

Sequential Logic Circuits

Sequential Logic Circuits use flip-flops as memory elements and in which their output is dependent on the input state

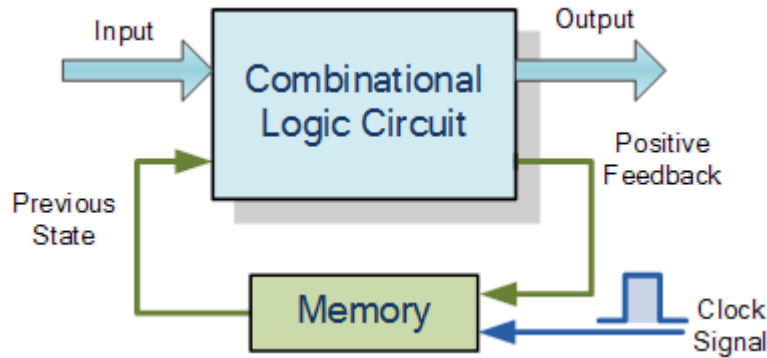
Unlike **Combinational Logic** circuits that change state depending upon the actual signals being applied to their inputs at that time, **Sequential Logic** circuits have some form of inherent “Memory” built in.

This means that sequential logic circuits are able to take into account their previous input state as well as those actually present, a sort of “before” and “after” effect is involved with sequential circuits.

In other words, the output state of a “sequential logic circuit” is a function of the following three states, the “present input”, the “past input” and/or the “past output”. *Sequential Logic circuits* remember these conditions and stay fixed in their current state until the next clock signal changes one of the states, giving sequential logic circuits “Memory”.

Sequential logic circuits are generally termed as *two state* or Bistable devices which can have their output or outputs set in one of two basic states, a logic level “1” or a logic level “0” and will remain “latched” (hence the name latch) indefinitely in this current state or condition until some other input trigger pulse or signal is applied which will cause the bistable to change its state once again.

Sequential Logic Representation

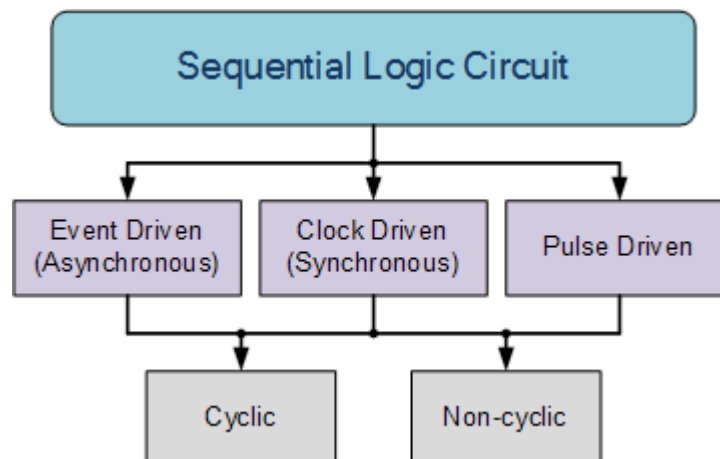


The word “Sequential” means that things happen in a “sequence”, one after another and in **Sequential Logic** circuits, the actual clock signal determines when things will happen next. Simple sequential logic circuits can be constructed from standard **Bistable** circuits such as: *Flip-flops*, *Latches* and *Counters* and which themselves can be made by simply connecting together universal NAND Gates and/or NOR Gates in a particular combinational way to produce the required sequential circuit.

Classification of Sequential Logic

As standard logic gates are the building blocks of combinational circuits, bistable latches and flip-flops are the basic building blocks of sequential logic circuits. Sequential logic circuits can be constructed to produce either simple edge-triggered flip-flops or more complex sequential circuits such as storage registers, shift registers, memory devices or counters. Either way sequential logic circuits can be divided into the following three main categories:

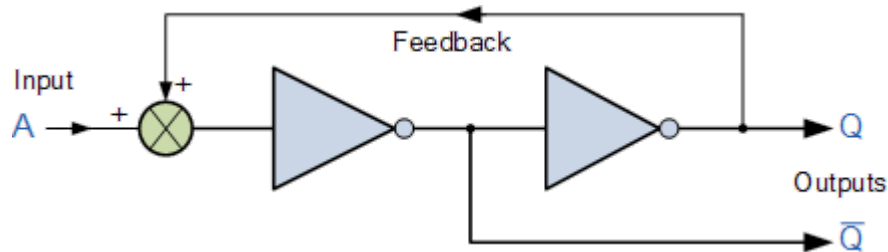
1. Event Driven – asynchronous circuits that change state immediately when enabled.
2. Clock Driven – synchronous circuits that are synchronised to a specific clock signal.
3. Pulse Driven – which is a combination of the two that responds to triggering pulses.



As well as the two logic states mentioned above logic level “1” and logic level “0”, a third element is introduced that separates **sequential logic** circuits from their **combinational logic** counterparts, namely *TIME*. Sequential logic circuits return back to their original steady state once reset and sequential circuits with loops or feedback paths are said to be “cyclic” in nature.

We now know that in sequential circuits changes occur only on the application of a clock signal making it synchronous, otherwise the circuit is asynchronous and depends upon an external input. To retain their current state, sequential circuits rely on feedback and this occurs when a fraction of the output is fed back to the input and this is demonstrated as:

Sequential Feedback Loop



The two inverters or NOT gates are connected in series with the output at Q fed back to the input. Unfortunately, this configuration never changes state because the output will always be the same, either a “1” or a “0”, it is permanently set. However, we can see how feedback works by examining the most basic sequential logic components, called the SR flip-flop.

SR Flip-Flop

The **SR flip-flop**, also known as a *SR Latch*, can be considered as one of the most basic sequential logic circuit possible. This simple flip-flop is basically a one-bit memory bistable device that has two inputs, one which will “SET” the device (meaning the output = “1”), and is labelled **S** and one which will “RESET” the device (meaning the output = “0”), labelled **R**.

Then the SR description stands for “Set-Reset”. The reset input resets the flip-flop back to its original state with an output Q that will be either at a logic level “1” or logic “0” depending upon this set/reset condition.

A basic NAND gate SR flip-flop circuit provides feedback from both of its outputs back to its opposing inputs and is commonly used in memory circuits to store a single data bit. Then the SR flip-flop actually has three inputs, Set, Reset and its current output Q relating to its current state or history. The term “Flip-flop” relates to the actual operation of the device, as it can be “flipped” into one logic Set state or “flopped” back into the opposing logic Reset state.

The NAND Gate SR Flip-Flop

The simplest way to make any basic single bit set-reset SR flip-flop is to connect together a pair of cross-coupled 2-input NAND gates as shown, to form a Set-Reset Bistable also known as an active LOW SR NAND Gate Latch, so that there is feedback from each output to one of the other NAND gate inputs. This device consists of two inputs, one called the *Set*, S and the other called the *Reset*, R with two corresponding outputs Q and its inverse or complement \bar{Q} (not-Q) as shown below.

The Basic SR Flip-flop



The Set State

Consider the circuit shown above. If the input R is at logic level "0" ($R = 0$) and input S is at logic level "1" ($S = 1$), the NAND gate Y has at least one of its inputs at logic "0" therefore, its output \bar{Q} must be at a logic level "1" (NAND Gate principles). Output \bar{Q} is also fed back to input "A" and so both inputs to NAND gate X are at logic level "1", and therefore its output Q must be at logic level "0".

Again NAND gate principals. If the reset input R changes state, and goes HIGH to logic "1" with S remaining HIGH also at logic level "1", NAND gate Y inputs are now $R = "1"$ and $B = "0"$. Since one of its inputs is still at logic level "0" the output at \bar{Q} still remains HIGH at logic level "1" and there is no change of state. Therefore, the flip-flop circuit is said to be "Latched" or "Set" with $\bar{Q} = "1"$ and $Q = "0"$.

Reset State

In this second stable state, \bar{Q} is at logic level "0", (not $Q = "0"$) its inverse output at Q is at logic level "1", ($Q = "1"$), and is given by $R = "1"$ and $S = "0"$. As gate X has one of its inputs at logic "0" its output Q must equal logic level "1" (again NAND gate principles). Output Q is fed back to input "B", so both inputs to NAND gate Y are at logic "1", therefore, $\bar{Q} = "0"$.

