Available online at www.sciencedirect.com
SCIENCE DIRECT®
Microelectronics Reliability 45 (2005) 163–173
MICROELECTRONICS RELIABILITY
www.elsevier.com/locate/microrel

Test generation for technology-specific multi-faults based on detectable perturbations

Andrej Zemva *, Baldomir Zajc

Laboratory for Integrated Circuits Design, Faculty of Electrical Engineering, University of Ljubljana, Trzaska c. 25, SI 1000 Ljubljana, Slovenia

Received 28 January 2004; received in revised form 14 April 2004
Available online 17 July 2004

Abstract

In this paper, we introduce the concept of detectable perturbations as a method to generate tests that cover any technology-specific faults such as multiple bridging, open and stuck-at faults. Rather than devising a customized test pattern generation system for each class of technology-specific faults, we implemented a generic system to generate tests for single and multiple perturbations. We demonstrate the versatility of this approach by generating tests for a set of large benchmark circuits that have been mapped into single- and multi-output modules. These tests cover single stuck-at, multi-output bridging, stuck-at, as well as any mutation faults in the functionality of the technology-mapped cells. Experimental results provide useful insights about the quality of single stuck-at test patterns versus coverages for the additional classes of faults.

1. Introduction

Advances in CMOS technology and logic synthesis are being combined to introduce multi-output logic cells for placement and routing. Merging functionality of several single-output modules into a single cell typically reduces both the area occupied by active devices as well as the area taken by the interconnect between the cells [1,2].

The presence of multi-output cells raises new questions about fault modeling and testing: how effective is the traditional single stuck-at fault test when considering that module outputs may be pairwise bridging, stuck-at some constant values, may be interchanged, or are simply performing some other function not originally intended. Familiar examples of functional mutations on wires are related to manufacturing defects such as AND/OR bridging faults or pairwise stuck-at faults. Manufacturing defects may subtly change behavior of complex logical nodes in ways not detectable by single stuck-at fault tests [3,4], Alternatively, such nodes can also be incorrectly assigned functions during the specification phase—for example, instead of OR one may well specify XOR and escape detection.

In this paper, we address the problem of testing for multi-faults by considering the problem of detecting *functional mutations* on wires as well as on logic nodes. The space of mutation faults, even without considering mutations of logic nodes, grows exponentially with the number of wires. Just for a pair wires, we can consider 255 cases of distinct 2-input 2-output functional mutations. Single, pairwise stuck-at and bridging faults are only special cases of such mutations. Rather than devising specialized algorithms to detect each specific mutation of interest, we introduce a generic concept of detectable perturbations. By detecting a relatively small set of such perturbations, we will have covered *completely* a much larger set of all possible functional mutations. We demonstrate that detection of such perturbations is closely related to detection of a sequence of independent perturbations from a *single source*, and can

* Corresponding author. Tel.: +386-14768346; fax: +386-14264630.
E-mail address: andrej.zemva@fe.uni-lj.si (A. Zemva).

0026-2714/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.microrel.2004.04.025

thus use state-of-the-art single stuck-at test generation techniques to generate *complete* perturbation tests for a set of large benchmark circuits.

We motivate the proposed approach in Section 2, then formalize the notion of detectable perturbations in Section 3. In Section 4, we present the covering of *any* mutation fault in terms of detectable perturbations and set up the benchmarking experiment. In Section 5, we report on the experimental results.

2. Motivation

Consider a *black box* with k input pins from the set χ and p output pins from the set $\mathscr{Y}$. Nominally, this box realizes a function $\mathbf{Z}(\chi): B^k \to B^p$, representing the behavior of either a hardware module or a software module. Upon embedding into a larger hardware system or a software program, its I/O may no longer be directly accessible to control and to observe. The problem of testing is the same whether the *black box* contained in the larger system is faulty due to a manufacturing defect, whether its specification has been entered incorrectly, or whether an entirely wrong box has been embedded by mistake. The community concerned with testing hardware failures relies to a large extent on *single stuck-at* (0/1) *fault models* on the boundary of the box [5,6]. The community concerned with testing computer programs and specifications considers the *single* stuck-at fault models inadequate and attempts to derive alternative strategies [7]. Some of such strategies also consider notions of *mutation faults* [8,9]. In the context of the *black box* with k input pins and p output pins, one may thus consider two extremes: between $2(k+p)$ single stuck-at faults and $p^{(p2^k)} - 1$ mutation faults.

