VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh

Department of Electronics and Communication Engineering
IIIT-Delhi

Lecture 29
Static Timing Analysis- II

Hello everybody, welcome to the course VLSI design flow RTL-2 GDS. This is the 23rd lecture. In this lecture, we will be continuing with static timing analysis. In the previous lecture, we had looked into how to model the setup requirement and hold requirement in a synchronous sequential circuit. In this lecture, we will be looking into how these requirements are checked or what is the mechanism of static timing analysis. Now, why do we want to look into the mechanism of static timing analysis? So, the motivation is that if we understand the mechanism of static timing analysis, then we can easily analyze or interpret the results that are produced by the static timing analysis tools.

And then we can make very efficient changes in our design to fix timing problems or improve the timing of our design. Now, before moving into the mechanism of STA, let us first get familiar with some important definitions that are related to the ST. So the STA considers two types of paths, data paths and clock paths. Now what are data paths? So data paths start from timing start points.

And what are timing start points? So timing start points can be input ports of our design or timing start points can be the clock pin of the flip flops. So the input ports of our design and the clock pin of our flip flop, these are considered as the start points of the data path from the perspective of static timing analysis. And then this path can go through the combinational circuit elements, only combinational circuit elements, no sequential circuit elements such as flip flops are allowed in the data path. So it starts from timing start points, then goes through combinational circuit elements; only the number of combinational circuit elements can be as many as possible. And then the data path ends at the timing end points.

And what are timing end points? So timing end points are the D pin of the flip flops or the output ports or in general the points where we are actually doing the or checking the setup and hold requirements. So it is very clear that why do we want to do setup and hold require or why do we want to check setup and hold requirement at the D pin of the flip flops. Why do we want to also make some checks of setup and hold requirements at the output ports? The reason is that

the signal that goes out from the output port will be further sampled by some flip flop external to our circuit. And to ensure that the timing and hold requirements of the external flip flops are also met we need to make some checks on the output ports also.

So we will look into this aspect of checking at the output ports in more detail in the subsequent lectures. So the time to summarize what is a data path from the perspective of STA. Data path is a path which starts from either the input port or the clock pin of the flip flop and it goes through a sequence of combinational circuit elements and finally it ends on the timing end points where the checks are made of the setup requirement or hold requirement and setup and hold requirements are typically checked at the D pin of the flip flops and the output ports. So let us take an example. Consider this circuit.

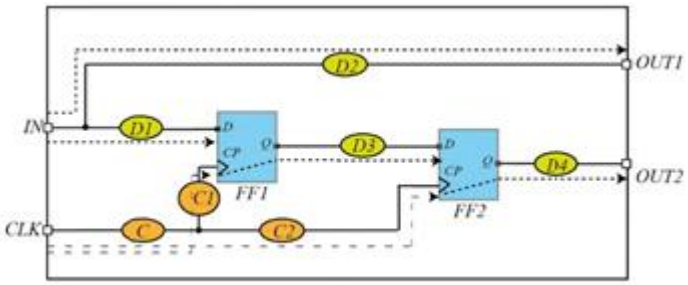
So the circuit has 2 flip flops this and this and then we have an input port in and 2 output ports out 1 and out 2. There is a clock port which is also an input port. Now in this circuit see the combinational logic cloud or a group of combinational logic circuit elements that are shown as D1, D2, D3, C1, C2 and C. These are combinational circuit elements only a group of combinational circuit elements. Now in this circuit let us first identify what are the timing start points.

So timing start points are the input ports and we are not considering clocks as the timing start points because that is a special clock, a special input port which we treat as a clock port. And then there are the other timing start points and those are the clock pins, clock pins, this one and this one. So these are 3 timing start points and what are the timing end points. So the timing endpoints are the D pins of the flip flops. So these are the timing end points and the output ports.

So these are the timing end points. Now in this circuit how many data paths can we identify. So of course the first path is from In1 through this D1 and ends on the D pin. This is one of the data paths. Then another data path we can identify from In1 going through D2 and finally ending on the output. So this is another data path and then we can identify a third data path starting from the clock pin of FF1 going through the Q pin of FF1 and then D3 is a combinational logic cloud and finally ending at the D pin of FF2.

So this is the third one and the fourth one is from the clock pin of FF2 and going through D4 and finally ending on output. So we can identify 4 types of data paths in this circuit. Now the other type of path in an STA or from the perspective of STA is the clock path. Now what is a clock path? The clock path starts at the clock source and typically the clock source is identified by the constraints that we specify. So remember from our earlier discussions that we specify constraints in an SDC file SDC file which is in the synopsys design constraints file and in that constraints file we define the clocks whose pins or ports in our design refer to the start points of the clock.

So let us assume that the start point in this example is this clock and that is why we did not consider clock as the start point of the data path. So the clock is supposed to be a constraint here, say create a clock or we will go into details of constraints in the next lecture. So let us so a clock path starts from where the clock is specified in the SDC file the start point of that. So in this case the clock port is a start point and then it can go through combinational logic elements typically buffers and inverters are only used in the in the in the in the clock path there can be some special elements like clock gaiters also. So the clock path will pass through those circuit elements and then it ends at the and then it ends at the clock pin of the flip-flop.



So it ends at the clock pin of the flip-flop. So the start point of the start point of the clock path is the clock source as specified in the SDC file. This point is the start point and the end points are the clock pins of the flip-flops. So this is the end point and this is the end point. So there are two clock paths that we can identify in this circuit. First is the clock going to C1 and then ending at FF1 CP and the other path is starting from clock going to C and C2 and then ending at the CP pin of FF2. So there are two clock paths in our circuit.

So these are very important definitions from the perspective of ST. What are timing end points, what are timing start points and what are clock start points and what are clock end points. Now let us look into the mechanism of static timing analysis. Now static timing analysis starts by building a timing graph of a given circuit and what is a timing graph?

G = (V, E) , Where V is the Vertex set and E is the edge set

Timing graph like any other graph is a graph right and it consists of a set of vertices and a set of edges set off so we can say that the timing graph G is equal to V, E where V is the set of vertices and E is the set of edges. Now these vertices and edges are defined in a special way for a timing graph. So the vertices in a timing graph correspond to pins or ports in the given circuit.

