J Electron Test (2008) 24:203–222
DOI 10.1007/s10836-007-5020-8

On the Use of ZBDDs for Implicit and Compact Critical Path Delay Fault Test Generation

Kyriakos Christou · Maria K. Michael · Spyros Tragoudas

Received: 1 December 2006 / Accepted: 18 June 2007 / Published online: 8 January 2008
© Springer Science + Business Media, LLC 2007

Abstract A new framework for generating test sets with high test efficiency for path delay faults (PDFs) is presented. The proposed method is based on a data structure that can implicitly represent all sensitizable PDFs in a circuit, along with all their corresponding tests. A path and test implicit method to construct such a data structure, for various path sensitization types, is presented. It uses zero-suppressed binary decision diagram (ZBDD) representations of irredundant sum-of-products (ISOPs), and requires only a polynomial number of standard ZBDD operations. Consequently, an ATPG algorithm that can exploit the properties of the proposed structure to derive tests with maximal test efficiency is presented. The obtained experimental results on the ISCAS'85 and enhanced full-scanned version of the ISCAS'89 benchmarks demonstrate that the proposed framework is scalable in terms of test efficiency and can generate compact test sets for critical PDFs.

Keywords Zero-suppressed binary decision diagram · Path delay faults · Irredundant sum-of-products · Delay testing · Critical path delay faults · Compact test generation

Part of this work has appeared in "Towards finding path delay fault tests with high test efficiency using ZBDDs," Proc. of the ICCD'2005 and in "Implicit Critical PDF Test Generation with Maximal Test Efficiency," Proc. of DFT'2006.

Responsible Editor: N. A. Touba

K. Christou · M. K. Michael (✉)
Department of Electrical and Computer Engineering, University of Cyprus, Nicosia, Cyprus
e-mail: mmichael@ucy.ac.cy

K. Christou
e-mail: christou@ucy.ac.cy

S. Tragoudas
Department of Electrical and Computer Engineering, Southern Illinois University at Carbondale, Carbondale, USA
e-mail: spyros@engr.siu.edu

1 Introduction

The path delay fault (PDF) model is one of the most popular models used for delay testing, since it is the most accurate model that can detect both lumped and distributed delay defects. For practical reasons, PDF testing usually considers functionally sensitizable PDFs [1, 2, 4, 6, 8, 10–12, 18–20, 22–24, 27–29], even though it has been shown in [25] that a subset of the multiple PDFs, called the primitive PDFs, needs and suffices to be tested. Furthermore, due to the large number of faults, which is exponential to the circuit size in the worst case, often only the critical paths are considered [18, 24, 28]. A critical path is one with large delay. Traditionally, the delay of a path has been calculated based on discrete-valued models, such as the fixed and bounded delay models. More recently, statistical models are also proposed for the selection of critical paths [28]. Even when the problem is restricted to critical PDF testing, its complexity remains prohibitive, mainly because: (1) the number of critical PDFs to be considered can still be very large and (2) often, only a small number of the critical PDFs are sensitizable. For the above two reason, the problem of deriving compact test

sets for critical PDFs in an implicit manner is crucial, especially for path intensive circuits.

This work considers the PDF classification of [4], where faults can be tested robustly, non-robustly, or with functional sensitization. The proposed framework includes function-based formulations, with appropriate representations and test generation algorithms. In particular, it investigates the use of zero-suppressed binary decision diagrams (ZBDDs) [14] in compact and efficient PDF test generation, concentrating on critical PDFs. In order to derive a compact test set, tests with high test efficiency must be generated. *Test efficiency* (*TE*) is defined here as the *number of new* (*critical*) *PDFs detected by a test*. For each sensitization type, a Boolean function is formulated whose solution is the set of all targeted sensitizable PDFs along with their corresponding set of sensitization cubes (PDF tests). The function produced is represented by a ZBDD-based canonical data structure. Both, circuit paths and PDF tests are captured in a non-enumerative fashion.

The first challenge is to show how the tests and the corresponding sensitized PDFs can be mapped by a single Boolean function and represented appropriately and efficiently. For completeness and clarity, it is first shown how such a function can be constructed for all the PDFs in a circuit, not just the critical ones. In this case, the problem of exact identification of sensitizable PDFs (also, that of unsensitizable PDFs) is solved non-enumeratively. This is a major problem in PDF ATPG, as demonstrated in [4, 6, 10, 11, 18, 23, 27], among many others. Consequently, it is shown how the proposed data structure can be constructed for any subset of PDFs, such as the potentially critical PDFs. In this case, the proposed PDF-sensitization function is restricted such that only sensitizable PDFs from a targeted set of PDFs are included.

The motivation is to investigate how the generated data structure, containing all the targeted sensitizable PDFs and their corresponding tests, can be efficiently manipulated to derive a compact test set. A challenging task here involves exploiting the properties of the structure to generate a test with high *TE*, without enumerating any of the paths or tests represented by the structure. A major contribution of this work involves deriving a test with maximal *TE* in linear time to the size of the data structure, using appropriate graph traversals. Additional tests with high *TE* are derived by removing already detected PDFs from the data structure and repeating the method to derive another test with maximal *TE*. Thus, each generated test guarantees to detect a large number of PDFs that have not been already detected.

The recently proposed function-based method of [18] stores all sensitizable PDFs in a circuit using ZBDDs, but functions are maintained only for each sensitizable segment of the PDFs, as BDDs [3] (a segment is defined between any consecutive fan-out stems). This method examines pairs of path segments that are on common unsensitizable paths, similarly to the structural techniques of [6, 10] and [23],[1] which identify either such pairs of segments or lines. The advantage of [18] over these structural techniques is on the representation of the PDFs (or segments), such that it gains in computation time. However, the test efficiency of this method is very low, almost a test per fault, since the data structures it uses cannot assist in compact test pattern generation. Low test efficiency also exists with the structural method of [27], which identifies testable and untestable PDFs using an implication graph to generate test sets with very high fault coverage.

Methods that explicitly target the generation of compact test sets for PDFs have been proposed in [2, 7, 19, 22, 29]. The test compaction procedures of [2, 7, 19], use the concept of primary and secondary target faults. Once a test is found for a primary fault, it is expanded to detect one or more secondary faults. The level of compaction in these techniques depends greatly on the selection order of the primary and secondary faults. [22] finds maximal sets of potentially compatible faults. Even though they may not target all faults explicitly, the above methods remain enumerative since they are based on the principle of first targeting a single fault and then attempting to find one or more faults that can be tested mutually with the original fault. The methods of [29] and [7], which focus on critical paths, report considerably higher test efficiencies that the other enumerative methods, however, they both remain restrictive due to their path enumerative nature.

Non-enumerative ATPG methods, such as [8, 12, 20], were proposed to overcome the problem of path enumeration. [8, 20] are structural-based methods relying on graph theoretic arguments. Still, each of their generated test detects a very small number of PDFs. The recent function-based (BDD-based) non-enumerative tool of [12] outperforms both [8, 20] in terms of *TE*. Although this method offers scalability (*TE* does not drop much as coverage increases), it is not complete since it may not be able to achieve 100% coverage since it generates a test that sensitizes a number of PDFs simultaneously, however, the PDFs detected may not

[1]These methods estimated a lower bound on the number of unsensitizable PDFs, which can be used to indicate the completion of the ATPG process more accurately, however, they do not provide any specific guidance to the ATPG on path or test selection.

always be new. Furthermore, [12] cannot handle unsensitizable paths implicitly (i.e., cannot avoid targeting such paths). More importantly, none of the existing non-enumerative methods can handle critical PDF test generation in an absolute non-enumerative manner.

The rest of the paper is organized as follows. Section 2 gives necessary background and notation. Section 3 shows the proposed function formulation along with its representation can be derived, under each of the considered sensitization types. Section 4 extends the framework to consider only a subset of the PDFs, such as the potentially critical PDFs. The compact ATPG process is discussed in Section 5. Section 6 reports and discusses the obtained experimental results and Section 7 concludes the paper.

2 Preliminaries and Notation

Delay fault testing involves the application of a sequence of tests at the circuits primary inputs. A test is a pair of vectors, $t = (v^1, v^2)$, such that v^1 stabilizes the circuit lines to known values, and v^2 initiates and propagates appropriate transitions through circuit paths to the primary output(s). We consider the single path delay fault (PDF) model with the classification of [4], which categorizes PDFs as robustly testable, non-robustly testable, functionally sensitizable (multiple path sensitization criterion) and functionally unsensitizable. The latter type never affects the timing behavior of a circuit and, thus, functionally unsensitizable PDFs do not need to be considered during testing.

We use established decision diagrams to represent the proposed functions. Binary decision diagrams (BDDs) [3, 5], are canonical forms for Boolean function representation, where the absence of a variable on a path from the root of the diagram to the terminal-1 node is interpreted as a don't care (x). BDDs can also represent combinatorial sets, however, such sets can be more effectively represented using zero-suppressed binary decision diagrams (ZBDDs) [14], a variant of BDDs, where the absence of a variable is interpreted as a zero assignment. A very efficient application of ZBDD representation of sparse combinatorial sets is given in [17], where all the PDFs of path-intensive circuits can be represented compactly by a ZBDD. ZBDDs can also store Boolean functions by introducing some additional variables but without increasing the number of paths in the ZBDD [13]. Namely, two variables are introduced for each variable in the function, and an appropriate encoding protocol is activated so that when both variables are suppressed, the original variable is a don't care. A ZBDD where such pairs of variables are used for each original variable is a ZBDD-based representation of an irredundant sum-of-products (ISOPs) [13, 15, 21].

Finding sensitizable PDFs requires considering the sensitization conditions at the path off-inputs, which can be derived by examining the functionality of the circuit's lines. Thus, sensitization conditions can be represented using functions, referred to as *test functions*. [1] has shown BDD-based test functions per single PDF and [12] derived BDD-based test functions for sets of PDFs. A test function contains all tests that detect one (or a set) of PDFs. However, it does not contain any information on which PDF(s) is detected. Here, the goal is to find the sensitizable PDFs along with their tests and store them compactly in a common structure. The PDFs are represented using ZBDDs, for efficiency and compactness, as demonstrated in [17]. To incorporated the tests in the same data structure with the PDFs, we generate ZBDD-based ISOPs to represent the sensitization conditions (instead of BDDs, as in [1] and [12]. The resulting common structure is called an ISOPs/ZBDD because of the interpretation as far as variable appearances are concerned. Some variables participate in pairs and thus are interpreted according to the ZBDD-based ISOPs encoding whereas the remaining variables are interpreted as usual. Each node in this structure obeys standard ZBDD decomposition and reduction rules. Table 1 lists all the standard ZBDD operations used by the proposed methodology.

Next, we present basic notation used in this paper and some more details on ZBDD representations of PDFs and derivation of ZBDD-based ISOPs from standard BDDs.

