VLSI Physical Design
Prof. Indranil Sengupta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 34
Time Closure (Part3)

So, in the last lecture we have been talking about static timing analysis. So, if you recall in STA what we were doing, given a combinational circuit netlist typically at the level of gates we were calculating for every interesting points of the circuit something called actual arrival times, required arrival times and the difference that is the slack. So, by looking at the slack we can tell which of the nets are which of the segments of the net list are critical, in the sense that some of the slacks may become negative. If a slag becomes negative, then during a subsequence step we have to adjust the delays and some sizing of the gates or wires or inserting buffers. So, that slacks can be made positive, now there is another thing to be considered as well.

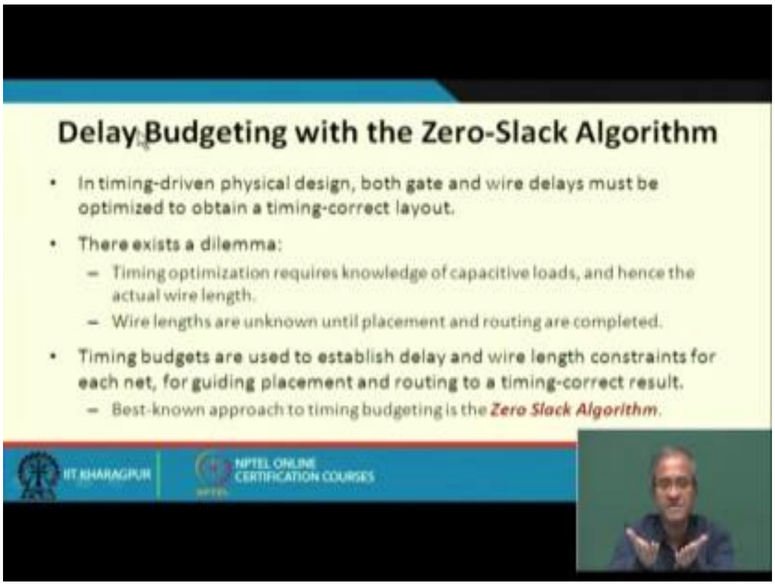
Suppose we find that our static timing analysis is not giving any timing violations, but what we find is that in many of the lines we have a very high positive slack value. What does a positive slack mean? It means our required arrival time is larger than the actual arrival time. Suppose we find that it is significantly larger. So, why would you want that? Would not like to have that, because we want to have a circuit that is running as fast as possible, so if unnecessary give a very large positive slack; that means, I am trying to run the circuit at a slower speed.

So, even if this signal has arrived means my requirement is this I am unnecessarily making this signal arrive earlier. So, I can introduce some delays, I can make some gates smaller, I can do some other kind of optimizations, as to make this slacks as close to 0 as possible. That is one of the means optimization criteria in the step right. So, I do not want to you can say compromise on the quality of the timing, but I can reduce the complexity of my design by introducing delays in some places.

By introducing delays, I can make some of the positive slacks close to 0, and if it is close to 0 then I am happy with my design, that I am trying to do some kind of optimization with respect to the resources that are available to me.

So, in this lecture, we shall be actually taking about one kind of an extension to the static timing analysis, that is very well known algorithm called the 0 slack algorithm. So, process by which we try to make this slacks 0 in all the lines. So, let us look into it.

(Refer Slide Time: 03:28)



Delay Budgeting with the Zero-Slack Algorithm

- In timing-driven physical design, both gate and wire delays must be optimized to obtain a timing-correct layout.
- There exists a dilemma:
  - Timing optimization requires knowledge of capacitive loads, and hence the actual wire length.
  - Wire lengths are unknown until placement and routing are completed.
- Timing budgets are used to establish delay and wire length constraints for each net, for guiding placement and routing to a timing-correct result.
  - Best-known approach to timing budgeting is the ***Zero Slack Algorithm***.

So, we use a concept called delay budgeting, which means at the output of each gate we are putting some delay budget; that means, how much additional delay we required there. So, so once we have the delay budgeting at a subsequent step we can do some physical process by which you can actually increase the delays in those lines by some means.

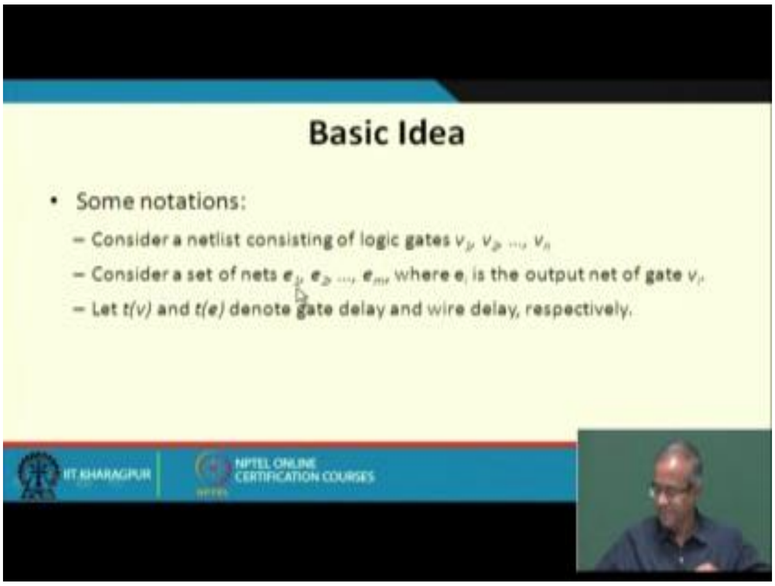
Delay budgeting means at the output of some gates we can specify that well I need 2 additional units of delay here. I need 3 additional units of delay here and so on. Delay budgeting concerns that, that some delay values I dynamically calculating in a alterative way and at the end it will give me by how much units I have to increase the delay in certain lines. So, there are a few interesting points here.

