A Probabilistic Analysis For Fault Detectability of Code Coverage Metrics

Sreekumar V. Kodakara*, Deepak A. Mathaikutty†, Ajit Dingankar‡, Sandeep Shukla†, and David Lilja*
* The University of Minnesota, Minneapolis, MN 55455
† CESCA, Virginia Tech, Blacksburg, VA 24061
‡Validation Technology, Intel Corporation,Folsom, CA 95630
{sreek,lilja}@ece.umn.edu, {mathaikutty, shukla}@vt.edu, and ajit.dingankar@intel.com

***Abstract*— Functional validation of modern microprocessors is an important and complex problem. One of the problems in functional validation is the generation of test cases that has higher potential to find bugs in the design. Coverage directed test generation techniques create test suites that satisfy a coverage metric and are used extensively in validating software and hardware. In this paper, we address the question *"Which coverage metric is better in finding the commonly seen design bugs during microprocessor validation?"*. In order to answer the question, we do a probabilistic analysis on the ability of code coverage metrics to find eight fault classes proposed by Lau and Yu. The code coverage metrics analyzed in this paper include statement coverage, branch coverage and modified condition and decision coverage (MCDC), a coverage metric that is mandatory for validating critical software systems. We show that statement coverage has very low average probability ($\leq$0.2) of activating commonly seen design bugs in boolean expressions. Branch coverage is significantly superior to statement coverage only in 4 of the 8 fault classes analyzed in the paper. The fault detection probability of statement and branch coverage varies based on the size of boolean expression, making them inconsistent in its fault detecting ability. MCDC always detects 6 of the 8 fault classes and consistently detects the other two with a probability 0.5. Hence, we conclude that MCDC is the most appropriate coverage metric for microprocessor validation.**

I. Introduction

Functional validation has become a key ingredient in the development cycle of current and future microprocessors. Simulation is widely used to validate large systems like microprocessors. In simulation based validation (refer Figure 1), a *test* is executed in a golden reference model as well as in the design under test, which in this case is the RTL implementation of the microprocessor. A bug is detected only if it is exercised by the test, which results in the state of the microprocessor to deviate from that predicted by the golden model. Therefore, the success of simulation based validation depends on the quality of tests. Coverage metrics inspired from software validation (statement and branch), state machine representation (state, transition and path coverage), property specification (formal, temporal logic and assertions) and functionality of the processor (Functional coverage) are employed to measure the quality of tests generated for validation. The effectiveness of the coverage-directed test depends on (i) the *"types of design bugs"* that commonly occur, (ii) the strength of the *coverage metric* to activate these bugs, and (iii) the capability of the test to propagate the effects of the activated bugs to the circuit outputs. We address the first two quality related issues in this work, namely what categorization of the commonly seen bugs is useful and how to analyze the coverage metric's capability to activate these bug categories. The observability of a test (third issue) is tightly coupled with the test generation methodology and requires more information than the *controllability* provide by a coverage metric. This *observability* information needs to be maintained by the test generator [5].

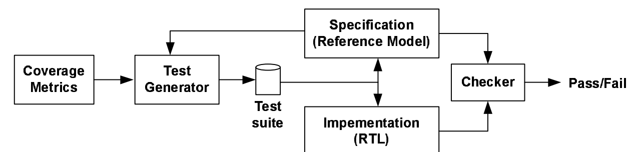

Fig. 1. Validation Flow

Campenhout [2] and Velev [8] conducted empirical studies on errors/bugs observed in academic microprocessor projects. A similar study of faults observed during validation of software systems [4] were analyzed and an extended fault model comprising of **nine types of faults** was proposed by Lau *et al.* in [7]. A correlation study of the bugs in [2], [8] to the extended fault model in [7] was performed in [3]. The analysis was capable to segregate the microprocessor bugs into *eight of the nine fault classes* proposed by the extended fault model. Therefore, we choose the fault classes proposed in [7] as *representative classes* to obtain a good categorization for the typically seen bugs during microprocessor validation.

Code coverage metrics are used to determine if the code structure is adequately covered by a test suite during validation. These metrics are used in addition to functional coverage metrics, to determine the quality of validation. The most commonly used code coverage metrics in microprocessor validation are statement and branch. Besides the traditional coverage metrics, we also analyze the *modified condition and decision coverage (MCDC)*, a metric that is mandatory for validating critical software systems like avionics [1]. In this paper, we perform a probabilistic analysis on the ability of these code coverage metrics to activate bugs specific to the representative classes from the extended fault model. In other words, we find the probability of a bug from one of the fault classes being activated by a test suite that achieves 100% coverage of a code coverage metric. We illustrate that statement coverage performs poorly when compared to branch

TABLE I

FAULT CLASSES

S.No	*Fault Class*	*Description*	*Expression*
1	Term Omission Fault (TOF)	A term t_i is omitted	$S^{t_i}_{TOF} = t_1 + ... + t_{i-1} + t_{i+1} + ... + t_m$, where $1 \leq i \leq m$.
2	Term Negation Fault (TNF)	A term t_i is erroneously negated	$S^{t_i}_{TNF} = t_1 + .. + \overline{t_i} + ... + t_m$, where $1 \leq i \leq m$
3	Literal Omission Fault (LOF)	A literal x^i_j is omitted from a term t_i	$S^{x^i_j}_{LOF} = t_1 + ... + t_{i-1} + x^i_1 ... x^i_{j-1} . x^i_{j+1} ... x^i_{k_i} + t_{i+1} + ... + t_m$, where $1 \leq i \leq m$ and $1 \leq j \leq k_i$.
4	Literal Insertion Fault (LIF)	An extra literal $x^i_{k_{i+1}}$ is inserted into a term t_i	$S^{x^i_{k_{i+1}}}_{LIF} = t_1 + ... + t_i . x^i_{k_{i+1}} + ... + t_m$, where $1 \leq j \leq k_i$
5	Literal Negation Fault (LNF)	A literal x^i_j is erroneously negated in a term t_i	$S^{x^i_j}_{LNF} = t_1 + ... + t_{i-1} + x^i_1 ... \overline{x}^i_j ... x^i_{k_i} + t_{i+1} + ... + t_m$, where $1 \leq i \leq m$ and $1 \leq j \leq k_i$
6	Literal Reference Fault (LRF)	A literal y is referenced instead of x^i_j in a term t_i	$S^{x^i_j}_{LRF} = t_1 + ... + t_{i-1} + x^i_1 ... x^i_{j-1} . y . x^i_{j+1} ... x^i_{k_i} + t_{i+1} + ... + t_m$, where $1 \leq j \leq k_i$
7	Disjunctive Operator Reference Fault (ORF[+])	A boolean operator (+) in S is replaced by (.)	$S^{t_i}_{ORF[+]} = t_1 + ... + t_i . t_{i+1} + ... + t_m$, where $1 \leq i \leq m$
8	Conjunctive Operator Reference Fault (ORF[.])	A boolean operator (.) in S is replaced by (+)	$S^{x^i_j}_{ORF[.]} = t_1 + ... + t_{i-1} + x^i_1 ... x^i_j + x^i_{j+1} ... x^i_{k_i} + t_{i+1} + ... + t_m$, where $1 \leq i \leq m$ and $1 \leq j \leq k_i$

