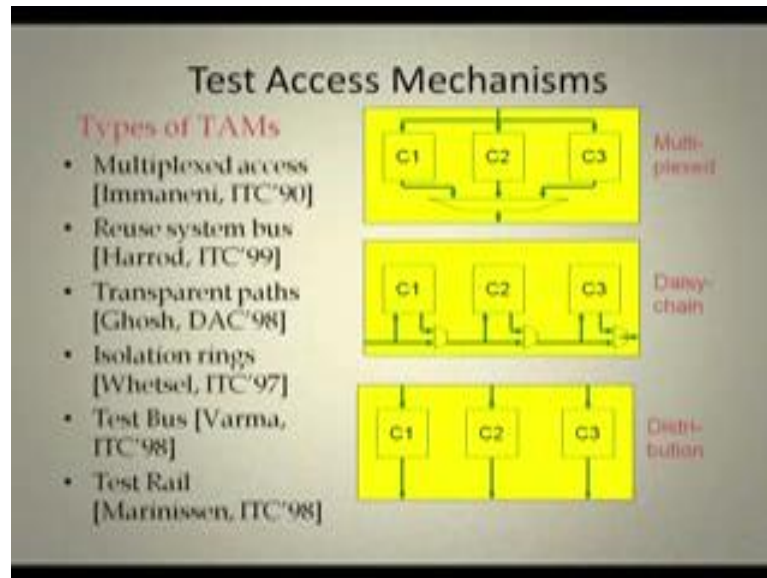


Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture - 48
System/Network – On – Chip Test (Contd.)

(Refer Slide Time: 00:26)



So, after the wrapper has been designed the next thing that we need to do for SOC testing is designing the test access mechanism. So, test access mechanism means from the system on chip pin, some pins will be dedicated for supplying the test patterns from the automatic test equipment to the SOC; and from those pins we need to transfer the test patterns to the respective code that we want to test.

Now, as we know that it is if there are large numbers of cores, so it is not possible to have dedicated lines for each of these cores. So, many of these cores are to be put onto the same set of test pins. So, and also there is a question of doing an allocation like may be after consulting with the designer it is found that we can accommodate say 60 such pins or 60 such inputs for the for the test pass design part. But out of the 60, so if we say that all 60s will be dedicated for testing one core, so that may be an inefficient mechanism. So, we may need to find out a partitioning of those 60 lines, so that it is divided into say a number of subsets and each subset is dedicated to testing a subset of cores that are there in the system. And apart from codes, so we can also test the user

defined logic UDL and the interconnections, so as we have seen in different testing mode in the wrappers, so intest, extest etcetera.

Now, about the test access mechanism, so several types of test access mechanism have been proposed in the literature. So, first one is multiplexed access. So, in a multiplexed access, this test bus is coming and it is divided into three cores, so C 1 C 2 and C 3. So, the test pattern which is meant for C 1, so there may be some device id register with this C 1, C 2, C 3 that will identify whether the test pattern is for C 1 or for C 2 or for C 3. In the worst case the same test pattern will be distributed to all the cores, so that way that may or may not be beneficial for some cores, but it is also somehow the pattern is transferred to the core.

And at the output, so there is a multiplexer. So, these responses from these cores are collected, and it is passed through a multiplexer that combines them onto the bus. So, this TAM is consisting of this line which is coming from coming to the system chip from the ATE and the lines which is going out of the system chip to the ATE. Now, in between we have got this multiplexed access. So, at one point of time only one core is being tested

So, the basic problem with this type of structure is that it will require large amount of time. Since this testing is totally serial, so we need to consider them serially one after the other all the core, so it will take long time. But at the same time it will have the flexibility of having the minimum power consumption because at any point of time only one core is being tested. So, the power consumed by that core will only come into picture. So, if one core consume the maximum power consumed by any core is say p then the maximum power consumption is given by p . However, the total the maximum power consumption of the intact test session is given by p . However, it can; so the since the test is prolonged over say long time, so this test time is may be quite high and that will cause the testing cost to go high as well.

