Technical Article

Circuit and Operation of a D Flip-Flop

5 days ago by Robert Keim

D-type flip-flops are fundamental digital memory components. Here, we examine their input-output behavior and internal circuitry.

Logic gates are the building blocks of all digital technology. However, the modern world's vast array of computational functionality wouldn't be possible with only combinational logic circuits. In a circuit composed entirely of combinational logic, the output depends only on the present state of the input. All of the signals move directly from input to output through interconnected logic gates.

To enable the advanced computation and signal processing we associate with today's digital technology, we need combinational logic *and* digital memory. In other words, we need sequential logic. While combinational logic allows us to build useful devices such as adders, multiplexers, and encoders, it's sequential logic that lets us move into the realm of microprocessors.

In a sequential logic circuit, the output depends on the *sequence*—both the present state and the history—of input signals. This allows for the creation of digital devices with memory, such as latches and flip-flops. However, though both of these circuit types function as digital memory devices, there's an important difference between them:

- A latch is *level-triggered*, meaning it's responsive to input signals when an "enable" input is active—during the logic-high pulse of a clock signal, for example.
- A flip-flop is *edge-triggered*, meaning that it's responsive to input signals when an "enable" input changes state—for example, at the rising edge of a clock signal.

In practical applications, it's usually more desirable to have a memory device that's sensitive to input states only at the moment of a particular event. For that reason, flip-flops are much more widely used.

The D flip-flop is, in my view, the most important flip-flop—I would even go so far as to call it the most important digital memory subcircuit. In this article, we'll learn about its electrical behavior and internal structure.

What is a D Flip-Flop?

Named for its single data input, the D flip-flop does exactly what a memory cell needs to do—it stores the input logic level as an output voltage at the moment of the active transition of a control signal, while at all other times simply retaining the output voltage.

Figure 1 shows a basic D flip-flop. As you can see, it has four terminals:

1. A data input (D).
2. A clock or control input (C).
3. An output (Q).
4. An inverted version of the output ($\bar{Q}$).

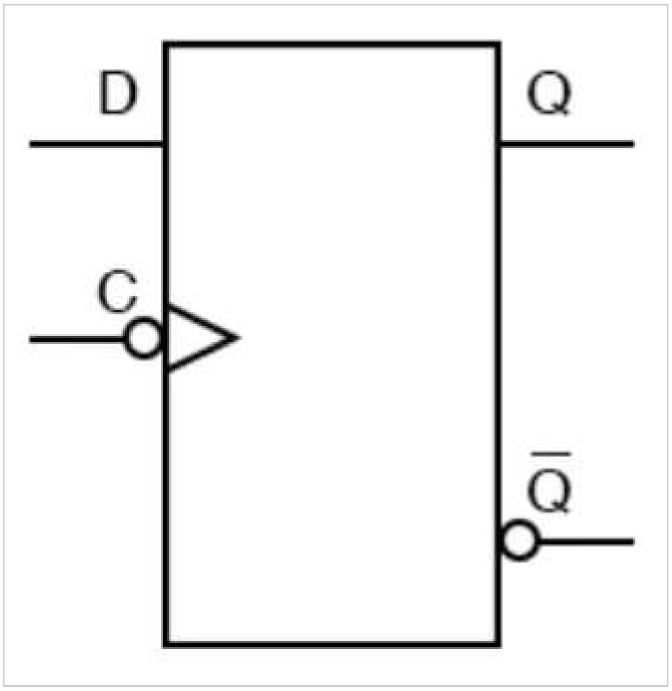

Figure 1. Circuit symbol for a basic D flip-flop. The terminals are a data input (D), a clock or control input (C), an output (Q), and an inverted version of the output ($\bar{Q}$). Image used courtesy of Tony R. Kuphaldt

Table 1 describes the operation of a D flip-flop that responds to the rising edge of a clock signal.

Table 1. Truth table for a D flip-flop.

Clock	**D**	**Q**
Rising edge	0	0
Rising edge	1	1
Falling edge	X	Q_{last}

The X in the falling-edge row means that the logic level of *D* is irrelevant. *Q* is only affected by the input state at the rising edge. On the rising edges of the clock signal, the data input is sampled and passed to the *Q* output. We can see this in the time domain by reviewing the timing diagram in Figure 2.

