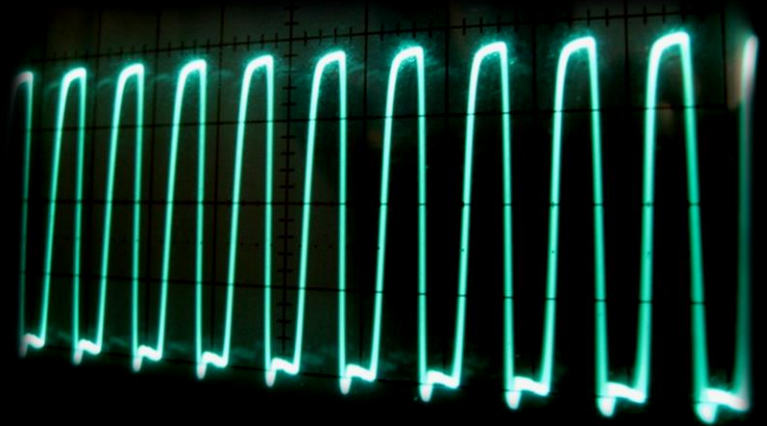


# Week 4, part D: Clocks and clocked latches



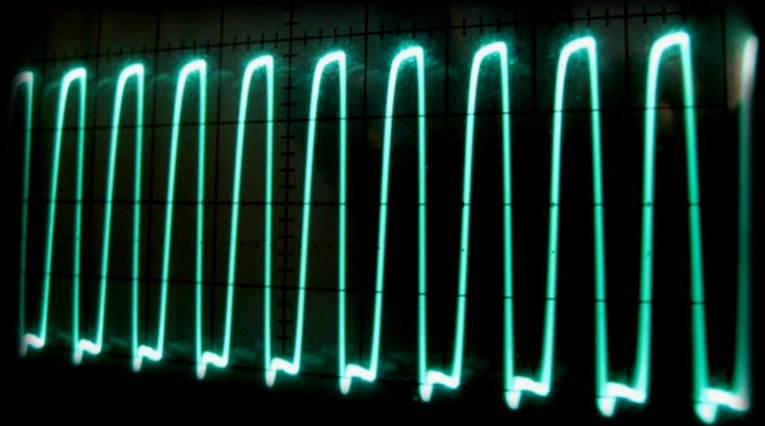
# Why A Clock



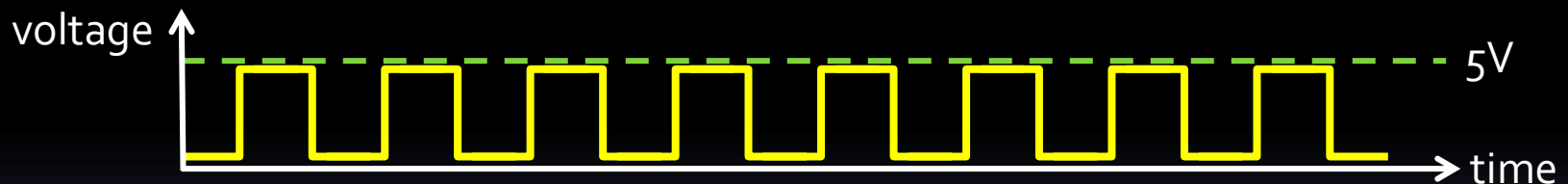
- How do we distinguish between:
  - “high 5 times in a row”
  - “high 10 times in a row”
- What do “5 times” or “10 times” mean?
- Need a way to tell when a signal (input or output) is “ready” to be sampled



