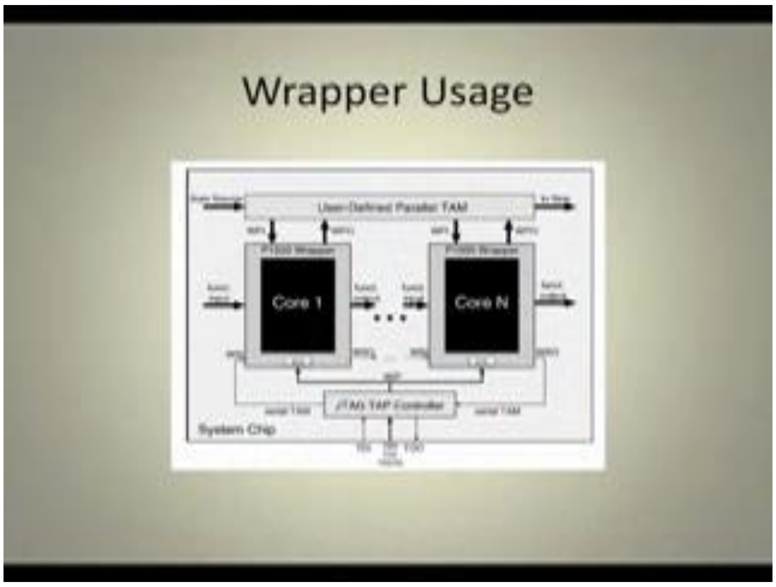
Digital VLSI Testing

Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 47
System/Network – On – Chip Test (Contd.)

(Refer Slide Time: 00:21)



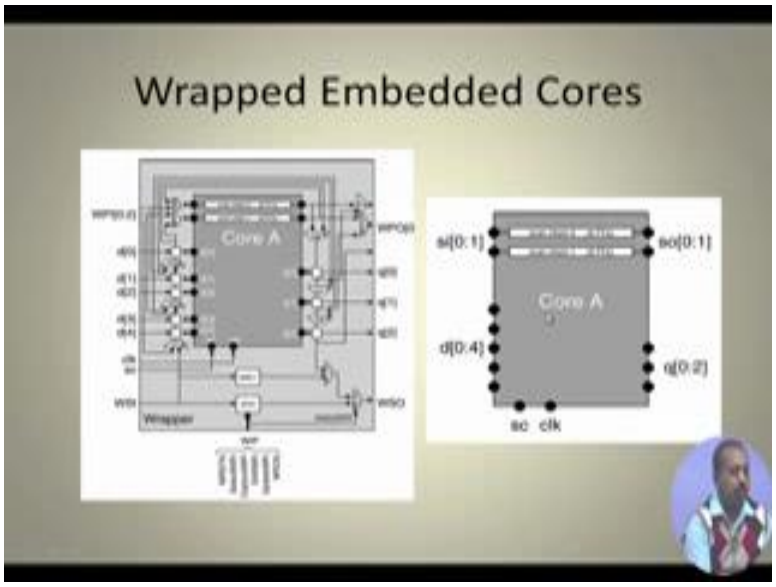
Wrapper Usage

So, how can we use the wrapper like you see that, here this core 1 core 2, so they are connected they are there in the system. So, here it is put as P 1500 wrapper actually, this P should not be there now because when these are originally proposed to IEEE, so that was when it is proposed this P prefix remains, so once it is accepted then this p goes. Now, this 1500 is an accepted standard, so this P should not be there, but anyway in this slide it has come in, so we just keep in mind that t is no more required, but for new proposals this P is required.

So, what we have is this user defined parallel TAM may be designed from source to sink; and through this WPI and WPO lines, so they will be fed to this core, individual core wrappers and that will finally go to the core. So, individual cores they get input from this functional inputs, they get input from this WSI line and also parallely from this WPI lines. So, based on that whatever input points that this core have so they will be getting the inputs and their responses can be given to functional output, or it can be for normal operation it will go to functional output. For test operation, it will either go to WSO or it will go to the WPO this parallel output, so that way this WPI, WPO lines, so they may be configuring the whole thing.

