Integer Programming Models for Optimization Problems in Test Generation

João P. Marques Silva

Cadence European Laboratories
IST/INESC
1000 Lisboa, Portugal
Tel: 351-1-3100387
e-mail: jpms@inesc.pt

Abstract— Test Pattern Generation for combinational circuits entails the identification of primary input assignments for detecting each fault in a set of target faults. An extension to this procedure consists of reducing the number of required test patterns by using heuristic test compaction techniques. In this paper we show that finding the optimally compacted test set can be cast as an integer linear programming (ILP) optimization problem, thus providing a formal framework for characterizing this optimization problem as well as the heuristics commonly used in its solution. One significant property of the proposed ILP model is that its size is polynomial in the size of the original circuit description. Moreover, we describe techniques for reducing the size of the proposed ILP formulation. These techniques include, for example, identification of fault independence relations, removal of redundant faults by preprocessing, and using empirical upper bounds.

I. Introduction

The test pattern generation problem (ATPG) for different fault models, as so many other key tasks in electronic design automation, is computationally hard, being NP-complete [7, 9]. Furthermore, the problem of computing the minimum number of tests for detecting all faults in an irredundant circuit is known to be NP-hard [10]. The generalization of this problem to arbitrary combinational circuits is also necessarily NP-hard and will be henceforth referred to as the ***minimum test set*** problem. Modeling and algorithmic solutions to this optimization problem can be particularly significant for large circuits and for circuits aimed for large scale production, since a smaller number of tests necessarily implies smaller testing time as well as smaller memory requirements for storing the test patterns [1, 15, 16, 20]. An immediate application is for example in the synthesis of Built-In Self-Test (BIST) logic, since a smaller number of test patterns necessarily simplifies the logic and the test application time.

Even though in this paper we solely address the minimum test set problem, other optimization problems in test pattern generation can be readily formulated using the same modeling approach, each of which with potential practical application. (See for example [5] for a description of additional optimization problems in testing.)

The main purpose of this paper is to develop integer linear program (ILP) models for solving the minimum test set problem. The proposed formulation is polynomial in the original size of the circuit. To our best knowledge, no other explicit formulations exist for solving this optimization problem. In general, most work on tackling the minimization of test sets has consisted of heuristic techniques for test compaction [15, 16]. Nevertheless, a BDD-based approach for the computation of minimum test sets has been described in [13]. However, this approach is based on implicitly representing *all* test patterns for each fault, thus requiring worst-case exponential space, and so it is impractical except for very small circuits. Another relevant research topic has been on estimating the minimum number of tests for circuits with special properties [14], but no results have been established for arbitrary combinational circuits. We should also note that the proposed ILP formulation can easily be solved for small-size circuits. Hence, besides its theoretical interest, the proposed ILP formulations can be used in validating new and existing test compaction heuristics [10, 15, 16, 20].

The paper is organized as follows. We start, in Section II, by introducing several definitions that will be used throughout the paper. In particular we review Conjunctive Normal Form (CNF) formulas for representing circuits and fault detection problems, ILP formulations, and straightforward algebraic generalizations of CNF formulas. Section III develops an ILP formulation for the minimum test set problem, shows that this formulation is correct and describes techniques for reducing the size of the resulting ILPs. Finally we conclude with directions for

future research work, which include addressing the practical implementation of dedicated algorithms for solving the proposed ILP formulation as well as additional techniques for their simplification.

II. Definitions

We start by introducing unified representations for circuits, fault detection problems, and associated optimization problems. These representations are used throughout the paper and are key for developing the proposed ILP formulations. A combinational circuit C is represented as a directed acyclic graph $C = (V_C, E_C)$, where the elements of V_C, i.e. the circuit nodes, are either primary inputs or gate outputs, with $|V_C| = N$. The set of edges $E_C \subseteq V_C \times V_C$ identifies gate input-output connections. We shall assume gates with bounded fanin, and so $|E_C| = O(|N|)$. For every circuit node x in V_C, the following definitions apply:

- $O(x)$ denotes the ***fanout*** nodes of node x, i.e. nodes y in V_C such that $(x, y) \in E_C$.
- $O^*(x)$ denotes the ***transitive fanout*** of node x, i.e. the set of all nodes y such that there is a path connecting x to y.
- $I(x)$ denotes the ***fanin*** nodes of node x, i.e. nodes y in V_C such that $(y, x) \in E_C$.
- $I^*(x)$ denotes the ***transitive fanin*** of node x, i.e. the set of all nodes y such that there is a path connecting y to x.
- $K_O(x)$ denotes ***immediate fanout cone of influence*** of x, being defined as follows:

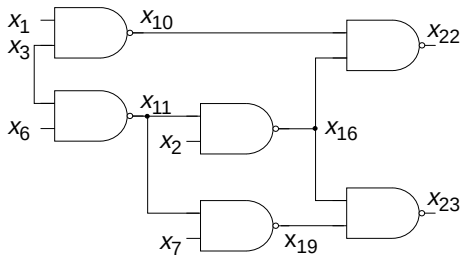
$$K_O(x) = \{y | y \in O^*(x) \vee y \in I(w) \wedge w \in O^*(x)\} \ . \quad (1)$$

- $K_I(x)$ denotes ***immediate fanin cone of influence*** of x, being defined as follows:

$$K_I(x) = \left[\bigcup_{y \in O^*(x)} I^*(y) \right] - (O^*(x) \cup \{x\}) \ . \quad (2)$$

The set of primary inputs is referred to as PI, and the set of primary outputs as PO. Simple gates are assumed: AND, NAND, OR, NOR, NOT and BUFF. Finally, the number of stuck-at faults in the circuit is M, with $M = O(N)$, since $|E_C| = O(|N|)$, being numbered $1, \ldots, M$. The example in Fig. 1 illustrates the previous definitions.

For Automatic Test Pattern Generation (ATPG), the following definitions apply. The single stuck-at line (SSF) fault model is assumed [1]. We say that a stuck-at fault is



(a) Circuit

Node x	x_{11}
$O(x)$	$\{x_{16}, x_{19}\}$
$O^*(x)$	$\{x_{16}, x_{19}, x_{22}, x_{23}\}$
$I(x)$	$\{x_3, x_6\}$
$I^*(x)$	$\{x_3, x_6\}$
$K_O(x)$	$\{x_2, x_7, x_{10}, x_{16}, x_{19}, x_{22}, x_{23}\}$
$K_I(x)$	$\{x_{10}, x_2, x_7, x_1, x_3, x_6\}$

(b) Topological data for x_{11}

Stuck-at faults	34
Collapsed faults	17

(c) Stuck-at faults

Fig. 1. Example circuit — C17 [3]

detectable if and only if there exists an assignment of logic values to the circuit primary inputs such that the effect of the fault can be observed at one of the circuit primary outputs.

A. CNF Formulas and Test Pattern Generation

The application of Conjunctive Normal Form (CNF) representations of circuits and fault detection problems in ATPG has been extensively studied [4, 11, 18, 19]. In this section we provide very simple and non-optimized CNF representations of circuits and fault detection problems, which will be assumed in the remainder of the paper. These representations form the basis for the ILP models introduced in the remainder of the paper.

The CNF formula of a circuit is the conjunction of the CNF formulas for each gate output, where the CNF formula of each gate denotes the valid input-output assignments to the gate. (Derivation of the CNF formulas for simple gates can be found for example in [11, 18].) If we view a CNF formula as a set of clauses, the CNF formula

φ for the circuit is defined by the set union[1] of the CNF formulas for each gate:

$$\varphi = \bigcup_{x \in V_C} \varphi_x \tag{3}$$

In the context of test pattern generation, and for capturing the fault detection problem, each node x is characterized by three propositional variables:

- x^G denotes the logic value assumed by the node in the ***good*** circuit.
- x^F denotes the logic value assumed by the node in the ***faulty*** circuit.
- x^S denotes whether x^G and x^F assume different logic value [11]. We shall refer to this variable as the ***sensitization status*** of node x. (Other semantic definitions of the sensitization status have been proposed [4, 19]. Nevertheless, the above definition is used since it simplifies the ILP formulations derived in subsequent sections. We should note, however, that the other semantic definitions could also be used.)

