

SEQUENTIAL LOGIC · LECTURES 1–2

Latches and Flip-Flops

How a circuit remembers: from two cross-coupled gates to the edge-triggered D flip-flop.

Part 1 · SR latch, enable, D latch

Part 2 · Clock, edge triggering, D flip-flop

Based on Mano & Ciletti, Digital Design (6th ed.), Sections 5.1–5.4

PART 1 • WHY SEQUENTIAL LOGIC?

Combinational circuits have no memory. Sequential circuits do.

COMBINATIONAL

Output = $f(\text{present inputs})$

A truth table says it all: the same inputs always give the same output. The past doesn't matter.

SEQUENTIAL

Output = $f(\text{inputs, stored state})$

The same inputs can give different outputs, depending on what happened before.

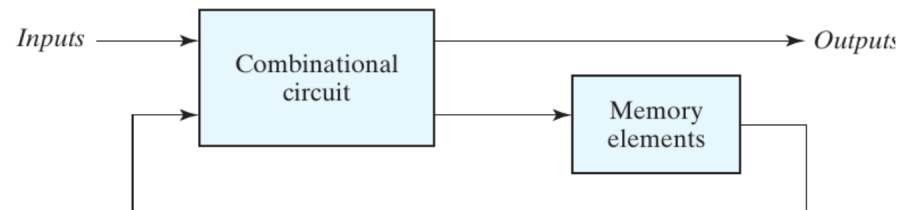
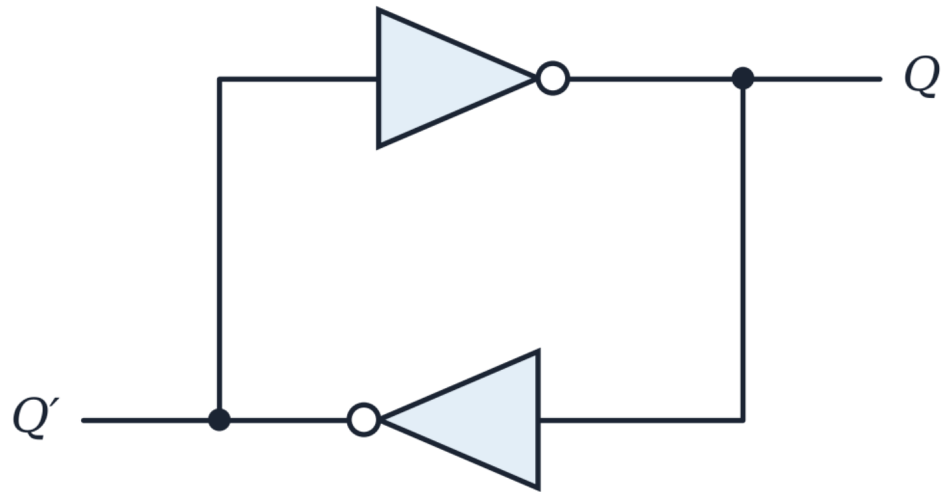


Figure 5.1, Block diagram of sequential circuit. Mano & Ciletti, Digital Design, 6th ed.

Two inverters in a loop can hold a bit



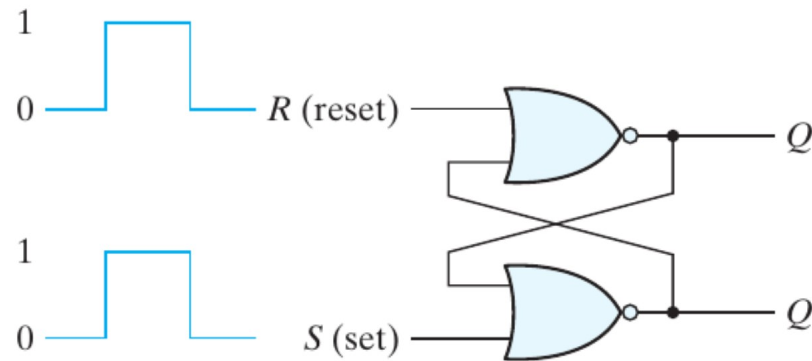
If $Q = 1$, the lower inverter makes $Q' = 0$, which keeps $Q = 1$. **Stable.**

If $Q = 0$, the same loop keeps $Q = 0$. **Also stable.**

Two stable states: one bit of memory. But there is no input, so we can't change it.

PART 1 • SR LATCH, NOR VERSION

Replace the inverters with NOR gates, and add inputs S and R



(a) Logic diagram

S	R	Q	Q'
1	0	1	0
0	0	1	0 (after $S = 1, R = 0$)
0	1	0	1
0	0	0	1 (after $S = 0, R = 1$)
1	1	0	0 (forbidden)

(b) Function table

S = 1, R = 0: set, $Q = 1$.

