Sequential Logic

How do we store a bit?

Combinational vs. Sequential

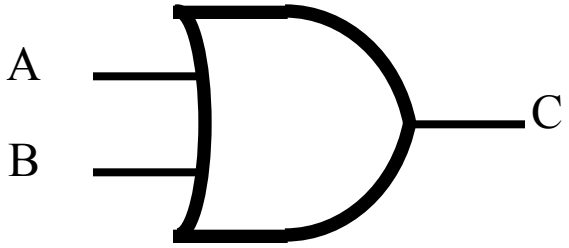
- Combinational: output depends completely on the value of the inputs.
  - time doesn't matter.

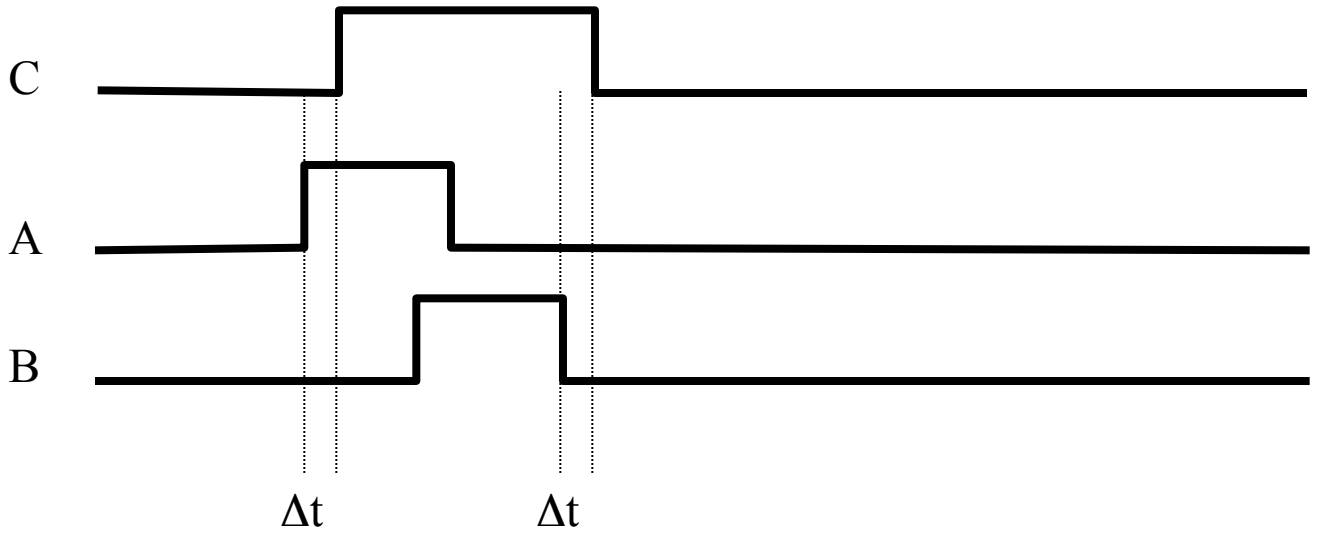
- Sequential: output also depends on the *state* a *little while ago*.
  - can depend on the value of the output some time in the past.

Memory

- Think about how you might design a combinational circuit that could be used as a single bit *memory.*
- Use your *memory* to recall that the output of a gate can change whenever the inputs change.

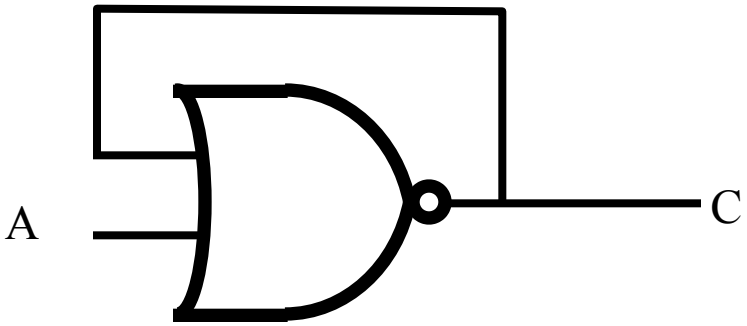
Gate Timing





A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0

Feedback



What happens when A changes from 1 to 0?

