

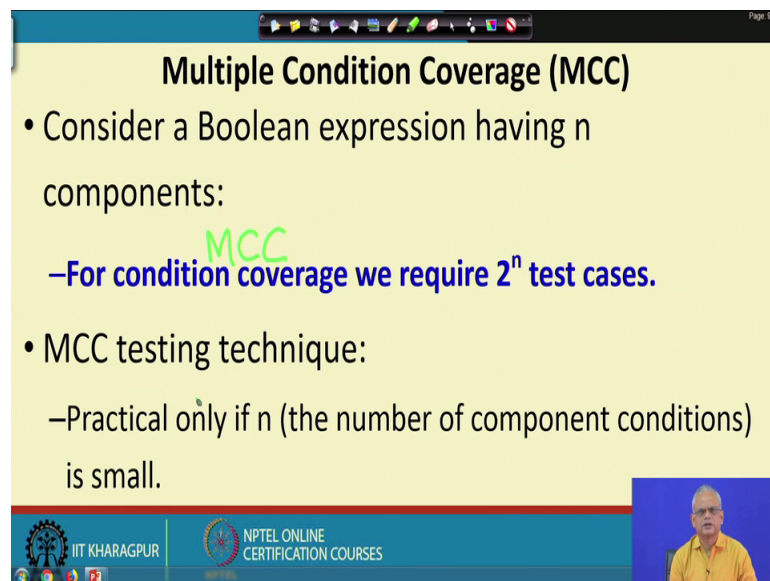
Software Engineering
Prof. Rajib Mall
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 57
MC/DC Coverage

Welcome to this lecture. In the last lecture, we are discussing testing branches and we said that the simple branch decision coverage testing is not really a good enough testing bugs can remain, if the branch condition contains multiple sub expressions. And, then we looked at the basic condition coverage testing and then we said that even though it overcome some short comings of the branch coverage testing, but it is not really a stronger testing than branch coverage. And therefore, we considered the condition decision coverage testing.

Where, the basic condition testing is achieved and the same time branch coverage is achieved but then we are trying see, which is the strongest branch testing strategy and we said multiple condition coverage testing. Where each of the sub expression is given all possible combinations of truth values, but then the number of test cases became exponential let us continue from that point.

(Refer Slide Time: 01:45)



Multiple Condition Coverage (MCC)

- Consider a Boolean expression having n components:
 - For condition **MCC** coverage we require 2^n test cases.
- MCC testing technique:
 - Practical only if n (the number of component conditions) is small.

The slide includes a video inset of Prof. Rajib Mall in the bottom right corner. The footer contains the IIT Kharagpur logo and the text 'NPTEL ONLINE CERTIFICATION COURSES'.

Let us consider a Boolean expression having n sub expressions or components.

The multiple condition coverage MCC, MCC or the multiple condition coverage will required 2^n test cases. And, that is the reason why MCC testing is not considered practical, if the numbers of sub expressions are more.

(Refer Slide Time: 02:37)

MCC for Compound conditions: Exponential complexity

$((a || b) \&\& c) || d) \&\& e$

$2^5=32$

Test Case	a	b	c	d	e
(1)	T	—	T	—	T
(2)	F	T	T	—	T
(3)	T	—	F	T	T
(4)	F	T	F	T	T
(5)	F	F	—	T	T
(6)	T	—	T	—	F
(7)	F	T	T	—	F
(8)	T	—	F	T	F
(9)	F	T	F	T	F
(10)	F	F	—	T	F
(11)	T	—	F	F	—
(12)	F	T	F	F	—
(13)	F	F	—	F	—

• Short-circuit evaluation often reduces number of test cases to a more manageable number, but not always...

Even, if we have a 5 clauses here this example expression just 5 clauses. And, to test for multiple condition coverage, we need 2^5 test cases and that is 32.

But, then we will see that short circuit evaluation reduces the number of test cases to a more manageable level. The short circuit evaluation means, that compiler while evaluating, it infers whether it needs to evaluate further. For example, let us say a is true and let us say c is true. Then the others the compiler does not really bother because they will have no effect if a is true then this expression is true.

And, if c is true then this is true and if e is true and then terms have to be true d does not matter. So, b and c whether it is true and false it does not matter. So, you can just consider one of that. And therefore, the number of test cases reduces by 1, but what about if let say a is true and c is false. Then this becomes false irrespective of the value of the b, and let say d is false so; that means, this is false. And therefore, e does not matter.

