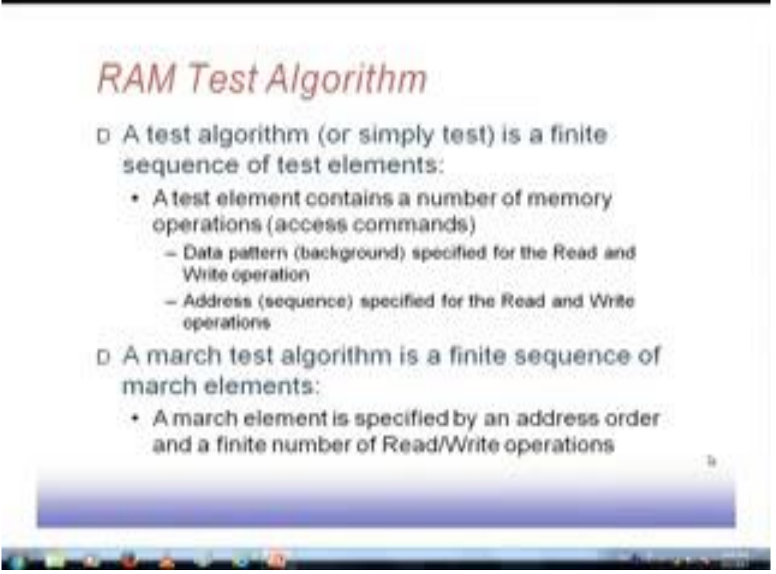
Digital VLSI Testing
Prof. Santanu Chattopadhyay
Department of Electronics and EC Engineering
Indian Institute of Technology, Kharagpur

Lecture – 58
Memory Testing (Contd.)

(Refer Slide Time: 00:25)



So, we have been talking about RAM test generation algorithms like how to for generating the test patterns for RAM, what should be the technique. And this test generation algorithm essentially it comes out with the set of the read-write sequences, in fact, for the testing a memory, ultimately we need to write some bit pattern there and check whether the bit pattern has been correctly written there or not by doing a read operation. And similarly if we want to modify one location, so it should not be the case that some other location also gets modified due to that writing. So, basically we need to do some read-write operation over the memory cells to check that the memory cells are working properly or not.

So, when we talk about RAM test algorithm, so that is basically talking about is sequence of test pattern test element. So, sequence of patterns to be applied for testing the particular memory. So, a test algorithm or simply test is a finite sequence of test elements, where each element has got a number of memory operations. So, memory operations are read and write operations. So, it may be operation like read followed by it

may be the operation like write followed by a read. So, if you write a 0, and then we would expect that while we are reading we will get a 0, so that write zero followed by read zero may be testing a cell for possible fault of stuck-at-one. So, like that we can have a set of these patterns to be written. So, it contains a number of operations, so that is why it is called access commands or operations to memory operations to be done. So, while doing this, so this memory commands, so they will create some solid some background pattern we earned for the memory. So, while we are writing some pattern, so it will create some data pattern through this write and read operation that will take place, so that is one part.

And the second thing is that we have written say at cell number ten then where are you going to do the next memory operation, is it at cell number 9 or at cell number 11 that is whether we are going to apply the same the next read-write command on the next memory cell or the previous memory cell. So, we are traversing the memory in an increasing address order or a decreasing address order, so that actually the address sequence specified for the read-write operation. So, while we are talking about a test element, we are talking about these two things what are the commands to be executed and the sequence of addresses of the cells in which this commands have to be executed.

So, the most popular memory test algorithms, so they are known as march test algorithms, because they talk about some marching pattern through the memory cells. So, we write some pattern and read it. So, it is a marching pattern. So, it has got a finite sequence of march element. So, march elements are applied and that way it will test the faults that may have cropped up into the memory array. So, a march element is naturally specified by an address order and a finite number of read-write operations.