- *Try connecting the microphone input to the earphone output on your PC.*
- *Try holding a mirror in front of you while you are looking in a mirror.*
- *Try using google.com to search for the term "google".*

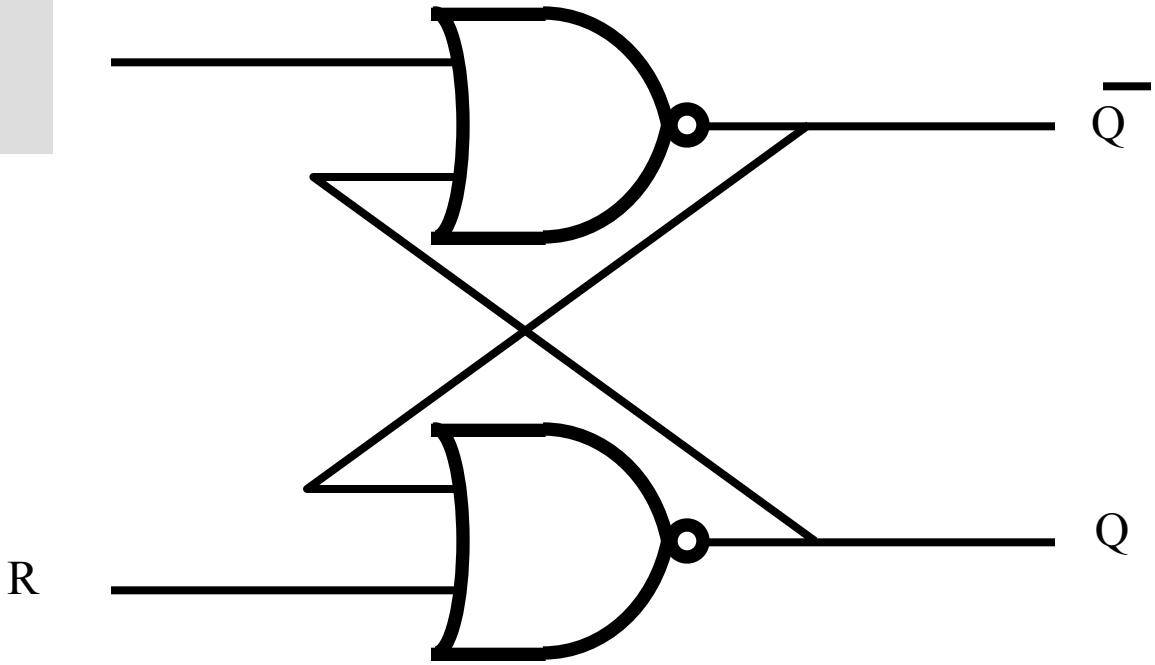
Caution: the above experiments may result in opening a worm hole in your immediate surroundings!

Feedback can be stable

- It is possible to use feedback in a circuit that is predictable and *stable*.
- It is harder to tell what is going on (when compared to combinational circuits), but it is possible:
  - figure out what the output is at some initial time.
  - use that output to determine the next output.
  - continue until the worm hole contracts…

