جامعة الملك سعود
King Saud University

CSC 220: Computer Organization

Unit 7
Sequential Circuits

Prepared by:
Md Saiful Islam, PhD

Department of Computer Science
College of Computer and Information Sciences

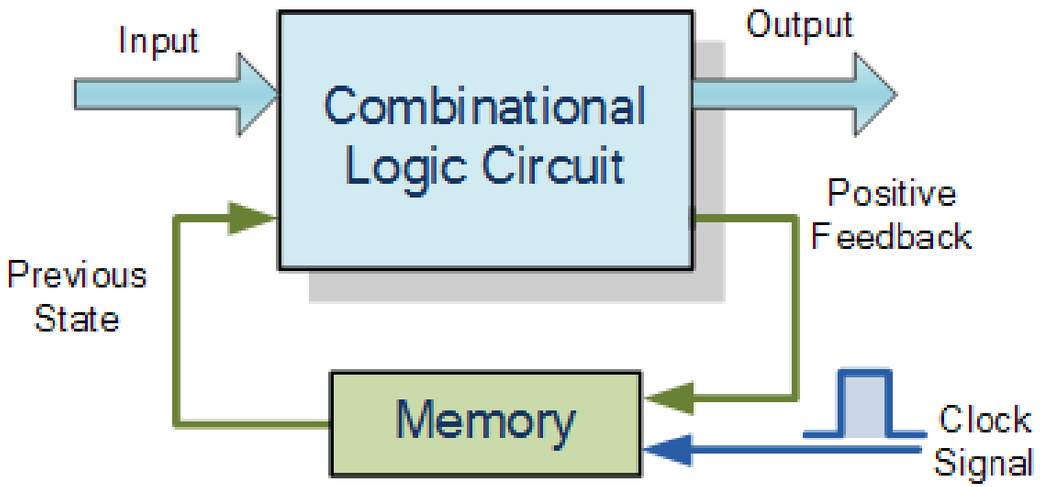
Overview

- SR Latch
- D Latch
- Clock and Synchronization
- Flip-flops
- Positive edge-triggered D Flip-flop
- JK Flip-flop
- Direct Inputs

Chapter-4

M. Morris Mano, Charles R. Kime and Tom Martin, **Logic and Computer Design Fundamentals**, Global (5th) Edition, Pearson Education Limited, 2016. ISBN: 9781292096124

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.

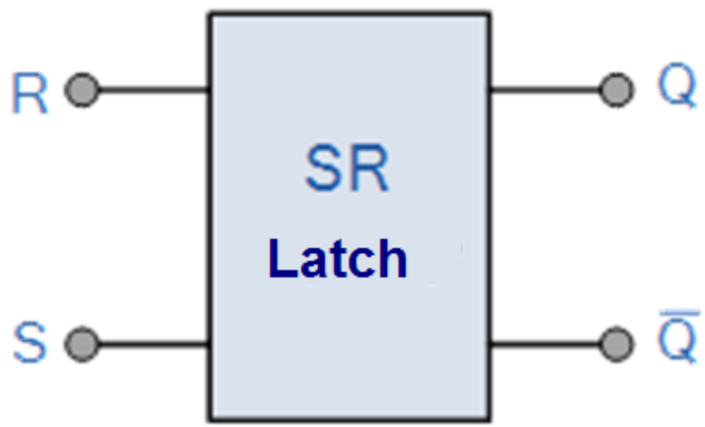
SR Latch

The **SR Latch**, can be considered as one of the most basic sequential logic circuit possible.
This simple circuit is basically a one-bit memory bistable device.

It has two inputs,

- S: will “SET” the device (meaning the output Q = “1”)
- R: will “RESET” the device (meaning the output Q = “0”)

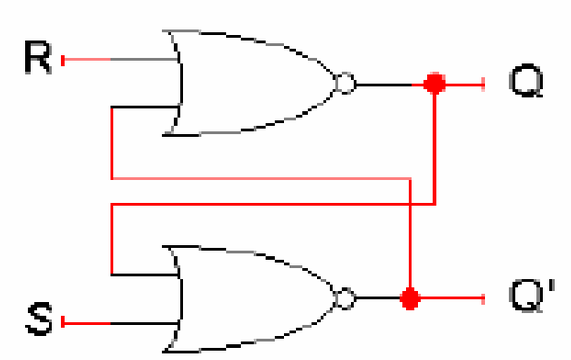
Output Q′ is always **complement of Q**



Symbol

A really confusing circuit

- Let's use NOR gates instead of inverters. The SR latch here has two inputs S and R, which will let us control the outputs Q and Q'.