(Refer Slide Time: 03:46)

March Test Notation

- $\Uparrow$: address sequence is in the ascending order
- $\Downarrow$: address changes in the descending order
- $\Updownarrow$: address sequence is either $\Uparrow$ or $\Downarrow$
- r: the Read operation
  - Reading an expected 0 from a cell (r0); reading an expected 1 from a cell (r1)
- w: the Write operation
  - Writing a 0 into a cell (w0); writing a 1 into a cell (w1)
- Example (MATS+): $\{\Updownarrow(w0); \Uparrow(r0,w1); \Downarrow(r1,w0)\}$

So, march test notations that will be using if these are very standard notations like say this up arrow coming before the march element, so it will identify that the operations specified in the element should be applied to the memory cells in an ascending order. So, address will be in ascending order first it will be say writing at cell zero, then cell one, cell two that way the address will be in ascending order. Or it may be the down symbol. So, that is the address changes in the descending order, so while we are applying the patterns to the memory array. So, for testing it so address should start with the highest possible address and then the address would go down after the read-write operations at individual cells.

And the third one is either way. So, this here actually this means that the element can be applied in any order. It does not matter whether it goes in the upward memory address increasing order or in decreasing order that does not matter. So, test coverage and everything will remain unaltered. And as for as the operations are concerned, so as we know the memory the operations are a read operations, so it may be represented by r and or it may be a write operation, so will represented by w. So, while we are writing, so we can write a zero to the cell or we can write a one to the cell, so this w 0 or w 1.

Similarly, while we are reading from the memory cell, so we expect that we may expect to read zero or we may expect to read a one. For example, if the just previous command was written as zero to the cell then the next command should be trying to read the value

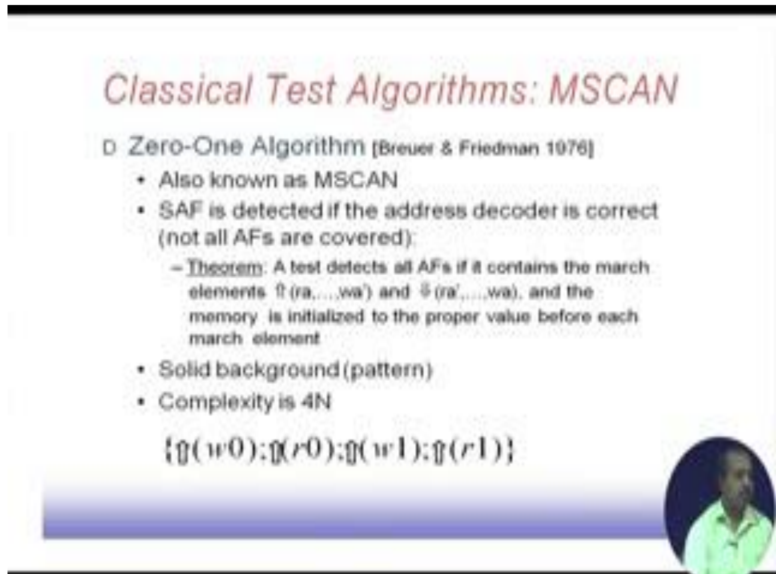
from the cell and the expected value is zero. So, if you get the value read from the cell as one, so you understand that there is some problem with the cell. So, there may be some stuck-at-fault, so that way we can identify the faulty situation.

So, this reading is the expected value that we expect to read is written after this after r, and similarly the value that we are going to write to the cell is written after the letter w. So, for if this is MATS plus is a march test pattern, so this has got three march elements. So, this is the first one this is the second one and this is the third one. It says that the first one we can be applied in any order, so either increasing or decreasing order, and it will fill up the entire memory with zero because it says that the write zero, say all the cells they will contain now 0.

The next, so the first template will be for the entire all the cells this first template will be applied. So, all the cells are now containing zero. Then it will come to the next templates. So, here it say that do it in the ascending order, first you read the content of the cell and expect the value read to be 0. Naturally we written a 0, so we expect to read a 0. And if it is not correct, then there you can detect the errors that stuck at one type of error and then we change the pattern to 1 for that cell, so we have read a 0 and then we are trying to write a 1. And similarly, when this template is over that means, for all the cells they are now having the value 1. So, next we do it in the decreasing memory order starting with the highest memory address.

