VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh

Department of Electronics and Communication Engineering
IIIT-Delhi

Lecture 32
Constraints I

Hello everybody, welcome to the course VLSI design flow RTL-2 GDS. This is the 25th lecture. In this lecture we will be looking at constraints. Specifically we will be covering these topics in today's lecture. First we will be looking into the basics of constraints, then we will be looking into various types of constraints such as clock constraints, input output constraints and timing exceptions. So let us first start with what is a constraint? So constraints are basically the requirements of a design that as a designer we want to be fulfilled.

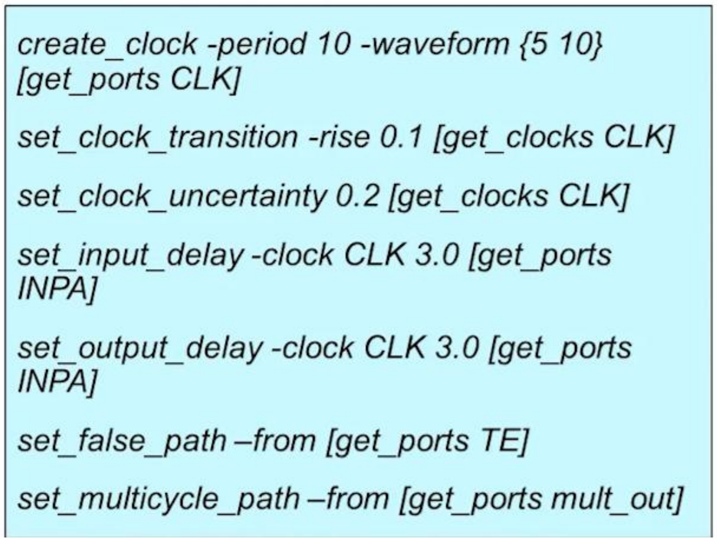
So the constraints convey the designer's requirements to the implementation and verification tools. So the requirements that need to be honored in our design and the EDA tools which implement our design they try to honor those constraints or requirements and the verification tools verifies that those constraints or requirements are indeed fulfilled in our design. So the constraints are basically a mechanism to convey our requirements to the tool. And using constraints we can also convey information about a design that could potentially be exploited by the EDA tools to improve its PPA or perform verification task and these information typically are the information which the tool cannot easily derive on its own or the tool cannot come up with this information on their own or is very difficult for the tools to derive.

So as a designer these information are easily available to us and we provide those information to the EDA tools using the constraints. Now how are these constraints defined? So these constraints are normally specified or defined in a format which is known as Synopsys design constraints format or in short we say SDC format. So the files that contain the constraints typically provide an extension to those files as sdc.sdc. For example if the name of our design is top then we will typically create an sdc file with the name top.

sdc. And what is this Synopsys design constraints or sdc format? So this sdc format is

ASCII format meaning that the file that we write in sdc format we can easily read it in using an editor and we can also make editing or do editing in these files manually. So these sdc formats basically follow the syntax and semantics of tool command language. This is a scripting language tool command language or which in short it is known as TCL. So TCL is a scripting language which is widely used in VLSI designs. So if you are not familiar with TCL language I will suggest that please look into this TCL language it is very easy to learn and lots of materials are available over the internet using which you can learn the at least the initial syntax and semantics of language in a few hours only it is very simple language.

So if you want to make a career in the VLSI design or in digital VLSI design then I will strongly suggest that please at least get familiarized with the language TCL. So let us take an example of the sdc file how it looks like.



```
create_clock -period 10 -waveform {5 10}
[get_ports CLK]
set_clock_transition -rise 0.1 [get_clocks CLK]
set_clock_uncertainty 0.2 [get_clocks CLK]
set_input_delay -clock CLK 3.0 [get_ports
INPA]
set_output_delay -clock CLK 3.0 [get_ports
INPA]
set_false_path –from [get_ports TE]
set_multicycle_path –from [get_ports mult_out]
```

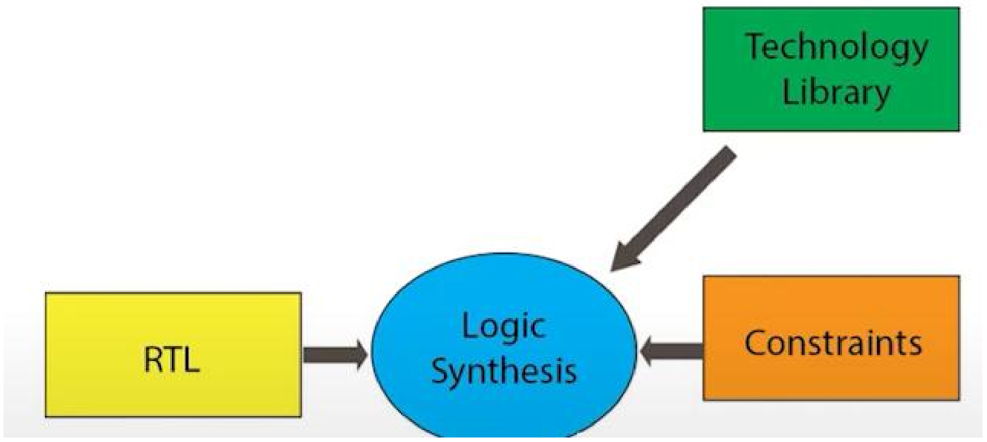
So this is an example of an sdc file. So when we this this sdc file will be or this information that is shown here will be contained in the file if we write top dot sdc then in this top dot sdc these lines will be written which is shown here. Now these lines contain commands the sdc command like for example this is an one command another command and so on.

