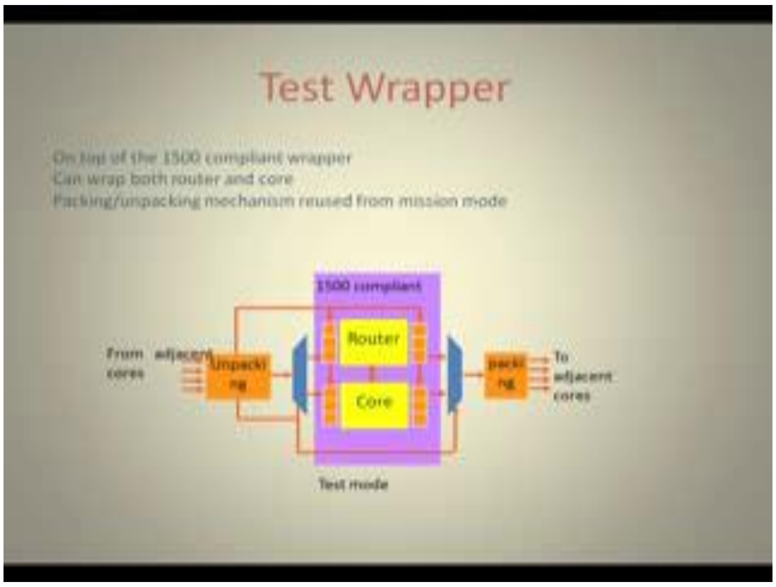
Digital VLSI Testing

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

Lecture - 54
System / Network – On - Chip Test (Contd.)

The basic design of this wrapper is similar to what we have previously; 1500 wrapper.

(Refer Slide Time: 00:21).



Test Wrapper

On top of the 1500 compliant wrapper
Can wrap both router and core
Packing/unpacking mechanism reused from mission mode

So, 1500 wrapper previously for the SOC testing, so we could wrap the core, so we have to consider this scan chain and input outputs of this core and put them on to this wrapper scan chain. So, we have to somehow design the wrapper so that we can it can be applied serial mode parallel mode and those all things. Now that problem gets extended further because for testing the router also I need to put it into a wrapper. So, since this core and router they became an integral part in any system, so they are taken together and this entire wrapper design is done taking both of them together.

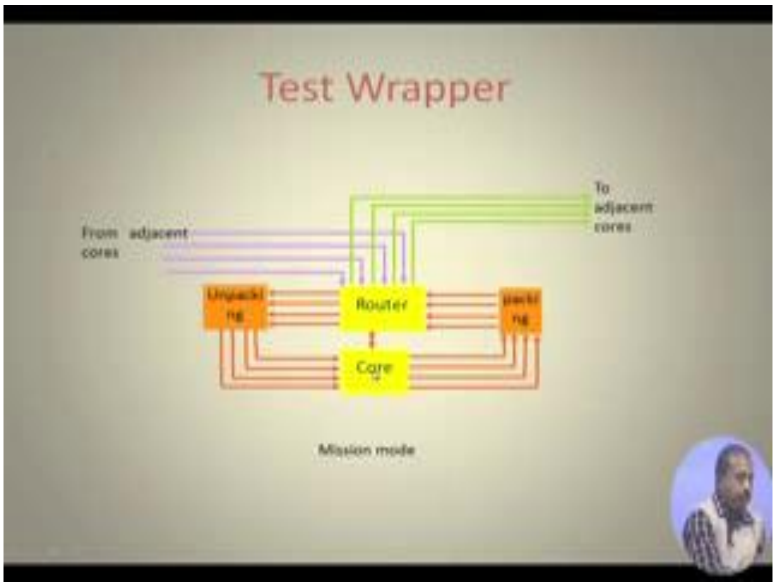
So, this core and router they can talk to each other there are signals connected through the local channel of the router the core is connected. Now, it may have a number of scan cells here the boundary scan cells the router will also have some boundary scan cell. So, they may be connected in a scan chain fashion like this. Now from adjacent cores this input will be coming so there that will get that unpackaging will be done. So, this is basically the job of this network interface module.