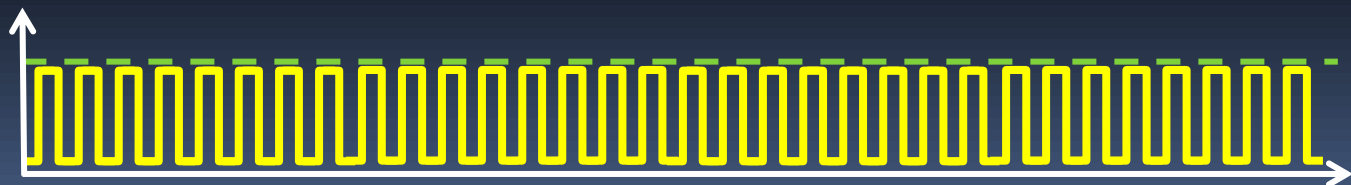
# Clock Signals



- **Clock**: a regular signal, where high value indicates that the output of the latch may be sampled.
- Usually drawn as:



- But looks more like:



# Clock Frequency

- **Frequency** = the number of pulses occur per second.
  - measured in Hertz (Hz).
- Higher frequency → can do more every second.

5 Hz



1 second

19 Hz



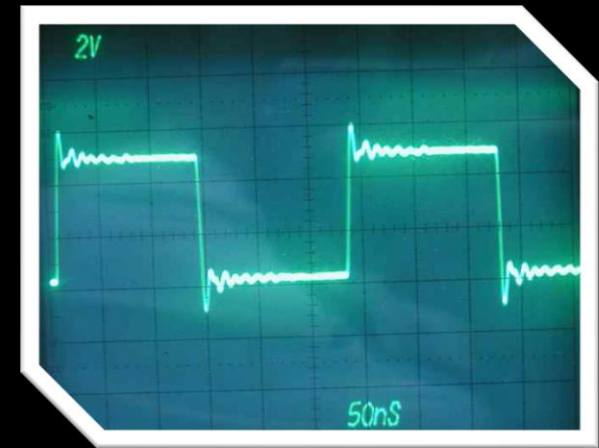
1 second



# Signal Restrictions

- What limits the clock frequency?

- Circuit complexity
  - Max propagation delay!
- Physical limits
  - Latency of transistors
- Manufacturing variations
- Physical clock limits
  - Gibbs phenomenon
  - Jitter



Beyond the scope  
of CSCB58

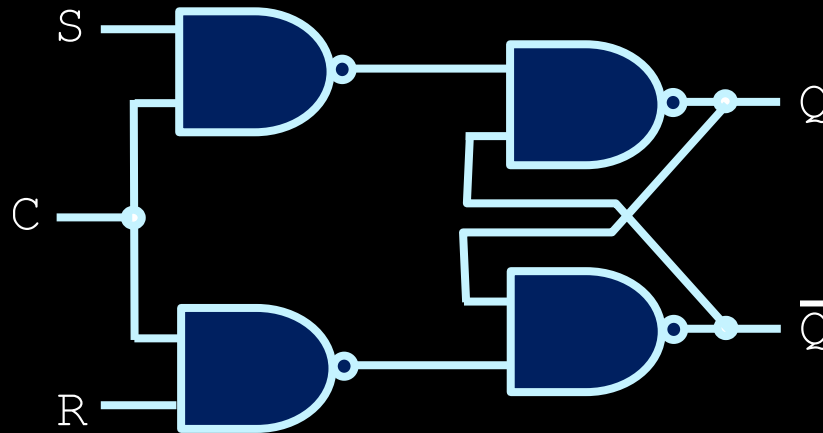


# Signal Restrictions

- CPU makers try to increase it every year.
- 40 years ago: 1 Mhz
  - Apple II – MOS 6502
  - C64 – MOS 6510
- Today: 3-5 GHz
- x 5000 increase in 40 years!



