**Layout design flow**

Let's continue with the more elaborated stages of layout design.

So the first step is to implement the function itself using, for example, if you have been given this particular function, you need to implement this function based on the set of PMOS and NMOS transistor connections.

So this particular set of transistors will represent this function.

The next step is to derive the PMOS network graph and the NMOS network graph out of this.

So if you look into a typical art of layout, it's an Euler's path plus state diagram.

So if you follow the steps, the layout that you want to derive out of this particular function will be giving you the best performance in the best area.

So the next step is to, after you get your PMOS network graph and the NMOS network graph, the next step is to obtain the Euler's path.

So in order to obtain the Euler's path, you might want to look into one of my courses on custom layout that will help you to get the Euler's path.

Euler's path is basically a path which is being traced only once.

So that's basically called as an Euler's path.

For details, you might want to go into my custom layout course and have a look into that.

So once you get the Euler's path for the PMOS and the NMOS network graph, so in this So in this case the Euler's path is A, C, E, F, D, and D. So based on this Euler's path, the next step is to draw a stick diagram out of it.

So if you look at the right hand side, this is a stick diagram.

A, C, E, F, D, and B are nothing but the inputs.

And these are nothing but the gate inputs.

So you place the polysilicon A, C, E, F, D, B in that particular order.

And you also make the connections which is being mentioned by your circuit.

So this circuit which was present over here, based on the circuit connections of the circuit topography, you make the connections over here.

In this case, for example, if you look into this one, if you look into the E input of your transistor, of this transistor, this represents your NMOS, this represents your PMOS.

And it says your, this is source, this is drain, it says your E input source and C input drain are all connected.

Or basically E transistor and the C transistor are all, the source and drain are connected.

And your F transistor source and your D transistor drain are connected.

Basically the PMOS transistor source and drain are connected.

And this particular block means your A transistor source and your C transistor source are connected.

That's why you see your contact over here.

And this in turn is connected to your D input, or the D transistor, D NMOS transistor drain and the B NMOS transistor drain.

Okay, so this is how you interpret the stick diagram.

Or basically this stick diagram is being derived out of the circuit diagram or the circuit design that you did just in the previous stage.

Okay, so the next step is to convert this stick diagram into a typical layout, into a proper layout, adhering to the rules that we have got from the input.

So remember sometime back we were looking into the layout rules in the PDKs.

When we received this PDKs, we also got a DRC rules.

So the next step is to convert this stick diagram into a layout while we adhere to the DRC rules which has been given by the founder.

So let's, and also adhering to the rules and the user defined specifications which has been given by the top level user.

For example, the drawn gate length is over here, the contact is over here.

It might be possible that the contact places, the contact placements or the pin locations has been given over here, over here and over here.

And that's the reason we have placed the pins somewhere over here.

So those are some of the rules or some of the user defined specifications.

Or those are the inputs that we got from our previous stages.

And again the metal layers, so for example it has got metal 1, it has got polysilicon and so on.

So this thing has been built on metal 1 layer and it could be again, it could be taken up to the metal 2 or metal 3 layer depending upon the requirement that comes from the user or from the user defined specs.

So that's a layout which has been drawn using the circuit that we have over here and using the inputs of the DRC, DRC rule inputs that we got from the input stage or from the founders.

So that's the next stage.

And finally if you see, you take this particular hand drawn layout and put it into a tool.

In this case I've used a tool called as Magic which is an open source tool.

So I've drawn the layout using Magic and this is the typical layout that you see out of this complete stack.

So this is the layout design part and these are the different steps that you need to go while designing your layout.

And this step has been very much elaborated in my custom layout course.

You might want to have a look into that course.

So finally once you have this particular layout, now you have the cell width, you have the cell height, you have the cell width over here.

And in this particular layout we'll add that to the drain current requirements and all the pin locations and the all user defined specifications that we were asked to do so.

Okay, so this is your library cell.

The next step and the final step that is remaining in this case is to extract the parasitics out of this particular layout and characterize it in terms of timing.

So before that the output of the layout design will be GDS2.

GDS2 is nothing but a typical layout file.

Left basically defines the width and the height of the cell and the extracted spike netlist.

So when we say about extracted spike netlist, it's nothing but the parasitics or the resistance and the capacitance of each and every element over here has been extracted out of this particular layout.

Okay, so that's an extracted spike netlist and it's a very useful information to do a characterization.

So next step after you get the extracted spike netlist and after you get the typical layout over here, the next step is to characterize this thing and that's what this course is all about.

So characterization is the step that will help you to get your timing, noise and power information.

And the output of the characterization is nothing but timing, noise and power dot loops and the functionality of this particular circuit.

So what we'll do is we'll stop at this point of time.

From the next video we'll look into characterization flow.

So a typical flow, a typical cell design flow for an inverter has to go through all these complete steps.

So inverter by definition, it looks very simple.

It's a single input gate, but the steps involved in designing this inverter is humongous.

Okay, so the next step is to look into the characterization flow.

**Typical characterization flow**

And everyone, let's continue with the characterization flows.

First of all, let me figure out what are the things we have as of now, based on the three different steps that we saw in the past video.

So in the past video, we saw three steps like the input stage or the cell design, so there was an input stage, there was a design steps, and there was output.

So from the inputs and the design steps, let us figure out what are the things we have.

So let's say we have done the circuit design and we have done the layout design.

So out of that, the inputs that is available with us are this one.

You have the layout with you.

Okay, so this is a layout of, let's say, a buffer.

This is an inverter.

This is an inverter.

And I have used open source magic layout tool to draw this particular buffer.

So you have a buffer over here.

So this is a layout of a buffer.

You have the description of the buffer.

So for example, two inverters connected back to back, you have the description of the entire buffer, along with the power sources and everything that is getting connected.

Okay, so you have this description as well.

You have the SPICE extracted netlist.

So whatever you see in this particular layout, you have this metal layers, you have this contacts and everything.

So each and every element will have associated resistance and capacitances with them.

So we have extracted all of them in terms of a SPICE netlist.

So in the SPICE netlist, you will see a lot of like capacitances and resistances over here.

It's not evident over here, but in a real SPICE netlist, which is extracted, or in an extracted SPICE netlist out of the layout, you will see a lot of resistances and capacitances statements over here.

So that's called an extracted SPICE netlist.

Finally, you have the inverter subcircuit.

So if you see the lines over here, this is called X1, IN1 net, and there's one net and there's another net.

And the subcircuit name is myINV.

Again, there's something called X2, and the subcircuit name is myINV.

So this subcircuit has to be loaded from somewhere.

So I've included the subcircuit file over here, and this is how the subcircuit looks like.

So the subcircuit actually contains the NMOS and the PMOS models, and has got the connectivity of an inverter.

So this inverter has been loaded twice.

So what you see in this particular diagram on the right side, if you have this inverter, this inverter is being called from the subcircuit.

This inverter is being called from the subcircuit.

And when you attach them both together in the form of a SPICE netlist, you get the description of a buffer.

Along with the description of the buffer, you have these capacitances and resistances, which will give you a description of an extracted SPICE netlist of a buffer.

So you have included, you have the subcircuits as well.

Now in the subcircuits, you have the SPICE models, or you have the NMOS SPICE models.

And as I mentioned in my past videos, the NMOS SPICE models, or the SPICE models are nothing but the characteristics of the NMOS, of the NMOS transistors.

So you have the N-channel and N-channel behavior, or the N-channel doping concentration over here.

So basically, these all things I have explained a lot in my circuit design and SPICE simulation course.

So you might want to have a look into those courses.

Most of them, we have explained in that course.

You have the level, the kind of simulation that you need to do.

And you have a lot of things.

If you go through the course, you will be able to figure out all of them, mostly all of them.

Okay.

So you have two different SPICE models.

One is for the NMOS, one is for the PMOS.

So these are the inputs that are available with us.

So let us try to build the characterization flow based on the inputs that we have.

And that's a standard characterization flow that has been followed in industries.

So let's do that.

So the first step is to read in the models.

First of all, reading the models, because this is the place where things start from.

So you need to read in the models and also the tech file that is needed for the layout, but that comes at a later stage.

So the first step that comes out of the industries or out of the foundry is the model file.

So the first step is to read the model files.

The second step is to read the extracted SPICE netlist.

The third step is to define or to recognize the behavior of the buffer.

Okay.

That's a really important step.

The fourth step is to read, the fourth step is to read the SPICE, the sub-circuits of the inverter.

So this is a sub-circuit of the inverter that also you need to read in.

And then you have the fifth step where you attach the necessary power sources.

So for example, over here, the VDD from this point and this point are being connected and it's been connected to a VDD supply.

And that is, this has been connected to the ground.

So you have to read in the necessary power supplies.

That is our fifth step.

Your sixth step is to apply the stimulus.

Now stimulus, we have varieties of stimulus and we'll be talking a lot about the inputs of the stimulus that has been given to the characterization setup.

So this is called as a characterization setup.

So there is a lot that we're going to talk about the stimulus and that will follow this particular video.

So this is a stimulus application, which has been followed as a step six.

Next you have to provide the necessary output capacitances.

So over here, I've given an output load capacitance as 10 femtofarad.

And this has to be varied.

If you look into a nonlinear delay models or an LLDM, the output capacitances is varied in a range.

So that's what we have to do.

So let's say for a simple setup, we are going to look into a fixed capacitance and figure out what is the characterization behavior looks like.

And then finally, we have the eighth step, which is you need to provide the necessary simulation command.

So over here, we are doing a transient simulation.

So we give dot plan.

If you would do a DC simulation, we have to give dot DC and so on.

So we have talked a lot about this thing also in the Selected Design and Spice Simulation course.

You might want to have a look into that.

So there are the eight different unique steps that we have for the characterization flow.

Next step is to feed in all these inputs from one to eight in a form of a configuration file to the characterization software called as GUNA.

**Timing Threshold Definitions**

So, let's begin with the timing characterization and this is the step, this is the video onwards where we are trying to understand the various syntax and semantics of a timing.lib, a power.lib and a noise.lib.

So, we are going to understand these concepts behind the syntax and also the syntax and what does the syntax itself represents.

And these syntax are really very important to understand the Guna software, the characterization software because the software works on variables and these are the variables that are being available with us in order to feed into the software.

So, let's try to understand those variables as of now.

So, to begin with, we are going to look into the SPICE or the characterization setup that we had explained in the last video.

You have these inverters connected back to back.

You have the power sources.

You have the stimulus that is being applied over here.

You plot waveforms over here.

So, this is the definition setup that we had explained in the last video.

You have this inverter connected back to back, you have the power sources, you have the stimulus that is being applied over here, you plot waveforms over here, you plot waveforms over here.

So, this brings us to a very important point of understanding the different threshold points of a waveform itself and that's what we call it as timing threshold definitions.

Okay, so these are the points or these are the inputs or these are the variables that are related to any waveform that you see that we will be seeing henceforth.

So, let's begin with the definition.

So, first of all, we have the slew low rise threshold.

So, what I have done is I have taken the waveforms from the output of the first inverter which has been given as an input to the second inverter.

So, this is your input and this is the waveform from the output of the buffer.

So, for example, I had shown two inverters back to back.

So, this is the output or this is the input to the second.

So, that's why it's called inverter out.

So, this is the this is the output of the first inverter which goes as an input to the second inverter and we have written buffer out which says this is the output of the buffer.

Okay, so we'll use these waveforms to basically define or to understand the different syntax that are associated with these variables that you see on the screen.

So, the first of all is slew low rise threshold.

So, whenever we say low, whatever we say low over here, those depicts the values that are close to your zero power supply.

So, for example, when we say slew low rise threshold, it basically says if you want to calculate the slope of this particular waveform or the slew of this particular waveform, you need to have two different points.

Those points could be this point and this point or those points could be this point and this point or those could be this one and somewhere over here.

But you need to define this value.

So, whenever we say slew low, it basically defines the point towards the lowest of your power supply which is zero volts and typically value, the typical value of slew low rise threshold is about 20 percent.

It could be 30 percent as well.

We'll get back to that in a moment from now.

Okay, so let's begin with, let's assume that the slew low rise threshold is 20 percent and we put a value over here.

But that's not enough to calculate the slew.

We need the slew high rise threshold as well.

So, slew high rise threshold comes somewhere over here.

Again, we have for the sake of simplicity, we have kept it 20 percent.

So, whenever you want to calculate the slew of this waveform, you just take the point from 20 percent, you just take the point at, you just take the point 20 percent from the bottom power supply, you take a 20 percent point from the top power supply and just identify the time difference between this and that's where you calculate the slew.

I'll get back to calculation of the slew just after this particular video.

Okay, so that's basically slew low threshold and slew high threshold and this is all for the rising waveform.

Similar definitions also apply for the falling waveform as well.

So, in this case, the slew low fall threshold lies somewhere over here and the slew high fall threshold lies somewhere over here.

So, what are the applications of identifying these thresholds and these values?

So, I'll get back to that when we go to the propulsion delay concept.

So, let's try to understand the definitions first of all.

Okay, so you need these values to calculate the slews of the waveform.

That was one thing.

And next, let us try to look into the other definitions.

For example, you have the in-rise, in-fall, out-rise and out-fall.

So, these are all related to your input waveforms.

Let's try to look into those as well.

So, for example, this is the input stimulus that we have applied.

So, remember in the last video, we were doing a transient analysis.

So, those transient analysis we have given, we have specified some input waveforms.

So, let's say this was the input waveform and this was the output of the first of the buffer.

So, we have taken the complete buffer as basically two inverters connected back-to-back.

We have taken the complete waveform and we have given this stimulus as the input to the waveform, as the input to the circuit and we have taken the output from the buffer.

And this is how it looks like.

So, this is your input waveform, which is an ideal waveform, because this is the thing that we are applying and this is the output of the buffer.

So, now let us look into the other definitions.

In-rise threshold.

So, similarly, what you had for the thresholds for the slew, you also have thresholds for your delays as well.

So, for example, if you want to calculate the delay of this particular cell, of that particular inverter, you need some points to calculate the delay.

Those points could be this point, this point or this and this or this and this.

This becomes the best point that has been defined in order to calculate the delay.

And the points which have been taken across this waveform or across this waveform, they are also possible.

But in this case, we take the 50 percent point of this, of this particular slew waveform.

So, 50 percent lies somewhere over here.

That is what you call as the in-rise threshold.

So, you need this point and one point on the output waveform in order to calculate the delay.

So, this is the in-rise threshold, which is the typical value of the in-rise threshold is 50 percent and it says input.

So, this is the input waveform.

This is the rise.

So, this is the rising waveform.

And similarly, you have the out-rise threshold also as 50 percent.

Out-rise threshold is basically the rising, the output waveform.

Whichever is the output waveform, the rising point or the output rising waveform, the point at which you can even calculate the delay, that point is basically out-rise threshold.

So, now if you want to calculate the delay between the cell, you just need to take this time period.

You need to take this time period and that's it.

You will get your delay.

So, the software uses this thing to calculate the delay.

So, that was about the in-rise.

And similarly, you will have something for the fall as well.

So, if you apply a fall waveform, let us say this thing, if you give a fall waveform to your input, you get the out-buffer, output will be a fall waveform.

And the definitions remains the same.

The wordings remains the same.

Only the things are now getting applied for a fall waveform.

So, out, in-fall threshold will be something over here, which is 50 percent.

The typical value is 50 percent.

And out-fall threshold will also be 50 percent.

If you want to find out the fall delay.

So, a buffer has got two kinds of delay.

Any one input gate, a single input gate has got two kinds of delay.

One is the rise delay and one is the fall delay.

So, this is in order to calculate the fall delay, you just need to take the 50 percent value from this point.

You need to take the 50 percent value from this point.

Take the time points from this one and take a difference of it.

So, I will talk more about the propagation delay when we reach there.

So, these were all the definitions of the timing threshold definitions.

Based on this thing, based on these eight values, we are going to calculate the further values.

For example, we are going to use these terminologies to calculate the sluice, to calculate the propagation delay, to calculate the current, a lot of things.

So, we are going to use all of them as we move ahead.

So, let us try to stop at this point.

**Propagation Delay**

Hello everyone, let's continue with the next timing characterization topic which is propagation delay.

So propagation delay sounds to be very easy, but if not taken care, it might lead to some unexpected results.

So let me show you what I was just talking about.

So let's say you have the same characterization set up as we have shown in the past video.

You have the inputs, you have the input waveform, you have the output waveform.

And we also have defined the characteristics of the waveform, of the threshold points of the waveform.

With that, on that basis, let's say you have this in-rise threshold, in-fall, out-rise and out-fall.

Then if you want to calculate the delay of anything, you just need to subtract out minus in.

That's the standard definition.

So for example, let me write down the definition over here.

You just need to find out the time at the out threshold, you just need to find out the time at the in threshold and subtract out minus in.

And why out minus in?

Because out comes later and in comes early.

So out minus in should give you the exact delay of let's say this particular inverter or this complete buffer.

Now let's try to put some valid values over here.

So let's say we'll take a typical value of 50% which is being followed across the industries.

It might vary across 50%, but let's say we'll take a typical value of 50% as of now.

So with this 50% in mind, and with this thing definition in mind, this is a very important definition because this is the one that leads to unexpected results and I'll show you how.

So this is the definition and this 50%, let's take a waveform, let's take a sample waveform and see how does, how can you apply all this in that particular waveform.

So let's say you have this as an input waveform, if you have this as an output waveform, this is the same waveform that we have taken from the output of the second inverter, okay.

So if you choose the threshold point as 50%, so let's say this point of 50% lies somewhere over here, and the time point at the 50% is 3.207 nanosecond, and the time point at the out fall threshold is about 3.230 nanosecond.

If you want to calculate the delay, it's pretty simple, it's just two three point out minus in.

So out minus in is 3.23 minus 3.07.

So 3.207, that comes to about 23 picosecond.

All well, all good.

So but the next problem is, if there is a movement, if there's a movement of the thresholds over here, it creates a problem over here.

So for example, in this case, the in rise threshold was kept at 50%, out fall threshold was kept at 50%.

But by any chance, if the threshold points moves somewhere to the top over here, the threshold points have changed.

Your time in rise, the in rise will be 3.263 nanosecond, which is, which is this point if you see on the screen, which lies somewhere over here, your time out fall threshold lies over here, which is 3.221 nanosecond.

Typically, what you see is your output comes before the input.

And that's why you see a negative delay of negative 42 picosecond.

And that's not correct, because negative delay are not expected, you need a positive delay.

So the reasons behind having this kind of negative delay is poor choice of your threshold point.

So this is why the choice of threshold points is really, really very important.

And this is a very common problem that we see when we try to do not live a very busy negative negative delay.

So these are really very important thing to look about in the characterization.

So this is also choice of threshold is really very important.

Again, there is another example where we choose the correct threshold, but still you can get a negative delay.

And that could be if you have a waveform something like this.

So if you see this particular waveform is your input waveform, it's from the slew of the input waveform, you can figure out that the that the wire delays are pretty high.

Okay, it could be this could be one of the reasons where your layout, the two buffers, the two inverters are placed very apart.

If you remember, we have seen a layout over here.

So the two inverters are very far apart, and you have a huge wire between two inverters, and that that might lead to the output of the first inverter to give such huge slew.

Okay, so even with this particular kind of waveform, even with the proper choice of thresholds, let's say you have that you have this particular, we'll try to zoom into this area and see what are the values at the in threshold and the out threshold.

Let's try to zoom into this area.

Okay, so if you look if you look into deeply look into that area, your in rise threshold comes to about which is 50%, 50% is nothing but nothing but 0.9, 1.8, if you take a 50% of 1.8 volts, it's 0.9.

So if you if you look into the in rise threshold, which comes somewhere over here, it will it will fall about 4.215 nanosecond, your out fall threshold will fall about 4.207 nanosecond, which is over here.

