

We use cookies on our website to give you the most relevant experience by remembering your preferences and repeat visits. By clicking "Accept", you consent to the use of ALL the cookies.

[Do not sell my personal information.](#)

[Cookie settings](#)

ACCEPT

have landed at the right place.

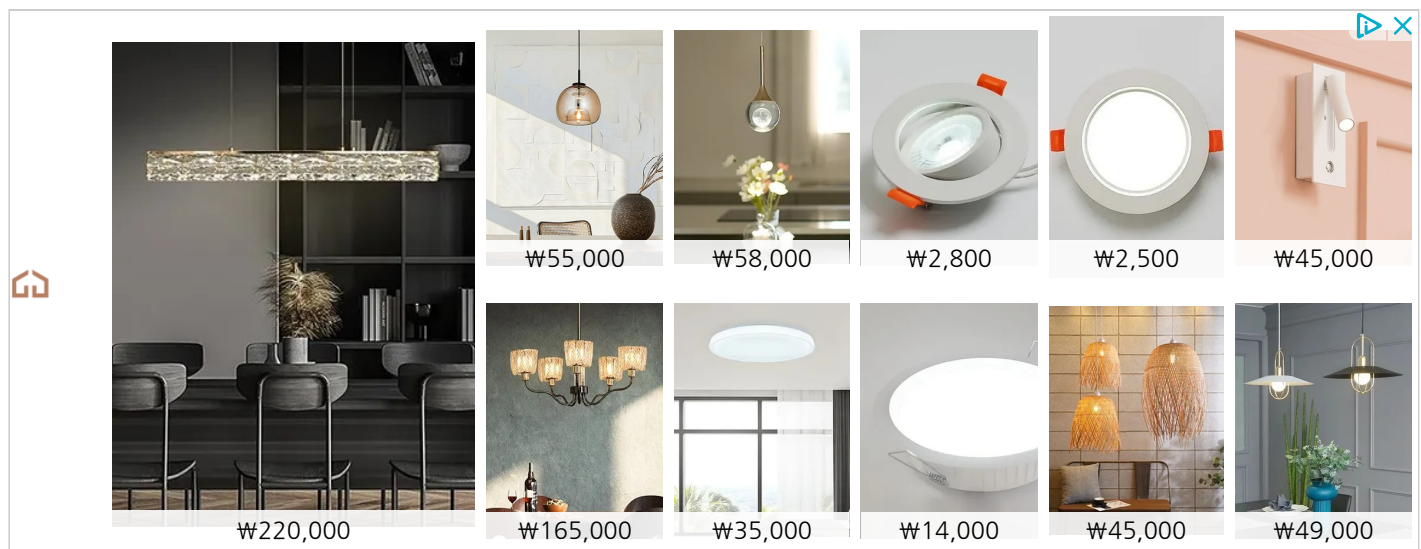


Table of Contents [[hide](#)]

1. What is the definition of MDCD or MC/DC?
2. What is Condition?
3. What is Decision?
4. What is Statement Coverage?
 - 4.1. Example of Statement Coverage:
 - 4.2. Test Cases for 100% Statement Coverage
 - 4.3. Drawbacks of Statement Coverage:
5. What is Decision Coverage?
 - 5.1. Example of Decision Coverage:
 - 5.2. Test Cases for 100% Decision Coverage:
 - 5.3. Drawbacks of Decision Coverage:
6. What is Condition Coverage?
 - 6.1. Example of Condition Coverage:
 - 6.2. Test Cases for 100% Decision Coverage:
 - 6.3. Drawbacks of Decision Coverage:
7. What is condition/decision coverage?
 - 7.1. Example of Condition/Decision Coverage:
 - 7.2. Test Cases for 100% Decision Coverage:
 - 7.3. Drawbacks of Decision Coverage:
8. What is Modified condition/decision coverage?
 - 8.1. Example of Modified Condition/Decision Coverage:

8.2. Test Cases for 100% Decision Coverage:

8.3. Achievements of Modified Condition/Decision Coverage:

9. What is Multiple Condition coverage?
10. Can you write the test cases to achieve MC/DC for $C=(A \text{ and } B)$?
11. Can you write the test cases to achieve MC/DC for the following expression: $D=(A \text{ and } B \text{ and } C)$?
12. Can you write the test cases to achieve MC/DC for the following expression: $C=(A \text{ or } B)$?
13. Can you write the test cases to achieve MC/DC for $D=(A \text{ or } B \text{ or } C)$?
14. Can you write the test cases to achieve MC/DC for $C=(A \text{ xor } B)$?
15. Can you write the test cases to achieve MC/DC for the expression: $B = \text{not } A$?
16. Can you explain how to achieve MC/DC for comparators?
17. How to achieve MC/DC for if-else statement?
18. How to achieve MC/DC for while loop?
19. How to achieve MC/DC for Bit-wise operations?
20. Have you used any coverage analysis automation tool?
21. How do you ensure that the instrumentation of code did not inject an additional bug in the software?
22. What are the commonly misunderstood facts about MC/DC?
23. What is the Independent effect in MC/DC?
24. Can you explain Masking in the context of MC/DC?
25. Can you write the MC/DC cases for the following expression:
 $Z := (A \text{ and not } B) \text{ or } (C \text{ xor } D)$?
26. $Z := (A \text{ and not } B) \text{ or } (C \text{ xor } D)$?
27. What is Coverage? Why Coverage?
28. What is the difference between structural coverage analysis and structural testing?
29. Why MC/DC? Why not do multiple-condition coverage?
30. Conclusion:
31. Related posts:

In this article, I have tried to create a collection of **MCDC interview questions** in the context of [DO178C](#). These questions are being asked in most of the aerospace companies for entry-level as well as experienced level positions. I have spent many hours compiling these sample interview questions. Some of the questions, I have collected informally from various aerospace professionals and also from other online resources.

So, I hope you will enjoy these **MCDC interview questions** and this would be helpful for

both fresher and experienced professionals.



망향비빔국수 풍세점

망향비빔국수 풍세점

열기

After reading the complete article if you have any questions, please write in the comment box. Our team will try to address your questions at their earliest possible time. Good luck for your interview!

What is the definition of MCDC or MC/DC?

▶ 0:00 / 0:49

Audio Answer

MCDC or **MC/DC** stands for **Modified condition/decision coverage**.

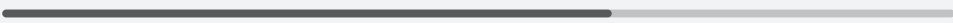


The official definition of MCDC or MC/DC as per [DO178C](#) is:

"Every point of entry and exit in the program has been invoked at least once, every condition in a decision in the program has taken all possible outcomes at least once, every decision in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect that decision's outcome. A condition is shown to independently affect a decision's outcome by: (1) varying just that condition while holding fixed all other possible conditions, or (2) varying just that condition while holding fixed all other possible conditions that could affect the

outcome."

DO 178C / ED 12 C

What is Condition?

▶ 0:00 / 0:25   

Audio Answer

The official definition of condition as per DO178C is:

"A Boolean expression containing no Boolean operators except for the unary operator (NOT)."

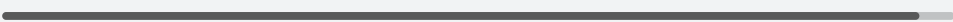


DO 178C / ED 12C

Let's take an example code snippet to understand the concept of condition:

```
if(speed_disp<0) OR (speed_sig_fault == 1) then
    speed_disp_fault = True;
end if;
```

Here in this case, (speed_disp < 0), (speed_sig_fault == 1) are two different conditions or clause.

What is Decision?

