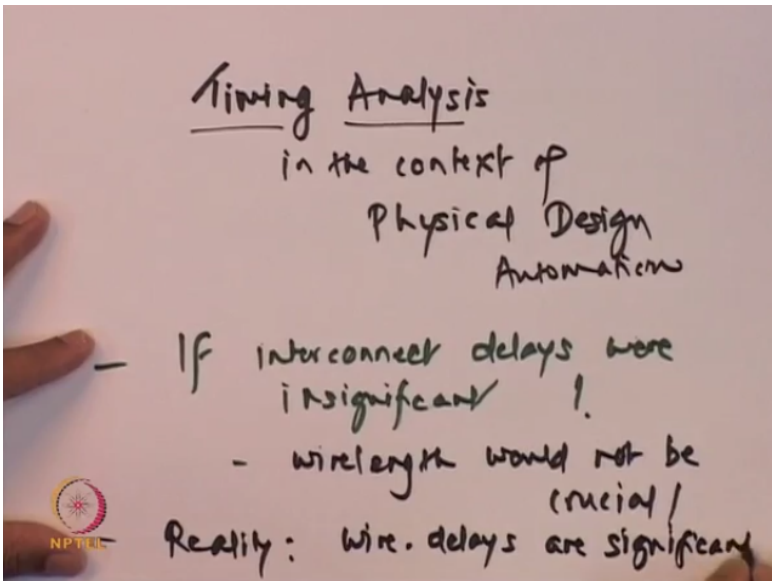
Advanced VLSI Design
Prof. Sachin Patkar
Department of Electrical Engineering
Indian Institute of Technology- Bombay

Lecture - 31
Timing Analysis in the Context of Physical Design Automation

(Refer Slide Time: 00:23)



Today's lecture, the plan is to discuss briefly sketchily role of so-called timing analysis in the context of physical design flow, okay physical design automation flow. Most of you would be aware of timing analysis in particular in fact static timing analysis you would have used in while using the tools, FPGA CAD tools or ASIC tools, guidance tools.

Just this lecture would aim to give you some brief idea behind about the core techniques, core algorithmic idea used in this timing analysis, the concepts, the notions of focused directed or cyclic graph, topological sort, couple of notions which are only present in the algorithms of this kind, okay. So first of all, I mean physical design I say it is about layout rate, so what is the connection with timing analyses, I mean what is the connection of timing analysis with physical design.

First of all, most importantly you recall that we I mean I mentioned that in the automated process of layout generation, lot of the importance is given to minimizing the wire length, right. So why I

mean first of all must a remark that if number of wires or length of the wires is very large that would kind of lead to more complexity for routing that is if you have one of rout within some limited area then of course it becomes harder to route if you have long wires, wires are running very long from and so on.

Of course number of wires is going to be same as number of nets, but if we have done a poor placement then it could possibly mean that a lot of wires are running very long. Okay from one side of the chip to other side, so that is why while doing placement one of primary goals of good placement is to ensure as low as possible total wire length.

Of course, until one does routing, one does not really exactly accurately know what wire length is, but after placement one can get some estimates of how much the wire length would be and based on those estimates, the replacement heuristics would be guided to like sort of make a better placement and so on. So why was this wire length crucial okay, so in particular suppose this interconnect delays, okay the wired delays were insignificant or negligible.

Okay then would we have been worrying so much about the wire length, of course wire length would mean like complexity of router, it will be more headaches for the router, but in terms of delay or timing, the length of the wires would not matter. If delays of interconnects were extremely insignificant compared to the delays of the gates themselves, of the modules or cells themselves, okay.

So in this case wire length would not be so crucial, but definitely that is not a situation. Of course to some extent it will be crucial in terms of area optimization, but not in terms of timing, but reality is that wire delays are significant or becoming increasingly more significant. So they have to be kind of given good attention.

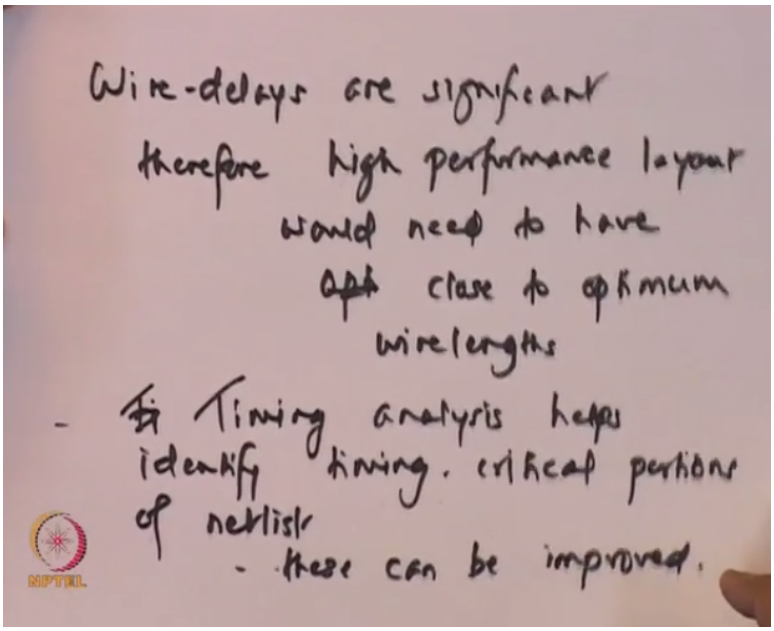
So in this sense, like because timing is very important and estimates of timings are important one would get an idea of what would be the acceptable wire length and so and so. Also like certain tools or the so-called static timing analysis would tell a designer about like there is an automation tool about what is timing critical portion of the netlist and if one gets an idea about that.

Then in a manual or automated fashion, the timing critical portion which could involve some gates which could involve some wires, those gates could be resize to improve the delays, the inertia or the wires could be widened so as to restrict the resistance or some appropriate buffer insertion would be done to reduce the delays along the long wires in the timing critical portion of the netlist.

So netlist restructuring can be done in case one gets a timing estimates as accurately as possible or as quickly as possible so one would have to kind of this will not be one-step procedure because everytime you make some tentative placement or routing layout decisions, you have one like you know feasible kind of routing, but then you might want to improve upon it and then you look at a timing and identify the critical portions trying to improve the hoping to get overall improvement in the timing and so and so forth.

So this is how the timing analysis tools will be interacting with the layout generation tool, so many of this placement routing algorithms would actually be getting a quite a bit of feedback from the timing analysis every now and then. So we aim to look at some core notions in this timing analysis with the help of examples illustrations.

(Refer Slide Time: 06:56)

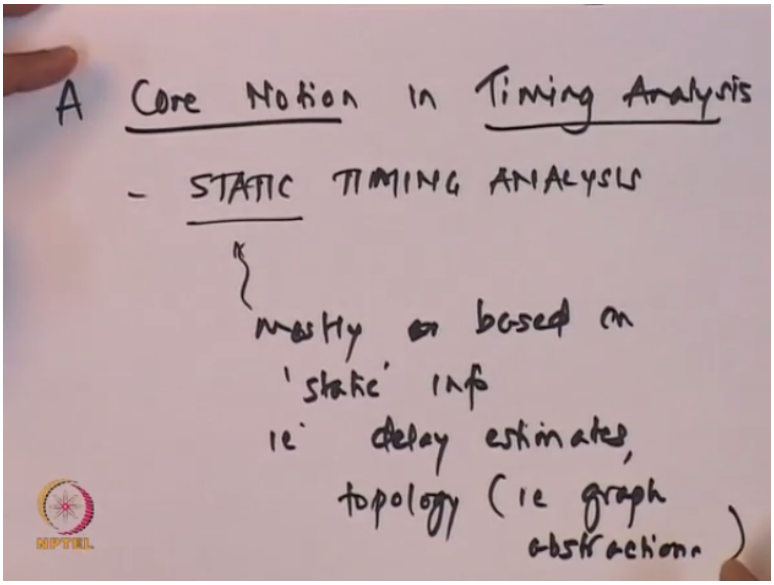


So as I mentioned since wire delays are significant therefore it is now interest to have high

performance or optimum or close to optimum wire length okay or appropriates sizing. Another point I mentioned is that the timing analysis helps identify the critical portions of netlist, and this can be improved in various ways as it is like by upsizing resizing the gates so as to reduce the inertia or like you know improving this interconnect or inserting buffers at appropriate locations.