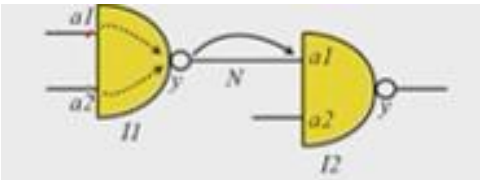
v ∈ V , The vertex corresponding to a pin,

e ∈ E , represents a timing arc in the circuit

$$e(i,j) = (Vi, Vj)$$

So the pin and the port that are in our circuit consist of basically the vertices of the timing graph and what are the edges of the timing graph? An edge E represents a timing arc in the circuit. So what is an edge in a timing graph? It refers to the timing arcs in our circuit. So an edge Ei, j exists between two vertices Vi and Vj if and only if there is a timing arc between the corresponding pins or ports in the circuit. What this means is that suppose there is a vertex Vi and there is another vertex Vj now there will be an edge existing between them with the directions same as the timing arc. So if there is a for the if for the given vertices Vi and Vj there exists a timing arc in the circuit.

So there are two types of edges in a timing graph. The first one is the cell arc. So a cell arc is a timing arc between two pins of the same cell. And there is another type of edge in a timing in a timing graph and that is known as net arc. So what is a net arc? Net arc is a timing arc between two pins of different cells that are connected by a net directly.

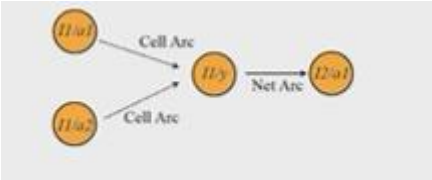


So let us understand these definitions by taking an example. Suppose there is a portion of circuit that is given to us. Now we can identify that there are three pins I1, A1 rite I1, A2, I1, Y. These are three pins of the instances I1 right. Similarly I2 will have the I2 has pins A1, A2 and Y right.

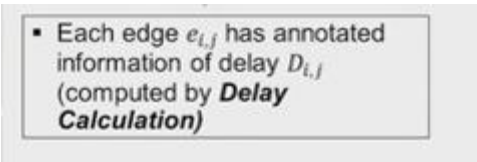
Now these pins will form the vertices of the timing graph right. So in this figure I am showing just these three sorry, these four vertices: first vertex, second vertex, third vertex and the fourth vertex right. So I1, A1 corresponds to this one, I1, A2 corresponds to this first one, I1, Y corresponds to this and I2, A1 corresponds to this. So we are only considering four vertices in this example just for simplification. There will be other vertices corresponding to I2, A2 and I2, Y also which we have not shown.

Now if we consider the instance I1 then there is a timing arc between A1 and Y. Similarly there is a timing arc between A2 and Y. So these timing arcs will be defined in our library dot lib file that we have discussed in the earlier lectures. So there will be a cell arc there will be two cell arcs for these arcs which exist within a cell right. So the cell arc is a timing arc between two pins of the same cell.

So these two arcs are cell arcs right. Now the signal that is generated at the point at the pin I1, Y that will propagate through the net n and it will reach the pin I2, A1 right. So it means that there is a timing arc also existing between I1, Y and I2, A1 right. So this arc is known as net arc because this is external to the cell. It is not within the cell but outside the cell.



So the net arc is a timing arc between two pins of different cells. So this is a different cell that is another cell that is connected directly by a net. So in this case they are directly connected by the net n right. So this is how a timing graph will be made for this small portion of our circuit. Now each edge Eij has annotated information of delay Dij.

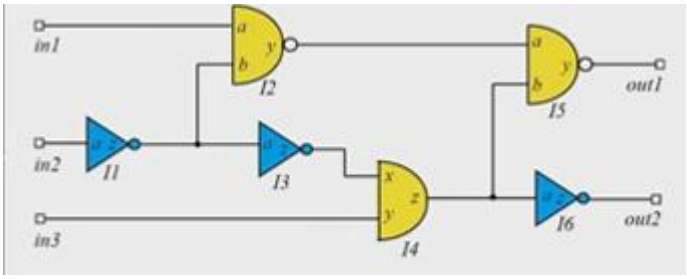


So when the timing analysis is done or when a STA is performed then on each edge the tool will internally annotate the delay corresponding to that arc. For example here there will be a delay of say 10 picosecond here it can be 12 picosecond at this edge there can be a delay of say 5 picosecond or so on the right. So at each edge in our in the timing graph the tool will annotate its D right and on the vertices the tool will annotate various other timing information such as arrival time, required time, slack and other information. Now in the timing graph there are some special vertices for example there are some vertices that are identified as source vertices. Now what are source vertices? Vertices with no incoming edge suppose there is a vertex and from this vertex we have only outgoing edge may and there is no incoming edge into this vertex then we say that it is the source of our timing graph.

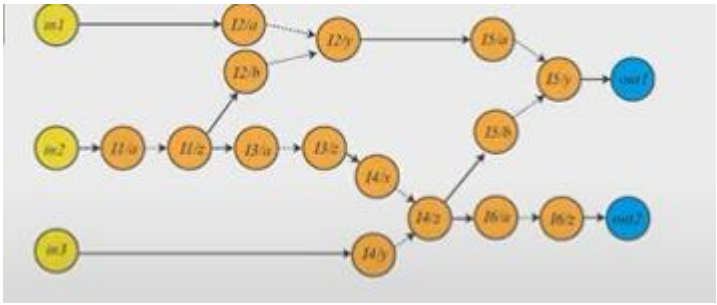
So timing start points such as the input ports, the clock pins or flip flops are and the clock start points are basically the source vertices in the timing graph right because no edge comes to them, only edges leave those vertices. Similarly there are some vertices in our timing graph which act only as sink meaning that those vertices will not have any outgoing edge but it may have multiple incoming edges. So these will happen typically for the timing end points. So at the timing end points such as say output ports or the D pins of the flip flops there the edges will only come, it will not go out from them and therefore those are the sink vertices in our timing graph. Now let us take an example of a circuit and understand how the timing graph will be built for this right.

So we have just for the case of simplicity we have just taken a combination circuit right. So in

this circuit there are in the input ports in 1, in 2 and in 3. So this will be timing start points for the data pass and therefore will be considered that these will form the source vertices. So the edges will leave those vertices well no edge will come to these vertices and out 1 and out 2 will be the sink vertices for the timing graph right. So let us draw the timing graph for this case.



So in this case the timing graph will look something like this. Now let me explain how we get this timing graph.



So for the source vertices we have only the input ports or the timing start points we have only the source vertices in 1, in 2, in 3 right. And for the output ports we have only the sink vertices out 1 and out. Now from in 1 there is a net arc right from in 1 to the pin A of I 2.