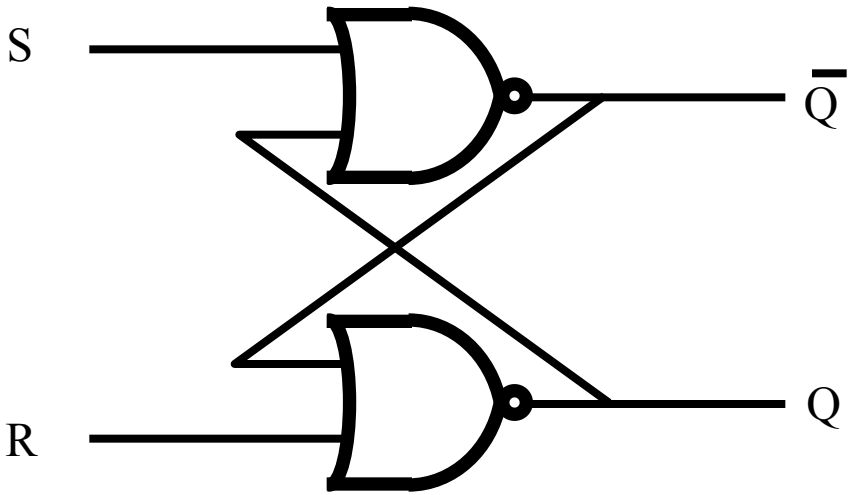
S-R latch

A	B	A nor B
0	0	1
0	1	0
1	0	0
1	1	0



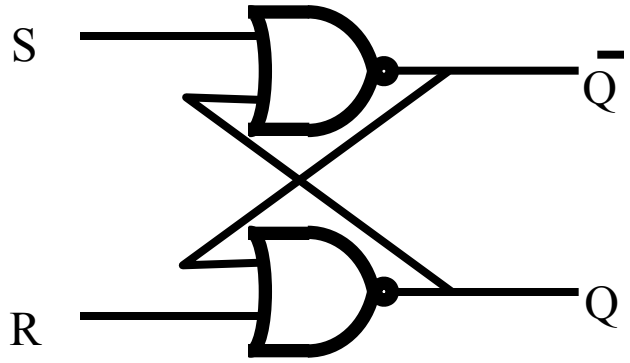
S stands for “Set” as in “Set to a 1”
R stands for “Reset” (set to a 0).

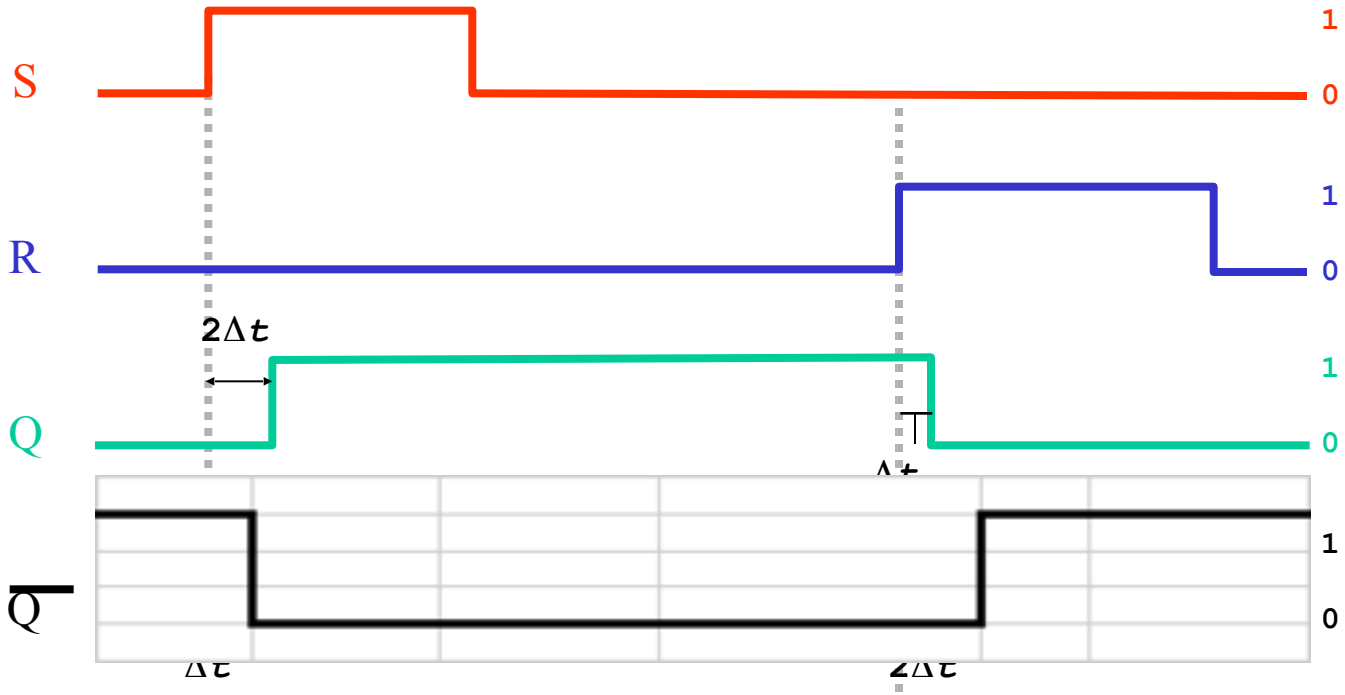
S-R latch Truth Table



Q_t	S_t	R_t	Q_{t+1}
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	?
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	?

