Combinatorial Group Testing Methods for the BIST Diagnosis Problem

Andrew B. Kahng

CSE & ECE Departments
University of CA, San Diego
La Jolla, CA 92093
e-mail: abk@cs.ucsd.edu

Sherief Reda

CSE Department
University of CA, San Diego
La Jolla, CA 92093
e-mail sreda@cs.ucsd.edu

Abstract— We examine an abstract formulation of *BIST diagnosis* in digital logic systems. The BIST diagnosis problem has applications that include identification of erroneous test vectors, faulty scan cells, faulty modules, and faulty logic blocks in FPGAs. We develop an abstract model of this problem and show a fundamental correspondence to the well-established subject of Combinatorial Group Testing (CGT) [7]. Armed with this new perspective, we show how to improve on a number of existing techniques in the VLSI diagnosis literature. In addition, we adapt and apply a number of CGT algorithms that are well-suited to the diagnosis problem in the digital realm. We also propose completely new methods and empirically evaluate the different algorithms. Our experiments show that results of the proposed algorithms outperform recent diagnosis techniques [8, 9, 14]. Finally, we point out future directions that can lead to new solutions for the BIST diagnosis problem.

I. Introduction

In this paper, we introduce the diagnosis problem in the Combinatorial Group Testing (CGT) field [7] and then examine how it is closely related to the BIST diagnosis problem. We use this link to establish a wealth of techniques for certain classes of BIST diagnosis problems. The abstract diagnosis problem in CGT is concerned with identifying a subset of faulty items from within a general set of items using a *tester*. Given a subset of items, the tester gives a *yes* answer if *queried* with a subset that has faulty items, and a *no* answer if the subset is fault-free. The diagnosis problem is formally defined as follows.

Diagnosis Problem: Given a set M consisting of n modules (items), identify the subset of faulty modules $F \subseteq M$, where $d = |F|$ is unknown, using the *minimum* number of *queries* to the tester.

Recently, Built-In-Self-Test (BIST) has emerged as the leading solution for addressing today's test challenges. BIST offers the promise of low hardware overhead with the clear advantage of at-speed testing. For example, a scan-based BIST session will generate a number of test patterns through a built-in test generator (a linear feedback shift register, or LFSR). Test responses are captured by the scan chain and then compacted by feeding into a multiple-input shift register (MISR), as shown in Figure 1.

One of the main shortcomings of the above test approach is that once a faulty chip is detected, the signature offered by the MISR gives no help toward diagnosing the failure. The MISR output is typically interpreted as pass/fail information with little additional value. Hence, fault diagnosis in scan-based environments usually falls into one of two problem types. The first problem is concerned with identifying the scan cells that capture erroneous responses during the BIST session [14, 9, 8, 4, 3]. The second problem is concerned with identifying the set of failing test vectors [12, 13, 8]. A third related BIST-diagnosis problem is concerned with the identification of faulty logic blocks in FPGAs [15, 1].

These classes of diagnosis problems in BIST environments may at first glance seem unrelated. Yet abstractly, *in their essence* they are identical to the diagnosis problem in CGT. In particular, they involve a set of items (scan cells, or test vectors), which we refer to generically as *modules*, with some number of these modules being faulty. The BIST diagnosis problem seeks to identify all faulty modules using the minimum number of tests, i.e., with minimum use of the response compactor (e.g., the MISR in scan-based BIST).

Some of the challenges in the diagnosis process are:

1. achieving full diagnostic information, i.e., detecting

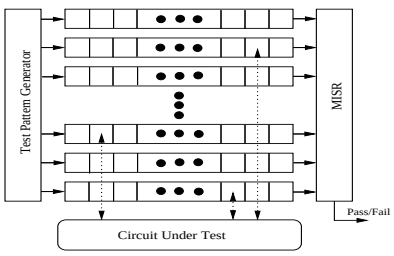

Fig. 1. Scan-Based BIST Environment.

all faulty items (whether scan cells, test vectors or logic blocks);
2. minimizing diagnosis time, since this translates to reduction in total testing time; and
3. minimizing hardware overhead, i.e., the amount of hardware needed to support BIST diagnosis.