So we have a net arc right and corresponding to all the pins in our circuit so I 2 A, I 2 B, I 2 Y there are vertices right. Similarly for this inverter we have I 1 A, I 1 Z so we have 2 vertices. Similarly for I 3 we have I 3 A, I 3 Z for I 4 we have I 4 X, I 4 Y, I 4 Z. Similarly for I 5 and I 6 so we have various vertices for all the pins and ports in our design. And then we have edges corresponding to the net arc and the cell arcs.

So inside this NAND gate for I 2 we will have an edge between A and Y and B and Y. So these are cell arcs. So we have cell arcs. So cell arcs are shown by this dotted line. So we have these cell arcs for the inverter. For this inverter we again have cell arcs between I 3 A and I 3 Z and then we have cell arcs between X and Z, Y and Z right.

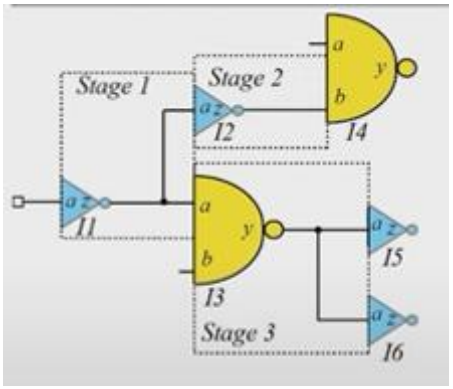
And similarly we have A and Y cell arcs for between A and Y and B and Y and we have the cell arc between I 6 A and I 6 Z. And for each of the nets we have the net arcs right. So we have a net arc for this net, this net and then from I 1 Z to I 2 B we have a net arc right. This corresponds to this and then there is another net arc between I 1 Z to I 3 A so this is the net arc and so on. So this is how we build a timing graph for our given circuit.

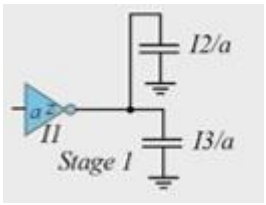
So now once the timing arc sorry timing graph has been built for our circuit the next stage comes that we need to compute the delay also at each edge in the timing graph. Now how does the tool compute the delay for the timing for the timing arcs in a timing graph. So the STA tool needs to know the delay for the timing arcs existing in the timing graph because once the delay is computed then only the other attributes of timing such as arrival time and required time and those things can be computed. So typically what happens is that the STA tool is coupled closely with something known as a delay calculator. And so the STA tool will retrieve the information of the delay of each timing arc using a delay calculator.

So either this delay calculator will be built inside the STA tool or it can be coupled loosely and the STA tool will retrieve the information from the delay calculator as and when required by the tool. Now let us understand a little more deeper into how delay calculation for a circuit or timing graph is done. So given a complicated circuit the tool what internally does is that it computes the delay in small small steps. It decomposes a large circuit into smaller circuits and then computes or invokes a delay calculator on each small circuit in sequence. So how does a tool decompose a big circuit? So it creates what is known as a stage for a small sub circuit of our design.

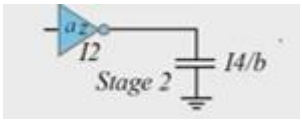
So it decomposes a large circuit into lots of small small stages and then delay calculation is done for each of the stages individually. Now what does a stage consist of? A stage is composed of a driving cell and its driven pins connected through wires. So let us understand this with an example. Suppose this is the portion of the circuit which is for which we need to do delay calculation.

Then it will decompose it into various stages. So for example the tool can consider this as a stage. Now what does this stage compose of? So this stage consists of the driving cell, a driving cell I1. So the I1 is the driving cell and what it is driving? It is driving this net and the pins which are driven by this net are I2a and I3a. So these are the pins being driven. So a stage will be created in which we have a driving cell that is I1 and the interconnect that is this net and the pins that are driven by this interconnect that is I2a.

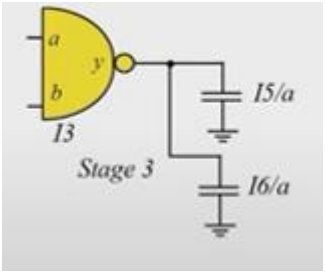




So we can model this as a capacitor because we are considering a CMOS circuit and input to a CMOS logic gate is nothing but the gate terminal of the MOSFET and which we can model as a capacitor. Similarly we can have the terminal I3a, I3a is modeled as a capacitor. So this is the first stage. The stage will consist of I1 and the net and the pins being driven by this stage.

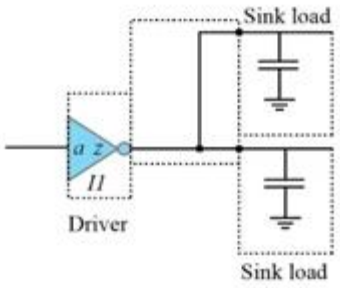


Then there will be another stage corresponding to this. So stage 2 will consist of the driving cell I2, the net and the pin I4b. Similarly we will have another driving stage corresponding to I3. This is the driving cell, this is the interconnect and then we have two driven pins that are I5a and I6a. So similarly there will be another stage for I4 and so on which we have not. So this is how the stage will be formed or multiple stages will be formed from a large circuit, a small small circuit.



And then delay calculation will be invoked for each of the stages. For example stage 1 will be first computed, then stage 2 will be computed and then stage 3 or delay calculation for stage 3 will be done. So now how do the delay calculator compute the delay for a given stage? So to compute the delay for a given stage, the tool will need various kinds of information. For example, what are the characteristics of this driver? Now this driver is driving the interconnect and the pin loads. Now if this driver was very strong then it would charge very faster and the delay will be much smaller.

So the information about what is the characteristics of the driver I1 that is contained in the .lib file that we had discussed earlier. So the tool will look into the.lib file.



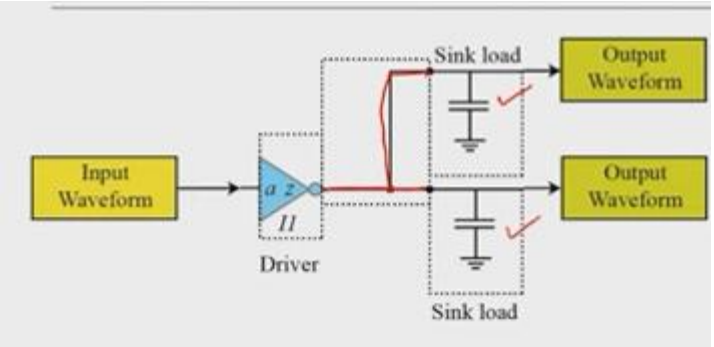
lib file which is typically in the liberty format from the.lib file the tool will get the driver model for this driver I1 which may have delay model as either as NLDM that is non-linear delay model which we had discussed in the earlier lectures or it can be advanced delay model like such as CCS or ECS. So this is the first part. So the driver model of the cells will be or the driving cells will be necessary for computing the delay of a given stage. Then comes the attributes of the interconnect.

