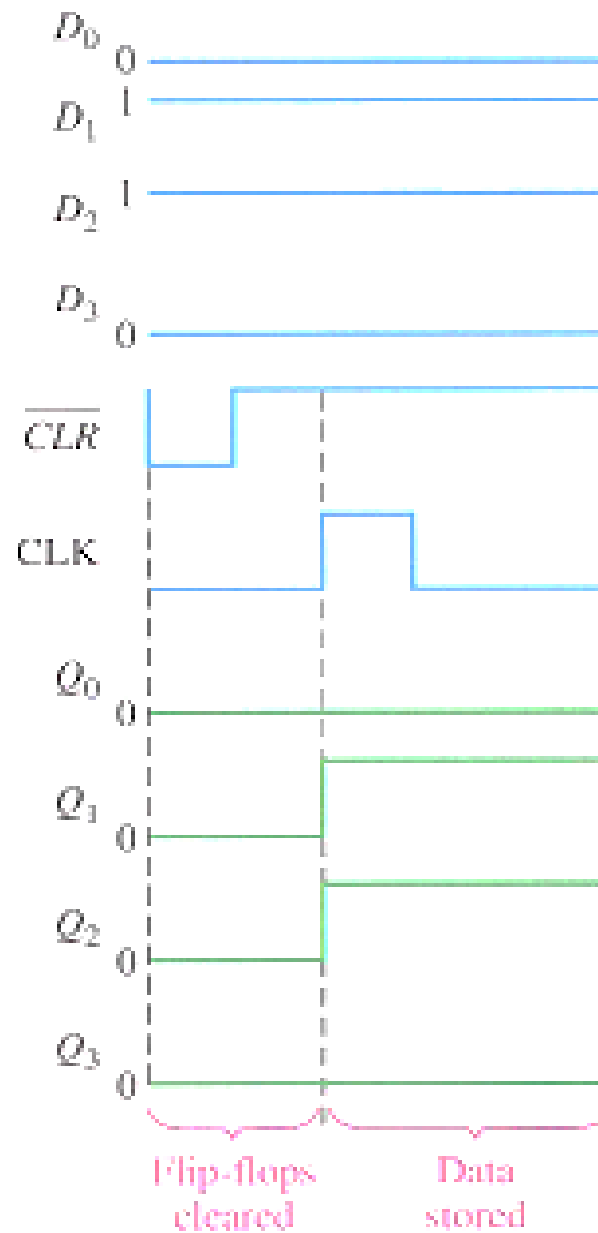
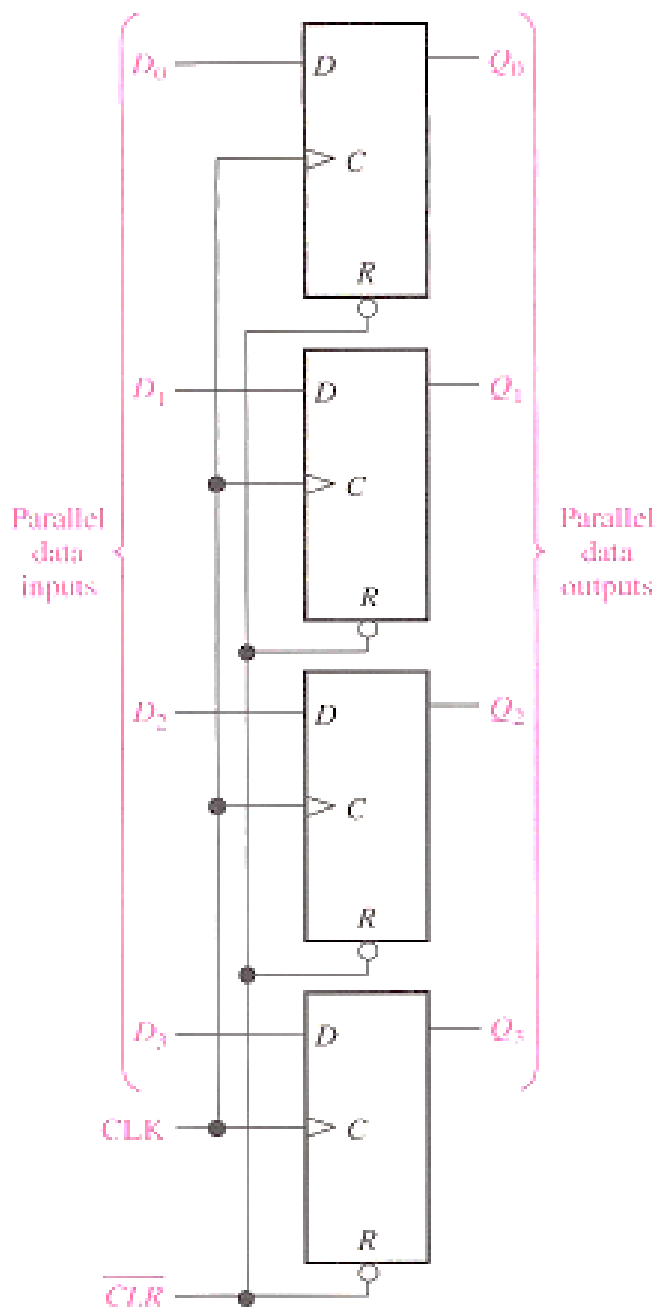
Counting

