European Journal of Operational Research 199 (2009) 630–639

Contents lists available at ScienceDirect

European Journal of Operational Research

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

An adaptive heuristic algorithm for VLSI test vectors selection

Walid Ibrahim [a,*], Hesham El-Sayed [a], Amr El-Chouemie [b], Hoda Amer [a]

[a] College of Information Technology, UAE University, Al ain, Abu Dhabi, United Arab Emirates
[b] Hewlett-Packard, Austin, TX, USA

ARTICLE INFO

Article history:
Received 31 March 2007
Accepted 25 March 2008
Available online 13 April 2008

Keywords:
Heuristic algorithms
Integer programming
VLSI
Verification
SCP

ABSTRACT

The increasing complexity of today's system-on-a-chip designs is putting more pressure on the already stressed design verification process. The verification plan must cover several individual cores as well as the overall chip design. Conditions to be verified are identified by the system's architects, the designers, and the verification team. Testing for these conditions is a must for the design to tape out, especially for high priority conditions. A significant bottleneck in the verification process of such designs is that not enough time is usually given to the final coverage phase, which makes computing cycles very precious. Thus, intelligent selection of test vectors that achieve the best coverage using the minimum number of computing cycles is crucial for on time tape out. This paper presents a novel heuristic algorithm for test vectors selection. The algorithm attempts to achieve the best coverage level while minimizing the required number of computing cycles.

1. Introduction

System-on-a-chip (SOC) applications are moving towards larger and more complex designs. This makes the design process a time-consuming process with huge associated development costs. A typical SOC device can have one or several integrated processor cores, a memory subsystem, pipelines and parallel processing blocks, and multiple interfaces. The demand to enhance the performance and shorten the time to market has put severe pressure on designers and architects to get the design of such a complex device correct at the first attempt. However, the increased complexity of these designs has made the verification phase one of the most critical phases of the whole process. Making sure that the design is bug-free is not an easy task given the large number of conditions that has to be verified.

In general, verification is done in two different phases: pre-silicon and post-silicon. The pre-silicon verification has more controllability and more visibility into the design; however, its simulation speed is considerably low. This low simulation speed makes the verification for some conditions impractical. The opposite can be said about the post-silicon verification, which runs at a much higher speed, but lacks the visibility and the controllability of the pre-silicon verification. Projects usually start the verification process as early as possible once they have a functional register transistor logic (RTL) code. The verification efforts will go in parallel with the design activities throughout the lifetime of the project.

Different projects will define their own coverage methodology. In general, three main styles of coverage methodologies are common: line coverage, toggle coverage and functional coverage. In line coverage, the verification process insures that each line in the RTL is executed. This should be the first level of verification for any design. A better level of verification will come from toggle coverage, where each bit in the design is monitored for changing state. The last and the most expensive style is the functional coverage, where different scenarios that traverse the code in specified sequences are identified. In all cases, we considered each data point as a condition. Priorities are given to these conditions based on the risk and the complexity associated with them. In most cases, the RTL code is not written from scratch and a large percentage of the code comes from previous projects. This is very typical when proliferation is done from one design to another. The RTL changes from one proliferation to another can be any where from 10% to 20%. A new RTL code represents a high risk area, since it has not been exercised before. There are also known areas of bugs that will always be considered high risk areas such as memory coherency protocols, memory locking in multi-processing platforms, and self-modifying code.

Coverage for these conditions is necessary for the processor to tape out (the point at which the circuit's description is sent to manufacture), especially these conditions with high risk. Data coverage is collected once the RTL comes to a semi-stable condition. The RTL does not come to a complete freeze state until the end of the project. At that point, coverage is reset and all the coverage data is recollected to insure that a predefined number of conditions have been verified with the final version of the RTL. However, not enough time is normally given to this final coverage phase and

* Corresponding author.
E-mail addresses: walidibr@uaeu.ac.ae (W. Ibrahim), helsayed@uaeu.ac.ae (H. El-Sayed), amr.elchouemie@hp.com (A. El-Chouemie), hamer@uaeu.ac.ae (H. Amer).

computing cycles become very precious. Thus, intelligent selection of test vectors that achieve a target coverage using the minimum amount of computing cycles is essential for on time tape out.

The problem of selecting the best set of test vectors during the verification phase is analogous to the well-known set covering problem (SCP) (refer to Section 2 for a brief overview of SCP). Numerous approaches have been introduced in the literature to solve SCP; however, all the proposed approaches considered the standard full covering problem where each condition is covered at least once Balas and Carrera (1996), Haddadi (1997), Mannino and Sassano (1995). Moreover, the standard SCP schemes cannot handle the case where some conditions are strongly correlated. This research studies the generalization of the SCP to cover either the entire conditions or only a desired subset of them, based on some predefined criteria. In addition, it takes the priority of the conditions and their relationship into consideration. This allows the proposed algorithm to fulfill different testing coverage objectives. For example, the management team can use the proposed algorithm to select the minimum number of test vectors that covers at least 90% of the high risk conditions and 80% of the low risk conditions. Furthermore, the proposed algorithm can also be used to find the best possible coverage level that could be achieved using a specific number of computing cycles.

