VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh

Department of Electronics and Communication Engineering
IIIT-Delhi

Lecture 30
Static Timing Analysis- III

Hello everybody, welcome to the course VLSI design flow RTL to GDS. This is the 24th lecture, in this lecture we will be continuing with static timing analysis. Specifically in this lecture we will be looking at slew propagation and how we account for variations in static timing analysis. So let us first look into slew propagation. Now before moving to slew propagation let us recap what is slew of a signal. So suppose there is a signal which is rising, so we quantify how steeply or gently a signal rises by the slew of the signal.

For example, if this signal was rising very steeply then the slew will be smaller and if suppose the same signal was rising very gently as shown here then the slew of the signal will be much larger. Now in the last lecture we had seen that a stage or a timing tool needs to compute delays depending on various information or using various information. For example, if there is an inverter which is driving an interconnect and an input pin right this forms a stage. Now if we want to compute the delay of this inverter then we need to know what are the characteristics of the incoming signal, meaning whether it is rising fastly or steeply or gently .

So we need to know the slew of the incoming signal right based on that and the library information and other information of the interconnect the delay calculator will compute the output slew of the stage right. So now since for computing the delay for a stage the slew is very important the slew must be propagated in our design meaning that given a stage suppose this inverter was somewhere in the middle of our design. So suppose there was a design and our inverter was lying somewhere in the middle in the so we know the slew of the incoming signal to our design through the constraints file but we also need when we want to compute the delay of this inverter we also need to know the slew of the signal at the input of this inverter right and that is the motivation of carrying out slew propagation. So static timing analysis internally needs to propagate the slew from the input ports to all the stages in the fan out in a sequential manner and this is what is referred to as slew propagation. Now the propagated slew can be different through different combinational paths.

So let us take a very simple example a very simple example suppose there is a there is a NAND gate and suppose there are three inputs associated with this NAND gate. Now there are three arcs timing arcs for this NAND gate right the first timing arc the second and the third right. Now whenever the income signal will come through one let us call the in the input pins of the R of the NAND gate as A B and C. So when the signal will come to the pin A then the the that signal will encounter some delay and also have some output slew that will be exhibited at the output pin say Z of this of of the of this NAND gate right. So this output slew means that how fastly or or how gently or how steeply the output will be rising or falling depending on what the what the what the type of signal is there at the input pin right.

So for the arc from A to Z there may be some output slew associated with it meaning that the output at Z will be rising with some slew right. Let us say that if the slew was 10 picosecond right. Now for the other arc from B to Z also there will be some output slew depending on the characteristics of the timing arc B to Z and what incoming slew was there at the input pin B right. So let us assume that the slew form of output slew for the case for the timing arc B to Z is say 15 picoseconds. And similarly there will be an output slew for the arc C to Z and let us call it as say let take a value 8 picosecond right.

So what it means is that the propagated slew can be different through different combinational paths in one cell itself right. Now the problem is that now given that the output Z can have 3 different sluices from 3 different paths which slew to actually store for the subsequent stages. For example there may be another NAND gate which is driven by this NAND gate. It may be again say 3 input NAND gate or it may have more than 3 inputs or even less than that or the characteristics of this or the or the functionality of this driven gate may be different than what it will be what is shown here right. So now to compute the delay of the subsequent stage in this case another NAND gate so let us call this P Q and R these are the pins for this NAND gate.

To compute the delay of the arc P to Z again the input slew must be known right. Now what should the input slew be taken for in this case because there are 3 inputs and output slew associated with the pin Z right that is 10, 15 and 8 in our case right. Now if one solution could be that we store all 3 right but then the problem is that if there are multiple stages the number of sluice that we need to store will go on simply multiplying. For example if there are other drivers for the input pin Q right here also there will be say 3 suppose this was having say 3 different input pins then here also will have 3 different 3 different sluice right and similarly there will be 3 different sluice at the point R for the pin R right. So for the gate say let us call this gate as G 2 for the gate G 2 the output slew will be if each P Q R has got 3 different 3 different sluice then the total sluice that needs to be stored at the output of of G 2 will be 3 plus 3 plus 3 that is 9 right.