If the set input, S now changes state to logic "1" with input R remaining at logic "1", output \bar{Q} still remains LOW at logic level "0" and there is no change of state. Therefore, the flip-flop circuits "Reset" state has also been latched and we can define this "set/reset" action in the following truth table.

Truth Table for this Set-Reset Function

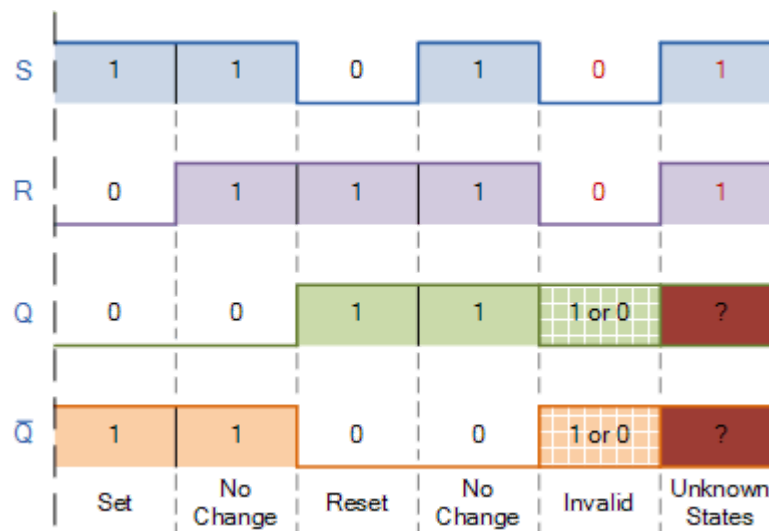
State	S	R	Q	\bar{Q}	Description
Set	1	0	0	1	Set $\bar{Q} \gg 1$
	1	1	0	1	no change

Reset	0	1	1	0	Reset $\bar{Q} \gg 0$
	1	1	1	0	no change
Invalid	0	0	1	1	Invalid Condition

It can be seen that when both inputs $S = "1"$ and $R = "1"$ the outputs Q and \bar{Q} can be at either logic level "1" or "0", depending upon the state of the inputs S or R BEFORE this input condition existed. Therefore the condition of $S = R = "1"$ does not change the state of the outputs Q and \bar{Q} .

However, the input state of $S = "0"$ and $R = "0"$ is an undesirable or invalid condition and must be avoided. The condition of $S = R = "0"$ causes both outputs Q and \bar{Q} to be HIGH together at logic level "1" when we would normally want \bar{Q} to be the inverse of Q . The result is that the flip-flop loses control of Q and \bar{Q} , and if the two inputs are now switched "HIGH" again after this condition to logic "1", the flip-flop becomes unstable and switches to an unknown data state based upon the unbalance as shown in the following switching diagram.

S-R Flip-flop Switching Diagram

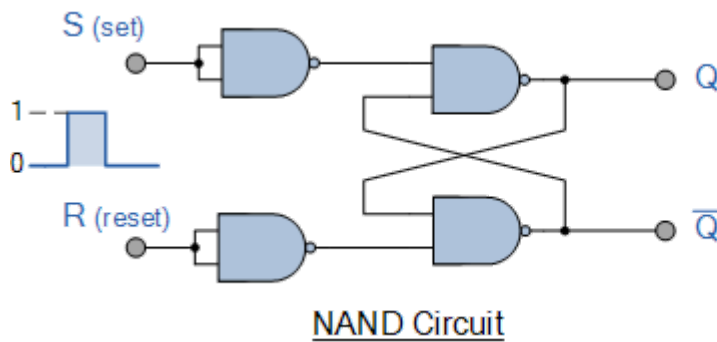


This unbalance can cause one of the outputs to switch faster than the other resulting in the flip-flop switching to one state or the other which may not be the required state and data corruption will exist. This unstable condition is generally known as its **Meta-stable** state.

Then, a simple NAND gate SR flip-flop or NAND gate SR latch can be set by applying a logic "0", (LOW) condition to its Set input and reset again by then applying a logic "0" to its Reset input. The SR flip-flop is said to be in an "invalid" condition (Meta-stable) if both the set and reset inputs are activated simultaneously.

As we have seen above, the basic NAND gate SR flip-flop requires logic "0" inputs to flip or change state from Q to \bar{Q} and vice versa. We can however, change this basic flip-flop circuit to one that changes state by the application of positive going input signals with the addition of two extra NAND gates connected as inverters to the S and R inputs as shown.

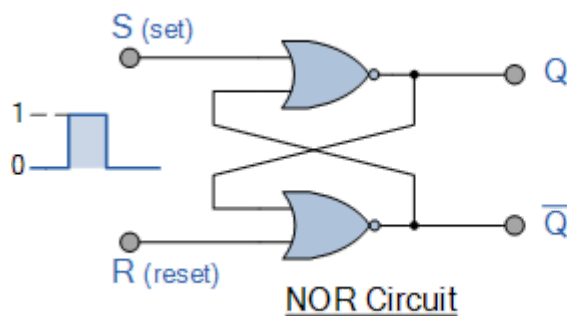
Positive NAND Gate SR Flip-flop



S	R	Q	\bar{Q}
0	0	No change	
0	1	0	1
1	0	1	0
1	1	X	X
(Invalid)			

As well as using NAND gates, it is also possible to construct simple one-bit **SR Flip-flops** using two cross-coupled NOR gates connected in the same configuration. The circuit will work in a similar way to the NAND gate circuit above, except that the inputs are active HIGH and the invalid condition exists when both its inputs are at logic level "1", and this is shown below.

The NOR Gate SR Flip-flop



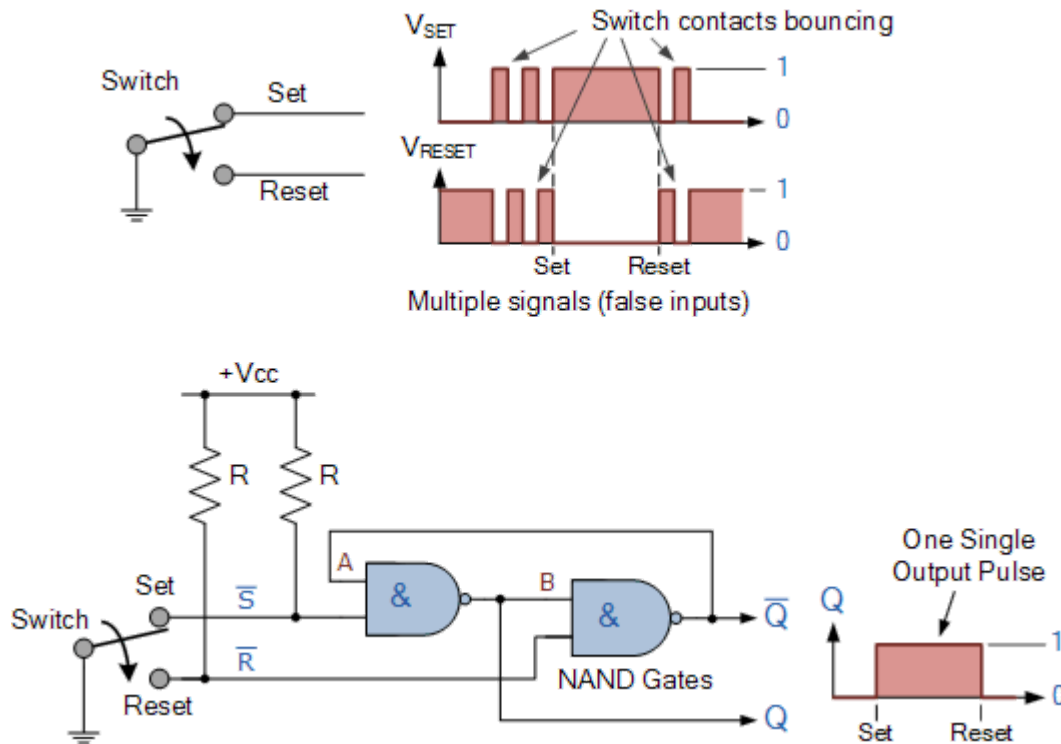
S	R	Q	\bar{Q}
0	0	No change	
0	1	1	0
1	0	0	1
1	1	X	X
(Invalid)			

Switch Debounce Circuits

Edge-triggered flip-flops require a nice clean signal transition, and one practical use of this type of set-reset circuit is as a latch used to help eliminate mechanical switch "bounce". As its name implies, switch bounce occurs when the contacts of any mechanically operated switch, push-button or keypad are operated and the internal switch contacts do not fully close cleanly, but bounce together first before closing (or opening) when the switch is pressed.