And after that this may be for the router or it may be for the core. So, if it is for the router then it will be coming to this part and it will be loaded into the router or if it is from the core then it will be loaded from this side to be loaded on to this or it may be done serially through this. So, there are many options that are created. So, on top of the 1500 complaint wrapper this test wrapper as to be designed. It should wrap both the router and core. Packing and unpacking mechanism is reused from the mission mode. So, mission mode means in the functional operation of the system there is the network interface module which does this packing and unpacking. So, here that part of the hardware will be reused in the test mode. So, it is modified to work in a different fashion.

So, these test patterns it will be selecting this unpackaging mode, so it will select this one, it will made the test pattern to go through this channel. Instead of the functional input being fade from this site, so test pattern will be fade from this site so that will be loaded and this responses will get collected and the responses will be transferred to this one and then it will be going to the next package.

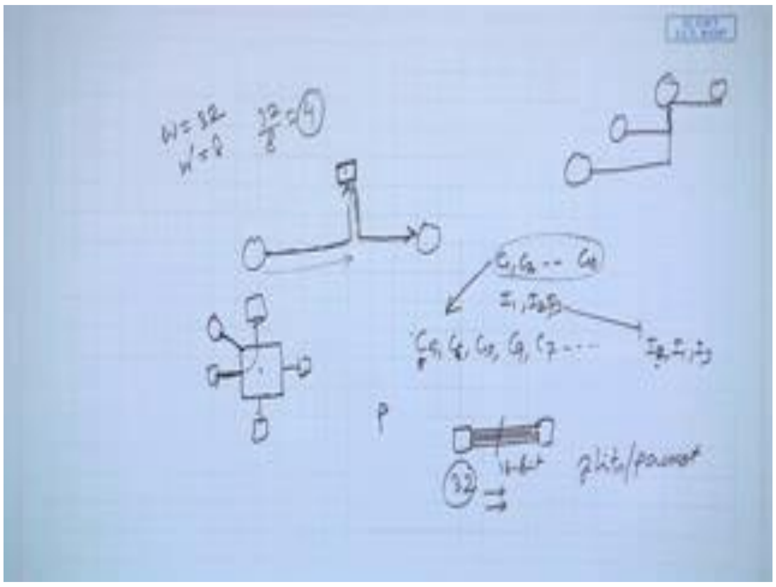
This part works in functional mode, because the test pattern and responses have been collected from this particular core. So, they have to be applied to the next they have to be applied to the next core just to transfer to the destination. So, rest of the thing they operate in the normal mode only this core and router they will operate in the test mode.

(Refer Slide Time: 03:32)



So, this shows the detail in mission mode of operation; this is this is router is from adjacent core so will get this input coming here and they are actually from the router they will go to the adjacent core. So, basically this router has got function like it may have some data that has come from the neighbor.

(Refer Slide Time: 03:58)



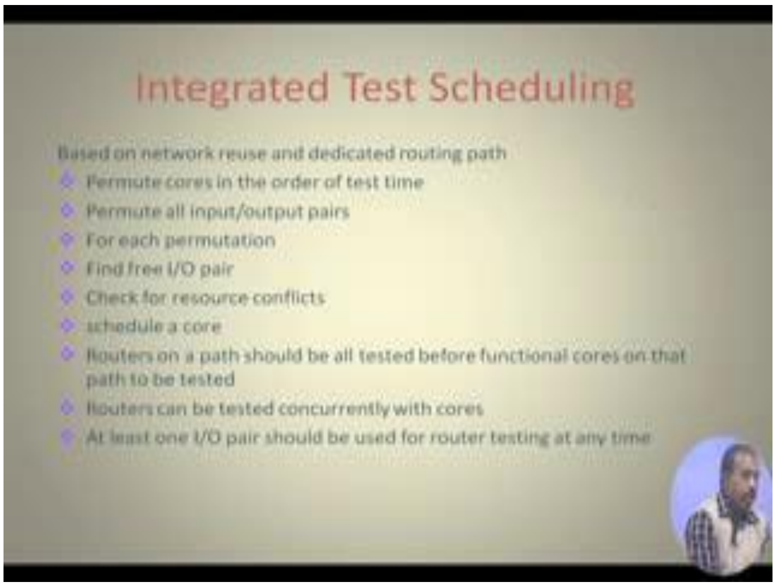
So, if you look into a mesh router. So, this is a mesh router then it has got a number of neighbours, this is top neighbor, this is left right and bottom; so neighbours are there. Now it has got a local port as well to which the core is attached. Now, when a packet comes to this router it may be from this neighbours or it may from the local port. So, whatever it is, it has to transfer it to the next step. So, like here; if it is coming from these are the four channels coming from adjacent cores then it as sent the data some adjacent core depending upon the address requirement. It may be the data that is coming from this left it has to be routed to this top; so that as to be done.

