VLSI Design Flow: RTL to GDS

Dr. Sneh Saurabh

Department of Electronics and Communication Engineering
IIIT-Delhi

Lecture 42
Automatic Test Pattern Generation (ATPG)

Hello everybody, welcome to the course VLSI Design Flow RTLto GDS. This is the thirty-third lecture. In this lecture we will be looking at Automatic Test Pattern Generation or ATPG. Specifically in this lecture, first we will be looking at some terminologies which are relevant to Automatic Test Pattern Generation. Then we will be looking at the general approach to ATPG and then we will be looking at the concept of redundant faults. So before moving to the ATPG method, let us look at an interesting paragraph from the book Through the Looking Glass by Lewis Caron.

So the paragraph is, they always purr. So here they refer to the kitten. They always purr if they would only purr for yes and mu for no or any rule of that sort she had said so that one could keep up a conversation. But how can you talk with a person if they always say the same thing? On this occasion the kitten only purred and it was impossible to guess whether it meant yes or no.

So this is true for the kitten and this is also true for our chip. Now if our chip produces the same output for defective chip and a fault free chip then we cannot distinguish whether our chip is defective or not. So to make our conversation with a chip reasonable we use a test pattern. So in the earlier lecture we have seen that test patterns are basically the inputs that we apply to our chip so that the output that is produced by the chip is different for a faulty circuit and a fault free circuit. Now let us look into how we can derive those test patterns and then use that for the purpose of testing.

Now what is the objective of automatic test pattern generator? So the objective is to generate test patterns and test patterns we understand that are the input patterns which produce different outputs for a faulty circuit and a fault free circuit. So these test patterns are also known as test vectors. So the objective of ATPG is to generate test patterns that can be used to detect faults in our circuit and once we detect a fault then we stop that chip to go to the consumer or the customer. So that is what our objective of testing is to basically find out or detect whether our circuit is faulty or not and then stop a faulty circuit to reach the end user. So what should be the desired characteristics of the test vectors that the ATPG tool generates? So the desired characteristic is that the set of test vectors that will be produced by an ATPG tool should be as small as possible.

So the test vectors that detect all faults considered for the circuit now for detecting faults in our circuit or all the faults in our circuit only need a set of test vectors. Only a few test vectors may not be sufficient but a set of test vectors will be sufficient to detect all the faults in our circuit. Now what should be the characteristics of that set of test vectors? We want that set to be as small as possible. Why do we want to have that set to be as small as possible? Because it will be reducing the cost of testing. Note that once we have a set of test vectors we need to apply those test vectors using automatic test equipment to our chip.

And once we apply those test patterns then we can detect whether the circuit is faulty or not. Now if the set of test vectors is large there are too many test vectors for example say there are 10,000. Then in that case the time required to apply those test vectors during testing will be very high. And therefore we want to reduce that set. If we reduce the set of test vectors then the application or the time to apply the test vectors during testing will be less and therefore the cost of testing will reduce.

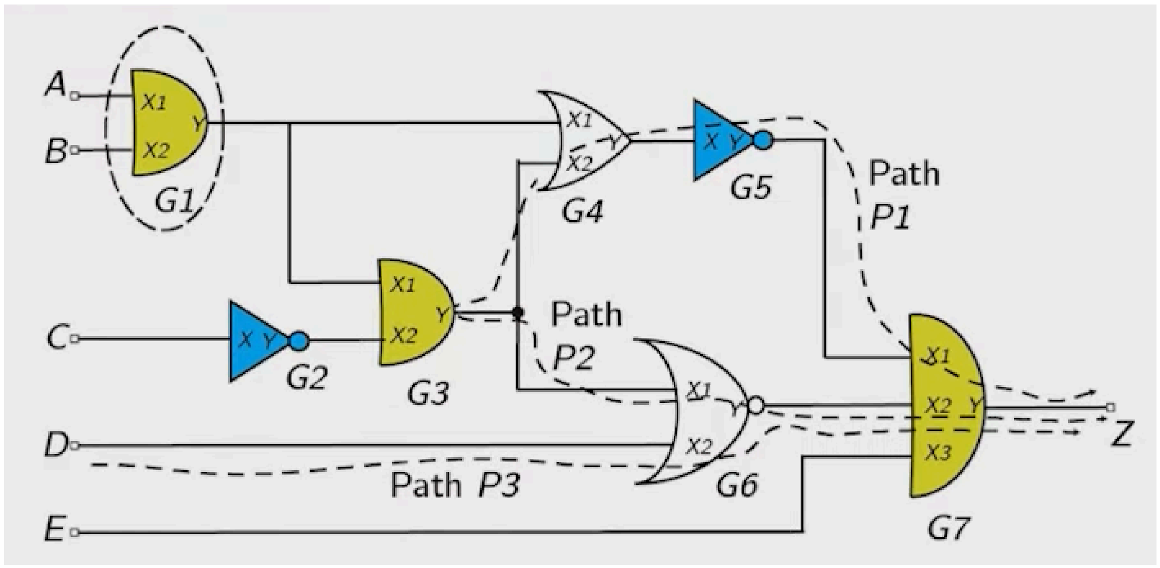
So the desired characteristics of the set of test vectors is that it should be as small as possible. Now if we consider a generalized sequential circuit for a generalized sequential circuit generating test vectors or sequences is very difficult. Why is it difficult? Because to apply the test vectors to our say combinational circuit in our dis in a sequential combinational circuit components inside our in a sequential circuit we need to take it to some certain states. So if we can, a generalized form of sequential circuit can be viewed as an FSM in which there are lots of states. Now to apply a particular test pattern to a combinational circuit element we have seen in the last lecture that we need to go through a set of states.