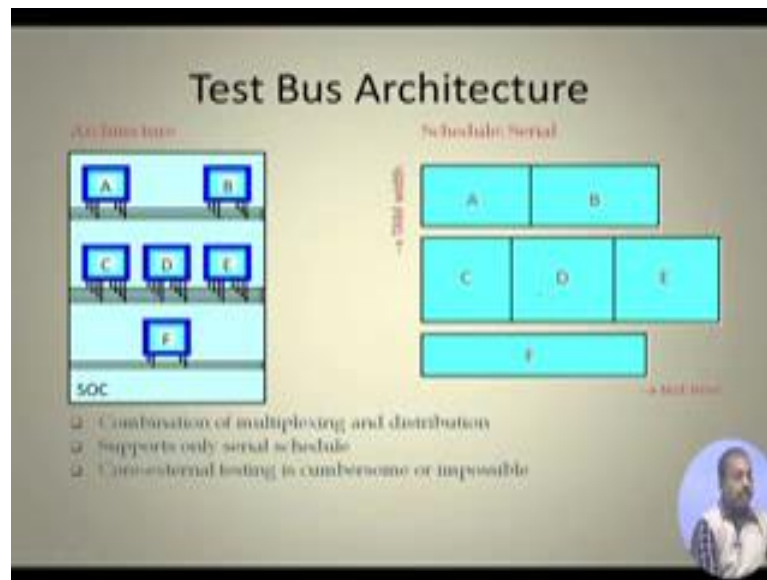
Second approach is daisy-chained approach. So, where we have got this the say the test bus is running; and from the test bus this input is going to the core and it is passed to the either the response will be passed or the test bus will be passed. So, if the response is passed then in the remaining part of the bus, it will carry this response through the bus. So, here also at one point of time only one core is being tested, but instead of having a

one multiplexer here, so we have got multiplexers at the end of a individual core just to select the response which response to be carried out. So, this is some sort of daisy-chained structure because if C 1 is not using the test bus, then only the test bus will be available for C 2. If C 1 and C 2 both are not using the test bus then the test bus should be available for C 3. So, this is known as the daisy-chained approach.

Another possibility is the distribution. So, that point I was talking of at the beginning is that in the first two cases the entire test bus is getting allocated to testing of a particular core, but that may not be very beneficial because that makes the testing fully serial. So, if we can distribute this test bus into a number of sub buses, like here we have got we have divided them divided the test bus into three parallel buses sub buses and they are actually feeding the individual cores. So, the point is that here I can think of some parallel testing like a sub this C 1, C 2, C 3 they can be tested in a serial fashion. They can be tested in a parallel fashion as well may be C 1, C 2 are tested parallelly, C 3 tested separately or all C 1 C 2 C 3 are tested parallelly.

So, it depends like, but of course, again the test bus is getting distributed it is getting divided into the three into the three parts. So, the width of individual test buses are going to be small, so from the wrapper design example we have seen that if we allocate lesser number of lines for that parallel access lines lesser number of WPI lines to individual cores then the test time of that core will also be high, so that is there, but still we have to decide on how this partitioning can be done. But if the partitioning is followed then we can go for parallel testing of course, we need to take care of the power consumption that is there in the that will come if a number of cores are tested in parallel. So, there are several approaches like multiplexed access, then there is reuse system bus then there is transparent paths, isolation rings, test bus, test rail etcetera which are actually some combination of these philosophies. So, we will look into them in the successive analysis.

(Refer Slide Time: 07:13)



First of all the test bus architecture, so in the test bus architecture, so this SOC; so these are actually the test lines that are running. So, here the width of the original bus is 3 plus 4 plus 2 that is 9; and that 9 width bus is divided into three sub buses you can say. So, those I can say there are three buses; as if they are three buses one bus is of width 3, another of width 4, another of width 2. Now, this core A and core B, so they are put onto the bus number one having 3, width equal to 3 bus width equal to 3. C, D and E they are put on bus number two having width 4; and the core F is on bus number three of width 2.

