Computational Complexity of Test-Point Insertions and Decompositions

N. S. V. Rao

Department of Computer Science
Old Dominion University
Norfolk, VA 23529-0162

S. Toida

Department of Computer Science
Old Dominion University
Norfolk, VA 23529-0162

Abstract

We consider two basic computational problems that arise in the areas of pseudo-exhaustive testing, design of polynomial-time testable classes, test-point insertion, and CrossCheck, in the context of test generation and design for testability of combinational circuits. The first problem is to decompose a circuit into subcircuits such that the number of inputs to each subcircuit is bounded by K; we show that this problem is NP-complete. This result establishes that the detection (minimization) problems associated with three polynomial-time testable classes and the method of pseudo-exhaustive testing are all NP-complete (hard). We then present simple approximation algorithms for solving these problems. The second problem deals with placing k test points on a circuit so as to facilitate the observability and/or controllability; this problem is also shown to be NP-hard. This problem arises in the methods of CrossCheck and segment-cell placement.

1 Introduction

There are several methods in the area of test generation and design for testability, and in these methods some basic computational problems appear in disguises in seemingly different contexts. We consider two such problems: circuit decompositions with bounded number of inputs for each subcircuit, and computing a subset of nets which can be used to inject and/or inspect signals. The first problem arises in the design of polynomial-time testable classes of combinational circuits [2,3,13], and in the method of pseudo-exhaustive testing [6,9,11]. The second problem occurs in the method of segment cell placement [15] and in some parts of the method of CrossCheck [16]. Despite the differences in the origins of their inception, these two problems capture the computational characteristics inherent in the underlying formulations; an investigation of these problems will be important to all these above mentioned areas.

The identification of classes of circuits that support polynomial-time test generation algorithms is very important from testing and design viewpoints; these classes are called *Polynomial-Time Testable* (PTT) classes. Typically test generation algorithms for these classes exploit the special structure of the circuits. One known approach to the design of a PTT class deals with decomposing the circuits by viewing them as graphs [2,3,13]. Fujiwara [3] proposed a polynomial-time class based on a novel decomposition of the given circuit. Each circuit is decomposed into disjoint blocks such that the graph formed by denoting each block by a node is free of cycles, and each block has no more than K inputs lines. Chakradhar et al [2] generalize this definition to include a larger class while ensuring almost the same complexity for the test generation. They allow the graph that interconnects the blocks to be a partial K-tree. However, for some polynomial-time testable circuits, both definitions yield unnecessarily high complexity. Let m and n be the number of signal lines and the logic gates respectively in the circuit. The circuits with tree topology can support the decompositions of [2,3] and thereby yield a test generation algorithm with a time complexity of $O(m16^K)$. But, the complexity of test generation for a tree circuit is $O(m)$ which in general has a very low proportionality constant compared to 16^K [4]. This problem is circumvented in Rao and Toida [13] by resorting to a decomposition that can be defined in terms of fanout and reconvergent pairs, which are regarded as a main source of intractability of test generation problems [4].

The essence of the above polynomial-time testable circuits is to decompose (under some constraints) the circuits into subcircuits such that the number of in-

0-8186-2465-5/91 $1.00 © 1991 IEEE

puts for each subcircuit is at most K [2,3,13]. Similar decompositions occur in the method of pseudo-exhaustive testing where additional elements, called the *segment cells*, are placed strategically on the nets of the circuit [6,9,11,15]; these cells are used to inject signals into the circuit and also to sense the logic levels of the nets. Here if the circuit can be decomposed into subcircuits such that each subcircuit takes its input from at most K nets then the subcircuit can be tested in at most 2^K patterns. Notice that in this case the decomposition specifies the locations for segment cells, whereas in the case of PTT-classes the decomposition enables the computation of tests. In summary, a basic computational problem in these methods is to obtain a decomposition of a graph such that the number of edges into each decomposed part is bounded by K. We show that the problem of computing such decomposition is NP-complete, and discuss some approximation algorithms.

