Slide Set 7

for ENEL 353 Fall 2019

Steve Norman, PhD, PEng

Electrical & Computer Engineering
Schulich School of Engineering
University of Calgary

Fall Term, 2019

Contents

Outline of Slide Set 7

Combinational versus *Sequential* Logic

This is review:

> The outputs of a *combinational* logic circuit depend only the **current** values of its inputs.
>
> The outputs of a *sequential* logic circuit depend on the **history** of its input values.

We've just seen that the above definition of combinational logic is very slightly untrue, due to very tiny delays.

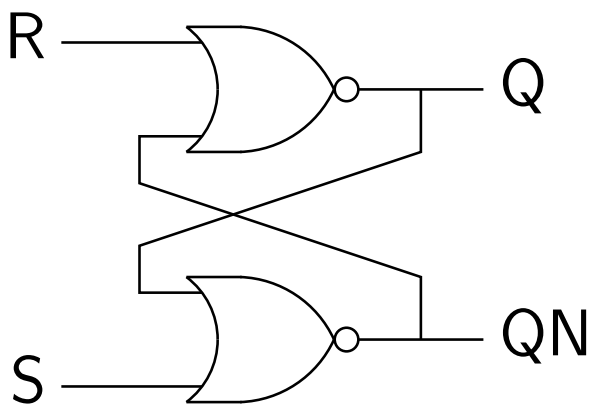
However, **sequential logic is totally different**. Outputs of sequential logic circuits may depend on the history of input values **indefinitely far back in the past**—minutes, hours, or days, not just picoseconds.

Outline of Slide Set 7

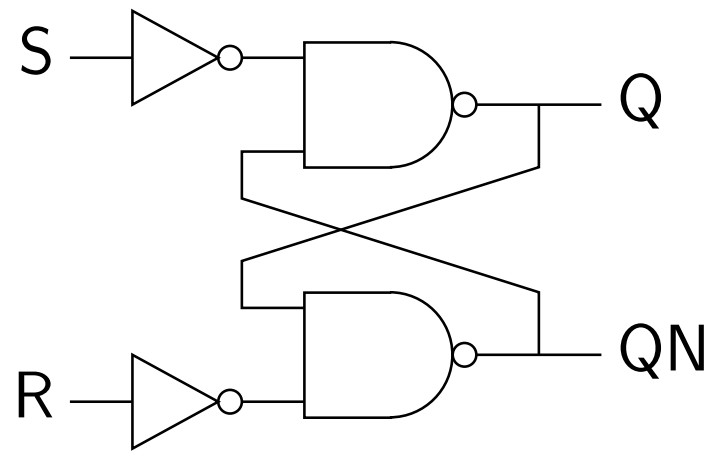
SR latches

Here are two ways to build an SR latch, perhaps the simplest sequential circuit element:

NOR-based



NAND-based

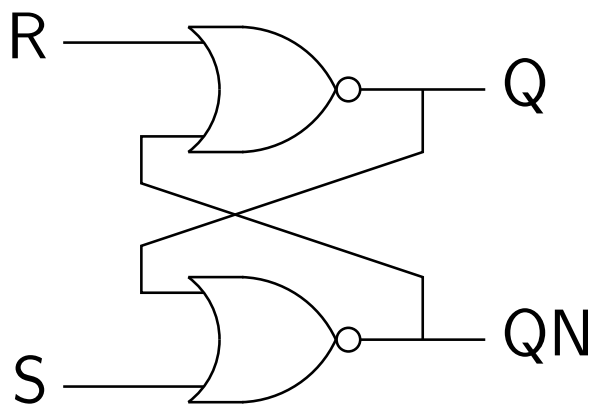


Notice that it's possible to wire together combinational devices in ways that produce sequential devices!

Harris & Harris use Q and $\overline{Q}$ as names of outputs, but I prefer Q and QN because—as we'll soon see—it's not always true that QN = NOT(Q).

Static analysis of the SR latch

We'll look at the NOR-based circuit. (Analysis of the NAND-based circuit is very similar.)



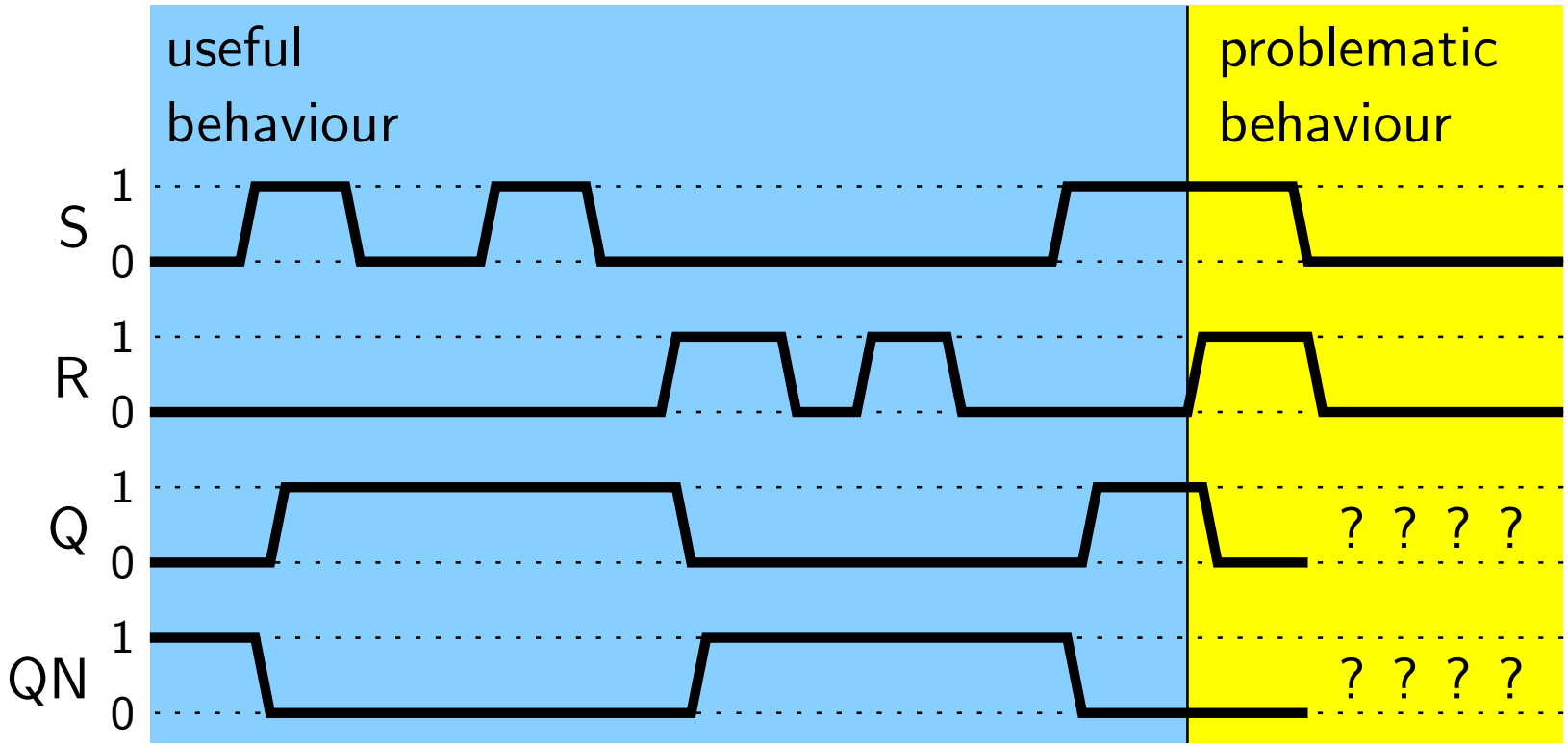
$$Q = \overline{(R + QN)}$$
$$QN = \overline{(S + Q)}$$

Q depends on QN, and QN depends on Q.