So, since we are written a one at the last place. So, we expect to read a 1. So, we read a 1 and then we write a 0, so that again fills up the location with 0. So, this is one particular march testing sequence. So, these are got three march elements in it. So, this will detect different types of errors, like it can detect some sort of coupling between the cells also, like you see that say suppose there is a coupling between two cells, so for the first cell which came earlier, so they are while we are doing this write operation the second cell also got modified. So, it was 0, but it become 1. So, when this r 0 will be done on the second cell there will be a failure because it will the system will find that the cell is containing 1, so that type of coupling can also be detected. So, this way you can analyze what are the faults that can be detected, if we follow this type of read-write pattern.

(Refer Slide Time: 08:08)



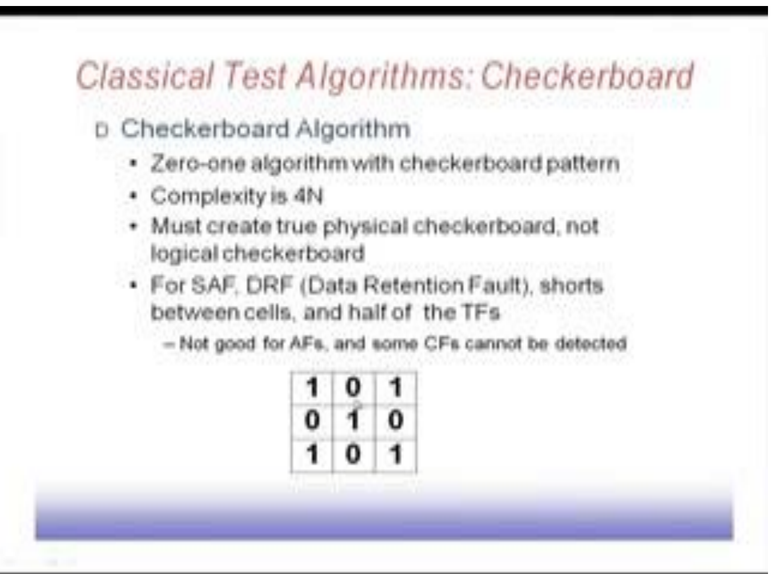
There are many such march algorithms; the first one is known as the MSCAN algorithm or memory scan type of algorithm, as because it scans the memory from one end to the other end. So, this is also known as the 0, 1 algorithm. And proposed long back, so it is 1976, it has been proposed, so long back. So, stuck-at-fault is detected if the address decoder is correct. So, this is the proper it can be shown that if your address decoder is correct then this all stuck-at-faults can be detected. So, you see that this ultimately the pattern is like this, so in an increasing order you write zero then in a increasing order you expect to read a 0 value, then in a increasing order, you write a 1, and in increasing order you write you read a 1. So, you see that if there is no address decoder fault, then this part will detect stuck-at-one fault and this part will detect stuck-at-zero fault, because it is a expected to you have written one and expect one. So, if that is remaining zero; that means, stuck at zero similarly this one.

So, if the address decoder has some problem then of course, some other location may get selected. So, you cannot be sure whether it is a fault stuck at fault or it is an address decoder fault, but assuming that address decoder ok, so we can detect all the stuck at faults. So, there is a theorem that says that a test will detect the address faults if it contains the march elements in this way, the increasing order r a and write a dash. So, a and a dash, so they are actually complimented it each other. So, if a is 0, and then a dash will be 1, so like that and vice versa; if a is 1, a dash will 0, so vice versa.

So, if your set of march elements contains these two march element that one march element starting with r a ending with write a dash read a and write a dash, and another march element coming in the descending order starting with read a dash and ending with write a. So, this will be able to detect all address decoder faults. So, this particular, so that actually tells that all address decoder faults are not covered by this particular this MSCAN or zero-one algorithm. So, it is a solid background pattern because the background is at any point of time if one template is over, so the memory background is having all zeros or all ones, so that is why it is called a solid background pattern.