S = 0, R = 1: reset, $Q = 0$.

S = 0, R = 0: no change.

S = 1, R = 1: forbidden.

PART 1 · SR LATCH, NOR VERSION

S = R = 0 remembers the last state

Step	S R	Q	What happened
1	1 0	1	Set
2	0 0	1	Remembers the set
3	0 1	0	Reset
4	0 0	0	Remembers the reset

Steps 2 and 4 have **the same inputs** but **different outputs**.

No truth table can describe this. The output depends on which input was last 1.

That is what makes the latch sequential.

Why is $S = R = 1$ forbidden?

01

Q and Q' stop being complements

A 1 on either NOR input forces its output to 0. So both Q and Q' become 0. The latch is neither set nor reset.

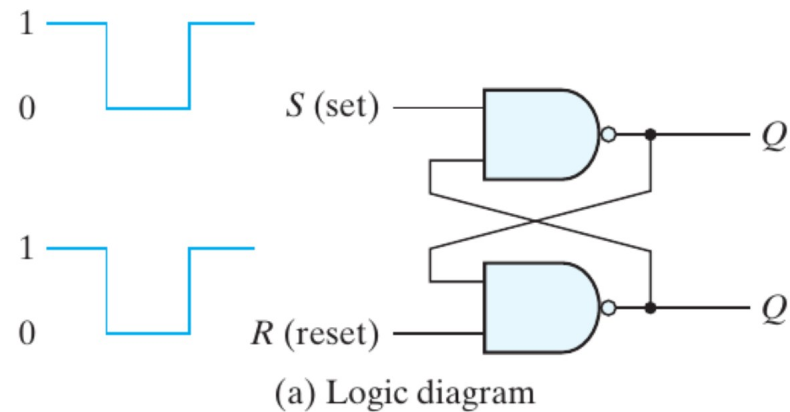
02

The next state is unpredictable

If S and R then return to 0 together, both gates try to go to 1 at once. Whichever gate is faster wins: a **race**. The result may even be metastable.

PART 1 • SR LATCH, NAND VERSION

With NAND gates, the inputs are active-low



S	R	Q	Q'
1	0	0	1
1	1	0	1
0	1	1	0
1	1	1	0
0	0	1	1

(b) Function table

A 0 on S sets; a 0 on R resets.

S = R = 1 holds: the NAND latch remembers here.

S = R = 0 is forbidden: $Q = Q' = 1$.

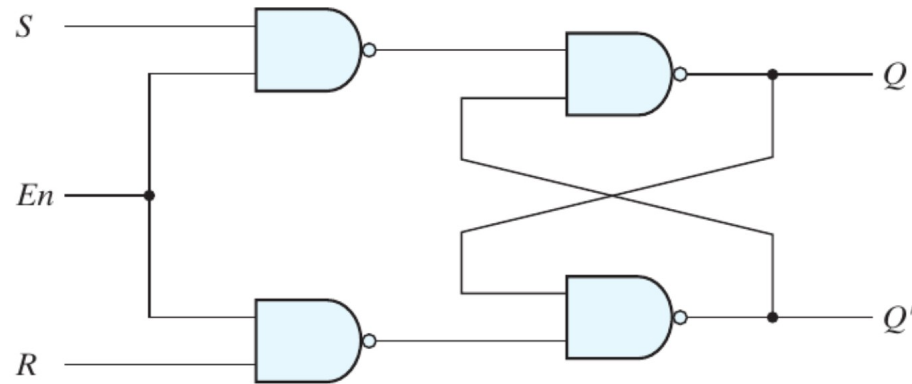
PART 1 • COMPARISON

NOR and NAND latches: same idea, inputs inverted

	NOR latch	NAND latch (S'R')
Inputs active	High (1)	Low (0)
Set	$S = 1, R = 0$	$S = 0, R = 1$
Reset	$S = 0, R = 1$	$S = 1, R = 0$
Hold (memory)	$S = 0, R = 0$	$S = 1, R = 1$
Forbidden	$S = 1, R = 1$	$S = 0, R = 0$

In both, the hold state and the forbidden state are the two cases where S equals R.