So, this way this router will be doing this routing operation, so it will be done there. Now this and for the core part; so for the core part it will be doing this unpacking and packing; so if there is some data destined to this core it will be going to this unpacking mode and then it will be coming to this core. And similarly if this core is willing to transfer some data, so data has to be packed first so it goes to the packing unit that network interface part. Actually this unpacking and packing these two modules are together forming the network interface part.

So, they are actually the network interface part of the system. So, between core and the router we have got this network interface module. So, from core whatever message has to be transferred so that is coming to this n i to a network interface packing module and then it will be doing the packing there and that will be given to the router for transmission to adjacent cores. And, if from some adjacent core it gets some packet to be delivered to the core so it will be going to this unpacking module of this network interface, it will do the unpacking and then the data will be delivered to the core.

So, this is the normal operation or mission mode operation of the wrapper.

(Refer Slide Time: 06:11)



Integrated Test Scheduling

Based on network reuse and dedicated routing path

- Permute cores in the order of test time
- Permute all input/output pairs
- For each permutation
- Find free I/O pair
- Check for resource conflicts
- schedule a core
- Routers on a path should be all tested before functional cores on that path to be tested
- Routers can be tested concurrently with cores
- At least one I/O pair should be used for router testing at any time

Now, in the test mode it changes because in the test mode this data comes from the serial input; so like here. So, from the testing from this data that is coming from this core so that may be used for from the adjacent core may be the test mode, which may be sending the test pattern so it will be loaded into this chain. So, instead of be delivered to the router for transmission to the next one, it will be delivered to this chain so that the test pattern gets loaded and this test pattern gets applied to the router and core and the responses are collected and then transferred to the next level. So, this way the same module can be used for mission mode operation and test mode operation.

So, for the test scheduling problem we can do it like this. So, first of all; this is again a heuristic algorithm, this is not an exact one; so heuristic algorithm. So, what it does it permute core in the order of test time. So, we arrange the cores in terms of their test time whichever core requires highest test times so that is done first and that way.

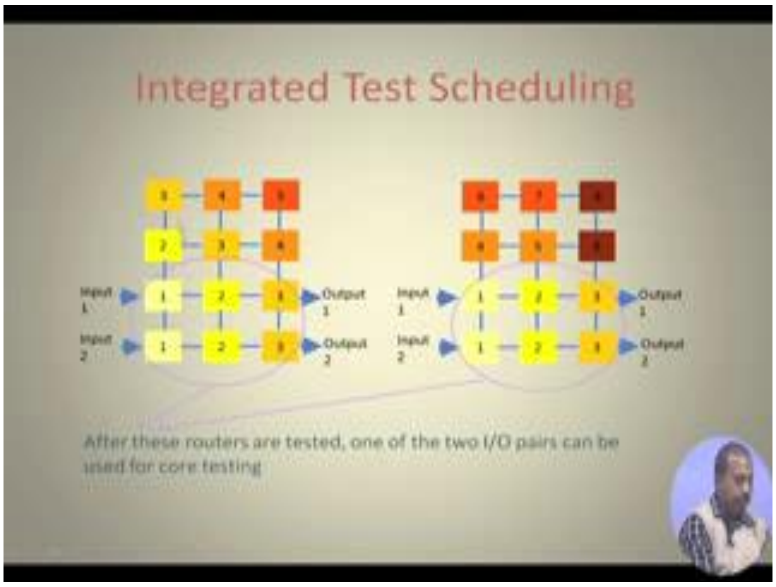
And permute all input output pairs. So, we try out some I O pair permutation. And for each permutation we find some free I O pair. So, it take up the core and see whether any I O pair is free or not, check for resource conflict, schedule the core. Now after scheduling the core we need to have this, the router on a path should be all tested before functional cores on that path to be tested. So, this is the additional thing that is ensured.