This is a system of two Boolean algebra equations in two unknowns!

For all four possible combinations of R and S, let's solve for Q and QN.

Dynamic behaviour of a NOR-based SR latch



A *pulse* on S or R is a transition from 0 to 1, followed later by a transition from 1 to 0.

Let's make some notes about useful and problematic behaviour of the SR latch.

The SR latch is an example of a bistable circuit

A *bistable* circuit is one that will sit in either one of two stable states.

We've just seen that an SR latch is bistable when $S = R = 0$: either $(Q, QN) = (0,1)$ or $(Q, QN) = (1,0)$.

It's important to understand that if there are no pulses on S or R, the state of an SR latch will **persist** as long as the circuit is powered up.

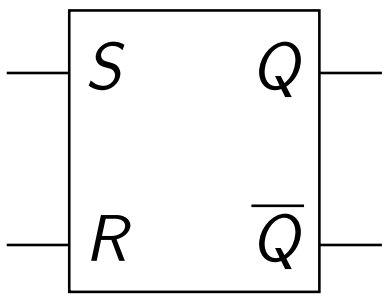
When $S = R = 0$, the state **will not** spontaneously flip between $(Q, QN) = (0,1)$ and $(Q, QN) = (1,0)$ (unless the latch is affected by **severe** electrical noise).

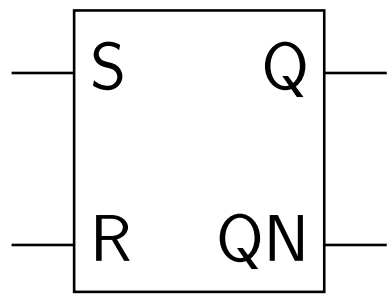
To understand **why** the state is stable when $S = R = 0$, you need to study the pull-up and pull-down networks of the gates that make up an SR latch. That is not an ENEL 353 topic.

Symbols for SR latches

It's less important to know what is going on inside an SR latch (NOR gates, NAND gates, inverters and/or other devices) than it is to know its behaviour as a "black box". (*Black box:* You can play with its inputs and observe its outputs, but you can't look inside it.)

Here are two symbols, one from our course textbook, and another from an author named Wakerly ...



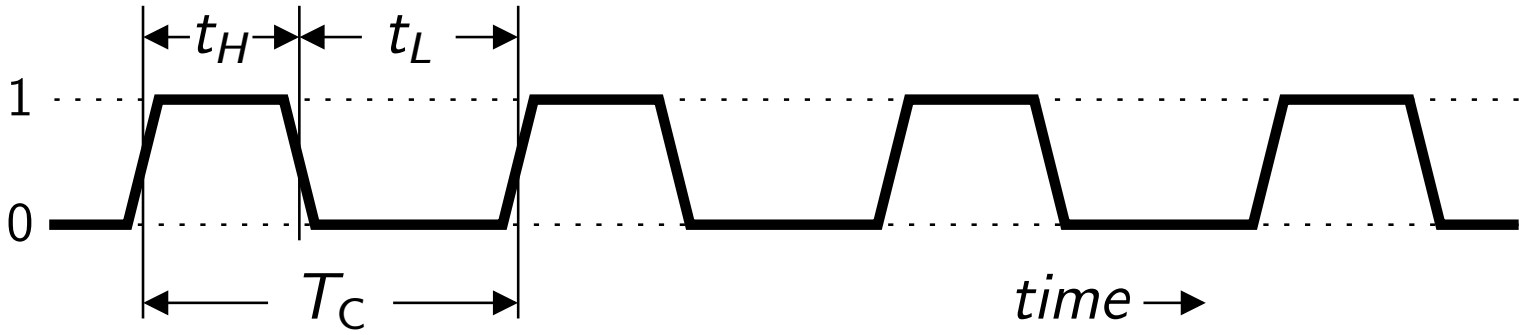


(Wakerly's *Digital Design* book is very good, but for a beginner, reading it may be somewhat like trying to drink from a firehose.)

Outline of Slide Set 7

Clock signals

A *clock signal* in a digital circuit is a periodic "square wave":



T_C is the *period* of the clock, also called the *cycle time*.

The *clock frequency* f_C is related to the period as $f_C = 1/T_C$.

If the frequency of a clock is 2.5 *GHz, what is its period?*

The *duty cycle* is defined as $(t_H/T_C) \times 100\%$. Usually $t_H = t_L = 0.5\,T_C$, so the duty cycle is 50%, but that's not true for all clock signals.

Clock signals and sequential logic systems

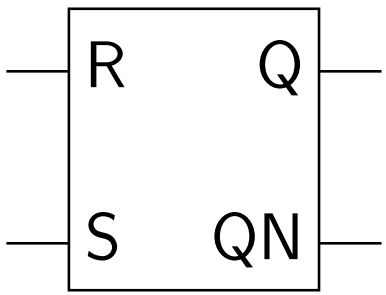
In the most common kind of sequential circuit, a **common clock signal** is supplied to all of the D latches and/or D flip-flops in the circuit.