The rest of the paper is organized as follows: Section 2 reviews the set covering problem and discusses different algorithms in the literature that have been introduced to solve it. The problem definition is represented in Section 3, followed by the proposed algorithm in Section 4. The experimental setup and results are presented in Sections 5 and 6. Finally, the paper concludes in Section 7.

2. Preliminary

The goal of SCP is to find the best subset of the columns of an $m \times n$ zero–one matrix $A = \{a_{ij} \in \{0,1\};\ i = 1,2,\ldots,m;\ j = 1,2,\ldots,n\}$ that covers all rows at a minimum cost, based on a set of costs $\{c_j; j = 1,2,\ldots,n\}$, where a row i is covered by a column j if the entry $a_{ij} = 1$:

$$\text{Minimize:} \quad \sum_{j=1}^{n} c_j x_j \tag{1}$$

$$\text{Subject to:} \quad \sum_{j=1}^{n} a_{ij} x_j \geqslant 1, \quad i \in I = \{1,2,\ldots,m\}, \tag{2}$$

$$x_j = \{0,1\}, \quad j \in J = \{1,2,\ldots,n\}. \tag{3}$$

Constraint Eq. (2) ensures that each row is covered by at least one column, and Eq. (3) requires that whole columns are used.

SCP has a wide number of applications such as construction of optimal logical circuits, resource allocation, aircrew scheduling, pattern recognition, machine learning, and information retrieval. Garey and Johnson (1979) proved that SCP is an NP-hard problem. More recently, De Bontridder et al. (2003) were able to show that the problem is NP-hard even if each column is positive for at most two rows. Integer linear programming (ILP) is the traditional exact method for solving SCP, Balas and Carrera (1996). Exact branch-and-bound algorithms for unweighted SCPs were early used and analyzed by Moret and Shapiro (1985). Later, De Bontridder et al. (2002) introduced new branching rules and various lower bounds and pruning criteria, as well as variable ordering heuristics for the SCP.

Due to the problem hardness, applying exact algorithms to solve large-scale instances of the problem is very time consuming. Therefore, a number of heuristic approaches have been developed to find a close to optimal SCP solution within a reasonable time. Grossman and Wool (1997) performed a comparative experimental study of nine effective approximation algorithms in the field of combinatorial optimization. Their study included algorithms such as the simple and randomized greedy algorithms, a randomized rounding algorithm, and an algorithm based on neural networks. The performance of these algorithms were tested on a large set of benchmark problems, and the results showed that the multi-start randomized greedy algorithm outperformed the other algorithms in almost all of the problem instances. Sen (1993) presented an approach based on simulated annealing, and Ohlsson et al. (2001) developed a field feedback algorithm based on artificial neural network.

Genetic algorithms (GAs) have also been extensively studied to solve this problem. For example, Beasley and Chu (1996) presented a GA-based heuristic with several modifications to the basic genetic procedures. Solar et al. (2002) used a parallel GA to interchange information between parallel searches. Aickelin (2002) solved the problem using a modular three-stage approach; in the first stage, the genetic algorithm finds the 'best' permutation of rows to prepare good parameters for the second stage. Then, a simple heuristic assigns good columns to rows in a given order. Finally, a post-hill-climber algorithm fully optimizes the solutions. Li and Kwan (2004) proposed an evolutionary meta-heuristic algorithm incorporating fuzzy evaluation for some large-scale SCPs originated from the public transport industry.

3. Problem description

During the SOC development process, the design team typically identifies a set of conditions "Θ" that needs to be covered during the verification phase. These conditions are classified into a set of category levels "Λ" based on their risk factor (e.g. high, medium, and low). Each category $k \in \Lambda$ has a set of conditions σ_k and a coverage target γ_k.

In parallel to the design process, the verification team works out their verification plan and creates a test vector suite "ϑ", either manually or automatically with the help of CAD tools. Each test vector $\varphi_j \in \vartheta$ covers a subset of conditions "ζ_j" and needs a number of computing cycles "cc_j" to complete its execution. The test vector suite should cover all the conditions identified by the design team. But since the SOC designs are usually complex, several verification sub-teams on different sites will contribute to the creation of the test vectors. However, due to the lack of management and coordination tools, there is a high possibility that verification engineers on different sites will create test vectors that covers the same conditions more than once. As a result, the created test suite will have a considerable number of redundant test vectors and redundantly covered conditions. We define here the "*Redundant Test Vector*" as the test vector whose entire conditions are covered by other test vectors. It is clear that executing a redundant test vector will unnecessarily consume valuable computing cycles without improving the coverage level.

Furthermore, due to the usual limited time before the final verification phase, it might be infeasible to have a full coverage of the device's conditions before the tape out deadline. Therefore, the management team might set the verification objective to fulfill a predefined partial coverage objective. For example, they may set the device's testing objective to verify only 95% of the high risk, 85% of the medium risk, and 80% of the low risk conditions before taping out the design. Alternatively, the management team can set the verification objective to achieve the best possible coverage using the remaining amount of computing cycles.

In this paper, the problem of selecting the best set of test vectors during the verification phase is modeled as an extended SCP problem. The solution is to find the best subset of the columns (test vectors) of an $m \times n$ zero–one matrix $A = \{a_{ij} \in \{0,1\};\ i = 1,2,\ldots,m;$ $j = 1,2,\ldots,n\}$ that achieves the target coverage of all risk categories