So, we can just consider one of the 2 possible test cases. And therefore, the number of test cases may not be 32 due to short circuit evaluation technique used by the compilers, because they want to execute fast and they see that if the truth value can be inferred as

they evaluate then they do not evaluate the other once. So, even though we are saying that 32 test cases are needed, but then depending on the compiler the short circuiting approach that, it takes we might need much less than 2 to the power n to achieve the multiple conditions coverage testing.

(Refer Slide Time: 06:01)

Subsumption

- Condition testing:
 - Stronger testing than branch testing.
- Branch testing:
 - Stronger than statement coverage testing.

The diagram illustrates the subsumption hierarchy of testing types:

- Multiple Condition** (top level)
- ↓
- Condition/Decision**
- ↓
- Decision**
- ↓
- Statement** (bottom level)

Additionally, there is a diagonal arrow from **Condition/Decision** to **Basic Condition**, indicating that condition/decision testing is stronger than basic condition testing.

The slide footer includes the IIT KHARAGPUR logo and NPTEL ONLINE CERTIFICATION COURSES text. A small video inset of a speaker is visible in the bottom right corner.

So, based on our discussion so far we can say that the statement coverage is the weakest testing. And, so is the basic condition coverage testing, but then they are not comparable because basic condition testing does not ensure statement coverage testing and vice versa, but then the branch coverage testing or the decision testing is stronger than the statement coverage testing.

And the condition decision coverage testing is stronger than both basic condition testing and the decision testing. And, the multiple condition testing is the strongest of all this, but the only problem is that is not practical, when the number of sub expressions is large.

(Refer Slide Time: 07:04)

The slide is titled "Shortcomings of Condition Testing" and contains the following content:

- **Redundancy of test cases:** Condition evaluation could be compiler-dependent:
 - Reason: Short circuit evaluation of conditions
- **Coverage may be Unachievable:** Possible dependencies among variables:
 - Example: `((chr=='A') || (chr=='E'))` can not both be true at the same time

Handwritten annotations in green include:

- A truth table for the expression `((chr=='A') || (chr=='E'))` with columns for 'A' and 'E' and rows for 'T' (True) and 'F' (False). The result column shows 'T' for (T, F) and (F, T), and 'F' for (T, T) and (F, F).
- The acronym "MCC" (Multiple Condition Coverage) written above the example.

The slide footer includes the IIT KHARAGPUR logo and the text "NPTEL ONLINE CERTIFICATION COURSES". A small video inset of a speaker is visible in the bottom right corner.

We had seen that the short circuit evaluation approach used by compilers often reduces the number of test cases to achieve multiple condition coverage. Also, we must remember that certain coverage may not be achievable. Let say we have expression like character is A or character is E. Here, we cannot achieve the basic condition coverage also, because we cannot have we can achieve basic condition coverage, because we can have this as true and this as false and this as true and, this as false.

We can achieve basic condition coverage testing we can achieve decision coverage testing, but can we achieve condition decision coverage testing. That also you can achieve, but what about multiple condition coverage testing. Can we achieve both of these true at the same time no, because the character cannot be same at the same time A and E. So, multiple condition coverage testing will we cannot achieve for this.

And, that we must keep in mind that we cannot achieve multiple condition coverage testing even though we deploy 1000s of test cases, we cannot achieve multiple condition coverage even for simple expressions.

(Refer Slide Time: 09:31)

Short-circuit Evaluation

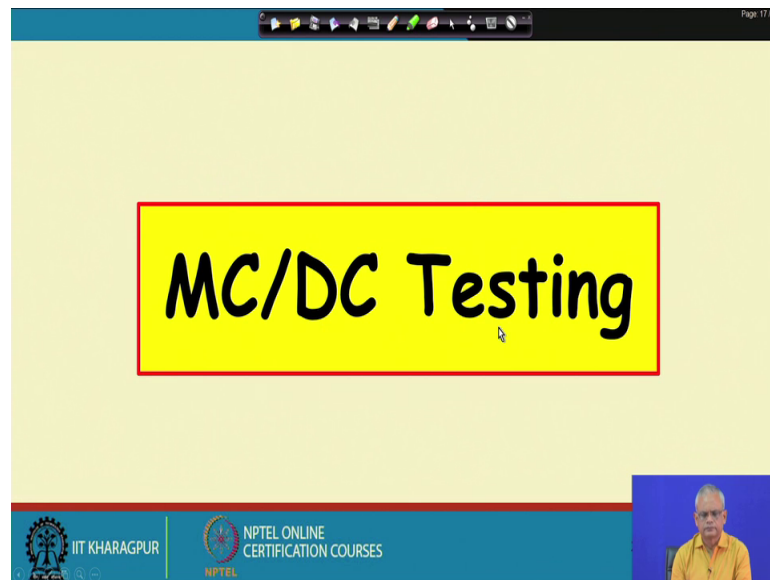
- **if(a>30 && b<50)...**
F — T
– If a>30 is FALSE compiler need not evaluate (b<50)
- Similarly, **if(a>30 || b<50)...**
T — T
– If a>30 is TRUE compiler need not evaluate (b<50)

