Flip-Flops and Latches

Table of contents

The D flip-flop 1
Latches vs flip-flops 4
Symbols used 5
The SR latch (cross-coupled NOR gates) 5
The SR latch from NAND gates (active low) 7
A real D flip-flop 8
D flip-flop chips 9
See also 10

Flip-flops and latches are the **storage elements** — the circuits that let a design remember a value even after the input that produced it has changed. They're what turns a combinational circuit into a sequential circuit[1].

Introduced in Lecture 5, 2026-08-10-introduction-to-sequential-circuits[2].

The D flip-flop

The D flip-flop is the **only** flip-flop type covered in this course. Its symbol is a rectangle with:

- **D** — the data input, on the left.
- **Q** — the output, on the right.
- **CLK** — the clock, a *control* input on the left, drawn with a small triangular wedge where it meets the box.

How it works: **Q copies the value of D (and remembers it) whenever CLK goes from 0 to 1** — the *rising edge*. In between clock edges the flip-flop is not responsive, so it holds the stored value. Q changes **only** on that 0→1 transition.

[1] courses/csse2010/sequential-circuits.pdf
[2] courses/csse2010/2026-08-10-introduction-to-sequential-circuits.pdf

Summary: a D flip-flop remembers a single bit — either a "1" or a "0". It keeps that value until the next clock edge, at which point the D input is transferred to output Q.

- To remember n bits you need n D flip-flops.
- An **n-bit register** is (by definition) n D flip-flops. See shift-registers[3].
- Flip-flops can themselves be made out of logic gates — see the SR latch and A real D flip-flop below.

Characteristic table

A **characteristic table** defines the operation of a flip-flop in tabular form. The left column is the input; the right column, $Q(t+1)$, is the *next* value of Q: what the output will be **on the next clock edge** (i.e. when CLK goes 0→1).

D	Q(t+1)	
0	0	Reset
1	1	Set

That is the whole behaviour: the flip-flop is *reset* (to 0) or *set* (to 1) according to D, at the clock edge.

A flip-flop's next-state column is written $Q(t+1)$ here. The state tables of state-machines[4] and counters[5] call the same thing *Next State*. The course uses both names (see csse2010-diagram-notation[6]).

Worked waveform

D is 0, 0, then 1, 1 across a clock with two rising edges. CLK starts low, pulses high, goes low, then pulses high again after D has risen.

Clock edge	D at that edge	Q after
(before the first edge)	—	unknown/undefined
1st rising edge	0	0
2nd rising edge	1	1

[3] courses/csse2010/shift-registers.pdf
[4] courses/csse2010/state-machines.pdf
[5] courses/csse2010/counters.pdf
[6] courses/csse2010/csse2010-diagram-notation.pdf

