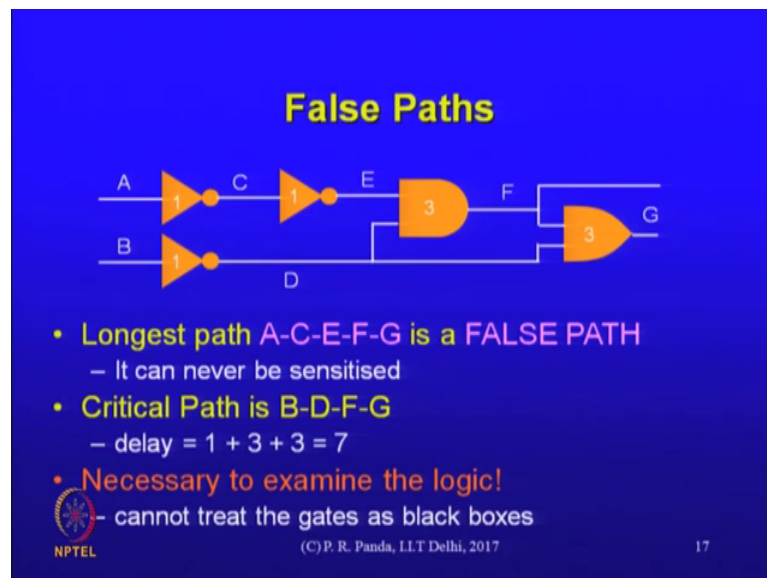


Synthesis of Digital Systems
Dr. Preeti Rajan Panda
Department of Computer Science & Engineering
Indian Institute of Technology, Delhi

Lecture – 26
Timing Analysis & Critical Paths

We will continue with the fall spark discussion.

(Refer Slide Time: 00:26)



As we have seen earlier the computation of the critical path is not merely structural it does depend on the logic and what else is going on in the rest of the logic. And not just what is happening on the critical path, the actual critical path may be different from what a structural traversal of the graph might indicate and in general it is necessary to examine the logic.

(Refer Slide Time: 00:58)

Propagation of an Event

- When does an event propagate along a path?
- An event propagates along a path P if $\partial f / \partial g \neq 1$ for all gates (input g, output f) along P



(C) P. R. Panda, IIT Delhi, 2017

20

So let us just try to arrive at the condition that determines the critical path. So, that one could make it a part of the analysis, the basic question then is related to the propagation of an event when does an event propagate along a path.

(Refer Slide Time: 01:25)

Boolean Difference

- $f = abc$
- $\partial f / \partial a = f(a=0) \oplus f(a=1)$
 $= f(a') \oplus f(a)$
- $\partial f / \partial a = 0$ means f is independent of a
- $\partial f / \partial a = 1$ means?

a	b	c	abc	a'bc	$\partial f / \partial a$
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	1	0	1



(C) P. R. Panda, IIT Delhi, 2008

19

For that one, definition is worth noting because the condition could be nicely captured in terms of this function called a Boolean difference. So, suppose you have a function f of three variables. So, this is an n gate with f as the output and a b c as the function inputs. So, that is the normal output a b c which is 1 when all the three are 1 otherwise it is 0. Let us define the Boolean difference operator as follows, Boolean difference of a

function with respect to 1 of its input variables is defined as, that function if you substitute a equal 0 x or that function when you substitute a equals 1.

It means f of a x or f if you just replace the a s by a prime and a function in this example here f of a is just this f of a prime is if you replace a by a prime, then that entry becomes 1, because the a s and the a primes get inverted. So, where you had a 1 earlier you have a 0 and where you had a 0 earlier you have 1.

So, that is the logic for a prime b c that is the column truth table column for a prime b c , x r of these two columns gives this column where these two are different and these two entries are different everywhere else it is 0. That is our function called a Boolean difference , Boolean difference of a function is defined with respect to its inputs with respect to one of its inputs, that is what the a is its one of the input variables of the function. What does this tell us, if this was 0 throughout what does it mean this is an x or function and if an x or function is 0 it means that.

Student: Both the entries are same.

Both the entries are same what is the meaning both the entries are same here what does it mean.

Student: That means, (Refer Time: 04:09) it does not matter if a is here (Refer Time: 04:09).

Does not matter you put a equal to 0 you put a equal to 1 if there is no difference then that function is independent of a what if it was 1.