▶ 0:00 / 0:25   

Audio Answer

The official definition of decision as per [DO178C](#) is:

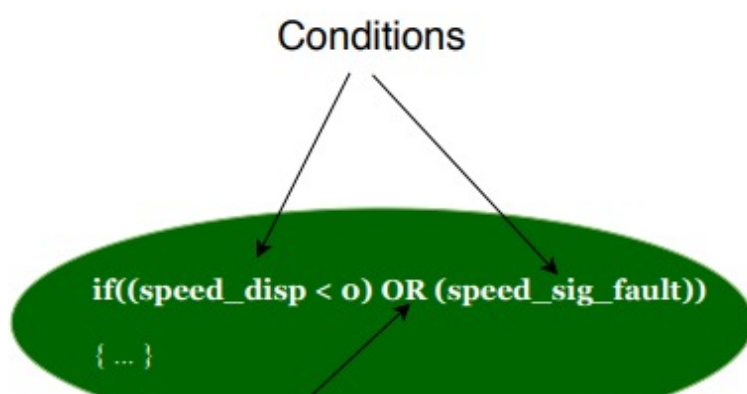
"A Boolean expression composed of conditions and zero or more Boolean operators. If a condition appears more than once in a decision, each occurrence is a distinct condition."

DO 178C / ED 12C

Let's take an example code snippet to understand the concept of decision:

```
if(speed_disp<0) OR (speed_sig_fault == 1) then
    speed_disp_fault = True;
end if;
```

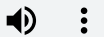
Here, in this case, in the if statement, the whole expression is called a decision or predicate, and "OR" is the Boolean operator, and "<", "==" are comparators.





What is Statement Coverage?

▶ 0:00 / 1:11



The statement coverage means that:

"Every statement in the program has been invoked at least once."

As per the above definition, every executable statement has to be invoked at least once to get 100% statement coverage. The 100% statement coverage does not exercise all the control structures in the software; therefore it is considered a very weak criterion in terms of structural coverage.

However, The statement coverage is very useful in detecting the dead code i.e. code that cannot be reached under any operating conditions or code that cannot be executed.

Example of Statement Coverage:

Now, let's consider the following requirement:

"The speed_disp_fault shall be set to TRUE, when the following conditions occur:

- a. speed_disp is less than Zero (0) <unit> OR*
- b. speed_sig_fault is TRUE"*

The correct pseudo code implementation for the above requirement would look like this:

```
if(speed_disp<0) OR (speed_sig_fault == 1) then
    speed_disp_fault = True;
end if;
```

Test Cases for 100% Statement Coverage

To achieve **100% statement coverage**, we can use the following inputs:

Input-1 speed_disp	Input-2 speed_sig_fault	Output speed_disp_fault
-2	1	True

However, achieving 100% statement coverage does not guarantee the logic is correctly implemented in software.

Drawbacks of Statement Coverage:

The software engineer could make a mistake in interpreting the requirement and use the “AND” gate instead of the “OR” condition. But the above test case will not be able to find this problem, even though we will achieve 100% statement coverage.

Therefore, in this case, even with 100% statement coverage does not confirm the absence of bugs.

What is Decision Coverage?

The decision coverage means that:

“Every point of entry and exit in the program has been invoked at least once and every decision in the program has taken on all possible outcomes at least once.”

As you can see the above definition, to achieve **100% decision coverage**, we need to have two test cases mainly:

- a. Exercise TRUE outcome of the decision
- b. Exercise FALSE outcome of the decision

When, we execute these test cases, “every point of entry and exit in the program” will also get cover.

Example of Decision Coverage:

Now, let's consider the following requirement:

"The speed_disp_fault shall be set to TRUE, when the following conditions occur:

- a. speed_disp is less than Zero (0) <unit> OR*
- b. speed_sig_fault is TRUE"*

The correct pseudo code implementation for the above requirement would look like this:

```
if(speed_disp<0) OR (speed_sig_fault == 1) then
    speed_disp_fault = True;
end if;
```

Test Cases for 100% Decision Coverage:

There are two conditions in this example:

- 1. Condition-1: speed_disp<0
- 2. Condition-2: speed_sig_fault==1

To achieve **100% decision coverage**, we need two test cases:

- 1. Condition-1 = True; Condition-2 = False; Decision Outcome = True
- 2. Condition-1 = False; Condition-2 = False; Decision Outcome = False

Condition-1	Condition-2	Output
True	False	True
False	False	False

We are getting into the details of actual test cases here, since the actual test cases not required here to understand the concept of Decision coverage.