So, I can have a number of cores and there are WIR - wrapper instruction register, and this wrapper instruction wrapper port, so that will be controlling all these points test access control, this JTAG tap control that is this is at the system level, so you have got this JTAG tap control which is 1149.1. It has got TDI, TDO and this test mode select test clock and TD set, so they are coming to this JTAG, so this makes the serial tam, so this is a one bit serial TAM that is made here and the response is coming back to this JTAG controller. So, this is a serial TAM one bit TAM and here I have got the parallel TAM. So, you can take help of both the tams for transporting these test patterns to the cores and getting the responses from the cores so, but in most of the cases we will see that it will be using this parallel TAM for getting for transporting the test pattern and responses rather than this. But this will be used more for configuration purpose, this serial part WSI, WSO is serial part serial TAM, so this will most be mostly be used for configuring the WIR register or this bypass register things like that.

(Refer Slide Time: 03:17)



So, we have got this is a typical example of say ram wrapped embedded core. So, this is a core, so suppose it has got it is a core A we call it, so it has got a number of functional inputs d 0 to 4, it has got three functional outputs q 0 to 2, and there are two scan chains scan chain 0 and scan chain 1. Scan chain 0 has 6 flip-flops, scan chain one has 8 flip-flops. And they are coming from this, so naturally there will be two scan input and two scan output, so s i 0 1 and s o 0 1, so these are the two scan in and scan outlines. And then there should be control like scan clock and the normal clock, so they should be there.

Now, you see that this one, so this core when we are putting it into a wrapper, so when we are putting it into a wrapper, so it may look something like this. So, you see here, so this d 0 to d 4, so they were the functional input, so they are going to this wrapper cells boundary cells, and then they may be connected to the d 0 line. This buffer cells, so it also gets input from many other places, so it may get, so there through this WSI line, so it is going to this buffer cell and it can say the first cell, so say d 4, so it can get input from this WSI and it can get input from this WPI. So, it may be parallely loaded it may be serially loaded. So, whatever mode it is selected, so based on that it will either be loaded serially from here or it will be loaded parallely from here. So, this particular boundary cell, so it has got a functional input and two test input one from serial interface one from parallel interface. So, this depending upon this instruction that we have here, so this multiplexer and this cell will be programmed, so that it the appropriate input comes to d 4.

Similarly, all these are put onto a chain, so we do not have, so similarly this d 3 it does not get any parallel input, so it has got one functional input and one serial input through this line. So, there are only two inputs, so there is no choice. Similarly, for d 2, so it is again put on to there is a choice here, it can get input from this scan cell or this boundary cell or it can get input from this parallel input or it can get input from this functional input. So, d 2 may is a combination of three points, three input points and that will go to d 2.

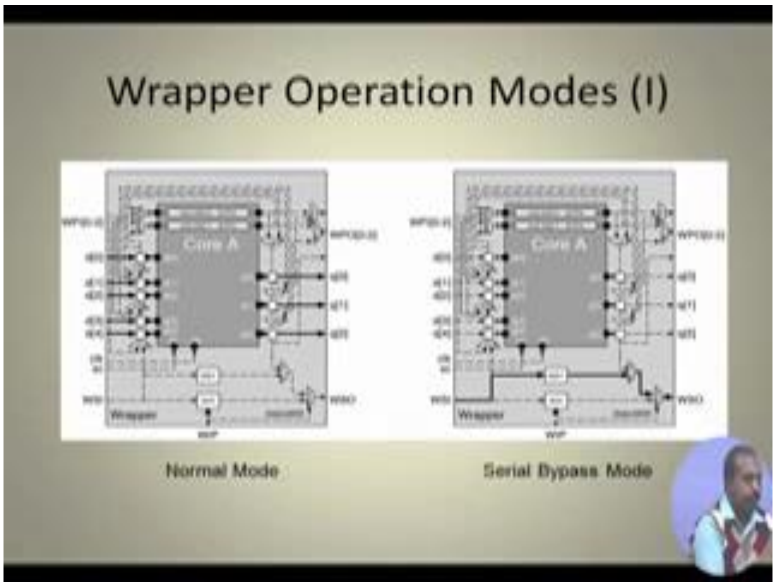
So, you see that this WPI lines, so WPI lines can be used to feed this data inputs, so since I have got say six input points, so all these WPI s cannot be used to feed them. So, even if you are doing a parallel loading, so the second cell - d 3 does not get a WPI, so if you are trying to load d 3 parallely, so still it will see a delay of two cycles, first cycle it will be loaded here second cycle it will be loaded here. So, based on the WPI lines, so I may have to make a choice I may have to make a choice like the cells onto which we are going to connect it.