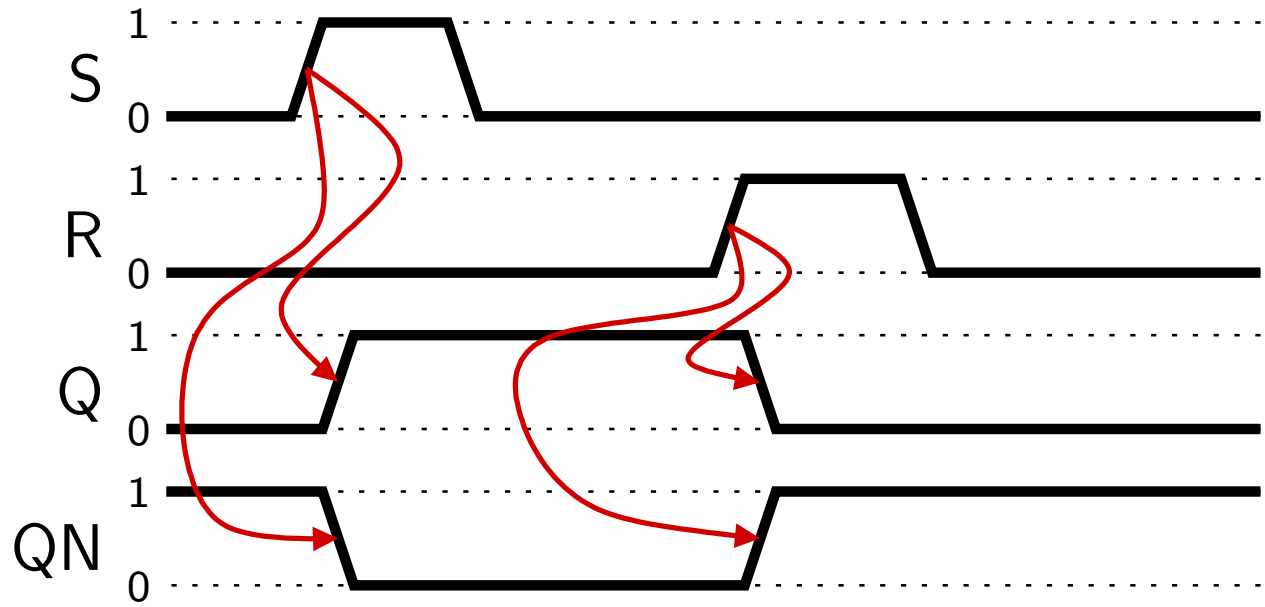
(D latches and D flip-flops are important sequential logic components that will be presented very soon.)

In digital integrated circuit design, distributing a common clock signal to all the latches and flip-flops in the circuit is just as important as making sure V_{DD} and ground are connected to all combinational and sequential elements.

Outline of Slide Set 7

Quick review of the SR latch, part 1



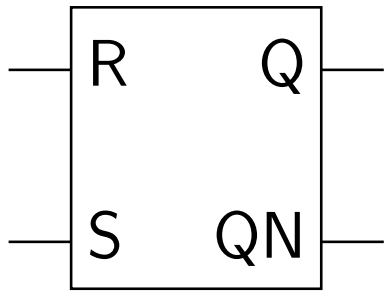


A **pulse** on S **sets** the state Q—drives it to 1.
A **pulse** on R **resets** the state Q—drives it to 0.

If there are no pulses on S or R, the state maintains its value as long as the circuit is powered up.

In normal operation, $QN = \overline{Q}$.

Quick review of the SR latch, part 2



Asserting S and R at the same time (in other words, making $S = R = 1$) should be avoided.

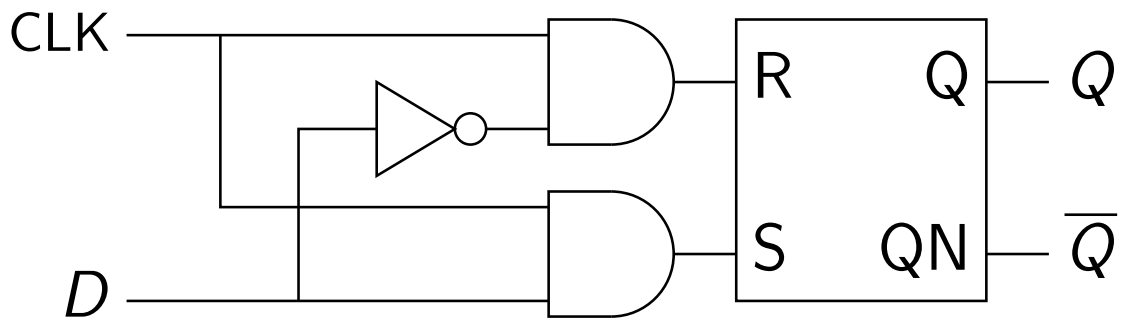
When $S = R = 1$, it's possible that $Q = QN$. For the NOR-based SR latch, we saw that when $S = R = 1$, $Q = QN = 0$. (For a NAND-based SR latch, when $S = R = 1$, it turns out that $Q = QN = 1$.)

Behaviour of an SR latch when S and R make $1 \rightarrow 0$ transitions at nearly the same time is **unpredictable**.

The D latch

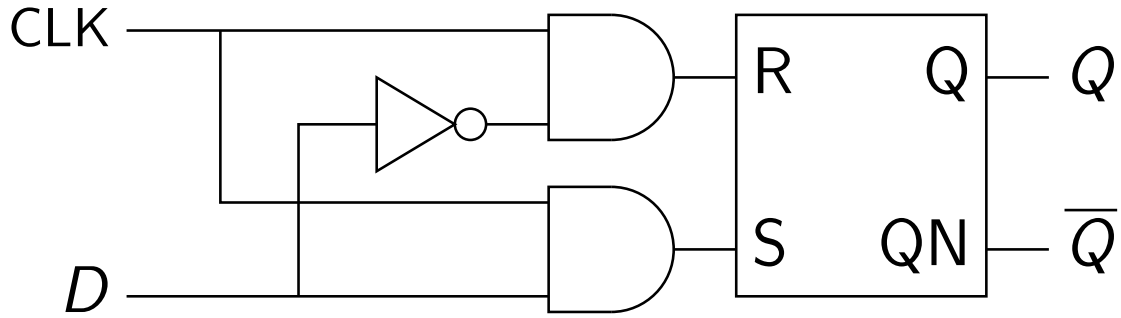
A *D latch* has two input wires. One of them is called *D*, for **data**. The other is usually called CLK, and is usually connected to a **clock signal**.

One way to make a D latch is with an SR latch ...