Drawbacks of Decision Coverage:

So, with these two test cases, we can achieve 100% decision coverage, but still, we cannot show the effect of Condition-2 on the decision outcome.

That means if the software engineer misinterprets the requirement and implements the code as follows:

```
If (speed_disp<0) then
    speed_disp_fault = True;
end if;
```

we will not be able to find out the error with these test cases even with 100% decision coverage.

What is Condition Coverage?

The condition coverage means that:

“Every condition in a decision has taken on all possible outcomes at least once.”

That means, to achieve 100% condition coverage, we need to make sure that each condition in the decision exercise both True and False outcome. But, it is not mandatory to have all possible outcomes for the decision.

Example of Condition Coverage:

Now, again, let's consider the same piece of requirement and code:

“The speed_disp_fault shall be set to TRUE, when the

following conditions occur:

- a. speed_disp is less than Zero (0) <unit> OR*
- b. speed_sig_fault is TRUE"*

The correct pseudo code implementation for the above requirement would look like this:

```
if(speed_disp<0) OR (speed_sig_fault == 1) then
    speed_disp_fault = True;
end if;
```

As we have seen previously, there are two conditions here in the above example:

- Condition-1: speed_disp<0
- Condition-2: speed_sig_fault==1

Test Cases for 100% Decision Coverage:

To achieve **100% condition coverage**, we need two test cases:

- Condition-1 = True; Condition-2 = False; Decision Outcome = True
- Condition-1 = False; Condition-2 = True; Decision Outcome = True

Condition-1	Condition-2	Decision Outcome
True	False	True
False	True	True

So, with these two test cases, we can achieve **100% condition coverage**.

Drawbacks of Decision Coverage:

But we can easily observe here, in both the test cases, **the decision outcome is always True**. The False outcome of the decision is not being exercised here.

What is condition/decision coverage?

The condition/decision coverage is nothing but the combination of both the requirements for decision coverage and condition coverage.

So, we need the following consider both cases as following to achieve **100% condition/decision coverage**:

1. Every decision in the program has taken on all possible outcomes at least once.
(Requirement from decision coverage)
2. Every condition in a decision has taken on all possible outcomes at least once
(Requirement from condition coverage)

Example of Condition/Decision Coverage:

Now, again, let's consider the same requirement from our previous example:

"The speed_disp_fault shall be set to TRUE, when the following conditions occur:
a. speed_disp is less than Zero (0) <unit> OR
b. speed_sig_fault is TRUE"

The correct pseudo code implementation for the above requirement would look like this:

```
if(speed_disp<0) OR (speed_sig_fault == 1) then
    speed_disp_fault = True;
end if;
```

Test Cases for 100% Decision Coverage:

We have already understood the meaning of condition in the context of this example. There are two conditions in the above example:

1. Condition-1: `speed_disp<0`
2. Condition-2: `speed_sig_fault==1`

The decision is: `(speed_disp<0) OR (speed_sig_fault == 1)`

To achieve **100% condition/decision coverage**, we need two test cases:

1. Condition-1 = True; Condition-2 = True; Decision Outcome = True
2. Condition-1 = False; Condition-2 = False; Decision Outcome = False

Condition-1	Condition-2	Decision Outcome
True	True	True
False	False	False

Drawbacks of Decision Coverage:

So, with these two test cases, we can achieve 100% condition/decision coverage.

But, these test cases cannot distinguish the correct expression (Condition-1 or Condition-2) from the expression (Condition-1 and Condition-2).

That means if the software engineer misinterprets the requirement and implements the code as follows:

```
If (speed_disp<0) AND (speed_sig_fault ==1) then
    speed_disp_fault = True;
```

```
end if;
```

The if condition is written as "(speed_disp<0) **AND** (speed_sig_fault ==1)" instead of "(speed_disp<0) **OR** (speed_sig_fault ==1)".

