MC/DC Implications for Software Testing from (Combinational) Logic Design

Kamal Z. Zamli, AbdulRahman A. Al-Sewari, Mohd Hafiz Mohd Hassin

Software Engineering Research Department,
Faculty of Computer Systems & Software Engineering,
Universiti Malaysia Pahang
26300 Gambang, Kuantan, Pahang, Malaysia
{kamalz,abdulrahman,hafizhassin}@ump.edu.my

Abstract. **S**tructural testing is often the most common sought criteria for exercising aspects of control flow (i.e. such as statement, branch and path coverage). In many cases, criteria based on statement, decision and path coverage appears sufficiently effective for testing (in terms of selecting the appropriate test cases for testing consideration) the various parts of the software implementation. In other cases involving complex predicates, criteria based on statement, branch, and path coverage appear problematic owing to the problem of masking (where one variable is "masking" the effects of other variables). Addressing this issue, this paper discusses the strategy for structural testing based on Multiple Condition/Decision Coverage (MC/DC). In doing so, this paper also highlights the implication of MC/DC for (combinational) logic design.

Keywords: MC/DC Test Generation, Structural Testing

1 Introduction

In the last 30 years, software grew tremendously in terms of complexity and size. Owing to its rapid growth, establishing software quality through testing can be an enormous task given finite resources at hand. Test engineers often under pressure to select the best strategies for testing both in terms of test cases effectiveness as well as its associated costs [1].

Concerning structural testing, criteria based on statement; branch and path coverage; has been the most common [2][3]. In many cases, criteria based on statement, decision and path coverage is sufficiently effective for testing the various parts of the software implementation (in terms of selecting the appropriate test cases for testing consideration). In other cases involving complex predicates, criteria based on statement, branch, and path coverage appear problematic owing to the problem of masking (where one variable is "masking" the effects of other variables). To illustrate further, assume two basic predicates – (A or B) and (A and B) respectively. The predicate (A or B) always evaluates to true when either A is true (regardless of B) and vice versa. Similarly, the predicate (A and B) is always false when B is false (regardless of A) and vice versa. In this case, A and B are said to have masked each other. Addressing this issue, this paper discusses the strategy for structural

testing based on Multiple Condition/Decision Coverage (MC/DC). In doing so, this paper also highlights the implication of MC/DC for (combinational) logic design

.

The rest of the paper is organized as follows. Section 2 illustrates an overview of Modified Condition/Decision Coverage criterion using a worked example. Section 3 elaborates the concept on Controllability and Observability based on combinational logic design. Section 4 highlights the current state-of-the-art and related work. Finally, section 4 provides the conclusion.

2 Illustrative Example on MC/DC Criterion

As a running example, consider the following if statements involving AND, OR, and NOT operations (see Fig. 1).

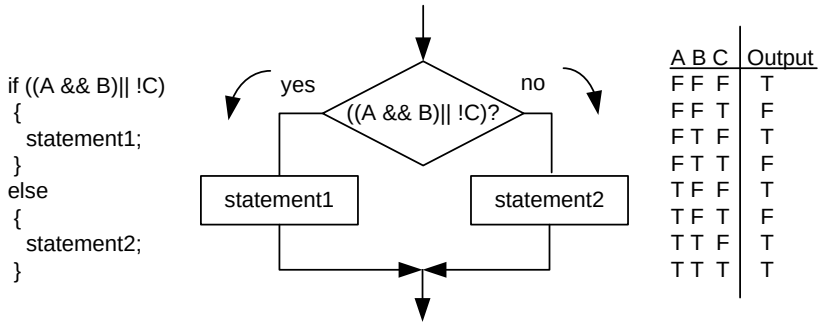

Fig. 1. Illustrative Example

Here, as long as A&&B holds TRUE, statement1 will always be executed regardless of the value of C. Similarly, given that NOT C is TRUE, statement2 will always get executed regardless of the values of A&&B. In this manner, the resulting predicate is masking each other given the wrong selection of inputs values.

