# MITOCW | MIT6_004S17_05-02-05_300k

In 6.004, we have a specific plan on how we'll use registers in our designs, which we call the single-clock synchronous discipline.

Looking at the sketch of a circuit on the left, we see that it consists of registers

— the rectangular icons with the edge-triggered symbol —

and combinational logic circuits, shown here as little clouds with inputs and outputs.

Remembering that there is no combinational path between a register's input and output, the overall circuit has no combinational cycles.

In other words, paths from system inputs and register outputs to the inputs of registers never visit the same combinational block twice.

A single periodic clock signal is shared among all the clocked devices.

Using multiple clock signals is possible, but analyzing the timing for signals that cross between clock domains is quite tricky, so life is much simpler when all registers use the same clock.

The details of which data signals change when are largely unimportant.

All that matters is that signals hooked to register inputs are stable and valid for long enough to meet the registers' setup time.

And, of course, stay stable long enough to meet the registers' hold time.

We can guarantee that the dynamic discipline is obeyed by choosing the clock period to be greater then the t_PD of every path from register outputs to register inputs,

plus, of course, the registers' setup time.

A happy consequence of choosing the clock period in this way is that at the moment of the rising clock edge, there are no other noise-inducing logic transitions happening anywhere in the circuit.

Which means there should be no noise problems when we update the stored state of each register.

Our next task is to learn how to analyze the timing of a single-clock synchronous system.

Here's a model of a particular path in our synchronous system.

A large digital system will have many such paths and we have to do the analysis below for each one in order to find the path that will determine the smallest workable clock period.

As you might suspect, there are computed-aided design programs that will do these calculations for us.

There’s an upstream register, whose output is connected to a combinational logic circuit which generates the input signal, labeled STAR, to the downstream register.

Let’s build a carefully-drawn timing diagram showing when each signal in the system changes and when it is stable.

The rising edge of the clock triggers the upstream register, whose output (labeled Q_R1) changes as specified by the contamination and propagation delays of the register.

Q_R1 maintains its old value for at least the contamination delay of REG1, and then reaches its final stable value by the propagation delay of REG1.

At this point Q_R1 will remain stable until the next rising clock edge.

Now let’s figure out the waveforms for the output of the combinational logic circuit, marked with a red star in the diagram.

The contamination delay of the logic determines the earliest time STAR will go invalid measured from when Q_R1 went invalid.

The propagation delay of the logic determines the latest time STAR will be stable measured from when Q_R1 became stable.

Now that we know the timing for STAR, we can determine whether STAR will meet the setup and hold times for the downstream register REG2.

Time t1 measures how long STAR will stay valid after the rising clock edge.

t1 is the sum of REG1’s contamination delay and the logic’s contamination delay.

The HOLD time for REG2 measures how long STAR has to stay valid after the rising clock edge in order to ensure correct operation.

So t1 has to be greater than or equal to the HOLD time for REG2.

Time t2 is the sum of the propagation delays for REG1 and the logic, plus the SETUP time for REG2.

This tells us the earliest time at which the next rising clock edge can happen and still ensure that the SETUP time for REG2 is met.

So t2 has to be less than or equal to the time between rising clock edges, called the clock period or tCLK.

If the next rising clock happens before t2, we'll be violating the dynamic discipline for REG2.

So we have two inequalities that must be satisfied for every register-to-register path in our digital system.

If either inequality is violated, we won't be obeying the dynamic discipline for REG2 and our circuit will not be guaranteed to work correctly.

Looking at the inequality involving tCLK, we see that the propagation delay of the upstream register and setup time for the downstream register take away from the time available useful work performed by the combinational logic.

Not surprisingly, designers try to use registers that minimize these two times.

What happens if there's no combinational logic between the upstream and downstream registers?

This happens when designing shift registers, digital delay lines, etc.

Well, then the first inequality tells us that the contamination delay of the upstream register had better be greater than or equal to the hold time of the downstream register.

In practice, contamination delays are smaller than hold times, so in general this wouldn't be the case.

So designers are often required to insert dummy logic, e.g., two inverters in series, in order to create the necessary contamination delay.

Finally we have to worry about the phenomenon called clock skew, where the clock signal arrives at one register before it arrives at the other.

We won't go into the analysis here, but the net effect is to increase the apparent setup and hold times of the downstream register, assuming we can't predict the sign of the skew.

The clock period, tCLK, characterizes the performance of our system.

You may have noticed that Intel is willing to sell you processor chips that run at different clock frequencies, e.g., a 1.7 GHz processor vs. a 2 GHz processor.

Did you ever wonder how those chips are different?

As is turns out they’re not!

What’s going on is that variations in the manufacturing process mean that some chips have better tPDs than others.

On fast chips, a smaller tPD for the logic means that they can have a smaller tCLK and hence a higher clock frequency.

So Intel manufactures many copies of the same chip, measures their tPDs and selects the fast ones to sell as higher-performance parts.

That’s what it takes to make money in the chip biz!