But, we will not be able to find out the error with these test cases even with 100% condition/decision coverage.

What is Modified condition/decision coverage?

The MCDC or MC/DC or Modified condition/decision coverage is the enhanced version of condition/decision coverage. All the requirements for condition/decision coverage still hold good, but the extra requirement is to show that each condition can affect the outcome of the decision independently.

Example of Modified Condition/Decision Coverage:

Now, let's go back to our sample requirement:

"The speed_disp_fault shall be set to TRUE, when the following conditions occur:

- a. speed_disp is less than Zero (0) <unit> OR*
- b. speed_sig_fault is TRUE"*

The correct pseudo code implementation for the above requirement would look like this:

```
if(speed_disp<0) OR (speed_sig_fault == 1) then
    speed_disp_fault = True;
end if;
```

Test Cases for 100% Decision Coverage:

We have already identified the condition and decision for this example before.

Now, let's see the test cases that are required to cover 100% MC/DC:

1. Condition-1 = True; Condition-2 = False; Decision Outcome = True
2. Condition-1 = False; Condition-2 = True; Decision Outcome = True
3. Condition-1 = False; Condition-2 = False; Decision Outcome = False

Condition-1	Condition-2	Decision Outcome
True	False	True
False	True	True
False	False	False

These three test cases will help us to achieve 100% MC/DC coverage and also get rid of the issues encountered in condition coverage or in decision coverage.

Note: This is one of the popular **MCDC Interview Questions**.

Achievements of Modified Condition/Decision Coverage:

The Modified Condition/Decision Coverage or MCDC or MC/DC overcomes all the drawbacks that we have seen in case of statement coverage, decision coverage, condition coverage, condition/decision coverage.

Apparently, writing the MC/DC scenarios need more attention and thoughtful selection of test cases.

As per DO 178C, Level-A software must achieve 100% MC/DC coverage.

What is Multiple Condition coverage?

The multiple condition coverage simply requires exercising each possible combination of inputs to a decision at least once.

This is indeed the exhaustive testing of the input combinations to a decision.

The multiple condition coverage requires a lot of test cases and requires a lot of time to execute the test cases. Therefore, normally we avoid following this coverage.

The DO 178C does not mandate following (talk about) this type of coverage criteria.

Can you write the test cases to achieve MC/DC for C=(A and B)?

Before we jump into writing the MC/DC scenarios, let's see the truth table for "C = A and B" expression.

Input-1 A	Input-2 B	Output C
True	True	True
True	False	False
False	True	False
False	False	False

Truth Table for: C=A and B

We can clearly observe, AND gate is responsive to false input; that means if we consider a base condition where all the inputs are set to True and the AND gate output is True,

changing a single input to False thereafter will affect/change the final output.

Therefore, we follow the following strategy to achieve MC/DC or MCDC for n-input AND gate:

1. Set all the inputs to True so that the outcome for and gate is True
2. Now, one by one, set each condition to False making the output to False. This shows the independent effect of every condition on the decision's outcome. This step would produce a total of "n" test cases.

So, here are the test cases for AND gate to achieve 100% MC/DC:

Input-1 A	Input-2 B	Output C
True	True	True
True	False	False
False	True	False

MCDC cases for C=A and B

Note: This is one of the most popular **MCDC Interview Questions** for fresh graduates.

Can you write the test cases to achieve MC/DC for the following expression: $D=(A \text{ and } B \text{ and } C)$?

We have explained the two-input AND gate MC/DC in the previous example. First of all, let's see the truth table for "D = A and B and C" expression:

Input-1 A	Input-2 B	Input-3 C	Output D
False	False	False	False
False	False	True	False
False	True	False	False
False	True	True	False
True	False	False	False
True	False	True	False
True	True	False	False
True	True	True	True

Truth Table for: D=A and B and C

Now, follow these steps to write the MDC cases for the above expression:

1. Set all the inputs to True so that the outcome for and gate is True
2. Now, one by one, set each condition to False making the output to False. This shows the independent effect of every condition on the decision's outcome. This step would produce a total of "n" test cases.