# Clocked SR latch

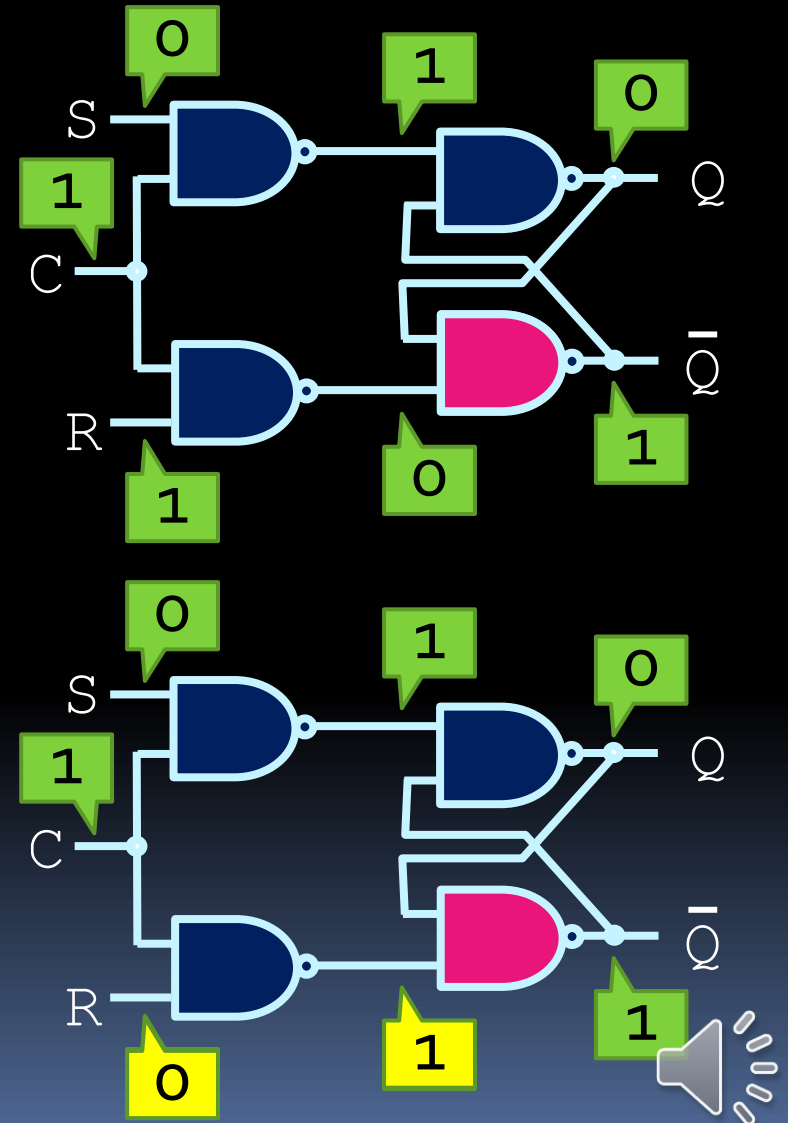


- Adding another layer of NAND gates to the  $\bar{S}\bar{R}$  latch gives us a **clocked SR latch** (or **gated SR latch**)
- Basically, a latch with a control input signal C.
- The input C is often connected to a clock signal



