# Latches and flip-flops

In the same way that [gates](http://www.labri.fr/perso/strandh/Teaching/AMP/Common/Strandh-Tutorial/gates.html) are the building blocks of [combinatorial circuits](http://www.labri.fr/perso/strandh/Teaching/AMP/Common/Strandh-Tutorial/combinatorial.html), latches and flip-flops are the building blocks of sequential circuits.

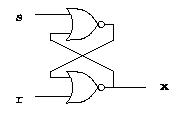
While gates had to be built directly from transistors, latches can be built from gates, and flip-flops can be built from latches. This fact will make it somewhat easier to understand latches and flip-flops.

Both latches and flip-flops are circuit elements whose output depends not only on the current inputs, but also on previous inputs and outputs. The difference between a latch and a flip-flop is that a latch does not have a clock signal, whereas a flip-flop always does.

## Latches

How can we make a circuit out of gates that is not combinatorial? The answer is feed-back, which means that we create loops in the circuit diagrams so that output values depend, indirectly, on themselves. If such feed-back is positive then the circuit tends to have stable states, and if it is negative the circuit will tend to oscillate.

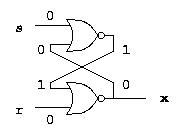
A latch has positive feedback. Here is an example of a simple latch:



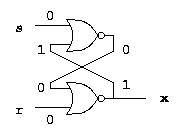
This latch is called SR-latch, which stands for set and reset.

It is not practical to use the methods that we have used to describe combinatorial circuits to describe the behavior of the SR-latch. Later, we will show a method for describing flip-flops and clocked sequential circuits. For now, we just rely on our intuition to describe how latches work.

The SR-latch is meant to have at most one of its inputs equal to 1 at any time. When both of its inputs are 0 it has two different stable states possible. Either x is 0, in which case we have the following signal values:

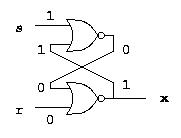


or else x is 1, in which case we have the following signal values:



The actual value depends on the history of input values as we will show next.

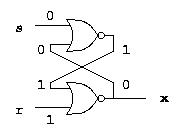
Now suppose that s is 1 (and therefore r is 0 since we allow at most one input to be 1 at any time). We get the following signal values:



The 1 on the s input makes sure the output of the upper nor-gate is 0, and the two 0s on the input of the lower nor-gate make sure the x output is 1.

Now suppose the s input goes from 1 to 0, while the r input remains at 0. The second input of the upper nor-gate is 1, so the transition from 1 to 0 of the s input, does not make any difference. The x output remains at 1. In this case, if the s and r inputs are both 0, there is only one possible stable state, the one that gives x the value 1.

Conversely, suppose that r is 1 (and therefore s is 0 since we allow at most one input to be 1 at any time). We get the following signal values:



The 1 on the r input makes sure the x output is 0, and the two 0s on the input of the upper nor-gate make sure the output of the upper nor-gate is 0.

Now suppose the r input goes from 1 to 0, while the s input remains at 0. The second input of the lower nor-gate is 1, so the transition from 1 to 0 of the r input, does not make any difference. The output of the upper nor-gate remains at 1. In this case, if the s and r inputs are both 0, there is only one possible stable state, the one that gives x the value 0.

From the discussion above, we conclude that the SR-latch is able to remember the last state of the inputs, in the sense that it remembers which of the two inputs, s or r, last had the value of 1.

When we need to draw an SR-latch, we use the following symbol:

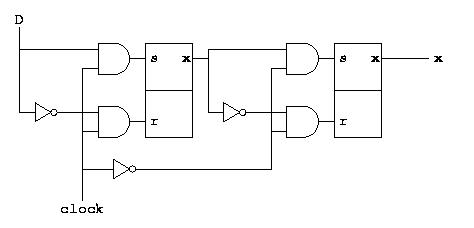


## Flip-flops

Latches are asynchronous, which means that the output changes very soon after the input changes. Most computers today, on the other hand, are synchronous, which means that the outputs of all the sequential circuits change simultaneously to the rhythm of a global clock signal.

A flip-flop is a synchronous version of the latch. To complicate the situation even more, there are several fundamental types of flip-flops. Here, we shall only consider a type called master-slave flip-flop.

In addition to the fundamental types of flip-flops, there are minor variations depending on the number of inputs and how they control the state of the flip-flop. Here, we shall only consider a very simple type of flip-flop called a D-flip-flop. A master-slave D-flip-flop is built from two SR-latches and some gates. Here is the circuit diagram:



The leftmost SR-latch is called the master and the rightmost is called the slave.