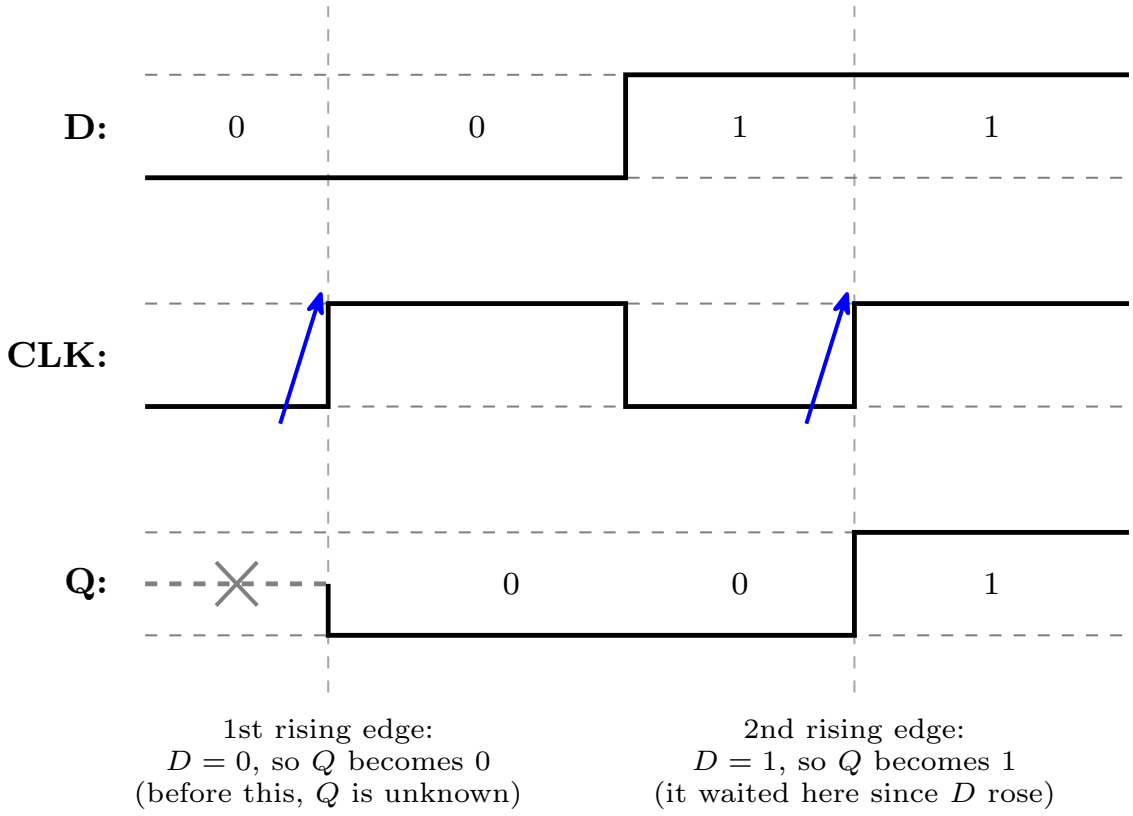


Figure 1: A D flip-flop waveform. The dashed verticals are the two rising clock edges, and the arrows mark them. Q is undefined (drawn dashed, with a cross) until the first edge. The point of the figure is the middle: D rises well before the second edge, and Q does not follow until that edge arrives.

Before the first rising edge Q is undefined, because nothing has been clocked in yet; an "X" marks that stretch. Between the two rising edges Q stays 0 even though D has already risen to 1: the flip-flop is only sensitive at the edge. Remember: Q changes **only** when the clock (CLK) goes from 0 to 1.

A second, harder worked waveform — four clock edges, with D both rising and falling between them — is in 2026-08-10-introduction-to-sequential-circuits[7].

Other flip-flop types

Other types exist — **JK flip-flops** and **T flip-flops** — but they are **not covered in this course**. Only the D flip-flop is used.

Latches vs flip-flops

- **Latches** are **level triggered** devices — they latch the output and respond to changes of logic *levels* on the inputs.
- A latch circuit can be modified so that it becomes sensitive to an **edge** (a momentary transition) of a control input, i.e. a clock signal. Such circuits are called **flip-flops**.
- A flip-flop can store 1 bit of information while being sensitive to a clock edge — it changes its output only *at* the clock edges, based on the inputs.
- So: **latches are level triggered; flip-flops are edge triggered.**

A clock signal has two edges:

Edge	Transition
Positive (rising) edge	$0 \rightarrow 1$
Negative (falling) edge	$1 \rightarrow 0$

A D flip-flop can therefore be **positive edge triggered** or **negative edge triggered**. In a positive edge triggered D flip-flop, D is copied to Q at the positive edge of the clock; in between clock edges the flip-flop is not responsive, thus stores the value.

The **state** of a flip-flop is the value it is storing. Flip-flops are more useful than latches in practice.

[7] courses/csse2010/2026-08-10-introduction-to-sequential-circuits.pdf

Symbols used

Four related symbols, all drawn as a rectangle with **D** in at the top left, **Q** out at the top right, and a clock input **CK** at the bottom left. What distinguishes them is the marking on that CK input.

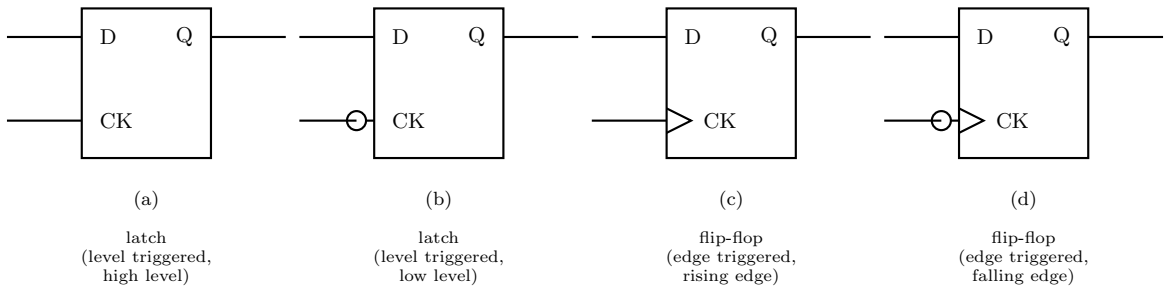


Figure 2: The four latch and flip-flop symbols, with the clock pin labelled CK (the course writes this pin CLK elsewhere). The wedge and the bubble read independently: the wedge says edge-triggered, therefore flip-flop; the bubble says inverted, therefore falling edge or low level. The reading under each symbol follows from those two rules.