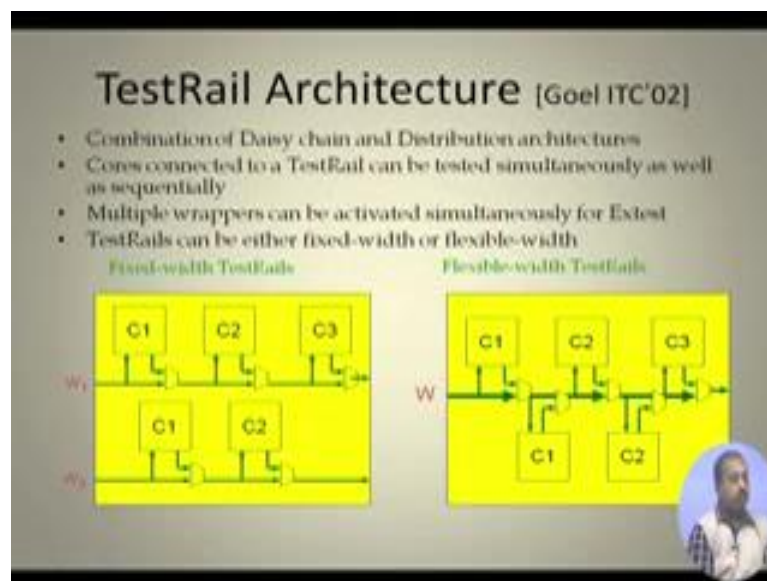
Now, so this A and B, they cannot be tested parallelly, because they are on the same bus. Similarly, C, D, E cannot be tested parallelly because they are on the same bus. But I can parallelly test A and C, I can parallelly test A, C and F provided other constants are met like the power constant precedence constant, so those are met, so we can go for parallel testing of all the three or these cores on the three parallel buses.

So, it is a combination of multiplexing and the distribution. So, we are distributing the original TAM width w of 9 into three buses of width 3, 4 and 2, so that is the distribution. And it is a multiplexing as well because at one point of time, only one core on a bus will be able to use that bus. So, it is a multiplexing. So, it supports only serial schedule. So, serial schedule means one after the other, they are to be tested. So, A and B on the same bus they can be tested serially only.

Core-external testing is cumbersome or impossible. Core-external testing means you have to somehow put this interconnects onto say suppose I want to test an interconnect from A to E, then what is required is that on the A bus I have to put the value that we want to pass through the interconnect at this at the output. And then that output has to be sent to E, and again I have to sense it here. So, this is difficult because it is not happening on the same bus. So, two different buses are getting involved, when we are trying to test this interconnect, so that makes the external testing part a bit difficult. So, may be impossible also in some cases, if we do not have this type of buses available.

So, a serial typical serial schedule will look like this that A and B, so they are tested one after the other in this in the TAM and in the first TAM. Similarly, C, D and E, they are tested serially on second TAM; and F is occupying the third TAM. And the total test time is given by the maximum of test times of individual TAMs, so that is basically given by the TAM two in this example.

(Refer Slide Time: 10:20)



Then we look into test rail architecture. So, test rail architecture, it is a combination of daisy chain and distribution. So, previous one was a combination of multiplexing and distribution; so this is a combination of daisy chain and distribution architectures. So, core connected to a test rail can be tested simultaneously as well as sequentially. So, what is happening is that the test pattern that is going through this going into this rail on

this on this rail, so they can be, so in the initially these multiplexers they can be programmed, so that this test pattern reaches all the cores.