So there is an interconnect. Now depending on the characteristics whether how much capacitance it has, how much resistance it has and so on the delay from the driver to the sinks can vary and therefore the interconnect model is also important. So during logic synthesis or in the earlier stages of VLSI design flow we can assume that the capacitance of the interconnect is 0. So that is a very crude approximation to or we are neglecting the effects of the capacitance. But as the design flow progresses and we have more or the design contains more information about the interconnects, how they are laid out and so on. Then more accurate models for the interconnect such as lumped capacitance model or ELMO delay model or a delay model which is based on what is known as asymptotic waveform evaluation model.

Those can be used for modeling the interconnects. So the second important information that a delay calculator needs is the characteristics of the interconnects. Characteristics of the interconnects in terms of the capacitance, resistance and so on. And then an extraction level shows how that information needs to be used in the delay calculation or what should be the delay model for the interconnects in our circuit. Now the capacitance resistance of the interconnects are typically extracted if we have a layout of your layout and in which the interconnects are laid out. Then from the from the layout the tool which is known as parasitic extraction tool that can extract the capacitance out of out of the layout and then represent that in some RC model or resistance capacitance model and put it in a file which is known as SPEF file or SPEF file which is standard parasitic exchange for the interconnects.

Format so this contains the information of the interconnects of the interconnects or the

resistance and capacitances of the interconnects and using this as a SPEF file the tool can use various interconnect delay models such as LUM model, ELMo model and so on. And then it can compute the delay associated with the interconnect. And the third important thing that a delay calculator needs for doing delay calculation for a stage is the receiver model meaning that how do we model the sinks, sink pins in this case suppose there are one sink pin another sink pin then either it can consider it as a as a simple lumped capacitor or it can use some more complicated receiver model or more advanced receiver model for our for the receiver. Now using this information the tool can compute the delay calculator and can compute the delay of the given stage. However, to compute the delay the tool will also need one information that is what is the input waveform right meaning that if the SNU is very very very steep right meaning that the the input waveform is rising very fast then the delay will be smaller and so on.



So, what is the characteristics of the signal which is going to this stage that is also important. Now given a stage and the input waveform and all the information that I have mentioned, the tool can compute the output waveform at each of the sink pins right. And using this output information then the tool can basically compute various attributes of the timing arc such as delay the slew and other things. So, if we need an input waveform at a for a stage then we cannot arbitrarily do the delay calculation; it has to be done in some topological order. So, as the transition at the input ports we typically assume something or we apply as a designer we can set it using some constraints that we will discuss later.

So, the transition or the or the shape of the waveform at the input ports are known to the delay calculated calculator tool right. However, and using that information of the input waveform at the stage directly connected to the input port the delay sorry the output waveform of the subsequent stages can be computed in a topological order right. So, that is how the delay calculation is done from the starting from the input port it moves forward in the fan out of the circuit and goes on performing delay calculation stage by stage right. So, this is how the delay

calculation for a stage for a timing R is done and that delay is annotated on the timing graph right. So that it can be easily used by subsequent stages of the timing analysis.

So, each edge E i j has annotated information of the delay and also of the slew because once we know the input slew then we can compute the delay of the next stage and so on. So, whatever the output waveform will be here it will be useful in the delay calculation of the next stage and so on. So, the tool will keep a track of this input waveform output waveform and the delay for the entire circuit. Now, let us understand how the arrival time of a given vertex in a timing graph is computed. So, you may remember from the last lecture that we model the timing constraints in terms of arrival time and the required time.

We say that for setup analysis the arrival time should always be less than the required time right. So, this constraint we had derived in the last lecture was right. So, if we want to derive a constraint or for the timing analysis we first need to compute this arrival time and then we also need to compute the required time and then see whether this inequality holds or not. If it does not hold then we say that there is a timing violation in our circuit and if this inequality holds then we say that the time the circuit is timing. So, basically we need to first understand the tool needed to compute the arrival time at each vertex in our design.

So, first let us understand what is an arrival time and how it can be computed. So, arrival time is the time at which a signal settles at a given vertex. So, there is a vertex and the signals will come from every from various paths and the signal can the the value at a given vertex can change right. Now, when what is the what is the time at which the signal settles at a given vertex that is known as the arrival time. So, when there is only one incoming edge, suppose there was a vertex v j and there was a vertex v i and there was a timing arc between.

$$A(j) = A(i) + D(ij)$$

So, and if we say that the delay between these two arcs is d i j then we can say that if the arrival time at this v i was a i then arrival time at this v j will be a j where a j is equal to a i plus d i right. So, this is very simple if there is only one incoming edge at a vertex then we can simply add it and we are done right. But if there are multiple incoming edges if there are multiple incoming edges to a given vertex then what will happen. So, now whenever there are multiple edges to a multiple incoming edges to a vertex then the signal can toggle multiple times before settling right. So, for example, let us take an example: let us consider a vertex vj right and suppose it has got three incoming vertices from three sorry incoming edges.

From three vertices right let us say that the arrival time at this was 100 arrival time at this vertex was 80 time units some arbitrary time units we can take and at the third vertex let us assume that the arrival time was say 150. Now let us assume that the delay of the edges of all these incoming edges is say 10, 20 and 50. So, at what time the signal will arrive from each of

the edges. So, for the first vertex from this vertex the signal will arrive with at v j at 100 plus 10 that is 110 time units right. Now for the other vertex the signal will arrive at 80 plus 20 that is 100 and for the third the signal will arrive at 150 plus 15 that is 165.

So, there will be multiple values at a vertex that will change right. So, whenever the signal comes at 100 picoseconds. So, at 100 picoseconds 100 time units the value at this v j will change when the first first when the the when the effect of the first path comes right that is the first part is this because it has got the minimum arrival time that is 100. And then the second it will change after say 110 picoseconds or time 110 time units and then from the third path that is whenever the signal comes then it will settle the the value will change at 165 time units and then it will settle right. So, as the signal arrives from multiple paths the value at a given vertex can change or toggle multiple times before settling. So, what it means is that we can associate multiple arrival times at a given vertex depending on how many edges are coming right.

