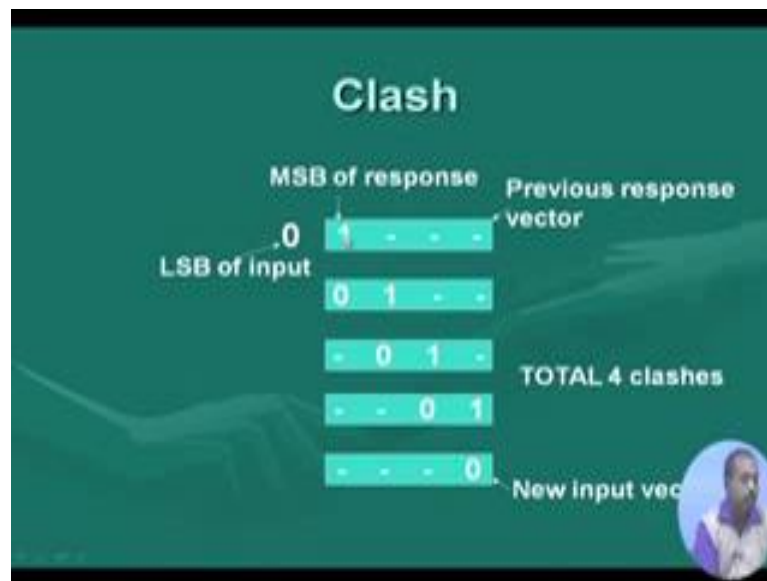


**Digital VLSI Testing**  
**Prof. Santanu Chattopadhyay**  
**Department of Electronics and EC Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture - 35**  
**Low Power Testing (Contd.)**

(Refer Slide Time: 00:22)

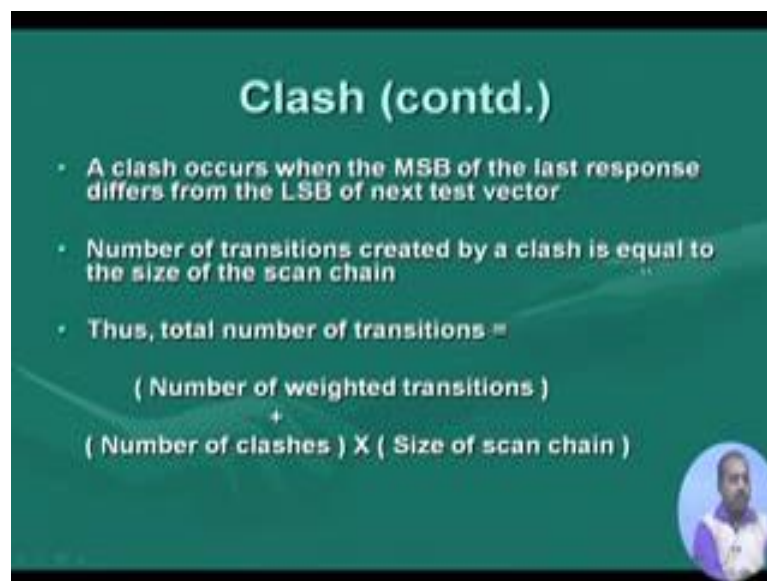


Another source of transition in the scan chain that comes due to this clash between the previous response and the next input test vector. So, it is like this supposed at present thus the last pattern was applied to the circuit and the response was such that the most significant bit of the response is 1, this bits we do not consider because there they will contribute to that weighted transition for the when the shifting out the response that has already been taken care of. But suppose this bit is 1 and the least significant bit of the next test vector is 0, as you know that this response shift out and this test pattern shifting they occur parallelly. So, and in the next shift cycle, this one will come to go to the next cell and this first beat of the test pattern. So, it will enter into the scan chain.

And in the next cycle what will happen this one we go to the next position next flip flop and this 0 will go to the next position. So, you see that there is a transition that is occurring here, there is a transition that is occurring here. So, this 1 to 0 transition will

affect the entire scan chain. So, here is a transition, here is a transition, here is a transition and here is a transition. So, this is a clash. So, if you consider, if it is the case that the MSB of the previous response is different from the LSB of the next test pattern. So, we will have a number of clashes generated and the number of clashes is equal to the size of the scan chain, length of the scan chain. So, that way this is the other source of transition that can occur when we are shifting in the test pattern.