Given the definition of variable x^S, the condition $\left[\left(x^G \neq x^F\right) \leftrightarrow x^S\right]$ must hold, which can be simplified to φ_x^S:

$$\begin{aligned}\varphi_x^S = &\left(x^G + \neg x^F + x^S\right) \cdot \left(\neg x^G + x^F + x^S\right) \cdot \\ &\left(\neg x^S + x^G + x^F\right) \cdot \left(\neg x^S + \neg x^G + \neg x^F\right)\end{aligned} \tag{4}$$

that basically states that the logic values of x^G and x^F differ if and only if x^S assumes logic value 1.

Let φ_x denote the CNF formula associated with gate output x. The notation φ_x^G denotes the CNF formula for x in the good circuit, i.e. using y^G variables, whereas φ_x^F denotes the CNF formula for x in the faulty circuit, i.e. using y^F variables. For a ***stem*** fault z-a-v[2], the CNF representation of the associated fault detection problem contains the following components:

- CNF formula for the circuit, denoting the good circuit, φ^G.
- CNF formula for the circuit, denoting the faulty circuit, φ^F. This formula only needs to contain the CNF formulas for the nodes that are relevant for detecting the given fault, i.e. nodes in the transitive fanout of node z.
- CNF formulas for defining the sensitization status of every node in the transitive fanout of the fault site, i.e. node z. Hence, for each of these nodes add φ_x^S, which states that $x^S = 1$ if and only if $x^G \neq x^F$.
- Clauses that prevent each node x from being sensitized, by having $x^S = 0$, whenever x is not in the transitive fanout of z but at least one fanout node of x is in the transitive fanout of z, i.e. x is in $K_O(z) - O^*(z)$.
- Clauses requiring $x^G = x^F$ on each node x such that x is not in the transitive fanout of z but at least one fanout node of x is in the transitive fanout of z, i.e. x is in $K_O(z) - O^*(z)$. (Observe that this condition and the previous one permit restricting the number of x^F and x^S variables that must actually be used.)
- Clauses capturing conditions for ***activating*** the fault, i.e. by requiring $z^G \neq z^F$ and by forcing a suitable logic value on z^G, φ^A.
- Clause requiring that at least one sensitization variable of a primary output in the transitive fanout of the fault site assumes value 1, φ^R.

(A more detailed derivation of formula φ^D for detecting a fault z s-a-v can be found in [18].) φ^D will henceforth be referred to as the ***fault detection formula***. Similarly, we define the ***fault-specific formula***, φ^{FS}, as follows,

$$\varphi^{FS} = \varphi^D - \left(\varphi^G \cup \varphi^R\right) \tag{5}$$

Fault-specific formulas contain only the clauses associated with propagating the error signal to the primary outputs and can be defined independently of the circuit formula. The fault-specific CNF formula for fault x_{11} s-a-1 is given in Fig. 2. Finally, we observe that a similar model can be constructed for ***fanout-branch*** faults [11, 19].

Given the proposed CNF formulation for the fault detection problem, we have the following formal results (proofs of which can be found in [17]):

Proposition 1. Given a stuck-at stem fault z s-a-v, the fault is detectable if and only if the associated fault detection formula $\varphi^D = \varphi^{FS} \cup \varphi^G \cup \varphi^R$ is satisfiable.

We should also note that the above result can also be established for the CNF model for a fanout-branch fault. Furthermore, this result gives a formal guarantee of correctness for the fault detection CNF models described in [4, 11, 19].

1. Set union in this context is to be understood as a product of clauses.
2. See [1] for the definitions used throughout the paper for ATPG.