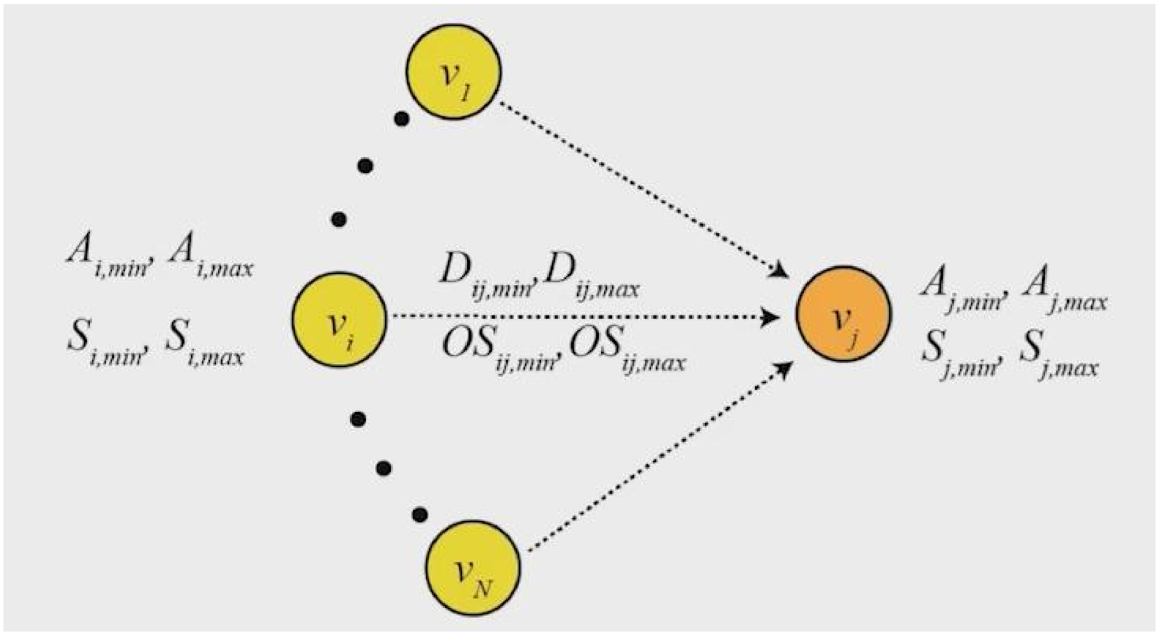
So it will go on simply multiplying as the number of stages goes on right. So simply storing all the slew that are possible for the arcs within a cell is not possible because the memory requirement for slew storage and propagation will simply be too high and computationally not possible right for practical circuits. So in that case we have to use some other technique of storing and propagating a sluice in our design in static timing analysis. So in this lecture we will be looking into how we can do the slew propagation more efficiently. So when we consider slew propagation fortunately there is a property of CMOS logic gates which allows us to efficiently manipulate sluice in slew propagation and what is that property? So the property is that the delay and output slew are typically monotonically non-decreasing function of the input slew meaning that if there is there are there are there is a timing arc between say pin A to Z then as we increase the slew at the input pin right suppose the input slew was say 10 picosecond initially then there will be some output slew and delay for example, the the output slew will output slew will be say 5 picosecond and the delay is say suppose 50 picosecond right.

Now if we increase the input slew the if the if we increase the input slew say from 10 picosecond to 20 picosecond at the input pin A then the output slew will not decrease it is a monotonically non-decreasing function it will be 5 picosecond or maybe some somewhat higher it will be say say 8 picosecond and similarly the delay will be non-decreasing it will be either 50 picosecond or it will be more right. So it will be say 80 picoseconds for an input slew of 20 picoseconds right. So this is what we mean by saying that the timing arcs, the delay and the output slew for a timing arc is a monotonically non-decreasing function of the input and what is its implication? The implication is that it allows us computing and storing and propagating only the minimum and maximum value right. So what it means is that suppose as in as in our earlier case the the slew at the from A to Z was say 10 picosecond from B to Z it was 15 picosecond and from C to Z it was 8 picosecond. So rather than storing all three values rather than all three values we can just store the maximum value and minimum slew at the output Z right.

So we will store only the minimum that is 8 and the maximum which is 50 and by just storing the minimum and maximum output slew at a at a at a at a output pin or at a vertex in a timing graph we can ensure that the the the bound on the delay can be computed in the subsequent stages right. So just by using 8 and 15 if it is driving some other logic gate in its fan out we can compute the bound on the delay of that why because delay for the circuit elements in the fan out also those are non-decreasing functions right. So if we compute delay corresponding to 8 and delay corresponding to

15 then the delay for any other slew that will be at the output pin Z will be within that range within the range given by 8 and 15 right. So because of this property of CMOS logic gates that the delay and output sluice are monotonically non-decreasing functions we can obtain the bounds on the delay and output sluice using the bounds on the input. So we can if we store only the bounds that are sufficient to ensure that the delay is within some bound and as such and similarly the arrival time will also be within some range right.

So note that the purpose of static timing analysis is to ensure the safety of your circuit and if we are and if we are able to derive the bound on the arrival time and the delays will be able to ensure the safety of the circuit just by the bounds right. We do not need the exact number and in slew propagation we utilize this property of just keeping the bounds on the sluice stored at the vertices and propagating the bounds that is it and by propagating the bounds will be able to derive that what is the bound on the arrival time and based on that we can derive that whether our circuit is meeting the setup requirement and hold requirement and so on.



So now let us look into that how the bounds are compute right so let us come let us assume that there is a timing graph and v j is one of the vertices let us call this is the output vertex and then these are the incoming vertex v 1 to v 1 v i to v n there are n incoming vertices right. Now for each edge we will have for now at each vertex is in addition to the arrival time bounds we also store the bound on the sluice right what is the minimum slew at the vertex i that is shown as s i min and what is the maximum slew at the incoming vertex v i is shown as s i max right. So in our last lecture we had seen that how do we derive the bound on the arrival time right so the formula was a j min the minimum bound on the arrival time at the vertex v j is the minimum of the a i min plus d i j min right.

