Let's try using the latch as the memory component in our sequential logic system.

To load the encoding of the new state into the latch, we open the latch by setting the latch's gate input HIGH,

letting the new value propagate to the latch's Q output, which represents the current state.

This updated value propagates through the combinational logic, updating the new state information.

Oops, if the gate stays HIGH too long, we've created a loop in our system and our plan to load the latch with new state goes awry as the new state value starts to change rapidly as information propagates around and around the loop.

So to make this work, we need to carefully time the interval when G is HIGH.

It has to be long enough to satisfy the constraints of the dynamic discipline,

but it has to be short enough that the latch closes again before the new state information has a chance to propagate all the way around the loop.

Hmm.

I think Mr. Blue is right: this sort of tricky system timing would likely be error-prone since the exact timing of signals is almost impossible to guarantee.

We have upper and lower bounds on the timing of signal transitions but no guarantees of exact intervals.

To make this work, we want to a load signal that marks an instant in time, not an interval.

Here's an analogy that will help us understand what's happening and what we can do about it.

Imagine a line cars waiting at a toll booth gate.

The sequence of cars represents the sequence of states in our sequential logic and the gated toll both represents the latch.

Initially the gate is closed and the cars are waiting patiently to go through the toll booth.

When the gate opens, the first car proceeds out of the toll both.

But you can see that the timing of when to close the gate is going to be tricky.

It has to be open long enough for the first car to make it through, but not too long lest the other cars also make it through.

This is exactly the issue we faced with using the latch as our memory component in our sequential logic.

So how do we ensure only one car makes it through the open gate?

One solution is to use TWO gates!

Here's the plan: Initially Gate 1 is open allowing exactly one car to enter the toll booth and Gate 2 is closed.

Then at a particular point in time, we close Gate 1 while opening Gate 2.

This lets the car in the toll booth proceed on, but prevents any other car from passing through.

We can repeat this two-step process to deal with each car one-at-time.

The key is that at no time is there a path through both gates.

This is the same arrangement as the escapement mechanism in a mechanical clock.

The escapement ensures that the gear attached to the clock's spring only advances one tooth at a time, preventing the spring from spinning the gear wildly causing a whole day to pass at once!

If we observed the toll booth's output, we would see a car emerge shortly after the instant in time when Gate 2 opens.

The next car would emerge shortly after the next time Gate 2 opens, and so on.

Cars would proceed through the toll booth at a rate set by the interval between Gate 2 openings.

Let's apply this solution to design a memory component for our sequential logic.

Taking our cue from the 2-gate toll both, we'll design a new component, called a D register, using two back-to-back latches.

The load signal for a D register is typically called the register's "clock", but the register's D input and Q output play the same roles as they did for the latch.

First we'll describe the internal structure of the D register, then we'll describe what it does and look in detail at how it does it.

The D input is connected to what we call the master latch and the Q output is connected to the slave latch.

Note that the clock signal is inverted before it's connected to the gate input of the master latch.

So when the master latch is open, the slave is closed, and vice versa.

This achieves the escapement behavior we saw on the previous slide: at no time is there active path from the register's D input to the register's Q output.

The delay introduced by the inverter on the clock signal might give us cause for concern.

When there's a rising 0-to-1 transition on the clock signal, might there be a brief interval when the gate signal is HIGH for both latches since there will be a small delay before the inverter's output transitions from 1 to 0?

Actually the inverter isn't necessary:

Mr Blue is looking at a slightly different latch schematic where the latch is open when G is LOW and closed when G is high.

Just what we need for the master latch!

By the way, you'll sometimes hear a register called a flip-flop because of the bistable nature of the positive feedback loops in the latches.

That's the internal structure of the D register.

In the next section we'll take a step-by-step tour of the register in operation.