Modified Condition Decision Coverage: A Hardware Verification Perspective

Mohamed A. Salem  
Computer Science Department, Faculty of Engineering  
University of Bristol, UK  
Email:mohamed.salem@bristol.ac.uk

Kerstin I. Eder  
Computer Science Department, Faculty of Engineering  
University of Bristol, UK  
Email:kerstin.eder@bristol.ac.uk

Abstract—Verification is a critical phase of the development cycle. It confirms the compliance of a design implementation with its functional specification. Coverage measures the progress of the verification plan. Structural coverage determines the code exercised by the functional tests. Modified Condition Decision Coverage (MC/DC) is a structural coverage type. This paper compiles a comprehensive overview of established MC/DC conventions, and develops novel MC/DC insights through conduction of experimental study for MC/DC in hardware verification. It provides a generic MC/DC overview while explaining MC/DC types, and criteria of MC/DC validation in the software domain. It introduces the motivation for adoption of MC/DC as a potential structural coverage type for hardware verification. The paper presents the experimental evaluation conducted over a diverse base of logic combinations. The introduced experimental results inferred distinct MC/DC insights. These insights present novel MC/DC aspects that optimize the minimal MC/DC coverage requirements, defines MC/DC compositionality concepts, and provide RTL design guidelines for MC/DC fulfillment.

I. INTRODUCTION

The typical product development flow starts off from a functional specification document that states the product functional requirements. Designers interpret the specification in terms of hardware descriptions. The objective of functional verification is to confirm that a hardware implementation satisfies the requirements defined in its specification. Coverage is a metric to measure the progress of the verification plan. Coverage can be classified as either functional, that is related to the design requirements, or structural, that is related to the design implementation. Modified Condition Decision Coverage (MC/DC) is a type of structural coverage. MC/DC is a key requirement for safety critical software certification [1] [2].

This paper gives an overview of MC/DC coverage, and the currently employed techniques for achieving MC/DC coverage. It presents a detailed experimental study of the application of MC/DC in the hardware domain using fundamental concepts of hardware logic analysis. The current MC/DC methods address the checks of MC/DC versus the functional requirements tests. The experimental evaluation compares the observations versus the conventional aspects of MC/DC in terms of minimal number of test cases required to achieve MC/DC coverage, the notion of compositionality of MC/DC coverage, and the influence of RTL code structures on MC/DC results.

We conclude with new insights for MC/DC coverage that can be applied to software and hardware verification. The paper shows that MC/DC coverage can be achieved by an optimized reduced minimal number of tests, lower than the currently known minimal requirements. The MC/DC compositionality has been clarified with insights that dispel the current conventions in regards of MC/DC achievement and compositionality requirements. The paper provides beneficial RTL coding guidelines to be adopted while addressing design for MC/DC, and the choice between inlined and non-inlined RTL code representations.

II. DEFINITIONS AND NOTATIONS

The following conventions are used throughout the paper: Boolean operators are denoted by bold italics and, or, xor, not. Boolean conditions are denoted by bold capital letters A, B, C. Truth values are written as either false or true, or F or T. A test case for a Boolean function with n inputs is denoted by V \((V_1 V_2 ... V_n)\), where \(V_i\) is F or T. Conventional logic gate symbols are used to represent Boolean operators in graphical representations. Code segments are written in SystemVerilog unless stated otherwise.

III. MC/DC BACKGROUND

This section will provide an overview of structural coverage types, MC/DC applications in software certification, types of MC/DC, and states the current convention for the minimal requirements of MC/DC fulfillment.

A. Structural Coverage Types: MC/DC Context

Statement coverage, also known as line coverage, requires that each statement in the code has been invoked at least once. This is typically regarded to be the minimum target for code coverage. The inputs to a Boolean expression are defined as the conditions on which the decision will be evaluated.

Condition coverage requires that each condition in a decision takes on all possible values at least once. Note that it does not require that the decision take on all possible outcomes at least once.