Student: Then is different (Refer Time: 04:21) the a sum of because.

It is dependent on a what else can we say in general Boolean difference is a function these are just two extremes in general what is the Boolean when you do an x r as it is here.

What is the Boolean difference actually here it is neither 0 nor 1, this is a column that consists of both 0s and 1, so what is the difference.

Student: It is an all of the (Refer Time: 04:51) in a way.

Is just another function right, so what function is it of a b and c.

Student: It has become an all of a plus c plus a prime b c become basically it all become (Refer Time: 05:04).

It has become b c.

Student: (Refer Time: 05:08).

Right its b c just become b c we see because it is a prime b c plus a b c which is just bc

Student: b c.

Does that make sense what would it mean if b c is the Boolean difference what we have say.

Student: The eliminated a (Refer Time: 05:17).

A is eliminated yeah, not sure.

Student: Not that elimination of a gets as a different function it is no longer basically it is not redundant that it will has a.

A is certainly not redundant as far as the function is concerned. So, if its 0 it means that it is actually redundant the function is independent of a if it is 1. It means that what is it mean that function is actually f of a prime x or f of a if it is 1 it means that f of a prime is always different from f of 1 is the inversion of the other.

Student: It means the gate is only (Refer Time: 06:01).

That is d x the other extreme that is dependent only on a, but in general this is a function and that function the Boolean difference, if you ensure it to be 1 for example, you set the input variable such that that Boolean difference is 1, then the function any change that you make in a becomes visible at the output of the function.

Student: (Refer Time: 06:33).

So this is b c.

Student: (Refer Time: 06:39).

Right, so if $b \cdot c$ is equal to 1 here what is this function anyway this function is $a \cdot b \cdot c$ right, if you hold $b \cdot c$ as 1 it means that changes in a are reflected directly in the changes in.

Student: (Refer Time: 06:57).

The output of the function that is what the Boolean difference captures for us, it is an important metric as we say it will help in establishing sensitization of a path. It establishes that you hold the other inputs of the function to whatever values in a way that your path is actually sensitized the specific input that $v = 1$ that is the input with respect to which we take Boolean difference right. So, this tells us what the other inputs should be, so that our changes at the input at the a input are reflected in changes in the output yeah.

Student: The input patterns are providedly using this function.

We will get to how to apply, but as of now that is just the definition and an informal interpretation of what it might be a.

Student: (Refer Time: 07:53) in the;

This is with respect to a single variable, although we have introduced this in the context of timing analysis this has other interesting applications if you were to do a power optimization of a combinational circuit there to you are interested in if an input changes how far will it propagate, in the circuit because it may get mask at a various other stages by values that don't allow the input change to propagate to the output, there to there are some interesting in the analysis with respect to let us say power estimation or even when you do power optimization such a concept is useful in the theoretical formulation ok. So, but that is just remember that is the Boolean difference we will use it in 1 context here yeah.

Student: If any (Refer Time: 08:48) that appear as 1 that $p \cdot f$ by right a partial derivative function its different function it is bound to upon that there will be another (Refer Time: 09:00) will be 1 because it will always happen in the (Refer Time: 09:05).

Fine.

Student: Because the (Refer Time: 09:06)

It is different so yeah so.

Student: And is there way to (Refer Time: 09:12) that in code where have an expression and need not create the (Refer Time: 09:18).

This is the definition.

Student: But I still need get all these inverse all these.

You have a 1 function right f of a prime is 1 function Boolean function this is another f of a is another Boolean function, the x r of these two if we can take in an efficient way that is what gives us the resulting function right.

Student: (Refer Time: 09:40) this covering what.

So, we covered representation of Boolean functions now I need to manipulate that representation to perform ands and ors and x ors and so on bttts could be used to achieve.

The basic operations they are defined we did not cover them actually we left them saying you work it out. Remember after we talked about the representation I said that it may be an instructive exercise to just try and understand how we you would do and how you would do or you have two different functions of the same or overlapping set of variables how would you do. And so two functions means you have done bttts on the same data structure and they are pointing to different entries of that dash table.