An enable input decides when the latch listens



(a) Logic diagram

En	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	$Q = 0$; reset state
1	1	0	$Q = 1$; set state
1	1	1	Indeterminate

(b) Function table

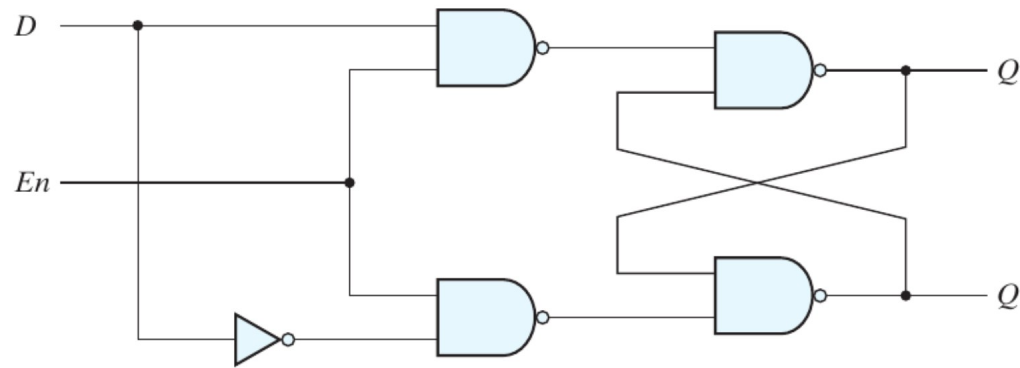
En = 0: the front gates block S and R.
The latch holds, whatever S and R do.

En = 1: it behaves as an ordinary SR latch.

Still a problem: $E_n = S = R = 1$ is indeterminate.

PART 1 • D LATCH

One input D and its complement: $S \neq R$ always



(a) Logic diagram

En	D	Next state of Q
0	X	No change
1	0	$Q = 0$; reset state
1	1	$Q = 1$; set state

(b) Function table

D drives the S side, **D'** drives the R side, both through the enable gates.

En = 1: Q follows D. The latch is **transparent**.

En = 0: Q keeps the value D had when En fell.

PART 1 · D LATCH

Why the D latch needs an enable to have memory

$S \neq R$ ALWAYS

Removes the forbidden state, where $S = R$.

But the **hold** state is also $S = R$. It is removed too.

WITHOUT ENABLE

Q would simply equal D at all times.

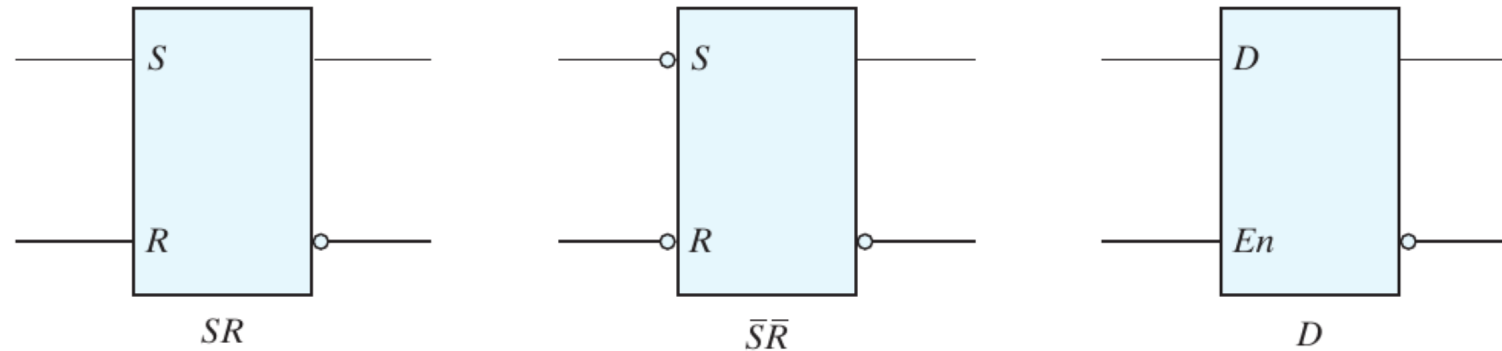
That's a buffer, not a memory.

WITH ENABLE

$En = 0$ puts the inner latch back in its hold state.

Memory returns, under our control.

Graphic symbols for latches



SR: active-high inputs (NOR latch).

$S'R'$: bubbles mean active-low inputs (NAND latch).

D: data and enable. The bubbled output is Q' .

PART 1 • RECAP

From feedback to a controllable 1-bit memory

SR latch

Remembers when $S = R = 0$ (NOR). One forbidden input.



+ Enable

Listens only when $En = 1$.
Forbidden input still possible.



D latch

$S = D, R = D'$. No forbidden input;
enable gives hold.

Next: what happens when the enable is driven by a clock, and why that is not good enough.