The link between digital system BIST and CGT affords a rich set of techniques to attack the aforementioned questions. We show that a number of established BIST techniques can be improved by CGT approaches. We also directly apply CGT techniques to the BIST diagnosis problem. Furthermore, we propose completely new methods that are based on "hybrid" integration of existing techniques. A number of research directions remain open, and offer the prospect of more cost-effective BIST diagnosis solutions as well as new theoretical frameworks. In the remainder of this paper, Section 2 presents several new diagnosis techniques. Section 3 empirically assesses these algorithms. Section 4 addresses some of the practical concerns usually associated with BIST environments. Finally, Section 5 lists directions for future research.

II. New Algorithms for Diagnosis

In this section we present new diagnosis algorithms for the BIST diagnosis problem. We adapt and present four algorithms from the CGT literature: *digging, multi-stage batching, doubling* and *jumping*. We also propose two completely new diagnosis algorithms, which we refer to as *batched digging* and *batched binary search*.

A. Digging

From the CGT literature, we present the *digging* algorithm which can be considered as an improvement to the notable Binary Search approach of Ghosh-Dastidar and Touba [9]. Digging reduces the number of queries (test sessions), especially for low values of d (faulty items) [10, 7]. Observe that if there are two defective sets M_1 and M_2, with $M_1 \subset M_2$, then the result of the query M_1 renders the result of the query on M_2 useless. Hence, with binary search there is the potential for many queries to produce no additional information for the diagnosis process. This suggests that once a faulty set of modules M_f is found, a faulty module m_f should be identified from the set. This process is referred to as *digging* [7]. Once a faulty module m_f is identified, m_f is removed from M_f, and digging is resumed on the remaining items. Digging requires $d \log n$ queries. For small values of d, digging improves binary search, as we confirm below in our discussion of experimental results. We present the DIG and DIG-BS (binary search with digging) procedures in Figures 2 and 3. Note that in this and subsequent algorithm descriptions, we use the term "ordered" to indicate that modules in M are indexed to permit reference to individual modules or ranges of modules.

Input: M = set of modules under test, with one or more faulty modules.
Output: m_f = faulty module.

Set $B = M$
do
  $B_1 = \lceil \frac{|B|}{2} \rceil$ modules from B
  If query(B_1) then $B = B_1$
  else $B = B \setminus B_1$
while $|B| > 1$
Return m_f, where $B = \{m_f\}$

Fig. 2. The DIG procedure to identify one module from a set of faulty modules.

Input: M = (ordered) set of modules under test.
Output: $F \subseteq M$ = set of faulty modules.

Set the queue $Q = \{M\}$, and $F = \emptyset$.
do
  Pop the head element S of Q
  If query(S) then
    m_f=DIG(S)
    Set $M = M \setminus \{m_f\}$ and $F = F \cup \{m_f\}$
    Bisect M into M_1 and M_2
    If $M_1 \neq \emptyset$ then insert M_1 in Q
    If $M_2 \neq \emptyset$ then insert M_2 in Q
while $Q \neq \emptyset$
Return F

Fig. 3. Binary search using a number of DIGs.

B. Multi-Stage Batching

In this technique [5, 11], the set M of modules under test is divided into disjoint, equal-size *batches* $b_1, b_2, \ldots, b_k$, where $1 \leq k \leq n$. Each of these batches is then tested. Once a set of faulty batches $b_{f_1}, b_{f_2}, \ldots, b_{f_m}$ (with $m < k$) have been identified, one can re-merge all the faulty batches into one set $M_{s_2} = \bigcup_{i=1}^{m} b_{f_i}$, then re-apply the same procedure recursively. This procedure can be repeated for any number of stages until the point where all the batches are identified as faulty. In this last case, we query all the modules of the batches. We remind the readers that the approach of [8] tests all elements in any faulty batch and hence can be considered as a "one-stage" batching algorithm. With multi-stage batching, the number of queries $q(n)$ is given by