(Refer Slide Time: 02:06)



**Clash (contd.)**

- A clash occurs when the MSB of the last response differs from the LSB of next test vector
- Number of transitions created by a clash is equal to the size of the scan chain
- Thus, total number of transitions =  
$$\begin{aligned} & \text{( Number of weighted transitions )} \\ & + \\ & \text{( Number of clashes ) } \times \text{ ( Size of scan chain )} \end{aligned}$$

The slide features a green background with white text. A small circular inset in the bottom right corner shows a person's head and shoulders.


So, clash occurs when the number, when the MSB of last response differs from the LSB of next test vector. Number of transitions created by a clash is equal to the size of the scan chain, so that you have already seen. So, total number of transition that are coming is equal to number of weighted transition plus number of clashes into size of scan chain because each clash will contribute to scan chain size of scan chain number of transition. So, if we sum them up will get the total number of transition given a set of test vectors and the corresponding responses without considering the circuit itself by just looking at the test vector and knowing the test vector and response and knowing the size of the scan chain we can compute what is the total number of transitions in the scan chain that is going to occur. So, we do not do any circuit simulation any logic simulation of the circuit we straight way do this competition with the knowledge that this parameter can be used to compare between 2 different modifications of the scan chain.

(Refer Slide Time: 03:18)

## Test Vector Ordering

- Can be applied to reduce inter-vector transitions, that is clashes
- Test vectors are grouped into four distinct categories based on first test vector bit and the last response bit

Group	Test vector bit	Response bit
I	0	0
II	0	1
III	1	0
IV	1	1



So, what are the approaches that we can build based on this particular knowledge about the transition. One thing is that test vector ordering, previously when we discussed about test vector ordering we assumed that from the test pattern from the ATE the test patterns are coming parallelly and they are getting applied to the circuit.

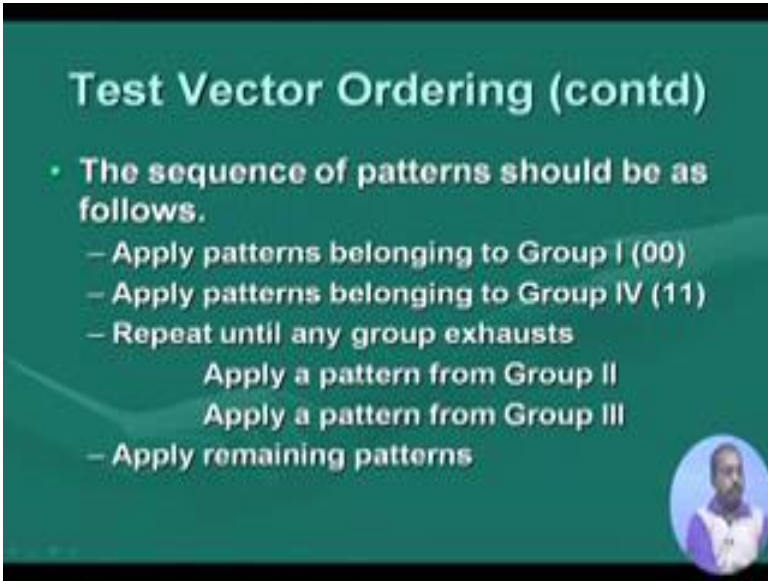
But in most of the cases or practical cases what happens is that the circuits will have scan chain and the test pattern will pass through the scan chain. So, scan transition minimization becomes a major issue. So, by this test vector ordering what we can do we can reduce the number of clashes that are going to occur how? Let us say that we group the test vectors into 4 groups I, II, III and IV, in group I we put those patterns whose test vector LSB and response bit MSB both are 0. So, it is LSB of test vector and MSB of response both are 0. In group II we put those patterns for which the LSB of test vector is 0, but MSB of responsibility is 1. Similarly group III we have got is test LSB bit 1 for test vector and MSB of response 0 and group for both are 1.