Many of these problems have been studied for the algorithmic solutions.

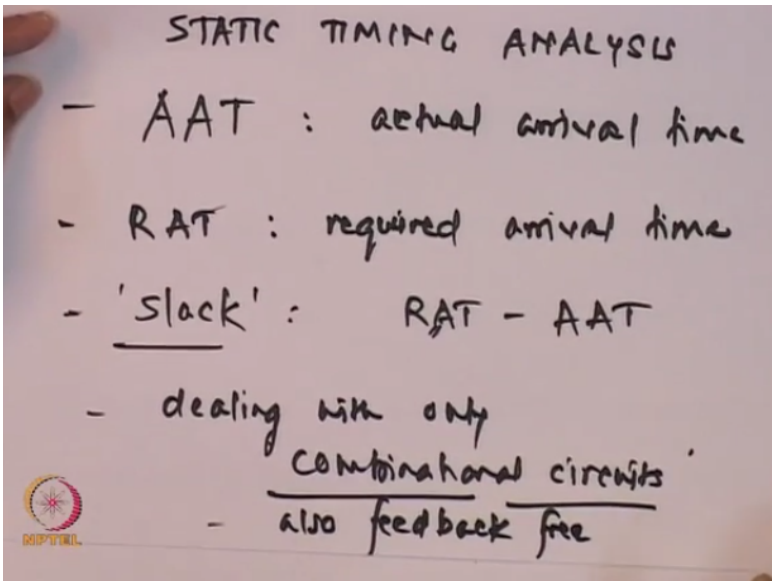
(Refer Slide Time: 08:56)



So core notion and timing analysis so one of the simplest to understand and develop some intuition for more difficult concepts and problems so that is so-called static timing analysis. So this static refers to static is not dynamic, is something that does not really put any, give any specific emphasis to the runtime behavior, dynamic behavior, more about the most, so static analysis means mostly based on static information not runtime information that is the delay estimate, worse case delay estimates.

Actually at the runtime the delays may not be as bad as the worst case delay estimates and also the topology, it is graph representation or graph abstractions. By looking at topology and worst case delay estimate one can fathom a lot, one can get a lot of useful information about timing requirements and timing optimizations and which will help in hoping to arrive at good high-performance layouts placement routing.

(Refer Slide Time: 10:48)



And you hear about static timing analysis, you will automatically hear about the terms like AAT, which stands for actual arrival time, RAT or some other notation for like mnemonics for this required arrival time, so these are the times arrival times of signal transitions. So we will obviously elaborate with the help of examples so things will be very clear and based on these two concepts, there is a notion of slack which is quite important slack.

Slack will be roughly speaking it is the difference between RAT and AAT of individual notes. It is just so in the timing analysis interesting we will be kind of dealing with only some kind of combinational, so if you start looking at timing analysis algorithm many of them seemed to be just talking about combinational circuit, so I mean what is it mean so does it mean that this things apply only to very restricted situation of purely combinational designs, no in fact this combinational portion of any design is really the core of it in terms of the logic part of it.

Other than in any sequential circuit there is going to be flip flops and there is going to be combinational logic portion, the flip-flops are being used for storing the data or the state of controller or the state of the computation data path state okay and combinational logic that is the portion of the circuit, which is actually like you no continuously working, continuously kind of updating its outputs in response to the changes at inputs.

Okay, so that is the combinational part of the sequential circuit is the one which is kind of trying

to compute the next state of computation, next state of controller and the next values of the variables, the data items and the flip flops are only like you know holding onto the state or the state of computation or the data current values of the variables, the data items and so and so forth.

So for the timing analysis the flip-flops, we will not have to worry about the flip-flops so much in fact at an abstract level, we will kind of be cutting the circuit at the flip-flops and once you get big sequential circuit at a flip flop, if it is a very descent sequential circuit then what will be left with is obviously purely combinational but more interestingly, there will not be any feedbacks within the combinational portions.

The circuit obviously will have feedback, there will be some like you know next state depends on the current state so that means and the next state is going to become the current state in the next clock cycle, so there is a roll of feedback in sequential algorithms of sequential circuits, but anytime there is a feedback then that feedback path will necessarily include one or more flip-flops.

So if we cut at the flip-flops you are going to have all the feedbacks removed and whatever the purely combinational circuit that is left is feedback free okay, of course there is nothing wrong really in having feedbacks in the inner circuit which has just combination gates put those kind of feedbacks can give you latches, can give rise to memory elements or can give rise to oscillators configurations so there is no need like for designing latches and oscillators.

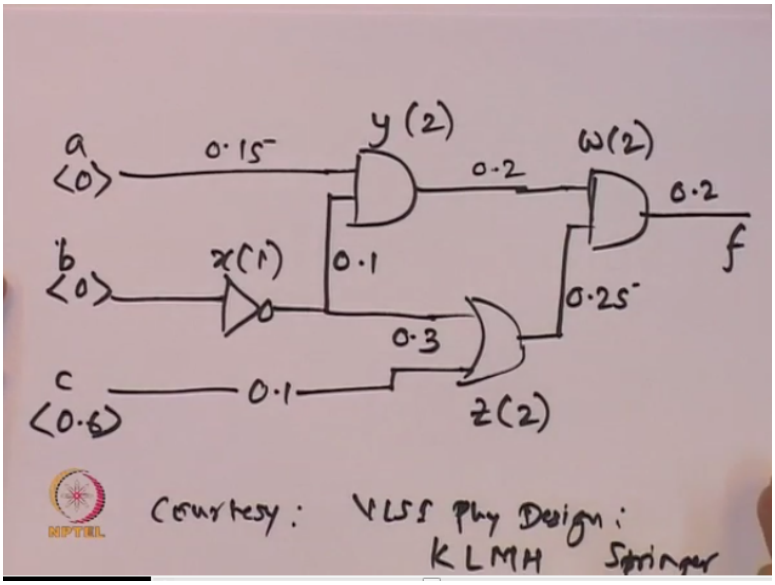
In large distance one would typically make use of them as black boxes or as well descent components and the combinational gates would be purely used in the feedback free manner for generating the next state logic or the output logic milli or more output logic, okay so one of the things about static timing analysis you will notice is that this example circuits that we will be dealing with will be shown to be purely combinational and without feedback.

Also so I am emphasizing cautioning that like feedback freeness is not the absolute necessity in designs, especially feedback free, but like in a very descent circuit the combinational part did not

have free feedbacks. If at all feedback is meant for holding like designing a memory element like a latch then you could as well use a well-designed latch in that particular implementation technology.

And it helps a lot to like you know assume that there are no feedbacks that makes the algorithm design process, analysis process very smooth and interesting and inefficient, okay. So I will take an example right away which we will be kind of exploring to shows some concepts.

(Refer Slide Time: 16:51)



So here is one circuit. I have some labels or notations over here. In about a moment, I will explain what this mean. So this is again an example curtsy. Okay, here it is 0.15, the book VLSI physical design from graph partition into timing closure by Kahng, Lienig, Markov and Hu, Springer, I think it came out last year ago or couple of years ago maybe. Good book, fairly elementary introductory and well-written.

