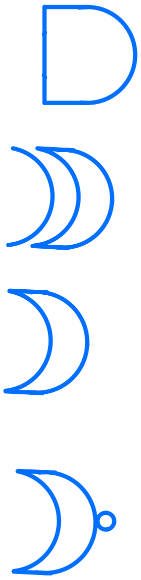
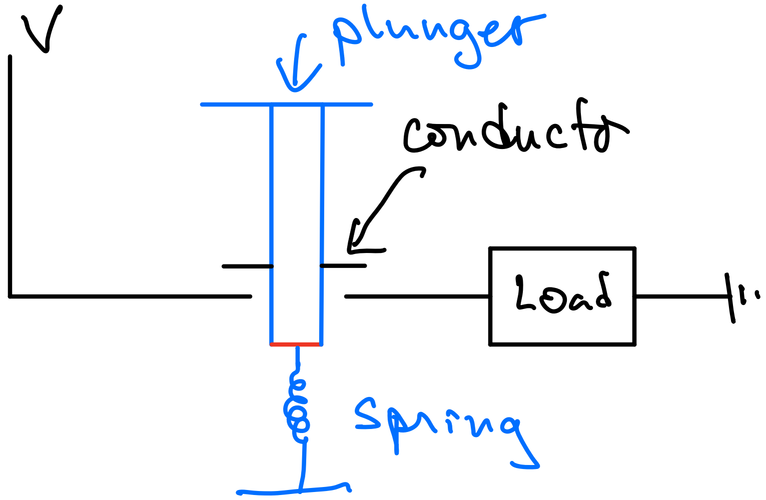
Application of SR latch:

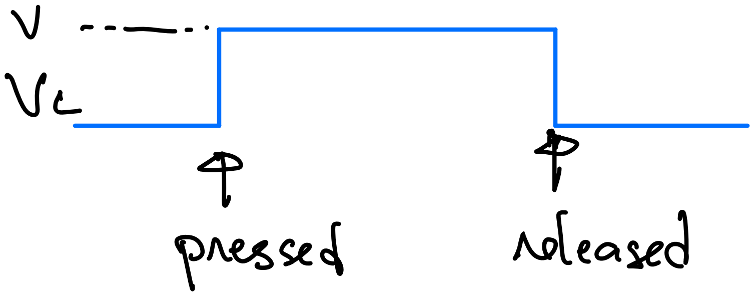


Simple analog circuit to turn on a load



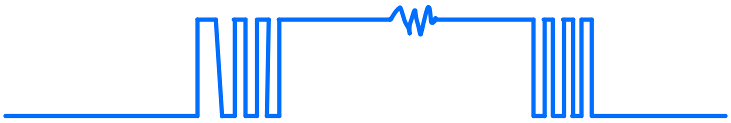
press plunger, current flows from source V to gnd through load

plot V across load V_L vs time



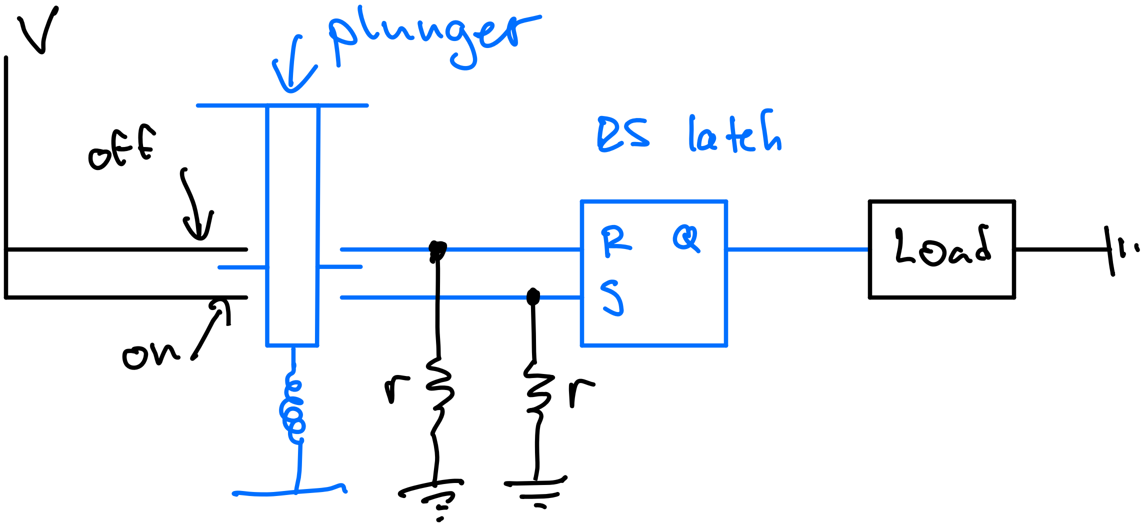
"ideal"