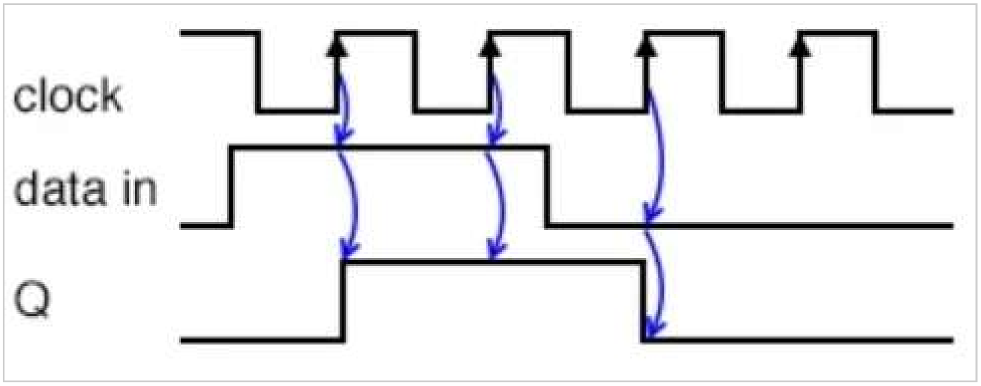

Figure 2. Timing diagram of a D flip-flop. Image (modified) used courtesy of Tony R. Kuphaldt

Eight D flip-flops controlled by the rising or falling edge of the same clock signal can function as a one-byte (8-bit) register. When combined with decoding circuitry, a collection of one-byte registers functions as a memory bank.

To understand the D flip-flop's internal logic-gate structure, let’s examine the following functional blocks:

- The S-R latch.
- The D latch.
- The pulse detector.

The S-R Latch

The set-reset (S-R) latch is the foundational subcircuit for sequential logic. This latch, which achieves memory through feedback, can be built from NOR gates or NAND gates. Figure 3 shows an S-R latch implemented as two cross-coupled NOR gates.

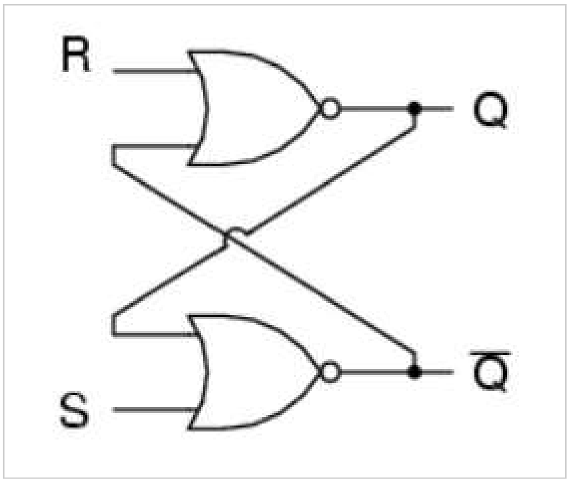

Figure 3. S-R latch implemented as two cross-coupled NOR gates. Image used courtesy of [Tony R. Kuphaldt]

Feeding the output signals back to input terminals allows the S-R latch to retain a previous output state. The truth table below specifies the latch’s input-output behavior.

Scroll to continue with content

Table 2. S-R latch truth table.

S	**R**	**Q**	**Q̄**
0	0	latch	latch
0	1	0	1
1	0	1	0
1	1	invalid	

To understand this behavior, consider the following sequence of events:

1. To start with, let's assume *S* is high and *R* is low. *Q* will therefore be high.
2. Now drive *S* to logic-low. Both inputs are now low, and *Q* is still high—it's "latched," meaning it retains its previous state.
3. Now drive *R* to logic-high. With *R* logic-high and *S* logic-low, *Q* will be low.
4. Now drive *R* to logic-low. Once again, both inputs are low, but this time *Q* is low.

The same input combination produces a different output state in Step 4 than in Step 2. This is because the circuit is influenced by both the present input state and the previous output state.

To make the S-R latch more useful, we need additional circuitry that allows us to determine when the latch will or won't respond to its input state. Figure 4 shows a "gated" S-R latch, meaning an S-R latch with an enable/disable signal. The control signal is labeled *E*, for "Enable."