coverage and MCDC. Branch coverage performs better than statement, but fails to activate more half of the fault classes. We illustrate through our probabilistic analysis that MCDC attains 100% coverage on six of the eight fault classes in the extended fault model. Therefore, we motivate the usage of MCDC as a metric for microprocessor validation, through our theoretical analysis of the fault detection[1] capability for the different code coverage metrics.

A. Main Contributions

Our main contribution in this paper can be summarized as follows: (i) A theoretical analysis of code coverage metrics (statement, branch, MCDC) in its ability to detect design bugs, (ii) Application of the analysis to the fault classes in [7] and deriving the probability expressions for the different code coverage metrics, (iii) Illustrating that MCDC when compared to the other code coverage metrics has the highest probability of detecting the fault classes, and (iv) In our results, we demonstrate how the probability varies with respect to the complexity of the faulting logic for the different fault classes.

II. BACKGROUND & RELATED WORK

We outline the fault classes and introduce the coverage metrics considered in the context of our work.

Fault Classes: Lau and Yu [7] proposed a comprehensive set of nine fault classes related to boolean expressions in disjunctive normal form (DNF). Let us consider a boolean expression in DNF form with m terms $S = t_1 + .. + t_m$. Let t_i denote the ith term in S such that t_i is a conjunction of k_i literals. Let x^i_j denote the jth literal in t_i, where $1 \leq j \leq k_i$. S^e_f is defined as the faulty implementation of S, where f is the name of the fault class and e identifies the exact fault in S. Table 1 lists the eight different fault classes relevant to our analysis.

Coverage Metrics: We define the three structural coverage criteria, namely statement, branch and MCDC, for which tests are generated in our framework.

[1] Activating a bug that belongs to the fault class.

To achieve *statement coverage*, a test suite should invoke every executable statement in the code. *Branch coverage* mandates that the test suite should execute both the *true* and *false* outcomes of all *decisions* (eg., *if* statements) in the code. This metric is stronger than statement coverage, but is still weak due to the presence of *conditions* with in a *decision*. A *condition* is defined as a boolean expression with **no** boolean operators. A *decision* is defined as a boolean expression with **zero or more** boolean operators. For example, a decision (A $>$ 10 $\|$ B) has two conditions (A $>$ 10) and B.

MCDC [4] was developed to test the effect of *conditions* within a *decision*. MCDC requires that each *condition* with in a *decision* should *independently* affect the outcome of the *decision*. This coverage ensures that the effect of each *condition* is tested relative to other *conditions* within the *decision*. This implies that no *condition* is left untested due to logical masking. For a decision with m conditions, MCDC requires at least $m+1$ test cases [4]. Attaining 100% MCDC typically requires more number of test cases when compared to statement and branch coverage. For example, consider a decision d with three conditions (A $<$ 0), B and (C $<>$ 10), the test cases generated are shown below:

Example:
d = ((A$<$0) $\wedge$ B $\vee$ (C $<>$ 10))
Test cases generated for MCDC:
case (A$<$0): T_1 = [0, 1, 0] (d=0) and T_2 = [1, 1, 0] (d=1)
case B: T_3 = [1, 0, 0] (d=0) and T_4 = [1, 1, 0] (d=1)
case (C $<>$ 10): T_5 = [0, 1, 0] (d=0) and T_6 = [0, 1, 1] (d=1)
One possible MCDC Test suite: $\{ T_1, T_2, T_3, T_6 \}$

The MCDC test criteria is discussed in [4] and many variant have been proposed in [1]. Based on a probability model, a formal analysis of MCDC has been presented in [9] applicable to software testing, but assumes all the outputs of the boolean expression to be incorrect with a fixed probability. Furthermore the model is not directly applicable to fault sensitivity and cannot distinguish the classes of faulty expressions that are easier to test. A formal framework is presented in [6], where different forms of MCDC are mathematical defined and compared w.r.t their fault detection effectiveness on some of

the fault classes from the extended fault model in [7].

III. Theoretical Analysis

Preliminary Definition: Let S and S^e_f represent the fault free specification and faulty implementation of the boolean expression in DNF form, respectively. Let the set of unique terms and literals in S be denoted by T_S and L_S, with the cardinality n_T and n_L respectively. A term t_i in T_S is a conjunction of n_{t_i} literals and is given by $t_i = l_1 \wedge l_2 \wedge \ldots \wedge l_{n_{t_i}}$, where $l_i \in L_S$. A literal l_i is equally likely to take values '0' or '1' (i.e., $p_0 = p_1 = 0.5$). A test is the instantiation of the expressions S and S^e_f with truth values assigned to their respective literal set L_S and $L_{S^e_f}$. We will assume that the test is generated from the fault free expression S. Let $V_{tc}(S)$ and $V_{tc}(S^e_f)$ represent the value of S and S^e_f evaluation for a test case **tc** generated from S, respectively. The fault f in S^e_f is detected if $V_{tc}(S) \neq V_{tc}(S^e_f)$.

