COMP 122/L Lecture 24

Kyle Dewey

Outline

- Sequential circuits
- D flip-flops

Sequential Circuits

Motivation

```
Output = Input1 + Input2
```

-You’re working with a circuit that can add numbers

Motivation

```
Output = Input1 + Input2
```

```
mov r0, #5
add r1, r0, r0
```

-...however, adding numbers is only one part of the problem
-For a processor, we need a way to store values in between instructions. That is, we need registers.
-None of the components you've worked with so far do this sort of thing

Combinatorial Logic

Outputs entirely determined by the inputs

Combinatorial Logic

Outputs entirely determined by the inputs

```
Output = Input1 + Input2
```

-You've been working with combinatorial logic throughout the course so far

Combinatorial Logic

Outputs entirely determined by the inputs

```
Output = Input1 + Input2
```

Sequential Logic

Outputs determined by the inputs,
along with the **current state**

-You've been working with combinatorial logic throughout the course so far

Combinatorial Logic

Outputs entirely determined by the inputs

```
Output = Input1 + Input2
```

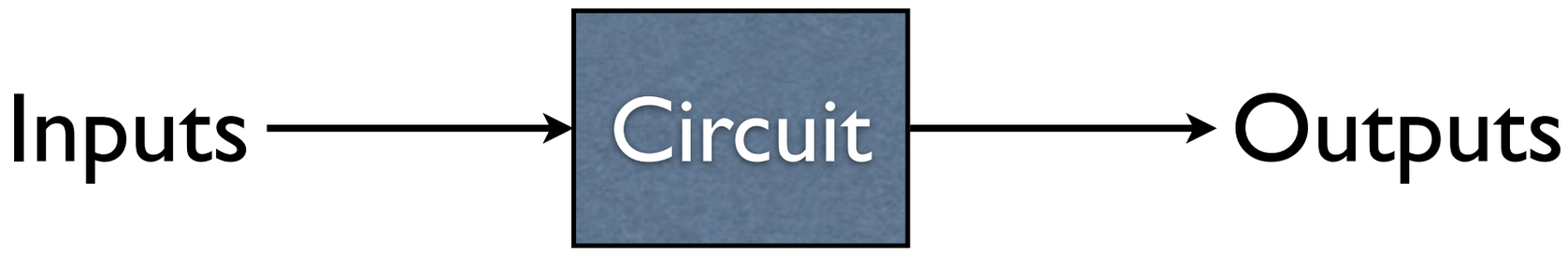
Sequential Logic

Outputs determined by the inputs,
along with the **current state**

```
mov r0, #5
add r1, r0, r0
```

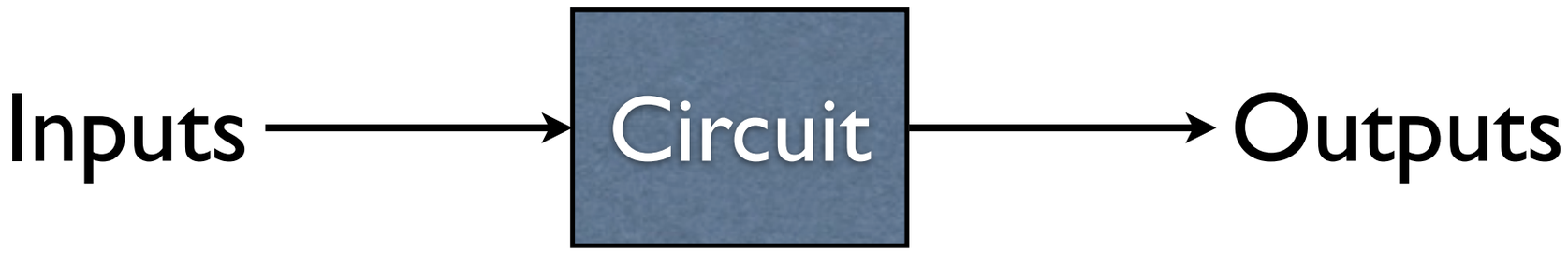
-You've been working with combinatorial logic throughout the course so far
-Registers act to hold the current state
-This ARM snippet does what it does because it remembers that r0 held 5 in between the two instructions

Combinatorial Logic



-Same information with a diagram

Combinatorial Logic



Sequential Logic

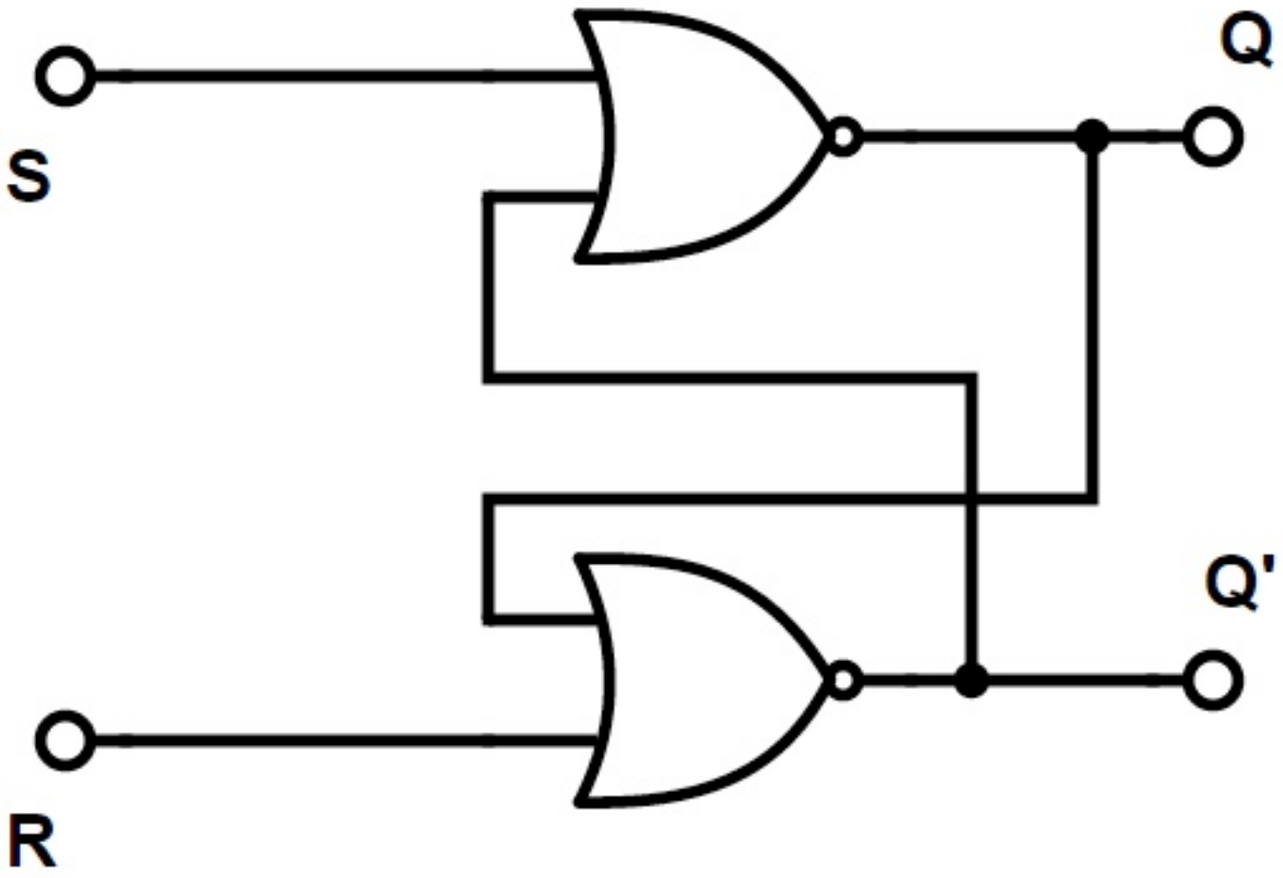
-Same information with a diagram
-Interesting point: sequential circuits contain combinatorial circuits. This isn't an entirely independent topic; it builds on everything you've seen and used so far

Saving Bits

We can utilize *feedback*: putting the output of the circuit back into itself.