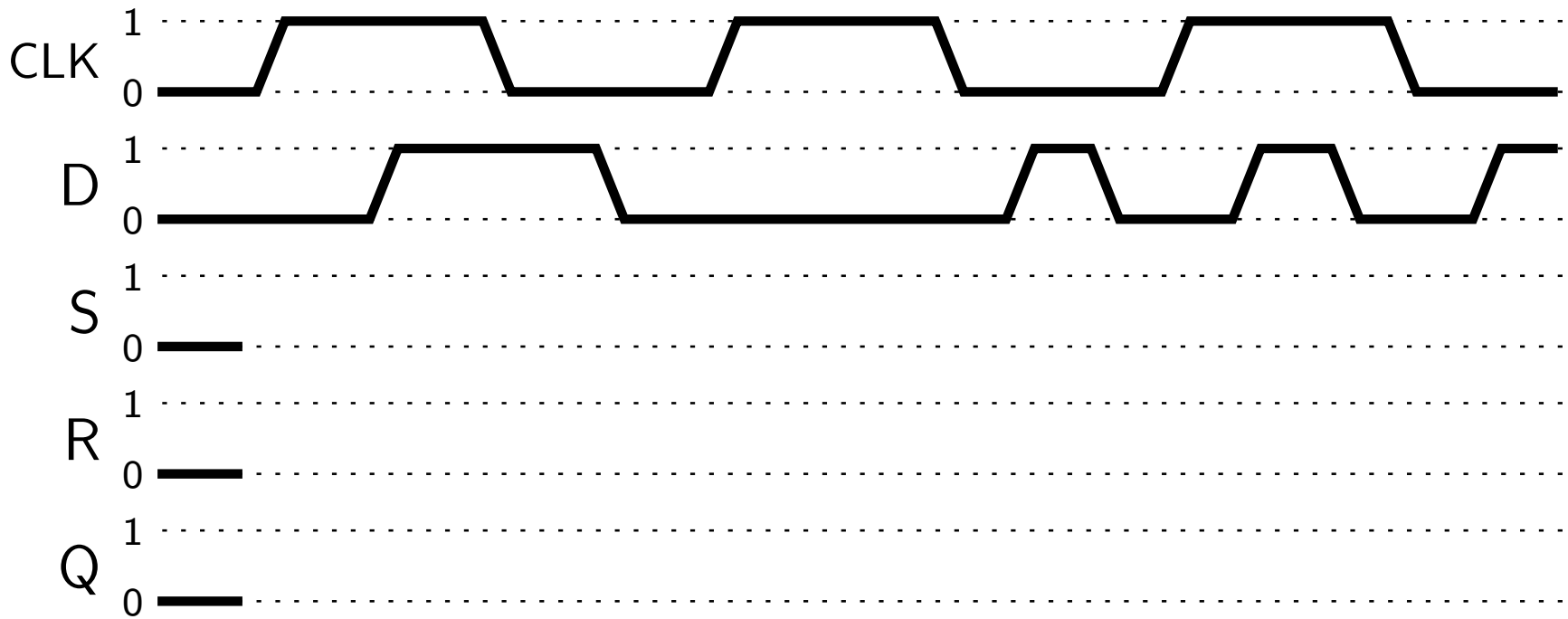


Why is it safe to label the D latch outputs as Q and $\overline{Q}$, rather than Q and QN as was done for the SR latch?

Behaviour of a D latch

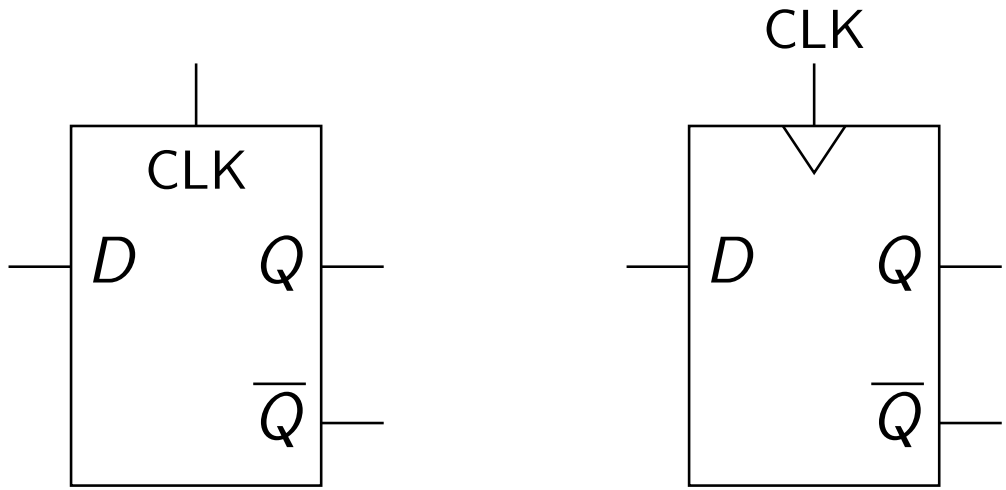


Let's complete the timing diagram below, then make some notes about D latch behaviour.



Symbols for D latches and D flip-flops

Left: D latch. Right: D flip-flop.



The symbols look very similar, but there is a really significant difference in behaviour!

We'll now move on to studying D flip-flops, which are **very important** sequential logic elements.

Outline of Slide Set 7

D Flip-Flops: Essential components in almost all sequential circuits!!!

In learning about **combinational** logic circuits, it would be impossible to make progress without knowing **exactly** what NOT, AND and OR gates do.

Similarly, it is impossible to understand most **sequential** circuits without knowing **exactly** what the basic behaviour of a **D flip-flop** is ... how the state of a D flip-flop changes in response to its input signals.

Clock edges

Transitions between logic levels in a clock signal are usually called *clock edges*.

A $0 \rightarrow 1$ transition is called a *rising edge* or a *positive edge*.

A $1 \rightarrow 0$ transition is called a *falling edge* or a *negative edge*.

Let's make a sketch of a clock signal and label the rising and falling edges.

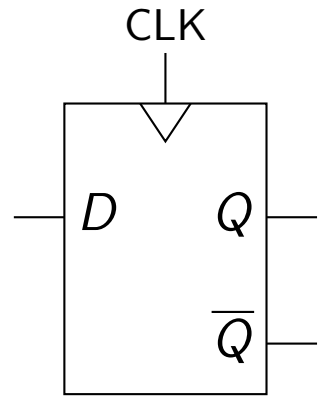
A good quote from your textbook

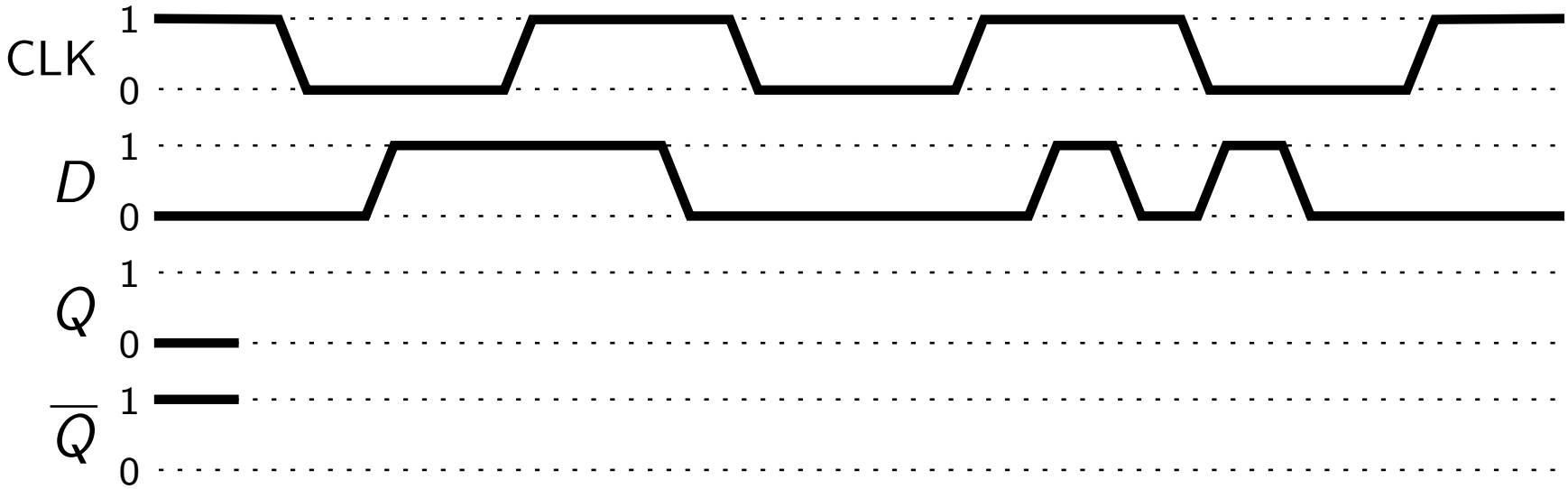
From page 114 of Harris and Harris:

> *A D flip-flop copies D to Q on the rising edge of the clock, and remembers its state at all other times.* Reread this definition until you have it memorized; one of the most common problems for beginning digital designers is to forget what a flip-flop does.

