When N-Detect Doesn't Detect: Cell-Aware-Type Faults and Stuck-At ATPG

Fanchen Zhang, Micah Thornton, Jennifer Dworak

***Abstract*—Cell-aware faults have been proposed to more effectively detect defects within gates. We investigate the effectiveness of different types of ATPG for efficiently detecting cell-aware-type faults that are most important during functional operation.**

CONTENTS

I. INTRODUCTION

II. PREVIOUS WORK

Chip design manufacturers have always needed ways to speed up the testing process in order to push out their products more quickly. As a side effect of that, there has been much work done on n-detect, cell-aware, and stuck-at testing as these methods have been shown to significantly speed up the process. The optimization of test sets has been a long sought after goal one method of achieving this optimization is fault criticality as discussed by Shi et al [**?**] [**?**].In their paper on fault models, Hapke, and Schloeffel introduced the method of cell-aware test pattern generation [**?**]. Similarly, in their paper: Rajski et al examined the usefulness of the cell-aware model in improving tests as well as maintaining the efficiency of previous models (as related to 20 nm technologies) [**?**]. Again in later research Hapke, this time collaborating with Rajski et al examined the results of their use of cell aware test pattern generation in order to test 1 900 different cells in 800 000, 32 nm components, and determined that defect-rate was reduced overall [**?**]. Hapke et al again proved the usefulness of the new cell aware fault modeling system when they tested 1 500 CMOS 65 nm cells that were being used in automobiles [**?**]. In yet another earlier paper Hapke et al compared the new cell aware approach to previous approaches such as: "gate-exhaustive, n-detect, or embedded multi-test" [**?**]. In our research we also take these other methods into account, but also realize that we don't need to have a product which is 100% defect free, and that we are really using the error tolerance model as described by Fang et al in relation to video compression hardware [**?**].

N-Detect generally seems as though it decreases the defect rate in hardware while increasing the test set size, as shown by Huang [**?**] as well as McCluskey et al [**?**]. Later Alves et al examined the implicit increase in size of the test set based on n-detect and proposed methods for decreasing the test set size [**?**].

Our research inherently relies on being able to define our own fault model to use in test simulation. There are several ways to come up with your own fault model. We defined our model in a UDFM(User Defined Fault Model). This method relies on having what are called fault tuples as described by Dwarakanath [**?**].

III. ALGORITHM/ PROBLEM ETC.

A. Cell-Aware Fault Models and their Relation to Stuck-at ATPG

1) Cell-aware fault detection conditions:

2) Why n-detect ATPG Test Sets May Be Biased:

B. Determining which Cell-aware faults to target

IV. EXPIREMENTAL SETUP AND RESULTS

A. Expiremental Setup

To approve the effectiveness of functional cell-aware detection, s38584, s38417, s15850, s13207 and s9234 are chosen as benchmarks since they have more kind and larger number of gates among ISCAS89.

The experiment Testing flow is shown in Fig.4. We create two different Cell-Aware fault sets except for two inputs gates as described in section 3. (Cell-Aware Fault Models and their Relation to Stuck-at ATPG), replaced all primitive gates with two more inputs by cell library. One Scan-Chain is inserted in each benchmark for testing. Then ATPG testing pattern sets for Stuck-at and Cell-aware faults are obtained on Mentor Graphics Tessent. To achieve N-detect of Stuck-at model effects on Cell-Aware faults, four setting up arguments for ATPG Stuck-at are implemented; they are n0 (disable multiple detection function), n1 (guaranteed detections=1, desired detections=3), n2 (guaranteed detections=2, desired detections=5) and n3 (guaranteed detections=3, desired detections=7). According to section 3, a test bench in which initializes DFFs as 0 to engender DFFs into stable state is employed for capturing Good-State sets when Synopsys VCS simulates a golden benchmark. Note because the functionality of benchmarks are unknown, random patterns are created to represent the real functional inputs. In order to balancing randomness, we do two times of creating random pattern sets for each number of Good-States. Since UDFM of Tessent does not support N detection, we split Good-state sets and APTG sets into 5 patterns per simulation pattern file and repeat simulation processes multiple times.

V. CONCLUSIONS