Saving Bits

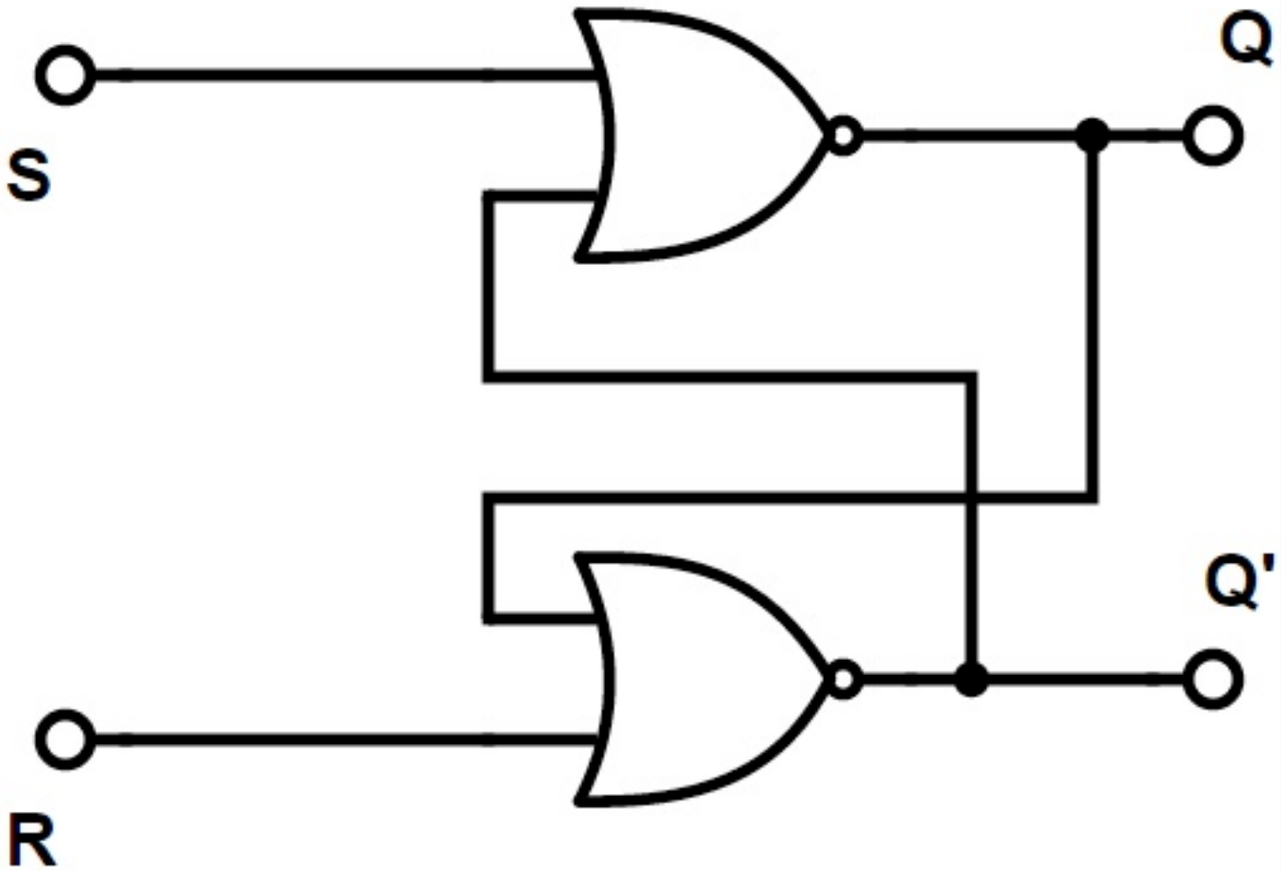
We can utilize *feedback*: putting the output of the circuit back into itself.



(SR Latch)

Saving Bits

We can utilize *feedback*: putting the output of the circuit back into itself.



(SR Latch)

S	R	Next Q
0	0	Q
0	1	0
1	0	1
1	1	X

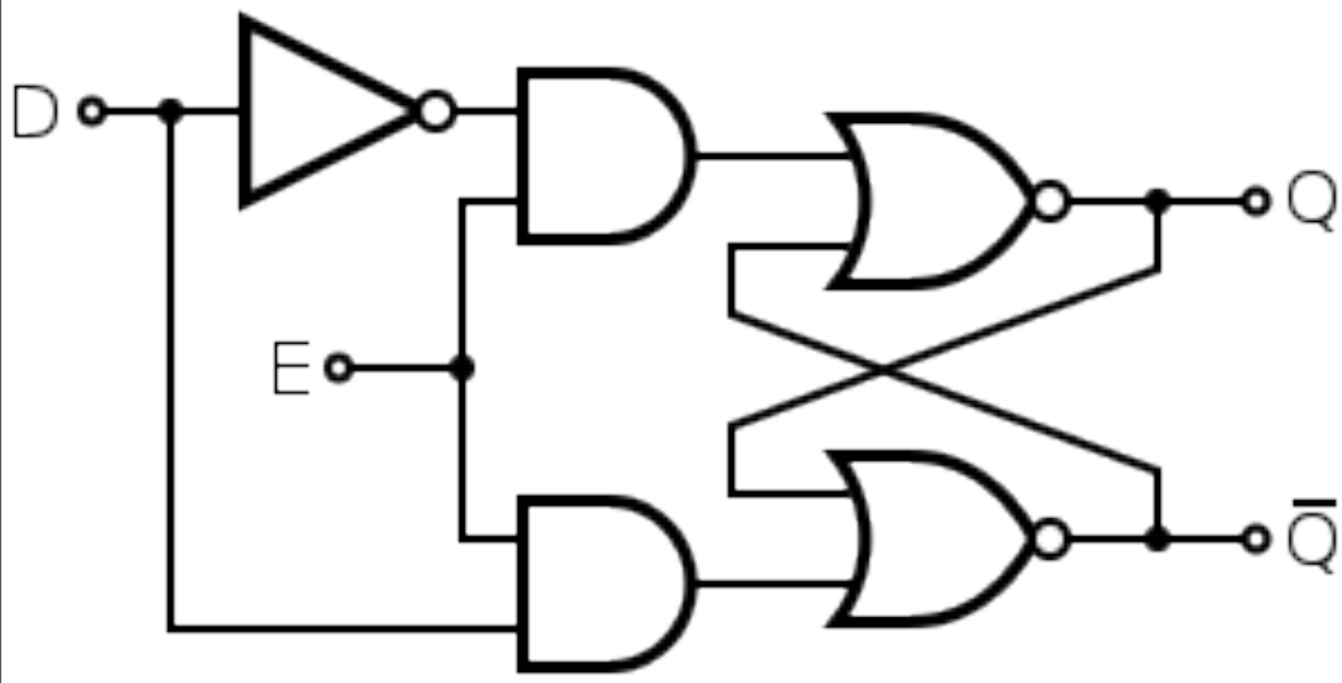
-The animations for SR NOR latches are really good here (https://en.wikipedia.org/wiki/Flip-flop_(electronics)#D_flip_flop)