And so reaching certain states may be exponential and therefore it may take a lot of time during testing and also deriving those test vectors for a generalized sequential circuit is more difficult. Why is it difficult? It is difficult because the ATPG tool needs to explore the set of states. So it will require state exploration and we have seen in earlier lectures that the number of states in a circuit can be exponential in the number of state elements and it may be computationally very challenging to explore or to do state space exploration. So for a generalized sequential circuit finding a test sequence of test vectors is challenging. So how do we tackle this problem? So in the last lecture we have seen that scan design flow and we have seen that in scan design flow what we do is that we make some changes in our circuit and by making changes in our circuit we get a set of we get scan chains or the sequential circuit elements or flip flops are connected in form of a shift register in the scan mode.

When that happens then we can apply or we can get go to any number any state in linear number of clock cycles and therefore the flip flops the Q pins of the flip flops can be considered as pseudo primary input that is the Q pins of the flip flop and the D pins of the flip flops can be considered as pseudo primary output. So we have seen that we can consider the D pins and the Q pins as pseudo primary output and pseudo primary input in the last lecture for a scan design design scan inserted design. Now if that is the case then we can actually treat the need to find a set of test vectors only for combinational logic. We can treat the sequential circuit elements like flip flops that are similar to input ports. So we can assume that we need to find test vectors only for a given logic of combinational circuit element and the inputs to this logic inputs to this combinational logic will be either the primary inputs the PIs the primary inputs or input ports of our design or it can be the Q pins of the sequential circuit elements which acts as pseudo primary input and this is our combinational circuit element.

So the problem in a scan design flow sequence is that the finding test pattern boils down to finding test patterns for combinational logic elements. Why because we can consider that thus the sequential circuit element the flip flops have been converted to scan cells and the value that we want at the Q pins of the flip flops those can be directly be fed through the scan input port in a scan design flow and we can consider the Q pins of the flip flops similar as the input port of our design and proceed with finding the test pattern for the combinational logic code. And then we will later on when we want say the other test pattern for this it comes turns out to be say 0 1 0 1 then the input the values that are coming directly from the primary inputs or PIs we can apply that value at the primary input port and the values which are being driven or the pins of the or the inputs of this combinational logic to own is being driven by the by the Q pins of the flip flops those can be directly loaded using the scan input port while test. So the sequential ATPG problem gets converted to combinational ATPG problem in scan design flow and combinational ATPG problem though it is NP complete it is relatively much simpler than the sequential ATPG problem and there are many efficient algorithms which can solve the computational ATPG problem. So in this lecture we will be looking only at combinational ATPG problems and we will confine ourselves to the single stuck at fault model that we had discussed in the last to last lecture.

Now before proceeding to the ATPG method or how to find the test pattern let us be familiar with some terminologies which will be required in understanding the algorithm. So let us assume that this is our combinational circuit given that A, B, C, D, E can be either the primary input ports or it can be pseudo primary input ports meaning that these can be the Q pins of the Q pins of the flip flops. Now in this combinational circuit we can define a path. Now what is a path? Path is a sequence of pins in topological order . For example we can say that this is a path we call P1 .



Now how can we represent it? We can represent P1 as a sequence of pins in topological order. So this path starts at the pin Y so we say that G1 sorry G3 the name of the instance is G3 . So G3 and the name of the pin is Y so G3 Y comma then the second pin is G4 X2. So G4 X2 then G4 Y then G5 X this pin G5 X then G5 Y then it goes to G7 X1 G7 X1 then G7 Y and finally to Z this is the primary output of Po Z . So this is a path . Here we have listed the pins in topological order G3 Y G4 X2 and so on.

Similarly there is another path P2 we can list the pins for the same then there is a third path which has been shown . There are many other paths in our combinational circuit combinational logic and we are not showing them . So there can be so many pins that the path is a sequence of pins in topological order. Now what is on path input or on input? So for a given path the pins lying on that path the input pins lying on that path those are known as on input. So if we take the path P1 if we are taking the path P1 this is the path P1.