First is that in timing driven design, we are typically considering both the wire and gate delays right. So, we are trying to optimize them, but there exists a dilemma of which will has to be done first like for example. So, when we do timing optimization. So, we are considering weights of the wires, we are considering wire delays. Which means that we have good knowledge of capacitive loads we have good knowledge about the actual wire delays because unless we have knowledge about the capacitive loads. So, our estimates of the wire delays can be quite inaccurate right, this is the point to note.

And the second point is that unless placement and outing are completed we do not know the wire lengths. So, unless we know the wire lengths, we do not know the capacitive loads. So, means unless we do the timing optimization we cannot do placement and routing in an optimum way. So, there is dilemma which one of the 2 to do first. Shall you do placement and routing that are timing aware first or shall you do the timing optimization first. So, because of this dilemma typically this process is integrated as much as possible they are done I means almost at the same time hand to hand. So, in the 0 slack algorithm, we are using the concept of a time budget. So, what is the time budget? For every net we are establishing some delay constrains in terms of the wire length in terms of the gate delays.

These time budgets can later on be used for fine tuning placement and routing. So, that these delays and wire length constrains can be met. So, we shall see we shall illustrative through an example also, but how it works.

(Refer Slide Time: 06:40)

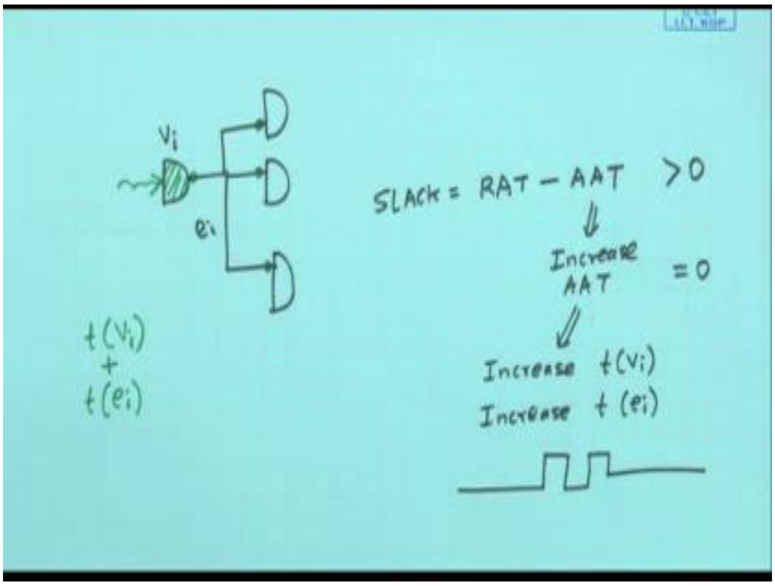


Basic Idea

- Some notations:
  - Consider a netlist consisting of logic gates $v_1, v_2, \ldots, v_n$
  - Consider a set of nets $e_1, e_2, \ldots, e_m$, where e_i is the output net of gate v_i.
  - Let $t(v)$ and $t(e)$ denote gate delay and wire delay, respectively.

Let us look in to it, first let us introduce some notations. So, we consider a netlist consist of a number of gates let call the gates as v 1, v 2 to v n. Then we consider a set of nets like, what is a net.

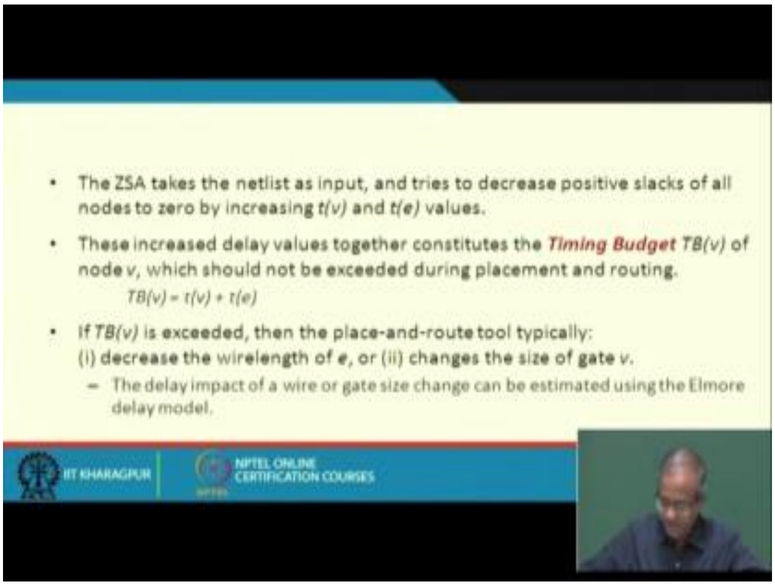
(Refer Slide Time: 07:00)



Suppose the output of a gate goes to the input of 3 gates let say, suppose this gate is v i then this interconnection this output is going to 3 place. This is a single net this net I am call it as e i. So, e i is the net corresponding to the output of gate v i, right. So, so we consider a set of such nets which indicate interconnections where e i is the output net of gate v and t v and t e will denote the gate delay and wire delay respectively.

So, here for example, in this example when I say t of v i it means the delay of this gate and when I say t of e i, it will consider the delay of this whole net the output of this gate going to the inputs of these gates. So, the worst case delays of this entire net right. So, we have 2 parameters t v i plus t e i. So, when we say that some signal is coming to the input of v i and it is reaching the input of the next gates. So, the total time will be t v i plus t e i right. So, these are the notations.

(Refer Slide Time: 08:35)



- The ZSA takes the netlist as input, and tries to decrease positive slacks of all nodes to zero by increasing *t(v)* and *t(e)* values.
- These increased delay values together constitutes the ***Timing Budget*** *TB(v)* of node *v*, which should not be exceeded during placement and routing.