LATCHES AND FLIP-FLOPS · PART 2

The clock, and the edge-triggered flip-flop

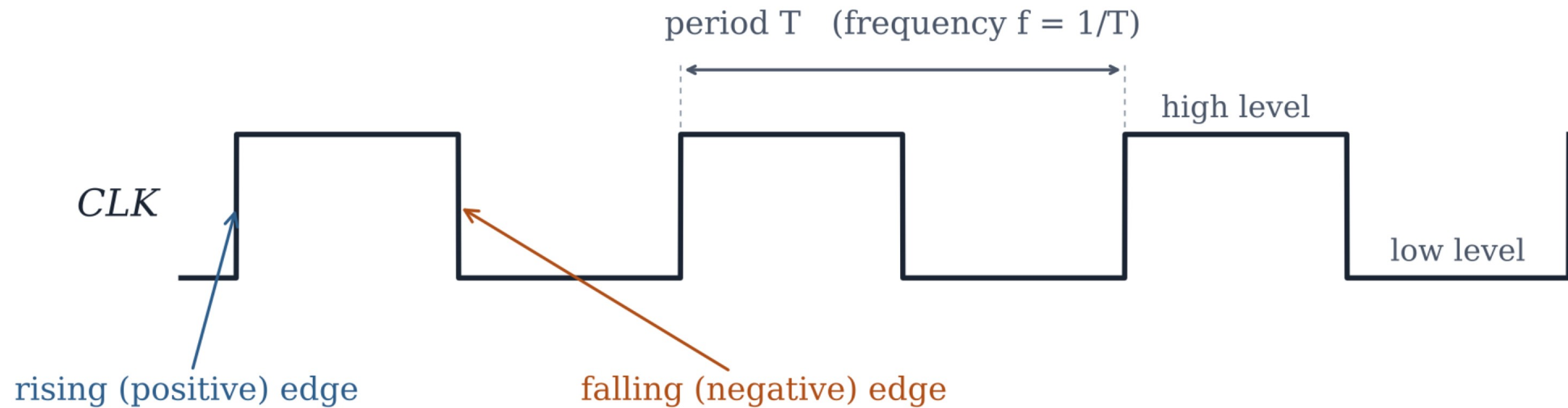
Clock signals

Level-sensitive vs edge-triggered

Master–slave D flip-flop

PART 2 • THE CLOCK

A clock is a periodic train of pulses



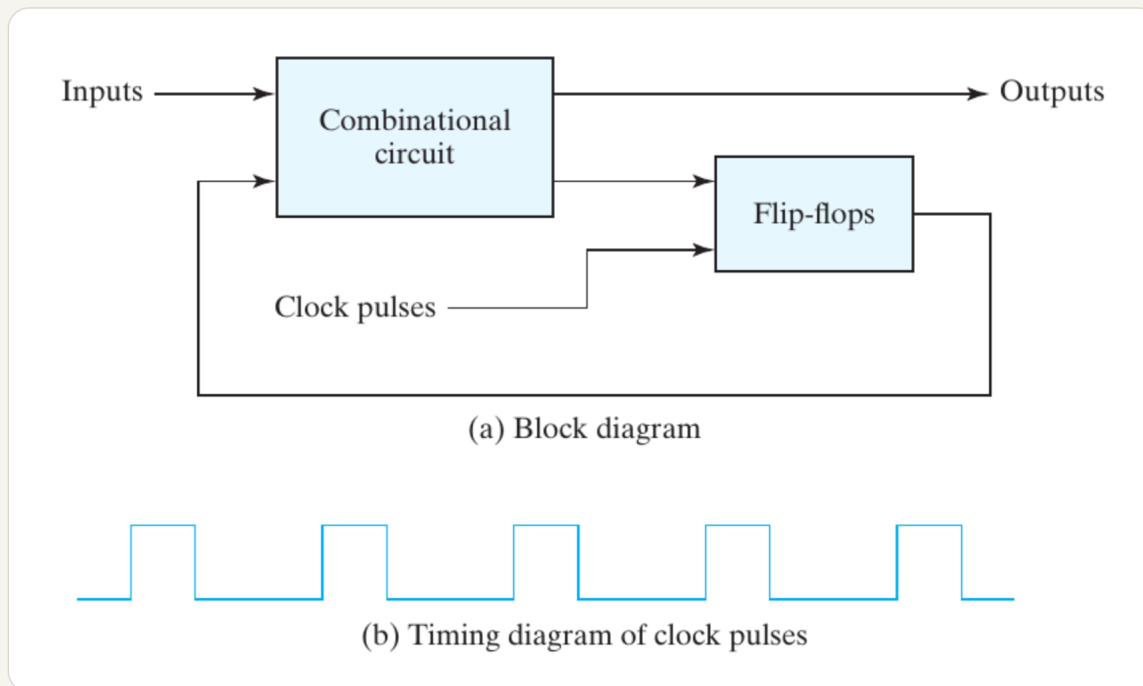
Two **levels**: high and low.

Two **edges** per period: rising and falling.

Example: 100 MHz means $T = 10$ ns.

PART 2 • SYNCHRONOUS CIRCUITS

One clock decides when every storage element updates



The **clock** decides *when* state changes. The inputs and logic decide *what* it changes to.

The logic must settle before the next clock pulse, which limits the clock speed.

Note the feedback loop: flip-flop outputs go back into the logic.