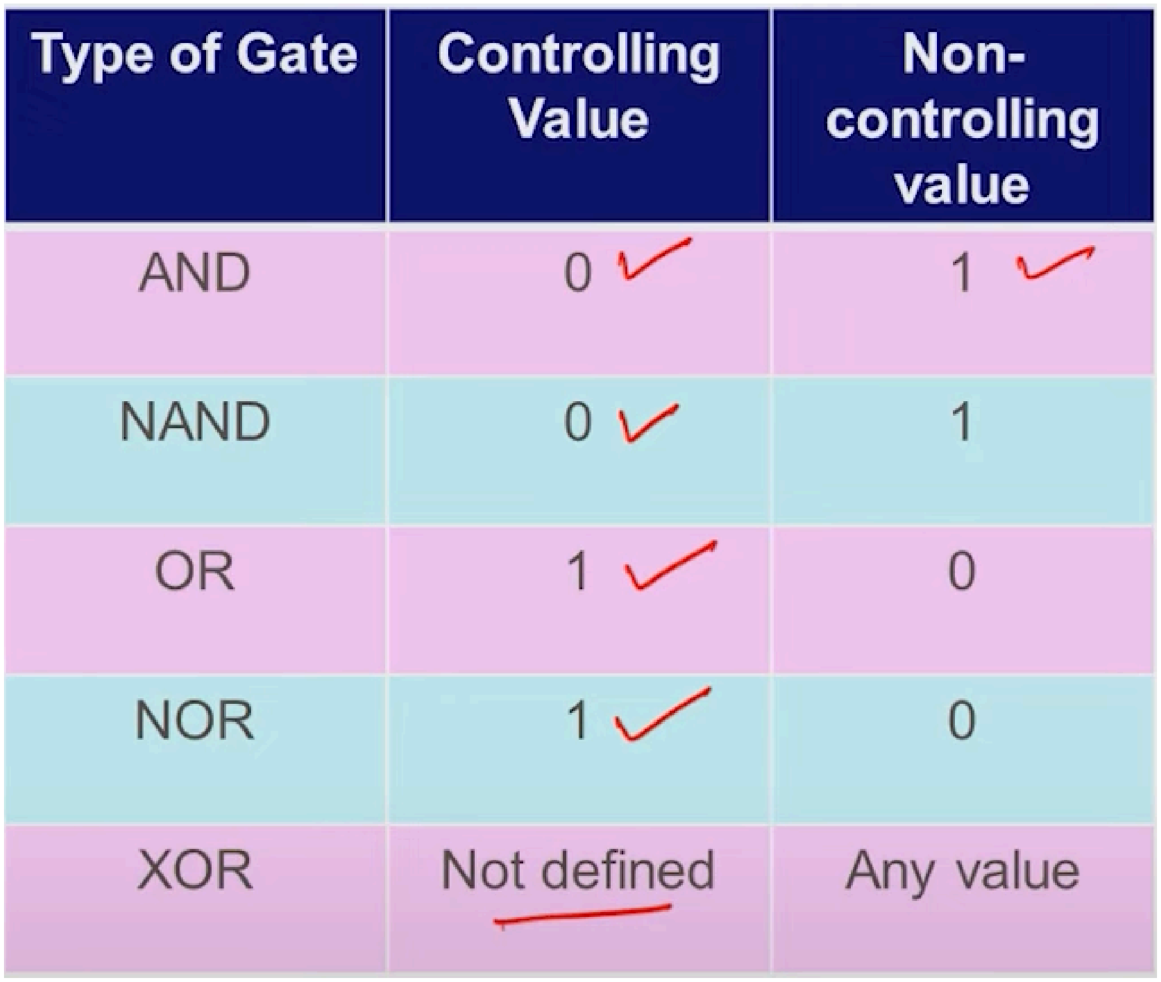
Now the input pins lying on this path are known as the on input on input pins or on path input pins of this path. So what are the input pins? The input pins are X2 X G5 X so G4 X2 G5 X and G7 X . So what are on input pins? On input pins for the path for P1 so they are G4 X2 G5 X . So G4 X2 G5X and G7X . These are on input pins for P.

Now what are side path inputs or side inputs? So the side inputs are the input pins other than the on inputs of the instances lying on that path . Now if we consider this path P1 then the instances lying on the path are this . Now what are the inputs of these instances other than the on inputs? Those are the side inputs. So for G4 X1 input is the one other than the on input . For G5 there is no side input .

Now for G7 X2 and X3 these all are these both are the side inputs. So the side inputs for P1 are G4 X1 . So note that G4 X2 was on input and G4 X1 is the side input and for G5 there is no side input. For G7 X2 and G7 X3 these are side inputs . So now we have understood that for a combinational circuit what are paths, what are on inputs and what are side inputs. We will use these definitions in the later part of this lecture.