Again, your output comes before input, that will give you a negative delay of 80% which is not expected.

So again, if you if you have if you have taken the right thresholds, but if your circuit is not designed properly, if it has got huge wireless, it might lead to negative these two things that should be taken care of by choosing the thresholds and by designing the circuit.

Circuit design step is a really very important step to if you if you try to figure out what if you try to conclude from this video, okay, next we are looking into the next topic, which is having that transition for transition time.

So transition time is pretty easy to identify because you have the values of slew low rise and slew high rise.

For the high rise waveform, you just subtract the high minus low and that will give you a slew that will give you the slew of a waveform.

Okay.

Similarly, for the falling for the falling waveform, you just need to subtract the high minus low for the fall and you will get the slew waveform.

Okay, so let's try to take some values or take some typical values for the slew rise and slew fall.

So for example, the values that we take is 20% generally, it could be 30% and also it could also be 30%.

So slew low rise threshold is 20% and slew low fall threshold is 20% and we'll take a value of 80% let's say for the rise.

So when you say 20%, it's 20% of VDD, 80% of VDD.

So with these definitions or with this timing threshold in mind, let's try to identify some transition time.

We have the input transition time or the rise transition time and the fall transition time.

Okay.

In case of a setup, in case of a circuit, there is no input or output.

There is a rise transition, there is fall transition and there is rise delay and there is fall delay.

So let's try to look into one of them.

So let's say you have this definition set aside and that 80% of your VDD is 1.44 volts, 20% of your VDD is 0.36 volts and with this now you have got all your threshold points.

So this is your slew low threshold.

This is your slew high threshold for the rising waveform.

Okay.

This is your slew low threshold, slew high threshold for your falling waveform.

Now if you want to find the rise transition, just subtract this point minus this point.

Let's see what value do we get.

So if you have this point, the input slew comes to about 26 picosecond, which is from this point to this point, which is subtracted.

Your output slew comes to about 54 picosecond, which is again high minus low.

It has always to be high minus low.

It can't be the other way around.

So that brings us to an end of the timing characterization for the transition time.

Let's look into the next one.

And since I'm running a bit late out of time, let's try to look into the output current waveform.

This is again a very important timing characterization thing that we need to look into.

Output current waveform and the output voltage waveform.

And these are the outputs.

Your output slew comes to about 54 picosecond.

It has always to be high minus low.

It can't be the other way around.

So that brings us to an end of the timing characterization for the transition time.

Let's look into the next one.

And since I'm running a bit late out of time, let's try to look into the output current waveform.

This is again a very important timing characterization thing that we need to look into.

Output current waveform and the output voltage waveform.

And these are the waveforms that leads us to the CCS timing dotlibs and ECSM timing dotlibs.

SPICE DECK

But connectivity, connectivity information about the, about the netlist, so it's a basically a netlist that has everything.

It has got connectivity information, it has got the, it has got the inputs that has to be provided to the simulation, it has got tap points at which we take the output and so on.

So it has got everything.

So the first step is to create the SPICE tag.

Okay?

In the past videos we had created SPICE tags for a single NMOS, for NMOS only.

In this case we have to create a SPICE tag for the complete netlist, and in this case the netlist looks a bit different than what we did before.

So in this case you have the PMOS over here.

The symbol of the PMOS is a bit different.

Initially we saw the symbol of the PMOS with a circle over here.

In this case the symbol of the PMOS looks something like this, where you see an arrow which is pointing towards outside.

In case of an NMOS the arrow is pointing towards inside.

There are significances of these arrows.

I'll come to that in a bit from now.

So this is how your PMOS looks like.

This is the substrate terminal, and this is the substrate terminal of your NMOS.

And this is the PMOS, and this is the NMOS symbol.

And the reason we are doing this, because in your SPICE tag or in your SPICE netlist, we need to define the connectivity of your substrate as well.

Substrate is a potential, a potential component or a potential pin on your NMOS and PMOS or the transistors.

So you need to define the connectivity of that substrate pin as well.

Okay?

And we also looked into the significance of the substrate pin, where it tunes the threshold voltage of your PMOS and NMOS.

So we have already looked into all those.

Okay?

So this is your PMOS.

This is your NMOS.

There's a connectivity information.

The source of your PMOS is connected to VDD.

The source of your NMOS is connected to VSS.

And you have your input voltage, and you have your output load capacitor.

So there is a lot of theory behind this output load capacitor.

The value of the output load capacitor doesn't come just like that.

There are a lot of equations.

There are a lot of theory that goes behind the computation of this particular load capacitor.

But in this case we'll assume a load capacitor value, let's say 10 femtofarads and move on.

And in the next one, in the advanced courses, we'll be looking into how do we exactly get this load capacitor.

Okay?

So it is basically decided by the circuit that is connected, by the input capacitances of the circuit which is connected at this point.

So we'll get into all those in the more advanced videos, in the videos where we're looking into the dynamic characteristics of your CMOS.

In this case, we are looking into the static behavior of your CMOS.

So in this case, we'll just assume load value and move on.

Okay?

So we have to define the component values.

The next thing is to define the component values.

By the component values, I mean the values for your PMOS and NMOS.

So in this case, the PMOS is given the value of W by L as 0.375 by 0.25.

It says your channel length is 250 nanometers or 0.25 micron and your channel width is 0.375 micron or 375 nanometers.

And same we are going for M2.

So there is a scenario where we understand the PMOS should be wider than the NMOS and we'll come to that, the reasons why the PMOS should be wider than NMOS.

In this case, we'll just assume the values of 0.375 and 0.375 both for PMOS and NMOS.

So we are taking the same size PMOS and NMOS and see what happens in the characteristics.

Ideally, your PMOS should be twice or thrice bigger than your NMOS.

By that, I mean 0.375 should be 0.375 multiplied by 2 in that range.

But there are again reasons and there are applications for both of them.

You need this one, you need this kind of inverter as well as an inverter which has got a PMOS bigger in size.

You need both kind of inverters for your application.

So we'll get into those in a bit from now.

For now, we'll just look into the PMOS and the NMOS of the same size and see what happens to the characteristics.

And the next component value that we have to decide is the output load.

Basically, we just assume the value of 10 femtofarads over here.

Basically, this value comes from a lot of calculations.

Basically, the circuit that is connected at this node, the following circuit, which is connected at this node, has got some input capacitances and all those things.

So we have to look into it.

We have to compute those load capacitances.

In this case, we'll assume a value of 10 femtofarads.

Okay.

Next, we'll define the values of the input gate voltage.

So we'll assume a value of 2.5 volts.

So generally, if you have a node value, if you have a channel length of 250 nanometers, usually the voltage is kept in the multiple of 250 nanometers.

So in this case, it's 2.5 volts.

Had the channel length been 0.1 micron, this would have been 1 volts or so on.

So it's just a calculation that happens and I'll get into the calculations in some time from now.

For now, we'll take, we'll assume a gate voltage of 2.5 volts.

Okay.

We'll assume a drain voltage or the main supply voltage as 2.5 volts and we'll move on.

So these are the component values that we are picking up.

Okay.

We have these values for PMOS.

We have the values for NMOS, the load capacitor and the input and the output voltage.

Sorry, input and the supply voltage.

Okay.

And these are all the VSS terminals.

So let's club the VSS together.

So we have this common VSS for all the supplies that is present over here.

Okay.

Next step is to identify the nodes.

So for example, in the past couple of SPICE videos, we have looked into what does a node mean.

So in this case, node means those two points between which there is a component.

For example, over here, this and this becomes a node because in between there is one component.

So in order to define this component in a SPICE netlist, you need a node.

You need to say that this component lies between this and this node.

So that's basically a definition of the node.

For example, let's look into, let's look into this and this node.

So, so the, sorry, this and this node.

So in this case, this M1 transistor is defined between three nodes, this node, this one and this one.

Okay.

So there are basically no components connected between this node and the transistor, this transistor and this node, this transistor and this node.

So there are no components defined in between them.

So we can very easily define this particular transistor in between these three nodes.

Okay.

Let's pick up another component.

For example, let's say we have this load capacitor.

The load capacitor can be very well defined between this node and this node because this is a node which is common to all of them, common to the VSS common node and we have this particular node which is, which is present at the end of the parallel plate capacitor of your output load capacitor.

Okay.

So there is nothing other that is coming in.

Had there been a resistor over here, then there would have been this node and there will be another node which is present over here.

Since there is nothing present between this point and this point, this can be considered as a potential node.

So these nodes are basically required to define your SPICE netlist.

And I'll, I'll show you in some time from now how, how basically these nodes are important.

Similarly, if you look into this VDD, this VDD can be very well defined between this node and this node.

Okay.

So let's name this, name these nodes.

We'll give this node name.

We'll call this as in.

We'll call this node as VDD.

We'll call this node as out and this node as zero.

Since VSS, we'll call it as zero.

So basically if you want to define an output load capacitor, we'll say that the load capacitor lies between out and zero.

If you want to define the VDD, we'll say that the VDD lies between VDD and zero.

Okay.

And I'll, we'll look into, we'll look into one of the, one of the definitions of this, one of the applications of this particular nodes.

So let's start writing this SPICE net.

Okay.

So for example, these are just comments.

Anything that starts with star are comments.

So this is a model description and we are, we will, now we'll start with the netlist description.

So let's say M1, the MOSFET M1, you have, in the past video, we have looked into the, we have looked into the syntax of the MOSFET in a SPICE.

It's drain gate source, source, drain gate substrate source.

So M1 will be drained.

The drain is out.

If you see at this particular point, your drain is out for the M1 MOSFET, your drain is connected to the out node, your gate is connected to the in node, okay, your substrate is connected to the VDD node, and your source is also connected to the VDD node.

So this particular, this point, this defines the connectivity of M1.

PMOS says that this M1 is of type PMOS, okay, W is 0.375 and L is 0.25.

So this line completely describes this PMOS transistor, which is defined by the name M1.

So M1 is connected between out, in, VDD, and VDD, okay.

It's a type PMOS transistor.

It has got a W, a width of 0.375 micron.

It has got an L of 0.25 micron.

Okay.

So, what we'll do is we'll, since I'm running a bit late out of time, we'll stop at this moment.

**SPICE DECK contin**

Let us continue our discussion on describing the SPICE deck for the CMOS inverter.

So we had described already the PMOS transistor in the last video.

So this was the PMOS transistor and this was its respective connectivity information.

So it lies between D, G, S, S so drain, gate, source and substrate.

In the last video I had mentioned this as substrate and this as source.

So just correcting over here it is drain, gate, source and substrate.

So your out terminal, the drain terminal of your PMOS is connected to the out node, your gate terminal of your PMOS is connected to the in node, your PMOS transistor source and substrate are connected to VDDs respectively.

And this is the description for it, it is a PMOS transistor, I will come to where we get this PMOS name and all.

And this is the channel width and channel length of the PMOS transistor.

So moving on to the NMOS transistor.

So we have this as the NMOS transistor and it is connected between out, in, 0, 0.

So when you say we have given a name to it, it is called M-thread.

So drain of your M2 is connected to out, the gate of your NMOS is connected to in node and your drain and source and substrate are connected to 0 node.

So that is what we have mentioned.

And it is of the type NMOS and width of the NMOS is 0.35 micron and the channel length is 0.5 micron.

So this was about the connectivity information of this CMOS inverter only.

Now we need to describe the connectivity information for others as well.

So let us look into the other components.

The first one we will look into the output load capacitor.

So the output load capacitor is connected between the out node and the node 0 and its value is 10 femtofarads.

And that is what is precisely described over here.

Your C load, the name of the output load capacitor is connected between out and 0 and the value of it is 10 femtofarads.

Next you have the supply voltage which is VDD.

The supply voltage VDD is connected between VDD and 0 and the value of it is 2.5 volts.

So that is a very simple description of your supply voltage.

And similarly you have the input voltage which is Vin and Vin is connected between the in node and the node 0 and its value is 2.5 volts.

So these are very simple compared to the nMOS and pMOS descriptions.

So next we have to, we have to give some simulation commands.

So the simulation commands come something like this, we say that, so we had seen some simulation commands in the past when we were trying to describe an nMOS SPICE simulation.

So in that case we had sweeped the input, the gate input voltage of your nMOS transistor from certain voltage.

Similar thing we will be doing over here.

We will be sweeping the gate input voltage from 0 to 2.5 at steps of 0.05.

So the reason we are doing this, we need to calculate the voltage at the output or the waveform at the output while we sweep the input voltage because that is the voltage transfer characteristics.

Your output voltage versus the input voltage.

And that is what we are trying to do.

We will be sweeping the input voltage by from 0 to 2.5 at steps of 50 millivolts or 0.05 volts and measuring the output voltage or measuring the output waveform.

Okay, so that's what we will be doing over here.

And then the final step is to describe the model file.

So this is very important.

If you see this particular model file, this is the one which has got the complete description of your nMOS and pMOS transistors.

So all the technological parameters, all the parameters which is related to 250 nanometer technology node are all described in this model file and I'll open this model file in a bit from now.

So this is the file from which it takes all the descriptions of the nMOS and pMOS.

So in this file you will see, you will see an attribute called as nMOS and all the descriptions are related to that particular nMOS.

Similarly, you will see a keyword called as pMOS inside this model file and all the technological parameters are related to this particular pMOS.

Okay, so without wasting any time, let's try to move on and do the SPICE simulation.

So we will be doing a SPICE simulation for this, for this particular specifications.

You have Wn which is the channel width of your nMOS transistor is equal to the channel width of your pMOS transistor is equal to 0.375.

And that's, that's what we had seen over here.

Your channel width we have kept constant.

Okay.

And the channel length for your nMOS and pMOS are kept as 0.5 micron, second constant.

And your W by L ratio for your nMOS is equal to W by L ratio for your pMOS.

Ideally we expect W by L ratio of the pMOS to be slightly greater than W by L ratio of the nMOS.

And I'll come to that, the reasons and under what, under what conditions do we need them to be same.

There are various applications for both of them and I'll show the applications as well.

So we'll do a SPICE simulation for this particular scenario and I have the SPICE window open.

So this is the SPICE window, which is already open.

We'll start with the model file.

So let's, let me show you the model file first.

So this is your model file.

So this is where it has got all the description.

You have the model nMOS.

Its name is nMOS and these are the all technological parameters with respect to this particular nMOS.

Similarly, we'll have something like pMOS also.

So this is the dot, this is the syntax of describing the pMOS technological parameters.

So you have this pMOS, which is the transistor name and this is the complete description of your pMOS transistor.

So it will have all, all of the parameters that we described in our past videos.

For example, it will have the oxide thickness.

It will have the threshold voltage under zero substrate voltage.

It will have all the coefficients, K1, K2, K3.

It will have, basically if you go through this file, I'll upload this file.

You'll have everything which is related to this particular pMOS and everything that we discussed in our past videos.

So this is your model file.

So this is where it has got all the description.

You have the model nMOS.

Its name is nMOS and these are the all technological parameters with respect to this particular nMOS.

Similarly, we'll have something like pMOS also.

So this was about the model file and this is what we have included in our, in our netlist description.

So let's look at the netlist first.

So we'll look into, this is the netlist.

So this is the netlist that we had just described above.

You have your, you have your pMOS transistor.

You have your nMOS transistor and the load output capacitor and everything.

So we will do a quick simulation on this one.

So we'll pick up the SPICE.

So we'll first go to this area over here.

Okay.

And next we have to source this.

Source the circuit file.

Let's do it.

Okay.

So, so now this says that when we actually source this, it says that the circuit is now present in this particular simulator.

Okay.

Now we'll execute the circuit.

The command is run.

Okay.

We will set plot.

And when you do, when you just type set plot, you will see which, which plot is now currently present in your SPICE simulator.

So it's a DC1.

And this is a DC transfer characteristics.

Okay.

So we'll give DC1.

Okay.

Now we'll look into the, we'll look into the, the nodes that are, the node voltages that are present to us.

So we have these voltages that is, that is currently present, present with us.

And we have to plot out versus in, because that's what we're looking at.

This is the node where you supply and which is the voltage which you have been sweeping.

And this is the output node that you want to look into the waveform.

Okay.

So we'll plot this and see what do we get.

And this is, this is what you get.

You get a, you get a need waveform or need V voltage transfer characteristics of the, of the CMOS transistor.

And this is the kind of waveform that we have been looking into.

But if you see, this particular thing is slightly shifted towards the left.

Okay.

And I'll come to that why it is shifted towards the left.

But if you see, this particular transfer characteristics closely represents what we described in the theory lectures.

Okay.

So let's minimize this.

So let's move the waveform in this particular slide and we'll move into the next, next kind of simulations where we have kept the, kept the NMOS bit as 0.375, but we have shifted the PMOS bit to 0.9375.

So 0.9375 is basically 2.5 times of, of your NMOS standard bit and let's confirm that.

So 0.375 multiplied by 2.5.

So you have 0.9375 and we have, we have increased the width of your PMOS transistor 2.5 times than that of the NMOS transistor.

Why are we doing so?

I'll come to that in a bit from now.

Okay.

So let's try to do this by simulation for this one.

We have, let's close this and we have a separate, this will be the, this will be the spice tech for a similar scenario that we just talked about.

Everything else remains the same except that your PMOS bit goes to 0.9375.

So we'll do a, we'll do a similar spice simulation.

We'll go to the, we'll go to the spice, spice simulator.

We'll source, let's copy this.

We'll source the circuit file.

We'll execute this circuit now.

We'll set the plot.

We'll see what is the new plot that is available.

So these were the past plots that were available and we plotted DC1 for your, for your previous case where your PMOS and NMOS bit was same.

Okay.

DC2.

And in this case where your PMOS and NMOS transistor bit vary by 2.5 times.

So this is, this is DC2.

So let's set that plot.

Okay.

Display, let's look into what is available with us.

So we have again the in node and the out node which is available.

So let's plot them.

Plot out versus in.

And yes, this is what you get.

Okay.

So if you closely look into this waveform and the previous waveform, you will notice a difference over here.

And what is that difference?

Let's, let me try to come back on that particular difference in the next video.

Just observe this particular difference that, that this particular thing lies somewhere at the left side.

So if this was a complete range of the voltage, 2.5 volts, and if this was the center of your voltage, if this was the center of your graph, so this particular transfer characteristics is shifted towards the left.

In this case, it's exactly at the middle.

**SPICE DECK contin 2**

simulations for a device where pMOS was 2.5 times greater than nMOS and basically the width of the pMOS was 0.9375 and this is 2.5 times of your nMOS.

So we did that for a purpose to show the difference in the SPICE characteristics.

So we'll try to analyze this particular SPICE waveform.

For that what we'll do, let's keep these two waveforms side by side.

So we have, in this side we will have the SPICE waveform for Wn equal to Wp where your pMOS nMOS sizes were exactly the same and it is 0.375 is the width and the length is 0.25.

That's constant across both the analysis that we did.

So in this case W by L ratio of the nMOS is equal to W by L ratio of the pMOS and that's equal to 1.5.

Okay and the next next to the scenario where your Wn was kept to be same 0.375 just as the previous experiment.

Your Wp was 2.5 times of Wn and your nMOS pMOS length channel length was kept to be same.

So in this case W by L ratio of pMOS was 1.5.

In this case we increase the W by L ratio 2.5 times of 1.5 which is 3.75.

You can just do this divided by this and we'll get this number.

So it basically says your pMOS is bigger in size than nMOS and each one of them has their own uses, has their own applications.

I'll very soon tell you where this kind of waveforms or where this kind of buffers are used in a real circuit, in a real physical design circuit.

So I'll come to that.

So these two have their own applications.

There is nothing called as good or bad.

Everyone has got their own applications.

So let's start with the waveform.