In terms of test generation, the notions of controllability and observability of a net are very important [3]. The controllability corresponds to the complexity of generating a specified logic level at a net by suitably computing the required inputs, and the observability corresponds to the complexity of computing the inputs that will propagate a logic value of the net to the primary outputs. Now consider the problem of locating test-points - which are segment cells in case of pseudo-exhaustive testing [15] and sense points in case of CrossCheck [16] - such that the controllability and observability can be improved. In the method of CrossCheck [16] extra hardware is provided so that the logic levels of several nets can be sensed by suitably addressing and activating the sensing lines. In the cases where the sensing is time-consuming, we want to sense a minimum number of nets and infer the other levels using a fast algorithm (preferably one with a complexity of lower order polynomial in the size of the circuit). In other words, we are attempting to improve the observability of the circuit by selecting a few nets to be sensed. In the context of pseudo-exhaustive testing, we want to compute a minimum number of nets such that any logic level can be generated at any net by suitably activating the segment cells; note that the problem of placing the segment cells to improve the observability is the same as in the case of CrossCheck. We show that the problem of test-point insertion is NP-hard; in particular both the above problems turn out to be NP-hard. Our proof involves a reduction from the alarm placement problem of Rao [12], and is not directly reduced from the Boolean satisfaction problem (unlike many intractability proofs in test generation [4,8]).

The organization of this paper is as follows: We provide precise formulations of various computational problems in Section 2. We formulate the general graph decomposition problem and show its NP-completeness in Section 3; we discuss some simple approximation algorithms in Section 4. We show the intractability of the test-point insertion in Section 5.

2 Basic Computational Problems

The subsections 2.1 and 2.2 are recapitulations from [2,3,13], and the Subsection 2.3 and 2.4 are formulated based on [6,9,15,16].

2.1 Structural Decompositions

Consider a partition $\Pi=\{C_1, C_2, \ldots, C_t\}$ of a circuit C, where $C_1, C_2, \ldots, C_t$ are sub-circuits of C, called the *blocks*, and satisfy $C_i \cap C_j = \phi$ and $C = C_1 \cup C_2 \cup \ldots \cup C_t$ [3]. Consider a graph G_Π with respect to Π such that each vertex represents a block and each edge corresponds to a signal line from a gate in one block to a gate in another block. A combinational circuit C is said to be *K-bounded* if there exists a partition $\Pi=\{C_1, C_2, \ldots, C_t\}$ of C such that [3]: (a) the number of inputs of each block C_i is at most K, and (b) graph G_Π has no cycle, i.e., G_Π is a tree.

Fujiwara [3] presents a $O(16^K m)$ algorithm to find a test for a single stuck-at fault in C, where m is the number of lines in C. This algorithm works in steps. In the first step, which we call *preprocessing*, a graph based on the decomposition is computed in $O(16^K m)$ time. After preprocessing, a test can be generated in $O(m)$ time.

Chakradhar et al [2] generalize the above definition, while preserving the polynomial-time test generation, as follows: A combinational circuit is called (k, K)-*circuit* if it can be partitioned into blocks such that: (a) C_i $(1 \leq i \leq t)$ has at most K inputs, and (b) G_Π is a partial k-tree. A partial k-tree graph is a subgraph of a k-tree [1]. A k-tree is a graph that can be reduced to the k-complete graph (i.e., a fully connected graph on k vertices) by a sequence of removal of degree k vertices with completely connected neighbors. A graph G_Π with no cycles is a special case of a partial k-tree (partial 1-tree) and thus all K-bounded circuits are $(1, K)$-circuits. Chakradhar et al [2] present a test generation algorithm that computes the energy function of a neural network in a preprocessing step. The preprocessing time can be shown to be $O(16^K m)$. Then a test corresponds to a suitable minima of the

energy function, and this minima can be computed in $O(m^2)$ time.

In order to apply these algorithms in practice, we must first know that the given circuit supports these decompositions. We define the *circuit decomposition problem I* (II), denoted by CDP I (II), as follows: Given a circuit C, does there exist a decomposition of t blocks that makes it K-bounded ((k,K)-circuit)?