- Q and Q' feed back into the circuit, so they're not only outputs, they're also inputs!
- To figure out how Q and Q' change, we must look at not only the inputs S and R, but also the *current* values of Q and Q'.

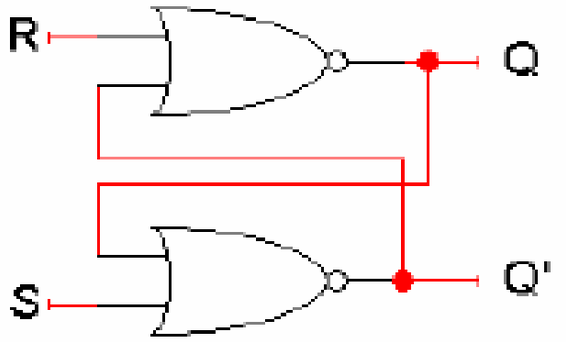
$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$

- Let's see how different input values for S and R affect this thing.

Storing a value: SR = 00

- What if S = 0 and R = 0?
- The equations on the right reduce to:

$$Q_{next} = (0 + Q'_{current})' = Q_{current}$$
$$Q'_{next} = (0 + Q_{current})' = Q'_{current}$$



$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$

- So when SR = 00, then $Q_{next} = Q_{current}$.
- This is exactly what we need to store values in the latch.

Setting the latch: SR = 10

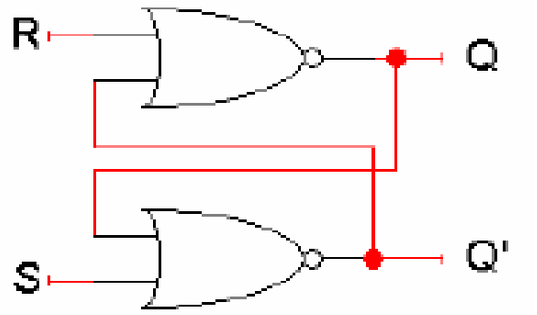
- What if S = 1 and R = 0?
- Since S = 1, Q'_{next} is 0, *regardless* of $Q_{current}$.

$$Q'_{next} = (1 + Q_{current})' = 0$$

- Then this new value of Q' goes into the top NOR gate, along with R = 0.

$$Q_{next} = (0 + 0)' = 1$$

- So when SR = 10, then $Q'_{next} = 0$ and $Q_{next} = 1$. This is how you set the latch to 1; the S input stands for "set."
- Notice it can take up to two steps (two gate delays) from the time S becomes 1 to the time Q_{next} becomes 1.
- But once Q_{next} becomes 1, the outputs will stop changing. This is a stable state.



$$Q_{next} = (R + Q'_{current})'$$
$$Q'_{next} = (S + Q_{current})'$$

SR latches are memories!

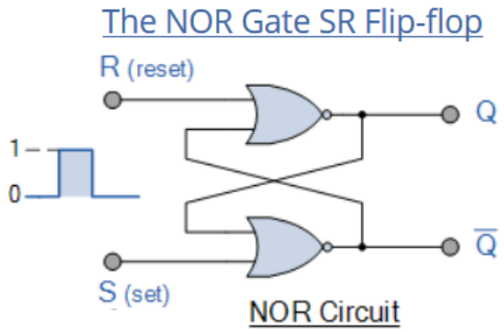
- This characteristic table shows that our latch provides everything we need in a memory: we can set it, reset it, or keep the current value.

S	R	Q
0	0	No change
0	1	0 (reset)
1	0	1 (set)

- The output Q represents the data stored in the latch. It is also called the state of the latch.

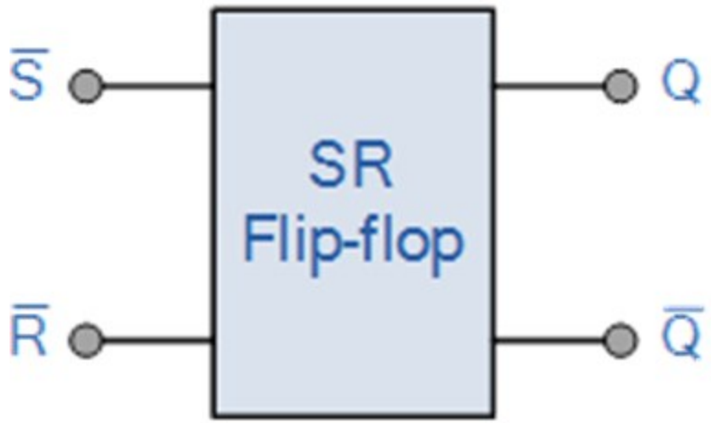
- We can expand the table above into a state table, which explicitly shows that the *next* values of Q and Q' depend on their *current* values, as well as on the inputs S and R.