So, here these two flip these two cells boundary cells, so they are put on say WPI 0, the next two are put on WPI 1 and the third-one is there, so it is put on WPI two. So, that way if we are trying to load these registers, this wrapper scan cells with this particular values from WPI, so at least, so I will require two cycles. So, in one cycle the value will come to the first cell second cycle it will go there. So, similarly parallely the first cycles of this cell this cell and this cell may be loaded; and in the second cycle, so this cell and this cell will give the value, so that way it requires two cycles for loading the pattern even if I am going by a parallel loading of patterns.

And serial loading, so everything is a chain, so I have to go by all the 5 cycles; then comes the point of this one this scan chain loading. So, scan chain loading you see that I can load the scan chain serially, so this is put onto a serial one or I can load it from this parallel point; so this WPI 1, so it may be loaded by this. And this scan cells, so it can be loaded by WPI 0, and it can also be loaded from this point. So, this is basically this scan cell can be output of this scan cell is connected to the input of this scan cell, so if we choose that particular mode that means, these two scan chains are combined into a single scan chain of 14 flip-flops, so that is one type of design. Of course, by programming this multiplexers, so we can keep this scan chain lengths to 6, 8 or 14 based on our choice.

Similarly, this q output that we have, so q output can directly go to the chip outputs like here this q 0, q 1, q 2 or it can be put through this multiplexer, so it can be put into this say this line, so it is going to through this multiplexer, so it is going to WPO line. So, there are 3 WPO lines, WPO 0, 1 and 2, so they are actually, so the last one we are taking directly, so this output is fed to this and the second output is fed here, so that way this WP, it can be put onto this TAM lines or it can be taken out via this WSO. So, this particular multiplexer, so it is getting input from this chain of say boundary cells or it is getting input from this bypass cell. So, based on that, so it will come here and finally this multiplexer, so it will select whether it is selecting the instruction register or it is selecting the data. So, the select WIR based on the value of this, so it will either pass this WIR value or it will pass the content of whatever is coming through this line bypass or this boundary scan register, so boundary scan cell. So, that way you see this wrapper design can be done, so by programming this multiplexers properly, so we can have different types of configurations now so that we will see slowly.

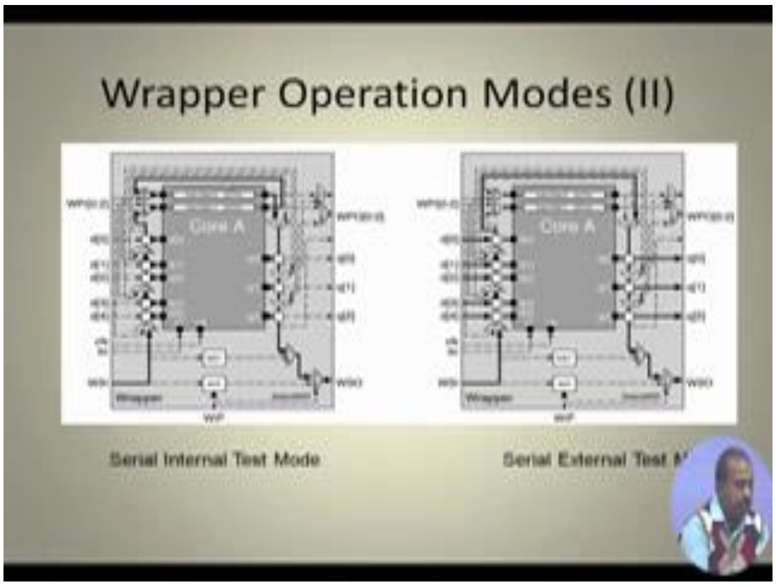
(Refer Slide Time: 10:20)