  $TB(v) = t(v) + t(e)$
- If *TB(v)* is exceeded, then the place-and-route tool typically:
  (i) decrease the wirelength of *e*, or (ii) changes the size of gate *v*.
  - The delay impact of a wire or gate size change can be estimated using the Elmore delay model.

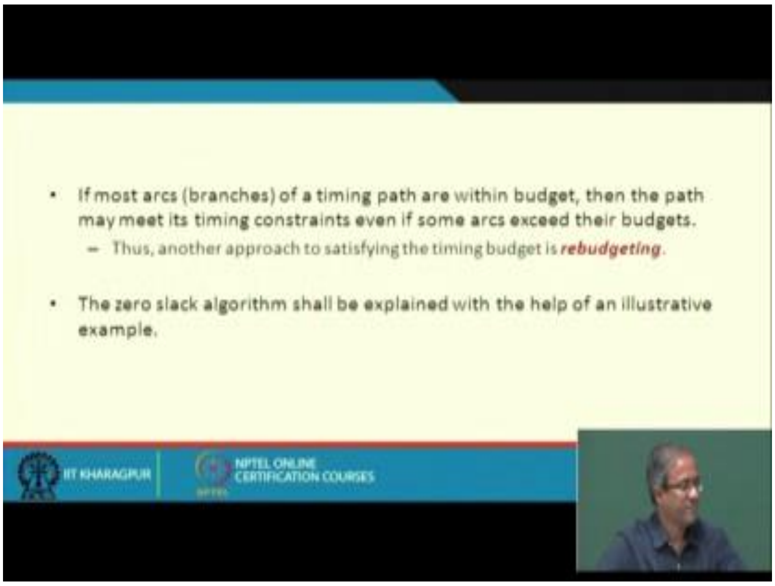
This 0 slack algorithm what it does, it takes the net list as input and, as I had said well here I am assuming one thing that there are no negative slacks. If there are negative slacks I have already meet some adjustments. So, that the slacks are become all positive that I am assuming. So, once I have a netlist with this slack values calculated and this slacks are positive to start from that point.

So, we try to decrease the positive slacks and try to make them 0. So, how we are trying to decrease the positive slacks? So, we can decrease the slack, slack is what required arrival time minus actual arrival time. So, how I can reduce this slack? You see just to recall slack is defined as required arrival time minus actual arrival time. Suppose slack is a value which is greater than 0 positive which means RAT is greater than AAT. So, how I can make it 0 I can make it 0 by increasing AAT. So, how I can make this AAT increase? So, I can either increase the delays of the gates like t v i, or I can increase the delays of the nets t e i. So, delays of the gates I can do by scaling down of the gates making them smaller. So, increase of the interconnection delay I can make by appropriator routing. So, instead of a straight router maybe I can make a zig zag kind of a rout and then rout like this.

So, roughly the idea is this fine. So, including t v and t e, what we are doing is that we are trying to decrease the slacks to 0 by increasing t v and t e that constitutes the timing budget. So, the timing budget for a node or a vertex v is defined as, the delay of that particular gate plus the delay of the output net. So, this corresponds to the total timing budget of that node. So, once the timing budgets are calculated and the slacks have been reduced to 0. Your objective will be to try and achieve this timing budget by appropriately adjusting the physical parameters of the layout either making the gate smaller making the wires longer and so on. Making the wires thinner there are so many ways.

So, here as I had said if t v is exceeded, so you have to make it smaller. Then you can decrease the wire length you can increase the size of the gate, but if is the other route you have to do the reverse. Now the delay impact is often estimated using the Elmore delay model because it gives a quite accurate estimate provided you have made a relatively accurate estimate of the capacity when the resistive, I mean effects of the neighborhood the other lines. So, once we have had a fair estimate of the resistance and capacitance, we shall be talking about this later how we can make these estimates. Then you can use the Elmore delay model to make a fear estimate of the delay how we can make that calculation right.

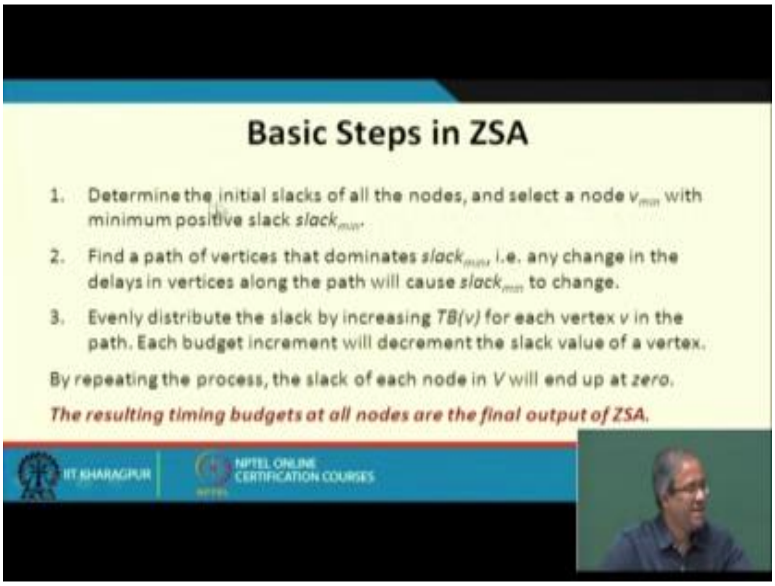
(Refer Slide Time: 12:23)



- If most arcs (branches) of a timing path are within budget, then the path may meet its timing constraints even if some arcs exceed their budgets.
  - Thus, another approach to satisfying the timing budget is ***rebudgeting***.
- The zero slack algorithm shall be explained with the help of an illustrative example.

So, the point is that if we find that in a timing path, if most of the where I mean edges are within the timing budget, then it is possible that the path can meet the timing constrains the entire path even if some arcs cross their timing budget. So, there is a process curve rebudgeting means it is an iterative process, where the time budgets are calculated and modified in an iterative way. So, at the end you get the final values which you have to meet at the end right, this is sometimes called rebudgeting. So, we shall be explaining the 0 slack algorithm with the help of an example.