Learning D flip-flip behaviour by example

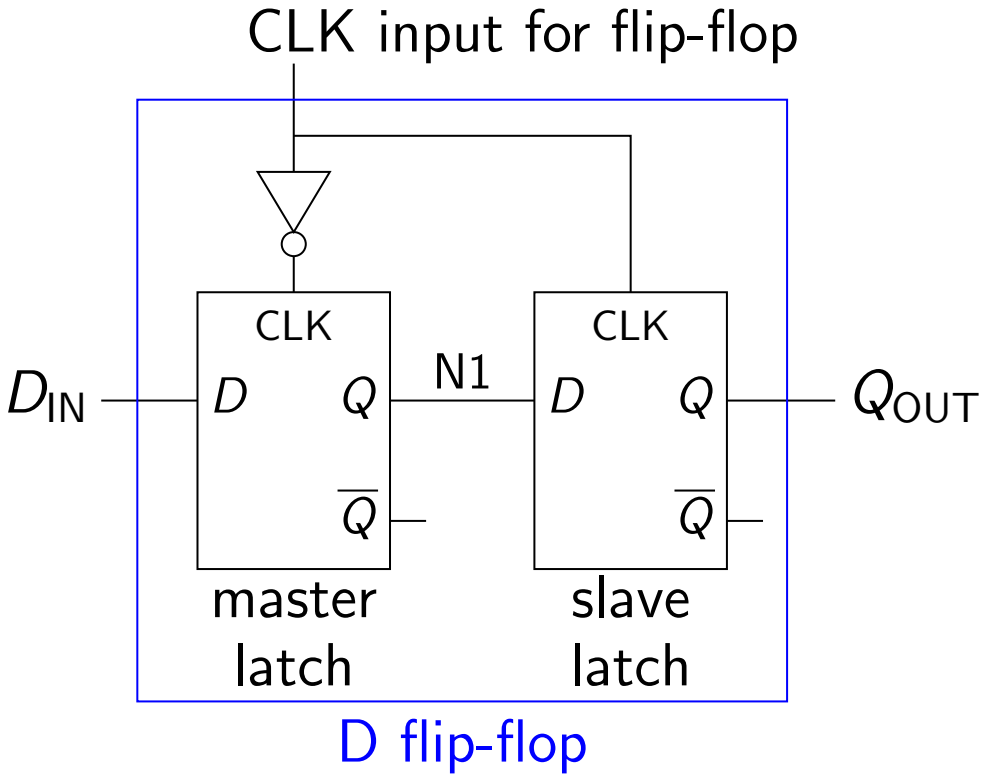
Let's complete the timing diagram.