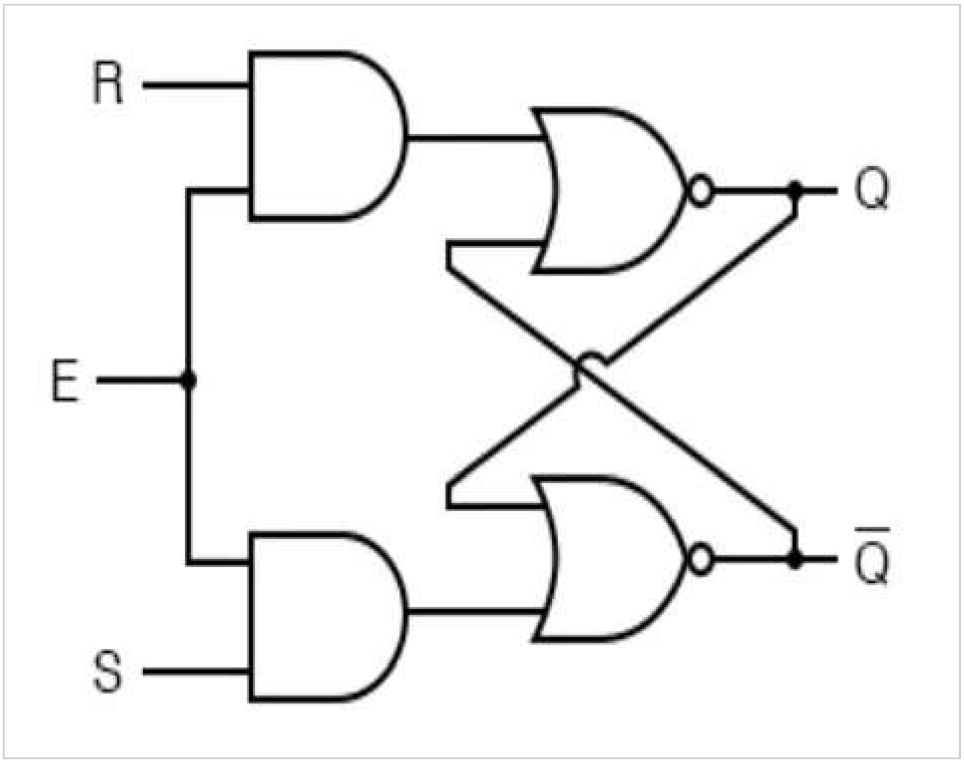

Figure 4. A gated S-R latch. Image used courtesy of Tony R. Kuphaldt

With this addition, we now have a level-triggered latch—when *E* is low, the latch acts as though *S* and *R* are low. In other words, a logic-low on *E* places the latch in its retain-the-previous-output mode, regardless of the true state of the *S* and *R* inputs.

The D Latch

Technically, we don't need both a set and reset input to store a single bit of memory. A memory cell that simply stores the logic level of one input signal makes more intuitive sense. That's why we have the D latch (Figure 5).

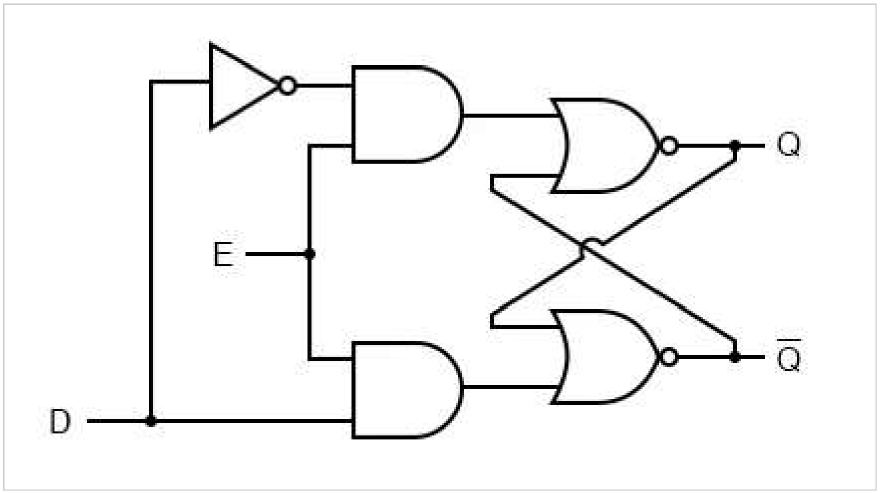

Figure 5. A D latch. Image used courtesy of [Tony R. Kuphaldt]

As the diagram shows, a D latch is simply a gated S-R latch in which the *S* input is renamed *D* and the *R* input is always the inverse of *D*. This arrangement makes the *S* = *R* = 1 (invalid) state and the *S* = *R* = 0 (latch) state impossible. As we see in Table 3, *Q* follows *D*, and latching mode is achieved only by means of the *E* signal.