(Refer Slide Time: 13:11)



Basic Steps in ZSA

1. Determine the initial slacks of all the nodes, and select a node v_{min} with minimum positive slack $slack_{min}$.
2. Find a path of vertices that dominates $slack_{min}$, i.e. any change in the delays in vertices along the path will cause $slack_{min}$ to change.
3. Evenly distribute the slack by increasing $TB(v)$ for each vertex v in the path. Each budget increment will decrement the slack value of a vertex.

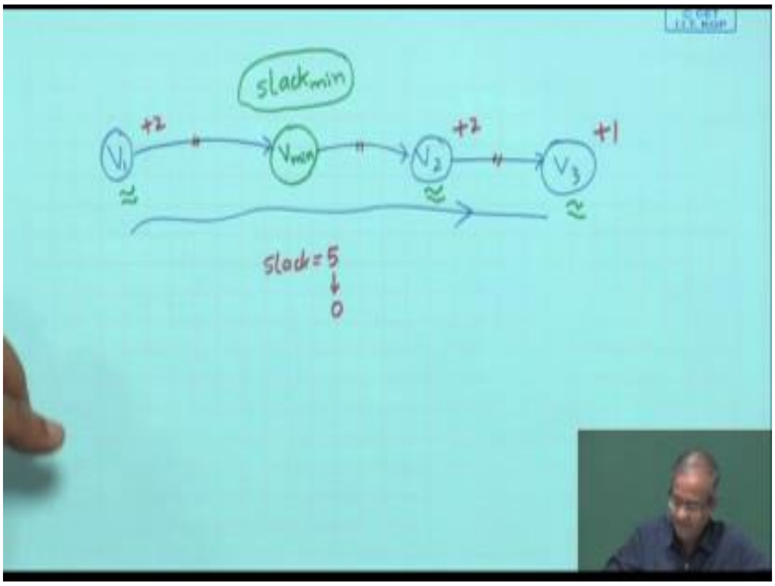
By repeating the process, the slack of each node in V will end up at *zero*.

The resulting timing budgets at all nodes are the final output of ZSA.

Let see, first the basic steps. See you have to clearly understand the steps because otherwise it will be difficult for to understand the steps. You see first step is the conventional static timing analysis, given the netlist you calculate AAT, RAT and the slack values.

So, you determine the slack values of all the nodes. You select a node let us call it v mean which has the smallest positive slack, slack mean well here I am assuming there is slack values are all positive because if a slack value is negative I am not touching that. Because negative slag values can be handled post this timing analysis by using some geometric modification to the layout well. So, once we have found out vertex v min, with the minimum positive slack then you find out path.

(Refer Slide Time: 14:23)



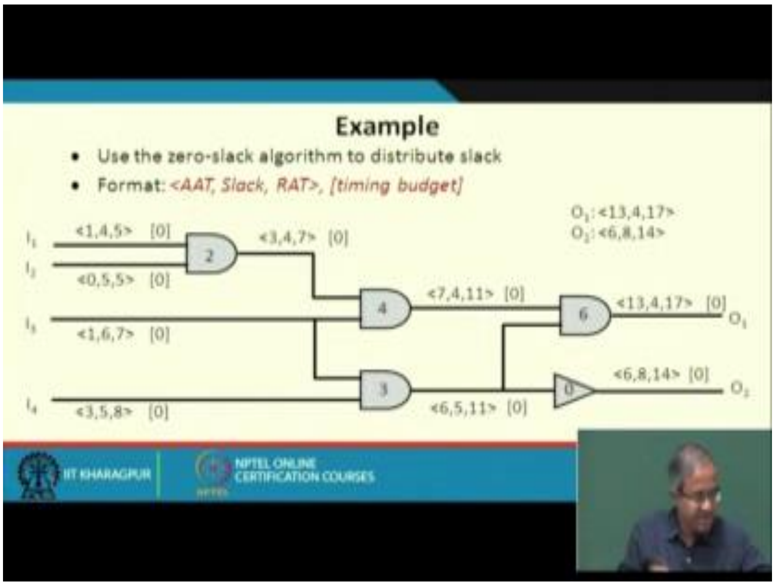
For example, you have a netlist you have found out a vertex called V min. Then your next step will be to find out a path through some other vertices let say V 1 let say V 2 let say V 3 that is a this is the path, this is the whole is path, such a way that this particular node is having the minimum slack. So, I all it as slack mean, what I am saying is that this path is dominating the value of slack mean. What is the meaning of this dominating means if we make any changes in the delays here either in the delay of the nets or the vertices then slack mean will also change this means dominating?

So if I can find path in the circuit such that this property is true then you consider that is path together at a time. So, the second step says to find such a path; that means, any change in the delays in vertices along the path will also ca cause the slack mean to change. Now suppose in this case this step is important let us understand. Suppose you find that in this vertex your slack was let say slack value 6 plus 6. Let say slack value was let say 5. So, the third step says. So, you have to make this slack value 0 somehow. So, this slack value of 5 you try to distribute among these vertices along the path. So, there is v 1 v 2 and v 3. Let say you add plus 2 here you add plus 2 here you add plus 1 here. So, this slag value of 5 we distribute evenly across this slag value. So, that what was 5 here now this become 0, right.

This is the basic idea you are trying to divide or distribute the slack value across all vertices in the path. Your objective is to make this slack mean equal to 0, right. So, this is the third step. So, each budget increment whatever you do will decrement the slack value of a vertex. So, if you increase the delay of a vertex, let us for example, here V 1. So, you are increasing the delay budget by 2. What does that mean you are increasing the actual arrival time by 2? If we increase actual arrival time by 2 means you are also decreasing slack by 2, slack equal to RAT minus AAT. So, increase in the time budget of a node will also cause a decrease in the slack value. So, slack values will all go towards the downward side from a higher positive value to a lower values fine.