And complexity is 4 into N, so N is the number of cells that we have in the memory. So, there are, so how many such read-write operation or how many memory accesses we are doing, so that data determines the complexity of this any this march algorithm. So, you see that there are four such operations to be done on every cell, so that is why (Refer Time: 11:01) complexity is 4 into N.

(Refer Slide Time: 11:05)



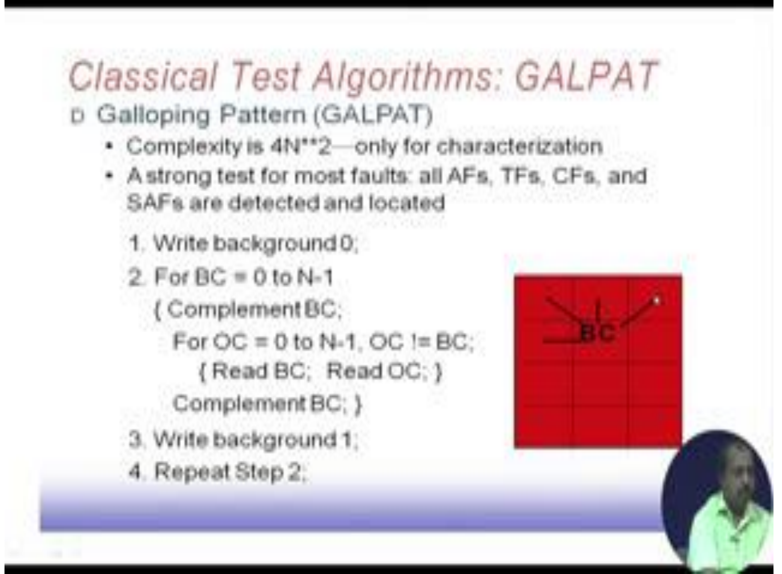
Another algorithm, which is known as checkerboard algorithm, so checkerboard algorithm, so this is again zero-one algorithm with checkerboard pattern. So, checkerboard pattern means that at any point of time when we have complicate one particular template of this march sequence, the pattern in the memory should be like this, say alternative locations we left ones and zeros. So, again this is complexity is 4 into N.

So, this point is important it says that must create true physical checkerboard not logical checkerboard.

So, what it is essentially means is that see if we have got say a memory of say 1024 locations. So, 1024 you can if you are breaking in a 2D forms, so you can break it in different ways. Now, while we are doing it, so it is not mandatory that a particular cell i and the cell i plus 1, which are logically close to each other will also be physically close to each other. So, they may be far away in the actual layout. So, what it says is in the actual layout this checkerboard has to be created. So, it is not a logical checkerboard, but a physical checkerboard that we have to do.

So, this type of checkerboard algorithm, so they are good for stuck at fault and data retention fault. So, data retention fault means we write some value there and it is expected that the value will be retained and after sometimes we again do a read operation. So, we find that the value has been changed. So, it is normally detected by two or three successive read operations. So, that way if the data is leaking if the cell is leaking then the data will get lost, so that is a data retention fault. Then we have got shorts between the cells and half of the transition faults, so they will be covered. But not good for address faults and not how the coupling faults are also not detected by this checkerboard type of algorithm. Complexity is 4 into N.

(Refer Slide Time: 13:08)



Another very good algorithm for this testing memory testing is known as GALPAT or galloping pattern algorithm. So, this diagram actually tells what it essentially tries to do. So, it says that I have got say total N number of cells, so out of that this BC is the cell that we are considering now. So, what we do we first write background to be all 0. So, first entire array is filled with 0, then we take any of the cells. Suppose, we have taken a say this particular cell, so we compliment this cell value to 1. Then we read the now we consider all the cells all other cells in the array. So, we take up this cell, we read the content of BC and then again read the content of the cell the OC cell, OC are basically other cells. So, OC you can read it as other cells, and BC is the cell that we are considering now.