Sub-formula/Condition	Clause Set
Faulty Circuit	$\varphi^F = \{\varphi^F_{x_{16}} \cup \varphi^F_{x_{19}} \cup \varphi^F_{x_{22}} \vee \varphi^F_{x_{23}}\}$
Node Sensitization	$\varphi^S = \{\varphi^S_{x_{16}} \cup \varphi^S_{x_{19}} \cup \varphi^S_{x_{22}} \cup \varphi^S_{x_{23}}\}$
Propagation Blocking	$\varphi^B = \left(\neg x^S_2\right) \cdot \left(\neg x^S_7\right) \cdot \left(\neg x^S_{10}\right)$
Side Input Equivalence	$\varphi^E = \left(\neg x^G_2 + x^F_2\right) \cdot \left(x^G_2 + \neg x^F_2\right) \cdot \left(\neg x^G_7 + x^F_7\right) \cdot \left(x^G_7 + \neg x^F_7\right) \cdot \left(\neg x^G_{10} + x^F_{10}\right) \cdot \left(x^G_{10} + \neg x^F_{10}\right)$
Fault Activation Conditions	$\varphi^A = \left(x^S_{11}\right) \cdot \left(\neg x_{11}{}^G\right) \cdot \left(x_{11}{}^F\right)$
Fault Detection Requirement	$\varphi^R = \left(x^S_{22} + x^S_{23}\right)$

Fig. 2. Example of fault-specific and fault detection CNF

Proposition 2. For any fault in a combinational circuit composed of simple gates, the size of the associated fault detection formula φ^D is $O(N)$, clauses or literals, where N is the number of circuit nodes.

The proposed CNF formulations can be simplified and improved (see for example [4, 11, 18, 19] for further details). Nevertheless, for the purposes of this paper the proposed formulation suffices and shall be assumed.

B. Integer Programs and Generalized Clauses

An integer linear program consists of a set of variables, which can only assume integer values, a set of linear constraints on those variables, and a cost function that is to be maximized or minimized. For the purposes of this paper we further restrict the value of each variable to either be 0 or 1. Hence, a 0-1 integer linear program (ILP) is defined as follows:

$$\begin{aligned} &\text{minimize} && \boldsymbol{c}^T \cdot \boldsymbol{x} \\ &\text{subject to} && \boldsymbol{A} \cdot \boldsymbol{x} \geq \boldsymbol{b} \qquad \boldsymbol{x} \in \{0, 1\}^N \end{aligned} \tag{6}$$

where $\boldsymbol{x}$ is the vector of Boolean variables, $\boldsymbol{c}^T$ is the vector of coefficients for the cost function, and the matrices $\boldsymbol{A}$ and $\boldsymbol{b}$ define the set of linear constraints of the ILP.

The clausal formulations of the previous section were defined assuming a Boolean algebra domain. Nevertheless, we can generalize the algebraic domain of each clause. This generalization is commonly used in the field of Operations Research for describing different optimization problems [8]. Let us consider, for example, the clause $(\neg x_1 + x_2 + \neg x_3)$. Since we want the clause to be satisfied, then it is also true that the linear inequality $\neg x_1 + x_2 + \neg x_3 \geq 1$ must hold [2]. Furthermore, given that Boolean variables can only be assigned value 0 and 1, we have $\neg x_i \equiv 1 - x_i$. The application of this relation to the above example clause yields $-x_1 + x_2 - x_3 \geq -1$. Hence, clauses in CNF formulas can also be viewed as inequalities and be handled as such. Given this view of clauses, the set of constraints $\boldsymbol{A} \cdot \boldsymbol{x} \geq \boldsymbol{b}$ in an ILP formulation may include CNF formulas as well as other additional constraints with arbitrary multiplicative constants.

III. Minimum Test Sets

In this section we address the computation of the minimum number of test patterns which detect all detectable faults in a combinational circuit. First, we provide an ILP formulation for computing the minimum number of test patterns which detect all faults in an irredundant circuit. Afterwards, we extend this formulation to arbitrary combinational circuits, which may include redundant faults.