Let $\mathbb{T}$ represent the set of all unique test cases generated from S. Here $|\mathbb{T}| = 2^{n_L}$. Let $\mathbb{T}^0$ and $\mathbb{T}^1$ represent the set of all test cases that evaluate S to *false* and *true*, respectively. Here $\mathbb{T} = \mathbb{T}^0 \cup \mathbb{T}^1$. Since S is in DNF form, a test evaluates S to *false* only when all the terms in S evaluates to *false*. Therefore, a term t_i with n_{t_i} literals, has $2^{n_{t_i}}$ tests, of which $(2^{n_{t_i}} - 1)$ tests evaluates t_i to *false* and one test evaluates t_i to *true*. The total number of test cases generated from S that evaluates S to *false* and *true*, respectively, is given by

$|\mathbb{T}^0| = \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1)$, where $t_i \in T_S$
$|\mathbb{T}^1| = |\mathbb{T}| - |\mathbb{T}^0| = 2^{n_L} - \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1)$

Probabilistic Model: Let $\mathbb{T}_f$ denote the subset of test cases that reveal the fault f. In our analysis, we derive the expression for the probability $P(cov_f)$, that at least one test case from $\mathbb{T}_f$ is included in the test suite that satisfies the coverage metric cov, where cov is either st, br or $mcdc$ for statement, branch or MCDC coverage metric, respectively. The probability equations for each coverage metric's capability of detecting a fault f is derived as follows:

Statement coverage is satisfied by choosing a test case **tc** from $\mathbb{T}$. The probability $P(st_f)$ that the fault f is detected by **tc** is the probability that **tc** belongs to the set $\mathbb{T}_f$. Hence the equation for $P(st_f)$ is given by

$$P(st_f) = \frac{|\mathbb{T}_f|}{|\mathbb{T}|}$$

Branch coverage is satisfied if two test cases tc_i and tc_j exercises S, where $V_{tc_i}(S)$ = *true* and $V_{tc_j}(S)$ = *false*. Therefore, the test suite for branch coverage should consist of one test case each from $\mathbb{T}^0$ and $\mathbb{T}^1$. Each of these test cases are *independently* chosen from their respective sets. In order to evaluate the probability $P(br_f)$ that atleast one of the two test cases tc_i and tc_j finds the fault f, we compute the test set $\mathbb{T}^0_f \subseteq \mathbb{T}^0$ and $\mathbb{T}^1_f \subseteq \mathbb{T}^1$, which consist of test cases that find the fault f and evaluates the expression S to *false* and *true*, respectively. Therefore, $\mathbb{T}_f = \mathbb{T}^0_f \cup \mathbb{T}^1_f$. Hence the probability $P(br_f)$ that the fault f is detected by branch coverage is the probability that at least one of the two test cases selected from $\mathbb{T}^0$ and $\mathbb{T}^1$ belongs to $\mathbb{T}^0_f$ or $\mathbb{T}^1_f$ respectively. Hence, the equation for $P(br_f)$ is given by

$$\begin{aligned} P(br_f) &= \frac{|\mathbb{T}^0_f|}{|\mathbb{T}^0|}\frac{|\overline{\mathbb{T}^1_f}|}{|\mathbb{T}^1|} + \frac{|\overline{\mathbb{T}^0_f}|}{|\mathbb{T}^0|}\frac{|\mathbb{T}^1_f|}{|\mathbb{T}^1|} + \frac{|\mathbb{T}^0_f|}{|\mathbb{T}^0|}\frac{|\mathbb{T}^1_f|}{|\mathbb{T}^1|} \\ &= 1 - \left\{ \frac{|\overline{\mathbb{T}^0_f}|}{|\mathbb{T}^0|}\frac{|\overline{\mathbb{T}^1_f}|}{|\mathbb{T}^1|} \right\} \\ &= 1 - \left\{ \frac{|\mathbb{T}^0| - |\mathbb{T}^0_f|}{|\mathbb{T}^0|}\frac{|\mathbb{T}^1| - |\mathbb{T}^1_f|}{|\mathbb{T}^1|} \right\} \end{aligned}$$

MCDC requires that every condition independently affects the output of the decision. A condition maps to a literal and a decision maps to an expression, therefore MCDC for an expression S requires that each literal independently affects the output of S. The test set $\mathbb{T}$ for S generated by performing MCDC is divided into n_L non-disjoint sets. Let $\mathbb{T}^{l_i}$ denote the test set for the literal l_i. We further divide $\mathbb{T}^{l_i}$ into two disjoint sets namely $\mathbb{T}^{l_i\text{-}0}$ and $\mathbb{T}^{l_i\text{-}1}$ such that $\mathbb{T}^{l_i} = \mathbb{T}^{l_i\text{-}0} \cup \mathbb{T}^{l_i\text{-}1}$. A test case **tc** $\in \mathbb{T}^{l_i\text{-}0}$ evaluates S such that $V_{tc}(S) = 0$ and vice versa.

For a test case **tc** w.r.t to literal l_i in term t_i to belong to $\mathbb{T}^{l_i\text{-}0}$, the following criteria should be satisfied by **tc**:

1) $V_{tc}(S) = 0$ **s.t.** $V_{tc}(t_i) = 0 \wedge \forall\ t_j \in T$, $V_{tc}(t_j) = 0$ where $t_j \neq t_i$,
2) $l_i = 0 \wedge \forall\ l_j \in t_i$, $l_j = 1$, where $l_i \neq l_j$

The first criterion states that S should be set to '0'. Since S is in DNF form, this implies that all terms in S should evaluate to '0'. The second criterion states that, all literals besides l_i in term t_i are forced to '1' and l_i is set to '0'. Literals in other terms t_j, where $t_j \neq t_i$, can take any value, as long as t_i evaluates to '0'. All the test cases that satisfy these two criteria belongs to $\mathbb{T}^{l_i\text{-}0}$. For a literal l_i in t_i, $|\mathbb{T}^{l_i\text{-}0}| = \prod_{j=1}^{n_T}$ $(2^{n_{t_j}}$ - 1), where $t_i \neq t_j$.