Building Up

Can use this to store any bit

Building Up

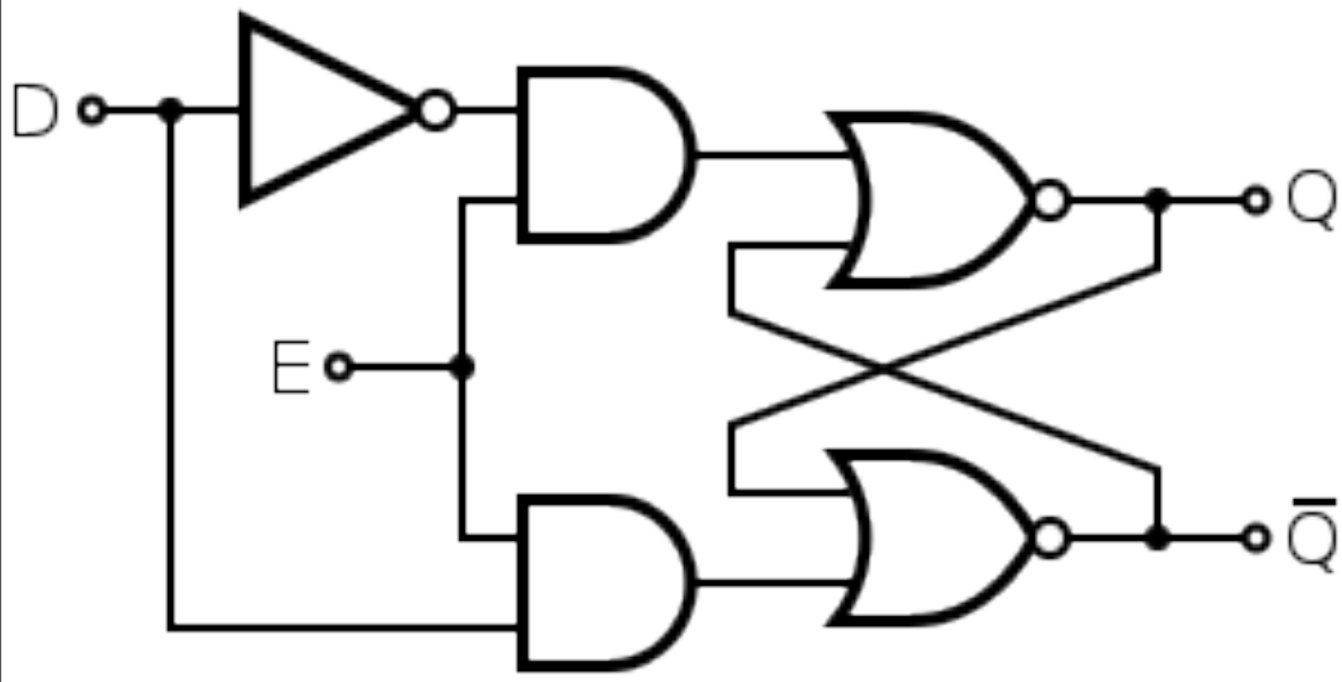
Can use this to store any bit



(D Latch)

Building Up

Can use this to store any bit



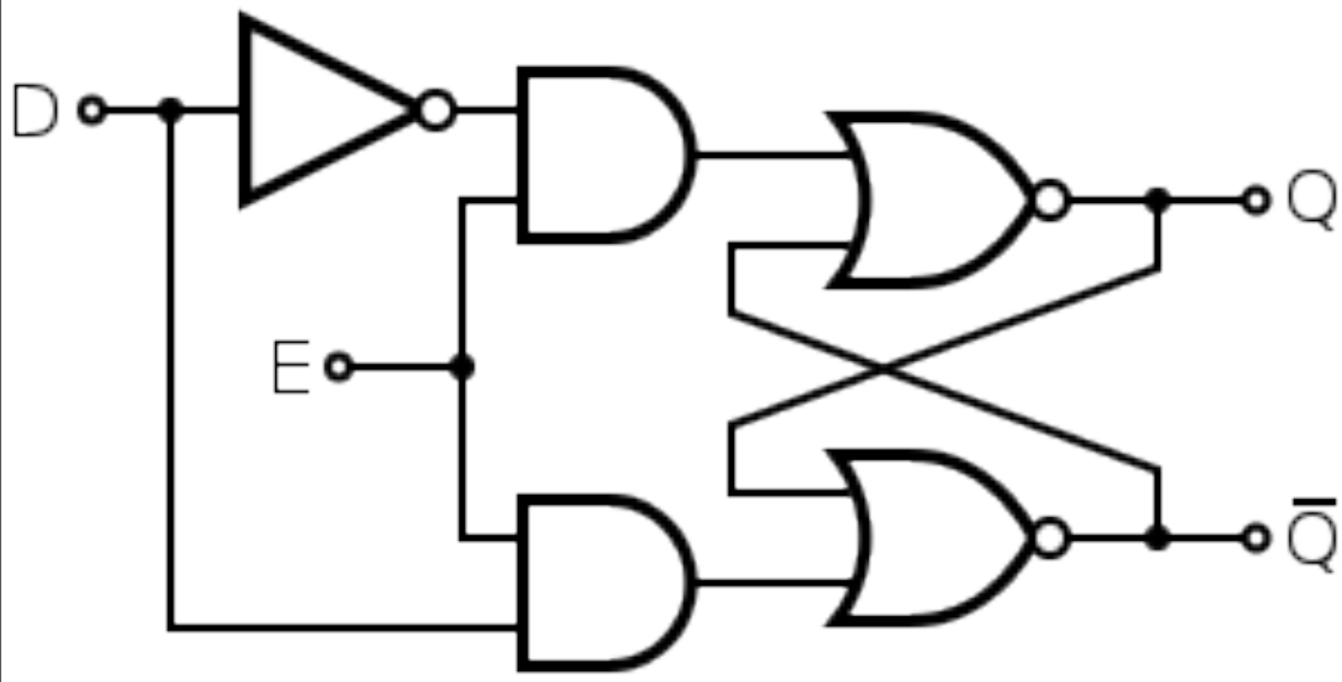
(D Latch)

D	E	Next Q
0	0	Q
0	1	0
1	0	Q
1	1	1

-D is short for data; E is short for enable
-Basically, if E is set, store the data

Building Up

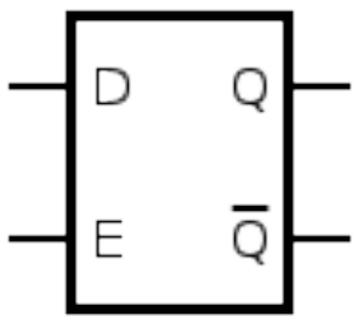
Can use this to store any bit



(D Latch)