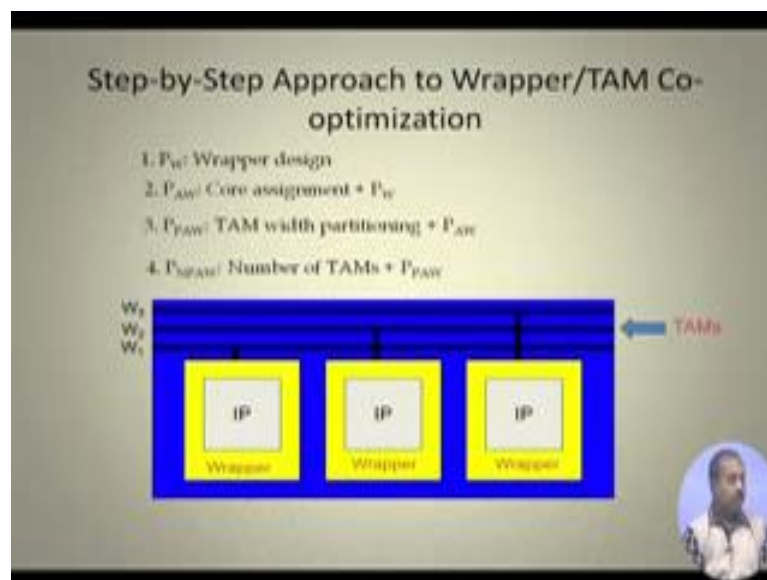
So, if it is possible that all the cores they are going to have the same test pattern, then the same test pattern can be fed or it can be that first these test patterns are shifted serially for all the three cores and then they are applied to the core. And then when the responses come then responses are also serially shifted out, so that can be done. So, that is why it is a combination of, so it can be tested simultaneously this C 1, C 2, C 3 can be tested simultaneously as well as sequentially, sequentially we can always do. So, we can do it sequentially.

So, here actually this width, so if it is; so a multiple wrappers can be activated simultaneously for extest because I can have this wrapper of C 1 and this wrapper of C 2. So, they may be there may be some connection between C 1 and C 2. So, for extest, so I can send the test pattern here and I can get the responses from this point, so that way I can simultaneously do this extest operation. And now this test rail can be of fixed width or it can be of flexible width. The first example this is a case of fixed width test rail. So, I have divided the total rail width W into two rails W 1 and W 2. So, on W 1, we have got C 1, C 2, C 3; and W 2 actually these two should be C 4, C 5, so this is wrong not C 1, C 2; C 4, C 5. So, this on W 2, we have got C 4, C 5 that way the code distribution is done over the rails.

On the other hand, in this case, in the second example, the width is not fixed. So, initially the total W comes here out of that some part of it may be fed here; may be for C 1, we have taken say W 1 w equal to say 5. So, five bits will be used as for transferring test pattern to C 1 and the remaining bits are coming here. So, now so that similarly for C 2, so we may have say we may require ten bits in the wrapper. So, the WPI lines for C 2 may be 10, so as a result we have got say we have got say 10 lines going from this W into C 2, so that actually gives the scope for parallel testing further. Because if the distribution is such that say the first 5 lines are connected to this C 1, second 5 lines are connected to this, third five lines are connected to this, then i can send fifteen bit pattern and that can feed this C 1, C 2, C 3 parallely. So, that can be done that is another way of getting parallelism with flexible width rails.

And of course, in this case, what is happening is that may be the preferred width preferred TAM width for C 1 is not same as the preferred TAM width of C 2, but we do not have any choice. Once we have made the distribution to W 1 and W 2, we have to test them with this fixed width only. So, the test time is going to increase compared to their best possible time; whereas in the second case since it is flexible, so we have got the choice. So, we can as long as the total time width requirement does not exceed W, we can distribute this time width into different cores differently. So, there is no distinct test rail onto which the cores are placed, so that becomes that gives the flexibility to the overall scheme.

(Refer Slide Time: 14:43)



So, there are a number of problems that have been formulated for this wrapper TAM Co-optimization problem. So, what is happening is that. So, this wrapper optimization and wrapper selection wrapper design and this TAM design, so they are going to be two problems which are dependent on each other. Like for every core we have seen that giving different TAM width in the wrapper, different giving different WPI lines in the wrapper, so it generates it different test time for the cores.