Now for a multi input gate we should also understand what is controlling value . Now what is a controlling value for a multi input gate? So the value that can be assigned to any input of the gate such that output is known irrespective of the values on the other inputs. Those are known as the controlling value. So let us understand it with an example . Let us consider an AND gate .

If we can assign a value to any input of the gate for an AND gate that the output is known irrespective of what their values are at the other inputs . So for an AND gate if suppose we make this input A is equal to 0 then irrespective of what is there at the other input the output is known that why will the output of this AND gate be 0 . So 0 is a controlling value for an AND gate . Similarly for a NAND gate the controlling value will be 0 because if in a NAND gate if we make any input as 0 then the output is known to be 1 irrespective of other inputs . Similarly for an OR gate if the input is made 1 then the output becomes 1 irrespective of other inputs and therefore 1 is the controlling value for an OR gate and 1 similarly 1 is controlling value for an OR gate.

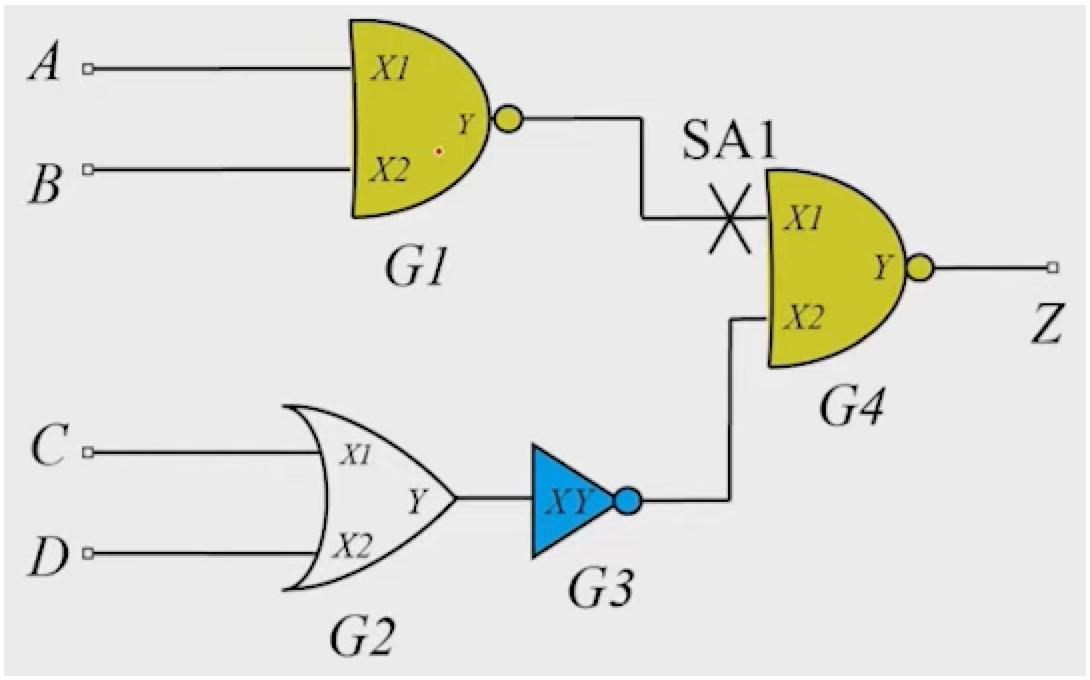


Type of Gate	Controlling Value	Non-controlling value
AND	0 ✓	1 ✓
NAND	0 ✓	1
OR	1 ✓	0
NOR	1 ✓	0
XOR	Not defined	Any value

For an XOR gate the output always depends on the other input . For example if we have 0 then at the input of an XOR gate the output will be 0 if the other input is 0 or it will be 1 if the other input is 1 . So irrespective similarly if we put any value or if we put 1 at the input of an XOR gate if the other input is say 0 the output will be 1 and if the other input is 1 then the output will be 0 . So even if we put 0 or 1 we do not know what the output will be because the output will be defined by the other input and therefore there is no controlling value for an XOR gate. Now what are non-controlling values of a multi input gate? So a value that can be assigned to any input of the gate such that the output is decided by the value of the other inputs .

Now it is easy to guess for an AND gate what will be the non-controlling value. The non-controlling value will be 1 because if we put 1 on an AND gate the value will be decided by other inputs whether it is sorry other inputs whether the other inputs are having 0 or 1 depending on whether the output will get a value of 0 or 1 . So the 1 is a non-controlling value for AND gate NAND gate and 0 is a non-controlling value for OR gate and NOR gate because when they take non-controlling value at an input the output will decide will be decided by some other inputs of that gate and for XOR gate we have seen that if it takes 0 or 1 the output will be decided by the other input and therefore 0 and 1 both are non-controlling value for an XOR gate. Now let us look into a generalized approach to generating a test pattern for a combinational circuit for single-stuck-at fault model. So there are many algorithms for generating test patterns but let us look into a method which is known as path sensitization method.