$$q(n) \leq \frac{n}{k_1} + \frac{dk_1}{k_2} + \cdots + \frac{dk_{s-1}}{k_s} + dk_s \qquad (1)$$

where k_i, $1 \leq i \leq s$, is the number of batches for stage i. The values of k_i, $1 \leq i \leq s$ set the performance of the algorithm. In order to estimate a reasonable value for k_i, we investigate the special case of one faulty item, i.e., $d = 1$. If n is viewed as continuous, then from basic calculus $q(n)$ is minimized when $k = \sqrt{\frac{n}{d}}$. Hence, we may set $k_i = \sqrt{n_{i-1}}$, where n_i is the total number of modules in faulty batches at stage i ($n_i \leq dk_i$) and $k_1 = \sqrt{n}$. We will see in the experimental results section below that the multi-stage strategy offers a considerable reduction in the number of queries compared to the approach of [8].

The Multi-Stage Batching algorithm, which we refer to as MULTISTAGE, is formally described in Figure 4.

Input: M = (ordered) set of modules under test.
Output: $F \subseteq M$ = set of faulty modules.

Set $n = |M|, b = \sqrt{n}, k = 0$, $U = \emptyset$, and $F = \emptyset$
While(true)
  Set $l = k \cdot b$
  Set $h = \min((k+1) \cdot b, |M|)$
  If $l > |M|$ then break
  If query($M[l..h]$) then $U = U \cup M[l..h]$
  Set $k = k + 1$
If $|U| \neq n$ then $F = F \cup$ MULTISTAGE(U)
else for $u = 1$ to $|U|$:
  If query($M[u]$) then $F = F \cup M[u]$
Return F

Fig. 4. MULTISTAGE procedure for multi-stage batching diagnosis [11].

C. Doubling and Jumping

Another diagnosis algorithm is due to [2]. Given that the value of d is unknown, the algorithm attempts to estimate the value of d. If d is small then the algorithm finds large fault-free sets; otherwise, the algorithm finds small faulty sets. To deliver this functionality, the algorithm tests disjoint sets of sizes $1, 2, 4, \ldots, 2^i$ until a faulty set is found. At this point, the algorithm has identified $2^i - 1$ fault-free modules and a faulty set of size 2^i, using i tests. The algorithm then identifies a faulty module from the faulty set using binary search, which requires i queries. Consequently, the algorithm uses $2i + 1$ queries and detects 2^i items ($2^i - 1$ fault-free and 1 faulty). We present this DOUBLING algorithm in Figure 5.

An interesting modification to Doubling, which is called Jumping, is due to [6]. Instead of testing disjoint sets of sizes $1, \ldots, 2^i$ as in Doubling, Jumping tests sets having sizes $1 + 2, 4 + 8, \ldots, 2^i + 2^{i+1}$ until a faulty set is found. Using these "jumps" (i.e., in the ordering of the modules), the algorithms identifies fault-free modules with $\frac{i}{2}$ tests instead of i tests. However, a faulty set is of size $3 \cdot 2^i$, rather than of size 2^i as in Doubling; it therefore requires more than one query on a subset of size 2^i to reduce the faulty set to either 2^i or to size 2^{i+1} with 2^i fault-free items. We do not describe details of the Jumping algorithm here, but we assess it empirically in the experimental section below, and refer the interested reader to [6] or [7] (page 134).

D. New Hybrid Methods: Batched BS and DIG-BS

In this subsection, we propose a new modification of existing algorithms. One of the main shortcomings of batching algorithms is their relative poor performance for small and moderate values of d. This is mainly due to the need to query all faulty items in the faulty batches. On the other hand, binary search based algorithms excel for small values of d, and rapidly deteriorate as d approaches

Input: M = (ordered) set of modules under test.
Output: $F \subseteq M$ = set of faulty modules.