blow up edges → always see "bouncing"



too much coffee!!

solution: use RS latch



push down turns Q on, release and Q→0

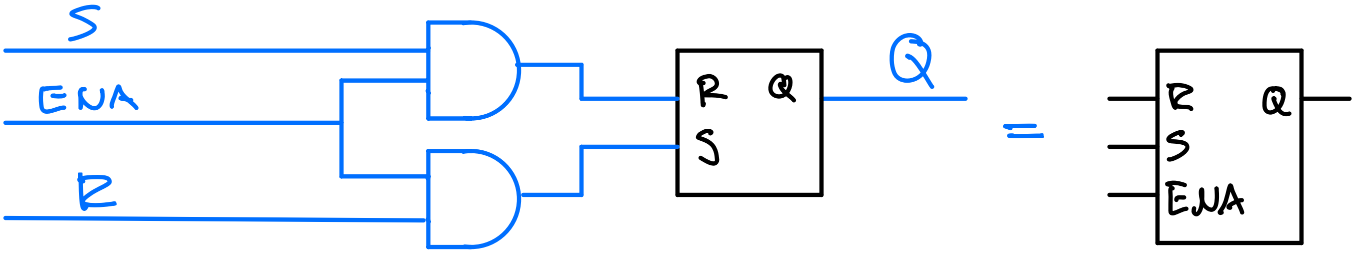
bouncing will be "forgotten" due to latching

Note: pull-down resistors set EM voltage levels to 0,0 when not active so that R & S are well-defined

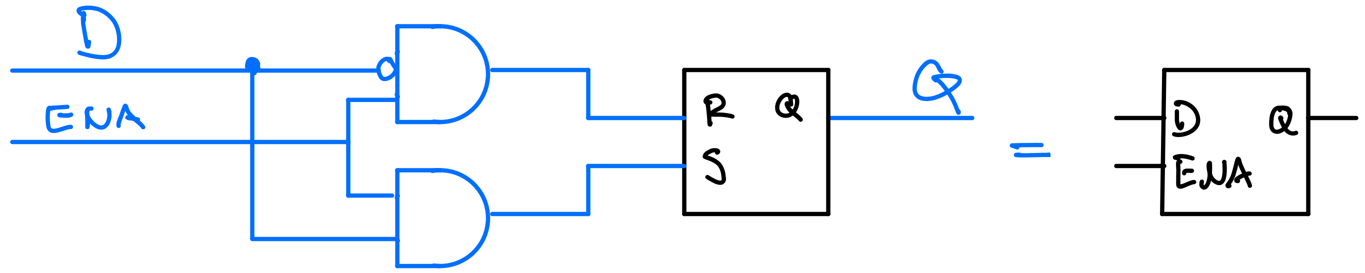
Value for r should be $r \gg R_{LOAD}$ so very little current is diverted before latch

