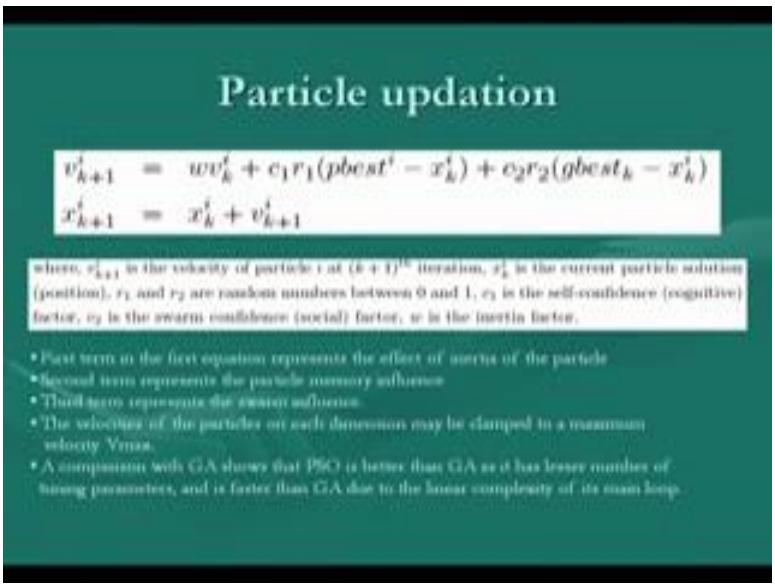
Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 38
Thermal Aware Testing (Contd.)

(Refer Slide Time: 00:23)



So, particle updation, this starts with the local best and global best information of the generation. So, local best as we have seen local best is the based over the entire generation for entire generations for which the particle has evolved so over its own history. And global best is the global best of the current population. So, if v i k is the velocity of the particle i at kth iteration, so at kth iteration the velocity of the particle was v k i. So, that is updated to v k plus 1 i using this formula. So, there are 3 factors in it, the first part of it is inertia, so there is an inertia factor w plus there is, so this is the difference in the intelligence, this is the current position of the particle and this is the local best intelligence of the particle. So, difference between them, so that gives the local knowledge like how far am I from the best position that I have known over the number of generations.

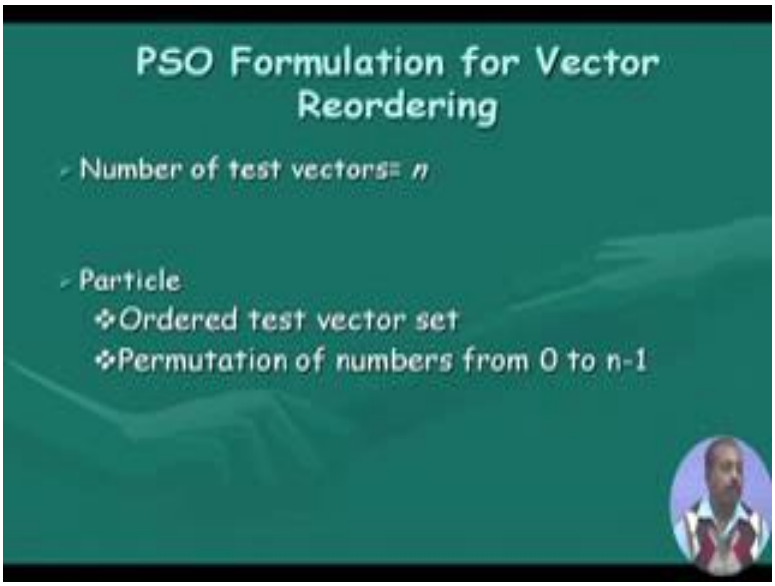
And this second part is the is coming from the difference from the global best that current prop generations of this is the global best estimation, and this is the current position of the particle. So, over the so in the current generation among the number with

the best position what is the position difference. So, this multiplied by this c 1 and c 2 they are called confidence factors. And this r 1 and r 2, they are 2 random numbers between 0 and 1. C 1 is called self-confidence because this is actually contributing to the confidence that I have on my own history, and this c 2 is known as the swarm confidence or the social confidence because that gives the confidence over the global situation the global best situation. And this r 1 and r 2, so they are having some values so, they are randomly generated otherwise what will happen is that this c 1, c 2 values, so they multiplication factors they will remain constant, so that is not desirable for a evolutionary algorithm, so it is because of that. And w is the inertia factor.

Basically a particle, so even if it wants to change, so when a bird is flying in certain direction, it cannot change its velocity all on a sudden, so that is because of that is giving contribution to the inertia factor. And based on its local intelligence and global intelligence, it tries to change its position by modifying the velocity. So, these are the other 2 factors. Now, once we have completed the velocity, since we are assuming the time is advancing in unit steps, so this next position of the particle x k plus 1 i is given by x k i plus v k plus 1 i. So, the first term in the first equation it represents the effect of inertia of the particle. Second term represents the particle memory influence. Third term represents the swarm influence.