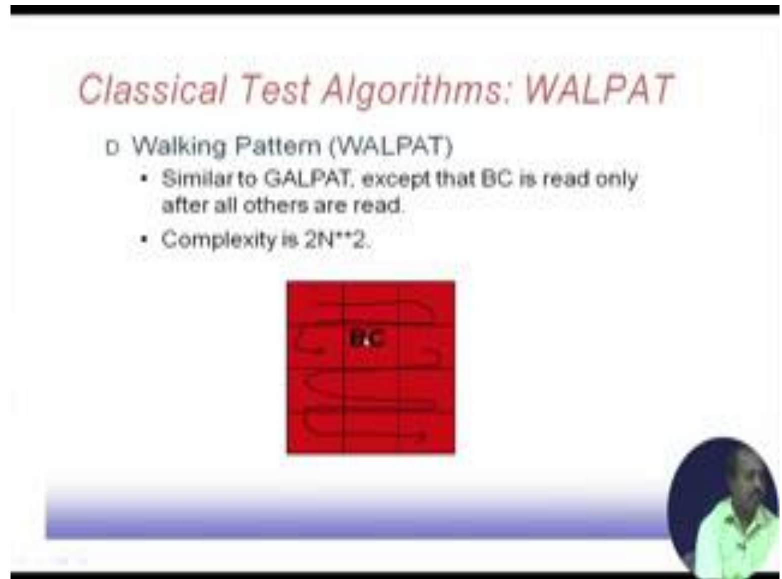
So, what we have done, we have complemented the value of BC then we read the value of then we have read the value of BC here, and then again read the value of OC here then again I am reading the value of BC here. I am reading the value of this particular cell then again then again reading value of BC and again reading value of these cells. So, this way actually it after finishing or reading all the bits in this all this bits again, we compliment the BC and again repeat the whole process. So, again it is for the next cell it will be done. So, you see after complimenting this is that is only for one cell then again we try with the next cell. So, this way it is for every cell, we are reading all other cell contents, so that way the algorithm is very costly. Then that is way to background zero.

And after that we make all entire background one and again do the same thing. So, this way it can cover most of the faults. So, it is a very strong test and it covers all address decoder faults, all transition faults, all coupling faults, and stuck-at-faults. So, stuck-at-fault, in fact, you can locate specifically why exact because stuck at fault has occurred. So, that way it is a very strong testing. So, it can detect large number of faults now different types of faults, but the complexities vary. So, it is 4 N square.

So, in some lecture, we have seen that seems that value of N is very large, so some algorithm which is more than linear time consuming one, so that is consider to be very costly. So, if 4 N square is a really costly algorithm, so that is why it is say it is a only used for characterization. So, if you have to do some sort of bench marking of some memory test algorithm, so you can compare it with GALPAT. For some small memory you can try doing GALPAT and your proposed algorithm and try to see which how close

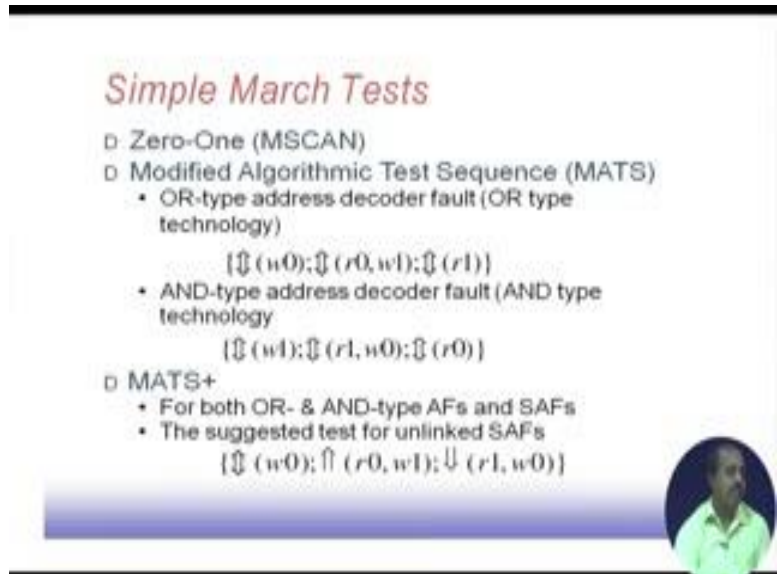
you are in detecting the fault that are detected by GALPAT as well, so that is why it is only use for characterization.