S-R latch Timing





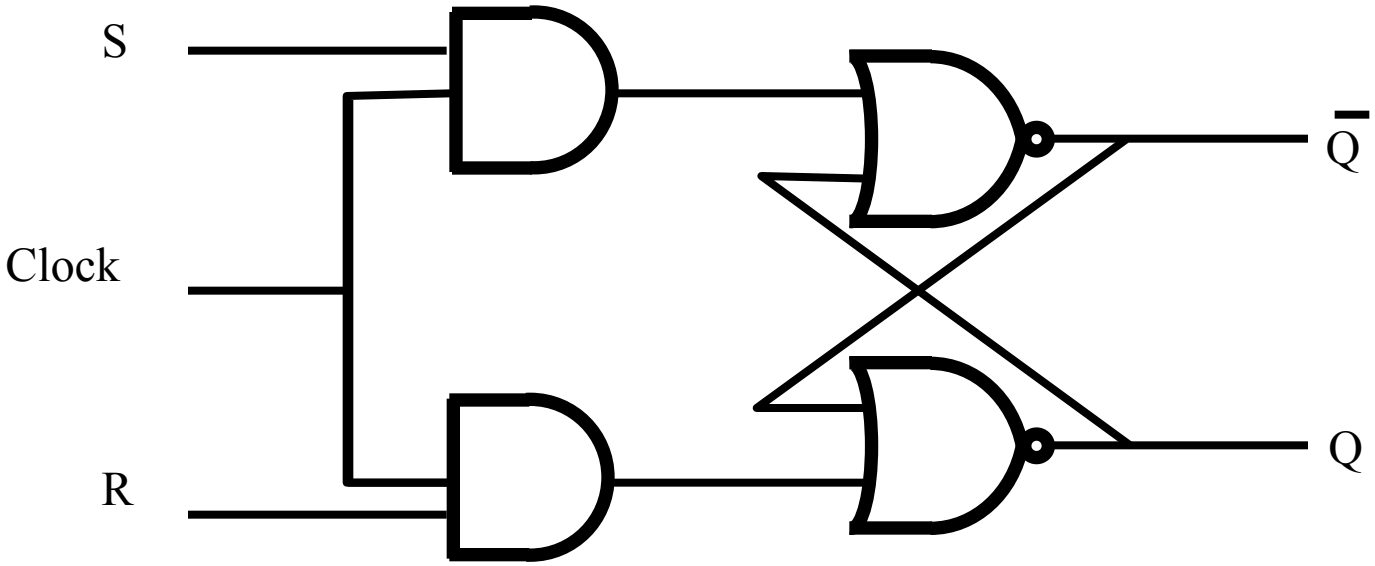
Δt = *One Gate Delay*

Clocked S-R Latch

- Inside a computer we want the output of gates to change only at specific times. All ops. in a computer are synchronized.