Now, at the same time when I am having a TAM given to me I need to partition them for example and this partitioning should be such that it is doing the best one for all the cores ideally, but that is not possible. So, we have to have some trade off. So, we have to find out a good TAM distribution, so that these cores are getting the TAM lines which are, so

that the wrappers will not require prohibitively high amount of test time in the after the wrapper design. At the same time, the TAM width should not be so high for a few cores, so that for other cores we do not have enough time width left.

So, this is actually the problem. So, we have to look into this wrapper design and this TAM partitioning together. So, there are some works in this direction, the first one in that direction is the known as the problem P W, which is actually the wrapper design. So, here only the wrapper design problem is solved. So, for a particular core, we find out what are the different wrapper configurations that can give it give very good results.

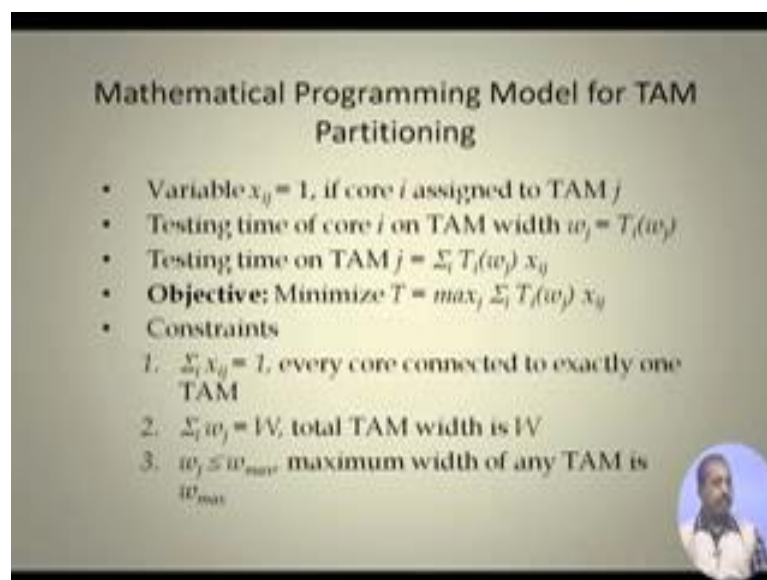
Then the next one next problem is known as P AW that is core assignment plus P W - so plus wrapper design. So, in the P AW problem it is assumed that the TAM partitioning has already been done, so that is like this W it is told that it is divided into three tams of width W 1, W 2 and W 3. And now the problem is to do core assignment like which core should be placed on which TAM; and then for example, in this case you see the first IP, it has been put on W 1; second IP has been put on W 2, third one on W 3. So, if there are more number of IPs they will be sharing this, W 1, W 2, W 3 buses, they will be sharing them, but how to decide like what is an optimal distribution of the cores to this buses.

And of course, so we have to this distribution of this cores to the TAMs. Now, of course, we have the issue like if we have different width given to different cores, then they will take different amount of time. So, overall objective is to minimize test time, but this P AW problem, so it solves this wrapper design problem plus core assignment problem together in an integrated fashion. So, it is expected to have better result than this P W based problem formulation.

Then the third problem formulation is known as the TAM width partitioning plus core assignment plus wrapper design. So, here it is told that total width is W and I want to distribute it into 3 or 4 or 10 number of TAMs. So, but how do you do the distribution, so that is not told; so the a solution to the problem will suggest me the distribution of individual of this TAM widths to different TAMS; and second thing is that it will do this core assignment and it will also do the wrapper design. So, it will do three things together. So, TAM width partitioning plus core assignment plus wrapper design, so that is known as P PAW.