D	E	Next Q
0	0	Q
0	1	0
1	0	Q
1	1	1

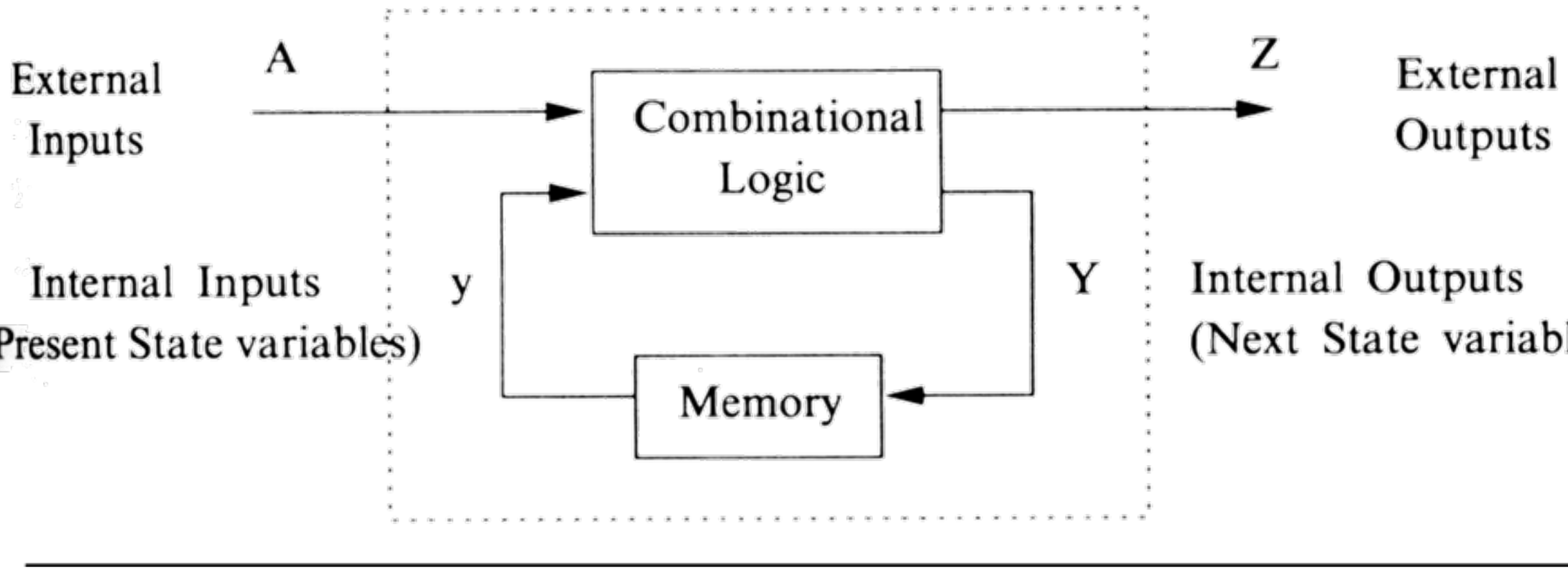
Component shorthand:



-D is short for data; E is short for enable
-Basically, if E is set, store the data

Question

How fast can this go?



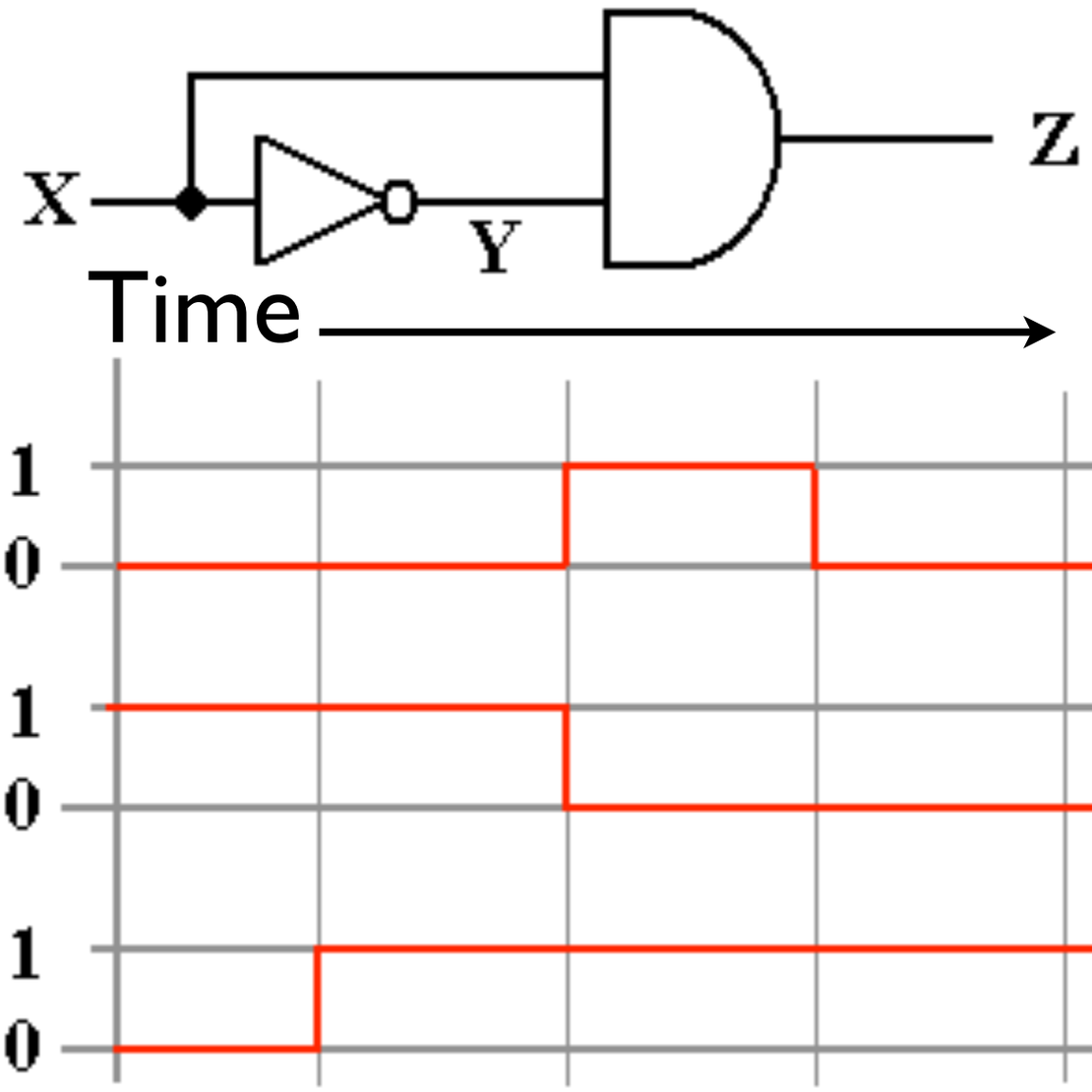
-How quickly I can access memory is one thing, but what does a memory access even mean here? We're talking physical devices - how do I know it's ready?