So this was the waveform and this was the waveform for a device which was bigger than nMOS.

So clearly if you see the shapes of the waveforms are same.

You can see the shape of this one and the shape of this one irrespective of the voltage levels at which they are switching.

If you look into just the shape, they're almost the same.

There are a few glitches in this region and this region and also over here as well but the shape remains the same.

So what does this tell us?

This tells us that the CMOS inverter is a very robust device.

There are certain parameters that will define the robustness of the CMOS but this says that your CMOS whenever your input is zero, this is your Wn, whenever your Wn is zero, your output is high, whenever your Wn is high, your output is low.

So this characteristic is being maintained across all kinds of CMOS inverters, across all sizes of CMOS inverters.

So CMOS as a circuit itself is a very robust device and that's why CMOS logic is the one that is being widely used for any of the logic gate designing.

So that brings us to the parameters that will define the robustness of the CMOS and one of them is the switching threshold.

For the first one is the switching threshold.

Switching threshold means the point at which the device switches.

So we are going to evaluate the static behavior of your CMOS and we'll come up with some parameters like switching threshold, noise margin and so on that will define the robustness of your CMOS.

So basically it will prove to us that CMOS inverter is the most widely used for building any of the logic gates and why it is used because the characteristics of your CMOS inverter is such that it becomes very useful for these devices to be used in building of any of the logics.

And one such parameter that will define the robustness of your CMOS inverter is the switching threshold.

So let's find out what is switching threshold.

Switching threshold is a point where your VIN is equal to Vout.

So how do we derive it?

We basically draw a line, a tan 45 degree line from this point till the end and then identify the point at which your VIN is equal to Vout.

So in this case, let's try to zoom in this one.

If you look into this case.

So this is the point where your VIN is equal to Vout.

So this is the point where your VIN is equal to Vout in this case and this particular point tends to be somewhere around 0.9 volts.

So it says that for this particular device, your switching threshold, this is a point where your VIN is equal to Vout and that point lies somewhere at the intersection of this one and this one.

So it lies somewhere over here.

If you look into, if you zoom into, if you zoom more into this area, it will be about 0.9.

Okay.

That's with this kind of device where your WP is 0.375 and WP by LP is 1.5.

And in the other case where you have another device, a bigger one.

So in this case, if you zoom into this area, the point roughly comes to 1.2 by something.

It's not exactly at the center if you see this one.

This point comes to roughly about 1.2.

This is the point where your VIN is equal to Vout.

So that's basically an area at which you see the switching threshold to be somewhere where your VIN is equal to Vout.

So this brings us to many conclusions over here.

So this area that you see for your CMOS inverter is a very critical area because your input is equal to your input is equal to output.

That is first thing.

Second thing, we have seen in the past that this is the area where your PMOS and NMOS both are in saturation region.

It means both are kind of turned on.

If both are turned on, there is a very high chances of leakage.

I'll talk about leakage in a bit from now.

So there's a high possibility that the current flows directly from power to ground.

So in this area, in this area you see a lot of current that has been flowing.

In this area, since one of your devices is off, same thing with this area.

In these two areas, since one of your devices are on or off, in this case your PMOS is off and NMOS is on.

Sorry, your PMOS is on and NMOS is off.

In this case your NMOS is on and PMOS is off.

In that case, there is no direct current that is flowing from your power to ground.

But in this area, this is a very critical area where you see a lot of power that is leaking, a lot of current that gets flowed from power to ground.

So this is a kind of short circuit device.

I'll talk about these concepts in a bit from now.

Let's move on and look into the switching threshold values.

These are the switching thresholds for this inverter and for this inverter and the values we just identified.

In this case, it was close to 1 volt, so it's 0.98.

In this case, it was not exactly at the center, but it is about 1.2 volts.

So these are the areas in which your PMOS and NMOS both are in saturation region.

So we had looked into the waveform that while deriving this particular waveform, we had looked into each and every point of this one and what regions of operation is your PMOS and NMOS are in.

So in this case, we identified that your PMOS and NMOS are in saturation region.

It means that they are fully, they are kind of turned on in this area.

And the reason they are turned on is because their gate voltage, I'll come to that point, the gate voltage almost crosses, it's very much above the threshold region.

So in this case, this is a case where your V in is equal to V out.

When you say V in is equal to V out, your gate voltage is equal to your drain voltage, which means VGS is very, very greater than your threshold voltage.

So this is the conditions, your VGS is equal to VDS.

And also another point is, at this particular point, at this particular operating region condition, the current that is flowing in this area, that's almost same.

Only the directions will be different, but the current is at flow.

So for example, if there's a current flow from this area, from PMOS to NMOS and from this capacitor to the NMOS, the direction of the currents are different, but the values of the currents will be exactly the same.

That's why we write at this particular area, your VGS is equal to VDS.

And your current has got an inverse direction, your IDSP, the drain current of your PMOS will be this direction and the drain current of your NMOS will be in this direction.

So if you note the directions of the current at this node, your PMOS current in this way, your NMOS current in this way, so the direction of the currents are inverse.

And as a result of that, you see your IDSP is your negative IDSN.

So we have the boundary conditions, we have the conditions at which your switching threshold will arrive.

So your IDSP is equal to negative IDSN, your VGS is equal to VDS.

And we also know what is IDSP, what is IDSN.

So the next part is to, next work is to derive the value of VM.

So basically we need to solve certain equations so that we can get an early validation or early value of VM by just having the values of WN by LN or WP by LV.

So it's basically, we have these values and we need to calculate the VM or vice versa.

**Static and dynamic simulation of CMOS Inverter**

So in the last video we came up with a task where we modified the PMOS sizes based on an integer of numbers of NMOS and see what is the value of switching threshold, okay.

So basically we are trying to prove the robustness of switching threshold and we'll try to plot switching threshold against how does the switching threshold varies between PMOS and NMOS sizes, okay.

So let's do that.

So I have a spice simulator which is open over here, okay.

And we'll take the same circuit that we took in the past.

So we have your PMOS which is 0.375, your NMOS is 0.375 and based on that values of W by L ratio is 1.5, okay.

So this is what we are going to do.

We have your dot DC as, so the same thing that we did in the past, okay.

So let's try to do a spice simulator.

We have a spice simulator open, okay.

We will source the circuit file and copy it over.

Okay.

We'll source it over here.

We'll run the circuit.

We'll set plot, the plot that is available to us as DC1.

So let's do that, okay.

And let's see what are the plots which are available.

So we'll plot out versus, this is what we have done in the past and this is the DC transfer characteristics of your CMOS inverter.

So if you want to calculate the switching threshold, you just need to plot a line, a 45 degree line from the 00 point and see where does that line intersects with this particular, with this particular curve.

It basically says that, because whenever you draw a 45 degree line, its slope is 1.

And whenever that particular point intersects or this particular curve, it basically says that that is the point when your V in is equal to V out and that's your switching threshold.

So this was, this was the DC transfer characteristics.

In the same time, in the meantime, what we'll do is we'll also try to do a delay calculation, a dynamic simulation of your CMOS inverter.

The reason we are doing this is because this is a very good chance to identify what is the value, what is the value of your rise and the fall delay of your CMOS inverter and how does it vary with varying VMs.

It's a very good point to start and we'll be doing a whole course on dynamic simulation.

So let's wait when we go over there.

The dynamic simulation is a huge set of lectures and we'll be covering it in a separate section, of course.

In this case, we'll just touch upon the dynamic simulation where we identify what is the, what is the value of your rise and fall delay and how does that vary with varying switching thresholds.

So let's do that.

With the same configuration, we'll bring up one more spice deck.

So this is how you do your dynamic simulation.

Everything else remains the same except the input that you provide will be a pulse.

And the simulation command will be a transient, it's called a transient analysis.

I'll come to the definition of transient analysis.

This is basically what it's called.

It's called a transient analysis and the input that you will be providing that is a pulse.

A pulse is a waveform, which is a function of time.

So I'll show you what does a pulse mean.

So what we'll do, let's try to show what a pulse means.

Or how does the definition of a pulse is being written into a spice deck.

So if this is the pulse and if this is the spice deck, this is what it means.

If you have a pulse definition, it starts from zero.

It starts from zero volts.

It ends to 2.5 volts.

It ends to 2.5 volts over here.

The shift is zero.

So it says that your pulse is starting exactly at zero dot zero, at time zero.

So if this would have been, let's say 50 picoseconds, this pulse would have been started after 50 picoseconds.

So that's what a pulse means.

You have a rise time of 10 picoseconds, rise time of 10 picoseconds.

You have a fall time of 10 picoseconds.

And the pulse width is one nanosecond and this complete cycle is two nanoseconds.

So this is a complete cycle of two nanoseconds.

And if you just try to calculate the pulse width, which is a 50 percent duty cycle pulse width, it's one nanosecond.

So this is what your pulse means.

And we are going to supply this particular waveform as an input to your CMOS and do a transient analysis and see what we get.

So let's do that as well.

So we'll again bring up our spice simulator.

And we have the netlist present over here.

So let's copy it.

So we'll source the SEPIC5.

CIR.

Now we'll run it.

And we'll set the plot.

So you'll see a difference in the previous spice simulations and this one.

In this case, the SEPIC plot will tell you it's a transient analysis.

It basically says that we are doing a transient analysis.

In the previous case, since we did a DC transfer characteristics, it basically says it's a DC one or DC transfer characteristics.

In this case, we are doing a transient analysis, so it's called TRAN2.

So we'll put TRAN2.

And let's see what we get.

So we'll see what we get.

So we'll see what we get.

So we'll see what we get.

So we'll see what we get.

So we'll see what we get.

So we'll put time to, and let's see what are the plots available to us.

So we need to plot the input as well as the output waveform to calculate the delay.

When you calculate the delay, it's basically 50, 50, 50 percent from, it's basically a point at which your input crosses 50 percent to a point where your output crosses 50 percent.

Let me show you that.

So we'll plot out versus time, okay, and the input waveform.

So this is what time meant.

So it's not really visible, I'll make it visible.

So this is your input, and this is how your output switches.

So if you want to calculate the rise, let's say you're going to calculate the rise delay.

So this is your rise, this is your point where your output voltage rises, and you need to go to 50 percent of this particular waveform, so 50 percent is 1.25 volts.

We'll zoom in, this is the 1.25 volts, we'll zoom in even more, okay.

And once you get there, you just need to click on this one.

So this is your time axis, which is in nanoseconds, this is your voltage axis, so if you want to calculate the delay, you just need to identify the difference between this point and this point.

So this is your rise waveform, it's a zoomed in version of your rise waveform.

This is the falling edge of your input waveform.

So if you just click on this one, if you click on this one, or basically this is the way the spice simulator works, you just need to identify what is this point from the time point of view, you need to identify what is this point from a time point of view to calculate the difference, and then we'll get the rise delay.

So let's do that, we'll click on this one.

So this is basically 1.014 on your x, 1.014 nanosecond, 10 raised to minus 9 is nanosecond, and y 0 is equal to 1.25, or approximately equal to 1.25.

So this is the one, this is one point, we'll click on the other one, and you'll get the other one, 1.16277, and then take the difference between this one and this one, and this will give you a rise delay, so let's do that.

So we'll clear this, 1.16277 minus 1.01446.

So 1.14, or 148 picosecond is your rise delay.

We'll do the same thing for fall delay as well, let's close this one.

For the fall delay, we need to find out the point at which the output falls.

In this case, the output falls at this point, and this is when your input is rising, and we'll need to take the 1.25, the 50% of your input voltage waveform, and let's see how you want to do it as equal delay, we'll zoom in, 1.25, we'll zoom in even more, we'll zoom in even more, and let's see what we get.

We'll follow the same strategy, we'll click on 1.25 over here, so your X0 or the X axis shows as 2.00486 nanosecond, same thing we'll click over here, we'll get 2.07653 nanosecond, and then calculate the difference between them.

So we have our calculator.

We'll take 2.07653 minus 2.00486, and this comes to about 71 picoseconds, or 0.01 nanosecond.

So this was about how do you calculate the rise and fall delay for the inverter, and let me jot it down.

So for a value of WL by WB, sorry WB by LB, equal to WN by LN, this is the rise delay and the fall delay, and we just calculated it, 1.8 picosecond is the rise delay, and the fall delay is 71 picosecond, and you draw a line, you draw a 45-degree line from 0 to 0, and this is the point at which data sits.

And if you zoom in over here, so let's zoom in over here, the quick way to do it is click on this line and just zoom in.

So if you zoom in over here, if you see, this particular is the point where your, this curve is intersecting with the 45-degree curve, and that particular curve is somewhere at 0.98, it's somewhere lying at 0.99 volts.

So your switching threshold is equal to 0.99 volts, your rise delay is 1.8 picosecond, your fall delay is 71 picosecond.

Similarly, what we'll be doing is we'll be doing the same thing for all the set of combinations, I know it's a tedious work, but we need to do it, and we'll plot a table specifying what is the value of your switching threshold, what is the value of your rise and fall delay, and eventually deduce a conclusion about the CMOS inverter robustness.

**Create Active Regions**

a substrate.

A substrate is something on which you fabricate your complete design.

So basically, we do a physical design step.

We do all those kind of steps.

And finally, the GDS2 or the layout that you see on your physical design step or any step, that gets actually fabricated onto a substrate.

So, the first step is to select a substrate.

Normally, we will go for various kinds of substrates that are available, but we'll go for the most common one, which we use for most of your mobile devices or any kind of devices that you see or any kind of chips that you see in the real time.

So, we'll go for a P-type silicon substrate.

It will have some different properties, and some of them are high resistivity or the resistivity property of the substrate.

The doping level.

Doping is the process of adding a foreign impurity into a P-type substrate.

So, something which comes as a foreign material, it's called as doping.

That's why the name doping has been put over here.

The doping level of this particular substrate is meant to be 10 to the power 15 per centimeter cube, and the orientation of this particular silicon substrate is 100.

We'll talk about orientation in a separate course, in a separate step.

In this case, just focus on the kind of substrate that we have, the resistivity, and the doping level.

The reason for having this kind of doping level is that we have to maintain a doping level which is less than the weld doping.

So, when we talk about weld doping, welds are nothing but something which we'll be creating over here to fabricate your PMOS and NMOS separately.

I'll come to that in a bit from now.

It is coming in the further sections.

So, the doping level is maintained low so that it is lower than the weld doping, and the weld doping are the ones which we'll be using to fabricate your PMOS and NMOS.

I'll come to that in a bit from now.

So, we have selected a substrate.

It's called a P-type substrate.

The next step is to create the active region for transistors.

So, when we talk about active region, these are the places where you actually see your PMOS and NMOS.

So, on your P-type substrate, we are going to create some small pockets, or some small pockets which will be called as active region, and in those pockets, we are going to create the PMOS and the NMOS transistor.

So, those pockets will be connected, and the connections to those pockets will be brought about in the higher metal layers, but the first step is to create those buckets, or create those pockets.

So, let's try to do that.

Those are called as the active region for transistors.

So, the first step to do that is to create an isolation between those pockets, because the pockets within itself, or the active region, or the active transistors onto the P-type substrate should not interfere with each other's working.

So, that's the goal, and the first step towards that is to create isolation between each and every pocket.

So, let's see how do we do that.

The first step is to grow a silicon dioxide.

It acts as an insulator.

It acts as a very good insulator.

So, we grow a 40 nanometer of silicon dioxide on your P-type substrate.

That's the first step.

The next step is to deposit a layer of silicon nitride.

It's about 80 nanometers.

So, the reason we're depositing Si3N4 or the silicon nitride onto a silicon dioxide will be very much evident in the upcoming videos, so let's stick to that.

So, we'll deposit an 80 nanometer layer of silicon nitride onto the 40 nanometer layer of silicon dioxide, and now is the step where we have to create the pockets.

Now, the first step to create the pockets is to identify the regions where we have to create the pockets.

So, the first step is to deposit a layer of photoresist.

So, I'll come to the word resist.

So, when we say photoresist, it's something as a film.

It's a negative or a positive film that you see for a camera.

Let's say when you, in the older generations camera, you had a positive film and a negative film.

So, the concept of that camera can also be applied onto a photoresist.

So, photoresist is a film on which we are going to do some process that will clearly define some regions over here.

I'll come to that in a bit from now.

So, let's see how do we do it.

So, the first step is to deposit one micron of photoresist, okay, and this is the one on which we will create some mask.

So, your mask one.

Your mask one is nothing but now we are talking about layout.

So, when we talk about layouts, those are nothing but called as masks in a fabrication term.

So, when we talk about polysilicon or any kind of layout that you see in your custom design, those are converted to mask, and this is the mask.

This is the first mask that we'll be picking up from that particular area.

So, let's look into the mask one.

So, it will look something like this.

This is from your top level.

The layout looks a bit different.

This is a cross-sectional view of your mask, one of your layout.

It is nothing but a protection layer.

So, for example, now I decided to create my weld region only in this area and in this area.

So, I have to protect this area from getting exposed to anything else.

I have to protect this particular area and also I have to protect this area.

Now, how do I do that?

The way to do this is using photolithography.

So, the photoresist that you see over here, that is now being part of the photoresist is being now protected by something called as mask.

So, when the ultraviolet light hits the photoresist, it doesn't hit this area, okay, it doesn't hit this area, but the only area that the ultraviolet light will be hitting will be this area and this area and this one.

So, these are the regions that are getting exposed to your ultraviolet light and hence there will be some chemical reaction that will be happening for this area, this area and this area, but there will be no chemical reaction for the area which is underneath your mask, okay.

So, these areas are getting protected.

These areas are exposed to the UV light and while these areas are exposed to the UV light, we do some chemical combination.

We basically wash this extra resist off from this region and this region and this region and we get patterns something like this, okay.

So, the extra resist that was exposed to the UV light is now getting washed away.

So, you have now a protected area over here on which you can do some process or basically you can protect this area from the further processes that I will be showing now, okay.

So, the next step is to remove the mask.

So, now, when we remove the mask, so now there is an area which is this small area, this small area and this small area under which if you do certain kind of chemical actions or if you do some kind of deposition or something or etching something, it will affect only this, only the exposed area and not the ones which are under the photoresist.

So, it is a protection layer for you, okay.

And that is very common and that is the reason we have all this custom layout and everything layout that we are going to discuss.

This is just an example of how does the layout actually helps you to do a photolithography process while you try to fabricate your chip onto a silicon substrate.

So, moving on, what we will do is we will now create patterns.

First of all, we will etch off the silicon nitride.

Now, since we have the photoresist present over here, only the area underneath the photoresist will be protected and the remaining area will be etched off.

The silicon nitride is being etched off from the other areas.

The next step is to remove the photoresist itself because the silicon nitride itself will act as a very good protection layer to grow the oxides on the other areas, okay.

Now, I will now show you how do you grow this particular, how do you grow the oxide in the remaining area and that will eventually act as an isolation layer.

But the point over here is the silicon dioxide area which is being under silicon nitride is protected from growing more, okay.

Similarly, for this area and the other areas when we put this particular thing into a furnace, into a very high temperature furnace, the other areas of the silicon dioxide will grow while the areas which is under the silicon nitride is getting protected.

So, let us see how does it look like.

We will put this particular complete thing in an oxidation furnace.

So, a furnace is a place where which burns at very high temperature like up to 9,000, 9,000 to 2,000 degrees Celsius and we will, that actually helps to grow the oxide in the other area.

So, we have used, we have grown the first level of silicon dioxide in an oxidation furnace itself but there is a second level of growth of oxidation.

So, we will do that now and it will look something like this.

So, if you see over here how the silicon nitride has actually protected the areas underneath to grow while it was not able to protect the area which were at the edges because the growth is so strong.