The proposed function is expressed over two sets of variables, the *test* variables which encode the test cubes and the *path* variables which encode the PDFs. Let $\mathcal{P}$ define the set of path variables, PI the set of primary inputs, and L the set of circuit lines other than primary inputs. There is one path variable per line in L, and two path variables per line in PI to represent the rising and falling transitions on the primary inputs. Hence $|\mathcal{P}| =$

Table 1 Standard ZBDD operators used

Expression	Description
$!(S, v)$	Complement v for all combinations in set S
$S \cup Q$	Set union
$S \cap Q$	Set intersection
$S \backslash Q$	Set difference
$\exists(S, v)$	Existentially abstract variable v from set S
$\|S\|$	Number of combinations in set S
$\subset (S, v)$	Subset of set S such that $v = 1$

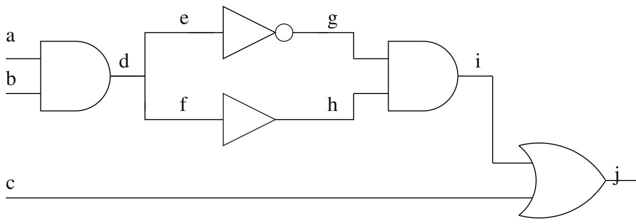


Fig. 1 Example circuit C_1

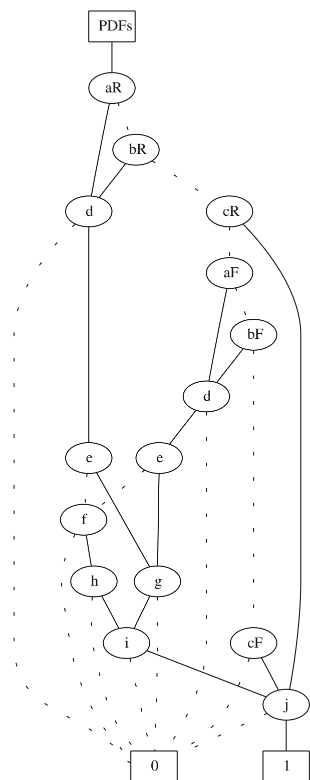
$|L| + 2 \times |PI|$. Consider the circuit in Fig. 1 which has seven internal lines $\{d, e, f, g, h, i, j\}$ and three primary inputs $\{a, b, c\}$. There are 13 path variables denoted by $\mathcal{P} = \{aR, aF, bR, bF, cR, cF, d, e, f, g, h, i, j\}$, where iR (iF) is the rising (falling) transition variable for primary input i. A PDF is encoded by a combination over the variables of $\mathcal{P}$. For example, the rising PDF on path $a - d - e - g - i - j$ is represented by $aR \cdot d \cdot e \cdot g \cdot i \cdot j$. Missing path variables assume a 0 value. The ZBDD representation of all the PDFs for the circuit C_1 in Fig. 1, is shown in Fig. 2. The variable ordering follows the topological order of the lines in the circuit. There are exactly ten paths from the root node aR to the terminal-1 node, one for each PDF in the circuit.

A ZBDD-based ISOPs representation is a compact and implicit cube set representation [13, 15, 21]. Converting a BDD to a ZBDD-based ISOPs involves defining two variables i_0 and i_1 per i variable in the BDD, such that $i = i_1 \cdot \overline{i_0}$ and $\overline{i} = \overline{i_1} \cdot i_0$. The combination $\overline{i_1} \cdot \overline{i_0}$ in the ZBDD implies that $i = x$ (don't care). Suppressed pairs of variables also imply the don't care value. Finally, $i_1 \cdot i_0$ is never allowed to appear since, based on the encoding, it implies that i has both 0 and 1 values at the same time. Let $\mathcal{T}$ define the set of test variables. For BDD-based test functions, the number of test variables used is two times the number of lines in PI, since for every line $i \in PI$, variables i^1 and i^2 are defined to represent values in test vectors v^1 and v^2, respectively [1, 12]. Converting the BDD to a ZBDD-based ISOPs, doubles the number of variables since for every $i^1(i^2)$, variables $i^1_0(i^2_0)$ and $i^1_1(i^2_1)$ are needed. Hence, $|\mathcal{T}| = 2 \times |PI| \times 2$. The circuit in Fig. 1 has 12 test variables denoted by $\mathcal{T} = \{a^1_1, a^1_0, a^2_1, a^2_0, b^1_1, b^1_0, b^2_1, b^2_0, c^1_1, c^1_0, c^2_1, c^2_0\}$, where the superscript denotes the vector the variable corresponds to (v^1 or v^2) and the subscript denotes variables i_0 or i_1 for a variable i.

Let $\mathcal{T}^1 \subset \mathcal{T}$ be the set of test variables corresponding to vector v^1, i.e., $\{i^1_0, i^1_1\} \in \mathcal{T}^1$, $\forall i \in PI$. Similarly we define $\mathcal{T}^2 \subset \mathcal{T}$ for vector v^2. $f^1_l(\mathcal{T}^1)$ and $f^2_l(\mathcal{T}^2)$ are the functions realized at a circuit line l expressed with respect to variables in $\mathcal{T}^1$ and $\mathcal{T}^2$, respectively. These functions are derived by generating the appropriate BDD per line and then using the procedure of [13] to derive their ZBDD-based ISOPs representation. Figure 3 shows the line functionalities using ISOPs-based ZBDDs for various lines of circuit C_1 of Fig. 1, with respect to variables in $\mathcal{T}^1$ for the first vector v^1. For clarity, the variable notation used in Fig. 3 is a bit simplified from the one presented in the previous paragraph. Thus, $a1(b1, c1)$ is actually $a^1_1(b^1_1, c^1_1)$ and $a0(b0)$ is $a^1_0(b^1_0)$, i.e., the superscript 1 is implied in these figures (all the figures in this paper follow this labelling, unless otherwise stated). Observe the ISOPs-based ZBDD of Fig. 3a. Variable $a1$ appears in its normal form and the missing variable $a0$ is implied to appear in a complemented form. Thus, $f^1_a(\mathcal{T}^1) = a1 = a1\overline{a0} = a$ (where a is the original variable in the BDD). A function $f^2_l(\mathcal{T}^2)$, for some line l, is identical to $f^1_l(\mathcal{T}^1)$ when every variable in $\mathcal{T}^2$ is substituted by its corresponding variable in $\mathcal{T}^1$. It is necessary to define different line functions (and hence, input variables) over the two time frames of (v^1, v^2), since an input can assume different values over the two time frames of a delay test.

We also define stability functions $S^0_l(\mathcal{T})$ and $S^1_l(\mathcal{T})$ per line l, to contain all solutions of (v^1, v^2) that bring a stable-at-0 and stable-at-1 value at l, respectively. Stability functions were introduced in [1] for single-input change tests (also used in [12]) and later generalized in [9] for multi-input change tests. The later method

Fig. 2 ZBDD with all PDFs of circuit C_1



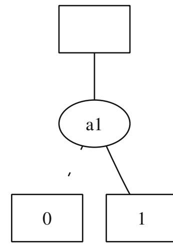


Fig. 3.a $f_a^1() = a1\overline{a0} = a$

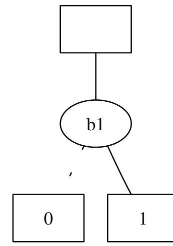


Fig. 3.b $f_b^1() = b1\overline{b0} = b$

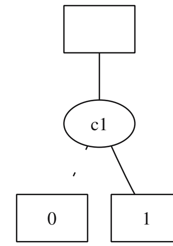


Fig. 3.c $f_c^1() = c1\overline{c0} = c$

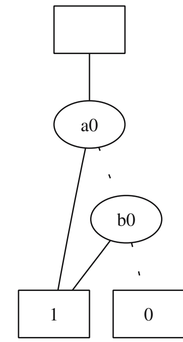


Fig. 3.d $f_g^1() = \overline{a1}a0 + \overline{b1}b0 = \overline{a} + \overline{b}$

Fig. 3 Line functionalities using ISOPs-based ZBDDs. **a** $f_a^1() = a1\overline{a0} = a$, **b** $f_b^1() = b1\overline{b0} = b$, **c** $f_c^1() = c1\overline{c0} = c$, **d** $f_g^1() = \overline{a1}a0 + \overline{b1}b0 = \overline{a} + \overline{b}$

is used to derive multi-input stability functions for all circuit lines in BDD format which are then converted to ZBDD-based ISOPs.

3 ISOP/ZBDD Graph for Sensitizable PDFs

This section discusses the desired function formulation, using a polynomial number, to the circuit size, of standard ZBDD operations. Both circuit paths and PDF tests are captured implicitly (no path or segment or test enumeration is performed), which gives a strictly non-enumerative solution to the problem of exact identification of (un)sensitizable PDFs. The focus of this Section is on path sensitization, how to capture the various sensitization conditions [4] in a single function, along with the sensitized paths. Given correct sensitization conditions, restricting appropriately to only critical paths is consequently considered in Section 4.

Assume a circuit line l with set of immediate predecessor lines denoted by $FI(l)$. Functions $G_l^R(\mathcal{T} \cup \mathcal{P})$ and $G_l^F(\mathcal{T} \cup \mathcal{P})$ denote the desired sensitization functions at some line l. We show how functions $G_l^R()$ and $G_l^F()$ are formulated at l, based on functions $G_i^R()$ and $G_i^F()$, $i \in FI(l)$. Let $ncv_g, cv_g \in \{0, 1\}$ denote the non-controlling and controlling value of gate g, respectively.

3.1 Non-robust PDF Sensitization

Under the non-robust sensitization criterion, path off-inputs at some gate g must be set to $x \rightarrow ncv_g$ under (v^1, v^2), irrespective of the type of transition propagated on the path on-input. This implies that only input values for v^2 need to be examined (values in v^1 are implied based on the values of v^2). Hence, only test variables in $\mathcal{T}^2 \subset \mathcal{T}$ are used.

Let i be a primary input. We start by defining functions $G_i^R()$ and $G_i^F()$ at the primary inputs in PI. Each $i \in PI$ is associated with two path variables, iR and iF, and two test variables, i_0^2 and i_1^2. The functionality of i, based on the ZBDD-based ISOPs representation, is given by $f_i^2() = \overline{i_0^2} \cdot i_1^2$ and its complement is $\overline{f_i^2()} = i_0^2 \cdot \overline{i_1^2}$. (The complementation operation is not the standard Boolean one. It is based on ZBDD-based ISOPs encoding.) To represent a transition on i, the appropriate path variable (iR or iF) and corresponding value at i in v^2 (value 0 for falling and 1 for rising transition) need to be included. The necessary function formulations at a primary input i are given below:

$$G_i^R\left(\mathcal{T}^2 \cup \mathcal{P}\right) = !\left(f_i^2(\mathcal{T}^2),\ iR\right),\ \ i \in PIs \tag{1}$$

$$G_i^F\left(\mathcal{T}^2 \cup \mathcal{P}\right) = !\left(\overline{f_i^2(\mathcal{T}^2)},\ iF\right),\ \ i \in PIs \tag{2}$$

The change operator (!) appends the appropriate variable to each cube in the set, and gives $G_i^R() = \overline{i_0^2} \cdot i_1^2 \cdot iR$ and $G_i^F() = i_0^2 \cdot \overline{i_1^2} \cdot iF$. Observe how the desired information is encoded in these functions. For example, $G_i^R = \overline{i_0^2}.i_1^2.iR$ gives the rising segments up to line i (which is just line i in this case encoded as iR in the cube) that is non-robustly sensitized by setting $i = 1$ ($\overline{i_0^2}.i_1^2 = i$) in v^2. The value of i is implied in v^1 to the 0 value, since iR encodes a rising transition. Consider the circuit C_1 of Fig. 1. The ISOPs/ZBDD graphs for the sensitization functions for some of C_1's input lines are given in Fig. 4. For example, in Fig. 4b $G_a^F() = a0.aF$. Variable $a1$ has a 0 value and, therefore, is suppressed. Here, the falling segment up to a line a (aF) is sensitized by setting a to 0 in v^2 ($\overline{a1}.a0 = \overline{a}$). For v^1, $a = 1$ is implied by the existence of variable aF in the cube.