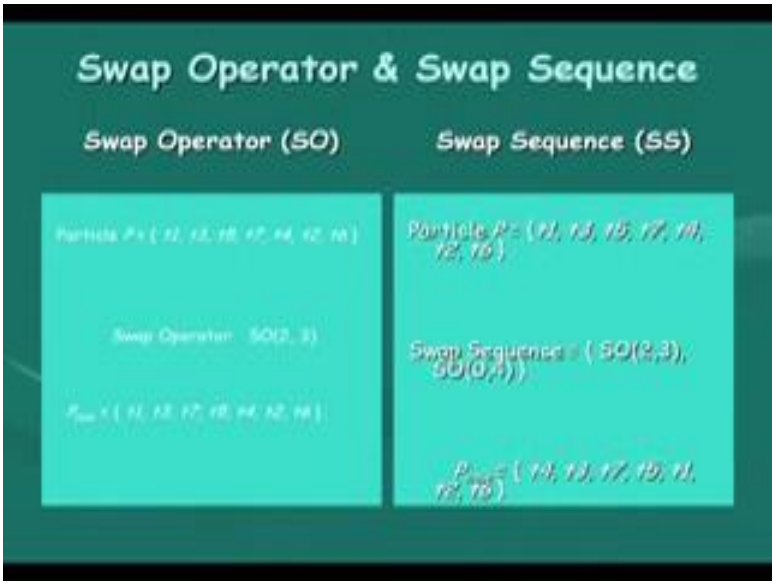
The velocity is now it may so happen that did you in some application, the velocity may become too large at some point of time. So, they have to be clamped at some times. So, velocities of the particles on each dimension, it may be claimed by a to a maximum velocity v max. So, if you compare the genetic algorithm with PSO, it is generally seen that GA has it has lesser number of tuning parameters and it is also faster than GA. And because of this main loop, it is just going over number of generations and number of particle, so that way the complexity is less. And it has been seen that in many optimization problems this particle swarm optimization it gives better result than this genetic algorithm based approaches.

(Refer Slide Time: 04:22)



So, what about this test vector reordering problem that we were looking into, and for that how can we make the particle swarm optimization formulation. So, suppose the number of test vectors is equal to n, so a particle is an ordering of this test vectors. So, if the text vectors are numbered from 0 to n minus 1, so the particle any particle is a permutations of numbers from 0 to n minus 1, so that is the particle structure that is taken for this reordering problem.

(Refer Slide Time: 04:56)

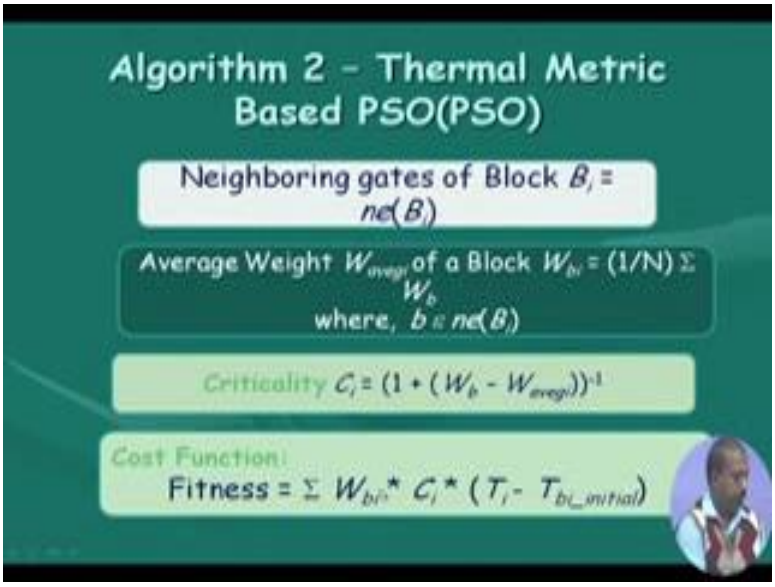


Next part is how can we change, how can one particle get modified. So, for that purpose,

so one swap operator is used; so if this is the particles say particle P 1 is t 1, t 3, t 5, t 7, t 4, t 2, t 6, so I have got this test patterns t 1 to t 7. So, in this example that pattern 0 has been excluded it has been taken from 1 to 7 that way. But whatever so if it is so to generate a new particle from this, so one possibility is to swap between 2 entries. Like I can have a swap operator that swaps between the entries 2 and 3, if entries 2 and 3 are swapped then you see it is the entry number 0, 1, 2, 3.