Parallel Data Storage

- Flip flops store outputs from combinational logic.
- Multiple flops can store a collection of data

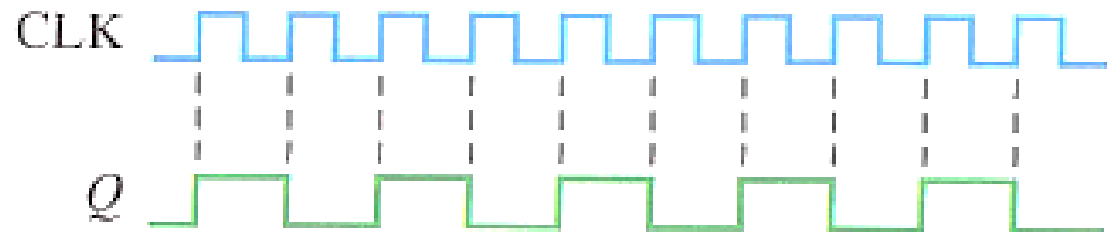
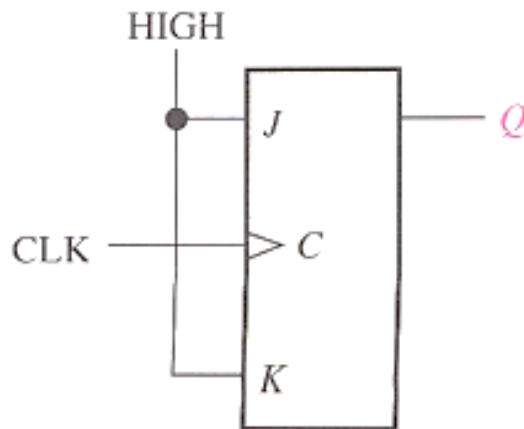


*After occurrence of NGT

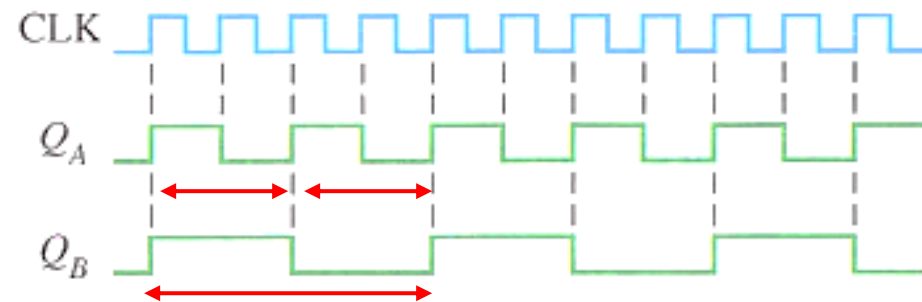
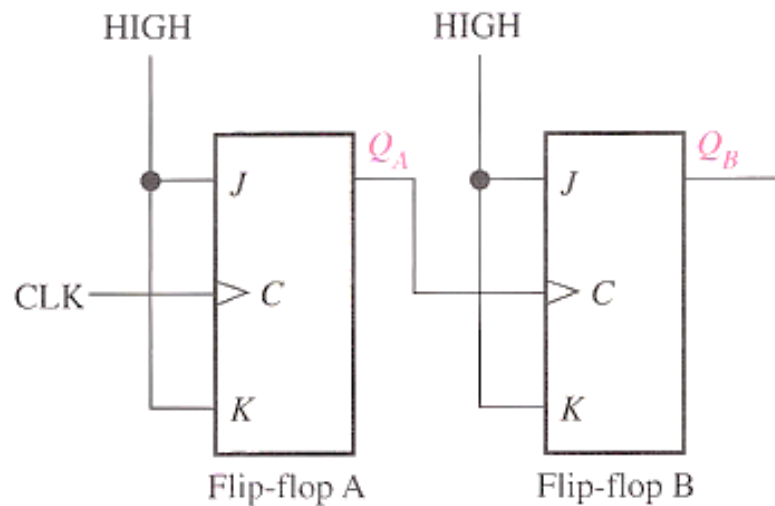


Frequency Division

Another application of a flip-flop is dividing (reducing) the frequency of a periodic waveform. When a pulse waveform is applied to the clock input of a J-K flip-flop that is connected to toggle ($J = K = 1$), the Q output is a square wave with one-half the frequency of the clock input. Thus, a single flip-flop can be applied as a divide-by-2 device, as is illustrated in Figure 7-37. As you can see, the flip-flop changes state on each triggering clock edge (positive edge-triggered in this case). This results in an output that changes at half the frequency of the clock waveform.