So, the dash table is the same its just I have defined two different functions how would you do and how would you do or it is non-trivial it is not obvious try to do that yourself. The book does have the formal way of doing it, but x or any such operation this is a standard way to do this getting back to this question of when does an event propagate along a path an event propagates along a path b if the Boolean difference for all gates along the path this is that is one of the gates right. So, d f by d z means for every gate along the path the Boolean difference of the output with respect to the input.

Student: In the;

That is there in the path.

Student: (Refer Time: 11:24).

If that is true it means that events on the input will propagate to the output. So, that is what we looked at so it defines the conditions for the rest of the inputs of the gate. So, those conditions are true then we have established the propagation of an event yeah.

Student: (Refer Time: 11:49) basically now we have a new function that function we have a one that is the (Refer Time: 11:57) for which we have sensitizes path yes, but some there again will make to build (Refer Time: 12:03) btt for.

Oh, yeah, yeah, yeah, yeah, we are not saying this is trivial and, in fact the actual determination of the condition is still there we have only defined what condition needs to hold for us to propagate the path, but we will get to.

(Refer Slide Time: 12:20)

Static Sensitisation of a Path

- A path P is **statically sensitisable** if there is an assignment of primary inputs such that $\partial f / \partial g = 1$ for all gates (input g , output f) along P

(C) P. R. Panda, IIT Delhi, 2017

21

So, couple of things static sensitization of a path is defined as follows we say a path b is statically sensitizable if there is an assignment of the primary inputs such that that Boolean difference is one for all the gates along the path. So, assignment of primary input means here some condition has to hold right for the other inputs on that gate some condition has to hold on all the other inputs these gates are not of that nature, but it is like this to establish that path you have to collect all of those conditions on the rest of the inputs of the gates; it is some function.

So, ultimately you express that in terms of the primary input side all the primary inputs. That condition; if you are able to obtain an assignment of primary input such that all of

those Boolean differences turn out to be true then you have established the sensitization, this is what we call a static sensitization this is actually independent of the delays of the gates themselves assuming that once everything is stable if this condition holds, then that part that we are interested in is sensitizable ok, that is not the only condition under which its sensitizable, but if this is true if this condition is true then it is sensitizable it.

(Refer Slide Time: 13:59)

Controlling and Non-controlling Values of a Gate

- For AND gate, 0 is a **CONTROLLING** input value
- For OR gate, 1 is a **CONTROLLING** input value

Controlling Value 0 at input forces output value

Controlling Value 1 at input forces output value

NPTEL (C) P. R. Panda, LLT Delhi, 2017 22

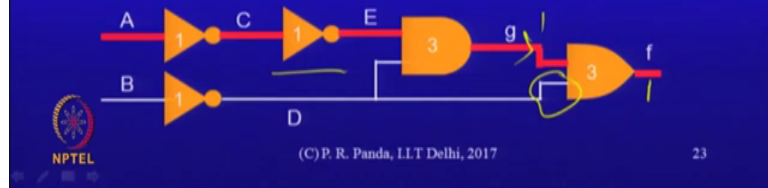
So, that is what is called as static sensitivity of path hm, let us just define one more term that says a controlled value and a controlling value for an and gate we say that a 0 is a controlling input value, what does it mean it is just that if that value is 0 on 1 of the inputs then irrespective of what is there on the other inputs you know the output, similarly for an or gate the 1 is a controlling value once you have 1 on 1 of the inputs then you don't care about the others the output is forced in both the cases.

So, that is what it is, so there is; this idea of a controlling input and correspondingly there is a controlled value at the output whatever that corresponding value is when you have the controlling input then the controlled value on the and gate is 0 controlled value on the output of the or gate that would be 1.

(Refer Slide Time: 14:57)

Static Co-sensitisation of a Path

- A path P is **statically co-sensitizable** if there is an assignment of primary inputs such that for all gates (input g, output f) along P, g has **controlling value** whenever f has **controlled value**



Static co sensitization is defined in the following way for a path p, it is statically co sensitizable. If there is an assignment of primary input such that for all the gates along the path g has a controlling value whenever f has a controlled value means, if this has 1 there is an output of an or gate if that is 1 then this has a 1 if it is this what about the inverter that there is only 1 input. So, if this question does not really arise both values are controlling.

So, that is what this is g has a controlling value whenever f has control value we are not saying anything about the other input that is how it is different from the static sensitization , but in reality of course, whether that path actually is sensitized or not does depend on what is happening on the other input. So, we have to formulate this a little further, but this is definition is it is statically co sensitizable if that condition holds if a control value is there on the output then the controlling value is there on the input of our interest that forms the path ok.