# Clocked SR latch behaviour

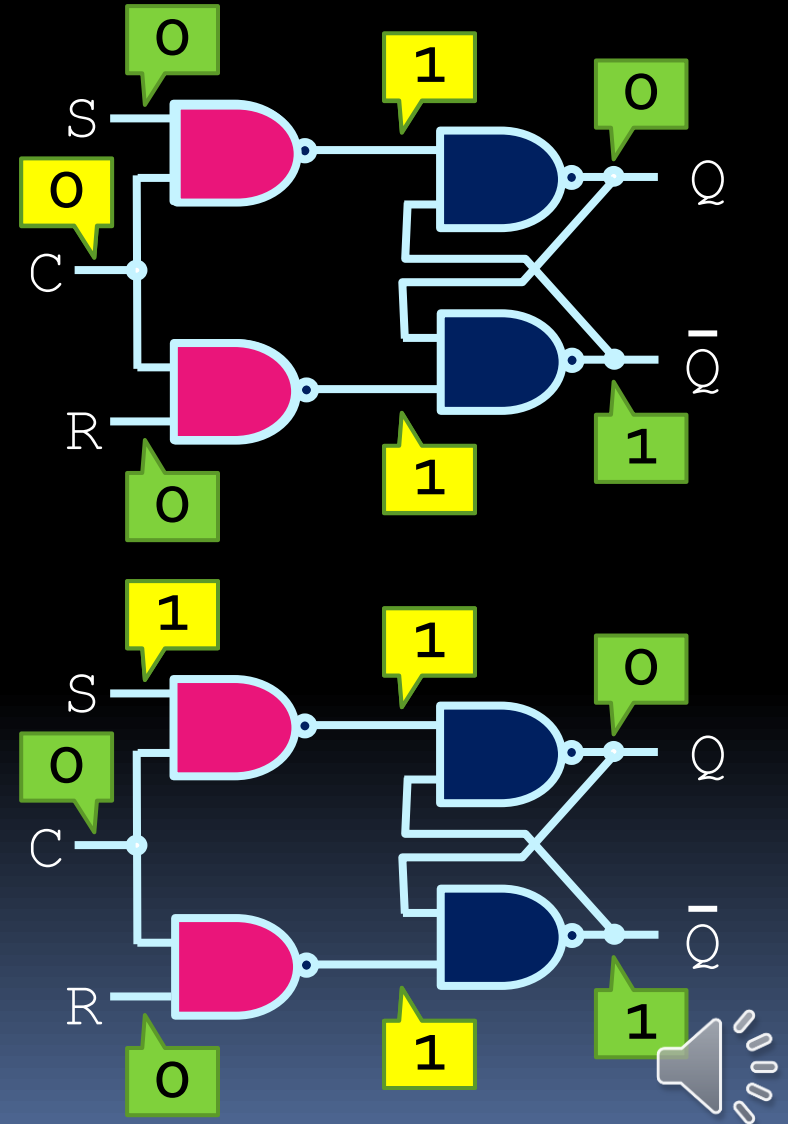
- Same behaviour as SR latch, but with timing:
  - Start off with  $S=0$  and  $R=1$ , like earlier example.
  - If clock is high, the first NAND gates invert those values, which get inverted again in the output.
  - Setting both inputs to 0 maintains the output values.



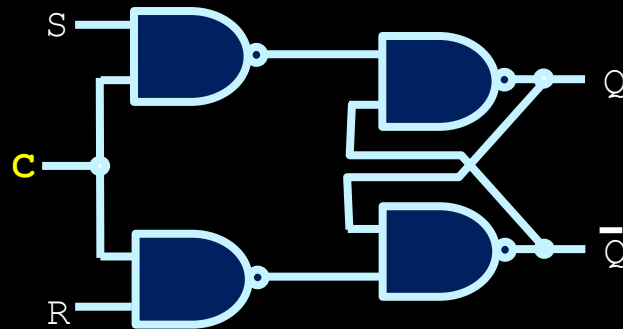


# Clocked SR latch behaviour

- Continued from previous:
  - Now set the clock low.
  - Even if the inputs change, the low clock input prevents the change from reaching the second stage of NAND gates.
  - Result: the clock needs to be high in order for the inputs to have any effect.