Further division of a clock frequency can be achieved by using the output of one flip-flop as the clock input to a second flip-flop, as shown in Figure 7–38. The frequency of the Q_A output is divided by 2 by flip-flop B. The Q_B output is, therefore, one-fourth the frequency of the original clock input. Propagation delay times are not shown on the timing diagrams.

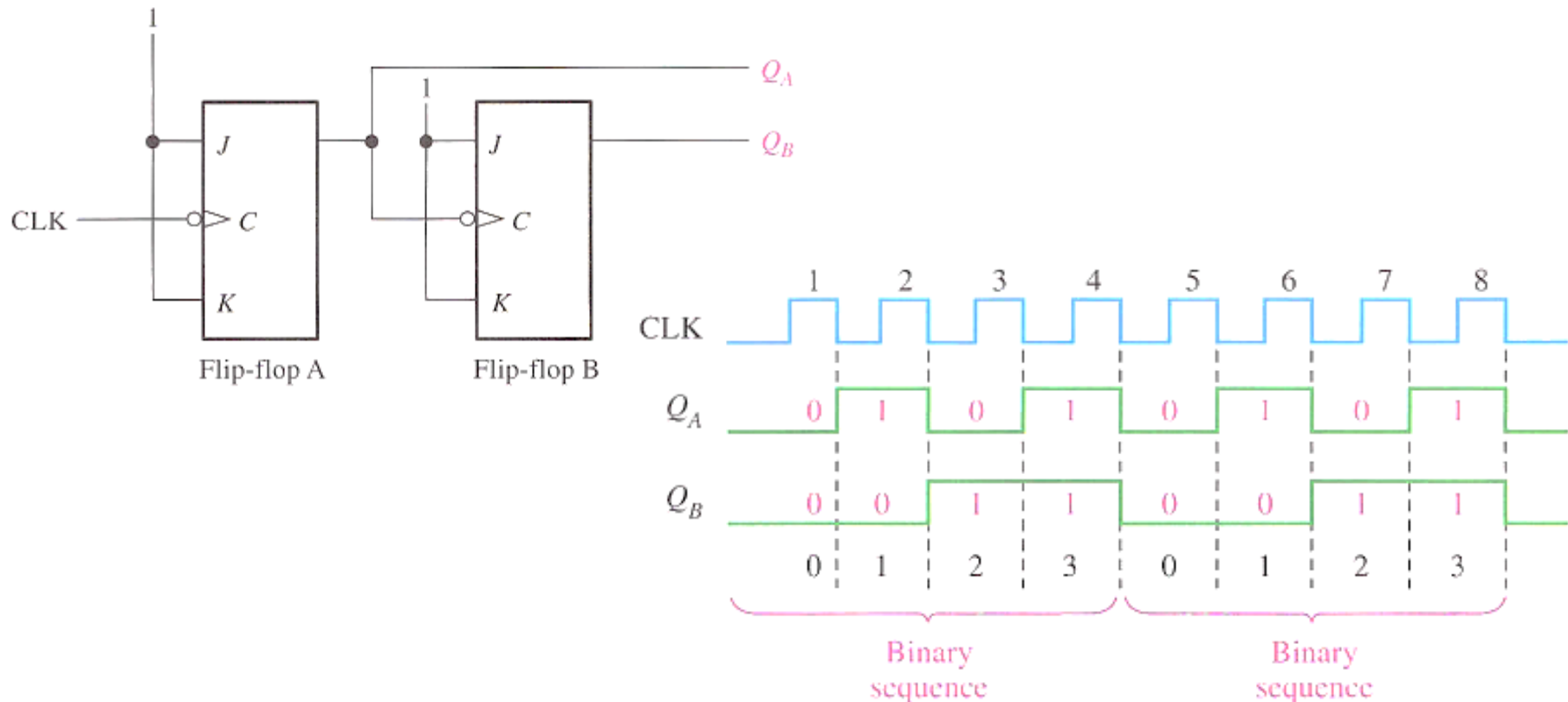


By connecting flip-flops in this way, a frequency division of 2^n is achieved, where n is the number of flip-flops. For example, three flip-flops divide the clock frequency by $2^3 = 8$; four flip-flops divide the clock frequency by $2^4 = 16$; and so on.

