MC/DC

- MC/DC is defined in DO-178B/ED-12B, -"Software Considerations in Airborne Systems and Equipment Certification", dated December 1, 1992.

- Definition of MC/DC:

(1) Every point of entry and exit in the program has been invoked at least once

(2) Every condition in a decision in the program has taken all possible outcomes at least once

(3) Every decision in the program has taken all possible outcomes at least once

(4) 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 varying just that condition while holding fixed all other possible conditions

MC/DC

- MC/DC criteria is stronger than Condition/Decision
- 100% MC/DC will guarantee that each simple condition will not be masked by the other conditions.
- Consider the following decision: $x < 0$ OR $y < 0$
- If $x = -1$, then $x < 0$ is true and it will mask the condition $y < 0$, since no matter $y < 0$ is true or not, the whole decision will be evaluated to true.
- 100% MC/DC guarantees 100% C/D

Difference Between Coverage Criterias

Table 1. Types of Structural Coverage

Coverage Criteria	Statement Coverage	Decision Coverage	Condition Coverage	Condition/ Decision Coverage	MC/DC	Multiple Condition Coverage
Every point of entry and exit in the program has been invoked at least once		•	•	•	•	•
Every statement in the program has been invoked at least once	•					
Every decision in the program has taken all possible outcomes at least once		•		•	•	•
Every condition in a decision in the program has taken all possible outcomes at least once			•	•	•	•
Every condition in a decision has been shown to independently affect that decision's outcome					•	•[8]
Every combination of condition outcomes within a decision has been invoked at least once						•

Hayhurst, Kelly; Veerhusen, Dan; Chilenski, John; Rierson, Leanna (May 2001). "A Practical Tutorial on Modified Condition/ Decision Coverage". NASA.

MC/DC Example

- Considering the following code:

```
int isReadyToTakeOff(int a, int b, int c, int d)
{
  if(((a == 1) ||(b == 1)) && ((c == 1) || (d == 1)))
          return 1; else return 0;
}
```

$$T_1 = \begin{Bmatrix} t_1 : < a = 0, b = 1, c = 1, d = 1 > \\ t_2 : < a = 0, b = 0, c = 0, d = 1 > \\ t_3 : < a = 1, b = 0, c = 0, d = 0 > \end{Bmatrix} \qquad 100\%\ C/D$$

$$T_2 = \begin{Bmatrix} t_1 : < a = 1, b = 0, c = 1, d = 0 > \\ t_2 : < a = 1, b = 0, c = 0, d = 1 > \\ t_3 : < a = 0, b = 1, c = 0, d = 1 > \\ t_4 : < a = 1, b = 0, c = 0, d = 0 > \\ t_5 : < a = 0, b = 0, c = 0, d = 1 > \end{Bmatrix} \qquad 100\%\ MC/DC$$

MC/DC Example

$$T_2 = \begin{Bmatrix} t_1: < a = 1, b = 0, c = 1, d = 0 > \\ t_2: < a = 1, b = 0, c = 0, d = 1 > \\ t_3: < a = 0, b = 1, c = 0, d = 1 > \\ t_4: < a = 1, b = 0, c = 0, d = 0 > \\ t_5: < a = 0, b = 0, c = 0, d = 1 > \end{Bmatrix} \quad 100\%\ MC/DC$$

$t_2 + t_5$ shows the effect of $a = 1$;

- Values of b, c, d in t_2 and t_5 are same.
- when $a = 1$, t_2 → true;
- when $a = 0$, t_5 → false;

$t_3 + t_5$ shows the effect of $b = 1$;

- Values of a, c, d in t_3 and t_5 are same.
- when $b = 1$, t_3 → true;
- when $b = 0$, t_5 → false;

$t_1 + t_4$ shows the effect of $c = 1$;