So, if you take this routers and cores as two integral module and all of them are to be tested and there is a precedence relationship we can say that it gives rise to precedence relationship between the test module; that is router testing should be done all the routers on the path should be tested first then only the a course on the path can be tested. So, this gives rise to some precedence constant. And then we can use this constrain test scheduling problem that we have seen previously in the SOC testing part. So, that can be utilized for doing the testing. Of course, with the modification that now it has to go through these routers only

Routers can be tested concurrently with core; so this is possible. So, when we are testing one router one core, so we can test more routers because they are not going to conflict. And this one if we assume that the router responses are evaluated locally near the router in that case we do not need to transfer router responses to the I O to the output port. So, that way it will give rise to saving.

So, at least one I O pair should be used for router testing at anytime, because I have to always do that router testing continually to make this core testing problem proceed to proceed through the stages. So, I need to have this router testing done continually. So, this is the thing that we have to have one I O pair dedicating for testing the routers.

(Refer Slide Time: 09:44)

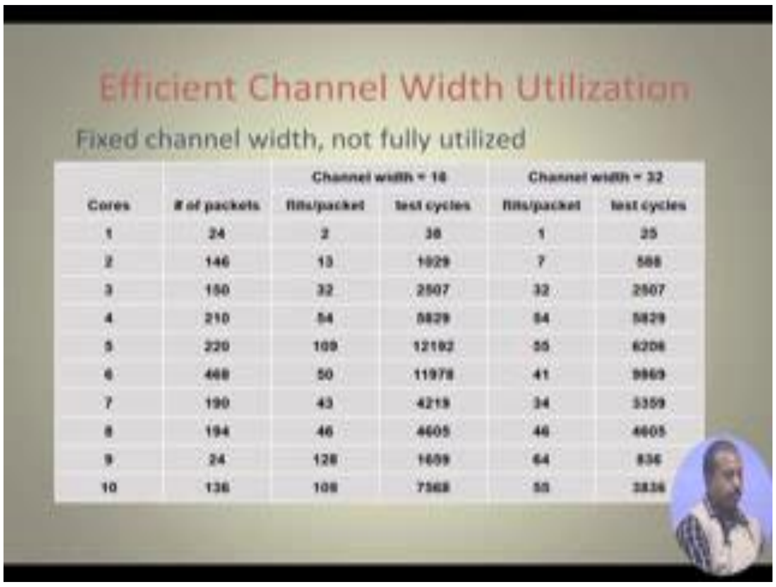


Integrated Test Scheduling

After these routers are tested, one of the two I/O pairs can be used for core testing

So, this is the integrated test scheduling problem. So, it says that all these routers are tested one of the two I O pairs can be used for testing. So, we are first this input 1 there are two I O pairs input 1 and input 2. So, since these are the channels through which the test will be done, if we are preceding like this then these I O pairs are these routers are to be tested whether this is for the multi cast situation, this is for the unicast situation, but whatever we do we need to test the routers on the path. So, all these routers are tested one of the two I O pairs can be used for core testing.

(Refer Slide Time: 10:29)



Efficient Channel Width Utilization

Fixed channel width, not fully utilized

		Channel width = 16		Channel width = 32	
Cores	# of packets	flits/packet	test cycles	flits/packet	test cycles
1	24	2	38	1	25
2	146	13	1029	7	588
3	150	32	2507	32	2507
4	210	54	5829	54	5829
5	220	109	12192	55	6206
6	468	50	11978	41	9969
7	190	43	4219	34	3359
8	194	46	4605	46	4605
9	24	128	1659	64	836
10	136	108	7368	55	3836