Symbol	CK input marking	Device	Triggered on
(a)	plain input	Latch	High level of CK
(b)	**bubble** (inversion circle)	Latch	Low level of CK
(c)	**triangle/wedge**	Flip-flop	**Rising** (positive) edge of the clock
(d)	**bubble + triangle**	Flip-flop	**Falling** (negative) edge of the clock

The two markings read independently and are the whole key to the notation:

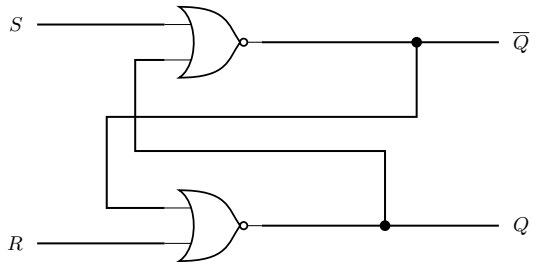
- The **triangle** indicates **edge-triggered**, and therefore that the device is a flip-flop rather than a latch. No triangle means level-triggered, i.e. a latch.
- The **bubble** indicates inversion, i.e. the *falling* edge (or the low level) rather than the rising edge (or the high level).

The SR latch (cross-coupled NOR gates)

Structure: two 2-input NOR gates cross-coupled.

- The **top** NOR gate takes S as one input and the *bottom* gate's output as its other input; its output is $\bar{Q}$.
- The **bottom** NOR gate takes R as one input and the *top* gate's output ($\bar{Q}$) as its other input; its output is Q.

So each gate's output feeds back into the other gate's input — that feedback loop is what stores the bit.



A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

$\overline{0+x} = \bar{0}\cdot\bar{x} = 1\cdot\bar{x} = \bar{x}$

a NOR gate with one input held at 0 is an inverter for the other input

$\overline{1+x} = \bar{1}\cdot\bar{x} = 0$

a NOR gate with one input held at 1 has output 0, whatever the other input

Figure 3: The SR latch: two cross-coupled NOR gates, with the NOR truth table to remember and the two special cases the analysis needs. Between them they are the whole analysis: a NOR with a 0 on one input passes the other input inverted, and a NOR with a 1 on one input is pinned to 0. Where the two feedback wires cross there is no junction dot, so it is two wires passing, not a connection.

The analysis needs both identities in the figure. Together they are what makes each row fall out in a line. Both are just De Morgan applied to a NOR (see boolean-algebra[8]): $\overline{A+B} = \bar{A}\cdot\bar{B}$.

The truth table:

S	R	Q	$\bar{Q}$	Meaning
0	0	Q_{n-1}	—	**Storage / remember**
0	1	0	1	**Reset**
1	0	1	0	**Set**
1	1	—	—	**Invalid** $\times$

Q_{n-1} means "whatever Q was before" — the hold row's entry is not a bit but a reference to the previous value. Three cells are left blank: $\bar{Q}$ in the hold row, and both outputs in the last row, which is *Invalid*.

Reasoning for each row, using the two identities above:

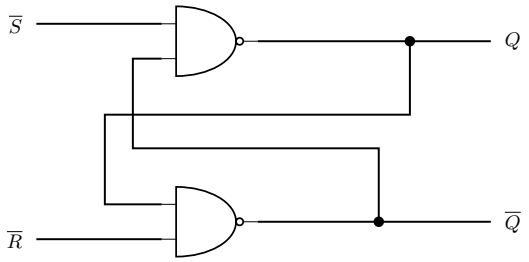
[8] courses/csse2010/boolean-algebra.pdf