Inputs S	Inputs R	Current Q	Current Q'	Next Q	Next Q'
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0

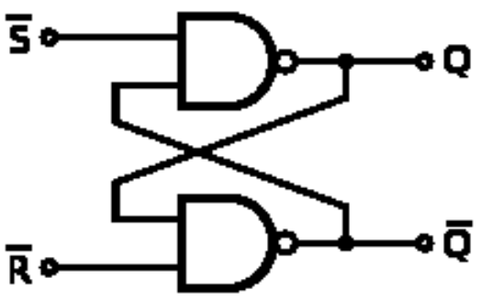


S′R′ Latch

The **S′R′ Latch**, is build with two cross-coupled NAND gate.



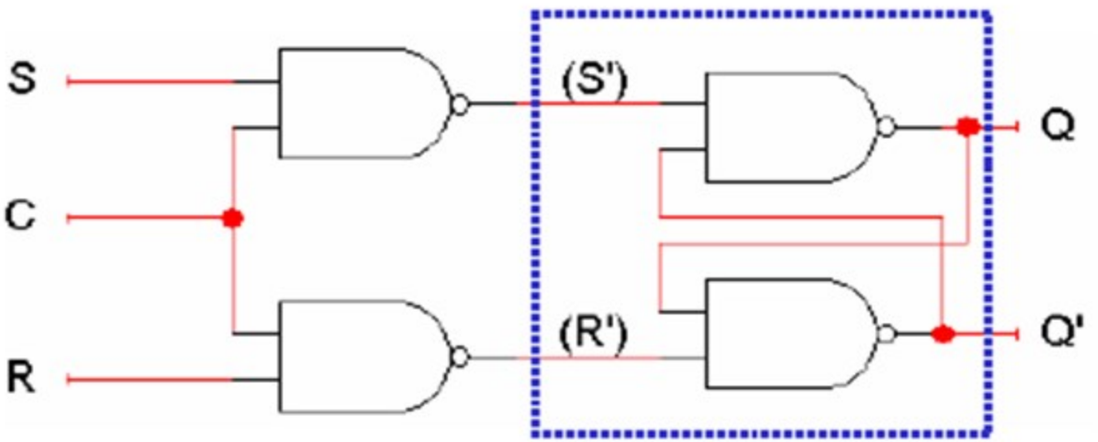
Symbol



S′	R′	Q
0	1	1
1	0	0
1	1	No Change
0	0	!

An SR latch with a control input

- Here is an SR latch with a control input C, which acts like an enable.

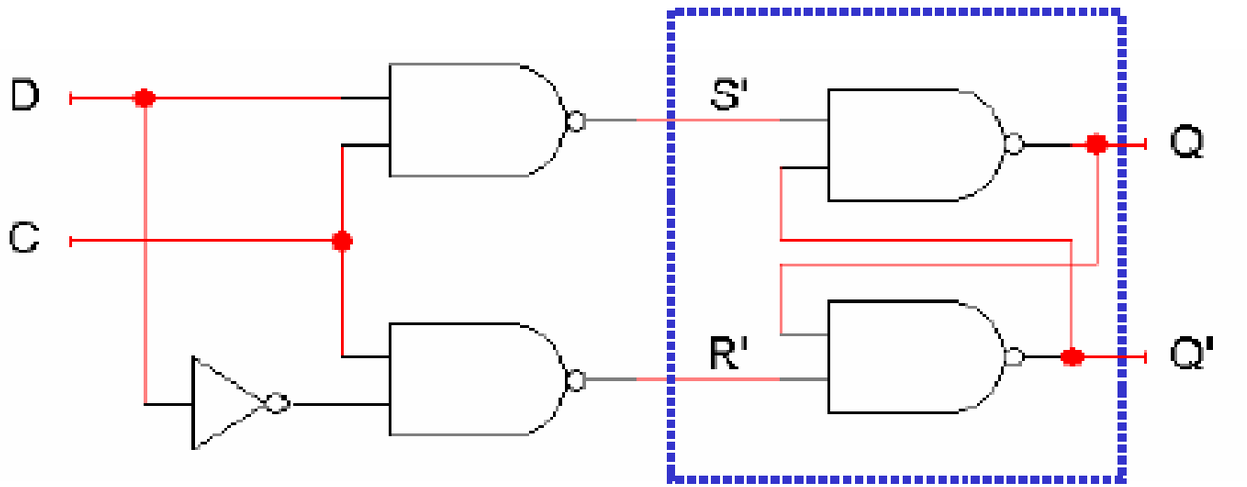


C	S	R	S'	R'	Q
0	x	x	1	1	No change
1	0	0	1	1	No change
1	0	1	1	0	0 (reset)
1	1	0	0	1	1 (set)
1	1	1	0	0	!