Now after you have done this division of the test vectors into 4 different groups we can frame a policy to reduce the clashes why? See this group one vectors and group 4 vectors say they do not create any clash between themselves, group 0 if all vectors belong to

group I then there are no clashes because MSB and LSB those bits are same. So, no clash will be generated.


Similarly, this group IV also there is no problem because both of them are 1. So, we can apply them anyway. So, there will be no clash that will be generated, but the problem occurs with group II and group III - 2 successive group II pattern. So, will generate a clash similarly 2 successive group III patterns will also generate a clash; however, there is a catch here. So, if you can alternate between group II and group III, first pattern we apply is from group II the next pattern we apply is from group III. So, there is no clash. So, this bit is one and this bit is also one as a result no clash is generated and then again we apply a pattern for group II. So, this previous response MSB was 0 and next test vector LSB is 0 again there is no clash.

(Refer Slide Time: 06:12)



**Test Vector Ordering (contd)**

- The sequence of patterns should be as follows.
  - Apply patterns belonging to Group I (00)
  - Apply patterns belonging to Group IV (11)
  - Repeat until any group exhausts
    - Apply a pattern from Group II
    - Apply a pattern from Group III
  - Apply remaining patterns



So, you see if you can alternate between this group II and group III patterns then again the same thing, we can avoid the clash is being generated. So, this way we can frame a very simple algorithm, for this test vector ordering under scan chain situation. So, first we apply patterns belonging to group I, then we apply patterns belonging to group IV, so this is done, then we repeat until any group exhaust apply a pattern from group II followed by apply a pattern from group III. So, this way we can go on alternately putting

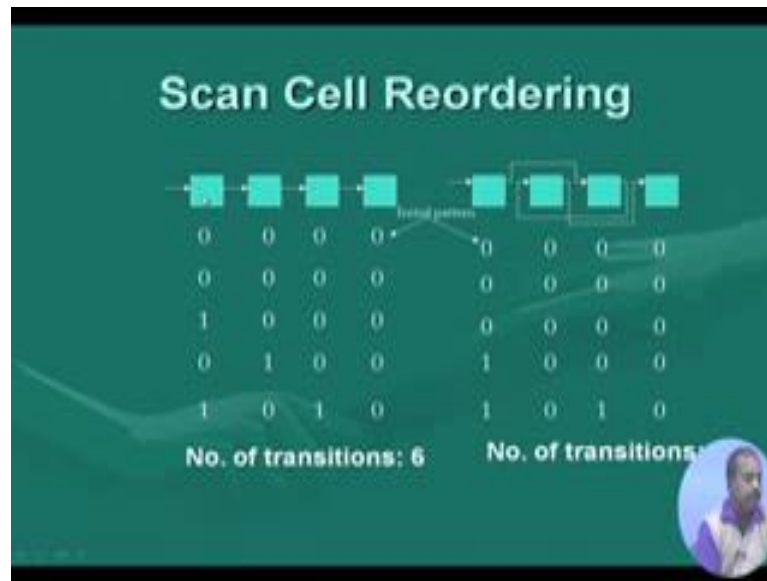
the patterns of course, you can see that this group IV is ending with 1. So, I should apply group 3 first and then group II because then this at one more clash can be avoided this one and one this clash can be avoided.

So, I think after placing group IV, this should be reversed. Apply pattern from group III followed by apply a pattern from group II that will save one more clash actually. But not very significant, but still at least some reduction will be there, and then after sometime one this group will exhaust, even alternating between group II and group III, it is not necessary that they are both the groups are equal number of test patterns in them. So, remaining patterns we have to apply. So, those clashes will be generated.

So, the clashes are getting generated only in the remaining patterns whereas, when the from group I to group IV transition one clash will be generated and another clash will be generated at remaining patterns that way there will be only a few very few clashes that will come.

So, within the within a test vector this weighted transition that we cannot avoid or within a response the transitions we cannot avoid because that anyway has to propagate through the scan chains. So, we cannot avoid those cases; however, for these clashes we can reduce and you see one clash contributes to size of chain number of transitions. So, if can say one clash that may be a significant savings in the number of transitions.