Consider now an AND gate g with output line l and fanins $FI(l)$. PDF segments can be sensitized up to line l through any of the lines in $FI(l)$. Thus, any line in $FI(l)$ can be considered as an on-input with the remaining lines being off-inputs that must settle

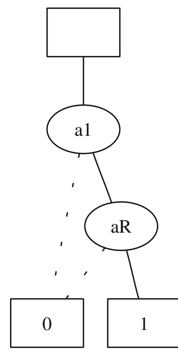


Fig. 4.a $G_a^R() = a1.aR$

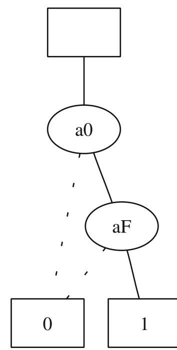


Fig. 4.b $G_a^F() = a0.aF$

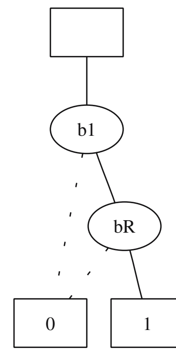


Fig. 4.c $G_b^R() = b1.bR$

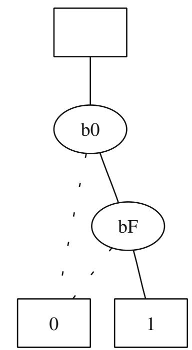
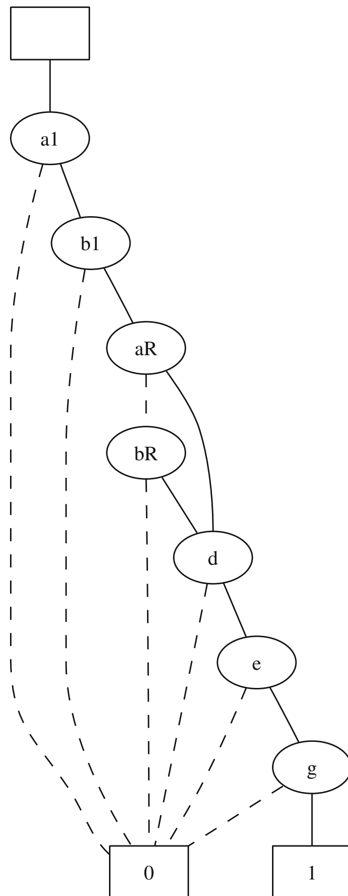


Fig. 4.d $G_b^F() = b0.bF$

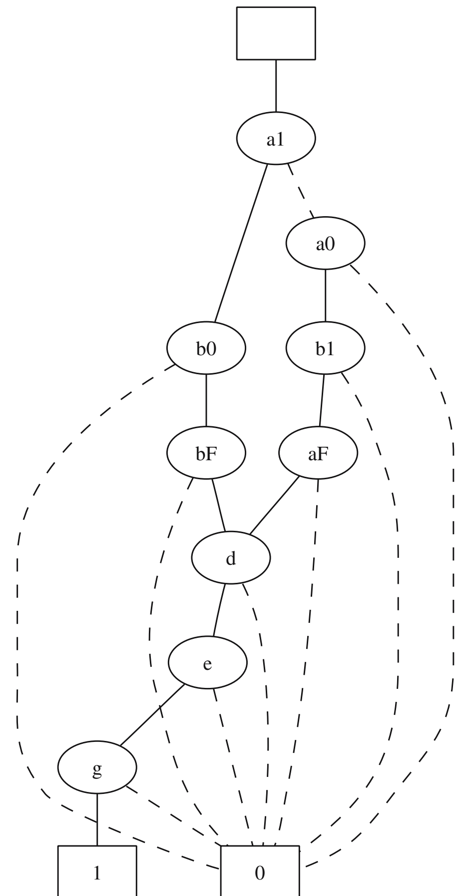
Fig. 4 Non-robust sensitization functions for primary inputs of circuit C_1 of Fig. 1. **a** $G_a^R() = a1.aR$, **b** $G_a^F() = a0.aF$, **c** $G_b^R() = b1.bR$, **d** $G_b^F() = b0.bF$

to ncv_g. Consider line $y \in FI(l)$ to be an on-input. Function $\bigcap_{x \in FI(l), x \neq y} f_x^2()$ will give all test cubes that allow all remaining off-inputs in $FI(l)$ to settle to $ncv_g = 1$ (given by $f_x^2()$ in the expression). Let $tr \in \{R, F\}$ denote a Rising or a Falling transition. Function $!((G_y^{tr}() \cap (\bigcap_{x \in FI(l), x \neq y} f_x^2())), l)$ will give all non-robustly sensitizable PDFs up to line l, through line y, along with the sensitization cubes. Every line in *FI(l)* is a possible on-input, thus function $G_l^{tr}()$ given by:

$$G_l^{tr}() = !\left(\left(\bigcup_{y \in FI(l)} \left(G_y^{tr}() \cap \left(\bigcap_{x \in FI(l), x \neq y} f_x^2()\right)\right)\right), l\right) \quad (3)$$



5.a $G_g^F()$



5.b $G_g^R()$

Fig. 5 Non-robust sensitization functions for line g of circuit C_1 of Fig. 1. **a** $G_g^F()$, **b** $G_g^R()$

For an OR gate g the off-inputs in $FI(l)$ need to settle to $ncv_g = 0$. Thus, it suffices to replace $f_x^2()$ with $\overline{f_x^2()}$ in Eq. 3, as given below:

$$G_l^{tr}() = !\left(\left(\bigcup_{y\in FI(l)}\left(G_y^{tr}()\cap\left(\bigcap_{x\in FI(l),x\neq y}\overline{f_x^2()}\right)\right)\right),\ l\right) \quad (4)$$

The $G_l^R()$ and $G_l^F()$ functions for the output line l of a NOT gate with input line y are given below.

$$G_l^R() = !\left(G_y^F(),\ l\right) \quad (5)$$

$$G_l^F() = !\left(G_y^R(),\ l\right) \quad (6)$$

An illustrating example is presented next. Consider the circuit of Fig. 1, which has five physical paths and, thus, ten PDFs. There are three primary inputs and a total of ten lines in the circuit. As already mentioned in Section 2, there are 13 path variables denoted by $\mathcal{P}$. Since non-robust sensitization is considered, $\mathcal{T} = \mathcal{T}^2$. For clarity, variable $i_0^2(i_1^2)$ is denoted by $i0(i1)$ in the figures. In this example, there are six test variables in $\mathcal{T}$ ={*a1, a0, b1, b0, c1, c0*}. Figure 5 shows the falling and rising sensitizable PDF segments up to the internal line g of the circuit. All sensitizable segments up to g, that arrive at g with a falling transition, are shown in Fig. 5a. Only two such segments exists, $aR-d-e-g$ and $bR-d-e-g$. The ISOPs/ZBDD graph in Fig. 5a contains two paths from the root node to the terminal-1 node, $\{a1.b1.aR.d.e.g, a1.b1.bR.d.e.g\}$, one per existing sensitized segment. Furthermore, the graph contains all sensitization cubes for each of these two segments, given by $a1b1 = a1\overline{a0}b1\overline{b0} = ab = 11$ for v^2. For v^1, $a=0$ for segment $aR.d.e.g$ and $b=0$ for $bR.d.e.g$ (implied by the aR and bR variables, respectively). Thus, $(v^1, v^2) = (a^1b^1c^1, a^2b^2c^2) = (0xx,11x)$ is the complete set of tests for $aR.d.e.g$. Similarly, $(v^1, v^2) = (x0x, 11x)$ for $bR.d.e.g$.

The ISOPs/ZBDD for the sensitization functions for complete PDFs, derived only after the output line j is processed, are given in Fig. 6. All sensitizable PDFs, arriving at the output line j with a falling transition, are contained in the graph of Fig. 6a (Fig. 6b). Five (one) such PDFs exists for C_1 and exactly 5 (1) paths from the root node of $G_j^F()$ ($G_j^R()$) to the terminal-1 node exist. The circuit contains 4 non-robustly unsensitizable PDFs ($aF.d.e.g.i.j, bF.d.e.g.i.j, aR.d.f.h.i.j, bR.d.f.h.i.j$). All of these PDFs arrive with a rising transition to one of the input lines of the AND gate with output line i. The fact that they are unsensitizable

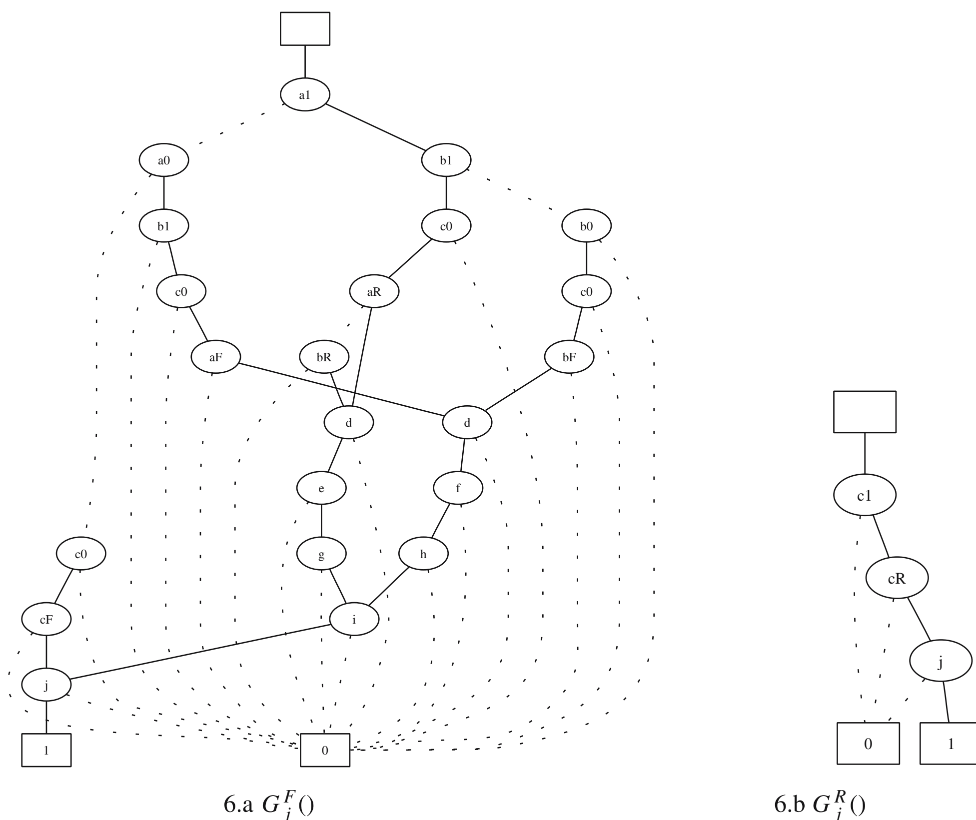


Fig. 6 Non-robust sensitization functions for output line j of circuit C_1 of Fig. 1. **a** $G_j^F()$, **b** $G_j^R()$

is recognized when forming $G_i^F()$, which equals to the constant function 0 implying that no PDF segment can be sensitized with a rising transition on line i. This occurs since none of the necessary sensitization conditions imposed by the function formulation is satisfied and, implicitly, all PDF segments with a rising transition at line i are dropped (removed) from the corresponding ISOPs/ZBDD. In the case where only a subset of segments is identified as unsensitizable, the function formulation ensures that only the unsensitizable segments are dropped, as it happens in $G_j^R()$ of Fig. 6b.

3.2 Other PDF Sensitization Types

For robust and functional sensitization, certain input values in both v^1 and v^2 vectors need to be explicitly fixed [4]. Thus, the set of $\mathcal{T} = \mathcal{T}^1 \cup \mathcal{T}^2$ test variables is used. For a primary input i, the $G_i^R()$ and $G_i^F()$ functions for either robust or functionally sensitizable PDFs are given below:

$$G_i^R(\mathcal{T} \cup \mathcal{P}) = !\left(\overline{f_i^1()} \cap f_i^2(),\ iR\right), \quad i \in PIs \qquad (7)$$

$$G_i^F(\mathcal{T} \cup \mathcal{P}) = !\left(f_i^1() \cap \overline{f_i^2()},\ iF\right), \quad i \in PIs \qquad (8)$$

In this case, the sensitization cubes for a rising transition on input i are given in $\overline{f_i^1}.f_i^2 = i_0^1.\overline{i_1^1}.i_0^2.\overline{i_1^2} = \overline{i^1}.i^2$, which denotes that $i = 0$ in v^1 and $i = 1$ in v^2. Similarly, for a falling transition on i, $f_i^1.\overline{f_i^2}$ gives $i = 1$ in v^1 and $i = 0$ in v^2.