For, our example, we will first set all the inputs i.e. A, B, and C to True (A=True, B=True, C=True) and the Decision outcome would be True.

Then we will set the input-1 : A to False and keep all other inputs value as it was in base

case (A=False, B=True, C=True). The outcome is False in this case. So, clearly changing the input value independently affecting the decision outcome.

Now, let's see test cases to achieve 100% MC/DC for three-input AND gate:

Input-1 A	Input-2 B	Input-3 C	Output D
True	True	True	True
False	True	True	False
True	False	True	False
True	True	False	False

MCDC cases for: D=A and B and C

Therefore, if you have n input AND gate, you will require (n+1) test case to achieve 100% MC/DC.

Note: This is one of the most commonly asked **MCDC Interview Questions**.

Can you write the test cases to achieve MC/DC for the following expression: C=(A or B)?

Before we jump into writing the MC/DC scenarios, let's see the **truth table for C=(A or B)** expression.

Input-1 A	Input-2 B	Output C
False	False	False
False	True	True
True	False	True
True	True	True

Truth Table for C=A or B

We can clearly observe, OR gate is responsive to True input. That means if we consider a base condition where all the inputs are set to False and the OR gate output is False, changing a single input to True thereafter will affect/change the final output.

Therefore, we follow the following strategy to achieve 100% MC/DC:

1. Set all the inputs to False so that the outcome for "OR" gate is False
2. Now, one by one, set each condition to True making the output to True. This shows the independent effect of every condition on the decision's outcome. This step would produce a total of "n" test cases.

So, here are the test cases for OR gate to achieve 100% MC/DC:

Input-1 A	Input-2 B	Output C
False	False	False
True	False	True

Input-1 A	Input-2 B	Output C
False	True	True

MCDC cases for: C=A OR B

Can you write the test cases to achieve MC/DC for D=(A or B or C)?

We have explained the two-input OR gate MC/DC in the previous example. First of all, let's see the truth table for "D = A or B or C" expression:

Input-1 A	Input-2 B	Input-3 C	Output D
False	False	False	False
False	False	True	True
False	True	False	True
False	True	True	True
True	False	False	True
True	False	True	True
True	True	False	True

True	True	True	True
------	------	------	------

Truth Table for: D = A or B or C

Now, follow these steps to achieve 100% MC/DC cases for three-input OR gate:

- 1. Set all the inputs to False so that the outcome for "OR" gate is False
- 2. Now, one by one, set each condition to True making the output to True. This shows the independent effect of every condition on the decision's outcome. This step would produce a total of "n" test cases.

Test Case Number	Input A	Input B	Input C	Outcome D
1	False	False	False	False
2	True	False	False	True
3	False	True	False	True
4	False	False	True	True

Therefore, if you have n input OR gate, you will require (n+1) test cases [One test case for base case and other test cases for n-inputs] to achieve 100% MC/DC.

Note: This is one of the most commonly asked MCDC Interview Questions.

Can you write the test cases to achieve MC/DC for C=(A xor B)?

Before we jump into writing the MC/DC scenarios, let's see the truth table for (A xor B)

expression.

Input-1 A	Input-2 B	Output C
True	True	False
True	False	True
False	true	True
False	False	False

Now, let's see how to achieve 100% MC/DC cases for three-input XOR gate:

Input-1 A	Input-2 B	Output C
True	True	False
True	False	True
False	True	True

MCDC cases for: $C = A \text{ xor } B$

Can you write the test cases to achieve MC/DC for the expression: $B = \text{not } A$?

To achieve 100% MC/DC cases for NOT gate, we need the following cases:

Input-1 A	Output B
True	False
False	True

Can you explain how to achieve MC/DC for comparators?