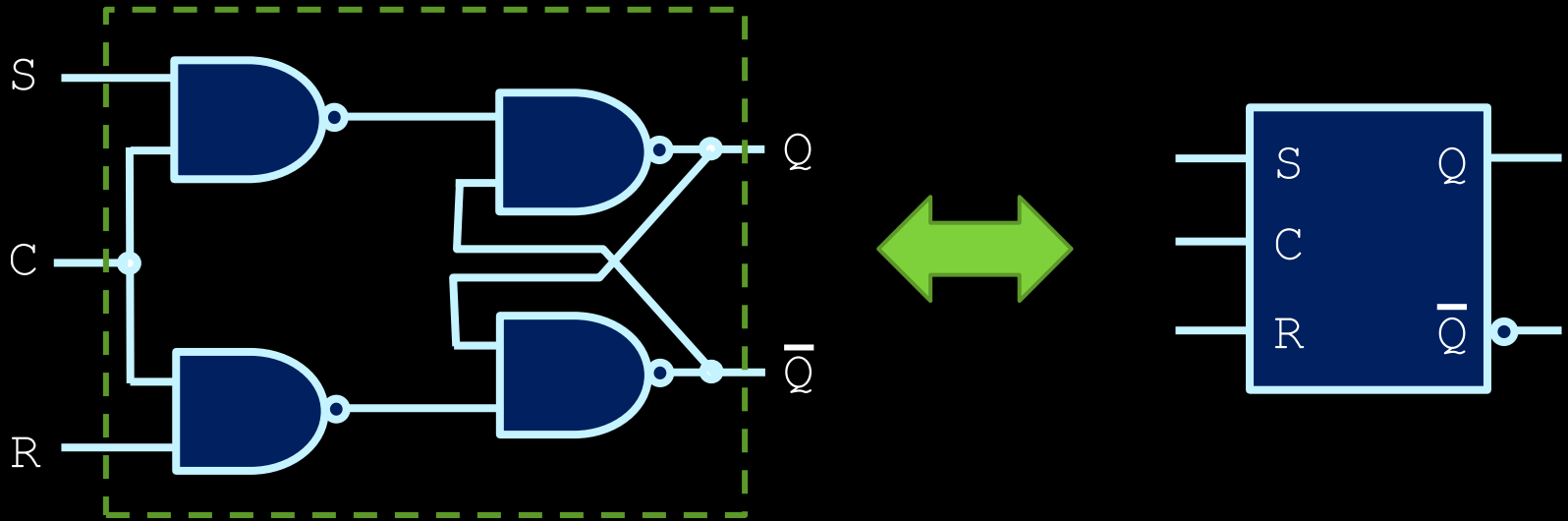
# Clocked SR latch - summary



The S and R signals are only allowed to affect the circuit when the clock input (C) is high.

- When clock is high, behave like a SR latch.
- When clock is low, S and R are blocked and there is no way to change the output.

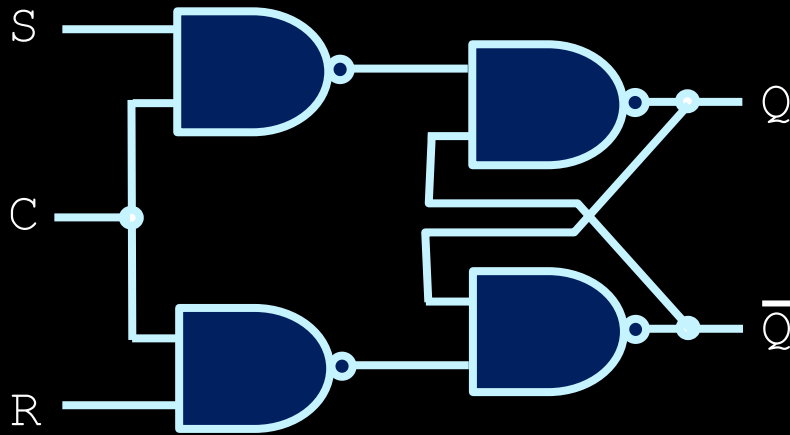
# Clocked SR latch



- This is the typical symbol for a clocked SR latch.
- Note: the small NOT circle after the  $Q$  output is simply the notation to use to denote the inverted output value. It's not an extra NOT gate.