But it has clearly protected the area which is almost 90 to 95 percent of the region underneath has been protected from the silicon dioxide growth while the silicon dioxide has been grown in this area, this area and this area, okay.

And these are basically the thing called as isolation region.

Anything that any transistor that will be fabricating over here, they will lie in this area and this area.

So, these two transistors will not will now not communicate with each other because you have you have isolation, clearly defined isolation area which is made over here.

And this process that we have done is referred to as locus.

So, this is called as field oxide and the process is referred to as locus.

Locus is nothing but called as local oxidation of silicon, okay.

So, the process is referred to as local oxidation of silicon and this area is referred to as burn speaker.

So, this is how you protect your two different transistors from communicating with each other.

The next step is to is to remove the silicon nitride or the etch out, or etch out the silicon nitride or strip it out in a hot phosphoric acid, okay.

So, this is how you get your, you get your isolation region.

So, this areas and this one are your active region where you actually grow your transistors and the isolation region will protect this transistor not to be communicating with this transistor.

So, you have actually created electrical isolation over here, okay.

**Formation of Gate Terminal**

in your discussion with formation of gate.

So, gate by far is the most important terminal of your NMOS or PMOS transistor, because that is where you actually control your threshold voltage and threshold voltage is nothing but the turning on voltage of your transistor.

So, that is mentoring that voltage is really very important.

And as a result of that this application of gate terminals becomes really very important over here.

So, let us look into that, but before jumping into the fabrication steps we had looked into let us look into one of the snippet from one of my course called circuit design and spice simulation.

In that course we had talked about the threshold voltage equation over here, ok.

And it was the point it was the threshold voltage was very much dependent upon the body on the body of equation which was gamma.

And gamma is factored by by the doping concentration Na and the oxide capacitance.

So, in turn if you if you control the doping concentration and the oxide capacitance over here that will impact your threshold voltage.

So, the way things are done is in a fabrication steps you try to you try to maintain a doping voltage and an oxide capacitance in such a fashion that you get your required threshold voltage for the transistor.

And these are all fabrication steps, ok.

So, that is what are really very important to control over here, because these are quite controlled experiments that we are trying to do over here.

So, now, you see how does the fabrication are closely related to your threshold voltage and threshold voltage decides the turn on voltage of your gates which in turn decides the functioning of the gate.

So, these are all interrelated, ok.

So, let us try to look into how do we do that.

So, we are going to control the oxide capacitance and the doping concentration through the further steps.

So, the next step for the formation of gate is to control is to maintain the doping concentration.

So, let us see how do we do that.

So, for example, you have a P-well already over here where we are trying to fabricate the NMOS transistor.

So, we will mask we will use another mask force.

So, this is a 16 mask process out of which we are now completing the fourth mask.

So, the fourth mask the steps remains the same you actually spin photoresist on the complete surface area, then you protect one of the area by a mask force, you give ultraviolet light you expose this to ultraviolet light, this section of the photoresist gets again exposed to the ultraviolet light and then there is some chemical action that happens on this resist and this resist gets washed away, ok.

These steps are the repetitive steps that we have to do every time we are trying to do a photolithography or every time we are trying to introduce a mask or a layout, this is the mask phase.

So, this is the mask phase, this is the mask force, this is the mask phase, this is the mask force, this is the mask phase, this is the mask force.

**Lightly Doped Drain**

So, let us continue our discussion with the LDD or the Lightly Doped Drain Formation.

So, what we want to exactly achieve over here is this doping profile.

So, if you see over here, the doping profile over here, we try to attain is P plus P minus N.

So, what do you mean by this?

This is your N well where we are trying to fabricate the PMOS.

So, in case of PMOS, you have source and drain.

And source and drain are both are P type.

So, that is this one, P plus.

That is P plus doping profile which is coming over here.

P minus is the Lightly Doped Drain Formation or the LDD that we are trying to bring up now.

And N is what you already have.

So, this is the kind of profile or the source drain profile that we are trying to see over here. this is the kind of profile or the source drain profile that we are trying to see over here.

Similarly, for your NMOS which is being fabricated in the P-well, you need the N plus will be the source and drain.

So your source and drain for your NMOS are all N-type.

So N plus will be the source and drain doping.

N minus is the one that we are going to form now, the LED formation over here.

And P is what you already see.

So this is the kind of doping profile that we are trying to attain over here.

But the question is why do we want this kind of doping profile?

Why not P plus and N?

Or N plus and P?

Why do we need this P minus and N minus over here?

So there are two reasons for it.

So let me try to brief out the two reasons over here.

The first reason is the hot electron effect.

And the second is short channel effect.

So these are more of a device physics thing, but let me try to brief you about this kind of effects.

So about the hot electron effect, when the device size actually reduces, for the general practice is to not change or not redesign the power supply.

So in that case, your electric field, so let us consider this electric field formula, it is equal to V by D. So when your device size reduces, D reduces, your electric field increases.

And as a result of that, there are various effects that happen.

The first of the effects is that the high energy carriers, the energy of the electrons and the holes attain a tremendous amount of energy.

And these energy carriers can break your SI-SI bonds leading to some more additional electron and holes, which we do not want because we have controlled the doping profile very well.

And the second reason is the energies might be so high that it crosses the 3.2 electron volts barrier between the SI conduction band and the SI-O2 bands.

So this is a general energy that holds on, that is basically maintained between the conduction bands.

And if it crosses this band, it might just enter into the oxide layer, which is just present above your substrate.

And it might create reliability issues.

So this is the hot electron effect.

The next is the short channel effect.

So basically when your device size reduces, therefore short channels, basically when you move from 1 micron to 1 micron gate length to 0.5 micron gate length, the drain field, the drain area or the drain voltage just penetrates into your gate, just enter into the channel area.

Thus it becomes very difficult for the gate to control the current, the source and the drain current.

So these are the effects that we will be covering in detail in the device physics course.

Let us not get into more details of that, but we will try to control these two effects in the fabrication process.

So let us try to look into the process that we will be doing to generate the LED structure.

The process remains simple.

It is pretty simple what we did in the past.

And in this case, the gates that we have formed over here also help us to create this particular area open up for the further process that we want to do.

So it goes something like this.

So you create a mask basically.

And then this is mass 7.

You spin a photoresist over here.

You create a mask of mass 7.

It is called mass 7.

You protect this area.

UV light is getting directed over here.

And the UV light reacts with the photoresist.

And the photoresist reacts with the, there is a chemical reaction that happens in the photoresist.

And it gets phosphorized.

So the standard steps for photolithography still remains the same.

It holds good for whatever steps we do.

Okay.

And the next step is to, now in the case of B-well, we are trying to fabricate the N-MOS over here.

So we need an N-type impurity which is phosphorous.

So we have seen in the past phosphorous and arsenic as the N-type impurity.

Okay.

So we are trying to use the phosphorous or we can use some other N-type like arsenic.

But in this case, we will use a phosphorous N-type impurity and try to do an implantation over here.

Okay.

It will look something like this.

It will be the doses and the energy of phosphorous is carefully chosen so that the N- implant doesn't penetrate well into the well.

But it doesn't penetrate very much into the well.

It just stays over here.

So those are very much maintained to create an N- looping concentration.

N plus generally means heavy concentration.

N minus generally means lightly.

So that's why it's called lightly doped N.

This becomes a lightly doped.

Okay.

So this is your N- implant.

And this particular gate protects the phosphorous from entering into this channel area.

So next is basically again we do the same thing.

We apply a photoresist and do the photolithography step and we create this mask gate.

So this is a 16 mask process out of which now we are calling the mask gate.

So the next step is again to protect this particular area from further implantation and expose of this area for a P-type implant.

So boron is again a P-type implant.

The doses are again carefully chosen in such a fashion that it creates a P- implant over here.

And these are all lightly doped.

So what you see over here are too much lightly doped.

So we are not done with this one because we have to somehow maintain the further source and drain.

So when we try to generate the actual source and drain, it might get entered into this one or this structure might just get disrupted.

So we have to protect this lightly doped structure.

So how do we do that?

We create some spacers over here.

We try to create some side wall spacers and this is how we do it.

So what we'll do is let me try to first create the spacer over here and then I'll try to explain how does it protect from the further source and drain formation.

So we'll deposit a thick SiO2 or Si3N4 layer on the complete structure.

We'll do some anisotropic etching.

So anisotropic plasma etching is more of a directed etching, directional etching.

So what we'll do, it will try to etch each and everything over here, but the oxide which is present at the side wall will remain over here the way the anisotropic etching was.

If it hadn't been isotropic etching, it had removed all the oxide from each and every corner of the structure.

But in this case, for the anisotropic etching, we'll talk more about anisotropic etching in a separate course.

So this helps us to prevent this particular oxide layer from getting etched away.

And these are called as side wall spacers.

So now how does the side wall spacers actually help us?

So when we look into the further videos where we are trying to create the source and drain, the source and drain implantation will be effective only in this area.

And hence this particular portion of your lightly doped end is kept intact.

Similarly, when we try to do an N plus implantation over here, the N plus implant will be only in this area and this area.

And the side wall spacers help to maintain the N minus implant over here.

And hence we get the N plus, N minus and deep doping propellant that we expect.

**Source and Drain Formation**

If you see over here I just added a small thin layer of screen oxide, the reason to do that is to avoid the effect of channeling.

So, channeling is an effect where when you do a lot of ion implantation, if the vector velocity of our ions matches with that of the crystalline structure of your p-type substrate, the ions might go deep inside the p-type substrate without even hitting any of the silicon atoms.

So, in that case, it might just fill up this entire p-type substrate without even getting blocked at this point.

So, by adding a thin amount of screen oxide, we try to randomize the direction of the ions.

So, that is the purpose.

We will talk a lot about channeling effects when we go further in this video.

So, a thin screen oxide is added to avoid channeling during implants.

It will just help to randomize the direction of the implantation, and they will try to settle down in the areas that we want them to settle.

So, the next step is to use the mask 9 now.

The process is still the same, the photoresist is being spun on this particular structure.

It is being opened to ultraviolet light, and there is some chemical reaction, and this photoresist gets washed away.

So, this portion of your p-type substrate or of your structure is being now opened to other implants.

We will now remove the mask, which is present over here, and we will expose this structure to arsenic.

So, there are two types of negative or p-type impurities that we have been talking about, it is arsenic and phosphorus.

Let us try to go with arsenic at this point.

So, the amount of energy that will be needed is 75 kiloelectron volts, and this will just go right inside over here, and the sidewalls over here, which are actually the sidewall spaces, and the reason that we have added the sidewall spaces is to have the LDD being intact.

So, I will show you what happens when you have a sidewall spaces.

So, in this case, so arsenic will try to deposit only in this area and this area.

So, this is what happens.

So, now you see the structure over here, the LDD, the lightly doped drains are still maintained.

There is some amount of N or there is some amount of N minus implants present.

Now, these are the N plus implants that we just did using arsenic.

So, in this case, we have got a source, which is N plus, we have got an LDD, which is N minus, and we have a p-type area over here.

This is the kind of structure that we actually needed and we have discussed in the past video, ok.

So, similarly, we will do for the next thing also.

We will try to, this is called the N plus implant, and now we will do the same thing for the p-type also, for the pMOS as well.

So, we will have a mask 10 now.

So, we are closing to 16 mask.

So, this is the mask 10.

The same process, resist is being spun, it is being exposed to UV light and so on, and then this area is now exposed to any kind of implants.

So, we will now use the boron.

So, boron is again a p-type impurity, and it is being lighter than arsenic.

So, the energy is being less that is being used for arsenic.

Arsenic we use about 75 kilo electron volts, and boron we are using 50 kilo electron volts, and this case now the boron does the same thing, it gets entered into this particular open area.

So, this is the open area for boron.

It will go here and settle down in this area, and this is now the p plus implant.

So, this is the similar structure that we needed for your pMOS.

We wanted a p plus, p minus, and an N kind of structure in this case, ok.

So, this is your pMOS, this is your NMOS, and we will try to remove the mask and put this particular MOSFET, the half built MOSFET into an high temperature annealing.

We will put them into a high temperature furnace.

Furnace could be about 1000 degree Celsius, and we will expose this to high temperature annealing.

So, because it is very similar to what we did for the P wells and N wells like the drive-in diffusion.

It will force or push this particular N plus and the p plus impurities and p plus and the N plus impurities into the even more into the p type substrate.

So, this is how your source and drain will look like.

It will penetrate more into the N well, ok.

The N plus will penetrate more into the p well, and this is your source and drain for your pMOS.

This becomes a source and drain for your NMOS, and this process is referred to as high temperature annealing.

So, what we will do is we will stop at this point.

The next video onwards we have to look into the steps to form the contacts and interconnects.

So, we have your, we have the gate, we have source and drain, and similarly for your NMOS also we have the same.

**Local interconnect Formation**

Hello, everyone.

Let's continue our discussion on building the contacts.

So contacts are really very important because that's only that's the only thing that is accessible for a user and through which you can control the electrical characteristics of your PMOS and NMOS.

So that's the only thing that is available for us.

So let's try to build up contacts.

So the first step is to remove the thin screen oxide that we had deposited.

So remember sometime back in the previous video, we had deposited the screen oxide just to avoid the channeling effect.

So it's time to remove them and open up the contacts, open up the source, drain and gate agent for building up the contacts.

So let's let's etch that off.

So we will remove this using a hydrofluoric acid solution.

So it's a solution that reacts with your oxide and etches it off isotropically.

So let's try to do that.

And now we have etched off your surface.

Now this surface, the gate and the source and drain, this gate and source and drain are now open for any type of contact building that we want to do.

So let's start with the local interconnect.

So there are various steps to do it.

The first step is to deposit a titanium.

So titanium is a metal which has got very high connectivity or very high, very low resistivity.

So we'll use titanium as a metal for now.

And the way we are going to deposit titanium is through sputtering.

Now let me try to give you a brief idea on what sputtering is.

So sputtering is basically you take a titanium metal over here.

You place a titanium metal and then you hit the titanium metal with some argon gases.

So what happens when you do that, the titanium metals, the particles of the titanium metals will get sputtered out, will get removed from the titanium and it will get deposited onto the substrate.

And these are all the controlled experiments that we do.

So titanium metals, the metals which are there on the surface of titanium, they are being hit by the argon gas and they emit out titanium metals and they get deposited on the substrate.

So this process is called sputtering.

We'll talk details about sputtering in a separate course.

So let's understand the basic behavior, how does physical sputtering behaves.

So this is the way we'll deposit titanium.

So the titanium will be deposited all over the structure.

So it will look something like this.

So in this way, the titanium has got deposited all over them.

We know that the titanium is a metal.

The next step is to create a reaction or create a contact between this titanium metal that we have deposited and the source drain and the gate engines.

So the best way to do is to put it in or heat it in a nitrogen ambient and the result for that and the heating is being done at a very high temperature.

It's about 650 to 700 degrees Celsius.

So when we do this with the nitrogen as an ambient, there's some chemical reaction happens and the result is you will get some titanium silicon dioxide, TiSi2 material, which is this one.

These are low resistant contacts.

So now if you see the poly contacts and the contacts which is present over here, they're all being made now.

So let me take you to the previous figure.

Previously it was like this.

So now this titanium over here, the titanium over here, over here, whichever are the ones which are in contact with the silicon, they react with silicon to form titanium silicon dioxide.

So TiSi2 is a low resistant metal which can be used for your local interconnects.

And not only that, there's another chemical reaction that happens between titanium and the nitride and the nitrogen ambient, which is Ti, and it's called titanium nitride and that is used only for local communication because of the kind of resistivity that it has.

It can be used only for your local communication.

So now you have two kinds of metal contacts over here.

You have this contacts which is titanium Si2, TiSi2 over here, over here, over here.

And also for your NMOS you have three titanium Si2.

And there's another TiN available which is on, which is deposited on top of each and every structure.

So the next step is the deciding factor of which all are the contacts that you want to bring to the next level.

Because there are some contacts which can be connected internally.

So for example, if you look into a CMOS structure, the drain part of your CMOS, the drain of your NMOS and the PMOS are tied to the output.

So that can be locally connected.

But the output can be taken off through some higher level connections.

Similarly, you have to decide on which kind of connections can be done locally.

And those connections are done using the titanium TiSi2 that you have over here.

And for the connections that you want to bring at the top, you can use the next level of metal which is TiN.

This level of contact can be used to bring up the bottom connections right to the top.

So let's try to do that.

What we'll do, we'll use the same process.

We have our mask 11 now.

So we have done till mask 10 and now there's a mask 11 and there are four more, five more masks to go.

So we have the same mask, the same process.

We deposit resist, we spin resist, basically we spin the resist on the structure, we expose it to UV light, chemical reaction happens and the resist gets washed away.

So this is the structure that we're actually looking for.

So now what we want is we want this contact to be coming up, which is, for example, if you consider this as the drain of your PMOS and NMOS, we want the source to be coming out, we want this source to be coming out and we want the gate to be, we want one of the drains to be coming out for the output.

Let's say this is just an example.

We can have various combinations of that.

So let's say we want this three section of the contact to be coming out of your chip.

So what we do is we create a mask 11.

Now we remove the mask from the structure and we have the photoresist present over here.

Now we'll etch off the extra titanium nitrate we have.

And that is done through RCA cleaning.

So RCA is nothing but a solution.

It consists of various combinations of various parts of chemicals.

So for example, in this case, RCA cleaning will consist of deionized water, five parts of deionized water.

So deionized water is a more clear form of water.

So we have ammonium hydroxide, we'll use one part of ammonium hydroxide and we will use one part of hydrogen peroxide.

So this, by this combination, we can create a solution and that solution is used for etching of titanium nitrate.

And that process is referred to as RCA cleaning.

We'll cover all of them in the device physics course.

So let's wait for it.

Let's not get into the chemical formulas for each and every one of them.

So we'll cover them in the device physics course.

So the titanium nitrate is being etched off using RCA cleaning, using the solution that we just explained.

And this is how it will look like.

So now you have this contacts, you have this one and you have this one available to be taken to the top level or to be taken out of the chip.

So that's how we'll be going to, that's how we'll be doing it to the higher metal levels.

So the first level of contact has been done.

Okay.

And this is the, this is the TI and local interconnects that we can use to connect, to connect each other locally.

And also we can bring this up to the top.

**Higher level metal formation**Hello everyone, let us look into the higher level metal formation.

So, the good news is if you have looked into the steps till now and if you have understood it well, these steps will remain very similar, very familiar to the previous steps.

So, there is nothing new that we are going to introduce over here.

Only the reasoning why we are depositing certain layers and everything that will be the theoretical part is the newer one, but other than that the steps remains the same, ok.

So, let us move forward.

So, the first step to notice over here is the non-planar surface topography.

So, the surface topography that you see over here that is completely non-planar and it is not very good idea to use the metal interconnects for or deposit the metal interconnect from this layer, ok.

There are some problems with respect to metal discontinuities and so on.

So, we will talk more about that when we go there, but the bottom line is you cannot use this surface topography for your metal interconnects.

What do we do about it?

So, the best step is to planarize this particular surface.

So, how do we do it?

The next step is to deposit a thick layer of SiO2, ok.

And there is something special about this particular SiO2 that is being deposited over here.

It is being doped by phosphorus or boron.

Now, there are multiple reasons to do this.

One of the reasons to dope it with phosphorus is to phosphorus acts as a protection layer for the sodium ions.

So, there are some mobile sodium ions that flows inside this SiO2.

So, we need to take care of that.

So, phosphorus acts as a barrier or protection against them and we will talk more about that when we enter into VLSI technology classes.

So, let us talk about that over here.

The next step or there is another reason why we use boron.

So, boron is basically used to reduce the temperature.

Now, this wafer will be again exposed to certain higher temperature or certain processes which will need high temperature.

So, boron actually helps us to reduce the temperature of this particular surface.