This gives rise to a series of individual pulses which can be as long as tens of milliseconds that an electronic system or circuit such as a digital counter may see as a series of logic pulses instead of one long single pulse and behave incorrectly. For example, during this bounce period the output voltage can fluctuate wildly and may register multiple input counts instead of one single count. Then set-reset SR Flip-flops or Bistable Latch circuits can be used to eliminate this kind of problem and this is demonstrated below.

SR Flip Flop Switch Debounce Circuit



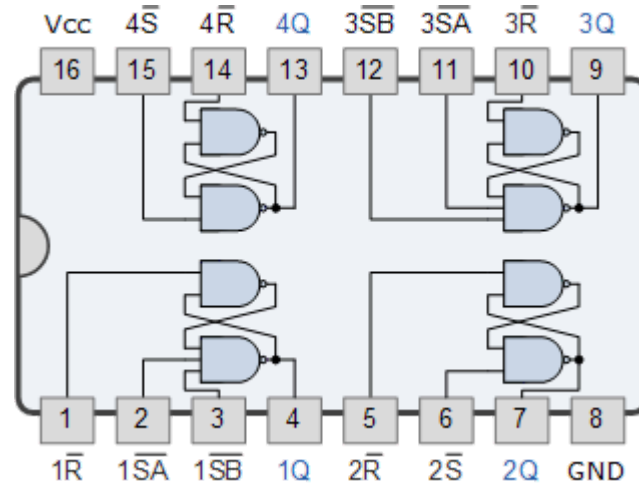
Depending upon the current state of the output, if the set or reset buttons are depressed the output will change over in the manner described above and any additional unwanted inputs (bounces) from the mechanical action of the switch will have no effect on the output at Q .

When the other button is pressed, the very first contact will cause the latch to change state, but any additional mechanical switch bounces will also have no effect. The SR flip-flop can then be RESET automatically after a short period of time, for example 0.5 seconds, so as to register any additional and intentional repeat inputs from the same switch contacts, such as multiple inputs from a keyboard's "RETURN" key.

Commonly available IC's specifically made to overcome the problem of switch bounce are the MAX6816, single input, MAX6817, dual input and the MAX6818 octal input switch debouncer IC's. These chips contain the necessary flip-flop circuitry to provide clean interfacing of mechanical switches to digital systems.

Set-Reset bistable latches can also be used as Monostable (one-shot) pulse generators to generate a single output pulse, either high or low, of some specified width or time period for timing or control purposes. The 74LS279 is a Quad SR Bistable Latch IC, which contains four individual NAND type bistable's within a single chip enabling switch debounce or monostable/astable clock circuits to be easily constructed.

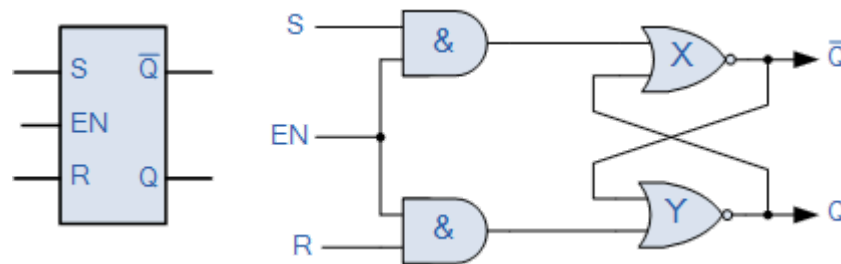
Quad SR Bistable Latch 74LS279



Gated or Clocked SR Flip-Flop

It is sometimes desirable in sequential logic circuits to have a bistable SR flip-flop that only changes state when certain conditions are met regardless of the condition of either the Set or the Reset inputs. By connecting a 2-input AND gate in series with each input terminal of the SR Flip-flop a Gated SR Flip-flop can be created. This extra conditional input is called an “Enable” input and is given the prefix of “EN”. The addition of this input means that the output at Q only changes state when it is HIGH and can therefore be used as a clock (CLK) input making it level-sensitive as shown below.

Gated SR Flip-flop

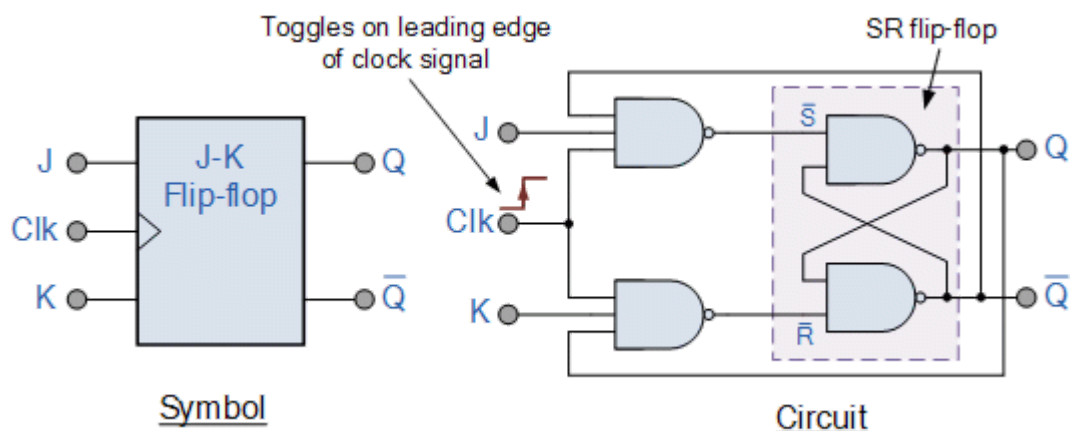


When the Enable input “EN” is at logic level “0”, the outputs of the two AND gates are also at logic level “0”, (AND Gate principles) regardless of the condition of the two inputs S and R, latching the two outputs Q and \bar{Q} into their last known state. When the enable input “EN” changes to logic level “1” the circuit responds as a normal SR bistable flip-flop with the two AND gates becoming transparent to the Set and Reset signals.

This additional enable input can also be connected to a clock timing signal (CLK) adding clock synchronisation to the flip-flop creating what is sometimes called a “Clocked SR Flip-flop”. So a **Gated Bistable SR Flip-flop** operates as a standard bistable latch but the outputs are only activated when a logic “1” is applied to its EN input and deactivated by a logic “0”.

In the next tutorial about **Sequential Logic Circuits**, we will look at another type of simple edge-triggered flip-flop which is very similar to the **RS flip-flop** called a *JK Flip-flop* named after its inventor, Jack Kilby. The JK flip-flop is the most widely used of all the flip-flop designs as it is

considered to be a universal device.



The JK Flip Flop

The JK Flip-flop is similar to the SR Flip-flop but there is no change in state when the J and K inputs are both LOW

The basic S-R NAND flip-flop circuit has many advantages and uses in sequential logic circuits but it suffers from two basic switching problems.

1. the Set = 0 and Reset = 0 condition ($S = R = 0$) must always be avoided
2. if Set or Reset change state while the enable (EN) input is high the correct latching action may not occur

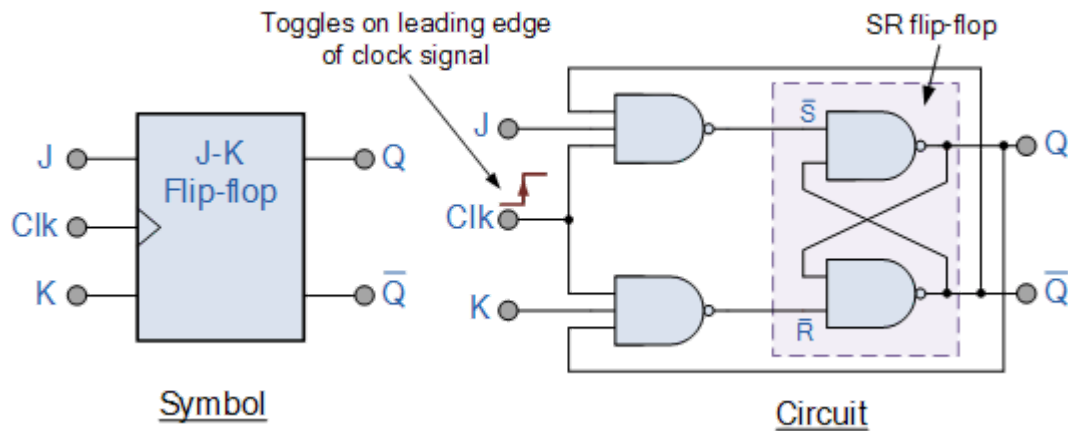
Then to overcome these two fundamental design problems with the SR flip-flop design, the **JK flip Flop** was developed.