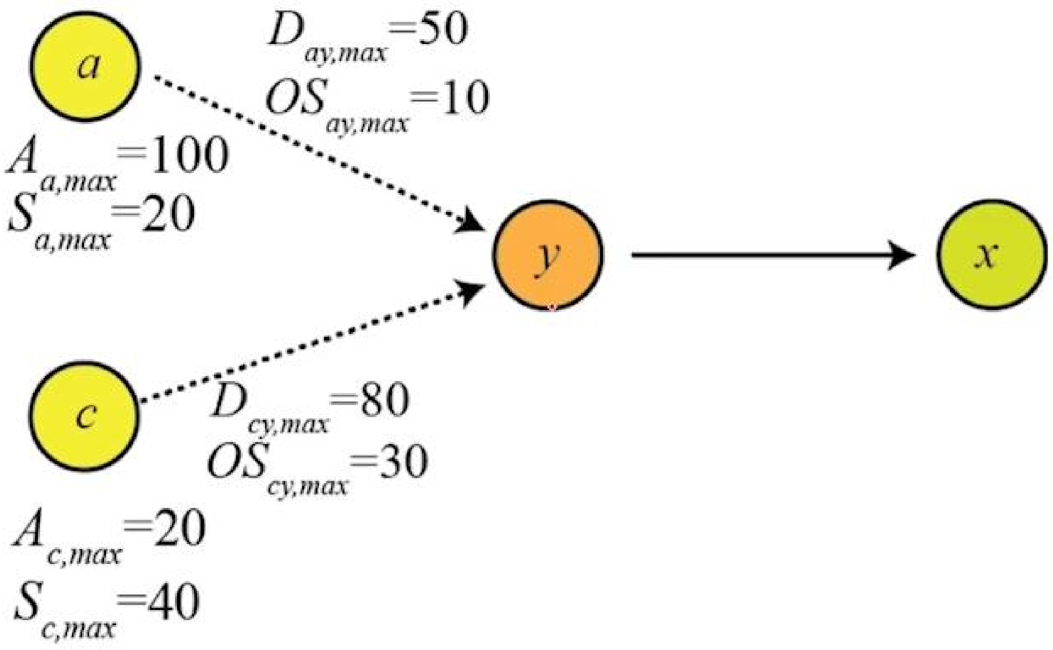
$$A_{j,min} = Min[A_{i,min} + D_{i,min}]$$

$$A_{j,max} = Max[A_{i,max} + D_{i,max}]$$

So the minimum so each vertex will have each edge its own a i min so we take that and add it with the minimum delay right so each delay will Sorry each edge will have associated delay right. Now this associated delay will be with respect to two bounds of the slew right one will be with respect to the lower bound on the slew and the other on the higher bound on the slew and that is the those two bounds are shown as d i j min is the lower bound on the delay obtained by lower bound on the on the input slew and then dij max corresponds to the delay obtained when the input slew was the maximum that is s i max at v i. So each edge will now have rather than one delay two delays one corresponding to the lower bound on the input slew other corresponding to higher bound on the input slew. Similarly the the the there will be output slew bounds for each of the timing arc right from the timing arc between v i and v j I am showing the output slew as o s i j min so this i j o s i j min is corresponds to s i min right that the delay obtained when the the when the when the in the input slew was s i min right and o s i j max corresponds to the output slew when the incoming vertex v i was having a slew of s i max right. So, each edge will now have rather than one delay number two delays number corresponding to two two bounds on the input slew similarly the output slew will not be one number but two numbers corresponding to two bounds on the on the on the input slew right.

Now given that each edge has got two delay numbers and two output slew numbers how do we derive the arrival time bounds. So, the arrival time bound is obtained by taking the minimum arrival time of a given vertex, add it with the minimum delay right and out of all the incoming vertex like v 1 to v n take the one which is showing the minimum value that will give us the minimum arrival minimum bound on the arrival time. Similarly the maximum bound on the arrival time can be obtained by adding i i i i i i i max corresponding to a vertex with d i j max with that for that edge corresponding to this vertex and then out of all the n input vertices take the one which is showing the maximum right. So, this is how the arrival time bound on the arrival time will be derived. Now similarly we can derive the bound on the on the on the output slew for the vertex v j and how do we derive it we simply out of all the incoming edges n incoming edges take the one which is exhibiting the minimum slew that is the lower bound on the on the on the on the slew for the vertex j right.

Similarly out of all the n incoming vertex we take the one which is showing the maximum slew maximum maximum slew and that is gives us the upper bound on the slew at this vertex ok. So, this is how we do the slew propagation from the input side that is all the v i s nodes to the output right. Now let us take an example to illustrate this ok.



So, for simplicity let us consider only the maximum bound right the upper bound on the sluice and delays and arrival times and others and similarly it can be extended to the lower bound also but for simplicity let us first take the case for the maximum maximum slew propagation ok. Let us consider that there are three vertices in the timing graph a c and that these are the incoming vertices for the vertex y and then this y is driving another vertex x right.