- 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.
- We'll see more of the control input later today.

D latch

- 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 C ("control").
  - When C = 0, S' and R' are both 1, so Q does not change.
  - When C = 1, the latch output Q will equal the input D.



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

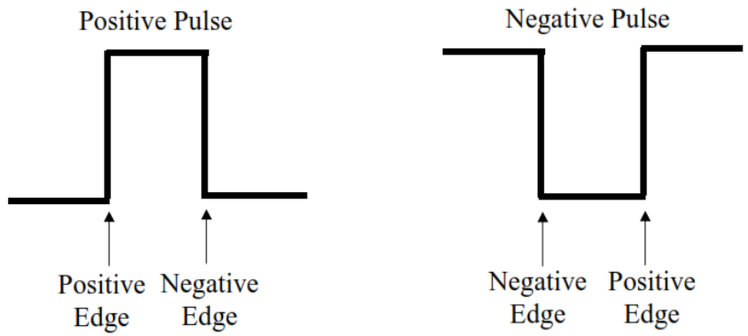
- There are two main advantages of a D latch.
  - No more messing with one input for set and another input for reset!
  - This latch has no "bad" input combinations to avoid. Any of the four possible assignments to C and D are valid.

Clocks and synchronization

- A clock is a special device that continuously outputs 0s and 1s.
  - The time it takes the clock to change from 1 to 0 and back to 1 is called the clock period, or clock cycle time.
  - The clock frequency is the inverse of the cl measurement for frequency is the hertz.