Now if we try to keep track of all the arrival times from all the paths then it will lead to a lot of memory requirements to keep at a given vertex. So, rather than keeping all the track of all the arrival times what we do is that we we keep track of only the minimum arrival time and the maximum arrival time or a kind of bound on the arrival time at a given vertex v j can be computed if the arrival time at all its input vertices are known right. So, in if so let us what we do is that we at at each vertex we associate a a a bound on the arrival time the minimum arrival time and the maximum arrival time and using those and using these expressions a j min is equal to min of a i min plus d i j and a j max is equal to max a i max plus d i j we can a we will be able to compute the bound at any other vertex v j. Let us understand this equation and what this formula means. Suppose there is a vertex v j a vertex is v j right and there are say three vertices three vertices we call them as v i's right.

$$A(j,min) = Min(A(i,min) + D(ij)$$

$$A(j,max) = Max(A(i,max) + D(ij)$$

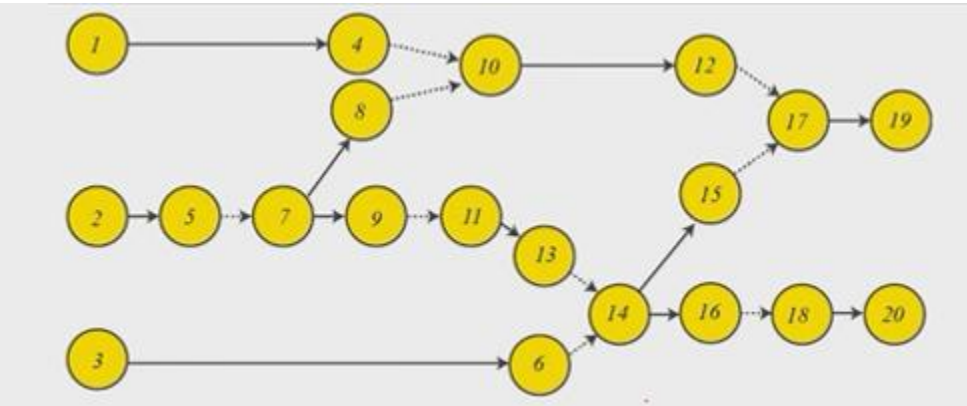
So, these are v i's. So, there are three v i's in this case and there are edges between right. So, at each vertex we keep a bound on the arrival time. Let us call this vertex as vertex one this as two and vertex three. So, on vertex one let us let the bound be say the minimum arrival time is say 100 and the maximum is 150. For the second vertex let us assume that the minimum arrival

time is 80 and maximum arrival time is 120. For the third vertex let us assume that the minimum arrival time is 150 and the maximum is 200.

Now for simplicity let us assume that the delay of each of the edges is say 10 time units right. Now given this we can compute the bound on the arrival time at v j and what we can do with the help of these two equations. Now what is the minimum arrival time at this vertex v j we take the minimum arrival time of each vertex and add with the delay and then out of them which one gives the minimum we take. For example, if we take 100 plus 10 right so that will be the arrival time from 1 to v j right. So, 110 will be the number which if it comes from this if it comes from this then 80 plus 10 is 90 and if it comes from the third vertex then it is 150 plus 10 that is 160.

Now out of this the minimum is 90 that is coming from 2 to v j right. So, the minimum arrival time we can get from this formula right minimum of the so a i's in this case are corresponding to a i mean corresponds to 100, 80 and 150 in this example right. So, the minimum arrival time at v j will be 90 and what will be the maximum arrival time we take the maximum for each vertex and then add with the delay and see which one is giving us the maximum. So, for the first path from 1 to v j 150 plus 10 is 160, 120 plus 10 is 130. So, 120 plus 10 is 130 and 200 plus 10 is 210.

So, out of them the maximum is 210 which will be taken right. So, this is how we can compute the bound on the arrival time given that the bound on the incoming vertex is already a vertex that is already known. Now, how do we do it in a timing graph? So, the arrival time is computed and stored at each vertex in a in a timing graph and arrival time at the input ports is specified by the constraints that it is s d c file or it is assumed to be 0 ok by the 2 and then arrival time is done by the forward traversal of the timing graph. So, we start from the input ports or start the computation from the input port and proceed in the fan out of the timing graph and go on computing the arrival time in the timing graph. So, a vertex is chosen for computing an arrival time such that all the input all the arrival times or the arrival time at all the input vertices are already known right. So, now, to compute the bound on the arrival time we need to know the arrival time of the signal at all the input vertices or then only we can compute the right.



So, all the input vertices of the given vertex have their arrival time already computed, then we progress with or proceed with computing the arrival time for a given vertex. Now, an arrival time can be computed in one traversal of the vertices and edges of the timing graph. So, this is an important thing and this is as we will see that to compute the arrival time we need to visit the vertex only once in the timing graph and this is why that static timing analysis is very efficient to compute the on the compute the bound on the arrival time we just need to visit a given vertex only once in the in the in the timing graph and it makes that computation of arrival time very efficient. So, let us take an example. Let us take an example and understand how maximum arrival time for all the vertices in a timing graph can be computed. So, similar things can be done for the minimum arrival time also for illustration purposes let us look only at the maximum arrival time for now right.