Table 3. Truth table for a D latch.

E	D	Q	$\bar{Q}$
0	0	latch	latch
0	1	latch	latch
1	0	0	1
1	1	1	0

The Pulse Detector

To turn a level-triggered D latch into an edge-triggered D flip-flop, we need a pulse detector—a circuit that converts a wide pulse into a very narrow pulse aligned with the pulse's rising or falling edge. The diagram in Figure 6 depicts one way of implementing a pulse detector. Here, multiple inverters connected in series create a short delay. The AND gate uses the primary and delayed inputs to create a narrow pulse.

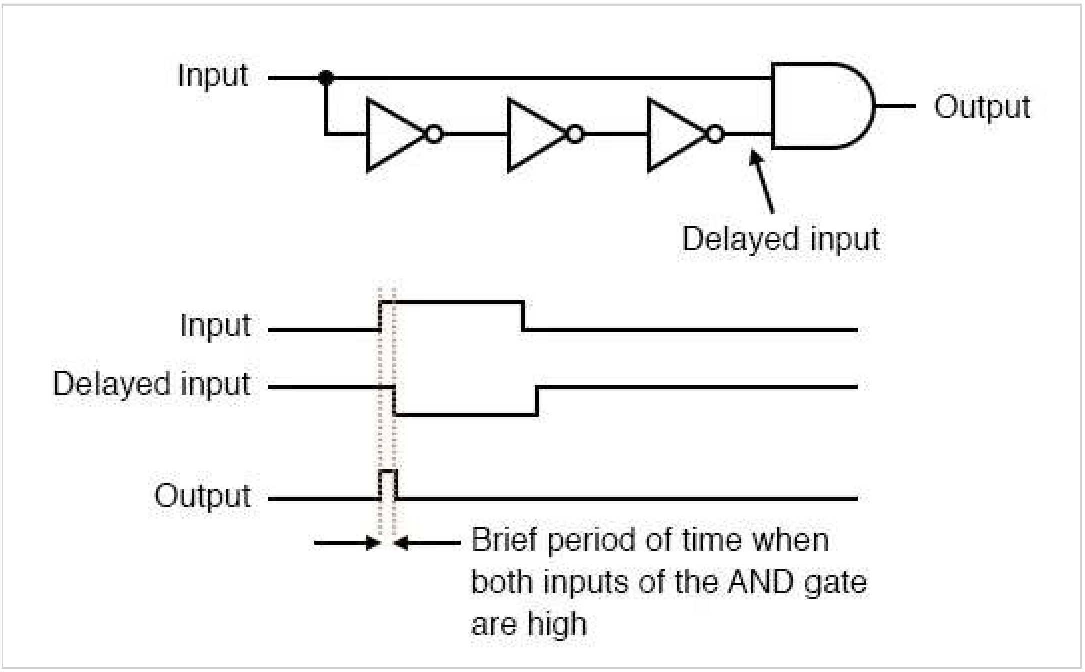
Input
Output
Delayed input
Input
Delayed input
Output
Brief period of time when both inputs of the AND gate are high

Figure 6. A pulse detector built using multiple inverters connected in series. Image used courtesy of [Tony R. Kuphaldt]

If we pass the D latch’s *E* signal through a pulse detector before feeding it to the AND gates, we’ll have a D flip-flop that acts as though it’s responsive to the inputs only at the moment of the *E* signal’s rising or falling edge. In reality, the flip-flop is still level-triggered, but now the level of interest is the active state of the pulse-detector output. This active state is always very brief.

Wrapping Up

The D flip-flop is an important building block for even the most advanced digital technologies. Before we finish up, I should point out that the structure described above isn’t the only method of creating one. For example, a rising-edge-triggered D flip-flop can be made from three S-R latches connected together. Either way, the circuit remains fairly straightforward in design and operation.

Featured image (modified) used courtesy of Adobe Stock