Connect the clock to En: the latch is level-sensitive

Clock low

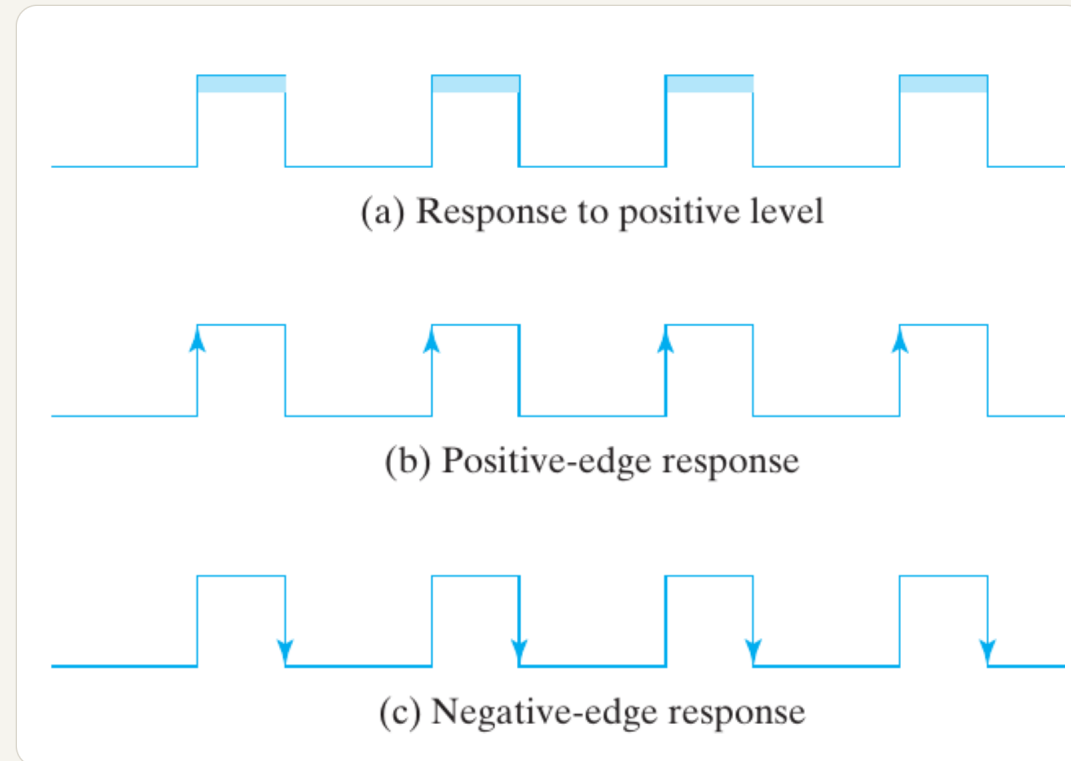
Q and Q' hold steady for the whole low part of the cycle. Good.

Clock high

The latch is transparent: **every** change in D reaches Q while the clock stays high.

Why that breaks a synchronous circuit: Q feeds back through the logic to D. While the clock is high, a new Q makes a new D, which makes a new Q... The state can change several times in one pulse, and where it ends up depends on gate delays.

The fix: respond only at a clock edge



(a) Latch: level-sensitive

Responds during the whole active level.

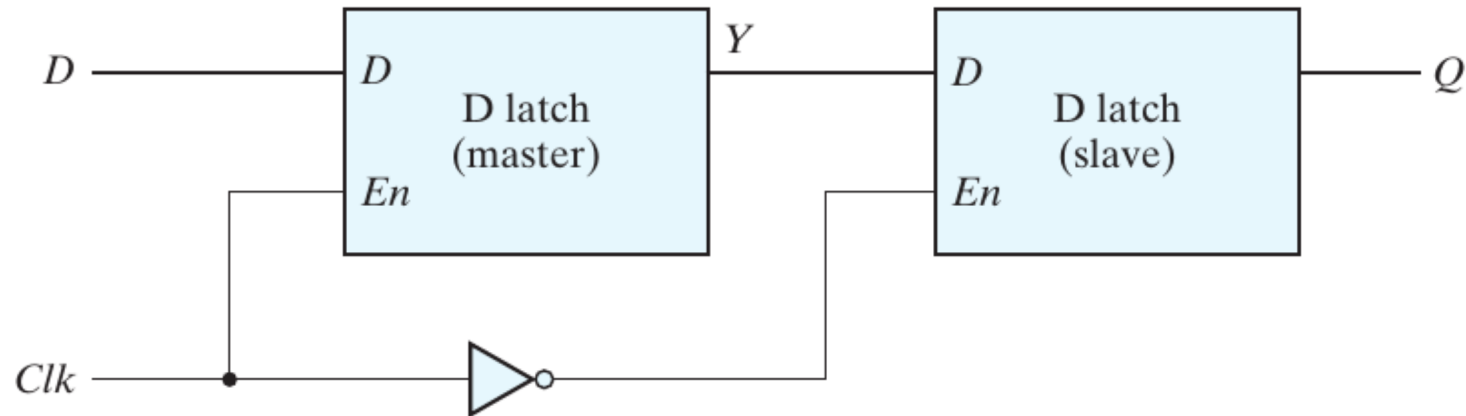
(b), (c) Flip-flop: edge-triggered

Responds only at the rising edge, or only at the falling edge. Once per clock cycle.