Decision coverage requires the decision value to toggle between true and false. This is also known as branch coverage [3]. For example, for the decision (A or B) test cases TF and FF toggle the decision outcome between true and false. However the effect of B is not tested; that is, these tests can not distinguish between the decision (A or B) and the decision A.

Condition/Decision coverage requires that each condition toggles its value between true and false and that the decision toggles its values as well. For (A or B), tests (TT,FF) meet this criterion; note that these tests do not distinguish the expression from (A and B).

Multiple Condition/Decision coverage requires exhaustive testing of all the possible input conditions combinations to a decision. This coverage type has high cost due to the exponential growth of the required coverage set tests versus the number of input conditions. A decision statement with N input conditions requires \(2^N\) coverage set tests.
B. MC/DC Definition and Types

MC/DC is formally defined when every point of entry and exit in the software program or RTL source code has been invoked at least once, every condition in the program has taken all possible outcomes at least once, and each condition in a decision has been shown to independently affect a decision outcome. The independence requirement ensures that the effect of each condition is tested relative to the other conditions [2].

- **Unique-Cause MC/DC Type**
  The independence requirement of each condition on the decision outcome is achieved by changing only the condition of interest while holding all other conditions fixed.

- **Masking MC/DC Type**
  Masking defines the concept that specific inputs to a logic function can mask the effect of other inputs to the function. These specific input values for that logic operators dominate the behavior. For example, a false input to an and gate masks all other inputs, and likewise a true input to an or gate masks all other inputs. Masking MC/DC allows changing masked input conditions that have no influence on the decision output, while checking the independent effect of the condition of interest. However, this requires logic analysis of the decision expression [4].

- **NASA MC/DC Determination approach**
  NASA has developed an approach to determine the MC/DC coverage fulfillment of software source code in the context of requirements-based testing as per DO-178B [1]. The details of this approach are given in [2].

C. MC/DC Independence Pairs

A condition independently affects a decision outcome if that condition alone can determine the value of the decision outcome. Two test cases that show the independent effect of a condition within a decision are referred to as an independence pair [5]. For example, test vectors (FT, TT) form an independence pair for a 2-input and gate due to the independent effect of the changing condition on the output value.

D. Conventional MC/DC Minimal Requirements

The survey of the MC/DC literature has shown that there is a convention for the minimum number of tests required for MC/DC fulfillment for a decision that has \(N\) conditions. This minimal number is \((N + 1)\) tests.

There has been no proof shown for this statement other than it is intuitively inferred from the unique-cause MC/DC notion. Indeed, it is easy to see that under unique-cause MC/DC, from an initial state, the minimal number of tests needed to achieve unique-cause requirements in terms of one condition change while all others are held fixed will be \((N + 1)\).

The paper will show that there is an optimized reduced minimal number of tests for MC/DC in terms of Masking MC/DC as will be illustrated in the experimental work Section VI. It is worth noting that MC/DC is widely applied to control flow logic that practically does not have a huge number of conditions [6].

E. MC/DC in the Software Domain

MC/DC has been used successfully in safety critical software verification. One of the main objectives of DO-178B [1] that addresses level-A software certification for aviation systems, dictates that tests achieve MC/DC requirements for the program code structure.

IV. CONTROLLABILITY AND OBSERVABILITY

Controllability represents the ability to control input values of a certain logic operator in an expression. For example, for the logic operation \((C \text{ xor } D)\), controllability represents the ability to control the values of the inputs \(C\) and \(D\) in order to control the output value of the xor logic operator.

Observability is the state of being able to propagate a certain logic operator output value to a certain node in the logic structure. Figure 1 shows an example of a logic structure and illustrates how to observe the input of concern by setting the values of the control inputs such that it can propagate to the output node [7]. MC/DC coverage is by default addressing controllability and observability [8].

![Logic structure observability example](image1)

Fig. 1. Logic structure observability example

V. MOTIVATION

MC/DC has been successfully applied in the software domain. The RTCA/DO-178B [1] avionics software certification document requires that the level A software functional tests achieve MC/DC coverage. The acceptance in the software domain motivates an introduction of MC/DC to the hardware domain.