A. Irredundant Combinational Circuits

The first problem we address is to identify the minimum number of test patterns that detect all faults in an irredundant combinational circuit. Since there are M faults to be detected, we need at least to specify a fault detection formula for each fault. Moreover, M tests suffice for detecting all M faults. As a result, we can create M copies of each circuit formula, and for each such formula we can create M associated fault-specific formulas, one for each fault. A set of tests that detects all faults will do so in such a composed formula (see Fig. 3). In general, we refer to each copy of the circuit formula as φ^G_i, and refer to the *fault-specific* formula associated with each fault j and copy φ^G_i, of the circuit formula, as $\varphi^{FS}_{i,j}$. In each copy φ^G_i each variable $x \in V_C$ will be represented as x^G_i. Accordingly, variables associated with fault detection problems and with node x will be represented as $x^F_{i,j}$ and $x^S_{i,j}$ in each formula $\varphi^{FS}_{i,j}$. Given the above, the problem of minimizing the number of tests is now reduced to finding a set of test patterns that detect all faults and that minimize the number of copies φ^G_i used.

Let us associate a Boolean variable $d_{i,j}$ with each fault-specific formula $\varphi^{FS}_{i,j}$ which assumes value true whenever fault j is ***detected*** with copy i of the circuit for-

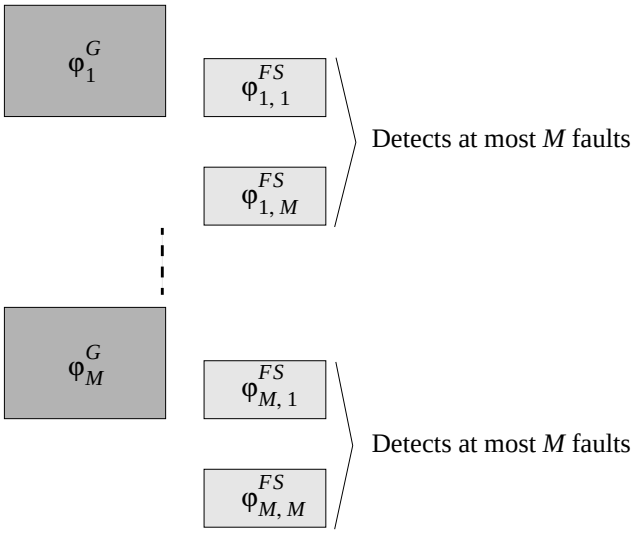


Fig. 3. Global formula organization for detecting all faults

mula and associated fault-specific formula. This leads to the following conditions for all *i, j*:

$$\bigcup_{x \in PO}\left(x_{i,j}^{S} \rightarrow d_{i,j}\right) \Leftrightarrow \bigcup_{x \in PO}\left(\neg x_{i,j}^{S} + d_{i,j}\right) \Leftrightarrow \bigcup_{x \in PO}\left(d_{i,j} - x_{i,j}^{S} \geq 0\right) \tag{7}$$

and,

$$\left(d_{i,j} \rightarrow \sum_{x \in PO} x_{i,j}^{S}\right) \Leftrightarrow \left(\neg d_{i,j} + \sum_{x \in PO} x_{i,j}^{S}\right) \Leftrightarrow \sum_{x \in PO} x_{i,j}^{S} - d_{i,j} \geq 0 \tag{8}$$

Hence, fault *j* is detected with $\varphi_{i,j}^{FS}$, if at least one sensitization variable $x_{i,j}^{S}$ evaluates to true. In addition, either $d_{i,j}$ assumes value 0, or the error signal must reach at least one primary output; hence condition (8). Clearly, since each fault *j* is detectable and must be detected by a test set, then the following conditions must hold,

$$\left(\sum_{i=1}^{M} d_{i,j}\right) \Leftrightarrow \left(\sum_{i=1}^{M} d_{i,j} \geq 1\right) \tag{9}$$

for $j \in \{1, \ldots, M\}$. Let us now introduce a variable $s_{i,j}$, for each variable $d_{i,j}$, such that,

$$[d_{i,j} \rightarrow s_{i,j}] \Leftrightarrow (\neg d_{i,j} + s_{i,j}) \Leftrightarrow s_{i,j} - d_{i,j} \geq 0 \tag{10}$$

and,

$$[s_{i-1,j} \rightarrow s_{i,j}] \Leftrightarrow (\neg s_{i-1,j} + s_{i,j}) \Leftrightarrow s_{i,j} - s_{i-1,j} \geq 0 \tag{11}$$