Exists in pairs, MC/DC criteria insists that each variable at the atomic level is able to independently influence the overall outcome – while keeping other variable(s) unchanged. Referring from the given truth table in Fig. 1, there are:

- 1 pair for MC/DC coverage for variable A
- 1 pair for MC/DC coverage for variable B
- 3 possible pairs for MC/DC coverage for variable C

The test case selection for A and B is straightforward as there is only one pair of each respectively. However, the test case selection for C requires further elaboration. Based on the analysis in Fig. 2, if the objective is to get the most minimum number of test cases, then the obvious selected test cases would be between test suite #2 and test

suite #2 as both test sizes are 4 as opposed to 5 for test suite #1. The question is whether such a selection is the right choice as far as coverage is concerned?

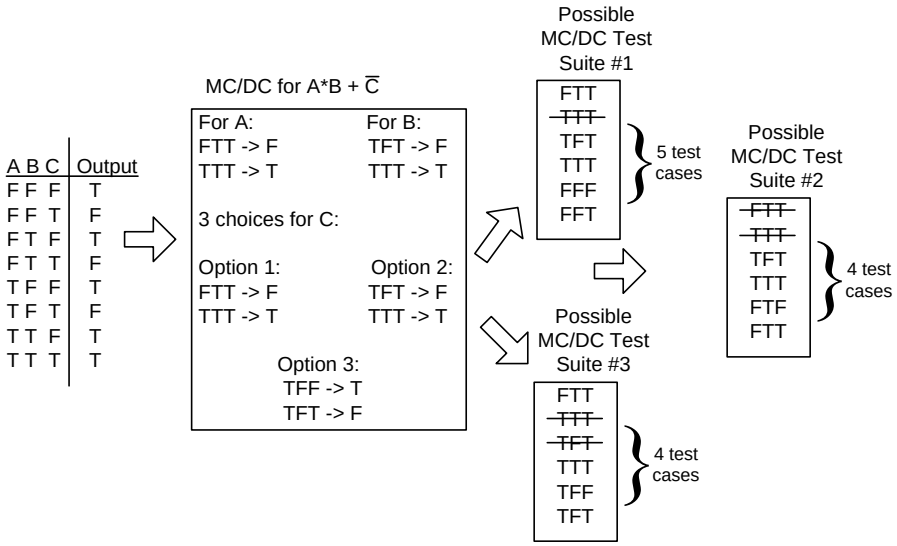

Fig. 2. Test Suite Fulfilling the MC/DC Criterion

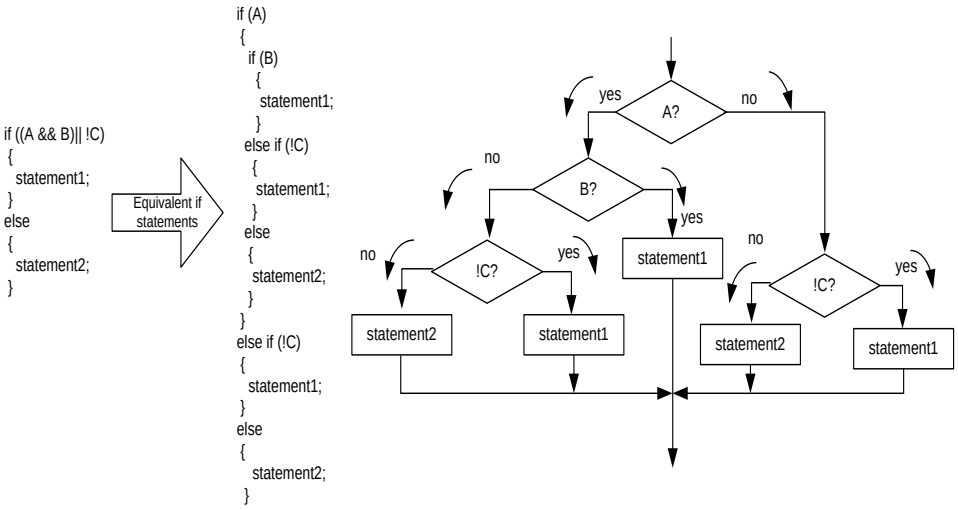

Fig. 3. Equivalent If Statements

To investigate further, there is a need to analyse the effectiveness of each of the test suite. Here, the contribution of each atomic Boolean value in the original predicate must be proved to act independently in the final outcome in line with the MC/DC requirements. This process can be achieved via using the equivalent if statements as in Fig. 3.