This simple **JK flip Flop** is the most widely used of all the flip-flop designs and is considered to be a universal flip-flop circuit. The two inputs labelled "J" and "K" are not shortened abbreviated letters of other words, such as "S" for Set and "R" for Reset, but are themselves autonomous letters chosen by its inventor Jack Kilby to distinguish the flip-flop design from other types.

The sequential operation of the JK flip flop is exactly the same as for the previous SR flip-flop with the same "Set" and "Reset" inputs. The difference this time is that the "JK flip flop" has no invalid or forbidden input states of the SR Latch even when S and R are both at logic "1".

The **JK flip flop** is basically a gated SR flip-flop with the addition of a clock input circuitry that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level "1". Due to this additional clocked input, a JK flip-flop has four possible input combinations, "logic 1", "logic 0", "no change" and "toggle". The symbol for a JK flip flop is similar to that of an *SR Bistable Latch* as seen in the previous tutorial except for the addition of a clock input.

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 two 2-input AND gates of the gated SR bistable have now been replaced by two 3-input NAND gates with the third input of each gate connected to the outputs at Q and \bar{Q} . This cross coupling of the SR flip-flop allows the previously invalid condition of $S = "1"$ and $R = "1"$ state to be used to produce a "toggle action" as the two inputs are now interlocked.

If the circuit is now "SET" the J input is inhibited by the "0" status of \bar{Q} through the lower NAND gate. If the circuit is "RESET" the K input is inhibited by the "0" status of Q through the upper NAND gate. As Q and \bar{Q} are always different we can use them to control the input. When both inputs J and K are equal to logic "1", the JK flip flop toggles as shown in the following truth table.

The Truth Table for the JK Function

	Clock	Input		Output		Description
	Clk	J	K	Q	\bar{Q}	
same as for the SR Latch	X	0	0	1	0	Memory no change
	X	0	0	0	1	
	$\bar{\downarrow}$	0	1	1	0	Reset Q » 0
	X	0	1	0	1	
	$\bar{\downarrow}$	1	0	0	1	Set Q » 1
	X	1	0	1	0	
toggle	$\bar{\downarrow}$	1	1	0	1	Toggle

action	\bar{J}	1	1	1	0

Then the JK flip-flop is basically an SR flip flop with feedback which enables only one of its two input terminals, either SET or RESET to be active at any one time thereby eliminating the invalid condition seen previously in the SR flip flop circuit.

Also when both the J and the K inputs are at logic level "1" at the same time, and the clock input is pulsed "HIGH", the circuit will "toggle" from its SET state to a RESET state, or visa-versa. This results in the JK flip flop acting more like a T-type toggle flip-flop when both terminals are "HIGH".

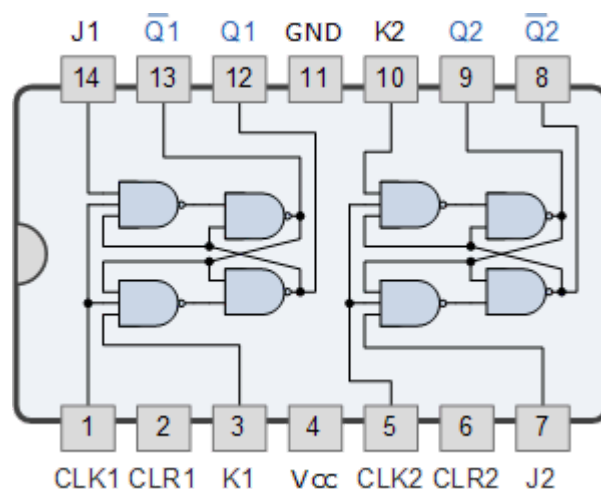
Although this circuit is an improvement on the clocked SR flip-flop it still suffers from timing problems called "race" if the output Q changes state before the timing pulse of the clock input has time to go "OFF". To avoid this the timing pulse period (T) must be kept as short as possible (high frequency). As this is sometimes not possible with modern TTL IC's the much improved **Master-Slave JK Flip-flop** was developed.

Master-Slave JK Flip-flop

The master-slave flip-flop eliminates all the timing problems by using two SR flip-flops connected together in a series configuration. One flip-flop acts as the "Master" circuit, which triggers on the leading edge of the clock pulse while the other acts as the "Slave" circuit, which triggers on the falling edge of the clock pulse. This results in the two sections, the master section and the slave section being enabled during opposite half-cycles of the clock signal.

The TTL 74LS73 is a Dual JK flip-flop IC, which contains two individual JK type bistable's within a single chip enabling single or master-slave toggle flip-flops to be made. Other JK flip flop IC's include the 74LS107 Dual JK flip-flop with clear, the 74LS109 Dual positive-edge triggered JK flip flop and the 74LS112 Dual negative-edge triggered flip-flop with both preset and clear inputs.

Dual JK Flip-flop 74LS73



Other Popular JK Flip-flop ICs

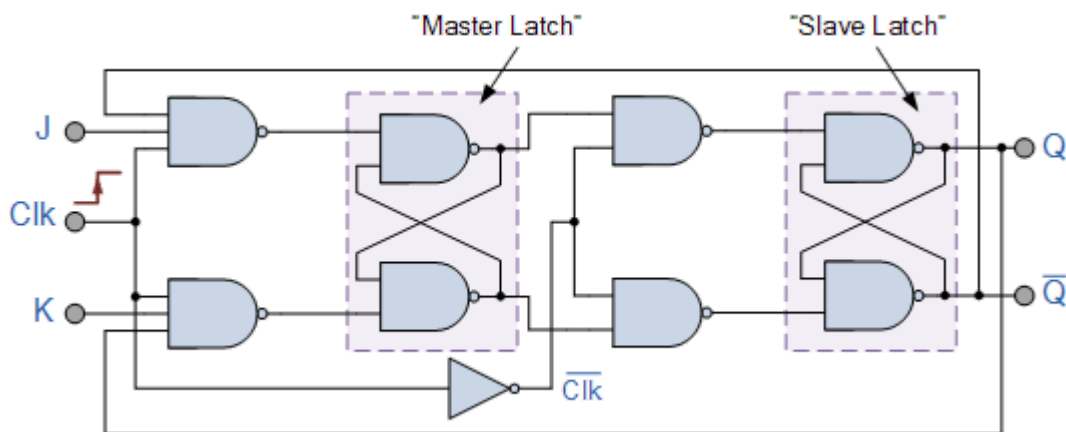
Device Number	Subfamily	Device Description

74LS73	LS TTL	Dual JK-type Flip Flops with Clear
74LS76	LS TTL	Dual JK-type Flip Flops with Preset and Clear
74LS107	LS TTL	Dual JK-type Flip Flops with Clear
4027B	Standard CMOS	Dual JK-type Flip Flop

The Master-Slave JK Flip-flop

The **Master-Slave Flip-Flop** is basically two gated SR flip-flops connected together in a series configuration with the slave having an inverted clock pulse. The outputs from Q and \bar{Q} from the "Slave" flip-flop are fed back to the inputs of the "Master" with the outputs of the "Master" flip flop being connected to the two inputs of the "Slave" flip flop. This feedback configuration from the slave's output to the master's input gives the characteristic toggle of the JK flip flop as shown below.