Wrapper Operation Modes (I)

So, in normal mode of operation, what is going to happen? So, this d 0 will come to d 0, d 1 will come to; so these are the functional inputs that we have seen and in the wrapper this functional inputs are coming and they are transferred to the functional input of the cores. Similarly, here these responses q 0, q 1, q 2; so they are transferred to the wrapper outputs q 0, q 1 and q 2, so that is the normal mode of operation. In the serial bypass mode, so rest of the thing is not connected, because this WIR will select this bypass, so this is not there, so this bypass will get selected, so whatever WSI is coming, so it will be bypassed like this, it will go to WSI. So, this is the serial bypass mode of operation.

(Refer Slide Time: 11:10)



Wrapper Operation Modes (II)

Then serial internal test, so in serial internal test, so I have to apply the test pattern. So, what is happening, so you see that it is serial, so serial means I have to pass the test patterns serially through this scan chain. So, it is through this line, so it comes through this chain it reaches this scan chain then you see that pattern shifts through this scan chain and reaches the scan chain 0, it shifts through scan chain one and reaches scan chain 0. And then scan chain 0's result 0's output is actually connected on connected to this chain of this q 0, q 1, q 2. So, what is happening is that, so this is serial internal test, so the first the test pattern will be shifted like this then the response will be captured on to this scan chains and this flip-flops, this bound boundary scan cells and then they will be transferred through this WSO line to the output. So, this is the serial internal test mode that can be implemented.

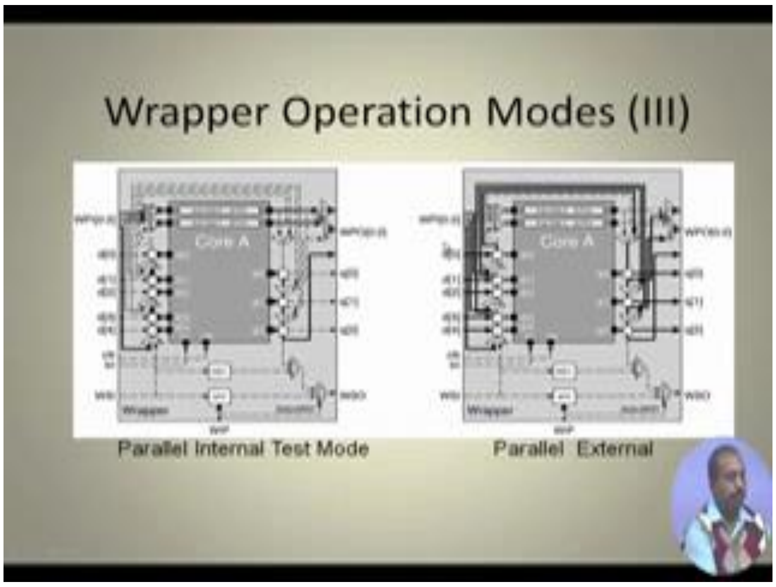
Then we have got this serial external test mode. In serial external test mode, so this d 0, d 1, d 2, so through this functional patterns will be applied, then this WSO; so this is serial input is applied through this. And through this WSO line, so we can load this pattern here and then this through this the pattern may be loaded through this WSO line, so it is loaded now you see this scans this chains, so this flip-flops that we have, so they are actually transferring the pattern coming from here or coming from this WSI. So, based on the pattern that we need to transfer, so it will be going there. So, now it is this scan chains are not affected, so it comes to this point.

You see, so this is external testing, so what is required is that it is for the interconnect testing. So, say this output from here, so it is going to be connected to another cores input, the interconnect between say core A is connected to core B. So, this part is actually the core B's point first through this shifting pattern through this WSI line, we will be shifting in the pattern that we want to apply here for doing the serial test for doing that interconnect testing.

