Microprocessors and Microsystems xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

On don't cares in test compression

Jiří Balcárek, Petr Fišer *, Jan Schmidt

Dept. of Digital Design, Faculty of Information Technology, Czech Technical University in Prague, Prague, Czech Republic

ARTICLE INFO

Article history:
Received 3 December 2013
Revised 16 May 2014
Accepted 22 July 2014
Available online xxxx

Keywords:
ATPG
Test don't cares
Satisfiability
Symbolic simulation
Test compression
Embedded cores

ABSTRACT

Both test compression tools and ATPGs directly producing compressed test greatly benefit from don't care values present in the test. Actually, presence of these don't cares is essential for success of the compression. Contemporary ATPGs produce tests having more than 97% of don't cares for large industrial circuits, thus high compression ratios can be expected. However, these don't cares are placed in the test in an "uninformed" way. There are many possibilities of constructing a complete test for a circuit, while the ATPG chooses just one particular, without respect to the subsequent compression process. Therefore, the don't cares cannot be fully exploited. In this paper we show how severe this issue is. A novel ATPG algorithm directly producing compressed test patterns for the RESPIN decompression architecture is presented. Test don't cares are placed in an informed way, so that they are maximally exploited by compression. We compare the results with several ways of uninformed don't care generation to show the benefits of the proposed method. Results for the ISCAS and ITC'99 benchmark circuits are shown and compared to state-of-the-art test compression techniques.

1. Introduction

As the complexity of integrated circuits and systems continually increases, their testing becomes more and more difficult. The test data volume increases with the circuit size, making the test storing and application unfeasibly memory- and time-consuming. Therefore, using some kind of test compression becomes inevitable. According to the ITRS roadmap [1], the required test data volume compression reaches tremendous ratios: 2700-times in 2015 and almost 50,000-times in 2028.

The compression (and subsequent decompression) can be accomplished by several means. The test compression is performed algorithmically, whereas the decompression always involves some additional hardware. Basically, there are three major approaches:

1. A non-compressed test is generated by a *conventional* Automatic Test Pattern Generation tool (ATPG) and then it is algorithmically compressed. The decompression is then performed by a *special dedicated non-intrusive hardware*, usually a kind of FSM. This approach comprises Huffman encoding based algorithms [2], Golomb codes [3], statistical (FDR) codes [4], but also the well-known LFSR reseeding [5,6] to some extent, and the Embedded Deterministic Test (EDT) technique [7], which is now the industrial state-of-the-art.
2. *Generic design-for-testability (DFT) architectures* are used for test decompression, while the test generation process still relies on a *conventional* ATPG. Random access scan [8,9], Illinois scan [10] and RESPIN-based [11–13] architectures belong to this category, together with rather theoretical papers with no particular architecture proposed [14].
3. *Dedicated ATPGs* are used to generate test for *generic or dedicated architectures*. Such an approach theoretically offers the highest possible compression ratio. The algorithm has the highest flexibility, since the compressed test generation is not performed in two subsequent and separated phases. Methods presented in [15–17] are typical representatives. Here the ATPG is constrained or modified, so that the compressed test stream for the target architecture is generated directly. This is also the approach we have adopted in this paper.

As for ATPGs, there are two major baselines: circuit-based ATPGs [18–21] and approaches transforming the ATPG problem to the satisfiability (CNF-SAT) problem [22,23]. Modern ATPGs then combine benefits of both, mostly by introducing structural information to help the SAT-solver compute the solution faster [24–26].

Current ATPGs are able to produce tests containing unspecified values – we call them *test don't cares*. The *test vectors* (*patterns*) that

* Corresponding author. Address: Dept. of Computer Science & Engineering, Czech Technical University in Prague, FIT, Thákurova 9, CZ-160 00 Prague 6, Czech Republic. Tel.: +420 22435 9842; fax: +420 22435 9819.

E-mail addresses: jiri.balcarek@fit.cvut.cz (J. Balcárek), petr.fiser@fit.cvut.cz (P. Fišer), jan.schmidt@fit.cvut.cz (J. Schmidt).

http://dx.doi.org/10.1016/j.micpro.2014.07.006
0141-9331/

could be *incompletely specified* are then referred to as *test cubes*. Don't cares can be efficiently used in the test compression process, since they introduce a kind of flexibility, as any value can be assigned to them. However, there are many ways of forming a (complete) test. The following two aspects must be accounted for:

1. Usually every single fault can be detected by many different test vectors (test patterns). Moreover, a complete set of such vectors usually cannot be described by a single test cube.
2. Test compaction [27,28] performed by ATPG tools, merging test cubes to reduce the test size, can be executed in different ways. Actually, it is an NP-hard process [22] and heuristics are used in practice.

