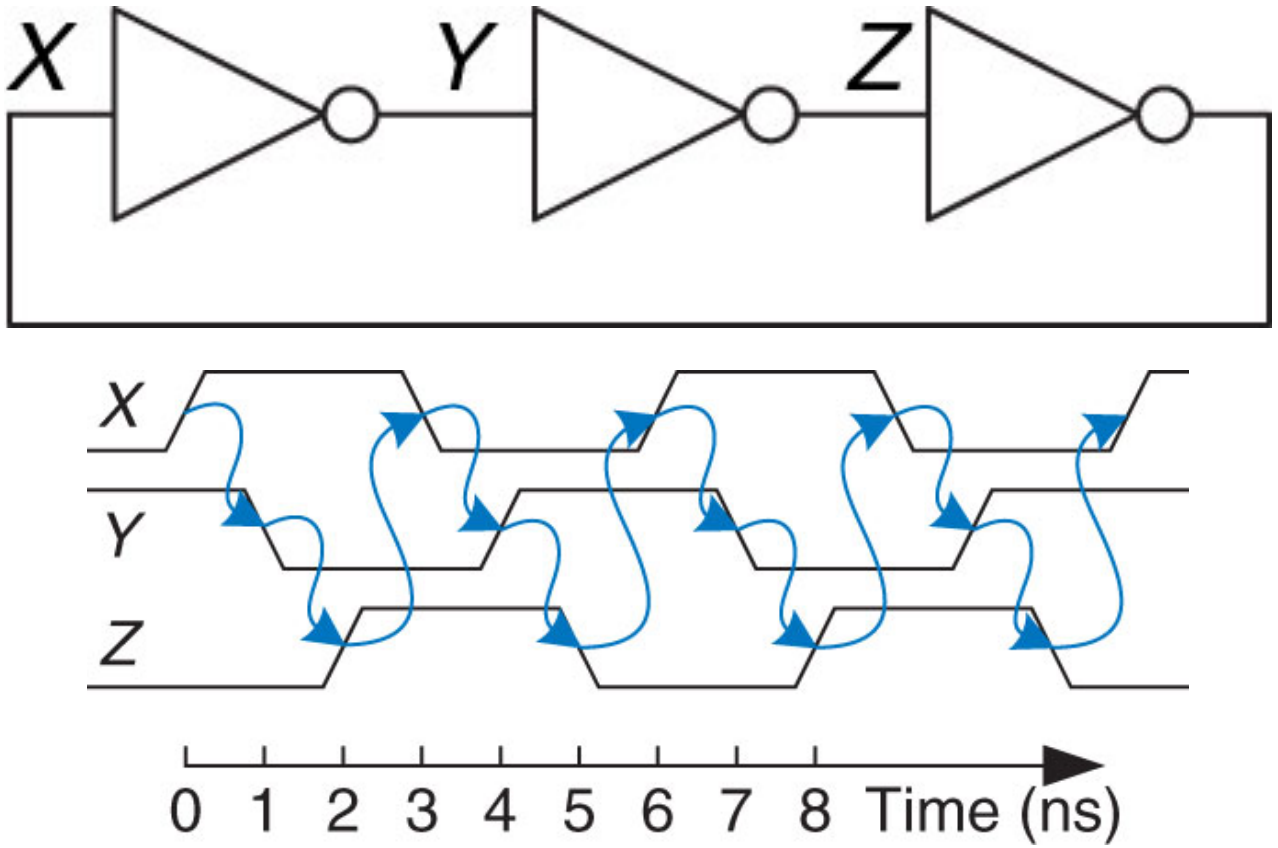
Synchronous Sequential Circuits

Sequential Circuits

- SR latch – holds 1 bit of state. Modified by activating S or R
- D latch – holds 1 bit of state. Activating clock allows D to set value
- D flip-flop – holds 1 bit of state. D sets the state only on the rising edge of the clock
- Register – several D flip-flops that act together. Stores as many bits as there are flip-flops.

- All of the sequential building locks we have looked at are *memory*!