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Let us look at few more short circuit evaluations. In this expression, there is a and here. And therefore, wherever there is a and if one of the sub expression evaluates to false, then the clause has to the decision has to evaluate to false irrespective of the other clauses.

The compilers take advantage of this and as soon as they find a false here in this kind of expression, they will not evaluate the others. They become don't cares and the outcome will be false. Whenever there is a or condition, whenever there is any sub expression evaluates to true, then the others become don't care and the decision evaluates to true. The decision will become true, even if one of that is true. And, if the compiler deploys short circuit evaluation, it can save lot of time and skip evaluation of sub expressions.

(Refer Slide Time: 11:01)



Now, let us look at a test strategy, which is the strong testing strong testing for decisions. It overcomes the problem of the multiple condition coverage testing, which required large number of test cases. When we had a decision involving let say 10 sub expressions for multiple condition coverage might need 2 to the power 10 test cases which is a about a 1000 test case, but if we use the MC DC testing that is multiple condition and decision coverage. Then we might need let say 7 or 8 test cases, but this 7 or 8 test cases, we can see experimentally that it is almost as good as the multiple condition coverage testing.

And, undoubtedly the multiple condition decision coverage and MC DC coverage testing is a very powerful strategy with very small number of test cases; it can achieve a thorough testing of the branches. And therefore, it is a very popular technique and incorporated into many testing standards now, let us look at the MC DC testing.

(Refer Slide Time: 12:38)

Modified Condition/Decision Coverage (MC/DC)

- **Motivation:** Effectively test important combinations of conditions, without exponential blowup to test suite size:
 - “**Important**” combinations means: Each basic condition should independently affect the outcome of each decision

`((c == CHAR) | | (c == DIGIT))`

The slide includes a video player interface at the top with a toolbar and a small video inset of a speaker in the bottom right corner. The footer contains the IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES logos.

Here, we need to test all possible outcomes for each of the conditions. These are the basic conditions, but then we require that they independently affect the outcome of the entire decision.

What it means is that, we not only require that each of the sub expression achieve true and false values, but also while the rest of the expression is held at some truth value. If you make this true to false then the decision should also change. The decision outcome will become false, if we make this as the falls and if we make this as the true then the decision outcomes would become true. Our test cases would be able to achieve that and then we will say that this satisfies MC DC coverage criteria.

(Refer Slide Time: 14:01)

Modified Condition/Decision Coverage (MC/DC)

- **Motivation:** Effectively test important combinations of conditions, without exponential blowup to test suite size:
 - “**Important**” combinations means: Each basic condition should independently affect the outcome of each decision
- **Requires:** `((c == CHAR) || (c == DIGIT))`
 - For each basic condition *c*, Compound condition as a whole evaluates to true or false as *c* becomes T or F

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, here the MC DC coverage requires that for every basic condition in a complex conditional statement. The compound condition evaluates to true and false as the basic conditions are given true and false values. Let us explore further about this test technique.

(Refer Slide Time: 14:29)

Condition/Decision Coverage

- Condition: true, false.
- Decision: true, false.

Multiple Condition coverage (MCC)

- all possible combinations of condition outcomes in a decision
- for a decision with *n* conditions

2ⁿ test cases are required

Modified Condition/Decision coverage (MC/DC)

- Bug-detection effectiveness almost similar to MCC
- Number of test cases linear in the number of basic conditions.

Subsumption hierarchy

```
graph TD; MCC[MCC] --> MCDC[MC/DC]; MCDC --> CD[Condition/Decision]; CD --> BCC[BCC]; CD --> Decision[Decision]; Decision --> Statement[Statement]
```

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

So, for we looked at the condition decision coverage testing, it achieves both condition coverage and decision condition coverage. The multiple condition coverage testing which required exponential number of test cases, but it was a strong strategy. And, now

we are going to discuss about the MC DC, where the bug detection effectiveness is very similar to the MCC, comparable to the MCC.