As a result, the don't cares are placed in the test *randomly*, from the point of view of their subsequent usage. Even though the amount of test don't cares typically reaches very high values (more than 97% for industrial designs [7]), still even more flexibility could be exploited by compression. In other words, the test compression process is provided a single set of test cubes only, out of numerous possibilities. There is no guarantee that the compression would not perform better, when given a different test set.

In this paper we show how severe this issue is. We present a SAT-based ATPG producing a compressed test directly. The test cube generation is driven by the compression process, so that most suitable test cubes are used. Naturally, don't cares in test patterns are beneficial for the compression. To obtain these don't cares we propose a method of generating test don't cares in an *informed* way. Then we compare the results with methods where don't cares are obtained in an *uninformed* way, to show the benefits of the former one.

Note that the terms "informed" and "uninformed" used throughout this paper come from the concept of informed and uninformed local search heuristics. The informed methods use some additional information to properly guide the neighborhood exploration. Here, the neighborhood are incompletely specified test patterns, and the heuristic function is the number of faults covered by them.

We extend the SAT-Compress algorithm [16,17] by injection of "don't cares" into test patterns. The SAT-Compress ATPG algorithm generates the compressed test stream by constraining a conventional SAT-based ATPG. Conventional SAT solvers [29,30] used as the vital part of most of SAT-based ATPG tools produce completely specified solutions, where all variables have a specified value in the satisfying solution. There are several ways of introducing don't cares (unspecified variables) into the SAT solution. First, there are SAT-solvers producing incompletely specified solutions directly [31–34]. Here satisfying solutions comprised of minimum literals (minimal models, prime implicants) are generated. However, the optimization criterion is computed over *all* variables, which is unsuitable for our application, where values of only *some* variables are of interest.

Next, optimization version of SAT can be transformed to Integer Linear Programming (ILP) [35]. Here the optimization criterion can be modified for our purposes, so that only some variables are accounted in its computation. In this paper we propose a similar method, particularly the conversion of the SAT problem minimizing the number of specified variables in the satisfying solution to Pseudo-Boolean Optimization (PBO) [36].

Finally, don't cares can be injected into a completely specified vector obtained from a conventional SAT solver [29,30], while the coverage is checked by symbolic fault simulation. When fault simulation is performed, we get additional information on the obtained test cube – its fault coverage [38]. Then we can, e.g., inject don't cares while respecting the fault coverage. This is the informed way of obtaining test don't cares proposed in this paper. We compare this simulation-based method with the uninformed ones and show its benefits in test compression, in terms of both the compressed test stream size and test compression time.

The paper is organized as follows: the target architecture and basic principles of the studied compression algorithms are shown in Section 2. Possibilities of obtaining don't cares in the SAT-Compress process are described in Section 3, Section 4 presents experimental results, to show the role of don't cares in test compression and to illustrate the benefits of informed don't care injection. Section 5 concludes the paper.

2. The decompression architecture and compression algorithms

2.1. The RESPIN architecture

The SAT-Compress algorithm [16,17] and also its enhancements proposed in this paper and [38] are based on the RESPIN architecture [11], which is targeted to System-on-Chip (SoC) designs compliant with the IEEE Std. 1500 [39,40]. Only a very small modification of IEEE Std. 1500 (addition of one multiplexer) can accomplish the test decompression job.

The basic idea of RESPIN is illustrated in Fig. 1. Multiple embedded cores are considered here. To test one core (CUT – Core under Test), the test decompression is performed by another core (ETC – Embedded Tester Core).

RESPIN uses two features of IEEE Std. 1500 – the serial and parallel test access modes. The compressed test bitstream serially enters the ETC, which is configured as a shift-register. Then the decompressed data is applied to the CUT, which is tested in the parallel scan-chain mode.

The ETC is provided with a multiplexer, enabling *rotation* of the pattern. Thereby, if no data come from the external test equipment (ATE), no information on the stored pattern is lost. This opens a simple way to compression: when the deterministic non-compressed test patterns *overlap* when rotated by p bits, each test pattern to be applied to the CUT involves only p bits coming from the ATE. Actually, rotation needs not be used in practice. Typically, one bit comes from ATE in each cycle ($p = 1$), while the remaining bits of the pattern are formed by shifting the previous pattern by one bit. This approach eliminates the need for any control data provided by ATE. For details see [11].

2.2. Patterns overlapping based approaches

An illustrative example of an overlapping-based compression is shown in Fig. 2. Here the non-compressed test length equals to the number of patterns multiplied by the number of CUT scan-chain

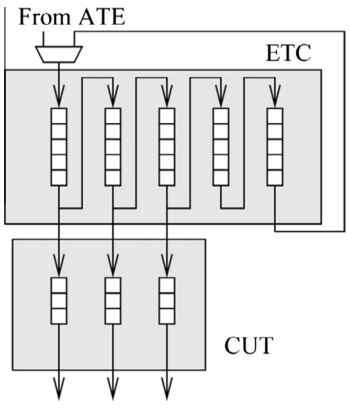

Fig. 1. RESPIN architecture [11].