So in the path sensitization method we generate a test pattern in three steps. The first step is fault sensitization, second is fault propagation and the third is line justification. So we will be looking at what these steps do with the help of an example. So let us assume that we have a circuit which is a combinational circuit which is shown here and there is a stuck-at fault or we want to generate or get a test vector or test pattern for the stuck-at fault or stuck-at-1 at this input pin that is G4 X1. So let us generate or determine the test pattern to detect a stuck-at-1 fault at the input or input pin G4 X1 .



Now how do we generate the test pattern? So the first step is to do fault sensitization. So in fault sensitization we find the value at the input ports that will be set the value at the fault side opposite of the fault value . So in this case the stuck-at-1 fault is there and the opposite value to the fault is 0 . It is stuck-at-1 so we take the opposite of that and that is 0. So now we want first to find the value at the input ports of the combinational circuit which will make this or which will lead to the value of 0 in this case .

So why we have been confined to finding a set of values at the input ports is because during testing we have access only to the input ports and the output ports in our design or in our circuit . And therefore test patterns are the values that will be applied at the input ports . So first what we do is that we find the value at the input port which will drive or lead to a value at the fault side opposite to what the fault we have assumed. In this case the fault that we have assumed is stuck-at-1 . So we want a 0 at the fault side that is opposite of it and we want to find the value at the input port which will lead to a value 0 at this point .

So why do we want to do this or why do we do fault sensitization? So intuitively what it means is that we are able to get the opposite of this value if opposite of the value of the fault case . So the fault is 1 thus this is a stuck-at-1 fault. So the faulty value at this point at the fault side is 1 and we are trying to get a value of 0 . So once we get the value at the input ports opposite of the fault value, what it means is that at this particular point at the fault side the behavior of a faulty circuit and a fault free circuit will be different . Because in the fault free circuit the value will be opposite it will be having a value 0 and in the faulty circuit it will have a value 1 .

Now in this case if we want to get a value 0 at the fault side what will be the value we must apply at the input port? If this is a NAND gate we want to get a 0 and we can get a 0 at the output of the NAND gate only when all the inputs are having a value of 1 . So a should be 1 and b should be 1 . So we find that to get a value 0 at g4 x1 at this point we must have a is equal to 1 and b is equal to 1. So this is what is done in fault sensitization. Then we go to the next step, which is fault propagation.

What we do in fault propagation is that for the path from the fault side to the output port set the value of the side inputs to the non-controlling value . Now what we do is that we go from this fault side to the output port . This is the path we are considering . So the path here is G4X1 G4 XY and Z . Now for this path what is the side input? The side input is X2.

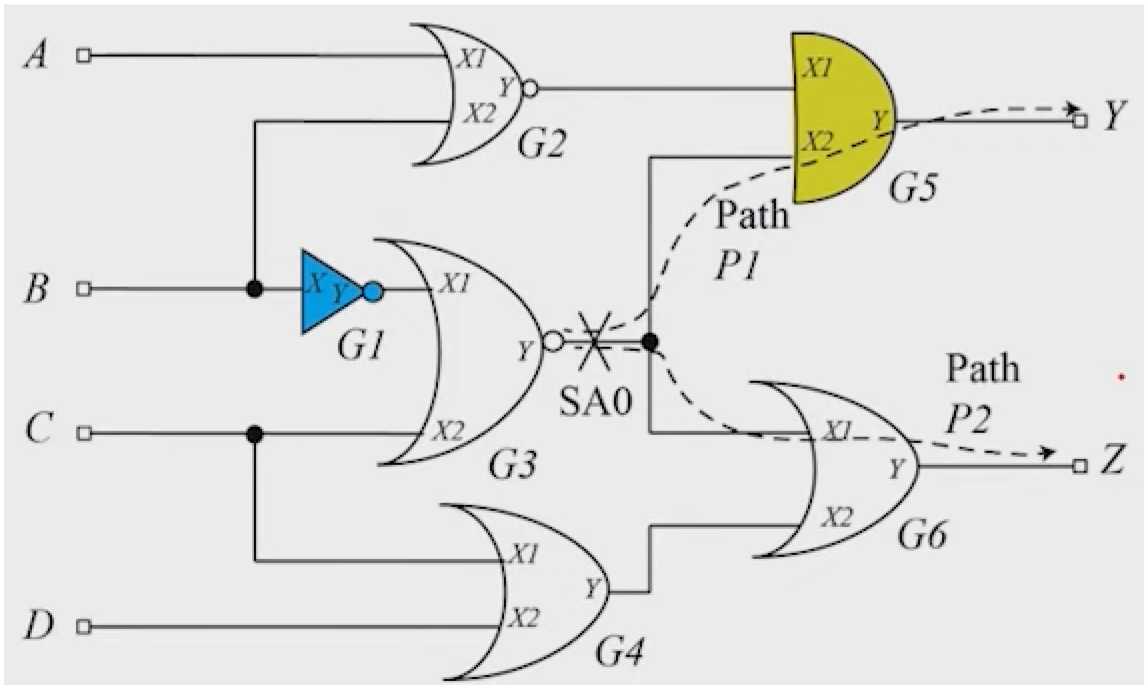
The side input is G4 X2. Now we want this G4 X2 that is side input to take a non-controlling value . So we want to set G4 X2 to a non-controlling value and this is a NAND gate . This is a NAND gate and the non-controlling value of a NAND gate is 1 . So we want to set g 4 x 2 to equal 1 . So why do we want to do this fault propagation and why do we want to set the side inputs to a non-controlling value? Intuitively what we are doing is that if we set the side inputs to non-controlling values then the output of that gate will be decided by the ON input .