For a test case **tc** w.r.t to literal l_i in term t_i to belong to $\mathbb{T}^{l_i\text{-}1}$, the following criteria should hold true:

1) $V_{tc}(S) = 1$ **s.t.** $V_{tc}(t_i) = 1 \wedge \forall\ t_j \in T$, $V_{tc}(t_j) = 0$ where $t_j \neq t_i$,
2) $l_i = 1 \wedge \forall\ l_j \in t_i$, $l_j = 1$, where $l_i \neq l_j$

The first criterion states that S should be set to '1' due to the term t_i. All other terms in S should be set to '0'. Since t_i should evaluate to '1', this implies that all literals in term t_i should be set to '1'. This condition is stated in the second criterion. Note that a test case w.r.t l_i in term t_i that belongs to $\mathbb{T}^{l_i\text{-}1}$ assigns all literals in t_i to '1'(follows from criterion 2). Therefore for every literal l_i in t_i, the same test case set $\mathbb{T}^{l_i\text{-}1}$ is generated. Hence, $\forall\ l_i,\ l_k \in t_i$, $|\mathbb{T}^{l_i\text{-}1}| = |\mathbb{T}^{l_k\text{-}1}| = \prod_{j=1}^{n_T}$ $(2^{n_{t_j}}$ - 1), where $t_i \neq t_j \wedge l_i \neq l_k$.

In order to evaluate the probability $P(mcdc_f)$ that atleast one test case from the test suite that satisfies MCDC will find the fault f, we compute the test suite w.r.t to every literal l, such that $\mathbb{T}^{l\text{-}0}_f \subseteq \mathbb{T}^{l\text{-}0}$ and $\mathbb{T}^{l\text{-}1}_f \subseteq \mathbb{T}^{l\text{-}1}$, which consist of test cases that find the fault type f, respectively. Therefore, $\mathbb{T}^l_f = \mathbb{T}^{l\text{-}0}_f \cup \mathbb{T}^{l\text{-}1}_f$. Hence the probability $P(mcdc^{l_i}_f)$ that the fault type f is detected by MCDC is the probability that at least one of the two test cases selected from $\mathbb{T}^{l_i\text{-}0}$ and $\mathbb{T}^{l_i\text{-}1}$ obtained

w.r.t to literal l_i finds the fault, which is given by

$$\begin{aligned} P(mcdc_f^{l_i}) &= \frac{|\mathbb{T}_f^{l_i\text{-}0}|}{|\mathbb{T}^{l_i\text{-}0}|}\frac{|\overline{\mathbb{T}_f^{l_i\text{-}1}}|}{|\mathbb{T}^{l_i\text{-}1}|} + \frac{|\overline{\mathbb{T}_f^{l_i\text{-}0}}|}{|\mathbb{T}^{l_i\text{-}0}|}\frac{|\mathbb{T}_f^{l_i\text{-}1}|}{|\mathbb{T}^{l_i\text{-}1}|} + \frac{|\mathbb{T}_f^{l_i\text{-}0}|}{|\mathbb{T}^{l_i\text{-}0}|}\frac{|\mathbb{T}_f^{l_i\text{-}1}|}{|\mathbb{T}^{l_i\text{-}1}|} \\ &= 1 - \left\{ \frac{|\overline{\mathbb{T}_f^{l_i\text{-}0}}|}{|\mathbb{T}^{l_i\text{-}0}|}\frac{|\overline{\mathbb{T}_f^{l_i\text{-}1}}|}{|\mathbb{T}^{l_i\text{-}1}|} \right\} \\ &= 1 - \left\{ \frac{|\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}_f^{l_i\text{-}0}|}{|\mathbb{T}^{l_i\text{-}0}|}\frac{|\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}_f^{l_i\text{-}1}|}{|\mathbb{T}^{l_i\text{-}1}|} \right\} \end{aligned}$$

In our analysis, we assume $P(mcdc_f) = P(mcdc_f^{l_i})$ because for all the faults under consideration, it is sufficient to determine whether the test suite generated w.r.t a literal l_i finds the fault f. For the faults such as LNF and LOF, if the fault is associated with some literal say l, then $P(mcdc_f^{l})$ is sufficient to determine whether MCDC finds the fault. If the test suite $\mathbb{T}^l$ does not detect the fault ($\mathbb{T}_f^l \cap \mathbb{T}^l = 0$), then no test suite generated as a result of MCDC on any other literal would either. For fault classes TNF and TOF, a particular term say t is omitted or negated, which implies every literal in t is either omitted or negated. Therefore for any literal l in term t, $P(mcdc_f^{l})$ is sufficient to determine whether MCDC finds the fault associated with t because the expression is written in DNF. The fault classes $ORF[.]$ and $ORF[+]$ are also determinable w.r.t to any literal. Finally for the fault classes LRF and LIF, an external literal either replaces a literal in a term or is erroneously inserted in some term. For these, $P(mcdc_f)$ is highly coupled to the relation of the literal being replaced and the replacement in the case of LRF or the relation of the term and the extra literal inserted into it. Therefore, the validity of our assumption of using $P(mcdc_f^{l})$ instead of $P(mcdc_f)$ need to be justified on by case by case basis, which is addressed in the Section 4.

Our analysis makes the following assumptions: (i) the expressions S and S_f^e are in DNF, (ii) S_f^e has only one instance of one type of fault (iii) All literals in S and S_f^e are distinct. Campenhout et al [2] and Velev [8] illustrate that typical microprocessor bugs arise from incorrect signals/gates in the control logic and missing or extra signals/gates in the control logic. These errors translate to the omission, insertion, or incorrect reference of boolean operands or operators, when restricted to simple faults relevant to Boolean expressions in DNF. Therefore, we use expressions in DNF for our analysis. In our analysis, we have not considered the effect of coupling in the DNF expression. Coupling arises when one or more literals is present in two or more terms in the expression. The fault detection probability of coverage metrics could increase or decrease depending upon the number of literals coupled and the type of coupling present in the expression. As a part of our future work, we plan to evaluate fault detection ability of coverage metrics in the presence of multiple faults and the effect of coupling on our probability model.

IV. Application on Fault Classes

We apply the theoretical analysis on the fault classes discussed in Section II and derive the equations for the probability that the coverage metrics find these faults. For each fault, we perform the following steps: (i) Describe the fault, (ii) Enumerate the detection criteria for a test case, (iii) Generate the formulas for the number of test cases in $|\mathbb{T}_f|$, $|\mathbb{T}_f^0|$ and $|\mathbb{T}_f^1|$ and (iv) Derive the probabilities based on the equations in Section III.

A. TNF

Term Negation Fault occurs when a term is erroneously negated in a boolean expression.

Expressions: Let the term t_j in the expression S be negated in expression $S_{TNF}^{t_j}$.