Now let us assume that the maximum arrival time at the vertex a is 100 maximum arrival time at the vertex c is 20 right and the input slew at the vertex a is 20 and the input slew at the vertex c is 40 right. Now corresponding to this input slew of 20 the delay that is that that that delay corresponding to the arc a to y is 50 and the output slew is 10 right and for the so these are the given data assume that these these exist in a circuit then we will derive that how the slew will be propagated to the vertex y right. Now for the arc between c and y the delay is 80 picosecond for the input slew of 40 picosecond and the output slew is 30 picosecond let us assume that this is given. Now what will be the arrival time at the vertex y and what will be the slew maximum slew at the vertex y ok let us compute that. Now for the if we consider this this edge right so the arrival time is 100 and delay is 50 right so the arrival time corresponding to this this edge is 150 right 100 plus 50 150 for the other case from c to y the arrival time is 20 plus delay of 80 that is 100 right so this is 150 and this is 100 so if we take the max of it we will get 150.

So the arrival time at the vertex y will be 150 right now what about the slew now for the edge a to y the slew is 10 and for the edge c to y the the the the slew is 30 now out of 10 and 30 what is the maximum the maximum is 30 so we take the output slew as 30 right so this is what what the slew will be propagated at the vertex y ok.

- $A_{y,max} = Max[\underline{100+50}, \underline{20+80}] = \underline{150}$
- $S_{y,max} = Max[10,30] = \underline{30}$

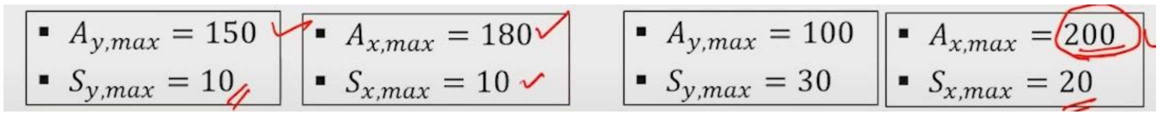
Input Slew S_y	Delay D_{yx}	Output Slew S_x
10	30	10
30	100	20

Now let us compute compute that what will be the slew and the arrival time at the vertex x now to know that we need to know the characteristics of this edge the edge between y and x so let us assume that this is the characteristic given characteristic that will be the that is in the library which is shown here give meaning that if the input slew at y is 10 then the delay of this arc is 30 right delay of the this timing arc is 30 and the output slew is 10 and if the input slew is is 30 at y then the delay is 100 between y to x and the output slew between y to x is 20 let us assume that this is given in our lab now with this information now we can do a slew the compute the slew at the vertex x and also the arrival time right so what will be the arrival time so the arrival time is very easy because the arrival time at y is 150 and the output slew we know is 30 now corresponding to 30 we have to read this row right now in this row the delay is 100 right so we take the delay of this arc between y to x as 100 right so the arrival time at y was 150 we add 100 to it we get a arrival time at x as 250

and what is the output slew at the at the vertex x it will be simply we take the read this row and we get the value as 20 so we get the arrival time at this point as 250 right and the maximum

- $A_{x,max} = 150 + 100 = 250$
- $S_{x,max} = 20$

slew as 20 right so this method of slew propagation based on the formula that we saw in the last slide and as illustrated this is known as graph based analysis so we are doing we are computing the delays and the arrival times based on the worst case slew at a point and based on that we are propagating but note here that there is a problem the problem is that we have computed the arrival time at this vertex y through the path a to y right that was showing the worst arrival time 100 plus 50 we have taken as 150 not the other way path right but the slew that we took at the at the point at the at the vertex y that was coming from the edge c to y rather than a to y right so the arrival time is from one edge while the maximum slew is propagating from the other edge right that is what what is what is done in in this calculations right thus the output slew was taken as 30 from the edge c to y not the other right now this is somewhat pessimistic why it is pessimistic because we are the the the that signal will either propagate through a to y or through c to y right that one of them will show the worst behavior right it mean it is not that the the arrival time behavior will be worst for one one path and the slew behavior worst will be shown by the by the other path simultaneously that cannot be the there right so the only one of these two paths will show the worst worst behavior right so let us do a more more more realistic analysis by first so the correct way or more realistic or analysis is to consider the slew propagation and arrival time propagation first through the path a to y to x and then through the path c to y to x and then through the path c to y and then take the worst of the situation that is visible at the vertex x that is the right so now let us do the right way and see how the computation changes. So if let us assume that the slew and the and the arrival time both are computed through a to y right so if we compute see or see the path from a to y right so the arrival time will be 100 plus 50 that is 150 and the slew at y will not be 30 but only 10 right that is what is exhibited by the the edge a to y so the slew will be here as 10 and the what will be the the arrival time at x and the slew at x now since the the the a slew at this point is 10 we have to read the other row in this table right not the lower row but this row right so if we look into this row then the delay is 30 right so delay of this edge is 30 so the arrival time at x will be 150 plus 30 that is 180 and the slew that will be that will be obtained at x will be corresponding to this row and it will be 10 right so this