All latches are level-sensitive. Flip-flops are edge-triggered.

PART 2 • MASTER-SLAVE D FLIP-FLOP

Two D latches, enabled on opposite clock levels



Clk = 1

Master open: Y follows D . Slave closed: Q holds.

Clk falls 1 $\rightarrow$ 0

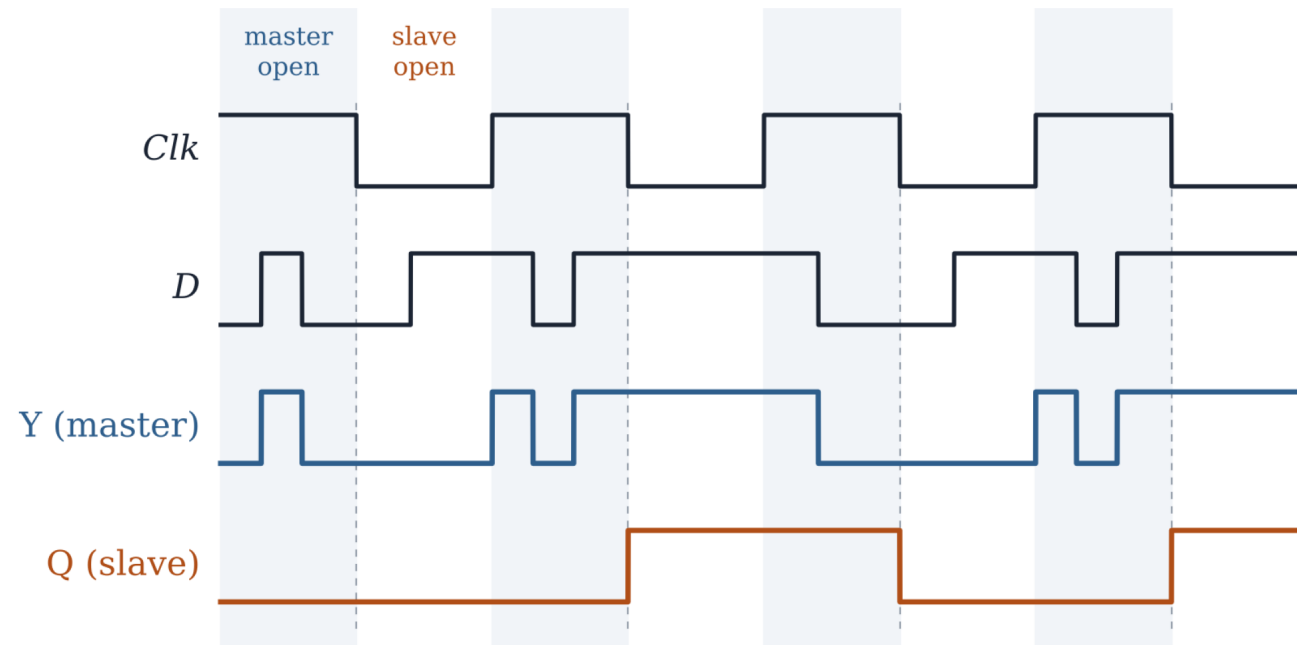
Master closes, freezing Y . Slave opens: Q takes Y .

Clk = 0

Master closed: D can't get in. Q stays put.

PART 2 • MASTER-SLAVE D FLIP-FLOP

Inside the flip-flop: Y wiggles, Q changes only at falling edges



"The slave is transparent while $\text{Clk} = 0$. Why doesn't Q change then?"

Because its input is frozen

The slave's input is Y, and the master is closed while $\text{Clk} = 0$. A transparent latch with a constant input gives a constant output.