Criteria: For a test case **tc** to detect TNF, the following conditions should be satisfied:

1) $V_{tc}(t_i) = 0$ where $1 \leq i, j \leq n_T$ and $i \neq j$
2) $V_{tc}(t_j) = 1(0) \implies V_{tc}(S) = 1(0)$

Criterion 1 states that the effect of the term t_j on the outcome of the expression S should not masked by any other term in S. For S in DNF, this implies that all terms other than t_j should set to '0' in **tc**. The number of test cases that satisfy criterion 1 is $\prod_{i=1}^{n_T}(2^{n_{t_i}} - 1)$, where $t_i \neq t_j$. For a fault that causes the term t_j in S to be negated in $S_{TNF}^{t_j}$, any value assignment to t_j in **tc** will result in $V_{tc}(S) \neq V_{tc}(S_{TNF}^{t_j})$. Therefore, t_j can be any of the possible $2^{n_{t_j}}$ cases in **tc** to satisfy criterion 2. The total number of test cases that finds the fault is given by

$$|\mathbb{T}_{TNF}| = 2^{n_{t_j}} \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1)$$

Furthermore, we know that t_j has $2^{n_{t_j}} - 1$ test cases that evaluate $V(t_j)$ to '0' and one test case that evaluates $V(t_j)$ to '1'. Hence the number of fault finding test cases that evaluate $V(S)$ to '0' and '1' is given by

$$\begin{aligned} |\mathbb{T}_{TNF}^0| &= (2^{n_{t_j}} - 1)\prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1) = \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1) \\ |\mathbb{T}_{TNF}^1| &= \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1) \end{aligned}$$

Similarly, from MCDC in Section III, we know that t_j has one case each to test a literal that evaluates $V(t_j)$ to '0' and '1', respectively. Hence the number of fault finding test cases for one literal in t_j that evaluates $V(S)$ to '0' and '1' is given by,

$$|\mathbb{T}_{TNF}^{l_i\text{-}0}| = |\mathbb{T}_{TNF}^{l_i\text{-}1}| = \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1)$$

The values for $|\mathbb{T}^0| - |\mathbb{T}_{TNF}^0|$ and $|\mathbb{T}^1| - |\mathbb{T}_{TNF}^1|$ is derived below to calculate the probability of branch coverage.

$$\begin{aligned} |\mathbb{T}^0| - |\mathbb{T}_{TNF}^0| &= \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1) - \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1) = 0 \\ |\mathbb{T}^1| - |\mathbb{T}_{TNF}^1| &= 2^{n_L} - \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1) - \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1) \\ &= 2^{n_L} - 2^{n_{t_j}}\prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1) \end{aligned}$$

Similarly values for $|\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}_{TNF}^{l_i\text{-}0}|$ and $|\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}_{TNF}^{l_i\text{-}1}|$ are derived below to calculate the probability of MCDC coverage.

$$\begin{aligned} |\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}_{TNF}^{l_i\text{-}0}| &= \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1) - \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1) = 0 \\ |\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}_{TNF}^{l_i\text{-}1}| &= \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1) - \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{t_i}} - 1) = 0 \end{aligned}$$

Probabilities: We substitute the value of $|\mathbb{T}_{TNF}|$ in the formula for statement coverage given in Equation III to get the value for $P(st_{TNF})$, the probability of a test suite that satisfies statement coverage detecting the fault. The probability is given by

$$P(st_{TNF}) = \frac{|\mathbb{T}_{TNF}|}{|\mathbb{T}|} = \frac{2^{n_{l_j}} \prod_{\substack{i=1 \\ i \neq j}}^{n_T}(2^{n_{l_i}} - 1)}{2^{n_L}}$$

The value for $P(br_{TNF})$ is obtained by substituting the above derived values into the formula given in Equation III. The probability is given by

$$
\begin{aligned}
P(br_{TNF}) &= 1 - \left\{ \frac{|\mathbb{T}^0| - |\mathbb{T}^0_{TNF}|}{|\mathbb{T}^0|} \frac{|\mathbb{T}^1| - |\mathbb{T}^1_{TNF}|}{|\mathbb{T}^1|} \right\} \\
&= 1 - \left\{ \frac{0}{\prod_{i=1}^{n_T}(2^{n_{t_i}}-1)} \frac{2^{n_L} - 2^{n_{t_j}} \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}}-1)}{2^{n_L} - \prod_{i=1}^{n_T}(2^{n_{t_i}}-1)} \right\} = 1
\end{aligned}
$$

The value for $P(mcdc_{TNF})$ is obtained by substituting the above derived values into the formula given in Equation III. The probability is given by

$$
\begin{aligned}
P(mcdc_{TNF}) &= 1 - \left\{ \frac{|\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}^{l_i\text{-}0}_{f}|}{|\mathbb{T}^{l_i\text{-}0}|} \frac{|\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}^{l_i\text{-}1}_{f}|}{|\mathbb{T}^{l_i\text{-}1}|} \right\} \\
&= 1 - \left\{ \frac{0}{\prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}}-1)} \frac{0}{\prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}}-1)} \right\} = 1
\end{aligned}
$$

B. TOF

The most frequent kinds of bugs in the control logic are from the missing of a condition (56.8% in [8]). These types of bugs occur when one or more scenarios are erroneously ignored in the expressions that generate the control signals. It translates to a term omission fault (TOF).

Expressions: Let the term t_j in the expression S be omitted in expression $S^{t_j}_{TOF}$.

Criteria: For a test case **tc** to detect TOF, the following conditions should be satisfied:

1) $V_{tc}(t_i) = 0$ where $1 \leq i, j \leq n_T$ and $i \neq j$
2) $V_{tc}(t_j) = 1 \implies V_{tc}(S) = 1$

Criterion 1 is satisfied similar to the TNF scenario. Therefore, the number of test cases that satisfy criterion 1 is $\prod_{i=1}^{n_T}(2^{n_{t_i}} - 1)$, where $t_i \neq t_j$. For a fault that causes the term t_j in S to be omitted in $S^{t_j}_{TOF}$, only the value assignment '1' to t_j in **tc** will result in $V_{tc}(S) \neq V_{tc}(S^{t_j}_{TOF})$. Therefore, **tc** assigns every literal in the t_i to '1' to satisfy criterion 2. The total number of test cases that finds the fault is given by

$$
|\mathbb{T}_{TOF}| = \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1)
$$