So, t 5 and t 7 they will be swapped and as a result it can generate a new particle t 7, t 5, now so that is by a that is a single swap operator. So, we can have a sequence of such swap operators which you call a swap sequence like say swap of first operator is swapping between 2 and 3, second operator swapping between 0 and 4. So, like that I can have a sequence of operations, so that is called a swap sequence. So, on a particle you can apply a swap sequence to generate a new particle P new.

(Refer Slide Time: 06:27)



So, with this, what we can do when one particle tries to align with the global best or local best, it can find out what are the swap, swap sink to be done in this in this particle to align it to the local best or global best. So, as we are telling that in this case, here it is a pbest i minus x k i. So, this formulation is modified, because it is no more a velocity, but it is some sort of discrete operation. So, what you will try to do is that we try to directly align a particle with its local best. And for that purpose, so we find out how many if this operation is not a negation, but rather it is finding out like what is the swap sequence, so

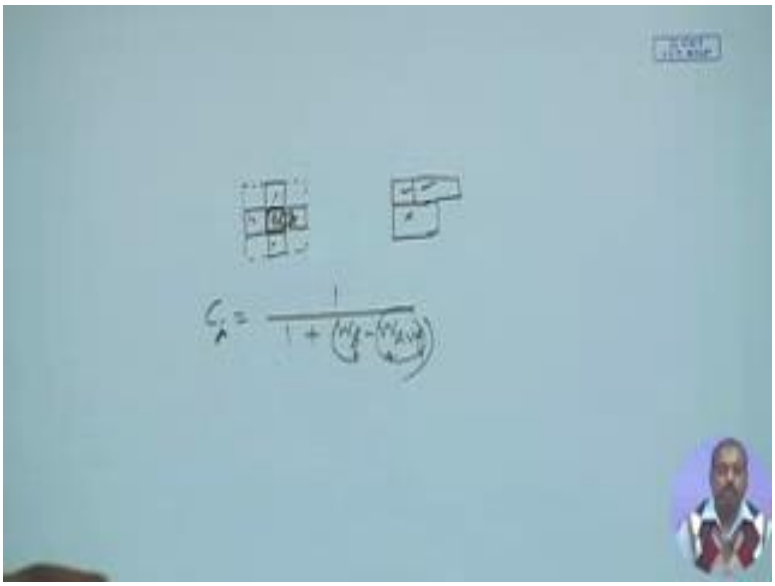
that this x k gate align to pbest. Similarly, in this minus operation is basically finding the swap sequence that will change this x k to the global best. So, that way this definition of this minus has to be modified.

And similarly this plus operator, they are nothing but some concatenation of those swaps; like this one says that there is no change. So, that is here the he particle remained unaltered, so I can say that swapping is basically identity swapping where every position is swapped to itself, so that is a that is the first sequence swap sequence for this. For this part, I have got a swap sequence; and for the third part also, I can find a swap sequence.

And then c 1 and c 2 multiplied by r 1 and r 2 respectively they can determine the probability with which this swap sequence will be applied on the particle. So, based on that, so it remains identical for this much of probability, it becomes align to this local best based on this probability and it gets align to global best with the probability of c 2 r 2. So, that is how this formulation is slightly modified compared to a continuous PSO. So, this is basically a discrete particle swarm optimization that has been used here, where the swap sequence actually determines how one particle get aligned to the another particle. So, the swap sequence is actually the velocity part that we are talking about.

Now, we can have now the second part of it is to find out this fitness function. So, how can we find the fitness function. So, one possibility is that we use a thermal metric for this fitness function. So, how can we find a metric. So, it is like this. Suppose, we are trying to find the criticality of a block Bi to be optimized thermally, its temperature needs to be reduced. So, we define the neighborhood of Bi, so neighborhood of Bi, it maybe 5 neighborhood it may be 7 neighborhood.

(Refer Slide Time: 09:25)



Like, if this is the block, so you can take these as the neighboring blocks. So, all of them this constitutes a 5 neighborhood. So, all these if this is the block B, so these are all the neighbors. And you can also take 9 neighborhoods like including this. So, whatever we do now it may so happen here the blocks are very uniform in nature, but it may so happened that this block is like this, there is another block like this, and there is another block like this. So, these 2 also become neighbors for this block.

So, if it is very regular if the block demarcation is very regular then you will get 5 neighborhoods, 9 neighborhoods like that, but if the block demarcation is not that regular then you can get different types of neighborhood. But anyway, so we find out actually the blocks which are surrounding the block Bi, so that is the neighborhood of Bi then we find out. So, for every block we had previously computed some weight of the blocks. So, weight of the block was computed by applying a sequence of random patterns and finding out like what is the criticality of this block, what is the weight of this block by finding the temperature of this block and the maximum temperature. So, here that weight factor is those weights are added, so this weight of this for each block we find out this we take those weights sum them up and take the average. So, W bi is the average weight of a block.