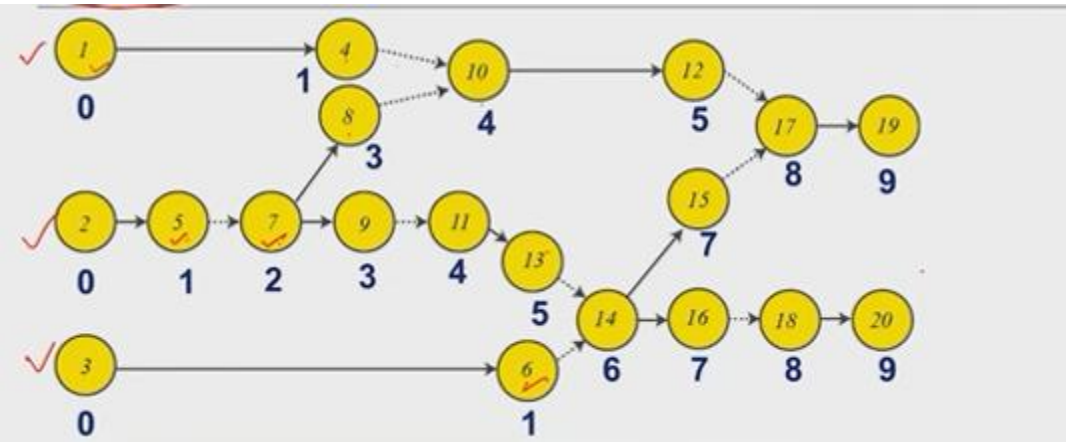
Suppose this is the timing graph that is given to us right and assume that the delay of all the edges is one time unit just for the sake of simplicity we assume that all edges in our graph has a delay of one time unit and then we say that the arrival time at the at the source vertices these are the source vertices are given as 0 right. So, now, with this information how do we proceed computing the arrival time for all other vertices in our graph. So, we are looking only at the maximum arrival time which would be useful in the setup analysis right. So, if we know the arrival time at the vertex 1 right. So, we can easily compute the arrival time at vertex 4 vertex 5 and vertex 6 because the arrival time at the incoming or input vertices 1, 2 and 3 are already known and we just add the delay and we get the numbers right.

So, we can compute the arrival time at 4 as 0 plus 1 that is 1 and from at 5 0 plus 1 is 1 and for 6 0 plus 1 is 1. Now, let us look at the vertices which are in close proximity to these vertices. Now, for this vertex 10 can we compute the arrival time now we cannot do it because we do not know the arrival time at the vertex 8 right. So, we have to wait now let us consider this vertex 7 can we compute the arrival time for vertex 7 yes we can because only one edge is in the is come is there incoming edge is there and we know the arrival time at the vertex input vertex 5 and we simply add 1 to it we get the arrival time at 7. So, we get the arrival time at 7 s 2. Now do we know if we can compute the arrival time of 14? We cannot because we do not know the arrival time at 30 right which is in the input vertex.

So, we have to wait now given that we know the arrival time at 7. We can compute the arrival time at 8 and 9 right because those have values 7 as input vertices right. So, we add 2 plus 1 that is 3 we get the arrival time maximum arrival time at 8 and 9 right. Now we can know the arrival time for the vertex 10. We know the arrival time for both the incoming edges 4 and 8 and therefore, we can compute it right similarly for 11 right for the vertex 11 we can compute simply 3 plus 1 is 4 which we can compute. Now at the vertex 10 there are 2 possibilities of the arrival time: if we add 1 plus 1 from the path from 4 to 10 then we get an arrival time of 1 and if

we look into the path from 8 to 10 then 3 plus 1 is again 4 is 4.

So, for the maximum arrival time we have to take the maximum of these 2 right and the maximum is 4. So, the arrival time at 10 will be 4 and the arrival time at 11 will also be 4 right. Now similarly we can compute the arrival time at 13 and 12 . Those will be 5 and then we cannot compute for 17 because 15 we do not know now for 14 we can compute the arrival time as 5 plus 1 is 6 and 1 plus 1 is 2. So, out of this the maximum is maximum of 2 and 6 is 6 so the arrival time at 14 will be 6 right and the arrival time at 15 will be 7 and arrival time at 16 will also be 7. Then we can now compute the arrival time at 17 that is 7 plus 1 is 8 the other path is not maximum right and similarly for 18 the maximum arrival time is 8 and finally at the sink vertices 19 and 20 we get the arrival time as 9.



So, this is how we can compute the arrival time throughout our graph. So, now as we have done for maximum arrival time we can similarly do for the minimum arrival time computation also in that we need to consider the minimum arrival time of all the input vertices and then we have to use the min operation right. That is only the difference. Now in this computation of arrival times there are some complications. Now what are those complications? The first is that the rise delay and fall delay may be different right in a general case the rise delay and fall delay time can be different and therefore, we need to compute separately the arrival time for the maximum case sorry for the rise case and also for the fall case. So, given a vertex now we have to compute 4 arrival times. One arrival time maximum for rise then arrival time maximum for fall and the second and then arrival time minimum for rise and arrival time minimum for fall.

So, at a given vertex will have 4 arrival times right. So, this is this is this is a added complication that we need to consider when computing the arrival times and another important thing is that as we saw that when we do delay calculation we need to know the delay of the incoming waveform sorry the form of the incoming waveform or the slew of the incoming

waveform right. So, if that means that we have to also store the slew at each of the points in our vertices in our in our in our timing graph then only we can compute the delay of the subsequent stages in the fan out right. So, the dependence of delay on the input slew that is the rise fall transition time because of this dependence the slew also needs to be propagated and stored at each vertex in addition to delay right. So, what it means is that we will need to also compute and store the slew at each of the vertices in our timing graph right.

So, that is added complications of computing arrival time in a timing graph. Now, we know how to compute the arrival time and other important things that we need to consider when deciding whether the timing constraints or timing requirements are met is the required time right. So, we need to also compute the required time in our circuit. So, what is the required time? So, required time is the time constraint for a given vertex to avoid the setup and hold violations. So, the required time will be different for the setup analysis and for the hold analysis.

So, how these times are sorry and how that required time differs in different analysis times let us look into that. So, in the setup analysis or late analysis the required time represents the maximum time by which the signal should arrive to avoid setup violation right. So, that is giving us the latest arrival by that time the signal should arrive it should not be more late than that right. So, it is giving a constraint on how late a signal can be and beyond that if the signal becomes late then we get a violation right. So, for the setup analysis the required time gives the maximum time by which a signal should arrive at a given vertex to avoid the setup violation. And for the hold analysis the minimum for the required time represents the minimum time after which a signal should arrive to avoid a hold violation, meaning that if it gives us a bound on the earliest time that a signal should arrive.

If it arrives before the earliest time determined by the required time then there will be a hold violation in our sign. So, that the signal can come later than what is there for the specified by the required time in early mode it can be later, but it cannot come earlier than what is there represented by the required time right. So, now given given a vertex and and and outgoing edges we can compute the arrival sorry required time by looking in the Feynman code right and and based on it the tool will find two types of bounds on the on the arrival on the required time and one is related to the hold and the other is related to the setup. So, if the required time for a vertex vi can be computed so suppose there is a vertex vi then the required time at the vertex vi can be computed if the required time at all its output vertices suppose there are three output vertices all of them denoted as vj.

$$R(i,hold) = Max(R(i,hold) - Dij)$$

$$R(i, setup) = Min(R(j,setup) - D(ij))$$

So, there are three vj's in this case. So, if the required time for the setup and hold are known at the output vertices then we can compute the required time at the vertex vi right. So, let us assume that let us call this as vertex 1 this as vertex 2 and vertex 3. Let us assume that the required time for the hold case for the vertex 1 was 10 and for the setup it was say 150 for the for the second vertex let us say that the required time for hold was 15 and for the setup it was say 120 and for the third vertex let us say that the required time was 20 for the hold case and for the setup case let us say that the required time was 180. Now, for the sake of simplicity let us say that the delay of each of the edges is 10 time units.