(Refer Slide Time: 16:26)



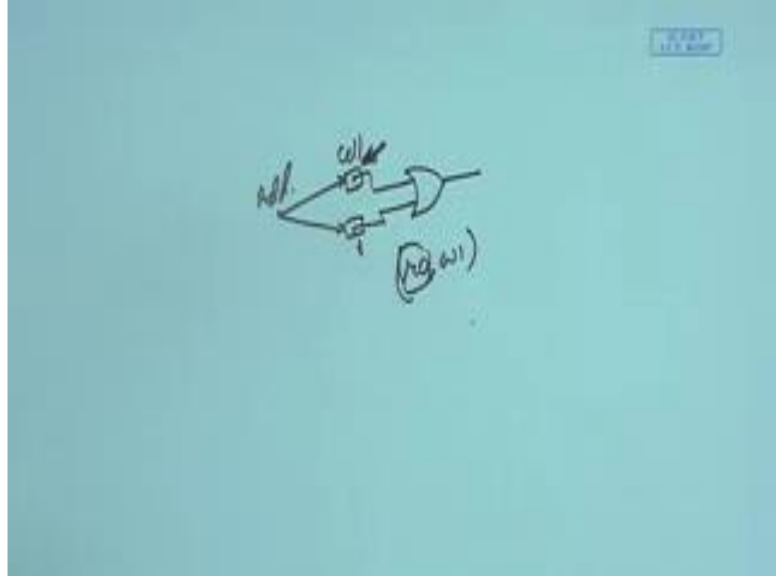
Another variant of this GALPAT is known as the WALPAT algorithm or walking pattern algorithm. So, what it says it is similar to GALPAT, but expect that the BC base cell that we have consider that is read only after all others are read. So, we complement this that is fine, but then we read all these cells. So, in case of GALPAT, what was happening that after reading every cell, so we are again reading the base cell, so that was repeated every time. But now it is not that in WALPAT, so we once we are reading this cell, so we read all these cells which are excluding BC, and then again we read BC and again we repeat the process, compliment the BC and again read the repeat the process. So, that way this WALPAT algorithm complexity is less compared to GALPAT. So, it is 2 N square, but still it is in the N square domain, so that is why it is still a very costly algorithm.

(Refer Slide Time: 17:25)



Now, simple tests, so this is like zero-one that we have already discussed the MSCAN algorithm. So, there are some modified arithmetic test sequence or MATS, so in a short form it is called mats. So, it can detect address decoder faults how, so say this one say this can detect OR-type address decoder fault. So, OR-type address decoder fault means for technologies where the result is OR-type. So, what we mean is that suppose one address selects two different cells, so this cell as well as this cell due to the problem with the address decoder. So, these are the memory cells and this is the address.

(Refer Slide Time: 18:05)



So, both the cells are selected. Now, what will be the output because ultimately RAM output is coming via that sends amplifier and all those things. So, in OR-type technology it says that the result is basically an OR of these two, the result that you get is OR of these two. So, for this type of technology, so this particular the sequence that is shown here in this MATS sequence, so first you initialize the entire array entire mode with zero write zero then we then again we start then we check individual memory locations for a read a value zero and then we write a one. And then after that we expect that we will when you read will get a one.

Now, you see in OR-type of situation, what will happen when say this cell was accessed from this cell was accessed in the second in the in this cell was accessed in the second template that is r 0 w 1. So, while was write operation here maybe this cell also got modified, so this cell also become one. So, as a result this r 0 w 1, this particular template, when this r 0 is done on this cells, so it will fail because it is already one. So, for OR-type of technology, so this will be in determined, for OR-type of technology, so this will be detecting the fault that is decoder fault.