But, the number of test cases is linear in the number of basic conditions. If, we can draw the subsumption hierarchy, we can see here that the MC DC testing is requires small number of test cases. It is stronger than condition decision coverage testing, but of course, it is weaker than the MCC multiple condition coverage testing. But multiple condition coverage testing requires huge number of test cases, but MC DC testing achieves testing, which is very close to the MCC testing. So, that is the reason why MC DC testing is a important test strategy used extensively during testing.

(Refer Slide Time: 16:01)

• MC/DC stands for **Modified Condition / Decision Coverage**

• It is a condition coverage technique

– **Condition:** Atomic conditions in expression.

– **Decision:** Controls the program flow.

• **Main idea:** Each condition must be shown to independently affect the outcome of a decision.

– **The outcome of a decision changes as a result of changing a single condition.**

What is MC/DC?

$((a > 10) || (c == 5)) \& (d > 50)$

T	T	T	T	T
T	T	F	T	F
T	T	F	F	F
T	F	T	T	F
T	F	T	F	F
T	F	F	T	F
T	F	F	F	F
F	T	T	T	F
F	T	T	F	F
F	T	F	T	F
F	T	F	F	F
F	F	T	T	F
F	F	T	F	F
F	F	F	T	F
F	F	F	F	F

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

Now, let us try to understand, what is MC DC and how to design the MC DC test cases. As we have been already using the terms condition and decision, the condition is the atomic condition in expression or is a sub expression. The decision is the outcome of a decision statement and controls the program flow, already seen what are the basic conditions and what is the decision. The main idea in the MC DC testing is that given an expression having multiple atomic expressions or conditions, each condition must be shown to independently affect the decision.

That is if we have a expression like a greater than b, a greater than 10 or c equal to 5 and d greater than 50. Let say this is our expression or the decision statement, now for achieving the MC DC will hold it with some value let say c equal 5 and d is equal to let

say 60. And, then as we make a greater than 10 that is true, then the outcome should be true and if I make this is false the outcome will become false.

And, similarly if you hold this and this 2 some values some truth outcomes let say this is true and this is let say true. And, then if we make this as true and false the decision output the decision outcome should also become true and false. Similarly, for this clause the third clause sorry the second clause, if we hold these 2 in some true and false values let say this is true and this is true. And, if we toggle this sorry this is false and this is true. And as we toggle, this second sub expression to true and false then the outcome will toggle to true and false. Let see does it really do we have held this first sub expression to false.

The last sub expression to true, now if we make the second sub expression to let say true then this evaluates to true and then this is also true. So, the outcome will be true, but what if we make this as false? So, let say now then try with false here so, this is false already this is so, this becomes false and this is true. And therefore, the outcome will be false. So, we can say that if I hold the first sub expression to false, the second sub expression sorry the last sub expression to true, then if we toggle the second sub expression to true and false the decision outcome also toggles to true and false. We need to do that for every sub expression, that it will independently affect the outcome of the decision. So, that is the main idea here in the MC DC testing.

(Refer Slide Time: 20:24)

Requirement 1: Three Requirements for MC/DC

- Every decision in a program must take T/F values.

Requirement 2:

- Every condition in each decision must take T/F values.

Requirement 3:

- Each condition in a decision should independently affect the decision's outcome.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES

If, we formally write down what we need during MC DC testing, the first requirement is that the decision should have got true and false value by the test cases. Every condition or the sub expression must have got true and false values during the testing. And, also each sub expression should independently affect determine the outcome of the decision. So, these are the 3 requirements.

(Refer Slide Time: 21:11)

The slide is titled "MC/DC Requirement 1" and contains the following text:

- The decision is made to take both T/F values.

Below the text is a diagram of an if-statement: `If ((a>10) && ((b<50) && (c==0))) then`. A blue bracket underlines the entire condition. Two arrows point from the bracket to the words "true" and "false".

- This is as in Branch coverage.

The slide footer includes the IIT KHARAGPUR logo, the NPTEL ONLINE CERTIFICATION COURSES logo, and a small video inset of a man in a yellow shirt.

