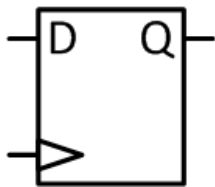


Reading: D Flip-Flops

This is a replacement reading for sections 3.4.1–3.4.2 of the book.

Until now, we've looked at **combinational** circuits – those whose output(s) are a function only of the current value of their inputs. However, many interesting circuits have output(s) that also depend on past input behavior. Circuits in this second category are called **sequential** circuits.

Building a circuit whose function depends on past input values requires storage elements that allow us to retain (store) information for later use. There are many different types of storage elements, and different reasons for using the different types. However, in this class we will only look at one common type of storage element: the D flip-flop, whose symbol is shown below.



The D flip-flop is itself a logic circuit, built from gates. The exact design of this structure is not important in the context of this class; you will learn more about that in ECE/CS 352. What you need to know and understand now is how the D flip-flop **behaves**.

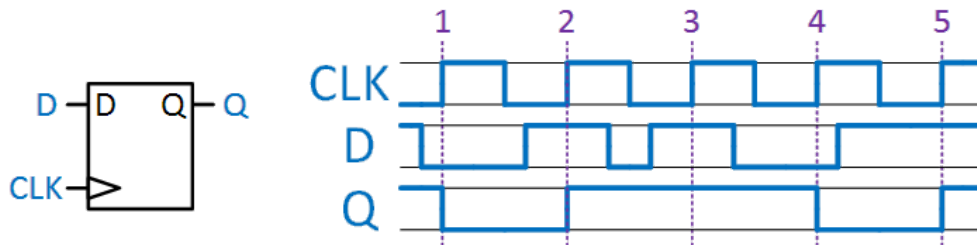
A D flip-flop can store a single bit of information. The currently-stored value appears at the flip-flop output, which is labeled **Q** on the flip-flop symbol. A value (either a 1 or a 0) appearing on the input labeled **D** in the symbol will periodically be stored into the flip-flop, overwriting the value that was previously stored. This only happens at certain times governed by the other input signal – the **clock**. The clock input is labeled with a triangle on the D flip-flop symbol.

The clock is a special signal that oscillates between 0 and 1 at a specific fixed rate called its **frequency**. An example clock signal is shown below. A "2 GHz" processor has a clock signal that has a frequency of 2 GHz, meaning that its clock period (one full oscillation) is $1 / (2 \times 10^9 \text{ Hz}) = 0.5 \times 10^{-9} \text{ s} = 0.5 \text{ ns}$.



In this class, we will assume that a flip-flop stores whatever value appears on the **D** input each time the clock signal value changes from 0 to 1. This transition is called the **positive edge** (or **rising edge**) of the clock signal because of how it is represented on the waveform (going from "low" to "high").

The important feature of the D flip-flop is that the value stored in it stays there, even if the **D** input changes, until the next positive edge allows the flip-flop to store a new value. The below image shows a flip-flop where the **D** input pin has been connected to a signal also named **D**, the clock input pin to a signal named **CLK**, and the **Q** output pin to a signal named **Q**. It also shows a waveform with input signals **D** and **CLK**, and the resulting output signal **Q**. The positive edge of the clock has been marked with vertical dashed lines, which have been labeled with the numbers 1–5 to make it easier to describe the diagram.



Signal **Q** (which is equal to the value stored in the flip-flop) is only able to change at the times that the clock transitions from 0 to 1 (the times marked by the vertical dashed lines). Specific things to notice about this waveform:

- At the transition #1, **D** is 0, so a 0 is written into the flip-flop, overwriting the 1 that had previously been stored there. The **Q** signal then updates to be equal to the newly-stored value.
- **D** then changes back to 1 before transition #2, but the new **D** value is not stored into the flip-flop until transition #2 (and thus **Q** does not change until then either).
- Although **D** becomes 0 between transition #2 and transition #3, it is back to 1 by the time transition #3 happens, so at transition #3 the flip-flop stores a 1 again, overwriting the previous 1 it had stored at transition #2. Notice that **Q** does not have to change to a different value every positive clock edge; rather, it changes to be equal to the current **D** value at every positive clock edge.

Again, for this course, do not worry about the design of the logic circuit inside of the flip-flop. That is a topic for a later course. However, you should know that it is constructed using logic gates, and you should know how it works according to the description above.