Variable $s_{i,j}$ evaluates to true whenever $d_{i,j} = 1$. In such a situation, for all $k > i$, $s_{k,j} = 1$. Thus, each variable $s_{i,j}$ permits ***selecting*** the least *i* for which fault *j* is detected. Notice that the least *i* for which fault *j* is detected is such that $s_{i-1,j} = 0$ and $s_{i,j} = 1$.

In addition to the previous variables, we need to associate a Boolean variable u_i that is assigned value true whenever copy *i* of the circuit formula is ***used*** to detect at least one fault which had not been detected with a smaller *i*. Hence, we must have,

$$\begin{aligned} &[s_{1,j} \rightarrow u_1] \Leftrightarrow [-s_{1,j} + u_1 \geq 0] \\ &[(\neg s_{i-1,j} \wedge s_{i,j}) \rightarrow u_i] \Leftrightarrow [s_{i-1,j} - s_{i,j} + u_i \geq 0] \end{aligned} \tag{12}$$

for $i \in \{2, \ldots, M\}$, $j \in \{1, \ldots, M\}$. This condition basically states that the first copy *i*, for which fault *j* is detected, is declared to be used for detecting fault *j*. Hence, from the definition of u_i we have the following result,

Proposition 3. Given the definition of u_i in (12), each fault *j* can set exactly one variable u_i to true. Moreover, more than one fault *j* can set the same variable u_i.

Finally, we must define the cost function that we want to minimize. Clearly, this cost function should minimize the number of copies of the circuit formula that are used for detecting all faults, and so we get,

$$min\left[\sum_{i=1}^{M} u_i\right] \tag{13}$$

The *M* replicas of the clausal representation of the circuit, each with its *M* copies of the fault detection problems, as well as (7) through (13) capture the problem of computing the minimum number of tests for a given irredundant circuit. For our running example, the data for the associated ILP formulation is summarized in Fig. 4. Finally, we can establish the following formal results (see [17] for proofs of these results).

Proposition 4. The size of the ILP for the minimum test set problem (described above) is $O\left(M^2 \cdot N\right) = O\left(N^3\right)$.

Proposition 5. The minimum value of (13) denotes the minimum number of tests that detect all faults in the irredundant combinational circuit *C*. Furthermore, each of the

	# Clauses	# Formulas
Circuit formula		34
Fault-specific formula		$34^2 = 1156$
Detection requirement	34	
Fault detection selection	$34^2 = 1156$	
Propagation of selection	$33 \times 34 = 1122$	
Usage of replica *i*	$34^2 = 1156$	
Cost function	1	

Fig. 4. Upper bounds on the ILP formulation

assignments to the primary inputs of each circuit copy φ_i^G, for which $u_i = 1$, denote a test pattern.

B. Arbitrary Combinational Circuits

The next problem we address is the identification of the minimum number of test patterns which detect all detectable faults in a given arbitrary combinational circuit, as well as the identification of all the redundant faults in that circuit. This problem is quite similar to the problem of the previous section, with the added difficulty that some faults are now redundant.