- $A_{y,max} = 150$
- $S_{y,max} = 10$

- $A_{x,max} = 180$
- $S_{x,max} = 10$

- $A_{y,max} = 100$
- $S_{y,max} = 30$

- $A_{x,max} = 200$
- $S_{x,max} = 20$

is the this is the correct arrival time if the signal propagates through a to y to x now let us take the other case right let us take the other case that is the signal is propagating through c to y and then to x so what will be the arrival time in this case so the arrival time at y will be 20 plus 80 that is 100 and the slew will be 30 corresponding to this edge the slew will be 30 now since the slew is 30 we have to read this row right in that in this row in the in the in the table right now for the 30 the delay of this edge is 100 right delay is 100 and output slew is 20 so the arrival time at x now will be 20 plus 80 100 plus the 100 for this edge that will become 200 right and the output slew at the vertex x will be 20 corresponding to this edge right so the we see that the if we compute the arrival time and the slew exhaustively through these two paths we will get the value as arrival time as 180 or 200 and out of them which one is worse the worse is the 200 or the more pessimistic is the 200 right so the real bound on the arrival time is 200 and not 250 that we computed in the graph based analysis right so graph based analysis is safe meaning that the bound that will be given by the the the graph based analysis will be safe because it is not only taking the arrival time as worse but the slew as also the what worse could be right in our circuit and therefore the bound that will get using the arrival using the slew propagation by graph based method will always be safe right but it may not be tight as illustrated in this example because the real maximum arrival time in our circuit is 200 at the vertex x not 250 so what it means is that if we want to do so what typically STA tools do is that they do timing analysis based on GPA or graph based analysis why because in that case it needs to only store two values of slew at each vertex one maximum one minimum for one type of transition right and therefore it is very efficient and based on that it can do the timing analysis if the design is is passing the timing analysis based on GPA then of course our design is correct right so the timing analysis based on GPA then of course our design is correct in terms of timing there is no problem at all right but the bound that we get through GPA may not be a tight bound it will have some pessimism and to remove that pessimism what we need to do we need to do analysis path wise right we have to analyze the path from say a to y to x and then c to y to as y to x and then take the verse of them right this is just an example here only two paths were involved but in realistic design there will be lots of paths right so we cannot exhaustively do the timing analysis path wise right path wise because it there are too many paths in our circuit so what typically design STA tools or commercial STA tools do is that they provide some mechanism to carry out path based analysis also right so this is known as path based analysis or PBA so by default the tools do what is known as GBA and that the timing analysis done through GBA is always same right but it may contain some pessimism as a result we may get some artificial timing violations and to remove those artificial timing violations we may we can do what is known as path based analysis take the path that is failing do a timing analysis do the timing analysis for that particular path and see that whether the constraints are whether the timing requirements are met or not so that will remove the pessimism of GBA so that will remove the pessimism of GBA that will remove the pessimism of GBA and we may be easily we may be able to close the design ok. So now let us look into another important feature of static timing analysis that is to take into consideration the variations that can happen in our design so why do we need to take into account variations so the because of many reasons for example, because of say process induced variations we are say fabricating a line which is expected to be say of thickness one one macron and the and the and because of the fabrication or the randomness in the fabrication process there was some finite difference in the thickness of that line or the wire or interconnect and as such the property of the interconnect can change similarly there can be process induced variation in the devices and the transistors also right so there can be process induced variations and there can be fluctuations in temperature the environment and the voltage in our design right because of these fluctuations so this process voltage and temperature these three combined together these are known as PVT P for process V for voltage and T for temperature. So PVT variations can be there in our circuit so because of this PVT variations the behavior or the properties of the transistors in our circuit and also of the interconnect those can change right and because of those changes what can happen is that the timing attributes of the devices and the timing attributes of the interconnects those can change for example, delay the nominal delay was say 100 picosecond but after fabrication or because of other fluctuations in the temperature or voltage the delay is now say 110 picosecond.

So now because of these variations in the delay what can happen is that the constraints that were earlier passing that can fail right because we did timing analysis based on our setup and hold analysis or we check the inequalities whether that inequality was being satisfied or not for set up case and the hold case and based on that we closed our design or considered our design to be timing safe. But after fabrication there can be because of process induced variations the delay and other parameters of the circuit can change and the inequality which was earlier valid for our design may now become invalid and if that happens then it can result in failure of our circuit. So in static timing analysis we need to take into account this and take corrective measures that despite some variations our circuit is tolerant to it and does not become does not have any timing problem because of these variations. So to tackle variations different techniques are used in static timing analysis. So these techniques differ in how accurate the modeling is, how much effort we need to take into account the variations in design and what is the computational requirement for some of the algorithms that will run to account for these variations in static timing analysis.

So therefore various kinds of methods are used in design flows and we will be looking at some of these techniques in the subsequent slides. So, the easiest way to account for variations is to add safety margins to our circuit. So, what we say is that suppose the timing requirement was that to arrive for setup requirement the required time should be greater than the arrival time right. So, suppose the required time was say 1000 picosecond. So, it means that the delay of the signals and arriving at the deep end of a flip flop should be such that it is less than 100-1000 picoseconds right.