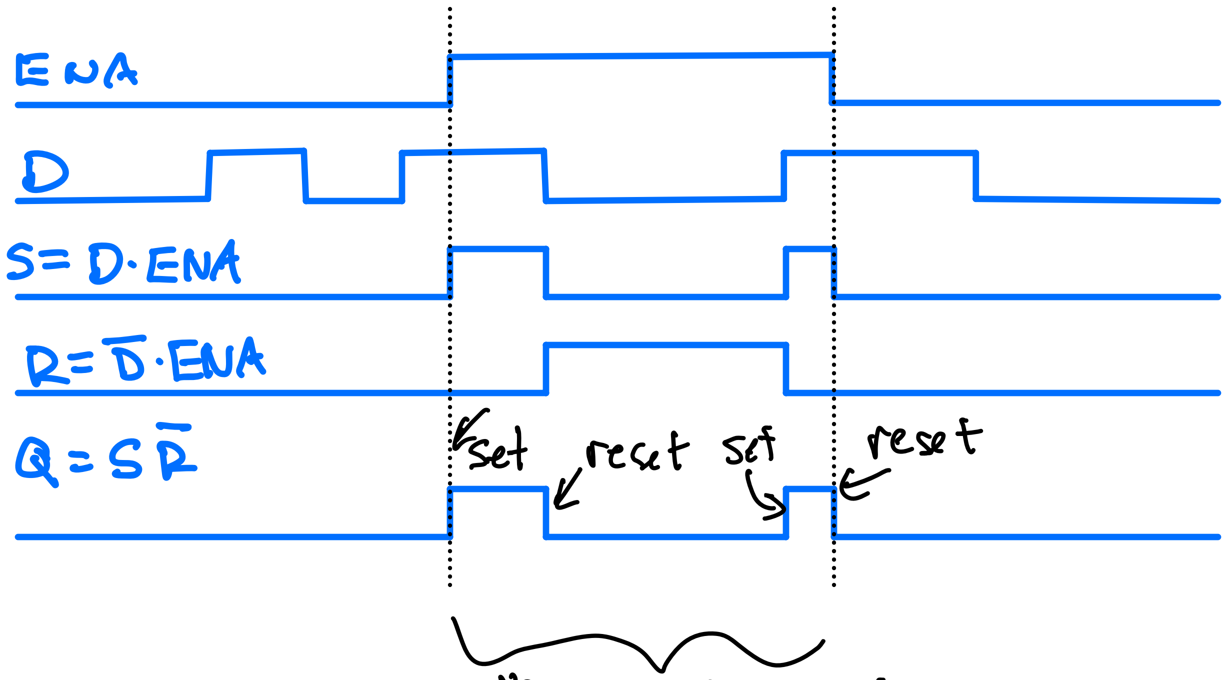
Gated R-S latch

control when R-S is active: enable signal



Gated D-latch - uses data input to set/reset



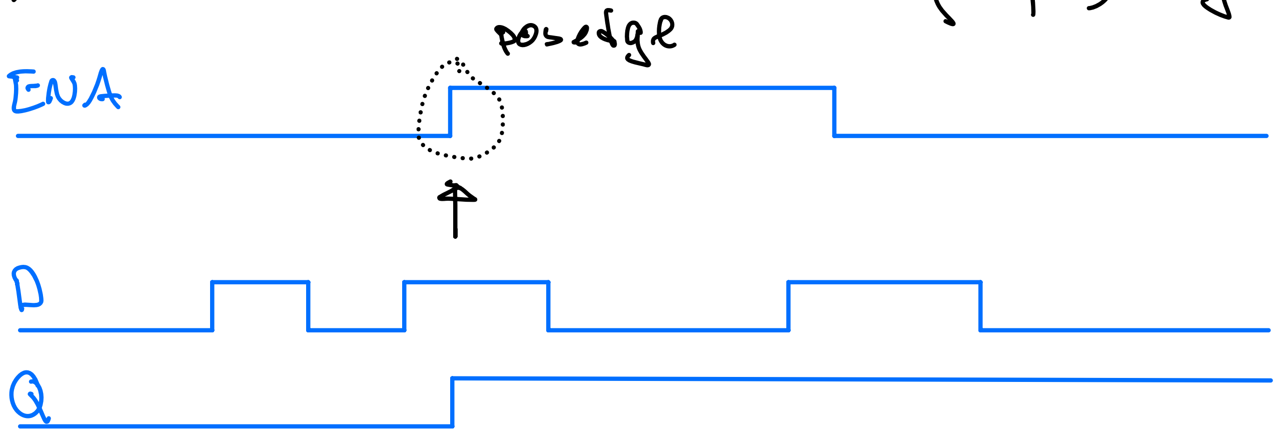


Q "follows" D when ENA is asserted

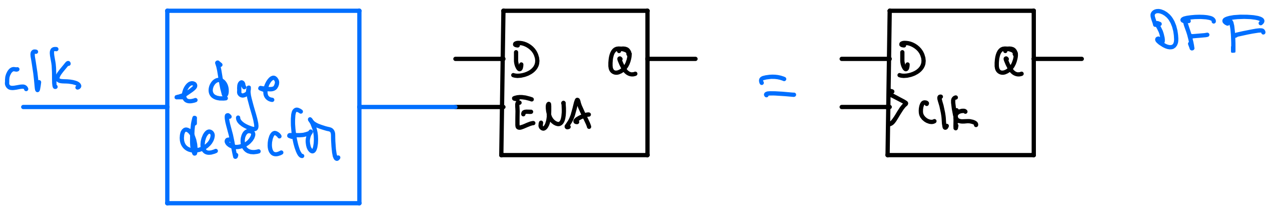
D Flip-flop (DFF)

Want Q to follow D at a specified time as opposed to a time interval, and otherwise stay latched

ex: want Q to follow D at rising (pos) edge of ENA



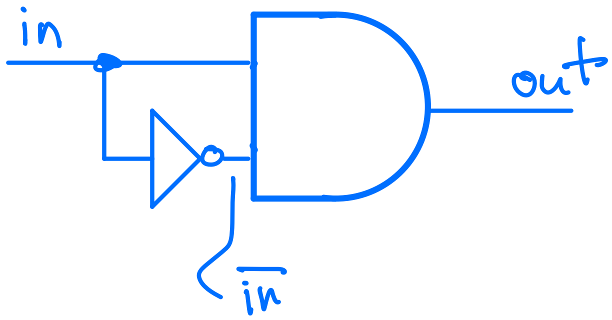