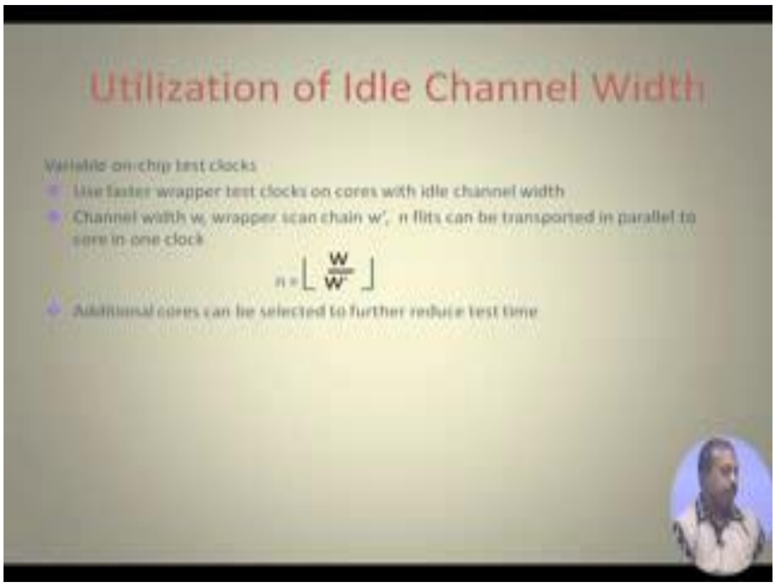
Now, channel width utilization if we see then. So, if we have situation like this that if the channel width is say 16 bit. So, we say flits per second; actually what is happening is that if we have got say- if we have got a packet P between one router to the next router I have got an number of parallel lines. Now, in one time you need how many bits we can transfer; so that is equal to the width of this channel fine.

Now if the width of the channel is say 16 bit and a packet has got say 32 bit; that means, I have to transfer it over two successive transfers; send the packet over two successive transfers. So, this is known as flits. So, in one clock how much data you are transferring; so in NOC terminology that is called a flit. So, we have got something like flits per packet, per packet how many flits are necessary. And we know that this transfer is taking a flit and one flit takes one clock cycle to traverse through a link and depending upon the router design one flit may take 1 to say 5 cycles it would go through the router. So, that depends on the router design, but that is known at the time of design.

So, you see that if we have got say 24 packets flits per packet is two and in this the case test cycles needed for this one is 38. So, this is basically after taking care of this over lapping this parallelization and pipe lining of packets being transferred and all that, so it can be computed; that is equal to 2; because in one 16 bits to be transferred and the next one is 8 bit, 8 bit will be transferred as 16 bit. So, this requires 38 clock that is cycles. So, this is similarly if there are 146 packets for core 2. So, there are 13 flits per packet and this is the number of test cycles, whereas for channel width 32 we have got the corresponding figures like this.

So, since the channel width is fixed, so between two routers the channel width is not flexible unlike this SOC testing where the TAM width may be flexible may be fixed. And we have seen that making TAM width flexible would reduce the test time significantly. So, that way that option does not exist in NOC testing, so channel width is fixed. As a result there will be lot of channel width is not fully utilized; like here the first time I am sending 16 bit, but for the next time I am sending only 8 bits; but I have to have data for 16 bits in the second case as well.

(Refer Slide Time: 13:29).



Utilization of Idle Channel Width

Variable on-chip test clocks

- Use faster wrapper test clocks on cores with idle channel width
- Channel width w, wrapper scan chain w', n flits can be transported in parallel to core in one clock

$$n = \left\lfloor \frac{w}{w'} \right\rfloor$$

- Additional cores can be selected to further reduce test time

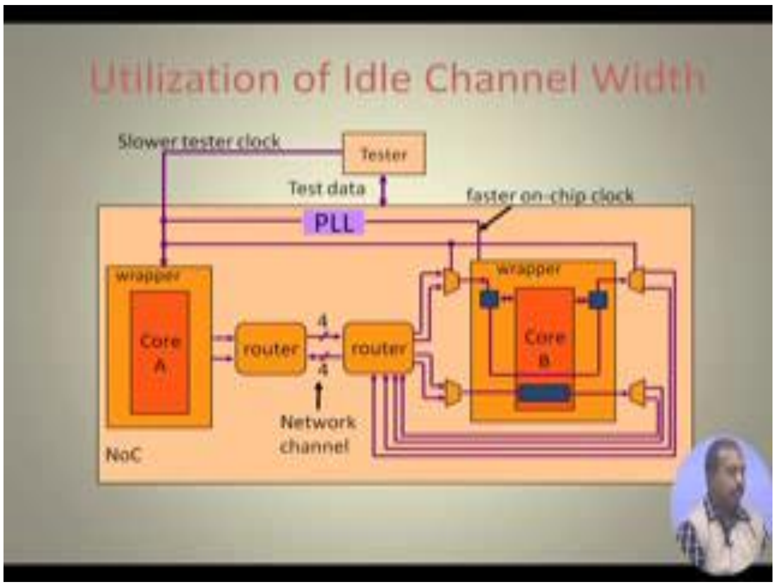
So, how can we use this idle channel width? We can have some variable on chip test clocks. So, this on chip test clock, if it can operate at different frequencies. So, we use faster wrapper test clocks on cores with idle channel width. So, they have got idle; the channel is remaining idle we can transfer more test data on to that time. As a result these cores they will get more test patterns in shorter time. So, this can be useful, this can be done and we can use this idle channel.