There are several comparators can be considered:

1. Greater than
2. Greater than equal to
3. Less than equal to
4. Less than
5. Equal to
6. Not equal to

Let's see the previous example that we have seen before in this article:

"The speed_disp_fault shall be set to TRUE, when the following conditions occur:

- a. speed_disp is less than Zero (0) <unit> OR*
- b. speed_sig_fault is TRUE"*

The correct pseudo code implementation for the above requirement would look like this:

```
if(speed_disp<0) OR (speed_sig_fault == 1) then
    speed_disp_fault = True;
```

```
end if;
```

There are two comparators that are being used in the above example i.e. less than (<) and equal to (==).

For less than (<) operator, we need the following test cases to achieve MC/DC:

1. Set speed_disp to a value(-5) which is below 0
2. Set speed_disp to a value(3) which is above 0

Here, the comparison point is 0.

How to achieve MC/DC for if-else statement?

The if-else control structure is used to achieve the program execution control based on certain conditions.

The minimum test cases required for if-else condition:

1. Set the inputs such that True-path executes
2. Set the inputs such that the False-path executes
3. Exercise MC/DC cases for any other logical gates in the decision

How to achieve MC/DC for while loop?

The while loop control structure is used to execute the repetitive tasks in the program.

The minimum test cases required for the while loop:

1. Set the inputs such that the statement inside the loop executes
 2. Set the inputs such that the loop exits
 3. Exercise MC/DC cases for any other logical gates in the decision
-

How to achieve MC/DC for Bit-wise operations?

Using bit-wise operation in embedded system software development is very common. There are several instances where bit-wise operations are used especially in the low-level software which interacts with the hardware.

In the case of bit-wise operations, each individual bit represents a condition. To achieve MC/DC for Bit-wise operations, you need to ensure to achieve the MC/DC for each individual bit.

For example, if we have an integer variable of 32-bits, every 32 bits are treated as different conditions in the context of DO178C.

Have you used any coverage analysis automation tool?

Yes. There are several structural coverage analysis tools such as LDRA, VectorCast etc.

How do you ensure that the instrumentation of code did not inject an additional bug in the software?

Normally, most of the structural coverage analysis tools instrument the source code to inject a mechanism to monitor the code execution and calculate the structural coverage percentage.

To prove that the automation tool (such as LDRA, VectorCast etc) instrumented the source code without injecting any additional error, we need to execute the same set of test cases on both the original code as well as instrumented code.

Note: These automation instrument the code to track which part of the code got exercised by the test inputs and provide us the coverage report. The coverage that we get from these tools is based on the instrumented code. That means, the instrumented code meets the coverage – MC/DC or statement coverage. We are assuming that the un-instrumented code would also achieve the same coverage percentage.

Note: This is one of the popular **MCDC Interview Questions** for experienced professionals.

What are the commonly misunderstood facts about MC/DC?

Here are the most commonly misunderstood facts about MC/DC:

1. Meeting MC/DC is more important than requirements-based testing
 2. Trying to find functional errors using MC/DC cases
 3. Relying on MC/DC to find problems
-

What is the Independent effect in MC/DC?

If a condition alone can determine the outcome of the decision, we say that the condition independently affects the decision outcome.

To show the independent effect in MC/DC, we need two cases:

1. Base case
2. Change one condition and show the decision's output changes only because of the changed condition.

Note: This is one of the most commonly asked **MCDC Interview Questions**.

Can you explain Masking in the context of MC/DC?

Masking means that a specific value of an input to a logical expression hides the effect of other inputs on the logical expression.

For example, A True input to an OR gate will mask all other inputs since the OR gate is True sensitive.

Similarly, a False input to an AND gate will mask all other inputs since AND gate is False

sensitive.

Can you write the MC/DC cases for the following expression:

Z := (A and not B) or (C xor D)?

The MC/DC cases for the above expression:

Input-1 A	Input-2 B	Input-3 C	Input-4 D	Output Z
True	True	False	False	False
True	False	False	False	True
False	False	False	False	False
False	False	True	False	True
False	False	False	True	True

What is Coverage? Why Coverage?

