Software-Based Embedded Core Test Using Multi-Polynomial for Test Data Reduction

Soyeon Kang, Inhyuk Choi, Hyeonchan Lim, Sungyoul Seo and Sungho Kang
Department of Electrical & Electronic Engineering
Yonsei University
Seoul, Korea
{adhoner, ihchoi, lhcy92, sungyoul}@soc.yonsei.ac.kr and shkang@yonsei.ac.kr

Abstract— Software-based self-test (SBST) is a self-test where processors and intellectual property (IP) cores test itself using an embedded memory. However, an environment-limited memory size is one of the biggest challenges. In this paper, we present a new SBST solution using multiple polynomials. For reducing the required test data, the polynomials consist of a primitive polynomial and (BM)-algorithm based polynomials and each polynomial generates pseudo random patterns and deterministic patterns respectively. Experimental results show that this SBST method reduces the size of the test program without a reduction of the fault coverage.

Keywords; SBST; IP-core test; Polynomial

I. INTRODUCTION

Increased number of transistors and smaller size of system-on-chip (SoC) made the testing more difficult. The main reason is that the accessibility of the embedded cores is limited and the amount of the data for testing is increased. Built-in self-test (BIST) and scan architecture are regarded as a solution to improve testability. BIST and scan architecture can achieve high fault coverage in return for introducing some modifications of the design. This modification increases the area overhead, deteriorates the performance, and generates excessive power during testing.

To overcome these problems, software-based self-test (SBST) is getting attention. Instead of utilizing an external tester, SBST uses on-chip resources. It downloads a test program to the memory, and tests processor and intellectual property (IP) cores. For this reason, SBST can test at-speed without any modification of the circuit which may result in the degradation of the circuit function [1].

The proposed SBST is depicted in Figure 1. The proposed method generates both pseudo random and deterministic patterns to test IP cores. Polynomials and seeds are downloaded from ATE to the on-chip memory. Based on the polynomials, the processor generates software-based linear-feedback shift-register (LFSR) and produces test vectors using each corresponding seed. The test vectors test the IP cores and the responses are transmitted back to the memory after compressed by multiple-input signature register (MISR). Experimental results show that the suggested method reduces the software size while the fault coverage remains same to the original test vectors.

II. PROPOSED IDEA

Since the size of a memory inside a chip is limited, it is impossible to store all deterministic patterns for all IP cores in the memory. Therefore, the effort to reduce the test software size is inevitable. The proposed SBST uses the pseudo random patterns in advance the deterministic patterns to reduce the amount of data which needs to be stored in the memory. The rest of the faults which are random pattern resistant, are detected by the deterministic patterns.

Both pseudo random and deterministic patterns are stored as a form of polynomials and seeds. Pseudo random patterns are generated from the primitive polynomials and deterministic patterns are generated from the BM-algorithm based polynomials [2,3]. Procedure 1 shows sequences generating the polynomials, Gen Poly. To generate the polynomials, original test vectors should be generated by automatic test pattern generator (ATPG). Then a primitive polynomial and a seed is generated based on the length $l$ which is user defined parameter (line1-2). Using this primitive polynomials, pseudo random patterns are generated and the original test vectors collapsing with the pseudo random patterns are removed (line3-10). The remainder patterns are used as inputs to generate BM-based polynomials (line11-14).

In the procedure 2, test vector generating sequences inside the chip processor are described. The polynomials, seeds, the
The number of random patterns and deterministic patterns are inputs to the procedure Gen_Test_Vec. First, the pseudo random patterns are generated by first polynomial and stored in vector $r$ (line1-3). The processor generates software-LFSR based on the primitive polynomial which generates the test vectors. Then the deterministic patterns are generated by the other polynomials and saved in vector $d$ in similar ways (line 4-6). Each of the BM-algorithm based polynomial generates a deterministic test vectors. The vectors $r$ and $d$ are the final outputs of the processor to test the IP cores.

### EXPERIMENTAL RESULTS

The experiments are performed based on some ISCAS89 benchmark circuits assuming the circuits are implemented on a single SoC which needs to be tested by SBST. The necessary polynomials and seeds for each core are compared with the test patterns of [4] in Table 1 row 1 to 4. The patterns used in [4] are deterministic test patterns which guarantee high fault coverage in return for high test data size. Contrastively, the proposed idea use polynomials to reduce the test data size without the loss of the fault coverage.

For the proposed method, 32-bit software-based LFSR is used to generate 100 different pseudo random test patterns. The length of the polynomials for the deterministic patterns are all summed and shown in column 8. The sizes of both polynomials are then combined and shown in column 9. It shows that in all cores, the amount of memory needs to store the test data are reduced.

In row 5, the size of the test programs is compared. The size for [4] is smaller than the proposed one. Though, the total size for both test pattern and test program shown in row 6 shows that the proposed method is about 50% smaller than the size of [4].

### CONCLUSION

In this paper, a new SBST method of low memory size is proposed. The proposed idea uses primitive and BM-algorithm based polynomials. The processor loads the polynomials and generates software-based LFSR which produces test patterns. The experiment results show that the proposed method reduced the size of data necessary to be stored in the on-chip memory.

### REFERENCES


### Table I. Comparison of the Size of the Test Patterns and the Test Programs

<table>
<thead>
<tr>
<th>Core</th>
<th>[4]</th>
<th>Proposed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number</td>
<td>Length</td>
</tr>
<tr>
<td>s38417</td>
<td>387</td>
<td>1,664</td>
</tr>
<tr>
<td>s15850</td>
<td>121</td>
<td>611</td>
</tr>
<tr>
<td>s13207</td>
<td>147</td>
<td>700</td>
</tr>
<tr>
<td>Test Pattern</td>
<td>-</td>
<td>102.6 K</td>
</tr>
<tr>
<td>Test Program</td>
<td>-</td>
<td>0.4 K</td>
</tr>
<tr>
<td>Total</td>
<td>-</td>
<td>103 K</td>
</tr>
</tbody>
</table>