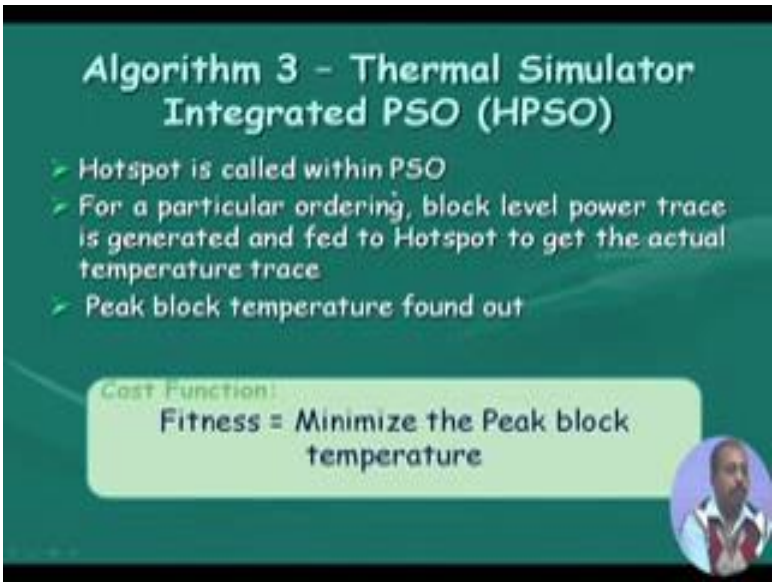
So, the once we have found out this average weight of a block, now criticality is defined as 1 plus W b minus W average i to the power minus 1. So, basically what we are doing,

for a particular block, we are determining what is for a particular block we are determining, what is its average? So, average in the sense that if this block says very hot and this block is relatively low cool, this temperature is relatively cool, then there is a high chance that in actual operation, the heat will go in this way as a result this block will no more remain that much hot. Whereas, if his block is hot as well as its neighbors are hot then there is a very high chance that this block will remain a hot, and it will contribute a lot to the damage of the circuits. So, it may cause the circuit to be damaged.

So, what is done we compute this parameter. So, if we look into this criticality value, so it is said that it is 1 upon 1 plus W b minus W average. So, average is the average heat that average temperature some measure of average temperature that we have in the neighboring blocks. So, if this difference is low that means, if this difference is low that means, this locks they are very high, their temperature values are very close to each other as a result their cooling chances are less. On the other hand, if this value is pretty high that means, though this block may be hot, but it surroundings are rather cool. So, it will not have that much effect in the heat generation process. So, it not build up that much of temperature.

So, what we do, we find out the criticality measure, this block is critical to be optimized provided its surroundings are also hot, so based on that this criticality measure has been defined. So, it is 1 upon 1 plus W b minus W average i. So, this is one this factor one has been added because this value may become 0 or close to 0. So, just to avoid that calculation error, so this one has been added. Now, finally we have define the cost function that is for a entire test set, so that is for every block we take the weight of the block that is how hot it is, how important it is to be optimized with respect to the; so all other blocks in the circuit. So, they are the W bi is calculated previously multiplied by C i that is a criticality value multiplied by T i minus T bi initial. So, T i is the temperature of the block that is resulting from by that is resulting from applying the test pattern set; and T bi initial is the initial temperature set. So, based on this, we can compute the fitness.

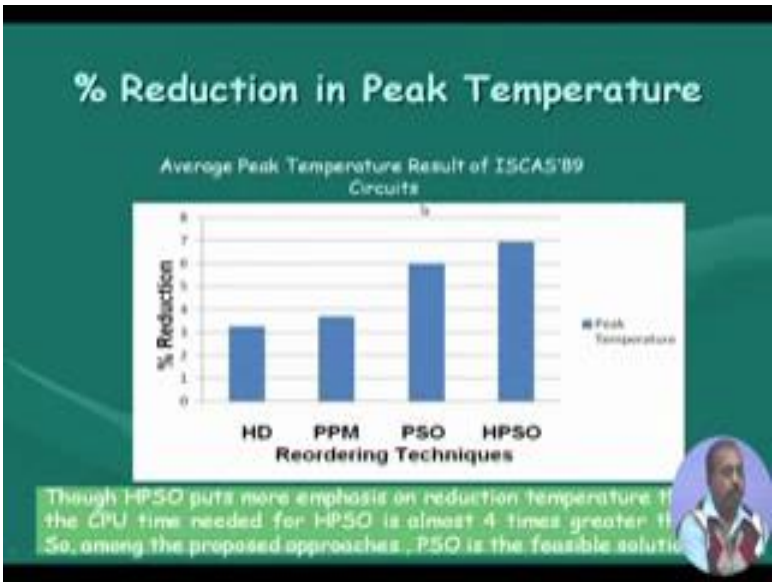
(Refer Slide Time: 14:09)



Another way for computing the fitness function is to call the thermal is to call the thermal simulator within the particle swarm optimization itself. So, for a particular ordering, block level power test will be generated and fed to the hotspot to get the actual temperature trace. So, and then we find the peak block temperature. So, fitness is the fitness function that will try to minimize block peak block temperature, so fitness function is the maximum peak block temperature that we get and we try to optimize it. So, this way we can go for this thermal optimization.