Of course there are other books, but I chosen to use examples because this is not a core course like full-fledged course on VLSI design automation just for the overview of some concept try to keep the matter simple example and notion simple, simplistic overview essentially. In this particular example, here we have a netlist again important to notice is that there are no flip-flops here, it is purely combinational gates and also what you notice is that there is no looping, there is no feedback here.

This a, b, c, they have to be regarded as inputs and f is only output, of course you could have multiple outputs and many more inputs, much more complex such combinational circuit. There are notations, the labels here 0, 0 and 0.6, they represent the arrival time of signal transitions at these inputs, so you know from somewhere like with respect to certain synchronizing time, let us say positive age of a synchronizing clock in the system, this signal becomes stable at 0.6 nanosecond after the rising edge of the clock.

This signal is let us say immediately stable and so on, taken to the pinch of salt, this thing need not be zero. Then the labels on this wire segments are like they indicate the delays of this interconnects. The labels on the gates and this label name is y and this quantity in 2 indicate delay, the worst-case delay of this particular gate and this is the inverter with delay 1, this is or gate with delay 2 and gate with delay 2 and these are the wire delays, 0.2, 0.25 nanosecond, 0.3 nanosecond, 0.1 nanosecond.

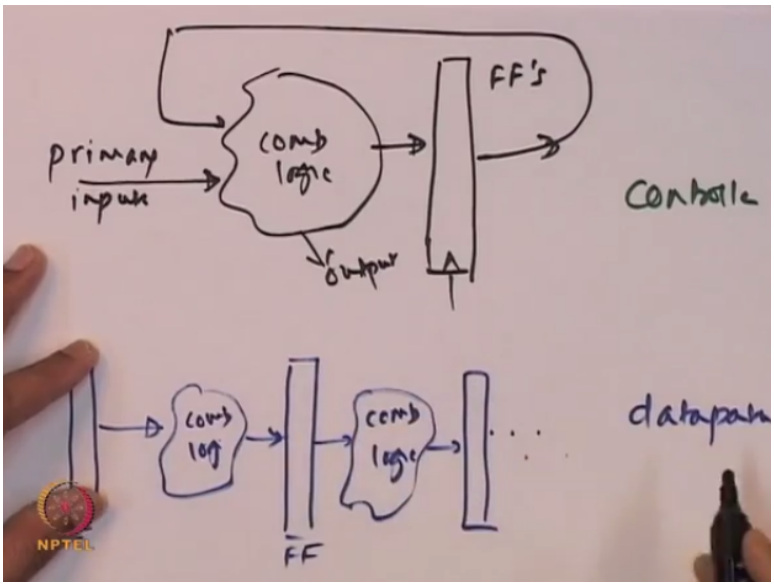
So you do not worry too much about you know this net is being driven by this, this is going here it is going here, so like you know what is the delay of this common portion, what is the delay of this portion, one can study at that level. So if anyone studies more delay, RC trees, but right now, just assume that this 0.1 means that it is a delay, the worst-case delay for the effect of signal transition at this point to reach this particular pin of this gate and so on.

So 0.3 is a delay between signal transition here and corresponding signal transition over here, okay and let us assume that for the purpose of so-called pessimistic static timing analysis the whole circuit, all the elements of circuit, the wires and gates are behaving in the worst-case fashion that is all this numbers that we are showing which are worst-case estimates, they actually like you know the signals are stabilizing or like you know making transitions with the worst-case delays, okay.

So things might turn out to be better than at runtime and those things are also studied in this subject but that will complicate our discussion a bit, okay. So based on this feedback free combinational circuit which would have been obtained like you know out of general kind of

sequential circuit which has this kind of structure.

(Refer Slide Time: 22:02)



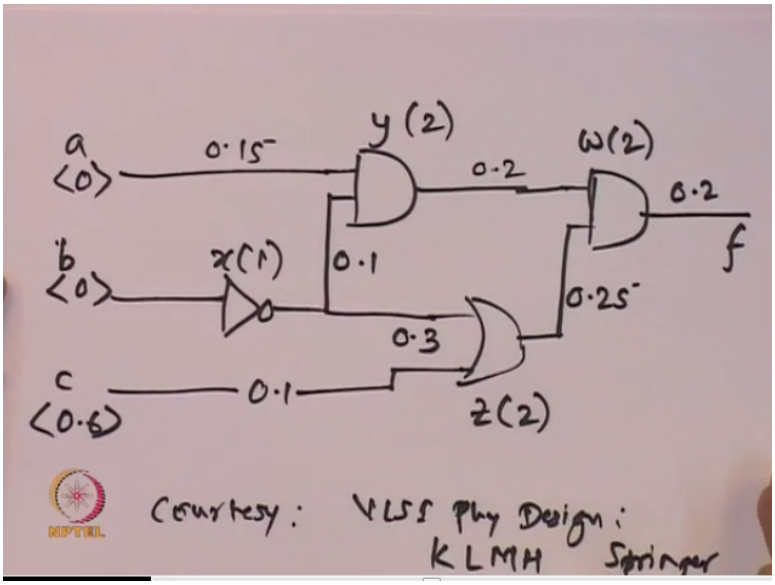
There is a controller, a controller is basically a set of flip-flops which maintain the state and combinational logic which will generate the next state based on the current state, these flip-flops maintain the current state and there is a clock and this combination logic is computing the next state which is to be loaded into the flip-flops at the beginning of the next clock cycle or the end of the current clock cycle.

Also, this combination logic is also influenced by primary inputs or inputs from some other source not inputs from this flip-flop okay. These are considered to be primary inputs of this particular control, this is the controller and then there is a data path. So the data path might have some flip-flops albeit there are some combination logics and there is a set of registers followed by and so on.

Okay, it could be this is kind of linear picture I drawn just for us, in general data path can be far more complex, there will be registers, combination logic or there will be multiplexers and so on, but again the data path you see that there are flip-flops and there are combinational logic clouds, okay. Controller also has added flip-flops for maintain a state and combinational logic for generating the next state and the outputs.

The outputs of this controller go to the data path and so and so forth, we are aware of that. We are not going back I mean like leading the discussion back to FSM + data path, but the important like here I want you to identify that this combinational logic is the one that we have to focus on for static timing analysis, just cut this flip-flops like you remove them from the picture what you are left with is possibly disconnected state of combinational logic clusters or clouds we can think of them as which are feedback free okay.

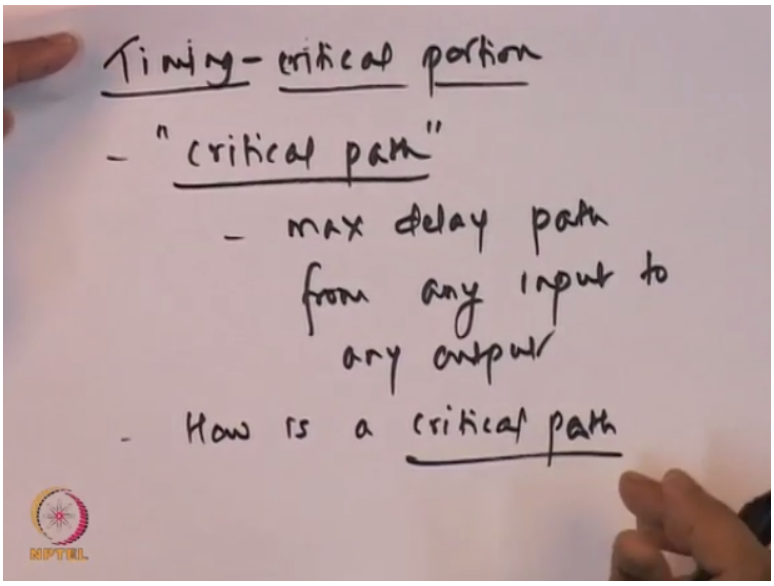
(Refer Slide Time: 24:42)



Like in this example this circuit is feedback free, of course this must be a very small part of a very small sequential circuit if at all, but in general it could be hundreds of thousands or millions or tens of billions whatever, very big, not everything would be one like you know monolithic combinational circuit, there could be lots of separate parts of it and after partitioning they can be separately identified, separately analyzed,

So partitioning has one big role in this timing analysis also, okay. So this was just a picture of a general sequential circuit which will have controller data path, so highlighting which portion is the one that we are going to look at in this timing analysis, this combination logic.