Gate Delay

```
Z = X!X
```

Gate Delay

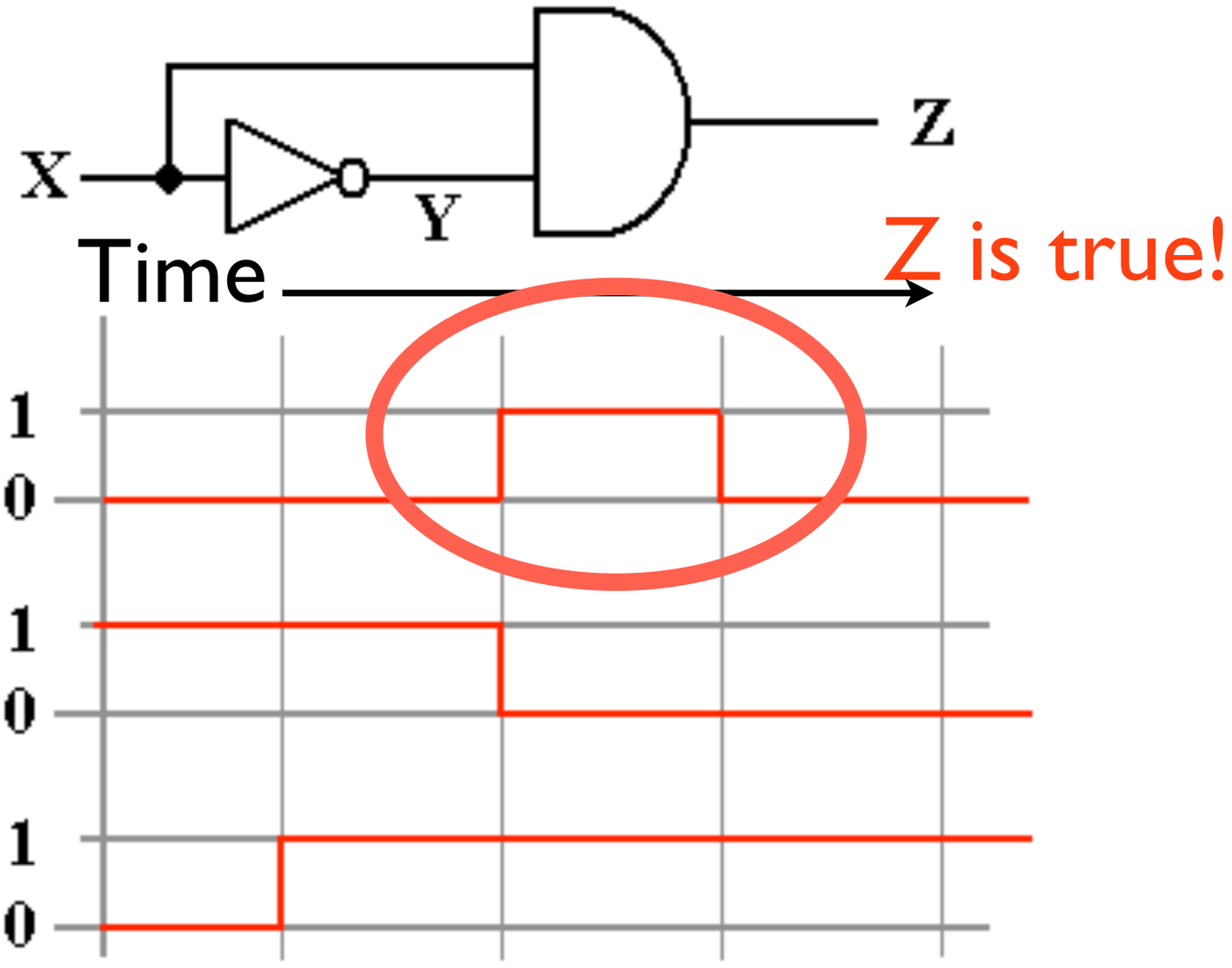
$$Z = X!X$$



-This is a timing diagram: it shows how outputs change over time with inputs
-This one also shows the values as it goes through a sub-portion of the circuit, namely Y
-Key point: while logically this statement should always be false, there is actually a point when Z is true!

Gate Delay

$$Z = X ! X$$

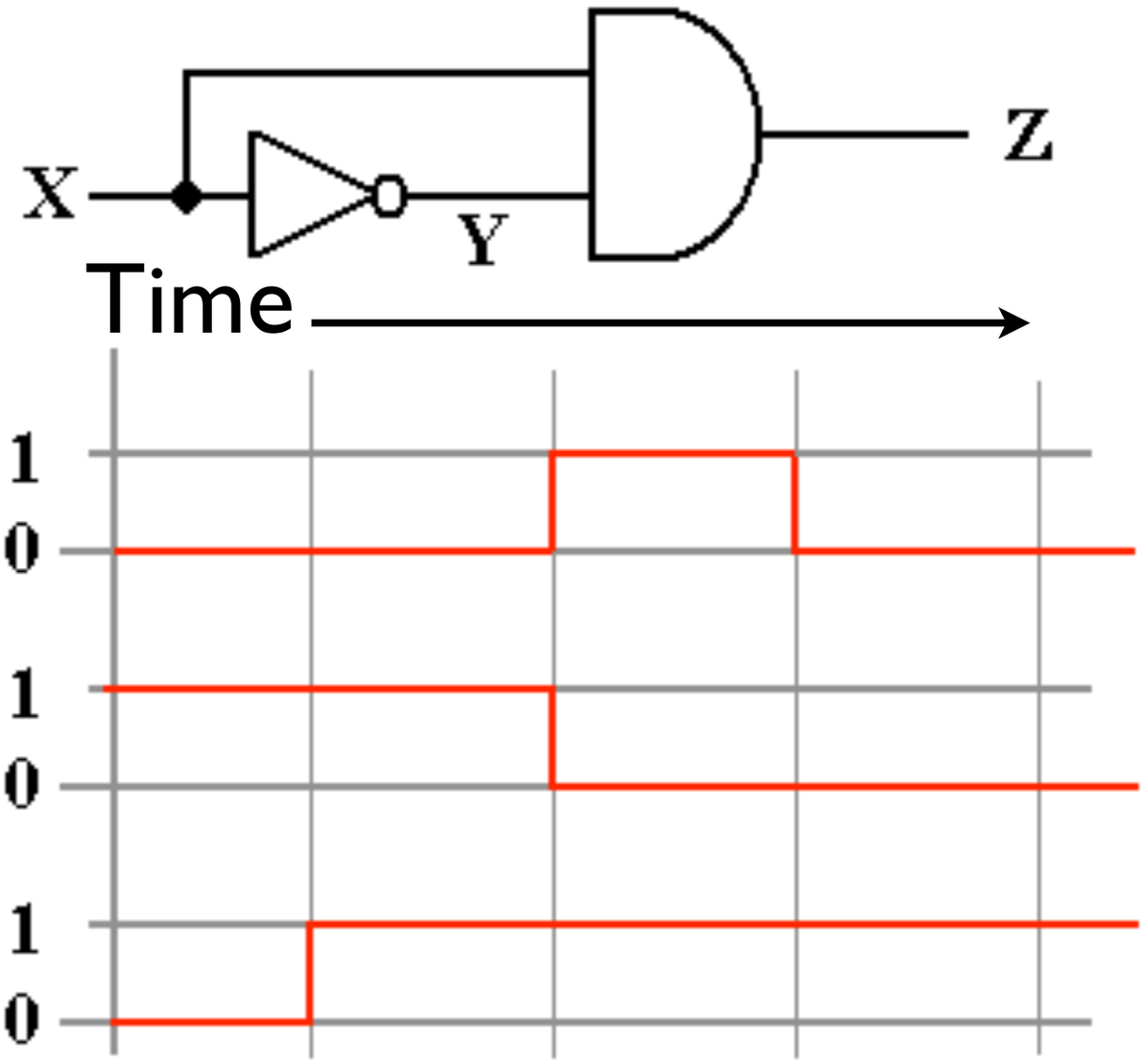


-This is a timing diagram: it shows how outputs change over time with inputs
-This one also shows the values as it goes through a sub-portion of the circuit, namely Y
-Key point: while logically this statement should always be false, there is actually a point when Z is true!

Solution

Only look at the outputs at preset intervals.

Space the intervals so the circuit will always be stable by that point.