So, if the channel width is W and this wrappers scan chain is W dash, n flits can be transported in parallel to core in one clock cycle n is equal to W (Refer Time: 14:20) W dash, because wrappers scan chain W dash so that means in one transfer I can need only W dash bits. So, if I have got channel width given is W which is for example; if say W equal to say 32 and this W that is the channel width from channel width of the NOC. And for a particular core this wrappers scan chain width W dash equal to say 8; that means I can transfer 32 by 8 that is 4. So, four numbers of flits can be transported in parallel.

So, that way the destination it will receive more amount of test data per unit time. So, if the destination has got; if the core can be tested at a higher frequency then this can be used for that. So, additional course can be selected to further reduce test time. So, you can also say that in this test time I will not be transferring a test pattern only for this core, but for some other cores as well. It actually violent to some extent that reservation policy that is the resources are reserved for the test session for a core, but the that variation is there but at the same time it saves the transfer time for the other cores. So, that way we use that other core data to be transmitted while we have got the idle time for the first core.

So, this way we can utilize the idle time of the channels.

(Refer Slide Time: 16:09)

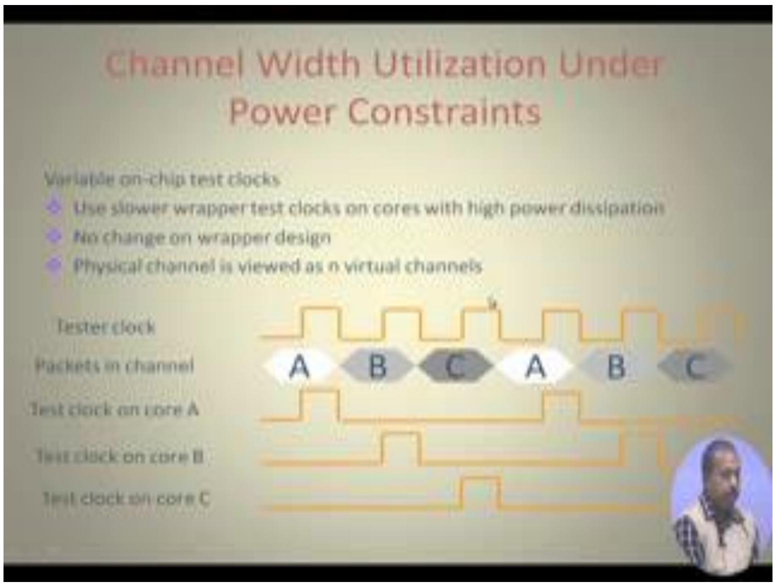


So, this is the situation. So, this tester; this slower tester clock can be given to the wrapper. And then we have got a phase lock loop which is on chips, so that generates from this slower clock it generates some faster clock. And this faster clock: this wrapper is operating at a faster clock compare to this wrapper.

Now, the test pattern, so that is being transferred from this tester the test data is getting transferred. So, it can be transferred to this wrapper at a slower rate because it is operating at a slower frequency, for this wrapper the test data can be coming at a higher rate. So, from core A to core B if the test patterns are going to be transmitted then through this network channel, so this is 4 bit 4 bit so that way it going to the router of the next level the four lines.