2.2 Decompositions Based On f-r Pairs

Given a circuit C and a fanout and reconvergent pair (f,r) of C, we define the node set $N(f,r)$ to be the union of all nodes taken over each path from f to r. We identify a set of subcircuits, called *groups*, of C as follows. Each (f,r) pair belongs to precisely one *group*, denoted by $B(f,r)$, and each group contains at least one fanout and reconvergent pair. Now we say that a circuit C is K-(f,r) *decomposable* [13], if each fanout and reconvergent pair can be included into a group such that: (a) the groups are disjoint, and (b) each group is a (k,K)-circuit for $k \leq K$, i.e., each group can be decomposed into subcircuits given by $\Pi=\{C_1, C_2, \ldots, C_t\}$ such that G_Π is a partial k-tree, and each C_i has at most K inputs.

For any $K-(f,r)$-decomposable circuit, we can obtain a polynomial time test generation algorithm as follows. We identify all $B(f,r)$s and generate the truth tables for each group while including the D and $\bar{D}$ as additional logic levels. After this step, the test for any net can be computed in a manner similar to that for a circuit without fanout and reconvergent points. Thus the test generation of a single net takes $O(m)$ time after a preprocessing step of complexity of $O(16^K n)$ [13].

We define the *circuit decomposition problem III*, denoted by CDP III, as follows: Given a circuit C, does there exist a decomposition of t blocks that makes it $K-(f,r)$-decomposable ?

Now we consider a variant of the $K-(f,r)$-decomposability where the groups need not be disjoint. We call these decompositions as K-pair-decompositions. We say that a circuit C is *K-pair-decomposable* [13], if each fanout and reconvergent pair can be included into a group such that each group is a (k,K)-circuit for $k \leq K$, i.e., each group can be decomposed into subcircuits given by $\Pi=\{C_1, C_2, \ldots, C_t\}$ such that G_Π is a partial k-tree, and each C_i has at most K inputs.

The preprocessing based on this decomposition takes $O(n^2 16^K)$ time, after which the complexity of generating a single test is given by $O(m)$ [13]. The decomposition problem corresponding to this case is denoted by CDP IV. In general for a circuit, the two decompositions based on $f-r$ pairs yield different values for K.

2.3 Pseudo-Exhaustive Testing

A circuit can be exhaustively tested by applying all possible input patterns to it; this method, however, involves prohibitively large number of test patterns given by 2^n. In the method of *pseudo-exhaustive testing*, the circuit is partitioned into subcircuits, called *segments*, and each segment is exhaustively tested [6,15]. If the number of inputs to every segment is restricted to some predetermined value K, each segment can be tested with at most 2^K patterns. A minimum number of special segmentation cells [6,9] can be placed in the original circuit to form segments. The segments need not be disjoint and each segment can have multiple outputs. The method of *verification testing* [11] is based on special type of partitions called *cones*; each cone corresponds to all gates that feed into a single output. The computational problem inherent in this case is to detect if a given circuit can be decomposed such that each subcircuit can have at most K inputs. This problem is formally defined and addressed in the next section.

2.4 Test-Point Insertion

Now consider the problem of placing test-points in a circuit such that the complexities of the problems of observability and/or controllability can be reduced to polynomials in n; in the case of pseudo-exhaustive testing a test-point corresponds to a segment-cell and in the case of CrossCheck a test point is a net to be sensed. Since the latter is subsumed by the former, we use the placement of the segment cells in our formulation. The problem of *test-point insertion* deals with placing k test-points such that (a) (*observability condition*) the logic level of output of any gate can be computed in polynomial time (in n) by observing the logic levels at the test points; (b) (*controllability condition*) a desired logic value at any net can be produced by suitably computing, in polynomial time (in n), the logic values to be injected at the test-points. The requirement (b) is not applicable to CrossCheck. In pseudo-exhaustive testing we reduce the extra hardware needed for this purpose by finding a set of locations of minimum size. In CrossCheck, by obtaining smallest subset of nets we minimize the number of sensing operations, each of which involves activating the addressing hardware and reading the logic values.