-This is a timing diagram: it shows how outputs change over time with inputs

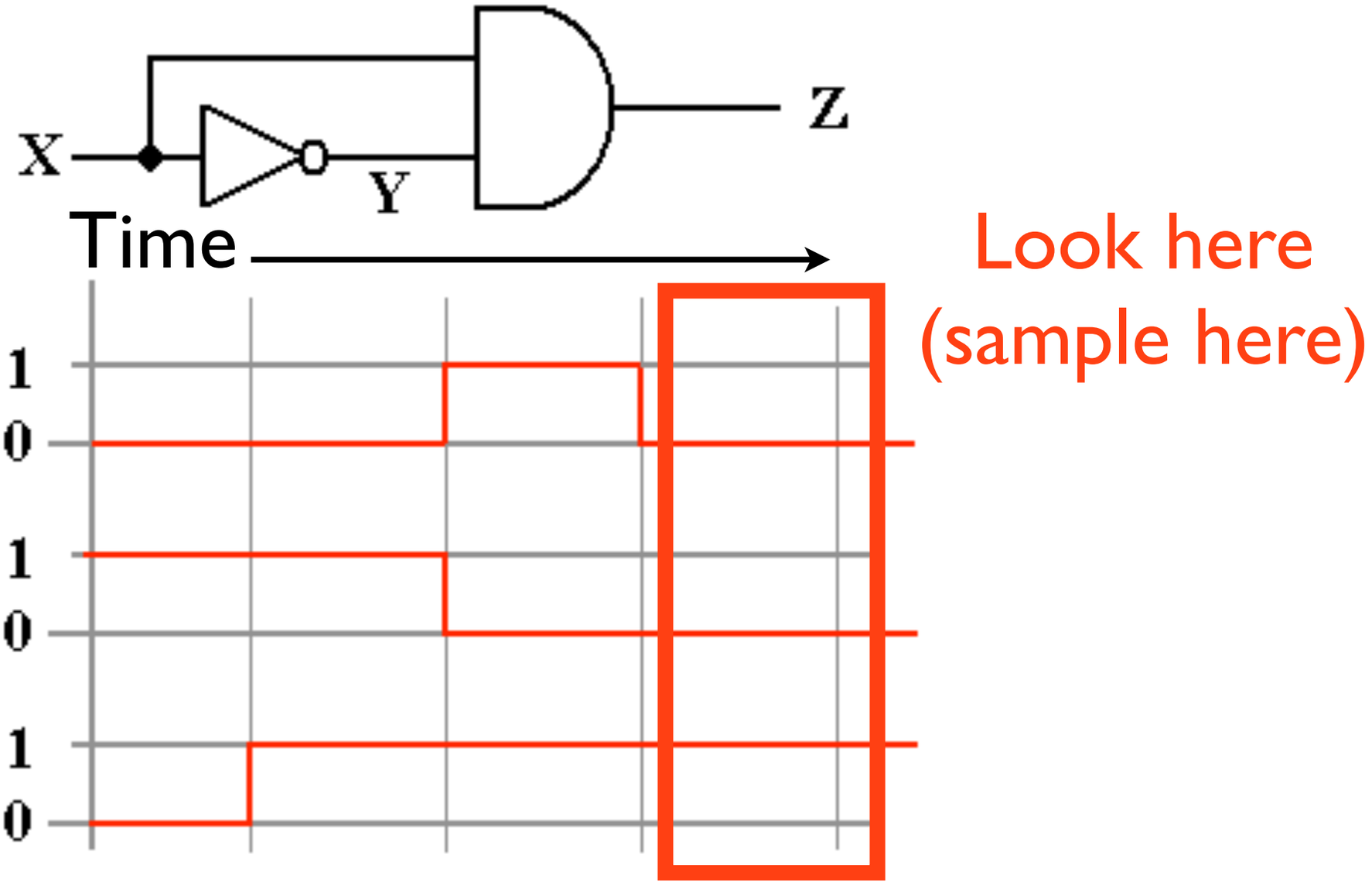
-This one also shows the values as it goes through a sub-portion of the circuit, namely Y

-Key point: while logically this statement should always be false, there is actually a point when Z is true!

Solution

Only look at the outputs at preset intervals.

Space the intervals so the circuit will always be stable by that point.



-This process of looking is more formally called sampling

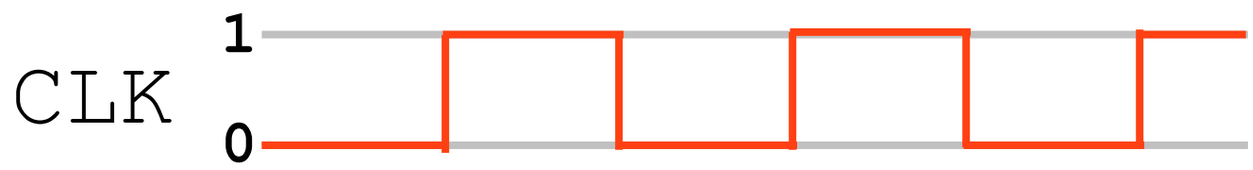
Intervals

Produced by a clock generator: another kind of circuit

-These exist as physical devices

Intervals

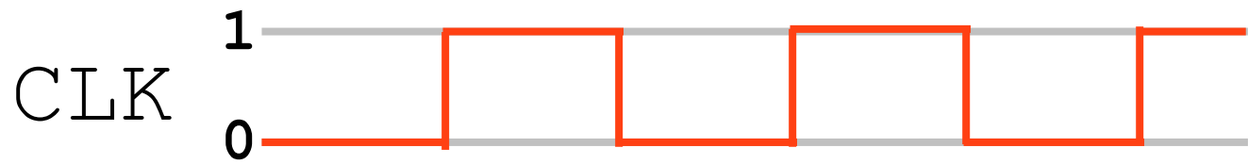
Produced by a clock generator: another kind of circuit