this is called an "edge-triggered" flip flop
so we need to make an "edge detector" to
drive ENA input of gated latch ≡ "D-flip-flop"

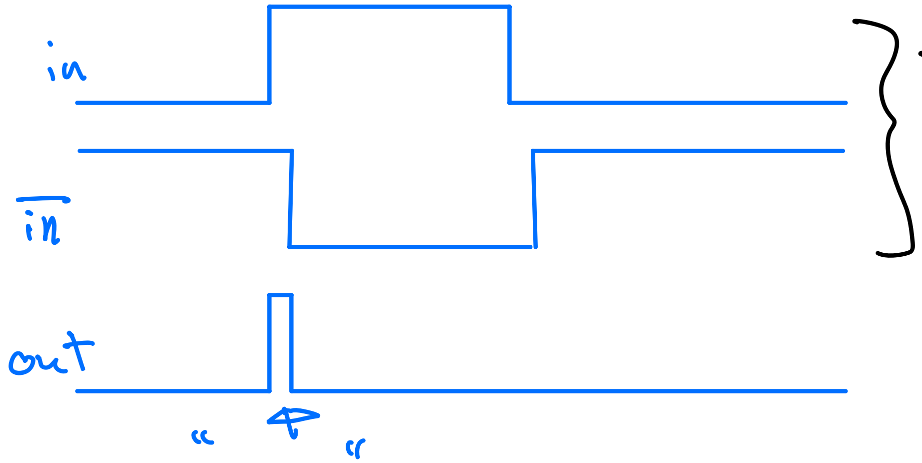


This is the basic building block of "synchronous logic"
as opposed to previous "combinatorial" logic

Synchronous logic: everything happens in sync
with a "clock" signal

But how to make edge detector? Easy!





$\overline{in}$ signal has to propagate
thru inverter.
takes small $\Delta t > 0$