3 Decomposition Problems

We define a general graph decomposition problem and show that it is NP-complete, and then present similar results for the decompositions of [2,3,13].

3.1 General Decomposition

Given a graph $G = (V, E)$, the *general K-decomposition* is a partition $\{V_1, V_2, \ldots, V_p\}$ of V, ($V_i \cap V_j = \phi$ for $i \neq j$ and $\sum_{i=1}^{p} V_i$) such that the number of edges into each V_i is no more than K, i.e., $|\{(u,v)|u \notin V_i, v \in V_i\}| \leq K$. We consider the problem of computing a K-decomposition for given K and G, and this problem is denoted by K-GDP. Now we define *(k,K)-decomposition* of a graph $G = (V, E)$ to be a K-decomposition of G with precisely k partitions of V. The problem of computing (k,K)-decomposition is denoted by (k,K)-GDP. We only present the results in this section and the proofs can be found in our technical report [14].

Theorem 3.1. *(i) (k,K)-GDP is NP-complete. (ii) K-GDP is polynomially reducible to (k,K)-GDP.* □

This theorem establishes that (k,K)-GDP is at least as hard as K-GDP. We shall now show that K-GDP is NP-complete. First we show that a related problem, denoted by K-RGDP, is NP-complete. K-RGDP is defined as the problem of finding if G supports a K-decomposition such that at least one block has precisely K edges into it.

Theorem 3.2. *K-RGDP is NP-complete.* □

Theorem 3.3. *K-GDP is NP-complete.* □

We have shown that the problem of decomposing a circuit in a general manner is NP-complete. However, this general decomposition does not guarantee polynomial test generation algorithm. This is because there can be fanout and reconvergent pairs in different blocks, thereby preventing any polynomial time combination algorithm even after the truth tables of each block have been computed. However, even if we restrict that the resultant G_Π satisfy the conditions of [3] or [2], the complexity of these problems does not change as will be shown in the next subsection.

3.2 Polynomially Testable Classes

The problems CDP I thorough CDP IV are version of the problem (t,K)-GDP with additional constraints on the decompositions. The last section shows the latter to be NP-complete, but the complexity of the former four problems can not be directly concluded by that of the latter. Two restricted problems are obtained by restricting C to a forest and a tree respectively. More precisely, in each case we pose the problem: can a given C be decomposed into t blocks such that the number of input lines into each block is no more than K ? The former is called the *Forest Decomposition Problem* (FDP) and the latter is called the *Tree Decomposition Problem* (TDP).

Lemma 3.1. *[13] FDP is polynomially reducible to TDP.* □

Theorem 3.4. *[13] TDP is NP-complete.* □

Note that the combinational circuit with tree topology has zero fanout and reconvergent pairs, and hence the test for a single stuck-at fault can be obtained in $O(m)$ time [4]. It is interesting to note that for the same circuit obtaining the decomposition is NP-complete. The following general result has been shown in [13].

Theorem 3.5. *Given G, (i) the problem of computing a decomposition that is K-bounded is NP-complete, and (ii) the problems of computing a decomposition that is (k,K)-circuit, $K - (f,r)$-decomposable, or K-pair-decomposable are all NP-hard.* □

4 Approximation Algorithms

We first consider some useful properties of decompositions and then present simple algorithms that yield very crude approximations for the required decompositions.

Properties 4.1. *For a circuit without fanout and reconvergent points (i) best possible value of K is the maximum indegree of any node. (ii) there is always a trivial partition for $K \geq \max_{v \in V}\{indegree(v)\}$.* □

The outline of the approximation algorithm for K-bounded circuits is as follows. We identify a node v with highest indegree K_1. We obtain all fanout points that form a f-r pair with v. If there is none then by Properties 4.1, we have $K = K_1$; hence, we form each node of the graph into a component. If f-r pairs exist then we compute the number of inputs into the block formed by each pair. If there is no pair with less number of input lines than indegree of v, then return each node as a component. If not, we combine the

pair into a block and repeat the same process on the rest of the graph.