So, we have got one technique based on this cone based hamming distance measure minimization then we have got 2 the PSO based techniques one is by means of some thermal metric that based on some criticality measure, and the third one is a more direct method. So, here actually it is trying to integrate the thermal simulator within the optimization process. So, naturally the result of third one will be the best, but it will take more time for computation.

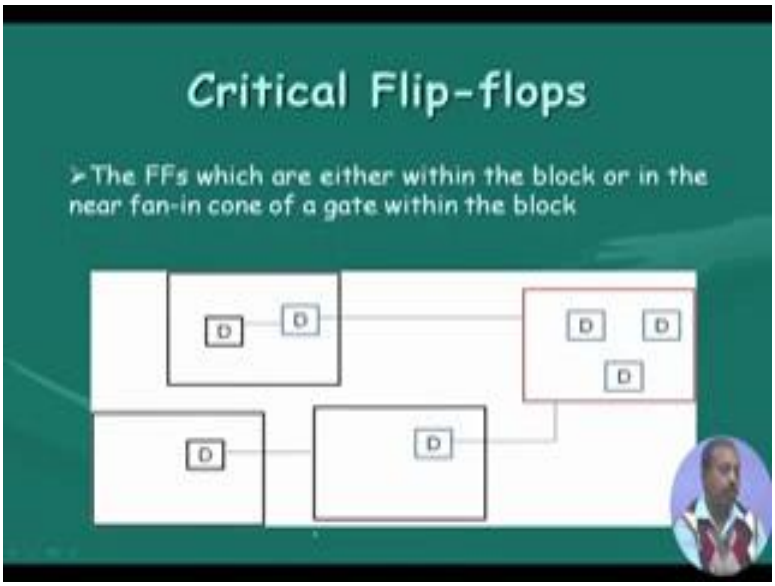
(Refer Slide Time: 15:18)



So, if you look into this graph, so you will see that percentage reductions if these are the reordering techniques. So, if you are doing this hamming distance based real minimization then this is reducing the temperature by about 3 percent. So, this PPM is a power peak power minimization process, so that actually does better than this hamming distance based minimization approach. And here it move goes more than 3.5 percent reduction. This basic PSO is that criticality based blocking factor, so that gives about 6 percent minimization and this is the about 7 percent when we directly integrate hotspot thermal simulator with the PSO optimization technique. So, these results are obtained over a benchmark set of circuit ISCAS 89 benchmark circuits. And over that the average results have been reported.

Next, we look into another way of minimizing this temperature during testing is via do not care filling. So, you already know that this do not care filling can be utilized for power minimization, but this can also contribute to temperature minimization. But one thing we must keep in mind that, we cannot go for this scan based minimization and all that because this scan the weighted transition metric based power minimization, so that does not have any effect here, because we need to simulate the circuit because it is very much dependent on the floor plan. So, we cannot directly say that if weighted transition count is less that means, temperature will be low, so that does not happen actually.

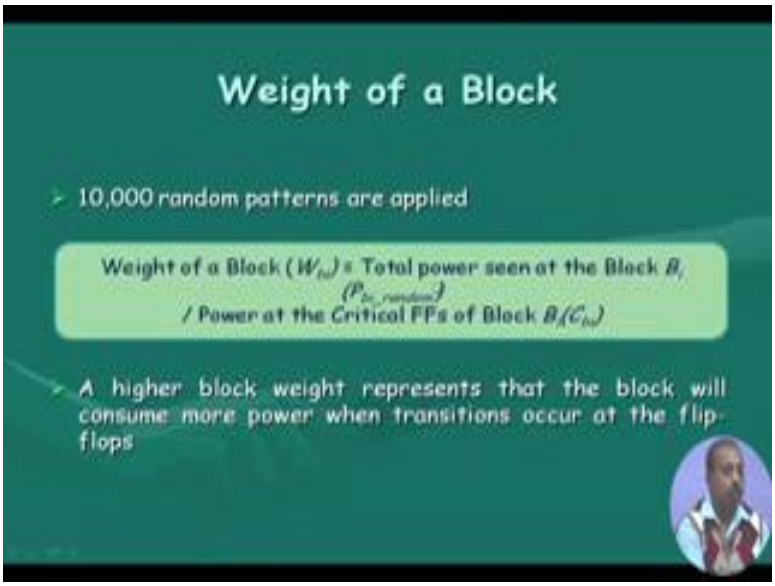
(Refer Slide Time: 17:10)



So, this is the way we approach this problem. So, we define some flip-flops. So, initially the circuit is divided into blocks as we have discussed previously. The circuit has been divided into blocks, suppose these are the various blocks, so we have got 4 different blocks here. Now, see suppose this is the block that we are considering. So, when this scan transition will take place, so all these flip-flops that we have in the blocks. So, they will be converted into scanned flip-flops. Now, when they are converted into scan flip-flop, so there will be transitions in these scan flip-flops, so we just take some sort of measure like how critical is it flip-flop for a particular block.