Similarly, AND-type of technology means the content of two cells getting selected, the content should be ended to know what is the final output. So, for that that type of situation, so this particular sequence will be doing the thing, so it is r 1. So, first you write all one then you read one and write 0. So, AND it will modify the second cell also to 0, when you do write 0 on the first cell, so it will also modify the other selected cell as well, so that will also become 0. So, this read one will fail in that case.

So, we see that for OR-type and AND type this MATS algorithm, so it can detect the address faults. Now, modification of this MATS is MATS plus, so it says that it will, so it will be determine both OR and AND type faults. So, you see that this one r 0 w 1, so this will be determining OR-type of faults and this r 1 w 0 will determining the AND type of fault. So, both are detected by the MATS plus sequence.

(Refer Slide Time: 20:53)

March Tests: Marching-1/0

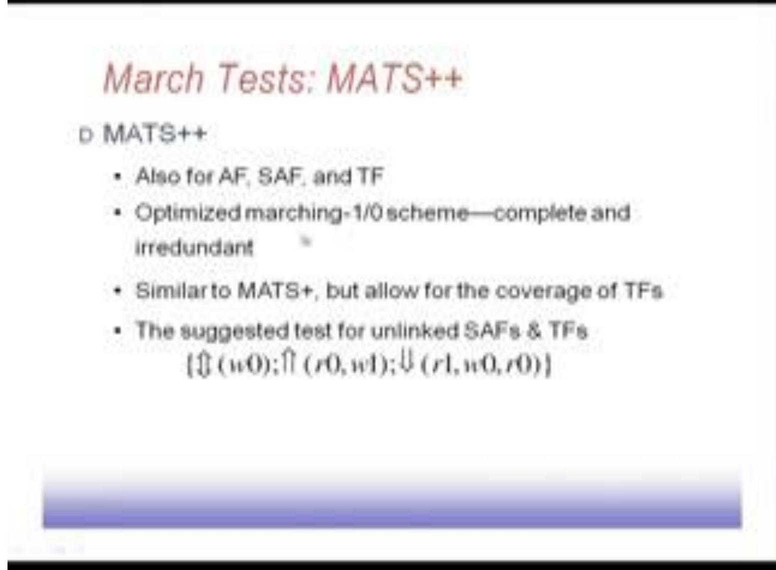
- Marching-1/0
  - Marching-1: begins by writing a background of 0s, then read and write back complement values (and read again to verify) for all cells (from cell 0 to n-1, and then from cell n-1 to 0), in 7N time
  - Marching-0: follows exactly the same pattern, with the data reversed
  - For AF, SAF, and TF (but only part of the CFs)
  - It is a *complete test*, i.e., all faults that should be detected are covered
  - It however is a *redundant test*, because only the first three march elements are necessary

$$\{\Uparrow(w0); \Uparrow(r0, w1, r1); \Downarrow(r1, w0, r0);$$
$$\Uparrow(w1); \Uparrow(r1, w0, r0); \Downarrow(r0, w1, r1)\}$$

Then we looking into marching one-zero pattern. So, this is a another march test marching one-zero. So, marching one it begins by writing a background of zero, and then read and write back complement values and read again to verify. So, for all the cells, so this is basically this one. So, first it will write background of zero. So, this write w 0. So, this will do that then it will say read and write back complement values. So, I am reading the value writing the complement value and then we are again reading it to verify. So, and reading to verify, so and read again to verify, so this read one again to verify. So, this will be this is done here. So, this is for zero and then I have to do it for one. So, again I am doing a read one write zero and read zero. So, this is one sequence marching again marching one. Similarly, I can having marching zero, so this will have a marching zero sort of thing.