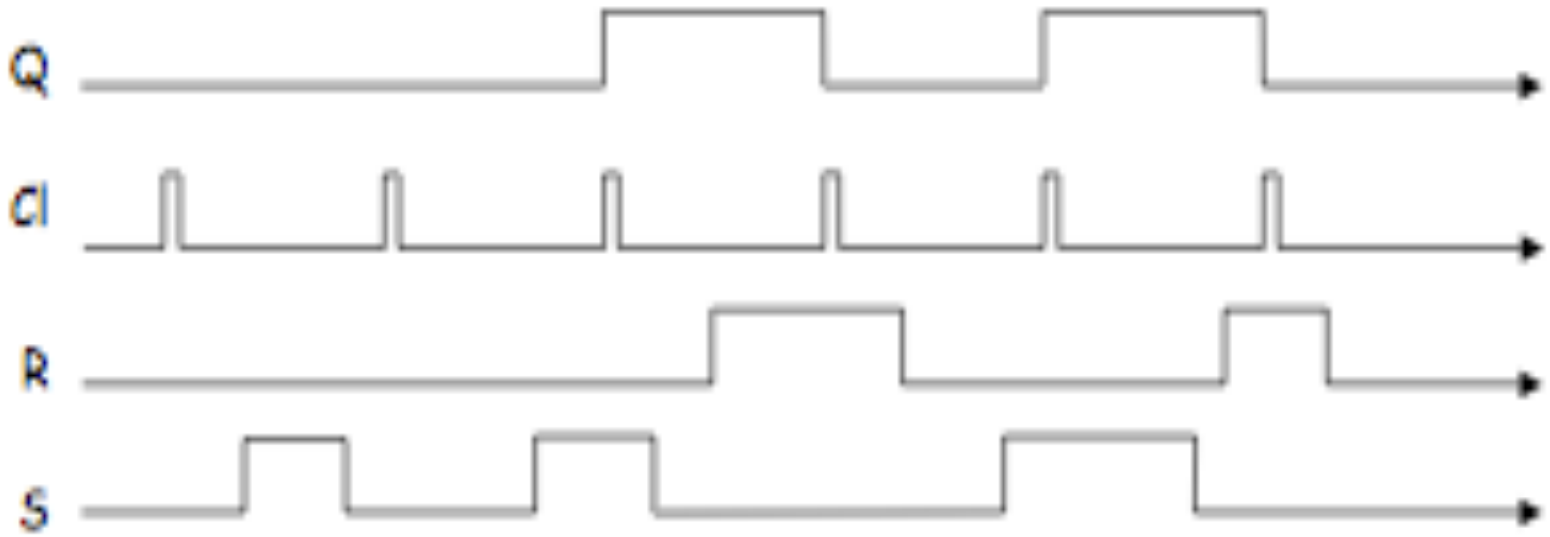
- We can add some circuitry to make sure that changes occur only at the *clock* edges (when the clock changes from 0 to 1).

Clocked S-R Latch



- Q only changes when the Clock is a 1.
- If Clock is 0, neither S or R *reach* the NOR gates.

Clocked SR Latch Timing Diagram example



Characteristic table for clocked SR Latch

Characteristic Eqn: Q(t+1) = Q(t)(not R) + S & S*R = 0

S(t)	R(t)	Q(t)	Q(t+1)	Condition
0	0	0	0	no change
0	0	1	1	nochange
0	1	0	0	reset to 0
0	1	1	0	reset to 0
1	0	0	1	set to 1
1	0	1	1	set to 1
1	1	0	---	forbidden
1	1	1	---	forbidden

aracteristic tables specify state of table after one clock pulse
given input and initial state.

SR Latch, Excitation table

Q(t)	Q(t+1)	S(t)	R(t)
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

X above means “does not matter”

at the current i/p must be to obtain the
nsitions Q(t) -> Q(t+1)

What if S=R=1?

- The truth table shows "--" when S=R=1.
- The value of Q is undetermined. The

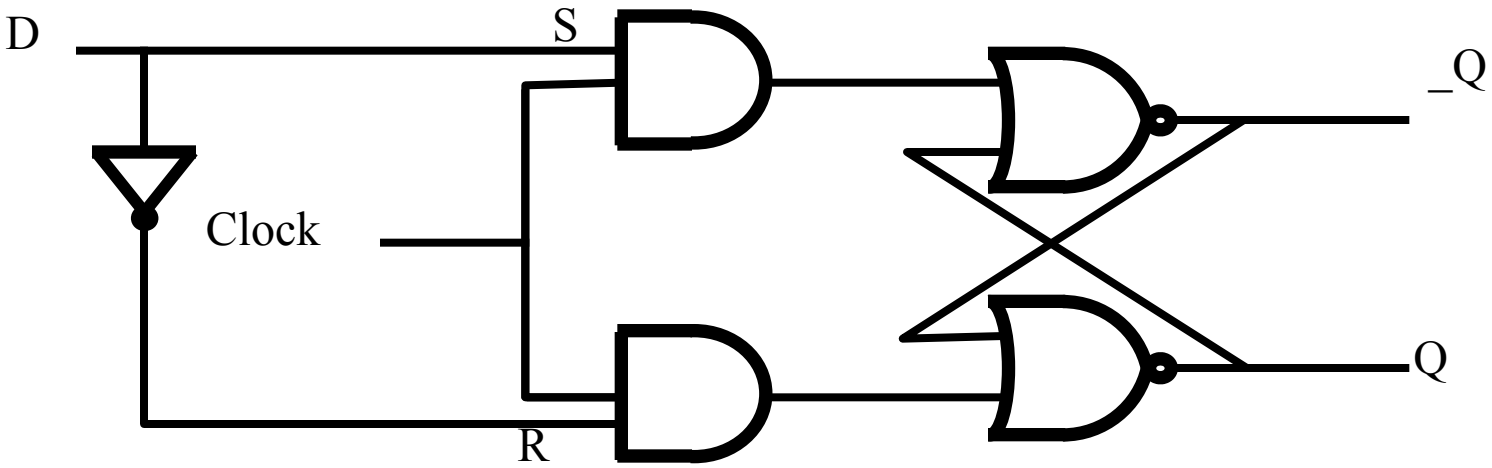
situation gets worse when SR=00: Both gates try and drive their outputs to 1. This results in a