So whenever the tool reads this sdc file then it executes this command one line by line and provides definitions to the design objects which are being referred to in these commands. So what these commands mean and what are design objects and what are those definitions to the design objects that is provided by the sdc command will be looked at in detail in today's lecture or in this lecture. So just to get an initial understanding of how the sdc file looks like I have shown this file. So it contains commands like create clock set clock transition set clock uncertainty set input delay set output delay set false path set multi cycle path. So we will be looking at all these commands in today's lectures.

Now let us understand how the tool uses this constraints file or sdc file when it gives it to the tool. So most of the constraints that are the sdc command that we give in the sdc file are related to the timing of the design. And therefore these constraints are also known as timing constraints. Now how are these constraints used by the implementation tool? So implementation tools means the logic synthesis tool or the physical design tool.

So those are actually making changes in our designs. Those are the implementation tools. So these sdc constraints are employed by the implementation tool to gather information about the expected timing behavior. For example what should be the clock frequency at which we expect that our design should run or design should be able to deliver the frequency. So this is just one of the examples of what is a constraint and which are relevant to the implementation tool.

Now the implementation tool will actually use this sdc command and then make changes in the design such that the design is able to achieve a desired frequency. So if you from our earlier discussions we have said that the logic synthesis takes three inputs. . Primarily three inputs.



The first is the RTL, the RTL model that we want our design to be implemented or that defines the functionality of our design and the technology library.

And the third thing was the constraints. So this is what the constraints we are discussing in this lecture. So logic synthesis will take these constraints as an input and based on this RTL and the technology library it will generate a netlist. So this netlist should be such that the constraints are obeyed by the netlist ideally. But so for example suppose we are writing an operation like y is equal to a plus b .

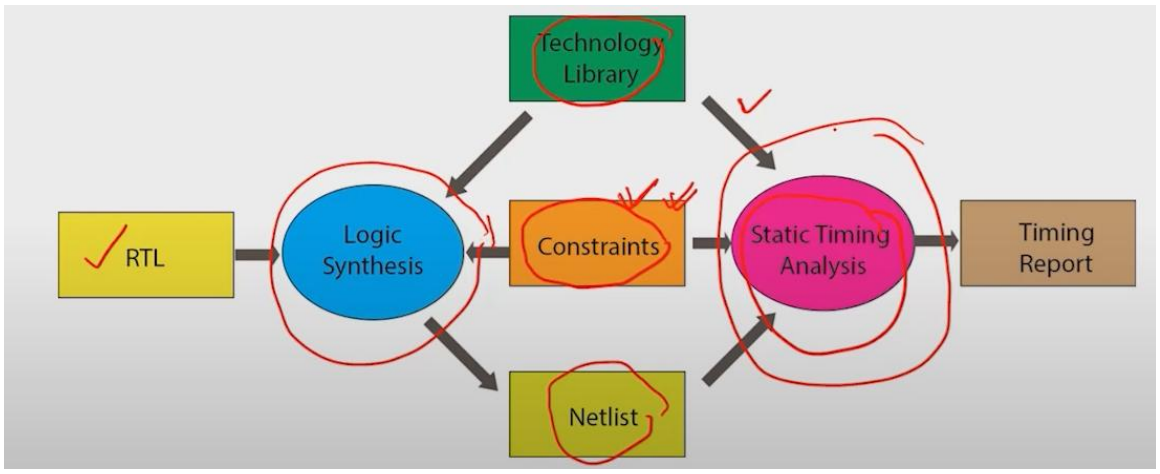
Now this plus operator can be implemented in many ways. For example it can be implemented using say ripple adder ripple carry adder or it can be implemented using carry look ahead adder carry look ahead adder. Now the tool while implementing which of these two adders should it put in our design or instantiate in our design . So it will depend on what is the expected performance or the expected frequency at which we want our design to run. For example if we say that I want that frequency to be very high that means that the circuit operates at very high frequency.

So in that case the tool will probably use the implementation tool and will use a carry look ahead adder because it is fast. But the downside is that it will take more area. Now on the other hand if we say that I do not want my circuit to operate at very high frequency even a moderate frequency is acceptable. In that case probably the tool will use a ripple carry adder which is not as fast as carry look ahead adder but it saves in the area . So the constraints is basically directing our synthesis tool or implementation tool towards one solution and this constraints is providing or the constraints meaning that the targeted frequency of our design is what is giving hint to the tool that what kind of implementation should be in our netlist.

So this is the first area where this constraints file will be used or the SDC file will be used. The next part is the is the FTA tool that is the FTA tool as we have seen in earlier lectures is very basically verifying whether our design is operating at the or meeting our timing constraints in terms of required times and the arrival time is such that the required time is or the required time constraint is honored by the arrival time both for the setup case and the hold case and so on . That is the purpose of the STA tool. Now what inputs does the STA tool take? So STA tools take input as the technology libraries because those technology libraries contain information of the delay of the cells that are there in the in the in the netlist and it also takes the constraints . So it takes constraints which need to be satisfied by our design .

So and of course the netlist that was generated by a logic synthesis tool is what our design is. So static timing analysis will do an analysis and generate a timing report based on the constraints that we are giving. For example if in the constraint we are setting the clock frequency as very high . In that case the required time for setup analysis will become very small and therefore the chances of violation increases . So now the constraints is basically providing the information of the clock frequency to the STA tool and using that information the STA tool is checking whether the constraints are honored in our design or not.

Our design in this case is the netlist or netlist plus the combination of the libraries . So what constraints are there? Constraints are basically providing a common denominator both for the logic synthesis tool as well as the static timing analysis. So the common denominator for both of them is the constraints for these two tools. The logic synthesis tool tries to meet the constraint and the STA tool tries to verify whether the logic synthesis tool is able to meet the constraint or not. Therefore the logic synthesis tool tries to meet the constraint but it may not be able to meet the constraint all the time.