The Master-Slave JK Flip Flop



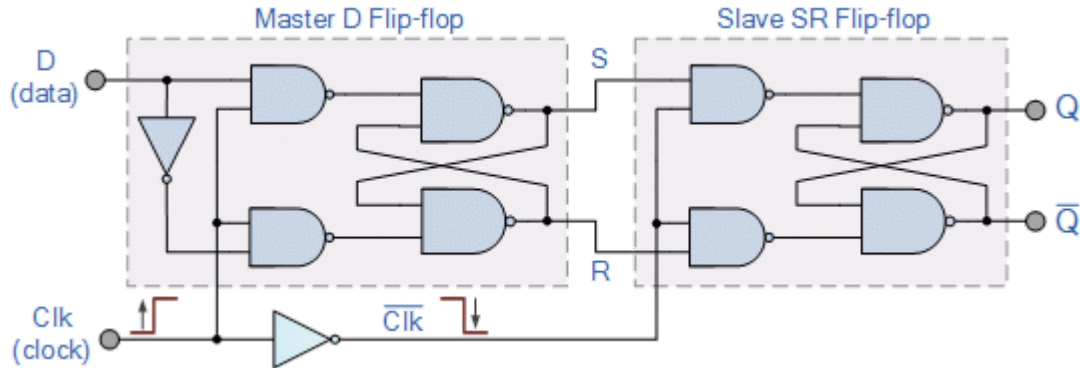
The input signals J and K are connected to the gated "master" SR flip flop which "locks" the input condition while the clock (Clk) input is "HIGH" at logic level "1". As the clock input of the "slave" flip flop is the inverse (complement) of the "master" clock input, the "slave" SR flip flop does not toggle. The outputs from the "master" flip flop are only "seen" by the gated "slave" flip flop when the clock input goes "LOW" to logic level "0".

When the clock is "LOW", the outputs from the "master" flip flop are latched and any additional changes to its inputs are ignored. The gated "slave" flip flop now responds to the state of its inputs passed over by the "master" section.

Then on the "Low-to-High" transition of the clock pulse the inputs of the "master" flip flop are fed through to the gated inputs of the "slave" flip flop and on the "High-to-Low" transition the same inputs are reflected on the output of the "slave" making this type of flip flop edge or pulse-triggered.

Then, the circuit accepts input data when the clock signal is "HIGH", and passes the data to the output on the falling-edge of the clock signal. In other words, the **Master-Slave JK Flip flop** is a "Synchronous" device as it only passes data with the timing of the clock signal.

In the next tutorial about **Sequential Logic Circuits**, we will look at *Multivibrators* that are used as waveform generators to produce the clock signals to switch sequential circuits.



The D-type Flip Flop

The D-type flip-flop is a modified Set-Reset flip-flop with the addition of an inverter to prevent the S and R inputs from being at the same logic level

One of the main disadvantages of the basic **SR NAND Gate Bistable** circuit is that the indeterminate input condition of SET = "0" and RESET = "0" is forbidden.

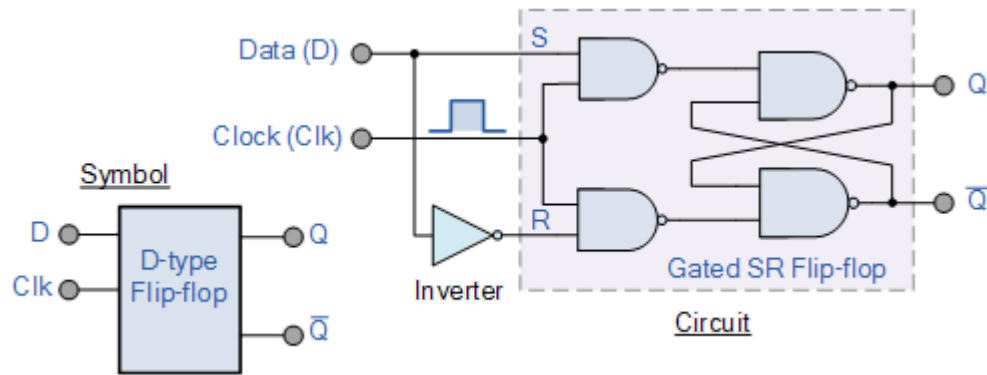
This state will force both outputs to be at logic "1", over-riding the feedback latching action and whichever input goes to logic level "1" first will lose control, while the other input still at logic "0" controls the resulting state of the latch.

But in order to prevent this from happening an inverter can be connected between the "SET" and the "RESET" inputs to produce another type of flip flop circuit known as a *Data Latch, Delay flip flop, D-type Bistable, D-type Flip Flop* or just simply a **D Flip Flop** as it is more generally called.

The **D Flip Flop** is by far the most important of the clocked flip-flops as it ensures that ensures that inputs S and R are never equal to one at the same time. The D-type flip flop are constructed from a gated SR flip-flop with an inverter added between the S and the R inputs to allow for a single D(Data) input.

Then this single data input, labelled "D" and is used in place of the "Set" signal, and the inverter is used to generate the complementary "Reset" input thereby making a level-sensitive D-type flip-flop from a level-sensitive SR-latch as now $S = D$ and $R = \text{not } D$ as shown.

D-type Flip-Flop Circuit



We remember that a simple SR flip-flop requires two inputs, one to “SET” the output and one to “RESET” the output. By connecting an inverter (NOT gate) to the SR flip-flop we can “SET” and “RESET” the flip-flop using just one input as now the two input signals are complements of each other. This complement avoids the ambiguity inherent in the SR latch when both inputs are LOW, since that state is no longer possible.

Thus this single input is called the “DATA” input. If this data input is held HIGH the flip flop would be “SET” and when it is LOW the flip flop would change and become “RESET”. However, this would be rather pointless since 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 Dinput 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**.

The “D flip flop” will store and output whatever logic level is applied to its data terminal so long as the clock input is HIGH. Once the clock input goes LOW the “set” and “reset” inputs of the flip-flop are both held at logic level “1” so it will not change state and store whatever data was present on its output before the clock transition occurred. In other words the output is “latched” at either logic “0” or logic “1”.

Truth Table for the D-type Flip Flop

Clk	D	Q		Description
↓ » 0	X	Q	\bar{Q}	Memory no change
↑ » 1	0	0	1	Reset Q » 0
↑ » 1	1	1	0	Set Q » 1

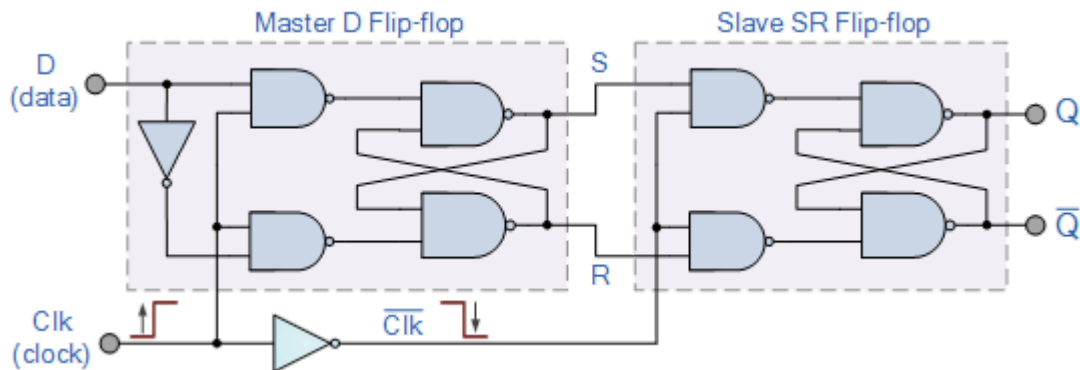
Note that: ↓ and ↑ indicates direction of clock pulse as it is assumed D-type flip flops are edge triggered

The Master-Slave D Flip Flop

The basic **D-type flip flop** can be improved further by adding a second SR flip-flop to its output that is activated on the complementary clock signal to produce a “Master-Slave D-type flip flop”. On the leading edge of the clock signal (LOW-to-HIGH) the first stage, the “master” latches the input condition at D , while the output stage is deactivated.

On the trailing edge of the clock signal (HIGH-to-LOW) the second “slave” stage is now activated, latching on to the output from the first master circuit. Then the output stage appears to be triggered on the negative edge of the clock pulse. “Master-Slave D-type flip flops” can be constructed by the cascading together of two latches with opposite clock phases as shown.