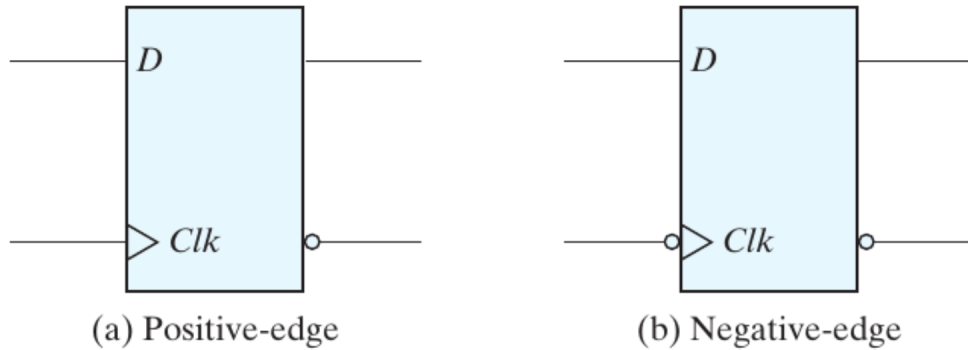
So only one value of D matters

The value D has just before the falling edge, when the master closes. Earlier wiggles of D are overwritten.

D at the falling edge goes to Q, and stays there until the next falling edge.

PART 2 · POSITIVE OR NEGATIVE EDGE

Move the inverter, and the flip-flop triggers on the rising edge



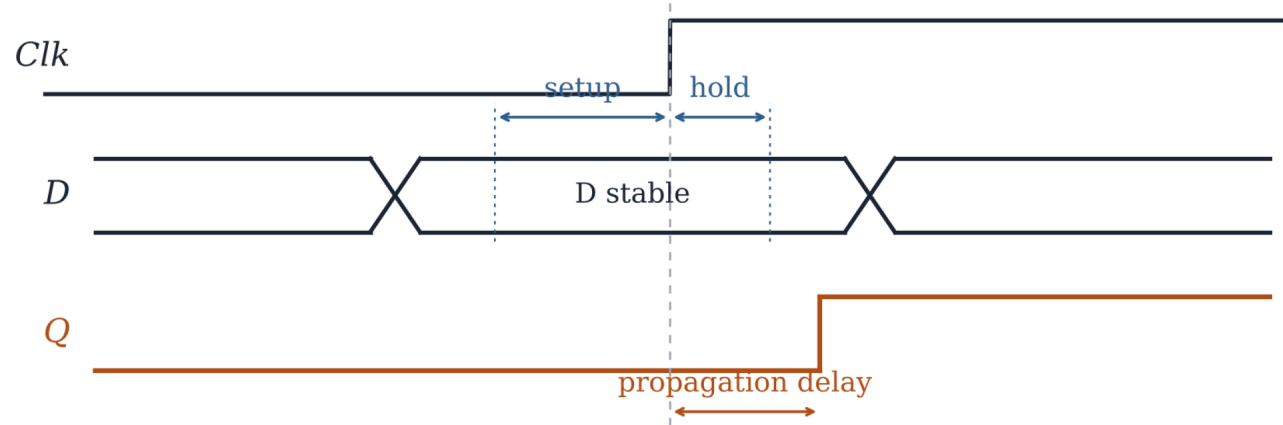
Positive-edge: invert the clock to the master instead of the slave.

Triangle (dynamic indicator) at Clk: edge-triggered.

Bubble + triangle: negative-edge triggered.

PART 2 • TIMING

"Captured at the edge" really means a short window around it



Setup time: D must be steady this long *before* the edge.

Hold time: D must stay steady this long *after* the edge.

Propagation delay: from the edge until Q settles.