Controllability and observability are known engineering aspects used mainly in logic circuits analysis in hardware designs. They are the basis of MC/DC analysis in software, and hence also a conventional MC/DC analytical basis in the hardware domain. Hardware description languages (HDL); like Verilog, SystemVerilog, VHDL; are used to describe hardware designs. In principle, hardware designs are developed using constructs akin to those found in programming languages. Code coverage for software programs is therefore identical to that of RTL designs.

Hence, it would be worth studying MC/DC coverage from a hardware perspective and logic analysis. Statement coverage is not enough, condition/decision coverage includes possible branches but lacks the independent effect of each condition on the decision output. Multiple condition decision coverage (full expression) is impractical due to state space explosion. MC/DC provides independent effect of each condition on the decision output and requires a test suite far
smaller than \(2^N\). MC/DC would be an efficient and economical coverage method for hardware designs. The MC/DC success in the software domain, the greater emphasis on controllability and observability for hardware logic analysis, the use of HDL for hardware designs, and the requirement of a powerful, and cost effective structural coverage technique are the potential motivational factors toward conduction of this research.

### VI. EXPERIMENTAL WORK

This section presents samples from the hardware experimental investigation of the properties of MC/DC into various combinations of RTL, design statements from industrial designs. The selection criteria provide statements that describe a decision that has multiple conditions in the format of Boolean logic expressions using the basic logic operators. The samples cover a good variety in terms of unimodal, and bimodal expression types.

The experimental results present new insights into the minimum number of test cases required to achieve MC/DC coverage. MC/DC compositionality, and the RTL coding options. Table I presents a sample of the selected expressions, and this section proceeds with sample case studies that demonstrate the novel MC/DC insights noted on the full experimental set.

#### A. Experiments Setup

1) **Experimental test bench:** The experiments have been conducted under QuestaSim version 6.6 [9] using Linux. Figure 2 represents code fragment of the generic design module (top.v) which will be used in all experiments. The generic code will be customized to apply the test suite stimulus for each experiment. The composite decision statement will be expressed by a high level function \(f\), and sub-function statements \((x)\) and \((y)\), and an equivalent decision statement \((z)\) in terms of \((x)\), and \((y)\). These logically equivalent variants of the decision statement representation facilitate the study of compositionality, and also of RTL coding styles’ influence on MC/DC coverage.

2) **Experimental Compile/Simulate script:** Figure 3 shows the commands to compile the (top.v) verilog module, and how to simulate it while informing the tool to investigate the observability of \(true\) and \(false\) values of each input using the FEC (Focused Expression Coverage) [10] code coverage under Questa that fulfills the MC/DC requirements.

---

**TABLE I. Sample MC/DC Experimental Logic Expressions**

<table>
<thead>
<tr>
<th>Function</th>
<th>Type</th>
<th>Min. Tests</th>
</tr>
</thead>
<tbody>
<tr>
<td>(F = (A \text{ and } B ) \text{ or } (C \text{ and } D))</td>
<td>unimodal</td>
<td>4</td>
</tr>
<tr>
<td>(F = (A \text{ and } B ) \text{ or } (A \text{ and } C) \text{ or } (B \text{ and } C))</td>
<td>unimodal</td>
<td>4</td>
</tr>
<tr>
<td>(F = [(A \text{ and } B ) \text{ or } (A \text{ and } C)] \text{ xor } [(A \text{ and } B ) \text{ or } (C \text{ and } D)])</td>
<td>bimodal</td>
<td>4</td>
</tr>
<tr>
<td>(F = [(A \text{ and } B ) \text{ or } (\text{not } A) \text{ and } (\text{not } B)])</td>
<td>bimodal</td>
<td>3</td>
</tr>
<tr>
<td>(F = (A \text{ and } B \text{ or } C \text{ and } D))</td>
<td>bimodal</td>
<td>3</td>
</tr>
<tr>
<td>(F = (A \text{ or } B \text{ or } C \text{ or } D))</td>
<td>unimodal</td>
<td>5</td>
</tr>
<tr>
<td>(F = (A \text{ or } B \text{ or } C \text{ or } D))</td>
<td>unimodal</td>
<td>5</td>
</tr>
</tbody>
</table>