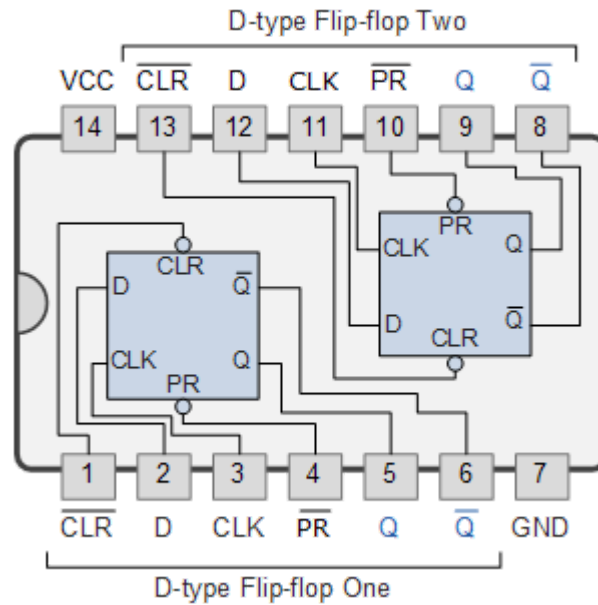
The Master-Slave D Flip Flop Circuit



We can see from above that on the leading edge of the clock pulse the master flip-flop will be loading data from the data D input, therefore the master is “ON”. With the trailing edge of the clock pulse the slave flip-flop is loading data, i.e. the slave is “ON”. Then there will always be one flip-flop “ON” and the other “OFF” but never both the master and slave “ON” at the same time. Therefore, the output Q acquires the value of D , only when one complete pulse, ie, 0-1-0 is applied to the clock input.

There are many different D flip-flop IC’s available in both TTL and CMOS packages with the more common being the 74LS74 which is a Dual D flip-flop IC, which contains two individual D type bistable’s within a single chip enabling single or master-slave toggle flip-flops to be made. Other D flip-flop IC’s include the 74LS174 HEX D flip-flop with direct clear input, the 74LS175 Quad D flip-flop with complementary outputs and the 74LS273 Octal D-type flip flop containing eight D-type flip flops with a clear input in one single package.

74LS74 Dual D-type Flip Flop



Other Popular D-type flip-flop ICs

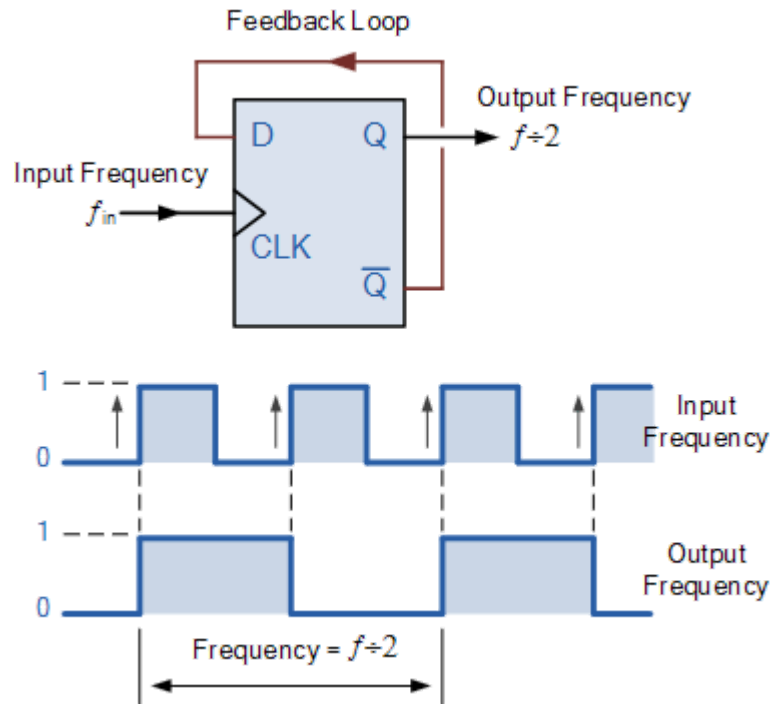
Device Number	Subfamily	Device Description
74LS74	LS TTL	Dual D-type Flip Flops with Preset and Clear
74LS175	LS TTL	Quad D-type Flip Flops with Clear
74LS273	LS TTL	Octal D-type Flip Flops with Clear
4013B	Standard CMOS	Dual type D Flip Flop
40174B	Standard CMOS	Hex D-type Flip Flop with Master Reset

Using The D-type Flip Flop For Frequency Division

One main use of a D-type flip flop is as a Frequency Divider. If the \overline{Q} output on a D-type flip-flop is connected directly to the D input giving the device closed loop “feedback”, successive clock pulses will make the bistable “toggle” once every two clock cycles.

In the counters tutorials we saw how the **Data Latch** can be used as a “Binary Divider”, or a “Frequency Divider” to produce a “divide-by-2” counter circuit, that is, the output has half the frequency of the clock pulses. By placing a feedback loop around the D-type flip flop another type of flip-flop circuit can be constructed called a **T-type flip-flop** or more commonly a T-type bistable, that can be used as a divide-by-two circuit in binary counters as shown below.

Divide-by-2 Counter

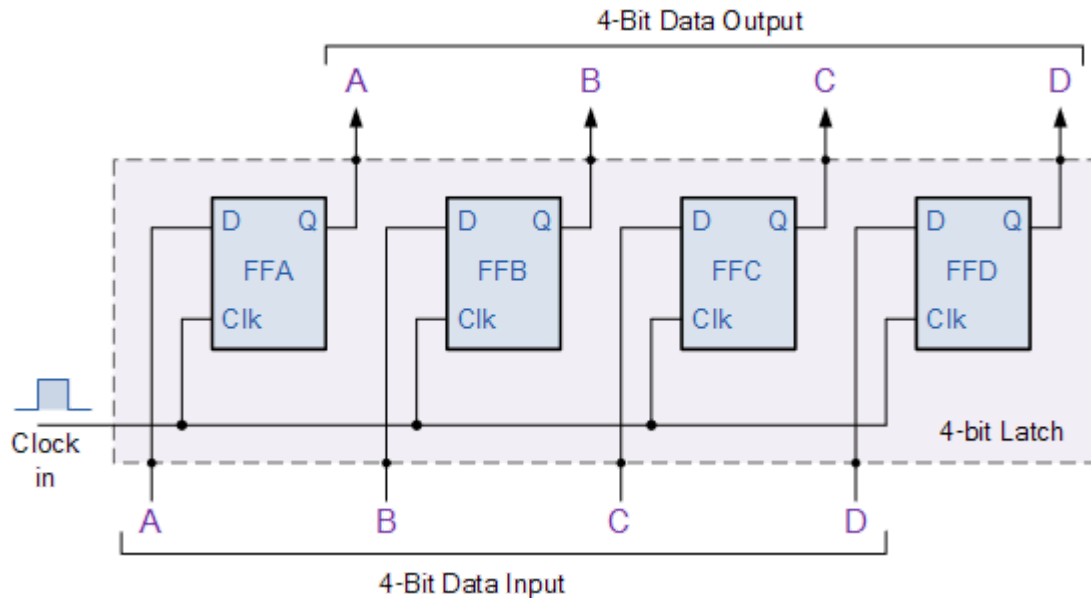


It can be seen from the frequency waveforms above, that by “feeding back” the output from \bar{Q} to the input terminal D, the output pulses at Q have a frequency that are exactly one half ($f/2$) that of the input clock frequency, (f_{IN}). In other words the circuit produces **frequency division** 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

As well as frequency division, another useful application of the D flip flop is as a **Data Latch**. 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.

4-bit Data Latch



Transparent Data Latch

The **Data Latch** is a very useful device in electronic and computer circuits. They can be designed to have very high output impedance at both outputs Q and its inverse or complement output \bar{Q} to reduce the impedance effect on the connecting circuit when used as a buffer, I/O port, bi-directional bus driver or even a display driver.