- $S = 0, R = 0$: each gate has a 0 on its external input, so each is an inverter of the other's output — $\bar{Q} = \overline{Q}$ and $Q = \overline{\bar{Q}}$. Both $Q = 1$ and $Q = 0$ satisfy that, so the latch simply holds whatever it already had. **This row is the whole point of the circuit**: it is the row in which the latch remembers.
- $S = 1, R = 0$: the top gate has a 1 on an input, so $\bar{Q} = 0$ regardless. The bottom gate then sees $R = 0$ and $\bar{Q} = 0$, so $Q = 1$. S **sets** the latch.
- $S = 0, R = 1$: mirror image — the bottom gate has a 1 on an input, so $Q = 0$; the top gate then sees $S = 0$ and $Q = 0$, so $\bar{Q} = 1$. R **resets** the latch.
- $S = 1, R = 1$: both gates have a 1 on an input, so both outputs are 0: $Q = 0$ and $\bar{Q} = 0$. That result is what condemns the combination: the two outputs are supposed to be complements and here they are equal, so the labels Q and $\bar{Q}$ stop meaning anything. Worse, what happens *after* both inputs return to 0 simultaneously is not uniquely determined (the two gates race), so the resulting state is **indeterminate**. Hence **invalid** — a combination you must not apply.

This circuit is the **active high SR latch**: the inputs do their job when driven to 1, and the resting/hold condition is $S = R = 0$. That name is what distinguishes it from the NAND version below.

The SR latch from NAND gates (active low)

Exercise: analyse the SR latch circuit when the NOR gates are replaced with NAND gates, and complete the truth table. The answer circuit is below. The gate symbols are not merely swapped: **the inputs are relabelled $\bar{S}$ and $\bar{R}$, and which gate's output is called Q is swapped**, so that the set/reset roles survive the change.



A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

$\overline{1 \cdot x} = \overline{1} + \overline{x} = 0 + \overline{x} = \overline{x}$

a NAND gate with one input held at 1 is an inverter for the other input

$\overline{0 \cdot x} = \overline{0} + \overline{x} = 1$

a NAND gate with one input held at 0 has output 1, whatever the other input

Figure 4: The NAND SR latch, the active low SR latch, with the NAND truth table and the two identities that drive the analysis. Compare against the NOR version: the gate outputs have swapped names, so the top gate now produces Q, and the inputs are drawn overlined because they are active low.

Worked through with the two NAND identities in the figure:

$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$	Meaning
0	0	1	1	**Invalid** — outputs not complementary
0	1	1	0	**Set** ($\bar{S}$ asserted)
1	0	0	1	**Reset** ($\bar{R}$ asserted)
1	1	Q_{n-1}	—	**Storage / remember**

(Rows in ascending binary order of the inputs, which is what every truth table in the course does — see csse2010-diagram-notation[9].)

- $\bar{S} = \bar{R} = 1$: each gate has a 1 on its external input, so each inverts the other's output — the same self-consistent pair as the NOR latch's $S = R = 0$ row. **Hold.**
- $\bar{S} = 0$: the top gate is pinned to 1, so $Q = 1$; the bottom gate then sees $\bar{R} = 1$ and $Q = 1$, so $\bar{Q} = 0$. **Set.**
- $\bar{R} = 0$: mirror image — $\bar{Q} = 1$, and the top gate then gives $Q = 0$. **Reset.**
- $\bar{S} = \bar{R} = 0$: both gates pinned to 1, so $Q = \bar{Q} = 1$. Same objection as before, so **invalid**.

The headline result, and the reason the inputs are drawn overlined: with NAND gates the inputs become **active low**. The latch rests (holds) at $\bar{S} = \bar{R} = 1$ instead of at $S = R = 0$, and the forbidden combination moves from both-high to both-low. Everything else — set, reset, hold, one invalid corner — is the same circuit doing the same job.

This active-low convention is general: it is why the 74HCT74's asynchronous pins are $\overline{\text{SET}}$ and $\overline{\text{CLR}}$ (see device-pinouts[10]). The SET and CLR inputs are typically active low.

A real D flip-flop

A real D flip-flop is built from cross-coupled NAND gates — **six** NAND gates in three cross-coupled pairs: two input latches driven by D and CLK, and a third pair forming the output latch that carries Q and $\bar{Q}$. That is the same storage idea as the SR latch above, wrapped in enough extra gating to make it respond to an *edge* rather than a *level*.

As well as D, CLK and the outputs Q and $\bar{Q}$, it has two extra inputs:

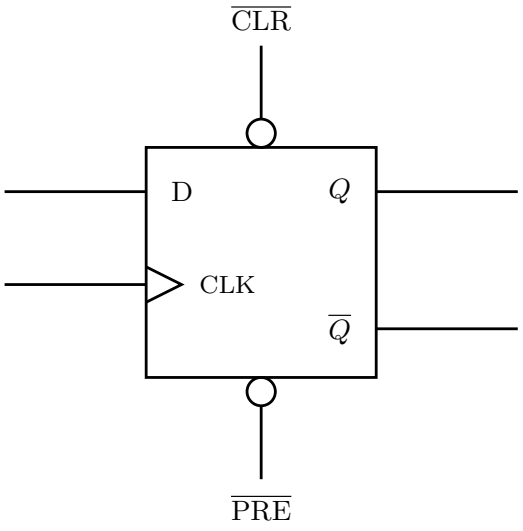
- **$\overline{\text{PRE}}$ (Preset)** — forces Q to 1.
- **$\overline{\text{CLR}}$ (Clear)** — forces Q to 0.