Surprisingly, despite being the most minimum, test suite #2 and #3 do not give 100% branch coverage (see Fig. 4). In fact, only test suite #1 gives full coverage. Given such consideration, the question now is what is the best strategy to select the suitable MC/DC test suite? The most minimum test suite might not necessarily be the best one.

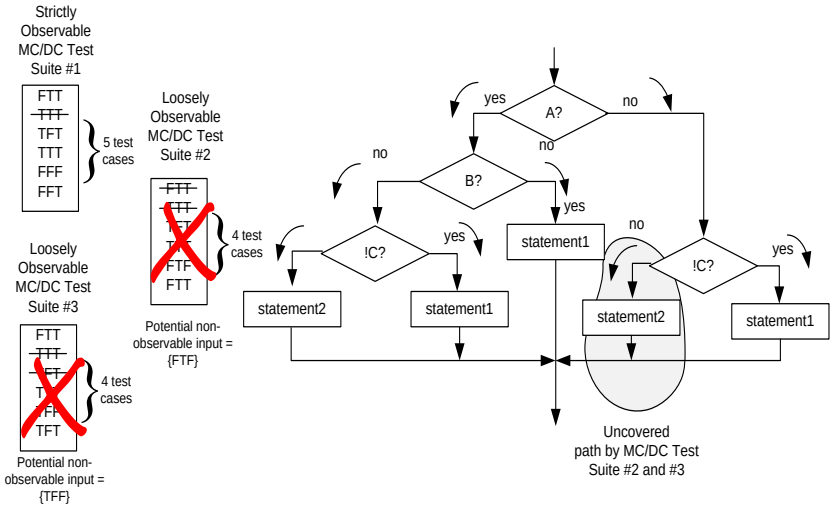

Fig. 4. Branch Coverage Analysis

3 Controllability and Observability Analysis for Combinational Logic Design

This section will give the MC/DC perspectives from combinational hardware point of view involving logic design. In the last 20 years, researchers in the field of digital/combinational logic design have proposed two main concepts namely controllability; and observability in order to address so-called design for testability requirements [4]. The former, controllability can be described loosely as the ability to test each logical operator of an expression by setting the values of the expression's inputs. The latter, observability refers to the ability to propagate the output of a logical operator under test to an observable point.

Revisiting earlier section (as in Fig. 1 till 4), Fig. 5 depicts the gate level logic implementation for the predicate given earlier. By inspection, controllability analysis is straightforward as each variable is directly accessible, that is, their values can easily be changed when required. Observability analysis is slightly subtle. Superficially, observability analysis can be described as follows:

- A will contribute to the output when B is TRUE and C is either TRUE or FALSE
- B will contribute to the output when A is TRUE and C is either TRUE or FALSE
- C will contribute to the output when either A and B or both is FALSE