Consider the complexity of this algorithm. For each vertex v we have to consider at most n fanout points. The complexity of computing the number of input lines into the block formed by v and a fanout point is $O(m)$. Hence the total complexity of this algorithm is $O(m^3)$. The same algorithm can be extended by considering t fanout points to form a block with v. This will in general result in better values for K with an increased complexity of $O(m^t + 2)$.

The same algorithm can be applied for obtaining approximate $K - (f, r)$-decompositions. We obtain each group of the decomposition by using a variant of depth first search. Then we treat each group as a block. The time complexity of this algorithm is $O(m)$. The value of K in this case is maximum number of the input lines into any block. This algorithm can be applied to obtain K-pair-decompositions by computing the groups where each group corresponds to a f-r pair. We treat each group as a single block. The complexity of generating all groups is $O(nm)$ by using at most n invocations of a variant of depth-first search algorithm. The K values in this case will be in general smaller than that of $K - (f, r)$-decomposition.

5 Test-Point Insertion

We consider two special types of circuits: the ALL-AND and ALL-OR circuits consist exclusively of AND and OR gates respectively. The ALL-OR systems are considered by Rao [12] in solving operative diagnosis of graph-based systems; these systems are represented by graphs where nodes represent components and the edges represent fault propagation between the components. A fault can originate at any node and propagates to all nodes adjacent to it, and further propagates from those nodes and so on. Some nodes are equipped with alarms which ring (operate at logical level 1) when a fault originates at the node or propagates to the node. Under normal operations all alarms operate under logical level 0. These systems can be viewed as special cases of combinational circuits, and the NP-completeness of some problems on these systems can be used to prove NP-completeness of the test-point insertion problems. We particularly consider the *alarm placement problem for unique single fault diagnosis* [12]: can we place k alarms on ALL-OR system such that fault source of any single fault can be uniquely identified by noting the status of the alarms only. This problem has been shown to be NP-complete in [12].

First consider the observability problem that arises in CrossCheck: we would like to identify certain set of k nets such that the logic levels at the other nets can be inferred - either by direct table look-up or by an algorithm with polynomial time complexity - from the sensed values. This problem subsumes the above alarm placement problem as follows: if we solve the alarm placement problem, then for any set of readings for the alarms we can uniquely identify the fault source a. Then we perform a depth-first search from the fault source, and infer that the logic level at each visited node to be 1, and all the nodes that have not been visited will have logical value of 0. Also, a solution to the observability problem can be easily converted to that of the alarm placement problem: the gate with output 1 with all inputs being 0 will be the fault source (under the single fault assumption). Thus the test-point placement problem for observability is NP-hard.

Now consider the controllability problem, where we would like to place k segment cells such that we can produce a desired logic level at any net by suitably activating the pseudo inputs of the segment cells. We show a reduction to the controllability problem in ALL-AND systems from the alarm placement problem in ALL-OR systems; thereby, we show the NP-hardness of the former. For this reduction, convert the given ALL-OR system S_{OR} to and ALL-AND system S_{AND} by reversing all the directions of the arcs and changing the OR gates to AND gates. Let a_{AND} be the AND gate of S_{AND} corresponding to the OR gate a_{OR} of S_{OR}. First observe that a logical 0 value at any node of S_{AND} can be produced by making all the signals at the test-points to be 0. Any assignment of values to produce a logical 1 at a node a_{AND} of S_{AND} corresponds to a set of alarm readings of S_{OR} such that a_{OR} is the unique source for single fault. Thus the controllability problem on S_{AND} has a solution if and only if the alarm placement problem on S_{OR} has a solution. Since ALL-AND systems form a subclass of Boolean circuits, we established the NP-hardness of the test-point insertion problem. In summary we have the following theorem.