So, this is what is mentioned each budget increment will decrement the slack value of a vertex. So, you go on repeating the process after making this change, again you recalculate this I slack values and you again find out the minimum slack node, again find out a path again, distribute go on repeating this. At the end you will be finding that all this slack values have been reducing to 0. This is the basic objective of the 0 slack algorithm. So, let us just illustrate this with the help of an example. So, the process will become clear. So, at the end whatever you get that is the output of zsa.

(Refer Slide Time: 18:37)



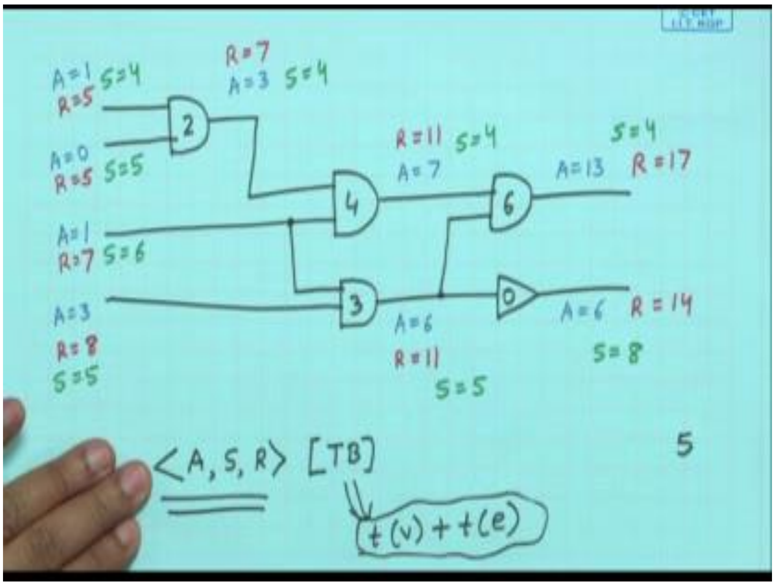
Example

- Use the zero-slack algorithm to distribute slack
- Format: *<AAT, Slack, RAT>, [timing budget]*

Let us take an example like this there are 5 gates. So, let us understand the notations first. There are 4 normal gates, well I am showing it has end gates where does not matter any kind of gates, and this is a bugger let say. And the values which I shown inside the gates they indicate the gate delays. So, by properly sizing the gates I can adjust the gate delays this already know right.

So, let us I just work this out, because it will be easier for you to understand. Let us I just work out this circuit here.

(Refer Slide Time: 19:23)



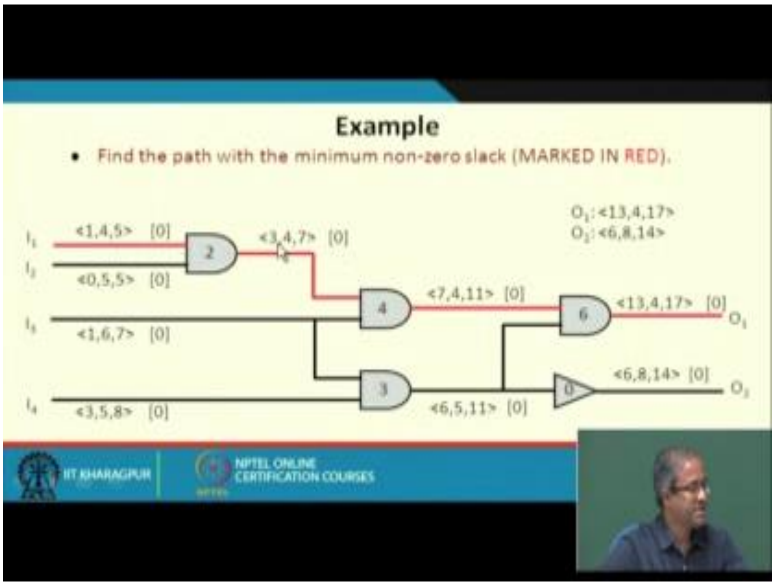
Suppose I have a gate like this. Output of this is going to another gate; this is a delay of 4. This is another gate delay of 3. This is another gate 6; this is the output buffer delay of 0. So, this is your circuit initial circuit. So, let us assume that your actual arrival time I am calculating AAT. Let us denoted as a say AAT of this what is 1. So, I am directly calculated, I am not showing that s, node S is implied. I am just showing it this A is 1, and this A is 3. Let say the inputs are arriving at these time steps and just initially we assume that all delay budgets are equal to 0 there is no delay. So, if you calculate the AAT value. So, these delays of the wires are assuming to be 0 initially.

So, the value of A here will be 1 plus 2. This A will be 3. Here it will be 3 plus 4, A equal to 7. Here it will be 1 plus 3 and 3 plus there whichever is larger 6; 6 and 7 whichever is larger plus 6 - 13, 7 plus 6, and 6 plus 0 6. These are the arrival times right. Similarly, if you calculate the required arrival time, suppose the required arrival time for this node was specified as 17. For this node it was specified as 14 let see. So, again you do a back stress for this. So, if you go back here it will be 17 minus 6 it will be 11. If you come here there are 2 paths 17 minus 6 coming here and 14 minus 0 coming here. So, it will be 11 that will be shorter minimum - R equal to 11. So, you come back here 11 minus 3 it will be 8. Similarly, 11 minus 4 or 11 minus 3 whichever is smaller 7 is smaller it will be 7 and here R equal to 11 it comes here.

So, it will be 11 minus 4 7, 7 minus 2 it will be 5. So, you have seen once you have calculated the actual arrival time and the required arrival time, you can also calculate this slack R 8 minus S e t. What will be the slack here, S will be 5 here S will be 6 7 minus 1 S will be 5, S will be 4, S will be 4, S will be 4, S will be 5, S will be 8 and here S will be 4. Now we are actually representing this 3 values in the form of a topple, where he shows the actual arrival time first, then this slack then the required arrival time, this is the notation we use. And within bracket we are showing the timing budget, that timing budget actually will indicate the delay of the gate plus delay of the output net. So, we are considering them together. Suppose I find that for some node t b has being calculated as 5. So, my gate delay plus the net delay total has to be 5, that way I will have to adjust it.