(Refer Slide Time: 08:13)



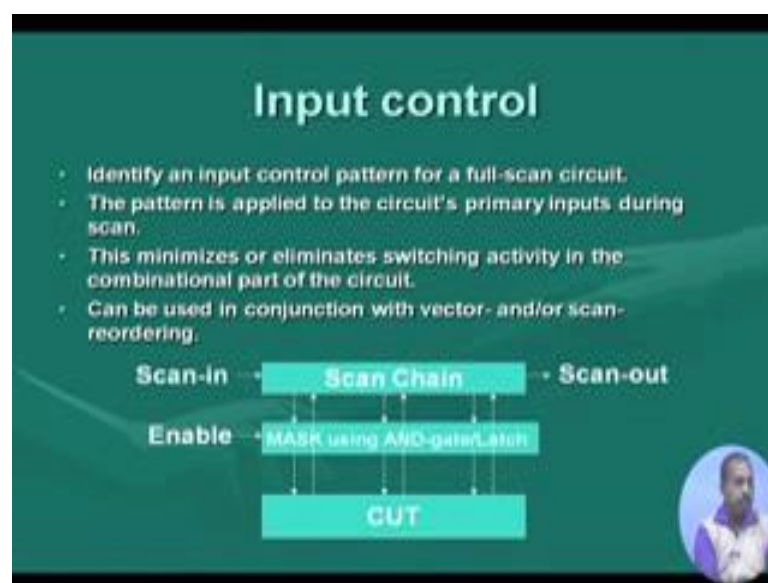
Another possibility of this transition reduction during scan shifting is by scan cell reordering. So, suppose we have got a for cell scan chain where these are the scan chain cells. Now we finally, want this particular test patterns, 1 0 1 0 to be shifted. So, initially it is these all flip flops are reset. So, this is in 0 0 0 then this 0 is shifted, it becomes this 0 comes here than this 1 is shifted. So, the 1 comes here and 0 goes to this point then again this 0 is shifted. So, 0 comes here 1 get shifted and this 0 also gets shifted and finally, this 1 comes into the scan chain. So, you see if you compute number of transitions. So, here there is 1 2 3 transitions 4 5 6 transition.

Now if I can change this scan stitching patterns, instead of connecting them in this sequence. So, if we connect them in a different fashion like this then you can just go through the same exercise and see that to get the same pattern 1 0 1 0 number of transitions needed is equal to 2; however, this is not very much very much practiced because what happens is that this scan stitching pattern is not always in the hand of the test engineer because it says that this scan flip flops that we have. So, there they may be distributed the flip flops may be distributed in my design at various places. So, if you stitch them in a different order then it may enter, it may require long wires, long interconnects may get integrated into the system. So, who is the designer may not permit because this delay of this line and they routing of this interconnection that become a

problem. So, normally this scan stitching is done in such a fashion that the closest flip flop so that gets connected to the next to the current flip flop that way the stitching is done. So, this is a bit problem.

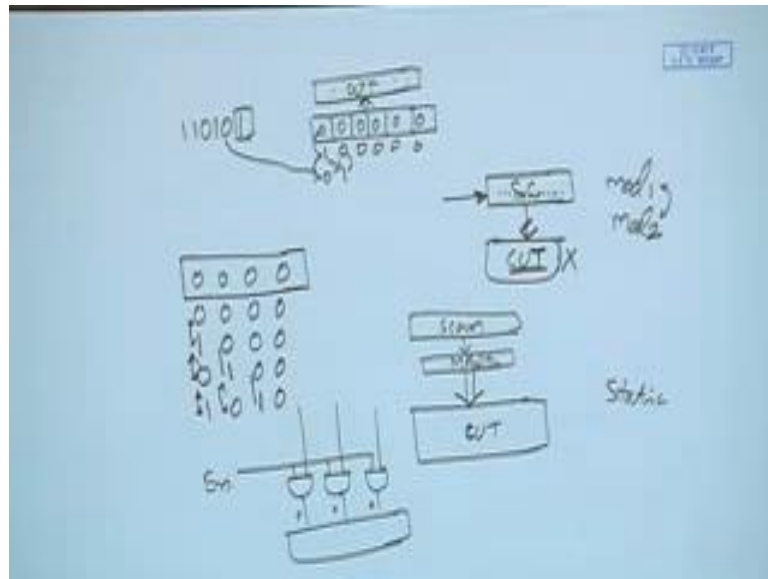
So, this scan cell reordering though it was initially advocated as one of the very promising technique, but now it is not followed much accepting some situation where we have got the freedom to do this stitching at the reordering.