The number of fault finding test cases that evaluate $V(S)$ to '0' and '1' is given by

$$
\begin{aligned}
|\mathbb{T}^0_{TOF}| &= 0 \\
|\mathbb{T}^1_{TOF}| &= \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) \\
|\mathbb{T}^{l_i\text{-}0}_{TOF}| &= |\mathbb{T}^{l_i\text{-}1}_{TOF}| = \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1)
\end{aligned}
$$

The values for $|\mathbb{T}^0| - |\mathbb{T}^0_{TOF}|$, $|\mathbb{T}^1| - |\mathbb{T}^1_{TOF}|$, $|\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}^{l_i\text{-}0}_{TOF}|$ and $|\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}^{l_i\text{-}1}_{TOF}|$ to calculate the probability of branch and MCDC coverage is shown below:

$$
\begin{aligned}
|\mathbb{T}^0| - |\mathbb{T}^0_{TOF}| &= \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1) - 0 = \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1) \\
|\mathbb{T}^1| - |\mathbb{T}^1_{TOF}| &= 2^{n_L} - \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1) - \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) \\
&= 2^{n_L} - 2^{n_{t_j}} \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1)
\end{aligned}
$$

$$
\begin{aligned}
|\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}^{l_i\text{-}0}_{TOF}| &= \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) - \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) = 0 \\
|\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}^{l_i\text{-}1}_{TOF}| &= \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) - \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) = 0
\end{aligned}
$$

Probabilities: The probability values for statement, branch and MCDC detecting a TOF fault are given by:

$$
\begin{aligned}
P(st_{TOF}) &= \frac{\prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}}-1)}{2^{n_L}} \\
P(br_{TOF}) &= \frac{\prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}}-1)}{2^{n_L} - \prod_{i=1}^{n_T}(2^{n_{t_i}}-1)} \\
P(mcdc_{TOF}) &= 1
\end{aligned}
$$

C. LIF

Literal Insertion Fault occurs when a literal is erroneously included in a term in the boolean expression.

Expressions: Let the term t_j in the expression S be inserted with literal l in expression $S^{t_j}_{LIF}$.

Criteria: For a test case **tc** to detect LIF, the following conditions should be satisfied:

1) $l = 0 \implies V_{tc}(S^{t_j}_{LIF}) = 0$

Criterion 1 is satisfied by: (1.1) Setting l to '0', (1.2) Setting all other literals in t_j to '1' and (1.3) Setting all terms in S other than t_j to '0'. Note that the tests are generated from S which does not have the additional literal l in its term t_j. Thus the generated test will not impose any value on l. In our analysis, we assume that l is independent of the literals present in S and is equally likely to take values '0' or '1'. Therefore, the probability of satisfying (1.1) is $\frac{1}{2}$. The number of tests generated from S that satisfies (1.2) & (1.3) is given by

$$
\begin{aligned}
|\mathbb{T}_{LIF}| &= \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) \\
|\mathbb{T}^0_{LIF}| &= 0 \\
|\mathbb{T}^1_{LIF}| &= \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) \\
|\mathbb{T}^{l_i}_{LIF}| &= \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) \\
|\mathbb{T}^{l_i\text{-}0}_{LIF}| &= 0 \\
|\mathbb{T}^{l_i\text{-}1}_{LIF}| &= \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1)
\end{aligned}
$$

Using these values, we calculate

$$
\begin{aligned}
|\mathbb{T}^0| - |\mathbb{T}^0_{LIF}| &= \prod_{i=1}^{n_T}(2^{n_{t_i}} - 1) \\
|\mathbb{T}^1| - |\mathbb{T}^1_{LIF}| &= 2^{n_L} - \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1)\left\{2^{n_{t_j}} - 1\right\} \\
|\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}^{l_i\text{-}0}_{LIF}| &= \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}} - 1) \\
|\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}^{l_i\text{-}1}_{LIF}| &= 0
\end{aligned}
$$

Probabilities: The probability that a test generated from S will reveal the fault is the product of the probability of generating a test case from S that satisfies (1.2) & (1.3) and the probability of satisfying (1.1). The former depends on the coverage metric and the latter is a constant ($\frac{1}{2}$). The probability values for each coverage metric are given below:

$$
\begin{aligned}
P(st_{LIF}) &= \frac{1}{2}\frac{|\mathbb{T}_{LIF}|}{|\mathbb{T}|} = \frac{1}{2}\frac{\prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}}-1)}{2^{n_L}} \\
P(br_{LIF}) &= \frac{1}{2}\left\{1 - \frac{|\mathbb{T}^0| - |\mathbb{T}^0_{LIF}|}{|\mathbb{T}^0|}\frac{|\mathbb{T}^1| - |\mathbb{T}^1_{LIF}|}{|\mathbb{T}^1|}\right\} \\
&= \frac{1}{2}\left\{1 - \frac{2^{n_L} - \prod_{\substack{i=1\\ i\neq j}}^{n_T}(2^{n_{t_i}}-1)\left\{2^{n_{t_j}}-1\right\}}{2^{n_L} - \prod_{i=1}^{n_T}(2^{n_{t_i}}-1)}\right\} \\
P(mcdc_{LIF}) &= \frac{1}{2}\left\{1 - \frac{|\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}^{l_i\text{-}0}_{LIF}|}{|\mathbb{T}^{l_i\text{-}0}|}\frac{|\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}^{l_i\text{-}1}_{LIF}|}{|\mathbb{T}^{l_i\text{-}1}|}\right\} \\
&= \frac{1}{2}
\end{aligned}
$$

TABLE II

EXPRESSIONS FOR THE PROBABILITY OF LOF, LNF, LRF AND ORF.