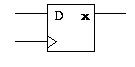
Let us first consider what happens when the clock signal is 1. In this case, the two and-gates in front of the input of the master are open, i.e., they let the value of the D-input through to the s input of the master, and the inverse of the D input to the r input of the master. Thus, the value of the D input will go straight trough the master to the x output of the master. But the two and-gates of the slave re closed, i.e., their outputs are always 0, so the slave keeps its old value.

When instead the clock signal is 0, the reverse is true, i.e., the and-gates at the input of the master are closed, whereas the ones at the input of the slave are open. In this case, the flip-flop is completely insensitive to changes in input.

Now, let us consider what happens when the clock goes from 1 to 0. For this to work, we have to assume that the input remains the same during a brief period from right before to right after the clock signal changes. The first thing that happens is that the and-gates at the input of the master turn off, i.e., they become insensitive to further changes in input. The value of the x output of the master is now the value of the D input right before the clock started changing. A brief moment later, the clock signal transition has traversed the inverter and reaches the and-gates of the slave. These gates open, allowing the x output of the master to be propagated to the x value of the slave. The x value of the slave, and therefore that of the entire flip-flop now contains the value of the D input right before the clock started changing. We can say that the clock transition copied the input to the output of the flip-flop. But at no point in time is there a direct path from input to output. The output changes only as a result of clock transitions from 1 to 0.

Finally, let us see what happens when the clock goes from 0 to 1. First, the and-gates of the master open, letting the value of the D input into the master. By the time the D value reaches the master, the clock signal transition reaches the and-gates of the slave, and turns them off before the possibly modified output of the master reaches the slave. Thus, the slave keeps its old value. From the outside, nothing seems to happen, since the output does not change. From now on, however, the master is open to changes in the input.

Here is the symbol we use for D-flip-flops:



The little triangle for the clock input indicates that this input is sensitive only to transitions as opposed to levels as described in the previous paragraph. Sometimes we do not draw the clock input at all when it is understood that it is there. Clock signals are boring since they are all just connected to each other. There is therefore little use to draw them all, and thereby clutter the diagram unnecessarily.

## Summary

We have shown how to build a D-flip-flop. It copies its input to its output as a result of a clock signal transition from 1 to 0. The value copied is the value the input has immediately before the clock transition. Most of the clock period, the D-flip-flop is insensitive to changes in the input.

We shall use this key characteristic of the D-flip-flop to build synchronous sequential circuits.

2

A **latch** (called **sneck** in Northern England and Scotland) is a type of mechanical fastener that is used to join two (or more) objects or surfaces together while allowing for the regular or eventual separation of the objects or surfaces. A latch typically engages another piece of hardware on the other mounting surface. Depending upon the type and design of the latch, this engaged bit of hardware may be known as a [*keeper* or *strike*](https://en.wikipedia.org/wiki/Strike_plate). Note that a latch is not the same as the [locking mechanism](https://en.wikipedia.org/wiki/Lock_%28security_device%29) of a door or window, although often they are found together in the same product.

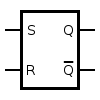
Latches range in complexity from flexible one-piece flat springs of metal or plastic, such as are used to keep blow molded plastic power tool cases closed, to multi-point cammed latches used to keep large doors closed.

A latch is an example of a bistable multivibrator, that is, a device with exactly two stable states. These states are high-output and low-output. A latch has a feedback path, so information can be retained by the device. Therefore latches can be memory devices, and can store one bit of data for as long as the device is powered. As the name suggests, latches are used to "latch onto" information and hold in place. Latches are very similar to [flip-flops](https://en.wikibooks.org/wiki/Digital_Circuits/Flip-Flops), but are not synchronous devices, and do not operate on clock edges as flip-flops do.

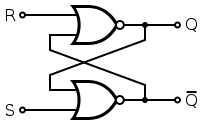
**Contents**

* [1 SR latch](https://en.wikibooks.org/wiki/Digital_Circuits/Latches#SR_latch)
  + [1.1 Gated SR latch](https://en.wikibooks.org/wiki/Digital_Circuits/Latches#Gated_SR_latch)
* [2 D latch](https://en.wikibooks.org/wiki/Digital_Circuits/Latches#D_latch)
* [3 See also](https://en.wikibooks.org/wiki/Digital_Circuits/Latches#See_also)
* [4 Further reading](https://en.wikibooks.org/wiki/Digital_Circuits/Latches#Further_reading)

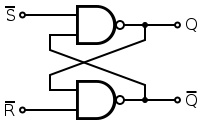
**SR latch**

[](https://commons.wikimedia.org/wiki/File:SR_%28NAND%29_Flip-flop.svg)

Circuit symbol for an SR latch.