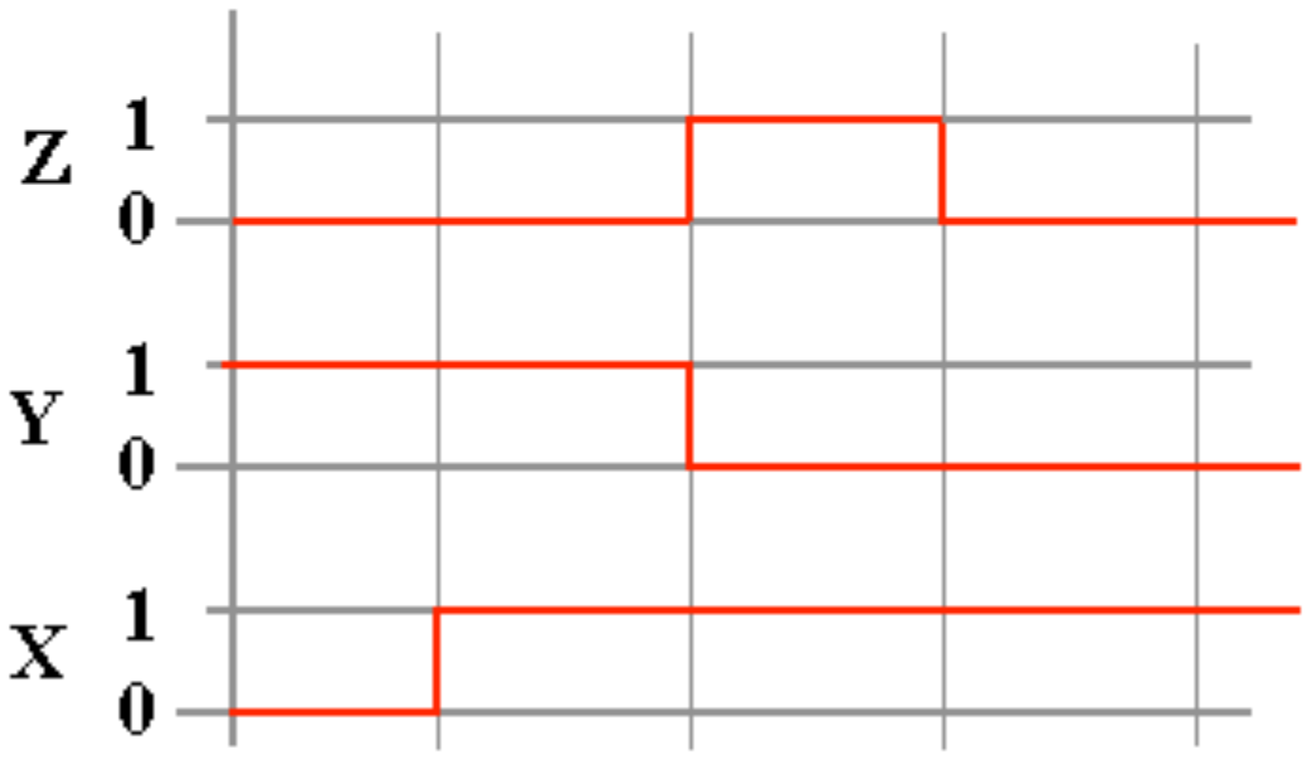


-These exist as physical devices

Intervals

Produced by a clock generator: another kind of circuit

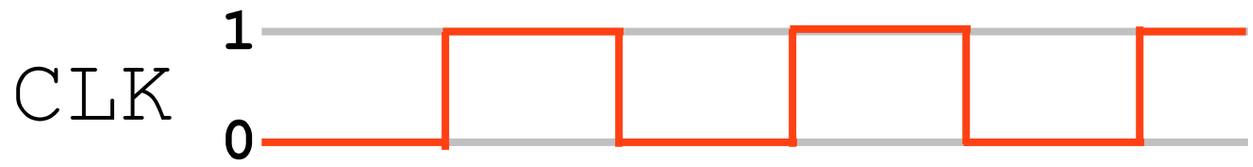


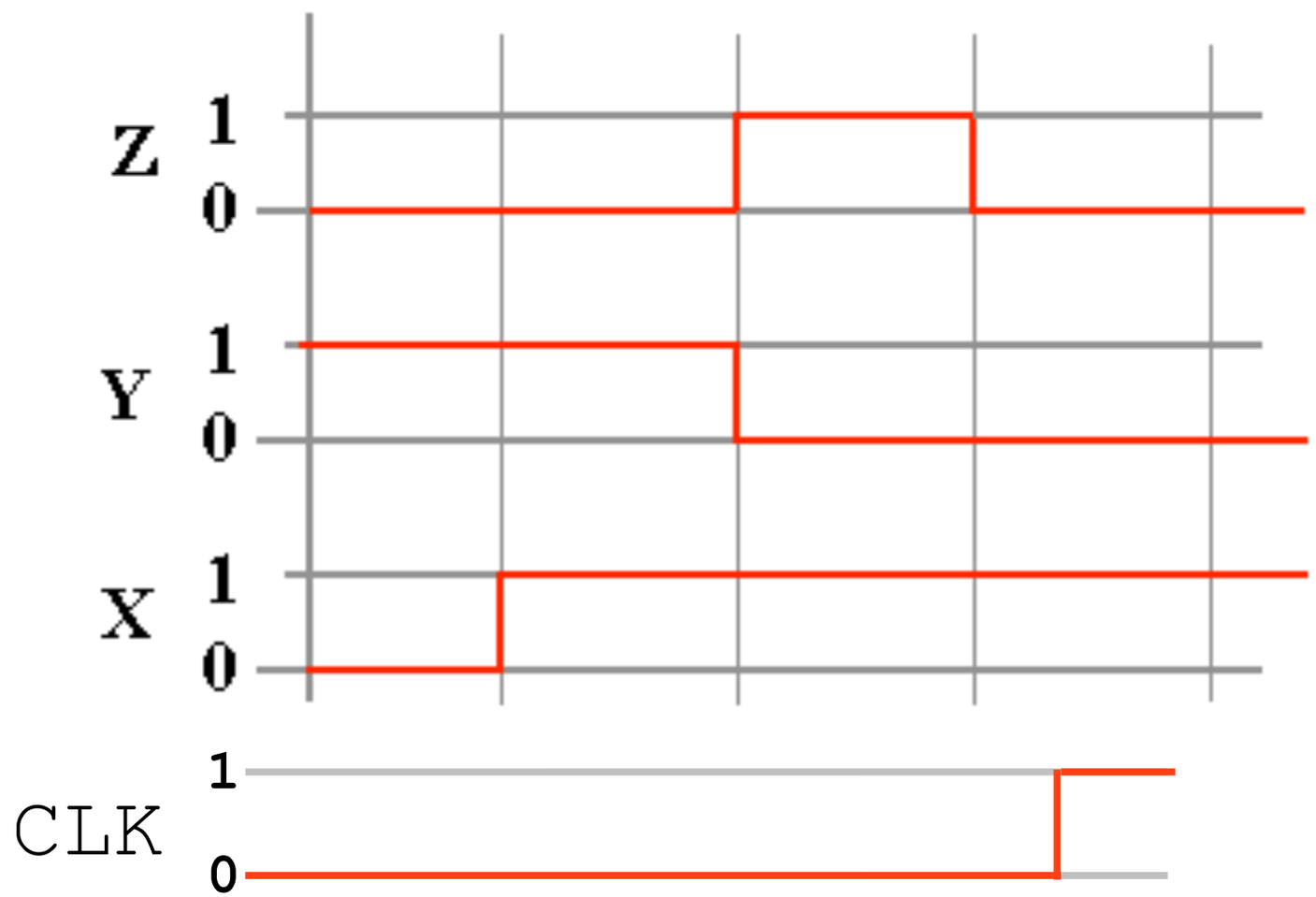


-These exist as physical devices

Intervals

Produced by a clock generator: another kind of circuit

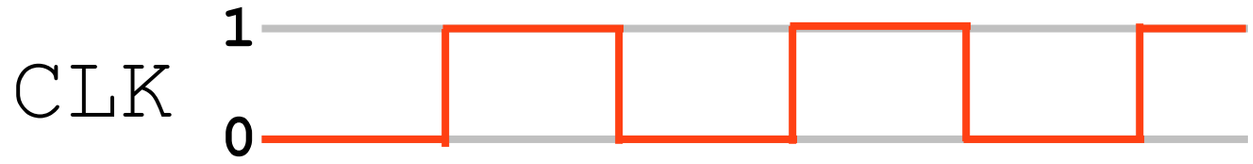


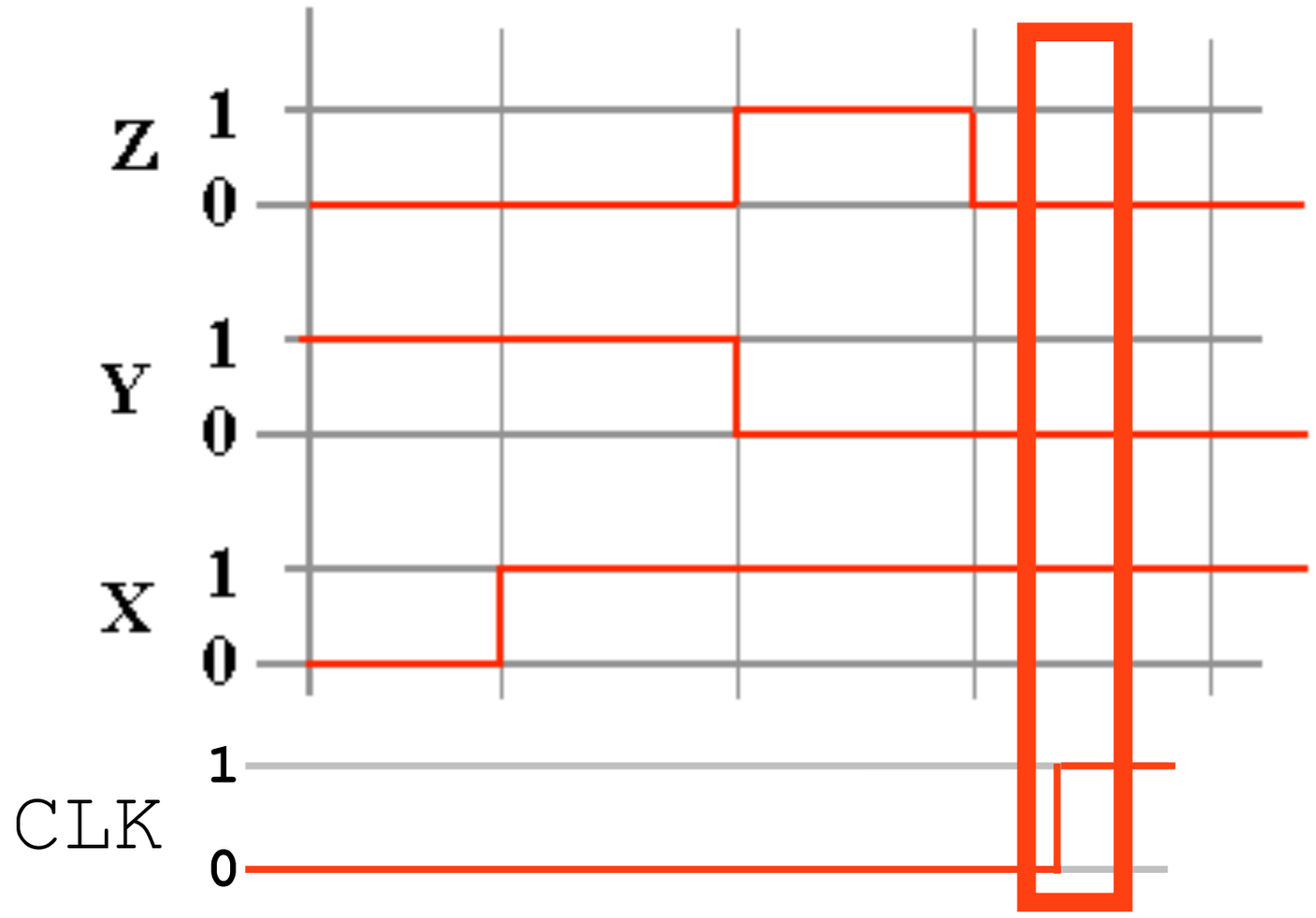


-These exist as physical devices

Intervals

Produced by a clock generator: another kind of circuit