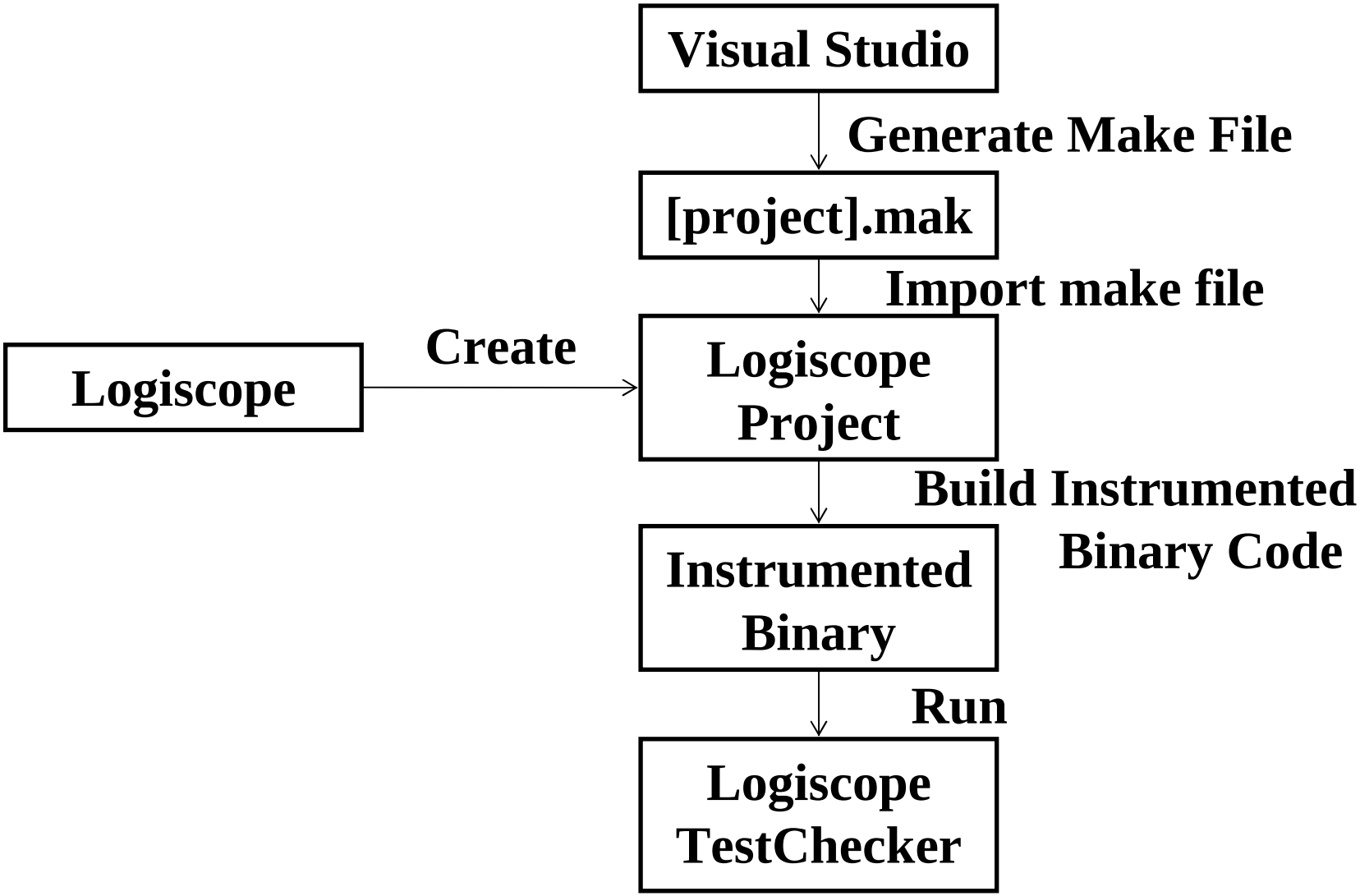
- Values of a, b, d in t_1 and t_4 are same.
- when $c = 1$, t_1 → true;
- when $c = 0$, t_4 → false;

$t_2 + t_4$ shows the effect of $d = 1$;

- Values of a, b, c in t_2 and t_4 are same.
- when $d = 1$, t_2 → true;
- when $d = 0$, t_4 → false;