So that way this pattern gets loaded onto these output cells, and then as soon as this has been loaded onto the values, so values will be available at core B's d 0, d 1, so these inputs, so they will be loaded here. And now that will be scanned out from the core B's chain actually, so then the values will be available at WSO of core B. So, we can see whether the values that we have put at q 0, q 1, q 2 of core A got loaded into this d 0, d 1, d 2 those lines of core B, so that is why this connection is shown to both the lines are both the connections are there.

But for the core A, if it is a core A to core B connection this WSI line will first load this q 0, q 1, q 2, these cells then in core B it will be checking it will be loading these values d 0, d 1, d 2 etcetera to the functional input values onto the corresponding scan cells and then it will be shifted out at core B. So, at core B, WSO we see whether it has been tested properly or not, so that is why it is called extest mode. Or if there is some user defined logic that we need to check, so the user defined logic can be given value through this q 0, q 1, q 2 and the responses may be collected from this UDL into this d 0, d 1, d 2 of the next core that can be done.

(Refer Slide Time: 15:18)



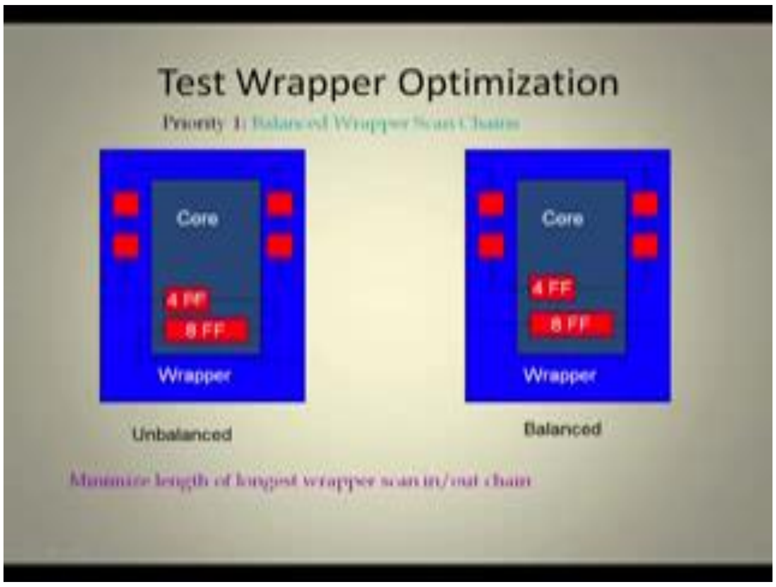
Wrapper Operation Modes (III)

Then we have got parallel internal test mode. So, parallel internal test mode means this values will be loaded parallely from this parallel interface. So, the serial interface does not do any work. So, this you see that there are three input lines, so the three WPI lines 0, 1 and 2, so this is loaded, so this first line will be feeding this one, so there it will be loaded here. Then it is this line itself will be used to parallely to load all these flip-flops. So, it will be loading this 5 flip-flops, then this line WPI 1 will be feeding this scan chain and WPI 2 will be feeding this scan chain. Because that is the best way to do the balancing because this has got 6 flip-flops, this has got 8 flip-flops and here we have got 5 flip-flops, so they can that is the most balanced distribution. So, of this 5, 6 and 8 length as if there are three scan chains, now one of 5, one of 6, another of 8 flip-flops, so they are fed by three different WPI lines.

Similarly, at the WPO side, so these primary outputs from the core, they are forming a chain, so that is one line. Now, this scan chain one, so that is that is forming another line and this scan chain 0 that is forming another line. So, WPO 0, 1 and 2, so they are actually these scan outputs are taken and they are forming the parallel outputs. So, in the parallel internal test mode, so we are applying the test pattern parallely to these primary inputs, and the scan inputs of core A, so core A evaluates the input. And the results are available in the scan chain as well as this q, q 0, q 1, q 2, so they are scanned out through this WPO lines, so that is the parallel internal test mode.