[](https://commons.wikimedia.org/wiki/File:RS_Flip-flop_%28NOR%29.svg)

An SR latch made from two NOR gates.

[](https://commons.wikimedia.org/wiki/File:SR_Flip-flop_Diagram.svg)

An SR latch made from two NAND gates.

An **SR latch** (Set/Reset) is an asynchronous device: it works independently of control signals and relies only on the state of the S and R inputs. In the image we can see that an SR latch can be created with two NOR gates that have a cross-feedback loop. SR latches can also be made from NAND gates, but the inputs are swapped and negated. In this case, it is sometimes called an **SR latch**.

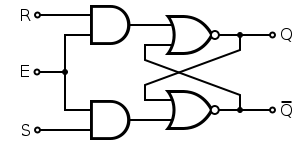
When a high is applied to the *Set* line of an SR latch, the *Q* output goes high (and Q low). The feedback mechanism, however, means that the *Q* output will remain high, even when the *S* input goes low again. This is how the latch serves as a memory device. Conversely, a high input on the *Reset* line will drive the *Q* output low (and *Q* high), effectively resetting the latch's "memory". When both inputs are low, the latch "latches" – it remains in its previously set or reset state.

When both inputs are high at once, however, there is a problem: it is being told to simultaneously produce a high *Q* and a low *Q*. This produces a "race condition" within the circuit - whichever flip flop succeeds in changing first will feedback to the other and assert itself. Ideally, both gates are identical and this is "metastable", and the device will be in an undefined state for an indefinite period. In real life, due to manufacturing methods, one gate will always win, but it's impossible to tell which it will be for a particular device from an assembly line. The state of *S* = *R* = 1 is therefore "illegal" and should never be entered.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S** | **R** |  | **Q** | **Q** |
| 0 | 0 | **Latched** | |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | Metastable | |

When the device is powered up, a similar condition occurs, because both outputs, *Q* and Q, are low. Again, the device will quickly exit the metastable state due to differences between the two gates, but it's impossible to predict which of *Q* and *Q* will end up high. To avoid spurious actions, you should always set SR flip-flops to a known initial state before using them - you must not assume that they will initialise to a low state.

**Gated SR latch**

[](https://commons.wikimedia.org/wiki/File:SR_%28Clocked%29_Flip-flop_Diagram.svg)

A Gated SR latch made from two NOR and two AND gates.

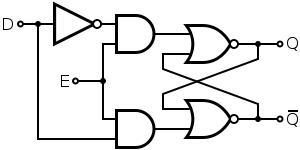
In some situations it may be desirable to dictate when the latch can and cannot latch. The **gated SR latch** is a simple extension of the SR latch which provides an *Enable* line which must be driven high before data can be latched. Even though a control line is now required, the SR latch is not synchronous, because the inputs can change the output even in the middle of an enable pulse.

When the *Enable* input is low, then the outputs from the AND gates must also be low, thus the *Q* and *Q* outputs remain latched to the previous data. Only when the enable input is high can the state of the latch change, as shown in the truth table. When the enable line is asserted, a gated SR latch is identical in operation to an SR latch.

The *Enable* line is sometimes a clock signal, but is usually a read or write [strobe](https://en.wikibooks.org/wiki/Digital_Circuits/Glossary#S).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Enable** | **S** | **R** |  | **Q** | **Q** |
| 0 | 0 | 0 | **Latched** | |
| 0 | 0 | 1 | **Latched** | |
| 0 | 1 | 0 | **Latched** | |
| 0 | 1 | 1 | **Latched** | |
| 1 | 0 | 0 | **Latched** | |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | Metastable | |

**D latch**

[](https://commons.wikimedia.org/wiki/File:D-type_Transparent_Latch_%28NOR%29.svg)

D latch: A gated SR latch extended to prevent invalid output conditions

The **D latch** (D for "data") or **transparent latch** is a simple extension of the gated SR latch that removes the possibility of invalid input states.

Since the gated SR latch allows us to latch the output without using the *S* or *R* inputs, we can remove one of the inputs by driving both the *Set* and *Reset* inputs with a complementary driver: we remove one input and automatically make it the inverse of the remaining input.

The D latch outputs the *D* input whenever the *Enable* line is high, otherwise the output is whatever the *D* input was when the *Enable* input was last high. This is why it is also known as a transparent latch - when *Enable* is asserted, the latch is said to be "transparent" - it signals propagate directly through it as if it isn't there.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Enable** | **D** |  | **Q** | **Q** |
| 0 | 0 | **Latched** | |
| 0 | 1 | **Latched** | |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

D latches are often used in I/O ports of integrated circuits and are available as discrete devices, often multiply packaged. An example is the 74HC75, part of the 7400 series of ICs, containing four separate D latches.