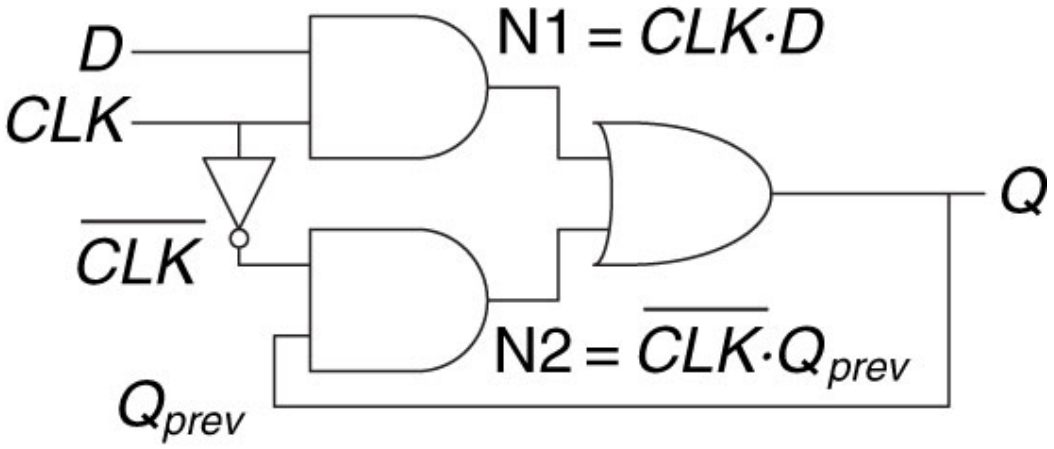
What is this?



How about this?

CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$



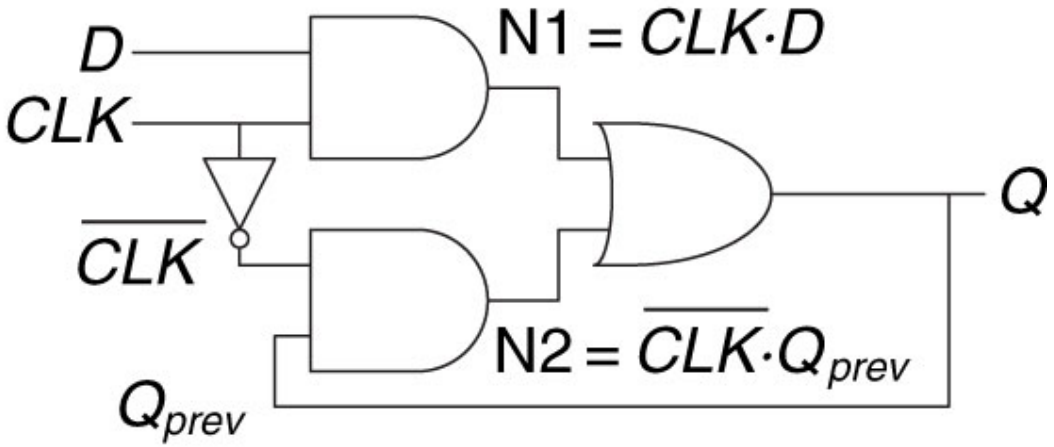
How about this?

Studying characteristic table, we see that previous circuit is alternative implementation of D-latch

This implementation is a bit dangerous

Make the (unusual) assumption that inverter takes twice as long as other gates, and we can find scenario where circuit does not work

CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1


D
CLK
$\overline{CLK}$
N1 = CLK·D
N2 = $\overline{CLK}$·Q_{prev}
Q
Q_{prev}

Problematic Circuitry

We made a weird assumption and the circuit did not work as designed – so what?

We do not want to have to worry about things like the speeds of particular gates when determining how circuit will behave

Subtle errors can arise when timing is an issue. How likely were you to spot that last problem just by studying the circuit?

Problematic Circuitry

Key: Easy to make mistakes when designing sequential circuits.

How can we mitigate these potential problems?

Fixing the Problematic Circuitry

As we saw with combinational circuits, sometimes we can make ourselves more productive by limiting our options

When we restrict a circuit to have no loops (i.e., be combinational), we can use truth tables and Boolean algebra to understand it more easily

We make a similar restriction when designing sequential circuits: in our circuits, loops are allowed, but *only if there is a register on the looping path*

Synchronous Sequential Circuit

Circuit that contains loops, but only loops with registers on the path, is **synchronous sequential circuit**

We will see that designing synchronous sequential circuits is not much harder than designing combinational circuits

Working with synchronous sequential circuits is *much* easier than working with general sequential circuits