(Refer Slide Time: 25:30)

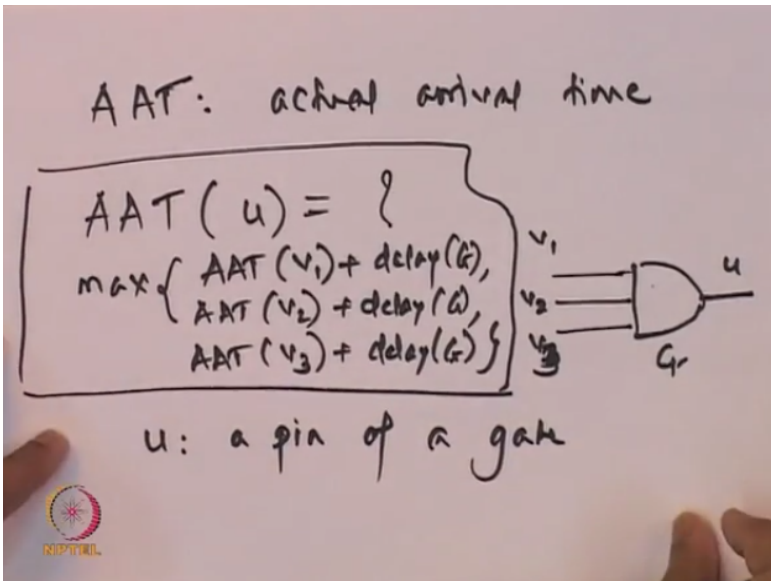


So the notion of timing critical, this is like vaguely remarked at the timing analysis, static timing analysis is going to compute this so-called actual arrival times and required arrival times which will be listed on this example, but after doing that and after computing the slacks the analysis will identify the timing critical portion and in particular.

This critical pass is an important like concept which all of you will be very well aware of, critical path is the one which is important because that is the one which has the maximum delay from any input to any output and which will like influence the maximum frequency at which the sequential circuit can be driven so that will influence the speed of your implementation.

So critical path is the path which has max delay from any input, any output, okay, so how is critical path identified. So the state of art timing analysis has much more ability than just identifying critical paths, but it will help to begin with, like you know understanding how critical paths are found and like that is a core notion, what kind of algorithmic idea is more in the graph models, graph algorithms get used that gives you some kind of insight motivation to study the next level more advanced concepts so I will just focus on the basics.

(Refer Slide Time: 27:36)



So this concept of actual arrival time, so actual arrival time at U say where U is a pin of a gate, okay, say it could be output or input pin, so this actual arrival time at a you at a pin U say U, this is a gate and this is say U okay, so how is it going to be defined and what will determine this particular actual arrival time, which is the arrival time of the signal transition.

The signal transitions are happening at the inputs and like you know because of the delays of the wires and delays of the gates, the corresponding signal transition will happen at certain time and the time at which that signal transition occurs as being that all it gates and all the wires are working, are acting in the worse case fashion with worst-case delays, the time at which this corresponding signal transition occurs and after which this signal at U achieves its correct value is the actual arrival time.

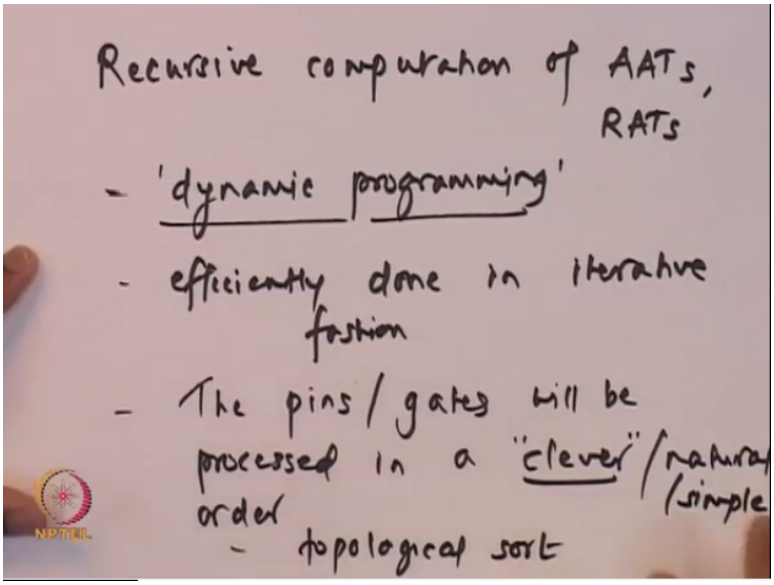
So this will clearly be like you know based on say this is V1, this is V2, this is V3, so if these are the signals at the input of this, then this will be max of AAT in this example, AAT of V1 + what the delay of the gate G, okay, AAT of V2 + delay of G, AAT of course I can say it will be much more elegant fashion, but anyway let us see how our thought ideas develop. So clearly this rate.

So it will be influenced by the one, by the pin which has the latest or actual arrival time, so this particular input might like become stable very early, but it will not like you know, it would be of no use, if some other input is going to arrive much later, so the one which is going to arrive input

that is going to arrive at the latest time instant is going to determine the actual arrival time at the output of this gate.

The delay of this gate is also include, that is common for all these cases. So it looks like you know it hints at the fact that this actual arrival time recursively defined and that is the main key of things, okay. Similarly, the notion of RAT, required arrival time, is also going to be recursively defined and I will just come to that in a moment**.**

(Refer Slide Time: 31:14)



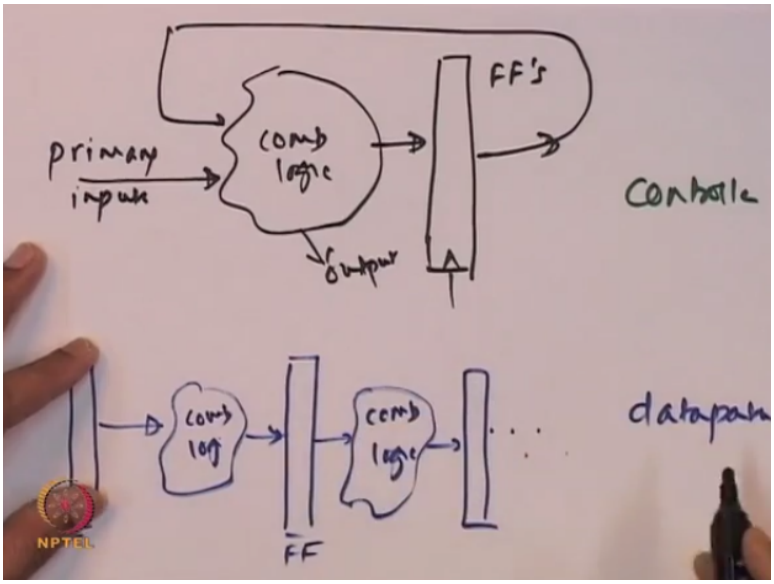
To do this recursive computation of AATs and similarly of RATs, these recurrence relations is with the help of dynamic programming. I did not mention this but this if one is interested one can look at this as note that these are very interesting important examples of dynamic programming otherwise one can also invent these ideas or discover these ideas without like you know appealing to this sophisticated notions of dynamic programming, very intuitive concepts.