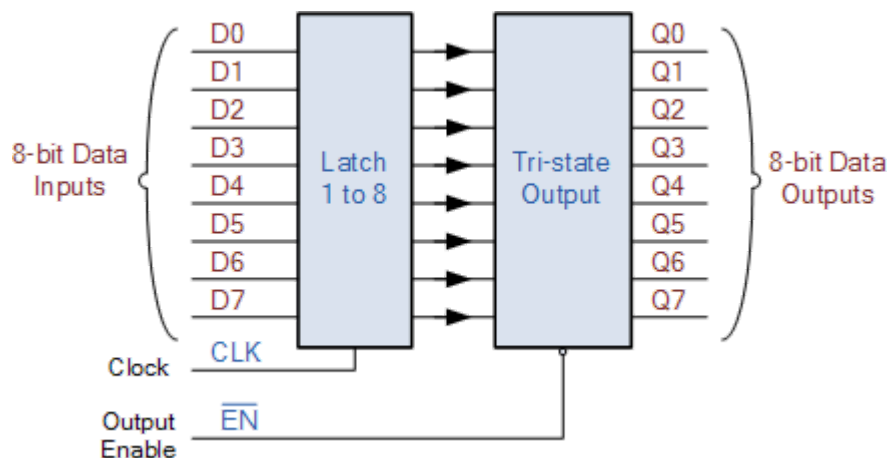
But a single “1-bit” data latch is not very practical to use on its own and instead commercially available IC’s incorporate 4, 8, 10, 16 or even 32 individual data latches into one single IC package, and one such IC device is the 74LS373 Octal D-type transparent latch.

The eight individual data latches or bistables of the 74LS373 are “transparent” D-type flip-flops, meaning that when the clock (CLK) input is HIGH at logic level “1”, (but can also be active low) the outputs at Q follows the data D inputs.

In this configuration the latch is said to be “open” and the path from \bar{D} input to \bar{Q} output appears to be “transparent” as the data flows through it unimpeded, hence the name transparent latch.

When the clock signal is LOW at logic level “0”, the latch “closes” and the output at \bar{Q} is latched at the last value of the data that was present before the clock signal changed and no longer changes in response to \bar{D} .

8-bit Data Latch



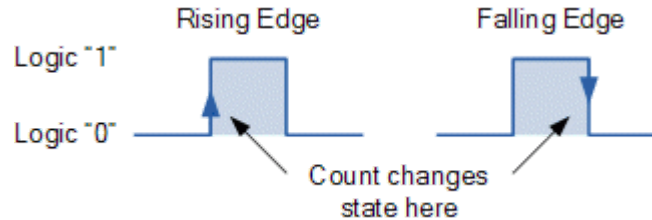
Functional diagram of the 74LS373 Octal Transparent Latch

The D-type Flip Flop Summary

The data or **D-type Flip Flop** can be built using a pair of back-to-back SR latches and connecting an inverter (NOT Gate) between the S and the R inputs to allow for a single D (data) input. The basic D flip flop circuit can be improved further by adding a second SR flip-flop to its output that is activated on the complementary clock signal to produce a “Master-Slave D flip-flop” device.

The difference between a D-type latch and a D-type flip-flop is that a latch does not have a clock signal to change state whereas a flip-flop always does. The D flip-flop is an edge triggered device which transfers input data to Q on clock rising or falling edge. Data Latches are level sensitive devices such as the data latch and the transparent latch.

In the next tutorial about **Sequential Logic Circuits**, we will look at connecting together data latches to produce another type of sequential logic circuit called a *Shift Register* that are used to convert parallel data into serial data and vice versa.



Synchronous Counter

Synchronous Counters are so called because the clock input of all the individual flip-flops within the counter are all clocked together at the same time by the same clock signal

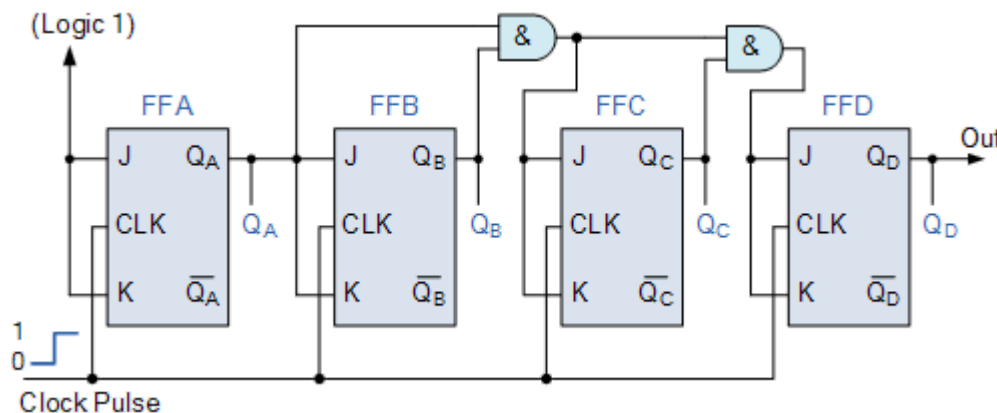
In the previous Asynchronous binary counter tutorial, we saw that the output of one counter stage is connected directly to the clock input of the next counter stage and so on along the chain.

The result of this is that the Asynchronous counter suffers from what is known as “Propagation Delay” in which the timing signal is delayed a fraction through each flip-flop.

However, with the **Synchronous Counter**, the external clock signal is connected to the clock input of EVERY individual flip-flop within the counter so that all of the flip-flops are clocked together simultaneously (in parallel) at the same time giving a fixed time relationship. In other words, changes in the output occur in “synchronisation” with the clock signal.

The result of this synchronisation is that all the individual output bits changing state at exactly the same time in response to the common clock signal with no ripple effect and therefore, no propagation delay.

Binary 4-bit Synchronous Up Counter



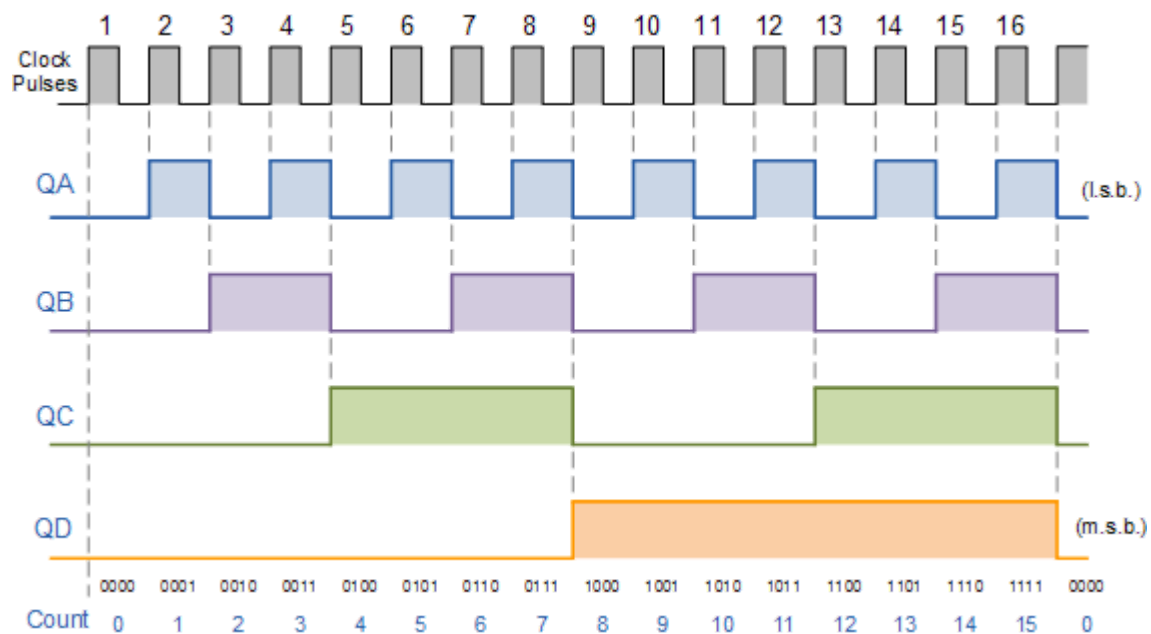
It can be seen above, that the external clock pulses (pulses to be counted) are fed directly to each of the **J-K flip-flops** in the counter chain and that both the J and K inputs are all tied together in toggle mode, but only in the first flip-flop, flip-flop FFA (LSB) are they connected HIGH, logic "1" allowing the flip-flop to toggle on every clock pulse. Then the synchronous counter follows a predetermined sequence of states in response to the common clock signal, advancing one state for each pulse.