clock period

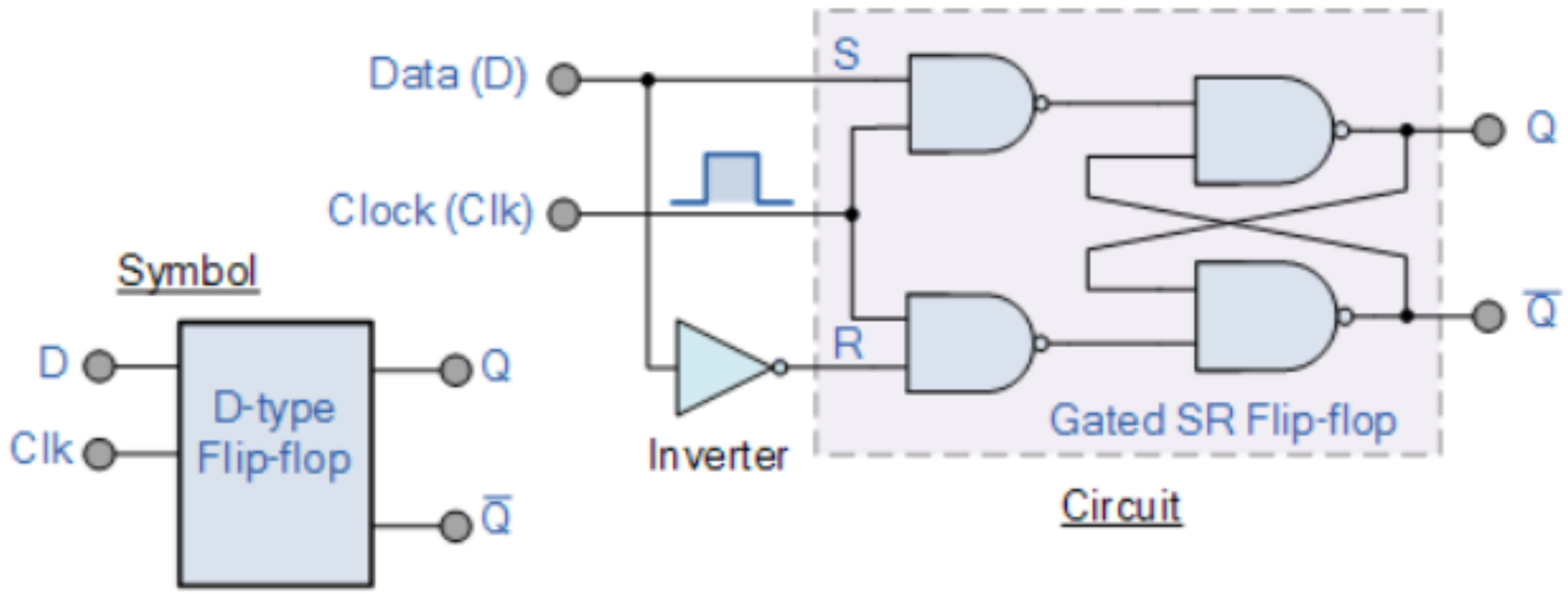
Clock Pulse Definition



- Clocks are often used to synchronize circuits.
  - They generate a repeating, predictable pattern of 0s and 1s that can trigger certain events in a circuit, such as writing to a latch.
  - If several circuits share a common clock signal, they can coordinate their actions with respect to one another.
- This is similar to how humans use real clocks for synchronization.

Flip-Flops

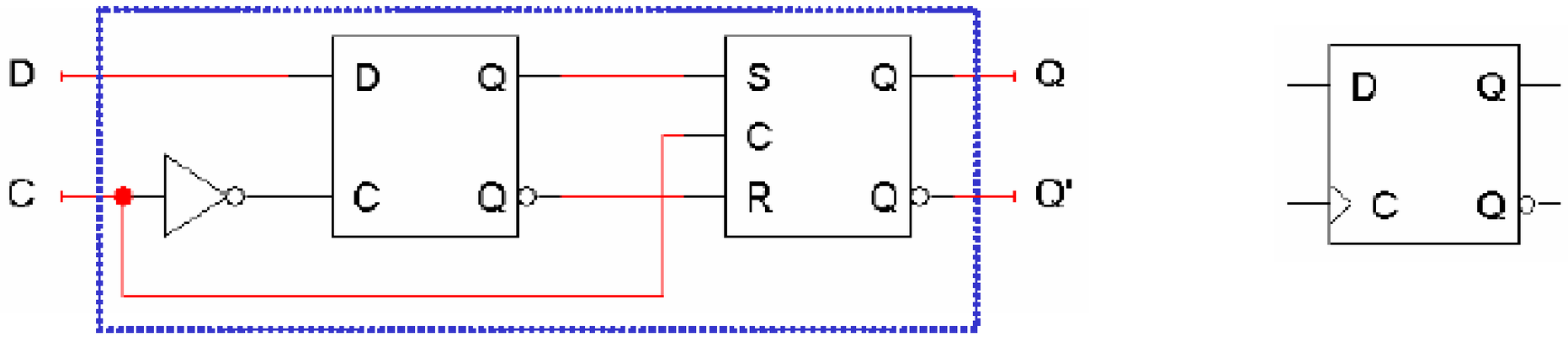
D-type Flip-Flop Circuit



The output of the flip flop would always change on every pulse applied to this data input.

To avoid this an additional input called the "CLOCK" or "ENABLE" input is used to isolate the data input from the flip flop's latching circuitry after the desired data has been stored. The effect is that D input condition is only copied to the output Q when the clock input is active. This then forms the basis of another sequential device called a **D Flip Flop**.