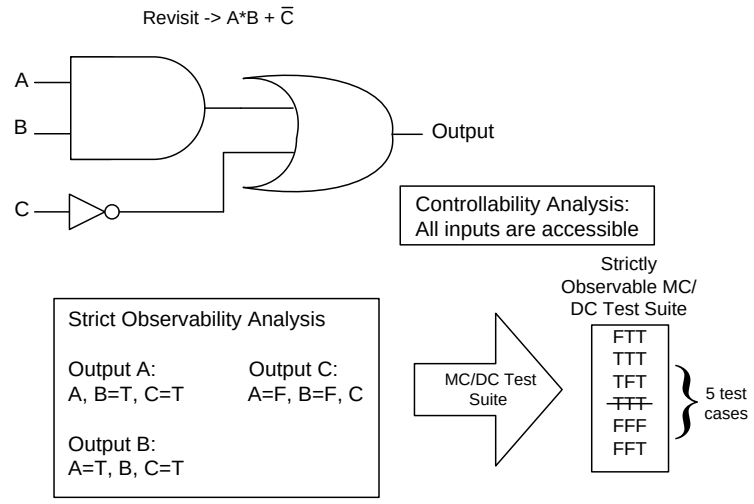

Fig. 5. Controllability and Strict Observability Analysis

Here, the contribution for A or B or C as output lacks independence. For instance, when A is TRUE, B is TRUE, and C is FALSE, the output contribution will be TRUE. In this case, the output TRUE may well come from A, B, and C. For this reason, these combinations are considered loosely observable, hence, not preferred. In the case of wrong implementation of the gates, these input output combinations might well be interpreted as correct.

From a different perspective, observability analysis can also be strictly achieved as follows:

- Only A will contribute to the output when B and C is TRUE
- Only B will contribute to the output when A and C is TRUE
- Only C will contribute to the output when A and B is FALSE

Now, one output variable is strictly observable independently of other variables. It must be observed that this is the MC/DC test suite #1 that gives 100% branch coverage in earlier section. Given such an observation, the selection of MC/DC compliant test suite must not be based on the minimum test suite but rather on its observability analysis.

4 Related Work

The consideration of MC/DC is not new as there are already a number of related works that deals with test case generation for MC/DC coverage. Jones and Harrold [5] introduce two strategies for generating MC/DC compliant test cases. The first strategy is based on the breakdown algorithm whilst the second strategy is based on the prioritization algorithm. At the start, both strategies generate the exhaustive MC/DC pairs as the basis for selection. For the first strategy, the selection of the test candidates is based on iterative generation of essential test cases. Here, essential test cases are established by summing up contribution of each test case towards MC/DC coverage. The least contributing test case is systematically removed leaving only essential ones for selection during iteration. The iteration stops when no other test cases are available for selection. For the second strategy, the selection of test candidates is also done iteratively. In this case, the contribution for each candidate test case is prioritized based on greedy ordering, that is, to cover the most pairs. The iteration stops when no more pairs are available for selection. Although helpful, both strategies appear unsuitable for handling large predicates owing to the need to generate all exhaustive MC/DC pairs.

Jun-Ru and Chin-Yu [6] adopt n-cube graph in order to generate appropriate MC/DC compliant test suite. In this case, the vertex of the cube represents the resultant Boolean enumeration for predicates under evaluation. Each vertex is traversed and arranged and evaluated using Gray code sequence ordering until all the required sequences are covered. As the sequence of ordering for MC/DC pairs are non-unique (i.e. not generalizable to only Gray code sequence), this strategy appears not optimized as far as the number of test cases are concerned.

Ghani and Clark [7] adopt optimization algorithm based on Simulated Annealing(SA) for MC/DC test suite generation. SA works based on the process of maximizing material's crystal size via heating and slow cooling [8][9]. The heating process excites the atom to move from its initial position (i.e., to avoid a local minima of internal energy) while the slow cooling process allows the atom to settle for lower internal energy configurations (i.e., for better crystal size). Analogous to the physical process, SA based strategy starts with a randomly generated MC/DC pair of test cases (i.e. initial state) and applies a series of transformations according to a pre-defined probability equation. Here, the probability equation depends heavily on parameter T (i.e. the controlling temperature of the simulation) to simulate the heating and cooling process.