However, here which is operating at a higher frequency, so more number of test bits have can come to this core B. And this wrapper by doing a multiplexing so it can take care of this, because for some time it can send the data, select line; it can send that the data from it this router data it can be send through this multiplex to this wrapper chain or it can be; either of this two lines can be send to this wrapper chain. So, if the more data is arrived here then they can be transferred using this clock to this individual (Refer Time: 17:58) cells this buffer cells.

(Refer Slide Time: 18:00)



Channel Width Utilization Under Power Constraints

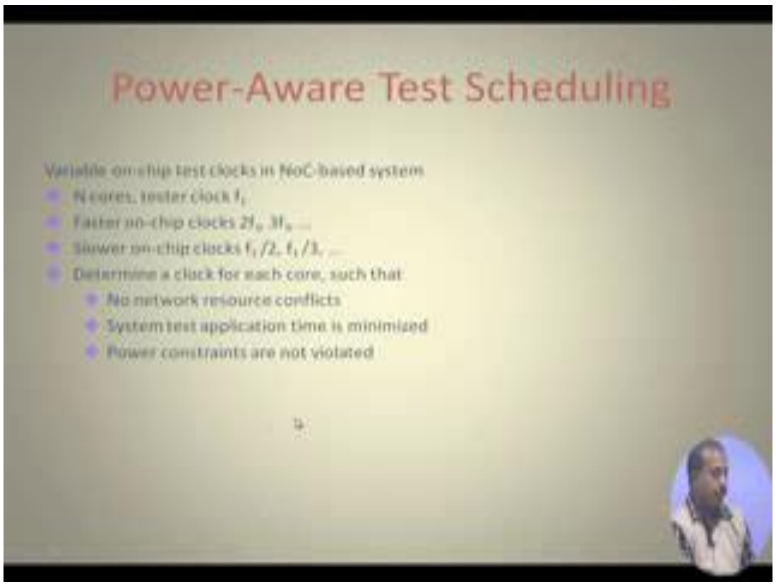
Variable on-chip test clocks
- Use slower wrapper test clocks on cores with high power dissipation
- No change on wrapper design
- Physical channel is viewed as n virtual channels

So, use slower wrapper clocks on cores with high power dissipation and the naturally no change on the wrapper design policy, but physical channel can be viewed as in virtual channels now. So, physical channels; so this is the tester clock that is applied. So, packets in channel are like this A B C; first A B C- A B C like that. So, there is a (Refer Time: 18:24). So, this clock on core A is applied as in this way it gets the core for it gets the test for test pattern for A and so whenever this A pattern comes a packets are put on to the channel test clock is given to the core A. Whenever b (Refer Time: 18:42) are put this is given on this clock is given to core B.

So, this tester clock from this we derive three clocks; this test clock for core A test clock for core B test clock for core C. So, this tester clock is high frequency so it is putting the test data for these channels A B C one after the other on to the channel in a multiplex fashion. And then at the individual cores they are sampled at their own frequencies as a result they get the original test patterns.

So, physical channel that we have it is viewed as n virtual channels. So, if there are n numbers of core data core test data that is being sent over the same channel here then that is basically n virtual channels are created. So, all those n channels getting data simultaneously and as if there are data simultaneously and they are getting sampled at the corresponding clocks.

(Refer Slide Time: 19:43)



Power-Aware Test Scheduling

Variable on-chip test clocks in NoC-based system

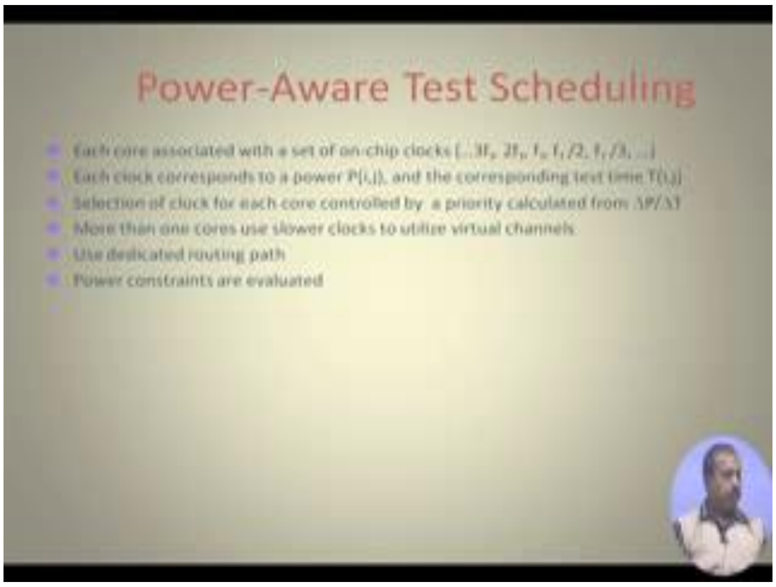
- N cores, tester clock f_t
- Faster on-chip clocks $2f_t$, $3f_t$, ...
- Slower on-chip clocks $f_t/2$, $f_t/3$, ...
- Determine a clock for each core, such that
  - No network resource conflicts
  - System test application time is minimized
  - Power constraints are not violated