A positive edge-triggered D flip-flop

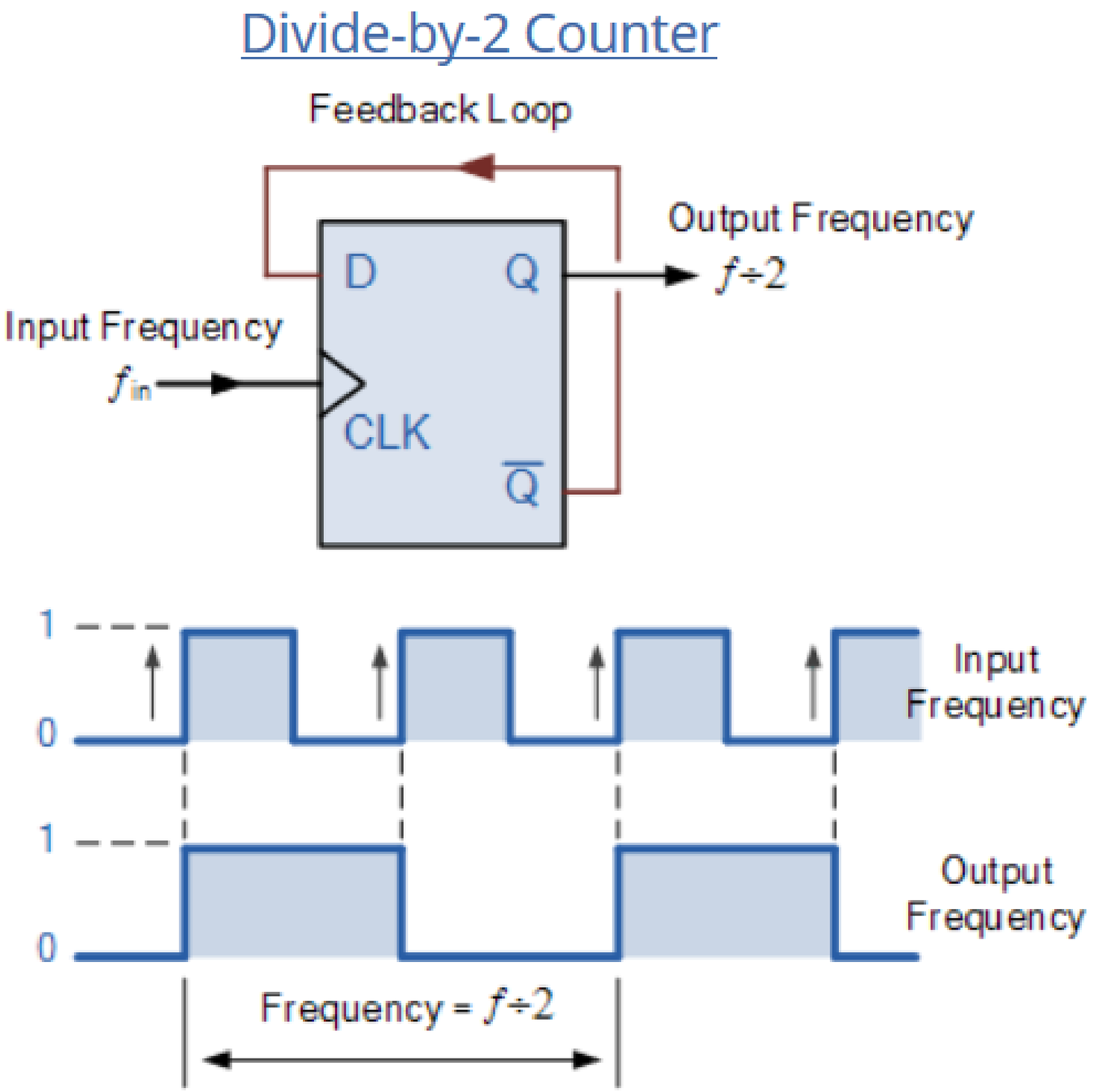


- This positive edge-triggered D flip-flop includes two latches.
  - The flip-flop output Q changes only *after* the positive edge of C.
  - The change is based on the flip-flop input value D that was present at the positive edge of the clock signal.
- A D flip-flop behaves like a D latch except for its positive edge-triggered nature, which is not explicit in the table below.

C	D	Q
0	x	No change
1	0	0 (reset)
1	1	1 (set)