(Refer Slide Time: 16:32)

Path Sensitisation Condition

- **Static Sensitisation**
 - Side inputs on a path should have **non-controlling** values
- **Static Co-sensitisation**
 - path input is
 - earliest controlling
 - latest non-controlling
 - Necessary condition for critical path

(C) P. R. Panda, IIT Delhi, 2017

Then let us outline the path sensitization condition if it is static sensitization then everything is fine, we insist that all the other inputs on that gate should have non-controlling values. So, that our path is sensitized, so that is 1 possibility if this condition is true then it is fine then it means that we have identified a sensitized able path , but the co sensitization I need to go a little further as follows that is the path of our interest right .

And I say that it is for that other input to also have controlling inputs, but under the following conditions, if both of these have controlling inputs right both have zeros then we want that along our part the controlling input should appear earlier. Then it does not matter that the other input also gets a controlling value the change in the output is because of our input the is a logic clear.

Student: Sir, we only talk about a single change while sensitization.

What do you mean a single change all edges here result in edges there on the output.

Student: (Refer Time: 18:09).

Right, so both the edges when we say path it means that any change on the first input ultimately gets reflected on the output no matter how long that path is. So, this is a relaxation instead of strictly insisting that all the other inputs must have non-controlling values it could be that we allow a controlling value on the others also, but the timing situation is such that this signal arrives first if it is a controlling value then along our path

of interest the controlling value must appear first, then two we say that it is sensitized able.

So, this is part of a co sensitization correspondingly its for us to have both inputs being non controlling right both of them being one, but if that is the case then we want our input to arrive later, they say both of these are 1 you see that the output was 0 even here it does not matter that the other inputs became 1 earlier , but this is the 1 that is going to make a difference if, if this is going to make a difference there.

Then we want that even if the other inputs have non-controlling values our input is not controlling even if the others have non-controlling. If ours is the last 1 then you would still see that path being sensitized, this is an outline of what is a necessary condition for a critical path that critical path of course, is 1 that is sensitize remember we define the critical path in two terms 1 is it is the longest delay structurally, but also it should be sensitized able right if it is not sensitizable its a false part and that isn't the critical part .

So, these conditions can be the additional conditions that could be imposed, this 1 if it is true its fine, but otherwise combination of these two requirements can form the necessary condition for identifying a critical path.

Student: Cosensitization.

Cosensitization just refers to that there is a relationship with the other inputs also it is not.

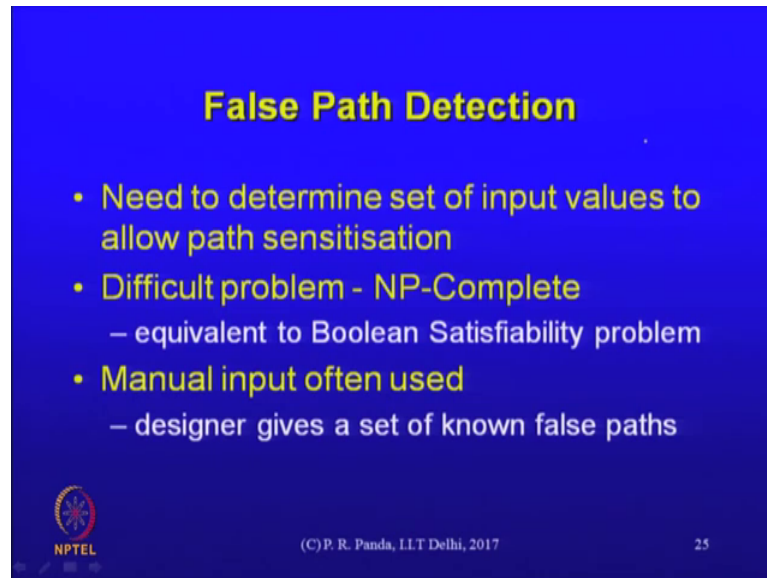
Student: (Refer Time: 20:42) an example which where it is used in some process in the (Refer Time: 20:50).

This is a tighter determination of the critical path right.

Student: Right, definitely.