So, there are other physics related reasons for this.

We will talk about that when we go to that course, ok.

So, you still this, if you still look into the surface topography, you still see this hills and valleys present over here.

So, the next step is to polish this and get a flat surface and the step is referred to as chemical mechanical polishing, a CMP technique.

This is primarily used to planarize the wafer.

So, basically, it is nothing but a simple technique where you take the structure or take the substrate, you topple it upside down and expose it to some machine which actually planarizes the surface, ok.

That is a pretty simple step to say, but it is a very mechanical, it is a mechanically very very complex structure.

So, it is a mechanically very complex thing to do.

So, that is the step that is used to planarize the surface and now we are good to go to for the higher level metal interconnection.

So, what do we do now?

The next step is to create the contact holes.

So, for example, if we decide that we want to create some contact holes over here, let us say somewhere over here and somewhere over here, we do so by using by drilling some contact holes.

So, drilling the contact holes is something like you create some, you take a pin and create a contact hole over here, ok, in a broader language.

But for a CMOS fabrication process, there is a various step to do it and we use the same photolithography step to do the contact holes, ok.

So, this is how do we do it.

So, the steps remain the same.

Now, we are at mass dwell.

We will spin this particular structure with photoresist and expose off certain areas with the UV light.

We use mass dwell to protect the remaining layers.

So, mass, this is a mass dwell, we are about to reach mass 16.

So, the next step is to remove the mass and etch off the SiO2 which is for which we want to drill the contact holes.

Now, we decide to drill contact holes over here, over here and as well over here, ok.

So, this is how you do it.

So, when I say we drill the contact layers, we drill the contact holes over here.

So, we are actually making this area accessible to the top layers, ok.

That is what we exactly do.

So, the next step is to remove the photoresist and now deposit a small, a thin layer of titanium nitride.

So, titanium nitride, so the question is why titanium nitride?

TiN acts as a very good adhesion layer for the SiO2, ok.

That is one of the reason and secondly it acts as a very good barrier layer between the bottom levels or the bottom interconnects and the top interconnects.

The next step is to deposit a blanket tungsten layer over here and this will help us to create a very good contact from the bottom to the top.

The next step, the next step immediately after this one is CMP which is chemical mechanical polishing.

We will remove all the extra tungsten that we have over here and planarize the surface, ok.

So, this is how your surface will look like now.

And the further steps are again pretty simple.

You need to, you need to use some, use a mask and photolithographic techniques to bring this contact holes outside your chip.

So, let us see how do we do it, ok.

So, the next step is to again first of all take a, now we have an aluminum, now we have a blanket tungsten layer over here which is the contact holes.

Now, this contact holes has to be connected to a higher metal level, higher metal layer.

So, we use aluminum as a metal layer, we deposit an aluminum, aluminum layer over here and then we patent this aluminum layer to take it to the top level.

So, we use again mask 13.

Mask 13, the process again still remains the same.

Mask 13 is used to block the, block certain areas of the aluminum where you want to take out the contact holes right to the top and the remaining areas of your aluminum is being etched out, ok.

So, this is how we do it.

So, now, so, now you have this aluminum is in contact with the bottom layer over here.

You have this aluminum which is in contact with this particular layer.

You have this aluminum which is in contact with this layer.

So, the contact is being produced or the interconnect sizes is being increasing as we go to the higher metal level layers.

So, the next step is to remove the resist and now you have your second level of metal, ok.

So, you have your local interconnect, you have the first level or the zero level of metal which is local interconnect, you have your first level of metal which is the aluminum interconnects.

Now, the next step is to again take this particular metal levels to the higher level metal, ok.

Let us see how do we do it.

The steps remains pretty simple.

You deposit oxide, you deposit SiO2, the same thing that we did for this layer, you deposit SiO2 and you again do a chemical mechanical polishing over here.

And the next step is again to pattern the same thing.

I have not shown the patterning or the photolithographic techniques over here, but we know that there is a mask 14, we will use another mask which is mask 14 to drill the contact holes.

The steps remains pretty much the same.

You use the mask 14 to drill the contact holes in a similar way that you did for here, for here and for here.

So, now, when you use this you again do the same thing.

You deposit a TiN, TiN, a small a thin TiN layer and the reason for depositing TiN remains the same.

It acts as an adhesion layer for the SiO2 and also acts as a barrier between the lower metal layers and the higher metal layers.

And the next step is to again deposit tungsten, the same thing that we did over here, over here and over here.

It is basically now we are making the contact holes.

So, the steps again from the bottom till the top remains the same.

You have your first level of interconnects, you drill contact holes.

You have your second level of interconnects, you again drill your contact holes.

You have your third metal of, you have a third layer of interconnects that we are going to deposit now.

It will look something like this, ok.

And you will use mask 15 now because this was mask 14, now this is the mask 15 to make the third level of interconnects over here and which is the, if you see the thickness of this one and this one, this one is a bit thicker than the bottom layers and that is what it exactly happened.

So, if you look into the outside pins of your chip, it looks much thicker in size and this is the reason for that.

As you, as and when you go from bottom to top, the thickness of your metal layer or thickness of your interconnects is actually increased.

And once we have, once we have done that, the higher steps again remains the same, you deposit again a layer of Si3N4 or SiO2.

So, it could be anything basically.

Si3N4 is the one which is used to protect the chip.

So, assuming this is the last level of your chip, this is where your interconnects come out.

We will use the, we will use the Si3N4 which is a stronger dielectric to protect the, protect the bottom layer from the top.

So, this will actually protect your chip.

This actually acts as your protection layer for your, for your entire chip, ok.

And finally, we will use mask 16 to drill the contact holes and bring up this particular contact outside your chip.

And now, this is how your CMOS, 16 mask process will look like.

So, you have your contacts, you have your source, you have your gate over here, you have your drain over here, drain is connected right from this point and reaches this till this point.

You have your source over here, you have your gate over here, you have, again you have your drain over here.

So, this was your 16 mask process and this finishes the, the fabrication steps that we wanted to do now.

**SKY 130 Basic Layers Layout and LEF using Inverter.**

to the input.

So before we begin anything, let us understand the layers.

So on the right side, you see all the color palettes.

These are basically the layers from which you can choose.

So in Skyward to 130, the first layer is the local interconnect layer.

That is short for local eye.

So if you see, this is the color.

This is the local eye.

This is the first layer.

Above that, it's metal 1.

It's a light purple color, metal 1.

Metal 2 has a pinkish cross.

This is metal 2.

And then there is this anvil.

Anvil is the solid slanting dash lens.

So let's, before we jump into any conclusion and say, like, assume that it's an inverter, let's see if our devices are present or not.

So we know that when an n-diffusion region, this green is n-diffusion.

How do I see the green as n-diffusion?

You just go to the color palette, move your cursor over the color, and you can see on the top, n-diffusion.

Just see on the right top corner, n-diffusion.

And similarly, there's brown color.

That is p-diffusion.

So we know that when an n-diffusion, when a poly causes an n-diffusion, it's an nMOS.

And similarly, when a poly causes the p-diffusion, it's a pMOS.

It's well and good in theory.

If it's true or not.

So what I will do, just as we had selected the pins, if you remember, during float length, we will move the mouse over this intersection area, like the area where the poly and the n-diffusion is crossing.

We will keep our cursor over there, and we will press S. So S will select only that area.

And then we will move to this tkcon window, and we will do a what.

So the what will tell is, what it's basically telling us, what is the highlighted portion.

And what it says, you can see, it's an nMOS.

So the definition holds true.

That is, when the polysegon crosses the n-diffusion, it's an nMOS.

That is correct.

Similarly, for pMOS, we move the mouse over here.

Don't click on anything.

Mouse over here.

And do an S. So again, we will do a what.

That's a pMOS.

So that's good.

That is big.

And so we know that the nMOS and pMOS, the gates are connected.

That is good.

Next, we will see whether the drain of the pMOS and the drain of the nMOS are connected.

So in magic, what you do, if you want to see if there are connections between two different parts of the circuit, all you have to do is press S three times.

Because when you press it once, it is in the place where your cursor is that gets selected.

When you press again twice, that entire thing to which the highlighted area is connected to gets selected.

So as you can see, the white is output.

And it is now connected to both the drain of the pMOS as well as the drain of the nMOS.

So that connectivity is fine.

Now, according to CMOS definition, the drain, the source of the pMOS, should be connected to the VDD.

And source of the nMOS needs to be connected to the ground.

So we'll check for pMOS.

The white is output.

And it is now connected to both the drain of the pMOS as well as the drain of the nMOS.

So that connectivity is fine.

Now, according to CMOS definition, the drain, the source of the pMOS, should be connected to the VDD.

And source of the nMOS needs to be connected to the ground.

So we'll check for pMOS first.

We'll do S. S, that selects the contact.

So keeping your cursor over a particular bit selects that object.

So earlier, we had kept our cursor over Y.

And we had pressed S. So that Y was basically a port.

So when you press S again, then that thing was selected.

You can see, then that thing was selected.

Similarly, we'll keep it over this contact.

The contact was selected.

We'll press S again.

The diffusion got selected.

And this.

So you can see, this is the connectivity of the source of the pMOS to the VDDM.

And similarly, when you go to nMOS, we'll do an S. S again.

So you can see, this entire thing is one connection.

So how to build an inverter from scratch is all described in this Git page, GitHub link.

So this gives a basic idea about OpenLAN also, what OpenLAN is all about, what are the tools that is used in OpenLAN for different functionalities.

There's an introduction called, or something called, LEP.

LEP is basically a library exchange format.

So this figure is actually an indication of what is the difference between a layout and a LEP.

What we saw here is a layout.

And this is basically how a LEP looks like.

So LEP is basically, it will just have all the meta layers.

It won't have all the information about the logic part.

You can see in this figure, there is no information about the logic part.

That is basically, because you don't need, for a PNR placement of any cell, any macro, you don't need information on the logic.

Because all you need to know is where are the PR boundaries.

If I want to place a cell, I just need to know where are the pins of my cell, and where are the PR boundaries, where is the VDDM, where is the ground.

All of that information is needed to place a cell.

So that is LEP.

And LEP serves also the functionality of protecting the IP.

So when you buy an IP from the vendor, so it is their product.

They won't want their logic to be visible to the void.

So they will, using LEP, they protect it.

**Creation of Cell Layout and Extract Spice Netlist**

And this is the introduction to how do you create a standard cell in MagicWindow.

So this is step-by-step creation.

As you can see in this figure, that is this empty MagicWindow.

And first you create a fixed B box with a 0, 0.

The first 0 is a lower left.

So if you see here, you can see some known glitches.

Width and height is fine.

The width and height of the box which you created.

And the LLX stands for lower left X value.

That means, if you can see my mouse, this is the lower left X value.

LLY stands for lower left Y value.

URX is upper right X value.

URY is upper right Y value.

So to define a rectangle, that is the only coordinate values you need to know.

So this is the fixed B box.

We are creating a rectangle with lower left corner.

And the lower left corner is 0, 0.

Upper right corner coordinate values are 138 and 272.

So in microns, it is 1.38 and 2.72.

So that is what it got created.

When you do a property, this command, this is what gets created.

So when you select on this box, just as I told you before, right-click, left-click.

And then when you type box, that will give you the dimensions of the box.

You can see.

You can try this.

And following that, we build CMOS step-by-step.

So first, we do, since I told you the first layer is a local eye.

So we lay our ground and the UVD pins on local eye.

And since we will be taking connections on Metal 1.

So the standard cell rows are in Metal 1.

Power and the ground lines are in Metal 1.

I will show you where do we have the definition there.

And why do we have to have the standard cell power and ground in Metal 1?

Where do we have that definition?

So this is the, and the dashed line, as I told you before, that's the end bit.

So you can see this is the end bit.

This was the first thing we drew, the fixed B box.

The purple thing you see hidden behind the dashed line is the Metal 1.

The bluish line, light blue line is local eye.

And there is one more layer, n-substrative, that is actually behind this local eye.

That is one of the design rules you need to fulfill if you want to have a contact.

So, now see, we just have made the layers.

We have not made connectivity.

So this square bluish checked contact is the contact between the local, the first layer, that is the local eye, local interconnect layer, and the Metal 1.

So that is the contact.

That is called LI-con.

So we will just go to the layout window and see where that contact is.

So yeah, this is what I was talking about, LI-con.

LI-con is the connectivity between the local eye layer and the Metal 1.

Before we make that contact, we have to make a contact between the, for example, the PMOS.

The PMOS is actually inside a well.

The dashed line is inside the well.

Then it's to have to be a contact between the n-well and the LI-con.

That is this.

As you can see, that is more clear in the image.

You can see here.

This cross line, that is the n-substrate contact.

That is the first contact.

That is the contact between the n-well and the local eye.

So that first layer connectivity is done.

And then you need to have contact between local eye and Metal 1.

So this is visible.

This checkered line, this checkered box is LI-con.

I have marked it here.

Similarly, for NMOS.

Since NMOS, there is no well, so it's directly substrate.

So you need to have first contact between the P-substrate and the local eye layer.

So that is this.

You can see here.

It's P-substrate contact.

And then you need to have a contact between the local eye layer and the Metal 1.

So that is this LI-con.

Right?

Hope that's clear.

It's logical, right?

So for PMOS, there is n-well.

Above that n-well, there is local eye layer.

Above local eye, there is Metal 1.

So to connect these layers, first you need to connect the n-well and the local eye.

For that, you need n-substrate contact.

Yeah, n-substrate contact.

N-substrate contact will ensure connectivity between n-well and the local eye.

Now you need to have a connectivity between local eye and Metal 1.

For that, LI-con is used.

Similarly, for the N-site, for the ground rays.

So now that you have fixed your power and the ground rays, all you need to worry about is the logic.

So you need to follow the DRC.

So magic is one of the impact of DRC.

As in when you have a DRC, it will show up in the DRC.

Right now, the DRC is zero.

So for example, I remove this layer.

Okay, this is fine.

This won't cause any problem.

I'll do one thing.

I'll just try to remove this.

Right now, as you can see, the DRC has changed from zero to seven.

And this dashed line, this dotted line, the white dotted line is the error.

To know the error, you go to this DRC tab.

And then you click DRC, find next error.

And the tool will zoom into where the error is.

And if you want to know what actually is the error, once the tool highlights this error, it will be written in there.

If you can't remember.

So these are the DRCs which you are seeing.

This constraint, the physical DRC, is not getting violated.

So this is important from the manufacturing point of view.

If the DRCs are violated, it cannot be manufactured.

So you have to ensure your final design needs to be DRC-clean.

So in this case, we can see, we can troubleshoot this.

So first, we will ensure that this entire area is n-well.

So we will select that area.

And this is the all-well peeling.

So this will basically be n-well.

So if you want to select the layer, move the cursor over that layer and press your middle mouse button.

So the DRCs have been reduced from 7 to 5.

So this is what I was talking about.

It's an interactive DRC.

And then you keep doing this DRC's radius.

So we have covered this area with n-well.

Second n-well, over n-well we have.

So there is no L icon.

So this is purely metal one.

So we will select this area.

And then we will use metal one.

Metal one is here.

So our DRCs are back to zero.

So I select it, I will centralize it.

So this is my inverter.

So this is all layers.

How do we know what is the logical function of this inverter?

For that, we would first extract the SPICE.

And post that, we will do simulations on SPICE, in ng-spice.

So to extract it on SPICE, so we will open the ticket converter.

We will right now see where we are.

We are in this directory.

This is what we have told.

Open link, we have just added self-design.

So the first step is to create an EXT file, extraction file.

So the command is extract all.

So this has said that extracting our design into sky1.inv.ext.

So let's go to that location.

Where was it?

Yeah, here.

And go to ls-ltr.

Let's see if something new has been created or not.

Yes, it has. sky1.inv.ext.

So next what we'll do is we will use this EXT file.

So let's do that.

So let's do that.

So let's do that.

So let's do that.

So let's do that.

So let's do that.

So let's do that.

So let's do that. to create a SPICE file to be used with our ng-spice2.

So for that, what you need to do is you do x2 SPICE, extract to SPICE.

And I think the command is extract to SPICE.

Yeah.

That's x2 SPICE, that is the command.

So we choose a cthresh 0 rthresh 0.

So doing this will extract all the parasitic capacities also.

So these are actually as this will have parasitic capacities.

So we'll do this.

So this has not created anything new.

So you can see it's the ext file is there.

I think it has been created.

Now we'll do x2 SPICE.

That means the ext file to SPICE.

Let's see.

Ext to SPICE.

Click Finished.

So let's see if anything new has been created or not.

Yeah, it has.

To SPICE.

Let's see.

Ext to SPICE.

Click Finished.

So let's see if anything new has been created or not.

Yeah, it has.

SPICE file has been created.

Let's clear this window.

And let's see what's inside the SPICE file.

OK. So not much.

So before we begin our simulation, let's first understand what the SPICE is.

Let's clear this window.

And let's see what's inside the SPICE file.

OK. So not much.

So before we begin our simulation, let's first understand what the SPICE deck contains.

**Steps to create final spice deck using sky130A**

So, as you can see, we first have this PMOS, this 10.1 calculator PMOS, how do I know it is a PMOS?

You can see the model name, PSHOT.

Similarly, this is NMOS.

For PMOS, this Y is the drain, the gate, followed by the source and the substrate.

Similarly, this is drain, gate, source, substrate.

So, let's understand the netlist first.

I am going to draw on the screen.

My drawing might be a little horrible, but I hope I can get the message conveyed.

So, this is our PMOS.

It has got source, drain, substrate and gate.

So, PMOS, you can see the PMOS 10.1 thousand, its drain is connected to Y.

So, it is connected to this node.

It is important to realize here that the attributes are just nodes.

It is not a value, it is not a voltage.

It is connected to node Y.

The gate is connected to, you can see the gate is connected to node A. The source and substrate are connected to node B, BWR.

So, this is what the first line says.

We are done here.

The second line says that it is NMOS.

It says that its drain is connected to Y.

That means this connection is complete.

And it says that its gate is connected to A. That means these two are connected.

Then it is saying that its source and substrate is connected to a node.

So, it is a node whose name is VGAD.

Node's name is VGAD, not VSD.

Now, what do we need to do for this transit analysis?

First, we would want this V ground to be connected to a ground, let's say VSS.

Then we would want a supply voltage that needs to be connected from VBWR to ground.

So, we will create a node.

We will create an extra node, say V0 and we will create a VDD at a value of 3.3V.

Now, we have to see that we give a certain voltage, a false voltage.

A false voltage that needs to be connected between A input and the V ground on 0.

It is a variable name.

So, we need to do this and this connection.

We have to define this connection.

The directions which have been assigned are these double decodes.

Yeah, that's all.

Double decodes.

A and Y.

We have to define the single decodes.

So, we will do something.

Let's see if I can.

I will just go over the thing.

Here is the screen.

Okay.

Insert.

Then.

Right.

So, this is my screen again.

You can see.

First things first.

We need to ensure that the scaling is proper.

Right now, the scaling is any dimension into 10,000 microns.

That is not what we need.

We need the values to be in the grid value which was assigned to the layout.

Since, if you recall, in the spice we had extracted from the layout.

So, the minimum value of layout window is measured in the grid here.

So, this is the measurement.

Any measurement that needs to be taken in the spice side.

So, we will see this box one more.

Let's see the dimension of the box.

The dimension of the box is 0.01.

That's why we edit the scale to 0.01.

Any measurements will be with respect to this.

So, the width will be 0.01 and it is 0.37 microns.

So, this is done.

Similarly, now we have to compute the T-MOS and N-MOS Lipschitz.

So, that's what we do.

And, if you recall correctly, it is inside the Lipschitz folder.

We have these two files.