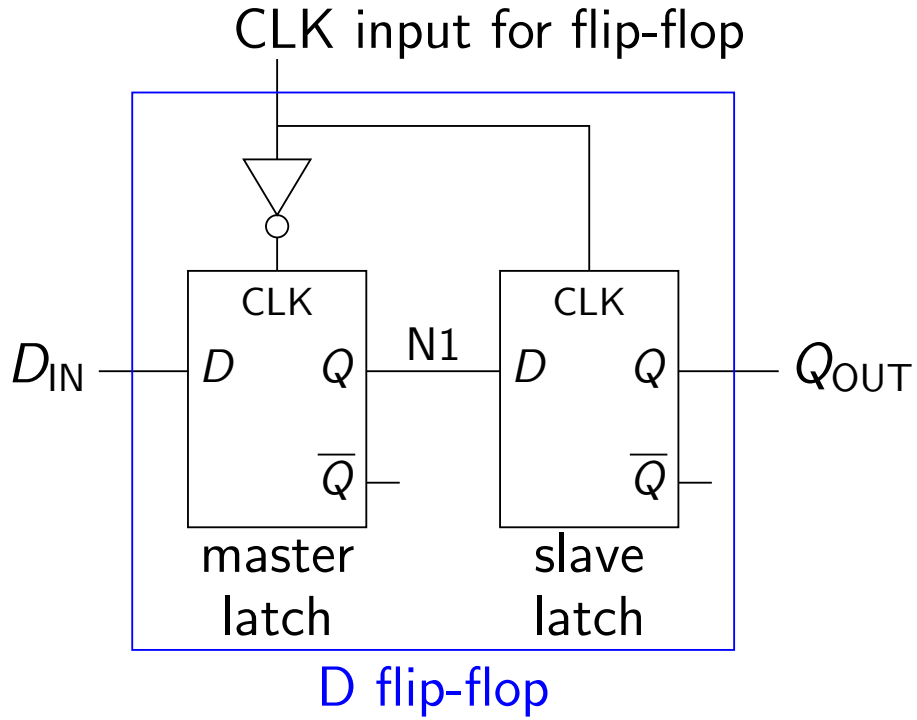
Outline of Slide Set 7

D flip-flop implementation using two D latches: "master-slave" configuration

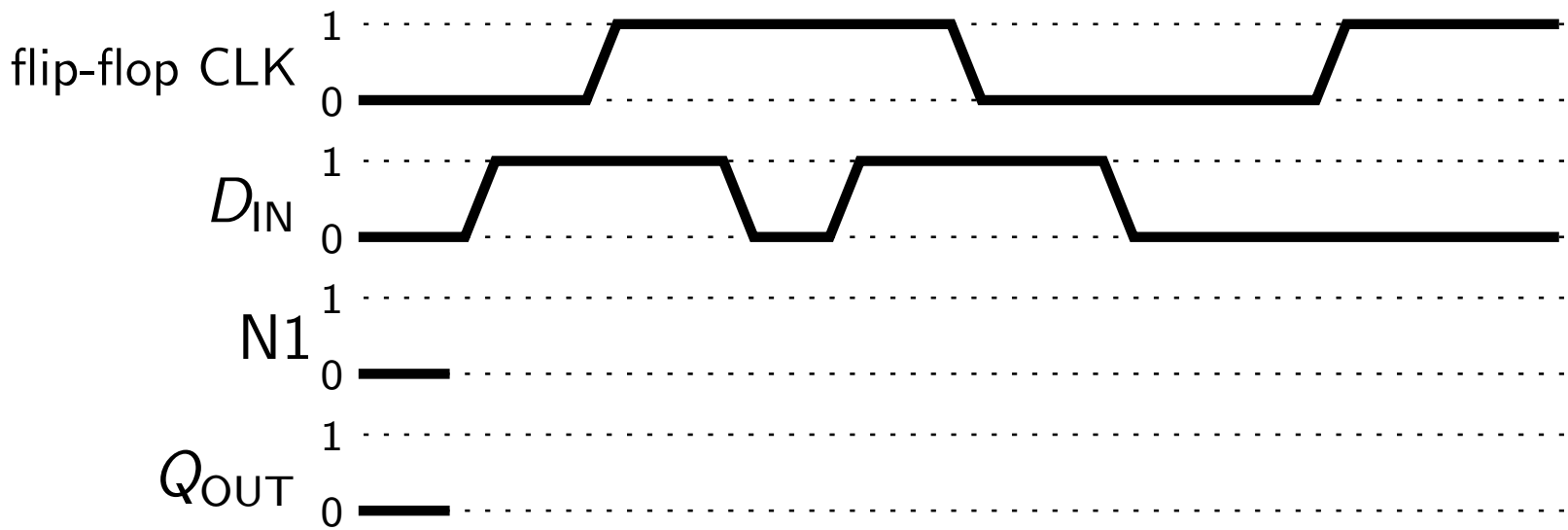


Your instructor thinks that **grabber-holder** would be a more descriptive (and less creepy) name than **master-slave** for this kind of D flip-flop design.

Let's make some notes on the jobs done by the master and slave latches in this circuit.



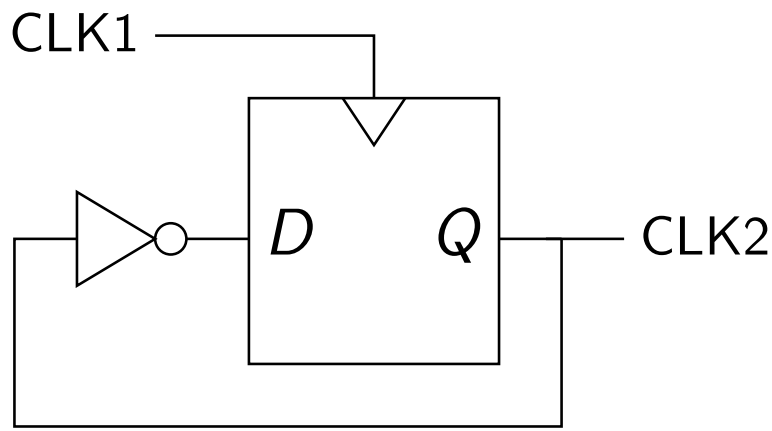
Let's see how this circuit works by completing this diagram . . .



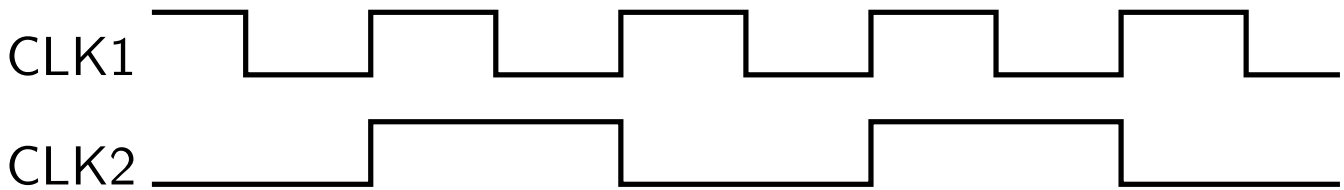
Outline of Slide Set 7

The clock divider

This simple and useful circuit can be built with a D flip-flop and an inverter. The output CLK2 is a clock signal with **half the frequency** of the input CLK1.



If the frequency of CLK1 is, say, 100 kHz, the signals will look like this on an oscilloscope . . .

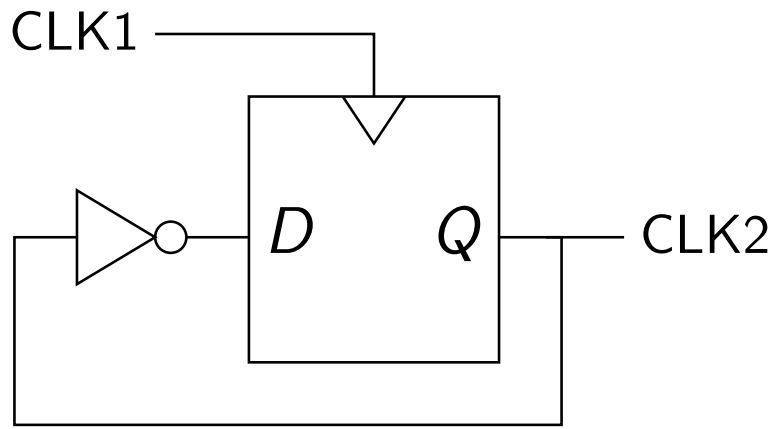


The basic behaviour of a D flip-flop explains why CLK2 is constant between rising edges of CLK1. **But what is going on at those rising edges?** According to the inverter, $D = \overline{Q}$, but according to the flip-flop, $Q = D$. It **seems** like D has to be 0 and 1 at the same time!

How the clock divider works

We already know that the inverter has a minimum delay t_{cd}.

The flip-flop is a physical device, so it also has a minimum delay, which is called t_{ccq}. *Let's make some notes about* t_{ccq}.



Now let's study what happens when $Q = 0$ *just before a rising edge of CLK1, and when* $Q = 1$ *just before a rising edge of CLK1.*

A small amount of delay is a good thing!

We've just seen that delays in the clock divider circuit are **essential** in making it work.

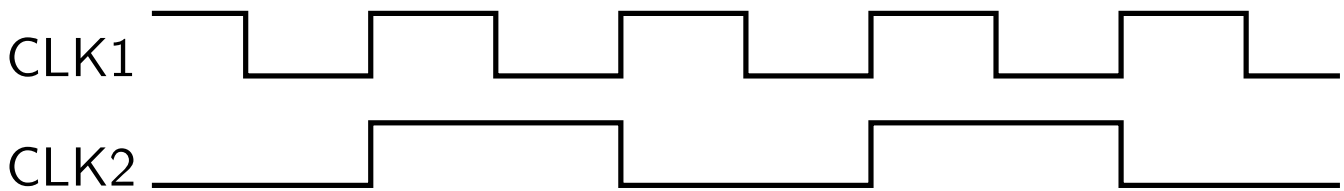
The same idea is true for most other systems built using D flip-flops.

Of course, **long delays** are **bad**—they result in circuits that are slow or unreliable, or both.

Note: There is much more to learn about timing of flip-flop circuits, but we won't do that until we get to Section 3.5 of Harris & Harris.

About the clock edges visible on the 'scope . . .

A few slides back it was suggested that if the clock divider input frequency was 100 kHz, the input and output signals would look like this on an oscilloscope . . .