Counting

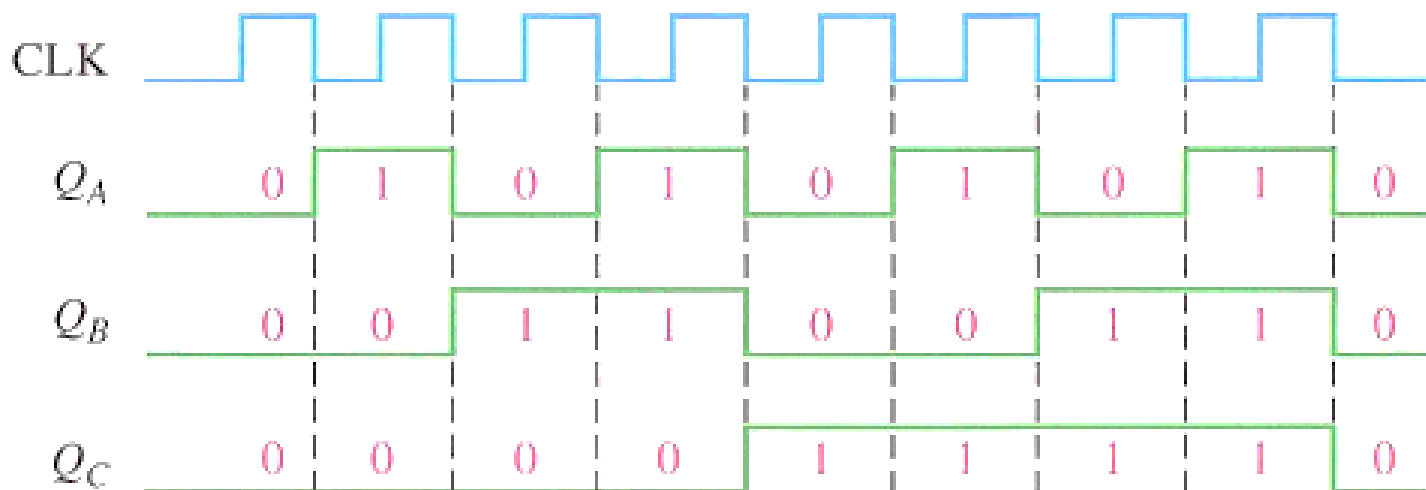
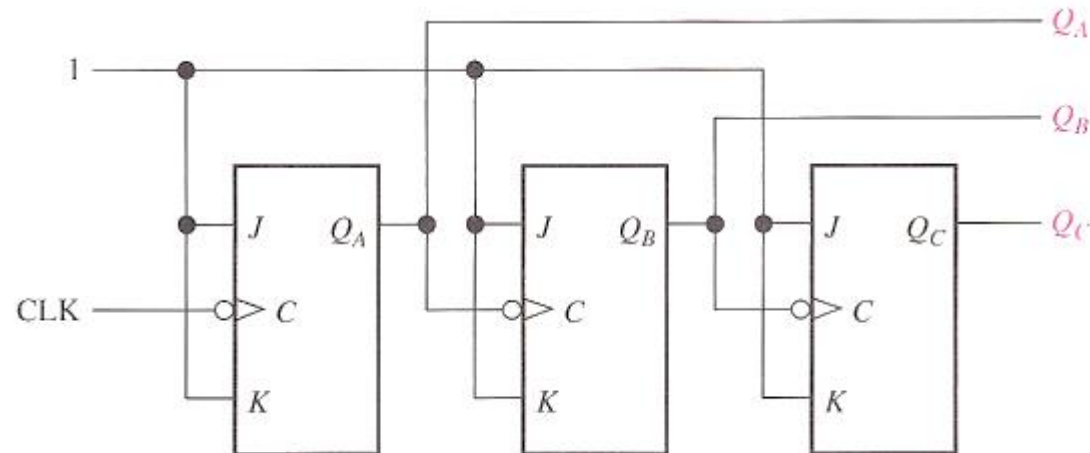
Another important application of flip-flops is in digital counters

Both flip-flops are initially RESET. Flip-flop A toggles on the negative-going transition of each clock pulse. The Q output of flip-flop A clocks flip-flop B, so each time Q_A makes a HIGH-to-LOW transition, flip-flop B toggles. The resulting Q_A and Q_B waveforms are shown in the figure.



EXAMPLE 7-11

Determine the output waveforms in relation to the clock for Q_A , Q_B , and Q_C in the circuit of Figure 7-42 and show the binary sequence represented by these waveforms.





Reference:

- [1] Thomas L. Floyd, "Digital Fundamentals" 11th edition, Prentice Hall.
- [2] M. Morris Mano, "Digital Logic & Computer Design" Prentice Hall.
- [3] Mixed contents from Vahid And Howard.





Thanks