Now, using these two formulas let us compute the required time at vertex vi a how we can compute it. Now, for the hold analysis what we need to do is that we take the hold value for all the incoming vertices or outgoing vertices or output vertices 1 2 3. So, that the hold requirements or hold time required hold required time is 10 15 and 20 we subtract the delay that is 10 in this case and then take the maximum out of it right. So, if we take 10 10 minus 10 is 0 15 minus 10 is 5 and 20 minus 3 is sorry 20 minus 10 is 10 for the third vertex right. So, out of this we have to take the maximum and the maximum is 10 in this case.

So, the hold required time for this vertex will be 10 and what will be the setup required time. So, we have to again take the setup required time subtract 10 and out of them take the minimum value right. So, 150 minus 10 is 140 120 minus 10 is 110 180 minus 10 is 170 right. Now, out of this which one is the minimum the minimum is 110 right. So, we have to get the setup required time right. So, the definition of this required time for hold and setup has been chosen such that we always get the pessimistic value right.

Now, if the required time for the edges are say for different edges sorry from different paths are 140 110 and 170 for the setup analysis. If we take the minimum of it then what it means is that if the required requirement for the minimum of them for the circuit sorry minimum required time is satisfied meaning that the arrival time is less than 110 the other constraints of 140 and 170 will automatically be satisfied right. And that is why these definitions of required time have been created right. So that the most pessimistic of the required times or most pessimistic of the requirements to avoid the setup and hold violations at a given vertex those are considered and stored in the on the vertices. Now, how does for a given timing graph how does a required time computed? So, at each vertex in the timing graph required time is computed and stored similar to the arrival time.

Now, the arrival required time at the output ports or the timing end point is specified or inferred from constraints right. So, the constraint can be the at the output port we specified in SDC file or this confer constraint can be inferred using the clock constraint or the clock period at the at the at the clock spins of the of at the clock pins of the flip flops and based on that the required time

can be computed at the D pin of the of the flip flop and then that can be propagated in the fanning cone of the of the of the circuit or the timing graph. So, required time computation is done by backward traversal of the timing graph right. So, if we have a node to traverse in the fanning cone right, that is what we mean by backward traversal of the timing graph. So, required time computation starts from the vertices corresponding to the output ports or the or the end points of the timing graph.

So, the required time is computed from the end points and into the fanning of the circuit. So, a vertex is chosen for computing required time such that all the output vertices of a given vertex have their required time already computed. Then only those max and min operations that we described in the previous slide can only be computed if we know the max and min time for the output vertices right. So, therefore, to compute the required time for a vertex we should know the required times at the output vertices. So, required time can be computed in one traversal of the vertices and edges of the timing graph and that is why the required time computation is also very efficient for static timing analysis and this makes static timing analysis a very efficient way of ensuring the safety or timing safety of our circuit.

So, typically the required time is computed after the arrival time. So, once we have computed the arrival time during that computation we already annotated the delay at each of the edges right then only we computed the arrival time. So, the required time computation will reuse the delay calculation result that was done in the arrival time computation and therefore, save the run time. So, the required time constraints are primarily determined by the clock period meaning that if the clock frequency is very high then the required time may be much smaller. So, it becomes stricter in that sense and therefore, the required time that will be valid for the output ports or the timing endpoints will be derived using the clock constraints that we specify in the SDC file right.

So, now let us look into an example like this graph that is the same graph that is a timing graph that we saw in the earlier slides. Now let us assume that the delay of all edges are one time units that are similar to what we had assumed earlier and assume that the required time at the sink vertices is 19 and 20 these vertices are given to us right. So, 12 or 11. So, these can come from other constraints SDC files or this can be derived from the clock constraints or clock frequency at the timing endpoints. Now, assume that these numbers are there. So, at vertex 19 the required time is 12 and at vertex 20 the required time is 11 and let us for the case of simplicity let us consider the setup analysis case right similar to what we had done for the arrival time.

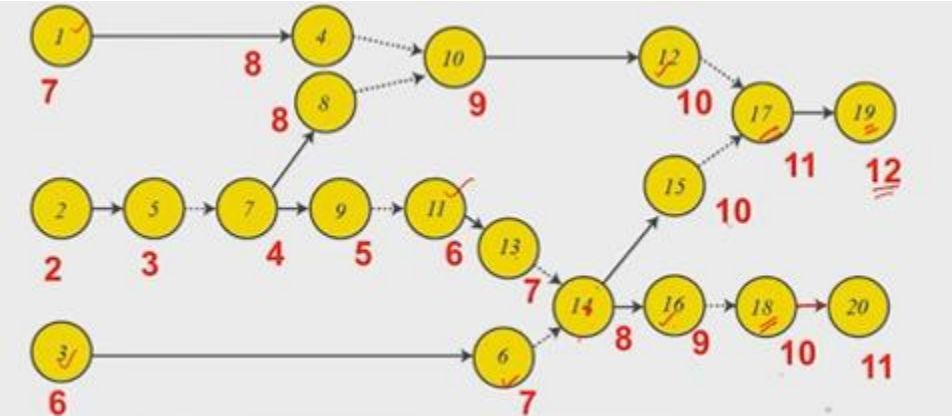
We are now assuming that we are looking at computing the required time for the setup case not for the hold case right. Now let us for this timing graph let us see how required time can be computed right. So now if the required time at 19 is 12 then 12 minus 1 is the required time at 70 at this point. Similarly the required time at this 8 this vertex 18 is 11 minus 10 sorry 11 minus

1 that is the delay of this edge we have the required time as 10 right.

Now next we can compute the required time at 12 as 11 minus 1 is 10 and 11 minus 1 is 10 for the vertices 12 and 15. Similarly for vertex 16 we can compute the required time as 9 10 minus 1 is 9. Now for the vertex 14 there are 2 paths, one coming from 15 to 14 and 16 to 14. Now if we take this path right the required time will be 10 minus 10 10 minus 1 that is 9 and if we look into the path 9 sorry path from 16 to 14 then 9 minus 1 that is 8 right. So there are 2 required times at this point.