So, to specify that, we have to go to the Lipschitz folder.

And, .pshot.

Similarly, we do the same thing for nshot. nshot.p.

So, these two files are included.

We will comment on this once.

Because, we are trying to input the controls also.

So, we will put that command.

This is fine.

Now, we need to make the definition for the supply voltage.

So, we will create a variable VDD, which is connected between VWR and the node 0, which has a value of 3.3 volts.

So, this is the supply voltage.

So, we will create a VSS.

That is the voltage connected between VGND and 0.

VGND and 0.

Both values are 0 volts.

The ground and the supply is done.

Now, we need to specify the input pulse.

We will get a value of V8.

It is connected between V8 and VGND, which is now equal to the node 0 having a value of 0.

This is how we get the pulse value.

The low value is 0 volts.

The high value is 3.3 volts.

The starting point is 0 nanosecond.

The pulse is rising.

The rise time is 0.1 nanosecond.

0.1 nanosecond is the fall time.

This 2 nanosecond is the on pulse width.

And 4 nanosecond is the time duration.

This is how we define pulse in the SPICE deck.

And post that, we will also include the type of analysis that we want to do.

So, this I have already.

This is analysis.

This is how we specify the analysis.

We do a dot run.

Dot run is what we run the analysis from 1 nanosecond to 20 nanosecond.

And then, we run it.

So, this is how we create the flow curve.

So, this is the SPICE deck.

So, we will run this in the NG SPICE.

So, before that, we will just see what is inside the model files.

So, as you can see, this model file actually starts here.

So, this is the name for the model file.

The PMOS model name.

So, you can see all the parameters that is defined under this SPICE model file.

There is one more small modification we would have to include in our SPICE deck.

As you can see here, this M1000, I have told you, is a PMOS.

How does the NG SPICE know it is a PMOS?

By using this.

But this is not the model name for PMOS.

Then, what we saw in the SPICE model file was this PSHOT underscore model dot 0.

Only then, we specify this.

We will take all the SPICE parameters from the model file and assign it to this model and do the simulation.

Similarly, for NMOS dot 0.

Now, I think our SPICE deck is ready.

To run our SPICE simulation, what you do is, NG SPICE, that you can go from the tool, and pass this as the source file.

**Introduction to Delay Tables**

And we'll enable a feature of the AND gate in such a way that it will allow, it will decide the connection of the clock to the complete clock.

So it was something like this.

We had this one pin of the AND gate which was connected to logic 1.

And whenever this particular pin, we call it an enable pin, whenever this particular pin was connected to logic 1, then and only your clock will be propagated through the AND gate to the rest of the circuit.

Else, it won't be.

If it is at logic 0, the clock will be blocked at this point, but it won't be propagated to the complete, to the rest of the circuit.

Similar thing we did for OR gate also.

OR gate can also be used as a very good gate to block the clock.

So whenever the logic, whenever the enable pin is at logic 0, the clock propagates through the OR gate to the rest of the circuit.

Else, whenever the logic, whenever the enable pin is at logic 1, the clock stops at this point.

And the rest of the circuit, the clock doesn't reach to the rest of the circuit.

So the advantage that we can get out of it is, the rest of the circuit for that amount of time, there will be no switching and short circuit power that will be consumed by the clock during that period of time.

So during that period of time, you're saving a lot of power.

You're saving a lot of switching and the dynamic switching, sorry, switching and the short circuit power during that period of time.

So that's a very big advantage that we have by using this kind of scenario, this kind of circuit.

So the question is, how do we use this in the clock page?

So for example, let's say if we have this kind of clock page.

So this is a snippet of the clock page that we have taken from the existing, from the past videos.

This has been referred from the past videos.

So we haven't changed the tree by itself.

So as to keep consistency in the explanation.

So this tree was used, this particular diagram was used for load splitting, load splitting at this particular node in the past videos.

So we can refer to those videos to get an idea of this particular buffer tree.

So this buffer, this tree has this buffer which is present at the input for the first buffer.

And this buffer was driving a load of these two buffers.

So this is what we have done.

We have split it.

In the past videos, we have split the complete load of all these four flops directly onto a single buffer.

Rather than, we have split the loads of these four buffers, of these four flops into two buffers.

And the load of these two buffers was given to this particular buffer.

So that's what we have done in the last video.

You may have a look into the previous videos.

So the gating concept that we learned just now.

So is it that we have to just swap this particular buffer to a gate and things will work for us?

That is the first question that we need to ask.

The gate, the AND gate concept that we have just learned, the clock gating tech, basically this technique is called as clock gating.

We are gating the clock to go to the rest of the circuit under certain conditions.

So that is called as clock gating technique.

So is it that we have to just swap this buffer directly to an AND gate and the rest of the things will work?

By rest of the things I mean the other quality check parameters, that is Q, quality check latency, and all those things.

Will all those things be still good or be still valid as we have derived in the past?

So we have to look into all this.

So before looking into all this, what we have to do is, we have to look into certain timing characteristics of this particular buffer, say this buffer is this AND gate.

We have to look into the timing characteristics of this buffer.

So what we will be doing is, let us try to move this circuit.

We will come back to this kind of circuit.

We will come back to what happens when we just blindly swap a buffer with an AND gate, what can move and all that.

So we will look into those things some time from now.

The assumption was that, the capacitance at node A was 60 FF and the capacitance at node B and node C was 50 FF.

So this was the assumption that we had done in the past and also we did some observations.

So the observations was that, there were two levels of buffering, that was the first observation.

So we have this level 1 buffering, we have this level 2 buffering.

So in the level 2, we have two buffers.

So that was the first observation, that this particular clock tree has two levels of buffering.

The second observation was that at every level, each node was driving the same load.

For example, at level 1, the clock buffer which is present at level 1 was driving a load of 60 FF.

So there is no comparison point over here to check for.

At level 2, this particular buffer was driving a load of 50 FF.

If you see over here, we have mentioned, it was driving a load of 50 FF and the second buffer in the same level was again driving the same load, that is 50 FF.

So the second observation is also, we are also meeting the second observation and the third observation was that, at every level, all the buffers that are present at every level, that should be identical buffer.

For example, what we mean by this is, the buffer size at level 1 is, let us say, is of size 1.

The buffer size at level 2 is of size 2.

So we have to maintain this particular observation to meet something.

Let us talk about that in some time from now.

So what we have to do is, the size of the buffer at every level, for example, if there had been another level, level 3 buffer, all the buffers should be of the same size.

It could be a size 1, size 2 or whatever.

But if it is of size 1, all the buffers at level 3 should be of size 1.

If it is size 2, all the buffers at level 3 should be of size 2.

It is something like that.

So we have looked about this in the past.

We have made a clock table for 16 clock endpoints and there we had looked into all this.

So now, let us try to understand why are we doing this particular observation.

Why do we even need this kind of observation?

What can go wrong and what cannot go wrong if we do this and if we do not maintain this?

So what happens is, this particular buffer, for example, let us say this particular buffer has got an output capacitance of 60 FF.

This particular buffer has got an output capacitance of 60 FF.

So the point is the output capacitance of a buffer for the entire clock tree is not constant.

It is not constant at 60 FF or not constant at 50 FF.

Basically, the load at the output for, let us say, the other level of the clock, if you look into one more level of the clock tree, the load at the output will be varying.

It will not be 60 or 50.

It could be something else.

It could be 70 FF.

So that is one point.

The other point is that capacitance of the load at the output load of each and every buffer in the complete clock tree of the chip is varying.

That is first point.

And also, since the load is varying, for example, in this case, the output load of this particular buffer becomes the input of this particular buffer.

So if the load is varying, the input transition is also varying.

So for all the buffers which are present in the clock tree, the input transition is also not constant.

It is also varying.

It is varying within a range of 10 picoseconds to, let us say, 100 picoseconds.

So that is the second thing.

So we have varying input transition at the input of the buffer and we have varying output load at the output, varying output load at the output of any buffer.

So the problem now here is we will have varying, we will have a variety of delays.

So how to capture that?

So what a brilliant idea engineers came up with, Bayless engineers came up with, they brought something called as an LFO.

So LFO is a two-dimensional table.

So two-dimensional table and it will, how the delay table is first of all prepared.

So what happens is, this particular buffer, for example, buffer of size 2.

So since we are talking about size 2, let us take this buffer.

So this particular buffer was taken out of this particular complete circuit.

It was taken separately.

The input of that particular, the input transition of that particular buffer was varied.

It was varied from a range of transition and the output load of that particular buffer was varied with a range of various loads.

So for example, the input load was input transition, input transition was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

So the input transition of the particular buffer was the input transition of the particular buffer.

**Delay Table Usage Part 2**

Hello everyone.

So let us try to continue from where we left at the last step.

So the next job that we have over here is to calculate the delay for the buffer of size 2.

And then eventually we need to calculate the latency from clock mode to this clock end point.

So that's what we have to do.

And eventually we will also calculate the skew between the clock end points.

So let us try to move on.

So now what we have over here is at the output node you have a capacitance of 60 femtofarads.

So there will be some transition, there will be again some transition and input transition which will be present at this port and at this port.

And that will be the same, that will be the common, because the node A is common to these two input ports.

So let us try to assume that transition is now 60 picoseconds.

So by the way, output transition of a cell is also a function of input transition and the output capacitance.

So basically the 60 picoseconds transition came from, itself is a function, 60 picosecond transition you are seeing at the output of this buffer.

So this 60 picosecond transition is also a function of input load, input transition and output load.

So we are not looking into that in detail in this video.

There will be a separate step to look into this thing also.

So there is a separate delay table for transition also.

Just as we have a delay table for a C-Buff, we will be having a separate transition table for the C-Buff also, for this particular C-Buff.

So let us not get into that detail in this video.

Let us try to assume that the output capacitance after getting, output capacitance, sorry, output transition of this particular buffer is after getting, after referring to the transition table, we get the value to be at 60 picoseconds.

So it is very straight and very simple.

So let us keep it very simple for now.

So the input transition of these two buffers is 60 picoseconds and the output load that these two buffers are driving is 50 femtofarads each.

So the output load, the output load of B is having a load of 50 femtofarads and output load C is again having a load of 50 femtofarads.

And we have been taking care of all this in the tree itself.

So let us try to look into that part.

So with a transition of 60 picoseconds, the portion of the delay table, we will be going back to the delay table for C-Buff of size 2 and looking which rows we should be looking at.

So it will be this one.

So 60 picoseconds.

So all the delays with respect to input transition of 60 picoseconds are located in this row.

Okay.

Now what we have to do is we have to take the node, output node capacitance, which is 50 femtofarads, and try to locate 50 femtofarads, try to locate the column of 50 femtofarads, which is here.

So now the intersection point between 50 femtofarads and 60 picoseconds is nothing but Y50.

And this Y50 is not only valid for this, this buffer, it's also valid for this buffer also.

And the reason is the input transition is the same, the buffer type is the same, and the output load that this node is driving and that this node is driving is again same.

It's 50 femtofarads.

So the delay value of Y50 is valid for both the sizes of buffers, for this one and this one as well.

So what we'll do is we'll take, so now we know the delay of this particular buffer is Y50.

So let's try to print it out over here.

Okay.

So the delay is Y50 for this buffer and the delay is Y50 for this buffer.

Now, when we try to calculate the latency from this point till this point, let us try to ignore the wireless for now.

Let us try to focus only on the cell delays.

So when we try to calculate the delay from this point till this point, the delay is X9 dash plus Y50 at this point.

The delay over here is again it is X9 dash plus Y50.

The latency at this point is again X9 dash plus Y50.

And the latency at this point is again X9 dash plus Y50.

So that brings us to a very, very, that brings us to a conclusion that skew at this point and this point is zero.

The skew at this point and this point is zero.

That's basically the skew at any point, if you consider any point of the, at any point of time, the skew is zero.

And the reason is this, and now we can go back to the observations that we did, that at every level, each node should be driving the same load.

So what if each, at every level, each node was driving a different load?

What could have gone wrong?

So for example, the buffer two was driving a load of, let's say, now it's at 50 femtofarads.

And let's say the buffer, this particular buffer at node C was driving a load of say 90 femtofarads.

So in that case, what would have happened is the input transition is same, which was like 60 picoseconds, but the output, but the output load, since it is varying, it is, in one case it is 50 femtofarads, in the other case it is 90 femtofarads.

So the delay values that will be taken for this buffer.

So for this buffer, it will be taking as Y15.

For this buffer, it will be taking as Y17.

Now when you calculate the delay, let us say from this point to this point, it will be X9-plus-Y15 in this case, and it would have been X9-plus-Y17 in this case, because of two different, because of this node is driving two different loads.

So in that case, the skew value will, it will have a non-zero skew value over here.

So the skew value will be like, it will be a difference of Y15 minus Y17.

So you will be seeing a different delay number for this clock endpoint.

You will be seeing a different delay number for this clock endpoint, and hence the total skew between the clock endpoints would have been non-zero.

That's why it's always preferred to have, at each, at every level, each node should be driving the same load.

That was the one observation that we have justified over here.

The second observation that we did was identical buffer at the same level.

So, so let us try to take one more case over here.

So let us try to take a, let's go back to the same slide, this one.

So for example, now this buffer 2 has, this is buffer 2, and let's say this was not buffer 2, but this was buffer 1.

So this buffer, the same type of a size 1 was present over here.

So in that case, what, what would have happened?

This particular, the delay of this particular buffer is based on the input transition of 60 picoseconds and output transition of 50 femtofarads.

So, so that, that became, the number came to be Y15.

In this case, again, the input transition and the output load was the same, but this buffer was of type 1.

It was of type 1.

So if you actually look into the delay for type 1 and with the same environment, like 60 picoseconds and 50 femtofarads, the delay of this particular buffer would have been X15.

So if you compare X15 and Y15, they are actually two different numbers.

So being these two as two different numbers, you will be facing a, the latency from this point to this clock endpoint, in this case would have been X9-plus-Y15, and in this part, it would have been X9-plus-X15.

So that would have resulted into a non-zero skew.

So this is, this looks like to be a very small thing to be, to be noticed, but it's a very important thing because as you propagate further down the levels, this will create, this, this will actually create a huge mess, and at the end, clearing, clearing, clearing this part, clearing these two observations in the entire clock train, it becomes a very difficult job.

So these are the things that need to be taken care of at a very early stage of building the clock train.

So, so now, so now it's very clear from this delay table concept and the concept of the, and the concept of applying the delay table as timing models for this buffer set, it's very clear now that how having a different buffer at the same level would create a problem.

Also, how each buffer at each level driving a different load will create a problem.

So for this, for this small clock train, it would create a small non-zero skew, but as you try to propagate this complete, complete skew numbers down the complete chain of, let's say, millions of locks clock train, in that case, this difference in the, difference in the delay numbers at the same level will be, will be propagated, will be propagated and it will eventually come up to a very large number.

And that might lead to other problems which are, which will be static timing analysis problems, there will be setup time problem, there will be hold time problem, which we will be talking further in the further slides.

But, so this is a very important observation and this is the thing that you need to look into at a very early stage of the clock train.

Okay?

So what we'll do is, since this part of the concept is now, we have covered, we have covered to a greater extent in this part of the concept, now we have to move into the, the main part of the concept, which was power-aware clock train synthesis.

So in that case, what we have to consider is, we have to consider this, for example, let's say, for a time period, for a certain time period, this particular thing, these two blocks are, are not active.

Or, or the other way round, we can say, these are active only under certain conditions.

So these are special functionality of the chip, this section of the chip is, has got some special, special functionality and this part of the chip will turn on only under certain conditions.

In this case, what we can do is, we need not propagate the clock, this complete clock train, we can actually stop the clock over here.

So that is how we can achieve the power-aware clock train synthesis.

So let's try to look into, into, into a more deep about this particular problem in the next video.

 Setup timing analysis and introduction to flip-flop setup time

And then we'll do a timing analysis with the ideal clocks first to understand what are the basic structures and what are the basic parameters that we need to get used to.

And then we'll be introducing the real clocks and then doing the timing analysis again.

OK?

So let's begin our timing analysis with ideal clocks.

So it's also single clock.

So there are scenarios where your launch flop is triggered by some clock and your capture flop is triggered by some other clock.

So let's now focus on a very basic and simple scenario of a single clock.

So this is the specifications we have.

The clock frequency is being given to us as 1 gigahertz.

So the clock period, which is inverse of the clock frequency is 1 nanosecond.

So we have this launch flop, which is sitting over here.

We have this capture flop, which is sitting over here.

And we have this combinational logic, which is between the launch and the capture flop.

And we have the clock network, which is in this case, it's the ideal clock network.

But ideal clock network, we say, we say that it's the clock that is not yet built.

When the clock is built, you'll see a lot of buffers in this part.

But right now, there is no, the clock is not built yet.

So it's an ideal scenario.

OK?

And this is a typical scenario for identifying for a set of timing analysis and whole timing analysis.

This scenario is a very typical scenario.

So let's move on.

For example, we'll take, this is the clock period of 1 nanosecond.

This is your 0th edge, and this is your 1 nanosecond edge.

OK?

And this is the clock period from 0 to T. OK?

So in a basic set of timing analysis, what happens is, you send this particular edge to the launch flop.

OK?

And then the other edge to the capture flop.

But let's try to create it in a more graphical way.

And this particular edge, at this particular time instant, this particular edge, it reaches the launch flop clock edge.

So let's put an arrow over here.

OK?

So this arrow on the time axis says that on the 0th clock period, on the 0th time, there is one clock edge which reaches to the launch flop.

OK?

And on the T, the time period, which is at 1 nanosecond, at T equal to 1 nanosecond, the second edge, the next edge, just this one, the second edge reaches the capture flop.

And let's put the arrow here.

So whatever analysis we have to do, we have to do between 0 and T. That is in this particular area.

OK?

So with that said, let's assume that the combinational delay between the combinational logic has got a delay of theta.

OK?

And this set of timing analysis says that if for this particular combinational logic to work, or for this particular circuit to work, the combinational delay, which is right from launch flop, the internal delay of the launch flop to the combinational delay over here, that should be less than the time period.

And it's pretty obvious.

Because if the combinational delay, which is present over here, exceeds the time period.

For example, it was T is equal to 1 nanosecond.

And if the combinational delay is 1.5 nanoseconds, for example, then your clock period has to be shifted to more right-hand side.

Your clock period will become 1.5 nanoseconds.

And your clock frequency will reduce from 1 gigahertz to 1 divided by 1.5, roughly around 700, 800 megahertz.

And the reason is, as you increase your time period, your frequency decreases.

It's an inverse ratio over here.

OK?

So that can't be allowed, because the system has been designed to work at 1 nanosecond, or at 1 gigahertz.

So any activity that is happening, or any combinational delay between the launch and the capture, should be happening within a period which is less than 1 nanosecond.

That is in a period which is less than T. OK?

So this is the first basic equation for SEPTAP-10.

It's just that the combinational delay, whatever you see over here, should be less than your clock period.

That is T. So this condition is enough to say that this system will work at 1 gigahertz.

But now we have to introduce more and more practical scenarios into this.

OK?

The first practical scenario is the flop itself.

So let's open up this particular flop.

And then we open up this particular flop.

You see some combinational circuit over here.

So this flop, the box that you see over here, it has got several MOSFETs.

It has got several logic gates inside it.

It's not just a black box.

There are a lot of gates inside it.

And there are a lot of resistances and capacitances which are present inside the capture flop.

So let's look into a more higher level thing of this particular flop.

That is a structure which consists of two MUX.

So this is the MUX 1, this is the MUX 2.

And the way a flop works, the way a flop works, it will be more clear from this particular timing graph.

So when your clock is at 0, when your clock is at 0, your D input, whatever is present at the D input, or whatever is present at the D input, that reaches from this particular point to this particular point through this particular MUX.