Synchronous logic

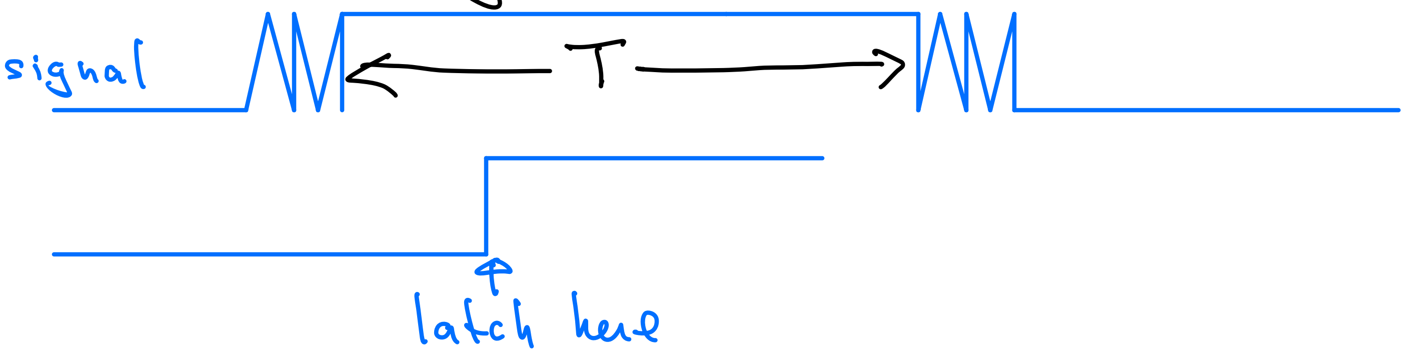
Example: A sends B 8-bits of info over an 8-bit "bus" (8 wires)
B's job is to latch the 8 signals

Problem:

1. could be noise on the lines when not being driven
2. transitions of each line won't be perfectly simultaneous

Both of these problems happen near transition time

Solution: latch the lines when things are stable, away from edges



Easy if T is large. If T ~ 100ns (f = 10 MHz), easy to find region to latch

If each bit is transitioning at 10MHz and sending 8 bits then bit rate = 80Mb/s = 10MB/s (byte/sec)
At 100MHz, 10ns per transition, 100MB/s for 8 bits

⇒ even 10ns is "easy" to handle

What if you want 1GB/s rates?

Can go from 8 bits to 80 bits, add more lines!
and still run at 100MHz per line

But now you have larger probs that an edge will fluctuate, so single-bit-errors will increase

solution: serial transmission (later)

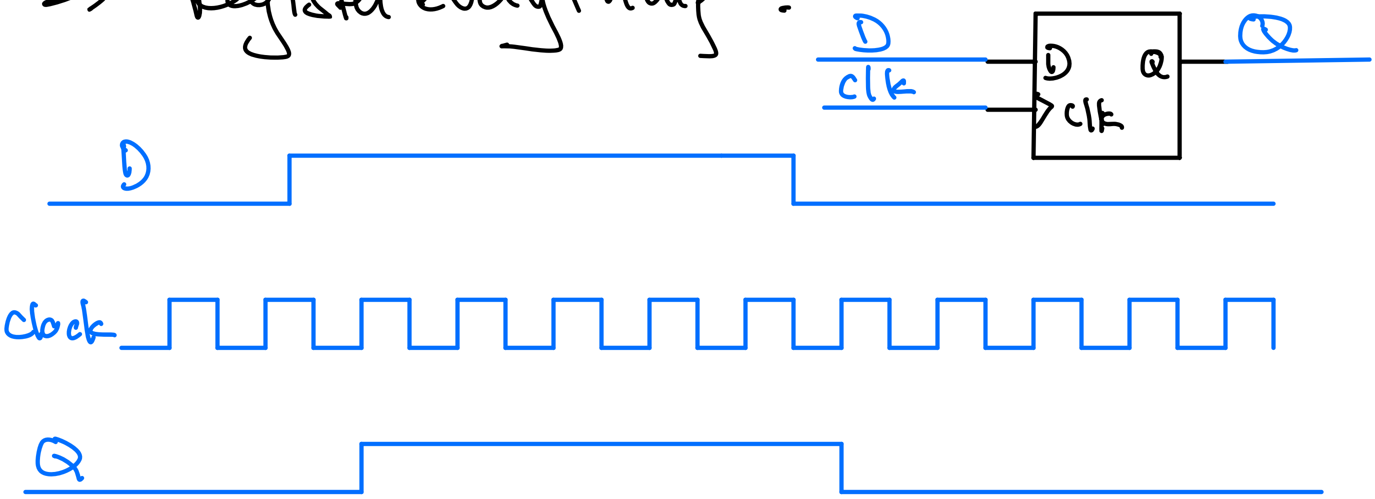
Synchronizing

You have a circuit w/ many inputs

want to implement logic on those signals, all making transitions

Want to use DFF's & latches inside circuit, need to know where the transitions are

⇒ "Register everything"!



Q is now synchronous w/ clock