Fault Class	P_{stmt}	P_{br}	P_{mcdc}
LOF	$\frac{\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)}{2^{n_L}}$	$1-\frac{2^{n_L}-\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)\left\{2^{n_{t_j}}-1\right\}}{2^{n_L}-\prod_{i=1}^{n_T}(2^{n_{t_i}}-1)}$	1
LNF	$\frac{2\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)}{2^{n_L}}$	$1-\left\{\left\{\frac{\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)\left\{2^{n_{t_j}}-1\right\}}{\prod_{i=1}^{n_T}(2^{n_{t_i}}-1)}\right\}\left\{\frac{2^{n_L}-\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)\left\{2^{n_{t_j}}-1\right\}}{2^{n_L}-\prod_{i=1}^{n_T}(2^{n_{t_i}}-1)}\right\}\right\}$	1
LRF	$\frac{\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)}{2^{n_L}}$	$\frac{1}{2}\left\{1-\frac{\left\{2^{n_L}-\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)\left\{2^{n_{t_j}}-1\right\}\right\}^2}{\prod_{i=1}^{n_T}(2^{n_{t_i}}-1)\left\{2^{n_L}-\prod_{i=1}^{n_T}(2^{n_{t_i}}-1)\right\}}\right\}$	$\frac{1}{2}$
$ORF[.]$	$\frac{\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)\left\{(2^{n}_{t_p}-1)+(2^{n}_{t_q}-1)\right\}}{2^{n_L}}$	$1-\left\{\frac{2^{n_L}-\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)\left\{2^{n_{t_j}}-2^{n_{t_p}}-2^{n_{t_q}}-1\right\}}{2^{n_L}-\prod_{i=1}^{n_T}(2^{n_{t_i}}-1)}\right\}$	1

D. ORF[+]

Disjunctive Operator Reference Fault (ORF[+]) occurs when a binary boolean operator '+' immediately following a term in the boolean expression is implemented as a conjunctive operator '.'.

Expressions: Let the '+' immediately following t_j in expression S is replaced to '.' in $S^{t_j}_{ORF[+]}$.

Criteria: For a test case **tc** to detect ORF[+], the following conditions should be satisfied:

1) $V_{tc}(t_i) = 0$ where $1 \leq i, j \leq n_T$, $i \neq j$ and $i \neq j+1$
2) $V_{tc}(t_j) \neq V_{tc}(t_{j+1}) \implies V_{tc}(S) \neq V_{tc}(S^{t_j}_{ORF[+]})$

Criterion 1 is the non masking criterion which states that all other terms in S other than t_j and t_{j+1} should evaluate to '0'. The second criterion states that the boolean value of the term t_j should not be equal to that of t_{j+1}. The number of test cases that satisfies the two criteria is given by

$$|\mathbb{T}_{ORF[+]}| = \prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1) + \prod_{\substack{i=1\\i\neq j+1}}^{n_T}(2^{n_{t_i}}-1)$$

Since all the tests generated from S that satisfies the two criteria evaluate S to '0', we have

$$\begin{aligned}
|\mathbb{T}^0_{ORF[+]}| &= 0\\
|\mathbb{T}^1_{ORF[+]}| &= \prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1) + \prod_{\substack{i=1\\i\neq j+1}}^{n_T}(2^{n_{t_i}}-1)\\
|\mathbb{T}^{l_i\text{-}0}_{ORF[+]}| &= 0\\
|\mathbb{T}^{l_i\text{-}1}_{ORF[+]}| &= \prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)\\
|\mathbb{T}^0| - |\mathbb{T}^0_{ORF[+]}| &= \prod_{\substack{i=1\\i\neq j\\i\neq j+1}}^{n_T}(2^{n_{t_i}}-1)\left\{(2^{n_{t_j}}-1)(2^{n_{t_{j+1}}}-1)\right\}\\
|\mathbb{T}^1| - |\mathbb{T}^1_{ORF[+]}| &= 2^{n_L} - \prod_{i=1}^{n_T}(2^{n_{t_i}}-1)\\
|\mathbb{T}^{l_i\text{-}0}| - |\mathbb{T}^{l_i\text{-}0}_{ORF[+]}| &= \prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)\\
|\mathbb{T}^{l_i\text{-}1}| - |\mathbb{T}^{l_i\text{-}1}_{ORF[+]}| &= 0
\end{aligned}$$

Probabilities: The probability values for detecting the ORF[+] fault by statement, branch and MCDC coverage metric are shown below:

$$\begin{aligned}
P(st_{ORF[+]}) &= \frac{\prod_{\substack{i=1\\i\neq j}}^{n_T}(2^{n_{t_i}}-1)+\prod_{\substack{i=1\\i\neq j+1}}^{n_T}(2^{n_{t_i}}-1)}{2^{n_L}}\\
P(br_{ORF[+]}) &= 1-\left\{\frac{\prod_{\substack{i=1\\i\neq j\\i\neq j+1}}^{n_T}(2^{n_{t_i}}-1)\left\{(2^{n_{t_j}}-1)(2^{n_{t_{j+1}}}-1)\right\}}{\prod_{i=1}^{n_T}(2^{n_{t_i}}-1)}\right\}\\
P(mcdc_{ORF[+]}) &= 1
\end{aligned}$$

The probability expressions for the other fault classes (LOF, LNF, LRF and $ORF[.]$) are summarized in Table IV-C.

V. RESULTS

DNF Expressions: The probability expressions derived in Section IV has three variables namely the total number of literals (n_L), the total number of terms (n_T) and the number of literals in each term n_{t_i}. In order to understand the effect of these variables on the probability of detecting a fault and coverage metric, we varied n_L between 1 and 20, and for each value for n_L, we generated DNF expressions with all possible combinations for n_T and n_{t_i}. The probability of detecting the eight fault classes by three different coverage metrics were evaluated for all the generated DNF expressions using the formulas in Section IV.

Metrics: The mean and coefficient of variation (CV) were calculated for the probabilities obtained for each fault class and coverage metric. The mean probability indicates the strength of the coverage metric in detecting the fault. CV is the ratio of standard deviation (SD) and mean and it measures the variation in the probabilities in detecting a fault. CV measures the consistency of the coverage metric. A coverage metric for a fault class that has a CV closer to zero indicates that the coverage metric is consistent and will detect the fault.

Anaylsis: In Table 3, we shows the mean and CV of the probabilities obtained for each fault class by statement, branch and MCDC coverage metric, respectively. The following observations can be made from the table:

TABLE III

SUMMARY PROBABILITY

Fault Class	Statement			Branch			MCDC		
	Mean	SD	CV	Mean	SD	CV	Mean	SD	CV
LIF	0.009	0.019	2.0	0.028	0.074	2.6	0.5	0	0
LNF	0.037	0.074	2.0	0.793	0.330	0.4	1.0	0	0
LOF	0.018	0.037	2.0	0.057	0.149	2.6	1.0	0	0
LRF	0.018	0.037	2.0	0.397	0.165	0.4	0.5	0	0
$ORF[.]$	0.181	0.166	0.9	0.412	0.199	0.5	1.0	0	0
$ORF[+]$	0.035	0.063	1.8	0.057	0.136	2.4	1.0	0	0
TNF	0.018	0.037	2.0	1.000	0.000	0.0	1.0	0	0
TOF	0.018	0.037	2.0	0.057	0.149	2.6	1.0	0	0

Statement Coverage: (i) The mean probability of detecting a fault by statement coverage is very low for any fault class.