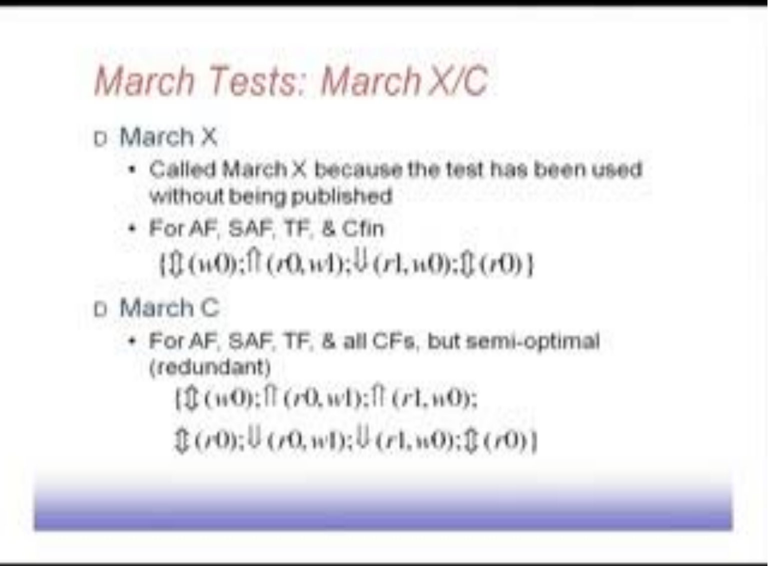
So, here this particular sequence, it can detect address decoder for stuck-at-fault and transition faults, but only a part of the coupling faults. It is a complete test. So, this is important it says about all faults that should be detected or covered. So, all the detectable faults, all the address decoder faults, stuck-at-fault and transition faults, they are covered by this one. However, so redundant test because only the first three march elements are necessary the rest part though it is telling marching zero, but that is not necessary. So, first part is sufficient for doing this. So, this is redundancy tools and all that we have skipping. So, it is there, so if you refer to the corresponding paper, so you can find it.

(Refer Slide Time: 22:47)



Then we have got MATS plus plus. So, MATS plus plus it is good for addressing address decoder fault stuck at fault transition fault and all that. Optimized marching of one-zero sequence, it is complete and irredundant, so this is the MATS plus plus sequence. So, first it writes zero, then it reads zero and writes one, then reads one write zero and read zero. So, again it can detect all this stuck-at-faults and transition faults.

(Refer Slide Time: 23:20)



Then we have got march X algorithm and march C algorithm. So, these are the two other march algorithms. Actually what happened is from the inception of this memory when

designer started doing this manufacturing the memory, the memory testing is always there. So, before this march algorithm came into market, so this researchers came up with this proper concept of march element or march test memory testing was being done. So, some of the march test, so they were used even before they are actually available in the literature.

So, march X is a sequence is a test testing a sequence that is useful that was used before this before it was reported in the literature. So, it can detect address decoder faults, stuck-at-fault, transition fault and coupling fault, so all of them can be detected and then you can do it in a, so this is the sequence. So, first it write zero to the background then it does read zero, write one; then it does read one, write zero; and then it again reads this whether the zero has been written properly or not, so that is done. So, this can detect all these types of faults address fault, stuck-at-fault, transition fault and coupling fault.

Then there is a march c algorithm. So, march c algorithm, so it the sequence is proposed is like this. First write zero, then read zero write one, then read one and write zero, then read zero write one etcetera. So, this is again good for address faults stuck-at-fault transition fault coupling fault, but it is semi optimal because some of this operations can be removed. So, they are it is it is there is redundancy. So, some of the operations can be removed.

(Refer Slide Time: 25:19)

So, if it is on the march c, if we remove this redundancy part then what we get is a march c minus. So, in march c we have got this sequence; in march c minus, some of these redundant parts are removed. So, this is the march c minus proposal. Actually all these are test algorithm. So, different people at different points of time have come up with this sequence this test algorithms, so that is why we are reporting all of them. So, march c minus is reported to be doing like this. So, write zero then read zero write one, then read one write zero, read zero write one so like that it is done. And then extension of march c minus it covers this stuck-open-faults in addition to this faults like this address faults stuck-at-faults and all that. So, a stuck-open-fault is also covered. So, this is the sequence fault.

So, to get a confidence that it really determines these errors, so we have to take individual errors and go through this individual sequences to locate where exactly which particular pattern is going to catch it. So, here we are just taking it as accepted that they are doing this faults are getting detected. So, the actual analysis, you can find in the corresponding literature.