(Refer Slide Time: 10:33)



Another possibility is to control the input. So, what happens is see I have said that when the tests patterns are going through this scan chain. So, the circuit is seeing those transitions, circuit is seeing those transitions as a result we have got this transition coming to the circuit, but if I can somehow do it like this that is we have got a mask in between and there is a enable signal. So, when this scan shifting is going on, with this mask will disable this inputs coming to this side, it may be a transmission gate based mask and gate based mask or latch gate based mask whatever. So, we do it, they actually do not allow this scan chain values to reach the circuit as a result the circuit will not see any transition. So, this is possible, and also it is the case that we can put this circuit into some low power mode in the sense that this is my scan chain, this is my circuit.

(Refer Slide Time: 11:38)

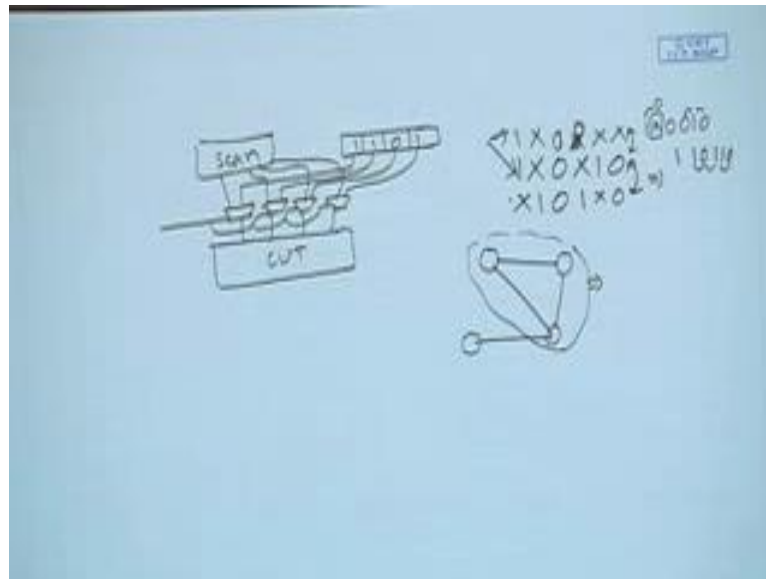


Now circuit as I have said that when it is idle when there is no activity at the input of the circuit, it is going to consume only the static power, it is going to consume only the static power. So, what I have done? I have put a mask here I have put a mask here and this is a scan transitions. So, this is the scan chain. So, this scan chain transitions are getting masked of here. So, they are not reaching the circuit. Now if I use an AND gate based mask. So, my mask is like this. So, these there are number of AND gates and these lines are coming from the scan chain and we have got this enable signal connected here. So, this is the enable and this is going to the circuit.

Now when this enable line is 0, I will see a 0 here. So, if the circuit says 0 0 0 0, but when it says 0 0 0 0. So, it will have some static power consumption because dynamically the values are not changed. So, dynamic power is not there, but static power will be there. Now this static power may not be minimum going the input pattern is 0 0 0 0. So, it may depend upon the circuit, we can have different input pattern that will reduce this static power. So, it is possible that we put the circuit input in some particular value. So, some input control pattern is there and that input control pattern will be applied to the circuit. So, this situation is modified.