One simple solution to this problem is to introduce additional variables and modify some of the equations of the ILP of the previous section. First, let us define a variable r_j, with $j \in \{1, \ldots, M\}$, that is true when fault *j* is declared ***redundant***. Given that some faults are indeed redundant, (12) must be modified to capture this fact:

$$\left(\sum_{i=1}^{M} d_{i,j} + r_j \right) \Leftrightarrow \left(\sum_{i=1}^{M} d_{j,i} + r_j \geq 1 \right) \qquad (14)$$

for $j \in \{1, \ldots, M\}$. This condition requires that either the fault is detected by at least one copy of the circuit formula or otherwise it must be declared redundant. The next step is to modify the cost function so as to penalize the existence of redundant faults. This can be done by giving a sufficiently large weight to variables declared redundant, such that the cost of declaring a fault redundant is larger than the sum of all u_i variables. Hence, the minimum value is achieved only when the actual redundant variables are declared redundant, since these faults cannot be detected. Consequently, we obtain the following cost function:

TABLE I. ILP FOR THE MINIMUM TEST SET PROBLEM

	Clauses	Ranges of indices
Circuit representation	φ_i^G	$1 \leq j \leq M$
Fault detection	$\varphi_{i,j}^{FS}$	$1 \leq i \leq M, 1 \leq j \leq M$
Detection requirement	$\sum_{i=1}^{M} d_{i,j} + r_j \geq 1$	$1 \leq j \leq M$
Fault detection selection	$s_{i,j} - d_{i,j} \geq 0$	$1 \leq i \leq M, 1 \leq j \leq M$
Propagation of selection	$s_{i,j} - s_{i-1,j} \geq 0$	$2 \leq i \leq M, 1 \leq j \leq M$
Usage of replica *i*	$-s_{1,j} + u_1 \geq 0$	$1 \leq j \leq M$
	$s_{i-1,j} - s_{i,j} + u_i \geq 0$	$2 \leq i \leq M, 1 \leq j \leq M$
Cost function	(15)	

$$min \left[\sum_{i=1}^{M} u_i + M \cdot \sum_{j=1}^{M} r_j \right] \qquad (15)$$

This cost function splits the range of possible values into disjoint sets according to the number of variables r_j set to true. Valid ranges are, $1..M$, $(M+1)..(2 \cdot M - 1)$, $(2 \cdot M + 1)..(3 \cdot M - 2)$, etc. The size of these ranges is, respectively, $M, M-1, M-2$, etc. Intuitively, the minimum value will be achieved when all detectable faults are indeed detected and in the least number of copies of the circuit formula. The ILP formulation for this problem is summarized in Table I.

Proposition 6. The minimum value of (15), assuming (14) instead of (9), denotes the minimum number of tests that detect all detectable faults and identify all redundant faults in a combinational circuit *C*. In addition, the assignments to the primary inputs of each circuit copy φ_i^G, for which $u_i = 1$, denote a test pattern. Finally, each variable r_j set to true indicates that fault *j* is redundant.

C. Practical Considerations

Different techniques for reducing the size of the ILPs associated with minimum test set computation are described in [17]. These techniques involve removing redundant faults from the fault set, applying fault dominance and independence relationships, and using empirical upper bounds. For the example circuit of Fig. 1, C17, there are a total of 34 stuck-at faults. This leads to the ILP model described in Fig. 4. In contrast, by running the ATPG algorithm ATALANTA [12], four test patterns are

identified. Thus, instead of an ILP model that contains $34^2 = 1156$ fault-specific formulas, using the empirical upper bound of 4 tests, the number of fault-specific formulas is guaranteed to be at most $4 \times 34 = 136$. Moreover, by taking into account dominance relations, the set of faults can be collapsed into 17 faults [3] (see Fig. 1-(c)). Thus, only $17 \times 4 = 68$ fault-specific formulas need to be considered. Clearly, similar savings are expected in most practical examples.

IV. Conclusions and Ongoing Work

In this paper we describe CNF formulations for the identification of test patterns and show how these formulations can be used for constructing integer linear programs (ILP) for solving optimization problems in testing, in particular the minimum test set problem. The size of the proposed ILPs is in the worst-case polynomial in the number of circuit nodes, in contrast with previous solutions, which require worst-case exponential-size representations [13]. Besides their theoretical interest, the proposed ILP formulations can be used in small-size circuits for validating new heuristics for test set compaction.

Current research work targets the simplification of the proposed ILP models, by simplifying the models themselves, and by using problem-specific information. In addition, we have developed ILP models for other optimization problems in testing [5], with promising experimental results.