---

**Fig. 2.** Experiment Setup: MC/DC decision testbench

**Fig. 3.** Experiment Setup: Compile/Simulate script

under Questa. The FEC is a sufficient technique to assess the independent effect of each condition on the decision output. The FEC reporting format presents the test vectors that achieve observability of each condition \(true\), and \(false\) value in a tabular format as shown in sample FEC report in Figure 4 for an or gate. It shows also the hits status of the applied test suite. The validation of a deduced MC/DC test suite implies achieving 100% FEC coverage as achieved in all experiments.

---

**C. \(F = (A \text{ and } B ) \text{ or } (C \text{ and } D)\)**

1) **MC/DC Sample Test Suite:** The following sample set of test suites achieve the MC/DC requirements:

\[
(3,5,10,12), (3,5,10,13), (3,5,10,14), (3,6,9,12), (3,6,9,13), (3,6,9,14), (7,5,10,12), (7,5,10,13), (7,5,10,14), (7,6,9,12), (7,6,9,13), (11,5,10,12), (11,5,10,13), (11,5,10,14), (11,6,9,12), (11,6,9,13), (11,6,9,14).
\]

Figure 5 shows the gate schematic representation of the \(F\) decision statement logic structure. It has two \(and\) gates at the first level of logic and one \(or\) gate at the second logic level. The MC/DC test suite (3,5,10,12) is mapped on the logic structure to illustrate the results.
2) Minimal MC/DC Requirements: The experiment results show that 100% FEC which is MC/DC full coverage achieved by at least four test cases. The number of conditions (inputs) for this decision is four. The experiment shows a counter example that the minimum number of test cases required to achieve MC/DC is \( N \). This presents an insight for the minimal requirement to cover MC/DC, and implies that MC/DC fulfillment could be achieved by less than \((N+1)\) test cases.

3) Control Inputs Gate Crossing: Control inputs gate crossing gives rise to a higher number of test cases that fulfill observability for the input of concern. For example, when \( A \) is an observability target, the control inputs that are not sharing the path with \( A \) are \( C \) and \( D \). They are inputs to an \textit{and} gate without contribution from \( A \). This would be defined as (control inputs gate-crossing), as \( C \) and \( D \) are control inputs that cross a gate that is not shared with the input of concern \( A \).

This generates varieties of test cases to achieve the same observability target. For example, input \( A \) observability can be achieved while \( C \), and \( D \) have values \((FF, TF, FT)\) to force their \textit{and} gate output to \textit{false}, and pass input \( A \) observability required item \( A0 \) for \( A \) \textit{false} value or \( A1 \) for \( A \) \textit{true} value to the output node \( F \). This rich generation of observability test vectors, would enable optimization of the number of test vectors in an MC/DC test suite to be less than \((N + 1)\) as in this case. It allows a larger number of test suites that would meet MC/DC coverage to be selected from the large spectrum of test cases that can construct an MC/DC test suite.

4) MC/DC Test Suite Capacity rule: There is a limit in the level of optimization that can be achieved for an MC/DC test suite. This limit is inferred from the semantics of the MC/DC theory. MC/DC coverage requires test suite elements (test vectors) that form a sufficient number of MC/DC independence pairs.

Each input must have a corresponding independence pair to fulfill its MC/DC observability requirements. Hence, the number of generated independence pairs that could be constructed from a test suite must be greater or equal to the number of input conditions for the decision expression.

This paper formulates this novel MC/DC test suite capacity rule. \( C_2^T \geq N \), where \( T \) is the number of test cases in the MC/DC test suite, and \( N \) is the number of input conditions to the decision statement.

The MC/DC capacity rule states that the mathematical combination of the test vectors of the MC/DC test suite must be greater or equal to the number of inputs. This enables possibility of creation of valid independence pairs formed by the mathematical combination to achieve the MC/DC coverage requirements.