Now, this is the notational contention right. So, these with these initial values as have calculated. So, this diagram actually shows that exactly. So, whatever I have calculated these are the 5 values, AAT slack RAT 1 4 5 0 5 5. So, exactly what I have calculated. And the timing budget initially is assuming to be all zeros. So, this output O 1 and O 2 I am showing it separately. Now let us moves step by step. The first step says you identify the node with the minimum positive slack. So, you see here there the minimum positive slack is 4 it is 4 here, 4 here, 4 here, 4 here. So, you pick any of the nodes this entire path will be a dominating path.

(Refer Slide Time: 24:50)

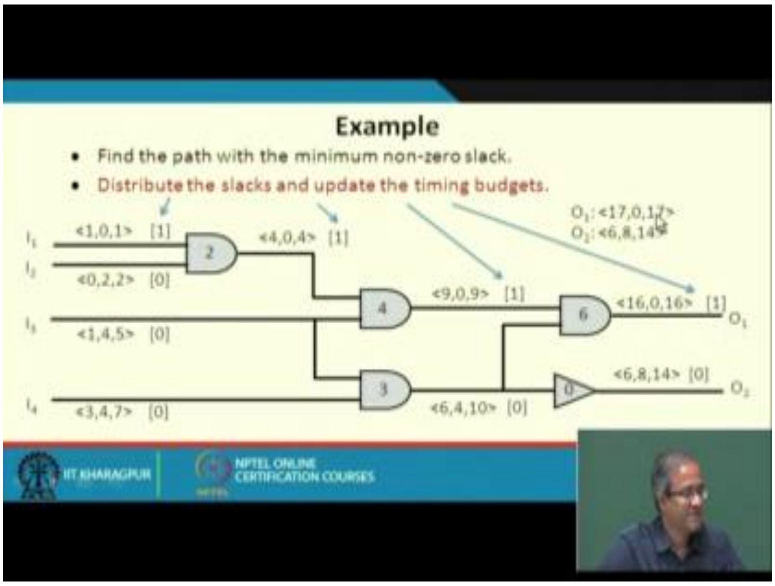


Example

- Find the path with the minimum non-zero slack (MARKED IN RED).

So, the path marked red that will be the minimum non 0 slack. So, now, you know what you do as I said let say you pick up this, you make this 0 4 and distribute this slack of 4 among the different paths.

(Refer Slide Time: 25:12)



Example

- Find the path with the minimum non-zero slack.
- Distribute the slacks and update the timing budgets.

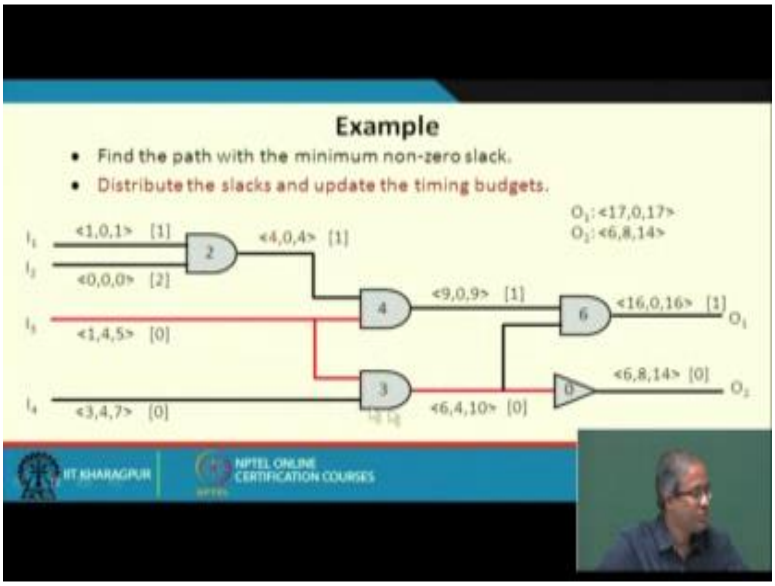
So, this 4 whatever was there, this was 4, this 4 you distributed as 1 1 1 and 1; that means, along the wires you distribute.

So, if you do it the delay the slack that was 4 that will get decremented by 4 because now the entire path delay has increased by 4 units. So, the slack will be reduced by 4 units.

This is the first step. So, you see earlier it was 1 4 5 if you decrease this slack to 0 this RAT will also become 1 because RAT minus AAT is slack right. This will be first step and this is 1. So, it O 1 it will be 16 plus 1 it will become 17, with this t I you repeat this same process.

Next step you find this 2 is this smallest positive slack and it does not know other node is dominating this only this 1. So, just reduce it to 0 increase this slack here, reduce it to 0 increase the time budget here, make this wire longer making this 0. So, this has become 0 repeat this process.