So, if a transition occurs in that block then, what is going to happen in the power and temperature profile of the block? So, we define something called the in the flip-flops in the fan in cone of a gate or a block. So, this is called the critical flip-flop. So, these flip-flops like say any flip-flop which is inside the block, so that is definitely critical for that block because any transition there will affect the gates in that block. And also the flip-flops in the previous stage like from this block this flip-flop is feeding this, so naturally this also becomes a critical flip-flop for this block and this also becomes a critical flip-flop for this block. So, this way all the critical flip-flops are identified.

(Refer Slide Time: 18:40)

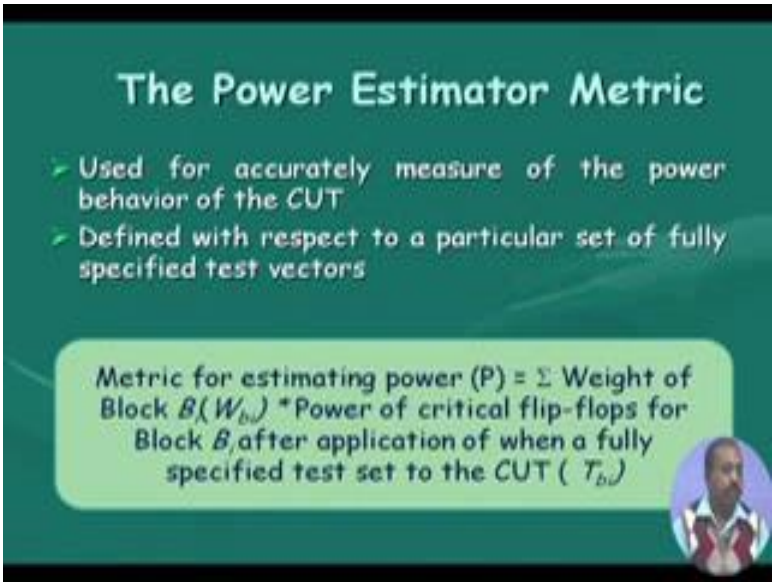


Once we have identified the critical flip-flops, then we apply 10,000 random patterns. Now, weight of a block is defined as total power seen at the block Bi, which is known as P bi random divided by power at the critical flip-flops of block Bi which is C bi. So, basically this P bi random, so we apply 10,000 random patterns and see what is the power consumed by this particular block. So, we can have some sort of power simulator which does this gate level simulation counting number of transitions and for every type of gate we can have gate and flip-flop. So, we can have an estimate like if this gate makes a transition then what is the power that is consumed, so that type of estimates we may have. And then by multiplying that by number of transition, so we can get some estimate about the power that is that will be consumed at that gate.

By now by summing them up, so we can get the total power seen at the block Bi. These divided by power at the critical flip-flops of block Bi. So, this is that is the power consumed by the critical flip-flops. So, that means, a higher block weight it represents that the block will consume more power when transitions occur at this flip-flops. So, if the block weight is high that means, if this flip-flops make transition then more power will be consumed. So, if this weight is low that means, even if this flip-flops they make transitions not much power will be consumed of course, the that is taking into view that the total power of a block, so that is more or less the fixed and we are putting more emphasis on this in the critical flip-flops.

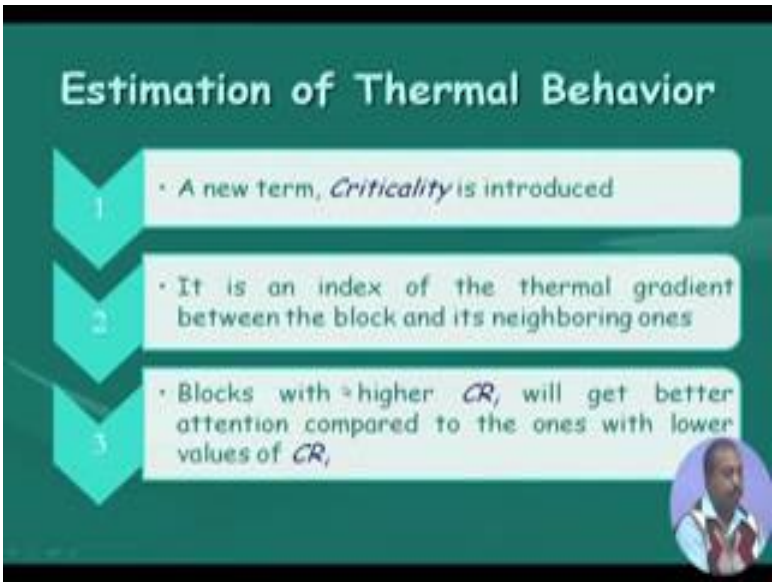
So, critical flip-flop, if they consume a lot of power; that means, and if the critical flip-flops consume lot of power and this power consumed of the block is also high that means, this ratio will be close to 1. So, in that case this factor will be the value will be high, as a result it will be weight of the block Bi will be high so that means, the block will consume. So, these flip-flop transitions will cause more power consumption in the block. So, during scan transition or during test application process, we have to try to reduce this number of transitions in these flip-flops.