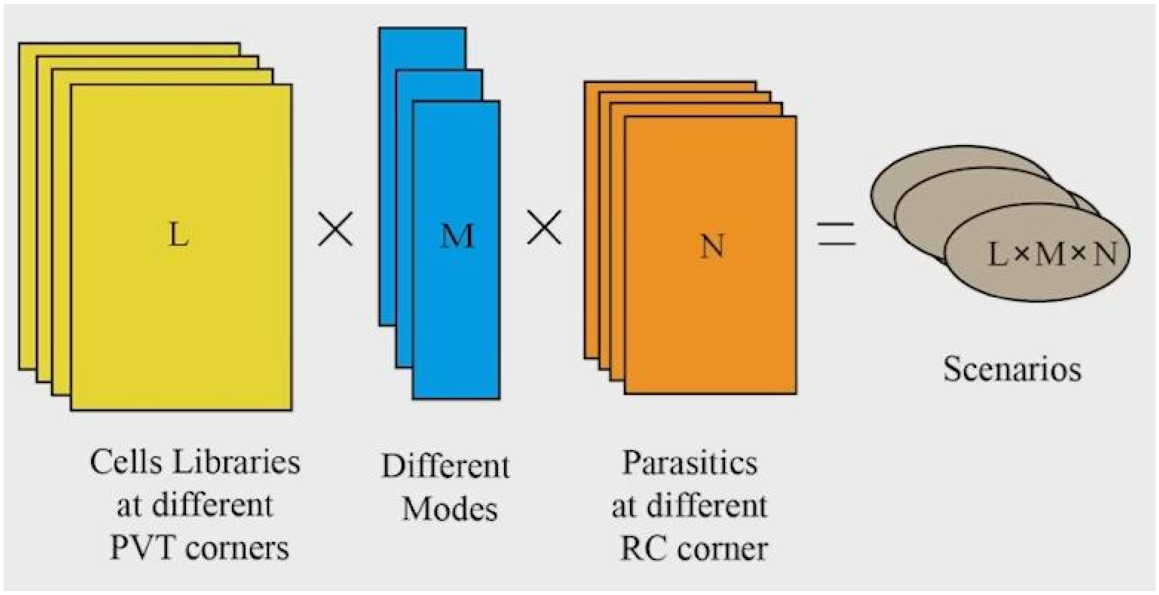
Now by adding margin what we mean is that we make the constraint more pessimistic we say that ok instead of checking for the inequality one arrival time is less than 1000 picosecond let us check for the inequality that arrival time is less than 900 picosecond. So, we have added an artificial pessimism of 100 picoseconds that is a safety margin that even if the delay varies by say 100 picoseconds our circuit will still be safe right. So, this is the very easy method of adding or ensuring the or accounting for variations in timing analysis we add some margin. So, say in this case the margin was 100 picosecond. So, we can convey these margins to the static timing analysis tool through the constraints file the SDC file that we write in. We can use some constraints to specify how much extra margin we want in our timing analysis.

So, when we add those constraints in our SDC file then the required time will be adjusted appropriately to make it more stricter. So, in this case the if the the extra margin was 100 picosecond the then for setup analysis the required time will be subtracted or one or 100 picosecond will be subtracted from the required time and the new required time that will be checked by the tool will be 900 picosecond instead of say 1000 picosecond. Now, the problem with this technique is that how do we come up with a good number right from where we get say 100 picoseconds in the first place. So, we have to do some based on past experience we can come up with some number, but if we make it very pessimistic because may be say instead of say 1000 picosecond sorry 100 picosecond as the safety margin if we add a safety margin of say200 picosecond then what problems are we and we will what problem will be we will encounter in our design. So, the problem will be that there can be some loss in P P A why because to maintain a tighter tighter timing requirement we may need to change the implementation.

For example, we might need to use a larger NAND gate or inverters or logic cells in our design right to meet the timing requirement or stricter timing requirement that is coming after adding the safety margin. So, if we add a large margin then it is likely that there will be more area overhead and also the power overhead can also be larger because in this area and power are very much correlated in our design. So, we should not make the time the timing margin very very broad or overly pessimistic. And the problem with small margin is that if we say instead of 100 picosecond we add a margin of say 10 picoseconds right. So, that is in that case the area penalty will not be there in our design, but the problem is that if the delay variation was more than say 10 picosecond it was say 20 picosecond then the earth circuit will fail right and therefore, there there are there will be chances of timing failure and the yield loss.

So, as a designer we need to consider these two trade offs while defining the margin in our circuit right. So, typically what is done is that these margins are not a very good method of adding and are not a very accurate method of modeling or accounting for variations in static timing analysis. So, typically we use these margins in earlier stages of the design flow. For example, during the logic synthesis step when we can we can use this margins because at the at those stage we do not know even the delay of the interconnects right because that we have not yet laid out the wire and therefore, the delay of the interconnects at their capacitances are also not known. So, there is lots of uncertainty in our design and therefore, in that case since we do not have much information we can use these safety margins to do the synthesis and do timing analysis in the initial stages of the design flow.