MCDC Demo
Using Logiscope TestChecker

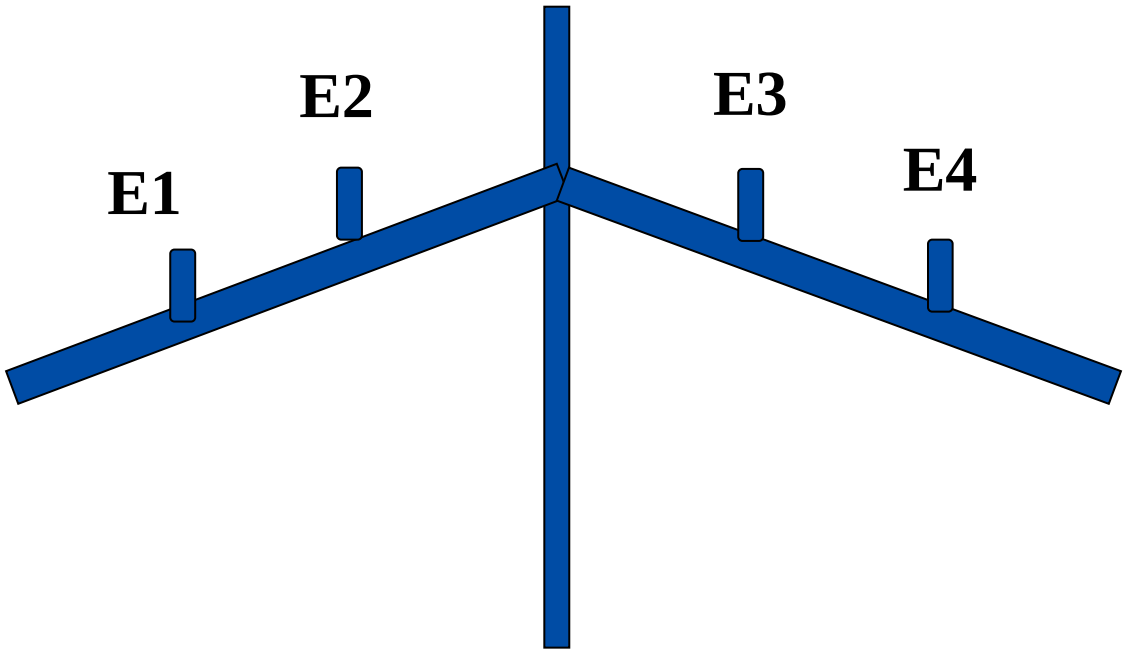
How Does Logiscope TestChecker Work?



Requirement

The self-check module will check the status of 4 engines of a airplane, then return if the airplane can take off.

- The airplane shall be able to take off with at least one of the engine1 and engine2 on, and at least one of the engine 3 and engine 4 on.



Source Code

```
int isReadyToTakeOff(int engine1, int engine2, int engine3, int engine4)
{
        if(((engine1 == 1) ||(engine2 == 1))
        && ((engine3 == 1) || (engine4 == 1)))
        {
                return 1;
        }
        else
        {
                return 0;
        }
}
```

Source Code

100% C/D coverage

Test cases	engine 1	engine 2	engine 3	engine 4	Result	Oracle
1	0	1	1	0	1	1
2	0	0	0	1	0	0
3	1	0	0	0	0	0

Requirement

The self-check module will check the status of 4 engines of a airplane, then return if airplane can take off.

- The airplane shall be able to take off with at least one of the engine1 and engine2 on, and at least one of the engine3 and engine4 on.

- New requirement:
- The airplane shall not be able to take off with engine3 off.

Source Code

Test cases	engine 1	engine 2	engine 3	engine 4	Result	Oracle
1	0	1	1	0	1	1
2	0	0	0	1	0	0
3	1	0	0	0	0	0

100% *C/D*

- Although these test cases achieved 100% C/D coverage, bug is not revealed, since with respect to all test cases, engine3 == 0 can not directly effect the decision's outcomes.
- In another word, with respect to all test cases, engine3 == 0 is masked by other conditions.

Source Code

Test cases	engine 1	engine 2	engine 3	engine 4	Result	Oracle
1	1	0	1	0	1	1
2	1	0	0	1	1	0
3	0	1	0	1	1	0
4	1	0	0	0	0	0
5	0	0	0	1	0	0

100% *MC/DC*

- Test cases t_2 and t_3 will reveal the bug.