Let us try to understand with some examples. Let us look at this expression a greater than 10 and b less than 50 and c equal to 0 then something. And, we want to test it such that the test cases will achieve MC DC coverage.

So, we need to assign the first sub expression true and false values such that the decision will become true and false. So, the first thing is that the decision as a whole should become true and false.

(Refer Slide Time: 22:01)

MC/DC Requirement 2

- Test cases make every condition in the decision to evaluate to both T and F at least once.

If ((a>10) && ((b<50) && (c==0))) then...

true false

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

And if we assign true and false to this that is a is 20, which is true and a is 5 false then the outcome should also become true and false. So, for what values of these 2, if you held hold them constant will this toggle the output as it become true and false. So, if we have this true and this true that is b is 20 and c equal to 0, then this sub expression becomes true and see here there is a and here. So, is this is true then the outcome the decision becomes true, if this is false then this is true and outcome becomes false.

So, as the first expression becomes true and false, if we hold these true to true and true then the outcome the decision outcome toggles. Similarly, we need to do for the other two sub expressions.

(Refer Slide Time: 23:32)

MC/DC Requirement 2

- Test cases make every condition in the decision to evaluate to both T and F at least once.

If ((a>10) && ((b<50) && ((c==0))) then...

true false true false true false

IIT KHARAGPUR NPTEL ONLINE CERTIFICATION COURSES

So, for these as it becomes true and false, we need to hold these 2 such that the outcome becomes true and false and also for the last sub expression so, that is the requirement for MC DC.

(Refer Slide Time: 23:52)

MC/DC Requirement 3

- Every condition in the decision independently affects the decision's outcome.

If ($(a > 10) \&\& ((b < 50) \parallel (c == 0))$) then...

true	false	true	false
------	-------	------	-------

If ($(a > 50) \&\& (b < 50) \parallel (c == 0)$) then...

true	true	false	false
------	------	-------	-------

If ($(a > 50) \&\& ((b < 50) \parallel (c == 0))$) then...

true	false	true	false
------	-------	------	-------

The slide includes logos for IIT KHARAGPUR and NPTEL ONLINE CERTIFICATION COURSES, and a small video inset of a presenter in the bottom right corner.

And, what are the values which will achieve MC DC coverage for this; one is that we hold this to true and this to false, if this is or and then as it toggles between true and false this will the decision will toggle between true and false. And, for this situation for the second sub expression, if we hold this to true and this to false then the outcome decision outcome will toggle between true and false. Similarly, if you hold the first one to true and second one to false, then the third sub expression as it becomes true and false the decision will toggle between true and false, but then the question remains that given a complex expression, which is having a large number of sub expressions are conditions. How do we design the test cases such that MC DC coverage will be achieved?

(Refer Slide Time: 25:03)

Page 21/21


MC/DC: An Example

- N+1 test cases required for N basic conditions
- Example:
$$(((a>10 \ || \ b<50) \ \&\& \ c==0) \ || \ d<5) \ \&\& \ e==10)$$

Test Case	a>10	b<50	c==0	d<5	e==10	outcome
(1)	<u>true</u>	false	<u>true</u>	false	<u>true</u>	true
(2)	false	<u>true</u>	true	false	true	true
(3)	true	false	false	<u>true</u>	true	true
(6)	true	false	true	false	<u>false</u>	false
(11)	true	false	<u>false</u>	<u>false</u>	true	false
(13)	<u>false</u>	<u>false</u>	true	false	true	false

- Underlined values independently affect the output of the decision

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES



Let us first look at an example so, it will another examples such so, that we can get the idea, about how to achieve MC DC coverage? We first develop the truth table and then we check that if the first one is true, then how can the output this is true? And for this one the outcome is true, this one first sub expression is true and the outcome is true and scan here and see that for same values of the other sub expressions; here as we toggles between true and false the outcome becomes false.

So, these 2 test cases together will achieve the independent evaluation of the first condition. What about the second condition? The second condition, we scan through the truth table. And, find that when this is true and others are true false true this is true. And, when this is false then the others are held true false true and we get false. So, these 2 together achieve the independent evaluation of the second clause, the third clause is when this is true and the others are the false true and true and this is false others are false true and true then the outcome toggles. So, we also need this and so on.

So, for every condition we look at the truth table and then we examine that when does the output toggle. The decision outcome toggle when the specific condition toggles. So, this is the main idea behind MC DC, we are almost at the end of this lecture will stop here and continue in the next lecture.

Thank you.