Set $F = \emptyset$.
while $M \neq \emptyset$
  if query(M) then $M = \emptyset$
  Set $k = 1$ and $S = \emptyset$
  while (true)
    $S = \min(k, |M|)$ items from M
    if query(S) = false then
      set $k = 2 \cdot k$ and $M = M \setminus S$
    else break
  Set m_f =DIG(S)
  Set $M = M \setminus \{m_f\}$ and $F = F \cup \{m_f\}$
Return F

Fig. 5. The DOUBLING algorithm.

the total number of modules n. Hence, one way to improve results for small and moderate values of d is to start by batching; then, once the faulty batching sets are identified, Binary Search (BS) or Digging (DIG-BS) may be applied to identify the faulty modules within these sets. In this hybrid method, batching is used to initially prune a large portion of the search space, clearing the way for binary search to identify the faulty modules with fewer queries. We will see below that the proposed strategy outperforms the other techniques for a significant range of practical values of d. The Batched DIG-BS algorithm is described in Figure 6; a Batched BS algorithm based on binary search rather than digging may be similarly conceived.

Input: M = (ordered) set of modules under test.
Output: $F \subseteq M$ = set of faulty modules.

Set $n = |M|, b = \sqrt{n}, k = 0$, and $F = \emptyset$
While(true)
  Set $l = k \cdot b$
  Set $h = \min((k+1) \cdot b, j)$
  If $l > n$ then break
  If query(M, l, h) then DIG-BS($M[l..h]$)
  Set $k = k + 1$
Return F

Fig. 6. Batched DIG-BS diagnosis.

III. Experimental Results

In this section, we empirically assess the aforementioned diagnosis techniques[1]. We use an experimental

[1]While all the techniques that we have discussed are adaptive, the CGT literature is also rich with non-adaptive algorithms. A number of non-adaptive algorithms have been proposed for the diagnosis problem as well. For example, Rajski et al. [14] propose a probabilistic approach for partitioning scan cells to identify the faulty ones, while Bayraktaroglu et al. [3] extend this approach to deterministically partition the set of scan cells to achieve full-diagnostic resolution. This deterministic partitioning approach is

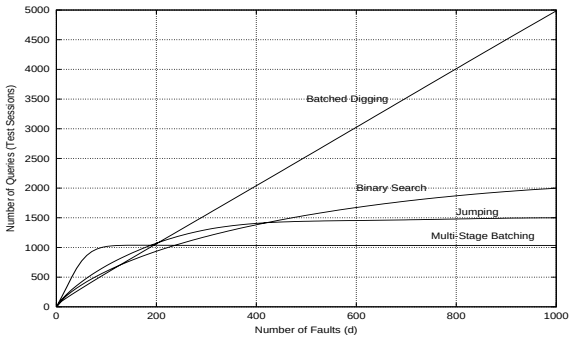

Fig. 7. Performance of selected algorithms for $n = 1000$.

setup similar to [9] where we assume a set of 1000 modules representing scan cells (experiments for test-vectors, electronic modules or FPGA logic blocks can be similarly devised). We randomly set d scan cells to be faulty and calculate the number of queries (test sessions) needed to detect all faults. For each value of d, we generate 100 random instances and report the average number of queries. *The number of queries is directly proportional to the amount of time needed to diagnose the BIST system.*

We compare the proposed algorithms against our implementations of [8][2] and [9] which are reported to dominate [14]. For space limitations, we give our results in Figure 7 and summarize our experimental conclusions as follows.

- For a number of faulty modules less than $\approx 1.3\%$, *Digging* noticeably outperforms *Binary Search* reducing the number of test sessions by about 15%. After that point, Binary Search dominates Digging.
- *Multi-Stage* Batching dominates *One-Stage Batching* for most values of d leading to a reduction in the number of test sessions by around 25-50%. For large percentages of faulty modules $> 25\%$, both versions of batching dominate binary search based algorithms by a very wide margin.
- *Jumping* dominates *Doubling* for any number of faulty modules. Furthermore, *Jumping* dominates *binary search* based techniques when the incidence of faulty modules exceeds $d > 500$. This reduces the number of test session by around 25%.
- For percentages of faulty modules that are greater than 1.3% and less than 10%, *Batched BS and Batched DIG-BS* outperform other algorithms by a wide margin. They reduce the number of test sessions by around 7-20%.

IV. conclusions

Our work has addressed the issue of BIST diagnosis and revealed previously unexplored connections to the field of Combinatorial Group Testing. We summarize our contributions as follows.