race condition - behavior of ckt. depends upon the relative transit times of the two signals over the combinatorial paths- The circuit is not *stable*

- Now that we have a clock we can make sure that we PREVENT the S=R=1 condition.

Avoiding S=R=1: The D Latch

Clocked by definition.

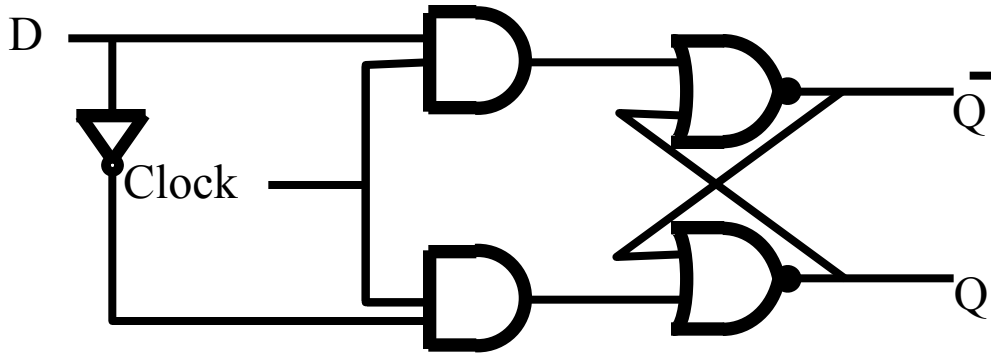


D	Clock	Q(t+1)
X	0	Q(t)
0	1	0
1	1	1

Char. Eq.: Q(t+1) = D

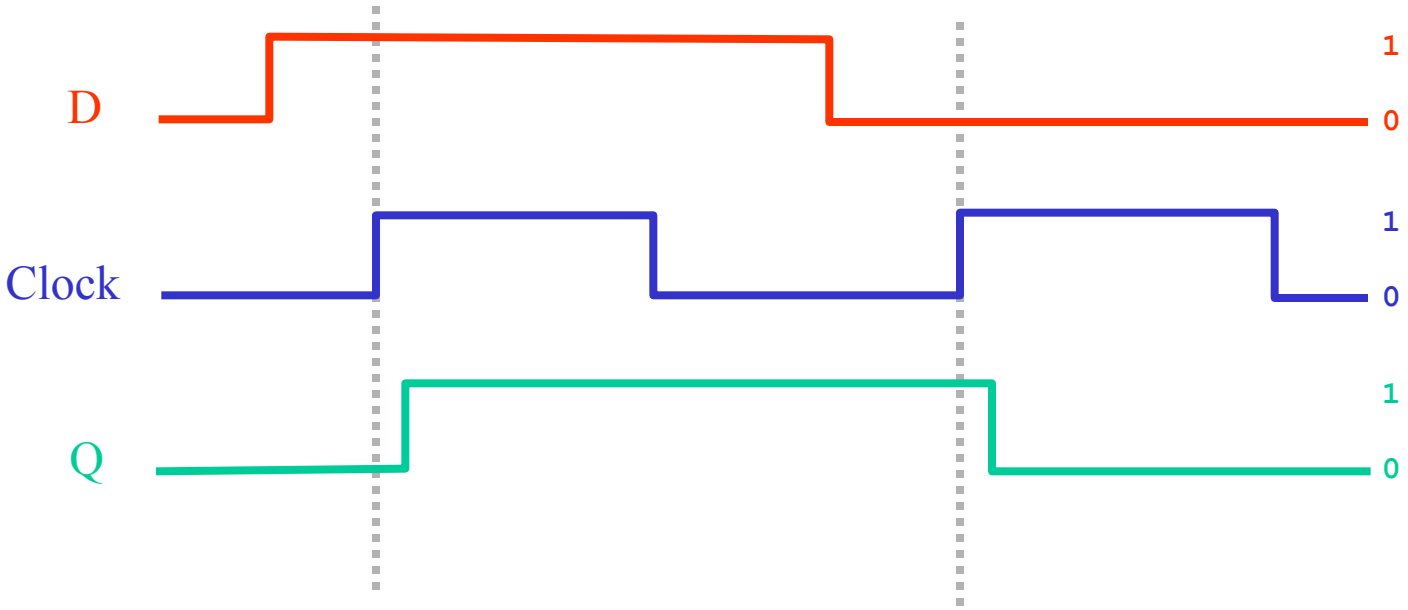
D Latch

(D: Delay or Data)



- Now have only one input: D.
- If D is a 1 when the clock changes from 1 to 0, the circuit will *remember* the value 1 (Q=1).