When a $cv_g \rightarrow ncv_g$ transition is propagated on a PDF on-input, the off-input sensitization criteria are identical for all types of sensitization [4]. Thus, Eqs. 3 and 4 also hold for robust and functional sensitization. When the propagating on-input transition is $ncv_g \rightarrow cv_g$,

Fig. 7 The sensitization algorithm

INPUT: Circuit C, Sensitization Type ST
OUTPUT: ISOPs/ZBDD for $G_{circuit}()$
% Gate Types (GT) = {BUFF, NOT, AND, NAND, OR, NOR}
% Sensitization Types (ST) = {R, NR, FS}
1: Declare set of test variables $\mathcal{T} = \mathcal{T}^1 \cup \mathcal{T}^2$
    $\mathcal{T}_i^2 = \{i^2 \mid \forall i \in PI\}$
    if (ST == (R or FS)) **then** $\mathcal{T}_i^1 = \{i^1 \mid \forall i \in PI\}$
    else $\mathcal{T}^1 = \emptyset$
2: Declare set of path variables $\mathcal{P}$
    $\mathcal{P} = \{iR, iF, l \mid \forall i \in PI \text{ and } \forall l \in L\}$
3: Build BDDs for line functionalities $f_l^1(\mathcal{T}^1)$, $f_l^2(\mathcal{T}^2)$ $\forall l \in (L \cup PI)$
4: **if** (ST == (R or FS)) **then** Build BDDs for stability functions $S_l^0(\mathcal{T})$, $S_l^1(\mathcal{T})$, $\forall l \in (L \cup PI)$
5: Convert all BDDs to ISOP-based ZBDDs
% Traverse C in topological manner
6: **for each** line $l \in (L \cup PI)$
7: g = gate that drives line l
8: $FI(l)$ = fanin list of g
9:   **if** ($l \in PI$) **then**
10:   **if** (ST == NR) **then** use Eq. 1 and 2 for $G_l^R()$ and $G_l^F()$
11:   **else if** (ST == (R or FS)) **then** use Eq. 7 and 8 for $G_l^R()$ and $G_l^F()$
12:   **else if** (GT(g) == BUFF) **then** $G_l^R() = G_j^R()$ and $G_l^F() = G_j^F()$, $j \in FI(l)$
13:   **else if** (GT(g) == NOT) **then** $G_l^R() = G_j^F()$ and $G_l^F() = G_j^R()$, $j \in FI(l)$
14:   **else if** (GT(g) $\in$ {AND, NAND, OR, NOR}) **then**
15:   **if** (($cv_g \rightarrow ncv_g$) or (ST == NR)) **then**
16:   **if** (GT(g) == (AND or NAND)) **then** use Eq. 3 for $G_l^{tr}()$, $tr \in$ {R, F}
17:   **if** (GT(g)== (OR or NOR)) **then** use Eq. 4 for $G_l^{tr}()$, $tr \in$ {R, F}
18:   **else if** (($ncv_g \rightarrow cv_g$) and (ST == R)) **then** use Eq. 9 for $G_l^{tr}()$, $tr \in$ {R, F}
19:   **else if** (($ncv_g \rightarrow cv_g$) and (ST == FS)) **then** use Eq. 10 for $G_l^{tr}()$, $tr \in$ {R, F}
20:   **if** g is an inverting gate **then**
            $G_{temp}() = G_l^R()$, $G_l^R() = G_l^F()$ and $G_l^F() = G_{temp}()$
21: $G_{circuit}(\mathcal{T} \cup \mathcal{P}) = \bigcup_{O \in PO} \left(G_O^R() \cup G_O^F()\right)$

robust sensitization requires all off-inputs to be stable at the non-controlling value, while functional sensitization allows any value other than stable at the controlling value at the off-inputs. Stability functions are used in order to express the necessary off-input constraints. For robust sensitization, function $G_l^{tr}()$ for any non-inverting gate g with output line l under $tr = ncv_g \rightarrow cv_g$, is given by:

$$G_l^{tr}() = !\left(\left(\bigcup_{y \in FI(l)} \left(G_y^{tr}() \cap \left(\bigcap_{x \in FI(l), x \neq y} S_x^{ncv_g}()\right)\right)\right), l\right) \quad (9)$$

In Eq. 9, $S_x^{ncv_g}()$ denotes the stable at non-controlling value function at line x, which is the $S_x^0()$ function for an OR gate or the $S_x^1()$ function for an AND gate. Similarly to Eq. 9, function $G_l^{tr}()$ for functional sensitization is defined below, for $tr = ncv_g \rightarrow cv_g$:

$$G_l^{tr}() = !\left(\left(\bigcup_{y \in FI(l)} \left(G_y^{tr}() \cap \left(\bigcap_{x \in FI(l), x \neq y} \overline{S_x^{cv_g}}()\right)\right)\right), l\right) \quad (10)$$

Function $S_x^{cv_g}()$ denotes the stable at controlling value function at line x, which is the $S_x^1()$ function if g is an OR gate or the $S_x^0()$ function if g is an AND gate.

3.3 ISOPs/ZBDD for All Sensitizable PDFs

The algorithm for constructing the ISOPs/ZBDD traverses the circuit in a topological order, starting from the primary inputs. When a circuit line l is visited, its corresponding sensitization functions are generated, based on the previously generated sensitization functions of the line's immediate predecessor lines. The number of standard ZBDD operations per line l is $(n-1)^2$, where n is the number of immediate predecessor lines of l (gate fanin), and thus polynomial.

Let PO define the set of all primary outputs. After all circuit lines are processed, the union of all primary output line functions, $\bigcup_{l \in PO} G_l^{tr}()$, gives the function that contains all sensitizable PDFs and their corresponding test cubes. Let this function be denoted by $G_{circuit}()$. The basic steps of the proposed algorithm are listed in the pseudocode of Fig. 7. The input parameters are the circuit under consideration, C, and the desired sensitization type, $ST = \{R, NR, FS\}$ for robust, non-robust and functional sensitization, respectively. The set of internal lines is denoted by L.

Lines 1–5 list necessary preprocessing steps to define all Boolean variables and construct line functionalities as BDDs, which are then converted to ISOPs-based ZBDDs (line 5) using the method of [13]. If the examined sensitization type is either R or FS, then the necessary stability functions are also created, first as BDDs (as in [9, 12]) and then converted to an ISOPs-based ZBDD format. The ISOPs/ZBDD graphs (G_l^R and G_l^F) per line l are constructed based on the type of sensitization (ST) considered and the type of the gate driving l, as outlined in lines 9–20. The pseudocode refers to Eqs. 1–10, for each appropriate case, which give the exact ZBDD operations performed.

To find the ZBDD that contains only all sensitizable PDFs, denoted by $P_{circuit}()$, suffices to Existentially abstract all variables in $\mathcal{T}$ from $G_{circuit}()$:

$$P_{circuit}(\mathcal{P}) \;=\; \underset{\forall v \in \mathcal{T}}{\exists} (G_{circuit}(),\; v)$$

$|P_{circuit}()|$ will give the exact number of sensitizable PDFs in the circuit. The ZBDD with all unsensitizable PDFs is derived by taking the set difference of the ZBDD representing all PDFs with $P_{circuit}()$.

The function that contains only all tests, denoted by $T_{circuit}()$, can also be derived by existentially abstracting all variables in $\mathcal{P}$ from $G_{circuit}()$.

Figure 8 shows the ISOPs/ZBDD graph for $G_{circuit} = G_j^F() \cup G_j^R()$ of Fig. 1, for the non-robust case. Observe the paths from the root to the terminal-1 node, implying that there are six sensitizable PDFs that reach output j, 5 with a falling transition and 1 with a rising transition at j. Figure 9 shows the ZBDD that contains only the sensitizable PDFs, after all test variables are existentially abstracted. The number of sensitizable PDFs can

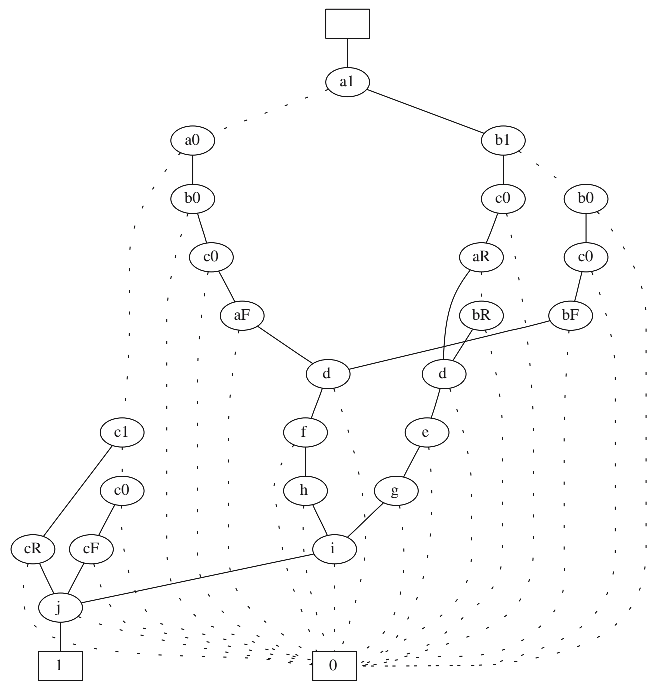


Fig. 8 $G_{circuit}()$ for circuit C_1 of Fig. 1

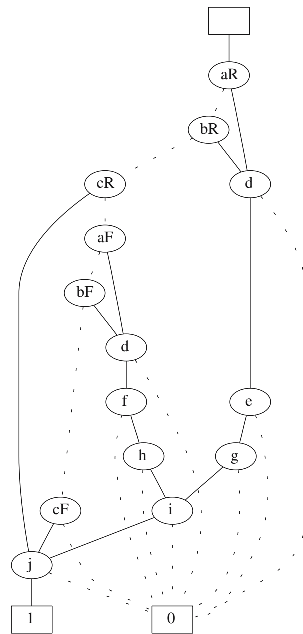


Fig. 9 $P_{\text{circuit}}()$ for circuit C_1 of Fig. 1

be calculated using the cardinality operator ($|P_{\text{circuit}}()|$), which is linear to the size of the graph.

4 Finding Critical Sensitizable PDFs

The previous Section presented how the sensitizable PDFs and their tests can be derived and stored in an ISOPs/ZBDD graph. To consider critical PDF test generation, it is necessary to generate an ISOPs/ZBDD which contains only the critical faults. We show how the sensitization function of Section 3 can be restricted to only contain critical PDFs. This is achieved in a non-enumerative manner by appropriate manipulation of the ISOPs/ZBDD per circuit line, while taking under consideration a selected set of potentially critical PDFs represent by a ZBDD. A potentially critical PDF is not necessarily sensitizable.

Any previously proposed method for critical path selection, under various delay models, can be considered. Consequently, the selected paths can be encoded efficiently in ZBDD format. Depending on the path selection methodology used the encoding can be done in an enumerative or non-enumerative manner. In [18] for example, it is shown how a ZBDD containing all potentially critical PDFs, under the bounded delay model, can be derived without any path enumeration. The same method applies under the other traditional gate delay models, such as the fixed and min-max delay models. The remaining of this Section focuses on reformulating the ISOPs/ZBDD sensitization function to only include critical PDFs.

Let $\mathcal{Z}$ denote the ZBDD containing the set of potentially critical PDFs. When considering critical paths, three major issues need to be considered when forming the sensitization function:

1. How to only consider potentially critical circuit lines. A potentially critical line is one that is at least on one potentially critical PDF. This is achieved by appropriate examination of the paths in $\mathcal{Z}$.
2. How to identify the unsensitizable PDFs in $\mathcal{Z}$ and exclude them from the generated sensitization function. This is done automatically since the sensitization function formulation of Section 3, considers the various sensitization criteria and discards unsensitizable segments implicitly.
3. How to avoid introducing segments/PDFs that are not included in $\mathcal{Z}$. This problem appears when processing fanout stem lines that drive several branches, and it is common in methods that process paths in an non-enumerative manner [8, 17, 20].