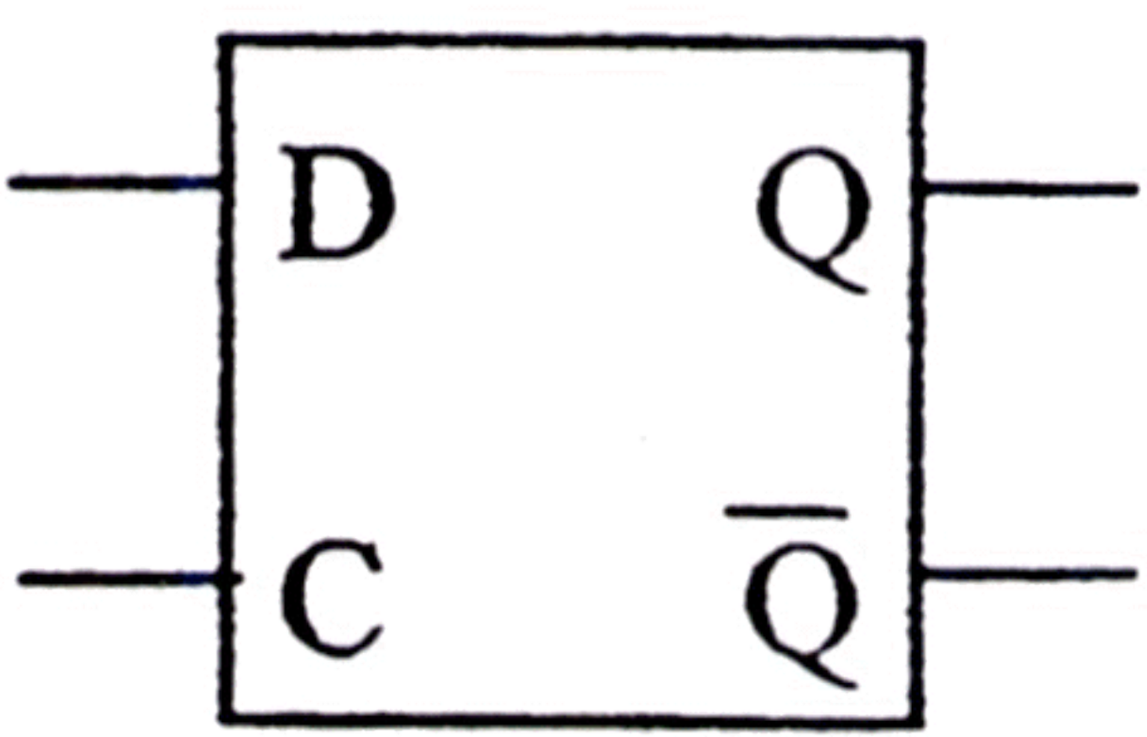


For example: sample on rising edge of clock signal

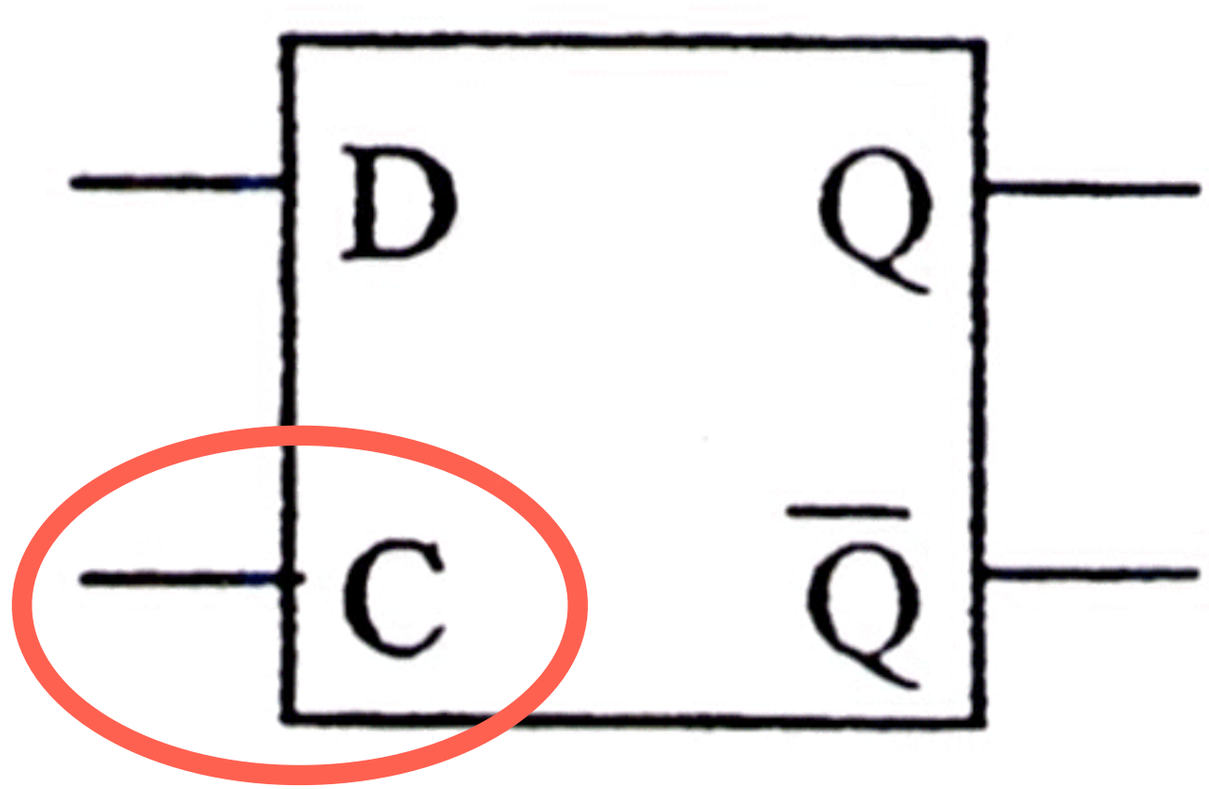
-These exist as physical devices

-With the “for example” part, we can design components that will trigger on the rising edge, the falling edge, the high plateau, or the low plateau. Real systems may use multiple one for different parts of the circuit

D flip-flop



D flip-flop



Clock input.

This could be triggered on the rising edge, depending on the component