Both are drawn with overbars and with a bubble on the pin, i.e. they are **active low**, and both are **asynchronous**: they act on the latch directly, independent of the clock, rather than waiting for a clock edge. They run straight into the NAND gates of the latches, bypassing

[9] courses/csse2010/csse2010-diagram-notation.pdf
[10] courses/csse2010/device-pinouts.pdf

the D/CLK path entirely — $\overline{\text{PRE}}$ reaches the gate that produces Q and $\overline{\text{CLR}}$ the gate that produces $\bar{Q}$, which is exactly how each forces its output.

The two things to remember about these pins: they are *active low inputs* and *asynchronous inputs*.



Asynchronous, active-low inputs
they act the instant they are driven low — they do *not* wait for a clock edge

$\overline{\text{CLR}} = 0 \Rightarrow Q = 0$ (clear)
$\overline{\text{PRE}} = 0 \Rightarrow Q = 1$ (preset)
$\overline{\text{CLR}} = \overline{\text{PRE}} = 1 \Rightarrow$ normal clocked operation

The bubble is the active-low marking; the overbar on the name says the same thing.

Figure 5: The D flip-flop symbol with its asynchronous pins, CLR on top and PRE underneath. The 74HCT74 in device-pinouts is drawn the other way up (SET on top, CLR below), which is the course's usual order; the course is also inconsistent about the *naming* (PRE/CLR against SET/CLR). The three consequences of the pins being asynchronous and active low are on the right.

Both pins being asserted at once ($\overline{\text{PRE}} = \overline{\text{CLR}} = 0$) is the same kind of contradiction as $S = R = 1$ on the NOR latch — it demands $Q = 1$ and $Q = 0$ simultaneously — and is likewise a combination to avoid. It follows from the SR-latch analysis above.

When neither is needed they can simply be **omitted from a drawing**: the course's *Device Pinouts and Symbols* sheet says so in words, and its own logic-diagram symbol for a D flip-flop leaves them off. See device-pinouts[11] and circuit-schematics[12].

D flip-flop chips

- **74HCT74** — dual D flip-flop (two independent D flip-flops in a 14-pin package), each with its own clock, its own CLR and its own PR (preset), and $Q/\bar{Q}$ outputs. Pinout and

[11] courses/csse2010/device-pinouts.pdf
[12] courses/csse2010/circuit-schematics.pdf

schematic symbol in device-pinouts[13].

- **74HCT273** — eight D flip-flops in a 20-pin package, so it can hold **one byte** (8 bits) of information. It is not in the course's device-symbols document, so there is no pinout for it here.

The full symbol set is in the device symbols PDF on Blackboard.

See also

- 2026-08-10-introduction-to-sequential-circuits[14] — Lecture 5, where all of this is introduced
- sequential-circuits[15] — what you build once you have a storage element
- shift-registers[16] · counters[17] · state-machines[18]
- memory-technologies[19] — static RAM, a memory that keeps each bit in a latch
- week3-lab-logisim-and-flip-flops[20] — the lab that simulates these circuits
- logic-gates[21] · boolean-algebra[22] — the NOR/NAND behaviour the latch analysis rests on
- timing-diagrams[23] — how to read and draw the waveforms above
- csse2010-diagram-notation[24] — the drawing conventions these figures follow

[13] courses/csse2010/device-pinouts.pdf
[14] courses/csse2010/2026-08-10-introduction-to-sequential-circuits.pdf
[15] courses/csse2010/sequential-circuits.pdf
[16] courses/csse2010/shift-registers.pdf
[17] courses/csse2010/counters.pdf
[18] courses/csse2010/state-machines.pdf
[19] courses/csse2010/memory-technologies.pdf
[20] courses/csse2010/week3-lab-logisim-and-flip-flops.pdf
[21] courses/csse2010/logic-gates.pdf
[22] courses/csse2010/boolean-algebra.pdf
[23] courses/csse2010/timing-diagrams.pdf
[24] courses/csse2010/csse2010-diagram-notation.pdf