We first address issue (1) and show the formulation of the critical PDF sensitization function for fanout-free circuits. Consequently, we discuss issue (3) and present the solution. Issue (2) has already been addressed in the function formulation of Section 3. Without any loss of generality, we describe how to derive the ISOPs/ZBDD for critical non-robustly sensitizable PDFs. The formulation extensions for the robust and functional sensitization case can be derived in a similar fashion as in Section 3. The underlying operations necessary for the critical path extension are independent to the type of the PDF sensitization considered.

4.1 Fan-Out Free Circuits

Let $G_l^{tr}()$, $tr = \{R, F\}$, denotes a rising or falling sensitization function containing all *critical* PDF segments up to line l, along with all their corresponding sensitization cubes.

When at some a line l, it is necessary to determine if l is a potentially critical line. If not, $G_l^{tr}() = \emptyset$. To determine if l is potentially critical, it suffices to find the subset of $\mathcal{Z}$ such that $l = 1$, given by $\mathcal{Z}_l =\subset (\mathcal{Z}, l)$ (see Table 1 for the standard ZBDD subset operation). If $\mathcal{Z}_l = \emptyset$ then l is not critical. Otherwise, $\mathcal{Z}_l$ will contain all potentially critical PDFs through line l. According to the ZBDD subset operation, the variable of l will not appear in $\mathcal{Z}_l$, since it is cofactored. The next step is to determine which of the immediate predecessor lines of l, given in *FI(l)*, are potentially critical. This is also determined using the subset operation. A line $y \in$ *FI(l)* is potentially critical if $\mathcal{Z}_y =\subset (\mathcal{Z}, y) \neq \emptyset$. Let *CI(l)* $\subseteq$

FI(l) denote the set of all immediate potentially critical predecessors of l.

Function $G_l^{tr}()$ is formulated with respect to the critical sensitization functions of its predecessor lines in *CI(l)*. The main idea here is to be able to find all potentially critical PDF segments up to line l and through some line $y \in CI(l)$. Once this is derived, the sensitization conditions of the underlying sensitization type can be incorporated for the off-inputs, in order to drop any unsensitizable segments from the function. Thus, for the case of fanout-free circuits, the sensitization function for all critical PDF segments up to the output line l of an AND gate, is given by:

$$G_l^{tr}() = !\left(\left(\bigcup_{y\in CI(l)}\left(G_y^{tr}()\cap\left(\bigcap_{x\in FI(l),x\neq y} f_x^2()\right)\right)\right), l\right) \quad (11)$$

This formulation is identical to that of Eq. 3 in Section 3, with the exception that only critical predecessor lines of l (*CI(l)*) are considered. The off-inputs sensitization conditions are enforced on all immediate predecessor in *FI(l)*.

4.2 Treating Fan-Out Stems

When considering circuits with fanout sections, the formulation of Eq. 11 can introduce non-critical PDFs in $G_l^{tr}()$. We discuss why this occurs with the help of an example. Consider circuit C_2 in Fig. 10. It contains 12 PDFs. Let's consider only the rising PDFs. Let $\mathcal{Z}()$ contain two potentially critical PDFs, $\mathcal{Z}() = aR.c.d.f + bR.c.e.g$. For the primary inputs $G_a^R() = f_a^2().aR$ and $G_b^R() = f_b^2().bR$. At line c, both segments $aR - c$ and $bR - c$ are included, since they are both critical. Thus, $G_c^R() = f_a^2().f_b^2().aR.c + f_a^2().f_b^2().bR.c$. Up to this point, all functions contain the correct information.[2] Consider lines f and g. Observe that only a subset of $G_c^R()$ is needed when computing the functions for these two lines, since $G_f^R()$ should only include path $aR - c - f$ and $G_g^R()$ should only contain path $bR - c - g$. However, based on the formulation of Eq. 11, $G_f^R() = f_a^2().f_b^2().f_i^2().aR.c.f + f_a^2().f_b^2().f_i^2().bR.c.f$, which includes the non-critical path $bR - c - f$. Similarly, $G_g^R()$ will include the non-critical path $aR - c - g$.

This problem is alleviated by appropriate manipulations on $\mathcal{Z}()$. Consider again a line l and corresponding $\mathcal{Z}_l =\subset (\mathcal{Z}, l)$ which contains all potentially critical paths through line l. Removing from $\mathcal{Z}_l()$ all variables

[2]Branch lines and single input gates have been explicitly represented mainly for illustration and clarification purposes. We do not use them in our implementation.

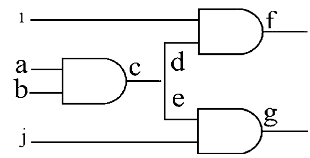


Fig. 10 Example circuit C_2

corresponding to lines not driving l, gives all potentially critical PDF segments up to l. This is done using the existential abstraction operator, as given below. The set of all lines not driving line l is denoted by *ND(l)*, and it can be derived via a single circuit traversal.

$$\mathcal{Z}_{l,seg}() = \exists_{v\in ND(l)}(\mathcal{Z}_l, v)$$

Let $G_y^{tr}()$, $y \in CI(l)$, be the function with all critical PDF segments up to y. When computing the critical sensitization function for line l, it is necessary to only consider those critical segments in $G_y^{tr}()$, and corresponding tests, that also pass through line l. This is achieved by computing the intersection of $G_y^{tr}()$ with $\mathcal{Z}_{l,seg}()$. $\mathcal{Z}_{l,seg}()$ must be extended with all test variables, otherwise the outcome of the intersection will be the empty set. This occurs because $G_y^{tr}()$ also contains the test cubes per critical segment. In order to maintain the correctness of the test set, all test variables are incorporated in $\mathcal{Z}_{l,seg}()$ at their don't care state, as shown below:

$$\mathcal{Z}_{l,seg}^X() = !(!(\mathcal{Z}_{l,seg}(), \overline{v_0}), \overline{v_1}), \forall v \in \mathcal{T}$$

The intersection will give the ISOPs/ZBDD with all critical segments up to line l through line y, along with the sensitizatizing tests up to line y. Consequently, the sensitization function for all critical segments up to the output line l of an AND gate, is given by:

$$G_l^{tr}() = !\left(\left(\bigcup_{y\in CI(l)}\left(\left(\mathcal{Z}_{l,seg}^X()\cap G_y^{tr}()\right)\cap\left(\bigcap_{x\in FI(l),x\neq y} f_x^2()\right)\right)\right), l\right) \quad (12)$$

The functions for other gate types and sensitization criteria are derived in a similar manner as in Eq. 12. Continuing with the example of Fig. 10, $\mathcal{Z}_{f,seg}^X() = aR.c$ (branch variable d can be ignored). $G_c^R() \cap \mathcal{Z}_{f,seg}^X() = aR.c$ removes segment $bR.c$ and, thus, $G_c^R() = f_a^2().f_b^2().f_i^2().aR.c.f$ is now correct.

The following example illustrates the new operations introduced in this subsection. Circuit C_3 of Fig. 11 contains 12 PDFs. Assume that all paths that pass through three or more gates are considered critical. In this case, $\mathcal{Z}()$ contains eight potentially critical PDFs, as shown in Fig. 12. These are $\{bR - f - j - l - m$, $bF - f - j - l - m$, $cR - j - l - m$, $cF - j - l - m$,

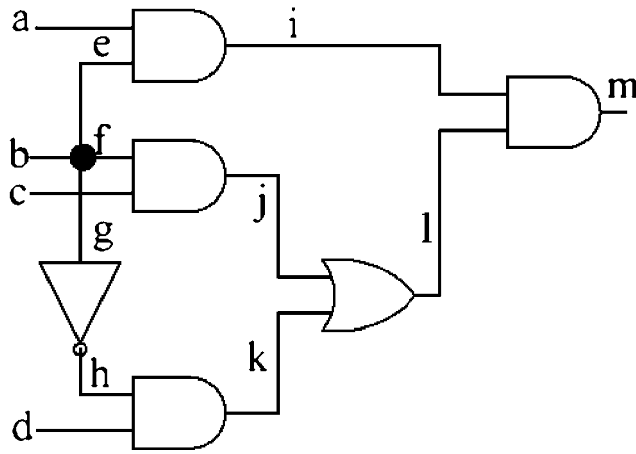


Fig. 11 Example circuit C_3

$bR-g-h-k-l-m$, $bF-g-h-k-l-m$, $dR-k-l-m$, $dF-k-l-m\}$. The potentially 4 critical PDFs through line k are given in $\mathcal{Z}_k()$ in Fig. 13. Variable k does not appear in Fig. 13, since it is cofactored by the subset operation $\mathcal{Z}_k() = \subset(\mathcal{Z}, k)$. The ZBDD with all PDF segments up to k (excluding k) is given in $\mathcal{Z}_{k,seg}()$ in Fig. 14. The ISOPs/ZBDD for all critical PDFs is shown in Fig. 16. It contains only 4 critical PDFs $\{bR-f-j-l-m$, $cR-j-l-m$, $cF-j-l-m$, $bR-g-h-k-l-m\}$. The remaining 4 PDFs in $\mathcal{Z}()$ are implicitly identified as non-sensitizable and are not included. Figure 15 shows the ISOPs/ZBDD for all sensitizable PDFs in C_3 which is actually a superset of Fig. 16.

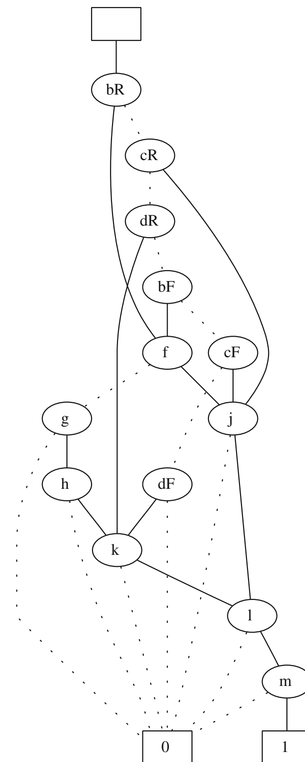


Fig. 12 $\mathcal{Z}()$ for C_3 of Fig. 11

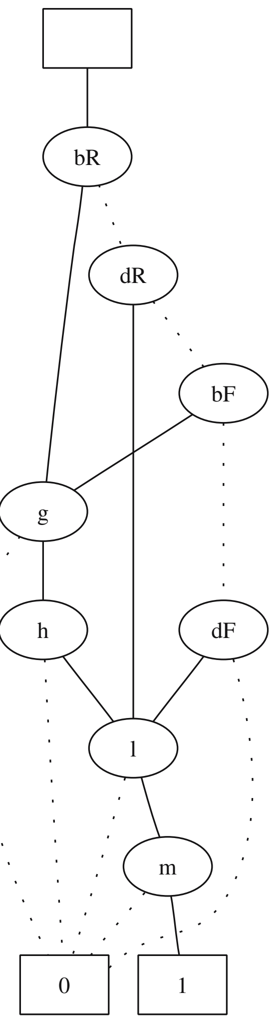


Fig. 13 $\mathcal{Z}_k()$ for C_3 of Fig. 11

The critical PDF sensitization algorithm follows the same basic steps outlined in Fig. 7. In addition, it accepts or creates the ZBDD of the potentially critical PDFs $\mathcal{Z}()$, and, before forming the critical sensitization function for a line l, it computes $\mathcal{Z}^X_{l,seg}()$. Equation 12 replaces Eq. 3 in Fig. 7. All other equations are modified appropriately in a similar manner as Eq. 12.