Why do all the clock edges appear to be perfectly vertical?

Why do the edges on CLK2 appear to occur at exactly the same time as rising edges on CLK1?

Outline of Slide Set 7

Abbreviations for "D flip-flop"

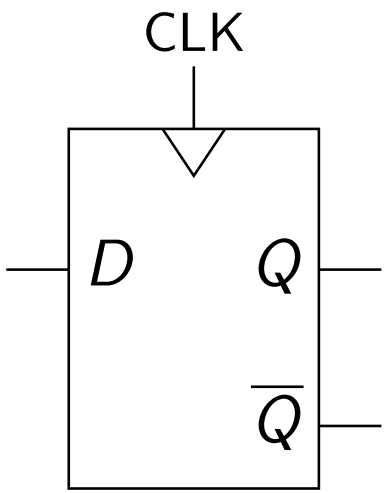
DFF is short, simple and obvious, so we'll use it in this course.

Some literature uses the term **flop**, which is short but possibly ambiguous.

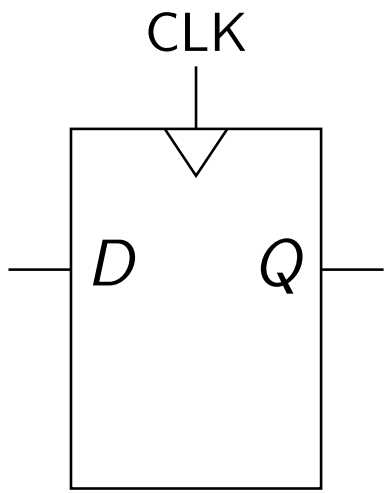
Symbols for D flip-flops

In symbols, the **triangle** on the CLK input indicates an **edge-triggered** device.

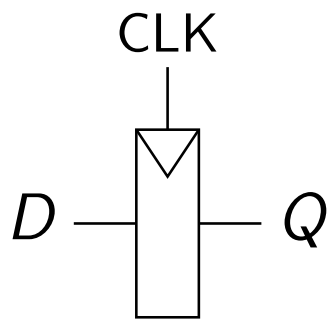
With $\overline{Q}$ output ...



Lacking $\overline{Q}$ output ...



Lacking $\overline{Q}$ output, condensed symbol ...



All DFF designs have an internal $\overline{Q}$ signal, but many of them, to save space and power, do not make $\overline{Q}$ available as an output.

Outline of Slide Set 7

N-bit registers

An N-bit *register* is a group of N DFFs with a common CLK input.

At right, (a) shows 4 DFFs configured as a 4-bit register, and (b) is a symbol for that register.

Let's make some notes about the symbol.

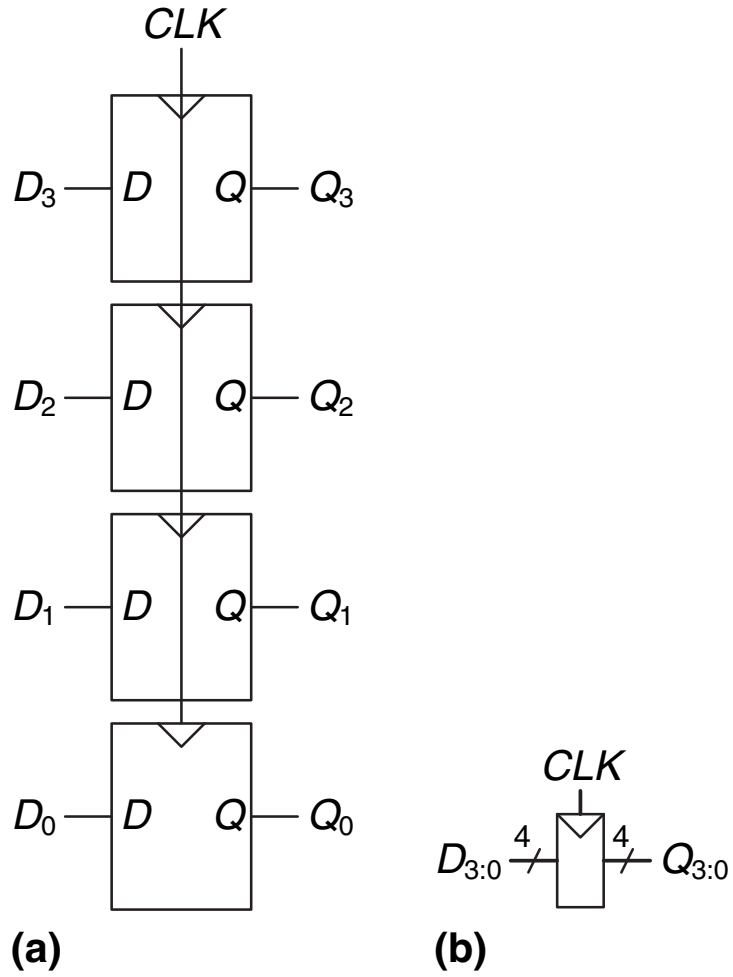
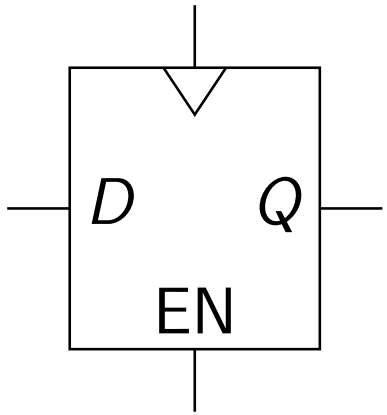


Image is taken from Figure 3.9 from Harris D. M. and Harris S. L., *Digital Design and Computer Architecture, 2nd ed.*, © 2013, Elsevier, Inc.

Enabled D flip-flops



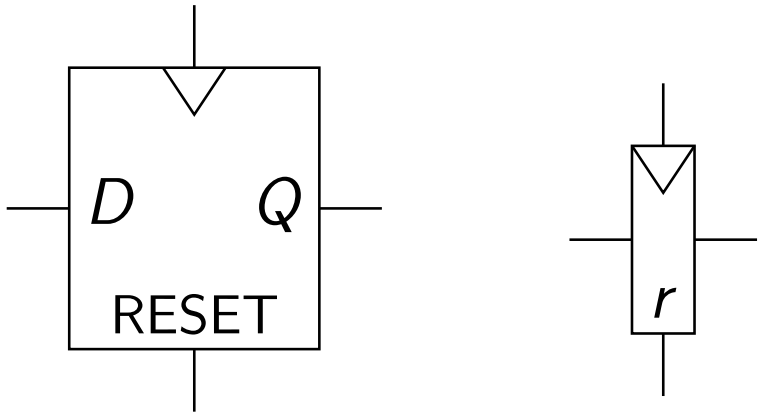
This kind of DFF is good for a circuit design in which it is useful to have a flip-flop sometimes hold its state for many clock cycles, rather than copy D on every single rising edge of the clock.

Let's write a precise description of the behaviour of an enabled DFF.