(Refer Slide Time: 21:09)



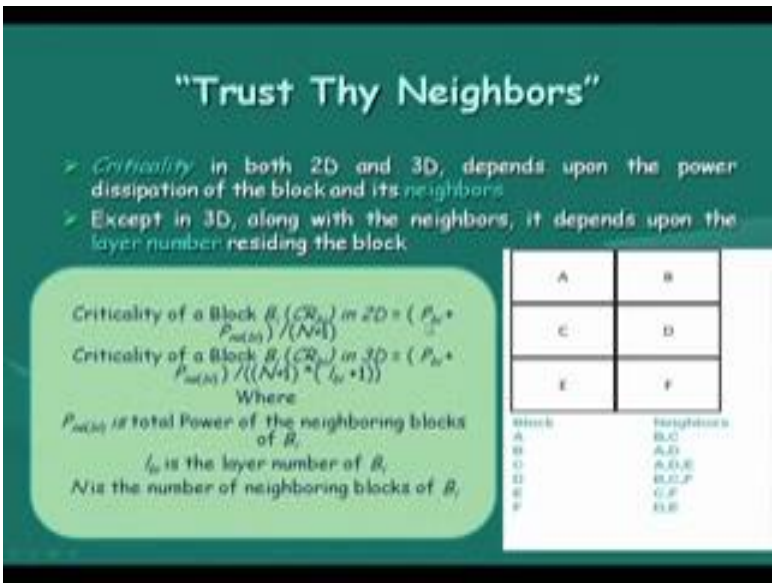
Then you take a power estimator metric. So, we accurately measure the power behavior of the circuit. So, defined with respect to a particular set of fully specified test vector, so what is this pa power estimation metric is the weight of block Bi either is W bi multiplied by power of critical flip-flops of block Bi after application when a fully specified test set to the cut T bi. So, after apply applying this, so these weights of the block multiplied by this power of this critical flip-flop, so that is giving us some estimate about the power some estimating power of the block.

(Refer Slide Time: 21:53)



Now, for thermal behavior estimations, so what we do we define something called criticality. So, it is an index of thermal gradient between block and its neighboring ones. So, it is to some extent similar to what we were discussing previously. Blocks with higher criticality will get better attention compared to one with lower values of criticality. So, this will be the first function that we will consider.

(Refer Slide Time: 22:17)

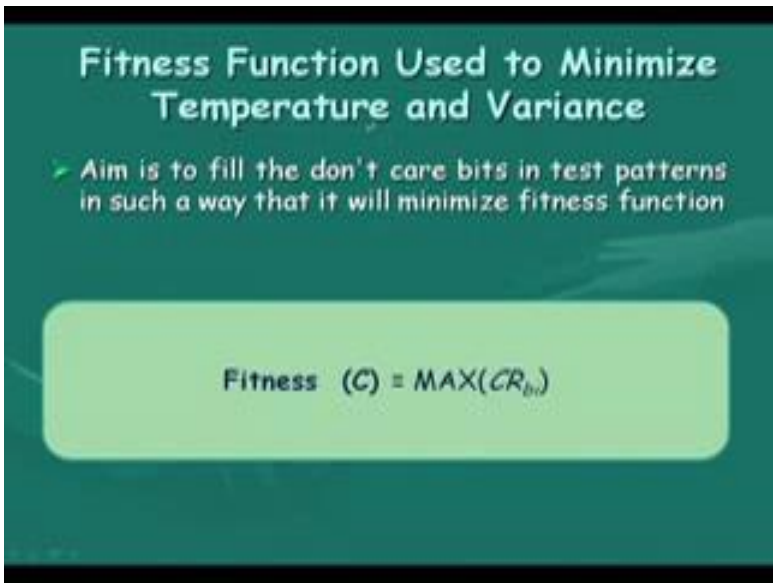


So, criticality of a block is defined as, so this is the power P bi, P bi is the power consumed by a block Bi, and this P ne Bi is the power consumed by the neighboring

blocks of the block Bi. So, block A has neighboring block B and C; block b has neighboring blocks A and D, so like that we have got this thing. So, criticality of a block Bi in 2 D, so this is P bi plus P ne Bi, so this is the power consumed when those test patterns have been applied divided by N plus 1. So, N plus 1 is the number of; N is the number of neighboring blocks of a block, so that is plus 1 has been added to avoid this 0 factor. So, P ne Bi is the total power of the neighboring blocks of Bi, and this P Bi is the power of the current block and ne Bi is the power of the neighboring block, n is the number of neighboring blocks.