# Clocked SR latch behaviour

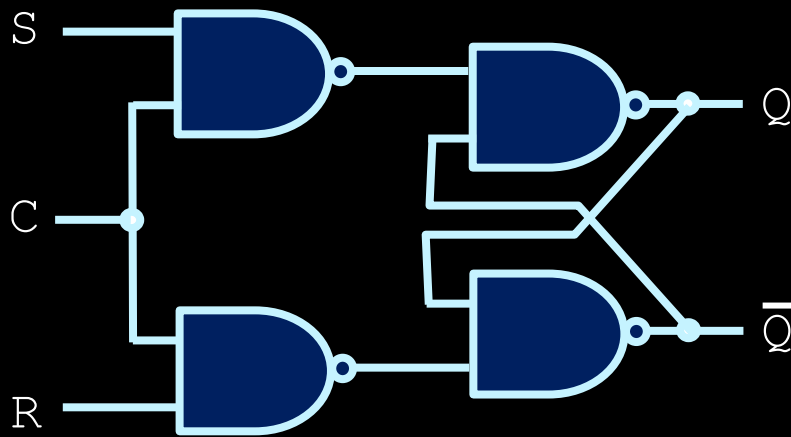


$Q_T$	S	R	$Q_{T+1}$	Result
0	0	0	0	no change
0	0	1	0	reset
0	1	0	1	set
0	1	1	?	???
1	0	0	1	no change
1	0	1	0	reset
1	1	0	1	set
1	1	1	?	???

- Wait!
- Where's the clock?
- There's a better way to look at this....



# Clocked SR latch behaviour



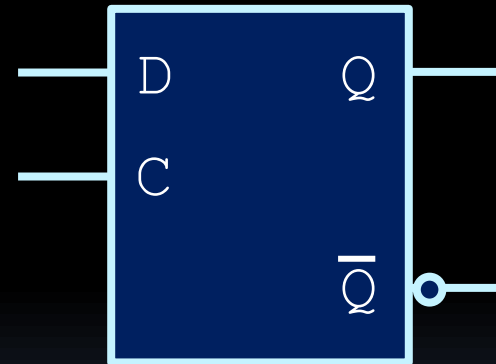
C	S	R	$Q_{T+1}$	Result
0	X	X	$Q_T$	no change
1	0	0	$Q_T$	no change
1	0	1	0	reset
1	1	0	1	set
1	1	1	?	Undefined

- Assuming the clock is 1, we still have a problem when S and R are both 1, since the state of Q is indeterminate.
- A better design** would prevent S and R from both going high at the same time.

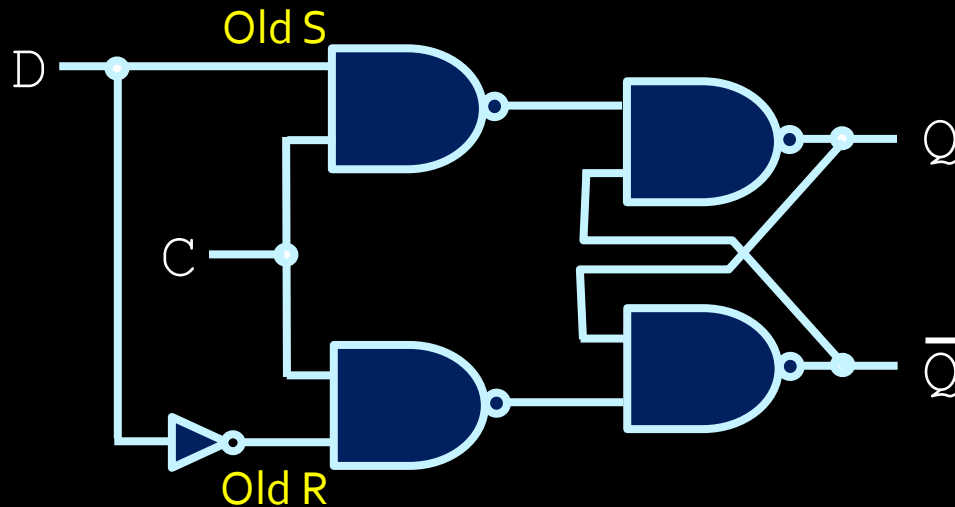


# D latch

- Prevent S and R from going high at the same time.
- How?