The ON input in this case is G4X1 . So the value at G4X1 will be actually deciding the value of this gate and therefore we can observe the fault and if we do this throughout the path we will be able to propagate the effect of the fault at the output or we can observe the effect of the we can observe the difference between the faulty circuit and the fault free circuit at the output . Now in the third step we do line justification. So in the line justification we find the value at the input ports that will set the side inputs as found in fault propagation. So in fault propagation we found that this G4X1 must be equal to 1. Now we have to find the value of the input ports which will give us the value of 1 .

Now how can we do it? So if we want this value to be 1 so the inverter should have at its input a value of 0. Now one can have a value of 0 at the output of an OR gate when both its inputs are 0 . So c is equal to 0 and d is equal to 0. So we get c is equal to 0 and d is equal to 0 as the value that we want . So now what is the combination of all this? What values do we get at the input ports?

So in the fault sensitization we had got a is equal to 1 and b is equal to 1 and in line justification we got c is equal to 0 and d is equal to 0 . So now what is the test vector? So the test vector is a b is equal to 1 b is equal to 1 c is equal to 0 and d is equal to 0. Now we can check whether this is a valid test pattern or not . Now if there was no fault in our circuit and we apply 1 1 0 0 what will be the output. So if this is 1 1 we will get a value here as 0 and here both are 0 we get a value of 0 0 and we will have here as 1 here we have 0 we have 1 and this is a NAND gate one input is 0 so it will output will be 1.

So this will get z is equal to 1 for fault free circuit. Now what will we get for a faulty circuit? For a faulty circuit there will be a stuck at 1 fault at this point will have here will have 1 and the other values here c d will have value 0 0 the output here will be 0 the in output of the inverter will be 1 and both the inputs of this NAND gate is getting a value of 1 and therefore red will be is equal to 0 when the when the when the circuit is faulty . So we are getting different values for a faulty circuit and a faulty circuit for this input pattern . So we have got the required test vectors. Now there can be complications in this method of finding a test pattern using fault sensitization method .



Now what can be the complication? Let us take an example. Suppose we are given that we want to find a test pattern for something stuck at 0 at this pin so we first do fault sensitization. Now to get this fault sensitization we must get a value of 1 opposite to the value at this point at the fault side. Now what should be the value at the input ports? If you want 1 here and this is a NOR gate both its input should be 0. So C is equal to 0 and to make this point output of G1 as 0 we must have B is equal to 1 .

So we get B is equal to 1 and C is equal to 0 by fault sensitization. Now we want to do fault propagation. Let us first take this path . Now for this path the side input is G5 X1. We want to make it as 1 because this is an AND gate G5 is an AND gate so we want to make G5X1 the side input to the non-controlling value that is 1.

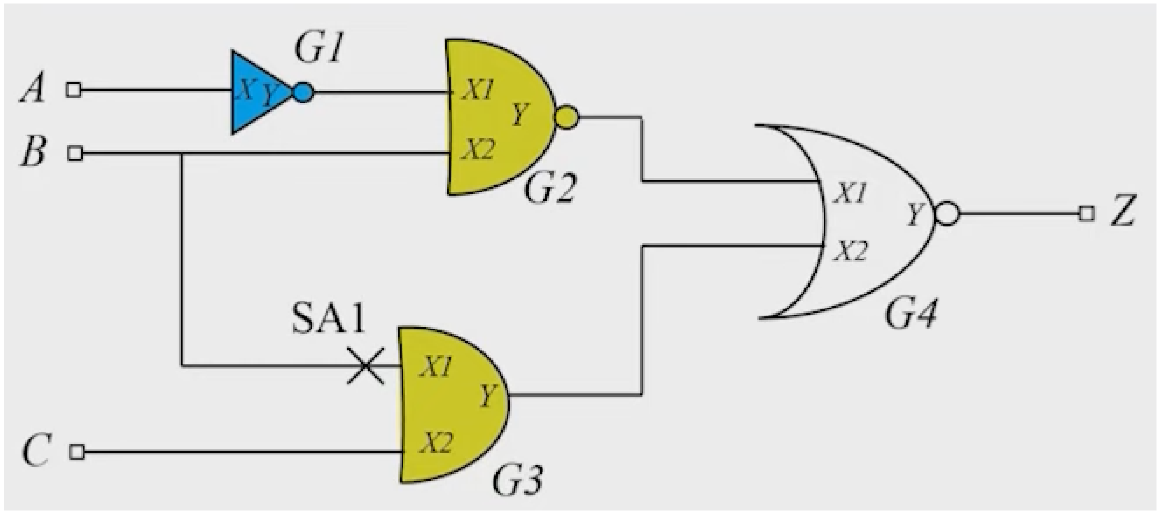
Now to get 1 when we go to line justification, when will the output of a NOR gate G2 become 1 when both its inputs are 0 . Now if we want to make G2 X2 as 0 B lines or B input port it should get a value of 0 . Now in line fault sensitization we have found that B is equal to 1. Now we want to make it 0 . So at the same time we cannot have two values at B and therefore there is a conflict.