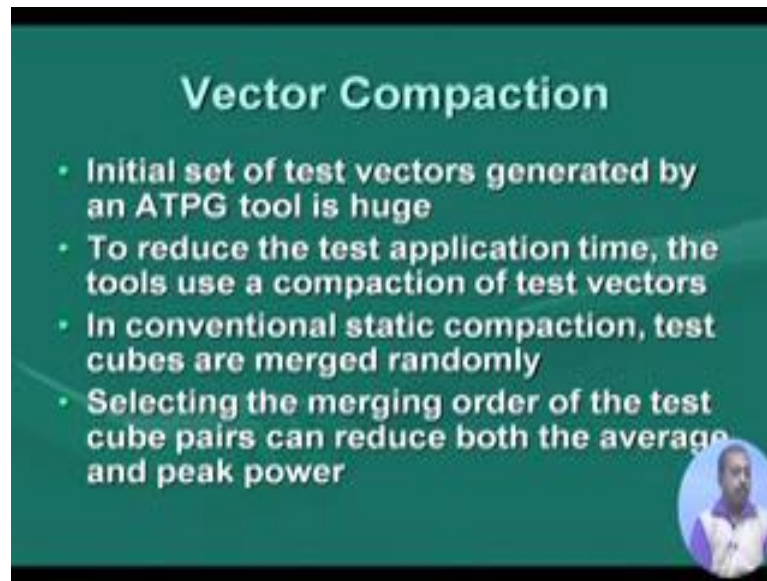
(Refer Slide Time: 13:13)



Like we have got this input control pattern that maybe 1 1 0 1 something like this, now this is my scan chain, now this is the circuit, we put some multiplexers here - 4 multiplexers, one input is coming from scan another input is coming from here like this we can do it like this and this enable line can be fed as the select line of this multiplexers. Now what happens? When the circuit is idle, by enable equal to 0 so the circuit will see 1 1 0 1 which is a pattern for which this leakage power consumption of the circuit is minimum, so this simulation can be done based on the type of gates that we have in the circuit. So, we can do a static simulation and find out like which what is the power consumption corresponding to individual gates for different input patterns and input pattern according we could select for input pattern for to work well for the entire circuit.

So, this will if we do this static power is reduced at the same time the switching activity gets eliminated. So, that there is very little switching activity this only switching activity will come when this mask come into existence. At that point there will be some changes, but otherwise not. And this outputs are of course, directly going to scan chain. So, if we are putting this capture mode when we put this in capture mode then this will be captured in to the scan chain. So, there that is not asked definitely. So, also we can use this in conjunction with vector and scan chain reordering, scan cell reordering techniques. So, that it does not go opposite to that, does not cause any harm to that.

(Refer Slide Time: 15:08)



### Vector Compaction

- Initial set of test vectors generated by an ATPG tool is huge
- To reduce the test application time, the tools use a compaction of test vectors
- In conventional static compaction, test cubes are merged randomly
- Selecting the merging order of the test cube pairs can reduce both the average and peak power

Another possibility of test power reduction is by vector compaction. So, vector compactions strategies we have already seen, but we can do it in the context of this test power reduction also. So, we have seen that this initial test vector generated by a ATPG tool is huge because it is initially targets one fault generates pattern of that then they need target some other fault generous pattern for that, as a result that test patterns is huge. After that it goes into a compaction of test vector stage. So, we have already seen that test what we do? If there are 2 test patterns one is like say 1 X 0 X 1 0 another is X 1 0 1 X 0 then these 2 can be merged into a pattern like say 1 1 0 1 1 0. So, it can be merged into a pattern like this. So, that it takes care of both the patterns. So, that is actually the compaction. So, vector compaction.

So, what we do, if we take each of this vectors as a node and each of this test vectors generated by the ATPG tool as a node and if 2 vectors are compactable we are connecting them by an edge. So, it may so happen that these 3 vectors are compatible. So, by click partitioning we identify this click and for this 3 of them we replace it by a single pattern so that technique we have seen.