# D latch

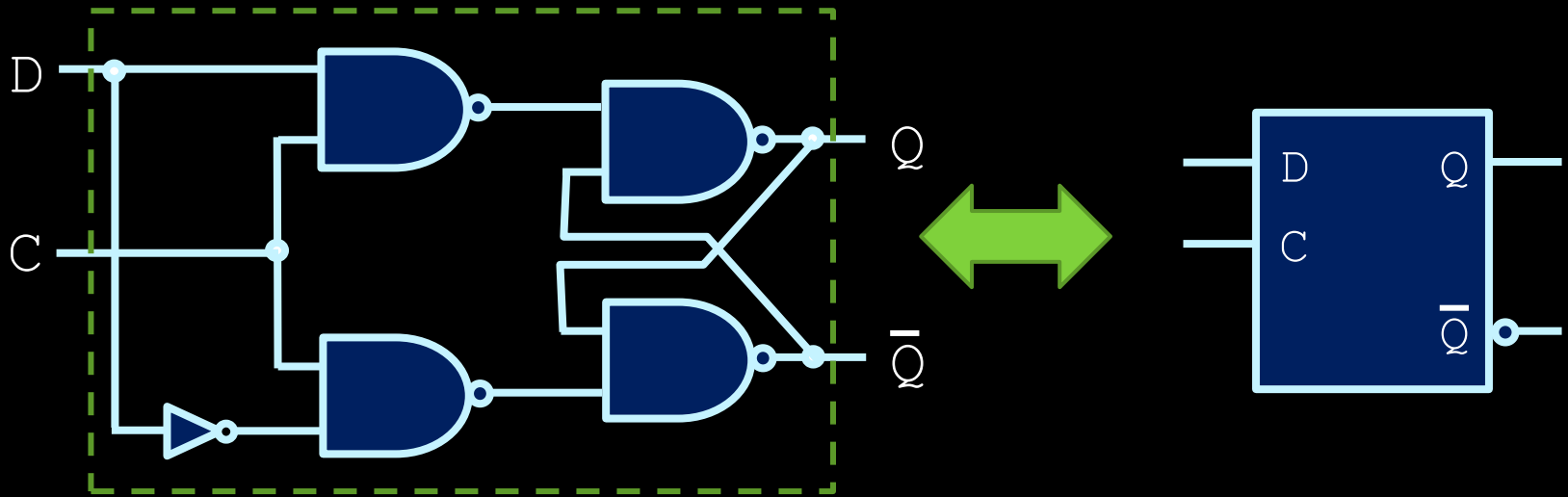


$Q_T$	D	$Q_{T+1}$
0	0	0
0	1	1
1	0	0
1	1	1

- By making the inputs to R and S dependent on a single signal D, you avoid the forbidden input problem – no more race condition.
- Input D now directly sets output Q (when C is high).
  - D latch is always clocked (makes no sense otherwise!)



# D latch



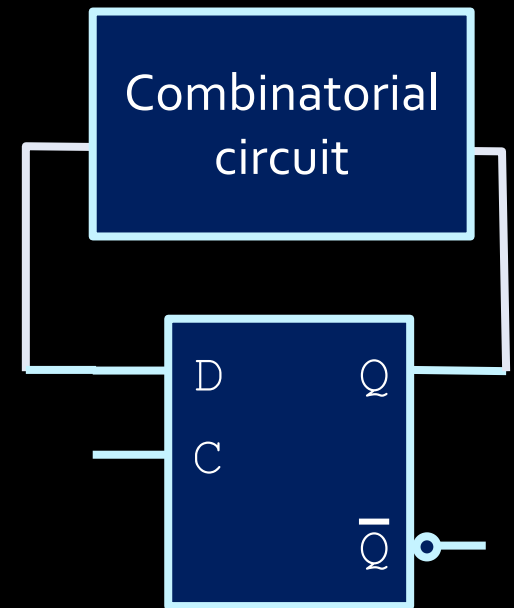
- This design is good!
  - ▣ Easy to store a bit: just set D to what you want to store.
  - ▣ Can maintain state as long as C is low
  - ▣ No weird forbidden inputs.