Theorem 5.1. *The problem of test-point insertion for observability and/or controllability for combinational circuits is NP-hard.* □

6 Summary and Conclusions

Two basic computational problems arise in the areas of pseudo-exhaustive testing, design of polynomial-time testable classes, test-point insertion, and Cross-

Check. The first problem is to decompose a circuit into subcircuits such that the number of inputs to each subcircuit is bounded by K; we show that this problem is NP-complete. The second problem deals with placing k test points on a circuit so as to facilitate the observability and/or controllability; this problem is also shown to be NP-hard. The first problem occurs in the context of pseudo-exhaustive testing and polynomial-time testable circuits and the second problem occurs in segment-cell placement and CrossCheck. These two problems do not exhaust the computational issues of these methods, and it would interesting to see if there are any other common computational problems to these methods. Also, it is worthwhile to abstract some underlying computational problems in other areas of test generation and design for testability of combinational as well as sequential circuits.

Acknowledgements

The work of the first author has been partially funded by Old Dominion University Summary faculty Award for 1991 and by National Science Foundation under grant #IRI-9108610.

References

[1] S. Arnborg, D.G. Corneil, A. Proskurowski, "Complexity of finding embeddings in a k-tree," *ISIAM J. Algebraic and Discrete Methods*, Vol. 8, No. 2, pp. 277-284, 1987.

[2] S.T. Chakradhar, V.D. Agrawal, M.L. Bushnell, "Polynomial time solvable fault detection problems," *Proc. Fault-Tolerant Computing Symp.*, pp. 56-63, 1990.

[3] H. Fujiwara, "Computational complexity of controllability/observability problems for combinational circuits," *Proc. Fault-Tolerant Computing Symp.*, pp. 64-69, 1988.

[4] H. Fujiwara and S. Toida, "The complexity of fault detection problems for combinational logic circuits," *IEEE Trans. Computers*, Vol. C-31, No.6, pp. 553-560, 1982.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., San Francisco, 1979.

[6] S. Hellerbrand and H.J. Wunderlich, "Tools and devices supporting the pseudo-exhaustive test," *Proc. 1st European Design Automation Conf.*, March 1990.

[7] T. Ibaraki, T. Kameda and S. Toida, "On minimal test sets for locating single link failures in networks," *IEEE Trans. on Computers*, Vol. C-30, No.3, pp. 182-190, 1981.

[8] P. H. Ibarra and S. K. Sahni, "Polynomially complete fault detection problems," *IEEE Trans. Computers*, Vol. C-24, pp. 242-249, 1975.

[9] W.B. Jone and C.A. Papachristou, "A coordinated approach to partitioning and test pattern generation for pseudoexhaustive testing," *Proc. Design Automation Conf.*, pp. 525-530, 1989.

[10] D.S. Johnson, "Approximation algorithms for combinatorial problems," *J. of Computer and System Sciences*, Vol.9, pp. 256-278, 1974.

[11] E.J. McCluskey, "Built-in verification test," *Proc. Int. Test Conf.*, pp. 183-190, 1982.

[12] N.S.V. Rao, Computational complexity issues in operative diagnosis of graph-based systems, Tech. Rep. #91-08, Dept. Computer Sci., Old Dominion Uni., 1991.

[13] N.S.V. Rao and S. Toida, "On polynomial-time testable classes of combinational circuits," *Proc. 1991 IEEE VLSI Test Symp.*, Atlantic City, NJ, pp. 172-177,1991.

[14] N.S.V. Rao and S. Toida, "Computational complexity of test-point insertions and decompositions," Tech. Rep., Dept. Computer Sci., Old Dominion Uni., 1991.

[15] R. Srinivasan, C.A. Njinda, M.A. Breuer, "A partitioning method for achieving maximal test concurrency in pseudo-exhaustive testing," *Proc. 1991 IEEE VLSI Test Symp.*, pp. 34-39, 1991.

[16] G. Swan, Y. Trivedi, D. Wharton, "CrossCheck - A practical solution for ASIC testability," *Proc. 1990 Int. Test Conf.*, pp. 903-908.