D. \( F = (A \text{ and } B) \) or \((A \text{ and } C) \) or \((B \text{ and } C)\)

1) MC/DC Sample Test Suite: The following sample set of test suites achieve the MC/DC requirements: \((1,2,3,5)\), \((1,3,6,4)\), \((1,3,5,4)\), \((1,3,6,2)\).

Figure 6 shows the gate schematic representation of the \( F \) decision statement logic structure. It has three \textit{and} gates at the first level of logic, one \textit{or} gate at the second logic level, and one \textit{or} gate at the third level of logic. The MC/DC test suite \((1,2,3,5)\) are mapped onto the logic structure.

2) MC/DC Compositionality: The test suite \((1,3,6,2)\) is considered, and this gives 100% decision coverage. The \( X, Y \), and \( M \) sub-gates are not MC/DC covered. There are multiple observability paths for the input conditions. Input \( A \) observability paths are through gates \( X \) and \( Y \), input \( B \) observability paths are through gates \( X \) and \( Z \), input \( C \) observability paths are through gates \( Y \) and \( Z \). The input condition observability objective can be achieved through any of these paths, which creates an observability coverage gap (observability hole) on the encountered sub-gates of the other observability paths for this particular input condition. This fact presents a novel insight, i.e. that MC/DC may not be hierarchically bottom-top compositional whenever multiple alternative paths are leading to inputs’ observability propagation to the decision statement output node.

3) Coupling: Coupling takes place whenever an input is common to multiple gates. The gates are placed on distinct observability paths. For instance, input \( A \) is common to gates \( X \), and \( Y \) which exist on distinct observability paths for input \( A \) to the decision output node. This imposes constraints as both \textit{and} gates share the same value of input \( A \). Experiment 1 in Section VI-C is identical to this logic structure until node \( M \). Experiment 1 has distinct inputs to the two \textit{and} gates, first gate inputs are \((A, B)\), second gate inputs are \((C, D)\). This creates three possibilities to observe \( A0 \) which are \((FFTF)\), \((FTTF)\), and \((FTTF)\) which are \((4,5,6)\). The \( A0 \) value is imposed on both \((A \text{ and } B)\) and \((A \text{ and } C)\) gates, which represents a coupling. The coupling limits the number of test vectors to observe \( A0 \) to only one test case \((FTF)\).

4) Minimal MC/DC Requirements: The sample test suites show that MC/DC coverage is achieved by at least four test cases. This complies with the conventional minimal MC/DC requirements stated in Section III MC/DC background. This experiment can not achieve fewer than \((N+1)\) test cases in its minimal test suite due to coupling. The coupling limits the number of possible test cases that would address the MC/DC observability requirements.
5) RTL Coding: Design for MC/DC: RTL coding can incrementally describe a composite logic expression. For instance, the RTL statement $F' = M$ or $Z$ is logically equivalent to $F$ in Figure 6, noting that $Z$ is the output of the and gate between $B$ and $C$. The experiment stimulates the RTL incremental model with the test suite (0,3,6). This suite results in full MC/DC coverage for the $F'$ statement. The explicit composite function $F = (A$ and $B)$ or $Z$, however, is not covered. This observation can lead to false interpretation by designers who would assume MC/DC compliance due to the RTL equivalent function $F'$ coverage. This motivates a guideline to encourage “design for MC/DC” which recommends the use of the full explicit decision statement $F$ rather than the logically equivalent statement $F'$ in order to produce MC/DC precise coverage results. The software community has also identified that MC/DC is sensitive to the program and software model structures as investigated in [11].

E. $F = [(A$ or $B)$ and $(C$ or $D)]$ xor $[(A$ and $B)$ or $(C$ and $D)]$

1) MC/DC Sample Test Suite: The following sample test of set suites achieve the MC/DC requirements:

$(1,9,6,2), (1,9,6,4), (1,9,6,8), (1,10,5,2), (1,10,5,4), (1,10,5,8), (1,12,3,2), (1,12,3,4), (1,12,3,8).$ Figure 7 shows the gate schematic representation of the $F$ decision statement logic structure. It has two and gates, two or gates at the first level of logic and one or gate, and one and gate at the second logic level, and finally an xor gate at the third level of logic. The MC/DC test suites (1,2,9,6) is mapped onto the logic structure.