(ii) The CV is very high for all fault classes, which indicates that statement coverage is inconsistent in detecting the faults.

Branch Coverage: (i) The mean probability of detecting a fault significantly increases for LNF, LRF, $ORF[.]$ and TNF when compared to statement coverage. The increase is attributed to selecting test cases that cause the DNF expression to evaluate to 0 and 1. (ii) Branch coverage always detect TNF. (ii) The mean and CV of the probabilities for detecting LIF, LOF, TOF and $ORF[+]$ are very similar to statement coverage. The correlation study [3] concluded that the omission fault class accounted for a large number of the bugs encountered in the microprocessor designs that were evaluated. Hence there is no benefit in using branch coverage over statement coverage for detecting such faults.

MCDC: (i) The probability of detecting a fault by MCDC is independent of the DNF expression.(i) MCDC always detects seven of the eight fault types analyzed in this paper. (ii) Insertion (LIF) and reference (LRF) faults are detected with a probability 0.5. The probability of detecting LIF and LRF is the product of the probability of generating a test case which can expose the fault (by satisfying the conditions 1.2 and 1.3 described in section IV-B) and the probability that the extra literal takes the value '0'. The test case that could expose the fault is present in the MCDC test suite, and hence the probability of detecting a fault, is the probability that the literal takes the value '0', which is 0.5, if we assume l to be equally likely to take values '0' and '1'. (iii) The SD is zero for all test cases, which indicates that MCDC is a consistent in detecting the faults.

TABLE IV
4-LITERAL & 20-LITERAL SCENARIOS

Fault Class	4 Literals			20 Literals		
	Stmt	Branch	MCDC	Stmt	Branch	MCDC
LIF	0.09	0.21	0.50	0.02	0.09	0.50
LNF	0.38	0.62	1.00	0.10	0.23	1.00
LOF	0.19	0.43	1.00	0.05	0.18	1.00
LRF	0.19	0.31	0.50	0.05	0.12	0.50
$ORF[.]$	0.38	0.67	0.50	0.35	0.49	1.00
$ORF[+]$	0.38	0.86	1.00	0.10	0.35	1.00
TNF	0.19	1.00	1.00	0.05	1.00	1.00
TOF	0.19	0.43	1.00	0.05	0.18	1.00

Table 4 shows the variability in the mean probability of detecting the eight fault classes in a DNF expression with 4 and 20 literals. Statement coverage has low probability (<0.5) for detecting any of the fault classes, even with just four literals in the DNF expression. The probability for branch coverage is high (>0.5) in detecting LNF, $ORF[.]$, $ORF[+]$ and TNF for an expression with four literals. However, the probability decreases (<0.5) with the increase in the number of literals in the expression. MCDC consistently performs well for all fault classes, and is independent of the number of literals in the expression.

From the above analysis, we can conclude that statement coverage and branch coverage is not adequate enough to detect the eight fault classes analyzed in this paper. MCDC is able to always detect 6 of the 8 fault classes analyzed in this paper and is a consistent coverage metric. This coupled with the linear increase in number of test cases with the number of literals in the DNF expression when compared to the exponential increase in exhaustive testing, makes it an attractive structural coverage metric to be used in microprocessor validation.

VI. Conclusion & Future Work

Coverage directed test generation is considered as a practical solution to the functional validation problem. In this paper, we address the question *Which code coverage metric is better in finding commonly found design bugs during microprocessor validation?*. To answer this question, we identified eight fault classes that are a good capture for modeling errors seen during microprocessor validation. We then performed a probabilistic analysis on three code coverage metrics namely, statement, branch and MCDC, to understand its ability to detect the eight fault types. From the analysis, we were able to conclude that average probability of statement coverage to detect the eight fault classes is very low. Branch coverage is significantly superior to statement coverage only in 4 of the 8 fault classes analyzed in the paper. Furthermore, their fault detection probability varies based on the size of boolean expression, making them inconsistent. MCDC always detects negation and omission faults and consistently detects insertion and reference faults with a probability 0.5. Therefore, we conclude that MCDC is a good metric and propose its application to microprocessor validation. As future work, we will extend our analysis to include multiple faults and coupling effect in the boolean expression.

References

[1] John J Chilenski, *An Investigation of three forms of the modified condition decision coverage (MCDC) criterion*, Tech. Report DOT/FAA/AR-01/18, U.S. Department of Transportation, Federal Aviation Administration, 2001.

[2] David Van Campenhout et al., *Collection and analysis of microprocessor design errors*, IEEE Design and Test **17** (2000), no. 4, 51–60.

[3] Deepak A. Mathaikutty et al., *Design fault directed test generation for microprocessor validation*, FERMAT Technical Report 2006-16, 2006.

[4] Kelly Hayhurst et al., *A Practical Tutorial on Modified Condition/Decision Coverage*, Report NASA/TM, 2001.

[5] Farzan Fallah, Srinivas Devadas, and Kurt Keutzer, *OCCOM: Efficient computation of observability-based code coverage metrics for functional verification*, Design Automation Conference, 1998, pp. 152–157.

[6] Kalpesh Kapoor and Jonathan P. Bowen, *A formal analysis of mcdc and rcdc test criteria*, Software Testing, Verification and Reliability **15** (2005), 21–40.

[7] Man F. Lau and Yuen T. Yu, *An extended fault class hierarchy for specification-based testing*, ACM Trans. Softw. Eng. Methodol. **14** (2005), no. 3, 247–276.

[8] M. Velev, *Collection of High-Level Microprocessor Bugs from Formal Verification of Pipelined and Superscalar Designs*, International Test Conference (ITC), 2003.

[9] A. White, *Comments on modified condition/decision coverage for software testing*, Proceedings of the IEEE Aerospace Conference **6** (2001), 2821–2828.