The J and K inputs of flip-flop FFB are connected directly to the output Q_A of flip-flop FFA, but the J and K inputs of flip-flops FFC and FFD are driven from separate AND gates which are also supplied with signals from the input and output of the previous stage. These additional AND gates generate the required logic for the JK inputs of the next stage.

If we enable each JK flip-flop to toggle based on whether or not all preceding flip-flop outputs (Q) are "HIGH" we can obtain the same counting sequence as with the asynchronous circuit but without the ripple effect, since each flip-flop in this circuit will be clocked at exactly the same time.

Then as there is no inherent propagation delay in synchronous counters, because all the counter stages are triggered in parallel at the same time, the maximum operating frequency of this type of frequency counter is much higher than that for a similar asynchronous counter circuit.

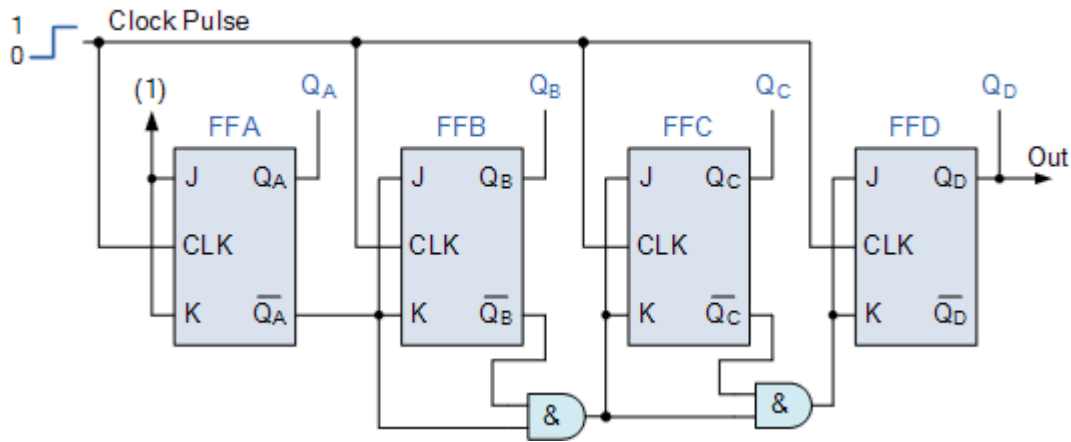
4-bit Synchronous Counter Waveform Timing Diagram



Because this 4-bit synchronous counter counts sequentially on every clock pulse the resulting outputs count upwards from 0 (0000) to 15 (1111). Therefore, this type of counter is also known as a **4-bit Synchronous Up Counter**.

However, we can easily construct a **4-bit Synchronous Down Counter** by connecting the AND gates to the \overline{Q} output of the flip-flops as shown to produce a waveform timing diagram the reverse of the above. Here the counter starts with all of its outputs HIGH (1111) and it counts down on the application of each clock pulse to zero, (0000) before repeating again.

Binary 4-bit Synchronous Down Counter

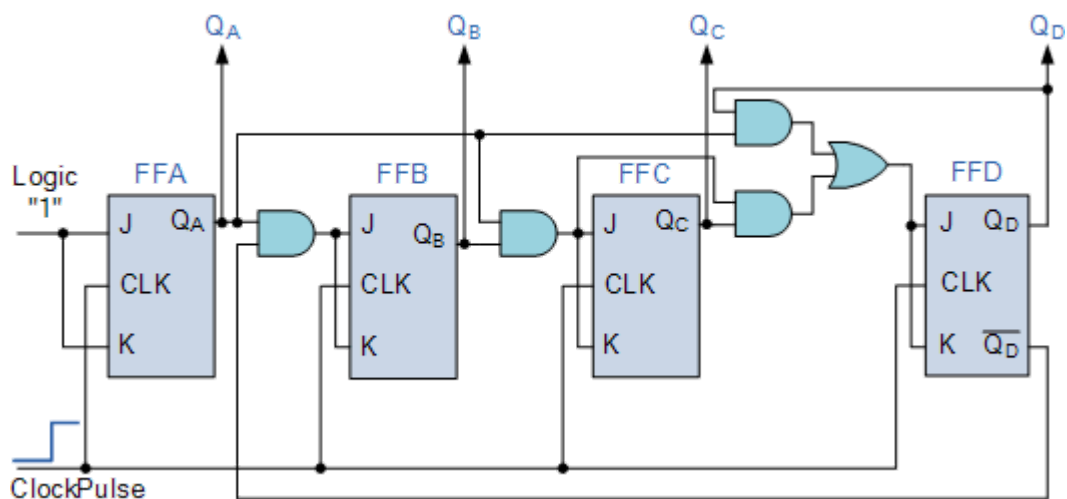


As synchronous counters are formed by connecting flip-flops together and any number of flip-flops can be connected or “cascaded” together to form a “divide-by-n” binary counter, the modulo’s or “MOD” number still applies as it does for asynchronous counters so a Decade counter or BCD counter with counts from 0 to $2^n - 1$ can be built along with truncated sequences. All we need to increase the MOD count of an up or down synchronous counter is an additional flip-flop and ANDgate across it.

Decade 4-bit Synchronous Counter

A 4-bit decade synchronous counter can also be built using synchronous binary counters to produce a count sequence from 0 to 9. A standard binary counter can be converted to a decade (decimal 10) counter with the aid of some additional logic to implement the desired state sequence. After reaching the count of “1001”, the counter recycles back to “0000”. We now have a decade or **Modulo-10** counter.

Decade 4-bit Synchronous Counter



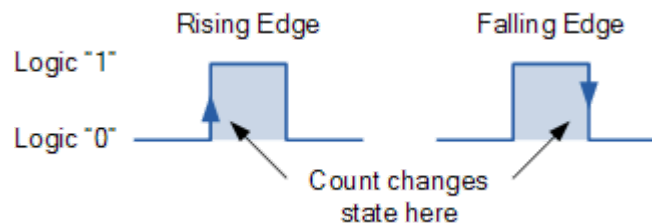
The additional AND gates detect when the counting sequence reaches “1001”, (Binary 10) and causes flip-flop FF3 to toggle on the next clock pulse. Flip-flop FF0 toggles on every clock pulse. Thus, the count is reset and starts over again at “0000” producing a synchronous decade counter.

We could quite easily re-arrange the additional AND gates in the above counter circuit to produce other count numbers such as a Mod-12 counter which counts 12 states from "0000" to "1011" (0 to 11) and then repeats making them suitable for clocks, etc.

Triggering A Synchronous Counter

Synchronous Counters use edge-triggered flip-flops that change states on either the "positive-edge" (rising edge) or the "negative-edge" (falling edge) of the clock pulse on the control input resulting in one single count when the clock input changes state.

Generally, synchronous counters count on the rising-edge which is the low to high transition of the clock signal and asynchronous ripple counters count on the falling-edge which is the high to low transition of the clock signal.



It may seem unusual that ripple counters use the falling-edge of the clock cycle to change state, but this makes it easier to link counters together because the most significant bit (MSB) of one counter can drive the clock input of the next.

This works because the next bit must change state when the previous bit changes from high to low – the point at which a carry must occur to the next bit. Synchronous counters usually have a carry-out and a carry-in pin for linking counters together without introducing any propagation delays.

Synchronous Counter Summary

Then to summarise some of the main points about Synchronous Counters:

Synchronous Counters can be made from Toggle or D-type flip-flops.

Synchronous counters are easier to design than asynchronous counters.

They are called synchronous counters because the clock input of the flip-flops are all clocked together at the same time with the same clock signal.

Due to this common clock pulse all output states switch or change simultaneously.

With all clock inputs wired together there is no inherent propagation delay.

Synchronous counters are sometimes called parallel counters as the clock is fed in parallel to all flip-flops.

The inherent memory circuit keeps track of the counters present state.

The count sequence is controlled using logic gates.

Overall faster operation may be achieved compared to Asynchronous counters.

In the next tutorial about Counters we will look at bi-directional counters which are able to count in both an upward direct and a downward direction.