(Refer Slide Time: 26:26)



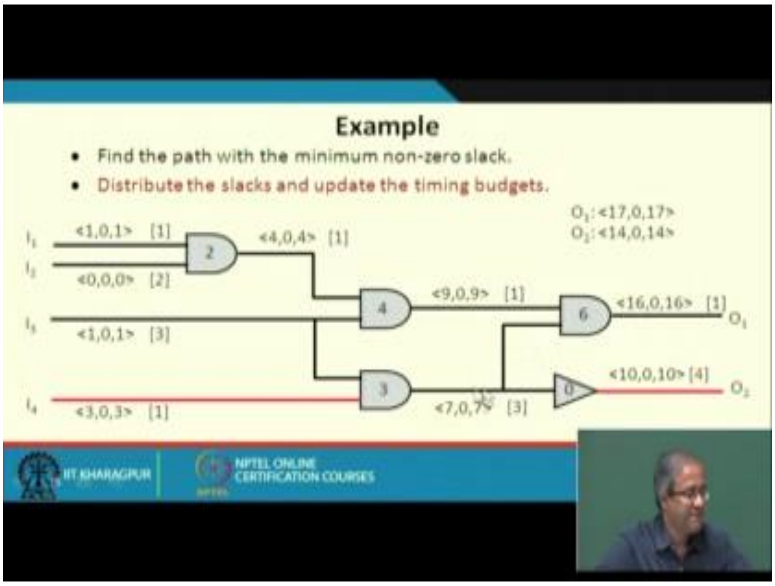
Example

- Find the path with the minimum non-zero slack.
- Distribute the slacks and update the timing budgets.

O_1: <17,0,17>
O_2: <6,8,14>

Now, you see 4 is the smallest. So, let us choose this path 4 and 4. So, this delay of 4 you can distribute similarly. Let say distribute us 2 and 2. So, it was 1 4 5 it becomes 1 2. 6 4 10 becomes 6 2. So, this 2 and these 2 the RAT also gets decreased by 2 each 5 and 10 becomes 3 and 8 right. So, same process you repeat. Now you have again you see it is not become 0 again this is 2 is the minimum. So, again choose the same path. So, again distribute this 2 across 1 1 this is become 3 this is become 3, and now this 2 will become 0 this 2 will also become 0. So, this has become 0 this will become 0.

(Refer Slide Time: 27:15)



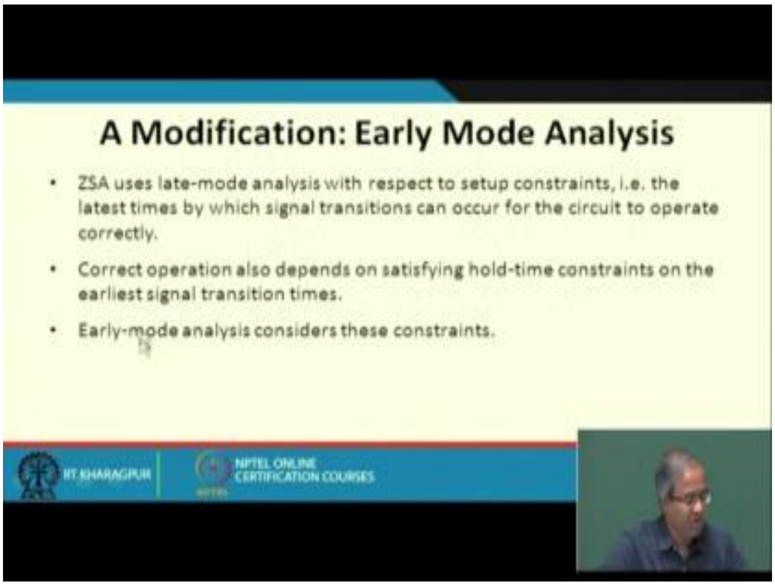
Example

- Find the path with the minimum non-zero slack.
- Distribute the slacks and update the timing budgets.

Now, the non 0 ones are this and this. Now you can choose this to into separate steps, but I am showing in one step. This is one path this is another path. So, to make this 0 you have to increase the budget here, to make this 0 you have to increase the budget here. Because it was 10, 4, 14 to make is 0 to make this t b as 4. So, at the end you see you have obtained time budget values, that will indicate what will be the delay of this different watts. So, that 0 slack can be achieved in all lines.

So, 1 4 1 3 1 1 2 3 one these are some guidelines for your next step of I means routing let say, where you can actually rout your wires such that or size the gates such that these delay values are actually achieved right.

(Refer Slide Time: 28:13)

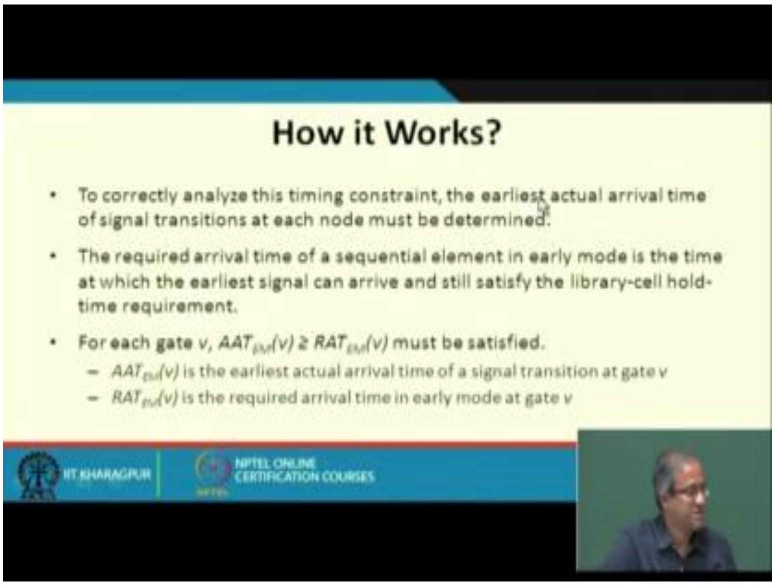


A Modification: Early Mode Analysis

- ZSA uses late-mode analysis with respect to setup constraints, i.e. the latest times by which signal transitions can occur for the circuit to operate correctly.
- Correct operation also depends on satisfying hold-time constraints on the earliest signal transition times.
- Early-mode analysis considers these constraints.

So, you can have one small modification to this ZSA algorithm. You see ZSA actually uses late mod analysis; that means, we are looking at the worst case delays; that means, we are looking at only the set up constrains. The latest time of signal transitions, but we were also interested to look at the whole time constrains. There is modification to these ZSA algorithm called early mode analysis where, we will look at the whole time constraint and make some small modification to the algorithm.