So there is a delay associated with this particular transit of information.

So your information moves from D to QM.

And that takes an amount of delay which is equal to the delay of this MUX 1.

Because this MUX 1 is, again, built on various resistances, MOSFETs, DMOS, NMOS, and everything.

So there is a finite delay that it takes for the D input to reach from this point to this point.

And all this happens when your clock is equal to 0.

And in this case, when your clock is equal to 0, the output is fed back to the input of the MUX.

And that is being driven.

So basically, the output is not moved.

Your output doesn't change.

But when does the change exactly happen?

When your clock switches from logic 0 to logic 1.

So when your clock switches from logic 0 to logic 1, you see a logic 1 over here.

When you say logic 1, the output of this particular MUX is now fed back to its input.

And this particular data just rotates over here.

But over here, when you spherify the logic 1, your QM, the data which is present at QM, moves from QM to Q. So again, there's a second delay that is associated over here.

So there is a MUX 2 delay that it takes to reach from this point to this point.

But the point is, why are we even looking into this?

What will this thing impact on our setup timing?

So that's exactly the point.

The point is the MUX 1 delay or the MUX 2 delay, when there's a logic 0 on the clock or logic 1 in the clock, that will actually affect or that will actually restrict your combinational delay requirement.

So initially, we have seen that the combinational delay theta was supposed to be less than D. But now, things will change.

Why things will change?

Let's see over here.

So when you see there is an edge that is coming over here.

For example, it's a capture flop clock edge.

So when the edge at D clock reaches at the clock port, the capture flop knows that it will need some finite amount of time for input that is coming at D to bring it somewhere at the center of the flop, to bring it somewhere over here, OK?

So the capture flop is aware of that.

It is aware that it will need some finite amount of time, which is MUX 1, which is a delay of MUX 1, for whatever input which comes at D to settle it properly at the center.

So as a result of that, that amount of time has to be subtracted from the complete clock period T. So initially, if the complete clock period T was available, we'll bring it over here.

Initially, the complete time period T was available for a combinational delay to settle.

Now, the time period gets reduced, because there is some amount of time before T that is needed by the capture flop for the input which is present over here to settle at the center of the flop, which is the MUX 1 delay.

So that is the MUX 1 delay.

And that MUX 1 delay is referred to as setup time.

It's referred to as setup time, which is mentioned over here.

So let's put some setup time.

So there is some finite amount of time, which is required for the D input to settle from this point to the center of the flop.

And that amount of time is referred to as setup time.

So now your clock is OK. So if the combinational delay is somewhere between at this particular point, basically somewhere before this one, so the capture flop has got enough time to bring the data which is present at this point, which is present at the D pin to the center of the flop.

It has got enough time.

It has got that is basically called as the setup time.

It's called the flop setup time.

So now your equation changes to the combinational delay requirement now comes.

Initially, it was combinational delay should be less than T. Now it should be less than T minus S. It should be less than T minus S, where S is the amount of time required by the capture flop to settle the information which is present at D somewhere at the center of the flop, which is the MUX 1 delay.

So we have brought one more practical scenario over here.

So let's take an example.

For example, if T was 1 nanosecond, and let's say it takes only 10 picosecond of delay.

Let's say we are at a very advanced load.

And the capture flop takes only 10 picoseconds.

So your combinational delay should be less than 1 nanosecond minus 0.01 nanosecond, which is 0.99 nanosecond.

So combinational delay should be less than 0.99 nanosecond.

And this will ensure that your system will definitely work at 1 gigahertz.

So this was the first level of practical scenario that we brought into the picture.

Introduction to clock jitter and uncertainty

and the clock period minus the set of time.

So this was after bringing one of the practical scenario onto the equation.

So let's try to move on and bring in more practical scenarios.

The next one that we'll be talking about is about something called a jitter.

So what happens is, this clock has been created by a phase-locked loop, so there's some clock source that is present over here.

So that clock source is expected to send the clock signals exactly zero, T, two T, and all those time periods.

But the clock circuitry that has been built over here, all the phase-locked loops, basically any clock source which is present over here, which is sitting over here, that's again based on, or made on some real circuitry.

It has got some bias, it has got some variations, it has got some positive and negative variations.

So that clock source might or might not be able to generate a clock period which provides a clock edge exactly at zero nanosecond, or which provides a clock edge exactly at T nanosecond.

So that clock source might have got its own in-built variation, and due to that, it might provide a clock edge which can come exactly at zero, or somewhere before zero, or somewhere after zero.

And similarly, the clock edge at T nanosecond, the second clock edge that the clock source sends to the circuitry, that can arrive at T nanosecond, or before T nanosecond, or somewhere after T nanosecond.

The variation will be very minimal, but there is an amount of variation, and that variation is basically because of the in-built variation of the clock generation, clock source generation, okay?

So that's basically called as a jitter, and there's a temporary variation of the clock, and as a result of that, there is a temporary variation of the clock period.

For example, if let's say that there was some variation in the clock source, or the PLL that is supplying the clock, the PLL is basically a circuitry that generates the clock period, that generates the required clock period, okay?

And because of its inherent variations inside the PLL, it sends a clock edge at, so let's say at this edge, not at zero, but at this edge, the launch block, and the next edge that the PLL sends to the capture block is at this particular edge, and not at one nanosecond.

So initially, the period flows from zero to T, and now it has shrunk from this point to this point, which is definitely less than one nanosecond.

So this temporary variation of, and this is a very less variation, but this still accounts for failure of the circuitry.

And the reason is because the combinational delay that we have analyzed the circuit for was kept as less than T minus S, but now the combinational delay has to be, the combinational delay requirement becomes even more stringent because of this kind of variations.

And those variations are captured on a term called as jitter.

Jitter is referred to as temporary variation of the clock period, okay?

And as a result of that, this has to be modeled somewhere in this particular equation only, because there's no other way to control the non-idealities of the PLL.

So to capture or to model the non-idealities of the PLL, let's try to do it over here itself, and let's try to bring this particular equation to a more real scenario.

So initially, we had the combinational delay should be less than the clock period minus the setup time.

Now let's try to bring out one more parameter to model the jitter, or to model this particular temporary variation of the clock, and let's club this variation and this variation onto a single parameter called as uncertainty, and let's bring it over here, okay?

So now your clock period, your combinational delay, which was supposed to be less than T minus S, it was supposed to be less than T minus S, and now it will be less than T minus S minus SU.

SU is the setup uncertainty because we are doing a setup timing analysis.

Things are different for the whole timing analysis because in that case, there's only a single edge that comes at the launch and the capture flop.

So this can be more relaxed.

We'll come to that in some time from now, but in the setup timing analysis, since the different edges goes to the launch and capture flop, and as a result of that, that there will be different jitter values for the launch flop and the capture flop clock edges, and as a result of that, the window looks a bit bigger over here.

And as a result of that, now the combinational delay should be less than T minus S minus the setup time and the setup uncertainty, okay?

So if you try to bring into some real numbers over here, so if you try to bring in real numbers over here, so the clock period of one nanosecond and the setup time of 10 picosecond and uncertainty of 90 picosecond, your combinational delay requirement becomes T minus S minus S U, which is one nanosecond minus 10 picosecond minus 90 picosecond, which comes to roughly 1.9 nanosecond.

So this is your new combinational delay requirement.

So with that understanding, let's try to take up these particular specifications at the top and see what, and identify the timing parts from our existing scenario.

So we'll have this, we have this clock period as one nanosecond, the setup time is 10 picosecond or 0.01 nanosecond, and the uncertainty is 90 picosecond.

So with that specifications, let's try to identify the timing part from our existing scenario, okay?

So we have to identify the timing part with single clock.

So we have this set of logic, which has the single clock, which is going flip-flop one and flip-flop two, and we have this set of logic, which is again having a single clock, going flip-flop one and flip-flop two, okay?

So, this is one set of logic, which has got ideal clocks and single clock, and this is another set of logic.

So with that, what we have to do is, we have to identify the combinational part, okay?

So the theta that we were mentioning some time back, we have to identify the theta, the value of the combinational delay over here.

So the combinational delay value will be something like this.

It will look something like this.

So the combinational delay like theta will be flip-flop one clock to Q delay, plus the estimated wire delay between flip-flop one and one, okay?

Plus the delay of one, plus the estimated wire delay between one and two, plus the delay of two, plus the estimated wire delay between two and flip-flop two, okay?

This will be the combinational delay of this particular section, and this combinational delay has to match the satisfying condition that we described some time back, okay?

Let's look at the combinational delay of this one.

So combinational delay will be flip-flop one clock to Q delay, plus, so since there is no wire, or there is very minimal wire, we can just ignore it, and say that the wire delay is almost zero.

So the second term that we have to add is the delay of this particular cell.

The next term that we have to add is delay of this particular cell.

And finally, we have to add the, so yeah, we have to add the delay of this particular cell.

So these two combinational delays, theta that we just derived, that has to be less than the clock period, minus the set-up time, minus the set-up uncertainty.

So combinational delay is expected to be less than 0.9 nanosecond, as per the specifications.

So once we assure that these will be the delays of this particular combinational cell, then we will be assured by ourselves that the placement, whatever optimization part we have done, that is good for the set of cells which have been driven by single clocks, okay?

But the next challenge comes when we try to do a similar kind of timing analysis with multiple clocks, okay?

So, what we'll do, since this is a fresh topic, and we need to start it from scratch, so we'll do a set-up timing analysis with ideal clocks and with multiple clocks.

**Clock Tree Routing and H tree Algorithm**

How do we do it?

Let's try to blindly connect them.

Without any rules, let's try to connect them.

We'll have this clock port connected to flip-flop 1 in this fashion.

So this is a physical wire.

So there's basically resistance and capacitance that comes along with this particular wire.

Let's try to connect clock 1 to this flip-flop 2 in this fashion.

Let's do the same for the blue-coloured flip-flop 1 and let's do the same for the green-coloured flip-flop 2.

So this is what we have done the connections based on the connectivity information.

Now let's try to see what is the problem with this particular clock round.

So this clock section belongs to this section of the circuit.

Let's bring it out.

Let's try to analyze what this section already does.

So if you try to measure physical distance from clock 1 to flip-flop 1, the time required to reach the clock 1 to flip-flop 1, let's say it's called T1.

And the physical distance from clock 1 to flip-flop 2 is what you see.

And the time required for the clock to reach from clock 1 to flip-flop 2 is T2.

So due to physical reasons and due to obvious reasons, the T2 is greater than T1.

Which means T2 is not equal to T1.

And the difference between the time required for the clock to reach at this point and the clock to reach at this point is referred to as Q. And ideally, we will expect this Q to be as close to 0 as possible.

Not exactly equal to 0, but as close to 0 as possible.

Or basically a minimal value.

And the reasons to have this Q to be a minimal value is related to timing.

And we'll look into a timing section where we do an STA with a real clock and see how does this Q impact the timing.

So for now, just assume, just take it from us that the time, the Q should be close to 0.

And to achieve a Q which is close to 0, this route will definitely not help.

We need to do something else.

And this tree, this tree whatever we have built, that's not a good tree.

That's a bad tree that we have built over here.

It's not a good tree.

So what we'll do, let's try to do something more smarter.

So we'll take this particular clock.

There's a concept of HT.

So HT basically, what does HT does is, it takes this particular clock out.

It calculates the distance from this point to all its end points.

Try to come at the midpoint of the tree and build a tree from that point.

So for example, in this case, the clock 1 will come somewhere at the midpoint of all the 4 flops.

From this point it will again get diverted and come to the next midpoint of the remaining 2 flops.

For example, now this is the midpoint for this flop and this flop.

And this is the midpoint for this flop and this flop.

And then it corrects it.

So with this midpoint strategy, the things are very much sure that the clock now reaches each and every flop at the same time.

Or almost at the same time.

Similar thing we'll do for the next flop also, clock 2.

So clock 2 has to reach at this flip-flop 1.

It has to reach this flip-flop 2.

It has to reach this flip-flop 1 and this flip-flop 2.

So what we'll do, we'll again go to the midpoint of all the flops, which is this one.

We'll go to the midpoint of the next 2 flops, which is this one.

And then correct it.

So with this midpoint strategy, we have made one thing very sure that the time required for this clock to reach either of its flops will be almost similar.

So the skew value that we were expecting should be close to 0, it will be close to 0.

The next thing is clock 2 synthesis buffering.

So by the way, the H3 algorithm has been very well defined or very well described in the clock 2 synthesis videos.

You might quickly want to have a look into those videos.

So next is clock 3 buffering.

So we are not done yet with the clock 3.

When you see this particular clock 3, these are all physical wires.

And these physical wires have got some resistances and capacitances with respect to them.

So for example, when this signal is being sent at the clock port, it has to actually travel to this complete length of the wire.

It actually means it has to charge all the capacitances that it sees in its path and then reach over here.

But there are other problems where it tries to charge all these capacitances on this path.

It's very similar to what we saw in the previous videos.

There will be a huge amount of capacitances.

And as a result of that, there will be a signal integrity problem.

The transition might just go bad.

And the signal that reaches over here, that might not be a replicate of the signal which was sent.

We have seen this in the past in the placement stage.

And the solution to this particular problem was adding repeaters.

So let's take a quick look at the problem first.

So the problem is you have this particular clock route that has to go through this complete physical distance and reach to the clock end points.

And in its path, it sees a lot of resistances and capacitances in its path.

And if you pick up one of the RC network over here, if you supply a step input, assuming that we give a step input at the input of the clock port, if you give the step input at the input of an RC network, this is what you get.

You get a slope of waveform.

And when this waveform is being fed to the next RC network, the output waveform will be even bad.

So as a result of that, the waveform that you see at the input, and we expect this kind of, this is a slightly ideal waveform, but we expect a waveform something like this.

Or whatever was fed at the input, we expect the same waveform at the output as well.

But it doesn't happen.

It just doesn't happen because of the wires, because of the complete wire length it has to go through, and because of this RC network.

So the best solution is to add repeaters.

So we have seen in the past how does the repeaters actually help to reproduce your signal, and what the repeaters actually meant.

The repeaters are nothing but buffers.

These blue, these yellow colored buffers are the buffers that you see.

These are basically repeaters.

We will do similar things for the clock as well.

The only difference between the repeaters that we use for clock and the repeaters that we use for the data path is that the repeaters will have equal rise and fall time.

So we have talked about all of them, all the complete concepts, or the difference between the clock buffer and a regular buffer, or a clock repeater and a regular repeater in the clock synthesis videos.

So it has been very well explained over there.

So basically, in a brief way, the repeaters that are being used in the clock path and the repeaters which are used over here, those have got this particular specific difference where the clock repeaters has got equal rise and fall time.

So let's try to place repeaters and see what happens.

So the first step is we'll remove this clock route, we'll place two repeaters and allow the clock to go through this particular repeater.

So in this case, whatever signal is being generated over here, that is being reproduced at the output of this buffer and then sent again.

And again, the output of this buffer reproduces the waveform and is being sent to these particular remaining flops with the help of these repeaters.

So now clock, from this port, reaches to all these flops through this chain of repeaters.

The red buffers are nothing but your clock buffers.

Similarly, we'll do for the other route also.

We'll take this clock route, let's say we have taken this particular clock route, we have added some repeaters at the center somewhere over here and somewhere over here, and then we went on continuing the route.

From this point onwards, we'll add some more repeaters.

We'll add repeaters over here, add repeaters over here, and then do the routing.

And finally, we'll add some repeaters at the end points also.

Over here, we added two repeaters and we added two more repeaters over here.

And then finally, the clock reaches from this port to these flop endpoints through this chain of repeaters.

So this basically makes sure that the clock signal which is being sent, the clock waveform which is being sent at the input, those have been continuously and faithfully reproduced at the flop clock endpoints.

So with the addition of repeaters, we have solved the problem.

Now, the second problem or second analysis that we need to do is, we need to do a timing analysis with real clocks.

So now when you put clocks into picture, timing scenario just changes.

And we have to see how does the timing scenario changes.

**Crosstalk and Clock net shielding**

We have built, maintained a zero skew.

By skew, we mean the latency difference between the clock port to this particular flip-flop clock pin minus the clock port with this particular flip-flop clock pin.

OK, that's basically the latency, the difference.

And that's basically the skew, the difference in the latency of this clock and this clock.

And we have talked a lot about latency, skew, and all those things in the CTS videos.

In CTS videos, it has been uploaded already.

You might want to have a look into those videos.

So next is the clock net shielding.

So clock nets are considered to be the critical nets in your design, because we have built the clock in such a fashion that we have zero skew.

But accidentally, if there is something, if there is a phenomenon of crosstalk that can happen, if that phenomenon occurs on these lines, your everything, whatever you have designed in the clock side, that might get deteriorated.

And how?

We have a complete course on crosstalk that has been uploaded on Udemy.

You might want to have a look into those videos.

But before that, let me give you a brief discussion on clock net shielding.

So what we do is, we take all the clock nets.

For example, let's say we take an example of this particular clock net.

We take this particular clock net and shield it, something like this.

By shielding, what we do, we protect this clock net from the outside world.

It's like we are encapsulating this clock net to get to be protected from the outside world.

It's like a house on the clock net.

So any signal which is lying over here, any wire which is lying over here, there will be some activities happening on that particular wire.

And that activity is, if the coupling capacitance between that wire and this particular net is very huge, then there might be some coupling happening between the wire, which is sitting over here, and this clock net.

And as a result of that, there are a couple of problems.

But there are two problems, basically.

One is glitch, and one is delta delay.

So we have talked about glitch and delta delay in complete detail in the crosstalk videos.

So let me give a brief discussion on that.

So what happens if there is a glitch?

First of all, what is a glitch?

So just, for example, consider this as one of the clock nets, though it shows the reset pin.

Let's consider this as one of the clock nets, or one of the nets, one of the nets.

And this is the aggressor.

So whenever there is a switching activity happening on the aggressor, because of the coupling capacitance, and the coupling capacitance is so strong, that any activity happening over here will directly impact the net which is sitting close to it.

And this is a net we are talking about without shielding.

This is a clock net without shielding.

What can go wrong?

And as a result of that, you see there is a glitch over here.

So this dip in was supposed to be a logic one.

But because of this particular aggressor getting from logic one to logic zero, you see there is a dip in the voltage.

And as a result of that, you see a glitch.

And what can go wrong if you see this kind of glitch?

So for example, if this was your memory, and memory was initiated with numbers, like whatever you see over here.

This was supposed to be the memory contents.

Accidentally, because of this particular glitch getting inverted by this particular inverter, you might receive a logic high at the reset pin.

And your complete memory might get reset.

And this memory could be a part of any automobile chip, or any of the critical chip.

And if that thing, if it's a part of a critical chip, and this thing goes wrong, then the complete system goes wrong.

So that's why the glitch has become an increasingly important factor to be considered while doing a block tree building, or basically while doing a VLSI design, while doing physical design, while routing.

This particular phenomenon has become an increasing point of concern for all the VLSI engineers.

And all of them are working towards it.

So shielding is one of the techniques that will actually protect this particular victim.

So shielding basically says that, we put a line over here.

We put a wire.

We put a wire over here.

And these wires are connected either to VDD or ground.

Because VDD and ground are the ones which won't switch.

And as a result of that, your victim is protected.

So we have talked about all of them, all of these phenomena in the cross talk videos, which has been uploaded on Odemy.

You might want to have a look into those videos.

So yes, this is the technique.

Basically, by shielding, we are breaking the coupling capacitance between the address and the victim.

So shields are either connected to both the other.

If you place a wire over here, you place a wire over here.

And either both the shields are connected to VDD, or they're connected to ground.

Or one is connected to VDD, one is connected to ground.

The idea is, the shields don't switch.

And if the shields don't switch, there's very high, very less possibility that the victim will switch.