- If D is a 0 when the clock changes from 1 to 0, the circuit will *remember* the value 0 (Q=0).

D Latch Timing

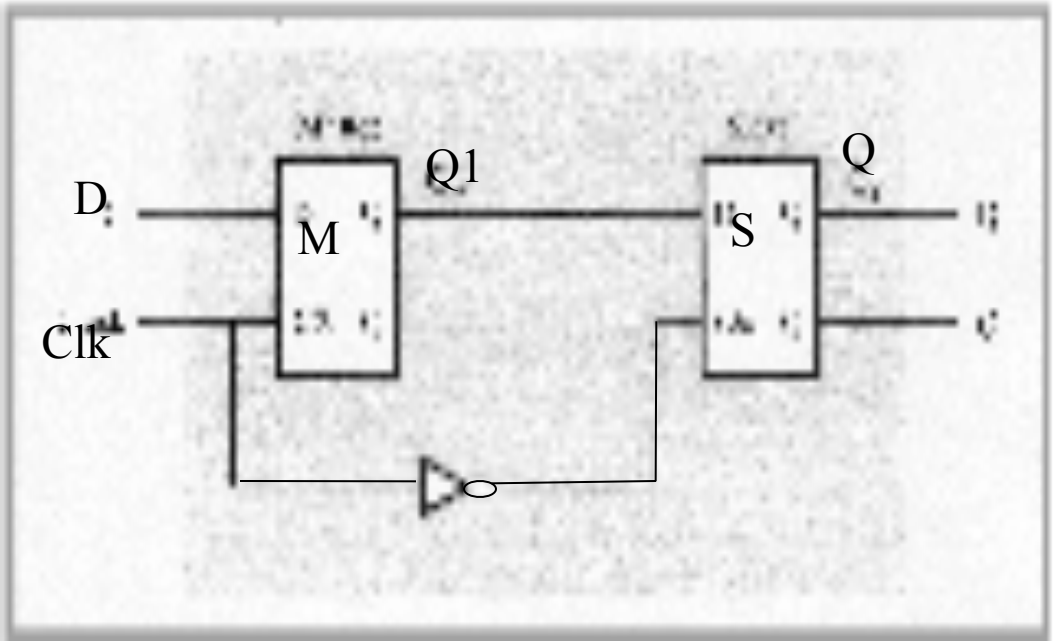


D Flip-Flops

Flip-flops: Sensitive to the clock edge.

When clock = 0, the slave latch is enabled and its o/p is equal the master o/p and the master latch is disabled.

When clock = 1, the master is enabled the data i/p is stored in the master latch. The slave however is disabled. Any changes in i/p change Q1, but the value at Q1 does not affect the slave o/p Q. When pulse returns to zero, the master is isolated from the input and the current value of Q1 is transferred to the o/p Q.