(Refer Slide Time: 28:45)

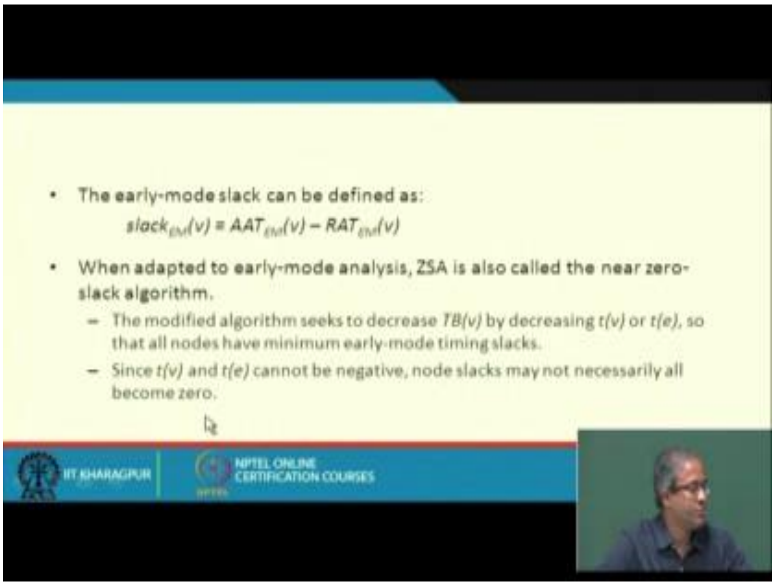


How it Works?

- To correctly analyze this timing constraint, the earliest actual arrival time of signal transitions at each node must be determined.
- The required arrival time of a sequential element in early mode is the time at which the earliest signal can arrive and still satisfy the library-cell hold-time requirement.
- For each gate *v*, $AAT_{EM}(v) \geq RAT_{EM}(v)$ must be satisfied.
  - $AAT_{EM}(v)$ is the earliest actual arrival time of a signal transition at gate *v*
  - $RAT_{EM}(v)$ is the required arrival time in early mode at gate *v*

So, the modification looks like this. Here the earliest actual arrival time must be determined not the latest. Normally when you look at the set up constrains we will look at the latest arrival times, what for hold constraints; we will look at the shortest paths the earliest arrival times. So, with respect to this we will have to satisfy that the actual arrival time in the early mode must be greater than equal to the required arrival time. Here it is grater then equal to right. So, we will have to satisfy this because for whole time constraints AAT can never be less than this.

(Refer Slide Time: 29:30)



- The early-mode slack can be defined as:

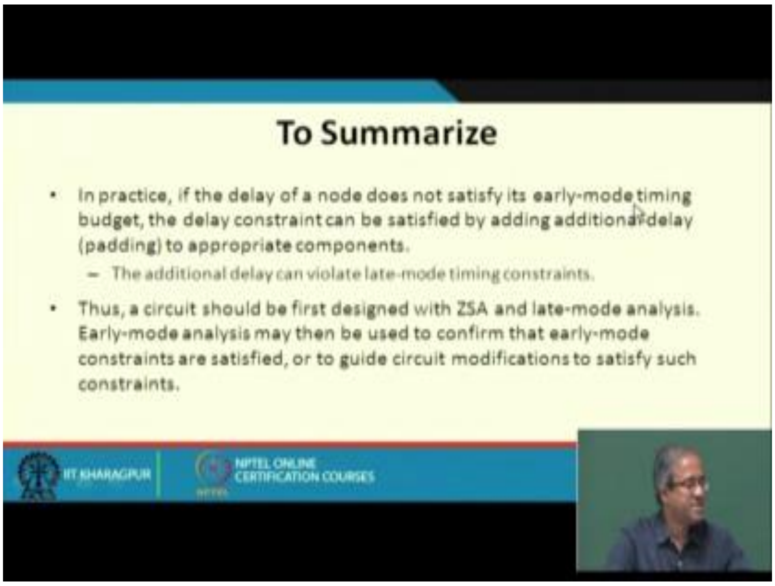
  $$slack_{EM}(v) = AAT_{EM}(v) - RAT_{EM}(v)$$

- When adapted to early-mode analysis, ZSA is also called the near zero-slack algorithm.
  - The modified algorithm seeks to decrease *TB(v)* by decreasing *t(v)* or *t(e)*, so that all nodes have minimum early-mode timing slacks.
  - Since *t(v)* and *t(e)* cannot be negative, node slacks may not necessarily all become zero.

So, slack you can calculate similarly. And you can adopt the ZSA algorithm to handle this, but that the thing is that here you may not be able to set all these slacks to exactly 0, because that might 4 some t v and t value to become negative in that case in early mode analysis.

So, this is one constraint here. That is why it is sometimes called near to 0 slack algorithm; not exactly 0 slack.

(Refer Slide Time: 30:04)



To Summarize

- In practice, if the delay of a node does not satisfy its early-mode timing budget, the delay constraint can be satisfied by adding additional delay (padding) to appropriate components.
  - The additional delay can violate late-mode timing constraints.
- Thus, a circuit should be first designed with ZSA and late-mode analysis. Early-mode analysis may then be used to confirm that early-mode constraints are satisfied, or to guide circuit modifications to satisfy such constraints.

So, to summarize, we can have early mode or latest mode both kind of 0 slack analysis. So, if it does not satisfy early mode budget, the delay constraint can be satisfied by adding some delays, but if you add some delay, then some late mode constraint might get violet it. So, what is normally done is that you first consider late mode analysis, and design the circuit that way. And subsequently you use early mode analysis just to confirm that the early mode constrains are satisfied because normally hold time constraints are not as serious as the set up time constraints. So, first you do the late mode analysis then you see which of the early mode constraints are getting violate it still, you try to tune this circuit netlist and trying to address them. So, I think with this we come to the end of this lecture.

Thank you.