So this recursive computation uses the help of dynamic programming or otherwise can be efficiently done in iterative fashion. By that I mean like we are going to compute the AATs at all pins of combinational circuit which does not have feedback, we are going to start at the inputs and then like iteratively compute the AAT, actual arrival timings at different pins, you know insert in order.

The pins or gates will be processed in a clever but natural and simple, clever idea like so called topological sort, clever, simple, natural okay. So I do not want clever to necessarily mean that it is something very hard to fathom or understand, very simple idea. So in some kind of topological sort again its one of the very simple notions in algorithms and quite likely you are already familiar with it.

It is going to be used to like you know visit these notes, the gates or pins and set an order and compute at AATs and similarly for RATs, may be the order would be different over there but the idea is the same.

(Refer Slide Time: 34:05)

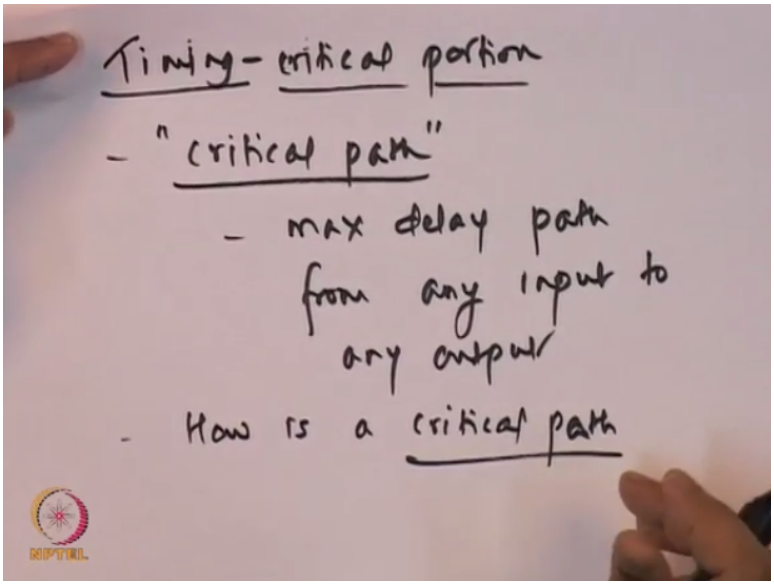


Incidentally I think I have missed mentioning one small point, one of the reason I was like you know bringing your attention to the flip flops and like you know while talking about in that we will see the focus on combinational circuit, although there is most genuine like digital designs filled of flip-flops, the flip-flops are for the purpose of storage, combinational logic is for the purpose of computing the next state, doing the data processing and computing the outputs.

So what are the so-called inputs of this combinational logic and outputs from the combinational logic. Of course some of the inputs from the combinational logic are coming from the external world and some of the outputs are going to the external world, so they are primary inputs and primary outputs, okay.

But other than that we should regard the outputs of flip-flop, also as kind of secondary input to combinational logic like output of this flip-flop can be regarded as an input to this combination logic and to differentiate it from the input coming from external sources, I will call it secondary input. Similarly, output of this combination logic is not going to the external world, but is going to like over here, it is going to flip-flops again, so this is to be regarded as a secondary output of this combinational logic. So should be in some sense regarded as an output.

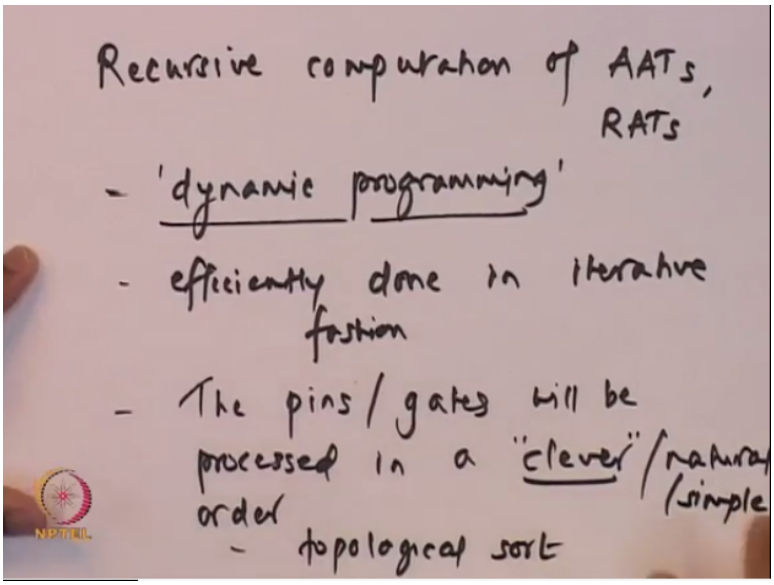
(Refer Slide Time: 35:27)



And that is of course crucial because this notion of critical path that I mentioned which you are really familiar with which says, it is max delay path from any input to any output, but that any input, any output include not just primary input, primary output but also this secondary input, secondary output which are respectively the flip-flop outputs and flip-flop inputs, okay.

So when says loosely like critical path is the max delay flip-flop to flip-flop path, but also should include primary inputs and primary outputs. So basically any input or output path where input-outputs could be either of the primary kind or the secondary kind.

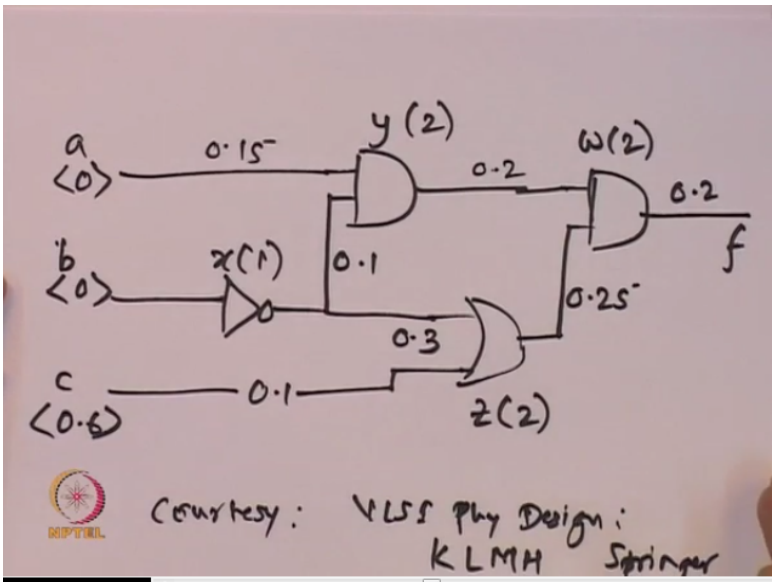
(Refer Slide Time: 36:18)



So I am sorry for the digression but we are just going to get some more idea about how to compute actual arrival times and required arrival times and I say they are like at a sophisticated level one can think of all these as very good example of so-called dynamic programming very important idea in algorithms as per as in particular VLSI CAD has many applications of this and this notion of topological sort some kind of ordering.

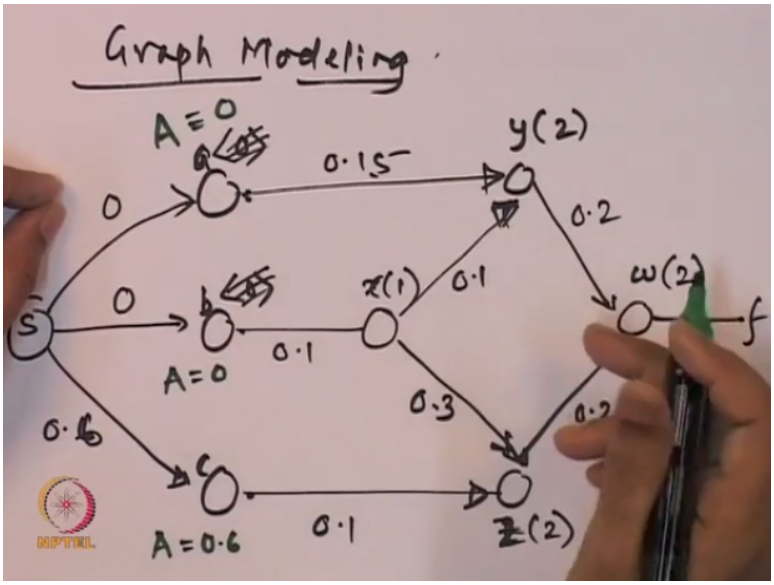
Anyway this is what you will get to study when you do that course on design automation or course on algorithm data structures. So a graph modeling is preferred, okay one can do this on the netlist itself, but like to make good use of standard packages, standard libraries for implementation one should like do a clean analysis, clean algorithm development and use of libraries like the abstractions are very useful so a graph modeling and again there is no just single particular way of modeling things by graph.