Similarly, the parallel external test mode, so here it the operation will be similar, but now this, so this d 0, d 1, d 2, so they will be loaded parallely. So, first I need to load the values in q 0, q 1, q 2, so that will be done by means of this q 0, q 1, q 2, so that will be that will be done by means of; what is it q 0, so this is basically this one. So, q 0 is loaded serially, so from d 0 it is loaded here; and from d 1, so the values that are coming they are loaded. In fact, this WPI is 0 that is actually forming this chain, so loading this patterns, so loading this patterns, so it is otherwise similar as that serial input, but excepting the thing is that now it is loaded parallely through the WPI interface, but it is taking the, it is testing the interconnect between core A and core B.

(Refer Slide Time: 18:23)



Test Wrapper Optimization

Priority 1: Balanced Wrapper Scan Chains

Minimize length of longest wrapper scan in/out chain

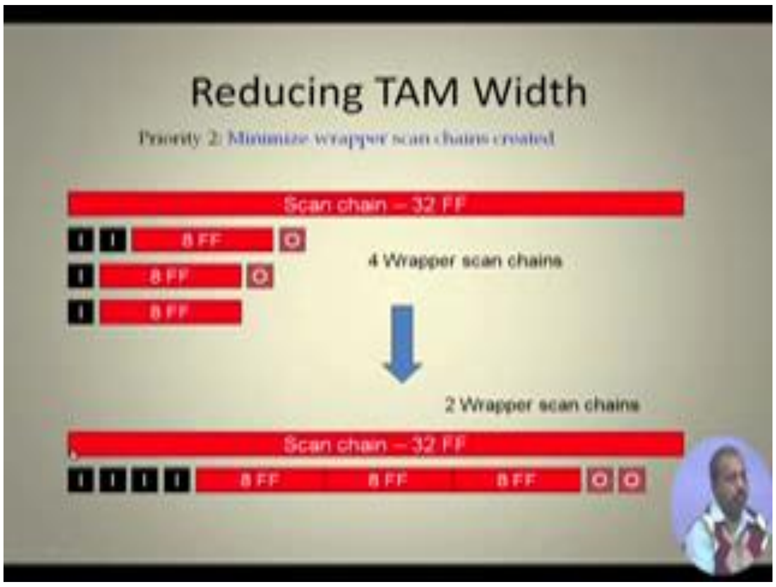
So, if we come to this test wrapper optimization part then this may be a typical situation like a core, it has got two scan chains in it, one consisting of 4 flip-flops another consisting of 8 flip-flops; and then there are two inputs and two outputs. So, one possible wrapper design may be that, so if I have got only two lines, so this number of TAM wires given to me is only two. Now, what I can do, I can put these flip-flops onto a one chain this primary input primary outputs onto one chain like this, and then this scan chains they may be combined and we can do a connection like this.

So, this way what happens is that I have got 4 flip-flops in one chain and 12 flip-flops in another chain, so that is a highly unbalanced design. Now, when we are doing testing, so for every pattern you see I have to load all these flip-flops. Now, number of cycles the shifting that will be needed for loading this pattern is any test pattern is now 12. So, if we do it in an unbalanced fashion, it is like this.

Other option, maybe I have got again I have got only two lines, but what I do I connect these two inputs then these flip-flops 4 scan flip-flops and these two outputs onto one chain, so I get one chain of length of 8 here, another chain only the scan chain, so that is also of length 8. So, here the maximum scan chain length is 8, for loading a test pattern, so it will be requiring a shifting of 8 cycles, so that way it is a balanced wrapper design, so number of shifting needed part test pattern will be 8 compared to the case that previously in this case, so it was 12.

So, now we can understand that this test application time will be dependent on this wrapper design. So, basically this distribution of this primary input, primary output and this scan cells into different chains, so that makes it that has got a definite impact on the test time. So, this is one way. So, the objective of this optimization procedure is to minimize the length of longest wrapper scan in out chain, so basically that is the objective. So, we want to minimize the longest wrapper scan chain, so that has to be done.