5 Generation of Tests with Maximal Test Efficiency

This section discuses the generation of a compact test set by generating tests with high test efficiency. It is shown how a test with maximum test efficiency, under the given ordering of variables in the ISOPs/ZBDD (i.e., changing the variable ordering can lead to different TE), is derived. Consequently, additional tests with high test efficiency are derived by appropriately

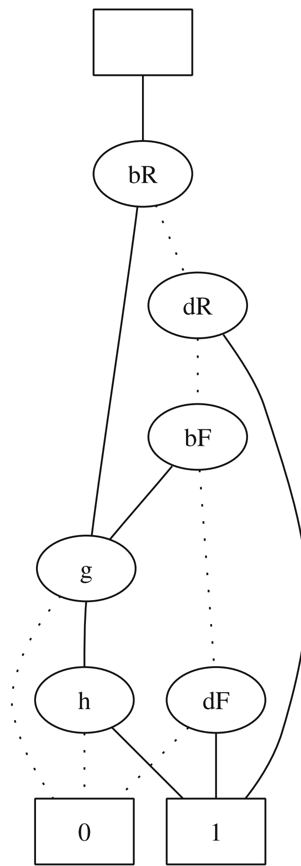


Fig. 14 $\mathcal{Z}_{k,seg}()$ for C_3 of Fig. 11

removing the detected PDFs and repeating the process. Thus, each generated test detects a large number of PDFs that have not been detected by already generated tests. No unsensitizable PDFs are targeted in this process.

The challenging task in finding a compact test set is finding a test with high test efficiency. Therefore, the first part of this Section is focused on this task, while the second part describes how additional tests with the same property can be derived.

5.1 Generation of the $\mathcal{T}$-graph

The basic idea is centered on finding one test in the ISOPs/ZBDD graph that shares the largest number of paths from the root to the terminal-1 node. This amounts to finding a set of test nodes in the graph that cover the maximum number of such paths.

The ISOPs/ZBDD nodes are classified as test nodes ($\mathcal{T}$-node) if they correspond to variables in $\mathcal{T}$ or path nodes ($\mathcal{P}$-node) if they correspond to variables in $\mathcal{P}$. Observe that, in decision diagrams such as the one considered here, several nodes in the graph can correspond to the same variable. Let the ISOPs/ZBDD graph be denoted by G, and $N^{\mathcal{T}}$ denote all $\mathcal{T}$-nodes in G. A graph that contains all $\mathcal{T}$-nodes in the original ISOP/ZBDD graph, plus the terminal-1 node, is derived. In this graph, referred to as the $\mathcal{T}$-graph, an edge from a $\mathcal{T}$-node v to a $\mathcal{T}$-node (or terminal-1 node) u exists if u is the first $\mathcal{T}$-node reachable from v through some path in G. This information for all $\mathcal{T}$-nodes can be found via a single traversal of G. Furthermore, a weight $w(v \rightarrow u)$ is defined per edge, to be the number of paths from v to u in G. The weights can also be calculated via a graph traversal. Without any loss of generality, it is assumed that the root of G is a $\mathcal{T}$-node (this always holds with variable reordering). The $\mathcal{T}$-graph has a single source, the root of G, and a single destination, the terminal-1 node. A post-processing step is necessary after all edges and weights have been derived to remove $\mathcal{T}$-nodes (and their incoming edges), other than the terminal-1 node, in the $\mathcal{T}$-graph that have no successors. Such nodes do not have any paths to the terminal-1 node in G, which implies that they are not in the function on-set. The basic steps of the algorithm that builds the $\mathcal{T}$-graph are listed in Fig. 17. V is the set of nodes and E the set of edges of the generated $\mathcal{T}$-graph.

As an example, consider circuit C_3 of Fig. 11. The ISOPs/ZBDD for all (critical) PDFs in C_3 was given in Fig. 15 (Fig. 16). The $\mathcal{T}$-graph for each of these case is shown in Figs. 18 and 19, respectively. All $\mathcal{T}$-nodes and the terminal-1 node from the ISOPs/ZBDDs appear in the $\mathcal{T}$-graphs (ignore the numbers in the

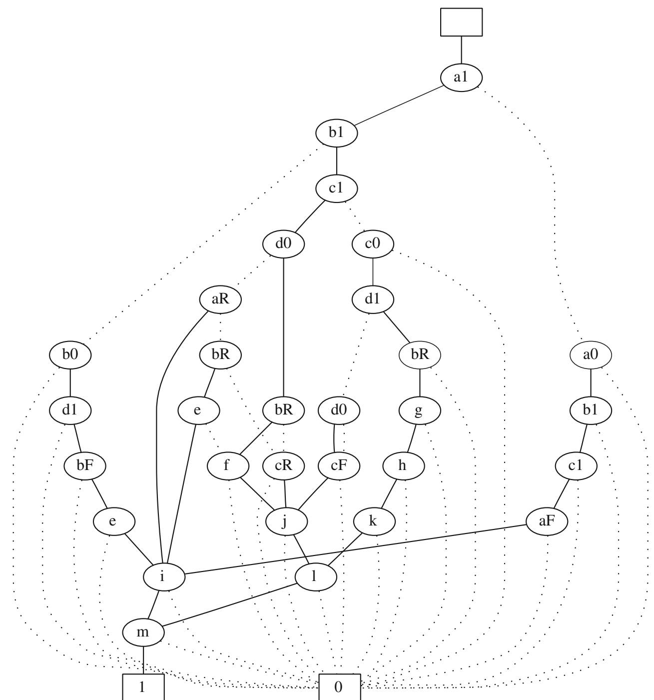


Fig. 15 ISOPs/ZBDD for all sensitizable PDFs in C_3 of Fig. 11

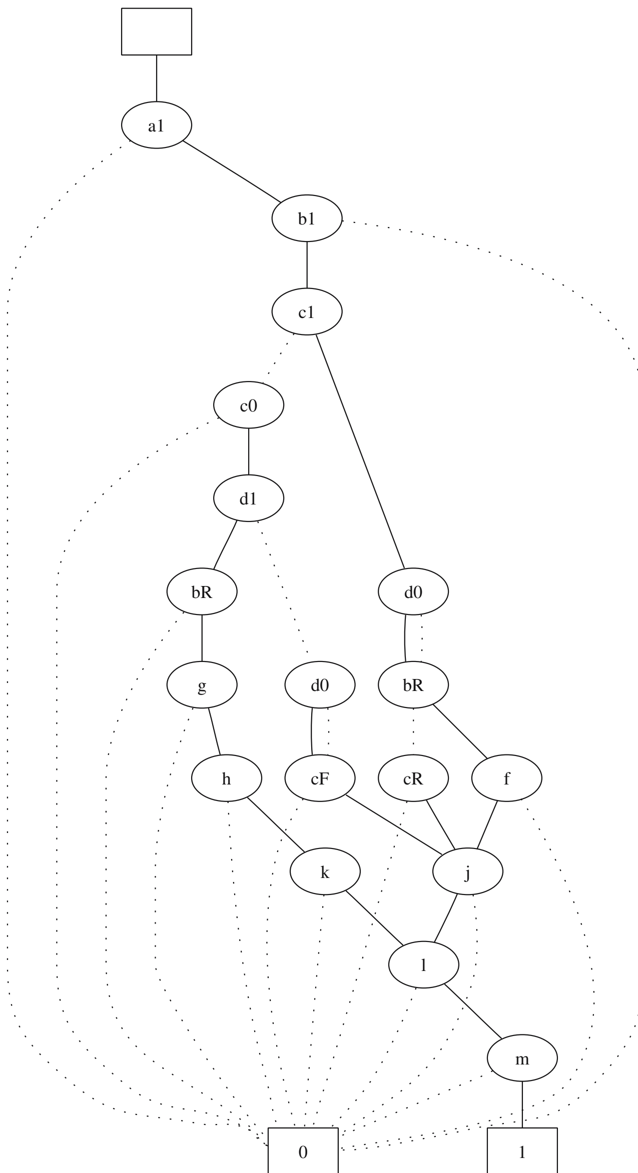


Fig. 16 ISOPs/ ZBDD for critical PDFs in C_3 of Fig. 11

square brackets in each of the nodes of $\mathcal{T}$-graph at this point). Edges can be of three types. A solid (dotted, dashed) edge leaving some node v implies that $v = 1$ ($v = 0$,$v = X$) in the test cube.

The test that detects the maximum number of PDFs, under the variable ordering of the ISOPs/ZBDD graph, is the path in $\mathcal{T}$-graph that terminates to the terminal-1 node with the maximum weight. In general, let a path P be denoted as a collection of consecutive nodes $e_1 \rightarrow e_2 \rightarrow \ldots \rightarrow e_n$. The weight of P, denoted by $W(P)$ is the product of the weights on its edges. Thus,

$$W(e_n) = \prod_{i=1\ldots n} w(e_i)$$

A topological traversal of the $\mathcal{T}$-graph suffices to calculate the maximum weight among all paths up to each node, which leads to the calculation of the maximum weight of a path in the graph. The root node r is always initialized to $W(r) = 1$. When at some node v, with immediate predecessor nodes in $FI(v)$, then $W(v)$ is given by:

$$W(v) = \max_{i \in FI(v)} \{W(i) \times w(i \rightarrow v)\}$$

The maximum weighted path can be easily found by backtracking appropriately from the terminal-1 node to the immediate predecessors, until the root node is reached.

Figures 18 and 19 show the $W(v)$ value per node in the $\mathcal{T}$-graph in square brackets inside each node. The maximum weighted path in the $\mathcal{T}$-graph of Fig. 18 is $a1 - b1 - c1 - d0 - 1$ with weight 4 and, thus, the test cube $a \cdot b \cdot c \cdot \overline{d}$ is the highest quality test (it can sensitize 4 PDFs). The maximum weighted path in the $\mathcal{T}$-graph of Fig. 19 is $a1 - b1 - c1 - 1$, with weight 2.

For this example, where non-robust sensitization was considered, the derived test determines the value of vector v^2 for the PDF test (v^1, v^2). A traversal on the ISOP/ZBDD can determine the PDFs detected by the test. Then, v^1 is determined by the primary input path variables on the detected PDFs, by a linear number (to the number of primary inputs) subset operations. Continuing with the example of Fig. 18, $v^2 = a \cdot b \cdot c \cdot \overline{d}$ $(=a1 \cdot b1 \cdot c1 \cdot d0)$ leads to PDFs $aR - i - m$, $bR - e - i - m$, $bR - f - j - l - m$ and $cR - j - l - m$ and, hence, $v^1 = \overline{a} \cdot \overline{b} \cdot \overline{c} \cdot x$ (d is a don't care). This process is not necessary for robust or functional sensitization, since the test will contain the values for both v^1 and v^2 vectors.

INPUT: $G(\mathcal{T} \cup P)$ ISOPs/ZBDD graph
OUTPUT: $\mathcal{T}$-graph(V, E)
1: Find node set $N^{\mathcal{T}} = \{v, \mid v \in \mathcal{T}\text{-node of } G(\mathcal{T} \cup P)\}$
2: Vertex set $V = \{N^{\mathcal{T}} \cup$ (terminal-1 node)$\}$
% Traverse G in topological order, starting from the root node
3: for each node $v \in N^{\mathcal{T}}$
4: $s(v)$ = list of all immediate successors of v in $N^{\mathcal{T}}$
5: **for each** node $u \in s(v)$
6: Add edge (v, u) in E, if there exists a path from v to u
7: $w(v, u)$ = number of paths from v to u
8: for each node $u \in N^{\mathcal{T}}$
9: **if** $((u \neq$ terminal-1 node$)$ and $(s(u) == \emptyset))$ **then**
$N^{\mathcal{T}} = N^{\mathcal{T}} - u$

Fig. 17 Pseudocode for $\mathcal{T}$-Graph_Create

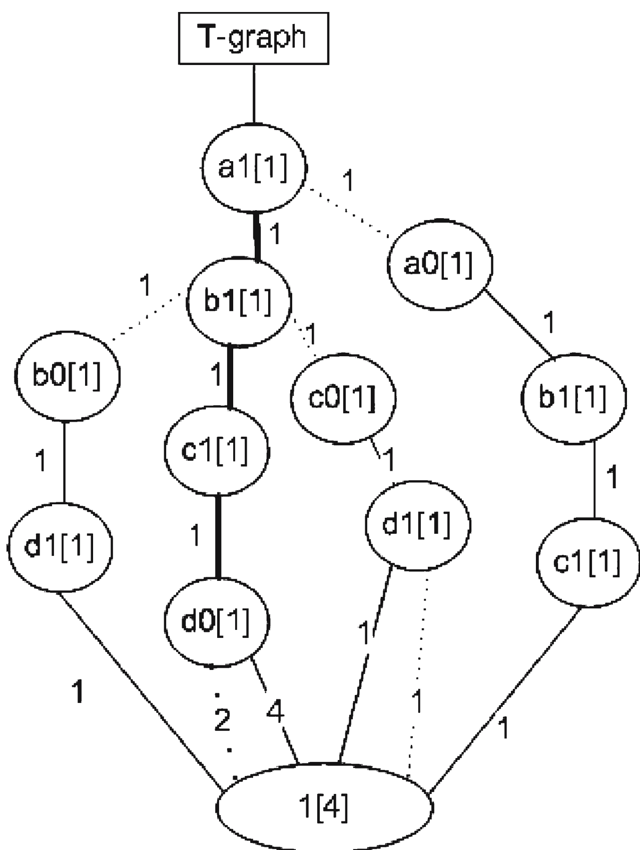


Fig. 18 T-graph for the ISOPs/ZBDD of Fig. 15

5.2 Deriving Additional Tests

We now discuss how additional tests with maximal TE can be derived. The idea here is to be able to remove the PDFs detected by the maximum weighted test extracted by the $\mathcal{T}$-graph from the ISOPs/ZBDD, and repeat the process to find another test with high test efficiency.