So we cannot this means that we cannot propagate this fault to the fault to the output Y with this earlier decision of propagating the effect of this fault to the output Y output port Y needs to be back track means to be changed. So let us look at what is the other option. The other option is that we can propagate this fault to the other output port that is Z . So let us do fault propagation through Z . Now if we consider a propagation through

Z then the side input is G6 X2 and this is an OR gate which must make this as 0 that is non-controlling value.

Now to make this in G6 X2 is equal to 0 what must be the we need to do line justification. So when we do this G4 is an OR gate and its output must be made 0. Now when can be and we can make it 0 by making both its input C is equal to 0 and D is equal to 0. Now C is equal to 0 is consistent with fault sensitization and therefore there is no conflict and we have got the test pattern. So what is the test pattern? So the test pattern is the value at X is immaterial the value at X input A the input A is immaterial the B must be held to a value 1 C must be held to a value 0 and D must be held to a value of 0.

Now I leave it as an exercise to for you to check whether this is a valid test vector meaning that if there is a stuck at fault stuck at 0 fault at this point whether the output produced at Z at the output port Z whether you are getting two different values for a faulty circuit and fault free circuit for this test vector. Now let us look at another example: suppose we want to find a test vector for the stuck at 1 fault at the input G3 X. So to find that first we need to do fault sensitization. Now the fault value is 1 so we must get a value of 0 at this point. Now to get 0 what value we must set at the input port we must set B is equal to 0.



Now the next step is fault propagation. So we want to propagate the fault from this point to the output Z. Now what are the side inputs in this case? The side inputs are G3 X2 and G4 X1. We want to set them to non-controlling values. So we must set this to a value of 1 because this is an AND gate and non-controlling value for an AND gate is 1 and for G4 X1 we must set it to a value of 0.

Next step is the line justification. So in line justification we must get these values at the side input using the input port. Now for the side input G3 X2 we must make C equal to

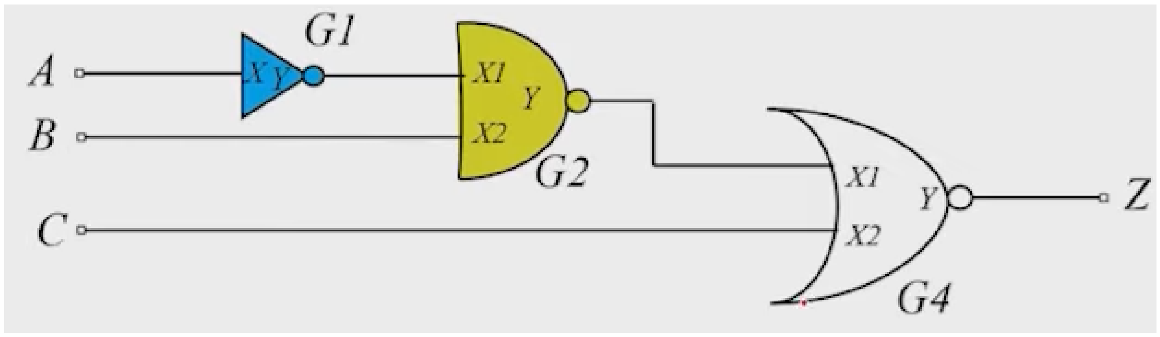
1. Now for the other input that is the side input that is G4 X1 this must be made as 0. Now to get a 0 for a NAND gate which must have both input as 1. Now again if we want to make G2 X2 as 1 we must make the line B equal to 1.

But in earlier fault sensitization we have seen that B must be held a value of 0. So we have got a conflict again. Now in this case there is no other path to explore and therefore there is nothing to backtrack. So if there is no way in which we can get a consistent value or there is no conflict for a given fault then in that case there will be no test vector which will exist to detect that fault and such types of faults are known as redundant. So what are redundant faults? The faults in which the faults which cannot be tested using any test pattern.