![Gate schematic diagram](image)

Fig. 7. $F = [(A$ or $B)$ and $(C$ or $D)]$ xor $[(A$ and $B)$ or $(C$ and $D)]$

2) Minimal MC/DC Requirements: The experiment results show that full MC/DC coverage has been achieved by at least four test cases. The number of inputs for this decision is four. Hence, the experiment shows that the minimum number of test cases required to achieve MC/DC is $N$. The control inputs gate crossing enables a larger number of test cases to achieve observability requirements. It facilitates optimization of the test vectors required for MC/DC test suite coverage while maintaining the MC/DC test suite capacity rule. This provides a sufficient number of independence pairs required for MC/DC observability objectives corresponding to the decision input conditions.

3) MC/DC Compositionality: The function has been hierarchically built from sub-functions $X$, $Y$ and an equivalent function $Z$, in addition to the explicit complete decision statement $F$. The sub-gates in the verilog module will be $X = (A$ or $B)$ and $(C$ or $D)$, and $Y = (A$ and $B)$ or $(C$ and $D)$. The equivalent statement to the full decision expression $F$ will be the statement $Z = X$ xor $Y$. Application of the test suite (1,9,6,2), shows that the functions $X$, $Y$, $Z$ are not MC/DC covered though the full expression $F$ has 100% coverage. The logical structure for the function $F$ shows that it has two distinct logical paths to approach the output node of the circuit. The observability requirements $A_1, B_1, C_1, D_1$, which are coverage gaps for $Y$, have been achieved through the logical path of the $X$ function. Similarly, the observability requirements $C_0, D_0$, which are coverage gaps for $X$, have been achieved through the logical path of the $Y$ function. The multiple alternative logical paths that would guide a particular input condition to be observable at the final output node, create options to be selected during the MC/DC coverage computation. The selection of a determined observability path skips the other alternative optional paths sub-gates. Consequently these skipped paths’ sub-gates will have coverage holes for that particular observability requirement. This emphasizes the introduced insight that hierarchical bottom-top compositionality is not mandatory to achieve MC/DC coverage closure.

4) RTL Coding: Design-for-MC/DC: The experiment shows that the high level statement $F$ has 100% MC/DC coverage, though the RTL equivalent function $Z = (X$ xor $Y)$ is not fully covered. The experiment analysis shows that RTL coding style influences MC/DC results. The designer options for $F$, and its equivalent function $Z$, are both valid in terms of RTL design description and Boolean logic. The MC/DC results for $F$ and $Z$ are completely different though they are logically equivalent in terms of their RTL description. RTL FEC computation processes each statement separately, and this will be expected from any mechanism or EDA that parses code statements rather than logic structure. Masking MC/DC analyzes the gate structure represented by each RTL statement. For example, $Z = X$ xor $Y$ has a gate structure that is comprised of only an xor gate.

$F = [(A$ or $B)$ and $(C$ or $D)]$ xor $[(A$ and $B)$ or $(C$ and $D)]$ has a gate structure as in Figure 7 that is comprised of the complete gate structure of the function $F$. Hence, Design-for-MC/DC requires full explicit expression of the decision statement as $F$, which inherently leads to the complete logical gate structure being analyzed during MC/DC computation. It is recommended that designers adopt complete description of a logical function in order to generate precise MC/DC results. Decision statements logical gate equivalence leads to MC/DC coverage equivalence. On the contrary, RTL equivalence is not mandatory to retain MC/DC coverage equivalence for decision statements under analysis.