Consider an example with $k = 4$, $p = 2$ and two expressions. For purpose of illustration, we share no logic:

$$Z_1 = (x_1 \text{ OR } x_2) \text{ OR } (x_3 \text{ OR } x_4),$$

$$Z_2 = (x_1 \text{ OR } x_2) \textit{ NOR } (x_3 \text{ OR } x_4).$$

A test set $\{0000, 1000, 0100, 0010, 0001\}$ covers all single stuck-at faults. However, the *same test set* also covers all single stuck-at faults in these expressions:

$$Z_1 = (x_1 \text{ XOR } x_2) \text{ XOR } (x_3 \text{ XOR } x_4),$$

$$Z_2 = (x_1 \text{ XOR } x_2) \textit{ NOR } (x_3 \text{ XOR } x_4).$$

In this example, the size of a single stuck-at fault set at the pins is simply $2 \times 6 = 12$ while there are $2^{(2 \times 2^4)} - 1 = 4,294,967,295$ mutations that may be considered in the extreme case. Clearly, the *choice for a best subset of mutations* to generate a 'good test' for most 'typical mutations', given the initial *black box* specification, will continue to be subject of conjectures. What may be a reasoned conjecture for a hardware test, supported by reports of failures during device testing and returns from the field, may not lead to a good fault model for testing the hardware specification in software, and vice versa.

The concept of *detectable perturbations* as introduced in this paper induces tests that will implicitly cover the set of *all* $2^{(p2^k)} - 1$ mutation faults. Given a *black box* with k input and p output pins, we consider 2^k possibilities, inducing a k-in-p *perturbation* in up to $(2^p - 1)$ possible ways onto the p output pins. The upper bound on the set of all perturbations that we will consider is thus $2^k(2^p - 1)$—a bound much lower than the size of the set of all possible mutations. For the example above, the 4, 294, 967, 295 mutations of a *single module* can *all* be covered by tests for up to $2^4(2^2 - 1) = 48$ perturbations!

3. Detectable perturbations

Traditional single and multiple stuck-at and bridging faults are special cases from the set of mutation faults. We consider a combinational or a sequential synchronous multiple-level Boolean network modeled as a directed hypergraph. In Fig. 1 we show an example, including a 4-in-3 perturbation. More specific illustrations are introduced in Fig. 2. Several representations of the same example are shown in Fig. 3(a)–(c). Fig. 3(a) is the initial circuit specification graph, Fig. 3(b) is the optimized granular representation of the same circuit, and Fig. 3(c) is the hypergraph created after the technology mapping into a specific library set. Mutation faults can be associated with wires as well as logic nodes.

3.1. Boolean network

We model this network as a directed hypergraph $\mathscr{G} = (\mathscr{V}, \mathscr{E})$. Since we will relate nodes and wires in this graph to fault sets later on, we will refer to edges in this graph as wires and elements of the vertex set as nodes. All nodes are considered to be of one of the following types: primary and pseudo-primary inputs (PIs, PPIs), primary and pseudo-primary outputs (POs, PPOs), register nodes, cluster nodes, and fanout nodes. By definition, all register nodes are implicitly synchronized with a single clock, have a single data input driven by a pseudo-primary output node and a single data output sourced by a pseudo-primary input node. A cluster node can have any number of inputs and any number of outputs, each output can describe a logic function of arbitrary complexity. Each cluster node is a direct acyclic graph (DAG) where, by definition, all logic nodes have single outputs only. By replacing all cluster nodes in the hypergraph with their respective graph representation, we have a DAG between (PIs, PPIs) and (POs,