For example if we say give an arbitrarily very high clock frequency then there would not be any solution which will be able to deliver that clock frequency . And therefore the implementation tool may not honor the constraint and the STA tool is basically checking whether the logic synthesis tool or the implementation tool was able to meet the constraint . So that is why the constraints provide a common ground for both these tools: the synthesis tool or implementation tool and the STA tool. Now what is the origin of the constraint from or the constraints file how do we get it . So normally constraints files are written manually .

So why they are written manually is because the designer has certain knowledge or the information which the tool cannot derive. For example in our design there is a port which will receive the clock from the external source . So the frequency of that of the external clock cannot be derived by the tool by any tool on its own. That information is only with the designer and the designer needs to give that information and the tool cannot derive it . So that therefore the constraints file needs to be written by the designer because the information in the constraint file is not available in the design or it may be very difficult in some situation about the functionality though it may be may be derivable but that may be very very difficult and therefore we do not give the responsibility of deriving those things on the tool but provide as a designer those constraints or that those information to these tools by manually writing the STC file.

So what is the information we will see in the subsequent slides? So however there are certain tools which automate auto which help in writing the constraints file or they claim to be saying automatic constraint generation tool but what these tools typically do is that they provide a template . So the template is that these are the commands but exact information of that that the command will take that has to be actually written by or the field by the designer. So even though there are some automatic constraint generation tool some user intervention is always required. For example, what is the targeted clock frequency.

The tool cannot derive on its own that it needs to be entered by the or that information needs to be provided by a designer to the tool through the constraints. So now since the constraints files are manually written there is a greater chance of introducing errors and therefore while writing constraints files we have to be very very careful . So if the constraints are wrong then the design implementation tool can produce unexpected results . Of course the design implementation tool will work on what information we are giving as a constraint to the tool . Now if that information is not correct there can be many sources of error then the expected design behavior sorry the design behavior will not match our expectation.

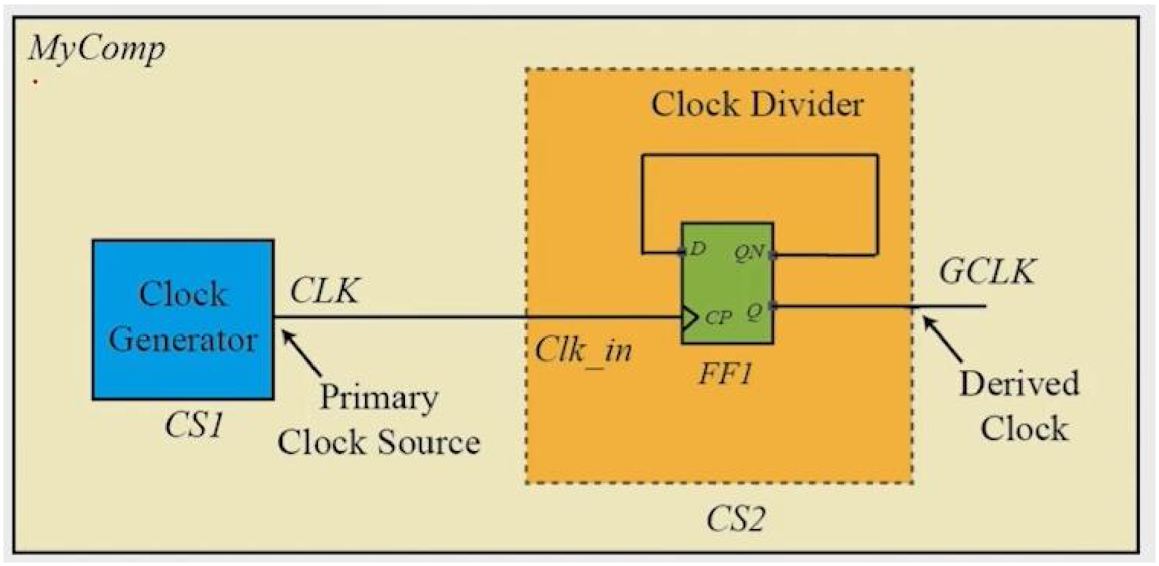
So there should be a consistency between different constraints and the constraints should be consistent and should be consistent with the design attributes. So what we mean by this consistency between different constraints and what we mean by it should be consistent with the design attributes we will see in subsequent slides. Now what are the various types of constraints? So there are broadly four types of constraints. The first is related to the clock and its attributes such as frequency, duty cycle, skew, uncertainty and delay.

Constraint is related to the environment in which we expect our design to work. So these constraints will tell us what is the behavior of that signal that is coming to our circuit and what is the expected behavior of the signal that is going out from our circuit. So those are environmental constraints and that needs to be provided by the designer. The third type of constraints is related to the functionality of our design and these are informative constraints. So for example there could be some timing exceptions where we do not want the normal SDA analysis to be done but in some different manner.

What is that different manner we will see in subsequent slides or can we inform the tool about this? The given constraint is valid for which mode of our design and so on. And the four types of constraints are related to design rules and optimization constraints. For example the maximum slew that should be that can be there at the input port at a port or what would be the maximum capacitance possible at a pin and some soft constraints like maximum area and so on. So these are four types of constraints and we will see the major types of constraints in today's lecture. So let us first look into the constraints related to the clock.

So when we define a constraint file the first information is related to the clock signals in our design. So to specify the clock signal we have to specify source. So in a design there can be two types of clock sources. So what are these two types? The first is the primary clock source. So primary clock sources are those clock sources whose waveform is independent of other clock sources in the design.