(Refer Slide Time: 21:02)



Reducing TAM Width

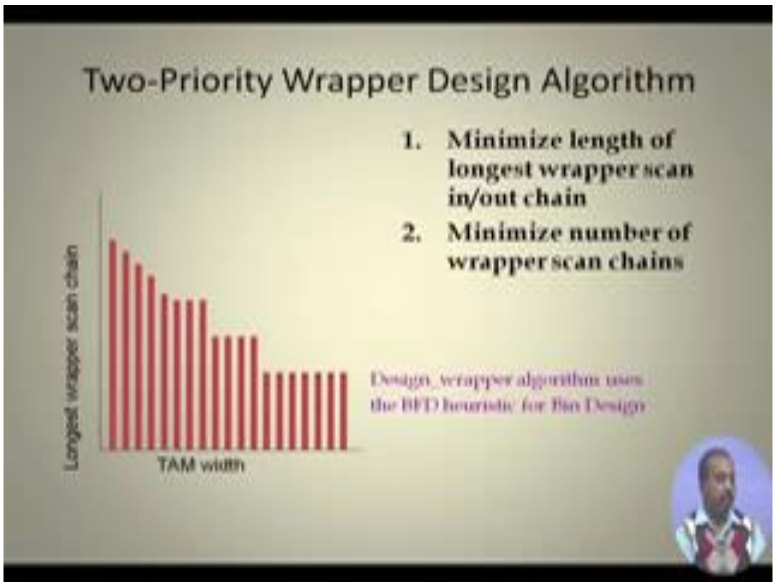
Priority 2: Minimize wrapper scan chains created

Another objective may be to minimize the wrapper scan chains that are created. So, you see that in one suppose I have got 4 chains, so in one scan chain there are 32 flip-flop, other scan chains they have got 8, 8, 8 flip-flop. So, what I do since this has got 32 flip-flops, so I cannot connect anything else to it, so what I do I, so there are more inputs and outputs there are 4 inputs and 2 outputs, so I just try to do the best one that I can do. So, I put two here, one here, and then two here, one here, one here, and this has got only one there. So, I just I do not connect between the scan chain I just attach the primary input and primary outputs to this wrapper.

So, now the TAM width is equal to 4, because I have to feed four such channels, so TAM width is equal to 4, , but for testing every pattern, so every test every test pattern, so I need to give 32 pulses, so 32 shift pulses, so 32 cycles will be needed to load the pattern. So, on the other hand, so I can do a bit more justice, so if I can do it like this, so instead of taking 4 lines, if the 4 lines are provided, but I do not use those 4 lines I use only 2 lines. In one line, I feed this scan chain with 32 flip-flop; on the other line I have this 8 flip-flops of 8 chains of 8 flip-flops we can concatenate them, so that gives me 24flip-flops. Then 4 flip-flops - 4 input flip-flops at the end 2 output flip-flops at the other end, so that makes it 24 plus 6 - 30 flip-flops, so the chains are more or less balanced now and the number of scan chains are also less.

So, there can be two objectives that we can have one part first part is minimize the length of maximum case. So, in both the solutions the first priority is satisfied, so the minimizing the length of this maximum chain, so both the cases the answer is 32. But still we see that the second answer is better than the first one, because it has reduced the number of TAM lines that are needed, so number of TAM wires to be dedicated for this particular core, so that will be less, so that way we can have this wrapper design part done taking care of these two priorities.

(Refer Slide Time: 23:35)



Two-Priority Wrapper Design Algorithm

1. **Minimize length of longest wrapper scan in/out chain**
2. **Minimize number of wrapper scan chains**