VII. EXPERIMENTAL EVALUATION

The experimental work analysis has resulted in the following summary:

- Masking MC/DC imposes logical structure analysis of decision statements.
- Control inputs gate crossing is defined as the existence of gates to be crossed only by control inputs before approaching a gate that is shared with the input of concern.
- Control inputs gate crossing enables generation of a larger number of test vectors to achieve a certain observability requirement.
- Control inputs gate crossing enables a larger number of generated MC/DC test suites.
• Control inputs gate crossing enables the possibility of optimization of the number of test vectors in an MC/DC test suite to be less than the conventional \((N + 1)\) minimal requirements for MC/DC coverage.

• The number of the gates in the logical path of control inputs would be directly proportional to the number of test cases that enable observability for a certain input of concern, or it is directly proportional to the possibility of optimization for the MC/DC test suite.

• Minimal MC/DC test suite test vectors should fulfill the MC/DC test suite capacity rule: \(C_T^2 \geq N\). \(T\) is the number of test cases in the MC/DC test suite, and \(N\) is the number of input conditions to the decision statement.

• Decisions that have unique logical path for the input conditions to the output node will develop MC/DC bottom-top compositionality.

• Decisions that have alternative distinct logical paths for the input conditions to the output node may not develop MC/DC bottom-top compositionality, as reaching observability from one path will leave observability holes (MC/DC coverage holes) in other paths gates.

• Hierarchical bottom-top compositionality is not mandatory to achieve full MC/DC coverage.

• Design-for-MC/DC requires full explicit expression of the decision statement.

• Decision statements’ logical gate equivalence leads to MC/DC coverage equivalence.

• RTL equivalence is not mandatory to retain Masking MC/DC coverage equivalence for decision statements under analysis.

• Coupling takes place whenever an input is common to multiple gates that are placed on distinct observability paths.

• Coupling introduces a limitation on the possibility of MC/DC test suite optimization.

VIII. Conclusion and Future Work

This paper has provided a comprehensive overview for MC/DC coverage in general and introduced insights about the application of MC/DC coverage from a logical analysis perspective. It has provided background on MC/DC coverage in terms of the traditional structural coverage types, MC/DC types, conventional MC/DC minimal requirements, and the success of MC/DC in safety critical software applications. The controllability and observability concepts which are common for hardware logic analysis have been presented as these are the main concepts underlying MC/DC analysis. The paper has provided the motivation for porting MC/DC from software to hardware based on the success of MC/DC in the software domain, the suitability of controllability and observability for hardware logic analysis, the similarity of HDLs with programming languages, and the requirement of a powerful, and cost effective structural coverage technique. Details of the conventional MC/DC coverage approaches, represented by unique-cause, masking MC/DC and the NASA gate-level evaluation approach have been given. The paper has presented a comprehensive experimental study of MC/DC in hardware verification. The experimental setup has been provided with sample experimental analysis and corresponding results and observation.

The experimental results have revealed several new insights in terms of the minimal MC/DC requirements, the MC/DC compositionality, and the RTL coding and design impact on MC/DC coverage results. These insights apply to the use of MC/DC coverage in the hardware and software domain alike. We have illustrated the possibility of optimization of the minimum requirements for MC/DC in terms of the number of test vectors in a test suite, due to control inputs gate crossing. However, the MC/DC test suite optimization is limited by the MC/DC test suite capacity rule. Our research has also shown that bottom-top compositionality is not mandatory due to possible existence of alternative logical paths for input conditions observability at the output. New RTL coding guidelines recommend the use of full explicit expression statements rather than incremental build-up of logic expressions from sub-expressions during RTL coding in order to comply with the introduced design-for-MC/DC guidelines.

This paper has experimentally demonstrated how MC/DC coverage can be used in hardware verification. MC/DC coverage notation and analysis fit very naturally to hardware designs. This investigation resulted in novel insights and rules to be adopted in general for both software and hardware. In our future research we intend to investigate MC/DC fault detection strength, the relation of MC/DC coverage to functional coverage as used in the hardware design domain, and MC/DC test suite qualification by design mutation. An algorithm for MC/DC test suite generation is currently under development to complement existing techniques with a cost-effective and powerful technique for MC/DC coverage closure.

REFERENCES