References

[1] M. Abramovici, M. A. Breuer and A. D. Friedman, *Digital Systems Testing and Testable Design*, Computer Science Press, 1990.

[2] P. Barth, "A Davis-Putnam Based Enumeration Algorithm for Linear Pseudo-Boolean Optimization," Technical Report MPI-I-95-2-003, Max-Planck-Institut für Informatik, January 1995.

[3] F. Brglez and H. Fujiwara, "A Neutral List of 10 Combinational Benchmark Circuits and a Target Translator in FORTRAN," in *Proceedings of the International Symposium on Circuits and Systems*, 1985.

[4] S. T. Chakradhar, V. D. Agrawal and S. G. Rothweiler, "A Transitive Closure Algorithm for Test Generation," *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 7, pp. 1015-1028, July 1993.

[5] P. F. Flores, J. P. M. Silva and H. C. Neto, "An Exact Solution to the Minimum-Size Test Pattern Problem," submitted for publication, October 1997.

[6] H. Fujiwara and T. Shimono, "On the Acceleration of Test Generation Algorithms," *IEEE Transactions on Computers*, vol. 32, no. 12, pp. 1137-1144, December 1983.

[7] H. Fujiwara and S. Toida, "The Complexity of Fault Detection Problems for Combinational Logic Circuits," in *IEEE Transactions on Computers*, vol. 31, no. 6, pp. 555-560, June 1982.

[8] J.N. Hooker, "Logic-Based Methods for Optimization," ORSA CSTS Newsletter, vol. 15, no. 2, pp. 4-11, 1994.

[9] O.H. Ibarra and S.K. Sahni, "Polynomially Complete Fault Detection Problems," in *IEEE Transactions on Computers*, vol. 24, no. 3, pp. 242-249, March 1975.

[10] B. Krishnamurthy and S.B. Akers, "On the Complexity of Estimating the Size of a Test Set," in *IEEE Transactions on Computers*, vol. 33, no. 8, pp. 750-753, August 1984.

[11] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability," *IEEE Transactions on Computer-Aided Design*, vol. 11, no. 1, pp. 4-15, January 1992.

[12] H. K. Lee and D. S. Ha, "On the Generation of Test Patterns for Combinational Circuits," Technical Report No. 12_93, Department t of Electrical Engineering, Virginia Polytechnic Institute and State University, 1993.

[13] Y. Matsunaga, "MINT—An Exact Algorithm for Finding Minimum Test Set," in *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E76-A, no. 10, pp. 1652-1658, October 1993.

[14] I. Pomeranz and Z. Kohavi, "The Minimum Test Set Problem for Circuits with Nonreconvergent Fanout," in *Journal of Electronic Testing: Theory and Applications*, vol. 2, pp. 339-349, 1991.

[15] I. Pomeranz, L.N. Reddy, S.M. Reddy, "COMPACTEST: A Method to Generate Compact Test Sets for Combinational Circuits," in *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 7, pp. 1040-1049, July 1993.

[16] M. H. Schulz and E. Auth, "Improved Deterministic Test Pattern Generation with Applications to Redundancy Identification," *IEEE Transactions on Computer-Aided Design*, vol. 8, no. 7, pp. 811-816, July 1989.

[17] J. P. M. Silva, "On Computing Minimum Size Test Sets in Combinational Circuits," Technical Report RT/01/97, INESC, Portugal, January 1997.

[18] J. P. M. Silva and K. A. Sakallah, "Robust Search Algorithms for Test Pattern Generation," in *Proceedings of the IEEE Fault-Tolerant Computing Symposium*, June 1997.

[19] P. R. Stephan, R. K. Brayton and A. L. Sangiovanni-Vincentelli, "Combinational Test Generation Using Satisfiability," Memorandum no. UCB/ERL M92/112, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, October 1992.

[20] G.-J. Tromp, "Minimal Test Sets for Combinational Circuits," in *Proceedings of the International Test Conference*, pp. 204-209, 1991.