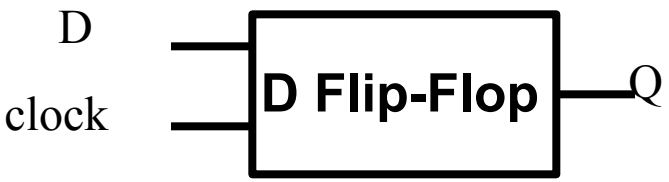


D	Clock	Q(t+1)
X	0	Q(t)
0	↓	0
1	↓	1
X	↑	Q(t)

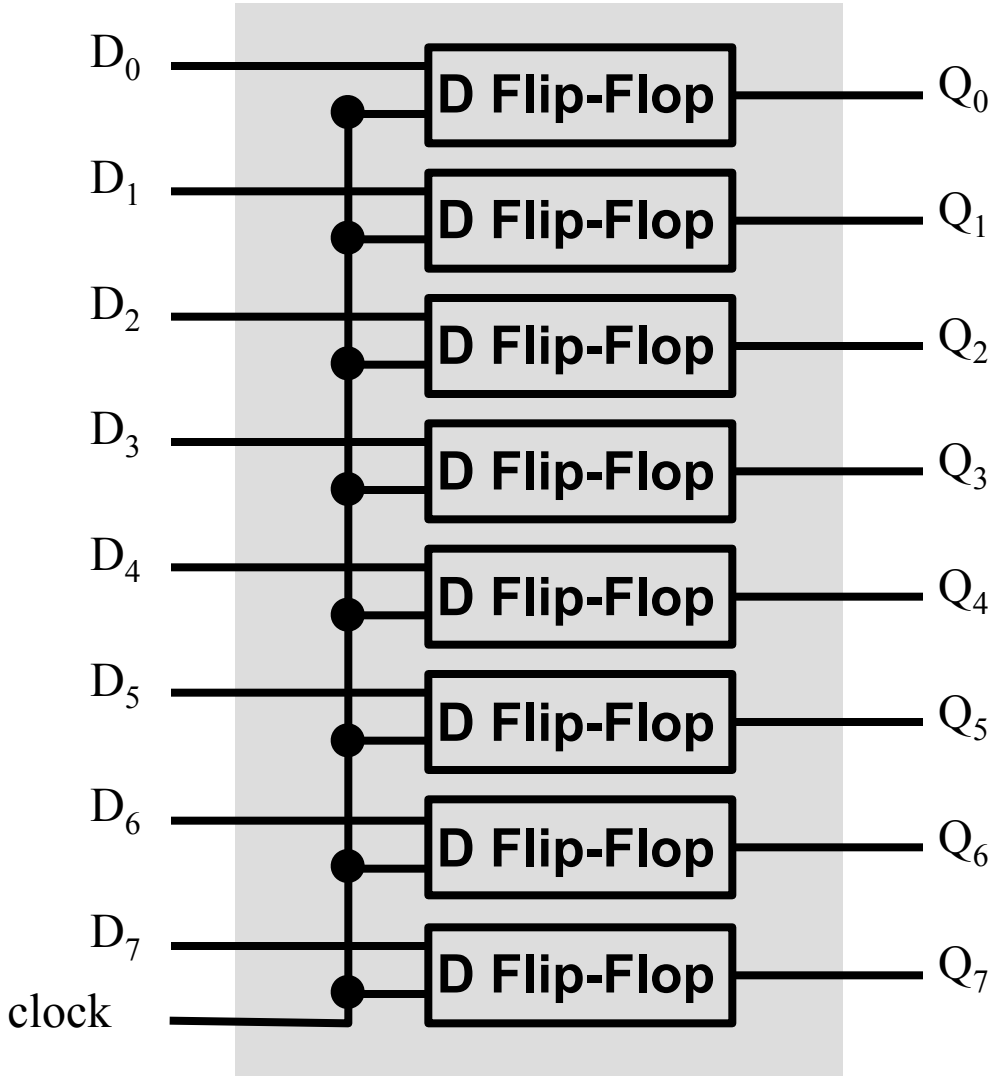
8 Bit Memory

- We can use 8xD Flip-Flops to create an 8 bit memory.

- We have 8 inputs that we want to *store*, all are *written* at the same time.
  - all 8 flip-flops use the same clock.


D
clock
D Flip-Flop
Q

8 Bit Memory


D_0
D_1
D_2
D_3
D_4
D_5
D_6
D_7
clock
D Flip-Flop
D Flip-Flop
D Flip-Flop
D Flip-Flop
D Flip-Flop
D Flip-Flop
D Flip-Flop
D Flip-Flop
Q_0
Q_1
Q_2
Q_3
Q_4
Q_5
Q_6
Q_7