(Refer Slide Time: 37:26)



So that circuit that I drawn this one, I am going to modulate as a graph.

(Refer Slide Time: 37:30)



So I will just draw the graph first, recall that there are a, b and c, there are this primary inputs or it could be secondary input, but inputs and there were labels here, 0, 0 and 0.6, okay. So this label we are denoting the arrival times of signal transitions at this input, so I am going to have some kind of dummy source and arcs of this kind, directed edges of this kind and this I am going to label with this.

So this is going to mean that a signal transition arrives at this input c at 0.6, but at a and b, it is there from time zero itself, okay. Then a is connected to input of the gate y, so I am going to

represent y by one node okay and I am going to label this by 2, remember that y is an end gate with worst-case delay 2 and this connects this wire from a to y has delay of 0.15, so that IO am going to represent it as directed edge because this is the direction in which the signal travels is 0.15.

Similarly, I am going to draw the rest, so B drives this inverter x which has delay 1 and I think I forgot this first 0.1, so 0.1, this is x with delay 1, output of x drives y as well as this gate z, so z is or gate with delay 2, z is driven by c as well as by x. The length of delay along this wire from c to z is 0.1, this delay is 0.3 like you know do not go by the lengths, they could be because of the width of the interconnects and some other factors, okay.

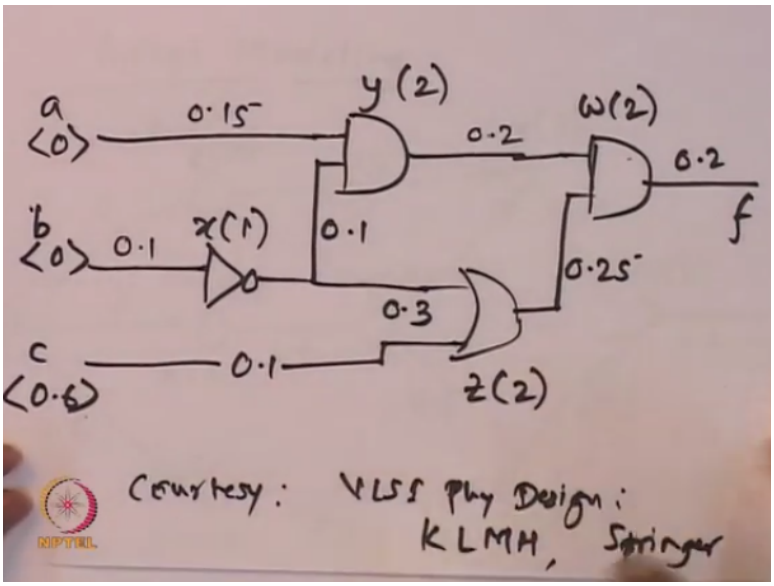
X also drives y with delay 0.1. so this 0.1 and 0.3 correspond to this x driving this pin of y up to 0.1 delay and this with 0.3 delay. C driving z with 0.1 delay, okay. So let us complete the rest of it, like there is another gate called w which is and gate with delay 2 which is driven by y. Delay of the wire is 0.2 and also driven by z, this delay is 0.25 and this is generating as shown in that picture like it is going to output and wire connecting the output of w to this output f is of delay 0.2.

Okay, this is your f, so I have introduced this dummy source for the purpose of kind of capturing this arrival time information at the inputs a, b, c, so a, b, c are not really gates, think of them as some kind of input pads or some input terminals at which signal transitions arriving at certain specific time, time zero, time zero and times 0.6 so all information has been captured here and you should convince yourself that like this will suffice to like you know the sense of the timing information, the input information that is delays of wires and delays of gates.

Now we can start building up this propagating actual arrival time information and similarly the required time arrival information so actual arrival time at a, b, c are 0, 0 and 0.6 okay, so it is going to be a bitter of character here, so I say A stands for actual arrival time, A at this is going to be 0, A at this node is zero, A at this node is 0.6, yeah that is enough because signal comes here, these nodes are not really gates or you can think of them as a simple dummy gates with 0 delay.

So at the output of this like over here we know that the arrival time is 0, 0, arrival time is 0.6 now based on this for which of the other nodes their outputs we can find arrival times.

(Refer Slide Time: 42:42)



The signal is arriving here at times 0 so based on this can we decide what would be the arrival time of signal at the output of y, not exactly I mean we know this information, this 0.15 +2 but this signal transition here to a final value possibly could depend on some other input path. For example, it could depend on a signal going along this wire getting delayed here, getting converted and other delays.

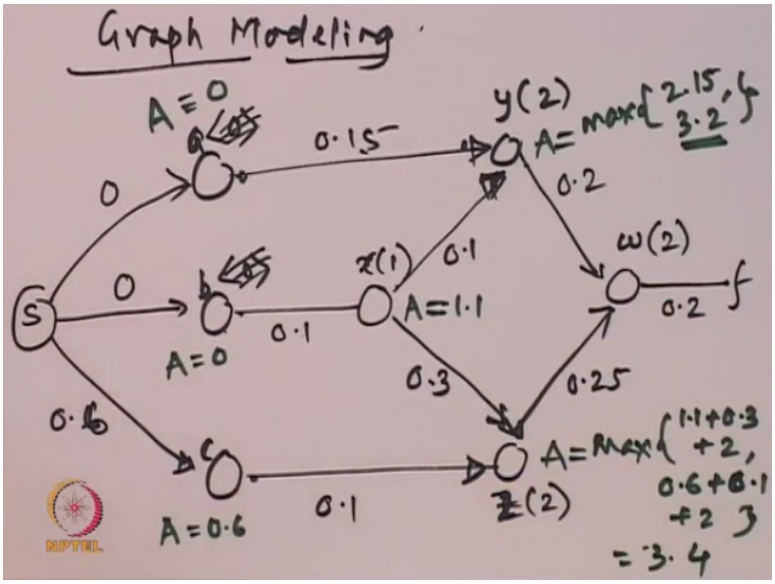
So this would have led to 0.1+1 that is 1.1+0.1, 1.2+2, 3.2 okay. so just knowing this alone would not give this knowledge 0.1 and 0.2 will not give us the actual arrival time here, okay. To know the actual arrival time at this point you will need to know the actual arrival time over here and over here. To know this, you will need to know the actual arrival time of this okay and so on. So we have to kind of process this, visit this in certain order.

So before visiting y we should visit x simply because y depends on x, okay so this so-called topological order is ordering of these gates or nodes as in graph in such a way that we visit a node only after we have visited all the nodes on which it depends. So we will visit y, we will process only after we process a and we process x, because y depends on a and x, okay x we can process as soon as we know information about b.

Similarly, z we will process only after we have processed x and we have processed c. C would have processed quite quickly, once we have processed x and a, b, c then we are eligible to process y and do a computation at z, and once we have computed at z and we have got AAT also at Y, then we can go onto computing this at f or the output of this w and so on. We have to go in this particular order.

And that is an example of so-called topological order that is simply visit the nodes in a manner such that you do not visit something before having visited all the nodes that it depends on, okay.