- We show that the BIST diagnosis problem corresponds precisely to the heart of the well-established field of CGT. This relationship between the two fields opens up an enormous wealth of techniques to the diagnosis problem in BIST.
- We improve on existing techniques, and furthermore, we point out parallels between BIST diagnosis results and earlier CGT results.
- From the CGT literature, we have adapted and extended several algorithms for the diagnosis problem, and moreover proposed new techniques. Empirical assessment of these techniques show that they outperform existing diagnosis results.

Four directions for future work are:

- competitive CGT for theoretical benchmarking of the different diagnosis techniques;
- non-adaptive diagnosis techniques using binary superimposed codes;
- special cases of diagnosis wherein the number of faults is known *a priori*; and
- diagnosis in the presence of unreliable tests, where the query response might be erroneous as with aliasing cases of MISRs.

achieved through what the authors call *Quotient Uniform Partitions*. However, *Quotient Uniform Partitions* correspond to the *Balanced Incomplete Block Design* (BIBD) codes for non-adaptive diagnosis (see [7], page 75). We refer readers interested in non-adaptive diagnosis techniques to Chapter 4 of [7].

[2]We use a batch (partition) size of $\sqrt{n}$.

References

[1] M. Abramovici and C. E. Stroud. BIST-based Test and Diagnosis of FPGA Logic Blocks. *IEEE Trans. on Very Large Scale Integration Systems*, 9(1):159–172, 2001.

[2] A. Bar-Noy, F. Hwang, H. Kessler and S. Kutten. A New Competitive Algorithm For Group Testing. *Discrete Applied Mathematics*, 52:29–38, 1994.

[3] I. Bayraktaroglu and A. Orailoglu. Deterministic Partitioning Techniques for Fault Diagnosis in Scan-Based BIST. In *Proc. of International Test Conference*, pages 273–282, 2000.

[4] I. Bayraktaroglu and A. Orailoglu. Improved Fault Diagnosis in Scan-based BIST via Superposition. In *ACM/IEEE Proc. of Design Automation Conference*, pages 55–58, 2000.

[5] R. Dorfman. The Detection of Defective Members of Large Populations. *Ann. Math. Statistics*, 14:436–440, 1943.

[6] D. Z. Du, G.-L Xue, S.-Z Sun and S.-W. Cheng. Modifications of Competitive Group Testing. *SIAM Journal on Computing*, pages 82–96, 1994.

[7] Ding-Zhu Du and Frank K. Wang. *Combinatorial Group Testing And Its Applications*. World Scientific, first ed., 1994.

[8] J. Ghosh-Dastidar, D. Das and N. Touba. Fault Diagnosis in Scan-based BIST using Both Time and Space Information. In *Proc. of International Test Conference*, pages 95–102, 1999.

[9] J. Ghosh-Dastidar and N. Touba. A Rapid and Scalable Diagnosis Scheme for BIST Environments with a Large Number of Scan Chains. In *Proc. of VLSI Test Symposium*, pages 79–85, 2000.

[10] F. K. Hwang. A Method for Detecting All Defective Members in A Population By Group Testing. *J. Amer. Statist. Assoc*, 67:605–608, 1972.

[11] C. H. Li. A Sequential Method For Screening Experimental Variables. In *J. Amer. Statist. Assoc.*, volume 57, pages 455–477, 1962.

[12] W. H. McAnney and J. Savir. There is Information in Faulty Signatures. In *Proc. of International Test Conference*, pages 630–636, 1987.

[13] Jacob Savir. Salvaging Test Windows in BIST Diagnostics. *IEEE Transactions on Computers*, 47(4):486–491, 1998.

[14] J. Rajski and J. Tyszer. Fault Diagnosis in Scan-Based BIST. In *Proc. of International Test Conference*, pages 894–902, 1997.

[15] S. Wang and T. Tsai. Test and Diagnosis of Faulty Logic Blocks in FPGAs. In *IEEE Proc. of International Conference on Computer Aided Design*, pages 722–727, 1997.