Assume that the first test, t, has been generated. Let the original ISOPs/ZBDD be denoted by $G()$. A single traversal from the root to the terminal-1 node of $G()$, restricted by the values of the variables in t, will give the ZBDD that contains all PDFs detected by t, denoted by $G_t()$. Then, $G_t()$ is extended to an ISOPs/ZBDD, denoted by $G'()$, by insisting that all test variables in the circuit appear as don't cares, which is encoded as $\overline{v_0} \cdot \overline{v_1}$ for a variable v in ISOPs-based ZBDD form. This is achieved by a linear number, to the primary inputs, of change (!) operators, as shown below:

$$G'() = \;!(\;!(G_t(), \overline{v_0}),\; \overline{v_1}), \forall v \in \mathcal{T}$$

It suffices to remove $G'()$ from $G()$, using the set difference operation $G() \backslash G'()$, to remove all PDFs detected by t from $G()$. Consequently, the next test generated will be maximal and guarantees to detect only new PDFs. The process just described can be repeated until a fixed number of tests is derived or the desired fault coverage is achieved. The basic steps of the proposed ATPG algorithm are listed in Fig. 20. T_{cov} and T_{th} denote user defined parameters that can be used to terminate the ATPG process when a desired coverage or number of tests has been reached. Alternately the process can run until 100% coverage is achieved. The algorithm accepts as input an ISOPs/ZBDD for the targeted PDFs, denoted by $G()$. $\mathcal{T}$ denotes the set of test variables.

The ATPG process is linear to the size of the ISOPs/ZBDD graph, per generated test. Each iteration is faster than the previous one since the size of the ISOPs/ZBDD graph decreases. The coverage obtained during each operation is trivially calculated without employing fault simulation. If $G() == \emptyset$, then a 100% coverage is reached since no more undetectable PDFs remain in the ISOPs/ZBDD. The achieved coverage is calculated by dividing the number of PDFs detected by the total number of PDFs in $G()$. This requires finding the cardinality of $G()$, which is a standard ZBDD operation that amounts to a linear traversal on $G()$. The coverage calculation is exact, in contrast to all existing

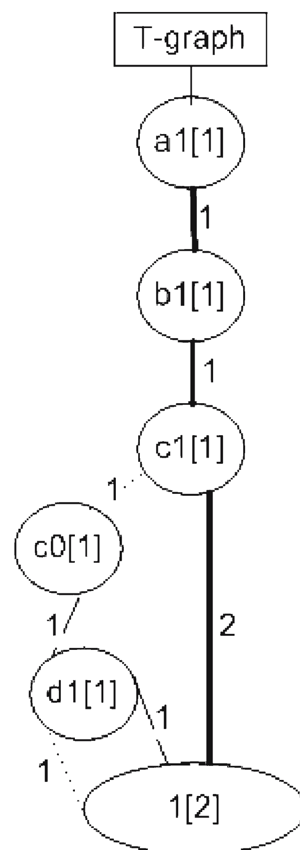


Fig. 19 T-graph for the ISOPs/ZBDD of Fig. 16

Fig. 20 Pseudocode for compact-ATPG algorithm

INPUT: $G()$, T_{cov}, T_{th}
OUTPUT: Compact test set T
1: Repeat
2: $\mathcal{T}$-graph-Create($G()$, $\mathcal{T}$-graph(V, E))
3: **for each** node $u \in V$
4: $FI(u)$ = immediate predecessor nodes of u
% calculate maximum path weight up to u
5: $W(u) = \max_{i \in FI(u)} \{w(i) \times w(i \rightarrow u)\}, i \in FI(u)$
6: t = path with maximum weight in $\mathcal{T}$-graph
7: Derive $G_t()$ from $G()$
8: $G'() = !(\,!(G_t(), \overline{v_0}),\ \overline{v_1}), \forall v \in \mathcal{T}$
9: $G() = G() \backslash G'()$.
10: Until (Coverage == 100%) % or (Coverage == T_{cov}) or (Coverage == T_{th})

practical ATPG methods, since the exact number of sensitizable PDFs is known.

6 Experimental Results

The proposed method was implemented in C language on top of the decision diagram package of [26] (for all BDD and ZBDD related operations), and run on a 1GHz SunBlade 1500 workstation. We report results for all $ISCAS'85$, except for C6288, and the enhanced full-scanned $ISCAS'89$ circuits. C6288 cannot be represented directly using BDDs, however, such circuits could be represented in another appropriate canonical form, such as partitioned-BDDs [16], which can then be converted to ZBDD-based ISOPs. The ZBDD representation of all $\sim 10^{20}$ PDFs of C6288 has been shown to be very efficient in [17]. The initial variable ordering in the BDDs was the one provided along with the tool of [26]. No variable reordering, which can be used to reduce the size of the decision diagrams, was activated in any of the reported experiments. We present results for the non-robust sensitization criterion.

Table 2 reports the resource requirements, for time and space, for the generated ISOPs/ZBDDs. Column 2 lists the total number of PDFs per circuit. Column 3 gives the exact number of sensitizable PDFs per circuit, obtained after generating the ISOPs/ZBDD graph for the sensitization function presented in Section 3. No fault was aborted. The maximum number of PDFs detected by a single test, denoted by MaxTE, is listed in Column 5, obtained by the ATPG algorithm of Section 5. Time (in CPU seconds) and memory (in MBytes) needed to construct the sensitization function and generate the test of MaxTE are given in Columns 6 and 7, respectively. Both requirements are very small for the small circuits. For the larger circuits, memory requirements are increased due to the large number of PDFs in the circuits and, more importantly, the enormous number of tests that exists under the non-robust sensitization criterion. For certain circuits, the single traversal approach to examine all circuit paths, required a large amount of memory to build the sensitization functions. In such cases, the circuit was divided into partitions such that no two partitions contained the same PDFs. The approach was applied on each of these partitions. These circuits are indicated by a * in Column 1 of Table 2. Partitions were derived either by considering all paths terminating to the same primary output or, in certain cases, all paths starting at the same primary input and terminating at the same primary output. The total number of sensitizable PDFs reported is the sum of the sensitizable PDFs in each partition. Since the partitions are non-overlapping, they can be examined independently and, thus, can all be processed in parallel. Time and memory reported for each of these circuits is the maximum required among all partitions. TE can be impacted when partitions are considered, since only a subset of the circuit paths are considered at one time.

Column 4 lists the number of critical PDFs, obtained after generating the ISOPs/ZBDD graph for critical PDFs (Section 4). The proposed method can consider various delay models and critical path selection methods. In our experimentations, we used the bounded delay model. Delay ranges for each gate were obtained from the TSMC 0.18 *micron* technology files using the corner values for the nMoS and pMOS transistors. The delay threshold was set as *90%* of the circuit delay, obtained using static timing analysis. A single topological traversal was used to implicitly identify all *potentially* critical PDFs. The numbers listed in Column 4 are those identified by our approach as critical. The MaxTE, time and space required to derive the critical PDFs and

Table 2 Resource requirements for the ISOPs/ZBDD and MaxTE

Circuit name	Total PDFs	Sens. PDFs	Critical PDFs	All PDFs			Critical PDFs		
				MaxTE	Time (s)	Mem (MBs)	MaxTE	Time (s)	Mem (MBs)
s208.1	284	284	12	10	0.05	0.53	3	0.06	0.22
s386	414	414	78	28	0.07	0.66	5	0.12	0.38
s298	462	364	45	14	0.03	0.29	10	0.11	0.30
s344	710	654	50	10	0.04	0.60	10	0.22	0.30
s349	730	656	50	10	0.07	0.80	10	0.23	0.31
s510	738	738	14	31	0.18	5.20	3	0.18	0.28
s382	800	734	94	19	0.08	0.62	6	0.46	0.39
s526n	820	718	45	14	0.06	0.57	10	0.18	0.25
s420.1	948	948	20	18	2.19	28.31	4	0.39	0.38
s820	984	984	84	34	0.38	6.56	8	0.79	0.48
s832	1,012	996	84	36	0.41	6.68	8	0.78	0.42
s444	1,070	813	72	19	0.11	4.81	12	0.36	4.25
s1488	1,924	1,916	6	50	0.77	16.82	2	0.21	0.09
s1494	1,952	1,927	4	50	0.77	16.40	2	0.11	0.08
s953n	2,312	2,312	54	56	0.87	16.79	12	0.77	0.24
s641	3,488	2,270	142	21	1.29	21.48	21	3.68	1.09
s1196	6,196	3,759	104	81	3.92	30.81	26	3.85	0.79
s1238	7,118	3,684	90	96	7.32	38.81	17	2.98	1.03
C880	17,284	16,652	3,768	94	70.50	43.59	72	63.08	39.05
s3271	38,388	19,292	2,421	87	12.94	95.27	46	18.49	40.87
s713	43,624	4,922	613	185	2.41	24.71	160	6.05	3.02
s1423	89,452	45,198	890	100	144.60	547.57	48	12.44	6.50
C2670*	1,359,920	130,626	56,982	407	24.22	55.91	600	20.17	51.39
C7552*	1,452,988	277,244	21,680	280	254.70	92.65	512	197.41	81.03
C1908*	1,458,114	355,168	67,584	117	1,150.80	196.58	768	1053.80	180.32
s38584.1*	216,1446	334,927	20,444	603	1,034.91	670.54	470	975.15	597.03
C5315*	2,682,610	342,117	64,032	593	159.13	72.31	864	147.35	68.45
s13207*	2,690,738	476,145	64,681	271	367.22	233.70	792	1,105.81	679.31
C1355*	8,346,432	1,110,304	1,008,800	345	539.18	252.31	1,152	489.81	231.13

Asterisk denotes partitioning-based processing on primary output or primary input/primary output basis for All PDFs (Columns 5-7)