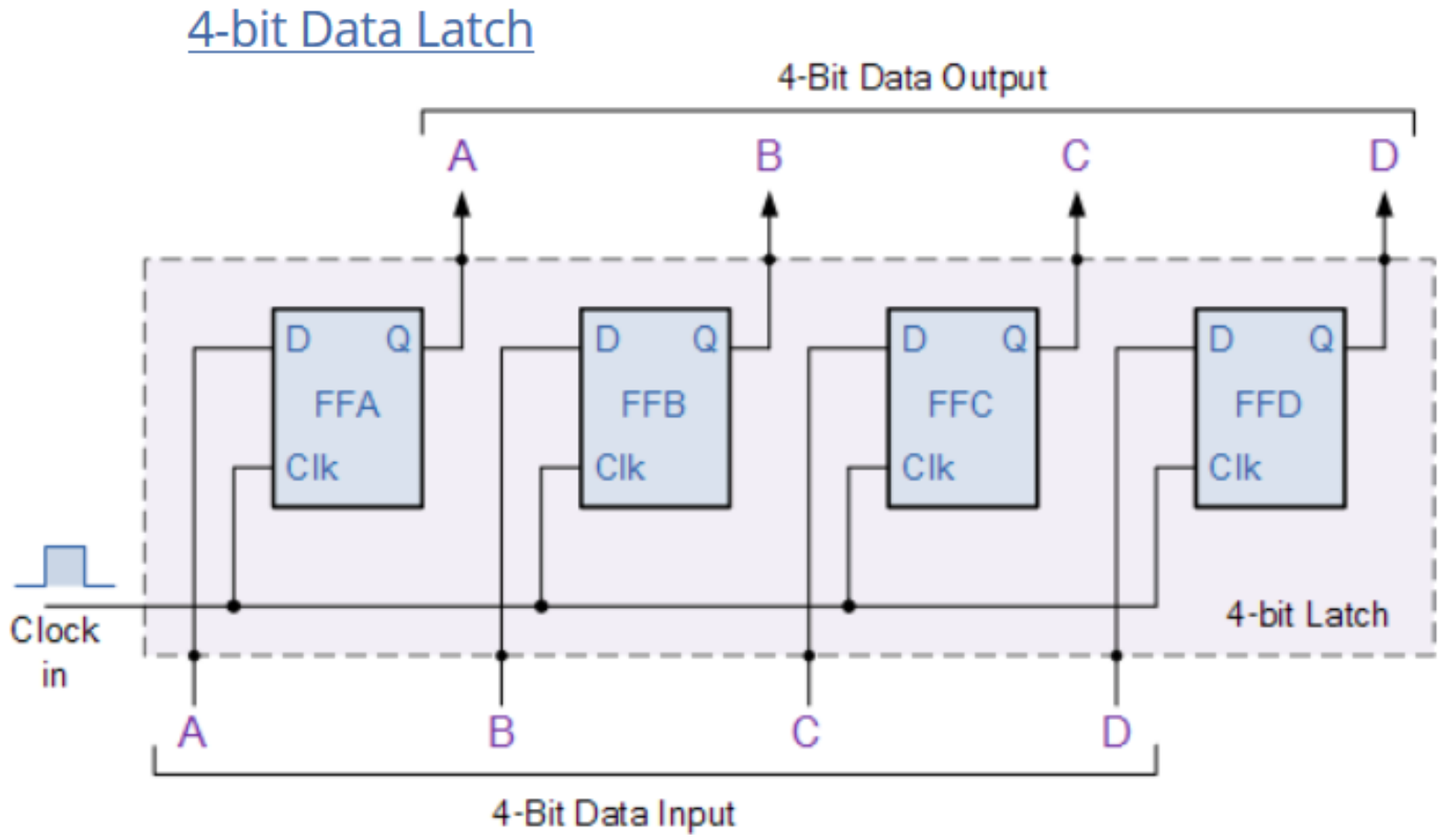
Divide-by-2 Counter

It can be seen from the frequency waveforms above, that by "feeding back" the output from Q to the input terminal D, the output pulses at Q have a frequency that are exactly one half (ƒ/2) that of the input clock frequency, (f_{IN}). In other words the circuit produces **frequency division** as it now divides the input frequency by a factor of two (an octave) as Q = 1 once every two clock cycles.