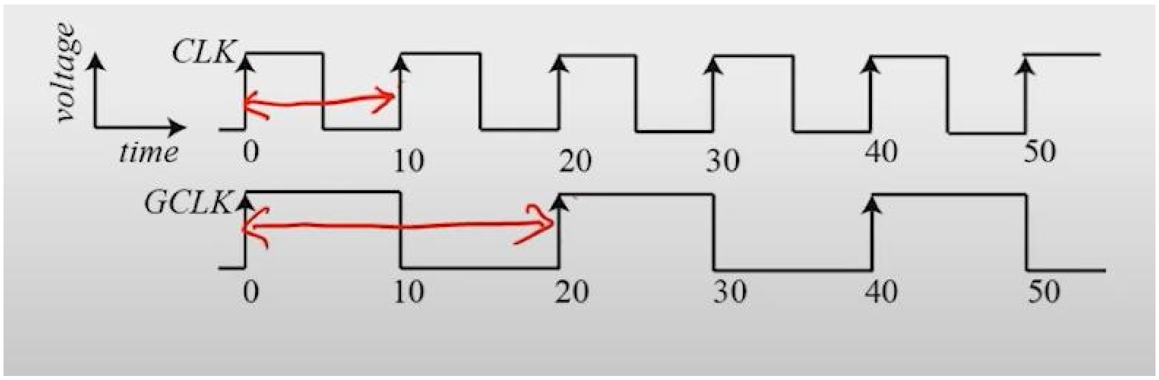
So the second type of clock source is derived clock source. So in this case the waveform depends on the waveform of other clock sources. So the primary clock source the waveform depends on does not depend on any other clock source in our design and the derived clock sources are those clock sources whose waveform depends on the other clock sources in our design. So let us take an example. Suppose this is our design, we are calling it MyComp the name of our design and there is a clock generator which generates a clock signal at the net clock .



We call this clock generator since the frequency and the clock signal attributes are not dependent on any other clock source. We can say that this clock generator is a primary clock source generator. This clock generator is a primary clock source generator. Now this signal is going to another clock generator. So the first clock generator we call it as CS1 and the next other clock generator we are calling it as CS2. So it is going to another clock generator .

This module is known as CS2, which is the other clock generator. Now this clock generator receives the clock signal CLK from the primary clock source and internally makes some changes. So what is this change? So we can see that the Qn that is the naught of Q is going to the D pin and what is going to the clock pin is the original clock. So if suppose the initial value of Q was 1 so the Q bar will be 0 Qn will be 0. So now in the next whenever the next clock edge comes D will become D will take a value D since is 0 the Q output will take a value of 0.

Now since Q is 0 Qn will be 1 . So in the next clock side when the next clock edge comes then it will take a value of 1. So the Q pin here will be taking a value of 1 0 1 0 at each whenever the clock edge comes. So if we suppose that the clock signal that was generated here was generating a clock signal at clock with a clock period of 10 time units . Then initially the value was supposed to be 1 the Q value here was 1 . So it will remain 1 until the next clock edge comes.



Whenever the next clock edge comes since the Q value was 1 the Q naught value will be Qn value will be 0 and that will be at the D pin. So whenever the clock edge comes it will become 0 and again whenever since the Q value is now 0 . So the Qn will be 1 and whenever the next clock edge comes it will take a value of 1 and similarly whenever the next clock edge comes then it will take a value 0 and 1 . So we see that the waveform that will be produced is the waveform that will be produced at the point GCLK here that will be a function or that will be dependent on what is the clock attribute and what are the attributes of the clock signal CLK . For example here the clock is frequent if the time period was 10 time units.

So the time period of the derived clock or clock or GCLK is 20 minutes. Now if we change it from say 10 time units to say 15 time units the time period of the signal clock then the GCLK value time period will be 2 times of 15 that will be 30 . So the frequency of GCLK is related or is dependent on the frequency of CLK. So CLK is considered as the or CS1 is considered as the primary clock source and CS2 is considered as the derived clock source or generated clock source. So the clock from which we derive another clock is known as the master clock and in this case CS1 is the master clock. This one is the master master clock of the clock source CS2.

Now how do we give this information to the tool using the SDC file let us look into it.

So we use the command create clock to define the primary clock source in our design. The primary clock source in our design is defined using the command create clock . Suppose this was our design so the name of the design is mycom . Now let us see how we specify or write the SDC file for it so that we provide the definition of the primary clock sources. So suppose so before writing the SDC file typically we can, if we want specifically to mention the name of the design then we can write current design and the name of our design so mycom in this case .

So at times the tool can infer the current design of that that it is operating on and if we want to explicitly mention that this SDC file is valid for a given design then at the top of our file we can write simply as current design and my design name. So the design name is mycom in this case and then we define the clock source so suppose that our design mycom was receiving a clock signal from an external source and it was receiving it at the port clock underscore in . In that case how do we write the clock constraint so we write use the command create underscore clock and then give some name to this clock so we give the name as ext underscore clock whatever so this name is arbitrary only thing is that two different clock should not have the so distinct clock should not have the same name . So we may give the name name to this clock as ext underscore clock and later on we can refer to this clock by this name and in the later part of our SDC file we need to refer to this clock again and again . So while referring to this clock will use this name ext underscore clock then we say that what is the period time period of our clock so we said that minus period 10 and then we say that that what is the waveform so what does the waveform show or indicate so waveform is defines the time when the clock edges occur starting from the rises so the rise edge will start at 0 and then the fall edge will come at 4 and the time period will be 10.

And then we define the we give the information to the tool that where this clock definition needs to be applied and that that is we are using this command get ports get ports clock underscore in so the clock underscore in is the name of the port and we have used the command SDC command get underscore ports. So the SDC supports commands like get underscore ports get underscore pins get underscore cells and other others there are other commands which we can say get underscore star by star that can be various types of design objects such as ports pins cells clocks and other . So what does this command get underscore ports do so this command basically gives us the the reference to the design object that we are pointing to by this name so clock underscore in is the name of the port in our design so get underscore ports is returning the reference of our reference of that design object clock underscore in in this case . So now whenever we apply constraints or whenever we write a constraints or the SDC command then that command is defined on some design object. So in this case this create clock command is associated or is defined or attached with the clock port with the port whose name is clock underscore in.