Table 3 AvgTE for different test sets

Circuit	Critical PDFs	Max # tests=10		Max # tests=20		Max # tests=30		Max # tests=40		Max # tests=50		100% Cov. AvTE
		Ded	AvgTE	Ded	AvgTE	Ded	AvgTE	Ded	AvgTE	Ded	AvgTE	
s208.1	12	12	1.50	–	–	–	–	–	–	–	–	1.50
s386	78	34	3.40	49	2.45	59	1.97	69	1.73	78	1.59	1.59
s298	45	41	4.10	45	3.21	–	–	–	–	–	–	3.21
s344	50	36	3.60	48	2.40	50	2.27	–	–	–	–	2.40
s349	50	36	3.60	48	2.40	50	2.27	–	–	–	–	2.40
s510	14	14	1.55	–	–	–	–	–	–	–	–	1.55
s382	94	39	3.90	59	2.95	74	2.47	84	2.10	94	1.88	1.88
s526n	45	45	4.10	–	3.21	–	–	–	–	–	–	3.21
s420.1	20	20	2.22	–	–	–	–	–	–	–	–	2.22
s820	84	39	3.90	57	2.85	67	2.23	77	1.93	84	1.79	1.79
s832	84	39	3.90	57	2.85	67	2.23	77	1.93	84	1.79	1.79
s444	72	40	4.00	58	2.90	68	2.27	72	2.12	–	–	2.12
s1488	6	6	1.20	–	–	–	–	–	–	–	–	1.20
s1494	4	4	1.33	–	–	–	–	–	–	–	–	1.33
s953n	54	54	6.00	–	–	–	–	–	–	–	–	6.00
s641	142	99	9.90	121	6.05	131	4.37	141	3.53	142	3.46	3.53
s1196	104	67	6.70	88	4.40	103	3.43	104	3.35	–	–	3.35
s1238	90	77	7.70	90	5.29	–	–	–	–	–	–	5.29
C880	3,768	504	50.40	821	41.05	1,069	35.63	1,285	32.13	1,472	29.44	9.78
s3271	2,421	393	39.30	709	35.45	949	31.63	1,149	28.73	1,319	26.38	13.37
s713	613	524	52.40	575	28.75	591	19.70	601	15.03	611	12.22	23.24
s1423	890	304	30.40	464	23.20	572	19.07	652	16.30	716	14.32	13.21
C2670	56,982	4,440	444.00	7,428	371.40	10,116	337.20	12,132	303.30	13,932	278.64	77.24
C7552	21,680	3,378	337.80	5,842	292.10	7,558	251.93	8,840	221.00	9,838	196.76	56.96
C1908	67,584	8,160	816.00	13,847	692.35	18,403	613.43	22,897	572.42	25,607	512.14	85.37
s38584.1	20,444	3,154	315.40	5,424	271.20	7,052	235.06	7,909	197.72	8,766	175.32	51.29
C5315	64,032	7,368	736.80	13,208	660.40	17,358	578.60	20,844	521.10	24,016	480.32	95.87
s13207	64,681	6,980	698.00	12,866	643.30	15,945	531.50	19,610	490.25	23,627	472.54	79.54
C1355	1,008,800	11,520	1,152.00	23,040	1,152.00	34,560	1,152.00	41,472	1,036.80	47,232	944.64	127.51

– denotes that 100% critical fault coverage is already achieved

the test with the maximum test efficiency are given in Columns 8, 9, and 10, respectively. For critical PDFs, a single sensitization function was derived for every circuit (no partitions were considered). This explains why the time required is sometimes larger for critical PDFs than the one required for all sensitizable PDFs. However, considering the entire circuit at once contributes to a considerable increase in MaxTE in most of the larger circuits, as it can be seen by comparing the results reported in Column 5 with those in Column 8.

The overwhelming majority of the CPU time spent and the memory used, in both cases (all sensitizable PDFs and critical PDFs), is attributed to the ISOPs/ZBDD generation and not to the ATPG task, since the latter is archived by only a linear traversal of the ISOPs/ZBDD. This is very important, since it allows for the generation of complete test sets in linear time to the size of the ISOPs/ZBDD. The only factor that impacts the ATPG time is the number of generated tests (since an ISOPs/ZBDD traversal per test is necessary) but, this is also optimized since the tool's goal is to maximize the test efficiency and hence, the generated test set is inherently compact.

Table 3 reports results that demonstrate the scalability of the approach for critical PDF test generation. The average test efficiency for test sets with different number of tests is given. The average test efficiency, denoted by *AvgTE*, listed in Columns 4, 6, 8, 10, 12 and 13, is the ratio of all detected PDFs (given in Columns 3, 5, 7, 9 and 11) over the total number of generated tests. For example, Column 3 (Ded) lists the number of critical PDFs detected and Column 4 lists their *AvgTE*, when up to the ten tests are generated. Dividing Ded by *AvgTE* gives the exact number of tests generated in each case. We observe that the *AvgTE* is maintained in many occasions, i.e., additional tests with large TE are derived. These are very encouraging results since they demonstrate that very compact test sets can be derived by the proposed approach. Column 13 lists the *AvgTE* when all the critical PDFs listed in Column 2, are detected. The drop in *AvgTE* in this case is attributed to the fact that, after many PDFs are detected and removed from the ISOPs/ZBDD, the remaining critical PDFs are less likely to be tested by common tests. This is inherent to the circuit structure and not to the proposed method that considers all possible tests per PDF.

To our knowledge, no previous method that examines the proposed problem exists and, therefore, no comparison with previous work is possible. All existing methods concentrate either on the compactness constraint of the problem, but do not consider critical PDFs (such as [2, 7, 8, 12, 19, 20, 22]), or on critical PDF selection and test generation but do not consider the compactness issue (such as [18, 24, 28]). The methods of the later case enumerate the critical PDFs (even in the case of the recent method of [18], unsensitizable PDFs are identified non-enumeratively but ATPG examines the critical PDFs explicitly) and, thus, the AvgTE of their generated test sets is bounded to be close to 1.

7 Conclusion

This work presents a test generation methodology for PDF tests with high test efficiency for critical PDFs. It is shown how tests with maximal test efficiency can be derived by linear manipulations of a canonical decision diagram that represents PDFs along with all their associated test patterns compactly. The method can apply to all circuit paths or to sets of circuit paths, such as the set of critical paths. The experimental results clearly demonstrate the practicality of the method and its superiority over existing methods in terms of high test efficiency for critical PDFs. Besides compact test generation, the proposed decision diagram can be important to a variety of other delay test-related and timing analysis problems.

References

1. Bhattacharya D, Agrawal P, Agrawal VD (1995) Test pattern generation for path delay faults using binary decision diagrams. IEEE Trans Comput 44(3):434–447, March
2. Bose S, Agrawal P, Agrawal VD (1993) Generation of compact delay tests by multiple path activation. Proc ITC, pp 714–723
3. Bryant R (1986) Graph-based algorithms for Boolean function manipulation. IEEE Trans Comput C-35(8):677–691, August
4. Cheng KT, Chen HC (1996) Classification and identification of nonrobust untestable path delay faults. IEEE Trans CAD 15(8):845–853, August
5. Drechsler R, Sieling D (2001) Binary decision diagrams in theory and practice. Int J STTT No. 3, pp 112–136
6. Heragu K, Patel JH, Agrawal VD (1997) Fast identification of untestable delay faults using implications. Proc. CAD, pp 642–647
7. Kajihara S, Fukunaga M, Wen M, Maeda T, Hamada S, Sato Y (2005) Path delay test compaction with process variation tolerance. Proc DAC, pp 845–850, June
8. Karayiannis D, Tragoudas S (1999) A fast non-enumerative automatic test pattern generator for path delay faults. IEEE Trans CAD 18(7):1050–1057, July
9. Kirkpatrick DA, Sangiovanni-Vincentelli AL (1996) Digital sensitivity: predicting signal interaction using functional analysis. Proc ICCAD, pp 536–541, November
10. Li ZC, Brayton RK, Min Y (1997) Efficient identification of non-robustly untestable path delay faults using implications. Proc. ITC, pp 992–997
11. McGeer PC, Saldanha A, Stephan PR, Brayton RK, Sangiovanni-Vincetelli AL (1991) Timing analysis and de-

lay fault test generation using path recursive functions. Proc CAD, pp 180–183
12. Michael MK, Tragoudas S (2005) Function-based compact test pattern generation for path delay faults. IEEE Trans Very Large Scale Integr 13(8):996–1001, April
13. Minato S-I (1993) Fast generation of prime-irredundant covers from binary decision diagrams. IEICE Trans Fundam E76-A(6):976–973, June
14. Minato S-I (1993) Zero-suppressed BDDs for set manipulation in combinatorial problems. Proc DAC, pp 272–277
15. Minato S-I (1996) Fast factorization method for implicit cube set representation. IEEE Trans CAD 15(3):377–384, April
16. Narayan A, Jain J, Fujita M, Vincentelli AS (1996) Partitioned ROBDDs - a compact, canonical and efficiently manipulable representation for boolean expressions. Proc ICCAD, pp 547–554
17. Padmanaban S, Michael MK, Tragoudas S (2003) Exact path delay fault coverage with fundamental ZBDD operations. IEEE Trans CAD 22(3):305–316, March
18. Padmanaban S, Tragoudas S (2005) Efficient identification of (Critical) testable path delay faults using decisions diagrams. IEEE Trans CAD 24(1):77–87, January
19. Pomeranz I, Reddy SM (2003) Test enrichment for path delay faults using multiple sets of target faults. IEEE Trans CAD 22(1):82–89, January
20. Pomeranz I, Reddy SM, Uppaluri P (1995) NEST: a nonenumerative test generation method for path delay faults in combinational circuits. IEEE Trans CAD 14(12):1505–1515, December
21. Sasao T, Butler JT (2001) Worst and best irredundant sum-of-products expressions. IEEE Trans Comput 50(9):935–948, September
22. Saxena J, Pradhan DK (1993) A method to derive compact test sets for path delay faults in combinational circuits. Proc ITC, pp 724–733
23. Shao Y, Reddy SM, Kajihara S, Pomeranz I (2001) An efficient method to identify untestable path delay faults. Proc ITC, p 233
24. Sharma M, Patel JH (2001) Testing of critical paths for delay faults. Proc ITC, pp 634–641
25. Sivaraman M, Strojwas AJ (2000) Primitive path delay faults: identification and their use in timing analysis. IEEE Trans CAD 19(11):1347–1362, November
26. Somenzi F (1999) CUDD: CU decision diagram package. Dept. of ECE, The University of Colorado, release 2.3.0
27. Tafertshofer P, Ganz A, Antreich KJ (2000) IGRAINE – an implication GRaph bAsed engINE for fast implication, justification, and propagation. IEEE Trans CAD 19(8):907–927, August
28. Wang LC, Liou JJ, Cheng KT (2004) Critical path selection for delay fault testing based upon a statistical timing model. IEEE Trans CAD 23(11):1550–1565, November
29. Xiang D, Li K, Fujiwara H, Sun J (2006) Generating compact robust and non-robust tests for complete coverage of path delay faults based on stuck-at tests. Proc. ICCD, pp 446–451

Kyriakos Christou received a B.S.c degree in Computer Science from University of Cyprus 2001 and a MSE degree in Computer and Information Science from the University of Pennsylvania, USA in 2003. Since 2004, he has been a Ph.D. Candidate at the Electrical and Computer Engineering Dept. of the University of Cyprus. His research interests include timing verification and delay testing, testable design, test generation, verification and formal methods, model checking.

Maria K. Michael received a M.Sc. degree in Computer Science and a Ph.D. degree in Electrical and Computer Engineering in 1998 and 2002, respectively, from Southern Illinois University, USA. During 2002–2003, she was an Assistant Professor of Computer Science and Engineering at the University of Notre Dame, USA. She is currently and Assistant Professor at the Electrical and Computer Engineering Dept. of the University of Cyprus. Her research include timing verification and delay testing, testable design, test generation and diagnosis, and logic synthesis and verification.

Spyros Tragoudas (S'87–M'87) received the diploma in computer engineering and informatics from the University of Patras, Patras, Greece, in 1986 and the M.S. and Ph.D. degrees in computer science from the University of Texas, Dallas, in 1988 and 1991, respectively. He is currently a Professor in the Computer Science Department, Southern Illinois University, Carbondale, and the Electrical and Computer Engineering Department, University of Arizona, Tucson. He has published over 130 papers in journals and peer-reviewed conference proceedings in these areas. He has been funded from the National Science Foundation and industry for research in very large scale integration design (VLSI) testing. His research interests are in VLSI and test automation, and computer networks. Prof. Tragoudas has received the ICCD'94, ICCD'97, and ISQED'01 Outstanding Paper Awards for research in VLSI testing.