Then there is another version of the problem which is known as P NPAW. So, where these number of TAMs is also not told. So, it is asked that you come up with a proper number of TAMs then do the partitioning then do the core assignment and also the wrapper design. So, this P NPAW is the most generic version of the problem when we are talking about this partitioned TAM based approach, but naturally this P NPAW is the most difficult problem to solve. So, there are many formulation that are been reported in the literature some of them are based on integer linear programming, some of them are based on heuristics, some of them are based on meta search techniques, so these have been reported. So, in the literature, you can find out, but this P NPW is the most generic one. So, normally there are lots of works solving the P PAW problem, and there are some works based on P NPAW problem.

(Refer Slide Time: 19:41)



Mathematical Programming Model for TAM Partitioning

- Variable $x_{ij} = 1$, if core i assigned to TAM j
- Testing time of core i on TAM width $w_j = T_i(w_j)$
- Testing time on TAM $j = \sum_i T_i(w_j) x_{ij}$
- **Objective:** Minimize $T = \max_j \sum_i T_i(w_j) x_{ij}$
- Constraints
 1. $\sum_i x_{ij} = 1$, every core connected to exactly one TAM
 2. $\sum_i w_j = IV$, total TAM width is IV
 3. $w_j \leq w_{max}$, maximum width of any TAM is w_{max}

So, we will see how can we formulate it as a mathematical problem. So, it can be formulated as an integer linear programming type of problem, so for that we have got some decision variables x_{ij} . So, x_{ij} this decision variable is equal to 1 if core i is assigned to TAM j . So, if I have got say n number of cores and M number of TAMs then there will be this x_{ij} variables it will be n into m number of x_{ij} variables will be there. Now, every TAM, so once I say that it is assigned to TAM j , so it is assumed that TAM j width is known. So, it is basically a formulation for that P AW problem where the TAM widths are known, the core assignment is not done. So, this width the width is known.

So, here the this will be, so this x_{ij} equal 1 means that core i is assigned to TAM j . Testing time of core i on TAM width of j which is given by w_j is T_{ij} , so these value we can find out from the wrapper design procedure, so we can do a best possible wrapper design with the available number of TAM lines. And then we can compute what is the time needed for transferring the test patterns to that core based on this particular TAM width and compute what is the total time needed. So, test time on TAM j , so test time on TAM j is given by this sigma of this T_{ij} multiplied by x_{ij} . So, if x_{ij} equal to 1 that means, the core i is being tested on TAM j in that case this T_{ij} factor will get added. If x_{ij} is equal to 0 that means, this core is not tested on this TAM, so it does not contribute to the testing time of TAM j . So, this computes the testing time of TAM j and the objective is to minimize this maximum value. So, this maximum over all TAMs this test time values, so that gives the total test time for the for the system.

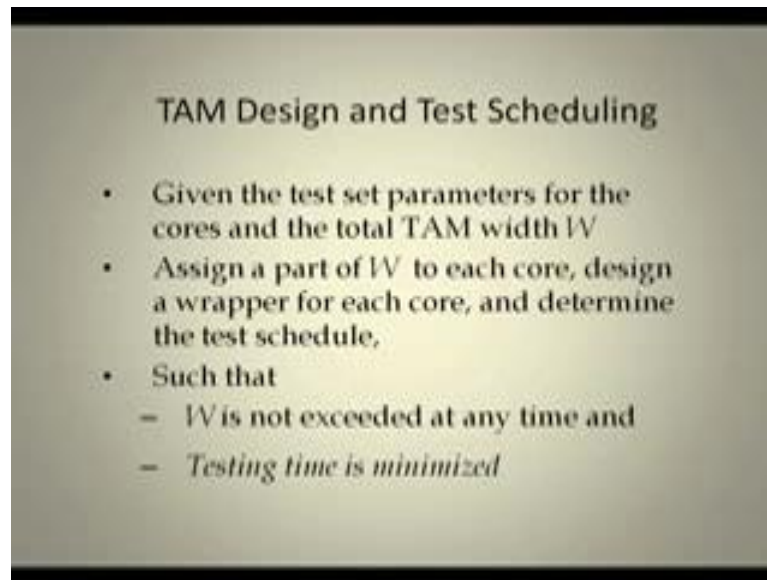
Like in this example that we have talked about; so, here you see that for this TAM the test time is given by the this plus this. So, these two TAMs summation; similarly, this TAM it is test angle C, D, E, so we take the maximum of these three. So, this gives me the maximum time. So, in the objective function we would like to minimize that maximum value, so that is what is written here. So, we want to minimize the total time T which is given by maximum of all these individual time testing TAMs.