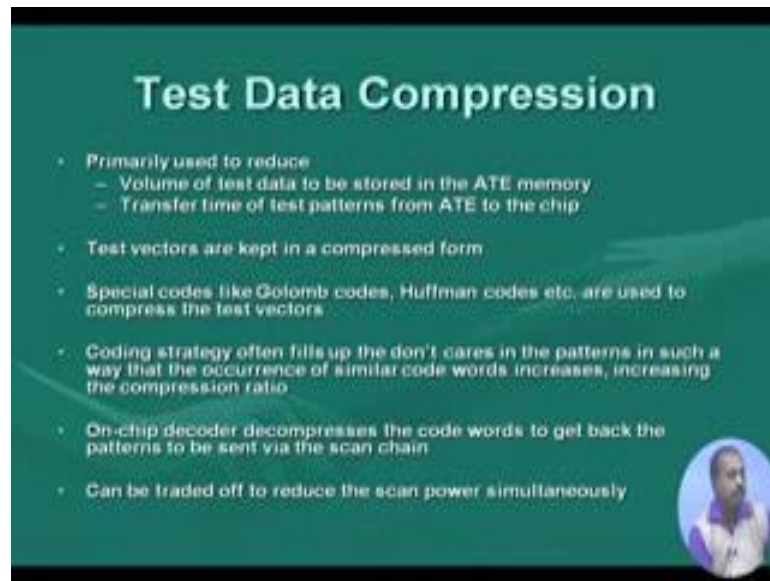
So, in conventional static compaction technique, this test cubes are merged randomly. So, randomly select this clicks and then it merges those patterns, but this merging order of

test cube pairs can reduce the average and peak powers, so both of them. How? Because it may so happen that after. So, there may be another pattern which is say this bit is 1, this bit is say 1 this is there, this is say X X, this is also X, now you see that this pattern is also. Let me make it 0 let us make it 0. So, this pattern is also compatible with this pattern, but this is not compatible with the third one. So, there if you construct this conflict graph compatibility graph, between these 2 nodes that there will be an edge.

Now I have a choice, I can merge this pattern with this pattern or I can merge this pattern with that pattern, now which one to do? Now if I merge this if I do this particular merging the pattern I am getting is 1 don't care then sorry this has to be 0, 1 do not care then 0, then 0, then 1, then 0 I get a pattern line like this. Now you see that if this do not care bit is still free in my hand, I can make it a 0. So, if I make it a 0 when it is going through a scan chain you see there is a transition here there is a transition here and there is a transition here. So, 3 transitions I can see where as in this case I have got a transition here I have got a transition here. So, I have got a transition here, in this case also I have got 3 transition, but in general what can happen is that these 2 merging they can give rise to different number of transitions in the scan chain.

So, which ever margin gives us less number of transitions. So, I should select that particular merging. Now there maybe choice there may be trade off because some merging may be readable combining a large number of vectors into one. So, that way the test length will reduce where as some merging may have very good patterns, where the number of transitions are very less as a result that will lead to low power pattern. So, we can have a choice. So, we can have a preceding merging order can be selected so that it can reduce this average and peak power consumption.

(Refer Slide Time: 19:05)



## Test Data Compression

- Primarily used to reduce
  - Volume of test data to be stored in the ATE memory
  - Transfer time of test patterns from ATE to the chip
- Test vectors are kept in a compressed form
- Special codes like Golomb codes, Huffman codes etc, are used to compress the test vectors
- Coding strategy often fills up the don't cares in the patterns in such a way that the occurrence of similar code words increases, increasing the compression ratio
- On-chip decoder decompresses the code words to get back the patterns to be sent via the scan chain
- Can be traded off to reduce the scan power simultaneously

So, another avenue for this power reduction is via test data compression technique. So, it is a test data compression we know that it is primarily used to reduce the test volume test data volume that we have to keep in the ATE memory and also to reduce the transfer time of test pattern from ATE to the chip. So, this we have already seen.

Test vectors are kept in a compressed form and we have seen some spec codes like Golomb code, Huffman code etcetera to compress the test vector. So, coding strategy what it does is that it takes help of this don't care. So, it spurge the do not cares such that the occurrence of similar code words increases and that increase that improves the compression ratio. So, this is the standard technique that we have seen that, any of this coding techniques they try to make use of the do not care bits in the test pattern. So, that the commonality between this cubes increase so that they have to store less number of cubes in the dictionary or the coding can be done. And on chip decoder, it will decompress the code word to get back the original patterns from the; and then it will be send via scan chain.