Next, we will look into another very important issue like how to handle this power aware testing scheduling problem when we are talking about testing of this NOC based system. So, we have got say n cores and the tester clock is f T. We can have some faster on chip clock 2 f T, 3 f T etcetera. And then we saw slower on chip clock f T by 2 f T by 3 f T. So, these are the derived clocks from the system clock.

Then the test scheduling policy it is not only determining the assignment of cores to I O pairs to their schedule all that, it also needs to determine a clock for each cores; so which core has to be tested on at which clock such that there is no resource conflict that is the first thing then the system test application time is minimized, overall test time gets minimized and power constraints are not violated so power constraints are to be taken care off.

So, these are three things to be done and that is actually the power aware test scheduling problem.

(Refer Slide Time: 20:54)



Power-Aware Test Scheduling

- Each core associated with a set of on-chip clocks {…3f_t, 2f_t, f_t, f_t/2, f_t/3, …}
- Each clock corresponds to a power P(i,j), and the corresponding test time T(i,j)
- Selection of clock for each core controlled by a priority calculated from ΔP/ΔT
- More than one cores use slower clocks to utilize virtual channels
- Use dedicated routing path
- Power constraints are evaluated

So, each core as got associated with it a set of on chip clocks. It is assumed that all these different clocks are available and this is actually done at the design time. So, designer might have derived a number of clocks, it is normally not the case that for testing we use number of derived clocks. So, whatever derived clocks are available from the design side we have to use it. So, if it is the case that so many derived clocks are available 3 f T, 2 f T, f T is the normal rate and then f T by 2 and these are the slower clocks these are faster one. Then each clock corresponds to a power P ij and corresponding test time T ij, because power consumption the dynamic power consumption is proportional to the frequency. So, as frequency increases the test power consumption also increases. So, we have got this clock. So, for every clock we can find out what is the corresponding power consumption.

Similarly, if you are testing at a higher frequency the test time will be less, because it will be able to apply more number of test patterns in a shorter time so that the test time will get reduced by a factor of this higher frequency rate. On the other hand if you are trying on a slower clock side then this power consumption will be low, but this test time will be also be high.

Selection of clock of each core is controlled by a priority calculated by delta P by delta T. So, delta P is if you go to that particular frequency what is increase in the power consumption, and delta T is what is the increase in the time requirement? So, this ratio if you take then if some core has got has a very high figure of delta P by delta T; that means it requires lot of power to be consumed, but without much saving in the test time.

On the other hand if these value is low; that means, it may be possible that is requiring very less amount of power increase, but at a considerably high saving in the test time. So, that way this delta P by delta T can be used for selecting the core and then clock frequency for a core. And more than one cores use slower clocks to utilize virtual channels; so this is other possibility. So, we use more than one core to use the same slower clock in a virtual channel facility. So, that we can see the virtual channel concept that can be used; so that is possible.

Then another possibility is that use dedicated routing path. So, we can have some instead of putting all traffic on to one side may be will be put some dedicated path for the individual core testing. And that way for xy routing it is definitely dedicated, but if some other routing policy is followed. So, it may be that for some part of the router, some part of the network if we find it is used very frequently then we may try to reduce the load on that and try to give some idle time for that path and give it to some other path. So this way we can put some time gap and on that path that way. So, dedicated routing path can help us in that way.

Power constraints are definite to be evaluated. So, power aware test scheduling tries to address all these issues.