And as the design flow progresses we can reduce those margins and may eliminate most of those margins in the later stages of the design flow. The other method to account for variation is what is known as multimode multi corner analysis or MMMC analysis. So, what is done in multimode multi corner analysis is that we carry out STA at some discrete set of scenarios. So, we do a timing analysis for some discrete set of scenarios and what are to account for variations and what are these scenarios. So, these scenarios are derived by the combination of three things.


L
×
M
×
N
=
L×M×N
Scenarios
Cells Libraries at different PVT corners
Different Modes
Parasitics at different RC corner

So, what are these three things? The first one is the PVT corners for technology libraries. So, typically we use this method to account for global variations, global variations in our design meaning that because of the process induced variation the characteristics of all the devices in our circuit are shifting right. So, that is what is known as global variations all are making. For example, all the devices in our circuit are fabricated such that they take the worst delay or the maximum delay or the best delay or the typical delay right. So, if the variations are of global types then what we do is that we have libraries not only of one we do not have only one library, but we have a set of libraries for various corners. Maybe we will we can have a library for the worst case meaning that the the dot lib file for the worst case will have the timing attributes of all the cells in the library for the worst case meaning that that the delay will be larger for for or will be will be the highest or the worst for the various ah timing arcs in our design for setup analysis right.

Similarly, there can be the best best best library library for the best case or there can be a library for the typical right. So, there will be a set of libraries for different corners and we will use those sets of libraries to derive the scenarios. How we will derive the scenario we will just see and then do the timing analysis. So, the first thing is the set of libraries for various PVT corners and the second thing is the multiple modes of our design. So, our design may be working in various kinds of modes for example, it can work on in a functional mode meaning that it is doing the function carrying out the normal functionality of our design or it is working in a test mode when it is doing for say going for DFT or it is in the sleep mode. For example, if we want that our chip is not doing some active computation to save power we can take it to a sleep mode or we want to do a very high performance job or the requirement is that a chip deliver the maximum performance then there can be say a turbo mode or there can be various modes in our shell.

So, in these different modes what are the basic differences? So, differences will be with respect to say the clock frequency the clock frequency in the turbo mode will be the maximum and in the sleep mode it will be minimum or some controlling signal will be taking some value. For example, in the sleep mode there will be some signal called sleep which will be high right which will make most of our circuit as switched off or so on. So, there will be some controlling signal which will take a constant value or and the clock frequencies and other things can change in our design for different modes and those kinds of effects are taken into account by writing different SDC files. So, there will be an SDC file or the constrained file for the functional mode, another SDC file for the test mode, another for sleep mode, another for turbo mode right.

So, these different SDC files define various modes in our design and then the third one is the RC corners. So, RC corners so in the last lecture we saw that the parasitics or or or or the interconnect parasitics are important for computing the delay of a stage and those parasitics are extracted using a tool which is known as a parasitic extraction tool. Now the characteristics of this parasitics for example, a metal wire is very much dependent on the on the on the how it is fabricated. For example, there will be small deviations in the thickness in the width of these lines and because of those the RC values or the capacitance and resistance values of the interconnects can change. So, to account for that what we do is that we do not extract one parasitic file for our design, we extract multiple parasitic files for different what is known as RC corner R stands for resistance C for capacitance.

So, for various RC corners we extract different SPEP files or standard parasitic exchange format files which contain the parasitic information of the interconnect. For example, there can be a parasitic extra any SPEP file for the minimum capacitance case for another can be for the maximum capacitance case and so on. So, using these three three things or the PVT corners multiple modes and RC corners we derive scenarios. How do we derive it? Let us take an example: suppose in our design there were three corners three corners and and three dot lib files one was say slow dot lib the other was fast dot lib and third one was say typical dot lib and there were say four modes in our design. So, we have funcs, func dot sdc for functional mode then we have say test dot sdc for the test mode and then we have say sleep dot sdc for the sleep mode and a turbo dot sdc for the turbo mode.

Similarly, suppose there are four parasitic extraction corners. Maybe one is C worst capacitance worst dot SPEF then C best dot SPEF and then say RC max dot SPEF and RC min dot SPEF. Now, these are discrete sets of variations that we want to analyze in our design. So, using these three three corners four modes and four RC corners we can derive three into four into four that is forty eight scenarios by combining for example, slow dot lib then func dot sdc and C worst dot SPEF and so on. So, if we take each combination of this and then we can derive say forty eight scenarios out of it. So, one of the ways is to analyze or perform STF forty eight times for each of the scenarios, but that will be inefficient.

So, rather than doing forty eight different STA or timing analysis what we do is that we carry out static timing analysis using what is known as multimode multi corner analysis simultaneously for all the forty eight scenarios. And as a result what will become more efficient why? Because we need for the forty eight scenarios that we got we need not analyze all of them. If one of them is dominating the other, see if the purpose of static timing analysis is to ensure safety. If the safety of one scenario can tell us that what conditions for others for other scenarios need not be evaluated then we are still safe by just analyzing one of the scenarios out of them. So, what MMMC analysis does is that it avoids computation of dominated scenarios.