So, now we can have a trade off as soon as we do this. So, how much compression we do and how much transition it creates. So, some of it may be contradictory like 2 test patterns maybe they do not care it can be filled in 2 different ways - one way the test



(Refer Slide Time: 22:12)

Power Compression Trade-off in Golomb Coding		
Partially-specified scan vector	Fully-specified vector (Minimum WTM)	Fully-specified vector (Don't cares mapped to 0s)
01XXX10XXX01	011111000001 Golomb code length: 19 bits (m=4) WTM = 18	010001000001 Golomb code length: 10 bits (m=4) WTM = 26
01X1010XXXX1	011101011111 Golomb code length: 27 bits (m=4) WTM = 23	010101000001 Golomb code length: 13 bits (m=4) WTM = 32

Now, suppose I have got one test vector like this. So, this is that the ATPG tool it has generated a test vector like this 0 1, then 3 these are all don't cares 1 0 then again 3 do not cares then 0 1. Now if I want to reduce this weighted transition metric then what I should do? I should try to minimize the transitions that are occurring in between, since this is 0 I do not want transitions, here actually both the sides are 0, this is also 0 and this side is also 0. So, that immediately suggests that all this x they should be filled with 0. So, I get it like this similarly this group of xs. So, it is bounded by 2 1's, so they should be filled up with 1's. So, this is the test pattern that will get if we want to minimize the weighted transition metric.

And if we code this particular test vector we can see that this Golomb code length will be 19 bit. So, it is not shown here explicitly, but if you do the exercise like finding the runs of 0s and then coding it using the table that you have shown previously. So, the length will become 19 bit whereas, for if you compute the weighted transition metric for this pattern say it is 18, by using the formula that we have discussed previously the weighted transition matrix so that is equal to 18.

Now, what about if this do not care bits are filled in such a fashion that this compression is aided. So, for aiding the compression what I would like to do is I would like to since I

am this Golomb code is based on runs of 0s. So, I will try to increase the runs of 0s. So, this part is filled, the first part is field as we have done here, but the second part we feel with 0s because that will give me a run of 3 0s, it will give me a run of 3 0s and in this case if you do the Golomb coding, the code length will be 10 bits. So, code length reduces, but if you compute the weighted transition matrix for this particular case it will turn out to be a 25 because it will be shifting from this site. So, as a result it will be turning out to be 25.

On the other hand if say another example say this pattern then again the same thing if you fill it up for minimum weighted transition matrix. So, this will be the fill up and Golomb length code will be 27 bits and weighted transition metric value is 23 whereas, if you are filling with for compression then this code length can be 13 and transition metric value can be 32. So, that way there is a huge difference.

So, we have got the trade off like which pattern we chose for which purpose so that has to be traded off. So, it depends on the choice of the test engineer, it depends on the amount of memory we have, it depends on the power budget of the chip. So, depending on all those parameters, we can select the appropriate mechanism.

(Refer Slide Time: 25:18)

### Best Primary Input Change Time

- Each test vector  $V_i = \langle X_i, Y_i \rangle$ , where  $X_i$  is the primary input part and  $Y_i$  is the pseudo input fed via scan chain
- For  $m$  scan cells,  $Y_i$  is shifted in  $m$  clock cycles
- Primary input can be changed at any of these  $m$  cycles without affecting test efficiency
- When should the primary input be changed to minimize number of transitions in the circuit**

Primary Inputs, X

Scan Inputs, Y

Next one is when are we going to change the primary input. So, as I was telling that every test pattern. So, it has got a number of primary inputs and number of scan input the pseudo primary inputs and in a scan environment, this scan inputs are fed in a serial fashion to this scan chain. Now this, now when do we apply this test pattern? This test pattern we can go to the long chain capture cycle only when this entire shifting has been done, but this primary input part it can be changed at any time from the shifting of the first bit till the shifting of the last bit, at any point of time we can change this primary input part. So, what is the best input pattern? So, when should the primary input be changed to minimize the number of transitions in the circuit?

So, if there are  $m$  number of scan cells then this test this pseudo input primary input part why I shifted in  $m$  clock cycles and primary input can be changed at any of those  $m$  cycle, these  $m$  cycles without affecting the test efficiency. So, when are you going to change the primary, so that is the concept of best primary input change time and in fact, there is no straight cut guideline like when they should be changed, but this depends on the circuit to be very much. So, maybe we need to do a proper simulation of the circuit to see when this primary input is going to change, when this primary input should change so that we get the minimum power consumption. We continue in the next lecture.