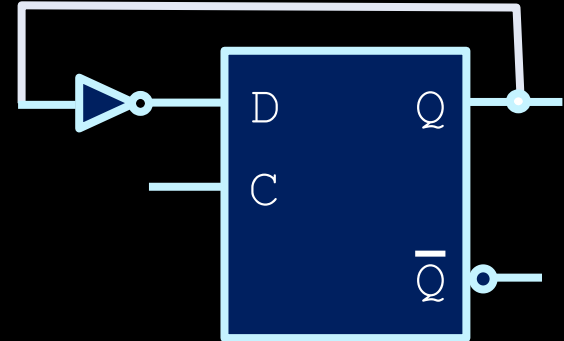
# Timing issues

- Can we use D latch to build circuits with feedback?
- Unfortunately, no.
- The D latch has **timing issues**
- The main issue is that changes in D are almost immediately reflected in Q



# Latch timing issues

- Consider the circuit on the right:
- When the clock signal is high, the output looks like the waveform below:



- Output keeps toggling back and forth.

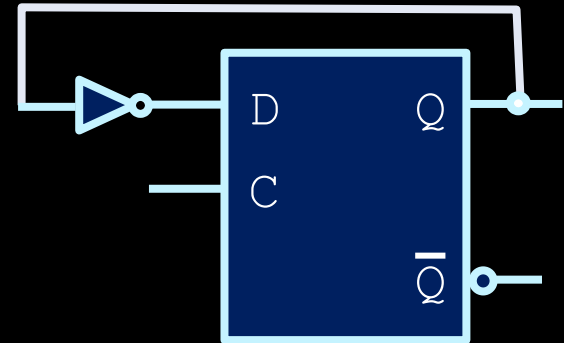


...what happens next?

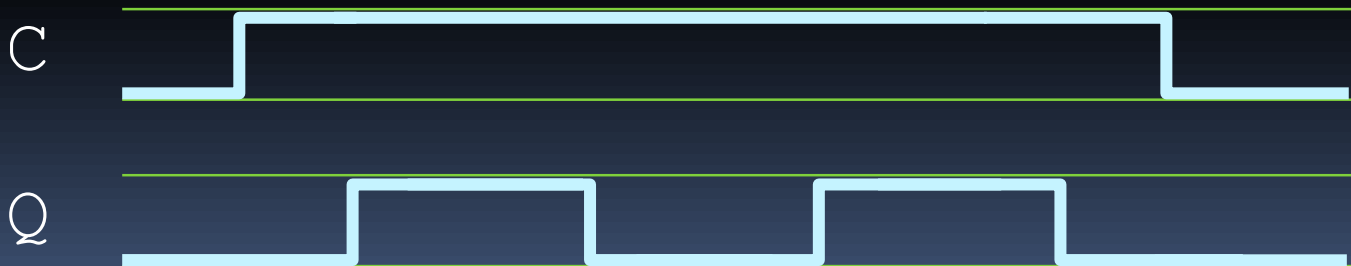


# Latch timing issues

- Consider the circuit on the right:
- When the clock signal is high, the output looks like the waveform below:



- Output keeps toggling back and forth.



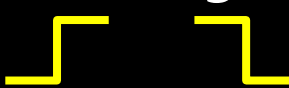
# D-Latch is transparent!

- **Transparent** means that
  - Any changes to its inputs are visible to the output when control signal (Clock) is 1.
- **Key Take-away:**

The output of a latch **should not** be applied directly or through combinational logic to the input of the same or another latch when they all have the same control (clock) signal.



# Fixing latch timing issues

- Preferable behaviour:
  - ▣ Have output change only once when the clock changes. 
- Solution: create **disconnect between circuit output and circuit input**, to prevent unwanted feedback and changes to output.
- More on this in part E