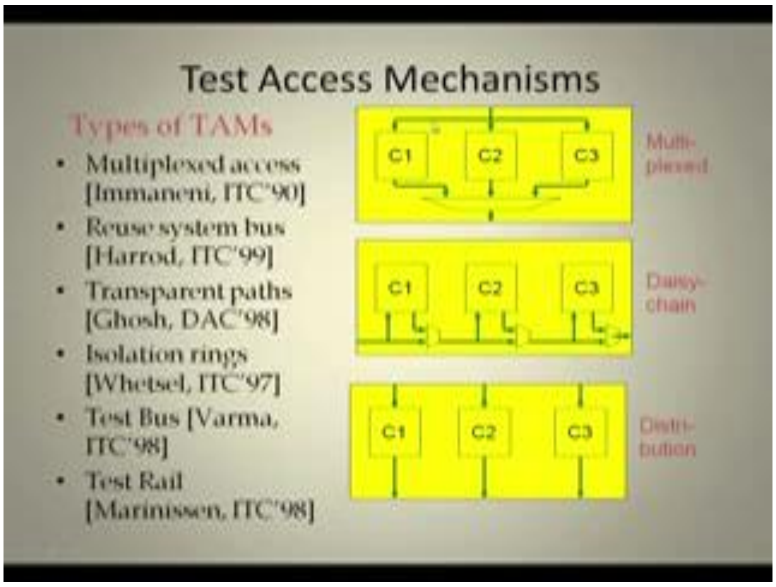
Design_wrapper algorithm uses the BFD heuristic for Bin Design

And in fact, so if we just plot this wrapper TAM width and this longest wrapper scan chain, so in general it is expected that if I allow more TAM width then this length of wrapper chain the longest wrapper scan chain length will reduce, but in some cases it may not happen. And in fact, this is the situation, so this is for one TAM width one only one bit is provided, so everything all the flip-flops, all the scan chains, all the input output points, so they are put on a single chain, so this is the length of the longest chain.

Then there are two number of TAM lines provided is 2. So, in that case, we see it is it is a slightly 2, 3, 4, 5 also it reduces, but 6 up to 6 it reduces, but for 7 and 8 there is no reduction. So, it may be the case that at that point, so even if I go for this 7 and 8, so there is no this maximum chain length cannot be modified, because the wrappers this scan chains may be such that it for this values it does not reduce. So, again after sometime it reduces, so 7, 8, 9 from 10 again it reduces, so that way it gives some sort of staircase behavior, so you get this type of staircase behavior.

So, this particular observation has been taken into this design wrapper algorithm, so this design wrapper algorithm, what it does is that it takes one scan chain and tries to attach more scan chains and these flip-flops to it, so that this maximum length is reduced maximum length of all scan chains is reduced. And it also reduces the number of scan chains. So, it uses some sort of best fit decreasing heuristic for the design of bin, and that way it works, so this is the wrapper design algorithm that is followed.

(Refer Slide Time: 25:49)



Test Access Mechanisms

Types of TAMs

- Multiplexed access [Immaneni, ITC'90]
- Reuse system bus [Harrod, ITC'99]
- Transparent paths [Ghosh, DAC'98]
- Isolation rings [Whetsel, ITC'97]
- Test Bus [Varma, ITC'98]
- Test Rail [Marinissen, ITC'98]

C1
C2
C3
Multi-plexed
Daisy-chain
Distri-bution

So, once this wrapper design has been done, so we can go for a different types of TAMs. So, there can be different types of test access mechanism like multiplexed access, so the TAM that this is the TAM width at different time instants, so this TAM is given to different cores, so C 1, C 2, C 3, so they are given tams at different tams and they are combined in this fashion. Or it may be that it is daisy-chained, so this first TAM is given to core 1. So, if core 1 does not use it then it will be given to core 2, so somehow this control signal has to be generated by this core 1. So, if it is not using the TAM then it will be passing this TAM to the core 2. And then this it will be given to core 2 some lines will be going there and if it not using the TAM then it will tell that it is not using it, so this TAM will be given to the core 2. So, this way we can have different types of TAM access mechanism.

So, once we have done this wrapper design part then we know that for every core what is the good wrapper design what is a what is a good distribution of this wrapper scan cells. And once we have that then we can go for generating this, we can go into this test access mechanism design. And knowing that this will if I give it this much of time width it will require this much of test time. So, we can knowing the volume of testing for individual cores, so we can figure out like which core test time we want to minimize maximally. So, if some port is taking lot of type, so we have to do something, so that its test time is less and for that purpose we need to have these scan chains this wrapper design information available to us. And we may like to provide more time width to that core, so that this test time is less for that core. So, this way this TAM this wrapper design can help in the TAM design process.