Complementing the work from Ghani and Clark, Awedikian et al [3] adopts two optimization algorithms based on Hill Climbing (HC) and Genetic Algorithm (GA) respectively to generate MC/DC compliant test cases. For HC, the algorithm starts by choosing a random test case as an initial solution. The quality of the test case is evalu-

ated based on the defined fitness function. HC attempts to improve the current test case by moving to better points in a neighborhood of the current solution. This iterative process continues until a termination criterion. There are two termination conditions. First, for the given major clause, HC terminates if test case satisfying the MC/DC clause assignment are found. If after a fixed number of attempts, the algorithm is not able to satisfy the MC/DC major clause constraints, the search is stopped and another set of possible MC/DC assignments is selected. Concerning GA, the algorithm starts by creating an initial population of n test cases chosen randomly. Each chromosome represents a test case; genes are values of the input variables. In an iterative process, GA tries to improve the population from one generation to another. Test cases in a generation are selected according to their fitness in order to perform reproduction, i.e., crossover and/or mutation. Then, a new generation is constituted by the fittest test cases of the previous generation and the offspring obtained from crossover and mutation. The iterative process continues until a stopping criterion is met. Here, two stopping criteria are defined. First, for the given major clause, GA terminates if test input data satisfying the MC/DC clause assignment are found. GA is also stopped when an upper limit in computation is reached.

Summing up, adopting strategies based on optimization algorithm for MC/DC test suite generation appear to be the current trend. Despite achieving useful progress, much work on synergizing work from combinational logic design with software testing is deemed necessary. For instance, the "objective functions" for each strategy must also consider not only the test suite size but also the observability property. As discussed earlier, such consideration has not been sufficient dealt with by existing strategies in an acceptable manner.

5 Conclusion

In conclusion, this paper has highlighted the need to support MC/DC in software testing. Additionally, this paper has also highlighted the current state-of-the-art on existing work involving MC/DC and identifies novel areas for further research. Specifically, this paper has also suggests the use of observability property for MC/DC test suite generation, the concept borrowed from combinatorial logic design, as the criteria for test case selection.

Acknowledgments

This research is partially funded by myGrants: A New Design of An Artifact-Attribute Social Research Networking Eco-System for Malaysian Greater Research Network, UMP RDU Short Term Grant: Development of a Pairwise Interaction Testing Strategy with Check-Pointing Recovery Support, and ERGS Grant: CSTWay: A Computational Strategy for Sequence Based T-Way Testing.

References

[1] K. Z. Zamli, M. F. J. Klaib, M. I. Younis, N. A. M. Isa, and R. Abdullah, "Design and Implementation of a T-WayTest Data Generation Strategy with Automated Execution Tool Support," Information Sciences, Elsevier, Vol. 181(9), January 2011, pp. 1741-1758.

[2] "Software considerations in airborne systems and equipment certification," RTCA/DO-178B, pp. 2455–2464, December 1992, RTCA Inc.

[3] Z. Awedikian, K. Ayari, and G. Antoniol, "MCDC Automatic Test Input Generation", Proceedings of the 11th Annual conference on Genetic and Evolutionary Computation (GECCO '09), 2009, pp. 1657-1664.

[4] K.J.Hayhurst, D.S. Veerhusen, J.J. Chilenski, and L.K. Rierson, "A Practical Tutorial on Modified Condition/ Decision Coverage, NASA Technical Memorandum TM-2001-210876, May 2001, NASA Langley Research Center.

[5] J.A. Jones and M.J. Harrold, "Test-suite Reduction and Prioritization for Modified Condition/Decision coverage", IEEE Transactions on Software Engineering, 29(3), 2003, pp.195-209.

[6] C. Jun-Ru and H. Chin-Yu,"A Study of Enhanced MC/DC Coverage Criterion for Software Testing", in Proceedings of the 31st Annual International Computer Software and Applications Conference (COMPSAC 2007), pp. 457-464.

[7] K.Ghani and J.A. Clark, "Automatic Test Data Generation for Multiple Condition and MCDC Coverage", in Proceedings of the 4th International Conference on Software Engineering Advances, 2009, Porto, Portugal, pp. 152-157.

[8] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," Journal of Chem. Phys, vol. 21, pp. 1087–1091, 1953.

[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," Science, vol. 220, pp. 671–680,1983.