So in this case we have 3 inputs. So the number of possible test patterns are 2 to the power 3 or 8. Now out of these 8 patterns there is no pattern if we use any of these patterns from this test from this 8 combination. 8 combination there will be no combination which will give different values for a faulty circuit and a fault free circuit. Now if that is the case then no test vector exists for this circuit to detect the stuck at 1 fault at the input pin G3 X1.

So in that case we say that being stuck at 1 fault at G3 X1 is a redundant fault. We cannot detect it using test vectors. Now if that is the case then we can actually use these redundant faults to optimize our circuit. How can we use it? So redundant fault means that the behavior of the circuit with and without fault is the same. If that is the case we can assume that at the fault side we have a constant value 0 or 1 as per the as given in the fault. So for example if it is stuck at 1 fault we can assume that 1 is always there in our circuit and using that assumption we can optimize the hardware in our circuit.

How can we do it? Let us see. So this is the same circuit that we saw in the previous slide and we have seen that being stuck at 1 fault at this input pin is a redundant fault. So now we assume that stuck at 1 at G3 X1 is always present. Now if we tie this value this G3 X1 to a value 1 then what will be the output G3 Y? So G3 Y output will always be the value at the input C. So we can say that G3 this AND gate is serving no purpose but this output is something at the or the output of this AND gate is nothing but C. And therefore we can redraw this circuit as in which we are driving this G4 X2 not through this AND gate but through the input port C.



So now we are able to reduce the hardware in our design because we have removed one AND gate. Now we can verify that the original circuit and the reduced circuit both are functionally equivalent. So I leave it again as an exercise for you using simple algebraic manipulation and De Morgan's law you can check for yourself or by drawing the truth table of these two circuits you can check that for any input combination these two circuits will produce the same output. So these two circuits are equivalent functionally but the modified circuit has got less hardware because it has got one less AND gate.

Now let us look at what challenges do we encounter in ATPG. So the challenges that we encounter in ATPG are primarily due to reconvergent fanouts. Now what is a reconvergent fan out? Suppose there is a gate and from there there is a multiple fan out and later on through combination circuit elements they converge back to some logic gate. So this kind of a structure is known as a reconvergent fan out structure and if this kind of structure appears in our circuit then it becomes more challenging for ATPG too. Why does it become more challenging? Because in ATPG at various levels then it needs to make some decisions and decisions lead to implications and these implications can actually lead to conflicts. So multiple implications can lead to conflicts and then we have to backtrack our earlier decisions and these backtracking leads to more run time and therefore it makes the algorithm more complicated.

So in ATPG tools what we do is that we set some backtracking limit. Why we want to set a backtracking limit is because there can be redundant faults in our circuit and for redundant faults there is no test pattern actually in a circuit which can detect the fault. So we can do many backtracking and still will not be able to get the required test pattern. So ATPG tools have a backtrack limit. They say that if there is a backtrack limit say 1000 then if that number of backtracking has been done for a given fault then it will assume that that fault is a redundant fault and it will not and it will move to the next fault or find a test vector for the next fault. So there is some backtracking limit typically used in ATPG tools to avoid spending too much run time on the redundant fault.

Now while setting this backtracking limit we must be careful because if we make this backtrack limit very small say 10 or 20 in that case what will happen is that there will be many faults which are actually non redundant and for them it will hit this backtrack limit and the tool will avoid finding the test vector for that fault. So it will not it shouldn't be too small and if it is too large, say maybe 10,000 then in that case the tool may be doing so many backtracking for redundant faults and at the end of because for redundant faults as many times you do backtracking will not get a test vector. So in that case for redundant faults it might spend too much run time which is not actually useful and therefore the reasonable value of backtrack limit should be used. Now the ATPG algorithm has been studied for the last 50 years or so and a lot of advancements have been made. In the discussions that we have seen in the previous slide we have looked into a very simple kind of fault sensitization method based on ATPG.

The first algorithm was proposed in 1966 or that was known as D algorithm by Roth and later PODM algorithm was proposed by Prabhu Goel in 1981 and FAN algorithm was proposed by Fujiwara in early 1980s. So these are some of the algorithmic advancement or important algorithmic advancements in the ATPG algorithm. Now if you want to know more about ATPG I suggest that you go and refer to this book. Now to summarize in this lecture what we have done is that we have looked into a method of generating a test pattern for a combinational circuit stuck at fault and that method is fault sensitization method. And we also looked into what are redundant faults and how they can be used to optimize our circuit.

Now in last two lectures we have looked into scan design method and combinational ATPG and a combination of a scan design method and combinational ATPG is very popular and that is basically for auto that has to be used using automatic test equipment or those test or DFT methods are basically targeted for automatic test equipment based testing. Now in the next lecture we will be looking into another testing paradigm which is known as a built in self test. Thank you very much.