Let's show how an enabled DFF can be built using a "plain" DFF and a 2:1 multiplexer.

Resettable D flip-flops

Here are two symbols for the same thing . . .



Let's write a precise description of the behaviour of a resettable DFF, then build one using a "plain" DFF, an AND gate, and an inverter

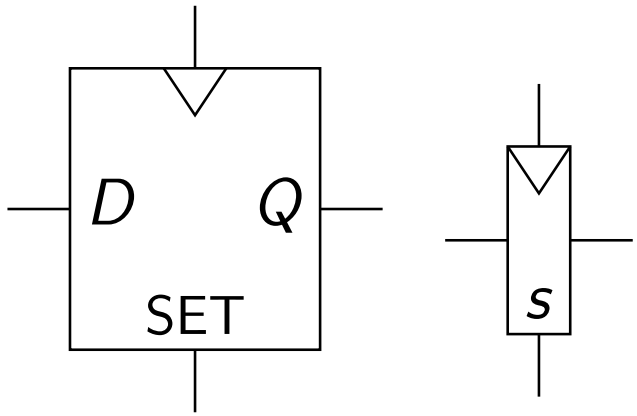
What would DFFs with reset inputs be useful for?

Variations on enabled and resettable DFFs

Here is a pretty obvious variation on the resettable DFF ...

On each rising edge of CLK,

$$Q = \begin{cases} D & \text{if SET} = 0 \\ 1 & \text{if SET} = 1 \end{cases}$$



Many textbooks use the names PRESET and CLEAR instead of SET and RESET. Also note that the above is a **synchronous** SET—some DFFs have **asynchronous** SET and/or RESET.

Some DFFs are designed to support two or all three of RESET, SET and EN inputs.

The rest of Section 3.2 in Harris & Harris

Section 3.2.7 presents and explains the most common present-day transistor-level designs for D latches and DFFs.

We will **not** cover this topic in ENEL 353. If you are curious about this material, you will have to go back and read Section 1.7 before reading Section 3.2.7.

Section 3.2.8 has a good example illustrating the difference between a D latch and a DFF. Check it out carefully!

Outline of Slide Set 7

Synchronous and asynchronous sequential circuits

A *synchronous* sequential circuit is a sequential logic system that

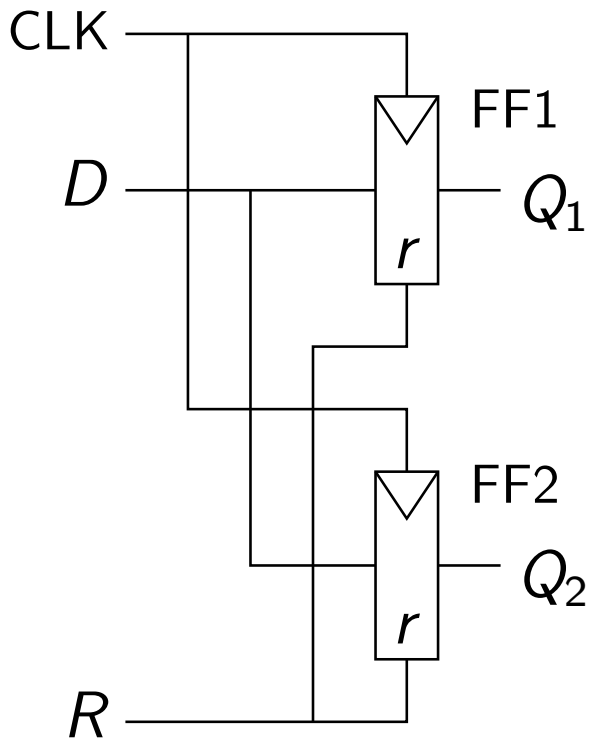
- has one or more bits of **state**; and
- has its state updates controlled by a **clock signal**, so that the state updates are **synchronized** by the clock.

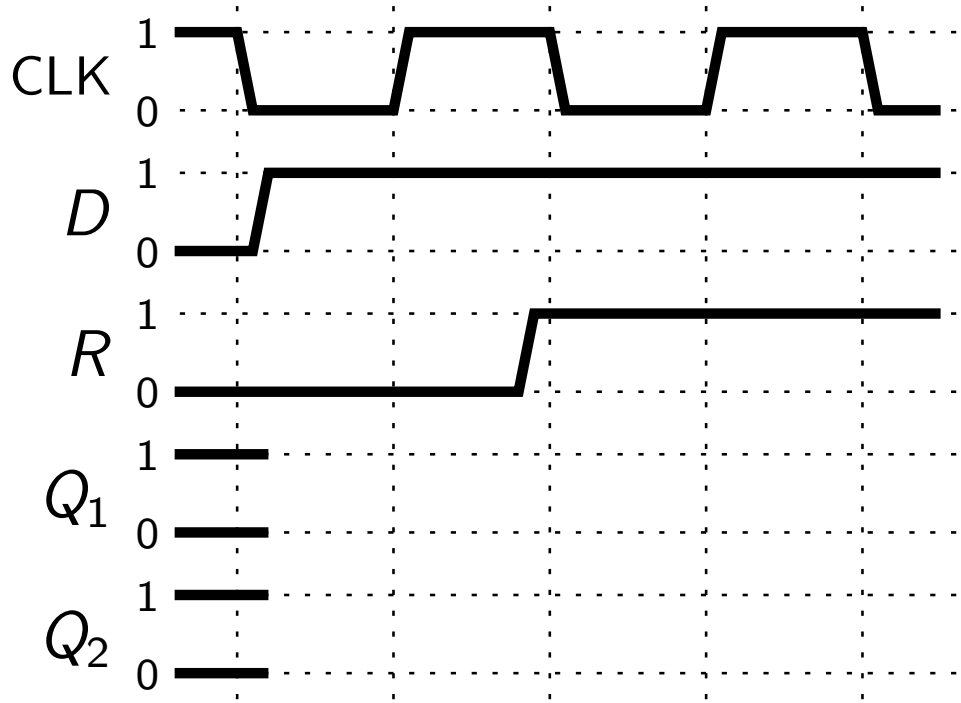
Making digital systems synchronous is a very powerful design technique—the **vast majority** of digital circuits, including just about all practical computer processors, are **synchronous** sequential systems.

An *asynchronous* sequential circuit is a sequential logic system in which some state updates occur **not synchronized** by the clock.

Synchronous and asynchronous reset of DFFs

FF1 has synchronous reset but FF2 has asynchronous reset.
Let's complete the timing diagram to show the difference in behaviour.





Textbook examples of asynchronous sequential circuits

Section 3.3.1 in Harris & Harris presents two asynchronous sequential circuits with problematic behaviour:

- A **ring oscillator** made from 3 inverters.
- A **D latch design that fails** if delays within its components aren't exactly right.

In ENEL 353, we're going to move on to synchronous sequential circuits, but it's worth studying these examples to get an idea of the difficulties that can arise in sequential circuits that **are not** run by a **clock signal**.