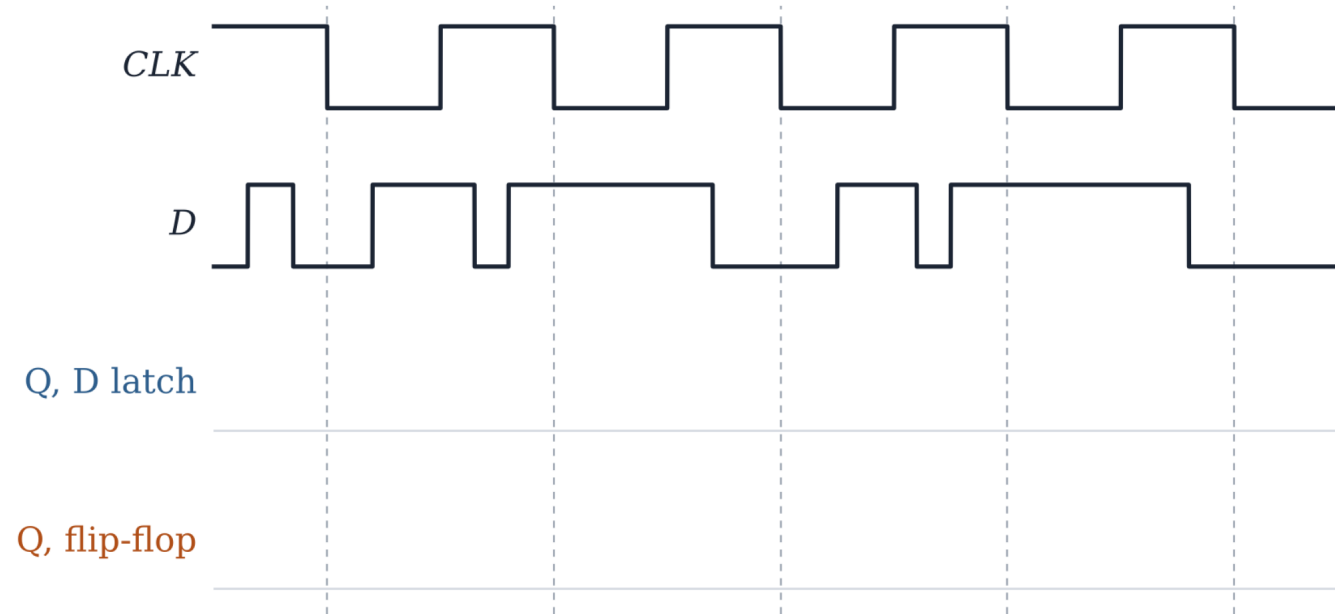
D latch versus D flip-flop

	D latch	D flip-flop
Responds to	A level of En (level-sensitive)	An edge of Clk (edge-triggered)
While active	Transparent: Q follows D	No transparent phase
Output changes	Any time during the active level	Once per clock cycle, at the edge
Built from	SR latch plus steering gates	Two D latches (master–slave)
Symbol	D, En	D, Clk with triangle (bubble = negative edge)

Clocked sequential circuits use flip-flops. From here on, every circuit we build uses D flip-flops.

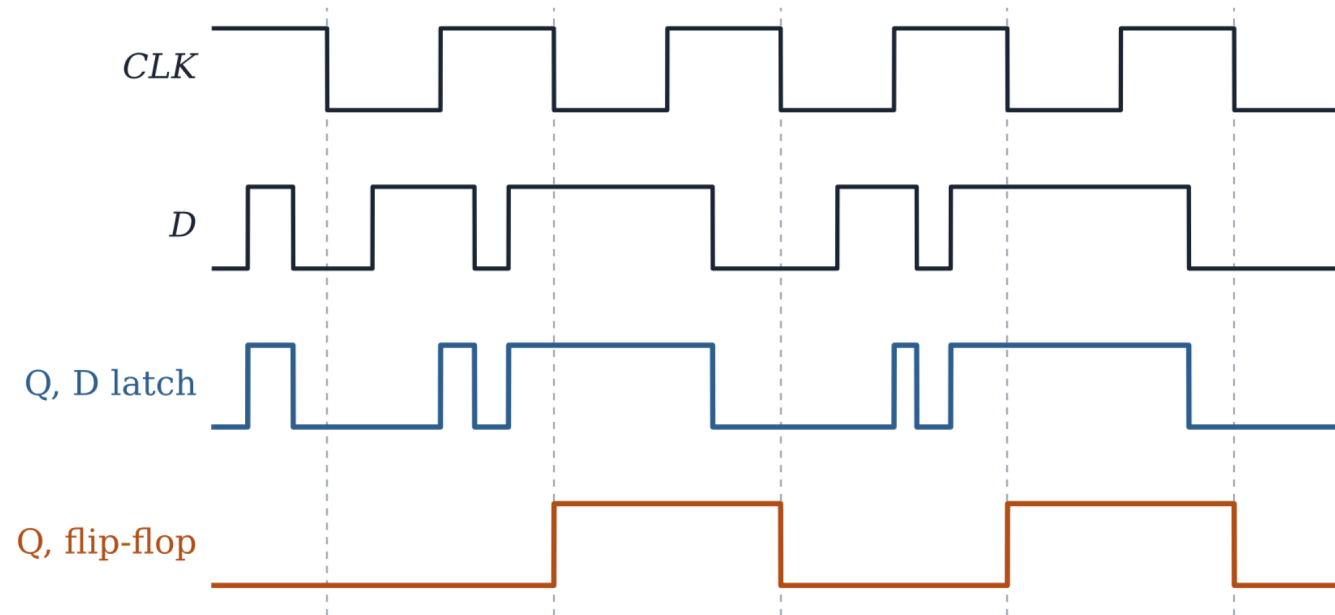
YOUR TURN

Draw Q for a D latch ($En = CLK$) and for a negative-edge D flip-flop



ANSWER

The latch copies every glitch; the flip-flop changes only at falling edges



SUMMARY

From a loop of gates to a reliable one-bit memory

SR latch

Feedback gives memory. Hold and forbidden are both $S = R$.

D latch

$S = D$, $R = D'$ removes the forbidden input; enable restores hold.

Level-sensitive

A clocked latch is transparent for the whole active level: unsafe with feedback.

Edge-triggered D flip-flop

Master–slave: D at the edge goes to Q and stays for one clock cycle.

Next: many flip-flops on one clock make a register.