(Refer Slide Time: 45:17)



So topological order here will be first we will do x, because x does not depend on once we had done a, b, c, x can be done immediately but neither y nor z or w can be done because they will depend on x, so what will be the arrival time of the signals transitioned at the output of x, A at this point will be see that because of this path 0+0.1+1, 1.1, there is no other way output of x will be influenced. So it is 0+0.1+1, 1.1, so A is decided over here.

Now, A is decided at this point and at the output of x. Now because of the signal has made a transition here to its correct value after another 0.1 there will be a stable value available at this input pin of y and at this input pin of y the stable value is available at 0+0.15 that time, then there is a further delay of 2 nanoseconds, let us say at this gate y, so at the output of y now we are

ready to compute A, actual arrival time and that will be max of how much 0+0.15+2 that is 2.15, 1.1+0.1+2 that is 3.2.

1.1 at the output of this +0.1+ the inertia of this gate that is 2, so that is 3.2, so this will be 3.2. Similarly, A over here can be computed because it is going to be max of 1.1+ 0.3+2, 0.6+0.1+6+0.1+2 which is equal to 3.4. And similarly what will be the actual arrival time at the output over here, A will be, this is 3.4+0.25+2 that is 3.65+2 that is 5.65 and from this side it is 3.2+0.2 that is3.4+2 that is 5.4, so a maximum is 5.65.

And at f, A will be 5.85, okay 5.65+0.2, so it is a very simple calculation, very easy to implement as an algorithm in any language like it is just looking at topology we define, figure out an order in which this node should be processed, all labels are indicating all the information that you need, the delays of the wires, delays of the gates and based on that in a systematic in a very linear efficient time you can get all these values, actual arrival times at the outputs of all the gates in this particular design.

So, now that means we know as a result of signal transitions arriving at a, b, c at this given times, at the time 5.85 we will get a correct value of signal f as a result of I mean correct combinational value at signal F okay, so now the related question is like if this is the actual arrival time what like you know what would have been the required arrival times of the signals at different nodes, was it really necessary that at A the signal should have arrived at time 0.

Was it really necessary that at this c the signal should have arrived at 0.6 could it have been that signal would have been allowed to arrive a bit later, so what are the required arrival times at the nodes. Now we know that we are being able to receive a correct value at time 5.85 and that is because of these assumptions that we have made about arrival times here, this is the actual arrival times, but what could be the required arrival time so that we meet a certain target required arrival time at the destination at this sink at a target.

So that is a dual kind of, opposite kind of question and as your intuition would be telling you that it would really amount to making a backward pass, here we are going in some kind of forward

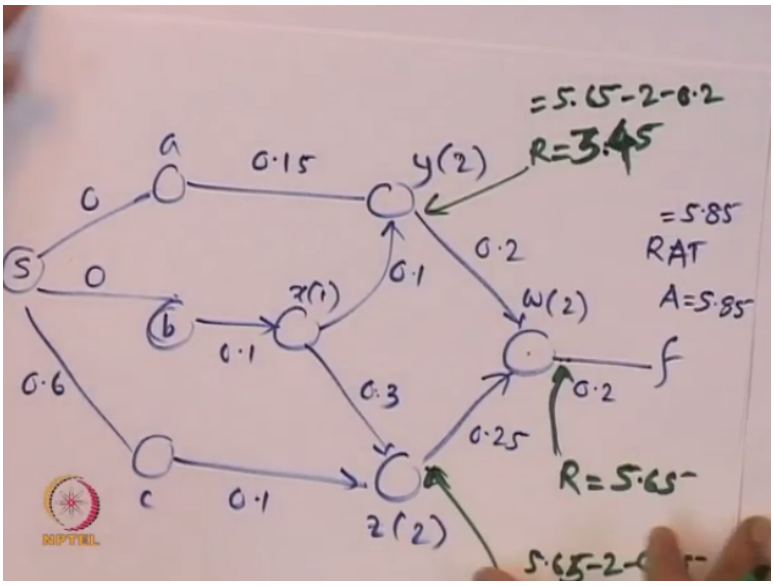
direction of the signal flow, okay we process a node only after we process all nodes which drive that particular node, so that is we are going in the direction of the signal flow.

If you go from the target in the direction backwards, reverse direction of the signal flow, we will be able to similarly compute this logical like intuitive notion of required arrival times and why is that required arrival time useful or interesting is because we get to know like whether we have been a bit to like you know too pessimistic and like you know trying to generate a signal very early.

We could have afforded a bit of delay, the signal might have at certain time it pins, the signal might arrive later and still we will have a computation or like you know our result available at the required time say at 5.85 okay. Definitely 5.85 is the best that you can get because these are the actual arrival times of signals, but could the signals have arrived late, supposing we get an information that signal can arrive at node A at time 0.2 okay and still we will be able to get at time 5.85 correct value.

If the required arrival time at a is found as 0.2 then we know that there is some kind of slack over here 0.2 that means we could have some circuit that is generating this signal transition here, we can afford it to be less efficient in terms of delay and generate this signal transition a bit later, okay.

(Refer Slide Time: 52:28)



So having computer the arrival times we know that A here has been found to be 5.85 okay. Now we say that this is also the required arrival time RAT is also 5.85 okay, we require this as like you know f has been given by this actual arrival time. So let us say we require it exactly at this time. Then we go back analysis to see what are the required arrival times of signals at the other nodes.

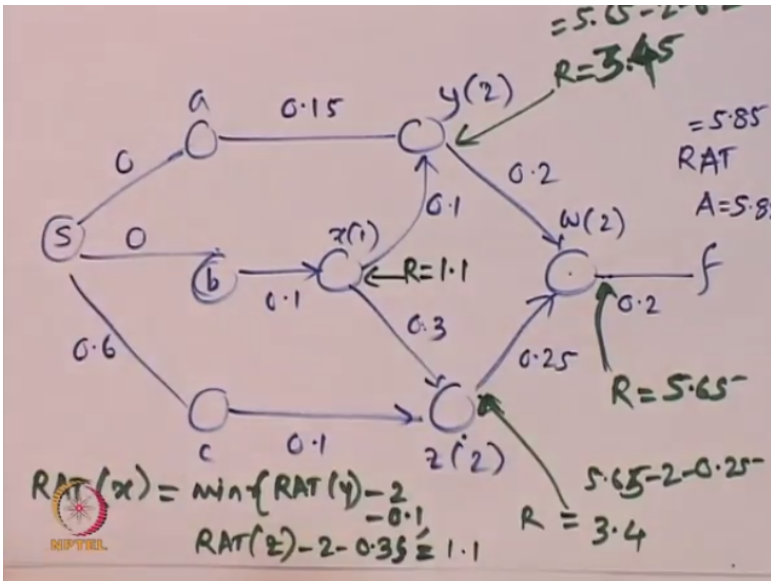
So here we want it to be at 5.85 then over here the required arrival time will be clearly 5.85-0.2 that is 5.65 okay. Then can we immediately go back to finding the like you know guessing the required arrival time over here, no right, because the required arrival time here would like you know intuitively naturally depend on the required arrival times here and so on.

So we have to patiently in certain kind of opposite order in a direction opposite to the signal flow that is in this direction and process nodes, so w is known. So we hope to find the required arrival time here and required arrival time here so how much at z. At the output of the gate w, it is 5.65, so over here at output of z will be, this is required at 5.65.

Because of the 2 ns delay of this and 0.25 ns delay of this wire over here at output of z, we must require the signal transition to take place at 5.65-2-0.25 that is 3.4 again. It just happens to be same as A, like understand the reason for this phenomenon that why it turned out to be the same that bit later over here, so this is 5.65 we required the signal, there is a 2 ns delay, 0.2 ns delay.