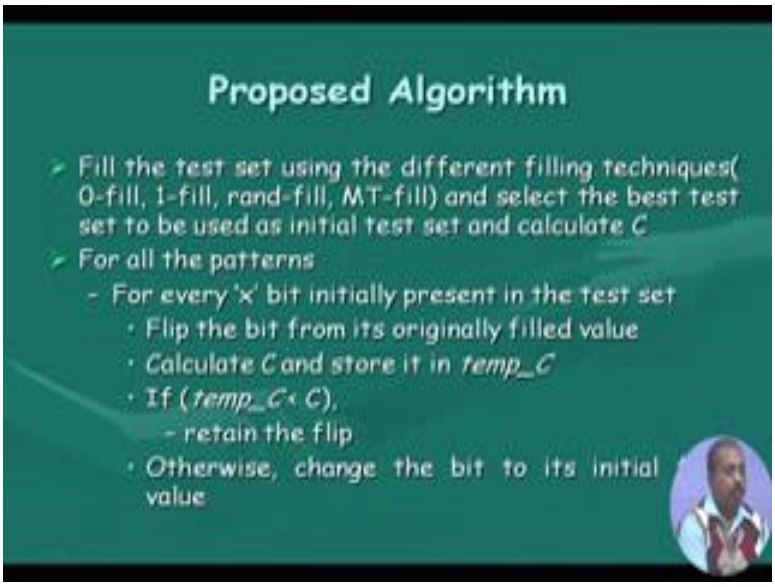
Now, for 2D, we have got in only the current plane we have got neighbors. For 3D, we may have neighbors in the other levels as well. So, this is basically so and also the layer factor becomes an important issue because in 3D we have got multiplayer design and the layer which is close to the heat sinks, so that is the heat generated there is much less compared to the heat generated at the upper layers, where it is away from the heat sink. So, that way this layer number has got an important role. So, a block which is located a high, so that will be that will be consuming that criticality is high compared to a block which is located down words in the lower layer which is close to the sink, and then this layer factor layer number factor becomes an important part. So, that is why in this cost function this layer number part has been introduced. So, l bi is the layer number of the block Bi and n is the number of the neighboring block that we have already seen. So, this way we define the criticality.

(Refer Slide Time: 24:36)



Now, fitness function that we used to minimize temperature is to we try to fill the do not cares bit in test pattern such that it will minimize the fitness function. So, this is the maximum of this criticality value. So, once we have defined this criticality values for every block, so then we try to maximize the criticality value that will this maximum criticality value. So, that sorry this maximum criticality value that we try to minimize, so fitness function it will try to minimize this maximum criticality value.

(Refer Slide Time: 25:12)



So, the algorithm that is used is some sort of bit flipping. So, it says that we have a particular bit in the do not care in the test pattern say, so we will try to sleep it and see whether its effect is reducing the criticality value or not. So, to start with, we fill the test set using different filling techniques a 0-fill, 1-fill, random fill, empty fill. So, 0-fill means whenever we have got a do not care bits, so fill it with 0. So, 1-fill means fill it with 1; random is randomly filling. Empty fill is the minimum transition fill the minimum translation fill means if the neighboring bits are bit is 0, then the x bit it will be filled with 0; the neighboring bit is 1 it will be filled with 1, so that is the empty fill algorithm minimum transition fill algorithm.

So, these are the well known techniques for do not care filling. So, we find out for every, so the given a test pattern set, we fill out this test patterns using this techniques and we calculate the criticality of every block which we call C. Now, what we do for every x bit that we have in the test pattern set we flip the bit from its originally filled value to the

other one. If it originally filled value was 0; it becomes 1 and vice versa; if it is was 1, so it becomes 0. Then you calculate this criticality of the value again C and store it in the variable temp C. Now, if this new criticality is max this C is actually that maximum criticality among all the blocks, so this that we have seen previously that they have cost function. So, this temp C, if this temp C is less than this C that means this flip was beneficial. So, we retain the flip otherwise we change the bit to its initially filled value and restart the process.

So, this way this algorithm works. So, it is it is a very greedy technique. So, starts with best possible filling pattern out of these well-known filling patterns 0-fill, 1-fill, random fill, empty fill, and then it tries to change every x bit to its complement value. And see, what is the effect? What is what is happening to this criticality value, the criticality value is improving then it is accepted. We will continue in the next class.