Now for the set up analysis we have to take the minimum right that is the most pessimistic bound right. So we will take 8 in this case minimum of 8 and 9 is 8 right. So similarly for the vertex 10 there was no ambiguity 10 minus 1 is 9 right. Similarly we can compute for vertex 6 right 8 minus 1 is 7 8 minus 1 is 7 right. Similarly for 13 and similarly for 4 and 8 we can compute the required time right.

And go on progress doing vertex 11 and then vertex 1 and vertex 3 right. At vertex 9 we have the required time as 5 now at vertex 7 we have to apply the min operation. So 8 minus 8 minus 1 is 7 right and 5 minus 1 is 4 out of this the minimum is 4. So at vertex 7 the required time will be 4 right. Now for vertex 5 the required time will be 3 and for vertex 2 it will be 2 right.



So we got the required time at each vertex in the timing graph. Now this we did for the set up analysis similarly we can do for hold analysis also we have to use the other equation the max and the max operation. Now once we have done the set up, sorry the arrival time computation and the required time computation then we can compute that whether it is timing that is our circuit is timing safe or not right meaning that what is the slack in our circuit right. So now that slack will be dependent on whether we are looking into the set up analysis or we are looking into the hold analysis. Now for the set up or late analysis what is the requirement? requirement is that the required time should be greater than the arrival time right. So we define slack as RT minus AT and then the slack the requirement will be that the slack should be positive if this

constraint RT is greater than AT is satisfied then the slack should be positive right.

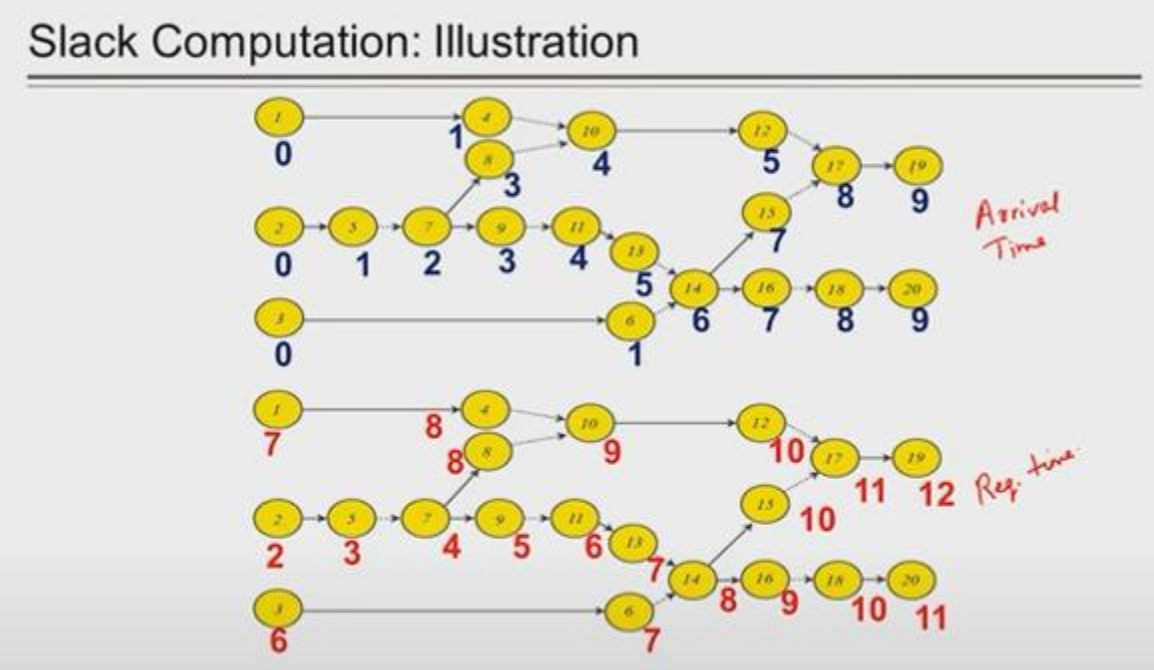
Slack = RT - AT

So what does slack represent? The slack is the time by which arrival time at a vertex can be increased without causing setup violation rights. So setting the slack is basically showing us a margin that we have in increase in the arrival time at a given vertex for the set up analysis right. So if we say that at a given vertex the slack was say 20 picosecond it means that if the arrival time is increased up to 20 picosecond then there will not be any timing violation in our circuit. But if we increase our arrival time by say more than 20 picosecond for example make it say 25 picosecond increase the arrival time by 25 picosecond then we will have a timing violation in our circuit.

So the arrival time can be increased till slack becomes 0. Now for the hold analysis case the required as we saw in our last lecture is that the arrival time should be greater than the required time then the hold or early early constraint is satisfied otherwise it is violated. And therefore we define slack as arrival time minus required time if the slack is positive then we say that the hold requirement is met otherwise there is a hold violation in our circuit. So slack is the time by which the arrival time at a vertex can be decreased without causing hold violation. So arrival time can be decreased till slack becomes 0 for the hold time requirements right. So a negative slack implies that there is a timing violation if there is a negative slack for setup case then there is a setup violation if there is a negative negative slack for hold case there is a hold violation.

And if that happens then we need to probably fix the problem in our circuit for the circuit to behave properly in terms of time right. Now let us look into the example that we just saw. So we already computed the timing graph for the arrival time. So this is how we computed the arrival time and we also computed the required time right. Now using these two values for each of the vertices we can compute the slack at or the setup slack at each of the vertices.

For example the setup slack at the vertex 19 is 12 minus 9 is equal to 3 right. Similarly for the vertex 20 the setup slack is 11 minus 9 is equal to 2 right. So out of these two timing end points the setup slack is lower for the timing endpoint 20 right. However the slack is positive for both the cases and therefore there is no timing violation in this circuit. If you want to know more about what we have discussed in this lecture you may refer to these books and to summarize what we have discussed in today's lecture.



So we looked into a few important definitions related to STA: what are timing start points, what are endpoints, what are clock sources and so on. And then we looked into what is a timing graph and then how a delay calculation is done for a given stage in a timing graph and then how the delays are used to compute the arrival time, required time and slack in our design. And in the next lecture we will be looking at some more advanced concepts related to static time analysis. So thank you very much.