So we require this at 3.45, like 5.65-2-0.2, right, 3.45. Can you compare it with the actual time here? This output of the actual arrival time was 3.2, over here it was 3.4.

(Refer Slide Time: 56:33)

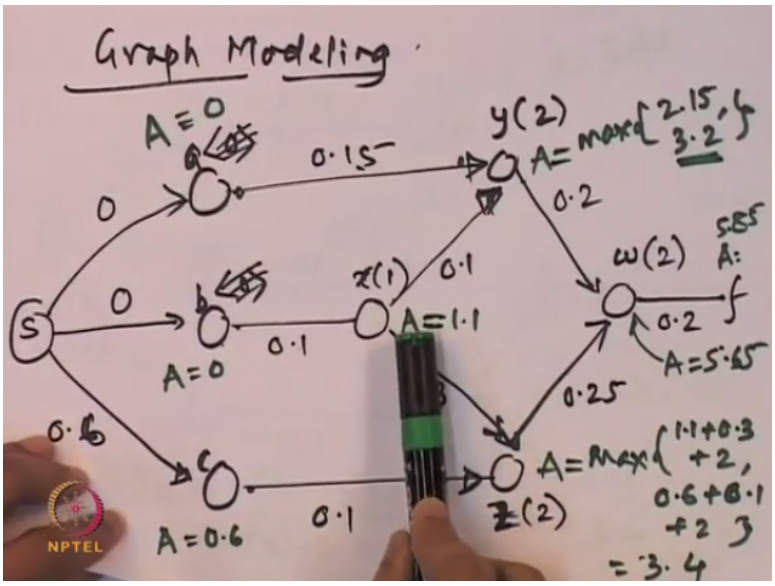


Somehow maybe a coincidence but like actually it is not, over here we have found this required arrival time at the output of z to be 3.4 which is same as the actual arrival time at the output of z, which is 3.4 but the AAT at output of y is 3.2 but the required arrival time at the output of y is 3.45, so there is a discrepancy here and there is like you know thing matching over here. Anyway, so just you would have probably guessed the reason for that also.

So this is going to tell us something about critical path, okay, now having known the required arrival time at y and at z we can compute a required arrival time at x and that will be how much will that be, that is interesting, so RAT at the output of x. By x I mean that output of x is going to be, you can check for yourself. I will encourage you to kind of think about it on your own because all these are very intuitive natural concepts.

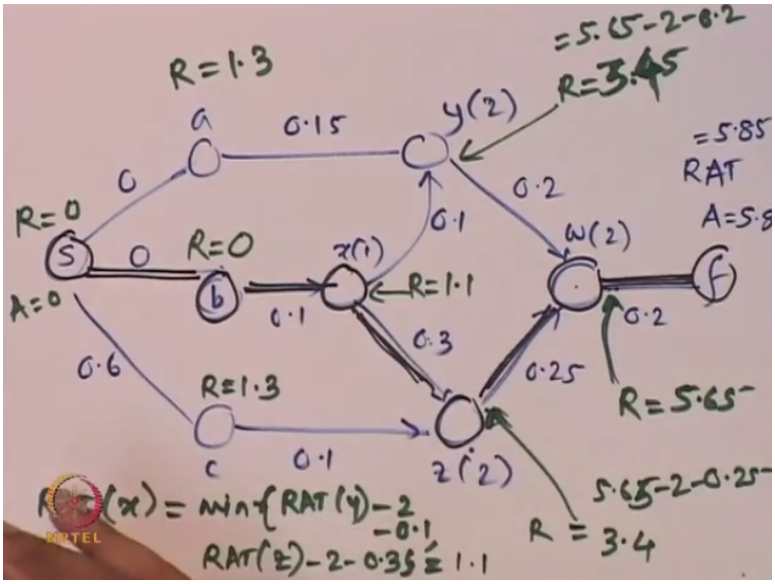
Even though, you may not have done a formal course on this subject or algorithm, your natural thinking will lead you to such recurrence relationships. RAT at x is going to be minimum of RAT at y - whatever, 2-0.1, -2-0.1, RAT at z -2-0.3 and this will turn out to be 3.45-2 that is 1.45-0.1 that is 1.35, that is this term and from here 3.4-2 that is 1.4-0.3 that is 1.1, so 1.1 is going to be 1.1.

(Refer Slide Time: 58:40)



Once again you notice that this is matching this particular actual arrival time of 1.1 over here.

(Refer Slide Time: 58:44)



Again, it is not a coincidence so let us. In fact, wherever we find things matching like you know 5.85 and by design I have chosen this RAT also to be 5.85 and then RAT here was 5.65 which is matching this actual arrival time of 5.65. So let me like you know color it differently, okay. Similarly, here R and AR matching 3.4. See that 3.4 here and 3.4 and here also they are matching okay.

So from this let us quickly compute the remaining, once we know the RAT here, RAT at A is

known and that will be R here will be 3.45-2 that is 1.45-0.15 that is 1.3. Please compare it with and it is obvious to note that here AAT was 0 but RAT is 1.3 that means you know we got the signal here a bit too early. We not even got a signal transition, here it is time 0, it could have got it as late as time 1.3 and still you would have been able to make it that is the meaning of it.

So now over here this is 1.1, this is 1 and that is 0.1, so it is actually is 0, that is interesting, it is exactly matching the actual arrival time over here and that is why I will like kind of mark this, so I will mark all the things along which I am seeing things R and A matching. And over here, it is going to be 3.4-2-0.1 that is 1.4-0.1 that is R is going to be 1.3. So here A was 0.6, but here it is now it is 1.3.

So they are not matching so now based on that R at this will be 1.3-0 or minimum of that, minimum of 1.3-0 or 0-0 or 1.3-0.6 and that is really going to be 0. A was also 0 because this is kind of dummy source arrival at some signal it is at times 0 here and after 0.6 it arrived here and so on. So this also I will mark it.

So what I have marked over here is a path along which I have seen this R and A matching, provided this RAT assume that I required a signal transition at f to be same at time at which it actually arrives, so RAT and AAT were matched over here and based on that, I retraced and computed RAT and along this path maybe along some other path also it could have been matching but definitely along these path things matched.

What is it? tell you, in fact this is an example of a critical time which is timing critical in the sense that if these components here, the gates here are like you know, if the design happened to be poorer then the timing will become worse or on the other hand if they become better marginally, then actual timing can improve, it is not necessary that they become much better, the timing improves that much better, but there is sensitivity to that.

So the timing of performance of this is going to be sensitive to the timing performance of the individual, the wires along this path, the gates along this path, so this is what is considered to be critical path and a lot of attention can be given to this critical path, things can be resized,

restructured, the interconnects can be made more like efficient in terms of delay, buffers can be inserted along longer wires of this kind 0.3 and things can be improved and design will improve.

So this is what roughly what happens since timing analysis. After doing this AAT and RAT calculation, the difference is calculated wherever there is a discrepancy that means there is a slack, things could have been worse, but where there is no slack that is along this, RAT and AAT are matching, the slack is 0 that is a critical path and that helps us to identify the critical portions which can be improved for better timing and effectively better layouts.

So this is how the timing analysis can give feedback to the layout automation algorithms, so I did not show the slacks, slacks were just differences between the R and A, I am not describing this formally, I just wanted to bring out the intuition and insights, it is something quite simple. So you are most encouraged to read the book by Kahng, Lienig, Markov and Hu, there are two titles, various physical design from graph algorithms, graph partitioning to timing closure.

In fact, the last chapter of this book is on timing closure, bulk of the book is about physical design and last chapter tells you how timing analysis interacts, what kind of role it has in the context of physical design automation and so on. So there are some more interesting ideas that is the timing analysis again is a big subject by itself with lot of interesting research contributions which should appeal to you.

You are most encouraged to take a look at it and help of our specific course on VLSI CAD.