So, what do you mean is it used this is the timing analysis as part of a timing analysis function. So, I could do just this just the static sensitization, but it turns out that is too restrictive this would be a way to relax that and actually have a more technically correct interpretation.

(Refer Slide Time: 21:22)



False Path Detection

- Need to determine set of input values to allow path sensitisation
- Difficult problem - NP-Complete
 - equivalent to Boolean Satisfiability problem
- Manual input often used
 - designer gives a set of known false paths

NPTEL (C) P. R. Panda, IIT Delhi, 2017 25

Then the false path detection involves what things we need to determine the set of input values that will allow a path sensitization, which means that we have picked 1 path and now we require a bunch of other functions to be true right. All those Boolean differences need to be true which means that essentially it is like you have the product of a bunch of expressions those need to be true how do you ensure those are true you have to solve the satisfiability problem that ultimately assigns values to primary inputs such that these functions are all true, right.

So, this of course, is the classical Boolean satisfiability problem often manual inputs are used the designer may have some intuition himself about which are the parts based on external knowledge not necessarily obvious from the netlist itself, but other knowledge might be there that the designer can actually used to instruct the tool that these parts are false.

(Refer Slide Time: 22:32)

Sequential Circuit Timing

- Earlier analysis valid for combinational logic part
- Flip-flop timing has to be accounted for

(C) P. R. Panda, IIT Delhi, 2017
26

So, that is about the false part lets quickly go through what is essentially just definition and putting together once you have computed the critical paths through the combinational logic what else remains to be done. So, that analysis was valid for combinational logic. So, far, but of course, real designs are sequential and there are flip flops along the way. So, how do you account for them?

(Refer Slide Time: 23:04)

Flip-flop Timing

- D-Flip-flop operation
 - On rising edge of clock, D propagates to Q after a delay $T_{clk \rightarrow q}$
- Timing constraints
 - D must stabilise T_{setup} before clock edge
 - D must remain stable for T_{hold} after clock edge

(C) P. R. Panda, IIT Delhi, 2017
27

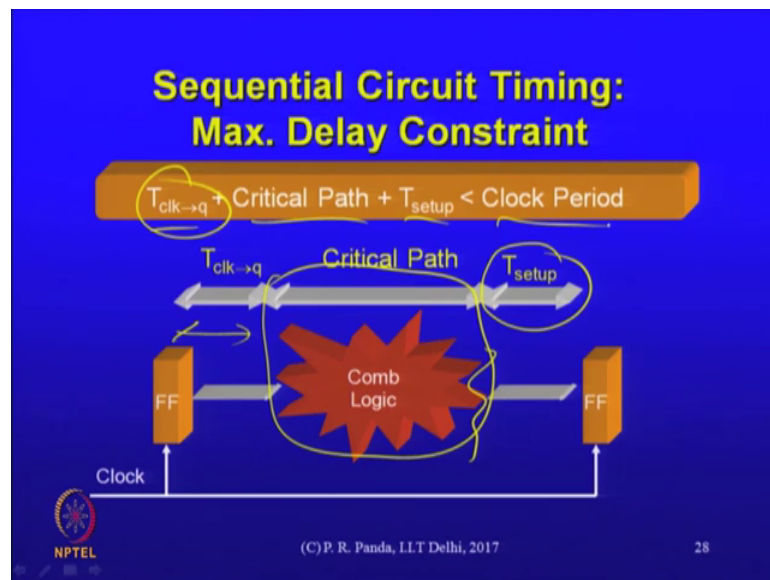
You need to at a very basic level of course, take into account some of the timing features of the flip flops themselves, what kind of timing features are there in a d flip flop things are defined with respect to the rising edge of clock right. What are the timing parameters

there is the requirement that before a clock edge that input must stabilize for a certain amount of time that we call the set up time right it is it is this so before that.

So, if that is the clock edge then the d input must be ready by this time otherwise it will not be properly placed. So, that is 1 parameter we do need to take care of there is a hold time which essentially requires us to keep that d input stable past the rising edge of the clock. So, it has to be stable for that much time right sometime setup time before the clock edge and whole time beyond the clock edge. One other important timing parameter that is relevant for us is there is a delay from the time the clock rises to the time the output the q output stabilizes the called a clock to q delay.