If one scenario is dominated by some others then that dominated scenario need not be evaluated separately and therefore, computation can be received. Also the parallel processing can be invoked and all these corners can be sorry all these scenarios can be evaluated in parallel using multiprocessing or parallel processing and the and and we can efficiently derive the safety of our design for all forty eight scenarios rather by doing MMMC analysis rather than explicitly analyzing the analyzing and explicitly analyzing separately for all forty eight scenarios. So, commercial tools typically give us flexibility to do this kind of multimode multi corner analysis in the design. And this method is one of the most popular methods of accounting for variations in our design. Now the method that we saw for MMMC analysis that is effective if there are global variations meaning that all the devices are being affected in one particular manner all are showing the say worst delay or best delay or so right.

So, that is kind of a global variation. But we also need to account for local variations in the properties of devices and interconnects. Why? Because when we fabricate our devices, say transistors and our chip contains millions of transistors and lots of interconnects then there will be some local variations in their properties because of random effects. And we cannot avoid that we have to tackle that in our design. So, to tackle these local variations we do that we define on chip variations to delay right. So, we say that there are some delay factors associated with the delays and we are making our analysis somewhat pessimistic by applying this derating factor.

So, what when we apply when we supply derate value to the timing analysis tool then what it does internally is that it computes effective delay. And how does it compute an effective delay? It is the effective delays computed by multiplying the nominal delay with the OCV derating factor that we have given for our design right. Now this OCV derating factor can be based on say path bounds meaning that we can say that for early a some derating factor is used and for late paths some other derating factor is used right. Similarly we can say that for data paths use some derating factor, for clock paths use some other derating factor right. So, we can make these derating factors selective based on the path type or path bounds and also delay type. Maybe we want to derate the gate delay by say 10 percent while interconnect delay we want to derate by just by say 5 percent or.

So, those kinds of things typically a a a STA commercial STA tools provide us those facilities that how we can define derate for different situations or different types of path or different types of timing arcs. And differently we can also define over different OCV derating factors for say best case or worst case or typical case we can specify different numbers for that. So, let us take an example of how we specify the derating factor and what will be its impact on timing. So, let us say that when we are running a tool we have specified that for the late path the derating factor is 1.

1 and for the early path the derating factor is 0.9. So, what it means is that for the late path suppose the delay was 100 it will multiply by 1.1 and the delay will be taken as 110. So, this is the derating factor and for the early path suppose the delay was 100 then the derating factor is 0.9 then it will take a 9 the delay as effective delay as 90. Now, this kind of a specification will impact the setup analysis and hold analysis differently.

So, in setup analysis what are the late paths. So, the late paths are the paths which we use for the data path and the clock launch path right. So, those are the late paths, that is what if we want to if we consider the constraint for setup analysis then if the data path and the clock launch path those become late then the design is more likely to fail right. And therefore, this data path and clock launch path are taken as late paths and the derating factor of 1.1 will be applied meaning that if there was an AND gate on the data path and its nominal delay was 100 then that AND gate's delay will now be taken as 110 rather than 100 right.

So, it will be multiplied by the derating factor of 1.1. However, on the clock path capture path the delay will be decreased by 0.9. For example, if we have a say flip flop this is the launch flip flop and through combinational logic it is being captured by a capture flip flop. And there is a say inverter or a chain of inverters on the capture flip path. So, if we want to do a setup analysis then if we decrease the delay of the circuit element on the capture clock path then the analysis of the circuit will be more likely to fail right.

So, suppose the delay of this inverter was 100 picosecond right. So, because the early part derating is 0.9 the delay of this inverter will now be taken as 90 it will be multiplied by 0.9 right and it will be taken as 90 as the effective delay right. Similarly, for the other inverter it will be directed by a point factor of point right. Now, in the whole analysis similarly, for the data path and the clock launch path and those become the early path right and if it is derated by 0.

9 Then it is more likely to fail right. And similarly, for the clock capture path that is multiplied by 1.1 because then the circuit will be more likely to fail right. So, whole for whole analysis for the data path becomes the early path right and therefore, for the the derating factor of 0.9 is 0 right. So, if we want to look more or on these things you can refer to these references that I have shown here.

Now, to summarize in this lecture what we have done. So, we have looked into two important aspects of static timing analysis. The first is how slew propagation is done and we looked into two modes of slew propagation. The GBA method which is computationally more efficient and widely used, but not but may not give a tighter bound or may contain some pessimism and other is the path based analysis based slew propagation which is in which the slew is propagated depending on the path we are analyzing. So, the arrival time and the slew will be propagated through the same path right and it is more accurate right and then we also looked into some of the techniques to account for variations in our static timing analysis.

So, in the next lecture we will be looking into the constraints. So, in the earlier lectures we have discussed that constraints is the basic information from which static timing analysis infers the timing constraints that we want our circuit to follow or obey. So, in the next lecture we will see how we can write those constraints and convey that information to the static timing analysis. Thank you very much.