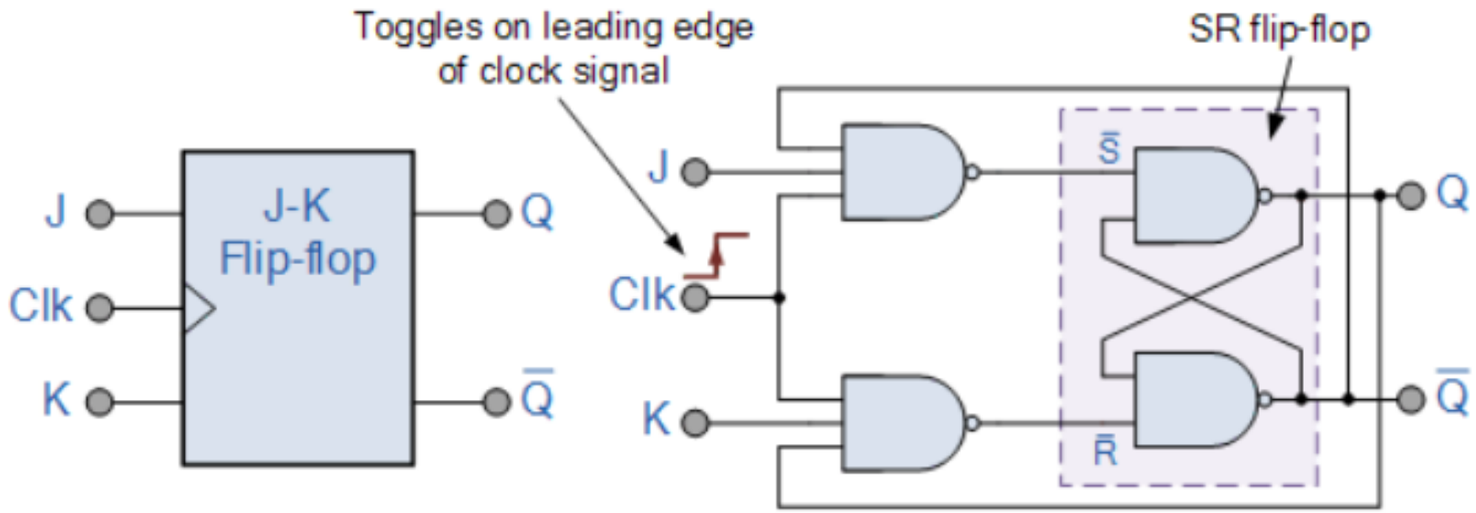
D Flip Flops as Data Latches

A data latch can be used as a device to hold or remember the data present on its data input, thereby acting a bit like a single bit memory device and IC's such as the TTL 74LS74 or the CMOS 4042 are available in Quad format exactly for this purpose. By connecting together four, *1-bit* data latches so that all their clock inputs are connected together and are "clocked" at the same time, a simple "4-bit" Data latch can be made as shown below.



JK Flip-flop

The Basic JK Flip-flop



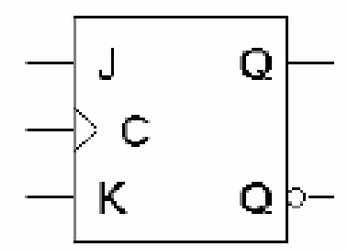
Both the S and the R inputs of the previous SR bistable have now been replaced by two inputs called the J and K inputs, respectively after its inventor Jack Kilby. Then this equates to: J = S and K = R.

The Truth Table for the JK Function

	Input		Output		Description
	J	K	Q	Q_{next}	
same as for the SR Latch	0	0	0	0	Memory no change
	0	0	1	1	
	0	1	1	0	Reset Q » 0
	0	1	0	0	
	1	0	0	1	Set Q » 1
	1	0	1	1	
toggle action	1	1	0	1	Toggle
	1	1	1	0	

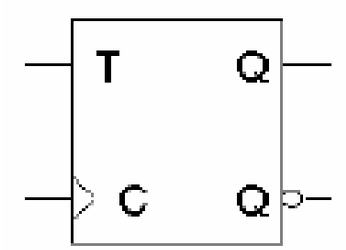
Flip-flop variations

- We can make different versions of flip-flops based on the D flip-flop, just like we made different latches based on the S'R' latch.
- The JK flip-flop has inputs that act like S and R, but JK = 11 *complements* the flip-flop's current state.



C	J	K	Q_{next}
0	x	x	No change
1	0	0	No change
1	0	1	0 (reset)
1	1	0	1 (set)
1	1	1	$Q'_{current}$

- 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	$Q'_{current}$

Characteristic tables

- The tables that we've made so far are called characteristic tables.
  - They show the next state Q(t+1) in terms of the current state Q(t) and the inputs.
  - For simplicity, the control input C is usually not listed.
  - Again, these tables don't indicate the positive edge-triggered nature of the flip-flops.

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

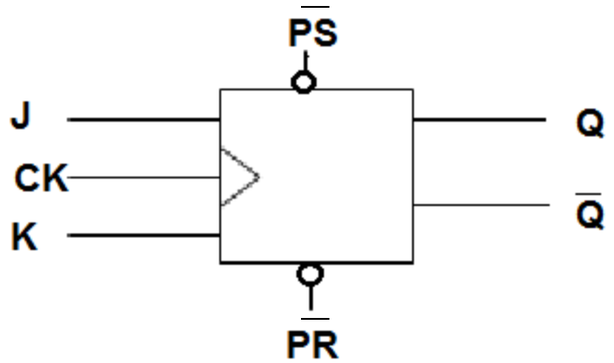
J	K	Q(t+1)	Operation
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Complement

T	Q(t+1)	Operation
0	Q(t)	No change
1	Q'(t)	Complement

Direct Inputs

Flip-flops often provides special inputs **PS, PR** for setting and resetting them:

- Independent of clock input
- Initializes the flip-flop



PS′	PR′	CK	J	K	Q(t+1)
1	1	↑	0	0	Q(t)
1	1	↑	0	1	0
1	1	↑	1	0	1
1	1	↑	1	1	Q′(t)
0	1	X	X	X	1
1	0	X	X	X	0
0	0	X	X	X	Undefined