So, at the very least these three timing parameters I must take care of there is a further dependence on the actual value, value 0 value is 1 the timings may be different, but at least these are these you can take the max out of the different values if there is a sensitivity with respect to the value, let us say clock to q may be different for q equals 0, then for q equal to 1 if it is then you just take the max, so at least these three I need to in a simple way capture in my formulation.

(Refer Slide Time: 25:13)



Then what is the max delay constraint if there is a clock period, then we required that whatever is the critical path of that combinational logic I must account for that much delay for the clock to q because the inputs to that communist logic may be coming from flip flop outputs. So, that output is ready only after that much time right, so that is 1 thing

other thing is whatever outputs we compute here maybe d inputs of a flip flop and therefore, I also need to complete before the set up time for it.

So, my requirement could be that in addition to whatever is the critical path that I have computed I have their propagation delay and that set up time together all of this must be less than them (Refer Time: 26:08) of course, this is still at a logical level remember there are routing delays and other delays that complicate this further, but we can proceed in this direction if there are other delays that we are able to analyze and incorporate then that equation should have those calls what about the hold.

Student: Hold should be less than clock.

Hold should be less than clock period that is fine.

Student: (Refer Time: 26:41).

Is there a relationship between the critical path just like we had 1 equation here can we come up with a different equation involving the whole; oh there is no need.

Student: (Refer Time: 27:02).

There is so then.

Student: U should be less than (Refer Time: 27:05) logic and the clock (Refer Time: 27:09) minimum delay.

The minimum delay of the logic this critical path is essential max delay, but this propagation delay plus the minimum delay through this must be.

Student: Greater than hold.

(Refer Slide Time: 27:30)

Sequential Circuit Timing: Min. Delay Constraint

$T_{clk \rightarrow q} + \text{Shortest Path Delay} > T_{hold}$

Clock

FF Comb Logic FF

NPTEL (C) P. R. Panda, IIT Delhi, 2017 29

Greater than hold so such an equation also can be there, normally, you may not have an issue, but if you do then you may have to put some buffers or something like that to make sure that the whole time whole time of which flip flop are we talking about here this 1 because right that value has to stabilize (Refer Time: 27:51) yeah fine, so such computations.

(Refer Slide Time: 27:52)

Delay Modeling: Extensions

- Variable gate delays
 - depends on output load
 - depends on input slew
- Unequal rising and falling transitions
 - depends on transistor types
- Significant wire delays
 - wire capacitance needs to be modelled

NPTEL (C) P. R. Panda, IIT Delhi, 2017 30

Ought to be part of a the timing computation what other extensions we can think of a lot of simplifications we made along the way gate delays are variable in the general case of course, there is a dependence on the output load there is a dependence on the input slew the rate of which the input is changing both of these do contribute to the gate delays.

In ways that are non-trivial because they are not; obviously, known when you start off the netlist if it is known then at least partially the output load is known to us because we can compute the load based on fan out certain model can be used, this information is necessarily partial because the actual load is known only and your routing is complete without that you have to make some assumptions. So, that is one thing there is a variability the rise time and the fall time might be unequal these transitions might be unequal for a variety of reasons it depends and the transistor types it depends on.

So, many other things the timing characterization of the gates would usually have this information separately characterized to simplify an analysis you may just take max or min or whatever makes sense, but these delays are of course, separately characterized and as we have indicated in several places there why delays might be significant and therefore, the load due to the nets would need to be estimated in some way that completes the discussion I had in mind for the timing analysis of course, it is a very short discussion a lot of other things that we didn't go into, but this is sort of a very elementary logic level discussion on timing analysis problems, that completes the topics I had in mind for the synthesis course.

We did manage to go through most of the topics that I wanted to cover there was 1 that was left as optional unfortunately if I had a full class I would have paid some attention to it those involved a couple of ideas from the physical synthesis that is also synthesis and we didn't as well this is supposed to be synthesis of digital systems the physical synthesis would involve things like place and route right those of course, are topics that are complex enough in themselves that they merit a separate course really, but nevertheless some of the basic ideas it is not hard to cover, but at a very elementary level standard cell routing what is involved what is the objective function and what kind of optimization is involved.

It turns out that particularly standard cell routing the problem and its solution has some very nice parallel analogy to some of the synthesis problems that we have already seen there is also a global placement and global routing and. So, on the elementary ideas involved there can certainly be discussed in a relatively small amount of time, but since this is sort of a logical break I decided to skip those topics.