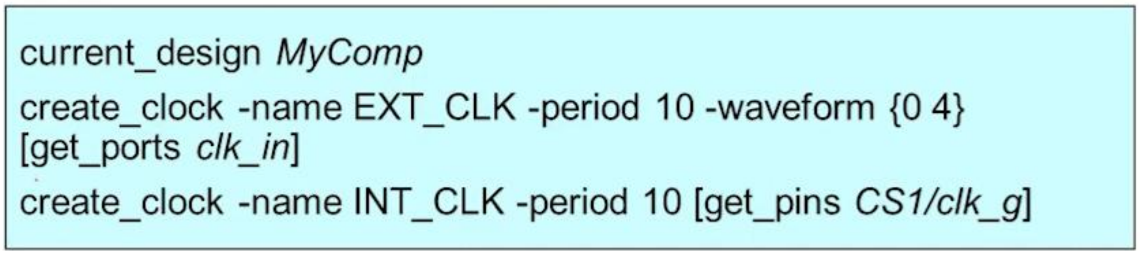
So now when we run this command then this clock underscore in port will be understood that there is a clock source there with the given time period and the waveform . So what will be the waveform? It will be something like this waveform. We said that it starts as 0 the positive edge and then 4 defines when it is falling and 10 defines the clock period . So this is what the waveform will be or the clock signals waveform will be something like this and that waveform will be associated with this design object clock underscore in . Similarly if suppose there is a pin whose name is clock underscore g and it is an internal clock it is generated inside our design itself and it is an internal clock and it is a primary clock source meaning that it is not derived from any other clock source . So in that case we can provide a definition to this pin: the pin name is CS1 CS1 is the name of the instance and clock underscore g is the name of the pin.

So we use the command get underscore pins note that we are using the pin get underscore pin not the port. Port refers to only the external interface of our design pin refers to the internal interfaces of the instances. So here it is a pin clock underscore g is a pin so we write the command get underscore pins CS1 this is the instance name and the pin name . So this is how we define the name of the pin. So we write get underscore pins this name so the create clock definition now is applied to or this definition is applied on this pin.

So the tools will now understand that clock underscore g is a special type of pin. It is basically generating a clock which has got a clock frequency of or clock time period of 10 . So in this case we have not specified the waveform in this case though the tool will infer that its duty cycle is 50 percent so the edge the falling edge will be at 5 half of 10 and so on . So the tool will derive on this. So this is how we define the definition of primary clock sources in our design.

So one point of or rather to a few points that needs to be noted. So the first point is that we are not providing any time units of time here . So for example 10 we have written 0 4 and so on. So the unit of time will be taken from the library. So the library that we load together with our constraints file the technology library will contain the units of time and from there the tools can derive that what will be the what will be the or what what is the time unit for that that should be used for an SDC file. Otherwise if you want to explicitly mention the time units then there is a SDC command set underscore unit to define what is the what is the time unit that needs to be considered.

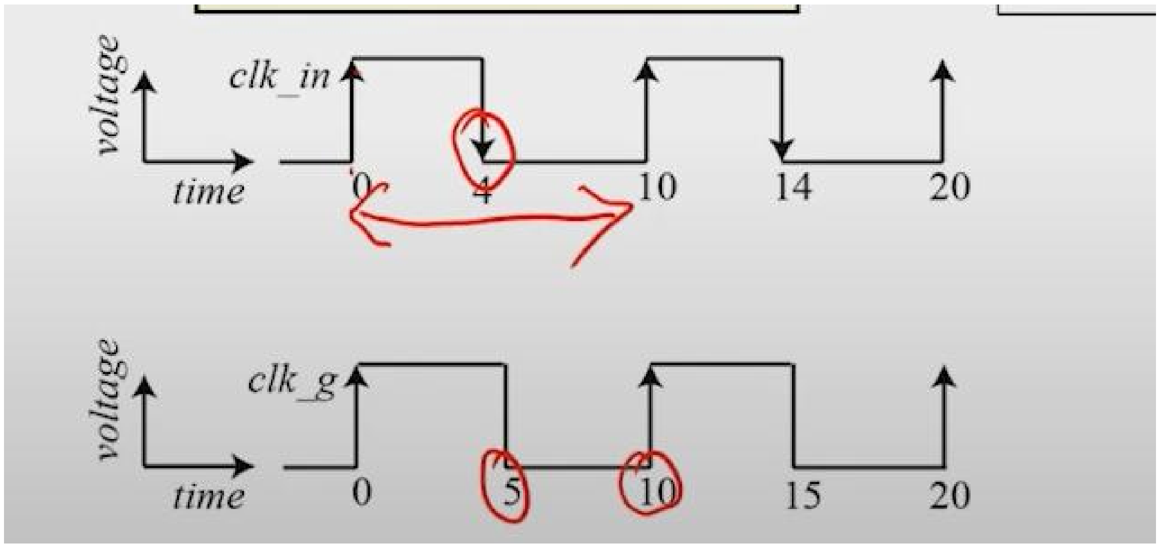
So this is the first point. The second point is that though the name of the command is to create a clock . So you should not interpret that whenever we run this command that tool implementation tool will actually create a clock generator for you . So sometimes students get confused that if I write a create clock command then the implementation tool might actually create a PLL or a clock generator in our circuit that is not correct. So what this clock these SDC files do is that these SDC files provide definitions and constraints to our design.



```
current_design MyComp
create_clock -name EXT_CLK -period 10 -waveform {0 4}
[get_ports clk_in]
create_clock -name INT_CLK -period 10 [get_pins CS1/clk_g]
```

These SDC files do not directly change our design. They do not directly change our design. They only give us some instructions or constraints to our design which are honored by the implementation tool. So suppose there was where we gave a name of a pin CS1C clock underscore j lg and there was no clock generator in our circuit and there was no pin with the name say clock underscore g and the instance name as CS1. In that case what the tool will do the tool might provide some warning or an error .