Now, there must be some constraints that has to be satisfied like say sigma i x_{ij} equal to 1. So, every core must be assigned to only one TAM. So, for one particular core say core i x_{i1} is equal to 1, if it is assigned to TAM 1; otherwise it is 0. Similarly, x_{i2} is equal to 1 if it is assigned to TAM 2, and it is 0 otherwise like that. So, this sigma i x_{ij} , so this actually gives us the thing that this sum of all those x_{ij} variables and this equal to 1 signifies that only one of those x_{ij} s is equal to 1 and that means, the core i has been assigned to exactly one tam. So, that is the first constant.

Second constant that we have is that the width of all these TAMs. So, this partitioning that has been done, so sigma i w_j equal to Q . So, for all the tams, so this should be sigma one j this is the sigma 1_j sigma 1_j w_j should be equal to W - the total time width. And finally, any time width w_j must be less or equal w_{max} , so maximum width of any TAM cannot be this is another constant, so that is it cannot be more than the w_{max} some max value may be put. So, definitely it cannot be more than the total TAM width, so that is one type of w_{max} and also the designer may say or the designer or the equipment the

testing equipment may put a restriction on the number of such lines. So, there may be some restriction like this the w_j less or equal less or equal w_{max} that may be put into the system.

(Refer Slide Time: 24:15)



The slide is titled "TAM Design and Test Scheduling" and contains a bulleted list of requirements for the design and scheduling process.

- Given the test set parameters for the cores and the total TAM width W
- Assign a part of W to each core, design a wrapper for each core, and determine the test schedule,
- Such that
 - W is not exceeded at any time and
 - Testing time is minimized

So, the TAM design and test scheduling problem is given like this, given at the set of the set the test set parameters for the cores and the total time width w , so this for every core we know what is the test time for that whatever. So, if you put different wrappers then what are the corresponding test times, so it is not a single test time. So, the test set parameters mean that we know that for each core what is the test type if different TAM widths are assigned to it. So, different wrapper designs are done with them.

So, assigned part of W to each core, so that is basically designing for every core we are giving some TAM to it. And design a wrapper for each core, so we also have to design the wrapper and determine the test schedule such that W is not exceeded any time and test time is minimized, so that is the goal the test time has to be minimized. So, this formulation is a very generic formulation. So, it does not tell whether you are going for this partitioning of the TAM width or the fixed partitioning or flexible partitioning nothing; it just says that I have to assign some TAM width to every core. So, whether that is exclusive to that core or not, it does not tell anything like that. So, if we look into different variants of the problem, so we will see that it is giving us different types; it will

give us different formulations. So, that way it is going to give us different solutions for the problem under different constants

So, if you are telling that ok, I will do a fixed width partitioning then that will make it fixed width and then if we go for say flexible width then this again the W will be some part of W is assigned to individual cores, but that is more flexible, so that can be done. So, anyway, so this way the test time has to be minimized and this test time minimization is one of the constant, the other constant that we can have is basically the power limit constant that is at any point of time the total power consumption of the system should not exceed the power budget for the chip. And also we can have some total energy limit like the total this energy time into power.

So, if you sum them up, so you get the total energy, so that energy has to be minimized, so that this battery life can be extended and all that. So, there are many variants of this problem, but essentially this is the basic problem. So, we have this TAM design and the test scheduling. So, these are the two problems, and these two problems are solved together to get this test schedule for the SOC.