Fixing the Problematic Circuitry

Introduce registers into any cyclic path

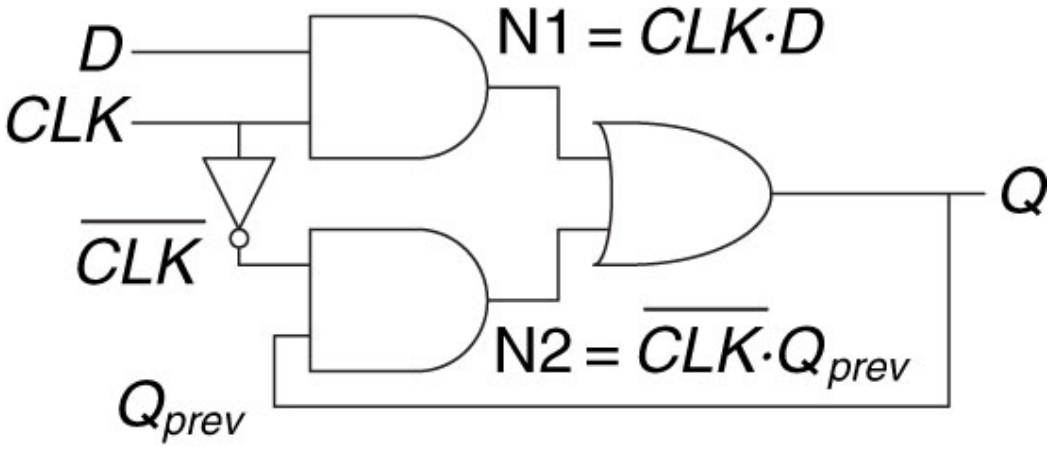
As long as clock period is long enough (more on that later), small variations in timing of individual gates will not cause circuit to misbehave

For now, assume clock period is long enough (i.e., clock “ticks” slowly enough) that inputs to registers settle before clock edge

Correct this circuit

CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$

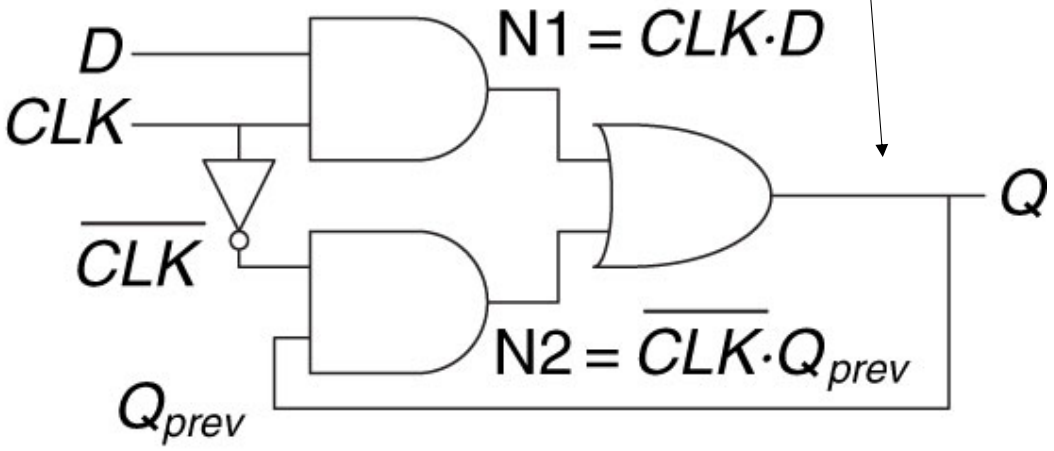


Correct this circuit

CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Add a register here. Wait for combinational logic to finish before updating Q.

$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$



Note: This example would not make sense in practice. Why not?

Rules of Synchronous Sequential Circuits

Every circuit element is either a register or a combinational circuit

At least one circuit element is a register

All registers receive the same clock signal

Every cyclic path contains at least one register.

Synchronous Sequential vs Combinational

With combinational circuits, main rule was “no cycles”

With synchronous sequential circuits, main rule is “no cycles *unless there is a register on the cyclic path*”

Once we add necessary registers, combinational and synchronous sequential circuits are very similar

Synchronous Sequential vs Combinational

Registers store state of circuit

By treating current state as just another input, we can act as though synchronous sequential circuit is combinational

Characteristic tables and Boolean algebra will both be available to us, just like truth tables and Boolean algebra for combinational circuits

Understanding Synchronous Sequential Circuits

Track changes to circuits over clock ticks – track both output and state

Remember: state just means “values currently in registers”

Registers change only on rising edge of clock, so state can change only at rising edge of clock

Understanding Synchronous Sequential Circuits

Circuit is initialized in some state

After delay for combinational logic, circuit produces two things based on inputs *and* current state:

1. Output(s)
2. Next state

Characteristic table has inputs and current state as “inputs” and has outputs and *next* state as “outputs”

Synchronous Sequential Circuits

Blocks of combinational logic and registers

Behavior of the circuit can be distilled into a set of discrete states

Next state depends only on values in registers and current inputs

If you squint, looks like combinational circuit

- Each step we are looking only at inputs and current state
- Current state is like another input