So that is basically a wrong constraint. You shouldn't expect that the tool will actually implement a clock generator or instantiate a clock generator in our design when we give this command create clock. Then how do we derive how we define the derived clock source in our design. So we use the command create generated clock for this . So we define and use generated clocks to define derived clock sources.

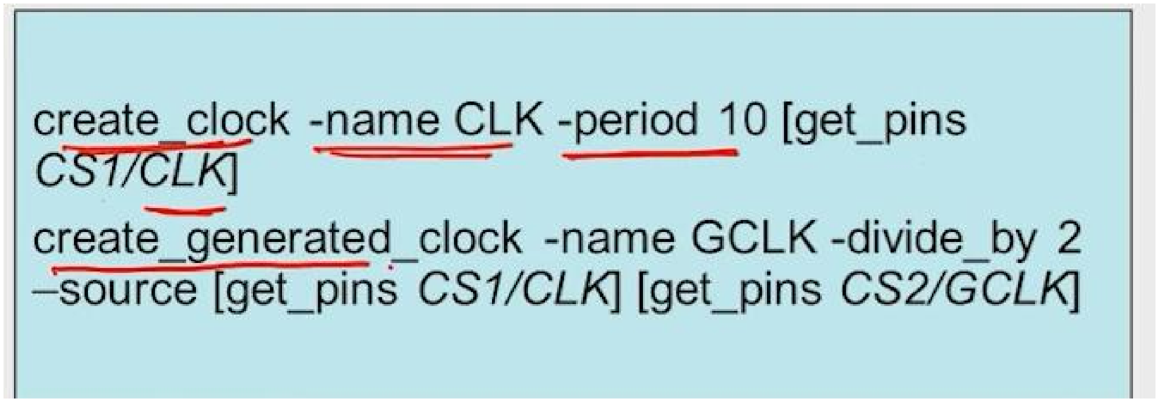


So let's take the same example that we saw earlier. This is the primary clock generator and this is the derived clock generator. We have seen that the clock frequency or the time period of the derived clock generator will be two times of the primary clock source in this case . Now let's see how we can define the derived clock source in this case. So first we have to define the master clock.

So this is the master clock CS1 which is the primary clock source. So we have to first define the characteristics of the master master clock. So we use the command create clock as we saw in the last slide and then provide a name. Name is clk in this case and the period. So time period let's say that is 10 time units and then where we are defining it. So we are defining an internal pin whose pin name is clk and the instance name is cs1.

So we write to get underscore pins cs1 slash clk . So this is the definition of the primary clock source . Then we define the derived clock source. How do we define it? We use the command create generated clock . We use the command create generated clock.

Then we use some names . So this name we can give anything but it shouldn't clash with other clock names . So we give in this case the name as gclk and then we define what is the relationship between the master clock and the derived clock. So the relationship is divided by 2 . So we saw that this is a clock divider circuit. So if the clock frequency is say 10 megahertz for clk then the clock frequency of gclk will be 5 megahertz.

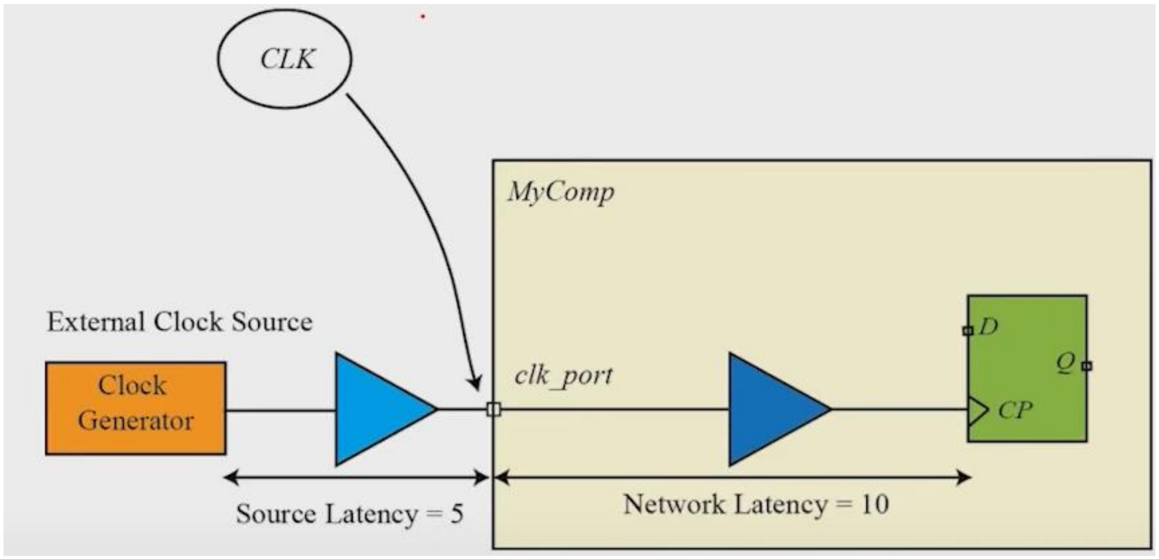


```
create_clock -name CLK -period 10 [get_pins
CS1/CLK]
create_generated_clock -name GCLK -divide_by 2
–source [get_pins CS1/CLK] [get_pins CS2/GCLK]
```

It will be divided by 2. Then we define what is the source and what is the source of this clock generator meaning that from where it will derive the characteristics of the derived clock. So we are saying manners are pin cs1 and clk . So it will derive the characteristics from this point. And here we have already defined the create clock with this command . So using this command and the source command and the relationship here the frequency and other attributes of the clock signal can be derived for this clock source gclk .