When we say “coverage” in the context of DO178C, it means “measure” not “testing”. So, Coverage is not related to the testing activity.

Furthermore, there are two types of coverage:

1. Requirements coverage

2. Software structural coverage

Requirements coverage analysis indicates how well the requirements-based test cases verified the requirements.

On the other hand, the software structural coverage indicates how well the software code structure got executed/covered by the requirement-based test cases. The structural coverage is used to showcase the absence of unintended functions in the software.

What is the difference between structural coverage analysis and structural testing?

The structural coverage analysis is different than structural testing.

Structural coverage indicates how well the software code structure got executed/covered by the requirement-based test cases.

Structural testing means writing test scenarios based on software code. The structural testing cannot find non-implementation of requirements since the basis for the structural test scenarios is software code (not requirement).

Why MC/DC? Why not do multiple-condition coverage?

Multiple condition coverage means that we need to exercise all possible combinations of inputs for each decision to ensure that the correct decision outcome is reached in all possible cases.

Therefore, in case of multiple condition coverage, for n -inputs, we need 2^n test cases. When n is a small number, doing multiple-condition coverage seems reasonable. However, for a larger n , running 2^n test cases could be time-consuming and maybe not possible at all.

For example, in avionics software, there are cases with 20-40 inputs in a Boolean expression. Let's say, we have 30 inputs in a Boolean expression and we want to perform

multiple-condition coverage.

For a 30-input Boolean expression, we need 2^{30} test cases = 1,07,37,41,824.

Now, let's assume we can execute one test case in one second. So, we need 1,07,37,41,824 seconds ~ **34 years!**

So, multiple-condition coverage is impractical. That's why we normally perform MC/DC.

Note: This is one of the most commonly asked **MCDC Interview Questions**.

Conclusion:

DO178B or DO178C is a very important guideline document for any software that is being developed for civil aircraft. Achieving 100% MC/DC is absolutely required for any Level-A software as per [DO178C](#).

I have tried to cover most of the MC/DC or MCDC questions that you could eventually face during the interview.

However, there could be areas that I have missed to cover. Please feel free to comment below, if you think anything else could be added here in the context of MCDC interview questions.

Note: I have also compiled all the DO178C tables in one document along with the explanation and created a pdf copy of it.

If you want to get the FREE pdf copy, please comment your email ID in the below comment box, and I will send you the copy. This document would definitely give you a far better understanding of the [DO178C](#) objectives.

Hopefully, this article could help you!

Good luck with your interview.

This post was published by Admin.

Email: admin@TheCloudStrap.Com



Related Posts:

1. [50+ DO178C Interview Questions | DO178C Standards](#)
2. [SOI Audit Interview Questions](#)
3. [DO-254 Interview Questions](#)
4. [Top 15+ Avionics Verification and Validation Interview Questions](#)
5. [Avionics Verification and Validation Interview Questions](#)
6. [Top 45 C++ Interview Questions For Freshers and Experienced Professionals](#)
7. [Demystifying DO-178C: A Comprehensive Guide to Software Considerations in Airborne Systems Certification](#)
8. [DO-178C Objectives List | Must Read](#)
9. [Decoding DO-178C Software Levels: A Comprehensive Guide](#)
10. [Exploring the Use of UAVs in Wildlife Conservation and Research](#)

◀ [DO-178C PSAC](#)

[DO-254 Interview Questions](#) ▶

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *


Name *

Email *


Website

☐ Save my name, email, and website in this browser for the next time I comment.

☐ I'm not a robot


reCAPTCHA
[Privacy](#) - [Terms](#)

☐ Notify me of new posts by email.

All new comments 

Notify me of followup comments via e-mail. You can also subscribe without commenting.

Post Comment

[About Us](#)

[Terms & Conditions](#)

[Privacy Policy](#)

[Write For Us](#)

[Contact Us](#)

Copyright © 2023 TheCloudStrap.Com All rights reserved.