There are other problems where there are ground bounds and voltage droop.

So we have talked about all those in the cross talk videos.

And there is one more thing that can happen.

This is, we talked about glitch.

But what happens if the victim itself is switching?

And the address is also switching?

Then what can go wrong?

So let's say this was the clock that we have built.

We have just taken the clock from the above circuit and placed it in an ideal way.

So for example, this is the clock board.

This is one of the flops.

And there is one flop over here, and there is one flop over here.

And we have built the clock in such a fashion that we maintain a 0 skew.

So by 0 skew, we mean that, let's say, the latency from this clock board to this point of the flop is L1.

And let's say the latency from this clock board to this flop, which is sitting over here, it is not shown, but it is sitting over here, is L2.

Before cross talk, if it was L2.

So we may have built this clock in such a fashion that L1 is equal to L2.

But now what happens because of cross talk, if you see there is some amount of delta delay.

There's some amount of glitch.

So for example, this is the line, victim line.

The input of the victim line is switching from logic 0 to logic 1.

So output of the inverter will be logic 1 to logic 0.

It's supposed to go from logic 1 to logic 0.

But what can go wrong?

Because of this particular glitch that you see, it might impact the logic 1 to logic 0 direction by some amount of delta delay.

If you see this particular thing, it was going from logic 1 to logic 0.

But on its path, it sees the address moving from logic 0 to logic 1.

As a result of that, this particular thing tries to charge, and then it gets discharged.

And because of this amount of bump, you see an amount of delay.

You see an amount of delta delay while switching from logic 1 to logic 0.

And as a result of that, the delay of this particular cell gets impacted.

So initially, if the delay of this particular cell is D, after the cross talk delta delay, you see the delay of this particular cell is D plus delta.

And let's say if this clock cell was a part of L2.

If this clock cell was part of L2, this particular cell delay, which was supposed to be D, now it is D plus delta.

Your complete L2, which was supposed to be your complete latency, which was supposed to be L2, is now L2 plus delta.

So now your skew becomes L1 minus L2 plus delta.

So it's no more 0.

There's some finite value delta.

And imagine we have talked about only, let's say, a handful of buffers, like five to six buffers.

If the clock was for a multi-million chip, this delta might grow very exponentially or linearly.

OK, so that's the impact of cross talk.

And we are trying to protect our clock nets from cross talk by putting a shield around the clock nets.

So shielding is one of the techniques to protect the nets from cross talk.

There are other techniques as well.

We have talked about them in the cross talk videos.

You might want to have a look into those videos.

These slides have been taken from there.

And there are a lot more slides which talks on the similar lines.

OK, so that's the idea of cross talk.

And this is the reason we shield the clock nets.

Ideally, we are supposed to shield this one also.

But we have just lifted that way.

So this net will also get shielded something like this.

So the idea is we have to shield all the critical nets.

Clock nets is one of the critical nets.

And let's say there are some data nets which are also supposed to be critical.

We'll be drawing the data nets in the next video.

And if there are data nets which are very critical, it becomes necessary to shield them.

But it's not always possible to shield all of the nets in the design, because that might just increase the routing resources over here.

As a result of that, we take the clock nets.

We take the critical nets and shield them.

And clock nets are one of them.

So now that we have shielded the clock, we have a most robust clock tree over here.

Setup timing analysis using real clocks

with real clocks.

So with real clocks, the circuitry looks a bit different.

So initially if the circuit looks something like this and the clock tree was ideal, now the clock tree will look something like this.

We'll have buffers, the clock edge will now reach this launch flop and the capture flop through a set of buffers and wires and these are real buffers and wires.

So if the initial equation looks something like this, now your equation, the combinational delay requirement, since we are looking into real clocks, this particular combinational delay or this particular clock delay also has to be taken into consideration.

So now what happens is, this particular clock edge which is at 0, for example the clock at 0 clock edge, this particular clock edge was supposed to reach the launch flop clock end point at 0 nanosecond.

But now because of these buffers in the place, this clock edge will not reach at 0 nanosecond, but it will reach at 0 plus 1 plus 2 buffer delay.

So now your combinational delay initially was theta and the requirement was theta should be less than T. So let's try to modify this one by one.

So let's try to bring up this one first.

Your clock network delay, your clock edge at the 0 clock edge which was supposed to reach at 0 time period at launch flop, it will now reach at 1 plus 2.

So your theta is your combinational delay and 1 plus 2 is the clock network delay, is the launch flop clock network delay.

And similarly, the clock period, the clock edge which was supposed to arrive capture flop clock end point at T nanosecond, it will not arrive at T, but it will arrive at 1 plus 3 plus 4 buffer delay.

So it's T plus 1 plus 3 plus 4.

The clock will send the clock or the PLL or the clock generation, the clock source will send the clock edge at 0 nanosecond over here, but it will reach the launch flop and the capture flop after this buffer delay only.

This is the best way to interpret this.

So let's call this 1 plus 2, this 1 plus 2 which is present over here, let's call it as delta 1.

And the delta 1 is nothing but your launch flop clock network delay, because this is the delta 1 is the time required for the clock to reach from the clock end point to the launch flop clock clock pin.

And this, let's call 1 plus 3 plus 4 as delta 2.

And delta 2 is the time required for the clock edge to reach from this point to the capture flop clock end point.

So basically, initially this particular arrow which was at 0, and the reason it was at 0 was that the clock was supposed to reach the launch flop clock end point at 0 nanoseconds.

And the arrow was placed at T because the clock, the second edge of the clock was supposed to reach at capture flop clock end point at T nanosecond, which is 1 nanosecond.

But now this clock network coming into picture, if you see, the delta, the arrow which was supposed to be at 0 is now shifted at delta 1, shifted by delta 1.

And similarly, the arrow at the right-hand side, which was supposed to be at T, is now shifted by delta 2.

So now initially the arrow, the window was between from this point to this point, and now the window is shifted from this point to this point.

And that's what has been written down in the form of equation over here.

It says that theta, which is the commercial delay, plus delta 1, which is this particular shift, should now be less than T, which is the time period of 1 nanosecond, plus delta 2, which is the capture flop clock network delay.

So this is basically the visualization of a setup time equation looking into real clocks.

And now we have to add the remaining ones, because the concepts of setup time and uncertainty will still hold good in this particular case as well.

In this case, we have just shifted the clock by some edges, but the concept of setup time and uncertainty are still valid for this shifted edges as well.

So the equation will remain still the same.

The delta 1 and delta 2, if you subtract delta 1 minus delta 2, it's called skew.

It's called clock skew.

You might want to have a look at the clock the synthesis videos that has been uploaded.

There we have explained that delta 1 minus delta 2 is called as a skew.

And again, you have the setup time for this particular capture flop, which has to be subtracted from S. Initially in form of one second T, which is separate from the complete equation.

So now, this thing becomes your real complete equation for a setup time analysis.

Any circuitry on the chip satisfying this particular equation is ready to work at 1 GHz.

Anything which is valid in this one, we have a term for that, and that term is called a slack.

Let me try to introduce you to the term slack.

So the right-hand side is called the data required time.

And why is it so?

It says that this complete circuitry, the circuit tells to the designer that this complete circuitry needs at least this much amount of time to function at 1 GHz.

And what the user sees to the circuit is that this is the data arrival time.

I am arriving at this amount of time.

Now, data arrival time should be less than data required time.

So basically it says that if you subtract data required time minus data arrival time, it should be a positive value.

That is what we have said over here.

If it is negative, then we call it as a slack.

Even if it is a positive, it is called as a slack.

But when it is valid, then we call it as a negative slack or we call it as a positive slack.

And slack is expected to be either positive or zero.

So this is what is the concept of data required and data arrival time.

And this is very frequently a term being asked or being used in the VLSI world.

What is the data required time and what is the data arrival time?

So this is what it actually means.

In case of a setup time analysis.

So this becomes your data required time, which says that this much amount of time, that the data should be arriving within this much amount of time.

And this data arrival time says that I arrived at that time.

And now if data arrival time is less than data required time, which means your slack is positive.

Which means that data required time is greater than data arrival time.

Your slack is positive and we are good to go.

If the data arrival time is more than data required time, your slack becomes negative and we call it as a violation.

And then we have to find out ways to fix the violations and so on.

But this is the basic concept.

Okay, so that was about the setup timing analysis.

Let's move on to hold timing analysis.

So hold timing, it's still a timing analysis based on launch and capture flow.

But things change a bit over here.

The analysis becomes a bit different.

Let's see how different is the hold timing analysis is.

So we'll do the same thing.

We'll do the first timing analysis with ideal clocks and then move to real clocks.

The topic says with real clocks because we have built the clock tree.

We have already seen the clock tree.

So let's move on with the real clocks only.

Okay, so the specifications remain the same.

We are at clock frequency one gigahertz.

It is clocked at one nanosecond.

But actually this really doesn't matter in case of hold timing analysis.

We'll see how, okay.

So for example, we have the circuit.

We have the launch flow.

We have the capture flow.

And we have this particular commercial delay of data.

Okay, so in this case, it's a bit different kind of analysis compared to the setup timing analysis.

And the reason is in setup timing analysis, we used to send this particular edge to launch flow and this particular edge to capture flow.

But in this case, we'll be sending this edge to launch flow to launch the data.

And we'll be sending this particular edge to capture flow to capture the data and also to check for the hold timing analysis or the hold condition.

So the hold condition says that the commercial delay of this particular circuitry should be greater than the hold time of the capture flow.

So now the point over here to understand is what is this hold time.

So remember some time back, we had introduced you to the internals of capture flow, which had two muxes.

One mux was from D to the internal pin.

And the second mux was from the internal or the data to send from internal of this particular flow to the outside world.

So this hold time refers to the second mux delay.

So let's quickly understand what is happening over here.

Okay, so for example, we have this particular mux, which represents the interiors of the capture flow.

Okay, so this is the capture flow.

This is the mux, this is the interiors of this particular capture flow.

And this is how it functions.

So when the data is at log, sorry, when the clock is at logic one, when the clock is at logic one, the data which is present at the QM is sent outside this particular flow.

And that time period that it takes to data send outside of this particular flow is the delay of mux two.

Okay, so for example, in the setup timing case analysis, the data was sitting somewhere over here.

And the time required for the data to reach from deep into somewhere at the center of the flop was the mux one delay.

And that was captured as a setup time of this particular capture flow.

In case of a hold time, now since we have a data in this particular capture flow, and the data has to be sent outside of this particular capture flow, the time required for the data to send outside of the capture flow is this mux two delay.

So capture flow knows all these things, and it doesn't expect any data to arrive when it is sending data outside.

And that amount of time that it is sending data outside is the hold time of this particular capture flow.

So capture flow sends a message to the launch flow that whatever your data is, please send it after mux two delay, or please send it after my hold time.

Please hold your data till I send that existing data outside of this particular flow.

Okay, so that's what this capture flow is saying, is coming to this circuitry that please hold your data, please hold your data till I send the data, till I send my existing data outside of the capture flow.

And that's basically hold time.

So this is how you're satisfying or satisfying condition for your hold time.

If you try to add the clock network delays over here, so the equation, it is just a matter of adding the clock network delays in this particular equation.

So in this case, when you add the clock network delays, it says that the combinational delay is now theta plus one plus two, which is the launch flow clock network delay.

And the hold time is now shifted because this particular edge will now reach the capture flow clock endpoint after one plus three plus four.

So now the equation for this particular thing becomes something like this.

It says that theta plus delta one, now theta has to be greater than edge, that is true because capture flow wants the data to hold for some amount of time.

And on the top of that, you have the delta one and delta two, which is the clock network delays of launch flow and capture flow.

So now the equation becomes something like this.

It says that the launch flow will receive the clock edge after delta one, so that has been captured over here.

Your capture flow will receive the clock edge after delta two, after this three buffer delays, which is captured as delta two.

**Hold Time Analysis**

delay, to start with, the commercial delay should be greater than the whole time of the capture flop.

And once the clock network delay comes into the picture, if you see this particular edge, the first thing that we need to do is shift this particular edge that will reach the launch flop and the capture flop by the required amount of time.

So that's what we did.

So this particular clock edge, when it reaches the launch flop, it takes around two buffer delays, and that's basically captured as delta 1.

This particular clock edge, once it reaches the capture flop, it takes around three buffer delays, so that's being captured as delta 2, and the rest of the equation remains the same.

So if you remove delta 1 and delta 2, for example, if delta 1 and delta 2 are equal, the equation still remains the same.

Basically, your commercial delay should be greater than the whole time.

So this was about the whole time equation.

Now to add more and more uncertainties to it, so we have to assume we have a whole time of 10 picoseconds and an uncertainty of 50 picoseconds.

Over here, actually, really, if you say the uncertainty doesn't matter much, because in this case, the edge that is going to the launch flop and the capture flop is the same.

So the amount of jitter that will play a role for the launch flop and the capture flop will be the same.

For example, if this particular clock edge, let's say, doesn't arrive at zero, but at somewhere 10 nanoseconds, 10 picoseconds, the arrival difference will be the same for the launch flop and the capture flop, because it's the same edge that is going to the launch flop and capture flop.

So uncertainty over here actually really doesn't matter, but still to be on a more realistic side, we have a number, which is 50 picoseconds, a very, very low value for the uncertainty.

So a PLL, which is supplying a clock for the launch flop, it's sending the same edge to the launch flop and the capture flop.

So the uncertainties and the jitter will be the same for both the clock edges that is reaching at this point and this point, because they're essentially the same.

Hence, the uncertainty value is kept a bit low over here, as compared to the setup time analysis, where the uncertainty was close to 90, 200 picoseconds, or even more.

Uncertainty over there are even more, because there are different edges that is going to the launch flop and the capture flop.

This edge goes to the launch flop, and this edge goes to the capture flop.

So the amount of uncertainty is even more.

In this case, since it's the same edge, things have become very simple for us, and the uncertainty can be kept as a low value.

So with these numbers in place, we have to just subtract or we have to just add the uncertainty value as well.

So it says that, apart from the hold time requirement of this particular capture flop, I need some additional value for the uncertainty also.

So please hold your data for my internal hold time and for some uncertainty value.

And finally, if you want to give a name to this particular equation, it will be called as the right-hand side will be called as the data required time, and the left-hand side will be called as the data arrival time.

And in this case, the definition of slack will be a bit different.

So slack will be set as data arrival time minus data required time, because in this case, we expect the data arrival time to be greater than data required time.

This is in contrast with the setup time analysis, where the data required time was expected to be greater than data arrival time.

Or the other way around, the data arrival time was supposed to be less than data required time.

Remember, it was the combinational delay theta should be less than t.

That was a requirement in the case of setup time analysis.

But in this case, since the arrival time, the requirement becomes a bit different.

In this case, the arrival time is expected to be greater than the data required time, and the difference is referred to as slack.

And as it is, the slack is expected to be either positive or zero.

If the slack goes to negative, it is referred to as a violation.

And if it goes to a negative value, it says that the data arrival is much before the data required time, which means your combinational delay is very fast.

Basically, if this time goes less than the data required time, your delay is very fast, and we have to slow down.

That's what it actually means.

So what we saw in both old and the setup analysis is there is an introduction of new terms, and those terms are this delta 1 and delta 2.

So let's try to identify the delta 1 and delta 2 part from the existing design that we were looking into.

So this was the design, and we were doing a timing analysis with real clocks.

We have all sets of equations.

We have equations for setup.

We have done a similar kind of analysis for old, and the only difference was that now there are two new terms, which is delta 1 and delta 2, and that is the launch clock delay and the capture clock delay.

So let's try to trace those parts from this particular design.

And since we did it for single clock, let's try to identify those parts which have been driven by single clocks, for example, clock 1 and clock 2, and then identify the launch and capture clock delays.

So let's take one of them.

So we will take these are the two circuits, or these are the two sections of the circuit that works on single clock.

They have been clocked by clock 1, and these have been clocked by clock 2, or whatever we are going to identify from the design.

So these have been driven by clock 2.

These have been driven by clock 1.

So let's try to identify the delta 1.

So the combinational delay we already derived in some previous videos, where the combinational delay was the clock 2 Q flip flop delay plus the estimated wire delay plus delay of 1 estimated wire delay and so on.

So these were all the estimated wire delays.

For the clock path, you have the real wire delay.

So there is a mix match of delays that we are talking about over here.

So if you talk about delta 1, which is the launch clock network delay, so this is your launch flop.

This is the clock port.

And let's talk about what the delay will be.

So you have the real wire RC delay.

Over here, it was always the estimated RC delay.

Over here, you have the real wire RC delay plus the buffer delay.

This is the clock buffer delay plus you have the real wire RC delay between these two buffers.

You have this buffer delay.

You have this real wire RC delay.

You have the other buffer delay.

Then you have the second real wire RC delay.

Then we have to move in this particular direction where we are moving towards the launch flop.

So you have the launch flop, this buffer delay, and finally, you have this wire RC delay.

So this forms your delta 1, the launch clock network delay.

Next, let's move to delta 2.

So let's place it somewhere at the top.

So delta 2 will be, again, there will be some amount of common path from the clock until this point, and that part will still remain the same.

It's like wire RC delay plus the buffer delays plus real wire RC delay of the second one plus the next buffer delay.

Next, we have the RC delay of the next wire.

Next, we have the buffer delay.

Next, we have this wire delay.

So this particular wire delay will be different from this particular wire delay because we have two different buffers over here, two different kinds of buffers.

Over here, it looks the same because it's all of the same size, but we might have different kinds of buffers over here and different kinds of buffer over here.

That is the first thing.

Second, the wire length, if you see, that is being constant.

So when you try to calculate the capacitance from this point till this particular wire, so it remains the same.

Only the input capacitance of the buffers plays a different role.

So next, we have this particular buffer delay.

Then, you have the final wire delay.

So this part goes to your capture clock network delay.

So this forms your delta 2.

And as per the equations or as per the analysis that we did for setup and hold, you need to have this.

So this is the skew value.

Skew is delta 1 minus delta 2, which is the launch clock network delay minus capture clock network delay, and more of it.

So whichever is bigger, this equation just turns out to be positive.

It's for that.

It is more.

So skew is, this is delta 1 minus delta 2 is your skew value, and then you have your setup time equation.

So setup time equation says that your combinational delay plus the launch clock network delay, the combinational delay plus your launch clock network delay should be less than the time period, the period of the circuit or the clock frequency plus the capture clock network delay.

So everything else remains the same, except that there is two new terms over here.

We have delta 1, we have delta 2.

Similarly, for hold, we have got everything else remains the same.

We have these two new terms, delta 1 and delta 2, and the hold uncertainty.

So in the setup case, it was setup uncertainty minus setup time.

In this case, it is the hold uncertainty, hold time, and plus we have these two new terms, delta 1 and delta 2.

So there are other things that happens in this timing analysis.

For example, if you see, from this point to this point, you have this common clock path.

So there is some amount of extra precision that gets added when you try to do an on-chip variation, when you try to put some delays and all those things.

So we have talked about that in the upcoming videos.

You might want to have a look into those videos, what is CPPR and what is the OCB factor that we are talking about.

So we have talked about that.

But from a basic understanding, from a flow point of view, from a flow completion point of view, this is what it does.

So at this point of time, we have done the clock pre-synthesis, and we have done a timing analysis with real clocks, for single clock.

The next step is to do a timing analysis for multiple clocks, using the real clocks.

So now we have got real clocks.

In the previous slide, we did the real clock timing analysis, but with single clock.

Now also, we had a scenario in our design where we had multiple clocks, going to multiple clocks.

So the next step is to do a timing analysis with real clocks, with multiple clocks.

So things are getting more and more complex.

Things are getting more and more interesting.

So let's try to start start from this point in the next video.

Thank you.