And where we want to define this clock source. So we want to define this derived clock source on the pin cs2 gclk. cs2 is the name of this instance and gclk is the name of this pin . So now we can see what will be the waveform if the waveform of the primary clock source was like this: it has a duty cycle of 50 because the waveform is not specified and the clock period is 10 time units . And the time period of the derived clock source gclk will be 20 time units . Because it is a divide by 2 the frequency is being divided by 2 the time period is multiplied by 2.

Now in addition to the frequency and the waveform of the clock signal there are other attributes of the clock signals which are important in our design. What are these attributes? One of them is the latency meaning that there is a clock source in our design now that clocks the signal that will start from this clock source. It will encounter delays in our circuit because of buffer inverters that will be in its path and also via delays and so on. So the clock that is generated at time 0 that may reach the flip flops at a later stage and how much that time is that is basically defining the latency of the clock . So we define the latency that we expect in our design using the command set clock latency.



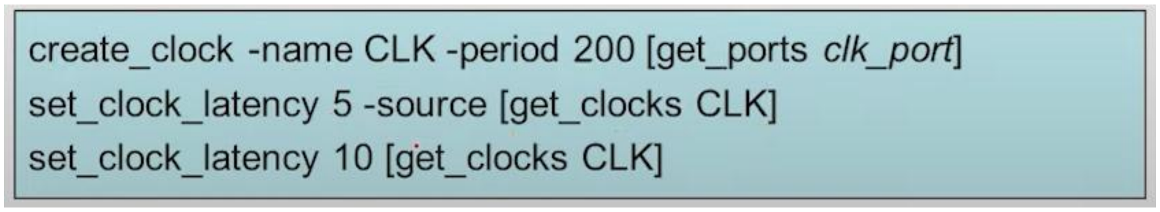
Suppose this is the design which we were making . So this is our design for which we are writing the constraints file . Now this design is receiving a clock at the port clock underscore port . So this is so our design will provide a definition to this clock at this point . Now this clock is basically coming from an external source . So there can be an external clock generator and this clock generator was in generating a clock signal or clock edge at time say 0 .

Now because of the delays in the path from the clock generator to this pin there will be some delay . Let us say that this delay is 5 units. Then we say that the source latency source latency means that the delay that the clock signal encounters before reaching our design or the before reaching before generated at the point where we have specified the clock signal. So in this case we will specify the clock signal at the clock underscore port.

So how much delay the clock signal encounters before being generated at this point. So in this case 5 that is the source latency and from this point the clock will be propagating in the in our design and it will encounter delay because of buffers inverters in the path and so and why delays. So let us consider this as how much delay it was, it is say 10 time units. So we say that in the network in our network in our design the clock undergoes a delay or latency of 10 time units. So what we are saying is that now the latency or the delay are of 2 types.

One is related to the source . So the latency the source latency refers to the delay that the clock signal will encounter at the source itself before generating and the network latencies along the path from the clock source to the end flip-flop that is the network latency and we can define this in the constraint file. First we need to create a clock, create a clock name as clk and then we define the period suppose the time period was 200 time units and then we write get underscore ports clock port . So this is a port where we are providing a create clock definition . So we define the latency for it. So we define set clock latency 5 and we say that this is a source latency minus source and so this 5 is coming from this the delay that the so that there will be at the starting point at the clock source.

So we define that using the command set clock latency and use the option minus source to define that it is a source latency and then we use a command to get underscore clocks and here we use the name that we have given to the clock. So the name was clk . So we have used here the get clocks underscore clk. So this is how the name of the clock is used in the SDC file.


```
create_clock -name CLK -period 200 [get_ports clk_port]
set_clock_latency 5 -source [get_clocks CLK]
set_clock_latency 10 [get_clocks CLK]
```

We can refer to any clock in our SDC file using the command get underscore clock . So this is the source latency. Similarly we can define the network latency as 10. Here we have not given a minus source. So by default the tool assumes that if you are not giving it as a minus source option then it is a network latency. Let us look into how we can model the unpredictable uncertainty that can exist in our clock signal .

So there can be some unpredictable deviation of the clock edges from the ideal value and these unpredictable deviations are modeled using the command set clock uncertainty . So why can this unpredictable deviation in the clock edges happen? So there can be many reasons. The first reason is related to the jitter. So because the clock signal is generated by a clock source and the clock source may not be ideal .

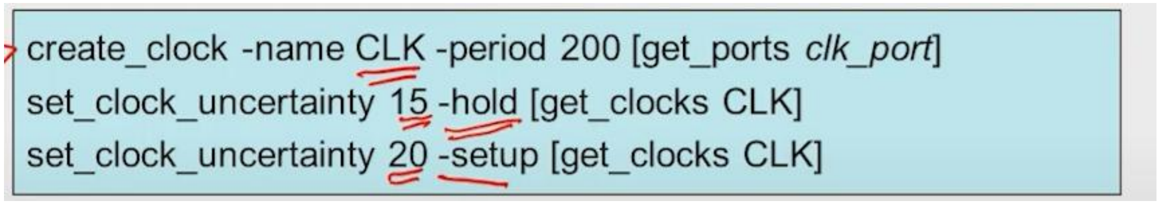
So the real clock sources can have some jitter meaning that the frequency or the time period can change slightly about what we are we are we have decided or we have we have defined in our create clock command. For example if we say that we have defined a created clock with a time period of say 2000 picosecond . So if one edge is coming at time 0 the next edge will come exactly at 200 sorry 2000 picosecond that is the ideal value of the difference or the difference between the edges of the clock . But because of jitter what can happen is that this time period may be slightly less or more for various clocks for various clock cycles .

So these are temporal variations which are unavoidable. So to model this we use a command set clock uncertainty and we say that say 2000 is the ideal time period but there is an uncertainty of say 10 picosecond. It may be less or more than that. Then what the tool will do is that it will use a more pessimistic approach of the time period to compute the timing requirements. For example if we say that uncertainty is 10 picosecond then what it means is that the clock edge next clock edge can come at say 1990 picosecond rather than at 2000 picosecond assuming that the worst case is there .

It is a pessimistic approach. So for setup analysis we can say that the time period gets reduced by this clock uncertainty and there can be other reasons for uncertainty and that can be because of the skew in our design . So ideally if we have a clock source defined somewhere and there are multiple flip flops which are being driven by this clock source ideally all these flip flops should receive this clock exactly at the same time . But what happens in really real designs in real design when we do the layout of the clock tree then at one clock one at one flip flop clock pin the next clock edge can come say at 2000 picosecond while at other it may be delayed and it may be coming at 2010 picosecond. So the difference between these two times is or that is 10 picoseconds in this case that is known as the clock skew . So if in the earlier parts of our design flow we have not yet implemented our design and if we have we have not yet implemented our design in terms of clock tree .

So the clock tree does not exist and therefore the tool cannot compute these skews on its own . So to model this expected skew we can use the set clock uncertainty command. So we say that there can be a skew of say 20 picoseconds in our design and therefore because the skew is defined remember that in the earlier discussion in static timing analysis we saw that the setup and hold requirements depended on the clock skew. So when we give this uncertainty then assuming that there will be some clock skew then the tool what it will do is that it will do a more pessimistic analysis. Remember that the purpose of STA is to ensure the safety of our design and to ensure the safety of the tool will take the more pessimistic value that we are specified .

And the third thing is related to the safety margins. In the last lecture we saw that there can be impact of process induced variations which are not easy to model and there can be other reasons for which we want to add safety margins in our timing analysis. For example when we are doing logic synthesis we don't know the delay of the wires so that the wire does not exist and therefore the exact wire delay cannot be computed. So as to may if we do not give any uncertainty in the logic synthesis then the wire delay can be assumed as 0 and the timing will be met very easily and the tool may not be doing sufficient timing optimization to such that the timing of our circuit is met . So in that case is what we can do is that we can specify some uncertainty in the clock period so that timing analysis become artificially more pessimistic than what it is with this 0 wire delay and therefore the tool can do more optimization timing related optimization in our design and in the later stages whenever the wire delay comes in our circuit when we do the physical design then the timing constraint is automatically met . So these are three major reasons why we want to add uncertainty in our design and what is the command to add the uncertainty to the clock period.

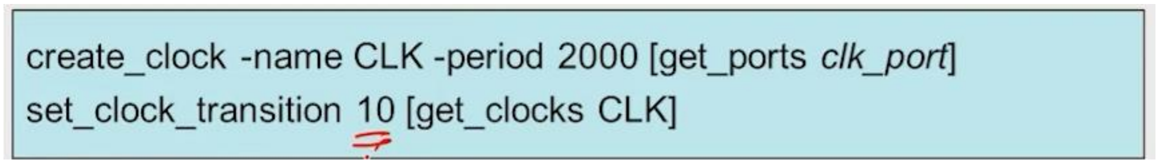


```
create_clock -name CLK -period 200 [get_ports clk_port]
set_clock_uncertainty 15 -hold [get_clocks CLK]
set_clock_uncertainty 20 -setup [get_clocks CLK]
```

So the command is set to clock uncertainty. So first we define the clock so in this case we define using say create clock the name of the clock is clk and then we say that set clock uncertainty 50, 15 and we are saying that use this much uncertainty for hold analysis and use uncertainty of 20 time units for the setup setup analysis. So what uncertainty specification will do this will make the timing analysis more pessimistic by adjusting the required time. For example in the hold analysis we have said that the hold requirement is such that the required time should be less than required time should be less than the arrival time. So if we add uncertainty then what the tool will do is that it will increase this required time it will increase it so that the analysis becomes more pessimistic. Now considering the setup analysis what in the setup analysis does is that it checks whether required time is greater than the arrival time .

So arrival time should always be less than the required time. Now when we say that 20 is the clock uncertainty so what it will do is that it will subtract 20 from the required time so that this constraint is more likely to fail and a more pessimistic view is taken and then do the timing analysis. So when we add uncertainty to the clock then the tool so that the timing analysis will become more pessimistic. Now the third attribute of the clock signal is related to the transition or how the clock signal is rising or falling. So for an ideal clock the rise edge and the fall edge all are very abrupt; the edges are very sharp and abrupt . But in real design what happens is that the clock signal propagates through our design and undergoes delays and other things.

So the waveforms get even though at the clock source the clock signal may be ideal but as it propagates it will undergo delay and distortion in the waveform and the edges may be having gentler slopes like this and so on . Now in the earlier discussion we saw that if we consider a flip flop then the setup time of this flip flop and the whole time of the flip flop depends on the slew of the clock signal . And if we assume it to be perfectly abrupt then the setup time that will be computed for this flip flop that will be artificially more optimistic it will be less . Therefore to adjust for these realistic realistic edges of the clock that will be will be there in our design we use the command set clock transition we create the clock and then use the command set clock transition for example the value is 10 meaning that the slew or the rise time of the clock clock signal is 10 picosecond rather than abrupt or 0 picosecond .



```
create_clock -name CLK -period 2000 [get_ports clk_port]
set_clock_transition 10 [get_clocks CLK]
```

So this will actually make the timing analysis more realistic so we need to use this set clock transition to model these effects.

So if you want to go deeper into the topics that we have discussed in this lecture we can refer to these references. So to summarize in this lecture what we have done we have looked into constraints and why it is used and the format of SDC and we looked into some of the basic constraints related to clocks. So in the next lecture we will be looking into other constraints such as related to input output and related to timing exceptions. Thank you very much.